ACTIONSCRIPT® 3.0 개발자 안내서

마지막 업데이트 **2014** 년 **12** 월 **1** 일

법적 고지 사항

법적 고지 사항은 http://help.adobe.com/ko_KR/legalnotices/index.html을 참조하십시오.

목차

22장: Flash Text Engine 사용

23장: Text Layout Framework 사용

24장: 사운드를 사용한 작업

25장: 비디오를 사용한 작업

1장: 날짜 및 시간을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

시간은 소프트웨어 응용 프로그램에서 매우 중요한 요소입니다. ActionScript 3.0에서는 달력 날짜, 시간 및 시간 간격을 관리할 수 있는 강력한 도구를 제공합니다. 두 개의 기본 클래스인 flash.utils 패키지의 Date 클래스 및 새로운 Timer 클래스에서 대부분의 시간 기능을 제공합니다.

날짜와 시간은 ActionScript 프로그램에서 사용되는 일반적인 정보 유형입니다. 예를 들어, 오늘이 무슨 요일인지 알고 싶거나 다른 여러 화면 중 특정 화면에서 사용자가 시간을 얼마나 소비하는지를 측정할 수 있습니다. ActionScript에서는 Date 클래스를 사용하여 날짜 및 시간 정보를 비롯한 해당 시점을 나타낼 수 있습니다. Date 인스턴스 내에는 년, 월, 일, 요일, 시간, 분, 초, 밀리초 및 시간대를 포함하여 개별 날짜 및 시간 단위에 대한 값이 있습니다. ActionScript에 포함된 Timer 클래스를 사용하면 특정 지연 시간 후 또는 반복된 간격으로 작업을 수행하는 등, 고급 기능을 활용할 수 있습니다.

기타 도움말 항목

DATE

flash.utils.Timer

달력 날짜 및 시간 관리

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0의 달력 날짜 및 시간 관리 함수는 최상위 Date 클래스에 집중되어 있습니다. Date 클래스에는 UTC(Coordinated Universal Time) 또는 시간대별 현지 시간에서 날짜 및 시간을 처리할 수 있는 메서드 및 속성이 들어 있습니다. UTC는 기본적으로 그리니치 표준시(GMT)와 동일한 표준 시간 정의입니다.

Date 객체 생성

Flash Player 9 이상, Adobe AIR 1.0 이상

Date 클래스는 모든 기본 클래스 중에서 가장 다양한 생성자 메서드를 제공하는 클래스입니다. 이 메서드는 네 가지 방법으로 호출할 수 있습니다.

첫 번째로, 매개 변수가 없는 경우 Date() 생성자에서 사용자 시간대의 현지 시간을 기준으로 현재 날짜와 시간을 포함하는 Date 객체를 반환합니다. 다음 예제을 참조하십시오.

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

두 번째로, 단일 숫자 매개 변수가 지정된 경우 Date() 생성자는 이를 1970년 1월 1일 이후로 경과된 밀리초로 처리하여 해당하는 Date 객체를 반환합니다. 즉, 전달된 밀리초 값은 표준시(UTC) 기준 1970년 1월 1일 이후로 경과된 밀리초로 처리됩니다. 하지만 Date 객체는 사용자가 UTC 특정 메서드를 사용하여 검색하고 표시하지 않는 한 현지 시간대로 값을 표시합니다. 단일 밀리초 매개 변수를 사용하여 새로운 Date 객체를 생성하는 경우 현지 시간과 표준시(UTC) 사이의 시차를 고려해야 합니다. 다음 명령문은 표준시(UTC) 기준 1970년 1월 1일 자정으로 설정된 Date 객체를 생성합니다.

```
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);
```

세 번째로, 여러 숫자 매개 변수를 Date() 생성자로 전달할 수 있습니다. 생성자는 전달된 매개 변수를 각각 연, 월, 일, 시, 분, 초 및 밀리초로 처리하고 해당하는 Date 객체를 반환합니다. 이러한 입력 매개 변수는 표준시(UTC)가 아니라 현지 시간으로 간주됩니다. 다음 명령문은 현지 시간 2000년 1월 1일 자정으로 설정된 Date 객체를 가져옵니다.

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

네 번째로, 단일 문자열 매개 변수를 Date() 생성자로 전달할 수 있습니다. 생성자는 해당 문자열을 날짜 및 시간 구성 요소로 파싱한 다음 해당하는 Date 객체를 반환합니다. 이 방법을 사용하는 경우 Date() 생성자를 try..catch 블록에 포함하여 모든 파싱 오류를 트랩하는 것이 좋습니다. Date() 생성자는 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에 나열된 여러 문자열 형식을 사용합니다. 다음 명령문은 문자열 값을 사용하여 새 Date 객체를 초기화합니다.

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

Date() 생성자가 문자열 매개 변수를 성공적으로 파싱하지 못하는 경우에도 예외가 발생하지 않습니다. 하지만 결과 Date 객체에 잘못된 날짜 값이 포함됩니다.

시간 단위 값 가져오기

Flash Player 9 이상, Adobe AIR 1.0 이상

Date 클래스의 속성 또는 메서드를 사용하여 Date 객체 내의 여러 시간 단위 값을 추출할 수 있습니다. 다음 각 속성을 사용하여 Date 객체에서 시간 단위 값을 가져올 수 있습니다.

- fullYear 속성
- month 속성(1월의 0에서 12월의 11까지 숫자 형식 사용)
- date 속성(1부터 31까지 범위의 달력 날짜 사용)
- day 속성(일요일에 0 등 요일에 숫자 형식 사용)
- hours 속성(0 ~ 23 사용)
- minutes 속성
- seconds 속성
- milliseconds 속성

  실제로 Date 클래스를 사용하면 여러 가지 방법으로 이 값을 가져올 수 있습니다. 예를 들어, 다음 네 가지 방법으로 Date 객체의 월 값을 가져올 수 있습니다.

- month 속성
- getMonth() 메서드
- monthUTC 속성
- getMonthUTC() 메서드

  기본적으로 네 방법 모두 효율적이기 때문에 응용 프로그램에 따라 가장 적합한 방법을 사용하면 됩니다.

  나열된 속성은 모두 총 날짜 값의 구성 요소를 나타냅니다. 예를 들어, milliseconds 속성은 999보다 클 수 없으며, 이는 값이 1000에 도달하면 초 값이 1 증가하고 milliseconds 속성이 0으로 재설정되기 때문입니다.

  표준시 기준 1970년 1월 1일 이후로 경과된 밀리초로 Date 객체 값을 얻으려면 getTime() 메서드를 사용할 수 있습니다. 이에 대응되는 setTime() 메서드를 사용하면 표준시 기준 1970년 1월 1일 이후로 경과된 밀리초를 사용하여 기존 Date 객체를 변경할 수 있습니다.

날짜 및 시간 계산 수행

Flash Player 9 이상, Adobe AIR 1.0 이상

Date 클래스를 사용하여 날짜 및 시간에 대해 더하기 및 빼기를 수행할 수 있습니다. 날짜 값은 내부적으로 밀리초로 단위로 보관되므로 Date 객체에서 날짜 값을 더하거나 빼기 전에 값을 밀리초로 변환해야 합니다.

응용 프로그램에서 날짜 및 시간 계산을 많이 수행하는 경우에는 다음과 같이 상수를 생성하여 밀리초로 환산된 공통 시간 단위 값을 보관하는 것이 유용합니다.

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

표준 시간 단위를 사용하여 날짜 계산을 편리하게 수행할 수 있습니다. 다음 코드는 getTime() 및 setTime() 메서드를 사용하여 날짜 값을 현재 시간부터 한 시간으로 설정합니다.

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

날짜 값을 설정하는 또 다른 방법은 단일 밀리초 매개 변수를 사용하여 새 Date 객체를 생성하는 것입니다. 예를 들어, 다음 코드는 한 날짜에 30일을 더하여 다른 날짜를 계산합니다.

```
// 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));
```

그런 다음 millisecondsPerDay 상수에 30을 곱해 30일의 시간을 나타내고 그 결과를 invoiceDate 값에 더하여 dueDate 값을 설정합니다.

시간대 간 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

날짜 및 시간 계산은 날짜를 한 시간대에서 다른 시간대로 변환하고 싶은 경우에도 유용합니다. getTimezoneOffset() 메서드도 마찬가지이며, 이 메서드는 Date 객체의 시간대와 표준시의 차이를 분 값으로 반환합니다. 분 값으로 반환하는 이유는 일부 시간대가 시간 간격으로 시차가 설정되지 않고 옆 시간대에서 30분 차이가 나기도 하기 때문입니다.

다음 예제에서는 시간대 오프셋을 사용하여 현지 시간에서 표준시로 날짜를 변환합니다. 다음과 같이 먼저 밀리초 단위로 시간대 값을 계산한 다음 해당 값만큼 Date 값을 조정합니다.

```
// 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);
```

시간 간격 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe Flash CS4 Professional을 사용하여 응용 프로그램을 개발할 때는 응용 프로그램의 일정한 프레임 단위 진행을 가능하게 하는 타임라인을 사용할 수 있습니다. 하지만 ActionScript 프로젝트만 진행하는 경우에는 다른 시간 메커니즘을 사용해야 합니다.

루프와 타이머

Flash Player 9 이상, Adobe AIR 1.0 이상

일부 프로그래밍 언어에서는 for 또는 do..while과 같은 루프 문을 사용하여 고유한 시간 체계를 고안해야 합니다.

일반적으로 루프 명령문은 로컬 시스템 성능에 따라 실행 속도가 달라지므로, 이에 따라 일부 시스템에서는 응용 프로그램이 빠르게 실행되지만 다른 시스템에서는 느리게 실행될 수 있습니다. 응용 프로그램에 일정한 시간 간격이 필요하면 실제 달력이나 시계를 연결해야 합니다. 게임, 애니메이션, 실시간 컨트롤러 등의 많은 응용 프로그램에는 시스템 모두에 일관되게 적용되는 규칙적인 시간 기반 티킹(ticking) 메커니즘이 필요합니다.

ActionScript 3.0 Timer 클래스는 매우 강력한 솔루션을 제공합니다. ActionScript 3.0 이벤트 모델을 사용하여, Timer 클래스는 지정된 시간 간격에 도달할 때마다 타이머 이벤트를 전달합니다.

Timer 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서 시간 함수를 처리하는 좋은 방법은 지정된 시간 간격에 도달할 때마다 이벤트를 전달하는 Timer 클래스(flash.utils.Timer)를 사용하는 것입니다.

타이머를 시작하려면 먼저 Timer 클래스의 인스턴스를 생성하여 타이머 이벤트 생성 빈도 및 중단 시까지의 생성 횟수를 지정합니다.

예를 들어, 다음 코드는 매 초마다 이벤트를 전달하고 이 작업을 60초 동안 계속하는 Timer 인스턴스를 생성합니다.

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

Timer 객체는 지정된 간격에 도달할 때마다 TimerEvent 객체를 전달합니다. TimerEvent 객체의 이벤트 유형은 timer(TimerEvent.TIMER 상수로 정의됨)입니다. TimerEvent 객체에는 표준 Event 객체와 동일한 속성이 들어 있습니다.

Timer 인스턴스가 고정된 간격 수로 설정된 경우에는 최종 간격에 도달할 때 timerComplete 이벤트(TimerEvent.TIMER_COMPLETE 상수로 정의됨)도 전달됩니다.

다음은 Timer 클래스의 작동을 보여주는 샘플 응용 프로그램입니다.

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

ShortTimer 클래스를 생성하면 5초 동안 1초에 한 번 작동하는 Timer 인스턴스가 생성됩니다. 그런 다음 타이머에 두 개의 리스너를 추가하여 하나는 각 틱(tick)을 수신하고 다른 하나는 timerComplete 이벤트를 수신하도록 합니다.

그러면 타이머 티킹이 시작되어 그 시점 이후부터 onTick() 메서드가 1초 간격으로 실행됩니다.

onTick() 메서드는 단순히 현재의 틱 카운트를 표시합니다. 5초가 지난 후에는 onTimerComplete() 메서드가 실행되어 시간이 종료되었음을 알립니다.

이 샘플을 실행하면 다음과 같은 행이 콘솔이나 추적 윈도우에 1초에 한 행씩 나타나는 것을 볼 수 있습니다.

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

flash.utils 패키지의 시간 함수

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 ActionScript 2.0에 사용된 시간 함수와 유사한 여러 함수가 포함되어 있습니다. 이러한 함수는 flash.utils 패키지의 패키지 수준 함수로 제공되며 ActionScript 2.0에서와 동일한 방식으로 작동됩니다.

함수	설명
clearInterval(id:uint):void	지정된 setInterval() 호출을 취소합니다.
clearTimeout(id:uint):void	지정된 setTimeout() 호출을 취소합니다.
getTimer():int	Adobe® Flash® Player 또는 Adobe® AIR™를 초기화한 이후 경과된 밀리초 수를 반환합니다.
setInterval(closure:Function, delay:Number, ... arguments):uint	지정된 밀리초 단위 간격으로 함수를 실행합니다.
setTimeout(closure:Function, delay:Number, ... arguments):uint	밀리초 단위로 지정된 지연 시간 후 지정된 함수를 실행합니다.

이러한 함수는 이전 버전과의 역호환성을 위해 ActionScript 3.0에 남아 있습니다. 하지만 새 ActionScript 3.0 응용 프로그램에는 이러한 함수를 사용하지 않는 것이 좋습니다. 일반적으로 응용 프로그램에 Timer 클래스를 사용하는 것이 보다 쉽고 효율적입니다.

날짜 및 시간 예제: 간단한 아날로그 시계

Flash Player 9 이상, Adobe AIR 1.0 이상

간단한 아날로그 시계 예제를 통해 날짜와 시간이라는 두 개념에 대해 설명합니다.

- 현재 날짜 및 시간을 가져와서 시간, 분, 초 값으로 추출
- 타이머를 사용하여 응용 프로그램 속도 설정

  이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. SimpleClock 응용 프로그램 파일은 Samples/SimpleClock 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
SimpleClockApp.mxml 또는 SimpleClockApp.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/simpleclock/SimpleClock.as	기본 응용 프로그램 파일입니다.
com/example/programmingas3/simpleclock/AnalogClockFace.as	시간에 따라 둥근 시계 문자판과 시침, 분침, 초침을 그립니다.

SimpleClock 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

시계 예제는 간단한 프로그램이지만, 아주 간단한 응용 프로그램이라도 나중에 쉽게 확장할 수 있도록 잘 구성하는 것이 좋습니다. 이 예제에서 SimpleClock 응용 프로그램은 SimpleClock 클래스를 사용하여 시작 및 시간 맞추기 작업을 처리한 다음 AnalogClockFace라는 다른 클래스를 사용하여 실제로 시간을 표시합니다.

다음은 SimpleClock 클래스를 정의하고 초기화하는 코드입니다. Flash 버전에서 SimpleClock을 실행하면 Sprite 클래스가 확장됩니다.

```
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;
```

이 클래스에는 다음과 같은 중요한 두 가지 속성이 들어 있습니다.

- face 속성(AnalogClockFace 클래스의 인스턴스임)
- ticker 속성(Timer 클래스의 인스턴스임)

  SimpleClock 클래스는 기본 생성자를 사용합니다. initClock() 메서드는 실제 설정 작업을 처리하여 시계 문자판을 만들고 Timer 인스턴스 티킹을 시작합니다.

시계 문자판 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

SimpleClock 코드의 다음 행에서는 시간을 표시하는 데 사용하는 시계 문자판을 만듭니다.

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

문자판의 크기는 initClock() 메서드로 전달될 수 있습니다. 전달된 faceSize 값이 없는 경우에는 기본 크기인 200 픽셀이 사용됩니다.

그런 다음 응용 프로그램은 해당 문자판을 초기화한 후 DisplayObjectContainer 클래스에서 상속된 addChild() 메서드를 사용하여 표시 목록에 추가합니다. 그리고 나서 AnalogClockFace.draw() 메서드를 호출하여 시계 문자판을 현재 시간과 함께 한 번 표시합니다.

타이머 시작

Flash Player 9 이상, Adobe AIR 1.0 이상

시계 문자판을 만든 후에는 initClock() 메서드에서 타이머를 설정합니다.

```
        // 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();
```

이 메서드는 먼저 1초(1000밀리초)에 한 번 이벤트를 전달하는 Timer 인스턴스를 인스턴스화합니다. 다른 repeatCount 매개 변수가 Timer() 생성자로 전달되지 않으므로 타이머는 무한정으로 계속 반복됩니다.

timer 이벤트를 수신하면 SimpleClock.onTick() 메서드가 1초에 한 번씩 실행됩니다.

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

AnalogClockFace.draw() 메서드는 간단히 시계 문자판과 시계 바늘을 그립니다.

현재 시간 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

AnalogClockFace 클래스 코드의 대부분은 시계 문자판의 표시 요소를 설정하는 것과 관련되어 있습니다. AnalogClockFace를 시작하면 둥근 외곽선을 그리고 각 시간 표시 부분에 숫자 텍스트 레이블을 놓은 다음 각각 시계의 시침, 분침, 초침에 해당하는 세 개의 Shape 객체를 생성합니다.

SimpleClock 응용 프로그램을 실행하면 다음과 같이 매 초마다 AnalogClockFace.draw() 메서드가 호출됩니다.

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

이 메서드는 현재 시간을 변수로 저장하므로 시계 바늘을 그리는 도중에는 시간을 변경할 수 없습니다. 그런 다음 showTime() 메서드를 호출하여 다음과 같이 시계 바늘을 표시합니다.

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

이 메서드는 먼저 현재 시간의 시, 분, 초 값을 추출합니다. 그런 다음 이 값을 사용하여 각 시계 바늘의 각도를 계산합니다. 초침은 60초에 한 번 회전하기 때문에 각 초마다 6도(360/60)씩 움직이게 됩니다. 분침은 매 분마다 동일한 각도씩 움직입니다.

시침도 매 분 업데이트되어 분침이 이동할 때마다 일정 정도 진행됩니다. 시침은 매 시간마다 30도(360/12)씩 움직이고 매 분마다 0.5도(30도/60분)씩 움직입니다.

2장: 문자열을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

String 클래스에는 텍스트 문자열을 사용하여 작업할 수 있도록 하는 메서드가 포함되어 있습니다. 문자열은 많은 객체를 사용하는 작업에 중요합니다. 이 장에서 설명된 메서드는 TextField, StaticText, XML, ContextMenu, FileReference와 같은 객체에 사용된 문자열을 사용하여 작업할 때 유용합니다.

문자열은 문자의 시퀀스입니다. ActionScript 3.0에서는 ASCII 및 유니코드 문자가 지원됩니다.

기타 도움말 항목

String

RegExp

parseFloat()

parseInt()

문자열의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

프로그래밍 용어에서 문자열은 텍스트 값, 즉 단일 값으로 묶인 일련의 문자, 숫자 또는 기타 문자를 뜻합니다. 예를 들어 다음 코드 행은 데이터 유형이 String인 변수를 만들고 리터럴 문자열 값을 해당 변수에 지정합니다.

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

이 예제에서 보듯이 ActionScript에서는 큰따옴표나 작은따옴표로 텍스트를 묶어서 문자열 값을 나타낼 수 있습니다. 다음은 몇 가지 문자열의 예입니다.

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

ActionScript에서 텍스트를 조작하는 경우 문자열 값으로 작업하는 것입니다. ActionScript String 클래스는 텍스트 값을 다룰 때 사용할 수 있는 데이터 유형입니다. 문자열 인스턴스는 다른 많은 ActionScript 클래스에서 속성, 메서드 매개 변수 등에 자주 사용됩니다.

중요한 개념 및 용어

다음 참조 목록에는 이 장에 사용되는 문자열 관련 중요 용어가 포함되어 있습니다.

ASCII 컴퓨터 프로그램에서 텍스트 문자 및 심볼을 나타내기 위한 시스템입니다. ASCII 시스템은 26자의 영어 알파벳과 제한된 추가 문자 집합을 지원합니다.

문자 텍스트 데이터의 최소 단위입니다(단일 문자 또는 심볼).

결합 문자열 뒤에 다른 문자열을 추가하여 새로운 문자열 값을 만들 때와 같이 여러 문자열 값을 결합하는 것을 뜻합니다.

빈 문자열 텍스트가 없거나 공백 또는 기타 문자가 포함된 문자열이며 ""로 작성됩니다. 빈 문자열은 null 값을 가진 String 변수와 다릅니다. null 값의 String 변수는 변수에 지정된 문자열 인스턴스가 없는 변수지만, 빈 문자열은 값에 문자가 포함되지 않은 인스턴스를 가집니다.

String 텍스트 값입니다(일련의 문자).

문자열 리터럴(또는 "리터럴 문자열") 코드에서 명시적으로 작성된 문자열로, 텍스트 값을 큰따옴표 또는 작은따옴표로 묶어 표기합니다.

하위 문자열 다른 문자열의 일부가 되는 문자열입니다.

유니코드 컴퓨터 프로그램에서 텍스트 문자 및 심볼을 나타내기 위한 표준 시스템입니다. 유니코드 시스템을 사용하면 모든 쓰기 시스템에서 어느 문자든지 사용할 수 있습니다.

문자열 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

String 클래스는 ActionScript 3.0에서 문자열(텍스트) 데이터를 나타내는 데 사용됩니다. ActionScript 문자열에서는 ASCII 및 유니코드 문자가 모두 지원됩니다. 문자열을 만드는 가장 간단한 방법은 문자열 리터럴을 사용하는 것입니다. 문자열 리터럴을 선언하려면 곧은 큰따옴표(")나 작은따옴표(') 문자를 사용합니다. 예를 들어 다음 두 문자열은 동일합니다.

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

다음과 같이 new 연산자를 사용하여 문자열을 선언할 수도 있습니다.

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

다음 두 문자열은 동일합니다.

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

작은따옴표(') 구분 기호로 정의된 문자열 리터럴에 작은따옴표(')를 사용하려면 이스케이프 문자인 백슬래시(\)를 사용합니다. 또한 큰따옴표(") 구분 기호로 정의된 문자열 리터럴에 큰따옴표(")를 사용할 때도 이스케이프 문자인 백슬래시(\)를 사용합니다. 다음 두 문자열은 동일합니다.

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

다음과 같이 문자열 리터럴에 있는 작은따옴표나 큰따옴표를 기준으로 작은따옴표나 큰따옴표를 선택하여 사용할 수 있습니다.

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

ActionScript에서는 곧은 작은따옴표(')와 왼쪽이나 오른쪽 작은따옴표(' 또는 ')가 구분됩니다. 큰따옴표의 경우에도 마찬가지입니다. 문자열 리터럴을 구분하려면 곧은 따옴표를 사용합니다. 다른 소스의 텍스트를 ActionScript로 붙여 넣을 경우 정확한 문자를 사용해야 합니다.

다음 표와 같이 이스케이프 문자인 백슬래시(\)를 사용하여 문자열 리터럴에 다른 문자를 정의할 수 있습니다.

이스케이프 시퀀스	**문자**
\b	백스페이스
\f	용지 공급
\n	개행
\r	캐리지 리턴
\t	탭
\u*nnnn*	16진수 **nnnn**으로 지정된 문자 코드(예: \u263a)가 있는 유니코드 문자는 스마일리 문자입니다.

이스케이프 시퀀스	문자
\\x*nn*	16진수 **nn**으로 지정된 문자 코드가 있는 ASCII 문자
\'	작은따옴표
\"	큰따옴표
\\	단일 백슬래시 문자

length 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 문자열에는 length 속성이 있습니다. 이 속성은 문자열에 있는 문자의 수와 같습니다.

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

빈 문자열 및 null 문자열의 길이는 다음 예제와 같이 0입니다.

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

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

문자열 내의 문자 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

문자열의 모든 문자에는 인덱스 위치(정수)가 있습니다. 첫 번째 문자의 인덱스 위치는 0입니다. 예를 들어 다음 문자열에서 문자 y는 위치 0에 있고 문자 w는 위치 5에 있습니다.

```
"yellow"
```

다음 예제와 같이 charAt() 메서드 및 charCodeAt() 메서드를 사용하여 문자열 내의 다양한 위치에서 개별 문자를 검사할 수 있습니다.

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

이 코드를 실행하면 다음과 같이 출력됩니다.

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

다음 예제와 같이 문자 코드를 사용하여 fromCharCode() 메서드를 통해 문자열을 정의할 수도 있습니다.

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

문자열 비교

Flash Player 9 이상, Adobe AIR 1.0 이상

<, <=, !=, ==, => 및 > 연산자를 사용하여 문자열을 비교할 수 있습니다. 이러한 연산자는 다음 예제와 같이 if 및 while과 같은 조건문과 함께 사용할 수 있습니다.

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

문자열에서 이러한 연산자를 사용할 경우 ActionScript에서는 다음과 같이 문자열에 있는 각 문자의 문자 코드 값을 고려하여 왼쪽부터 오른쪽 순으로 문자를 비교합니다.

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

다음 예제와 같이 문자열을 서로 비교하고 문자열을 다른 유형의 객체와 비교하려면 == 및 != 연산자를 사용합니다.

```
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
```

다른 객체의 문자열 표현 가져오기

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 종류의 객체에 대한 문자열 표현을 가져올 수 있습니다. 모든 객체에는 이 용도로 사용되는 toString() 메서드가 있습니다.

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

String 객체와 문자열이 아닌 객체를 결합하기 위해 + 연결 연산자를 사용할 때 toString() 메서드를 사용하지 않아도 됩니다. 연결에 대한 자세한 내용은 다음 단원을 참조하십시오.

String() 전역 함수는 지정된 객체에 대해 toString() 메서드를 호출하여 반환된 값과 동일한 값을 반환합니다.

문자열 연결

Flash Player 9 이상, Adobe AIR 1.0 이상

문자열 연결은 두 개의 문자열을 가지고 순차적으로 하나로 결합하는 것을 의미합니다. 예를 들어 + 연산자를 사용하여 두 문자열을 연결할 수 있습니다.

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

+= 연산자를 사용해도 다음 예제와 같이 동일한 결과를 생성할 수 있습니다.

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

또한 String 클래스에는 concat() 메서드가 포함되며, 다음과 같이 사용될 수 있습니다.

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

String 객체와 문자열이 아닌 객체에 + 연산자(또는 += 연산자)를 사용할 경우 ActionScript에서는 다음 예제와 같이 표현식을 평가하기 위해 문자열이 아닌 객체를 String 객체로 자동 변환합니다.

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

하지만 다음 예제와 같이 그룹 괄호를 사용하여 + 연산자에 대한 컨텍스트를 제공할 수 있습니다.

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

문자열의 패턴 및 하위 문자열 찾기

Flash Player 9 이상, Adobe AIR 1.0 이상

하위 문자열은 문자열 내의 연속 문자입니다. 예를 들어 문자열 "abc"에는 "", "a", "ab", "abc", "b", "bc", "c"와 같은 하위 문자열이 있습니다. ActionScript 메서드를 사용하여 문자열의 하위 문자열을 찾을 수 있습니다.

패턴은 ActionScript에서 문자열이나 일반 표현식으로 정의됩니다. 예를 들어 다음 일반 표현식은 글자 A, B, C 다음에 숫자가 오는 특정 패턴을 정의합니다(슬래시는 일반 표현식 구분 기호임).

```
/ABC\d/
```

ActionScript에는 문자열의 패턴을 찾고 찾은 항목을 대체 하위 문자열로 바꾸는 메서드가 포함되어 있습니다. 이러한 메서드에 대해서는 다음 단원에서 설명합니다.

일반 표현식은 복잡한 패턴을 정의할 수 있습니다. 자세한 내용은 67페이지의 “일반 표현식 사용”을 참조하십시오.

문자 위치로 하위 문자열 찾기

Flash Player 9 이상, Adobe AIR 1.0 이상

substr() 및 substring() 메서드는 서로 비슷합니다. 두 메서드 모두 문자열의 하위 문자열을 반환합니다. 또한 두 개의 매개 변수를 사용합니다. 두 메서드에서 첫 번째 매개 변수는 지정된 문자열에서 시작 문자의 위치입니다. 그러나 substr() 메서드에서 두 번째 매개 변수는 반환할 하위 문자열의 **길이**인 반면 substring() 메서드에서 두 번째 매개 변수는 반환된 문자열에 포함되지 않는 하위 문자열의 끝에 있는 문자의 위치입니다. 다음 예제에서는 두 메서드 간의 차이점을 보여 줍니다.

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

slice() 메서드는 substring() 메서드와 작동 방식이 유사합니다. 두 개의 음수가 아닌 정수를 매개 변수로 지정하면 두 메서드는 똑같이 작동합니다. 하지만 slice() 메서드는 매개 변수로 음의 정수를 사용할 수 있으며, 이 경우 문자 위치는 다음 예제에서와 같이 문자열 끝에서 가지고 옵니다.

```
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
```

음수가 아닌 정수와 음수를 slice() 메서드의 매개 변수로 결합할 수 있습니다.

일치하는 하위 문자열의 문자 위치 찾기

Flash Player 9 이상, Adobe AIR 1.0 이상

indexOf() 및 lastIndexOf() 메서드를 사용하여 다음 예제와 같이 한 문자열 내에서 일치하는 하위 문자열을 찾을 수 있습니다.

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

indexOf() 메서드는 대/소문자를 구분합니다.

두 번째 매개 변수를 지정하여 다음과 같이 검색을 시작할 문자열의 인덱스 위치를 나타낼 수 있습니다.

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

lastIndexOf() 메서드는 문자열에서 마지막 하위 문자열을 찾습니다.

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

lastIndexOf() 메서드로 두 번째 매개 변수를 포함하면 문자열의 해당 인덱스 위치부터 역방향(오른쪽에서 왼쪽)으로 검색이 실행됩니다.

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

구분 기호로 분리된 하위 문자열 배열 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

split() 메서드를 사용하여 하위 문자열의 배열을 만들 수 있습니다. 하위 문자열은 구분 기호를 기준으로 분리됩니다. 예를 들어 쉼표로 구분되거나 탭으로 구분된 문자열을 여러 문자열로 분리할 수 있습니다.

다음 예제에서는 구분 기호로 앰퍼샌드(&) 문자를 사용하여 배열을 하위 문자열로 분리하는 방법을 보여 줍니다.

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

split() 메서드의 두 번째 매개 변수는 선택 사항이며 반환되는 배열의 최대 크기를 정의합니다.

구분 기호 문자로 일반 표현식을 사용할 수도 있습니다.

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

자세한 내용은 67페이지의 “일반 표현식 사용” 및 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

문자열의 패턴 찾기 및 하위 문자열 바꾸기

Flash Player 9 이상, Adobe AIR 1.0 이상

String 클래스에는 문자열의 패턴을 사용하여 작업하는 다음과 같은 메서드가 포함됩니다.

- match() 및 search() 메서드를 사용하여 패턴과 일치하는 하위 문자열을 찾습니다.
- replace() 메서드를 사용하여 패턴과 일치하는 하위 문자열을 찾고 지정한 하위 문자열로 바꿉니다.

이러한 메서드에 대해서는 다음 단원에서 설명합니다.

문자열이나 일반 표현식을 사용하여 이러한 메서드에 사용할 패턴을 정의할 수 있습니다. 일반 표현식에 대한 자세한 내용은 67페이지의 “일반 표현식 사용”을 참조하십시오.

일치하는 하위 문자열 찾기

search() 메서드는 이 예제와 같이 지정된 패턴과 일치하는 첫 번째 하위 문자열의 인덱스 위치를 반환합니다.

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

일반 표현식을 사용하여 이 예제와 같이 일치하는 패턴을 정의할 수도 있습니다.

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

문자열의 첫 번째 문자는 인덱스 위치 0이므로 trace() 메서드의 출력은 0이며, i 플래그가 일반 표현식에 설정되어 있으므로 검색 시 대/소문자를 구분하지 않습니다.

search() 메서드는 g(global) 플래그가 일반 표현식에 설정되어 있어도 일치하는 한 항목만 찾으며 그 시작 인덱스 위치를 반환합니다.

다음 예제에서는 큰따옴표로 묶인 문자열과 일치하는 보다 복잡한 일반 표현식을 보여 줍니다.

```
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.)
```

match() 메서드는 이와 유사하게 작동하며, 일치하는 하위 문자열을 검색합니다. 하지만 다음 예제와 같이 일반 표현식 패턴에서 전역 플래그를 사용할 경우 match() 메서드는 일치하는 하위 문자열의 배열을 반환합니다.

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

results 배열은 다음과 같이 설정됩니다.

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

일반 표현식에 대한 자세한 내용은 67페이지의 “일반 표현식 사용”을 참조하십시오.

일치하는 하위 문자열 바꾸기

replace() 메서드를 사용하여 다음 예제와 같이 문자열에서 지정된 패턴을 검색하고 지정된 대체 문자열로 일치 항목을 바꿀 수 있습니다.

```
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.
```

이 예제에서는 i(ignoreCase) 플래그가 일반 표현식에 설정되어 있으므로 일치 문자열이 대/소문자를 구분하지 않으며, g(global) 플래그가 설정되어 있으므로 여러 일치 항목이 모두 바뀝니다. 자세한 내용은 67페이지의 “일반 표현식 사용”을 참조하십시오.

대체 문자열에서 다음 $ 대체 코드를 사용할 수 있습니다. 다음 표에 표시된 대체 텍스트는 $대체 코드 위치로 삽입됩니다.

$ 코드	**대체 텍스트**
$$	$
$&	일치하는 하위 문자열
$`	일치하는 하위 문자열 앞에 오는 문자열의 일부. 이 코드에는 곧은 작은따옴표(')나 왼쪽으로 굽은 작은따옴표(')가 아닌 왼쪽으로 기울고 곧은 작은따옴표 문자(`)가 사용됩니다.
$'	일치하는 하위 문자열 다음에 오는 문자열의 일부. 이 코드에는 곧은 작은따옴표(')가 사용됩니다.
$n	괄호로 둘러싼 일치 그룹 중 **n**번째로 캡처된 항목. 여기에서 n은 1에서 9 사이의 한 자리 숫자이고 $n 다음에는 10진수 숫자가 오지 않습니다.
$nn	괄호로 둘러싼 일치 그룹 중 **nn**번째로 캡처한 그룹. 여기에서 **nn**은 01에서 99 사이의 두 자리 10진수입니다. **nn**번째 캡처 항목이 정의되어 있지 않으면 빈 문자열이 대체 텍스트가 됩니다.

예를 들어 다음 예제에서는 일치 그룹 중 첫 번째 및 두 번째 캡처 항목을 나타내는 $2 및 $1 대체 코드를 사용하는 방법을 보여 줍니다.

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

함수를 replace() 메서드의 두 번째 매개 변수로 사용할 수도 있습니다. 이 경우 일치하는 텍스트는 반환된 함수 값으로 바뀝니다.

```
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;
}
```

함수를 replace() 메서드의 두 번째 매개 변수로 사용할 때 다음 인수가 함수로 전달됩니다.

- 문자열의 일치 부분
- 괄호로 둘러싼 일치 그룹 중 캡처할 항목. 이렇게 전달되는 인수의 수는 괄호로 둘러싼 일치 항목의 수에 따라 달라집니다. 괄호로 둘러싼 일치 항목의 수를 확인하려면 함수 코드 내에서 arguments.length - 3을 확인합니다.
- 문자열에서 검색을 시작할 인덱스 위치
- 전체 문자열

대/소문자 간 문자열 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 예제와 같이 toLowerCase() 메서드 및 toUpperCase() 메서드는 문자열에 있는 알파벳 문자를 각각 소문자와 대문자로 변환합니다.

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

이 메서드를 실행하더라도 소스 문자열은 변경되지 않습니다. 소스 문자열을 변환하려면 다음 코드를 사용합니다.

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

이러한 메서드는 단순히 a-z 및 A-Z 문자뿐 아니라 확장 문자에도 적용됩니다.

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

문자열 예제: ASCII Art

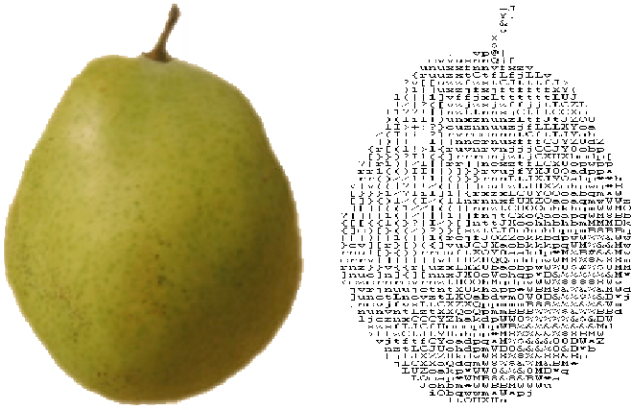
Flash Player 9 이상, Adobe AIR 1.0 이상

ASCII Art 예제에서는 ActionScript 3.0에서 String 클래스를 사용한 작업의 다양한 기능을 보여 줍니다. 다음은 그 기능 중 일부입니다.

- String 클래스의 split() 메서드는 문자 구분 문자열에서 값을 추출(탭 구분 텍스트 파일의 이미지 정보)하는 데 사용됩니다.
- split(), 연결 및 substring(), substr()을 사용한 문자열 일부 추출 등을 비롯한 여러 문자열 조작 기술은 이미지 제목에서 각 단어의 첫 번째 글자를 대문자화하는 데 사용됩니다.

- getCharAt() 메서드는 문자열에서 단일 문자를 가져오는 데 사용됩니다(회색 음영 비트맵 값에 해당하는 ASCII 문자를 확인하는 데 사용).
- 문자열 연결은 한 번에 하나의 문자씩 이미지의 ASCII Art 표현을 작성하는 데 사용됩니다.

ASCII Art는 Courier New 문자와 같은 고정폭 글꼴 문자로 구성된 격자를 사용하여 이미지를 표시하는 것으로, 이미지를 텍스트로 표현한 것을 나타냅니다. 다음 이미지는 응용 프로그램에서 생성된 ASCII Art의 예입니다.



오른쪽 이미지는 그래픽의 ASCII Art 버전입니다.

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. ASCIIArt 응용 프로그램 파일은 Samples/AsciiArt 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
AsciiArtApp.mxml 또는 AsciiArtApp.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/asciiArt/AsciiArtBuilder.as	텍스트 파일의 이미지 메타데이터 추출, 이미지 로딩, 이미지-텍스트 변환 프로세스 관리 등을 비롯한 응용 프로그램의 주요 기능을 제공하는 클래스입니다.
com/example/programmingas3/asciiArt/BitmapToAsciiConverter.as	이미지 데이터를 문자열 버전으로 변환하기 위한 parseBitmapData() 메서드를 제공하는 클래스입니다.
com/example/programmingas3/asciiArt/Image.as	로드된 비트맵 이미지를 나타내는 클래스입니다.
com/example/programmingas3/asciiArt/ImageInfo.as	제목, 이미지 파일 URL 등과 같은 ASCII Art 이미지에 대한 메타데이터를 나타내는 클래스입니다.
image/	응용 프로그램에서 사용된 이미지가 포함되어 있는 폴더입니다.
txt/ImageData.txt	응용 프로그램에 로드되는 이미지에 대한 정보가 포함되어 있는, 탭으로 구분된 텍스트 파일입니다.

탭 구분 값 추출

Flash Player 9 이상, Adobe AIR 1.0 이상

이 예제에서는 응용 프로그램 자체와 응용 프로그램 데이터를 별도로 저장하는 일반적인 규칙이 사용되어, 데이터가 변경된 경우에도(예: 다른 이미지가 추가되거나 이미지 제목이 변경된 경우) SWF 파일을 다시 만들 필요가 없습니다. 이 경우 이미지 제목, 실제 이미지 파일의 URL, 이미지를 조작하는 데 사용된 일부 값 등을 비롯한 이미지 메타데이터는 텍스트 파일에 저장됩니다(프로젝트의 txt/ImageData.txt 파일). 이 텍스트 파일의 내용은 다음과 같습니다.

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

이 파일은 특정 탭 구분 포맷을 사용합니다. 첫 번째 줄(행)은 머리글 행입니다. 나머지 행에는 로드할 각 비트맵에 대한 다음 데이터가 포함됩니다.

- 비트맵의 파일 이름
- 비트맵의 표시 이름
- 비트맵의 흰색 임계값 및 검정 임계값. 16진수 값으로 이 값 위 및 아래의 픽셀은 완전한 흰색이나 완전한 검정으로 간주됩니다.

응용 프로그램이 시작되면 즉시 AsciiArtBuilder 클래스에서는 AsciiArtBuilder 클래스의 parseImageInfo() 메서드의 다음 코드를 사용하여, 표시할 이미지의 "스택"을 만들기 위해 텍스트 파일의 내용을 로드하고 파싱합니다.

```
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);
    }
}
```

텍스트 파일의 전체 내용은 단일 문자열 인스턴스인 _imageInfoLoader.data 속성에 포함됩니다. 개행 문자("\n")를 매개 변수로 한 split() 메서드를 사용하여, 문자열 인스턴스를 각 요소가 텍스트 파일의 개별 행인 배열(lines)로 분할합니다. 그런 다음, 코드에서 루프를 사용하여 각 행에 대한 작업을 실행합니다(첫 번째 행은 실제 내용이 아닌 머리글만 포함되므로 제외). 루프 내에서 split() 메서드를 한 번 더 사용하여 한 행의 내용을 하나의 값 세트(imageProperties라는 Array 객체)로 분할합니다. 이때 split() 메서드와 함께 사용되는 매개 변수는 각 행의 값이 탭 문자로 구분되어 있으므로 탭("\t") 문자입니다.

String 메서드를 사용하여 이미지 제목 정규화

Flash Player 9 이상, Adobe AIR 1.0 이상

이 응용 프로그램의 디자인 결정 사항 중 하나는 모든 이미지 제목을 각 단어의 첫 번째 글자를 대문자로 표시하는 표준 포맷을 사용하여 표시하는 것입니다(영어 제목에서 일반적으로 대문자로 쓰이지 않는 일부 단어 제외). 텍스트 파일에 적절하게 포맷된 제목이 포함되었다고 가정하는 것이 아니라, 이 응용 프로그램은 텍스트 파일에서 제목을 추출할 때 직접 제목을 포맷합니다.

이전 코드 샘플에서 개별 이미지 메타데이터 값 추출의 일부로 다음 코드 행이 사용됩니다.

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

이 코드에서 텍스트 파일의 이미지 제목은 ImageInfo 객체에 저장되기 전에 normalizeTitle() 메서드를 통해 전달됩니다.

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

이 메서드는 split() 메서드를 사용하여 제목을 개별 단어(공백 문자로 구분)로 분리한 뒤 capitalizeFirstLetter() 메서드를 통해 각 단어를 전달하고 Array 클래스의 join() 메서드를 사용하여 단어를 단일 문자열로 다시 결합합니다.

이름에서 알 수 있듯이 capitalizeFirstLetter() 메서드는 각 단어의 첫 번째 글자를 대문자로 바꾸는 작업을 합니다.

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

영어에서는 제목에 "and", "the", "in", "an", "or", "at", "of" 또는 "a"와 같은 단어가 있을 경우 해당 단어의 첫 자는 대문자로 쓰지 않습니다(간소화된 규칙 버전). 이 논리를 실행하기 위해 코드에서는 먼저 switch 문을 사용하여 단어가 대문자로 쓰면 안 되는 단어인지 확인합니다. 대문자로 쓰면 안 되는 단어 중 하나일 경우 코드는 switch 문을 빠져 나옵니다. 반면 대문자로 써야 할 단어일 경우 다음과 같이 여러 단계를 수행합니다.

1. 단어의 첫 번째 글자가 substr(0, 1)을 사용하여 추출됩니다. 이 메서드는 인덱스 0에 있는 문자(첫 번째 매개 변수 0으로 지정된 대로 문자열의 첫 번째 글자)로 시작되는 하위 문자열을 추출합니다. 하위 문자열은 두 번째 매개 변수 1로 지정된 길이 대로 한 문자가 됩니다.
2. 이 문자를 toUpperCase() 메서드를 사용하여 대문자화합니다.
3. 원래 단어의 나머지 문자는 substring(1)을 사용하여 추출되며, 이 메서드는 인덱스 1에서 시작하는 하위 문자열(두 번째 글자)을 문자열의 끝까지(substring() 메서드의 두 번째 매개 변수를 생략하여 지정) 추출합니다.
4. 최종 단어는 문자열 연결(firstLetter + otherLetters)을 사용하여 대문자로 바뀐 첫 번째 글자와 나머지 글자를 결합하여 만들어집니다.

ASCII Art 텍스트 생성

Flash Player 9 이상, Adobe AIR 1.0 이상

BitmapToAsciiConverter 클래스는 비트맵 이미지를 ASCII 텍스트 표현으로 변환하는 기능을 제공합니다. 이 프로세스는 parseBitmapData() 메서드에서 수행되며 아래에 프로세스 일부가 나와 있습니다.

```
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;
```

이 코드는 비트맵 이미지의 ASCII Art 버전을 작성하는 데 사용될 result라는 문자열 인스턴스를 먼저 정의합니다. 그런 다음 소스 비트맵 이미지의 개별 픽셀에 대해 작업을 반복합니다. 즉, 여러 색상 조작 기술을 사용하여(지면 관계상 자세한 내용은 생략) 개별 픽셀의 빨강, 녹색, 파랑 색상 값을 단일 회색 음영 값(0-255)으로 변환합니다. 그런 다음 위의 코드와 같이 이 값을 4로 나누어 0-63 사이의 값으로 변환한 후, 변수 index에 저장합니다. 이 응용 프로그램에서 사용되는 사용 가능한 ASCII 문자 "팔레트"에 64개의 값이 포함되므로 0-63 사이의 값이 사용됩니다. 문자의 팔레트는 BitmapToAsciiConverter 클래스에서 문자열 인스턴스로 정의됩니다.

```
// 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><+_~-;,. ";
```

index 변수가 비트맵 이미지의 현재 픽셀에 해당하는 팔레트의 ASCII 문자를 정의하므로, 문자는 charAt() 메서드를 사용하여 palette String에서 가져옵니다. 그런 다음 이 문자를 연결 대입 연산자(+=)를 사용하여 result 문자열 인스턴스에 추가합니다. 또한 각 픽셀 행의 끝에서, 개행 문자를 result String의 끝에 연결하여 새 문자 "픽셀" 행을 줄을 바꿔 표시하도록 합니다.

3장: 배열을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

배열을 통해 여러 값을 하나의 데이터 구조에 저장할 수 있습니다. 정해진 순서의 정수 인덱스를 사용하여 값을 저장하는 간단한 인덱스 배열이나 임의의 키를 사용하여 값을 저장하는 복잡한 연관 배열을 사용할 수 있습니다. 배열은 자체가 배열인 요소를 포함하는 다차원으로 구성될 수도 있습니다. 마지막으로, 모든 요소가 동일한 데이터 유형의 인스턴스인 배열에는 Vector를 사용할 수 있습니다.

기타 도움말 항목

Array

Vector

배열의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

프로그래밍할 때는 단일 객체보다 여러 개의 항목을 사용하여 작업해야 하는 경우가 많습니다. 예를 들어 뮤직 플레이어 응용 프로그램에서는 재생할 노래 목록을 포함해야 할 수 있습니다. 이 경우 노래 목록에 있는 각 노래에 대해 별도의 변수를 만들기 보다는 모든 Song 객체를 한 그룹으로 묶어 작업하는 것이 좋습니다.

배열이란 노래 목록과 같은 항목 집합의 컨테이너 역할을 하는 프로그래밍 요소를 의미합니다. 대체적으로 배열에 포함된 모든 항목은 동일한 클래스의 인스턴스이지만 ActionScript에서는 반드시 그렇지만은 않습니다. 배열의 개별 항목은 배열 **요소**라고 합니다. 배열은 변수 보관용 파일 서랍으로 생각할 수도 있습니다. 파일 서랍에 서류철을 넣듯이 변수를 배열의 요소로 추가할 수 있습니다. 서랍 전체를 다른 위치로 옮겨 갈 때처럼 배열을 하나의 변수로 사용할 수 있으며, 여러 서류철을 하나씩 넘기면서 정보를 찾을 때처럼 여러 변수를 하나의 그룹으로 사용할 수 있습니다. 또한 서랍을 열고 하나의 서류철을 선택할 때처럼 각 변수에 개별적으로 액세스할 수도 있습니다.

예를 들어 여러 곡을 선택하여 재생 목록에 추가할 수 있는 뮤직 플레이어 응용 프로그램을 개발한다고 가정해 보겠습니다. 이 경우 ActionScript 코드에서 단일 배열을 매개 변수로 받는 addSongsToPlaylist()라는 메서드를 사용합니다. 목록에 추가할 곡 수가 몇 곡 되지 않든, 아주 많든, 혹은 단 1곡이든 관계없이 addSongsToPlaylist() 메서드는 한 번만 호출하며 이 메서드에는 Song 객체가 들어 있는 배열을 전달합니다. addSongsToPlaylist() 메서드에서는 루프를 사용하여 배열의 요소(노래)를 하나씩 확인하고 이를 재생 목록에 실제로 추가할 수 있습니다.

가장 일반적인 유형의 ActionScript 배열은 **인덱스 배열**입니다. 인덱스 배열에서 각 항목은 번호가 지정된 슬롯(**인덱스**)에 저장되며, 항목에 액세스할 때는 번호를 주소처럼 사용합니다. 인덱스 배열은 프로그래밍 시 배열이 필요한 대부분의 경우에 사용할 수 있습니다. Array 클래스는 인덱스 배열을 나타내는 데 사용되는 일반적인 클래스 중 하나입니다.

인덱스 배열은 동일한 유형의 항목(동일한 클래스의 인스턴스인 객체)을 여러 개 저장하는 데 사용되기도 합니다. 그러나 Array 클래스에서는 배열에 포함할 항목의 유형을 제한할 수 있는 방법이 없습니다. Vector 클래스는 단일 배열의 모든 항목이 동일한 유형인 인덱스 배열의 한 종류입니다. 따라서 Array 인스턴스 대신 Vector 인스턴스를 사용하면 성능을 향상시킬 수 있을 뿐 아니라 그 밖에 다른 장점도 있습니다. Vector 클래스는 Flash Player 10 및 Adobe AIR 1.5부터 사용할 수 있습니다.

인덱스 배열을 특수한 형태로 활용한 것이 **다차원 배열**로, 이는 인덱스 배열이 요소로 포함되고 이 배열에 또 다른 요소가 포함되는 인덱스 배열입니다.

또 다른 배열 유형으로 **연관 배열**이 있는데, 여기에서는 숫자 인덱스 대신 문자열 **키**를 사용하여 개별 요소를 식별합니다. 마지막으로 ActionScript 3.0에는 Dictionary 클래스가 포함되어 있습니다. Dictionary 클래스는 문자 그대로 **사전**을 의미하며, 객체 유형을 키로 사용하여 요소를 구별할 수 있습니다.

중요한 개념 및 용어

배열 및 벡터 처리 루틴을 프로그래밍할 때 사용되는 중요한 용어가 아래 참조 목록에 정리되어 있습니다.

배열 여러 객체를 그룹화할 수 있도록 컨테이너 역할을 하는 객체입니다.

배열 액세스([]) 연산자 배열 요소를 고유하게 식별하는 인덱스 또는 키를 묶는 대괄호 쌍입니다. 이 구문은 배열 변수 이름 다음에서 전체 배열이 아니라 배열의 단일 요소를 지정하는 데 사용됩니다.

연관 배열 문자열 키를 사용하여 개별 요소를 식별하는 배열입니다.

기본 유형 Vector 인스턴스에 저장할 수 있는 객체의 데이터 유형입니다.

사전 항목이 객체 쌍(키와 값)으로 구성된 배열입니다. 숫자 인덱스 대신 키를 사용하여 각 요소를 식별합니다.

요소 배열 내의 각 항목을 나타냅니다.

색인 인덱스 배열에 있는 각 요소를 식별하는 "주소"로, 숫자 값을 가집니다.

인덱스 배열 번호가 지정된 위치에 각 요소를 저장하는 표준 유형의 배열이며, 번호(인덱스)로 개별 요소를 식별합니다.

키 연관 배열 또는 사전의 각 요소를 식별하는 문자열 또는 객체입니다.

다차원 배열 단일 값이 아니라 배열이 항목으로 포함된 배열을 나타냅니다.

T 이 설명서에서 Vector 인스턴스의 기본 유형이 무엇이든 관계없이 이 기본 유형을 나타내는 데 사용되는 표준 표기입니다. T 표기는 Type 매개 변수에 대한 설명에서 언급한 것과 같이 클래스 이름을 나타내는 데 사용됩니다. "T"는 "데이터 유형(data type)"의 "유형(type)"을 의미합니다.

Type 매개 변수 Vector 클래스 이름과 함께 사용되어 Vector의 기본 유형(Vector에 저장되는 객체의 데이터 유형)을 지정하는 구문입니다. 이 구문은 마침표(.)와 각괄호(<>)로 묶인 데이터 유형 이름으로 구성됩니다. 즉, Vector.<T>와 같은 형태가 됩니다. 이 설명서에서는 일반적으로 type 매개 변수에 지정된 클래스를 T로 나타냅니다.

벡터 모든 요소가 동일한 데이터 유형의 인스턴스인 배열의 한 종류입니다.

인덱스 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

인덱스 배열은 일련의 하나 이상 값을 부호 없는 정수 값을 사용하여 각 값에 액세스할 수 있도록 구성하여 저장합니다. 첫 번째 인덱스는 항상 숫자 0이며 배열에 요소를 추가할 때마다 인덱스가 1씩 증가합니다. ActionScript 3.0에서는 Array 클래스와 Vector 클래스가 인덱스 배열로 사용됩니다.

인덱스 배열은 인덱스 번호에 부호 없는 32비트 정수를 사용합니다. 인덱스 배열의 최대 크기는 $2^{32} - 1$ 또는 4,294,967,295입니다. 배열의 크기가 최대 크기를 초과할 경우 런타임 오류가 발생합니다.

인덱스 배열의 개별 요소에 액세스하려면 배열 액세스([]) 연산자를 사용하여 액세스하려는 요소의 인덱스 위치를 지정합니다. 예를 들어 다음 코드는 songTitles라는 인덱스 배열 내의 첫 번째 요소(인덱스 0에 있는 요소)를 나타냅니다.

```
songTitles[0]
```

배열 변수 이름 다음에 대괄호로 묶은 인덱스를 지정하여 단일 식별자로 사용할 수 있습니다. 즉, 변수 이름을 사용할 수 있는 어떤 방법에나 이 식별자를 사용할 수 있습니다. 대입문의 왼쪽에 이름과 인덱스를 사용하여 인덱스 배열 요소에 값을 할당할 수 있습니다.

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

마찬가지로, 대입문의 오른쪽에 이름과 인덱스를 사용하여 인덱스 배열 요소의 값을 가져올 수 있습니다.

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

명시적 값을 지정하는 대신 변수를 대괄호로 묶어 사용할 수도 있습니다. 이때 변수에는 uint, positive int 또는 positive integer Number 인스턴스와 같이 음수가 아닌 정수 값이 들어 있어야 합니다. 이 방법은 일반적으로 인덱스 배열의 요소를 "반복"하여 일부 또는 모든 요소에 대해 작업을 수행하려는 경우에 사용됩니다. 다음 코드 샘플에서는 이 방법을 보여 줍니다. 이 코드에서는 루프를 사용하여 oddNumbers라는 Array 객체의 각 값에 액세스합니다. 또한 trace() 문을 사용하여 각 값을 "oddNumber[**index**] = **value**" 형식으로 출력합니다.

```
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());
}
```

Array 클래스

인덱스 배열의 첫 번째 유형은 Array 클래스입니다. Array 인스턴스에는 모든 유형의 데이터 값을 저장할 수 있습니다. 동일한 Array 객체에 데이터 유형이 서로 다른 객체를 저장할 수도 있습니다. 예를 들어 단일 Array 인스턴스의 인덱스 0, 인덱스 1 및 인덱스 2에 String 값, Number 인스턴스 및 XML 객체를 각각 저장할 수 있습니다.

Vector 클래스

ActionScript 3.0에서 사용할 수 있는 또 다른 종류의 인덱스 배열로는 Vector 클래스가 있습니다. Vector 인스턴스는 **유형 배열**입니다. 즉, Vector 인스턴스의 모든 요소는 데이터 유형이 항상 동일합니다.

참고: Vector 클래스는 Flash Player 10 및 Adobe AIR 1.5부터 사용할 수 있습니다.

Vector 변수를 선언하거나 Vector 객체를 인스턴스화할 때는 Vector에 포함할 수 있는 객체의 데이터 유형을 명시적으로 지정합니다. 지정된 데이터 유형을 Vector의 **기본 유형**이라고 합니다. 런타임과 컴파일 타임(엄격 모드)에는 Vector 요소를 설정하거나 Vector에서 값을 가져오는 모든 코드가 검사됩니다. 추가하거나 가져오는 객체의 데이터 유형이 Vector의 기본 유형과 일치하지 않으면 오류가 발생합니다.

Vector 클래스에는 데이터 유형 제한뿐만 아니라 Array 클래스의 경우와 다른 몇 가지 제한이 더 있습니다.

- Vector는 밀착형 배열입니다. Array 객체의 경우 인덱스 1부터 6까지의 위치에는 값이 없더라도 인덱스 0 및 7에는 값이 있을 수 있습니다. 그러나 Vector의 경우에는 각 인덱스에 값 또는 null이 있어야 합니다.
- Vector는 고정 길이일 수도 있습니다. 고정 길이란 Vector에 포함되는 요소 수를 변경할 수 없음을 의미합니다.
- Vector의 요소에 액세스할 때는 경계가 검사됩니다. 마지막 요소(length -1)보다 큰 인덱스에서는 값을 읽을 수 없습니다. 또한 현재 마지막 인덱스보다 2 이상 큰 인덱스에는 값을 설정할 수 없습니다. 즉, 기존 인덱스나 [length] 값과 같은 인덱스에만 값을 설정할 수 있습니다.

이러한 제한 사항으로 인해 Vector에는 모든 요소가 단일 클래스의 인스턴스인 Array 인스턴스보다 세 가지 우수한 점이 있습니다.

- 성능: Vector 인스턴스를 사용하면 Array 인스턴스를 사용할 때보다 배열 요소의 액세스 및 반복 속도가 훨씬 빠릅니다.
- 안전한 유형: 엄격 모드에서 컴파일러가 데이터 유형 오류를 식별할 수 있습니다. 예를 들어 Vector에 잘못된 데이터 유형의 값을 할당하거나 Vector에서 값을 읽을 때 잘못된 데이터 유형을 사용하는 경우가 이러한 오류에 포함됩니다. 런타임에 Vector 객체에 값을 추가하거나 Vector 객체에서 값을 읽을 때도 데이터 유형이 검사됩니다. 그러나 push() 메서드나 unshift() 메서드를 사용하여 Vector에 값을 추가할 때는 인수의 데이터 유형이 컴파일 타임에 검사되지 않습니다. 이러한 메서드를 사용할 때도 값은 런타임에 검사됩니다.
- 안정성: 런타임 범위 검사(또는 고정 길이 검사)로 Array에 비해 안정성이 크게 향상됩니다.

추가적인 제한 사항과 장점을 제외하면 Vector 클래스는 Array 클래스와 매우 유사합니다. Vector 객체의 속성 및 메서드는 Array의 속성 및 메서드와 비슷하며 대부분은 동일합니다. 사용하는 Array의 모든 요소가 같은 데이터 유형인 대부분의 경우 Vector 인스턴스를 사용하는 것이 좋습니다.

배열 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

Array 인스턴스나 Vector 인스턴스를 만드는 방법에는 몇 가지가 있습니다. 그러나 각 배열 종류를 만드는 방법은 약간 다릅니다.

Array 인스턴스 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

Array 객체를 만들려면 Array() 생성자를 호출하거나 Array 리터럴 구문을 사용합니다.

Array() 생성자 함수는 세 가지 방법으로 사용할 수 있습니다. 첫 번째, 인수 없이 생성자를 호출할 경우 빈 배열을 얻습니다. Array 클래스의 length 속성을 사용하여 배열에 요소가 없는지 확인할 수 있습니다. 예를 들어 다음 코드는 인수 없이 Array() 생성자를 호출합니다.

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

두 번째, Array() 생성자에 대한 유일한 매개 변수로 숫자를 사용할 경우 이 숫자에 해당하는 길이의 배열이 만들어지고 각 요소의 값은 undefined로 설정됩니다. 인수는 0과 4,294,967,295 사이의 부호 없는 정수여야 합니다. 예를 들어 다음 코드는 단일 숫자 인수로 Array() 생성자를 호출합니다.

```
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
```

세 번째, 생성자를 호출하고 매개 변수로 요소 목록을 전달할 경우 각 매개 변수에 해당하는 요소로 배열이 만들어집니다. 다음 코드는 Array() 생성자에 세 가지 인수를 전달합니다.

```
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
```

Array 리터럴을 사용하여 배열을 만들 수도 있습니다. Array 리터럴은 다음 예제와 같이 배열 변수에 직접 할당할 수 있습니다.

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

Vector 인스턴스 만들기

Flash Player 10 이상, Adobe AIR 1.5 이상

Vector 인스턴스를 만들려면 Vector.<T>() 생성자를 호출합니다. Vector.<T>() 전역 함수를 호출하여 Vector를 만들 수도 있습니다. 이 함수는 지정된 객체를 Vector 인스턴스로 변환합니다. Flash Professional CS5 이상, Flash Builder 4 이상, Flex 4 이상의 경우 Vector 리터럴 구문을 사용하여 벡터 인스턴스를 만들 수도 있습니다.

Vector 변수를 선언할 때나 Vector 메서드 매개 변수 또는 메서드 반환 유형을 선언할 때는 항상 Vector 변수의 기본 유형을 지정해야 합니다. Vector.<T>() 생성자를 호출하여 Vector 인스턴스를 만들 때 기본 유형을 지정할 수도 있습니다. 바꾸어 말하면 ActionScript에서 Vector라는 용어를 사용할 때는 항상 기본 유형을 지정해야 합니다.

type 매개 변수 구문을 사용하여 Vector의 기본 유형을 지정할 수 있습니다. type 매개 변수는 코드에서 Vector라는 단어의 바로 뒤에 옵니다. 이 매개 변수는 다음 예제와 같이 점(.)과 각괄호(<>)로 묶인 기본 클래스 이름으로 구성됩니다.

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

예제의 첫 번째 행에서는 변수 v를 Vector.<String> 인스턴스로 선언합니다. 즉, 이 변수는 String 인스턴스만 포함할 수 있는 인덱스 배열을 나타냅니다. 두 번째 행에서는 Vector() 생성자를 사용하여 동일한 Vector 유형의 인스턴스, 즉 모든 요소가 String 객체인 Vector를 만듭니다. 이 행에서는 해당 객체를 v에 할당합니다.

Vector.<T>() 생성자 사용

Vector.<T>() 생성자를 인수 없이 사용하면 빈 Vector 인스턴스가 만들어집니다. Vector의 length 속성을 검사하면 Vector가 비어 있는지 테스트할 수 있습니다. 예를 들어 다음 코드에서는 Vector.<T>() 생성자를 인수 없이 사용합니다.

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

Vector에 필요한 초기 요소 수를 미리 알고 있으면 Vector의 요소 수를 미리 정의할 수 있습니다. 일정 개수의 요소가 있는 Vector를 만들려면 요소 수를 첫 번째 매개 변수(length 매개 변수)로 전달합니다. Vector 요소는 비어 있을 수 없으므로 기본 유형의 인스턴스로 채워집니다. 기본 유형이 null 값을 허용하는 참조 유형일 경우 모든 요소에는 null이 포함됩니다. 그렇지 않은 경우에는 모든 요소에 해당 클래스의 기본값이 포함됩니다. 예를 들어 uint 변수는 null일 수 없으므로 다음 코드 샘플에서 ages라는 Vector는 각각 값 0이 들어 있는 7개의 요소를 사용하여 만들어집니다.

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

마지막으로, Vector.<T>() 생성자를 사용하면 두 번째 매개 변수(fixed 매개 변수)에 true를 전달하여 고정 길이의 Vector를 만들 수도 있습니다. 이 경우 Vector는 지정된 개수의 요소를 사용하여 만들어지며 요소 수는 변경할 수 없습니다. 그러나 고정 길이 Vector의 요소 값은 변경할 수 있습니다.

Vector 리터럴 구문 생성자 사용

Flash Professional CS5 이상, Flash Builder 4 이상, Flex 4 이상의 경우 값 목록을 Vector.<T>()생성자에 전달하여 Vector의 초기 값을 지정할 수 있습니다.

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

이 구문에는 다음 정보가 적용됩니다.

- 뒤에 오는 쉼표는 선택 사항입니다.
- 배열에서 빈 항목은 지원되지 않습니다. 예를 들어 var v:Vector.<int> = new <int>[0,,2,]와 같은 명령문은 컴파일 오류가 발생시킵니다.
- Vector 인스턴스는 기본 길이를 지정할 수 없습니다. 대신, 초기화 목록의 요소 수와 동일하게 길이가 설정됩니다.
- Vector 인스턴스는 고정 길이를 사용할지 여부를 지정할 수 없습니다. 대신, fixed 속성을 사용해야 합니다.
- 값으로 전달된 항목이 지정된 유형과 일치하지 않으면 데이터 손실 또는 오류가 발생할 수 있습니다. 예를 들면 다음과 같습니다.

  ```
  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
  ```

Vector.<T>() 전역 함수 사용

Vector.<T>() 및 Vector 리터럴 구문 생성자 외에, Vector.<T>()전역 함수를 사용하여 Vector 객체를 만들 수도 있습니다. Vector.<T>() 전역 함수는 변환 함수입니다. Vector.<T>() 전역 함수를 호출할 때는 메서드에서 반환하는 Vector의 기본 유형을 지정해야 합니다. 단일 인덱스 배열(Array 또는 Vector 인스턴스)을 인수로 전달합니다. 그러면 메서드는 지정된 기본 유형을 사용하며 소스 배열 인수의 값이 들어 있는 Vector를 반환합니다. 다음 코드 샘플에서는 Vector.<T>() 전역 함수를 호출하기 위한 구문을 보여 줍니다.

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

Vector.<T>() 전역 함수는 두 수준에서 데이터 유형 변환을 수행합니다. 먼저, 함수에 Array 인스턴스가 전달될 때는 Vector 인스턴스가 반환됩니다. 두 번째로, 소스 배열이 Array 인스턴스이든 Vector 인스턴스이든 관계없이 이 함수는 소스 배열의 요소를 기본 유형의 값으로 변환하려고 합니다. 변환에는 표준 ActionScript 데이터 유형 변환 규칙이 사용됩니다. 예를 들어 다음 코드 샘플에서는 소스 Array의 문자열 값을 결과 Vector에서 정수로 변환합니다. 결과에서 첫 번째 값("1.5")의 소수 부분은 잘리고 숫자가 아닌 세 번째 값("Waffles")은 0으로 변환됩니다.

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

변환할 수 없는 소스 요소가 있으면 오류가 발생합니다.

코드에서 Vector.<T>() 전역 함수를 호출할 때 소스 배열의 요소가 지정된 기본 유형의 하위 클래스 인스턴스이면 해당 요소가 결과 Vector에 추가되고 오류는 발생하지 않습니다. Vector.<T>() 전역 함수를 사용하는 방법은 기본 유형이 T인 Vector를 기본 유형이 T의 수퍼 클래스인 Vector로 변환하는 유일한 방법입니다.

배열 요소 삽입

Flash Player 9 이상, Adobe AIR 1.0 이상

인덱스 배열에 요소를 추가하는 가장 기본적인 방법은 배열 액세스([]) 연산자를 사용하는 것입니다. 인덱스 배열 요소의 값을 설정하려면 대입문의 왼쪽에 Array 또는 Vector 객체 이름과 인덱스 번호를 사용합니다.

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

Array 또는 Vector의 해당 인덱스에 요소가 아직 없으면 해당 인덱스가 만들어지고 여기에 값이 저장됩니다. 해당 인덱스에 값이 있으면 새 값이 기존 값을 대체합니다.

Array 객체의 경우 어느 인덱스에나 요소를 만들 수 있습니다. 그러나 Vector 객체의 경우에는 기존 인덱스나 사용 가능한 다음 인덱스에만 값을 할당할 수 있습니다. 사용 가능한 다음 인덱스는 Vector 객체의 length 속성 값과 같습니다. Vector 객체에 새 요소를 추가하는 가장 안전한 방법은 다음 샘플과 같은 코드를 사용하는 것입니다.

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

Array 및 Vector 클래스의 push(), unshift() 및 splice() 메서드 세 가지를 사용하여 인덱스 배열에 요소를 삽입할 수 있습니다. push() 메서드는 배열 끝에 하나 이상의 요소를 추가합니다. 즉, push() 메서드를 사용하여 배열에 삽입된 마지막 요소는 최상위 인덱스 번호를 가지게 됩니다. unshift() 메서드는 배열 시작 부분에 하나 이상의 요소를 삽입하며 이 위치의 인덱스 번호는 항상 0입니다. splice() 메서드는 지정된 인덱스 위치의 모든 항목을 배열에 삽입합니다.

다음 예제는 이 세 가지 메서드를 모두 보여 줍니다. planets 배열은 태양에 가까운 순서로 행성의 이름을 저장하기 위해 만들어진 배열입니다. 먼저, push() 메서드가 첫 항목 Mars를 추가하기 위해 호출됩니다. 그 다음, unshift() 메서드가 배열 맨 앞에 있어야 할 항목인 Mercury를 삽입하기 위해 호출됩니다. 마지막으로, splice() 메서드가 Mercury 뒤 및 Mars 앞에 Venus와 Earth 항목을 삽입하기 위해 호출됩니다. splice()에 전달된 첫 번째 인수인 정수 1은 인덱스 1에서 삽입을 시작하도록 지시합니다. splice()로 전달된 두 번째 인수인 정수 0은 삭제해야 할 항목이 없음을 의미합니다. 마지막으로 splice()에 전달된 세 번째 및 네 번째 인수인 Venus 및 Earth는 삽입할 항목입니다.

```
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
```

push() 및 unshift() 메서드는 모두 수정된 배열의 길이를 나타내는 부호 없는 정수를 반환합니다. splice() 메서드는 요소를 삽입하기 위해 사용될 때 빈 배열을 반환합니다. 이상해 보이지만 splice() 메서드의 다양한 용도를 생각해 보면 이해할 수 있습니다. splice() 메서드를 사용하여 배열에 요소를 삽입할 수 있을 뿐만 아니라 배열에서 요소를 제거할 수도 있습니다. 요소를 제거하기 위해 사용할 경우 splice() 메서드는 제거된 요소를 포함하는 배열을 반환합니다.

참고: Vector 객체의 fixed 속성이 true일 경우에는 Vector의 전체 요소 수를 변경할 수 없습니다. 여기에서 설명한 방법을 사용하여 고정 길이의 Vector에 새 요소를 추가하려고 하면 오류가 발생합니다.

값 가져오기 및 배열 요소 제거

Flash Player 9 이상, Adobe AIR 1.0 이상

인덱스 배열에서 요소 값을 가져오는 가장 간단한 방법은 배열 액세스([]) 연산자를 사용하는 것입니다. 인덱스 배열 요소의 값을 가져오려면 대입문의 오른쪽에 Array 또는 Vector 객체 이름과 인덱스 번호를 사용합니다.

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

요소가 없는 Array 또는 Vector에서는 값을 가져올 수 없습니다. 이 경우 Array 객체에서는 undefined 값이 반환되고 Vector 객체에서는 RangeError 예외가 발생합니다.

Array 및 Vector 클래스의 pop(), shift() 및 splice() 메서드 세 가지를 사용하여 요소를 제거할 수 있습니다. pop() 메서드는 배열 끝에서 요소를 제거합니다. 즉, 인덱스 번호가 가장 높은 요소를 제거합니다. shift() 메서드는 배열 시작 부분에서 요소를 제거합니다. 즉, 항상 인덱스 번호 0의 위치에 있는 요소를 제거합니다. 요소를 삽입하는 데에도 사용할 수 있는 splice() 메서드는 이 메서드에 전달된 첫 번째 인수에 의해 지정된 인덱스 번호부터 시작하여 임의의 개수의 요소를 제거합니다.

다음 예제에서는 이 세 가지 메서드를 모두 사용하여 Array 인스턴스에서 요소를 제거합니다. oceans라는 Array는 거대한 수역의 이름을 저장하기 위해 만들어진 배열입니다. Array의 일부 이름은 바다가 아닌 호수라서 제거해야 합니다.

먼저, splice() 메서드를 사용하여 Aral 및 Superior 항목을 제거하고 Atlantic 및 Indian 항목을 삽입합니다. splice()에 전달된 첫 번째 인수인 정수 2는 인덱스 2 위치에 있는 목록의 세 번째 항목으로 작업을 시작해야 함을 나타냅니다. 두 번째 인수 2는 두 개의 항목이 제거되어야 함을 나타냅니다. 남은 인수인 Atlantic 및 Indian은 인덱스 2 위치에 삽입할 값입니다.

그 다음, pop() 메서드를 사용하여 배열에서 마지막 요소인 Huron을 제거합니다. 세 번째로 shift() 메서드를 사용하여 배열의 첫 번째 항목인 Victoria를 제거합니다.

```
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
```

pop() 및 shift() 메서드는 모두 제거된 항목을 반환합니다. Array 인스턴스의 경우 배열에 모든 데이터 유형의 값을 저장할 수 있으므로 반환 값의 데이터 유형은 Object입니다. Vector 인스턴스의 경우 반환 값의 데이터 유형은 Vector의 기본 유형입니다. splice() 메서드는 제거된 값을 포함하는 Array 또는 Vector를 반환합니다. oceans Array 예제를 변경하여, 다음 예제와 같이 splice()에 대한 호출에서 반환된 Array가 새 Array 변수에 할당되도록 할 수 있습니다.

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

Array 객체 요소에 대해 delete 연산자를 사용하는 코드가 있을 수 있습니다. delete 연산자는 Array 요소 값을 undefined로 설정하지만 Array에서 요소를 제거하지는 않습니다. 예를 들어 다음 코드에서는 oceans Array의 세 번째 요소에 대해 delete 연산자를 사용하지만 Array의 길이는 여전히 5입니다.

```
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
```

배열의 length 속성을 사용하여 Array 또는 Vector를 자를 수 있습니다. 인덱스 배열의 length 속성을 배열의 현재 길이보다 작게 설정할 경우, 배열이 잘리고 새 length 값에서 1을 뺀 값보다 큰 인덱스 번호 위치에 저장된 모든 요소가 제거됩니다. 예를 들어 다음 코드와 같이 oceans 배열을 배열 시작 부분에 모든 유효 항목이 오도록 정렬한 경우 length 속성을 사용하여 배열 끝에 있는 항목을 제거할 수 있습니다.

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

참고: Vector 객체의 fixed 속성이 true일 경우에는 Vector의 전체 요소 수를 변경할 수 없습니다. 여기에서 설명한 방법을 사용하여 고정 길이의 Vector에서 요소를 제거하거나 해당 Vector를 자르려고 하면 오류가 발생합니다.

배열 정렬

Flash Player 9 이상, Adobe AIR 1.0 이상

reverse(), sort() 및 sortOn() 메서드 세 가지를 사용하면 인덱스 배열을 정렬 또는 역정렬하여 순서를 변경할 수 있습니다. 이 세 가지 메서드는 모두 기존 배열을 수정합니다. 다음 표에는 이러한 메서드와 Array 및 Vector 객체에 대한 메서드 비헤이비어가 요약되어 있습니다.

메서드	Array 비헤이비어	Vector 비헤이비어
reverse()	마지막 요소가 첫 번째 요소가 되고 끝에서 두 번째 요소가 두 번째 요소가 되는 식으로 요소 순서를 변경합니다.	Array 비헤이비어와 동일합니다.
sort()	알파벳순이나 숫자순과 같이 미리 정의된 다양한 방법으로 Array의 요소를 정렬할 수 있습니다. 사용자 정의 정렬 알고리즘을 지정할 수도 있습니다.	사용자가 지정하는 사용자 정의 정렬 알고리즘에 따라 요소를 정렬합니다.
sortOn()	하나 이상의 공통 속성이 있는 객체를 정렬할 수 있습니다. 이때 정렬 키로 사용할 속성을 지정합니다.	Vector 클래스에서는 사용할 수 없습니다.

reverse() 메서드

reverse() 메서드는 매개 변수를 취하지 않으며 값을 반환하지도 않습니다. 다만 이 메서드를 사용하여 배열의 순서를 현재 상태에서 역순으로 전환할 수 있습니다. 다음 예제는 oceans 배열에 나열된 바다의 순서를 역순으로 전환합니다.

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

sort() 메서드를 사용한 기본 정렬(Array 클래스만 해당)

Array 인스턴스의 경우 sort() 메서드는 기본 정렬 순서를 사용하여 배열의 요소를 다시 정렬합니다. 기본 정렬 순서의 특징은 다음과 같습니다.

- 대/소문자를 구분하며 대문자는 소문자보다 우선합니다. 예를 들어 글자 D가 글자 b보다 우선합니다.
- 오름차순으로 낮은 문자 코드(예: A)가 높은 문자 코드(예: B)보다 우선합니다.
- 동일한 값은 특정 순서 없이 인접하여 배치합니다.
- 문자열 기반이므로, 요소를 비교하기 전에 문자열로 변환합니다. 예를 들어 문자열 "1"이 문자열 "3"보다 낮은 문자 코드를 가지므로 10이 3보다 먼저 오게 됩니다.

Array를 대/소문자 구분 없이 정렬하거나 내림차순으로 정렬하거나 숫자가 포함된 배열을 알파벳순이 아닌 숫자순으로 정렬해야 할 경우가 있습니다. Array 클래스의 sort() 메서드에는 기본 정렬 순서의 각 특성을 변경할 수 있는 options 매개 변수가 있습니다. 이 옵션은 다음 목록과 같이 Array 클래스의 정적 상수 세트로 정의됩니다.

- Array.CASEINSENSITIVE: 이 옵션은 정렬 시 대/소문자가 무시되도록 합니다. 예를 들어 소문자 글자 b가 대문자 글자 D보다 우선합니다.
- Array.DESCENDING: 이 옵션은 기본 오름차순 정렬을 역으로 설정합니다. 예를 들어 글자 B가 글자 A보다 우선합니다.
- Array.UNIQUESORT: 이 옵션은 동일한 값이 두 개 있을 경우 정렬을 중단하도록 합니다.
- Array.NUMERIC: 숫자순으로 정렬하는 옵션이므로 3이 10보다 먼저 오게 됩니다.

다음 예제는 이 옵션 중 일부를 보여 줍니다. poets라는 Array가 만들어져 다양한 여러 옵션을 사용하여 정렬됩니다.

```
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
```

sort() 메서드를 사용한 사용자 정의 정렬(Array 및 Vector 클래스)

Array 객체에 사용할 수 있는 기본 정렬 외에 사용자 정의 정렬 규칙을 정의할 수도 있습니다. 이 방법은 Vector 클래스에 사용할 수 있는 유일한 형태의 sort() 메서드입니다. 사용자 정의 정렬을 정의하려면 사용자 정의 정렬 함수를 작성하여 sort() 메서드에 인수로 전달합니다.

예를 들어 각 목록 요소에 개인의 전체 이름이 포함된 이름 목록이 있는 경우, 이 목록을 성(姓)을 사용하여 정렬하려고 하면 사용자 정의 정렬 함수를 사용하여 각 요소를 파싱하고 정렬 함수에서 성을 사용해야 합니다. 다음 코드는 Array.sort() 메서드에 대해 사용자 정의 함수를 매개 변수로 사용하여 이 작업을 수행하는 방법을 보여 줍니다.

```
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
```

사용자 정의 정렬 함수 orderLastName()은 일반 표현식을 사용하여 각 요소에서 비교 작업에 사용할 성을 추출합니다. 함수 식별자 orderLastName은 names 배열에서 sort() 메서드를 호출할 때 유일한 매개 변수로 사용됩니다. 정렬 함수는 한 번에 두 개의 배열 요소로 작업하므로 두 개의 매개 변수 a 및 b를 취합니다. 정렬 함수의 반환 값은 요소가 정렬되는 방법을 나타냅니다.

- 반환 값 -1은 첫 번째 매개 변수인 a가 두 번째 매개 변수인 b보다 우선함을 나타냅니다.
- 반환 값 1은 두 번째 매개 변수인 b가 첫 번째 매개 변수인 a보다 우선함을 나타냅니다.
- 반환 값 0은 요소의 정렬 우선 순위가 동일함을 나타냅니다.

sortOn() 메서드(Array 클래스만 해당)

sortOn() 메서드는 요소에 객체가 포함된 Array 객체를 위해 설계되었습니다. 이러한 객체에는 정렬 키로 사용할 수 있는 공통 속성이 최소한 하나는 있어야 합니다. 다른 유형의 배열에 sortOn() 메서드를 사용할 경우 예기치 못한 결과를 초래할 수 있습니다.

참고: Vector 클래스에는 sortOn() 메서드가 포함되어 있지 않습니다. 이 메서드는 Array 객체에만 사용할 수 있습니다.

다음 예제에서는 각 요소가 문자열이 아닌 객체가 되도록 poets Array를 수정합니다. 각 객체에는 시인의 성과 출생 연도가 포함되어 있습니다.

```
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"});
```

sortOn() 메서드를 사용하면 born 속성을 기준으로 배열을 정렬할 수 있습니다. sortOn() 메서드는 두 개의 매개 변수 fieldName 및 options를 정의합니다. fieldName 인수는 문자열로 지정해야 합니다. 다음 예제에서는 두 개의 인수 "born" 및 Array.NUMERIC으로 sortOn()이 호출됩니다. Array.NUMERIC 인수는 알파벳순이 아닌 숫자순으로 정렬하려고 할 때 사용합니다. 이는 모든 숫자의 자릿수가 같은 경우에도 유용한 방법입니다. 나중에 이 배열에 자릿수가 많거나 적은 숫자를 추가하게 되더라도 예상한 대로 정렬이 수행될 수 있기 때문입니다.

```
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
*/
```

원래 배열을 수정하지 않는 정렬(Array 클래스만 해당)

일반적으로 sort() 및 sortOn() 메서드는 Array를 수정합니다. 기존 배열을 수정하지 않고 Array를 정렬하려면 Array.RETURNINDEXEDARRAY 상수를 options 매개 변수의 일부로 전달합니다. 이렇게 하면 메서드는 정렬이 적용된 새 Array를 반환하고 원래의 Array은 수정하지 않은 채로 둡니다. 메서드에서 반환된 Array는 새 정렬 순서가 적용된 간단한 인덱스 번호의 Array이며 원래 Array의 요소는 전혀 포함하지 않습니다. 예를 들어 poets Array를 수정하지 않고 출생 연도순으로 정렬하려면 Array.RETURNINDEXEDARRAY 상수를 options 매개 변수에 전달되는 인수의 일부로 포함합니다.

다음 예제에서는 반환된 인덱스 정보를 indices라는 Array에 저장하고, 이 indices 배열을 수정되지 않은 poets 배열과 함께 사용하여 출생 연도순으로 시인의 목록을 출력합니다.

```
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
*/
```

배열 쿼리

Flash Player 9 이상, Adobe AIR 1.0 이상

Array 및 Vector 클래스의 concat(), join(), slice() 및 toString() 메서드 네 가지는 배열에서 정보를 쿼리하지만 배열을 수정하지는 않습니다. join() 및 toString() 메서드가 모두 문자열을 반환하는 반면. concat() 및 slice() 메서드는 모두 새 배열을 반환합니다. concat() 메서드는 인수로 요소 목록이나 새 배열을 취하며 이를 기존 배열과 결합하여 새 배열을 만듭니다. slice() 메서드에는 startIndex와 endIndex라는 두 개의 매개 변수가 있으며 기존 배열에서 "분할"된 요소의 복사본이 포함된 새로운 배열을 반환합니다. 분할은 startIndex 위치의 요소에서 시작되어 endIndex 직전 위치에 있는 요소에서 종료됩니다. 즉, endIndex 위치의 요소는 반환 값에 포함되지 않습니다.

다음 예제에서는 다른 배열의 요소를 사용하여 새 배열을 만들기 위해 concat() 및 slice()를 사용합니다.

```
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
```

join() 및 toString() 메서드를 사용하여 배열에 쿼리를 실행하고 그 내용을 문자열로 반환할 수 있습니다. join() 메서드에 매개 변수를 사용하지 않는 경우 두 메서드는 동일하게 동작합니다. 즉, 배열에 있는 모든 요소의 목록이 쉼표로 구분되어 포함된 문자열을 반환합니다. toString() 메서드와는 달리 join() 메서드는 delimiter라는 매개 변수를 취하여, 반환된 문자열에서 각 요소 간을 분리하는 데 사용되는 구분 기호를 선택할 수 있습니다.

다음 예제에서는 rivers라는 Array를 만들고 join() 및 toString()을 모두 호출하여 Array의 값을 문자열로 반환합니다. toString() 메서드는 쉼표로 구분된 값(riverCSV)을 반환하는 데 사용되는 반면, join() 메서드는 + 문자로 구분된 값을 반환하는 데 사용됩니다.

```
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
```

단, 다음 예제와 같이 중첩된 Array 또는 Vector 인스턴스는 join() 메서드에서 항상 쉼표로 구분된 값으로 반환됩니다. 이때 기본 배열 요소에 지정한 분리 기호는 무시됩니다.

```
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
```

연관 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

해시 또는 맵이라고도 하는 연관 배열은 숫자 인덱스 대신 키를 사용하여 저장된 값을 구성합니다. 연관 배열의 각 키는 저장된 값에 액세스하기 위해 사용되는 고유한 문자열입니다. 연관 배열은 Object 클래스의 인스턴스이며 각 키는 속성 이름에 해당합니다. 연관 배열은 정렬되지 않은 키/값 쌍 모음입니다. 코드에서 연관 배열의 키가 특정한 순서로 되어 있으면 안 됩니다.

ActionScript 3.0에는 사전이라는 고급 연관 배열 유형이 포함되었습니다. 사전은 flash.utils 패키지에 있는 Dictionary 클래스의 인스턴스로, 모든 데이터 유형의 키를 사용합니다. 즉, 사전 키에는 String 유형이 아닌 값도 사용할 수 있습니다.

문자열 키가 있는 연관 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 두 가지 방법으로 연관 배열을 만들 수 있습니다. 첫 번째 방법은 Object 인스턴스를 사용하는 것입니다. Object 인스턴스를 사용하면 객체 리터럴로 배열을 초기화할 수 있습니다. 일반 객체라고도 하는 Object 클래스의 인스턴스는 연관 배열과 기능면에서 동일합니다. 일반 객체의 각 속성 이름은 저장된 값에 대한 액세스를 제공하는 키 역할을 합니다.

다음 예제는 monitorInfo라는 연관 배열을 만들고, 객체 리터럴을 사용하여 두 개의 키/값 쌍으로 배열을 초기화합니다.

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

선언 시 배열을 초기화할 필요가 없을 경우에는 다음과 같이 Object 생성자를 사용하여 배열을 만들 수 있습니다.

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

객체 리터럴이나 Object 클래스 생성자를 사용하여 배열을 만든 후에는 배열 액세스([]) 연산자나 도트 연산자(.)를 사용하여 배열에 새 값을 추가할 수 있습니다. 다음 예제에서는 두 개의 새로운 값을 monitorArray에 추가합니다.

```
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
```

aspect ratio 키에는 공백 문자가 포함됩니다. 이는 배열 액세스([]) 연산자를 사용하는 경우에만 가능하고 도트 연산자를 사용하는 경우에 공백 문자가 포함되면 오류가 발생합니다. 그러나 키 이름에 공백을 사용하는 것은 권장되지 않습니다.

연관 배열을 만드는 두 번째 방법은 Array 생성자(또는 동적 클래스의 생성자)를 사용한 다음 배열 액세스([]) 연산자나 도트 연산자(.)를 사용하여 배열에 키/값 쌍을 추가하는 것입니다. 연관 배열을 배열 유형으로 선언할 경우에는 객체 리터럴을 사용하여 배열을 초기화할 수 없습니다. 다음 예제에서는 Array 생성자를 사용하여 monitorInfo라는 연관 배열을 만들고 type이라는 키와 resolution이라는 키를 각각의 값과 함께 추가합니다.

```
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
```

Array 생성자를 사용하여 연관 배열을 만들어도 별다른 이점은 없습니다. Array 생성자나 Array 데이터 유형을 사용하는 경우에도 연관 배열에 Array.length 속성이나 Array 클래스의 메서드를 모두 사용할 수 없습니다. Array 생성자는 인덱스 배열을 만드는 데 사용하는 것이 가장 좋습니다.

객체 키가 있는 연관 배열(Dictionary)

Flash Player 9 이상, Adobe AIR 1.0 이상

Dictionary 클래스를 사용하여 문자열이 아닌 객체를 키로 사용하는 연관 배열을 생성할 수 있습니다. 이러한 배열을 사전, 해시 또는 맵이라고도 합니다. 예를 들어 특정 컨테이너와의 연관성을 기준으로 Sprite 객체의 위치를 결정하는 응용 프로그램을 살펴봅니다. 이 경우 Dictionary 객체를 사용하여 컨테이너에 각 Sprite 객체를 매핑할 수 있습니다.

다음 코드는 Dictionary 객체의 키 역할을 수행할 Sprite 클래스의 세 가지 인스턴스를 만듭니다. 각 키에는 GroupA 또는 GroupB의 값이 할당됩니다. 값은 모든 데이터 유형이 될 수 있지만 이 예제에서는 GroupA 및 GroupB가 모두 Object 클래스의 인스턴스입니다. 그런 후 다음 코드와 같이 배열 액세스([]) 연산자를 사용하여 각 키와 연관된 값에 액세스할 수 있습니다.

```
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");
}
```

객체 키로 반복

for..in 루프나 for each..in 루프를 사용하여 Dictionary 객체의 내용을 반복할 수 있습니다. for..in 루프를 사용하면 키를 기준으로 반복할 수 있으며, for each..in 루프를 사용하면 각 키와 연관된 값을 기준으로 반복할 수 있습니다.

Dictionary 객체의 객체 키에 직접 액세스하려면 for..in 루프를 사용합니다. 배열 액세스([]) 연산자를 사용하여 Dictionary 객체의 값에 액세스할 수도 있습니다. 다음 코드에서는 groupMap 사전의 이전 예제를 사용하여 for..in 루프를 통해 Dictionary 객체를 반복하는 방법을 보여 줍니다.

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

Dictionary 객체의 값에 직접 액세스하려면 for each..in 루프를 사용합니다. 다음 코드에서는 groupMap 사전을 사용하여 for each..in 루프를 통해 Dictionary 객체를 반복하는 방법을 보여 줍니다.

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

객체 키 및 메모리 관리

Adobe® Flash® Player 및 Adobe® AIR™에서는 가비지 컬렉션 시스템을 사용하여 더 이상 사용되지 않는 메모리를 복구합니다. 객체에 객체를 가리키는 참조가 없을 경우 객체는 가비지 컬렉션의 대상이 되며 다음 번 가비지 컬렉션 시스템 실행 시 메모리가 복구됩니다. 예를 들어 다음 코드는 새 객체를 만들어 변수 myObject에 객체에 대한 참조를 할당합니다.

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

객체에 대한 참조가 남아 있는 한, 가비지 컬렉션 시스템에서는 객체가 사용하고 있는 메모리를 복구하지 않습니다. 다른 객체를 가리키거나 null 값으로 설정되는 등 myObject 값이 변경되면, 원래 객체에 의해 사용되던 메모리는 가비지 컬렉션의 대상이 됩니다. 그러나 이 원래 객체에 대한 다른 참조가 없는 경우에만 해당됩니다.

Dictionary 객체에서 myObject를 키로 사용하는 경우 원래 객체에 대한 다른 참조가 만들어집니다. 예를 들어 다음 코드에서는 객체에 대한 두 개의 참조 즉, myObject 변수 및 myMap 객체의 키를 만듭니다.

```
import flash.utils.Dictionary;

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

myObject로 참조되는 객체를 가비지 컬렉션 대상으로 만들려면 객체에 대한 참조를 모두 제거해야 합니다. 이 경우, 다음 코드와 같이 myObject 값을 변경하고 myMap에서 myObject 키를 삭제해야 합니다.

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

또는 Dictionary 생성자의 useWeakReference 매개 변수를 사용하여 사전 키를 모두 **약한 참조**로 만들 수 있습니다. 가비지 컬렉션 시스템에서는 약한 참조를 무시하므로 약한 참조만 있는 객체는 가비지 컬렉션의 대상이 됩니다. 예를 들어 다음 코드에서는 객체를 가비지 컬렉션 대상으로 만들기 위해 myMap에서 myObject 키를 삭제할 필요가 없습니다.

```
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.
```

다차원 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

다차원 배열에는 다른 배열이 요소로 포함됩니다. 예를 들어 문자열에 대한 인덱스 배열로 저장된 작업 목록을 생각해 보십시오.

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

별도의 작업 목록을 매일 저장하려는 경우 매일 하나의 요소를 사용하여 다차원 배열을 만들 수 있습니다. 각 요소에는 작업 목록을 저장하는 tasks 배열과 비슷한 인덱스 배열이 포함됩니다. 다차원 배열에서는 인덱스 배열이나 연관 배열을 조합하여 사용할 수 있습니다. 다음 단원의 예제에서는 두 개의 인덱스 배열이나 인덱스 배열의 연관 배열을 사용합니다. 필요한 경우 다른 조합을 시도해 볼 수도 있습니다.

두 개의 인덱스 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

두 개의 인덱스 배열을 사용할 경우 결과를 표나 스프레드시트로 표시할 수 있습니다. 첫 번째 배열의 요소는 표의 행을 나타내며 두 번째 배열의 요소는 열을 나타냅니다.

예를 들어 다음 다차원 배열에서는 두 개의 인덱스 배열을 사용하여 요일별로 작업 목록을 추적합니다. 첫 번째 배열인 masterTaskList는 Array 클래스 생성자를 사용하여 만들어집니다. 배열의 각 요소는 요일을 나타내며 인덱스 0은 월요일, 인덱스 6은 일요일을 나타냅니다. 이 요소를 표의 행이라고 생각할 수 있습니다. masterTaskList 배열에서 만든 7개의 요소에 대해 각각 배열 리터럴을 할당하여 각 요일의 작업 목록을 만들 수 있습니다. 배열 리터럴은 표의 열을 나타냅니다.

```
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"];
```

배열 액세스([]) 연산자를 사용하여 모든 작업 목록에서 개별 항목에 액세스할 수 있습니다. 첫 번째 대괄호 세트는 요일을 나타내며 두 번째 대괄호 세트는 해당 요일에 대한 작업 목록을 나타냅니다. 예를 들어 수요일 목록에서 두 번째 작업을 검색하려면, 먼저 수요일에 해당하는 인덱스 2를 사용한 다음 목록에서 두 번째 작업을 나타내는 인덱스 1을 사용합니다.

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

일요일 목록에서 첫 번째 작업을 검색하려면 일요일에 해당하는 인덱스 6을 사용하고 목록에서 첫 번째 작업을 나타내는 인덱스 0을 사용합니다.

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

인덱스 배열을 포함한 연관 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

개별 배열에 보다 쉽게 액세스할 수 있도록 하기 위해, 요일에는 연관 배열을 사용하고 작업 목록에는 인덱스 배열을 사용할 수 있습니다. 연관 배열을 사용하면 특정 요일을 참조할 때 도트 구문을 사용할 수 있지만, 이 경우 연관 배열의 각 요소에 액세스하기 위한 런타임 처리 시간이 길어집니다. 다음 예제에서는 작업 목록의 기본으로 요일별 키/값 쌍이 있는 연관 배열을 사용합니다.

```
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"];
```

도트 구문을 사용하면 여러 개의 대괄호 세트를 사용하지 않아도 되므로 코드가 한결 읽기 쉬어집니다.

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

for..in 루프를 사용하여 작업 목록을 반복할 수 있지만 각 키와 연관된 값에 액세스하려면 도트 구문 대신 배열 액세스([]) 연산자를 사용해야 합니다. masterTaskList는 연관 배열이기 때문에, 다음 예제와 같이 요소가 원하는 순서로 검색되지 않습니다.

```
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
*/
```

배열 복제

Flash Player 9 이상, Adobe AIR 1.0 이상

Array 클래스에는 배열의 복사본을 만들기 위한 메서드가 내장되어 있지 않습니다. 인수 없이 **concat()** 또는 **slice()** 메서드를 호출하여 배열의 단순복사본을 만들 수 있습니다. 단순 복사본에서 원래 배열에 객체인 요소가 있는 경우 객체는 복사되지 않고 객체에 대한 참조만 복사됩니다. 복사본은 원래 배열과 동일한 객체를 가리킵니다. 객체에 대한 모든 변경 사항은 두 배열에 모두 반영됩니다.

전체 복사본에서는 원래 배열에 있던 객체도 모두 복사되어 새 배열이 원래 배열과 동일한 객체를 가리키지 않습니다. 전체 복사본을 작성하려면 둘 이상의 코드 행이 있어야 하며 일반적으로 함수를 작성해야 합니다. 이 함수는 전역 유틸리티 함수나 Array 하위 클래스의 메서드로 작성될 수 있습니다.

다음 예제에서는 전체 복사본을 작성하는 clone() 함수를 정의합니다. 알고리즘은 일반적인 Java 프로그래밍 기술을 사용합니다. 이 함수는 배열을 ByteArray 클래스 인스턴스로 직렬화하고 배열을 새 배열로 다시 읽어 전체 복사본을 만듭니다. 이 함수는 객체를 허용하므로 다음 코드와 같이 인덱스 배열 및 연관 배열 모두에 사용할 수 있습니다.

```
import flash.utils.ByteArray;

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

Array 클래스 확장

Flash Player 9 이상, Adobe AIR 1.0 이상

Array 클래스는 최종이 아닌 몇 개의 핵심 클래스 중 하나이므로 사용자 고유의 Array 하위 클래스를 만들 수 있습니다. 이 단원에서는 Array 하위 클래스를 만드는 방법에 대한 예제와 이 과정 동안 발생할 수 있는 일부 문제에 대해 살펴봅니다.

앞서 언급했듯이 ActionScript에서 배열의 유형은 정의되어 있지 않지만 특정 데이터 유형의 요소만 허용하는 Array 하위 클래스를 만들 수 있습니다. 다음 단원의 예제에서는 요소를 첫 번째 매개 변수에 지정된 데이터 유형 값으로 제한하는 TypedArray 라는 Array 하위 클래스를 정의합니다. TypedArray 클래스는 Array 클래스를 확장하는 방법을 설명하는 예제로만 사용되며 다음과 같은 여러 가지 이유로 실제로 사용하기에는 적합하지 않을 수 있습니다. 첫 번째, 컴파일 타임이 아닌 런타임에 유형 검사가

실행됩니다. 두 번째, TypedArray 메서드에서 불일치가 발생할 경우, 불일치가 무시되고 예외가 발생하지 않습니다. 메서드를 간단히 수정하여 예외가 발생되도록 할 수는 있습니다. 세 번째, 이 클래스에서는 배열에 모든 유형의 값을 삽입할 수 있는 배열 액세스 연산자의 사용을 막을 수 없습니다. 네 번째, 이 코딩 스타일에서는 성능 최적화보다 간결성을 선호합니다.

참고: 여기에서 설명한 방법을 사용하여 유형 배열을 만들 수 있습니다. 그러나 더 나은 방법은 Vector 객체를 사용하는 것입니다. Vector 인스턴스는 진정한 유형 배열로서, Array 클래스나 모든 하위 클래스에 비해 성능 및 그 밖의 면에서 우수합니다. 이 단원의 목적은 Array 하위 클래스를 만드는 방법을 보여 주는 것입니다.

하위 클래스 선언

extends 키워드를 사용하여 클래스가 Array의 하위 클래스임을 나타낼 수 있습니다. Array 하위 클래스는 Array 클래스와 같이 dynamic 특성을 사용해야 합니다. 그렇지 않으면 하위 클래스가 올바르게 작동하지 않습니다.

다음 코드에서는 TypedArray 클래스의 정의를 보여 줍니다. 이 클래스에는 데이터 유형, 생성자 메서드 및 배열에 요소를 추가할 수 있는 네 가지 메서드를 보유할 수 있는 상수가 포함됩니다. 이 예제에서는 각 메서드의 코드가 생략되어 있지만 다음 단원에서 모두 자세히 살펴봅니다.

```
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 {}
}
```

이 예제에서는 컴파일러 옵션 -as3이 true로 설정되어 있고 컴파일러 옵션 -es가 false로 설정되어 있다고 가정하므로 네 가지 대체 메서드에서는 모두 public 특성이 아닌 AS3 네임스페이스를 사용합니다. 이는 Adobe Flash Builder 및 AdobeFlashProfessional의 기본 설정입니다.

프로토타입 상속 사용을 선호하는 고급 개발자일 경우 TypedArray 클래스에서 두 가지 사항을 약간 변경하여 컴파일러 옵션 -es를 true로 설정하여 컴파일할 수 있습니다. 첫 번째, override 특성의 모든 항목을 제거하고 AS3 네임스페이스를 public 특성으로 바꿉니다. 두 번째, 네 가지 super 항목을 모두 Array.prototype으로 교체합니다.

TypedArray 생성자

생성자에서 임의의 길이의 인수 목록을 허용해야 하므로 하위 클래스 생성자에서 이와 관련한 문제가 발생할 수 있습니다. 문제는 배열을 만들기 위해 superconstructor로 인수를 전달하는 방법입니다. 인수 목록을 배열로 전달할 경우 superconstructor는 이것을 Array 유형의 단일 인수로 생각하여 결과 배열은 항상 1 요소 길이가 됩니다. 인수 목록 전달을 처리하는 기존의 방법은 Function.apply() 메서드를 사용하는 것이며, 이 메서드는 인수 배열을 두 번째 매개 변수로 취하지만 함수 실행 시에는 이것을 인수 목록으로 변환합니다. 그러나 Function.apply() 메서드는 생성자와 함께 사용할 수 없습니다.

남은 유일한 옵션은 TypedArray 생성자에서 Array 생성자의 논리를 다시 만드는 것입니다. 다음 코드는 Array 클래스 생성자에서 사용되는 알고리즘을 보여 줍니다. 이 알고리즘은 Array 하위 클래스 생성자에서 재사용할 수 있습니다.

```
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]
            }
        }
    }
}
```

TypedArray 생성자는 다음과 같은 네 가지 변경 사항을 제외하고는 Array 생성자의 코드 대부분을 공유합니다. 첫 번째, 매개 변수 목록에 배열 데이터 유형의 지정을 허용하는 Class 유형의 새로운 필수 매개 변수가 포함됩니다. 두 번째, 생성자로 전달되는 데이터 유형이 dataType 변수에 할당됩니다. 세 번째, else 문에서 length 속성 값이 for 루프 뒤에 할당되어 length에는 올바른 유형의 인수만 포함됩니다. 네 번째, for 루프 본문은 push() 메서드의 대체 버전을 사용하므로 올바른 데이터 유형의 인수만 배열에 추가됩니다. 다음 예제에서는 TypedArray 생성자 함수를 보여 줍니다.

```
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;
        }
    }
}
```

TypedArray 대체 메서드

TypedArray 클래스는 배열에 요소를 추가할 수 있는 네 가지 Array 클래스 메서드를 대체합니다. 각각의 경우 대체 메서드는 정확하지 않은 데이터 유형이 요소가 추가되는 것을 막기 위한 유형 검사를 추가합니다. 이렇게 하면 각 메서드가 해당 수퍼 클래스 버전을 호출합니다.

push() 메서드는 for..in 루프를 사용하여 인수 목록을 반복하며 각 인수에서 유형 검사를 수행합니다. 정확하지 않은 유형의 인수는 모두 splice() 메서드를 사용하여 args 배열에서 제거됩니다. for..in 루프가 종료되면 args 배열에는 dataType 유형 값만 포함됩니다. 그런 후 다음 코드와 같이 업데이트된 args 배열로 push()의 수퍼 클래스 버전이 호출됩니다.

```
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));
}
```

concat() 메서드는 passArgs라는 임시 TypedArray를 만들어 유형 검사를 통과한 인수를 저장합니다. 이를 통해 push() 메서드에 있는 유형 검사 코드를 재사용할 수 있습니다. for..in 루프는 args 배열을 반복하며 각 인수에서 push()를 호출합니다. passArgs는 TypedArray 유형이므로 push()의 TypedArray 버전이 실행됩니다. 그러면 concat() 메서드가 다음 코드와 같이 해당 수퍼 클래스 버전을 호출합니다.

```
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));
}
```

splice() 메서드는 임의의 인수 목록을 취하지만 처음 두 개의 인수는 항상 삭제할 인덱스 번호 및 요소의 수를 나타냅니다. 이것이 바로 대체 splice() 메서드가 인덱스 위치 2 이상의 args 배열 요소에 대해서만 유형을 검사하는 이유입니다. 코드에서 눈 여겨 볼 것은 for 루프 내에서 splice()에 대한 재귀 호출이 있는 것처럼 보이지만, args가 TypedArray가 아닌 Array 유형이므로 실제로는 재귀 호출이 아니라는 점입니다. 즉, args.splice()에 대한 호출은 메서드의 수퍼 클래스 버전에 대한 호출입니다. for..in 루프가 끝나면 다음 코드와 같이 args 배열에는 인덱스 위치 2 이상의 정확한 유형의 값만 포함되며 splice()는 해당 수퍼 클래스 버전을 호출합니다.

```
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));
}
```

unshift() 메서드는 배열의 시작 부분에 요소를 추가하며 임의의 인수 목록을 허용합니다. 대체 unshift() 메서드는 다음 예제 코드와 같이 push() 메서드에서 사용한 것과 매우 유사한 알고리즘을 사용합니다.

```
    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));
    }
}
```

배열 예제: PlayList

Flash Player 9 이상, Adobe AIR 1.0 이상

PlayList 예제는 노래 목록을 관리하는 음악 재생 목록 응용 프로그램에서 배열을 사용한 작업에 관련된 기술에 대해 설명합니다. 이러한 기술은 다음과 같습니다.

- 인덱스 배열 만들기
- 인덱스 배열에 항목 추가
- 다른 정렬 옵션을 사용하여 다른 속성별로 객체 배열 정렬
- 문자 구분 문자열로 배열 변환

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. PlayList 응용 프로그램 파일은 Samples/PlayList 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
PlayList.mxml 또는 PlayList.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/playlist/PlayList.as	노래 목록을 나타내는 클래스입니다. Array를 사용하여 목록을 저장하고 목록 항목의 정렬을 관리합니다.
com/example/programmingas3/playlist/Song.as	노래 한 곡에 대한 정보를 나타내는 값 객체. PlayList 클래스에서 관리되는 항목은 Song 인스턴스입니다.
com/example/programmingas3/playlist/SortProperty.as	Song 객체 목록을 정렬할 수 있는 Song 클래스 속성을 나타내는 사용 가능한 값의 의사 열거 항목입니다.

PlayList 클래스 개요

Flash Player 9 이상, Adobe AIR 1.0 이상

PlayList 클래스는 Song 객체 세트를 관리하며, 재생 목록에 노래를 추가(addSong() 메서드)하고 목록에서 노래를 정렬(sortList() 메서드)할 수 있는 기능을 제공하는 공용 메서드를 가지고 있습니다. 또한 이 클래스에는 읽기 전용 접근자 속성인 songList가 포함되며, 이 속성을 통해 재생 목록에 있는 노래 세트에 실제로 액세스할 수 있습니다. 내부적으로 PlayList 클래스는 전용 Array 변수를 사용하여 노래를 추적합니다.

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

노래 목록을 추적하기 위해 PlayList 클래스에서 사용되는 _songs Array 변수 외에도 두 개의 다른 전용 변수를 통해 목록을 정렬해야 하는지 여부(_needToSort) 및 지정된 시간에 정렬해야 하는 노래 목록의 속성(_currentSort)을 추적할 수 있습니다.

다른 객체와 마찬가지로 Array 인스턴스를 선언하는 것은 Array를 만드는 작업에서 반 정도를 완료한 것이며, Array 인스턴스의 속성이나 메서드에 액세스하려면 먼저 PlayList 클래스의 생성자에서 이를 인스턴스화해야 합니다.

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

생성자의 첫 번째 행에서 _songs 변수를 인스턴스화하면 사용 준비가 완료됩니다. 또한 초기 정렬 기준 속성을 설정할 수 있는 sortList() 메서드가 호출됩니다.

목록에 노래 추가

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 응용 프로그램에 새로운 노래를 넣으면 데이터 입력 양식의 코드가 PlayList 클래스의 addSong() 메서드를 호출합니다.

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

addSong() 내에서 _songs 배열의 push() 메서드가 호출되어 addSong()에 전달된 Song 객체가 해당 배열의 새 요소로 추가됩니다. push() 메서드를 사용하면 이전에 적용된 정렬에 관계없이 새 요소가 배열의 끝에 추가됩니다. 즉, push() 메서드가 호출된 후에는 노래 목록이 제대로 정렬되어 있지 않게 되므로 _needToSort 변수가 true로 설정됩니다. 이론적으로는 목록이 지정된 시간에 정렬되었는지 여부를 추적할 필요 없이 sortList() 메서드를 바로 호출할 수 있습니다. 하지만 실제로는 노래 목록을 검색하기 바로 전까지는 노래 목록을 정렬할 필요가 없습니다. 이렇게 정렬 작업을 지연하면, 노래를 검색하기 전에 여러 노래가 목록에 추가되는 경우 등 응용 프로그램에서 불필요한 정렬을 수행하지 않아도 됩니다.

노래 목록 정렬

Flash Player 9 이상, Adobe AIR 1.0 이상

재생 목록에서 관리되는 Song 인스턴스는 복잡한 객체이므로, 응용 프로그램 사용자가 재생 목록을 노래 제목이나 발표 연도와 같은 다른 속성에 따라 정렬하고 싶어할 수 있습니다. PlayList 응용 프로그램에서 노래 목록을 정렬하는 작업은 세 가지 부분으로 나눌 수 있습니다. 첫 번째는 목록을 정렬해야 하는 속성을 식별하는 것이며 두 번째는 해당 속성별로 정렬할 때 사용해야 하는 정렬 옵션을 지정하는 것이며 세 번째는 실제로 정렬 작업을 수행하는 것입니다.

정렬 속성

Song 객체는 노래 제목, 아티스트, 발표 연도, 파일 이름 및 사용자가 선택한 노래 장르 등을 비롯한 여러 속성을 추적합니다. 이 중, 처음 세 가지만 정렬에 사용할 수 있습니다. 개발자의 편의에 따라 이 예제에는 정렬 시 사용 가능한 속성을 나타내는 값이 나열된 SortProperty 클래스가 포함됩니다.

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

SortProperty 클래스에는 세 가지 상수 즉, TITLE, ARTIST 및 YEAR가 포함되어 있으며, 이러한 상수는 각각 정렬 시 사용할 수 있는 연관된 Song 클래스 속성의 실제 이름이 포함된 문자열을 저장합니다. 나머지 코드에서 정렬 속성이 지정될 때마다 이 열거 항목을 사용하여 정렬이 수행됩니다. 예를 들어 PlayList 생성자에서 목록은 초기에는 다음과 같이 sortList() 메서드를 호출하여 정렬됩니다.

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

정렬 속성이 SortProperty.TITLE로 지정되므로 노래가 제목에 따라 정렬됩니다.

속성순 정렬 및 정렬 옵션 지정

노래 목록을 실제로 정렬하는 작업은 다음과 같이 sortList() 메서드의 PlayList 클래스에 의해 수행됩니다.

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

제목이나 아티스트순으로 정렬할 경우 알파벳순으로 정렬하는 것이 좋은 방법이지만 연도순으로 정렬할 경우에는 숫자 정렬이 가장 적합합니다. switch 문은 적합한 정렬 옵션을 정의하는 데 사용되며 sortProperty 매개 변수에 지정된 값에 따라 sortOptions 변수에 저장됩니다. 여기서 다시 하드 코딩된 값이 아닌 지정된 열거 항목이 속성을 구별하는 데 사용됩니다.

결정된 정렬 속성 및 정렬 옵션을 사용하여 sortOn() 메서드를 호출하고 매개 변수로 두 개의 값을 전달하면 _songs 배열이 실제로 정렬됩니다. 그러면 현재 정렬된 노래 목록에 따라 현재 정렬 속성이 기록됩니다.

문자 구분 문자열로 배열 요소 결합

Flash Player 9 이상, Adobe AIR 1.0 이상

PlayList 클래스의 노래 목록을 관리하기 위해 배열이 사용되는 것 이외에도 이 예제에서는 지정된 노래가 속한 장르 목록을 관리하기 위해 Song 클래스에서도 배열이 사용됩니다. Song 클래스 정의에서 다음 코드를 살펴보십시오.

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

새로운 Song 인스턴스를 만들 때 노래가 속한 장르를 지정하는 데 사용되는 genres 매개 변수는 Array 인스턴스로 정의됩니다. 이렇게 하면 여러 장르를 생성자로 전달할 수 있는 하나의 변수로 묶을 때 편리합니다. 하지만 내부적으로 Song 클래스는 전용 _genres 변수에서 장르를 세미콜론으로 구분된 String 인스턴스로 관리합니다. Array 매개 변수는 리터럴 문자열 값 ";"을 지정된 구분 기호로 하여 join() 메서드를 호출함으로써 세미콜론으로 구분된 문자열로 변환됩니다.

마찬가지로, genres 접근자를 통해 장르를 Array로 설정하거나 검색할 수 있습니다.

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

genresset 접근자는 생성자와 정확히 동일하게 동작합니다. 즉, Array를 허용하며 join() 메서드를 호출하여 이것을 세미콜론으로 구분된 문자열로 변환합니다. 반면 get 접근자는 이와 반대의 작업을 수행합니다. 즉 _genres 변수의 split() 메서드를 호출하여 지정된 구분 기호(앞서 언급한 리터럴 문자열 값인 ";")를 사용하여 String을 값의 배열로 분할합니다.

4장: 오류 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

오류 "처리"란 오류에 응답하거나 오류를 수정할 수 있는 논리를 응용 프로그램에 작성하는 것입니다. 오류는 응용 프로그램이 컴파일될 때 또는 컴파일된 응용 프로그램이 실행될 때 발생합니다. 응용 프로그램에서 오류를 처리하는 경우 응답이 없거나 오류가 발생한 프로세스가 자동으로 실패하지 않고, 오류 발생에 대한 응답으로 **특정 작업**이 수행됩니다. 정확하게 사용한다면 오류 처리를 통해 예기치 못한 비헤이비어로부터 응용 프로그램 및 사용자를 보호할 수 있습니다.

하지만 오류 처리는 컴파일 중이나 응용 프로그램이 실행되는 동안 throw되는 수많은 오류에 대한 응답을 비롯한 광범위한 범주를 포함하고 있습니다. 여기에서는 런타임 오류 처리 방법(응용 프로그램 실행 중 throw되는 오류), 발생할 수 있는 다양한 오류 유형 및 ActionScript 3.0에서 제공되는 오류 처리 시스템의 장점에 대해 중점적으로 살펴봅니다.

오류 처리의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

런타임 오류는 ActionScript 코드에 문제가 발생하여 ActionScript 내용이 의도한 대로 실행되지 않는 것입니다. ActionScript 코드가 올바르게 실행되도록 하려면 응용 프로그램 내에서 오류를 처리하는 코드를 작성합니다. 이 코드는 오류를 수정 및 해결하거나 적어도 사용자에게 오류가 발생했음을 알리는 역할을 합니다. 이 프로세스가 바로 **오류 처리** 프로세스입니다.

오류 처리는 컴파일 중이나 응용 프로그램이 실행되는 동안 throw되는 수많은 오류에 대한 응답 등이 포함된 광범위한 범주입니다. 컴파일 작업 시 발생하는 오류는 쉽게 확인되는 경우가 많습니다. SWF 파일 생성 프로세스를 완료하기 위해서는 이 오류를 수정합니다.

런타임 오류는 오류 코드를 실제로 실행할 경우에만 발생하기 때문에 감지하기가 쉽지 않습니다. if..then..else 문처럼 프로그램의 한 세그먼트에 여러 개의 코드 분기가 있을 경우, 코드에 오류가 없음을 확인하기 위해 실제 사용자가 입력할 수 있는 가능한 모든 입력 값을 사용하여 가능한 모든 조건을 테스트합니다.

런타임 오류는 두 가지 범주로 구분할 수 있습니다. **프로그램 오류**는 메서드 매개 변수에 잘못된 데이터 유형을 지정하는 것과 같은 ActionScript 코드 내 오류를 뜻하며, **논리 오류**는 자금 관리용 프로그램에서 잘못된 공식으로 이율을 계산하는 경우 등의 프로그램의 논리(데이터 확인 및 값 조작) 오류입니다. 다시 말하지만, 이 두 유형의 오류 모두 대개 응용 프로그램을 꼼꼼히 테스트함으로써 사전에 감지하고 수정할 수 있습니다.

가장 이상적인 것은 최종 사용자에게 응용 프로그램을 릴리스하기 전에 모든 오류를 식별하여 제거하는 것이지만, 모든 오류를 사전에 발견하거나 예방하는 것은 불가능합니다. 예를 들어 ActionScript 응용 프로그램이 프로그래머가 통제할 수 없는 특정 웹 사이트로부터 정보를 불러온다고 가정해 보겠습니다. 이 경우 특정 시점에 해당 웹사이트에 접근할 수 없으면 외부 데이터에 의존하고 있는 응용 프로그램의 일부가 제대로 동작하지 않을 수 있습니다. 오류 처리에서 가장 중요한 부분은 이러한 알려지지 않은 경우에 대비하여 이를 효과적으로 처리할 수 있도록 하는 것입니다. 사용자가 응용 프로그램을 계속 사용할 수 있거나 적어도 왜 작동하지 않는지에 대한 친절한 오류 메시지가 표시되어야 합니다.

런타임 오류는 ActionScript에서 다음과 같은 두 가지 방법으로 나타납니다.

- Error 클래스: 상당수의 오류는 해당 오류와 연관된 Error 클래스가 있습니다. 오류가 발생하면 Flash 런타임(Flash Player 또는 Adobe AIR)에서 해당 오류와 연관된 특정 Error 클래스의 인스턴스를 만듭니다. 그러면 코드에서 오류 객체에 포함된 정보를 확인한 후 해당 오류에 대해 적절히 응답합니다.
- 오류 이벤트: Flash 런타임에서 정상적으로 이벤트를 트리거하는 경우에도 오류가 발생할 때가 있습니다. 이러한 경우 오류 이벤트가 대신 트리거됩니다. 각 오류 이벤트에는 연관된 클래스가 있으며, Flash 런타임은 해당 클래스의 인스턴스를 오류 이벤트를 구독하는 메서드로 전달합니다.

특정 메서드에서 오류 또는 오류 이벤트를 트리거할 수 있는지 알아보려면 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 메서드 항목을 확인하십시오.

중요한 개념 및 용어

다음 참조 목록에는 오류 처리 루틴을 프로그래밍하는 데 중요한 용어가 나열되어 있습니다.

비동기 즉각적인 결과를 제공하지 않는 프로그램 명령(예: 메서드 호출)으로, 대신 결과 또는 오류가 이벤트 형태로 제공됩니다.

catch 예외(런타임 오류) 발생 시 코드에서 해당 예외를 인식하면 코드가 예외를 **catch**했다고 합니다. 예외가 catch되면 Flash 런타임에서 다른 ActionScript 코드에 예외를 알리는 작업이 중단됩니다.

디버거 버전 사용자에게 런타임 오류를 알리는 코드가 포함된 특수 버전의 Flash 런타임(Flash Player 디버거 버전 또는 ADL(AIR Debug Launcher))입니다. 가장 많이 사용되는 Flash Player 또는 Adobe AIR 표준 버전에서 ActionScript 코드에 의해 처리되지 않은 오류는 무시됩니다. 디버거 버전(Adobe Flash CS4 Professional 및 Adobe Flash Builder와 함께 제공됨)에서는 처리되지 않은 오류가 발생할 경우 경고 메시지가 나타납니다.

예외 응용 프로그램이 실행되는 동안 발생하여 Flash 런타임에서 자체적으로 해결할 수 없는 오류를 나타냅니다.

Re-throw 코드에서 예외를 catch하면 Flash 런타임에서는 예외를 더 이상 다른 객체에 알리지 않습니다. 예외를 다른 객체에서 수신하는 것이 중요할 경우 코드에서 해당 예외를 **re-throw**하여 알림 프로세스를 다시 시작해야 합니다.

동기 즉각적인 결과가 제공되거나 오류를 즉시 throw하는 프로그램 명령(예: 메서드 호출)으로, 동일한 코드 블록 내에서 해당 응답을 사용할 수 있습니다.

throw Flash 런타임을 비롯하여 결과적으로 다른 객체 및 ActionScript 코드에 오류가 발생했음을 알리는 동작을 오류를 **throw**한다고 합니다.

오류 유형

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램을 개발하고 실행할 때 다양한 오류 유형 및 오류 용어를 접하게 됩니다. 다음 목록은 주요 오류 유형 및 용어에 대한 설명입니다.

- 컴파일 시간 오류: 코드 컴파일 중 ActionScript 컴파일러에서 발생합니다. 컴파일 시간 오류는 코드의 구문 문제로 인해 응용 프로그램을 제대로 빌드하지 못할 때 발생합니다.
- 런타임 오류: 컴파일 후 응용 프로그램을 실행할 때 발생합니다. 런타임 오류는 SWF 파일이 Flash 런타임(Adobe Flash Player 또는 Adobe AIR)에서 재생될 때 발생하는 오류를 나타냅니다. 대부분의 경우 런타임 오류가 발생하면 사용자에게 오류 내용을 보고하고 응용 프로그램 실행을 유지하기 위한 절차를 수행하여 오류를 처리합니다. 원격 웹사이트에 연결할 수 없거나 필수 데이터를 로드할 수 없는 등 치명적 오류일 경우 오류 처리를 사용하여 응용 프로그램을 적절하게 종료할 수 있습니다.
- 동기 오류: 함수 호출 시 발생하는 런타임 오류입니다. 예를 들어 특정 메서드를 사용하려고 할 때 이 메서드로 전달한 인수가 유효하지 않은 경우 Flash 런타임에서 예외를 throw합니다. 대부분의 오류는 명령문 실행 시 동기적으로 발생하며, 가장 적합한 catch 문으로 제어의 흐름이 즉시 전달됩니다.

  예를 들어 다음 코드 예제에서는 프로그램에서 파일 업로드를 시도하기 전에 browse() 메서드가 호출되지 않아 런타임 오류가 발생합니다.

```
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.
}
```

이 경우 Flash Player에서 파일 업로드를 시도하기 전에 browse() 메서드가 호출되지 않았음을 확인하므로 런타임 오류가 동기적으로 발생합니다.

동기 오류 처리에 대한 자세한 내용은 50페이지의 "응용 프로그램에서 동기 오류 처리"를 참조하십시오.

- **비동기오류**는 정상적인 프로그램 흐름을 벗어나 발생하는 런타임 오류로, 이벤트를 발생시키고 이벤트 리스너는 이를 catch합니다. 비동기 작업은 함수에서 작업을 시작하지만 완료될 때까지 기다리지 않는 작업입니다. 이때 오류 이벤트 리스너를 만들어 응용 프로그램이나 사용자가 특정 작업을 시도할 때까지 기다릴 수 있으며 작업에 실패할 경우 이벤트 리스너로 오류를 catch하여 오류 이벤트에 응답할 수 있습니다. 그러면 이벤트 리스너는 이벤트 핸들러 함수를 호출하여 오류 이벤트에 적절하게 응답합니다. 예를 들어 이벤트 핸들러는 사용자에게 오류를 해결하도록 하는 대화 상자를 표시할 수 있습니다.

  앞에서 설명했던 파일 업로드 동기 오류를 살펴봅니다. 파일 업로드를 시작하기 전에 browse() 메서드를 성공적으로 호출한 경우 Flash Player에서 여러 이벤트를 전달하게 됩니다. 예를 들어 업로드가 시작되면 open 이벤트가 전달됩니다. 그리고 파일 업로드 작업이 성공적으로 완료되면 complete 이벤트가 전달됩니다. 이벤트 처리는 비동기적이므로(즉, 특정 시간, 알려진 시간 또는 미리 지정된 시간에 발생하지 않음) 다음 코드와 같이 이러한 특정 이벤트를 수신하려면 addEventListener() 메서드를 사용합니다.

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

  비동기 오류 처리에 대한 자세한 내용은 55페이지의 "오류 이벤트 및 상태에 응답"을 참조하십시오.

- **catch되지 않는 예외**: 오류에 응답하기 위한 해당 논리(예: catch 문)가 없을 때 발생하는 오류입니다. 응용 프로그램에서 오류가 발생하고 현재 또는 상위 수준에 오류를 처리할 수 있는 적합한 catch 문이나 이벤트 핸들러가 없는 경우 이 오류는 catch되지 않는 예외로 간주됩니다.

  catch되지 않는 오류가 발생하면 런타임이 uncaughtError 이벤트를 전달합니다. 이 이벤트를 "전역 오류 핸들러"라고도 합니다. 이 이벤트는 LoaderInfo.uncaughtErrorEvents 속성을 통해 사용 가능한 SWF의 UncaughtErrorEvents 객체에 의해 전달됩니다. uncaughtError 이벤트에 대해 등록된 리스너가 없는 경우 런타임이 catch되지 않는 오류를 무시하고 오류로 인해 SWF가 중지되지 않는 한 실행을 계속하려고 시도합니다.

디버거 버전의 Flash 런타임은 uncaughtError 이벤트를 전달하는 것 외에도 현재 스크립트를 종료하여 catch되지 않는 오류에 응답합니다. 그런 다음 catch되지 않는 오류를 trace 명령문 출력에 표시하거나 오류 메시지를 로그 파일에 기록합니다. 예외 객체가 Error 클래스 또는 해당 하위 클래스 중 하나의 인스턴스인 경우 출력에 스택 추적 정보도 표시됩니다. Flash 런타임의 디버거 버전 사용에 대한 자세한 내용은 49페이지의 "Flash 런타임의 디버거 버전 작업"을 참조하십시오.

참고: uncaughtError 이벤트를 처리하는 동안 uncaughtError 핸들러에서 error 이벤트가 발생하면 이벤트 핸들러가 여러 번 호출됩니다. 이 경우 예외가 무한 반복되므로 이러한 상황이 일어나지 않도록 주의해야 합니다.

ActionScript 3.0에서 오류 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

많은 응용 프로그램이 오류 처리를 위한 논리 작성 없이도 실행될 수 있으므로 개발자들은 응용 프로그램에 대한 오류 처리 관련 작업을 뒤로 미루는 경향이 있습니다. 하지만 오류 처리가 없으면 예상대로 작업이 수행되지 않을 경우 응용 프로그램이 쉽게 멈추거나 사용자 작업을 방해할 수 있습니다. ActionScript 2.0에는 사용자가 논리를 사용자 정의 함수에 작성하여 특정 메시지와 함께 예외를 발생시킬 수 있는 Error 클래스가 있습니다. 오류 처리는 사용자에게 친숙한 응용 프로그램을 만들기 위해 매우 중요하므로 ActionScript 3.0에는 오류를 catch하기 위한 확장된 아키텍처가 포함되어 있습니다.

참고: Adobe Flash Platform용 ActionScript 3.0 참조 설명서에는 여러 메서드에서 throw되는 예외가 설명되지만 각 메서드에 대해 가능한 모든 예외가 포함되지 않을 수 있습니다. 메서드 설명 부분에 메서드에서 발생하는 일부 예외가 나열되어 있는 경우에도 이 설명에서 명확하게 언급하지 않은 다른 문제나 구문 오류에 의해 메서드에서 예외가 발생할 수 있습니다.

ActionScript 3.0 오류 처리 요소

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 다음을 비롯하여 오류 처리를 위한 여러 도구가 포함되어 있습니다.

- Error 클래스: ActionScript 3.0에는 광범위한 Error 클래스가 포함되어 오류 객체를 생성할 수 있는 상황 범위를 확장합니다. 각 Error 클래스는 시스템 오류(예: MemoryError의 경우), 코딩 오류(예: ArgumentError의 경우), 네트워킹 및 통신 오류(예: URIError의 경우) 등의 특정 오류 또는 기타 상황을 응용 프로그램에서 처리하고 응답할 수 있도록 도와 줍니다. 각 클래스에 대한 자세한 내용은 58페이지의 "Error 클래스 비교"를 참조하십시오.
- 자동 실패 감소: 이전 Flash Player 버전에서는 명시적으로 throw 문을 사용한 경우에만 오류가 생성되고 보고되었습니다. Flash Player 9 이상의 경우 Flash 런타임, 기본 ActionScript 메서드 및 속성은 런타임 오류를 throw합니다. 이러한 오류에서 사용자는 오류가 발생할 때 예외를 보다 효과적으로 처리하고 각 예외에 대해 개별적인 방식으로 반응할 수 있습니다.
- 디버깅 시 명확한 오류 메시지 표시: Flash 런타임의 디버거 버전을 사용하는 경우 문제가 될 수 있는 코드나 상황에서 확실한 오류 메시지가 발생되며 이를 통해 특정 코드 블록이 실패하는 이유를 쉽게 파악할 수 있습니다. 이러한 메시지는 오류 수정을 보다 효율적으로 수행할 수 있게 해줍니다. 자세한 내용은 49페이지의 "Flash 런타임의 디버거 버전 작업"을 참조하십시오.
- 정확한 오류를 통해 사용자에게 명확한 오류 메시지 표시: 이전 버전의 Flash Player에서는 upload() 호출이 실패할 경우 FileReference.upload() 메서드에서 5가지 가능한 오류 중 하나를 나타내는 false 부울 값을 반환했습니다. ActionScript 3.0에서 upload() 메서드 호출 시 오류가 발생하면 4가지 특정 오류를 통해 최종 사용자에게 보다 정확한 오류 메시지를 표시할 수 있습니다.
- 개선된 오류 처리: 많은 공통적인 상황에서 서로 다른 오류가 발생합니다. 예를 들어 ActionScript 2.0에서는 FileReference 객체가 채워지기 전에는 name 속성이 null 값을 가지고 있습니다. 따라서 name 속성을 사용하거나 표시하기 전에, 값이 설정되었고 null이 아닌지 확인합니다. ActionScript 3.0에서는 FileReference 객체가 채워지기 전에 name 속성에 액세스하려고 할 경우 Flash Player 또는 AIR에서 IllegalOperationError를 발생시켜, 값이 설정되지 않았으며 try..catch..finally 블록을 사용하여 오류를 처리할 수 있음을 알려 줍니다. 자세한 내용은 50페이지의 "try..catch..finally 문"을 참조하십시오.

- 성능상의 중대한 단점 없음: try..catch..finally 블록을 사용하면 이전 버전의 ActionScript에 비해 리소스를 거의 추가로 사용하지 않고도 오류를 처리할 수 있습니다.
- ErrorEvent 클래스를 통해 특정 비동기 오류 이벤트에 대한 리스너 작성 가능: 자세한 내용은 55페이지의 "오류 이벤트 및 상태에 응답"을 참조하십시오.

오류 처리 전략

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램에 문제 상황이 발생하지 않는 한 오류 처리 논리를 코드에 작성하지 않아도 응용 프로그램을 성공적으로 실행할 수 있습니다. 하지만 오류를 적극적으로 처리하지 않고 응용 프로그램에 문제가 발생한 경우 사용자는 문제 발생 시 원인을 알 수 없습니다.

여러 가지 방법을 사용하여 응용 프로그램에서 오류를 처리할 수 있으며, 다음 목록에는 오류를 처리하는 세 가지 주요 옵션이 요약되어 있습니다.

- try..catch..finally 문 사용: 이러한 명령문은 동기 오류 발생 시 오류를 catch합니다. 계층으로 명령문을 중첩하여 다양한 코드 실행 수준에서 예외를 catch할 수 있습니다. 자세한 내용은 50페이지의 "try..catch..finally 문"을 참조하십시오.
- 고유한 사용자 정의 오류 객체 만들기: Error 클래스를 사용하여 고유의 사용자 정의 오류 객체를 만들어 내장된 오류 유형에 포함되지 않는 응용 프로그램의 특정 작업을 추적할 수 있습니다. 그런 다음 사용자 정의 오류 객체에 try..catch..finally 문을 사용할 수 있습니다. 자세한 내용은 54페이지의 "사용자 정의 오류 클래스 만들기"를 참조하십시오.
- 오류 이벤트에 응답할 이벤트 리스너 및 핸들러 작성: 이 전략을 사용하면 try..catch..finally 블록에서 많은 코드를 중복하지 않고도 유사한 이벤트를 처리할 수 있는 전역 오류 핸들러를 만들 수 있습니다. 이 전략을 사용하여 비동기 오류를 catch할 수도 있습니다. 자세한 내용은 55페이지의 "오류 이벤트 및 상태에 응답"을 참조하십시오.

Flash 런타임의 디버거 버전 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe에서는 개발자의 디버깅 작업을 지원하기 위해 특별 버전의 Flash 런타임을 제공합니다. Adobe Flash Professional 또는 Adobe Flash Builder를 설치할 때 Flash Player의 디버거 버전을 받을 수 있습니다. 이러한 도구 중 하나를 설치할 때 또는 Adobe AIR SDK의 일부로 ADL이라는 Adobe AIR 응용 프로그램의 디버깅 유틸리티를 받을 수도 있습니다.

Flash Player 및 Adobe AIR의 디버거 버전과 릴리스 버전에서 오류를 표시하는 방법에는 확연한 차이가 있습니다. 디버거 버전에는 오류 유형(예: 일반 오류, IOError 또는 EOFError), 오류 번호 및 사용자가 쉽게 이해할 수 있는 오류 메시지가 표시됩니다. 반면 릴리스 버전에는 오류 유형과 오류 번호만 표시됩니다. 예를 들어 다음과 같은 코드를 살펴봅니다.

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

Flash Player의 디버거 버전에서 readBoolean() 메서드가 EOFError를 발생시키면 tf 텍스트 필드에 "EOFError: Error #2030: End of file was encountered."라는 메시지가 표시됩니다.

Flash Player 또는 Adobe AIR 릴리스 버전에서는 동일한 코드에서 "EOFError: Error #2030"이라는 텍스트가 표시됩니다.

참고: 디버거 플레이어는 "allComplete" 이벤트를 브로드캐스팅하여 "allComplete" 이름을 갖는 사용자 정의 이벤트를 생성하지 않도록 방지합니다. 그렇지 않으면 디버깅 시 예기치 않은 비헤이비어가 발생할 수 있습니다.

릴리스 버전에서는 리소스 및 크기를 최소화하기 위해 오류 메시지 문자열이 표시되지 않습니다. 대신 설명서(Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 부록)에서 오류 번호와 관련된 오류 메시지를 찾아볼 수 있습니다. 또는 전체 메시지를 확인하기 위해 Flash Player 및 AIR의 디버거 버전을 사용하여 오류를 다시 생성할 수 있습니다.

응용 프로그램에서 동기 오류 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

가장 일반적인 오류 처리는 동기 오류 처리 논리이며, 응용 프로그램이 실행되는 동안 동기 오류를 catch하기 위해 코드에 명령문을 삽입합니다. 이러한 유형의 오류 처리를 통해 함수가 실패할 경우 응용 프로그램에 런타임 오류를 알리고 복구할 수 있습니다. 동기 오류를 catch하기 위한 논리에는 try..catch..finally 문이 포함됩니다. 이 문은 글자 그대로 작업을 시도하고 Flash 런타임에서 모든 오류 응답을 catch한 다음 실패한 작업을 처리하기 위해 다른 작업을 실행합니다.

try..catch..finally 문

Flash Player 9 이상, Adobe AIR 1.0 이상

동기 런타임 오류가 발생하면 try..catch..finally 문을 사용하여 오류를 catch합니다. 런타임 오류가 발생하면 Flash 런타임에서 예외를 throw하여 정상적인 실행이 중지되고 Error 유형의 특수 객체가 만들어집니다. 그런 다음 이 Error 객체는 첫 번째 사용 가능한 catch 블록에 전달됩니다.

try 문에는 오류가 발생할 가능성이 있는 명령문이 포함됩니다. 따라서 try 문에는 항상 catch 문을 함께 사용합니다. try 문 블록의 명령문 중 하나에서 오류가 감지되면 해당 try 문에 연결된 catch 문이 실행됩니다.

finally 문에는 try 블록에서 오류가 발생하는지 여부와 관계없이 실행되는 명령문이 포함됩니다. 오류가 없으면 finally 블록 내의 명령문이 try 블록 명령문이 완료된 후 실행됩니다. 오류가 있으면 먼저 적합한 catch 문이 실행되고 finally 블록의 명령문이 뒤따라 실행됩니다.

다음 코드에서는 try..catch..finally 문을 사용하는 구문을 보여 줍니다.

```
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.
}
```

각 catch 문은 명령문이 처리하는 특정 예외 유형을 식별합니다. catch 문은 Error 클래스의 하위 클래스인 오류 클래스만 지정할 수 있으며, 각 catch 문은 순서대로 검사됩니다. 오류 발생 유형과 일치하는 첫 번째 catch 문만 실행됩니다. 즉, 먼저 상위 수준의 Error 클래스를 검사한 다음 Error 클래스의 하위 클래스를 검사할 경우 상위 수준의 Error 클래스에서만 일치가 발생합니다. 다음 코드는 이런 상황에 대한 예입니다.

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

이전 코드는 다음을 출력합니다.

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

ArgumentError를 정확하게 catch하려면 다음 코드와 같이 가장 세부적인 오류 유형을 먼저 나열하고 보다 일반적인 오류 유형을 그 다음에 나열합니다.

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

ActionScript API의 여러 메서드 및 속성은 실행 시 오류가 발생하면 런타임 오류를 throw합니다. 예를 들어 다음 코드와 같이 메서드에서 오디오 스트림을 닫을 수 없는 경우 Sound 클래스의 close() 메서드에서 IOError를 throw합니다.

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

Adobe Flash Platform용 ActionScript 3.0 참조 설명서에 익숙해지면 각 메서드 설명에 자세히 나와 있듯이 어떤 메서드에서 예외를 throw하는지 확인할 수 있습니다.

throw 문

Flash Player 9 이상, Adobe AIR 1.0 이상

실행 중인 응용 프로그램에서 오류가 발생하면 Flash 런타임에서 예외를 throw합니다. 또한 사용자가 직접 throw 문을 사용하여 예외를 명시적으로 발생시킬 수 있습니다. 오류를 명시적으로 발생시킬 경우 Adobe에서는 Error 클래스나 그 하위 클래스의 인스턴스를 생성할 것을 권장합니다. 다음 코드는 오류가 발생한 후 응답하기 위해 Error 클래스 인스턴스인 MyErr를 생성하고 myFunction() 함수를 호출하는 throw 문을 보여 줍니다.

```
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.
}
```

가장 세부적인 데이터 유형이 먼저 나열되도록 catch 문이 정렬되어 있습니다. Number 데이터 유형에 대한 catch 문이 먼저 나열된 경우 uint 데이터 유형에 대한 catch 문이나 int 데이터 유형에 대한 catch 문이 모두 실행되지 않습니다.

참고: Java 프로그래밍 언어에서는 예외를 발생시킬 수 있는 각 함수의 경우, 함수 선언에 연결된 throws 절에서 발생할 수 있는 예외 클래스를 나열하여 이를 선언해야 합니다. ActionScript에서는 함수에서 발생하는 예외를 선언하지 않아도 됩니다.

간단한 오류 메시지 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

새로운 예외 및 오류 이벤트 모델의 가장 큰 장점 중 하나는 사용자에게 액션이 실패한 시기 및 이유를 알려줄 수 있도록 한다는 것입니다. 즉, 메시지를 표시할 코드를 작성하고 그에 따른 응답 옵션을 제공할 수 있습니다.

다음 코드는 텍스트 필드에 오류를 표시하기 위한 간단한 try..catch 문을 보여 줍니다.

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

ActionScript 3.0은 다양한 오류 클래스 및 내장 컴파일러 오류를 사용하여 ActionScript의 이전 버전보다 작업 실패 이유에 대한 보다 자세한 정보를 제공합니다. 이 정보를 참고하면 보다 나은 오류 처리를 통해 응용 프로그램을 안정적으로 작성할 수 있습니다.

오류 rethrow

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램 작성 시 오류를 올바르게 처리할 수 없는 경우 오류를 rethrow해야 하는 상황이 있을 수 있습니다. 예를 들어 다음 코드는 중첩된 try..catch 블록에서 오류를 처리할 수 없는 경우 사용자 정의 ApplicationError를 rethrow하는 중첩된 catch 블록을 보여 줍니다.

```
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);
}
```

이전 코드의 출력은 다음과 같습니다.

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

중첩된 try 블록은 이후 catch 블록에서 catch한 사용자 정의 ApplicationError 오류를 발생시킵니다. 이 중첩된 catch 블록에서 오류를 처리하려고 시도할 수 있으며 실패할 경우 이 블록을 포함하고 있는 try..catch 블록에 ApplicationErrorblock 객체를 throw합니다.

사용자 정의 오류 클래스 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript에서 표준 Error 클래스 중 하나를 확장하여 고유한 오류 클래스를 만들 수 있습니다. 고유한 오류 클래스를 만드는 데는 다음과 같은 여러 이유가 있습니다.

- 응용 프로그램에 고유한 특정 오류나 오류 그룹을 식별하기 위해 만듭니다.

  예를 들어 Flash 런타임에서 트랩된 오류뿐만 아니라 자체적으로 작성한 코드에서 throw된 오류에 대해 다른 액션을 취할 수 있습니다. 이 경우 Error 클래스의 하위 클래스를 만들어 try..catch 블록에서 새로운 오류 데이터 유형을 추적할 수 있습니다.

- 응용 프로그램에서 생성된 오류에 대해 고유한 오류 표시 기능을 제공하기 위해 만듭니다.

  예를 들어 특정 방식으로 오류 메시지를 포맷하는 새로운 toString() 메서드를 만들 수 있습니다. 오류 코드를 사용하여 사용자 언어 환경 설정을 기준으로 올바른 메시지를 검색하는 lookupErrorString() 메서드를 정의할 수도 있습니다.

고유한 오류 메시지는 핵심 ActionScript Error 클래스를 확장해야 합니다. 다음은 Error 클래스를 확장하는 고유한 AppError 클래스에 대한 예제입니다.

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

다음은 프로젝트에서 AppError를 사용하는 예제를 보여 줍니다.

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

참고: 하위 클래스에서 Error.toString() 메서드를 대체하려면 하나의 ...(rest) 매개 변수를 제공합니다. ActionScript 3.0의 기반이 되는 ECMAScript 언어 사양에서는 이런 방식으로 Error.toString() 메서드가 정의되며 ActionScript 3.0에서는 역호환성을 위해 이 메서드를 동일한 방식으로 정의합니다. 그러므로 Error.toString() 메서드를 대체할 경우 매개 변수를 정확하게 일치시킵니다. 런타임에 toString() 메서드에 다른 매개 변수를 전달해도 이 매개 변수는 무시됩니다.

오류 이벤트 및 상태에 응답

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0 오류 처리에서 가장 주목할 만한 개선 사항 중 하나는 응용 프로그램이 실행되는 동안 비동기 오류에 응답하기 위한 오류 이벤트 처리가 지원된다는 것입니다. 비동기 오류에 대한 정의는 46페이지의 “오류 유형”을 참조하십시오.

이벤트 리스너 및 이벤트 핸들러를 만들어 오류 이벤트에 응답할 수 있습니다. 많은 클래스에서 다른 이벤트를 전달하는 것과 같은 방식으로 오류 이벤트를 전달합니다. 예를 들어 XMLSocket 클래스의 인스턴스는 일반적으로 Event.CLOSE, Event.CONNECT, DataEvent.DATA의 세 가지 유형의 이벤트를 전달합니다. 하지만 문제 발생 시 XMLSocket 클래스는 IOErrorEvent.IOError 또는 SecurityErrorEvent.SECURITY_ERROR를 전달할 수 있습니다. 이벤트 리스너 및 이벤트 핸들러에 대한 자세한 내용은 113페이지의 “이벤트 처리”를 참조하십시오.

오류 이벤트는 다음 두 범주 중 하나에 해당됩니다.

- ErrorEvent 클래스를 확장하는 오류 이벤트

  flash.events.ErrorEvent 클래스에는 실행 중인 응용 프로그램의 네트워킹 및 통신 작업과 관련한 오류를 관리하기 위한 속성 및 메서드가 포함됩니다. AsyncErrorEvent, IOErrorEvent, SecurityErrorEvent 클래스는 ErrorEvent 클래스를 확장합니다. Flash 런타임의 디버거 버전을 사용하는 경우에는 런타임에 대화 상자가 표시되어 플레이어에서 발생하는 리스너 함수가 없는 모든 오류 이벤트를 알립니다.

- 상태 기반 오류 이벤트

  상태 기반 오류 이벤트는 네트워킹 및 통신 클래스의 netStatus 및 status 속성과 관련됩니다. Flash 런타임에서 데이터를 읽거나 쓸 때 문제가 발생하면 사용 중인 클래스 객체에 따라 netStatus.info.level 또는 status.level 속성의 값이 "error" 값으로 설정됩니다. 이벤트 핸들러 함수에서 level 속성에 "error" 값이 포함되었는지 여부를 확인하여 이 오류에 응답합니다.

오류 이벤트를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ErrorEvent 클래스 및 그 하위 클래스에는 데이터를 읽거나 쓸 때 Flash 런타임에서 전달한 오류 처리를 위한 오류 유형이 포함됩니다.

다음 예제는 try..catch 문 및 오류 이벤트 핸들러를 모두 사용하여 로컬 파일을 읽으려고 할 때 감지된 모든 오류를 표시합니다. 보다 정교한 처리 코드를 추가하여 사용자에게 옵션을 제공하거나 "your error-handling code here" 주석이 표시된 위치에서 자동으로 오류를 처리할 수 있습니다.

```
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
        }
    }
}
```

상태 변경 이벤트를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash 런타임은 응용 프로그램이 실행되는 동안 level 속성을 지원하는 클래스에 대해 netStatus.info.level 또는 status.level 속성 값을 동적으로 변경합니다. netStatus.info.level 속성이 있는 클래스는 NetConnection, NetStream 및 SharedObject이며, status.level 속성이 있는 클래스는 HTTPStatusEvent, Camera, Microphone 및 LocalConnection입니다. 핸들러 함수를 작성하여 level 값의 변경에 응답하고 통신 오류를 추적할 수 있습니다.

다음 예제는 netStatusHandler() 함수를 사용하여 level 속성의 값을 테스트합니다. level 속성이 오류가 발생했음을 나타내는 경우 코드에서 "Video stream failed" 메시지를 생성합니다.

```
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);
        }
    }
}
```

Error 클래스 비교

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript에서는 미리 정의된 여러 가지 Error 클래스가 제공됩니다. 하지만 자체적으로 작성한 코드에서 동일한 Error 클래스를 사용할 수도 있습니다. ActionScript 3.0에는 두 가지 주요 Error 클래스 즉, ActionScript 핵심 Error 클래스 및 flash.error 패키지 Error 클래스가 있습니다. flash.error 패키지에는 ActionScript 3.0 응용 프로그램 개발 및 디버깅을 지원하기 위한 추가 클래스가 포함됩니다.

핵심 Error 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

핵심 Error 클래스에는 Error, ArgumentError, EvalError, RangeError, ReferenceError, SecurityError, SyntaxError, TypeError, URIError 및 VerifyError 클래스가 포함됩니다. 이 클래스는 각각 최상위 네임스페이스에 있습니다.

클래스 이름	설명	참고 사항
Error	Error 클래스는 예외 발생을 위한 것이며 ECMAScript(EvalError, RangeError, ReferenceError, SyntaxError, TypeError 및 URIError)에서 정의된 다른 예외 클래스에 대한 기본 클래스입니다.	Error 클래스는 모든 런타임 오류에 대해 기본 클래스 역할을 수행하며 모든 사용자 정의 오류 클래스에 대해 권장된 기본 클래스입니다.
ArgumentError	ArgumentError 클래스는 함수 호출 시 제공된 매개 변수 값이 해당 함수에 대해 정의된 매개 변수와 일치하지 않을 때 발생하는 오류를 나타냅니다.	인수 오류는 다음과 같은 상황에서 발생할 수 있습니다. • 메서드에 제공된 인수가 너무 적거나 너무 많은 경우 • 인수가 열거된 항목에 속해야 하지만 아닌 경우
EvalError	EvalError 예외는 매개 변수가 Function 클래스의 생성자에 전달되거나 사용자 코드에서 eval() 함수를 호출할 경우 발생합니다.	ActionScript 3.0에서는 eval() 함수에 대한 지원이 없어져 이 함수를 사용하려고 하면 오류가 발생합니다. 이전 버전의 Flash Player에서는 eval() 함수를 사용하여 변수, 속성, 객체 또는 동영상 클립을 이름으로 액세스할 수 있었습니다.
RangeError	RangeError 예외는 숫자 값이 허용되는 범위를 벗어난 경우 발생합니다.	예를 들어 지연 시간이 음수이거나 유한수가 아닌 경우 Timer 클래스에서 RangeError가 발생합니다. 잘못된 심도에 표시 객체를 추가하려는 경우에도 RangeError가 발생할 수 있습니다.
ReferenceError	ReferenceError 예외는 봉인된 비동적 객체에서 정의되지 않은 속성을 참조하려고 하는 경우 발생합니다. ActionScript 3.0 이전의 ActionScript 컴파일러 버전에서는 undefined 속성에 액세스하려고 할 때 오류가 발생하지 않았습니다. 그러나 ActionScript 3.0에서는 이 경우 ReferenceError 예외가 발생합니다.	정의되지 않은 변수에 대한 예외는 잠재적인 버그를 식별할 수 있도록 해 주며 소프트웨어 품질을 향상시키는 데 도움이 됩니다. 하지만 변수를 초기화하는 데 익숙하지 않은 경우 이 새로운 ActionScript 비헤이비어로 인해 코딩 습관을 바꿔야 합니다.
SecurityError	SecurityError 예외는 보안 위반이 발생하여 액세스가 거부될 경우 발생합니다.	보안 오류는 다음과 같은 상황에서 발생할 수 있습니다. • 보안 샌드박스 경계를 벗어나 무단으로 속성에 액세스하거나 메서드를 호출하는 경우 • 보안 샌드박스에서 허용되지 않는 URL에 액세스하는 경우 • 필수 소켓 정책 파일이 없는 포트에 소켓 연결을 시도하는 경우 • 사용자의 카메라 또는 마이크에 액세스하려는 시도가 있었으며, 사용자가 장치에 대한 액세스를 거부한 경우
SyntaxError	SyntaxError 예외는 ActionScript 코드에서 파싱 오류가 발생할 경우 발생합니다.	SyntaxError는 다음과 같은 상황에서 발생할 수 있습니다. • RegExp 클래스가 잘못된 일반 표현식을 파싱할 때 ActionScript가 SyntaxError 예외를 발생시키는 경우 • XMLDocument 클래스가 잘못된 XML을 파싱할 때 ActionScript가 SyntaxError 예외를 발생시키는 경우

클래스 이름	설명	참고 사항
TypeError	피연산자의 실제 유형이 예상 유형과 다르면 TypeError 예외가 발생합니다.	TypeError는 다음과 같은 상황에서 발생할 수 있습니다. • 함수 또는 메서드의 실제 매개 변수를 형식 매개 변수 유형으로 강제 변환할 수 없는 경우 • 변수에 할당된 값을 해당 변수의 유형으로 강제 변환할 수 없는 경우 • is 또는 instanceof 연산자에서 오른쪽 객체의 유형이 잘못된 경우 • super 키워드를 잘못 사용한 경우 • 속성을 조회한 결과 여러 바인딩이 발견되어 결과가 모호한 경우 • 호환되지 않는 객체에 메서드를 호출한 경우. 예를 들어 RegExp 클래스의 메서드를 일반 객체로 "이식"하여 호출하면 TypeError 예외가 발생합니다.
URIError	URIError 예외는 전역 URI 처리 함수 중 하나를 해당 정의와 호환되지 않는 방식으로 사용할 경우 발생합니다.	URIError는 다음과 같은 상황에서 발생할 수 있습니다. Socket.connect()와 같은 유효한 URI를 예상하는 Flash Player API 함수에 유효하지 않은 URI가 지정되는 경우
VerifyError	VerifyError 예외는 형식이 잘못되었거나 손상된 SWF 파일이 발견되는 경우 발생합니다.	SWF 파일이 다른 SWF 파일을 로드할 때 부모 SWF 파일은 로드된 SWF 파일에서 발생한 VerifyError를 catch할 수 있습니다.

flash.error 패키지 Error 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.error 패키지에는 Flash 런타임 API의 일부인 Error 클래스가 포함됩니다. 설명한 Error 클래스와 달리 flash.error 패키지는 Flash 런타임(Flash Player, Adobe AIR 등)과 관련된 오류 이벤트를 전달합니다.

클래스 이름	설명	참고 사항
EOFError	EOFError 예외는 사용 가능한 데이터의 끝 부분을 지나 계속 읽으려고 하면 발생합니다.	예를 들어 IDataInput 인터페이스에서 읽기 메서드 중 하나가 호출되고 이 읽기 요청을 수행하기 위한 데이터가 부족한 경우 EOFError가 발생합니다.
IllegalOperationError	IllegalOperationError 예외는 메서드를 구현하지 않았거나 현재의 사용이 구현 범위에서 벗어나는 경우 발생합니다.	잘못된 작업 오류 예외는 다음과 같은 상황에서 발생할 수 있습니다. • DisplayObjectContainer와 같은 기본 클래스에서 스테이지에서 지원할 수 있는 것보다 많은 기능을 제공하는 경우. 예를 들어 스테이지에서 마스크를 가져오거나 설정하려고 할 경우(stage.mask 사용) Flash 런타임에서 "Stage 클래스는 이 속성이나 메서드를 구현하지 않습니다."라는 메시지와 함께 IllegalOperationError를 throw합니다. • 하위 클래스에서 필요하지 않으며 지원하기를 원치 않는 메서드를 상속하는 경우 • 액세스 가능성 기능 지원 없이 Flash Player가 컴파일되었을 때 특정 액세스 가능성 메서드가 호출되는 경우 • Flash Player의 런타임 버전에서 제작 전용 기능이 호출된 경우 • 타임라인에 있는 객체의 이름을 설정하려고 한 경우
IOError	IOError 예외는 일부 I/O 예외 유형이 발생할 경우 발생합니다.	예를 들어 연결되지 않았거나 연결이 해제된 소켓에서 읽기/쓰기 작업을 시도한 경우 IOError 예외가 발생합니다.
MemoryError	MemoryError 예외는 메모리 할당 요청이 실패하면 발생합니다.	기본적으로 ActionScript Virtual Machine 2는 ActionScript 프로그램에서 할당할 수 있는 메모리 크기에 제한을 두지 않습니다. 데스크톱 시스템에서는 메모리 할당에 실패하는 경우가 드물지만, 시스템에서 작업에 필요한 메모리를 할당할 수 없을 경우 이 오류가 발생합니다. 그러므로 데스크톱 시스템에서 이 예외는 극도로 큰 메모리 할당을 요청하는 경우가 아니면 발생하지 않습니다. 예를 들어 32비트 Microsoft® Windows® 프로그램의 경우 액세스할 수 있는 주소 공간이 2GB로 제한되므로 30억 바이트 요청은 불가능합니다.
ScriptTimeoutError	ScriptTimeoutError 예외는 15초 스크립트 타임아웃 간격에 도달할 경우 발생합니다. ScriptTimeoutError 예외를 catch하면 스크립트 타임아웃을 보다 적절하게 처리할 수 있습니다. 예외 핸들러가 없는 경우 catch되지 않는 예외 핸들러에서 오류 메시지와 함께 대화 상자를 표시합니다.	악의적 개발자가 예외를 catch하여 무한 루프를 유지하는 것을 방지하기 위해, 특정 스크립트 동안 발생한 첫 번째 ScriptTimeoutError 예외만 catch할 수 있습니다. 이후의 ScriptTimeoutError 예외는 코드에서 catch할 수 없으며 catch되지 않는 예외 핸들러로 바로 전달됩니다.
StackOverflowError	StackOverflowError 예외는 스크립트에 사용할 수 있는 스택이 소진된 경우 발생합니다.	StackOverflowError 예외는 무한 재귀가 발생했음을 나타낼 수 있습니다.

오류 처리 예제: CustomErrors 응용 프로그램

Flash Player 9 이상, Adobe AIR 1.0 이상

CustomErrors 응용 프로그램은 응용 프로그램 작성 시 사용자 정의 오류 작업에 관련된 기술에 대해 설명합니다. 이러한 기술은 다음과 같습니다.

- XML 패킷 유효성 검사
- 사용자 정의 오류 작성
- 사용자 정의 오류 발생
- 오류 발생 시 사용자에게 알림

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. CustomErrors 응용 프로그램 파일은 Samples/CustomError 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
CustomErrors.mxml 또는 CustomErrors.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/errors/ApplicationError.as	FatalError 및 WarningError 클래스 모두에 대해 기본 오류 클래스 역할을 하는 클래스입니다.
com/example/programmingas3/errors/FatalError.as	응용 프로그램에서 발생하는 FatalError 오류를 정의하는 클래스입니다. 사용자 정의 ApplicationError 클래스를 확장합니다.
com/example/programmingas3/errors/Validator.as	사용자가 제공한 직원의 XML 패킷 유효성을 검사하는 단일 메서드를 정의하는 클래스입니다.
com/example/programmingas3/errors/WarningError.as	응용 프로그램에서 발생하는 WarningError 오류를 정의하는 클래스입니다. 사용자 정의 ApplicationError 클래스를 확장합니다.

CustomErrors 응용 프로그램 개요

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램이 로드되면 initApp() 메서드가 Flex 응용 프로그램에 대해 호출되거나 타임라인(비함수) 코드가 Flash Professional 응용 프로그램에 대해 실행됩니다. 이 코드는 Validator 클래스에서 검증할 샘플 XML 패킷을 정의합니다. 다음 코드가 실행됩니다.

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

XML 패킷은 스테이지의 TextArea 구성 요소 인스턴스에서 나중에 표시됩니다. 이 단계에서는 유효성을 다시 검사하기 전에 XML 패킷을 수정할 수 있습니다.

사용자가 Validate 버튼을 클릭하면 validateData() 메서드가 호출됩니다. 이 메서드는 Validator 클래스의 validateEmployeeXML() 메서드를 사용하여 직원 XML 패킷의 유효성을 검사합니다. 다음 코드는 validateData() 메서드를 보여 줍니다.

```
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);
    }
}
```

먼저 TextArea 구성 요소 인스턴스 xmlText의 내용을 사용하여 임시 XML 객체가 만들어집니다. 그 다음 사용자 정의 Validator 클래스(com.example.programmingas3/errors/Validator.as)의 validateEmployeeXML() 메서드가 호출되고 임시 XML 객체가 매개 변수로 전달됩니다. XML 패킷이 유효한 경우 status Label 구성 요소 인스턴스에 성공 메시지가 표시되며 응용 프로그램이 종료됩니다. validateEmployeeXML() 메서드에서 사용자 정의 오류(FatalError, WarningError 또는 일반 오류 발생)가 발생한 경우 적합한 catch 문이 실행되며 showFatalError(), showWarningError() 또는 showGenericError() 메서드가 호출됩니다. 이 메서드는 각각 사용자에게 특정 오류 발생을 알리기 위해 statusText라는 텍스트 영역에 적합한 메시지를 표시하며, 특정 메시지로 status Label 구성 요소도 업데이트합니다.

직원 XML 패킷에 대한 유효성 검사 중 치명적 오류가 발생하면 다음 코드와 같이 statusText 텍스트 영역에 오류 메시지가 표시되고 xmlText TextArea 구성 요소 인스턴스 및 validateBtn Button 구성 요소 인스턴스가 비활성화됩니다.

```
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();
}
```

치명적 오류가 아닌 경고 오류가 발생한 경우에는 statusText TextArea 인스턴스에 오류 메시지가 표시되지만 xmlText TextField 및 Button 구성 요소 인스턴스는 비활성화되지 않습니다. showWarningError() 메서드는 statusText 텍스트 영역에 사용자 정의 오류 메시지를 표시하며, 이 메시지는 사용자에게 XML 유효성 검증을 계속할지 아니면 스크립트를 취소할지 선택하도록 요청합니다. 다음 예제는 showWarningError() 메서드를 보여 줍니다.

```
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;
}
```

사용자가 Yes 또는 No 버튼을 클릭하면 closeHandler() 메서드가 호출됩니다. 다음 예제는 closeHandler() 메서드를 보여 줍니다.

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

사용자가 Yes를 클릭하여 스크립트를 취소하도록 선택한 경우 FatalError가 발생하여 응용 프로그램이 종료됩니다.

사용자 정의 유효성 검사기 작성

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자 정의 Validator 클래스에는 단일 메서드인 validateEmployeeXML()이 포함됩니다. validateEmployeeXML() 메서드는 유효성을 검사하려는 XML 패킷인 employee를 단일 인수로 취합니다. validateEmployeeXML() 메서드는 다음과 같습니다.

```
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);
    }
}
```

유효성을 검사하기 위해서는 직원이 단일 비용 센터에 속해 있어야 합니다. 직원이 어느 비용 센터에도 속해 있지 않은 경우 메서드에서 FatalError가 발생하며, 이는 기본 응용 프로그램 파일에서 validateData() 메서드가 실행되도록 합니다. 직원이 둘 이상의 비용 센터에 속해 있는 경우 WarningError가 발생합니다. XML 유효성 검사기의 최종 점검 절차는 사용자의 주민등록 번호가 정확하게 정의되었는지 확인하는 것입니다(XML 패킷의 ssn 노드). 정확히 하나의 ssn 노드가 없는 경우 FatalError 오류가 발생합니다.

validateEmployeeXML() 메서드에 ssn 노드에 유효한 번호가 포함되었는지 또는 직원에게 최소한 하나의 전화 번호 및 전자 메일 주소가 정의되어 있는지 및 두 값이 모두 유효한지 등의 점검 사항을 추가할 수 있습니다. 또한 XML을 수정하여 각 직원이 고유한 ID를 가지고 해당 관리자 ID를 지정하도록 할 수 있습니다.

ApplicationError 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

ApplicationError 클래스는 FatalError 및 WarningError 클래스 모두에 대해 기본 클래스 역할을 합니다. ApplicationError 클래스는 Error 클래스를 확장하며 오류 ID, 심각도 및 사용자 정의 오류 코드 및 메시지를 포함하는 XML 객체를 비롯한 고유한 사용자 정의 메서드 및 속성을 정의합니다. 이 클래스는 또한 각 오류 유형의 심각도를 정의하는 데 사용되는 두 개의 정적 상수도 정의합니다.

ApplicationError 클래스 생성자 메서드는 다음과 같습니다.

```
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>;
}
```

XML 객체의 각 오류 노드에는 고유한 숫자 코드 및 오류 메시지가 포함됩니다. 오류 메시지는 다음 getMessageText() 메서드에서 보는 바와 같이 E4X를 사용하여 오류 코드별로 쉽게 조회할 수 있습니다.

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

getMessageText() 메서드는 단일 정수 인수인 id를 취하여, 문자열을 반환합니다. id 인수는 조회할 오류에 대한 오류 코드입니다. 예를 들어 id로 9001을 전달하면 직원은 하나의 비용 센터에만 할당되어야 한다는 오류 메시지가 검색됩니다. 둘 이상의 오류에 동일한 오류 코드가 있을 경우 ActionScript는 발견된 첫 번째 결과에 대해서만 오류 메시지를 반환합니다(반환된 XMLList 객체의 message[0]).

이 클래스의 다음 메서드인 getTitle()은 아무 매개 변수도 취하지 않으며, 이 특정 오류의 오류 ID가 포함된 문자열 값을 반환합니다. 이 값은 XML 패킷 유효성 검사 중 발생한 오류를 쉽고 정확하게 식별하는 데 사용됩니다. 다음 예제는 getTitle() 메서드를 보여 줍니다.

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

ApplicationError 클래스의 마지막 메서드는 toString() 입니다. 이 메서드는 Error 클래스에서 정의된 함수를 대체하므로, 이를 통해 오류 메시지의 표시 형식을 사용자 정의할 수 있습니다. 이 메서드는 발생한 특정 오류 번호 및 메시지를 식별하는 문자열을 반환합니다.

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

FatalError 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

FatalError 클래스는 사용자 정의 ApplicationError 클래스를 확장하며, FatalError 생성자, getTitle() 및 toString()의 세 가지 메서드를 정의합니다. 첫 번째 메서드인 FatalError 생성자는 단일 정수 인수인 errorID를 취하며, ApplicationError 클래스에서 정의된 정적 상수 값을 사용하여 오류 심각도를 설정하고, ApplicationError 클래스의 getMessageText() 메서드를 호출하여 특정 오류의 오류 메시지를 가져옵니다. FatalError 생성자는 다음과 같습니다.

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

FatalError 클래스의 다음 메서드인 getTitle()은 ApplicationError 클래스에서 이전에 정의된 getTitle() 메서드를 대체하며, 제목에 "-- FATAL" 텍스트를 추가하여 사용자에게 치명적 오류가 발생했음을 알립니다. getTitle() 메서드는 다음과 같습니다.

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

이 클래스의 마지막 메서드인 toString()은 ApplicationError 클래스에서 정의된 toString() 메서드를 대체합니다. toString() 메서드는 다음과 같습니다.

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

WarningError 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

WarningError 클래스는 ApplicationError 클래스를 확장하며 다음 코드와 같이 문자열이 조금 변경되고 오류 심각도를 ApplicationError.FATAL이 아닌 ApplicationError.WARNING으로 설정하는 것만 제외하면 FatalError 클래스와 거의 동일합니다.

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

5장: 일반 표현식 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

일반 표현식은 문자열에서 일치하는 텍스트를 찾고 조작하는 데 사용되는 패턴을 기술합니다. 이러한 일반 표현식은 문자열과 비슷하지만 패턴과 반복을 기술하는 특수 코드를 포함할 수 있습니다. 예를 들어, 다음 일반 표현식은 A로 시작하고 그 뒤에 하나 이상의 숫자가 순차적으로 나오는 문자열을 찾습니다.

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

다음 항목에서는 일반 표현식을 구성하는 기본 구문에 대해 설명합니다. 그러나 일반 표현식은 복잡하고 함축적일 수 있으므로 웹 또는 서점에서 일반 표현식에 대한 자세한 리소스를 참조하면 도움이 됩니다. 또한 프로그래밍 환경에 따라 다양한 방법으로 일반 표현식을 구현한다는 사실을 기억해야 합니다. ActionScript 3.0에서는 ECMAScript 버전 3 언어 사양(ECMA-262)에 정의된 일반 표현식을 구현합니다.

기타 도움말 항목
RegExp

일반 표현식의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

일반 표현식은 문자 패턴을 기술합니다. 일반 표현식은 사용자가 입력한 전화 번호의 자릿수가 올바른지 확인하는 것처럼 텍스트 값이 특정 패턴을 따르는지 확인하거나, 텍스트 값에서 특정 패턴과 일치하는 부분을 바꾸는 데 주로 사용됩니다.

일반 표현식은 단순하게 나타낼 수 있습니다. 예를 들어, 특정 문자열이 "ABC"와 일치하는지 확인하거나 문자열에 있는 모든 "ABC"를 다른 텍스트로 대체하려 할 경우 다음과 같은 일반 표현식을 사용하여 A, B, C 문자가 차례로 포함된 패턴을 정의할 수 있습니다.

```
/ABC/
```

일반 표현식 리터럴은 슬래시(/) 문자로 나타냅니다.

일반 표현식 패턴은 유효한 전자 메일 주소를 찾는 다음 표현식과 같이 복잡할 수도 있으며 경우에 따라 암호처럼 보일 수도 있습니다.

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

일반 표현식은 문자열의 패턴을 찾아서 문자를 바꿀 때 흔히 사용됩니다. 이와 같은 경우 일반 표현식 객체를 만들어 여러 String 클래스 메서드 중 하나에 대한 매개 변수로 사용할 수 있습니다. String 클래스의 match(), replace(), search() 및 split() 메서드는 일반 표현식을 매개 변수로 사용합니다. 이러한 메서드에 대한 자세한 내용은 15페이지의 “문자열의 패턴 찾기 및 하위 문자열 바꾸기”를 참조하십시오.

RegExp 클래스에는 test() 및 exec() 메서드가 포함되어 있습니다. 자세한 내용은 80페이지의 “문자열에 일반 표현식을 사용하는 데 필요한 메서드”를 참조하십시오.

중요한 개념 및 용어
이 기능과 관련된 중요한 용어들이 아래 참조 목록에 정리되어 있습니다.

이스케이프 문자 다음에 오는 문자가 리터럴 문자가 아닌 메타문자로 처리되어야 함을 나타내는 문자입니다. 일반 표현식 구문에서 백슬래시 문자(\)는 이스케이프 문자로, 백슬래시 뒤에 오는 문자는 일반 문자가 아닌 특수 코드로 처리됩니다.

플래그 일반 표현식 패턴 사용 방법에 대한 옵션을 지정하는 문자입니다(예: 대/소문자 구분 여부).

메타문자 일반 표현식 패턴에서 특별한 의미를 갖는 문자로, 해당 패턴에서 문자를 글자 그대로 나타내는 문자와 대비되는 개념입니다.

한정 기호 패턴을 구성하는 한 부분의 반복 횟수를 나타내는 문자(또는 여러 문자)입니다. 예를 들어 우편 번호를 반드시 5자리 또는 9자리 숫자로 구성하도록 지정할 때 한정 기호를 사용할 수 있습니다.

일반 표현식 문자 패턴을 정의하는 프로그램 명령문으로, 다른 문자열이 해당 패턴과 일치하는지 여부를 확인하거나 문자열의 일부를 바꾸기 위해 사용할 수 있습니다.

일반 표현식 구문

Flash Player 9 이상, Adobe AIR 1.0 이상

이 단원에서는 ActionScript 일반 표현식 구문의 모든 요소에 대해 설명합니다. 일반 표현식은 복잡하고 함축적일 수 있으므로 웹 또는 서점에서 일반 표현식에 대한 자세한 리소스를 참조하면 도움이 됩니다. 또한 프로그래밍 환경에 따라 다양한 방법으로 일반 표현식을 구현한다는 사실을 기억해야 합니다. ActionScript 3.0에서는 ECMAScript 버전 3 언어 사양(ECMA-262)에 정의된 일반 표현식을 구현합니다.

일반적으로 간단한 문자열보다 복잡한 패턴과 일치하는 일반 표현식을 사용합니다. 예를 들어, 다음 일반 표현식은 A, B, C 문자가 차례로 나오고 그 다음에 숫자가 나오는 패턴을 정의합니다.

```
/ABC\d/
```

여기에서 \d 코드는 "임의의 숫자"를 나타냅니다. 백슬래시(\) 문자는 이스케이프 문자라고 하며 뒤에 나오는 문자(이 예제에서는 d)와 결합할 경우 일반 표현식에서 특별한 의미를 가집니다.

다음 일반 표현식은 ABC 문자 다음에 숫자(개수 제한 없음)가 나오는 패턴을 정의합니다. 이때 별표에 주의하십시오.

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

별표 문자(*)는 **메타문자**입니다. 메타문자는 일반 표현식에서 특별한 의미를 갖는 문자입니다. 별표는 메타문자의 특정 유형으로 **한정 기호**라고 하며 반복되는 단일 문자 또는 문자 그룹의 범위(개수)를 지정하는 데 사용됩니다. 자세한 내용은 73페이지의 “한정 기호”를 참조하십시오.

일반 표현식에는 패턴뿐만 아니라 플래그를 포함하여 일반 표현식을 일치시키는 방식을 지정할 수 있습니다. 예를 들어, 다음 일반 표현식에서는 i 플래그를 사용하여 일치하는 문자열에서 대/소문자를 구분하지 않도록 지정합니다.

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

자세한 내용은 77페이지의 “플래그 및 속성”을 참조하십시오.

String 클래스의 match(), replace() 및 search() 메서드와 함께 일반 표현식을 사용할 수 있습니다. 이러한 메서드에 대한 자세한 내용은 15페이지의 “문자열의 패턴 찾기 및 하위 문자열 바꾸기”를 참조하십시오.

일반 표현식 인스턴스 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

두 가지 방법을 사용하여 일반 표현식 인스턴스를 만들 수 있는데, 첫 번째 방법은 슬래시 문자(/)를 사용하여 일반 표현식을 나타내고 다른 방법은 new 생성자를 사용합니다. 예를 들어, 다음 일반 표현식은 동일합니다.

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

슬래시는 따옴표를 사용하여 문자열 리터럴을 나타내는 것과 같은 방식으로 일반 표현식 리터럴을 나타냅니다. 슬래시 내의 일반 표현식 부분은 **패턴을 정의합니다.** 또한 일반 표현식에서 마지막 슬래시 뒤에 **플래그**를 포함할 수도 있습니다. 이러한 플래그는 일반 표현식의 일부로 간주되지만 패턴과 별개입니다.

new 생성자를 사용할 때는 두 문자열을 사용하여 일반 표현식을 정의하는데, 다음 예제와 같이 첫 번째 문자열은 패턴을 정의하고 두 번째 문자열은 플래그를 정의합니다.

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

슬래시 기호를 사용하여 정의된 일반 표현식 **내에** 다시 슬래시를 포함하는 경우에는 슬래시 앞에 백슬래시(\) 이스케이프 문자를 추가해야 합니다. 예를 들어, 다음 일반 표현식은 1/2 패턴과 일치하는 문자열을 찾습니다.

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

new 생성자를 사용하여 정의된 일반 표현식 내에 따옴표를 포함하려면 문자열 리터럴을 정의할 때처럼 따옴표 앞에 백슬래시(\) 이스케이프 문자를 추가해야 합니다. 예를 들어, 다음 일반 표현식은 eat at "joe's" 패턴과 일치하는 문자열을 찾습니다.

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

슬래시 기호를 사용하여 정의된 일반 표현식에서 백슬래시 이스케이프 문자를 따옴표와 함께 사용하면 안 됩니다. 마찬가지로 new 생성자를 사용하여 정의된 일반 표현식에서 이스케이프 문자를 슬래시와 함께 사용하면 안 됩니다. 다음 일반 표현식은 동일하며 1/2 "joe's" 패턴을 정의합니다.

```
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"', '');
```

또한 new 생성자를 사용하여 정의된 일반 표현식에서 임의의 숫자와 일치하는 \d와 같이 백슬래시(\) 문자로 시작하는 메타시퀀스를 사용하려면 백슬래시 문자를 두 번 입력합니다.

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

이 경우 RegExp() 생성자 메서드의 첫 번째 매개 변수가 문자열이고 문자열 리터럴에서 단일 백슬래시 문자를 인식하려면 백슬래시 문자를 두 번 입력해야 하므로 결과적으로 백슬래시 문자를 두 번 입력해야 합니다.

다음에 나오는 단원에서는 일반 표현식 패턴을 정의하는 구문에 대해 설명합니다.

플래그에 대한 자세한 내용은 77페이지의 “플래그 및 속성”을 참조하십시오.

문자, 메타문자 및 메타시퀀스

Flash Player 9 이상, Adobe AIR 1.0 이상

가장 간단한 일반 표현식은 다음 예제와 같이 문자 시퀀스와 일치하는 일반 표현식입니다.

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

그러나 메타문자라고 하는 다음 문자는 일반 표현식에서 특별한 의미를 가집니다.

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

예를 들어, 다음 일반 표현식은 A 문자 다음에 B 문자 인스턴스가 0개 이상 나오고(별표 메타문자는 이 반복을 나타냄) 그 다음에 C 문자가 나오는 문자열을 찾습니다.

```
/AB*C/
```

일반 표현식 패턴에서 특별한 의미가 없는 메타문자를 포함하려면 백슬래시(\) 이스케이프 문자를 사용해야 합니다. 예를 들어, 다음 일반 표현식은 A 문자, B 문자, 별표, C 문자가 차례로 나오는 문자열을 찾습니다.

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

메타문자와 같이 **메타시퀀스**도 일반 표현식에서 특별한 의미를 가집니다. 메타시퀀스는 둘 이상의 문자로 구성됩니다. 다음 단원에서는 메타문자와 메타시퀀스를 사용하는 방법에 대해 자세히 설명합니다.

메타문자 정보

다음 표에는 일반 표현식에 사용할 수 있는 메타문자가 요약되어 있습니다.

메타문자	설명
^(캐럿)	문자열의 시작 부분에서 찾습니다. m(multiline) 플래그를 설정하면 캐럿 문자는 행의 시작 부분도 찾습니다(77페이지의 "플래그 및 속성" 참조). 또한 문자 클래스의 맨 처음에 사용되는 경우에는 문자열의 시작 지점이 아니라 부정을 나타냅니다. 자세한 내용은 72페이지의 "문자 클래스"를 참조하십시오.
$(달러 기호)	문자열의 끝 부분에서 찾습니다. m(multiline) 플래그를 설정하면 $는 개행(\n) 문자 앞의 위치도 찾습니다. 자세한 내용은 77페이지의 "플래그 및 속성"을 참조하십시오.
\(백슬래시)	특수 문자의 특별한 메타문자 의미를 이스케이프합니다. 또한 /1\/2/와 같이 문자 1, 슬래시 문자, 문자 2가 차례로 나오도록 슬래시 문자를 일반 표현식 리터럴로 사용하려는 경우에도 백슬래시 문자를 사용합니다.
. (도트)	임의의 단일 문자를 찾습니다. 도트는 s(dotall) 플래그가 설정된 경우에만 개행 문자(\n)를 찾습니다. 자세한 내용은 77페이지의 "플래그 및 속성"을 참조하십시오.
*(별표)	바로 앞의 항목이 0번 이상 반복된 것을 찾습니다. 자세한 내용은 73페이지의 "한정 기호"를 참조하십시오.
+(더하기)	바로 앞의 항목이 1번 이상 반복된 것을 찾습니다. 자세한 내용은 73페이지의 "한정 기호"를 참조하십시오.
?(물음표)	바로 앞의 항목이 0번 또는 1번 반복된 것을 찾습니다. 자세한 내용은 73페이지의 "한정 기호"를 참조하십시오.

메타문자	설명
(및)	일반 표현식 내에서 그룹을 정의합니다. 다음과 같은 경우 그룹을 사용합니다. • | 문자의 범위를 제한하려는 경우(예: /(a|b|c)d/) • 한정 기호 범위를 정의하려는 경우(예: /(walla.){1,2}/) • 역참조를 사용하는 경우. 예를 들어, 다음 일반 표현식에서 \1은 패턴의 첫 번째 괄호 그룹과 일치하는 모든 항목을 찾습니다. • /(\w*) is repeated: \1/ 자세한 내용은 75페이지의 "그룹"을 참조하십시오.
[및]	단일 문자에 대해 일치하는 항목을 정의하는 문자 클래스를 정의합니다. /[aeiou]/ 지정된 문자 중 하나를 찾습니다. 문자 클래스 내에서 문자 범위를 지정하려면 하이픈(-)을 사용합니다. /[A-Z0-9]/ 대문자 A~Z 또는 0~9를 찾습니다. 문자 클래스 내에서] 및 - 문자를 이스케이프하려면 백슬래시를 - 문자: /[+\-]\d+/ 하나 이상의 숫자 앞에서 + 또는 -를 찾습니다. 문자 클래스 내에서 다른 문자(일반적으로 메타문자)는 메타문자가 아니라 일반 문자로 처리되므로 백슬래시를 사용할 필요가 없습니다. /[$]/£ $또는 £를 찾습니다. 자세한 내용은 72페이지의 "문자 클래스"를 참조하십시오.
|(파이프)	왼쪽 항목 또는 오른쪽 항목 중 하나를 찾습니다. /abc|xyz/abc 또는 xyz를 찾습니다.

메타시퀀스 정보

메타시퀀스는 일반 표현식 패턴에서 특별한 의미를 갖는 문자 시퀀스입니다. 다음 표에서는 이러한 메타시퀀스에 대해 설명합니다.

메타시퀀스	설명
{*n*} {*n*,} 및 {*n*,*n*}	앞의 항목에 대한 숫자 한정 기호 또는 한정 기호 범위를 지정합니다. /A{27}/27번 반복된 A 문자를 찾습니다. /A{3,}/3번 이상 반복된 A 문자를 찾습니다. /A{3,5}/3~5번 반복된 A 문자를 찾습니다. 자세한 내용은 73페이지의 "한정 기호"를 참조하십시오.
\b	단어 문자와 단어가 아닌 문자 사이의 위치에서 찾습니다. 또한 문자열의 첫 번째 문자 또는 마지막 문자가 단어이면 문자열의 시작 또는 끝 부분도 찾습니다.
\B	두 개의 단어 문자 사이의 위치에서 찾습니다. 또한 단어가 아닌 두 문자 사이의 위치에서도 찾습니다.
\d	10진수를 찾습니다.
\D	숫자 이외의 다른 문자를 찾습니다.
\f	용지 공급 문자를 찾습니다.

메타시퀀스	설명
\n	개행 문자를 찾습니다.
\r	캐리지 리턴 문자를 찾습니다.
\s	빈 칸, 탭, 개행 문자 또는 캐리지 리턴 문자 등을 포함하여 모든 공백 문자를 찾습니다.
\S	공백이 아닌 다른 문자를 찾습니다.
\t	탭 문자를 찾습니다.
\u*nnnn*	16진수 **nnnn**으로 지정된 문자 코드를 사용하는 유니코드 문자를 찾습니다. 예를 들어, \u263a는 스마일 문자입니다.
\v	수직 탭 문자를 찾습니다.
\w	단어 문자(AZ–, az–, 0-9 또는 _)를 찾습니다. 이때 \w는 é, ñ 또는 ç 등 영어 이외의 문자는 찾지 않습니다.
\W	단어가 아닌 다른 문자를 찾습니다.
\\x*nn*	16진수 **nn**으로 정의된 ASCII 값을 사용하는 문자를 찾습니다.

문자 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

문자 클래스를 사용하면 일반 표현식에서 한 위치에 대응되는 문자 목록을 지정할 수 있습니다. 문자 클래스를 정의할 때는 대괄호([및])를 사용합니다. 예를 들어, 다음 일반 표현식은 bag, beg, big, bog 또는 bug와 일치하는 문자 클래스를 정의합니다.

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

문자 클래스의 이스케이프 시퀀스

대개 일반 표현식에서 특별한 의미를 갖는 메타문자와 메타시퀀스의 대부분은 문자 클래스 내에서는 같은 의미를 갖지 **않습니다**. 예를 들어, 별표는 일반 표현식에서 반복을 나타내는 데 사용되지만 문자 클래스에 나타날 때는 그렇지 않습니다. 다음 문자 클래스는 나열된 다른 모든 문자와 함께 별표를 문자 그대로 찾습니다.

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

그러나 다음 표에 나열된 세 단어는 메타문자로 사용되어 문자 클래스에서도 특별한 의미를 가집니다.

메타문자	문자 클래스에 사용할 경우 의미
]	문자 클래스의 끝을 정의합니다.
-	문자 범위를 정의합니다("문자 클래스의 문자 범위" 참조).
\	메타시퀀스를 정의하고 메타문자의 특별한 의미를 제거합니다.

이러한 문자가 특별한 메타문자 의미를 갖지 않고 리터럴 문자로 인식되게 하려면 해당 문자 앞에 백슬래시 이스케이프 문자를 추가해야 합니다. 예를 들어, 다음 일반 표현식에는 네 가지 심볼($, \,] 또는 -) 중 하나와 일치하는 문자 클래스가 포함되어 있습니다.

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

특별한 의미를 유지하는 메타문자뿐만 아니라 다음 메타시퀀스도 문자 클래스 내에서 특별한 의미를 갖는 메타시퀀스로 사용됩니다.

메타시퀀스	문자 클래스에 사용할 경우 의미
\n	개행 문자를 찾습니다.
\r	캐리지 리턴 문자를 찾습니다.
\t	탭 문자를 찾습니다.
\u*nnnn*	16진수 **nnnn**으로 정의된 유니코드 값을 사용하는 문자를 찾습니다.
\\x*nn*	16진수 **nn**으로 정의된 ASCII 값을 사용하는 문자를 찾습니다.

다른 일반 표현식 메타시퀀스 및 메타문자는 문자 클래스 내에서 일반 문자로 처리됩니다.

문자 클래스의 문자 범위

하이픈을 사용하면 A-Z, a-z 또는 0-9와 같이 문자 범위를 지정할 수 있습니다. 이러한 문자는 문자 세트에서 유효한 범위를 구성해야 합니다. 예를 들어, 다음 문자 클래스는 a-z 범위의 문자 중 하나 또는 임의의 숫자를 찾습니다.

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

\\x**nn** ASCII 문자 코드를 사용하여 ASCII 값으로 범위를 지정할 수도 있습니다. 예를 들어, 다음 문자 클래스는 확장 ASCII 문자(예: é 및 ê) 세트의 문자를 찾습니다.

```
\\x
```

문자 클래스 부정

캐럿(^) 문자를 문자 클래스의 맨 앞에 사용하면 해당 클래스를 제외하고 나열되지 않은 모든 문자를 찾습니다. 다음 문자 클래스는 소문자(a–z) 또는 숫자 **이외의** 모든 문자를 찾습니다.

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

부정을 나타내려면 캐럿(^) 문자를 문자 클래스의 **맨 앞에** 입력해야 합니다. 그렇지 않으면 캐럿 문자가 문자 클래스의 문자에 추가됩니다. 예를 들어, 다음 문자 클래스는 캐럿을 포함하여 많은 심볼 문자 중 하나를 찾습니다.

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

한정 기호

Flash Player 9 이상, Adobe AIR 1.0 이상

한정 기호를 사용하면 다음과 같이 패턴에서 문자 또는 시퀀스의 반복을 지정할 수 있습니다.

한정 기호 메타문자	설명
*(별표)	바로 앞의 항목이 0번 이상 반복된 것을 찾습니다.
+(더하기)	바로 앞의 항목이 1번 이상 반복된 것을 찾습니다.
?(물음표)	바로 앞의 항목이 0번 또는 1번 반복된 것을 찾습니다.
{*n*} {*n*,} 및 {*n*,*n*}	앞의 항목에 대한 숫자 한정 기호 또는 한정 기호 범위를 지정합니다. /A{27}/ 27번 반복된 A 문자를 찾습니다. /A{3,}/ 3번 이상 반복된 A 문자를 찾습니다. /A{3,5}/ 3~5번 반복된 A 문자를 찾습니다.

한정 기호를 단일 문자, 문자 클래스 또는 그룹에 적용할 수 있습니다.

- /a+/ 1번 이상 반복된 a 문자를 찾습니다.

- /\d+/ 하나 이상의 숫자를 찾습니다.
- /[abc]+/a, b 또는 c 중에서 하나 이상의 문자가 반복되는 항목을 찾습니다.
- /(very,)*/ very라는 단어와 쉼표, 공백이 차례로 0번 이상 반복되는 항목을 찾습니다.

한정 기호가 적용된 괄호 그룹 내에서 한정 기호를 사용할 수 있습니다. 예를 들어, 다음 한정 기호는 word 및 word-word-word와 같은 문자열을 찾습니다.

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

기본적으로 일반 표현식은 **최장 일치를 수행합니다.** 즉, 일반 표현식의 하위 패턴(예: .*)은 문자열에서 가능한 한 많은 문자를 찾은 후 해당 일반 표현식의 다음 부분으로 이동합니다. 예를 들어, 다음 일반 표현식과 문자열을 검토해 보십시오.

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

이 일반 표현식은 전체 문자열을 찾습니다.

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

하지만 ... 그룹을 하나만 찾으려는 경우를 가정해 봅니다. 다음과 같이 하면 이렇게 할 수 있습니다.

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

한정 기호를 **최단 일치 한정 기호**로 바꾸려면 해당 한정 기호 뒤에 물음표(?)를 추가합니다. 예를 들어, 최단 일치 한정 기호 *?를 사용하는 다음 일반 표현식은 , 최소 문자 수(최단 일치), 가 차례로 나오는 항목을 찾습니다.

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

한정 기호를 사용할 경우에는 다음과 같은 내용을 항상 기억해야 합니다.

- {0} 및 {0,0} 한정 기호는 일치시킬 때 항목을 제외하지 않습니다.
- /abc+*/와 같이 여러 한정 기호를 함께 사용하지 마십시오.
- 도트(.)는 * 한정 기호가 뒤에 나오더라도 s(dotall) 플래그가 설정되어 있지 않으면 행을 확장하지 않습니다. 예를 들어 다음과 같은 코드를 살펴봅니다.

```
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>
```

자세한 내용은 77페이지의 “플래그 및 속성”을 참조하십시오.

여러 항목 중 하나 선택

Flash Player 9 이상, Adobe AIR 1.0 이상

일반 표현식에 |(파이프) 문자를 사용하면 일반 표현식 엔진에서 여러 항목 중 하나를 찾습니다. 예를 들어, 다음 일반 표현식은 cat, dog, pig, rat 단어 중 하나를 찾습니다.

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

괄호를 사용하여 그룹을 정의하면 | 문자의 범위를 제한할 수 있습니다. 다음 일반 표현식은 nap 또는 nip이라는 단어가 뒤에 나오는 cat을 찾습니다.

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

자세한 내용은 75페이지의 “그룹”을 참조하십시오.

각각 | 문자와 문자 클래스([및] 문자를 사용하여 정의)를 사용하는 다음 두 일반 표현식은 동일합니다.

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

자세한 내용은 72페이지의 “문자 클래스”를 참조하십시오.

그룹

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같이 괄호를 사용하여 일반 표현식에서 그룹을 지정할 수 있습니다.

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

그룹은 패턴의 하위 영역입니다. 그룹을 사용하면 다음과 같은 작업을 수행할 수 있습니다.

- 둘 이상의 문자에 한정 기호를 적용합니다.
- | 문자를 사용하여 선택할 하위 패턴을 기술합니다.
- 하위 문자열 일치 항목을 캡처합니다. 예를 들어, 일반 표현식에 \1을 사용하여 이전의 일치 그룹을 찾거나, 이와 비슷하게 String 클래스의 replace() 메서드에서 $1을 사용합니다.

다음 단원에서는 이러한 그룹을 사용하는 방법에 대해 자세히 설명합니다.

한정 기호와 함께 그룹 사용

그룹을 사용하지 않으면 한정 기호는 다음과 같이 해당 한정 기호 앞에 나오는 문자 또는 문자 클래스에 적용됩니다.

```
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
```

그러나 그룹을 사용하면 둘 이상의 문자 또는 문자 클래스에 한정 기호를 적용할 수 있습니다.

```
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
```

한정 기호에 대한 자세한 내용은 73페이지의 “한정 기호”를 참조하십시오.

| 문자와 함께 그룹 사용

그룹을 사용하면 다음과 같이 | 문자를 적용할 문자 그룹을 정의할 수 있습니다.

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

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

그룹을 사용하여 하위 문자열 일치 항목 캡처

패턴에서 표준 괄호 그룹을 정의하는 경우 나중에 일반 표현식에서 이 그룹을 참조할 수 있습니다. 이를 **역참조**라고 하고 이러한 그룹을 **캡처 그룹**이라고 합니다. 예를 들어, 다음 일반 표현식에서 \1 시퀀스는 캡처 괄호 그룹과 일치하는 모든 하위 문자열을 찾습니다.

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

\1, \2, ... , \99을 입력하여 일반 표현식에서 이러한 역참조를 최대 99개까지 지정할 수 있습니다.

마찬가지로 String 클래스의 replace() 메서드에서 $1–$99를 사용하여, 캡처한 그룹 하위 문자열 일치 항목을 대체 문자열에 삽입할 수 있습니다.

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

또한 캡처 그룹을 사용하는 경우 RegExp 클래스의 exec() 메서드 및 String 클래스의 match() 메서드는 캡처 그룹과 일치하는 하위 문자열을 반환합니다.

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

비캡처 그룹 및 예측 그룹 사용

비캡처 그룹은 그룹화에만 사용되는 그룹으로, "수집"되지 않으며 번호가 지정된 역참조와 일치하지 않습니다. 비캡처 그룹을 정의하려면 다음과 같이 (?: 및)를 사용하십시오.

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

예를 들어, (com|org)를 각각 캡처 그룹에 추가하는 것과 비캡처 그룹에 추가하는 경우의 차이점에 주의하십시오. exec() 메서드는 완전히 일치하는 항목 다음에 캡처 그룹을 나열합니다.

```
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
```

비캡처 그룹에는 **예측 그룹**이라는 특수한 유형의 그룹이 있는데, 이 그룹은 **긍정적 예측 그룹**과 **부정적 예측 그룹**이라는 두 가지 유형으로 구분됩니다.

(?= 및)를 사용하여 긍정적 예측 그룹을 정의할 수 있습니다. 이 그룹은 그룹 내의 하위 패턴이 지정된 위치에서 일치해야 함을 지정합니다. 그러나 긍정적 예측 그룹과 일치하는 문자열 부분은 일반 표현식의 나머지 패턴과도 일치할 수 있습니다. 예를 들어, 다음 코드에서 (?=e)는 긍정적 예측 그룹이므로 이 그룹과 일치하는 e는 일반 표현식의 나머지 부분(이 코드에서는 캡처 그룹 \w*))과도 일치할 수 있습니다.

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

(?! 및)를 사용하여 부정적 예측 그룹을 정의할 수 있습니다. 이 그룹은 그룹 내의 하위 패턴이 지정된 위치에서 일치하지 않아야 함을 지정합니다. 예를 들면 다음과 같습니다.

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

명명된 그룹 사용

명명된 그룹은 일반 표현식에서 특정 식별자가 지정된 그룹 유형입니다. 명명된 그룹을 정의하려면 (?P<name> 및)를 사용합니다. 예를 들어, 다음 일반 표현식에는 digits라는 식별자가 지정된 명명된 그룹이 포함되어 있습니다.

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

exec() 메서드를 사용하면 일치하는 명명된 그룹이 result 배열의 속성으로 추가됩니다.

```
var myPattern:RegExp = /([a-z]+)(?P<digits>\d+)[a-z]+/;
var str:String = "a123bcd";
var result:Array = myPattern.exec(str);
trace(result.digits); // 123
```

다음 예제에서는 각각 name과 dom이라는 식별자가 지정된 두 개의 명명된 그룹을 사용합니다.

```
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
```

참고: 명명된 그룹은 ECMAScript 언어 사양에 포함되어 있지 않으며 ActionScript 3.0에 추가된 기능입니다.

플래그 및 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 표에서는 일반 표현식에 설정할 수 있는 다섯 개의 플래그에 대해 설명합니다. 각 플래그는 일반 표현식 객체의 속성으로 액세스할 수 있습니다.

플래그	속성	설명
g	global	둘 이상의 일치 항목을 찾습니다.
i	ignoreCase	일치하는 항목을 찾을 때 대/소문자를 구분하지 않습니다. 또한 A-Z 및 a-z 문자에는 적용되지만 É 및 é와 같은 확장 문자에는 적용되지 않습니다.
m	multiline	이 플래그가 설정되어 있는 경우 $와 ^은 각각 행의 시작 부분과 끝 부분을 찾을 수 있습니다.
s	dotall	이 플래그가 설정되어 있는 경우 .(도트)는 개행 문자(\n)를 찾을 수 있습니다.
x	extended	확장 일반 표현식을 사용할 수 있습니다. 패턴의 일부로 무시되는 공백을 일반 표현식에 입력할 수 있으며 이를 통해 일반 표현식 코드를 읽기 쉽게 입력할 수 있습니다.

이러한 속성은 읽기 전용이며 다음과 같이 일반 표현식 변수를 설정할 때 플래그(g, i, m, s, x)를 설정할 수 있습니다.

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

하지만 명명된 속성을 직접 설정할 수는 없습니다. 예를 들어, 다음 코드를 실행하면 오류가 발생합니다.

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

기본적으로 일반 표현식 선언에 플래그를 지정하지 않으면 해당 플래그가 설정되지 않으며 해당 속성이 false로 설정됩니다.

또한 다음과 같이 일반 표현식의 다른 두 속성도 사용할 수 있습니다.

- lastIndex 속성은 일반 표현식의 exec() 또는 test() 메서드를 다음에 호출할 때 사용하기 위해 문자열의 인덱스 위치를 지정합니다.
- source 속성은 일반 표현식의 패턴 부분을 정의하는 문자열을 지정합니다.

g(global) 플래그

g(global) 플래그가 포함되지 않은 일반 표현식은 일치 항목을 하나만 찾습니다. 예를 들어, g 플래그가 일반 표현식에 포함되어 있지 않은 경우 String.match() 메서드는 일치하는 하위 문자열을 하나만 반환합니다.

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

g 플래그가 설정되면 Sting.match() 메서드는 다음과 같이 일치 항목을 여러 개 반환합니다.

```
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
```

i(ignoreCase) 플래그

기본적으로 일반 표현식으로 일치 항목을 찾을 때는 대/소문자를 구분합니다. 그러나 i(ignoreCase) 플래그를 설정하면 대/소문자 구분 특성이 무시됩니다. 예를 들어, 일반 표현식에 소문자 s를 포함하면 문자열의 첫 번째 문자인 대문자 S를 찾을 수 없습니다.

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

그러나 i 플래그를 설정하면 일반 표현식에서 대문자 S를 찾습니다.

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

i 플래그는 A-Z 및 a-z 문자에 대해서만 대/소문자 구분 특성을 무시하고 É와 é 같은 확장 문자에는 적용되지 않습니다.

m(multiline) 플래그

m(multiline) 플래그가 설정되어 있지 않은 경우 ^은 문자열의 시작 부분을 찾고 $는 문자열의 끝 부분을 찾습니다. m 플래그가 설정되어 있는 경우 이러한 문자는 각각 행의 시작 부분과 끝 부분을 찾습니다. 개행 문자가 포함된 다음 문자열을 검토하십시오.

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

일반 표현식에 g(global) 플래그가 설정되어 있더라도 문자열의 시작 지점인 ^에 대해서는 일치 항목이 하나뿐이므로 match() 메서드는 하위 문자열을 하나만 찾습니다. 따라서 출력 결과는 다음과 같습니다.

```
Test
```

다음은 같은 코드에 m 플래그를 설정한 경우입니다.

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

그러면 두 행의 시작 부분에 있는 단어가 모두 출력에 포함됩니다.

```
Test,Multiline
```

행의 끝을 나타내는 문자는 \n뿐이며 다음 문자로는 행의 끝을 나타낼 수 없습니다.

- 캐리지 리턴(\r) 문자
- 유니코드 행 분리 기호(\u2028) 문자
- 유니코드 단락 분리 기호(\u2029) 문자

s(dotall) 플래그

s(dotall 또는 "dot all") 플래그가 설정되어 있지 않은 경우 일반 표현식 패턴에 도트(.)를 포함해도 개행 문자(\n)를 찾을 수 없습니다. 다음 예제의 경우 일치하는 항목이 없습니다.

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

하지만 s 플래그를 설정하면 개행 문자를 찾을 수 있습니다.

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

이 경우 개행 문자를 포함하여 <p> 태그 내에 있는 전체 하위 문자열이 검색됩니다.

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

x(extended) 플래그

메타심볼이나 메타시퀀스가 많이 포함된 일반 표현식은 쉽게 읽을 수 없습니다. 예를 들면 다음과 같습니다.

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

일반 표현식에 x(extended) 플래그를 사용하면 일반 표현식 패턴에 입력하는 공백이 무시됩니다. 예를 들어, 다음 일반 표현식은 위의 예제에 나오는 일반 표현식과 동일합니다.

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

x 플래그를 설정한 경우 공백 문자를 찾지 않으려면 공백 앞에 백슬래시를 추가하십시오. 예를 들어, 다음 두 일반 표현식은 동일합니다.

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

lastIndex 속성

lastIndex 속성은 문자열에서 다음 검색을 시작할 인덱스 위치를 지정합니다. 이 속성은 g 플래그가 true로 설정된 일반 표현식에서 호출되는 exec() 및 test() 메서드에 영향을 줍니다. 예를 들어 다음과 같은 코드를 살펴봅니다.

```
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);
}
```

lastIndex 속성은 기본적으로 0으로 설정되는데, 이렇게 하면 문자열의 맨 앞에서 검색을 시작합니다. 각 일치 항목을 찾은 후에는 이 속성이 해당 항목 다음의 인덱스 위치로 설정됩니다. 따라서 위의 코드를 실행하면 다음과 같이 출력됩니다.

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

global 플래그가 false로 설정되어 있는 경우 exec() 및 test() 메서드는 lastIndex 속성을 사용하거나 설정하지 않습니다.

String 클래스의 match(), replace() 및 search() 메서드는 해당 메서드를 호출할 때 사용된 일반 표현식의 lastIndex 속성 설정에 관계없이 문자열의 맨 앞에서 모든 검색을 시작합니다. 그러나 match() 메서드는 lastIndex 속성을 0으로 설정합니다.

lastIndex 속성을 설정하여 문자열에서 일반 표현식으로 일치 항목을 검색할 시작 위치를 조정할 수 있습니다.

source 속성

source 속성은 일반 표현식의 패턴 부분을 정의하는 문자열을 지정합니다. 예를 들면 다음과 같습니다.

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

문자열에 일반 표현식을 사용하는 데 필요한 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

RegExp 클래스에는 exec()와 test()라는 두 메서드가 포함되어 있습니다.

문자열에서 일반 표현식을 사용하여 일치 항목을 찾을 때는 RegExp 클래스의 exec() 및 test() 메서드뿐만 아니라 String 클래스에 포함된 match(), replace(), search() 및 splice() 메서드도 사용할 수 있습니다.

test() 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

RegExp 클래스의 test() 메서드는 다음 예제와 같이 제공된 문자열에 일반 표현식에 대해 일치하는 항목이 포함되어 있는지 여부만 확인합니다.

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

exec() 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

RegExp 클래스의 exec() 메서드는 제공된 문자열에 일반 표현식에 대해 일치하는 항목이 포함되어 있는지 확인한 후 다음 항목이 포함된 배열을 반환합니다.

- 일치하는 하위 문자열
- 일반 표현식의 괄호 그룹에 대해 일치하는 하위 문자열

이 배열에는 하위 문자열 일치 항목의 시작 인덱스 위치를 나타내는 index 속성도 포함되어 있습니다.

예를 들어 다음과 같은 코드를 살펴봅니다.

```
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
```

일반 표현식에 g(global) 플래그가 설정된 경우 하위 문자열을 여러 개 찾으려면 exec() 메서드를 여러 번 사용하십시오.

```
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
```

RegExp 매개 변수를 사용하는 String 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

String 클래스의 match(), replace(), search() 및 split() 메서드는 일반 표현식을 매개 변수로 사용합니다. 이러한 메서드에 대한 자세한 내용은 15페이지의 “문자열의 패턴 찾기 및 하위 문자열 바꾸기”를 참조하십시오.

일반 표현식 예제: Wiki 파서

Flash Player 9 이상, Adobe AIR 1.0 이상

다음에 나오는 간단한 Wiki 텍스트 변환 예제에서는 다양한 일반 표현식 사용 방법을 보여 줍니다.

- 소스 Wiki 패턴과 일치하는 텍스트 행을 적절한 HTML 출력 문자열로 변환
- 일반 표현식을 사용하여 URL 패턴을 HTML <a> 하이퍼링크 태그로 변환
- 일반 표현식을 사용하여 미국 달러 문자열(예: "$9.95")을 유로 문자열(예: "8.24 €")로 변환

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. WikiEditor 응용 프로그램 파일은 Samples/WikiEditor 폴더에 있으며 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
WikiEditor.mxml 또는 WikiEditor.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.

파일	설명
com/example/programmingas3/regExpExamples/WikiParser.as	일반 표현식을 사용하여 Wiki 입력 텍스트 패턴을 동일한 HTML 출력으로 변환하는 메서드가 포함된 클래스입니다.
com/example/programmingas3/regExpExamples/URLParser.as	일반 표현식을 사용하여 URL 문자열을 HTML <a> 하이퍼링크 태그로 변환하는 메서드가 포함된 클래스입니다.
com/example/programmingas3/regExpExamples/CurrencyConverter.as	일반 표현식을 사용하여 미국 달러 문자열을 유로 문자열로 변환하는 메서드가 포함된 클래스입니다.

WikiParser 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

WikiParser 클래스에는 Wiki 입력 텍스트를 동일한 HTML 출력으로 변환하는 메서드가 포함되어 있습니다. 이 클래스는 강력한 기능의 Wiki 변환 응용 프로그램은 아니지만 일반 표현식을 사용하여 패턴 일치 및 문자열 변환을 수행하는 몇 가지 방법을 자세히 보여 줍니다.

생성자 함수는 다음과 같이 setWikiData() 메서드와 함께 Wiki 입력 텍스트의 샘플 문자열을 초기화합니다.

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

사용자가 샘플 응용 프로그램에서 [Test] 버튼을 클릭하면 응용 프로그램에서 WikiParser 객체의 parseWikiString() 메서드를 호출하고, 이 메서드는 결과 HTML 문자열을 차례로 어셈블하는 다른 여러 메서드를 호출합니다.

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

호출된 parseBold(), parseItalic(), linesToParagraphs() 및 parseBullets() 메서드는 각각 문자열의 replace() 메서드를 사용하여 일반 표현식으로 정의된 일치 패턴을 바꿉니다. 이를 통해 Wiki 입력 텍스트를 HTML 형식 텍스트로 변환할 수 있습니다.

굵은체 및 기울임체 패턴 변환

parseBold() 메서드는 다음과 같이 Wiki 굵은체 텍스트 패턴(예: "'foo'")을 검색한 후 동일한 HTML 형식(예: foo)으로 변환합니다.

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

이때 일반 표현식의 (.?*) 부분은 정의된 두 ''' 패턴 사이에 있는 모든 문자(*)를 찾습니다. 이때 ? 한정 기호는 최장 일치가 수행되지 않도록 하는 역할을 합니다. 즉, "'aaa'" bbb "'ccc'"와 같은 문자열에 대해 첫 번째로 일치하는 문자열은 ''' 패턴으로 시작하고 끝나는 전체 문자열이 아니라 "'aaa'"가 됩니다.

일반 표현식의 괄호는 캡처 그룹을 정의하고 replace() 메서드는 대체 문자열에서 $1 코드를 사용하여 이 그룹을 참조합니다. 일반 표현식에 g(global) 플래그를 설정하고 replace() 메서드를 호출하면 문자열에서 일치하는 첫 번째 항목 및 모든 항목이 대체됩니다.

parseItalic() 메서드는 parseBold() 메서드와 비슷하지만 기울임체 텍스트에 대한 구분 기호로 세 개가 아니라 두 개의 아포스트로피(')를 확인한다는 차이가 있습니다.

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

불릿 패턴 변환

다음 예제와 같이 parseBullet() 메서드는 Wiki 불릿 행 패턴(예: * foo)을 검색한 후 동일한 HTML 형식(예: foo)으로 변환합니다.

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

일반 표현식의 맨 앞에 ^ 심볼이 있으면 행의 시작 위치를 찾으며, m(multiline) 플래그가 설정되어 있으면 ^ 심볼이 문자열의 시작 위치 및 행의 시작 위치에 대응됩니다.

* 패턴은 별표 문자를 찾습니다. 이때 * 한정 기호 대신 리터럴 별표를 나타내기 위해 백슬래시가 사용됩니다.

일반 표현식의 괄호는 캡처 그룹을 정의하고 replace() 메서드는 대체 문자열에서 $1 코드를 사용하여 이 그룹을 참조합니다. 일반 표현식에 g(global) 플래그를 설정하고 replace() 메서드를 호출하면 문자열에서 일치하는 첫 번째 항목 및 모든 항목이 대체됩니다.

Wiki 단락 패턴 변환

linesToParagraphs() 메서드는 Wiki 입력 문자열의 각 행을 HTML <p> 단락 태그로 변환합니다. 메서드의 이러한 행은 Wiki 입력 문자열에서 빈 행을 제거합니다.

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

일반 표현식에 ^ 및 $ 심볼이 있으면 행의 시작 및 끝 위치를 찾으며, m(multiline) 플래그가 설정되어 있으면 ^ 심볼이 문자열의 시작 위치 및 행의 시작 위치에 대응됩니다.

replace() 메서드는 일치하는 모든 하위 문자열(빈 행)을 빈 문자열("")로 바꿉니다. 일반 표현식에 g(global) 플래그를 설정하고 replace() 메서드를 호출하면 문자열에서 일치하는 첫 번째 항목 및 모든 항목이 대체됩니다.

URL을 HTML <a> 태그로 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 urlToATag 체크 상자를 선택한 경우 샘플 응용 프로그램에서 [Test] 버튼을 클릭하면 응용 프로그램에서 URLParser.urlToATag() 정적 메서드를 호출하여 Wiki 입력 문자열의 URL 문자열을 HTML <a> 태그로 변환합니다.

```
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>");
```

RegExp() 생성자 함수는 다양한 구성 요소에서 일반 표현식(urlPattern)을 어셈블하는 데 사용됩니다. 이러한 구성 요소는 일반 표현식 패턴의 항목을 정의하는 각 문자열입니다.

protocol 문자열로 정의된 일반 표현식 패턴의 첫 번째 항목은 URL 프로토콜(http:// 또는 ftp://)을 정의합니다. 괄호는 ? 심볼로 나타내는 비캡처 그룹을 정의합니다. 즉, 괄호는 | 패턴의 그룹을 정의하는 데만 사용되며 이 그룹은 replace() 메서드의 대체 문자열에서 역참조 코드($1, $2, $3)와 일치하지 않습니다.

일반 표현식의 다른 구성 요소는 각각 일반 표현식 패턴에서 괄호로 표시되는 캡처 그룹을 사용하고 이러한 그룹은 replace() 메서드의 대체 문자열에서 역참조 코드($1, $2, $3)에 사용됩니다.

urlPart 문자열에 의해 정의된 패턴 부분은 **a-z**, 0-9, _ 또는 - 문자 중 최소한 하나 이상을 찾습니다. + 한정 기호는 최소한 하나 이상의 문자를 찾는다는 것을 나타내고 \.는 필수 도트(.) 문자를 나타냅니다. 또한 나머지 항목은 a-z, 0-9, _ 또는 - 문자 중 최소한 하나 이상의 다른 문자열을 찾습니다.

optionalUrlPart 문자열에 의해 정의된 패턴 부분은 도트(.) 문자와 모든 영숫자 문자(_ 및 - 포함)가 차례로 나오는 항목을 0개 이상 찾습니다. * 한정 기호는 0개 이상의 문자를 찾는다는 것을 나타냅니다.

replace() 메서드를 호출하면 일반 표현식이 사용되고 역참조를 통해 대체 HTML 문자열이 어셈블됩니다.

그러면 urlToATag() 메서드에서 emailToATag() 메서드를 호출합니다. 이 메서드는 비슷한 방법을 사용하여 전자 메일 패턴을 HTML <a> 하이퍼링크 문자열로 바꿉니다. 이 샘플 파일에서는 사용자의 이해를 돕기 위해 매우 간단한 일반 표현식을 제공하여 HTTP, FTP 및 전자 메일 URL을 찾습니다. 그러나 이러한 URL을 정확하게 찾기 위해 사용되는 일반 표현식은 훨씬 복잡합니다.

미국 달러 문자열을 유로 문자열로 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 dollarToEuro 체크 상자를 선택한 경우 샘플 응용 프로그램에서 [Test] 버튼을 클릭하면 응용 프로그램에서는 다음과 같이 정적 메서드 CurrencyConverter.usdToEuro()를 호출하여 미국 달러 문자열(예: "$9.95")을 유로 문자열(예: "8.24 €")로 변환합니다.

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

첫 번째 행은 미국 달러 문자열을 찾기 위한 간단한 패턴을 정의합니다. 이때 $ 문자 앞에 백슬래시(\) 이스케이프 문자가 추가됩니다.

replace() 메서드는 일반 표현식을 패턴 매처로 사용하고 usdStrToEuroStr() 함수를 호출하여 대체 문자열, 즉 유로 값을 확인합니다.

함수 이름이 replace() 메서드의 두 번째 매개 변수로 사용되는 경우에는 호출된 함수에 다음 항목이 매개 변수로 전달됩니다.

- 문자열의 일치 부분
- 캡처한 괄호 그룹 일치 항목. 이런 방식으로 전달되는 인수의 수는 캡처한 괄호 그룹 일치 항목의 수에 따라 다릅니다. 함수 코드 내에서 arguments.length - 3을 확인하면 캡처한 괄호 그룹 일치 항목의 수를 알 수 있습니다.
- 문자열에서 검색을 시작할 인덱스 위치
- 전체 문자열

usdStrToEuroStr() 메서드는 다음과 같이 미국 달러 문자열 패턴을 유로 문자열로 변환합니다.

```
private function usdToEuro(...args):String
{
    var usd:String = args[1];
    usd = usd.replace(",", "");
    var exchangeRate:Number = 0.828017;
    var euro:Number = Number(usd) * exchangeRate;
    trace(usd, Number(usd), euro);
    const euroSymbol:String = String.fromCharCode(8364); // €
    return euro.toFixed(2) + " " + euroSymbol;
}
```

args[1]은 usdPrice 일반 표현식을 사용하여 캡처한 괄호 그룹을 나타냅니다. 이는 미국 달러 문자열의 숫자 부분, 즉 $ 심볼이 없는 달러 금액 부분입니다. 이 메서드는 환율 변환을 적용하고 결과 문자열(앞의 $ 심볼 대신 뒤에 € 심볼 추가)을 반환합니다.

6장: XML을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 E4X(ECMAScript for XML) 사양인 ECMA-357 버전 2에 기초한 클래스 그룹이 포함되어 있습니다. 이러한 클래스에서는 XML 데이터로 작업할 수 있도록 강력하고 사용하기 쉬운 기능을 제공합니다. E4X를 사용하면 이전의 프로그래밍 기법보다 빠르게 XML 데이터로 코드를 개발할 수 있습니다. 또한 코드를 쉽게 읽을 수 있다는 이점도 있습니다.

기타 도움말 항목

XML 클래스

ECMA-357 사양

XML의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

XML은 구조화된 정보를 표시하는 표준 방법으로, 컴퓨터에서 처리가 간편하고 사용자 역시 비교적 쉽게 작성하고 이해할 수 있습니다. XML은 eXtensible Markup Language의 약어이며, XML 표준은 www.w3.org/XML/에서 볼 수 있습니다.

XML은 편리하게 데이터를 범주화할 수 있는 표준 방식을 통해 데이터에 대한 읽기, 액세스 및 조작 작업을 용이하게 합니다. XML에서는 HTML에서와 유사한 트리 구조 및 태그 구조를 사용합니다. 다음은 간단한 XML 데이터의 예입니다.

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

XML 데이터는 속성 및 기타 구조적 구성 요소뿐만 아니라 다른 태그에 중첩된 태그가 있는 더 복잡한 형태일 수 있습니다. 다음은 복잡한 XML 데이터의 예입니다.

```
<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>
```

이 XML 문서에는 다른 완전한 XML 구조(예: 자식이 있는 song 태그)가 들어 있습니다. 또한 특성(예: song 태그의 tracknumber 및 length)과 같은 다른 XML 구조를 비롯하여 데이터가 아닌 태그로 구성된 태그(예: tracks 태그)를 확인할 수 있습니다.

XML 시작

XML 관련 지식이 전무하거나 거의 없다면 여기에서 소개하는 XML 데이터의 가장 일반적인 사항에 대한 설명을 참조하십시오. XML 데이터는 일반 텍스트 형식으로 작성되며, 정보를 구조화된 형식으로 구성하는 특수 구문이 사용됩니다. 일반적으로 하나의 XML 데이터 집합을 **XML 문서**라고 합니다. XML 형식에서 데이터는 계층 구조에 따라 **요소**로 구성됩니다. 요소는 하나의 데이터 항목이거나 다른 여러 요소의 컨테이너일 수 있습니다. 모든 XML 문서는 최상위 수준(기본 항목)에 하나의 요소를 갖고 있습니다. 대부분 여러 중첩 요소로 구성되는 경우가 많지만 이 루트 요소 내에는 단일 정보가 포함됩니다. 한 예로, 음악 앨범 정보가 포함된 다음 XML 문서를 살펴보겠습니다.

```
<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>
```

각 요소는 여러 **태그**, 즉 각괄호(보다 작음 및 보다 큼 기호)로 둘러싸인 요소 이름으로 구분됩니다. 요소의 시작을 나타내는 여는 태그의 이름은 다음과 같습니다.

```
<title>
```

요소의 끝을 나타내는 닫기 태그에는 다음과 같이 요소 이름 앞에 슬래시가 사용됩니다.

```
</title>
```

요소에 내용이 없는 경우에는 빈 요소(자체 닫기 요소라고도 함)로 작성될 수 있습니다. XML에서 다음 요소는

```
<lastplayed/>
```

아래 요소와 동일합니다.

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

열기 태그와 닫기 태그 사이에 포함된 내용 외에도 요소에는 열기 태그로 정의된 특성이라는 다른 값이 포함될 수 있습니다. 예를 들어 다음 XML 요소는 값이 "4:19"인 한 개의 특성(length)을 정의합니다.

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

각 XML 요소에는 단일 값 또는 하나 이상의 XML 요소가 있거나 빈 요소의 경우 내용이 없습니다.

XML에 대한 세부 정보

XML에 대해 보다 자세한 내용을 원하는 경우, 다음 웹 사이트를 비롯하여 다양한 서적 및 리소스를 활용할 수 있습니다.

- W3Schools XML 자습서: http://w3schools.com/xml/
- XMLpitstop 자습서, 토론 목록 및 기타: http://xmlpitstop.com/

XML 작업을 위한 ActionScript 클래스

ActionScript 3.0에는 XML 구조의 정보로 작업할 때 사용할 수 있는 여러 클래스가 포함되어 있습니다. 그 중 두 가지 주요 클래스는 다음과 같습니다.

- XML: 하나의 XML 요소를 나타내며, 복수의 자식 또는 단일 값 요소로 구성된 XML 문서일 수 있습니다.
- XMLList: 일련의 XML 요소를 나타냅니다. XMLList 객체는 "형제" XML 요소가 여러 개 있는 경우에 사용됩니다. 형제 XML 요소란 XML 문서의 계층 내에서 부모 및 수준이 동일한 요소를 의미합니다. 예를 들어, 다음과 같은 일련의 XML 요소(XML 문서에 포함된 요소라고 가정)와 관련된 작업에서는 XMLList 인스턴스를 사용하는 것이 가장 간편할 수 있습니다.

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

ActionScript에 포함된 Namespace 및 QName 클래스를 사용하면 XML 네임스페이스와 관련된 고급 기능을 활용할 수 있습니다. 자세한 내용은 100페이지의 “XML 네임스페이스 사용”을 참조하십시오.

XML 작업을 지원하기 위한 내장 클래스 이외에도 ActionScript 3.0에는 XML 데이터 액세스 및 조작을 위한 특수 기능을 제공하는 여러 연산자가 포함되어 있습니다. 이러한 클래스 및 연산자를 사용한 XML 작업 방법은 E4X(ECMAScript for XML)라고 하며 ECMA-357 Edition 2 사양에 정의되어 있습니다.

중요한 개념 및 용어

다음 참조 목록에는 XML 처리 루틴을 프로그래밍할 때 사용되는 중요한 용어가 포함되어 있습니다.

요소 XML 문서에 있는 단일 항목으로, 시작 태그 및 끝 태그 사이의 내용 및 태그로 식별됩니다. XML 요소는 텍스트 데이터 또는 기타 요소로 구성되거나 비어 있을 수 있습니다.

비어 있는 요소 자식 요소가 없는 XML 요소입니다. 비어 있는 요소는 자체 닫기 태그(예: <element/>)라고도 합니다.

문서 하나의 XML 구조를 나타냅니다. XML 문서는 여러 개의 요소 또는 비어 있는 하나의 요소로 구성될 수 있지만, 문서의 기타 모든 요소를 포함하는 최상위 수준 요소가 반드시 하나씩 있어야 합니다.

노드 XML 요소를 지칭하는 다른 이름입니다.

특성 요소와 관련이 있으며 이름을 갖는 값으로, 요소 내에 중첩된 별도의 자식 요소로 작성되지 않고 요소의 여는 태그에 attributename="value" 형식과 같이 작성됩니다.

E4X를 사용하여 XML 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

ECMAScript for XML 사양은 XML 데이터 작업을 위한 클래스 및 기능 집합을 정의합니다. 이러한 클래스와 기능을 하나로 묶어 **E4X**라고 합니다. ActionScript 3.0에는 XML, XMLList, QName 및 Namespace와 같은 E4X 클래스가 포함되어 있습니다.

E4X 클래스의 메서드, 속성 및 연산자는 다음을 구현할 수 있도록 디자인되었습니다.

- 단순성 - 가능한 경우 E4X를 사용하면 XML 데이터 작업을 위한 코드를 쉽게 작성하고 이해할 수 있습니다.
- 일관성 - E4X에 내포되어 있는 이론 및 메서드는 내부적으로 일관되고 ActionScript의 다른 요소와 일치합니다.
- 익숙함 - 도트(.) 연산자와 같이 잘 알려진 연산자를 사용하여 XML 데이터를 조작합니다.

참고: ActionScript 2.0에는 다른 XML 클래스가 있습니다. ActionScript 3.0에서는 이 클래스가 E4X에 포함된 ActionScript 3.0 XML 클래스와 충돌하지 않도록 XMLDocument로 이름이 바뀌었습니다. ActionScript 3.0에서는 이전 클래스, 즉 XMLDocument, XMLNode, XMLParser 및 XMLTag가 이전 버전을 지원하기 위해 flash.xml 패키지에 포함되어 있습니다. 새로 추가된 E4X 클래스는 기본 클래스이므로 사용할 때 패키지를 가져올 필요가 없습니다. 이전 ActionScript 2.0 XML 클래스에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 flash.xml 패키지를 참조하십시오.

다음은 E4X를 사용하여 데이터를 조작하는 예제입니다.

```
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>
```

대개 응용 프로그램에서는 웹 서비스 또는 RSS 피드와 같은 외부 소스에서 XML 데이터를 로드합니다. 그러나 사용자가 쉽게 이해할 수 있도록 여기에서 제공된 코드 예제는 XML 데이터를 리터럴로 할당합니다.

다음 코드와 같이 E4X에는 도트(.) 및 특성 식별자(@) 연산자처럼 XML에서 속성과 특성에 액세스하는 데 사용할 수 있도록 직관적인 연산자가 포함되어 있습니다.

```
trace(myXML.item[0].menuName); // Output: burger
trace(myXML.item.(@id==2).menuName); // Output: fries
trace(myXML.item.(menuName=="burger").price); // Output: 3.95
```

다음 코드 예제와 같이 appendChild() 메서드를 사용하면 새 자식 노드를 XML에 지정할 수 있습니다.

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

myXML.appendChild(newItem);
```

다음과 같이 @ 및 . 연산자를 사용하면 데이터를 읽을 수 있을 뿐만 아니라 지정할 수도 있습니다.

```
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";
```

다음과 같이 for 루프를 사용하면 XML의 노드를 반복할 수 있습니다.

```
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 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 객체는 XML 요소, 특성, 주석, 처리 명령 또는 텍스트 요소를 나타낼 수 있습니다.

XML 객체는 **간단한 내용**을 포함하는 객체 또는 **복잡한 내용**을 포함하는 객체로 분류됩니다. 자식 노드가 있는 XML 객체는 복잡한 내용을 포함하는 객체로 분류되고 특성, 주석, 처리 명령 또는 텍스트 노드 중 하나에 해당하는 XML 객체는 간단한 내용을 포함하는 객체로 분류됩니다.

예를 들어, 다음 XML 객체에는 주석과 처리 명령 등의 복잡한 내용이 포함되어 있습니다.

```
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>
```

이제 다음 예제와 같이 comments() 및 processingInstructions() 메서드를 사용하여 새 XML 객체, 즉 주석과 처리 명령을 만들 수 있습니다.

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

XML 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 클래스에는 다음과 같은 다섯 개의 정적 속성이 포함되어 있습니다.

- ignoreComments 및 ignoreProcessingInstructions 속성은 XML 객체를 파싱할 때 주석 또는 처리 명령을 무시할지 여부를 결정합니다.
- ignoreWhitespace 속성은 공백 문자로만 구분되는 포함된 표현식 및 요소 태그에서 공백 문자를 무시할지 여부를 결정합니다.
- prettyIndent 및 prettyPrinting 속성은 XML 클래스의 toString() 및 toXMLString() 메서드에서 반환하는 텍스트의 서식을 지정하는 데 사용됩니다.

이러한 속성에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

XML 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 메서드를 사용하면 계층 구조 형식의 XML 객체로 작업할 수 있습니다.

- appendChild()
- child()
- childIndex()
- children()
- descendants()
- elements()
- insertChildAfter()
- insertChildBefore()
- parent()
- prependChild()

다음 메서드를 사용하면 XML 객체 특성으로 작업할 수 있습니다.

- attribute()
- attributes()

다음 메서드를 사용하면 XML 객체 속성으로 작업할 수 있습니다.

- hasOwnProperty()
- propertyIsEnumerable()
- replace()
- setChildren()

다음 메서드를 사용하면 정규화된 이름 및 네임스페이스로 작업할 수 있습니다.

- addNamespace()
- inScopeNamespaces()
- localName()
- name()

- namespace()
- namespaceDeclarations()
- removeNamespace()
- setLocalName()
- setName()
- setNamespace()

다음 메서드를 사용하면 특정 유형의 XML 내용을 결정하고 작업할 수 있습니다.

- comments()
- hasComplexContent()
- hasSimpleContent()
- nodeKind()
- processingInstructions()
- text()

다음 메서드를 사용하면 XML 객체를 문자열로 변환하고 서식을 지정할 수 있습니다.

- defaultSettings()
- setSettings()
- settings()
- normalize()
- toString()
- toXMLString()

다음과 같은 메서드도 추가로 사용할 수 있습니다.

- contains()
- copy()
- valueOf()
- length()

이러한 메서드에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

XMLList 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

XMLList 인스턴스는 임의의 XML 객체 컬렉션을 나타냅니다. 여기에는 전체 XML 문서, XML 조각 또는 XML 쿼리 결과 등이 포함될 수 있습니다.

다음 메서드를 사용하면 계층 구조 형식의 XMLList 객체로 작업할 수 있습니다.

- child()
- children()
- descendants()

- elements()
- parent()

다음 메서드를 사용하면 XMLList 객체 특성으로 작업할 수 있습니다.

- attribute()
- attributes()

다음 메서드를 사용하면 XMLList 속성으로 작업할 수 있습니다.

- hasOwnProperty()
- propertyIsEnumerable()

다음 메서드를 사용하면 특정 유형의 XML 내용을 결정하고 작업할 수 있습니다.

- comments()
- hasComplexContent()
- hasSimpleContent()
- processingInstructions()
- text()

다음 메서드를 사용하면 XMLList 객체를 문자열로 변환하고 서식을 지정할 수 있습니다.

- normalize()
- toString()
- toXMLString()

다음과 같은 메서드도 추가로 사용할 수 있습니다.

- contains()
- copy()
- length()
- valueOf()

이러한 메서드에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

XML 요소가 하나만 포함된 XMLList 객체는 XML 객체와 같은 것으로 간주되므로 XML 클래스의 모든 속성과 메서드를 사용할 수 있습니다. 예를 들어 다음 코드에서 doc.div는 요소가 하나만 포함된 XMLList 객체이므로 XML 클래스에서 appendChild() 메서드를 사용할 수 있습니다.

```
var doc:XML =
        <body>
            <div>
                <p>Hello</p>
            </div>
        </body>;
doc.div.appendChild(<p>World</p>);
```

XML 속성 및 메서드 목록을 보려면 90페이지의 “XML 객체”를 참조하십시오.

XML 변수 초기화

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같이 XML 객체에 XML 리터럴을 지정할 수 있습니다.

```
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>
```

다음 코드 예제와 같이 new 생성자를 사용하여 XML 데이터가 포함된 문자열에서 XML 객체의 인스턴스를 만들 수도 있습니다.

```
var str:String = "<order><item id='1'><menuName>burger</menuName>"
                            + "<price>3.95</price></item></order>";
var myXML:XML = new XML(str);
```

닫는 태그가 없는 것과 같이 문자열의 XML 데이터 형식이 잘못된 경우에는 런타임 오류가 발생합니다.

다음 예제와 같이 다른 변수의 데이터를 참조로 XML 객체에 전달할 수도 있습니다.

```
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>
```

URL에서 XML 데이터를 로드하려면 다음 예제와 같이 URLLoader 클래스를 사용합니다.

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

소켓 연결에서 XML 데이터를 읽으려면 XMLSocket 클래스를 사용합니다. 자세한 내용은 ActionScript 3.0 Reference for the Adobe Flash Platform에서 XMLSocket 클래스를 참조하십시오.

XML 객체 어셈블 및 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 객체 속성 목록의 맨 앞이나 끝에 속성을 추가하려면 다음 예제와 같이 prependChild() 메서드 또는 appendChild() 메서드를 사용합니다.

```
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>
```

지정된 속성의 앞이나 뒤에 속성을 추가하려면 다음과 같이 insertChildBefore() 메서드 또는 insertChildAfter() 메서드를 사용합니다.

```
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>)
```

다음 예제와 같이 XML 객체를 구성할 때 중괄호 연산자({ 및 })를 사용하여 다른 변수의 데이터를 참조로 전달할 수도 있습니다.

```
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)
}
```

다음과 같이 = 연산자를 사용하여 속성 및 특성을 XML 객체에 지정할 수 있습니다.

```
var x:XML =
    <employee>
        <lastname>Smith</lastname>
    </employee>
x.firstname = "Jean";
x.@id = "239";
```

이렇게 하면 XML 객체 x가 다음과 같이 설정됩니다.

```
<employee id="239">
    <lastname>Smith</lastname>
    <firstname>Jean</firstname>
</employee>
```

+ 및 += 연산자를 사용하면 XMLList 객체를 연결할 수 있습니다.

```
var x1:XML = <a>test1</a>
var x2:XML = <b>test2</b>
var xList:XMLList = x1 + x2;
xList += <c>test3</c>
```

이렇게 하면 XMLList 객체 xList가 다음과 같이 설정됩니다.

```
<a>test1</a>
<b>test2</b>
<c>test3</c>
```

XML 구조 순회

Flash Player 9 이상, Adobe AIR 1.0 이상

XML의 강력한 기능 중 하나는 선형 텍스트 문자열을 통해 복잡한 중첩 데이터를 제공하는 것입니다. 데이터를 XML 객체로 로드하면 ActionScript에서 해당 데이터를 파싱하고 계층 구조를 메모리로 로드합니다. 이때 XML 데이터 형식이 잘못된 경우에는 런타임 오류가 발생합니다.

XML 및 XMLList 객체의 연산자와 메서드를 사용하면 XML 데이터 구조를 쉽게 순회할 수 있습니다.

도트(.) 연산자 및 자손 접근자(..) 연산자를 사용하면 XML 객체의 자식 속성에 액세스할 수 있습니다. 다음과 같은 XML 객체를 검토하십시오.

```
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>
```

myXML.book 객체는 이름이 book인 myXML 객체의 자식 속성을 포함하는 XMLList 객체입니다. 이러한 두 XML 객체는 myXML 객체의 두 book 속성에 대응됩니다.

myXML..lastName 객체는 이름이 lastName인 하위 속성을 포함하는 XMLList 객체입니다. 이러한 두 XML 객체는 myXML 객체의 두 lastName 속성에 대응됩니다.

myXML.book.editor.lastName 객체는 lastName이라는 자식 속성을 포함하는 XMLList 객체입니다. 여기에서 lastName은 editor의 자식 속성이며 editor는 book의 자식 속성입니다. 또한 book은 myXML 객체의 자식입니다. 이 예제의 경우에는XML 객체(값이 "Case"로 설정된 lastName 속성)를 하나만 포함하는 XMLList 객체입니다.

부모 및 자식 노드 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

parent() 메서드는 XML 객체의 부모를 반환합니다.

자식 목록의 서수 인덱스 값을 사용하여 특정 자식 객체에 액세스할 수 있습니다. 예를 들어 myXML이라는 XML 객체에 이름이 book인 자식 속성이 두 개 포함되어 있는 경우를 가정해 봅니다. 이때 각 자식 속성 book에는 인덱스 번호가 연결되어 있습니다.

```
myXML.book[0]
myXML.book[1]
```

특정 손자 항목에 액세스하려면 자식 이름과 손자 이름에 모두 인덱스 번호를 지정해야 합니다.

```
myXML.book[0].title[0]
```

그러나 이름이 title인 x.book[0]의 자식이 하나뿐인 경우에는 다음과 같이 인덱스 참조를 생략할 수 있습니다.

```
myXML.book[0].title
```

마찬가지로 x 객체의 book 자식이 하나뿐이고 해당 자식 객체에 title 객체가 하나뿐인 경우에는 다음과 같이 두 인덱스 참조를 모두 생략할 수 있습니다.

```
myXML.book.title
```

다음 예제와 같이 child() 메서드를 사용하여 변수 또는 표현식을 기반으로 이름에 따라 자식 항목을 탐색할 수 있습니다.

```
var myXML:XML =
        <order>
            <book>
                <title>Dictionary</title>
            </book>
        </order>;

var childName:String = "book";

trace(myXML.child(childName).title) // output: Dictionary
```

특성 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 코드와 같이 XML 또는 XMLList 객체에서 특성에 액세스하려면 @ 심볼(특성 식별자 연산자)을 사용합니다.

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@id); // 6401
```

다음 코드와 같이 * 와일드카드 심볼을 @ 심볼과 함께 사용하면 XML 또는 XMLList 객체의 모든 특성에 액세스할 수 있습니다.

```
var employee:XML =
    <employee id="6401" code="233">
        <lastName>Wu</lastName>
        <firstName>Erin</firstName>
    </employee>;
trace(employee.@*.toXMLString());
// 6401
// 233
```

다음 코드와 같이 attribute() 또는 attributes() 메서드를 사용하면 XML 또는 XMLList 객체의 특정 특성이나 모든 특성에 액세스할 수 있습니다.

```
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
```

다음 예제와 같이 아래 구문을 사용하여 특성에 액세스할 수도 있습니다.

```
employee.attribute("id")
employee["@id"]
employee.@["id"]
```

이러한 구문은 각각 employee.@id. employee.@id 구문을 사용하는 것이 좋습니다.

특성 또는 요소 값으로 필터링

Flash Player 9 이상, Adobe AIR 1.0 이상

괄호 연산자인 (및)를 사용하면 특정 요소 이름 또는 특성 값으로 요소를 필터링할 수 있습니다. 다음과 같은 XML 객체를 검토하십시오.

```
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>
```

다음 표현식은 모두 유효합니다.

- x.employee.(lastName == "McGee") - 두 번째 employee 노드입니다.
- x.employee.(lastName == "McGee").firstName - 두 번째 employee 노드의 firstName 속성입니다.
- x.employee.(lastName == "McGee").@id - 두 번째 employee 노드의 id 특성 값입니다.
- x.employee.(@id == 347) - 첫 번째 employee 노드입니다.
- x.employee.(@id == 347).lastName - 첫 번째 employee 노드의 lastName 속성입니다.
- x.employee.(@id > 300) - 두 employee 속성이 모두 포함된 XMLList입니다.
- x.employee.(position.toString().search("analyst") > -1) - 두 position 속성이 모두 포함된 XMLList입니다.

존재하지 않는 특성 또는 요소를 필터링하려고 하면 예외가 발생합니다. 예를 들어 다음 코드의 경우 두 번째 p 요소에 id 특성이 없으므로 마지막 행에서 오류가 발생합니다.

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(@id == '123'));
```

마찬가지로 다음 코드에서도 두 번째 p 요소의 b 속성이 없으므로 마지막 줄에서 오류가 발생합니다.

```
var doc:XML =
            <body>
                <p id='123'>Hello, <b>Bob</b>.</p>
                <p>Hello.</p>
            </body>;
trace(doc.p.(b == 'Bob'));
```

이러한 오류가 발생하지 않게 하려면 다음 코드와 같이 attribute() 및 elements() 메서드를 사용하여 대응하는 특성 또는 요소가 있는 속성을 식별해야 합니다.

```
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'));
```

다음 코드와 같이 hasOwnProperty() 메서드를 사용할 수도 있습니다.

```
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'));
```

for..in 및 for each..in 문 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 XMLList 객체를 반복할 수 있도록 for..in 문과 for each..in 문이 포함되어 있습니다. 예를 들어 myXML이라는 XML 객체와 myXML.item이라는 XMLList 객체가 있다고 가정해 봅니다. 이때 XMLList 객체 myXML.item은 XML 객체의 두 item 노드로 구성되어 있습니다.

```
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>;
```

for..in 문을 사용하면 XMLList의 속성 이름 집합을 반복할 수 있습니다.

```
var total:Number = 0;
for (var pname:String in myXML.item)
{
    total += myXML.item.@quantity[pname] * myXML.item.price[pname];
}
```

for each..in 문을 사용하면 XMLList의 속성을 반복할 수 있습니다.

```
var total2:Number = 0;
for each (var prop:XML in myXML.item)
{
    total2 += prop.@quantity * prop.price;
}
```

XML 네임스페이스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 객체 또는 문서의 네임스페이스는 해당 객체에 포함된 데이터의 유형을 식별합니다. 예를 들어, SOAP 메시징 프로토콜을 사용하는 웹 서비스에 XML 데이터를 전송하는 경우 다음과 같이 XML의 여는 태그에 네임스페이스를 선언합니다.

```
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>;
```

이 네임스페이스에는 접두어 soap와 네임스페이스를 정의하는 URI http://schemas.xmlsoap.org/soap/envelope/이 포함되어 있습니다.

ActionScript 3.0에는 XML 네임스페이스를 사용하여 작업할 수 있도록 Namespace 클래스가 포함되어 있습니다. 위의 예제에 제공된 XML 객체의 경우 다음과 같이 Namespace 클래스를 사용할 수 있습니다.

```
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";
```

XML 클래스에는 네임스페이스를 사용하여 작업할 수 있도록 addNamespace(), inScopeNamespaces(), localName(), name(), namespace(), namespaceDeclarations(), removeNamespace(), setLocalName(), setName() 및 setNamespace() 등의 메서드가 포함되어 있습니다.

default xml namespace 지시문을 사용하면 XML 객체의 기본 네임스페이스를 지정할 수 있습니다. 다음 예제 코드에서는 x1과 x2 둘 다에 동일한 기본 네임스페이스가 지정되어 있습니다.

```
var ns1:Namespace = new Namespace("http://www.example.com/namespaces/");
default xml namespace = ns1;
var x1:XML = <test1 />;
var x2:XML = <test2 />;
```

XML 유형 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 객체 및 XMLList 객체를 문자열 값으로 변환할 수 있습니다. 마찬가지로 문자열을 XML 객체 및 XMLList 객체로 변환할 수도 있습니다. 또한 모든 XML 특성 값, 이름, 텍스트 값 등은 문자열이라는 사실을 기억하십시오. 다음 단원에서는 이러한 모든 XML 유형 변환 형식에 대해 설명합니다.

XML 및 XMLList 객체를 문자열로 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 및 XMLList 클래스에는 toString() 메서드와 toXMLString() 메서드가 포함되어 있습니다. toXMLString() 메서드는 XML 객체의 모든 태그, 특성, 네임스페이스 선언 및 내용이 포함된 문자열을 반환합니다. 복잡한 내용, 즉 자식 요소를 포함하는 XML 객체의 경우 toString() 메서드는 toXMLString() 메서드와 완전히 동일하고, 간단한 내용, 즉 텍스트 요소 하나만 포함하는 XML 객체의 경우 toString() 메서드는 다음 예제와 같이 해당 요소의 텍스트 내용만 반환합니다.

```
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
```

이 코드와 같이 toString() 또는 toXMLString()을 지정하지 않고 trace() 메서드를 사용하면 기본적으로 toString() 메서드를 사용하여 데이터가 변환됩니다.

```
var myXML:XML =
    <order>
        <item id='1' quantity='2'>
            <menuName>burger</menuName>
            <price>3.95</price>
        </item>
    <order>;

trace(myXML.item[0].menuName);
    // burger
```

trace() 메서드를 사용하여 코드를 디버깅하는 경우에는 trace() 메서드에서 좀 더 완전한 데이터를 출력하도록 toXMLString() 메서드를 사용할 수 있습니다.

문자열을 XML 객체로 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같이 new XML() 생성자를 사용하여 문자열에서 XML 객체를 만들 수 있습니다.

```
var x:XML = new XML("<a>test</a>");
```

유효하지 않거나 형식이 잘못된 XML을 나타내는 문자열을 XML로 변환하려고 하면 다음과 같이 런타임 오류가 발생합니다.

```
var x:XML = new XML("<a>test"); // throws an error
```

문자열 형식의 특성 값, 이름 및 텍스트 값 변환

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 XML 특성 값, 이름 및 텍스트 값은 String 데이터 유형이며 이러한 값을 다른 데이터 유형으로 변환해야 할 수 있습니다. 예를 들어 다음 코드에서는 Number() 함수를 사용하여 텍스트 값을 숫자로 변환합니다.

```
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;
```

이 코드에서 Number() 함수를 사용하지 않은 경우에는 + 연산자를 문자열 연결 연산자로 해석하므로 마지막 줄의 trace() 메서드에 의해 다음과 같이 출력됩니다.

```
01.003.95
```

외부 XML 문서 읽기

Flash Player 9 이상, Adobe AIR 1.0 이상

URLLoader 클래스를 사용하면 URL에서 XML 데이터를 로드할 수 있습니다. 응용 프로그램에서 다음 코드를 사용하려면 예제에 나오는 XML_URL 값을 올바른 URL로 바꾸십시오.

```
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.");
}
```

XMLSocket 클래스를 사용하여 서버와의 비동기 XML 소켓 연결을 설정할 수도 있습니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

ActionScript의 XML 예제: 인터넷에서 RSS 데이터 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

RSSViewer 샘플 응용 프로그램에서는 다음과 같이 ActionScript에서 XML을 사용하여 작업할 수 있는 다양한 기능을 보여 줍니다.

- XML 메서드를 사용하여 RSS 피드 형식으로 XML 데이터를 순회합니다.
- XML 메서드를 사용하여 XML 데이터를 텍스트 필드에 사용하기 위해 HTML 형식으로 어셈블합니다.

RSS 형식은 XML을 통해 뉴스를 게시하는 데 주로 사용됩니다. 다음은 간단한 RSS 데이터 파일입니다.

```
<?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>
```

SimpleRSS 응용 프로그램은 인터넷에서 RSS 데이터를 읽은 후 헤드라인(제목), 링크 및 설명을 위해 데이터를 파싱한 다음 해당 데이터를 반환합니다. SimpleRSSUI 클래스는 UI를 제공하고, 모든 XML 처리 작업을 실행하는 SimpleRSS 클래스를 호출합니다.

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. RSSViewer 응용 프로그램 파일은 Samples/RSSViewer 폴더에 있으며 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
RSSViewer.mxml 또는 RSSViewer.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/rssViewer/RSSParser.as	E4X를 사용하여 RSS(XML) 데이터를 순회하고 이에 해당하는 HTML 표현을 생성하는 메서드가 포함된 클래스입니다.
RSSData/ak.rss	샘플 RSS 파일입니다. 이 응용 프로그램은 Adobe에서 호스팅하는 Flex RSS 피드의 웹에서 RSS 데이터를 읽도록 설정되어 있습니다. 그러나 Flex RSS 피드의 스키마와 약간 다른 스키마를 사용하는 이 문서에서 RSS 데이터를 읽도록 응용 프로그램을 쉽게 변경할 수 있습니다.

XML 데이터 읽기 및 파싱

Flash Player 9 이상, Adobe AIR 1.0 이상

RSSParser 클래스에는 rssXML 변수에 저장된 입력 RSS 데이터를 HTML 형식 출력(rssOutput)이 들어 있는 문자열로 변환하는 xmlLoaded() 메서드가 포함되어 있습니다.

소스 RSS 데이터에 기본 네임스페이스가 포함되어 있는 경우 코드에서는 메서드의 시작 부분에서 기본 XML 네임스페이스를 설정합니다.

```
if (rssXML.namespace("") != undefined)
{
    default xml namespace = rssXML.namespace("");
}
```

그러면 다음 줄에서 소스 XML 데이터의 내용을 반복하여 item이라는 각 하위 속성을 검사합니다.

```
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);
}
```

처음 세 줄에서는 문자열 변수를 설정하여 XML 데이터에 대한 item 속성의 제목, 설명 및 링크 속성을 나타냅니다. 그러면 다음 줄에서 buildItemHTML() 메서드를 호출하여 HTML 데이터를 XMLList 객체 형식으로 가져옵니다. 이때 세 개의 새 문자열 변수를 매개 변수로 사용합니다.

XMLList 데이터 어셈블

Flash Player 9 이상, Adobe AIR 1.0 이상

HTML 데이터(XMLList 객체)는 다음과 같은 형식으로 되어 있습니다.

```
<b>itemTitle</b>
<p>
    itemDescription
    <br />
    <a href="link">
        <font color="#008000">More...</font>
    </a>
</p>
```

메서드의 첫 번째 줄에서는 기본 xml 네임스페이스를 제거합니다.

```
default xml namespace = new Namespace();
```

default xml namespace 지시문에는 함수 블록 수준 범위가 있습니다. 즉, 이 선언의 범위가 buildItemHTML() 메서드입니다.

그 다음에 나오는 줄에서는 함수에 전달된 문자열 인수에 따라 XMLList를 어셈블합니다.

```
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;
```

이 XMLList 객체는 ActionScript HTML 텍스트 필드에 적합한 문자열 데이터를 나타냅니다.

xmlLoaded() 메서드는 buildItemHTML() 메서드의 반환값을 사용하고 이 값을 문자열로 변환합니다.

```
XML.prettyPrinting = false;
rssOutput = outXML.toXMLString();
```

RSS 피드 제목 추출 및 사용자 정의 이벤트 보내기

Flash Player 9 이상, Adobe AIR 1.0 이상

xmlLoaded() 메서드는 소스 RSS XML 데이터의 정보를 기초로 rssTitle 문자열 변수를 설정합니다.

```
rssTitle = rssXML.channel.title.toString();
```

마지막으로 xmlLoaded() 메서드는 데이터가 파싱되어 사용할 수 있음을 응용 프로그램에 알리는 이벤트를 생성합니다.

```
            dataWritten = new Event("dataWritten", true);
```

7장: 기본 JSON 기능 사용

ActionScript 3.0에서는 JSON(JavaScript Object Notation) 형식을 사용하여 ActionScript 객체를 인코딩 및 디코딩하기 위한 기본 API를 제공합니다. JSON 클래스 및 지원 멤버 함수는 변경 내용이 거의 없는 ECMA-262 제5판 사양을 따릅니다.

커뮤니티 멤버인 Todd Anderson은 기본 JSON API와 타사 as3corelib JSON 클래스를 비교한 결과를 내놓았습니다. 자세한 내용은 Flash Player 11에서의 기본 JSON 작업을 참조하십시오.

기타 도움말 항목

JSON

JSON API 개요

ActionScript JSON API는 JSON 클래스와 몇 개의 기본 클래스에 대한 toJSON() 멤버 함수로 구성됩니다. 임의의 클래스에 대한 사용자 정의 JSON 인코딩이 필요한 응용 프로그램의 경우 ActionScript 프레임워크는 기본 인코딩을 재정의하는 방법을 제공합니다.

JSON 클래스는 toJSON() 멤버를 제공하지 않는 ActionScript 클래스의 가져오기 및 내보내기를 내부에서 처리합니다. 그러한 경우 JSON은 각 객체에서 발견되는 공용 속성을 순회합니다. 객체에 다른 객체가 포함된 경우 JSON은 중첩된 객체를 재귀적으로 검색하고 동일한 순회를 수행합니다. toJSON() 메서드를 제공하는 객체가 있는 경우 JSON은 내부 알고리즘 대신에 해당 사용자 정의 메서드를 사용합니다.

JSON 인터페이스는 인코딩 메서드 stringify() 및 디코딩 메서드 parse()로 구성됩니다. 이러한 각 메서드는 고유한 논리를 JSON 인코딩 또는 디코딩 작업 과정에 삽입할 수 있도록 하는 매개 변수를 제공합니다. stringify()의 경우 이 매개 변수는 replacer이고 parse()의 경우에는 reviver입니다. 이러한 매개 변수는 다음과 같은 서명을 통해 두 개의 인수가 포함된 함수 정의를 사용합니다.

```
function(k, v):*
```

toJSON() 메서드

toJSON() 메서드의 서명은 다음과 같습니다.

```
public function toJSON(k:String):*
```

JSON.stringify()는 객체를 순회하는 동안 발견된 각 공용 속성에서 toJSON()(있을 경우)을 호출합니다. 속성은 키-값 쌍으로 구성됩니다. stringify()는 toJSON()을 호출할 경우 현재 검사 중인 속성의 키 k를 전달합니다. 일반적인 toJSON() 구현에서는 각 속성 이름을 평가하고 해당 값의 필요한 인코딩을 반환합니다.

toJSON() 메서드는 단순히 문자열이 아닌 모든 유형의 값(*로 표시)을 반환할 수 있습니다. 이 변수 반환 유형에서는 toJSON()이 해당되는 경우 객체를 반환할 수 있습니다. 예를 들어 사용자 정의 클래스의 속성에 다른 타사 라이브러리의 객체가 포함된 경우 toJSON()이 속성을 발견할 때 해당 객체가 반환될 수 있습니다. 이 경우 JSON은 타사 객체를 재귀적으로 검색합니다. 인코딩 프로세스 흐름은 다음과 같이 작동합니다.

- toJSON()이 문자열로 평가되지 않는 객체를 반환할 경우 stringify()는 해당 객체를 재귀적으로 검색합니다.
- toJSON()이 문자열을 반환하는 경우 stringify()는 이 값을 다른 문자열에 래핑하고, 래핑된 문자열을 반환한 후 다음 값으로 이동합니다.

대부분의 경우 사용자의 응용 프로그램에서 만든 JSON 문자열을 반환하는 대신 객체를 반환하는 것이 낫습니다. 객체를 반환하면 기본 제공 JSON 인코딩 알고리즘을 활용하는 것은 물론 JSON에서 중첩된 객체를 재귀적으로 검색하도록 할 수 있습니다.

toJSON() 메서드는 Object 클래스 또는 대부분의 다른 기본 클래스에 정의되지 않습니다. 이 메서드가 없으면 JSON은 객체의 공용 속성에 대해 표준 순회를 수행합니다. 또한 원하는 경우 toJSON()을 사용하여 객체의 전용 속성을 노출할 수도 있습니다.

일부 기본 클래스에는 ActionScript 라이브러리가 모든 경우에 효과적으로 해결할 수 없는 문제가 있습니다. 이러한 클래스의 경우 ActionScript는 클라이언트가 각자의 요구에 맞게 다시 구현할 수 있는 간단한 구현을 제공합니다. 간단한 toJSON() 멤버를 제공하는 클래스는 다음과 같습니다.

- ByteArray
- DATE
- Dictionary
- XML

ByteArray 클래스를 하위 클래스로 지정하여 toJSON() 메서드를 대체할 수도 있고, 해당 프로토타입을 다시 정의할 수도 있습니다. final로 선언되는 Date 및 XML 클래스를 사용하려면 클래스 프로토타입을 활용하여 toJSON()을 다시 정의해야 합니다. Dictionary 클래스는 dynamic으로 선언되므로 보다 자유롭게 toJSON()을 대체하여 사용할 수 있습니다.

사용자 정의 JSON 비헤이비어 정의

기본 클래스에 대한 JSON 인코딩 및 디코딩을 직접 구현하기 위해 다음과 같은 여러 가지 옵션 중에서 선택할 수 있습니다.

- final이 아닌 기본 클래스의 사용자 정의 하위 클래스에서 toJSON() 정의 또는 재정의
- 클래스 프로토타입에서 toJSON() 정의 또는 다시 정의
- 동적 클래스에서 toJSON 속성 정의
- JSON.stringify() replacer 및 JSON.parser() reviver 매개 변수 사용

기타 도움말 항목

ECMA-262, 제 5판

기본 제공 클래스의 프로토타입에서 toJSON() 정의

ActionScript의 기본 JSON 구현은 ECMA-262, 제 5판에 정의된 ECMAScript JSON 메커니즘을 미러링합니다. ECMAScript가 클래스를 지원하지 않으므로 ActionScript는 프로토타입 기반 전달과 관련하여 JSON 비헤이비어를 정의합니다. 프로토타입은 ActionScript 3.0 클래스의 이전 버전이며, 시뮬레이션된 상속과 멤버 추가 및 재정의를 허용합니다.

ActionScript에서는 모든 클래스의 프로토타입에서 toJSON()을 정의하거나 다시 정의할 수 있습니다. 이러한 권한은 final로 표시된 클래스에도 적용됩니다. 클래스 프로토타입에서 toJSON()을 정의할 경우 응용 프로그램 범위 내에 있는 해당 클래스의 모든 인스턴스에 해당 정의가 적용됩니다. 예를 들어 여기서는 MovieClip 프로토타입에서 toJSON() 메서드를 정의하는 방법을 보여 줍니다.

```
MovieClip.prototype.toJSON = function(k):* {
    trace("prototype.toJSON() called.");
    return "toJSON";
}
```

응용 프로그램이 임의의 MovieClip 인스턴스에서 stringify()를 호출할 경우 stringify()는 toJSON() 메서드의 출력을 반환합니다.

```
var mc:MovieClip = new MovieClip();
var js:String = JSON.stringify(mc); //"prototype toJSON() called."
trace("js: " + js); //"js: toJSON"
```

또한 이 메서드를 정의하는 기본 클래스에서 toJSON()을 재정의할 수 있습니다. 예를 들어 다음 코드는 Date.toJSON()을 재정의합니다.

```
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"
```

클래스 수준에서 toJSON() 정의 또는 재정의

응용 프로그램에서 프로토타입을 사용하여 toJSON()을 다시 정의할 필요가 없는 경우도 있습니다. 또한 부모 클래스가 final로 표시되지 않은 경우 toJSON()을 하위 클래스의 멤버로 정의할 수도 있습니다. 예를 들어 ByteArray 클래스를 확장하고 공용 toJSON() 함수를 정의할 수 있습니다.

```
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"
```

클래스가 dynamic인 경우에는 다음과 같이 toJSON 속성을 해당 클래스의 객체에 추가하고 함수를 할당할 수 있습니다.

```
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."}
```

임의의 ActionScript 클래스에서 toJSON()을 재정의하거나 정의하거나 다시 정의할 수 있습니다. 하지만 대부분의 기본 제공 ActionScript 클래스는 toJSON()을 정의하지 않습니다. Object 클래스는 toJSON을 기본 프로토타입에 정의하거나 클래스 멤버로 선언하지 않습니다. 소수의 기본 클래스만 이 메서드를 프로토타입 함수로 정의합니다. 따라서 대부분의 경우 toJSON()을 전통적인 의미로 재정의할 수 없습니다.

toJSON()을 정의하지 않는 기본 클래스는 내부 JSON 구현을 통해 JSON에 직렬화됩니다. 가능하면 이 기본 제공 기능을 대체하지 마십시오. toJSON() 멤버를 정의하는 경우 JSON 클래스에서는 고유한 기능 대신 사용자의 논리를 활용합니다.

JSON.stringify() replacer 매개 변수 사용

프로토타입에서 toJSON()을 대체하면 응용 프로그램 전체에 걸쳐 클래스의 JSON 내보내기 비헤이비어를 변경하는 데 도움이 됩니다. 하지만 경우에 따라 내보내기 논리가 일시적인 조건에서 특수 클래스에만 적용될 수도 있습니다. 이러한 작은 범위의 변경을 수용하려면 JSON.stringify() 메서드의 replacer 매개 변수를 사용하면 됩니다.

stringify() 메서드는 replacer 매개 변수를 통해 전달된 함수를 인코딩 중인 객체에 적용합니다. 이 함수의 서명은 toJSON()의 서명과 비슷합니다.

```
function (k,v):*
```

toJSON()과 달리 replacer 함수에는 v 값과 k 키가 필요합니다. 이러한 차이가 필요한 이유는 stringify()가 인코딩 중인 객체 대신 정적 JSON 객체에서 정의되기 때문입니다. JSON.stringify()가 replacer(k,v)를 호출하는 경우 원래 입력 객체를 순회하는 중임을 나타냅니다. replacer 함수에 전달된 암시적 this 매개 변수는 키와 값을 보유하는 객체를 참조합니다. JSON.stringify()는 원래 입력 객체를 수정하지 않으므로 이 객체는 순회 중인 컨테이너에서 변경되지 않고 그대로 유지됩니다. 따라서 this[k] 코드를 사용하여 원래 객체에서 키를 쿼리할 수 있습니다. v 매개 변수는 toJSON()을 통해 변환되는 값을 보유합니다.

toJSON()과 마찬가지로 replacer 함수는 임의 유형의 값을 반환할 수 있습니다. replacer가 문자열을 반환하는 경우 JSON 엔진에서는 내용을 따옴표로 이스케이프한 다음 이스케이프된 내용을 다시 따옴표로 둘러쌉니다. 이렇게 따옴표로 둘러싸면 stringify()가 이후의 JSON.parse() 호출에서 문자열이 유지되는 유효한 JSON 문자열 객체를 수신하게 됩니다.

다음 코드에서는 replacer 매개 변수 및 암시적 this 매개 변수를 사용하여 Date 객체의 time 및 hours 값을 반환합니다.

```
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;
});
```

JSON.parse() reviver 매개 변수 사용

JSON.parse() 메서드의 reviver 매개 변수는 replacer 함수의 반대 기능을 수행합니다. 즉, JSON 문자열을 사용 가능한 ActionScript 객체로 변환합니다. reviver 인수는 두 개의 매개 변수를 사용하고 모든 유형을 반환하는 함수입니다.

```
function (k,v):*
```

이 함수에서 k는 키이고 v는 k의 값입니다. stringify()와 마찬가지로, parse()는 JSON 키-값 쌍을 순회하고 각 쌍에 reviver 함수(있는 경우)를 적용합니다. 잠재적인 문제는 JSON 클래스가 객체의 ActionScript 클래스 이름을 출력하지 않는다는 점입니다. 따라서 다시 사용할 객체 유형을 파악하기가 어려울 수 있습니다. 이 문제는 객체가 중첩되는 경우에 특히 까다로워질 수 있습니다. toJSON(), replacer 및 reviver 함수를 디자인하는 과정에서 원래 객체를 그대로 유지하면서 노출되는 ActionScript 객체를 식별하는 방법을 고안할 수 있습니다.

파싱 예

다음 예제에서는 JSON 문자열에서 파싱되는 객체를 다시 사용하는 방법을 보여 줍니다. 이 예제에서는 JSONGenericDictExample과 JSONDictionaryExtnExample이라는 두 개의 클래스를 정의하는데, JSONGenericDictExample 클래스는 사용자 정의 사전 클래스입니다. 각 레코드에는 개인의 이름과 생일은 물론 고유한 ID까지 포함되어 있습니다. 생성자 JSONGenericDictExample이 호출될 때마다 해당 ID가 정적으로 증가하는 정수인 내부 정적 배열에 새로 만들어진 객체가 추가됩니다. 또한 JSONGenericDictExample 클래스는 보다 긴 id 멤버에서 정수 부분만 추출하는 revive() 메서드를 정의합니다. revive() 메서드에서는 이 정수를 사용하여 다시 사용 가능한 올바른 객체를 조회하고 반환합니다.

JSONDictionaryExtnExample 클래스는 ActionScript Dictionary 클래스를 확장합니다. 이 클래스의 레코드에는 설정된 구조가 없으며 어떤 데이터든 포함될 수 있습니다. 데이터는 클래스에서 정의된 속성이라기보다 JSONDictionaryExtnExample 객체가 생성된 후에 할당됩니다. JSONDictionaryExtnExample 레코드에서는 JSONGenericDictExample 객체를 키로 사용합니다. JSONDictionaryExtnExample 객체를 다시 사용하는 경우 JSONGenericDictExample.revive() 함수에서는 JSONDictionaryExtnExample과 연결된 ID를 사용하여 올바른 키 객체를 다시 사용합니다.

무엇보다도, JSONDictionaryExtnExample.toJSON() 메서드는 JSONDictionaryExtnExample 객체 외에 표시자 문자열을 반환한다는 점이 중요합니다. 이 문자열은 JSON 출력을 JSONDictionaryExtnExample 클래스에 속하는 것으로 식별하는 역할을 합니다. 이 표시자가 있으면 JSON.parse()가 실행되는 중에 어떤 객체 유형이 처리되고 있는지를 확실히 알 수 있습니다.

```
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;
        }
    }
}
```

다음 런타임 스크립트가 JSONDictionaryExtnExample 객체에 JSON.parse()를 호출하면 reviver 함수는 JSONDictionaryExtnExample의 각 객체에 JSONGenericDictExample.revive()를 호출합니다. 이러한 호출을 통해 객체 키를 나타내는 ID가 추출됩니다. JSONGenericDictExample.revive()함수는 이 ID를 사용하여 전용 정적 배열에서 저장된 JSONDictionaryExtnExample 객체를 검색 및 반환합니다.

```
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]>]
```

8장: 이벤트 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

이벤트 처리 시스템을 사용하면 프로그래머가 편리한 방식으로 사용자 입력 및 시스템 이벤트에 응답할 수 있습니다. ActionScript 3.0 이벤트 모델은 사용이 편리할 뿐만 아니라 표준 규격을 준수하며 표시 목록과 잘 통합됩니다. 새 이벤트 모델은 산업 표준 이벤트 처리 아키텍처인 DOM(Document Object Model) 레벨 3 이벤트 사양을 기초로 ActionScript 프로그래머를 위해 강력하고 직관적인 이벤트 처리 도구를 제공합니다.

ActionScript 3.0 이벤트 처리 시스템은 표시 목록과 밀접하게 상호 작용합니다. 표시 목록에 대한 기초적인 지식을 얻으려면 135페이지의 “디스플레이 프로그래밍”을 읽어 보십시오.

기타 도움말 항목

flash.events 패키지

DOM(Document Object Model) 레벨 3 이벤트 사양

이벤트 처리의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

이벤트는 SWF 파일에서 발생하는 동작이며 프로그래머에게 매우 중요합니다. 예를 들어 대부분의 SWF 파일은 사용자 상호 작용을 지원합니다. 이러한 상호 작용은 마우스 클릭에 응답하는 것처럼 간단할 수도 있고 양식에 입력된 데이터를 적용 및 처리하는 것처럼 복잡할 수도 있습니다. SWF 파일과의 이러한 상호 작용을 이벤트라고 합니다. 그러나 서버에서 데이터 로드를 마쳤을 때 또는 연결된 카메라가 활성화될 때와 같이 직접적인 사용자 상호 작용 없이도 이벤트가 발생할 수 있습니다.

ActionScript 3.0에서 각 이벤트는 Event 클래스 또는 해당 하위 클래스의 인스턴스인 이벤트 객체로 나타냅니다. 이벤트 객체는 특정 이벤트에 대한 정보를 저장할 뿐만 아니라 해당 이벤트 객체를 쉽게 조작하는 데 사용할 수 있는 메서드도 포함합니다. 예를 들어 Flash Player 또는 AIR에서는 마우스 클릭을 감지하면 특정 마우스 클릭 이벤트를 나타내는 이벤트 객체(MouseEvent 클래스의 인스턴스)가 만들어집니다.

이벤트 객체를 만든 후에는 Flash Player 또는 AIR에서 이 이벤트 객체를 이벤트 대상 객체에 **전달**합니다. 전달되는 이벤트 객체의 대상으로 사용되는 객체를 **이벤트 대상**이라고 합니다. 예를 들어 연결된 카메라가 활성화되면 Flash Player에서 이벤트 객체를 직접 이벤트 대상에 전달하는데, 이 경우 이벤트 대상은 해당 카메라를 나타내는 객체입니다. 그러나 이벤트 대상이 표시 목록에 있는 경우에는 표시 목록 계층 구조를 통해 해당 표시 목록에서 이벤트 대상을 찾을 때까지 이벤트 객체가 전달됩니다. 경우에 따라서는 이벤트 객체가 같은 경로를 따라 표시 목록 계층 구조에서 다시 위로 "버블링"될 수도 있습니다. 이러한 표시 목록 계층 구조 순회를 **이벤트 흐름**이라고 합니다.

코드에서 이벤트 리스너를 사용하여 이벤트 객체를 "수신"할 수 있습니다. **이벤트 리스너**는 특정 이벤트에 응답하기 위해 작성하는 함수 또는 메서드입니다. 프로그램에서 이벤트에 응답할 수 있도록 하려면 이벤트 대상 또는 이벤트 객체의 이벤트 흐름에 포함된 표시 목록 객체에 이벤트 리스너를 추가해야 합니다.

이벤트 리스너 코드를 작성할 때는 다음과 같은 기본 구조를 따릅니다. 여기서 굵게 표시된 요소는 특정한 경우에 입력하는 자리 표시자입니다.

```
function eventResponse(eventObject:EventType):void
{
    // Actions performed in response to the event go here.
}

eventTarget.addEventListener(EventType.EVENT_NAME, eventResponse);
```

이 코드는 두 가지 작업을 수행합니다. 먼저 함수를 정의하여 이벤트에 대한 응답으로 수행할 작업을 지정할 수 있습니다. 그런 다음 소스 객체의 addEventListener() 메서드를 호출하여 지정된 이벤트에 대해 함수를 "제공"함으로써 이벤트 발생 시 함수 작업이 수행되도록 합니다. 실제로 이벤트가 발생하면 이벤트 대상이 이벤트 리스너로 등록된 모든 함수와 메서드 목록을 확인합니다. 그런 다음 각 함수와 이벤트를 차례로 호출하여 이벤트 객체를 매개 변수로 전달합니다.

이벤트 리스너를 직접 만들려면 이 코드에서 네 가지를 변경해야 합니다. 먼저 함수 이름을 사용자가 사용할 이름으로 변경해야 합니다. 코드에서 **eventResponse**라고 표시된 두 군데를 변경합니다. 두 번째로, 수신할 이벤트에서 전달한 이벤트 객체에 적절한 클래스 이름(코드에서는 **EventType**)을 지정해야 하며 특정 이벤트에 적절한 상수(샘플의 **EVENT_NAME**)를 지정해야 합니다. 세 번째로, 이벤트를 전달할 객체에서 addEventListener() 메서드(이 코드에서는 **eventTarget**)를 호출해야 합니다. 필요에 따라 함수의 매개 변수로 사용된 변수의 이름(이 코드에서는 **eventObject**)을 변경할 수 있습니다.

중요한 개념 및 용어

이벤트 처리 루틴을 쓸 때 사용되는 중요한 용어들이 아래 참조 목록에 정리되어 있습니다.

버블링 특정 이벤트에 버블링이 발생할 경우 부모 표시 객체가 해당 자식이 전달하는 이벤트에 응답할 수 있습니다.

버블링 단계 이벤트 흐름 중 이벤트가 부모 표시 객체까지 전달되는 단계를 가리킵니다. 버블링 단계는 캡처 단계와 대상 단계 뒤에 일어납니다.

캡처 단계 이벤트 흐름 중 이벤트가 가장 일반적인 대상에서 가장 구체적인 대상 객체로 전달되는 단계를 가리킵니다. 캡처 단계는 대상 단계와 버블링 단계 앞에 일어납니다.

기본 비헤이비어 일부 이벤트에는 일반적으로 이벤트와 함께 발생하는 비헤이비어가 포함되어 있는데, 이를 기본 비헤이비어라고 합니다. 예를 들어 사용자가 텍스트 필드에 텍스트를 입력하면 텍스트 입력 이벤트가 발생합니다. 이 이벤트의 기본 비헤이비어는 텍스트 필드에 입력한 문자를 실제로 표시하는 것이지만, 어떠한 이유로든 입력한 문자를 표시하지 않으려는 경우 이 기본 비헤이비어를 재정의할 수 있습니다.

전달 이벤트가 발생했음을 이벤트 리스너에 알리는 것입니다.

이벤트 객체에 발생하는 것으로, 이에 대해 객체가 다른 객체에 알릴 수 있습니다.

이벤트 흐름 표시 목록의 객체(화면에 표시되는 객체)에 이벤트가 발생하면 해당 객체를 포함하는 모든 객체에 이벤트를 알리고 해당 객체가 이를 다시 이벤트 리스너에 알립니다. 이 프로세스는 스테이지에서 시작하여 표시 목록을 통해 이벤트가 발생한 실제 객체로 이동한 다음 다시 스테이지로 돌아갑니다. 이 프로세스를 이벤트 흐름이라고 합니다.

Event 객체 특정 이벤트의 발생에 대한 정보가 포함된 객체입니다. 이 객체는 이벤트를 전달할 때 모든 리스너에 보내집니다.

이벤트 대상 이벤트를 실제로 전달하는 객체입니다. 예를 들어 사용자가 스테이지 안에 있는 Sprite 안의 버튼을 클릭하면 해당하는 모든 객체가 이벤트를 전달하지만 이벤트 대상은 실제로 이벤트가 발생한 객체(이 경우, 클릭한 버튼)입니다.

리스너 특정 이벤트가 발생하는 경우 알려야 함을 나타내기 위해 객체에 등록된 객체 또는 함수입니다.

대상 단계 이벤트 흐름 중 이벤트가 가장 구체적인 가능 대상에 도달한 시점을 가리킵니다. 대상 단계는 캡처 단계와 버블링 단계 사이에 일어납니다.

ActionScript 3.0과 이전 버전의 이벤트 처리 방식 비교

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0과 이전 버전의 ActionScript에서 이벤트를 처리하는 방식 중 가장 큰 차이점은 ActionScript 3.0에는 이벤트 처리 시스템이 하나뿐인 반면 이전 버전의 ActionScript에는 이벤트 처리 시스템이 여러 개 있다는 것입니다. 이 단원에서는 먼저 이전 버전의 ActionScript에서 이벤트를 처리하는 방식에 대해 간략하게 설명한 다음, ActionScript 3.0에서 이벤트 처리 방식이 어떻게 변경되었는지 설명합니다.

이전 버전의 ActionScript에서 이벤트 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0 이전 버전의 ActionScript에서는 다음과 같은 다양한 방법을 사용하여 이벤트를 처리합니다.

- on() 이벤트 핸들러 - Button 및 MovieClip 인스턴스에 직접 배치할 수 있습니다.
- onClipEvent() 핸들러 - MovieClip 인스턴스에 직접 배치할 수 있습니다.
- 콜백 함수 속성 - XML.onload 및 Camera.onActivity 등이 있습니다.
- 이벤트 리스너 - addListener() 메서드를 사용하여 등록합니다.
- UIEventDispatcher 클래스 - DOM 이벤트 모델을 부분적으로 구현했습니다.

이러한 메커니즘에는 각각 고유한 장점과 단점이 있습니다. on() 및 onClipEvent() 핸들러는 쉽게 사용할 수 있지만 버튼과 동영상 클립에 직접 배치한 코드를 찾기가 어려우므로 이후 프로젝트를 관리하는 데 어려움이 있습니다. 콜백 함수도 간단하게 구현할 수는 있지만 지정된 이벤트에 대해 콜백 함수를 하나만 사용할 수 있습니다. 이와 달리 이벤트 리스너는 리스너 객체와 함수를 만들어야 할 뿐만 아니라 이벤트를 생성하는 객체에 리스너를 등록해야 하므로 쉽게 구현할 수 없습니다. 그러나 이를 통해 리스너 객체를 여러 개 만들어 같은 이벤트에 모든 리스너를 등록할 수 있습니다.

ActionScript 2.0의 구성 요소 개발을 통해 다른 이벤트 모델이 생성되었습니다. UIEventDispatcher 클래스에 구현된 이 새 모델은 DOM 이벤트 사양의 하위 집합을 기초로 하므로 구성 요소 이벤트 처리 작업에 익숙한 개발자는 비교적 큰 어려움 없이 새 ActionScript 3.0 이벤트 모델을 사용할 수 있습니다.

다양한 이벤트 모델에 사용되는 구문은 여러 가지 면에서 공통되는 부분이 있지만 나머지 부분에서는 서로 다릅니다. 예를 들어 ActionScript 2.0에서는 TextField.onChanged와 같은 일부 속성을 콜백 함수 또는 이벤트 리스너로 사용할 수 있습니다. 그러나 리스너 객체 등록 구문은 리스너를 지원하는 여섯 개의 클래스 중 하나를 사용하는지 아니면 UIEventDispatcher 클래스를 사용하는지에 따라 다릅니다. Key, Mouse, MovieClipLoader, Selection, Stage 및 TextField 클래스의 경우 addListener() 메서드를 사용하지만 구성 요소 이벤트 처리의 경우에는 addEventListener() 메서드를 사용합니다.

또한 이벤트 처리 모델이 여러 개인 경우에는 사용된 메커니즘에 따라 이벤트 핸들러 함수의 범위가 크게 달라진다는 문제도 있습니다. 즉, this 키워드의 의미가 이벤트 처리 시스템 간에 일치하지 않습니다.

ActionScript 3.0의 이벤트 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 이전 버전의 언어에 사용된 다양한 이벤트 처리 메커니즘을 대체하는 단일 이벤트 처리 모델을 사용합니다. 새 이벤트 모델은 DOM(Document Object Model) 레벨 3 이벤트 사양을 기초로 합니다. SWF 파일 형식은 Document Object Model 표준을 따르지는 않지만 표시 목록과 DOM 구조에는 DOM 이벤트 모델을 구현할 수 있도록 비슷한 점이 많습니다. 표시 목록의 객체는 DOM 계층 구조의 노드에 해당하므로 여기에서 **표시 목록 객체**와 **노드**라는 용어를 바꾸어 사용할 수 있습니다.

Flash Player 및 AIR에서 DOM 이벤트 모델을 구현하는 경우 기본 비헤이비어라는 개념이 포함됩니다. **기본 비헤이비어**는 특정 이벤트의 결과로 Flash Player 또는 AIR에서 실행되는 액션입니다.

기본 비헤이비어

일반적으로 개발자는 이벤트에 응답하는 코드를 작성합니다. 그러나 경우에 따라서는 해당 이벤트와 일반적으로 연결된 비헤이비어가 있어 개발자가 비헤이비어 취소 코드를 추가하지 않는 한 Flash Player 또는 AIR에서 자동으로 비헤이비어를 실행합니다. Flash Player 또는 AIR에서 자동으로 실행하는 이러한 비헤이비어를 기본 비헤이비어라고 합니다.

예를 들어 사용자가 TextField 객체에 텍스트를 입력하는 경우 일반적으로 이 텍스트가 TextField 객체에 표시될 것으로 예상하므로 이러한 비헤이비어가 Flash Player 및 AIR에 내장됩니다. 이 기본 비헤이비어가 발생하지 않도록 하려면 새 이벤트 처리 시스템을 사용하여 취소하면 됩니다. 보다 구체적으로 설명하면, 사용자가 TextField 객체에 텍스트를 입력하는 경우 Flash

Player 또는 AIR에서 TextEvent 클래스의 인스턴스를 만들어 사용자 입력을 표시합니다. Flash Player 또는 AIR에서 TextField 객체에 텍스트를 표시하지 않도록 하려면 특정 TextEvent 인스턴스에 액세스하여 해당 인스턴스의 preventDefault() 메서드를 호출해야 합니다.

기본 비헤이비어 중 일부는 취소할 수 없습니다. 예를 들어 사용자가 TextField 객체에서 단어를 두 번 클릭하면 Flash Player 및 AIR에서 MouseEvent 객체를 생성합니다. 이때 커서 아래의 단어가 강조 표시되는 기본 비헤이비어는 취소할 수 없습니다.

대부분의 이벤트 객체 유형에는 기본 비헤이비어가 연결되어 있지 않습니다. 예를 들어 네트워크 연결이 설정되면 Flash Player에서 connect 이벤트 객체를 전달하지만 이 객체에는 기본 비헤이비어가 연결되어 있지 않습니다. Event 클래스 및 하위 클래스에 대한 API 설명서에서는 각 이벤트 유형 목록을 제공하고 해당 이벤트에 연결된 기본 비헤이비어에 대한 내용 및 각 비헤이비어를 취소할 수 있는지 여부를 설명합니다.

기본 비헤이비어는 Flash Player 또는 AIR에서 전달하는 이벤트 객체에만 연결되고 ActionScript를 통해 프로그래밍 방식으로 전달되는 이벤트 객체에는 연결되지 않는다는 사실을 이해해야 합니다. 예를 들어 EventDispatcher 클래스의 메서드를 사용하면 textInput 유형의 이벤트 객체를 전달할 수 있지만 이 이벤트 객체에는 기본 비헤이비어가 연결되지 않습니다. 즉, Flash Player 및 AIR에서는 개발자가 프로그래밍 방식으로 전달한 textInput 이벤트의 결과로 TextField 객체에 문자를 표시하지 않습니다.

ActionScript 3.0 이벤트 리스너의 새로운 기능

ActionScript 2.0의 addListener() 메서드를 사용해 본 개발자는 ActionScript 2.0 이벤트 리스너 모델과 ActionScript 3.0 이벤트 모델의 차이점을 쉽게 파악할 수 있습니다. 다음 목록에서는 두 이벤트 모델의 몇 가지 주요 차이점을 설명합니다.

- ActionScript 2.0에서는 이벤트 리스너를 추가할 때 상황에 따라 addListener() 또는 addEventListener()를 사용하지만 ActionScript 3.0에서는 항상 addEventListener()를 사용합니다.
- ActionScript 2.0에는 이벤트 흐름이 없으므로 이벤트를 브로드캐스팅하는 객체에서만 addListener() 메서드를 호출할 수 있지만 ActionScript 3.0에서는 이벤트 흐름에 포함된 모든 객체에서 addEventListener() 메서드를 호출할 수 있습니다.
- ActionScript 2.0에서는 함수, 메서드 또는 객체를 이벤트 리스너로 사용할 수 있지만 ActionScript 3.0에서는 함수 또는 메서드만 이벤트 리스너로 사용할 수 있습니다.

이벤트 흐름

Flash Player 9 이상, Adobe AIR 1.0 이상

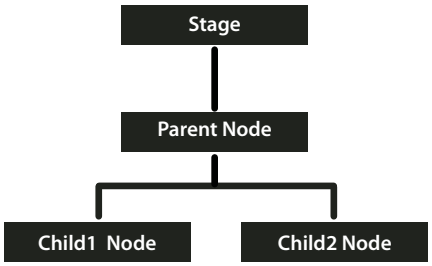
Flash Player 또는 AIR에서는 이벤트가 발생할 때마다 이벤트 객체를 전달합니다. 이때 이벤트 대상이 표시 목록에 없으면 이벤트 객체를 이벤트 대상에 직접 전달합니다. 예를 들어 Flash Player에서는 progress 이벤트 객체를 URLStream 객체에 직접 전달합니다. 그러나 이벤트 대상이 표시 목록에 있는 경우에는 이벤트 객체를 표시 목록에 전달하고 이 이벤트 객체는 해당 이벤트 대상을 찾을 때까지 표시 목록을 순회합니다.

이벤트 흐름은 이벤트 객체가 표시 목록을 이동하는 방식을 기술합니다. 표시 목록은 트리로 나타낼 수 있는 계층 구조로 구성되어 있으며, 표시 목록 계층 구조의 맨 위에는 Stage, 즉 표시 목록의 루트로 사용되는 특수한 표시 객체 컨테이너가 있습니다. Stage는 flash.display.Stage 클래스로 나타내고 표시 객체를 통해서만 액세스할 수 있습니다. 모든 표시 객체에는 해당 응용 프로그램의 Stage를 참조하는 stage 속성이 있습니다.

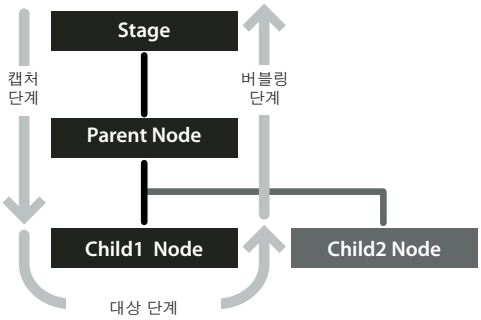
Flash Player 또는 AIR에서 표시 목록 관련 이벤트에 대한 이벤트 객체를 전달하는 경우 해당 이벤트 객체는 Stage에서 대상 노드까지 포함하는 라운드 트립을 만듭니다. DOM 이벤트 사양에서는 대상 노드를 이벤트 대상을 나타내는 노드로 정의합니다. 즉, 대상 노드는 이벤트가 발생한 표시 목록 객체입니다. 예를 들어 사용자가 child1이라는 표시 목록 객체를 클릭하면 Flash Player 또는 AIR에서 child1을 대상 노드로 사용하여 이벤트 객체를 전달합니다.

이벤트 흐름은 개념적으로 세 부분으로 구분됩니다. 첫 번째 부분은 캡처 단계라고 하고 Stage에서 대상 노드의 부모까지 모든 노드를 포함합니다. 두 번째 부분은 대상 단계라고 하고 대상 노드만 포함합니다. 마지막으로 세 번째 부분은 버블링 단계라고 하고 대상 노드의 부모에서 다시 Stage로 돌아오는 과정에서 만나게 되는 노드를 포함합니다.

다음 다이어그램과 같이 표시 목록을 Stage가 맨 위에 있는 세로 계층 구조로 생각해 보면 각 단계의 이름을 쉽게 이해할 수 있습니다.



사용자가 Child1 Node를 클릭하면 Flash Player 또는 AIR에서는 이벤트 객체를 이벤트 흐름에 전달합니다. 다음 그림과 같이 객체는 Stage에서 시작하여 아래에 있는 Parent Node와 Child1 Node로 차례로 이동한 다음, 다시 Stage까지 위로 "버블링"됩니다. 버블링 단계에서 Stage로 이동할 때는 다시 Parent Node를 통과합니다.



이 예제에서 캡처 단계는 처음에 아래로 이동하는 동안 Stage와 Parent Node를 포함하고 대상 단계는 Child1 Node에서 소비한 시간을 포함합니다. 마지막으로 버블링 단계는 다시 루트 노드로 이동하는 동안 만나는 Parent Node와 Stage를 포함합니다.

이벤트 흐름 덕분에 ActionScript 프로그래머는 이전에 비해 더욱 강력한 이벤트 처리 시스템을 사용할 수 있습니다. 이전 버전의 ActionScript에는 이벤트 흐름이 없으므로 이벤트를 생성하는 객체에만 이벤트 리스너를 추가할 수 있습니다. 그러나 ActionScript 3.0에서는 대상 노드뿐만 아니라 이벤트 흐름을 따라 모든 노드에 이벤트 리스너를 추가할 수 있습니다.

이벤트 흐름을 따라 이벤트 리스너를 추가할 수 있는 기능은 사용자 인터페이스 구성 요소에 객체가 두 개 이상 포함되어 있는 경우에 유용합니다. 예를 들어 버튼 객체에 버튼 레이블로 사용되는 텍스트 객체가 포함될 수 있습니다. 이벤트 흐름에 리스너를 추가하는 기능이 없는 경우 버튼에서 발생하는 모든 클릭 이벤트에 대한 알림을 받으려면 버튼 객체와 텍스트 객체 둘 다에 리스너를 추가해야 합니다. 그러나 이벤트 흐름 기능을 사용하면 텍스트 객체 또는 텍스트 객체에 의해 식별되는 버튼 객체 영역에서 발생하는 클릭 이벤트를 처리하는 단일 이벤트 리스너를 버튼 객체에 추가할 수 있습니다.

그러나 이벤트 흐름의 세 단계 중 일부 단계에 참여하지 못하는 이벤트 객체도 있습니다. enterFrame 및 init 이벤트 유형 등 일부 유형의 이벤트는 대상 노드에 직접 전달되고 캡처 단계 및 버블링 단계에 참여하지 않습니다. Socket 클래스의 인스턴스에 전달되는 이벤트와 같이 표시 목록에 없는 객체를 대상으로 하는 이벤트도 있습니다. 이러한 이벤트 객체도 캡처 및 버블링 단계에 참여하지 않고 대상 객체에 직접 전달됩니다.

특정 이벤트 유형의 동작 방식을 확인하려면 API 설명서를 확인하거나 이벤트 객체의 속성을 검토하십시오. 이벤트 객체의 속성을 검토하는 데 대한 자세한 내용은 다음 단원에서 설명합니다.

이벤트 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

이벤트 객체는 새 이벤트 처리 시스템에서 크게 두 가지 목적으로 사용됩니다. 첫 번째로 이벤트 객체는 속성 집합에 특정 이벤트에 대한 정보를 저장하여 실제 이벤트를 나타내고, 두 번째로 이벤트 객체에는 이벤트 객체를 조작하고 이벤트 처리 시스템의 비헤이비어에 영향을 줄 수 있는 메서드 집합이 포함되어 있습니다.

이러한 속성과 메서드에 쉽게 액세스할 수 있도록 Flash Player API에서는 모든 이벤트 객체의 기본 클래스로 사용되는 Event 클래스를 정의합니다. Event 클래스는 모든 이벤트 객체에 공통으로 사용되는 기본 속성 및 메서드 집합을 정의합니다.

이 단원에서는 먼저 Event 클래스 속성과 Event 클래스 메서드에 대해 차례로 설명하고 Event 클래스의 하위 클래스를 사용하는 이유에 대해 설명합니다.

Event 클래스 속성 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

Event 클래스는 이벤트 객체에 대해 다음과 같은 중요한 정보를 제공하는 많은 읽기 전용 속성 및 상수를 정의합니다.

- 이벤트 객체 유형은 상수로 나타내고 Event.type 속성에 저장됩니다.
- 이벤트의 기본 비헤이비어를 취소할 수 있는지 여부는 부울 값으로 나타내고 Event.cancelable 속성에 저장됩니다.
- 이벤트 흐름 정보는 나머지 속성에 포함됩니다.

이벤트 객체 유형

모든 이벤트 객체에는 이벤트 유형이 연결되어 있으며 이벤트 유형은 Event.type 속성에 문자열 값으로 저장됩니다. 코드에서 유형이 다른 객체를 구별할 수 있도록 이벤트 객체의 유형을 알고 있으면 유용합니다. 예를 들어 다음 코드에서는 clickHandler() 리스너 함수가 myDisplayObject에 전달되는 마우스 클릭 이벤트 객체에 응답하도록 지정합니다.

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

Event 클래스 자체에 연결된 이벤트 유형은 24개 정도이며 각각 Event 클래스 상수로 나타냅니다. Event 클래스 정의에서 발췌한 다음 코드에서는 이러한 이벤트 유형 중 일부를 보여 줍니다.

```
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
    }
}
```

이러한 상수를 사용하면 특정 이벤트 유형을 쉽게 참조할 수 있습니다. 실제로 이벤트 유형을 나타내는 문자열 대신 상수를 사용해야 합니다. 코드에 상수 이름 철자를 잘못 입력하면 컴파일러에서 이러한 오류를 catch하지만 문자열을 사용하는 경우에는 컴파일할 때 철자 오류가 나타나지 않으므로 쉽게 디버깅할 수 없는 비헤이비어가 발생할 수 있습니다. 예를 들어 이벤트 리스너를 추가할 때 다음 코드를 사용하십시오.

```
myDisplayObject.addEventListener(MouseEvent.CLICK, clickHandler);
```

이때 다음 코드를 사용하지 않는 것이 좋습니다.

```
myDisplayObject.addEventListener("click", clickHandler);
```

기본 비헤이비어 정보

cancelable 속성에 액세스하여 특정 이벤트 객체의 기본 비헤이비어를 취소할 수 있는지 여부를 검사하도록 코드를 작성할 수 있습니다. cancelable 속성에는 기본 비헤이비어를 취소할 수 있는지 여부를 나타내는 부울 값이 포함되어 있습니다. preventDefault() 메서드를 사용하면 소수의 이벤트와 연결된 기본 비헤이비어를 취소할 수 있습니다. 자세한 내용은 120페이지의 “Event 클래스 메서드 이해”에서 기본 이벤트 비헤이비어 취소를 참조하십시오.

이벤트 흐름 정보

나머지 Event 클래스 속성에는 이벤트 객체 및 해당 이벤트 객체와 이벤트 흐름 간의 관계에 대한 중요 정보가 포함되어 있습니다(다음 목록의 설명 참조).

- bubbles 속성에는 해당 이벤트 객체가 참여하는 이벤트 흐름의 단계에 대한 정보가 포함됩니다.
- eventPhase 속성은 이벤트 흐름에서 현재 단계를 나타냅니다.
- target 속성에는 이벤트 대상에 대한 참조가 저장됩니다.
- currentTarget 속성에는 현재 이벤트 객체를 처리하고 있는 표시 목록 객체에 대한 참조가 저장됩니다.

bubbles 속성

이벤트 객체가 이벤트 흐름의 버블링 단계에 참여하는 경우, 즉 이벤트 객체가 대상 노드에서 다시 전달되어 조상 노드를 거쳐 Stage에 도달하는 경우 해당 이벤트가 버블링된다고 합니다. Event.bubbles 속성에는 이벤트 객체가 버블링 단계에 참여하는지 여부를 나타내는 부울 값이 저장됩니다. 버블링되는 모든 이벤트는 캡처 및 대상 단계에도 참여하므로 버블링되는 이벤트는 이벤트 흐름의 세 단계에 모두 참여하게 됩니다. 이 값이 true이면 이벤트 객체가 세 단계에 모두 참여하고, 이 값이 false이면 이벤트 객체가 버블링 단계에 참여하지 않습니다.

eventPhase 속성

eventPhase 속성을 검토하면 이벤트 객체의 이벤트 단계를 확인할 수 있습니다. eventPhase 속성에는 이벤트 흐름의 세 단계 중 하나를 나타내는 부호 없는 정수 값이 포함됩니다. Flash Player API에서는 다음 코드와 같이 부호 없는 세 정수 값에 해당하는 세 개의 상수가 포함된 별도의 EventPhase 클래스를 정의합니다.

```
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;
    }
}
```

이러한 상수는 eventPhase 속성의 유효한 세 값에 해당합니다. 상수를 사용하면 쉽게 읽을 수 있는 코드를 작성할 수 있습니다. 예를 들어 이벤트 대상이 대상 단계에 있는 경우에만 myFunc() 함수를 호출하려면 다음 코드를 사용하여 이 조건을 테스트할 수 있습니다.

```
if (event.eventPhase == EventPhase.AT_TARGET)
{
    myFunc();
}
```

target 속성

target 속성에는 이벤트 대상 객체에 대한 참조가 있습니다. 이는 경우에 따라 간단할 수 있는데, 예를 들어 마이크가 활성화되는 경우 이벤트 객체의 대상은 Microphone 객체입니다. 그러나 대상이 표시 목록에 있는 경우에는 표시 목록 계층 구조를 고려해야 합니다. 예를 들어 표시 목록 객체가 겹쳐 있는 지점에서 사용자가 마우스 클릭을 입력하면 Flash Player 및 AIR에서는 항상 Stage와 가장 멀리 떨어진 객체를 이벤트 대상으로 선택합니다.

복잡한 SWF 파일, 특히 크기가 작은 여러 개의 자식 객체로 장식된 버튼이 있는 SWF 파일의 경우 target 속성은 버튼이 아니라 버튼의 자식 객체를 가리키는 경우가 많으므로 자주 사용되지 않습니다. 이러한 경우 target 속성은 버튼의 자식 객체를 가리키지만 이 속성은 버튼을 가리키므로 일반적으로 버튼에 이벤트 리스너를 추가하고 currentTarget 속성을 사용하는 것이 좋습니다.

currentTarget 속성

currentTarget 속성에는 현재 이벤트 객체를 처리하고 있는 객체에 대한 참조가 포함됩니다. 검토 중인 이벤트 객체를 어떤 노드에서 처리하고 있는지 모른다는 것이 잘 이해가 되지 않을 수도 있지만 이벤트 객체의 이벤트 흐름에 있는 모든 표시 객체에 리스너 함수를 추가할 수 있으며 아무 위치에나 리스너 함수를 배치할 수 있음을 기억하십시오. 또한 같은 리스너 함수를 여러 표시 객체에 추가할 수도 있습니다. 프로젝트가 더 복잡해지고 크기가 커질수록 currentTarget 속성의 유용성도 커집니다.

Event 클래스 메서드 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

Event 클래스 메서드는 다음과 같은 세 가지 범주로 구성되어 있습니다.

- 유틸리티 메서드 - 이벤트 객체의 복사본을 만들거나 문자열로 변환할 수 있습니다
- 이벤트 흐름 메서드 - 이벤트 흐름에서 이벤트 객체를 제거합니다.
- 기본 비헤이비어 메서드 - 기본 비헤이비어를 취소하거나 취소 여부를 확인합니다

Event 클래스 유틸리티 메서드

Event 클래스에는 두 가지 유틸리티 메서드가 있습니다. clone() 메서드를 사용하면 이벤트 객체의 복사본을 만들 수 있으며, toString() 메서드를 사용하면 이벤트 객체 속성의 문자열 표현과 해당 속성 값을 생성할 수 있습니다. 이러한 두 메서드는 모두 이벤트 모델 시스템에서 내부적으로 사용되지만 일반적인 용도로 개발자에게 노출됩니다.

Event 클래스의 하위 클래스를 만드는 고급 개발자의 경우 이벤트 하위 클래스가 제대로 작동하도록 두 유틸리티 메서드 버전을 모두 재정의하여 새로 구현해야 합니다.

이벤트 흐름 중단

Event.stopPropagation() 메서드 또는 Event.stopImmediatePropagation() 메서드를 호출하면 이벤트 흐름에서 이벤트 객체가 계속 진행되지 않도록 중단할 수 있습니다. 두 메서드는 다음과 같이 현재 노드의 다른 이벤트 리스너를 실행할 수 있는지 여부만 다르고 나머지는 거의 같습니다.

- Event.stopPropagation() 메서드를 사용하면 현재 노드에서 다른 이벤트 리스너가 실행된 경우에만 이벤트 객체가 다음 노드로 이동되지 않습니다.
- Event.stopImmediatePropagation() 메서드를 사용하면 이벤트 객체가 다음 노드로 이동되지 않지만 현재 노드에서 다른 이벤트 리스너를 실행할 수 없습니다.

두 메서드 중 하나를 호출해도 이벤트와 연결된 기본 비헤이비어의 발생 여부에는 아무 영향을 주지 않습니다. 기본 비헤이비어를 취소하려면 Event 클래스의 기본 비헤이비어 메서드를 사용하십시오.

기본 이벤트 비헤이비어 취소

기본 비헤이비어 취소와 관련된 두 메서드는 preventDefault() 및 isDefaultPrevented() 메서드입니다. 이벤트와 연결된 기본 비헤이비어를 취소하려면 preventDefault() 메서드를 호출합니다. 이벤트 객체에서 preventDefault() 메서드가 이미 호출되었는지 여부를 확인하려면 isDefaultPrevented() 메서드를 호출합니다. 이때 메서드가 이미 호출되었으면 true를 반환하고 그렇지 않으면 false를 반환합니다.

preventDefault() 메서드는 이벤트의 기본 비헤이비어를 취소할 수 있는 경우에만 작동합니다. 이를 확인하려면 API 설명서에서 해당 이벤트 유형에 대한 내용을 참조하거나, ActionScript를 사용하여 해당 이벤트 객체의 cancelable 속성을 검토하면 됩니다.

기본 비헤이비어를 취소해도 이벤트 흐름에서 이벤트 객체의 진행 상태에는 아무 영향을 주지 않습니다. 이벤트 흐름에서 이벤트 객체를 제거하려면 Event 클래스의 이벤트 흐름 메서드를 사용하십시오.

Event 클래스의 하위 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

대부분의 이벤트는 Event 클래스에 정의된 공통 속성 집합만 있어도 충분합니다. 그러나 Event 클래스에 있는 속성으로 캡처할 수 없는 고유한 특성을 갖고 있는 이벤트도 있습니다. 이러한 이벤트를 위해 ActionScript 3.0에서는 Event 클래스의 하위 클래스를 정의합니다.

각 하위 클래스는 해당 이벤트 범주에 고유한 속성 및 이벤트 유형을 추가로 제공합니다. 예를 들어 마우스 입력과 관련된 이벤트에는 Event 클래스에 정의된 속성으로 캡처할 수 없는 몇 가지 고유한 특성이 있습니다. MouseEvent 클래스는 마우스 이벤트의 위치, 마우스 이벤트를 실행하는 동안 특정 키를 눌렀는지 여부 등의 정보가 포함된 10개의 속성을 추가하여 Event 클래스를 확장합니다.

또한 Event 하위 클래스에는 하위 클래스와 관련된 이벤트 유형을 나타내는 상수도 포함되어 있습니다. 예를 들어 MouseEvent 클래스는 click, doubleClick, mouseDown 및 mouseUp 등 몇 가지 마우스 이벤트 유형에 대한 상수를 정의합니다.

118페이지의 “이벤트 객체”의 "Event 클래스 유틸리티 메서드" 단원에서 설명한 것처럼 Event 하위 클래스를 만들 때는 clone() 및 toString() 메서드를 재정의하여 해당 하위 클래스에 고유한 기능을 제공해야 합니다.

이벤트 리스너

Flash Player 9 이상, Adobe AIR 1.0 이상

이벤트 리스너는 Flash Player 및 AIR에서 특정 이벤트에 대한 응답으로 실행되는 함수이며 이벤트 핸들러라고도 합니다. 이벤트 리스너를 추가할 때는 두 단계 과정을 거쳐야 하는데, 먼저 Flash Player 또는 AIR에서 이벤트에 대한 응답으로 실행할 함수 또는 클래스 메서드를 만듭니다. 이를 리스너 함수 또는 이벤트 핸들러 함수라고도 합니다. 그런 다음 addEventListener() 메서드를 사용하여 리스너 함수를 이벤트 대상 또는 적절한 이벤트 흐름을 따라 놓여 있는 표시 목록 객체에 등록합니다.

리스너 함수 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

리스너 함수 만들기는 ActionScript 3.0 이벤트 모델이 DOM 이벤트 모델과 다른 부분 중 하나입니다. DOM 이벤트 모델에서는 이벤트 리스너와 리스너 함수 간의 차이가 명확합니다. 이벤트 리스너는 EventListener 인터페이스를 구현하는 클래스의 인스턴스이고 리스너 함수는 handleEvent()라는 클래스의 메서드입니다. 또한 DOM 이벤트 모델에서는 실제 리스너 함수가 아니라 리스너 함수를 포함하는 클래스 인스턴스를 등록합니다.

반면에 ActionScript 3.0 이벤트 모델에서는 이벤트 리스너와 리스너 함수 간에 차이가 없습니다. ActionScript 3.0에는 EventListener 인터페이스가 없으며 리스너 함수를 클래스 외부에 또는 클래스의 일부로 정의할 수 있습니다. 또한 리스너 함수의 이름을 handleEvent()로 지정할 필요가 없으며 유효한 식별자를 사용하여 명명할 수 있습니다. ActionScript 3.0에서는 실제 리스너 함수의 이름을 등록한다는 것도 DOM 이벤트 모델과 다릅니다.

클래스 외부에 정의되는 리스너 함수

다음 코드에서는 빨간색 정사각형 모양을 표시하는 간단한 SWF 파일을 만듭니다. 여기에서 클래스 외부에 정의된 clickHandler() 리스너 함수는 빨간색 정사각형에서 마우스 클릭 이벤트를 수신합니다.

```
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);
}
```

사용자가 사각형을 클릭하여 그 결과 나타나는 SWF 파일과 상호 작용하면 Flash Player 또는 AIR에서 다음과 같은 추적 출력이 생성됩니다.

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

이벤트 객체는 clickHandler()에 인수로 전달됩니다. 이를 통해 리스너 함수에서 이벤트 객체를 검토할 수 있습니다. 이 예제에서는 이벤트 객체의 type 속성을 사용하여 이벤트가 click 이벤트인지 확인합니다.

또한 이 예제에서는 this 키워드의 값도 확인합니다. 이 경우에는 함수가 사용자 정의 클래스 또는 객체의 외부에 정의되므로 this가 전역 객체를 나타냅니다.

클래스 메서드로 정의되는 리스너 함수

다음 예제는 clickHandler() 함수가 ChildSprite 클래스의 메서드로 정의된다는 것만 제외하면 ClickExample 클래스를 정의하는 위의 예제와 같습니다.

```
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);
    }
}
```

사용자가 빨간색 사각형을 클릭하여 그 결과로 나타나는 SWF 파일에 대해 작업을 처리하면 Flash Player 또는 AIR에서 다음과 같은 추적 출력이 생성됩니다.

```
clickHandler detected an event of type: click
the this keyword refers to: [object ChildSprite]
```

여기에서 this 키워드는 child라는 ChildSprite 인스턴스를 참조하는데, 이는 ActionScript 2.0의 비헤이비어와 비교하여 달라진 점입니다. ActionScript 2.0 구성 요소를 사용한 경우, 클래스 메서드를 UIEventDispatcher.addEventListener()에 전달하면 메서드 범위가 리스너 메서드가 정의된 클래스 대신 이벤트를 브로드캐스팅하는 구성 요소에 바인딩되었음을 유념해야 합니다. 즉, ActionScript 2.0에서 이 방법을 사용한 경우에는 this 키워드가 ChildSprite 인스턴스 대신 이벤트를 브로드캐스팅하는 구성 요소를 참조합니다.

이렇게 되면 리스너 메서드를 포함하는 클래스의 다른 메서드 및 속성에 액세스할 수 없으므로 일부 프로그래머에게는 이것이 중요한 문제였고 이를 해결하기 위해 ActionScript 2.0 프로그래머는 mx.util.Delegate 클래스를 사용하여 리스너 메서드의 범위를 변경할 수 있었습니다. 그러나 ActionScript 3.0에서는 addEventListener()가 호출될 때 바인딩 메서드를 만들기 때문에 이 방법이 더 이상 필요하지 않습니다. 결과적으로 this 키워드는 child라는 ChildSprite 인스턴스를 참조하고 프로그래머는 ChildSprite 클래스의 다른 메서드와 속성에 액세스할 수 있습니다.

사용할 수 없는 이벤트 리스너

동적으로 지정된 리스너 함수를 가리키는 속성으로 일반 객체를 만드는 제 3의 방법이 있지만 사용하지 않는 것이 좋습니다. 이 방법은 ActionScript 2.0에서 일반적으로 사용되었기 때문에 여기에서 설명하지만 ActionScript 3.0에서는 사용하지 않아야 합니다. 이 방법을 사용하면 this 키워드가 리스너 객체 대신 전역 객체를 참조하므로, 권장할만한 방법은 아닙니다.

다음 예제는 리스너 함수가 myListenerObj라는 일반 객체의 일부로 정의된다는 것만 제외하면 위에 나오는 ClickExample 클래스 예제와 같습니다.

```
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);
}
```

추적 결과는 다음과 같습니다.

```
clickHandler detected an event of type: click
the this keyword refers to: [object global]
```

this 키워드가 myListenerObj를 참조하고 추적 출력이 [object Object]로 표시된다고 예상할 수 있지만 실제로는 전역 객체를 참조합니다. 동적 속성 이름을 addEventListener()에 인수로 전달하면 Flash Player 또는 AIR에서 바인딩 메서드를 만들 수 없습니다. 이는 listener 매개 변수로 전달하는 대상이 리스너 함수의 메모리 주소이고 Flash Player 및 AIR에서는 해당 메모리 주소를 myListenerObj 인스턴스에 링크할 수 없기 때문입니다.

이벤트 리스너 관리

Flash Player 9 이상, Adobe AIR 1.0 이상

IEventDispatcher 인터페이스의 메서드를 사용하여 리스너 함수를 관리할 수 있습니다. IEventDispatcher 인터페이스는 DOM 이벤트 모델에서 사용되는 EventTarget 인터페이스의 ActionScript 3.0 버전입니다. IEventDispatcher라는 이름을 보면 이 인터페이스가 주로 이벤트 객체를 보내거나 전달하는 데 사용되는 것 같지만 실제로 이 클래스의 메서드는 이벤트 리스너를 등록, 확인 또는 제거하는 데 자주 사용됩니다. IEventDispatcher 인터페이스는 다음 코드와 같이 다섯 개의 메서드를 정의합니다.

```
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;
    }
}
```

Flash Player API는 이벤트 대상 또는 이벤트 흐름 구성 요소일 수 있는 모든 클래스의 기본 클래스로 사용되는 EventDispatcher 클래스를 사용하여 IEventDispatcher 인터페이스를 구현합니다. 예를 들어 DisplayObject 클래스는 EventDispatcher 클래스에서 상속됩니다. 즉, 표시 목록의 모든 객체는 IEventDispatcher 인터페이스의 메서드에 액세스할 수 있습니다.

이벤트 리스너 추가

addEventListener()는 IEventDispatcher 인터페이스에서 많은 역할을 담당하는 중요한 메서드입니다. 이 메서드는 리스너 함수를 등록할 때 사용됩니다. 이때 type과 listener라는 두 개의 매개 변수가 필요합니다. type 매개 변수는 이벤트 유형을 지정하는 데 사용되고, listener 매개 변수는 이벤트가 발생할 때 실행할 리스너 함수를 지정하는 데 사용됩니다. listener 매개 변수는 함수 또는 클래스 메서드에 대한 참조일 수 있습니다.

listener 매개 변수를 지정할 때 괄호를 사용하지 마십시오. 예를 들어 다음과 같이 addEventListener() 메서드를 호출하는 경우 clickHandler() 함수를 지정할 때 괄호를 추가하지 않습니다.

```
addEventListener(MouseEvent.CLICK, clickHandler)
```

addEventListener() 메서드의 useCapture 매개 변수를 사용하면 리스너를 활성화할 이벤트 흐름 단계를 제어할 수 있습니다. 예를 들어 useCapture가 true로 설정되면 이벤트 흐름의 캡처 단계를 진행하는 동안 리스너가 활성화되고, useCapture가 false로 설정되면 이벤트 흐름의 대상 단계 및 버블링 단계를 진행하는 동안 리스너가 활성화됩니다. 이벤트 흐름의 모든 단계를 진행하는 동안 이벤트를 수신하려면 addEventListener()를 두 번 호출해야 하는데, 첫 번째는 useCapture를 true로 설정하고 두 번째는 useCapture를 false로 설정해야 합니다.

addEventListener() 메서드의 priority 매개 변수는 DOM 레벨 3 이벤트 모델의 공식 구성 요소가 아닙니다. 이 매개 변수는 이벤트 리스너를 좀 더 융통성 있게 구성할 수 있도록 하기 위해 ActionScript 3.0에 포함되었습니다. addEventListener()를 호출하는 경우 정수 값을 priority 매개 변수로 전달하여 해당 이벤트 리스너의 우선 순위를 설정할 수 있습니다. 기본값은 0이지만 음의 정수 또는 양의 정수 값으로 설정할 수 있습니다. 이 값이 클수록 이벤트 리스너의 실행 순서가 빠릅니다. 우선 순위가 같은 이벤트 리스너는 추가된 순서대로 실행되므로 먼저 추가된 리스너의 실행 순서가 빠릅니다.

useWeakReference 매개 변수를 사용하면 리스너 함수에 대한 참조를 약한 참조 또는 일반 참조로 지정할 수 있습니다. 이 매개 변수를 true로 설정하면 더 이상 필요하지 않은 리스너 함수가 메모리에 남아있는 것을 방지할 수 있습니다. Flash Player 및 AIR에서는 가비지 컬렉션이라는 기법을 활용하여 더 이상 사용되지 않는 객체를 메모리에서 제거합니다. 객체에 대한 참조가 없는 경우 해당 객체는 더 이상 사용되지 않는 것으로 간주됩니다. 가비지 수집기는 참조가 약한 참조인지 여부에 관계없이 작업을 실행합니다. 즉, 약한 참조만 포함하는 리스너 함수를 가비지 컬렉션 대상으로 포함할 수 있습니다.

이벤트 리스너 제거

removeEventListener() 메서드를 사용하면 더 이상 필요하지 않은 이벤트 리스너를 제거할 수 있습니다. 더 이상 사용하지 않을 모든 리스너를 제거하는 것이 좋습니다. 이 메서드를 사용하는 경우에는 addEventListener() 메서드를 사용할 때와 마찬가지로 eventName 및 listener 매개 변수가 반드시 필요합니다. 앞에서 설명했듯이 모든 이벤트 단계를 진행하는 동안 이벤트를 수신하려면 addEventListener()를 두 번 호출해야 하는데, 첫 번째는 useCapture를 true로 설정하고 두 번째는 false로 설정해야 합니다. 마찬가지로, 두 이벤트 리스너를 모두 제거하려면 removeEventListener()를 두 번 호출해야 하는데, 첫 번째는 useCapture를 true로 설정하고 두 번째는 false로 설정해야 합니다.

이벤트 전달

dispatchEvent() 메서드는 고급 프로그래머가 사용자 정의 이벤트 객체를 이벤트 흐름에 전달할 때 사용할 수 있습니다. 이 메서드에 사용할 수 있는 매개 변수는 이벤트 객체에 대한 참조뿐이며, 이는 Event 클래스의 인스턴스 또는 Event 클래스의 하위 클래스여야 합니다. 이벤트가 전달된 후 해당 이벤트 객체의 target 속성은 dispatchEvent()가 호출된 객체로 설정됩니다.

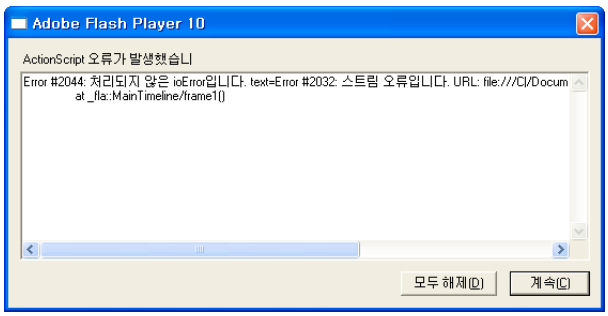
기존 이벤트 리스너 확인

IEventDispatcher 인터페이스의 마지막 두 메서드는 이벤트 리스너가 있는지 여부에 대해 유용한 정보를 제공합니다. 지정된 이벤트 유형의 이벤트 리스너가 특정 표시 목록 객체에 있는 경우 hasEventListener() 메서드에서 true를 반환합니다. 특정 표시 목록 객체에 리스너가 있는 경우 willTrigger() 메서드에서도 true를 반환하지만 willTrigger() 메서드는 해당 표시 객체뿐만 아니라 이벤트 흐름의 모든 단계에 대해 해당 표시 목록 객체의 모든 조상에서도 리스너를 확인합니다.

리스너가 없는 오류 이벤트

Flash Player 9 이상, Adobe AIR 1.0 이상

예외(이벤트 아님)는 ActionScript 3.0에서 오류 처리를 위한 주요 메커니즘으로 사용되지만 파일 로드와 같은 비동기 작업에는 예외 처리가 작동하지 않습니다. 이러한 비동기 작업을 실행하는 동안 오류가 발생하면 Flash Player 및 AIR에서는 오류 이벤트 객체를 전달합니다. 오류 이벤트에 대한 리스너를 만들지 않은 경우에는 Flash Player 및 AIR의 디버거 버전에서 오류 정보가 포함된 대화 상자가 표시됩니다. 예를 들어 응용 프로그램이 잘못된 URL에서 파일을 로드하려고 하면 Flash Player의 디버거 버전에 다음과 같이 오류를 설명하는 대화 상자가 표시됩니다.



대부분의 오류 이벤트는 ErrorEvent 클래스를 기초로 하며 Flash Player 또는 AIR에 표시되는 오류 메시지를 저장하는 데 사용되는 text 속성을 포함합니다. 이때 StatusEvent 클래스와 NetStatusEvent 클래스는 예외입니다. 이러한 두 클래스에는 모두 level 속성(StatusEvent.level 및 NetStatusEvent.info.level)이 있으며, level 속성의 값이 "error"이면 이러한 이벤트 유형이 오류 이벤트로 간주됩니다.

오류 이벤트가 발생해도 SWF 파일은 계속 실행됩니다. 이러한 오류 이벤트는 디버거 버전의 브라우저 플러그인과 독립 실행형 플레이어의 대화 상자, 제작 플레이어 출력 패널의 메시지, Adobe Flash Builder 로그 파일의 항목 등으로만 표시되며, 릴리스 버전의 Flash Player 또는 AIR에는 표시되지 않습니다.

이벤트 처리 예제: 알람 시계

Flash Player 9 이상, Adobe AIR 1.0 이상

알람 시계 예제는 사용자가 알람이 울리는 시간 및 알람이 울릴 때 표시되는 메시지를 지정하는 데 사용할 수 있는 시계로 구성됩니다. 알람 시계 예제는 1페이지의 "날짜 및 시간을 사용한 작업"의 SimpleClock 응용 프로그램을 기초로 구성되며, 다음을 포함하여 ActionScript 3.0에서 이벤트를 사용하는 몇 가지 작업을 보여 줍니다.

- 이벤트 수신 및 응답
- 리스너에 이벤트 알림
- 사용자 정의 이벤트 유형 만들기

이 샘플에 대한 Flash Professional 응용 프로그램 파일을 가져오려면 http://www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. 이 샘플에 대한 Flex 응용 프로그램 파일을 가져오려면 http://www.adobe.com/go/as3examples_kr을 참조하십시오. Alarm Clock 응용 프로그램 파일은 Samples/AlarmClock 폴더에 있으며 이 응용 프로그램은 다음과 같은 파일을 포함합니다.

파일	설명
AlarmClockApp.mxml 또는 AlarmClockApp.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/clock/AlarmClock.as	알람 시계 기능을 추가하여 SimpleClock 클래스를 확장하는 클래스입니다.
com/example/programmingas3/clock/AlarmEvent.as	AlarmClock 클래스의 alarm 이벤트에 대한 이벤트 객체로 사용되는 사용자 정의 이벤트 클래스(flash.events.Event의 하위 클래스)입니다.
com/example/programmingas3/clock/AnalogClockFace.as	현재 시간을 기준으로 시계의 둥근 문자판과 시침, 분침, 초침을 그립니다(SimpleClock 예제의 설명 참조).
com/example/programmingas3/clock/SimpleClock.as	간단한 시간 계측 기능을 포함하는 시계 인터페이스 구성 요소입니다(SimpleClock 예제의 설명 참조).

알람 시계 개요

Flash Player 9 이상, Adobe AIR 1.0 이상

이 예제에 나오는 시계의 주요 기능(예: 시간 추적 및 시계 문자판 표시)은 6페이지의 "날짜 및 시간 예제: 간단한 아날로그 시계"에서 설명하는 SimpleClock 응용 프로그램 코드를 다시 사용합니다. AlarmClock 클래스는 SimpleClock 예제에서 알람 시간 설정, 알람이 "울릴" 때 알림 제공 등 알람 시계에 필요한 기능을 추가하여 SimpleClock 클래스를 확장합니다.

특정 동작이 발생할 때 알림을 제공하는 것은 이벤트의 역할입니다. AlarmClock 클래스는 원하는 액션을 실행하기 위해 다른 객체에서 수신할 수 있는 Alarm 이벤트를 노출합니다. 또한 AlarmClock 클래스는 Timer 클래스의 인스턴스를 사용하여 알람 트리거 시기를 결정할 수 있습니다. Timer 클래스는 AlarmClock 클래스와 같이 지정된 시간이 경과하면 다른 객체(이 예제의 경우 AlarmClock 인스턴스)에 알리는 이벤트를 제공합니다. 대부분의 ActionScript 응용 프로그램과 마찬가지로 이벤트는 Alarm Clock 샘플 응용 프로그램 기능의 중요한 부분입니다.

알람 트리거

Flash Player 9 이상, Adobe AIR 1.0 이상

앞에서 설명했듯이 AlarmClock 클래스에서 실제로 제공하는 기능은 알람 설정 및 트리거뿐입니다. 개발자는 내장 Timer 클래스(flash.utils.Timer)를 통해 지정된 시간이 경과한 후 실행할 코드를 정의할 수 있습니다. AlarmClock 클래스는 Timer 인스턴스를 사용하여 알람 실행 시기를 결정합니다.

```
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);
}
```

AlarmClock 클래스에 정의된 Timer 인스턴스 이름은 alarmTimer입니다. initClock() 메서드는 AlarmClock 인스턴스에 필요한 설정 작업을 수행하며 alarmTimer 변수를 사용하여 두 가지 작업을 실행합니다. 먼저 Timer 인스턴스에서 0밀리초 동안 대기한 후 타이머 이벤트를 한 번 트리거하도록 지시하는 매개 변수를 사용하여 변수가 인스턴스화되고, alarmTimer 변수를 인스턴스화한 후에 이 변수의 addEventListener() 메서드를 호출하여 변수의 timer 이벤트를 수신 대기하도록 요청합니다. 지정된 시간이 경과한 후 timer 이벤트를 전달하면 Timer 인스턴스가 작동합니다. 알람을 울리려면 AlarmClock 클래스에서 timer 이벤트 전달 시기를 알고 있어야 합니다. addEventListener()를 호출하면 AlarmClock 코드가 alarmTimer에 리스너로 등록됩니다. 두 매개 변수는 AlarmClock 클래스에서 timer 이벤트(TimerEvent.TIMER 상수로 나타냄)를 수신 대기할 수 있도록 요청하며, 이벤트가 발생하면 해당 이벤트에 대한 응답으로 AlarmClock 클래스의 onAlarm() 메서드를 호출해야 함을 나타냅니다.

실제로 알람을 설정하기 위해 다음과 같이 AlarmClock 클래스의 setAlarm() 메서드가 호출됩니다.

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

이 메서드는 알람 메시지를 저장하고, 실제로 알람이 울리는 시기를 나타내는 Date 객체(alarmTime)를 만드는 등 여러 가지 작업을 실행합니다. 메서드 정의의 마지막 몇 행에서는 현재 문서에서 주로 다루었던 내용인 alarmTimer 변수의 타이머를 설정하고 활성화합니다. 먼저 reset() 메서드가 호출되어 이미 실행되고 있는 타이머를 중단하고 재설정합니다. 그런 다음, alarmTime 변수의 값에서 현재 시간(now 변수로 나타냄)을 빼 알람이 울리기 전에 경과해야 할 시간(밀리초)을 확인합니다. Timer 클래스는 절대 시간으로 timer 이벤트를 트리거하지 않으므로 alarmTimer의 delay 속성에는 이러한 상대 시간 차이가 지정됩니다. 마지막으로 start() 메서드를 호출하여 실제로 타이머를 시작합니다.

지정된 시간이 경과하면 alarmTimer에서 timer 이벤트를 전달합니다. AlarmClock 클래스는 onAlarm() 메서드를 해당 이벤트의 리스너로 등록했으므로 timer 이벤트가 발생하면 onAlarm()이 호출됩니다.

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

이벤트 리스너로 등록되는 메서드는 적절한 서명, 즉 메서드의 매개 변수 집합 및 반환 유형을 사용하여 정의해야 합니다. 예를 들어 Timer 클래스의 timer 이벤트에 대한 리스너로 등록할 메서드는 데이터 유형이 Event 클래스의 하위 클래스인 TimerEvent(flash.events.TimerEvent)로 설정된 하나의 매개 변수를 정의해야 합니다. 그러면 Timer 인스턴스에서 이벤트 리스너를 호출하는 경우 TimerEvent 인스턴스가 이벤트 객체로 전달됩니다.

다른 코드에 알람 알림

Flash Player 9 이상, Adobe AIR 1.0 이상

AlarmClock 클래스는 Timer 클래스와 마찬가지로 알람이 울릴 때 다른 코드에서 알림을 받을 수 있도록 이벤트를 제공합니다. 클래스에서 ActionScript에 내장된 이벤트 처리 프레임워크를 사용하려면 해당 클래스에서 flash.events.IEventDispatcher 인터페이스를 구현해야 합니다. 일반적으로 이 작업을 수행하려면 flash.events.EventDispatcher 클래스를 확장하여 IEventDispatcher를 구현하거나, EventDispatcher의 하위 클래스 중 하나를 확장합니다. 앞에서 설명했듯이 AlarmClock 클래스는 SimpleClock 클래스를 확장하고 결국 상속 체인을 통해 EventDispatcher 클래스를 확장합니다. 즉, AlarmClock 클래스에는 고유한 이벤트를 제공하는 기능이 이미 포함되어 있습니다.

EventDispatcher에서 AlarmClock에 상속되는 addEventListener() 메서드를 호출하면 다른 코드에 AlarmClock 클래스의 alarm 이벤트를 알리도록 등록할 수 있습니다. AlarmClock 인스턴스에서 alarm 이벤트가 발생했음을 다른 코드에 알릴 준비가 되면 EventDispatcher에서 상속되는 dispatchEvent() 메서드를 호출합니다.

```
var alarm:AlarmEvent = new AlarmEvent(this.alarmMessage);
this.dispatchEvent(alarm);
```

이러한 코드 행은 AlarmClock 클래스의 onAlarm() 메서드에서 가져온 것입니다(전체 코드는 앞에 있음). AlarmClock 인스턴스의 dispatchEvent() 메서드가 호출되면 등록된 모든 리스너에 AlarmClock 인스턴스의 alarm 이벤트가 트리거되었음을 알립니다. dispatchEvent()에 전달되는 매개 변수는 리스너 메서드에 전달될 이벤트 객체이며, 여기에서는 AlarmEvent 클래스의 인스턴스입니다. AlarmEvent 클래스는 이 예제를 위해 만든 Event 하위 클래스입니다.

사용자 정의 alarm 이벤트 제공

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 이벤트 리스너는 트리거되는 특정 이벤트에 대한 정보가 포함된 이벤트 객체 매개 변수를 받습니다. 대부분의 경우 이벤트 객체는 Event 클래스의 인스턴스입니다. 그러나 경우에 따라서는 이벤트 리스너에 추가 정보를 제공하면 유용합니다. 일반적으로 이렇게 하려면 Event 클래스의 하위 클래스인 새 클래스를 정의하고 해당 클래스의 인스턴스를 이벤트 객체로 사용하면 됩니다. 이 예제에서는 AlarmClock 클래스의 alarm 이벤트가 전달되면 AlarmEvent 인스턴스가 이벤트 객체로 사용됩니다. 여기에 표시된 AlarmEvent 클래스는 alarm 이벤트, 특히 알람 메시지에 대한 추가 정보를 제공합니다.

```
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;
    }
    ...
}
```

사용자 정의 이벤트 객체 클래스를 만들려면 위의 예제와 같이 Event 클래스를 확장하는 클래스를 정의하는 것이 가장 좋습니다. 상속된 기능을 보완하기 위해 AlarmEvent 클래스는 이벤트와 관련된 알람 메시지의 텍스트를 포함하는 message 속성을 정의합니다. 이 message 값은 AlarmEvent 생성자의 매개 변수로 전달됩니다. 또한 AlarmEvent 클래스는 AlarmClock 클래스의 addEventListener() 메서드를 호출할 때 특정 이벤트(alarm)를 참조하는 데 사용할 수 있는 ALARM 상수도 정의합니다.

모든 Event 하위 클래스는 사용자 정의 기능을 추가해야 할 뿐만 아니라 ActionScript 이벤트 처리 프레임워크의 일부로 상속 clone() 메서드를 재정의해야 합니다. 또한 Event 하위 클래스는 상속된 toString() 메서드를 재정의하여 toString() 메서드 호출의 반환값에 사용자 정의 이벤트의 속성이 포함되도록 할 수도 있습니다.

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

재정의된 clone() 메서드는 모든 사용자 정의 속성이 현재 인스턴스와 일치하도록 설정된 사용자 정의 Event 하위 클래스의 새 인스턴스를 반환해야 합니다. 재정의된 toString() 메서드에서 유틸리티 메서드 formatToString()(Event에서 상속됨)은 사용자 정의 유형의 이름 및 모든 속성의 이름과 값으로 이루어진 문자열을 제공하는 데 사용됩니다.

9장: 응용 프로그램 도메인 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ApplicationDomain 클래스의 목적은 ActionScript 3.0 정의 표를 저장하는 것입니다. SWF 파일의 모든 코드는 응용 프로그램 도메인에 존재하도록 정의됩니다. 응용 프로그램 도메인을 사용하여 같은 보안 도메인에 있는 여러 클래스를 구분합니다. 그러면 같은 클래스에 대해 여러 가지 정의가 존재할 수 있으며 자식이 부모의 정의를 다시 사용할 수 있습니다.

Loader 클래스 API를 사용하여 ActionScript 3.0에서 작성한 외부 SWF 파일을 로드하는 경우 응용 프로그램 도메인을 사용할 수 있습니다. ActionScript 1.0 또는 ActionScript 2.0에서 작성한 이미지나 SWF 파일을 로드할 때는 응용 프로그램 도메인을 사용할 수 없습니다. 로드된 클래스에 포함된 모든 ActionScript 3.0 정의는 응용 프로그램 도메인에 저장됩니다. SWF 파일을 로드할 때 LoaderContext 객체의 applicationDomain 매개 변수를 ApplicationDomain.currentDomain으로 설정하여 Loader 객체와 동일한 응용 프로그램 도메인에 파일이 포함되도록 지정할 수 있습니다. 로드된 SWF 파일을 동일한 응용 프로그램 도메인에 저장하면 해당 클래스에 직접 액세스할 수 있습니다. 이 방법은 관련된 클래스 이름을 통해 액세스 가능한 포함된 미디어가 들어 있는 SWF 파일을 로드하려는 경우나 로드된 SWF 파일의 메서드에 액세스하려는 경우에 유용할 수 있습니다.

다음 예제에서는 welcome()이라는 공용 메서드를 정의하는 별도의 Greeter.swf 파일에 대한 액세스 권한이 있다고 가정합니다.

```
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
        }
    }
}
```

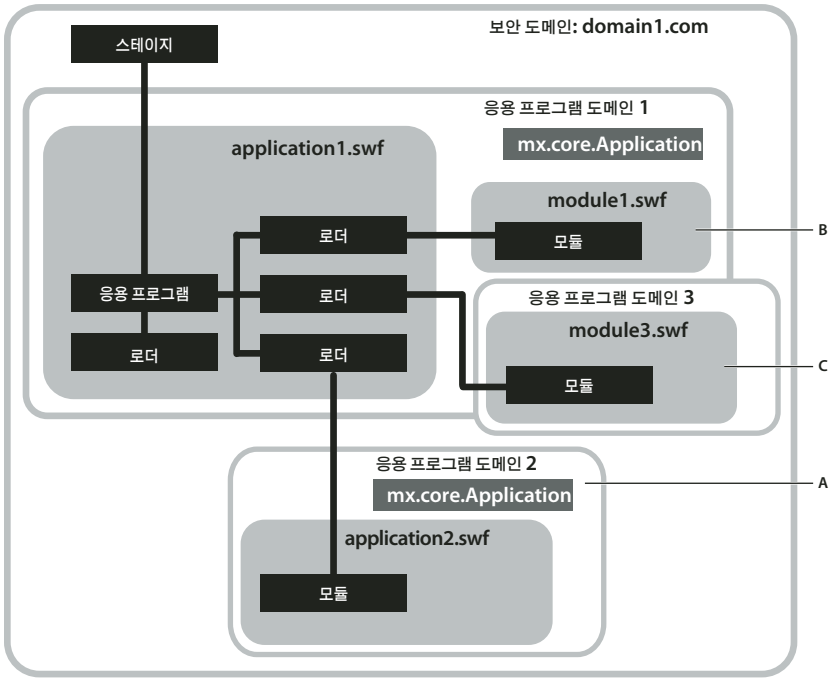
자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 ApplicationDomain 클래스 예제를 참조하십시오.

응용 프로그램 도메인을 사용하는 경우 주의해야 할 또 다른 사항은 다음과 같습니다.

- SWF 파일의 모든 코드는 응용 프로그램 도메인에 존재하도록 정의됩니다. **현재 도메인**은 기본 응용 프로그램이 실행되는 위치입니다. **시스템 도메인**에는 현재 도메인을 포함하여 모든 응용 프로그램이 포함되므로 모든 Flash Player 클래스가 들어 있습니다.

- 시스템 도메인을 제외한 모든 응용 프로그램 도메인에는 연결된 부모 도메인이 있습니다. 기본 응용 프로그램 도메인의 부모 도메인은 시스템 도메인입니다. 로드된 클래스는 부모가 이러한 클래스를 아직 정의하지 않은 경우에만 정의됩니다. 로드된 클래스 정의를 새 정의로 재정의할 수는 없습니다.

다음 다이어그램은 단일 도메인인 domain1.com의 다양한 SWF 파일에서 내용을 로드하는 응용 프로그램을 보여 줍니다. 로드한 내용에 따라 서로 다른 응용 프로그램 도메인이 사용될 수 있습니다. 다음에 나오는 텍스트는 응용 프로그램의 각 SWF 파일에 적합한 응용 프로그램 도메인을 설정하는 데 사용되는 논리를 설명합니다.



A. 구문 A: **B.** 구문 B **C.** 구문 C

기본 응용 프로그램 파일은 application1.swf입니다. 여기에는 다른 SWF 파일에서 내용을 로드하는 Loader 객체가 포함되어 있습니다. 이 시나리오에서 현재 도메인은 응용 프로그램 도메인 1입니다. 구문 A, 구문 B, 구문 C는 응용 프로그램에서 각 SWF 파일에 적합한 응용 프로그램 도메인을 설정하는 여러 가지 기술을 보여 줍니다.

구문 A: 시스템 도메인의 자식을 만들어 자식 SWF 파일을 구분합니다. 다이어그램에서는 응용 프로그램 도메인 2가 시스템 도메인의 자식으로 만들어집니다. application2.swf 파일이 응용 프로그램 도메인 2에 로드되므로, 해당 클래스 정의는 application1.swf에 정의된 클래스에서 구분됩니다.

이 기술은 이전 응용 프로그램에서 같은 응용 프로그램의 새 버전을 충돌 없이 동적으로 로드하는 데 사용할 수 있습니다. 같은 클래스 이름을 사용하더라도 다른 응용 프로그램 도메인으로 구분되므로 충돌이 없습니다.

다음 코드는 시스템 도메인의 자식인 응용 프로그램 도메인을 만들고 이 응용 프로그램 도메인을 사용하여 SWF 로드를 시작합니다.

```
var appDomainA:ApplicationDomain = new ApplicationDomain();

var contextA:LoaderContext = new LoaderContext(false, appDomainA);
var loaderA:Loader = new Loader();
loaderA.load(new URLRequest("application2.swf"), contextA);
```

구문 B: 현재 클래스 정의에 새 클래스 정의를 추가합니다. module1.swf의 응용 프로그램 도메인이 현재 도메인(응용 프로그램 도메인 1)으로 설정됩니다. 그러므로 새 클래스 정의를 사용하여 응용 프로그램의 현재 클래스 정의 설정에 추가할 수 있습니다. 이것은 기본 응용 프로그램의 런타임 공유 라이브러리에 사용할 수 있습니다. 로드된 SWF는 RSL(Remote Shared Library)로 처리됩니다. 응용 프로그램이 시작하기 전에 프리로더로 RSL을 로드하려면 이 기술을 사용하십시오.

다음 코드는 응용 프로그램 도메인을 현재 도메인으로 설정하여 SWF를 로드합니다.

```
var appDomainB:ApplicationDomain = ApplicationDomain.currentDomain;

var contextB:LoaderContext = new LoaderContext(false, appDomainB);
var loaderB:Loader = new Loader();
loaderB.load(new URLRequest("module1.swf"), contextB);
```

구문 C: 현재 도메인의 새로운 자식 도메인을 만들어 부모의 클래스 정의를 사용합니다. module3.swf의 응용 프로그램 도메인은 현재 도메인의 자식이며 자식은 부모 버전의 모든 클래스를 사용합니다. 이 기술은 기본 응용 프로그램 유형을 사용하는 기본 응용 프로그램의 자식으로 로드되는 다중 화면 RIA(Rich Internet Applications)의 모듈로 사용할 수 있습니다. 모든 클래스를 항상 이전 버전과 호환되도록 업데이트하고 로드하는 응용 프로그램이 로드되는 대상보다 항상 새로운 버전인 경우 자식이 부모 버전을 사용합니다. 자식 SWF에 대한 참조를 계속 포함하지 않는 경우 새 응용 프로그램 도메인을 포함하면 가비지 컬렉션의 모든 클래스 정의를 언로드할 수도 있습니다.

이 기술을 사용하면 로드된 모듈이 로더의 단일 객체 및 정적 클래스 멤버를 공유할 수 있습니다.

다음 코드는 현재 도메인의 새 자식 도메인을 만들고 이 응용 프로그램 도메인을 사용하여 SWF 로드를 시작합니다.

```
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);
```

10장: 디스플레이 프로그래밍

Flash Player 9 이상, Adobe AIR 1.0 이상

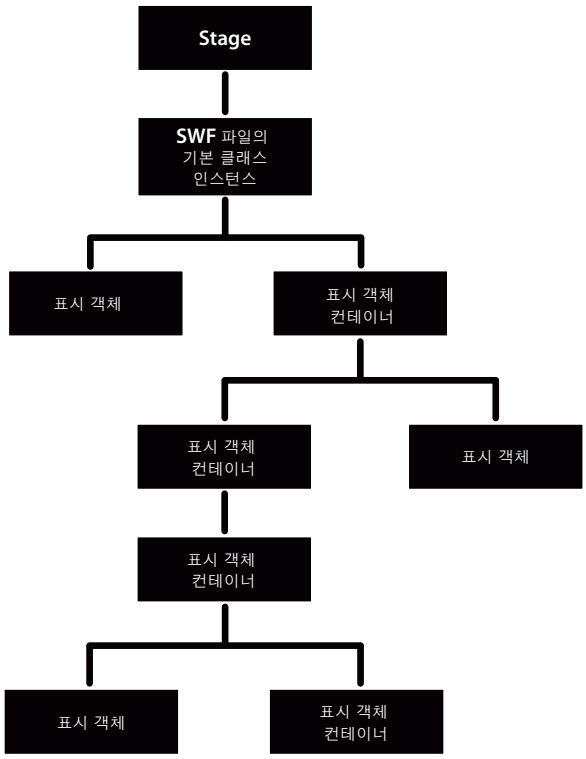
표시 스테이지의 표시 객체를 사용하여 작업하는 방식으로 Adobe® ActionScript® 3.0에서 시각적 요소를 프로그래밍할 수 있습니다. 예를 들어 ActionScript 디스플레이 프로그래밍 API를 사용하여 3차원 변형을 수행하고, 벡터 및 비트맵 그래픽을 그리거나 필터와 마스크를 적용하고 표시 객체를 추가, 이동, 제거, 순서 지정할 수 있습니다. 디스플레이 프로그래밍에 사용되는 기본 클래스는 flash.display 패키지의 일부입니다.

참고: Adobe® AIR™는 HTML 내용을 렌더링하고 표시할 수 있는 HTMLoader 객체를 제공합니다. HTMLLoader는 HTML DOM의 시각적 요소를 하나의 표시 객체로 렌더링합니다. ActionScript 표시 목록 계층 구조를 통해 DOM의 개별 요소에 직접 액세스할 수는 없으며 대신 HTMLLoader가 제공하는 별도의 DOM API를 사용하여 이러한 DOM 요소에 액세스할 수 있습니다.

디스플레이 프로그래밍의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0으로 작성된 각 응용 프로그램에는 아래 그림과 같이 표시 목록이라는 표시된 객체의 계층이 있습니다. 이 표시 목록에는 응용 프로그램에 보이는 요소가 모두 포함되어 있습니다.



그림에 표시된 것과 같이 표시 요소는 다음 그룹 중 하나 이상에 속합니다.

- 스테이지

  스테이지는 표시 객체의 기본 컨테이너입니다. 각 응용 프로그램에는 모든 화면 표시 객체를 포함하는 Stage 객체가 하나 있습니다. 스테이지는 최상위 컨테이너이며 표시 목록 계층의 맨 위에 있습니다.

  각 SWF 파일에는 **SWF 파일의 기본 클래스**라는 관련된 ActionScript 클래스가 있습니다. Flash Player 또는 Adobe AIR에서 SWF 파일이 열리면 해당 클래스의 생성자 함수가 호출됩니다. 그러면 생성되는 인스턴스(항상 표시 객체의 유형)가 Stage 객체의 자식으로 추가됩니다. SWF 파일의 기본 클래스는 항상 Sprite 클래스를 확장합니다. 자세한 내용은 140페이지의 “표시 목록 방식의 장점”을 참조하십시오.

  DisplayObject 인스턴스의 stage 속성을 통해 스테이지에 액세스할 수 있습니다. 자세한 내용은 148페이지의 “Stage 속성 설정”을 참조하십시오.

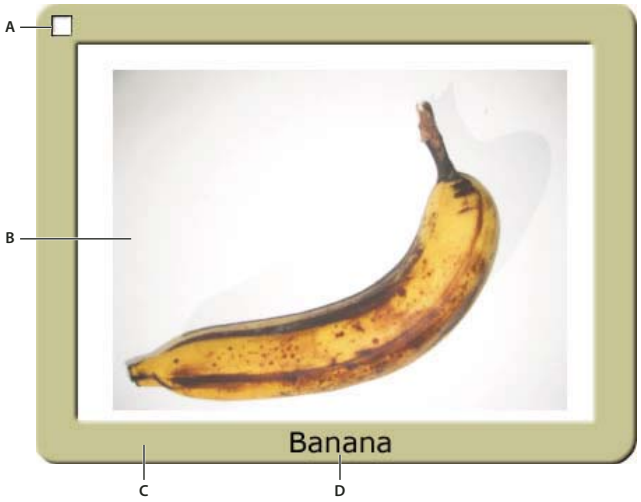
- 표시 객체

  ActionScript 3.0에서 응용 프로그램의 화면에 표시되는 모든 요소는 표시 객체 유형입니다. flash.display 패키지에는 다른 여러 클래스에 의해 확장되는 기본 클래스인 DisplayObject 클래스가 포함되어 있습니다. 이러한 각 클래스는 벡터 모양, 동영상 클립, 텍스트 필드 등과 같은 서로 다른 유형의 표시 객체를 나타냅니다. 이러한 클래스에 대한 개요는 140페이지의 “표시 목록 방식의 장점”을 참조하십시오.

- 표시 객체 컨테이너

  표시 객체 컨테이너는 고유한 시각적 표현이 있고 자식 객체를 포함할 수 있는 특수 유형의 표시 객체입니다. 이때 자식 객체도 표시 객체입니다.

  DisplayObjectContainer 클래스는 DisplayObject 클래스의 하위 클래스입니다. DisplayObjectContainer 객체는 자식 목록에 여러 표시 객체를 포함할 수 있습니다. 예를 들어, 다음 그림에서는 다양한 표시 객체를 포함하는 Sprite라는 DisplayObjectContainer 객체 유형을 보여 줍니다.



A. SimpleButton 객체. 이 유형의 표시 객체는 "업", "다운" 및 "오버" 상태를 가집니다. **B.** Bitmap 객체. 이 경우 Bitmap 객체는 Loader 객체를 통해 외부 JPEG에서 로드됩니다. **C.** Shape 객체. "그림 프레임"에는 ActionScript에서 그린 둥근 사각형이 포함되어 있습니다. 이 Shape 객체에는 그림자 필터가 적용되어 있습니다. **D.** TextField 객체

표시 객체에 대해 언급할 때는 DisplayObjectContainer 객체도 표시 객체 컨테이너 또는 간단히 컨테이너라고 합니다. 앞에서 설명한 것처럼 스테이지는 표시 객체 컨테이너입니다.

모든 보이는 표시 객체는 DisplayObject 클래스에서 상속되지만 각 유형은 DisplayObject 클래스의 특정 하위 클래스입니다. 예를 들어, Shape 클래스 또는 Video 클래스에 대한 생성자 함수가 있지만, DisplayObject 클래스에 대한 생성자 함수는 없습니다.

중요한 개념 및 용어

ActionScript 그래픽을 프로그래밍할 때 사용되는 중요한 용어가 아래 참조 목록에 정리되어 있습니다.

알파 색상의 투명도(보다 정확하게는 불투명도)를 나타내는 색상 값입니다. 예를 들어 알파 채널 값이 60%인 색상은 전체 강도의 60%만 표시되고 40%는 투명하게 나타납니다.

비트맵 그래픽 컴퓨터에서 색상 픽셀의 격자(행 및 열)로 정의되는 그래픽입니다. 일반적으로 비트맵 그래픽에는 디지털 사진 및 이와 유사한 이미지가 포함됩니다.

블렌딩 모드 겹치는 두 개의 이미지 내용이 서로 영향을 미치는 방식을 지정하는 것입니다. 일반적으로 불투명한 이미지가 다른 이미지 위에 놓이게 되면 아래쪽의 이미지를 가려서 아래쪽 이미지가 전혀 보이지 않지만, 블렌딩 모드를 다르게 하면 이미지 색상의 블렌드 방식이 변경되어 두 개 이미지가 혼합되어 나타납니다.

표시 목록 Flash Player 및 AIR에 의해 가시 화면 내용으로 렌더링되는 표시 객체 계층 구조입니다. 스테이지는 표시 목록의 루트이며, 스테이지 또는 해당 자식 중 하나에 연결된 모든 표시 객체가 표시 목록을 구성합니다(객체가 스테이지 경계 외부에 있는 등 실제로 렌더링되지 않은 경우도 포함).

표시 객체 Flash Player 또는 AIR의 시각적 내용 중 일부 유형을 나타내는 객체입니다. 표시 객체만 표시 목록에 포함될 수 있으며 모든 표시 객체 클래스는 DisplayObject 클래스의 하위 클래스입니다.

표시 객체 컨테이너 일반적으로 시각적 표현을 가지며 자식 표시 객체를 포함할 수 있는 특수한 유형의 표시 객체입니다.

SWF 파일의 기본 클래스 SWF 파일에서 가장 바깥쪽 표시 객체의 비헤이비어를 정의하는 클래스로, 개념적으로 SWF 파일 자체의 클래스를 나타냅니다. 예를 들어 Flash 제작에서 만들어진 SWF에서 기본 클래스는 문서 클래스입니다. 이 클래스에는 다른 모든 타임라인을 포함하는 기본 타임라인이 있으며, SWF 파일의 기본 클래스는 기본 타임라인이 인스턴스가 되는 클래스입니다.

마스크 이미지의 특정 부분이 보이지 않도록 숨기는, 즉 이미지의 특정 부분만 표시되도록 하는 기술입니다. 마스크 이미지 부분이 투명하게 되므로 아래쪽 내용도 볼 수 있습니다. 이 용어는 특정 영역을 칠하지 않을 때 사용하는 마스크 테이프와 관계 있습니다.

Stage SWF의 모든 시각적 내용의 기반이나 배경이 되는 시각적 컨테이너입니다.

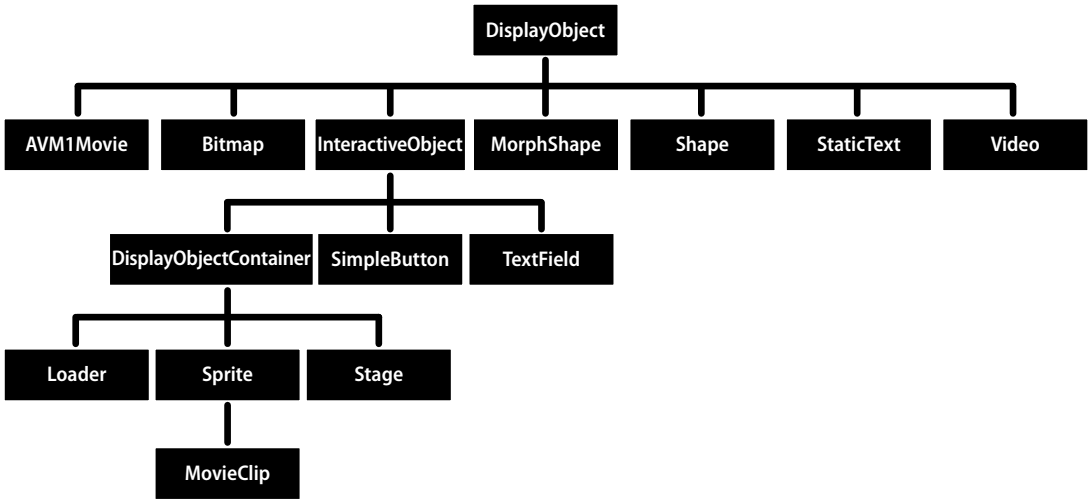
변형 객체 회전, 크기 변경, 모양 기울이기 또는 왜곡, 색상 변경 등 그래픽의 시각적 특성을 조절합니다.

벡터 그래픽 컴퓨터에서 특정 특성(예: 두께, 길이, 크기, 각도, 위치)으로 그려진 선 및 모양으로 정의되는 그래픽입니다.

기본 표시 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0 flash.display 패키지에는 Flash Player 또는 AIR에서 표시될 수 있는 시각적 객체에 대한 클래스가 포함되어 있습니다. 다음 그림에서는 이러한 기본 표시 객체 클래스의 하위 클래스 관계를 보여 줍니다.



이 그림에서는 표시 객체 클래스의 클래스 상속을 보여 줍니다. StaticText, TextField 및 Video와 같은 일부 클래스는 flash.display 패키지에 없지만, DisplayObject 클래스에서 여전히 상속됩니다.

DisplayObject 클래스를 확장하는 모든 클래스는 해당 메서드와 속성을 상속합니다. 자세한 내용은 142페이지의 "DisplayObject 클래스의 속성 및 메서드"를 참조하십시오.

flash.display 패키지에 포함된 다음 클래스의 객체를 인스턴스화할 수 있습니다.

- Bitmap - Bitmap 클래스를 사용하여 외부 파일에서 로드되거나 ActionScript를 통해 렌더링되는 비트맵 객체를 정의할 수 있습니다. Loader 클래스를 통해 외부 파일에서 비트맵을 로드할 수 있습니다. GIF, JPG 또는 PNG 파일을 로드할 수 있습니다. 또한 사용자 정의 데이터를 사용하여 BitmapData 객체를 만든 다음 해당 데이터를 사용하는 Bitmap 객체를 만들 수 있습니다. BitmapData 클래스의 메서드를 사용하면 ActionScript에서 로드하거나 만든 비트맵을 변경할 수 있습니다. 자세한 내용은 178페이지의 "표시 객체 로드" 및 219페이지의 "비트맵을 사용한 작업"을 참조하십시오.
- Loader - Loader 클래스를 사용하여 외부 에셋(SWF 파일 또는 그래픽)을 로드할 수 있습니다. 자세한 내용은 178페이지의 "표시 내용을 동적으로 로드"를 참조하십시오.
- Shape - Shape 클래스를 사용하여 사각형, 선, 원 등과 같은 벡터 그래픽을 만들 수 있습니다. 자세한 내용은 200페이지의 "드로잉 API 사용"을 참조하십시오.
- SimpleButton - SimpleButton 객체는 Flash 제작 도구로 만든 버튼 심볼을 ActionScript로 표현한 것입니다. SimpleButton 인스턴스에는 업, 다운, 오버 및 히트 테스트(마우스 및 키보드 이벤트에 응답하는 영역)의 네 가지 버튼 상태가 있습니다.
- Sprite - Sprite 객체는 자체 그래픽과 자식 표시 객체를 포함할 수 있습니다. Sprite 클래스는 DisplayObjectContainer 클래스를 확장합니다. 자세한 내용은 143페이지의 "표시 객체 컨테이너 작업" 및 200페이지의 "드로잉 API 사용"을 참조하십시오.
- MovieClip - MovieClip 객체는 Flash 제작 도구로 만든 동영상 클립 심볼의 ActionScript 형식입니다. 실제로 MovieClip은 타임라인이 있다는 점을 제외하고 Sprite 객체와 비슷합니다. 자세한 내용은 291페이지의 "동영상 클립을 사용한 작업"을 참조하십시오.

flash.display 패키지에 없는 다음 클래스는 DisplayObject 클래스의 하위 클래스입니다.

- flash.text 패키지에 포함된 TextField 클래스는 텍스트 표시 및 입력을 위한 표시 객체입니다. 자세한 내용은 334페이지의 "텍스트를 사용한 작업의 기초"를 참조하십시오.
- flash.text.engine 패키지에 포함된 TextLine 클래스는 Flash Text Engine 및 Text Layout Framework에서 구성된 텍스트의 선을 표시하는 데 사용되는 표시 객체입니다. 자세한 내용은 359페이지의 "Flash Text Engine 사용" 및 386페이지의 "Text Layout Framework 사용"을 참조하십시오.
- flash.media 패키지에 포함된 Video 클래스는 비디오 파일을 표시하는 데 사용되는 표시 객체입니다. 자세한 내용은 429페이지의 "비디오를 사용한 작업"을 참조하십시오.

flash.display 패키지에 있는 다음 클래스는 DisplayObject 클래스를 확장하지만 해당 클래스의 인스턴스를 만들 수 없습니다. 이러한 클래스는 공통 기능을 하나의 클래스로 결합하여 다른 표시 객체에 대한 부모 클래스 역할을 합니다.

- AVM1Movie - AVM1Movie 클래스는 ActionScript 1.0 및 2.0에서 만든 로드된 SWF 파일을 나타내는 데 사용됩니다.
- DisplayObjectContainer - Loader, Stage, Sprite 및 MovieClip 클래스는 각각 DisplayObjectContainer 클래스를 확장합니다. 자세한 내용은 143페이지의 "표시 객체 컨테이너 작업"을 참조하십시오.
- InteractiveObject - InteractiveObject 클래스는 마우스 및 키보드와 상호 작용하는 데 사용되는 모든 객체에 대한 기본 클래스입니다. SimpleButton, TextField, Loader, Sprite, Stage 및 MovieClip 객체는 모두 InteractiveObject 클래스의 하위 클래스입니다. 마우스 및 키보드 상호 작용 만들기에 대한 자세한 내용은 505페이지의 "사용자 상호 작용의 기초"를 참조하십시오.
- MorphShape - 이러한 객체는 Flash 제작 도구에서 모양 트윈을 만들 때 만들어집니다. 이러한 객체는 ActionScript를 사용하여 인스턴스화할 수 없지만 표시 목록에서 액세스할 수 있습니다.
- Stage - Stage 클래스는 DisplayObjectContainer 클래스를 확장합니다. 하나의 응용 프로그램에는 표시 목록 계층의 맨 위에 하나의 Stage 인스턴스가 있습니다. Stage에 액세스하려면 DisplayObject 인스턴스의 stage 속성을 사용합니다. 자세한 내용은 148페이지의 "Stage 속성 설정"을 참조하십시오.

또한 flash.text 패키지의 StaticText 클래스는 DisplayObject 클래스를 확장하지만, 코드에 해당 인스턴스를 만들 수 없습니다. 정적 텍스트 필드는 Flash에서만 만들 수 있습니다.

다음 클래스는 표시 객체 또는 표시 객체 컨테이너가 아니며 표시 목록에 나타나지 않지만 스테이지에 그래픽을 표시합니다. 이러한 클래스는 스테이지의 상대적 위치에 뷰포트라고 하는 사각형을 그립니다.

- StageVideo - StageVideo 클래스는 가능한 경우 하드웨어 가속을 사용하여 비디오 내용을 표시합니다. 이 클래스는 Flash Player 10.2부터 사용할 수 있습니다. 자세한 내용은 463페이지의 "하드웨어 가속 프레젠테이션에 StageVideo 클래스 사용"을 참조하십시오.
- StageWebView - StageWebView 클래스는 HTML 내용을 표시합니다. 이 클래스는 AIR 2.5부터 지원됩니다. 자세한 내용은 933페이지의 "StageWebView 객체"를 참조하십시오.

다음 fl.display 클래스는 flash.display.Loader 및 LoaderInfo 클래스와 유사한 기능을 제공합니다. Flash Professional 환경(CS5.5 이상)에서 개발하는 경우 flash.display 클래스 대신 이 클래스를 사용하십시오. Flash Professional 환경(CS5.5 이상)에서는 이 클래스가 RSL을 미리 로드할 때 발생하는 TLF 관련 문제를 해결하는 데 도움이 됩니다. 자세한 내용은 181페이지의 "ProLoader 및 ProLoaderInfo 클래스 사용"을 참조하십시오.

- fl.display.ProLoader—flash.display.Loader와 유사함
- fl.display.ProLoaderInfo—flash.display.LoaderInfo와 유사함

표시 목록 방식의 장점

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 표시 객체 유형마다 별도의 클래스가 있습니다. ActionScript 1.0 및 2.0에는 많은 동일한 유형의 객체가 MovieClip 클래스 하나에 모두 포함되어 있습니다.

이 클래스 개별화 및 표시 목록 계층 구조에는 다음과 같은 장점이 있습니다.

- 보다 효율적인 렌더링 및 메모리 사용 감소
- 심도 관리 개선
- 표시 목록 전체 순회
- 목록 외 표시 객체
- 보다 쉬운 표시 객체 하위 클래스화

보다 효율적인 렌더링 및 파일 크기 축소

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 1.0 및 2.0에서는 MovieClip 객체에서만 모양을 그릴 수 있습니다. ActionScript 3.0에는 모양을 그릴 수 있는 간단한 표시 객체 클래스가 있습니다. 이러한 ActionScript 3.0 표시 객체 클래스는 MovieClip 객체에 포함되는 전체 메서드와 속성을 포함하지 않기 때문에, 메모리 및 프로세서 리소스 사용을 줄일 수 있습니다.

예를 들어, 각 MovieClip 객체는 동영상 클립의 타임라인 속성을 포함하지만 Shape 객체는 이 속성을 포함하지 않습니다. 타임라인 관리를 위한 속성은 많은 메모리와 프로세서 리소스를 사용할 수 있습니다. ActionScript 3.0에서는 Shape 객체를 사용하여 성능을 높일 수 있습니다. Shape 객체는 복잡한 MovieClip 객체보다 오버헤드가 적습니다. Flash Player 및 AIR에서 사용되지 않는 MovieClip 속성을 관리할 필요가 없으므로, 속도가 향상되고 객체에서 사용하는 메모리 용량도 감소됩니다.

심도 관리 개선

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 1.0 및 2.0에서는 심도가 getNextHighestDepth()와 같은 선형 심도 관리 스키마와 메서드를 통해 관리됩니다.

ActionScript 3.0에는 표시 객체의 심도를 관리하는 보다 편리한 메서드와 속성을 가진 DisplayObjectContainer 클래스가 포함되어 있습니다.

ActionScript 3.0에서는 표시 객체를 DisplayObjectContainer 인스턴스의 자식 목록에서 새 위치로 이동하면 표시 객체 컨테이너에 있는 다른 자식 객체들의 위치가 자동으로 다시 배치되고 표시 객체 컨테이너의 적절한 자식 인덱스 위치에 할당됩니다.

또한 ActionScript 3.0에서는 표시 객체 컨테이너의 모든 자식 객체를 항상 검색할 수 있습니다. 모든 DisplayObjectContainer 인스턴스에는 표시 객체 컨테이너에 있는 자식 수를 나열하는 numChildren 속성이 있습니다. 표시 객체 컨테이너의 자식 목록에는 항상 인덱스가 지정되어 있으므로, 인덱스 위치 0부터 마지막 인덱스 위치(numChildren - 1)까지 목록의 모든 객체를 검사할 수 있습니다. 이 작업은 ActionScript 1.0 및 2.0에 있는 MovieClip 객체의 메서드와 속성으로는 가능하지 않습니다.

ActionScript 3.0에서는 표시 객체 컨테이너 자식 목록의 인덱스 번호에 빠진 번호가 없기 때문에 표시 목록을 차례대로 쉽게 순회할 수 있습니다. ActionScript 1.0 및 2.0에서보다 표시 목록을 순회하고 객체 심도를 관리하기가 훨씬 쉬워졌습니다. ActionScript 1.0 및 2.0에서는 동영상 클립이 심도 순서에 간격을 두고 객체를 포함할 수 있기 때문에 객체 목록을 순회하기가 어려울 수 있습니다. ActionScript 3.0에서는 표시 객체 컨테이너의 각 자식 목록이 내부적으로 단일 배열로 캐시되므로 인덱스를 기준으로 매우 빠르게 조회할 수 있습니다. 표시 객체 컨테이너의 모든 자식을 매우 빠르게 반복할 수도 있습니다.

ActionScript 3.0에서는 DisplayObjectContainer 클래스의 getChildByName() 메서드를 사용하여 표시 객체 컨테이너의 자식에 액세스할 수도 있습니다.

표시 목록 전체 순회

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 1.0 및 2.0에서는 Flash 제작 도구에서 그린 벡터 모양과 같은 일부 객체에 액세스할 수 없습니다. ActionScript 3.0에서는 ActionScript에서 만든 객체와 Flash 제작 도구에서 만든 모든 표시 객체를 포함하여 표시 목록에 있는 모든 객체에 액세스할 수 있습니다. 자세한 내용은 147페이지의 “표시 목록 순회”를 참조하십시오.

목록 외 표시 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 보이는 표시 목록에 없는 표시 객체를 만들 수 있습니다. 이러한 표시 객체를 **목록 외** 표시 객체라고 합니다. 표시 객체는 표시 목록에 이미 추가된 DisplayObjectContainer 인스턴스의 addChild() 또는 addChildAt() 메서드를 호출할 경우에만 보이는 표시 목록에 추가됩니다.

목록 외 표시 객체를 사용하면, 여러 표시 객체가 포함된 표시 객체 컨테이너가 여러 개 있는 객체처럼 복잡한 표시 객체를 모을 수 있습니다. 표시 객체를 목록 외 상태로 유지하면, 이러한 표시 객체를 렌더링하는 데 처리 시간을 사용하지 않고도 복잡한 객체를 모을 수 있습니다. 그런 다음 필요에 따라 표시 목록에 목록 외 표시 객체를 추가할 수 있습니다. 또한 표시 객체 컨테이너의 자식을 표시 목록 내외 및 표시 목록 내의 원하는 위치로 이동할 수 있습니다.

보다 쉬운 표시 객체 하위 클래스화

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 1.0 및 2.0에서는 기본 모양을 만들거나 비트맵을 표시하려면 SWF 파일에 새 MovieClip 객체를 추가해야 합니다. ActionScript 3.0에는 DisplayObject 클래스에 Shape, Bitmap 등과 같은 많은 내장 하위 클래스가 포함되어 있습니다. ActionScript 3.0의 클래스는 특정 객체 유형에 대해 더 구체화되어, 내장 클래스의 기본 하위 클래스를 쉽게 만들 수 있습니다.

예를 들어, ActionScript 2.0에서 원을 그리려면 사용자 정의 클래스 객체가 인스턴스화될 때 MovieClip 클래스를 확장하는 CustomCircle 클래스를 만들 수 있습니다. 그러나, 이 클래스는 해당 클래스에 적용되지 않는 MovieClip 클래스의 많은 속성과 메서드(예: totalFrames)를 포함하고 있습니다. ActionScript 3.0에서는 Shape 객체를 확장하는 CustomCircle 클래스를 만들 수 있으므로, MovieClip 클래스에 포함되어 있지만 관련 없는 속성과 메서드는 이 클래스에 포함되지 않습니다. 다음 코드는 CustomCircle 클래스의 예를 보여 줍니다.

```
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);
    }
}
```

표시 객체 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

지금까지 스테이지, 표시 객체, 표시 객체 컨테이너 및 표시 목록의 기본 개념에 대해 살펴보았으며, 이 단원에서는 ActionScript 3.0을 사용한 표시 객체 작업에 대해 보다 자세한 정보를 제공합니다.

DisplayObject 클래스의 속성 및 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 표시 객체는 DisplayObject 클래스의 하위 클래스이므로 DisplayObject 클래스의 속성과 메서드를 상속합니다. 상속되는 속성은 모든 표시 객체에 적용되는 기본 속성입니다. 예를 들어, 각 표시 객체에는 표시 객체 컨테이너에서 객체의 위치를 지정하는 x 속성과 y 속성이 있습니다.

DisplayObject 클래스 생성자를 사용하여 DisplayObject 인스턴스를 만들 수 없습니다. new 연산자를 사용하여 객체를 인스턴스화하려면 Sprite와 같은 다른 유형의 객체(DisplayObject 클래스의 하위 클래스인 객체)를 만들어야 합니다. 또한 사용자 정의 표시 객체 클래스를 만들려면 Shape 클래스 또는 Sprite 클래스처럼 사용 가능한 생성자 함수가 있는 표시 객체 하위 클래스 중 하나의 하위 클래스를 만들어야 합니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 DisplayObject 클래스 설명을 참조하십시오.

표시 목록에 표시 객체 추가

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체를 인스턴스화할 경우 표시 목록에 있는 표시 객체 컨테이너에 표시 객체 인스턴스를 추가할 때까지는 해당 표시 객체가 스테이지의 화면에 표시되지 않습니다. 예를 들어, 다음 코드에서 myText TextField 객체는 코드의 마지막 행을 생략하면 표시되지 않습니다. 코드의 마지막 행에서 this 키워드는 표시 목록에 이미 추가된 표시 객체 컨테이너를 나타내야 합니다.

```
import flash.display.*;
import flash.text.TextField;
var myText:TextField = new TextField();
myText.text = "Buenos dias.";
this.addChild(myText);
```

스테이지에 시각적 요소를 추가하면 해당 요소가 Stage 객체의 자식이 됩니다. 응용 프로그램에 로드되는 첫 번째 SWF 파일(예: HTML 페이지에 포함된 파일)은 Stage 객체의 자식으로 자동 추가됩니다. Sprite 클래스를 확장하는 모든 유형의 객체가 자식이 될 수 있습니다.

ActionScript를 사용하지 않고(예: Flex MXML 파일에서 MXML 태그를 추가하거나 Flash Professional의 Stage에 항목 배치) 만든 모든 표시 객체가 표시 목록에 추가됩니다. 이러한 표시 객체를 ActionScript를 통해 추가하지 않지만 ActionScript를 통해 액세스할 수 있습니다. 예를 들어, 다음 코드는 ActionScript가 아니라 제작 도구에서 추가한 button1이라는 객체의 폭을 조절합니다.

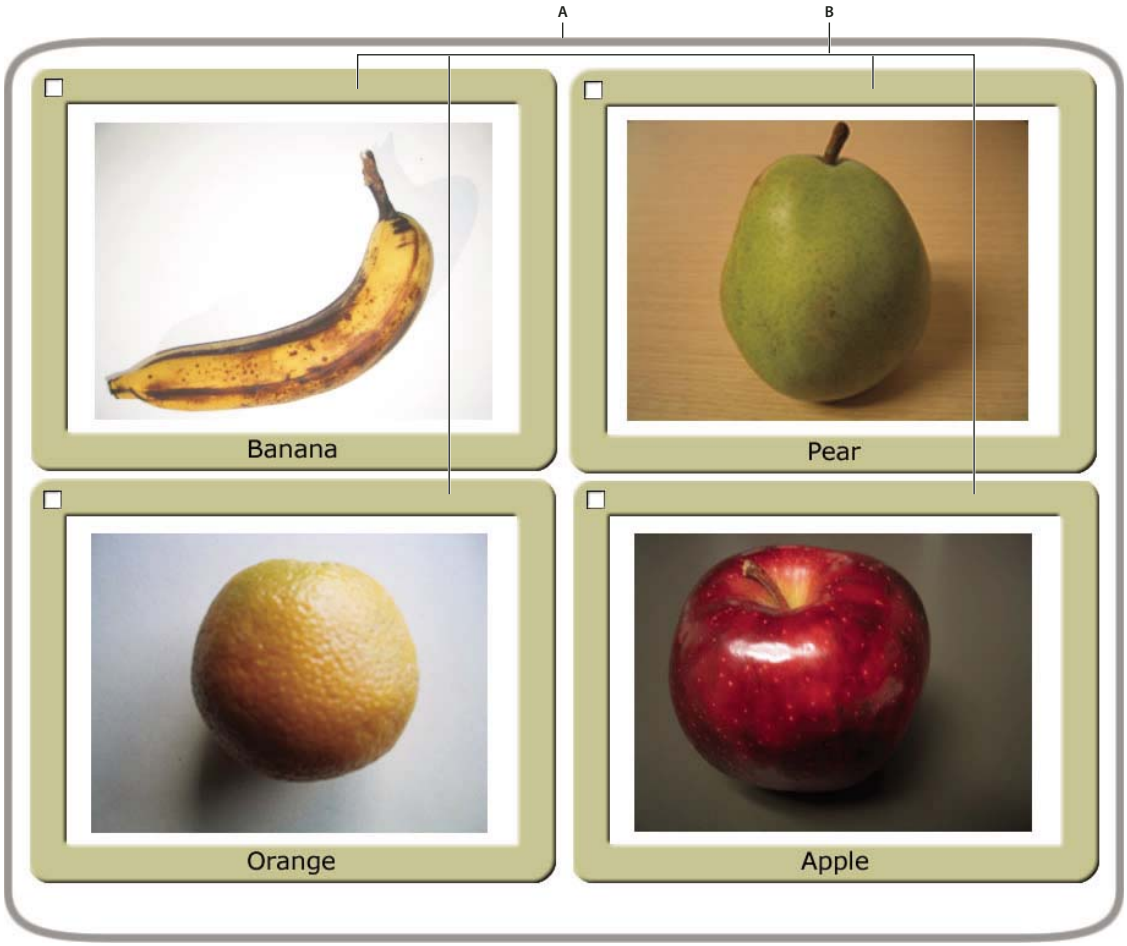
```
button1.width = 200;
```

표시 객체 컨테이너 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

DisplayObjectContainer 객체가 표시 목록에서 삭제되거나 다른 방식으로 이동 또는 변형될 경우 DisplayObjectContainer의 각 표시 객체도 함께 삭제, 이동 또는 변형됩니다.

표시 객체 컨테이너는 그 자체가 표시 객체의 한 유형이므로 다른 표시 객체 컨테이너에 추가될 수 있습니다. 예를 들어, 다음 이미지는 하나의 외곽선 모양과 다른 네 개의 표시 객체 컨테이너(PictureFrame 유형)를 포함하는 표시 객체 컨테이너 pictureScreen을 보여 줍니다.



A. pictureScreen 표시 객체 컨테이너의 테두리를 정의하는 모양 **B.** pictureScreen 객체의 자식인 네 개의 표시 객체 컨테이너

표시 객체가 표시 목록에 나타나게 하려면 표시 객체를 표시 목록에 있는 표시 객체 컨테이너에 추가해야 합니다. 이때 컨테이너 객체의 addChild() 메서드 또는 addChildAt() 메서드를 사용합니다. 예를 들어, 다음 코드의 마지막 행이 없으면 myTextField 객체가 표시되지 않습니다.

```
var myTextField:TextField = new TextField();
myTextField.text = "hello";
this.root.addChild(myTextField);
```

이 코드 목록에서 this.root는 코드를 포함하는 MovieClip 표시 객체 컨테이너를 가리킵니다. 실제 코드에서는 다른 컨테이너를 지정할 수 있습니다.

addChildAt() 메서드를 사용하여 표시 객체 컨테이너 자식 목록의 특정 위치에 자식을 추가합니다. 자식 목록에서 0부터 시작하는 인덱스 위치는 표시 객체의 레이어 위치(전후 순서)와 관련이 있습니다. 예를 들어, 다음 세 표시 객체를 생각해 볼 수 있습니다. 각 객체는 Ball이라는 사용자 정의 객체로부터 만들어졌습니다.

addChildAt() 메서드를 사용하여 컨테이너에서 이러한 표시 객체의 레이어 위치를 조정할 수 있습니다. 예를 들어 다음과 같은 코드를 살펴봅니다.

```
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);
```

이 코드를 실행하면 container DisplayObjectContainer 객체에서 표시 객체가 다음과 같이 배치됩니다. 객체의 레이어 위치에 주목하십시오.

객체의 위치를 표시 목록의 맨 위로 이동하려면 목록에 해당 객체를 다시 추가하면 됩니다. 예를 들어, 이전 코드를 실행한 후에 ball_A를 스택의 맨 위로 이동하려면 다음 코드 행을 사용합니다.

```
container.addChild(ball_A);
```

이 코드는 container 표시 목록의 해당 위치에서 ball_A를 효과적으로 제거한 후 목록의 맨 위에 다시 추가합니다. 그러면 해당 객체가 스택의 맨 위로 이동합니다.

getChildAt() 메서드를 사용하여 표시 객체의 레이어 순서를 확인할 수 있습니다. getChildAt() 메서드는 전달된 인덱스 번호를 기반으로 컨테이너의 자식 객체를 반환합니다. 예를 들어, 다음 코드는 container DisplayObjectContainer 객체의 자식 목록에서 서로 다른 위치에 있는 표시 객체의 이름을 표시합니다.

```
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_C
trace(container.getChildAt(2).name); // ball_B
```

부모 컨테이너의 자식 목록에서 표시 객체를 제거하면 목록에서 더 높은 위치에 있는 요소가 자식 인덱스에서 한 위치씩 아래로 이동합니다. 예를 들어, 이전 코드에서 다음 코드는 자식 목록에서 하위에 있는 표시 객체를 제거함에 따라 container DisplayObjectContainer의 위치 2에 있는 표시 객체가 위치 1로 어떻게 이동하는지 보여 줍니다.

```
container.removeChild(ball_C);
trace(container.getChildAt(0).name); // ball_A
trace(container.getChildAt(1).name); // ball_B
```

removeChild() 및 removeChildAt() 메서드는 표시 객체 인스턴스를 완전히 삭제하지 않고, 컨테이너의 자식 목록에서만 제거합니다. 따라서 다른 변수가 인스턴스를 계속 참조할 수 있습니다. 객체를 완전히 제거하려면 delete 연산자를 사용합니다.

표시 객체는 부모 컨테이너가 하나뿐이므로, 표시 객체 인스턴스를 하나의 표시 객체 컨테이너에만 추가할 수 있습니다. 예를 들어, 다음 코드는 표시 객체 tf1이 하나의 컨테이너(이 경우 DisplayObjectContainer 클래스를 확장하는 Sprite)에만 존재할 수 있음을 보여 줍니다.

```
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
```

한 표시 객체 컨테이너에 포함된 표시 객체를 다른 표시 객체 컨테이너에 추가하면 해당 표시 객체가 첫 번째 표시 객체 컨테이너의 자식 목록에서 제거됩니다.

위에서 설명한 메서드 외에도 DisplayObjectContainer 클래스는 자식 표시 객체 작업을 위한 다음과 같은 다양한 메서드를 정의합니다.

- contains(): 표시 객체가 DisplayObjectContainer의 자식인지 여부를 결정합니다.
- getChildByName(): 이름을 기준으로 표시 객체를 검색합니다.
- getChildIndex(): 표시 객체의 인덱스 위치를 반환합니다.
- setChildIndex(): 자식 표시 객체의 위치를 변경합니다.
- removeChildren(): 여러 자식 표시 객체를 제거합니다.
- swapChildren(): 두 표시 객체의 순서(전후 순서)를 교체합니다.
- swapChildrenAt(): 인덱스 값으로 지정된 두 표시 객체의 순서(전후 순서)를 교체합니다.

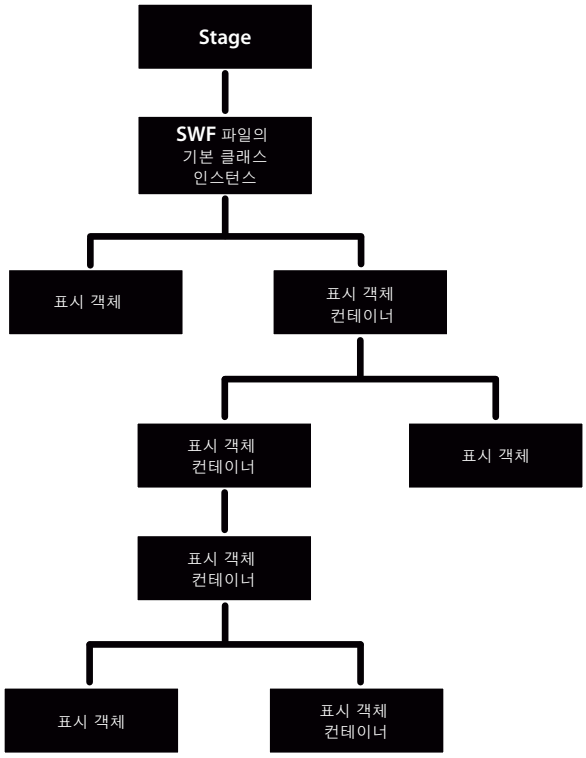
자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 관련 항목을 참조하십시오.

표시 목록에 없는 표시 객체(Stage 객체의 자식인 표시 객체 컨테이너에 포함되지 않은 표시 객체)를 **목록 외** 표시 객체라고 합니다.

표시 목록 순회

Flash Player 9 이상, Adobe AIR 1.0 이상

이미 알고 있듯이 표시 목록은 트리 구조입니다. 트리의 맨 위에는 스테이지가 있고 여기에는 여러 표시 객체를 포함할 수 있습니다. 자체가 표시 객체 컨테이너인 이러한 표시 객체는 다른 표시 객체나 표시 객체 컨테이너를 포함할 수 있습니다.



DisplayObjectContainer 클래스에는 표시 객체 컨테이너의 자식 목록을 사용하여 표시 목록을 순회하기 위한 속성과 메서드가 포함되어 있습니다. 예를 들어, 두 표시 객체 title과 pict를 container 객체(Sprite 객체, Sprite 클래스는 DisplayObjectContainer 클래스를 확장)에 추가하는 다음 코드를 살펴봅니다.

```
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);
```

getChildAt() 메서드는 특정 인덱스 위치에 있는 표시 목록의 자식을 반환합니다.

```
trace(container.getChildAt(0) is TextField); // true
```

이름을 기준으로 자식 객체에 액세스할 수도 있습니다. 각 표시 객체에는 name 속성이 있습니다. 이 속성을 할당하지 않을 경우 Flash Player 또는 AIR는 "instance1"과 같은 기본값을 할당합니다. 예를 들어, 다음 코드는 getChildByName() 메서드를 사용하여 이름이 "banana loader"인 자식 표시 객체에 액세스하는 방법을 보여 줍니다.

```
trace(container.getChildByName("banana loader") is Loader); // true
```

getChildByName() 메서드를 사용하면 getChildAt() 메서드를 사용할 때보다 속도가 느려질 수 있습니다.

표시 객체 컨테이너는 다른 표시 객체 컨테이너를 표시 목록에 자식 객체로 포함할 수 있기 때문에 응용 프로그램의 전체 표시 목록을 트리로 순회할 수 있습니다. 예를 들어, 앞에서 인용한 코드에서 pict Loader 객체의 로드 작업이 완료되면 pict 객체에 비트맵인 자식 표시 객체 하나가 로드됩니다. 이 비트맵 표시 객체에 액세스하려면 pict.getChildAt(0)을 입력합니다. 또한 container.getChildAt(0).getChildAt(0)(container.getChildAt(0) == pict이기 때문)를 입력할 수도 있습니다.

다음 함수는 표시 객체 컨테이너에서 표시 목록의 trace() 출력을 들여쓰기 형식으로 표시합니다.

```
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

Flex를 사용하는 경우 Flex가 다양한 구성 요소 표시 객체 클래스를 정의하며, 이러한 클래스는 DisplayObjectContainer 클래스의 표시 목록 액세스 메서드를 재정의한다는 것을 알아야 합니다. 예를 들어, mx.core 패키지의 Container 클래스는 Container 클래스가 확장하는 DisplayObjectContainer 클래스의 addChild() 메서드 및 기타 메서드를 재정의합니다. addChild() 메서드의 경우 이 클래스는 Flex의 Container 인스턴스에 모든 유형의 표시 객체를 추가할 수는 없도록 메서드를 재정의합니다. 이 경우 재정의된 메서드에서는 추가 중인 자식 객체가 mx.core.UIComponent 객체 유형이어야 합니다.

Stage 속성 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

Stage 클래스는 DisplayObject 클래스의 대부분의 속성과 메서드를 무시합니다. 이 무시되는 속성 또는 메서드 중 하나를 호출하면 Flash Player 및 AIR에서 예외가 발생합니다. 예를 들어, Stage 객체는 응용 프로그램에 대한 기본 컨테이너로 위치가 고정되기 때문에 x 또는 y 속성이 없습니다. x 및 y 속성은 컨테이너를 기준으로 표시 객체의 위치를 나타냅니다. 스테이지는 다른 표시 객체 컨테이너에 포함되지 않기 때문에 이러한 속성이 적용되지 않습니다.

참고: Stage 클래스의 일부 속성과 메서드는 로드된 첫 번째 SWF 파일과 동일한 보안 샌드박스에 있는 표시 객체에만 사용할 수 있습니다. 자세한 내용은 968페이지의 “스테이지 보안”을 참조하십시오.

재생 프레임 속도 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

Stage 클래스의 frameRate 속성은 응용 프로그램에 로드된 모든 SWF 파일의 프레임 속도를 설정하는 데 사용됩니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

스테이지 크기 조절 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

화면에서 Flash Player 또는 AIR를 나타내는 부분의 크기가 조절되면 런타임의 스테이지 내용도 그에 맞게 조절됩니다. Stage 클래스의 scaleMode 속성은 스테이지 내용 조절 방식을 결정합니다. 이 속성은 flash.display.StageScaleMode 클래스의 상수로 정의되는 서로 다른 네 가지 값으로 설정할 수 있습니다.

- StageScaleMode.EXACT_FIT 내용의 원래 종횡비에 상관없이 새 스테이지 크기에 맞게 SWF를 조절합니다. 폭과 높이의 비율 인수는 변경될 수도 있으므로 스테이지의 종횡비가 변경됨에 따라 내용의 모양은 눌려지거나 늘어날 수도 있습니다.
- StageScaleMode.SHOW_ALL 내용의 종횡비를 변경하지 않고 새 스테이지 크기 전체를 채우도록 SWF를 조절합니다. 이 크기 조절 모드를 사용하면 내용의 모든 부분을 표시할 수 있지만, 표준 TV에서 와이드스크린 영화를 볼 때 나타나는 검은색 테두리와 비슷한 "레터박스" 테두리가 나타날 수도 있습니다.
- StageScaleMode.NO_BORDER 내용의 종횡비를 변경하지 않고 새 스테이지 크기를 모두 채우도록 SWF를 조절합니다. 이 크기 조절 모드는 스테이지 표시 영역을 모두 활용하지만 잘림이 일어날 수도 있습니다.
- StageScaleMode.NO_SCALE SWF의 크기를 조절하지 않습니다. 새 스테이지 크기가 더 작을 경우 내용이 잘리며 더 클 경우에는 추가된 공간은 비게 됩니다.

  StageScaleMode.NO_SCALE 크기 조절 모드에서만 Stage 클래스의 stageWidth 및 stageHeight 속성을 사용하여 크기 조절된 스테이지의 실제 픽셀 크기를 지정할 수 있습니다. 다른 크기 조절 모드에서는 stageWidth 및 stageHeight 속성이 항상 SWF의 원래 폭 및 높이를 반영합니다. 또한 scaleMode가 StageScaleMode.NO_SCALE로 설정되고 SWF 파일의 크기가 조절된 경우 Stage 클래스의 resize 이벤트가 전달되어 이에 따라 조절할 수 있습니다.

  따라서 scaleMode를 StageScaleMode.NO_SCALE로 설정하면 원하는 경우 윈도우 크기 조절에 맞춰 화면 내용을 조절하는 방법을 보다 세밀하게 제어할 수 있습니다. 예를 들어, 비디오와 컨트롤 막대가 포함된 SWF에서 스테이지의 크기를 조절할 때 컨트롤 막대를 동일한 크기로 유지하고 스테이지 크기 변경에 맞춰 비디오 윈도우의 크기만 변경할 수 있습니다. 다음 예제에서 이를 확인할 수 있습니다.

```
// 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;
}
```

AIR 윈도우에 대한 스테이지 크기 조절 모드 설정

스테이지 scaleMode 속성은 윈도우 크기를 조정할 때 스테이지에서 자식 표시 객체를 크기 조정 및 클리핑하는 방식을 결정합니다. AIR에서는 noScale 모드만 사용해야 합니다. 이 모드에서는 스테이지가 크기 조절되지 않습니다. 대신 윈도우의 경계에 따라 스테이지의 크기가 직접적으로 변경됩니다. 윈도우 크기를 줄이면 객체가 클리핑될 수 있습니다.

스테이지 크기 조절 모드는 스테이지의 크기 또는 종횡비를 조절하지 못할 수도 있는 웹 브라우저와 같은 환경에서 사용하도록 설계되었습니다. 스테이지가 응용 프로그램의 이상적인 크기 또는 종횡비와 일치하지 않을 경우 이 모드를 통해 결과를 좀 더 개선할 수 있습니다. AIR에서는 항상 스테이지를 제어할 수 있으므로 대부분의 경우에 내용을 재배치하거나 윈도우의 크기를 조정하면 스테이지 크기 조절을 활용하는 것보다 더 좋은 결과를 얻을 수 있습니다.

브라우저 및 초기 AIR 윈도우에서는 윈도우 크기와 초기 비율 인수 사이의 관계가 로드된 SWF 파일로부터 읽혀집니다. 하지만 NativeWindow 객체를 만들 때는 AIR이 윈도우 크기와 비율 인수 72:1 사이의 임의의 관계를 선택합니다. 따라서 윈도우가 72x72 픽셀인 경우 윈도우에 10x10 사각형을 추가하면 10x10 픽셀의 올바른 크기로 그려집니다. 하지만 윈도우가 144x144 픽셀인 경우에는 10x10 픽셀 사각형이 20x20 픽셀로 크기가 조절됩니다. 그래도 윈도우 스테이지에 대해 noScale 대신 scaleMode를 사용한다면 윈도우의 모든 표시 객체에 대한 비율 인수를 스테이지의 현재 폭 및 높이에 대한 72 픽셀 비율로 설정하여 보정할 수 있습니다. 예를 들어 다음 코드에서는 client라는 표시 객체에 대한 필수 비율 인수를 계산합니다.

```
if(newWindow.stage.scaleMode != StageScaleMode.NO_SCALE){
client.scaleX = 72/newWindow.stage.stageWidth;
client.scaleY = 72/newWindow.stage.stageHeight;
}
```

참고: Flex 및 HTML 윈도우는 스테이지 scaleMode를 noScale로 자동으로 설정합니다. scaleMode를 변경하면 이러한 유형의 윈도우에서 사용되는 자동 레이아웃 메커니즘이 올바르게 작동하지 않습니다.

전체 화면 모드 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

전체 화면 모드를 사용하면 컨테이너 테두리 또는 메뉴를 사용하지 않고 뷰어의 전체 모니터를 채우도록 동영상의 스테이지를 설정할 수 있습니다. Stage 클래스의 displayState 속성은 SWF에 대한 전체 화면 모드를 설정 및 해제하는 데 사용됩니다. displayState 속성은 flash.display.StageDisplayState 클래스에 상수로 정의된 값 중 하나로 설정할 수 있습니다. 전체 화면 모드를 설정하려면 displayState 속성을 StageDisplayState.FULL_SCREEN으로 설정합니다.

```
stage.displayState = StageDisplayState.FULL_SCREEN;
```

전체 화면 상호 작용 모드(Flash Player 11.3의 새로운 기능)를 사용하려면 displayState 속성을 StageDisplayState.FULL_SCREEN_INTERACTIVE로 설정합니다.

```
stage.displayState = StageDisplayState.FULL_SCREEN_INTERACTIVE;
```

Flash Player에서 전체 화면 모드는 ActionScript에서 마우스 클릭(오른쪽 버튼 클릭 포함) 또는 키 입력을 통해서만 시작할 수 있습니다. 응용 프로그램 보안 샌드박스에서 실행 중인 AIR 내용의 경우 사용자 동작에 응답하여 전체 화면 모드를 시작할 필요가 없습니다.

전체 화면 모드를 종료하려면 displayState 속성을 StageDisplayState.NORMAL로 설정합니다.

```
stage.displayState = StageDisplayState.NORMAL;
```

또한 사용자는 포커스를 다른 윈도우로 전환하거나 Esc 키(모든 플랫폼), Ctrl+W(Windows), Command+W(Mac), Alt+F4(Windows) 등과 같은 여러 키 조합 중 하나를 사용하여 전체 화면 모드를 해제할 수 있습니다.

Flash Player에서 전체 화면 모드 활성화

HTML 페이지에 포함된 SWF 파일에 대해 전체 화면 모드를 활성화하려면, 다음과 같이 Flash Player를 포함할 HTML 코드는 이름이 allowFullScreen이고 값이 true인 param 태그와 embed 특성을 포함해야 합니다.

```
<object>
    ...
    <param name="allowFullScreen" value="true" />
    <embed ... allowFullScreen="true" />
</object>
```

Flash 제작 도구에서 [파일] > [제작 설정]을 선택하고 [제작 설정] 대화 상자의 [HTML] 탭에서 [Flash 전용 - 전체 화면 가능] 템플릿을 선택합니다.

Flex에서는 HTML 템플릿에 전체 화면을 지원하는 <object> 및 <embed> 태그가 들어 있는지 확인합니다.

웹 페이지에서 JavaScript를 사용하여 SWF 포함 태그를 생성할 경우 allowFullScreen param 태그 및 특성을 추가하도록 JavaScript를 변경해야 합니다. 예를 들어 HTML 페이지에서 AC_FL_RunContent() 함수(Flash Professional과 Flash Builder로 생성한 HTML 페이지 모두에 사용됨)를 사용하는 경우, 다음과 같이 allowFullScreen 매개 변수를 해당 함수 호출에 추가해야 합니다.

```
AC_FL_RunContent(
    ...
    'allowFullScreen','true',
    ...
    ); //end AC code
```

독립 실행형 Flash Player에서 실행되는 SWF 파일에는 적용되지 않습니다.

참고: 윈도우 모드(HTML에서 wmode)를 불투명 윈도우 없음(opaque) 또는 투명 윈도우 없음(transparent)으로 설정하면 전체 화면 윈도우는 항상 불투명하게 됩니다.

또한 브라우저에서 Flash Player로 전체 화면 모드를 사용하는 데는 보안 관련 제한 사항이 있습니다. 이러한 제한 사항은 948 페이지의 "보안"에 설명되어 있습니다.

Flash Player 11.3 이상에서 전체 화면 상호 작용 모드 사용

Flash Player 11.3 이상에서는 전체 화면 상호 작용 모드를 지원합니다. 이 모드에서는 모든 키보드 키(전체 화면 상호 작용 모드를 종료하는 **Esc** 키 제외)를 사용할 수 있습니다. 전체 화면 상호 작용 모드는 게임을 할 때 유용합니다(예: 멀티 플레이 게임에서 채팅을 사용하거나, FPS(First-Person Shooter) 게임에서 WASD 키보드 제어를 사용하는 경우).

HTML 페이지에 포함된 SWF 파일에 대해 전체 화면 상호 작용 모드를 활성화하려면 다음과 같이 Flash Player를 포함할 HTML 코드에 param 태그와 embed 특성(이름이 allowFullScreenInteractive이고 값이 true)이 포함되어야 합니다.

```
<object>
    ...
    <param name="allowFullScreenInteractive" value="true" />
    <embed ... allowFullScreenInteractive="true" />
</object>
```

Flash 제작 도구에서 [파일] > [제작 설정]을 선택하고 [제작 설정] 대화 상자의 [HTML] 탭에서 [Flash 전용 - 전체 화면 가능] 템플릿을 선택합니다.

Flash Builder와 Flex에서 HTML 템플릿에 전체 화면 상호 작용 모드를 지원하는 <object> 및 <embed> 태그가 포함되어야 합니다.

웹 페이지에서 JavaScript를 사용하여 SWF 포함 태그를 생성할 경우 allowFullScreenInteractive param 태그 및 특성을 추가하도록 JavaScript를 변경해야 합니다. 예를 들어 HTML 페이지에서 AC_FL_RunContent() 함수(Flash Professional과 Flash Builder에서 생성한 HTML 페이지에서 사용됨)를 사용하는 경우, 다음과 같이 allowFullScreenInteractive 매개 변수를 해당 함수 호출에 추가해야 합니다.

```
AC_FL_RunContent(
    ...
    'allowFullScreenInteractive','true',
    ...
    ); //end AC code
```

독립 실행형 Flash Player에서 실행되는 SWF 파일에는 적용되지 않습니다.

전체 화면 스테이지 크기 및 크기 조절

Stage.fullScreenHeight 및 Stage.fullScreenWidth 속성은 전체 화면 모드로 즉시 전환될 때 사용되는 모니터의 높이 및 너비를 반환합니다. 사용자가 이러한 값을 검색한 후 전체 화면 모드로 전환하기 전에 한 모니터에서 다른 모니터로 브라우저를 이동할 경우 해당 값이 잘못될 수 있습니다. Stage.displayState 속성을 StageDisplayState.FULL_SCREEN으로 설정한 이벤트 핸들러에서 이러한 값을 검색할 경우 올바른 값을 갖게 됩니다. 여러 모니터를 사용하는 사용자의 경우 SWF 내용을 확장하면 모니터 하나만 채워집니다. Flash Player 및 AIR에서는 메트릭을 사용하여 SWF의 가장 많은 부분이 포함된 모니터를 확인하고, 해당 모니터에 전체 화면 모드를 적용합니다. fullScreenHeight 및 fullScreenWidth 속성은 전체 화면 모드에 사용되는 모니터의 크기만 반영합니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 Stage.fullScreenHeight 및 Stage.fullScreenWidth를 참조하십시오.

전체 화면 모드에 대한 스테이지 크기 조절 비헤이비어는 일반 모드와 동일합니다. 즉, Stage 클래스의 scaleMode 속성에 의해 크기 조절이 제어됩니다. scaleMode 속성을 StageScaleMode.NO_SCALE로 설정하면 SWF에서 차지하는 화면의 크기(이 경우 전체 화면)에 따라 Stage의 stageWidth 및 stageHeight 속성이 변경됩니다. 브라우저에서 보는 경우 이 컨트롤의 HTML 매개 변수가 설정을 제어합니다.

Stage 클래스의 fullScreen 이벤트를 사용하여 전체 화면 모드가 설정 또는 해제될 때를 감지하여 응답할 수 있습니다. 예를 들어, 다음 예제처럼 전체 화면 모드를 시작하거나 해제할 때 화면에서 항목을 재배치하거나, 추가 또는 제거할 수 있습니다.

```
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);
```

이 코드에 표시된 것처럼 fullScreen 이벤트에 대한 이벤트 객체는 flash.events.FullScreenEvent 클래스의 인스턴스이며, 전체 화면 모드가 활성화되는지(true) 또는 비활성화되는지(false)를 나타내는 fullScreen 속성을 포함합니다.

전체 화면 모드에서 키보드 지원

브라우저에서 Flash Player가 실행될 때 키보드 이벤트 및 TextField 인스턴스의 텍스트 입력 등과 같은 모든 키보드 관련 ActionScript는 전체 화면 모드에서 비활성화됩니다. 다음과 같은 예외(키 활성화)가 있습니다.

- 인쇄되지 않는 키(예: 화살표 키, 스페이스바, Tab 키) 선택
- 전체 화면 모드를 종료하는 키보드 단축키: Esc(Windows 및 Mac), Ctrl-W(Windows), Command-W(Mac) 및 Alt-F4

SWF 내용을 독립 실행형 Flash Player 또는 AIR에서 실행 중인 경우에는 이러한 제한 사항이 없습니다. AIR에서는 키보드 입력을 허용하는 대화형 전체 화면 모드를 지원합니다.

전체 화면 모드의 마우스 지원

기본적으로 전체 화면 모드의 마우스 이벤트는 전체 화면 모드가 아닌 경우와 동일하게 작동합니다. 그러나 전체 화면 모드에서는 선택적으로 Stage.mouseLock 속성을 설정하여 마우스 잠금을 활성화할 수 있습니다. 마우스 잠금을 활성화하면 커서가 비활성화되고 마우스를 제한 없이 움직일 수 있습니다.

참고: 마우스 잠금은 데스크톱 응용 프로그램의 전체 화면 모드에서만 활성화할 수 있습니다. 전체 화면 모드가 아닌 응용 프로그램이나 휴대·장치의 응용 프로그램에서 설정하면 예외가 발생합니다.

다음과 같은 경우에는 마우스 잠금이 자동으로 비활성화되고 마우스 커서가 다시 표시됩니다.

- 사용자가 Esc 키(모든 플랫폼), Ctrl+W(Windows), Command+W(Mac) 또는 Alt+F4(Windows)를 사용하여 전체 화면 모드를 종료하는 경우
- 응용 프로그램 윈도우가 포커스를 잃는 경우
- [개인 정보] 대화 상자를 포함한 모든 설정 UI가 표시되는 경우
- [파일 업로드] 대화 상자 같은 기본 대화 상자가 표시되는 경우

마우스 이동과 관련된 이벤트(예: mouseMove 이벤트)는 MouseEvent 클래스를 사용하여 이벤트 객체를 나타냅니다. 마우스 잠금이 비활성화된 경우 MouseEvent.localX 및 MouseEvent.localY 속성을 사용하여 마우스 위치를 확인합니다. 마우스 잠금이 활성화된 경우에는 MouseEvent.movementX 및 MouseEvent.movementY 속성을 사용하여 마우스 위치를 확인합니다. movementX 및 movementY 속성에는 마우스 위치의 절대 좌표 대신 마지막 이벤트 이후 마우스 위치의 변경 사항이 포함됩니다.

하드웨어를 통해 전체 화면 모드로 크기 조절

Stage 클래스의 fullScreenSourceRect 속성을 사용하여 Flash Player 또는 AIR에서 스테이지의 특정 영역이 전체 화면 모드로 확대되도록 설정합니다. Flash Player 및 AIR는 하드웨어를 통해 크기를 조절합니다. 이 경우 사용자의 컴퓨터에 장착된 그래픽 및 비디오 카드를 사용하며 일반적으로 소프트웨어로 크기를 조절할 때보다 더욱 빠르게 내용을 표시합니다.

하드웨어적인 크기 조절을 활용하려면 모든 스테이지나 스테이지의 일부를 전체 화면 모드로 설정합니다. 다음 ActionScript 3.0 코드는 모든 스테이지를 전체 화면 모드로 설정합니다.

```
import flash.geom.*;
{
    stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
    stage.displayState = StageDisplayState.FULL_SCREEN;
}
```

이 속성이 유효한 사각형으로 설정되고 displayState 속성이 전체 화면 모드로 설정된 경우 Flash Player 및 AIR가 지정된 영역의 크기를 조절합니다. ActionScript 내에서 픽셀 단위의 실제 스테이지 크기는 변경되지 않습니다. Flash Player 및 AIR는 "전체 화면 모드를 종료하려면 Esc를 누르십시오"라는 표준 메시지를 수용하도록 사각형의 크기에 대해 최소 제한을 적용합니다. 이 제한은 대개 약 260 x 30 픽셀이지만 플랫폼과 Flash Player 버전에 따라 다를 수 있습니다.

fullScreenSourceRect 속성은 Flash Player 또는 AIR가 전체 화면 모드가 아닌 경우에만 설정할 수 있습니다. 이 속성을 정확히 사용하려면 먼저 이 속성을 설정한 다음 displayState 속성을 전체 화면 모드로 설정합니다.

크기 조절을 활성화하려면 fullScreenSourceRect 속성을 rectangle 객체로 설정합니다.

```
stage.fullScreenSourceRect = new Rectangle(0,0,320,240);
```

크기 조절을 비활성화하려면 fullScreenSourceRect 속성을 null로 설정합니다.

```
stage.fullScreenSourceRect = null;
```

Flash Player의 모든 하드웨어 가속 기능을 사용하려면 [Flash Player 설정] 대화 상자를 통해 활성화합니다. 브라우저에서 Flash Player 내용을 마우스 오른쪽 버튼으로 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Mac)하여 대화 상자를 로드합니다. 첫 번째 탭인 [표시] 탭을 선택하고 [하드웨어 가속 사용] 체크 상자를 클릭합니다.

직접 및 GPU 합성 윈도우 모드

Flash Player 10에는 직접 및 GPU 합성 모드라는 두 가지 윈도우 모드가 새로 추가되었습니다. 이러한 모드는 Flash 제작 도구의 제작 설정을 통해 활성화할 수 있습니다. AIR에서는 이러한 모드가 지원되지 않습니다. 이러한 모드를 사용하려면 Flash Player의 하드웨어 가속을 사용하도록 설정해야 합니다.

직접 모드는 가장 빠르고 직접적인 경로를 사용하여 화면에 그래픽을 표시하므로 비디오 재생에 적합합니다.

GPU 합성은 비디오 카드의 GPU(Graphics Processing Unit)를 사용하여 합성 속도를 빠르게 합니다. 비디오 합성은 단일 비디오 이미지를 만들기 위해 여러 개의 이미지를 겹치는 프로세스입니다. GPU로 합성을 빠르게 할 경우 YUV 변환, 색상 교정, 회전 또는 크기 조절, 블렌딩의 성능을 향상시킬 수 있습니다. YUV 변환은 전송에 사용되는 합성 아날로그 신호를 비디오 카메라 및 디스플레이에 사용되는 RGB(빨강, 녹색, 파랑) 색상 모델로 색상 변환하는 것을 나타냅니다. GPU를 사용하여 합성을 빠르게 하면 CPU에 대한 메모리 및 계산 요청을 줄일 수 있습니다. 또한 표준 정의 비디오를 보다 부드럽게 재생할 수 있습니다.

이러한 윈도우 모드를 구현할 때는 주의해야 합니다. GPU 합성을 사용할 경우 메모리 및 CPU 리소스가 많이 소모될 수 있습니다. 블렌드 모드, 필터링, 자르기, 마스크 적용 등의 일부 작업은 GPU에서 수행할 수 없는 경우 소프트웨어에서 수행됩니다. 이러한 모드를 사용하고 있으며 배너에는 이러한 모드를 사용하지 않아야 하는 경우 HTML 페이지당 하나의 SWF 파일을 사용하도록 제한하는 것이 좋습니다. Flash 동영상 테스트 기능에는 하드웨어 가속이 사용되지 않지만 [제작 미리 보기] 옵션을 통해 하드웨어 가속을 사용할 수 있습니다.

SWF 파일의 프레임 속도를 60보다 높게 설정할 경우 최대 화면 새로 고침 속도는 쓸모가 없습니다. 프레임 속도를 50에서 55 사이로 설정하면 이따금 다양한 이유로 발생할 수 있는 프레임 삭제가 허용됩니다.

직접 모드를 사용하려면 Microsoft DirectX 9와 VRAM 128MB(Windows의 경우) 및 OpenGL(Apple Macintosh, Mac OS X v10.2 이상의 경우)이 필요합니다. GPU 합성을 사용하려면 128MB의 VRAM이 장착된 Windows에서 Microsoft DirectX 9 및 Pixel Shader 2.0이 지원되어야 합니다. Mac OS X 및 Linux에서 GPU 합성을 사용하려면 OpenGL 1.5 및 여러 OpenGL 확장(예: framebuffer 객체, multitexture, shader 객체, 세이딩 언어, 부분 세이더)이 필요합니다.

Flash [제작 설정] 대화 상자의 [Flash] 탭에 있는 [하드웨어 가속] 메뉴를 통해 SWF별로 직접 및 GPU 가속 모드를 활성화할 수 있습니다. [없음]을 선택한 경우 윈도우 모드는 [HTML] 탭의 [윈도우 모드] 설정에 지정된 대로 default, transparent 또는 opaque로 돌아갑니다.

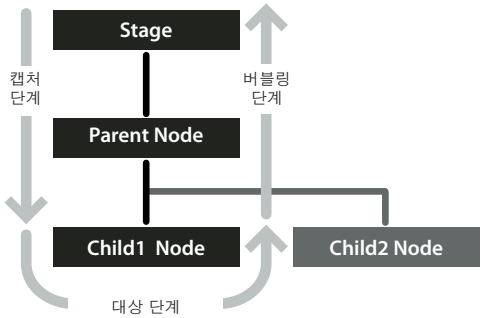
표시 객체에 대한 이벤트 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

DisplayObject 클래스는 EventDispatcher 클래스에서 상속됩니다. 즉, 모든 표시 객체는 이벤트 모델에 모두 사용할 수 있습니다(113페이지의 "이벤트 처리" 참조). 모든 표시 객체는 EventDispatcher 클래스에서 상속되는 addEventListener() 메서드를 사용하여 특정 이벤트를 수신할 수 있습니다. 이때 수신 객체가 해당 이벤트에 대한 이벤트 흐름의 일부여야 합니다.

Flash Player 또는 AIR에서 이벤트 객체를 전달할 경우 해당 이벤트 객체는 스테이지부터 이벤트가 발생한 표시 객체까지를 왕복합니다. 예를 들어, child1이라는 표시 객체를 클릭하면 Flash Player는 스테이지에서 표시 목록 계층 아래의 child1 표시 객체까지 이벤트 객체를 전달합니다.

다음 다이어그램에 표시된 것처럼 이벤트 흐름은 개념적으로 3단계로 구분됩니다.



자세한 내용은 113페이지의 "이벤트 처리"를 참조하십시오.

표시 객체 이벤트 작업을 할 경우 주의해야 할 중요 사항은, 표시 객체를 표시 목록에서 제거할 때 표시 객체가 메모리에서 자동으로 제거(가비지 컬렉션)되는지 여부에 대해 이벤트 리스너가 미치는 영향입니다. 표시 객체에 이벤트에 대한 리스너로 구독되는 객체가 있는 경우 해당 표시 객체는 표시 목록에서 제거되더라도 해당 리스너 객체에 대한 참조가 여전히 존재하기 때문에 메모리에서 제거되지 않습니다. 자세한 내용은 124페이지의 "이벤트 리스너 관리"를 참조하십시오.

DisplayObject 하위 클래스 선택

Flash Player 9 이상, Adobe AIR 1.0 이상

선택할 다양한 옵션이 있는 표시 객체로 작업할 경우 사용할 표시 객체와 용도를 결정해야 합니다. 다음과 같이 결정에 도움이 되는 몇 가지 지침이 있습니다. 클래스의 인스턴스가 필요한 경우나 만들 클래스에 대한 기본 클래스를 선택하는 경우에도 동일한 제안 사항이 적용됩니다.

- 다른 표시 객체의 컨테이너로 사용할 객체가 필요하지 않는 경우(즉, 독립 실행형 화면 요소 역할을 하는 객체만 필요한 경우) 용도에 따라 DisplayObject 또는 InteractiveObject 하위 클래스 중 하나를 선택합니다.
  - 비트맵 이미지 표시를 위한 Bitmap

  - 텍스트 추가를 위한 TextField
  - 비디오 표시를 위한 Video
  - 화면 상에 내용을 그리기 위한 "캔바스" Shape. 특히, 화면에서 모양을 그리기 위한 인스턴스를 만들 경우 해당 인스턴스가 다른 표시 객체의 컨테이너가 아니라면, Sprite 또는 MovieClip 대신 Shape를 사용하면 성능이 크게 향상됩니다.
  - Flash 제작 도구로 만든 항목에 대한 MorphShape, StaticText 또는 SimpleButton. 이러한 클래스의 인스턴스를 프로그래밍 방식으로 만들 수는 없지만, Flash 제작 도구를 사용하여 만든 항목을 나타내도록 이러한 데이터 유형으로 변수를 만들 수 있습니다.
- 변수가 기본 스테이지를 나타내야 하는 경우 Stage 클래스를 데이터 유형으로 사용합니다.
- 외부 SWF 파일 또는 이미지 파일을 로드하기 위한 컨테이너가 필요한 경우 Loader 인스턴스를 사용합니다. 로드된 내용은 표시 목록에 Loader 인스턴스의 자식으로 추가됩니다. 데이터 유형은 로드된 내용의 특성에 따라 다음과 같이 달라집니다.
  - 로드된 이미지는 Bitmap 인스턴스입니다.
  - ActionScript 3.0에서 작성되어 로드된 SWF 파일은 Sprite 또는 MovieClip 인스턴스이거나 내용 작성자에 의해 지정된 해당 클래스의 하위 클래스 인스턴스가 됩니다.
  - ActionScript 1.0 또는 ActionScript 2.0에서 작성되어 로드된 SWF 파일은 AVM1Movie 인스턴스가 됩니다.
- ActionScript를 사용하여 표시 객체에 그릴지 여부에 관계없이 다른 표시 객체에 대한 컨테이너 역할을 할 객체가 필요한 경우 DisplayObjectContainer 하위 클래스 중 하나를 선택합니다.
  - ActionScript만 사용하여 객체를 만들거나 ActionScript만 사용하여 만들고 조작할 사용자 정의 표시 객체에 대한 기본 클래스로 객체를 만드는 경우 Sprite를 선택합니다.
  - Flash 제작 도구에서 만든 동영상 클립 심볼을 가리키는 변수를 만들 경우 MovieClip을 선택합니다.
- Flash 라이브러리에서 동영상 클립 심볼과 연결될 클래스를 만들 경우 DisplayObjectContainer 하위 클래스 중 하나를 클래스의 기본 클래스로 선택합니다.
  - 관련된 동영상 클립 심볼의 여러 프레임에 내용이 있는 경우 MovieClip을 선택합니다.
  - 관련된 동영상 클립 심볼의 첫 번째 프레임에만 내용이 있는 경우 Sprite를 선택합니다.

표시 객체 조작

Flash Player 9 이상, Adobe AIR 1.0 이상

사용하려고 선택한 표시 객체에 관계없이 모든 표시 객체에는 화면에 표시되는 요소로서의 공통적인 다양한 조작 방법이 있습니다. 예를 들어, 모든 표시 객체를 화면에 배치한 다음 표시 객체가 쌓이는 순서를 앞 또는 뒤로 이동하거나, 표시 객체의 크기를 조절하거나 회전할 수 있습니다. 모든 표시 객체는 공통 기본 클래스(DisplayObject)로부터 이 기능을 상속하기 때문에 TextField 인스턴스, Video 인스턴스, Shape 인스턴스 또는 다른 표시 객체를 조작하든 관계없이 이 기능은 동일하게 동작합니다. 다음 단원에서는 이러한 공통 표시 객체 조작 방법 중 몇 가지를 자세히 설명합니다.

위치 변경

Flash Player 9 이상, Adobe AIR 1.0 이상

화면에 표시 객체를 배치하는 것은 표시 객체에 대한 가장 기본적인 조작입니다. 표시 객체의 위치를 설정하려면 객체의 x 및 y 속성을 변경합니다.

```
myShape.x = 17;
myShape.y = 212;
```

표시 객체 위치 지정 시스템에서는 스테이지를 직교 좌표계(가로 x축과 세로 y축이 있는 일반 격자 시스템)로 취급합니다. 좌표계의 원점(x축과 y축이 만나는 0,0 좌표)은 스테이지의 왼쪽 위 모서리에 있습니다. x 값은 원점으로부터 오른쪽이 양수이고 왼쪽이 음수이지만, y 값은 아래쪽이 양수이고 위쪽이 음수로 일반 그래프 시스템과 반대입니다. 예를 들어, 이전 코드 행은 myShape 객체를 x 좌표 17(원점의 오른쪽으로 17 픽셀) 및 y 좌표 212(원점의 아래로 212 픽셀)로 이동합니다.

기본적으로 ActionScript를 사용하여 표시 객체를 만들 경우 x 속성과 y 속성이 모두 0으로 설정되어 객체가 부모 내용의 왼쪽 위 모서리에 배치됩니다.

스테이지를 기준으로 위치 변경

x 및 y 속성은 항상 부모 표시 객체 축의 0,0 좌표를 기준으로 표시 객체의 위치를 나타냅니다. Sprite 인스턴스에 포함된 Shape 인스턴스(예: circle)에 대해 Shape 객체의 x 및 y 속성을 0으로 설정하면 원이 Sprite의 왼쪽 위 모서리에 배치됩니다. 이 위치가 반드시 스테이지의 왼쪽 위 모서리가 되는 것은 아닙니다. 전역 스테이지 좌표를 기준으로 객체를 배치하려면 표시 객체의 globalToLocal() 메서드를 사용하여 좌표를 전역(스테이지) 좌표에서 로컬(표시 객체 컨테이너) 좌표로 다음과 같이 변환할 수 있습니다.

```
// 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;
```

마찬가지로 DisplayObject 클래스의 localToGlobal() 메서드를 사용하여 로컬 좌표를 스테이지 좌표로 변환할 수 있습니다.

마우스로 표시 객체 이동

ActionScript에서는 두 가지 다른 기법을 사용하여 사용자가 마우스로 표시 객체를 이동할 수 있도록 설정할 수 있습니다. 두 방법 모두 두 마우스 이벤트가 사용됩니다. 즉, 마우스 버튼을 누르면 마우스 커서를 따라 이동하도록 객체에게 지시하고, 마우스 버튼을 놓으면 마우스 커서를 따라 이동하는 것을 중지하도록 객체에게 지시합니다.

참고: Flash Player 11.3 이상 및 AIR 3.3 이상: MouseEvent.RELEASE_OUTSIDE 이벤트를 사용하여 사용자가 포함 Sprite 경계 외부에서 마우스 버튼을 놓는 경우에 대응할 수도 있습니다.

startDrag() 메서드를 사용하는 첫 번째 기법은 더 간단하지만 보다 제한적입니다. 마우스 버튼을 누르면 드래그할 표시 객체의 startDrag() 메서드가 호출됩니다. 마우스 버튼을 놓으면 stopDrag() 메서드가 호출됩니다. Sprite 클래스는 이러한 두 함수를 정의하므로 이동된 객체는 Sprite이거나 이 클래스의 하위 클래스여야 합니다.

```
// 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);
```

이 기술은 startDrag()를 사용하여 한 번에 하나의 항목만 드래그할 수 있다는 한 가지 중요한 제한이 있습니다. 표시 객체를 드래그하는 중에 startDrag() 메서드가 다른 표시 객체에서 호출되면, 마우스를 따라 이동하던 첫 번째 표시 객체의 이동이 즉시 중지됩니다. 예를 들어, startDragging() 함수가 아래에 표시된 것처럼 변경되면 square.startDrag() 메서드를 호출하더라도 circle 객체만 드래그됩니다.

```
function startDragging(event:MouseEvent):void
{
    square.startDrag();
    circle.startDrag();
}
```

startDrag()를 사용하여 객체를 한 번에 하나씩만 드래그할 수 있기 때문에, 임의의 표시 객체에 대해 stopDrag() 메서드를 호출하여 현재 드래그 중인 객체의 이동을 중지시킬 수 있습니다.

여러 표시 객체를 드래그해야 하거나 여러 객체가 startDrag()를 사용하여 발생할 수 있는 충돌을 방지하려면 mouse-following 기술을 사용하여 드래그 효과를 만드는 것이 좋습니다. 이 기술을 사용할 경우 마우스 버튼을 누르면 함수가 스테이지의 mouseMove 이벤트에 대한 리스너로 구독됩니다. 그러면 마우스를 이동할 때마다 이 함수가 호출되어 드래그되는 객체가 마우스의 x, y 좌표로 바로 이동됩니다. 마우스 버튼을 놓으면 함수가 구독 해제되어, 마우스를 이동해도 함수가 더 이상 호출되지 않으므로 객체가 마우스 커서를 따라 이동하지 않습니다. 다음은 이 기술을 설명하는 일부 코드입니다.

```
// 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);
```

표시 객체를 마우스 커서를 따라 이동하게 하는 것 외에도, 드래그된 객체를 표시의 전면으로 이동시켜 모든 다른 객체 위에 떠 있는 것처럼 보이게 해야 할 수 있습니다. 예를 들어 원과 정사각형의 두 객체가 있고 두 객체 모두 마우스로 이동할 수 있다고 가정해보십시오. 원이 표시 목록에서 사각형의 아래에 있을 때 원을 클릭한 상태에서 드래그하여 커서를 사각형 위에 놓으면, 원이 사각형 뒤로 서서히 이동하는 것처럼 표시되어 드래그 앤 드롭 효과가 제거됩니다. 대신 원을 클릭하면 원이 표시 목록의 맨 위로 이동하여 항상 다른 내용의 위에 표시되게 할 수 있습니다.

앞의 예제에서 가져온 다음 코드에서는 원과 정사각형인 두 표시 객체를 마우스로 이동할 수 있도록 합니다. 어느 한 객체를 마우스 버튼으로 누르면 해당 항목이 스테이지 표시 목록의 위로 이동하여 드래그된 항목이 항상 위에 표시됩니다. 새 코드나 이전 샘플에서 변경된 코드는 굵은 글꼴로 표시됩니다.

```
// 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);
```

토큰이나 카드가 더미 간에 이동하는 게임에서처럼 이 효과를 더 확장하려면, 객체를 "선택"하여 드래그된 객체를 스테이지의 표시 목록에 추가한 다음, 마우스 버튼을 놓아 해당 객체를 다른 표시 목록(예: 드롭되는 "더미")에 추가할 수 있습니다.

마지막으로 효과를 향상시키기 위해 클릭할 때(드래그를 시작할 때) 표시 객체에 그림자 필터를 적용하고 객체를 놓을 때 그림자를 제거할 수 있습니다. ActionScript에서 그림자 필터 및 기타 표시 객체 필터를 사용하는 방법에 대한 자세한 내용은 242페이지의 “표시 객체 필터링”을 참조하십시오.

표시 객체 패닝 및 스크롤

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체가 표시하려는 영역에 비해 너무 큰 경우 scrollRect 속성을 사용하여 표시 객체의 표시 가능한 영역을 정의할 수 있습니다. 또한 사용자 입력에 따라 scrollRect 속성을 변경하여 내용을 왼쪽/오른쪽으로 패닝하거나 위/아래로 스크롤할 수 있습니다.

scrollRect 속성은 사각형 영역을 단일 객체로 정의하는 데 필요한 값을 결합하는 클래스인 Rectangle 클래스의 인스턴스입니다. 표시 객체의 표시 가능한 영역을 처음으로 정의하려면 새 Rectangle 인스턴스를 만들어 표시 객체의 scrollRect 속성에 할당합니다. 나중에 스크롤 또는 패닝하려면 scrollRect 속성을 개별 Rectangle 변수로 읽고 원하는 속성을 변경합니다. 예를 들어, Rectangle 인스턴스의 x 속성을 변경하여 패닝하거나 y 속성을 변경하여 스크롤합니다. 그런 다음 Rectangle 인스턴스를 scrollRect 속성에 다시 할당하여 표시 객체에 변경된 값을 알립니다.

예를 들어, 다음 코드에서는 너무 커서 SWF 파일의 경계 안에 들어가지 않는 bigText라는 TextField 객체에 대한 표시 가능한 영역을 정의합니다. 두 버튼 up 및 down을 클릭하면 scrollRect Rectangle 인스턴스의 y 속성을 수정하여 TextField 객체의 내용을 위 또는 아래로 스크롤하는 함수가 호출됩니다.

```
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);
```

이 예제에서 보듯이 표시 객체의 scrollRect 속성으로 작업할 경우 cacheAsBitmap 속성을 사용하여 Flash Player 또는 AIR에서 표시 객체의 내용을 비트맵으로 캐시하도록 지정하는 것이 좋습니다. 그렇게 하면 표시 객체의 내용을 스크롤할 때마다 Flash Player 또는 AIR에서 표시 객체의 전체 내용을 다시 그릴 필요가 없을 뿐만 아니라, 캐시된 비트맵을 사용하여 필요한 부분을 화면에 직접 렌더링할 수 있습니다. 자세한 내용은 164페이지의 "표시 객체 캐싱"을 참조하십시오.

객체 크기 조작 및 크기 조절

Flash Player 9 이상, Adobe AIR 1.0 이상

크기 속성(width 및 height) 또는 크기 조절 속성(scaleX 및 scaleY)을 사용하여 표시 객체의 크기를 두 가지 방식으로 측정하고 조작할 수 있습니다.

모든 표시 객체에는 width 속성과 height 속성이 있습니다. 이러한 속성은 처음에는 객체의 크기(픽셀)로 설정됩니다. 이러한 속성의 값을 읽어서 표시 객체의 크기를 측정할 수 있습니다. 다음과 같이 새 값을 지정하여 객체의 크기를 변경할 수도 있습니다.

```
// Resize a display object.
square.width = 420;
square.height = 420;

// Determine the radius of a circle display object.
var radius:Number = circle.width / 2;
```

표시 객체의 height 또는 width를 변경하면 객체의 크기가 조절됩니다. 즉, 객체의 내용이 새 영역에 맞게 확대되거나 축소됩니다. 표시 객체에 벡터 모양만 포함되어 있는 경우 품질 손상 없이 해당 모양이 새 크기로 다시 그려집니다. 표시 객체에 있는 비트맵 그래픽 요소는 다시 그려지지 않고 크기가 조절됩니다. 예를 들어, 폭과 높이가 이미지의 실제 픽셀 정보 크기보다 크게 확대되는 디지털 사진은 픽셀화되어 들쭉날쭉하게 보입니다.

표시 객체의 width 또는 height 속성을 변경하면 Flash Player 및 AIR에서 객체의 scaleX 및 scaleY 속성도 함께 업데이트합니다.

참고: TextField 객체에는 이러한 크기 조절 비헤이비어가 적용되지 않습니다. 텍스트 필드는 텍스트 줄 바꿈과 글꼴 크기에 맞게 자체적으로 크기 조절되어야 하므로 크기 조절 후 scaleX 또는 scaleY 값이 1로 재설정됩니다. 그러나 TextField 객체의 scaleX 또는 scaleY 값을 조정할 경우 width 및 height 값은 사용자가 지정한 크기 조절 값에 맞게 변경됩니다.

이러한 속성은 원본 크기와 비교한 표시 객체의 상대적인 크기를 나타냅니다. scaleX 및 scaleY 속성은 분수(소수) 값을 사용하여 백분율을 나타냅니다. 예를 들어, 표시 객체의 width 속성이 원본 폭의 1/2 크기로 변경된 경우 객체의 scaleX 속성 값은 .5(50%)가 됩니다. 높이가 두 배로 커지면 scaleY 속성 값이 2(200%)가 됩니다.

```
// 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
```

크기는 비례적으로 변경되지 않습니다. 즉, 정사각형의 height 속성만 변경하고 width 속성은 변경하지 않을 경우 가로와 세로의 비율이 달라져서 정사각형이 아니라 직사각형이 됩니다. 표시 객체의 크기를 상대적으로 변경하려면 width 또는 height 속성 대신 scaleX 및 scaleY 속성 값을 설정하여 객체의 크기를 조절할 수 있습니다. 예를 들어, 다음 코드는 square라는 표시 객체의 width 속성을 변경한 다음 세로 비율(scaleY)을 가로 비율과 일치하도록 변경하여 정사각형의 가로 세로 비율을 유지합니다.

```
// 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;
```

크기 조절 시 왜곡 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

일반적으로 표시 객체의 크기를 조절하면(예: 가로로 확대) 각 부분이 동일한 양만큼 확장되어 객체 전체에서 균일하게 왜곡이 발생합니다. 그래픽 및 디자인 요소의 경우 이런 결과를 원할 수도 있습니다. 그러나, 표시 객체의 확장되는 부분과 변경되지 않는 부분을 제어해야 할 경우도 있습니다. 그러한 일반적인 예로 모서리가 둥근 사각형 버튼이 있습니다. 일반 크기 조절에서는 버튼의 모서리가 확장되어 버튼 크기를 변경하면 모서리 반경도 변경됩니다.

그러나, 이 경우에는 왜곡 현상 없이 크기가 조절되도록 크기 조절을 제어해야 합니다. 즉, 크기를 조절할 영역(직선 변 및 중간)과 크기를 조절하지 않을 영역(모서리)을 지정합니다.

9-슬라이스 크기 조절(Scale-9)을 사용하여 객체 크기 조절 방법을 제어할 표시 객체를 만들 수 있습니다. 9-슬라이스 크기 조절을 사용하면 표시 객체가 9개의 개별 사각형(3 x 3 격자, tic-tac-toe 보드의 격자와 비슷함)으로 나뉘어집니다. 각 사각형의 크기가 반드시 동일할 필요는 없습니다. 사용자가 격자선의 위치를 지정합니다. 버튼의 둥근 모서리처럼 네 모서리의 사각형에 놓이는 내용은 표시 객체의 크기를 조절해도 확장되거나 축소되지 않습니다. 위쪽 가운데 사각형과 아래쪽 가운데 사각형은 가로로 크기가 조절되고 세로는 변경되지 않는 반면에, 왼쪽 중앙 및 오른쪽 중앙 사각형은 세로로 크기가 조절되고 가로는 변경되지 않습니다. 가운데 사각형은 가로와 세로 방향 모두 크기가 조절됩니다.

표시 객체를 만들 때 특정 내용의 크기가 조절되지 않게 하려면 해당 내용이 모서리 사각형 중 하나에서 끝나도록 9-슬라이스 크기 조절 격자의 구분선을 배치해야 합니다.

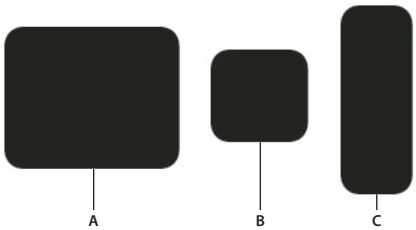
ActionScript에서 표시 객체의 scale9Grid 속성 값을 설정하여 객체에 대한 9-슬라이스 크기 조절을 지정하고 객체의 Scale-9 격자로 사각형 크기를 정의합니다. 다음과 같이 Rectangle 클래스의 인스턴스를 scale9Grid 속성 값으로 사용할 수 있습니다.

```
myButton.scale9Grid = new Rectangle(32, 27, 71, 64);
```

Rectangle 생성자의 네 매개 변수는 x 좌표, y 좌표, 폭 및 높이입니다. 이 예제에서 사각형의 왼쪽 위 모서리는 myButton 표시 객체에서 x: 32, y: 27 지점에 위치합니다. 사각형은 폭이 71픽셀이고 높이가 64픽셀입니다. 따라서 사각형의 오른쪽 가장자리는 표시 객체의 x 좌표 103 위치에 있고 아래쪽 가장자리는 표시 객체의 y 좌표 92 위치에 있습니다.

Rectangle 인스턴스에 의해 정의된 영역에 포함된 실제 영역은 Scale-9 격자의 가운데 사각형을 나타냅니다. 다음 그림에 표시된 것처럼 Flash Player 및 AIR에서 Rectangle 인스턴스의 변을 확장하여 다른 사각형을 계산합니다.

이 경우 버튼 크기를 확대하거나 축소하면 둥근 모서리는 확장되거나 축소되지 않지만, 다른 영역은 해당 크기 조절에 따라 조절됩니다.



A. myButton.width = 131;myButton.height = 106; **B.** myButton.width = 73;myButton.height = 69; **C.** myButton.width = 54;myButton.height = 141;

표시 객체 캐싱

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램을 작성하든 스크립팅된 복잡한 애니메이션을 작성하든 Flash에서 작업하는 디자인은 규모가 커질 수 있으므로, 성능 및 최적화를 고려해야 합니다. 사각형 Shape 인스턴스와 같이 정적으로 유지되는 내용은 Flash Player와 AIR에서 자동으로 최적화되지 않습니다. 따라서 사각형의 위치를 변경할 경우 Flash Player나 AIR에서는 전체 Shape 인스턴스가 다시 그려집니다.

지정된 표시 객체를 캐시하여 SWF 파일의 성능을 향상시킬 수 있습니다. 표시 객체는 기본적으로 인스턴스 벡터 데이터의 비트맵 버전인 표면이며, 이것은 SWF 파일을 만드는 과정에서 자주 변경하지 않는 데이터입니다. 그러므로 캐싱이 활성화된 인스턴스는 SWF 파일이 재생될 때 계속적으로 다시 그려지지 않습니다. 따라서 SWF 파일은 빠르게 렌더링됩니다.

참고: 벡터 데이터는 표면이 다시 만들어질 때 업데이트할 수 있습니다. 그러므로 표면에 캐싱된 벡터 데이터는 전체 SWF 파일에서 동일하게 유지될 필요가 없습니다.

표시 객체의 cacheAsBitmap 속성을 true로 설정하면 표시 객체가 자체 비트맵 표현을 캐시합니다. Flash Player나 AIR에서는 인스턴스에 대해 표면 객체가 만들어지며 이 객체는 벡터 데이터 대신에 비트맵으로 캐시된 것입니다. 표시 객체의 경계를 변경하면 표면은 크기가 조절되지 않고 다시 만들어집니다. 표면은 서로 다른 표면 안에 중첩이 가능합니다. 이 경우, 자식 표면은 부모 표면으로 비트맵을 복제하게 됩니다. 자세한 내용은 166페이지의 “비트맵 캐싱 활성화”를 참조하십시오.

DisplayObject 클래스의 opaqueBackground 및 scrollRect 속성은 cacheAsBitmap 속성을 사용하여 비트맵 캐싱과 연관됩니다. 위 세 가지 속성은 서로 독립적이지만, opaqueBackground 및 scrollRect 속성은 객체가 비트맵으로 캐시되는 경우 가장 잘 작동합니다. 즉, opaqueBackground 및 scrollRect 속성은 cacheAsBitmap이 true로 설정된 경우에만 성능상의 이점을 제공합니다. 표시 객체 내용 스크롤에 대한 자세한 내용은 161페이지의 "표시 객체 패닝 및 스크롤"을 참조하십시오. 불투명 배경 설정에 대한 자세한 내용은 167페이지의 "불투명한 배경색 설정"을 참조하십시오.

알파 채널 마스크를 사용하려면 cacheAsBitmap 속성을 true로 설정해야 합니다. 자세한 내용은 171페이지의 "표시 객체 마스크"를 참조하십시오.

캐싱 기능이 필요한 경우

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체에 캐싱을 활성화하면 표면이 만들어지며, 복잡한 벡터 애니메이션을 빠르게 렌더링할 수 있는 등 여러 이점이 있습니다. 몇몇 상황에서는 캐싱을 활성화하는 것이 좋습니다. SWF 파일의 성능을 향상하려면 항상 캐싱을 활성화하는 것이 좋을 것처럼 보이지만, 캐싱을 활성화해도 성능이 향상되지 않거나 오히려 저하되는 경우도 있습니다. 이 단원에서는 캐싱을 사용해야 할 경우와 일반 표시 객체를 사용해야 할 경우에 대해 설명합니다.

캐시된 데이터의 전체적인 성능은 인스턴스 벡터 데이터의 복잡도, 변경하려는 데이터의 양, opaqueBackground 속성의 설정 여부 등에 달려 있습니다. 작은 영역을 변경할 경우 표면과 벡터 데이터 중 어느 것을 사용해도 결과는 비슷합니다. 응용 프로그램을 배포하기 전에 두 가지 경우를 모두 테스트하는 것도 좋습니다.

비트맵 캐싱 기능이 필요한 경우

다음과 같은 경우에는 일반적으로 비트맵 캐싱을 사용할 경우 큰 이점을 얻을 수 있습니다.

- 복잡한 배경 이미지: 섬세하고 복잡한 벡터 데이터 배경 이미지가 포함된 응용 프로그램을 사용할 경우로, 비트맵 추적 명령을 적용한 이미지 또는 Adobe Illustrator®에서 작성한 아트웍 작업 등입니다. 배경에 있는 문자들에 애니메이션 효과를 적용할 수도 있으며, 그 결과 배경에서 벡터 데이터를 지속적으로 다시 생성해야 하므로 해당 애니메이션의 속도는 느려집니다. 이때 성능을 향상시키려면 배경 표시 객체의 opaqueBackground 속성을 true로 설정합니다. 배경이 비트맵으로 렌더링되고 빠르게 다시 그려지므로 애니메이션은 훨씬 더 빠르게 재생됩니다.
- 스크롤 텍스트 필드: 스크롤 텍스트 필드에 대량의 텍스트를 표시하는 응용 프로그램을 사용할 경우입니다. 스크롤 경계선으로 스크롤이 가능하도록 설정(즉, scrollRect 속성을 설정함)한 표시 객체 안에 텍스트 필드를 배치할 수 있습니다. 이렇게 하면 지정된 인스턴스에 대해 빠른 픽셀 스크롤을 수행할 수 있습니다. 사용자가 표시 객체 인스턴스를 스크롤하면 Flash Player나 AIR에서 스크롤된 픽셀은 위쪽으로 이동하며 전체 텍스트 필드가 다시 생성되는 대신 새롭게 표시되는 영역만 생성됩니다.
- 윈도우 시스템: 윈도우가 겹쳐지는 복잡한 구조가 포함된 응용 프로그램을 사용할 경우입니다. 각 윈도우는, 예를 들면 웹 브라우저 윈도우처럼 열려 있거나 닫혀 있을 수도 있습니다. cacheAsBitmap 속성을 true로 설정하여 각 윈도우를 표면으로 표시하면 각 윈도우는 서로 분리되고 캐시됩니다. 사용자는 윈도우를 드래그하여 서로 겹쳐지게 할 수 있으며, 각 윈도우에서는 벡터 내용을 다시 생성할 필요가 없습니다.
- 알파 채널 마스크: 알파 채널 마스크를 사용할 경우 cacheAsBitmap 속성을 true로 설정해야 합니다. 자세한 내용은 171페이지의 "표시 객체 마스크"를 참조하십시오.

이와 같은 모든 경우에 비트맵 캐싱을 활성화하면 벡터 그래픽이 최적화되므로 응용 프로그램의 응답성 및 상호 작용성이 향상됩니다.

또한 표시 객체에 필터를 적용할 때마다 cacheAsBitmap이 자동으로 true로 설정됩니다. 이 속성을 false로 명시적으로 설정한 경우에도 마찬가지입니다. 표시 객체에서 모든 필터를 지우면 cacheAsBitmap 속성이 마지막 설정된 값으로 돌아갑니다.

비트맵 캐싱의 사용을 피해야 할 경우

이 기능을 잘못된 환경에서 사용하면 SWF 파일의 성능이 저하될 수도 있습니다. 비트맵 캐싱을 사용할 경우 다음 지침을 따라야 합니다.

- 표면(캐싱이 활성화된 표시 객체)을 과도하게 사용하지 않습니다. 각 표면은 일반 표시 객체보다 메모리를 더 사용하므로 렌더링 성능을 향상시킬 경우에만 표면을 활성화해야 합니다.

  캐시된 비트맵은 일반 표시 객체보다 훨씬 더 많은 메모리를 사용할 수도 있습니다. 예를 들어, 스테이지 위의 크기가 250 x 250 픽셀인 Sprite 인스턴스가 캐시될 경우 250KB가 사용되는 반면에, 캐시되지 않은 일반 Sprite 인스턴스는 1KB를 사용합니다.

- 캐시된 표면에 확대/축소를 적용하지 않습니다. 비트맵 캐싱을 남용하면 특히 내용을 확대 및 축소할 경우 상당한 양의 메모리가 소비됩니다(앞의 항목 참조).
- 표면은 대부분 정적인, 즉 움직임이 거의 없는 표시 객체 인스턴스에 대해 사용합니다. 이 경우에 인스턴스를 드래그하거나 이동할 수 있지만 인스턴스의 내용은 움직이거나 크게 변경되어서는 안 됩니다. 애니메이션 또는 변경되는 내용은 애니메이션 또는 Video 인스턴스가 포함된 MovieClip 인스턴스와 비슷합니다. 예를 들어 인스턴스를 회전하거나 변형하면 해당 인스턴스는 표면 및 벡터 데이터간에서 전환되는데, 이는 처리하기 어려운 작업이며 SWF 파일에 부정적인 영향을 미치기도 합니다.
- 표면을 벡터 데이터와 혼합하면 Flash Player 및 AIR(때로는 컴퓨터)에서 수행해야 하는 처리 작업이 증가됩니다. 가능하면 표면을 서로 그룹화하는 것이 좋으며, 예를 들어 윈도우 응용 프로그램을 만드는 경우가 여기에 해당합니다.
- 그래픽이 자주 변경되는 객체는 캐시하지 않는 것이 좋습니다. 크기 조절, 기울이기, 표시 객체 회전, 알파 또는 색상 변형 변경, 자식 표시 객체 이동, 그래픽 속성을 사용한 그리기 등을 수행할 때마다 비트맵 캐시는 다시 그려집니다. 이러한 작업이 모든 프레임에서 이루어지면 런타임에서 객체를 비트맵으로 그린 다음 해당 비트맵을 스테이지로 복사해야 하며, 이 경우 캐시되지 않은 객체를 스테이지에 그릴 때와 달리 추가적인 작업이 발생합니다. 캐싱 및 업데이트 빈도 간 성능상의 균형 조건은 표시 객체의 복잡도와 크기에 따라 달라지며 특정 내용을 테스트한 후에만 결정할 수 있습니다.

비트맵 캐싱 활성화

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체에 대한 비트맵 캐싱을 활성화하려면 cacheAsBitmap 속성을 true로 설정합니다.

```
mySprite.cacheAsBitmap = true;
```

cacheAsBitmap 속성을 true로 설정하면 표시 객체가 전체 좌표를 기준으로 자동으로 픽셀에 물리는 것을 볼 수 있습니다. SWF 파일을 테스트하면 복잡한 벡터 이미지에서 수행되는 애니메이션이 더욱 빠르게 렌더링되는 것이 보입니다.

다음 중 한 가지 이상의 경우에 해당되면 cacheAsBitmap이 true로 설정되었더라도 표면(캐시된 비트맵)이 만들어지지 않습니다.

- 비트맵의 높이나 너비가 2880 픽셀을 초과한 경우
- 메모리 부족으로 비트맵 할당에 실패한 경우

캐시된 비트맵 변형 행렬

Adobe AIR 2.0 이상(휴대 장치 프로파일)

휴대 장치용 AIR 응용 프로그램에서 cacheAsBitmap 속성을 설정할 때마다 cacheAsBitmapMatrix 속성도 설정해야 합니다. 이 속성을 설정하면 재렌더링을 트리거하지 않고 표시 객체에 더 다양한 변형을 적용할 수 있습니다.

```
mySprite.cacheAsBitmap = true;
mySprite.cacheAsBitmapMatrix = new Matrix();
```

이 행렬 속성을 설정하면 객체를 다시 캐시하지 않고 표시 객체에 다음의 추가적인 변형을 적용할 수 있습니다.

- 픽셀 물리기 없는 이동 또는 변환
- 회전

- 크기 조절
- 경사
- 알파 변경(0~100%의 투명도 범위)

이러한 변형은 캐시된 비트맵에 직접 적용됩니다.

불투명한 배경색 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체에 불투명한 배경을 설정할 수 있습니다. 예를 들어, SWF에 복잡한 벡터 아트가 포함된 배경이 있을 경우 opaqueBackground 속성을 지정된 색상(대개 스테이지와 같은 색상)으로 설정하면 됩니다. 색상은 숫자(일반적으로 16진수 색상 값)로 지정됩니다. 그러면 배경은 비트맵으로 처리되어 성능을 최적화하는 데 도움이 됩니다.

cacheAsBitmap을 true로 설정하고 또한 opaqueBackground 속성을 지정된 색상으로 설정하면, opaqueBackground 속성으로 인해 내부 비트맵이 불투명해지고 더 빠르게 렌더링됩니다. cacheAsBitmap을 true로 설정하지 않으면 opaqueBackground 속성은 표시 객체의 배경에 불투명한 벡터 사각형 모양을 추가하게 됩니다. 그러면 비트맵이 자동으로 만들어지지 않습니다.

다음 예제에서는 성능을 최적화하기 위해 표시 객체의 배경을 설정하는 방법을 보여 줍니다.

```
myShape.cacheAsBitmap = true;
myShape.opaqueBackground = 0xFF0000;
```

이 경우 myShape라는 Shape의 배경색은 빨강(0xFF0000)으로 설정됩니다. Shape 인스턴스에 녹색 삼각형 드로잉이 포함되어 있다고 가정할 때, 배경이 흰색인 스테이지에서는 Shape 인스턴스의 경계 상자(모양을 완전히 감싸고 있는 사각형) 내의 공백이 빨간색인 녹색 삼각형으로 표시됩니다.

이 코드는 단색 빨강 배경을 가진 스테이지에서 사용될 경우에 더 적합합니다. 다른 색상의 배경에서는 해당 색상이 대신 지정됩니다. 예를 들어, 배경이 흰색인 SWF의 경우 opaqueBackground 속성은 일반적으로 0xFFFFFF(순백색)로 설정됩니다.

블렌딩 모드 적용

Flash Player 9 이상, Adobe AIR 1.0 이상

블렌딩 모드에서는 하나의 이미지(기본 이미지)의 색상을 또 다른 이미지(블렌드 이미지)의 색상과 결합하여 제3의 이미지, 즉 화면에 실제로 표시되는 결과 이미지를 만들어냅니다. 기본 이미지의 개별 픽셀 값을 이에 대응되는 혼합 이미지의 픽셀 값과 함께 처리하여 같은 위치에 새로운 픽셀 값을 산출합니다.

모든 표시 객체에는 다음 블렌딩 모드 중 하나로 설정될 수 있는 blendMode 속성이 있습니다. 이러한 값은 BlendMode 클래스에 정의된 상수입니다. 또한 상수의 실제 값인 문자열 값을 괄호로 묶어 사용할 수도 있습니다.

- BlendMode.ADD("add"): 두 이미지 사이에서 색상을 밝게 하는 디졸브 애니메이션 효과를 만드는 데 주로 사용됩니다.
- BlendMode.ALPHA("alpha"): 배경 위에 전경의 투명도를 적용하는 데 주로 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.DARKEN("darken"): superimpose 유형에 주로 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.DIFFERENCE ("difference"): 보다 강렬한 색상을 만드는 데 주로 사용됩니다.

- BlendMode.ERASE("erase"): 전경의 알파를 사용하여 배경의 일부를 잘라내는(지우는) 데 주로 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.HARDLIGHT("hardlight"): 음영 효과를 만드는 데 주로 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.INVERT("invert"): 배경을 반전시키는 데 사용됩니다.
- BlendMode.LAYER("layer"): 특정 표시 객체의 사전 합성(precomposition)을 위한 임시 버퍼를 만드는 데 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.LIGHTEN("lighten"): superimpose 유형에 주로 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.MULTIPLY("multiply"): 그림자 및 심도 효과를 만드는 데 주로 사용됩니다.
- BlendMode.NORMAL("normal"): 혼합 이미지의 픽셀 값이 기본 이미지의 픽셀 값을 무시하도록 지정하는 데 사용됩니다.
- BlendMode.OVERLAY("overlay"): 음영 효과를 만드는 데 주로 사용됩니다. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.SCREEN("screen"): 강조 및 렌즈 플레어를 만드는 데 주로 사용됩니다.
- BlendMode.SHADER("shader"): Pixel Bender 셰이더를 사용하여 사용자 정의 블렌딩 효과를 만들도록 지정하는 데 사용됩니다. 셰이더 사용에 대한 자세한 내용은 271페이지의 "Pixel Bender 셰이더를 사용한 작업"을 참조하십시오. GPU 렌더링에서는 지원되지 않습니다.
- BlendMode.SUBTRACT("subtract"): 두 이미지 사이에서 색상을 어둡게 하는 디졸브 애니메이션 효과를 만드는 데 주로 사용됩니다.

DisplayObject 색상 조절

Flash Player 9 이상, Adobe AIR 1.0 이상

ColorTransform 클래스의 메서드(flash.geom.ColorTransform)를 사용하여 표시 객체의 색상을 조절할 수 있습니다. 각 표시 객체에는 Transform 클래스의 인스턴스이고 표시 객체에 적용되는 다양한 변환(예: 회전, 크기 또는 위치 변경 등)에 대한 정보가 들어 있는 transform 속성이 있습니다. 기하학적 변형 외에도 Transform 클래스에는 ColorTransform 클래스의 인스턴스이고 표시 객체에 대한 색상 조절을 위한 액세스를 제공하는 colorTransform 속성이 포함되어 있습니다. 표시 객체의 색상 변형 정보에 액세스하려면 다음과 같은 코드를 사용할 수 있습니다.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;
```

ColorTransform 인스턴스를 만든 경우 속성 값을 보고 이미 적용된 색상 변형을 확인하거나, 해당 값을 설정하여 표시 객체의 색상을 변경할 수 있습니다. 변경 후에 표시 객체를 업데이트하려면 ColorTransform 인스턴스를 transform.colorTransform 속성에 다시 할당해야 합니다.

```
var colorInfo:ColorTransform = myDisplayObject.transform.colorTransform;

// Make some color transformations here.

// Commit the change.
myDisplayObject.transform.colorTransform = colorInfo;
```

코드를 사용하여 색상 값 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

ColorTransform 클래스의 color 속성을 사용하여 표시 객체에 특정 RGB(Red, Green, Blue) 색상 값을 할당할 수 있습니다. 다음 예제에서는 color 속성을 사용하여 사용자가 blueBtn 버튼을 클릭할 때 square 표시 객체의 색상을 파랑으로 변경합니다.

```
// 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);
```

color 속성을 사용하여 표시 객체의 색상을 변경하면 객체에 사용된 색상 수에 관계없이 전체 객체의 색상이 완전히 변경됩니다. 예를 들어, 위에 검정 텍스트가 있는 녹색 원이 들어 있는 표시 객체가 있는 경우 해당 객체와 연관된 ColorTransform 인스턴스의 color 속성을 빨강 음영으로 설정하면 전체 객체와 원 및 텍스트가 빨강으로 변경됩니다. 따라서 텍스트가 객체의 나머지 부분과 더 이상 구분되지 않습니다.

코드를 사용하여 색상 및 밝기 효과 변경

Flash Player 9 이상, Adobe AIR 1.0 이상

디지털 사진처럼 여러 색상으로 된 표시 객체가 있을 때 객체의 색상을 완전히 변경하지 않고 기존 색상을 기반으로 표시 객체의 색상을 조절한다고 가정합니다. 이 경우 ColorTransform 클래스는 이러한 유형의 조정에 사용할 수 있는 일련의 승수 및 오프셋 속성을 포함합니다. redMultiplier, greenMultiplier, blueMultiplier 및 alphaMultiplier 승수 속성은 컬러 사진 필터 또는 컬러 썬글라스처럼 작동하여 표시 객체에서 특정 색상을 증가 또는 감소시킵니다. 오프셋 속성(redOffset, greenOffset, blueOffset, alphaOffset)을 사용하여 객체에 특정 색상을 추가하거나, 특정 색상이 가질 수 있는 최소값을 지정할 수 있습니다.

이러한 승수 및 오프셋 속성은 속성 관리자의 [색상] 팝업 메뉴에서 [고급]을 선택할 때 Flash 제작 도구에서 동영상 클립 심볼에 사용할 수 있는 고급 색상 설정과 동일합니다.

다음 코드에서는 JPEG 이미지를 로드하고 색상 변환을 적용합니다. 이 변환에서는 마우스 포인터가 x축 및 y축을 따라 이동하면서 빨강 및 녹색 채널을 조정합니다. 이 경우 오프셋 값이 지정되어 있지 않기 때문에 화면에 표시된 각 색상 채널의 색상 값은 이미지의 원본 색상 값의 백분율입니다. 즉, 특정 픽셀에 표시되는 가장 진한 빨강 또는 녹색이 해당 픽셀의 원본 빨강 또는 녹색의 양입니다.

```
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);
```

객체 회전

Flash Player 9 이상, Adobe AIR 1.0 이상

rotation 속성을 사용하여 표시 객체를 회전할 수 있습니다. 이 값을 보고 객체가 회전되었는지 여부를 확인할 수 있습니다. 객체를 회전하려면 이 속성을 숫자로 설정합니다. 이 숫자는 객체에 적용할 회전 각도를 나타냅니다. 예를 들어, 다음 코드 행은 square 객체를 45도(전체 회전의 1/8) 회전합니다.

```
square.rotation = 45;
```

189페이지의 “기하 도형을 사용한 작업”에 설명된 변형 행렬을 사용하여 표시 객체를 회전할 수도 있습니다.

객체에 페이드 인/아웃 효과 적용

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체의 투명도를 제어하여 객체를 부분적으로 또는 완전히 투명하게 만들 수 있습니다. 또는 투명도를 변경하여 객체에 페이드 인/아웃 효과를 나타낼 수 있습니다. DisplayObject 클래스의 alpha 속성은 표시 객체의 투명도(정확히 불투명도)를 정의합니다. alpha 속성은 0과 1 사이의 값으로 설정할 수 있습니다. 여기서 0은 완전 투명을, 1은 완전 불투명을 나타냅니다. 예를 들어, 다음 코드 행은 마우스를 클릭하면 myBall 객체를 부분적(50%)으로 투명하게 만듭니다.

```
function fadeBall(event:MouseEvent):void
{
    myBall.alpha = .5;
}
myBall.addEventListener(MouseEvent.CLICK, fadeBall);
```

ColorTransform 클래스를 통해 사용할 수 있는 색상 조절 기능을 사용하여 표시 객체의 투명도를 변경할 수도 있습니다. 자세한 내용은 168페이지의 “DisplayObject 색상 조절”을 참조하십시오.

표시 객체 마스크

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체를 마스크로 사용하여 다른 표시 객체의 내용을 볼 수 있는 구멍을 만들 수 있습니다.

마스크 정의

표시 객체를 다른 표시 객체의 마스크로 지정하려면 마스크 객체를 마스크를 적용할 표시 객체의 mask 속성으로 설정합니다.

```
// Make the object maskSprite be a mask for the object mySprite.
mySprite.mask = maskSprite;
```

마스크가 적용된 표시 객체는 마스크로 작용하는 표시 객체의 모든 불투명 영역 아래에 나타납니다. 예를 들어, 다음 코드는 100 x 100 픽셀의 빨강 사각형을 포함하는 Shape 인스턴스와 반경이 25 픽셀인 파랑 원을 포함하는 Sprite 인스턴스를 만듭니다. 원을 클릭하면 해당 원이 사각형에 대한 마스크로 설정되므로, 사각형에서 원으로 가려진 부분만이 표시됩니다. 즉, 빨강 원만 보이게 됩니다.

```
// 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);
```

마스크 역할을 하는 표시 객체는 드래그할 수 있고, 애니메이션 처리하거나 동적으로 크기를 조절할 수 있으며, 단일 마스크 내에서 개별 모양을 사용할 수 있습니다. 마스크 표시 객체를 표시 목록에 추가할 필요는 없습니다. 그러나, 스테이지 크기를 조절할 때 마스크 객체의 크기를 조절하거나, 마스크와의 사용자 상호 작용(예: 사용자 제어 드래그 및 크기 조절)을 가능하도록 하려면 마스크 객체를 표시 목록에 추가해야 합니다. 마스크 객체를 표시 목록에 추가할 경우 표시 객체의 실제 z 인덱스(전후 순서)는 중요하지 않습니다. 마스크 객체는 화면에는 표시되지 않고 마스크로만 표시됩니다. 마스크 객체가 프레임이 여러 개 있는 MovieClip 인스턴스인 경우 마스크 객체는 마스크 역할을 하지 않을 때와 마찬가지로 해당 타임라인에 모든 프레임을 재생합니다. mask 속성을 null로 설정하여 마스크를 제거할 수 있습니다.

```
// remove the mask from mySprite
mySprite.mask = null;
```

마스크를 사용하여 다른 마스크에 마스크를 적용할 수는 없습니다. 마스크 표시 객체의 alpha 속성은 설정할 수 없습니다. 마스크로 사용되는 표시 객체에는 채우기만 사용되고 획은 무시됩니다.

AIR 2

cacheAsBitmap 및 cacheAsBitmapMatrix 속성을 설정하여 마스크 처리된 표시 객체를 캐시하는 경우 마스크는 마스크 처리된 표시 객체의 자식이어야 합니다. 마찬가지로 마스크 처리된 표시 객체가 캐시되는 표시 객체 컨테이너의 자손인 경우 마스크 및 표시 객체 모두 해당 컨테이너의 자손이어야 합니다. 마스크 처리된 객체가 둘 이상의 캐시된 표시 객체 컨테이너의 자손인 경우, 마스크는 표시 목록에서 마스크 처리된 객체와 가장 가까이에 있는 캐시된 컨테이너의 자손이어야 합니다.

장치 글꼴 마스크

표시 객체를 사용하면 장치 글꼴로 설정된 텍스트에 마스크를 적용할 수 있습니다. 표시 객체를 사용하여 장치 글꼴에 설정된 텍스트에 마스크를 적용할 때는 마스크의 사각형 경계 상자가 마스크 모양으로 사용됩니다. 즉, 장치 글꼴 텍스트에 대해 사각형이 아닌 표시 객체 마스크를 만들면 SWF 파일에 나타나는 마스크의 모양은 마스크 자체의 모양이 아니라 마스크의 사각형 경계 상자의 모양이 됩니다.

알파 채널 마스크

다음과 같이 알파 채널 마스크는 마스크 표시 객체와 마스크가 적용된 표시 객체 모두 비트맵 캐싱을 사용할 경우에 지원됩니다.

```
// maskShape is a Shape instance which includes a gradient fill.
mySprite.cacheAsBitmap = true;
maskShape.cacheAsBitmap = true;
mySprite.mask = maskShape;
```

예를 들어, 마스크가 적용된 표시 객체에 적용되어 있는 필터와 독립적으로 마스크 객체에 필터를 사용할 경우 알파 채널 마스크를 적용합니다.

다음 예제에서는 외부 이미지 파일을 스테이지에 로드합니다. 이 이미지(정확히 말해서 이미지가 로드되는 Loader 인스턴스)는 마스크가 적용되는 표시 객체입니다. 이미지 위에 그래디언트 타원(중앙이 검정이고 가장자리로 갈수록 점점 투명해짐)이 그려집니다. 이것이 알파 마스크가 됩니다. 두 표시 객체는 모두 비트맵 캐싱 기능이 설정되어 있습니다. 타원이 이미지에 대한 마스크로 설정되고 드래그할 수 있게 됩니다.

```
// 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);
```

객체 애니메이션

Flash Player 9 이상, Adobe AIR 1.0 이상

애니메이션은 어떤 것을 움직이게 하거나 또는 시간에 따라 변화되도록 하는 프로세스입니다. 스크립트 애니메이션은 비디오 게임의 기본이 되는 요소이며, 기타 응용 프로그램에 세련되고 유용한 상호 작용을 추가하기 위해서도 자주 사용됩니다.

스크립트 애니메이션의 기본 아이디어는 변경이 발생해야 하며 이 변경이 시간의 경과에 따라 여러 단계로 나뉘어 진행되어야 한다는 것입니다. ActionScript에서는 일반 루프 문을 사용하여 어떤 작업을 쉽게 반복할 수 있습니다. 그러나 루프는 표시를 업데이트하기 이전에 모든 반복을 완료합니다. 스크립트 애니메이션을 만들려면 시간에 따라 일부 작업을 반복적으로 수행하고 반복이 실행될 때마다 화면을 업데이트하는 ActionScript를 작성해야 합니다.

예를 들어, 화면을 따라 공이 이동하는 간단한 애니메이션을 만든다고 가정합니다. ActionScript에는 시간 경로를 추적하여 화면을 적절하게 업데이트할 수 있는 간단한 메커니즘이 포함되어 있습니다. 즉, 공이 목표 지점에 도달할 때까지 조금씩 이동하는 코드를 작성할 수 있습니다. 이동할 때마다 화면이 업데이트되어 스테이지를 가로지르는 모션이 뷰어에 표시됩니다.

실질적인 관점에서 스크립트 애니메이션을 SWF 파일의 프레임 속도(이 속도에 따라 Flash Player 또는 AIR에서 화면을 업데이트하는 빈도가 정의되기 때문)와 동기화하여, 새 프레임이 표시될 때마다 애니메이션 변경을 하나씩 만들 수 있습니다. 각 표시 객체에는 SWF 파일의 프레임 속도에 따라 전달되는 enterFrame 이벤트(프레임별로 하나의 이벤트)가 있습니다. 스크립트 애니메이션을 만드는 대부분의 개발자는 enterFrame 이벤트를 사용하여 일정 시간 반복되는 작업을 만듭니다. enterFrame 이벤트를 수신하여 애니메이션 처리된 공을 프레임별로 특정 양만큼 이동하고 각 프레임에 대해 화면이 업데이트되면 공을 새 위치에 다시 그려 모션을 생성하는 코드를 작성할 수 있습니다.

참고: 일정 시간 작업을 반복적으로 수행하는 다른 방법은 Timer 클래스를 사용하는 것입니다. Timer 인스턴스는 지정된 시간이 경과할 때마다 이벤트 알림을 트리거합니다. Timer 클래스의 timer 이벤트를 처리하고 시간 간격을 작게(몇 분의 일 초) 설정하여 애니메이션을 수행하는 코드를 작성할 수 있습니다. Timer 클래스 사용에 대한 자세한 내용은 3페이지의 "시간 간격 제어"를 참조하십시오.

다음 예제에서는 circle이라는 원 Sprite 인스턴스를 스테이지에 만듭니다. 사용자가 원을 클릭하면 스크립트 애니메이션 시퀀스가 시작되어 circle이 완전히 투명해질 때까지 페이드됩니다(alpha 속성이 감소됨).

```
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);
```

사용자가 원을 클릭하면 fadeCircle() 함수가 enterFrame 이벤트의 리스너로 구독되어 프레임마다 한 번씩 호출됩니다. 이 함수는 alpha 속성을 변경하여 circle에 페이드 효과를 적용하기 때문에 프레임마다 한 번씩 원의 alpha가 .05(5%)씩 감소하고 화면이 업데이트됩니다. alpha 값이 0(circle이 완전 투명한 상태)이 되면 fadeCircle() 함수가 이벤트 리스너 구독에서 제거되어 애니메이션이 종료됩니다.

예를 들어, 동일한 코드를 사용하여 페이드 효과 대신 애니메이션 모션을 만들 수 있습니다. 함수에서 alpha를 enterFrame 이벤트 리스너인 다른 속성으로 대체하면 해당 속성이 대신 애니메이션화됩니다. 예를 들어, 다음 행을

```
    circle.alpha -= .05;
```

다음 코드로 변경하면

```
    circle.x += 5;
```

x 속성이 애니메이션화되어 원이 스테이지의 오른쪽으로 이동합니다. 애니메이션이 종료되는 조건을 변경하여 원하는 x 좌표에 도달하면 애니메이션이 종료(즉, enterFrame 리스너 구독 해제)되도록 할 수 있습니다.

스테이지 방향

AIR 2.0 이상

일반적으로 사용자가 휴대 장치를 회전하는 경우 디스플레이가 똑바로 보이도록 사용자 인터페이스의 방향이 재조정됩니다. 응용 프로그램에서 자동 방향을 활성화하면 장치에서 디스플레이의 방향이 적절하게 유지되지만 스테이지의 종횡비가 변경되는 경우에는 사용자가 직접 해당 내용이 올바르게 표시되도록 조정해야 합니다. 자동 방향을 비활성화하면 사용자가 방향을 수동으로 변경하지 않는 이상 장치 디스플레이가 고정된 상태로 유지됩니다.

AIR 응용 프로그램은 여러 다양한 휴대 장치 및 운영 체제에서 실행됩니다. 기본적인 방향 비헤이비어는 운영 체제마다 서로 다르며 동일한 운영 체제에서도 장치에 따라 서로 다릅니다. 모든 장치 및 운영 체제에서 작동하는 단순한 설계 전략을 채택하면 자동 방향을 설정하고 스테이지 resize 이벤트를 수신하여 응용 프로그램 레이아웃을 새로 고쳐야 할 때를 결정할 수 있습니다.

또는 응용 프로그램에서 세로 종횡비나 가로 종횡비 중 하나만 지원하는 경우 자동 방향을 해제하고 AIR 응용 프로그램 설명자에서 지원되는 종횡비를 설정해도 됩니다. 이러한 설계 전략을 활용하면 일관된 비헤이비어가 가능하고 선택된 종횡비에 대해 최적의 방향을 지정할 수 있습니다. 예를 들어 가로 종횡비를 지정하는 경우 선택한 방향은 가로 모드의 슬라이드 아웃 키보드가 있는 장치에 적합합니다.

현재의 스테이지 방향 및 종횡비 가져오기

방향은 장치의 정상 위치에 상대적인 위치로 보고됩니다. 대부분의 장치에는 명확한 수직 위치가 있습니다. 이 위치는 **기본** 방향으로 간주됩니다. 가능한 세 가지 다른 방향은 **왼쪽 회전**, **오른쪽 회전** 및 **거꾸로**입니다. StageOrientation 클래스는 방향 값을 설정하거나 비교할 때 사용되는 문자열 상수를 정의합니다.

Stage 클래스는 방향을 보고하는 다음 두 개의 속성을 정의합니다.

- Stage.deviceOrientation - 기본 위치를 기준으로 장치의 상대적 실제 방향을 보고합니다.

  참고: deviceOrientation은 응용 프로그램이 처음 시작할 때 또는 장치가 눕혀져 있을 때는 사용하지 못할 수도 있습니다. 이 경우 장치 방향은 **알 수 없음**으로 보고됩니다.

- Stage.orientation - 기본 위치를 기준으로 스테이지의 상대적 방향을 보고합니다. 자동 방향이 설정된 경우 스테이지는 장치의 반대 방향으로 회전하여 수직을 유지합니다. 따라서 orientation 속성에서 보고되는 오른쪽 및 왼쪽 위치는 deviceOrientation 속성에서 보고되는 위치와 반대가 됩니다. 예를 들어 deviceRotation이 **오른쪽 회전**을 보고하면 orientation은 **왼쪽 회전**을 보고합니다.

스테이지의 종횡비는 해당 스테이지의 현재 폭과 높이를 서로 비교하여 간단히 산출할 수 있습니다.

```
var aspect:String = this.stage.stageWidth >= this.stage.stageHeight ? StageAspectRatio.LANDSCAPE :
StageAspectRatio.PORTRAIT;
```

자동 방향

자동 방향이 설정되어 있는 상태에서 사용자가 장치를 회전하면 운영 체제는 시스템 작업 표시줄 및 응용 프로그램을 비롯하여 전체 사용자 인터페이스의 방향을 재조정합니다. 그 결과 스테이지의 종횡비는 세로에서 가로로 또는 그 역으로 변경됩니다. 종횡비가 변경되면 스테이지 크기 또한 변경됩니다.

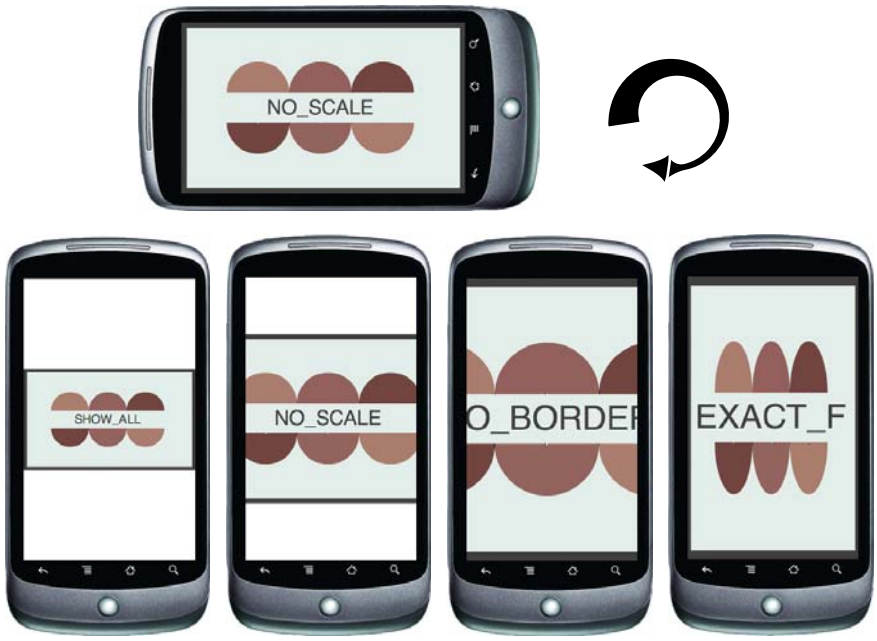
Stage의 autoOrients 속성을 true 또는 false로 설정하면 런타임에 자동 방향을 설정하거나 해제할 수 있습니다. 또한 <autoOrients> 요소를 사용하여 AIR 응용 프로그램 설명자에서 이 속성의 초기 값을 설정할 수 있습니다. AIR 2.6 이전 버전에서 autoOrients는 읽기 전용 속성이며 응용 프로그램 설명자에서만 설정할 수 있습니다.

가로 또는 세로 종횡비를 지정하고 자동 방향도 사용하도록 설정하면 AIR에서 자동 방향을 지정된 종횡비로 제한합니다.

스테이지 크기 변경

스테이지 크기가 변경되면 스테이지 내용은 Stage 객체의 scaleMode 및 align 속성에 지정된 대로 크기와 위치가 조정됩니다. 대부분의 경우 Stage scaleMode 설정에서 제공되는 자동 비헤이비어에 의존하면 좋은 결과를 얻을 수 없으며, 그래픽 및 구성 요소를 다시 배치하거나 다시 그려서 둘 이상의 종횡비를 지원하도록 하는 것이 좋습니다. 또한 유연한 레이아웃 논리를 제공하면 화면 크기 및 종횡비가 서로 다른 여러 장치에서 응용 프로그램이 더욱 원활하게 작동합니다.

다음 그림은 일반적인 휴대 장치를 회전할 경우에 나타나는 여러 scaleMode 설정의 효과를 보여 줍니다.



가로 종횡비에서 세로 종횡비로의 회전

위 그림은 여러 크기 조절 모드로 가로 종횡비에서 세로 종횡비로 회전할 때 발생하는 크기 조절 비헤이비어를 보여 줍니다. 세로에서 가로로 회전하면 서로 비슷한 효과 세트가 나타납니다.

방향 변경 이벤트

Stage 객체는 방향 변경을 감지하고 이에 응답하는 데 사용할 수 있는 두 유형의 이벤트를 전달합니다. 자동 방향이 설정되면 스테이지 resize 및 orientationChange 이벤트가 전달됩니다.

resize 이벤트는 자동 방향을 통해 디스플레이를 똑바로 유지하려는 경우에 활용할 수 있는 최적의 이벤트입니다. 스테이지에서 resize 이벤트를 전달하면 해당 내용이 필요에 따라 다시 배치되거나 다시 그려집니다. resize 이벤트는 스테이지 크기 조절 모드가 noScale로 설정된 경우에만 전달됩니다.

orientationChange 이벤트 또한 방향 변경을 감지하는 데 사용할 수 있습니다. orientationChange 이벤트는 자동 방향이 설정된 경우에만 전달됩니다.

참고: 일부 모바일 플랫폼에서 스테이지는 resize 또는 orientationChange 이벤트를 전달하기 전에 취소 가능한 orientationChanging 이벤트를 전달합니다. 이 이벤트는 일부 플랫폼에서는 지원되지 않으므로 너무 의존하지 않는 것이 좋습니다.

수동 방향

AIR 2.6 이상

Stage setOrientation() 또는 setAspectRatio() 메서드를 사용하여 스테이지 방향을 제어할 수 있습니다.

스테이지 방향 설정

Stage 객체의 setOrientation() 메서드를 사용하여 런타임에 스테이지 방향을 설정할 수 있습니다. 다음과 같이 StageOrientation 클래스를 통해 정의된 문자열 상수를 사용하여 원하는 방향을 지정합니다.

```
this.stage.setOrientation( StageOrientation.ROTATED_RIGHT );
```

일부 장치 및 운영 체제에서는 특정 방향을 지원하지 않습니다. 예를 들어 Android 2.2에서는 세로 표준 장치에서 왼쪽으로 회전된 방향을 프로그래밍 방식으로 선택할 수 없을 뿐 아니라 위/아래가 뒤집힌 방향은 전혀 지원하지 않습니다. 스테이지의 supportedOrientations 속성은 setOrientation() 메서드로 전달할 수 있는 방향의 목록을 제공합니다.

```
var orientations:Vector.<String> = this.stage.supportedOrientations;
for each( var orientation:String in orientations )
{
    trace( orientation );
}
```

스테이지 종횡비 설정

주로 염려되는 부분이 스테이지의 종횡비인 경우 종횡비를 세로 또는 가로로 설정할 수 있습니다. 종횡비는 AIR 응용 프로그램 설명자에서 또는 런타임에 Stage setAspectRatio() 메서드를 사용하여 설정할 수 있습니다.

```
this.stage.setAspectRatio( StageAspectRatio.LANDSCAPE );
```

런타임에서는 지정된 종횡비에 대해 사용 가능한 두 가지 방향 중 하나를 선택합니다. 이는 현재 장치 방향과 일치하지 않을 수 있습니다. 예를 들어 기본 방향이 위/아래가 뒤집힌 방향에 우선하여 선택되고(AIR 3.2 이하), 슬라이드 키보드에 적합한 방향이 반대 방향에 우선하여 선택됩니다.

(AIR 3.3 이상) AIR 3.3(SWF 버전 16)부터는 StageAspectRatio.ANY 상수를 사용할 수도 있습니다. Stage.autoOrients가 true로 설정된 상태에서 setAspectRatio(StageAspectRatio.ANY)를 호출하면 응용 프로그램에서 모든 방향(가로-왼쪽, 가로-오른쪽, 세로, 세로-거꾸로)을 다시 지정할 수 있습니다. 또한 AIR 3.3부터는 종횡비가 영구적이며, 지정된 방향에서 장치를 더 회전할 수 없습니다.

예: 장치 방향과 일치하도록 스테이지 방향 설정

다음 예제에서는 현재 장치 방향과 일치하도록 스테이지 방향을 업데이트하는 함수를 보여 줍니다. 스테이지의 deviceOrientation 속성은 자동 방향이 해제되어 있는 경우에도 장치의 물리적 방향을 나타냅니다.

```
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
    }
}
```

방향은 비동기적으로 변경됩니다. 스테이지에서 전달한 orientationChange 이벤트를 수신하여 변경이 완료되는 것을 감지할 수 있습니다. 장치에서 방향이 지원되지 않는 경우 오류가 발생하지 않은 상태로 setOrientation() 호출이 실패합니다.

표시 내용을 동적으로 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같은 외부 표시 에셋을 ActionScript 3.0 응용 프로그램으로 로드할 수 있습니다.

- ActionScript 3.0에서 제작된 SWF 파일 - 이 파일은 Sprite 또는 MovieClip 클래스이거나 Sprite를 확장하는 모든 클래스가 될 수 있습니다. iOS의 AIR 응용 프로그램에서는 ActionScript 바이트코드가 포함되지 않은 SWF 파일만 로드할 수 있습니다. 따라서 이미지 및 사운드 같은 포함된 데이터가 들어 있는 SWF 파일은 로드할 수 있지만 실행 코드가 들어 있는 SWF 파일은 로드할 수 없습니다.
- 이미지 파일 - JPG, PNG 및 GIF 파일을 포함합니다.
- AVM1 SWF 파일 - ActionScript 1.0 또는 2.0에서 작성된 SWF 파일입니다. 모바일 AIR 응용 프로그램에서는 지원되지 않습니다.

Loader 클래스를 사용하여 이러한 에셋을 로드합니다.

표시 객체 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

Loader 객체는 SWF 파일과 그래픽 파일을 응용 프로그램으로 로드하는 데 사용됩니다. Loader 클래스는 DisplayObjectContainer 클래스의 하위 클래스입니다. Loader 객체는 하나의 자식 표시 객체(로드하는 SWF 또는 그래픽 파일을 나타내는 표시 객체)만 표시 목록에 포함할 수 있습니다. 다음 코드에서처럼 Loader 객체를 표시 목록에 추가할 때 로드된 자식 표시 객체를 표시 목록에 함께 추가합니다.

```
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
this.addChild(pictLdr);
```

SWF 파일 또는 이미지가 로드되면 로드된 표시 객체를 다른 표시 객체 컨테이너(예: 다음 예제의 container DisplayObjectContainer 객체)로 이동할 수 있습니다.

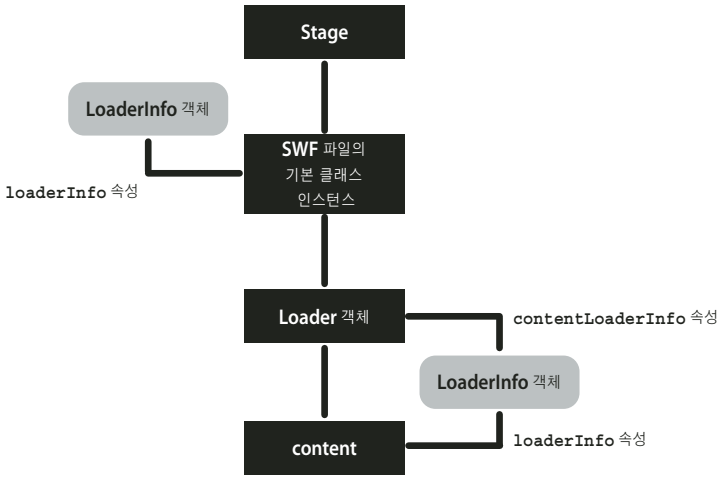
```
import flash.display.*;
import flash.net.URLRequest;
import flash.events.Event;
var container:Sprite = new Sprite();
addChild(container);
var pictLdr:Loader = new Loader();
var pictURL:String = "banana.jpg"
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, imgLoaded);
function imgLoaded(event:Event):void
{
    container.addChild(pictLdr.content);
}
```

로드 진행률 모니터링

Flash Player 9 이상, Adobe AIR 1.0 이상

파일 로드가 시작되면 LoaderInfo 객체가 만들어집니다. LoaderInfo 객체는 로드 진행률, 로더 및 로드 대상의 URL, 해당 미디어의 총 바이트 수, 미디어의 공칭 높이 및 폭 등과 같은 정보를 제공합니다. 또한 LoaderInfo 객체는 로드 진행률을 모니터링하는 이벤트를 전달합니다.

다음 다이어그램에서는 LoaderInfo 객체가 SWF 파일의 기본 클래스 인스턴스, Loader 객체 그리고 Loader 객체에 의해 로드된 객체에서 서로 다른 용도로 사용되는 방법을 보여 줍니다.



LoaderInfo 객체를 Loader 객체와 로드된 표시 객체 모두의 속성으로 액세스할 수 있습니다. 로드가 시작되면 Loader 객체의 contentLoaderInfo 속성을 통해 LoaderInfo 객체에 즉시 액세스할 수 있습니다. 표시 객체에서 로드가 완료된 경우 표시 객체의 loaderInfo 속성을 통해 LoaderInfo 객체를 로드된 표시 객체의 속성으로 액세스할 수도 있습니다. 로드된 표시 객체의 loaderInfo 속성은 Loader 객체의 contentLoaderInfo 속성과 동일한 LoaderInfo 객체를 나타냅니다. 즉, LoaderInfo 객체는 로드된 객체와 해당 객체를 로드한 Loader 객체 간(로더와 로드 대상 간)에 공유됩니다.

로드된 내용의 속성에 액세스하려면 다음 코드처럼 이벤트 리스너를 LoaderInfo 객체에 추가합니다.

```
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;
}
```

자세한 내용은 113페이지의 “이벤트 처리”를 참조하십시오.

로드 컨텍스트 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

Loader 클래스의 load() 또는 loadBytes() 메서드를 통해 외부 파일을 Flash Player 또는 AIR로 로드할 때 context 매개 변수를 선택적으로 지정할 수 있습니다. 이 매개 변수는 LoaderContext 객체입니다.

LoaderContext 클래스에는 로드된 내용을 사용하는 방법에 대한 컨텍스트를 정의할 수 있는 세 가지 속성이 포함되어 있습니다.

- checkPolicyFile: 이 속성은 SWF 파일이 아닌 이미지 파일을 로드할 때만 사용합니다. 이 속성을 true로 설정하면 Loader는 원래 서버에서 정책 파일을 확인합니다(956페이지의 "웹 사이트 컨트롤(정책 파일)" 참조). 이러한 설정은 Loader 객체를 포함하는 SWF 파일의 도메인과 다른 도메인에서의 내용에 대해서만 필요합니다. 서버에서 Loader 도메인에 권한을 부여하면 Loader 도메인에 있는 SWF 파일의 ActionScript는 로드된 이미지에서 데이터를 액세스할 수 있습니다. 즉, BitmapData.draw() 명령을 사용하여 로드된 이미지에서 데이터를 액세스할 수 있습니다.

  Loader 객체의 도메인과 다른 도메인에서의 SWF 파일은 Security.allowDomain()을 호출하여 특정 도메인을 허용할 수 있습니다.

- securityDomain: 이 속성은 이미지가 아닌 SWF 파일을 로드할 때만 사용합니다. Loader 객체를 포함하는 파일의 도메인과 다른 도메인에서의 SWF 파일에 대해 이 속성을 지정합니다. 이 옵션을 지정하면 Flash Player에서는 정책 파일이 있는지 확인합니다. 이 파일이 있는 경우 크로스 정책 파일에 허용된 도메인에서의 SWF 파일은 로드된 SWF 내용을 크로스 스크립팅할 수 있습니다. flash.system.SecurityDomain.currentDomain을 이 매개 변수로 지정할 수 있습니다.

- applicationDomain: 이 속성은 ActionScript 3.0으로 작성된 SWF 파일을 로드할 때만 사용합니다. 이미지 파일이나 ActionScript 1.0 또는 2.0으로 작성된 SWF 파일은 해당하지 않습니다. 파일을 로드할 때 applicationDomain 매개 변수를 flash.system.ApplicationDomain.currentDomain으로 설정하여 Loader 객체와 동일한 응용 프로그램 도메인에 파일을 포함시키도록 지정할 수 있습니다. 로드된 SWF 파일을 동일한 응용 프로그램 도메인에 저장하면 해당 클래스에 직접 액세스할 수 있습니다. 이렇게 하면 연관된 클래스 이름을 통해 액세스할 수 있는 포함된 미디어가 들어 있는 SWF 파일을 로드할 때 유용합니다. 자세한 내용은 132페이지의 "응용 프로그램 도메인 작업"을 참조하십시오.

다음은 다른 도메인에서 비트맵을 로드할 때 정책 파일을 확인하는 예제입니다.

```
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);
```

다음은 파일을 Loader 객체와 동일한 보안 샌드박스에 저장하기 위해 다른 도메인에서 SWF를 로드할 때 정책 파일을 확인하는 예제입니다. 이 코드는 로드된 SWF 파일에 있는 클래스를 Loader 객체와 동일한 응용 프로그램 도메인에 추가합니다.

```
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);
```

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 LoaderContext 클래스를 참조하십시오.

AIR for iOS의 SWF 파일 로드

Adobe AIR 3.6 이상, iOS에만 해당

iOS 장치에서는 런타임에 코드를 로드하고 컴파일하는 데 제약이 있습니다. 이 제약 때문에 외부 SWF 파일을 응용 프로그램에 로드하는 작업이 약간 다를 수밖에 없습니다.

- ActionScript 코드가 포함된 모든 SWF 파일은 응용 프로그램 패키지에 포함되어야 합니다. 코드가 포함된 SWF는 네트워크 등의 외부 소스로부터 로드할 수 없습니다. 응용 프로그램을 패키지화할 때 응용 프로그램 패키지에 포함된 모든 SWF 파일의 ActionScript 코드가 iOS 장치용 기본 코드로 컴파일됩니다.
- SWF 파일을 로드하고 언로드했다가 다시 로드할 수는 없습니다. 이렇게 하려고 하면 오류가 발생합니다.
- 메모리에 로드한 다음 언로드하는 동작은 데스크톱 플랫폼에서와 동일합니다. SWF 파일을 로드했다가 언로드하면 SWF에 포함된 모든 시각적 에셋이 메모리에서 언로드됩니다. 하지만 로드된 SWF의 ActionScript 클래스에 대한 클래스 참조는 메모리에 그대로 유지되며 ActionScript 코드에서 이를 액세스할 수 있습니다.
- 로드되는 모든 SWF는 기본 SWF 파일과 동일한 응용 프로그램 도메인에 로드되어야 합니다. 이는 기본 동작으로 수행되지 않습니다. 따라서 로드하는 각 SWF에 대해 기본 응용 프로그램 도메인을 지정하는 LoaderContext 객체를 생성하고 이 LoaderContext 객체를 Loader.load() 메서드 호출에 전달해야 합니다. SWF를 기본 SWF 응용 프로그램 도메인이 아닌 응용 프로그램 도메인에 로드하려고 하면 오류가 발생합니다. 이는 로드 대상 SWF에 시각적 에셋만 있고 ActionScript 코드가 없는 경우에도 마찬가지입니다.

  다음 예제에서는 SWF를 응용 프로그램 패키지에서 기본 SWF 응용 프로그램 도메인으로 로드할 때 사용하는 코드를 보여줍니다.

```
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);
```

에셋만 있고 코드가 없는 SWF 파일은 응용 프로그램 패키지 또는 네트워크에서 로드할 수 있습니다. 어느 쪽에서 로드하든 SWF 파일은 기본 응용 프로그램 도메인으로 로드되어야 합니다.

AIR 3.6 이전의 AIR 버전에서는 모든 코드가 컴파일 과정에서 기본 응용 프로그램 SWF 이외의 모든 SWF에서 제거됩니다. 시각적 에셋만 있는 SWF 파일은 런타임 시 응용 프로그램 패키지에 포함하고 로드할 수 있지만 코드는 그렇지 않습니다. ActionScript 코드가 포함된 SWF를 로드하려 하면 오류가 발생합니다. 이 오류로 인해 응용 프로그램에 "컴파일되지 않은 ActionScript"라는 오류 대화 상자가 나타납니다.

참고 사항

iOS의 AIR 응용 프로그램에서의 여러 SWF 패키지화 및 로드

ProLoader 및 ProLoaderInfo 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상 및 Flash Professional CS5.5 필요

Flash Professional CS5.5에는 RSL(런타임 공유 라이브러리) 미리 로드를 지원하기 위해 fl.display.ProLoader 및 fl.display.ProLoaderInfo 클래스가 도입되었습니다. 이러한 클래스는 flash.display.Loader 및 flash.display.LoaderInfo 클래스를 미러링하지만 로딩 환경의 일관성이 더 뛰어납니다.

특히, ProLoader는 TLF(Text Layout Framework)과 RSL 미리 로드를 함께 사용하는 SWF 파일을 로드하는 데 도움이 됩니다. 다른 SWF 파일 또는 SWZ 파일(예: TLF)을 미리 로드하는 SWF 파일을 런타임에 사용하려면 내부 전용 SWF 래퍼 파일이 있어야 합니다. SWF 래퍼 파일로 인해 레이어가 추가로 복잡해지면 원치 않는 비헤이비어가 발생할 수 있는데, ProLoader는 이러한 복잡성을 해결하여 이들 파일을 일반적인 SWF 파일인 것처럼 로드합니다. ProLoader 클래스에서 사용되는 솔루션은 사용자에게 투명한 방식으로 실행되며, 솔루션을 사용하기 위해 ActionScript에서 특별히 처리할 필요가 없습니다. 또한 ProLoader는 일반적인 SWF 내용을 올바르게 로드합니다.

Flash Professional CS 5.5 이상에서는 모든 Loader 클래스를 대신하여 ProLoader 클래스를 사용해도 무방합니다. 그런 다음에는 ProLoader가 필요한 ActionScript 기능에 액세스할 수 있도록 응용 프로그램을 Flash Player 10.2 이상으로 내보냅니다. ActionScript 3.0을 지원하는 이전 버전의 Flash Player를 대상으로 지정하는 경우에도 ProLoader를 사용할 수 있지만, ProLoader의 기능을 최대한 활용하려면 Flash Player 10.2 이상과 함께 사용해야 합니다. Flash Professional CS5.5 이상에서 TLF를 사용할 경우 항상 ProLoader를 사용하십시오. Flash Professional 이외의 환경에서는 ProLoader를 사용할 필요가 없습니다.

중요: Flash Professional CS5.5 이상에서 제작된 SWF 파일의 경우에는 flash.display.Loader 및 flash.display.LoaderInfo 대신 항상 fl.display.ProLoader 및 fl.display.ProLoaderInfo 클래스를 사용해도 됩니다.

ProLoader 클래스를 통해 해결되는 문제

ProLoader 클래스는 기존 Loader 클래스로 처리하지 못하는 문제를 해결합니다. 이러한 문제는 TLF 라이브러리의 RSL 미리 로드로 인해 발생하며, 특히 Loader 객체를 사용하여 다른 SWF 파일을 로드하는 SWF 파일에 적용됩니다. 해결되는 문제는 다음과 같습니다.

- **로드하는 파일과 로드되는 파일 간의 스크립팅이 예상대로 작동하지 않습니다.** ProLoader 클래스는 로드하는 SWF 파일을 로드되는 SWF 파일의 부모로 자동 설정합니다. 따라서 로드하는 SWF 파일의 통신 내용이 로드되는 SWF 파일로 직접 전달됩니다.
- **SWF 응용 프로그램이 로드 프로세스를 관리해야 합니다.** 이렇게 하려면 added, removed, addedToStage 및 removedFromStage 같은 추가 이벤트를 구현해야 합니다. 응용 프로그램이 Flash Player 10.2 이상을 대상으로 하는 경우 ProLoader는 이러한 추가 작업을 수행할 필요가 없도록 합니다.

Loader 대신 ProLoader를 사용하기 위한 코드 업데이트

ProLoader는 Loader 클래스를 미러링하기 때문에 코드에서 두 클래스를 손쉽게 전환할 수 있습니다. 다음 예제는 기존 코드를 업데이트하여 새 클래스를 사용하는 방법을 보여 줍니다.

```
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!');
}
```

이 코드는 다음과 같이 ProLoader를 사용하도록 업데이트할 수 있습니다.

```
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!');
}
```

표시 객체 예제: SpriteArranger

Flash Player 9 이상, Adobe AIR 1.0 이상

SpriteArranger 샘플 응용 프로그램은 **ActionScript 3.0** 학습에서 별도로 설명한 Geometric Shapes 샘플 응용 프로그램을 바탕으로 합니다.

SpriteArranger 샘플 응용 프로그램에서는 표시 객체를 처리하는 다음과 같은 다양한 개념을 설명합니다.

- 표시 객체 클래스 확장
- 표시 목록에 객체 추가
- 표시 객체 레이어 및 표시 객체 컨테이너 작업
- 표시 객체 이벤트에 응답
- 표시 객체의 속성 및 메서드 사용

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. SpriteArranger 응용 프로그램 파일은 Examples/SpriteArranger 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
SpriteArranger.mxml 또는 SpriteArranger.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/SpriteArranger/CircleSprite.as	화면에 원을 렌더링하는 Sprite 객체 유형을 정의하는 클래스입니다.
com/example/programmingas3/SpriteArranger/DrawingCanvas.as	GeometricSprite 객체를 포함하는 표시 객체 컨테이너인 canvas를 정의하는 클래스입니다.
com/example/programmingas3/SpriteArranger/SquareSprite.as	화면에 사각형을 렌더링하는 Sprite 객체 유형을 정의하는 클래스입니다.
com/example/programmingas3/SpriteArranger/TriangleSprite.as	화면에 삼각형을 렌더링하는 Sprite 객체 유형을 정의하는 클래스입니다.
com/example/programmingas3/SpriteArranger/GeometricSprite.as	화면 상의 모양을 정의하는 데 사용되는 Sprite 객체를 확장하는 클래스로, CircleSprite, SquareSprite 및 TriangleSprite가 이 클래스를 확장합니다.
com/example/programmingas3/geometricshapes/IGeometricShape.as	모든 기하학적 모양 클래스에서 구현될 메서드를 정의하는 기본 인터페이스입니다.
com/example/programmingas3/geometricshapes/IPolygon.as	변이 여러 개인 기하학적 모양 클래스에서 구현될 메서드를 정의하는 인터페이스입니다.
com/example/programmingas3/geometricshapes/RegularPolygon.as	동일한 길이의 변이 모양의 중심을 따라 대칭적으로 배치된 기하학적 모양 유형입니다.
com/example/programmingas3/geometricshapes/Circle.as	원을 정의하는 기하학적 모양 유형입니다.

파일	설명
com/example/programmingas3/geometricshapes/EquilateralTriangle.as	모든 변의 길이가 같은 삼각형을 정의하는 RegularPolygon의 하위 클래스입니다.
com/example/programmingas3/geometricshapes/Square.as	네 변의 길이가 모두 같은 사각형을 정의하는 RegularPolygon의 하위 클래스입니다.
com/example/programmingas3/geometricshapes/GeometricShapeFactory.as	지정된 모양 유형과 크기로 모양을 만드는 "factory 메서드"가 포함된 클래스입니다.

SpriteArranger 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

SpriteArranger 응용 프로그램을 사용하여 화면 상의 "캔바스"에 다양한 표시 객체를 추가할 수 있습니다.

DrawingCanvas 클래스는 사용자가 화면에 모양을 추가할 수 있는 드로잉 영역, 표시 객체 컨테이너 유형 등을 정의합니다. 화면에 추가할 수 있는 모양은 GeometricSprite 클래스의 하위 클래스 중 하나의 인스턴스입니다.

DrawingCanvas 클래스

Flex에서 Container 객체에 추가된 모든 자식 표시 객체는 mx.core.UIComponent 클래스에서 파생된 클래스여야 합니다. 이 응용 프로그램은 SpriteArranger.mxml 파일의 MXML 코드에 정의된 대로 DrawingCanvas 클래스의 인스턴스를 mx.containers.VBox 객체의 자식으로 추가합니다. 이 상속은 다음과 같이 DrawingCanvas 클래스 선언에 정의되어 있습니다.

```
public class DrawingCanvas extends UIComponent
```

UIComponent 클래스는 DisplayObject, DisplayObjectContainer 및 Sprite 클래스로부터 상속되며, DrawingCanvas 클래스의 코드는 해당 클래스의 메서드와 속성을 사용합니다.

DrawingCanvas 클래스는 Sprite 클래스를 확장하며 이 상속은 다음과 같이 DrawingCanvas 클래스 선언에 정의되어 있습니다.

```
public class DrawingCanvas extends Sprite
```

Sprite 클래스는 DisplayObjectContainer 및 DisplayObject 클래스의 하위 클래스이며, DrawingCanvas 클래스는 해당 클래스의 메서드 및 속성을 사용합니다.

DrawingCanvas() 생성자 메서드는 Rectangle 객체인 bounds를 설정합니다. 이 속성은 나중에 캔바스의 윤곽을 그릴 때 사용됩니다. 그런 다음 initCanvas() 메서드를 다음과 같이 호출합니다.

```
this.bounds = new Rectangle(0, 0, w, h);
initCanvas(fillColor, lineColor);
```

다음 예제에서 보듯이 initCanvas() 메서드는 생성자 함수에 인수로 전달된 DrawingCanvas 객체의 다양한 속성을 정의합니다.

```
this.lineColor = lineColor;
this.fillColor = fillColor;
this.width = 500;
this.height = 200;
```

그런 다음 initCanvas() 메서드는 drawBounds() 메서드를 호출합니다. 이 메서드는 DrawingCanvas 클래스의 graphics 속성을 사용하여 캔바스를 그립니다. graphics 속성은 Shape 클래스에서 상속됩니다.

```
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();
```

응용 프로그램과의 사용자 상호 작용을 기반으로 DrawingCanvas 클래스의 다음과 같은 추가 메서드를 호출합니다.

- addShape() 및 describeChildren() 메서드(185페이지의 "캔바스에 표시 객체 추가" 참조)
- moveToBack(), moveDown(), moveToFront() 및 moveUp() 메서드(187페이지의 "표시 객체 레이어 다시 정렬" 참조)
- onMouseUp() 메서드(187페이지의 "표시 객체 클릭 및 드래그" 참조)

GeometricSprite 클래스 및 하위 클래스

사용자가 캔바스에 추가할 수 있는 각 표시 객체는 GeometricSprite 클래스의 다음 하위 클래스 중 하나의 인스턴스입니다.

- CircleSprite
- SquareSprite
- TriangleSprite

GeometricSprite 클래스는 flash.display.Sprite 클래스를 확장합니다.

```
public class GeometricSprite extends Sprite
```

GeometricSprite 클래스는 모든 GeometricSprite 객체에 공통되는 여러 속성을 포함합니다. 이러한 속성은 생성자 함수에서 함수에 전달된 매개 변수를 기반으로 설정됩니다. 예를 들면 다음과 같습니다.

```
this.size = size;
this.lineColor = lColor;
this.fillColor = fColor;
```

GeometricSprite 클래스의 geometricShape 속성은 모양의 시각적 속성이 아니라 수학적 속성을 정의하는 IGeometricShape 인터페이스를 정의합니다. IGeometricShape 인터페이스를 구현하는 클래스는 **ActionScript 3.0 학습**에서 설명한 GeometricShapes 샘플 응용 프로그램에 정의되어 있습니다.

GeometricSprite 클래스는 drawShape() 메서드를 정의합니다. 이 메서드는 GeometricSprite의 각 하위 클래스에서 세부적으로 재정의됩니다. 자세한 내용은 다음에 이어지는 "캔바스에 표시 객체 추가" 단원을 참조하십시오.

또한 GeometricSprite 클래스는 다음과 같은 메서드를 제공합니다.

- onMouseDown() 및 onMouseUp() 메서드(187페이지의 "표시 객체 클릭 및 드래그" 참조)
- showSelected() 및 hideSelected() 메서드(187페이지의 "표시 객체 클릭 및 드래그" 참조)

캔바스에 표시 객체 추가

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 [모양 추가] 버튼을 클릭하면 응용 프로그램이 DrawingCanvas 클래스의 addShape() 메서드를 호출합니다. 그러면 다음 예제에서 보듯이 GeometricSprite 하위 클래스 중 하나의 해당 생성자 함수가 호출되어 새 GeometricSprite가 인스턴스화됩니다.

```
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);
}
```

각 생성자 메서드는 drawShape() 메서드를 호출합니다. 이 메서드는 Sprite 클래스에서 상속되는 클래스의 graphics 속성을 사용하여 해당 벡터 그래픽을 그립니다. 예를 들어, CircleSprite 클래스의 drawShape() 메서드는 다음과 같은 코드를 포함합니다.

```
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);
```

addShape() 함수의 끝에서 두 번째 행은 DisplayObject 클래스에서 상속되는 표시 객체의 alpha 속성을 설정하므로, 캔바스에 추가된 각 표시 객체는 약간 투명하여 그 뒤에 있는 객체를 볼 수 있습니다.

addChild() 메서드의 마지막 행은 이미 표시 목록에 있는 DrawingCanvas 클래스 인스턴스의 자식 목록에 새 표시 객체를 추가합니다. 그러면 새 표시 객체가 스테이지에 표시됩니다.

응용 프로그램의 인터페이스에는 selectedSpriteTxt와 outputTxt의 두 텍스트 필드가 포함됩니다. 이러한 텍스트 필드의 텍스트 속성은 캔바스에 추가되거나 사용자가 선택한 GeometricSprite 객체에 대한 정보로 업데이트됩니다. GeometricSprite 클래스는 toString() 메서드를 다음과 같이 재정의하여 이 정보 보고 작업을 처리합니다.

```
public override function toString():String
{
    return this.shapeType + " of size " + this.size + " at " + this.x + ", " + this.y;
}
```

shapeType 속성은 각 GeometricSprite 하위 클래스의 생성자 메서드에 있는 해당 값으로 설정됩니다. 예를 들어, toString() 메서드는 DrawingCanvas 인스턴스에 최근에 추가된 CircleSprite 인스턴스에 대해 다음 값을 반환할 수 있습니다.

```
Circle of size 50 at 0, 0
```

DrawingCanvas 클래스의 describeChildren() 메서드는 DisplayObjectContainer 클래스에서 상속되는 numChildren 속성을 사용하여 for 루프에 대한 한도를 설정함으로써 캔바스의 자식 목록을 반복합니다. 다음과 같이 각 자식을 나열하는 문자열이 생성됩니다.

```
var desc:String = "";
var child:DisplayObject;
for (var i:int=0; i < this.numChildren; i++)
{
child = this.getChildAt(i);
desc += i + ": " + child + '\n';
}
```

결과 문자열은 outputTxt 텍스트 필드의 text 속성을 설정하는 데 사용됩니다.

표시 객체 클릭 및 드래그

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 GeometricSprite 인스턴스를 클릭하면 응용 프로그램은 onMouseDown() 이벤트 핸들러를 호출합니다. 다음과 같이 이 이벤트 핸들러는 GeometricSprite 클래스의 생성자 함수에서 마우스 누름 이벤트를 수신하도록 설정됩니다.

```
this.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
```

그런 다음 onMouseDown() 메서드는 GeometricSprite 객체의 showSelected() 메서드를 호출합니다. 객체에 대해 이 메서드가 처음으로 호출된 경우 이 메서드는 selectionIndicator라는 새 Shape 객체를 만들고 Shape 객체의 graphics 속성을 사용하여 다음과 같이 빨간색의 강조 표시된 사각형을 그립니다.

```
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);
```

onMouseDown() 메서드가 처음으로 호출된 경우가 아니면 이 메서드는 selectionIndicator 모양의 visible 속성(DisplayObject 클래스에서 상속됨)을 간단히 다음과 같이 설정합니다.

```
this.selectionIndicator.visible = true;
```

hideSelected() 메서드는 visible 속성을 false로 설정하여 이전에 선택된 객체의 selectionIndicator 모양을 숨깁니다.

또한 onMouseDown() 이벤트 핸들러 메서드는 Sprite 클래스에서 상속되는 startDrag() 메서드를 호출합니다. 이 메서드에는 다음과 같은 코드가 포함되어 있습니다.

```
var boundsRect:Rectangle = this.parent.getRect(this.parent);
boundsRect.width -= this.size;
boundsRect.height -= this.size;
this.startDrag(false, boundsRect);
```

사용자는 boundsRect 사각형에 의해 설정된 경계 내에서 선택된 객체를 캔바스 주위로 드래그할 수 있습니다.

마우스 버튼을 놓으면 mouseUp 이벤트가 전달됩니다. DrawingCanvas의 생성자 메서드는 다음과 같은 이벤트 리스너를 설정합니다.

```
this.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
```

이 이벤트 리스너는 개별 GeometricSprite 객체가 아니라 DrawingCanvas 객체에 대해 설정됩니다. GeometricSprite 객체가 드래그될 때 마우스 버튼을 놓으면 다른 표시 객체(다른 GeometricSprite 객체) 뒤에서 드래그가 중단될 수 있기 때문입니다. 전경의 표시 객체는 마우스 놓음 이벤트를 수신하지만 사용자가 드래그 중인 표시 객체는 이 이벤트를 수신하지 못합니다. DrawingCanvas 객체에 리스너를 추가하면 이벤트가 항상 처리되도록 할 수 있습니다.

onMouseUp() 메서드는 GeometricSprite 객체의 onMouseUp() 메서드를 호출하고, 이 메서드는 GeometricSprite 객체의 stopDrag() 메서드를 호출하게 됩니다.

표시 객체 레이어 다시 정렬

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램의 사용자 인터페이스에는 Move Back, Move Down, Move Up, Move to Front라는 버튼이 있습니다. 사용자가 이 버튼 중 하나를 클릭하면 응용 프로그램은 DrawingCanvas 클래스의 해당 메서드(moveToBack(), moveDown(), moveUp() 또는 moveToFront())를 호출합니다. 예를 들어, moveToBack() 메서드는 다음과 같은 코드를 포함합니다.

```
public function moveToBack(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, 0);
    }
}
```

이 메서드는 DisplayObjectContainer 클래스에서 상속되는 setChildIndex() 메서드를 사용하여 DrawingCanvas 인스턴스(this)의 자식 목록에서 인덱스 위치 0에 표시 객체를 배치합니다.

moveDown() 메서드는 DrawingCanvas 인스턴스의 자식 목록에서 표시 객체의 인덱스 위치가 1씩 감소된다는 점을 제외하고는 유사하게 작동합니다.

```
public function moveDown(shape:GeometricSprite):void
{
    var index:int = this.getChildIndex(shape);
    if (index > 0)
    {
        this.setChildIndex(shape, index - 1);
    }
}
```

moveUp() 및 moveToFront() 메서드는 moveToBack() 및 moveDown() 메서드와 유사하게 작동합니다.

11장: 기하 도형을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.geom 패키지에는 점, 사각형, 변형 행렬 등의 기하학적 객체를 정의하는 클래스가 포함되어 있습니다. 이러한 클래스를 사용하여 다른 클래스에 사용되는 객체의 속성을 정의할 수 있습니다.

기타 도움말 항목
flash.geom 패키지

기하 도형의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.geom 패키지에는 점, 사각형, 변형 행렬 등의 기하학적 객체를 정의하는 클래스가 포함되어 있습니다. 이러한 클래스는 자체적인 기능이 있는 것은 아니지만 다른 클래스에서 사용되는 객체 속성을 정의하는 데 사용됩니다.

모든 기하 도형 클래스는 화면에 표시되는 위치가 2차원 평면으로 나타난다는 개념을 기본으로 합니다. 화면을 가로축(x)과 세로축(y)이 있는 평면 그래프로 생각할 수 있습니다. 화면의 모든 위치(또는 점)는 x, y 값 쌍, 즉 위치 좌표로 나타낼 수 있습니다.

스테이지를 포함한 모든 표시 객체에는 고유한 좌표 공간이 있습니다. 좌표 공간은 자식 표시 객체, 드로잉 등의 위치를 구성하기 위한 객체 고유의 그래프입니다. 원점은 좌표 위치 0, 0(x 축과 y 축이 만나는 지점)에 있으며 표시 객체의 왼쪽 위 모서리에 배치됩니다. 스테이지에서는 이 원점 위치가 항상 적용되지만 다른 표시 객체의 경우에는 그렇지 않을 수도 있습니다. x 축의 값은 오른쪽으로 갈수록 커지며, 왼쪽으로 갈수록 작아집니다. 원점의 왼쪽 위치의 경우 x 좌표가 음수입니다. 하지만 기존의 좌표계와는 달리 y 축의 Flash 런타임 좌표 값은 화면 아래로 갈수록 커지고 화면 위로 갈수록 작아집니다. 원점 위의 값은 음수 y 좌표 값을 가집니다. 스테이지의 왼쪽 위 모서리가 좌표 공간의 원점이므로, 스테이지에서 대부분의 객체는 x 좌표가 0보다 크고 스테이지 폭보다 작습니다. 또한 동일 객체는 y 좌표가 0보다 크고 스테이지 높이보다 작습니다.

Point 클래스 인스턴스를 사용하여 좌표 공간에 있는 각각의 점을 나타낼 수 있습니다. 또한 Rectangle 인스턴스를 만들어 좌표 공간을 구성하는 사각형 영역을 나타낼 수 있습니다. 고급 사용자의 경우, Matrix 인스턴스를 사용하여 표시 객체에 여러 가지 복잡한 변형을 적용할 수도 있습니다. 회전, 위치, 배율 변경 등 상당수의 간단한 변형은 객체 속성을 사용하여 표시 객체에 직접 적용할 수 있습니다. 표시 객체 속성을 사용한 변형 적용에 대한 자세한 내용은 156페이지의 “표시 객체 조작”을 참조하십시오.

중요한 개념 및 용어
다음 참조 목록에는 중요한 기하 용어가 정리되어 있습니다.

직교 좌표 좌표는 일반적으로 숫자 쌍(예: 5, 12 또는 17, -23)으로 기록됩니다. 두 숫자는 각각 x 좌표와 y 좌표를 나타냅니다.

좌표 공간 표시 객체에 포함된 좌표의 그래프이며 해당 객체의 자식 요소가 배치됩니다.

원점 좌표 공간에서 x 축과 y 축이 만나는 지점입니다. 이 점의 좌표는 0, 0입니다.

점 좌표 공간에서의 한 위치를 나타냅니다. ActionScript에 사용되는 2D 좌표계에서 점은 x 축과 y 축(지점의 좌표)에 따른 위치로 정의됩니다.

등록 포인트 표시 객체에서 좌표 공간의 원점(0, 0 좌표)입니다.

크기 조절 객체의 원래 크기에 비례한 크기를 나타냅니다. 동사로 '크기 조절'이라고도 하는데, 이 경우에는 객체를 늘리거나 줄여 크기를 변경한다는 의미를 가집니다.

평행 이동 점의 좌표를 한 좌표 공간에서 다른 좌표 공간으로 변경합니다.

변형 객체 회전, 크기 변경, 모양 기울이기 또는 왜곡, 색상 변경 등 그래픽의 시각적 특성을 조절합니다.

X 축 ActionScript에서 사용하는 2차원 좌표계의 수평 축입니다.

Y 축 ActionScript에서 사용하는 2차원 좌표계의 수직 축입니다.

Point 객체 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

Point 객체는 직교 좌표 쌍을 정의합니다. 이 객체는 2차원 좌표계에서의 위치를 나타냅니다. 여기에서 **x**는 가로 축을 나타내고 **y**는 세로 축을 나타냅니다.

Point 객체를 정의하려면 다음과 같이 x 및 y 속성을 설정해야 합니다.

```
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;
```

두 점 사이의 거리 확인

Flash Player 9 이상, Adobe AIR 1.0 이상

Point 클래스의 distance() 메서드를 사용하면 좌표 공간에서 두 점 사이의 거리를 확인할 수 있습니다. 예를 들어 다음 코드에서는 같은 표시 객체 컨테이너에 있는 두 표시 객체(circle1과 circle2)의 등록 포인트 사이의 거리를 확인합니다.

```
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);
```

좌표 공간 평행 이동

Flash Player 9 이상, Adobe AIR 1.0 이상

두 표시 객체가 서로 다른 표시 객체 컨테이너에 있는 경우에는 객체가 서로 다른 좌표 공간에 있을 수 있습니다. 이런 경우 DisplayObject 클래스의 localToGlobal() 메서드를 사용하여 좌표를 스테이지의 전역 좌표 공간으로 평행 이동할 수 있습니다. 예를 들어 다음 코드에서는 서로 다른 표시 객체 컨테이너에 있는 두 표시 객체(circle1과 circle2)의 등록 포인트 사이의 거리를 확인합니다.

```
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);
```

마찬가지로 스테이지의 특정 지점으로부터 target이라는 표시 객체의 등록 지점 거리를 확인하려면 DisplayObject 클래스의 localToGlobal() 메서드를 사용합니다.

```
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);
```

각도 및 거리를 지정하여 표시 객체 이동

Flash Player 9 이상, Adobe AIR 1.0 이상

Point 클래스의 polar() 메서드를 사용하면 표시 객체를 특정 각도 및 거리만큼 이동할 수 있습니다. 예를 들어 다음 코드에서는 myDisplayObject 객체를 60° 방향으로 100 픽셀만큼 이동합니다.

```
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;
```

Point 클래스의 다른 사용 방법

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같은 메서드 및 속성과 함께 Point 객체를 사용할 수 있습니다.

클래스	메서드 또는 속성	설명
DisplayObjectContainer	`areInaccessibleObjectsUnderPoint()getObjectsUnderPoint()`	표시 객체 컨테이너 내의 지점에서 객체 목록을 반환하는 데 사용됩니다.
BitmapData	hitTest()	BitmapData 객체 내의 픽셀 및 히트 검사 대상 지점을 정의하는 데 사용됩니다.
BitmapData	applyFilter() copyChannel() merge() paletteMap() pixelDissolve() threshold()	작업을 정의하는 사각형 위치를 정의하는 데 사용됩니다.
Matrix	deltaTransformPoint() transformPoint()	변환을 적용할 지점을 정의하는 데 사용됩니다.
Rectangle	bottomRight size topLeft	이러한 속성을 정의하는 데 사용됩니다.

Rectangle 객체 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

Rectangle 객체는 사각형 영역을 정의합니다. 이 객체에는 왼쪽 위 모서리의 **x** 및 **y** 좌표로 정의된 위치, width 속성, height 속성 등이 있습니다. 다음과 같이 Rectangle() 생성자 함수를 호출하여 새 Rectangle 객체에 대해 이러한 속성을 정의할 수 있습니다.

```
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);
```

Rectangle 객체의 크기 및 위치 조정

Flash Player 9 이상, Adobe AIR 1.0 이상

다양한 방법을 사용하여 Rectangle 객체의 크기와 위치를 조정할 수 있습니다.

x 및 y 속성을 변경하면 Rectangle 객체의 위치를 직접 조정할 수 있습니다. 이때 Rectangle 객체의 폭이나 높이에는 아무런 영향을 주지 않습니다.

```
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)
```

다음 코드에서와 같이 Rectangle 객체의 left 또는 top 속성을 변경하면 사각형의 위치가 다시 지정됩니다. 사각형의 x 및 y 속성은 각각 left 및 top 속성과 일치합니다. 그러나 Rectangle 객체의 왼쪽 아래 모서리 위치는 변경되지 않고 크기만 조정됩니다.

```
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)
```

또한 다음 예제와 같이 Rectangle 객체의 bottom 또는 right 속성을 변경하면 왼쪽 위 모서리 위치가 변경되지 않습니다. 대신 사각형 크기만 조정됩니다.

```
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)
```

다음과 같이 offset() 메서드를 사용하여 Rectangle 객체의 위치를 조정할 수도 있습니다.

```
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)
```

offsetPt() 메서드도 비슷한 방식으로 작동하지만 **x** 및 **y** 오프셋 값 대신 Point 객체를 매개 변수로 사용한다는 차이가 있습니다.

dx와 dy라는 두 개의 매개 변수를 포함하는 inflate() 메서드를 사용하여 Rectangle 객체의 크기를 조정할 수도 있습니다. dx 매개 변수는 사각형의 왼쪽 및 오른쪽 측면이 중심으로부터 이동하는 픽셀 수를 나타냅니다. dy 매개 변수는 사각형의 위쪽 및 아래쪽 측면이 중심으로부터 이동하는 픽셀 수를 나타냅니다.

```
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)
```

inflatePt() 메서드도 비슷한 방식으로 작동하지만 dx 및 dy 값 대신 Point 객체를 매개 변수로 사용한다는 차이가 있습니다.

Rectangle 객체의 통합 및 교차 영역 얻기

Flash Player 9 이상, Adobe AIR 1.0 이상

union() 메서드를 사용하면 두 사각형의 경계선에 의해 형성되는 사각형 영역을 얻을 수 있습니다.

```
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)
```

intersection() 메서드를 사용하면 두 사각형의 서로 겹치는 영역에 의해 형성되는 사각형 영역을 얻을 수 있습니다.

```
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)
```

두 사각형이 교차하는지 여부를 확인하려면 intersects() 메서드를 사용합니다. 또한 표시 객체가 스테이지의 특정 영역 내에 있는지 여부를 확인하는 경우에도 intersects() 메서드를 사용할 수 있습니다. 다음 코드 예제에서는 circle 객체를 포함하는 표시 객체 컨테이너의 좌표 공간이 스테이지의 좌표 공간과 같다고 가정합니다. 이 예제에서는 intersects() 메서드를 사용하여 표시 객체 circle이 target1과 target2라는 Rectangle 객체로 정의된 스테이지 특정 영역과 교차하는지 여부를 확인하는 방법을 보여 줍니다.

```
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
```

마찬가지로, 두 표시 객체의 경계 사각형이 교차하는지 여부를 확인하는 경우에도 intersects() 메서드를 사용할 수 있습니다. 표시 객체의 획으로 인해 경계 영역에 추가되는 공간을 포함하도록 하려면 DisplayObject 클래스의 getRect() 메서드를 사용합니다.

Rectangle 객체의 다른 사용 방법

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같은 메서드 및 속성과 함께 Rectangle 객체를 사용할 수 있습니다.

클래스	메서드 또는 속성	설명
BitmapData	applyFilter(), colorTransform(), copyChannel(), copyPixels(), draw(), drawWithQuality(), encode(), fillRect(), generateFilterRect(), getColorBoundsRect(), getPixels(), merge(), paletteMap(), pixelDissolve(), setPixels() 및 threshold()	BitmapData 객체의 영역을 정의하기 위해 일부 매개 변수의 유형으로 사용됩니다.
DisplayObject	getBounds(), getRect(), scrollRect, scale9Grid	속성의 데이터 유형 또는 반환되는 데이터 유형으로 사용됩니다.
PrintJob	addPage()	printArea 매개 변수를 정의하는 데 사용됩니다.
Sprite	startDrag()	bounds 매개 변수를 정의하는 데 사용됩니다.
TextField	getCharBoundaries()	반환값 유형으로 사용됩니다.
Transform	pixelBounds	데이터 유형으로 사용됩니다.

Matrix 객체 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

Matrix 클래스는 특정 좌표 공간의 점을 다른 좌표 공간에 매핑하는 방법을 결정하는 변형 행렬을 나타냅니다. Matrix 객체의 속성을 설정하여 Matrix 객체를 Transform 객체의 matrix 속성에 적용한 다음, 이 Transform 객체를 표시 객체의 transform 속성으로 적용하면 표시 객체에서 다양한 그래픽 변환 작업을 수행할 수 있습니다. 이러한 변환 기능에는 평행 이동(**x**와 **y** 위치 변경), 회전, 크기 조절, 기울이기 등이 있습니다.

Matrix 객체의 속성(a, b, c, d, tx, ty)을 직접 조정하여 행렬을 정의할 수도 있지만 createBox() 메서드를 사용하면 더 쉽습니다. 이 메서드에는 결과 행렬의 크기 조절, 회전 및 평행 이동 효과를 직접 정의할 수 있는 매개 변수가 포함되어 있습니다. 예를 들어 다음 코드를 사용하면 가로 방향으로 2.0배 크기 조절, 세로 방향으로 3.0배 크기 조절, 45° 각도로 회전, 오른쪽으로 10픽셀 이동 및 아래쪽으로 20픽셀 이동하는 Matrix 객체가 만들어집니다.

```
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);
```

scale(), rotate() 및 translate() 메서드를 사용하여 Matrix 객체의 크기 조절, 회전 및 평행 이동 효과를 조정할 수도 있습니다. 이러한 메서드는 기존 Matrix 객체의 값을 결합합니다. 예를 들어 다음 코드에서는 scale() 메서드와 rotate() 메서드가 두 번 호출되므로 Matrix 객체가 4배로 확대되고 60° 회전합니다.

```
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;
```

Matrix 객체에 기울이기 변환을 적용하려면 b 또는 c 속성을 조정해야 합니다. b 속성을 조정하면 행렬이 세로 방향으로 기울어지고 c 속성을 조정하면 행렬이 가로 방향으로 기울어집니다. 다음 코드에서는 myMatrix라는 Matrix 객체를 세로 방향, 2의 배율로 기울입니다.

```
var skewMatrix:Matrix = new Matrix();
skewMatrix.b = Math.tan(2);
myMatrix.concat(skewMatrix);
```

표시 객체의 transform 속성에 Matrix 변환을 적용할 수 있습니다. 예를 들어 다음 코드에서는 myDisplayObject라는 표시 객체에 행렬 변환을 적용합니다.

```
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;
```

첫 번째 행에서는 Matrix 객체를 myDisplayObject 표시 객체에서 사용하는 기존 변형 행렬로 설정합니다(myDisplayObject 표시 객체에 대한 transformation 속성의 matrix 속성). 따라서 사용자가 호출하는 Matrix 클래스 메서드에는 표시 객체의 기존 위치, 크기 조절 및 회전 효과가 누적 적용됩니다.

참고: flash.geometry 패키지에는 ColorTransform 클래스도 포함되어 있습니다. 이 클래스는 Transform 객체의 colorTransform 속성을 설정하는 데 사용되지만 어떠한 기하학적 변환도 적용되지 않으므로 여기에서는 자세히 설명하지 않습니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 ColorTransform 클래스를 참조하십시오.

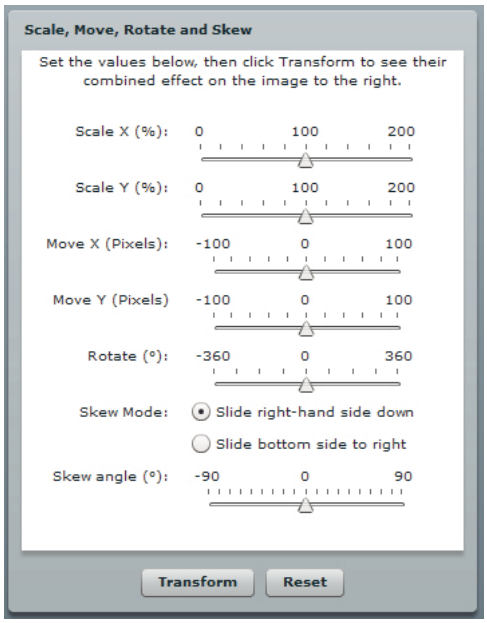
기하 도형 예제: 표시 객체에 행렬 변환 적용

Flash Player 9 이상, Adobe AIR 1.0 이상

DisplayObjectTransformer 샘플 응용 프로그램에서는 다음과 같이 Matrix 클래스를 사용하여 표시 객체를 변환하는 다양한 기능을 보여 줍니다.

- 표시 객체 회전
- 표시 객체 크기 조절
- 표시 객체 평행 이동(위치 변경)
- 표시 객체 기울이기

이 응용 프로그램에는 다음과 같이 행렬 변환 매개 변수를 조정할 수 있는 인터페이스가 있습니다.



사용자가 [Transform] 버튼을 클릭하면 응용 프로그램에서 적절한 변환을 적용합니다.

원본 표시 객체와 -45° 회전 및 **50%** 축소 효과가 적용된 표시 객체

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. DisplayObjectTransformer 응용 프로그램 파일은 Samples/DisplayObjectTransformer 폴더에 있으며 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
DisplayObjectTransformer.mxml 또는 DisplayObjectTransformer.fla	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/geometry/MatrixTransformer.as	행렬 변환을 적용하는 데 필요한 메서드가 포함된 클래스입니다.
img/	응용 프로그램에 사용되는 샘플 이미지 파일이 포함된 디렉토리입니다.

MatrixTransformer 클래스 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

MatrixTransformer 클래스에는 Matrix 객체의 기하학적 변환을 적용하는 정적 메서드가 포함되어 있습니다.

transform() 메서드

transform() 메서드에는 다음 항목에 대한 각 매개 변수가 포함되어 있습니다.

- sourceMatrix - 이 메서드로 변환하는 입력 행렬
- xScale 및 yScale - **x** 및 **y** 비율 인수
- dx 및 dy - **x** 및 **y** 평행 이동 크기(픽셀)
- rotation - 회전 크기(도)
- skew - 기울이기 인수(백분율)
- skewType - 기울이기 방향("right" 또는 "left")

반환값은 결과 행렬입니다.

transform() 메서드는 클래스의 다음 정적 메서드를 호출합니다.

- skew()
- scale()
- translate()
- rotate()

각 메서드는 변환 효과가 적용된 소스 행렬을 반환합니다.

skew() 메서드

skew() 메서드는 행렬의 b 및 c 속성을 조정하여 행렬을 기울입니다. 선택적 매개 변수인 unit은 기울이기 각도를 정의하는 데 사용되는 단위를 결정하고, 필요한 경우 이 메서드는 angle 값을 라디안으로 변환합니다.

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
```

기울이기 변환을 적용하기 위해 skewMatrix라는 Matrix 객체를 만들어 조정합니다. 처음에는 다음과 같이 단위 행렬입니다.

```
var skewMatrix:Matrix = new Matrix();
```

skewSide 매개 변수는 기울이기 변환을 적용할 변을 결정합니다. 이 매개 변수가 "right"로 설정되면 다음 코드에서는 행렬의 b 속성을 설정합니다.

```
skewMatrix.b = Math.tan(angle);
```

그렇지 않은 경우에는 다음과 같이 Matrix 객체의 c 속성을 조정하여 아래쪽 변이 기울어집니다.

```
skewMatrix.c = Math.tan(angle);
```

그런 다음 아래 예제와 같이 두 행렬을 연결하여 기존 행렬에 결과 기울이기가 적용됩니다.

```
sourceMatrix.concat(skewMatrix);
return sourceMatrix;
```

scale() 메서드

다음 예제에서는 비율 인수가 백분율로 지정되어 있는 경우 scale() 메서드로 비율 인수를 조정하고 행렬 객체의 scale() 메서드를 사용하는 방법을 보여 줍니다.

```
if (percent)
{
    xScale = xScale / 100;
    yScale = yScale / 100;
}
sourceMatrix.scale(xScale, yScale);
return sourceMatrix;
```

translate() 메서드

translate() 메서드는 다음과 같이 행렬 객체의 translate() 메서드를 호출하여 dx 및 dy 평행 이동 인수를 적용합니다.

```
sourceMatrix.translate(dx, dy);
return sourceMatrix;
```

rotate() 메서드

rotate() 메서드는 입력 회전 인수가 도 또는 그래디언트 단위로 지정되어 있는 경우 이를 라디안으로 변환한 다음, 행렬 객체의 rotate() 메서드를 호출합니다.

```
if (unit == "degrees")
{
    angle = Math.PI * 2 * angle / 360;
}
if (unit == "gradients")
{
    angle = Math.PI * 2 * angle / 100;
}
sourceMatrix.rotate(angle);
return sourceMatrix;
```

응용 프로그램에서 MatrixTransformer.transform() 메서드 호출

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램에는 사용자가 변환 매개 변수를 입력할 수 있는 사용자 인터페이스가 포함되어 있습니다. 그런 다음 표시 객체의 transform 속성의 matrix 속성과 함께 이러한 값을 다음과 같이 Matrix.transform() 메서드에 전달합니다.

```
tempMatrix = MatrixTransformer.transform(tempMatrix,
    xScaleSlider.value,
    yScaleSlider.value,
    dxSlider.value,
    dySlider.value,
    rotationSlider.value,
    skewSlider.value,
    skewSide );
```

그런 다음 응용 프로그램이 표시 객체의 transform 속성의 matrix 속성에 반환값을 적용하여 변환을 트리거합니다.

```
img.content.transform.matrix = tempMatrix;
```

12장: 드로잉 API 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

가져온 이미지 및 아트웍도 중요하지만 드로잉 API 기능을 사용하면 ActionScript에서 선과 도형을 그릴 수 있으므로 컴퓨터에서 빈 캔바스처럼 응용 프로그램을 시작하여 원하는 이미지를 만들 수 있습니다. 그래픽을 직접 작성할 수 있는 기능은 응용 프로그램의 무한한 가능성을 열어줍니다. 여기서 다루는 기술을 통해 드로잉 프로그램을 생성하거나 움직이는 대화형 아트를 만들고 사용자 인터페이스 요소를 프로그래밍 방식으로 작성할 수 있습니다.

기타 도움말 항목
flash.display.Graphics

드로잉 API의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

드로잉 API는 ActionScript를 사용하여 벡터 그래픽(선, 곡선, 모양, 채우기, 그래디언트)을 만들고 화면에 이를 표시하는 ActionScript 내장 기능의 이름입니다. flash.display.Graphics 클래스가 이 기능을 제공합니다. 이러한 각 클래스에 정의된 graphics 속성을 사용하여 Shape, Sprite 또는 MovieClip 인스턴스에서 ActionScript로 그림을 그릴 수 있습니다. 이러한 각 클래스의 graphics 속성은 실제로는 Graphics 클래스의 인스턴스입니다.

코드를 사용한 드로잉에 이제 막 입문한 사용자를 위해 Graphics 클래스에는 원, 타원, 사각형, 모서리가 둥근 사각형과 같은 일반 도형을 쉽게 그릴 수 있는 몇 가지 메서드가 포함되어 있습니다. 이러한 도형은 빈 선이나 채워진 모양으로 그릴 수 있습니다. 고급 기능을 필요로 하는 사용자를 위해 Graphics 클래스에는 선과 2차 베지어 곡선을 그릴 수 있는 메서드도 포함되어 있습니다. 이러한 메서드를 Math 클래스의 삼각 함수와 함께 사용하여 필요한 모양을 만들 수 있습니다.

Flash Player 10 및 Adobe AIR 1.5 이상 버전 같은 Flash 런타임에는 단일 명령으로 전체 모양을 프로그래밍 방식으로 그리는 데 사용할 수 있는 추가 드로잉 API가 포함되었습니다. "드로잉 API 사용의 기초"에서 다루는 Graphics 클래스 및 작업에 익숙해진 사용자는 212페이지의 “드로잉 API의 고급 사용”에서 이러한 드로잉 API 기능에 대해 자세히 배울 수 있습니다.

중요한 개념 및 용어

다음 참조 목록에는 드로잉 API 사용 시 사용되는 중요한 용어가 정리되어 있습니다.

앵커 포인트 2차 베지어 곡선의 두 끝점 중 하나입니다.

제어 포인트 2차 베지어 곡선의 방향 및 굴곡 정도를 정의하는 점입니다. 곡선이 제어 포인트에 닿지는 않지만 제어 포인트를 향하는 곡선이 그려집니다.

좌표 공간 표시 객체에 포함된 좌표의 그래프이며 해당 객체의 자식 요소가 배치됩니다.

채우기 색상선으로 그린 모양에서 단색의 내부 또는 외곽선이 없는 모양 전체를 말합니다.

그래디언트 한 가지 색상에서 하나 이상의 다른 색상으로 점진적으로 변하는 색상입니다(단색의 반대).

점 좌표 공간에서의 한 위치를 나타냅니다. ActionScript에서 사용되는 2차원 좌표계에서는 x축과 y축 위의 해당 위치(점의 좌표)에 의해 점이 정의됩니다.

2차 베지어 곡선 특정 수학 공식으로 정의된 곡선 유형입니다. 이 유형의 곡선에서 곡선의 모양은 앵커 포인트(곡선의 두 끝점)의 위치와 곡선의 방향 및 굴곡 정도를 정의하는 제어 포인트의 위치를 기반으로 계산됩니다.

크기 조절 객체의 원래 크기에 비례한 크기를 나타냅니다. 동사로 '크기 조절'이라고도 하는데, 이 경우에는 객체를 늘리거나 줄여 크기를 변경한다는 의미를 가집니다.

획 색상선으로 그린 모양에서 외곽선 부분 또는 채워지지 않은 모양의 선입니다.

평행 이동 점의 좌표를 한 좌표 공간에서 다른 좌표 공간으로 변경합니다.

X축 ActionScript에서 사용하는 2차원 좌표계의 수평 축입니다.

Y축 ActionScript에서 사용하는 2차원 좌표계의 수직 축입니다.

Graphics 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

각 Shape, Sprite 및 MovieClip 객체에는 Graphics 클래스의 인스턴스인 graphics 속성이 있습니다. Graphics 클래스에는 선, 채우기 및 모양을 그리기 위한 메서드와 속성이 포함됩니다. 내용을 그리기 위한 캔바스로만 표시 객체를 사용하려는 경우에는 Shape 인스턴스를 사용할 수 있습니다. Shape 인스턴스는 Sprite 및 MovieClip 클래스의 추가 기능에 대한 오버헤드가 없으므로 다른 그리기용 표시 객체보다 성능이 뛰어납니다. 그래픽 내용을 그릴 수 있는 표시 객체가 필요하고 이 표시 객체가 다른 표시 객체를 포함하도록 하려는 경우에는 Sprite 인스턴스를 사용할 수 있습니다. 다양한 작업에 사용할 표시 객체를 결정하는 자세한 방법은 155페이지의 “DisplayObject 하위 클래스 선택”을 참조하십시오.

선 및 곡선 그리기

Flash Player 9 이상, Adobe AIR 1.0 이상

Graphics 인스턴스를 사용하여 수행되는 모든 드로잉은 선 및 곡선이 있는 기본 드로잉을 기반으로 합니다. 따라서 모든 ActionScript 드로잉은 다음의 동일한 단계들을 통해 수행되어야 합니다.

- 선 및 채우기 스타일 정의
- 초기 드로잉 위치 설정
- 선, 곡선 및 모양 그리기(선택적으로 드로잉 포인트 이동)
- 채우기 생성 완료(필요한 경우)

선 및 채우기 스타일 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

Shape, Sprite 또는 MovieClip 인스턴스의 graphics 속성을 사용하여 그리려면 드로잉에 사용할 스타일(선 크기 및 색상, 채우기 색상)을 먼저 정의해야 합니다. Adobe® Flash® Professional 또는 다른 드로잉 응용 프로그램의 드로잉 도구를 사용할 때와 마찬가지로, ActionScript를 사용하여 그릴 때는 획이나 채우기 색상을 사용할 수도 있고 사용하지 않을 수도 있습니다. 획 모양은 lineStyle() 또는 lineGradientStyle() 메서드를 사용하여 지정하고, 실선을 만들려면 lineStyle() 메서드를 사용합니다. 이 메서드를 호출할 때 가장 일반적으로 지정하는 값은 처음 세 개의 매개 변수인 선 두께, 색상 및 알파입니다. 예를 들어 다음 코드 행은 myShape라는 Shape가 2픽셀 두께, 빨강(0x990000) 및 불투명도 75%의 선을 그리도록 지시합니다.

```
myShape.graphics.lineStyle(2, 0x990000, .75);
```

알파 매개 변수의 기본값은 1.0(100%)이며, 완전히 불투명한 선을 원하는 경우 이 매개 변수를 해제할 수 있습니다. lineStyle() 메서드는 또한 픽셀 힌트 및 크기 조절 모드를 위한 두 개의 추가 매개 변수를 사용합니다. 이러한 매개 변수 사용에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 Graphics.lineStyle() 메서드에 대한 설명을 참조하십시오.

그래디언트 선을 만들려면 lineGradientStyle() 메서드를 사용합니다. 이 메서드에 대해서는 204페이지의 “그래디언트 선 및 채우기 만들기”에 설명되어 있습니다.

채워진 모양을 만들려면 드로잉을 시작하기 전에 beginFill(), beginGradientFill(), beginBitmapFill() 또는 beginShaderFill() 메서드를 호출합니다. 그 중에서 가장 기본적인 beginFill() 메서드는 채우기 색상과 채우기 색상에 대한 알파 값(선택 사항)이라는 두 개의 매개 변수를 사용합니다. 예를 들어 녹색으로 채워진 모양을 그리는 경우 다음과 같은 코드를 사용할 수 있습니다(myShape라는 객체에서 그린다고 가정).

```
myShape.graphics.beginFill(0x00FF00);
```

채우기 메서드를 호출하면 이전의 모든 채우기가 암시적으로 종료된 후 새 메서드가 시작됩니다. 획 스타일을 지정하는 메서드를 호출하면 이전 획이 대체되지만, 이전에 지정한 채우기는 바뀌지 않으며 그 반대의 경우도 마찬가지입니다.

선 스타일 및 채우기 속성을 지정했으면 다음 단계는 드로잉 시작점을 지정하는 것입니다. Graphics 인스턴스에는 종이 위의 펜촉과 같은 드로잉 포인트가 있습니다. 드로잉 포인트가 있는 위치가 다음 드로잉 액션이 시작되는 지점이 됩니다. 처음에 Graphics 객체는 드로잉 포인트 0에서 시작하는데, 0은 드로잉이 수행되는 객체의 좌표 공간에서의 시작점입니다. 다른 포인트에서 그리기를 시작하려면 moveTo() 메서드를 먼저 호출한 다음 드로잉 메서드 중 하나를 호출하십시오. 이 과정은 종이에서 펜을 들어 다른 위치로 옮기는 것에 견줄 수 있습니다.

드로잉 포인트를 지정했으면 드로잉 메서드인 lineTo()(직선 그리기용) 및 curveTo()(곡선 그리기용)를 연속적으로 호출하여 그리기를 수행하십시오.

 그리는 동안 언제든지 moveTo() 메서드를 호출하여 드로잉 포인트를 다른 새로운 위치로 이동할 수 있습니다.

그리기 과정에서 채우기 색상을 지정한 경우 endFill() 메서드를 호출하여 채우기를 종료할 수 있습니다. 닫힌 모양 그리기가 아닌 경우(즉, endFill()을 호출했을 때 드로잉 포인트가 모양의 시작점에 있지 않은 경우), endFill() 메서드를 호출하면 Flash 런타임이 현재 드로잉 포인트에서 가장 최근의 moveTo() 호출에 지정된 위치까지 직선을 그려 모양을 자동으로 닫습니다. 채우기를 시작했는데 endFill()을 호출하지 않은 경우, beginFill()(또는 다른 채우기 메서드 중 하나)을 호출하면 현재 채우기가 종료되고 새로운 채우기가 시작됩니다.

직선 그리기

Flash Player 9 이상, Adobe AIR 1.0 이상

lineTo() 메서드를 호출하면 Graphics 객체는 현재 드로잉 포인트에서 메서드 호출에서 두 개의 매개 변수로 지정한 좌표까지 지정한 선 스타일을 사용하여 직선을 그립니다. 예를 들어 다음 코드 행은 드로잉 포인트를 100, 100 좌표에 두고 200, 200 좌표까지 선을 그립니다.

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.lineTo(200, 200);
```

다음 예제에서는 높이가 100픽셀인 빨강과 녹색 삼각형을 그립니다.

```
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);
```

곡선 그리기

Flash Player 9 이상, Adobe AIR 1.0 이상

curveTo() 메서드는 2차 베지어 곡선을 그립니다. 이 곡선은 두 개의 포인트(앵커 포인트)를 연결하는 호를 그리며 이 호는 세 번째 포인트(제어 포인트) 방향으로 구부러집니다. Graphics 객체는 현재 드로잉 위치를 첫 번째 앵커 포인트로 사용합니다. curveTo() 메서드를 호출하면 4개의 매개 변수(제어 포인트의 x, y 좌표와 두 번째 앵커 포인트의 x, y 좌표)가 전달됩니다. 예를 들어 다음 코드는 100, 100 포인트에서 시작하여 200, 200 포인트에서 끝나는 곡선을 그립니다. 제어 포인트가 175, 125이므로 오른쪽으로 이동하여 아래로 향하는 곡선이 만들어집니다.

```
myShape.graphics.moveTo(100, 100);
myShape.graphics.curveTo(175, 125, 200, 200);
```

다음 예제에서는 폭 및 높이가 100픽셀인 빨강과 녹색 원형 객체를 그립니다. 2차 베지어 수식의 특성상 완벽한 원이 되지는 않습니다.

```
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);
```

내장 메서드를 사용하여 모양 그리기

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 원, 타원, 사각형, 모서리가 둥근 사각형 등의 일반 모양을 쉽게 그릴 수 있는 몇 가지 메서드가 포함되어 있습니다. Graphics 클래스의 drawCircle(), drawEllipse(), drawRect() 및 drawRoundRect() 메서드가 이에 해당합니다. 이러한 메서드는 lineTo() 및 curveTo() 메서드 대신 사용할 수 있습니다. 하지만 이러한 메서드를 호출하기 전에 선 및 채우기 스타일도 지정해야 합니다.

다음 예제에서는 폭과 높이가 100픽셀인 빨강, 녹색 및 파랑 사각형을 그리는 예제를 다시 구성한 것입니다. 다음 코드는 drawRect() 메서드를 사용하고 채우기 색상의 알파를 50%(0.5)로 지정합니다.

```
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);
```

Sprite 또는 MovieClip 객체에서 graphics 속성으로 만든 드로잉 내용은 항상 해당 객체에 포함된 모든 자식 표시 객체의 뒤에 나타납니다. 또한 graphics 속성 내용은 별도의 표시 객체가 아니므로 Sprite 또는 MovieClip 객체의 자식 목록에 나타나지 않습니다. 예를 들어 다음 Sprite 객체는 graphics 속성을 사용하여 그린 원으로, 해당 자식 표시 객체 목록에 TextField 객체가 있습니다.

```
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);
```

TextField는 graphics 객체로 그린 원형의 위쪽에 표시됩니다.

그래디언트 선 및 채우기 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

graphics 객체는 또한 단색보다 그래디언트를 사용하여 획 및 채우기를 그릴 수 있습니다. 그래디언트 획은 lineGradientStyle() 메서드를 사용하여 만들고, 그래디언트 채우기는 beginGradientFill() 메서드를 사용하여 만듭니다.

두 메서드 모두 동일한 매개 변수를 사용합니다. 처음 4개의 매개 변수인 유형, 색상, 알파, 비율은 필수 항목입니다. 나머지 4개의 매개 변수는 선택 사항이지만 고급 사용자 정의 시 유용합니다.

- 첫 번째 매개 변수는 만들려는 그래디언트 유형을 지정하며 사용할 수 있는 값은 GradientType.LINEAR 또는 GradientType.RADIAL입니다.

- 두 번째 매개 변수는 사용할 색상 값의 배열을 지정합니다. 선형 그래디언트에서는 색상이 왼쪽에서 오른쪽으로 배열되고 방사형 그래디언트에서는 안쪽에서 바깥쪽으로 배열됩니다. 배열 색상의 순서는 그래디언트에서 색상이 그려지는 순서를 나타냅니다.
- 세 번째 매개 변수는 이전 매개 변수의 해당 색상에 대한 알파 투명도 값을 지정합니다.
- 네 번째 매개 변수는 비율, 즉 각 색상이 그래디언트에서 가지는 강도를 지정합니다. 사용할 수 있는 값은 0에서 255까지입니다. 이러한 값은 폭이나 높이를 나타내는 것이 아니라 그래디언트에서의 위치를 나타냅니다. 즉, 0은 그래디언트의 처음을 나타내고 255는 그래디언트의 끝을 나타냅니다. 비율의 배열은 순차적으로 증가해야 하며 두 번째 및 세 번째 매개 변수에 지정된 색상 및 알파 배열과 동일한 항목 수를 가져야 합니다.

다섯 번째 매개 변수인 변형 행렬은 선택 사항이지만 간단한 방법으로 그래디언트의 모양을 효율적으로 제어할 수 있으므로 흔히 사용됩니다. 이 매개 변수는 Matrix 인스턴스를 사용합니다. 그래디언트에 대한 Matrix 객체를 만드는 가장 쉬운 방법은 Matrix 클래스의 createGradientBox() 메서드를 사용하는 것입니다.

그래디언트에 사용할 Matrix 객체 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.display.Graphics 클래스의 beginGradientFill() 및 lineGradientStyle() 메서드를 사용하면 모양에 사용할 그래디언트를 정의할 수 있습니다. 그래디언트를 정의하는 경우 행렬을 이러한 메서드의 매개 변수 중 하나로 제공합니다.

행렬을 정의하는 가장 쉬운 방법은 Matrix 클래스의 createGradientBox() 메서드를 사용하여 그래디언트 정의에 사용되는 행렬을 만드는 것입니다. createGradientBox() 메서드에 전달되는 매개 변수를 사용하여 그래디언트의 크기, 회전 및 위치를 정의합니다. createGradientBox() 메서드는 다음과 같은 매개 변수를 사용합니다.

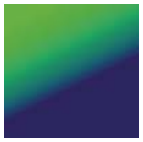
- 그래디언트 상자 폭: 그래디언트가 펼쳐지는 폭(단위: 픽셀)
- 그래디언트 상자 높이: 그래디언트가 펼쳐지는 높이(단위: 픽셀)
- 그래디언트 상자 회전: 그래디언트에 적용될 회전(단위: 라디안)
- 수평 이동: 그래디언트가 수평으로 이동되는 정도(단위: 픽셀)
- 수직 이동: 그래디언트가 수직으로 이동되는 정도(단위: 픽셀)

예를 들어 다음과 같은 특성을 갖는 그래디언트를 검토하십시오.

- GradientType.LINEAR
- ratios 배열이 [0, 255]로 설정된 녹색과 파랑의 두 색상
- SpreadMethod.PAD
- InterpolationMethod.LINEAR_RGB

다음 예제에서는 createGradientBox() 메서드의 rotation 매개 변수만 다르고 다른 모든 설정은 동일한 그래디언트를 보여 줍니다.

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;
	

다음 예제에서는 녹색에서 파란색으로 바뀌는 선형 그래디언트 효과를 보여 줍니다. 이때 createGradientBox() 메서드의 rotation, tx 및 ty 매개 변수만 다르고 다른 모든 설정은 동일합니다.

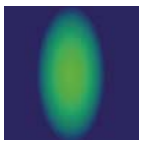
width = 50;
height = 100;
rotation = 0;
tx = 0;
ty = 0;
	

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;
	

다음 예제와 같이 createGradientBox() 메서드의 width, height, tx 및 ty 매개 변수는 **방사형** 그래디언트 채우기의 크기와 위치에도 영향을 줍니다.

width = 50;
height = 100;
rotation = 0;
tx = 25;
ty = 0;
	

다음 코드는 마지막에 보여 준 방사형 그래디언트를 만듭니다.

```
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);
```

그래디언트 채우기의 폭과 높이는 Graphics 객체로 그린 폭이나 높이가 아닌 그래디언트 행렬의 폭과 높이에 따라 결정됩니다. Graphics 객체를 사용하여 그리면 그래디언트 행렬의 해당 좌표에 존재하는 객체가 그려집니다. Graphics 객체의 shape 메서드 중 하나(예: drawRect())를 사용하더라도 그려지는 모양의 크기까지 그래디언트를 확장할 수는 없습니다. 그래디언트의 크기는 그래디언트 행렬 자체에 지정해야 합니다.

다음을 통해 그래디언트 행렬의 차원과 드로잉 자체의 차원 간에 존재하는 시각적 차이를 확인할 수 있습니다.

```
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);
```

이 코드는 빨강, 녹색, 파랑이 똑같이 배분된 동일한 채우기 스타일을 사용하여 세 가지 그래디언트를 그립니다. 그래디언트는 픽셀 폭이 각각 50, 100, 150인 drawRect() 메서드를 사용하여 그려집니다. beginGradientFill() 메서드에 지정된 그래디언트 행렬은 100픽셀의 폭을 사용하여 만들어집니다. 즉, 첫 번째 그래디언트에는 그래디언트 스펙트럼의 반만 포함되고, 두 번째 그래디언트에는 스펙트럼 전체가 포함되며, 세 번째 그래디언트에는 스펙트럼 전체를 포함하면서 추가적으로 50픽셀의 파랑이 오른쪽으로 확장됩니다.

lineGradientStyle() 메서드는 그래디언트를 정의하는 것 외에는 beginGradientFill()과 유사하게 동작하지만 그리기 전에 lineStyle() 메서드를 사용하여 획 두께를 지정해야 합니다. 다음 코드는 빨강, 녹색 및 파랑 그래디언트 획을 가진 상자를 그립니다.

```
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);
```

Matrix 클래스에 대한 자세한 내용은 195페이지의 “Matrix 객체 사용”을 참조하십시오.

드로잉 메서드와 Math 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

Graphics 객체는 원과 사각형을 그리지만 특히 드로잉 메서드를 Math 클래스의 속성 및 메서드와 함께 사용할 경우 보다 복잡한 형태를 그릴 수도 있습니다. Math 클래스에는 일반적인 수학적 비율을 나타내는 상수가 포함되어 있습니다. 예를 들어 원주율에 대한 상수인 Math.PI(약 3.14159265...)가 이에 해당합니다. 또한 Math.sin(), Math.cos() 및 Math.tan() 등의 삼각 함수에 대한 메서드도 포함합니다. 이러한 메서드 및 상수를 사용하여 모양을 그리면 특히 반복 또는 재귀와 함께 사용할 경우 더욱 동적인 시각 효과를 만들 수 있습니다.

Math 클래스의 많은 메서드는 원 치수를 각도 단위가 아닌 라디안 단위로 예상하므로, Math 클래스의 일반적인 용도는 이 두 유형의 단위를 상호 변환하는 것입니다.

```
var degrees = 121;
var radians = degrees * Math.PI / 180;
trace(radians) // 2.111848394913139
```

다음 예제에서는 사인파와 코사인파를 만들어 특정한 값에 대한 Math.sin() 메서드와 Math.cos() 메서드의 차이점을 강조 표시합니다.

```
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);
}
```

드로잉 API를 사용한 애니메이션

Flash Player 9 이상, Adobe AIR 1.0 이상

드로잉 API를 사용하여 내용을 만들면 내용을 한 번만 배치하도록 제한되지 않는다는 장점이 있습니다. 그릴 때 사용하는 변수를 유지하거나 수정하여 드로잉 내용을 수정할 수 있습니다. 특정 프레임 기간 동안 또는 타이머를 사용하여 변수를 변경하거나 다시 그림으로써 애니메이션을 전달할 수 있습니다.

예를 들어 다음 코드는 Event.ENTER_FRAME 이벤트 수신을 통해 각 전달 프레임으로 표시를 변경하고 현재 각도를 증가시켜 graphics 객체를 지우고 업데이트된 위치에서 다시 그리도록 지시합니다.

```
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;
}
```

현저히 다른 결과를 얻기 위한 실험으로 currentDegrees, radius, satelliteRadius 및 해당 코드의 시작 부분에 있는 초기 난수 변수를 수정해 볼 수 있습니다. 예를 들어 radius 변수를 감소시키거나 totalSatellites 변수를 증가시켜 봅니다. 다음은 드로잉 API를 시각적으로 표시하는 방법을 보여 주는 예제로서, 시각적으로는 복잡하지만 생성 원리는 단순합니다.

드로잉 API 예제: Algorithmic Visual Generator

Flash Player 9 이상, Adobe AIR 1.0 이상

Algorithmic Visual Generator 예제에서는 원 궤도를 따라 움직이는 여러 가지 "위성" 또는 원형을 스테이지에 동적으로 그립니다. 살펴본 기능은 다음과 같습니다.

- 드로잉 API를 사용하여 동적 모양을 가진 기본 모양 그리기
- 드로잉에 사용된 속성과 사용자 상호 작용 연결
- 각 프레임에서 스테이지를 지우고 다시 그려서 애니메이션 전달

이전 하위 단원의 예제에서는 Event.ENTER_FRAME 이벤트를 사용하여 단일 "위성"에 애니메이션을 적용했습니다. 이 예제에서는 이를 바탕으로 여러 위성의 시각적 표시를 즉시 업데이트하는 각종 슬라이더의 제어판을 만듭니다. 이 예제는 코드를 외부 클래스로 정형화하고 위성 생성 코드를 루프 내에 포함시켜 각 위성에 대한 참조를 satellites 배열에 저장합니다.

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. 응용 프로그램 파일은 Samples/AlgorithmicVisualGenerator 폴더에 있습니다. 이 폴더에는 다음과 같은 파일이 있습니다.

파일	설명
AlgorithmicVisualGenerator.fla	Flash Professional(FLA) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/algorithmic/AlgorithmicVisualGenerator.as	스테이지에 대한 위성을 드로잉하고 제어판의 이벤트에 응답하여 위성 드로잉에 영향을 미치는 변수를 업데이트하는 등의 기본 응용 프로그램 기능을 제공하는 클래스입니다.
com/example/programmingas3/algorithmic/ControlPanel.as	몇 개의 슬라이더로 사용자 상호 작용을 관리하고 상호 작업 발생 시 이벤트를 전달하는 클래스입니다.
com/example/programmingas3/algorithmic/Satellite.as	중앙 포인트 주위를 궤도로 회전하는 표시 객체를 나타내고 현재 드로잉 상태와 관련된 속성을 포함하는 클래스입니다.

리스너 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램은 먼저 세 개의 리스너를 만듭니다. 첫 번째 리스너는 제어판에서 전달 이벤트를 수신하며 위성 재구성이 필요합니다. 두 번째 리스너는 SWF 파일의 스테이지 크기 변경 사항을 수신합니다. 세 번째 리스너는 SWF 파일의 각 전달 프레임을 수신하고 doEveryFrame() 함수를 사용하여 그리기를 다시 수행합니다.

위성 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

이러한 리스너가 설정되면 build() 함수가 호출됩니다. 이 함수는 먼저 clear() 함수를 호출하여 satellites 배열을 비우고 스테이지에 대한 이전 드로잉을 모두 지웁니다. 제어판이 이와 같은 이벤트를 보낼 때마다(예를 들어 색상 설정이 변경될 때마다) build() 함수를 다시 호출할 수 있으므로 이 과정이 필요합니다. 이 경우 위성을 제거하고 다시 만들어야 합니다.

그런 다음 함수는 생성에 필요한 초기 속성(예: 궤도의 임의 위치에서 시작하는 position 변수, 이 예제에서 위성이 생성되면 변경되지 않는 color 변수)을 설정하여 위성을 생성합니다.

각 위성이 생성되었으면 해당 참조가 satellites 배열에 추가됩니다. doEveryFrame() 함수가 호출되면 해당 함수가 이 배열의 모든 위성에 대해 업데이트됩니다.

위성 위치 업데이트

Flash Player 9 이상, Adobe AIR 1.0 이상

doEveryFrame() 함수는 응용 프로그램의 애니메이션 프로세스의 핵심 요소로서, SWF 파일의 프레임 속도와 동일한 속도로 각 프레임마다 호출됩니다. 그리기 변수는 약간씩 달라지므로 이 함수는 애니메이션 모양을 전달합니다.

함수는 먼저 이전 드로잉을 모두 지우고 배경을 다시 그립니다. 그런 다음 각 위성 컨테이너를 반복하여 각 위성의 position 속성을 증가시키고 제어판의 사용자 상호 작용으로 인해 변경되었을 수 있는 radius 및 orbitRadius 속성을 업데이트합니다. 마지막으로 위성은 Satellite 클래스의 draw() 메서드를 호출하여 새로운 위치로 업데이트됩니다.

카운터 i는 visibleSatellites 변수까지만 증가됩니다. 제어판을 통해 표시되는 위성의 수를 사용자가 제한한 경우 루프의 나머지 위성은 다시 그려지지 않고 숨겨져야 하기 때문입니다. 이러한 경우는 그리기를 담당하는 루프 바로 다음의 루프에서 발생합니다.

doEveryFrame() 함수가 완료되면 visibleSatellites 수가 화면 위치에서 업데이트됩니다.

사용자 상호 작용에 대한 응답

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자 상호 작용은 제어판을 통해 발생하며 ControlPanel 클래스에서 관리합니다. 이 클래스는 각 슬라이더의 개별적인 최소값, 최대값 및 기본값과 함께 리스너를 설정합니다. 사용자가 이러한 슬라이더를 이동하면 changeSetting() 함수가 호출되며 이 함수는 제어판의 속성을 업데이트합니다. 변경 사항으로 인해 표시를 재구성해야 하는 경우 이벤트가 전달되어 기본 응용 프로그램 파일에서 처리됩니다. 제어판 설정이 변경되면 doEveryFrame() 함수는 업데이트된 변수를 사용하여 각 위성을 그립니다.

구체적인 사용자 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

이 예제는 드로잉 API를 사용하여 시각적 표시 생성 방법을 보여 주는 간단한 예제로서, 비교적 간단한 코드 행을 사용하여 꽤 복잡해 보이는 대화형 환경을 만듭니다. 하지만 이 예제를 약간 수정하여 기능을 추가할 수 있습니다. 몇 가지 아이디어는 다음과 같습니다.

- doEveryFrame() 함수는 위성의 색상 값을 증가시킬 수 있습니다.
- doEveryFrame() 함수는 시간 경과에 따라 위성 반경을 축소하거나 확장할 수 있습니다.
- 위성 반경은 원형일 필요가 없습니다. 예를 들어 Math 클래스를 사용하여 사인파에 따라 이동할 수 있습니다.
- 위성은 다른 위성과의 히트 감지를 사용할 수 있습니다.

드로잉 API를 사용하여 Flash 제작 환경에서 시각 효과를 생성함으로써 런타임에 기본 모양을 그릴 수도 있습니다. 하지만 손으로 제작할 수 없는 다양하고 넓은 범위의 시각 효과를 생성하는 데도 사용될 수 있습니다. ActionScript 제작자는 드로잉 API와 약간의 수학 기능을 사용하여 여러 가지 다양한 작품에 생동감을 불어 넣을 수 있습니다.

드로잉 API의 고급 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Player 10 및 Adobe AIR 1.5 이상의 Flash 런타임은 고급 드로잉 기능 집합을 지원합니다. 이러한 런타임의 향상된 드로잉 API는 데이터 집합을 설정하여 모양을 생성하고, 런타임에 모양을 변경하고, 3차원 효과를 만들 수 있도록 이전 릴리스의 드로잉 메서드를 확장합니다. 또한 기존 메서드를 대체 명령으로 통합합니다. 이러한 명령은 벡터 배열 및 열거형 클래스를 활용하여 드로잉 메서드에 데이터 집합을 제공합니다. 벡터 배열을 사용하면 보다 복잡한 모양을 빠르게 렌더링할 수 있을 뿐 아니라 개발자가 동적 모양 렌더링을 위해 런타임에 배열 값을 프로그래밍 방식으로 변경할 수 있습니다.

Flash Player 10에 도입된 드로잉 기능에 대한 자세한 내용은 213페이지의 “패스 그리기”, 214페이지의 “굴곡 규칙 정의”, 216페이지의 “그래픽 데이터 클래스 사용” 및 218페이지의 “drawTriangles() 사용 정보” 단원에서 설명합니다.

ActionScript에서 고급 드로잉 API를 사용하여 다음과 같은 작업을 수행할 수 있습니다.

- Vector 객체를 사용하여 드로잉 메서드를 위한 데이터 저장
- 프로그래밍 방식을 이용해 단일 작업으로 드로잉 모양의 패스 정의
- 굴곡 규칙을 정의하여 겹쳐진 모양이 채워지는 방식 결정

- 표시 객체의 벡터 그래픽 내용 읽기(예: 그래픽 데이터 직렬화 및 저장, 런타임에 스프라이트 시트 생성, 벡터 그래픽 내용의 복사본 그리기)
- 3차원 효과에 삼각형 및 드로잉 메서드 사용

중요한 개념 및 용어

다음 참조 목록에는 이 단원에 사용된 중요한 용어가 포함되어 있습니다.

- 벡터: 데이터 유형이 모두 동일한 값의 배열입니다. Vector 객체는 드로잉 메서드에서 단일 명령으로 선과 모양을 생성하는 데 사용하는 값의 배열을 저장할 수 있습니다. Vector 객체에 대한 자세한 내용은 23페이지의 "인덱스 배열"을 참조하십시오.
- 패스: 하나 이상의 직선 또는 곡선 선분으로 구성됩니다. 각 선분의 시작과 끝은 전선을 제 위치에 고정시켜 주는 핀과 같은 역할을 하는 좌표로 표시됩니다. 패스는 닫힌 패스(예: 원)일 수도 있고 명확한 끝점이 있는 열린 패스(예: 물결선)일 수도 있습니다.
- 굴곡: 렌더러에서 해석되는 패스 방향으로 양수 방향(시계 방향)이나 음수 방향(반시계 방향)입니다.
- GraphicsStroke: 선 스타일을 설정하기 위한 클래스입니다. "stroke"라는 용어는 향상된 드로잉 API에 포함되지 않았지만 자체 fill 속성을 통해 선 스타일을 지정하는 클래스가 새로운 드로잉 API에 포함되었습니다. GraphicsStroke 클래스를 사용하여 선의 스타일을 동적으로 조정할 수 있습니다.
- Fill 객체: 드로잉 명령 Graphics.drawGraphicsData()에 전달되는 flash.display.GraphicsBitmapFill 및 flash.display.GraphicsGradientFill과 같이 표시 클래스를 사용하여 만든 객체입니다. Fill 객체와 향상된 드로잉 명령에서는 Graphics.beginBitmapFill() 및 Graphics.beginGradientFill()을 복제하는 데 보다 객체 지향적인 프로그래밍 방식을 사용합니다.

패스 그리기

Flash Player 10 이상, Adobe AIR 1.5 이상

선 및 곡선 그리기에 대한 단원(201페이지의 "선 및 곡선 그리기" 참조)에서는 단일 선(Graphics.lineTo()) 또는 곡선(Graphics.curveTo())을 그린 다음 해당 선을 다른 지점(Graphics.moveTo())으로 이동하여 모양을 형성하기 위한 명령을 소개했습니다. Graphics.drawPath() 및 Graphics.drawTriangles() 메서드는 매개 변수와 동일한 드로잉 명령을 나타내는 객체 집합을 허용합니다. 이 메서드를 사용하면 일련의 Graphics.lineTo(), Graphics.curveTo() 또는 Graphics.moveTo() 명령을 제공하여 Flash 런타임을 단일 명령문으로 실행할 수 있습니다.

GraphicsPathCommand 열거형 클래스는 드로잉 명령에 해당하는 상수 집합을 정의합니다. Vector 인스턴스에서 래핑되는 이 상수 집합을 Graphics.drawPath() 메서드의 매개 변수로 전달합니다. 그러면 단일 명령을 사용하여 한 모양 전체 또는 몇 개 모양을 렌더링할 수 있습니다. 이러한 메서드에 전달되는 값을 변경하여 기존 모양을 변경할 수도 있습니다.

drawPath() 메서드는 드로잉 명령의 Vector 외에도 각 드로잉 명령의 좌표에 해당하는 좌표 집합이 필요합니다. 좌표(Number 인스턴스)가 포함된 Vector 인스턴스를 만들고 이를 drawPath() 메서드에 두 번째(data) 인수로 전달합니다.

참고: 벡터의 값은 Point 객체가 아니고, 벡터는 일련의 숫자이며 두 개의 숫자로 된 각 그룹이 x/y 좌표 쌍을 나타냅니다.

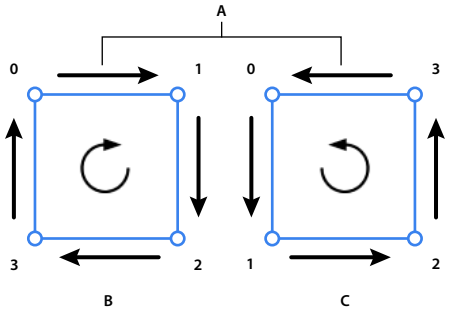
Graphics.drawPath() 메서드는 각 명령을 해당 점 값(두 개 또는 네 개의 숫자로 된 컬렉션)과 대응시켜 Graphics 객체에 패스를 생성합니다.

```
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);
        }
    }
}
```

굴곡 규칙 정의

Flash Player 10 이상, Adobe AIR 1.5 이상

향상된 드로잉 API에서는 패스의 방향을 나타내는 패스 "굴곡"이라는 개념도 도입되었습니다. 패스의 굴곡은 양수(시계 방향)이거나 음수(반시계 방향)입니다. data 매개 변수의 벡터에서 제공하는 좌표를 렌더러에서 해석하는 순서에 따라 굴곡이 결정됩니다.



양수 및 음수 굴곡
A. 그리기 방향을 나타내는 화살표 **B.** 양수 방향(시계 방향) **C.** 음수 방향(반시계 방향)

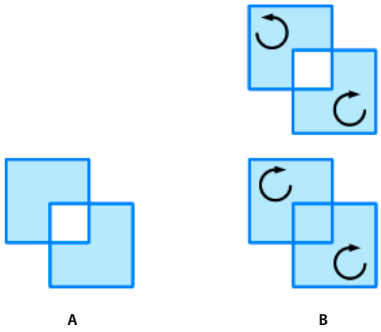
또한 Graphics.drawPath() 메서드에는 "굴곡"이라는 세 번째 선택적 매개 변수가 있습니다.

```
drawPath(commands:Vector.<int>, data:Vector.<Number>, winding:String = "evenOdd"):void
```

이 컨텍스트에서 세 번째 매개 변수는 교차하는 패스에 대한 굴곡 또는 채우기 규칙을 지정하는 문자열이나 상수입니다. 상수 값은 GraphicsPathWinding 클래스에 GraphicsPathWinding.EVEN_ODD 또는 GraphicsPathWinding.NON_ZERO로 정의되어 있습니다. 패스가 교차할 때는 굴곡 규칙이 중요합니다.

짝수-홀수 규칙은 표준 굴곡 규칙으로서 이전 드로잉 API에서 사용되었습니다. 짝수-홀수 규칙은 Graphics.drawPath() 메서드에 대한 기본 규칙이기도 합니다. 짝수-홀수 굴곡 규칙을 사용하면 교차하는 패스가 열린 채우기와 닫힌 채우기 사이에서 교대로 반복됩니다. 채우기 교차 영역이 같은 두 사각형을 그리면 교차 영역이 채워집니다. 일반적으로 인접 영역은 둘 중 하나만 채워집니다.

반대로 0이 아닌 규칙은 굴곡(드로잉 방향)에 따라 교차하는 패스에 정의된 영역을 채울지 여부를 결정합니다. 반대 방향의 굴곡 패스가 교차할 때 정의된 영역은 짝수-홀수 굴곡의 경우와 같이 채워지지 않습니다. 패스의 굴곡이 같은 경우에는 채워지지 않은 영역이 채워집니다.

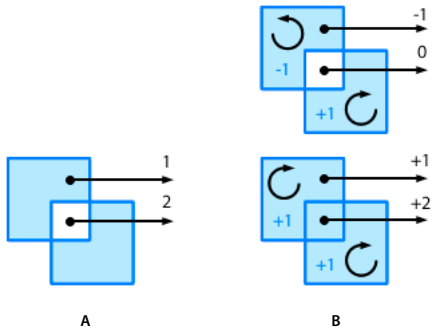


교차하는 영역에 대한 굴곡 규칙
A. 짝수-홀수 굴곡 규칙 **B.** 0이 아닌 굴곡 규칙

굴곡 규칙 이름

Flash Player 10 이상, Adobe AIR 1.5 이상

굴곡 규칙 이름은 이러한 채우기가 관리되는 방식을 정의하는 보다 구체적인 규칙을 나타냅니다. 양수 방향 굴곡 패스에는 값 +1이 지정되고 음수 방향 굴곡 패스에는 값 -1이 지정됩니다. 모양의 막힌 영역 내에 있는 점에서부터 시작하여 무한정으로 확장하여 선을 그립니다. 선이 패스와 교차하는 횟수와 이러한 패스의 결합된 값을 통해 채우기가 결정됩니다. 짝수-홀수 굴곡의 경우에는 선이 패스와 교차하는 횟수가 사용됩니다. 이 횟수가 홀수이면 영역이 채워지고 짝수이면 영역이 채워지지 않습니다. 0이 아닌 굴곡의 경우 패스에 지정된 값이 사용됩니다. 패스의 결합된 값이 0이 아니면 영역이 채워지고 0이면 영역이 채워지지 않습니다.



굴곡 규칙 개수 및 채우기
A. 짝수-홀수 굴곡 규칙 **B.** 0이 아닌 굴곡 규칙

굴곡 규칙 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

이러한 채우기 규칙은 복잡하지만 일부 경우에 필요할 수 있습니다. 예를 들어 별 모양을 그린다고 가정합니다. 표준 짝수-홀수 규칙을 사용할 경우 해당 모양에는 각기 다른 10개의 선이 필요합니다. 0이 아닌 굴곡 규칙을 사용할 경우 이 10개의 선은 다섯 개로 줄어듭니다. 다음은 다섯 개의 선과 0이 아닌 굴곡 규칙으로 별을 그리기 위한 ActionScript입니다.

```
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);
```

다음은 별 모양입니다.

서로 다른 굴곡 규칙을 사용한 별 모양
A. 짝수-홀수 굴곡, 10개의 선 **B.** 짝수-홀수 굴곡, 5개의 선 **C.** 0이 아닌 굴곡, 5개의 선

또한 이미지가 3차원 객체에서 텍스처로 사용되거나 애니메이션되고 겹칠 때는 굴곡 규칙이 더욱 중요해집니다.

그래픽 데이터 클래스 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

향상된 드로잉 API에는 IGraphicsData 인터페이스를 구현하는 flash.display 패키지의 클래스 집합이 포함되어 있습니다. 이 클래스 집합은 드로잉 API의 드로잉 메서드를 나타내는 값 객체(데이터 컨테이너) 역할을 합니다.

IGraphicsData 인터페이스를 구현하는 클래스는 다음과 같습니다.

- GraphicsBitmapFill
- GraphicsEndFill
- GraphicsGradientFill
- GraphicsPath
- GraphicsShaderFill
- GraphicsSolidFill
- GraphicsStroke
- GraphicsTrianglePath

이 클래스를 사용하면 전체 드로잉을 IGraphicsData 유형의 Vector 객체(Vector.<IGraphicsData>)에 저장할 수 있습니다. 그런 다음 그래픽 데이터를 다른 모양 인스턴스의 데이터 소스로 재사용하거나 나중에 사용하도록 드로잉 정보를 저장할 수 있습니다.

각 채우기 스타일에 사용할 수 있는 채우기 클래스는 여러 개가 있지만 획 클래스는 하나만 있습니다. 획 클래스는 채우기 클래스를 사용하여 스타일을 정의하므로 ActionScript에는 획에 대한 IGraphicsData 클래스는 하나만 있습니다. 따라서 모든 획은 실제로 채우기 클래스 및 획 클래스 조합에 따라 정의됩니다. 그렇지 않은 경우 이러한 그래픽 데이터 클래스의 API는 flash.display.Graphics 클래스에서 해당 API가 나타내는 메서드를 미러링합니다.

Graphics 메서드	해당 클래스
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

또한 GraphicsPath 클래스에는 고유한 GraphicsPath.moveTo(), GraphicsPath.lineTo(), GraphicsPath.curveTo(), GraphicsPath.wideLineTo() 및 GraphicsPath.wideMoveTo() 유틸리티 메서드가 있으므로 GraphicsPath 인스턴스에 대한 이러한 명령을 손쉽게 정의할 수 있습니다. 이러한 유틸리티 메서드를 사용하면 명령 및 데이터 값을 직접 정의하거나 업데이트하기가 쉽습니다.

벡터 그래픽 데이터로 그리기

IGraphicsData 인스턴스의 컬렉션이 있는 경우 Graphics 클래스의 drawGraphicsData() 메서드를 사용하여 그래픽을 렌더링합니다. drawGraphicsData() 메서드는 IGraphicsData 인스턴스의 벡터에서 순서대로 드로잉 명령 집합을 수행합니다.

```
// 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);
```

예제의 드로잉에 사용된 패스의 값 하나를 수정하여 보다 복잡한 이미지로 해당 모양을 여러 번 다시 그릴 수 있습니다.

```
// 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);
```

IGraphicsData 객체는 채우기 및 획 스타일을 정의할 수 있지만 채우기 및 획 스타일이 반드시 필요한 것은 아닙니다. 즉, Graphics 클래스 메서드는 스타일을 설정하는 데 사용하고 IGraphicsData 객체는 패스의 저장된 컬렉션을 그리는 데 사용하거나 그 반대로 사용할 수 있습니다.

참고: 위의 예제와 같이 원래 드로잉에 드로잉을 추가하려는 경우가 아니면 새 드로잉을 시작하기 전에 Graphics.clear() 메서드를 사용하여 이전 드로잉을 지웁니다. 패스의 단일 부분이나 IGraphicsData 객체의 컬렉션을 변경할 때는 드로잉 전체를 다시 그려 변경 내용을 확인합니다.

그래픽 데이터 클래스를 사용할 경우 모양은 본질적으로 세 번째 이후의 점에서 닫히므로 세 개 이상의 점을 그릴 때마다 채우기가 렌더링됩니다. 채우기가 닫히더라도 획은 닫히지 않으며 이 비헤이비어는 여러 개의 Graphics.lineTo() 또는 Graphics.moveTo() 명령을 사용할 때와는 다릅니다.

벡터 그래픽 데이터 읽기

Flash Player 11.6 이상, Adobe AIR 3.6 이상

벡터 내용을 표시 객체에 그리는 동작 외에도 Flash Player 11.6 및 Adobe AIR 3.6 이상에서는 Graphics 클래스의 readGraphicsData() 메서드를 사용하여 표시 객체의 벡터 그래픽 내용에 대한 데이터 표현을 얻을 수 있습니다. 이를 통해 그래픽의 스냅샷을 만들어 런타임에 스프라이트 시트를 저장, 복사, 생성하는 등의 작업을 할 수 있습니다.

readGraphicsData() 메서드를 호출하면 IGraphicsData 객체가 포함된 Vector 인스턴스가 반환됩니다. 이는 drawGraphicsData() 메서드로 벡터 그래픽을 그리는 데 사용되는 객체와 동일한 객체입니다.

readGraphicsData() 메서드로 벡터 그래픽을 읽는 데에는 몇 가지 제약이 있습니다. 자세한 내용은 ActionScript 언어 참조 설명서의 readGraphicsData() 항목을 참조하십시오.

drawTriangles() 사용 정보

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Player 10 및 Adobe AIR 1.5에 추가된 또 다른 고급 메서드인 Graphics.drawTriangles()는 Graphics.drawPath() 메서드와 비슷합니다. Graphics.drawTriangles() 메서드는 Vector.<Number> 객체를 사용하여 패스를 그릴 점 위치를 지정합니다.

그러나 Graphics.drawTriangles() 메서드의 실제 목적은 ActionScript를 통해 3차원 효과를 쉽게 만드는 것입니다. Graphics.drawTriangles()를 사용하여 3차원 효과를 만드는 방법에 대한 자세한 내용은 327페이지의 “3D 효과에 삼각형 사용”을 참조하십시오.

13장: 비트맵을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 벡터 드로잉 기능 이외에도 비트맵 이미지 만들기 기능 또는 SWF로 로드되는 외부 비트맵 이미지의 픽셀 데이터 조작 기능을 제공합니다. 개별 픽셀 값을 액세스하고 변경할 수 있으므로 필터와 유사한 고유 이미지 효과를 연출하고 내장 노이즈 함수를 사용하여 텍스처 및 불규칙한 노이즈를 만들 수 있습니다.

- Renaun Erickson: ActionScript에서 블리팅 기법을 사용하여 게임 에셋 렌더링
- Bitmap programming: Essential ActionScript 3 26장(저자: Colin Moock, O'Reilly Media, 2007년)
- Mike Jones: Pushbutton Engine에서의 스프라이트 작업
- Flash & Math: 단순해진 픽셀 입자
- Flixel

비트맵 작업의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

디지털 이미지를 사용하여 작업할 때 보통 두 가지의 기본 그래픽 유형인 비트맵 그래픽과 벡터 그래픽을 접하게 됩니다. 래스터 그래픽이라고도 하는 비트맵 그래픽은 사각형 격자 구조로 배열된 작은 정사각형(픽셀)으로 구성되며, 벡터 그래픽은 선, 곡선 및 다각형 등과 같이 수학적으로 생성된 기하학적 형태로 구성됩니다.

비트맵 이미지는 픽셀 단위로 측정된 폭 및 높이와 각 픽셀의 비트 수(각 픽셀에 포함시킬 수 있는 색상 수)로 정의됩니다. RGB 색상 모델을 사용하는 비트맵 이미지의 경우 픽셀은 빨강, 녹색, 파랑이라는 세 개의 바이트로 구성되며 각 바이트는 0 - 255 범위의 값을 가집니다. 픽셀 내에서 바이트가 결합되면 미술가가 혼합한 페인트 색과 유사한 새로운 색상이 만들어집니다. 예를 들어, 바이트 값이 각각 빨강-255, 녹색-102 및 파랑-0일 경우 강렬한 주황색의 픽셀이 만들어집니다.

비트맵 이미지의 품질은 이미지의 해상도와 해당 색상 심도 비트 값을 결합한 결과에 따라 결정됩니다. **해상도**는 이미지 내에 포함된 픽셀 수와 관련이 있습니다. 따라서 픽셀 수가 많을수록 해상도가 높아지며 이미지는 선명해집니다. **색상 심도**는 한 픽셀에 포함될 수 있는 정보의 양과 관련이 있습니다. 예를 들어, 픽셀당 색상 심도 값이 16비트인 이미지의 경우 색상 심도가 48비트인 이미지와 동일한 색상 수를 표현할 수 없습니다. 따라서 48비트 이미지의 음영은 16비트 이미지보다 훨씬 매끄럽게 표현됩니다.

비트맵 그래픽은 해상도의 영향을 받기 때문에 크기 조절이 원활하게 이루어지지 않습니다. 특히 비트맵 이미지의 크기를 확장하게 되면 이러한 단점이 두드러집니다. 일반적으로 비트맵의 크기를 확장하면 디테일과 품질이 떨어집니다.

비트맵 파일 형식

비트맵 이미지는 일반적인 몇 가지 파일 형식으로 그룹화됩니다. 이러한 형식은 각기 다른 압축 알고리즘 유형을 사용하여 파일 크기를 줄이며 이미지의 최종 목적에 적합한 방식으로 이미지 품질을 최적화합니다. Adobe 런타임에서 지원하는 비트맵 이미지 형식은 BMP, GIF, JPG, PNG 및 TIFF입니다.

BMP

BMP(비트맵) 형식은 Microsoft Windows 운영 체제에서 사용되는 기본 이미지 형식으로, 어떠한 형태의 압축 알고리즘도 사용하지 않으므로 일반적으로 파일 크기가 큽니다.

GIF

GIF(Graphics Interchange Format)는 256 색상(8비트 색상) 이미지를 전송하기 위한 수단으로 CompuServe에서 1987년에 처음 개발된 형식으로, 작은 파일 크기를 제공하므로 웹 기반 이미지에 적합합니다. 그러나 색상 팔레트가 한정되어 있기 때문에 GIF 이미지는 일반적으로 높은 수준의 음영과 색상 그래디언트를 요구하는 사진에는 대체로 적합하지 않습니다. GIF 이미지는 색상을 투명하게 매핑하도록 하는 단일 비트 투명도를 허용합니다. 따라서 웹 페이지 배경색은 투명도가 매핑된 이미지를 통해 표시됩니다.

JPEG

JPEG(Joint Photographic Experts Group)에서 개발한 JPEG(또는 JPG) 이미지 형식은 손실 압축 알고리즘을 사용하여 작은 파일 크기의 24비트 색상 심도를 구현합니다. 손실 압축이란 이미지를 저장할 때마다 이미지의 품질과 데이터는 손실되지만 파일 크기가 줄어드는 것을 의미합니다. JPEG 형식은 무수한 색상을 표시할 수 있기 때문에 사진에 적합합니다. 아울러 이미지에 적용할 압축 수준을 제어할 수 있어 이미지 품질 및 파일 크기를 조작할 수 있습니다.

PNG

PNG(Portable Network Graphics) 형식은 특허받은 GIF 파일 형식에 대한 오픈 소스 대안으로 개발되었으며, PNG는 최대 64비트 색상 심도를 지원하여 천육백만 색상 표현을 가능하게 합니다. PNG는 앞서 설명한 두 형식보다 상대적으로 최근에 개발된 형식이기 때문에 일부 구 버전의 브라우저에서는 PNG 파일을 지원하지 않습니다. JPG와는 다르게 PNG는 손실 없는 압축 방법을 사용하므로 이미지를 저장하더라도 이미지 데이터가 손실되지 않습니다. PNG 파일은 또한 최대 256 레벨의 투명도를 가능하게 하는 알파 투명도를 지원합니다.

TIFF

TIFF(Tagged Image File Format)는 PNG가 도입되기 전에 선택할 수 있었던 형식으로, 플랫폼의 영향을 받지 않습니다. TIFF 형식은 그 종류가 매우 다양해서 모든 버전을 처리할 수 있는 단일 리더가 없다는 단점이 있습니다. 또한 현재 이 형식을 지원하는 웹 브라우저도 없습니다. TIFF에서는 손실 압축이나 무손실 압축을 사용할 수 있으며 CMYK와 같은 장치 고유의 색상 공간을 처리할 수 있습니다.

투명 비트맵 및 불투명 비트맵

GIF 또는 PNG 형식을 사용하는 비트맵 이미지의 경우 각 픽셀에 바이트(알파 채널)를 추가할 수 있습니다. 이 추가 픽셀 바이트는 픽셀의 투명도 값을 나타냅니다.

GIF 이미지는 단일 비트 투명도를 지원하므로 256 색상 팔레트에서 단 한 개의 색상만 투명으로 지정할 수 있습니다. 반면 PNG 이미지의 경우 최대 256 레벨의 투명도를 표현할 수 있습니다. 이 기능은 이미지 또는 텍스트를 배경에 블렌드해야 할 때 특히 유용합니다.

ActionScript 3.0에서는 BitmapData 클래스 내에서 이 추가 투명도 픽셀 바이트를 복제합니다. ActionScript는 PNG 투명도 모델과 유사하게 최대 256 레벨의 투명도를 제공합니다.

중요한 개념 및 용어

비트맵 그래픽과 관련한 중요한 용어가 아래 목록에 정리되어 있습니다.

알파 색상 또는 이미지의 투명도 수준(보다 정확히는 불투명도)입니다. 알파의 양은 흔히 **알파 채널** 값으로 설명됩니다.

ARGB 색상 각 픽셀의 색상이 빨강, 녹색 및 파랑 색상 값의 혼합인 색상 체계로 투명도가 알파 값으로 지정됩니다.

색상 채널 색상은 일반적으로 몇 가지 기본 색상(컴퓨터 그래픽의 경우 대개 빨강, 녹색, 파랑)의 혼합으로 표현됩니다. 각각의 기본 색상은 색상 채널로 간주되며 각 색상 채널의 색상 양이 서로 혼합되어 최종 색상을 만듭니다.

색상 심도 비트 심도라고도 하며, 각 픽셀에 할당된 컴퓨터 메모리 크기를 나타냅니다. 색상 심도는 이미지에서 표현할 수 있는 색상 수를 결정합니다.

픽셀 비트맵 이미지의 최소 정보 단위로, 본질적으로는 색상 도트를 나타냅니다.

해상도 이미지의 픽셀 크기로, 이미지에 포함된 세부 요소의 세분화 수준을 결정합니다. 해상도는 보통 픽셀 수를 단위로 하는 폭 및 높이로 표시됩니다.

RGB 색상 각 픽셀의 색상이 빨강, 녹색 및 파랑 색상 값의 혼합으로 표현되는 색상 체계를 의미합니다.

Bitmap 클래스 및 BitmapData 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

Bitmap 클래스는 비트맵 이미지 작업에 사용되는 ActionScript 3.0의 기본 클래스로, 화면에 비트맵 이미지를 표시하는 데 사용됩니다. BitmapData 클래스는 비트맵의 원시 이미지 데이터에 액세스하여 이를 조작하는 데 사용됩니다.

기타 도움말 항목
flash.display.Bitmap

flash.display.BitmapData

Bitmap 클래스의 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

DisplayObject 클래스의 하위 클래스인 Bitmap 클래스는 비트맵 이미지를 표시하는 데 사용되는 ActionScript 3.0의 기본 클래스입니다. 이러한 이미지는 flash.display.Loader 클래스를 통해 로드되었거나 Bitmap() 생성자를 사용하여 동적으로 생성되었을 수 있습니다. 외부 소스에서 이미지를 로드할 경우 Bitmap 객체는 GIF, JPEG 또는 PNG 형식의 이미지만 사용할 수 있습니다. 인스턴스화된 Bitmap 인스턴스는 스테이지로 렌더링해야 할 BitmapData 객체의 래퍼로 간주할 수 있습니다. Bitmap 인스턴스는 표시 객체이므로 표시 객체의 모든 특징 및 기능을 Bitmap 인스턴스 조작에도 사용할 수 있습니다. 표시 객체 작업에 대한 자세한 내용은 135페이지의 “디스플레이 프로그래밍”을 참조하십시오.

픽셀에 물리기 및 픽셀 다듬기

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 표시 객체에 공통적으로 사용할 수 있는 기능 이외에도 Bitmap 클래스는 비트맵 이미지에만 사용할 수 있는 추가 기능을 제공합니다.

Bitmap 클래스의 pixelSnapping 속성은 Bitmap 객체를 가장 가까운 픽셀에 물릴지 여부를 결정합니다. 이 속성은 또한 PixelSnapping 클래스에 정의된 세 가지 상수(ALWAYS, AUTO, NEVER)를 사용합니다.

픽셀 물리기 기능을 적용하는 구문은 다음과 같습니다.

```
myBitmap.pixelSnapping = PixelSnapping.ALWAYS;
```

비트맵 이미지의 배율을 조절하면 이미지가 흐려지거나 왜곡되는 경우가 많습니다. 이러한 왜곡 현상을 줄이기 위해 BitmapData 클래스의 smoothing 속성이 사용됩니다. 이 부울 속성이 true로 설정된 경우, 이미지 배율을 조절할 때 이미지의 픽셀이 앤티앨리어싱 처리되어 다듬어집니다. 그러면 이미지의 모양이 더 뚜렷하고 자연스러워집니다.

BitmapData 클래스의 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.display 패키지에 포함된 BitmapData 클래스는 로드되었거나 동적으로 생성된 비트맵 이미지에 포함된 픽셀의 사진 같은 스냅샷에 비유할 수 있습니다. 이 스냅샷은 객체 내 픽셀 데이터 배열로 표현됩니다. BitmapData 클래스에는 픽셀 데이터를 만들고 조작하는 데 유용한 일련의 내장 메서드도 포함되어 있습니다.

BitmapData 객체를 인스턴스화하려면 다음 코드를 사용하십시오.

```
var myBitmap:BitmapData = new BitmapData(width:Number, height:Number, transparent:Boolean,
fillColor:uinit);
```

width 및 height 매개 변수는 비트맵 크기를 지정합니다. AIR 3 및 Flash Player 11부터는 BitmapData 객체의 크기 제한이 없어졌습니다. 비트맵의 최대 크기는 운영 체제에 따라 다릅니다.

AIR 1.5 및 Flash Player 10에서는 BitmapData 객체의 최대 크기가 8,191픽셀(폭 또는 높이)이며 총 픽셀 수는 16,777,215픽셀을 초과할 수 없습니다. 따라서 BitmapData 객체의 폭이 8,191픽셀이면 높이가 2,048픽셀 이하여야 합니다. Flash Player 9 이전 버전 및 AIR 1.1 이전 버전에서는 이 제한이 높이 2,880픽셀 및 폭 2,880픽셀입니다.

transparent 매개 변수는 비트맵 데이터에 알파 채널이 포함되어 있는지(true) 또는 포함되어 있지 않은지(false) 여부를 지정합니다. fillColor 매개 변수는 32비트 색상 값으로 배경색은 물론 투명도 값(true로 설정된 경우)을 지정합니다. 다음 예제에서는 투명도 50%의 주황색 배경을 가진 BitmapData 객체를 만듭니다.

```
var myBitmap:BitmapData = new BitmapData(150, 150, true, 0x80FF3300);
```

새로 만든 BitmapData 객체를 화면에 렌더링하려면 해당 객체를 Bitmap 인스턴스에 할당하거나 그 안에 래핑하십시오. 이를 위해 BitmapData 객체를 Bitmap 객체 생성자의 매개 변수로 전달하거나 기존 Bitmap 인스턴스의 bitmapData 속성에 할당할 수 있습니다. 또한, Bitmap 인스턴스를 포함할 표시 객체 컨테이너의 addChild() 또는 addChildAt() 메서드를 호출하여 표시 목록에 Bitmap 인스턴스를 추가해야 합니다. 표시 목록을 사용한 작업에 대한 자세한 내용은 142페이지의 "표시 목록에 표시 객체 추가"를 참조하십시오.

다음 예제에서는 빨간색으로 칠한 BitmapData 객체를 만든 후 해당 객체를 Bitmap 인스턴스에 표시합니다.

```
var myBitmapDataObject:BitmapData = new BitmapData(150, 150, false, 0xFF0000);
var myImage:Bitmap = new Bitmap(myBitmapDataObject);
addChild(myImage);
```

픽셀 조작

Flash Player 9 이상, Adobe AIR 1.0 이상

BitmapData 클래스에는 픽셀 데이터 값을 조작할 수 있는 메서드 집합이 포함되어 있습니다.

개별 픽셀 조작

Flash Player 9 이상, Adobe AIR 1.0 이상

픽셀 수준에서 비트맵 이미지의 모양을 바꿀 때에는 먼저 조작할 영역 내에 포함된 픽셀의 색상 값을 확인해야 합니다. 이러한 픽셀 값은 getPixel() 메서드를 사용하여 확인할 수 있습니다.

getPixel() 메서드는 매개 변수로 전달된 x, y(픽셀) 좌표에서 RGB 값을 가져옵니다. 조작하려는 픽셀에 투명도(알파 채널) 정보가 포함된 경우에는 getPixel32() 메서드를 사용해야 합니다. 이 메서드도 RGB 값을 가져오지만 getPixel()과는 다릅니다. getPixel32()에는 선택된 픽셀의 알파 채널(투명도) 값을 나타내는 추가 데이터가 포함되어 있습니다.

또는 비트맵에 포함된 픽셀의 색상이나 투명도만 변경하려는 경우에는 setPixel() 또는 setPixel32() 메서드를 사용할 수 있습니다. 픽셀 색상을 설정하려면 x, y 좌표 및 색상 값을 이러한 메서드 중 하나에 전달하기만 하면 됩니다.

다음 예제에서는 setPixel() 메서드를 사용하여 녹색 BitmapData 배경에 십자가를 그립니다. 그런 다음 getPixel() 메서드를 사용하여 좌표 50, 50에 있는 픽셀의 색상 값을 가져오고 반환된 값을 추적합니다.

```
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));
```

단일 픽셀이 아닌 픽셀 그룹의 값을 읽으려는 경우에는 getPixels() 메서드를 사용합니다. 이 메서드는 매개 변수로 전달된 픽셀 데이터의 사각형 영역에서 바이트 배열을 생성합니다. 각각의 바이트 배열 요소(즉, 픽셀 값)는 부호 없는 정수(32비트, 곱하지 않은 픽셀 값)입니다.

반대로 픽셀 그룹의 값을 변경하거나 설정하려면 setPixels() 메서드를 사용합니다. 이 메서드는 두 매개 변수(rect 및 inputByteArray)가 입력될 때까지 기다린 후 이 둘을 결합하여 픽셀 데이터(inputByteArray)의 사각형 영역(rect)을 출력합니다.

inputByteArray에서 데이터를 읽고 기록하면 배열의 각 픽셀에 대해 ByteArray.readUnsignedInt() 메서드가 호출됩니다. 어떤 이유로 인해 inputByteArray에 픽셀 데이터에 해당하는 전체 사각형이 포함되어 있지 않으면 해당 시점에서 메서드가 이미지 처리를 중단합니다.

픽셀 데이터를 가져오고 설정하려면 바이트 배열이 32비트 알파, 빨강, 녹색, 파랑(ARGB) 픽셀 값이어야 합니다.

다음 예제에서는 getPixels() 및 setPixels() 메서드를 사용하여 픽셀 그룹을 한 BitmapData 객체에서 다른 BitmapData 객체로 복사합니다.

```
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;
```

픽셀 수준 충돌 감지

Flash Player 9 이상, Adobe AIR 1.0 이상

BitmapData.hitTest() 메서드는 비트맵 데이터와 다른 객체 또는 점 사이에서 픽셀 수준의 충돌이 있었는지 여부를 감지합니다.

BitmapData.hitTest() 메서드는 다음 5개의 매개 변수를 사용합니다.

- firstPoint(점): 히트 테스트가 수행되는 첫 번째 BitmapData의 왼쪽 위 모서리의 픽셀 위치를 나타내는 매개 변수입니다.
- firstAlphaThreshold(단위): 해당 히트 테스트에 대해 불투명으로 간주되는 최고 알파 채널 값을 지정하는 매개 변수입니다.
- secondObject(객체): 영향을 받는 영역을 나타내는 매개 변수입니다. secondObject 객체는 Rectangle, Point, Bitmap 또는 BitmapData 객체일 수 있습니다. 또한 충돌 감지가 수행되는 히트 영역을 나타냅니다.
- secondBitmapDataPoint(점): 2차 BitmapData 객체의 픽셀 위치를 정의하는 데 사용되는 선택적 매개 변수로서, secondObject의 값이 BitmapData 객체일 경우에만 사용됩니다. 기본값은 null입니다.
- secondAlphaThreshold(단위): 2차 BitmapData 객체에서 불투명으로 간주되는 최고 알파 채널 값을 나타내는 선택적 매개 변수로서, 기본값은 1이며 secondObject의 값이 BitmapData 객체이고 두 BitmapData 객체가 모두 투명 객체일 때만 사용됩니다.

불투명 이미지에서 충돌 감지 작업을 수행할 경우 ActionScript에서는 이미지를 완전히 불투명한 사각형(또는 경계 상자)처럼 처리한다는 것을 염두에 두십시오. 또는 투명한 이미지에서 픽셀 수준의 히트 테스트를 수행할 경우 두 이미지 모두 투명해야 합니다. 이외에도 ActionScript는 알파 임계값 매개 변수를 사용하여 픽셀이 투명에서 불투명으로 바뀌는 지점을 결정합니다.

다음 예제에서는 세 개의 비트맵 이미지를 만든 다음 두 개의 서로 다른 충돌점(한 점에서는 false를, 다른 점에서는 true를 반환)을 사용하여 픽셀 충돌 여부를 확인합니다.

```
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
```

비트맵 데이터 복사

Flash Player 9 이상, Adobe AIR 1.0 이상

이미지 간에 비트맵 데이터를 복사하기 위해 clone(), copyPixels(), copyChannel(), draw(), drawWithQuality()(drawWithQuality 메서드는 Flash Player 11.3 이상, AIR 3.3 이상에서 사용 가능)와 같은 여러 메서드를 사용할 수 있습니다.

clone() 메서드는 이름에서도 알 수 있듯이 비트맵 데이터를 한 BitmapData 객체에서 다른 BitmapData 객체로 복제하거나 샘플링합니다. 이 메서드를 호출하면 BitmapData 객체를 복제한 원본 인스턴스를 똑같이 복제한 새 BitmapData 객체가 반환됩니다.

다음 예제에서는 주황색(부모) 정사각형을 복제한 후 원본 부모 정사각형 옆에 해당 복제본을 배치합니다.

```
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;
```

copyPixels() 메서드는 BitmapData 객체 간에 픽셀을 복사하는 쉽고 빠른 방법입니다. 이 메서드는 소스 이미지의 사각형 스냅샷(sourceRect 매개 변수로 정의됨)을 가져와서 같은 크기의 다른 사각형 영역으로 복사합니다. 새로 "붙여 넣은" 사각형의 위치는 destPoint 매개 변수 내에 정의됩니다.

copyChannel() 메서드는 소스 BitmapData 객체에서 미리 정의된 색상 채널 값(알파, 빨강, 녹색 또는 파랑)을 샘플링하여 대상 BitmapData 객체의 채널에 복사합니다. 이 메서드를 호출하더라도 대상 BitmapData 객체의 다른 채널은 영향을 받지 않습니다.

draw() 및 drawWithQuality() 메서드는 소스 스프라이트, 동영상 클립 또는 기타 표시 객체의 그래픽 내용을 새 비트맵에 그리거나 렌더링합니다. matrix, colorTransform, blendMode 및 대상 clipRect 매개 변수를 사용하면 새 비트맵이 렌더링되는 방식을 수정할 수 있습니다. 이 메서드는 Flash Player 및 AIR의 벡터 렌더러를 사용하여 데이터를 생성합니다.

draw() 또는 drawWithQuality()를 호출하면 여기서 설명한 것처럼 소스 객체(스프라이트, 동영상 클립 또는 기타 표시 객체)가 첫 번째 매개 변수로 전달됩니다.

```
myBitmap.draw(movieClip);
```

소스 객체가 처음 로드된 후에 색상 및 행렬 등의 변형이 적용된 경우, 이러한 변형은 새 객체로 복사되지 않습니다. 변형을 새 비트맵으로 복사하려면 원본 객체의 transform 속성 값을 새 BitmapData 객체를 사용하는 Bitmap 객체의 transform 속성에 복사해야 합니다.

비트맵 데이터 압축

Flash Player 11.3 이상, AIR 3.3 이상

flash.display.BitmapData.encode() 메서드를 사용하면 비트맵 데이터를 다음 이미지 압축 형식 중 하나로 기본적으로 압축할 수 있습니다.

- **PNG** - PNG 압축을 사용합니다. 파일 크기보다 압축 속도를 강조하는 빠른 압축을 사용할 수도 있습니다. PNG 압축을 사용하려면 새 flash.display.PNGEncoderOptions 객체를 BitmapData.encode() 메서드의 두 번째 매개 변수로 전달합니다.

- **JPEG** - JPEG 압축을 사용합니다. 이미지 품질을 지정할 수도 있습니다. JPEG 압축을 사용하려면 새 flash.display.JPEGEncoderOptions 객체를 BitmapData.encode() 메서드의 두 번째 매개 변수로 전달합니다.
- **JPEGXR** - JPEG XR(Extended Range) 압축을 사용합니다. 색상 채널, 손실률 및 엔트로피 설정을 지정할 수도 있습니다. JPEGXR 압축을 사용하려면 새 flash.display.JPEGXREncoderOptions 객체를 BitmapData.encode() 메서드의 두 번째 매개 변수로 전달합니다.

이미지 처리를 위해 이 기능을 서버 업로드 또는 다운로드 작업 과정의 일부로 사용할 수 있습니다.

다음 예제 코드 조각은 JPEGEncoderOptions를 사용하여 BitmapData 객체를 압축합니다.

```
// 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);
```

노이즈 함수를 사용하여 텍스처 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

noise() 메서드 또는 perlinNoise() 메서드를 통해 비트맵에 노이즈 효과를 적용하여 비트맵 모양을 수정할 수 있습니다. 노이즈 효과는 튜닝되지 않은 텔레비전 화면에 나타나는 잡음에 비유할 수 있습니다.

비트맵에 노이즈 효과를 적용하려면 noise() 메서드를 사용하십시오. 이 메서드는 비트맵 이미지의 특정 영역에 있는 픽셀에 임의의 색상 값을 적용합니다.

이 메서드는 다음 5개의 매개 변수를 사용합니다.

- randomSeed(정수): 패턴을 결정하는 난수 초기값입니다. 이름과는 달리 이 숫자는 사실상 동일한 숫자가 전달될 경우 동일한 결과를 생성합니다. 따라서 의미 있는 임의의 결과를 얻으려면 Math.random() 메서드를 사용하여 이 매개 변수의 난수를 전달해야 합니다.
- low(단위): 각 픽셀(0 ~ 255)에 대해 생성되는 최저 값을 나타내는 매개 변수로서, 기본값은 0입니다. 이 값을 낮게 설정하면 노이즈 패턴이 어두워지고 높게 설정하면 노이즈 패턴이 밝아집니다.
- high(단위): 각 픽셀(0 ~ 255)에 대해 생성되는 최고 값을 나타내는 매개 변수로서, 기본값은 255입니다. 이 값을 낮게 설정하면 노이즈 패턴이 어두워지고 높게 설정하면 노이즈 패턴이 밝아집니다.
- channelOptions(단위): 노이즈 패턴이 적용될 Bitmap 객체의 색상 채널을 지정하는 매개 변수로서, 네 가지 색상 채널 ARGB 값을 임의로 결합한 숫자가 될 수 있습니다. 기본값은 7입니다.
- grayScale(부울): true로 설정된 경우 randomSeed 값을 비트맵 픽셀에 적용하여 이미지의 모든 색상을 효과적으로 지웁니다. 알파 채널은 이 매개 변수의 영향을 받지 않습니다. 기본값은 false입니다.

다음 예제에서는 비트맵 이미지를 생성한 후 이 이미지에 파란색 노이즈 패턴을 적용합니다.

```
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);
        }
    }
}
```

좀 더 유기적인 텍스처를 만들려면 perlinNoise() 메서드를 사용합니다. perlinNoise() 메서드는 연기, 구름, 물, 불 또는 폭발 등의 효과를 표현하는 데 적합한 사실적이고 유기적인 텍스처를 생성합니다.

perlinNoise() 메서드는 알고리즘을 통해 텍스처를 생성하기 때문에 비트맵 기반 텍스처보다 사용되는 메모리가 적습니다. 그러나 프로세서 사용에는 영향을 주므로 내용이 느리게 표시될 수 있으며 그 결과 특히 오래된 컴퓨터에서는 화면이 프레임 속도보다 느린 속도로 다시 그려집니다. 이와 같은 문제는 주로 Perlin 노이즈 알고리즘을 처리하기 위해 부동 소수점 계산을 수행할 때 발생합니다.

이 메서드는 다음과 같은 9개의 매개 변수를 사용하며, 이 중 처음 6개는 필수 매개 변수입니다.

- baseX(숫자): 만들어진 패턴의 x(크기) 값을 결정합니다.
- baseY(숫자): 만들어진 패턴의 y(크기) 값을 결정합니다.
- numOctaves(단위): 이 노이즈를 생성하기 위해 결합할 옥타브, 즉 개별 노이즈 함수의 수입니다. 옥타브 수가 많을수록 보다 상세한 이미지가 만들어지지만 처리하는 데 시간이 오래 걸립니다.
- randomSeed(정수): 난수 초기값으로 noise() 함수에서 작동하는 것과 같은 방식으로 작동합니다. 따라서 의미 있는 임의의 결과를 얻으려면 Math.random() 메서드를 사용하여 이 매개 변수의 난수를 전달해야 합니다.
- stitch(부울): true로 설정된 경우 이 메서드는 비트맵 채우기 작업 시 연속 타일링 텍스처를 만들어내기 위해 이미지의 가장자리를 매끄럽게 하려고 시도합니다.
- fractalNoise(부울): 메서드에 의해 생성된 그래디언트의 가장자리와 관련된 매개 변수입니다. true로 설정된 경우 해당 효과의 가장자리를 매끄럽게 하는 프랙탈 노이즈를 생성합니다. false로 설정된 경우에는 난류가 생성됩니다. 난류가 포함된 이미지의 경우 경사면에서 불연속 선이 드러나기 때문에 불꽃과 파도 같은 시각적 효과를 만드는 데 적합합니다.
- channelOptions(단위): channelOptions 매개 변수는 noise() 메서드에서 작동하는 것과 같은 방식으로 작동합니다. 또한 노이즈 패턴이 적용되는 비트맵의 색상 채널을 지정합니다. 네 가지 색상 채널 ARGB 값을 임의로 결합한 숫자가 될 수 있습니다. 기본값은 7입니다.
- grayScale(부울): grayScale 매개 변수는 noise() 메서드에서 작동하는 것과 같은 방식으로 작동합니다. true로 설정된 경우 randomSeed 값을 비트맵 픽셀에 적용하여 이미지의 모든 색상을 효과적으로 지웁니다. 기본값은 false입니다.
- offsets(배열): 각 옥타브의 x 및 y 오프셋에 해당하는 점 배열입니다. 오프셋 값을 조작하면 이미지 레이어를 부드럽게 스크롤할 수 있습니다. 오프셋 배열의 각 점은 특정 옥타브 노이즈 함수에 영향을 미칩니다. 기본값은 null입니다.

다음 예제에서는 perlinNoise() 메서드를 호출하여 녹색 및 파란색 구름 효과를 생성하는 150 x 150픽셀의 BitmapData 객체를 만듭니다.

```
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);
        }
    }
}
```

비트맵 스크롤

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 지도를 움직일 때마다(단 몇 개의 픽셀만 움직였을 경우에도) 화면을 업데이트해야 하는 주소 매핑 응용 프로그램을 개발했다고 가정합시다.

이러한 기능을 구현하는 방법 중 하나는 사용자가 지도를 움직일 때마다 업데이트된 맵 화면이 포함된 새 이미지를 다시 렌더링하는 것입니다. 또는 한 개의 커다란 이미지와 scroll() 메서드를 사용하는 방법도 생각할 수 있습니다.

scroll() 메서드는 화면상에 있는 비트맵을 복사하여 x, y 매개 변수에서 지정한 새로운 오프셋 위치에 붙여 넣습니다. 비트맵의 일부가 안 보이는 위치에 있을 경우 이 메서드는 이미지가 이동한 것처럼 보이는 효과를 연출합니다. 또한 타이머 함수(또는 enterFrame 이벤트)와 함께 사용할 경우 이미지에 애니메이션 효과나 스크롤 효과를 줄 수 있습니다.

다음 예제에서는 앞서 설명한 Perlin 노이즈 예제를 활용하여 이미지의 3/4이 스테이지에 표시되지 않는 큰 비트맵 이미지를 생성합니다. 그런 다음 scroll() 메서드와 이미지를 대각선 아래쪽 방향으로 1픽셀씩 오프셋하는 enterFrame 이벤트 리스너를 적용합니다. 이 메서드는 해당 프레임에 진입할 때마다 호출되며 그 결과, 이미지를 아래로 스크롤할 때 화면 밖의 이미지 부분을 스테이지로 렌더링합니다.

```
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);
}
```

밉매핑 이용

Flash Player 9 이상, Adobe AIR 1.0 이상

MIP 맵(밉매핑이라고도 함)은 런타임 렌더링 품질과 성능을 향상시키기 위해 함께 그룹화되고 텍스처와 연결되는 비트맵입니다. MIP 맵의 각 비트맵 이미지는 기본 비트맵 이미지의 한 버전이지만 기본 이미지보다 세부 수준이 줄어듭니다.

예를 들어 최고 품질 수준으로 64 × 64 픽셀인 기본 이미지를 포함하는 MIP 맵이 있을 수 있습니다. MIP 맵에서 품질이 더 낮은 이미지는 32 × 32, 16 × 16, 8 × 8, 4 × 4, 2 × 2, 1 × 1 픽셀이 됩니다.

텍스처 스트리밍을 사용하면 먼저 가장 낮은 품질의 비트맵을 로드한 후, 비트맵이 로드됨에 따라 더 높은 품질의 비트맵을 점진적으로 표시할 수 있습니다. 비트맵의 품질이 낮으면 크기가 작기 때문에 기본 이미지보다 빨리 로드할 수 있습니다. 따라서 응용 프로그램 사용자는 기본 고품질 비트맵을 로드하기 전에 응용 프로그램에서 이미지를 볼 수 있습니다.

Flash Player 9.115.0 이상 버전과 AIR에서는 50%에서 시작하여 다양한 배율로 각 비트맵의 최적화된 버전을 만들어 이 기술을 구현합니다. 이 프로세스를 **밉매핑**이라고 합니다.

Flash Player 11.3 및 AIR 3.3에서는 streamingLevels 매개 변수(Context3D.createCubeTexture() 및 Context3D.createTexture() 메서드)를 통해 텍스처 스트리밍을 지원합니다.

텍스처 압축을 사용하면 GPU에서 직접 압축된 형식으로 텍스처 이미지를 저장할 수 있으므로 GPU 메모리와 메모리 대역폭이 절약됩니다. 일반적으로 압축된 텍스처는 오프라인으로 압축되며, 압축된 형식으로 GPU에 업로드됩니다. 그러나 Flash Player 11.4 및 AIR 3.4는 런타임 텍스처 압축을 지원하므로 벡터 아트에서 동적 텍스처를 렌더링하는 경우 등 특정 상황에서 유용합니다. 런타임 텍스처 압축을 사용하려면 다음 단계를 수행하십시오.

- Context3D.createTexture() 메서드를 호출하고, 세 번째 매개 변수에서 flash.display3D.Context3DTextureFormat.COMPRESSED 또는 flash.display3D.Context3DTextureFormat.COMPRESSED_ALPHA를 전달하여 텍스처 객체를 만듭니다.
- flash.display3D.textures.Texture 인스턴스(createTexture()에서 반환됨)를 사용하여 flash.display3D.textures.Texture.uploadFromBitmapData() 또는 flash.display3D.textures.Texture.uploadFromByteArray()를 호출합니다. 이러한 메서드는 한 단계로 텍스처를 업로드하고 압축합니다.

MIP 맵은 다음 유형의 비트맵에 대해 생성됩니다.

- ActionScript 3.0 Loader 클래스를 사용하여 표시되는 비트맵(JPEG, GIF 또는 PNG 파일)
- Flash Professional 문서의 라이브러리에 있는 비트맵

- BitmapData 객체
- ActionScript 2.0 loadMovie() 함수를 사용하여 표시되는 비트맵

필터링된 객체나 비트맵 캐싱된 동영상 클립에는 MIP 맵이 적용되지 않습니다. 그러나 비트맵이 마스크 처리된 내용에 있는 경우에도 필터링된 표시 객체 내에 비트맵 변형이 있으면 MIP 맵이 적용됩니다.

밉매핑은 자동으로 수행되지만 다음 몇 가지 지침에 따라 이미지에서 이 최적화를 최대한 활용할 수 있습니다.

- 비디오 재생의 경우 Video 객체의 smoothing 속성을 true로 설정합니다(Video 클래스 참조).
- 비트맵의 경우 smoothing 속성을 true로 설정할 필요는 없지만 비트맵에 다듬기를 사용할 때 품질 향상이 보다 명확하게 나타납니다.
- 2차원 이미지의 경우 4 또는 8로 나눌 수 있는 비트맵 크기(예: 640 x 128을 320 x 64 > 160 x 32 > 80 x 16 > 40 x 8 > 20 x 4 > 10 x 2 > 5 x 1로 줄일 수 있음)를 사용합니다.

  3차원 텍스처의 경우 각 이미지가 2의 거듭제곱(즉 2^n)인 해상도가 되는 MIP 맵을 사용합니다. 예를 들어 기본 이미지의 해상도가 1024 x 1024 픽셀인 경우, MIP 맵에서 더 낮은 품질 이미지는 512 x 512, 256 x 256, 128 x 128에서 1 x 1 픽셀까지이므로 해당 MIP 맵에 총 11개의 이미지가 있게 됩니다.

  밉매핑은 홀수 폭이나 높이를 갖는 비트맵 내용에 대해서는 발생하지 않습니다.

비트맵 예제: 회전하는 달 애니메이션

Flash Player 9 이상, Adobe AIR 1.0 이상

회전하는 달 애니메이션 예제는 Bitmap 객체 및 비트맵 이미지 데이터(BitmapData 객체)를 사용하는 방법을 보여 줍니다. 이 예제에서는 평평한 달 표면의 이미지를 원시 이미지 데이터로 사용하여 회전하는 둥근 달의 애니메이션을 만듭니다. 다음과 같은 작업을 수행하는 방법을 볼 수 있습니다.

- 외부 이미지를 로드하고 해당 이미지의 원시 이미지 데이터에 액세스
- 소스 이미지의 다른 부분에서 픽셀을 반복적으로 복사하여 애니메이션 만들기
- 픽셀 값을 설정하여 비트맵 이미지 만들기

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을참조하십시오. 회전하는 달 애니메이션 응용 프로그램 파일은 Samples/SpinningMoon 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
SpinningMoon.mxml 또는 SpinningMoon.fla	Flex(MXML) 또는 Flash(FLA) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/moon/MoonSphere.as	달의 로드, 표시 및 애니메이션 기능을 수행하는 클래스입니다.
moonMap.png	달 표면의 사진을 포함하는 이미지 파일로 이 파일이 로드되어 회전하는 달 애니메이션을 만드는 데 사용됩니다.

외부 이미지를 비트맵 데이터로 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

이 샘플에서의 첫 번째 중요한 작업은 달 표면의 사진인 외부 이미지 파일을 로드하는 것입니다. 로드 작업은 MoonSphere 클래스의 두 메서드에 의해 처리되는데 하나는 로드 프로세스가 시작되는 MoonSphere() 생성자이고 다른 하나는 외부 이미지가 완전하게 로드되면 호출되는 imageLoadComplete() 메서드입니다.

외부 이미지를 로드하는 것은 외부 SWF를 로드하는 것과 비슷합니다. 둘 다 flash.display.Loader 클래스의 인스턴스를 사용하여 로드 작업을 수행합니다. 이미지를 로드하기 시작하는 MoonSphere() 메서드의 실제 코드는 다음과 같습니다.

```
var imageLoader:Loader = new Loader();
imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoadComplete);
imageLoader.load(new URLRequest("moonMap.png"));
```

첫 번째 행에서는 imageLoader라는 Loader 인스턴스를 선언합니다. 세 번째 행에서는 Loader 객체의 load() 메서드를 호출하고 로드할 이미지의 URL을 나타내는 URLRequest 인스턴스를 전달하여 로드 프로세스를 실제로 시작합니다. 두 번째 줄에서는 이미지가 완전히 로드되면 트리거될 이벤트 리스너를 설정합니다. addEventListener() 메서드는 Loader 인스턴스 자체에서 호출되지 않고 Loader 객체의 contentLoaderInfo 속성에서 호출됩니다. Loader 인스턴스 자체에서는 로드되는 내용과 관련된 이벤트를 전달하지 않습니다. 그러나 이 인스턴스의 contentLoaderInfo 속성에는 Loader 객체로 로드되는 내용(여기서는 외부 이미지)과 연결된 LoaderInfo 객체에 대한 참조가 들어 있습니다. LoaderInfo 객체는 외부 내용 로드의 진행과 완료에 관련된 여러 이벤트를 제공합니다. 이러한 이벤트에는 이미지가 완전하게 로드되면 imageLoadComplete() 메서드에 대한 호출을 트리거할 complete 이벤트(Event.COMPLETE) 등이 있습니다.

외부 이미지 로드를 시작하는 것도 프로세스에서 중요한 부분이지만 로드가 완료될 때 해야 할 작업에 대해 알고 있는 것도 똑같이 중요합니다. 위의 코드에서와 같이 이미지가 로드될 때 imageLoadComplete() 함수가 호출됩니다. 이 함수는 로드된 이미지 데이터를 사용하여 여러 작업을 수행하는데, 이 내용은 뒷부분에서 이어서 설명합니다. 이미지 데이터를 사용하려면 해당 데이터에 액세스해야 합니다. Loader 객체를 사용하여 외부 이미지를 로드하는 경우 로드된 이미지는 Bitmap 인스턴스가 되며 이 인스턴스는 Loader 객체의 자식 표시 객체로 연결됩니다. 이 경우에 Loader 인스턴스는 이벤트 리스너 메서드에 매개 변수로 전달되는 이벤트 객체에 포함되므로 이벤트 리스너 메서드에서 이 인스턴스를 사용할 수 있습니다. imageLoadComplete() 메서드의 첫 번째 줄은 다음과 같습니다.

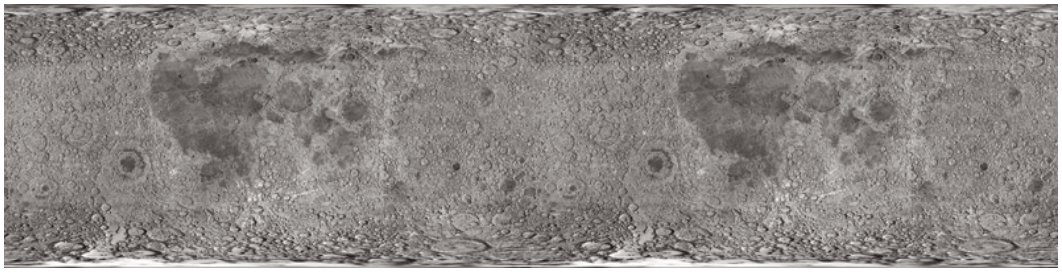
```
private function imageLoadComplete(event:Event):void
{
    textureMap = event.target.content.bitmapData;
    ...
}
```

이벤트 객체 매개 변수의 이름은 event이고 이는 Event 클래스의 인스턴스입니다. Event 클래스의 모든 인스턴스에는 target 속성이 있으며 이 속성은 이 이벤트를 트리거하는 객체(이 경우 앞에서 설명한 대로 addEventListener() 메서드가 호출된 LoaderInfo 인스턴스)를 참조합니다. 또한 LoaderInfo 객체에는 content 속성이 있는데 로드 프로세스가 완료되면 이 속성에는 로드된 비트맵 이미지가 들어 있는 Bitmap 인스턴스가 포함됩니다. 이미지를 화면에 바로 표시하려면 이 Bitmap 인스턴스(event.target.content)를 표시 객체 컨테이너에 연결하면 됩니다. 또한 Loader 객체를 표시 객체 컨테이너에 연결할 수도 있습니다. 그러나 이 샘플에서는 로드된 내용이 화면에 표시되지 않고 원시 이미지 데이터의 소스로 사용됩니다. 결과적으로 imageLoadComplete() 메서드의 첫 번째 줄은 로드된 Bitmap 인스턴스의 bitmapData 속성(event.target.content.bitmapData)을 읽어 이를 textureMap이라는 인스턴스 변수에 저장합니다. 이 변수는 회전하는 달의 애니메이션을 만들기 위한 이미지 데이터의 소스로 사용됩니다. 이 내용에 대해서는 다음 단원에서 설명합니다.

픽셀을 복사하여 애니메이션 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

애니메이션의 기본 정의는 이미지를 시차를 두고 계속 바꾸어서 움직이거나 변화하는 듯한 효과를 주는 것입니다. 이 샘플의 목표는 세로 축을 중심으로 회전하는 둥근 달의 효과를 만드는 것입니다. 그러나 애니메이션의 목적을 위해 샘플의 구형 왜곡 측면은 무시할 수 있습니다. 달 이미지 데이터의 소스로 로드되고 사용된 실제 이미지를 살펴 보십시오.



보는 바와 같이 이미지는 한 개나 여러 개의 구가 아니라 사각형의 달 표면 사진입니다. 이 사진은 정확하게 달의 적도에서 촬영되었기 때문에 이미지의 맨 위쪽과 아래쪽으로 가까울수록 이미지가 확장되고 왜곡되어 있습니다. 이미지에서 왜곡 현상을 제거하고 구형으로 보이도록 하기 위해 뒷부분에 설명된 대로 위치 변경 맵 필터를 사용할 것입니다. 그러나 소스 이미지가 사각형이기 때문에 구가 회전하는 효과를 만들기 위해 코드는 단지 달 표면 사진을 가로로 움직이기만 하면 됩니다.

이미지에는 실제로 달 표면 사진의 복사본 두 개가 서로 나란히 포함되어 있습니다. 이 이미지는 움직이는 모양을 만들기 위해 이미지 데이터가 반복적으로 복사된 것의 소스 이미지입니다. 이미지의 복사본 두 개가 서로 나란히 있으면 끊기지 않는 연속적인 스크롤 효과를 쉽게 만들 수 있습니다. 애니메이션의 프로세스를 단계별로 진행하여 애니메이션이 어떻게 작동하는지 살펴보겠습니다.

이 프로세스에는 실제로 두 개의 별도 ActionScript 객체가 사용됩니다. 먼저 로드된 소스 이미지가 있는데 이 이미지는 코드에서 textureMap이라는 BitmapData 인스턴스로 표현됩니다. 앞에서 설명한 대로 textureMap은 외부 이미지가 로드되는 즉시 다음 코드를 사용하여 이미지 데이터로 채워집니다.

```
textureMap = event.target.content.bitmapData;
```

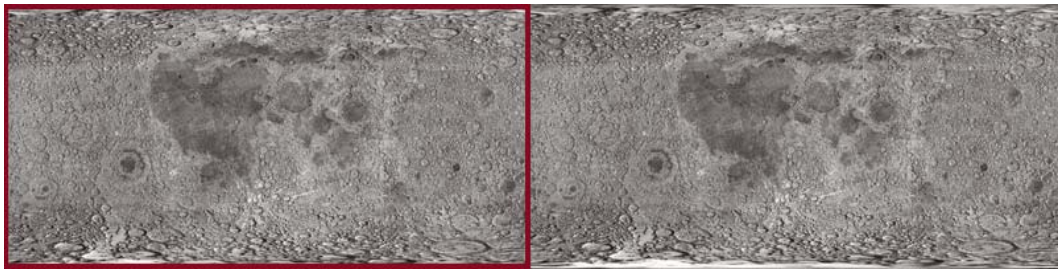
textureMap의 내용은 사각형 달 이미지입니다. 또한 회전 애니메이션을 만들기 위해 코드에서는 sphere라는 Bitmap 인스턴스를 사용하는데 이는 달 이미지를 화면에 보여 주는 실제 표시 객체입니다. textureMap처럼 sphere 객체는 imageLoadComplete() 메서드에서 다음 코드를 사용하여 만들어지고 초기 이미지 데이터로 채워집니다.

```
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));
```

코드에 표시된 대로 sphere가 인스턴스화됩니다. sphere에 의해 표시되는 원시 이미지 데이터인 bitmapData 속성은 textureMap과 같은 높이에 절반의 너비로 생성됩니다. 즉, textureMap 이미지에는 두 개의 달 사진이 나란히 있으므로 sphere의 내용은 한 장의 달 사진 크기가 됩니다. 다음으로 bitmapData 속성은 copyPixels() 메서드를 사용하여 이미지 데이터로 채워집니다. copyPixels() 메서드 호출에서 매개 변수는 다음과 같은 사항을 나타냅니다.

- 첫 번째 매개 변수는 이미지 데이터가 textureMap에서 복사된다는 것을 나타냅니다.
- 두 번째 매개 변수인 새 Rectangle 인스턴스는 textureMap의 어느 부분에서 이미지 스냅샷을 가져올 지 지정합니다. 이 경우에 스냅샷은 textureMap의 왼쪽 위 모서리(Rectangle()의 처음 두 매개 변수인 0, 0으로 표시됨)에서 시작하는 사각형이고, 사각형 스냅샷의 폭과 높이는 sphere의 width 및 height 속성과 일치합니다.
- 세 번째 매개 변수인 새 Point 인스턴스는 x와 y 값이 0이며 픽셀 데이터의 대상을 정의합니다. 이 경우에 대상은 sphere.bitmapData의 왼쪽 위 모서리 (0, 0)입니다.

시각적으로 표시하면 코드에서는 다음 이미지에서 외곽선이 표시된 textureMap에서 픽셀을 복사하여 sphere로 붙여 넣습니다. 즉, sphere의 BitmapData 내용은 다음에서 강조 표시된 textureMap의 일부분입니다.



그러나 이것은 sphere의 초기 상태, 즉 sphere로 복사된 첫 번째 이미지 내용일 뿐입니다.

소스 이미지가 로드되고 sphere가 만들어지면 imageLoadComplete() 메서드가 수행하는 마지막 작업은 애니메이션을 설정하는 것입니다. 애니메이션은 rotationTimer라는 Timer 인스턴스에 의해 작동되는데 이는 다음 코드에 의해 만들어지고 시작됩니다.

```
var rotationTimer:Timer = new Timer(15);
rotationTimer.addEventListener(TimerEvent.TIMER, rotateMoon);
rotationTimer.start();
```

코드에서는 먼저 rotationTimer라는 Timer 인스턴스를 만듭니다. Timer() 생성자로 전달되는 매개 변수는 rotationTimer가 15밀리초마다 자신의 timer 이벤트를 트리거해야 한다는 것을 나타냅니다. 다음으로 timer 이벤트(TimerEvent.TIMER)가 발생하면 rotateMoon() 메서드를 호출하도록 지정하여 addEventListener() 메서드를 호출합니다. 마지막으로 start() 메서드를 호출하여 타이머가 실제로 시작됩니다.

rotationTimer가 정의된 방식으로 인해 약 15밀리초마다 Flash Player가 MoonSphere 클래스의 rotateMoon() 메서드를 호출하며 여기에서 달의 애니메이션이 발생합니다. rotateMoon() 메서드의 소스 코드는 다음과 같습니다.

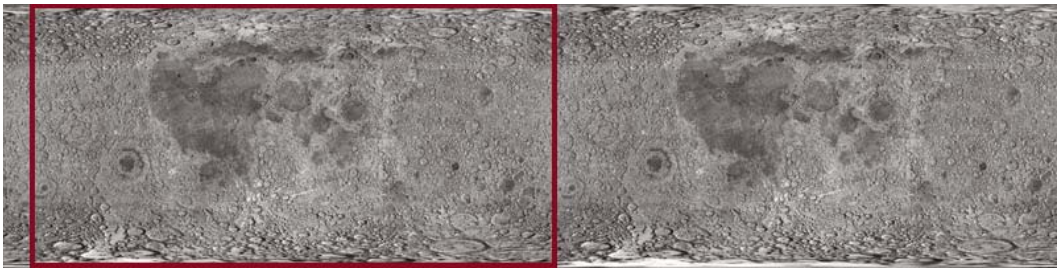
```
private function rotateMoon(event:TimerEvent):void
{
    sourceX += 1;
    if (sourceX > textureMap.width / 2)
    {
        sourceX = 0;
    }

    sphere.Data.copyPixels(textureMap,
                                    new Rectangle(sourceX, 0, sphere.width, sphere.height),
                                    new Point(0, 0));

    event.updateAfterEvent();
}
```

이 코드는 다음과 같은 세 가지 작업을 수행합니다.

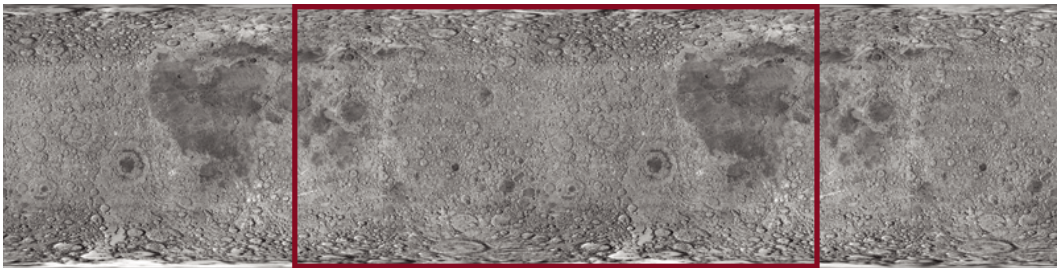
1 처음에 0으로 설정된 변수 sourceX의 값이 1씩 증가합니다.

```
sourceX += 1;
```

sourceX는 sphere로 픽셀을 복사할 textureMap 내부의 위치를 결정하는 데 사용되므로 이 코드는 textureMap에서 사각형을 한 픽셀만큼 오른쪽으로 이동하는 효과가 있습니다. 시각적 표현으로 돌아가면, 애니메이션 주기를 몇 번 거친 후 소스 사각형은 다음과 같이 오른쪽으로 몇 픽셀만큼 이동하게 됩니다.

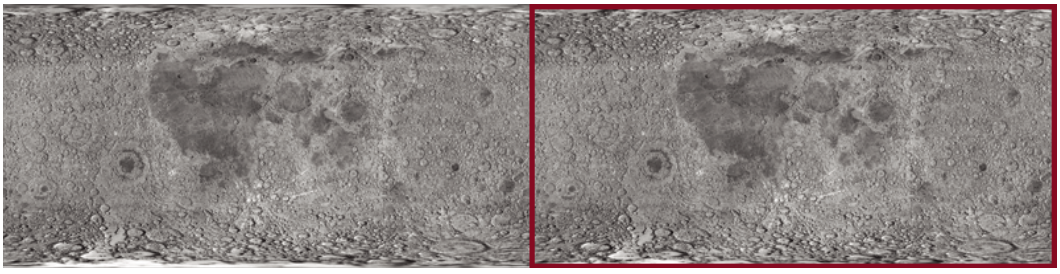


주기가 몇 번 더 지나면 사각형은 더 멀리 이동하게 됩니다.



이렇게 픽셀이 복사되는 위치가 조금씩 일정하게 이동하는 것이 이 애니메이션의 핵심입니다. 소스 위치를 천천히 계속해서 오른쪽으로 이동함으로써 sphere에서 화면에 표시되는 이미지는 계속 미끄러지듯이 왼쪽으로 이동하는 것처럼 보입니다. 소스 이미지(textureMap)에 두 개의 달 표면 사진 복사본이 필요한 이유가 여기에 있습니다. 사각형이 계속 오른쪽으로 이동하므로 대부분 사각형이 한 개의 달 사진이 아니라 두 개의 달 사진에 걸쳐 있게 됩니다.

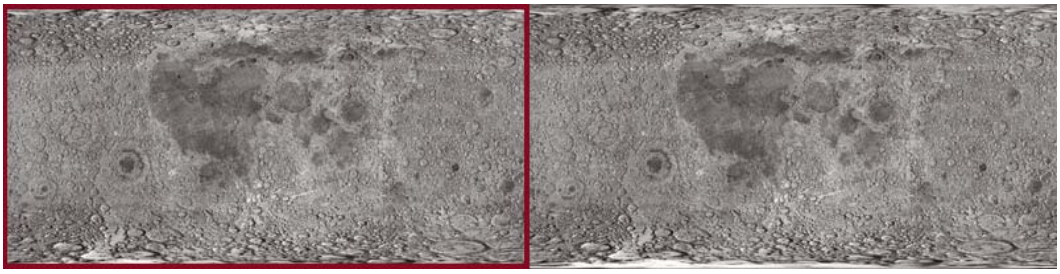
2 소스 사각형이 오른쪽으로 천천히 이동함에 따라 한 가지 문제가 발생합니다. 결국 사각형이 textureMap의 오른쪽 가장자리에 도달하면 sphere로 복사할 달 사진 픽셀이 부족하게 됩니다.



다음 코드 줄이 이 문제를 해결합니다.

```
if (sourceX >= textureMap.width / 2)
{
    sourceX = 0;
}
```

코드에서는 sourceX(사각형의 왼쪽 가장자리)가 textureMap의 한가운데에 도달했는지 확인합니다. 한가운데에 도달한 경우 sourceX가 0으로 다시 설정되므로 textureMap의 왼쪽 가장자리로 돌아가서 주기가 다시 시작됩니다.



3 적절한 sourceX 값이 계산되면 애니메이션 만들기의 마지막 단계는 새 소스 사각형 픽셀을 실제로 sphere에 복사하는 것입니다. 이 작업을 수행하는 코드는 앞에서 설명한 sphere를 처음 채우는 코드와 비슷합니다. 이 경우 유일한 차이점은 new Rectangle() 생성자 호출에서 사각형의 왼쪽 가장자리가 sourceX에 위치하는 것뿐입니다.

```
sphere.bitmapData.copyPixels(textureMap,
                             new Rectangle(sourceX, 0, sphere.width, sphere.height),
                             new Point(0, 0));
```

이 코드는 15밀리초마다 반복해서 호출된다는 점에 유의하십시오. 소스 사각형의 위치가 계속 이동하고 픽셀이 sphere로 복사됨에 따라 화면에는 sphere가 나타내는 달 사진 이미지가 미끄러지듯 계속 이동하는 것으로 나타납니다. 즉, 달이 계속해서 회전하는 것처럼 보입니다.

구형 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

달은 당연히 구형이며 사각형이 아닙니다. 따라서 샘플에서는 지속적으로 움직이는 사각형 달 표면 사진을 가져와서 구형으로 변환해야 합니다. 이 작업은 별도의 두 단계로 이루어집니다. 마스크를 사용하여 달 표면 사진의 원형 영역을 제외한 모든 내용을 숨기고, 위치 변경 맵 필터를 사용하여 3차원으로 보이도록 달 사진의 모양을 왜곡합니다.

먼저 원형 마스크를 사용하여 필터에 의해 생성된 구를 제외한 MoonSphere 객체의 내용을 모두 숨깁니다. 다음 코드는 마스크를 Shape 인스턴스로 만들어서 MoonSphere 인스턴스의 마스크로 적용합니다.

```
moonMask = new Shape();
moonMask.graphics.beginFill(0);
moonMask.graphics.drawCircle(0, 0, radius);
this.addChild(moonMask);
this.mask = moonMask;
```

Sprite 클래스에 기반을 둔 MoonSphere는 표시 객체이므로 상속된 mask 속성을 사용하여 마스크를 MoonSphere 인스턴스에 직접 적용할 수 있습니다.

원형 마스크를 사용하여 사진의 일부를 숨기는 것만으로는 실제처럼 보이는 회전하는 구의 효과를 충분히 낼 수 없습니다. 달 표면 사진의 촬영 방식으로 인해 사진의 치수가 비례적이지 않으며 적도 부분과 비교해 이미지의 맨 위쪽이나 아래쪽으로 갈수록 이미지가 더욱 왜곡되고 늘어납니다. 달 사진의 모양을 왜곡하여 3차원으로 보이도록 하기 위해 위치 변경 맵 필터를 사용합니다.

위치 변경 맵 필터는 이미지를 왜곡하는 데 사용하는 필터 유형입니다. 이 경우 달 사진이 좀 더 사실적으로 보이도록 이미지의 가운데 부분은 바꾸지 않고 그대로 둔 채 맨 위쪽과 맨 아래쪽을 가로로 눌러서 달 사진을 "왜곡"합니다. 필터가 사진의 사각형 부분에 작동한다고 가정하고 가운데를 제외하고 맨 위쪽과 아래쪽을 누르면 사각형이 원형으로 변합니다. 이렇게 왜곡된 이미지에 애니메이션 효과를 적용하면 이미지의 가운데 부분이 맨 위쪽과 아래쪽에 가까운 부분보다 실제 픽셀 거리에서 더 멀리 이동하는 것처럼 보이므로 원이 실제로 3차원 객체(구)처럼 보이게 됩니다.

다음 코드는 displaceFilter라고 하는 위치 변경 맵 필터를 만드는 데 사용됩니다.

```
var displaceFilter:DisplacementMapFilter;
displaceFilter = new DisplacementMapFilter(fisheyeLens,
                                new Point(radius, 0),
                                BitmapDataChannel.RED,
                                BitmapDataChannel.GREEN,
                                radius, 0);
```

첫 번째 매개 변수인 fisheyeLens는 맵 이미지로 알려져 있습니다. 이 경우에는 프로그래밍 방식으로 작성되는 BitmapData 객체입니다. 해당 이미지를 만드는 방법은 237페이지의 "픽셀 값을 설정하여 비트맵 이미지 만들기"에서 설명합니다. 다른 매개 변수는 필터링된 이미지에서 필터가 적용될 위치, 위치 변경 효과를 조정하는 데 사용될 색상 채널 및 색상 채널이 위치 변경에 미치는 영향의 정도에 대해 설명합니다. 위치 변경 맵 필터가 만들어지면 이 필터는 imageLoadComplete() 메서드 내부에서 sphere에 적용됩니다.

```
sphere.filters = [displaceFilter];
```

마스크 및 위치 변경 맵 필터가 적용된 최종 이미지는 다음과 같은 모양입니다.

회전하는 달 애니메이션의 매 주기마다 소스 이미지 데이터의 새로운 스냅샷이 구의 BitmapData 내용을 덮어씁니다. 그러나 필터는 매번 다시 적용되지 않아도 됩니다. 필터가 비트맵 데이터(원시 픽셀 정보)에 적용되지 않고 Bitmap 인스턴스(표시 객체)에 적용되기 때문입니다. Bitmap 인스턴스는 실제 비트맵 데이터가 아니라 비트맵 데이터를 화면에 표시하는 표시 객체라는 것에 유의하십시오. 비유하자면 Bitmap 인스턴스는 화면에 사진 슬라이드를 표시하는 데 사용되는 슬라이드 프로젝터와 비슷하고 BitmapData 객체는 슬라이드 프로젝터를 통해 나타낼 수 있는 실제 사진 슬라이드와 비슷합니다. 필터는 BitmapData 객체에 직접 적용할 수 있는데 이는 이미지를 변경하기 위해 사진 슬라이드에 직접 그림을 그리는 것과 비슷합니다. 필터를 Bitmap 인스턴스 등의 모든 표시 객체에 적용할 수도 있는데 이는 원래 슬라이드를 전혀 변경하지 않고 화면에 보여 주는 출력을 왜곡하기 위해 슬라이드 프로젝터의 렌즈 앞에 필터를 두는 것과 같습니다. 원시 비트맵 데이터는 Bitmap 인스턴스의 bitmapData 속성을 통해 액세스할 수 있으므로 필터를 원시 비트맵 데이터에 직접 적용했을 수도 있습니다. 그러나 이 경우 필터를 비트맵 데이터에 적용하지 않고 Bitmap 표시 객체에 적용하는 것이 더 좋습니다.

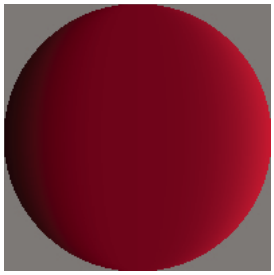
ActionScript의 위치 변경 맵 필터 사용에 대한 자세한 내용은 242페이지의 "표시 객체 필터링"을 참조하십시오.

픽셀 값을 설정하여 비트맵 이미지 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

위치 변경 맵 필터의 한 가지 중요한 점은 실제로는 두 개의 이미지가 사용된다는 것입니다. 한 이미지는 소스 이미지로 필터에 의해 실제로 변경되는 이미지입니다. 이 샘플에서 소스 이미지는 sphere라는 Bitmap 인스턴스입니다. 필터에서 사용하는 다른 이미지는 맵 이미지라고 합니다. 맵 이미지는 실제로 화면에는 표시되지 않습니다. 그 대신 각 픽셀의 색상은 위치 변경 함수의 입력으로 사용됩니다. 즉, 맵 이미지에서 특정 x, y 좌표에 있는 픽셀의 색상은 위치 변경(물리적 위치 이동)이 소스 이미지에서 해당 x, y 좌표에 있는 픽셀에 얼마나 적용되는지 결정합니다.

결과적으로 구 효과를 만드는 데 위치 변경 맵 필터를 사용하려면 샘플에 적절한 맵 이미지가 있어야 합니다. 이 이미지에는 다음과 같이 회색 배경과 어두운 색에서 밝은 색으로 가로로 나타나는 단색(빨강)의 그래디언트로 채워진 원이 있습니다.



이 샘플에서는 단 한 개의 맵 이미지와 필터를 사용하므로 맵 이미지는 외부 이미지의 로드가 완료될 때 imageLoadComplete() 메서드에서 한 번만 만들어집니다. fisheyeLens라는 맵 이미지는 MoonSphere 클래스의 createFisheyeMap() 메서드를 호출하여 만들어집니다.

```
var fisheyeLens:BitmapData = createFisheyeMap(radius);
```

createFisheyeMap() 메서드에서 맵 이미지는 BitmapData 클래스의 setPixel() 메서드를 사용하여 실제로 한 번에 한 픽셀씩 그려진 것입니다. createFisheyeMap() 메서드의 전체 코드는 다음과 같습니다. 그 다음에는 이 코드의 작동에 대해 단계별로 설명합니다.

```
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;
}
```

먼저 메서드가 호출되면 만들 원 모양 이미지의 반지름을 나타내는 radius라는 매개 변수를 받습니다. 다음으로 코드는 원이 그려질 BitmapData 객체를 만듭니다. result라는 이름의 이 객체는 결국 메서드의 반환 값으로 다시 전달됩니다. 다음 코드와 같이 result BitmapData 인스턴스는 원지름과 같은 크기의 너비와 높이로 투명하지 않게(세 번째 매개 변수에 false가 설정됨) 만들어지며 0x808080(중간 회색) 색상으로 미리 채워집니다.

```
var result:BitmapData = new BitmapData(diameter,
                                       diameter,
                                       false,
                                       0x808080);
```

다음으로 코드는 두 개의 루프를 사용하여 이미지의 각 픽셀을 반복 실행합니다. 바깥쪽 루프가 이미지의 각 열을 왼쪽에서 오른쪽으로 이동하는(i 변수를 사용하여 현재 처리 중인 픽셀의 가로 위치 표시) 반면 안쪽 루프는 현재 열의 각 픽셀을 맨 위쪽에서 아래쪽으로(j 변수를 사용하여 현재 픽셀의 세로 위치 표시) 이동합니다. 이 루프의 코드는 다음과 같습니다(안쪽 루프의 내용 생략).

```
for (var i:int = 0; i < diameter; i++)
{
    for (var j:int = 0; j < diameter; j++)
    {
        ...
    }
}
```

루프가 픽셀을 하나씩 차례로 이동하면 각 픽셀에서 값(맵 이미지에서 해당 픽셀의 색상 값)이 계산됩니다. 이 프로세스는 4단계로 이루어집니다.

1. 코드가 x축을 따라 원의 중심에서부터 현재 픽셀의 거리를 계산합니다(i - radius). 이 값을 반지름으로 나누어 절대적인 거리가 아닌 반지름의 백분율((i - radius) / radius)로 만듭니다. 다음 코드에서와 같이 이 백분율 값은 pctX라는 변수에 저장되고 y축에 해당하는 값은 계산되어 pctY 변수에 저장됩니다.

   ```
   var pctX:Number = (i - radius) / radius;
   var pctY:Number = (j - radius) / radius;
   ```

2. 피타고라스의 원리인 표준 삼각법 공식을 사용하여 원의 중심에서 현재 지점 간의 직선 거리가 pctX와 pctY에서 계산됩니다. 이 값은 다음과 같이 pctDistance 라는 변수에 저장됩니다.

   ```
   var pctDistance:Number = Math.sqrt(pctX * pctX + pctY * pctY);
   ```

3. 다음으로 거리 백분율이 1보다 작은지 확인합니다. 1은 반지름의 100%라는 뜻이므로 검토 중인 픽셀이 원의 반지름 내에 있는지 확인하는 것과 같습니다. 픽셀이 원 내부에 있는 경우 계산된 색상 값(여기서는 생략되었지만 4단계에서 설명함)이 지정됩니다. 픽셀이 원 밖에 있는 경우에는 픽셀은 더 이상 변경되지 않으며 기본값인 중간 회색으로 유지됩니다.

```
if (pctDistance < 1)
{
    ...
}
```

4 원의 내부에 있는 픽셀의 경우 픽셀에 대한 색상 값이 계산됩니다. 최종 색상은 원 왼쪽 가장자리의 검정(빨강 0%)에서 원 오른쪽 가장자리의 밝은 빨강(100%) 사이의 빨강 음영이 됩니다. 색상 값은 처음에는 다음과 같이 빨강, 녹색 및 파랑의 세 부분으로 계산됩니다.

```
red = 128 * (1 + 0.75 * pctX * pctX * pctX / (1 - pctY * pctY));
green = 0;
blue = 0;
```

색상의 빨강 부분(red 변수)만 실제로 값을 가지고 있습니다. 명확성을 위해 여기에서 녹색 및 파랑 값(green 및 blue 변수)을 표시했지만 해당 값은 생략할 수 있습니다. 이 메서드의 목적은 빨강 음영을 포함하는 원을 만드는 것이므로 녹색이나 파랑 변수는 필요하지 않습니다.

세 가지 개별 색상 값이 각각 결정되면 이들은 다음 코드와 같이 표준 비트 이동 알고리즘을 사용하여 단일 정수 색상 값으로 통합됩니다.

```
rgb = (red << 16 | green << 8 | blue);
```

마지막으로 색상 값이 계산되면 이 값은 다음과 같이 result BitmapData 객체의 setPixel() 메서드를 사용하여 실제로 현재 픽셀에 지정됩니다.

```
result.setPixel(i, j, rgb);
```

비트맵 이미지의 비동기 디코딩

Flash Player 11 이상, Adobe AIR 2.6 이상

비트맵 이미지를 사용하여 작업하는 경우에 비트맵 이미지를 비동기적으로 디코딩하고 로드하여 응용 프로그램의 인지 성능을 향상시킬 수 있습니다. 비트맵 이미지를 비동기적으로 디코딩하면 대부분의 경우 동기적으로 디코딩할 때와 동일한 시간이 소요될 수 있습니다. 하지만 연결된 Loader 객체에서 COMPLETE 이벤트를 보내기 전에 비트맵 이미지가 별도의 스레드에서 디코딩됩니다. 따라서 큰 이미지는 로드한 후에 비동기적으로 디코딩할 수 있습니다.

flash.system 패키지의 ImageDecodingPolicy 클래스를 사용하면 비트맵 로드 스킴을 지정할 수 있습니다. 로드 스킴의 기본값은 동기 방식입니다.

비트맵 디코딩 정책	비트맵 로드 스킴	설명
ImageDecodingPolicy.ON_DEMAND	동기	로드된 이미지는 이미지 데이터에 액세스할 때 디코딩됩니다. 이 정책은 작은 이미지를 디코딩할 때 사용합니다. 응용 프로그램에서 복잡한 효과나 전환 효과에 의존하지 않는 경우에도 사용할 수 있습니다.
ImageDecodingPolicy.ON_LOAD	비동기	로드된 이미지는 COMPLETE 이벤트가 전달되기 전에 로드 시 디코딩됩니다. 이 정책은 큰 이미지(10MP 초과)에 사용하는 것이 가장 좋습니다. 페이지 전환 효과가 있는 AIR 기반 모바일 응용 프로그램을 개발하는 경우 이 비트맵 로드 정책을 사용하면 응용 프로그램의 인지 성능이 향상됩니다.

참고: 로드되는 파일이 비트맵 이미지이고 사용된 디코딩 정책이 ON_LOAD인 경우 이미지는 COMPLETE 이벤트가 전달되기 전에 비동기적으로 디코딩됩니다.

다음 코드는 ImageDecodingPolicy 클래스의 사용을 보여 줍니다.

```
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);
```

ON_DEMAND 디코딩은 여전히 Loader.load() 및 Loader.loadBytes() 메서드와 함께 사용할 수 있습니다. 하지만 LoaderContext 객체를 인수로 받는 다른 모든 메서드에서는 전달되는 모든 ImageDecodingPolicy 값을 무시합니다.

다음 예제는 비트맵 이미지의 동기식 디코딩과 비동기식 디코딩 간 차이를 보여 줍니다.

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

다양한 디코딩 정책의 효과를 보여 주는 데모를 확인하려면 Thibaud Imbert: Adobe Flash 런타임의 비동기 비트맵 디코딩을 참조하십시오.

14장: 표시 객체 필터링

Flash Player 9 이상, Adobe AIR 1.0 이상

이전에는 비트맵 이미지에 필터 효과를 적용하는 것은 Adobe Photoshop®과 Adobe Fireworks® 같은 특수한 이미지 편집 소프트웨어 영역으로 한정되어 있었습니다. ActionScript 3.0에는 일련의 비트맵 효과 필터 클래스를 포함하는 flash.filters 패키지가 들어 있습니다. 이러한 효과를 사용하면 개발자가 프로그래밍 방식으로 비트맵과 표시 객체에 필터를 적용할 수 있으므로 그래픽 조작 응용 프로그램에서 사용할 수 있는 것과 동일한 효과를 낼 수 있습니다.

표시 객체 필터링의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램을 한층 더 세련되게 만드는 방법 중 하나는 간단한 그래픽 효과를 추가하는 것입니다. 예를 들어 사진 뒤에 그림자를 추가하여 3차원 효과를 내거나 버튼 주위에 빛 효과를 주어 해당 버튼이 활성화되어 있음을 알릴 수 있습니다. ActionScript 3.0은 모든 표시 객체 또는 BitmapData 인스턴스에 적용 가능한 10개의 필터를 내장하고 있는데, 여기에는 그림자 및 광선 필터 등과 같은 기본 필터에서 위치 변경 맵 필터 및 회선 필터와 같은 복잡한 필터가 포함됩니다.

참고: 내장된 기본 제공 필터뿐 아니라, Pixel Bender를 사용하여 사용자 정의 필터 및 효과를 직접 프로그래밍할 수도 있습니다. 자세한 내용은 271페이지의 “Pixel Bender 셰이더를 사용한 작업”을 참조하십시오.

중요한 개념 및 용어

다음 참조 목록에는 필터를 만들 때 사용되는 중요한 용어가 정리되어 있습니다.

경사 두 면의 픽셀은 밝게 하고 반대편 두 면의 픽셀은 어둡게 하여 만들어진 가장자리로, 위로 솟은 버튼 또는 안으로 살짝 들어간 버튼이나 이와 유사한 그래픽을 만들 때 자주 사용되는 3차원 테두리 효과를 연출합니다.

회선 다양한 비율을 적용하여 각 픽셀의 값을 인접한 픽셀 중 일부 또는 전부의 값과 결합하여 이미지의 픽셀을 왜곡시키는 것입니다.

위치 변경 이미지의 픽셀을 새 위치로 옮기는 것입니다.

Matrix 격자의 수를 다양한 값에 적용한 다음 결과를 결합하여 특정 수학적 계산을 수행하는 데 사용되는 숫자 격자입니다.

기타 도움말 항목

flash.filters 패키지

flash.display.DisplayObject.filters

flash.display.BitmapData.applyFilter()

필터 작성 및 적용

Flash Player 9 이상, Adobe AIR 1.0 이상

필터를 사용하면 비트맵 및 표시 객체에 그림자, 경사 및 흐림 효과 등 다양한 효과를 적용할 수 있습니다. 각 필터는 클래스로 정의되므로 필터를 적용하면 필터 객체의 인스턴스가 기타 다른 객체가 생성되는 것과 마찬가지 방식으로 만들어집니다. 일단 필터 객체를 만든 후에는 해당 필터를 객체의 filters 속성이나, BitmapData 객체의 경우 applyFilter() 메서드를 사용하여 간단하게 표시 객체에 적용할 수 있습니다.

필터 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

필터 객체를 만들려면 선택한 필터 클래스의 생성자 메서드를 호출하면 됩니다. 예를 들어 DropShadowFilter 객체를 만들려면 다음 코드를 사용하십시오.

```
import flash.filters.DropShadowFilter;
var myFilter:DropShadowFilter = new DropShadowFilter();
```

여기에 표시되어 있지는 않지만 모든 필터 클래스 생성자 등의 DropShadowFilter() 생성자는 필터 효과 모양 사용자 정의에 사용할 수 있는 여러 선택적 매개 변수를 사용합니다.

필터 적용

Flash Player 9 이상, Adobe AIR 1.0 이상

생성한 필터 객체는 표시 객체 또는 BitmapData 객체에 적용할 수 있습니다. 필터 적용 방법은 필터를 적용하는 객체에 따라 달라집니다.

표시 객체에 필터 적용

표시 객체에 필터 효과를 적용하는 경우에는 filters 속성을 통해 적용합니다. 표시 객체의 filters 속성은 Array 인스턴스이며, 그 요소는 표시 객체에 적용된 필터 객체입니다. 표시 객체에 단일 필터를 적용하려면 다음과 같이 필터 인스턴스를 만들고 이를 Array 인스턴스에 추가한 다음 표시 객체의 filters 속성에 Array 객체를 지정합니다.

```
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;
```

객체에 여러 필터를 지정하려면 filters 속성에 Array 인스턴스를 지정하기 전에 먼저 모든 필터를 해당 인스턴스에 추가합니다. Array 생성자에 여러 객체를 매개 변수로 전달하면 해당 객체를 Array에 추가할 수 있습니다. 예를 들어 다음 코드는 이전에 만든 표시 객체에 경사 필터 및 광선 필터를 적용합니다.

```
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;
```

필터를 포함하는 배열을 만드는 경우에는 이전 예제와 같이 new Array() 생성자를 사용하여 만들거나 Array 리터럴 구문을 사용하여 해당 필터를 대괄호([])로 묶습니다. 다음 코드 행을 예로 들어 보겠습니다.

```
var filters:Array = new Array(dropShadow, blur);
```

위 코드 행은 아래 코드 행과 동일한 작업을 수행합니다.

```
var filters:Array = [dropShadow, blur];
```

표시 객체에 여러 개의 필터를 적용할 경우 필터는 누적형의 순차적인 방식으로 적용됩니다. 예를 들어 두 개의 요소가 있는 필터 배열의 경우 먼저 경사 필터 그 다음으로 그림자 필터가 추가되며, 그림자 필터는 경사 필터와 표시 객체 모두에 적용됩니다. 그 이유는 그림자 필터가 필터 배열에서 두 번째 위치에 있기 때문입니다. 비 누적 방식으로 필터를 적용하려면 표시 객체의 새 복사본에 각각의 필터를 적용해야 합니다.

표시 객체에 한 개 또는 소수의 필터만 지정하려는 경우 필터 인스턴스를 만든 다음 단일 명령문을 사용하여 객체에 해당 인스턴스를 지정할 수 있습니다. 예를 들어 다음 코드 행은 흐림 필터를 myDisplayObject라는 표시 객체에 적용합니다.

```
myDisplayObject.filters = [new BlurFilter()];
```

이전 코드는 Array 리터럴 구문(각괄호)을 사용하여 Array 인스턴스를 만들고 Array의 요소로 BlurFilter 인스턴스를 만든 다음 해당 Array를 myDisplayObject라는 표시 객체의 filters 속성에 지정합니다.

표시 객체에서 필터 제거

표시 객체에서 모든 필터를 제거하려면 다음과 같이 단순히 filters 속성에 null 값을 지정하면 됩니다.

```
myDisplayObject.filters = null;
```

객체에 여러 객체를 적용한 경우 하나의 필터만 제거하려면 여러 단계를 거쳐 filters 속성 배열을 변경해야 합니다. 자세한 내용은 245페이지의 “필터 작업의 잠재적 문제점”을 참조하십시오.

BitmapData 객체에 필터 적용

BitmapData 객체에 필터를 적용하려면 BitmapData 객체의 applyFilter() 메서드를 사용해야 합니다.

```
var rect:Rectangle = new Rectangle();
var origin:Point = new Point();
myBitmapData.applyFilter(sourceBitmapData, rect, origin, new BlurFilter());
```

applyFilter() 메서드는 필터를 BitmapData 객체에 적용하여 필터링된 새 이미지를 생성합니다. 이 메서드는 원본 소스 이미지를 수정하지 않는 대신 소스 이미지에 적용된 필터 결과가 applyFilter() 메서드를 호출하는 BitmapData 인스턴스에 저장됩니다.

필터 작동 방법

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 객체 필터링은 원본 객체의 복사본을 투명 비트맵으로 캐싱하는 방식으로 작동됩니다.

표시 객체에 필터를 적용한 후에는 런타임에서 해당 객체에 유효한 필터 목록이 있는 한 해당 객체를 비트맵으로 캐시합니다. 이 소스 비트맵은 이후에 적용되는 모든 필터 효과의 원본 이미지로 사용됩니다.

일반적으로 각 표시 객체에는 두 개의 비트맵이 포함되어 있는데, 한 개는 필터링되지 않은 원본 소스 표시 객체에 대한 비트맵이고 다른 한 개는 필터링 이후의 최종 이미지에 대한 비트맵입니다. 최종 이미지는 렌더링 시 사용됩니다. 표시 객체가 변경되지 않으면 업데이트할 필요가 없습니다.

필터 작업의 잠재적 문제점

Flash Player 9 이상, Adobe AIR 1.0 이상

필터를 사용하여 작업할 경우 혼동하거나 문제가 발생되지 않도록 유념해야 할 몇 가지 사항이 있습니다.

필터 및 비트맵 캐싱

표시 객체에 필터를 적용하기 위해서는 해당 객체에 대한 비트맵 캐싱이 활성화되어야 합니다. cacheAsBitmap 속성이 false로 설정된 표시 객체에 필터를 적용하는 경우 해당 객체의 cacheAsBitmap 속성은 자동으로 true로 설정됩니다. 나중에 표시 객체에서 모든 필터를 제거하면 cacheAsBitmap 속성이 마지막으로 설정된 값으로 다시 설정됩니다.

런타임에 필터 변경

표시 객체에 이미 하나 이상의 필터가 적용된 경우에는 filters 속성 배열에 다른 필터를 추가하거나 제거하여 필터 세트를 변경할 수 없습니다. 대신 적용되는 필터 세트에 추가하거나 변경하려면 개별 배열을 변경한 다음 객체에 적용할 필터에 대한 표시 객체의 필터 속성에 해당 배열을 할당합니다. 이 작업을 수행하는 가장 간단한 방법은 filters 속성 배열을 Array 변수로 읽어온 다음 이 임시 배열을 수정하는 것입니다. 표시 객체의 filters 속성에 이 배열을 다시 할당합니다. 보다 복잡한 경우 별도의 마스터 필터 배열을 유지해야 할 수도 있습니다. 마스터 필터 배열을 변경한 다음 각 변경 후에 표시 객체의 filters 속성에 마스터 배열을 다시 할당합니다.

다른 필터 추가

다음 코드에서는 한 개 이상의 필터가 이미 적용되어 있는 표시 객체에 다른 필터를 추가하는 프로세스를 보여 줍니다. 처음에는 광선 필터가 myDisplayObject라는 표시 객체에 적용되고 나중에 해당 표시 객체를 클릭하면 addFilters() 함수가 호출됩니다. 이 함수에서 다음과 같이 두 개의 추가 필터가 myDisplayObject에 적용됩니다.

```
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);
```

필터 세트에서 필터 한 개 제거

표시 객체에 여러 개의 필터가 적용되어 있고 다른 필터는 계속 객체에 적용하는 동시에 필터 중 한 개를 제거하려면 임시 배열에 필터를 복사하고 해당 배열에서 불필요한 필터를 제거한 다음 표시 객체의 filters 속성에 임시 배열을 다시 할당합니다. 임의 배열에서 요소를 한 개 이상 제거하는 여러 가지 방법은 28페이지의 "값 가져오기 및 배열 요소 제거"에 설명되어 있습니다.

가장 간단한 경우는 객체의 맨 위에 있는 필터(객체에 마지막으로 적용된 필터)를 제거하는 것입니다. Array 클래스의 pop() 메서드를 사용하여 배열에서 필터를 제거합니다.

```
// 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;
```

마찬가지로, 맨 아래에 있는 필터(객체에 적용된 첫 번째 필터)를 제거하려면 동일한 코드를 사용하고 Array 클래스의 shift() 메서드로 pop() 메서드를 대체합니다.

필터 배열의 가운데에서 필터를 제거하려면(배열에 필터가 세 개 이상 있다고 가정) splice() 메서드를 사용할 수 있습니다. 제거할 필터의 인덱스(배열에서의 위치)를 알고 있어야 합니다. 예를 들어, 다음 코드는 표시 객체에서 두 번째 필터(인덱스 1의 필터)를 제거합니다.

```
// 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;
```

필터의 인덱스 확인

필터의 인덱스를 확인하려면 배열에서 제거할 필터를 알고 있어야 합니다. 응용 프로그램의 디자인 방법에 의해 제거할 필터의 인덱스를 알고 있거나 계산해야 합니다.

제거할 필터가 항상 필터 세트의 동일한 위치에 있도록 응용 프로그램을 디자인하는 것이 좋습니다. 예를 들어 회선 필터와 그림자 필터가 이 순서대로 적용되어 있는 단일 표시 객체가 있고 그림자 필터만 제거하고 회선 필터를 유지하려는 경우 필터가 알려진 위치(맨 위 필터)에 있기 때문에 사용할 Array 메서드(이 경우 그림자 필터를 제거하기 위해 Array.pop() 사용)를 미리 알 수 있습니다.

제거할 필터가 항상 특정 유형이지만 필터 세트에서 항상 동일한 위치에 있는 것이 아니면 배열에서 각 필터의 데이터 유형을 검사하여 제거할 필터를 확인할 수 있습니다. 예를 들어, 다음 코드는 필터 세트 중 광선 필터인 필터를 확인하고 해당 필터를 세트에서 제거합니다.

```
// 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;
```

제거할 필터가 런타임에 선택되는 경우처럼 보다 복잡한 경우에서는 마스터 필터 목록으로 동작하는 별도의 필터 배열 영구 복사본을 유지하는 것이 좋습니다. 언제든지 필터 세트를 변경하고 마스터 목록을 변경한 다음 해당 필터 배열을 표시 객체의 filters 속성으로 적용할 수 있습니다.

예를 들어, 다음 코드 샘플에서는 여러 개의 회선 필터가 표시 객체에 적용되어 다양한 시각적 효과를 만든 다음, 응용 프로그램의 이후 지점에서 이러한 필터 중 한 개가 제거되고 다른 필터는 유지됩니다. 이 경우 코드는 제거할 필터에 대한 참조는 물론 필터 배열의 마스터 복사본을 유지합니다. 특정 필터를 찾아서 제거하는 방법은 필터 배열의 임시 복사본을 만드는 대신 마스터 복사본을 조작한 다음 표시 객체에 적용한다는 점을 제외하고 이전 방법과 유사합니다.

```
// 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;
}
```

이 방법에서 필터 배열의 항목에 대해 저장된 필터 참조를 비교하여 제거할 필터를 확인하는 경우 **반드시** 별도의 필터 배열 복사본을 유지해야 합니다. 표시 객체의 filters 속성에서 복사된 임시 배열의 요소에 대해 저장된 필터 참조를 비교하는 경우에는 코드가 작동하지 않습니다. 이것은 filters 속성에 배열을 할당할 때 런타임에서 내부적으로 배열에 있는 각 필터 객체의 복사본이 만들어지기 때문입니다. 원본 객체 대신 이러한 복사본이 표시 객체에 적용되며, filters 속성을 임시 배열로 읽어올 때 원본 필터 객체에 대한 참조가 아니라 복사한 필터 객체에 대한 참조가 임시 배열에 포함됩니다. 따라서 앞의 예제에서 임시 필터 배열의 필터와 비교하여 filterToRemove의 인덱스를 확인하려고 하면 일치 항목이 없습니다.

필터 및 객체 변형

표시 객체의 경계 상자 사각형 외부의 필터링되지 않은 영역(예: 그림자)은 인스턴스가 다른 인스턴스와 겹치거나 교차하는지 여부를 판단하는 히트 감지를 목적으로 하는 표면의 일부로 간주됩니다. DisplayObject 클래스의 히트 감지 메서드는 벡터에 기반을 두기 때문에 비트맵 결과에 대해서는 히트 감지를 수행할 수 없습니다. 예를 들어, 버튼 인스턴스에 경사 필터를 적용하는 경우 경사가 적용된 인스턴스 부분에 대해서는 히트 감지를 사용할 수 없습니다.

필터에서는 크기 조절, 회전 및 기울이기를 지원하지 않습니다. 필터링된 표시 객체 자체의 크기가 조절되는 경우(scaleX 및 scaleY가 100%가 아닌 경우)에는 인스턴스에 필터 효과가 적용될 때 크기가 조절되지 않습니다. 다시 말해서 인스턴스의 원래 모양에만 회전, 크기 조절 또는 기울이기가 수행되며, 필터가 적용된 인스턴스에는 회전, 크기 조절 또는 기울이기가 수행되지 않습니다.

실제 효과 또는 중첩 인스턴스를 만드는 필터를 사용하여 인스턴스에 애니메이션을 적용하고 BitmapData 클래스를 사용하여 이 효과를 나타내도록 필터에 애니메이션을 적용할 수 있습니다.

필터 및 비트맵 객체

BitmapData 객체에 필터를 적용하면 cacheAsBitmap 속성이 자동으로 true로 설정됩니다. 실제로는 이런 방식을 사용하여 필터가 원본이 아닌 객체의 복사본에 적용됩니다.

그런 다음 이 복사본은 가장 가까운 픽셀과 최대한 가깝게 기본 디스플레이의 원본 객체 위에 배치됩니다. 원본 비트맵 경계가 변경되면 필터링된 비트맵 복사본은 확장되거나 왜곡되지 않고 원본에서 다시 만들어집니다.

표시 객체의 모든 필터를 지우면 cacheAsBitmap 속성이 필터 적용 이전 상태로 재설정됩니다.

사용 가능한 표시 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 다음과 같이 표시 객체와 BitmapData 객체에 적용할 수 있는 10개의 필터 클래스가 포함되어 있습니다.

- 경사 필터(BevelFilter 클래스)
- 흐림 필터(BlurFilter 클래스)
- 그림자 필터(DropShadowFilter 클래스)
- 광선 필터(GlowFilter 클래스)
- 그래디언트 경사 필터(GradientBevelFilter 클래스)
- 그래디언트 광선 필터(GradientGlowFilter 클래스)
- 색상 행렬 필터(ColorMatrixFilter 클래스)
- 회선 필터(ConvolutionFilter 클래스)
- 위치 변경 맵 필터(DisplacementMapFilter 클래스)
- 셰이더 필터(ShaderFilter 클래스)

처음 여섯 개 필터는 하나의 특정 효과를 만들 때 사용할 수 있는 간단한 필터로 몇 가지 효과 사용자 정의 기능을 제공합니다. 이 여섯 가지 필터는 ActionScript를 사용하여 적용할 수 있으며 필터 패널을 사용하여 Flash Professional의 객체에도 적용할 수 있습니다. 따라서 ActionScript를 사용하여 필터를 적용하는 경우라도 Flash Professional을 통해 시각 인터페이스에서 여러 가지 필터 및 설정을 신속하게 시험하여 원하는 효과를 얻는 방법을 알 수 있습니다.

마지막 네 필터는 ActionScript에서만 사용할 수 있는 필터입니다. 색상 행렬 필터, 회선 필터, 위치 변경 맵 필터 및 셰이더 필터는 훨씬 다양한 유형의 효과를 만드는 데 사용할 수 있으며 하나의 효과를 내기 위해 최적화되는 대신 강력한 기능과 유연성을 제공합니다. 예를 들어 여러 행렬 값을 선택함으로써 흐리게 하기, 엠보싱, 선명하게 하기, 색상 가장자리 찾기, 변형 등의 효과를 내는 데 회선 필터를 사용할 수 있습니다.

간단한 필터 또는 복잡한 필터 모두 각각 해당 속성을 사용하여 사용자 정의할 수 있습니다. 필터 속성은 일반적으로 두 가지 방법을 통해 설정할 수 있습니다. 모든 필터의 속성은 필터 객체의 생성자에 매개 변수 값을 전달하여 설정할 수 있습니다. 또는 매개 변수 전달을 통해 필터 속성을 설정하는지 여부와 상관없이 필터 객체의 속성 값을 설정하여 나중에 해당 필터를 조정할 수 있습니다. 대부분의 예제 코드 샘플은 예제를 쉽게 따라 할 수 있도록 속성을 직접 설정하고 있습니다. 그러나 대개 필터 객체 생성자에 값을 매개 변수로 전달하면 보다 간단한 코드로 동일한 결과를 얻을 수 있습니다. 각 필터, 필터 속성 및 생성자 매개 변수에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 flash.filters 패키지 목록을 참조하십시오.

경사 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

BevelFilter 클래스를 사용하면 필터가 적용된 객체에 경사진 3D 가장자리를 추가할 수 있습니다. 이 필터는 객체의 단단한 모서리 또는 가장자리가 옆으로 경사진 것처럼 보이게 만듭니다.

BevelFilter 클래스 속성을 사용하여 경사 모양을 사용자 정의할 수 있습니다. 즉, 강조 색상 및 그림자 색상을 설정할 수 있으며 경사 가장자리 흐리게 하기, 경사 각도, 경사 가장자리 위치 등을 설정할 수 있으며 녹아웃 효과를 연출할 수도 있습니다.

다음 예제에서는 외부 이미지를 로드하여 경사 필터를 적용합니다.

```
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];
```

흐림 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

BlurFilter 클래스는 표시 객체와 객체의 내용을 칠하거나 흐리게 합니다. 흐림 효과는 객체가 초점을 벗어나 있다는 인상을 심거나 모션 흐림에서처럼 빠른 움직임을 시뮬레이트하는 데 유용합니다. 흐림 필터의 quality 속성을 낮게 설정하면 부드럽게 흐린 렌즈 효과를 시뮬레이션할 수 있습니다. quality 속성을 높음으로 설정하면 가우시안 흐림과 유사한 매끄러운 흐림 효과가 나타납니다.

다음 예제에서는 Graphics 클래스의 drawCircle() 메서드를 사용하여 원형 객체를 만들고 여기에 흐림 필터를 적용합니다.

```
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];
```

그림자 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

그림자는 대상 객체 위에 별도의 광원이 있다는 느낌을 줍니다. 이 광원의 위치 및 강도를 수정하여 여러 다양한 그림자 효과를 연출할 수 있습니다.

DropShadowFilter 클래스는 흐림 필터의 알고리즘과 유사한 알고리즘을 사용합니다. 그러나 가장 큰 차이점은 그림자 필터에서는 보다 많은 속성을 수정하여 알파, 색상, 오프셋 및 밝기 등의 다양한 광원 특성을 시뮬레이트할 수 있다는 점입니다.

그림자 필터를 사용하면 또한 그림자 스타일에 내부 또는 외부 그림자 및 녹아웃(또는 컷아웃) 모드 등의 사용자 정의 변형 옵션을 적용할 수 있습니다.

다음은 사각형 상자 Sprite를 만든 후 그림자 필터를 적용하는 코드입니다.

```
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];
```

광선 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

GlowFilter 클래스는 표시 객체에 광선 효과를 적용하여 객체 아래쪽에서 빛이 비추어 객체가 부드럽게 빛나는 것처럼 보이도록 할 수 있습니다.

그림자 필터와 유사한 광선 필터에는 거리, 각도 및 광원의 색상을 수정할 수 있는 속성이 있어 다양한 효과 연출이 가능합니다. GlowFilter에는 또한 내부 또는 외부 광선 및 녹아웃 모드를 비롯한 광선 스타일을 수정할 수 있는 몇 가지 옵션이 있습니다.

다음은 Sprite 클래스를 사용하여 십자가를 만든 후 광선 필터를 적용하는 코드입니다.

```
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];
```

그래디언트 경사 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

GradientBevelFilter 클래스를 사용하면 표시 객체 또는 BitmapData 객체에 향상된 경사 효과를 적용할 수 있습니다. 경사에 그래디언트 색상을 사용하면 경사의 공간 심도가 현저히 개선되기 때문에 가장자리에 보다 사실적인 3D 효과를 줄 수 있습니다.

다음 코드는 Shape 클래스의 drawRect() 메서드를 사용하여 사각형 객체를 만들고 여기에 그래디언트 경사 필터를 적용합니다.

```
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);
```

그래디언트 광선 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

GradientGlowFilter 클래스를 사용하면 표시 객체 또는 BitmapData 객체에 향상된 광선 효과를 적용할 수 있습니다. 즉, 광선의 색상을 효율적으로 제어하여 한층 더 현실적인 광선 효과를 연출하도록 합니다. 또한 그래디언트 광선 필터를 사용하면 객체의 내부, 외부 또는 상단 가장자리에 그래디언트 광선을 적용할 수도 있습니다.

다음 예제에서는 스테이지에 원을 그린 다음 그래디언트 광선 필터를 적용합니다. 마우스를 점점 오른쪽 아래로 옮길수록 가로 및 세로 방향 모두에서 흐림의 양이 증가합니다. 또한 스테이지를 클릭할 때마다 흐림 강도가 높아집니다.

```
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);
```

예제: 기본 필터 결합

Flash Player 9 이상, Adobe AIR 1.0 이상

다음은 반복 액션을 만드는 Timer와 여러 기본 필터를 결합하여 신호등 애니메이션을 시뮬레이트하는 코드 예제입니다.

```
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();
```

색상 행렬 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

ColorMatrixFilter 클래스는 필터링된 객체의 색상 및 알파 값을 조작하는 데 사용됩니다. ColorMatrixFilter를 통해 채도 변경, 색조 회전(한 색상 범위에서 다른 색상 범위로 팔레트를 이동하는 것), 광도 변경 작업 등을 수행할 수 있으며 한 색상 채널의 값을 사용하고 이 값을 다른 채널에 적용하여 다른 색상 조작 효과를 연출할 수 있습니다.

개념적으로 이 필터는 소스 이미지의 픽셀을 하나씩 거쳐가면서 각 픽셀을 빨강, 녹색, 파랑 및 알파 요소로 분리합니다. 그런 다음 분리된 각 픽셀 값을 색상 행렬에 제공된 값으로 곱한 후 결과를 더해 화면에 표시될 해당 픽셀의 색상 값을 결정합니다. 필터의 matrix 속성은 최종 색상을 계산하는 데 사용되는 20개 숫자의 배열입니다. 색상 값 계산에 사용되는 특정 알고리즘에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 ColorMatrixFilter 클래스의 matrix 속성에 대한 항목을 참조하십시오.

회선 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

ConvolutionFilter 클래스를 사용하면 BitmapData 객체 또는 표시 객체에 흐리게 하기, 가장자리 감지, 선명하게 하기, 엠보싱, 경사 등의 다양한 이미지 변형을 적용할 수 있습니다.

회선 필터는 개념적으로 소스 이미지의 각 픽셀을 하나씩 거쳐가면서 각 픽셀 및 그 주변 픽셀의 값을 사용하여 해당 픽셀의 최종 색상을 결정합니다. 숫자 값 배열로 지정되는 행렬은 인접한 각 특정 픽셀 값이 최종 결과 값에 영향을 미치는 정도를 나타냅니다.

가장 일반적으로 사용되는 3x3 행렬을 예로 들어 보겠습니다. 아래 행렬에는 9개의 값이 있습니다.

```
N   N   N
N   P   N
N   N   N
```

특정 픽셀에 회선 필터를 적용하는 경우에는 픽셀 자체의 색상 값(이 예제에서는 "P")과 함께 주변 픽셀 값(이 예제에서 "N"으로 레이블 표시)을 검토합니다. 그러나 행렬의 값을 설정하여 각 픽셀이 결과 이미지에 영향을 미치는 우선 순위를 지정하십시오.

예를 들어 회선 필터를 사용하여 적용된 다음 행렬은 이미지를 원래 모습 그대로 유지합니다.

```
0   0   0
0   1   0
0   0   0
```

이미지가 변경되지 않는 것은 원래 픽셀 값에 최종 픽셀 색상을 결정하는 상대 강도가 1인 반면, 주변 픽셀 값의 상대 강도는 0으로 해당 색상이 최종 이미지에 반영되지 않기 때문입니다.

마찬가지로 다음 행렬은 이미지의 픽셀을 왼쪽으로 한 픽셀씩 이동하게 합니다.

```
0   0   0
0   0   1
0   0   0
```

이 경우 픽셀 자체는 최종 이미지의 해당 위치에 표시된 최종 픽셀 값에 영향을 미치지 않고, 오른쪽 픽셀 값만 해당 픽셀의 최종 값을 결정하는 데 사용됩니다.

ActionScript에서는 행렬의 행과 열 수를 지정하는 두 개의 속성과 값이 포함된 Array 인스턴스의 조합으로 행렬을 만들 수 있습니다. 다음 예제에서는 이미지를 로드하고 이미지 로딩이 끝나면 앞에서 설명한 행렬을 사용하여 해당 이미지에 회선 필터를 적용합니다.

```
// 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);
```

이 코드에서는 행렬에 1 또는 0 이외의 다른 값을 사용했기 때문에 명확하지 않은 부분이 있습니다. 예를 들어 오른쪽에 1이 아닌 8이 있는 동일한 행렬은 동일한 액션을 수행합니다(왼쪽으로 픽셀 이동). 그러나 이미지의 색상에 영향을 미쳐 해당 이미지를 8배 밝게 만듭니다. 이는 최종 픽셀 색상이 해당 행렬 값에 원본 픽셀 값을 곱하고 이 두 값을 더한 다음 필터의 divisor 속성 값으로 나누어서 계산되기 때문입니다. 예제 코드에서 divisor 속성은 1로 설정됩니다. 일반적으로 원본 이미지와 동일한 색상 밝기를 유지하려는 경우 제수를 행렬 값의 합과 동일하게 해야 합니다. 따라서 행렬 값 합계가 8이고 제수가 1인 행렬의 결과 이미지는 원래 이미지보다 8배 가량 밝게 됩니다.

이 행렬의 효과는 그다지 두드러지지 않지만 다른 행렬 값을 사용하여 다양한 효과를 연출할 수 있습니다. 다음은 3x3 행렬을 통해 연출할 수 있는 여러 다양한 효과의 표준이 되는 몇 가지 행렬 값 집합입니다.

- 기본 흐림(제수 5):

  ```
  0 1 0
  1 1 1
  0 1 0
  ```

- 선명 효과(제수 1):

  ```
   0, -1, 0
  -1, 5, -1
   0, -1, 0
  ```

- 가장자리 감지(제수 1):

  ```
   0, -1, 0
  -1, 4, -1
   0, -1, 0
  ```

- 엠보싱 효과(제수 1):

  ```
  -2, -1, 0
  -1, 1, 1
   0, 1, 2
  ```

  대부분의 이러한 효과에서 제수는 1입니다. 이것은 양의 행렬 값에 음의 행렬 값을 추가하면 1(가장자리 감지의 경우 0이지만 divisor 속성 값은 0이 될 수 없음)이 되기 때문입니다.

위치 변경 맵 필터

Flash Player 9 이상, Adobe AIR 1.0 이상

DisplacementMapFilter 클래스는 BitmapData 객체(위치 변경 맵 이미지)의 픽셀 값을 사용하여 새 객체에서 위치 변경 효과를 수행합니다. 위치 변경 맵 이미지는 필터가 적용되는 실제 표시 객체나 BitmapData 인스턴스와는 일반적으로 다릅니다. 위치 변경 효과는 필터링된 이미지에 있는 픽셀의 위치를 변경하는 것 즉, 이미지의 픽셀을 원래 위치에서 다른 곳으로 이동하는 것입니다. 이 필터는 이동, 비틀기 또는 얼룩 효과를 연출하는 데 사용할 수 있습니다.

지정된 픽셀에 적용되는 위치 변경의 위치와 정도는 위치 변경 맵 이미지의 색상 값에 의해 결정됩니다. 필터를 사용하여 작업할 때는 맵 이미지를 지정하는 것 이외에도 다음 값을 지정하여 맵 이미지에서 위치 변경이 계산되는 방법을 제어합니다.

- 맵 포인트: 필터링된 이미지의 위치를 나타내며, 위치 변경 필터의 왼쪽 위 모서리가 적용됩니다. 이미지의 일부에만 필터를 적용하려면 이 값을 사용합니다.
- X 구성 요소: 맵 이미지의 색상 채널이 픽셀의 x 위치에 영향을 미칩니다.
- Y 구성 요소: 맵 이미지의 색상 채널이 픽셀의 y 위치에 영향을 미칩니다.
- X 배율: x축 위치 변경의 정도를 지정하는 승수 값입니다.
- Y 배율: y축 위치 변경의 정도를 지정하는 승수 값입니다.
- 필터 모드: 픽셀 이동으로 인해 생긴 빈 공간이 처리되는 방식을 결정합니다. DisplacementMapFilterMode 클래스에서 상수로 정의된 옵션에는 원본 픽셀 표시(필터 모드 IGNORE), 이미지의 다른 쪽에서 해당 픽셀 감싸기(필터 모드 WRAP - 기본값), 가장 가깝게 이동된 픽셀 사용(필터 모드 CLAMP) 또는 색상으로 공간 채우기(필터 모드 COLOR)가 있습니다.

기본적인 예제를 통해 위치 변경 맵 필터의 작동 방식을 알아보겠습니다. 다음 코드는 이미지를 로드한 후 해당 이미지를 스테이지의 중앙에 배치하고 위치 변경 맵 필터를 적용하여 전체 이미지의 모든 픽셀을 왼쪽 가로 방향으로 이동시킵니다.

```
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);
```

위치 변경 정의에 사용된 속성은 다음과 같습니다.

- 맵 비트맵: 위치 변경 비트맵은 코드에 의해 만들어진 새로운 BitmapData 인스턴스로서, 이 비트맵의 크기는 로드된 이미지의 크기와 일치하므로 위치 변경이 전체 이미지에 적용됩니다. 이 비트맵은 단색 빨강 픽셀로 채워집니다.
- 맵 포인트: 위치 변경이 전체 이미지에 적용되도록 값이 0, 0 점으로 설정됩니다.
- X 구성 요소: 이 값은 상수 BitmapDataChannel.RED로 설정되어 있으며 이는 맵 비트맵의 빨강 값이 x축을 따라 위치 변경(이동)된 픽셀의 양을 결정한다는 것을 의미합니다.
- X 배율: 이 값은 250으로 설정됩니다. 위치 변경(완전히 빨강인 맵 이미지에서 시작)의 전체 양은 이미지의 위치를 조금밖에(대략 1/2 픽셀) 변경하지 못하므로 이 값을 1로 설정할 경우 이미지가 가로로 0.5 픽셀만큼만 이동합니다. X 배율을 250으로 설정하면 이미지가 약 125 픽셀만큼 이동합니다.

이러한 설정은 필터링된 이미지의 픽셀이 왼쪽으로 250 픽셀만큼 이동하도록 합니다. 이동 방향(왼쪽 또는 오른쪽) 및 이동 정도는 맵 이미지에 있는 픽셀의 색상 값을 기반으로 합니다. 개념적으로 필터는 필터링된 이미지의 픽셀을 하나씩 거쳐가면서(최소한 필터가 적용된 영역에 있는 픽셀, 이 예제의 경우에는 모든 픽셀) 각 픽셀에 대해 다음 작업을 수행합니다.

1. 맵 이미지에서 대응하는 픽셀을 찾습니다. 예를 들어 필터에서는 필터링된 이미지의 왼쪽 위 모서리에 있는 픽셀의 위치 변경 정도를 계산할 때 맵 이미지의 왼쪽 위 모서리에 있는 픽셀을 확인합니다.
2. 맵 픽셀에서 특정 색상 채널의 값을 확인합니다. 이 예제에서 x 구성 요소 색상 채널이 빨강 채널이므로 필터에서는 맵 이미지의 해당 픽셀에서 빨강 채널의 값을 확인합니다. 맵 이미지가 단색 빨강이므로 픽셀의 빨강 채널 값은 0xFF 또는 255이며, 이 값이 위치 변경 값으로 사용됩니다.

3 위치 변경 값을 "중간" 값(0과 255의 중간 값인 127)과 비교합니다. 위치 변경 값이 중간 값보다 낮으면 픽셀이 양의 방향(x축 위치 변경의 경우 오른쪽, y축 위치 변경의 경우 아래쪽)으로 이동합니다. 반대로 이 예제에서처럼 위치 변경 값이 중간 값보다 높으면 픽셀이 음의 방향(x축 위치 변경의 경우 왼쪽, y축 위치 변경의 경우 위쪽)으로 이동합니다. 보다 정확히 설명하자면 필터는 127에서 위치 변경 값을 감산하며, 그 결과(양의 결과 또는 음의 결과)가 적용할 위치 변경의 상대적 양이 됩니다.

4 마지막으로 상대적인 위치 변경 값이 전체 위치 변경에서 차지하는 백분율을 파악하여 실제 위치 변경 양을 결정합니다. 이 예제에서 전체 빨강은 100%의 위치 변경을 의미합니다. 이 백분율을 이후 x 배율 또는 y 배율 값으로 곱하면 적용할 위치 변경의 픽셀 수가 산출됩니다. 이 예제에서 100%에 승수 250을 곱하면 위치 변경의 양이 구해집니다(대략 왼쪽으로 125 픽셀).

y 구성 요소 및 y 배율에 지정된 값이 없기 때문에 기본값(위치 변경이 수행되지 않음)이 사용되었으며 결과 이미지가 세로 방향으로 이동하지 않습니다.

기본 필터 모드 설정인 WRAP이 이 예제에 사용되어 픽셀을 왼쪽으로 이동하면 오른쪽의 빈 공간이 이미지의 왼쪽 가장자리로 이동된 픽셀로 채워집니다. 이 값을 활용하여 다른 효과를 확인해 볼 수 있습니다. 예를 들어 위치 변경 속성이 설정된 코드 부분(loader.filters = [displacementMap] 행 앞)에 다음 행을 추가하면 이미지가 스테이지 전체에 퍼진 것처럼 보입니다.

```
displacementMap.mode = DisplacementMapFilterMode.CLAMP;
```

보다 복잡한 예제의 경우 다음 샘플에서 위치 변경 맵 필터를 사용하여 이미지에 돋보기 효과를 만듭니다.

```
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);
```

먼저 코드는 두 개의 그래디언트 원을 생성하고 해당 원은 서로 결합되어 위치 변경 맵 이미지를 형성합니다. 빨강 원은 x축 위치 변경(xyFilter.componentX = BitmapDataChannel.RED)을 만들고, 파랑 원은 y축 위치 변경(xyFilter.componentY = BitmapDataChannel.BLUE)을 만듭니다. 위치 변경 맵 이미지 모양에 대한 이해를 돕기 위해 코드에서 원본 원과 함께 해당 맵 이미지인 결합 원을 화면의 맨 아래에 추가합니다.

그런 다음 이미지를 로드하고 마우스 움직임에 따라 마우스 아래에 있는 이미지의 부분에 위치 변경 필터를 적용합니다. 위치 변경 맵 이미지로 사용된 그래디언트 원은 위치 변경된 영역이 포인터에서부터 퍼지도록 합니다. 위치 변경 맵 이미지의 회색 영역에서는 위치 변경이 발생하지 않습니다. 회색 색상은 0x7F7F7F입니다. 회색 음영의 파랑 채널 및 빨강 채널은 해당 색상 채널의 중간 음영과 정확히 일치하므로 맵 이미지의 회색 영역에서는 위치 변경이 발생하지 않습니다. 마찬가지로 원 중심에서도 위치 변경이 발생하지 않습니다. 색상에 회색이 없다고 하더라도 파랑 채널과 빨강 채널이 중간 회색의 파랑 채널 및 빨강 채널과 똑같고, 위치 변경을 발생시키는 색상이 파랑 및 빨강이기 때문에 원 중심에서 위치 변경이 발생하지 않습니다.

셰이더 필터

Flash Player 10 이상, Adobe AIR 1.5 이상

ShaderFilter 클래스를 사용하면 Pixel Bender 셰이더로 정의된 사용자 정의 필터 효과를 사용할 수 있습니다. 필터 효과가 Pixel Bender 셰이더로 작성되었기 때문에 효과를 완전히 사용자 정의할 수 있습니다. 필터링된 내용은 이미지 입력으로 셰이더에 전달되고 셰이더 작업의 결과는 필터 결과가 됩니다.

참고: Flash Player 10 및 Adobe AIR 1.5부터 셰이더 필터를 사용할 수 있습니다.

객체에 셰이더 필터를 적용하려면 먼저 사용 중인 Pixel Bender 셰이더를 나타내는 Shader 인스턴스를 만듭니다. Shader 인스턴스를 만드는 절차 및 입력 이미지와 매개 변수 값을 지정하는 방법에 대한 자세한 내용은 271페이지의 “Pixel Bender 셰이더를 사용한 작업”을 참조하십시오.

셰이더를 필터로 사용하는 경우 다음 세 가지 중요한 사항을 고려하십시오.

- 입력 이미지를 한 개 이상 허용하도록 셰이더를 정의해야 합니다.
- 필터링된 객체(필터가 적용된 표시 객체 또는 BitmapData 객체)는 첫 번째 입력 이미지 값으로 셰이더에 전달됩니다. 이 때문에 첫 번째 이미지 입력의 값을 수동으로 지정하면 안 됩니다.
- 셰이더가 입력 이미지를 두 개 이상 정의하는 경우 Shader 인스턴스에 속하는 ShaderInput 인스턴스의 input 속성을 설정하여 수동으로 추가 입력을 지정해야 합니다.

셰이더에 대한 Shader 객체가 있으면 ShaderFilter 인스턴스를 만듭니다. 이 인스턴스는 다른 필터와 동일한 방식으로 사용되는 실제 필터 객체입니다. Shader 객체를 사용하는 ShaderFilter를 만들려면 다음 샘플에 표시된 대로 ShaderFilter() 생성자를 호출하고 Shader 객체를 인수로 전달합니다.

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
```

셰이더 필터의 사용법을 보여 주는 전체 예제는 287페이지의 "셰이더를 필터로 사용"을 참조하십시오.

표시 객체 필터링 예제: Filter Workbench

Flash Player 9 이상, Adobe AIR 1.0 이상

Filter Workbench에서 제공하는 사용자 인터페이스를 통해 이미지 및 기타 시각적 내용에 다양한 필터를 적용해 보고 ActionScript에서 동일한 효과를 내는 결과 코드를 확인할 수 있습니다. 이 응용 프로그램은 필터를 시험해 볼 수 있는 도구를 제공할 뿐만 아니라 다음과 같은 작업을 수행하는 방법도 보여 줍니다.

- 다양한 필터 인스턴스 만들기
- 표시 객체에 여러 개의 필터 적용

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr를 참조하십시오. Filter Workbench 응용 프로그램 파일은 Samples/FilterWorkbench 폴더에 있으며 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
com/example/programmingas3/filterWorkbench/FilterWorkbenchController.as	필터가 적용되는 내용을 전환하고 내용에 필터를 적용하는 등 응용 프로그램의 주요 기능을 제공하는 클래스입니다.
com/example/programmingas3/filterWorkbench/IFilterFactory.as	각 필터 팩토리 클래스를 통해 구현되는 공통 메서드를 정의하는 인터페이스입니다. 이 인터페이스는 FilterWorkbenchController 클래스가 개별 필터 팩토리 클래스와 상호 작용하기 위해 사용하는 공통 기능을 정의합니다.
com/example/programmingas3/filterWorkbench/ 폴더에 있는 파일: BevelFactory.as BlurFactory.as ColorMatrixFactory.as ConvolutionFactory.as DropShadowFactory.as GlowFactory.as GradientBevelFactory.as GradientGlowFactory.as	IFilterFactory 인터페이스를 구현하는 각 클래스의 집합입니다. 각 클래스는 단일 필터 유형의 값을 만들고 설정하는 기능을 제공합니다. 응용 프로그램의 필터 속성 패널이 이러한 팩토리 클래스를 사용하여 특정 필터의 인스턴스를 만들면 FilterWorkbenchController 클래스는 이러한 인스턴스를 검색하여 이미지 내용에 적용합니다.
com/example/programmingas3/filterWorkbench/IFilterPanel.as	응용 프로그램의 필터 값을 조작하는 데 사용되는 사용자 인터페이스 패널을 정의하는 클래스로 구현되는 공통 메서드를 정의하는 인터페이스입니다.
com/example/programmingas3/filterWorkbench/ColorStringFormatter.as	숫자 색상 값을 16진수 문자열 형식으로 변환하는 메서드를 포함하는 유틸리티 클래스입니다.

파일	설명
com/example/programmingas3/filterWorkbench/GradientColor.as	GradientBevelFilter 및 GradientGlowFilter의 각 색상과 연결된 세 가지 값(색상, 알파 및 비율)을 단일 객체로 결합하는 값 객체 역할을 하는 클래스입니다.
사용자 인터페이스(Flex)	
FilterWorkbench.mxml	응용 프로그램의 사용자 인터페이스를 정의하는 기본 파일입니다.
flexapp/FilterWorkbench.as	기본 응용 프로그램의 사용자 인터페이스에 대한 기능을 제공하는 클래스입니다. 이 클래스는 응용 프로그램 MXML 파일의 코드 숨김 클래스로 사용됩니다.
flexapp/filterPanels 폴더에 있는 파일: BevelPanel.mxml BlurPanel.mxml ColorMatrixPanel.mxml ConvolutionPanel.mxml DropShadowPanel.mxml GlowPanel.mxml GradientBevelPanel.mxml GradientGlowPanel.mxml	단일 필터에 대한 옵션을 설정하는 데 사용되는 각 패널의 기능을 제공하는 MXML 구성 요소 집합입니다.
flexapp/ImageContainer.as	화면에 로드된 이미지의 컨테이너 역할을 하는 표시 객체입니다.
flexapp/controls/BGColorCellRenderer.as	DataGrid 구성 요소에서 셀의 배경색을 변경하는 데 사용되는 사용자 정의 셀 렌더러입니다.
flexapp/controls/QualityComboBox.as	여러 필터 패널의 품질 설정에 사용할 수 있는 콤보 상자를 정의하는 사용자 정의 컨트롤입니다.
flexapp/controls/TypeComboBox.as	여러 필터 패널의 유형 설정에 사용할 수 있는 콤보 상자를 정의하는 사용자 정의 컨트롤입니다.
사용자 인터페이스(Flash)	
FilterWorkbench.fla	응용 프로그램의 사용자 인터페이스를 정의하는 기본 파일입니다.
flashapp/FilterWorkbench.as	기본 응용 프로그램의 사용자 인터페이스에 대한 기능을 제공하는 클래스입니다. 이 클래스는 응용 프로그램 FLA 파일의 문서 클래스로 사용됩니다.

파일	설명
flashapp/filterPanels 폴더에 있는 파일: BevelPanel.as BlurPanel.as ColorMatrixPanel.as ConvolutionPanel.as DropShadowPanel.as GlowPanel.as GradientBevelPanel.as GradientGlowPanel.as	단일 필터에 대한 옵션을 설정하는 데 사용되는 각 패널의 기능을 제공하는 클래스 집합입니다. 기본 응용 프로그램 FLA 파일의 라이브러리에는 각 클래스의 이름과 일치하는 연결된 MovieClip 심볼이 있습니다. 예를 들어 "BlurPanel" 심볼은 BlurPanel.as에 정의된 클래스와 링크되어 있습니다. 사용자 인터페이스를 구성하는 구성 요소는 이러한 심볼 내부에 위치하고 이름이 지정됩니다.
flashapp/ImageContainer.as	화면에 로드된 이미지의 컨테이너 역할을 하는 표시 객체입니다.
flashapp/BGColorCellRenderer.as	DataGrid 구성 요소에서 셀의 배경색을 변경하는 데 사용되는 사용자 정의 셀 렌더러입니다.
flashapp/ButtonCellRenderer.as	DataGrid 구성 요소의 셀에 Button 구성 요소를 포함하기 위해 사용되는 사용자 정의 셀 렌더러입니다.
필터링된 이미지 내용	
com/example/programmingas3/filterWorkbench/ImageType.as	응용 프로그램에서 로드하여 필터를 적용할 수 있는 단일 이미지 파일의 유형 및 URL을 포함하는 값 객체 역할을 하는 클래스입니다. 이 클래스에는 사용할 수 있는 실제 이미지 파일을 나타내는 상수 집합도 포함되어 있습니다.
images/sampleAnimation.swf, images/sampleImage1.jpg, images/sampleImage2.jpg	응용 프로그램에서 필터가 적용되는 이미지 및 기타 시각적 내용입니다.

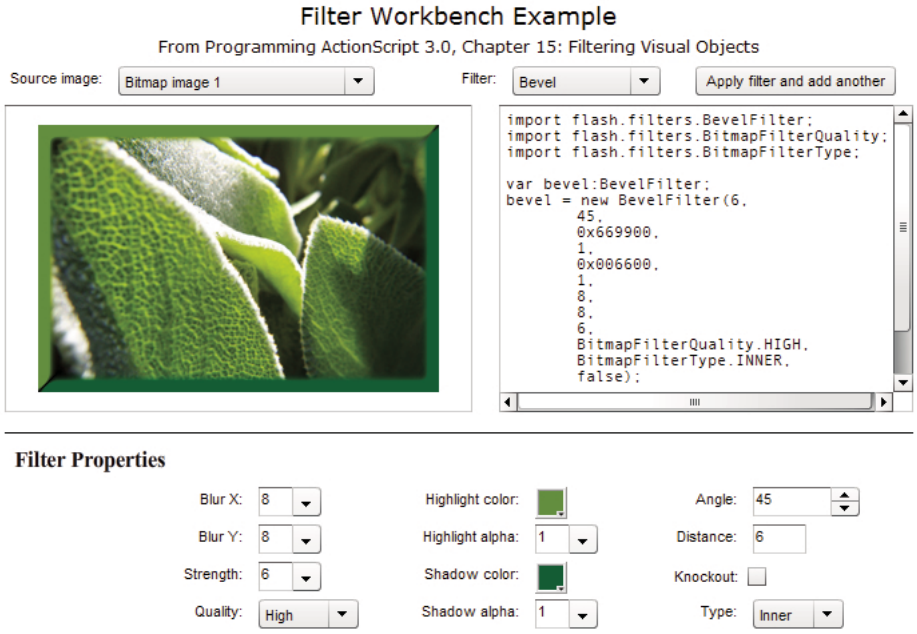
ActionScript 필터 시험하기

Flash Player 9 이상, Adobe AIR 1.0 이상

Filter Workbench 응용 프로그램은 사용자가 다양한 필터 효과를 시험해 보고 이런 효과에 대해 관련 ActionScript 코드를 생성할 수 있도록 설계되었습니다. 이 응용 프로그램을 사용하면 비트맵 이미지 및 Flash에서 만들어진 애니메이션 등의 시각적 내용을 포함하는 세 가지 다른 파일을 선택할 수 있으며 선택한 이미지에 8개의 다른 ActionScript 필터를 개별적으로 적용하거나 다른 필터와 조합하여 적용할 수 있습니다. 응용 프로그램에는 다음 필터가 포함되어 있습니다.

- 경사(flash.filters.BevelFilter)
- 흐림(flash.filters.BlurFilter)
- 색상 행렬(flash.filters.ColorMatrixFilter)
- 회선(flash.filters.ConvolutionFilter)
- 그림자(flash.filters.DropShadowFilter)
- 광선(flash.filters.GlowFilter)
- 그래디언트 경사(flash.filters.GradientBevelFilter)
- 그래디언트 광선(flash.filters.GradientGlowFilter)

사용자가 이미지와 이 이미지에 적용할 필터를 선택하면 응용 프로그램은 선택한 필터의 특정 속성을 설정하는 컨트롤이 있는 패널을 표시합니다. 예를 들어, 다음 이미지는 [경사] 필터가 선택된 응용 프로그램을 보여 줍니다.



사용자가 필터 속성을 조정하면 미리 보기가 실시간으로 업데이트됩니다. 사용자는 한 필터를 사용자 정의하고 [적용] 버튼을 클릭한 다음 다른 필터를 사용자 정의하고 [적용] 버튼을 클릭하는 방식으로 여러 개의 필터를 적용할 수도 있습니다.

응용 프로그램의 필터 패널의 몇 가지 기능과 제한 사항은 다음과 같습니다.

- 색상 행렬 필터에는 밝기, 대비, 채도 및 색조를 포함하는 공통 이미지 속성을 직접 조작할 수 있는 컨트롤 집합이 포함되어 있습니다. 또한 사용자 정의 행렬 값을 지정할 수 있습니다.
- ActionScript를 통해서만 사용할 수 있는 회선 필터에는 주로 사용되는 회선 행렬 값 집합이 포함되어 있지만 사용자 정의 값을 지정할 수도 있습니다. ConvolutionFilter 클래스에는 모든 크기의 행렬을 사용할 수 있지만 Filter Workbench 응용 프로그램은 가장 일반적으로 사용되는 고정된 3 x 3 행렬을 사용합니다.
- ActionScript에서만 사용할 수 있는 위치 변경 맵 필터와 셰이더 필터는 Filter Workbench 응용 프로그램에서 사용할 수 없습니다.

필터 인스턴스 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

Filter Workbench 응용 프로그램에는 사용할 수 있는 필터별로 하나씩 지정된 클래스 집합이 포함되어 있는데 개별 패널은 이러한 클래스를 사용하여 필터를 만듭니다. 사용자가 필터를 선택하면 필터 패널과 연결된 ActionScript 코드에서는 해당하는 필터 팩토리 클래스의 인스턴스를 만듭니다. (이러한 클래스는 실제 팩토리에서 개별 제품을 만드는 것처럼 다른 객체의 인스턴스를 만들므로 팩토리 클래스라고 불립니다.)

사용자가 패널에서 속성 값을 변경할 때마다 패널의 코드는 팩토리 클래스의 해당하는 메서드를 호출합니다. 각 팩토리 클래스에는 패널이 적절한 필터 인스턴스를 만들 때 사용하는 특정 메서드가 포함되어 있습니다. 예를 들어 사용자가 [흐림] 필터를 선택하는 경우 응용 프로그램은 BlurFactory 인스턴스를 만듭니다. BlurFactory 클래스에는 blurX, blurY 및 quality라는 세 개의 매개 변수를 사용하는 modifyFilter() 메서드가 있는데 이 매개 변수는 원하는 BlurFilter 인스턴스를 만드는 데 함께 사용됩니다.

```
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));
}
```

반면 사용자가 유연성이 훨씬 더 높은 [회선] 필터를 선택하는 경우 제어할 수 있는 속성 집합은 더 커지게 됩니다. 사용자가 필터 패널에서 다른 값을 선택하면 ConvolutionFactory 클래스에서 다음 코드가 호출됩니다.

```
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));
}
```

각각의 경우에 필터 값이 변경되면 팩토리 객체는 Event.CHANGE 이벤트를 전달하여 리스너에게 필터 값이 변경되었음을 알립니다. 필터링된 내용에 필터를 실제로 적용하는 작업을 수행하는 FilterWorkbenchController 클래스는 필터의 새 복사본을 검색하여 필터링된 내용에 다시 적용해야 할 시기를 확인하기 위해 이벤트를 수신 대기합니다.

FilterWorkbenchController 클래스는 각 필터 팩토리 클래스의 자세한 내용을 알 필요가 없으며 단지 필터가 변경되었음을 알고 필터의 복사본에 액세스할 수만 있으면 됩니다. 이를 위해 응용 프로그램에는 응용 프로그램의 FilterWorkbenchController 인스턴스가 작업을 수행할 수 있도록 필터 팩토리 클래스에서 제공해야 하는 비헤이비어를 정의하는 IFilterFactory 인터페이스가 포함되어 있습니다. IFilterFactory는 FilterWorkbenchController 클래스에서 사용되는 getFilter() 메서드를 정의합니다.

```
function getFilter():BitmapFilter;
```

getFilter() 인터페이스 메서드 정의에는 특정 유형의 필터가 아니라 BitmapFilter 인스턴스가 반환되도록 지정되어 있습니다. BitmapFilter 클래스는 특정 유형의 필터를 정의하지 않습니다. BitmapFilter는 모든 필터 클래스의 기본 클래스입니다. 각 필터 팩토리 클래스는 자신이 작성한 필터 객체에 대한 참조를 반환하는 getFilter() 메서드의 특정 구현을 정의합니다. 예를 들어 다음은 ConvolutionFactory 클래스 소스 코드의 단축된 버전입니다.

```
public class ConvolutionFactory extends EventDispatcher implements IFilterFactory
{
    // ------- Private vars -------
    private var _filter:ConvolutionFilter;
    ...
    // ------- IFilterFactory implementation -------
    public function getFilter():BitmapFilter
    {
        return _filter;
    }
    ...
}
```

ConvolutionFactory 클래스가 구현하는 getFilter() 메서드는 ConvolutionFilter 인스턴스를 반환하는데, getFilter()를 호출하는 객체는 이런 사실을 알 필요가 없습니다. ConvolutionFactory가 따르는 getFilter() 메서드의 정의에 따르면 이 메서드는 모든 ActionScript 필터 클래스의 인스턴스가 될 수 있는 BitmapFilter 인스턴스를 반환해야 합니다.

표시 객체에 필터 적용하기

Flash Player 9 이상, Adobe AIR 1.0 이상

앞에서 설명한 대로 Filter Workbench 응용 프로그램은 선택한 시각적 객체에 필터를 실제로 적용하는 작업을 수행하는 FilterWorkbenchController 클래스의 인스턴스(이후 "컨트롤러 인스턴스"라고 함)를 사용합니다. 컨트롤러 인스턴스는 필터를 적용하기 전에 먼저 필터를 적용할 이미지 또는 시각적 내용에 대해 알고 있어야 합니다. 사용자가 이미지를 선택하면 응용 프로그램은 ImageType 클래스에 정의된 상수 중 하나를 전달하여 FilterWorkbenchController 클래스의 setFilterTarget() 메서드를 호출합니다.

```
public function setFilterTarget(targetType:ImageType):void
{
    ...
    _loader = new Loader();
    ...
    _loader.contentLoaderInfo.addEventListener(Event.COMPLETE, targetLoadComplete);
    ...
}
```

해당 정보를 사용하여 컨트롤러 인스턴스는 지정된 파일을 로드하고 이를 _currentTarget이라는 인스턴스 변수에 저장합니다.

```
private var _currentTarget:DisplayObject;

private function targetLoadComplete(event:Event):void
{
    ...
    _currentTarget = _loader.content;
    ...
}
```

사용자가 필터를 선택하면 응용 프로그램은 관련 필터 팩토리 객체에 대한 참조를 지정하여 컨트롤러 인스턴스의 setFilter() 메서드를 호출하고, 이 컨트롤러 인스턴스는 해당 참조를 _filterFactory라는 인스턴스 변수에 저장합니다.

```
private var _filterFactory:IFilterFactory;

public function setFilter(factory:IFilterFactory):void
{
    ...

    _filterFactory = factory;
    _filterFactory.addEventListener(Event.CHANGE, filterChange);
}
```

앞에서 설명한 것처럼 컨트롤러 인스턴스는 지정된 필터 팩토리 인스턴스의 특정 데이터 유형에 대해 알지 못하며 단지 이 객체가 IFilterFactory 인스턴스를 구현하므로 getFilter() 메서드를 가지고 있고, 이 메서드는 필터가 변경되면 change(Event.CHANGE) 이벤트를 전달한다는 것만 알고 있습니다.

사용자가 필터 패널에서 필터 속성을 변경하면 컨트롤러 인스턴스는 컨트롤러 인스턴스의 filterChange() 메서드를 호출하는 필터 팩토리의 change 이벤트를 통해 필터가 변경되었음을 확인합니다. 이 메서드는 다시 applyTemporaryFilter() 메서드를 호출합니다.

```
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();
}
```

표시 객체에 필터를 적용하는 작업은 applyTemporaryFilter() 메서드에서 발생합니다. 먼저 컨트롤러는 필터 팩토리의 getFilter() 메서드를 호출하여 필터 객체에 대한 참조를 검색합니다.

```
var currentFilter:BitmapFilter = _filterFactory.getFilter();
```

컨트롤러 인스턴스에는 _currentFilters라고 하는 Array 인스턴스 변수가 있는데 이 변수에는 표시 객체에 적용된 모든 필터가 저장되어 있습니다. 다음 단계는 새로 업데이트된 필터를 해당 배열에 추가하는 것입니다.

```
_currentFilters.push(currentFilter);
```

그런 다음, 이미지에 필터를 실제로 적용하는 표시 객체의 filters 속성에 필터의 배열을 지정합니다.

```
_currentTarget.filters = _currentFilters;
```

마지막으로, 가장 최근에 추가된 이 필터는 표시 객체에 영구적으로 적용되지 않아야 하는 "작업 중인" 필터이므로 이 필터는 _currentFilters 배열에서 제거됩니다.

```
_currentFilters.pop();
```

표시 객체는 filters 속성에 지정될 때 필터 배열의 복사본을 만들어서 원래 배열 대신 이 내부 배열을 사용합니다. 따라서 배열에서 이 필터를 제거해도 필터링된 표시 객체는 영향을 받지 않습니다. 따라서 필터의 배열을 변경해도 배열이 표시 객체의 filters 속성에 다시 지정될 때까지 표시 객체는 영향을 받지 않습니다.

15장: Pixel Bender 셰이더를 사용한 작업

Flash Player 10 이상, Adobe AIR 1.5 이상

Adobe Pixel Bender Toolkit을 통해 개발자는 그래픽 효과를 만들고 다른 이미지 및 데이터 처리를 수행하는 셰이더를 작성할 수 있습니다. ActionScript에서 Pixel Bender 바이트코드를 실행하여 이미지 데이터나 시각적 내용에 효과를 적용할 수 있습니다. ActionScript에서 Pixel Bender 셰이더를 사용하면 사용자 정의 시각적 효과를 만들 수 있으며 ActionScript의 내장 기능보다 우수한 데이터 처리를 수행할 수 있습니다.

참고: Pixel Bender는 Flash Player 10 및 Adobe AIR 1.5부터 지원됩니다. Pixel Bender 블렌드, 필터 및 채우기는 GPU 렌더링에서 지원되지 않습니다.

기타 도움말 항목

Adobe Pixel Bender Technology Center

Pixel Bender 개발자 안내서

Pixel Bender 참조 설명서

flash.display.Shader

flash.filters.ShaderFilter

Flash용 Pixel Bender 기초

Flex용 Pixel Bender 기초

Pixel Bender 셰이더의 기초

Flash Player 10 이상, Adobe AIR 1.5 이상

Adobe Pixel Bender는 이미지 내용을 만들거나 조작하는 데 사용되는 프로그래밍 언어입니다. Pixel Bender를 사용하면 셰이더라고도 하는 커널을 만들 수 있습니다. 셰이더는 이미지의 각 픽셀에 대해 개별적으로 실행되는 단일 함수를 정의합니다. 함수에 대한 각 호출의 결과는 이미지에서 해당 픽셀 좌표의 출력 색상입니다. 입력 이미지 및 매개 변수 값을 지정하여 작업을 사용자 정의할 수 있습니다. 단일 셰이더 실행에서 입력 및 매개 변수 값은 상수입니다. 변경되는 유일한 요소는 색상으로 함수 호출의 결과를 나타내는 픽셀의 좌표입니다.

가능한 경우 셰이더 함수는 여러 출력 픽셀 좌표에 대해 동시에 호출됩니다. 이 경우 셰이더 성능이 향상되며 고성능 처리 작업을 제공할 수 있습니다.

ActionScript에서 셰이더를 사용하여 다음 세 가지 유형의 효과를 쉽게 만들 수 있습니다.

- 드로잉 채우기
- 블렌드 모드
- 필터

셰이더는 독립 실행형 모드에서도 실행할 수 있습니다. 독립 실행형 모드를 사용하면 의도된 사용 방식을 미리 지정하지 않고 셰이더의 결과에 직접 액세스할 수 있습니다. 결과는 이미지 데이터나 이진 또는 숫자 데이터로 액세스할 수 있습니다. 데이터는 이미지 데이터일 필요가 전혀 없습니다. 이러한 방식으로 셰이더에 데이터 집합을 입력으로 제공할 수 있습니다. 셰이더는 데이터를 처리하며, 사용자는 셰이더에서 반환된 결과 데이터에 액세스할 수 있습니다.

Pixel Bender는 Flash Player 10 및 Adobe AIR 1.5부터 지원됩니다. Pixel Bender 블렌드, 필터 및 채우기는 GPU 렌더링에서 지원되지 않습니다. 휴대 장치에서 Pixel Bender 셰이더는 CPU 렌더링에서 실행됩니다. 그러나 성능은 데스크톱 컴퓨터에서 실행할 때와 같지 않습니다. 많은 셰이더 프로그램은 초당 몇 프레임 정도의 속도로 실행될 수 있습니다.

중요한 개념 및 용어

다음 참조 목록에는 Pixel Bender 셰이더를 만들고 사용할 때 사용되는 중요한 용어가 나와 있습니다.

커널 Pixel Bender의 경우 커널은 셰이더와 동일합니다. Pixel Bender를 사용하면 코드에서 커널을 정의할 수 있습니다. 커널은 이미지의 각 픽셀에 대해 개별적으로 실행되는 단일 함수를 정의합니다.

Pixel Bender 바이트코드 Pixel Bender 커널이 컴파일되면 해당 커널은 Pixel Bender 바이트코드로 변형됩니다. 바이트코드는 런타임에 액세스되고 실행됩니다.

Pixel Bender 언어 Pixel Bender 커널을 만드는 데 사용되는 프로그래밍 언어입니다.

Pixel Bender Toolkit Pixel Bender 소스 코드에서 Pixel Bender 바이트코드 파일을 만드는 데 사용되는 응용 프로그램입니다. Pixel Bender Toolkit을 사용하면 Pixel Bender 소스 코드를 작성하고 테스트하고 컴파일할 수 있습니다.

Shader 이 문서에서 셰이더는 Pixel Bender 언어로 작성된 기능 집합을 말합니다. 셰이더의 코드는 시각적 효과를 만들거나 계산을 수행합니다. 어떠한 경우이든 셰이더는 데이터 집합(대개 이미지의 픽셀)을 반환합니다. 셰이더는 출력 픽셀의 좌표만 다를 뿐 각 데이터 점에서 동일한 작업을 수행합니다. 셰이더는 ActionScript로 작성되지 않습니다. Pixel Bender 언어로 작성되며 Pixel Bender 바이트코드로 컴파일됩니다. 셰이더는 컴파일 타임에 SWF 파일에 포함되거나 런타임에 외부 파일로 로드될 수 있습니다. 두 경우 모두 ActionScript에서 Shader 객체를 만들고 해당 객체를 셰이더 바이트코드에 연결하여 셰이더에 액세스할 수 있습니다.

셰이더 입력 계산에 사용하기 위해 셰이더에 제공되는 복잡한 입력(대개 비트맵 이미지 데이터)입니다. 셰이더에 정의된 각 입력 변수에 대해 단일 값, 즉 단일 이미지 또는 이진 데이터 집합이 셰이더의 전체 실행에 사용됩니다.

셰이더 매개 변수 계산에 사용하기 위해 셰이더에 제공되는 단일 값 또는 제한된 값 집합입니다. 각 매개 변수 값은 단일 셰이더 실행에 대해 정의되며 전체 셰이더 실행에 동일한 값이 사용됩니다.

코드 예제를 사용하여 작업

제공된 예제 코드 샘플을 테스트해 볼 수도 있습니다. 코드 테스트는 코드를 실행하는 작업과 만들어진 SWF에서 결과를 확인하는 작업으로 이루어집니다. 모든 예제에서는 드로잉 API를 사용하여 내용을 만들며 이 내용은 셰이더 효과를 사용하거나 셰이더 효과에 의해 수정됩니다.

대부분의 예제 코드 샘플에는 두 부분이 포함되어 있습니다. 한 부분은 예제에 사용되는 셰이더의 Pixel Bender 소스 코드입니다. 먼저 Pixel Bender Toolkit을 사용하여 소스 코드를 Pixel Bender 바이트코드 파일로 컴파일해야 합니다. Pixel Bender 바이트코드 파일을 만드는 단계는 다음과 같습니다.

1. Adobe Pixel Bender Toolkit을 엽니다. 필요한 경우 [Build] 메뉴에서 [Turn on Flash Player warnings and errors]를 선택합니다.
2. Pixel Bender 코드 샘플을 복사하여 Pixel Bender Toolkit의 코드 편집기 창에 붙여넣습니다.
3. [File] 메뉴에서 [Export kernel filter for Flash Player]를 선택합니다.
4. Pixel Bender 바이트코드 파일을 Flash 문서와 동일한 디렉토리에 저장합니다. 파일의 이름은 예제 설명에 지정된 이름과 일치해야 합니다.

각 예제의 ActionScript 부분은 클래스 파일로 작성되어 있습니다. Flash Professional에서 예제를 테스트하려면 다음 작업을 수행하십시오.

1. 빈 Flash 문서를 만들고 컴퓨터에 저장합니다.
2. 새 ActionScript 파일을 만들고 Flash 문서와 같은 디렉토리에 저장합니다. 파일 이름은 코드 샘플에 있는 클래스의 이름과 일치해야 합니다. 예를 들어 코드 샘플에서 MyApplication이라는 클래스를 정의하는 경우 MyApplication.as라는 이름으로 ActionScript 파일을 저장합니다.
3. ActionScript 파일에 코드 샘플을 복사하고 파일을 저장합니다.

4 Flash 문서에서 스테이지 또는 작업 영역의 빈 부분을 클릭하여 문서 속성 관리자를 활성화합니다.

5 텍스트에서 복사한 ActionScript 클래스의 이름을 속성 관리자의 [문서 클래스] 필드에 입력합니다.

6 [컨트롤] > [동영상 테스트]를 사용하여 프로그램을 실행합니다.

미리 보기 윈도우에서 예제 결과를 확인합니다.

예제 코드 목록을 테스트하는 이러한 방법은 998페이지의 "ActionScript 예제 사용 방법"에 자세히 설명되어 있습니다.

셰이더 로드 또는 포함

Flash Player 10 이상, Adobe AIR 1.5 이상

ActionScript에서 Pixel Bender 셰이더를 사용하려면 첫 번째 단계로 ActionScript 코드에서 셰이더에 액세스해야 합니다. 셰이더는 Adobe Pixel Bender Toolkit을 사용하여 만들어지고 Pixel Bender 언어로 작성되기 때문에 ActionScript에서 직접 액세스할 수 없습니다. 대신 Pixel Bender 셰이더를 ActionScript에 나타내는 Shader 클래스의 인스턴스를 만듭니다. Shader 객체를 사용하면 매개 변수 또는 입력 이미지 값이 필요한지 여부와 같은 셰이더에 대한 정보를 확인할 수 있습니다. 실제로 셰이더를 사용하려면 Shader 객체를 다른 객체에 전달합니다. 예를 들어 셰이더를 필터로 사용하려면 ShaderFilter 객체의 shader 속성에 Shader 객체를 지정합니다. 또는 셰이더를 드로잉 채우기로 사용하려면 Shader 객체를 Graphics.beginShaderFill() 메서드에 대한 인수로 전달합니다.

ActionScript 코드는 다음 두 가지 방법으로 Adobe Pixel Bender Toolkit에서 만든 셰이더(.pbj 파일)에 액세스할 수 있습니다.

- 런타임에 로드: URLLoader 객체를 사용하여 셰이더 파일을 외부 에셋으로 로드할 수 있습니다. 이 기술은 텍스트 파일과 같은 외부 에셋을 로드하는 것과 같습니다. 다음 예제에서는 런타임에 셰이더 바이트코드 파일을 로드하여 Shader 인스턴스에 연결하는 방법을 보여 줍니다.

```
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
}
```

- SWF 파일에 포함: [Embed] 메타데이터 태그를 사용하여 컴파일 타임에 셰이더 파일을 SWF 파일에 포함할 수 있습니다. 이 [Embed] 메타데이터 태그는 Flex SDK를 사용하여 SWF 파일을 컴파일하는 경우에만 사용할 수 있습니다. 아래 예제와 같이 [Embed] 태그의 source 매개 변수는 셰이더 파일을 가리키고, 해당 mimeType 매개 변수는 "application/octet-stream"입니다.

```
[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
```

두 경우 모두, 원시 셰이더 바이트코드(URLLoader.data 속성 또는 [Embed] 데이터 클래스의 인스턴스)를 Shader 인스턴스에 연결합니다. 앞의 예제에서 볼 수 있듯이 다음 두 가지 방법으로 Shader 인스턴스에 바이트코드를 할당할 수 있습니다. 즉, 셰이더 바이트코드를 Shader() 생성자에 인수로 전달할 수도 있고, Shader 인스턴스의 byteCode 속성으로 설정할 수도 있습니다.

Pixel Bender 셰이더가 만들어져 Shader 객체에 연결된 경우 셰이더를 사용하여 다양한 방법으로 효과를 만들 수 있습니다. 셰이더를 필터, 블렌드 모드, 비트맵 채우기로 사용하거나 비트맵 또는 다른 데이터의 독립 실행형 처리에 사용할 수 있습니다. Shader 객체의 data 속성을 사용하여 셰이더의 메타데이터에 액세스하고, 입력 이미지를 지정하고, 매개 변수 값을 설정할 수도 있습니다.

셰이더 메타데이터에 액세스

Flash Player 10 이상, Adobe AIR 1.5 이상

Pixel Bender 셰이더 커널을 만드는 동안 작성자는 Pixel Bender 소스 코드에 셰이더에 대한 메타데이터를 지정할 수 있습니다. ActionScript에서 셰이더를 사용하는 동안 셰이더를 검사하고 해당 메타데이터를 추출할 수 있습니다.

Shader 인스턴스를 만들고 Pixel Bender 셰이더에 연결하면 셰이더에 대한 데이터가 포함된 ShaderData 객체가 만들어져 Shader 객체의 data 속성에 저장됩니다. ShaderData 클래스는 자체 속성을 정의하지 않습니다. 그러나 런타임에 셰이더 소스 코드에 정의된 각 메타데이터 값에 대해 한 개의 속성이 동적으로 ShaderData 객체에 추가됩니다. 각 속성에 지정되는 이름은 메타데이터에 지정된 이름과 같습니다. 예를 들어 Pixel Bender 셰이더의 소스 코드에 다음과 같은 메타데이터 정의가 포함되어 있다고 가정해 봅니다.

```
namespace : "Adobe::Example";
vendor : "Bob Jones";
version : 1;
description : "Creates a version of the specified image with the specified brightness.";
```

해당 셰이더에 대해 작성되는 ShaderData 객체는 다음 속성과 값을 사용하여 만들어집니다.

- namespace(문자열): "Adobe::Example"
- vendor(문자열): "Bob Jones"
- version(문자열): "1"
- description(문자열): "Creates a version of the specified image with the specified brightness"

메타데이터 속성이 동적으로 ShaderData 객체에 추가되기 때문에 for..in 루프를 사용하여 ShaderData 객체를 검사할 수 있습니다. 이 기술을 사용하면 셰이더에 메타데이터가 있는지 여부 및 해당 메타데이터 값을 확인할 수 있습니다. 메타데이터 속성 외에도 ShaderData 객체는 셰이더에 정의된 입력과 매개 변수를 나타내는 속성을 가질 수 있습니다. for..in 루프를 사용하여 ShaderData 객체를 검사하는 경우 각 속성의 데이터 유형을 검사하여 속성이 입력(ShaderInput 인스턴스), 매개 변수(ShaderParameter 인스턴스) 또는 메타데이터 값(String 인스턴스)인지 확인합니다. 다음 예제에서는 for..in 루프를 사용하여 셰이더 data 속성의 동적 속성을 검사하는 방법을 보여 줍니다. 각 메타데이터 값은 metadata라는 Vector 인스턴스에 추가됩니다. 이 예제에서는 myShader라는 Shader 인스턴스가 이미 작성되었다고 가정합니다.

```
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
```

셰이더 입력과 매개 변수도 추출하는 이 예제의 버전을 보려면 275페이지의 "셰이더 입력 및 매개 변수 식별"을 참조하십시오. 입력 및 매개 변수 속성에 대한 자세한 내용은 275페이지의 "셰이더 입력 및 매개 변수 값 지정"을 참조하십시오.

셰이더 입력 및 매개 변수 값 지정

Flash Player 10 이상, Adobe AIR 1.5 이상

많은 Pixel Bender 셰이더는 셰이더 처리에 사용되는 입력 이미지를 한 개 이상 사용하도록 정의되어 있습니다. 예를 들어 셰이더가 소스 이미지를 받아들인 다음 특정 효과를 적용하여 해당 이미지를 출력하는 것이 일반적입니다. 셰이더의 사용 방법에 따라 입력 값이 자동으로 지정되거나 명시적으로 값을 제공해야 할 수 있습니다. 마찬가지로, 많은 셰이더는 셰이더의 출력을 사용자 정의하는 데 사용되는 매개 변수를 지정합니다. 또한 셰이더를 사용하기 전에 각 매개 변수에 대해 명시적으로 값을 설정해야 합니다.

Shader 객체의 data 속성을 사용하여 셰이더 입력과 매개 변수를 설정하고 특정 셰이더에 입력 또는 매개 변수가 필요한지 확인합니다. data 속성은 ShaderData 인스턴스입니다.

셰이더 입력 및 매개 변수 식별

Flash Player 10 이상, Adobe AIR 1.5 이상

셰이더 입력 및 매개 변수 값을 지정하는 경우 첫 번째 단계로 사용 중인 특정 셰이더에 입력 이미지 또는 매개 변수가 필요한지 확인합니다. 각 Shader 인스턴스에는 ShaderData 객체가 포함된 data 속성이 있습니다. 셰이더가 입력이나 매개 변수를 정의하면 해당 ShaderData 객체의 속성으로 액세스됩니다. 속성 이름은 셰이더 소스 코드에서 입력 및 매개 변수에 대해 지정된 이름과 일치합니다. 예를 들어 셰이더에서 src라는 입력을 정의하면 ShaderData 객체에 해당 입력을 나타내는 src 속성이 있습니다. 입력을 나타내는 각 속성은 ShaderInput 인스턴스이며, 매개 변수를 나타내는 각 속성은 ShaderParameter 인스턴스입니다.

셰이더의 작성자가 셰이더에 필요한 입력 이미지 값 및 매개 변수, 입력 이미지 값 및 매개 변수가 나타내는 대상, 적합한 값 등을 나타내는 셰이더 설명서를 제공하는 것이 좋습니다.

그러나 셰이더가 문서화되어 있지 않으며 해당 소스 코드가 없는 경우 셰이더 데이터를 검사하여 입력과 매개 변수를 확인할 수 있습니다. 입력과 매개 변수를 나타내는 속성은 동적으로 ShaderData 객체에 추가됩니다. 따라서 for..in 루프를 사용하여 ShaderData 객체를 검사하고 연관된 셰이더에서 입력 또는 매개 변수를 정의하는지 확인할 수 있습니다. 274페이지의 "셰이더 메타데이터에 액세스"에 설명된 것처럼 셰이더에 대해 정의된 모든 메타데이터 값은 Shader.data 속성에 추가된 동적 속성으로도 액세스됩니다. 이 기술을 사용하여 셰이더 입력과 매개 변수를 확인하는 경우 동적 속성의 데이터 유형을 검사합니다. 속성이 ShaderInput 인스턴스이면 입력을 나타내고, ShaderParameter 인스턴스이면 매개 변수를 나타냅니다. 그렇지 않으면 메타데이터 값입니다. 다음 예제에서는 for..in 루프를 사용하여 셰이더 data 속성의 동적 속성을 검사하는 방법을 보여 줍니다. 각 입력(ShaderInput 객체)은 inputs이라는 Vector 인스턴스에 추가됩니다. 각 매개 변수(ShaderParameter 객체)는 parameters라는 Vector 인스턴스에 추가됩니다. 마지막으로, 모든 메타데이터 속성은 metadata라는 Vector 인스턴스에 추가됩니다. 이 예제에서는 myShader라는 Shader 인스턴스가 이미 작성되었다고 가정합니다.

```
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
```

셰이더 입력 값 지정

Flash Player 10 이상, Adobe AIR 1.5 이상

셰이더 처리에 사용되는 입력 이미지가 한 개 이상 필요한 셰이더가 많습니다. 그러나 대부분의 경우 입력은 Shader 객체를 사용할 때 자동으로 지정됩니다. 예를 들어 셰이더에 한 개의 입력이 필요하며 해당 셰이더가 필터로 사용된다고 가정합니다. 표시 객체나 BitmapData 객체에 필터를 적용하면 해당 객체가 입력으로 자동 설정됩니다. 이 경우 명시적으로 입력 값을 설정하지 않습니다.

그러나 경우에 따라, 특히 셰이더에서 여러 입력을 정의하는 경우 입력 값을 명시적으로 설정합니다. 셰이더에 정의된 각 입력은 ActionScript에서 ShaderInput 객체로 나타납니다. 275페이지의 “셰이더 입력 및 매개 변수 식별”에 설명된 것처럼 ShaderInput 객체는 Shader 객체의 data 속성에 있는 ShaderData 인스턴스의 속성입니다. 예를 들어 셰이더에서 src라는 입력을 정의하며 해당 셰이더가 myShader라는 Shader 객체에 연결되어 있다고 가정합니다. 이 경우 다음 식별자를 사용하여 src 입력에 해당하는 ShaderInput 객체에 액세스합니다.

```
myShader.data.src
```

각 ShaderInput 객체에는 입력 값을 설정하는 데 사용되는 input 속성이 있습니다. input 속성을 BitmapData 인스턴스로 설정하여 이미지 데이터를 지정합니다. input 속성을 BitmapData 또는 Vector.<Number> 인스턴스로 설정하여 이진 또는 숫자 데이터를 지정할 수도 있습니다. BitmapData 또는 Vector.<Number> 인스턴스를 입력으로 사용하는 방법에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 ShaderInput.input 목록을 참조하십시오.

input 속성 외에도 ShaderInput 객체에는 입력에 필요한 이미지 유형을 확인하는 데 사용할 수 있는 속성이 있습니다. 이러한 속성에는 width, height 및 channels 속성이 포함됩니다. 각 ShaderInput 객체에는 입력에 대해 명시적 값을 제공해야 하는지 여부를 확인하는 데 유용한 index 속성도 있습니다. 셰이더에 자동으로 설정되는 개수보다 많은 입력이 필요한 경우 해당 입력의 값을 설정합니다. 셰이더를 사용하는 다양한 방법 및 입력 값이 자동으로 설정되는지 여부에 대한 자세한 내용은 280페이지의 “셰이더 사용”을 참조하십시오.

셰이더 매개 변수 값 지정

Flash Player 10 이상, Adobe AIR 1.5 이상

일부 셰이더는 셰이더가 결과를 만들 때 사용하는 매개 변수 값을 정의합니다. 예를 들어 이미지의 밝기를 변경하는 셰이더는 작업이 밝기에 미치는 영향을 결정하는 밝기 매개 변수를 지정할 수 있습니다. 셰이더의 매개 변수 정의에 따라 셰이더에 정의된 단일 매개 변수에 하나 또는 여러 개의 값이 필요할 수 있습니다. 셰이더에 정의된 각 매개 변수는 ActionScript에서 ShaderParameter 객체로 나타납니다. 275페이지의 "셰이더 입력 및 매개 변수 식별"에 설명된 것처럼 ShaderParameter 객체는 Shader 객체의 data 속성에 있는 ShaderData 인스턴스의 속성입니다. 예를 들어 셰이더에서 brightness라는 매개 변수를 정의하며 myShader라는 Shader 객체로 해당 셰이더를 나타낸다고 가정합니다. 이 경우 다음 식별자를 사용하여 brightness 매개 변수에 해당하는 ShaderParameter에 액세스합니다.

```
myShader.data.brightness
```

매개 변수에 대해 값을 한 개 이상 설정하려면 값이 한 개 이상 포함된 ActionScript 배열을 만든 다음 ShaderParameter 객체의 value 속성에 해당 배열을 지정합니다. 단일 셰이더 매개 변수에 값이 여러 개 필요할 수 있으므로 value 속성은 Array 인스턴스로 정의됩니다. 셰이더 매개 변수에 단일 값이 필요한 경우에도 Array 객체에 값을 래핑하여 ShaderParameter.value 속성에 지정해야 합니다. 다음 샘플에서는 단일 값을 value 속성으로 설정하는 방법을 보여 줍니다.

```
myShader.data.brightness.value = [75];
```

셰이더에 대한 Pixel Bender 소스 코드에서 매개 변수의 기본값을 정의하는 경우 Shader 객체를 만들 때 기본값이 포함된 배열이 만들어져 ShaderParameter 객체의 value 속성에 지정됩니다. value 속성에 배열이 지정되고 나면(기본 배열인 경우 포함) 배열 요소의 값을 변경하여 매개 변수 값을 변경할 수 있습니다. 새 배열을 만들어 value 속성에 지정할 필요가 없습니다.

다음 예제에서는 ActionScript에서 셰이더의 매개 변수 값을 설정하는 방법을 보여 줍니다. 이 예제에서 셰이더는 color라는 매개 변수를 정의합니다. color 매개 변수는 Pixel Bender 소스 코드에 float4 변수로 선언되어 있으므로 4개의 부동 소수점 숫자 배열입니다. 이 예제에서 color 매개 변수 값은 계속 변경되며, 변경될 때마다 셰이더가 사용되어 화면에 색상이 설정된 사각형을 그립니다. 그 결과 애니메이션 색상 변경이 만들어집니다.

참고: 이 예제의 코드는 Ryan Taylor에 의해 작성되었습니다. 이 예제를 공유해 주신 Ryan 씨께 감사드립니다. Ryan의 포트폴리오와 그가 작성한 내용은 www.boostworthy.com/에서 보고 읽을 수 있습니다.

ActionScript 코드는 다음 세 가지 메서드를 중심으로 합니다.

- init(): init() 메서드의 코드에서는 셰이더가 들어 있는 Pixel Bender 바이트코드 파일을 로드합니다. 파일이 로드되면 onLoadComplete() 메서드가 호출됩니다.
- onLoadComplete(): onLoadComplete() 메서드의 코드에서는 shader라는 Shader 객체를 만듭니다. 또한 texture라는 Sprite 인스턴스를 만듭니다. renderShader() 메서드의 코드에서는 프레임마다 한 번씩 셰이더 결과를 texture로 그립니다.
- onEnterFrame(): onEnterFrame() 메서드는 프레임당 한 번 호출되어 애니메이션 효과를 만듭니다. 이 메서드의 코드에서는 셰이더 매개 변수 값을 새 색상으로 설정한 다음 renderShader() 메서드를 호출하여 셰이더 결과를 사각형으로 그립니다.
- renderShader(): renderShader() 메서드의 코드에서는 Graphics.beginShaderFill() 메서드를 호출하여 셰이더 채우기를 지정합니다. 그런 다음 채우기가 셰이더 출력(생성된 색상)에 의해 정의되는 사각형을 그립니다. 이와 같은 방식으로 셰이더를 사용하는 방법에 대한 자세한 내용은 280페이지의 "셰이더를 드로잉 채우기로 사용"을 참조하십시오.

다음은 이 예제에 대한 ActionScript 코드입니다. 이 클래스는 Flash Builder에서 ActionScript 전용 프로젝트의 기본 응용 프로그램 클래스로 사용하거나 Flash Professional에서 FLA 파일의 문서 클래스로 사용합니다.

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

다음은 ColorFilter 셰이더 커널에 대한 소스 코드이며, "ColorFilter.pbj" Pixel Bender 바이트코드 파일을 만드는 데 사용됩니다.

```
<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;
    }
}
```

매개 변수가 문서화되어 있지 않은 셰이더를 사용하는 경우 ShaderParameter 객체의 type 속성을 검사하여 배열에 포함해야 하는 요소의 유형 및 개수를 확인할 수 있습니다. type 속성은 셰이더 자체에 정의된 매개 변수의 데이터 유형을 나타냅니다. 각 매개 변수 유형에 필요한 요소 수와 유형 목록은 ActionScript 3.0 참조 설명서의 ShaderParameter.value 속성 목록을 참조하십시오.

각 ShaderParameter 객체에는 셰이더의 매개 변수 순서에서 각 매개 변수의 위치를 나타내는 index 속성도 있습니다. 이러한 속성 외에도 ShaderParameter 객체는 셰이더의 작성자가 제공한 메타데이터 값이 포함된 추가 속성을 가질 수 있습니다. 예를 들어 작성자는 매개 변수에 대해 최소값, 최대값 및 기본값과 같은 메타데이터 값을 지정할 수 있습니다. 작성자가 지정한 모든 메타데이터 값은 ShaderParameter 객체에 동적 속성으로 추가됩니다. 이러한 속성을 검사하려면 for..in 루프를 사용하여 ShaderParameter 객체의 동적 속성을 반복함으로써 해당 메타데이터를 확인합니다. 다음 예제에서는 for..in 루프를 사용하여 ShaderParameter 객체의 메타데이터를 확인하는 방법을 보여 줍니다. 각 메타데이터 값은 metadata라는 Vector 인스턴스에 추가됩니다. 이 예제에서는 myShader라는 Shader 인스턴스가 이미 만들어졌으며 brightness라는 매개 변수가 있다고 가정합니다.

```
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
```

셰이더 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

ActionScript에서 Pixel Bender 셰이더를 Shader 객체로 사용할 수 있게 되면 다음과 같은 여러 방법으로 셰이더를 사용할 수 있습니다.

- 셰이더 드로잉 채우기: 셰이더는 드로잉 API를 사용하여 그려진 모양의 채우기 부분을 정의합니다.
- 블렌드 모드: 셰이더는 겹치는 두 표시 객체 간의 블렌드를 정의합니다.
- 필터: 셰이더는 시각적 내용의 모양을 수정하는 필터를 정의합니다.
- 독립 실행형 셰이더 처리: 셰이더 처리는 출력의 의도된 용도를 지정하지 않고 실행됩니다. 셰이더는 선택적으로 백그라운드에서 실행될 수 있으며, 처리가 완료되면 결과를 사용할 수 있습니다. 이 기술을 사용하여 비트맵 데이터를 생성하고 비시각적 데이터를 처리할 수도 있습니다.

셰이더를 드로잉 채우기로 사용

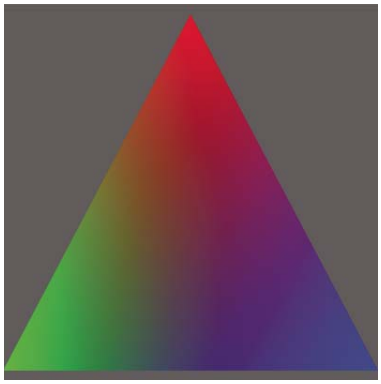
Flash Player 10 이상, Adobe AIR 1.5 이상

셰이더를 사용하여 드로잉 채우기를 만드는 경우 드로잉 API 메서드를 사용하여 벡터 모양을 만듭니다. 드로잉 API를 사용하여 비트맵 이미지를 비트맵 채우기로 사용할 때와 동일하게 셰이더의 출력은 모양을 채우는 데 사용합니다. 셰이더 채우기를 만들려면 모양 그리기를 시작할 코드 지점에서 Graphics 객체의 beginShaderFill() 메서드를 호출합니다. 이 샘플에서와 같이 beginShaderFill() 메서드에 Shader 객체를 첫 번째 인수로 전달합니다.

```
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
```

셰이더를 드로잉 채우기로 사용할 때는 셰이더에 필요한 입력 이미지 값과 매개 변수 값을 설정합니다.

다음 예제에서는 셰이더를 드로잉 채우기로 사용하는 방법을 보여 줍니다. 이 예제에서 셰이더는 3점 그래디언트를 만듭니다. 이 그래디언트는 삼각형의 각 점에 세 개의 색상을 표시하며 세 점 사이에서 그래디언트가 블렌드됩니다. 또한 색상이 회전하여 애니메이션이 적용된 회전하는 색상 효과를 만듭니다.



참고: 이 예제의 코드는 Petri Leskinen에 의해 작성되었습니다. 이 예제를 공유해 주신 Petri 씨께 감사드립니다. Petri의 예제와 자습서는 http://pixelero.wordpress.com/에서 보다 자세히 볼 수 있습니다.

ActionScript 코드는 다음 세 가지 메서드를 중심으로 합니다.

- init(): init() 메서드는 응용 프로그램이 로드될 때 호출됩니다. 이 메서드의 코드에서는 삼각형의 점을 나타내는 Point 객체의 초기 값을 설정합니다. 또한 canvas라는 Sprite 인스턴스를 만듭니다. 나중에 updateShaderFill()의 코드에서는 프레임마다 한 번씩 셰이더 결과를 canvas로 그립니다. 마지막으로 코드에서는 셰이더 바이트코드 파일을 로드합니다.
- onLoadComplete(): onLoadComplete() 메서드의 코드에서는 shader라는 Shader 객체를 만듭니다. 또한 초기 매개 변수 값을 설정합니다. 마지막으로 코드에서는 enterFrame 이벤트에 대한 리스너로 updateShaderFill() 메서드를 추가합니다. 따라서 이 메서드는 프레임마다 한 번씩 호출되어 애니메이션 효과를 만듭니다.
- updateShaderFill(): updateShaderFill() 메서드는 프레임마다 한 번씩 호출되어 애니메이션 효과를 만듭니다. 이 메서드의 코드에서는 셰이더 매개 변수의 값을 계산하고 설정합니다. 그런 다음 beginShaderFill() 메서드를 호출하여 셰이더 채우기를 만들고 다른 드로잉 API 메서드를 호출하여 셰이더 결과를 삼각형에 그립니다.

다음은 이 예제에 대한 ActionScript 코드입니다. 이 클래스는 Flash Builder에서 ActionScript 전용 프로젝트의 기본 응용 프로그램 클래스로 사용하거나 Flash Professional에서 FLA 파일의 문서 클래스로 사용합니다.

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

다음은 ThreePointGradient 셰이더 커널에 대한 소스 코드이며, "ThreePointGradient.pbj" Pixel Bender 바이트코드 파일을 만드는 데 사용됩니다.

```
<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);
    }
}
```

참고: GPU(Graphics Processing Unit)에서 렌더링할 때 셰이더 채우기를 사용하는 경우 채워진 영역은 녹청 색상이 됩니다.

드로잉 API를 사용하여 모양을 그리는 방법에 대한 자세한 내용은 200페이지의 “드로잉 API 사용”을 참조하십시오.

셰이더를 블렌드 모드로 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

셰이더를 블렌드 모드로 사용하는 방법은 다른 블렌드 모드를 사용하는 방법과 비슷합니다. 셰이더는 두 개의 표시 객체를 시각적으로 함께 블렌드하여 생성되는 모양을 정의합니다. 셰이더를 블렌드 모드로 사용하려면 전경 표시 객체의 blendShader 속성에 Shader 객체를 할당합니다. blendShader 속성에 null 이외의 값을 할당하면 자동으로 표시 객체의 blendMode 속성이 BlendMode.SHADER로 설정됩니다. 다음 샘플에서는 셰이더를 블렌드 모드로 사용하는 방법을 보여 줍니다. 이 예제에서는 다른 표시 내용과 동일한 표시 목록 상의 부모에 foreground라는 표시 객체가 포함되어 있는 것으로 가정합니다. 이때 foreground는 다른 내용에 겹쳐 표시됩니다.

```
foreground.blendShader = myShader;
```

셰이더를 블렌드 모드로 사용하는 경우 셰이더는 적어도 두 개의 입력을 사용하여 정의해야 합니다. 예제와 같이 코드에서 입력 값을 설정하지 마십시오. 대신 두 개의 블렌드된 이미지가 자동으로 셰이더 입력으로 사용됩니다. 전경 이미지는 두 번째 이미지로 설정됩니다. 이 이미지는 블렌드 모드가 적용된 표시 객체입니다. 배경 이미지는 전경 이미지의 경계 상자 뒤에 있는 모든 픽셀을 합성하여 만들어집니다. 이 배경 이미지는 첫 번째 입력 이미지로 설정됩니다. 세 개 이상의 입력이 필요한 셰이더를 사용하는 경우에는 처음 두 개 외의 입력을 위한 값을 제공합니다.

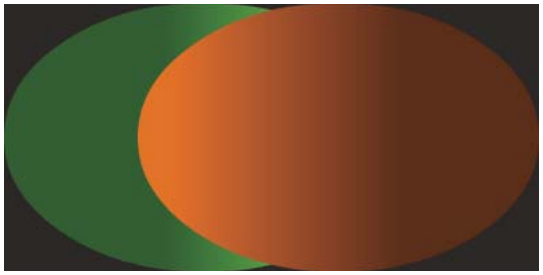
다음 예제에서는 셰이더를 블렌드 모드로 사용하는 방법을 보여 줍니다. 이 예제에서는 광도를 기준으로 하는 lighten 블렌드 모드를 사용합니다. 블렌드 후에는 블렌드된 객체 중 하나에서 가장 밝은 픽셀 값이 표시되는 픽셀이 됩니다.

참고: 이 예제의 코드는 Mario Klingemann에 의해 작성되었습니다. 이 예제를 공유해 주신 Mario 씨께 감사드립니다. Mario의 작업과 그가 쓴 내용은 www.quasimondo.com/에서 보다 자세히 보고 읽을 수 있습니다.

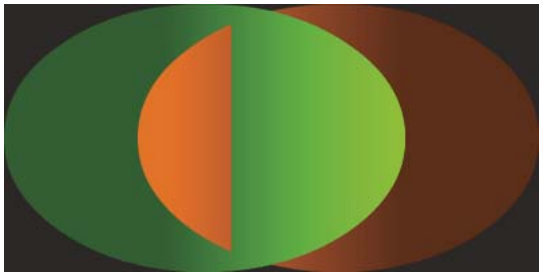
중요한 ActionScript 코드는 다음 두 가지 메서드를 중심으로 합니다.

- init(): init() 메서드는 응용 프로그램이 로드될 때 호출됩니다. 이 메서드의 코드에서는 셰이더 바이트코드 파일을 로드합니다.
- onLoadComplete(): onLoadComplete() 메서드의 코드에서는 shader라는 Shader 객체를 만듭니다. 그런 다음 세 개의 객체를 그립니다. 첫 번째 객체인 backdrop은 블렌딩된 객체 뒤의 진한 회색 배경입니다. 두 번째 객체인 backgroundShape는 녹색 그래디언트 타원입니다. 세 번째 객체인 foregroundShape는 주황색 그래디언트 타원입니다.

  foregroundShape 타원은 블렌드의 전경 객체입니다. 블렌드의 배경 이미지는 foregroundShape 객체의 경계 상자와 겹치는 backdrop 부분 및 backgroundShape 부분으로 구성됩니다. foregroundShape 객체는 표시 목록에서 가장 앞에 있는 객체입니다. 이 객체는 backgroundShape와 부분적으로 겹치고 backdrop과 완전히 겹칩니다. 이 겹침으로 인해 블렌드 모드가 적용되지 않은 경우 주황색 타원(foregroundShape)은 완전히 표시되며 녹색 타원(backgroundShape)은 일부가 숨겨집니다.



  그러나 블렌드 모드가 적용된 경우에는 녹색 타원의 밝은 부분이 겹치는 foregroundShape 부분보다 밝으므로 이 부분이 표시됩니다.



다음은 이 예제에 대한 ActionScript 코드입니다. 이 클래스는 Flash Builder에서 ActionScript 전용 프로젝트의 기본 응용 프로그램 클래스로 사용하거나 Flash Professional에서 FLA 파일의 문서 클래스로 사용합니다.

```
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;
        }
    }
}
```

다음은 LumaLighten 셰이더 커널에 대한 소스 코드이며, "LumaLighten.pbj" Pixel Bender 바이트코드 파일을 만드는 데 사용됩니다.

```
<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;
    }
}
```

블렌드 모드 사용에 대한 자세한 내용은 167페이지의 “블렌딩 모드 적용”을 참조하십시오.

참고: Pixel Bender 음영 처리 프로그램이 Flash Player 또는 AIR에서 블렌드 모드로 실행되는 경우 샘플링 및 outCoord() 함수가 다른 컨텍스트와는 다르게 동작합니다. 블렌드 모드에서 샘플링 함수는 항상 음영 처리 프로그램으로 계산되는 현재 픽셀을 반환합니다. 예를 들어 인접 픽셀을 샘플링하기 위해 outCoord()에 오프셋 추가를 사용할 수 없습니다. 마찬가지로 샘플링 함수 외부에서 outCoord() 함수를 사용하는 경우 좌표가 항상 0으로 계산됩니다. 예를 들어 픽셀의 위치를 사용하여 블렌딩된 이미지의 결합 방법에 영향을 줄 수 없습니다.

셰이더를 필터로 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

셰이더를 필터로 사용하는 방법은 ActionScript에서 다른 필터를 사용하는 방법과 비슷합니다. 셰이더를 필터로 사용하면 필터링된 이미지(표시 객체 또는 BitmapData 객체)가 셰이더에 전달됩니다. 셰이더는 입력 이미지를 사용하여 일반적으로 원본 이미지의 수정된 버전인 필터 출력을 만듭니다. 필터링된 객체가 표시 객체인 경우 필터링된 표시 객체 대신 셰이더의 출력이 화면에 표시됩니다. 필터링된 객체가 BitmapData 객체인 경우 셰이더의 출력은 BitmapData 객체의 내용이 되며 이 객체의 applyFilter() 메서드가 호출됩니다.

셰이더를 필터로 사용하려면 먼저 273페이지의 "셰이더 로드 또는 포함"에 설명된 대로 Shader 객체를 만듭니다. 그런 다음 Shader 객체에 연결되는 ShaderFilter 객체를 만듭니다. ShaderFilter 객체는 필터링된 객체에 적용되는 필터입니다. 객체에 이 필터를 적용하는 방법은 다른 모든 필터를 적용하는 방법과 동일합니다. 표시 객체의 filters 속성에 필터를 전달하거나 BitmapData 객체의 applyFilter() 메서드를 호출합니다. 예를 들어 다음 코드에서는 ShaderFilter 객체를 만들고 homeButton이라는 표시 객체에 필터를 적용합니다.

```
var myFilter:ShaderFilter = new ShaderFilter(myShader);
homeButton.filters = [myFilter];
```

셰이더를 필터로 사용할 때는 적어도 하나의 입력을 사용하여 셰이더를 정의해야 합니다. 예제와 같이 코드에서 입력 값을 설정하지 마십시오. 대신 필터링된 표시 객체 또는 BitmapData 객체를 입력 이미지로 설정합니다. 두 개 이상의 입력이 필요한 셰이더를 사용하는 경우에는 처음 한 개 외의 입력을 위한 값을 제공합니다.

일부 경우 필터는 원본 이미지의 크기를 변경합니다. 예를 들어 일반적인 그림자 효과는 이미지에 추가되는 그림자가 들어 있는 픽셀을 추가합니다. 이미지 크기를 변경하는 셰이더를 사용하는 경우에는 leftExtension, rightExtension, topExtension 및 bottomExtension 속성을 설정하여 변경할 이미지 크기를 나타냅니다.

다음 예제에서는 셰이더를 필터로 사용하는 방법을 보여 줍니다. 이 예제의 필터는 이미지의 빨강, 녹색 및 파랑 채널 값을 반전시킵니다. 결과는 이미지의 "네거티브" 버전입니다.

참고: 이 예제에서 사용하는 셰이더는 Pixel Bender Toolkit에 포함된 invertRGB.pbk Pixel Bender 커널입니다. Pixel Bender Toolkit 설치 디렉토리에서 커널에 대한 소스 코드를 로드할 수 있습니다. 소스 코드를 컴파일하고 소스 코드와 동일한 디렉토리에 바이트코드 파일을 저장합니다.

중요한 ActionScript 코드는 다음 두 가지 메서드를 중심으로 합니다.

- init(): init() 메서드는 응용 프로그램이 로드될 때 호출됩니다. 이 메서드의 코드에서는 셰이더 바이트코드 파일을 로드합니다.
- onLoadComplete(): onLoadComplete() 메서드의 코드에서는 shader라는 Shader 객체를 만듭니다. 그런 다음 target이라는 객체의 내용을 만들고 그립니다. target 객체는 왼쪽이 빨강이고 가운데가 황록색이며 오른쪽이 연한 파랑인 선형 그래디언트 색상으로 채워진 사각형입니다. 필터링되지 않은 객체는 다음과 같이 나타납니다.

필터가 적용된 경우 색상이 반전되며 사각형이 다음과 같이 나타납니다.

이 예제에서 사용하는 셰이더는 Pixel Bender Toolkit에 포함된 "invertRGB.pbk" 샘플 Pixel Bender 커널입니다. 소스 코드는 Pixel Bender Toolkit 설치 디렉토리의 "invertRGB.pbk" 파일에 있습니다. 소스 코드를 컴파일하고 해당 바이트코드 파일을 ActionScript 소스 코드와 동일한 디렉토리에 "invertRGB.pbj"라는 이름으로 저장합니다.

다음은 이 예제에 대한 ActionScript 코드입니다. 이 클래스는 Flash Builder에서 ActionScript 전용 프로젝트의 기본 응용 프로그램 클래스로 사용하거나 Flash Professional에서 FLA 파일의 문서 클래스로 사용합니다.

```
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];
        }
    }
}
```

필터를 적용하는 방법에 대한 자세한 내용은 243페이지의 “필터 작성 및 적용”을 참조하십시오.

독립 실행형 모드에서 셰이더 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

셰이더를 독립 실행형 모드에서 사용하는 경우 셰이더 처리는 의도한 출력 사용 방법과 관계없이 실행됩니다. 실행할 셰이더를 지정하고 입력 및 매개 변수 값을 설정한 다음 결과 데이터를 배치할 객체를 지정합니다. 다음과 같은 두 가지 목적으로 셰이더를 독립 실행 모드에서 사용할 수 있습니다.

- 이미지가 아닌 데이터 처리: 독립 실행형 모드에서는 비트맵 이미지 데이터 이외에 임의의 이진 또는 숫자 데이터를 셰이더에 전달할 수 있습니다. 셰이더 결과가 비트맵 이미지 데이터뿐 아니라 이진 데이터 또는 숫자 데이터로 반환되도록 할 수도 있습니다.
- 백그라운드 처리: 셰이더를 독립 실행형 모드에서 실행할 경우 기본적으로 셰이더는 비동기적으로 실행됩니다. 즉, 셰이더는 응용 프로그램이 실행되는 동안 백그라운드에서 계속 실행되며 셰이더 처리가 완료되면 코드로 알림이 전달됩니다. 실행하는 데 오랜 시간이 걸리며 셰이더가 실행되는 동안 응용 프로그램 사용자 인터페이스나 그 밖의 처리를 중지시키지 않는 셰이더를 사용할 수 있습니다.

ShaderJob 객체를 사용하여 독립 실행형 모드에서 셰이더를 실행할 수 있습니다. 먼저 ShaderJob 객체를 만들고 이 객체를 실행할 셰이더를 나타내는 Shader 객체에 연결합니다.

```
var job:ShaderJob = new ShaderJob(myShader);
```

그런 다음 셰이더에 필요한 입력 또는 매개 변수 값을 설정합니다. 셰이더를 백그라운드에서 실행할 경우 ShaderJob 객체의 complete 이벤트에 대한 리스너도 등록합니다. 리스너는 셰이더의 작업이 완료될 때 호출됩니다.

```
function completeHandler(event:ShaderEvent):void
{
    // do something with the shader result
}

job.addEventListener(ShaderEvent.COMPLETE, completeHandler);
```

그런 다음 셰이더 작업이 완료될 때 해당 결과를 쓸 객체를 만듭니다. ShaderJob 객체의 target 속성에 해당 객체를 할당합니다.

```
var jobResult:BitmapData = new BitmapData(100, 75);
job.target = jobResult;
```

ShaderJob을 사용하여 이미지 처리를 수행하는 경우에는 target 속성에 BitmapData 인스턴스를 할당합니다. 이진 또는 숫자 데이터를 처리하는 경우에는 ByteArray 객체 또는 Vector.<Number> 인스턴스를 target 속성에 할당합니다. 이러한 경우 ShaderJob 객체의 width 및 height 속성을 설정하여 target 객체에 출력할 데이터의 양을 지정해야 합니다.

참고: var job:ShaderJob = new ShaderJob(myShader, myTarget, myWidth, myHeight);과 같이 ShaderJob() 생성자에 인수를 전달하여 ShaderJob 객체의 shader, target,width 및 height 속성을 한 번에 설정할 수 있습니다.

셰이더를 실행할 준비가 되면 ShaderJob 객체의 start() 메서드를 호출합니다.

```
job.start();
```

기본적으로 start()를 호출하면 ShaderJob이 비동기적으로 실행됩니다. 이 경우 셰이더가 완료될 때까지 기다리지 않고 코드의 다음 행에서 즉시 프로그램 실행을 계속할 수 있습니다. 셰이더 작업이 완료되면 ShaderJob 객체는 작업이 완료되었음을 알리는 complete 이벤트 리스너를 호출합니다. 이때, 즉 complete 이벤트 리스너의 본문에서 target 객체에는 셰이더 작업 결과가 포함됩니다.

참고: target 속성 객체를 사용하는 대신 리스너 메서드에 전달된 이벤트 객체에서 직접 셰이더 결과를 가져올 수 있습니다. 이벤트 객체는 ShaderEvent 인스턴스입니다. ShaderEvent 객체에는 target 속성으로 설정한 객체의 데이터 유형에 따라 결과에 액세스하는 데 사용할 수 있는 ShaderEvent.bitmapData, ShaderEvent.byteArray 및 ShaderEvent.vector 속성이 있습니다.

또한 start() 메서드에 true 인수를 전달할 수 있습니다. 이 경우 셰이더 작업은 동기적으로 실행됩니다. 사용자 인터페이스 및 다른 이벤트와의 상호 작용을 포함하는 모든 코드는 셰이더가 실행되는 동안 일시 중지됩니다. 셰이더가 완료되면 target 객체에는 셰이더 결과가 포함되고 프로그램은 다음 코드 행에서 계속 실행됩니다.

```
job.start(true);
```

16장: 동영상 클립을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

MovieClip 클래스는 Adobe® Flash® 개발 환경에서 생성한 애니메이션 및 동영상 클립 심볼의 기본 클래스입니다. MovieClip 클래스는 표시 객체의 모든 비헤이비어 및 기능은 물론 동영상 클립의 타임라인을 제어하기 위한 추가 속성 및 메서드까지 가지고 있습니다.

동영상 클립의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

동영상 클립은 Flash 제작 도구를 사용하여 애니메이션 내용을 만들고 ActionScript를 사용하여 그 내용을 제어하는 사람들에게 중요한 요소입니다. Flash에서 동영상 클립 심볼을 만들 때마다 해당 Flash 문서의 라이브러리에 심볼이 추가됩니다. 기본적으로 이 심볼은 MovieClip 클래스의 인스턴스가 되고 MovieClip 클래스의 속성과 메서드를 갖게 됩니다.

동영상 클립 심볼의 인스턴스를 스테이지에 배치하면 ActionScript를 사용하여 동영상 클립의 재생을 변경하지 않는 한, 동영상 클립이 자동으로 타임라인을 따라 진행됩니다(프레임이 둘 이상일 경우). MovieClip 클래스를 구별하는 것은 이 타임라인으로, Flash 제작 도구에서 모션 트윈이나 모양 트윈을 통해 애니메이션을 만들 수 있습니다. 반면 Sprite 클래스의 인스턴스인 표시 객체의 경우에는 프로그래밍을 통해 값을 변경하는 방법으로만 애니메이션을 만들 수 있습니다.

이전 버전의 ActionScript에서는 MovieClip 클래스가 Stage의 모든 인스턴스의 기본 클래스였습니다. ActionScript 3.0에서 동영상 클립은 화면에 표시할 수 있는 여러 표시 객체 중 하나일 뿐입니다. 표시 객체의 기능에 타임라인이 필요하지 않은 경우 MovieClip 클래스 대신 Shape 클래스나 Sprite 클래스를 사용하면 렌더링 성능을 높일 수 있습니다. 작업에 적합한 표시 객체를 선택하는 것에 대한 자세한 내용은 155페이지의 “DisplayObject 하위 클래스 선택”을 참조하십시오.

중요한 개념 및 용어

동영상 클립과 관련한 중요한 용어가 아래 참조 목록에 정리되어 있습니다.

AVM1 SWF ActionScript 1.0 또는 ActionScript 2.0으로 만든 SWF 파일로 대개 Flash Player 8 또는 이전 버전을 대상으로 합니다.

AVM2 SWF ActionScript 3.0을 사용하여 생성하는 Adobe Flash Player 9 이상 또는 Adobe AIR용 SWF 파일입니다.

외부 SWF 프로젝트 SWF 파일과는 별도로 만들어지는 SWF 파일로, 프로젝트 SWF 파일에 로드되어 이 프로젝트 SWF 파일 내에서 재생됩니다.

프레임 타임라인의 가장 작은 시간 구획입니다. 영화 필름스트립과 같이 각 프레임은 특정 시점의 애니메이션 스냅샷과 유사하며, 이러한 프레임을 차례로 빠르게 재생하면 애니메이션 효과가 연출됩니다.

타임라인 동영상 클립의 애니메이션 시퀀스를 구성하는 일련의 프레임을 은유적으로 표현한 것입니다. MovieClip 객체의 타임라인은 Flash 제작 도구의 타임라인에 해당합니다.

재생 헤드 특정 순간에 표시되는 타임라인 내 위치(프레임)를 나타내는 표시자입니다.

MovieClip 객체를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

SWF 파일을 제작하면 스테이지의 모든 동영상 클립 심볼 인스턴스가 MovieClip 객체로 변환됩니다. 속성 관리자의 [인스턴스 이름] 필드에 동영상 클립 심볼의 인스턴스 이름을 지정하면 ActionScript에 심볼을 사용할 수 있습니다. SWF 파일이 만들어지면 스테이지에 MovieClip 인스턴스를 만드는 코드가 생성되고 인스턴스 이름을 사용하여 변수가 선언됩니다. 이름이 지정된 동영상 클립 내에 이름이 지정된 다른 동영상 클립이 중첩되어 있는 경우 그러한 자식 동영상 클립은 부모 동영상 클립의 속성으로 처리되므로 도트 구문을 사용하여 자식 동영상 클립에 액세스할 수 있습니다. 예를 들어, 인스턴스 이름이 childClip인 동영상 클립이 인스턴스 이름이 parentClip인 다른 클립 내에 중첩되어 있는 경우, 다음 코드를 호출하여 자식 클립의 타임라인 애니메이션이 재생되도록 할 수 있습니다.

```
parentClip.childClip.play();
```

참고: : Flash 제작 도구에서 스테이지에 있는 자식 인스턴스는 코드 실행 시점에서 생성되지 않기 때문에 부모 인스턴스의 생성자 내에서 코드를 통해 액세스할 수 없습니다. 자식에 액세스하기 전에 부모는 대신 코드를 통해 자식 인스턴스를 생성하거나 자식이 Event.ADDED_TO_STAGE 이벤트를 전달하기 위해 수신하는 콜백 함수에 대한 액세스를 지연시켜야 합니다.

ActionScript 2.0 MovieClip 클래스의 이전 메서드와 속성 중 일부는 그대로이지만 변경된 것도 있습니다. 밑줄로 시작하는 속성 이름은 모두 변경되었습니다. 예를 들어, _width 및 _height 속성은 이제 width 및 height로 액세스되고 _xscale 및 _yscale은 이제 scaleX 및 scaleY로 액세스됩니다. MovieClip 클래스의 전체 속성 및 메서드 목록은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

동영상 클립 재생 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash에서는 타임라인 메타포를 사용하여 애니메이션이나 상태 변경을 나타냅니다. 타임라인을 사용하는 시각적 요소는 MovieClip 객체이거나 MovieClip 클래스에서 확장해야 합니다. ActionScript 명령으로 동영상 클립을 중지하거나 재생하거나 타임라인의 다른 지점으로 이동할 수는 있지만 ActionScript를 사용하여 동적으로 타임라인을 만들거나 특정 프레임에 내용을 추가할 수는 없습니다. 이러한 작업은 Flash 제작 도구에서만 가능합니다.

MovieClip을 재생하는 경우 SWF 파일의 프레임 속도에 의해 지정된 속도로 타임라인을 진행합니다. 또는 ActionScript에서 Stage.frameRate 속성을 설정하여 이 설정을 덮어쓸 수 있습니다.

동영상 클립 재생 및 재생 중지

Flash Player 9 이상, Adobe AIR 1.0 이상

play() 및 stop() 메서드를 사용하면 타임라인 전체에서 동영상 클립의 기본 컨트롤을 사용할 수 있습니다. 예를 들어, 자전거가 화면을 가로질러 움직이는 애니메이션이 포함된 동영상 클립 심볼이 스테이지에 있고 이 클립의 인스턴스 이름이 bicycle로 지정되었다고 가정해 봅니다. 이 경우 기본 타임라인의 키프레임에 다음 코드가 첨부되면,

```
bicycle.stop();
```

자전거는 움직이지 않게 됩니다. 다시 말해 애니메이션이 재생되지 않습니다. 자전거의 움직임은 다른 사용자 조작을 통해서도 시작할 수 있습니다. 예를 들어, startButton이라는 버튼이 있는 경우, 기본 타임라인의 키프레임에 다음 코드를 적용하면 버튼을 클릭해야 애니메이션이 재생되도록 만듭니다.

```
// 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);
```

빨리 감기 및 되감기

Flash Player 9 이상, Adobe AIR 1.0 이상

play() 및 stop() 메서드를 사용하는 방법 외에도 동영상 클립에서 재생을 제어하는 다른 방법이 있습니다. nextFrame() 및 prevFrame() 메서드를 사용하여 재생 헤드를 타임라인을 따라 수동으로 빨리 감거나 되감을 수도 있습니다. 이러한 메서드 중 하나를 호출하면 재생이 중지되고 재생 헤드가 앞으로 또는 뒤로 각각 이동합니다.

play() 메서드를 사용하는 것은 동영상 클립 객체의 enterFrame 이벤트가 트리거될 때마다 nextFrame()을 호출하는 것과 비슷합니다. 이러한 맥락에서, 다음과 같이 enterFrame 이벤트에 대한 이벤트 리스너를 만들고 bicycle이 해당 리스너 함수 내의 이전 프레임으로 이동하도록 지시하는 방법으로 bicycle 동영상 클립의 되감기를 실행할 수 있습니다.

```
// 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);
```

일반적으로 재생할 때 동영상 클립에 둘 이상의 프레임이 있는 경우 이 동영상 클립을 재생하면 무한정 반복됩니다. 즉, 마지막 프레임을 지나면 프레임 1로 돌아옵니다. prevFrame() 또는 nextFrame()을 사용하면 이러한 비헤이비어가 자동으로 발생하지 않습니다. 다시 말해서, 재생 헤드가 프레임 1에 있을 때 prevFrame()을 호출해도 재생 헤드가 마지막 프레임으로 이동하지 않습니다. 위 예제에서 if 조건은 재생 헤드가 첫 번째 프레임으로 되감기를 진행했는지 확인하고, 재생 헤드를 마지막 프레임 앞에 오도록 설정하여 되감기를 실행하는 동영상 클립이 효과적으로 반복되도록 만듭니다.

다른 프레임으로 이동 및 프레임 레이블 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

새 프레임에 동영상 클립을 보내는 작업은 간단합니다. gotoAndPlay() 또는 gotoAndStop()을 호출하면 동영상 클립이 매개 변수로 지정된 프레임 번호로 이동합니다. 또는 프레임 레이블 이름과 일치하는 문자열을 전달할 수 있습니다. 타임라인의 모든 프레임에 레이블을 지정할 수 있습니다. 그렇게 하려면 타임라인에서 프레임을 선택하고 속성 관리자의 [프레임 레이블] 필드에 이름을 입력합니다.

특히 복잡한 동영상 클립을 만드는 경우에 번호 대신 프레임 레이블을 사용하는 것이 좋습니다. 애니메이션의 프레임, 레이어 및 트윈 개수가 많아지면 중요한 프레임에 대해 동영상 클립 비헤이비어의 변화를 나타내는 설명(예: "off", "walking" 또는 "running")으로 레이블을 지정하는 것이 좋습니다. 레이블이 지정된 프레임으로 가는 ActionScript 호출은 특정 프레임 번호가

아닌 하나의 참조, 즉 레이블을 가리키므로 이렇게 하면 코드의 가독성도 향상되고 유연성도 증가합니다. 나중에 애니메이션의 특정 선분을 다른 프레임으로 이동할 경우 새 위치에서 해당 프레임의 레이블을 동일하게 유지하는 한 ActionScript 코드를 변경할 필요가 없습니다.

ActionScript 3.0은 프레임 레이블을 코드로 표시할 수 있도록 FrameLabel 클래스를 제공합니다. 이 클래스의 각 인스턴스는 단일 프레임 레이블을 나타내며, name 속성(속성 관리자에 지정된 프레임 레이블 이름을 나타냄) 및 frame 속성(레이블이 배치될 타임라인 프레임의 프레임 번호를 나타냄)을 가지고 있습니다.

동영상 클립 인스턴스와 연관된 FrameLabel 인스턴스에 액세스할 수 있도록 MovieClip 클래스에는 FrameLabel 객체를 직접 반환하는 두 개의 속성이 포함되어 있습니다. currentLabels 속성은 동영상 클립 전체 타임라인의 모든 FrameLabel 객체로 이루어진 배열을 반환합니다. currentLabel 속성은 최근에 타임라인에 나타난 FrameLabel 객체의 이름이 포함된 문자열을 반환합니다.

로봇이라는 동영상 클립을 만들었으며 애니메이션의 여러 가지 상태에 레이블을 지정했다고 가정할 경우 다음 코드와 같이 로봇의 현재 상태에 액세스할 수 있는 currentLabel 속성을 확인하는 조건을 설정할 수 있습니다.

```
if (robot.currentLabel == "walking")
{
    // do something
}
```

Flash Player 11.3 및 AIR 3.3에서는 FrameLabel 클래스에frameLabel 이벤트가 추가되었습니다. 프레임 레이블을 나타내는 FrameLabel 인스턴스에 이벤트 핸들러를 지정할 수 있습니다. 재생 헤드가 프레임에 들어오면 이벤트가 전달됩니다.

다음 예에서는 MovieClip의 프레임 레이블 배열에서 두 번째 프레임 레이블의FrameLabel 인스턴스를 만듭니다. 그런 다음 frameLabel 이벤트에 대해 이벤트 핸들러를 등록합니다.

```
var myFrameLabel:FrameLabel = robot.currentLabels[1];
myFrameLabel.addEventListener(Event.FRAME_LABEL, onFrameLabel);

function onFrameLabel(e:Event):void {
    //do something
}
```

장면을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash 제작 환경에서는 SWF 파일이 진행되는 타임라인을 연속으로 보여 주는 장면을 사용할 수 있습니다. gotoAndPlay() 또는 gotoAndStop() 메서드의 두 번째 매개 변수를 사용하면 재생 헤드를 보낼 수 있는 장면을 지정할 수 있습니다. 모든 FLA 파일은 첫 장면으로만 시작하지만 새 장면을 만들 수도 있습니다.

장면에는 단점이 많으므로 장면을 사용하는 것이 항상 최선은 아닙니다. 여러 장면이 포함된 Flash 문서는 유지 관리가 어려울 수 있습니다. 특히 제작자가 여러 명인 경우 더욱 그렇습니다. 제작 프로세스 동안 모든 장면을 하나의 타임라인에 병합하므로 장면이 여러 개인 경우 대역폭이 불충분할 수도 있습니다. 따라서 장면이 재생된 적이 없는 경우에도 모든 장면이 점진적으로 다운로드될 수 있습니다. 따라서 여러 타임라인을 기반으로 하는 장편 애니메이션을 구성하는 경우를 제외하고는 여러 장면을 사용하지 않는 것이 좋습니다.

MovieClip 클래스의 scenes 속성은 SWF 파일에 있는 모든 장면을 나타내는 Scene 객체의 배열을 반환합니다. currentScene 속성은 현재 재생 중인 장면을 나타내는 Scene 객체를 반환합니다.

Scene 클래스에는 장면 정보를 제공하는 여러 속성이 있습니다. labels 속성은 해당 장면 내의 프레임 레이블을 나타내는 FrameLabel 객체의 배열을 반환합니다. name 속성은 장면의 이름을 문자열로 반환합니다. numFrames 속성은 장면의 전체 프레임 수를 나타내는 int를 반환합니다.

ActionScript를 사용하여 MovieClip 객체 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash에서 화면에 내용을 추가하는 한 가지 방법은 라이브러리의 에셋을 스테이지로 드래그하는 것입니다. 그러나 이외에 다른 작업 과정도 있습니다. 복잡한 프로젝트의 경우 일반적으로 숙련된 개발자는 프로그래밍 방식으로 동영상 클립을 만드는 경우가 많습니다. 이러한 방식은 코드를 다시 사용하기가 쉽고, 컴파일 시간 속도가 빠르며, ActionScript에서만 가능한 정교한 수정 작업을 수행할 수 있다는 장점이 있습니다.

ActionScript 3.0의 표시 목록 API에서는 동적으로 MovieClip 객체를 만드는 프로세스가 간편해졌습니다. MovieClip 인스턴스를 표시 목록에 추가하지 않고도 직접 인스턴스화할 수 있으므로 제어 기능을 그대로 유지하면서 작업을 유연하고 간편하게 수행할 수 있습니다.

ActionScript 3.0에서 프로그래밍 방식으로 동영상 클립(또는 기타 표시 객체) 인스턴스를 만들면, 표시 객체 컨테이너에서 addChild() 또는 addChildAt() 메서드를 호출하여 만든 인스턴스를 표시 목록에 추가해야만 화면에 나타납니다. 따라서 동영상 클립을 만들어 속성을 설정하고, 메서드를 화면에 렌더링하기 전에 호출할 수 있습니다. 표시 목록을 사용한 작업에 대한 자세한 내용은 143페이지의 “표시 객체 컨테이너 작업”을 참조하십시오.

ActionScript의 라이브러리 심볼 내보내기

Flash Player 9 이상, Adobe AIR 1.0 이상

기본적으로 Flash 문서의 라이브러리에 있는 동영상 클립 심볼 인스턴스는 동적으로 만들 수 없습니다. 다시 말해서, ActionScript를 이용해서만 만들 수 있습니다. 이는 심볼을 ActionScript에서 사용하기 위해 내보낼 때마다 SWF 파일의 크기가 늘어날 뿐 아니라 일부 심볼은 스테이지용에서 사용할 수 있도록 만들어지지 않았기 때문입니다. 이러한 이유로 ActionScript에서 심볼을 사용하려면 해당 심볼을 ActionScript용으로 내보내도록 지정해야만 합니다.

심볼을 ActionScript에 사용할 수 있도록 내보내려면:

1. [라이브러리] 패널에서 심볼을 선택하고 [심볼 속성] 대화 상자를 엽니다.
2. 필요한 경우 [고급 설정]을 활성화합니다.
3. [링크] 섹션에서 [ActionScript에 내보내기] 체크 상자를 선택합니다.

   그러면 [클래스] 및 [기본 클래스] 필드가 활성화됩니다.

   기본적으로 [클래스] 필드에는 공백이 제거된 심볼 이름(예: 이름이 "Tree House"인 심볼의 경우 "TreeHouse")이 표시됩니다. 심볼이 비헤이비어에 사용자 정의 클래스를 사용하도록 지정하려면 이 필드에 패키지 이름을 포함한 전체 클래스 이름을 입력합니다. 비헤이비어를 추가하지 않고 ActionScript에 심볼의 인스턴스를 만들려는 경우 클래스 이름을 그대로 두면 됩니다.

   [기본 클래스] 필드의 기본값은 flash.display.MovieClip입니다. 심볼이 기타 사용자 정의 클래스의 기능을 확장하도록 만들려면, 해당 클래스가 Sprite 또는 MovieClip 클래스를 확장하는 경우 클래스의 이름을 지정할 수 있습니다.
4. [확인] 버튼을 눌러 변경 내용을 저장합니다.

   이때 Flash에서 지정된 클래스가 정의되어 있는 외부 ActionScript 파일이나 연결된 SWC 파일을 찾지 못하면(예를 들어, 심볼에 비헤이비어를 추가할 필요가 없는 경우) 다음과 같은 경고 메시지가 표시됩니다.

   이 클래스의 정의는 클래스 경로에서 찾을 수 없으므로 내보낼 때 SWF 파일에서 자동으로 생성됩니다.

   라이브러리 심볼에 MovieClip 클래스의 기능 외에 고유한 기능이 필요하지 않다면 이 경고를 무시해도 됩니다.

사용자가 심볼에 대한 클래스를 제공하지 않으면 Flash에서 다음과 같이 심볼에 대한 클래스를 만듭니다.

```
package
{
    import flash.display.MovieClip;

    public class ExampleMovieClip extends MovieClip
    {
        public function ExampleMovieClip()
        {
        }
    }
}
```

심볼에 기타 ActionScript 기능을 추가하려면 아래 코드 구조에 해당 속성과 메서드를 추가합니다. 예를 들어, 폭과 높이가 각각 50픽셀인 원이 포함된 동영상 클립 심볼이 있고, 이름이 Circle인 클래스를 사용하여 심볼을 ActionScript에 내보내도록 지정했다고 가정해 봅시다. 다음 코드가 Circle.as 파일에 배치되면 MovieClip 클래스를 확장하고 추가 메서드 getArea() 및 getCircumference()가 있는 심볼을 제공합니다.

```
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;
        }
    }
}
```

다음 코드가 Flash 문서 프레임 1의 키프레임에 배치되면 심볼의 인스턴스를 만들어 화면에 표시합니다.

```
var c:Circle = new Circle();
addChild(c);
trace(c.width);
trace(c.height);
trace(c.getArea());
trace(c.getCircumference());
```

이 코드는 개별 에셋을 스테이지로 드래그하는 대안으로 ActionScript 기반 인스턴스화를 보여 줍니다. 동영상 클립의 모든 속성과 Circle 클래스에서 정의한 사용자 정의 메서드가 있는 원이 만들어집니다. 이것은 매우 기본적인 예로서, 라이브러리 심볼은 클래스에서 원하는 수만큼 속성과 메서드를 지정할 수 있습니다.

ActionScript 기반 인스턴스화는 수동으로 정렬하기에는 지루한 대량의 인스턴스를 동적으로 만들 수 있으므로 강력합니다. 또한 각 인스턴스를 만들 때 해당 속성을 사용자 정의할 수 있으므로 유연합니다. 루프를 사용하여 여러 Circle 인스턴스를 동적으로 만들면 이러한 장점을 모두 이용할 수 있습니다. 앞서 설명한 Flash 문서 라이브러리의 Circle 심볼 및 클래스를 사용하여 프레임 1의 키프레임에 다음 코드를 배치합니다.

```
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);
}
```

이 예는 코드를 사용하여 심볼에 대한 여러 개의 인스턴스를 빨리 만들고 사용자 정의할 수 있는 방법을 보여 줍니다. 루프 내에 있는 현재 개수에 따라 각 인스턴스가 배치되고, transform 속성(Circle이 MovieClip 클래스를 확장하여 상속함)을 설정하여 각 인스턴스에 임의의 색이 지정됩니다.

외부 SWF파일 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서 SWF 파일은 Loader 클래스를 사용하여 로드됩니다. 외부 SWF 파일을 로드하려면 ActionScript에서는 다음 네 가지를 수행합니다.

1. 파일의 URL을 사용하여 새 URLRequest 객체를 만듭니다.
2. 새 Loader 객체를 만듭니다.
3. 이 Loader 객체의 load() 메서드를 호출하여 URLRequest 인스턴스를 매개 변수로 전달합니다.
4. 표시 객체 컨테이너(예: Flash 문서의 기본 타임라인)의 addChild() 메서드를 호출하여 Loader 인스턴스를 표시 목록에 추가합니다.

완성된 코드는 다음과 같습니다.

```
var request:URLRequest = new URLRequest("http://www.[yourdomain].com/externalSwf.swf");
var loader:Loader = new Loader()
loader.load(request);
addChild(loader);
```

위와 동일한 코드를 사용하여 JPEG, GIF, PNG 이미지 등의 외부 이미지 파일을 로드할 수 있습니다. 이때 SWF 파일 URL이 아니라 해당 이미지 파일의 URL을 지정합니다. SWF 파일은 이미지 파일과 달리 ActionScript를 포함할 수 있습니다. SWF 파일 로드 프로세스는 이미지 로드와 동일하지만, Flash Player 또는 AIR에서 SWF를 재생하고 있으며 ActionScript를 사용하여

외부 SWF 파일과 통신하려는 경우 외부 SWF 파일을 로드할 때 로드하는 SWF 파일과 로드되는 SWF 파일이 모두 동일한 보안 샌드박스에 있어야 합니다. 또한 외부 SWF 파일의 클래스 네임스페이스가 로드하는 SWF 파일의 클래스 네임스페이스와 동일한 경우 네임스페이스의 충돌을 방지하기 위해 로드되는 SWF 파일에 대한 새 응용 프로그램 도메인을 만들어야 할 수도 있습니다. 보안 및 응용 프로그램 도메인 고려 사항에 대한 자세한 내용은 132페이지의 "응용 프로그램 도메인 작업" 및 964페이지의 "내용 로드"를 참조하십시오.

외부 SWF 파일이 성공적으로 로드되면 Loader.content 속성을 통해 액세스할 수 있습니다. 외부 SWF 파일이 ActionScript 3.0용으로 제작된 경우 이 파일은 확장하는 클래스에 따라 동영상 클립이나 스프라이트 중 하나가 됩니다.

AIR for iOS에서 SWF 파일을 로드하는 동작은 다른 플랫폼과 다른 몇 가지 차이점이 있습니다. 자세한 내용은 181페이지의 "AIR for iOS의 SWF 파일 로드"를 참조하십시오.

이전 SWF 파일을 로드할 때 고려할 사항

Flash Player 9 이상, Adobe AIR 1.0 이상

이전 버전의 ActionScript를 사용하여 외부 SWF 파일을 제작한 경우 고려해야 할 중요한 제한 사항이 있습니다. AVM2(ActionScript Virtual Machine 2)에서 실행되는 ActionScript 3.0 SWF 파일과는 달리 ActionScript 1.0 또는 2.0용으로 제작된 SWF 파일은 AVM1(ActionScript Virtual Machine 1)에서 실행됩니다.

ActionScript 3.0 SWF 파일 로드와 비교해 볼 때 ActionScript 1.0 또는 2.0 SWF 파일을 ActionScript 3.0 SWF 파일로 로드하는 데는 중요한 차이점이 있습니다. Flash Player에서는 이전에 게시된 내용과의 호환성이 완벽하게 유지됩니다. Flash Player의 이전 버전에서 실행되는 모든 내용이 ActionScript 3.0을 지원하는 Flash Player 버전에서도 실행됩니다. 단, 다음과 같은 제한 사항이 적용됩니다.

- ActionScript 3.0 코드는 ActionScript 1.0 또는 2.0으로 작성된 SWF 파일을 로드할 수 있습니다. ActionScript 1.0 또는 2.0 SWF 파일이 로드되면 로드된 객체(Loader.content 속성)는 AVM1Movie 객체가 됩니다. AVM1Movie 인스턴스는 MovieClip 인스턴스와 동일하지 않습니다. 표시 객체이지만 동영상 클립과 달리 타임라인 관련 메서드나 속성이 없습니다. 부모 파일인 AVM2 SWF는 로드된 AVM1Movie 객체의 속성, 메서드 또는 객체에 대한 액세스 권한이 없습니다.
- ActionScript 1.0 또는 2.0으로 작성된 SWF 파일에서는 ActionScript 3.0으로 작성된 SWF 파일을 로드할 수 없습니다. 이는 Flash 8 또는 Flex Builder 1.5 이하 버전에서 작성된 SWF 파일에서 ActionScript 3.0 SWF 파일을 로드할 수 없음을 의미합니다.

  이 규칙에 대한 유일한 예외는 특정 ActionScript 2.0 SWF 파일에 의해 로드된 파일이 어떠한 수준에서도 전혀 없는 경우 해당 ActionScript 2.0 SWF 파일을 ActionScript 3.0 SWF 파일로 대체할 수 있다는 것입니다. ActionScript 2.0 SWF 파일에서는 loadMovieNum()의 level 매개 변수에 값 0을 전달하여 호출함으로써 이를 수행할 수 있습니다.
- 일반적으로 ActionScript 3.0으로 작성된 SWF 파일과 함께 작동할 ActionScript 1.0 또는 2.0으로 작성된 SWF 파일은 마이그레이션해야 합니다. 예를 들어 ActionScript 2.0을 사용하여 미디어 플레이어를 만들었다고 가정해 봅니다. 미디어 플레이어에서 ActionScript 2.0을 사용하여 만들어진 다양한 내용을 로드합니다. ActionScript 3.0에서 만든 새로운 내용은 미디어 플레이어에서 로드할 수 없습니다. 비디오 플레이어를 ActionScript 3.0으로 마이그레이션해야 합니다.

  그러나 ActionScript 3.0으로 미디어 플레이어를 만드는 경우 해당 미디어 플레이어에서 ActionScript 2.0으로 만든 내용을 간단히 로드할 수 있습니다.

다음 표에는 새로운 내용 및 실행 코드의 로드와 관련된 이전 버전의 Flash Player에 대한 제한 사항 및 다른 버전의 ActionScript로 작성된 SWF 파일 간의 크로스 스크립팅에 대한 제한 사항이 요약되어 있습니다.

지원되는 기능	Flash Player 7	Flash Player 8	Flash Player 9 및 10
로드 가능한 SWF의 Flash Player 버전	7 이하	8 이하	9(또는 10) 이하
포함된 AVM	AVM1	AVM1	AVM1 및 AVM2
실행할 SWF의 ActionScript 버전	1.0 및 2.0	1.0 및 2.0	1.0, 2.0 및 3.0

다음 표에서 "지원되는 기능"은 Flash Player 9 이상에서 실행되는 내용을 나타냅니다. Flash Player 8 이하에서 실행되는 내용은 ActionScript 1.0 및 2.0만 로드, 표시, 실행 및 크로스 스크립팅할 수 있습니다.

지원되는 기능	ActionScript 1.0 및 2.0에서 만든 내용	ActionScript 3.0에서 만든 내용
로드 및 코드 실행 가능한 내용의 작성 버전	ActionScript 1.0 및 2.0만	ActionScript 1.0, 2.0 및 ActionScript 3.0
크로스 스크립팅 가능한 내용의 작성 버전	ActionScript 1.0 및 2.0만(LocalConnection을 사용하는 ActionScript 3.0)	LocalConnection을 사용하는 ActionScript 1.0 및 2.0 ActionScript 3.0

동영상 클립 예제: RuntimeAssetsExplorer

Flash Player 9 이상, Adobe AIR 1.0 이상

[ActionScript에 내보내기] 기능은 둘 이상의 프로젝트에 유용하게 사용할 수 있는 라이브러리의 경우 특히 장점이 많습니다. Flash Player 또는 AIR에서 SWF 파일을 실행하는 경우 ActionScript로 내보낸 심볼은 심볼을 로드하는 SWF와 동일한 보안 샌드박스 내의 모든 SWF 파일에 사용할 수 있습니다. 이런 방식으로 하나의 Flash 문서는 그래픽 에셋을 보유하기 위한 용도로만 지정된 SWF 파일을 생성할 수 있습니다. 이 기술은 "래퍼" SWF 파일을 만든 다음 런타임 시 그래픽 에셋 SWF 파일을 로드하는 개발자와 디자이너가 함께 시각적 에셋을 사용하는 큰 프로젝트에 특히 유용합니다. 이 메서드를 사용하여 일련의 버전 관리된 파일을 유지 관리할 수 있습니다. 이러한 파일에서는 그래픽 에셋이 프로그래밍 개발 과정의 영향을 받지 않습니다.

RuntimeAssetsExplorer 응용 프로그램은 RuntimeAsset의 하위 클래스인 모든 SWF 파일을 로드하며 해당 SWF 파일의 사용 가능한 에셋을 찾아볼 수 있습니다. 다음은 이러한 경우에 대한 예제입니다.

- Loader.load()를 사용하여 외부 SWF 파일 로드
- ActionScript에 내보낸 라이브러리 심볼을 동적으로 만들기
- MovieClip 재생의 ActionScript 컨트롤

시작하기 전에 Flash Player에서 실행할 각 SWF 파일을 동일한 보안 샌드박스에 반드시 배치하십시오. 자세한 내용은 950페이지의 “보안 샌드박스”를 참조하십시오.

이 샘플의 응용 프로그램 파일을 얻으려면 Flash Professional 샘플을 다운로드하십시오. RuntimeAssetsExplorer 응용 프로그램 파일은 Samples/RuntimeAssetsExplorer 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
RuntimeAssetsExample.mxml 또는 RuntimeAssetsExample.fla	Flex(MXML) 또는 Flash(FLA)용 응용 프로그램의 사용자 인터페이스입니다.
RuntimeAssetsExample.as	Flash(FLA) 응용 프로그램용 문서 클래스입니다.
GeometricAssets.as	RuntimeAsset 인터페이스를 구현하는 클래스에 대한 예제입니다.
GeometricAssets.fla	ActionScript용으로 내보내진 심볼을 포함한 GeometricAssets 클래스(FLA의 문서 클래스)에 연결된 FLA 파일입니다.

파일	설명
com/example/programmingas3/runtimeassetexplorer/RuntimeLibrary.as	탐색기 컨테이너로 로드될 모든 런타임 에셋 SWF 파일을 반환할 필수 메서드를 정의하는 인터페이스입니다.
com/example/programmingas3/runtimeassetexplorer/AnimatingBox.as	회전 상자 모양에 있는 라이브러리 심볼의 클래스입니다.
com/example/programmingas3/runtimeassetexplorer/AnimatingStar.as	회전 별 모양에 있는 라이브러리 심볼의 클래스입니다.

런타임 라이브러리 인터페이스 구축

Flash Player 9 이상, Adobe AIR 1.0 이상

탐색기가 SWF 라이브러리와 적절히 상호 작용하려면 런타임 에셋 라이브러리의 구조를 정형화해야 합니다. 이러한 구조는 예상 구조를 보여 주는 메서드의 청사진이라는 점에서 클래스와 유사하지만 클래스와는 다르게 메서드 본문이 포함되지 않은 인터페이스를 만들어 정형화할 수 있습니다. 이 인터페이스는 런타임 라이브러리와 탐색기 모두에 서로 통신할 수 있는 방법을 제공합니다. 브라우저에 로드되는 런타임 에셋의 각 SWF는 이 인터페이스를 구현합니다. 인터페이스 및 그 활용에 대한 자세한 정보는 **ActionScript 3.0** 학습에서 인터페이스를 참조하십시오.

RuntimeLibrary 인터페이스는 매우 간단합니다. 심볼을 내보내고 런타임 라이브러리에서 사용할 수 있는 클래스 경로 배열을 탐색기에 제공할 수 있는 함수만 있으면 됩니다. 이 작업을 수행할 수 있도록 인터페이스에 getAssets()라는 메서드 하나가 있습니다.

```
package com.example.programmingas3.runtimeassetexplorer
{
    public interface RuntimeLibrary
    {
        function getAssets():Array;
    }
}
```

에셋 라이브러리 SWF 파일 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

RuntimeLibrary 인터페이스를 정의하여 다른 SWF 파일로 로드할 수 있는 여러 에셋 라이브러리 SWF 파일을 만들 수 있습니다. 에셋의 개별 SWF 라이브러리를 만들려면 다음 네 가지 작업을 수행해야 합니다.

- 에셋 라이브러리 SWF 파일의 클래스 만들기
- 라이브러리에 포함된 개별 에셋의 클래스 만들기
- 실제 그래픽 에셋 만들기
- 그래픽 요소를 클래스와 연결하고 라이브러리 SWF 제작

RuntimeLibrary 인터페이스 구현을 위한 클래스 만들기

다음으로 RuntimeLibrary 인터페이스를 구현할 GeometricAssets 클래스를 만듭니다. 이 클래스가 FLA의 문서 클래스가 되며 이 클래스의 코드는 RuntimeLibrary 인터페이스와 매우 유사합니다. 단, 클래스 정의에는 getAssets() 메서드에 메서드 본문이 있다는 점이 다릅니다.

```
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" ];
        }
    }
}
```

두 번째 런타임 라이브러리를 만들려는 경우 자체 getAssets() 구현을 제공하는 다른 클래스(예: AnimationAssets)를 기초로 다른 FLA를 만들 수 있습니다.

각 MovieClip 에셋의 클래스 만들기

이 예제에서는 사용자 정의 에셋에 기능을 추가하지 않고 MovieClip 클래스만 확장할 것입니다. AnimatingStar에 대한 다음 코드는 AnimatingBox에 대한 코드와 비슷합니다.

```
package com.example.programmingas3.runtimeassetexplorer
{
    import flash.display.MovieClip;

    public class AnimatingStar extends MovieClip
    {
        public function AnimatingStar() {
        }
    }
}
```

라이브러리 제작

이제 새 FLA를 만들고 속성 관리자의 [문서 클래스] 필드에 GeometricAssets를 입력하여 MovieClip 기반 에셋을 새로운 클래스에 연결합니다. 이에 대한 예를 들기 위해 360개가 넘는 프레임에 하나의 시계 방향 회전을 만들기 위해 타임라인 트윈을 사용하는 매우 기본적인 모양 두 개를 만들 것입니다. animatingBox 및 animatingStar 심볼은 모두 [ActionScript에 내보내기]로 설정되고 [클래스] 필드는 getAssets() 구현에 지정된 각각의 클래스 경로로 설정됩니다. 표준 MovieClip 메서드를 하위 클래스로 사용하려고 하면 flash.display.MovieClip의 기본 클래스가 그대로 유지됩니다.

심볼의 내보내기 설정을 지정한 후 FLA를 제작합니다. 이제 첫 번째 런타임 라이브러리가 있습니다. 이 SWF 파일은 다른 AVM2 SWF 파일에 로드될 수 있으며 새 SWF 파일에 대해 AnimatingBox 및 AnimatingStar 심볼을 사용할 수 있습니다.

다른 SWF 파일로 라이브러리 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

마지막으로 설명할 기능은 에셋 탐색기의 사용자 인터페이스입니다. 이 예제에서 런타임 라이브러리 경로는 ASSETS_PATH라는 변수로 하드 코딩되어 있습니다. FileReference 클래스를 사용할 수도 있습니다. 예를 들어, 하드 드라이브에서 특정 SWF 파일을 탐색하는 인터페이스를 만들려는 경우에 FileReference 클래스를 사용합니다.

런타임 라이브러리가 성공적으로 로드되면 Flash Player는 runtimeAssetsLoadComplete() 메서드를 호출합니다.

```
private function runtimeAssetsLoadComplete(event:Event):void
{
    var rl:* = event.target.content;
    var assetList:Array = rl.getAssets();
    populateDropdown(assetList);
    stage.frameRate = 60;
}
```

이 메서드에서 변수 rl은 로드된 SWF 파일을 나타냅니다. 코드는 로드된 SWF 파일의 getAssets() 메서드를 호출하고 사용 가능한 에셋 목록을 가져온 다음, 이 에셋을 사용하여 populateDropDown() 메서드 호출로 사용 가능한 에셋 목록을 ComboBox 구성 요소에 채웁니다. 그리고 해당 메서드는 각 에셋의 전체 클래스 경로를 저장합니다. 사용자 인터페이스에서 [추가] 버튼을 클릭하면 addAsset() 메서드가 트리거됩니다.

```
private function addAsset():void
{
    var className:String = assetNameCbo.selectedItem.data;
    var AssetClass:Class = getDefinitionByName(className) as Class;
    var mc:MovieClip = new AssetClass();
    ...
}
```

이 메서드는 ComboBox에 현재 선택되어 있는 에셋의 클래스 경로를 가져오고(assetNameCbo.selectedItem.data), flash.utils 패키지의 getDefinitionByName() 함수를 사용하여 에셋의 클래스에 대한 실제 참조를 가져와 에셋의 새 인스턴스를 만듭니다.

17장: 모션 트윈을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

173페이지의 “객체 애니메이션”에서는 ActionScript에서 스크립트 애니메이션을 구현하는 방법을 설명합니다.

이 장에서는 애니메이션을 만들기 위한 모션 트윈이라는 다른 기술을 설명합니다. Adobe® Flash® Professional에서 이 기술을 사용하면 문서에서 모션을 대화식으로 설정하여 움직임을 만들 수 있습니다. 그런 다음 런타임에 동적 ActionScript 기반 애니메이션에서 해당 모션을 사용할 수 있습니다.

Flash Professional에서는 모션 트윈을 구현하고 이를 복사하거나 다시 사용할 수 있게 해 주는 ActionScript가 자동으로 생성됩니다.

모션 트윈을 만들려면 Flash Professional의 라이센스가 있어야 합니다.

기타 도움말 항목

fl.motion 패키지

모션 트윈의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

모션 트윈을 사용하면 애니메이션을 손쉽게 만들 수 있습니다.

모션 트윈은 위치 또는 회전과 같은 표시 객체 속성을 프레임 단위로 수정합니다. 모션 트윈을 사용하면 다양한 필터와 기타 속성을 적용하여 표시 객체가 이동하는 동안 해당 모양을 변경할 수도 있습니다. Flash Professional에서 대화식으로 모션 트윈을 만들면 모션 트윈의 ActionScript가 생성됩니다. Flash 내에서 [ActionScript 3.0으로 모션 복사] 명령을 사용하여 모션 트윈을 만든 ActionScript를 복사합니다. 그런 다음 해당 ActionScript를 다시 사용하여 런타임에 사용자의 동적 애니메이션에 움직임을 만들 수 있습니다.

모션 트윈 만들기에 대한 자세한 내용은 **Flash Professional** 사용의 "모션 트윈" 단원을 참조하십시오.

중요한 개념 및 용어

이 기능과 관련한 중요한 용어는 다음과 같습니다.

모션 트윈 표시 객체에 대해 상태 및 시간이 각각 다른 중간 프레임을 생성하는 구문으로서, 첫 번째 상태가 자연스럽게 두 번째 상태로 진행되는 모양을 만듭니다. 모션 트윈은 스테이지에서 표시 객체를 이동할 뿐 아니라 표시 객체를 크기 조절하거나 회전하거나 페이드 인/아웃 효과를 적용하거나 시간에 따라 색상이 변경되도록 하는 데 사용됩니다.

모션 트윈 스크립트 복사 Flash

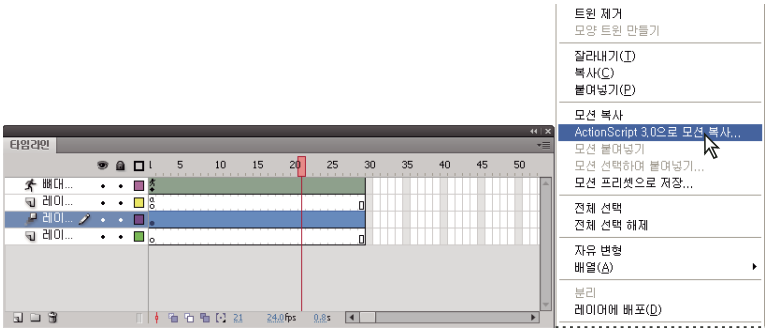
Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

트윈은 타임라인의 서로 다른 두 프레임에 각기 다른 상태로 표시 객체를 표시하는 중간 프레임을 생성합니다. 또한 첫 번째 프레임의 이미지가 두 번째 프레임의 이미지로 자연스럽게 진행되는 모양을 만듭니다. 모션 트윈에서 모양이 변경되면 일반적으로 표시 객체의 위치가 변경되어 움직임이 만들어집니다. 모션 트윈은 표시 객체의 위치를 변경할 뿐 아니라 표시 객체를 회전하거나 기울이거나 크기를 조절하거나 표시 객체에 필터를 적용할 수도 있습니다.

Flash에서 모션 트윈을 만들려면 타임라인에서 키프레임 간에 표시 객체를 이동합니다. 그러면 트윈을 설명하는 ActionScript 코드가 자동으로 생성되며 이 코드를 복사하여 파일에 저장할 수 있습니다. 모션 트윈 만들기에 대한 자세한 내용은 **Flash Professional** 사용의 "모션 트윈" 단원을 참조하십시오.

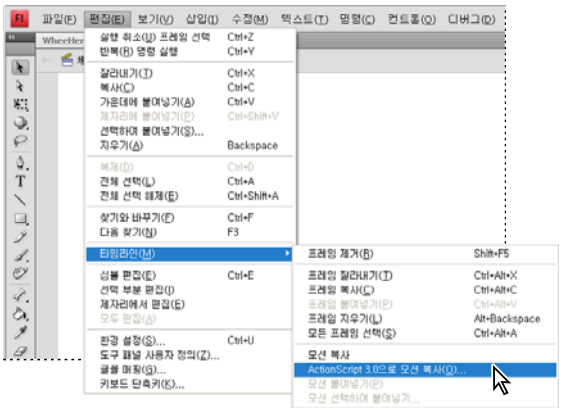
Flash에서는 두 가지 방법으로 [ActionScript 3.0으로 모션 복사] 명령에 액세스할 수 있습니다. 첫 번째 방법은 스테이지의 트윈 컨텍스트 메뉴를 사용하는 것입니다.

1. 스테이지에서 모션 트윈을 선택합니다.
2. 마우스 오른쪽 버튼을 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Macintosh)합니다.
3. [ActionScript 3.0으로 모션 복사...]를 선택합니다.



두 번째 방법은 Flash의 [편집] 메뉴에서 직접 명령을 선택하는 것입니다.

1. 스테이지에서 모션 트윈을 선택합니다.
2. [편집] > [타임라인] > [ActionScript 3.0으로 모션 복사]를 선택합니다.



스크립트를 복사한 후 이를 파일에 붙여넣고 저장합니다.

모션 트윈을 만들고 스크립트를 복사 및 저장한 후에는 사용자의 동적 ActionScript 기반 애니메이션에서 이를 그대로 다시 사용하거나 수정할 수 있습니다.

모션 트윈 스크립트 통합

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

Flash에서 복사하는 ActionScript 코드의 머리글에는 모션 트윈을 지원하는 데 필요한 모든 모듈의 목록이 나와 있습니다.

모션 트윈 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

주요 모션 트윈 클래스는 fl.motion 패키지의 AnimatorFactory, MotionBase 및 Motion 클래스입니다. 모션 트윈이 조작하는 속성에 따라 추가 클래스가 필요할 수도 있습니다. 예를 들어 모션 트윈이 표시 객체를 변형하거나 회전하는 경우에는 적절한 flash.geom 클래스를 가져오고, 필터를 적용하는 경우에는 flash.filter 클래스를 가져옵니다. ActionScript에서 모션 트윈은 Motion 클래스의 인스턴스입니다. Motion 클래스는 시각적 객체에 적용할 수 있는 키프레임 애니메이션 시퀀스를 저장합니다. 애니메이션 데이터에는 위치, 크기, 회전, 기울이기, 색상, 필터 및 여유가 포함됩니다.

다음 ActionScript는 인스턴스 이름이 Symbol1_2인 표시 객체에 애니메이션 효과를 적용하기 위해 Flash에서 만든 모션 트윈에서 복사한 것입니다. 이 ActionScript는 __motion_Symbol1_2라는 MotionBase 객체의 변수를 선언합니다. MotionBase 클래스는 Motion 클래스의 부모 클래스입니다.

```
var __motion_Symbol1_2:MotionBase;
```

그러면 스크립트에서는 Motion 객체를 만듭니다.

```
__motion_Symbol1_2 = new Motion();
```

Motion 객체 이름

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

앞의 경우에서 Flash는 Motion 객체에 대해 __motion_Symbol1_2라는 이름을 자동으로 생성하고 __motion_이라는 접두어를 표시 객체 이름에 연결합니다. 따라서 자동으로 생성되는 이름은 Flash에서 모션 트윈의 대상 객체에 대한 인스턴스 이름을 기초로 합니다. Motion 객체의 duration 속성은 모션 트윈의 총 프레임 수를 나타냅니다.

```
__motion_Symbol1_2.duration = 200;
```

기본적으로 Flash에서는 인스턴스 이름이 아직 없는 경우 복사하는 모션 트윈의 표시 객체 인스턴스 이름이 자동으로 지정됩니다.

Flash에서 만든 ActionScript를 사용자의 애니메이션에 다시 사용할 경우 Flash에서 자동으로 생성된 트윈 이름을 유지하거나 다른 이름으로 대체할 수 있습니다. 트윈 이름을 변경하는 경우에는 스크립트 전체에서 해당 이름을 변경해야 합니다.

또는 Flash에서 선택한 이름을 모션 트윈의 대상 객체에 할당할 수 있습니다. 그런 다음 모션 트윈을 만들고 스크립트를 복사합니다. 어떠한 이름 지정 방법을 사용하든 ActionScript 코드에서 각 Motion 객체의 이름은 고유해야 합니다.

애니메이션 설명

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

MotionBase 클래스의 addPropertyArray() 메서드는 값 배열을 추가하여 트위닝된 모든 속성을 설명합니다.

이 배열에는 모션 트윈의 모든 키프레임에 대한 배열 항목이 들어 있을 수 있습니다. 이러한 배열 중 일부에는 모션 트윈의 총 키프레임 수보다 적은 항목이 들어 있는 경우도 있습니다. 이와 같은 경우는 나머지 프레임에 대해 배열의 마지막 값이 변경되지 않을 때 발생합니다.

배열 인수의 길이가 Motion 객체의 duration 속성보다 긴 경우 addPropertyArray()는 그에 따라 duration 속성의 값을 조정합니다. 그러나 이전에 추가된 속성에 대해서는 키프레임을 추가하지 않습니다. 새로 추가된 키프레임은 애니메이션의 추가 프레임에서 유지됩니다.

Motion 객체의 x 및 y 속성은 애니메이션이 실행될 때 트위닝된 객체의 변경 위치를 나타냅니다. 이러한 좌표는 표시 객체의 위치가 변경될 때 모든 키프레임에서 변경될 가능성이 가장 큰 값입니다. addPropertyArray() 메서드를 사용하여 다른 모션 속성을 추가할 수 있습니다. 예를 들어 트위닝된 객체의 크기가 조절된 경우 scaleX 및 scaleY 값을 추가합니다. 트위닝된 객체가 기울어진 경우에는 skewX 및 skewY 값을 추가하고, 회전된 경우에는 rotationConcat 속성을 추가합니다.

addPropertyArray() 메서드를 사용하여 다음과 같은 트윈 속성을 정의합니다.

x	부모의 좌표 영역에서 객체의 변형점에 대한 가로 위치입니다.
y	부모의 좌표 영역에서 객체의 변형점에 대한 세로 위치입니다.
z	부모의 좌표 영역에서 객체의 변형점에 대한 깊이(z축) 위치입니다.
scaleX	변형점에서 적용된 객체의 수평 비율(백분율)입니다.
scaleY	변형점에서 적용된 객체의 수직 비율(백분율)입니다.
skewX	변형점에서 적용된 객체의 수평으로 기울이기 각도(도)입니다.
skewY	변형점에서 적용된 객체의 수직으로 기울이기 각도(도)입니다.
rotationX	객체의 원래 방향에서 x축을 기준으로 한 회전입니다.
rotationY	객체의 원래 방향에서 y축을 기준으로 한 회전입니다.
rotationConcat	변형점에서 이전에 적용했던 방향을 기준으로 z축을 따라 모션의 객체를 회전하는 값입니다.
useRotationConcat	이 속성을 설정한 경우 addPropertyArray()에서 모션 데이터를 제공하면 대상 객체가 회전합니다.
blendMode	객체의 색상과 그 아래의 그래픽을 혼합하는 방법을 지정하는 BlendMode 클래스 값입니다.
matrix3D	키프레임에 대해 존재하는 경우 matrix3D 속성으로서, 3D 트윈에 사용되며, 이 속성이 사용된 경우 이전의 모든 변형 속성은 무시됩니다.
rotationZ	객체가 3D 부모 컨테이너를 기준으로 원래 방향으로부터 z축 회전한 각도로서, rotationConcat 대신 3D 트윈에 사용됩니다.

자동으로 생성된 스크립트에 추가되는 속성은 Flash에서 모션 트윈에 할당된 속성에 따라 달라집니다. 사용자 고유의 스크립트 버전을 사용자 정의할 경우 이러한 속성 중 일부를 추가, 제거 또는 수정할 수 있습니다.

다음 코드에서는 __motion_Wheel이라는 모션 트윈의 속성에 값을 지정합니다. 이 경우 트위닝된 표시 객체는 위치가 변경되는 것이 아니라 모션 트윈의 29개 프레임 전체에서 적절하게 회전합니다. rotationConcat 배열에 지정된 여러 값은 회전을 정의합니다. 이 모션 트윈의 다른 속성 값은 변경되지 않습니다.

```
__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"]);
```

다음 예제에서는 Leaf_1이라는 표시 객체가 스테이지에서 이동합니다. x 및 y 속성 배열에는 애니메이션의 100개 프레임 각각에 대한 서로 다른 값이 들어 있습니다. 또한 객체는 스테이지에서 이동할 때 z축을 기준으로 회전합니다. rotationZ 속성 배열의 여러 항목이 이 회전을 결정합니다.

```
__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"]);
```

필터 추가

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

모션 트윈의 대상 객체에 필터가 들어 있는 경우 이러한 필터는 Motion 클래스의 initFilters() 및 addFilterPropertyArray() 메서드를 사용하여 추가됩니다.

필터 배열 초기화

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

initFilters() 메서드는 필터를 초기화합니다. 첫 번째 인수는 표시 객체에 적용된 모든 필터의 정규화된 클래스 이름이 들어 있는 배열입니다. 이 필터 이름 배열은 Flash에서 모션 트윈에 대한 필터 목록을 사용하여 생성됩니다. 스크립트 복사본에서 flash.filters 패키지의 필터를 제거하거나 이 배열에 추가할 수 있습니다. 다음 호출에서는 대상 표시 객체에 대한 필터 목록을 초기화합니다. 또한 DropShadowFilter, GlowFilter 및 BevelFilter를 적용하고 목록을 Motion 객체의 각 키프레임에 복사합니다.

```
__motion_Box.initFilters(["flash.filters.DropShadowFilter", "flash.filters.GlowFilter",
"flash.filters.BevelFilter"], [0, 0, 0]);
```

필터 추가

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

addFilterPropertyArray() 메서드는 다음 인수로 초기화된 필터의 속성을 나타냅니다.

1. 첫 번째 인수는 필터를 인덱스로 식별합니다. 인덱스는 initFilters()에 대한 이전 호출에서 전달된 필터 클래스 이름 배열에서 해당 필터 이름이 있는 위치를 나타냅니다.
2. 두 번째 인수는 각 키프레임의 해당 필터에 대해 저장할 필터 속성입니다.
3. 세 번째 인수는 지정된 필터 속성의 값입니다.

앞에 나온 initFilters() 호출에서 다음과 같이 addFilterPropertyArray()를 호출하면 DropShadowFilter의 blurX 및 blurY 속성에 값 5가 할당됩니다. DropShadowFilter는 초기화된 필터 배열의 첫 번째 항목(인덱스 0)입니다.

```
__motion_Box.addFilterPropertyArray(0, "blurX", [5]);
__motion_Box.addFilterPropertyArray(0, "blurY", [5]);
```

다음 세 개의 호출에서는 초기화된 필터 배열의 두 번째 항목(인덱스 1)인 GlowFilter의 quality, alpha 및 color 속성에 값을 할당합니다.

```
__motion_Box.addFilterPropertyArray(1, "quality", [BitmapFilterQuality.LOW]);
__motion_Box.addFilterPropertyArray(1, "alpha", [1.00]);
__motion_Box.addFilterPropertyArray(1, "color", [0xff0000]);
```

다음 네 개의 호출에서는 초기화된 필터 배열의 세 번째 항목(인덱스 2)인 BevelFilter의 shadowAlpha, shadowColor, highlightAlpha 및 highlightColor에 값을 할당합니다.

```
__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]);
```

ColorMatrixFilter를 사용한 색상 조정

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

ColorMatrixFilter가 초기화된 후에는 적절한 AdjustColor 속성을 설정하여 트위닝된 표시 객체의 밝기, 대비, 채도 및 색조를 조정할 수 있습니다. 일반적으로 모션 트윈이 Flash에서 만들어진 경우 AdjustColor 필터가 적용되며 ActionScript의 복사본에서 세밀하게 조정할 수 있습니다. 다음 예제에서는 표시 객체가 이동할 때 해당 객체의 색조 및 채도를 변형합니다.

```
__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);
```

모션 트윈과 표시 객체 연결

Flash Player 9 이상, Adobe AIR 1.0 이상, Flash CS3 이상 필요

마지막 작업은 모션 트윈과 모션 트윈이 조작하는 표시 객체를 연결하는 것입니다.

AnimatorFactory 클래스는 모션 트윈과 대상 표시 객체 간의 연결을 관리합니다. AnimatorFactory 생성자의 인수는 Motion 객체입니다.

```
var __animFactory_Wheel:AnimatorFactory = new AnimatorFactory(__motion_Wheel);
```

AnimatorFactory 클래스의 addTarget() 메서드를 사용하여 대상 표시 객체를 모션 트윈과 연결합니다. Flash에서 복사된 ActionScript는 addTarget() 행을 주석 처리하며 인스턴스 이름을 지정하지 않습니다.

```
// __animFactory_Wheel.addTarget(<instance name goes here>, 0);
```

복사본에서 표시 객체를 지정하여 모션 트윈과 연결합니다. 다음 예제에서 대상은 greenWheel 및 redWheel로 지정됩니다.

```
__animFactory_Wheel.AnimatorFactory.addTarget(greenWheel, 0);
__animFactory_Wheel.AnimationFactory.addTarget(redWheel, 0);
```

addTarget()을 여러 번 호출하면 동일한 모션 트윈에 여러 표시 객체를 연결할 수 있습니다.

18장: 역기구학을 사용한 작업

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

역기구학(IK)은 실감나는 모션을 만들기 위한 뛰어난 기술입니다.

IK를 사용하면 IK 뼈대라고 하는 연결 부분의 체인 내에서 좌표가 지정된 움직임을 만들 수 있으므로 해당 부분들이 실제처럼 함께 이동합니다. 뼈대는 뼈와 연결점으로 구성됩니다. 뼈대의 끝점을 지정하면 IK는 해당 끝점에 도달하는 데 필요한 연결점 각도를 계산합니다.

이러한 각도를 직접 계산하기는 어렵습니다. 이 기능의 장점은 Adobe® Flash® Professional을 사용하여 뼈대를 대화식으로 만들 수 있다는 것입니다. 그런 다음 ActionScript를 사용하여 뼈대에 애니메이션 효과를 적용할 수 있습니다. Flash Professional에 포함된 IK 엔진은 뼈대의 움직임을 나타내기 위한 계산을 수행합니다. ActionScript 코드에서는 몇 가지 매개 변수로 움직임을 제한할 수 있습니다.

Flash Professional CS5 버전의 IK에는 뼈 반동 개념이 새로 포함되었습니다. 이 개념은 일반적으로 고성능 애니메이션 응용 프로그램과 관련됩니다. 이 기능을 새로운 동적 물리 엔진과 함께 사용하면 실제와 같은 움직임을 구성할 수 있습니다. 또한 이 효과는 런타임 및 제작 중에도 볼 수 있습니다.

역기구학 뼈대를 만들려면 Flash Professional 라이센스가 있어야 합니다.

기타 도움말 항목

fl.ik 패키지

역기구학의 기초

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

역기구학(IK)을 사용하면 여러 부분이 서로 유기적으로 실감나게 움직이도록 각 부분을 연결하여 실제적인 애니메이션을 만들 수 있습니다.

예를 들어 IK를 사용하면 원하는 포즈를 만드는 데 필요한 다리의 관절 움직임을 구조화하여 다리를 특정 위치로 이동할 수 있습니다. IK에서는 여러 개의 뼈가 IK 뼈대라는 하나의 구조로 서로 연결된 구성을 사용합니다. fl.ik 패키지는 실제 모션과 비슷한 애니메이션을 만드는 데 유용합니다. 이 패키지를 사용하면 IK 알고리즘의 기반이 되는 물리학적 지식이 많지 않아도 무리 없이 여러 IK 뼈대에 애니메이션을 적용할 수 있습니다.

Flash Professional을 사용하여 보조 뼈와 연결점이 있는 IK 뼈대를 만듭니다. 그런 다음 IK 클래스에 액세스하여 런타임에 IK 뼈대에 애니메이션을 적용할 수 있습니다.

IK 뼈대를 만드는 방법에 대한 자세한 내용은 **Flash Professional** 사용의 "역기구학 사용" 단원을 참조하십시오.

중요한 개념 및 용어

이 기능과 관련된 중요한 용어들이 아래 참조 목록에 정리되어 있습니다.

뼈대 실감나는 모션을 시뮬레이션하기 위해 컴퓨터 애니메이션에 사용되는 뼈와 연결점으로 구성된 기구학 체인입니다.

뼈 뼈대의 딱딱한 부분으로, 동물 뼈대의 뼈와 비슷합니다.

역기구학(IK) 기구학 체인 또는 뼈대라고 하는 유연한 연결 객체의 매개 변수를 결정하는 프로세스입니다.

연결점 두 개의 뼈가 연결되는 위치에서 뼈의 움직임이 가능하도록 구성된 연결 부분으로, 동물의 관절과 비슷합니다.

물리 엔진 애니메이션에 실제와 같은 동작을 제공하기 위해 사용되는 물리 관련 알고리즘의 모음입니다.

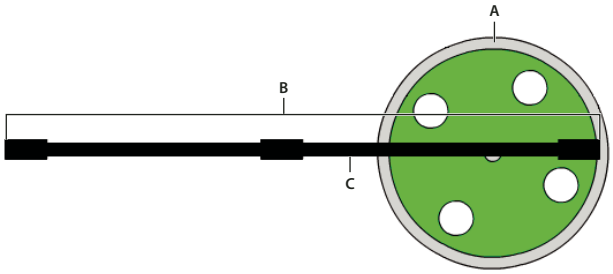
반동 상위 뼈대가 이동할 때 이동 및 반응하고 시간이 지날수록 점차 사라지는 뼈대의 품질입니다.

IK 뼈대 애니메이션 개요

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

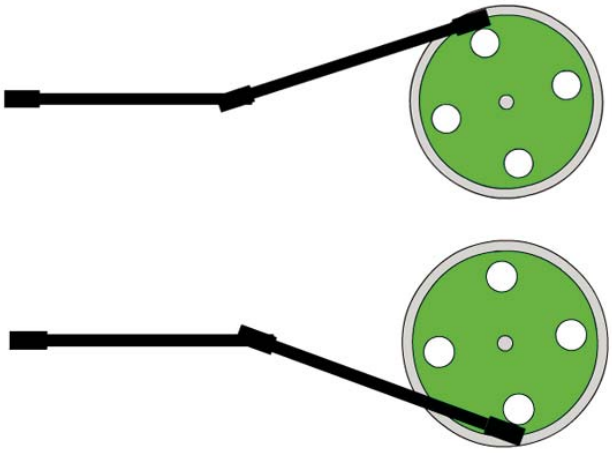
Flash Professional에서 IK 뼈대를 만든 후에는 fl.ik 클래스를 사용하여 움직임을 제한하고 이벤트를 추적하며 런타임에 움직임을 애니메이션 처리합니다.

다음 그림에서는 Wheel이라는 동영상 클립을 보여 줍니다. 축은 Axle이라는 IKArmature 인스턴스입니다. IKMover 클래스는 바퀴의 회전에 맞춰 뼈대를 움직입니다. 뼈대의 IKBone인 ikBone2는 꼬리 연결점에서 바퀴에 연결됩니다.



A. Wheel **B.** Axle **C.** ikBone2

런타임에 바퀴는 애니메이션 설명에서 설명한 305페이지의 "애니메이션 설명" 모션 트윈과 연관되어 회전합니다. IKMover 객체는 축의 움직임을 시작하고 제어합니다. 다음 그림에서는 회전하는 바퀴에 연결된 축 뼈대의 스냅샷 두 개를 각기 다른 회전 프레임으로 보여 줍니다.



런타임에 다음 ActionScript가 수행하는 작업은 아래와 같습니다.

- 뼈대 및 해당 구성 요소에 대한 정보 얻기
- IKMover 객체 인스턴스화
- 바퀴의 회전과 함께 축 이동

```
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);
    }
}
```

축을 이동하는 데 사용되는 IK 클래스는 다음과 같습니다.

- IKArmature: 뼈와 연결점으로 구성된 트리 구조의 뼈대를 나타내며, Flash Professional을 사용하여 만들어야 합니다.
- IKManager: 문서의 모든 IK 뼈대에 대한 컨테이너 클래스로서, Flash Professional을 사용하여 만들어야 합니다.
- IKBone: IK 뼈대의 한 부분입니다.
- IKJoint: 두 개의 IK 뼈가 연결되는 부분입니다.
- IKMover: 뼈대의 IK 움직임을 시작하고 제어합니다.

이러한 클래스에 대한 모든 자세한 내용은 ik 패키지를 참조하십시오.

IK 뼈대에 대한 정보 얻기

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

먼저 움직일 부분을 구성하는 뼈대, 뼈 및 연결점에 대한 변수를 선언합니다.

다음 코드에서는 IKManager 클래스의 getArmatureByName() 메서드를 사용하여 IKArmature의 tree 변수에 Axle 뼈대의 값을 할당합니다. Axle 뼈대는 Flash Professional을 사용하여 이미 만들어졌습니다.

```
var tree:IKArmature = IKManager.getArmatureByName("Axle");
```

마찬가지로 다음 코드에서는 IKArmature 클래스의 getBoneByName() 메서드를 사용하여 IKBone 변수에 ikBone2 뼈의 값을 할당합니다.

```
var bone:IKBone = tree.getBoneByName("ikBone2");
```

ikBone2 뼈의 꼬리 연결점은 뼈대에서 회전하는 바퀴에 연결되는 부분입니다.

다음 행에서는 endEffector 변수를 선언한 다음 ikBone2 뼈의 tailjoint 속성에 이 변수를 할당합니다.

```
var endEffector:IKJoint = home.tailjoint;
```

pos 변수는 endEffector 연결점의 현재 위치를 저장하는 부분입니다.

```
var pos:Point = endEffector.position;
```

이 예제에서 pos는 연결점이 축의 끝에서 바퀴에 연결되는 위치입니다. 이 변수의 원래 값은 IKJoint의 position 속성에서 가져옵니다.

IK Mover 인스턴스화 및 움직임 제한

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

IKMover 클래스의 인스턴스는 축을 움직입니다.

다음 행에서는 IKMover 객체 ik를 인스턴스화하고 해당 생성자에 이동할 요소와 이동 시작점을 전달합니다.

```
var ik:IKMover = new IKMover(endEffector, pos);
```

IKMover 클래스의 속성을 사용하면 뼈대의 움직임을 제한할 수 있습니다. 움직임의 거리, 반복 수 및 시간에 따라 움직임을 제한할 수 있습니다.

다음 속성 쌍은 이러한 제한을 적용합니다. 이 속성 쌍은 움직임이 제한되는지 여부를 나타내는 Boolean 값과 제한 정도를 지정하는 정수로 구성됩니다.

Boolean 속성	Integer 속성	제한 설정
limitByDistance:Boolean	distanceLimit:int	IK 엔진이 각 반복에 대해 움직이는 최대 거리(픽셀)를 설정합니다.
limitByIteration:Boolean	iterationLimit:int	IK 엔진이 각 움직임에 대해 수행하는 최대 반복 수를 설정합니다.
limitByTime:Boolean	timeLimit:int	움직임을 수행하기 위해 IK 엔진에 할당되는 최대 시간(밀리초)을 설정합니다.

기본적으로 모든 Boolean 값은 false로 설정되어 있으므로 Boolean 값을 true로 명시적으로 설정하지 않는 한 움직임이 제한되지 않습니다. 제한을 적용하려면 해당하는 속성을 true로 설정한 다음 해당하는 정수(integer) 속성의 값을 지정하십시오. 해당 Boolean 속성을 설정하지 않고 제한을 값으로 설정한 경우에는 제한이 무시됩니다. 이 경우 IK 엔진은 다른 제한 또는 IKMover의 대상 위치에 도달할 때까지 객체를 계속 움직입니다.

다음 예제에서는 뼈대 움직임의 최대 거리를 각 반복 당 0.1픽셀로 설정합니다. 모든 움직임의 최대 반복 수는 10으로 설정됩니다.

```
ik.limitByDistance = true;
ik.distanceLimit = 0.1;
ik.limitByIteration = true;
ik.iterationLimit = 10;
```

IK 뼈대 움직임

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

IKMover는 바퀴의 이벤트 리스너 내에서 축을 움직입니다. 바퀴의 각 enterFrame 이벤트에서는 뼈대의 새 대상 위치가 계산됩니다. moveTo() 메서드를 사용하면 IKMover는 꼬리 연결점을 limitByDistance, limitByIteration 및 limitByTime 속성으로 설정된 제한 범위 내에서 가능한 멀리 또는 대상 위치로 이동합니다.

```
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);
    }
}
```

반동 사용

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS5 이상 필요

Flash Professional CS5의 역기구학은 뼈 반동을 지원합니다. 뼈 반동은 제작 중에 설정할 수 있으며, 뼈 반동 특성은 런타임에 추가하거나 수정할 수 있습니다. 반동은 뼈 및 연결점의 속성입니다. 여기에는 반동의 크기를 설정하는 IKJoint.springStrength와 강도 값에 대한 저항을 추가하고 반동의 감소율을 변경하는 IKJoint.springDamping이라는 두 가지 특성이 있습니다.

반동 강도는 기본값 0(완전히 고정)에서 100(매우 느슨하며 물리 기능으로 제어됨) 사이의 퍼센트 값입니다. 반동이 있는 뼈는 연결점의 이동에 반응합니다. 다른 변환(회전, x 또는 y)이 함께 설정되어야만 반동 설정이 나타납니다.

반동 감폭은 기본값 0(저항 없음)에서 100(감폭 큼) 사이의 퍼센트 값입니다. 감폭은 뼈가 처음 이동을 시작하여 안정된 위치로 돌아올 때까지의 시간을 변경합니다.

IKArmature 객체에 반동이 설정되었는지 여부는 IKArmature.springsEnabled 속성으로 확인할 수 있습니다. 다른 반동 속성 및 메서드는 개별 IKJoint 객체에 속합니다. 연결점에는 각도 회전과 x 축 및 y 축에 따른 평행 이동을 설정할 수 있습니다. IKJoint.setSpringAngle을 사용하여 회전 연결점의 반동 각도를 배치하고, IKJoint.setSpringPt를 사용하여 평행 이동 연결점의 반동 위치를 배치할 수 있습니다.

아래 예에서는 이름으로 뼈를 선택하고 해당 tailJoint를 식별합니다. 이 코드에서는 부모 뼈대를 테스트하여 반동이 설정되었는지 확인한 후 연결점에 대해 반동 속성을 설정합니다.

```
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 이벤트 사용

Flash Player 10 이상, Adobe AIR 1.5 이상, Flash CS4 이상 필요

IKEvent 클래스를 사용하면 IK 이벤트에 대한 정보가 들어 있는 이벤트 객체를 만들 수 있습니다. IKEvent 정보는 지정된 시간, 거리 또는 반복 제한이 초과되었기 때문에 종료된 모션을 설명합니다.

다음 코드에서는 추적 시간 제한 이벤트에 대한 이벤트 리스너 및 핸들러를 보여 줍니다. 이 이벤트 핸들러는 IKMover의 시간 제한이 초과되면 발생하는 이벤트의 시간, 거리, 반복 수 및 연결점 속성을 보고합니다.

```
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);
}
```

19장: 3차원(3D)에서 작업

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Player 및 AIR 런타임에서는 3D 그래픽을 두 가지 방법으로 지원합니다. Flash 표시 목록의 3차원 표시 객체를 사용할 수 있는데, 이는 Flash 내용에 3차원 효과를 추가하는 경우에는 물론 다각형 수가 적은 객체에도 적합합니다. Flash Player 11 및 AIR 3 이상에서는 Stage3D API를 사용하여 복잡한 3D 장면을 렌더링할 수 있습니다.

Stage3D 뷰포트는 표시 객체가 아닙니다. 대신 3D 그래픽은 Flash 표시 목록 아래와 StageVideo 뷰포트 평면 위에 표시되는 뷰포트에 렌더링됩니다. Flash DisplayObject 클래스를 사용하여 장면을 만드는 대신, OpenGL 및 Direct3D와 비슷한 프로그래밍 가능한 3D 파이프라인을 사용합니다. 이 파이프라인은 삼각형 데이터 및 텍스처를 입력으로 사용하고, 사용자가 제공한 셰이더 프로그램을 통해 장면을 렌더링합니다. 하드웨어 가속은 클라이언트 컴퓨터에 지원되는 드라이버와 함께 호환 가능한 그래픽 처리 장치(GPU)가 있는 경우에 사용됩니다.

Stage3D에서는 매우 낮은 수준의 API를 제공합니다. 응용 프로그램에서는 Stage3D를 지원하는 3D 프레임워크를 사용하는 것이 좋습니다. 프레임워크를 직접 만들 수도 있고, 이미 사용 가능한 여러 상용 또는 오픈 소스 프레임워크 중 하나를 사용할 수도 있습니다.

Stage3D를 사용하여 3D 응용 프로그램을 개발하는 방법과 사용 가능한 3D 프레임워크에 대한 자세한 내용은 Flash Player Developer Center: Stage 3D를 참조하십시오.

3D 표시 객체의 기초

Flash Player 10 이상, Adobe AIR 1.5 이상

2차원 화면에 투영되는 2차원(2D) 객체와 3차원(3D) 객체의 주요 차이점은 객체에 세 번째 차원이 추가된다는 것입니다. 세 번째 차원을 통해 객체를 사용자가 보는 시점에서 가까이 또는 멀리 이동할 수 있습니다.

표시 객체의 z 속성에 숫자 값을 명시적으로 설정하면 3D 변형 행렬이 자동으로 만들어집니다. 이 행렬을 변경하여 해당 객체의 3D 변형 설정을 수정할 수 있습니다.

또한 3D 회전은 2D 회전과 다릅니다. 2D에서 회전 축은 항상 x/y 평면에 수직, 즉 z축에 있습니다. 3D에서 회전 축은 x, y 또는 z축을 기준으로 합니다. 회전을 설정하고 표시 객체의 크기 속성을 조절하여 3D 공간에서 객체를 이동할 수 있습니다.

중요한 개념 및 용어

다음 참조 목록에는 3D 그래픽을 프로그래밍할 때 사용되는 중요한 용어가 정리되어 있습니다.

원근 2D 평면에서 깊이 및 거리 효과를 주기 위해 평행선들이 소실점에 수렴하도록 표현하는 것입니다.

투영 2D 이미지를 보다 높은 차원의 객체로 제작하는 것으로, 3D 투영의 경우 3D 점을 2D 평면에 매핑합니다.

회전 객체에 포함된 모든 점을 원형 모션으로 이동하여 객체의 방향과 위치를 변경하는 것입니다.

변형 평행 이동, 회전, 크기 조절, 기울이기 또는 이러한 액션의 조합을 통해 3D 점 또는 점 집합을 변경하는 것입니다.

평행 이동 객체에 포함된 모든 점을 동일한 방향으로 같은 거리만큼 이동하여 객체의 위치를 변경하는 것입니다.

소실점 선형 원근으로 표시될 때 뒤쪽으로 멀어지는 평행선들이 만나는 것처럼 보이는 점입니다.

벡터 3D 벡터는 직교 좌표 x, y 및 z를 사용하여 3차원 공간에서의 점 또는 위치를 나타냅니다.

정점 모퉁이 점입니다.

텍스처 메쉬 3D 공간에서 객체를 정의하는 점입니다.

UV 매핑 3D 표면에 텍스처 또는 비트맵을 적용하는 방법입니다. UV 매핑은 이미지 상의 좌표에 가로(U) 축 및 세로(V) 축의 백분율로 값을 지정합니다.

T 값 객체를 현재 시점에서 앞으로 또는 뒤로 이동할 때 3D 객체의 크기를 결정하기 위한 배율 인수입니다.

컬링 특정 굴곡이 있는 표면을 렌더링하거나 렌더링하지 않는 것입니다. 컬링을 사용하면 현재 시점에서 보이지 않는 표면을 숨길 수 있습니다.

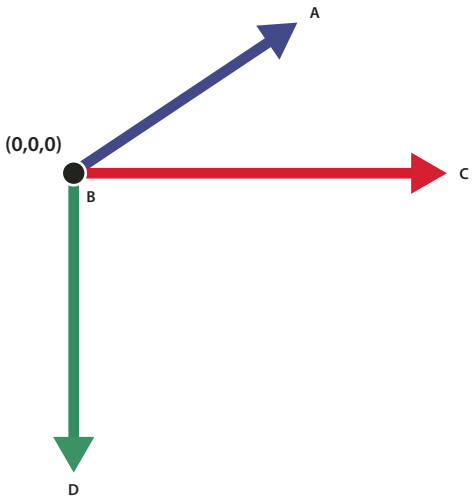
Flash Player 및 AIR 런타임에서 사용되는 3D 표시 객체 이해

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Player 10 이전의 Flash Player 버전과 Adobe AIR 1.5 이전의 Adobe AIR 버전에서는, 표시 객체에 2D 평면에서 위치를 지정하기 위한 두 가지 속성 x 및 y가 있습니다. Flash Player 10 및 Adobe AIR 1.5부터는 모든 ActionScript 표시 객체에 z 속성이 추가되었습니다. 이 속성을 사용하면 깊이나 거리를 나타내는 데 일반적으로 사용되는 z축을 따라 표시 객체의 위치를 지정할 수 있습니다.

Flash Player 10 및 Adobe AIR 1.5부터는 3D 효과가 지원됩니다. 그러나 표시 객체는 본질적으로 평면입니다. MovieClip 객체나 Sprite 객체와 같은 각 표시 객체는 최종적으로 단일 평면에 2차원으로 렌더링됩니다. 3D 기능을 사용하면 이러한 평면 객체를 세 차원 모두에서 배치, 이동, 회전 및 변형할 수 있습니다. 또한 3D 점을 관리하고 2D x, y 좌표로 변환하여 3D 객체를 2D 보기에 투영할 수 있습니다. 이러한 기능을 사용하여 여러 종류의 3D 환경을 시뮬레이션할 수 있습니다.

ActionScript에서 사용되는 3D 좌표계는 다른 좌표계와 차이점이 있습니다. ActionScript에서 2D 좌표를 사용하는 경우 x축을 따라 오른쪽으로 이동하면 x 값이 증가하고 y축을 따라 아래로 이동하면 y 값이 증가합니다. 3D 좌표계에서는 이러한 규칙이 유지되면서 시점에서 멀리 이동할 때 값이 증가하는 z축이 추가됩니다.



ActionScript 3D 좌표계에서 x, y 및 z축의 양의 방향
A. +Z축 **B.** 원점 **C.** +X축 **D.** +Y축

참고: Flash Player와 AIR에서는 항상 3D를 레이어로 나타냅니다. 즉, 객체 A는 표시 목록에서 객체 B 앞에 표시되며 Flash Player 또는 AIR에서는 두 객체의 z축 값에 관계없이 항상 B 앞에 A를 렌더링합니다. 표시 목록 순서와 z축 순서 간의 이러한 충돌을 해결하려면 transform.getRelativeMatrix3D() 메서드를 사용하여 저장한 다음 3D 표시 객체의 레이어 순서를 다시 지정합니다. 자세한 내용은 326페이지의 "Matrix3D 객체를 사용하여 표시 순서 재지정"을 참조하십시오.

새로운 3D 관련 기능을 지원하는 ActionScript 클래스는 다음과 같습니다.

1 flash.display.DisplayObject 클래스에는 3D 공간에서 표시 객체를 조작하기 위한 z 속성과 새로운 회전 및 크기 조절 속성이 포함되어 있습니다. DisplayObject.local3DToGlobal() 메서드를 사용하면 3D 기하 도형을 2D 평면에 손쉽게 투영할 수 있습니다.

2 flash.geom.Vector3D 클래스는 3D 점을 관리하기 위한 데이터 구조로 사용할 수 있습니다. 또한 이 클래스는 벡터 수학을 지원합니다.

3 flash.geom.Matrix3D 클래스는 3D 기하 도형의 회전, 크기 조절 및 평행 이동과 같은 복잡한 변형을 지원합니다.

4 flash.geom.PerspectiveProjection 클래스는 3D 기하 도형을 2D 보기에 매핑하기 위한 매개 변수를 제어합니다.

ActionScript에서는 다음 두 가지 방법으로 3D 이미지를 시뮬레이션할 수 있습니다.

1 3D 공간에서 평면 객체를 배열하고 애니메이션을 적용합니다. 이 방법을 사용할 경우 표시 객체의 x, y 및 z 속성을 사용하여 표시 객체에 애니메이션을 적용하거나, DisplayObject 클래스를 사용하여 회전 및 크기 조절 속성을 설정합니다. DisplayObject.transform.matrix3D 객체를 사용하면 보다 복잡한 모션을 얻을 수 있습니다. DisplayObject.transform.perspectiveProjection 객체는 표시 객체를 3D 원근으로 그리는 방법을 사용자 정의합니다. 주로 평면으로 구성된 3D 객체에 애니메이션을 적용하려면 이 방법을 사용합니다. 예를 들어 3D 공간에 3D 이미지 갤러리 또는 2D 애니메이션 객체를 정렬하는 방법이 이러한 방법에 포함됩니다.

2 3D 기하 도형에서 2D 삼각형을 생성하고 해당 삼각형을 텍스처로 렌더링합니다. 이 방법을 사용하려면 먼저 3D 객체에 대한 데이터를 정의 및 관리한 다음 렌더링을 위해 해당 데이터를 2D 삼각형으로 변환해야 합니다. 비트맵 텍스처를 이러한 삼각형에 매핑한 다음 Graphics.drawTriangles() 메서드를 사용하여 해당 삼각형을 그래픽 객체로 그릴 수 있습니다. 예를 들어 파일에서 3D 모델 데이터를 로드하고 모델을 화면에 렌더링하거나, 3D 지형을 삼각형 메쉬로 생성하고 그리는 방법이 이러한 방법에 포함됩니다.

3D 표시 객체 만들기 및 이동

Flash Player 10 이상, Adobe AIR 1.5 이상

2D 표시 객체를 3D 표시 객체로 변환하려면 표시 객체의 z 속성을 숫자 값으로 명시적으로 설정합니다. z 속성에 값을 할당하면 표시 객체에 대한 새 Transform 객체가 만들어집니다. DisplayObject.rotationX 또는 DisplayObject.rotationY 속성을 설정해도 새 Transform 객체가 만들어집니다. Transform 객체에는 표시 객체가 3D 공간에 표시되는 방식을 제어하는 Matrix3D 속성이 있습니다.

다음 코드에서는 "leaf"라는 표시 객체의 좌표를 설정합니다.

```
leaf.x = 100; leaf.y = 50; leaf.z = -30;
```

leaf라는 Transform 객체의 matrix3D 속성에서 이러한 값과 해당 값에서 파생된 속성을 볼 수 있습니다.

```
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);
```

Transform 객체의 속성에 대한 자세한 내용은 Transform 클래스를 참조하십시오. Matrix3D 객체의 속성에 대한 자세한 내용은 Matrix3D 클래스를 참조하십시오.

3D 공간에서 객체 이동

Flash Player 10 이상, Adobe AIR 1.5 이상

x, y 또는 z 속성의 값을 변경하여 3D 공간에서 객체를 이동할 수 있습니다. z 속성의 값을 변경하면 객체는 뷰어에서 더 가깝게 또는 더 멀리 표시됩니다.

다음 코드에서는 이벤트에 대한 응답으로 두 타원의 z 속성 값을 변경하여 해당 타원을 z축을 따라 뒤로 및 앞으로 이동합니다. ellipse2는 ellipse1보다 빠르게 이동합니다. 즉, 해당 z 속성은 각 프레임 이벤트에서 20의 배수만큼씩 증가하는 반면 ellipse1의 z 속성은 10의 배수만큼씩 증가합니다.

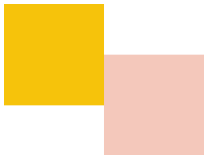
```
var depth:int = 1000;

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;
    }
}
```

3D 공간에서 객체 회전

Flash Player 10 이상, Adobe AIR 1.5 이상

객체의 회전 속성인 rotationX, rotationY 및 rotationZ를 어떻게 설정하느냐에 따라 세 가지 방법으로 객체를 회전할 수 있습니다.

다음 그림에서는 회전되지 않은 두 개의 사각형을 보여 줍니다.

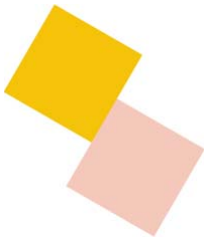


다음 그림에서는 두 사각형의 컨테이너에 대한 rotationY 속성 값을 늘려 y축을 기준으로 사각형을 회전한 모양을 보여 줍니다. 두 사각형의 컨테이너 또는 부모 표시 객체를 회전하면 두 사각형이 모두 회전합니다.

```
container.rotationY += 10;
```

다음 그림에서는 사각형의 컨테이너에 대한 rotationX 속성을 설정할 때 나타나는 결과를 보여 줍니다. 이렇게 하면 사각형이 x 축을 기준으로 회전합니다.

다음 그림에서는 사각형에 대한 컨테이너의 rotationZ 속성 값을 늘릴 때 나타나는 결과를 보여 줍니다. 이렇게 하면 사각형이 z 축을 기준으로 회전합니다.



표시 객체는 3D 공간에서 이동과 회전을 동시에 할 수 있습니다.

2D 보기에 3D 객체 투영

Flash Player 10 이상, Adobe AIR 1.5 이상

flash.geom 패키지의 PerspectiveProjection 클래스를 사용하면 3D 공간에서 표시 객체를 이동할 때 기본적인 원근을 손쉽게 적용할 수 있습니다.

3D 공간에 대한 원근 투영을 명시적으로 만들지 않을 경우 3D 엔진에서는 루트에 있으며 모든 자식 객체에 전달되는 기본 PerspectiveProjection 객체를 사용합니다.

다음은 PerspectiveProjection 객체의 3D 공간 표시 방식을 정의하는 세 가지 속성입니다.

- fieldOfView
- projectionCenter
- focalLength

fieldOfView와 focalLength는 상호 종속적이므로 한 쪽 값을 수정하면 다른 쪽 값도 자동으로 수정됩니다.

fieldOfView 값이 주어진 경우 focalLength를 계산하는 데 사용되는 수식은 다음과 같습니다.

```
focalLength = stageWidth/2 * (cos(fieldOfView/2) / sin(fieldOfView/2)
```

일반적으로 fieldOfView 속성은 명시적으로 수정합니다.

시야

Flash Player 10 이상, Adobe AIR 1.5 이상

PerspectiveProjection 클래스의 fieldOfView 속성을 조작하면 보는 사람 쪽을 향하는 3D 표시 객체는 크게 나타나고 보는 사람으로부터 벗어나는 객체는 작게 나타나도록 할 수 있습니다.

fieldOfView 속성은 원근 투영의 강도를 결정하는 각도를 0도에서 180도 사이로 지정합니다. 값이 클수록 z축을 따라 이동하는 표시 객체가 더 왜곡되어 나타납니다. fieldOfView 값이 작으면 크기가 매우 작게 조절되며 객체가 공간에서 뒤쪽으로 약간만 이동한 것처럼 나타납니다. fieldOfView 값이 크면 왜곡이 커지고 더 많이 이동한 것처럼 나타납니다. 최대값인 179.9999...도를 사용하면 카메라 어안 렌즈 효과가 최대로 나타납니다. fieldOfView의 최대값은 179.9999...이고 최소값은 0.00001...입니다. 정확히 0과 180은 잘못된 값입니다.

투영 중심

Flash Player 10 이상, Adobe AIR 1.5 이상

projectionCenter 속성은 원근 투영의 소실점을 나타냅니다. 소실점은 스테이지의 왼쪽 위 모서리에 있는 기본 등록 포인트(0,0)에 대한 오프셋으로 적용됩니다.

객체는 보는 사람으로부터 멀리 이동함에 따라 소실점을 향해 기울어지다가 마지막에는 소실됩니다. 즉, 무한히 긴 구멍을 상상하면 됩니다. 이 구멍을 내려다 보면 벽의 가장자리가 구멍 아래의 깊은 곳에서 소실점에 수렴됩니다.

소실점이 스테이지의 중심에 있으면 이 구멍은 중심점 쪽으로 사라집니다. projectionCenter 속성의 기본값은 스테이지의 중심입니다. 예를 들어 요소가 스테이지의 왼쪽에 나타나고 3D 공간이 오른쪽에 나타나도록 하려면 projectionCenter를 스테이지의 오른쪽에 있는 한 점으로 설정하여 해당 점을 3D 보기 공간의 소실점으로 만듭니다.

초점 거리

Flash Player 10 이상, Adobe AIR 1.5 이상

focalLength 속성은 z축 상에서 시점의 원점(0,0,0)과 표시 객체가 있는 위치 사이의 거리를 나타냅니다.

초점 거리를 길게 설정하면 망원 렌즈로 볼 때처럼 시야가 좁고 객체 간의 거리가 압축되어 나타납니다. 초점 거리를 짧게 설정하면 광각 렌즈로 볼 때처럼 시야가 넓고 왜곡이 심하게 나타납니다. 초점 거리를 중간 정도로 설정하면 사람의 눈으로 볼 때와 비슷하게 나타납니다.

일반적으로 focalLength는 표시 객체가 이동할 때 원근 변형 과정에서 동적으로 다시 계산되지만 사용자가 이를 명시적으로 설정할 수도 있습니다.

기본 원근 투영 값

Flash Player 10 이상, Adobe AIR 1.5 이상

루트에 만들어진 기본 PerspectiveProjection 객체의 각 속성 값은 다음과 같습니다.

- fieldOfView: 55
- perspectiveCenter: stagewidth/2, stageHeight/2
- focalLength: stageWidth/ 2 * (cos(fieldOfView/2) / sin(fieldOfView/2))

이러한 값은 사용자가 PerspectiveProjection 객체를 만들지 않을 경우에 사용되는 값입니다.

projectionCenter 및 fieldOfView를 직접 수정하려는 경우 PerspectiveProjection 객체를 직접 인스턴스화할 수 있습니다. 이 경우 새로 만든 객체는 기본 스테이지 크기인 500 x 500에서 다음과 같은 기본값을 가집니다.

- fieldOfView: 55
- perspectiveCenter: 250,250
- focalLength: 480.24554443359375

예제: 원근 투영

Flash Player 10 이상, Adobe AIR 1.5 이상

다음 예제에서는 원근 투영을 사용하여 3D 공간을 만드는 방법을 보여 줍니다. 또한 projectionCenter 속성을 통해 소실점을 수정하고 공간의 원근 투영을 변경하는 방법을 보여 줍니다. 이렇게 수정하면 focalLength와 fieldOfView가 다시 계산되고 그에 따라 3D 공간이 왜곡됩니다.

이 예제에서 수행하는 작업은 다음과 같습니다.

1. 십자형이 있는 원으로 center라는 스프라이트를 만듭니다.
2. 루트의 transform 속성의 perspectiveProjection 속성의 projectionCenter 속성에 center 스프라이트의 좌표를 할당합니다.
3. 마우스 이벤트에 대해 center 객체의 위치를 따르도록 projectionCenter를 수정하는 핸들러를 호출하는 이벤트 리스너를 추가합니다.
4. 원근 공간의 벽을 형성하는 네 개의 아코디언 스타일 상자를 만듭니다.

이 ProjectionDragger.swf 예제를 테스트할 때는 다른 위치에 원을 그립니다. 소실점은 이 원을 따라가다가 사용자가 원을 놓는 위치에 고정됩니다. 투영 중심을 스테이지의 중심에서 멀리 이동할 때 상자 내부의 공간이 확장되면서 왜곡되는 것을 살펴보십시오.

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. ProjectionDragger 응용 프로그램 파일은 Samples/ProjectionDragger 폴더에 있습니다.

```
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;
        }
    }
}
```

보다 복잡한 원근 투영의 경우에는 Matrix3D 클래스를 사용합니다.

복잡한 3D 변형 수행

Flash Player 10 이상, Adobe AIR 1.5 이상

Matrix3D 클래스를 사용하면 좌표 공간 내의 3D 점을 변경하거나 한 좌표 공간의 3D 점을 다른 좌표 공간에 매핑할 수 있습니다.

행렬 수학을 이해하지 못해도 Matrix3D 클래스를 사용할 수 있습니다. 일반적인 변형 작업은 대부분 이 클래스의 메서드를 사용하여 처리할 수 있습니다. 행렬에 있는 각 요소의 값을 명시적으로 설정하거나 계산하지 않아도 됩니다.

표시 객체의 z 속성을 숫자 값으로 설정한 후에는 표시 객체의 Transform 객체에 대한 Matrix3D 속성을 사용하여 객체의 변형 행렬을 가져올 수 있습니다.

```
var leafMatrix:Matrix3D = this.transform.matrix3D;
```

Matrix3D 객체의 메서드를 사용하여 표시 객체에 대한 평행 이동, 회전, 크기 조절 및 원근 투영을 수행할 수 있습니다.

3D 점을 관리하려면 Vector3D 클래스와 이 클래스의 x, y 및 z 속성을 사용합니다. 이 클래스는 방향과 크기가 있는 물리학적 공간 벡터를 나타낼 수도 있습니다. Vector3D 클래스의 메서드를 사용하면 공간 벡터를 통해 더하기, 내적 및 외적 계산과 같은 일반적인 계산을 수행할 수 있습니다.

참고: Vector3D 클래스는 ActionScript Vector 클래스와는 관련이 없습니다. Vector3D 클래스에는 3D 점을 정의하고 조작하기 위한 속성 및 메서드가 포함되어 있는 반면 Vector 클래스는 유형이 있는 객체의 배열을 지원합니다.

Matrix3D 객체 만들기

Flash Player 10 이상, Adobe AIR 1.5 이상

Matrix3D 객체를 만들거나 가져오는 기본적인 방법으로는 다음 세 가지가 있습니다.

- Matrix3D() 생성자 메서드를 사용하여 새 행렬을 인스턴스화합니다. Matrix3D() 생성자는 16개의 숫자 값이 들어 있는 Vector 객체를 인수로 받아 각 값을 행렬의 셀에 배치합니다. 예를 들면 다음과 같습니다.

  ```
  var rotateMatrix:Matrix3D = new Matrix3D(1,0,0,1, 0,1,0,1, 0,0,1,1, 0,0,0,1);
  ```

- 표시 객체의 z 속성 값을 설정합니다. 그런 다음 해당 객체의 transform.matrix3D 속성에서 변형 행렬을 가져옵니다.
- 루트 표시 객체의 perspectiveProjection.toMatrix3D() 메서드를 호출하여 스테이지에 3D 객체가 표시되는 방식을 제어하는 Matrix3D 객체를 가져옵니다.

여러 가지 3D 변형 적용

Flash Player 10 이상, Adobe AIR 1.5 이상

Matrix3D 객체를 사용하여 여러 가지 3D 변형을 한 번에 적용할 수 있습니다. 예를 들어 정육면체의 회전, 크기 조절 및 이동을 수행하려는 경우 정육면체의 각 점에 세 가지 개별 변형을 적용할 수 있습니다. 그러나 하나의 Matrix3D 객체에서 여러 변형을 미리 계산한 다음 각 점에 대해 하나의 행렬 변환을 수행하는 것이 훨씬 더 효율적입니다.

참고: 행렬 변환이 적용되는 순서는 중요합니다. 행렬 계산에서는 교환법칙이 성립되지 않기 때문입니다. 예를 들어 회전을 적용한 다음 평행 이동을 적용할 때와 평행 이동을 적용한 다음 회전을 적용할 때의 결과는 서로 다릅니다.

다음 예제에서는 여러 가지 3D 변형을 수행하는 두 가지 방법을 보여 줍니다.

```
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;
        }
    }
}
```

doTransforms() 메서드의 첫 번째 코드 블록에서는 DisplayObject 속성을 사용하여 사각형 모양의 회전, 크기 조절 및 위치를 변경합니다. 두 번째 코드 블록에서는 Matrix3D 클래스의 메서드를 사용하여 동일한 변형을 수행합니다.

Matrix3D 메서드를 사용할 때의 주된 장점은 모든 계산이 행렬에서 먼저 수행된다는 것입니다. 그런 다음 transform.matrix3D 속성이 설정되었을 때 모든 계산이 표시 객체에 한 번만 적용됩니다. DisplayObject 속성을 설정하면 소스 코드가 좀 더 읽기 쉽게 됩니다. 그러나 회전 또는 크기 조절 속성이 설정될 때마다 여러 번 계산이 수행되고 여러 개의 표시 객체 속성이 변경됩니다.

코드에서 표시 객체에 동일한 복잡한 변형을 두 번 이상 적용할 경우에는 Matrix3D 객체를 변수로 저장한 다음 이를 반복해서 다시 적용합니다.

Matrix3D 객체를 사용하여 표시 순서 재지정

Flash Player 10 이상, Adobe AIR 1.5 이상

앞에서 언급했듯이 표시 목록에 있는 표시 객체의 레이어 순서는 상대적 z축에 관계없이 표시 레이어 순서를 결정합니다. 애니메이션에서 표시 객체의 속성을 표시 목록 순서와는 다른 순서로 변형할 경우 보는 사람에게는 표시 객체 레이어가 z축 레이어와 일치하지 않게 표시됩니다. 따라서 보는 사람에게서 멀리 떨어진 객체가 보는 사람과 가까이 있는 객체의 앞에 나타날 수 있습니다.

3D 표시 객체의 레이어가 객체의 상대적 깊이와 일치하게 하려면 다음과 같은 방법을 사용합니다.

1. 변형 객체의 getRelativeMatrix3D() 메서드를 사용하여 자식 3D 표시 객체의 상대적 z축을 가져옵니다.
2. removeChild() 메서드를 사용하여 객체를 표시 목록에서 제거합니다.
3. 상대적 z축 값에 따라 표시 객체를 정렬합니다.
4. addChild() 메서드를 사용하여 자식 객체를 표시 목록에 역순으로 다시 추가합니다.

이렇게 순서를 다시 지정하면 객체가 상대적 z축에 따라 표시됩니다.

다음 코드에서는 3D 상자의 여섯 개 면을 올바르게 표시합니다. 상자에 회전을 적용한 후 상자의 면 순서를 다시 지정합니다.

```
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);
    }
}
```

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. 응용 프로그램 파일은 Samples/ReorderByZ 폴더에 있습니다.

3D 효과에 삼각형 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

ActionScript에서는 Graphics.drawTriangles() 메서드를 사용하여 비트맵 변형을 수행합니다. 공간에서 3D 모델은 삼각형의 모음으로 나타나기 때문입니다. 그러나 Flash Player와 AIR에서는 깊이 버퍼를 지원하지 않으므로 표시 객체는 여전히 본질적으로는 평면, 즉 2D입니다. 자세한 내용은 317페이지의 “Flash Player 및 AIR 런타임에서 사용되는 3D 표시 객체 이해”를 참조하십시오. Graphics.drawTriangles() 메서드는 Graphics.drawPath() 메서드와 같이 일련의 좌표를 인수로 받아 삼각형 패스를 그립니다.

Graphics.drawPath()의 사용 방법을 이해하려면 213페이지의 “패스 그리기”를 참조하십시오.

Graphics.drawTriangles() 메서드는 Vector.<Number>를 사용하여 삼각형 패스의 점 위치를 지정합니다.

```
drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null, uvtData:Vector.<Number> = null,
culling:String = "none"):void
```

drawTriangles()의 첫 번째 매개 변수는 유일한 필수 매개 변수인 vertices 매개 변수입니다. 이 매개 변수는 삼각형을 그릴 좌표를 정의하는 숫자의 벡터입니다. 세 개의 좌표 집합(여섯 개의 숫자) 모두가 하나의 삼각형 패스를 나타냅니다. 각 삼각형에는 세 개의 좌표 쌍(두 개의 x/y 값으로 구성된 집합 세 개)이 필요하므로 indices 매개 변수를 사용하지 않을 경우 벡터의 길이는 항상 6의 인수가 됩니다. 예를 들면 다음과 같습니다.

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([
        10,10,  100,10,  10,100,
        110,10, 110,100, 20,100]));
```

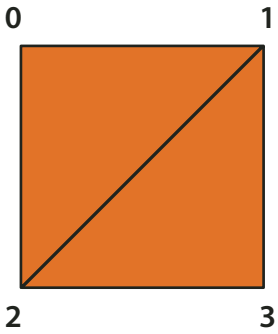
이러한 삼각형은 모두 어떤 점도 공유하지 않지만 삼각형의 점이 공유될 경우에는 drawTriangles()의 두 번째 매개 변수인 indices를 사용하여 두 개 이상의 삼각형에 vertices 벡터의 값을 다시 사용할 수 있습니다.

indices 매개 변수를 사용할 경우 indices 값은 vertices 배열 요소와 직접적으로 관련된 인덱스가 아니라 점 인덱스여야 합니다. 즉, indices로 정의된 vertices 벡터의 인덱스는 사실상 실제 인덱스를 2로 나눈 것입니다. 예를 들어 vertices 벡터의 세 번째 점의 경우 해당 점의 첫 번째 숫자 값은 벡터 인덱스 4에서 시작하지만 indices 값으로는 2를 사용합니다.

예를 들어 다음과 같이 indices 매개 변수를 사용하여 두 개의 삼각형이 대각선 가장자리를 공유하도록 병합할 수 있습니다.

```
graphics.beginFill(0xFF8000);
graphics.drawTriangles(
    Vector.<Number>([10,10, 100,10, 10,100, 100,100]),
    Vector.<int>([0,1,2, 1,3,2]));
```

지금은 두 개의 삼각형을 사용하여 사각형이 그려지지는 않았지만 vertices 벡터에 네 개의 점만 지정되었습니다. indices를 사용하면 두 삼각형이 공유하는 두 개의 점은 각 삼각형에 대해 다시 사용됩니다. 이렇게 하면 전체 정점 수가 6개(12개의 숫자)에서 4개(8개의 숫자)로 줄어듭니다.



vertices 매개 변수를 사용하여 두 개의 삼각형으로 그려진 사각형

이 기술은 대부분의 점이 여러 삼각형에서 공유되는 큰 삼각형 메쉬에서 유용합니다.

삼각형에는 모든 채우기를 적용할 수 있습니다. 채우기는 다른 모양에서와 마찬가지로 결과 삼각형 메쉬에 적용됩니다.

비트맵 변형

Flash Player 10 이상, Adobe AIR 1.5 이상

비트맵 변형을 사용하면 3차원 객체에 원근 또는 "텍스처" 효과를 줄 수 있습니다. 특히 비트맵을 소실점 쪽으로 왜곡시켜 이미지가 소실점 쪽으로 이동할수록 작게 나타나도록 할 수 있습니다. 또는 2차원 비트맵을 사용하여 3차원 객체의 표면을 만들고 이를 통해 3차원 객체에 텍스처 또는 "둘러싸기" 효과를 줄 수 있습니다.

소실점을 사용한 2차원 표면과 비트맵으로 둘러싼 3차원 객체

UV 매핑

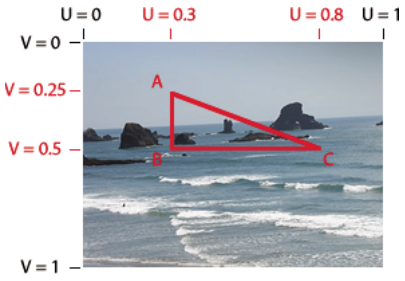
Flash Player 10 이상, Adobe AIR 1.5 이상

텍스처 작업을 시작한 후에는 drawTriangles()의 uvtData 매개 변수를 사용할 수 있습니다. 이 매개 변수를 사용하면 비트맵 채우기를 위한 UV 매핑을 설정할 수 있습니다.

UV 매핑은 객체에 텍스처를 적용하기 위한 방법으로, U 가로(x) 값과 V 세로(y) 값을 사용합니다. 이 두 값은 픽셀 값이 아니라 백분율을 기준으로 합니다. 0U 및 0V는 이미지의 왼쪽 위 모서리이고 1U 및 1V는 오른쪽 아래 모서리입니다.

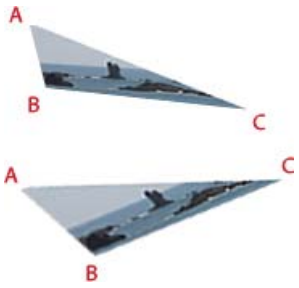
비트맵 이미지상의 UV 0 및 1 위치

삼각형의 벡터에 UV 좌표를 지정하면 벡터를 이미지 상의 각 위치와 연결할 수 있습니다.



비트맵 이미지에 있는 삼각형 영역의 UV 좌표

UV 값은 삼각형의 점과 일관되게 유지됩니다.



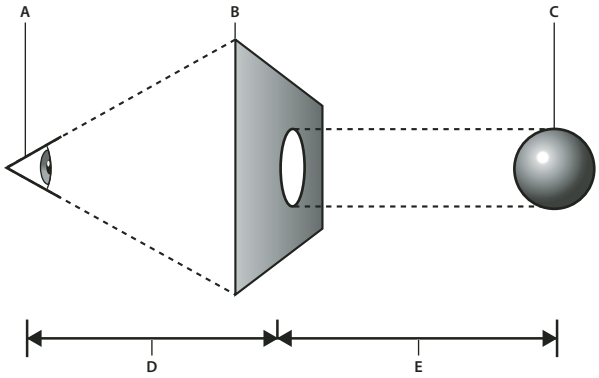
삼각형의 정점이 이동하면서 개별 점의 UV 값이 동일하도록 비트맵이 왜곡된 모양

비트맵과 연결된 삼각형에 ActionScript 3D 변형이 적용될 때 비트맵 이미지는 UV 값을 기준으로 삼각형에 적용됩니다. 따라서 행렬 계산을 사용하는 대신 UV 값을 설정하거나 조정하여 3차원 효과를 만듭니다.

Graphics.drawTriangles() 메서드는 3차원 변형을 위해 T 값이라는 선택적 정보를 사용하기도 합니다. uvtData의 T 값은 3D 원근, 보다 구체적으로는 연결된 정점의 비율 인수를 나타냅니다. UVT 매핑에서는 UV 매핑에 원근 교정을 추가로 사용합니다. 예를 들어 객체가 3D 공간에서 시점으로부터 멀리 떨어져 있어 "원래" 크기의 50%로 나타날 경우 해당 객체의 T 값은 0.5가 됩니다. 3D 공간에서는 객체를 나타내기 위해 삼각형을 그리므로 z축 상에서의 객체 위치가 객체의 T 값을 결정합니다. T 값을 결정하는 수식은 다음과 같습니다.

```
T = focalLength/(focalLength + z);
```

이 수식에서 focalLength는 초점 거리나 보기에 제공되는 원근 정도를 지정하는 계산된 "화면" 위치를 나타냅니다.



초점 거리 및 z 값
A. 시점 **B.** 화면 **C.** 3D 객체 **D.** focalLength 값 **E.** z 값

T 값은 기본 모양의 크기를 조절하여 해당 모양이 더 멀리 있는 것처럼 보이게 하는 데 사용됩니다. 이 값은 일반적으로 3D 점을 2D 점으로 변환하는 데 사용되는 값입니다. UVT 데이터의 경우에는 원근이 적용된 삼각형 내의 점 사이에서 비트맵 크기를 조절하는 데도 이 값이 사용됩니다.

UVT 값을 정의할 때 T 값은 정점에 대해 정의된 UV 값을 직접적으로 따릅니다. T 값이 포함된 경우 uvtData 매개 변수의 세 값(U, V 및 T)은 모두 vertices 매개 변수(x 및 y)의 두 값 모두와 일치합니다. UV 값만 사용할 경우 uvtData.length == vertices.length입니다. T 값이 포함된 경우 uvtData.length = 1.5*vertices.length입니다.

다음 예제에서는 UVT 데이터를 사용하여 3D 공간에서 평면을 회전하는 방법을 보여 줍니다. 이 예제에서는 ocean.jpg라는 이미지를 BitmapData 객체에 할당할 수 있도록 ocean.jpg 이미지와 "도우미" 클래스인 ImageLoader를 사용하여 해당 이미지를 로드합니다.

다음은 ImageLoader 클래스의 소스입니다. 이 코드를 ImageLoader.as라는 파일에 저장하십시오.

```
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);
    }
    }
}
```

다음은 삼각형, UV 매핑 및 T 값을 사용하여 이미지가 소실점 쪽으로 점점 작아지고 회전하는 것처럼 보이도록 만드는 ActionScript입니다. 이 코드를 Spinning3dOcean.as라는 파일에 저장하십시오.

```
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);
```

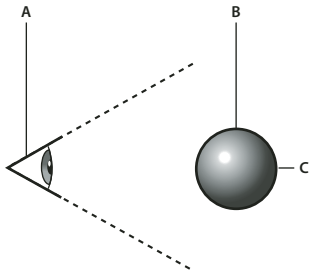
```
            container = new Sprite(); // container to draw triangles in
            container.x = 200;
            container.y = 200;
            addChild(container);

            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);
        }
    }
}
```

이 예제를 테스트하려면 이 두 클래스 파일을 "ocean.jpg"라는 이미지와 동일한 디렉토리에 저장하십시오. 원본 비트맵이 3D 공간에서 먼 곳으로 소실되고 회전하는 것처럼 보이도록 변형되는 방식을 볼 수 있습니다.

컬링

Flash Player 10 이상, Adobe AIR 1.5 이상

컬링은 3차원 객체의 표면 중 현재 시점에서 숨겨지기 때문에 렌더러에서 렌더링하지 않아야 하는 표면을 결정하는 프로세스입니다. 3D 공간에서 3차원 객체의 "뒤쪽"에 있는 표면은 시점에서 숨겨집니다.



3D 객체의 뒤쪽은 시점에서 숨겨집니다.
A. 시점 **B.** 3D 객체 **C.** 3차원 객체의 뒤쪽

본질적으로 모든 삼각형은 크기, 모양 또는 위치에 관계없이 항상 렌더링됩니다. 컬링은 Flash Player나 AIR에서 3D 객체를 올바르게 렌더링하도록 합니다. 또한 렌더링 주기를 줄이기 위해 필요한 경우 렌더러에서 일부 삼각형을 건너뛰도록 할 수 있습니다. 예를 들어 공간에서 회전하는 정육면체를 가정해 봅니다. 보기에 나타나지 않는 면은 정육면체 반대쪽의 다른 방향을 향하고 있는 것이므로 한 번에 볼 수 있는 정육면체의 면은 최대 세 개입니다. 이러한 면은 표시되지 않으므로 렌더러에서 해당 면을 그리지 말아야 합니다. 컬링을 사용하지 않을 경우 Flash Player나 AIR에서는 앞쪽 면과 뒤쪽 면을 모두 렌더링합니다.

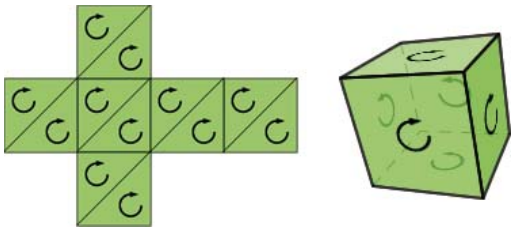
현재 시점에서 보이지 않는 면이 있는 정육면체

따라서 Graphics.drawTriangles() 메서드는 네 번째 변수를 사용하여 컬링 값을 설정합니다.

```
public function drawTriangles(vertices:Vector.<Number>, indices:Vector.<int> = null,
uvtData:Vector.<Number> = null, culling:String = "none"):void
```

컬링 매개 변수는 TriangleCulling 열거형 클래스의 값인 TriangleCulling.NONE, TriangleCulling.POSITIVE 및 TriangleCulling.NEGATIVE 중 하나입니다. 이러한 값은 객체의 표면을 정의하는 삼각형 패스의 방향에 따라 달라집니다. 컬링을 결정하기 위한 ActionScript API에서는 3D 모양에서 바깥쪽을 향하는 모든 삼각형이 동일한 패스 방향으로 그려져 있다고 가정합니다. 삼각형 면을 돌리면 패스 방향도 변경됩니다. 해당 지점에서 삼각형을 컬링(렌더링에서 제거)할 수 있습니다.

따라서 TriangleCulling 값이 POSITIVE이면 패스가 양수 방향(시계 방향)인 삼각형이 제거되고, TriangleCulling 값이 NEGATIVE이면 패스가 음수 방향(반시계 방향)인 삼각형이 제거됩니다. 정육면체의 경우 앞쪽을 향하는 표면의 패스는 양수 방향이고 뒤쪽을 향하는 표면의 패스는 음수 방향입니다.



패스 방향을 보여 주기 위해 "펼친" 정육면체. "접힌" 상태에서는 뒤쪽 면의 패스 방향이 반대가 됩니다.

컬링의 작동 방식을 보려면 앞에 나온 328페이지의 “UV 매핑”의 예제에서 drawTriangles() 메서드의 컬링 매개 변수를 TriangleCulling.NEGATIVE로 설정합니다.

```
container.graphics.drawTriangles(vertices, indices, uvtData, TriangleCulling.NEGATIVE);
```

객체가 회전할 때 이미지의 "뒤쪽" 면은 렌더링되지 않습니다.

20장: 텍스트를 사용한 작업의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe® Flash® Player 또는 Adobe® AIR™에서 텍스트를 화면에 표시하려면 TextField 클래스의 인스턴스를 사용하거나 Flash Text Engine 클래스를 사용합니다. 이러한 클래스를 사용하면 텍스트를 생성 및 표시하고 형식을 지정할 수 있습니다. 또는 Flash Text Engine 클래스에 기반하지만 보다 쉽게 사용할 수 있도록 설계된 구성 요소 라이브러리인 TLF(Text Layout Framework)를 사용할 수도 있습니다. 휴대 장치에서 StageText 클래스를 텍스트 입력에 사용할 수 있습니다.

텍스트 필드에 특정 내용을 구성하거나 텍스트의 소스를 지정한 다음 해당 텍스트의 모양을 설정할 수 있습니다. 또한 사용자가 텍스트를 입력하거나 하이퍼텍스트 링크를 클릭할 때 사용자 이벤트에 응답할 수도 있습니다.

TextField 클래스와 Flash Text Engine 클래스는 모두 Flash Player 및 AIR에서 텍스트를 표시하고 관리할 수 있도록 해 줍니다. TextField 클래스를 사용하여 표시 및 입력을 위한 텍스트 객체를 만들 수 있습니다. TextField 클래스는 TextArea, TextInput 등의 다른 텍스트 기반 구성 요소에 대한 기초를 제공합니다. TextFormat 클래스를 사용하여 TextField 객체의 문자 및 단락 서식을 설정하고 Textfield.styleSheet 속성과 StyleSheet 클래스를 사용하여 CSS(CSS 스타일 시트)를 적용할 수 있습니다. 동영상 클립, SWF 파일, GIF 파일, PNG 파일 및 JPEG 파일 등의 포함된 미디어를 포함할 수 있는 HTML 형식의 텍스트를 텍스트 필드에 직접 할당할 수 있습니다.

Flash Player 10 및 Adobe AIR 1.5부터 사용할 수 있는 Flash Text Engine은 낮은 수준에서 텍스트 메트릭, 서식 및 양방향 텍스트를 정교하게 제어할 수 있으며 향상된 텍스트 흐름 및 언어 지원도 제공합니다. Flash Text Engine은 텍스트 요소를 만들고 관리하는 데 사용할 수 있지만, 기본적으로 텍스트 처리 구성 요소를 만들기 위한 기반으로 디자인되었으며 이를 사용하려면 보다 높은 수준의 프로그래밍 전문성이 필요합니다. Flash Text Engine 기반 텍스트 처리 구성 요소를 포함하는 Text Layout Framework를 사용하면 새로운 텍스트 엔진의 고급 기능을 더욱 쉽게 사용할 수 있습니다. Text Layout Framework는 ActionScript 3.0에 완전하게 구축되어 있는 확장 가능한 라이브러리입니다. 기존 TLF 구성 요소를 사용하거나, 프레임워크를 사용하여 고유한 텍스트 구성 요소를 구축할 수 있습니다.

AIR 3부터 사용할 수 있는 StageText 클래스는 기본 텍스트 입력 필드를 제공합니다. 이 필드는 장치 운영 체제에서 제공되므로 장치 사용자에게 가장 익숙한 환경을 제공합니다. StageText 인스턴스는 표시 객체가 아닙니다. 이 인스턴스를 표시 목록에 추가하는 대신 스테이지와 뷰포트라고 하는 해당 스테이지의 표시 영역을 인스턴스에 할당합니다. StageText 인스턴스는 모든 표시 객체 앞에 표시됩니다.

이러한 항목에 대한 자세한 내용은 다음을 참조하십시오.

- 336페이지의 “TextField 클래스 사용”
- 359페이지의 “Flash Text Engine 사용”
- 386페이지의 “Text Layout Framework 사용”
- StageText를 사용한 기본 텍스트 입력

중요한 개념 및 용어

다음 참조 목록에는 텍스트 처리와 관련된 중요한 용어가 정리되어 있습니다.

CSS 스타일 시트 XML(또는 HTML) 형식으로 구성된 내용의 스타일 및 서식을 지정하는 표준 구문입니다.

장치 글꼴 사용자의 컴퓨터에 설치되어 있는 글꼴입니다.

동적 텍스트 필드 사용자 입력이 아닌 ActionScript를 통해 내용을 변경할 수 있는 텍스트 필드입니다.

포함된 글꼴 응용 프로그램 SWF 파일에 문자 외곽선 데이터가 저장되어 있는 글꼴입니다.

HTML 텍스트 ActionScript를 사용하여 텍스트 필드에 입력한 텍스트 내용으로, 실제 텍스트 내용과 함께 HTML 서식 태그가 포함됩니다.

입력 텍스트 필드 사용자 입력 또는 ActionScript를 통해 내용을 변경할 수 있는 텍스트 필드입니다.

커닝 단어 내 간격의 균형이 맞고 텍스트를 보다 쉽게 읽을 수 있도록 두 문자 사이의 간격을 조정하는 것입니다.

정적 텍스트 필드 제작 도구에서 만든 텍스트 필드로, SWF 파일이 실행 중인 경우 해당 내용을 변경할 수 없습니다.

텍스트 선 메트릭 텍스트 기준선, 문자 상단의 높이, 디센더 크기(일부 소문자에서 기준선 아래로 내려가는 부분) 등 텍스트 필드의 텍스트 내용 중 다양한 부분의 크기를 측정합니다.

자간 밀도를 늘리거나 줄이고 텍스트를 보다 쉽게 읽을 수 있도록 문자 그룹 또는 텍스트 블록 사이의 간격을 조정하는 것입니다.

21장: TextField 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

TextField 클래스 인스턴스를 사용하여 Adobe® Flash® Player 또는 Adobe® AIR™에서 화면에 텍스트를 표시하거나 텍스트 입력 필드를 만들 수 있습니다. TextField 클래스는 TextArea 구성 요소나 TextInput 구성 요소 등의 다른 텍스트 기반 구성 요소의 기초가 됩니다.

텍스트 필드의 내용은 SWF 파일에서 미리 지정하거나, 텍스트 파일 또는 데이터베이스에서 로드하거나, 응용 프로그램과 상호 작용하는 사용자가 입력할 수 있습니다. 텍스트 필드 내에서 텍스트는 렌더링된 HTML에 포함된 이미지와 함께 렌더링된 HTML 내용으로 표시될 수 있습니다. 텍스트 필드 인스턴스를 만든 후에는 TextFormat 및 StyleSheet와 같은 flash.text 클래스를 사용하여 텍스트의 모양을 제어할 수 있습니다. flash.text 패키지에는 ActionScript의 텍스트 생성, 관리 및 서식 지정과 관련된 거의 모든 클래스가 포함되어 있습니다.

TextFormat 객체를 사용하여 서식을 정의하고 해당 객체를 텍스트 필드에 할당하여 텍스트의 서식을 지정할 수 있습니다. 텍스트 필드에 HTML 텍스트가 포함된 경우 해당 텍스트 필드에 StyleSheet 객체를 적용하여 텍스트 필드 내용의 특정 부분에 스타일을 할당할 수 있습니다. TextFormat 객체 또는 StyleSheet 객체에는 색상, 크기, 두께 등 텍스트의 모양을 정의하는 속성이 포함됩니다. TextFormat 객체는 한 텍스트 필드 내의 모든 내용이나 특정 텍스트 범위에 속성을 할당합니다. 예를 들어 동일한 텍스트 필드 내에서 한 문장은 굵은 빨간색 텍스트로 지정하고, 다음 문장은 파란색 이탤릭체 텍스트로 지정할 수 있습니다.

flash.text 패키지의 클래스뿐 아니라 flash.events.TextEvent 클래스를 사용하여 텍스트와 관련된 사용자 액션에 대응할 수 있습니다.

기타 도움말 항목

342페이지의 “텍스트 서식 지정”

337페이지의 “HTML 텍스트 표시”

343페이지의 “CSS 스타일 시트 적용”

텍스트 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Builder 및 Flash Professional과 같은 제작 도구에서는 텍스트 관련 구성 요소 또는 텍스트 도구 등 여러 가지 텍스트 표시 옵션을 제공하지만 가장 간단한 텍스트 표시 방법은 텍스트 필드를 통해 프로그래밍 방식으로 텍스트를 표시하는 것입니다.

텍스트 유형

Flash Player 9 이상, Adobe AIR 1.0 이상

텍스트 필드 내의 텍스트 유형은 다음과 같이 해당 소스에 따라 특징이 결정됩니다.

- 동적 텍스트

  동적 텍스트에는 텍스트 파일, XML 파일 또는 원격 웹 서비스 등 외부 소스에서 로드한 내용이 포함됩니다.

- 입력 텍스트

입력 텍스트는 사용자가 입력한 모든 텍스트 또는 사용자가 편집할 수 있는 동적 텍스트입니다. 스타일 시트를 설정하여 입력 텍스트의 서식을 지정하거나 flash.text.TextFormat 클래스를 사용하여 입력 내용의 텍스트 필드에 속성을 할당할 수 있습니다. 자세한 내용은 340페이지의 "텍스트 입력 캡처"를 참조하십시오.

- 정적 텍스트

  정적 텍스트는 Flash Professional를 통해서만 만듭니다. ActionScript 3.0을 사용하여 정적 텍스트 인스턴스를 만들 수 없습니다. 그러나 StaticText 및 TextSnapshot과 같은 ActionScript 클래스를 사용하여 기존의 정적 텍스트 인스턴스를 조작할 수 있습니다. 자세한 내용은 348페이지의 "정적 텍스트를 사용한 작업"을 참조하십시오.

텍스트 필드 내용 수정

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.text.TextField.text 속성에 문자열을 할당하여 동적 텍스트를 정의할 수 있습니다. 다음과 같이 속성에 문자열을 직접 할당합니다.

```
myTextField.text = "Hello World";
```

다음 예와 같이 스크립트에 정의된 변수의 값을 text 속성에 할당할 수도 있습니다.

```
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;
        }
    }
}
```

또는 text 속성에 원격 변수의 값을 할당할 수도 있습니다. 원격 소스에서 텍스트 값을 로드할 수 있는 세 가지 옵션이 있습니다.

- flash.net.URLLoader 및 flash.net.URLRequest 클래스는 로컬 또는 원격 위치의 텍스트에 대한 변수를 로드합니다.
- FlashVars 특성은 SWF 파일을 호스트하는 HTML 페이지에 포함되어 있으며 텍스트 변수의 값을 포함할 수 있습니다.
- flash.net.SharedObject 클래스는 값의 영구적 저장소를 관리합니다. 자세한 내용은 639페이지의 "로컬 데이터 저장"을 참조하십시오.

HTML 텍스트 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.text.TextField 클래스에는 텍스트 문자열을 내용의 서식을 지정하기 위한 HTML 태그가 포함된 텍스트 문자열로 식별하는 데 사용할 수 있는 htmlText 속성이 포함됩니다. 텍스트를 HTML로 렌더링하려면 다음과 같이 Flash Player 또는 AIR의 htmlText 속성(text 속성 아님)에 문자열 값을 할당해야 합니다.

```
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 및 AIR에서는 htmlText 속성에 대해 일부 HTML 태그 및 항목을 지원합니다. ActionScript 3.0 참조 설명서의 flash.text.TextField.htmlText 속성 설명에는 지원되는 HTML 태그 및 항목에 대한 자세한 내용이 나와 있습니다.

htmlText 속성을 사용하여 내용을 지정한 후에는 스타일 시트 또는 textformat 태그를 사용하여 내용의 서식을 관리할 수 있습니다. 자세한 내용은 342페이지의 “텍스트 서식 지정”을 참조하십시오.

텍스트 필드에 이미지 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

내용을 HTML 텍스트로 표시하는 것의 또 다른 장점은 텍스트 필드에 이미지를 포함할 수 있다는 것입니다. img 태그를 사용하여 로컬 또는 원격 이미지를 참조하고 연결된 텍스트 필드 내에 표시되도록 할 수 있습니다.

다음 예제에서는 myTextBox라는 텍스트 필드를 만들고 SWF 파일과 동일한 디렉토리에 저장된 눈 모양의 JPG 이미지를 표시되는 텍스트 내에 포함합니다.

```
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;
        }
    }
}
```

img 태그는 JPEG, GIF, PNG 및 SWF 파일을 지원합니다.

텍스트 필드의 텍스트 스크롤

Flash Player 9 이상, Adobe AIR 1.0 이상

많은 경우 입력할 텍스트의 길이가 해당 텍스트를 표시하는 텍스트 필드보다 길 수 있습니다. 또는 한 번에 표시할 수 있는 것보다 더 많은 텍스트를 입력 필드에 입력할 수 있는 경우도 있습니다. flash.text.TextField 클래스의 스크롤 관련 속성을 사용하여 길이가 긴 내용을 세로 또는 가로로 관리할 수 있습니다.

스크롤 관련 속성으로는 TextField.scrollV, TextField.scrollH, maxScrollV 및 maxScrollH가 있습니다. 이러한 속성을 사용하여 마우스 클릭 또는 키 누르기와 같은 이벤트에 대응할 수 있습니다.

다음 예제에서는 설정된 크기의 텍스트 필드를 만들고 해당 필드에서 한 번에 표시할 수 있는 것보다 더 많은 텍스트를 포함합니다. 사용자가 텍스트 필드를 클릭하면 텍스트가 세로로 스크롤됩니다.

```
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++;
        }
    }
}
```

텍스트 선택 및 조작

Flash Player 9 이상, Adobe AIR 1.0 이상

동적 텍스트 또는 입력 텍스트를 선택할 수 있습니다. TextField 클래스의 텍스트 선택 속성 및 메서드는 인덱스 위치를 사용하여 조작할 텍스트 범위를 설정하므로 내용을 모르더라도 프로그래밍 방식으로 동적 텍스트 또는 입력 텍스트를 선택할 수 있습니다.

참고: Flash Professional을 사용하여 정적 텍스트 필드에서 선택 가능한 옵션을 선택하는 경우 표시 목록에 내보내거나 배치하는 텍스트 필드는 일반 동적 텍스트 필드가 됩니다.

텍스트 선택

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.text.TextField.selectable 속성은 기본적으로 true로 설정되어 있고 setSelection() 메서드를 사용하여 프로그래밍 방식으로 텍스트를 선택할 수 있습니다.

예를 들어 다음과 같이 사용자가 텍스트 필드를 클릭할 때 텍스트 필드 내의 특정 텍스트가 선택되도록 설정할 수 있습니다.

```
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);
}
```

마찬가지로, 텍스트가 처음 표시될 때 텍스트 필드 내의 텍스트가 선택되도록 하려는 경우에는 텍스트 필드가 표시 목록에 추가될 때 호출되는 이벤트 핸들러 함수를 만듭니다.

사용자가 선택한 텍스트 캡처

Flash Player 9 이상, Adobe AIR 1.0 이상

프로그래밍 방식으로 텍스트를 선택하도록 설정할 수 없는 "읽기 전용" TextField selectionBeginIndex 및 selectionEndIndex 속성을 사용하여 현재 사용자가 선택한 항목을 캡처할 수 있습니다. 또한 입력 텍스트 필드는 caretIndex 속성을 사용할 수 있습니다.

예를 들어 다음 코드는 사용자가 선택한 텍스트의 인덱스 값을 추적합니다.

```
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);
}
```

선택 항목에 TextFormat 객체 속성 컬렉션을 적용하여 텍스트 모양을 변경할 수 있습니다. 선택한 텍스트에 TextFormat 속성 컬렉션을 적용하는 방법에 대한 자세한 내용은 345페이지의 "텍스트 필드 내 텍스트 범위 서식 지정"을 참조하십시오.

텍스트 입력 캡처

Flash Player 9 이상, Adobe AIR 1.0 이상

기본적으로 텍스트 필드의 type 속성은 dynamic으로 설정됩니다. TextFieldType 클래스를 사용하여 type 속성을 input으로 설정하면 사용자 입력을 수집하여 해당 값을 응용 프로그램의 다른 부분에서 사용할 수 있도록 저장할 수 있습니다. 입력 텍스트 필드는 프로그램의 다른 위치에서 사용하기 위해 사용자가 텍스트 값을 정의하도록 할 모든 응용 프로그램 및 양식에 유용합니다.

예를 들어 다음 코드는 myTextBox라는 입력 텍스트 필드를 만듭니다. 사용자가 필드에 텍스트를 입력하면 textInput 이벤트가 트리거됩니다. textInputCapture라는 이벤트 핸들러는 입력된 텍스트 문자열을 캡처하고 해당 문자열에 변수를 할당합니다. Flash Player 또는 AIR는 myOutputBox라는 또 다른 텍스트 필드에 새 텍스트를 표시합니다.

```
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;
        }

    }
}
```

텍스트 입력 제한

Flash Player 9 이상, Adobe AIR 1.0 이상

입력 텍스트 필드는 응용 프로그램의 양식이나 대화 상자에 사용되는 경우가 많으므로 텍스트 필드에 입력할 수 있는 문자의 유형을 제한하거나 텍스트(예: 암호)를 숨겨야 할 수도 있습니다. flash.text.TextField 클래스에는 사용자 입력을 제어하기 위해 설정할 수 있는 displayAsPassword 속성과 restrict 속성이 있습니다.

displayAsPassword 속성은 사용자가 텍스트를 입력할 때 해당 텍스트를 별표로 표시하여 숨깁니다. displayAsPassword를 true로 설정하는 경우 잘라내기 및 복사 명령과 해당 키보드 단축키가 작동하지 않습니다. 다음 예제에서 볼 수 있는 것과 같이 배경 및 색상 등의 다른 속성과 마찬가지 방법으로 displayAsPassword 속성을 할당합니다.

```
myTextBox.type = TextFieldType.INPUT;
myTextBox.background = true;
myTextBox.displayAsPassword = true;
addChild(myTextBox);
```

restrict 속성은 입력 텍스트 필드에 사용자가 입력할 수 있는 문자 유형을 지정해야 하므로 조금 더 복잡합니다. 특정 문자, 숫자 또는 문자 및 숫자 범위를 허용할 수 있습니다. 다음 코드는 사용자가 텍스트 필드에 대문자만 입력할 수 있고 숫자나 특수 문자는 입력할 수 없도록 합니다.

```
myTextBox.restrict = "A-Z";
```

ActionScript 3.0에서는 하이픈을 사용하여 범위를 정의하고, 캐럿을 사용하여 제외되는 문자를 정의합니다. 입력 텍스트 필드에서 제한되는 항목을 정의하는 방법에 대한 자세한 내용은 ActionScript 3.0 참조 설명서의 flash.text.TextField.restrict 속성 항목을 참조하십시오.

참고: flash.text.TextField.restrict 속성을 사용할 경우 런타임에서는 제한된 문자를 허용된 대소문자로 자동으로 변환합니다. fl.text.TLFTextField.restrict 속성을 사용할 경우, 즉 TLF 텍스트 필드를 사용할 경우 런타임에서는 제한된 문자를 무시합니다.

텍스트 서식 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

프로그래밍 방식으로 텍스트 표시 서식을 지정할 수 있는 여러 가지 옵션이 있습니다. TextField 인스턴스에 직접 TextFIeld.thickness, TextField.textColor 및 TextField.textHeight 속성 등의 속성을 설정할 수 있습니다.또는 htmlText 속성을 사용하여 텍스트 필드의 내용을 지정하고 b, i, u 등 지원되는 HTML 태그를 사용할 수도 있습니다. 그러나 일반 텍스트가 포함된 텍스트 필드에 TextFormat 객체를 적용하거나 htmlText 속성이 포함된 텍스트 필드에 StyleSheet 객체를 적용할 수도 있습니다. TextFormat 및 StyleSheet 객체를 사용하면 응용 프로그램 전체에서 텍스트의 모양을 최대한 제어하고 일관성을 유지할 수 있습니다. TextFormat 또는 StyleSheet 객체를 정의하여 응용 프로그램 내 여러 텍스트 필드 또는 모든 텍스트 필드에 적용할 수 있습니다.

텍스트 서식 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

TextFormat 클래스를 사용하여 서로 다른 여러 텍스트 표시 속성을 설정하고 이를 한 TextField 객체의 전체 내용 또는 특정 텍스트 범위에 적용할 수 있습니다.

다음 예제에서는 한 TextFormat 객체를 한 TextField 객체 전체에 적용하고, 또 다른 TextFormat 객체를 해당 TextField 객체 내의 특정 텍스트 범위에 적용합니다.

```
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);
```

TextField.setTextFormat() 메서드는 텍스트 필드에 이미 표시되어 있는 텍스트에만 영향을 미칩니다. TextField의 내용이 변경되는 경우 해당 서식을 다시 적용하려면 응용 프로그램에서 TextField.setTextFormat() 메서드를 다시 호출해야 할 수 있습니다. TextField defaultTextFormat 속성을 설정하여 사용자가 입력한 텍스트에 사용할 형식을 지정할 수도 있습니다.

CSS 스타일 시트 적용

Flash Player 9 이상, Adobe AIR 1.0 이상

텍스트 필드에는 일반 텍스트 또는 HTML 형식의 텍스트가 포함될 수 있습니다. 일반 텍스트는 인스턴스의 text 속성에 저장되고, HTML 텍스트는 htmlText 속성에 저장됩니다.

CSS 스타일 선언을 사용하여 서로 다른 여러 텍스트 필드에 적용할 수 있는 텍스트 스타일을 정의할 수 있습니다. CSS 스타일 선언은 응용 프로그램 코드에서 만들거나 런타임에 외부 CSS 파일에서 로드할 수 있습니다.

flash.text.StyleSheet 클래스는 CSS 스타일을 처리합니다. StyleSheet 클래스는 제한된 CSS 속성 집합을 인식합니다. StyleSheet 클래스에서 지원하는 스타일 속성의 자세한 목록은 ActionScript 3.0 참조 설명서의 flash.textStylesheet 항목을 참조하십시오.

다음 예제에서 볼 수 있는 것과 같이 코드에서 CSS를 만들고 StyleSheet 객체를 사용하여 해당 스타일을 HTML 텍스트에 적용할 수 있습니다.

```
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);
```

StyleSheet 객체를 만든 후 예제 코드에서 스타일 선언 속성 집합을 포함할 단순 객체를 만듭니다. 그런 다음 StyleSheet.setStyle() 메서드를 호출하여 ".darkred"라는 스타일 시트에 새 스타일을 추가합니다. 그 다음에는 StyleSheet 객체를 TextField styleSheet 속성에 할당하여 스타일 시트 서식을 적용합니다.

CSS 스타일의 효과가 나타나려면 htmlText 속성을 설정하기 전에 TextField 객체에 스타일 시트를 적용해야 합니다.

설계상 스타일 시트가 있는 텍스트 필드는 편집할 수 없습니다. 입력 텍스트 필드가 있고 해당 필드에 스타일 시트를 할당하는 경우 해당 텍스트 필드에 스타일 시트의 속성이 표시되지만 사용자가 새 텍스트를 입력할 수는 없습니다. 또한 스타일 시트가 할당된 텍스트 필드에는 다음 ActionScript API를 사용할 수 없습니다.

- TextField.replaceText() 메서드

- TextField.replaceSelectedText() 메서드
- TextField.defaultTextFormat 속성
- TextField.setTextFormat() 메서드

텍스트 필드에 할당된 스타일 시트가 있지만 나중에 TextField.styleSheet 속성을 null로 설정하는 경우 TextField.text 및 TextField.htmlText 속성의 내용이 모두 해당 내용에 태그와 특성을 추가하여 이전에 할당된 스타일 시트의 서식을 통합합니다. 원래 htmlText 속성을 유지하려면 한 변수에 해당 속성을 저장한 후 스타일 시트를 null로 설정합니다.

외부 CSS 파일 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

CSS를 통해 서식을 지정하는 방법은 런타임에 외부 파일에서 CSS 정보를 로드할 수 있는 경우 더욱 유용합니다. CSS 데이터가 응용 프로그램 외부에 있는 경우 ActionScript 3.0소스 코드를 변경하지 않고도 응용 프로그램에서 텍스트의 시각적 스타일을 변경할 수 있습니다. 응용 프로그램을 배포한 후 응용 프로그램 SWF 파일을 다시 배포하지 않고도 외부 CSS 파일을 변경하여 응용 프로그램의 모양을 바꿀 수 있습니다.

StyleSheet.parseCSS() 메서드는 CSS 데이터가 포함된 문자열을 StyleSheet 객체의 스타일 선언으로 변환합니다. 다음 예제는 외부 CSS 파일을 읽고 TextField 객체에 해당 스타일 선언을 적용하는 방법을 보여 줍니다.

우선 example.css라는 CSS 파일의 내용을 로드합니다.

```
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;
}
```

다음은 example.css 파일을 로드하고 TextField 내용에 해당 스타일을 적용하는 클래스의 ActionScript 코드입니다.

```
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;
        }
    }
}
```

CSS 데이터가 로드되면 onCSSFileLoaded() 메서드가 StyleSheet.parseCSS() 메서드를 실행 및 호출하여 스타일 선언을 StyleSheet 객체로 전송합니다.

텍스트 필드 내 텍스트 범위 서식 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

flash.text.TextField 클래스에서 유용한 메서드는 setTextFormat() 메서드입니다. setTextFormat()을 사용하면 텍스트 필드 일부의 내용에 특정 속성을 할당하여 사용자 입력에 응답할 수 있습니다. 예를 들어 이 메서드는 사용자가 텍스트의 일부를 선택할 때 사용자에게 텍스트 필드 내 텍스트 구문의 하위 섹션 강조를 변경해야 하거나 특정 항목이 필요하다는 것을 알려야 하는 양식에 유용합니다.

다음 예제에서는 특정 문자 범위에 TextField.setTextFormat()을 사용하여 사용자가 텍스트 필드를 클릭할 때 myTextField의 내용 중 일부의 모양이 변경되도록 합니다.

```
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);
}
```

고급 텍스트 렌더링

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 포함된 글꼴, 앤티앨리어싱 설정, 알파 채널 컨트롤 및 기타 특정 설정 등 표시되는 텍스트의 속성을 제어할 수 있는 flash.text 패키지의 다양한 클래스를 제공합니다. ActionScript 3.0 참조 설명서에는 CSMSettings, Font, TextRenderer 클래스를 비롯하여 이러한 클래스 및 속성에 대한 자세한 설명이 나와 있습니다.

포함된 글꼴 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램에서 TextField의 특정 글꼴을 지정하면 Flash Player 또는 AIR가 이름이 같은 장치 글꼴(사용자의 컴퓨터에 있는 글꼴)을 찾습니다. 시스템에서 해당 글꼴을 찾지 못하거나, 이름은 같지만 글꼴의 버전이 약간 다른 경우에는 텍스트의 모양이 의도와 매우 다르게 표시될 수 있습니다. 기본적으로 텍스트는 Times Roman 글꼴로 표시됩니다.

정확히 원하는 글꼴이 표시되도록 하려면 응용 프로그램 SWF 파일에 해당 글꼴을 포함할 수 있습니다. 포함된 글꼴에는 다음과 같은 여러 가지 이점이 있습니다.

- 포함된 글꼴 문자에는 앤티앨리어싱이 적용되어 가장자리가 보다 부드럽게 표시됩니다(특히 큰 텍스트의 경우).
- 포함된 글꼴을 사용하는 텍스트를 회전할 수 있습니다.
- 포함된 글꼴 텍스트는 투명하거나 반투명하게 만들 수 있습니다.
- 포함된 글꼴과 함께 커닝 CSS 스타일을 사용할 수 있습니다.

포함된 글꼴 사용에 있어 가장 큰 제한 사항은 포함된 글꼴로 인해 파일 크기 또는 응용 프로그램의 다운로드 크기가 커진다는 것입니다.

글꼴 파일을 응용 프로그램 SWF 파일에 포함하는 정확한 방법은 개발 환경에 따라 다릅니다.

글꼴을 포함한 후에는 다음과 같이 TextField에서 올바른 포함된 글꼴을 사용하도록 할 수 있습니다.

- TextField의 embedFonts 속성을 true로 설정합니다.
- TextFormat 객체를 만들고 fontFamily 속성을 포함된 글꼴의 이름으로 설정한 다음 해당 TextFormat 객체를 TextField에 적용합니다. 포함된 글꼴을 지정할 때 fontFamily 속성에는 하나의 이름만 포함해야 하며 여러 글꼴 이름의 쉼표로 분리된 목록을 사용할 수 없습니다.
- CSS 스타일을 사용하여 TextField 또는 구성 요소의 글꼴을 설정하는 경우 font-family CSS 속성을 포함된 글꼴의 이름으로 설정합니다. 포함된 글꼴을 지정하려는 경우 font-family 속성에 하나의 이름을 포함해야 하며 여러 이름의 목록을 사용할 수 없습니다.

Flash에서 글꼴 포함

Flash Professional을 사용하면 TrueType 글꼴 및 Type 1 Postscript 글꼴 등 시스템에 설치한 거의 모든 글꼴을 포함할 수 있습니다.

다음과 같은 여러 가지 방법으로 응용 프로그램에 글꼴을 포함할 수 있습니다.

- 스테이지에서 TextField의 글꼴 및 스타일 속성을 설정하고 [글꼴 포함] 체크 상자를 클릭
- 글꼴 심볼 생성 및 참조
- 포함된 글꼴 심볼이 들어 있는 런타임 공유 라이브러리 생성 및 사용

응용 프로그램에 글꼴을 포함하는 방법에 대한 자세한 내용은 **Flash 사용**의 "동적 또는 입력 텍스트 필드의 포함된 글꼴"을 참조하십시오.

Flex에서 글꼴 포함

다음과 같은 여러 가지 방법으로 Flex 응용 프로그램에 글꼴을 포함할 수 있습니다.

- 스크립트에 [Embed] 메타데이터 태그 사용
- @font-face 스타일 선언 사용
- 글꼴에 대한 클래스를 구성하고 [Embed] 태그를 사용하여 해당 클래스를 포함합니다.

TrueType 글꼴만 Flex 응용 프로그램에 직접 포함할 수 있습니다. Type 1 Postscript 글꼴 등 다른 형식의 글꼴은 먼저 Flash Professional을 사용하여 SWF 파일에 포함한 다음 해당 SWF 파일을 Flex 응용 프로그램에 사용할 수 있습니다. Flex에서 SWF 파일의 포함된 글꼴을 사용하는 방법에 대한 자세한 내용은 **Using Flex 4**의 "Embedding fonts from SWF files"를 참조하십시오.

기타 도움말 항목

일관된 텍스트 모양을 위한 글꼴 포함

Peter deHaan: 글꼴 포함

Divillysausages.com: AS3 글꼴 포함 masterclass

선명도, 두께 및 앤티앨리어싱 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

기본적으로 텍스트의 크기 또는 색상을 변경하거나 텍스트를 다양한 배경에 표시할 때 Flash Player 또는 AIR에서 선명도, 두께, 앤티앨리어싱 등 텍스트 표시 컨트롤의 설정을 결정합니다. 텍스트의 크기가 매우 작거나 클 때 또는 여러 가지 고유한 배경에 텍스트를 표시할 때 등 일부 경우에는 이러한 설정을 직접 제어할 수도 있습니다. flash.text.TextRenderer 클래스 및 CSMSettings 클래스와 같은 관련 클래스를 사용하여 Flash Player 또는 AIR 설정을 무시할 수 있습니다. 이러한 클래스를 사용하면 포함된 텍스트의 렌더링 품질을 정밀하게 제어할 수 있습니다. 포함된 글꼴에 대한 자세한 내용은 346페이지의 “포함된 글꼴 사용”을 참조하십시오.

참고: flash.text.TextField.antiAliasType 속성에 AntiAliasType.ADVANCED 값이 있어야만 선명도, 두께 또는 gridFitType 속성을 설정하거나 TextRenderer.setAdvancedAntiAliasingTable() 메서드를 사용할 수 있습니다.

다음 예제에서는 myFont라는 포함된 글꼴을 사용하여 사용자 정의 CSM(Continuous Stroke Modulation) 속성 및 서식을 표시된 텍스트에 적용합니다. 사용자가 표시된 텍스트를 클릭하면 Flash Player 또는 Adobe AIR에서 사용자 정의 설정을 적용합니다.

```
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);
}
```

정적 텍스트를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

정적 텍스트는 Flash Professional 내에서만 만들 수 있습니다. ActionScript를 사용하여 정적 텍스트를 프로그래밍 방식으로 인스턴스화할 수 없습니다. 정적 텍스트는 텍스트가 짧고 변경할 의도가 없는 경우 유용합니다(동적 텍스트는 변경 가능). 정적 텍스트를 Flash Professional에서 스테이지에 그린 원이나 정사각형 같은 그래픽 요소와 비슷하다고 생각해 보십시오. 정적 텍스트가 동적 텍스트보다 더 제한적이기는 하지만 ActionScript 3.0에서는 StaticText 클래스를 사용하여 정적 텍스트의 속성 값을 읽을 수 있습니다. TextSnapshot 클래스를 사용하여 정적 텍스트에서 값을 읽을 수도 있습니다.

StaticText 클래스를 사용하여 정적 텍스트 필드 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

일반적으로 Flash Professional의 [액션] 패널에서 flash.text.StaticText 클래스를 사용하여 스테이지에 배치된 정적 텍스트 인스턴스와 상호 작용할 수 있습니다. 또한 정적 텍스트가 포함된 SWF 파일과 상호 작용하는 ActionScript 파일에서 작업할 수도 있습니다. 두 가지 경우 모두 정적 텍스트 인스턴스를 프로그래밍 방식으로 인스턴스화할 수 없습니다. 정적 텍스트는 Flash Professional에서 만듭니다.

기존 정적 텍스트 필드에 대한 참조를 만들려면 표시 목록의 항목을 반복하고 변수를 할당합니다. 예를 들면 다음과 같습니다.

```
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);
}
}
```

정적 텍스트 필드에 대한 참조가 있으면 ActionScript 3.0에서 해당 필드의 속성을 사용할 수 있습니다. 다음 코드는 타임라인의 특정 프레임에 연결되어 있고 myFieldLabel이라는 변수가 정적 텍스트 참조에 할당되어 있다고 가정합니다. myField라는 동적 텍스트 필드는 x 및 y 값(myFieldLabel의 값)을 기준으로 배치되고 myFieldLabel의 값을 다시 표시합니다.

```
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
```

TextSnapshot 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

기존의 정적 텍스트 인스턴스를 사용하여 프로그래밍 방식으로 작업하려는 경우 flash.text.TextSnapshot 클래스를 사용하여 flash.display.DisplayObjectContainer의 textSnapshot 속성으로 작업할 수 있습니다. 즉, DisplayObjectContainer.textSnapshot 속성에서 TextSnapshot 인스턴스를 만듭니다. 그런 다음 해당 인스턴스에 메서드를 적용하여 값을 검색하거나 정적 텍스트의 일부를 선택할 수 있습니다.

예를 들어 "TextSnapshot Example"이라는 텍스트가 포함된 정적 텍스트 필드를 스테이지에 배치합니다. 타임라인의 프레임 1에 다음 ActionScript를 추가합니다.

```
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);
```

TextSnapshot 클래스는 로드된 SWF 파일의 정적 텍스트 필드에서 가져온 텍스트를 응용 프로그램의 다른 부분에서 값으로 사용하려는 경우 해당 텍스트를 가져오는 데 유용합니다.

TextField 예제: 신문 스타일 텍스트 서식

Flash Player 9 이상, Adobe AIR 1.0 이상

News Layout 예제에서는 텍스트가 인쇄된 신문의 기사처럼 표시되도록 텍스트의 서식을 지정합니다. 입력 텍스트에는 헤드라인, 소제목 및 기사 본문이 포함될 수 있습니다. 표시 폭 및 높이가 주어진 상태에서 이 News Layout 예제에서는 헤드라인과 소제목이 표시 영역의 전체 폭을 차지하도록 서식이 지정됩니다. 기사 텍스트는 두 개 이상의 열에 나뉘어 배치됩니다.

이 예제에서는 다음과 같은 ActionScript 프로그래밍 기술을 보여 줍니다.

- TextField 클래스 확장
- 외부 CSS 파일 로드 및 적용
- CSS 스타일을 TextFormat 객체로 변환
- TextLineMetrics 클래스를 사용하여 텍스트 표시 크기에 대한 정보 확인

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. News Layout 응용 프로그램 파일은 Samples/NewsLayout 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
NewsLayout.mxml 또는 NewsLayout.fla	Flex(MXML) 또는 Flash(FLA)용 응용 프로그램의 사용자 인터페이스입니다.
com/example/programmingas3/newslayout/StoryLayoutComponent.as	StoryLayout 인스턴스를 배치하는 Flex UIComponent 클래스입니다.
com/example/programmingas3/newslayout/StoryLayout.as	표시를 위해 뉴스 기사의 모든 구성 요소를 배열하는 기본 ActionScript 클래스입니다.
com/example/programmingas3/newslayout/FormattedTextField.as	해당 TextFormat 객체를 관리하는 TextField 클래스의 하위 클래스입니다.
com/example/programmingas3/newslayout/HeadlineTextField.as	원하는 폭에 맞게 글꼴 크기를 조정하는 FormattedTextField 클래스의 하위 클래스입니다.
com/example/programmingas3/newslayout/MultiColumnTextField.as	텍스트를 두 개 이상의 열로 분할하는 ActionScript 클래스입니다.
story.css	해당 레이아웃용 텍스트 스타일을 정의하는 CSS 파일입니다.

외부 CSS 파일 읽기

Flash Player 9 이상, Adobe AIR 1.0 이상

News Layout 응용 프로그램에서는 우선 로컬 XML 파일에서 기사 텍스트를 읽습니다. 그런 다음 헤드라인, 소제목 및 주요 텍스트의 서식 정보를 제공하는 외부 CSS 파일을 읽습니다.

CSS 파일은 기사의 표준 단락 스타일, 헤드라인의 h1 스타일, 소제목의 h2 스타일 등 세 가지 스타일을 정의합니다.

```
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;
}
```

외부 CSS 파일을 읽는 데 사용되는 기술은 344페이지의 “외부 CSS 파일 로드”에서 설명한 기술과 동일합니다. CSS 파일이 로드되면 응용 프로그램에서 아래와 같이 onCSSFileLoaded() 메서드를 실행합니다.

```
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();
}
```

onCSSFileLoaded() 메서드는 StyleSheet 객체를 만들고 이 객체를 통해 입력 CSS 데이터를 파싱합니다. 기사의 주요 텍스트는 MultiColumnTextField 객체에 표시되며 이 객체는 StyleSheet 객체를 직접 사용할 수 있습니다. 그러나 헤드라인 필드에서는 서식 지정에 TextFormat 객체를 사용하는 HeadlineTextField 클래스를 사용합니다.

onCSSFileLoaded() 메서드는 getTextStyle() 메서드를 두 번 호출하여 두 HeadlineTextField 객체에 각각 사용할 수 있도록 CSS 스타일 선언을 TextFormat 객체로 변환합니다.

```
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;
}
```

속성 이름과 속성 값의 의미는 CSS 스타일 선언과 TextFormat 객체 간에 서로 다릅니다. getTextStyle() 메서드는 CSS 속성 값을 TextFormat 객체에서 기대하는 값으로 변환합니다.

페이지에 기사 요소 배열

Flash Player 9 이상, Adobe AIR 1.0 이상

StoryLayout 클래스는 헤드라인, 소제목 및 주요 텍스트 필드의 서식과 레이아웃을 신문 스타일 배열로 지정합니다. displayText() 메서드는 먼저 다양한 필드를 만들고 배치합니다.

```
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;
...
```

각 필드는 해당 y 속성을 이전 필드의 y 속성에 높이를 더한 값과 동일하게 설정하여 이전 필드 아래에 배치됩니다. HeadlineTextField 객체와 MultiColumnTextField 객체는 내용에 맞게 높이를 변경할 수 있으므로 이러한 동적 위치 계산이 필요합니다.

필드 크기에 맞게 글꼴 크기 변경

Flash Player 9 이상, Adobe AIR 1.0 이상

폭(픽셀 단위)과 표시할 행의 최대 개수가 주어지면 HeadlineTextField는 텍스트가 필드에 맞도록 글꼴 크기를 변경합니다. 텍스트가 짧으면 글꼴 크기가 커져 타블로이드 스타일의 헤드라인이 만들어지고, 텍스트가 길면 글꼴 크기가 작아집니다.

아래에 나와 있는 HeadlineTextField.fitText() 메서드는 글꼴 크기 조정 작업을 수행합니다.

```
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;
    }
}
```

HeadlineTextField.fitText() 메서드는 간단한 재귀 기술을 사용하여 글꼴 크기를 조정합니다. 먼저 텍스트의 문자당 평균 픽셀 수를 추정하여 시작점 크기를 계산합니다. 그런 다음 글꼴 크기를 변경하고 텍스트가 최대 텍스트 행 수를 초과하도록 줄 바꿈되었는지 확인합니다. 행이 너무 많은 경우에는 shrinkText() 메서드를 호출하여 글꼴 크기를 줄인 다음 다시 시도합니다. 행의 수가 너무 많지 않은 경우에는 growText() 메서드를 호출하여 글꼴 크기를 늘린 다음 다시 시도합니다. 글꼴 크기를 한 포인트 더 늘리면 행 수가 제한을 초과하게 되는 지점에서 이러한 프로세스가 중단됩니다.

여러 열에 텍스트 분할

Flash Player 9 이상, Adobe AIR 1.0 이상

MultiColumnTextField 클래스는 여러 TextField 객체에 텍스트를 분산하며 이러한 객체는 신문의 열처럼 배열됩니다.

MultiColumnTextField() 생성자는 먼저 다음과 같이 각 열에 TextField 객체가 하나씩 해당하는 배열을 만듭니다.

```
    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);
    }
```

addChild() 메서드를 사용하여 각 TextField 객체가 배열에 추가되고 표시 목록에 추가됩니다.

StoryLayout text 속성 또는 styleSheet 속성이 변경될 때마다 layoutColumns() 메서드를 호출하여 텍스트를 다시 표시합니다. layoutColumns() 메서드는 getOptimalHeight() 메서드를 호출하여 주어진 레이아웃 폭 내에 모든 텍스트를 맞추기 위해 필요한 올바른 픽셀 높이를 계산합니다.

```
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;
    }
}
```

먼저 getOptimalHeight() 메서드가 각 열의 폭을 계산합니다. 그런 다음 배열 내 첫 번째 TextField 객체의 폭과 htmlText 속성을 설정합니다. getOptimalHeight() 메서드가 이 첫 번째 TextField 객체를 사용하여 텍스트에서 줄 바꿈된 행의 총 수를 확인하고 이를 통해 각 열에 포함해야 하는 행의 수를 파악합니다. 그런 다음 TextField.getLineMetrics() 메서드를 호출하여 첫 번째 행의 텍스트 크기에 대한 세부 정보가 포함된 TextLineMetrics 객체를 검색합니다. TextLineMetrics.height 속성은 어센트, 디센트 및 행간을 포함한 텍스트 한 행의 전체 높이를 픽셀 단위로 나타냅니다. 그러면 MultiColumnTextField 객체의 최적 높이는 행 높이와 한 열의 행 수를 곱한 값에 4(TextField 객체의 위쪽 및 아래쪽 테두리 2픽셀씩)를 더한 값이 됩니다.

다음은 전체 layoutColumns() 메서드의 코드입니다.

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

preferredHeight 속성이 설정(getOptimalHeight() 메서드 호출)된 후 여러 TextField 객체에 layoutColumns() 메서드가 반복되어 각각의 높이가 preferredHeight 값으로 설정됩니다. 그런 다음 layoutColumns() 메서드가 각 필드에 적정한 수의 텍스트 행을 분산하여 개별 필드에서 스크롤이 발생하지 않고 이전 필드의 텍스트가 끝난 지점에서 각 다음 필드의 텍스트가 시작될 수 있도록 합니다. 텍스트 정렬 스타일이 "justify"로 설정된 경우에는 justifyLastLine() 메서드가 호출되어 텍스트의 마지막 행이 필드에 양쪽 정렬됩니다. 텍스트 정렬 스타일이 "justify"로 설정되지 않은 경우에는 마지막 행이 단락의 마지막 행으로 간주되고 양쪽 정렬되지 않습니다.

22장: Flash Text Engine 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Player 10 및 Adobe® AIR™1.5부터 사용할 수 있는 Adobe® Flash® Text Engine(FTE)은 저수준에서 텍스트 메트릭, 서식 및 양방향 텍스트를 정교하게 제어할 수 있으며 향상된 텍스트 방향 및 언어 지원을 제공합니다. FTE는 단순 텍스트 요소를 만들고 관리하는 데 사용할 수 있지만 기본적으로 개발자가 텍스트 처리 구성 요소를 만들기 위한 기반으로 설계되었습니다. 따라서 Flash Text Engine에서는 사용자가 높은 수준의 프로그래밍 전문 기술을 갖추고 있다고 가정합니다. 간단한 텍스트 요소를 표시하려면 336페이지의 “TextField 클래스 사용”을 참조하십시오.

FTE 기반의 텍스트 처리 구성 요소를 포함하는 Text Layout Framework는 해당 고급 기능을 사용하기 위한 보다 쉬운 방법을 제공합니다. Text Layout Framework는 ActionScript 3.0에 완전하게 구축되어 있는 확장 가능한 라이브러리입니다. 기존 TLF 구성 요소를 사용하거나, 프레임워크를 사용하여 고유한 텍스트 구성 요소를 구축할 수 있습니다. 자세한 내용은 386페이지의 “Text Layout Framework 사용”을 참조하십시오.

기타 도움말 항목

flash.text.engine 패키지

텍스트 생성 및 표시

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Text Engine을 구성하는 클래스를 사용하여 텍스트를 만들고 제어하며 텍스트에 서식을 지정할 수 있습니다. 다음 클래스는 Flash Text Engine을 사용하여 텍스트를 만들고 표시하기 위한 기본 구성 블록입니다.

- TextElement/GraphicElement/GroupElement - TextBlock 인스턴스의 내용 포함
- ElementFormat - TextBlock 인스턴스 내용의 서식 특성 지정
- TextBlock - 텍스트 단락을 구성하는 팩토리
- TextLine - TextBlock에서 만들어진 텍스트 행

텍스트를 표시하려면 String에서 TextElement 객체를 만들고 ElementFormat 객체를 사용하여 서식 지정 특성을 지정합니다. TextElement를 TextBlock 객체의 content 속성에 할당합니다. TextBlock.createTextLine() 메서드를 호출하여 표시할 텍스트 행을 만듭니다. createTextLine() 메서드가 지정된 폭에 알맞은 길이의 문자열을 포함하는 TextLine 객체를 반환합니다. 전체 문자열이 여러 행으로 서식이 지정될 때까지 반복해서 메서드를 호출합니다. 더 이상 만들 행이 없으면 TextBlock 객체의 textLineCreationResult 속성에 TextLineCreationResult.COMPLETE 값이 할당됩니다. 행을 표시하려면 적절한 x 및 y 위치 값을 사용하여 표시 목록에 행을 추가합니다.

예를 들어 다음 코드는 이러한 FTE 클래스를 사용하여 기본 서식 및 글꼴 값으로 "Hello World! This is Flash Text Engine!"을 표시합니다. 이 간단한 예제에서는 텍스트 행이 하나만 만들어집니다.

```
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;
        }
    }
}
```

createTextLine()의 매개 변수는 새로운 행을 시작할 행과 행의 폭(픽셀 단위)을 지정합니다. 새로운 행을 시작할 행은 일반적으로 이전 행이지만 첫 번째 행의 경우에는 null이 됩니다.

GraphicElement 및 GroupElement 객체 추가

Flash Player 10 이상, Adobe AIR 1.5 이상

GraphicElement 객체를 TextBlock 객체에 할당하여 이미지 또는 그래픽 요소를 표시할 수 있습니다. 그래픽 또는 이미지에서 GraphicElement 클래스의 인스턴스를 만들어 TextBlock.content 속성에 해당 인스턴스를 할당합니다. 일반적인 방식으로 TextBlock.createTextline()을 호출하여 텍스트 행을 만듭니다. 다음 예제에서는 두 개의 텍스트 행, 즉 GraphicElement 객체를 사용하는 텍스트 행 하나와 TextElement 객체를 사용하는 텍스트 행 하나를 만듭니다.

```
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;
        }
    }
}
```

GroupElement 객체를 만들어 TextElement, GraphicElement 및 기타 GroupElement 객체의 그룹을 만들 수 있습니다. GroupElement는 TextBlock 객체의 content 속성에 할당할 수 있습니다. GroupElement() 생성자에 대한 매개 변수는 Vector로, 해당 그룹을 구성하는 텍스트, 그래픽 및 그룹 요소를 가리킵니다. 다음 예제에서는 두 개의 그래픽 요소와 하나의 텍스트 요소를 그룹화하여 하나의 단위로 텍스트 블록에 할당합니다.

```
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;
        }
    }
}
```

텍스트 대체

Flash Player 10 이상, Adobe AIR 1.5 이상

TextElement.replaceText()를 호출하여 TextBlock.content 속성에 할당한 TextElement의 텍스트를 바꿈으로써 TextBlock 인스턴스의 텍스트를 대체할 수 있습니다.

다음 예제에서는 replaceText()를 사용하여 먼저 행 시작 부분에 텍스트를 삽입한 후 행 끝 부분에 텍스트를 추가하고, 마지막으로 행 중간의 텍스트를 바꿉니다.

```
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);
        }
    }
}
```

replaceText() 메서드는 beginIndex 및 endIndex 매개 변수로 지정된 텍스트를 newText 매개 변수로 지정된 텍스트로 바꿉니다. beginIndex 및 endIndex 매개 변수의 값이 동일한 경우 replaceText()가 지정된 텍스트를 해당 위치에 삽입하고, 값이 동일하지 않은 경우에는 beginIndex 및 endIndex로 지정된 문자를 새 텍스트로 바꿉니다.

FTE에서 이벤트 처리

Flash Player 10 이상, Adobe AIR 1.5 이상

다른 표시 객체와 마찬가지로 TextLine 인스턴스에도 이벤트 리스너를 추가할 수 있습니다. 예를 들어 사용자가 텍스트 행에 마우스를 롤 오버하거나 행을 클릭하는 경우 감지할 수 있습니다. 다음 예제에서는 이러한 두 가지 이벤트를 모두 감지합니다. 행에 마우스를 롤 오버하면 커서가 버튼 커서로 변경되고 행을 클릭하면 커서 색상이 바뀝니다.

```
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);
        }
    }
}
```

이벤트 미러링

Flash Player 10 이상, Adobe AIR 1.5 이상

또한 텍스트 블록의 이벤트 또는 텍스트 블록 일부의 이벤트를 이벤트 디스패처로 미러링할 수도 있습니다. 먼저 EventDispatcher 인스턴스를 만든 다음 TextElement 인스턴스의 eventMirror 속성에 해당 인스턴스를 할당합니다. 텍스트 블록이 하나의 텍스트 요소로 구성된 경우에는 텍스트 엔진에서 전체 텍스트 블록에 대한 이벤트를 미러링하고, 텍스트 블록이 여러 텍스트 요소로 구성된 경우에는 eventMirror 속성이 설정된 TextElement 인스턴스에 대한 이벤트만 미러링합니다. 다음 예제의 텍스트는 단어 "Click"과 "here", 문자열 "to see me in italic" 등 세 가지 요소로 구성됩니다. 이 예제에서는 이벤트 디스패처를 두 번째 텍스트 요소인 단어 "here"에 할당하고 이벤트 리스너인 clickHandler() 메서드를 추가합니다. clickHandler() 메서드는 해당 텍스트를 기울임체로 변경하고, 세 번째 텍스트 요소의 내용을 "Click here to see me in normal font!"로 바꿉니다.

```
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);
        }
    }
}
```

mouseOverHandler() 및 mouseOutHandler() 함수는 커서가 단어 "here" 위에 있을 때는 버튼 커서로 변경되고, 그렇지 않을 때는 다시 화살표로 돌아가도록 설정합니다.

텍스트 서식 지정

Flash Player 10 이상, Adobe AIR 1.5 이상

TextBlock 객체는 텍스트 행을 만드는 팩토리입니다. TextBlock의 내용은 TextElement 객체를 통해 할당됩니다. ElementFormat 객체는 텍스트의 서식 지정을 처리합니다. ElementFormat 클래스는 기준선 정렬, 커닝, 자간, 텍스트 회전 및 글꼴 크기, 색상, 대/소문자 등의 속성을 정의합니다. 이 클래스에는 FontDescription도 포함되며 이에 대한 설명은 370페이지의 "글꼴을 사용한 작업"에 자세히 나와 있습니다.

ElementFormat 객체 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

ElementFormat 객체의 생성자는 FontDescription을 비롯해 다양한 선택적 매개 변수를 모두 사용할 수 있습니다. 이러한 속성을 생성자 외부에서 설정할 수도 있습니다. 다음 예제에서는 단순 텍스트 행을 정의 및 표시하는 경우의 다양한 객체의 관계를 보여 줍니다.

```
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);
        }
    }
}
```

글꼴 색상 및 투명도(알파)

Flash Player 10 이상, Adobe AIR 1.5 이상

color 속성(ElementFormat 객체)은 글꼴 색상을 설정합니다. 값은 해당 색상의 RGB 구성 요소를 나타내는 정수입니다(예: 빨강의 경우 0xFF0000, 녹색의 경우 0x00FF00). 기본값은 검정입니다(0x000000).

alpha 속성은 요소(TextElement 및 GraphicElement 모두 해당)의 알파 투명도 값을 설정합니다. 값은 0(완전히 투명)과 1(완전히 불투명, 기본값) 사이의 값이 될 수 있습니다. alpha 값이 0인 요소는 보이지는 않지만 활성 상태입니다. 이 값에 상속된 알파 값이 곱해져 해당 요소가 더 투명해집니다.

```
var ef:ElementFormat = new ElementFormat();
ef.alpha = 0.8;
ef.color = 0x999999;
```

기준선 정렬 및 이동

Flash Player 10 이상, Adobe AIR 1.5 이상

한 행에서 가장 큰 텍스트의 글꼴 및 크기에 따라 주요 기준선이 결정됩니다. TextBlock.baselineFontDescription 및 TextBlock.baselineFontSize를 설정하여 이러한 값을 재정의할 수 있습니다. 텍스트 내 여러 기준선 중 하나를 기준으로 주요 기준선을 정렬할 수 있습니다. 이러한 기준선에는 어센트 선과 디센트 선이나 표의 문자 위쪽, 가운데 또는 아래쪽이 포함됩니다.

A. 어센트 **B.** 기준선 **C.** 디센트 **D.** x 높이

ElementFormat 객체에서는 세 가지 속성이 기준선과 정렬 특성을 결정합니다. alignmentBaseline 속성은 TextElement 또는 GraphicElement의 기본 기준선을 설정합니다. 이 기준선은 요소의 "물리기" 선이며 모든 텍스트의 주요 기준선이 정렬되는 기준 위치입니다.

dominantBaseline 속성은 요소의 다양한 기준선 중 사용할 기준선을 지정하며 이 기준선은 해당 행에 있는 요소의 세로 위치를 결정합니다. 기본값은 TextBaseline.ROMAN이지만 IDEOGRAPHIC_TOP 또는 IDEOGRAPHIC_BOTTOM 기준선이 주요 기준선이 되도록 설정할 수도 있습니다.

baselineShift 속성은 설정된 픽셀 수만큼 y 축에서 기준선을 이동합니다. 회전하지 않은 보통 텍스트에서 양수 값을 설정하면 기준선이 아래로 이동하고, 음수 값을 설정하면 위로 이동합니다.

입력 체계 대/소문자

Flash Player 10 이상, Adobe AIR 1.5 이상

TypographicCase 속성(ElementFormat)은 대문자, 소문자 또는 작은 대문자와 같은 텍스트 대/소문자를 지정합니다.

```
var ef_Upper:ElementFormat = new ElementFormat();
ef_Upper.typographicCase = TypographicCase.UPPERCASE;

var ef_SmallCaps:ElementFormat = new ElementFormat();
ef_SmallCaps.typographicCase = TypographicCase.SMALL_CAPS;
```

텍스트 회전

Flash Player 10 이상, Adobe AIR 1.5 이상

텍스트 한 블록이나 텍스트의 한 세그먼트 내 글리프를 90도 단위로 회전할 수 있습니다. TextRotation 클래스는 다음과 같은 상수를 정의하여 텍스트 블록 및 글리프 회전을 설정합니다.

상수	값	설명
AUTO	“auto”	시계 반대 방향으로 90도 회전을 지정합니다. 일반적으로 세로 방향의 아시아 언어 텍스트에서 회전이 필요한 글리프만 회전하는 데 사용됩니다.
ROTATE_0	“rotate_0”	회전을 지정하지 않습니다.
ROTATE_180	“rotate_180”	180도 회전을 지정합니다.
ROTATE_270	“rotate_270”	270도 회전을 지정합니다.
ROTATE_90	“rotate_90”	시계 방향으로 90도 회전을 지정합니다.

텍스트 블록의 텍스트 행을 회전하려면 TextBlock.lineRotation 속성을 설정한 후 TextBlock.createTextLine() 메서드를 호출하여 텍스트 행을 만듭니다.

한 텍스트 블록 또는 세그먼트 내에 있는 글리프를 회전하려면 ElementFormat.textRotation 속성을 원하는 글리프 회전 각도로 설정합니다. 글리프는 문자를 형성하는 모양 또는 여러 글리프로 구성되는 문자의 일부입니다. 예를 들어 "i"의 점, 문자 "a" 등이 글리프입니다.

글리프 회전은 행을 수직으로 회전하되 행 내에 있는 문자는 회전하지 않으려는 일부 아시아 언어에서 관련이 있습니다. 아시아 언어 텍스트를 회전하는 방법에 대한 자세한 내용은 374페이지의 “동아시아 언어 텍스트 양쪽 정렬”을 참조하십시오.

다음 예제에서는 아시아 언어 텍스트와 마찬가지로 텍스트 블록과 그 안의 글리프를 모두 회전합니다. 또한 예제에서는 일본어 글꼴을 사용합니다.

```
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 잠금 및 복제

Flash Player 10 이상, Adobe AIR 1.5 이상

ElementFormat 객체가 ContentElement의 한 유형에 할당되면 해당 locked 속성이 true로 자동 설정됩니다. 잠긴 ElementFormat 객체를 수정하려고 하면 IllegalOperationError가 발생합니다. 따라서 이러한 객체를 완전히 정의한 후 TextElement 인스턴스에 할당하는 것이 가장 좋습니다.

기존 ElementFormat 인스턴스를 수정하려면 먼저 locked 속성을 확인합니다. 해당 속성이 true인 경우 clone() 메서드를 사용하여 객체의 잠금 해제된 복사본을 만듭니다. 이 잠금 해제된 객체의 속성은 변경할 수 있으며, 그런 다음 TextElement 인스턴스에 할당할 수 있습니다. 해당 객체에서 만든 모든 새 행에는 새 서식이 적용됩니다. 이 동일한 객체에서 이전 서식을 사용하여 만든 이전 행은 변경되지 않습니다.

```
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);
        }
    }
}
```

글꼴을 사용한 작업

Flash Player 10 이상, Adobe AIR 1.5 이상

FontDescription 객체는 ElementFormat과 함께 사용하여 글꼴을 식별하고 일부 특성을 정의합니다. 이러한 특성에는 글꼴 이름, 두께, 포스처, 렌더링 및 글꼴 검색 방법(장치 글꼴 또는 포함된 글꼴)이 포함됩니다.

참고: FTE는 Type 1 글꼴이나 Type 3, ATC, sfnt-wrapped CID 또는 Naked CID와 같은 비트맵 글꼴을 지원하지 않습니다.

글꼴 특성 정의(FontDescription 객체)

Flash Player 10 이상, Adobe AIR 1.5 이상

fontName 속성(FontDescription 객체)은 하나의 이름이거나 여러 이름의 쉼표로 분리된 목록일 수 있습니다. 예를 들어 "Arial, Helvetica, _sans"와 같은 목록에서 텍스트 엔진은 먼저 "Arial"을 검색한 다음 "Helvetica"를 검색하고 처음 두 글꼴을 모두 찾을 수 없는 경우 마지막으로 "_sans"를 검색합니다. 글꼴 이름 집합에는 "_sans", "_serif", "_typewriter" 등 세 가지 일반 장치 글꼴 이름이 포함됩니다. 이러한 이름은 재생 시스템에 따라 특정 장치 글꼴에 매핑됩니다. 장치 글꼴을 사용하는 모든 글꼴 설명에 이와 같은 기본 이름을 지정하는 것이 좋습니다. fontName을 지정하지 않은 경우 "_serif"가 기본값으로 사용됩니다.

fontPosture 속성은 기본값(FontPosture.NORMAL) 또는 기울임체(FontPosture.ITALIC)로 설정할 수 있습니다. fontWeight 속성은 기본값(FontWeight.NORMAL) 또는 굵게(FontWeight.BOLD)로 설정할 수 있습니다.

```
var fd1:FontDescription = new FontDescription();
fd1.fontName = "Arial, Helvetica, _sans";
fd1.fontPosture = FontPosture.NORMAL;
fd1.fontWeight = FontWeight.BOLD;
```

포함된 글꼴과 장치 글꼴

Flash Player 10 이상, Adobe AIR 1.5 이상

fontLookup 속성(FontDescription 객체)은 텍스트 엔진이 텍스트를 렌더링하기 위해 장치 글꼴을 검색할지 아니면 포함된 글꼴을 검색할지 지정합니다. 장치 글꼴(FontLookup.DEVICE)을 지정한 경우 런타임이 재생 시스템에서 해당 글꼴을 검색합니다. 포함된 글꼴(FontLookup.EMBEDDED_CFF)을 지정하면 SWF 파일에서 지정된 이름을 가진 포함된 글꼴을 검색합니다. 이 설정은 포함된 CFF(Compact Font Format) 글꼴에서만 사용할 수 있습니다. 지정된 글꼴을 찾을 수 없는 경우에는 대체 장치 글꼴이 사용됩니다.

장치 글꼴을 사용하면 SWF 파일 크기가 작아지고, 포함된 글꼴을 사용하면 모든 플랫폼에서 정확도가 향상됩니다.

```
var fd1:FontDescription = new FontDescription();
fd1.fontLookup = FontLookup.EMBEDDED_CFF;
fd1.fontName = "Garamond, _serif";
```

렌더링 모드 및 힌팅

Flash Player 10 이상, Adobe AIR 1.5 이상

CFF(Compact Font Format) 렌더링은 Flash Player 10 및 Adobe AIR 1.5부터 사용할 수 있습니다. 이 유형의 글꼴 렌더링을 사용하면 텍스트를 보다 쉽게 읽을 수 있고 작은 크기에서도 글꼴의 표시 품질이 향상됩니다. 이 설정은 포함된 글꼴에만 적용됩니다. FontDescription은 기본적으로 이 설정(RenderingMode.CFF)이 renderingMode 속성에 대해 지정됩니다. 이 속성을 Flash Player 7 또는 이전 버전에서 사용되는 렌더링 유형과 일치하도록 RenderingMode.NORMAL로 설정할 수 있습니다.

CFF 렌더링을 선택하는 경우 두 번째 속성 cffHinting이 글꼴의 가로 줄이 하위 픽셀 격자에 맞는 정도를 제어합니다. 기본값 CFFHinting.HORIZONTAL_STEM은 CFF 힌팅을 사용합니다. 이 속성을 CFFHinting.NONE으로 설정하면 힌팅이 제거되므로 애니메이션이나 큰 글꼴 크기에 적합합니다.

```
var fd1:FontDescription = new FontDescription();
fd1.renderingMode = RenderingMode.CFF;
fd1.cffHinting = CFFHinting.HORIZONTAL_STEM;
```

FontDescription 잠금 및 복제

Flash Player 10 이상, Adobe AIR 1.5 이상

FontDescription 객체가 ElementFormat에 할당되면 해당 locked 속성이 true로 자동 설정됩니다. 잠긴 FontDescription 객체를 수정하려고 하면 IllegalOperationError가 발생합니다. 따라서 이러한 객체를 완전히 정의한 후 ElementFormat에 할당하는 것이 가장 좋습니다.

기존 FontDescription을 수정하려면 먼저 locked 속성을 확인합니다. 해당 속성이 true인 경우 clone() 메서드를 사용하여 객체의 잠금 해제된 복사본을 만듭니다. 이 잠금 해제된 객체의 속성은 변경할 수 있으며, 그런 다음 ElementFormat에 할당할 수 있습니다. 이 TextElement에서 만든 모든 새 행에는 새 서식이 적용됩니다. 이 동일한 객체에서 만든 이전 행은 변경되지 않습니다.

```
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);
        }
    }
}
```

텍스트 제어

Flash Player 10 이상, Adobe AIR 1.5 이상

FTE를 사용하면 새로운 텍스트 서식 컨트롤 집합을 통해 양쪽 정렬 및 문자 간격(커닝 및 자간)을 처리할 수 있습니다. 또한 행 분리 방식을 제어하고 행 내에서 탭 정지를 설정하는 속성도 있습니다.

텍스트 양쪽 정렬

Flash Player 10 이상, Adobe AIR 1.5 이상

텍스트 양쪽 정렬은 단어 또는 문자 사이의 간격을 조정하여 단락 내 모든 행의 길이를 동일하게 만듭니다. 그 결과, 텍스트가 양쪽 모두에 정렬되는 반면 단어 및 문자 사이의 간격은 달라집니다. 신문 및 잡지의 텍스트 열에는 양쪽 정렬이 적용되는 경우가 많습니다.

SpaceJustifier 클래스의 lineJustfication 속성을 사용하여 텍스트 블록에 포함된 행의 양쪽 정렬을 제어할 수 있습니다. LineJustification 클래스는 양쪽 정렬 옵션을 지정하는 데 사용할 수 있는 상수를 정의합니다. ALL_BUT_LAST는 마지막 행을 제외한 모든 텍스트를 양쪽 정렬하고, ALL_INCLUDING_LAST는 마지막 행을 포함한 모든 텍스트를 양쪽 정렬하며, 기본값인 UNJUSTIFIED는 텍스트를 양쪽 정렬하지 않습니다.

텍스트를 양쪽 정렬하려면 SpaceJustifier 클래스의 한 인스턴스에 lineJustification 속성을 설정한 다음 해당 인스턴스를 TextBlock 인스턴스의 textJustifier 속성에 할당합니다. 다음 예제에서는 마지막 행을 제외한 모든 텍스트가 양쪽 정렬된 단락을 만듭니다.

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

단어 사이뿐 아니라 문자 사이의 간격을 다르게 하려면 SpaceJustifier.letterspacing 속성을 true로 설정합니다. 문자 간격을 설정하면 단순 양쪽 정렬 사용 시 발생할 수 있는 단어 사이의 보기 좋지 않은 간격 발생이 줄어들 수 있습니다.

동아시아 언어 텍스트 양쪽 정렬

Flash Player 10 이상, Adobe AIR 1.5 이상

동아시아 언어 텍스트 양쪽 정렬에는 추가 고려 사항이 있습니다. 동아시아 언어 텍스트는 위쪽에서 아래쪽으로 쓸 수 있으며, 계속해서 문자 처리해야 하는 일부 문자(kinsoku)는 행의 시작 부분이나 끝 부분에 올 수 없습니다. JustificationStyle 클래스는 이러한 문자를 처리하는 옵션을 지정하는 다음 상수를 정의합니다. PRIORITIZE_LEAST_ADJUSTMENT는 어떤 것이 가장 바람직한 결과를 생성하는지에 따라 행 확장 또는 압축 방식을 사용하여 양쪽 정렬합니다. PUSH_IN_KINSOKU는 행의 끝 부분에서 계속해서 문자 처리해야 하는 문자를 압축하여 양쪽 정렬합니다. 계속해서 문자 처리해야 하는 문자가 없거나 해당 공간이 충분하지 않은 경우에는 행을 확장하여 양쪽 정렬합니다.

PUSH_OUT_ONLY는 행을 확장하여 양쪽 정렬합니다. 세로 방향 아시아 언어 텍스트 블록을 만들려면 TextBlock.lineRotation 속성을 TextRotation.ROTATE_90로 설정하고 ElementFormat.textRotation 속성을 기본값인 TextRotation.AUTO로 설정합니다. textRotation 속성을 AUTO로 설정하면 행을 회전할 때 텍스트의 글리프가 옆으로 회전하지 않고 세로로 유지됩니다. AUTO 설정을 사용하면 글리프의 유니코드 속성에 따라 전체 폭 글리프 및 넓은 글리프만 시계 반대 방향으로 90도 회전합니다. 다음 예제에서는 일본어 텍스트의 세로 블록을 표시하고 PUSH_IN_KINSOKU 옵션을 사용하여 양쪽 정렬합니다.

```
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
```

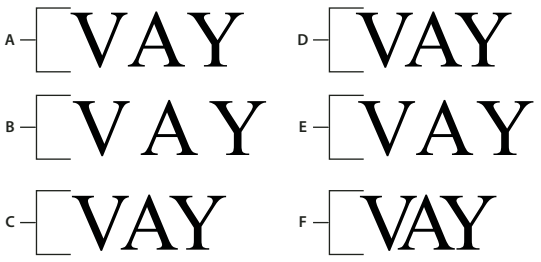
```
                font.fontName = String.fromCharCode(0x5C0F, 0x585A, 0x660E, 0x671D) + " Pro R";
            else
                font.fontName = "Kozuka Mincho Pro R";
            textBlock.content = new TextElement(Japanese_txt, format);
            var previousLine:TextLine = null;

            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;
            }
        }
    }
}
```

커닝 및 자간

Flash Player 10 이상, Adobe AIR 1.5 이상

커닝 및 자간은 텍스트 블록에서 인접한 문자 쌍 사이의 거리에 영향을 미칩니다. 커닝은 "WA" 또는 "Va"와 같은 문자 쌍을 서로 "맞추는" 방식을 제어합니다. 커닝은 ElementFormat 객체에서 설정됩니다. 커닝은 기본적으로 활성화되며(Kerning.ON) OFF 또는 AUTO로 설정할 수 있습니다. OFF 또는 AUTO로 설정하면 간지, 히라가나 또는 가타카나가 아닌 경우에만 문자 사이에 커닝이 적용됩니다.

자간은 텍스트 블록 내 모든 문자 사이에서 설정된 수의 픽셀을 더하거나 뺍니다. 자간도 ElementFormat 객체에서 설정됩니다. 포함된 글꼴과 장치 글꼴에서 모두 사용할 수 있습니다. FTE는 두 가지 자간 속성, 즉 문자의 왼쪽에서 픽셀을 더하거나 빼는 trackingLeft와 오른쪽에서 픽셀을 더하거나 빼는 trackingRight를 지원합니다. 커닝을 사용하는 경우 자간 값은 각 문자 쌍의 커닝 값에서 더해지거나 빼집니다.



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;
```

줄 바꿈된 텍스트의 행 분리

Flash Player 10 이상, Adobe AIR 1.5 이상

breakOpportunity 속성(ElementFormat 객체)은 줄 바꿈하는 텍스트가 여러 행으로 나뉘는 경우 행 분리에 사용할 수 있는 문자를 결정합니다. 기본값 BreakOpportunity.AUTO는 단어와 하이픈 사이를 분리하는 등 표준 유니코드 속성을 사용합니다. BreakOpportunity.ALL을 사용하면 모든 문자가 행 분리 기회로 간주될 수 있으므로 경로를 따라 표시되는 텍스트와 같은 효과를 만드는 데 유용합니다.

```
var ef:ElementFormat = new ElementFormat();
ef.breakOpportunity = BreakOpportunity.ALL;
```

탭 정지

Flash Player 10 이상, Adobe AIR 1.5 이상

텍스트 블록에서 탭 정지를 설정하려면 TabStop 클래스의 인스턴스를 만들어 탭 정지를 정의합니다. TabStop() 생성자에 대한 매개 변수는 텍스트가 탭 정지와 정렬되는 방식을 지정합니다. 이러한 매개 변수는 탭 정지의 위치를 지정하고, 소수점 정렬의 경우 문자열로 표현되는 정렬 기준 값을 지정합니다. 일반적으로 이 값은 소수점이지만 쉼표, 달러 기호 또는 엔이나 유로 심볼 등이 될 수도 있습니다. 다음 코드 행에서는 tab1이라는 탭 정지를 만듭니다.

```
var tab1:TabStop = new TabStop(TabAlignment.DECIMAL, 50, ".");
```

텍스트 블록의 탭 정지를 만든 후 TextBlock 인스턴스의 tabStops 속성에 해당 탭 정지를 할당합니다. 그러나 tabStops 속성에는 Vector가 필요하므로 먼저 Vector를 만들고 탭 정지를 해당 Vector에 추가해야 합니다. Vector를 사용하면 텍스트 블록에 탭 정지 집합을 할당할 수 있습니다. 다음 예제에서는 Vector<TabStop> 인스턴스를 만들고 TabStop 객체의 집합을 해당 인스턴스에 추가합니다. 그런 다음 탭 정지를 TextBlock 인스턴스의 tabStops 속성에 할당합니다.

```
var tabStops:Vector.<TabStop> = new Vector.<TabStop>();
tabStops.push(tab1, tab2, tab3, tab4);
textBlock.tabStops = tabStops
```

Vector에 대한 자세한 내용은 22페이지의 “배열을 사용한 작업”을 참조하십시오.

다음 예제에서는 각 TabStop 정렬 옵션의 효과를 보여 줍니다.

```
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;
            }
        }
    }
}
```

Flash Text Engine 예제: 뉴스 레이아웃

Flash Player 10 이상, Adobe AIR 1.5 이상

이 프로그래밍 예제에서는 Flash Text Engine을 사용하여 간단한 신문 페이지의 레이아웃을 지정하는 방법을 보여 줍니다. 페이지에는 큰 헤드라인, 소제목, 여러 열로 구성된 본문 섹션이 포함됩니다.

먼저 FLA 파일을 만들고 다음 코드를 기본 레이어의 프레임 #2에 연결합니다.

```
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.as는 이 예제의 컨트롤러 스크립트입니다. 내용을 설정하고 외부 스타일 시트에서 스타일 정보를 읽어 ElementFormat 객체에 해당 스타일을 할당합니다. 그런 다음 헤드라인, 소제목 및 여러 열로 구성된 텍스트 요소를 만듭니다.

```
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.as는 텍스트 블록을 만드는 기본 클래스로 사용됩니다. 여기에는 글꼴 크기와 대/소문자를 변경하는 유틸리티 함수도 포함되어 있습니다.

```
package com.example.programmingas3.newslayout
{
    import flash.text.engine.*;
    import flash.display.Sprite;

    public class FormattedTextBlock extends Sprite
    {
        public var tb:TextBlock;
        private var te:TextElement;
        private var ef1:ElementFormat;
        private var textWidth:int;
        public var totalTextLines:int;
        public var blockText:String;
        public var leading:Number = 1.25;
        public var preferredWidth:Number = 720;
        public var preferredHeight:Number = 100;

        public function FormattedTextBlock(ef:ElementFormat,txt:String, colW:int = 0)
        {
            this.textWidth = (colW==0) ? preferredWidth : colW;
            blockText = txt;
            ef1 = ef;
            tb = new TextBlock();
            tb.textJustifier = new SpaceJustifier("en",LineJustification.UNJUSTIFIED,false);
            te = new TextElement(blockText,this.ef1);
            tb.content = te;
            this.breakLines();
        }

        private function breakLines()
        {
            var textLine:TextLine = null;
            var y:Number = 0;
            var lineNum:int = 0;
            while (textLine = tb.createTextLine(textLine,this.textWidth,0,true))
            {
                textLine.x = 0;
                textLine.y = y;
                y += this.leading*textLine.height;
                this.addChild(textLine);
            }
            for (var i:int = 0; i < this.numChildren; i++)
            {
                TextLine(this.getChildAt(i)).validity = TextLineValidity.STATIC;
            }
            this.totalTextLines = this.numChildren;
        }

        private function rebreakLines()
        {
            this.clearLines();
            this.breakLines();
        }

        private function clearLines()
        {
            while(this.numChildren)
            {
                this.removeChildAt(0);
            }
        }
```

```
        public function changeSize(size:uint=12):void
        {
            if (size > 5)
            {
                var ef2:ElementFormat = ef1.clone();
                ef2.fontSize = size;
                te.elementFormat = ef2;
                this.rebreakLines();
            }
        }

        public function changeCase(newCase:String = "default"):void
        {
            var ef2:ElementFormat = ef1.clone();
            ef2.typographicCase = newCase;
            te.elementFormat = ef2;
        }
    }
}
```

HeadlineTextBlock.as는 FormattedTextBlock 클래스를 확장하며 헤드라인을 만드는 데 사용됩니다. 여기에는 페이지에서 정의된 공간 내에 텍스트를 맞추는 함수가 포함되어 있습니다.

```
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.as는 여러 열로 구성된 디자인에서 텍스트 서식 지정을 처리합니다. TextBlock 객체를 텍스트 행의 생성, 서식 지정 및 배치를 위한 팩토리로 유연하게 사용할 수 있음을 보여 줍니다.

```
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);
        }

    }
}
```

23장: Text Layout Framework 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

Text Layout Framework 개요

Flash Player 10 이상, Adobe AIR 1.5 이상

TLF(Text Layout Framework)는 확장 가능한 ActionScript 라이브러리입니다. TLF는 Adobe® Flash® Player 10 및 Adobe® AIR® 1.5의 텍스트 엔진에 내장되어 있습니다. TLF는 웹에서 고급 입력 체계와 혁신적인 입력 체계의 텍스트 레이아웃 기능을 제공합니다. 이 프레임워크는 Adobe® Flex® 또는 Adobe® Flash® Professional과 함께 사용할 수 있습니다. 개발자는 기존의 구성 요소를 사용하거나 확장할 수 있을 뿐 아니라 이 프레임워크를 활용하여 자신만의 텍스트 구성 요소를 만들 수 있습니다.

TLF의 기능은 다음과 같습니다.

- 양방향 텍스트, 세로 방향 텍스트, 30개 이상의 쓰기 스크립트(아랍어, 히브리어, 중국어, 일본어, 한국어, 태국어, 라오스어, 베트남어 등)
- 여러 열 및 연결된 컨테이너에 걸친 텍스트의 선택, 편집 및 흐름
- 세로 방향 텍스트, 문자 회전(세로 방향 텍스트 내 가로 방향), 동아시아 언어 입력 체계에 대한 정렬자
- 커닝, 합자, 입력 체계 대/소문자, 숫자 케이스, 숫자 폭, 임의 하이픈 등의 풍부한 입력 체계 컨트롤
- 편집을 위한 잘라내기, 복사, 붙여넣기, 실행 취소, 표준 키보드 및 마우스 동작
- 텍스트 내용, 레이아웃 및 마크업을 조작하고 사용자 정의 텍스트 구성 요소를 만들 수 있는 풍부한 개발자 API
- 사용자 정의 표시자 및 번호 매기기 형식 등의 강력한 목록 지원
- 인라인 이미지 및 위치 지정 규칙

TLF는 Flash Player 10에 도입된 FTE(Flash Text Engine)의 내장 ActionScript 3.0 라이브러리입니다. FTE는 Flash Player 10 API(응용 프로그래밍 인터페이스)의 일부인 flash.text.engine 패키지를 통해 액세스할 수 있습니다.

그러나 Flash Player API에서는 텍스트 엔진에 대해 저수준의 액세스만 지원하므로 일부 작업의 경우 상대적으로 매우 많은 양의 코드가 필요할 수 있습니다. TLF는 저수준 코드를 더 간단한 API 안으로 캡슐화합니다. 또한 TLF는 FTE에서 정의된 구성 블록을 사용이 더 간편한 시스템으로 구성하는 개념적 아키텍처를 제공합니다.

FTE와 달리 TLF는 Flash Player에 내장되어 있지 않습니다. 정확히 말해 이 프레임워크는 ActionScript 3.0으로 완전히 작성된 독립적인 구성 요소 라이브러리이며, 확장 가능하기 때문에 특정 환경에 맞게 사용자 정의할 수 있습니다. Flash Professional 및 Flex SDK는 모두 TLF 프레임워크를 기반으로 하는 구성 요소를 포함합니다.

기타 도움말 항목

"플로우" TLF 마크업 응용 프로그램

복잡한 스크립트 지원

Flash Player 10 이상, Adobe AIR 1.5 이상

TLF는 복잡한 스크립트 지원을 제공합니다. 복잡한 스크립트 지원에는 오른쪽에서 왼쪽 방향의 스크립트를 표시 및 편집할 수 있는 기능이 포함됩니다. 또한 TLF는 아랍어나 히브리어와 같이 왼쪽에서 오른쪽 방향 및 오른쪽에서 왼쪽 방향이 혼합된 스크립트를 표시하고 편집할 수 있는 기능을 제공합니다. 이 프레임워크는 중국어, 일본어 및 한국어에 대한 세로 방향 텍스트 레이아웃은 물론 TCY(문자 회전) 요소도 지원합니다. TCY 요소는 세로 방향 텍스트에 포함된 가로 쓰기 텍스트 블록입니다. 다음과 같은 스크립트가 지원됩니다.

- 라틴(영어, 스페인어, 프랑스어, 베트남어 등)
- 그리스어, 키릴 자모, 아르메니아어, 그루지야어, 게에즈 문자
- 아랍어 및 히브리어
- 한 표의 문자 및 가나(한국어, 중국어 및 일본어) 및 한글 조합(한국어)
- 태국어, 라오스어, 크메르어
- 데바나가리 문자, 벵골어, 굴묵키어, 말라얄람어, 텔루구어, 타밀어, 구자라트어, 오리야어, 카나다어, 티베트어
- 티푸나구어, 이 문자, 체로키어, 캐나다 음절, 데저렛 문자, 샤우 문자, 바이어, 타갈로그어, 하누누어, 부히드어, 타그반와어

Flash Professional 및 Flex에서 Text Layout Framework 사용

TLF 클래스를 직접 사용하여 Flash에서 사용자 정의 구성 요소를 만들 수 있습니다. 또한 Flash Professional CS5에서는 TLF 기능을 캡슐화하는 새 클래스 fl.text.TLFTextField를 제공합니다. TLFTextField 클래스를 사용하여 ActionScript에 TLF의 고급 텍스트 표시 기능을 사용하는 텍스트 필드를 만들 수 있습니다. TextField 클래스의 텍스트 필드를 만들 때와 같은 방법으로 TLFTextField 객체를 만듭니다. 그런 다음 textFlow 속성을 사용하여 TLF 클래스로부터 향상된 서식 지정 옵션 및 기능을 지정합니다.

또한 Flash Professional을 사용하여 텍스트 도구로 스테이지에 TLFTextField 인스턴스를 만들 수 있습니다. 그런 다음 ActionScript를 사용하면 TLF 클래스를 통해 텍스트 필드 내용의 서식 및 레이아웃을 제어할 수 있습니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 TLFTextField를 참조하십시오.

Flex에서 작업하는 경우 TLF 클래스를 사용하십시오. 자세한 내용은 387페이지의 “Text Layout Framework 사용”을 참조하십시오.

Text Layout Framework 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

Flex에서 작업하거나 사용자 정의 텍스트 구성 요소를 만드는 경우 TLF 클래스를 사용하십시오. TLF는 textLayout.swc 라이브러리 내에 완전히 포함된 ActionScript 3.0 라이브러리입니다. TLF 라이브러리에는 약 100개의 ActionScript 3.0 클래스 및 인터페이스가 10개의 패키지로 구성되어 있습니다. 이러한 패키지는 flashx.textLayout 패키지의 하위 패키지입니다.

Text Layout Framework 클래스

Flash Player 10 이상, Adobe AIR 1.5 이상

TLF 클래스는 다음 세 개의 범주로 그룹화됩니다.

- 데이터 구조 및 서식 지정 클래스

- 렌더링 클래스
- 사용자 상호 작용 클래스

데이터 구조 및 서식 지정 클래스

다음 패키지에는 TLF의 데이터 구조 및 서식 지정 클래스가 포함되어 있습니다.

- flashx.textLayout.elements
- flashx.textLayout.formats
- flashx.textLayout.conversion

TLF의 기본 데이터 구조는 텍스트 흐름 계층 구조이며 요소 패키지에서 정의됩니다. 이 구조 내에서 서식 패키지를 사용하여 텍스트 흐름에 스타일 및 특성을 할당할 수 있으며, 변환 패키지를 통해 텍스트를 데이터 구조로 가져오고 데이터 구조에서 내보내는 방식을 제어할 수도 있습니다.

렌더링 클래스

다음 패키지에는 TLF의 렌더링 클래스가 포함되어 있습니다.

- flashx.textLayout.factory
- flashx.textLayout.container
- flashx.textLayout.compose

이러한 패키지의 클래스를 사용하면 Flash Player에서 표시할 텍스트를 더욱 간편히 렌더링할 수 있습니다. 팩토리 패키지는 정적 텍스트를 표시하는 간단한 방법을 제공합니다. 컨테이너 패키지에는 동적 텍스트의 표시 컨테이너를 정의하는 클래스 및 인터페이스가 포함됩니다. 작성 패키지는 컨테이너에 동적 텍스트를 배치 및 표시하는 기술을 정의합니다.

사용자 상호 작용 클래스

다음 패키지에는 TLF의 사용자 상호 작용 클래스가 포함되어 있습니다.

- flashx.textLayout.edit
- flashx.textLayout.operations
- flashx.textLayout.events

편집 및 작업 패키지는 데이터 구조에 저장된 텍스트의 편집을 허용하는 데 사용할 수 있는 클래스를 정의합니다. 이벤트 패키지는 이벤트 처리 클래스를 포함합니다.

Text Layout Framework으로 텍스트를 작성하는 일반적인 단계

다음 단계에서는 Text Layout Framework을 사용하여 텍스트를 작성하는 일반적인 프로세스를 설명합니다.

1. 서식 있는 텍스트를 TLF 데이터 구조로 가져옵니다. 자세한 내용은 391페이지의 “TLF를 사용한 텍스트 구조화” 및 395페이지의 “TLF를 사용한 텍스트 서식 지정”을 참조하십시오.
2. 해당 텍스트에 대해 링크된 표시 객체 컨테이너를 하나 이상 만듭니다. 자세한 내용은 397페이지의 “TLF를 사용한 텍스트 컨테이너 관리”를 참조하십시오.
3. 데이터 구조의 텍스트를 컨테이너와 연결하고 편집 및 스크롤 옵션을 설정합니다. 자세한 내용은 398페이지의 “TLF를 사용한 텍스트 선택, 편집 및 실행 취소 활성화”를 참조하십시오.
4. resize 또는 기타 이벤트에 대한 응답으로 텍스트를 리플로우하는 이벤트 핸들러를 만듭니다. 자세한 내용은 398페이지의 “TLF를 사용한 이벤트 처리”를 참조하십시오.

Text Layout Framework 예제: 뉴스 레이아웃

Flash Player 10 이상, Adobe AIR 1.5 이상

다음 예제에서는 TLF를 사용하여 간단한 신문 페이지의 레이아웃을 구성하는 방법을 보여 줍니다. 페이지에는 큰 헤드라인, 소제목, 여러 열로 구성된 본문 섹션이 포함됩니다.

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

TLFNewsLayout 클래스는 두 개의 텍스트 컨테이너를 사용합니다. 컨테이너 하나는 헤드라인 및 소제목을 표시하고 다른 하나는 세 열의 본문 텍스트를 표시합니다. 간단하게 보여 주기 위해 텍스트는 TLF 마크업 텍스트로 예제에 하드 코드되었습니다. headlineMarkup 변수에는 헤드라인과 소제목이 모두 포함되고 bodyMarkup 변수에는 본문 텍스트가 포함됩니다. TLF 마크업에 대한 자세한 내용은 391페이지의 “TLF를 사용한 텍스트 구조화”를 참조하십시오.

초기화 후에 onAddedToStage() 함수는 헤드라인 텍스트를 TextFlow 객체로 가져오며 이 객체는 TLF의 기본 데이터 구조입니다.

```
hTextFlow = TextConverter.importToFlow(headlineMarkup, TextConverter.TEXT_LAYOUT_FORMAT);
```

다음으로 컨테이너에 대한 Sprite 객체가 만들어지고 컨트롤러가 만들어져 해당 컨테이너와 연결됩니다.

```
headContainer = new Sprite();
headlineController = new ContainerController(headContainer);
```

컨트롤러는 서식 지정, 스크롤 및 기타 옵션을 설정하기 위해 초기화됩니다. 이 컨트롤러에는 텍스트 흐름이 들어가는 컨테이너의 경계를 정의하는 기하 구조가 포함됩니다. TextLayoutFormat 객체에는 서식 지정 옵션이 포함됩니다.

```
hContainerFormat = new TextLayoutFormat();
```

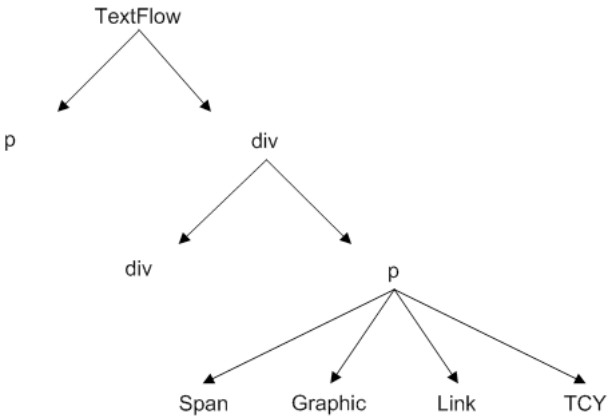
컨트롤러는 흐름 컴포저에 할당되고 함수는 표시 목록에 컨테이너를 추가합니다. 컨테이너의 실제 컴포지션 및 표시는 resizeHandler() 메서드가 담당하게 됩니다. 동일한 단계가 반복되어 본문 TextFlow 객체를 초기화합니다.

resizeHandler() 메서드는 컨테이너를 렌더링하는 데 사용할 수 있는 공간을 측정하고 그에 따라 컨테이너의 크기를 조정합니다. compose() 메서드에 대한 초기 호출을 통해 헤드라인 컨테이너의 적절한 높이를 계산할 수 있습니다. 그런 다음 resizeHandler() 메서드는 updateAllControllers() 메서드를 사용하여 헤드라인 컨테이너를 배치하고 표시할 수 있습니다. 마지막으로 resizeHandler() 메서드는 헤드라인 컨테이너의 크기를 사용하여 본문 텍스트 컨테이너의 배치를 결정합니다.

TLF를 사용한 텍스트 구조화

TLF는 계층 구조 트리를 사용하여 텍스트를 표현합니다. 트리에 있는 각 노드는 요소 패키지에서 정의한 클래스의 인스턴스입니다. 예를 들어 트리의 루트 노드는 항상 TextFlow 클래스의 인스턴스입니다. TextFlow 클래스는 전체 텍스트의 스토리를 나타냅니다. 스토리는 하나의 단위 또는 흐름으로 취급되는 텍스트 및 기타 요소의 모음입니다. 하나의 기사에 두 개 이상의 열 또는 텍스트 컨테이너를 표시해야 할 수 있습니다.

루트 노드 이외의 나머지 요소는 대체로 XHTML 요소에 기반합니다. 다음 다이어그램에서는 프레임워크의 계층 구조를 보여 줍니다.



TextFlow 계층 구조

Text Layout Framework 마크업

TLF의 구조를 이해하면 TLF 마크업을 처리할 때도 도움이 됩니다. TLF 마크업은 TLF에 포함된 텍스트의 XML 표현입니다. TLF에서는 다른 XML 형식도 지원하지만 TLF 마크업은 특히 TextFlow 계층 구조에 기반한다는 점에서 고유합니다. 이 마크업 형식을 사용하여 TextFlow에서 XML을 내보내는 경우 이 계층 구조가 그대로 유지된 상태로 XML이 내보내집니다.

TLF 마크업은 TextFlow 계층 구조를 통해 텍스트의 표현에 있어 최고의 정확도를 보여 줍니다. 이 마크업 언어는 TextFlow 계층 구조의 각 기본 요소에 대해 태그를 제공하며 TextLayoutFormat 클래스에서 사용 가능한 모든 서식 지정 속성에 대한 특성도 제공합니다.

다음 표에는 TLF Markup에서 사용할 수 있는 태그가 나와 있습니다.

요소	설명	하위	클래스
textflow	마크업의 루트 요소입니다.	div, p	TextFlow
div	TextFlow 내 구획입니다. 단락 그룹을 포함할 수 있습니다.	div, list, p	DivElement
p	단락입니다.	a, tcy, span, img, tab, br, g	ParagraphElement
a	링크입니다.	tcy, span, img, tab, br, g	LinkElement
tcy	가로 방향 텍스트(세로 방향 TextFlow에 사용됨)	a, span, img, tab, br, g	TCYElement
span	단락 내 텍스트 방향입니다.		SpanElement
img	단락의 이미지입니다.		InlineGraphicElement
tab	탭 문자입니다.		TabElement
br	줄 바꿈 문자입니다. 단락 내 행을 마무리할 때 사용되며 텍스트는 다음 행에서 계속되지만 같은 단락 안에서 유지됩니다.		BreakElement
linkNormalFormat	보통 상태의 링크에 사용되는 서식 속성을 정의합니다.	TextLayoutFormat	TextLayoutFormat
linkActiveFormat	마우스를 링크 위에 댈 때 활성화된 상태의 링크에 사용되는 서식 속성을 정의합니다.	TextLayoutFormat	TextLayoutFormat

요소	설명	하위	클래스
linkHoverFormat	마우스를 링크 경계 내에서 링크 위로 움직일 때 가리킨 상태의 링크에 사용되는 서식 속성을 정의합니다.	TextLayoutFormat	TextLayoutFormat
li	목록 항목 요소입니다. 목록 요소 안에 있어야 합니다.	div, li, list, p	ListItemElement
list	목록입니다. 목록을 중첩시키거나 서로 인접하게 배치할 수 있습니다. 목록 항목에 서로 다른 레이블 지정 또는 번호 매기기 기법을 적용할 수 있습니다.	div, li, list, p	ListElement
g	그룹 요소입니다. 요소를 단락으로 그룹화할 때 사용합니다. 이를 통해 단락 레벨 아래에 요소를 중첩할 수 있습니다.	a, tcy, span, img, tab, br, g	SubParagraphGroup Element

기타 도움말 항목

TLF 2.0 목록 마크업

TLF 2.0 SubParagraphGroupElements 및 typeName

번호가 매겨진 목록 및 글머리 기호 목록 사용

ListElement 및 ListItemElement 클래스를 사용하여 글머리 기호 목록을 텍스트 컨트롤에 추가할 수 있습니다. 글머리 기호 목록은 중첩시킬 수도 있고 사용자 정의하여 서로 다른 글머리 기호(또는 표시자) 및 자동 번호 매기기는 물론 외곽선 스타일 번호 매기기를 사용하도록 할 수도 있습니다.

텍스트 흐름에 목록을 만들려면 <list> 태그를 사용합니다. 그런 다음 목록에 있는 각 목록 항목에 대해 <list> 태그 안에 <li> 태그를 사용합니다. ListMarkerFormat 클래스를 사용하여 글머리 기호의 모양을 사용자 정의할 수 있습니다.

다음은 간단한 목록을 만드는 예제입니다.

```
<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>
```

다음 예제에서처럼 목록을 다른 목록 안에 중첩시킬 수 있습니다.

```
<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>
```

목록에 있는 표시자의 유형을 사용자 정의하려면 ListElement의 listStyleType 속성을 사용합니다. 이 속성은 ListStyleType 클래스로 정의된 모든 값을 사용할 수 있습니다(예: check, circle, decimal 및 box). 다음은 표시자 유형이 다양하고 카운터 증분을 사용자 정의할 수 있는 목록을 만드는 예제입니다.

```
<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>
```

카운터를 정의할 때는 ListMarkerFormat 클래스를 사용합니다. 카운터 증분을 정의하는 것은 물론 counterReset 속성으로 재설정하여 카운터를 사용자 정의할 수도 있습니다.

ListMarkerFormat의 afterContent 및 beforeContent 속성을 사용하면 목록에 있는 표시자의 모양도 사용자 정의할 수 있습니다. 이러한 속성은 표시자의 내용 앞뒤에 나타나는 내용에 적용됩니다.

다음 예제에서는 표시자 앞에 문자열 “XX”를 추가하고 표시자 뒤에는 문자열 “YY”를 추가합니다.

```
<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>
```

content 속성 자체로도 표시자 형식을 추가로 사용자 정의할 수 있습니다. 다음은 정렬된 대문자 로마 숫자 표시자를 표시하는 예제입니다.

```
<flow:list listStyleType="disc"  paddingLeft="96" paddingRight="24">
    <flow:listMarkerFormat>
        <flow:ListMarkerFormat fontSize="16"
            beforeContent="Section "
            content="counters(ordered,&amp;quot;*&amp;quot;,upperRoman)"
            afterContent=": "/>
    </flow:listMarkerFormat>
    <flow:li>Item 1</li>
    <flow:li>Item 2</li>
    <flow:li>Item 3</li>
</flow:list>
```

위의 예제에서처럼 content 속성도 접미어를 삽입할 수 있습니다. 접미어란 표시자 뒤에, afterContent 앞에 나타나는 문자열입니다. 흐름에 XML 내용을 넣을 때 이 문자열을 삽입하려면 인용 부호가 아닌 &quote; HTML 엔터티로 문자열을 묶습니다 ("<***string***>").

기타 도움말 항목

TLF 2.0 목록 마크업

TLF에 여백 사용

모든 FlowElement는 각 요소의 내용 영역 위치 및 내용 영역 간의 간격을 제어하는 데 사용하는 여백 속성을 지원합니다.

요소의 전체 폭은 내용의 폭을 모두 합한 값에다 paddingLeft 및 paddingRight 속성을 더한 것입니다. 요소의 전체 높이는 내용의 높이를 모두 합한 값에다 paddingTop 및 paddingBottom 속성을 더한 것입니다.

여백은 테두리와 내용 사이 간격을 말합니다. 여백 속성에는 paddingBottom, paddingTop, paddingLeft 및 paddingRight가 있습니다. 여백은 TextFlow 객체에는 물론 다음과 같은 자식 요소에 적용할 수 있습니다.

- div
- img
- li
- list
- p

범위 요소에는 여백 속성을 적용할 수 없습니다.

다음 예제에서는 TextFlow에 여백 속성을 설정합니다.

```
<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">
```

여백 속성의 유효한 값은 숫자(픽셀), “auto” 또는 “inherit”입니다. 기본값은 “auto”로, ListElement를 제외한 모든 요소에서 자동으로 계산되고 0으로 설정됩니다. ListElements의 경우 listAutoPadding 속성 값이 사용되는 목록의 시작 지점을 제외하고 “auto”는 0이 됩니다. listAutoPadding의 기본값은 40으로, 목록에 기본 들여쓰기를 지정합니다.

여백 속성은 기본적으로 상속하지 않습니다. “auto” 및 “inherit” 값은 FormatValue 클래스로 정의된 상수입니다.

여백 속성은 음수 값이 될 수 있습니다.

기타 도움말 항목

TLF 2.0의 여백 변경

TLF를 사용한 텍스트 서식 지정

Flash Player 10 이상, Adobe AIR 1.5 이상

flashx.textLayout.formats 패키지는 텍스트 흐름 계층 구조 트리의 FlowElement에 서식을 할당할 수 있도록 하는 인터페이스와 클래스를 포함합니다. 서식을 적용하는 방법은 두 가지가 있습니다. 특정 서식을 개별적으로 할당하거나 특수 서식 객체를 사용하여 서식 그룹을 동시에 할당할 수 있습니다.

ITextLayoutFormat 인터페이스는 FlowElement에 적용할 수 있는 모든 서식을 포함합니다. 일부 서식은 전체 컨테이너 또는 텍스트 단락에 적용되지만 개별 문자에는 논리적으로 적용되지 않습니다. 예를 들어 정렬 및 탭 정지 같은 서식은 전체 단락에 적용되지만 개별 문자에는 적용할 수 없습니다.

속성을 사용하여 FlowElement에 서식 할당

Flash Player 10 이상, Adobe AIR 1.5 이상

속성 할당을 통해 FlowElement에 서식을 설정할 수 있습니다. FlowElement 클래스는 ITextLayoutFormat 인터페이스를 구현하므로 FlowElement 클래스의 모든 하위 클래스도 이 인터페이스를 구현해야 합니다.

예를 들어 다음 코드는 ParagraphElement 인스턴스에 개별 서식을 할당하는 방법을 보여 줍니다.

```
var p:ParagraphElement = new ParagraphElement();
p.fontSize = 18;
p.fontFamily = "Arial";
```

TextLayoutFormat 클래스를 사용하여 FlowElement에 서식 할당

Flash Player 10 이상, Adobe AIR 1.5 이상

TextLayoutFormat 클래스를 사용하여 FlowElement에 서식을 적용할 수 있습니다. 이 클래스를 사용하면 필요한 모든 서식 값이 포함된 특별한 서식 지정 객체를 만들 수 있습니다. 그런 다음 FlowElement 객체의 property 속성에 이 객체를 할당할 수 있습니다. TextLayoutFormat 및 FlowElement는 모두 ITextLayoutFormat 인터페이스를 구현합니다. 다음 준비 작업을 통해 두 클래스에 모두 동일한 서식 속성이 포함되도록 합니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 TextLayoutFormat을 참조하십시오.

서식 상속

Flash Player 10 이상, Adobe AIR 1.5 이상

서식은 텍스트 흐름 계층 구조를 통해 상속됩니다. TextLayoutFormat 인스턴스를 자식이 있는 FlowElement 인스턴스에 할당하는 경우 프레임워크에서 **겹쳐 놓기(cascade)**라는 프로세스를 시작합니다. 겹쳐 놓기 프로세스 동안 프레임워크는 계층 구조에서 FlowElement로부터 상속하는 각 노드를 반복적으로 검사합니다. 그런 다음 상속된 값을 각 서식 속성에 할당할지 여부를 결정합니다. 겹쳐 놓기 프로세스가 수행되는 동안 다음과 같은 규칙이 적용됩니다.

1. 바로 위 요소(부모)의 속성 값만 상속됩니다.
2. 아직 속성의 값이 없는 경우, 즉 값이 undefined인 경우에만 속성 값이 상속됩니다.
3. 일부 특성의 경우에는 특성의 값이 "inherit" 설정되었거나 상수 flashx.textLayout.formats.FormatValue.INHERIT로 설정된 경우가 아니면 값이 정의되지 않았더라도 값이 상속되지 않습니다.

예를 들어 TextFlow 수준에서 fontSize 값을 설정하는 경우 TextFlow의 모든 요소에 해당 설정이 적용됩니다. 즉, 값이 텍스트 흐름 계층 구조에 따라 아래로 겹쳐 놓기 방식으로 적용됩니다. 그러나 특정 요소에 직접 새 값을 할당하여 해당 요소에서 이 값을 재정의할 수 있습니다. 반대로 TextFlow 수준에서 backgroundColor 값을 설정하는 경우에는 TextFlow가 해당 값을 상속하지 않습니다. backgroundColor 속성은 겹쳐 놓기 프로세스 동안 부모로부터 상속되지 않는 속성입니다. 각 자식에서 backgroundColor 속성을 flashx.textLayout.formats.FormatValue.INHERIT에 설정하여 이 비헤이비어를 재정의할 수 있습니다..

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 TextLayoutFormat을 참조하십시오.

TLF를 사용한 텍스트 가져오기 및 내보내기

Flash Player 10 이상, Adobe AIR 1.5 이상

flashx.textLayout.conversion의 TextConverter 클래스입니다.* 패키지를 사용하여 텍스트를 TLF로 가져오거나 TLF에서 내보낼 수 있습니다. 이 클래스는 텍스트를 SWF 파일로 컴파일하는 대신 런타임에 로드하려는 경우에 사용합니다. 또한 이 클래스를 사용하면 TextFlow 인스턴스에 저장된 텍스트를 String 또는 XML 객체로 내보낼 수 있습니다.

가져오기와 내보내기의 절차는 모두 간단합니다. TextConverter 클래스의 일부인 export() 메서드 또는 importToFlow() 메서드를 호출하면 됩니다. 두 가지 모두 정적 메서드이므로 TextConverter 클래스의 인스턴스가 아니라 TextConverter 클래스에서 메서드를 호출합니다.

flashx.textLayout.conversion 패키지의 클래스를 사용하면 텍스트의 저장 위치를 매우 유연하게 선택할 수 있습니다. 예를 들어 텍스트를 데이터베이스에 저장할 경우 표시를 위해 텍스트를 프레임워크로 가져올 수 있습니다. 그런 다음 flashx.textLayout.edit 패키지의 클래스를 사용하여 텍스트를 변경하고 변경된 텍스트를 데이터베이스로 다시 내보낼 수 있습니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 flashx.textLayout.conversion을 참조하십시오.

TLF를 사용한 텍스트 컨테이너 관리

Flash Player 10 이상, Adobe AIR 1.5 이상

텍스트가 TLF 데이터 구조에 저장된 후에는 Flash Player에서 해당 텍스트를 표시할 수 있습니다. 흐름 계층 구조에 저장된 텍스트는 Flash Player에서 표시할 수 있는 형식으로 변환되어야 합니다. TLF에서는 흐름으로부터 표시 객체를 만들 수 있는 두 가지 방식을 제공합니다. 첫 번째는 더욱 간편한 방법으로, 정적 텍스트를 표시하는 데 적합합니다. 더 복잡한 두 번째 방법을 사용하면 선택 및 편집이 가능한 동적 텍스트를 만들 수 있습니다. 두 가지 방법 중 어떤 것을 사용하든 텍스트는 결국 Flash Player 10에 포함된 flash.text.engine.* 패키지의 일부인 TextLine 클래스의 인스턴스로 변환됩니다.

정적 텍스트 만들기

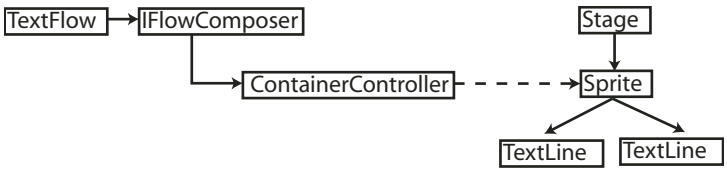
단순한 방법에서는 flashx.textLayout.factory 패키지에 포함된 TextFlowTextLineFactory 클래스를 사용합니다. 이 방법은 간단할 뿐 아니라 FlowComposer 방법에 비해 메모리를 적게 사용한다는 장점이 있습니다. 사용자가 편집, 선택 또는 스크롤할 필요가 없는 정적 텍스트에 이 방법을 사용하는 것이 좋습니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 TextFlowTextLineFactory를 참조하십시오.

동적 텍스트 및 컨테이너 만들기

TextFlowTextLineFactory에서 제공하는 것보다 더 높은 수준의 텍스트 표시 제어 기능이 필요한 경우에는 흐름 컴포저를 사용합니다. 예를 들어 흐름 컴포저를 사용하면 텍스트를 선택하고 편집할 수 있습니다. 자세한 내용은 398페이지의 "TLF를 사용한 텍스트 선택, 편집 및 실행 취소 활성화"를 참조하십시오.

흐름 컴포저는 flashx.textLayout.compose 패키지에 있는 StandardFlowComposer 클래스의 인스턴스입니다. 흐름 컴포저는 TextFlow를 TextLine 인스턴스로 변환하는 작업 및 변환된 TextLine 인스턴스를 하나 이상의 컨테이너에 배치하는 작업을 관리합니다.



IFlowComposer에는 0개 이상의 ContainerController가 있습니다.

각 TextFlow 인스턴스에는 IFlowComposer 인터페이스를 구현하는 해당 객체가 있습니다. 이 IFlowComposer 객체는 TextFlow.flowComposer 속성을 통해 액세스할 수 있습니다. 이 속성을 통해 IFlowComposer 인터페이스에 정의된 메서드를 호출할 수 있습니다. 이 메서드를 사용하면 텍스트를 하나 이상의 컨테이너와 연결하고 해당 텍스트를 컨테이너 안에 표시하도록 준비할 수 있습니다.

컨테이너는 DisplayObjectContainer 클래스의 하위 클래스인 Sprite 클래스의 인스턴스입니다. 두 클래스 모두 Flash Player 표시 목록 API의 일부입니다. 컨테이너는 TextLineFactory 클래스에서 사용되는 경계 사각형의 보다 발전된 형태입니다. 경계 사각형과 마찬가지로 컨테이너는 TextLine 인스턴스가 표시되는 영역의 경계를 정의합니다. 경계 사각형과 달리 컨테이너에는 상응하는 "컨트롤러" 객체가 있습니다. 컨트롤러는 컨테이너 또는 컨테이너 집합에 대한 스크롤, 컴포지션, 연결, 서식 지정 및 이벤트 처리 등을 관리합니다. 각 컨테이너는 flashx.textLayout.container 패키지에 있는 ContainerController 클래스의 인스턴스인 해당 컨트롤러 객체를 포함합니다.

텍스트를 표시하려면 컨테이너를 관리할 컨트롤러 객체를 만들어 흐름 컴포저와 연결하십시오. 컨테이너를 연결한 후에는 텍스트를 표시할 수 있도록 컴포지션해야 합니다. 따라서 컨테이너에는 컴포지션 및 표시의 두 가지 상태가 포함됩니다. 컴포지션은 텍스트 흐름 계층 구조의 텍스트를 TextLine 인스턴스로 변환하고 이러한 인스턴스가 컨테이너에 맞게 표시되는지 여부를 계산하는 프로세스입니다. 표시는 Flash Player 표시 목록을 업데이트하는 프로세스입니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 IFlowComposer, StandardFlowComposer 및 ContainerController를 참조하십시오.

TLF를 사용한 텍스트 선택, 편집 및 실행 취소 활성화

Flash Player 9.0 이상, Adobe AIR 1.0 이상

텍스트를 선택 또는 편집할 수 있는 기능은 텍스트 흐름 수준에서 제어됩니다. 모든 TextFlow 클래스 인스턴스에는 연결된 상호 작용 관리자가 있습니다. TextFlow 객체의 TextFlow.interactionManager 속성을 통해 TextFlow 객체의 상호 작용 관리자에 액세스할 수 있습니다. 텍스트 선택을 활성화하려면 SelectionManager 클래스의 인스턴스를 interactionManager 속성에 할당하고, 텍스트 선택과 편집을 모두 활성화하려면 SelectionManager 클래스의 인스턴스 대신에 EditManager 클래스의 인스턴스를 할당합니다. 실행 취소 작업을 활성화하려면 UndoManager 클래스의 인스턴스를 만들어 EditManager 생성자를 호출할 때 인수로 포함합니다. UndoManager 클래스를 사용하면 사용자의 최근 편집 작업 내역을 유지할 수 있고 사용자가 특정 편집 작업을 실행 취소하거나 다시 실행할 수 있습니다. 이러한 세 클래스는 모두 편집 패키지에 포함됩니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 SelectionManager, EditManager 및 UndoManager를 참조하십시오.

TLF를 사용한 이벤트 처리

Flash Player 10 이상, Adobe AIR 1.5 이상

TextFlow 객체는 다음과 같은 여러 상황에서 이벤트를 전달합니다.

- 텍스트 또는 레이아웃이 변경될 경우
- 작업이 시작되기 전과 작업이 완료된 후
- FlowElement 객체의 상태가 변한 경우
- 구성 작업이 완료될 때

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 flashx.textLayout.events를 참조하십시오.

기타 도움말 항목

TLF FlowElement/LinkElement 이벤트 및 EventMirrors

텍스트 내 이미지 위치 지정

텍스트 내에서 InlineGraphicElement의 위치를 지정하려면 다음 속성을 사용합니다.

- InlineGraphicElement 클래스의 float 속성
- FlowElement의 clearFloats 속성

float 속성은 그래픽 및 주변 텍스트의 배치를 제어합니다. clearFloats 속성은 float를 기준으로 단락 요소의 배치를 제어합니다.

텍스트 요소 내에서 이미지의 위치를 제어하려면 float 속성을 사용합니다. 다음 예제에서는 단락에 이미지를 추가하고 이미지를 왼쪽으로 정렬하여 텍스트가 오른쪽을 기준으로 나열되도록 합니다.

```
<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>
```

float 속성의 유효한 값은 “left”, “right”, “start”, “end” 및 “none”입니다. Float 클래스는 다음 상수를 정의합니다. 기본값은 “none”입니다.

clearFloats 속성은 대개 이미지를 둘러싸게 되는 후속 단락의 시작 지점을 조절할 때 유용합니다. 예를 들어 첫 번째 단락보다 크기가 큰 이미지가 있다고 가정할 경우 두 번째 단락이 이미지 **뒤에서부터** 시작되도록 하려면 clearFloats 속성을 설정합니다.

다음 예제에서는 첫 번째 단락의 텍스트보다 높이가 높은 이미지를 사용합니다. 두 번째 단락이 텍스트 블록의 이미지 뒤에서부터 시작되도록 하기 위해 이 예제에서는 두 번째 단락의 clearFloats 속성을 “end”로 설정합니다.

```
<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>
```

clearFloats 속성의 유효한 값은 “left”, “right”, “end”, “start”, “none” 및 “both”입니다. ClearFloats 클래스는 다음 상수를 정의합니다. clearFloats 속성을 “inherit”로 설정할 수도 있는데 이는 FormatValue 클래스로 정의되는 상수입니다. 기본값은 “none”입니다.

기타 도움말 항목

TLF Floats

24장: 사운드를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript로 대화식 몰입형 응용 프로그램을 만들 수 있습니다. 강력한 몰입형 응용 프로그램에서 자주 간과되는 요소가 사운드인데 비디오 게임에 사운드 효과를 추가하거나, 응용 프로그램 사용자 인터페이스에 오디오 피드백을 추가하거나, 인터넷에서 로드한 mp3 파일을 분석하는 프로그램을 만드는 등 사운드를 응용 프로그램에 중심적으로 활용할 수 있습니다.

외부 오디오 파일을 로드하고 SWF에 포함된 오디오를 사용하여 작업할 수 있습니다. 오디오를 제어하고, 사운드 정보의 시각적 표현을 만들고, 마이크에서 사운드를 캡처할 수 있습니다.

기타 도움말 항목

flash.media 패키지

flash.events.SampleDataEvent

사운드를 사용한 작업의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

컴퓨터에서 디지털 오디오(디지털화된 사운드 정보)를 캡처 및 인코딩하여 저장한 후 다시 가져와서 스피커를 통해 재생할 수 있습니다. Adobe® Flash® Player 또는 Adobe® AIR™과 ActionScript를 사용하여 사운드를 재생할 수 있습니다.

사운드 데이터가 디지털 형식으로 변환되면 사운드 볼륨과 스테레오 또는 모노 사운드 등의 다양한 특성이 생깁니다. ActionScript에서 사운드를 재생할 때 이러한 특성도 조정할 수 있습니다. 예를 들어 사운드를 더 크게 하거나 특정 방향에서 사운드가 나오는 듯한 효과를 연출할 수 있습니다.

ActionScript에서 사운드를 제어하려면 먼저 사운드 정보를 Flash Player 또는 AIR에 로드해야 합니다. ActionScript를 사용하여 작업할 수 있도록 오디오 데이터를 Flash Player 또는 AIR에 로드하는 방법에는 다섯 가지가 있습니다.

- mp3 파일과 같은 외부 사운드 파일을 SWF로 로드합니다.
- 사운드 정보를 작성되는 동안 직접 SWF 파일로 포함합니다.
- 사용자 컴퓨터에 연결된 마이크에서 오디오를 캡처합니다.
- 서버로에서 오디오를 스트리밍합니다.
- 오디오를 동적으로 생성하여 재생합니다.

외부 사운드 파일에서 사운드 데이터를 로드하는 경우 나머지 사운드 데이터를 로드하는 중에도 사운드 파일의 시작 부분을 먼저 재생할 수 있습니다.

디지털 오디오를 인코딩하는 데 사용되는 사운드 파일 형식은 다양하지만 ActionScript 3.0, Flash Player 및 AIR는 mp3 형식으로 저장된 사운드 파일을 지원합니다. WAV 또는 AIFF 등의 다른 형식으로 된 사운드 파일은 직접 로드하거나 재생할 수 없습니다.

ActionScript에서 사운드 관련 작업을 하는 동안에는 flash.media 패키지의 몇 가지 클래스를 다루게 될 것입니다. Sound 클래스는 사운드 파일을 로드하거나 사운드 데이터를 샘플링하는 이벤트에 함수를 할당한 다음 재생을 시작하여 오디오 정보에 대한 액세스 권한을 얻는 데 사용하는 클래스입니다. 사운드 재생을 시작하면 Flash Player 및 AIR 사용자에게 SoundChannel 객체에 대한 액세스 권한이 부여됩니다. 로드한 오디오 파일은 사용자의 컴퓨터에서 재생하는 몇 가지 사운드 중 하나일 수 있으므로 재생되는 각 개별 사운드는 자체의 SoundChannel 객체를 사용합니다. 함께 믹싱되는 모든 SoundChannel 객체의 조합된 출

력은 컴퓨터 스피커를 통해 실제로 재생되는 사운드입니다. 이 SoundChannel 인스턴스를 사용하여 사운드의 속성을 제어하거나 재생을 중지할 수 있습니다. 마지막으로, 조합된 오디오를 제어하려는 경우 SoundMixer 클래스를 통해 믹싱된 출력을 제어할 수 있습니다.

ActionScript에서 사운드 관련 작업을 할 때 기타 몇 가지 클래스를 사용하여 특정 작업을 수행할 수도 있습니다. 사운드 관련 클래스 전반에 대한 자세한 내용은 401페이지의 “사운드 아키텍처의 이해”를 참조하십시오.

중요한 개념 및 용어

다음 참조 목록에는 이 장에 사용되는 중요한 용어가 포함되어 있습니다.

진폭 사운드 파형의 한 점과 0 또는 기준선 사이의 거리입니다.

비트율 사운드 파일의 초당 인코딩 또는 스트리밍되는 데이터 양입니다. mp3 파일의 경우 비트율은 보통 kbps(초당 킬로비트) 단위로 표현됩니다. 비트율이 높을수록 일반적으로 음질이 좋습니다.

버퍼링 사운드 재생 전에 사운드 데이터를 수신하여 저장하는 과정입니다.

mp3 MPEG-1 Audio Layer 3, 즉 mp3는 널리 사용되는 사운드 압축 형식입니다.

패닝 스테레오 음장에서 왼쪽 채널과 오른쪽 채널 간 오디오 신호의 배치를 지정하는 과정입니다.

피크 파형에서 가장 높은 점입니다.

샘플링 속도 디지털 신호를 만들기 위해 아날로그 오디오 신호에서 추출하는 초당 샘플 수를 정의합니다. 표준 콤팩트 디스크 오디오의 샘플링 속도는 44.1kHz, 즉 44,100샘플/초입니다.

스트리밍 사운드 파일 또는 비디오 파일의 뒷부분을 서버에서 로드하는 동안, 이미 로드된 앞부분을 재생하는 프로세스입니다.

볼륨 사운드의 크기입니다.

파형 시간에 따른 사운드 신호의 진폭 변화를 나타내는 그래프 모양입니다.

사운드 아키텍처의 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램은 다음과 같은 5가지 소스로부터 사운드 데이터를 로드할 수 있습니다.

- 런타임에 로드된 외부 사운드 파일
- 응용 프로그램의 SWF 파일에 포함된 사운드 리소스
- 사용자의 시스템에 연결된 마이크로부터 입력된 사운드 데이터
- Flash Media Server 등의 원격 미디어 서버로부터 스트리밍된 사운드 데이터
- sampleData 이벤트 핸들러를 사용하여 동적으로 생성되는 사운드 데이터

사운드 데이터는 완전히 로드된 후에 재생되거나 스트리밍(로드 중 재생)될 수 있습니다.

ActionScript 3.0 사운드 클래스는 mp3 형식으로 저장된 사운드 파일을 지원합니다. WAV, AIFF 등의 다른 형식으로 된 사운드 파일은 직접 로드하거나 재생할 수 없습니다. 그러나 Flash Player 9.0.115.0부터는 NetStream 클래스를 사용하여 AAC 오디오 파일을 로드하고 재생할 수 있습니다. 이는 비디오 내용을 로드하고 재생하는 데 사용되는 것과 동일한 방법입니다. 이 방법에 대한 자세한 내용은 429페이지의 “비디오를 사용한 작업”을 참조하십시오.

Adobe Flash Professional을 사용하면 WAV 또는 AIFF 사운드 파일을 가져와서 이 파일을 mp3 형식으로 사용자 응용 프로그램의 SWF 파일에 포함할 수 있습니다. Flash 제작 도구를 사용하면 포함된 사운드 파일을 압축하여 파일 크기를 줄일 수도 있습니다. 단, 파일 크기가 작아지면 사운드 품질이 저하됩니다. 자세한 내용은 **Flash 사용**의 "사운드 가져오기"를 참조하십시오.

ActionScript 3.0 사운드 아키텍처는 flash.media 패키지에 있는 다음 클래스를 사용합니다.

클래스	설명
flash.media.Sound	Sound 클래스는 사운드의 로드를 처리하고 기본 사운드 속성을 관리하며 사운드 재생을 시작합니다.
flash.media.SoundChannel	응용 프로그램이 Sound 객체를 재생할 때는 새로운 SoundChannel 객체가 만들어져 재생을 제어합니다. SoundChannel 객체는 사운드의 왼쪽과 오른쪽 재생 채널의 볼륨을 제어합니다. 재생되는 각 사운드에는 자체의 SoundChannel 객체가 있습니다.
flash.media.SoundLoaderContext	SoundLoaderContext 클래스는 사운드를 로드할 때 사용할 버퍼링 시간(초)을 지정하고 파일을 로드할 때 서버에서 정책 파일을 찾을지 여부를 지정합니다. SoundLoaderContext 객체는 Sound.load() 메서드에 대한 매개 변수로 사용됩니다.
flash.media.SoundMixer	SoundMixer 클래스는 응용 프로그램의 모든 사운드와 관련된 재생 및 보안 속성을 제어합니다. 사실상, 다중 사운드 채널은 공통 SoundMixer 객체를 통해 믹싱되므로 SoundMixer 객체의 속성 값은 현재 재생되는 모든 SoundChannel 객체에 영향을 미칩니다.
flash.media.SoundTransform	SoundTransform 클래스에는 사운드 볼륨과 패닝을 제어하는 값이 포함됩니다. SoundTransform 객체는 무엇보다 개별 SoundChannel 객체, 전역 SoundMixer 객체 또는 Microphone 객체에 적용될 수 있습니다.
flash.media.ID3Info	ID3Info 객체에는 종종 mp3 사운드 파일에 저장되는 ID3 메타데이터 정보를 나타내는 속성이 포함됩니다.
flash.media.Microphone	Microphone 클래스는 사용자의 컴퓨터에 연결된 마이크 또는 기타 사운드 입력 장치를 나타냅니다. 마이크로부터 입력된 오디오는 로컬 스피커나 원격 서버로 라우팅될 수 있습니다. Microphone 객체는 게인, 샘플링 속도 및 자체 사운드 스트림의 기타 특성을 제어합니다.
flash.media.AudioPlaybackMode	AudioPlaybackMode 클래스는 SoundMixer 클래스의 audioPlaybackMode 속성에 대한 상수를 정의합니다.

로드 및 재생되는 각 사운드에는 Sound 클래스 및 SoundChannel 클래스의 고유한 인스턴스가 필요합니다. 그리고 다중 SoundChannel 인스턴스의 출력이 재생 중에 전역 SoundMixer 클래스에 의해 함께 믹싱됩니다.

Sound, SoundChannel, SoundMixer 클래스는 마이크를 통해 캡처한 사운드 데이터 또는 Flash Media Server 등의 스트리밍 미디어에서 캡처한 사운드 데이터에 사용되지 않습니다.

외부 사운드 파일 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

Sound 클래스의 각 인스턴스는 특정 사운드 리소스의 재생을 로드 및 트리거하기 위한 것입니다. 응용 프로그램에서 Sound 객체를 다시 사용하여 둘 이상의 사운드를 로드할 수 없습니다. 새로운 사운드 리소스를 로드하려면 새로운 Sound 객체를 만들어야 합니다.

작은 사운드 파일(예: 버튼에 연결될 클릭 사운드)을 로드하는 경우 응용 프로그램에서 새로운 Sound를 만들어 아래와 같이 사운드 파일을 자동으로 로드할 수 있습니다.

```
var req:URLRequest = new URLRequest("click.mp3");
var s:Sound = new Sound(req);
```

Sound() 생성자는 URLRequest 객체를 첫 번째 매개 변수로 받습니다. URLRequest 매개 변수의 값이 제공되면 새로운 Sound 객체는 지정된 사운드 리소스를 자동으로 로드합니다.

아주 단순한 경우를 제외하고는 응용 프로그램은 사운드의 로드 진행률을 살피고 로드 중에 오류가 발생하지 않는지 확인해야 합니다. 예를 들어 클릭 사운드의 용량이 꽤 큰 경우 사운드를 트리거하는 버튼을 사용자가 클릭할 때 사운드가 완전히 로드되지 않을 수 있습니다. 로드되지 않은 사운드를 재생하려고 하면 런타임 오류가 발생할 수 있습니다. 사운드 재생을 시작하는 액션을 취하도록 허용하기 전에 사운드가 완전히 로드될 때까지 기다리는 것이 안전합니다.

Sound 객체는 사운드 로드 프로세스 중에 여러 가지 이벤트를 전달합니다. 응용 프로그램은 이러한 이벤트를 수신하여 로드 진행률을 추적하고 사운드 재생 전에 사운드가 완전히 로드되도록 할 수 있습니다. 다음 표에서는 Sound 객체가 전달할 수 있는 이벤트를 설명합니다.

이벤트	설명
open(Event.OPEN)	사운드 로드 작업이 시작되기 직전에 전달됩니다.
progress(ProgressEvent.PROGRESS)	사운드 로드 프로세스 중에 파일 또는 스트림으로부터 데이터를 수신할 때 정기적으로 전달됩니다.
id3(Event.ID3)	mp3 사운드에 대해 ID3 데이터를 사용할 수 있는 경우 전달됩니다.
complete(Event.COMPLETE)	모든 사운드 리소스 데이터가 로드되었을 때 전달됩니다.
ioError(IOErrorEvent.IO_ERROR)	사운드 파일을 찾을 수 없거나 모든 사운드 데이터를 수신하기 전에 로드 프로세스가 중단된 경우 전달됩니다.

다음 코드는 사운드 로드 완료 후 재생 방법을 보여 줍니다.

```
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();
}
```

이 코드 샘플은 먼저 새 Sound 객체를 만듭니다. 이때 이 객체에 URLRequest 매개 변수에 대한 초기 값을 지정하지 않습니다. 그런 다음 Sound 객체로부터 Event.COMPLETE 이벤트를 수신하고, 모든 사운드 데이터가 로드되면 onSoundLoaded() 메서드를 실행합니다. 다음으로, 사운드 파일에 대한 새로운 URLRequest 값을 사용하여 Sound.load() 메서드를 호출합니다.

사운드 로드가 완료되면 onSoundLoaded() 메서드가 실행됩니다. Event 객체의 대상 속성은 Sound 객체에 대한 참조입니다. Sound 객체의 play() 메서드를 호출하면 사운드 재생이 시작됩니다.

사운드 로드 프로세스 모니터링

Flash Player 9 이상, Adobe AIR 1.0 이상

사운드 파일은 용량이 너무 커서 로드하는 데 오랜 시간이 걸릴 수 있습니다. Flash Player 및 AIR를 사용하면 사운드를 완전히 로드하기 전에도 응용 프로그램에서 사운드를 재생할 수 있으며 로드된 사운드 데이터의 양과 이미 재생된 사운드의 양을 표시할 수도 있습니다.

Sound 클래스는 사운드의 로드 진행률을 상대적으로 쉽게 표시하도록 하는 두 개의 이벤트인 ProgressEvent.PROGRESS 및 Event.COMPLETE를 전달합니다. 다음 예제에서는 이러한 이벤트를 사용하여 로드되는 사운드에 대한 진행률 정보를 표시하는 방법을 보여 줍니다.

```
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);
}
```

이 코드는 Sound 객체를 만든 다음 ProgressEvent.PROGRESS 및 Event.COMPLETE 이벤트에 대해 리스너를 이 객체에 추가합니다. Sound.load() 메서드가 호출되고 첫 번째 데이터가 사운드 파일로부터 수신된 후에 ProgressEvent.PROGRESS 이벤트가 발생하고 onSoundLoadProgress() 메서드가 트리거됩니다.

로드된 사운드 데이터의 백분율은 ProgressEvent 객체의 bytesLoaded 속성 값을 bytesTotal 속성 값으로 나눈 값과 같습니다. 동일한 bytesLoaded 및 bytesTotal 속성은 Sound 객체에서도 사용 가능합니다. 위 예제에서는 사운드 로드 진행률에 대한 메시지만 보여 주지만, 사용자가 bytesLoaded 및 bytesTotal 값을 사용하여 진행률 막대 구성 요소(Adobe Flex 프레임워크 또는 Adobe Flash 제작 도구와 함께 제공됨)를 쉽게 업데이트할 수 있습니다.

또한 이 예제에서는 사운드 파일을 로드할 때 응용 프로그램이 어떻게 오류를 인식하고 이에 응답하는지 보여 줍니다. 예를 들어 특정 파일 이름을 가진 사운드 파일을 찾을 수 없는 경우, Sound 객체가 Event.IO_ERROR 이벤트를 전달합니다. 이전 코드에서는 오류가 발생할 때 onIOError() 메서드가 실행되고 간단한 오류 메시지가 표시됩니다.

포함된 사운드를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 파일로부터 사운드를 로드하는 대신 포함된 사운드를 사용하는 것은 응용 프로그램의 사용자 인터페이스에서 지시자로 사용되는 작은 사운드(예: 버튼을 클릭했을 때 재생되는 사운드)에 가장 유용합니다.

사운드 파일을 응용 프로그램에 포함시키면 SWF 파일의 크기는 사운드 파일의 크기만큼 증가합니다. 즉, 응용 프로그램에 대용량 사운드 파일을 포함시키면 SWF 파일의 크기가 지나치게 커질 수 있습니다.

사운드 파일을 응용 프로그램의 SWF 파일에 포함시키는 정확한 방법은 개발 환경에 따라 다릅니다.

포함된 사운드 파일을 Flash에서 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash 제작 도구를 사용하면 다양한 사운드 포맷의 사운드를 가져와서 라이브러리에 심볼로 저장할 수 있습니다. 그런 다음, 사운드를 타임라인 내의 프레임 또는 버튼 상태의 프레임에 할당한 후 비헤이비어와 함께 사용하거나, 사운드를 직접 ActionScript 코드에서 사용할 수도 있습니다. 이 단원에서는 Flash 제작 도구를 활용하여 ActionScript 코드에 포함된 사운드를 사용하는 방법에 대해 설명합니다. 포함된 사운드를 Flash에서 사용하는 기타 방법에 대한 자세한 내용은 **Flash 사용**의 "사운드 가져오기"를 참조하십시오.

Flash 제작 도구를 사용하여 사운드 파일을 포함하려면

1. [파일] > [가져오기] > [라이브러리로 가져오기]를 선택한 다음 사운드 파일을 하나 선택하여 가져옵니다.
2. 가져온 파일 이름을 [라이브러리] 패널에서 마우스 오른쪽 버튼으로 클릭한 후 [속성]을 선택합니다. [ActionScript에 내보내기] 체크 상자를 클릭합니다.
3. [클래스] 필드에 이 포함된 사운드를 ActionScript에서 참조할 때 사용할 이름을 입력합니다. 기본적으로 이 필드의 사운드 파일 이름이 사용됩니다. "DrumSound.mp3"의 경우와 같이 파일 이름에 마침표가 포함된 경우 "DrumSound" 등으로 변경해야 합니다. ActionScript는 클래스 이름에 마침표 문자를 허용하지 않습니다. [기본 클래스] 필드에는 flash.media.Sound가 여전히 표시됩니다.
4. [확인]을 클릭합니다. 이 클래스에 대한 정의를 클래스 경로에서 찾을 수 없다는 대화 상자가 나타날 수 있습니다. [확인]을 클릭하고 계속합니다. 사용자 응용 프로그램의 클래스 경로에 있는 클래스 이름과 일치하지 않는 클래스 이름을 입력하는 경우, flash.media.Sound 클래스에서 상속된 새 클래스가 자동으로 생성됩니다.
5. 포함된 사운드를 사용하려면 ActionScript에서 해당 사운드에 대한 클래스 이름을 참조합니다. 예를 들어 다음 코드는 자동으로 생성된 DrumSound 클래스의 새 인스턴스를 만듭니다.

   ```
   var drum:DrumSound = new DrumSound();
   var channel:SoundChannel = drum.play();
   ```

   DrumSound는 flash.media.Sound 클래스의 하위 클래스이므로 위에 나와 있는 play() 메서드를 포함하여 Sound 클래스의 메서드와 속성을 상속합니다.

포함된 사운드 파일을 Flex에서 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

다음을 비롯한 다양한 방법으로 사운드 에셋을 Flex 응용 프로그램에 포함할 수 있습니다.

- 스크립트에 [Embed] 메타데이터 태그 사용
- MXML에 @Embed 지시문을 사용하여 Button 또는 SoundEffect와 같은 구성 요소의 속성으로 포함된 에셋 할당
- CSS 파일에 @Embed 지시문 사용

이 단원에서는 첫 번째 옵션([Embed] 메타데이터 태그를 사용하여 Flex 응용 프로그램 내의 ActionScript 코드에 사운드를 포함하는 방법)에 대해 설명합니다.

ActionScript 코드에 에셋을 포함하려면 [Embed] 메타데이터 태그를 사용합니다.

기본 소스 폴더나 프로젝트의 빌드 경로에 있는 다른 폴더에 사운드 파일을 배치합니다. 컴파일러가 Embed 메타데이터 태그를 발견하면 포함된 에셋 클래스가 만들어집니다. [Embed] 메타데이터 태그 바로 뒤에 선언하는 Class 데이터 유형의 변수를 통해 클래스에 액세스할 수 있습니다.

다음 코드는 smallSound.mp3라는 사운드를 포함하고 soundClass 변수를 사용하여 해당 사운드와 연관된 포함 에셋 클래스에 대한 참조를 저장합니다. 그런 다음 이 코드에서는 포함된 에셋 클래스의 인스턴스를 만들고 Sound 클래스 인스턴스로 형 변환한 후 해당 인스턴스에서 play() 메서드를 호출합니다.

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

포함된 사운드를 사용하여 Flex 구성 요소의 속성을 설정하려면 Sound 클래스 인스턴스 대신 mx.core.SoundAsset 클래스 인스턴스로 형 변환해야 합니다. SoundAsset 클래스를 사용하는 유사한 예제는 ActionScript 3.0 학습의 "포함된 에셋 클래스"를 참조하십시오.

사운드 파일 스트리밍 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

사운드 파일이나 비디오 파일의 데이터를 로드하고 있는 중에 먼저 로드된 부분을 재생하는 것을 **스트리밍**이라고 합니다. 원격 서버로부터 로드되는 외부 사운드 파일은 스트리밍되는 경우가 많으므로 전체 사운드 데이터의 로드가 완료되지 않아도 사운드를 들을 수 있습니다.

SoundMixer.bufferTime 속성은 사운드 재생 전에 Flash Player 또는 AIR가 수집해야 하는 사운드 데이터의 밀리초를 나타냅니다. 즉, bufferTime 속성이 5000인 경우 Flash Player 또는 AIR가 사운드 파일로부터 최소한 5000밀리초 상당의 데이터를 로드해야 사운드 재생을 시작할 수 있습니다. 기본 SoundMixer.bufferTime 값은 1000입니다.

응용 프로그램은 사운드를 로드할 때 새로운 bufferTime 값을 명시적으로 지정하여 개별 사운드에 대한 전역 SoundMixer.bufferTime 값을 재정의할 수 있습니다. 기본 버퍼링 시간을 재정의하려면 아래와 같이 먼저 SoundLoaderContext 클래스의 새 인스턴스를 만들고 해당 bufferTime 속성을 설정하여 Sound.load() 메서드에 매개 변수로 전달합니다.

```
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();
```

재생을 계속하는 동안 Flash Player 및 AIR는 사운드 버퍼 크기를 동일 수준 이상으로 유지하려고 합니다. 사운드 데이터가 재생 속도보다 빠르게 로드되면 재생은 중단 없이 계속됩니다. 하지만 네트워크 제한으로 인해 데이터 로드 속도가 저하되는 경우 재생 헤드가 사운드 버퍼의 끝에 도달할 수 있습니다. 이렇게 되면 재생이 일시 중단되지만 추가 사운드 데이터가 로드되면 재생이 자동으로 다시 시작됩니다.

Flash Player 또는 AIR가 데이터 로드를 대기하느라 재생이 일시 중단되었는지 확인하려면 Sound.isBuffering 속성을 사용합니다.

동적으로 생성되는 오디오를 사용한 작업

Flash Player 10 이상, Adobe AIR 1.5 이상

참고: 오디오를 동적으로 생성하는 기능은 Flash Player 10 및 Adobe AIR 1.5부터 사용할 수 있습니다.

기존 사운드를 로드하거나 스트리밍하는 대신 동적으로 오디오 데이터를 생성할 수 있습니다. Sound 객체의 sampleData 이벤트에 대한 이벤트 리스너를 지정할 때 오디오 데이터를 생성할 수 있습니다. sampleData 이벤트는 flash.events 패키지의 SampleDataEvent 클래스에 정의되어 있습니다. 이 환경에서 Sound 객체는 파일에서 사운드 데이터를 로드하지 않습니다. 대신 이 객체는 이 이벤트에 할당하는 함수를 통해 스트리밍되어 들어오는 사운드 데이터에 대한 소켓 역할을 합니다.

Sound 객체에 sampleData 이벤트 리스너를 추가하면 이 객체는 데이터를 주기적으로 요청하여 사운드 버퍼에 추가합니다. 이 버퍼에는 Sound 객체가 재생할 데이터가 포함됩니다. Sound 객체의 play() 메서드를 호출하면 새 사운드 데이터를 요청할 때 sampleData 이벤트가 전달됩니다. 이는 Sound 객체가 mp3 데이터를 파일에서 로드하지 않은 경우에만 해당합니다.

SampleDataEvent 객체에는 data 속성이 포함됩니다. 이벤트 리스너에서는 이 data 객체에 ByteArray 객체를 씁니다. 이 객체에 쓰는 바이트 배열은 Sound 객체가 재생하는 버퍼링된 사운드 데이터에 추가됩니다. 버퍼의 바이트 배열은 -1부터 1까지의 부동 소수점 값에 대한 스트림입니다. 각 부동 소수점 값은 사운드 샘플의 한 쪽(왼쪽 또는 오른쪽) 채널 진폭을 나타냅니다. 사운드는 44,100SPS(초당 샘플 수)의 속도로 샘플링됩니다. 각 샘플에는 바이트 배열에 부동 소수점 데이터로 인터리브된 왼쪽 및 오른쪽 채널이 들어 있습니다.

핸들러 함수에서는 ByteArray.writeFloat() 메서드를 사용하여 sampleData 이벤트의 data 속성에 값을 씁니다. 예를 들어 다음 코드에서는 사인파를 생성합니다.

```
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);
    }
}
```

Sound.play()를 호출하면 응용 프로그램에서는 사운드 샘플 데이터를 요청하는 이벤트 핸들러 호출을 시작합니다. 응용 프로그램에서는 사운드가 재생될 때 이벤트를 전송하며 이는 사용자가 데이터 제공을 중단하거나 SoundChannel.stop()을 호출할 때까지 계속됩니다.

이벤트의 지연 시간은 플랫폼마다 다르며 Flash Player 및 AIR의 이후 버전에서 변경될 수 있습니다. 따라서 특정 지연 시간을 사용하는 대신 직접 계산해야 합니다. 지연 시간을 계산하려면 다음 공식을 사용합니다.

```
(SampleDataEvent.position / 44.1) - SoundChannelObject.position
```

이벤트 리스너에 대한 각 호출에서 SampleDataEvent 객체의 data 속성에 2048~8192개의 샘플을 제공합니다. 최상의 성능을 위해 샘플을 최대한 많이(최대 8192개) 제공하는 것이 좋습니다. 제공하는 샘플이 적을수록 재생 중 잡음이 발생하거나 고르지 않게 재생될 가능성이 높습니다. 이러한 비헤이비어는 플랫폼마다 다를 수 있으며 여러 가지 상황(예: 브라우저 크기 조절 시)에서 발생할 수 있습니다. 2048개의 샘플만 제공할 경우 한 플랫폼에서 작동하는 코드가 다른 플랫폼에서 실행될 때는 작동하지 않을 수도 있습니다. 지연 시간을 최대한 낮추려면 사용자가 데이터 양을 선택할 수 있게 하는 것이 좋습니다.

sampleData 이벤트 리스너에 대한 각 호출에서 2048개 미만의 샘플을 제공하면 응용 프로그램은 나머지 샘플을 재생한 후 중단됩니다. 그런 다음 SoundChannel 객체에서 SoundComplete 이벤트를 전달합니다.

mp3 데이터에서 사운드 수정

Flash Player 10 이상, Adobe AIR 1.5 이상

Sound.extract() 메서드를 사용하여 Sound 객체에서 데이터를 추출할 수 있습니다. 이 데이터를 사용하고 수정하여 재생용으로 다른 Sound 객체의 동적 스트림에 쓸 수 있습니다. 예를 들어 다음 코드에서는 로드된 MP3 파일의 바이트를 사용하고 이를 필터 함수 upOctave()를 통해 전달합니다.

```
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;
}
```

생성되는 사운드에 대한 제한

Flash Player 10 이상, Adobe AIR 1.5 이상

sampleData 이벤트 리스너를 Sound 객체와 함께 사용할 때 활성화되는 다른 Sound 메서드는 Sound.extract() 및 Sound.play() 뿐입니다. 다른 메서드나 속성을 호출하면 예외가 발생합니다. SoundChannel 객체의 모든 메서드와 속성은 계속 사용할 수 있습니다.

사운드 재생

Flash Player 9 이상, Adobe AIR 1.0 이상

로드된 사운드의 재생은 아래와 같이 Sound 객체에 대한 Sound.play() 메서드를 호출하는 것처럼 간단할 수 있습니다.

```
var snd:Sound = new Sound(new URLRequest("smallSound.mp3"));
snd.play();
```

ActionScript 3.0을 사용하여 사운드를 재생할 경우 다음 작업을 수행할 수 있습니다.

- 특정 시작 위치에서 사운드 재생
- 사운드를 일시 정지하고 나중에 같은 위치에서 다시 재생 시작
- 사운드 재생이 끝나는 정확한 시간 알아보기
- 사운드의 재생 진행률 추적
- 사운드 재생 중에 볼륨 변경 및 패닝

재생 중에 이러한 작업을 수행하려면 SoundChannel, SoundMixer 및 SoundTransform 클래스를 사용합니다.

SoundChannel 클래스는 단일 사운드의 재생을 제어합니다. SoundChannel.position 속성은 재생 헤드로 간주할 수 있으며 재생 중인 사운드 데이터의 현재 위치를 나타냅니다.

응용 프로그램이 Sound.play() 메서드를 호출하면 SoundChannel 클래스의 새로운 인스턴스가 만들어져 재생을 제어합니다.

응용 프로그램은 특정 시작 위치로부터 해당 위치를 지나면서 사운드를 재생할 수 있습니다. 시작 위치는 기준이 밀리초이며 Sound.play() 메서드의 startTime 매개 변수로 지정됩니다. Sound.play() 메서드의 loops 매개 변수에 숫자 값을 전달하여 사운드를 연속으로 빠르게 반복할 고정 횟수를 지정할 수도 있습니다.

Sound.play() 메서드가 startTime 매개 변수 및 loops 매개 변수를 사용하여 호출되면 다음 코드에서 볼 수 있는 것처럼 사운드가 매번 동일한 시작 위치에서 반복적으로 재생됩니다.

```
var snd:Sound = new Sound(new URLRequest("repeatingSound.mp3"));
snd.play(1000, 3);
```

위 예제에서는 사운드가 시작된지 1초 후 시점에 재생되며 연속 세 번 반복됩니다.

사운드 정지 및 다시 시작

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램이 노래나 포드캐스트 등의 긴 사운드를 재생하는 경우 이러한 사운드의 재생을 일시 정지 및 다시 시작하도록 할 수 있습니다. ActionScript에서는 엄밀한 의미로 사운드를 재생 중에 일시 정지할 수는 없고 중단만 가능합니다. 하지만 어느 위치에서든 사운드 재생을 시작할 수 있습니다. 사운드가 중지된 시점의 위치를 기록하여 나중에 해당 위치에서 사운드 재생을 시작할 수 있습니다.

예를 들어 코드가 다음과 같이 사운드 파일을 로드하여 재생한다고 가정합니다.

```
var snd:Sound = new Sound(new URLRequest("bigSound.mp3"));
var channel:SoundChannel = snd.play();
```

사운드가 재생되는 동안 SoundChannel.position 속성은 사운드 파일에서 현재 재생 중인 위치를 나타냅니다. 응용 프로그램은 다음과 같이 사운드 재생이 중지되기 전에 위치 값을 저장할 수 있습니다.

```
var pausePosition:int = channel.position;
channel.stop();
```

사운드 재생을 다시 시작하려면 이전에 저장된 위치 값을 전달하여 사운드가 중지되었던 위치에서 다시 시작합니다.

```
channel = snd.play(pausePosition);
```

재생 모니터링

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램은 사운드 재생이 언제 중지되는지 인식하여 다른 사운드 재생을 시작하거나 이전 재생 중에 사용된 일부 리소스를 정리할 수 있습니다. SoundChannel 클래스는 사운드 재생이 끝날 때 Event.SOUND_COMPLETE 이벤트를 전달합니다. 응용 프로그램은 아래와 같이 이 이벤트를 수신하여 적절한 액션을 취할 수 있습니다.

```
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.");
}
```

SoundChannel 클래스는 재생 중에 진행률 이벤트를 전달하지 않습니다. 재생 진행률을 보고하려면 응용 프로그램은 자체의 시간 메커니즘을 설정하고 사운드 재생 헤드의 위치를 추적해야 합니다.

재생된 사운드 백분율을 계산하려면 다음과 같이 SoundChannel.position 속성 값을 재생 중인 사운드 데이터의 길이로 나누어야 합니다.

```
var playbackPercent:uint = 100 * (channel.position / snd.length);
```

이 코드는 재생 시작 전에 사운드 데이터가 완전히 로드된 경우에만 정확한 재생 백분율을 보고합니다. Sound.length 속성은 전체 사운드 파일의 최종 크기가 아닌 현재 로드된 사운드 데이터의 크기를 표시합니다. 아직 로드 중인 스트리밍 사운드의 재생 진행률을 추적하기 위해 응용 프로그램은 전체 사운드 파일의 최종 크기를 추정하여 그 값을 계산에 사용해야 합니다. 다음과 같이 Sound 객체의 bytesLoaded 및 bytesTotal 속성을 사용하여 사운드 데이터의 최종 길이를 추정할 수 있습니다.

```
var estimatedLength:int =
    Math.ceil(snd.length / (snd.bytesLoaded / snd.bytesTotal));
var playbackPercent:uint = 100 * (channel.position / estimatedLength);
```

다음 코드는 좀 더 큰 사운드 파일을 로드하며, 재생 진행률을 표시하는 데 Event.ENTER_FRAME 이벤트를 시간 메커니즘으로 사용합니다. 그리고 현재 위치 값을 사운드 데이터의 전체 길이로 나누어 계산되는 재생 백분율을 정기적으로 보고합니다.

```
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);
}
```

사운드 데이터 로드가 시작되고 나면 이 코드는 snd.play() 메서드를 호출하고 결과 SoundChannel 객체를 channel 변수에 저장합니다. 그런 다음 이벤트 리스너를 Event.ENTER_FRAME 이벤트에 대한 기본 응용 프로그램에 추가하고 또 다른 이벤트 리스너를 재생 완료 시 발생하는 Event.SOUND_COMPLETE 이벤트에 대한 SoundChannel 객체에 추가합니다.

응용 프로그램이 해당 애니메이션의 새로운 프레임에 도달할 때마다 onEnterFrame() 메서드가 호출됩니다. onEnterFrame() 메서드는 이미 로드된 데이터 양을 기준으로 사운드 파일의 전체 길이를 추정한 다음 현재 재생 백분율을 계산하여 표시합니다.

전체 사운드가 재생되었으면 onPlaybackComplete() 메서드가 실행되어 Event.ENTER_FRAME 이벤트에 대한 이벤트 리스너를 제거합니다. 그러면 재생이 완료된 후에 진행률 업데이트를 더 이상 표시하지 않습니다.

Event.ENTER_FRAME 이벤트는 초당 여러 번 전달될 수 있습니다. 어떤 경우에는 그렇게 빈번하게 재생 진행률을 표시할 필요가 없을 수 있습니다. 이런 경우에는 사용자의 응용 프로그램에서 flash.util.Timer 클래스를 사용하여 고유한 시간 메커니즘을 설정할 수 있습니다. 1페이지의 “날짜 및 시간을 사용한 작업”을 참조하십시오.

사운드 스트리밍 중단

Flash Player 9 이상, Adobe AIR 1.0 이상

스트리밍되는 사운드 즉, 재생하는 동안 계속 로드되는 사운드의 경우 재생 프로세스에 이상 현상이 발생하기도 합니다. 스트리밍 사운드를 재생 중인 SoundChannel 인스턴스에서 응용 프로그램이 SoundChannel.stop() 메서드를 호출하면 한 프레임에 대한 사운드 재생이 중지되고 다음 프레임에서 사운드를 처음부터 다시 재생합니다. 이러한 현상이 나타나는 이유는 사운드 로드 프로세스가 아직 진행 중이기 때문입니다. 스트리밍 사운드의 로드와 재생을 모두 중단하려면 Sound.close() 메서드를 호출합니다.

사운드 로드 및 재생 시의 보안 고려 사항

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램의 사운드 데이터 액세스 기능은 Flash Player 또는 AIR 보안 모델에 따라 제한됩니다. 각 사운드에는 두 가지 보안 샌드박스의 제한 사항이 적용됩니다. 보안 샌드박스는 내용 자체에 대한 샌드박스("내용 샌드박스")와 사운드를 로드하여 재생하는 응용 프로그램 또는 객체에 대한 샌드박스("소유자 샌드박스")가 있습니다. 응용 프로그램 보안 샌드박스에 있는 AIR 응용 프로그램 내용의 경우 다른 도메인에서 로드된 사운드를 비롯한 모든 사운드를 응용 프로그램 보안 샌드박스의 내용에서 액세스할 수 있습니다. 그러나 다른 보안 샌드박스의 내용은 Flash Player에서 실행되는 내용과 동일한 규칙을 따릅니다. 전반적인 Flash Player 보안 모델 및 샌드박스 정의에 대한 자세한 내용은 948페이지의 "보안"을 참조하십시오.

내용 샌드박스는 id3 속성 또는 SoundMixer.computeSpectrum() 메서드를 사용하여 사운드에서 세부 사운드 데이터를 추출할 수 있는지 여부를 제어하는 것으로, 사운드 파일 자체의 로드 또는 재생을 제한하지는 않습니다.

사운드 파일의 원본 도메인은 내용 샌드박스의 보안 제한 사항을 정의합니다. 일반적으로 사운드 파일을 로드하는 응용 프로그램 또는 객체의 SWF 파일과 동일한 도메인 또는 폴더에 사운드 파일이 있으면 응용 프로그램 또는 객체는 해당 사운드 파일에 액세스할 수 있습니다. 응용 프로그램과는 다른 도메인에서 사운드가 제공되는 경우에도 정책 파일을 사용하여 내용 샌드박스에서 사운드를 가져올 수 있습니다.

응용 프로그램은 checkPolicyFile 속성을 사용하여 SoundLoaderContext 객체를 Sound.load() 메서드에 매개 변수로 전달할 수 있습니다. checkPolicyFile 속성을 true로 설정하면 Flash Player 또는 AIR는 사운드가 로드된 서버에서 정책 파일을 찾습니다. 정책 파일이 존재하고 로드하는 SWF 파일의 도메인에 액세스 권한이 부여되면 SWF 파일은 사운드 파일을 로드하고 Sound 객체의 id3 속성에 액세스한 다음 로드된 사운드에 대한 SoundMixer.computeSpectrum() 메서드를 호출합니다.

소유자 샌드박스는 사운드의 로컬 재생을 제어합니다. 사운드 재생을 시작하는 응용 프로그램 또는 객체가 소유자 샌드박스를 정의합니다.

다음 조건에 맞는 경우 SoundMixer.stopAll() 메서드는 현재 재생 중인 모든 SoundChannel 객체의 사운드를 중지합니다.

- 동일한 소유자 샌드박스 내의 객체에 의해 시작된 사운드
- SoundMixer.stopAll() 메서드를 호출하는 응용 프로그램 또는 객체의 도메인에 대한 액세스를 허용하는 정책 파일을 가진 소스에서 제공되는 사운드

하지만 AIR 응용 프로그램에서 응용 프로그램 보안 샌드박스의 내용(AIR 응용 프로그램과 함께 설치된 내용)은 이러한 보안 제한에 의해 제한되지 않습니다.

SoundMixer.stopAll() 메서드가 모든 재생 사운드를 실제로 중지하는지 알아보기 위해 응용 프로그램이 SoundMixer.areSoundsInaccessible() 메서드를 호출할 수 있습니다. 해당 메서드가 true 값을 반환하면 재생 중인 일부 사운드는 현재 소유자 샌드박스의 제어 범위를 벗어나므로 SoundMixer.stopAll() 메서드에 의해 중지되지 않습니다.

또한 SoundMixer.stopAll() 메서드는 외부 파일로부터 로드된 모든 사운드에 대해 재생 헤드가 계속 진행되지 않도록 중지합니다. 하지만 Flash 제작 도구를 사용하여 FLA 파일에 포함된 사운드 및 타임라인 내 프레임에 연결된 사운드는 애니메이션이 새 프레임으로 이동하면 다시 재생되기 시작합니다.

사운드 볼륨 및 패닝 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

개별 SoundChannel 객체는 사운드에 대해 왼쪽 및 오른쪽 스테레오 채널을 모두 제어합니다. mp3 사운드가 모노인 경우, SoundChannel 객체의 왼쪽 및 오른쪽 스테레오 채널에 동일한 파형이 포함됩니다.

SoundChannel 객체의 leftPeak 및 rightPeak 속성을 사용하여 재생 중인 사운드에 대한 각 스테레오 채널의 진폭을 알 수 있습니다. 이러한 속성은 사운드 파형 자체의 피크 진폭을 표시하는 것으로, 실제 재생 볼륨을 나타내지는 않습니다. 실제 재생 볼륨은 SoundChannel 객체 및 SoundMixer 클래스에 설정된 볼륨 값 및 사운드 웨이브 진폭의 함수입니다.

SoundChannel 객체의 팬 속성은 재생 중에 왼쪽 및 오른쪽 채널 각각에 다른 볼륨 수준을 지정하는 데 사용될 수 있습니다. 팬 속성은 -1 ~ 1 범위의 값을 가질 수 있습니다. 여기서 -1은 오른쪽 채널 음을 소거한 상태에서 왼쪽 채널을 최고 볼륨으로 재생한다는 의미이며, 1은 왼쪽 채널 음을 소거한 상태에서 오른쪽 채널을 최고 볼륨으로 재생한다는 의미입니다. -1과 1 사이의 숫자 값은 왼쪽과 오른쪽 채널 값에 대한 비례 값을 설정하며, 0은 양쪽 채널이 균형 있는 중간 볼륨 수준으로 재생된다는 의미입니다.

다음 코드 예제는 볼륨 값이 0.6이고 팬 값이 -1(즉, 왼쪽 채널이 최고 볼륨으로 재생되고 오른쪽 채널은 음소거)인 SoundTransform 객체를 만듭니다. 이 코드는 SoundTransform 객체를 play() 메서드에 매개 변수로 전달하여 재생 제어를 위해 만들어진 새로운 SoundChannel 객체에 해당 SoundTransform 객체를 적용합니다.

```
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);
```

사운드 재생 도중 SoundTransform 객체의 pan 속성이나 volume 속성을 설정한 다음, 이 객체를 SoundChannel 객체의 soundTransform 속성으로 적용하여 패닝 및 볼륨을 변경할 수 있습니다.

다음 예제에서처럼 SoundMixer 클래스의 soundTransform 속성을 사용하여 한번에 모든 사운드에 대한 전역 볼륨 값과 팬 값을 설정할 수도 있습니다.

```
SoundMixer.soundTransform = new SoundTransform(1, -1);
```

또는 SoundTransform 객체를 사용하여 Microphone 객체(417페이지의 “사운드 입력 캡처” 참조), Sprite 객체 및 SimpleButton 객체에 대해 볼륨 및 팬 값을 설정할 수도 있습니다.

다음 예제에서는 사운드 재생 중에 사운드의 패닝을 왼쪽 채널에서 오른쪽 채널로 교대하고 또 그 반대로 전환합니다.

```
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);
}
```

이 코드는 사운드 파일을 로드한 다음 볼륨이 1(최대 볼륨)로, 팬이 0(좌우 밸런스 동일)으로 설정된 새 SoundTransform 객체를 만듭니다. 그런 다음 snd.play() 메서드를 호출하여 SoundTransform 객체를 매개 변수로 전달합니다.

사운드가 재생되는 동안 onEnterFrame() 메서드가 반복적으로 실행됩니다. onEnterFrame() 메서드는 Math.sin() 함수를 사용하여 -1과 1 사이의 값을 생성합니다. 이 범위는 SoundTransform.pan 속성으로 사용할 수 있는 값에 해당합니다. SoundTransform 객체의 pan 속성은 새로운 값으로 설정되고 채널의 soundTransform 속성은 변경된 SoundTransform 객체를 사용하도록 설정됩니다.

이 예제를 실행하려면 bigSound.mp3라는 파일 이름을 로컬 mp3 파일 이름으로 대체합니다. 그리고 예제를 실행합니다. 오른쪽 채널 볼륨이 작아지면서 왼쪽 채널 볼륨이 커지고, 왼쪽 채널 볼륨이 작아지면서 오른쪽 채널 볼륨이 커집니다.

이 예제에서 SoundMixer 클래스의 soundTransform 속성을 설정하여 동일한 효과를 거둘 수 있습니다. 하지만 이 경우 이 SoundChannel 객체에 의해 재생되는 단일 사운드뿐 아니라 현재 재생되는 모든 사운드의 패닝에 영향을 미칩니다.

사운드 메타데이터를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

mp3 형식을 사용하는 사운드 파일에는 해당 사운드에 대한 추가 데이터가 ID3 태그의 형태로 포함될 수 있습니다.

일부 mp3 파일에는 ID3 메타데이터가 없습니다. mp3 사운드 파일에 ID3 메타데이터가 포함된 경우, Sound 객체가 이 사운드 파일을 로드하면 Event.ID3 이벤트가 전달됩니다. 런타임 오류를 방지하려면 응용 프로그램에서 Event.ID3 이벤트를 수신할 때까지 기다린 후에 로드된 사운드의 Sound.id3 속성에 액세스해야 합니다.

다음 코드는 사운드 파일의 ID3 메타데이터가 로드된 시점을 인식하는 방법을 보여 줍니다.

```
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]);
    }
}
```

이 코드는 Sound 객체를 만든 후 Event.ID3 이벤트를 수신하도록 지시합니다. 사운드 파일의 ID3 메타데이터가 로드되면 onID3InfoReceived() 메서드가 호출됩니다. onID3InfoReceived() 메서드에 전달되는 Event 객체의 대상은 원본 Sound 객체이며, 이 메서드는 Sound 객체의 id3 속성을 가져온 다음, 이름이 지정된 모든 속성을 반복하여 그 값을 추적합니다.

원시 사운드 데이터 액세스

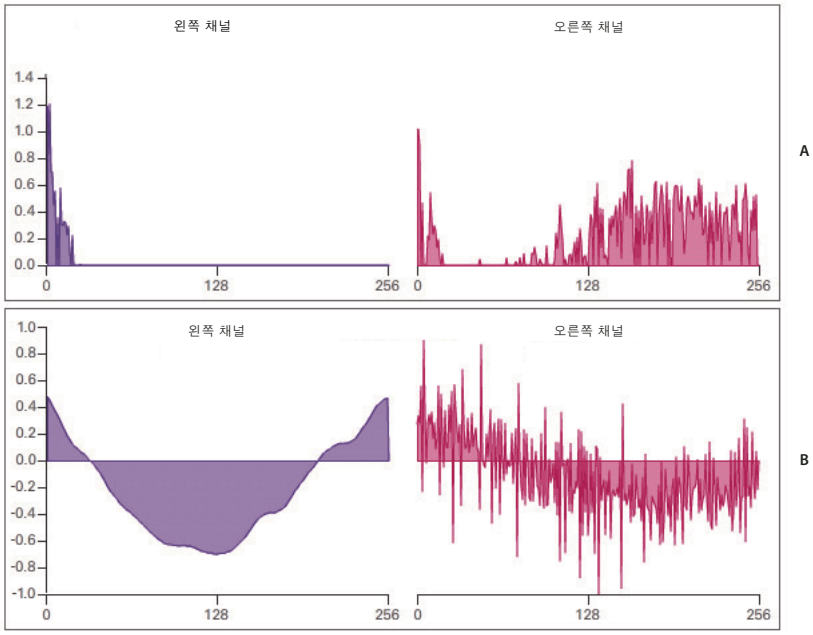
Flash Player 9 이상, Adobe AIR 1.0 이상

SoundMixer.computeSpectrum() 메서드를 사용하면 현재 재생 중인 파형에 대한 원시 사운드 데이터를 응용 프로그램이 읽을 수 있습니다. 둘 이상의 SoundChannel 객체가 현재 재생 중인 경우, SoundMixer.computeSpectrum() 메서드는 함께 믹싱된 모든 SoundChannel 객체의 조합된 사운드 데이터를 표시합니다.

사운드 데이터는 512바이트의 데이터가 포함된 ByteArray 객체로 반환되며 각 객체는 -1과 1 사이의 부동 소수점 값을 포함합니다. 이러한 값은 재생 중인 사운드 파형의 위치 진폭을 나타냅니다. 값은 두 개의 256바이트 그룹으로 제공됩니다. 첫 번째 그룹은 왼쪽 스테레오 채널용이고 두 번째 그룹은 오른쪽 스테레오 채널용입니다.

FFTMode 매개 변수가 true로 설정된 경우, SoundMixer.computeSpectrum() 메서드는 파형 데이터가 아니라 주파수 스펙트럼 데이터를 반환합니다. 주파수 스펙트럼은 가장 낮은 주파수에서 가장 높은 주파수까지 사운드 주파수에 따라 배열된 진폭을 표시합니다. FFT(Fast Fourier Transform)는 파형 데이터를 주파수 스펙트럼 데이터로 변환하는 데 사용됩니다. 결과 주파수 스펙트럼 값은 0에서 약 1.414(2의 제곱근)까지입니다.

다음 그림에서는 FFTMode 매개 변수가 true로 설정되었을 때와 false로 설정되었을 때 computeSpectrum() 메서드로부터 반환된 데이터를 비교합니다. 이 그림에 사용된 사운드 데이터는 왼쪽 채널은 큰 베이스 사운드, 오른쪽 채널은 드럼 사운드에 대한 것입니다.



SoundMixer.computeSpectrum() 메서드에 의해 반환된 값
A. fftMode=true **B.** fftMode=false

또한 computeSpectrum() 메서드는 더 낮은 비트율로 다시 샘플링된 데이터를 반환할 수 있습니다. 이렇게 하면 일반적으로 더 부드러운 파형 데이터 또는 주파수 데이터가 반환되고 세부적인 요소는 제거됩니다. stretchFactor 매개 변수는 computeSpectrum() 메서드 데이터가 샘플링되는 속도를 제어합니다. stretchFactor 매개 변수가 기본값인 0으로 설정된 경우 사운드 데이터는 44.1kHz의 속도로 샘플링됩니다. 이 속도는 stretchFactor 매개 변수의 각 연속 값에서 절반으로 되어 값 1은 22.05kHz, 값 2는 11.025kHz 등으로 속도를 지정합니다. computeSpectrum() 메서드는 더 높은 stretchFactor 값이 사용되어도 스테레오 채널당 256바이트를 반환합니다.

SoundMixer.computeSpectrum() 메서드는 다음과 같은 제한 사항이 있습니다.

- 마이크 또는 RTMP 스트림의 사운드 데이터는 전역 SoundMixer 객체를 통과하지 않으므로 SoundMixer.computeSpectrum() 메서드는 이러한 소스로부터 데이터를 반환하지 않습니다.
- 재생 중인 하나 이상의 사운드가 현재 내용 샌드박스 외부의 소스로부터 제공되는 경우 보안 제한 사항으로 인해 SoundMixer.computeSpectrum() 메서드에서 오류가 발생할 수 있습니다. SoundMixer.computeSpectrum() 메서드의 보안 제한에 대한 자세한 내용은 412페이지의 “사운드 로드 및 재생 시의 보안 고려 사항” 및 970페이지의 “데이터로 로드된 미디어 액세스”를 참조하십시오.

하지만 AIR 응용 프로그램에서 응용 프로그램 보안 샌드박스의 내용(AIR 응용 프로그램과 함께 설치된 내용)은 이러한 보안 제한에 의해 제한되지 않습니다.

간단한 사운드 파형 표시기 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

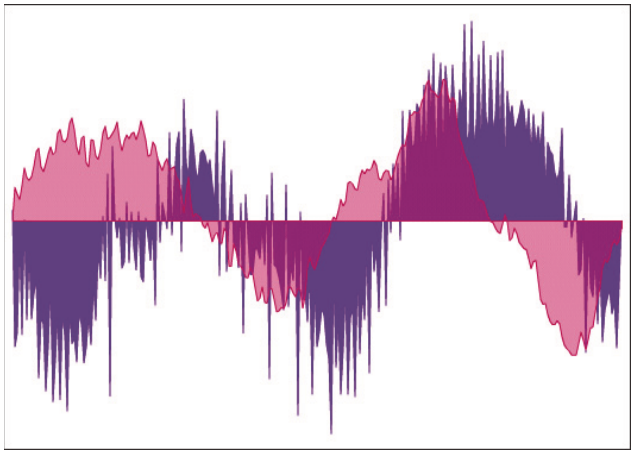
다음 예제에서는 SoundMixer.computeSpectrum() 메서드를 사용하여 각 프레임과 함께 움직이는 사운드 파형 차트를 표시합니다.

```
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);
}
```

이 예제는 사운드 파일을 로드하여 재생한 다음, 사운드 재생 동안 onEnterFrame() 메서드를 트리거할 Event.ENTER_FRAME 이벤트를 수신합니다. onEnterFrame() 메서드는 사운드 웨이브 데이터를 bytes ByteArray 객체에 저장하는 SoundMixer.computeSpectrum() 메서드를 호출하는 것으로 시작됩니다.

사운드 파형은 벡터 드로잉 API를 사용하여 그려집니다. for 루프는 왼쪽 스테레오 채널을 나타내는 처음 256개 데이터 값을 찾은 후 Graphics.lineTo() 메서드를 사용하여 각각의 위치 사이에 차례로 선을 그립니다. 두 번째 for 루프는 다음 세트의 256개 값을 순환한 후 이 값들을 역순으로(오른쪽에서 왼쪽으로) 배치합니다. 결과 파형 형태는 다음 이미지와 같이 흥미로운 거울 이미지 효과를 나타낼 수 있습니다.



사운드 입력 캡처

Flash Player 9 이상, Adobe AIR 1.0 이상

Microphone 클래스를 사용하면 응용 프로그램이 사용자 시스템의 마이크 또는 기타 사운드 입력 장치에 연결하여 입력 오디오를 해당 시스템의 스피커에 브로드캐스팅하거나 오디오 데이터를 Flash Media Server 등의 원격 서버에 보냅니다. 마이크의 원시 오디오 데이터를 액세스하여 기록하거나 처리할 수 있으며, 오디오를 시스템 스피커에 직접 전송하거나 압축된 오디오 데이터를 원격 서버에 전송할 수도 있습니다. 원격 서버로 보내는 데이터에는 Speex 또는 Nellymoser 코덱을 사용할 수 있습니다. Speex 코덱은 Flash Player 10 및 Adobe AIR 1.5부터 지원됩니다.

기타 도움말 항목

Michael Chaize: AIR, Android, and the Microphone

Christophe Coenraets: Voice Notes for Android

마이크 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

Microphone 클래스에는 생성자 메서드가 없습니다. 대신 다음과 같이 정적 Microphone.getMicrophone() 메서드를 사용하여 새 Microphone 인스턴스를 구합니다.

```
var mic:Microphone = Microphone.getMicrophone();
```

매개 변수 없이 Microphone.getMicrophone() 메서드를 호출하면 사용자 시스템에서 검색된 첫 번째 사운드 입력 장치가 반환됩니다.

시스템에는 둘 이상의 사운드 입력 장치가 연결되어 있을 수 있습니다. 응용 프로그램은 Microphone.names 속성을 사용하여 사용 가능한 모든 사운드 입력 장치의 이름 배열을 가져옵니다. 그런 다음 배열에 있는 장치 이름의 인덱스 값과 일치하는 index 매개 변수를 사용하여 Microphone.getMicrophone() 메서드를 호출할 수 있습니다.

시스템에 마이크 또는 기타 사운드 입력 장치가 연결되어 있지 않을 수도 있습니다. Microphone.names 속성 또는 Microphone.getMicrophone() 메서드를 사용하여 사용자의 시스템에 사운드 입력 장치가 설치되어 있는지 여부를 확인합니다. 사용자의 시스템에 사운드 입력 장치가 설치되어 있지 않으면 names 배열 길이는 0이고 getMicrophone() 메서드가 null 값을 반환합니다.

응용 프로그램이 Microphone.getMicrophone() 메서드를 호출하면 Flash Player는 [Flash Player 설정] 대화 상자를 표시합니다. 이 대화 상자에서 사용자는 시스템의 카메라 및 마이크에 대한 Flash Player의 액세스를 허용 또는 거부할 수 있습니다. 이 대화 상자에서 [허용] 또는 [거부] 버튼을 클릭하면 StatusEvent가 전달됩니다. 다음 예제에서와 같이, 해당 StatusEvent 인스턴스의 code 속성은 마이크 액세스의 허용 또는 거부를 나타냅니다.

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

StatusEvent.code 속성에는 "Microphone.Unmuted"(액세스가 허용된 경우) 또는 "Microphone.Muted"(액세스가 거부된 경우)가 포함됩니다.

사용자가 마이크 액세스를 허용 또는 거부할 때 Microphone.muted 속성은 각각 true 또는 false로 설정됩니다. 그러나 muted 속성은 StatusEvent가 전달되기 전에는 Microphone 인스턴스에 설정되지 않으므로 응용 프로그램에서 Microphone.muted 속성을 확인하려면 또한 StatusEvent.STATUS 이벤트가 전달될 때까지 대기해야 합니다.

Flash Player에서 설정 대화 상자를 표시하려면 응용 프로그램 창의 크기는 최소한 215×138 픽셀 이상이 되어야 합니다. 그렇지 않으면 액세스는 자동으로 거부됩니다.

AIR 응용 프로그램 샌드박스에서 실행되는 내용은 마이크에 액세스하는 데 사용자의 권한을 얻을 필요가 없습니다. 따라서 마이크의 음소거 및 음소거 해제에 대한 status 이벤트는 전달되지 않습니다. 응용 프로그램 샌드박스 외부의 AIR에서 실행되는 내용은 사용자의 권한을 필요로 하므로 이러한 status 이벤트가 전달될 수 있습니다.

마이크 오디오를 로컬 스피커로 라우팅

Flash Player 9 이상, Adobe AIR 1.0 이상

매개 변수 값을 true로 하여 Microphone.setLoopback() 메서드를 호출하면 마이크의 오디오 입력을 로컬 시스템 스피커로 라우팅할 수 있습니다.

로컬 마이크의 사운드가 로컬 스피커로 라우팅되면 오디오 피드백 루프를 만들어 커다란 빽 소리를 일으키고 잠재적으로 사운드 하드웨어의 손상을 초래할 위험이 있습니다. 매개 변수 값을 true로 하여 Microphone.setUseEchoSuppression() 메서드를 호출하면 오디오 피드백이 발생할 위험이 완전히 배제되지는 않지만 낮아집니다. 사용자가 헤드폰을 사용하거나 스피커 이외의 다른 장치를 사용하여 사운드를 재생하고 있음을 확신할 수 없는 경우라면 Microphone.setLoopback(true)을 호출하기 전에 항상 Microphone.setUseEchoSuppression(true)을 호출하는 것이 좋습니다.

다음 코드는 로컬 마이크의 오디오를 로컬 시스템 스피커로 라우팅하는 방법을 보여 줍니다.

```
var mic:Microphone = Microphone.getMicrophone();
mic.setUseEchoSuppression(true);
mic.setLoopBack(true);
```

마이크 오디오 변경

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램은 마이크로부터 전달되는 오디오 데이터를 두 가지 방법으로 변경할 수 있습니다. 먼저 입력 사운드의 게인을 변경할 수 있습니다. 게인은 입력 값을 지정된 양만큼 곱하여 더 크거나 작은 사운드를 생성합니다. Microphone.gain 속성은 0부터 100까지의 숫자 값을 받습니다. 값을 50으로 지정하면 1을 곱한 것과 같게 되어 보통 볼륨을 생성합니다. 값을 0으로 지정하면 0을 곱한 것과 같게 되어 입력 오디오를 제거합니다. 값을 50 이상으로 지정하면 보통 볼륨보다 높은 볼륨을 생성합니다.

응용 프로그램에서 입력 오디오의 샘플링 속도를 변경할 수도 있습니다. 샘플링 속도가 높을수록 사운드 품질은 향상되지만, 전송과 저장에 더 많은 리소스를 사용하는 좀 더 밀집된 데이터 스트림이 생성됩니다. Microphone.rate 속성은 kHz로 측정되는 오디오 샘플링 속도를 나타냅니다. 기본 샘플링 속도는 8kHz입니다. 마이크가 더 높은 속도를 지원하는 경우 Microphone.rate 속성 값을 8kHz 이상으로 설정할 수 있습니다. 예를 들어 Microphone.rate 속성의 값을 11로 설정하면 샘플링 속도가 11kHz로 지정되고, 22로 설정하면 샘플링 속도가 22kHz로 지정됩니다. 사용할 수 있는 샘플링 속도는 선택하는 코덱에 따라 달라집니다. Nellymoser 코덱을 사용할 경우에는 샘플링 속도로 5, 8, 11, 16, 22 및 44kHz를 지정할 수 있고, Speex 코덱을 사용할 경우(Flash Player 10 및 Adobe AIR 1.5부터 지원) 16kHz만 사용할 수 있습니다.

마이크 활동 감지

Flash Player 9 이상, Adobe AIR 1.0 이상

대역폭과 처리 리소스를 보존하기 위해 Flash Player는 마이크가 사운드를 전송하지 않는 때를 감지하려고 합니다. 마이크의 활동 레벨이 일정 시간 동안 묵음 레벨 임계값 미만을 유지하면 Flash Player는 오디오 입력 전송을 중지하고 대신 단순 ActivityEvent를 전달합니다. Speex 코덱을 사용할 경우(Flash Player 10 이상 및 Adobe AIR 1.5 이상에서 지원) 묵음 레벨을 0으로 설정해야 응용 프로그램에서 오디오 데이터가 연속적으로 전송됩니다. Speex 음성 활동 검출은 대역폭을 자동으로 줄입니다.

참고: Microphone 객체는 응용 프로그램에서 마이크를 모니터링하고 있는 경우 Activity 이벤트만 전달합니다. 따라서 setLoopBack(true)을 호출하지 않는 경우 샘플 데이터 이벤트의 리스너를 추가하거나 마이크를 NetStream 객체에 연결하면 activity 이벤트는 전달되지 않습니다.

Microphone 클래스의 다음 세 가지 속성은 활동을 모니터링하고 제어합니다.

- 읽기 전용 activityLevel 속성은 마이크가 감지하는 사운드의 양을 0 ~ 100의 등급으로 나타냅니다.
- silenceLevel 속성은 마이크 활성화에 필요한 사운드의 양을 지정하고 ActivityEvent.ACTIVITY 이벤트를 전달합니다. silenceLevel 속성도 0 ~ 100의 등급을 사용하고 기본값은 10입니다.
- silenceTimeout 속성은 마이크가 현재 사용 중이지 않음을 알리기 위해 ActivityEvent.ACTIVITY 이벤트가 전달될 때까지 활동 레벨이 묵음 수준 미만을 유지해야 하는 시간을 밀리초 단위로 정의합니다. 기본 silenceTimeout 값은 2000입니다.

Microphone.silenceLevel 속성과 Microphone.silenceTimeout 속성은 읽기 전용이지만 Microphone.setSilenceLevel() 메서드를 사용하여 이 값을 변경할 수 있습니다.

새로운 활동이 감지되었을 때 마이크를 활성화하는 프로세스에서 약간의 지연이 발생하는 경우도 있습니다. 마이크를 항상 활성화시켜 두면 이러한 활성화 지연을 방지할 수 있습니다. 응용 프로그램이 silenceLevel 매개 변수를 0으로 설정하여 Microphone.setSilenceLevel() 메서드를 호출하면 Flash Player는 사운드가 감지되지 않더라도 마이크의 활성 상태를 유지하고 오디오 데이터를 계속 수집합니다. 반대로, silenceLevel 매개 변수를 100으로 설정하면 마이크가 전혀 활성화되지 않습니다.

다음 예제는 마이크에 대한 정보를 표시하고 Microphone 객체가 전달한 activity 이벤트 및 status 이벤트를 보고합니다.

```
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);
}
```

위 예제를 실행할 때 시스템 마이크에 대고 말하거나 소리를 낸 다음, 콘솔이나 디버그 윈도우에 표시되는 결과 trace 문을 확인합니다.

미디어 서버에서의 오디오 송수신

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Media Server와 같은 스트리밍 미디어 서버와 함께 ActionScript를 사용할 때 추가 오디오 기능을 사용할 수 있습니다.

특히 응용 프로그램은 Microphone 객체를 NetStream 객체에 연결하여 마이크에서 서버로 직접 데이터를 전송할 수 있습니다. 또한 오디오 데이터는 서버에서 응용 프로그램으로 스트리밍되어 MovieClip의 일부로 재생되거나 Video 객체를 사용하여 재생될 수 있습니다.

Flash Player 10 및 Adobe AIR 1.5부터는 Speex 코덱을 사용할 수 있습니다. 미디어 서버로 전송되는 압축 오디오에 코덱을 설정하려면 Microphone 객체의 codec 속성을 설정합니다. 이 속성은 SoundCodec 클래스에 열거되는 두 개의 값을 가질 수 있습니다. 코덱 속성을 SoundCodec.SPEEX로 설정하면 오디오를 압축하는 데 Speex 코덱이 선택됩니다. 이 속성을 SoundCodec.NELLYMOSER(기본값)로 설정하면 오디오를 압축하는 데 Nellymoser 코덱이 선택됩니다.

자세한 내용은 www.adobe.com/go/learn_fms_docs_kr에서 온라인으로 제공되는 Flash Media Server 설명서를 참조하십시오.

마이크 사운드 데이터 캡처

Flash Player 10.1 이상, Adobe AIR 2 이상

Flash Player 10.1 및 AIR 2 이상 버전에서는 마이크의 데이터를 부동 소수점 값의 바이트 배열로 캡처할 수 있습니다. 각 값은 모노 오디오 데이터의 샘플을 나타냅니다.

마이크 데이터를 가져오려면 Microphone 객체의 sampleData 이벤트에 대한 이벤트 리스너를 설정합니다. Microphone 객체는 마이크 버퍼에 사운드 샘플이 채워지는 경우 정기적으로 sampleData 이벤트를 전달합니다. SampleDataEvent 객체에는 사운드 샘플의 바이트 배열인 data 속성이 포함됩니다. 샘플은 각각 부동 소수점 값으로 표시되며, 각각 모노 사운드 샘플을 나타냅니다.

다음 코드에서는 마이크 사운드 데이터를 soundBytes라는 이름의 ByteArray 객체로 캡처합니다.

```
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);
    }
}
```

샘플 바이트를 Sound 객체의 재생 오디오로 다시 사용할 수 있습니다. 이렇게 할 경우 Microphone 객체의 rate 속성을 Sound 객체에서 사용되는 샘플링 속도인 44로 설정해야 합니다. 보다 낮은 속도로 캡처된 마이크 샘플을 Sound 객체에 필요한 44kHz 속도로 변환할 수도 있습니다. 또한 Microphone 객체는 모노 샘플을 캡처하지만 Sound 객체에서는 스테레오 사운드가 사용된다는 점에 유의해야 합니다. 따라서 Microphone 객체에서 캡처한 각 바이트를 Sound 객체에 두 번씩 기록해야 합니다. 다음 예제에서는 4초 분량의 마이크 데이터를 캡처하고 이를 Sound 객체를 사용하여 재생합니다.

```
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.");
}
```

사운드 샘플 데이터의 사운드를 재생하는 방법에 대한 자세한 내용은 407페이지의 “동적으로 생성되는 오디오를 사용한 작업”을 참조하십시오.

사운드 예제: 포드캐스트 플레이어

Flash Player 9 이상, Adobe AIR 1.0 이상

포드캐스트는 인터넷이나 주문 또는 구독을 통해 배포되는 사운드 파일입니다. 일반적으로 포드캐스트는 포드캐스트 채널이라고도 하는 시리즈의 일부로 제작됩니다. 포드캐스트 에피소드는 길이가 1분이 될 수도 있고 몇 시간이 될 수도 있으므로 일반적으로 스트리밍으로 재생됩니다. 포드캐스트 에피소드는 아이템이라고도 하며 보통 mp3 파일 형식으로 전달됩니다. 비디오 포드캐스트도 널리 사용되지만 이 샘플 응용 프로그램은 mp3 파일을 사용하는 오디오 포드캐스트만 재생합니다.

이 예제는 모든 기능을 갖춘 포드캐스트 수집기 응용 프로그램이 아닙니다. 예를 들어 특정 포드캐스트에 대한 구독을 관리하지 않으며 사용자가 이전에 들은 포드캐스트가 다음에 응용 프로그램이 실행될 때 기억되지 않습니다. 이 예제는 더 복잡한 기능의 포드캐스트 수집기 작성을 위한 토대가 될 수 있습니다.

Podcast Player 예제는 다음과 같은 ActionScript 프로그래밍 기술을 보여 줍니다.

- 외부 RSS 피드 읽기 및 해당 XML 내용 파싱
- SoundFacade 클래스를 만들어 사운드 파일을 간편하게 로드 및 재생
- 사운드 재생 진행률 표시
- 사운드 재생 일시 정지 및 다시 시작

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. Podcast Player 응용 프로그램 파일은 Samples/PodcastPlayer 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
PodcastPlayer.mxml 또는 PodcastPlayer.fla	Flex(MXML) 또는 Flash(FLA)용 응용 프로그램의 사용자 인터페이스입니다.
comp/example/programmingas3/podcastplayer/PodcastPlayer.as	포드캐스트 플레이어용 사용자 인터페이스 논리를 포함하는 문서 클래스입니다(Flash 전용).
SoundPlayer.mxml	Flex에 대해서만 재생 버튼과 진행률 막대를 표시하고 사운드 재생을 제어하는 MXML 구성 요소입니다.
main.css	응용 프로그램 사용자 인터페이스의 스타일입니다(Flex 전용).
images/	버튼 스타일에 대한 아이콘입니다(Flex 전용).
comp/example/programmingas3/podcastplayer/SoundPlayer.as	사운드 플레이어용 사용자 인터페이스 논리를 포함하는 SoundPlayer 동영상 클립 심볼 클래스입니다(Flash 전용).
comp/example/programmingas3/podcastplayer/PlayButtonRenderer.as	데이터 격자 셀에 재생 버튼을 표시하기 위한 사용자 정의 셀 렌더러입니다(Flash 전용).
com/example/programmingas3/podcastplayer/RSSBase.as	RSSChannel 클래스와 RSSItem 클래스에 대한 일반 속성 및 메서드를 제공하는 기본 클래스입니다.
com/example/programmingas3/podcastplayer/RSSChannel.as	RSS 채널에 대한 데이터를 보관하는 ActionScript 클래스입니다.
com/example/programmingas3/podcastplayer/RSSItem.as	RSS 항목에 대한 데이터를 보관하는 ActionScript 클래스입니다.
com/example/programmingas3/podcastplayer/SoundFacade.as	응용 프로그램에 대한 기본 ActionScript 클래스입니다. Sound 클래스와 SoundChannel 클래스의 메서드 및 이벤트를 캡슐화하고 재생 일시 정지 및 다시 시작에 대한 지원을 추가합니다.

파일	설명
com/example/programmingas3/podcastplayer/URLService.as	원격 URL로부터 데이터를 가져오는 ActionScript 클래스입니다.
playerconfig.xml	포드캐스트 채널을 나타내는 RSS 피드 목록이 포함된 XML 파일입니다.
comp/example/programmingas3/utils/DateUtil.as	간편한 날짜 형식을 위해 사용되는 클래스입니다(Flash 전용).

포드캐스트 채널용 RSS 데이터 읽기

Flash Player 9 이상, Adobe AIR 1.0 이상

Podcast Player 응용 프로그램은 몇 개의 포드캐스트 채널과 해당 에피소드에 대한 정보를 읽는 것으로 시작합니다.

1. 먼저 응용 프로그램은 포드캐스트 채널 목록이 포함된 XML 구성 파일을 읽고 채널 목록을 사용자에게 표시합니다.

2. 사용자가 포드캐스트 채널 중 하나를 선택하면 응용 프로그램은 채널에 대한 RSS 피드를 읽고 채널 에피소드 목록을 표시합니다.

이 예제는 URLLoader 유틸리티 클래스를 사용하여 원격 위치 또는 로컬 파일로부터 텍스트 기반 데이터를 가져옵니다. Podcast Player는 먼저 playerconfig.xml 파일로부터 XML 형식의 RSS 피드 목록을 가져오기 위한 URLLoader 객체를 만듭니다. 그리고 사용자가 목록에서 특정 피드를 선택하면 새로운 URLLoader 객체가 만들어져 해당 피드의 URL로부터 RSS 데이터를 읽습니다.

SoundFacade 클래스를 사용하여 사운드 로드 및 재생 단순화

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0 사운드 아키텍처는 강력하지만 복잡합니다. 기본적인 사운드 로드와 재생 기능만 필요한 응용 프로그램은 간단한 메서드 호출 및 이벤트 집합을 제공하여 일부 복잡한 기능을 숨기는 클래스를 사용할 수 있습니다. 이러한 클래스는 소프트웨어 디자인 패턴 분야에서 **facade**라고 합니다.

SoundFacade 클래스는 다음과 같은 작업을 수행하는 단일 인터페이스를 제공합니다.

- Sound 객체, SoundLoaderContext 객체 및 SoundMixer 클래스를 사용하여 사운드 파일 로드
- Sound 객체 및 SoundChannel 객체를 사용하여 사운드 파일 재생
- 재생 진행률 이벤트 전달
- Sound 객체 및 SoundChannel 객체를 사용하여 사운드 재생 일시 정지 및 다시 시작

SoundFacade 클래스는 대부분의 ActionScript 사운드 클래스 기능을 보다 단순하게 제공하려고 합니다.

다음 코드는 클래스 선언, 클래스 속성 및 SoundFacade() 생성자 메서드를 보여 줍니다.

```
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();
        }
    }
```

SoundFacade 클래스는 EventDispatcher 클래스를 확장하여 자체의 이벤트를 전달하도록 합니다. 이 클래스 코드는 먼저 Sound 객체와 SoundChannel 객체의 속성을 선언합니다. 이 클래스는 사운드를 스트리밍할 때 사용할 bufferTime 속성과 사운드 파일의 URL 값도 저장합니다. 또한 로딩 및 재생 비헤이비어에 영향을 미치는 다음과 같은 몇 가지 부울 매개 변수 값을 받습니다.

- autoLoad 매개 변수는 객체가 만들어지면 바로 사운드 로드를 시작하도록 객체에 지시합니다.
- autoPlay 매개 변수는 충분한 사운드 데이터가 로드되면 바로 재생을 시작함을 나타냅니다. 스트리밍 사운드의 경우에는 bufferTime 속성에 지정된 충분한 데이터가 로드되었을 때 재생이 시작됩니다.
- streaming 매개 변수는 로드가 완료되기 전에 이 사운드 파일 재생을 시작할 수 있음을 나타냅니다.

bufferTime 매개 변수의 기본값은 -1입니다. 생성자 메서드가 bufferTime 매개 변수에서 음수 값을 감지하면 bufferTime 속성이 SoundMixer.bufferTime 값으로 설정됩니다. 이렇게 되면 필요한 전역 SoundMixer.bufferTime 값으로 응용 프로그램의 기본값이 설정될 수 있습니다.

autoLoad 매개 변수가 true로 설정된 경우, 생성자 메서드는 즉시 다음 load() 메서드를 호출하여 사운드 파일 로드를 시작합니다.

```
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);
}
```

load() 메서드는 새로운 Sound 객체를 생성한 후 모든 중요 사운드 이벤트에 대한 리스너를 추가합니다. 그런 다음 Sound 객체에 사운드 파일 로드를 지시하고 SoundLoaderContext 객체를 사용하여 bufferTime 값을 전달하도록 합니다.

url 속성은 변경 가능하므로, SoundFacade 인스턴스를 사용하여 여러 사운드 파일을 연속적으로 재생할 수 있습니다. url 속성을 변경하고 load() 메서드를 호출하기만 하면 새 사운드 파일이 로드됩니다.

다음 세 가지 이벤트 리스너 메서드는 SoundFacade 객체가 로드 프로세스를 추적하고 사운드 재생 시점을 결정하는 방법을 보여 줍니다.

```
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();
    }
}
```

사운드 로드가 시작되면 onLoadOpen() 메서드가 실행됩니다. 사운드가 스트리밍 모드에서 재생될 수 있는 경우 onLoadComplete() 메서드는 즉시 isReadyToPlay 플래그를 true로 설정합니다. isReadyToPlay 플래그는 사용자가 재생 버튼을 클릭하는 등의 액션을 수행할 때 응용 프로그램이 사운드 재생을 시작할 수 있는지 여부를 결정합니다. SoundChannel 클래스는 사운드 데이터의 버퍼링을 관리하므로 play() 메서드를 호출하기 전에 충분한 데이터가 로드되었는지 명시적으로 확인할 필요가 없습니다.

onLoadProgress() 메서드는 로드 프로세스 중에 정기적으로 실행됩니다. 이 메서드는 이 SoundFacade 객체를 사용하는 코드가 사용할 해당 ProgressEvent 객체의 복제본을 전달합니다.

사운드 데이터가 완전히 로드되었으면 onLoadComplete() 메서드가 실행되어, 필요한 경우 비스트리밍 사운드에 대해 play() 메서드를 호출합니다. play() 메서드는 아래와 같습니다.

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

사운드 재생 준비가 되었으면 play() 메서드가 Sound.play() 메서드를 호출합니다. 결과 SoundChannel 객체는 sc 속성에 저장됩니다. 그런 다음 play() 메서드는 재생 진행률 이벤트를 정기적으로 전달하는 데 사용될 Timer 객체를 만듭니다.

재생 진행률 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

재생 모니터링을 구동하는 Timer 객체 생성은 복잡한 작업으로 한 번만 코딩해야 합니다. 이 Timer 논리를 SoundFacade 클래스 등의 다시 사용할 수 있는 클래스에 캡슐화하면 사운드 로드와 재생 시 동일한 종류의 진행률 이벤트를 응용 프로그램이 수신할 수 있습니다.

SoundFacade.play() 메서드가 생성한 Timer 객체는 매초 TimerEvent 인스턴스를 전달합니다. 다음 onPlayTimer() 메서드는 새 TimerEvent가 도착할 때마다 실행됩니다.

```
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);
}
```

onPlayTimer() 메서드는 409페이지의 “재생 모니터링” 단원에 설명된 크기 예측 기법을 구현합니다. 그런 다음, 이벤트 유형이 SoundFacade.PLAY_PROGRESSbytesLoaded 속성이 SoundChannel 객체의 현재 위치로 설정되고 bytesTotal 속성이 사운드 데이터의 추정된 길이로 설정된 새로운 ProgressEvent 인스턴스를 만듭니다.

재생 일시 정지 및 다시 시작

Flash Player 9 이상, Adobe AIR 1.0 이상

앞에서 설명한 SoundFacade.play() 메서드는 사운드 데이터의 시작 위치에 해당하는 pos 매개 변수를 받습니다. pos 값이 0이면 사운드는 처음부터 재생을 시작합니다.

아래와 같이 SoundFacade.stop() 메서드도 pos 매개 변수를 받습니다.

```
public function stop(pos:int = 0):void
{
    if (this.isPlaying)
    {
        this.pausePosition = pos;
        this.sc.stop();
        this.playTimer.stop();
        this.isPlaying = false;
    }
}
```

SoundFacade.stop() 메서드가 호출될 때마다 동일한 사운드의 재생을 사용자가 다시 시작하려는 경우 응용 프로그램이 재생 헤드의 위치를 지정할 수 있도록 pausePosition 속성이 설정됩니다.

아래와 같은 SoundFacade.pause() 및 SoundFacade.resume() 메서드는 SoundFacade.stop() 및 SoundFacade.play() 메서드를 각각 호출하여 pos 매개 변수 값을 전달합니다.

```
public function pause():void
{
    stop(this.sc.position);
}

public function resume():void
{
    play(this.pausePosition);
}
```

pause() 메서드는 현재의 SoundChannel.position 값을 play() 메서드에 전달하고 play() 메서드는 해당 값을 pausePosition 속성에 저장합니다. resume() 메서드는 pausePosition 값을 시작 위치로 사용하여 동일한 사운드의 재생을 다시 시작합니다.

Podcast Player 예제 확장

Flash Player 9 이상, Adobe AIR 1.0 이상

이 예제에서는 재사용 가능한 SoundFacade 클래스의 기능을 보여 주는 Podcast Player의 핵심을 나타냅니다. 이 응용 프로그램의 기능을 확장하기 위해 다음과 같은 기타 기능을 추가할 수 있습니다.

- 사용자가 응용 프로그램을 다음에 실행할 때 사용할 수 있는 SharedObject 인스턴스에 각 에피소드와 관련한 피드 및 사용 정보 목록을 저장합니다.
- 사용자가 자신의 RSS 피드를 포드캐스트 채널 목록에 추가할 수 있도록 합니다.
- 사용자가 에피소드를 중지하거나 종료했을 때 재생 헤드의 위치를 기억하여 사용자가 다음에 응용 프로그램을 실행할 때 해당 위치에서 다시 시작할 수 있도록 합니다.
- 사용자가 인터넷에 연결되어 있지 않을 때 오프라인 상태에서 수신하도록 에피소드 mp3 파일을 다운로드합니다.
- 포드캐스트 채널에서 새로운 에피소드가 있는지 정기적으로 확인하고 에피소드 목록을 자동으로 업데이트하는 구독 기능을 추가합니다.
- Odeo.com 등의 포드캐스트 호스팅 서비스로부터 API를 사용하여 포드캐스트 검색 및 탐색 기능을 추가합니다.

25장: 비디오를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash 비디오는 인터넷에서 두각을 나타내고 있는 기술 중 하나입니다. 그러나 아래쪽에 진행률 막대와 제어 버튼이 있는 직사각형 화면에 비디오를 표시하는 것은 비디오를 사용한 작업 중 하나일 뿐입니다. ActionScript를 통해 비디오 로드, 표시 및 재생을 세밀하게 조정하고 제어할 수 있습니다.

비디오의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe® Flash® Player 및 Adobe® AIR™의 중요한 기능 중 하나는 이미지, 애니메이션, 텍스트 등의 기타 시각적 내용을 조작하는 것과 동일한 방법으로 ActionScript를 사용하여 비디오 정보를 표시하고 조작할 수 있는 것입니다. Adobe Flash CS4 Professional에서 Flash 비디오(FLV) 파일을 만드는 경우 일반적인 재생 컨트롤을 포함한 스킨을 선택하는 옵션이 있습니다. 그러나 제공되는 옵션 외에도 다양한 기능을 활용할 수 있습니다. ActionScript를 사용하면 비디오 로드, 표시 및 재생을 세밀하게 제어하여 자신만의 비디오 플레이어 스킨을 만들거나 자신이 원하는 새로운 방식으로 비디오를 사용할 수 있습니다. ActionScript에서 비디오를 사용한 작업을 수행하려면 다음과 같은 여러 클래스를 조합해서 사용해야 합니다.

- Video 클래스: 스테이지에 있는 클래식 비디오 내용 상자가 Video 클래스의 인스턴스입니다. Video 클래스는 표시 객체이므로 위치 지정, 변형 적용, 필터 및 블렌드 모드 적용을 비롯하여 다른 표시 객체에 적용할 수 있는 동일한 기술로 조작할 수 있습니다.
- StageVideo 클래스: 일반적으로 Video 클래스는 소프트웨어 디코딩 및 렌더링을 사용합니다. 장치가 GPU 하드웨어 가속을 지원할 경우 StageVideo 클래스로 전환하면 응용 프로그램이 하드웨어 가속 프레젠테이션의 이점을 최대한 활용할 수 있습니다. StageVideo API에는 StageVideo 및 Video 객체 간 전환 시기를 코드에 알려 주는 이벤트 집합이 포함되어 있습니다. 스테이지 비디오는 비디오를 재생하는 데 사소한 제약이 있습니다. 응용 프로그램에서 이 제약을 수용할 수 있으면 StageVideo API를 구현하십시오. 465페이지의 “지침 및 제한 사항”을 참조하십시오.
- NetStream 클래스: ActionScript에서 제어할 비디오 파일을 로드할 때 NetStream 인스턴스는 비디오 내용의 소스(이 경우에는 비디오 데이터 스트림)를 나타냅니다. NetStream 인스턴스를 사용하려면 NetConnection 객체도 사용해야 합니다. NetConnection 객체는 비디오 파일에 대한 연결이며 비디오 데이터가 제공되는 터널과 같습니다.
- Camera 클래스: 사용자의 컴퓨터에 연결된 카메라의 비디오 데이터를 사용하여 작업할 때 Camera 인스턴스는 비디오 내용의 소스, 즉 사용자의 카메라와 해당 카메라에서 제공하는 비디오 데이터를 나타냅니다. Flash Player 11.4 및 AIR 3.4에서 새로 추가된 클래스로, 카메라를 사용하여 StageVideo 내용을 공급할 수 있습니다.

외부 비디오를 로드할 때는 표준 웹 서버에서 파일을 로드하여 점진적으로 다운로드하거나, Adobe의 Flash® Media Server와 같은 특수한 서버에 의해 전달된 비디오를 스트리밍할 수도 있습니다.

중요한 개념 및 용어

큐 포인트 비디오 파일에서 시간상 특정 순간에 삽입할 수 있는 표시자입니다. 예를 들어 특정 시점을 찾기 위한 책갈피로 큐 포인트를 사용하거나 특정 순간과 관련된 추가 데이터를 제공하기 위해 큐 포인트를 사용할 수 있습니다.

인코딩 특정 형식의 비디오 데이터를 다른 비디오 데이터 형식으로 변환하는 프로세스입니다. 예를 들어 고해상도 소스 비디오를 인터넷으로 전송하는 데 적합한 형식으로 변환할 수 있습니다.

프레임 비디오 정보의 한 세그먼트로, 각 프레임은 시간상 한 순간의 스냅샷을 나타내는 스틸 이미지와 같습니다. 프레임을 고속으로 연속 재생하면 움직이는 것처럼 보입니다.

키프레임 프레임의 전체 정보가 포함된 비디오 프레임입니다. 키프레임 다음에 오는 다른 프레임에는 전체 프레임의 정보가 아닌 키프레임과 구분되는 정보만 포함됩니다.

메타데이터 비디오 파일 내에 포함되며 비디오가 로드된 경우 검색할 수 있는 비디오 파일에 대한 정보입니다.

점진적 다운로드 표준 웹 서버에서 비디오 파일을 전송하는 경우 비디오 데이터가 점진적 다운로드를 통해 로드됩니다. 즉, 비디오 정보가 차례로 로드됩니다. 점진적 다운로드를 사용하면 전체 파일이 다운로드되기 전에 비디오 재생을 시작할 수 있다는 장점이 있지만 아직 로드되지 않은 비디오 부분으로는 이동할 수 없습니다.

스트리밍 "트루 스트리밍"이라고도 하는 스트리밍 기술을 사용하여 특수 비디오 서버를 통해 인터넷으로 비디오를 전송하는 방법으로, 점진적 다운로드 대신 사용할 수 있습니다. 스트리밍을 사용하면 해당 컴퓨터에 전체 비디오가 한 번에 다운로드되지 않습니다. 다운로드 속도를 높이기 위해 컴퓨터에서는 언제나 전체 비디오 정보의 일부만을 필요로 합니다. 특수 서버에서 비디오 내용 전송을 제어하므로 비디오 액세스를 위해 다운로드될 때까지 기다릴 필요 없이 언제든지 비디오의 모든 부분에 액세스할 수 있습니다.

비디오 형식 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player와 Adobe AIR에서는 Adobe FLV 비디오 형식 외에도 MPEG-4 표준 파일 형식 내에서 H.264 및 HE-AAC로 인코딩된 비디오와 오디오를 지원합니다. 이러한 형식은 고품질 비디오를 더 느린 비트율로 스트리밍합니다. 개발자는 Adobe Premiere Pro 및 Adobe After Effects를 비롯한 업계 표준 도구를 사용하여 뛰어난 비디오 내용을 만들고 전송할 수 있습니다.

유형	형식	컨테이너
비디오	H.264	MPEG-4: MP4, M4V, F4V, 3GPP
비디오	Sorenson Spark	FLV 파일
비디오	ON2 VP6	FLV 파일
오디오	AAC+ / HE-AAC / AAC v1 / AAC v2	MPEG-4:MP4, M4V, F4V, 3GPP
오디오	Mp3	Mp3
오디오	Nellymoser	FLV 파일
오디오	Speex	FLV 파일

기타 도움말 항목

Flash Media Server: 지원되는 코덱

Adobe HTTP 동적 스트리밍

휴대 장치용 비디오 인코딩

Android에서 AIR는 광범위한 범위의 H.264 비디오를 디코딩할 수 있습니다. 그러나 H.264 비디오 중 일부만 휴대폰에서 부드럽게 재생될 수 있는데, 이는 많은 휴대 전화의 처리 전력에 제약이 있기 때문입니다. 휴대 장치용 Adobe Flash Player는 내장 하드웨어 가속을 통해 H.264 비디오를 디코딩할 수 있습니다. 이러한 디코딩을 사용하면 적은 전력을 소비하고도 더 나은 품질을 얻을 수 있습니다.

H.264 표준은 몇 가지 인코딩 기술을 지원합니다. 고사양 장치만이 프로파일 및 레벨이 복잡한 비디오를 부드럽게 재생할 수 있지만, 대부분의 장치는 기본 프로파일로 인코딩된 비디오를 재생할 수 있습니다. 휴대 장치에서 하드웨어 가속은 이러한 기술의 하위 집합에 사용할 수 있습니다. 프로파일 및 레벨 매개 변수는 인코딩 기술의 하위 집합과 인코더에 사용되는 설정을 정의합니다. 개발자의 입장에서 이는 곧 대부분의 장치에서 훌륭히 재생되는 해상도를 선택하여 비디오를 인코딩한다는 것을 의미합니다.

하드웨어 가속의 이점을 얻을 수 있는 해상도는 장치에 따라 다르지만 대부분의 장치는 다음의 표준 해상도를 지원합니다.

종횡비	권장 해상도		
4:3	640×480	512×384	480×360
16:9	640×360	512×288	480×272

참고: Flash Player는 H.264 표준의 모든 레벨 및 프로파일을 지원합니다. 이 권장 사항을 따르면 대부분의 장치에서 하드웨어 가속을 활용할 수 있고 더 나은 성능을 얻을 수 있습니다. 이 권장 사항은 의무 사항이 아닙니다.

Adobe Media Encoder CS5에 대한 자세한 내용 및 인코딩 설정은 휴대 장치에서 Flash Player 10.1용 H.264 비디오 인코딩 시 권장 사항을 참조하십시오.

참고: iOS에서는 Sorenson Spark 및 On2 VP6 코덱으로 인코딩된 비디오만 Video 클래스를 사용하여 재생할 수 있습니다. 장치 비디오 플레이어에서 H.264 인코딩 비디오를 재생하려면 flash.net.navigateToURL() 함수를 사용하여 비디오 URL을 실행하면 됩니다. StageWebView 객체에 표시된 HTML 페이지에 <video> 태그를 사용하여 H.264 비디오를 재생할 수도 있습니다.

Flash Player 및 AIR와 인코딩된 비디오 파일의 호환성

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 7에서는 Sorenson™ Spark™ 비디오 코덱을 사용하여 인코딩된 FLV 파일을 지원합니다. 또한 Flash Player 8에서는 Flash Professional 8에서 Sorenson Spark 또는 On2 VP6 인코더를 사용하여 인코딩된 FLV 파일을 지원합니다. On2 VP6 비디오 코덱은 알파 채널을 지원합니다.

Flash Player 9.0.115.0 이상 버전에서는 표준 MPEG-4 컨테이너 형식에서 파생된 파일을 지원합니다. 이러한 파일에는 F4V, MP4, M4A, MOV, MP4V, 3GP 및 3G2가 포함됩니다(H.264 비디오, HE-AAC v2 인코딩 오디오 또는 둘 모두가 포함된 경우). H.264는 Sorenson 또는 On2의 동일한 인코딩 프로파일에 비해 더 느린 비트율로 더 좋은 품질의 비디오를 제공합니다. HE-AAC v2는 MPEG-4 비디오 표준에 정의된 표준 오디오 형식인 AAC의 확장입니다. HE-AAC v2는 SBR(Spectral Band Replication) 및 PS(Parametric Stereo) 기술을 사용하여 낮은 비트율로 코딩 효율성을 향상시킵니다.

다음 표에서는 지원되는 코덱을 나열하고 해당 SWF 파일 형식과 이를 재생하는 데 필요한 Flash Player 및 AIR 버전을 보여 줍니다.

코덱	SWF 파일 형식 버전(지원되는 최소 제작 버전)	Flash Player 및 AIR(재생에 필요한 최소 버전)
Sorenson Spark	6	Flash Player 6, Flash Lite 3
On2 VP6	6	Flash Player 8, Flash Lite 3 Flash Player 8 이상의 버전에서만 On2 VP6 비디오의 제작 및 재생을 지원합니다.
H.264(MPEG-4 Part 10)	9	Flash Player 9 업데이트 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 업데이트 3, AIR 1.0
Speex(오디오)	10	Flash Player 10, AIR 1.5
Nellymoser	6	Flash Player 6

Adobe F4V 및 FLV 비디오 파일 형식 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe에서는 내용을 Flash Player 및 AIR로 스트리밍하기 위한 F4V 및 FLV 비디오 파일 형식을 제공합니다. 이러한 비디오 파일 형식에 대한 자세한 내용은 www.adobe.com/go/video_file_format_kr를 참조하십시오.

F4V 비디오 파일 형식

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 업데이트 3(9.0.115.0)과 AIR 1.0부터는 ISO MP4 형식을 기반으로 하는 Adobe F4V 비디오 형식을 지원합니다. 해당 형식의 하위 집합은 다른 기능을 지원합니다. Flash Player에서 올바른 F4V 파일은 다음 최상위 상자 중 하나로 시작해야 합니다.

- ftyp

  ftyp 상자는 특정 파일 형식을 재생하기 위해 프로그램에서 지원해야 하는 기능을 식별합니다.

- moov

  moov 상자는 실제로는 F4V 파일의 헤더입니다. 이 상자에는 다른 상자가 하나 이상 들어 있으며 이 상자에는 또한 F4V 데이터의 구조를 정의하는 다른 상자가 들어 있습니다. F4V 파일에는 moov 상자가 하나만 들어 있어야 합니다.

- mdat

  mdat 상자에는 F4V 파일의 데이터 페이로드가 들어 있습니다. FV 파일에는 mdat 상자가 하나만 들어 있습니다. mdat 상자는 그 자체로는 인식되지 않기 때문에 파일에 moov 상자도 있어야 합니다.

F4V 파일은 최상위 바이트가 가장 낮은 주소에서 먼저 나타나는 big-endian 바이트 순서의 멀티바이트 정수를 지원합니다.

FLV 비디오 파일 형식

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe FLV 파일 형식에는 전달을 위해 Flash Player로 인코딩된 오디오 및 비디오 데이터가 포함됩니다. Adobe Media Encoder 또는 Sorenson™ Squeeze와 같은 인코더를 사용하여 QuickTime 또는 Windows Media 비디오 파일을 FLV 파일로 변환할 수 있습니다.

참고: 비디오 파일을 Flash로 가져온 다음 이를 FLV 파일로 내보내는 방식으로 FLV 파일을 만들 수 있습니다. 지원되는 비디오 편집 응용 프로그램에서는 FLV 내보내기 플러그인을 사용하여 FLV 파일을 내보낼 수 있습니다. 웹 서버에서 FLV 파일을 로드하려면 웹 서버에 해당 파일 확장명과 MIME 유형을 등록합니다. 자세한 내용은 웹 서버 설명서를 참조하십시오. FLV 파일의 MIME 유형은 video/x-flv입니다. 자세한 내용은 457페이지의 “서버에 호스트할 수 있도록 FLV 파일 구성”을 참조하십시오.

FLV 파일에 대한 자세한 내용은 457페이지의 “비디오 파일의 고급 항목”을 참조하십시오.

외부 비디오와 포함된 비디오

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 비디오 파일을 사용하면 가져온 비디오를 사용할 때 지원되지 않는 다음과 같은 기능을 사용할 수 있습니다.

- 재생 속도를 느리게 하지 않고도 응용 프로그램에서 긴 비디오 클립을 사용할 수 있습니다. 외부 비디오 파일은 캐시된 메모리를 사용합니다. 즉, 큰 파일은 여러 개의 작은 부분으로 나뉘어 저장되고 동적으로 액세스됩니다. 따라서 외부 F4V 및 FLV 파일에는 포함된 비디오 파일보다 적은 메모리가 필요합니다.
- 외부 비디오 파일은 자신이 재생되는 SWF 파일과 프레임 속도가 다릅니다. 예를 들어 SWF 파일의 프레임 속도를 30fps로 설정하고 비디오 프레임 속도를 21fps로 설정해 보십시오. 이 설정을 사용하면 포함된 비디오의 경우보다 비디오를 더욱 세부

적으로 제어할 수 있으므로 비디오를 유연하게 재생할 수 있습니다. 또한 기존 SWF 파일 내용을 변경하지 않고 다른 프레임 속도로 비디오 파일을 재생할 수 있습니다.

- 외부 비디오 파일을 사용할 경우 비디오 파일을 로드하는 동안 SWF 내용의 재생을 중단할 필요가 없습니다. 가져온 비디오 파일은 CD-ROM 드라이브에 액세스하는 등 문서 재생을 방해하며 특정 기능을 실행하는 경우가 가끔 발생할 수 있습니다. 비디오 파일은 SWF 내용과는 별도로 작동되므로 SWF 내용의 재생을 중단하지 않습니다.
- 외부 FLV 파일의 경우 이벤트 핸들러를 사용하여 비디오의 메타데이터에 액세스할 수 있으므로 비디오 내용에 캡션을 쉽게 추가할 수 있습니다.

Video 클래스 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

Video 클래스를 사용하면 SWF 파일에 포함하지 않고도 응용 프로그램에서 라이브 스트리밍 비디오를 표시할 수 있습니다. Camera.getCamera() 메서드를 사용하여 라이브 비디오를 캡처하고 재생할 수 있습니다. Video 클래스를 사용하여 HTTP를 통하거나 로컬 파일 시스템으로부터 비디오 파일을 재생할 수도 있습니다. 프로젝트에서 Video 클래스를 사용하는 방법에는 여러 가지가 있습니다.

- NetConnection 및 NetStream 클래스를 사용하여 동적으로 비디오 파일을 로드하고 비디오를 Video 객체에 표시합니다.
- 사용자의 카메라에서 입력을 캡처합니다. 자세한 내용은 472페이지의 "카메라를 사용한 작업"을 참조하십시오.
- FLVPlayback 구성 요소를 사용합니다.
- VideoDisplay 컨트롤을 사용합니다.

참고: 스테이지에 있는 Video 객체의 인스턴스는 Video 클래스의 인스턴스입니다.

Video 클래스는 flash.media 패키지에 포함되어 있지만 flash.display.DisplayObject 클래스에서 상속됩니다. 따라서 행렬 변환 및 필터와 같은 모든 표시 객체 기능은 Video 인스턴스에도 적용됩니다.

자세한 내용은 156페이지의 "표시 객체 조작", 189페이지의 "기하 도형을 사용한 작업" 및 242페이지의 "표시 객체 필터링"을 참조하십시오.

비디오 파일 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

NetStream 및 NetConnection 클래스를 사용한 비디오 로드는 여러 단계로 구성된 프로세스입니다. 최상의 방법으로, Video 객체를 표시 목록에 추가하는 단계, NetStream 객체를 Video 인스턴스에 연결하는 단계, NetStream 객체의 play() 메서드를 호출하는 단계를 지정된 순서대로 수행해야 합니다.

1 NetConnection 객체를 만듭니다. 로컬 비디오 파일이나 Adobe Flash Media Server 2 등의 서버를 사용하지 않는 비디오 파일에 연결할 경우 connect() 메서드에 null을 전달하여 HTTP 주소나 로컬 드라이브에서 비디오 파일을 재생합니다. 서버와 연결 중이라면, 매개 변수를 서버에서 비디오 파일을 포함하고 있는 응용 프로그램의 URI로 설정합니다.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
```

2 다음 코드 조각과 같이 비디오를 표시하는 새 Video 객체를 만들어 스테이지 표시 목록에 추가합니다.

```
var vid:Video = new Video();
addChild(vid);
```

3 NetConnection 객체를 생성자에 인수로 전달하여 NetStream 객체를 만듭니다. 다음 코드 조각은 NetStream 객체를 NetConnection 인스턴스에 연결하고 스트림의 이벤트 핸들러를 설정합니다.

```
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 다음 코드 조각과 같이 Video 객체의 attachNetStream() 메서드를 사용하여 NetStream 객체를 Video 객체에 연결합니다.

```
vid.attachNetStream(ns);
```

5 비디오 파일 URL을 인수로 사용하는 NetStream 객체의 play() 메서드를 호출하여 비디오 재생을 시작합니다. 다음 코드 조각은 SWF 파일과 동일한 디렉토리에 있는 "video.mp4"라는 비디오 파일을 로드하고 재생합니다.

```
ns.play("video.mp4");
```

기타 도움말 항목

Flex: Spark VideoPlayer 컨트롤

spark.components.VideoDisplay

비디오 재생 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

NetStream 클래스는 비디오 재생 제어를 위한 네 가지 주요 메서드를 제공합니다.

pause(): 비디오 스트림 재생을 일시 중지합니다. 비디오가 이미 일시 정지된 경우 이 메서드를 호출해도 아무 것도 수행되지 않습니다.

resume(): 일시 중지된 비디오 스트림의 재생을 다시 시작합니다. 비디오가 이미 재생 중인 경우 이 메서드를 호출하면 아무 작업도 수행되지 않습니다.

seek(): 지정된 위치(스트림 시작 부분부터 초 단위의 오프셋)에 가장 가까운 키프레임을 찾습니다.

togglePause(): 스트림 재생을 일시 중지 또는 다시 시작합니다.

참고: stop() 메서드가 없습니다. 스트림을 멈추려면 재생을 일시 정지한 후 비디오 스트림의 시작 부분을 찾아야 합니다.

참고: play() 메서드는 재생을 다시 시작하지 않으며 비디오 파일 로드에 사용됩니다.

다음 예제에서는 여러 버튼을 사용하여 비디오를 제어하는 방법을 설명합니다. 다음 예제를 실행하려면 새 문서를 만들고 작업 영역에 버튼 인스턴스를 네 개(pauseBtn, playBtn, stopBtn, togglePauseBtn) 추가합니다.

```
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();
}
```

비디오가 재생되는 동안 pauseBtn 버튼 인스턴스를 클릭하면 비디오 파일이 일시 정지됩니다. 비디오가 이미 일시 정지된 경우에는 이 버튼을 클릭해도 아무 효과가 없습니다. 재생이 이미 일시 정지되어 있는 경우 playBtn 버튼 인스턴스를 클릭하면 비디오 재생이 다시 시작되지만, 비디오가 이미 재생 중인 경우에는 이 버튼을 클릭해도 아무 효과가 없습니다.

비디오 스트림의 끝 감지

Flash Player 9 이상, Adobe AIR 1.0 이상

비디오 스트림의 시작과 끝을 수신하려면 이벤트 리스너를 NetStream 인스턴스에 추가하여 netStatus 이벤트를 수신해야 합니다. 다음 코드에서는 비디오 재생 중 다양한 코드를 수신하는 방법을 보여 줍니다.

```
ns.addEventListener(NetStatusEvent.NET_STATUS, statusHandler);
function statusHandler(event:NetStatusEvent):void
{
    trace(event.info.code)
}
```

이전 코드는 다음을 출력합니다.

```
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
```

특히 수신할 두 코드는 비디오 재생의 시작과 끝을 알리는 "NetStream.Play.Start"와 "NetStream.Play.Stop"입니다. 다음 코드에서는 switch 문을 사용하여 이러한 두 코드를 필터링하고 메시지를 추적합니다.

```
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;
    }
}
```

사용자는 netStatus 이벤트(NetStatusEvent.NET_STATUS)를 수신하여 현재 비디오 재생 완료 시 재생 목록에 있는 다음 비디오를 로드하는 비디오 플레이어를 만들 수 있습니다.

전체 화면 모드로 비디오 재생

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 AIR를 사용하여 비디오 재생을 위한 전체 화면 응용 프로그램을 만들 수 있고 비디오를 전체 화면 크기로 조절할 수 있습니다.

데스크톱에서 전체 화면 모드로 실행 중인 AIR 내용의 경우 재생 중인 동안 비디오 입력을 중단하거나 전체 화면 모드를 종료할 때까지 시스템 화면 보호기 및 절전 옵션이 비활성화됩니다.

전체 화면 모드에 대한 자세한 내용은 151페이지의 “전체 화면 모드 작업”을 참조하십시오.

브라우저에서 Flash Player의 전체 화면 모드 활성화

브라우저에서 Flash Player에 대한 전체 화면 모드를 구현하려면 먼저 응용 프로그램의 제작 템플릿을 통해 전체 화면 모드를 사용하도록 설정해야 합니다. 전체 화면을 허용하는 템플릿에는 allowFullScreen 매개 변수가 들어 있는 <object> 및 <embed> 태그가 포함되어 있습니다. 다음 예제에서는 <embed> 태그 내의 allowFullScreen 매개 변수를 보여 줍니다.

```
<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>
```

Flash에서 [파일] > [제작 설정]을 선택하고 [제작 설정] 대화 상자의 [HTML] 탭에서 [Flash 전용 - 전체 화면 가능] 템플릿을 선택합니다.

Flex에서는 HTML 템플릿에 전체 화면을 지원하는 <object> 및 <embed> 태그가 들어 있는지 확인합니다.

전체 화면 모드 시작

브라우저에서 실행되는 Flash Player 내용의 경우 마우스 클릭 또는 키 누르기에 대한 응답으로 비디오에 대한 전체 화면 모드를 시작합니다. 예를 들어 사용자가 [전체 화면] 버튼을 클릭하거나 컨텍스트 메뉴에서 [전체 화면] 명령을 선택할 때 전체 화면 모드를 시작할 수 있습니다. 사용자에게 응답하려면 액션이 발생한 객체에 이벤트 리스너를 추가합니다. 다음 코드에서는 전체 화면 모드를 시작하기 위해 사용자가 클릭하는 버튼에 이벤트 리스너를 추가합니다.

```
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;

}
```

이 코드에서는 Stage.displayState 속성을 StageDisplayState.FULL_SCREEN으로 설정하여 전체 화면 모드를 시작합니다. 이 코드에서는 스테이지의 비디오 표시 공간에 비례한 비디오 크기 조절을 통해 전체 스테이지를 전체 화면으로 확대합니다.

fullScreenSourceRect 속성을 사용하면 스테이지의 특정 영역을 지정하여 전체 화면으로 확대할 수 있습니다. 먼저 전체 화면으로 확대할 사각형을 정의합니다. 그런 다음 이 사각형을 Stage.fullScreenSourceRect 속성에 할당합니다. 이 버전의 fullScreenButtonHandler() 함수는 비디오만을 전체 화면으로 확대하는 두 행의 코드를 추가합니다.

```
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;
}
```

이 예제에서는 마우스 클릭에 대한 응답으로 이벤트 핸들러를 호출하지만 전체 화면 모드로 전환하는 방법은 Flash Player와 AIR 모두에서 동일합니다. 크기를 조절할 사각형을 정의하고 Stage.displayState 속성을 설정합니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

다음에 나오는 전체 예제에서는 연결과 비디오에 대한 NetStream 객체를 만들고 비디오 재생을 시작하는 코드를 추가합니다.

```
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
        }
    }
}
```

onMetaData() 함수가 있는 경우 이 함수는 비디오 메타데이터를 처리하기 위한 콜백 함수입니다. 콜백 함수는 런타임에 특정 유형의 사건 또는 이벤트에 대한 응답으로 호출되는 함수입니다. 이 예제에서 onMetaData()함수는 함수를 제공하는 데 필요한 요구 사항을 충족시키는 스텁입니다. 자세한 내용은 440페이지의 “메타데이터 및 큐 포인트에 대한 콜백 메서드 작성”을 참조하십시오.

전체 화면 모드 종료

Esc 키와 같은 키보드 단축키 중 하나를 눌러 전체 화면 모드를 종료할 수 있습니다. ActionScript에서는 Stage.displayState 속성을 StageDisplayState.NORMAL로 설정하여 전체 화면 모드를 종료할 수 있습니다. 다음 예제의 코드에서는 NetStream.Play.Stop netStatus 이벤트가 발생할 때 전체 화면 모드를 종료합니다.

```
videoStream.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);

private function netStatusHandler(event:NetStatusEvent)
{
    if(event.info.code == "NetStream.Play.Stop")
        stage.displayState = StageDisplayState.NORMAL;
}
```

전체 화면 하드웨어 가속

스테이지의 사각형 영역을 전체 화면 모드로 확대하는 경우 Flash Player 또는 AIR에서는 활성화된 하드웨어 가속을 사용합니다. 런타임에서는 컴퓨터의 비디오 어댑터를 사용하여 비디오 또는 스테이지의 일부분을 전체 화면 크기로 빠르게 확대할 수 있습니다. 이러한 상황에서 Flash Player 응용 프로그램이 Video 클래스(또는 Camera 클래스: Flash Player 11.4/AIR 3.4 이상)에서 StageVideo 클래스로 전환하면 도움이 되는 경우가 많습니다.

전체 화면 모드에서의 하드웨어 가속에 대한 자세한 내용은 151페이지의 “전체 화면 모드 작업”을 참조하십시오. StageVideo에 대한 자세한 내용은 463페이지의 “하드웨어 가속 프레젠테이션에 StageVideo 클래스 사용”을 참조하십시오.

비디오 파일 스트리밍

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Media Server로부터 파일을 스트리밍하려는 경우 NetConnection 및 NetStream 클래스를 사용하여 원격 서버 인스턴스에 연결하고 지정된 스트림을 재생할 수 있습니다. RTMP(Real-Time Messaging Protocol) 서버를 지정하려면 NetConnection.connect() 메서드에 null이 아니라 원하는 RTMP URL(예: "rtmp://localhost/appName/appInstance")을 전달합니다. 지정된 Flash Media Server에서 특정 라이브 또는 녹화된 스트림을 재생하려면 NetStream.publish()에 의해 제작된 라이브 데이터를 식별하는 이름을, 녹화된 스트림을 재생하려면 재생 항목에 대해 기록된 파일 이름을 NetStream.play() 메서드에 전달합니다.

서버에 비디오 보내기

Flash Player 9 이상, Adobe AIR 1.0 이상

Video 또는 Camera 객체가 포함된 보다 복잡한 응용 프로그램을 제작하려는 사용자를 위해 Flash Media Server는 미디어 스트리밍 기능과 미디어 응용 프로그램을 제작하여 더 많은 사용자에게 제공할 수 있는 개발 환경을 모두 제공합니다. 이를 통해 개발자는 VOD(주문형 비디오), 라이브 웹이벤트 방송, Mp3 스트리밍, 비디오 블로그, 비디오 메시징, 멀티미디어 채팅 환경 등의 응용 프로그램을 제작할 수 있습니다. 자세한 내용은 www.adobe.com/go/learn_fms_docs_kr에서 온라인으로 제공되는 Flash Media Server 설명서를 참조하십시오.

큐 포인트 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

인코딩 중 Adobe F4V 또는 FLV 비디오 파일에 큐 포인트를 포함할 수 있습니다. 본래 큐 포인트는 영화에서 영사 기사에게 필름 릴이 거의 끝 부분에 도달했음을 나타내는 시각적 신호를 제공하기 위해 포함되었습니다. Adobe F4V 및 FLV 비디오 형식에서 큐 포인트를 사용하면 비디오 스트림에서 큐 포인트가 발생할 때 응용 프로그램에서 하나 이상의 다른 액션을 트리거할 수 있습니다.

Flash 비디오에서는 다양한 큐 포인트를 사용할 수 있습니다. ActionScript를 사용하여 비디오 파일을 만들 때 비디오 파일에 포함하는 큐 포인트와 상호 작용할 수 있습니다.

- 내비게이션 큐 포인트: 비디오 파일을 인코딩할 때 비디오 스트림과 메타데이터 패킷에 내비게이션 큐 포인트를 포함합니다. 내비게이션 큐 포인트를 사용하면 사용자가 파일의 지정된 부분을 검색할 수 있습니다.
- 이벤트 큐 포인트: 비디오 파일을 인코딩할 때 비디오 스트림과 메타데이터 패킷에 이벤트 큐 포인트를 포함합니다. 비디오 재생 중에 지정된 지점에서 트리거되는 이벤트를 처리하기 위한 코드를 작성할 수 있습니다.
- ActionScript 큐 포인트: ActionScript 큐 포인트는 Flash FLVPlayback 구성 요소에만 사용할 수 있습니다. ActionScript 큐 포인트는 ActionScript 코드를 사용하여 만들고 액세스하는 외부 큐 포인트입니다. 비디오의 재생과 관련하여 이러한 큐 포인트를 트리거하는 코드를 작성할 수 있습니다. 이러한 큐 포인트는 비디오 플레이어에서 개별적으로 추적하므로 포함된 큐 포인트보다 덜 정확합니다(최대 0.1초까지). 사용자에게 큐 포인트를 탐색할 수 있게 할 응용 프로그램을 작성하려는 경우 ActionScript 큐 포인트를 사용하는 대신에 파일을 인코딩할 때 큐 포인트를 만들거나 포함시킬 수 있습니다. 큐 포인트가 사용하는 데 보다 정확하므로 FLV 파일에 큐 포인트를 포함해야 합니다.

내비게이션 큐 포인트는 지정된 큐 포인트 위치에 키프레임을 만들므로 코드를 사용하여 비디오 플레이어의 재생 헤드를 해당 위치로 이동할 수 있습니다. 사용자가 찾을 수 있도록 특정 지점을 비디오 파일 안에 설정할 수 있습니다. 예를 들어 비디오에는 여러 장이나 세그먼트가 있을 수 있으므로 비디오 파일 안에 내비게이션 큐 포인트를 포함시켜 비디오를 제어할 수 있습니다.

큐 포인트를 사용한 Adobe 비디오 파일 인코딩에 대한 자세한 내용은 **Flash 사용**의 "큐 포인트 포함"을 참조하십시오.

ActionScript를 작성하여 큐 포인트 매개 변수에 액세스할 수 있습니다. 큐 포인트 매개 변수는 콜백 핸들러에서 수신된 이벤트 객체의 일부입니다.

FLV 파일이 특정 큐 포인트에 도달했을 때 코드에서 특정 액션을 트리거하려면 NetStream.onCuePoint 이벤트 핸들러를 사용합니다.

F4V 비디오 파일의 큐 포인트에 대한 액션을 동기화하려면 onMetaData() 또는 onXMPData() 콜백 함수에서 큐 포인트 데이터를 검색하고 ActionScript 3.0의 Timer 클래스를 사용하여 큐 포인트를 트리거해야 합니다. F4V 큐 포인트에 대한 자세한 내용은 451페이지의 “onXMPData() 사용”을 참조하십시오.

큐 포인트 및 메타데이터 처리에 대한 자세한 내용은 440페이지의 “메타데이터 및 큐 포인트에 대한 콜백 메서드 작성”을 참조하십시오.

메타데이터 및 큐 포인트에 대한 콜백 메서드 작성

Flash Player 9 이상, Adobe AIR 1.0 이상

플레이어에서 특정 메타데이터를 받거나 특정 큐 포인트에 도달할 때 응용 프로그램에서 액션을 트리거할 수 있습니다. 이러한 이벤트가 발생하면 특정 콜백 메서드를 이벤트 핸들러로 사용해야 합니다. NetStream 클래스는 재생 중에 발생할 수 있는 onCuePoint(FLV 파일만 해당), onImageData, onMetaData, onPlayStatus, onTextData 및 onXMPData 메타데이터 이벤트를 지정합니다.

이러한 핸들러에 대한 콜백 메서드를 작성해야 합니다. 그렇지 않으면 Flash 런타임에서 오류가 발생할 수 있습니다. 예를 들어 다음 코드에서는 SWF 파일과 같은 폴더에서 video.flv라는 FLV 파일을 재생합니다.

```
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);
```

이전 코드에서는 video.flv라는 로컬 비디오 파일을 로드하며, 전달할 asyncError(AsyncErrorEvent.ASYNC_ERROR)를 수신합니다. 기본 비동기 코드에서 예외가 발생하면 이 이벤트가 전달됩니다. 이 경우 비디오 파일에 메타데이터나 큐 포인트 정보가 있으면 이벤트가 전달되며 해당 리스너는 정의되어 있지 않습니다. 이전 코드에서는 asyncError 이벤트를 처리하며, 사용자가 비디오 파일의 메타데이터나 큐 포인트 정보에 관심이 없는 경우 오류를 무시합니다. 메타데이터와 여러 큐 포인트가 포함된 FLV가 있는 경우 trace() 함수는 다음 오류 메시지를 표시합니다.

```
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.
```

오류는 NetStream 객체가 onMetaData 또는 onCuePoint 콜백 메서드를 찾지 못해서 발생합니다. 다음과 같은 여러 가지 방법으로 응용 프로그램 내에서 이러한 콜백 메서드를 정의할 수 있습니다.

기타 도움말 항목

Flash Media Server: 스트림에서 메타데이터 처리

NetStream 객체의 client 속성을 Object로 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

client 속성을 Object 또는 NetStream 하위 클래스로 설정하여 onMetaData 및 onCuePoint 콜백 메서드를 다시 라우팅하거나 완전히 무시할 수 있습니다. 다음 예제에서는 asyncError 이벤트를 수신하지 않고 빈 Object를 사용하여 콜백 메서드를 무시하는 방법을 보여 줍니다.

```
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);
```

onMetaData 또는 onCuePoint 콜백 메서드 중 하나를 수신하려는 경우에는 다음 코드와 같이 해당 콜백 메서드를 처리할 메서드를 정의해야 합니다.

```
var customClient:Object = new Object();
customClient.onMetaData = metaDataHandler;
function metaDataHandler(infoObject:Object):void
{
    trace("metadata");
}
```

이전 코드에서는 onMetaData 콜백 메서드를 수신하며 문자열을 추적하는 metaDataHandler() 메서드를 호출합니다. Flash 런타임에서 큐 포인트를 발견하는 경우에는 onCuePoint 콜백 메서드가 정의되지 않았더라도 오류가 생성되지 않습니다.

사용자 정의 클래스 만들기 및 콜백 메서드를 처리할 메서드 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 코드에서는 NetStream 객체의 client 속성을 콜백 메서드에 대한 핸들러를 정의하는 사용자 정의 클래스인 CustomClient로 설정합니다.

```
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);
```

CustomClient 클래스의 주요 기능은 다음과 같습니다.

```
package
{
    public class CustomClient
    {
        public function onMetaData(infoObject:Object):void
        {
            trace("metadata");
        }
    }
}
```

CustomClient 클래스는 onMetaData 콜백 핸들러에 대한 핸들러를 정의합니다. 큐 포인트가 발견되고 onCuePoint 콜백 핸들러가 호출되면 "flash.net.NetStream은(는) 콜백 onCuePoint을(를) 호출할 수 없습니다."라는 asyncError 이벤트(AsyncErrorEvent.ASYNC_ERROR)가 전달됩니다. 이러한 오류를 방지하려면 CustomClient 클래스에 onCuePoint 콜백 메서드를 정의하거나 asyncError 이벤트에 대한 이벤트 핸들러를 정의해야 합니다.

NetStream 클래스 확장 및 콜백 메서드를 처리할 메서드 추가

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 코드에서는 CustomNetStream 클래스의 인스턴스를 만듭니다(이 클래스는 이후 코드 샘플에서 정의됨).

```
var ns:CustomNetStream = new CustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

다음 코드 샘플에서는 NetStream 클래스를 확장하고 필수 NetConnection 객체 생성을 처리하며 onMetaData 및 onCuePoint 콜백 핸들러 메서드를 처리하는 CustomNetStream 클래스를 정의합니다.

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

CustomNetStream 클래스의 onMetaData() 및 onCuePoint() 메서드 이름을 변경하려면 다음 코드를 사용합니다.

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

NetStream 클래스 확장 및 동적 클래스로 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

NetStream 클래스를 확장하고 그 하위 클래스를 동적으로 만들어 onCuePoint 및 onMetaData 콜백 핸들러가 동적으로 추가될 수 있도록 할 수 있습니다. 다음 샘플에서는 이 방법을 설명합니다.

```
var ns:DynamicCustomNetStream = new DynamicCustomNetStream();
ns.play("video.flv");

var vid:Video = new Video();
vid.attachNetStream(ns);
addChild(vid);
```

DynamicCustomNetStream 클래스의 주요 기능은 다음과 같습니다.

```
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);
        }
    }
}
```

onMetaData 및 onCuePoint 콜백 핸들러에 대한 핸들러가 없어도 DynamicCustomNetStream 클래스가 동적이므로 오류가 발생하지 않습니다. onMetaData 및 onCuePoint 콜백 핸들러에 대한 메서드를 정의하려면 다음 코드를 사용합니다.

```
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");
}
```

NetStream 객체의 client 속성을 this로 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

client 속성을 this로 설정하면 응용 프로그램은 현재 범위에서 onMetaData() 및 onCuePoint() 메서드를 찾습니다. 다음 예제에서 이를 확인할 수 있습니다.

```
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);
```

onMetaData 또는 onCuePoint 콜백 핸들러가 호출된 상태에서 콜백을 처리할 메서드가 없는 경우 오류가 발생하지 않습니다. 이러한 콜백 핸들러를 처리하려면 다음 코드와 같이 코드에 onMetaData() 및 onCuePoint() 메서드를 생성합니다.

```
function onMetaData(infoObject:Object):void
{
    trace("metadata");
}
function onCuePoint(infoObject:Object):void
{
    trace("cue point");
}
```

큐 포인트 및 메타데이터 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

비디오를 재생할 때 NetStream 콜백 메서드를 사용하여 큐 포인트 및 메타데이터 이벤트를 캡처하고 처리할 수 있습니다.

큐 포인트 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 표에서는 Flash Player 및 AIR에서 F4V 및 FLV 큐 포인트를 캡처하는 데 사용할 수 있는 콜백 메서드를 설명합니다.

런타임	F4V	FLV
Flash Player 9/AIR1.0		OnCuePoint
		OnMetaData
Flash Player 10		OnCuePoint
	OnMetaData	OnMetaData
	OnXMPData	OnXMPData

다음 예제에서는 간단한 for..in 루프를 사용하여 onCuePoint() 함수가 수신하는 infoObject 매개 변수의 각 속성을 반복합니다. 이 예제에서는 큐 포인트 데이터를 수신할 때 trace() 함수를 호출하여 메시지를 표시합니다.

```
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]);
    }
}
```

다음이 출력됩니다.

```
parameters:
name: point1
time: 0.418
type: navigation
```

이 코드에서는 여러 방법 중 한 가지를 사용하여 콜백 메서드가 실행되는 객체를 설정했지만 다른 방법을 사용할 수도 있습니다. 자세한 내용은 440페이지의 “메타데이터 및 큐 포인트에 대한 콜백 메서드 작성”을 참조하십시오.

비디오 메타데이터 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

OnMetaData() 및 OnXMPData() 함수를 사용하여 큐 포인트가 들어 있는 비디오 파일의 메타데이터 정보에 액세스할 수 있습니다.

OnMetaData() 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

메타데이터에는 지속 기간, 폭, 높이 및 프레임 속도와 같은 비디오 파일에 대한 정보가 포함됩니다. 비디오 파일에 추가되는 메타데이터 정보는 비디오 파일을 인코딩하는 데 사용하는 소프트웨어에 따라 다릅니다.

```
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]);
    }
}
```

이전 코드에서는 다음을 출력합니다.

```
width: 320
audiodelay: 0.038
canSeekToEnd: true
height: 213
cuePoints: ,,
audiodatarate: 96
duration: 16.334
videodatarate: 400
framerate: 15
videocodecid: 4
audiocodecid: 2
```

비디오에 오디오가 포함되지 않은 경우에는 인코딩 중에 메타데이터에 추가되는 오디오 정보가 없으므로 오디오 관련 메타데이터 정보(예: audiodatarate)가 undefined를 반환합니다.

위의 코드에서는 큐 포인트 정보가 표시되지 않았습니다. 큐 포인트 메타데이터를 표시하려면 Object에서 항목을 반복적으로 표시하는 다음과 같은 함수를 사용합니다.

```
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);
        }
    }
}
```

앞의 코드 예제를 사용하여 onMetaData() 메서드의 infoObject 매개 변수를 추적하면 다음이 출력됩니다.

```
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
```

다음 예제에서는 MP4 비디오에 대한 메타데이터를 표시합니다. 이 예제에서는 메타데이터를 쓰는 대상으로 metaDataOut이라는 TextArea 객체가 있다고 가정합니다.

```
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;
        }
    }
}
```

onMetaData() 함수는 이 비디오에 대해 다음과 같은 출력을 생성합니다.

```
moovposition=731965
height=352
avclevel=21
videocodecid=avc1
duration=2.36
width=704
videoframerate=25
avcprofile=88
trackinfo=[object Object]
```

정보 객체 사용

다음 표에서는 수신하는 Object의 onMetaData() 콜백 함수에 전달되는 비디오 메타데이터로 가능한 값을 보여 줍니다.

매개 변수	설명
aacaot	AAC 오디오 객체 유형으로, 0, 1 또는 2가 지원됩니다.
avclevel	10, 11, 20, 21 등의 AVC IDC 레벨 번호입니다.
avcprofile	55, 77, 100 등의 AVC 프로파일 번호입니다.
audiocodecid	사용된 오디오 코덱(코딩/디코딩 기술)을 나타내는 문자열(예: ".Mp3", "mp4a")입니다.
audiodatarate	오디오가 인코딩된 속도(초당 킬로바이트)를 나타내는 숫자입니다.
audiodelay	FLV 파일에서 원본 FLV 파일의 "time 0"이 있는 시간을 나타내는 숫자입니다. 비디오 내용은 오디오를 정확하게 동기화할 약간의 시간 동안 지연되어야만 합니다.
canSeekToEnd	부울 값으로서, FLV 파일이 점진적 다운로드 비디오 파일의 끝까지 검색할 수 있도록 마지막 프레임의 키프레임으로 인코딩된 경우 true입니다. FLV 파일이 마지막 프레임의 키프레임으로 인코딩되지 않은 경우에는 false입니다.
cuePoints	객체 배열이며, 배열의 각 요소는 FLV 파일에 포함된 각 큐 포인트에 해당합니다. FLV 파일에 아무런 큐 포인트도 없으면 값은 정의되지 않습니다. 각 객체에는 다음과 같은 속성이 있습니다. • type: 큐 포인트의 유형을 "navigation" 또는 "event" 중 하나로 지정하는 문자열입니다. • name: 큐 포인트의 이름을 나타내는 문자열입니다. • time: 큐 포인트의 시간을 소수 세 자리(밀리초)까지 정밀하게 초 단위로 나타내는 숫자입니다. • parameters: 큐 포인트를 만들 때 사용자가 지정한 이름/값 쌍을 포함하는 객체이며, 선택 사항입니다.
duration	비디오 파일의 지속 시간을 초 단위로 지정하는 숫자입니다.
framerate	FLV 파일의 프레임 속도를 나타내는 숫자입니다.
height	FLV 파일의 높이를 픽셀 단위로 나타내는 숫자입니다.
seekpoints	사용 가능한 키프레임이 밀리초 단위의 타임스탬프로 나열되는 배열입니다. 선택 사항입니다.
tags	"ilst" 아톰의 정보를 나타내는 키-값 쌍의 배열이며 MP4 파일의 ID3 태그에 해당합니다. iTunes는 이러한 태그를 사용합니다. 사용 가능한 경우 아트웍을 표시하는 데 사용할 수 있습니다.
trackinfo	샘플 설명 ID를 비롯하여 MP4 파일의 모든 트랙에 대한 정보를 제공하는 객체입니다.
videocodecid	비디오 인코딩에 사용된 코덱 버전을 나타내는 문자열(예: "avc1", "VP6F")입니다.
videodatarate	FLV 파일의 비디오 데이터 속도를 나타내는 숫자입니다.
videoframerate	MP4 비디오의 프레임 속도입니다.
width	FLV 파일의 폭을 픽셀 단위로 나타내는 숫자입니다.

다음 표에서는 videocodecid 매개 변수로 가능한 값을 보여 줍니다.

videocodecid	코덱 이름
2	Sorenson H.263
3	스크린 비디오(SWF 버전 7 이상에서만 사용 가능)
4	VP6(SWF 버전 8 이상에서만 사용 가능)
5	알파 채널을 사용하는 VP6 비디오(SWF 버전 8 이상에서만 사용 가능)

다음 표에서는 audiocodecid 매개 변수로 가능한 값을 보여 줍니다.

audiocodecid	코덱 이름
0	압축되지 않음
1	ADPCM
2	Mp3
4	Nellymoser, 16kHz 모노
5	Nellymoser, 8kHz 모노
6	Nellymoser
10	AAC
11	Speex

onXMPData() 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

onXMPData() 콜백 함수는 Adobe F4V 또는 FLV 비디오 파일에 포함된 Adobe XMP(Extensible Metadata Platform)와 관련된 정보를 수신합니다. XMP 메타데이터에는 큐 포인트와 그 밖의 비디오 메타데이터가 들어 있습니다. XMP 메타데이터는 Flash Player 10 및 Adobe AIR 1.5부터 도입되었으며 후속 버전에서 지원됩니다.

다음 예제에서는 XMP 메타데이터의 큐 포인트 데이터를 처리합니다.

```
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);
        }
    }
}
```

이 예제에서는 startrekintro.f4v라는 짧은 비디오 파일에 대해 다음과 같은 추적 행을 생성합니다. 이 행에서는 내비게이션용 큐 포인트 데이터와 XMP 메타데이터의 이벤트 큐 포인트를 보여 줍니다.

```
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
```

참고: XMP 데이터에서 시간은 초가 아니라 DVA 틱 수로 저장됩니다. 큐 포인트 시간을 계산하려면 시작 시간을 프레임 속도로 나눕니다. 예를 들어 시작 시간 7695905817600을 프레임 속도 254016000000으로 나누면 30:30이 됩니다.

프레임 속도가 포함된 전체 원시 XMP 메타데이터를 보려면 onXMPData() 함수의 시작 부분에서 두 번째 및 세 번째 trace() 문 앞의 주석 식별자(//)를 제거합니다.

XMP에 대한 자세한 내용은 다음을 참조하십시오.

- partners.adobe.com/public/developer/xmp/topic.html
- www.adobe.com/devnet/xmp/

이미지 메타데이터 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

onImageData 이벤트는 AMF0 데이터 채널을 통해 이미지 데이터를 바이트 배열로 보냅니다. 데이터는 JPEG, PNG 또는 GIF 형식일 수 있습니다. onCuePoint 및 onMetaData에 대한 콜백 메서드를 정의할 때와 동일한 방법으로 onImageData() 콜백 메서드를 정의하여 이 정보를 처리할 수 있습니다. 다음 예제에서는 onImageData() 콜백 메서드를 사용하여 이미지 데이터에 액세스하고 이를 표시합니다.

```
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);
}
```

텍스트 메타데이터 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

onTextData 이벤트는 AMF0 데이터 채널을 통해 텍스트 데이터를 보냅니다. 텍스트 데이터는 UTF-8 형식이며 3GP Timed-Text 사양을 기반으로 하는 형식에 대한 정보를 포함합니다. 이 사양은 표준화된 부제 형식을 정의합니다. onCuePoint 또는 onMetaData에 대한 콜백 메서드를 정의할 때와 동일한 방법으로 onTextData() 콜백 메서드를 정의하여 이 정보를 처리할 수 있습니다. 다음 예제에서 onTextData() 메서드는 트랙 ID 번호와 해당 트랙 텍스트를 표시합니다.

```
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);
}
```

NetStream 작업 모니터링

Flash Player 10.3 이상, Adobe AIR 2.7 이상

NetStream 작업을 모니터링하여 미디어 사용을 분석 및 보고할 때 필요한 정보를 수집할 수 있습니다. 이 섹션에서 설명하는 모니터링 기능을 사용하면 미디어 측정 라이브러리를 만들어, 미디어를 표시하는 특정 비디오 플레이어에 대한 연결을 종료하지 않고도 데이터를 수집할 수 있습니다. 이를 통해 클라이언트 개발자는 라이브러리를 사용할 때 자신이 선호하는 비디오 플레이어를 선택할 수 있습니다. 응용 프로그램에서 NetStream 객체의 작업과 생성 여부를 모니터링하려면 NetMonitor 클래스를 사용합니다. NetMonitor 클래스는 지정한 시간에 존재하는 활성 NetStream의 목록을 제공하고, NetStream 객체가 생성될 때마다 이벤트를 전달합니다.

NetStream 객체는 재생 중인 미디어 유형에 따라 다음 표에 나열된 이벤트를 전달합니다.

이벤트	점진적 다운로드	RTMP 스트리밍	HTTP 스트리밍
NetStream.Play.Start	예	예	아니요
NetStream.Play.Stop	예	예	아니요
NetStream.Play.Complete	예	예	아니요
NetStream.SeekStart.Notify	예	예	예
NetStream.Seek.Notify	예	예	예
NetStream.Unpause.Notify	예	예	예
NetStream.Unpause.Notify	예	예	예

이벤트	점진적 다운로드	RTMP 스트리밍	HTTP 스트리밍
NetStream.Play.Transition	해당 없음	예	해당 없음
NetStream.Play.TransitionComplete	해당 없음	예	해당 없음
NetStream.Buffer.Full	예	예	예
NetStream.Buffer.Flush	예	예	예
NetStream.Buffer.Empty	예	예	예

NetStream 인스턴스와 연결된 NetStreamInfo 객체는 미디어에서 발생했던 마지막 메타데이터와 XMP 데이터 객체를 저장합니다.

HTTP 스트리밍을 통해 미디어가 재생되면 응용 프로그램이 미디어 스트림에 대한 완벽한 제어권을 보유하므로 NetStream.Play.Start, NetStream.Play.Stop 및 NetStream.Play.Complete는 전달되지 않습니다. 비디오 플레이어는 HTTP 스트림에 대한 이들 이벤트를 합성하고 전달합니다.

마찬가지로, 점진적 다운로드나 HTTP 미디어에 대해 NetStream.Play.Transition 및 NetStream.Play.TransitionComplete가 전달되지 않습니다. 동적 비트 전송률 전환은 RTMP 기능입니다. HTTP 스트림을 사용하는 비디오 플레이어가 유사한 기능을 지원하면 플레이어는 전환 이벤트를 합성하고 전달할 수 있습니다.

기타 도움말 항목

Adobe Developer Connection: Flash에서 비디오 사용량 측정

NetStream 이벤트 모니터링

두 이벤트 유형 netStatus 및 mediaTypeData는 사용량과 관련하여 유용한 데이터를 제공합니다. 또한 타이머를 사용하여 NetStream 재생 헤드의 위치를 정기적으로 기록할 수도 있습니다.

netStatus 이벤트는 사용자가 보았던 스트림 양을 확인하는 데 사용할 수 있는 정보를 제공합니다. 버퍼와 RTMFP 스트림 전환 이벤트 또한 netStatus 이벤트를 발생시킵니다.

mediaTypeData 이벤트는 메타데이터와 XMP 데이터 정보를 제공합니다. Netstream.Play.Complete 이벤트는 mediaTypeData 이벤트로 전달됩니다. 큐 포인트, 텍스트, 이미지 등 스트림에 포함된 기타 데이터도 mediaTypeData 이벤트를 통해 사용할 수 있습니다.

다음 예제는 응용 프로그램의 활성 NetStream에서 상태 및 데이터 이벤트를 모니터링하는 클래스를 만드는 방법을 보여 줍니다. 일반적으로 이러한 클래스는 분석과 관련된 데이터를 서버에 업로드하여 수집합니다.

```
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
            }
        }

    }
}
```

플레이어 도메인 감지

현재 보고 있는 미디어 내용이 위치한 웹 페이지의 URL과 도메인은 항상 쉽게 사용할 수 있는 것은 아닙니다. 호스팅하는 웹 사이트에서 허용하는 경우 ExternalInterface 클래스를 사용하여 정확한 URL을 가져올 수 있습니다. 하지만 타사 비디오 플레이어를 허용하는 일부 웹 사이트는 ExternalInterface 클래스의 사용을 허용하지 않습니다. 그러한 경우 Security 클래스의 pageDomain 속성에서 현재 웹 페이지의 도메인을 가져올 수 있습니다. 전체 URL은 사용자 보안과 개인 정보 보호를 위해 노출되지 않습니다.

페이지 도메인은 Security 클래스의 정적 pageDomain 속성에서 사용할 수 있습니다.

```
var domain:String = Security.pageDomain;
```

비디오 파일의 고급 항목

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 항목에서는 FLV 파일을 사용한 작업과 관련된 몇 가지 특수한 문제를 설명합니다.

서버에 호스트할 수 있도록 FLV 파일 구성

Flash Player 9 이상, Adobe AIR 1.0 이상

FLV 파일을 사용하여 작업하는 경우 FLV 파일 형식을 사용하도록 서버를 구성해야 합니다. MIME(Multipurpose Internet Mail Extensions)는 인터넷 연결에서 비 ASCII 파일을 전송할 수 있도록 해 주는 표준화된 데이터 사양입니다. 웹 브라우저 및 전자 메일 클라이언트는 다양한 MIME 유형을 해석하도록 구성되어 있으므로 비디오, 오디오, 그래픽 및 서식 있는 텍스트를 보내고 받을 수 있습니다. 웹 서버로부터 FLV 파일을 로드하려면 웹 서버에서 파일 확장명과 MIME 유형을 등록해야 할 수 있습니다. 웹 서버 설명서를 확인하십시오. FLV 파일의 MIME 유형은 video/x-flv입니다. FLV 파일 유형에 대한 자세한 정보는 다음과 같습니다.

- Mime 유형: video/x-flv
- 파일 확장명: .flv
- 필수 매개 변수: 없음

- 선택 매개 변수: 없음
- 인코딩 고려 사항: FLV 파일은 이진 파일입니다. 따라서 일부 응용 프로그램에서는 application/octet-stream 하위 유형을 설정해야 할 수도 있습니다.
- 보안 문제: 없음
- 제작 사양: www.adobe.com/go/video_file_format_kr

Microsoft에서는 Microsoft 인터넷 정보 서비스(IIS) 6.0 웹 서버의 스트리밍 미디어 처리 방식을 이전 버전과는 다르게 변경했습니다. 이전 버전의 IIS에서는 Flash 비디오를 스트리밍하기 위해 변경할 필요가 없었습니다. IIS 6.0의 경우 Windows 2003과 함께 제공되는 기본 웹 서버에서 FLV 파일이 스트리밍 미디어임을 인식하려면 MIME 유형이 필요합니다.

외부 FLV 파일을 스트리밍하는 SWF 파일이 Microsoft Windows Server® 2003에 있고 브라우저에서 표시되는 경우 SWF 파일이 올바르게 재생되지만 FLV 비디오는 스트리밍되지 않습니다. 이 문제는 이전 버전의 Flash 제작 도구인 Adobe Macromedia Flash Video Kit for Dreamweaver MX 2004를 사용하여 작성하는 파일을 비롯하여 Windows Server 2003에 있는 모든 FLV 파일에 영향을 미칩니다. 이러한 파일을 다른 운영 체제에서 테스트하면 올바르게 작동됩니다.

FLV 비디오를 스트리밍하도록 Microsoft Windows 2003 및 Microsoft IIS Server 6.0을 구성하는 방법에 대한 자세한 내용은 www.adobe.com/go/tn_19439_kr를 참조하십시오.

Macintosh에서 논리적 FLV 파일 대상 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

상대 슬래시(/)를 사용하는 경로를 사용하여 Apple® Macintosh® 컴퓨터의 비 시스템 드라이브에서 논리적 FLV를 재생할 경우 비디오가 재생되지 않습니다. 비 시스템 드라이브에는 CD-ROM, 파티션된 하드 디스크, 이동식 저장소 미디어 및 연결된 저장소 장치 등이 포함됩니다.

참고: 이러한 오류는 Flash Player 또는 AIR의 제한이 아니라 운영 체제의 제한 때문에 발생합니다.

Macintosh의 비 시스템 드라이브로부터 FLV 파일을 재생하려면 슬래시 기반 표기법(/)이 아닌 콜론 기반 표기법(:)을 사용하여 파일을 절대 경로로 참조해야 합니다. 다음 목록은 두 가지 표기 종류의 차이점을 보여 줍니다.

- 슬래시 기반 표기**:** myDrive/myFolder/myFLV.flv
- 콜론 기반 표기**:** (Mac OS®) myDrive:myFolder:myFLV.flv

비디오 예제: 비디오 주크박스

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 예제에서는 순서대로 재생할 비디오 목록을 동적으로 로드하는 간단한 비디오 주크박스를 만듭니다. 이를 통해 사용자가 여러 비디오 설명서를 찾아볼 수 있는 응용 프로그램이나 사용자가 요청한 비디오를 전달하기 전에 재생할 광고를 지정하는 응용 프로그램을 만들 수 있습니다. 이 예제에서는 ActionScript 3.0의 다음 기능을 설명합니다.

- 비디오 파일의 재생 진행률에 따라 재생 헤드 업데이트
- 비디오 파일의 메타데이터 수신 및 파싱
- 넷스트림에서 특정 코드 처리
- 동적으로 로드된 FLV 로드, 재생, 일시 정지 및 중지
- 넷스트림의 메타데이터를 기반으로 표시 목록의 Video 객체 크기 조절

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. 비디오 주크박스 응용 프로그램 파일은 Samples/VideoJukebox 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
VideoJukebox.fla 또는 VideoJukebox.mxml	Flex(MXML) 또는 Flash(FLA) 형식의 기본 응용 프로그램 파일입니다.
VideoJukebox.as	응용 프로그램의 기본 기능을 제공하는 클래스입니다.
playlist.xml	비디오 주크박스로 로드할 비디오 파일이 나열된 파일입니다.

외부 비디오 재생 목록 파일 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 playlist.xml 파일은 로드할 비디오와 비디오 재생 순서를 지정합니다. XML 파일을 로드하려면 다음 코드와 같이 URLLoader 객체와 URLRequest 객체를 사용해야 합니다.

```
uldr = new URLLoader();
uldr.addEventListener(Event.COMPLETE, xmlCompleteHandler);
uldr.load(new URLRequest(PLAYLIST_XML_URL));
```

이 코드는 VideoJukebox 클래스의 생성자 내에 있으므로 다른 코드가 실행되기 전에 먼저 파일이 로드됩니다. XML 파일 로드가 완료되면 다음 코드와 같이 곧바로 xmlCompleteHandler() 메서드가 호출되어 외부 파일을 XML 객체로 파싱합니다.

```
private function xmlCompleteHandler(event:Event):void
{
    playlist = XML(event.target.data);
    videosXML = playlist.video;
    main();
}
```

playlist XML 객체에는 외부 파일에서 가져온 원시 XML이 포함되어 있고, videos XML은 해당 비디오 노드가 포함된 XMLList 객체입니다. 다음 코드에서 샘플 playlist.xml 파일을 확인할 수 있습니다.

```
<videos>
    <video url="video/caption_video.flv" />
    <video url="video/cuepoints.flv" />
    <video url="video/water.flv" />
</videos>
```

마지막으로, xmlCompleteHandler() 메서드는 외부 FLV 파일 로드에 사용되는 NetConnection 및 NetStream 객체뿐만 아니라 표시 목록의 다양한 구성 요소 인스턴스를 설정하는 main() 메서드도 호출합니다.

사용자 인터페이스 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자 인터페이스를 만들려면 Button 인스턴스 다섯 개를 표시 목록으로 드래그하여 각 인스턴스에 playButton, pauseButton, stopButton, backButton, forwardButton이라는 이름을 지정해야 합니다.

다음 코드와 같이 각 Button 인스턴스에 대해 click 이벤트에 대한 핸들러를 할당해야 합니다.

```
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);
```

buttonClickHandler() 메서드는 다음 코드와 같이 switch 문을 사용하여 클릭된 Button 인스턴스를 확인합니다.

```
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;
    }
}
```

그런 다음 Slider 인스턴스를 표시 목록에 추가하고 volumeSlider라는 인스턴스 이름을 지정합니다. 다음 코드에서는 Slider 인스턴스의 liveDragging 속성을 true로 설정하고 Slider 인스턴스의 change 이벤트에 대한 이벤트 리스너를 정의합니다.

```
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);
```

ProgressBar 인스턴스를 표시 목록에 추가하고 positionBar라는 인스턴스 이름을 지정합니다. 다음 코드와 같이 mode 속성을 manual로 설정합니다.

```
positionBar.mode = ProgressBarMode.MANUAL;
```

마지막으로 Label 인스턴스를 표시 목록에 추가하고 positionLabel이라는 인스턴스 이름을 지정합니다. Timer 인스턴스에서 이 Label 인스턴스의 값을 설정합니다.

Video 객체의 메타데이터 수신

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player에서 로드된 각 비디오에 대한 메타데이터가 발견되면 NetStream 객체의 client 속성에 대해 onMetaData() 콜백 핸들러가 호출됩니다. 다음 코드에서는 Object를 초기화하고 지정된 콜백 핸들러를 설정합니다.

```
client = new Object();
client.onMetaData = metadataHandler;
```

metadataHandler() 메서드는 데이터를 코드 내에 이전에 정의되어 있던 메타 속성에 복사합니다. 그러면 전체 응용 프로그램에서 언제든지 현재 비디오의 메타데이터에 액세스할 수 있습니다. 다음으로 스테이지에 있는 Video 객체의 크기가 메타데이터에서 반환된 크기에 맞게 조절됩니다. 마지막으로 현재 재생 중인 비디오의 크기에 맞게 positionBar 진행률 막대 인스턴스가 이동되고 크기가 조절됩니다. 다음 코드에는 전체 metadataHandler() 메서드가 포함되어 있습니다.

```
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;
}
```

동적 비디오 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

각 비디오를 동적으로 로드하기 위해 응용 프로그램에서 NetConnection 및 NetStream 객체를 사용합니다. 다음 코드는 NetConnection 객체를 생성하고 null 값을 connect() 메서드에 전달합니다. Flash Player는 null을 지정하여 Flash Media Server 등의 서버에 연결하는 것이 아니라 로컬 서버에 있는 비디오에 연결합니다.

다음 코드에서는 NetConnection 및 NetStream 인스턴스를 모두 만들고, netStatus 이벤트에 대한 이벤트 리스너를 정의하며, client 객체를 client 속성에 할당합니다.

```
nc = new NetConnection();
nc.connect(null);

ns = new NetStream(nc);
ns.addEventListener(NetStatusEvent.NET_STATUS, netStatusHandler);
ns.client = client;
```

비디오의 상태가 변경될 때마다 netStatusHandler() 메서드가 호출됩니다. 비디오가 재생을 시작하거나 중지하는 경우, 비디오를 버퍼링하는 경우 또는 비디오 스트림을 찾을 수 없는 경우 등이 여기에 해당합니다. 다음 코드에서는 netStatusHandler() 이벤트를 나열합니다.

```
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.
    }
}
```

이전 코드에서는 Info 객체의 코드 속성을 평가하여 코드를 "NetStream.Play.Start", "NetStream.Play.StreamNotFound" 또는 "NetStream.Play.Stop"으로 필터링합니다. 다른 코드는 모두 무시됩니다. 넷스트림이 시작되면 코드에서 재생 헤드를 업데이트하는 Timer 인스턴스를 시작합니다. 넷스트림이 없거나 중지되면 Timer 인스턴스가 중지되고 응용 프로그램에서 재생 목록의 다음 비디오를 재생하려고 합니다.

Timer가 실행될 때마다 positionBar 진행률 막대 인스턴스는 ProgressBar 클래스의 setProgress() 메서드를 호출하여 현재 위치를 업데이트하고, positionLabel Label 인스턴스에는 현재 비디오의 경과 시간 및 총 시간이 업데이트됩니다.

```
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.
    }
}
```

비디오 볼륨 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

동적으로 로드된 비디오에 대해 NetStream 객체의 soundTransform 속성을 설정하여 볼륨을 제어할 수 있습니다. 비디오 주크박스 응용 프로그램을 활용하면 volumeSlider Slider 인스턴스 값을 변경하여 볼륨 수준을 수정할 수 있습니다. 다음 코드에서는 NetStream 객체의 soundTransform 속성으로 설정되어 있는 SoundTransform 객체에 Slider 구성 요소 값을 할당하여 볼륨 수준을 변경하는 방법을 보여 줍니다.

```
private function volumeChangeHandler(event:SliderEvent):void
{
    volumeTransform.volume = event.value;
    ns.soundTransform = volumeTransform;
}
```

비디오 재생 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램의 나머지 부분은 비디오가 비디오 스트림의 끝에 도달하거나 사용자가 이전 또는 다음 비디오를 건너뛸 때 비디오 재생을 제어합니다.

다음 메서드에서는 현재 선택된 인덱스의 XMLList에서 비디오 URL을 검색합니다.

```
private function getVideo():String
{
    return videosXML[idx].@url;
}
```

playVideo() 메서드는 NetStream 객체의 play() 메서드를 호출하여 현재 선택되어 있는 비디오를 로드합니다.

```
private function playVideo():void
{
    var url:String = getVideo();
    ns.play(url);
}
```

playPreviousVideo() 메서드는 현재 비디오 인덱스를 감소시키고, playVideo() 메서드를 호출하여 새 비디오 파일을 로드하며, 진행률 막대를 표시합니다.

```
private function playPreviousVideo():void
{
    if (idx > 0)
    {
        idx--;
        playVideo();
        positionBar.visible = true;
    }
}
```

마지막 메서드인 playNextVideo()는 비디오 인덱스를 증가시키고 playVideo() 메서드를 호출합니다. 현재 비디오가 재생 목록의 마지막 비디오인 경우, Video 객체에 대해 clear() 메서드가 호출되고 진행률 막대 인스턴스의 visible 속성이 false로 설정됩니다.

```
private function playNextVideo():void
{
    if (idx < (videosXML.length() - 1))
    {
        idx++;
        playVideo();
        positionBar.visible = true;
    }
    else
    {
        idx++;
        vid.clear();
        positionBar.visible = false;
    }
}
```

하드웨어 가속 프레젠테이션에 StageVideo 클래스 사용

Flash Player 10.2 이상

Flash Player는 H.264 디코딩에 하드웨어 가속을 사용하여 비디오 성능을 최적화합니다. 여기에 StageVideo API를 사용하면 성능을 한층 더 개선할 수 있습니다. 스테이지 비디오로 응용 프로그램이 하드웨어 가속 프레젠테이션을 이용할 수 있습니다.

StageVideo API를 지원하는 런타임은 다음과 같습니다.

- Flash Player 10.2 이상

참고: Flash Player 11.4/AIR 3.4 이상에서는 StageVideo를 사용한 카메라 입력을 사용할 수 있습니다.

스테이지 비디오 시작을 방문하면 스테이지 비디오 기능의 소스 코드를 다운로드하고 기능에 대한 자세한 내용을 확인할 수 있습니다.

StageVideo 빠른 시작 자습서는 스테이지 비디오를 사용한 작업을 참조하십시오.

StageVideo를 사용한 하드웨어 가속 소개

비디오 크기 조절, 색상 변환, 블리팅 등이 포함되는 하드웨어 가속 프레젠테이션으로 하드웨어 가속 디코딩 성능을 향상시킵니다. GPU(하드웨어) 가속을 지원하는 장치에서는 flash.media.StageVideo 객체를 사용하여 장치 하드웨어에서 직접 비디오를 처리할 수 있습니다. 직접 처리는 CPU의 작업 부담을 덜어 GPU가 비디오를 처리하는 동안 CPU가 다른 작업을 수행할 수 있도록 합니다. 반면 이전의 Video 클래스는 주로 소프트웨어 프레젠테이션을 사용합니다. 소프트웨어 프레젠테이션은 CPU에서 실행되므로 시스템 리소스를 많이 소모할 수 있습니다.

현재 전체 GPU 가속 기능을 제공하는 장치는 극소수입니다. 하지만 스테이지 비디오를 사용하면 응용 프로그램이 사용 가능한 하드웨어 가속을 최대한 활용할 수 있습니다.

StageVideo 클래스로 인해 Video 클래스가 무용지물이 되지는 않습니다. 두 클래스를 함께 활용하면 주어진 시간에 장치 리소스에서 허용하는 최적의 비디오를 표시할 수 있습니다. 응용 프로그램은 적절한 이벤트를 수신하고 필요에 따라 StageVideo 및 Video 간에 전환함으로써 하드웨어 가속을 활용합니다.

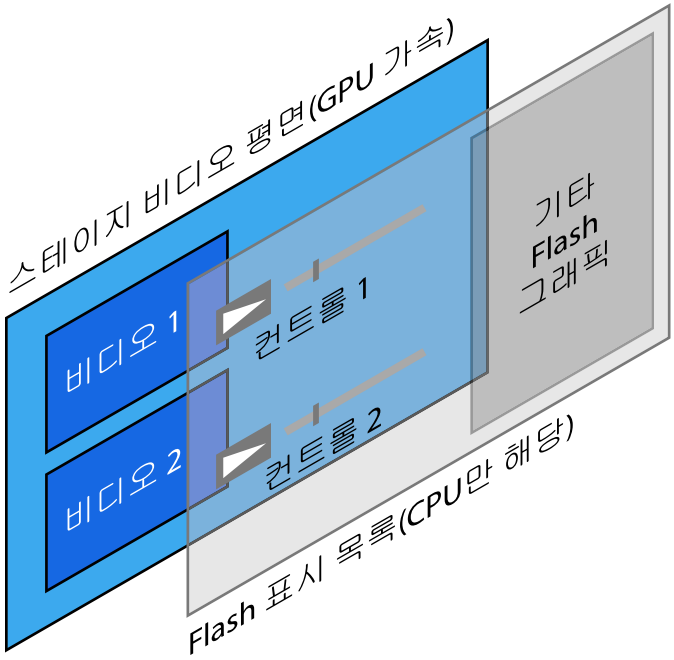
StageVideo 클래스를 구현하면 비디오를 사용하는 데 몇 가지 제약을 겪게 됩니다. 따라서 StageVideo를 구현하기 전에 지침을 읽어 보고 응용 프로그램에서 제약을 수용할 수 있는지 확인하십시오. 제약을 수용할 수 있으면 Flash Player에서 하드웨어 가속 프레젠테이션을 사용할 수 있다고 확인되는 경우에는 언제든지 StageVideo를 사용하십시오. 465페이지의 "지침 및 제한 사항"을 참조하십시오.

병렬 평면: 스테이지 비디오 및 Flash 표시 목록

스테이지 비디오 모델에서는 Flash Player가 비디오를 표시 목록과 분리할 수 있습니다. Flash Player는 두 개의 Z 순서로 정렬된 평면 간에 합성 표시를 나눕니다.

스테이지 비디오 평면 스테이지 비디오 평면은 배경에 배치되며 하드웨어 가속 비디오만 표시합니다. 이러한 설계로 인해, 스테이지 비디오 평면은 장치에서 하드웨어 가속이 지원되지 않거나 사용 불가능한 경우에는 사용할 수 없습니다. ActionScript에서 StageVideo 객체는 스테이지 비디오 평면에서 재생되는 비디오를 처리합니다.

Flash 표시 목록 평면 Flash 표시 목록 엔터티는 스테이지 비디오 평면 앞에 있는 평면에서 합성됩니다. 표시 목록 엔터티에는 재생 컨트롤을 비롯하여 런타임이 렌더링하는 모든 항목이 포함됩니다. 하드웨어 가속을 사용할 수 없는 경우 비디오는 Video 클래스 객체를 통해 이 평면에서만 재생될 수 있습니다. 스테이지 비디오는 항상 Flash 표시 목록 그래픽 뒤에서 표시됩니다.



비디오 표시 평면

StageVideo 객체는 화면에서 윈도우가 정렬된 사각 영역 안에 회전 없이 나타납니다. 스테이지 비디오 평면 뒤에 객체를 포갤 수는 없습니다. 하지만 Flash 표시 목록 평면을 사용하면 다른 그래픽을 스테이지 비디오 평면 위에 배치할 수 있습니다. 스테이지 비디오는 표시 목록과 동시에 실행됩니다. 따라서 두 메커니즘을 함께 사용하면 두 개의 개별적인 평면을 사용하는 통합된 시각 효과를 만들 수 있습니다. 예를 들어 앞쪽 패널을 배경에서 실행되는 스테이지 비디오 위에서 작동하는 재생 컨트롤에 사용할 수 있습니다.

스테이지 비디오 및 H.264 코덱

Flash Player 응용 프로그램에서 비디오 하드웨어 가속 구현은 두 단계로 이루어집니다.

1 비디오를 H.264로 인코딩

2 StageVideo API 구현

최선의 결과를 얻으려면 두 단계를 모두 수행합니다. H.264 코덱을 사용하면 비디오 디코딩에서 프레젠테이션에 이르기까지 하드웨어 가속을 최대한 활용할 수 있습니다.

스테이지 비디오는 GPU-CPU 다시 읽기를 할 필요가 없습니다. 즉, GPU가 더 이상 표시 목록 객체와의 합성을 위해 디코딩된 프레임을 CPU로 보내지 않습니다. 대신 GPU가 디코딩되고 렌더링된 프레임을 직접 표시 목록 객체 뒤에 있는 화면으로 전송합니다. 이 기법은 CPU와 메모리의 사용을 줄일 뿐 아니라 픽셀 화질도 높입니다.

기타 도움말 항목

430페이지의 “비디오 형식 이해”

지침 및 제한 사항

비디오를 전체 화면 모드에서 실행 중이면 장치가 하드웨어 가속을 지원하는 한 언제든지 스테이지 비디오를 사용할 수 있습니다. 하지만 Flash Player는 브라우저 내에서도 실행됩니다. 브라우저 컨텍스트 내에서 wmode 설정은 스테이지 비디오의 가용성에 영향을 미칩니다. 스테이지 비디오를 사용하려면 항상 wmode="direct"를 사용하는 것이 좋습니다. 스테이지 비디오는 전체 화면 모드가 아닌 경우에는 다른 wmode 설정과 호환되지 않습니다. 이러한 제한으로 인해 런타임에 스테이지 비디오가 사용 가능한 상태와 사용 불가능한 상태 사이에서 예상치 않게 자주 바뀔 수 있습니다. 예를 들어 사용자가 스테이지 비디오가 실행되는 동안 전체 화면 모드를 종료하면 비디오 컨텍스트는 브라우저로 되돌아갑니다. 브라우저의 wmode 매개 변수가 "direct"로 설정되어 있지 않으면 갑자기 스테이지 비디오를 사용할 수 없게 되는 경우가 발생할 수 있습니다. Flash Player는 일련의 이벤트를 통해 재생 컨텍스트 변경 사항을 응용 프로그램에 전달합니다. StageVideo API를 구현할 때 스테이지 비디오를 사용할 수 없는 경우에 대비하여 Video 객체를 백업으로 유지합니다.

스테이지 비디오는 하드웨어와의 직접적인 관계 때문에 일부 비디오 기능을 제한합니다. 스테이지 비디오는 다음과 같은 제한을 적용합니다.

- Flash Player는 동시에 비디오를 표시할 수 있는 StageVideo 객체의 수를 SWF 파일당 4개로 제한합니다. 그러나 장치 하드웨어 리소스에 따라 실제 제한 개수는 더 적을 수도 있습니다.
- 비디오 타이밍이 런타임에서 표시하는 내용의 타이밍과 동기화되지 않습니다.
- 비디오 표시 영역이 직사각형으로만 나타날 수 있습니다. 타원형이나 불규칙한 모양 등의 고급 표시 영역을 사용할 수 없습니다.
- 비디오를 회전할 수 없습니다.
- 비디오를 비트맵 캐싱할 수 없으며 또는 BitmapData 객체를 사용하여 액세스할 수 없습니다.
- 비디오에 필터를 적용할 수 없습니다.
- 비디오에 색상 변환을 적용할 수 없습니다.
- 비디오에 알파 값을 적용할 수 없습니다.
- 표시 목록 평면에 있는 객체에 적용되는 블렌드 모드는 스테이지 비디오에 적용되지 않습니다.
- 전체 픽셀 경계에만 비디오를 배치할 수 있습니다.
- GPU 렌더링은 주어진 장치 하드웨어에서 가장 효과적으로 사용할 수 있지만 여러 장치들 간에 픽셀이 완전히 동일하게 유지되지는 않습니다. 드라이버 및 플랫폼의 차이로 인해 약간의 변형이 발생합니다.
- 몇몇 장치는 필요한 색상 공간 중 일부를 지원하지 않습니다. 예를 들어 일부 장치는 H.264 표준인 BT.709를 지원하지 않습니다. 이 경우 빠른 표시를 위해 BT.601을 사용할 수 있습니다.

- 스테이지 비디오는 보통, 불투명, 투명 등의 WMODE 설정과 함께 사용할 수 없습니다. 스테이지 비디오는 전체 화면 모드가 아닌 경우 WMODE=direct만 지원합니다. WMODE는 Safari 4 이상 및 IE 9 이상에서는 효과가 없습니다.

보통 이러한 제한은 비디오 플레이어 응용 프로그램에는 영향을 주지 않습니다. 이상의 제한을 허용할 수 있으면 언제든지 스테이지 비디오를 사용하십시오.

기타 도움말 항목

151페이지의 “전체 화면 모드 작업”

StageVideo API 사용

스테이지 비디오는 비디오 재생 및 장치 성능을 향상시키는 런타임의 메커니즘입니다. 런타임은 이러한 메커니즘을 만들고 유지하며, 개발자의 역할은 이 메커니즘을 활용할 수 있도록 응용 프로그램을 구성하는 것입니다.

스테이지 비디오를 사용하려면 스테이지 비디오가 사용 가능하거나 사용 불가능한 때를 감지하는 이벤트 핸들러의 프레임워크를 구현합니다. 스테이지 비디오를 사용할 수 있다는 알림을 수신하면 Stage.stageVideos 속성으로부터 StageVideo 객체를 검색합니다. 런타임은 이 Vector 객체를 하나 이상의 StageVideo 객체로 채웁니다. 그러면 Video 객체가 아닌 제공된 StageVideo 객체 중 하나를 사용하여 스트리밍 비디오를 표시할 수 있습니다.

Flash Player에서 스테이지 비디오를 더 이상 사용할 수 없다는 알림을 수신하면 비디오 스트림을 다시 Video 객체로 전환합니다.

참고: StageVideo 객체를 만들 수는 없습니다.

Stage.stageVideos 속성

Stage.stageVideos 속성은 StageVideo 인스턴스에 액세스할 수 있도록 하는 Vector 객체입니다. 이 벡터에는 하드웨어 및 시스템 리소스에 따라 StageVideo 객체를 최대 4개까지 포함할 수 있습니다. 휴대 장치의 경우 그 수는 1개 또는 0개로 제한될 수 있습니다.

스테이지 비디오를 사용할 수 없는 경우 이 벡터는 아무 객체도 포함하지 않습니다. 런타임 오류를 방지하려면 최신 StageVideoAvailability 이벤트가 스테이지 비디오를 사용할 수 있다고 알리는 경우에만 이 벡터의 멤버에 액세스하십시오.

StageVideo 이벤트

StageVideo API 프레임워크에서 제공하는 이벤트는 다음과 같습니다.

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY Stage.stageVideos 속성이 변경되면 전송됩니다. StageVideoAvailabilityEvent.availability 속성은 AVAILABLE 또는 UNAVAILABLE 중 하나를 나타냅니다. 이 이벤트를 사용하면 Stage.stageVideos 벡터의 길이를 직접 확인하지 않고도 stageVideos 속성이 StageVideo 객체를 포함하는지 알아낼 수 있습니다.

StageVideoEvent.RENDER_STATE NetStream 또는 Camera 객체가 StageVideo 객체에 연결되어 재생되고 있는 경우에 전송됩니다. 현재 사용 중인 디코딩 유형(하드웨어, 소프트웨어 또는 사용 불가[아무 것도 표시되지 않음])을 나타냅니다. 이벤트 대상에는 비디오 뷰포트 크기 조정에 사용하기에 적합한 videoWidth 및 videoHeight 속성이 들어 있습니다.

중요: StageVideo 대상 객체로부터 구한 좌표는 표준 표시 목록에 속하지 않기 때문에 스테이지 좌표를 사용합니다.

VideoEvent.RENDER_STATE Video 객체가 사용되고 있는 경우에 전송되며, 소프트웨어 또는 하드웨어 가속 디코딩 중 어느 쪽을 사용하고 있는지 나타냅니다. 이 이벤트가 하드웨어 가속 디코딩을 나타낼 경우 가능하면 StageVideo 객체로 전환하십시오. Video 이벤트 대상에는 비디오 뷰포트 크기 조정에 사용하기에 적합한 videoWidth 및 videoHeight 속성이 들어 있습니다.

StageVideo 기능 구현을 위한 작업 과정

다음 최상위 단계에 따라 StageVideo 기능을 구현합니다.

1. StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY 이벤트를 수신하여 Stage.stageVideos 벡터가 변경된 경우를 확인합니다. 468페이지의 “StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY 이벤트 사용”을 참조하십시오.
2. StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY 이벤트가 스테이지 비디오를 사용할 수 있다고 보고하면 이벤트 핸들러 내의 Stage.stageVideos Vector 객체를 사용하여 StageVideo 객체에 액세스합니다.
3. StageVideo.attachNetStream()을 사용하여 NetStream 객체를 연결하거나, StageVideo.attachCamera()를 사용하여 Camera 객체를 연결합니다.
4. NetStream.play()를 사용하여 비디오를 재생합니다.
5. StageVideo 객체에서 StageVideoEvent.RENDER_STATE 이벤트를 수신하여 비디오 재생 상태를 확인합니다. 이 이벤트 수신은 비디오의 폭 및 높이 속성이 초기화되었거나 변경되었음을 나타내기도 합니다. 469페이지의 “StageVideoEvent.RENDER_STATE 및 VideoEvent.RENDER_STATE 이벤트 사용”을 참조하십시오.
6. Video 객체에서 VideoEvent.RENDER_STATE 이벤트를 수신합니다. 이 이벤트는 StageVideoEvent.RENDER_STATE와 동일한 상태를 제공하므로 GPU 가속의 사용 가능 여부를 결정하는 데도 사용할 수 있습니다. 이 이벤트 수신은 비디오의 폭 및 높이 속성이 초기화되었거나 변경되었음을 나타내기도 합니다. 469페이지의 “StageVideoEvent.RENDER_STATE 및 VideoEvent.RENDER_STATE 이벤트 사용”을 참조하십시오.

Initializing StageVideo 이벤트 리스너 초기화

응용 프로그램 초기화를 수행하는 동안 StageVideoAvailabilityEvent 및 VideoEvent 리스너를 설정하십시오. 예를 들어 이 리스너를 flash.events.Event.ADDED_TO_STAGE 이벤트 핸들러에서 초기화할 수 있습니다. 이 이벤트는 응용 프로그램이 스테이지상에서 계속 보이도록 합니다.

```
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);
        //...
    }
```

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY 이벤트 사용

StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY 핸들러에서 StageVideo의 가용성에 따라 Video 또는 StageVideo 객체 중 어느 것을 사용할지 결정하십시오. StageVideoAvailabilityEvent.availability 속성이 StageVideoAvailability.AVAILABLE로 설정되어 있으면 StageVideo를 사용합니다. 이 경우에 Stage.stageVideos 벡터를 활용해 하나 이상의 StageVideo 객체를 포함할 수 있습니다. Stage.stageVideos 속성에서 StageVideo 객체를 구하여 여기에 NetStream 객체를 연결합니다. StageVideo 객체는 항상 배경에 나타나므로 기존의 모든 Video 객체(항상 전경에 나타남)를 제거합니다. 또한 이 이벤트 핸들러를 활용하여 StageVideoEvent.RENDER_STATE 이벤트에 대한 리스너를 추가합니다.

StageVideoAvailabilityEvent.availability 속성이 StageVideoAvailability.UNAVAILABLE로 설정되어 있으면 StageVideo를 사용하지 않거나 Stage.stageVideos 벡터에 액세스하지 않도록 합니다. 이 경우 NetStream 객체를 Video 객체에 연결합니다. 마지막으로 StageVideo 또는 Video 객체를 스테이지에 추가하고 NetStream.play()를 호출합니다.

다음 코드에서는 StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY 이벤트를 처리하는 방법을 보여 줍니다.

```
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);
    }
}
```

중요: 응용 프로그램이 Stage.stageVideos[0]의 벡터 요소에 처음 액세스하면 기본 사각형이 0,0,0,0으로 설정되고 이동 및 확대/축소 속성이 기본값을 사용합니다. 항상 이 값을 기본 설정 값으로 재설정합니다. 비디오 뷰포트 크기를 계산하는 경우 StageVideoEvent.RENDER_STATE 또는 VideoEvent.RENDER_STATE 이벤트 대상의 videoWidth 및 videoHeight 속성을 사용할 수 있습니다.

이 샘플 응용 프로그램의 전체 소스 코드는 스테이지 비디오 시작에서 다운로드할 수 있습니다.

StageVideoEvent.RENDER_STATE 및 VideoEvent.RENDER_STATE 이벤트 사용

StageVideo 및 Video 객체는 디스플레이 환경이 변할 때 이를 응용 프로그램에 알려 주는 이벤트를 전송합니다. 이러한 이벤트는 StageVideoEvent.RENDER_STATE 및 VideoEvent.RENDER_STATE입니다.

StageVideo 또는 Video 객체는 NetStream 객체가 연결되어 재생이 시작되면 렌더링 상태 이벤트를 전달합니다. 또한 디스플레이 환경이 변경될 때(예: 비디오 뷰포트 크기가 조정될 때)도 이 이벤트를 전송합니다. 이러한 알림을 사용하여 뷰포트를 이벤트 대상 객체의 현재 videoHeight 및 videoWidth 값으로 재설정합니다.

보고되는 렌더링 상태는 다음과 같습니다.

- RENDER_STATUS_UNAVAILABLE
- RENDER_STATUS_SOFTWARE
- RENDER_STATUS_ACCELERATED

렌더링 상태는 현재 비디오를 재생하고 있는 클래스에 관계없이 하드웨어 가속 디코딩이 사용 중일 때를 나타냅니다. StageVideoEvent.status 속성에서 필요한 디코딩을 사용할 수 있는지 여부를 확인하십시오. 이 속성이 "unavailable"로 설정되면 StageVideo 객체가 비디오를 재생할 수 없습니다. 이 상태에서는 즉시 NetStream 객체를 Video 객체에 다시 연결해야 합니다. 그 밖의 상태는 응용 프로그램에 현재 렌더링 상태를 알립니다.

다음 표에는 Flash Player의 StageVideoEvent 및 VideoEvent 객체에 대한 모든 렌더링 상태 값의 의미가 나와 있습니다.

	VideoStatus.ACCELERATED	**VideoStatus.SOFTWARE**	**VideoStatus.UNAVAILABLE**
StageVideoEvent	디코딩과 프레젠테이션 모두 하드웨어에서 실행됩니다. (최고 성능)	프레젠테이션은 하드웨어에서, 디코딩은 소프트웨어에서 실행됩니다. (적절한 성능)	비디오 처리에 사용할 수 있는 GPU 리소스가 없으며 아무 것도 표시되지 않습니다. **Video 객체로 폴백합니다.**
VideoEvent	프레젠테이션은 소프트웨어에서, 디코딩은 하드웨어에서 실행됩니다. (신형 데스크톱 시스템에서만 적절한 성능이 발휘되며 전체 화면 성능이 저하됨)	프레젠테이션도 소프트웨어에서, 디코딩도 소프트웨어에서 실행됩니다. (최악의 성능. 전체 화면 성능이 저하됨)	(해당 사항 없음)

색상 공간

스테이지 비디오는 기본 하드웨어 기능을 사용하여 색상 공간을 지원합니다. SWF 내용은 기본 색상 공간을 알려 주는 메타데이터를 제공할 수 있습니다. 그러나 장치 그래픽 하드웨어는 해당 색상 공간을 사용할 수 있는지 여부를 결정합니다. 장치에 따라 여러 색상 공간을 지원할 수도 있고 하나도 지원하지 못할 수도 있습니다. 하드웨어에서 요청된 색상 공간을 지원하지 않는 경우 Flash Player는 지원되는 색상 공간 중에서 가장 근사하게 일치하는 색상 공간을 찾습니다.

하드웨어에서 지원하는 색상 공간을 쿼리하려면 StageVideo.colorSpaces 속성을 사용하십시오. 이 속성은 지원되는 색상 공간의 목록을 String 벡터로 반환합니다.

```
var colorSpace:Vector.<String> = stageVideo.colorSpaces();
```

현재 재생 중인 비디오에서 사용되는 색상 공간을 알아보려면 StageVideoEvent.colorSpace 속성을 확인하십시오. 이 속성은 StageVideoEvent.RENDER_STATE 이벤트의 이벤트 핸들러에서 확인할 수 있습니다.

```
var currColorSpace:String;

//StageVideoEvent.RENDER_STATE event handler
private function stageVideoRenderState(event:Object):void
{
    //...
    currColorSpace = (event as StageVideoEvent).colorSpace;
    //...
}
```

Flash Player에서 지원되지 않는 색상 공간의 대체 공간을 찾을 수 없는 경우 스테이지 비디오는 기본 색상 공간인 BT.601을 사용합니다. 예를 들어 H.264 인코딩의 비디오 스트림은 보통 BT.709 색상 공간을 사용합니다. 장치 하드웨어에서 BT.709를 지원하지 않는 경우 colorSpace 속성은 BT601을 반환합니다. StageVideoEvent.colorSpace의 unknown 값은 하드웨어에서 색상 공간의 쿼리 방법을 제공하지 않는다는 것을 나타냅니다.

응용 프로그램에서 현재의 색상 공간을 허용할 수 없는 것으로 간주하면 StageVideo 객체에서 Video 객체로 전환하도록 선택할 수 있습니다. Video 클래스는 소프트웨어 합성을 통해 모든 색상 공간을 지원합니다.

26장: 카메라를 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript를 사용하여 표시하고 조작할 수 있는 비디오 데이터의 소스로 사용자의 컴퓨터에 연결된 카메라를 사용할 수 있습니다. Camera 클래스는 컴퓨터 또는 장치 카메라로 작업하기 위해 ActionScript에 내장된 메커니즘입니다.

휴대 장치에서는 CameraUI 클래스를 사용할 수도 있습니다. CameraUI 클래스는 사용자가 정지 이미지 또는 비디오를 캡처할 수 있도록 별도의 카메라 응용 프로그램을 실행합니다. 사용자가 작업을 마치면 응용 프로그램에서 MediaPromise 객체를 통해 이미지나 비디오에 액세스할 수 있습니다.

기타 도움말 항목

Christian Cantrell: 플랫폼에 관계없이 CameraUI를 사용하는 방법

Michaël CHAIZE: Android, AIR 및 카메라

Camera 클래스 이해

Flash Player 9 이상, Adobe AIR 1.0 이상

Camera 객체를 사용하면 사용자의 로컬 카메라에 연결하고 비디오를 로컬로 사용자에게 다시 브로드캐스팅하거나 원격으로 Flash Media Server 등의 서버에 브로드캐스팅할 수 있습니다.

Camera 클래스를 사용하여 사용자 카메라에 대한 다음과 같은 정보에 액세스할 수 있습니다.

- 사용자의 컴퓨터 또는 장치에 설치된 카메라 중 사용할 수 있는 카메라
- 카메라 설치 여부
- 사용자의 카메라에 대한 Flash Player의 액세스 허용 여부
- 현재 활성화된 카메라
- 캡처 중인 비디오의 폭 및 높이

Camera 클래스에는 Camera 객체를 사용하는 데 유용한 몇 가지 메서드와 속성이 있습니다. 예를 들어 정적 Camera.names 속성에는 사용자 컴퓨터에 현재 설치되어 있는 카메라 이름 배열이 포함됩니다. name 속성을 사용하여 현재 활성화된 카메라의 이름을 표시할 수도 있습니다.

참고: 네트워크를 통해 카메라 비디오를 스트리밍할 때는 항상 네트워크 중단을 처리할 수 있어야 합니다. 네트워크 중단은 특히 휴대 장치에서 여러 가지 이유로 발생할 수 있습니다.

화면에 카메라 내용 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

카메라에 연결하는 데 필요한 코드는 NetConnection 및 NetStream 클래스를 사용하여 비디오를 로드할 때보다 더 적을 수 있습니다. 그러나 Flash Player를 사용할 경우 카메라에 액세스할 수 있게 Flash Player를 카메라에 연결하려면 사용자가 이를 허용해야 하므로 Camera 클래스가 복잡해질 수 있습니다.

다음 코드에서는 Camera 클래스로 사용자의 로컬 카메라에 연결하는 방법을 설명합니다.

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

참고: Camera 클래스에는 생성자 메서드가 없습니다. 새 Camera 인스턴스를 만들려면 정적 Camera.getCamera() 메서드를 사용합니다.

카메라 응용 프로그램 설계

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자의 카메라에 연결되는 응용 프로그램을 작성할 때 다음 사항을 고려해야 합니다.

- 사용자가 카메라를 현재 설치해 둔 상태인지 확인합니다. 사용 가능한 카메라가 없는 상황을 처리합니다.
- 사용자가 카메라에 대한 액세스를 명시적으로 허용했는지 확인합니다(Flash Player의 경우만 해당). 보안을 위하여 Flash Player가 [Flash Player 설정] 대화 상자를 표시하여 사용자가 카메라 액세스를 허용하거나 거부하도록 합니다. 이 대화 상자는 Flash Player에서 사용자의 허용 없이 카메라에 연결하여 비디오 스트림을 브로드캐스팅할 수 없게 합니다. 사용자가 [허용]을 클릭하면 응용 프로그램에서 사용자의 카메라에 연결할 수 있으며, [거부]를 클릭하면 응용 프로그램에서 사용자의 카메라에 액세스할 수 없습니다. 응용 프로그램에서 두 경우를 모두 적절하게 처리해야 합니다.
- Camera 클래스가 응용 프로그램에서 지원하는 장치 프로파일에 대해 지원되는지 확인합니다(AIR의 경우만 해당).
- Camera 클래스는 휴대 장치 브라우저에서는 지원되지 않습니다.
- Camera 클래스는 GPU 렌더링 모드를 사용하는 휴대 장치 AIR 응용 프로그램에서는 지원되지 않습니다.
- 모바일 장치에서는 한 번에 하나의 카메라만 활성화할 수 있습니다.

기타 도움말 항목

Christophe Coenraets: Multi-User Video Tic-Tac-Toe

Mark Doherty: Android Radar app(source)

사용자의 카메라에 연결

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자의 카메라에 연결하는 첫 번째 단계는 Camera 유형의 변수를 만든 다음 이 변수를 정적 Camera.getCamera() 메서드의 반환 값으로 초기화하여 새 Camera 인스턴스를 만드는 것입니다.

다음 단계는 새 Video 객체를 만들고 이 객체에 Camera 객체를 연결하는 것입니다.

세 번째 단계는 표시 목록에 Video 객체를 추가하는 것입니다. Camera 클래스는 DisplayObject 클래스를 확장하지 않아 표시 목록에 직접 추가할 수 없으므로 2단계와 3단계를 수행해야 합니다. 카메라에 캡처된 비디오를 표시하려면 새 Video 객체를 만들고 attachCamera() 메서드를 호출합니다.

다음 코드에서는 이러한 세 단계를 보여 줍니다.

```
var cam:Camera = Camera.getCamera();
var vid:Video = new Video();
vid.attachCamera(cam);
addChild(vid);
```

사용자에게 설치된 카메라가 없는 경우 응용 프로그램은 아무 것도 표시하지 않습니다.

실제로 응용 프로그램에 사용할 때는 몇 가지 단계를 추가로 수행해야 합니다. 자세한 내용은 474페이지의 "카메라 설치 여부 확인" 및 475페이지의 "카메라 액세스 허용 여부 확인"을 참조하십시오.

기타 도움말 항목

Lee Brimelow: How to access the camera on Android devices

카메라 설치 여부 확인

Flash Player 9 이상, Adobe AIR 1.0 이상

Camera 인스턴스에서 메서드나 속성을 사용하기 전에 사용자가 카메라를 설치했는지 확인해야 합니다. 다음 두 가지 방법으로 카메라 설치 여부를 확인할 수 있습니다.

- 사용할 수 있는 카메라 이름 배열이 포함된 정적 Camera.names 속성을 확인합니다. 대부분의 사용자는 한 번에 두 대 이상의 카메라를 설치하지 않으므로 일반적으로 이 배열에는 하나 이하의 문자열이 포함됩니다. 다음 코드에서는 Camera.names 속성을 점검하여 사용 가능한 카메라가 있는지 여부를 확인하는 방법을 보여 줍니다.

  ```
  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.");
  }
  ```

- 정적 Camera.getCamera() 메서드가 반환하는 값을 확인합니다. 사용 가능하거나 설치되어 있는 카메라가 없는 경우 이 메서드는 null을 반환하고, 카메라가 있는 경우에는 Camera 객체의 참조를 반환합니다. 다음 코드에서는 Camera.getCamera() 메서드를 점검하여 사용 가능한 카메라가 있는지 여부를 확인하는 방법을 보여 줍니다.

  ```
  var cam:Camera = Camera.getCamera();
  if (cam == null)
  {
      trace("User has no cameras installed.");
  }
  else
  {
      trace("User has at least 1 camera installed.");
  }
  ```

Camera 클래스는 DisplayObject 클래스를 확장하지 않으므로 addChild() 메서드를 사용하여 표시 목록에 직접 추가할 수 없습니다. 카메라에 캡처된 비디오를 표시하려면 새 Video 객체를 만들고 Video 인스턴스에 대해 attachCamera() 메서드를 호출해야 합니다.

다음 코드에서는 카메라가 있는 경우 카메라를 연결하는 방법을 보여 줍니다. 카메라가 없는 경우 응용 프로그램은 아무 것도 표시하지 않습니다.

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video();
    vid.attachCamera(cam);
    addChild(vid);
}
```

휴대 장치 카메라

Camera 클래스는 Flash Player 런타임의 모바일 브라우저에서는 지원되지 않습니다.

휴대 장치의 AIR 응용 프로그램에서 장치에 있는 하나 이상의 카메라에 액세스할 수 있습니다. 모바일 장치에서는 정방향 카메라와 역방향 카메라에 모두 액세스할 수 있지만 카메라 출력은 항상 하나만 표시할 수 있습니다. 두 번째 카메라에 연결하면 첫 번째 카메라의 연결이 해제됩니다. 정면 카메라는 iOS에서 수평으로 반사되지만 Android에서는 그렇지 않습니다.

카메라 액세스 허용 여부 확인

Flash Player 9 이상, Adobe AIR 1.0 이상

AIR 응용 프로그램 샌드박스에서 응용 프로그램은 사용자가 허용하지 않아도 카메라에 액세스할 수 있습니다. 그러나 Android에서는 응용 프로그램이 응용 프로그램 설명자에 Android CAMERA 권한을 지정해야 합니다.

사용자가 Flash Player에서 카메라에 액세스할 수 있도록 명시적으로 허용해야 Flash Player에서 카메라 출력을 표시할 수 있습니다. attachCamera() 메서드가 호출되면 Flash Player가 [Flash Player 설정] 대화 상자를 표시하여 사용자에게 Flash Player에서 카메라 및 마이크에 액세스하도록 허용할 것인지 여부를 선택하도록 요청합니다. 사용자가 [허용] 버튼을 클릭하면 Flash Player는 스테이지의 Video 인스턴스에 카메라의 출력을 표시합니다. 사용자가 [거부] 버튼을 클릭하면 Flash Player에서 카메라에 연결할 수 없으며 Video 객체가 아무 것도 표시하지 않습니다.

사용자가 카메라에 대한 액세스를 허용했는지 여부를 확인하려면 다음 코드와 같이 카메라의 status 이벤트(StatusEvent.STATUS)를 수신합니다.

```
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"
}
```

사용자가 [허용] 또는 [거부]를 클릭하면 곧바로 statusHandler() 함수가 호출됩니다. 다음 두 메서드 중 하나를 사용하여 사용자가 클릭한 버튼을 확인할 수 있습니다.

- statusHandler() 함수의 event 매개 변수에는 "Camera.Muted" 또는 "Camera.Unmuted" 문자열이 들어 있는 코드 속성이 포함되어 있습니다. 값이 "Camera.Muted"이면 사용자가 [거부] 버튼을 클릭한 것이므로 Flash Player에서 카메라에 액세스할 수 없습니다. 다음 코드에서 이를 확인할 수 있습니다.

```
function statusHandler(event:StatusEvent):void
{
    switch (event.code)
    {
        case "Camera.Muted":
            trace("User clicked Deny.");
            break;
        case "Camera.Unmuted":
            trace("User clicked Accept.");
            break;
    }
}
```

- Camera 클래스에는 muted라는 읽기 전용 속성이 포함되어 있으며 이 속성은 사용자가 카메라에 대한 액세스를 거부했는지(true) 아니면 허용했는지(false)를 Flash Player [개인 정보] 패널에 지정합니다. 다음 코드에서 이를 확인할 수 있습니다.

```
function statusHandler(event:StatusEvent):void
{
    if (cam.muted)
    {
        trace("User clicked Deny.");
    }
    else
    {
        trace("User clicked Accept.");
    }
}
```

전달할 상태 이벤트를 확인하여 카메라 액세스에 대한 사용자 허용 또는 거부를 처리하고 이에 맞게 정리하는 코드를 작성할 수 있습니다. 예를 들어 사용자가 [거부] 버튼을 클릭할 경우 비디오 채팅에 참여하려면 [허용]을 클릭해야 한다는 메시지를 사용자에게 표시하거나, 표시 목록의 Video 객체를 삭제하여 사용 가능한 시스템 리소스를 늘릴 수 있습니다.

AIR에서는 카메라를 사용하기 위한 권한이 동적이지 않기 때문에 Camera 객체에서 status 이벤트를 전달하지 않습니다.

카메라 비디오 품질 최대화

Flash Player 9 이상, Adobe AIR 1.0 이상

기본적으로 Video 클래스의 새 인스턴스 크기는 폭 320픽셀 x 높이 240픽셀입니다. 비디오 품질을 최대화하려면 항상 Video 객체 크기를 Camera 객체에서 반환하는 비디오 크기에 맞춰야 합니다. Camera 클래스의 width 및 height 속성을 사용하여 Camera 객체의 폭과 높이를 구할 수 있습니다. 그런 다음, 다음 코드 예제와 같이 Video 객체의 width 및 height 속성을 Camera 객체 크기에 일치하도록 설정하거나, 카메라의 폭과 높이를 Video 클래스의 생성자 메서드에 전달할 수 있습니다.

```
var cam:Camera = Camera.getCamera();
if (cam != null)
{
    var vid:Video = new Video(cam.width, cam.height);
    vid.attachCamera(cam);
    addChild(vid);
}
```

getCamera() 메서드는 Camera 객체에 대한 참조를 반환하므로(또는 사용 가능한 카메라가 없는 경우 null을 반환), 사용자가 카메라에 대한 액세스를 거부한 경우에도 카메라의 메서드 및 속성에 액세스할 수 있습니다. 그러면 카메라의 기본 높이와 폭을 사용하여 비디오 인스턴스의 크기를 설정할 수 있습니다.

```
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);
}
```

전체 화면 모드에 대한 자세한 내용은 148페이지의 “Stage 속성 설정”의 전체 화면 모드 단원을 참조하십시오.

카메라 상태 모니터링

Flash Player 9 이상, Adobe AIR 1.0 이상

Camera 클래스에는 Camera 객체의 현재 상태를 모니터링할 수 있는 몇 가지 속성이 있습니다. 예를 들어 다음 코드에서는 Timer 객체와 표시 목록의 텍스트 필드 인스턴스를 사용하여 카메라의 여러 속성을 표시합니다.

```
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");
}
```

매 1/10초(100밀리초)마다 Timer 객체의 timer 이벤트가 전달되고 timerHandler() 함수가 표시 목록의 텍스트 필드를 업데이트합니다.

27장: 디지털 권한 관리 사용

Flash Player 10.1 이상, Adobe AIR 1.5 이상

Adobe® Access™는 내용 보호 솔루션입니다. 유료 내용에 대한 원활한 액세스를 제공하므로 내용 소유자, 배포자 및 광고자가 새로운 수익원을 실현할 수 있습니다. 제작자는 Adobe Access로 내용을 암호화하고 정책을 생성하는 한편 라이센스를 발급합니다. Adobe Flash Player 및 Adobe AIR에는 DRM 라이브러리인 Adobe Access 모듈이 통합되어 있습니다. 이 모듈은 런타임에서 Adobe Access 라이센스 서버와 통신하여 보호된 내용을 재생할 수 있도록 합니다. 그에 따라 런타임은 Adobe Access로 보호되고 Flash Media Server로 배포되는 내용의 수명 주기를 완전하게 구현합니다.

내용 공급자는 Adobe Access를 통해 무료 내용과 유료 내용을 모두 제공할 수 있습니다. 예를 들어 소비자가 광고 없이 TV 프로그램을 보기 원하는 경우 내용 제작자에게 등록하고 비용을 지불합니다. 그런 다음 사용자 자격 증명을 입력하여 액세스한 후 광고 없이 프로그램을 재생할 수 있습니다.

또 다른 예로, 소비자가 인터넷에 액세스할 수 없는 상태로 이동 중에 오프라인으로 내용을 보려는 경우 이러한 오프라인 작업 과정이 AIR 응용 프로그램에서 지원됩니다. 사용자는 내용 제작자에게 신청하고 비용을 지불한 후 제작자 웹 사이트에 있는 내용에 액세스하여 내용 및 관련 AIR 응용 프로그램을 다운로드할 수 있습니다. 그리고 AIR 응용 프로그램을 통해 허가된 기간 동안 오프라인으로 내용을 볼 수 있습니다. 또한 도메인을 사용하여 동일한 장치 그룹의 다른 장치와 내용을 공유할 수 있습니다(**3.0의 새로운 기능**).

Adobe Access는 사용자 인증이 필요 없는 익명 액세스도 지원합니다. 예를 들어 제작자가 익명 액세스를 사용하여 광고가 지원되는 내용을 배포할 수 있습니다. 또한 제작자는 익명 액세스를 사용하여 지정된 기간(일) 동안 현재 내용에 대한 무료 액세스를 허용할 수도 있습니다. 또한 내용 공급자는 해당 내용에 필요한 플레이어의 유형과 버전을 지정 및 제한할 수 있습니다.

사용자가 Flash Player 또는 Adobe AIR로 보호된 내용을 재생하려고 시도하면 응용 프로그램에서 DRM API를 호출하고 DRM API에서 보호된 내용을 재생하기 위한 작업 과정을 시작합니다. 먼저 런타임에서 Adobe Access 모듈을 통해 라이센스 서버에 연결하고 라이센스 서버에서는 필요한 경우 사용자를 인증하고, 보호된 내용을 재생할 수 있도록 허용하는 라이센스를 발급합니다. 런타임에서 해당 라이센스를 받은 후 내용을 해독하여 재생합니다.

여기서는 응용 프로그램에서 Adobe Access로 보호되는 내용을 재생할 수 있게 하는 방법을 설명합니다. Adobe Access를 사용하여 내용을 암호화하거나 정책을 유지 관리하는 방법은 이 문서에서 다루지 않습니다. 하지만 여기서는 Adobe Access 라이센스 서버와 통신하여 사용자를 인증하고 라이센스를 가져오며 Adobe Access로 보호된 내용을 재생하기 위해 응용 프로그램을 설계하는 것으로 가정합니다.

정책 생성을 비롯한 Adobe Access에 대한 개요는 Adobe Access에 포함된 설명서를 참조하십시오.

기타 도움말 항목

flash.net.drm 패키지

flash.net.NetConnection

flash.net.NetStream

보호된 내용 작업 과정 이해

Flash Player 10.1 이상, Adobe AIR 2.0 이상

중요: Flash Player 11.5 이상에는 Adobe Access 모듈이 통합되어 있으므로 SystemUpdater.update(SystemUpdaterType.DRM))를 호출하는 업데이트 단계가 불필요합니다. 여기에는 다음 브라우저와 플랫폼이 포함됩니다.

- Flash Player 11.5 ActiveX 컨트롤(Intel 프로세서 기반 Windows 8의 Internet Explorer를 제외한 모든 플랫폼)

- Flash Player 11.5 플러그인(모든 브라우저)
- Adobe AIR(데스크톱 및 모바일)

따라서 다음과 같은 경우에는 업데이트 단계를 **계속 실행해야** 합니다.

- Intel 프로세서 기반 Windows 8의 Internet Explorer
- Flash Player 11.4 이하(Google Chrome 22 이상(모든 플랫폼) 또는 21 이상(Windows)은 제외)

참고: Flash Player 11.5 이상이 설치된 시스템에서는 여전히 SystemUpdater.update(SystemUpdaterType.DRM)를 안전하게 호출할 수 있지만 아무것도 다운로드되지 않습니다.

다음의 개략적인 작업 과정은 응용 프로그램에서 보호된 내용을 가져오고 재생하는 방법을 보여 줍니다. 이 작업 과정은 응용 프로그램이 Adobe Access로 보호된 내용을 재생하기 위해 설계된 것으로 가정합니다.

1. 내용 메타데이터를 가져옵니다.
2. 필요한 경우 Flash Player에 대한 업데이트를 처리합니다.
3. 라이센스를 로컬로 사용할 수 있는지 여부를 확인합니다. 사용할 수 있는 경우 라이센스를 로드하고 7단계로 이동합니다. 사용할 수 없는 경우 4단계로 이동합니다.
4. 인증이 필요한지 여부를 확인합니다. 필요하지 않은 경우 7단계로 이동합니다.
5. 인증이 필요한 경우 사용자에게서 인증 자격 증명을 얻어 라이센스 서버로 전달합니다.
6. 도메인 등록이 필요한 경우 도메인에 가입합니다(AIR 3.0 이상).
7. 인증이 성공하면 서버에서 라이센스를 다운로드합니다.
8. 내용을 재생합니다.

오류가 발생하지 않았고 사용자에게 내용을 볼 수 있는 권한이 있는 경우 NetStream 객체가 DRMStatusEvent 객체를 전달합니다. 그런 다음 응용 프로그램이 재생을 시작합니다. DRMStatusEvent 객체에는 사용자의 정책과 권한을 식별하는 관련 바우처 정보가 포함되어 있습니다. 예를 들어 오프라인에서 내용을 사용할 수 있게 할 수 있는지 여부 또는 라이센스가 만료되는 시기와 같은 정보가 포함됩니다. 응용 프로그램은 이 데이터를 사용하여 사용자의 정책 상태를 사용자에게 알릴 수 있습니다. 예를 들어 사용자가 내용을 볼 수 있는 남은 날짜 수를 상태 표시줄에 표시할 수 있습니다.

사용자에게 오프라인 액세스가 허용되는 경우 바우처가 캐시되고 암호화된 내용이 사용자의 시스템으로 다운로드됩니다. 내용은 라이센스 캐시 기간에 정의된 기간 동안 액세스할 수 있습니다. 이벤트의 detail 속성에 "DRM.voucherObtained"가 포함됩니다. 응용 프로그램은 해당 내용을 오프라인으로 사용할 수 있도록 하기 위해 내용을 로컬에 저장할 위치를 결정합니다. 또한 DRMManager 클래스를 사용하여 바우처를 미리 로드할 수도 있습니다.

참고: 바우처의 캐시 및 미리 로드는 AIR 및 Flash Player에서 모두 지원됩니다. 그러나 암호화된 내용을 다운로드 및 저장하는 것은 AIR에서만 지원됩니다.

해당 응용 프로그램에서 오류 이벤트를 명시적으로 처리해야 합니다. 이러한 이벤트에는 사용자가 유효한 자격 증명을 입력하지만 암호화된 내용을 보호하는 바우처에서 내용에 대한 액세스를 제한하는 경우가 포함됩니다. 예를 들어 내용에 액세스할 수 있는 권한에 대한 비용을 지불하지 않은 경우 인증된 사용자가 내용에 액세스할 수 없습니다. 동일한 제작자의 등록된 멤버 두 명이 둘 중 한 명만 비용을 지불한 내용을 공유하려고 하는 경우에도 이러한 오류가 발생할 수 있습니다. 응용 프로그램은 사용자에게 오류에 대해 알리고 대안을 제시해야 합니다. 일반적인 대안은 등록하고 보기 권한에 대한 비용을 지불하는 방법과 관련한 지침입니다.

상세한 API 작업 과정

Flash Player 10.1 이상, AIR 2.0 이상

여기서는 보호된 내용에 대한 작업 과정을 자세하게 살펴봅니다. 이 작업 과정에서는 Adobe Access로 보호된 내용을 재생하는 데 사용되는 특정 API에 대해 설명합니다.

1 URLLoader 객체를 사용하여 보호된 내용의 메타데이터 파일 바이트를 로드합니다. 이 객체를 metadata_bytes와 같은 변수로 설정합니다.

Adobe Access로 제어되는 모든 내용에는 Adobe Access 메타데이터가 있습니다. 내용을 패키지화하는 경우 내용과 함께 이 메타데이터를 별도의 메타데이터 파일(.metadata)로 저장할 수 있습니다. 자세한 내용은 Adobe Access 설명서를 참조하십시오.

2 DRMContentData 인스턴스를 만듭니다. 다음 코드를 try-catch 블록에 넣습니다.

new DRMContentData(***metadata_bytes***)

여기서 metadata_bytes는 1단계에서 가져온 URLLoader 객체입니다.

3 (Flash Player에만 해당) 런타임에서 Adobe Access 모듈이 있는지 여부를 확인합니다. 찾을 수 없는 경우 DRMErrorEvent 오류 코드 3344 또는 DRMErrorEvent 오류 코드 3343과 함께 IllegalOperationError가 발생합니다.

이 오류를 처리하려면 SystemUpdater API를 사용하여 Adobe Access 모듈을 다운로드합니다. 이 모듈이 다운로드되면 SystemUpdater 객체에서 COMPLETE 이벤트를 전달합니다. 여기에는 이 이벤트에 대한 이벤트 리스너도 포함되어 이 이벤트 전달 시 2단계로 돌아갑니다. 다음 코드는 이러한 단계를 보여 줍니다.

```
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);
}
```

플레이어 자체를 업데이트해야 하는 경우 상태 이벤트가 전달됩니다. 이 이벤트 처리에 대한 자세한 내용은 495페이지의 “업데이트 이벤트 수신”을 참조하십시오.

참고: AIR 응용 프로그램의 경우 AIR 설치 프로그램에서 Adobe Access 모듈 업데이트 및 필요한 런타임 업데이트를 처리합니다.

4 DRMManager 객체에서 전달된 DRMStatusEvent 및 DRMErrorEvent를 수신하기 위한 리스너를 만듭니다.

```
DRMManager.addEventListener(DRMStatusEvent.DRM_STATUS, onDRMStatus);
DRMManager.addEventListener(DRMErrorEvent.DRM_ERROR, onDRMError);
```

DRMStatusEvent 리스너에서는 바우처가 유효한지(null이 아닌지) 확인하고 DRMErrorEvent 리스너에서는 DRMErrorEvents를 처리합니다. 자세한 내용은 487페이지의 “DRMStatusEvent 클래스 사용” 및 492페이지의 “DRMErrorEvent 클래스 사용”을 참조하십시오.

5 내용을 재생하는 데 필요한 바우처(라이센스)를 로드합니다.

먼저, 로컬에 저장된 내용 재생용 라이센스를 로드하려고 시도합니다.

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.LOCAL_ONLY)
```

로드가 완료되면 DRMManager 객체에서 DRMStatusEvent.DRM_Status를 전달합니다.

6 DRMVoucher 객체가 null이 아닌 경우 해당 바우처는 유효합니다. 13단계로 건너뜁니다.

7 DRMVoucher 객체가 null인 경우 이 내용에 대한 정책에서 요구하는 인증 메서드를 확인합니다. DRMContentData.authenticationMethod 속성을 사용합니다.

8 인증 메서드가 ANONYMOUS인 경우 13단계로 이동합니다.

9 인증 메서드가 USERNAME_AND_PASSWORD인 경우 응용 프로그램에서 사용자가 자격 증명을 입력할 수 있도록 하는 메커니즘을 제공해야 합니다. 사용자를 인증하도록 이러한 자격 증명을 라이센스 서버로 전달합니다.

```
DRMManager.authenticate(metadata.serverURL, metadata.domain, username, password)
```

인증이 실패하는 경우 DRMAuthenticationErrorEvent, 인증이 성공하는 경우 DRMAuthenticationCompleteEvent를 DRMManager에서 전달합니다. 이러한 이벤트에 대한 리스너를 생성합니다.

10 인증 방법이 UNKNOWN인 경우 사용자 정의 인증 방법을 사용해야 합니다. 이 경우 ActionScript 3.0 API를 사용하지 않고 오프라인 방식으로 수행되는 인증을 내용 공급자가 준비해 둡니다. 사용자 정의 인증 절차에서는 DRMManager.setAuthenticationToken() 메서드에 전달할 수 있는 인증 토큰을 생성해야 합니다.

11 인증이 실패하면 응용 프로그램에서 9단계로 돌아가야 합니다. 응용 프로그램에 반복되는 인증 실패를 처리 및 제한하기 위한 메커니즘이 있어야 합니다. 예를 들어 세 번의 시도 후에 사용자에게 인증이 실패하여 내용을 재생할 수 없다는 메시지를 표시합니다.

12 사용자에게 자격 증명을 입력하라는 메시지를 표시하지 않고 저장된 토큰을 사용하려면 DRMManager.setAuthenticationToken() 메서드로 토큰을 설정합니다. 그러면 라이센스 서버에서 라이센스를 다운로드하여 8단계와 같이 내용을 재생할 수 있습니다.

13 (선택 사항) 인증이 성공하면 메모리에 캐시된 바이트 배열인 인증 토큰을 캡처할 수 있습니다. 이 토큰을 가져오는 데는 DRMAuthenticationCompleteEvent.token 속성을 사용합니다. 인증 토큰을 저장하여 사용하면 사용자가 이 내용에 대한 자격 증명을 매번 입력할 필요가 없습니다. 라이센스 서버에서 인증 토큰의 유효 기간을 확인합니다.

14 인증이 성공하면 라이센스 서버에서 라이센스를 다운로드합니다.

```
DRMManager.loadvoucher(drmContentData, LoadVoucherSetting.FORCE_REFRESH)
```

로드가 완료되면 DRMManager 객체에서 DRMStatusEvent.DRM_STATUS를 전달합니다. 이 이벤트를 수신하고 전달된 후에 내용을 재생할 수 있습니다.

15 NetStream 객체를 만든 다음 해당 play() 메서드를 호출하여 비디오를 재생합니다.

```
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 및 세션 객체

DRMContentData는 작성된 경우 Flash Player DRM 모듈을 참조하는 세션 객체로 사용됩니다. 이 DRMContentData를 수신하는 모든 DRMManager API는 해당 특정 DRM 모듈을 사용합니다. 그러나 DRMContentData를 사용하지 않는 두 개의 DRMManager API가 있습니다. 이러한 규칙은 다음과 같습니다.

1 authenticate()

2 setAuthenticationToken()

연결된 DRMContentData가 없으므로 이러한 DRMManager API를 호출하면 디스크의 최신 DRM 모듈이 사용됩니다. 이렇게 되면 응용 프로그램의 DRM 작업 과정 도중에 DRM 모듈의 업데이트가 발생할 경우 문제가 될 수 있습니다. 다음과 같은 시나리오를 생각해 보십시오.

1 응용 프로그램은 **AdobeCP1**을 DRM 모듈로 사용하는 DRMContentData 객체 contentData1을 만듭니다.

2 응용 프로그램은 DRMManager.authenticate(contentData1.serverURL,...) 메서드를 호출합니다.

3 응용 프로그램은 DRMManager.loadVoucher(contentData1, ...) 메서드를 호출합니다.

응용 프로그램이 2단계에 도달하기 전에 DRM 모듈에 업데이트가 발생할 경우 DRMManager.authenticate() 메서드는 **AdobeCP2**를 DRM 모듈로 사용하여 인증을 끝냅니다. 3단계의 loadVoucher() 메서드는 여전히 **AdobeCP1**을 DRM 모듈로 사용하므로 실패합니다. DRM 모듈 업데이트를 호출하는 다른 응용 프로그램으로 인해 업데이트가 발생했을 수 있습니다. 응용 프로그램 시작 시에 DRM 모듈 업데이트를 호출하면 이러한 시나리오를 방지할 수 있습니다.

DRM 관련 이벤트

응용 프로그램이 보호된 내용을 재생하려고 시도하면 런타임에서 다음과 같은 여러 이벤트를 전달합니다.

- DRMManager에 의해 전달된 DRMDeviceGroupErrorEvent(AIR에만 해당)
- NetStream에서 전달하는 DRMAuthenticateEvent(AIR에만 해당)
- DRMManager에서 전달하는 DRMAuthenticationCompleteEvent
- DRMManager에서 전달하는 DRMAuthenticationErrorEvent
- NetStream 및 DRMManager에서 전달하는 DRMErrorEvent
- NetStream 및 DRMManager에서 전달하는 DRMStatusEvent
- StatusEvent
- NetStatusEvent. 자세한 내용은 495페이지의 "업데이트 이벤트 수신"을 참조하십시오.

Adobe Access로 보호된 내용을 지원하려면 DRM 이벤트를 처리하는 이벤트 리스너를 추가합니다.

오프라인 재생을 위해 바우처 미리 로드

Adobe AIR 1.5 이상

Adobe Access로 보호된 내용을 재생하는 데 필요한 바우처(라이센스)를 미리 로드할 수 있습니다. 미리 로드된 바우처를 사용하면 사용자가 인터넷에 연결되어 있는지 여부에 관계없이 내용을 볼 수 있습니다. 이때 미리 로드 프로세스 자체에도 인터넷 연결이 필요합니다. NetStream 클래스 preloadEmbeddedMetadata() 메서드 및 DRMManager 클래스를 사용하여 바우처를 미리 로드할 수 있습니다. AIR 2.0 이상에서는 DRMContentData 객체를 사용하여 바우처를 직접 미리 로드할 수 있습니다. 내용과 독립적으로 DRMContentData 객체를 업데이트할 수 있기 때문에 이 방법을 사용하는 것이 좋습니다. preloadEmbeddedData() 메서드는 내용에서 DRMContentData를 가져옵니다.

DRMContentData 사용

Adobe AIR 2.0 이상

다음 단계에서는 DRMContentData 객체를 사용하여 보호된 미디어 파일에 대한 바우처를 미리 로드하는 작업 과정에 대해 설명합니다.

1. 패키지화된 내용에 대한 이진 메타데이터를 가져옵니다. Adobe Access Java Reference Packager를 사용하는 경우 이 메타데이터 파일이 **.metadata** 확장자를 사용하여 자동으로 생성됩니다. URLLoader 클래스를 사용하여 예를 들어 이 메타데이터를 다운로드할 수 있습니다.
2. DRMContentData 객체를 만들어 메타데이터를 생성자 함수에 전달합니다.

```
var drmData:DRMContentData = new DRMContentData( metadata );
```

3. 나머지 작업 과정은 479페이지의 "보호된 내용 작업 과정 이해"에서 설명한 작업 단계와 동일합니다.

preloadEmbeddedMetadata() 사용

Adobe AIR 1.5 이상

다음 단계에서는 preloadEmbeddedMetadata()를 사용하여 DRM으로 보호된 미디어 파일에 대한 바우처를 미리 로드하는 작업 과정에 대해 설명합니다.

1. 미디어 파일을 다운로드하고 저장합니다. DRM 메타데이터는 로컬로 저장된 파일에서만 미리 로드할 수 있습니다.
2. NetConnection 및 NetStream 객체를 만들어 NetStream 클라이언트 객체의 onDRMContentData() 및 onPlayStatus() 콜백 함수에 대한 구현을 제공합니다.
3. NetStreamPlayOptions 객체를 만들고 stream 속성을 로컬 미디어 파일의 URL로 설정합니다.
4. NetStream preloadEmbeddedMetadata()를 호출하여 파싱할 미디어 파일을 식별하는 NetStreamPlayOptions 객체를 전달합니다.
5. 미디어 파일에 DRM 메타데이터가 포함되어 있는 경우에는 onDRMContentData() 콜백 함수가 호출됩니다. 메타데이터는 이 함수에 DRMContentData 객체로 전달됩니다.
6. DRMContentData 객체를 사용하여 DRMManager loadVoucher() 메서드를 통해 바우처를 가져옵니다.

   DRMContentData 객체의 authenticationMethod 속성 값이 flash.net.drm.AuthenticationMethod.USERNAME_AND_PASSWORD인 경우에는 바우처를 로드하기 전에 미디어 권한 서버에서 사용자를 인증해야 합니다. DRMContentData 객체의 serverURL 및 domain 속성은 사용자의 자격 증명과 함께 DRMManager authenticate() 메서드로 전달할 수 있습니다.
7. 파일 파싱이 완료되면 onPlayStatus() 콜백 함수가 호출됩니다. onDRMContentData() 함수가 호출되지 않은 경우에는 파일에 바우처를 가져오는 데 필요한 메타데이터가 포함되지 않습니다. 이 호출이 누락되었다는 것은 Adobe Access가 이 파일을 보호하지 않는다는 것을 의미할 수도 있습니다.

AIR에 대한 다음 코드 예제에서는 로컬 미디어 파일에 대한 바우처를 미리 로드하는 방법을 보여 줍니다.

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

NetStream 클래스의 DRM 관련 멤버 및 이벤트

Flash Player 10.1 이상, Adobe AIR 1.0 이상

NetStream 클래스는 Flash Player 또는 AIR 응용 프로그램과 Flash Media Server 또는 로컬 파일 시스템 간의 단방향 스트리밍 연결을 제공합니다. NetStream 클래스는 점진적 다운로드도 지원합니다. NetStream 객체는 NetConnection 객체 안에 있는 채널입니다. NetStream 클래스는 다음 네 가지 DRM 관련 이벤트를 전달합니다.

이벤트	설명
drmAuthenticate (AIR에만 해당)	DRMAuthenticateEvent 클래스로 정의되며 이 이벤트는 NetStream 객체가 재생 전에 인증을 위해 사용자 자격 증명이 필요한 보호된 내용을 재생하려고 할 때 전달됩니다. 이 이벤트의 속성에는 사용자 자격 증명을 가져오고 설정할 때 사용할 수 있는 header, usernamePrompt, passwordPrompt 및 urlPrompt 속성이 포함됩니다. 이 이벤트는 NetStream 객체가 유효한 사용자 인증서를 가져올 때까지 반복해서 발생합니다.
drmError	DRMErrorEvent 클래스로 정의되며 보호된 내용을 재생하려고 시도하는 NetStream 객체에 DRM 관련 오류가 발생할 때 전달됩니다. 예를 들어 사용자 인증이 실패하면 DRM 오류 이벤트 객체가 전달됩니다. 이 오류는 사용자가 해당 내용을 볼 수 있는 권한을 구입하지 않았기 때문에 발생할 수 있습니다. 또한 내용 공급자가 보기 응용 프로그램을 지원하지 않기 때문에 발생할 수도 있습니다.
drmStatus	DRMStatusEvent 클래스로 정의되며 보호된 내용이 재생을 시작할 때 전달됩니다(사용자가 인증되었고 해당 내용을 재생하도록 권한이 부여된 경우). DRMStatusEvent 객체에는 바우처와 관련된 정보가 포함되어 있습니다. 예를 들어 내용을 오프라인으로 사용할 수 있게 만들 수 있는지 여부 또는 바우처가 만료되어 내용을 더 이상 볼 수 없게 되는 시점과 관련된 정보가 들어 있습니다.
status	events.StatusEvent에 정의되며 응용 프로그램이 NetStream.play() 메서드를 호출하여 보호된 내용을 재생하려고 시도할 때만 전달됩니다. 상태 코드 속성의 값은 "DRM.encryptedFLV"입니다.

NetStream 클래스에는 AIR에서만 사용되는 다음과 같은 DRM 관련 메서드가 포함되어 있습니다.

메서드	설명
resetDRMVouchers()	로컬로 캐시된 DRM(디지털 권한 관리) 바우처 데이터를 모두 삭제합니다. 사용자가 암호화된 내용에 액세스할 수 있으려면 응용 프로그램에서 바우처를 다시 다운로드해야 합니다. 예를 들어 다음 코드는 캐시에서 모든 바우처를 제거합니다. NetStream.resetDRMVouchers();
setDRMAuthenticationCredentials()	인증을 위해 인증 자격 증명 세트, 즉 사용자 이름, 암호 및 인증 유형을 NetStream 객체에 전달합니다. 유효한 인증 유형은 "drm" 및 "proxy" 입니다. "drm" 인증 유형을 사용하면 제공된 자격 증명이 Adobe Access에 대해 인증됩니다. "proxy" 인증 유형을 사용하면 자격 증명이 프록시 서버에 대해 인증되며 프록시 서버에 필요한 자격 증명과 일치해야 합니다. 예를 들어 기업에서 사용자가 먼저 프록시 서버에 대해 응용 프로그램을 인증해야만 인터넷에 액세스할 수 있도록 할 수 있습니다. 프록시 옵션은 이 유형의 인증을 허용합니다. 익명 인증을 사용하는 경우가 아니라면 프록시 인증 후에도 Adobe Access에 대해 인증해야 사용자가 바우처를 확보하여 내용을 재생할 수 있습니다. “drm" 옵션과 함께 setDRMAuthenticationCredentials()를 두 번 사용하여 Adobe Access에 대해 인증을 수행할 수 있습니다.
preloadEmbeddedMetadata()	포함된 메타데이터에 대한 로컬 미디어 파일을 파싱합니다. DRM 관련 메타데이터가 있을 경우 AIR에서는 onDRMContentData() 콜백 함수를 호출합니다.

또한 AIR에서 NetStream 객체는 preloadEmbeddedMetaData() 메서드에 대한 호출의 결과로 onDRMContentData() 및 onPlayStatus() 콜백 함수를 호출합니다. 미디어 파일에서 DRM 메타데이터가 발견되면 onDRMContentData() 함수가 호출되며 파일이 파싱되면 onPlayStatus() 함수가 호출됩니다. NetStream 인스턴스에 할당된 client 객체에는 onDRMContentData() 및 onPlayStatus() 함수를 정의해야 합니다. 같은 NetStream 객체를 사용하여 바우처를 미리 로드하고 내용을 재생하는 경우 preloadEmbeddedMetaData()에서 생성되는 onPlayStatus() 호출을 기다렸다가 재생을 시작해야 합니다.

AIR의 경우 다음 코드에서 사용자를 인증하기 위해 사용자 이름("administrator"), 암호("password") 및 "drm" 인증 유형을 설정합니다. setDRMAuthenticationCredentials() 메서드는 내용 공급자가 알고 있고 승인하는 자격 증명과 일치하는 자격 증명을 제공해야 합니다. 이러한 자격 증명은 사용자가 내용을 볼 수 있도록 허용하는 사용자 자격 증명과 동일합니다. 비디오를 재생하고 비디오 스트림과 성공적으로 연결되었는지 확인하는 코드는 여기에 포함되어 있지 않습니다.

```
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");
}
```

DRMStatusEvent 클래스 사용

Flash Player 10.1, Adobe AIR 1.0 이상

DRMStatusEvent 객체는 Adobe Access로 보호된 내용의 재생이 성공적으로 시작될 때 NetStream 객체에 의해 전달됩니다. 성공이란 라이센스가 확인되고 사용자가 인증되며 내용을 볼 수 있는 권한이 있음을 의미합니다. DRMStatusEvent는 액세스가 허용된 익명 사용자에 대해서도 전달됩니다. 인증이 필요하지 않은 익명 사용자가 내용에 액세스하여 재생할 수 있는지 여부를 바우처를 통해 확인합니다. 익명 사용자는 다양한 이유로 액세스가 거부될 수 있습니다. 예를 들어 라이센스가 만료되어 익명 사용자가 내용에 액세스하지 못할 수 있습니다.

DRMStatusEvent 객체에는 라이센스와 관련된 정보가 포함되어 있습니다. 예를 들어 라이센스를 오프라인으로 사용할 수 있게 만들 수 있는지 여부 또는 바우처가 만료되어 내용을 더 이상 볼 수 없게 되는 시기와 관련된 정보가 들어 있습니다. 응용 프로그램에서는 이 데이터를 사용하여 사용자의 정책 상태와 권한을 전달합니다.

DRMStatusEvent 속성

Flash Player 10.1, Adobe AIR 1.0 이상

DRMStatusEvent 클래스에는 다음과 같은 속성이 포함됩니다. 일부 속성은 AIR 1.0 이상 버전에서 사용할 수 있게 되었습니다. 전체 버전 정보는 **ActionScript 3.0 참조 설명서**를 참조하십시오.

Flash Player 10.1에서 지원되지 않는 속성의 경우 DRMVoucher 클래스에서 Flash Player와 유사한 속성을 제공합니다.

속성	설명
contentData	내용에 포함된 DRM 메타데이터가 들어 있는 DRMContentData 객체입니다.
detail (AIR에만 해당)	상태 이벤트의 컨텍스트를 설명하는 문자열입니다. DRM 1.0에서는 DRM.voucherObtained만 유효한 값입니다.
isAnonymous (AIR에만 해당)	사용자가 인증 자격 증명을 제공하지 않고도 Adobe Access로 보호된 내용을 사용할 수 있는지 여부(true 또는 false)를 나타냅니다. 값이 false이면 사용자는 내용 공급자가 알고 있고 예상하는 것과 일치하는 사용자 이름 및 암호를 제공해야 합니다.
isAvailableOffline(AIR에만 해당)	Adobe Access로 보호된 내용을 오프라인으로 사용할 수 있는지 여부(true 또는 false)를 나타냅니다. 디지털로 보호된 내용을 오프라인으로 사용하려면 해당 바우처를 사용자의 로컬 시스템에 캐싱해야 합니다.
isLocal	내용을 재생하는 데 필요한 바우처가 로컬로 캐시되어 있는지 여부를 나타냅니다.
offlineLeasePeriod(AIR에만 해당)	내용을 오프라인으로 볼 수 있는 나머지 일 수입니다.
policies (AIR에만 해당)	사용자 정의 DRM 속성을 포함할 수 있는 사용자 정의 객체입니다.
voucher	DRMVoucher입니다.
voucherEndDate (AIR에만 해당)	바우처가 만료되어 내용을 더 이상 볼 수 없는 절대 날짜입니다.

DRMStatusEvent 핸들러 만들기

Flash Player 10.1, Adobe AIR 1.0 이상

다음 예제에서는 이벤트를 발생시킨 NetStream 객체에 대한 DRM 내용 상태 정보를 출력하는 이벤트 핸들러를 만듭니다. 보호된 내용을 가리키는 NetStream 객체에 다음 이벤트 핸들러를 추가합니다.

```
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
function drmStatusEventHandler(event:DRMStatusEvent):void
{
    trace(event);
}
```

DRMAuthenticateEvent 클래스 사용

Adobe AIR 1.0 이상

DRMAuthenticateEvent 객체는 재생 전 NetStream 객체가 인증을 위해 사용자 자격 증명을 요구하는 보호된 내용을 재생하려고 시도할 때 전달됩니다.

DRMAuthenticateEvent 핸들러는 필수 자격 증명(사용자 이름, 암호 및 유형)을 수집하고 유효성 검증을 위해 NetStream.setDRMAuthenticationCredentials() 메서드에 해당 값을 전달합니다. 각 AIR 응용 프로그램은 사용자 자격 증명을 확보하기 위한 몇 가지 메커니즘을 제공해야 합니다. 예를 들어 응용 프로그램에서 사용자 이름 값 및 암호 값을 입력할 수 있는 간단한 사용자 인터페이스를 제공할 수 있을 것입니다. 또한 반복적인 인증 시도를 처리 및 제한하기 위한 메커니즘도 제공해야 합니다.

DRMAuthenticateEvent 속성

Adobe AIR 1.0 이상

DRMAuthenticateEvent 클래스에는 다음과 같은 속성이 포함됩니다.

속성	설명
authenticationType	제공된 자격 증명을 Adobe Access에 대해 인증하는지("drm") 프록시 서버에 대해 인증하는지("proxy")를 나타냅니다. 예를 들어 "proxy" 옵션을 사용하면 사용자가 인터넷에 액세스하기 전에 필요한 경우 응용 프로그램에서 프록시 서버에 대해 인증할 수 있습니다. 익명 인증을 사용하는 경우가 아니라면 프록시 인증 후에도 Adobe Access에 대해 인증해야 사용자가 바우처를 확보하여 내용을 재생할 수 있습니다. “drm" 옵션과 함께 setDRMAuthenticationcredentials()를 두 번 사용하여 Adobe Access에 대해 인증을 수행할 수 있습니다.
header	서버에서 제공한 암호화된 내용 파일 헤더입니다. 여기에는 암호화된 내용의 컨텍스트 정보가 포함됩니다. 이 헤더 문자열을 Flash 응용 프로그램에 전달하면 응용 프로그램에서 사용자 이름-암호 대화 상자를 생성할 수 있게 됩니다. 이 헤더 문자열을 대화 상자의 지침으로 사용할 수 있습니다. 예를 들어 "사용자 이름과 암호를 입력하십시오."를 헤더로 지정할 수 있습니다.
netstream	이 이벤트를 시작한 NetStream 객체입니다.
passwordPrompt	서버에서 제공한 암호 자격 증명에 대한 프롬프트입니다. 문자열에는 필요한 암호 유형에 대한 명령이 포함될 수 있습니다.
urlPrompt	서버에서 제공한 URL 문자열에 대한 프롬프트입니다. 문자열은 사용자 이름 및 암호가 전송되는 위치를 제공할 수 있습니다.
usernamePrompt	서버에서 제공한 사용자 이름 자격 증명에 대한 프롬프트입니다. 문자열에는 필요한 사용자 이름 유형에 대한 명령이 포함될 수 있습니다. 예를 들어 내용 공급자에 사용자 이름으로 전자 메일 주소가 필요할 수 있습니다.

앞에서 언급한 문자열은 FMRMS 서버에서만 제공되며, Adobe Access Server에서는 이러한 문자열을 사용하지 않습니다.

DRMAuthenticateEvent 핸들러 만들기

Adobe AIR 1.0 이상

다음 예제에서는 이벤트를 발생시킨 NetStream 객체에 하드 코딩된 인증 자격 증명 집합을 전달하는 이벤트 핸들러를 만듭니다. 비디오를 재생하고 비디오 스트림과 성공적으로 연결되었는지 확인하는 코드는 여기에 포함되어 있지 않습니다.

```
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");
}
```

사용자 자격 증명을 검색하기 위한 인터페이스 만들기

Adobe AIR 1.0 이상

보호된 내용을 보기 위해 사용자 인증이 필요한 경우 AIR 응용 프로그램은 일반적으로 사용자 인터페이스를 통해 사용자의 인증 자격 증명을 검색해야 합니다.

다음은 사용자 자격 증명을 검색하는 간단한 사용자 인터페이스에 대한 Flex 예제입니다. 이 인터페이스는 사용자 이름 및 암호 자격 증명에 대해 하나당 두 개의 TextInput 객체가 포함된 패널 객체로 구성됩니다. 이 패널에는 credentials() 메서드를 시작하는 버튼도 포함되어 있습니다.

```
<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>
```

credentials() 메서드는 사용자 이름 및 암호 값을 setDRMAuthenticationCredentials() 메서드에 전달하는 사용자 정의 메서드입니다. 값이 전달되면 credentials() 메서드는 TextInput 객체의 값을 재설정합니다.

```
<mx:Script>
    <![CDATA[
        public function credentials():void
        {
            videoStream.setDRMAuthenticationCredentials(uName, pWord, "drm");
            uName.text = "";
            pWord.text = "";
        }
    ]]>
</mx:Script>
```

이러한 유형의 단순한 인터페이스를 구현하는 한 가지 방법은 해당 패널을 새로운 상태의 일부로 포함하는 것입니다. 새로운 상태는 DRMAuthenticateEvent 객체가 발생할 때 기본 상태로부터 시작됩니다. 다음 예제에는 보호된 FLV 파일을 가리키는 소스 특성이 있는 VideoDisplay 객체가 포함되어 있습니다. 이 경우 credentials() 메서드가 응용 프로그램을 기본 상태로 반환하도록 수정됩니다. 이 메서드는 사용자 자격 증명을 전달하고 TextInput 객체 값을 재설정한 후 이러한 작업을 수행합니다.

```
<?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>
```

DRMErrorEvent 클래스 사용

Flash Player 10.1 이상, Adobe AIR 1.0 이상

보호된 내용을 재생하려고 시도하는 NetStream 객체에 DRM 관련 오류가 발생할 때 Adobe Flash Player 및 Adobe AIR은 DRMErrorEvent 객체를 전달합니다. AIR 응용 프로그램에서 사용자 자격 증명이 잘못된 경우 사용자가 올바른 자격 증명을 입력할 때까지 DRMAuthenticateEvent 객체가 반복적으로 전달합니다. 또는 AIR 응용 프로그램이 추가 시도를 거부합니다. 응용 프로그램은 다른 모든 DRM 오류 이벤트를 수신하여 DRM 관련 오류를 감지, 식별 및 처리해야 합니다.

유효한 사용자 자격 증명이 있더라도 해당 내용의 바우처 조건에 따라 사용자가 암호화된 내용을 볼 수 없을 수도 있습니다. 예를 들어 사용자가 권한 없는 응용 프로그램의 내용을 보려고 할 경우 액세스가 거부될 수 있습니다. 권한 없는 응용 프로그램이란 암호화된 내용의 제작자가 유효성 검사를 하지 않은 응용 프로그램입니다. 이 경우 DRMErrorEvent 객체가 전달됩니다.

내용이 손상되었거나 응용 프로그램의 버전이 바우처에 지정된 버전과 일치하지 않는 경우에도 오류 이벤트가 발생할 수 있습니다. 응용 프로그램에서는 오류를 처리하기 위한 적절한 메커니즘을 제공해야 합니다.

DRMErrorEvent 속성

Flash Player 10.1 이상, Adobe AIR 1.0 이상

전체 오류 목록은 ActionScript 3.0 참조 설명서의 런타임 오류 코드를 참조하십시오. DRM 관련 오류는 오류 코드 3300부터 시작합니다.

DRMErrorEvent 핸들러 만들기

Flash Player 10.1 이상, Adobe AIR 1.0 이상

다음 예제에서는 이벤트를 발생시킨 NetStream 객체에 대한 이벤트 핸들러를 만듭니다. 이 핸들러는 보호된 내용을 재생하려고 할 때 NetStream에 오류가 발생하는 경우 호출됩니다. 일반적으로 응용 프로그램에서 오류가 발생하는 경우 필요한 수의 정리 작업을 수행합니다. 그런 다음 사용자에게 오류에 대해 알리고 문제를 해결하기 위한 옵션을 제공합니다.

```
private function drmErrorEventHandler(event:DRMErrorEvent):void
{
    trace(event.toString());
}
```

DRMManager 클래스 사용

Flash Player 10.1 이상, Adobe AIR 1.5 이상

응용 프로그램에서 DRMManager 클래스를 사용하여 바우처 및 미디어 권한 서버 세션을 관리할 수 있습니다.

바우처 관리(AIR에만 해당)

사용자가 보호된 내용을 재생할 때마다 런타임에서 내용을 보는 데 필요한 라이센스를 가져오고 캐시합니다. 응용 프로그램에서 파일을 로컬로 저장하고 라이센스가 오프라인 재생을 허용하는 경우 사용자가 AIR 응용 프로그램의 내용을 볼 수 있습니다. 미디어 권한 서버에 대한 연결을 사용할 수 없는 경우에도 이러한 로컬 오프라인 재생이 수행됩니다. DRMManager 및 NetStream preloadEmbeddedMetadata() 메서드를 사용하면 바우처를 미리 캐시할 수 있습니다. 응용 프로그램에서 내용을 보는 데 필요한 라이센스를 가져올 필요가 없습니다. 예를 들어 응용 프로그램에서 미디어 파일을 다운로드한 다음 사용자가 여전히 온라인 상태에 있을 때 바우처를 가져올 수 있습니다.

바우처를 미리 로드하려면 NetStream preloadEmbeddedMetadata() 메서드를 사용하여 DRMContentData 객체를 가져옵니다. DRMContentData 객체에는 라이센스를 제공하며 사용자 인증이 필요한지 여부를 설명할 수 있는 미디어 권한 서버의 URL 및 도메인이 포함되어 있습니다. 이 정보를 사용하면 DRMManager loadVoucher() 메서드를 호출하여 바우처를 가져오고 캐시할 수 있습니다 바우처를 미리 로드하는 작업 과정은 483페이지의 "오프라인 재생을 위해 바우처 미리 로드"에서 자세히 설명합니다.

세션 관리

DRMManager를 사용하여 사용자를 미디어 권한 서버에 대해 인증하고 영구 세션을 관리할 수도 있습니다.

미디어 권한 서버를 사용하여 세션을 설정하려면 DRMManager authenticate() 메서드를 호출합니다. 인증이 성공적으로 완료되면 DRMManager는 DRMAuthenticationCompleteEvent 객체를 전달합니다. 이 객체에는 세션 토큰이 포함되어 있습니다. 사용자가 자신의 계정 자격 증명을 제공하지 않아도 되도록 이 토큰을 저장하여 이후 세션을 설정할 수 있습니다. 토큰을 setAuthenticationToken() 메서드에 전달하여 인증된 새 세션을 설정합니다. 토큰을 생성한 서버의 설정에 따라 토큰 만료 및 기타 특성이 결정됩니다. 이후 AIR 업데이트에서 토큰 데이터 구조가 변경될 수 있으므로 AIR 응용 프로그램 코드는 토큰 데이터 구조를 해석하지 않아야 합니다.

인증 토큰은 다른 컴퓨터로 전송할 수 있습니다. 토큰을 보호하려면 토큰을 AIR 암호화된 로컬 저장소에 저장합니다. 자세한 내용은 647페이지의 "암호화된 로컬 저장소"를 참조하십시오.

DRMStatus 이벤트

Flash Player 10.1 이상, Adobe AIR 1.5 이상

DRMManager는 loadVoucher() 메서드에 대한 호출이 성공적으로 완료되면 DRMStatusEvent 객체를 전달합니다.

바우처를 가져오면 이벤트 객체의 detail 속성(AIR에만 해당)에 "DRM.voucherObtained" 값이 포함되고 voucher 속성에는 DRMVoucher 객체가 포함됩니다.

바우처를 가져오지 못해도 detail 속성(AIR에만 해당)에 여전히 "DRM.voucherObtained" 값이 포함되지만 voucher 속성은 null이 됩니다. 예를 들어 **localOnly**의 LoadVoucherSetting을 사용하지만 로컬로 캐시된 바우처가 없는 경우 바우처를 가져오지 못할 수 있습니다.

인증 또는 통신 오류 등으로 인해 loadVoucher() 호출이 성공적으로 완료되지 않으면 DRMManager가 DRMErrorEvent 또는 DRMAuthenticationErrorEvent 객체를 대신 전달합니다.

DRMAuthenticationComplete 이벤트

Flash Player 10.1 이상, Adobe AIR 1.5 이상

DRMManager는 authenticate() 메서드 호출을 통해 사용자가 성공적으로 인증되면 DRMAuthenticationCompleteEvent 객체를 전달합니다.

DRMAuthenticationCompleteEvent 객체에는 전체 응용 프로그램 세션에서 사용자 인증을 유지하는 데 사용할 수 있는 재사용 가능 토큰이 포함되어 있습니다. 이 토큰을 DRMManager setAuthenticationToken() 메서드에 전달하면 세션이 다시 설정됩니다. 토큰 작성자가 만료 등의 토큰 특성을 설정합니다. Adobe에서는 토큰 특성을 확인하기 위한 API를 제공하지 않습니다.

DRMAuthenticationError 이벤트

Flash Player 10.1 이상, Adobe AIR 1.5 이상

DRMManager는 authenticate() 또는 setAuthenticationToken() 메서드에 대한 호출을 통해 사용자를 성공적으로 인증할 수 없으면 DRMAuthenticationErrorEvent 객체를 전달합니다.

DRMContentData 클래스 사용

Flash Player 10.1 이상, Adobe AIR 1.5 이상

DRMContentData 객체에는 Adobe Access로 보호된 내용의 메타데이터 속성이 포함되어 있습니다. DRMContentData 속성에는 내용을 보기 위한 라이센스 바우처를 가져오는 데 필요한 정보가 포함되어 있습니다. 481페이지의 "상세한 API 작업 과정"에서 설명한 대로 DRMContentData 클래스를 사용하여 내용과 연관된 메타데이터 파일을 가져올 수 있습니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서의 DRMContentData 클래스를 참조하십시오.

Adobe Access를 지원하기 위한 Flash Player 업데이트

Flash Player 10.1 이상

중요: Flash Player 11.5 이상에는 Adobe Access 모듈이 통합되어 있으므로 SystemUpdater.update(SystemUpdaterType.DRM))를 호출하는 업데이트 단계가 불필요합니다. 여기에는 다음 브라우저와 플랫폼이 포함됩니다.

- Flash Player 11.5 ActiveX 컨트롤(Windows 8의 Internet Explorer를 제외한 모든 플랫폼)
- Flash Player 11.5 플러그인(모든 브라우저)
- Adobe AIR(데스크톱 및 모바일)

따라서 다음과 같은 경우에는 업데이트 단계를 **계속 실행해야** 합니다.

- Windows 8의 Internet Explorer
- Flash Player 11.4 이하(Google Chrome 22 이상(모든 플랫폼) 또는 21 이상(Windows)은 제외)

참고: Flash Player 11.5 이상이 설치된 시스템에서는 여전히 SystemUpdater.update(SystemUpdaterType.DRM)를 안전하게 호출할 수 있지만 아무것도 다운로드되지 않습니다.

Adobe Access를 지원하려면 Flash Player에 Adobe Access 모듈이 필요합니다. Flash Player에서 보호된 내용을 재생하려고 시도하면 런타임에서 이 모듈 또는 Flash Player의 새 버전을 다운로드해야 하는지 여부를 나타냅니다. 따라서 SWF 개발자는 원하는 경우 Flash Player를 업데이트하지 않도록 선택할 수 있습니다.

대부분의 경우 보호된 내용을 재생하기 위해 SWF 개발자는 필요한 Adobe Access 모듈 또는 플레이어를 업데이트합니다. 업데이트하려는 경우 SystemUpdater API를 사용하여 최신 버전의 Adobe Access 모듈 또는 Flash Player를 가져올 수 있습니다.

SystemUpdater API를 사용하면 한 번에 하나의 업데이트만 가능합니다. 오류 코드 2202는 현재 런타임 인스턴스 또는 다른 인스턴스에서 업데이트가 이미 이루어지고 있음을 나타냅니다. 예를 들어 Internet Explorer에서 Flash Player 인스턴스의 업데이트가 이루어지고 있는 경우 Firefox에서 Flash Player 인스턴스를 업데이트할 수 없습니다.

SystemUpdater API는 데스크톱 플랫폼에 대해서만 지원됩니다.

참고: Flash Player 10.1 이전 버전의 경우 이전 버전에서 지원되는 업데이트 메커니즘을 사용해야 합니다 (www.adobe.com/kr/ 또는 ExpressInstall에서 수동 다운로드 및 설치). 또한 AIR 설치 프로그램은 Adobe Access를 위해 필요한 업데이트를 처리하며 SystemUpdater API를 지원하지 않습니다.

업데이트 이벤트 수신

Flash Player 10.1 이상

Adobe Access 모듈의 업데이트가 필요한 경우 NetStream 객체에서 코드 값 DRM.UpdateNeeded와 함께 NetStatusEvent를 전달합니다. 이 값은 NetStream 객체에서 현재 설치된 Adobe Access 모듈로 보호된 스트림을 재생할 수 없음을 나타냅니다. 이 이벤트를 수신한 후 다음 코드를 호출합니다.

```
SystemUpdater.update(flash.system.SystemUpdaterType.DRM)
```

이 코드는 Flash Player에 설치된 Adobe Access 모듈을 업데이트합니다. 이 모듈 업데이트에 대한 사용자 동의는 필요하지 않습니다.

Adobe Access 모듈이 없으면 오류가 발생합니다. 자세한 내용은 481페이지의 "상세한 API 작업 과정"의 3단계를 참조하십시오.

참고: Flash Player 10.1 이전 버전에서 암호화된 스트림에 대해 play()를 호출하면 코드 값 NetStream.Play.StreamNotFound와 함께 NetStatusEvent가 전달됩니다. 10.1 이전 버전에서는 해당 버전에 지원되는 업데이트 메커니즘을 사용해야 합니다(www.adobe.com/kr/ 또는 ExpressInstall에서 수동 다운로드 및 설치).

Flash Player 자체의 업데이트가 필요한 경우 SystemUpdater 객체에서 코드 값 DRM.UpdateNeededButIncompatible과 함께 StatusEvent를 전달합니다. Flash Player 업데이트에는 사용자 동의가 필요합니다. 사용자가 Flash Player 업데이트에 동의하고 업데이트를 시작하는 인터페이스를 응용 프로그램에 제공해야 합니다. StatusEvent 이벤트를 수신한 후 다음 코드를 호출합니다.

```
SystemUpdater.update(flash.system.SystemUpdaterType.SYSTEM);
```

이 코드는 Flash Player 업데이트를 시작합니다.

SystemUpdater 클래스의 추가 이벤트는 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에 설명되어 있습니다.

Flash Player 업데이트가 완료되면 사용자는 업데이트가 시작된 페이지로 리디렉션됩니다. 그런 다음 Adobe Access 모듈이 다운로드되고 스트림 재생을 시작할 수 있습니다.

오프라인 라이센스

Flash Player 11 이상, Adobe AIR 3.0 이상

storeVoucher 메서드를 사용해 바우처(라이센스)를 디스크 및 메모리에 저장하면 Adobe Access License Server에 연결하지 않고 오프라인 라이센스를 얻을 수 있습니다.

Flash Player 및 AIR에서 암호화된 비디오를 재생하려면 각각의 런타임에서 해당 비디오에 대한 DRM 바우처를 얻어야 합니다. DRM 바우처는 비디오의 암호 해독 키를 포함하며, 고객이 배포한 Adobe Access License Server를 통해 생성됩니다.

일반적으로 Flash Player/AIR 런타임은 비디오의 DRM 메타데이터(DRMContentData 클래스)에 지정된 Adobe Access License Server에 바우처 요청을 보내서 이 바우처를 얻습니다. Flash/AIR 응용 프로그램은 DRMManager.loadVoucher() 메서드를 호출하여 이 라이센스 요청을 트리거할 수 있습니다. 또는 디스크나 메모리의 내용에 대한 라이센스가 없는 경우 Flash Player/AIR 런타임은 암호화된 비디오 재생이 시작될 때 라이센스를 자동으로 요청합니다. 두 경우 모두 Adobe Access License Server와 통신하므로 Flash/AIR 응용 프로그램의 성능이 저하됩니다.

DRMManager.storeVoucher()를 통해 Flash/AIR 응용 프로그램은 오프라인으로 얻은 DRM 바우처를 Flash Player/AIR 런타임에 보낼 수 있습니다. 그런 다음 런타임은 라이센스 요청 프로세스를 건너뛰고 암호화된 비디오를 재생하기 위해 전달된 바우처를 사용할 수 있습니다. DRM 바우처를 오프라인으로 얻을 수 있으려면 여전히 Adobe Access License Server가 해당 바우처를 생성해야 합니다. 그러나 공개된 Adobe Access License Server 대신에 임의의 HTTP 서버에서 바우처를 호스팅할 수도 있습니다.

또한 여러 장치 간의 DRM 바우처 공유를 지원하기 위해 DRMManager.storeVoucher()가 사용됩니다. Adobe Access 3.0에서는 이 기능을 "도메인 지원"이라고 합니다. 현재 배포에서 이 기능을 지원할 경우 DRMManager.addToDeviceGroup() 메서드를 사용하여 여러 시스템을 장치 그룹에 등록할 수 있습니다. 지정된 내용에 대해 도메인에 바인딩된 유효한 바우처가 시스템에 있는 경우 AIR 응용 프로그램에서는 DRMVoucher.toByteArray() 메서드를 사용하여 직렬화된 DRM 바우터를 추출할 수 있고, 사용자는 자신의 다른 시스템에서 DRMManager.storeVoucher() 메서드를 사용하여 바우처를 가져올 수 있습니다.

장치 등록

DRM 바우처는 최종 사용자의 시스템에 바인딩됩니다. 따라서 Flash/AIR 응용 프로그램에서 직렬화된 올바른 DRM 바우처 객체를 참조하려면 응용 프로그램에 사용자의 시스템에 대한 고유한 ID가 필요합니다. 다음 시나리오에서는 장치 등록 프로세스를 보여 줍니다.

다음 작업을 수행했다고 가정합니다.

- Adobe Access Server SDK를 설정했습니다.
- 미리 생성된 라이센스를 얻기 위해 HTTP 서버를 설정했습니다.
- 보호된 내용을 보기 위해 Flash 응용 프로그램을 만들었습니다.

장치 등록 단계에는 다음 작업이 포함됩니다.

1. Flash 응용 프로그램은 임의로 생성된 ID를 만듭니다.
2. Flash 응용 프로그램에서는 DRMManager.authenticate() 메서드를 호출합니다. 응용 프로그램은 임의로 생성된 ID를 인증 요청에 포함해야 합니다. 예를 들어 사용자 이름 필드에 ID를 포함합니다.
3. 2단계에서 언급한 작업으로 인해 Adobe Access는 인증 요청을 고객의 서버로 보냅니다. 이 요청에는 장치 인증서가 포함되어 있습니다.
   a. 서버는 장치 인증서 및 생성된 ID를 요청에서 추출하여 저장합니다.
   b. 고객 하위 시스템은 이 장치 인증서에 대한 라이센스를 미리 생성하여 저장한 다음 생성된 ID와 연결하는 방식으로 액세스를 허용합니다.
4. 서버는 "성공" 메시지로 요청에 응답합니다.
5. Flash 응용 프로그램은 생성된 ID를 LSO(Local Shared Object)에 로컬로 저장합니다.

장치 등록 후에 Flash 응용 프로그램은 이전 체계에서 장치 ID를 사용했던 것과 동일한 방법으로 생성된 ID를 사용합니다.

1. Flash 응용 프로그램은 생성된 ID를 LSO에서 찾습니다.
2. 생성된 ID가 발견된 경우 Flash 응용 프로그램에서는 미리 생성된 라이센스를 다운로드하는 동안 해당 ID를 사용합니다. Flash 응용 프로그램은 DRMManager.storeVoucher() 메서드를 사용하여 Adobe Access 클라이언트로 라이센스를 보냅니다.
3. 생성된 ID가 발견되지 않는 경우 Flash 응용 프로그램에서는 장치 등록 절차를 진행합니다.

팩토리 재설정

장치 사용자가 팩토리 재설정 옵션을 호출할 경우 장치 인증서가 삭제됩니다. 보호된 내용의 재생을 계속하려면 Flash 응용 프로그램은 장치 등록 절차를 다시 진행해야 합니다. Flash 응용 프로그램이 미리 생성된 오래된 라이센스를 보낼 경우 이전 장치 ID에 맞게 라이센스가 암호화되었기 때문에 Adobe Access 클라이언트에서 라이센스를 거부합니다.

도메인 지원

Flash Player 11 이상, Adobe AIR 3.0 이상

내용 메타데이터에 도메인 등록이 필요하다고 지정되어 있는 경우 AIR 응용 프로그램은 API를 호출하여 장치 그룹에 가입할 수 있습니다. 이렇게 하면 도메인 서버에 보낼 도메인 등록 요청이 트리거됩니다. 라이센스가 장치 그룹에 발급되고 나면 라이센스를 내보내 장치 그룹에 가입한 다른 장치와 공유할 수 있습니다.

그런 다음 장치 그룹 정보는 바우처를 성공적으로 가져오고 사용하는 데 필요한 정보를 제공하는 DRMContentData의 VoucherAccessInfo 객체에 사용됩니다.

도메인 지원을 사용하여 암호화된 내용 재생

Adobe Access를 사용하여 암호화된 내용을 재생하려면 다음 단계를 수행합니다.

1. VoucherAccessInfo.deviceGroup을 사용하여 장치 그룹 등록이 필요한지 여부를 확인합니다.
2. 인증이 필요한 경우:
   a. DeviceGroupInfo.authenticationMethod 속성을 사용하여 인증이 필요한지 여부를 확인합니다.
   b. 인증이 필요한 경우 다음 단계 중 하나를 수행하여 사용자를 인증합니다.
      - 사용자의 사용자 이름과 암호를 얻습니다. DRMManager.authenticate(deviceGroup.serverURL, deviceGroup.domain, username, password)를 호출합니다.
      - 캐시/미리 생성된 인증 토큰을 얻고 DRMManager.setAuthenticationToken()을 호출합니다.
   c. DRMManager.addToDeviceGroup()을 호출합니다.
3. 다음 작업 중 하나를 수행하여 내용에 대한 바우처를 가져옵니다.
   a. DRMManager.loadVoucher() 메서드를 사용합니다.
   b. 동일한 장치 그룹에 등록된 다른 장치에서 바우처를 가져옵니다. DRMManager.storeVoucher() 메서드를 통해 바우처를 DRMManager에 제공합니다.
4. NetStream.play() 메서드를 사용하여 암호화된 내용을 재생합니다.

내용에 대한 라이센스를 내보내려면 임의의 장치가 Adobe Access License Server에서 라이센스를 가져온 후에 DRMVoucher.toByteArray() 메서드를 사용하여 라이센스의 원시 바이트를 제공할 수 있습니다. 일반적으로 내용 공급자는 장치 그룹의 장치 수를 제한합니다. 제한에 도달한 경우 현재 장치를 등록하기 전에 사용되지 않은 장치에서 DRMManager.removeFromDeviceGroup() 메서드를 호출해야 합니다.

라이센스 미리 보기

Flash 응용 프로그램은 라이센스 미리 보기 요청을 보낼 수 있습니다. 따라서 사용자의 시스템이 재생에 필요한 모든 조건을 실제로 충족하는 확인하기 위해 내용을 구입하도록 사용자에게 요청하기 전에 응용 프로그램에서 미리 보기 작업을 수행할 수 있습니다. 라이센스 미리 보기는 클라이언트가 내용을 미리 보는 것(구입 결정을 하기 전에 내용의 작은 일부 보기)과 달리 라이센스를 미리 볼 수 있다는 것(라이센스에 따라 허용되는 권한 확인)을 의미합니다. 각 시스템에 고유한 일부 매개 변수로는 사용 가능한 출력 및 보호 상태, 사용 가능한 런타임/DRM 버전, DRM 클라이언트 보안 수준 등이 있습니다. 라이센스 미리 보기 모드를 사용하면 런타임/DRM 클라이언트에서 라이센스 서버 비즈니스 논리를 테스트하고 사용자가 정확한 결정을 내릴 수 있도록 사용자

에게 정보를 다시 제공할 수 있습니다. 따라서 클라이언트는 유효한 라이센스가 무엇인지 볼 수 있지만 실제로 내용을 해독하기 위한 키를 받지는 않습니다. 라이센스 미리 보기에 대한 지원은 선택 사항이며 이 기능을 사용하는 사용자 정의 응용 프로그램을 구현하는 경우에만 필요합니다.

내용 전달

Flash Player는 네트워킹 레이어를 추출하고 보호된 내용을 Adobe Access 하위 시스템에 제공하기만 하므로 Adobe Access는 내용 전달 메커니즘에 종속되지 않습니다. 따라서 HTTP, HTTP 동적 스트리밍, RTMP 또는 RTMPE를 통해 내용을 전달할 수 있습니다.

그러나 Adobe Access가 내용을 해독하기 위한 라이센스를 얻으려면 보호된 내용의 메타데이터(일반적으로 '.metadata' 파일 형태)가 필요하므로 몇 가지 문제가 발생할 수도 있습니다. 특히 RTMP/RTMPE 프로토콜의 경우 FLV 및 F4V 데이터만 FMS(Flash Media Server)를 통해 클라이언트에 전달할 수 있습니다. 따라서 클라이언트는 다른 방법으로 메타데이터 BLOB을 가져와야 합니다. 이 문제를 해결하는 한 가지 방법은 HTTP 웹 서버에서 메타데이터를 호스팅하고, 클라이언트 비디오 플래이어를 구현하여 재생 중인 내용에 따라 적절한 메타데이터를 가져오는 것입니다.

```
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)는 자신만의 풍부한 미디어 경험을 만들기 위한 완벽한 유연성과 제어 기능을 제공하는 ActionScript 기반 프레임워크입니다. OSMF에 대한 자세한 내용은 OSMF 개발자 사이트를 참조하십시오.

보호된 내용을 재생하기 위한 작업 과정

1. MediaPlayer 인스턴스를 만듭니다.

   ```
   player = new MediaPlayer();
   ```

2. MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE 이벤트를 플레이어에 등록합니다. 내용이 DRM으로 보호된 경우 이 이벤트가 전달됩니다.

   ```
   player.addEventListener(MediaPlayerCapabilityChangeEvent.HAS_DRM_CHANGE, onDRMCapabilityChange);
   ```

3. 이벤트 핸들러에서 DRMTrait 인스턴스를 얻습니다. DRMTrait는 authenticate() 같은 DRM 관련 메서드를 호출하는 데 사용되는 인터페이스입니다. DRM 보호 내용을 로드할 경우 OSMF는 DRM 유효성 검사 작업을 수행하고 상태 이벤트를 전달합니다. DRMEvent.DRM_STATE_CHANGE 이벤트 핸들러를 DRMTrait에 추가합니다.

```
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 onDRMStateChange() 메서드에서 DRM 이벤트를 처리합니다.

```
        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;
    }
}
```

28장: AIR에서 PDF 내용 추가

Adobe AIR 1.0 이상

Adobe® AIR®에서 실행되는 응용 프로그램은 SWF 및 HTML 내용뿐만 아니라 PDF 내용도 렌더링할 수 있습니다. AIR 응용 프로그램은 HTMLLoader 클래스, WebKit 엔진 및 Adobe® Reader® 브라우저 플러그인을 사용하여 PDF 내용을 렌더링합니다. AIR 응용 프로그램에서 PDF 내용은 응용 프로그램의 전체 높이 및 폭에 맞게 확장하거나 인터페이스의 한 부분으로 확장할 수 있습니다. Adobe Reader 브라우저 플러그인은 AIR 응용 프로그램에서 PDF 파일의 표시를 제어합니다. Reader 도구 모음 인터페이스에서 위치, 앵커 및 표시 여부 등에 대한 컨트롤을 수정한 경우 이후 AIR 응용 프로그램 및 브라우저에서 PDF 파일을 볼 때에도 그러한 변경 사항이 지속됩니다.

중요: AIR에서 PDF 내용을 렌더링하려면 Adobe Reader 또는 Adobe® Acrobat® 버전 8.1 이상을 설치해야 합니다.

PDF 기능 검색

Adobe AIR 1.0 이상

Adobe Reader 또는 Adobe Acrobat 8.1 이상이 없으면 PDF 내용이 AIR 응용 프로그램에 표시되지 않습니다. 사용자가 PDF 내용을 렌더링할 수 있는지 여부를 검색하려면 먼저 HTMLLoader.pdfCapability 속성을 확인합니다. 이 속성은 HTMLPDFCapability 클래스의 다음 상수 중 하나로 설정됩니다.

상수	설명
HTMLPDFCapability.STATUS_OK	Adobe Reader의 최신 버전(8.1 이상)이 검색되었으므로 PDF 내용을 HTMLLoader 객체로 로드할 수 있습니다.
HTMLPDFCapability.ERROR_INSTALLED_READER_NOT_FOUND	검색된 Adobe Reader 버전이 없습니다. HTMLLoader 객체가 PDF 내용을 표시할 수 없습니다.
HTMLPDFCapability.ERROR_INSTALLED_READER_TOO_OLD	Adobe Reader가 검색되었지만 너무 오래된 버전입니다. HTMLLoader 객체가 PDF 내용을 표시할 수 없습니다.
HTMLPDFCapability.ERROR_PREFERRED_READER_TOO_OLD	Adobe Reader의 최신 버전(8.1 이상)이 검색되었지만 PDF 내용을 처리하도록 설정된 Adobe Reader의 버전이 Reader 8.1보다 이전 버전입니다. HTMLLoader 객체가 PDF 내용을 표시할 수 없습니다.

Windows의 경우 사용자의 시스템에서 Adobe Acrobat 또는 Adobe Reader 버전 7.x 이상이 실행 중이면 로드 중인 PDF를 지원하는 최신 버전이 설치되어 있어도 이 버전이 사용됩니다. 이 경우 pdfCapability 속성의 값이 HTMLPDFCapability.STATUS_OK이면 AIR 응용 프로그램이 PDF 내용을 로드하려고 할 때 이전 버전의 Acrobat 또는 Reader에 경고가 표시됩니다(AIR 응용 프로그램에서는 예외가 발생하지 않음). 최종 사용자에게 이러한 상황이 발생할 수 있는 경우 응용 프로그램을 실행하는 동안 Acrobat을 닫도록 알려 주는 것이 좋습니다. PDF 내용이 허용 가능한 시간 프레임 내에 로드되지 않는 경우 이러한 지침을 표시할 수도 있습니다.

Linux에서 AIR은 사용자가 내보낸 경로(acroread 명령이 포함된 경우)와 /opt/Adobe/Reader 디렉토리에서 Adobe Reader를 찾습니다.

다음 코드에서는 사용자가 AIR 응용 프로그램에서 PDF 내용을 표시할 수 있는지 여부를 검사합니다. 사용자가 PDF를 표시할 수 없으면 코드가 HTMLPDFCapability 오류 객체에 해당하는 오류 코드를 추적합니다.

```
if(HTMLLoader.pdfCapability == HTMLPDFCapability.STATUS_OK)
{
    trace("PDF content can be displayed");
}
else
{
    trace("PDF cannot be displayed. Error code:", HTMLLoader.pdfCapability);
}
```

PDF 내용 로드

Adobe AIR 1.0 이상

HTMLLoader 인스턴스를 만들고, 해당 크기를 설정하고, PDF 경로를 로드하여 AIR 응용 프로그램에 PDF를 추가할 수 있습니다.

다음 예제에서는 외부 사이트에서 PDF를 로드합니다. URLRequest를 사용 가능한 외부 PDF에 대한 경로로 바꾸십시오.

```
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);
```

app 및 app-storage와 같은 파일 URL 스킴과 AIR 고유의 URL 스킴에서 내용을 로드할 수도 있습니다. 예를 들어 다음 코드에서는 응용 프로그램 디렉토리의 PDF 하위 디렉토리에 있는 test.pdf 파일을 로드합니다.

app:/js_api_reference.pdf

AIR URL 스킴에 대한 자세한 내용은 741페이지의 “URI 스킴”을 참조하십시오.

PDF 내용 스크립팅

Adobe AIR 1.0 이상

브라우저의 웹 페이지에서와 마찬가지로 JavaScript를 사용하여 PDF 내용을 제어할 수 있습니다.

Acrobat에 대한 JavaScript 확장을 통해 제공되는 기능의 예는 다음과 같습니다.

- 페이지 탐색 및 확대/축소율 제어
- 문서 내 양식 처리
- 멀티미디어 이벤트 제어

Adobe Acrobat용 JavaScript 확장에 대한 자세한 내용은 Adobe Acrobat Developer Connection(http://www.adobe.com/devnet/acrobat/javascript.html)을 참조하십시오.

HTML-PDF 통신 기본 사항

Adobe AIR 1.0 이상

HTML 페이지의 JavaScript는 PDF 내용을 나타내는 DOM 객체의 postMessage() 메서드를 호출하여 PDF 내용의 JavaScript에 메시지를 보낼 수 있습니다. 예를 들어 다음 포함된 PDF 내용을 살펴봅니다.

```
<object id="PDFObj" data="test.pdf" type="application/pdf" width="100%" height="100%"/>
```

포함 HTML 내용의 다음 JavaScript 코드는 PDF 파일의 JavaScript에 메시지를 보냅니다.

```
pdfObject = document.getElementById("PDFObj");
pdfObject.postMessage(["testMsg", "hello"]);
```

PDF 파일은 이 메시지를 받기 위한 JavaScript를 포함할 수 있습니다. 문서 수준, 폴더 수준, 페이지 수준, 필드 수준, 일괄 처리 수준 컨텍스트 등 일부 컨텍스트에서 PDF 파일에 JavaScript 코드를 추가할 수 있습니다. 여기서는 PDF 문서가 열릴 때 평가되는 스크립트를 정의하는 문서 수준 컨텍스트에 대해서만 설명합니다.

PDF 파일은 hostContainer 객체에 messageHandler 속성을 추가할 수 있습니다. messageHandler 속성은 메시지에 응답하기 위한 핸들러 함수를 정의하는 객체입니다. 예를 들어 다음 코드에서는 PDF 파일이 호스트 컨테이너(PDF 파일을 포함하는 HTML 내용)에서 받은 메시지를 처리하기 위한 함수를 정의합니다.

```
this.hostContainer.messageHandler = {onMessage: myOnMessage};

function myOnMessage(aMessage)
{
    if(aMessage[0] == "testMsg")
    {
        app.alert("Test message: " + aMessage[1]);
    }
    else
    {
        app.alert("Error");
    }
}
```

HTML 페이지의 JavaScript 코드는 페이지에 포함된 PDF 객체의 postMessage() 메서드를 호출할 수 있습니다. 이 메서드를 호출하면 PDF 파일의 문서 수준 JavaScript에 "Hello from HTML"이라는 메시지가 전송됩니다.

```
<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>
```

Acrobat 8을 사용하여 PDF 파일에 JavaScript를 추가하는 방법에 대한 자세한 내용 및 고급 예제는 Adobe AIR에서 PDF 내용 크로스 스크립팅을 참조하십시오.

ActionScript에서 PDF 내용 스크립팅

Adobe AIR 1.0 이상

SWF 내용의 ActionScript 코드는 PDF 내용의 JavaScript와 직접 통신할 수 없습니다. 그러나 ActionScript는 PDF 내용을 로드하는 HTMLLoader 객체에 로드된 HTML 페이지의 JavaScript와 통신할 수 있으며 이 JavaScript 코드는 로드된 PDF 파일의 JavaScript와 통신할 수 있습니다. 자세한 내용은 892페이지의 “AIR에서 HTML 및 JavaScript 프로그래밍”을 참조하십시오.

AIR의 PDF 내용에 대해 알려진 제한 사항

Adobe AIR 1.0 이상

Adobe AIR의 PDF 내용에는 다음과 같은 제한 사항이 있습니다.

- PDF 내용은 투명한(transparent 속성이 true로 설정됨) 윈도우(NativeWindow 객체)에서 표시되지 않습니다.
- PDF 파일의 표시 순서는 AIR 응용 프로그램의 다른 표시 객체와 다르게 작동합니다. PDF 내용은 HTML 표시 순서에 따라 정확하게 고정되더라도 항상 AIR 응용 프로그램의 표시 순서에 따른 내용 위에 배치됩니다.
- PDF 문서가 포함된 HTMLLoader 객체의 특정 시각적 속성이 변경되면 PDF 문서가 표시되지 않습니다. 이러한 속성에는 filters, alpha, rotation 및 scaling 속성이 있습니다. 이러한 속성을 변경하면 속성이 재설정될 때까지 PDF 내용이 표시되지 않도록 렌더링됩니다. HTMLLoader 객체가 포함된 표시 객체 컨테이너의 이러한 속성을 변경하는 경우에도 PDF 내용이 표시되지 않습니다.
- PDF 내용이 포함된 NativeWindow 객체의 Stage 객체에 대한 scaleMode 속성을 StageScaleMode.NO_SCALE로 설정해야만 PDF 내용이 표시됩니다. 해당 속성을 다른 값으로 설정하면 PDF 내용이 표시되지 않습니다.
- PDF 파일 내부에 있는 내용에 대한 링크를 클릭하면 PDF 내용의 스크롤 위치가 업데이트됩니다. PDF 파일 외부에 있는 내용에 대한 링크를 클릭하면 링크의 대상이 새 윈도우인 경우에도 PDF가 포함된 HTMLLoader 객체가 리디렉션됩니다.
- AIR에서는 PDF 주석 처리 작업 과정이 작동하지 않습니다.

29장: 사용자 상호 작용의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램에서 ActionScript 3.0을 사용하여 대화형 작업을 만들어 사용자 작업에 응답할 수 있습니다. 이 단원의 내용은 사용자가 ActionScript 3.0 이벤트 모델을 사용하는 데 익숙하다는 것을 전제로 하여 설명합니다. 자세한 내용은 113페이지의 "이벤트 처리"를 참조하십시오.

사용자 입력 캡처

Flash Player 9 이상, Adobe AIR 1.0 이상

키보드, 마우스, 카메라 또는 이러한 장치의 조합을 통한 사용자 상호 작용은 가장 기본적인 상호 작용입니다. ActionScript 3.0에서 사용자 상호 작용에 대한 식별 및 응답은 주로 이벤트 수신과 관련되어 있습니다.

DisplayObject 클래스의 하위 클래스인 InteractiveObject 클래스는 사용자 상호 작용을 처리하는 데 필요한 일반적인 기능과 이벤트 구조를 제공합니다. InteractiveObject 클래스의 인스턴스를 직접 만들지는 않습니다. 대신 SimpleButton, Sprite, TextField와 여러 Flash 제작 도구 및 Flex 구성 요소 등의 표시 객체는 이 클래스로부터 사용자 상호 작용 모델을 상속하므로 일반 구조를 공유합니다. 따라서 InteractiveObject로부터 파생된 객체에서 사용자 상호 작용을 처리하기 위해 작성하는 코드와 배운 기술은 다른 모든 분야에도 적용이 가능합니다.

중요한 개념 및 용어

계속하기 전에, 사용자 상호 작용에 관한 다음의 주요 용어를 익혀야 합니다.

문자 코드 키보드에서 누르는 키와 관련하여 현재 문자 집합에서의 문자를 나타내는 숫자 코드입니다. 예를 들어 "D"와 "d"는 문자 코드는 서로 다르지만 영문 키보드에서 동일한 키로 입력됩니다.

컨텍스트 메뉴 사용자가 마우스 오른쪽 버튼을 클릭하거나 특정 키보드-마우스 조합을 사용할 때 나타나는 메뉴입니다. 컨텍스트 메뉴 명령은 일반적으로 클릭한 대상에 직접 적용되는 경우가 많습니다. 예를 들어 이미지에 대한 컨텍스트 메뉴에는 이미지를 별도의 윈도우에 표시하는 명령과 이미지를 다운로드하는 명령이 포함됩니다.

포커스 선택한 요소가 활성화되어 있으며 키보드 또는 마우스 상호 작용의 대상이 됨을 나타냅니다.

키 코드 키보드의 실제 키에 해당하는 숫자 코드입니다.

기타 도움말 항목

InteractiveObject

Keyboard

Mouse

ContextMenu

포커스 관리

Flash Player 9 이상, Adobe AIR 1.0 이상

대화형 객체는 프로그래밍 또는 사용자 액션을 통해 포커스를 받을 수 있습니다. 또한 tabEnabled 속성이 true로 설정되어 있는 경우에는 Tab 키를 눌러 특정 객체의 포커스를 다른 객체로 넘길 수 있습니다. 다음과 같은 경우를 제외하고, tabEnabled의 기본값은 false입니다.

- SimpleButton 객체의 경우 이 값이 true입니다.
- 입력 텍스트 필드의 경우 이 값이 true입니다.
- buttonMode가 true로 설정된 Sprite 객체 또는 MovieClip 객체의 경우 이 값이 true입니다.

이러한 경우 FocusEvent.FOCUS_IN 또는 FocusEvent.FOCUS_OUT에 대한 리스너를 추가하여 포커스 변경 시의 비헤이비어를 추가할 수 있습니다. 이는 특히 텍스트 필드 및 양식에 유용하지만 스프라이트, 동영상 클립 또는 InteractiveObject 클래스로부터 상속되는 모든 객체에서도 사용할 수 있습니다. 다음 예제에는 Tab 키를 사용하여 포커스를 순환할 수 있도록 설정하는 방법과 이후의 포커스 이벤트에 응답하는 방법이 나와 있습니다. 이 경우 각 정사각형에 포커스가 설정되면 색상이 변경됩니다.

참고: Flash Professional은 키보드 단축키를 사용하여 포커스를 관리합니다. 따라서 포커스 관리를 제대로 시뮬레이션하려면 Flash 대신 브라우저나 AIR에서 SWF 파일을 테스트해야 합니다.

```
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);
}
```

입력 유형 확인

Flash Player 10.1 이상, Adobe AIR 2 이상

Flash Player 10.1 및 Adobe AIR 2 릴리스에서는 런타임 환경을 테스트하여 특정 입력 유형이 지원되는지 여부를 확인할 수 있는 기능을 도입했습니다. 이제 ActionScript를 사용하여 런타임이 현재 배포된 장치에서 다음을 테스트할 수 있습니다.

- 스타일러스 또는 손가락 입력 지원(또는 터치 입력 지원 안 함)
- 사용자를 위한 가상 또는 물리적 키보드 제공(또는 키보드 제공 안 함)
- 커서 표시(표시하지 않을 경우 객체를 마우스로 가리켜야 사용 가능한 기능이 작동하지 않음)

입력 확인 ActionScript API는 다음과 같습니다.

- flash.system.Capabilities.touchscreenType 속성: 현재 환경에서 지원되는 입력 유형을 나타내며 런타임에 제공되는 값입니다.
- flash.system.TouchscreenType 클래스: Capabilities.touchscreenType 속성에 대한 열거형 값 상수의 클래스입니다.
- flash.ui.Mouse.supportsCursor 속성: 영구 커서를 사용할 수 있는지 여부를 나타내며 런타임에 제공되는 값입니다.
- flash.ui.Keyboard.physicalKeyboardType 속성: 물리적 키보드의 모든 기능을 사용할 수 있는지, 숫자 키패드를 사용할 수 있는지 또는 키보드를 전혀 사용할 수 없는지를 나타내며 런타임에 제공되는 값입니다.
- flash.ui.KeyboardType 클래스: flash.ui.Keyboard.physicalKeyboardType 속성에 대한 열거형 값 상수의 클래스입니다.
- flash.ui.Keyboard.hasVirtualKeyboard 속성: 가상 키보드(물리적 키보드 대신 또는 물리적 키보드에 추가로)가 사용자에게 제공되는지 여부를 나타내며 런타임에 제공되는 값입니다.

입력 확인 API를 사용하면 사용자의 장치 기능을 이용하거나 해당 기능이 없는 경우 대체 방법을 제공할 수 있습니다. 입력 확인 API는 휴대 장치 및 터치 지원 응용 프로그램을 개발하는 경우에 특히 유용합니다. 예를 들어 휴대 장치 인터페이스에 스타일러스용 작은 버튼을 있는 경우 사용자가 손가락 터치로 입력할 수 있도록 더 큰 버튼을 갖는 대체 인터페이스를 제공할 수 있습니다. 다음은 두 함수를 갖는 응용 프로그램에 대한 코드입니다. 여기에서 createStylusUI() 함수는 스타일러스 상호 작용에 적합한 사용자 인터페이스 요소 세트를 할당하며, createTouchUI() 함수는 손가락 상호 작용에 적합한 사용자 인터페이스 요소 세트를 할당합니다.

```
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();
}
```

서로 다른 입력 환경에 대한 응용 프로그램을 개발하는 경우 다음과 같은 호환성 차트를 고려해 볼 수 있습니다.

환경	**supportsCursor**	**touchscreenType == FINGER**	**touchscreenType == STYLUS**	**touchscreenType == NONE**
기존 데스크톱	true	false	false	true
정전식 터치 스크린 장치(미세한 사용자 터치를 감지하는 타블렛, PDA 및 Apple iPhone, Palm Pre 등의 휴대폰)	false	true	false	false
저항식 터치 스크린 장치(높은 압력의 정밀 접촉을 감지하는 타블렛, PDA 및 HTC Fuze 등의 휴대폰)	false	false	true	false
비 터치 스크린 장치(접촉을 감지하는 응용 프로그램을 실행하지만 접촉을 감지하는 스크린이 없는 일반 휴대폰)	false	false	false	true

참고: 장치 플랫폼에 따라 다양한 조합의 입력 유형을 지원할 수 있습니다. 이 차트는 일반적인 지침으로 사용하십시오.

30장: 키보드 입력

Flash Player 9 이상, Adobe AIR 1.0 이상

키보드 입력을 캡처하여 응답하는 한편, IME를 조작함으로써 사용자가 멀티바이트 언어에서 비 ASCII 텍스트 문자를 입력하게 할 수 있습니다. 이 단원의 내용은 사용자가 ActionScript 3.0 이벤트 모델을 사용하는 데 익숙하다는 것을 전제로 하여 설명합니다. 자세한 내용은 113페이지의 "이벤트 처리"를 참조하십시오.

사용 가능한 키보드 지원 유형(예: 물리적, 가상, 영숫자, 12버튼 숫자)을 런타임에서 확인하는 방법에 대한 자세한 내용은 507페이지의 "입력 유형 확인"을 참조하십시오.

IME(Input Method Editor)를 사용하면 사용자가 표준 키보드를 통해 복잡한 문자와 심볼을 입력할 수 있습니다. IME 클래스를 사용하면 사용자가 응용 프로그램에서 자신의 시스템 IME를 활용할 수 있습니다.

기타 도움말 항목

flash.events.KeyboardEvent

flash.system.IME

키보드 입력 캡처

Flash Player 9 이상, Adobe AIR 1.0 이상

InteractiveObject 클래스에서 상호 작용 모델을 상속한 표시 객체는 이벤트 리스너를 사용하여 키보드 이벤트에 응답할 수 있습니다. 예를 들어 스테이지에 이벤트 리스너를 배치하여 키보드 입력을 수신하고 이에 응답할 수 있습니다. 다음 코드에서는 이벤트 리스너가 키보드 입력을 캡처하고 키 이름과 키 코드 속성이 표시됩니다.

```
function reportKeyDown(event:KeyboardEvent):void
{
    trace("Key Pressed: " + String.fromCharCode(event.charCode) + " (character code: " + event.charCode +
")");
}
stage.addEventListener(KeyboardEvent.KEY_DOWN, reportKeyDown);
```

Ctrl 키와 같은 몇몇 키는 글리프 표현 없이도 이벤트를 생성합니다.

이전 코드 예제에서 키보드 이벤트 리스너는 전체 스테이지에 대한 키보드 입력을 캡처했습니다. 스테이지의 특정 표시 객체에 대한 이벤트 리스너를 작성할 수도 있습니다. 이 이벤트 리스너는 객체에 포커스가 맞춰지면 트리거됩니다.

다음 예제에서는 사용자가 TextField 인스턴스 내에 입력할 때만 입력한 키가 출력 패널에 반영됩니다. Shift 키를 누르고 있으면 TextField의 테두리 색상이 일시적으로 빨간색으로 바뀝니다.

이 코드는 스테이지에 tf라는 이름의 TextField 인스턴스가 있다고 가정합니다.

```
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;
    }
}
```

또한 TextField 클래스는 textInput 이벤트를 보고하며, 사용자는 텍스트를 입력할 때 이를 수신할 수 있습니다. 자세한 내용은 340페이지의 “텍스트 입력 캡처”를 참조하십시오.

참고: AIR 런타임에서는 키보드 이벤트를 취소할 수 있지만 Flash Player 런타임에서는 키보드 이벤트를 취소할 수 없습니다.

키 코드 및 문자 코드

Flash Player 9 이상, Adobe AIR 1.0 이상

키보드 이벤트의 keyCode 및 charCode 속성에 액세스하여 어떤 키를 눌렀는지 확인하고 다른 액션을 트리거할 수 있습니다. keyCode 속성은 키보드의 특정 키 값에 해당하는 숫자 값이고, charCode 속성은 현재의 문자 세트에서 해당 키가 가지는 숫자 값입니다. 이때 기본 문자 집합은 ASCII를 지원하는 UTF-8입니다.

키 코드와 문자 값에는 주요한 차이가 있는데, 키 코드 값은 키보드의 특정 키를 나타내고(키패드의 1 키는 키보드 상단의 1 키와 다르지만 "1"을 입력하는 키와 "!"를 입력하는 키는 동일), 문자 값은 특정 문자를 나타냅니다(R 문자와 r 문자는 서로 다름).

참고: 키와 ASCII의 문자 코드 값 간의 매핑은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 flash.ui.Keyboard 클래스를 참조하십시오.

키와 키 코드의 매핑은 장치 및 운영 체제에 따라 다릅니다. 따라서 키 매핑을 사용하여 액션을 트리거해서는 안 됩니다. 대신에 Keyboard 클래스에서 제공하는 미리 정의된 상수 값을 사용하여 적당한 keyCode 속성을 참조해야 합니다. 예를 들어 Shift 키의 키 매핑 대신 Keyboard.SHIFT 상수를 사용하십시오(위의 코드 샘플 참조).

KeyboardEvent 우선 순위

Flash Player 9 이상, Adobe AIR 1.0 이상

다른 이벤트와 마찬가지로 키보드 이벤트 시퀀스도 addEventListener() 메서드의 코드 내 지정 순서가 아닌 표시 객체의 계층 구조에 의해 결정됩니다.

예를 들어 다음 예제와 같이 container라는 동영상 클립 내에 tf 텍스트 필드를 배치하고 두 인스턴스에 키보드 이벤트의 이벤트 리스너를 추가하는 경우가 있을 수 있습니다.

```
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 + ")");
}
```

텍스트 필드와 부모 컨테이너에 모두 리스너가 있기 때문에 텍스트 필드에 키를 입력할 때마다 reportKeyDown() 함수가 두 번씩 호출됩니다. 키를 하나 누르면 텍스트 필드가 이벤트를 전달하고 그 뒤에 container 동영상 클립이 이벤트를 전달합니다.

운영 체제와 웹 브라우저는 Adobe Flash Player 또는 AIR에 앞서 키보드 이벤트를 처리합니다. 예를 들어 Microsoft Internet Explorer에서 Ctrl+W를 누르면 포함된 SWF 파일이 키보드 이벤트를 전달하기 전에 브라우저 윈도우가 닫힙니다.

IME 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

IME 클래스를 사용하면 Flash Player 또는 Adobe AIR 내에서 운영 체제의 IME를 조작할 수 있습니다.

ActionScript를 사용하여 다음을 확인할 수 있습니다.

- IME가 사용자의 컴퓨터에 설치되어 있는지 여부(Capabilities.hasIME)
- IME가 사용자의 컴퓨터에서 활성화되어 있는지 아니면 비활성화되어 있는지 여부(IME.enabled)
- 현재 IME가 사용하는 변환 모드(IME.conversionMode)

입력 텍스트 필드를 특정 IME 컨텍스트에 연결시킬 수 있습니다. 입력 필드 간에 전환할 때 IME를 히라가나(일본어), 최대 폭 숫자, 반폭 숫자, 직접 입력 등으로 서로 전환할 수도 있습니다.

IME를 사용하면 한국어, 중국어 및 일본어 등 ASCII가 아닌 멀티바이트 언어 텍스트 문자를 입력할 수 있습니다.

IME 사용에 대한 자세한 내용은 개발 중인 응용 프로그램을 실행할 운영 체제의 설명서를 참조하십시오. 추가 리소스를 보려면 다음 웹 사이트를 참조하십시오.

- http://www.msdn.microsoft.com/goglobal/
- http://developer.apple.com/library/mac/navigation/
- http://www.java.sun.com/

참고: 사용자의 컴퓨터에 IME가 활성화되어 있지 않으면 Capabilities.hasIME 이외의 IME 메서드 또는 속성이 호출되지 않습니다. 이때 IME를 수동으로 활성화시키면 ActionScript에서 이후에 IME 메서드 및 속성을 정상적으로 호출할 수 있습니다. 예를 들어 일본어 IME를 사용하는 경우 IME 메서드 또는 속성을 호출하기 전에 해당 IME를 활성화해야 합니다.

IME 설치 여부 및 활성화 여부 확인

Flash Player 9 이상, Adobe AIR 1.0 이상

IME 메서드나 속성을 호출하기 전에 현재 사용자의 컴퓨터에 IME가 설치되어 있으며 활성화되어 있는지 항상 확인해야 합니다. 다음 코드는 메서드를 호출하기 전에 IME가 설치 여부 및 활성화 여부를 확인하는 방법을 보여 줍니다.

```
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.");
}
```

앞의 코드는 먼저 Capabilities.hasIME 속성을 사용하여 IME 설치 여부를 확인합니다. 이 속성이 true로 설정되어 있으면 이 코드가 IME.enabled 속성을 사용하여 사용자의 IME가 현재 활성화되어 있는지 여부를 확인합니다.

현재 활성화된 IME 변환 모드 확인

Flash Player 9 이상, Adobe AIR 1.0 이상

다국어 응용 프로그램을 작성할 때 사용자가 현재 활성화되어 있는 변환 모드를 확인해야 할 수 있습니다. 다음 코드는 메서드를 호출하기 전에 IME 설치 여부 확인 방법 및 IME가 설치된 경우 현재 활성화된 IME 변환 모드 확인 방법을 보여 줍니다.

```
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.";
}
```

앞의 코드는 먼저 IME가 설치되어 있는지 여부를 확인합니다. 다음으로 IMEConversionMode 클래스의 각 상수와 IME.conversionMode 속성을 비교하여 현재 IME에서 사용하고 있는 변환 모드를 확인합니다.

IME 변환 모드 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자의 IME 변환 모드를 변경할 때 코드가 try..catch 블록에서 래핑되는지 확인해야 합니다. IME에서 변환 모드를 설정할 수 없는 경우에 conversionMode 속성을 사용하여 변환 모드를 설정하면 오류가 발생할 수 있습니다. 다음 코드는 try..catch 블록을 사용하여 IME.conversionMode 속성을 설정하는 방법을 보여 줍니다.

```
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;
    }
}
```

앞의 코드는 먼저 사용자에게 상태 메시지를 표시하는 데 사용되는 텍스트 필드를 만듭니다. 그 다음에는 IME가 설치되어 있는 경우 이 코드가 IME를 활성화하고 변환 모드를 한국어로 설정합니다. 사용자의 컴퓨터에 한국어 IME가 설치되어 있지 않으면 Flash Player 또는 AIR에서 오류가 발생하고 try..catch 블록에서 이 오류가 catch됩니다. try..catch 블록은 앞에서 만든 텍스트 필드에 오류 메시지를 표시합니다.

특정 텍스트 필드에 대해 IME 비활성화

Flash Player 9 이상, Adobe AIR 1.0 이상

경우에 따라 문자를 입력할 때 사용자의 IME를 비활성화할 수 있습니다. 예를 들어 숫자만 입력할 수 있는 텍스트 필드가 있는 경우 IME를 사용하지 않고 천천히 데이터를 입력할 수 있습니다.

다음 예제에서는 FocusEvent.FOCUS_IN 및 FocusEvent.FOCUS_OUT 이벤트를 수신하고 그에 따라 사용자의 IME를 비활성화하는 방법을 보여 줍니다.

```
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;
    }
}
```

다음 예제는 phoneTxt와 nameTxt라는 두 가지 입력 텍스트 필드를 만든 다음 두 개의 이벤트 리스너를 phoneTxt 텍스트 필드에 추가합니다. 사용자가 phoneTxt 텍스트 필드를 강조하도록 설정하면 FocusEvent.FOCUS_IN 이벤트가 전달되고 IME가 비활성화됩니다. phoneTxt 텍스트 필드가 강조되지 않으면 FocusEvent.FOCUS_OUT 이벤트가 전달되어 IME를 다시 활성화합니다.

IME 구성 이벤트 수신

Flash Player 9 이상, Adobe AIR 1.0 이상

구성 문자열을 설정하면 IME 구성 이벤트가 전달됩니다. 예를 들어 사용자가 IME를 활성화하고 일본어로 문자열을 입력한 경우 사용자가 구성 문자열을 선택하는 즉시 IMEEvent.IME_COMPOSITION 이벤트가 전달됩니다.
IMEEvent.IME_COMPOSITION 이벤트를 수신하려면 다음 예제와 같이 이벤트 리스너를 System 클래스 (flash.system.System.ime.addEventListener(...))의 정적 ime 속성에 추가해야 합니다.

```
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;
}
```

앞의 코드는 두 개의 텍스트 필드를 만들고 해당 필드를 표시 목록에 추가합니다. 첫 번째 텍스트 필드인 inputTxt는 사용자가 일본어 텍스트를 입력할 수 있는 입력 텍스트 필드입니다. 두 번째 텍스트 필드인 outputTxt는 사용자에게 오류 메시지를 표시하거나 사용자가 inputTxt 텍스트 필드에 입력한 일본어 문자열을 반복하여 표시하는 동적 텍스트 필드입니다.

가상 키보드

Flash Player 10.2 이상, AIR 2.6 이상

휴대 전화 및 타블렛 같은 휴대 장치에서는 물리적 키보드 대신 가상 소프트웨어 키보드를 제공하는 경우가 많습니다. Flash API의 클래스를 사용하면 다음을 수행할 수 있습니다.

- 가상 키보드가 표시될 때와 닫힐 때를 감지할 수 있습니다.
- 키보드가 표시되지 않도록 할 수 있습니다.
- 가상 키보드에 의해 가려지는 스테이지의 영역을 결정할 수 있습니다.
- 포커스를 얻으면 키보드를 표시하는 대화형 객체를 만들 수 있습니다. iOS의 AIR 응용 프로그램에서는 지원되지 않습니다.

- (AIR만 해당) 응용 프로그램에서 키보드에 맞게 디스플레이를 수정할 수 있도록 자동 패닝 비헤이비어를 비활성화할 수 있습니다.

가상 키보드 비헤이비어 제어

사용자가 텍스트 필드 또는 특별 구성된 대화형 객체의 안쪽을 누르면 런타임이 가상 키보드를 자동으로 엽니다. 키보드가 열리면 런타임은 사용자가 자신이 입력한 텍스트를 볼 수 있도록 응용 프로그램 내용을 이동하고 크기를 조정할 때 기본 플랫폼 규칙에 따릅니다.

키보드가 열리면 포커스가 있는 객체가 다음 이벤트를 순서대로 전달합니다.

softKeyboardActivating 이벤트 - 키보드가 스테이지 위로 올라오기 직전에 전달됩니다. 전달된 이벤트 객체의 preventDefault() 메서드를 호출하면 가상 키보드가 열리지 않습니다.

softKeyboardActivate 이벤트 - softKeyboardActivating 이벤트 처리가 끝난 후 전달됩니다. 포커스가 있는 객체가 이 이벤트를 전달할 때 스테이지 객체의 softKeyboardRect 속성은 가상 키보드에 의해 가려진 스테이지 영역을 반영하도록 업데이트된 상태입니다. 이 이벤트는 취소할 수 없습니다.

참고: 사용자가 키보드 유형을 변경하는 등의 이유로 키보드의 크기가 변경되면 포커스가 있는 객체는 두 번째 softKeyboardActivate 이벤트를 전달합니다.

softKeyboardDeactivate 이벤트 - 어떠한 이유로든 가상 키보드가 닫힐 때 전달됩니다. 이 이벤트는 취소할 수 없습니다.

다음 예제에서는 스테이지에 두 가지 TextField 객체를 추가합니다. 위쪽의 TextField는 사용자가 필드를 누를 때 키보드가 올라오지 못하게 하고 이미 올라와 있는 경우 키보드를 닫습니다. 아래쪽 TextField는 기본 비헤이비어를 보여 줍니다. 예제에서는 두 텍스트 필드가 전달하는 소프트 키보드 이벤트를 모두 보고합니다.

```
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 );
    }
}
}
```

대화형 객체를 위한 가상 키보드 지원 추가

Flash Player 10.2 이상, AIR 2.6 이상(iOS에서는 지원되지 않음)

일반적으로 가상 키보드는 TextField 객체가 눌러질 때만 열립니다. 포커스를 받으면 가상 키보드를 열도록 InteractiveObject 클래스의 인스턴스를 구성할 수 있습니다.

소프트 키보드를 열도록 InteractiveObject 인스턴스를 구성하려면 해당 needsSoftKeyboard 속성을 true로 설정합니다. 객체가 스테이지 포커스 속성에 할당되면 항상 소프트 키보드가 자동으로 열립니다. 또한 InteractiveObject의 requestSoftKeyboard() 메서드를 호출하여 키보드를 표시할 수도 있습니다.

다음 예제에서는 InteractiveObject가 텍스트 입력 필드로 사용되도록 프로그래밍하는 방법을 보여 줍니다. 예제에 나와 있는 TextInput 클래스는 필요한 경우 키보드가 표시되도록 needsSoftKeyboard 속성을 설정합니다. 그런 다음 이 객체는 keyDown 이벤트를 수신하고 입력된 문자를 필드에 삽입합니다.

이 예제에서는 Flash 텍스트 엔진을 사용하여 입력된 텍스트를 추가 및 표시하고 일부 중요한 이벤트를 처리합니다. 내용을 단순화하기 위해 완전한 기능을 갖춘 텍스트 필드는 구현하지 않습니다.

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

다음 기본 응용 프로그램 클래스에서는 키보드가 표시되거나 장치 방향이 변경될 때 TextInput 클래스를 사용하고 응용 프로그램 레이아웃을 관리하는 방법을 보여 줍니다. 기본 클래스는 TextInput 객체를 만들고, 스테이지를 채우도록 해당 경계를 설정합니다. 소프트 키보드가 표시되거나 스테이지의 크기가 변경되면 클래스에서 TextInput 객체의 크기를 조절합니다. 클래스는 TextInput 객체로부터 소프트 키보드 이벤트를 수신하고 스테이지에서 이벤트의 크기를 조절합니다. 응용 프로그램에서는 이벤트의 원인에 관계없이 스테이지의 표시 가능한 영역을 결정하고 이를 채우도록 입력 컨트롤의 크기를 조절합니다. 따라서 실제 응용 프로그램에서는 자연적으로 보다 정교한 레이아웃 알고리즘이 필요하게 됩니다.

```
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;
        }
    }
}
```

참고: scaleMode 속성이 noScale로 설정되어 있으면 스테이지가 방향 변경에 대해서만 resize 이벤트를 전달합니다. 다른 모드에서는 스테이지의 크기가 변경되지 않는 대신 보정을 위해 내용의 크기가 조절됩니다.

응용 프로그램 표시 변경 처리

AIR 2.6 이상

AIR에서는 응용 프로그램 설명자의 softKeyboardBehavior 요소를 none으로 설정하여 소프트 키보드를 표시하는 것과 관련된 기본 패닝 및 크기 조절 비헤이비어를 해제할 수 있습니다.

```
<softKeyboardBehavior>none</softKeyboardBehavior>
```

자동 비헤이비어를 해제하면 응용 프로그램 디스플레이에 맞게 필요한 조정을 수행하는 동작을 응용 프로그램에서 처리해야 합니다. softKeyboardActivate 이벤트는 키보드가 열릴 때 전달됩니다. softKeyboardActivate 이벤트가 전달되는 경우 열린 키보드에 의해 가려진 영역의 크기는 스테이지의 softKeyboardRect 속성에 들어 있습니다. 이러한 크기를 사용하여 내용을 이동하거나 내용의 크기를 조절하면 키보드가 열린 상태에서 사용자가 입력하는 동안 내용이 올바르게 표시됩니다. 키보드가 닫히면 softKeyboardRect 사각형의 크기는 모두 0이 됩니다.

키보드가 닫히면 softKeyboardDeactivate 이벤트가 전달되고 응용 프로그램 표시를 일반으로 되돌릴 수 있습니다.

```
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;
        }
    }
}
```

참고: Android에서는 전체 화면 모드를 포함하여 운영 체제에서 키보드의 정확한 크기를 사용할 수 없는 경우가 있습니다. 이러한 경우에는 예상 크기가 지정됩니다. 또한 가로 방향에서는 모든 텍스트 입력에 기본 전체 화면 IME 키보드가 사용됩니다. 이 IME 키보드는 텍스트 입력 필드가 기본 제공되며 전체 스테이지를 가립니다. 스크린을 채우지 않도록 가로 키보드를 표시할 수는 없습니다.

31장: 마우스 입력

Flash Player 9 이상, Adobe AIR 1.0 이상

마우스 입력을 캡처하고 응답함으로써 대화형 작업을 만들 수 있습니다. 이 단원의 내용은 사용자가 ActionScript 3.0 이벤트 모델을 사용하는 데 익숙하다는 것을 전제로 하여 설명합니다. 자세한 내용은 113페이지의 "이벤트 처리"를 참조하십시오.

사용 가능한 마우스 지원 유형(예: 영구 커서, 스타일러스 또는 터치 입력)을 런타임에서 확인하는 방법에 대한 자세한 내용은 507페이지의 "입력 유형 확인"을 참조하십시오.

기타 도움말 항목

flash.ui.Mouse

flash.events.MouseEvent

528페이지의 "터치, 다중 터치 및 동작 입력"

마우스 입력 캡처

Flash Player 9 이상, Adobe AIR 1.0 이상

마우스를 클릭하면 대화형 기능을 트리거하는 데 사용할 수 있는 마우스 이벤트가 만들어집니다. 이벤트 리스너를 스테이지에 추가하면 SWF 내에서 발생하는 마우스 이벤트를 수신할 수 있습니다. 또한 InteractiveObject로부터 상속되는 스테이지의 객체에(예: Sprite 또는 MovieClip) 이벤트 리스너를 추가할 수 있습니다. 이러한 리스너는 객체를 클릭하면 트리거됩니다.

키보드 이벤트와 마찬가지로 마우스 이벤트도 버블링됩니다. 다음 예제에서 square는 Stage의 자식이므로 정사각형을 클릭하면 Sprite square와 Stage 객체 둘 다에서 이벤트가 전달됩니다.

```
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 + "]");
}
```

위 예제의 마우스 이벤트에는 클릭 위치 정보가 포함되어 있습니다. localX 및 localY 속성에는 표시 체인의 최하위 자식에 대한 마우스 클릭 위치가 포함되어 있습니다. 예를 들어, square의 왼쪽 위 모서리를 클릭하면 square의 등록 포인트인 로컬 좌표 [0,0]이 보고됩니다. 또는 stageX 및 stageY 속성이 스테이지 클릭의 전역 좌표를 참조합니다. 이 좌표의 경우 같은 클릭으로 [50,50]이 보고됩니다. square가 해당 좌표로 이동되었기 때문입니다. 이러한 두 좌표 쌍은 사용자 상호 작용에 대한 응답 방식에 따라 유용할 수 있습니다.

참고: 전체 화면 모드에서 마우스 잠금을 사용하도록 응용 프로그램을 구성할 수 있습니다. 마우스 잠금을 활성화하면 커서가 비활성화되고 마우스를 제한 없이 움직일 수 있습니다. 자세한 내용은 151페이지의 "전체 화면 모드 작업"을 참조하십시오.

MouseEvent 객체에는 altKey, ctrlKey 및 shiftKey라는 부울 속성도 포함되어 있습니다. 이러한 속성을 사용하여 마우스 클릭 시 Alt, Ctrl 또는 Shift 키를 함께 눌렀는지 여부를 확인할 수 있습니다.

스테이지 주위에서 Sprite 드래그

Flash Player 9 이상, Adobe AIR 1.0 이상

Sprite 클래스의 startDrag() 메서드를 사용하여 사용자가 스테이지 주위에서 Sprite 객체를 드래그하도록 할 수 있습니다. 아래의 코드는 여기에 해당하는 예제입니다.

```
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);
}
```

자세한 내용은 156페이지의 “위치 변경”에서 마우스 드래그 상호 작용을 만드는 단원을 참조하십시오.

AIR에서 드래그 앤 드롭

Adobe AIR에서는 사용자가 데이터를 응용 프로그램 안팎으로 드래그할 수 있도록 드래그 앤 드롭 지원을 설정할 수 있습니다. 자세한 내용은 553페이지의 “AIR에서 드래그 앤 드롭”을 참조하십시오.

마우스 커서 사용자 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

스테이지의 표시 객체에 대해 마우스 커서(마우스 포인터)를 숨기거나 교체할 수 있습니다. 마우스 커서를 숨기려면 Mouse.hide() 메서드를 호출합니다. Mouse.hide()를 호출하고 해당 스테이지에서 MouseEvent.MOUSE_MOVE 이벤트를 수신하는 한편 표시 객체(사용자 정의 커서)의 좌표를 해당 이벤트의 stageX 및 stageY 속성으로 설정함으로써 커서를 사용자 정의할 수 있습니다. 다음 예제에서는 이 작업을 실행하는 방법을 보여 줍니다.

```
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;
}
```

마우스 입력 예제: WordSearch

Flash Player 9 이상, Adobe AIR 1.0 이상

이 예제는 마우스 이벤트 처리를 통한 사용자 상호 작용에 대한 것입니다. 사용자는 바둑판 무늬에 배열된 임의의 문자로 가능한 한 많은 단어를 조합해야 하며, 문자를 가로 또는 세로로 이동하여 철자를 맞추고 같은 문자를 두 번 사용할 수 없습니다. 이 예제에서는 다음과 같은 ActionScript 3.0의 기능이 사용됩니다.

- 구성 요소 배열을 동적으로 구성
- 마우스 이벤트에 응답
- 사용자 상호 작용에 기초하여 점수 유지 관리

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. WordSearch 응용 프로그램 파일은 Samples/WordSearch 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
WordSearch.as	응용 프로그램의 기본 기능을 제공하는 클래스입니다.
WordSearch.fla 또는 WordSearch.mxml	Flex(MXML) 또는 Flash(FLA) 형식의 기본 응용 프로그램 파일입니다.
dictionary.txt	입력한 단어가 득점 가능하며 정확하게 입력되었는지 확인하는 데 사용되는 파일입니다.

사전 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

단어 찾기와 관련된 게임을 만들기 위해서는 사전이 필요합니다. 이 예제에는 캐리지 리턴으로 구분된 단어 목록을 포함하는 dictionary.txt 텍스트 파일이 포함되어 있습니다. words라는 배열을 만들면 loadDictionary() 함수에서 이 파일을 요청하며, 로드된 파일은 긴 문자열이 됩니다. 그러면 split() 메서드를 사용하여 이 문자열을 단어 배열로 파싱할 수 있습니다. 이때 캐리지 리턴(문자 코드 10) 또는 새 행(문자 코드 13)의 각 인스턴스에서 줄 바꿈됩니다. 이 파싱 작업은 dictionaryLoaded() 함수에서 실행됩니다.

```
words = dictionaryText.split(String.fromCharCode(13, 10));
```

사용자 인터페이스 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

단어가 저장되면 사용자 인터페이스를 설정할 수 있습니다. 이때 Button 인스턴스를 두 개 만들어야 하는데, 하나는 단어를 제출할 때 사용되고 다른 하나는 현재 철자를 맞추고 있는 단어를 지울 때 사용됩니다. 각 상황에서 해당 버튼이 브로드캐스팅하는 MouseEvent.CLICK 이벤트를 수신한 다음 함수를 호출하여 사용자 입력에 응답해야 합니다. setupUI() 함수에서 이 코드는 두 개의 버튼에 리스너를 만듭니다.

```
submitWordButton.addEventListener(MouseEvent.CLICK,submitWord);
clearWordButton.addEventListener(MouseEvent.CLICK,clearWord);
```

게임 보드 생성

Flash Player 9 이상, Adobe AIR 1.0 이상

게임 보드는 바둑판 무늬로 된 임의의 문자 배열입니다. generateBoard() 함수에서 한 루프 내에 다른 루프를 중첩시켜 2차원 배열을 만듭니다. 첫 번째 루프에서는 행이 증가되고 두 번째 루프에서는 한 행당 총 열 수가 증가됩니다. 이러한 행과 열에 의해 만들어진 각 셀에는 보드의 문자를 나타내는 버튼이 포함됩니다.

```
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);
        }
    }
}
```

오직 한 행의 MouseEvent.CLICK 이벤트에만 리스너를 추가해도 이것이 for 루프이므로 각각의 Button 인스턴스에 지정됩니다. 또한 각 버튼에는 해당 행 및 열 위치에서 파생된 이름이 지정되므로 나중에 코드에서 각 버튼의 행과 열을 참조하기가 쉽습니다.

사용자 입력을 통해 단어 작성

Flash Player 9 이상, Adobe AIR 1.0 이상

가로 또는 세로로 인접한 문자를 선택하여 철자를 맞출 수 있지만 같은 문자를 두 번 사용할 수 없습니다. 한 번 클릭할 때마다 마우스 이벤트가 생성되는데 이 시점에서 사용자가 입력하는 단어를 점검하여 이전에 클릭한 문자에서 적절하게 이어지는지 확인해야 합니다. 제대로 연결되지 않으면 이전 단어는 제거되고 새 단어가 시작됩니다. 이 검사 작업은 isLegalContinuation() 메서드에서 실행됩니다.

```
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));
}
```

String 클래스의 charAt() 및 indexOf() 메서드는 현재 클릭한 버튼과 이전에 클릭한 버튼 둘 다에서 적절한 행과 열을 얻습니다. 이때 isLegalContinuation() 메서드는 행 또는 열이 변경되지 않았거나 변경된 행 또는 열의 위치가 이전 행 또는 열에서 한 칸 이내일 경우 true를 반환합니다. 대각선으로 단어의 철자를 맞출 수 있도록 게임 규칙을 변경하려면 변경되지 않은 행 또는 열에 대해 검사 작업을 제거해야 하며, 이 경우 마지막 행이 다음과 같습니다.

```
return (Math.abs(prevButtonRow - currButtonRow) <= 1) && Math.abs(prevButtonCol - currButtonCol) <= 1));
```

제안 단어 검사

Flash Player 9 이상, Adobe AIR 1.0 이상

게임에 대한 코드를 완성하려면 제안 단어 검사 및 점수 기록을 위한 메커니즘이 필요합니다. searchForWord() 메서드에는 두 가지 메커니즘이 모두 들어 있습니다.

```
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;
}
```

이 함수는 사전에서 모든 단어를 반복 탐색합니다. 사용자의 단어가 사전의 단어와 일치하면 사전에서의 위치가 반환됩니다. 그런 다음 submitWord() 메서드가 응답을 검사하여 위치가 유효한 경우 점수를 업데이트합니다.

사용자 정의

Flash Player 9 이상, Adobe AIR 1.0 이상

클래스 시작 부분에는 몇 개의 상수가 있습니다. 이러한 변수를 수정하여 게임을 수정할 수 있습니다. 예를 들어 TOTAL_TIME 변수의 값을 늘려 게임 플레이 시간을 변경할 수 있습니다. 또한 PERCENT_VOWELS 변수의 값도 약간 늘릴 수 있는데, 이렇게 하면 단어를 찾을 가능성이 커집니다.

32장: 터치, 다중 터치 및 동작 입력

Flash Player 10.1 이상, Adobe AIR 2 이상

Flash Platform에서는 터치 지원 장치에서의 단일 접점 또는 다중 접점 입력을 포함한 터치이벤트 처리 기능을 제공합니다. 또한 Flash 런타임에서는 다중 접점과 동작을 만드는 움직임을 결합한 이벤트도 처리합니다. 즉, Flash 런타임에서는 두 가지 입력 유형을 해석합니다.

터치 터치 지원 장치에서 손가락, 스타일러스 또는 기타 도구와 같은 단일 접점 장치를 사용한 입력입니다. 일부 장치는 손가락 또는 스타일러스를 사용한 다중 동시 접점을 지원합니다.

다중 터치 두 개 이상의 동시 접점을 갖는 입력입니다.

동작 한 가지 이상의 터치 이벤트에 대한 응답으로 장치 또는 운영 체제에서 해석한 입력입니다. 예를 들어 사용자가 두 손가락을 동시에 회전한 경우 장치 또는 운영 체제에서 해당 터치 입력을 회전 동작으로 해석할 수 있습니다. 일부 동작은 한 손가락 또는 터치 지점을 사용하여 수행되고 일부 동작은 여러 터치 지점이 필요합니다. 장치 또는 운영 체제에서는 입력에 할당할 동작 유형을 설정합니다.

사용자의 장치에 따라 터치와 동작 입력이 모두 다중 터치 입력이 될 수 있습니다. ActionScript는 터치 이벤트, 동작 이벤트 및 여러 입력에 대해 개별적으로 추적되는 터치 이벤트를 처리하는 API를 제공합니다.

참고: 터치 및 제스처 이벤트를 수신하는 것은 컴퓨팅 장치와 운영 체제에 따라 초당 몇 프레임을 렌더링하는 것과 맞먹는 상당한 프로세싱 리소스를 소모할 수 있습니다. 따라서 터치 또는 제스처를 통해 얻는 특별한 기능이 실제로 필요하지 않다면 대개 마우스 이벤트를 사용하는 것이 낫습니다. 터치 또는 제스처 이벤트를 사용하는 경우 발생할 수 있는 그래픽 변화를 줄이도록 하십시오. 특히 이동, 회전 또는 확대/축소 시와 같이 이벤트가 빠르게 전달되는 경우 그렇게 해야 합니다. 예를 들어 사용자가 확대/축소 제스처를 사용하여 크기를 조정하는 동안에는 구성 요소 내의 애니메이션이 중지되도록 할 수 있습니다.

기타 도움말 항목

flash.ui.Multitouch

flash.events.TouchEvent

flash.events.GestureEvent

flash.events.TransformGestureEvent

flash.events.GesturePhase

flash.events.PressAndTapGestureEvent

Paul Trani: 휴대 장치에서의 터치 이벤트와 제스처

Mike Jones: 가상 게임 컨트롤러

터치 입력의 기초

Flash Player 10.1 이상, Adobe AIR 2 이상

Flash Platform이 터치 입력을 지원하는 환경에서 실행되는 경우 InteractiveObject 인스턴스가 터치 이벤트 및 호출 핸들러를 수신할 수 있습니다. 일반적으로 터치, 다중 터치 및 동작은 ActionScript의 다른 이벤트와 마찬가지로 처리합니다. ActionScript를 사용한 이벤트 처리에 대한 기본적인 정보는 113페이지의 “이벤트 처리”를 참조하십시오.

그러나 Flash 런타임에서 터치 또는 동작을 해석하려면 터치 또는 다중 터치 입력을 지원하는 하드웨어 및 소프트웨어 환경에서 런타임을 실행해야 합니다. 서로 다른 터치 스크린 유형을 비교한 차트는 507페이지의 “입력 유형 확인”을 참조하십시오. 또한 런타임이 컨테이너 응용 프로그램(예: 브라우저) 내에서 실행되는 경우 해당 컨테이너에서 런타임에 입력을 전달합니다. 경우에 따라서는 현재 하드웨어 및 운영 체제 환경에서 다중 터치를 지원하지만 Flash 런타임을 포함하는 브라우저에서 입력을 해석한 후 런타임에 전달하지 않을 수 있습니다. 또한 해당 브라우저에서 입력을 완전히 무시할 수도 있습니다.

다음 다이어그램에서는 사용자에게서 런타임으로의 입력 흐름을 보여 줍니다.

사용자에게서 Flash Platform 런타임으로의 입력 흐름

터치 응용 프로그램 개발용 ActionScript API에는 런타임 환경에서 터치 또는 다중 터치 입력이 지원되는지 확인할 수 있도록 하는 클래스, 메서드 및 속성이 포함되어 있습니다. 터치 입력 지원을 확인하기 위해 사용하는 API는 터치 이벤트 처리에 대한 "확인 API"입니다.

중요한 개념 및 용어

다음 참조 목록에서는 터치 이벤트 처리 응용 프로그램을 작성하는 데 있어 중요한 용어를 설명합니다.

확인 API 런타임 환경에서 터치 이벤트 및 다양한 입력 모드가 지원되는지 테스트하기 위한 메서드 및 속성입니다.

터치 이벤트 단일 접점을 사용하여 터치 지원 장치에서 수행된 입력 액션입니다.

터치 지점 단일 터치 이벤트에 대한 접점입니다. 장치에서 동작 입력을 지원하지 않는 경우에도 여러 동시 터치 지점을 지원할 수 있습니다.

터치 시퀀스 단일 터치 수명을 나타내는 일련의 이벤트입니다. 이러한 이벤트에는 1개의 시작, 0개 이상의 움직임 및 1개의 종료가 포함됩니다.

다중 터치 이벤트 여러 접점(예: 손가락 두 개 이상)을 사용하여 터치 지원 장치에서 수행된 입력 액션입니다.

동작 이벤트 터치 지원 장치에서 일부 복잡한 움직임을 추적하며 수행된 입력 액션입니다. 예를 들어 손가락 두 개로 화면을 터치하면서 가상 원의 둘레를 동시에 움직여 회전을 나타내는 동작이 있을 수 있습니다.

단계 이벤트 흐름에서 고유한 시점(예: 시작 및 종료)입니다.

스타일러스 터치 스크린과 상호 작용하는 도구입니다. 스타일러스는 사람의 손가락보다 더 정밀하게 입력할 수 있습니다. 일부 장치는 특정 유형 스타일러스의 입력만 인식합니다. 스타일러스 입력을 인식하는 장치는 여러 동시 접점 또는 손가락 접점을 인식하지 못할 수 있습니다.

누르고 두드리기 사용자가 터치 지원 장치를 한 손가락으로 누르면서 다른 손가락 또는 포인팅 장치로 두드리는 특정 유형의 다중 입력 동작입니다. 이 동작은 다중 터치 응용 프로그램에서 마우스 오른쪽 클릭을 시뮬레이션하는 데 종종 사용됩니다.

터치 입력 API 구조

ActionScript 터치 입력 API는 터치 입력 처리가 Flash 런타임의 하드웨어 및 소프트웨어 환경에 의존하는 문제를 해결하기 위해 사용됩니다. 터치 입력 API는 주로 터치 응용 프로그램 개발의 세 가지 요구 사항인 확인, 이벤트, 단계를 다룹니다. 이러한 API를 적절히 사용하면 응용 프로그램 개발 시 대상 장치를 알지 못하더라도 사용자에게 예측 가능하고 응답성이 뛰어난 환경을 제공할 수 있습니다.

확인

확인 API는 런타임에서 하드웨어 및 소프트웨어 환경을 테스트할 수 있는 기능을 제공합니다. 런타임에 채워지는 값을 통해 현재 컨텍스트에서 Flash 런타임에 사용 가능한 터치 입력을 확인할 수 있습니다. 또한 확인 속성 및 메서드 컬렉션을 사용하여, 일부 터치 입력이 해당 환경에서 지원되지 않는 경우 응용 프로그램에서 터치 이벤트가 아닌 마우스 이벤트에 응답하도록 설정할 수 있습니다. 자세한 내용은 530페이지의 "터치 지원 확인"을 참조하십시오.

이벤트

ActionScript는 다른 이벤트와 마찬가지로 이벤트 리스너와 이벤트 핸들러로 터치 입력 이벤트를 관리합니다. 그러나 터치 입력 이벤트 처리의 경우 다음 사항도 고려해야 합니다.

- 터치는 장치 및 운영 체제에 따라 터치 시퀀스 또는 동작으로 달리 해석될 수 있습니다.
- 터치 지원 장치(손가락, 스타일러스 또는 포인팅 장치)에 대한 단일 터치는 항상 마우스 이벤트도 전달합니다. 마우스 이벤트는 MouseEvent 클래스의 이벤트 유형으로 처리할 수 있습니다. 또는 응용 프로그램이 터치 이벤트에만 응답하도록 설계하거나 두 가지 이벤트에 모두 응답하도록 설계할 수 있습니다.
- 응용 프로그램은 여러 동시 터치 이벤트에 응답하고 서로 별도로 처리할 수 있습니다.

일반적으로 확인 API는 응용 프로그램에서 처리하는 이벤트 및 처리 방식을 조건적으로 처리하는 데 사용합니다. 응용 프로그램에서 런타임 환경을 인식하면 사용자가 응용 프로그램과 상호 작용할 때 적절한 핸들러를 호출하거나 정확한 이벤트 객체를 설정할 수 있습니다. 또는 현재 환경에서 특정 입력을 처리할 수 없다는 점을 나타내고 사용자에게 대체 방법 또는 정보를 제공할 수 있습니다. 자세한 내용은 531페이지의 "터치 이벤트 처리" 및 535페이지의 "동작 이벤트 처리"를 참조하십시오.

단계

터치 이벤트 객체는 터치 및 다중 터치 응용 프로그램에 대해 사용자 상호 작용 단계를 추적하는 속성을 가지고 있습니다. 사용자 입력의 시작, 업데이트 또는 종료 단계와 같은 단계를 처리하도록 ActionScript를 작성하여 사용자에게 피드백을 제공할 수 있습니다. 사용자가 화면에서 터치 지점을 터치하고 움직일 때 이벤트 단계에 응답하여 시각적 객체가 변하게 할 수 있습니다. 또는 동작 변화에 따라 동작의 특정 속성을 추적하는 단계를 사용할 수 있습니다.

터치 지점 이벤트의 경우 사용자가 특정 대화형 객체에 머물러 있는 시간을 추적할 수 있습니다. 응용 프로그램은 여러 동시 터치 지점의 단계를 개별적으로 추적하고 각각 별도로 처리할 수 있습니다.

동작의 경우 동작이 발생할 때 동작 변형에 대한 특정 정보를 해석할 수 있습니다. 접점이 화면에서 움직일 때 접점의 좌표를 추적할 수 있습니다.

터치 지원 확인

Flash Player 10.1 이상, Adobe AIR 2 이상

다중 터치 클래스 속성을 사용하여 응용 프로그램 핸들의 터치 입력 범위를 설정합니다. 그런 다음 환경을 테스트하여 ActionScript에서 처리하는 이벤트가 지원되는지 확인합니다. 무엇보다 먼저 응용 프로그램의 터치 입력 유형을 설정합니다. touchPoint, gesture 또는 none(모든 터치 입력을 마우스 클릭으로 해석하고 마우스 이벤트 핸들러만 사용) 중에 선택할 수 있습니다. 그런 다음, Multitouch 클래스의 속성 및 메서드를 사용하여 응용 프로그램에 필요한 터치 입력을 지원하는지 런타임 환경을 확인합니다. 터치 입력 유형(예: 동작 해석 여부)이 지원되는지 런타임 환경을 테스트하고 그에 따라 응답합니다.

참고: Multitouch 클래스 속성은 정적 속성으로서 어떤 클래스 인스턴스에도 속하지 않습니다. 이러한 속성을 Multitouch.property 구문에 사용합니다. 예를 들면 다음과 같습니다.

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

입력 유형 설정

터치 이벤트가 여러 요소 또는 단계를 가질 수 있기 때문에 Flash 런타임에서는 해석할 터치 입력 유형을 알아야 합니다. 단순히 손가락 한 개로 터치 지원 화면을 터치하는 경우 런타임에서 터치 이벤트를 전달하는지, 동작을 기다리는지, 또는 런타임에서 터치를 마우스 누름 이벤트로 추적하는지 등 터치 입력을 지원하는 응용 프로그램에서 Flash 런타임에 대해 처리하는 터치 이벤트 유형을 설정해야 합니다. 런타임에 대해 터치 입력 유형을 설정하려면 Multitouch.inputMode 속성을 사용합니다. 입력 모드는 세 가지 옵션 중 하나가 될 수 있습니다.

없음 터치 이벤트에 대해 특별한 처리가 제공되지 않습니다. Multitouch.inputMode=MultitouchInputMode.NONE으로 설정하고 MouseEvent 클래스를 사용하여 입력을 처리합니다.

단일 터치 지점 모든 터치 입력이 개별적으로 해석되고 모든 터치 지점을 추적 및 처리할 수 있습니다. Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT로 설정하고 TouchEvent 클래스를 사용하여 입력을 처리합니다.

동작 입력 장치 또는 운영 체제에서 입력을 화면에서의 복잡한 손가락 움직임으로 해석하고 움직임을 전체적으로 단일 동작 입력 이벤트에 할당합니다. Multitouch.inputMode=MultitouchInputMode.GESTURE를 설정하고 TransformGestureEvent, PressAndTapGestureEvent 또는 GestureEvent 클래스를 사용하여 입력을 처리합니다.

자세한 내용은 531페이지의 "터치 이벤트 처리"에서 Multitouch.inputMode 속성을 사용하여 터치 이벤트를 처리하기 전에 입력 유형을 설정하는 예제를 확인하십시오.

터치 입력 지원 테스트

Multitouch 클래스의 기타 속성은 응용 프로그램을 현재 환경의 터치 지원에 맞게 세부 조정할 수 있는 값을 제공합니다. Flash 런타임에서 허용된 동시 터치 지점 또는 사용 가능한 동작 수에 대한 값이 채워집니다. 응용 프로그램에서 필요한 터치 이벤트 처리를 지원하지 않는 런타임 환경의 경우 사용자에게 대체 방법을 제공합니다. 예를 들어 마우스 이벤트 처리 또는 현재 환경에서 사용할 수 있는 기능과 사용할 수 없는 기능에 대한 정보를 제공합니다.

또한 키보드, 터치, 마우스 지원에 대한 API를 사용할 수도 있습니다. 자세한 내용은 507페이지의 "입력 유형 확인"을 참조하십시오.

호환성 테스트에 대한 자세한 내용은 539페이지의 "문제 해결"을 참조하십시오.

터치 이벤트 처리

Flash Player 10.1 이상, Adobe AIR 2 이상

기본적인 터치 이벤트는 ActionScript에서 마우스 이벤트를 비롯한 기타 이벤트를 처리하는 것과 마찬가지 방법으로 처리됩니다. TouchEvent 클래스의 이벤트 유형 상수에서 정의된 일련의 터치 이벤트를 수신할 수 있습니다.

참고: 다중 터치 지점 입력(예: 둘 이상의 손가락으로 장치 터치)의 경우 첫 번째 접점이 마우스 이벤트와 터치 이벤트를 전달합니다.

기본적인 터치 이벤트 처리:

1. flash.ui.Multitouch.inputMode 속성을 MultitouchInputMode.TOUCH_POINT로 설정하여 응용 프로그램에서 터치 이벤트를 처리하도록 설정합니다.
2. Sprite 또는 TextField와 같은 InteractiveObject 클래스에서 속성을 상속하는 클래스의 인스턴스에 이벤트 리스너를 추가합니다.
3. 처리할 터치 이벤트 유형을 지정합니다.
4. 이벤트에 응답하여 특정 작업을 수행할 이벤트 핸들러 함수를 호출합니다.

예를 들어 다음 코드는 mySprite에서 그린 사각형을 터치 스크린에서 누를 때 메시지를 표시합니다.

```
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);
}
```

터치 이벤트 속성

이벤트가 발생하면 이벤트 객체가 생성됩니다. TouchEvent 객체에는 터치 이벤트의 위치 및 상태에 대한 정보가 들어 있습니다. 이벤트 객체의 속성을 사용하여 해당 정보를 검색할 수 있습니다.

예를 들어 다음 코드는 TouchEvent 객체 **evt**를 만든 후 이 이벤트 객체의 stageX 속성(터치가 발생한 스테이지 공간의 x 좌표 지점)을 텍스트 필드에 표시합니다.

```
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);
}
```

이벤트 객체를 통해 사용할 수 있는 속성은 TouchEvent 클래스를 참조하십시오.

참고: 런타임 환경에 따라 일부 TouchEvent 속성이 지원되지 않을 수 있습니다. 예를 들어 일부 터치 지원 장치는 사용자가 터치 스크린에 가하는 압력 정도를 감지할 수 없습니다. 즉, 해당 장치에서는 TouchEvent.pressure 속성이 지원되지 않습니다. 응용 프로그램이 제대로 작동하도록 특정 속성 지원 여부를 테스트해야 합니다. 자세한 내용은 539페이지의 "문제 해결"을 참조하십시오.

터치 이벤트 단계

마우스 이벤트에서와 마찬가지로 InteractiveObject에 대해 다양한 단계에 걸쳐 터치 이벤트를 추적합니다. 또한 터치 상호 작용 시작, 중간, 종료에 걸쳐 터치 이벤트를 추적합니다. TouchEvent 클래스는 touchBegin, touchMove 및 touchEnd 이벤트를 처리하기 위한 값을 제공합니다.

예를 들어 touchBegin, touchMove, touchEnd 이벤트를 통해 사용자가 표시 객체를 터치하고 움직일 때 사용자에게 시각적 피드백을 줄 수 있습니다.

```
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;
}
```

참고: 초기 터치 리스너는 mySprite에 추가되지만 터치 이벤트를 움직이고 종료하기 위한 리스너는 추가되지 않습니다. 사용자의 손가락 또는 포인팅 장치가 표시 객체 위에서 움직이는 경우 스테이지에서 터치 이벤트를 계속해서 수신합니다.

터치 지점 ID

TouchEvent.touchPointID 속성은 터치 입력에 응답하는 응용 프로그램을 작성하는 데 필수적인 요소입니다. Flash 런타임은 각 터치 지점에 고유한 touchPointID 값을 할당합니다. 응용 프로그램에서 터치 입력 단계 또는 움직임에 응답할 때마다 touchPointID를 확인한 후 이벤트를 처리합니다. Sprite 클래스의 터치 입력 드래그 메서드는 touchPointID 속성을 매개 변수로 사용하여 올바른 입력 인스턴스가 처리되도록 합니다. touchPointID 속성은 이벤트 핸들러가 올바른 터치 지점에 응답하도록 보장합니다. 이 속성을 사용하지 않을 경우 이벤트 핸들러는 장치에서 터치 이벤트 유형의 모든 인스턴스(모든 touchMove 이벤트 포함)에 응답하므로 예측 불가능한 비헤이비어가 나타납니다. 이 속성은 사용자가 객체를 드래그하는 경우 특히 중요합니다.

touchPointID 속성으로 전체 터치 시퀀스를 관리할 수 있습니다. 터치 시퀀스는 1개의 touchBegin 이벤트, 0개 이상의 touchMove 이벤트, 1개의 touchEnd 이벤트를 포함하며 모두 동일한 touchPointID 값을 가집니다.

다음 예제에서는 터치 움직임 이벤트에 응답하기 전에 변수 **touchMoveID**를 설정하여 올바른 touchPointID 값인지 테스트합니다. 이렇게 하지 않으면 다른 터치 입력도 이벤트 핸들러를 트리거합니다. 움직임 및 종료 단계의 리스너가 표시 객체가 아닌 스테이지에 있는 것을 알 수 있습니다. 사용자의 터치가 표시 객체 경계를 벗어날 경우 스테이지에서 움직임 또는 종료 단계를 수신합니다.

```
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;
}
```

터치와 드래그

Flash Player 10.1 이상, Adobe AIR 2 이상

터치 지점 입력을 지원하는 터치 지원 응용 프로그램을 추가로 지원하기 위해 Sprite 클래스에 두 메서드, Sprite.startTouchDrag() 및 Sprite.stopTouchDrag()가 추가되었습니다. 이러한 메서드는 마우스 이벤트에 대한 Sprite.startDrag() 및 Sprite.stopDrag()와 같은 방식으로 동작합니다. 다른 점이 있다면 Sprite.startTouchDrag() 및 Sprite.stopTouchDrag() 메서드는 모두 touchPointID 값을 매개 변수로 사용한다는 점입니다.

런타임에서는 터치 이벤트에 대한 이벤트 객체에 touchPointID 값을 할당합니다. 이 값을 사용하면 런타임 환경에서 여러 동시 터치 지점을 지원하는 경우(동작을 처리하지 않더라도) 특정 터치 지점에 응답하게 됩니다. touchPointID 속성에 대한 자세한 내용은 533페이지의 “터치 지점 ID”를 참조하십시오.

다음 코드에서는 터치 이벤트에 대한 간단한 시작 드래그 이벤트 핸들러와 중지 드래그 이벤트 핸들러를 보여 줍니다. 변수 **bg**는 **mySprite**를 포함하는 표시 객체입니다.

```
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");
}
```

또한 다음 코드에서는 드래그와 터치 이벤트 단계를 결합한 고급 예제를 보여 줍니다.

```
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;
}
```

동작 이벤트 처리

Flash Player 10.1 이상, Adobe AIR 2 이상

동작 이벤트는 기본 터치 이벤트와 같은 방식으로 처리합니다. TransformGestureEvent 클래스, GestureEvent 클래스 및 PressAndTapGestureEvent 클래스의 이벤트 유형 상수에 정의된 일련의 동작 이벤트를 수신할 수 있습니다.

동작 터치 이벤트 처리:

1. flash.ui.Multitouch.inputMode 속성을 MultitouchInputMode.GESTURE로 설정하여 응용 프로그램이 동작 입력을 처리하도록 설정합니다.
2. Sprite 또는 TextField와 같은 InteractiveObject 클래스에서 속성을 상속하는 클래스의 인스턴스에 이벤트 리스너를 추가합니다.
3. 처리할 동작 이벤트 유형을 지정합니다.

4 이벤트에 응답하여 특정 작업을 수행할 이벤트 핸들러 함수를 호출합니다.

예를 들어 다음 코드는 **mySprite**에서 그린 사각형을 터치 스크린에서 스와이프할 때 메시지를 표시합니다.

```
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);
}
```

두 손가락 두드리기 이벤트는 같은 방식으로 처리되지만 GestureEvent 클래스를 사용합니다.

```
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);
}
```

누르고 두드리기 이벤트도 같은 방식으로 처리되지만 PressAndTapGestureEvent 클래스를 사용합니다.

```
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);
}
```

참고: 런타임 환경에 따라 GestureEvent, TransformGestureEvent 및 PressAndTapGestureEvent 이벤트 유형 중 일부가 지원되지 않을 수 있습니다. 예를 들어 일부 터치 지원 장치에서는 여러 손가락 스와이프를 감지할 수 없습니다. 따라서 이러한 장치에서는 InteractiveObject gestureSwipe 이벤트가 지원되지 않습니다. 응용 프로그램이 제대로 작동하도록 특정 이벤트 지원 여부를 테스트해야 합니다. 자세한 내용은 539페이지의 “문제 해결”을 참조하십시오.

동작 이벤트 속성

동작 이벤트는 기본적인 터치 이벤트보다 이벤트 속성 범위가 작습니다. 동작 이벤트는 이벤트 핸들러 함수의 이벤트 객체를 통해 터치 이벤트와 같은 방식으로 액세스합니다.

예를 들어 다음 코드에서는 사용자가 회전 동작을 수행할 때 **mySprite**를 회전합니다. 텍스트 필드에는 마지막 동작 이후 회전 수가 표시됩니다. 이 코드를 테스트할 때는 여러 번 회전하여 값이 변하는지 확인해야 합니다.

```
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();
}
```

참고: 런타임 환경에 따라 일부 TransformGestureEvent 속성이 지원되지 않을 수 있습니다. 예를 들어 일부 터치 지원 장치에서는 화면의 동작 회전을 감지할 수 없습니다. 즉, 해당 장치에서는 TransformGestureEvent.rotation 속성이 지원되지 않습니다. 응용 프로그램이 제대로 작동하도록 특정 속성 지원 여부를 테스트해야 합니다. 자세한 내용은 539페이지의 “문제 해결”을 참조하십시오.

동작 단계

또한 동작 이벤트는 여러 단계에 걸쳐 추적할 수 있으므로 동작이 발생할 때 속성을 추적할 수 있습니다. 예를 들어 스와이프 동작을 통해 객체가 움직일 때 **x** 좌표를 추적할 수 있습니다. 이러한 값을 사용하여 스와이프가 완료된 후 경로에 있는 모든 지점을 통과하는 선을 그리거나 이동 동작을 통해 화면에서 드래그될 때 표시 객체를 시각적으로 변경할 수 있습니다. 이동 동작이 완료되면 객체를 다시 변경합니다.

```
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;
}
}
```

참고: 업데이트 단계의 빈도는 런타임 환경에 따라 달라집니다. 일부 운영 체제 및 하드웨어 조합에서는 업데이트가 전혀 제공되지 않습니다.

단순한 동작 이벤트의 경우 동작 단계 "all"

일부 동작 이벤트 객체는 동작 이벤트의 개별 단계를 추적하지 않고 이벤트 객체의 단계 속성을 all 값으로 채웁니다. 스와이프 및 두 손가락 두드리기와 같은 단순한 동작은 이벤트를 여러 단계로 추적하지 않습니다. gestureSwipe 또는 gestureTwoFingerTap 이벤트를 수신하는 InteractiveObject에 대한 이벤트 객체의 phase 속성은 이벤트가 전달된 후 항상 all입니다.

```
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"
}
```

문제 해결

Flash Player 10.1 이상, Adobe AIR 2 이상

터치 입력에 대한 하드웨어 및 소프트웨어 지원은 급속도로 변화하고 있습니다. 이 참조 설명에서는 다중 터치를 지원하는 운영 체제 및 소프트웨어 조합을 갖춘 모든 장치에 대한 목록은 제공하지 않습니다. 그러나 응용 프로그램이 다중 터치를 지원하는 장치에 배포되어 있는지 확인 API를 통해 확인하는 지침과 ActionScript 코드의 문제를 해결하는 팁을 제공합니다.

Flash 런타임은 장치, 운영 체제 또는 포함 소프트웨어(예: 브라우저)가 런타임에 전달하는 터치 이벤트에 응답합니다. 소프트웨어 환경에 대한 이러한 의존성 때문에 다중 터치 호환성을 문서화하기가 어렵습니다. 일부 장치의 경우에는 동작 또는 터치 움직임을 다르게 해석합니다. 회전이 두 손가락을 동시에 회전하는 것으로 정의될 수도 있고 한 손가락으로 화면에 원을 그리는 것으로 정의될 수도 있습니다. 하드웨어 및 소프트웨어 환경에 따라 회전 동작이 둘 중 하나로 정의될 수도 있고 완전히 다르게 정의될 수도 있습니다. 즉, 장치에서 운영 체제에 사용자 입력을 알리고 운영 체제에서 해당 정보를 런타임에 전달합니다. 런타임이 브라우저 내에 있는 경우 브라우저 소프트웨어에서 동작 또는 터치 이벤트를 해석한 후 해당 입력을 런타임에 전달하지 않는 경우가 있습니다. 이러한 비헤이비어는 단축키 비헤이비어와 유사합니다. 브라우저 내에서 Flash Player가 특정 작업을 수행하도록 특정 키 조합을 사용하지만 브라우저에서 여전히 메뉴를 열 수 있습니다.

개별 API와 클래스는 특정 운영 체제와 호환되지 않을 수도 있습니다. 다음 페이지에서는 Multitouch 클래스부터 개별 API 항목을 살펴볼 수 있습니다.
http://help.adobe.com/ko_KR/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html

일반적인 동작 및 터치 설명은 다음과 같습니다.

이동 손가락을 왼쪽에서 오른쪽 또는 오른쪽에서 왼쪽 방향으로 움직입니다. 일부 장치에서는 이동을 위해 두 손가락을 사용해야 합니다.

회전 두 손가락으로 누른 다음 표면에서 가상 원을 동시에 추적하듯이 원 둘레를 따라 함께 움직입니다. 두 손가락 터치 지점 사이의 중간점에 pivot 점이 설정됩니다.

스와이프 세 손가락을 왼쪽에서 오른쪽 또는 오른쪽에서 왼쪽 방향이나 위쪽에서 아래쪽 또는 아래쪽에서 위쪽 방향으로 신속하게 움직입니다.

확대/축소 두 손가락으로 누른 다음 확대하려면 손가락 사이를 멀리 벌리고 축소하려면 손가락 사이를 가깝게 모읍니다.

누르고 두드리기 한 손가락으로 움직이거나 누른 다음 다른 손가락으로 표면을 두드립니다.

각 장치에는 장치가 지원하는 동작과 장치에서 각 동작을 수행하는 방법에 대한 자체 설명서가 있습니다. 운영 체제에 따라 달라지지만 일반적으로 사용자는 각 동작 사이에 장치에서 모든 손가락을 완전히 떼야 합니다.

응용 프로그램이 터치 이벤트 또는 동작에 응답하지 않는 경우 다음을 테스트합니다.

1. InteractiveObject 클래스에서 상속된 객체 클래스에 연결된 터치 또는 동작 이벤트에 대한 이벤트 리스너가 있습니까? InteractiveObject 인스턴스만 터치 및 동작 이벤트를 수신할 수 있습니다.
2. 응용 프로그램을 Flash Professional CS5 내에서 테스트하고 있습니까? 이 경우 Flash Professional에서 상호 작용을 차단할 수 있으므로 응용 프로그램을 제작하여 테스트해 봅니다.
3. 먼저 단순하게 시작해서 제대로 작동하는지 확인합니다. 다음 코드 예제는 Multitouch.inputMode에 대한 API 항목에서 가져온 것입니다.

```
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);
}
```

사각형을 두드립니다. 이 예제가 작동하는 경우 해당 환경이 단순한 두드리기를 지원하는 것입니다. 그런 다음 좀더 복잡한 처리를 시도해 볼 수 있습니다.

동작 지원 테스트는 좀더 복잡합니다. 개별 장치나 운영 체제에서 동작 입력의 모든 조합을 지원하거나, 그렇지 않을 경우 어떠한 조합도 지원하지 않습니다.

다음은 간단한 확대/축소 동작 테스트입니다.

```
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";
}
```

장치에서 확대/축소 동작을 수행하고 텍스트 필드가 Zoom is supported라는 메시지로 채워지는지 확인합니다. 스테이지에 이벤트 리스너가 추가되어 테스트 응용 프로그램의 모든 부분에서 동작을 수행할 수 있습니다.

다음은 간단한 패닝 동작 테스트입니다.

```
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";
}
```

장치에서 패닝 동작을 수행하고 텍스트 필드가 Pan is supported라는 메시지로 채워지는지 확인합니다. 스테이지에 이벤트 리스너가 추가되어 테스트 응용 프로그램의 모든 부분에서 동작을 수행할 수 있습니다.

두 동작을 모두 지원하는 운영 체제 및 장치 조합도 있고 일부만 지원하는 운영 체제 및 장치 조합도 있으며 아무것도 지원하지 않는 운영 체제 및 장치 조합도 있습니다. 확실하게 하기 위해 응용 프로그램의 배포 환경을 테스트합니다.

알려진 문제

다음은 터치 입력과 관련된 알려진 문제입니다.

1 Windows Mobile 운영 체제의 Mobile Internet Explorer는 SWF 파일 내용을 자동으로 확대/축소합니다.

이러한 Internet Explorer 확대/축소 비헤이비어는 SWF 파일을 호스팅하는 HTML 페이지에 다음을 추가하면 재정의됩니다.

```
<head>
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">
</head>
```

2 Windows 7(및 일부 다른 운영 체제)에서 사용자는 각 동작 사이에 포인팅 장치(또는 손가락)를 화면에서 완전히 떼어야 합니다. 예를 들어 이미지를 회전 및 확대/축소하려면 다음과 같이 합니다.

- 회전 동작을 수행합니다.
- 화면에서 손가락을 떼어 냅니다.
- 손가락을 다시 화면에 놓고 확대/축소 동작을 수행합니다.

3 Windows 7(및 기타 운영 체제)에서는 사용자가 동작을 매우 빨리 수행하는 경우 회전 및 확대/축소 동작으로 "업데이트" 단계가 생성되지 않을 수도 있습니다.

4 Windows 7 Starter Edition은 다중 터치를 지원하지 않습니다. 자세한 내용은 AIR Labs Forum(http://forums.adobe.com/thread/579180?tstart=0)을 참조하십시오.

5 Mac OS 10.5.3 이상에서는 하드웨어가 동작 이벤트를 지원하지 않는 경우에도 Multitouch.supportsGestureEvents 값이 항상 true입니다.

33장: 복사하여 붙여넣기

Flash Player 10 이상, Adobe AIR 1.0 이상

클립보드 API의 클래스를 사용하여 시스템 클립보드에서 정보를 복사합니다. Adobe® Flash® Player 또는 Adobe® AIR®에서 실행되는 응용 프로그램에서 또는 해당 응용 프로그램으로 전송할 수 있는 데이터 형식은 다음과 같습니다.

- 텍스트
- HTML 서식 텍스트
- 서식 있는 포맷의 데이터
- 직렬화된 객체
- 객체 참조(원본 응용 프로그램 내에서만 유효)
- 비트맵(AIR만 해당)
- 파일(AIR만 해당)
- URL 문자열(AIR만 해당)

복사하여 붙여넣기의 기초

Flash Player 10 이상, Adobe AIR 1.0 이상

복사하여 붙여넣기 API에는 다음 클래스가 포함됩니다.

패키지	클래스
flash.desktop	• Clipboard • ClipboardFormats • ClipboardTransferMode

정적 Clipboard.generalClipboard 속성은 운영 체제 클립보드를 나타냅니다. Clipboard 클래스는 Clipboard 객체에서 데이터를 읽고 쓰는 메서드를 제공합니다.

HTMLLoader 클래스(AIR의 경우)와 TextField 클래스는 일반적으로 복사하여 붙여넣기에 사용되는 키보드 단축키의 기본 비헤이비어를 구현합니다. 사용자 정의 구성 요소에 대한 복사하여 붙여넣기 단축키 비헤이비어를 구현하기 위해 이러한 키 입력을 직접 수신할 수 있습니다. 키 입력에 직접 응답하기 위해 키 상당 어구와 함께 기본 메뉴 명령을 사용할 수도 있습니다.

다른 응용 프로그램이 데이터를 이해하고 사용할 수 있는 기능을 향상시키기 위해 단일 Clipboard 객체에서 같은 정보를 다양하게 표현할 수 있습니다. 예를 들어 이미지를 이미지 데이터, 직렬화된 Bitmap 객체 및 파일로서 포함할 수 있습니다. 한 형식의 데이터를 읽을 때까지 해당 형식이 실제로 생성되지 않도록 한 형식의 데이터 렌더링을 지연할 수 있습니다.

시스템 클립보드에서 읽고 쓰기

Flash Player 10 이상, Adobe AIR 1.0 이상

운영 체제 클립보드를 읽으려면 다음과 같이 Clipboard.generalClipboard 객체의 getData() 메서드를 호출하며 이때 읽을 형식의 이름을 전달합니다.

```
import flash.desktop.Clipboard;
import flash.desktop.ClipboardFormats;

if(Clipboard.generalClipboard.hasFormat(ClipboardFormats.TEXT_FORMAT)){
    var text:String = Clipboard.generalClipboard.getData(ClipboardFormats.TEXT_FORMAT);
}
```

참고: Flash Player나 AIR의 비응용 프로그램 샌드박스에서 실행되는 내용은 paste 이벤트에 대한 이벤트 핸들러에서만 getData() 메서드를 호출할 수 있습니다. 즉, AIR 응용 프로그램 샌드박스에서 실행되는 코드만이 paste 이벤트 핸들러의 외부에서 getData() 메서드를 호출할 수 있습니다.

클립보드에 기록하려면 데이터를 하나 이상의 형식으로 Clipboard.generalClipboard 객체에 추가합니다. 같은 형식의 기존 데이터가 있으면 자동으로 덮어씁니다. 그러나 시스템 클립보드에 새 데이터를 쓰려면 먼저 클립보드를 지워 관련이 없는 다른 형식의 데이터도 함께 삭제하는 것이 좋습니다.

```
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);
```

참고: Flash Player나 AIR의 비응용 프로그램 샌드박스에서 실행되는 내용은 키보드 또는 마우스 이벤트와 같은 사용자 이벤트나 copy 또는 cut 이벤트에 대한 이벤트 핸들러에서만 setData() 메서드를 호출할 수 있습니다. 즉, AIR 응용 프로그램 샌드박스에서 실행되는 코드만이 사용자 이벤트 핸들러의 외부에서 setData() 메서드를 호출할 수 있습니다.

AIR의 HTML 복사하여 붙여넣기

Adobe AIR 1.0 이상

Adobe AIR의 HTML 환경에서는 복사하여 붙여넣을 수 있도록 자체 이벤트 및 기본 비헤이비어 집합을 제공합니다. 응용 프로그램 샌드박스에서 실행되는 코드만 AIR Clipboard.generalClipboard 객체를 통해 시스템 클립보드에 직접 액세스할 수 있습니다. 비 응용 프로그램 샌드박스의 JavaScript 코드는 HTML 문서의 한 요소에서 전달한 copy 또는 paste 이벤트 중 하나에 응답하여 전달된 이벤트 객체를 통해 클립보드에 액세스할 수 있습니다.

복사하여 붙여넣기 이벤트는 copy, cut 및 paste입니다. 이러한 이벤트에 대해 전달된 객체는 clipboardData 속성을 통해 클립보드에 대한 액세스를 제공합니다.

기본 비헤이비어

Adobe AIR 1.0 이상

기본적으로 AIR은 키보드 단축키나 컨텍스트 메뉴 중 하나로 생성될 수 있는 복사 명령에 응답하여 선택한 항목을 복사합니다. 편집 가능한 영역 내에서 AIR은 자르기 명령에 응답하여 텍스트를 자르거나 붙여넣기 명령에 응답하여 커서나 선택 영역에 텍스트를 붙여넣습니다.

기본 비헤이비어를 차단하기 위해 이벤트 핸들러는 전달된 이벤트 객체의 preventDefault() 메서드를 호출할 수 있습니다.

이벤트 객체의 clipboardData 속성 사용

Adobe AIR 1.0 이상

copy 또는 paste 이벤트 중 하나의 결과로 전달된 이벤트 객체의 clipboardData 속성을 사용하여 클립보드 데이터를 읽고 쓸 수 있습니다.

copy 또는 cut 이벤트를 처리할 때 클립보드에 기록하려면 복사할 데이터와 MIME 유형을 전달하여 clipboardData 객체의 setData() 메서드를 사용합니다.

```
function customCopy(event){
    event.clipboardData.setData("text/plain", "A copied string.");
}
```

붙여넣는 데이터에 액세스하기 위해 데이터 형식의 MIME 유형을 전달하여 clipboardData 객체의 getData() 메서드를 사용할 수 있습니다. 사용할 수 있는 형식은 types 속성에서 보고합니다.

```
function customPaste(event){
    var pastedData = event.clipboardData("text/plain");
}
```

getData() 메서드 및 types 속성은 paste 이벤트에서 전달한 이벤트 객체에서만 액세스할 수 있습니다.

다음 예제에서는 HTML 페이지의 기본 복사하여 붙여넣기 비헤이비어를 재정의하는 방법을 보여 줍니다. copy 이벤트 핸들러는 복사한 텍스트를 기울임체로 만들어 클립보드에 HTML 텍스트로 복사합니다. cut 이벤트 핸들러는 선택한 데이터를 클립보드에 복사하고 문서에서 제거합니다. paste 핸들러는 클립보드 내용을 HTML로 삽입하고 삽입 스타일을 굵은 텍스트로 지정합니다.

```
 <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>
```

클립보드 데이터 형식

Flash Player 10 이상, Adobe AIR 1.0 이상

클립보드 형식은 Clipboard 객체에 있는 데이터를 설명합니다. Flash Player 또는 AIR에서는 ActionScript 데이터 형식과 시스템 클립보드 형식 간에 표준 데이터 형식을 자동으로 변환합니다. 또한 ActionScript 기반 응용 프로그램 내에서와 ActionScript 기반 용 프로그램 간에 응용 프로그램 정의 형식을 사용하여 응용 프로그램 객체를 전송할 수 있습니다.

Clipboard 객체에는 같은 정보를 다른 형식으로 표현한 내용이 포함될 수 있습니다. 예를 들어 스프라이트를 나타내는 Clipboard 객체에는 동일한 응용 프로그램 내에서 사용하기 위한 참조 형식, Flash Player 또는 AIR에서 실행되는 다른 응용 프로그램에서 사용하기 위한 직렬화된 형식, 이미지 편집기에서 사용하기 위한 비트맵 형식 및 파일 목록 형식이 포함될 수 있으며 스프라이트 표현을 파일 시스템에 복사 또는 드래그하기 위한 PNG 파일 인코딩용 지연 렌더링도 포함될 수 있습니다.

표준 데이터 형식

Flash Player 10 이상, Adobe AIR 1.0 이상

표준 형식 이름을 정의하는 상수는 ClipboardFormats 클래스에서 제공됩니다.

상수	설명
TEXT_FORMAT	텍스트 형식 데이터와 ActionScript String 클래스는 서로 변환됩니다.
HTML_FORMAT	HTML 마크업이 있는 텍스트입니다.
RICH_TEXT_FORMAT	서식 있는 텍스트 데이터와 ActionScript ByteArray 클래스는 서로 변환됩니다. RTF 마크업은 어떤 식으로든 해석되거나 변환되지 않습니다.
BITMAP_FORMAT	(AIR만 해당) 비트맵 형식 데이터는 ActionScript BitmapData 클래스로 변환하거나 그 반대로 변환할 수 있습니다.
FILE_LIST_FORMAT	(AIR만 해당) 파일 목록 형식 데이터는 ActionScript File 객체의 배열로 변환하거나 그 반대로 변환할 수 있습니다.
URL_FORMAT	(AIR만 해당) URL 형식 데이터는 ActionScript String 클래스로 변환하거나 그 반대로 변환할 수 있습니다.

AIR 응용 프로그램에서 호스팅되는 HTML 내용에서 copy, cut 또는 paste 이벤트에 응답하여 데이터를 복사하고 붙여넣을 때 ClipboardFormat 문자열 대신 MIME 유형을 사용해야 합니다. 유효한 데이터 MIME 유형은 다음과 같습니다.

MIME 유형	설명
텍스트	"text/plain"
URL	"text/uri-list"
비트맵	"image/x-vnd.adobe.air.bitmap"
파일 목록	"application/x-vnd.adobe.air.file-list"

참고: 서식 있는 포맷의 데이터는 HTML 내용 내에서 paste 이벤트의 결과로 전달된 이벤트 객체의 clipboardData 속성에서 사용할 수 없습니다.

사용자 정의 데이터 형식

Flash Player 10 이상, Adobe AIR 1.0 이상

응용 프로그램에서 정의한 사용자 정의 형식을 사용하여 객체를 참조 또는 직렬화된 복사본으로 전송할 수 있습니다. 참조는 같은 응용 프로그램 안에서만 유효합니다. 직렬화된 객체는 응용 프로그램 간에 전송할 수 있지만 직렬화되고 역직렬화되었을 때 유효한 상태로 있는 객체에서만 사용할 수 있습니다. 대개 객체의 속성이 간단한 유형이거나 직렬화된 객체인 경우 객체를 직렬화할 수 있습니다.

Clipboard 객체에 직렬화된 객체를 추가하려면 Clipboard.setData() 메서드를 호출할 때 **serializable** 매개 변수를 true로 설정합니다. 형식 이름은 표준 형식 중 하나이거나 응용 프로그램에서 정의된 임의의 문자열일 수 있습니다.

전송 모드

Flash Player 10 이상, Adobe AIR 1.0 이상

사용자 정의 데이터 형식을 사용하여 클립보드에 객체를 쓸 경우 클립보드에서 해당 객체 데이터를 읽을 때는 참조나 원본 객체의 직렬화된 복사본으로 읽을 수 있습니다. AIR는 객체를 참조 또는 직렬화된 복사본으로 전송할 지 여부를 결정하는 4개의 전송 모드를 정의합니다.

전송 모드	설명
ClipboardTransferModes.ORIGINAL_ONLY	참조만 반환됩니다. 참조를 사용할 수 없는 경우 null 값이 반환됩니다.
ClipboardTransferModes.ORIGINAL_PREFFERED	가능한 경우 참조가 반환됩니다. 그렇지 않은 경우에는 직렬화된 복사본이 반환됩니다.
ClipboardTransferModes.CLONE_ONLY	직렬화된 복사본만 반환됩니다. 직렬화된 복사본을 사용할 수 없으면 null 값이 반환됩니다.
ClipboardTransferModes.CLONE_PREFFERED	가능한 경우 직렬화된 복사본이 반환됩니다. 그렇지 않은 경우 참조가 반환됩니다.

사용자 정의 데이터 형식 읽기 및 쓰기

Flash Player 10 이상, Adobe AIR 1.0 이상

클립보드에 객체를 쓸 때 **format** 매개 변수에는 예약된 접두어인 air: 또는 flash:로 시작하지 않는 모든 문자열을 이용할 수 있습니다. 객체를 읽을 형식과 같은 문자열을 사용합니다. 다음 예제에서는 객체를 클립보드에 읽고 쓰는 방법에 대해 설명합니다.

```
public function createClipboardObject(object:Object):Clipboard{
    var transfer:Clipboard = Clipboard.generalClipboard;
    transfer.setData("object", object, true);
}
```

드롭 또는 붙여넣기 작업 후 Clipboard 객체에서 직렬화된 객체를 추출하려면 동일한 형식 이름과 CLONE_ONLY 또는 CLONE_PREFFERED 전송 모드를 사용합니다.

```
var transfer:Object = clipboard.getData("object", ClipboardTransferMode.CLONE_ONLY);
```

참조는 항상 Clipboard 객체에 추가됩니다. 드롭 또는 붙여넣기 작업 후에 Clipboard 객체에서 참조를 추출하려면 직렬화된 복사본 대신 ORIGINAL_ONLY 또는 ORIGINAL_PREFFERED 전송 모드를 사용합니다.

```
var transferredObject:Object =
    clipboard.getData("object", ClipboardTransferMode.ORIGINAL_ONLY);
```

Clipboard 객체가 현재 응용 프로그램에서 발생한 경우에만 참조가 유효합니다. ORIGINAL_PREFFERED 전송 모드를 사용하여 참조가 사용 가능할 때 참조에 액세스하고 참조가 사용 가능하지 않을 때 직렬화된 클론에 액세스합니다.

지연된 렌더링

Flash Player 10 이상, Adobe AIR 1.0 이상

데이터 형식 만들기가 계산상으로 광범위한 경우 요청한 데이터를 제공하는 함수를 제공하여 지연된 렌더링을 사용할 수 있습니다. 드롭 또는 붙여넣기 작업의 받는 사람이 지연된 형식의 데이터를 요청하는 경우에만 이 함수가 호출됩니다.

렌더링 함수는 setDataHandler() 메서드를 사용하여 Clipboard 객체에 추가됩니다. 이 함수는 적절한 형식의 데이터를 반환해야 합니다. 예를 들어 setDataHandler(ClipboardFormat.TEXT_FORMAT, writeText)를 호출한 경우 writeText() 함수가 문자열을 반환해야 합니다.

Clipboard 객체에 동일한 유형의 데이터 형식이 setData() 메서드를 사용하여 추가되어 있는 경우에는 지연 버전보다 해당 데이터가 우선하므로 렌더링 함수가 호출되지 않습니다. 같은 클립보드 데이터에 두 번째 액세스하는 경우 렌더링 함수가 다시 호출될 수도 있고 호출되지 않을 수도 있습니다.

참고: Mac OS X의 경우 지연 렌더링은 사용자 정의 데이터 형식에 대해서만 작동합니다. 표준 데이터 형식의 경우에는 렌더링 함수가 즉시 호출됩니다.

지연된 렌더링 함수를 사용하여 텍스트 붙여넣기

Flash Player 10 이상, Adobe AIR 1.0 이상

다음 예제에서는 지연된 렌더링 함수를 구현하는 방법에 대해 설명합니다.

사용자가 복사 버튼을 누르면 응용 프로그램에서는 이전 클립보드 작업에서 복사된 데이터가 남아 있지 않도록 시스템 클립보드를 지웁니다. 그런 다음 setDataHandler() 메서드가 renderData() 함수를 클립보드 렌더러로 설정합니다.

사용자가 대상 텍스트 필드의 컨텍스트 메뉴에서 붙여넣기 명령을 선택하면 응용 프로그램에서는 클립보드에 액세스하여 대상 텍스트를 설정합니다. 문자열 대신 함수를 사용하여 클립보드의 텍스트 데이터 형식을 설정한 후에는 클립보드에서 renderData() 함수가 호출됩니다. renderData() 함수는 소스 텍스트의 텍스트를 반환하고 이 텍스트가 대상 텍스트에 할당됩니다.

[붙여넣기] 버튼을 누르기 전에 소스 텍스트를 편집하는 경우 [복사] 버튼을 누른 후에 편집이 발생하더라도 붙여넣은 텍스트에서 편집이 반영됩니다. 이는 붙여넣기 버튼을 누를 때까지 렌더링 함수가 소스 텍스트를 복사하지 않기 때문입니다. 실제 응용 프로그램에서 지연된 렌더링을 사용할 때 이 문제를 방지하기 위해 어떤 식으로든 소스 데이터를 저장하거나 보호할 수 있습니다.

Flash 예제

```
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 예제

```
<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>
```

34장: 가속도계 입력

Flash Player 10.1 이상, Adobe AIR 2 이상

Accelerometer 클래스는 장치의 모션 센서가 감지하는 작업에 기반하여 이벤트를 전달합니다. 이 데이터는 3차원 축을 따라 움직이는 장치의 위치 또는 이동을 나타냅니다. 장치가 이동하면 센서가 이 이동을 감지하고 장치의 가속 좌표를 반환합니다. Accelerometer 클래스는 가속도계가 지원되는지 여부를 쿼리하고 가속 이벤트를 전달하는 속도도 설정하는 메서드를 제공합니다.

가속도계 축은 장치의 실제 방향이 아닌 표시 방향을 기준으로 정규화됩니다. 장치에서 표시의 방향이 재조정되면 가속도계 축의 방향도 재조정됩니다. 따라서 사용자가 전화기를 정상적으로 들고 있는 경우 회전 방향에 상관없이 y축은 항상 대략적으로 수직 상태를 유지합니다. 예를 들어 자동 방향이 해제되어 있는 경우 브라우저에서 SWF 내용이 전체 화면 모드로 되어 있으면 장치가 회전할 때 가속도계 축의 방향이 재조정되지 않습니다.

기타 도움말 항목

flash.sensors.Accelerometer

flash.events.AccelerometerEvent

가속도계 지원 확인

Accelerometer.isSupported 속성을 통해 런타임 환경에서 이 기능을 사용할 수 있는지 여부를 테스트합니다.

```
if (Accelerometer.isSupported)
{
    // Set up Accelerometer event listeners and code.
}
```

Accelerometer 클래스 및 해당 멤버는 각 API 항목에 대해 나열된 런타임 버전에 액세스할 수 있습니다. 그러나 이 기능의 사용 가능 여부는 런타임에서의 현재 환경에 따라 결정됩니다. 예를 들어 Flash Player 10.1의 Accelerometer 클래스 속성을 사용하여 코드를 컴파일할 수 있지만 사용자 장치에서 Accelerometer 기능을 사용할 수 있는지 여부를 테스트하려면 Accelerometer.isSupported 속성을 사용해야 합니다. 런타임에 Accelerometer.isSupported가 true로 반환되면 가속도계가 현재 지원되는 것입니다.

가속도계 변경 감지

가속도계 센서를 사용하려면 Accelerometer 객체를 인스턴스화하고 이 객체가 전달하는 update 이벤트에 대해 등록합니다. update 이벤트는 Accelerometer 이벤트 객체로서 네 가지 속성을 가지며 모두 숫자 값입니다.

- accelerationX - X축을 기준으로 한 가속 값(g 단위)입니다. X축은 바로 선 위치일 때 장치의 왼쪽에서 오른쪽으로 진행합니다. 바로 선 위치는 장치의 윗면이 위를 향하는 경우를 가리킵니다. 장치가 오른쪽을 향해 움직일 경우 가속은 양수입니다.
- accelerationY - Y축을 기준으로 한 가속 값(g 단위)입니다. Y축은 바로 선 위치일 때 장치의 아래쪽에서 위쪽으로 진행합니다. 바로 선 위치는 장치의 윗면이 위를 향하는 경우를 가리킵니다. 장치가 이 축을 기준으로 위쪽으로 움직일 경우 가속은 양수입니다.
- accelerationZ - Z축을 기준으로 한 가속 값(g 단위)입니다. Z축은 장치 면과 직각을 이룹니다. 장치의 윗면이 위를 향하도록 움직일 경우 가속은 양수입니다. 장치의 윗면이 아래를 향할 경우 가속은 음수입니다.
- timestamp - 런타임이 초기화된 이후 이벤트가 발생하는 시점의 밀리초입니다.

1g는 중력으로 인한 표준 가속도로, 대략 9.8m/sec^{2}입니다.

다음은 텍스트 필드에 가속도계 데이터를 표시하는 기본적인 예제입니다.

```
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()
}
```

이 예제 코드를 사용하려면 먼저 accTextField 텍스트 필드를 만들어 표시 목록에 추가해야 합니다.

Accelerometer 객체의 setRequestedUpdateInterval() 메서드를 호출하여 가속도계 이벤트의 원하는 시간 간격을 조정할 수 있습니다. 이 메서드는 interval 매개 변수 하나를 사용합니다. 이 매개 변수는 밀리초 단위의 요청된 업데이트 간격입니다.

```
var accl:Accelerometer;
accl = new Accelerometer();
accl.setRequestedUpdateInterval(1000);
```

가속도계 업데이트의 실제 시간 간격은 이 값보다 크거나 작을 수 있습니다. 업데이트 간격을 변경하면 등록된 모든 리스너에 영향을 줍니다. setRequestedUpdateInterval() 메서드를 호출하지 않으면 응용 프로그램에서 장치의 기본 간격에 따라 업데이트를 받습니다.

가속도계 데이터는 어느 정도 부정확할 수 있습니다. 최근 데이터 이동 평균을 이용하여 가속도계 데이터를 다듬을 수 있습니다. 다음 예제와 같이 현재 판독값에 최근 가속도계 판독값을 고려하여 반올림한 결과를 얻을 수 있습니다.

```
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));
}
```

그러나 이러한 이동 평균은 가속도계 업데이트 간격이 작은 경우에만 사용하는 것이 좋습니다.

35장: AIR에서 드래그 앤 드롭

Adobe AIR 1.0 이상

Adobe® AIR™ 드래그 앤 드롭 API의 클래스를 사용하여 사용자 인터페이스 드래그 앤 드롭 동작을 지원합니다. 이러한 의미의 동작은 정보를 복사, 이동 또는 링크하려는 의도를 갖는 운영 체제 및 사용자 응용 프로그램을 매개로 하는 사용자의 조치입니다. 드래그아웃 동작은 사용자가 구성 요소나 응용 프로그램 바깥으로 객체를 드래그할 때 발생합니다. 드래그인 동작은 사용자가 외부에서 구성 요소 또는 응용 프로그램 안으로 객체를 드래그할 때 발생합니다.

끌어 놓기 API를 사용하면 사용자는 응용 프로그램 간에 그리고 응용 프로그램 내 구성 요소 간에 데이터를 드래그할 수 있습니다. 다음과 같은 전송 포맷이 지원됩니다.

- 비트맵
- 파일
- HTML 서식 텍스트
- 텍스트
- 서식 있는 포맷의 데이터
- URL
- 파일 프로미스
- 직렬화된 객체
- 객체 참조(원본 응용 프로그램 내에서만 유효)

AIR에서 드래그 앤 드롭의 기초

Adobe AIR 1.0 이상

AIR 응용 프로그램의 드래그 앤 드롭 사용에 대한 간략한 설명과 코드 예제를 보려면 Adobe Developer Connection의 다음 퀵 스타트 문서를 참조하십시오.

- 드래그 앤 드롭 및 복사하여 붙여넣기 지원(Flex)
- 드래그 앤 드롭 및 복사하여 붙여넣기 지원(Flash)

끌어 놓기 API에는 다음과 같은 클래스가 포함됩니다.

패키지	클래스
flash.desktop	• NativeDragManager • NativeDragOptions • Clipboard • URLFilePromise • IFilePromise 끌어 놓기 API와 함께 사용되는 상수는 다음과 같은 클래스에서 정의됩니다. • NativeDragActions • ClipboardFormat • ClipboardTransferModes
flash.events	NativeDragEvent

끌어 놓기 동작 단계

끌어 놓기 동작에는 세 단계가 있습니다.

시작 사용자는 마우스 버튼을 누른 채 구성 요소 또는 구성 요소 내 항목에서 드래그함으로써 끌어 놓기 작업을 시작합니다. 드래그된 항목의 소스 구성 요소가 일반적으로 드래그 시작자로 지정되며 nativeDragStart 및 nativeDragComplete 이벤트를 전달합니다. Adobe AIR 응용 프로그램은 mouseDown 또는 mouseMove 이벤트에 대한 응답으로 NativeDragManager.doDrag() 메서드를 호출함으로써 드래그 작업을 시작합니다.

AIR 응용 프로그램 외부에서 드래그 작업이 시작되는 경우 nativeDragStart 또는 nativeDragComplete 이벤트를 전달하는 시작자 객체가 없습니다.

드래그 마우스 버튼을 누른 채 사용자는 마우스 커서를 다른 구성 요소, 응용 프로그램 또는 데스크톱으로 이동합니다. 드래그가 진행 중인 동안 시작자 객체는 nativeDragUpdate 이벤트를 전달합니다. 단, Linux 환경인 경우 AIR에서 이 이벤트가 전달되지 않습니다. 사용자가 AIR 응용 프로그램에서 가능한 드롭 대상 위에서 마우스를 움직이면 드롭 대상은 nativeDragEnter 이벤트를 전달합니다. 이벤트 핸들러는 이벤트 객체를 검사하여 드래그된 데이터가 대상이 허용하는 포맷으로 사용할 수 있는지 여부를 확인하고, 사용할 수 있다면 NativeDragManager.acceptDragDrop() 메서드를 호출하여 데이터를 대상 위에 드롭합니다.

드래그 동작이 대화형 객체 위에 머물러 있을 경우 해당 객체는 nativeDragOver 이벤트를 전달합니다. 대화형 객체에 대한 드래그 동작이 중단되면 nativeDragExit 이벤트가 전달됩니다.

드롭 사용자는 유자격 드롭 대상 위에서 마우스를 놓습니다. 대상이 AIR 응용 프로그램 또는 구성 요소일 경우 대상 객체는 nativeDragDrop 이벤트를 전달합니다. 이벤트 핸들러는 이벤트 객체로부터 전송된 데이터에 액세스합니다. 대상이 AIR 외부에 있을 경우 운영 체제 또는 다른 응용 프로그램이 드롭을 처리합니다. 두 경우 모두 시작 객체는 nativeDragComplete 이벤트를 전달합니다(드래그가 AIR 내부에서 시작된 경우).

NativeDragManager 클래스는 드래그인 및 드래그아웃 동작 모두 제어합니다. NativeDragManager 클래스의 모든 멤버는 정적이며 이 클래스의 인스턴스를 만들지 않습니다.

Clipboard 객체

응용 프로그램 또는 구성 요소의 내부로 또는 외부로 드래그된 데이터는 클립보드 객체에 보관됩니다. 하나의 클립보드 객체는 동일한 정보를 서로 다르게 표시할 수 있도록 함으로써 다른 응용 프로그램이 해당 데이터를 이해 및 사용할 수 있는 가능성을 높입니다. 예를 들어 이미지는 이미지 데이터, 일련화된 비트맵 객체 및 파일로 포함될 수 있습니다. 한 가지 포맷으로 데이터를 렌더링하는 것은 데이터가 읽힐 때까지 호출되지 않은 렌더링 기능으로 지연시킬 수 있습니다.

드래그 동작이 시작되면 클립보드 객체는 nativeDragEnter, nativeDragOver 및 nativeDragDrop 이벤트의 이벤트 핸들러 내에서만 액세스할 수 있습니다. 드래그 동작이 종료되면 클립보드 객체를 읽거나 재사용할 수 없습니다.

응용 프로그램 객체는 참조 및 직렬화된 객체로 전송될 수 있습니다. 참조는 원본 응용 프로그램 내에서만 유효합니다. 직렬화된 객체 전송은 AIR 응용 프로그램 간에 유효하며, 직렬화 및 비직렬화될 때 유효한 객체에만 사용할 수 있습니다. 직렬화된 객체는 문자열 기반 데이터 전송 포맷인 AMF3(Action Message Format for ActionScript 3)로 변환됩니다.

Flex 프레임워크에 대한 작업

대개의 경우 Flex 응용 프로그램 구축 시 Adobe® Flex™ 끌어 놓기 API를 사용하는 것이 더 좋습니다. Flex 프레임워크는 Flex 응용 프로그램이 AIR에서 실행될 때 해당 기능 집합을 제공합니다(내부적으로 AIR NativeDragManager 사용). 또한 Flex는 응용 프로그램 또는 구성 요소가 보다 제한적인 브라우저 환경에서 실행될 때 더 제한적인 기능 집합을 제공합니다. AIR 클래스는 AIR 런타임 환경 외부에서 실행되는 구성 요소나 응용 프로그램에서는 사용할 수 없습니다.

드래그아웃 동작 지원

Adobe AIR 1.0 이상

드래그아웃 동작을 지원하려면 mouseDown 이벤트에 대한 응답으로 클립보드 객체를 만들어 이를 NativeDragManager.doDrag() 메서드로 보내야 합니다. 응용 프로그램은 시작 객체에서 nativeDragComplete 이벤트를 수신하여 사용자가 해당 동작을 완료하거나 중단할 때 어떤 조치를 취할 것인지를 결정합니다.

전송할 데이터 준비

Flash Player 9 이상, Adobe AIR 1.0 이상

드래그할 데이터 또는 객체를 준비하려면 Clipboard 객체를 만들고 하나 이상의 포맷으로 전송할 정보를 추가합니다. 기본 클립보드 포맷으로 자동 변환될 수 있는 데이터를 전달하기 위해 표준 데이터 형식을 사용하고, 객체를 전달하기 위해 응용 프로그램에서 정의한 형식을 사용할 수 있습니다.

특정 포맷으로 전송할 정보를 변환하는 계산 비용이 높은 경우 변환을 수행하는 핸들러 함수의 이름을 제공할 수 있습니다. 이 함수는 수신 구성 요소 또는 응용 프로그램이 연관된 포맷을 읽는 경우에만 호출됩니다.

클립보드 포맷에 대한 자세한 내용은 545페이지의 “클립보드 데이터 형식”을 참조하십시오.

다음 예제에서는 비트맵 객체, 기본 비트맵 포맷 및 비트맵이 처음 로드된 파일이 포함된 파일 목록 포맷 등, 여러 가지 포맷의 비트맵이 포함된 클립보드 객체를 만드는 방법을 보여 줍니다.

```
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;
}
```

드래그아웃 작업 시작

Adobe AIR 1.0 이상

드래그 작업을 시작하려면 마우스 클릭 이벤트에 대한 응답으로 NativeDragManager.doDrag() 메서드를 호출합니다. doDrag() 메서드는 다음과 같은 매개 변수를 사용하는 정적 메서드입니다.

매개 변수	설명
initiator	드래그가 시작되는 객체로 dragStart 및 dragComplete 이벤트를 전달합니다. 시작자는 대화형 객체여야 합니다.
clipboard	전송될 데이터가 포함된 Clipboard 객체입니다. Clipboard 객체는 끌어 놓기 시퀀스 중에 전달되는 NativeDragEvent 객체에서 참조됩니다.
dragImage	(선택적) 드래그 중에 표시할 BitmapData 객체입니다. 이미지는 alpha 값을 지정할 수 있습니다. (참고: Microsoft Windows는 항상 고정 alpha 페이드를 드래그 이미지에 적용합니다.)
offset	(선택적) 마우스 핫스팟으로부터 드래그 이미지의 오프셋을 지정하는 Point 객체입니다. 마우스 커서의 위와 왼쪽으로 드래그 이미지를 이동하려면 음수 좌표를 사용합니다. 오프셋이 제공되지 않은 경우 드래그 이미지의 왼쪽 위 모서리가 마우스 핫스팟 위치에 배치됩니다.
actionsAllowed	(선택적) 드래그 동작에 유효한 작업(복사, 이동 또는 연결)을 지정하는 NativeDragOptions 객체입니다. 인수를 지정하지 않으면 모든 작업이 허용됩니다. DragOptions 객체는 허용된 작업이 대상 구성 요소의 목적에 부합되는지를 잠재적인 드래그 대상이 검사할 수 있도록 NativeDragEvent 객체에서 참조됩니다. 예를 들어 "trash" 구성 요소는 이동 작업을 허용하는 드래그 동작만을 허용할 수 있습니다.

다음 예제에서는 파일에서 로드된 비트맵 객체에 대해 드래그 작업을 시작하는 방법을 보여 줍니다. 이 예제에서는 이미지를 로드하고 mouseDown 이벤트가 발생하면 드래그 작업을 시작합니다.

```
 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);
    }
}
}
```

드래그아웃 전송 완료

Adobe AIR 1.0 이상

사용자가 마우스를 놓음으로써 드래그된 항목을 드롭하면 시작자 객체는 nativeDragComplete 이벤트를 전달합니다. 이벤트 객체의 dropAction 속성을 검사한 다음 적절한 작업을 수행할 수 있습니다. 예를 들어 NativeDragAction.MOVE 작업의 경우 원래 위치에서 소스 항목을 제거할 수 있습니다. 커서가 유효한 드롭 대상 외부에 있을 때 마우스 버튼을 놓으면 드래그 동작이 중단됩니다. 드래그 관리자는 중단된 동작의 dropAction 속성을 NativeDragAction.NONE으로 설정합니다.

드래그인 동작 지원

Adobe AIR 1.0 이상

드래그인 동작을 지원하려면 응용 프로그램(또는 보다 일반적으로는 응용 프로그램의 시각적 구성 요소)이 nativeDragEnter 또는 nativeDragOver 이벤트에 응답해야 합니다.

일반적인 드롭 작업의 단계

Adobe AIR 1.0 이상

일반적으로 드롭 작업의 이벤트 시퀀스는 다음과 같습니다.

1. 사용자가 클립보드 객체를 구성 요소 위로 드래그합니다.
2. 구성 요소가 nativeDragEnter 이벤트를 전달합니다.
3. nativeDragEnter 이벤트 핸들러는 이벤트 객체를 검사하여 사용 가능한 데이터 형식 및 허용되는 작업을 확인합니다. 구성 요소가 드롭을 처리할 수 있으면 NativeDragManager.acceptDragDrop()을 호출합니다.
4. NativeDragManager는 해당 객체를 드롭할 수 있음을 나타내도록 마우스 커서를 변경합니다.
5. 사용자가 구성 요소 위에 객체를 드롭합니다.
6. 수신 구성 요소는 nativeDragDrop 이벤트를 전달합니다.

7 수신 구성 요소는 이벤트 객체 안의 Clipboard 객체로부터 원하는 포맷의 데이터를 읽습니다.

8 드래그 동작이 AIR 응용 프로그램 내부에서 시작된 경우 시작한 대화형 객체가 nativeDragComplete 이벤트를 전달합니다. 동작이 AIR 외부에서 시작된 경우 피드백을 보내지 않습니다.

드래그인 동작 확인

Adobe AIR 1.0 이상

사용자가 클립보드 항목을 시각적 구성 요소의 경계에 드래그하면 구성 요소는 nativeDragEnter 및 nativeDragOver 이벤트를 전달합니다. 구성 요소가 클립보드 항목을 허용할 수 있는지 여부를 확인하기 위해 이들 이벤트의 핸들러는 이벤트 객체의 clipboard 및 allowedActions 속성을 검사할 수 있습니다. 구성 요소가 드롭을 허용할 수 있음을 신호하기 위해 이벤트 핸들러는 NativeDragManager.acceptDragDrop() 메서드를 호출하여 수신 구성 요소에 참조를 전달해야 합니다. 두 개 이상의 등록된 이벤트 수신기가 acceptDragDrop() 메서드를 호출할 경우 목록의 마지막 핸들러가 우선권을 가집니다. acceptDragDrop() 호출은 수신 객체의 경계에서 마우스가 떠나고 nativeDragExit 이벤트를 트리거할 때까지 유효합니다.

doDrag()에 전달되는 allowedActions 매개 변수에서 둘 이상의 작업이 허용될 경우 사용자는 수정자 키를 눌러서 허용되는 작업 중 어떤 작업을 수행하려 하는지를 나타낼 수 있습니다. 사용자가 드래그 동작을 완료하면 드래그 관리자는 커서 이미지를 변경하여 어떤 작업이 발생할 것인지를 사용자에게 알려 줍니다. 의도한 작업은 NativeDragEvent object의 dropAction 속성에 의해 보고됩니다. 드래그 동작의 작업 집합은 참고용입니다. 전송에 관련된 구성 요소는 적절한 비헤이비어를 구현해야 합니다. 예를 들어 이동 작업을 완료하려면 드래그 시작자는 드래그된 항목을 제거하고 드롭 대상이 이를 추가합니다.

드래그 대상은 NativeDragManager 클래스의 dropAction 속성을 설정하여 세 가지 가능한 작업 중 한 가지로 드롭 작업을 제한할 수 있습니다. 사용자가 키보드를 사용하여 다른 작업을 선택하려 할 경우 NativeDragManager는 사용 불가 커서를 표시합니다. nativeDragEnter 및 nativeDragOver 이벤트의 핸들러에 dropAction 속성을 설정합니다.

다음 예제에서는 nativeDragEnter 또는 nativeDragOver 이벤트의 이벤트 핸들러를 보여 줍니다. 이 핸들러는 드래그되는 클립보드에 텍스트 포맷의 데이터가 있는 경우 드래그인 동작만 허용합니다.

```
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
    }
}
```

드롭 완료

Adobe AIR 1.0 이상

사용자가 해당 동작을 허용한 대화형 객체 위에 드래그된 항목을 드롭하면 대화형 객체는 nativeDragDrop 이벤트를 전달합니다. 이 이벤트의 핸들러는 이벤트 객체의 clipboard 속성으로부터 데이터를 추출할 수 있습니다.

클립보드에 응용 프로그램에서 정의한 포맷이 포함되어 있을 때, Clipboard 객체의 getData() 메서드로 전달된 transferMode 매개 변수는 드래그 관리자가 참조를 반환하는지 또는 객체의 일련화된 버전을 반환하는지를 판별합니다.

다음 예제에서는 nativeDragDrop 이벤트의 이벤트 핸들러를 보여 줍니다.

```
 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));
}
```

이벤트 핸들러가 종료되면 더 이상 Clipboard 객체가 유효하지 않습니다. 객체 또는 해당 데이터에 액세스하려 할 경우 오류가 발생합니다.

구성 요소의 시각적 형태 업데이트

Adobe AIR 1.0 이상

구성 요소는 NativeDragEvent 이벤트에 근거하여 시각적 형태를 업데이트할 수 있습니다. 다음 표에서는 다양한 이벤트에 대한 응답으로 일반적인 구성 요소가 변경할 수 있는 변경 유형에 대해 설명합니다.

이벤트	설명
nativeDragStart	시작 대화형 객체는 nativeDragStart 이벤트를 사용하여 드래그 동작이 해당 대화형 객체로부터 시작되었다는 시각적 피드백을 제공합니다.
nativeDragUpdate	시작 대화형 객체는 nativeDragUpdate 이벤트를 사용하여 동작 중에 해당 상태를 업데이트합니다. Linux 환경에서는 AIR에서 이 이벤트를 사용할 수 없습니다.
nativeDragEnter	잠재적인 수신 대화형 객체는 이 이벤트를 사용하여 초점을 맞추거나 드롭 허용 또는 허용 불가능을 시각적으로 나타냅니다.
nativeDragOver	잠재적인 수신 대화형 객체는 이 이벤트를 사용하여 지도 표시와 같이 복잡한 구성 요소의 "핫" 영역으로 마우스가 들어오는 경우와 같이 대화형 객체 내에서 마우스 움직임에 반응합니다.
nativeDragExit	잠재적인 수신 대화형 객체는 이 이벤트를 사용하여 드래그 동작이 해당 경계 외부에서 움직일 때 해당 상태를 복원합니다.
nativeDragComplete	시작 대화형 객체는 이 이벤트를 사용하여 목록에서 항목을 제거하는 것과 같이 연관된 데이터 모델을 업데이트하고 해당 시각적 상태를 복원합니다.

드래그인 동작 중 마우스 위치 추적

Adobe AIR 1.0 이상

드래그 동작이 구성 요소 위에 머물러 있는 동안 구성 요소는 nativeDragOver 이벤트를 전달합니다. 이 이벤트는 수 밀리초마다 그리고 마우스가 움직일 때마다 전달됩니다. nativeDragOver 이벤트 객체는 구성 요소 위에서 마우스 위치를 판별하는 데 사용할 수 있습니다. 마우스 위치를 파악하는 것은 구성 요소가 복잡하지만 하위 구성 요소로 구성되어 있지는 않은 경우 유용할 수 있습니다. 예를 들어 응용 프로그램에서 지도가 들어 있는 비트맵을 표시하고 사용자가 정보를 드래그하면 지도상의 구역이 강조 표시되도록 할 경우, nativeDragOver 이벤트에 보고되는 마우스 좌표를 사용하여 지도 안에서의 마우스 위치를 추적합니다.

HTML의 드래그 앤 드롭

Adobe AIR 1.0 이상

HTML 기반 응용 프로그램 내부로 또는 외부로(또는 HTMLLoader에 표시된 HTML 내부로 또는 외부로) 데이터를 드래그할 경우 HTML 끌어 놓기 이벤트를 사용할 수 있습니다. HTML 끌어 놓기 API를 사용하여 HTML 내용의 DOM 요소 내부로 또는 외부로 드래그할 수 있습니다.

참고: 또한 HTML 내용이 포함된 HTMLLoader 객체에서 이벤트를 청취함으로써 AIR NativeDragEvent 및 NativeDragManager API를 사용할 수도 있습니다. 그러나 HTML API가 HTML DOM과 더 잘 통합되며 기본 비헤이비어를 제어할 수 있습니다.

기본 끌어 놓기 비헤이비어

Adobe AIR 1.0 이상

HTML 환경은 텍스트, 이미지 및 URL과 관련된 드래그 앤 드롭 동작의 기본 비헤이비어를 제공합니다. 기본 비헤이비어를 사용하여 요소로부터 항상 이러한 유형의 데이터를 드래그할 수 있습니다. 그러나 텍스트만 요소로 드래그할 수 있으며, 페이지의 편집 가능 영역의 요소로만 드래그할 수 있습니다. 페이지의 편집 가능 영역 간에 또는 편집 가능 영역 내에서 텍스트를 드래그할 때 기본 비헤이비어는 이동 작업을 수행합니다. 편집 불가능 영역 또는 응용 프로그램 외부에서 편집 가능 영역으로 텍스트를 드래그할 때 기본 비헤이비어는 복사 작업을 수행합니다.

직접 끌어 놓기 이벤트를 처리하여 기본 비헤이비어를 재정의할 수 있습니다. 기본 비헤이비어를 취소하려면 끌어 놓기 이벤트를 위해 전달된 객체의 preventDefault() 메서드를 호출해야 합니다. 그런 다음 드롭 대상에 데이터를 삽입하고 필요에 따라 드래그 소스에서 데이터를 제거하여 선택한 작업을 수행합니다.

기본적으로 사용자는 어떤 텍스트든 선택 및 드래그할 수 있으며 이미지 및 링크도 드래그할 수 있습니다. WebKit CSS 속성인 -webkit-user-select를 사용하여 HTML 요소를 선택하는 방법을 제어할 수 있습니다. 예를 들어 -webkit-user-select를 none으로 설정할 경우 요소 내용을 선택할 수 없으며 따라서 드래그할 수도 없습니다. 또한 -webkit-user-drag CSS 속성을 사용하여 요소 전체를 드래그할 수 있는지 여부를 제어할 수 있습니다. 그러나 요소의 내용은 별도로 처리됩니다. 텍스트의 일부만 선택하여 드래그할 수도 있습니다. 자세한 내용은 887페이지의 “AIR에서의 CSS”를 참조하십시오.

HTML에서의 끌어 놓기 이벤트

Adobe AIR 1.0 이상

드래그가 시작되는 시작자 요소에 의해 전달되는 이벤트는 다음과 같습니다.

이벤트	설명
dragstart	사용자가 드래그 동작을 시작할 때 전달됩니다. 이 이벤트의 핸들러는 필요에 따라 이벤트 객체의 preventDefault() 메서드를 호출함으로써 드래그를 방지할 수 있습니다. 드래그된 데이터를 복사, 링크 또는 이동할 수 있는지 여부를 제어하려면 effectAllowed 속성을 설정합니다. 선택된 텍스트, 이미지 및 링크는 기본 동작에 의해 클립보드에 놓이지만 이벤트 객체의 dataTransfer 속성을 사용하여 드래그 동작에 다른 데이터를 설정할 수 있습니다.
drag	드래그 동작 중에 연속적으로 전달됩니다.
dragend	사용자가 드래그 동작을 끝내기 위해 마우스 버튼을 놓으면 전달됩니다.

드래그 대상에 의해 전달되는 이벤트는 다음과 같습니다.

이벤트	설명
dragover	요소 경계 내에서 드래그 동작이 남아 있는 동안 지속적으로 전달됩니다. 이 이벤트의 핸들러는 사용자가 마우스 버튼을 놓을 경우 드롭 결과 복사, 이동 또는 링크 작업이 발생하는지 여부를 나타내기 위해 dataTransfer.dropEffect 속성을 설정해야 합니다.
dragenter	드래그 동작이 요소의 경계에 진입할 때 전달됩니다. dragenter 이벤트 핸들러의 dataTransfer 객체의 속성을 변경할 경우 해당 변경 사항이 다음 dragover 이벤트에 의해 신속하게 재정의됩니다. 한편 dragenter와 첫 번째 dragover 이벤트 사이에는 짧은 지연이 있어서 서로 다른 속성이 설정되어 있는 경우 커서가 깜박거릴 수 있습니다. 많은 경우 두 가지 이벤트에 동일한 이벤트 핸들러를 사용할 수 있습니다.
dragleave	드래그 동작이 요소 경계를 떠날 때 전달됩니다.
drop	사용자가 데이터를 요소 위에 드롭할 때 전달됩니다. 드래그되는 데이터는 이 이벤트의 핸들러 내에서만 액세스할 수 있습니다.

이들 이벤트에 대한 응답으로 전달되는 이벤트 객체는 마우스 이벤트와 유사합니다. (clientX, clientY) 및 (screenX, screenY)와 같은 마우스 이벤트 속성을 사용하여 마우스 위치를 판별할 수 있습니다.

드래그 이벤트 객체의 가장 중요한 속성은 드래그되는 데이터가 포함되는 dataTransfer입니다. dataTransfer 객체에는 다음과 같은 속성 및 메서드가 들어 있습니다.

속성 또는 메서드	설명
effectAllowed	드래그 소스에서 허용되는 효과입니다. 일반적으로 dragstart 이벤트의 핸들러가 이 값을 설정합니다. 562페이지의 "HTML에서의 드래그 효과"를 참조하십시오.
dropEffect	대상 또는 사용자가 선택한 효과입니다. dragover 또는 dragenter 이벤트 핸들러에서 dropEffect를 설정할 경우 AIR은 사용자가 마우스를 놓았을 때 발생하는 효과를 나타내도록 마우스 커서를 업데이트합니다. dropEffect 집합이 허용되는 효과 중 일치하지 않는 효과가 있을 경우 드롭이 허용되지 않으며 **사용 불가** 커서가 표시됩니다. 최근의 dragover 또는 dragenter 이벤트에 대한 응답으로 dropEffect를 설정하지 않은 경우 사용자는 표준 운영 체제 수정자 키를 사용하여 허용되는 효과에서 선택할 수 있습니다. 최종적인 효과는 dragend에 대해 전달된 객체의 dropEffect 속성에 의해 보고됩니다. 유효한 대상 외부에서 마우스 버튼을 놓음으로써 드롭 동작을 중단할 경우 dropEffect는 none으로 설정됩니다.
types	각 데이터 형식에 대한 MIME 유형 문자열이 들어 있는 배열이 dataTransfer 객체에 들어 있습니다.
getData(mimeType)	mimeType 매개 변수로 지정한 포맷의 데이터를 가져옵니다. getData() 메서드는 drop 이벤트에 대한 응답으로만 호출할 수 있습니다.
setData(mimeType)	데이터를 mimeType 매개 변수로 지정한 포맷으로 dataTransfer에 추가합니다. 각 MIME 유형에 대해 setData()를 호출함으로써 여러 포맷의 데이터를 추가할 수 있습니다. 기본 드래그 비헤이비어에 의해 dataTransfer 객체에 위치한 모든 데이터는 제거됩니다. setData() 메서드는 dragstart 이벤트에 대한 응답으로만 호출할 수 있습니다.
clearData(mimeType)	mimeType 매개 변수로 지정한 포맷의 모든 데이터를 제거합니다.
setDragImage(image, offsetX, offsetY)	사용자 정의 드래그 이미지를 설정합니다. setDragImage() 메서드는 dragstart 이벤트에 대한 응답으로만, 그리고 HTML 요소의 -webkit-user-drag CSS 스타일을 element로 설정하여 전체 HTML 요소가 드래그되는 경우에만 호출할 수 있습니다. image 매개 변수는 JavaScript 요소 또는 Image 객체일 수 있습니다.

HTML 끌어 놓기를 위한 MIME 유형

Adobe AIR 1.0 이상

HTML 끌어 놓기 이벤트의 dataTransfer 객체와 함께 사용할 수 있는 MIME 유형은 다음과 같습니다.

데이터 형식	MIME 유형
텍스트	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
비트맵	"image/x-vnd.adobe.air.bitmap"
파일 목록	"application/x-vnd.adobe.air.file-list"

응용 프로그램에서 정의한 문자열을 포함하여 기타 MIME 문자열도 사용할 수 있습니다. 그러나 다른 응용 프로그램에서 전송된 데이터를 인식하지 못하거나 사용하지 못할 수 있습니다. 데이터를 해당 포맷으로 dataTransfer 객체에 추가하는 것은 사용자가 해야 합니다.

중요: 응용 프로그램 샌드박스에서 실행되는 코드만 드롭된 파일에 액세스할 수 있습니다. 비 응용 프로그램 샌드박스 내의 File 객체 속성을 읽거나 설정하려 할 경우 보안 오류가 발생합니다. 자세한 내용은 566페이지의 "비 응용 프로그램 HTML 샌드박스에서 파일 드롭 처리"를 참조하십시오.

HTML에서의 드래그 효과

Adobe AIR 1.0 이상

드래그 동작의 시작자는 dragstart 이벤트의 핸들러에서 dataTransfer.effectAllowed 속성을 설정하여 허용되는 드래그 효과를 제한할 수 있습니다. 다음과 같은 문자열 값이 사용될 수 있습니다.

문자열 값	설명
"none"	드래그 조작이 허용되지 않습니다.
"copy"	원본을 그 자리에 둔 상태에서 데이터가 목적지로 복사됩니다.
"link"	데이터가 원본과의 링크를 사용하여 드롭 목적지와 공유됩니다.
"move"	데이터가 목적지로 복사되고 원본 위치에서 제거됩니다.
"copyLink"	데이터가 복사 또는 연결될 수 있습니다.
"copyMove"	데이터가 복사 또는 이동될 수 있습니다.
"linkMove"	데이터가 연결 또는 이동될 수 있습니다.
"all"	데이터가 복사, 이동 또는 연결될 수 있습니다. **All**은 기본 비헤이비어를 차단할 때의 기본 효과입니다.

드래그 동작의 대상은 dataTransfer.dropEffect 속성을 설정하여 사용자가 드롭을 완료할 경우 수행할 작업을 나타냅니다. 드롭 효과가 허용되는 작업 중 하나인 경우 시스템은 적절한 복사, 이동 또는 연결 커서를 표시합니다. 그렇지 않은 경우 시스템은 사용 불가 커서를 표시합니다. 대상이 드롭 효과를 설정하지 않은 경우 사용자는 수정자 키를 사용하여 허용되는 작업 중에서 선택할 수 있습니다.

dragover 및 dragenter 이벤트에 대한 핸들러에서 dropEffect 값을 설정합니다.

```
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";
}
```

참고: dragenter에 대한 핸들러에서 항상 dropEffect 속성을 설정해야 하지만 다음 dragover 이벤트가 기본값으로 속성을 재설정함을 기억하십시오. 두 가지 이벤트에 대한 응답으로 dropEffect를 설정합니다.

HTML 요소로부터 데이터 드래그

Adobe AIR 1.0 이상

기본 비헤이비어를 통해 HTML 페이지의 대부분의 내용을 드래그하여 복사할 수 있습니다. CSS 속성인 -webkit-user-select 및 -webkit-user-drag를 사용하여 드래그할 수 있는 내용을 제어할 수 있습니다.

dragstart 이벤트에 대해 핸들러에서 기본 드래그아웃 비헤이비어를 재정의합니다. 이벤트 객체의 dataTransfer 속성의 setData() 메서드를 호출하여 사용자의 데이터를 드래그 동작에 놓습니다.

기본 비헤이비어를 사용하지 않을 경우 소스 객체가 어떤 드래그 효과를 지원하는지 나타내려면 dragstart 이벤트에 대해 전달된 이벤트 객체의 dataTransfer.effectAllowed 속성을 설정합니다. 효과를 자유롭게 선택하여 조합할 수 있습니다. 예를 들어 소스 요소가 복사 및 연결 효과를 지원할 경우 속성을 "copyLink"로 설정합니다.

드래그된 데이터 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

dataTransfer 속성을 사용하여 dragstart 이벤트의 핸들러에 드래그 동작에 대한 데이터를 추가합니다. dataTransfer.setData() 메서드를 사용하여 데이터를 클립보드에 놓고 전송할 MIME 유형 및 데이터를 전달합니다.

예를 들어 응용 프로그램 id가 **imageOfGeorge**인 이미지 요소가 있는 경우 다음과 같은 dragstart 이벤트 핸들러를 사용할 수 있습니다. 이 예에서는 여러 가지 데이터 형식의 George의 사진을 추가하며, 이는 다른 응용 프로그램에서 드래그된 데이터를 사용할 수 있는 가능성을 증가시킵니다.

```
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));
}
```

참고: dataTransfer 객체의 setData() 메서드를 호출할 때 기본 끌어 놓기 비헤이비어는 어떠한 데이터도 추가하지 않습니다.

HTML 요소로 데이터 드래그

Adobe AIR 1.0 이상

기본 비헤이비어에서는 페이지의 편집 가능한 영역으로 텍스트만 드래그할 수 있습니다. 요소의 여는 태그에 contenteditable 특성을 포함시키면 요소 및 그 자식을 편집할 수 있도록 지정할 수 있습니다. 문서 객체 designMode 속성을 "on"으로 설정하여 문서 전체를 편집할 수 있게 만들 수도 있습니다.

드래그된 데이터를 허용할 수 있는 요소에 대해 dragenter, dragover 및 drop 이벤트를 처리함으로써 페이지에서 다른 드래그인 비헤이비어를 지원할 수 있습니다.

드래그인 지원

Adobe AIR 1.0 이상

드래그인 동작을 처리하기 위해서는 먼저 기본 비헤이비어를 취소해야 합니다. 드롭 대상으로 사용하려는 HTML 요소에서 dragenter 및 dragover 이벤트를 수신합니다. 이들 이벤트에 대한 핸들러에서 전달된 이벤트 객체의 preventDefault() 메서드를 호출합니다. 기본 비헤이비어를 취소하면 편집 불가능한 영역에서 드롭을 수신할 수 있습니다.

드롭된 데이터 받기

Adobe AIR 1.0 이상

ondrop 이벤트에 대한 핸들러에서 드롭된 데이터에 액세스할 수 있습니다.

```
function doDrop(event){
    droppedText = event.dataTransfer.getData("text/plain");
}
```

dataTransfer.getData() 메서드를 사용하여 데이터를 클립보드에 읽어 들이고 읽을 데이터 형식의 MIME 유형을 전달합니다. dataTransfer 객체의 types 속성을 사용하여 어떤 데이터 형식을 사용할 수 있는지 알아낼 수 있습니다. types 배열에는 각각의 사용 가능한 포맷의 MIME 유형 문자열이 들어 있습니다.

dragenter 또는 dragover 이벤트에서 기본 비헤이비어를 취소하면, 문서의 적절한 위치에 드롭된 데이터를 삽입하는 것은 사용자가 해야 합니다. 요소 내에서 마우스 위치를 삽입점으로 변환하는 API는 없습니다. 이러한 한계로 인해 삽입 유형의 드래그 동작을 구현하는 것이 어려울 수 있습니다.

예제: 기본 HTML 드래그인 비헤이비어 재정의

Adobe AIR 1.0 이상

이 예제에서는 드롭된 항목에서 사용할 수 있는 각각의 데이터 형식을 보여 주는 표를 표시하는 드롭 대상을 구현합니다.

기본 비헤이비어를 사용하여 응용 프로그램 내에서 텍스트, 링크 및 이미지를 드래그할 수 있습니다. 이 예제에서는 드롭 대상으로 기능하는 div 요소의 기본 드래그인 비헤이비어를 재정의합니다. 편집 불가능한 내용에서 드래그인 동작을 허용하도록 하기 위한 주요 단계는 dragenter 및 dragover 이벤트에 대해 전달된 이벤트 객체의 preventDefault() 메서드를 호출하는 것입니다. drop 이벤트에 대한 응답으로 핸들러는 전송된 데이터를 HTML 줄 요소로 변환하고 이 줄을 표시할 표에 삽입합니다.

```
 <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><td> </td></tr>
       </table>
       </div>
</div>
</body>
</html>
```

비 응용 프로그램 HTML 샌드박스에서 파일 드롭 처리

Adobe AIR 1.0 이상

비 응용 프로그램 내용은 파일이 AIR 응용 프로그램으로 드래그될 때 생성되는 File 객체에 액세스할 수 없습니다. 이들 File 객체 중 하나를 샌드박스 브리지를 통해 응용 프로그램 내용으로 전달하는 것도 불가능합니다. (객체 속성은 일련화 과정에서 액세스해야 합니다.) 그러나 HTMLLoader 객체에서 AIR nativeDragDrop 이벤트를 수신함으로써 응용 프로그램 내에서 파일을 드롭할 수 있습니다.

일반적으로 사용자가 비 응용 프로그램 내용을 호스팅하는 프레임으로 파일을 드롭할 경우 드롭 이벤트는 자식에서 부모로 전달되지 않습니다. 그러나 HTMLLoader(AIR 응용 프로그램에서 모든 HTML 내용을 포함하는 컨테이너)가 전달하는 이벤트가 HTML 이벤트 흐름에 속하지 않는 경우 응용 프로그램 내용에서 드롭 이벤트를 수신할 수 있습니다.

파일 드롭에 대한 이벤트를 수신하려면 window.htmlLoader가 제공하는 참조를 사용하여 부모 문서는 HTMLLoader 객체에 이벤트 수신기를 추가합니다.

```
window.htmlLoader.addEventListener("nativeDragDrop",function(event){
    var filelist = event.clipboard.getData(air.ClipboardFormats.FILE_LIST_FORMAT);
    air.trace(filelist[0].url);
});
```

다음 예제에서는 원격 샌드박스로 자식 페이지를 로드하는 부모 문서를 사용합니다(http://localhost/). 부모는 HTMLLoader 객체에서 nativeDragDrop 이벤트를 수신하고 파일 url을 추적합니다.

```
 <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>
```

자식 문서는 HTML dragenter 및 dragover 이벤트 핸들러에서 Event 객체 preventDefault() 메서드를 호출함으로써 유효한 드롭 대상을 제시해야 합니다. 그렇지 않으면 드롭 이벤트가 절대 발생할 수 없습니다.

```
 <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>
```

파일 프로미스 드롭

Adobe AIR 2 이상

파일 프로미스는 사용자가 아직 AIR 응용 프로그램 외부에 없는 파일을 드래그할 수 있도록 해 주는 드래그 앤 드롭 클립보드 형식입니다. 예를 들어 파일 프로미스를 사용하면 응용 프로그램에서 사용자가 프록시 아이콘을 데스크톱 폴더로 드래그할 수 있습니다. 프록시 아이콘은 URL에서 사용할 수 있는 것으로 알려진 파일 또는 데이터를 나타냅니다. 사용자가 아이콘을 드롭한 후 런타임으로 데이터가 다운로드되고 파일이 드롭 위치에 기록됩니다.

AIR 응용 프로그램에서 URLFilePromise 클래스를 사용하여 URL에서 액세스할 수 있는 파일을 드래그 앤 드롭할 수 있습니다. URLFilePromise 구현은 AIR 2 SDK의 일부로 aircore 라이브러리에 제공됩니다. SDK frameworks/libs/air 디렉토리에 있는 aircore.swc 또는 aircore.swf 파일을 사용하면 됩니다.

또는 IFilePromise 인터페이스(런타임 flash.desktop 패키지에 정의됨)로 자체 파일 프로미스 논리를 구현할 수도 있습니다.

파일 프로미스는 개념상 클립보드에서 데이터 핸들러 함수를 사용하여 지연된 렌더링을 수행하는 것과 비슷합니다. 파일을 드래그 앤 드롭하는 경우 지연된 렌더링 대신 파일 프로미스를 사용하십시오. 지연된 렌더링 기법은 데이터가 생성 또는 다운로드될 때 드래그 동작으로 인해 원치 않는 일시 중지가 발생할 수 있습니다. 파일 프로미스가 지원되지 않는 복사하여 붙여넣기 작업에는 지연된 렌더링을 사용하십시오.

파일 프로미스 사용 시 제한 사항

파일 프로미스에는 드래그 앤 드롭 클립보드에서 사용할 수 있는 다른 데이터 형식에 비해 다음과 같은 제한 사항이 있습니다.

- 파일 프로미스는 AIR 응용 프로그램에서 드래그할 수만 있고, AIR 응용 프로그램으로 드롭할 수는 없습니다.
- 일부 운영 체제에서는 파일 프로미스가 지원되지 않습니다. 파일 프로미스가 호스트 시스템에서 지원되는지 여부를 테스트하려면 Clipboard.supportsFilePromise 속성을 사용하십시오. 파일 프로미스를 지원하지 않는 시스템에서는 파일 데이터를 다운로드하거나 생성할 수 있는 대체 메커니즘을 제공해야 합니다.
- 파일 프로미스는 복사하여 붙여넣기 클립보드(Clipboard.generalClipboard)와 함께 사용할 수 없습니다.

기타 도움말 항목

flash.desktop.IFilePromise

air.desktop.URLFilePromise

원격 파일 드롭

Adobe AIR 2 이상

URL에서 사용할 수 있는 파일 또는 데이터를 나타내는 파일 프로미스 객체를 만들려면 URLFilePromise 클래스를 사용합니다. FILE_PROMISE_LIST 클립보드 형식을 사용하여 클립보드에 하나 이상의 파일 프로미스 객체를 추가합니다. 다음 예제에서는 http://www.example.com/foo.txt에서 사용할 수 있는 단일 파일을 다운로드하여 bar.txt라는 드롭 위치에 저장합니다. 원격 및 로컬 파일 이름은 일치할 필요가 없습니다.

```
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 );
}
```

클립보드에 할당된 배열에 파일 프로미스 객체를 더 추가하여 사용자가 한 번에 두 개 이상의 파일을 드래그하도록 허용할 수 있습니다. 또한 작업에 포함된 파일의 일부 또는 전부가 드롭 위치에 상대적인 하위 폴더에 저장되도록 relativePath 속성에서 하위 디렉토리를 지정할 수도 있습니다.

다음 예제에서는 여러 파일 프로미스를 포함하는 드래그 작업을 시작하는 방법을 보여 줍니다. 이 예제에서 html 페이지 **article.html**은 두 개의 연결된 이미지 파일과 함께 클립보드에 하나의 파일 프로미스로 저장됩니다. 이미지는 상대 링크가 유지되도록 **images** 하위 폴더에 복사됩니다.

```
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 );
}
```

IFilePromise 인터페이스 구현

Adobe AIR 2 이상

URLFilePromise 객체를 사용하여 액세스할 수 없는 리소스에 대해 파일 프로미스를 제공하려면 사용자 정의 클래스에서 IFilePromise 인터페이스를 구현할 수 있습니다. IFilePromise 인터페이스는 파일 프로미스가 드롭된 후 파일에 기록할 데이터에 액세스하기 위해 AIR 런타임에서 사용되는 메서드 및 속성을 정의합니다.

IFilePromise 구현은 AIR 런타임에 파일 프로미스에 대한 데이터를 제공하는 또 다른 객체를 전달합니다. 이 객체는 AIR 런타임이 데이터를 읽기 위해 사용하는 IDataInput 인터페이스를 구현해야 합니다. 예를 들어 IFilePromise를 구현하는 URLFilePromise 클래스는 URLStream 객체를 데이터 공급자로 사용합니다.

AIR는 데이터를 동기적 또는 비동기적으로 읽을 수 있습니다. IFilePromise 구현은 isAsync 속성에 적합한 값을 반환하여 지원되는 액세스 모드를 보고합니다. 비동기 데이터 액세스가 제공되는 경우 데이터 공급자 객체는 IEventDispatcher 인터페이스를 구현하고 open, progress 및 complete와 같은 필요한 이벤트를 전달해야 합니다.

사용자 정의 클래스 또는 다음과 같은 기본 제공 클래스 중 하나를 파일 프로미스에 대한 데이터 공급자로 사용할 수 있습니다.

- ByteArray(동기)
- FileStream(동기 또는 비동기)
- Socket(비동기)
- URLStream(비동기)

IFilePromise 인터페이스를 구현하려면 다음 함수 및 속성에 대한 코드를 제공해야 합니다.

- open():IDataInput - 프로미스된 파일에 대한 데이터를 읽을 데이터 공급자 객체를 반환합니다. 객체는 IDataInput 인터페이스를 구현해야 합니다. 또한 데이터가 비동기적으로 제공되는 경우 객체는 IEventDispatcher 인터페이스를 구현하고 필요한 이벤트를 전달해야 합니다(571페이지의 “파일 프로미스에서 비동기 데이터 공급자 사용” 참조).
- get relativePath():String - 만들어진 파일에 대한 파일 이름을 포함하는 경로를 제공합니다. 경로는 사용자가 드래그 앤 드롭 작업에서 선택한 드롭 위치에 대한 상대 경로로 확인됩니다. 경로에 호스트 운영 체제에 대한 올바른 분리 기호 문자가 사용되도록 하려면 디렉토리를 포함하는 경로를 지정할 때 File.separator 상수를 사용합니다. setter 함수를 추가하거나 constructor 매개 변수를 사용하여 경로를 런타임에 설정하도록 허용할 수 있습니다.
- get isAsync():Boolean - 데이터 공급자 객체가 데이터를 비동기적으로 제공하는지 아니면 동기적으로 제공하는지를 AIR 런타임에 알립니다.

- close():void - 데이터를 완전히 읽은 경우 또는 오류로 인해 더 이상 읽을 수 없는 경우 런타임에서 호출됩니다. 이 함수를 사용하여 리소스를 정리할 수 있습니다.
- reportError(e:ErrorEvent):void - 데이터를 읽는 동안 오류가 발생하는 경우 런타임에서 호출됩니다.

모든 IFilePromise 메서드는 파일 프로미스와 관련된 드래그 앤 드롭 작업 중에 런타임에서 호출됩니다. 일반적으로 응용 프로그램 논리에서는 이러한 메서드를 직접 호출하지 않아야 합니다.

파일 프로미스에서 동기 데이터 공급자 사용

Adobe AIR 2 이상

IFilePromise 인터페이스를 구현하는 가장 간단한 방법은 ByteArray 또는 동기 FileStream과 같은 동기 데이터 공급자 객체를 사용하는 것입니다. 다음 예제에서는 open() 메서드가 호출될 때 ByteArray 객체가 만들어지고 데이터로 채워지며 반환됩니다.

```
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 );
        }
    }
}
```

실제로, 동기 파일 프로미스는 사용이 제한적입니다. 데이터 양이 적은 경우에는 임시 디렉토리에 파일을 만들어서 드래그 앤 드롭 클립보드에 일반적인 파일 목록 배열을 추가하면 간편합니다. 반면에 데이터 양이 많거나 데이터 생성의 계산이 복잡한 경우 긴 동기 프로세스가 필요합니다. 긴 동기 프로세스는 상당한 시간 동안 UI 업데이트를 차단하여 응용 프로그램이 응답하지 않는 것처럼 보이게 만들 수 있습니다. 이러한 문제를 방지하기 위해서는 타이머로 구동되는 비동기 데이터 공급자를 만들 수 있습니다.

파일 프로미스에서 비동기 데이터 공급자 사용

Adobe AIR 2 이상

비동기 데이터 공급자 객체를 사용하는 경우에는 IFilePromise isAsync 속성이 true여야 하며 open() 메서드로 반환되는 객체에서 IEventDispatcher 인터페이스를 구현해야 합니다. 런타임에서는 여러 기본 제공 객체를 데이터 공급자로 사용할 수 있도록 몇 가지 대체 이벤트를 수신합니다. 예를 들어 progress 이벤트는 FileStream 및 URLStream 객체에 의해 전달되지만 socketData 이벤트는 Socket 객체에 의해 전달됩니다. 런타임에서는 이러한 모든 객체로부터 적합한 이벤트를 수신합니다.

다음 이벤트는 데이터 공급자 객체에서 데이터를 읽는 프로세스를 구동합니다.

- Event.OPEN - 데이터 소스가 준비되었음을 런타임에 알립니다.
- ProgressEvent.PROGRESS - 데이터를 사용할 수 있음을 런타임에 알립니다. 런타임에서 데이터 공급자 객체로부터 사용 가능한 데이터 양을 읽습니다.
- ProgressEvent.SOCKET_DATA - 데이터를 사용할 수 있음을 런타임에 알립니다. socketData 이벤트는 소켓 기반 객체에 의해 전달됩니다. 다른 객체 유형의 경우 progress 이벤트를 전달해야 합니다. 런타임에서 두 이벤트를 모두 수신하여 데이터를 읽을 수 있는 경우를 감지합니다.
- Event.COMPLETE - 데이터를 모두 읽었음을 런타임에 알립니다.
- Event.CLOSE - 데이터를 모두 읽었음을 런타임에 알립니다. 이를 위해 런타임에서 close 및 complete를 모두 수신합니다.
- IOErrorEvent.IOERROR - 데이터를 읽는 동안 오류가 발생했음을 런타임에 알립니다. 런타임에서 파일 만들기를 중단하고 IFilePromise close() 메서드를 호출합니다.
- SecurityErrorEvent.SECURITY_ERROR - 보안 오류가 발생했음을 런타임에 알립니다. 런타임에서 파일 만들기를 중단하고 IFilePromise close() 메서드를 호출합니다.
- HTTPStatusEvent.HTTP_STATUS - 사용 가능한 데이터가 오류 메시지(예: 404 페이지)가 아니라 원하는 내용을 나타내도록 하기 위해 httpResponseStatus와 함께 런타임에서 사용됩니다. HTTP 프로토콜 기반 객체는 이 이벤트를 전달해야 합니다.

- HTTPStatusEvent.HTTP_RESPONSE_STATUS - 사용 가능한 데이터가 원하는 내용을 나타내도록 하기 위해 httpStatus와 함께 런타임에서 사용됩니다. HTTP 프로토콜 기반 객체는 이 이벤트를 전달해야 합니다.

데이터 공급자는 다음과 같은 순서로 이러한 이벤트를 전달해야 합니다.

1. open 이벤트
2. progress 또는 socketData 이벤트
3. complete 또는 close 이벤트

참고: 기본 제공 객체 FileStream, Socket 및 URLStream은 적합한 이벤트를 자동으로 전달합니다.

다음 예제에서는 사용자 정의 비동기 데이터 공급자를 사용하여 파일 프로미스를 만듭니다. 데이터 공급자 클래스는 ByteArray(IDataInput 지원)를 확장하고 IEventDispatcher 인터페이스를 구현합니다. 각 타이머 이벤트에서 객체는 데이터 청크를 생성하고 데이터를 사용할 수 있음을 런타임에 알리기 위해 progress 이벤트를 전달합니다. 충분한 데이터가 생성되면 객체가 complete 이벤트를 전달합니다.

```
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 );
    }
}
}
```

참고: 예제에서 AsyncDataProvider 클래스는 ByteArray를 확장하므로 EventDispatcher도 확장할 수 없습니다. IEventDispatcher 인터페이스를 구현하기 위해 클래스에서는 내부 EventDispatcher 객체를 사용하고 IEventDispatcher 메서드 호출을 해당 내부 객체에 전달합니다. 또한 EventDispatcher를 확장하고 IDataInput(또는 두 인터페이스 모두)를 구현할 수 있습니다.

비동기 IFilePromise 구현은 동기 구현과 거의 동일합니다. 주요 차이점은 isAsync는 true를 반환하고, open() 메서드는 비동기 데이터 객체를 반환한다는 것입니다.

```
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 );
        }
    }
}
```

36장: 메뉴 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

컨텍스트 메뉴 API의 클래스를 사용하여 웹 기반 Flex 및 Flash Player 응용 프로그램의 컨텍스트 메뉴를 수정할 수 있습니다.

기본 메뉴 API의 클래스를 사용하여 Adobe® AIR®의 응용 프로그램, 윈도우, 컨텍스트 및 팝업 메뉴를 정의할 수 있습니다.

메뉴 기본 사항

Flash Player 9 이상, Adobe AIR 1.0 이상

AIR 응용 프로그램의 기본 메뉴 생성에 대한 간략한 설명과 코드 예제를 보려면 Adobe Developer Connection의 다음 퀵 스타트 문서를 참조하십시오.

- AIR 응용 프로그램에 기본 메뉴 추가(Flex)
- AIR 응용 프로그램에 기본 메뉴 추가(Flash)

기본 메뉴 클래스를 사용하면 응용 프로그램이 실행되고 있는 운영 체제의 기본 메뉴 기능에 액세스할 수 있습니다. NativeMenu 객체는 응용 프로그램 메뉴(Mac OS X에서 사용할 수 있음), 윈도우 메뉴(Windows 및 Linux에서 사용할 수 있음), 컨텍스트 메뉴 및 팝업 메뉴에 대해 사용할 수 있습니다.

AIR 응용 프로그램 외부에서는 컨텍스트 메뉴 클래스를 사용하여, 응용 프로그램에서 객체를 마우스 오른쪽 버튼으로 클릭하거나 cmd 키를 누른 상태에서 클릭할 때 Flash Player가 자동으로 표시하는 컨텍스트 메뉴를 수정할 수 있습니다. AIR 응용 프로그램에는 자동 컨텍스트 메뉴가 표시되지 않습니다.

메뉴 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

Menu 클래스에는 다음이 포함됩니다.

패키지	클래스
flash.display	• NativeMenu • NativeMenuItem
flash.ui	• ContextMenu • ContextMenuItem
flash.events	• Event • ContextMenuEvent

메뉴 유형

Flash Player 9 이상, Adobe AIR 1.0 이상

AIR에서는 다음 메뉴 유형을 지원합니다.

컨텍스트 메뉴 컨텍스트 명령은 SWF 내용의 대화형 객체 또는 HTML 내용의 문서 요소에서 오른쪽 버튼 클릭이나 명령 클릭에 응답하여 열립니다.

Flash Player 런타임에서는 컨텍스트 메뉴가 자동으로 표시됩니다. ContextMenu 클래스와 ContextMenuItem 클래스를 사용하여 메뉴에 자체적으로 만든 명령을 추가할 수 있습니다. 또한 내장 명령의 일부를 제거할 수도 있습니다. 단, 내장 명령을 전부 제거할 수는 없습니다.

AIR 런타임에서는 NativeMenu 클래스나 ContextMenu 클래스를 사용하여 컨텍스트 메뉴를 만들 수 있습니다. AIR의 HTML 내용에서는 Webkit HTML 및 JavaScript API를 사용하여 HTML 요소에 컨텍스트 메뉴를 추가할 수 있습니다.

응용 프로그램 메뉴(AIR에만 해당) 응용 프로그램 메뉴는 전체 응용 프로그램에 적용되는 전역 메뉴입니다. 응용 프로그램 메뉴는 Mac OS X에서는 지원되지만 Windows 또는 Linux에서는 지원되지 않습니다. Mac OS X의 경우 운영 체제에서 자동으로 응용 프로그램 메뉴를 만듭니다. AIR 메뉴 API를 사용하여 표준 메뉴에 항목 및 하위 메뉴를 추가할 수 있습니다. 또한 기존 메뉴 명령을 처리하는 리스너를 추가하거나 기존 항목을 제거할 수 있습니다.

윈도우 메뉴 (AIR에만 해당) 윈도우 메뉴는 하나의 윈도우에 연결되어 있으며 제목 표시줄 아래에 표시됩니다. NativeMenu 객체를 만들고 NativeWindow 객체의 menu 속성에 할당하여 윈도우에 메뉴를 추가할 수 있습니다. 윈도우 메뉴는 Windows 및 Linux 운영 체제에서는 지원되지만 Mac OS X에서는 지원되지 않습니다. 시스템 크롬이 있는 윈도우에서만 기본 윈도우 메뉴를 사용할 수 있습니다.

도크 및 시스템 트레이 아이콘 메뉴(AIR에만 해당) 이러한 아이콘 메뉴는 컨텍스트 메뉴와 유사하며 Mac OS X 도크 또는 작업 표시줄의 Windows 및 Linux 알림 영역에 있는 응용 프로그램 아이콘에 연결됩니다. 도크 및 시스템 트레이 아이콘 메뉴는 NativeMenu 클래스를 사용합니다. Mac OS X에서는 메뉴의 항목이 표준 운영 체제 항목 위에 추가됩니다. Windows 또는 Linux에는 표준 메뉴가 없습니다.

팝업 메뉴 (AIR에만 해당) AIR 팝업 메뉴는 컨텍트스 메뉴와 유사하지만 특정 응용 프로그램 객체 또는 구성 요소와 항상 연결되지는 않습니다. NativeMenu 객체의 display() 메서드를 호출하여 팝업 메뉴를 윈도우에서 원하는 위치에 표시할 수 있습니다.

사용자 정의 메뉴 기본 메뉴는 전적으로 운영 체제에 의해 그려지므로 Flash 및 HTML 렌더링 모델 외부에 존재합니다. 기본 메뉴를 사용하는 대신에 MXML, ActionScript 또는 JavaScript를 이용하여 사용자 정의된 나만의 비기본 메뉴를 언제든지 직접 만들 수 있습니다(AIR의 경우). 이렇게 만든 메뉴는 응용 프로그램 내용 안에서 완전하게 렌더링되어야 합니다.

Flex 메뉴 Adobe® Flex™ 프레임워크에서는 일련의 Flex 메뉴 구성 요소를 제공합니다. Flex 메뉴는 운영 체제가 아닌 런타임에서 그리며 기본 메뉴가 아닙니다. Flex 메뉴 구성 요소는 시스템 크롬이 없는 Flex 윈도우에 사용할 수 있습니다. Flex 메뉴 구성 요소를 사용하면 메뉴를 MXML 형식으로 선언하여 지정할 수 있다는 이점도 얻을 수 있습니다. Flex 프레임워크를 사용할 경우 윈도우 메뉴에 기본 클래스 대신 Flex 메뉴 클래스를 사용할 수 있습니다.

기본 메뉴(AIR에만 해당)

다음과 같은 기본 메뉴는 운영 체제나 내장 AIR 클래스에서 제공합니다.

- Mac OS X의 응용 프로그램 메뉴
- Mac OS X의 도크 아이콘 메뉴
- HTML 내용에서 선택한 텍스트 및 이미지에 대한 컨텍스트 메뉴
- TextField 객체(또는 TextField를 확장하는 객체)에서 선택한 텍스트에 대한 컨텍스트 메뉴

컨텍스트 메뉴 정보

Flash Player 9 이상, Adobe AIR 1.0 이상

SWF 내용에서 InteractiveObject를 상속하는 모든 객체에는 해당 contextMenu 속성에 메뉴 객체를 할당하여 컨텍스트 메뉴를 지정할 수 있습니다. [앞으로 이동], [뒤로 이동], [인쇄], [품질], [확대/축소] 등의 몇 가지 명령은 기본적으로 포함됩니다. AIR 런타임에서 contextMenu에 할당되는 메뉴 객체는 NativeMenu 유형이거나 ContextMenu 유형일 수 있습니다. Flash Player 런타임에서는 ContextMenu 클래스만 사용할 수 있습니다.

ContextMenu 및 ContextMenuItem 클래스를 사용할 때는 기본 메뉴 이벤트 또는 컨텍스트 메뉴 이벤트를 수신할 수 있습니다. 두 이벤트 모두 전달됩니다. ContextMenuEvent 객체 속성의 이점 한 가지는 contextMenuOwner가 메뉴가 연결된 객체를 식별하고 mouseTarget이 메뉴를 열기 위해 클릭한 객체를 식별한다는 것입니다. NativeMenuEvent 객체에서는 이 정보를 사용할 수 없습니다.

다음 예제에서는 Sprite를 만들어 단순한 편집 컨텍스트 메뉴에 추가합니다.

```
 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");}
```

참고: 브라우저 환경에 표시되는 SWF 내용과 달리 AIR의 컨텍스트 메뉴에는 내장 명령이 없습니다.

Flash Player 컨텍스트 메뉴 사용자 정의

브라우저나 프로젝터에서 SWF 내용의 컨텍스트 메뉴에는 항상 기본 제공 명령이 내장되어 있습니다. [설정] 및 [정보] 명령을 제외한 모든 기본 명령은 메뉴에서 제거할 수 있습니다. 스테이지 속성 showDefaultContextMenu를 false로 설정하면 컨텍스트 메뉴에서 이러한 명령이 제거됩니다.

특정한 표시 객체의 컨텍스트 메뉴를 사용자 정의하려면 ContextMenu 클래스의 새 인스턴스를 만들고 hideBuiltInItems() 메서드를 호출한 다음 이 인스턴스를 해당하는 DisplayObject 인스턴스의 contextMenu 속성에 지정합니다. 다음 예제에서는 동적으로 그려지는 정사각형에 임의의 색상으로 변경할 수 있는 컨텍스트 메뉴 명령을 제공합니다.

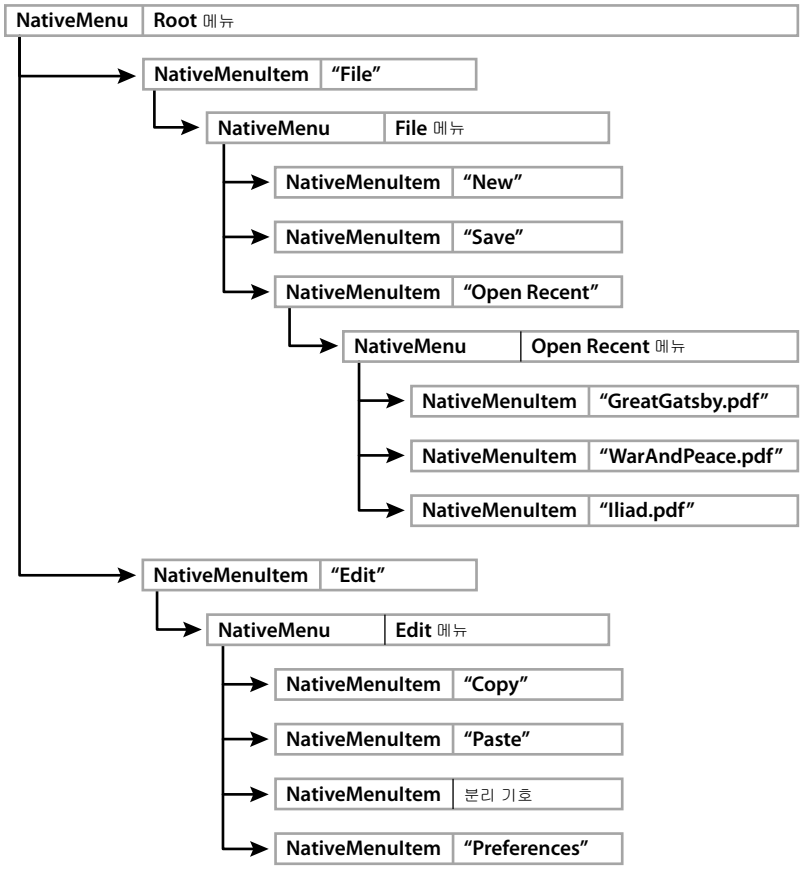
```
var square:Sprite = new Sprite();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0,0,100,100);
square.graphics.endFill();
square.x =
square.y = 10;
addChild(square);

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);
}
```

기본 메뉴 구조(AIR)

Adobe AIR 1.0 이상

기본 메뉴는 기본적으로 계층 구조를 가집니다. NativeMenu 객체에는 자식 NativeMenuItem 객체가 포함됩니다. 그리고 하위 메뉴를 나타내는 NativeMenuItem 객체에는 NativeMenu 객체가 포함될 수 있습니다. 구조에서 최상위 또는 루트 레벨 메뉴 객체는 응용 프로그램 및 윈도우 메뉴의 메뉴 모음을 나타냅니다. 컨텍스트, 아이콘 및 팝업 메뉴에는 메뉴 모음이 없습니다.

다음 다이어그램에서는 일반적인 메뉴의 구조를 보여 줍니다. 루트 메뉴는 메뉴 모음을 나타내며 **File** 하위 메뉴와 **Edit** 하위 메뉴를 참조하는 두 개의 메뉴 항목을 포함합니다. 이 구조에서 File 하위 메뉴에는 자체적으로 세 개의 항목을 포함한 **Open Recent Menu** 하위 메뉴와 두 개의 명령 항목이 포함됩니다. Edit 하위 메뉴에는 세 개의 명령과 분리 기호가 포함됩니다.



하위 메뉴를 정의하려면 NativeMenu 및 NativeMenuItem 객체가 모두 필요합니다. NativeMenuItem 객체는 부모 메뉴에 표시되는 레이블을 정의하고 사용자가 하위 메뉴를 열 수 있게 합니다. NativeMenu 객체는 하위 메뉴의 항목에 대한 컨테이너 역할을 합니다. NativeMenuItem 객체는 NativeMenuItem submenu 속성을 통해 NativeMenu 객체를 참조합니다.

이 메뉴를 만드는 코드 예제를 보려면 586페이지의 "기본 메뉴 예제: 윈도우 및 응용 프로그램 메뉴(AIR)"를 참조하십시오.

메뉴 이벤트

Adobe AIR 1.0 이상

NativeMenu 및 NativeMenuItem 객체는 모두 preparing, displaying, select 이벤트를 전달합니다.

preparing: 객체가 사용자 상호 작용을 시작할 때마다 메뉴 및 메뉴 항목은 등록된 모든 리스너에 preparing 이벤트를 전달합니다. 상호 작용에는 키보드 단축키를 사용한 메뉴 열기 또는 항목 선택 등이 포함됩니다.

참고: preparing 이벤트는 Adobe AIR 2.6 이상에서만 사용 가능합니다.

displaying: 메뉴가 표시되기 직전에 메뉴와 해당 메뉴 항목은 등록된 리스너에 displaying 이벤트를 전달합니다.

preparing 및 displaying 이벤트를 사용하면 메뉴 내용 또는 항목 모양을 사용자에게 보여 주기 전에 업데이트할 수 있습니다. 예를 들어 "최근에 사용한 파일 열기" 메뉴의 displaying 이벤트에 대한 리스너에서 최근에 본 문서의 최신 목록을 반영하여 메뉴 항목을 변경할 수 있습니다.

키보드 단축키가 preparing 이벤트를 트리거한 메뉴 항목을 제거하면 해당되는 메뉴 상호 작용이 실질적으로 취소되고 select 이벤트가 전달되지 않습니다.

이벤트의 target 및 currentTarget 속성은 모두 리스너가 등록되는 객체이며, 메뉴 자체 또는 메뉴 항목 중 하나가 됩니다.

preparing 이벤트는 displaying 이벤트에 앞서 전달됩니다. 일반적으로 두 이벤트는 한 번에 하나만 수신할 수 있습니다.

select: 사용자가 명령 항목을 선택하면 항목에서는 등록된 리스너에 select 이벤트를 전달합니다. 하위 메뉴와 분리 기호 항목은 선택할 수 없으므로 select 이벤트를 전달하지 않습니다.

select 이벤트는 메뉴 항목에서 이 항목을 포함하는 메뉴로, 최종적으로 루트 메뉴로 버블링됩니다. select 이벤트를 항목에서 직접 수신하거나 메뉴 구조의 상위에서 수신할 수 있습니다. select 이벤트를 메뉴에서 수신하는 경우 이벤트의 target 속성을 사용하여 선택한 항목을 식별할 수 있습니다. 이벤트가 메뉴 계층 구조에서 버블링될 때 이벤트 객체의 currentTarget 속성은 현재 메뉴 객체를 식별합니다.

참고: ContextMenu 및 ContextMenuItem 객체는 select, preparing 및 displaying 이벤트뿐 아니라 menuItemSelect 및 menuSelect 이벤트를 전달합니다.

기본 메뉴 명령에 해당하는 키(AIR)

Adobe AIR 1.0 이상

메뉴 명령에 해당하는 키(엑셀러레이터라고도 함)를 할당할 수 있습니다. 이 키 또는 키 조합을 누르면 메뉴 항목에서는 등록된 리스너에 select 이벤트를 전달합니다. 항목을 포함하는 메뉴는 호출될 명령의 응용 프로그램 또는 활성 윈도우 메뉴에 포함되어 있어야 합니다.

해당하는 키는 기본 키를 나타내는 문자열과 함께 눌러야 하는 수정자 키 배열의 두 부분으로 이루어집니다. 기본 키를 할당하려면 메뉴 항목의 keyEquivalent 속성을 해당 키의 단일 문자열로 설정합니다. 대문자를 사용하는 경우 수정자 배열에 shift 키가 자동으로 추가됩니다.

Mac OS X에서는 기본 수정자가 Command 키(Keyboard.COMMAND)입니다. Windows 및 Linux에서는 Ctrl 키(Keyboard.CONTROL)입니다. 이러한 기본 키는 수정자 배열에 자동으로 추가됩니다. 다른 수정자 키를 할당하려면 원하는 키 코드가 포함된 새 배열을 keyEquivalentModifiers 속성에 추가합니다. 기본 배열은 덮어씁니다. 기본 수정자를 사용하거나 사용자 고유의 수정자 배열을 할당하면 keyEquivalent 속성에 할당하는 문자열이 대문자를 사용하는 경우 Shift 키가 추가됩니다. 수정자 키에 사용할 키 코드에 대한 상수는 Keyboard 클래스에서 정의합니다.

할당된 키 문자열이 메뉴 항목 이름 옆에 자동으로 표시됩니다. 형식은 사용자의 운영 체제 및 시스템 환경 설정에 따라 다릅니다.

참고: Keyboard.COMMAND 값을 Windows 운영 체제의 키 수정자 배열에 할당하면 해당하는 키가 메뉴에 표시되지 않습니다. 그러나 메뉴 명령을 활성화하려면 Crtl 키를 시용해야 합니다.

다음 예제에서는 메뉴 항목에 해당하는 키로 Ctrl+Shift+G를 할당합니다.

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
```

이 예제에서는 수정자 배열을 직접 설정하여 Ctrl+Shift+G를 해당하는 키로 할당합니다.

```
var item:NativeMenuItem = new NativeMenuItem("Ungroup");
item.keyEquivalent = "G";
item.keyEquivalentModifiers = [Keyboard.CONTROL];
```

참고: 해당하는 키는 응용 프로그램 및 윈도우 메뉴에 대해서만 트리거됩니다. 해당하는 키를 컨텍스트나 팝업 메뉴에 추가하면 해당하는 키가 메뉴 레이블에 표시되지만 연결된 메뉴 명령은 호출되지 않습니다.

니모닉 (AIR)

Adobe AIR 1.0 이상

니모닉은 메뉴에 대한 운영 체제 키보드 인터페이스의 일부입니다. Linux, Mac OS X 및 Windows 모두에서 사용자가 키보드를 사용하여 메뉴를 열고 명령을 선택할 수 있지만 서로 약간의 차이가 있습니다.

Mac OS X에서는 메뉴나 명령의 첫 번째 문자나 두 문자를 입력한 다음 return 키를 누릅니다. mnemonicIndex 속성은 무시됩니다.

Windows에서는 단일 문자만 중요합니다. 기본적으로 중요한 문자는 레이블의 첫 번째 문자이지만 메뉴 항목에 니모닉을 할당하는 경우 지정하는 문자가 중요한 문자가 됩니다. 니모닉 할당 여부와 관계없이 메뉴의 두 항목이 중요한 문자가 동일한 경우 사용자 키보드와 메뉴의 상호 작용이 약간 변경됩니다. 이 경우 단일 문자를 눌러 메뉴나 명령을 선택하지 않고 문자를 필요한 만큼 여러 번 눌러 원하는 항목을 강조 표시한 다음 Enter 키를 눌러 선택을 완료해야 합니다. 일관된 비헤이비어를 유지하려면 윈도우 메뉴의 각 메뉴 항목에 고유한 니모닉을 할당하는 것이 좋습니다.

Linux에서는 기본 니모닉이 제공되지 않습니다. 니모닉을 제공하려면 mnemonicIndex 속성의 값을 지정해야 합니다.

니모닉 문자를 레이블 문자열에 대한 인덱스로 지정합니다. 레이블에서 첫 번째 문자의 인덱스는 0입니다. 따라서 “Format”이라는 메뉴 항목에 “r”을 니모닉으로 사용하려면 mnemonicIndex 속성을 2로 설정합니다.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.mnemonicIndex = 2;
```

메뉴 항목 상태

Adobe AIR 1.0 이상

메뉴 항목에는 checked 및 enabled의 두 가지 상태 속성이 있습니다.

checked true로 설정하면 항목 레이블 다음에 확인 표시를 표시할 수 있습니다.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.checked = true;
```

enabled true 및 false 사이를 전환하여 명령이 활성화되어 있는지 여부를 제어할 수 있습니다. 비활성화된 항목은 "회색으로 표시되며” select 이벤트를 전달하지 않습니다.

```
var item:NativeMenuItem = new NativeMenuItem("Format");
item.enabled = false;
```

메뉴 항목에 객체 첨부

Adobe AIR 1.0 이상

NativeMenuItem 클래스의 data 속성을 사용하여 각 항목에 있는 임의의 객체를 참조할 수 있습니다. 예를 들어 “Open Recent” 메뉴에서는 각 메뉴 항목에 각 문서의 File 객체를 할당할 수 있습니다.

```
var file:File = File.applicationStorageDirectory.resolvePath("GreatGatsby.pdf")
var menuItem:NativeMenuItem = docMenu.addItem(new NativeMenuItem(file.name));
menuItem.data = file;
```

기본 메뉴 만들기 (AIR)

Adobe AIR 1.0 이상

이 항목에서는 AIR에서 지원하는 여러 유형의 기본 메뉴를 만드는 방법에 대해 설명합니다.

루트 메뉴 객체 만들기

Adobe AIR 1.0 이상

메뉴의 루트로 사용할 NativeMenu 객체를 만들려면 NativeMenu 생성자를 사용합니다.

```
var root:NativeMenu = new NativeMenu();
```

응용 프로그램 및 윈도우 메뉴의 경우 루트 메뉴는 메뉴 표시줄을 나타내며 하위 메뉴를 여는 항목만 포함해야 합니다. 컨텍스트 메뉴와 팝업 메뉴에는 메모 표시줄이 없으므로 루트 메뉴에 하위 메뉴뿐 아니라 명령 및 분리 기호 선도 포함될 수 있습니다.

메뉴를 만든 후 메뉴 항목을 추가할 수 있습니다. 메뉴 객체의 addItemAt() 메서드를 사용하여 항목을 특정 인덱스에 추가하는 경우를 제외하고 메뉴의 항목은 추가되는 순서대로 표시됩니다.

다음 단원에 나와 있는 대로 메뉴를 응용 프로그램, 윈도우, 아이콘 또는 컨텍스트 메뉴로 할당하거나 팝업 메뉴로 표시합니다.

응용 프로그램 메뉴 설정 또는 윈도우 메뉴 설정

코드에서 응용 프로그램 메뉴(Mac OS에서 지원됨) 및 윈도우 메뉴(기타 운영 체제에서 지원됨)를 모두 수용할 수 있어야 합니다.

```
var root:NativeMenu = new NativeMenu();
if (NativeApplication.supportsMenu)
{
    NativeApplication.nativeApplication.menu = root;
}
else if (NativeWindow.supportsMenu)
{
    nativeWindow.menu = root;
}
```

참고: Mac OS에서는 모든 응용 프로그램의 표준 항목이 포함된 메뉴를 정의합니다. NativeApplication 객체의 menu 속성에 새 NativeMenu 객체를 할당하면 표준 메뉴가 대체됩니다. 또한 표준 메뉴를 대체하지 않고 사용할 수도 있습니다.

Adobe Flex에서는 여러 플랫폼에서 작동하는 메뉴를 쉽게 만들 수 있도록 FlexNativeMenu 클래스를 제공합니다. Flex 프레임워크를 사용할 경우 NativeMenu 클래스 대신 FlexNativeMenu 클래스를 사용하십시오.

대화형 객체에서 컨텍스트 메뉴 설정

```
interactiveObject.contextMenu = root;
```

도크 아이콘 메뉴 설정 또는 시스템 트레이 아이콘 메뉴 설정

코드에서 응용 프로그램 메뉴(Mac OS에서 지원됨) 및 윈도우 메뉴(기타 운영 체제에서 지원됨)를 모두 수용할 수 있어야 합니다.

```
if (NativeApplication.supportsSystemTrayIcon)
{
    SystemTrayIcon(NativeApplication.nativeApplication.icon).menu = root;
}
else if (NativeApplication.supportsDockIcon)
{
    DockIcon(NativeApplication.nativeApplication.icon).menu = root;
}
```

참고: Mac OS X에서는 응용 프로그램 도크 아이콘에 대한 표준 메뉴를 정의합니다. DockIcon 객체의 menu 속성에 새 NativeMenu를 할당하면 해당 메뉴의 항목이 표준 항목 위에 표시됩니다. 그러면 표준 메뉴 항목을 제거, 액세스 또는 수정할 수 없습니다.

메뉴를 팝업으로 표시

```
root.display(stage, x, y);
```

기타 도움말 항목

플랫폼의 영향을 받지 않는 AIR 응용 프로그램 개발

하위 메뉴 만들기

Adobe AIR 1.0 이상

하위 메뉴를 만들려면 부모 메뉴에 NativeMenuItem 객체를 추가한 다음 하위 메뉴를 정의하는 NativeMenu 객체를 항목의 submenu 속성에 할당합니다. AIR에서는 하위 메뉴 항목 및 이 항목과 연결된 메뉴 객체를 만드는 두 가지 방법을 제공합니다.

addSubmenu() 메서드를 사용하면 메뉴 항목 및 관련된 메뉴 객체를 하나의 단계로 만들 수 있습니다.

```
var editMenuItem:NativeMenuItem = root.addSubmenu(new NativeMenu(), "Edit");
```

또한 메뉴 항목을 만들고 해당 submenu 속성에 메뉴 객체를 별도로 할당할 수도 있습니다.

```
var editMenuItem:NativeMenuItem = root.addItem("Edit", false);
editMenuItem.submenu = new NativeMenu();
```

메뉴 명령 만들기

Adobe AIR 1.0 이상

메뉴 명령을 만들려면 메뉴에 NativeMenuItem 객체를 추가하고 메뉴 명령을 구현하는 함수를 참조하는 이벤트 리스너를 추가합니다.

```
var copy:NativeMenuItem = new NativeMenuItem("Copy", false);
copy.addEventListener(Event.SELECT, onCopyCommand);
editMenu.addItem(copy);
```

예제와 같이 명령 항목 자체에서 select 이벤트를 수신하거나 부모 메뉴 객체에서 select 이벤트를 수신할 수 있습니다.

참고: 하위 메뉴와 분리 기호 선을 나타내는 메뉴 항목은 select 이벤트를 전달하지 않으므로 명령으로 사용할 수 없습니다.

메뉴 분리 기호 선 만들기

Adobe AIR 1.0 이상

분리 기호 선을 만들려면 생성자의 isSeparator 매개 변수를 true로 설정하여 NativeMenuItem을 만듭니다. 그런 다음 메뉴의 올바른 위치에 분리 기호 항목을 추가합니다.

```
var separatorA:NativeMenuItem = new NativeMenuItem("A", true);
editMenu.addItem(separatorA);
```

분리 기호에 지정하는 레이블(있는 경우)은 표시되지 않습니다.

HTML의 컨텍스트 메뉴(AIR)

Adobe AIR 1.0 이상

HTMLLoader 객체를 사용하여 표시되는 HTML 내용에서 contextmenu 이벤트를 사용하면 컨텍스트 메뉴를 표시할 수 있습니다. 기본적으로 텍스트를 마우스 오른쪽 버튼으로 클릭하거나 Command 키를 누른 상태로 클릭하여 선택한 텍스트에 대해 컨텍스트 메뉴 이벤트를 호출하면 컨텍스트 메뉴가 표시됩니다. 기본 메뉴가 열리지 않게 하려면 contextmenu 이벤트를 수신하고 이벤트 객체의 preventDefault() 메서드를 호출합니다.

```
function showContextMenu(event){
    event.preventDefault();
}
```

그런 다음 DHTML 기술을 사용하거나 AIR 기본 컨텍스트 메뉴를 표시하여 사용자 정의 컨텍스트 메뉴를 표시할 수 있습니다. 다음 예제에서는 HTML contextmenu 이벤트에 대한 응답으로 메뉴의 display() 메서드를 호출하여 기본 컨텍스트 메뉴를 표시합니다.

```
<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>
```

팝업 기본 메뉴 표시(AIR)

Adobe AIR 1.0 이상

메뉴의 display() 메서드를 호출하여 NativeMenu 객체를 언제든 윈도우 위의 임의 위치에 표시할 수 있습니다. 이 메서드에는 스테이지에 대한 참조가 필요하므로 응용 프로그램 샌드박스의 내용만 메뉴를 팝업으로 표시할 수 있습니다.

다음 메서드는 마우스 클릭에 응답하여 popupMenu라는 NativeMenu 객체에 정의된 메뉴를 표시합니다.

```
private function onMouseClick(event:MouseEvent):void {
    popupMenu.display(event.target.stage, event.stageX, event.stageY);
}
```

참고: 이 메뉴는 이벤트에 대한 직접 응답으로 표시되지 않을 수도 있습니다. 모든 메서드에서 display() 함수를 호출할 수 있습니다.

메뉴 이벤트 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 메뉴를 선택하거나 메뉴 항목을 선택할 때 메뉴에서 이벤트를 전달합니다.

메뉴 클래스에 대한 이벤트 요약

Flash Player 9 이상, Adobe AIR 1.0 이상

메뉴나 개별 항목에 메뉴 이벤트를 처리할 이벤트 리스너를 추가합니다.

Object	전달되는 이벤트
NativeMenu(AIR)	Event.PREPARING(Adobe AIR 2.6 이상) Event.DISPLAYING Event.SELECT(자식 항목 및 하위 메뉴에서 전달)
NativeMenuItem(AIR)	Event.PREPARING(Adobe AIR 2.6 이상) Event.SELECT Event.DISPLAYING(부모 메뉴에서 전달)
ContextMenu	ContextMenuEvent.MENU_SELECT
ContextMenuItem	ContextMenuEvent.MENU_ITEM_SELECT Event.SELECT(AIR)

Select 메뉴 이벤트

Adobe AIR 1.0 이상

메뉴 항목에 대한 클릭을 처리하려면 NativeMenuItem 객체에 select 이벤트에 대한 이벤트 리스너를 추가합니다.

```
var menuCommandX:NativeMenuItem = new NativeMenuItem("Command X");
menuCommandX.addEventListener(Event.SELECT, doCommandX)
```

select 이벤트는 포함하는 메뉴로 버블링되므로 부모 메뉴의 select 이벤트도 수신할 수 있습니다. 메뉴 레벨에서 수신할 경우 이벤트 객체 target 속성을 사용하여 선택된 메뉴 명령을 확인할 수 있습니다. 다음 예제에서는 선택된 명령의 레이블을 추적합니다.

```
 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");
}
```

ContextMenuItem 클래스를 사용할 경우 select 이벤트 또는 menuItemSelect 이벤트를 수신할 수 있습니다. menuItemSelect 이벤트는 컨텍스트 메뉴를 소유하는 객체에 대한 추가 정보를 제공하지만 포함하는 메뉴로 버블링되지 않습니다.

displaying 메뉴 이벤트

Adobe AIR 1.0 이상

메뉴 열기를 처리하려면 메뉴가 표시되기 전에 전달되는 displaying 이벤트에 대한 리스너를 추가할 수 있습니다. displaying 이벤트를 사용하면 메뉴를 업데이트할 수 있습니다. 예를 들어 항목을 추가 또는 제거하거나 개별 항목의 활성화 또는 선택 상태를 업데이트할 수 있습니다. ContextMenu 객체에서 menuSelect 이벤트를 수신할 수도 있습니다.

AIR 2.6 이상의 버전에서는 preparing 이벤트를 통해 키보드 단축키를 사용한 메뉴 표시 또는 항목 선택에 대한 응답으로 메뉴를 업데이트할 수 있습니다.

기본 메뉴 예제: 윈도우 및 응용 프로그램 메뉴(AIR)

Adobe AIR 1.0 이상

다음 예제에서는 578페이지의 “기본 메뉴 구조(AIR)”에 나와 있는 메뉴를 만듭니다.

이 메뉴는 윈도우 메뉴만 지원되는 Windows와 응용 프로그램 메뉴만 지원되는 Mac OS X 모두에서 작동하도록 설계되었습니다. 이 두 운영 체제를 구별하기 위해 MenuExample 클래스 생성자에서는 NativeWindow 및 NativeApplication 클래스의 supportsMenu 속성을 확인합니다. NativeWindow.supportsMenu가 true이면 생성자는 윈도우의 NativeMenu 객체를 만든 다음 File 및 Edit 하위 메뉴를 만들어 추가합니다. NativeApplication.supportsMenu가 true이면 생성자는 File 및 Edit 메뉴를 만들어 Mac OS X 운영 체제에서 제공하는 기존 메뉴에 추가합니다.

이 예제에서는 메뉴 이벤트 처리에 대해서도 설명합니다. select 이벤트는 항목 레벨과 메뉴 레벨에서 처리됩니다. 선택한 항목을 포함하는 메뉴에서 루트 메뉴에 이르는 체인의 각 메뉴가 select 이벤트에 응답합니다. displaying 이벤트는 "Open Recent" 메뉴에서 사용됩니다. 메뉴가 열리기 바로 전에 최신 Documents 배열에서 메뉴의 항목이 새로 고쳐집니다(이 예제에서는 실제로 변경되지 않음). 이 예제에는 나와 있지 않지만 개별 항목에서도 displaying 이벤트를 수신할 수 있습니다.

```
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;
        }
    }
}
```

37장: AIR의 작업 표시줄 아이콘

Adobe AIR 1.0 이상

많은 운영 체제에서 응용 프로그램을 나타내는 아이콘이 포함될 수 있는 Mac OS X 도크와 같은 작업 표시줄을 제공합니다. Adobe® AIR®에서는 NativeApplication.nativeApplication.icon 속성을 통해 응용 프로그램 작업 표시줄 아이콘과 상호 작용하기 위한 인터페이스를 제공합니다.

- 시스템 트레이 및 도크 아이콘 사용 (Flex)
- 시스템 트레이 및 도크 아이콘 사용(Flash)

기타 도움말 항목

flash.desktop.NativeApplication

flash.desktop.DockIcon

flash.desktop.SystemTrayIcon

작업 표시줄 아이콘

Adobe AIR 1.0 이상

AIR에서는 NativeApplication.nativeApplication.icon 객체를 자동으로 만듭니다. 객체 형식은 운영 체제에 따라 DockIcon 또는 SystemTrayIcon입니다. NativeApplication.supportsDockIcon 및 NativeApplication.supportsSystemTrayIcon 속성을 사용하여 이러한 InteractiveIcon 하위 클래스 중에서 AIR이 현재 운영 체제에서 지원하는 하위 클래스를 확인할 수 있습니다. InteractiveIcon 기본 클래스는 아이콘에 사용되는 이미지를 변경하는 데 사용할 수 있는 width, height 및 bitmaps 속성을 제공합니다. 그러나 잘못된 운영 체제에서 DockIcon 또는 SystemTrayIcon과 관련된 속성에 액세스하면 런타임 오류가 생성됩니다.

아이콘에 사용되는 이미지를 설정하거나 변경하려면 하나 이상의 이미지가 포함된 배열을 만들어 NativeApplication.nativeApplication.icon.bitmaps 속성에 할당합니다. 작업 표시줄 아이콘의 크기는 운영 체제에 따라 다를 수 있습니다. 크기 조절로 인한 이미지 품질 저하를 방지하려면 여러 크기의 이미지를 bitmaps 배열에 추가하면 됩니다. 둘 이상의 이미지를 제공하는 경우 AIR에서는 작업 표시줄 아이콘의 현재 표시 크기에 가장 가까운 크기를 선택하고 필요한 경우에만 크기를 조절합니다. 다음 예제에서는 두 이미지를 사용하여 작업 표시줄 아이콘의 이미지를 설정합니다.

```
NativeApplication.nativeApplication.icon.bitmaps =
            [bmp16x16.bitmapData, bmp128x128.bitmapData];
```

아이콘 이미지를 변경하려면 새 이미지가 하나 이상 포함된 배열을 bitmaps 속성에 할당합니다. enterFrame 또는 timer 이벤트에 대한 응답으로 이미지를 변경하여 아이콘에 애니메이션을 적용할 수 있습니다.

Windows 및 Linux의 알림 영역에서 아이콘을 제거하거나 Mac OS X에서 기본 아이콘 모양을 복원하려면 bitmaps를 빈 배열로 설정합니다.

```
NativeApplication.nativeApplication.icon.bitmaps = [];
```

도크 아이콘

Adobe AIR 1.0 이상

AIR에서는 NativeApplication.supportsDockIcon이 true일 때 도크 아이콘을 지원합니다.
NativeApplication.nativeApplication.icon 속성은 도크의 응용 프로그램 아이콘을 나타냅니다(윈도우 도크 아이콘이 아님).

참고: AIR에서는 Mac OS X에서 도크의 윈도우 아이콘을 변경하는 작업을 지원하지 않습니다. 또한 응용 프로그램 도크 아이콘의 변경은 응용 프로그램이 실행되는 동안에만 적용됩니다. 응용 프로그램이 종료되면 아이콘이 일반 모양으로 되돌아갑니다.

도크 아이콘 메뉴

Adobe AIR 1.0 이상

명령이 포함된 NativeMenu 객체를 만들어 NativeApplication.nativeApplication.icon.menu 속성에 할당하여 표준 도크 메뉴에 명령을 추가할 수 있습니다. 메뉴의 항목은 표준 도크 아이콘 메뉴 항목 위에 표시됩니다.

도크 튀어오르게 하기

Adobe AIR 1.0 이상

NativeApplication.nativeApplication.icon.bounce() 메서드를 호출하여 도크 아이콘을 튀어오르게 할 수 있습니다. bounce() priority 매개 변수를 informational로 설정하면 아이콘이 한 번 튀어오릅니다. 이 매개 변수를 critical로 설정하면 사용자가 응용 프로그램을 활성화할 때까지 아이콘이 튀어오릅니다. priority 매개 변수의 상수는 NotificationType 클래스에 정의되어 있습니다.

참고: 응용 프로그램이 이미 활성화되어 있으면 아이콘이 튀어오르지 않습니다.

도크 아이콘 이벤트

Adobe AIR 1.0 이상

도크 아이콘을 클릭하면 NativeApplication 객체가 invoke 이벤트를 전달합니다. 응용 프로그램이 실행 중이 아니면 시스템에서 응용 프로그램을 시작하고, 그렇지 않으면 invoke 이벤트가 실행 중인 응용 프로그램 인스턴스에 전달됩니다.

시스템 트레이 아이콘

Adobe AIR 1.0 이상

AIR에서는 NativeApplication.supportsSystemTrayIcon이 true일 때(Windows 및 대부분의 Linux 배포판에서 현재 유일한 경우임) 시스템 트레이 아이콘을 지원합니다. Windows 및 Linux에서 시스템 트레이 아이콘은 작업 표시줄의 알림 영역에 표시됩니다. 기본적으로 아이콘이 표시되지 않습니다. 아이콘을 표시하려면 BitmapData 객체가 포함된 배열을 아이콘의 bitmaps 속성에 할당합니다. 아이콘 이미지를 변경하려면 새 이미지가 포함된 배열을 bitmaps에 할당합니다. 아이콘을 제거하려면 bitmaps를 null로 설정합니다.

시스템 트레이 아이콘 메뉴

Adobe AIR 1.0 이상

NativeMenu 객체를 만들어 NativeApplication.nativeApplication.icon.menu 속성에 할당하여 시스템 트레이 아이콘에 메뉴를 추가할 수 있습니다(기본 메뉴가 운영 체제에서 제공되지 않음). 시스템 트레이 아이콘 메뉴에 액세스하려면 아이콘을 마우스 오른쪽 버튼으로 클릭합니다.

시스템 트레이 아이콘 도구 설명

Adobe AIR 1.0 이상

tooltip 속성을 설정하여 아이콘에 도구 설명을 추가합니다.

```
NativeApplication.nativeApplication.icon.tooltip = "Application name";
```

시스템 트레이 아이콘 이벤트

Adobe AIR 1.0 이상

NativeApplication.nativeApplication.icon 속성이 참조하는 SystemTrayIcon 객체는 click, mouseDown, mouseUp, rightClick, rightMouseDown 및 rightMouseUp 이벤트에 대한 ScreenMouseEvent를 전달합니다. 이러한 이벤트를 아이콘 메뉴와 함께 사용하여 사용자가 표시되는 윈도우가 없는 응용 프로그램과 상호 작용하게 할 수 있습니다.

예제: 윈도우가 없는 응용 프로그램 만들기

Adobe AIR 1.0 이상

다음 예제에서는 시스템 트레이 아이콘이 있지만 표시되는 윈도우가 없는 AIR 응용 프로그램을 만듭니다. 응용 프로그램 설명자에서 응용 프로그램의 visible 속성을 true로 설정하지 않아야 합니다. 설정할 경우 응용 프로그램이 시작될 때 해당 윈도우가 표시됩니다.

```
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];
        }
    }
}
```

참고: Flex WindowedApplication 구성 요소를 사용하는 경우 WindowedApplication 태그의 visible 특성을 false로 설정해야 합니다. 이 특성은 응용 프로그램 설명자의 설정을 대체합니다.

참고: 이 예제에서는 응용 프로그램의 icons 하위 디렉토리에 AIRApp_16.png 및 AIRApp_128.png라는 이미지 파일이 있다고 가정합니다. 프로젝트 폴더에 복사할 수 있는 샘플 아이콘 파일이 AIR SDK에 포함되어 있습니다.

윈도우 작업 표시줄 아이콘 및 버튼

Adobe AIR 1.0 이상

일반적으로 윈도우를 아이콘으로 표현한 항목은 사용자가 배경 윈도우나 최소화된 윈도우에 쉽게 액세스할 수 있도록 도크나 작업 표시줄의 윈도우 영역에 표시됩니다. Mac OS X 도크에는 응용 프로그램의 아이콘과 최소화된 각 윈도우의 아이콘이 표시됩니다. Microsoft Windows 및 Linux 작업 표시줄에는 응용 프로그램의 각 일반 유형 윈도우에 대한 프로그램 아이콘과 제목이 포함된 버튼이 표시됩니다.

작업 표시줄 윈도우 버튼 강조 표시

Adobe AIR 1.0 이상

윈도우가 배경에 있으면 윈도우와 관련된 이벤트가 발생했음을 사용자에게 알릴 수 있습니다. Mac OS X에서는 591페이지의 “도크 튀어오르게 하기”에서 설명한 대로 응용 프로그램 도크 아이콘을 튀어오르게 하여 사용자에게 알릴 수 있습니다. Windows 및 Linux에서는 NativeWindow 인스턴스의 notifyUser() 메서드를 호출하여 윈도우 작업 표시줄 버튼을 강조 표시할 수 있습니다. 이 메서드에 전달된 type 매개 변수는 알림의 긴급도를 결정합니다.

- NotificationType.CRITICAL: 사용자가 윈도우를 전경으로 가져올 때까지 윈도우 아이콘이 깜박입니다.
- NotificationType.INFORMATIONAL: 윈도우 아이콘이 색상을 변경하여 강조 표시됩니다.

참고: Linux에서는 informational 유형의 알림만 지원되므로 둘 중 어느 유형의 값을 notifyUser() 함수에 전달해도 같은 효과를 냅니다.

다음 문은 윈도우의 작업 표시줄 버튼을 강조 표시합니다.

```
stage.nativeWindow.notifyUser(NotificationType.CRITICAL);
```

윈도우 수준 알림을 지원하지 않는 운영 체제에서 NativeWindow.notifyUser() 메서드를 호출하면 아무 효과가 없습니다. 윈도우 알림이 지원되는지 확인하려면 NativeWindow.supportsNotification 속성을 사용합니다.

작업 표시줄 버튼이나 아이콘이 없는 윈도우 만들기

Adobe AIR 1.0 이상

Windows 운영 체제에서 **utility** 또는 **lightweight** 유형으로 만든 윈도우는 작업 표시줄에 나타나지 않습니다. 표시되지 않는 윈도우도 작업 표시줄에 나타나지 않습니다.

윈도우가 작업 표시줄에 표시되지 않는 응용 프로그램을 만들려면 초기 윈도우가 **normal** 유형이어야 하므로 초기 윈도우를 닫거나 표시되지 않게 두어야 합니다. 응용 프로그램을 종료하지 않고 응용 프로그램의 모든 윈도우를 닫으려면 마지막 윈도우를 닫기 전에 NativeApplication 객체의 autoExit 속성을 false로 설정합니다. 초기 윈도우가 계속 표시되지 않게 하려면 응용 프로그램 설명자 파일의 <initalWindow> 요소에 <visible>false</visible>를 추가합니다(visible 속성을 true로 설정하거나 윈도우의 activate() 메서드를 호출하지 않음).

응용 프로그램에서 연 새 윈도우에서 윈도우 생성자에 전달된 NativeWindowInitOption 객체의 type 속성을 NativeWindowType.UTILITY 또는 NativeWindowType.LIGHTWEIGHT로 설정합니다.

Mac OS X에서는 최소화된 윈도우가 도크 작업 표시줄에 표시됩니다. 윈도우를 최소화하는 대신 숨겨서 최소화된 아이콘이 표시되지 않게 할 수 있습니다. 다음 예제에서는 nativeWindowDisplayState 변경 이벤트를 수신하고 윈도우가 최소화되는 중이면 최소화를 취소합니다. 대신 핸들러는 윈도우 visible 속성을 false로 설정합니다.

```
private function preventMinimize(event:NativeWindowDisplayStateEvent):void{
    if(event.afterDisplayState == NativeWindowDisplayState.MINIMIZED){
        event.preventDefault();
        event.target.visible = false;
    }
}
```

visible 속성을 false로 설정할 때 Mac OS X 도크에서 윈도우가 최소화되면 도크 아이콘이 제거되지 않습니다. 사용자는 여전히 아이콘을 클릭하여 윈도우를 다시 표시할 수 있습니다.

38장: 파일 시스템 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash® Player에서 FileReference 클래스를 통해 기본적인 파일 읽기 및 쓰기 기능을 수행할 수 있습니다. Flash Player를 실행할 때는 보안상의 이유로 항상 해당하는 권한이 있어야만 파일 읽기 또는 쓰기 작업을 할 수 있습니다.

Adobe® AIR®에서는 Flash Player에서보다 호스트 컴퓨터의 파일 시스템에 더욱 완전하게 액세스할 수 있습니다. AIR 파일 시스템 API를 사용하여 디렉토리 및 파일을 액세스, 관리하고 디렉토리 및 파일을 만들고 파일에 데이터를 쓰는 등의 작업을 수행할 수 있습니다.

기타 도움말 항목

flash.net.FileReference

flash.net.FileReferenceList

flash.filesystem.File

flash.filesystem.FileStream

FileReference 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

FileReference 객체는 클라이언트 또는 서버 컴퓨터의 데이터 파일을 나타냅니다. FileReference 클래스의 메서드를 사용하면 응용 프로그램에서 로컬로 데이터 파일을 로드 및 저장하고 원격 서버와 데이터 파일을 주고 받을 수 있습니다.

FileReference 클래스에서는 두 가지 방법으로 데이터 파일을 로드, 전송 및 저장할 수 있습니다. FileReference 클래스는 처음 소개되었을 때부터 browse() 메서드, upload() 메서드, download() 메서드가 포함되었습니다. browse() 메서드를 사용하면 사용자가 파일을 선택하도록 할 수 있습니다. upload() 메서드를 사용하면 파일 데이터를 원격 서버에 전송할 수 있습니다. download() 메서드를 사용하면 서버에서 데이터를 검색하고 이를 로컬 파일에 저장할 수 있습니다. Flash Player 10 및 Adobe AIR 1.5부터는 FileReference 클래스에 load() 및 save() 메서드가 포함됩니다. load() 및 save() 메서드에서는 로컬 파일을 직접 액세스하여 저장할 수도 있습니다. 이러한 메서드의 용도는 URLLoader 및 Loader 클래스의 같은 이름을 갖는 메서드와 비슷합니다.

참고: FileReference 클래스를 확장하는 File 클래스와 FileStream 클래스는 파일 및 로컬 파일 시스템을 사용하는 작업을 지원하는 함수를 추가적으로 제공합니다. File 클래스와 FileStream 클래스는 AIR에서만 지원되며 Flash Player에서는 지원되지 않습니다.

FileReference 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

각 FileReference 객체는 로컬 컴퓨터의 단일 데이터 파일을 나타냅니다. FileReference 클래스의 속성에는 파일의 크기, 유형, 이름, 파일 이름 확장명, 작성자, 만든 날짜 및 수정 날짜에 대한 정보가 들어 있습니다.

참고: creator 속성은 Mac OS에서만 지원됩니다. 다른 모든 플랫폼에서는 null을 반환합니다.

참고: extension 속성은 Adobe AIR에서만 지원됩니다.

FileReference 클래스의 인스턴스는 다음 두 가지 중 한 가지 방법으로 만들 수 있습니다.

- 다음 코드와 같이 new 연산자를 사용합니다.

```
import flash.net.FileReference;
var fileRef:FileReference = new FileReference();
```

- FileReferenceList.browse() 메서드를 호출합니다. 이 경우 대화 상자가 열리고 업로드할 파일을 하나 이상 선택하라는 메시지가 표시됩니다. 그런 다음 사용자가 하나 이상의 파일을 선택하면 FileReference 객체의 배열이 만들어집니다.

FileReference 객체를 만든 후에는 다음을 수행할 수 있습니다.

- FileReference.browse() 메서드를 호출합니다. 그러면 대화 상자가 열리고 로컬 파일 시스템에서 단일 파일을 선택하라는 메시지가 표시됩니다. 이러한 작업은 보통 FileReference.upload() 메서드 또는 FileReference.load() 메서드에 대한 후속 호출이 이뤄지기 전에 수행됩니다. 파일을 원격 서버에 업로드하려면 FileReference.upload() 메서드를 호출합니다. 로컬 파일을 열려면 FileReference.load() 메서드를 호출합니다.
- FileReference.download() 메서드를 호출합니다. download() 메서드를 호출하면 새 파일을 저장할 위치를 선택할 수 있는 대화 상자가 열립니다. 그런 다음 서버에서 데이터를 다운로드하여 새 파일에 저장합니다.
- FileReference.load() 메서드를 호출합니다. 그러면 browse() 메서드를 사용하여 이전에 선택한 파일에서 데이터가 로드되기 시작합니다. load() 메서드는 browse() 작업이 완료될 때까지(사용자가 파일 선택) 호출할 수 없습니다.
- FileReference.save() 메서드를 호출합니다. 그러면 대화 상자가 열리고 로컬 파일 시스템에서 단일 파일 위치를 선택하라는 메시지가 표시됩니다. 그런 다음 데이터가 지정한 위치에 저장됩니다.

참고: 어디에서나 하나의 대화 상자만 열 수 있으므로 한 번에 하나의 browse(), download() 또는 save() 액션만 실행할 수 있습니다.

name, size 또는 modificationDate와 같은 FileReference 객체 속성은 다음 중 하나가 수행되기 전까지는 정의되지 않습니다.

- FileReference.browse() 메서드나 FileReferenceList.browse() 메서드가 호출되었고 사용자가 대화 상자를 사용하여 파일을 선택했습니다.
- FileReference.download() 메서드가 호출되었고 사용자가 대화 상자를 사용하여 새 파일 위치를 지정했습니다.

참고: 다운로드를 수행하는 경우 다운로드가 완료되기 전에는 FileReference.name 속성만 채워집니다. 파일이 다운로드되면 모든 속성을 사용할 수 있습니다.

FileReference.browse(), FileReferenceList.browse(), FileReference.download(), FileReference.load() 또는 FileReference.save() 메서드에 대한 호출을 실행할 때 대부분의 플레이어에서는 이벤트 전달 및 코드 실행을 포함하여 계속 SWF 파일을 재생합니다.

SWF 파일은 정책 파일에 지정된 도메인을 포함하여 자체 도메인에 있는 파일에만 액세스하여 업로드 및 다운로드 작업을 수행할 수 있습니다. 업로드 또는 다운로드를 시작하는 SWF 파일이 서버와 같은 도메인에 속하지 않는 경우 파일 서버에 정책 파일을 배치해야 합니다.

자세한 내용은 FileReference를 참조하십시오.

파일에서 데이터 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

FileReference.load() 메서드를 사용하면 로컬 파일의 데이터를 메모리에 로드할 수 있습니다.

참고: 코드에서 먼저 FileReference.browse() 메서드를 호출하여 사용자가 로드할 파일을 선택할 수 있도록 해야 합니다. 이 제한은 응용 프로그램 보안 샌드박스에 있는 Adobe AIR에서 실행되는 내용에는 적용되지 않습니다.

FileReference.load() 메서드는 호출 후 즉시 반환되지만 로드되는 데이터는 즉시 사용할 수 없습니다. FileReference 객체는 이벤트를 전달하여 로드 프로세스의 각 단계에서 리스너 메서드를 호출합니다.

FileReference 객체는 로드 프로세스 중에 다음 이벤트를 전달합니다.

- open 이벤트(Event.OPEN): 로드 작업이 시작될 때 전달됩니다.
- progress 이벤트(ProgressEvent.PROGRESS): 데이터 바이트를 파일에서 읽을 때 주기적으로 전달됩니다.
- complete 이벤트(Event.COMPLETE): 로드 작업이 성공적으로 완료되었을 때 전달됩니다.
- ioError 이벤트(IOErrorEvent.IO_ERROR): 파일에서 데이터를 열거나 읽는 동안 발생한 입/출력 오류로 인해 로드 프로세스에 실패할 때 전달됩니다.

FileReference 객체가 complete 이벤트를 전달하면 FileReference 객체의 data 속성에서 ByteArray로 로드된 데이터에 액세스할 수 있습니다.

다음 예제에서는 파일을 선택하라는 메시지를 표시한 다음 해당 파일의 데이터를 메모리에 로드하는 방법을 보여 줍니다.

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

예제 코드에서는 먼저 fileRef라는 FileReference 객체를 만들고 browse() 메서드를 호출합니다. browse() 메서드를 호출하면 파일을 선택할 수 있는 대화 상자가 열립니다. 파일이 선택되면 코드가 onFileSelected() 메서드를 호출합니다. 이 메서드는 progress 및 complete 이벤트를 위해 리스너를 추가하고 FileReference 객체의 load() 메서드를 호출합니다. 이 예제의 다른 핸들러 메서드는 단지 로드 작업의 진행률을 보고하는 메시지를 출력합니다. 로드가 완료되면 trace() 메서드를 사용하여 로드된 파일의 내용이 표시됩니다.

Adobe AIR에서는 FileStream 클래스가 로컬 파일에서 데이터를 읽는 기능을 추가적으로 제공합니다. 자세한 내용은 629페이지의 “파일 읽기 및 쓰기”를 참조하십시오.

로컬 파일에 데이터 저장

Flash Player 9 이상, Adobe AIR 1.0 이상

FileReference.save() 메서드를 사용하여 데이터를 로컬 파일에 저장할 수 있습니다. 이 작업을 시작하면 먼저 사용자가 새 파일 이름 및 파일 저장 위치를 입력할 수 있는 대화 상자가 열립니다. 사용자가 파일 이름 및 위치를 선택하면 데이터가 새 파일에 쓰여집니다. 파일이 성공적으로 저장되면 FileReference 객체의 속성이 로컬 파일의 속성으로 채워집니다.

참고: 코드에서는 마우스 클릭 또는 키 누르기 이벤트와 같이 사용자가 실행한 이벤트에 대한 응답으로 FileReference.save() 메서드만 호출해야 합니다. 그렇지 않으면 오류가 발생합니다. 이 제한은 응용 프로그램 보안 샌드박스에 있는 Adobe AIR에서 실행되는 내용에는 적용되지 않습니다.

FileReference.save() 메서드는 호출 후 즉시 반환됩니다. 그런 다음 FileReference 객체는 이벤트를 전달하여 파일 저장 프로세스의 각 단계에서 리스너 메서드를 호출합니다.

FileReference 객체는 파일 저장 프로세스 중에 다음 이벤트를 전달합니다.

- select 이벤트(Event.SELECT): 사용자가 저장할 새 파일의 위치 및 파일 이름을 지정할 때 전달됩니다.
- cancel 이벤트(Event.CANCEL): 사용자가 대화 상자의 [취소] 버튼을 클릭할 때 전달됩니다.
- open 이벤트(Event.OPEN): 저장 작업이 시작될 때 전달됩니다.
- progress 이벤트(ProgressEvent.PROGRESS): 데이터 바이트를 파일에 저장할 때 주기적으로 전달됩니다.
- complete 이벤트(Event.COMPLETE): 저장 작업이 성공적으로 완료되었을 때 전달됩니다.
- ioError 이벤트(IOErrorEvent.IO_ERROR): 파일에 데이터를 저장하는 동안 발생한 입/출력 오류로 인해 저장 프로세스에 실패할 때 전달됩니다.

FileReference.save() 메서드의 data 매개 변수에 전달되는 객체 유형에 따라 데이터가 파일에 쓰여지는 방식이 결정됩니다.

- 객체 유형이 문자열 값인 경우 데이터는 UTF-8 인코딩을 사용하여 텍스트 파일로 저장됩니다.
- 객체 유형이 XML 객체인 경우 데이터는 파일에 XML 형식으로 쓰여지며 이때 모든 서식이 유지됩니다.
- 객체 유형이 ByteArray 객체인 경우 데이터 내용은 변환 없이 파일에 직접 쓰여집니다.
- 객체 유형이 다른 객체인 경우 FileReference.save() 메서드는 객체의 toString() 메서드를 호출한 다음 결과 문자열 값을 UTF-8 텍스트 파일에 저장합니다. 객체의 toString() 메서드를 호출할 수 없는 경우에는 오류가 발생합니다.

data 매개 변수의 값이 null인 경우에는 오류가 발생합니다.

다음 코드에서는 FileReference.load() 메서드에 대한 이전 예제를 확장합니다. 이 예제에서는 파일의 데이터를 읽은 후 파일 이름을 지정하라는 메시지를 표시하고 데이터를 새 파일에 저장합니다.

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

파일에서 모든 데이터가 로드되면 코드가 onComplete() 메서드를 호출합니다. onComplete() 메서드는 이벤트를 로드하기 위한 리스너를 제거한 다음 saveFile() 메서드를 호출합니다. saveFile() 메서드는 FileReference.save() 메서드를 호출합니다. FileReference.save() 메서드는 사용자가 새 파일 이름 및 파일 저장 위치를 입력할 수 있도록 새 대화 상자를 엽니다. 나머지 이벤트 리스너 메서드는 파일 저장 프로세스가 완료될 때까지 프로세스 진행률을 추적합니다.

Adobe AIR에서는 FileStream 클래스가 로컬 파일에 데이터를 쓰는 기능을 추가적으로 제공합니다. 자세한 내용은 629페이지의 “파일 읽기 및 쓰기”를 참조하십시오.

서버에 파일 업로드

Flash Player 9 이상, Adobe AIR 1.0 이상

서버에 파일을 업로드하려면 먼저 사용자가 하나 이상의 파일을 선택할 수 있도록 browse() 메서드를 호출하십시오. 그 다음에 FileReference.upload() 메서드를 호출하면 선택한 파일이 서버에 전송됩니다. 사용자가 FileReferenceList.browse() 메서드를 사용하여 여러 파일을 선택한 경우 Flash Player에서 FileReferenceList.fileList라는 선택한 파일의 배열을 만듭니다. 그러면 FileReference.upload() 메서드를 사용하여 각 파일을 업로드할 수 있습니다.

참고: FileReference.browse() 메서드를 사용하면 하나의 파일만 업로드할 수 있습니다. 사용자가 여러 파일을 업로드할 수 있도록 하려면 FileReferenceList.browse() 메서드를 사용합니다.

기본적으로 해당 시스템의 파일 선택 대화 상자에서 개발자는 FileFilter 클래스를 사용하고 파일 필터 인스턴스의 배열을 browse() 메서드에 전달하여 하나 이상의 사용자 정의 파일 형식 필터를 지정할 수 있지만 사용자는 로컬 컴퓨터에서 모든 파일 형식을 선택할 수 있습니다.

```
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);
```

사용자가 파일을 선택하고 해당 시스템의 파일 선택 대화 상자에서 [열기] 버튼을 클릭하면 Event.SELECT 이벤트가 전달됩니다. FileReference.browse() 메서드를 사용하여 업로드할 파일을 선택한 경우 다음 코드에서 파일을 웹 서버에 전송합니다.

```
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");
}
```

POST 또는 GET 메서드를 사용하는 변수를 전송하려면 URLRequest.method 및 URLRequest.data 속성을 사용하여 FileReference.upload() 메서드로 서버에 데이터를 전송할 수 있습니다.

FileReference.upload() 메서드를 사용하여 파일을 업로드하면 다음 이벤트가 전달될 수 있습니다.

- open 이벤트(Event.OPEN): 업로드 작업이 시작될 때 전달됩니다.
- progress 이벤트(ProgressEvent.PROGRESS): 파일에서 데이터 바이트가 업로드될 때 주기적으로 전달됩니다.
- complete 이벤트(Event.COMPLETE): 업로드 작업이 성공적으로 완료되었을 때 전달됩니다.
- httpStatus 이벤트(HTTPStatusEvent.HTTP_STATUS): HTTP 오류로 인해 업로드 프로세스에 실패할 때 전달됩니다.
- httpResponseStatus 이벤트(HTTPStatusEvent.HTTP_RESPONSE_STATUS): upload() 또는 uploadUnencoded() 메서드 호출로 HTTP를 통해 데이터 액세스를 시도하고 Adobe AIR에서 요청에 대한 상태 코드를 감지 및 반환할 수 있을 때 전달됩니다.
- securityError 이벤트(SecurityErrorEvent.SECURITY_ERROR): 보안 오류로 인해 업로드 작업에 실패할 때 전달됩니다.

- uploadCompleteData 이벤트(DataEvent.UPLOAD_COMPLETE_DATA): 업로드에 성공한 후 서버에서 데이터가 수신되면 전달됩니다.
- ioError 이벤트(IOErrorEvent.IO_ERROR): 다음과 같은 이유로 업로드 프로세스에 실패한 경우에 전달됩니다.
  - Flash Player에서 파일을 읽고, 쓰고 또는 전송하는 중에 입력/출력 오류가 발생했습니다.
  - SWF가 인증(예: 사용자 이름 및 암호)이 요구되는 서버에 파일을 업로드하려고 시도했습니다. 업로드 시 Flash Player는 암호 입력 방법을 제공하지 않습니다.
  - url 매개 변수에 잘못된 프로토콜이 포함되어 있습니다. FileReference.upload() 메서드는 HTTP 또는 HTTPS를 사용해야 합니다.

Flash Player에서는 인증이 필요한 서버에 대해 완전한 지원을 제공하지 않습니다. 브라우저에서 실행 중인 즉, 브라우저 플러그인이나 Microsoft ActiveX® 컨트롤을 사용하는 SWF 파일만이 인증 및 다운로드에 필요한 사용자 이름과 암호 입력을 요청하는 대화 상자를 제공할 수 있습니다. 플러그인이나 ActiveX 컨트롤을 사용하는 업로드 또는 독립형이나 외부 플레이어를 사용하는 업로드/다운로드의 경우에는 파일이 전송되지 않습니다.

Flash Player에서 파일 업로드를 허용할 수 있도록 ColdFusion에서 서버 스크립트를 만드는 경우 다음과 비슷한 코드를 사용할 수 있습니다.

```
<cffile action="upload" filefield="Filedata" destination="#ExpandPath('./')#" nameconflict="OVERWRITE" />
```

이 ColdFusion 코드는 Flash Player에서 보낸 파일을 업로드하고 ColdFusion 템플릿과 같은 디렉토리에 저장하여 이름이 같은 파일을 덮어씁니다. 앞의 코드에서는 파일 업로드를 허용하는 데 필요한 최소한의 코드를 보여 줍니다. 실제 업무 환경에서는 이 스크립트를 사용하면 안 됩니다. 데이터 유효성 검사를 추가하여 사용자가 잠재적 위험성이 있는 서버측 스크립트 대신 이미지와 같은 허용된 파일 형식만 업로드하도록 합니다.

다음 코드에서는 PHP를 사용한 파일 업로드를 보여 줍니다. 여기에는 데이터 유효성 검사가 포함되어 있습니다. 이 스크립트는 업로드 디렉토리에 업로드된 파일 수를 10개로 제한하고, 200KB 미만의 파일만 업로드하도록 하며, JPEG, GIF 또는 PNG 파일만 파일 시스템에 업로드하고 저장하도록 허용합니다.

```
<?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;
    }
}
?>
```

POST 또는 GET 요청 메서드를 사용하여 스크립트를 업로드할 수 있도록 추가 변수를 전달할 수 있습니다. 추가 POST 변수를 업로드 스크립트에 전송하려면 다음 코드를 사용할 수 있습니다.

```
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");
}
```

앞의 예제에서는 원격 서버측 스크립트에 전달하는 URLVariables 객체를 만듭니다. 이전 버전의 ActionScript에서는 쿼리 문자열의 값을 전달하여 서버 업로드 스크립트에 변수를 전달할 수 있습니다. ActionScript 3.0에서는 POST 또는 GET 메서드를 사용하는 데이터를 전달할 수 있는 URLRequest 객체를 사용하여 원격 스크립트에 변수를 전달할 수 있습니다. 그러면 더 큰 데이터 집합을 더 쉽고 간편하게 전달할 수 있습니다. GET 또는 POST 요청 메서드를 사용하여 변수를 전달할지 여부를 지정하려면 URLRequest.method 속성을 각각 URLRequestMethod.GET 또는 URLRequestMethod.POST로 설정할 수 있습니다.

또한 ActionScript 3.0에서는 앞의 예(기본값 Filedata를 Custom1로 대체)와 같이 두 번째 매개 변수를 upload() 메서드로 제공하여 기본 Filedata 업로드 파일 필드 이름을 덮어 쓸 수 있습니다.

true 값을 세 번째 매개 변수로 upload() 메서드에 전달하여 이 기본값을 재정의할 수 있는 경우에도 Flash Player에서는 기본적으로 테스트 업로드를 전송하지 않습니다. 테스트 업로드의 목적은 실제로 파일이 성공적으로 업로드될지 및 서버 인증이 필요한 경우 인증이 될지 여부를 확인하는 것입니다.

참고: 현재 테스트 업로드는 Windows 기반 Flash Player에서만 가능합니다.

파일 업로드를 처리하는 서버 스크립트에는 다음 요소가 포함된 HTTP POST 요청을 사용해야 합니다.

- multipart/form-data 값이 있는 Content-Type

- name 특성이 "Filedata"로 설정되고 filename 특성이 원본 파일의 이름으로 설정된 Content-Disposition. FileReference.upload() 메서드에서 uploadDataFieldName 매개 변수의 값을 전달하여 사용자 정의 name 속성을 지정할 수 있습니다.
- 파일의 이진 내용

다음은 샘플 HTTP POST 요청입니다.

```
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,,,)
```

다음 샘플 HTTP POST 요청은 api_sig, api_key 및 auth_token이라는 세 가지 POST 변수를 전송하고 "photo"의 사용자 정의 업로드 데이터 필드 이름 값을 사용합니다.

```
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--
```

서버에서 파일 다운로드

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 request 및 defaultFileName라는 두 가지 매개 변수가 필요한 FileReference.download() 메서드를 사용하여 서버에서 파일을 다운로드하도록 할 수 있습니다. 첫 번째 매개 변수는 다운로드할 파일의 URL이 포함된 URLRequest 객체입니다. 두 번째 매개 변수는 선택 항목으로, [파일 다운로드] 대화 상자에 표시되는 기본 파일 이름을 지정할 수 있습니다. 두 번째 매개 변수인 defaultFileName을 생략하면 지정된 URL에서 파일 이름이 사용됩니다.

다음 코드는 같은 디렉토리에서 index.xml이라는 파일을 SWF 파일로 다운로드합니다.

```
var request:URLRequest = new URLRequest("index.xml");
var fileRef:FileReference = new FileReference();
fileRef.download(request);
```

기본 이름을 index.xml 대신 currentnews.xml로 설정하려면 defaultFileName 매개 변수를 다음 코드 예제와 같이 지정하십시오.

```
var request:URLRequest = new URLRequest("index.xml");
var fileToDownload:FileReference = new FileReference();
fileToDownload.download(request, "currentnews.xml");
```

서버 파일 이름이 직관적이지 않거나 서버에서 생성된 경우 파일 이름을 바꾸면 유용할 수 있습니다. 파일을 직접 다운로드하지 않고 서버측 스크립트를 사용하여 파일을 다운로드할 때 defaultFileName 매개 변수를 명시적으로 지정하는 것도 좋습니다. 예를 들어, 전달되는 URL 변수에 따라 특정 파일을 다운로드하는 서버측 스크립트가 있다면 defaultFileName 매개 변수를 지정해야 합니다. 그렇지 않으면 다운로드된 파일의 기본 이름이 서버측 스크립트 이름이 됩니다.

서버 스크립트에서 파싱할 매개 변수를 URL에 첨부하면 download() 메서드를 사용하여 서버에 데이터를 보낼 수 있습니다. 다음 ActionScript 3.0 코드 예제는 ColdFusion 스크립트에 전달되는 매개 변수에 따라 문서를 다운로드합니다.

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

다음 코드는 URL 변수 값에 따라 서버에서 두 파일 중 하나를 다운로드하는 ColdFusion 스크립트인 download.cfm을 보여 줍니다.

```
<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 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

FileReferenceList 클래스를 사용하면 사용자가 서버측 스크립트로 업로드할 하나 이상의 파일을 선택할 수 있습니다. 파일 업로드는 FileReference.upload() 메서드로 처리되며 이 메서드는 선택한 파일 각각에 대해 별도로 호출해야 합니다.

다음 코드는 두 가지 FileFilter 객체(imageFilter 및 textFilter)를 만들고 배열에서 FileReferenceList.browse() 메서드로 전달합니다. 그러면 [운영 체제 파일] 대화 상자에 파일 형식에 대한 두 가지 가능한 필터가 표시됩니다.

```
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.");
}
```

FileReferenceList를 사용하면 둘 이상의 파일을 선택할 수 있지만 사용자가 FileReferenceList 클래스를 사용하여 하나 이상의 파일을 선택하고 업로드할 수 있도록 하는 것은 FileReference.browse()를 사용하여 파일을 선택하는 것과 같습니다. 여러 개의 파일을 업로드하려면 FileReference.upload()를 사용하여 선택한 파일을 각각 업로드해야 합니다.

```
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");
}
```

Event.COMPLETE 이벤트는 배열의 각 FileReference 객체에 추가되므로 각 파일의 업로드가 끝나면 Flash Player에서 completeHandler() 메서드를 호출합니다.

AIR 파일 시스템 API 사용

Adobe AIR 1.0 이상

Adobe AIR 파일 시스템 API에는 다음 클래스가 포함되어 있습니다.

- 파일
- FileMode
- FileStream

파일 시스템 API를 사용하여 다음을 비롯한 여러 가지 작업을 수행할 수 있습니다.

- 파일 및 디렉토리 복사, 만들기, 삭제, 이동
- 파일 및 디렉토리에 대한 정보 가져오기
- 파일 읽기 및 쓰기

AIR 파일 기본 사항

Adobe AIR 1.0 이상

AIR에서 파일 시스템을 사용하는 방법에 대한 빠른 설명 및 코드 예제를 보려면 Adobe Developer Connection에서 다음과 같은 퀵 스타트 문서를 참조하십시오.

- 텍스트 파일 편집기 만들기(Flash)
- 텍스트 파일 편집기 만들기(Flex)
- 디렉토리 검색 응용 프로그램 만들기(Flex)
- XML 환경 설정 파일 읽기 및 쓰기(Flex)
- 파일 및 데이터 압축(Flex)

Adobe AIR에서는 파일과 폴더를 액세스하고 만들고 관리하는 데 사용할 수 있는 클래스를 제공합니다. flash.filesystem 패키지에 포함된 이러한 클래스는 다음과 같이 사용됩니다.

File 클래스	설명
File	File 객체는 파일 또는 디렉토리 경로를 나타냅니다. File 객체를 사용하면 파일 또는 폴더에 대한 포인터를 만들어 파일 또는 폴더와의 상호 작용을 시작할 수 있습니다.
FileMode	FileMode 클래스는 FileStream 클래스에 대한 open() 및 openAsync() 메서드의 fileMode 매개 변수에 사용되는 문자열 상수를 정의합니다. 이 메서드의 fileMode 매개 변수는 파일을 연 후에 FileStream 객체에서 사용할 수 있는 기능(즉, 쓰기, 읽기, 추가 및 업데이트)을 결정합니다.
FileStream	FileStream 객체는 파일을 읽고 쓰기 위해 여는 데 사용합니다. 새 파일이나 기존 파일을 가리키는 File 객체를 만든 후 이 포인터를 FileStream 객체로 전달하면 이를 열고 데이터를 읽거나 쓸 수 있습니다.

File 클래스의 일부 메서드는 다음과 같이 동기 및 비동기 버전을 모두 포함하고 있습니다.

- File.copyTo() 및 File.copyToAsync()
- File.deleteDirectory() 및 File.deleteDirectoryAsync()
- File.deleteFile() 및 File.deleteFileAsync()
- File.getDirectoryListing() 및 File.getDirectoryListingAsync()
- File.moveTo() 및 File.moveToAsync()
- File.moveToTrash() 및 File.moveToTrashAsync()

또한 FileStream 객체가 파일을 여는 방법 즉, open() 메서드를 호출하는지 또는 openAsync() 메서드를 호출하는지에 따라 FileStream 작업이 동기적 또는 비동기적으로 작동합니다.

비동기 버전을 사용하면 백그라운드에서 실행되고 완료되거나 오류 이벤트가 발생하면 이벤트를 전달하는 프로세스를 시작할 수 있습니다. 이러한 비동기 백그라운드 프로세스가 진행되는 동안 다른 코드가 실행될 수 있습니다. 비동기 버전의 작업에서는 함수를 호출하는 File 또는 FileStream 객체의 addEventListener() 메서드를 사용하여 이벤트 리스너 함수를 설정해야 합니다.

동기 버전을 사용하면 이벤트 리스너 설정이 필요하지 않은 단순한 코드를 작성할 수 있습니다. 그러나 동기 메서드가 실행되고 있는 동안에는 다른 코드를 실행할 수 없으므로 표시 객체 렌더링 및 애니메이션과 같은 중요한 프로세스가 지연될 수 있습니다.

AIR의 File 객체 작업

Adobe AIR 1.0 이상

File 객체는 파일 시스템에 있는 파일 또는 디렉토리에 대한 포인터입니다.

File 클래스는 FileReference 클래스를 확장합니다. AIR은 물론 Adobe® Flash® Player에서도 사용할 수 있는 FileReference 클래스는 파일에 대한 포인터를 나타냅니다. File 클래스는 보안 문제로 인해 Flash Player(브라우저에서 실행 중인 SWF)에서 노출되지 않는 속성 및 메서드를 추가합니다.

File 클래스

Adobe AIR 1.0 이상

다음과 같은 작업에 File 클래스를 사용할 수 있습니다.

- 사용자 디렉토리, 사용자의 문서 디렉토리, 응용 프로그램이 시작된 디렉토리 및 응용 프로그램 디렉토리와 같은 특수 디렉토리에 대한 경로 가져오기
- 파일 및 디렉토리 복사
- 파일 및 디렉토리 이동
- 파일 및 디렉토리 삭제(또는 휴지통으로 이동)
- 디렉토리에 포함된 파일 및 디렉토리 나열
- 임시 파일 및 폴더 만들기

File 객체가 파일 경로를 가리키면 FileStream 클래스를 사용하여 파일 데이터를 읽고 쓸 수 있습니다.

File 객체가 아직 없는 파일이나 디렉토리의 경로를 가리킬 수 있습니다. 파일이나 디렉토리를 만들 때 이러한 File 객체를 사용할 수 있습니다.

File 객체의 경로

Adobe AIR 1.0 이상

각 File 객체에는 각각 해당 경로를 지정하는 두 개의 속성이 있습니다.

속성	설명
nativePath	파일에 대한 플랫폼별 경로를 지정합니다. 예를 들어 Windows의 경우 경로는 "c:\Sample directory\test.txt"일 수 있는 반면 Mac OS에서는 "/Sample directory/test.txt"일 수 있습니다. nativePath 속성은 디렉토리 분리 기호 문자로 Windows에서는 백슬래시(\) 문자를 사용하고 Mac OS 및 Linux에서는 슬래시(/) 문자를 사용합니다.
url	이 속성은 파일 URL 스킴을 사용하여 파일을 가리킬 수 있습니다. 예를 들어 Windows의 경우 경로는 "file:///c:/Sample%20directory/test.txt"일 수 있는 반면 Mac OS에서는 "file:///Sample%20directory/test.txt"일 수 있습니다. 런타임에는 file 외에도 다른 특수 URL 스킴이 포함되어 있으며 이에 대해서는 617페이지의 "지원되는 AIR URL 스킴"에서 설명합니다.

File 클래스에는 Mac OS, Windows 및 Linux의 표준 디렉토리를 가리키기 위한 다음과 같은 정적 속성이 포함되어 있습니다.

- File.applicationStorageDirectory - 설치된 각 AIR 응용 프로그램에 고유한 저장소 디렉토리. 이 디렉토리는 동적 응용 프로그램 에셋 및 사용자 환경 설정을 저장하는 데 적합한 장소입니다. 대용량의 데이터는 다른 장소에 저장하는 것이 좋습니다.

  Android 및 iOS에서 응용 프로그램 저장소 디렉토리는 해당 응용 프로그램이 제거되거나 사용자가 응용 프로그램 데이터를 지우는 경우 제거되지만 다른 플랫폼에서는 그렇지 않습니다.

- File.applicationDirectory - 응용 프로그램이 설치된 디렉토리(설치되어 있는 모든 에셋 포함). 일부 운영 체제에서 응용 프로그램은 실제 디렉토리가 아닌 단일 패키지 파일에 저장됩니다. 이 경우 해당 내용은 기본 경로를 통해 액세스할 수 없으며 응용 프로그램 디렉토리는 읽기 전용이 됩니다.
- File.desktopDirectory - 사용자 데스크톱 디렉토리. 플랫폼에서 데스크톱 디렉토리를 정의하지 않는 경우 파일 시스템의 다른 위치가 사용됩니다.
- File.documentsDirectory - 사용자 문서 디렉토리. 플랫폼에서 문서 디렉토리를 정의하지 않는 경우 파일 시스템의 다른 위치가 사용됩니다.
- File.userDirectory - 사용자 디렉토리. 플랫폼에서 사용자 디렉토리를 정의하지 않는 경우 파일 시스템의 다른 위치가 사용됩니다.

참고: 플랫폼에서 데스크톱, 문서 또는 사용자 디렉토리에 대한 표준 위치를 정의하지 않는 경우 File.documentsDirectory, File.desktopDirectory 및 File.userDirectory는 서로 같은 디렉토리를 참조할 수 있습니다.

이러한 속성들은 운영 체제에 따라 다양한 값을 가집니다. 예를 들어 Mac 및 Windows는 사용자 데스크톱 디렉토리의 기본 경로가 각각 다릅니다. 그러나 File.desktopDirectory 속성은 모든 플랫폼에서 적합한 디렉토리 경로를 가리킵니다. 여러 플랫폼에 걸쳐 작동하는 응용 프로그램에 쓰기를 수행하려면 이러한 속성에 기반하여 해당 응용 프로그램에서 사용하는 다른 디렉토리와 파일을 참조할 수 있습니다. 그런 다음 resolvePath() 메서드를 사용하여 경로를 세부적으로 지정하십시오. 예를 들어 이 코드는 아래의 응용 프로그램 저장소 디렉토리에서 preferences.xml 파일을 가리킵니다.

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("preferences.xml");
```

File 클래스를 통해 특정 파일 경로를 가리킬 수 있지만 그렇게 하면 응용 프로그램을 서로 다른 플랫폼에 걸쳐 사용하지 못할 수 있습니다. 예를 들어 C:\Documents and Settings\joe\와 같은 경로는 Windows 운영 체제에서만 사용할 수 있습니다. 이러한 경우를 피하기 위해 File.documentsDirectory 같은 File 클래스의 정적 속성을 사용하는 것이 좋습니다.

공통 디렉토리 위치

플랫폼	디렉토리 유형	일반적인 파일 시스템 위치
Android	응용 프로그램	/data/data/
	응용 프로그램 저장소	/data/data/air.applicationID/filename/Local Store
	캐시	/data/data/applicationID/cache
	데스크톱	/mnt/sdcard
	문서	/mnt/sdcard
	임시	/data/data/applicationID/cache/FlashTmp.randomString
	사용자	/mnt/sdcard
iOS	응용 프로그램	/var/mobile/Applications/uid/filename.app
	응용 프로그램 저장소	/var/mobile/Applications/uid/Library/Application Support/applicationID/Local Store
	캐시	/var/mobile/Applications/uid/Library/Caches
	데스크톱	액세스할 수 없음
	문서	/var/mobile/Applications/uid/Documents
	임시	/private/var/mobile/Applications/uid/tmp/FlashTmpNNN
	사용자	액세스할 수 없음
Linux	응용 프로그램	/opt/filename/share
	응용 프로그램 저장소	/home/userName/.appdata/applicationID/Local Store
	데스크톱	/home/userName/Desktop
	문서	/home/userName/Documents
	임시	/tmp/FlashTmp.randomString
	사용자	/home/userName
Mac	응용 프로그램	/Applications/filename.app/Contents/Resources
	응용 프로그램 저장소	/Users/*userName*/Library/Preferences/*applicationid*/Local Store(AIR 3.2 이하) *path*/Library/Application Support/*applicationid*/Local Store(AIR 3.3 이상). 여기서 'path'는 /Users/*userName*/Library/Containers/*bundle-id*/Data(샌드박스 환경) 또는 /Users/userName(샌드박스 환경 외부에서 실행되는 경우)
	캐시	/Users/userName/Library/Caches
	데스크톱	/Users/*userName*/Desktop
	문서	/Users/*userName*/Documents
	임시	/private/var/folders/JY/randomString/TemporaryItems/FlashTmp
	사용자	/Users/*userName*

플랫폼	디렉토리 유형	일반적인 파일 시스템 위치
Windows	응용 프로그램	C:\Program Files\filename
	응용 프로그램 저장소	C:\Documents and settings\userName\ApplicationData\applicationID\Local Store
	캐시	C:\Documents and settings\userName\Local Settings\Temp
	데스크톱	C:\Documents and settings\userName\Desktop
	문서	C:\Documents and settings\userName\My Documents
	임시	C:\Documents and settings\userName\Local Settings\Temp\randomString.tmp
	사용자	C:\Documents and settings\userName

이러한 디렉토리의 실제 기본 경로는 운영 체제 및 컴퓨터 구성에 따라 달라집니다. 이 표에 나온 경로는 일반적인 예일 뿐입니다. 응용 프로그램이 모든 플랫폼에서 제대로 작동하려면 항상 적합한 정적 File 클래스 속성을 사용하여 이러한 디렉토리를 참조해야 합니다. 실제 AIR 응용 프로그램에서 위 표에 나온 applicationID 및 filename의 값은 응용 프로그램 설명자로부터 가져 옵니다. 응용 프로그램 설명자에서 제작자 ID를 지정하면 해당 제작자 ID는 이와 같은 경로의 응용 프로그램 ID에 추가됩니다. userName의 값은 설치를 실행한 사용자의 계정 이름입니다.

File 객체로 디렉토리 가리키기

Adobe AIR 1.0 이상

디렉토리를 가리키도록 File 객체를 설정하는 여러 방법이 있습니다.

사용자 홈 디렉토리 가리키기

Adobe AIR 1.0 이상

File 객체로 사용자 홈 디렉토리를 가리킬 수 있습니다. 다음 코드에서는 홈 디렉토리의 AIR Test 하위 디렉토리를 가리키도록 File 객체를 설정합니다.

```
var file:File = File.userDirectory.resolvePath("AIR Test");
```

사용자 문서 디렉토리 가리키기

Adobe AIR 1.0 이상

File 객체로 사용자 문서 디렉토리를 가리킬 수 있습니다. 다음 코드에서는 문서 디렉토리의 AIR Test 하위 디렉토리를 가리키도록 File 객체를 설정합니다.

```
var file:File = File.documentsDirectory.resolvePath("AIR Test");
```

바탕 화면 디렉토리 가리키기

Adobe AIR 1.0 이상

File 객체로 바탕 화면을 가리킬 수 있습니다. 다음 코드에서는 바탕 화면의 AIR Test 하위 디렉토리를 가리키도록 File 객체를 설정합니다.

```
var file:File = File.desktopDirectory.resolvePath("AIR Test");
```

응용 프로그램 저장소 디렉토리 가리키기

Adobe AIR 1.0 이상

File 객체로 응용 프로그램 저장소 디렉토리를 가리킬 수 있습니다. 모든 AIR 응용 프로그램에 대해 응용 프로그램 저장소 디렉토리를 정의하는 연결된 고유한 경로가 있습니다. 이 디렉토리는 각 응용 프로그램 및 사용자에 대해 고유합니다. 이 디렉토리에는 사용자별, 응용 프로그램별 데이터(예: 사용자 데이터 또는 환경 설정 파일)를 저장할 수 있습니다. 예를 들어 다음 코드에서는 File 객체로 응용 프로그램 저장소에 포함된 환경 설정 파일 prefs.xml을 가리킵니다.

```
var file:File = File.applicationStorageDirectory;
file = file.resolvePath("prefs.xml");
```

일반적으로 응용 프로그램 저장소 디렉토리 위치는 사용자 이름 및 응용 프로그램 ID에 따라 결정됩니다. 다음 파일 시스템 위치는 응용 프로그램을 디버그하는 데 도움이 됩니다. 이러한 디렉토리의 파일을 확인하려면 항상 File.applicationStorage 속성 또는 app-storage: URI 스킴을 사용해야 합니다.

- Mac OS의 경우 AIR 버전에 따라 다름:

  AIR 3.2 이하: /Users/user name/Library/Preferences/***applicationID***/Local Store/

  AIR 3.3 이상: ***path***/Library/Application Support/***applicationID***/Local Store. 여기서 ***path***는 /Users/***username***/Library/Containers/***bundle-id***/Data(샌드박스 환경) 또는 /Users/***username***(샌드박스 환경 외부에서 실행되는 경우)

  예(AIR 3.2):

  ```
  /Users/babbage/Library/Preferences/com.example.TestApp/Local Store
  ```

- Windows의 경우 Documents and Settings 디렉토리에서:

  C:\Documents and Settings\user name\Application Data**applicationID**\Local Store\

  예를 들면 다음과 같습니다.

  ```
  C:\Documents and Settings\babbage\Application Data\com.example.TestApp\Local Store
  ```

- Linux의 경우:

  /home/**user name**/.appdata/**applicationID**/Local Store/

  예를 들면 다음과 같습니다.

  ```
  /home/babbage/.appdata/com.example.TestApp/Local Store
  ```

- Android의 경우:

  /data/data/androidPackageID/applicationID/Local Store

  예를 들면 다음과 같습니다.

  ```
  /data/data/air.com.example.TestApp/com.example.TestApp/Local Store
  ```

참고: 응용 프로그램에 제작자 ID가 있는 경우 해당 제작자 ID는 응용 프로그램 저장소 디렉토리 경로의 일부로도 사용됩니다.

File.applicationStorageDirectory로 만든 File 객체의 URL(및 url 속성)에서는 다음과 같이 app-storage URL 스킴을 사용합니다. 자세한 내용은 617페이지의 "지원되는 AIR URL 스킴"을 참조하십시오.

```
var dir:File = File.applicationStorageDirectory;
dir = dir.resolvePath("preferences");
trace(dir.url); // app-storage:/preferences
```

응용 프로그램 디렉토리 가리키기

Adobe AIR 1.0 이상

File 객체로 응용 프로그램이 설치된 디렉토리(응용 프로그램 디렉토리라고 함)를 가리킬 수 있습니다. File.applicationDirectory 속성을 사용하여 이 디렉토리를 참조할 수 있습니다. 이 디렉토리를 사용하여 응용 프로그램과 함께 설치되는 응용 프로그램 설명자 파일이나 기타 리소스를 검토할 수 있습니다. 예를 들어 다음 코드에서는 File 객체로 응용 프로그램 디렉토리에 있는 **images** 라는 디렉토리를 가리킵니다.

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
```

File.applicationDirectory로 만든 File 객체의 URL(및 url 속성)에서는 다음과 같이 app URL 스킴을 사용합니다. 자세한 내용은 617페이지의 "지원되는 AIR URL 스킴"을 참조하십시오.

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("images");
trace(dir.url); // app:/images
```

참고: Android에서 응용 프로그램 패키지의 파일은 nativePath를 통해서 액세스할 수 없습니다. nativePath 속성이 빈 문자열입니다. 응용 프로그램 디렉토리에 있는 파일에 액세스하려면 기본 경로가 아니라 항상 URL을 사용해야 합니다.

캐시 디렉토리 가리키기

Adobe AIR 3.6 이상

File.cacheDirectory 속성을 사용하여 File 객체가 운영 체제의 임시 디렉토리 또는 캐시 디렉토리를 가리키도록 할 수 있습니다. 이 디렉토리에는 응용 프로그램 실행에 필요하지 않고 삭제하더라도 문제나 데이터 손실이 발생하지 않는 임시 파일이 들어 있습니다.

대부분의 운영 체제에서는 캐시 디렉토리가 임시 디렉토리입니다. iOS에서는 캐시 디렉토리가 응용 프로그램 라이브러리의 Caches 디렉토리입니다. 이 디렉토리의 파일은 온라인 저장소에 백업되지 않으며 장치의 가용 저장 공간이 부족할 경우 운영 체제에 의해 삭제될 수도 있습니다. 자세한 내용은 618페이지의 "파일 백업 및 캐싱 제어"를 참조하십시오.

파일 시스템 루트 가리키기

Adobe AIR 1.0 이상

File.getRootDirectories() 메서드는 Windows 컴퓨터에서 C: 및 마운트된 볼륨과 같은 모든 루트 볼륨을 나열합니다. Mac OS 및 Linux에서 이 메서드는 항상 시스템의 고유한 루트 디렉토리("/" 디렉토리)를 반환합니다.
StorageVolumeInfo.getStorageVolumes() 메서드는 마운트된 저장소 볼륨에 대해 보다 자세한 정보를 제공합니다(627페이지의 "저장소 볼륨을 사용한 작업" 참조).

참고: Android에서 파일 시스템의 루트는 읽을 수 없습니다. 디렉토리를 참조하는 File 객체와 함께 기본 경로 '/'가 반환되지만 해당 객체의 속성은 정확한 값을 갖지 않습니다. 예를 들어 spaceAvailable은 항상 0입니다.

명시적 디렉토리 가리키기

Adobe AIR 1.0 이상

다음 예제(Windows의 경우)와 같이 File 객체의 nativePath 속성을 설정하여 File 객체로 명시적 디렉토리를 가리킬 수 있습니다.

```
var file:File = new File();
file.nativePath = "C:\\AIR Test";
```

중요: 이 방법으로 명시적 경로를 가리킬 경우 특정 플랫폼에만 사용할 수 있는 코드가 작성될 수 있습니다. 예를 들어 위 예제는 Windows에서만 사용할 수 있습니다. File 객체의 정적 속성(예: File.applicationStorageDirectory)를 이용하면 플랫폼에 관계없이 사용할 수 있는 디렉토리를 찾을 수 있습니다. 그런 다음 resolvePath() 메서드(다음 단원 참조)를 사용하여 상대 경로로 이동합니다.

상대 경로 탐색

Adobe AIR 1.0 이상

resolvePath() 메서드를 사용하여 지정된 다른 경로에 상대적인 경로를 가져올 수 있습니다. 예를 들어 다음 코드에서는 사용자 홈 디렉토리의 "AIR Test" 하위 디렉토리를 가리키도록 File 객체를 설정합니다.

```
var file:File = File.userDirectory;
file = file.resolvePath("AIR Test");
```

또한 File 객체의 url 속성을 사용하여 다음과 같이 URL 문자열을 기반으로 디렉토리를 가리킬 수 있습니다.

```
var urlStr:String = "file:///C:/AIR Test/";
var file:File = new File()
file.url = urlStr;
```

자세한 내용은 617페이지의 “File 경로 수정”을 참조하십시오.

사용자가 디렉토리를 찾아 선택할 수 있도록 설정

Adobe AIR 1.0 이상

File 클래스에는 사용자가 객체에 할당할 디렉토리를 선택할 수 있는 시스템 대화 상자를 표시하는 browseForDirectory() 메서드가 포함되어 있습니다. browseForDirectory() 메서드는 비동기적입니다. File 객체는 사용자가 디렉토리를 선택하고 [열기] 버튼을 클릭하는 경우 select 이벤트를 전달하거나, [취소] 버튼을 클릭하는 경우 cancel 이벤트를 전달합니다.

예를 들어 다음 코드에서는 사용자가 디렉토리를 선택할 수 있게 하고 경로 선택 시 디렉토리 경로를 출력합니다.

```
var file:File = new File();
file.addEventListener(Event.SELECT, dirSelected);
file.browseForDirectory("Select a directory");
function dirSelected(e:Event):void {
    trace(file.nativePath);
}
```

참고: Android에서 browseForDirectory() 메서드는 지원되지 않습니다. 이 메서드를 호출해도 아무런 효과가 없으며 즉시 cancel 이벤트가 전달됩니다. 사용자가 디렉토리를 선택할 수 있도록 하려면 사용자 정의를 통해 응용 프로그램에서 정의되는 대화 상자를 사용합니다.

응용 프로그램이 호출된 디렉토리 가리키기

Adobe AIR 1.0 이상

응용 프로그램이 호출될 때 전달된 InvokeEvent 객체의 currentDirectory 속성을 확인하여 응용 프로그램이 호출된 디렉토리 위치를 가져올 수 있습니다. 자세한 내용은 799페이지의 “명령줄 인수 캡처”를 참조하십시오.

File 객체로 파일 가리키기

Adobe AIR 1.0 이상

File 객체가 가리키는 파일을 설정하는 여러 방법이 있습니다.

명시적 파일 경로 가리키기

Adobe AIR 1.0 이상

중요: 명시적 경로를 가리킬 경우 특정 플랫폼에만 사용할 수 있는 코드가 작성될 수 있습니다. 예를 들어 C:/foo.txt와 같은 경로는 Windows에서만 사용할 수 있습니다. File 객체의 정적 속성(예: File.applicationStorageDirectory)을 이용하면 플랫폼에 관계없이 작동하는 디렉토리를 찾을 수 있습니다. 그런 다음 resolvePath() 메서드(617페이지의 "File 경로 수정" 참조)를 사용하여 상대 경로로 이동합니다.

다음과 같이 File 객체의 url 속성을 사용하여 URL 문자열을 기반으로 파일 또는 디렉토리를 가리킬 수 있습니다.

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File()
file.url = urlStr;
```

또한 다음과 같이 File() 생성자 함수에 URL을 전달할 수도 있습니다.

```
var urlStr:String = "file:///C:/AIR Test/test.txt";
var file:File = new File(urlStr);
```

url 속성은 항상 URI로 인코딩된 버전의 URL(예: 공백을 "%20으로 대체)을 반환합니다.

```
file.url = "file:///c:/AIR Test";
trace(file.url); // file:///c:/AIR%20Test
```

또한 File 객체의 nativePath 속성을 사용하여 명시적 경로를 설정할 수도 있습니다. 예를 들어 다음 코드는 Windows가 설치된 컴퓨터에서 실행할 경우 File 객체를 C: 드라이브의 AIR Test 하위 디렉토리에 있는 test.txt 파일로 설정합니다.

```
var file:File = new File();
file.nativePath = "C:/AIR Test/test.txt";
```

또한 다음과 같이 이 경로를 File() 생성자 함수에 전달할 수도 있습니다.

```
var file:File = new File("C:/AIR Test/test.txt");
```

슬래시(/) 문자를 nativePath 속성의 경로 구분 기호로 사용합니다. Windows의 경우 백슬래시(\) 문자를 사용할 수도 있지만 그렇게 하면 응용 프로그램을 서로 다른 플랫폼에 걸쳐 사용하지 못할 수 있습니다.

자세한 내용은 617페이지의 "File 경로 수정"을 참조하십시오.

디렉토리의 파일 열거

Adobe AIR 1.0 이상

File 객체의 getDirectoryListing() 메서드를 사용하여 디렉토리 루트 레벨의 파일 및 하위 디렉토리를 가리키는 File 객체 배열을 가져올 수 있습니다. 자세한 내용은 623페이지의 "디렉토리 열거"를 참조하십시오.

사용자가 파일을 찾아 선택할 수 있도록 설정

Adobe AIR 1.0 이상

File 클래스에는 사용자가 객체에 할당할 파일을 선택할 수 있는 시스템 대화 상자를 표시하는 다음 메서드가 포함되어 있습니다.

- browseForOpen()
- browseForSave()
- browseForOpenMultiple()

이러한 각각의 메서드는 비동기적입니다. browseForOpen() 및 browseForSave() 메서드는 사용자가 파일(또는 browseForSave()의 경우 대상 경로)을 선택하면 select 이벤트를 전달합니다. browseForOpen() 및 browseForSave() 메서드에서는 대상 File 객체 선택 시 선택한 파일을 가리킵니다. browseForOpenMultiple() 메서드는 사용자가 파일을 선택할 때 selectMultiple 이벤트를 전달합니다. selectMultiple 이벤트는 FileListEvent 유형이며 선택한 파일을 가리키는 File 객체 배열인 files 속성을 가집니다.

예를 들어 다음 코드에서는 사용자가 파일을 선택할 수 있는 [열기] 대화 상자를 표시합니다.

```
 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);
}
```

browse 메서드를 호출할 때 응용 프로그램에 다른 브라우저 대화 상자가 열려 있는 경우 런타임에서 Error 예외가 발생합니다.

참고: Android에서는 browseForOpen() 및 browseForOpenMultiple() 메서드로 이미지, 비디오 및 오디오 파일만 선택할 수 있습니다. 또한 browseForSave() 대화 상자에는 사용자가 임의의 파일 이름을 입력할 수 있지만 미디어 파일만 표시됩니다. 미디어가 아닌 파일을 열고 저장하려면 이러한 메서드 대신 사용자 정의 대화 상자를 사용해야 합니다.

File 경로 수정

Adobe AIR 1.0 이상

또한 다음 예제와 같이 resolvePath() 메서드를 호출하거나 객체의 nativePath 또는 url 속성을 수정하여 기존 File 객체의 경로를 수정할 수도 있습니다(Windows의 경우).

```
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
```

nativePath 속성을 사용할 때는 슬래시(/) 문자를 디렉토리 분리 기호 문자로 사용합니다. Windows의 경우 백슬래시(\) 문자를 사용할 수도 있지만, 특정 플랫폼에서만 사용할 수 있는 코드가 작성될 수 있으므로 권장되지 않습니다.

지원되는 AIR URL 스킴

Adobe AIR 1.0 이상

AIR에서는 다음과 같은 URL 스킴을 사용하여 File 객체의 url 속성을 정의할 수 있습니다.

URL 스킴	설명
file	파일 시스템의 루트에 상대적인 경로를 지정하는 데 사용합니다. 예를 들면 다음과 같습니다. `file:///c:/AIR Test/test.txt` URL 표준에서는 file URL이 file://<호스트>/<경로>의 형태를 가지도록 지정합니다. 특별한 경우 <호스트>가 빈 문자열일 수 있는데, 이 경우 "URL이 해석되고 있는 시스템"으로 해석됩니다. 이 때문에 file URL에는 세 개의 슬래시(///)가 있는 경우가 종종 있습니다.
app	설치된 응용 프로그램의 루트 디렉토리(설치된 응용 프로그램의 application.xml 파일이 포함된 디렉토리)에 대해 상대적인 경로를 지정하는 데 사용합니다. 예를 들어 다음 경로는 설치된 응용 프로그램의 images 하위 디렉토리를 가리킵니다. `app:/images`
app-storage	응용 프로그램 저장소 디렉토리에 상대적인 경로를 지정하는 데 사용합니다. 설치된 각 응용 프로그램에 대해 AIR은 해당 응용 프로그램 관련 데이터를 저장하기에 좋은 장소인 고유한 응용 프로그램 저장소 디렉토리를 정의합니다. 예를 들어 다음 경로는 응용 프로그램 저장소 디렉토리의 settings 하위 디렉토리에 있는 prefs.xml 파일을 가리킵니다. `app-storage:/settings/prefs.xml`

파일 백업 및 캐싱 제어

Adobe AIR 3.6 이상, iOS 및 OS X에만 해당

일부 운영 체제(대표적인 예로 iOS 및 Mac OS X)에는 응용 프로그램 파일을 자동으로 원격 저장소에 백업하는 기능이 있습니다. 또한 iOS에서는 파일 백업 여부 및 용도에 따른 파일 저장 장소에 대해 제약이 있습니다.

다음은 Apple의 파일 백업 및 저장에 관한 지침 준수 방법을 요약한 것입니다. 자세한 내용은 다음 섹션을 참조하십시오.

- 파일을 백업할 필요가 없고 장치 저장 공간이 부족할 경우 운영 체제가 파일을 삭제할 수 있도록 지정하려면(iOS에만 해당) 파일을 캐시 디렉토리(File.cacheDirectory)에 저장하십시오. 이는 iOS의 기본 저장 위치이며 다시 생성하거나 다시 다운로드할 수 있는 대부분의 파일에 대해 이 위치를 사용하는 것이 좋습니다.
- 파일을 백업할 필요는 없지만 운영 체제가 삭제할 수 없도록 지정하려면 파일을 응용 프로그램 저장 디렉토리(File.applicationStorageDirectory) 또는 문서 디렉토리(File.documentsDirectory)와 같은 응용 프로그램 라이브러리 디렉토리 중 하나에 저장하십시오. File 객체의 preventBackup 속성을 true로 설정하십시오. 이는 다시 생성하거나 다시 다운로드할 수 있지만 오프라인에서 사용할 때 응용 프로그램이 올바르게 작동하도록 하는 데 필요한 내용에 대한 Apple의 요구 사항입니다.

파일 백업 지정

백업 공간을 절약하고 네트워크 대역폭 사용을 줄이기 위해 iOS 및 Mac 응용 프로그램에 관한 Apple 지침은 사용자 입력 데이터 또는 다시 생성할 수 없고 다시 다운로드할 수 없는 데이터가 포함된 파일만 백업용으로 지정할 수 있도록 규정하고 있습니다.

기본 설정에 따라 응용 프로그램 라이브러리 폴더의 모든 파일이 백업됩니다. Mac OS X의 경우 이는 응용 프로그램 저장 디렉토리입니다. iOS에서는 여기에 응용 프로그램 저장 디렉토리, 응용 프로그램 디렉토리, 데스크톱 디렉토리, 문서 디렉토리 및 사용자 디렉토리가 포함됩니다(이들 디렉토리가 iOS의 응용 프로그램 라이브러리 폴더에 매핑되기 때문). 따라서 기본 설정에 따라 이들 디렉토리의 모든 파일이 서버 저장소에 백업됩니다.

파일을 응용 프로그램이 다시 생성할 수 있는 위치 중 하나에 저장하는 경우 파일을 운영 체제가 백업하지 않도록 플래그를 지정해야 합니다. 파일을 백업하지 않도록 지정하려면 File 객체의 preventBackup 속성을 true로 설정하십시오.

iOS에서는 응용 프로그램 라이브러리 폴더에 있는 파일의 경우 해당 파일의 preventBackup 속성이 true로 설정된 경우에도 운영 체제가 삭제할 수 없는 영구 파일로 플래그를 설정합니다.

파일 캐싱 및 삭제 제어

iOS 응용 프로그램에 관한 Apple의 지침에 따르면 다시 생성할 수 있는 내용은 장치의 저장 공간이 부족할 경우 가급적 운영 체제가 삭제하도록 하는 것이 좋습니다.

iOS에서는 응용 프로그램 라이브러리 폴더(예: 응용 프로그램 저장 디렉토리 또는 문서 디렉토리)의 파일이 영구 파일로 플래그가 설정되어 운영 체제에 의해 삭제되지 않습니다.

응용 프로그램이 다시 생성할 수 있고, 응용 프로그램 캐시 디렉토리에 저장 공간이 부족할 경우 삭제해도 무방한 파일을 저장하십시오. File.cacheDirectory 정적 속성을 사용하여 캐시 디렉토리에 액세스합니다.

iOS에서 캐시 디렉토리는 응용 프로그램의 캐시 디렉토리(<Application Home>/Library/Caches)입니다. 다른 운영 체제에서는 이 디렉토리가 해당 디렉토리에 매핑됩니다. 예를 들어 Mac OS X에서는 응용 프로그램 라이브러리의 Caches 디렉토리에 매핑됩니다. Android에서는 캐시 디렉토리가 응용 프로그램의 캐시 디렉토리에 매핑됩니다. Windows의 경우 캐시 디렉토리가 운영 체제 임시 디렉토리에 매핑됩니다. Android 및 Windows 모두 File 클래스의 createTempDirectory() 및 createTempFile() 메서드 호출 시 액세스하는 디렉토리가 동일합니다.

두 파일 간의 상대 경로 찾기

Adobe AIR 1.0 이상

getRelativePath() 메서드를 사용하여 두 파일 간의 상대 경로를 찾을 수 있습니다.

```
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
```

getRelativePath() 메서드의 두 번째 매개 변수인 useDotDot 매개 변수는 상위 디렉토리를 나타내는 .. 구문을 결과로 반환할 수 있습니다.

```
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
```

정규 버전의 파일 이름 가져오기

Adobe AIR 1.0 이상

파일 및 경로 이름은 Windows 및 Mac OS에서 대/소문자를 구분하지 않습니다. 다음에서 두 File 객체는 같은 파일을 가리킵니다.

```
File.documentsDirectory.resolvePath("test.txt");
File.documentsDirectory.resolvePath("TeSt.TxT");
```

그러나 문서 및 디렉토리 이름에는 대/소문자가 포함됩니다. 예를 들어 다음 예제에서는 문서 디렉토리에 AIR Test라는 폴더가 있다고 가정합니다.

```
var file:File = File.documentsDirectory.resolvePath("AIR test");
trace(file.nativePath); // ... AIR test
file.canonicalize();
trace(file.nativePath); // ... AIR Test
```

canonicalize() 메서드는 파일 또는 디렉토리 이름에 올바른 대/소문자를 사용하도록 nativePath 객체를 변환합니다. Linux와 같이 대/소문자가 구분되는 파일 시스템에서 이름의 대/소문자만 다른 여러 파일이 있는 경우 canonicalize() 메서드는 파일 시스템에 의해 결정되는 순서로 검색된 첫 번째 파일과 일치하도록 경로를 조정합니다.

또한 다음 예제와 같이 canonicalize() 메서드를 사용하여 Windows에서 짧은 파일 이름("8.3" 이름)을 긴 파일 이름으로 변환할 수 있습니다.

```
var path:File = new File();
path.nativePath = "C:\\AIR~1";
path.canonicalize();
trace(path.nativePath); // C:\AIR Test
```

패키지 및 심볼 링크 작업

Adobe AIR 1.0 이상

여러 운영 체제에서 패키지 파일과 심볼 링크 파일을 지원합니다.

패키지 - Mac OS에서는 디렉토리가 패키지로 지정될 수 있으며 Mac OS Finder에서 디렉토리가 아니라 하나의 파일로 표시됩니다.

심볼 링크 - Mac OS, Linux 및 Windows Vista는 심볼 링크를 지원합니다. 심볼 링크는 파일이 디스크에 있는 다른 파일이나 디렉토리를 가리키는 것을 허용합니다. 심볼 링크는 별칭과 비슷하긴 하지만 같지는 않습니다. 별칭은 항상 디렉토리가 아닌 파일로 보고되며, 별칭을 읽고 써도 해당 항목이 가리키는 원래 파일 또는 디렉토리는 전혀 영향을 받지 않습니다. 반면에 심볼 링크는 해당 항목이 가리키는 파일 또는 디렉토리와 똑같이 동작합니다. 파일 또는 디렉토리로 보고될 수 있으며, 심볼 링크를 읽거나 쓰면 심볼 링크 자체가 아니라 해당 항목이 가리키는 파일 또는 디렉토리에 영향을 줍니다. 또한 Windows에서는 교차점(NTFs 파일 시스템에서 사용됨)을 참조하는 File 객체에 대한 isSymbolicLink 속성이 true로 설정됩니다.

File 클래스에는 File 객체가 패키지 또는 심볼 링크를 참조하는지 확인하기 위한 isPackage 및 isSymbolicLink 속성이 포함되어 있습니다.

다음 코드에서는 사용자 바탕 화면 디렉토리를 반복하여 패키지가 **아닌** 하위 디렉토리를 나열합니다.

```
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);
    }
}
```

다음 코드에서는 사용자 바탕 화면 디렉토리를 반복하여 심볼 링크가 **아닌** 파일 및 디렉토리를 나열합니다.

```
var desktopNodes:Array = File.desktopDirectory.getDirectoryListing();
for (var i:uint = 0; i < desktopNodes.length; i++)
{
    if (!desktopNodes[i].isSymbolicLink)
    {
        trace(desktopNodes[i].name);
    }
}
```

canonicalize() 메서드는 링크가 참조하는 파일 또는 디렉토리를 가리키도록 심볼 링크의 경로를 변경합니다. 다음 코드에서는 사용자 바탕 화면 디렉토리를 반복하여 심볼 링크인 파일이 참조하는 경로를 보고합니다.

```
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);
    }
}
```

볼륨에서 사용 가능한 공간 확인

Adobe AIR 1.0 이상

File 객체의 spaceAvailable 속성은 File 위치에서 사용 가능한 공간(바이트)입니다. 예를 들어 다음 코드에서는 응용 프로그램 저장소 디렉토리에서 사용 가능한 공간을 확인합니다.

```
trace(File.applicationStorageDirectory.spaceAvailable);
```

File 객체가 디렉토리를 참조하는 경우 spaceAvailable 속성은 해당 디렉토리에서 파일이 사용할 수 있는 공간을 나타냅니다. File 객체가 파일을 참조하는 경우 spaceAvailable 속성은 해당 파일이 커질 수 있는 공간을 나타냅니다. 해당 파일 위치가 존재하지 않으면 spaceAvailable 속성은 0으로 설정됩니다. File 객체가 심볼 링크를 참조하는 경우 spaceAvailable 속성은 심볼 링크가 가리키는 위치에서 사용 가능한 공간을 나타냅니다.

일반적으로 디렉토리나 파일에 사용할 수 있는 공간은 해당 디렉토리나 파일이 들어 있는 볼륨에서 사용 가능한 공간과 같습니다. 그러나 여유 공간에 계정 할당량 또는 디렉토리별 제한을 고려할 수 있습니다.

일반적으로 볼륨에 파일이나 디렉토리를 추가할 때는 파일의 실제 크기나 디렉토리 내용의 실제 크기보다 큰 공간이 필요합니다. 예를 들어 운영 체제에서 인덱스 정보를 저장할 추가 공간이 필요할 수 있습니다. 또는 필요한 디스크 섹터로 인해 추가 공간이 사용될 수 있습니다. 사용 가능한 공간은 동적으로 변경되기도 합니다. 따라서 보고된 파일 저장 공간이 모두 할당되지 않을 수도 있습니다. 파일 시스템에 쓰는 방법에 자세한 내용은 629페이지의 “파일 읽기 및 쓰기”를 참조하십시오.

StorageVolumeInfo.getStorageVolumes() 메서드는 마운트된 저장소 볼륨에 대해 보다 자세한 정보를 제공합니다(627페이지의 “저장소 볼륨을 사용한 작업” 참조).

기본 시스템 응용 프로그램으로 파일 열기

Adobe AIR 2 이상

AIR 2에서는 운영 체제에서 해당 파일을 열도록 등록된 응용 프로그램을 사용하여 파일을 열 수 있습니다. 예를 들어 AIR 응용 프로그램에서는 DOC 파일을 열도록 등록된 응용 프로그램을 사용하여 DOC 파일을 열 수 있습니다. 파일을 열려면 File 객체의 openWithDefaultApplication() 메서드를 사용합니다. 예를 들어 다음 코드에서는 사용자의 데스크톱에서 test.doc라는 파일을 열며, DOC 파일에 대한 기본 응용 프로그램으로 이 파일을 엽니다.

```
var file:File = File.deskopDirectory;
file = file.resolvePath("test.doc");
file.openWithDefaultApplication();
```

참고: Linux에서는 파일 이름 확장자가 아닌 파일의 MIME 유형에 따라 파일의 기본 응용 프로그램이 결정됩니다.

다음 코드에서는 사용자가 mp3 파일로 이동하여 mp3 파일 재생을 위한 기본 응용 프로그램으로 해당 파일을 열 수 있습니다.

```
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();
}
```

응용 프로그램 디렉토리에 있는 파일에는 openWithDefaultApplication() 메서드를 사용할 수 없습니다.

AIR에서는 사용자가 openWithDefaultApplication() 메서드를 사용하여 특정 파일을 열 수 없습니다. Windows의 경우, AIR에서 사용자가 EXE 또는 BAT와 같은 특정 파일 유형을 열 수 없습니다. Mac OS 및 Linux의 경우 AIR가 특정 응용 프로그램에서 실행되는 파일을 열 수 없습니다. 이러한 응용 프로그램에는 Mac OS의 경우 Terminal 및 AppletLauncher, Linux의 경우 csh, bash 또는 ruby가 포함됩니다. openWithDefaultApplication() 메서드를 사용하여 이러한 파일 중 하나를 열려고 하면 예외가 발생합니다. 열 수 없는 파일 유형의 전체 목록은 File.openWithDefaultApplication() 메서드의 언어 참조 항목을 참조하십시오.

참고: 이 제한은 기본 설치 프로그램으로 설치한 AIR 응용 프로그램(확장된 데스크톱 응용 프로그램)에는 적용되지 않습니다.

파일 시스템 정보 가져오기

Adobe AIR 1.0 이상

File 클래스에는 파일 시스템에 대한 유용한 정보를 제공하는 다음과 같은 정적 속성이 포함되어 있습니다.

속성	설명
File.lineEnding	호스트 운영 체제에서 사용한 행 끝 문자 시퀀스입니다. Mac OS 및 Linux에서는 줄 바꿈 문자입니다. Windows에서는 캐리지 리턴 문자와 그 뒤에 있는 줄 바꿈 문자입니다.
File.separator	호스트 운영 체제의 경로 구성 요소 분리 기호 문자입니다. Mac OS 및 Linux에서는 슬래시(/) 문자입니다. Windows에서는 백슬래시(\) 문자입니다.
File.systemCharset	호스트 운영 체제에서 파일에 사용한 기본 인코딩입니다. 이는 운영 체제에서 사용하는 언어에 해당하는 문자 세트와 관련이 있습니다.

Capabilities 클래스에는 파일을 작업할 때 유용한 시스템 정보도 포함되어 있습니다.

속성	설명
Capabilities.hasIME	플레이어가 실행되는 시스템에 IME가 설치되어 있는지(true) 또는 설치되어 있지 않은지(false)를 지정합니다.
Capabilities.language	플레이어가 실행되는 시스템의 언어 코드를 지정합니다.
Capabilities.os	현재 운영 체제를 지정합니다.

참고: Capabilities.os를 사용하여 시스템 특성을 결정할 때는 주의해야 합니다. 시스템 특성을 결정할 수 있는 더 구체적인 속성이 있다면 해당 속성을 사용하십시오. 그렇지 않을 경우 특정 플랫폼에서만 사용할 수 있는 코드가 작성될 위험이 있습니다. 예를 들어 다음과 같은 코드를 살펴봅니다.

```
var separator:String;
if (Capablities.os.indexOf("Mac") > -1)
{
    separator = "/";
}
else
{
    separator = "\\";
}
```

이 코드는 Linux에서 문제를 일으킵니다. 따라서 이 경우에는 File.separator 속성을 사용하는 것이 더 좋습니다.

디렉토리 작업

Adobe AIR 1.0 이상

런타임에서는 로컬 파일 시스템의 디렉토리를 사용하여 작업하기 위한 기능을 제공합니다.

디렉토리를 가리키는 File 객체를 만드는 방법에 대한 자세한 내용은 612페이지의 “File 객체로 디렉토리 가리키기”를 참조하십시오.

디렉토리 만들기

Adobe AIR 1.0 이상

File.createDirectory() 메서드를 사용하여 디렉토리를 만들 수 있습니다. 예를 들어 다음 코드에서는 사용자 홈 디렉토리의 하위 디렉토리로 AIR Test라는 디렉토리를 만듭니다.

```
var dir:File = File.userDirectory.resolvePath("AIR Test");
dir.createDirectory();
```

디렉토리가 이미 있는 경우 createDirectory() 메서드는 아무 작업도 수행하지 않습니다.

또는 일부 모드에서는 파일을 열 때 FileStream 객체가 디렉토리를 만듭니다. FileStream() 생성자의 fileMode 매개 변수를 FileMode.APPEND 또는 FileMode.WRITE로 설정하여 FileStream 인스턴스를 인스턴스화하면 없는 디렉토리가 만들어집니다. 자세한 내용은 629페이지의 “파일 읽기 및 쓰기 작업 과정”을 참조하십시오.

임시 디렉토리 만들기

Adobe AIR 1.0 이상

File 클래스에는 다음 예제와 같이 시스템의 임시 디렉토리 폴더에 디렉토리를 만드는 createTempDirectory() 메서드가 포함되어 있습니다.

```
var temp:File = File.createTempDirectory();
```

createTempDirectory() 메서드는 자동으로 고유한 임시 디렉토리를 만들어 고유한 새 위치를 확인하는 작업을 저장합니다.

임시 디렉토리에는 응용 프로그램의 세션 동안 사용되는 임시 파일을 저장할 수 있습니다. 시스템 임시 디렉토리에 고유한 새 임시 파일을 만들기 위한 createTempFile() 메서드도 있습니다.

임시 디렉토리는 모든 장치에서 자동으로 삭제되지 않으므로 응용 프로그램을 닫기 전에 삭제하는 것이 좋습니다.

디렉토리 열거

Adobe AIR 1.0 이상

File 객체의 getDirectoryListing() 메서드 또는 getDirectoryListingAsync() 메서드를 사용하여 디렉토리의 파일 및 하위 폴더를 가리키는 File 객체 배열을 가져올 수 있습니다.

예를 들어 다음 코드에서는 사용자 문서 디렉토리의 내용을 나열합니다(하위 디렉토리는 확인하지 않음).

```
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);
}
```

비동기 버전의 메서드를 사용할 경우 directoryListing 이벤트 객체에는 디렉토리를 가리키는 File 객체 배열인 files 속성이 있습니다.

```
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);
    }
}
```

디렉토리 복사 및 이동

Adobe AIR 1.0 이상

파일을 복사하거나 이동할 때와 동일한 메서드를 사용하여 디렉토리를 복사하거나 이동할 수 있습니다. 예를 들어 다음 코드에서는 디렉토리를 동기적으로 복사합니다.

```
var sourceDir:File = File.documentsDirectory.resolvePath("AIR Test");
var resultDir:File = File.documentsDirectory.resolvePath("AIR Test Copy");
sourceDir.copyTo(resultDir);
```

copyTo() 메서드의 overwrite 매개 변수를 true로 지정하는 경우 대상 파일이 소스 디렉토리에 있는지 여부와 상관없이 기존 대상 디렉토리의 모든 파일 및 폴더가 삭제되고 소스 디렉토리의 파일 및 폴더로 대체됩니다.

copyTo() 메서드의 newLocation 매개 변수로 지정하는 디렉토리는 결과 디렉토리를 포함할 **상위** 디렉토리를 지정하지 **않고** 결과 디렉토리의 경로를 지정합니다.

자세한 내용은 626페이지의 “파일 복사 및 이동”을 참조하십시오.

디렉토리 내용 삭제

Adobe AIR 1.0 이상

File 클래스에는 deleteDirectory() 메서드 및 deleteDirectoryAsync() 메서드가 포함되어 있습니다. 이러한 메서드는 첫 번째는 동기식으로 작동하고 두 번째는 비동기적으로 작동하여 디렉토리를 삭제합니다. 자세한 내용은 608페이지의 “AIR 파일 기본 사항”을 참조하십시오. 두 메서드 모두에는 deleteDirectoryContents 매개 변수(부울 값 사용)가 포함되어 있습니다. 이 매개 변수를 true로 설정하고(기본값은 false) 메서드를 호출하면 비어 있지 않은 디렉토리가 삭제됩니다. 그렇지 않으면 빈 디렉토리만 삭제됩니다.

예를 들어 다음 코드에서는 사용자 문서 디렉토리의 AIR Test 하위 디렉토리를 동기적으로 삭제합니다.

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.deleteDirectory(true);
```

다음 코드에서는 사용자 문서 디렉토리의 AIR Test 하위 디렉토리를 비동기적으로 삭제합니다.

```
var directory:File = File.documentsDirectory.resolvePath("AIR Test");
directory.addEventListener(Event.COMPLETE, completeHandler)
directory.deleteDirectoryAsync(true);

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

또한 시스템 휴지통으로 디렉토리를 이동하는 데 사용할 수 있는 moveToTrash() 및 moveToTrashAsync() 메서드도 포함되어 있습니다. 자세한 내용은 627페이지의 "휴지통으로 파일 이동"을 참조하십시오.

파일 작업

Adobe AIR 1.0 이상

AIR 파일 API를 사용하면 기본적인 파일 상호 작용 기능을 응용 프로그램에 추가할 수 있습니다. 예를 들어 파일 읽기, 쓰기, 복사, 삭제 등을 수행할 수 있습니다. 응용 프로그램에서 로컬 파일 시스템에 액세스할 수 있으므로 아직 이렇게 하지 않은 경우 979페이지의 "AIR 보안"을 참조합니다.

참고: 파일 유형을 AIR 응용 프로그램과 연결할 수 있습니다. 그러면 해당 파일을 두 번 클릭하여 응용 프로그램을 열 수 있습니다. 자세한 내용은 807페이지의 "파일 연결 관리"를 참조하십시오.

파일 정보 가져오기

Adobe AIR 1.0 이상

File 클래스에는 File 객체가 가리키는 파일 또는 디렉토리에 대한 정보를 제공하는 다음과 같은 속성이 포함되어 있습니다.

File 속성	설명
creationDate	로컬 디스크의 파일을 만든 날짜입니다.
creator	더 이상 사용되지 않습니다. 대신 extension 속성을 사용합니다. 이 속성은 Mac Os X 이전의 Mac Os 버전에서만 사용된 파일의 Macintosh 작성자 유형을 보고합니다.
downloaded	(AIR 2 이상) 참조한 파일 또는 디렉토리를 (인터넷에서) 다운로드했는지 여부를 나타냅니다. 이 속성은 파일을 다운로드한 것으로 플래그를 지정할 수 있는 운영 체제에서만 의미가 있습니다. • Windows XP 서비스 팩 2 이상, Windows Vista • Mac OS 10.5 이상
exists	참조된 파일 또는 디렉토리가 있는지 여부입니다.
extension	마지막 도트(".") 다음에 오는 이름 부분(도트는 포함되지 않음)인 파일 확장명입니다. 파일 이름에 도트가 없으면 확장명은 null입니다.
icon	파일에 대해 정의된 아이콘을 포함하는 Icon 객체입니다.
isDirectory	File 객체가 디렉토리를 참조하는지 여부입니다.
modificationDate	로컬 디스크 파일 또는 디렉토리가 마지막으로 수정된 날짜입니다.
name	로컬 디스크의 파일 또는 디렉토리의 이름(파일 확장명이 있는 경우 확장명도 포함)입니다.
nativePath	호스트 운영 체제 표현의 전체 경로입니다. 609페이지의 "File 객체의 경로"를 참조하십시오.
parent	File 객체가 나타내는 폴더 또는 파일을 포함하는 폴더입니다. File 객체가 파일 시스템의 루트에 있는 파일 또는 디렉토리를 참조하는 경우 이 속성은 null입니다.
size	로컬 디스크에 있는 파일의 크기(바이트)입니다.
type	더 이상 사용되지 않습니다. 대신 extension 속성을 사용합니다. Macintosh에서 이 속성은 Mac Os X 이전의 Mac Os 버전에서만 사용된 4개의 문자로 이루어진 파일 유형입니다.
url	파일 또는 디렉토리의 URL입니다. 609페이지의 "File 객체의 경로"를 참조하십시오.

이러한 속성에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 File 클래스 항목을 참조하십시오.

파일 복사 및 이동

Adobe AIR 1.0 이상

File 클래스에는 파일 또는 디렉토리를 복사하기 위한 copyTo() 및 copyToAsync()라는 두 개의 메서드가 포함되어 있습니다. File 클래스에는 파일 또는 디렉토리를 이동하기 위한 moveTo() 및 moveToAsync()라는 두 개의 메서드가 포함되어 있습니다. copyTo() 및 moveTo() 메서드는 동기식으로 작동하고 copyToAsync() 및 moveToAsync() 메서드는 비동기적으로 작동합니다. 자세한 내용은 608페이지의 "AIR 파일 기본 사항"을 참조하십시오.

파일을 복사 또는 이동하려면 두 개의 File 객체를 설정합니다. 한 객체는 복사 또는 이동할 파일을 가리키고 복사 또는 이동 메서드를 호출하는 객체이며 다른 객체는 대상(결과) 경로를 가리킵니다.

다음 예제에서는 사용자 문서 디렉토리의 AIR Test 하위 디렉토리에 있는 test.txt 파일을 같은 디렉토리에 copy.txt라는 파일로 복사합니다.

```
var original:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var newFile:File = File.resolvePath("AIR Test/copy.txt");
original.copyTo(newFile, true);
```

이 예제에서는 copyTo() 메서드의 overwrite 매개 변수(두 번째 매개 변수)를 true로 설정합니다. overwrite 매개 변수를 true로 설정하면 기존 대상 파일을 덮어쓰게 됩니다. 이 매개 변수는 선택적입니다. 이 매개 변수를 false(기본값)로 설정하면 이 작업은 대상 파일이 있는 경우 IOErrorEvent 이벤트를 전달하며 파일이 복사되지 않습니다.

"비동기" 버전의 복사 및 이동 메서드는 비동기적으로 작동합니다. 다음 코드에서와 같이 addEventListener() 메서드를 사용하면 작업의 완료 또는 오류 조건을 모니터링할 수 있습니다.

```
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.");
}
```

또한 File 클래스에는 파일 또는 디렉토리를 시스템 휴지통으로 이동하는 File.moveToTrash() 및 File.moveToTrashAsync() 메서드가 포함되어 있습니다.

파일 삭제

Adobe AIR 1.0 이상

File 클래스에는 deleteFile() 메서드와 deleteFileAsync() 메서드가 포함되어 있습니다. 이러한 메서드는 첫 번째는 동기식으로 작동하고 두 번째는 비동기적으로 작동하여 파일을 삭제합니다. 자세한 내용은 608페이지의 "AIR 파일 기본 사항"을 참조하십시오.

예를 들어 다음 코드에서는 사용자 문서 디렉토리에 있는 test.txt 파일을 동기적으로 삭제합니다.

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.deleteFile();
```

다음 코드에서는 사용자 문서 디렉토리의 test.txt 파일을 비동기적으로 삭제합니다.

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.addEventListener(Event.COMPLETE, completeHandler)
file.deleteFileAsync();

function completeHandler(event:Event):void {
    trace("Deleted.")
}
```

또한 시스템 휴지통으로 파일 또는 디렉토리를 이동하는 데 사용할 수 있는 moveToTrash() 및 moveToTrashAsync 메서드도 포함되어 있습니다. 자세한 내용은 627페이지의 "휴지통으로 파일 이동"을 참조하십시오.

휴지통으로 파일 이동

Adobe AIR 1.0 이상

File 클래스에는 moveToTrash() 메서드 및 moveToTrashAsync() 메서드가 포함되어 있습니다. 이러한 메서드는 첫 번째는 동기식으로 작동하고 두 번째는 비동기적으로 작동하여 파일 또는 디렉토리를 시스템 휴지통으로 보냅니다. 자세한 내용은 608페이지의 "AIR 파일 기본 사항"을 참조하십시오.

예를 들어 다음 코드에서는 사용자 문서 디렉토리에 있는 test.txt 파일을 시스템 휴지통으로 동기적으로 이동합니다.

```
var file:File = File.documentsDirectory.resolvePath("test.txt");
file.moveToTrash();
```

참고: 복구 가능한 휴지통 폴더의 개념을 지원하지 않는 운영 체제에서는 파일이 즉시 제거됩니다.

임시 파일 만들기

Adobe AIR 1.0 이상

File 클래스에는 다음 예제와 같이 시스템의 임시 디렉토리 폴더에 파일을 만드는 createTempFile() 메서드가 포함되어 있습니다.

```
var temp:File = File.createTempFile();
```

createTempFile() 메서드는 자동으로 고유한 임시 디렉토리를 만들어 고유한 새 위치를 확인하는 작업을 저장합니다.

임시 파일에는 응용 프로그램의 세션 동안 사용되는 정보를 일시적으로 저장할 수 있습니다. 시스템 임시 디렉토리에 고유한 새 임시 디렉토리를 만들기 위한 createTempDirectory() 메서드도 있습니다.

임시 파일은 모든 장치에서 자동으로 삭제되지 **않으므로** 응용 프로그램을 닫기 전에 삭제하는 것이 좋습니다.

저장소 볼륨을 사용한 작업

Adobe AIR 2 이상

AIR 2에서는 대량 저장소 볼륨이 마운트 또는 마운트 해제되는 경우를 감지할 수 있습니다. StorageVolumeInfo 클래스는 단일 storageVolumeInfo 객체를 정의합니다. StorageVolumeInfo.storageVolumeInfo 객체는 저장소 볼륨이 마운트된 경우 storageVolumeMount 이벤트를 전달합니다. 그리고 볼륨이 마운트 해제되면 storageVolumeUnmount 이벤트를 전달합니다. 이러한 이벤트는 StorageVolumeChangeEvent 클래스에서 정의됩니다.

참고: 현재 배포된 Linux 시스템에서는 StorageVolumeInfo 객체가 특정 위치에 마운트된 물리적 장치와 네트워크 장치를 위해서만 storageVolumeMount 및 storageVolumeUnmount 이벤트를 전달합니다.

StorageVolumeChangeEvent 클래스의 storageVolume 속성은 StorageVolume 객체입니다. StorageVolume 클래스는 저장소 볼륨의 기본 속성을 정의합니다.

- drive - Windows의 볼륨 드라이브 문자입니다. 기타 운영 체제의 경우 null입니다.

- fileSystemType - 저장소 볼륨의 파일 시스템 유형(예: "FAT", "NTFS", "HFS" 또는 "UFS")입니다.
- isRemoveable - 볼륨을 제거할 수 있는지 여부입니다(true 또는 false).
- isWritable - 볼륨에 쓸 수 있는지 여부입니다(true 또는 false).
- name - 볼륨의 이름입니다.
- rootDirectory - 볼륨의 루트 디렉토리에 해당하는 파일 객체입니다.

StorageVolumeChangeEvent 클래스에는 또한 rootDirectory 속성이 포함됩니다. rootDirectory 속성은 마운트 또는 마운트 해제된 저장소 볼륨의 루트 디렉토리를 참조하는 파일 객체입니다.

StorageVolumeChangeEvent 객체의 storageVolume 속성은 마운트되지 않은 볼륨에 대해 정의되지 않습니다(null). 그러나 이벤트의 rootDirectory 속성에 액세스할 수 있습니다.

다음 코드에서는 저장소 볼륨이 마운트될 때 저장소 볼륨의 이름과 파일 경로를 출력합니다.

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME_MOUNT,
onVolumeMount);
function onVolumeMount(event:StorageVolumeChangeEvent):void
{
    trace(event.storageVolume.name, event.rootDirectory.nativePath);
}
```

다음 코드에서는 저장소 볼륨이 마운트 해제될 때 저장소 볼륨의 파일 경로를 출력합니다.

```
StorageVolumeInfo.storageVolumeInfo.addEventListener(StorageVolumeChangeEvent.STORAGE_VOLUME_UNMOUNT,
onVolumeUnmount);
function onVolumeUnmount(event:StorageVolumeChangeEvent):void
{
    trace(event.rootDirectory.nativePath);
}
```

StorageVolumeInfo.storageVolumeInfo 객체는 getStorageVolumes() 메서드를 포함합니다. 이 메서드는 현재 마운트된 저장소 볼륨에 해당하는 StorageVolume 객체의 벡터를 반환합니다. 다음 코드에서는 마운트된 모든 저장소 볼륨의 이름 및 루트 디렉토리를 나열하는 방법을 보여 줍니다.

```
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);
}
```

참고: 최신 Linux 배포판에서는 getStorageVolumes() 메서드가 특정 위치에 마운트된 물리적 장치와 네트워크 드라이브에 해당하는 객체를 반환합니다.

File.getRootDirectories() 메서드는 루트 디렉토리를 나열합니다(614페이지의 “파일 시스템 루트 가리키기” 참조). 그러나 StorageVolume 객체(StorageVolumeInfo.getStorageVolumes() 메서드에 의해 열거됨)는 저장소 볼륨에 대한 보다 자세한 정보를 제공합니다.

StorageVolume 객체의 spaceAvailable 속성(rootDirectory 속성)을 사용하여 저장소 볼륨에서 사용 가능한 공간을 확인할 수 있습니다. 621페이지의 “볼륨에서 사용 가능한 공간 확인”을 참조하십시오.

기타 도움말 항목

StorageVolume

StorageVolumeInfo

파일 읽기 및 쓰기

Adobe AIR 1.0 이상

FileStream 클래스를 사용하면 AIR 응용 프로그램이 파일 시스템을 읽거나 파일 시스템에 쓸 수 있습니다.

파일 읽기 및 쓰기 작업 과정

Adobe AIR 1.0 이상

파일 읽기 및 쓰기의 작업 과정은 다음과 같습니다.

경로를 가리키는 File 객체를 초기화합니다.

File 객체는 작업할 파일 또는 나중에 만들 파일의 경로를 나타냅니다.

```
var file:File = File.documentsDirectory;
file = file.resolvePath("AIR Test/testFile.txt");
```

이 예제에서는 File 객체의 File.documentsDirectory 속성 및 resolvePath() 메서드를 사용하여 File 객체를 초기화합니다. 그러나 File 객체가 파일을 가리키도록 하는 여러 다른 방법이 있습니다. 자세한 내용은 615페이지의 "File 객체로 파일 가리키기"를 참조하십시오.

FileStream 객체를 초기화합니다.

FileStream 객체의 open() 메서드 또는 openAsync() 메서드를 호출합니다.

파일을 동기 또는 비동기 작업용으로 열지의 여부에 따라 호출하는 메서드가 달라집니다. File 객체를 열기 메서드의 file 매개 변수로 사용합니다. fileMode 매개 변수에는 파일을 사용할 방법을 지정하는 FileMode 클래스의 상수를 지정합니다.

예를 들어 다음 코드에서는 파일을 만들고 기존 데이터를 덮어쓰는 데 사용하는 FileStream 객체를 초기화합니다.

```
var fileStream:FileStream = new FileStream();
fileStream.open(file, FileMode.WRITE);
```

자세한 내용은 630페이지의 "FileStream 객체 초기화, 파일 열기 및 닫기" 및 630페이지의 "FileStream 열기 모드"를 참조하십시오.

openAsync() 메서드를 사용하여 파일을 비동기적으로 연 경우 FileStream 객체에 대한 이벤트 리스너를 추가하고 설정합니다.

이러한 이벤트 리스너 메서드는 여러 상황에서 FileStream 객체에 의해 전달된 이벤트에 응답합니다. 이러한 상황에는 파일에서 데이터를 읽는 경우, I/O 오류가 발생하는 경우 또는 기록할 데이터의 전체 분량이 기록된 경우가 포함됩니다.

자세한 내용은 634페이지의 "비동기적으로 연 FileStream 객체가 생성하는 이벤트 및 비동기 프로그래밍"을 참조하십시오.

필요한 경우 데이터를 읽고 쓰는 코드를 포함합니다.

읽기 및 쓰기와 관련된 FileStream 클래스에는 여러 메서드가 있습니다. 이러한 메서드는 "read" 또는 "write"로 시작합니다. 데이터를 읽거나 쓰는 데 사용할 메서드는 대상 파일의 데이터 형식에 따라 달라집니다.

예를 들어 대상 파일의 데이터가 UTF로 인코딩된 텍스트인 경우 readUTFBytes() 및 writeUTFBytes() 메서드를 사용할 수 있습니다. 데이터를 바이트 배열로 처리하려면 readByte(), readBytes(), writeByte() 및 writeBytes() 메서드를 사용할 수 있습니다. 자세한 내용은 634페이지의 "데이터 형식 및 사용할 읽기 및 쓰기 메서드 선택"을 참조하십시오.

파일을 비동기적으로 연 경우 읽기 메서드를 호출하려면 충분한 데이터를 사용할 수 있어야 합니다. 자세한 내용은 632페이지의 "FileStream 객체의 bytesAvailable 속성 및 읽기 버퍼"를 참조하십시오.

파일에 쓰기 전에 사용 가능한 디스크 공간을 확인하려면 File 객체의 spaceAvailable 속성을 확인합니다. 자세한 내용은 621페이지의 "볼륨에서 사용 가능한 공간 확인"을 참조하십시오.

파일 작업을 마쳤으면 FileStream 객체의 close() 메서드를 호출합니다.
close() 메서드를 호출하면 파일을 다른 응용 프로그램에서 사용할 수 있습니다.

자세한 내용은 630페이지의 “FileStream 객체 초기화, 파일 열기 및 닫기”를 참조하십시오.

FileStream 클래스를 사용하여 파일을 읽고 쓰는 샘플 응용 프로그램을 보려면 Adobe AIR 개발자 센터에서 다음 문서를 참조하십시오.

- 텍스트 파일 편집기 만들기
- 텍스트 파일 편집기 만들기
- XML 환경 설정 파일 읽기 및 쓰기

FileStream 객체 작업

Adobe AIR 1.0 이상

FileStream 클래스에서는 파일을 열고, 읽고, 쓰는 메서드를 정의합니다.

FileStream 열기 모드

Adobe AIR 1.0 이상

FileStream 객체의 open() 및 openAsync() 메서드 각각에는 다음을 비롯한 파일 스트림에 대한 일부 속성을 정의하는 fileMode 매개 변수가 포함되어 있습니다.

- 파일을 읽을 수 있는 기능
- 파일에 쓸 수 있는 기능
- 데이터를 쓸 때 데이터가 항상 파일 끝 이후에 추가되는지 여부
- 파일이 없는 경우 및 해당 상위 디렉토리가 없는 경우 수행할 작업

다음은 open() 및 openAsync() 메서드의 fileMode 매개 변수로 지정할 수 있는 여러 파일 모드입니다.

파일 모드	설명
FileMode.READ	파일이 읽기 전용으로 열리도록 지정합니다.
FileMode.WRITE	파일이 쓰기용으로 열리도록 지정합니다. 파일이 없는 경우 FileStream 객체가 열릴 때 만들어집니다. 파일이 있는 경우에는 기존 데이터가 삭제됩니다.
FileMode.APPEND	파일이 추가용으로 열리도록 지정합니다. 파일이 없는 경우 만들어집니다. 파일이 있는 경우에는 기존 데이터를 덮어쓰지 않고 모든 쓰기 작업이 파일에 끝에서 시작됩니다.
FileMode.UPDATE	파일이 읽기 및 쓰기용으로 열리도록 지정합니다. 파일이 없으면 만들어집니다. 파일에 대한 임의 읽기/쓰기 권한이 필요하면 이 모드를 지정합니다. 파일의 임의 위치에서 읽을 수 있고 파일에 쓰는 경우에는 기록되는 바이트만 덮어쓰고 다른 모든 바이트는 그대로 유지됩니다.

FileStream 객체 초기화, 파일 열기 및 닫기

Adobe AIR 1.0 이상

FileStream 객체를 열 때 파일의 데이터를 읽거나 파일에 데이터를 쓰는 데 이 객체를 사용할 수 있도록 만들 수 있습니다. FileStream 객체의 open() 또는 openAsync() 메서드에 File 객체를 전달하여 FileStream 객체를 엽니다.

```
var myFile:File = File.documentsDirectory.resolvePath("AIR Test/test.txt");
var myFileStream:FileStream = new FileStream();
myFileStream.open(myFile, FileMode.READ);
```

fileMode 매개 변수(open() 및 openAsync() 메서드의 두 번째 매개 변수)는 파일을 열 모드 즉, 읽기용, 쓰기용, 추가용 또는 업데이트용을 지정합니다. 자세한 내용은 이전 단원인 630페이지의 "FileStream 열기 모드"를 참조하십시오.

openAsync() 메서드를 사용하여 비동기 파일 작업을 위해 파일을 여는 경우 비동기 이벤트를 처리할 이벤트 리스너를 설정합니다.

```
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 {
    // ...
}
```

open() 또는 openAsync() 메서드 중 어떤 것을 사용하는지에 따라 파일이 동기 또는 비동기 작업용으로 열립니다. 자세한 내용은 608페이지의 "AIR 파일 기본 사항"을 참조하십시오.

FileStream 객체의 열기 모드에서 fileMode 매개 변수를 FileMode.READ 또는 FileMode.UPDATE로 설정하는 경우 FileStream 객체를 여는 즉시 데이터를 읽기 버퍼로 읽어 옵니다. 자세한 내용은 632페이지의 "FileStream 객체의 bytesAvailable 속성 및 읽기 버퍼"를 참조하십시오.

FileStream 객체의 close() 메서드를 사용하여 연결된 파일을 닫아 다른 응용 프로그램에서 사용할 수 있게 만들 수 있습니다.

FileStream 객체의 position 속성

Adobe AIR 1.0 이상

FileStream 객체의 position 속성은 다음 읽기 또는 쓰기 메서드에서 데이터를 읽거나 쓰는 위치를 결정합니다.

읽기 또는 쓰기 작업 전에 position 속성을 파일의 유효한 위치로 설정합니다.

예를 들어 다음 코드에서는 문자열 "hello"(UTF 인코딩)를 파일의 위치 8에 기록합니다.

```
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");
```

FileStream 객체를 처음으로 열 때 position 속성은 0으로 설정되어 있습니다.

읽기 작업 전에 position의 값은 0 이상이고 파일의 바이트 수(파일의 기존 위치)보다 작아야 합니다.

position 속성 값은 다음과 같은 조건에 해당하는 경우에만 수정됩니다.

- position 속성을 명시적으로 설정하는 경우
- 읽기 메서드를 호출하는 경우
- 쓰기 메서드를 호출하는 경우

FileStream 객체의 읽기 또는 쓰기 메서드를 호출하면 position 속성이 읽거나 쓰는 바이트 수만큼 즉시 증가합니다. 사용하는 읽기 메서드에 따라 position 속성이 읽을 바이트 수만큼 또는 사용 가능한 바이트 수만큼 증가합니다. 이후에 읽기 또는 쓰기 메서드를 호출하면 읽기 또는 쓰기가 새 위치에서 시작됩니다.

```
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
```

그러나 FileStream 객체를 추가 모드로 연 경우에는 예외적으로 쓰기 메서드를 호출한 후에도 position 속성이 변경되지 않습니다. 추가 모드에서는 position 속성의 값과 상관없이 데이터가 항상 파일의 끝에 기록됩니다.

파일을 비동기 작업용으로 연 경우에는 다음 코드 행이 실행될 때까지 쓰기 작업이 완료되지 않습니다. 그러나 여러 비동기 메서드를 차례대로 호출할 수 있으며, 런타임에서 이러한 메서드를 순서대로 실행합니다.

```
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.");
}
```

이 코드의 trace 출력은 다음과 같습니다.

```
started.
finished.
```

읽기 또는 쓰기 메서드를 호출한 후 즉시 또는 언제든지 position 값을 지정할 수 있으며 다음 읽기 또는 쓰기 작업은 해당 위치에서 시작됩니다. 예를 들어 다음 코드에서는 writeBytes() 작업을 호출하는 즉시 position 속성을 설정하며, 쓰기 작업이 완료된 후에도 position은 해당 값(300)으로 설정되어 있습니다.

```
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
```

FileStream 객체의 bytesAvailable 속성 및 읽기 버퍼
Adobe AIR 1.0 이상

읽기 기능을 사용하여 FileStream 객체를 여는 경우(open() 또는 openAsync() 메서드의 fileMode 매개 변수를 READ 또는 UPDATE로 설정) 런타임에서 데이터를 내부 버퍼에 저장합니다. FileStream 객체의 open() 또는 openAsync() 메서드를 호출하여 파일을 여는 즉시 FileStream 객체는 데이터를 버퍼로 읽어 오기 시작합니다.

open() 메서드를 사용하여 파일을 동기 작업용으로 연 경우 다음 코드에서와 같이 항상 position 포인터를 파일 경계 내의 유효한 위치로 설정하고 파일 경계 내의 원하는 양의 데이터를 읽기 시작할 수 있습니다. 이 예제에서는 파일에 100바이트 이상이 포함되어 있다고 가정합니다.

```
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);
```

파일을 동기 작업용으로 열었는지 비동기 작업용으로 열었는지와 관계없이 읽기 메서드는 항상 bytesAvalable 속성으로 표시되는 "사용 가능한" 바이트를 읽습니다. 동기적으로 읽는 경우 파일의 모든 바이트를 항상 사용할 수 있습니다. 비동기적으로 읽는 경우에는 position 속성에 지정된 위치부터 바이트를 사용할 수 있게 되고 일련의 비동기 버퍼 채우기가 progress 이벤트로 표시됩니다.

파일을 **동기** 작업용으로 연 경우 bytesAvailable 속성은 position 속성에서 파일 끝까지의 바이트 수를 나타내도록 설정됩니다(파일의 모든 바이트를 항상 읽을 수 있음).

파일을 **비동기** 작업용으로 연 경우에는 읽기 메서드를 호출하기 전에 읽기 버퍼에서 충분한 데이터를 사용했는지 확인해야 합니다. 파일을 비동기적으로 연 경우 읽기 작업이 진행되면 읽기 작업이 시작될 때 지정한 position부터 시작하여 파일의 데이터가 버퍼에 추가되고 bytesAvailable 속성이 읽은 각 바이트만큼 증가합니다. bytesAvailable 속성은 position 속성에서 지정한 위치의 바이트에서 시작하여 버퍼의 끝까지 사용할 수 있는 바이트 수를 나타냅니다. FileStream 객체는 정기적으로 progress 이벤트를 보냅니다.

파일을 비동기적으로 연 경우 데이터가 읽기 버퍼에서 사용할 수 있게 되면 FileStream 객체는 정기적으로 progress 이벤트를 전달합니다. 예를 들어 다음 코드에서는 데이터를 버퍼로 읽어 올 때와 마찬가지로 ByteArray 객체 bytes로 읽어 옵니다.

```
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);
}
```

파일을 비동기적으로 연 경우 읽기 버퍼의 데이터만 읽을 수 있습니다. 또한 데이터를 읽으면 읽기 버퍼에서 제거됩니다. 읽기 작업의 경우 읽기 작업을 호출하기 전에 데이터가 읽기 버퍼에 있는지 확인해야 합니다. 예를 들어 다음 코드에서는 파일의 위치 4000에서 시작되는 8000바이트의 데이터를 읽습니다.

```
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");
    }
}
```

쓰기 작업 동안 FileStream 객체는 읽기 버퍼로 데이터를 읽어 오지 않습니다. 쓰기 작업이 완료되면(쓰기 버퍼의 모든 데이터가 파일에 기록되면) FileStream 객체는 새 읽기 버퍼를 시작하고(읽기 기능을 사용하여 연결된 FileStream 객체를 연 것으로 가정할 경우) position 속성에 지정된 위치에서 시작하여 데이터를 읽기 버퍼로 읽어 오기 시작합니다. position 속성은 마지막으로 쓴 바이트의 위치이거나 사용자가 쓰기 작업 후 position 객체에 다른 값을 지정하는 경우 다른 위치일 수 있습니다.

비동기적으로 연 FileStream 객체가 생성하는 이벤트 및 비동기 프로그래밍
Adobe AIR 1.0 이상

openAsync() 메서드를 사용하여 파일을 비동기적으로 여는 경우 파일 읽기 및 쓰기가 비동기적으로 수행됩니다. 데이터를 읽기 버퍼로 읽어 오고 출력 데이터가 기록되면 다른 ActionScript 코드를 실행할 수 있습니다.

따라서 비동기적으로 연 FileStream 객체가 생성한 이벤트를 등록해야 합니다.

다음 코드에서처럼 progress 이벤트를 등록하면 새 데이터를 읽을 수 있게 될 때 알림을 받을 수 있습니다.

```
 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");
}
```

다음 코드에서처럼 complete 이벤트를 등록하여 전체 데이터를 읽을 수 있습니다.

```
 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");
}
```

비동기적으로 읽을 수 있도록 입력 데이터가 버퍼링되는 것과 마찬가지로 비동기 스트림에 쓰는 데이터는 버퍼링되어 파일에 비동기적으로 기록됩니다. 데이터가 파일에 기록되면 FileStream 객체는 OutputProgressEvent 객체를 정기적으로 전달합니다. OutputProgressEvent 객체에는 쓸 나머지 바이트 수로 설정되는 bytesPending 속성이 포함되어 있습니다. outputProgress 이벤트를 등록하면 버퍼가 실제로 파일에 기록될 때 알림을 받을 수 있으며 대개 진행률 대화 상자를 표시할 수 있습니다. 그러나 일반적으로 이렇게 할 필요는 없습니다. 특히 기록되지 않는 바이트에 대해 걱정할 필요 없이 close() 메서드를 호출할 수 있습니다. FileStream 객체는 데이터 쓰기를 계속하며 최종 바이트가 파일에 기록되고 기본 파일이 닫힌 후에 close 이벤트가 전달됩니다.

데이터 형식 및 사용할 읽기 및 쓰기 메서드 선택
Adobe AIR 1.0 이상

모든 파일은 디스크에 있는 바이트의 집합입니다. ActionScript에서 파일의 데이터는 항상 ByteArray로 표시될 수 있습니다. 예를 들어 다음 코드에서는 파일의 데이터를 bytes라는 ByteArray 객체로 읽어 옵니다.

```
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);
}
```

마찬가지로 다음 코드에서는 bytes라는 ByteArray의 데이터를 파일에 씁니다.

```
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);
```

그러나 데이터를 ActionScript ByteArray 객체에 저장하지 않는 경우가 종종 있습니다. 그리고 데이터 파일이 지정된 파일 형식인 경우가 종종 있습니다.

예를 들어 파일의 데이터가 텍스트 파일 형식이고 이러한 데이터를 String 객체로 표시하려고 할 수 있습니다.

이 때문에 FileStream 클래스에는 ByteArray 객체 외의 유형에서 데이터를 읽고 쓰기 위한 읽기 및 쓰기 메서드가 포함되어 있습니다. 예를 들어 다음 코드에서처럼 readMultiByte() 메서드를 사용하여 파일에서 데이터를 읽고 문자열로 저장할 수 있습니다.

```
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");
}
```

readMultiByte() 메서드의 두 번째 매개 변수는 ActionScript에서 데이터를 해석하기 위해 사용하는 텍스트 형식(이 예제에서는 "iso-8859-1")을 지정합니다. Adobe AIR에서는 일상적인 문자 집합 인코딩이 지원됩니다(지원되는 문자 집합 참조).

또한 FileStream 클래스에는 UTF-8 문자 세트를 사용하여 읽기 버퍼의 데이터를 문자열로 읽는 readUTFBytes() 메서드가 포함되어 있습니다. UTF-8 문자 세트의 문자는 길이가 다양하므로 progress 이벤트에 응답하는 메서드에서는 readUTFBytes()를 사용하지 마십시오. 그럴 경우 읽기 버퍼 끝에 있는 데이터는 불완전한 문자를 표시할 수 있습니다. 가변 길이 문자 인코딩과 함께 readMultiByte() 메서드를 사용하는 경우도 마찬가지입니다. 따라서 FileStream 객체가 complete 이벤트를 전달할 때 전체 데이터 집합을 읽습니다.

또한 텍스트 파일의 String 객체로 작업하기 위한 writeMultiByte() 및 writeUTFBytes()와 같은 유사한 메서드가 있습니다.

readUTF() 및 writeUTF() 메서드(readUTFBytes() 및 writeUTFBytes()와 혼동해서는 안 됨)도 파일의 텍스트 데이터를 읽고 쓰지만 텍스트 데이터 앞에 표준 텍스트 파일에는 일반적으로 없는 텍스트 데이터의 길이를 지정하는 데이터가 옵니다.

일부 UTF로 인코딩된 텍스트 파일은 엔디안 및 인코딩 형식(예: UTF-16 또는 UTF-32)을 정의하는 "UTF-BOM"(바이트 순서 표시) 문자로 시작됩니다.

텍스트 파일 읽기 및 쓰기에 대한 예제는 637페이지의 “예제: XML 파일을 XML 객체로 읽기”를 참조하십시오.

readObject() 및 writeObject()는 복잡한 ActionScript 객체의 데이터를 저장 및 검색하는 편리한 방법입니다. 이 경우 데이터가 AMF(ActionScript Message Format)로 인코딩됩니다. Adobe AIR, Flash Player, Flash Media Server, Flex Data Services에는 이 형식으로 데이터를 작업할 수 있는 API가 포함되어 있습니다.

이 외에도 readDouble() 및 writeDouble()과 같은 몇 가지 다른 읽기 및 쓰기 메서드가 있습니다. 그러나 이러한 메서드를 사용하는 경우 파일 형식이 이러한 메서드에서 정의하는 데이터 형식과 일치하는 지 확인해야 합니다.

파일 형식이 단순한 텍스트 형식보다 복잡한 경우가 종종 있습니다. 예를 들어 MP3 파일에는 MP3 파일과 관련된 압축 해제 및 디코딩 알고리즘으로만 해석할 수 있는 압축 데이터가 포함되어 있습니다. 또한 MP3 파일에는 파일에 대한 메타태그 정보(노래 제목 및 가수 등)를 포함하는 ID3 태그도 포함되어 있을 수 있습니다. 여러 버전의 ID3 형식이 있지만 가장 단순한 형식(ID3 버전 1)에 대해 638페이지의 "예제: 임의 액세스를 사용하여 데이터 읽기 및 쓰기" 단원에서 설명합니다.

이미지, 데이터베이스, 응용 프로그램 문서 등을 위한 다른 파일 형식은 구조가 다르므로 ActionScript에서 이러한 데이터를 작업하려면 데이터의 구성 방법에 대해 알고 있어야 합니다.

load() 및 save() 메서드 사용

Flash Player 10 이상, Adobe AIR 1.5 이상

Flash Player 10에서는 FileReference 클래스에 load() 및 save() 메서드가 추가되었습니다. 이러한 메서드는 AIR 1.5에도 있으며 File 클래스는 FileReference 클래스의 메서드를 상속합니다. 이러한 메서드는 사용자가 Flash Player에서 파일 데이터를 안전하게 로드 및 저장할 수 있도록 설계되었습니다. 그러나 AIR 응용 프로그램에서 이러한 메서드를 사용하여 파일을 쉽게 비동기적으로 로드 및 저장할 수도 있습니다.

예를 들어 다음 코드에서는 문자열을 텍스트 파일에 저장합니다.

```
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.");
}
```

save() 메서드의 data 매개 변수는 String, XML 또는 ByteArray 값을 사용할 수 있습니다. 인수가 String 또는 XML 값이면 이 메서드는 파일을 UTF-8 인코딩 텍스트 파일로 저장합니다.

이 코드 샘플이 실행되면 응용 프로그램에 사용자가 저장된 파일 대상을 선택하는 대화 상자가 표시됩니다.

다음 코드에서는 UTF-8 인코딩 텍스트 파일에서 문자열을 로드합니다.

```
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);
}
```

FileStream 클래스는 load() 및 save() 메서드가 제공하는 것보다 많은 기능을 제공합니다.

- FileStream 클래스를 사용하여 동기 및 비동기적으로 데이터를 읽고 쓸 수 있습니다.
- FileStream 클래스를 사용하면 점증적으로 파일에 쓸 수 있습니다.
- FileStream 클래스를 사용하면 파일을 열어 임의로 액세스할 수 있습니다(파일의 모든 섹션에서 읽고 쓸 수 있음).
- FileStream 클래스를 사용하면 open() 또는 openAsync() 메서드의 fileMode 매개 변수를 설정하여 파일에 대한 파일 액세스 유형을 지정할 수 있습니다.
- FileStream 클래스를 사용하면 사용자에게 [열기] 또는 [저장] 대화 상자를 제공하지 않고도 파일에 데이터를 저장할 수 있습니다.
- FileStream 클래스로 데이터를 읽으면 바이트 배열 이외의 유형을 직접 사용할 수 있습니다.

예제: XML 파일을 XML 객체로 읽기

Adobe AIR 1.0 이상

다음 예제에서는 XML 데이터가 포함된 텍스트 파일을 읽고 쓰는 방법을 보여 줍니다.

파일을 읽으려면 다음과 같이 File 및 FileStream 객체를 초기화하고, FileStream의 readUTFBytes() 메서드를 호출하고 문자열을 XML 객체로 변환합니다.

```
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();
```

마찬가지로 파일에 데이터를 쓰려면 적절한 File 및 FileStream 객체를 설정한 다음 FileStream 객체의 쓰기 메서드를 호출하면 됩니다. 다음 코드에서와 같이 문자열 버전의 XML 데이터를 쓰기 메서드에 전달합니다.

```
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();
```

이 예제에서는 파일이 UTF-8 형식인 것으로 가정하고 readUTFBytes() 및 writeUTFBytes() 메서드를 사용합니다. 그렇지 않은 경우 다른 메서드를 사용해야 합니다. 자세한 내용은 634페이지의 “데이터 형식 및 사용할 읽기 및 쓰기 메서드 선택”을 참조하십시오.

이전 예제에서는 동기 작업용으로 연 FileStream 객체를 사용합니다. 또한 파일을 비동기 작업용으로 열 수도 있습니다. 이 경우 이벤트에 응답하는 이벤트 리스너 함수가 필요합니다. 예를 들어 다음 코드에서는 XML 파일을 비동기적으로 읽는 방법을 보여 줍니다.

```
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();
}
```

processXMLData() 메서드는 전체 파일을 읽기 버퍼로 읽어 온 경우(FileStream 객체가 complete 이벤트를 전달할 때) 호출됩니다. 이 메서드는 readUTFBytes() 메서드를 호출하여 읽은 데이터의 문자열 버전을 가져오고 해당 문자열을 기반으로 XML 객체 prefsXML을 만듭니다.

이러한 기능을 보여 주는 샘플 응용 프로그램을 보려면 XML 환경 설정 파일 읽기 및 쓰기를 참조하십시오.

예제: 임의 액세스를 사용하여 데이터 읽기 및 쓰기

Adobe AIR 1.0 이상

MP3 파일에는 파일의 시작 또는 끝에 있는 섹션이며 녹음을 식별하는 메타데이터가 들어 있는 ID3 태그가 포함될 수 있습니다. ID3 태그 형식 자체에는 여러 개정이 있습니다. 이 예제에서는 가장 단순한 ID3 형식(ID3 버전1.0)이 포함된 MP3 파일을 "데이터에 대한 임의 액세스"를 사용하여 읽고 쓰는 방법을 설명합니다. 즉, 파일의 임의 위치에서 읽고 씁니다.

ID3 버전 1 태그가 있는 MP3 파일에는 파일 끝의 최종 128 바이트에 ID3 데이터가 포함되어 있습니다.

임의 읽기/쓰기 액세스용으로 파일에 액세스할 경우 open() 또는 openAsync() 메서드에 대한 fileMode 매개 변수로 FileMode.UPDATE를 지정해야 합니다.

```
var file:File = File.documentsDirectory.resolvePath("My Music/Sample ID3 v1.mp3");
var fileStr:FileStream = new FileStream();
fileStr.open(file, FileMode.UPDATE);
```

이렇게 하면 파일을 읽고 쓸 수 있습니다.

파일이 열리면 position 포인터를 파일 끝 전의 위치 128바이트로 설정할 수 있습니다.

```
fileStr.position = file.size - 128;
```

ID3 v1.0 형식에서 ID3 태그 데이터를 파일의 마지막 128바이트에 저장하도록 지정하므로 이 코드는 position 속성을 파일의 위치로 설정합니다. 또한 이 사양에는 다음과 같이 규정되어 있습니다.

- 태그의 첫 번째 3바이트에는 문자열 "TAG"가 포함됩니다.
- 다음 30자에는 MP3 트랙의 제목이 문자열로 포함됩니다.
- 다음 30자에는 가수 이름이 문자열로 포함됩니다.
- 다음 30자에는 앨범 이름이 문자열로 포함됩니다.
- 다음 4자에는 연도가 문자열로 포함됩니다.
- 다음 30자에는 주석이 문자열로 포함됩니다.
- 다음 바이트에는 트랙의 장르를 나타내는 코드가 포함됩니다.
- 모든 데이터는 ISO 8859-1 형식입니다.

id3TagRead() 메서드는 데이터를 읽은 후(complete 이벤트가 전달되면) 데이터를 확인합니다.

```
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);
    }
}
```

또한 임의 액세스로 파일에 쓸 수 있습니다. 예를 들어 id3Title 변수를 파싱하여 대/소문자가 올바른지(String 클래스의 메서드 사용) 확인한 후 다음과 같이 수정된 문자열(newTitle이라고 함)을 파일에 기록합니다.

```
fileStr.position = file.length - 125;    // 128 - 3
fileStr.writeMultiByte(newTitle, "iso-8859-1");
```

ID3 버전 1 표준을 준수하려면 newTitle 문자열은 30자여야 하며 끝에 문자 코드 0(String.fromCharCode(0))을 추가해야 합니다.

39장: 로컬 데이터 저장

Flash Player 9 이상, Adobe AIR 1.0 이상

SharedObject 클래스를 사용하여 소량의 데이터를 클라이언트 컴퓨터에 저장할 수 있습니다. Adobe AIR에서는 EncryptedLocalStore 클래스를 사용하여 AIR 응용 프로그램에서 소량의 중요한 개인 정보 관련 사용자 데이터를 로컬 컴퓨터에 저장할 수도 있습니다.

또한 파일 시스템에서 파일을 읽고 쓰거나 Adobe AIR에서 로컬 데이터베이스 파일에 액세스할 수도 있습니다. 자세한 내용은 595페이지의 "파일 시스템 작업" 및 651페이지의 "AIR에서 로컬 SQL 데이터베이스를 사용한 작업"을 참조하십시오.

공유 객체와 관련해서는 수많은 보안 요인이 있습니다. 자세한 내용은 948페이지의 "보안"의 978페이지의 "공유 객체"를 참조하십시오.

공유 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

"Flash 쿠키"라고도 하는 공유 객체는 방문한 사이트에서 사용자의 컴퓨터에 만들 수 있는 데이터 파일입니다. 예를 들어 자주 방문하는 웹 사이트의 모양과 느낌을 개인 설정할 수 있도록 하는 등의 방식으로 웹 브라우징 환경을 향상시키는 데 공유 객체가 가장 자주 사용됩니다.

공유 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

공유 객체는 브라우저 쿠키처럼 작동합니다. SharedObject 클래스를 사용하여 데이터를 사용자의 로컬 하드 디스크에 저장해 놓았다가 동일한 세션 중에 또는 다른 세션에서 해당 데이터를 호출할 수 있습니다. 응용 프로그램은 동일한 도메인에서 실행되는 경우에 한해 해당 응용 프로그램 고유의 SharedObject 데이터에만 액세스할 수 있습니다. 데이터는 서버로 전송되지 않으며 다른 도메인에서 실행되는 다른 응용 프로그램에서는 액세스할 수 없습니다. 그러나 동일한 도메인의 응용 프로그램에서는 액세스 가능하도록 할 수 있습니다.

공유 객체와 쿠키 비교

Flash Player 9 이상, Adobe AIR 1.0 이상

쿠키와 공유 객체는 매우 비슷합니다. 대부분의 웹 프로그래머는 쿠키의 작동 방식을 잘 알고 있으므로 쿠키와 로컬 공유 객체를 비교하여 설명하면 도움이 될 것입니다.

RFC 2109 표준을 따르는 쿠키에는 일반적으로 다음과 같은 속성이 있습니다.

- 쿠키는 만료될 수 있으며 기본적으로 세션 종료 시 만료되는 경우가 많습니다.
- 클라이언트에서 사이트별로 쿠키를 비활성화할 수 있습니다.
- 쿠키 수는 총 300개, 사이트당 최대 20개로 제한됩니다.
- 일반적으로 각 쿠키의 크기는 4KB로 제한됩니다.
- 쿠키가 보안 위협 요소로 파악되어 클라이언트에서 비활성화되는 경우도 있습니다.
- 쿠키는 클라이언트 브라우저에서 지정한 위치에 저장됩니다.

- 쿠키는 HTTP를 통해 클라이언트에서 서버로 전송됩니다.

  반면에, 공유 객체에는 다음과 같은 속성이 있습니다.

- 공유 객체는 기본적으로 만료되지 않습니다.
- 기본적으로 각 공유 객체의 크기는 100KB로 제한됩니다.
- 공유 객체는 String, Array, Date 등의 단순 데이터 유형을 저장할 수 있습니다.
- 공유 객체는 응용 프로그램에서 지정한 위치(사용자의 홈 디렉토리 내)에 저장됩니다.
- 공유 객체는 클라이언트와 서버 간에 전송되지 않습니다.

SharedObject 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

SharedObject 클래스를 사용하면 사용 중인 SharedObject 객체의 현재 크기를 감지할 수 있을 뿐 아니라 공유 객체를 만들거나 삭제할 수도 있습니다.

공유 객체 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

SharedObject 객체를 만들려면 다음 구문이 있는 SharedObject.getLocal() 메서드를 사용합니다.

```
SharedObject.getLocal("objectName" [, pathname]): SharedObject
```

다음 예제에서는 mySO라는 공유 객체를 만듭니다.

```
public var mySO:SharedObject;
mySO = SharedObject.getLocal("preferences");
```

이를 실행하면 클라이언트 컴퓨터에 preferences.sol이라는 파일이 만들어집니다.

local이라는 용어는 공유 객체의 위치를 나타냅니다. 이 경우에는 Adobe® Flash® Player가 클라이언트의 홈 디렉토리에 로컬로 SharedObject 파일을 저장합니다.

공유 객체를 만들면 Flash Player가 샌드박스 내부에 응용 프로그램 및 도메인에 대한 새 디렉토리를 만듭니다. 또한 Flash Player가 SharedObject 데이터를 저장하는 *.sol 파일도 만듭니다. 이 파일의 기본 위치는 사용자 홈 디렉토리의 하위 디렉토리입니다. 다음 표에서는 이 디렉토리의 기본 위치를 보여 줍니다.

운영 체제	위치
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_n`**`ame.sol`
Linux/Unix	`/home/username/.macromedia/Flash_Player/#SharedObjects/`**`web_domain/path_to_appl`**`ication/`**`application_name/object_name`**`.sol`

#SharedObjects 디렉토리 아래에는 임의로 이름이 지정된 디렉토리가 있습니다. 그 아래에는 호스트 이름과 일치하는 디렉토리, 그 다음에는 응용 프로그램에 대한 경로, 마지막에는 *.sol 파일이 있습니다.

예를 들어 로컬 호스트의 /sos라는 하위 디렉토리 내에 있는 MyApp.swf라는 응용 프로그램을 요청하는 경우 Flash Player가 *.sol 파일을 Windows XP의 다음 위치에 저장합니다.

```
c:/Documents and Settings/fred/Application Data/Macromedia/Flash
Player/#SharedObjects/KROKWXRK/#localhost/sos/MyApp.swf/data.sol
```

참고: SharedObject.getLocal() 메서드에 이름을 입력하지 않으면 Flash Player가 파일 이름을 undefined.sol로 지정합니다.

기본적으로 Flash는 도메인당 최대 100KB의 영구 SharedObject 객체를 로컬로 저장할 수 있습니다. 이 값은 사용자가 구성할 수 있습니다. 응용 프로그램에서 공유 객체에 저장하려는 데이터가 100KB를 초과하는 경우 Flash Player에 [로컬 저장소] 대화 상자가 표시되어 액세스를 요청하는 도메인의 로컬 저장소 크기 증가 허용 또는 거부를 사용자가 선택할 수 있도록 합니다.

경로 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

선택적 매개 변수인 **pathname**을 사용하여 SharedObject 파일의 위치를 지정할 수 있습니다. 이 파일은 해당 도메인의 SharedObject 디렉토리 내 하위 디렉토리여야 합니다. 예를 들어 로컬 호스트에 있는 응용 프로그램을 요청하는 경우 다음과 같이 지정합니다.

```
mySO = SharedObject.getLocal("myObjectFile","/");
```

Flash Player는 SharedObject 파일을 /#localhost 디렉토리에 기록합니다. 또는 응용 프로그램이 오프라인인 경우에는 /localhost에 기록합니다. 이는 클라이언트에 있는 둘 이상의 응용 프로그램이 동일한 공유 객체에 액세스할 수 있도록 하려는 경우에 유용합니다. 이 경우 클라이언트는 두 Flex 응용 프로그램을 실행할 수 있으며 두 응용 프로그램 모두 도메인의 루트에 있는 공유 객체에 대한 경로를 지정합니다. 그러면 클라이언트는 두 응용 프로그램 모두에서 동일한 공유 객체에 액세스할 수 있습니다. 일시적으로 여러 응용 프로그램 간에 데이터를 공유하려면 LocalConnection 객체를 사용합니다.

존재하지 않는 디렉토리를 지정하면 Flash Player에서 SharedObject 파일을 만들지 않습니다.

공유 객체에 데이터 추가

Flash Player 9 이상, Adobe AIR 1.0 이상

SharedObject 객체의 data 속성을 사용하여 SharedObject의 *.sol 파일에 데이터를 추가합니다. 공유 객체에 새 데이터를 추가하려면 다음 구문을 사용합니다.

```
sharedObject_name.data.variable = value;
```

다음 예제에서는 SharedObject에 userName, itemNumbers 및 adminPrivileges 속성과 해당 값을 추가합니다.

```
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;
```

data 속성에 값을 할당한 후 SharedObject 파일에 해당 값을 기록하도록 Flash Player에 지시해야 합니다. SharedObject 파일에 값을 기록하도록 Flash Player에 지시하려면 다음과 같이 SharedObject.flush() 메서드를 사용합니다.

```
mySO.flush();
```

SharedObject.flush() 메서드를 호출하지 않으면 Flash Player가 응용 프로그램이 종료될 때 파일에 값을 기록합니다. 그러나 Flash Player가 저장해야 할 데이터의 크기가 기본 설정을 초과하는 경우 사용 가능한 공간을 늘릴 수 있는 기회가 사용자에게 제공되지 않습니다. 따라서 SharedObject.flush()를 호출하는 것이 좋습니다.

flush() 메서드를 사용하여 공유 객체를 사용자의 하드 드라이브에 기록하는 경우 다음 예제에서 보듯이 사용자가 Flash Player 설정 관리자(www.macromedia.com/support/documentation/kr/flashplayer/help/settings_manager07.html)를 사용하여 명시적으로 로컬 저장을 사용하지 않도록 설정했는지 확인해야 합니다.

```
var so:SharedObject = SharedObject.getLocal("test");
trace("Current SharedObject size is " + so.size + " bytes.");
so.flush();
```

공유 객체에 객체 저장

SharedObject의 data 속성에 Array 또는 String 등의 단순 객체를 저장할 수 있습니다.

다음 예제는 공유 객체와의 상호 작용을 제어하는 메서드를 정의하는 ActionScript 클래스입니다. 이러한 메서드를 사용하면 사용자가 공유 객체에서 객체를 추가하거나 제거할 수 있습니다. 이 클래스는 단순 객체가 포함된 ArrayCollection을 저장합니다.

```
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();
        }
    }

}
```

다음 Flex 응용 프로그램은 필요한 각 공유 객체 유형별로 ActionScript 클래스 인스턴스를 만듭니다. 그런 다음 사용자가 블로그나 사이트 URL을 추가 또는 제거할 때 해당 클래스에 대한 메서드를 호출합니다.

```
<?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>
```

공유 객체에 유형이 지정된 객체 저장

공유 객체에 유형이 지정된 ActionScript 인스턴스를 저장할 수 있습니다. flash.net.registerClassAlias() 메서드를 호출하여 클래스를 등록하면 됩니다. 클래스의 인스턴스를 만들어 공유 객체의 데이터 멤버에 저장한 후 나중에 객체를 읽으면 유형이 지정된 인스턴스를 가져옵니다. 기본적으로 SharedObject의 objectEncoding 속성은 AMF3 인코딩을 지원하며 SharedObject 객체에서 저장된 인스턴스의 압축을 풉니다. 따라서 저장된 인스턴스의 유형은 registerClassAlias() 메서드를 호출할 때 지정한 유형과 동일하게 유지됩니다.

(iOS에만 해당) 로컬 공유 객체의 클라우드 백업 방지

Adobe AIR 3.7 이상, iOS에만 해당

SharedObject.preventBackup 속성을 설정하여 로컬 공유 객체를 iOS 클라우드 백업 서비스에 백업할지 여부를 제어할 수 있습니다. 이는 다시 생성하거나 다시 다운로드할 수 있지만 오프라인에서 사용할 때 응용 프로그램이 올바르게 작동하도록 하는 데 필요한 내용에 대한 Apple의 요구 사항입니다.

여러 공유 객체 만들기

동일한 Flex 응용 프로그램에 대해 여러 공유 객체를 만들 수 있습니다. 다음 예제에서 볼 수 있는 것과 같이 각 공유 객체에 서로 다른 인스턴스 이름을 할당하면 됩니다.

```
public var mySO:SharedObject = SharedObject.getLocal("preferences");
public var mySO2:SharedObject = SharedObject.getLocal("history");
```

그러면 Flex 응용 프로그램의 로컬 디렉토리에 preferences.sol 파일과 history.sol 파일이 만들어집니다.

보안 SharedObject 만들기

Flash Player 9 이상, Adobe AIR 1.0 이상

getLocal() 또는 getRemote()를 사용하여 로컬 또는 원격 SharedObject를 만들 때 이 공유 객체에 대한 액세스를 HTTPS 연결을 통해 전달되는 SWF 파일로만 제한할지 결정하는 secure라는 선택 매개 변수가 있습니다. 이 매개 변수를 true로 설정하고 SWF 파일이 HTTPS를 통해 전달되면 Flash Player에서 새 보안 공유 객체를 만들거나 기존의 보안 공유 객체에 대한 참조를 가져옵니다. 이 보안 공유 객체는 보안 매개 변수가 true로 설정된 SharedObject.getLocal()을 호출하는 HTTPS를 통해 전달되는 SWF 파일에 의해서만 읽거나 쓸 수 있습니다. 이 매개 변수를 false로 설정하고 SWF 파일이 HTTPS를 통해 전달되면 Flash Player에서 새 공유 객체를 만들거나 기존의 공유 객체에 대한 참조를 가져옵니다.

이 공유 객체는 HTTPS가 아닌 연결을 통해 제공되는 SWF 파일에 의해 읽거나 쓸 수 있습니다. SWF 파일이 비 HTTPS 연결을 통해 전달되고 이 매개 변수를 true로 설정하려고 하면 새 공유 객체를 만들지 못하고(또는 이전에 만든 보안 공유 객체에 대한 액세스하지 못함), 오류가 발생하며, 공유 객체가 null로 설정됩니다. HTTPS가 아닌 연결에서 다음 코드를 실행하려고 하면 SharedObject.getLocal() 메서드에서 오류를 발생시킵니다.

```
try
{
    var so:SharedObject = SharedObject.getLocal("contactManager", null, true);
}
catch (error:Error)
{
    trace("Unable to create SharedObject.");
}
```

만들어진 공유 객체는 이 매개 변수 값에 상관없이 도메인에 허용된 총 디스크 공간에 반영됩니다.

공유 객체의 내용 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

값은 data 속성 내에 있는 공유 객체에 저장됩니다. 다음 예제에 표시된 대로 for..in 루프를 사용하여 공유 객체 인스턴스의 각 값을 반복할 수 있습니다.

```
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]);
}
```

공유 객체 삭제

Flash Player 9 이상, Adobe AIR 1.0 이상

클라이언트에서 SharedObject를 삭제하려면 SharedObject.clear() 메서드를 사용합니다. 이 메서드를 사용해도 응용 프로그램의 공유 객체에 대한 기본 경로의 디렉토리는 삭제되지 않습니다.

다음 예제를 실행하면 클라이언트에서 SharedObject 파일이 삭제됩니다.

```
public function destroySharedObject():void {
    mySO.clear();
}
```

SharedObject 예제

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 예제에서는 해당 객체를 수동으로 직렬화하거나 직렬화를 해제하지 않고도 Date 객체와 같은 간단한 객체를 SharedObject 객체에 저장할 수 있음을 보여 줍니다.

다음 예제는 첫 번째 방문을 환영한다는 메시지로 시작합니다. 사용자가 [Log Out]을 클릭할 때 응용 프로그램에서 공유 객체에 현재 날짜를 저장합니다. 다음에 이 응용 프로그램을 시작하거나 페이지를 새로 고치면 다시 방문한 것을 환영하는 메시지와 함께 지난 번에 로그아웃한 시간을 알려 줍니다.

응용 프로그램의 동작을 확인하려면 응용 프로그램을 시작하고 [Log Out]을 클릭한 다음 페이지를 새로 고치십시오. 그러면 이전 방문 시 [Log Out] 버튼을 클릭했던 날짜 및 시간이 표시됩니다. 언제든지 [Delete LSO] 버튼을 클릭하여 저장된 정보를 삭제할 수 있습니다.

```
<?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>
```

암호화된 로컬 저장소

EncryptedLocalStore(ELS) 클래스는 응용 프로그램의 개인 데이터에 대한 작은 캐시로 사용할 수 있는 암호화된 로컬 저장소 메커니즘을 제공합니다. ELS 데이터는 응용 프로그램 간에 공유할 수 없습니다. ELS는 응용 프로그램에서 로그인 자격 증명 및 다른 개인 정보 같은 쉽게 재생성되는 항목을 저장할 수 있도록 하기 위한 것입니다. ELS 데이터는 "암호화된 로컬 저장소의 제한 사항"과 아래 "유용한 방법"에 설명된 것처럼 영구적인 것으로 간주해서는 안 됩니다.

참고: AIR은 암호화된 로컬 저장소 외에도 SQL 데이터베이스에 저장된 내용에 대한 암호화 기능도 제공합니다. 자세한 내용은 689페이지의 “SQL 데이터베이스에서 암호화 사용”을 참조하십시오.

암호화된 로컬 저장소를 사용하여 웹 서비스에 대한 로그인 자격 증명과 같은 보호해야 하는 정보를 캐시할 수 있습니다. ELS는 다른 사용자에게 비공개로 유지해야 하는 정보를 저장하는 데 적합합니다. 그러나 ELS가 동일한 사용자 계정의 다른 프로세스 실행으로부터 데이터를 보호하는 것은 아닙니다. 따라서 DRM이나 암호화 키 같은 기밀 응용 프로그램 데이터를 보호하는 데는 적합하지 않습니다.

데스크톱 플랫폼에서 AIR는 DPAPI(Windows), KeyChain(Mac OS 및 iOS), KeyRing 또는 KWallet(Linux)을 사용하여 암호화된 로컬 저장소를 각 응용 프로그램 및 사용자와 연결합니다. 암호화된 로컬 저장소는 AES-CBC 128비트 암호화를 사용합니다.

Android에서 EncryptedLocalStorage 클래스가 저장하는 데이터는 암호화되지 않습니다. 대신 운영 체제에서 제공하는 사용자 수준 보안으로 데이터가 보호됩니다. Android 운영 체제는 모든 응용 프로그램에 별도의 사용자 ID를 할당합니다. 응용 프로그램은 해당 응용 프로그램 고유의 파일 및 공용 위치(예: 이동식 저장소 카드)에 만들어진 파일만 액세스할 수 있습니다. "루트로 지정된" Android 장치에서 루트 권한으로 실행 중인 응용 프로그램은 다른 응용 프로그램의 파일에 액세스할 수 있습니다. 따라서 루트로 지정된 장치에서 암호화된 로컬 저장소는 루트로 지정되지 않은 장치에서와 같은 높은 수준의 데이터 보호를 제공하지 않습니다.

암호화된 로컬 저장소의 정보는 응용 프로그램 보안 샌드박스의 AIR 응용 프로그램 내용에만 사용할 수 있습니다.

AIR 응용 프로그램을 업데이트할 경우 업데이트된 버전은 암호화된 로컬 저장소의 기존 데이터에 계속해서 액세스할 수 있습니다. 단, 다음 경우는 예외입니다.

- stronglyBound 매개 변수를 true로 설정한 상태로 항목을 추가한 경우
- 기존 버전과 업데이트 버전이 모두 AIR 1.5.3 이전에 게시되었고 업데이트가 마이그레이션 서명으로 서명된 경우.

암호화된 로컬 저장소의 한계

암호화된 로컬 저장소의 데이터는 사용자의 운영 체제 계정 자격 증명으로 보호됩니다. 해당 사용자로 로그인하지 않는 한은 저장소의 데이터에 액세스할 수 없는 것입니다. 하지만 인증받은 사용자가 실행하는 다른 응용 프로그램의 데이터 액세스까지 차단할 수는 없습니다.

이러한 데이터 액세스가 가능하려면 사용자 인증을 거쳐야 하기 때문에 사용자 계정이 직접 손상되지 않는 한 해당 사용자의 개인 데이터는 계속 보호됩니다. 단, DRM(디지털 권한 관리) 또는 라이센스에 사용되는 키처럼 다른 사용자의 무단 액세스로부터 보호되어야 하는 응용 프로그램 데이터는 안전하지 않습니다. 따라서 그와 같은 정보는 ELS에 저장하지 않는 것이 좋습니다. 그저 암호와 같이 사용자의 개인 데이터를 저장하기에 적합한 장소일 뿐입니다.

ELS의 데이터는 여러 가지 이유로 손실될 수 있습니다. 예를 들어 사용자가 응용 프로그램을 제거하거나 암호화된 파일을 삭제할 수 있습니다. 또는 업데이트 결과 게시자 ID가 변경될 수도 있습니다. 따라서 ELS는 영구적인 데이터 저장소가 아니라 개인 캐시로 간주되어야 합니다.

stronglyBound 매개 변수는더 이상 사용되지 않으므로 true로 설정해서는 안 됩니다. 이 매개 변수를 true로 설정해도 추가적인 데이터 보호 효과는 없습니다. 또한 게시자 ID가 변하지 않아도 응용 프로그램이 업데이트될 때마다 데이터 액세스 권한이 사라집니다.

암호화된 로컬 저장소는 저장된 데이터가 10MB를 초과할 경우 속도가 느려질 수 있습니다.

AIR 응용 프로그램을 제거해도 암호화된 로컬 저장소에 저장된 데이터가 삭제되지는 않습니다.

유용한 방법

ELS는 다음과 같이 활용하는 것이 바람직합니다.

- 암호 같은 민감한 사용자 데이터를 저장하는 데 사용합니다. 이때 stronglyBound를 false로 설정합니다.
- DRM 키나 라이센싱 토큰 같은 응용 프로그램 기밀 데이터를 저장하는 데 사용하지 마십시오.
- ELS 데이터가 손실될 경우 응용 프로그램에서 ELS에 저장된 데이터를 다시 만들 수 있는 방법을 제공합니다. 예를 들어 필요한 경우 사용자에게 계정 자격 증명을 다시 입력하라는 메시지를 표시합니다.
- stronglyBound 매개 변수를 사용하지 않습니다.
- stronglyBound를 true로 설정하더라도 저장된 항목은 업데이트 중에 마이그레이션되지 않습니다. 대신 업데이트 후 데이터를 다시 만듭니다.

- 상대적으로 적은 양의 데이터만 저장합니다. 많은 양의 데이터에는 암호화가 적용되는 AIR SQL 데이터베이스를 사용하십시오.

기타 도움말 항목

flash.data.EncryptedLocalStore

암호화된 로컬 저장소에 데이터 추가

EncryptedLocalStore 클래스의 setItem() 정적 메서드를 사용하여 로컬 저장소에 데이터를 저장할 수 있습니다. 데이터는 문자열을 키로 사용하여 바이트 배열로 해시 테이블에 저장됩니다.

예를 들어, 다음 코드에서는 암호화된 로컬 저장소에 문자열을 저장합니다.

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes);
```

setItem() 메서드의 세 번째 매개 변수인 stronglyBound 매개 변수는 선택 사항입니다. 이 매개 변수가 true로 설정되면 다음과 같이 암호화된 로컬 저장소가 저장된 항목을, 저장하는 AIR 응용 프로그램의 디지털 서명 및 비트에 바인딩합니다.

```
var str:String = "Bob";
var bytes:ByteArray = new ByteArray();
bytes.writeUTFBytes(str);
EncryptedLocalStore.setItem("firstName", bytes, false);
```

stronglyBound를 true로 설정하여 저장된 항목의 경우 이후의 getItem() 호출은 호출하는 AIR 응용 프로그램이 저장하는 응용 프로그램과 동일한 경우에만 성공합니다(응용 프로그램 디렉토리에 있는 파일의 데이터가 변경되지 않은 경우). 호출하는 AIR 응용 프로그램이 저장하는 응용 프로그램과 다른 경우 강력하게 바인딩된 항목에 대해 getItem()을 호출하면 응용 프로그램에서 Error 예외를 발생시킵니다. 응용 프로그램을 업데이트하는 경우 해당 응용 프로그램에서는 암호화된 로컬 저장소에 이전에 작성한 강력하게 바인딩된 데이터를 읽을 수 없습니다. 모바일 장치에서 stronglyBound를 true로 설정해도 무시됩니다. 이 매개 변수는 항상 false로 처리됩니다.

stronglyBound 매개 변수를 false(기본값)로 설정할 경우 제작자 ID만 동일하면 응용 프로그램이 데이터를 읽을 수 있습니다. 응용 프로그램의 비트는 변경될 수 있으며(동일한 제작자로 서명되어야 함) 데이터를 저장한 응용 프로그램의 비트와 정확히 동일해야 하는 것은 아닙니다. 원래 ID와 동일한 제작자 ID를 사용하여 업데이트된 응용 프로그램은 데이터에 계속 액세스할 수 있습니다.

참고: 실제로, stronglyBound를 true로 설정하더라도 데이터 보호가 강화되지는 않습니다. "악의적인" 사용자가 여전히 응용 프로그램을 변경하여 ELS에 저장된 항목에 액세스할 수 있습니다. 뿐만 아니라, stronglyBound가 true로 설정되어 있든 false로 설정되어 있든 데이터는 외부의 비사용자 위협으로부터 마찬가지 수준으로 보호됩니다. 따라서 stronglyBound를 true로 설정하는 것은 권장되지 않습니다.

암호화된 로컬 저장소에서 데이터 액세스

Adobe AIR 1.0 이상

다음 예제와 같이 EncryptedLocalStore.getItem() 메서드를 사용하여 암호화된 로컬 저장소에서 값을 가져올 수 있습니다.

```
var storedValue:ByteArray = EncryptedLocalStore.getItem("firstName");
trace(storedValue.readUTFBytes(storedValue.length)); // "Bob"
```

암호화된 로컬 저장소에서 데이터 제거

Adobe AIR 1.0 이상

다음 예제와 같이 EncryptedLocalStore.removeItem() 메서드를 사용하여 암호화된 로컬 저장소에서 값을 삭제할 수 있습니다.

```
EncryptedLocalStore.removeItem("firstName");
```

다음 예제와 같이 EncryptedLocalStore.reset() 메서드를 호출하여 암호화된 로컬 저장소에서 모든 데이터를 지울 수 있습니다.

```
EncryptedLocalStore.reset();
```

40장: AIR에서 로컬 SQL 데이터베이스를 사용한 작업

Adobe AIR 1.0 이상

Adobe® AIR®에는 로컬 SQL 데이터베이스를 만들고 사용하는 기능이 포함되어 있습니다. 런타임은 오픈 소스 SQLite 데이터베이스 시스템을 사용하여 다양한 표준 SQL 기능에 대한 지원과 함께 SQL 데이터베이스 엔진을 포함합니다. 로컬 SQL 데이터베이스를 사용하여 로컬 영구 데이터를 저장할 수 있습니다. 예를 들어, 응용 프로그램 데이터, 응용 프로그램 사용자 설정, 문서 또는 응용 프로그램에서 로컬로 저장할 수 있는 다른 유형의 데이터에 로컬 SQL 데이터베이스를 사용할 수 있습니다.

로컬 SQL 데이터베이스

Adobe AIR 1.0 이상

SQL 데이터베이스 사용에 대한 간략한 설명과 코드 예제를 보려면 Adobe Developer Connection의 다음 퀵 스타트 문서를 참조하십시오.

- 로컬 SQL 데이터베이스를 사용하여 비동기식으로 작업(Flex)
- 로컬 SQL 데이터베이스를 사용하여 동기적으로 작업(Flex)
- 암호화된 데이터베이스 사용(Flex)
- 로컬 SQL 데이터베이스를 사용하여 비동기적으로 작업(Flash)
- 로컬 SQL 데이터베이스를 사용하여 동기적으로 작업(Flash)
- 암호화된 데이터베이스 사용(Flash)

Adobe AIR에는 런타임에서 실행되는 SQL 기반 관계형 데이터베이스 엔진과 AIR 응용 프로그램이 실행되는 컴퓨터(예: 컴퓨터의 하드 드라이브)의 데이터베이스 파일에 로컬로 저장된 데이터가 포함되어 있습니다. 데이터베이스가 실행되고 데이터 파일이 로컬로 저장되어 있으므로 네트워크 연결을 사용할 수 있는지 여부와 관계없이 AIR 응용 프로그램에서 데이터베이스를 사용할 수 있습니다. 따라서 런타임의 로컬 SQL 데이터베이스 엔진은 특히 SQL과 관계형 데이터베이스에 익숙한 사용자에게 로컬 영구 응용 프로그램 데이터를 저장할 수 있는 간편한 메커니즘을 제공합니다.

로컬 SQL 데이터베이스의 용도

Adobe AIR 1.0 이상

사용자의 로컬 컴퓨터에 응용 프로그램 데이터를 저장하려는 모든 용도에 AIR 로컬 SQL 데이터베이스 기능을 사용할 수 있습니다. Adobe AIR에는 데이터를 로컬로 저장할 수 있는 몇 가지 메커니즘이 포함되어 있는데 각 메커니즘에는 서로 다른 장점이 있습니다. AIR 응용 프로그램에서 가능한 몇 가지 로컬 SQL 데이터베이스의 용도는 다음과 같습니다.

- 데이터 지향 응용 프로그램(예: 주소록)의 경우 데이터베이스를 사용하여 기본 응용 프로그램 데이터를 저장할 수 있습니다.
- 사용자가 저장하고 공유할 문서를 만드는 문서 지향 응용 프로그램의 경우 각 문서를 사용자가 지정한 위치에 데이터베이스 파일로 저장할 수 있습니다. 그러나 데이터베이스가 암호화되지 않은 경우에는 모든 AIR 응용 프로그램에서 데이터베이스 파일을 열 수 있습니다. 중요할 수 있는 문서에는 암호화를 사용하는 것이 좋습니다.

- 네트워크 인식 응용 프로그램의 경우 데이터베이스를 사용하여 응용 프로그램 데이터의 로컬 캐시를 저장하거나 네트워크 연결을 사용할 수 없을 때 일시적으로 데이터를 저장할 수 있습니다. 로컬 데이터베이스를 네트워크 데이터 저장소와 동기화할 수 있는 메커니즘을 만들 수도 있습니다.
- 모든 응용 프로그램의 경우 데이터베이스를 사용하여 윈도우 크기 및 위치 등의 응용 프로그램 정보나 사용자 옵션과 같은 개별 사용자의 응용 프로그램 설정을 저장할 수 있습니다.

기타 도움말 항목

Christophe Coenraets: Employee Directory on AIR for Android

Raymond Camden: jQuery and AIR - Moving from web page to application

AIR 데이터베이스 및 데이터베이스 파일

Adobe AIR 1.0 이상

개별 Adobe AIR 로컬 SQL 데이터베이스는 컴퓨터의 파일 시스템에 단일 파일로 저장됩니다. 런타임에는 데이터베이스 파일을 만들고 구성하는 작업과 데이터베이스 파일에서 데이터를 조작하고 검색하는 작업을 관리하는 SQL 데이터베이스 엔진이 포함되어 있습니다. 런타임은 데이터베이스 데이터가 파일 시스템에 저장되는 방식이나 위치를 지정하지 않습니다. 각 데이터베이스는 한 파일 안에 완전히 저장됩니다. 사용자가 파일 시스템에서 데이터베이스 파일이 저장되는 위치를 지정합니다. 한 AIR 응용 프로그램이 하나 이상의 개별 데이터베이스(즉, 개별 데이터베이스 파일)에 액세스할 수 있습니다. 런타임은 각 데이터베이스를 파일 시스템에서 한 파일로 저장하기 때문에 응용 프로그램의 설계와 운영 체제의 파일 액세스 제한에 따라 필요한 경우 데이터베이스를 찾을 수 있습니다. 각 사용자가 특정 데이터에 대한 별도의 데이터베이스 파일을 사용하거나, 한 컴퓨터의 모든 응용 프로그램 사용자가 공유 데이터를 사용하기 위해 단일 데이터베이스 파일에 액세스할 수 있습니다. 데이터는 한 컴퓨터에 로컬이기 때문에 여러 컴퓨터의 사용자에게 자동으로 공유되지 않습니다. 로컬 SQL 데이터베이스 엔진은 원격 또는 서버측 데이터베이스에 대해 SQL 문을 실행하는 기능을 제공하지 않습니다.

관계형 데이터베이스

Adobe AIR 1.0 이상

관계형 데이터베이스는 컴퓨터에서 데이터를 저장하고 검색할 수 있는 메커니즘입니다. 데이터는 테이블로 구성되어 있습니다. 행은 레코드나 항목을 나타내고 열("필드"라고도 함)은 각 레코드를 개별 값으로 나눕니다. 예를 들어, 주소록 응용 프로그램에는 "friends" 테이블이 포함될 수 있습니다. 테이블의 각 행은 데이터베이스에 저장된 한 친구를 나타냅니다. 테이블의 열은 성, 이름, 생일 등의 데이터를 나타냅니다. 데이터베이스는 테이블의 친구 행마다 각 열에 별도의 값을 저장합니다.

관계형 데이터베이스는 한 항목이 다른 유형의 항목과 연결되어 있거나 관련되어 있는 복잡한 데이터를 저장하도록 설계되었습니다. 관계형 데이터베이스에서 일대다 관계(한 레코드가 다른 유형의 여러 레코드와 관련될 수 있는 관계)인 모든 데이터는 여러 테이블에 나뉘어 있어야 합니다. 예를 들어, 주소록 응용 프로그램에 각 친구의 여러 전화 번호를 저장하려는 경우를 일대다 관계라고 할 수 있습니다. "friends" 테이블에는 각 친구의 모든 개인 정보가 포함됩니다. 별도의 "phone numbers" 테이블에는 모든 친구의 모든 전화 번호가 포함됩니다.

각 테이블에서는 친구와 전화 번호에 대한 데이터를 저장하는 것 외에도 개별 친구 레코드와 전화 번호를 일치시키기 위해 두 테이블의 관계를 추적하는 데이터가 필요합니다. 이 데이터를 기본 키라고 합니다. 기본 키는 테이블의 각 행을 해당 테이블의 다른 행과 구별하는 고유한 식별자입니다. 기본 키는 테이블의 각 레코드를 자연적으로 구분하는 데이터 항목 중 하나를 의미하는 "자연 키"일 수 있습니다. "friends" 테이블에서 친구 중에 생일이 같은 사람이 없다는 사실을 아는 경우 생일 열을 "friends" 테이블의 기본 키(자연 키)로 사용할 수 있습니다. 자연 키가 없으면 응용 프로그램에서 행을 구분하는 데 사용하는 인공 값인 "friend ID"와 같은 별도의 기본 키 열을 만듭니다.

기본 키를 사용하여 여러 테이블 간의 관계를 설정할 수 있습니다. 예를 들어, "friends" 테이블에 각 행(각 친구)에 대한 고유 숫자가 포함된 "friend ID"가 있는 경우 관련된 "phone numbers" 테이블을 두 열로 구성하여 한 열에는 전화 번호에 해당하는 친구의 "friend ID"를 넣고 다른 열에는 실제 전화 번호를 넣을 수 있습니다. 이런 식으로 한 친구가 사용하는 전화 번호가 몇 개이든

간에 모든 전화 번호를 “phone numbers” 테이블에 저장하고 “friend ID” 기본 키를 사용하여 관련된 친구와 연결할 수 있습니다. 한 테이블의 기본 키가 관련된 테이블에서 레코드 간의 연결을 지정하는 데 사용되는 경우 관련된 테이블의 값을 외래 키라고 합니다. 많은 데이터베이스와 달리 AIR 로컬 데이터베이스 엔진에서는 삽입되거나 업데이트된 외래 키 값에 해당하는 행이 기본 키 테이블에 있는지 자동으로 확인하는 제약 조건인 FOREIGN KEY 제약 조건을 만들 수 없습니다. 그렇지만 외래 키 관계는 관계형 데이터베이스 구조의 중요한 부분이며 데이터베이스에서 테이블 간의 관계를 만들 때 외래 키를 사용해야 합니다.

SQL 정보

Adobe AIR 1.0 이상

SQL(구조적 쿼리 언어)은 관계형 데이터베이스에서 데이터를 조작하고 검색하는 데 사용됩니다. SQL은 절차적 언어가 아니라 설명적 언어입니다. SQL문은 컴퓨터에 데이터를 검색하는 방법에 대한 명령을 제공하는 대신 원하는 데이터 집합을 설명합니다. 데이터베이스 엔진은 해당 데이터를 검색하는 방법을 결정합니다.

SQL 언어는 ANSI(American National Standards Institute)에서 표준화하고 있습니다. Adobe AIR 로컬 SQL 데이터베이스는 SQL-92 표준을 대부분 지원합니다.

Adobe AIR에서 지원되는 SQL 언어에 대한 구체적인 설명은 1005페이지의 “로컬 데이터베이스의 SQL 지원”을 참조하십시오.

SQL 데이터베이스 클래스

Adobe AIR 1.0 이상

ActionScript 3.0에서 로컬 SQL 데이터베이스로 작업하려면 flash.data 패키지에서 이러한 클래스의 인스턴스를 사용합니다.

클래스	설명
flash.data.SQLConnection	데이터베이스(데이터베이스 파일)를 만들고 열 수 있도록 하며 데이터베이스 수준 작업을 수행하고 데이터베이스 트랜잭션을 제어하는 메서드를 제공합니다.
flash.data.SQLStatement	문 텍스트 정의와 매개 변수 값 설정을 비롯하여 데이터베이스에서 실행되는 단일 SQL 문(단일 쿼리 또는 명령)을 나타냅니다.
flash.data.SQLResult	SELECT 문의 결과 행, UPDATE 또는 DELETE 문의 영향을 받는 행의 수를 비롯한 문 실행 결과에 대한 정보를 가져올 수 있도록 합니다.

데이터베이스 구조를 설명하는 스키마 정보를 얻으려면 flash.data 패키지에서 다음 클래스를 사용합니다.

클래스	설명
flash.data.SQLSchemaResult	SQLConnection.loadSchema() 메서드를 호출하여 생성된 데이터베이스 스키마 결과에 대한 컨테이너 역할을 합니다.
flash.data.SQLTableSchema	데이터베이스의 단일 테이블을 설명하는 정보를 제공합니다.
flash.data.SQLViewSchema	데이터베이스의 단일 뷰를 설명하는 정보를 제공합니다.
flash.data.SQLIndexSchema	데이터베이스에 있는 뷰나 테이블의 단일 열을 설명하는 정보를 제공합니다.
flash.data.SQLTriggerSchema	데이터베이스의 단일 트리거를 설명하는 정보를 제공합니다.

flash.data 패키지의 다른 클래스는 SQLConnection 클래스 및 SQLColumnSchema 클래스와 함께 사용되는 상수를 제공합니다.

클래스	설명
flash.data.SQLMode	SQLConnection.open() 및 SQLConnection.openAsync() 메서드의 openMode 매개 변수에 사용할 수 있는 값을 나타내는 상수의 집합을 정의합니다.
flash.data.SQLColumnNameStyle	SQLConnection.columnNameStyle 속성에 사용할 수 있는 값을 나타내는 상수의 집합을 정의합니다.
flash.data.SQLTransactionLockType	SQLConnection.begin() 메서드의 option 매개 변수에 사용할 수 있는 값을 나타내는 상수의 집합을 정의합니다.
flash.data.SQLCollationType	SQLColumnSchema() 생성자의 defaultCollationType 매개 변수와 SQLColumnSchema.defaultCollationType 속성에 사용할 수 있는 값을 나타내는 상수의 집합을 정의합니다.

또한 flash.events 패키지의 다음 클래스는 사용하는 이벤트와 지원하는 상수를 나타냅니다.

클래스	설명
flash.events.SQLEvent	작업이 성공적으로 실행되는 경우 SQLConnection 또는 SQLStatement 인스턴스가 전달하는 이벤트를 정의합니다. 각 작업에는 SQLEvent 클래스에 정의된 이벤트 유형 상수가 연결되어 있습니다.
flash.events.SQLErrorEvent	작업에서 오류가 발생하는 경우 SQLConnection 또는 SQLStatement 인스턴스가 전달하는 이벤트를 정의합니다.
flash.events.SQLUpdateEvent	연결된 데이터베이스 중 하나의 테이블 데이터가 INSERT, UPDATE 또는 DELETE SQL 문을 실행한 결과로 변경되는 경우 SQLConnection 인스턴스가 전달하는 이벤트를 정의합니다.

마지막으로 flash.errors 패키지의 다음 클래스는 데이터베이스 작업 오류에 대한 정보를 제공합니다.

클래스	설명
flash.errors.SQLError	시도된 작업과 실패의 원인을 비롯하여 데이터베이스 작업 오류에 대한 정보를 제공합니다.
flash.errors.SQLErrorOperation	오류가 발생한 데이터베이스 작업을 나타내는 SQLError 클래스의 operation 속성에 사용할 수 있는 값을 나타내는 상수의 집합을 정의합니다.

동기 및 비동기 실행 모드

Adobe AIR 1.0 이상

로컬 SQL 데이터베이스로 작업하는 코드를 작성하는 경우 비동기 실행 모드나 동기 실행 모드 중 하나에서 해당 데이터베이스 작업 실행을 지정합니다. 일반적으로 코드 예제에서는 두 방법으로 각 작업을 수행하는 방법을 보여 주므로 사용자의 요구에 가장 적합한 예제를 사용할 수 있습니다.

비동기 실행 모드에서는 사용자가 런타임에 명령을 제공하고 런타임은 요청된 작업이 완료되거나 실패할 때 이벤트를 전달합니다. 먼저 사용자가 데이터베이스 엔진에 작업을 수행하도록 지시합니다. 데이터베이스 엔진은 응용 프로그램이 실행되는 동안 백그라운드에서 작업을 수행합니다. 마지막으로 작업이 완료되거나 작업이 실패하면 데이터베이스 엔진은 이벤트를 전달합니다. 이벤트에 의해 트리거된 코드에서 이후 작업을 수행합니다. 이 방법에는 상당한 이점이 있습니다. 런타임은 기본 응용 프로그램 코드가 실행되는 동안 백그라운드에서 데이터베이스 작업을 수행하므로 데이터베이스 작업이 많은 시간을 차지하는 경우 응용 프로그램이 계속 실행됩니다. 가장 중요한 점은 화면이 정지하는 일 없이 사용자가 응용 프로그램과 계속 상호 작용할 수 있다는 것입니다. 그렇지만 비동기 작업 코드는 다른 코드보다 작성하기가 복잡할 수 있습니다. 이러한 현상은 여러 종속 작업을 다양한 이벤트 리스너 메서드에 분할해야 하는 경우에 나타납니다.

개념적으로 작업을 여러 단계로 이루어진 단일 시퀀스(동기 작업의 집합)로 코딩하는 것이 작업 집합을 몇 가지 이벤트 리스너 메서드로 분할하는 것보다 간단합니다. Adobe AIR에서는 비동기 데이터베이스 작업 외에도 데이터베이스 작업을 동기적으로 실행할 수도 있습니다. 동기 실행 모드에서는 작업이 백그라운드에서 실행되지 않고 다른 모든 응용 프로그램 코드와 동일한 실행 시퀀스에서 실행됩니다. 사용자가 데이터베이스 엔진에 작업을 수행하도록 지시합니다. 그러면 코드가 데이터베이스 엔진이 작업하는 시점에 일시 중지됩니다. 작업이 완료되면 코드의 다음 줄에서 실행이 계속됩니다.

작업이 비동기적으로 실행되는지, 아니면 동기적으로 실행되는지는 SQLConnection 수준에서 설정됩니다. 단일 데이터베이스 연결을 사용하는 경우 일부 작업이나 문을 동기적으로 실행하고 다른 작업이나 문을 비동기적으로 실행할 수 없습니다. SQLConnection 메서드를 호출하여 데이터베이스를 여는 방법으로 SQLConnection이 동기 실행 모드에서 작동하는지, 아니면 비동기 실행 모드에서 작동하는지를 지정합니다. SQLConnection.open()을 호출하면 연결이 동기 실행 모드에서 작동하고, SQLConnection.openAsync()를 호출하면 연결이 비동기 실행 모드에서 작동합니다. SQLConnection 인스턴스가 open() 또는 openAsync()를 사용하여 데이터베이스에 연결되면 데이터베이스 연결을 닫고 다시 열지 않는 한 동기 실행 모드나 비동기 실행 모드로 고정됩니다.

각 실행 모드에는 장점이 있습니다. 각 모드의 측면이 대부분 유사하지만 각 모드에서 작업할 때 명심해야 할 차이점이 있습니다. 이러한 항목에 대한 자세한 내용과 각 모드에서의 작업에 대한 제안은 685페이지의 “동기 및 비동기 데이터베이스 작업 사용”을 참조하십시오.

데이터베이스 생성 및 수정

Adobe AIR 1.0 이상

응용 프로그램에서 데이터를 추가하거나 검색하려면 응용 프로그램에서 액세스할 수 있는 테이블이 정의되어 있는 데이터베이스가 있어야 합니다. 여기에서는 데이터베이스를 만들고 데이터베이스에 데이터 구조를 만드는 작업에 대해 설명합니다. 이러한 작업은 데이터 삽입 및 검색보다 자주 사용되지 않지만 대부분의 응용 프로그램에 필요합니다.

기타 도움말 항목

Mind the Flex: Updating an existing AIR database

데이터베이스 생성

Adobe AIR 1.0 이상

데이터베이스 파일을 만들려면 먼저 SQLConnection 인스턴스를 만듭니다. open() 메서드를 호출하여 동기 실행 모드에서 이 인스턴스를 열거나 openAsync() 메서드를 호출하여 비동기 실행 모드에서 이 인스턴스를 엽니다. open() 및 openAsync() 메서드는 데이터베이스에 대한 연결을 여는 데 사용됩니다. reference 매개 변수(첫 번째 매개 변수)의 존재하지 않는 파일 위치를 참조하는 File 인스턴스를 전달하는 경우 open() 또는 openAsync() 메서드는 해당 파일 위치에 데이터베이스 파일을 만들고 새로 만든 데이터베이스에 대한 연결을 엽니다.

데이터베이스를 만들 때 호출한 메서드가 open()이든 openAsync()이든 간에 데이터베이스 파일의 이름은 임의의 파일 확장명이 있는 임의의 유효한 파일 이름일 수 있습니다. reference 매개 변수에 null을 사용하여 open() 또는 openAsync() 메서드를 호출하면 디스크에 데이터베이스 파일이 만들어지지 않고 새로운 메모리 내 데이터베이스가 만들어집니다.

다음 코드 샘플에서는 비동기 실행 모드를 사용하여 데이터베이스 파일(새 데이터베이스)을 만드는 과정을 보여 줍니다. 이 경우 데이터베이스 파일이 “DBSample.db”라는 파일 이름으로 613페이지의 “응용 프로그램 저장소 디렉토리 가리키기”에 저장됩니다.

```
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>
```

참고: File 클래스를 사용하면 특정한 기본 파일 경로를 가리킬 수 있지만, 이로 인해 응용 프로그램이 플랫폼 간에 작동하지 않을 수 있습니다. 예를 들어 경로 C:\Documents and Settings\joe\test.db는 Windows에서만 작동합니다. 따라서 앞의 예제에서처럼 File.applicationStorageDirectory 및 resolvePath() 메서드 같은 File 클래스의 정적 속성을 사용하는 것이 가장 좋습니다. 자세한 내용은 609페이지의 "File 객체의 경로"를 참조하십시오.

작업을 동기적으로 실행하려면 SQLConnection 인스턴스를 사용하여 데이터베이스 연결을 열 때 open() 메서드를 호출합니다. 다음 예제에서는 작업을 동기적으로 실행하는 SQLConnection 인스턴스를 만들고 여는 방법을 보여 줍니다.

```
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>
```

데이터베이스 테이블 생성

Adobe AIR 1.0 이상

데이터베이스에서 테이블을 만들려면 SELECT, INSERT 등의 SQL 문을 실행하는 데 사용하는 동일한 과정을 사용하여 해당 데이터베이스에 대해 SQL 문을 실행해야 합니다. 이때 사용하는 문은 새 테이블의 열과 제약 조건에 대한 정의가 포함된 CREATE TABLE입니다. SQL 문 실행에 대한 자세한 내용은 664페이지의 “SQL 문 작업”을 참조하십시오.

다음 예제에서는 비동기 실행 모드를 사용하여 기존 데이터베이스 파일에 "employees"라는 테이블을 만드는 방법을 보여 줍니다. 이 코드에서는 이미 인스턴스화되어 데이터베이스에 연결되어 있는 conn이라는 SQLConnection 인스턴스가 있다고 가정합니다.

```
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>
```

다음 예제에서는 동기 실행 모드를 사용하여 기존 데이터베이스 파일에서 “employees”라는 테이블을 만드는 방법을 보여 줍니다. 이 코드에서는 이미 인스턴스화되어 데이터베이스에 연결되어 있는 conn이라는 SQLConnection 인스턴스가 있다고 가정합니다.

```
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>
```

SQL 데이터베이스 데이터 조작

Adobe AIR 1.0 이상

로컬 SQL 데이터베이스로 작업할 때 수행하는 일반적인 작업이 있습니다. 이러한 작업에는 데이터베이스에 연결하고 데이터베이스의 테이블에서 데이터를 추가 및 검색하는 작업 등이 있습니다. 또한 이러한 작업을 수행할 때는 데이터 유형 작업 및 오류 처리와 같은 몇 가지 문제를 명심해야 합니다.

자주 처리하지는 않지만 일반적인 작업을 수행하려면 흔히 수행해야 하는 몇 가지 데이터베이스 작업도 있습니다. 예를 들어, 데이터베이스에 연결하고 테이블에서 데이터를 검색하려면 데이터베이스를 만들고 데이터베이스에 테이블 구조를 만들어야 합니다. 이러한 자주 수행하지 않는 초기 설정 작업은 655페이지의 “데이터베이스 생성 및 수정”에서 설명합니다.

데이터베이스 작업을 비동기적으로 수행하도록 선택할 수 있습니다. 이 경우 데이터베이스 엔진이 백그라운드에서 실행되고 작업이 성공하거나 실패할 때 이벤트를 전달하여 사용자에게 알립니다. 이러한 작업을 동기적으로 수행할 수도 있습니다. 이 경우 데이터베이스 작업이 차례로 수행되고 전체 응용 프로그램(화면 업데이트 포함)이 작업이 완료될 때까지 기다린 후 다른 코드를 실행합니다. 비동기 실행 모드나 동기 실행 모드에서 작업하는 방법에 대한 자세한 내용은 685페이지의 “동기 및 비동기 데이터베이스 작업 사용”을 참조하십시오.

데이터베이스 연결

Adobe AIR 1.0 이상

데이터베이스 작업을 수행하려면 먼저 데이터베이스 파일에 대한 연결을 엽니다. SQLConnection 인스턴스는 하나 이상의 데이터베이스에 대한 연결을 나타내는 데 사용됩니다. SQLConnection 인스턴스를 사용하여 연결된 첫 번째 데이터베이스를 "기본" 데이터베이스라고 합니다. 이 데이터베이스는 동기 실행 모드의 경우 open() 메서드를 사용하여 연결되고 비동기 실행 모드의 경우에는 openAsync() 메서드를 사용하여 연결됩니다.

비동기 openAsync() 작업을 사용하여 데이터베이스를 여는 경우 openAsync() 작업이 완료된 때를 알기 위해 SQLConnection 인스턴스의 open 이벤트에 대해 등록하고 작업이 실패한 경우를 확인하기 위해 SQLConnection 인스턴스의 error 이벤트에 대해 등록합니다.

다음 예제에서는 비동기 실행을 위해 기존 데이터베이스 파일을 여는 방법을 보여 줍니다. 613페이지의 "응용 프로그램 저장소 디렉토리 가리키기"라는 데이터베이스 파일이 사용자의 응용 프로그램 저장소 디렉토리에 있습니다.

```
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>
```

다음 예제에서는 동기 실행을 위해 기존 데이터베이스 파일을 여는 방법을 보여 줍니다. 613페이지의 “응용 프로그램 저장소 디렉토리 가리키기”라는 데이터베이스 파일이 사용자의 응용 프로그램 저장소 디렉토리에 있습니다.

```
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>
```

비동기 예제의 openAsync() 메서드 호출과 동기 예제의 open() 메서드 호출에서 두 번째 인수는 상수 SQLMode.UPDATE입니다. 두 번째 매개 변수(openMode)에 SQLMode.UPDATE를 지정한 경우 지정된 파일이 없으면 런타임에서 오류를 전달합니다. openMode 매개 변수에 SQLMode.CREATE를 전달하거나 openMode 매개 변수를 해제하는 경우 지정된 파일이 없으면 런타임에서 데이터베이스 파일을 만들려고 합니다. 그러나 파일이 있으면 파일이 열립니다. 이것은 SQLMode.Update를 사용할 때와 동일합니다. openMode 매개 변수에 SQLMode.READ를 지정하여 읽기 전용 모드로 기존 데이터베이스를 열 수도 있습니다. 이 경우 데이터베이스에서 데이터를 검색할 수 있지만 데이터를 추가, 삭제 또는 변경할 수 없습니다.

SQL 문 작업

Adobe AIR 1.0 이상

개별 SQL 문(쿼리 또는 명령)은 런타임에서 SQLStatement 객체로 표현됩니다. SQL 문을 만들고 실행하려면 다음 단계를 수행하십시오.

SQLStatement 인스턴스를 만듭니다.

SQLStatement 객체는 응용 프로그램에서 SQL 문을 나타냅니다.

```
var selectData:SQLStatement = new SQLStatement();
```

쿼리를 실행할 대상 데이터베이스를 지정합니다.

이렇게 하려면 SQLStatement 객체의 sqlConnection 속성을 원하는 데이터베이스와 연결된 SQLConnection 인스턴스로 설정합니다.

```
// A SQLConnection named "conn" has been created previously
selectData.sqlConnection = conn;
```

실제 SQL 문을 지정합니다.
문 텍스트를 String으로 만들어 SQLStatement 인스턴스의 text 속성에 할당합니다.

```
selectData.text = "SELECT col1, col2 FROM my_table WHERE col1 = :param1";
```

실행 작업의 결과를 처리하는 함수를 정의합니다(비동기 실행 모드에만 해당).
addEventListener() 메서드를 사용하여 SQLStatement 인스턴스의 result 및 error 이벤트에 대한 리스너로 함수를 등록합니다.

```
// 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
}
```

또는 Responder 객체를 사용하여 리스너 메서드를 지정할 수 있습니다. 이 경우 Responder 인스턴스를 만들고 리스너 메서드를 이 인스턴스에 연결합니다.

```
// 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
}
```

문 텍스트에 매개 변수 정의가 포함되어 있으면 해당 매개 변수의 값을 할당합니다.
매개 변수 값을 할당하려면 SQLStatement 인스턴스의 parameters 연관 배열 속성을 사용합니다.

```
selectData.parameters[":param1"] = 25;
```

SQL 문을 실행합니다.
SQLStatement 인스턴스의 execute() 메서드를 호출합니다.

```
// using synchronous execution mode
// or listener methods in asynchronous execution mode
selectData.execute();
```

비동기 실행 모드에서 이벤트 리스너 대신 Responder를 사용하는 경우에는 Responder 인스턴스를 execute() 메서드에 전달합니다.

```
// using a Responder in asynchronous execution mode
selectData.execute(-1, selectResponder);
```

이러한 단계를 보여 주는 특정 예제는 다음 항목을 참조하십시오.

668페이지의 “데이터베이스에서 데이터 검색”

677페이지의 “데이터 삽입”

682페이지의 “데이터 변경 또는 삭제”

문에서 매개 변수 사용

Adobe AIR 1.0 이상

SQL 문 매개 변수를 사용하면 다시 사용할 수 있는 SQL 문을 만들 수 있습니다. 문 매개 변수를 사용하면 INSERT 문에 추가되는 값과 같이 문에 있는 값이 변경될 수 있지만 기본 문 텍스트는 변경되지 않고 유지됩니다. 이에 따라 성능이 향상되고 응용 프로그램을 코딩하기가 쉬워집니다.

문 매개 변수 이해

Adobe AIR 1.0 이상

응용 프로그램에서 한 SQL 문을 약간 변형하여 여러 번 사용하는 경우가 많습니다. 예를 들어, 사용자가 새 재고 항목을 데이터베이스에 추가할 수 있는 재고 추적 응용 프로그램의 경우 재고 항목을 데이터베이스에 추가하는 응용 프로그램 코드에서는 데이터를 데이터베이스에 실제로 추가하는 SQL INSERT 문을 실행합니다. 그러나 이 문이 실행될 때마다 약간 변형되어 실행됩니다. 특히 테이블에 삽입되는 실제 값은 추가되는 재고 항목과 관련되기 때문에 서로 다릅니다.

서로 다른 값을 사용하여 SQL 문을 여러 번 실행하는 경우 SQL 텍스트에 리터럴 값 대신 매개 변수가 포함된 SQL 문을 사용하는 것이 가장 좋습니다. 매개 변수는 문이 실행될 때마다 실제 값과 바뀌는 문 텍스트의 자리 표시자입니다. SQL 문에서 매개 변수를 사용하려면 일반적인 경우처럼 SQLStatement 인스턴스를 만듭니다. text 속성에 할당된 실제 SQL 문의 경우 리터럴 값 대신 매개 변수 자리 표시자를 사용합니다. 그런 다음 SQLStatement 인스턴스의 parameters 속성에서 요소의 값을 설정하여 각 매개 변수의 값을 정의합니다. parameters 속성은 연관 배열이므로 다음 구문을 사용하여 특정 값을 설정합니다.

```
statement.parameters[parameter_identifier] = value;
```

parameter_identifier는 명명된 매개 변수를 사용하는 경우 문자열이고 명명되지 않은 매개 변수를 사용하는 경우에는 정수 인덱스입니다.

명명된 매개 변수 사용

Adobe AIR 1.0 이상

매개 변수가 명명된 매개 변수일 수 있습니다. 명명된 매개 변수에는 데이터베이스에서 매개 변수 값을 문 텍스트의 해당 자리 표시자 위치와 일치시키는 데 사용하는 특정 이름이 있습니다. 매개 변수 이름은 다음 예와 같이 “:” 또는 “@” 문자와 그 뒤의 이름으로 구성되어 있습니다.

```
:itemName
@firstName
```

다음 코드 샘플에서는 명명된 매개 변수를 사용하는 방법을 보여 줍니다.

```
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();
```

명명되지 않은 매개 변수 사용

Adobe AIR 1.0 이상

명명된 매개 변수를 사용하는 대신 명명되지 않은 매개 변수를 사용할 수도 있습니다. 명명되지 않은 매개 변수를 사용하려면 "?" 문자를 사용하여 SQL 문에 매개 변수를 나타냅니다. 문의 매개 변수 순서에 따라 첫 번째 매개 변수의 인덱스 0에서 시작하여 각 매개 변수에 숫자 인덱스가 할당됩니다. 다음 예제는 이전 예제에서 명명되지 않은 매개 변수를 사용한 것입니다.

```
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();
```

매개 변수 사용의 장점

Adobe AIR 1.0 이상

SQL 문에서 매개 변수를 사용하는 경우 몇 가지 장점이 있습니다.

성능 향상 매개 변수를 사용하는 SQLStatement 인스턴스는 실행될 때마다 SQL 텍스트를 동적으로 만드는 SQLStatement 인스턴스보다 효율적으로 실행될 수 있습니다. 문이 한 번 준비된 후 SQL 문을 다시 컴파일할 필요 없이 서로 다른 매개 변수 값을 사용하여 여러 번 실행될 수 있기 때문에 성능이 향상됩니다.

명시적 데이터 유형 지정 매개 변수를 사용하면 SQL 문을 생성할 때는 알 수 없는 값의 유형을 지정하여 대체할 수 있습니다. 데이터베이스에 전달되는 값에 대한 저장소 클래스를 확보하는 방법은 매개 변수를 사용하는 것 외에는 없습니다. 매개 변수를 사용하지 않으면 런타임에서 모든 값을 관련 열의 유형 선호도에 따라 텍스트 표현에서 저장소 클래스로 변환하려고 합니다.

저장소 클래스와 열 선호도에 대한 자세한 내용은 1024페이지의 "데이터 유형 지원"을 참조하십시오.

보안 강화 매개 변수를 사용하면 SQL 삽입 공격이라는 악의적인 기술을 방지할 수 있습니다. SQL 삽입 공격에서는 사용자가 액세스 가능한 위치(예: 데이터 입력 필드)에 SQL 코드를 입력합니다. 사용자 입력을 SQL 텍스트로 직접 연결하여 응용 프로그램 코드에서 SQL 문을 생성하면 사용자가 입력한 SQL 코드가 데이터베이스에 대해 실행됩니다. 다음은 사용자 입력을 SQL 텍스트로 연결하는 예제입니다. **이 방법을 사용하지 마십시오.**

```
// 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;
```

사용자가 입력한 값을 문의 텍스트로 연결하는 대신 문 매개 변수를 사용하면 SQL 삽입 공격을 방지할 수 있습니다. 매개 변수 값은 리터럴 문 텍스트의 일부가 되지 않고 대체된 값으로 명시적으로 처리되므로 SQL 삽입이 발생할 수 없습니다. 다음은 이전 샘플 대신 사용할 수 있는 권장 방법입니다.

```
// 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;
```

데이터베이스에서 데이터 검색

Adobe AIR 1.0 이상

데이터베이스에서 데이터를 검색하는 작업은 두 단계로 이루어집니다. 먼저, 데이터베이스에서 원하는 데이터 집합을 설명하는 SQL SELECT 문을 실행합니다. 다음으로, 검색된 데이터에 액세스하고 응용 프로그램에서 필요한 대로 표시하거나 조작합니다.

SELECT 문 실행

Adobe AIR 1.0 이상

데이터베이스에서 기존 데이터를 검색하려면 SQLStatement 인스턴스를 사용합니다. 적절한 SQL SELECT 문을 인스턴스의 text 속성에 할당한 다음 execute() 메서드를 호출합니다.

SELECT 문의 구문에 대한 자세한 내용은 1005페이지의 “로컬 데이터베이스의 SQL 지원”을 참조하십시오.

다음 예제에서는 비동기 실행 모드를 사용하는 경우 SELECT 문을 실행하여 "products"라는 테이블에서 데이터를 검색하는 방법을 보여 줍니다.

```
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>
```

다음 예제에서는 동기 실행 모드를 사용하는 경우 SELECT 문을 실행하여 "products"라는 테이블에서 데이터를 검색하는 방법을 보여 줍니다.

```
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>
```

비동기 실행 모드에서는 이 문의 실행이 완료되면 SQLStatement 인스턴스가 이 문이 성공적으로 실행되었음을 나타내는 result 이벤트(SQLEvent.RESULT)를 전달합니다. 또는 Responder 객체가 execute() 메서드에서 인수로 전달되면 Responder 객체의 결과 핸들러 함수가 호출됩니다. 동기 실행 모드에서는 execute() 작업이 완료될 때까지 실행이 일시 중지된 후 다음 코드 줄에서 계속됩니다.

SELECT 문 결과 데이터 액세스

Adobe AIR 1.0 이상

SELECT 문의 실행이 완료된 후의 다음 단계는 검색된 데이터에 액세스하는 것입니다. 결과 데이터는 SQLStatement 객체의 getResult() 메서드를 호출하여 SELECT 문을 실행해서 검색합니다.

```
var result:SQLResult = selectStatement.getResult();
```

getResult() 메서드는 SQLResult 객체를 반환합니다. SQLResult 객체의 data 속성은 SELECT 문의 결과를 포함하는 배열입니다.

```
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];
}
```

SELECT 결과의 각 데이터 행은 data 배열에 포함된 객체 인스턴스가 됩니다. 이 객체에는 결과 집합의 열 이름과 일치하는 이름의 속성이 있습니다. 이러한 속성에는 결과 집합의 열에 있는 값이 포함됩니다. 예를 들어, SELECT 문에서 "itemId", "itemName" 및 "price"라는 세 열이 포함된 결과 집합을 지정하는 경우 결과 집합의 행마다 itemId, itemName 및 price라는 속성과 함께 Object 인스턴스가 만들어집니다. 이러한 속성에는 해당하는 열의 값이 포함됩니다.

다음 코드 샘플에서는 SELECT 문이 텍스트로 포함된 SQLStatement 인스턴스를 정의합니다. 이 문은 employees라는 테이블에 있는 모든 행의 firstName 및 lastName 열 값이 포함된 행을 검색합니다. 이 예제에서는 비동기 실행 모드를 사용합니다. 실행이 완료되면 selectResult() 메서드가 호출되고 결과 데이터 행이 SQLStatement.getResult()를 사용하여 액세스되고 trace() 메서드를 사용하여 표시됩니다. 이 샘플에서는 이미 인스턴스화되고 데이터베이스에 연결된 conn이라는 SQLConnection 인스턴스가 있다고 가정합니다. 또한 "employees" 테이블이 이미 만들어져서 데이터로 채워져 있다고 가정합니다.

```
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>
```

다음 코드 샘플에서는 이전 코드와 동일한 방법을 보여 주지만 동기 실행 모드를 사용합니다. 이 예제에서는 SELECT 문이 텍스트로 포함된 SQLStatement 인수를 정의합니다. 이 문은 employees라는 테이블에 있는 모든 행의 firstName 및 lastName 열 값이 포함된 행을 검색합니다. 결과 데이터 행은 SQLStatement.getResult()를 사용하여 액세스되고 trace() 메서드를 사용하여 표시됩니다. 이 샘플에서는 이미 인스턴스화되고 데이터베이스에 연결된 conn이라는 SQLConnection 인스턴스가 있다고 가정합니다. 또한 "employees" 테이블이 이미 만들어져서 데이터로 채워져 있다고 가정합니다.

```
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>
```

SELECT 결과 데이터의 데이터 유형 정의

Adobe AIR 1.0 이상

기본적으로 SELECT 문에서 반환되는 각 행은 결과 집합의 열 이름과 동일하게 명명된 속성과 각 열의 값이 연결된 속성의 값으로 포함된 Object 인스턴스로 만들어집니다. 그러나 SQL SELECT 문을 실행하기 전에 SQLStatement 인스턴스의 itemClass 속성을 클래스로 설정할 수 있습니다. SELECT 문에서 반환되는 각 행은 itemClass 속성을 설정하여 지정된 클래스의 인스턴스로 만들어집니다. 런타임은 SELECT 결과 집합에 있는 열 이름과 itemClass 클래스에 있는 속성의 이름을 일치시켜 결과 열 값을 속성 값에 할당합니다.

itemClass 속성 값으로 할당된 모든 클래스에는 매개 변수가 필요하지 않은 생성자가 있어야 합니다. 또한 이러한 클래스에는 SELECT 문에서 반환되는 각 열에 대해 하나의 속성이 있어야 합니다. SELECT 목록의 열에 itemClass 클래스의 속성 이름과 일치하는 이름이 없는 경우 오류로 간주됩니다.

SELECT 결과 중 일부 검색

Adobe AIR 1.0 이상

기본적으로 SELECT 문을 실행하면 결과 집합의 모든 행이 한 번에 검색됩니다. 이 문이 완료되면 대개 객체를 만들거나 화면에 데이터를 표시하는 등의 방법으로 검색된 데이터를 처리합니다. 이 문에서 많은 행이 반환되는 경우 모든 데이터를 한 번에 처리하면 컴퓨터 리소스가 많이 사용될 수 있으며 이렇게 되면 사용자 인터페이스가 다시 그려지지 않습니다.

특정 수의 결과 행을 한 번에 반환하도록 런타임에 지시하여 응용 프로그램의 성능을 눈에 띄게 향상시킬 수 있습니다. 이렇게 하면 초기 결과 데이터가 더 빨리 반환됩니다. 또한 결과 행을 집합으로 분할할 수 있으므로 사용자 인터페이스가 각 행 집합이 처리된 후 업데이트됩니다. 이 방법은 비동기 실행 모드에서 사용할 때만 효과적입니다.

SELECT 결과 중 일부를 검색하려면 SQLStatement.execute() 메서드의 첫 번째 매개 변수(prefetch 매개 변수)에 대한 값을 지정합니다. prefetch 매개 변수는 이 문이 처음 실행될 때 검색할 행 수를 나타냅니다. SQLStatement 인스턴스의 execute() 메서드를 호출할 때 prefetch 매개 변수를 지정하면 해당하는 수의 행만 검색됩니다.

```
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
```

이 문은 result 이벤트를 전달하여 결과 행의 첫 번째 집합을 사용할 수 있음을 나타냅니다. 생성되는 SQLResult 인스턴스의 data 속성에는 데이터 행이 포함되어 있으며 complete 속서은 검색할 추가 결과 행이 있는지 여부를 나타냅니다. 추가 결과 행을 검색하려면 SQLStatement 인스턴스의 next() 메서드를 호출합니다. execute() 메서드와 마찬가지로 next() 메서드의 첫 번째 매개 변수는 다음에 결과 이벤트가 전달될 때 검색할 행 수를 나타내는 데 사용됩니다.

```
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);
        }
    }
}
```

next() 메서드가 결과 행의 이후 집합을 반환할 때마다 SQLStatement는 result 이벤트를 전달합니다. 따라서 동일한 리스너 함수를 사용하여 모든 행이 검색될 때까지 next() 호출을 통해 결과를 계속 처리할 수 있습니다.

자세한 내용은 SQLStatement.execute() 메서드(prefetch 매개 변수 설명) 및 SQLStatement.next() 메서드에 대한 설명서의 설명을 참조하십시오.

데이터 삽입

Adobe AIR 1.0 이상

데이터베이스에 데이터를 추가하려면 SQL INSERT 문을 실행해야 합니다. 이 문의 실행이 완료되면 새로 삽입된 행의 기본 키가 데이터베이스에서 생성된 경우 해당 키에 액세스할 수 있습니다.

INSERT 문 실행

Adobe AIR 1.0 이상

데이터베이스의 테이블에 데이터를 추가하려면 SQL INSERT 문이 텍스트로 포함된 SQLStatement 인스턴스를 만들고 실행합니다.

다음 예제에서는 SQLStatement 인스턴스를 사용하여 데이터 행을 기존 employees 테이블에 추가하며, 비동기 실행 모드를 사용하여 데이터를 삽입하는 방법을 보여 줍니다. 이 예제에서는 이미 인스턴스화되고 데이터베이스에 연결된 conn이라는 SQLConnection 인스턴스가 있다고 가정합니다. 또한 “employees” 테이블이 이미 만들어졌다고 가정합니다.

```
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>
```

다음 예제에서는 동기 실행 모드를 사용하여 데이터 행을 기존 employees 테이블에 추가합니다. 이 예제에서는 이미 인스턴스화되고 데이터베이스에 연결된 conn이라는 SQLConnection 인스턴스가 있다고 가정합니다. 또한 “employees” 테이블이 이미 만들어졌다고 가정합니다.

```
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>
```

삽입된 행에 대해 데이터베이스에서 생성된 기본 키 검색

Adobe AIR 1.0 이상

데이터 행을 테이블에 삽입한 후 코드에서는 새로 삽입된 행에 대해 데이터베이스에서 생성된 기본 키나 행 식별자 값을 알고 있어야 하는 경우가 많습니다. 예를 들어, 한 테이블에 행을 삽입하고 관련된 테이블에 행을 추가하려고 할 수 있습니다. 이 경우 관련된 테이블에서 기본 키 값을 외래 키로 삽입할 수 있습니다. 새로 삽입된 행의 기본 키는 문 실행으로 연관된 SQLResult 객체를 사용하여 검색할 수 있습니다. 이것은 SELECT 문이 실행된 후 결과 데이터에 액세스하는 데 사용되는 객체와 동일합니다. 다른 SQL 문의 경우와 마찬가지로 INSERT 문의 실행이 완료되면 런타임은 SQLResult 인스턴스를 만듭니다. 이벤트 리스너를 사용하거나 동기 실행 모드를 사용하는 경우 SQLStatement 객체의 getResult() 메서드를 호출하여 SQLResult 인스턴스에 액세스합니다. 또는 비동기 실행 모드를 사용하고 Responder 인스턴스를 execute() 호출에 전달하는 경우 SQLResult 인스턴스가 결과 핸들러 함수에 인수로 전달됩니다. 어떤 경우에서든 실행된 SQL 문이 INSERT 문인 경우 가장 최근에 삽입된 행의 행 식별자가 포함된 lastInsertRowID 속성이 SQLResult 인스턴스에 있습니다.

다음 예제에서는 비동기 실행 모드에서 삽입된 행의 기본 키에 액세스하는 방법을 보여 줍니다.

```
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
}
```

다음 예제에서는 동기 실행 모드에서 삽입된 행의 기본 키에 액세스하는 방법을 보여 줍니다.

```
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
}
```

행 식별자는 다음 규칙에 따라 테이블 정의에서 기본 키 열로 지정된 열의 값일 수도 있고 아닐 수도 있습니다.

- 선호도(열 데이터 유형)가 INTEGER인 기본 키 열을 사용하여 테이블이 정의된 경우 lastInsertRowID 속성에는 해당 행에 삽입된 값(또는 AUTOINCREMENT 열인 경우 런타임에서 생성된 값)이 포함되어 있습니다.
- 선호도가 INTEGER가 아닌 단일 기본 키 열이나 여러 기본 키 열(복합 키)을 사용하여 테이블이 정의된 경우 내부적으로 데이터베이스에서 행에 대한 정수 행 식별자 값을 생성합니다. 생성된 값은 lastInsertRowID 속성의 값입니다.
- 값은 항상 가장 최근에 삽입된 행의 행 식별자입니다. INSERT 문이 행을 삽입하는 트리거를 발생시키는 경우 lastInsertRowID 속성에는 INSERT 문으로 만들어진 행 대신 트리거로 삽입된 마지막 행의 행 식별자가 포함됩니다.

이러한 규칙에 따라 INSERT 명령 후 SQLResult.lastInsertRowID 속성을 통해 값을 사용할 수 있는 명시적으로 정의된 기본 키 열을 원하는 경우 INTEGER PRIMARY KEY 열로 열을 정의해야 합니다. 테이블에 명시적 INTEGER PRIMARY KEY 열이 포함되지 않은 경우에도 관련된 테이블과의 관계를 정의하는 측면에서는 데이터베이스에서 생성된 행 식별자를 테이블의 기본 키로 동일하게 사용할 수 있습니다. 행 식별자 열 값은 특수 열 이름 ROWID, _ROWID_ 또는 OID 중 하나를 사용하여 모든 SQL 문에서 사용할 수 있습니다. 명시적으로 선언된 INTEGER PRIMARY KEY 열을 사용하는 경우와 마찬가지로 관련된 테이블에서 외래 키 열을 만들고 행 식별자 값을 외래 키 열 값으로 사용할 수 있습니다. 이 점에서 자연 키 대신 임의의 기본 키를 사용하는 경우 런타임에서 기본 키 값을 생성하는 것을 꺼리지 않는 한 두 테이블 간의 외래 키 관계를 정의하기 위한 테이블의 기본 키로 INTEGER PRIMARY KEY 열을 사용하는 경우와 시스템에서 생성된 행 식별자를 사용하는 경우 사이에는 거의 차이가 없습니다.

기본 키와 생성된 행 식별자에 대한 자세한 내용은 1005페이지의 “로컬 데이터베이스의 SQL 지원”을 참조하십시오.

데이터 변경 또는 삭제

Adobe AIR 1.0 이상

다른 데이터 조작 작업의 실행 과정은 664페이지의 "SQL 문 작업"에 설명된 대로 SQL SELECT 또는 INSERT 문을 실행하는 데 사용되는 과정과 동일합니다. SQLStatement 인스턴스의 text 속성에 다른 SQL 문을 대체하기만 하면 됩니다.

- 테이블에서 기존 데이터를 변경하려면 UPDATE 문을 사용합니다,
- 테이블에서 하나 이상의 데이터 행을 삭제하려면 DELETE 문을 사용합니다.

이러한 명령문에 대한 설명은 1005페이지의 "로컬 데이터베이스의 SQL 지원"을 참조하십시오.

여러 데이터베이스 작업

Adobe AIR 1.0 이상

열려 있는 데이터베이스가 이미 있는 SQLConnection 인스턴스에서 추가 데이터베이스에 대한 연결을 열려면 SQLConnection.attach() 메서드를 사용합니다. attach() 메서드 호출에서 name 매개 변수를 사용하여 연결된 데이터베이스에 이름을 지정합니다. 이 데이터베이스를 조작하기 위해 문을 작성하는 경우 해당 이름을 접두어로 사용하여(database-name.table-name 형식 사용) SQL 문에서 테이블 이름을 정규화함으로써 명명된 데이터베이스에서 해당 테이블을 찾을 수 있음을 런타임에 나타내야 합니다.

동일한 SQLConnection 인스턴스에 연결된 여러 데이터베이스의 테이블이 포함된 하나의 SQL 문을 실행할 수 있습니다. 트랜잭션이 SQLConnection 인스턴스에서 만들어지면 SQLConnection 인스턴스를 사용하여 실행되는 모든 SQL 문에 적용됩니다. 이는 SQL 문이 실행되는 연결된 데이터베이스에 관계없이 해당하는 사실입니다.

또는 응용 프로그램에서 여러 SQLConnection 인스턴스를 만들 수도 있습니다. 각 SQLConnection 인스턴스는 하나 이상의 데이터베이스에 연결되어 있습니다. 그러나 동일한 데이터베이스에 대한 여러 연결을 사용하는 경우 데이터베이스 트랜잭션이 여러 SQLConnection 인스턴스에서 공유되지 않는 점을 명심해야 합니다. 따라서 여러 SQLConnection 인스턴스를 사용하여 동일한 데이터베이스 파일에 연결하는 경우 두 연결의 데이터 변경 사항이 예상되는 방식으로 적용되지 않을 수도 있습니다. 예를 들어, 두 UPDATE 또는 DELETE 문이 서로 다른 SQLConnection 인스턴스를 통해 동일한 데이터베이스에 대해 실행되는 경우 한 작업이 수행된 후 응용 프로그램 오류가 발생하면 데이터베이스 데이터가 되돌릴 수 없고 데이터베이스와 응용 프로그램의 무결성에 영향을 미칠 수도 있는 중간 상태로 남아 있을 수 있습니다.

데이터베이스 오류 처리

Adobe AIR 1.0 이상

일반적으로 데이터베이스 오류 처리는 다른 런타임 오류 처리와 유사합니다. 발생할 수 있는 오류에 대비하고 런타임에 오류 처리를 맡기는 대신 오류에 응답하는 코드를 작성해야 합니다. 일반적으로 가능한 데이터베이스 오류는 연결 오류, SQL 구문 오류 및 제약 조건 오류라는 세 범주로 나눌 수 있습니다.

연결 오류

Adobe AIR 1.0 이상

대부분의 데이터베이스 오류는 연결 오류이며 어떠한 작업 중에도 발생할 수 있습니다. 연결 오류를 방지하기 위한 전략이 있지만 데이터베이스가 응용 프로그램의 필수적 부분인 경우 연결 오류에서 적절하게 복구하는 간단한 방법은 거의 없습니다.

대부분의 연결 오류는 런타임에서 운영 체제, 파일 시스템 및 데이터베이스 파일과 상호 작용하는 방식과 관련이 있습니다. 예를 들어, 사용자가 파일 시스템의 특정 위치에서 데이터베이스를 만들 수 있는 권한이 없으면 연결 오류가 발생합니다. 다음 전략은 연결 오류를 방지하는 데 도움이 됩니다.

사용자별 데이터베이스 파일 사용 한 컴퓨터에서 응용 프로그램을 사용하는 모든 사용자에 대해 하나의 데이터베이스 파일을 사용하지 말고 각 사용자에게 개별 데이터베이스 파일을 제공합니다. 데이터베이스 파일은 사용자의 계정과 연결된 디렉토리에 있어야 합니다. 예를 들어, 데이터베이스 파일은 응용 프로그램의 저장소 디렉토리, 사용자의 문서 폴더, 사용자의 바탕 화면 등에 있을 수 있습니다.

여러 사용자 유형 고려 서로 다른 운영 체제에서 여러 유형의 사용자 계정을 사용하여 응용 프로그램을 테스트합니다. 사용자가 컴퓨터에서 관리자 권한이 있다고 가정하지 마십시오. 또한 응용 프로그램을 설치한 개인이 응용 프로그램을 실행하고 있는 사용자라고 가정하지도 마십시오.

다양한 파일 위치 고려 사용자가 데이터베이스 파일을 저장할 위치를 지정하거나 열 파일을 선택할 수 있게 하려면 사용자가 사용할 수 있는 가능한 파일 위치를 고려합니다. 또한 사용자가 데이터베이스 파일을 저장할 수 있거나 열 수 있는 위치에 대한 제한을 정의하는 것을 고려합니다. 예를 들어, 사용자가 사용자 계정의 저장소 위치에 있는 파일만 열 수 있게 할 수도 있습니다.

연결 오류는 처음으로 데이터베이스를 만들거나 열려고 할 때 발생할 가능성이 가장 큽니다. 따라서 사용자가 응용 프로그램에서 데이터베이스 관련 작업을 수행할 수 없습니다. 읽기 전용 또는 권한 오류와 같은 특정 유형의 오류에 대한 한 가지 가능한 복구 방법은 데이터베이스 파일을 다른 위치에 복사하는 것입니다. 응용 프로그램에서는 사용자가 파일을 만들고 파일에 쓸 수 있는 권한이 있는 다른 위치에 데이터베이스 파일을 복사하고 해당 위치를 대신 사용할 수 있습니다.

구문 오류

Adobe AIR 1.0 이상

구문 오류는 SQL 문의 형식이 잘못되고 응용 프로그램에서 SQL 문을 실행하려고 할 때 발생합니다. 로컬 데이터베이스 SQL 문이 문자열로 만들어지기 때문에 컴파일 타임 SQL 구문 검사를 수행할 수 없습니다. 모든 SQL 문은 구문을 검사하기 위해 실행되어야 합니다. SQL 구문 오류를 방지하려면 다음 전략을 사용하십시오.

모든 SQL 문을 완전히 테스트합니다. 가능한 경우 응용 프로그램을 개발하는 동안 SQL 문을 응용 프로그램 코드에서 문 텍스트로 인코딩하기 전에 별도로 테스트합니다. 또한 단위 테스트와 같은 코드 테스트 방법을 사용하여 코드에서 가능한 모든 옵션과 변형을 실행하는 테스트 집합을 만듭니다.

문 매개 변수를 사용하고 SQL 연결(동적 생성)을 피합니다. 매개 변수를 사용하고 동적으로 만들어진 SQL 문을 방지하는 것은 동일한 SQL 문 텍스트가 문이 실행될 때마다 사용된다는 의미입니다. 따라서 문을 테스트하고 가능한 변형을 제한하기가 훨씬 쉽습니다. SQL 문을 동적으로 생성해야 하는 경우 문의 동적 부분을 최소로 유지합니다. 또한 사용자 입력의 유효성을 신중하게 검사하여 구문 오류를 발생시키지 않게 합니다.

구문 오류에서 복구하려면 SQL 문을 검사하고 구문을 수정할 수 있는 복잡한 논리가 응용 프로그램에 필요합니다. 구문 오류 방지를 위한 위의 지침을 따르면 코드에서 SQL 구문 오류의 가능한 런타임 원인(예: 문에서 사용되는 사용자 입력)을 식별할 수 있습니다. 구문 오류에서 복구하려면 사용자에게 지침을 제공하고 문이 제대로 실행되게 하기 위해 수정할 것을 나타냅니다.

제약 조건 오류

Adobe AIR 1.0 이상

제약 조건 오류는 INSERT 또는 UPDATE 문에서 데이터를 열에 추가하려고 할 때 발생합니다. 새 데이터가 테이블이나 열에 대해 정의된 제약 조건 중 하나를 위반하면 이 오류가 발생합니다. 가능한 제약 조건의 집합은 다음과 같습니다.

UNIQUE 제약 조건 테이블의 모든 행에서 한 열에 중복 값이 있을 수 없도록 지정합니다. 또한 UNIQUE 제약 조건에서 여러 열이 결합된 경우 해당 열의 값 조합은 중복되지 않아야 합니다. 즉, 지정된 고유 열 측면에서 각 행은 고유해야 합니다.

PRIMARY KEY 제약 조건 제약 조건이 허용하고 허용하지 않는 데이터의 측면에서 PRIMARY KEY 제약 조건은 UNIQUE 제약 조건과 동일합니다.

NOT NULL 제약 조건 한 열에 NULL 값이 저장될 수 없고 이에 따라 모든 행에서 해당 열에는 값이 있어야 하도록 지정합니다.

CHECK 제약 조건 하나 이상의 테이블에 대한 임의의 제약 조건을 지정할 수 있습니다. 일반적인 CHECK 제약 조건은 열의 값이 특정 범위 안에 들어야 하도록(예를 들어, 숫자 열의 값이 0보다 커야 하도록) 정의하는 규칙입니다. CHECK 제약 조건의 또 다른 일반적 유형은 열 값 간의 관계를 지정합니다. 예를 들어, 열의 값이 동일한 행에 있는 다른 열의 값과 달라야 하도록 지정할 수 있습니다.

DATA TYPE(열 선호도) 제약 조건 런타임은 열 값의 데이터 유형을 적용하며 잘못된 유형의 값을 열에 저장하려고 하면 오류가 발생합니다. 그러나 많은 조건에서 값이 열의 선언된 데이터 유형과 일치하도록 변환됩니다. 자세한 내용은 685페이지의 "데이터베이스 데이터 유형 작업"을 참조하십시오.

런타임은 외래 키 값에 제약 조건을 적용하지 않습니다. 즉, 외래 키 값이 기존 기본 키 값과 일치할 필요가 없습니다.

런타임 SQL 엔진은 미리 정의된 제약 조건 유형 외에도 트리거 사용을 지원합니다. 트리거는 이벤트 핸들러와 유사하며 특정 작업이 발생할 때 수행되는 미리 정의된 명령의 집합입니다. 예를 들어, 데이터가 특정 테이블에서 삽입되거나 삭제될 때 실행되는 트리거를 정의할 수 있습니다. 데이터 변경 사항을 검사하고 지정된 조건이 충족되지 않으면 오류가 발생하게 하려는 경우 등에 트리거를 사용할 수 있습니다. 따라서 트리거는 제약 조건과 동일한 용도로 사용할 수 있으며 제약 조건 오류를 방지하고 제약 조건 오류에서 복구하기 위한 전략이 트리거에서 생성된 오류에도 적용됩니다. 그러나 트리거에서 생성된 오류의 ID는 제약 조건 오류의 ID와 다릅니다.

특정 테이블에 적용되는 제약 조건의 집합은 응용 프로그램을 설계하는 동안 결정됩니다. 제약 조건을 의식적으로 설계하면 제약 조건 오류를 방지하고 제약 조건 오류에서 복구하도록 응용 프로그램을 설계하기가 쉬워집니다. 그러나 제약 조건 오류는 체계적으로 예측하고 방지하기가 어렵습니다. 제약 조건 오류는 응용 프로그램 데이터가 추가될 때까지 나타나지 않기 때문에 예측이 어렵습니다. 제약 조건 오류는 데이터가 만들어진 후 데이터베이스에 추가될 때 발생합니다. 이러한 오류는 새 데이터와 데이터베이스에 이미 있는 데이터 간 관계의 결과인 경우가 많습니다. 다음 전략은 많은 제약 조건 오류를 방지하는 데 도움이 될 수 있습니다.

신중하게 데이터베이스 구조 및 제약 조건 계획 수립 제약 조건의 목적은 응용 프로그램 규칙을 적용하고 데이터베이스 데이터의 무결성을 보호하는 것입니다. 응용 프로그램에 대한 계획을 수립할 때 응용 프로그램을 지원할 데이터베이스의 구조를 고려합니다. 이 과정에서 특정 값이 필요한지 여부, 기본값이 있는지 여부, 중복 값이 허용되는지 여부 등의 데이터에 대한 규칙을 확인합니다. 이러한 규칙은 데이터베이스 제약 조건을 정의하는 지침이 됩니다.

명시적으로 열 이름 지정 값을 삽입할 열을 명시적으로 지정하지 않고 INSERT 문을 쓸 수 있지만 이렇게 하면 불필요한 위험이 초래됩니다. 값을 삽입할 열의 이름을 명시적으로 지정하면 자동으로 생성되는 값, 기본값이 있는 열 및 NULL 값을 허용하는 열을 허용할 수 있습니다. 또한 이렇게 하면 모든 NOT NULL 열에 명시적 값이 삽입되게 할 수 있습니다.

기본값 사용 열에 대해 NOT NULL 제약 조건을 지정하는 경우 가능하면 항상 열 정의에 기본값을 지정합니다. 응용 프로그램 코드에서도 기본값을 제공할 수 있습니다. 예를 들어, 코드에서 String 변수가 null인지 확인하고 이 변수를 사용하여 문 매개 변수 값을 설정하기 전에 이 변수에 값을 할당할 수 있습니다.

사용자가 입력한 데이터의 유효성 검사 사용자가 정의한 데이터를 미리 검사하여 제약 조건(특히 NOT NULL 및 CHECK 제약 조건의 경우)으로 지정된 제한을 준수하는지 확인합니다. UNIQUE 제약 조건의 경우 SELECT 쿼리를 실행하여 데이터가 고유한지 확인해야 하기 때문에 이러한 확인이 더 어렵습니다.

트리거 사용 삽입된 데이터의 유효성을 검사하거나(대체할 수도 있음) 잘못된 데이터를 수정하기 위해 다른 작업을 수행하는 트리거를 작성할 수 있습니다. 이 유효성 검사와 수정을 통해 제약 조건 오류를 방지할 수 있습니다.

다양한 측면에서 제약 조건 오류는 다른 유형의 오류보다 방지하기가 더 어렵습니다. 다행히도 응용 프로그램을 불안정하거나 사용할 수 없게 만들지 않는 방식으로 제약 조건 오류에서 복구하는 몇 가지 전략이 있습니다.

충돌 알고리즘 사용 열에 대한 제약 조건을 정의하는 경우와 INSERT 또는 UPDATE 문을 만드는 경우 충돌 알고리즘을 지정할 수 있습니다. 충돌 알고리즘은 제약 조건 위반이 발생할 때 데이터베이스에서 수행하는 작업을 정의합니다. 데이터베이스 엔진에서 수행할 수 있는 몇 가지 가능한 작업이 있습니다. 데이터베이스 엔진은 한 문이나 전체 트랜잭션을 끝낼 수 있습니다. 또한 오류를 무시할 수 있으며 이전 데이터를 제거하고 코드에서 저장하려는 데이터로 바꿀 수도 있습니다.

자세한 내용은 1005페이지의 "로컬 데이터베이스의 SQL 지원"에서 "ON CONFLICT(충돌 알고리즘)" 단원을 참조하십시오.

수정 피드백 제공 특정 SQL 명령에 영향을 미칠 수 있는 제약 조건의 집합을 미리 확인할 수 있습니다. 따라서 문에서 발생할 수 있는 제약 조건 오류를 예측할 수 있으며, 이 지식을 사용하여 제약 조건 오류에 응답하는 응용 프로그램 논리를 만들 수 있습니다. 예를 들어, 응용 프로그램에 새 제품을 입력하기 위한 데이터 항목이 포함되어 있는 경우 데이터베이스의 제품 이름 열이

UNIQUE 제약 조건으로 정의되면 데이터베이스에서 새 제품 행을 삽입하는 작업으로 인해 제약 조건 오류가 발생할 수 있습니다. 따라서 응용 프로그램이 제약 조건 오류를 예측하도록 설계됩니다. 오류가 발생하면 응용 프로그램에서 지정된 제품 이름이 이미 사용 중임을 알리고 다른 이름을 선택하도록 요구하는 경고를 사용자에게 표시합니다. 사용자가 동일한 이름의 다른 제품에 대한 정보를 볼 수 있게 할 수도 있습니다.

데이터베이스 데이터 유형 작업

Adobe AIR 1.0 이상

테이블이 데이터베이스에서 만들어지면 테이블을 만들기 위한 SQL 문이 테이블의 각 열에 대한 선호도 또는 데이터 유형을 정의합니다. 선호도 선언을 생략할 수 있지만 CREATE TABLE SQL 문에서 열 선호도를 명시적으로 선언하는 것이 좋습니다.

일반적인 규칙으로 INSERT 문을 사용하여 데이터베이스에 저장하는 모든 객체는 SELECT 문을 실행할 때 동일한 데이터 유형의 인스턴스로 반환됩니다. 그러나 검색된 값의 데이터 유형은 값이 저장된 데이터베이스 열의 선호도에 따라 다를 수 있습니다. 값이 열에 저장된 경우 데이터 유형이 열의 선호도와 일치하지 않으면 데이터베이스에서 열의 선호도와 일치하도록 값을 변환하려고 합니다. 예를 들어, 데이터베이스 열이 NUMERIC 선호도를 사용하여 선언된 경우 데이터베이스에서 데이터를 저장하기 전에 삽입된 데이터를 숫자 저장소 클래스(INTEGER 또는 REAL)로 변환하려고 합니다. 데이터를 변환할 수 없으면 데이터베이스에서 오류를 발생시킵니다. 이 규칙에 따라, String “12345”가 NUMERIC 열에 삽입되는 경우 데이터베이스에서 이 문자열을 데이터베이스에 저장하기 전에 정수 값 12345로 자동으로 변환합니다. 이 값이 SELECT 문으로 검색되는 경우 String 인스턴스가 아니라 숫자 데이터 유형(예: Number)의 인스턴스로 반환됩니다.

원하지 않는 데이터 유형 변환을 방지하는 가장 좋은 방법은 다음 두 규칙을 따르는 것입니다. 먼저, 저장할 데이터 유형과 일치하는 선호도로 각 열을 정의합니다. 다음으로, 데이터 유형이 정의된 선호도와 일치하는 값만 삽입합니다. 이러한 규칙을 따르면 두 가지 장점이 있습니다. 데이터를 삽입할 때 데이터가 예기치 않게 변환되지 않습니다(이에 따라 원하는 의미가 손실될 가능성이 제거됨). 또한 데이터를 검색할 때 데이터가 원래 데이터 유형으로 반환됩니다.

사용 가능한 열 선호도 유형과 SQL 문 내의 데이터 유형 사용에 대한 자세한 내용은 1024페이지의 “데이터 유형 지원”을 참조하십시오.

동기 및 비동기 데이터베이스 작업 사용

Adobe AIR 1.0 이상

이전의 여러 단원에서는 데이터를 검색, 삽입, 업데이트 및 삭제하고 데이터베이스 파일 및 테이블과 데이터베이스 안에 다른 객체를 만드는 등의 일반적인 데이터베이스 작업에 대해 설명했습니다. 예제에서는 이러한 작업을 비동기적으로 수행하는 방법과 동기적으로 수행하는 방법을 보여 주었습니다.

비동기 실행 모드에서는 사용자가 데이터베이스 엔진에 작업을 수행하도록 지시합니다. 데이터베이스 엔진은 응용 프로그램이 실행되는 동안 백그라운드에서 작업을 수행합니다. 작업이 완료되면 데이터베이스 엔진은 이 사실에 대해 알리는 이벤트를 전달합니다. 비동기 실행의 주요 장점은 기본 응용 프로그램 코드가 계속 실행되는 동안 런타임은 백그라운드에서 데이터베이스 작업을 수행하는 것입니다. 이는 작업이 실행되는 데 많은 시간이 걸리는 경우 특히 유용합니다.

반면에 동기 실행 모드에서는 작업이 백그라운드에서 실행되지 않습니다. 사용자가 데이터베이스 엔진에 작업을 수행하도록 지시합니다. 그러면 코드가 데이터베이스 엔진이 작업하는 시점에 일시 중지됩니다. 작업이 완료되면 코드의 다음 줄에서 실행이 계속됩니다.

단일 데이터베이스 연결을 사용하는 경우 일부 작업이나 문을 동기적으로 실행하고 다른 작업이나 문을 비동기적으로 실행할 수 없습니다. 데이터베이스에 대한 연결을 열 때 SQLConnection이 동기적으로 작동하는지, 아니면 비동기적으로 작동하는지를 지정합니다. SQLConnection.open()을 호출하면 연결이 동기 실행 모드에서 작동하고, SQLConnection.openAsync()를 호출하면 연결이 비동기 실행 모드에서 작동합니다. SQLConnection 인스턴스가 open() 또는 openAsync()를 사용하여 데이터베이스에 연결되면 동기 실행 모드나 비동기 실행 모드로 고정됩니다.

동기 데이터베이스 작업 사용

Adobe AIR 1.0 이상

비동기 실행 모드에 대한 코드와 비교할 때, 동기 실행을 사용하는 경우 작업을 실행하고 작업에 응답하는 데 사용하는 실제 코드에는 거의 차이점이 없습니다. 두 방법의 주요 차이점은 두 가지 영역으로 나뉩니다. 첫 번째 영역은 다른 작업에 의존하는 작업(예: SELECT 결과 행 또는 INSERT 문으로 추가된 행의 기본 키)의 실행입니다. 두 번째 영역은 오류 처리입니다.

동기 작업에 대한 코드 작성

Adobe AIR 1.0 이상

동기 실행과 비동기 실행의 주요 차이점은 동기 모드에서는 코드를 단일한 일련의 단계로 작성하는 것입니다. 이와 반대로, 비동기 코드에서는 이벤트 리스너를 등록하고 리스너 메서드 간에 작업을 나누는 경우가 많습니다. 데이터베이스가 동기 실행 모드에서 연결된 경우 한 코드 블록에서 일련의 데이터베이스 작업을 차례대로 실행할 수 있습니다. 다음 예제에서는 이 방법을 보여 줍니다.

```
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();
```

보시다시피 동기 실행을 사용하든 비동기 실행을 사용하든 간에 동일한 메서드를 호출하여 데이터베이스 작업을 수행합니다. 두 방법의 주요 차이점은 다른 작업과 오류 처리에 의존하는 작업을 실행하는 것입니다.

다른 작업에 의존하는 작업 실행

Adobe AIR 1.0 이상

동기 실행 모드를 사용하는 경우 작업이 완료될 때를 확인하기 위해 이벤트를 수신하는 코드를 작성할 필요가 없습니다. 대신 한 코드 줄에서 작업이 성공적으로 완료되면 실행이 다음 코드 줄에서 계속된다고 가정할 수 있습니다. 따라서 다른 작업의 성공에 의존하는 작업을 수행하려면 의존하는 작업 바로 뒤에 종속 코드를 작성하기만 하면 됩니다. 예를 들어, 트랜잭션을 시작하고 INSERT 문을 실행하여 삽입된 행의 기본 키를 검색한 후 다른 테이블의 다른 행에 삽입하고 마지막으로 트랜잭션을 커밋하는 응용 프로그램 코드를 작성하려면 일련의 문으로 코드를 모두 작성할 수 있습니다. 다음 예제에서는 이러한 작업을 보여 줍니다.

```
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();
```

동기 실행에서 오류 처리

Adobe AIR 1.0 이상

동기 실행 모드에서는 작업이 실패했음을 확인하기 위해 오류 이벤트를 수신하지 않습니다. 대신 오류를 트리거할 수 있는 코드를 try..catch..finally 코드 블록의 집합으로 둘러쌉니다. 오류를 발생시키는 코드를 try 블록으로 래핑합니다. 별도의 catch 블록에서 각 오류 유형에 응답하는 작업을 작성합니다. finally 블록에는 성공이나 실패에 관계없이 항상 실행할 코드(예: 더 이상 필요하지 않을 때 데이터베이스 연결을 닫음)를 배치합니다. 다음 예제에서는 오류 처리를 위해 try..catch..finally 블록을 사용하는 방법을 보여 줍니다. 이 예제는 이전 예제에 오류 처리 코드를 추가한 것입니다.

```
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();
}
```

비동기 실행 모드 이해

Adobe AIR 1.0 이상

비동기 실행 모드 사용에 대한 한 가지 일반적인 우려는 SQLStatement 인스턴스가 현재 동일한 데이터베이스 연결에 대해 실행되고 있으면 다른 SQLStatement 인스턴스의 실행을 시작할 수 없다는 가정입니다. 사실 이 가정은 올바르지 않습니다. SQLStatement 인스턴스가 실행되는 동안 문의 text 속성을 변경할 수는 없지만, 실행할 다른 SQL 문마다 별도의 SQLStatement 인스턴스를 사용하는 경우 다른 SQLStatement 인스턴스가 실행되는 동안 오류를 발생시키지 않고 SQLStatement의 execute() 메서드를 호출할 수 있습니다.

내부적으로 비동기 실행 모드를 사용하여 데이터베이스 작업을 실행하는 경우 각 데이터베이스 연결(각 SQLConnection 인스턴스)에는 수행하도록 지시를 받은 고유의 작업 큐 또는 목록이 있습니다. 런타임은 각 작업을 큐에 추가된 순서대로 실행합니다. SQLStatement 인스턴스를 만들고 execute() 메서드를 호출하면 이 문 실행 작업이 해당 연결에 대한 큐에 추가됩니다. 이 SQLConnection 인스턴스에서 작업이 현재 실행되지 않으면 문이 백그라운드에서 실행되기 시작합니다. 동일한 코드 블록에서 다른 SQLStatement 인스턴스를 만들고 해당 인스턴스의 execute() 메서드도 호출하는 경우 두 번째 문 실행 작업이 큐에서 첫

번째 문 뒤에 추가됩니다. 첫 번째 문의 실행이 완료되면 런타임은 즉시 큐의 다음 작업으로 이동합니다. 기본 응용 프로그램 코드에서 첫 번째 작업에 대한 result 이벤트가 전달되는 동안에도 큐에 있는 이후 작업의 처리는 백그라운드에서 이루어집니다. 다음 코드에서는 이 방법을 보여 줍니다.

```
// 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.
```

데이터베이스에서 큐에 있는 이후 문을 자동으로 실행하는 경우 중요한 부작용이 있습니다. 문이 다른 작업의 결과에 의존하는 경우 첫 번째 작업이 완료될 때까지 해당 문을 큐에 추가할 수 없습니다(즉, execute() 메서드를 호출할 수 없음). 이는 두 번째 문의 execute() 메서드를 호출하면 해당 문의 text 또는 parameters 속성을 변경할 수 없기 때문입니다. 이 경우 다음 작업을 시작하기 전에 첫 번째 작업이 완료된 것을 나타내는 이벤트를 기다려야 합니다. 예를 들어, 트랜잭션의 컨텍스트에서 문을 실행하려면 문 실행이 트랜잭션을 여는 작업에 의존합니다. SQLConnection.begin() 메서드를 호출하여 트랜잭션을 연 후 SQLConnection 인스턴스가 begin 이벤트를 전달할 때까지 기다려야 합니다. 이 이벤트가 전달된 후에만 SQLStatement 인스턴스의 execute() 메서드를 호출할 수 있습니다. 이 예제에서 작업이 제대로 실행되도록 응용 프로그램을 구성하는 가장 간단한 방법은 begin 이벤트의 수신기로 등록된 메서드를 만드는 것입니다. SQLStatement.execute() 메서드를 호출하는 코드는 해당 리스너 메서드 안에 배치됩니다.

SQL 데이터베이스에서 암호화 사용

Adobe AIR 1.5 이상

모든 Adobe AIR 응용 프로그램은 동일한 로컬 데이터베이스 엔진을 공유합니다. 따라서 모든 AIR 응용 프로그램에서 암호화되지 않은 데이터베이스 파일에 연결하고, 읽고, 쓸 수 있습니다. Adobe AIR 1.5부터 AIR에는 암호화된 데이터베이스 파일을 만들고 연결하는 기능이 포함되어 있습니다. 암호화된 데이터베이스를 사용하는 경우 데이터베이스에 연결하려면 응용 프로그램에서 올바른 암호화 키를 제공해야 합니다. 잘못된 암호화 키를 제공하거나 키를 제공하지 않으면 응용 프로그램이 데이터베이스에 연결할 수 없습니다. 따라서 응용 프로그램은 데이터베이스에서 데이터를 읽어오거나 데이터베이스에 데이터를 쓰거나 데이터를 변경할 수 없습니다.

암호화된 데이터베이스를 사용하려면 데이터베이스를 암호화된 데이터베이스로 만들어야 합니다. 기존의 암호화된 데이터베이스를 사용하여 데이터베이스에 대한 연결을 열 수 있습니다. 또한 암호화된 데이터베이스의 암호화 키를 변경할 수도 있습니다. 암호화된 데이터베이스를 만들고 연결한다는 점을 제외하고는 암호화된 데이터베이스로 작업하기 위한 기술은 암호화되지 않은 데이터베이스로 작업할 때와 동일합니다. 특히, 데이터베이스의 암호화 여부와 관계없이 동일하게 SQL 문을 실행합니다.

암호화된 데이터베이스 사용

Adobe AIR 1.5 이상

데이터베이스에 저장된 정보에 대한 액세스를 제한하려는 경우 언제든지 암호화를 사용할 수 있습니다. Adobe AIR의 데이터베이스 암호화 기능은 여러 목적으로 사용할 수 있습니다. 다음은 암호화된 데이터베이스를 사용할 수 있는 경우의 몇 가지 예입니다.

- 서버에서 다운로드한 개인 응용 프로그램 데이터의 읽기 전용 캐시
- 서버와 동기화되는 개인 데이터의 로컬 응용 프로그램 저장소(데이터가 서버로 전송되고 서버에서 로드됨)
- 응용 프로그램에서 만들고 편집한 문서의 파일 형식으로 사용되는 암호화된 파일. 이 파일은 한 사용자만 사용하거나 응용 프로그램의 모든 사용자가 공유하도록 설계할 수 있습니다.
- 로컬 데이터 저장소의 다른 사용 방법(예: 651페이지의 "로컬 SQL 데이터베이스의 용도"에서 설명한 것과 같이 시스템이나 데이터베이스 파일에 액세스할 수 있는 사용자로부터 데이터를 비공개로 유지해야 하는 경우)

암호화된 데이터베이스를 사용하려는 이유를 알고 있으면 응용 프로그램을 구축하는 방법을 결정하는 데 도움이 됩니다. 특히 이러한 결정은 응용 프로그램에서 데이터베이스의 암호화 키를 만들거나 가져오거나 저장하는 방법에 영향을 줄 수 있습니다. 이러한 고려 사항에 대한 자세한 내용은 694페이지의 "데이터베이스에서 암호화를 사용할 때 고려할 사항"을 참조하십시오.

암호화된 데이터베이스 외에 중요한 데이터를 비공개로 유지하는 다른 메커니즘은 암호화된 로컬 저장소입니다. 암호화된 로컬 저장소에서는 단일 ByteArray 값을 String 키를 사용하여 저장합니다. 암호화된 로컬 저장소에 값을 저장하는 AIR 응용 프로그램 및 해당 저장소가 있는 컴퓨터에서만 값에 액세스할 수 있습니다. 따라서 암호화된 로컬 저장소에서는 고유 암호화 키를 만들 필요가 없습니다. 이러한 이유 때문에 암호화된 로컬 저장소는 ByteArray로 쉽게 인코딩할 수 있는 값 집합이나 단일 값을 쉽게 저장하는 데 가장 적합합니다. 암호화된 데이터베이스는 구조적 데이터 저장소 및 쿼리가 필요한 큰 데이터 집합에 가장 적합합니다. 암호화된 로컬 저장소 사용에 대한 자세한 내용은 647페이지의 "암호화된 로컬 저장소"를 참조하십시오.

암호화된 데이터베이스 만들기

Adobe AIR 1.5 이상

암호화된 데이터베이스를 사용하려면 데이터베이스 파일을 만들 때 암호화해야 합니다. 데이터베이스를 암호화되지 않은 상태로 만들면 나중에 암호화할 수 없습니다. 마찬가지로 암호화된 데이터베이스를 나중에 암호화되지 않게 만들 수 없습니다. 필요한 경우 암호화된 데이터베이스의 암호화 키는 변경할 수 있습니다. 자세한 내용은 693페이지의 "데이터베이스의 암호화 키 변경"을 참조하십시오. 암호화되지 않은 기존 데이터베이스가 있을 경우 데이터베이스 암호화를 사용하려면 암호화된 데이터베이스를 새로 만들고 기존 테이블 구조 및 데이터를 새 데이터베이스로 복사할 수 있습니다.

암호화된 데이터베이스를 만드는 방법은 655페이지의 "데이터베이스 생성"에 설명된 암호화되지 않은 데이터베이스를 만드는 방법과 거의 동일합니다. 우선 데이터베이스에 대한 연결을 나타내는 SQLConnection 인스턴스를 만듭니다. 그리고 SQLConnection 객체의 open() 메서드나 openAsync() 메서드를 호출하여 데이터베이스를 만듭니다. 이때 아직 존재하지 않는 파일을 데이터베이스 위치로 지정합니다. 암호화된 데이터베이스를 만들 때는 encryptionKey 매개 변수(open() 메서드의 다섯 번째 매개 변수 및 openAsync() 메서드의 여섯 번째 매개 변수)에 값을 지정한다는 점만 다릅니다.

유효한 encryptionKey 매개 변수 값은 정확히 16바이트를 포함하는 ByteArray 객체입니다.

다음 예제에서는 암호화된 데이터베이스를 만드는 방법을 보여 줍니다. 간단하게 이 예제에서는 암호화 키를 응용 프로그램 코드에 하드 코딩합니다. 그러나 이 방법은 안전하지 않으므로 사용하지 않는 것이 좋습니다.

```
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);
```

암호화 키를 생성하는 좋은 방법을 보여 주는 예제는 695페이지의 “예: 암호화 키 생성 및 사용”을 참조하십시오.

암호화된 데이터베이스 연결

Adobe AIR 1.5 이상

암호화된 데이터베이스를 만드는 것과 마찬가지로 암호화된 데이터베이스에 대한 연결을 여는 절차는 암호화되지 않은 데이터베이스에 연결할 때와 동일합니다. 이 절차에 대해서는 662페이지의 “데이터베이스 연결”에서 자세히 설명합니다. open() 메서드를 사용하여 동기 실행 모드에서 이 연결을 열거나 openAsync() 메서드를 사용하여 비동기 실행 모드에서 이 연결을 엽니다. 암호화된 데이터베이스를 열려면 encryptionKey 매개 변수(open() 메서드의 다섯 번째 매개 변수 및 openAsync() 메서드의 여섯 번째 매개 변수)에 올바른 값을 지정한다는 점만 다릅니다.

제공한 암호화 키가 올바르지 않으면 오류가 발생합니다. open() 메서드의 경우 SQLError 예외가 발생되고, openAsync() 메서드의 경우 SQLConnection 객체가 SQLErrorEvent를 전달하여 이 이벤트의 error 속성에 SQLError 객체가 포함됩니다. 두 경우 모두 예외에서 생성되는 SQLError 객체의 errorID 속성 값은 3138입니다. 이 오류 ID는 오류 메시지 "연 파일은 데이터베이스 파일이 아닙니다"에 해당합니다.

다음 예제에서는 암호화된 데이터베이스를 비동기 실행 모드로 여는 방법을 보여 줍니다. 간단하게 이 예제에서는 암호화 키를 응용 프로그램 코드에 하드 코딩합니다. 그러나 이 방법은 안전하지 않으므로 사용하지 않는 것이 좋습니다.

```
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);
    }
}
```

다음 예제에서는 암호화된 데이터베이스를 동기 실행 모드로 여는 방법을 보여 줍니다. 간단하게 이 예제에서는 암호화 키를 응용 프로그램 코드에 하드 코딩합니다. 그러나 이 방법은 안전하지 않으므로 사용하지 않는 것이 좋습니다.

```
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);
    }
}
```

암호화 키를 생성하는 좋은 방법을 보여 주는 예제는 695페이지의 “예: 암호화 키 생성 및 사용”을 참조하십시오.

데이터베이스의 암호화 키 변경

Adobe AIR 1.5 이상

데이터베이스를 암호화하는 경우 나중에 데이터베이스의 암호화 키를 변경할 수 있습니다. 데이터베이스의 암호화 키를 변경하려면 먼저 SQLConnection 인스턴스를 만들고 이 인스턴스의 open() 또는 openAsync() 메서드를 호출하여 데이터베이스에 대한 연결을 엽니다. 데이터베이스에 연결되면 새 암호화 키를 인수로 전달하여 reencrypt() 메서드를 호출합니다.

대부분의 다른 데이터베이스 작업처럼 reencrypt() 메서드의 비헤이비어는 데이터베이스 연결에서 동기 실행 모드를 사용하는지 비동기 실행 모드를 사용하는지에 따라 달라집니다. open() 메서드를 사용하여 데이터베이스에 연결하는 경우 reencrypt() 작업이 동기적으로 실행됩니다. 작업이 완료되면 코드의 다음 줄에서 실행이 계속됩니다.

```
var newKey:ByteArray = new ByteArray();
// ... generate the new key and store it in newKey
conn.reencrypt(newKey);
```

반면 openAsync() 메서드를 사용하여 데이터베이스 연결을 여는 경우 reencrypt() 작업은 비동기적입니다. 즉, reencrypt()를 호출하면 재암호화 프로세스가 시작되고, 작업이 완료되면 SQLConnection 객체가 reencrypt 이벤트를 전달합니다. 재암호화가 완료되는 때를 확인하려면 이벤트 리스너를 사용합니다.

```
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
}
```

reencrypt() 작업은 자체의 트랜잭션에서 실행됩니다. 작업이 중단되거나 실패하면(예: 작업이 완료되기 전에 응용 프로그램이 닫히는 경우) 트랜잭션이 롤백됩니다. 이 경우 원래 암호화 키가 계속 데이터베이스의 암호화 키입니다.

reencrypt() 메서드를 사용하여 데이터베이스에서 암호화를 제거할 수는 없습니다. reencrypt() 메서드에 16바이트 ByteArray가 아닌 암호화 키나 null 값을 전달하면 오류가 발생합니다.

데이터베이스에서 암호화를 사용할 때 고려할 사항

Adobe AIR 1.5 이상

690페이지의 "암호화된 데이터베이스 사용" 단원에서는 암호화된 데이터베이스를 사용해야 하는 여러 경우를 제시합니다. 분명한 점은 이러한 시나리오 및 다른 시나리오를 포함한 서로 다른 응용 프로그램에 대한 사용 시나리오마다 개인 정보 보호 요구 사항이 다르다는 것입니다. 응용 프로그램에서 암호화 사용을 어떻게 구축하는지에 따라 데이터베이스 데이터의 비공개 정도를 제어하는 데 상당한 영향을 미칩니다. 예를 들어 암호화된 데이터베이스를 사용하여 개인 데이터를 같은 시스템의 다른 사용자를 비롯한 다른 사용자에게 비공개로 유지하려면 각 사용자의 데이터베이스에 자체의 암호화 키가 있어야 합니다. 보안을 강화하기 위해 응용 프로그램에서 사용자가 입력한 암호를 사용하여 키를 생성할 수 있습니다. 암호화 키가 암호를 기반으로 하는 경우 다른 사용자가 시스템에서 사용자의 계정을 가장할 수 있더라도 여전히 데이터에 액세스할 수 없습니다. 개인 정보 보호 범위의 다른 일면으로 데이터베이스 파일을 응용 프로그램의 모든 사용자가 읽을 수 있게 하지만 다른 응용 프로그램에서는 읽을 수 없게 하려는 경우를 가정합니다. 이 경우 설치된 각 응용 프로그램 사본에서 공유 암호화 키에 액세스할 수 있어야 합니다.

응용 프로그램, 그리고 특히 암호화 키를 생성하는 데 사용하는 기술을 설계할 때 응용 프로그램 데이터에 필요한 개인 정보 보호 수준을 기준으로 할 수 있습니다. 다음 목록에서는 여러 데이터 개인 정보 보호 수준에 대한 디자인 제안을 제공합니다.

- 모든 시스템에서 응용 프로그램에 액세스할 수 있는 모든 사용자가 데이터베이스에 액세스할 수 있게 하려면 응용 프로그램의 모든 인스턴스에 대해 사용할 수 있는 단일 키를 사용합니다. 예를 들어 처음으로 실행될 때 응용 프로그램은 SSL과 같은 보안 프로토콜을 사용하여 서버에서 공유 암호화 키를 다운로드할 수 있습니다. 그런 다음 키를 나중에 사용하기 위해 암호화된 로컬 저장소에 저장할 수 있습니다. 또는 데이터를 시스템의 사용자별로 암호화하고 서버와 같은 원격 데이터 저장소와 데이터를 동기화하여 데이터를 이식 가능하게 만듭니다.
- 모든 시스템의 단일 사용자가 데이터베이스에 액세스할 수 있게 만들려면 사용자 비밀(예: 암호)을 사용하여 암호화 키를 생성합니다. 특히 특정 컴퓨터에 연결된 값(예: 암호화된 로컬 저장소에 저장된 값)을 사용하여 키를 생성하지 않습니다. 또는 데이터를 시스템의 사용자별로 암호화하고 서버와 같은 원격 데이터 저장소와 데이터를 동기화하여 데이터를 이식 가능하게 만듭니다.
- 단일 시스템의 단일 사용자가 데이터베이스에 액세스할 수 있게 하려면 암호화 생성된 솔트를 사용하여 키를 생성합니다. 이 방법에 대한 예는 695페이지의 "예: 암호화 키 생성 및 사용"을 참조하십시오.

다음은 암호화된 데이터베이스를 사용할 응용 프로그램을 설계할 때 유념해야 할 추가 보안 고려 사항입니다.

- 시스템 보안은 가장 약한 링크의 보안 수준으로만 유지됩니다. 사용자 입력 암호를 사용하여 암호화 키를 생성하는 경우 암호에 최소 길이 및 복잡도 제한 사항을 적용하는 것이 좋습니다. 기본 문자만 사용하는 짧은 암호는 쉽게 추측할 수 있습니다.
- AIR 응용 프로그램의 소스 코드는 사용자 컴퓨터에 일반 텍스트(HTML 내용의 경우) 또는 쉽게 디컴파일 할 수 있는 이진 형식(SWF 내용의 경우)으로 저장됩니다. 소스 코드에 액세스할 수 있으므로 다음 두 가지 사항에 유념해야 합니다.
  - 암호화 키를 소스 코드에 하드 코딩하지 마십시오.
  - 암호화 키를 생성하는 데 사용하는 기술(예: 임의의 문자 생성기 또는 특정 해시 알고리즘)은 공격자가 쉽게 파악할 수 있는 것으로 항상 간주하십시오.
- AIR 데이터베이스 암호화에서는 CCM(Counter with CBC-MAC) 모드와 함께 AES(Advanced Encryption Standard)를 사용합니다. 이 암호화 암호에서는 보안을 위해 사용자가 입력한 키를 솔트 값과 결합해야 합니다. 이 방법에 대한 예는 695페이지의 "예: 암호화 키 생성 및 사용"을 참조하십시오.
- 데이터베이스를 암호화하는 경우 데이터베이스 엔진에서 사용하는 모든 디스크 파일이 데이터베이스와 함께 암호화됩니다. 그러나 데이터베이스 엔진에서는 트랜잭션의 읽기 및 쓰기 시간 성능을 개선하기 위해 일부 데이터를 메모리 내 캐시에 보관합니다. 메모리에 상주하는 모든 데이터는 암호화되지 않습니다. 공격자가 디버거를 사용하는 등의 방법으로 AIR 응용 프로

그램에서 사용하는 메모리에 액세스할 수 있으면 현재 열려 있고 암호화되지 않은 데이터베이스의 데이터를 사용할 수 있게 됩니다.

예: 암호화 키 생성 및 사용

Adobe AIR 1.5 이상

다음 예제 응용 프로그램에서는 암호화 키를 생성하는 한 가지 방법을 보여 줍니다. 이 응용 프로그램은 사용자 데이터에 대한 가장 높은 수준의 개인 정보 보호 및 보안을 제공하도록 설계됩니다. 개인 데이터를 보호하기 위한 중요한 방법 한 가지는 응용 프로그램에서 데이터베이스에 연결할 때마다 사용자에게 암호를 입력하도록 요구하는 것입니다. 이러한 맥락에서 높은 수준의 개인 정보 보호가 필요한 응용 프로그램에서는 다음 예제와 같이 데이터베이스 암호화 키를 직접 저장하지 않아야 합니다.

다음 응용 프로그램은 암호화 키를 생성하는 ActionScript 클래스(EncryptionKeyGenerator 클래스)와 이 클래스를 사용하는 방법을 보여주는 기본 사용자 인터페이스의 두 부분으로 구성되어 있습니다. 전체 소스 코드는 696페이지의 "암호화 키를 생성 및 사용하기 위한 전체 예제 코드"를 참조하십시오.

EncryptionKeyGenerator 클래스를 사용하여 보안 암호화 키 생성

Adobe AIR 1.5 이상

응용 프로그램에서 사용하기 위해 EncryptionKeyGenerator 클래스의 자세한 작동 방식을 이해할 필요는 없습니다. 이 클래스가 데이터베이스의 암호화 키를 생성하는 방식에 대한 자세한 내용은 700페이지의 "EncryptionKeyGenerator 클래스 이해"를 참조하십시오.

응용 프로그램에 EncryptionKeyGenerator 클래스를 사용하려면 다음 단계를 수행하십시오.

1. EncryptionKeyGenerator 클래스를 소스 코드 또는 컴파일된 SWC로 다운로드합니다. EncryptionKeyGenerator 클래스는 오픈 소스 ActionScript 3.0 기본 라이브러리(as3corelib) 프로젝트에 포함되어 있습니다. 소스 코드와 설명서를 포함한 as3corelib 패키지를 다운로드할 수 있습니다. 또한 이 프로젝트 페이지에서 SWC 또는 소스 코드 파일도 다운로드할 수 있습니다.
2. EncryptionKeyGenerator 클래스의 소스 코드(as3corelib SWC)를 응용 프로그램 소스 코드에서 찾을 수 있는 위치에 배치합니다.
3. 응용 프로그램 소스 코드에 EncryptionKeyGenerator 클래스를 가져오기 위한 import 문을 추가합니다.

   ```
   import com.adobe.air.crypto.EncryptionKeyGenerator;
   ```

4. 코드에서 데이터베이스를 만들거나 데이터베이스에 대한 연결을 여는 코드 이전에 EncryptionKeyGenerator() 생성자를 호출하여 EncryptionKeyGenerator 인스턴스를 만드는 코드를 추가합니다.

   ```
   var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
   ```

5. 사용자에게서 암호를 입력받습니다.

   ```
   var password:String = passwordInput.text;

   if (!keyGenerator.validateStrongPassword(password))
   {
       // display an error message
       return;
   }
   ```

   이 암호는 EncryptionKeyGenerator 인스턴스가 다음 단계에서 암호화 키를 생성하는 데 기반으로 사용됩니다. EncryptionKeyGenerator 인스턴스는 특정 강력한 암호 유효성 검사 요구 사항을 기준으로 암호를 테스트합니다. 유효성 검사에 실패하면 오류가 발생합니다. 예제 코드와 같이 EncryptionKeyGenerator 객체의 validateStrongPassword() 메서드를 호출하여 암호를 미리 검사할 수 있습니다. 이러한 방식으로 암호가 강력한 암호를 위한 최소 요구 사항을 충족하는지 확인하고 오류를 방지할 수 있습니다.

6 암호를 기반으로 암호화 키를 생성합니다.

```
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

getEncryptionKey() 메서드는 암호화 키(16바이트 ByteArray)를 생성하여 반환합니다. 이제 이 암호화 키를 사용하여 암호화된 데이터베이스를 새로 만들거나 기존 암호화된 데이터베이스를 열 수 있습니다.

getEncryptionKey() 메서드에는 필수 매개 변수가 하나 있는데 이는 5단계에서 가져온 암호입니다.

참고: 최고 수준의 개인 정보 보호 및 보안을 유지하려면 응용 프로그램에서 데이터베이스에 연결할 때마다 사용자에게 암호를 입력하도록 요구해야 합니다. 사용자 암호 또는 데이터베이스 암호화 키를 직접 저장하지 마십시오. 저장할 경우 보안 위험을 유발할 수 있습니다. 대신 이 예제와 같이, 응용 프로그램에서 암호화된 데이터베이스를 만들 때와 만든 데이터베이스에 연결할 때 모두 암호를 기반으로 암호를 생성하는 방법을 사용해야 합니다.

getEncryptionKey() 메서드는 선택적인 두 번째 매개 변수인 overrideSaltELSKey도 받습니다. EncryptionKeyGenerator는 암호화 키의 일부로 사용되는 솔트라고 하는 임의의 값을 만듭니다. 솔트 값은 암호화 키를 다시 만들 수 있도록 AIR 응용 프로그램의 암호화된 로컬 저장소(ELS)에 저장됩니다. 기본적으로 EncryptionKeyGenerator 클래스는 특정 String을 ELS 키로 사용하며 그럴 가능성은 적지만 이 기본 ELS 키가 응용 프로그램에서 사용하는 다른 키와 충돌할 수 있습니다. 따라서 기본 키를 사용하는 대신 고유 ELS 키를 지정하는 것도 좋습니다. 이렇게 하려면 다음과 같이 두 번째 getEncryptionKey() 매개 변수로 전달하여 사용자 정의 키를 지정하십시오.

```
var customKey:String = "My custom ELS salt key";
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password, customKey);
```

7 데이터베이스 만들기 또는 열기

getEncryptionKey() 메서드가 반환한 암호화 키를 사용하여 코드에서 암호화된 데이터베이스를 새로 만들거나 기존 암호화된 데이터베이스를 열 수 있습니다. 두 경우 모두 690페이지의 "암호화된 데이터베이스 만들기" 및 691페이지의 "암호화된 데이터베이스 연결"에서 설명한 대로 SQLConnection 클래스의 open() 또는 openAsync() 메서드를 사용합니다.

이 예제의 응용 프로그램은 데이터베이스를 비동기 실행 모드로 열도록 설계되었습니다. 다음 코드에서는 적절한 이벤트 리스너를 설정하고 암호화 키를 최종 인수로 전달하여 openAsync() 메서드를 호출합니다.

```
conn.addEventListener(SQLEvent.OPEN, openHandler);
conn.addEventListener(SQLErrorEvent.ERROR, openError);

conn.openAsync(dbFile, SQLMode.CREATE, null, false, 1024, encryptionKey);
```

리스너 메서드의 코드에서는 이벤트 리스너 등록을 제거합니다. 그런 다음 데이터베이스가 생성되었는지, 열렸는지 또는 오류가 발생했는지를 나타내는 상태 메시지를 표시합니다. 이러한 이벤트 핸들러에서 가장 주목할 부분은 openError() 메서드입니다. 이 메서드에서는 if 문이 데이터베이스가 존재하는지 확인하고(기존 데이터베이스에 연결 시도) 오류 ID가 상수 EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID와 일치하는지 확인합니다. 이러한 조건이 모두 충족되는 경우 사용자가 입력한 암호가 올바르지 않을 수 있습니다. 또한 지정한 파일이 아예 데이터베이스 파일이 아닐 수도 있습니다. 다음은 오류 ID를 확인하는 코드입니다.

```
if (!createNewDB && event.error.errorID == EncryptionKeyGenerator.ENCRYPTED_DB_PASSWORD_ERROR_ID)
{
    statusMsg.text = "Incorrect password!";
}
else
{
    statusMsg.text = "Error creating or opening database.";
}
```

예제 이벤트 리스너의 전체 코드는 696페이지의 "암호화 키를 생성 및 사용하기 위한 전체 예제 코드"를 참조하십시오.

암호화 키를 생성 및 사용하기 위한 전체 예제 코드

Adobe AIR 1.5 이상

다음은 예제 응용 프로그램 "암호화 키 생성 및 사용"의 전체 코드입니다. 이 코드는 두 부분으로 구성되어 있습니다.

이 예제에서는 EncryptionKeyGenerator 클래스를 사용하여 암호를 기반으로 암호화 키를 만듭니다. EncryptionKeyGenerator 클래스는 오픈 소스 ActionScript 3.0 기본 라이브러리(as3corelib) 프로젝트에 포함되어 있습니다. 소스 코드와 설명서를 포함한 as3corelib 패키지를 다운로드할 수 있습니다. 또한 이 프로젝트 페이지에서 SWC 또는 소스 코드 파일도 다운로드할 수 있습니다.

Flex 예제

응용 프로그램 MXML 파일에는 암호화된 데이터베이스를 만들거나 암호화된 데이터베이스에 대한 연결을 열기 위한 간단한 응용 프로그램의 소스 코드가 들어 있습니다.

```
<?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 예제

응용 프로그램 FLA 파일에는 암호화된 데이터베이스를 만들거나 암호화된 데이터베이스에 대한 연결을 열기 위한 간단한 응용 프로그램의 소스 코드가 들어 있습니다. FLA 파일에는 스테이지에 배치되는 네 가지 구성 요소가 있습니다.

인스턴스 이름	구성 요소 유형	설명
지침	레이블	사용자에게 제공되는 지침이 들어 있습니다.

인스턴스 이름	구성 요소 유형	설명
passwordInput	TextInput	사용자가 암호를 입력하는 입력 필드입니다.
openButton	버튼	사용자가 암호를 입력한 후 클릭하는 버튼입니다.
statusMsg	레이블	상태(성공 또는 실패) 메시지를 표시합니다.

응용 프로그램의 코드가 기본 타임라인에서 프레임 1의 키프레임에 정의됩니다. 다음은 응용 프로그램의 코드입니다.

```
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.";
    }
}
```

EncryptionKeyGenerator 클래스 이해

Adobe AIR 1.5 이상

EncryptionKeyGenerator 클래스를 사용하여 응용 프로그램 데이터베이스의 보안 암호화 키를 만들기 위해 이 클래스가 내부적으로 작동하는 방식을 이해할 필요는 없습니다. EncryptionKeyGenerator 클래스를 사용하는 과정은 695페이지의 “EncryptionKeyGenerator 클래스를 사용하여 보안 암호화 키 생성”에 설명되어 있습니다. 그러나 EncryptionKeyGenerator 클래스의 내부 작동 방식을 이해하면 도움이 될 수도 있습니다. 예를 들어 다른 수준의 개인 정보 보호가 필요한 경우 EncryptionKeyGenerator 클래스를 그에 맞게 조정하거나 이 클래스의 일부 작동 방식을 통합할 수 있습니다.

EncryptionKeyGenerator 클래스는 오픈 소스 ActionScript 3.0 기본 라이브러리(as3corelib) 프로젝트에 포함되어 있습니다. 소스 코드와 설명서를 포함한 as3corelib 패키지를 다운로드할 수 있습니다. 또한 프로젝트 사이트에서 소스 코드를 보거나 다운로드하여 설명과 함께 따라할 수도 있습니다.

코드에서 EncryptionKeyGenerator 인스턴스를 만들어 getEncryptionKey() 메서드를 호출하면 여러 단계를 거쳐 올바른 사용자만 데이터에 액세스할 수 있게 합니다. 이 프로세스는 데이터베이스를 새로 만들기 위해 사용자가 입력한 암호에서 암호화 키를 생성하고 데이터베이스를 열기 위해 암호화 키를 다시 생성할 때 동일하게 적용됩니다.

강력한 암호 가져오기 및 유효성 검사
Adobe AIR 1.5 이상

코드에서 getEncryptionKey() 메서드를 호출하면 매개 변수로 암호를 전달합니다. 암호는 암호화 키의 기준으로 사용됩니다. 이 디자인에서는 사용자만 아는 정보를 사용하여 암호를 아는 사용자만 데이터베이스의 데이터에 액세스할 수 있게 합니다. 공격자가 컴퓨터에서 사용자의 계정에 액세스하더라도 암호를 모르면 데이터베이스에 침투할 수 없습니다. 보안을 극대화하기 위해 응용 프로그램에서 암호를 저장하지 않습니다.

응용 프로그램의 코드는 EncryptionKeyGenerator 인스턴스를 만들고 해당 getEncryptionKey() 메서드를 호출하여 사용자가 입력한 암호를 인수(이 예제에서는 password 변수)로 전달합니다.

```
var keyGenerator:EncryptionKeyGenerator = new EncryptionKeyGenerator();
var encryptionKey:ByteArray = keyGenerator.getEncryptionKey(password);
```

getEncryptionKey() 메서드가 호출될 때 EncryptionKeyGenerator 클래스가 수행하는 첫 번째 단계는 사용자가 입력한 암호가 강력한 암호 요구 사항을 충족하는지 검사하는 것입니다. EncryptionKeyGenerator 클래스는 8 - 32자 길이의 문자를 요구합니다. 암호에는 대문자와 소문자가 섞여 있고 하나 이상의 숫자 또는 심볼 문자가 포함되어 있어야 합니다.

이 패턴을 확인하는 일반 표현식은 STRONG_PASSWORD_PATTERN이라는 상수로 정의됩니다.

```
private static const STRONG_PASSWORD_PATTERN:RegExp = /(?=^.{8,32}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-Z])(?=.*[a-z]).*$/;
```

암호를 검사하는 코드는 EncryptionKeyGenerator 클래스의 validateStrongPassword() 메서드에 있습니다. 코드는 다음과 같습니다.

```
public function vaidateStrongPassword(password:String):Boolean
{
    if (password == null || password.length <= 0)
    {
        return false;
    }

    return STRONG_PASSWORD_PATTERN.test(password))
}
```

내부적으로 getEncryptionKey() 메서드는 EncryptionKeyGenerator 클래스의 validateStrongPassword() 메서드를 호출하고 암호가 유효하지 않을 경우 예외를 발생시킵니다. validateStrongPassword() 메서드는 공용 메서드이므로 코드에서 응용 프로그램 코드가 getEncryptionKey() 메서드를 호출하지 않고 암호를 검사해도 오류가 발생하지 않습니다.

암호를 256비트로 확대
Adobe AIR 1.5 이상

이 프로세스의 뒷부분에서는 암호의 길이가 256비트여야 합니다. 각 사용자에게 정확히 256비트(32자)인 암호를 입력하게 하는 대신 이 코드에서는 암호 문자를 반복하여 더 긴 암호를 만듭니다.

getEncryptionKey() 메서드는 concatenatePassword() 메서드를 호출하여 긴 암호를 만드는 작업을 수행합니다.

```
var concatenatedPassword:String = concatenatePassword(password);
```

다음은 concatenatePassword() 메서드의 코드입니다.

```
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;
}
```

암호가 256비트 미만이면 코드는 암호를 암호 자체와 연결하여 256비트로 만듭니다. 길이가 정확히 처리되지 않으면 마지막 반복이 짧아져 정확히 256비트에 맞춥니다.

256비트 솔트 값 생성 또는 가져오기
Adobe AIR 1.5 이상

다음 단계는 이후의 단계에서 암호와 결합되는 256비트 솔트 값을 가져오는 것입니다. 솔트는 사용자가 입력한 값에 추가하거나 결합하여 암호를 형성하는 임의의 값입니다. 솔트와 암호를 함께 사용하면 사용자가 실제 단어나 일반 용어를 암호로 선택하더라도 시스템에서 사용하는 암호와 솔트 조합은 임의의 값이 됩니다. 이렇게 하면 공격자가 단어 목록을 사용하여 암호를 추측하려고 하는 사전 공격(Dictionary Attack)을 방지할 수 있습니다. 또한 솔트 값을 생성하여 암호화된 로컬 저장소에 저장하면 이 값이 데이터베이스 파일이 있는 시스템의 사용자 계정에 연결됩니다.

응용 프로그램이 getEncryptionKey() 메서드를 처음으로 호출하면 코드에서 임의의 256비트 솔트 값을 생성합니다. 그렇지 않은 경우에는 코드에서 암호화된 로컬 저장소의 솔트 값을 로드합니다.

솔트는 salt라는 변수에 저장됩니다. 코드는 암호화된 로컬 저장소에서 솔트를 로드하려고 시도하여 이미 솔트를 생성했는지 확인합니다.

```
var salt:ByteArray = EncryptedLocalStore.getItem(saltKey);
if (salt == null)
{
    salt = makeSalt();
    EncryptedLocalStore.setItem(saltKey, salt);
}
```

코드에서 새 솔트 값을 만들 경우 makeSalt() 메서드가 임의의 256비트 값을 생성합니다. 이 값은 결과적으로 암호화된 로컬 저장소에 저장되므로 ByteArray 객체로 생성됩니다. makeSalt() 메서드는 Math.random() 메서드를 사용하여 값을 임의로 생성합니다. Math.random() 메서드는 256비트를 한 번에 생성할 수 없습니다. 대신 루프를 사용하여 Math.random()을 여덟 번 호출합니다. 호출할 때마다 0에서 4294967295(최대 단위 값) 사이의 임의의 단위 값이 생성됩니다. 한 단위에서 정확히 32비트를 사용하므로 단위 값은 편의를 위해 사용됩니다. 여덟 개의 단위 값을 ByteArray에 쓰면 256비트 값이 생성됩니다. 다음은 makeSalt() 메서드의 코드입니다.

```
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;
}
```

코드에서 솔트를 암호화된 로컬 저장소(ELS)에 저장하거나 ELS에서 솔트를 검색하려면 솔트가 저장된 String 키가 필요합니다. 이 키를 모르면 솔트 값을 검색할 수 없습니다. 즉, 이 키를 모를 경우 데이터베이스를 다시 열 때마다 암호화 키를 다시 생성할 수 없습니다. 기본적으로 EncryptionKeyGenerator는 상수 SALT_ELS_KEY에 사전 정의된 ELS 키를 사용합니다. 기본 키를 사용하는 대신 응용 프로그램 코드에서 getEncryptionKey() 메서드 호출에 사용할 ELS 키를 지정할 수도 있습니다. 기본 솔트 ELS 키 또는 응용 프로그램에서 지정한 솔트 ELS 키는 saltKey라는 변수로 저장됩니다. 앞에 나온 것처럼 해당 변수가 EncryptedLocalStore.setItem() 및 EncryptedLocalStore.getItem()에 대한 호출에 사용됩니다.

XOR 연산자를 사용하여 256비트 암호와 솔트 결합

Adobe AIR 1.5 이상

이제 코드에는 256비트 암호와 256비트 솔트 값이 있습니다. 그 다음에 코드에서는 비트 XOR 작업을 사용하여 연결된 암호와 솔트를 하나의 값으로 결합합니다. 그런 다음 사용 가능한 전체 문자 범위의 문자로 구성된 256비트 암호를 만듭니다. 이 원칙이 적용되기는 하지만 실제 암호 입력은 주로 영숫자로 구성됩니다. 이렇게 높은 임의성으로 사용 가능한 암호 집합이 커지기 때문에 사용자가 길고 복잡한 암호를 입력하지 않아도 됩니다.

XOR 작업의 결과는 unhashedKey 변수에 저장됩니다. 두 값에 대한 비트 XOR을 수행하는 실제 프로세스는 xorBytes() 메서드에서 일어납니다:

```
var unhashedKey:ByteArray = xorBytes(concatenatedPassword, salt);
```

비트 XOR 연산자(^)는 두 단위 값을 가져와서 하나의 단위 값을 반환합니다. 하나의 단위 값은 32비트로 구성됩니다. xorBytes() 메서드에 인수로 전달되는 입력 값은 String(암호) 및 ByteArray(솔트)입니다. 따라서 코드에서는 루프를 사용하여 각 입력에서 한 번에 32비트를 추출하고 XOR 연산자를 사용하여 결합합니다.

```
private function xorBytes(passwordString:String, salt:ByteArray):ByteArray
{
    var result:ByteArray = new ByteArray();

    for (var i:uint = 0; i < 32; i += 4)
    {
        // ...
    }

    return result;
}
```

루프 내에서 먼저 32비트(4바이트)가 passwordString 매개 변수에서 추출됩니다. 이러한 비트는 추출된 후 두 부분으로 이루어진 프로세스에서 단위(o1)로 변환됩니다. 먼저 charCodeAt() 메서드는 각 문자의 숫자 값을 가져옵니다. 그런 다음 비트 왼쪽 시프트 연산자(<<)를 사용하여 해당 값을 단위의 적절한 위치로 이동하고 이동된 값을 o1에 추가합니다. 예를 들어 첫 번째 문자(i)는 비트 왼쪽 시프트 연산자(<<)를 사용하여 24비트 왼쪽으로 이동하고 o1에 할당하여 첫 번째 8비트가 됩니다. 두 번째 문자(i + 1)는 왼쪽으로 16비트 이동하고 결과를 o1에 추가하여 두 번째 8비트가 됩니다. 세 번째 및 네 번째 문자 값도 같은 방식으로 추가됩니다.

```
        // ...

        // 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);

        // ...
```

이제 o1 변수에는 passwordString 매개 변수의 32비트가 들어 있습니다. 다음으로 readUnsignedInt() 메서드를 호출하여 salt 매개 변수에서 32비트를 추출합니다. 32비트는 uint 변수 o2에 저장됩니다.

```
        // ...

        salt.position = i;
        var o2:uint = salt.readUnsignedInt();

        // ...
```

마지막으로 XOR 연산자를 사용하여 이 두 32비트(uint) 값을 결합하고 그 결과를 result라는 ByteArray에 씁니다.

```
        // ...

        var xor:uint = o1 ^ o2;
        result.writeUnsignedInt(xor);
        // ...
```

루프가 완료되면 XOR 결과를 포함하는 ByteArray가 반환됩니다.

```
        // ...
    }

    return result;
}
```

키 해시

Adobe AIR 1.5 이상

연결된 암호와 솔트가 결합되면 다음 단계로 이 값을 SHA-256 해시 알고리즘으로 해시하여 보안을 강화합니다. 값을 해시하면 공격자가 값을 리버스 엔지니어링하기 더 어렵게 만듭니다.

이제 코드에는 솔트가 결합된 연결된 암호를 포함하는 unhashedKey라는 ByteArray가 있습니다. ActionScript 3.0 기본 라이브러리(as3corelib) 프로젝트에는 com.adobe.crypto 패키지에 SHA256 클래스가 포함되어 있습니다. 이 클래스에는 ByteArray에 대한 SHA-256 해시를 수행하고 256비트 해시 결과를 포함하는 String을 16진수로 반환하는 SHA256.hashBytes() 메서드가 있습니다. EncryptionKeyGenerator 클래스는 SHA256 클래스를 사용하여 키를 해시합니다.

```
var hashedKey:String = SHA256.hashBytes(unhashedKey);
```

해시에서 암호화 키 추출

Adobe AIR 1.5 이상

암호화 키는 길이가 정확히 16바이트(128비트)인 ByteArray여야 합니다. SHA-256 해시 알고리즘의 결과는 항상 길이가 256비트입니다. 따라서 마지막 단계에서는 해시된 결과에서 128비트를 선택하여 실제 암호화 키로 사용합니다.

EncryptionKeyGenerator 클래스에서는 generateEncryptionKey() 메서드를 호출하여 키를 128비트로 줄입니다. 그런 다음 이 메서드 결과를 getEncryptionKey() 메서드의 결과로 반환합니다.

```
var encryptionKey:ByteArray = generateEncryptionKey(hashedKey);
return encryptionKey;
```

첫 번째 128비트를 암호화 키로 사용할 필요는 없습니다. 임의이 지점에서 시작하는 비트 범위를 선택할 수도 있고 하나씩 걸러서 비트를 선택할 수도 있고 그 밖에 다양한 방법으로 비트를 선택할 수 있습니다. 중요한 점은 코드에서 128개의 고유한 비트를 선택하고 이 동일한 128비트가 매번 사용된다는 것입니다.

generateEncryptionKey() 메서드에서는 18번째 바이트에서 시작되는 비트 범위를 암호화 키로 사용합니다. 앞에서 설명한 대로 SHA256 클래스는 256비트 해시가 포함된 String을 16진수로 반환합니다. 128비트의 단일 블록은 너무 많은 바이트여서 ByteArray에 한 번에 추가할 수 없습니다. 따라서 코드에서는 for 루프를 사용하여 16진수 String에서 문자를 추출하고 이를 실제 숫자 값으로 변환한 다음 ByteArray에 추가합니다. SHA-256 결과 String은 길이가 64자입니다. 128비트 범위는 String의 32자와 동일하므로 각 문자는 4비트를 나타냅니다. ByteArray에 추가할 수 있는 가장 작은 데이터 단위는 1바이트(8비트)이며, 이는 hash String에 있는 두 문자에 해당합니다. 따라서 루프는 0에서 31(32자)까지 2자씩 증가하여 계산합니다.

루프 내에서 코드는 먼저 현재 문자 쌍의 시작 위치를 결정합니다. 원하는 범위가 인덱스 위치 17(18번째 바이트)의 문자에서 시작되므로 position 변수는 현재 반복기 값(i)에 17을 더한 값으로 할당됩니다. 코드에서는 String 객체의 substr() 메서드를 사용하여 현재 위치에서 두 문자를 추출합니다. 이러한 문자는 hex 변수에 저장됩니다. 그런 다음 코드에서는 parseInt() 메서드를 사용하여 hex String을 10진수 정수 값으로 변환합니다. 그리고 이 값을 int 변수 byte에 저장합니다. 마지막으로 코드에서는 byte의 값을 writeByte() 메서드를 사용하여 result ByteArray에 추가합니다. 루프가 완료되면 result ByteArray는 16바이트를 포함하게 되며 데이터베이스 암호화 키로 사용할 수 있습니다.

```
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;
}
```

SQL 데이터베이스 작업을 위한 전략

Adobe AIR 1.0 이상

응용 프로그램에서 로컬 SQL 데이터베이스에 액세스하고 데이터베이스 작업을 수행할 수 있는 다양한 방법이 있습니다. 응용 프로그램 설계는 응용 프로그램 코드가 구성된 방식, 작업이 수행되는 순서와 시기 등의 측면에서 달라질 수 있습니다. 선택하는 방법은 응용 프로그램 개발의 용이성과 이후 업데이트에서 응용 프로그램을 수정하는 용이성에도 영향을 미칠 수 있습니다. 또한 사용자의 관점에서 응용 프로그램의 성능에도 영향을 미칠 수 있습니다.

미리 채워진 데이터베이스 배포

Adobe AIR 1.0 이상

AIR 로컬 SQL 데이터베이스를 응용 프로그램에서 사용하는 경우 응용 프로그램은 특정 구조의 테이블, 열 등이 포함된 데이터베이스를 기대합니다. 일부 응용 프로그램은 데이터베이스 파일에서 특정 데이터가 미리 채워져 있는 것을 기대하기도 합니다. 적합한 데이터베이스 구조를 확보하는 한 가지 방법은 응용 프로그램 코드에서 데이터베이스를 만드는 것입니다. 응용 프로그램은 로드되면 특정 위치에 데이터베이스 파일이 있는지 확인합니다. 데이터베이스 파일이 없으면 응용 프로그램은 일련의 명령을 실행하여 데이터베이스 파일을 만들고 데이터베이스 구조를 만든 다음 초기 데이터로 테이블을 채웁니다.

데이터베이스와 테이블을 만드는 코드는 복잡한 경우가 많습니다. 흔히 이 코드는 응용 프로그램이 설치되어 있는 동안 한 번만 사용되지만 이 코드로 인해 응용 프로그램의 크기와 복잡성이 커집니다. 데이터베이스, 구조 및 데이터를 프로그래밍 방식으로 만드는 대신 미리 채워진 데이터베이스를 응용 프로그램과 함께 배포할 수 있습니다. 미리 정의된 데이터베이스를 배포하려면 응용 프로그램의 AIR 패키지에 데이터베이스 파일을 포함합니다.

AIR 패키지에 포함된 모든 파일과 마찬가지로, 번들된 데이터베이스 파일은 응용 프로그램 디렉토리(File.applicationDirectory 속성이 나타내는 디렉토리)에 설치됩니다. 그러나 이 디렉토리의 파일은 읽기 전용입니다. AIR 패키지의 파일을 "템플릿" 데이터베이스로 사용합니다. 사용자가 처음 응용 프로그램을 실행할 때 원래 데이터베이스 파일을 사용자의 613페이지의 “응용 프로그램 저장소 디렉토리 가리키기”(또는 다른 위치)에 복사하고 이 데이터베이스를 응용 프로그램에서 사용합니다.

로컬 SQL 데이터베이스 작업을 위한 최상의 방법

Adobe AIR 1.0 이상

다음 목록은 로컬 SQL 데이터베이스로 작업할 때 응용 프로그램의 성능, 보안 및 유지 관리 용이성을 향상시키는 데 사용할 수 있는 방법을 제안한 것입니다.

데이터베이스 연결 미리 만들기

Adobe AIR 1.0 이상

응용 프로그램이 처음 로드될 때 문을 실행하지 않는 경우에도 SQLConnection 객체를 인스턴스화하고 open() 또는 openAsync() 메서드를 미리 호출하여(처음에 응용 프로그램이 시작된 후 등에) 문이 실행될 때 지연을 방지합니다. 자세한 내용은 662페이지의 “데이터베이스 연결”을 참조하십시오.

데이터베이스 연결 다시 사용

Adobe AIR 1.0 이상

응용 프로그램이 실행되는 동안 내내 특정 데이터베이스에 액세스하는 경우 SQLConnection 인스턴스에 대한 참조를 유지하고, 연결을 닫았다가 다시 여는 대신 응용 프로그램이 실행되는 동안 해당 인스턴스를 다시 사용합니다. 자세한 내용은 662페이지의 “데이터베이스 연결”을 참조하십시오.

비동기 실행 모드 우선 사용

Adobe AIR 1.0 이상

데이터 액세스 코드를 작성할 때 동기 작업을 사용하면 코드가 짧아지고 단순해지는 경우가 많기 때문에 작업을 비동기적보다는 동기적으로 실행하려고 할 수 있습니다. 그러나 685페이지의 “동기 및 비동기 데이터베이스 작업 사용”에서 설명했듯이 동기 작업은 사용자가 응용 프로그램을 사용할 때 느낄 수 있을 정도로 성능에 나쁜 영향을 미칠 수 있습니다. 한 작업을 수행하는 시간은 작업 및 관련된 데이터의 양에 따라 달라집니다. 예를 들어, 한 행만 데이터베이스에 추가하는 SQL INSERT 문은 수천 개의 데이터 행을 검색하는 SELECT 문보다 시간이 적게 걸립니다. 그러나 동기 실행을 사용하여 여러 작업을 수행하는 경우 작업이 대개 함께 연결됩니다. 한 작업을 수행하는 시간이 매우 짧다고 해도 모든 동기 작업이 완료될 때까지 응용 프로그램이 고정됩니다. 따라서 서로 연결된 여러 작업의 누적 시간이 응용 프로그램을 정지시킬 정도가 될 수 있습니다.

특히 많은 수의 행이 포함된 작업에서는 비동기 작업을 표준 방법으로 사용합니다. 676페이지의 “SELECT 결과 중 일부 검색”에서 설명한 것처럼 많은 SELECT 문 결과 집합의 처리를 분할하는 방법이 있습니다. 그러나 이 방법은 비동기 실행 모드에서만 사용할 수 있습니다. 비동기 프로그래밍을 사용하여 특정 기능을 수행할 수 없는 경우, 응용 프로그램의 사용자가 경험할 성능 영향을 고려한 경우, 응용 프로그램을 테스트하여 응용 프로그램의 성능이 받는 영향을 파악한 경우에만 동기 작업을 사용합니다. 비동기 실행을 사용하면 코딩이 복잡해질 수 있습니다. 그러나 코드는 한 번만 작성하지만 응용 프로그램의 사용자는 빠르거나 느리게 코드를 반복적으로 사용해야 함을 명심하십시오.

많은 경우에 실행될 각 SQL 문에 대해 별도의 SQLStatement 인스턴스를 사용하면 여러 SQL 작업이 한 번에 큐에 저장될 수 있으므로 코드가 작성된 방식의 측면에서 비동기 코드가 동기 코드와 유사해집니다. 자세한 내용은 688페이지의 "비동기 실행 모드 이해"를 참조하십시오.

별도의 SQL 문을 사용하고 SQLStatement의 text 속성을 변경하지 않기

Adobe AIR 1.0 이상

응용 프로그램에서 두 번 이상 실행되는 SQL 문의 경우 SQL 문마다 별도의 SQLStatement 인스턴스를 만듭니다. SQL 명령이 실행될 때마다 해당 SQLStatement 인스턴스를 사용합니다. 예를 들어, 여러 번 수행되는 네 가지 SQL 작업이 포함된 응용 프로그램을 빌드하는 경우 SQLStatement 인스턴스를 네 개 만들고 각 문의 execute() 메서드를 호출하여 인스턴스를 실행합니다. 모든 SQL 문에 하나의 SQLStatement 인스턴스를 사용하고 SQL 문을 실행하기 전에 매번 text 속성을 다시 정의하는 방법은 사용하지 마십시오.

문 매개 변수 사용

Adobe AIR 1.0 이상

SQLStatement 매개 변수를 사용하고 사용자 입력을 문 텍스트로 연결하지 않습니다. 매개 변수를 사용하면 SQL 삽입 공격의 가능성이 방지되기 때문에 응용 프로그램의 보안이 강화되고 쿼리에서 SQL 리터럴 값만 사용하는 대신 객체를 사용할 수 있습니다. 또한 문이 실행될 때마다 문을 다시 컴파일할 필요 없이 다시 사용할 수 있기 때문에 문이 더 효율적으로 실행됩니다. 자세한 내용은 666페이지의 "문에서 매개 변수 사용"을 참조하십시오.

41장: 바이트 배열 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

ByteArray 클래스를 사용하면 기본적으로 바이트 배열인 이진 데이터 스트림에 대해 읽기/쓰기 작업을 수행할 수 있습니다. 이 클래스는 가장 기본적인 수준에서 데이터에 액세스하는 방법을 제공합니다. 컴퓨터 데이터는 바이트 또는 8비트 그룹으로 구성되기 때문에 바이트 단위로 데이터를 읽을 수 있다는 것은 클래스 및 액세스 방법이 존재하지 않는 데이터에 액세스할 수 있음을 의미합니다. ByteArray 클래스를 사용하면 네트워크를 통해 이동하는 비트맵에서 데이터 스트림에 이르기까지 모든 데이터 스트림을 바이트 수준에서 파싱할 수 있습니다.

writeObject() 메서드를 사용하면 직렬화된 AMF(Action Message Format)로 객체를 ByteArray에 쓸 수 있으며 readObject() 메서드를 사용하면 직렬화된 객체를 ByteArray에서 원본 데이터 유형의 변수로 읽어 올 수 있습니다. 표시 목록에 배치할 수 있는 객체인 표시 객체를 제외한 모든 객체를 직렬화할 수 있습니다. 사용자 정의 클래스를 런타임에 사용할 수 있는 경우 직렬화된 객체를 사용자 정의 클래스 인스턴스에 다시 할당할 수도 있습니다. 객체를 AMF로 변환한 후에는 네트워크 연결을 통해 효율적으로 전송하거나 파일에 저장할 수 있습니다.

여기에 설명된 샘플 Adobe® AIR® 응용 프로그램은 바이트 스트림을 처리하는 예로 .zip 파일을 읽습니다. 이때 해당 응용 프로그램은 .zip 파일에 포함된 파일 목록을 추출하여 데스크톱에 씁니다.

기타 도움말 항목

flash.utils.ByteArray

flash.utils.IExternalizable

액션 메시지 형식 사양

ByteArray 읽기 및 쓰기

Flash Player 9 이상, Adobe AIR 1.0 이상

ByteArray 클래스는 flash.utils 패키지의 일부입니다. ActionScript 3.0에서 ByteArray 객체를 만들려면 다음 예제와 같이 ByteArray 클래스를 가져오고 생성자를 호출합니다.

```
import flash.utils.ByteArray;
var stream:ByteArray = new ByteArray();
```

ByteArray 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

의미 있는 모든 데이터 스트림은 원하는 정보를 분석하여 찾을 수 있는 형식으로 구성됩니다. 예를 들어 간단한 직원 파일의 레코드에는 ID 번호, 이름, 주소, 전화 번호 등이 포함됩니다. 또한 MP3 오디오 파일에는 다운로드 중인 파일의 제목, 제작자, 앨범, 제작 날짜 및 장르를 식별하는 ID3 태그가 포함됩니다. 이러한 형식을 사용하면 데이터 스트림에 있는 데이터의 예상 순서를 알 수 있으며 바이트 스트림을 지능적으로 읽을 수 있습니다.

ByteArray 클래스에는 데이터 스트림에 대한 읽기 및 쓰기 작업을 원활하게 하는 여러 메서드가 포함되어 있습니다. 이러한 메서드에는 readBytes() 및 writeBytes(), readInt() 및 writeInt(), readFloat() 및 writeFloat(), readObject() 및 writeObject(), readUTFBytes() 및 writeUTFBytes() 등이 있습니다. 이러한 메서드를 사용하면 데이터 스트림에서 특정 데이터 유형의 변수로 데이터를 읽어 오고 특정 데이터 유형에서 이진 데이터 스트림으로 데이터를 직접 쓸 수 있습니다.

예를 들어 다음 코드에서는 간단한 문자열 및 부동 소수점 숫자 배열을 읽고 각 요소를 ByteArray에 씁니다. 배열 구성을 사용하면 코드에서 적절한 ByteArray 메서드(writeUTFBytes() 및 writeFloat())를 호출하여 데이터를 쓸 수 있습니다. 반복되는 데이터 패턴으로 인해 루프를 사용하여 배열을 읽을 수 있습니다.

```
 // 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
```

position 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

position 속성은 읽거나 쓰는 동안 ByteArray를 인덱싱하는 포인터의 현재 위치를 저장합니다. position 속성의 초기 값은 다음 코드와 같이 0입니다.

```
var bytes:ByteArray = new ByteArray();
trace("bytes.position is initially: " + bytes.position); // 0
```

ByteArray에 대해 읽기/쓰기 작업을 수행할 때에는 사용하는 메서드를 통해 position 속성이 읽거나 쓴 마지막 바이트 바로 다음의 위치를 가리키도록 업데이트됩니다. 예를 들어 다음 코드에서는 ByteArray에 문자열을 씁니다. 그러면 이후에 position 속성이 ByteArray의 문자열 바로 다음에 오는 바이트를 가리키게 됩니다.

```
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
```

마찬가지로 읽기 작업은 읽은 바이트 수만큼 position 속성을 증가시킵니다.

```
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!
```

position 속성을 ByteArray의 특정 위치로 설정하여 해당 오프셋에서 읽거나 쓸 수 있습니다.

bytesAvailable 및 length 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

length 및 bytesAvailable 속성은 ByteArray의 길이 및 현재 위치에서 끝까지 배열에 남아 있는 바이트 수를 알려 줍니다. 다음 예제에서는 이러한 속성을 사용하는 방법을 보여 줍니다. 이 예제에서는 ByteArray에 텍스트 문자열을 쓴 다음 문자 "a"가 발생하거나 끝(bytesAvailable <= 0)에 도달할 때까지 한 번에 한 바이트씩 ByteArray를 읽습니다.

```
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
}
```

endian 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

저장하는 데 1바이트가 넘는 메모리가 필요한 멀티바이트 숫자를 저장하는 방법은 컴퓨터에 따라 다를 수 있습니다. 예를 들어 정수는 4바이트 또는 32비트의 메모리를 사용할 수 있습니다. 일부 컴퓨터는 숫자의 최상위 바이트를 가장 낮은 메모리 주소에 먼저 저장하고 다른 컴퓨터는 최하위 바이트를 먼저 저장합니다. 컴퓨터 또는 바이트 순서 지정의 이러한 특성을 **big endian**(최상위 바이트 우선) 또는 **little endian**(최하위 바이트 우선)이라고 합니다. 예를 들어 숫자 0x31323334는 big endian 및 little endian 바이트 순서 지정에 대해 다음과 같이 저장됩니다. 여기서 a0은 4바이트의 가장 낮은 메모리 주소를 나타내고 a3은 가장 높은 메모리 주소를 나타냅니다.

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

ByteArray 클래스의 endian 속성을 사용하면 처리하는 멀티바이트 숫자에 대해 이러한 바이트 순서를 나타낼 수 있습니다. 이 속성에 사용할 수 있는 값은 "bigEndian" 또는 "littleEndian"입니다. Endian 클래스는 이러한 문자열을 사용하여 endian 속성을 설정하기 위한 BIG_ENDIAN 및 LITTLE_ENDIAN 상수를 정의합니다.

compress() 및 uncompress() 메서드

Flash Player 9 이상, Adobe AIR 1.0 이상

compress() 메서드를 사용하면 매개 변수로 지정하는 압축 알고리즘에 따라 ByteArray를 압축할 수 있습니다. uncompress() 메서드를 사용하면 압축 알고리즘에 따라 압축된 ByteArray를 압축 해제할 수 있습니다. compress() 및 uncompress()를 호출한 후에는 바이트 배열의 길이가 새 길이로 설정되며 position 속성은 끝으로 설정됩니다.

CompressionAlgorithm 클래스는 압축 알고리즘을 지정하는 데 사용할 수 있는 상수를 정의합니다. ByteArray 클래스는 deflate(AIR만 해당), zlib 및 lzma 알고리즘을 지원합니다. zlib 압축 데이터 형식에 대한 설명은 http://www.ietf.org/rfc/rfc1950.txt를 참조하십시오. lzma 알고리즘은 Flash Player 11.4 및 AIR 3.4에서 추가되었습니다. 이 알고리즘에 대한 설명은 http://www.7-zip.org/7z.html을 참조하십시오.

deflate 압축 알고리즘은 zlib, gzip 및 일부 zip 구현과 같은 여러 압축 형식에 사용됩니다. deflate 압축 알고리즘에 대한 설명은 http://www.ietf.org/rfc/rfc1951.txt를 참조하십시오.

다음 예제에서는 lzma 알고리즘을 사용하여 bytes라는 ByteArray를 압축합니다.

```
bytes.compress(CompressionAlgorithm.LZMA);
```

다음 예제에서는 deflate 알고리즘을 사용하여 압축된 ByteArray를 압축 해제합니다.

```
bytes.uncompress(CompressionAlgorithm.LZMA);
```

객체 읽기 및 쓰기

Flash Player 9 이상, Adobe AIR 1.0 이상

readObject() 및 writeObject() 메서드는 직렬화된 AMF(Action Message Format)로 인코딩된 ByteArray에 대해 객체 읽기/쓰기 작업을 수행합니다. AMF는 Adobe에서 만들어 Netstream, NetConnection, NetStream, LocalConnection 및 공유 객체를 비롯한 다양한 ActionScript 3.0 클래스에 사용되는 고유 메시지 프로토콜입니다.

1바이트 유형 표시자는 뒤에 오는 인코딩된 데이터의 유형을 설명합니다. AMF는 다음의 13가지 데이터 유형을 사용합니다.

```
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
```

표시자가 null, true 또는 false와 같이 가능한 단일 값을 나타내지 않는 한(이 경우 다른 값이 인코딩되지 않음) 유형 표시자 뒤에 인코딩된 데이터가 옵니다.

AMF에는 AMF0과 AMF3의 두 가지 버전이 있습니다. AMF0은 참조를 통한 복잡한 객체 전송 작업을 지원하며 끝점에서의 객체 관계 복원을 허용합니다. AMF3은 참조를 통해 객체 참조 외에 객체 특징 및 문자열을 보내고 ActionScript 3.0에서 도입된 새로운 데이터 유형을 지원하여 AMF0을 개선합니다. ByteArray.objectEcoding 속성은 객체 데이터를 인코딩하는 데 사용되는 AMF의 버전을 지정합니다. flash.net.ObjectEncoding 클래스는 AMF 버전을 지정하는 상수인 ObjectEncoding.AMF0 및 ObjectEncoding.AMF3을 정의합니다.

다음 예제에서는 writeObject()를 호출하여 ByteArray에 XML 객체를 쓴 다음 Deflate 알고리즘을 사용하여 ByteArray를 압축하고 데스크톱의 order 파일에 씁니다. 이 예제에서는 작업 완료 시 레이블을 사용하여 "Wrote order file to desktop!"이라는 메시지를 AIR 윈도우에 표시합니다.

```
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();
    }
}
```

readObject() 메서드는 직렬화된 AMF의 객체를 ByteArray에서 읽어 지정된 유형의 객체에 저장합니다. 다음 예제에서는 order 파일을 데스크톱에서 ByteArray(inBytes)로 읽어 와 압축 해제한 다음 readObject()를 호출하여 XML 객체 orderXML에 저장합니다. 이 예제에서는 for each() 루프 구문를 통해 텍스트 영역에 각 노드를 추가하여 표시합니다. 또한 이 예제에서는 objectEncoding 속성의 값을 order 파일 내용의 헤더와 함께 표시합니다.

```
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();
    }
}
```

ByteArray 예제: .zip 파일 읽기

Adobe AIR 1.0 이상

이 예제에서는 다양한 유형의 여러 파일이 포함된 간단한 .zip 파일을 읽는 방법을 보여 줍니다. 이 예제에서는 각 파일을 ByteArray에 압축 해제하고 데스크톱에 쓰며 각 파일에 대한 메타데이터에서 관련 데이터를 추출합니다.

.zip 파일의 일반 구조는 http://www.pkware.com/documents/casestudies/APPNOTE.TXT에서 유지 관리되는 PKWARE Inc. 사양을 기반으로 합니다. .zip 아카이브의 첫 번째 파일에 대한 파일 헤더와 파일 데이터가 먼저 오고 그 다음에는 각 추가 파일에 대한 파일 헤더와 파일 데이터 쌍이 옵니다. 파일 헤더의 구조는 나중에 설명하겠습니다. 다음으로 .zip 파일에는 선택적으로 데이터 설명자 레코드가 포함됩니다(일반적으로 출력 zip 파일이 디스크에 저장된 것이 아니라 메모리에 만들어진 경우). 그 다음으로는 아카이브 암호 해독 헤더, 아카이브 추가 데이터 레코드, 중앙 디렉토리 구조, 중앙 디렉토리 레코드의 Zip64 끝, 중앙 디렉토리 로케이터의 Zip64 끝 및 중앙 디렉토리 레코드의 끝과 같은 여러 추가 선택적 요소가 옵니다.

이 예제의 코드는 폴더가 포함되지 않은 zip 파일을 파싱하기 위한 용도로만 작성되었으며 데이터 설명자 레코드를 필요로 하지 않습니다. 마지막 파일 데이터 다음에 오는 모든 정보는 무시됩니다.

각 파일에 대한 파일 헤더의 형식은 다음과 같습니다.

파일 헤더 서명	4바이트
필요한 버전	2바이트
일반 용도의 비트 플래그	2바이트
압축 방법	2바이트(8=DEFLATE, 0=압축 해제)
가장 최근에 파일을 수정한 시간	2바이트
가장 최근에 파일을 수정한 날짜	2바이트
crc-32	4바이트
압축된 크기	4바이트
압축 해제된 크기	4바이트
파일 이름 길이	2바이트

추가 필드 길이	2바이트
파일 이름	variable
추가 필드	variable

파일 헤더 다음에는 실제 파일 데이터가 오며 이 데이터는 압축 방법 플래그에 따라 압축하거나 압축 해제할 수 있습니다. 파일 데이터를 압축 해제하는 경우 플래그는 0이며 DEFLATE 알고리즘을 사용하여 데이터를 압축하는 경우 8, 기타 압축 알고리즘의 경우 다른 값입니다.

이 예제에 대한 사용자 인터페이스는 레이블 및 텍스트 영역(taFiles)으로 구성됩니다. 응용 프로그램은 .zip 파일에서 발견하는 각 파일에 대해 파일 이름, 압축된 크기 및 압축 해제된 크기와 같은 정보를 텍스트 영역에 씁니다. 다음 MXML 문서는 Flex 버전 응용 프로그램의 사용자 인터페이스를 정의합니다.

```
<?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>
```

프로그램의 시작 부분에서는 다음 작업을 수행합니다.

- 필요한 클래스 가져오기

```
import flash.filesystem.*;
import flash.utils.ByteArray;
import flash.events.Event;
```

- 사용자 인터페이스 정의 정의

```
 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);
```

- bytes ByteArray 정의

```
var bytes:ByteArray = new ByteArray();
```

- 파일 헤더의 메타데이터를 저장할 변수 정의

```
// 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;
```

- .zip 파일을 나타낼 File(zfile) 및 FileStream(zStream) 객체 정의 및 파일을 추출할 .zip 파일(데스크톱 디렉토리에서 "HelloAIR.zip"이라는 파일)의 위치 지정

```
// File variables for accessing .zip file
var zfile:File = File.desktopDirectory.resolvePath("HelloAIR.zip");
var zStream:FileStream = new FileStream();
```

Flex에서 프로그램 코드는 init() 메서드에서 시작하며 이 메서드는 루트 mx:WindowedApplication 태그에 대한 creationComplete 핸들러로 호출됩니다.

```
// for Flex
private function init():void
{
```

이 프로그램은 READ 모드로 .zip 파일을 열어 작업을 시작합니다.

```
zStream.open(zfile, FileMode.READ);
```

그런 다음 bytes의 endian 속성을 LITTLE_ENDIAN으로 설정하여 숫자 필드의 바이트 순서에서 최하위 바이트가 먼저 나옴을 나타냅니다.

```
bytes.endian = Endian.LITTLE_ENDIAN;
```

다음으로 while() 문은 파일 스트림의 현재 위치가 파일 크기보다 크거나 같을 때까지 계속되는 루프를 시작합니다.

```
while (zStream.position < zfile.size)
{
```

루프 내의 첫 번째 문은 파일 스트림의 첫 30바이트를 ByteArray bytes로 읽어 옵니다. 첫 30바이트는 첫 번째 파일 헤더의 고정 크기 부분을 구성합니다.

```
    // read fixed metadata portion of local file header
    zStream.readBytes(bytes, 0, 30);
```

다음으로 이 코드는 30바이트 헤더의 첫 번째 바이트에서 정수(signature)를 읽습니다. ZIP 형식 정의는 모든 파일 헤더의 서명이 16진수 값 0x04034b50임을 지정합니다. 서명이 다른 경우 코드가 .zip 파일의 파일 부분 밖으로 이동했으며 더 이상 추출할 파일이 없음을 나타냅니다. 이 경우 코드는 바이트 배열의 끝을 기다리는 대신 while 루프를 즉시 종료합니다.

```
    bytes.position = 0;
    signature = bytes.readInt();
    // if no longer reading data files, quit
    if (signature != 0x04034b50)
    {
        break;
    }
```

코드의 다음 부분은 오프셋 위치 8에서 헤더 바이트를 읽고 해당 값을 compMethod 변수에 저장합니다. 이 바이트에는 이 파일을 압축하는 데 사용된 압축 방법을 나타내는 값이 포함됩니다. 여러 가지 압축 방법이 허용되지만 실제로는 거의 모든 .zip 파일이 DEFLATE 압축 알고리즘을 사용합니다. 현재 파일이 DEFLATE 압축으로 압축된 경우 compMethod는 8이고 파일이 압축 해제된 경우 compMethod는 0입니다.

```
    bytes.position = 8;
    compMethod = bytes.readByte();  // store compression method (8 == Deflate)
```

첫 30바이트 다음에는 파일 이름과 추가 필드가 포함될 수 있는 헤더의 가변 길이 부분이 옵니다. offset 변수는 이 부분의 크기를 저장합니다. 크기는 오프셋 26 및 28의 헤더에서 읽은 파일 이름 길이 및 추가 필드 길이를 더하여 계산됩니다.

```
        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
```

다음으로 이 프로그램은 offset 변수에 저장된 바이트 수에 대해 파일 헤더의 가변 길이 부분을 읽습니다.

```
        // read variable length bytes between fixed-length header and compressed file data
        zStream.readBytes(bytes, 30, offset);
```

이 프로그램은 헤더의 가변 길이 부분에서 파일 이름을 읽어 파일의 압축된 크기 및 압축 해제된(원본) 크기와 함께 텍스트 영역에 표시합니다.

```
// 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';
```

이 예제에서는 파일의 나머지 부분을 파일 스트림에서 압축된 크기로 지정된 길이에 대한 bytes로 읽어 와 첫 30바이트의 파일 헤더를 덮어씁니다. 파일이 압축되지 않은 경우에도 압축된 크기는 정확합니다. 이는 압축된 크기가 파일의 압축 해제된 크기와 같기 때문입니다.

```
    // 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);
```

다음으로 이 예제에서는 압축된 파일을 압축 해제하고 outfile() 함수를 호출하여 출력 파일 스트림에 씁니다. 이 예제에서는 outfile()에 파일 이름과 파일 데이터가 포함된 바이트 배열을 전달합니다.

```
        if (compMethod == 8) // if file is compressed, uncompress
        {
            bytes.uncompress(CompressionAlgorithm.DEFLATE);
        }
        outFile(fileName, bytes);   // call outFile() to write out the file
```

앞서 언급한 예제에서 bytes.uncompress(CompressionAlgorithm.DEFLATE)는 AIR 응용 프로그램에서만 작동합니다. AIR 및 Flash Player에 대해 Deflate 방식의 데이터 압축을 수행하려면 ByteArray의 inflate() 함수를 호출해야 합니다.

닫는 괄호는 outFile() 메서드를 제외하고 while 루프, init() 메서드 및 Flex 응용 프로그램 코드의 끝을 나타냅니다. while 루프의 시작 부분으로 실행 작업이 다시 돌아가며 다른 파일을 추출하거나 마지막 파일이 처리된 경우 .zip 파일 처리를 끝내 .zip 파일의 다음 바이트 처리를 계속합니다.

```
    } // end of while loop
} // for Flex version, end of init() method and application
```

outfile() 함수는 데스크톱에서 WRITE 모드로 출력 파일을 열고 filename 매개 변수를 통해 제공된 이름을 해당 파일에 지정합니다. 그런 다음 이 함수는 data 매개 변수에서 출력 파일 스트림(outStream)으로 파일 데이터를 쓰고 파일을 닫습니다.

```
// 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();
}
```

42장: 네트워킹 및 통신의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 또는 AIR에서 응용 프로그램을 제작할 때 응용 프로그램 외부의 리소스에 액세스해야 하는 경우가 많습니다. 예를 들어 인터넷 웹 서버에 이미지 요청을 보내고 이미지 데이터를 받을 수 있습니다. 또는 응용 프로그램 서버에 대한 소켓 연결을 통해 직렬화된 객체를 주고 받을 수 있습니다. Flash Player 및 AIR API는 응용 프로그램이 이러한 교환에 참여할 수 있도록 하는 여러 클래스를 제공합니다. 이러한 API는 UDP, TCP, HTTP, RTMP, RTMFP 등의 프로토콜에 대한 IP 기반 네트워킹을 지원합니다.

네트워크를 통해 데이터를 주고받기 위해 사용할 수 있는 클래스는 다음과 같습니다.

클래스	지원되는 데이터 형식	프로토콜	설명
Loader	SWF, PNG, JPEG, GIF	HTTP, HTTPS	지원되는 데이터 형식을 로드하고 해당 데이터를 표시 객체로 변환합니다. 자세한 내용은 178페이지의 "표시 내용을 동적으로 로드"를 참조하십시오.
URLLoader	모두(텍스트, XML, 이진 등)	HTTP, HTTPS	임의의 데이터 형식을 로드합니다. 응용 프로그램에서 데이터 해석을 담당합니다. 자세한 내용은 742페이지의 "URLLoader 클래스 사용"를 참조하십시오.
FileReference	모두	HTTP	파일을 업로드 및 다운로드합니다. 자세한 내용은 595페이지의 "FileReference 클래스 사용"을 참조하십시오.
NetConnection	비디오, 오디오, AMF(ActionScript Message Format)	HTTP, HTTPS, RTMP, RTMFP	비디오, 오디오 및 원격 객체 스트림에 연결합니다. 자세한 내용은 429페이지의 "비디오를 사용한 작업"을 참조하십시오.
Sound	오디오	HTTP	지원되는 오디오 형식을 로드하고 재생합니다. 자세한 내용은 402페이지의 "외부 사운드 파일 로드"를 참조하십시오.
XMLSocket	XML	TCP	XMLSocket 서버와 XML 메시지를 교환합니다. 자세한 내용은 730페이지의 "XML 소켓"을 참조하십시오.
Socket	모두	TCP	TCP 소켓 서버에 연결합니다. 자세한 내용은 726페이지의 "이진 클라이언트 소켓"을 참조하십시오.
SecureSocket(AIR)	모두	TCP(SSLv3 또는 TLSv1 사용)	SSL 또는 TLS 보안이 필요한 TCP 소켓 서버에 연결합니다. 자세한 내용은 726페이지의 "보안 클라이언트 소켓(AIR)"을 참조하십시오.
ServerSocket(AIR)	모두	TCP	들어오는 TCP 소켓 연결에 대한 서버 역할을 합니다. 자세한 내용은 734페이지의 "서버 소켓"을 참조하십시오.
DatagramSocket(AIR)	모두	UDP	UDP 패킷을 보내고 받습니다. 자세한 내용은 736페이지의 "UDP 소켓(AIR)"을 참조하십시오.

웹 응용 프로그램을 만들 때 사용자의 응용 프로그램 상태에 대한 영구 정보를 저장하는 것이 유용한 경우가 많습니다. HTML 페이지 및 응용 프로그램은 이를 위해 일반적으로 쿠키를 사용합니다. Flash Player에서는 이를 위해 SharedObject 클래스를 사용할 수 있습니다. 자세한 내용은 639페이지의 "공유 객체"를 참조하십시오. AIR 응용 프로그램에서 SharedObject 클래스를 사용할 수 있지만 데이터를 일반 파일에 저장할 때 적용되는 제한이 적습니다.

Flash Player 또는 AIR 응용 프로그램이 동일한 컴퓨터에 있는 다른 Flash Player 또는 AIR 응용 프로그램과 통신해야 하는 경우 LocalConnection 클래스를 사용할 수 있습니다. 예를 들어 동일한 웹 페이지에 있는 두 개(또는 그 이상)의 SWF가 서로 통신할 수 있습니다. 마찬가지로 웹 페이지에서 실행되는 SWF는 AIR 응용 프로그램과 통신할 수 있습니다. 자세한 내용은 756페이지의 "다른 Flash Player 및 AIR 인스턴스와 통신"을 참조하십시오.

로컬 컴퓨터에 있는 SWF 이외의 프로세스와 통신해야 하는 경우 AIR 2에 추가된 NativeProcess클래스를 사용할 수 있습니다. NativeProcess 클래스는 AIR 응용 프로그램이 다른 응용 프로그램을 실행하고 함께 통신할 수 있도록 합니다. 자세한 내용은 762페이지의 "AIR의 기본 프로세스와 통신"을 참조하십시오.

AIR 응용 프로그램이 실행되는 컴퓨터의 네트워크 환경에 대한 정보가 필요한 경우 다음과 같은 클래스를 사용할 수 있습니다.

- NetworkInfo - 컴퓨터의 IP 주소와 같은 사용 가능한 네트워크 인터페이스에 대한 정보를 제공합니다. 자세한 내용은 720페이지의 "네트워크 인터페이스"를 참조하십시오.
- DNSResolver - DNS 레코드를 조회할 수 있도록 합니다. 자세한 내용은 723페이지의 "DNS(Domain Name System) 레코드"를 참조하십시오.
- ServiceMonitor - 서버의 가용성 여부를 모니터링할 수 있도록 합니다. 자세한 내용은 721페이지의 "서비스 모니터링"을 참조하십시오.
- URLMonitor - 특정 URL에서 리소스의 가용성을 모니터링할 수 있도록 합니다. 자세한 내용은 722페이지의 "HTTP 모니터링"을 참조하십시오.
- SocketMonitor 및 SecureSocketMonitor - 소켓에서 리소스의 가용성을 모니터링할 수 있도록 합니다. 자세한 내용은 722페이지의 "소켓 모니터링"을 참조하십시오.

중요한 개념 및 용어

네트워킹 및 통신 코드를 프로그래밍할 때 사용되는 중요한 용어가 아래 참조 목록에 정리되어 있습니다.

외부 데이터 응용 프로그램 외부의 일부 양식에 저장되는 데이터로 필요할 때 응용 프로그램에 로드됩니다. 이 데이터는 직접 로드한 파일에 저장하거나, 서버에서 실행되는 스크립트나 프로그램을 호출하여 가져온 데이터베이스나 다른 양식에 저장할 수 있습니다.

URL 인코딩된 변수 URL 인코딩 형식을 사용하면 여러 가지 변수(변수 이름과 값의 쌍)를 하나의 텍스트 문자열로 나타낼 수 있습니다. 개별 변수는 name=value 형식으로 작성되고, 각 변수(각 이름-값 쌍)는 앰퍼샌드 문자로 구분됩니다(예: variable1=value1&variable2=value2). 이러한 방법으로 무제한의 변수를 하나의 메시지로 보낼 수 있습니다.

MIME 유형 인터넷 통신에서 지정된 파일 유형을 확인하는 데 사용되는 표준 코드입니다. 모든 지정된 파일 유형에는 해당 유형을 식별하는 데 사용되는 특정 코드가 있습니다. 컴퓨터(예: 웹 서버 또는 사용자의 Flash Player 또는 AIR 인스턴스)에서 파일 또는 메시지를 보낼 때는 보내는 파일 유형을 지정합니다.

HTTP Hypertext Transfer Protocol의 약어로 인터넷을 통해 전송되는 웹 페이지 및 다양한 유형의 내용을 전달하기 위한 표준 포맷입니다.

요청 메서드 AIR 응용 프로그램이나 웹 브라우저 같은 응용 프로그램에서 메시지(HTTP 요청이라고 함)를 웹 서버에 보낼 때 전송되는 모든 데이터는 GET과 POST의 두 가지 요청 메서드 중 한 가지 방식을 사용하여 요청에 포함될 수 있습니다. 서버 끝에서 요청을 받는 프로그램은 데이터를 찾기 위해 요청의 적절한 부분을 조회해야 하므로, 응용 프로그램에서 데이터를 보내는 데 사용되는 요청 메서드는 서버에서 해당 데이터를 읽을 때 사용되는 요청 메서드와 일치해야 합니다.

소켓 연결 두 컴퓨터 간의 통신을 위한 영구 연결입니다.

업로드 파일을 다른 컴퓨터로 보냅니다.

다운로드 다른 컴퓨터에서 파일을 가져옵니다.

네트워크 인터페이스

Adobe AIR 2 이상

NetworkInfo 객체를 사용하면 응용 프로그램에서 사용할 수 있는 하드웨어 및 소프트웨어 네트워크 인터페이스를 검색할 수 있습니다. NetworkInfo 객체는 단일 객체이므로 만들 필요가 없습니다. 대신 정적 클래스 속성인 networkInfo를 사용하여 단일 NetworkInfo 객체에 액세스하십시오. NetworkInfo 객체는 또한 사용 가능한 인터페이스 중 하나가 변경되는 경우에도 networkChange 이벤트를 전달합니다.

NetworkInterface 객체 목록을 가져오려면 findInterfaces() 메서드를 호출합니다. 목록에 있는 각 NetworkInterface 객체는 사용 가능한 인터페이스 중 하나를 설명합니다. NetworkInterface 객체는 IP 주소, 하드웨어 주소, 최대 전송 단위 및 인터페이스의 활성 여부와 같은 정보를 제공합니다.

다음 코드 예제에서는 클라이언트 컴퓨터의 각 인터페이스에 대한 NetworkInterface 속성을 추적합니다.

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

자세한 내용은 다음 항목을 참조하십시오.

- NetworkInfo

- NetworkInterface
- InterfaceAddress
- Flexpert: Flex 4.5로 네트워크 연결 유형 감지

네트워크 연결 변경

Adobe AIR 1.0 이상

AIR 응용 프로그램은 네트워크 연결이 확실하지 않고 변경되는 환경에서 실행될 수 있습니다. 응용 프로그램에서 온라인 리소스에 대한 연결을 관리하는 것을 지원하기 위해 Adobe AIR은 네트워크 연결을 사용할 수 있게 되거나 사용할 수 없게 될 때마다 네트워크 변경 이벤트를 보냅니다. NetworkInfo 객체 및 응용 프로그램의 NativeApplication 객체는 모두 networkChange 이벤트를 전달합니다. 이 이벤트에 응답하려면 리스너를 추가합니다.

```
NetworkInfo.networkInfo.addEventListener(Event.NETWORK_CHANGE, onNetworkChange);
```

그런 다음 이벤트 핸들러 함수를 정의합니다.

```
function onNetworkChange(event:Event)
{
    //Check resource availability
}
```

networkChange 이벤트는 모든 네트워크 활동의 변경을 나타내지는 않으며 개별 네트워크 연결이 변경된 것만 나타냅니다. AIR에서는 네트워크 변경의 의미를 해석하려고 하지 않습니다. 네트워크에 연결된 컴퓨터에는 많은 실제 연결과 가상 연결이 있을 수 있으므로 연결이 손실되어도 리소스가 반드시 손실되지는 않습니다. 반면에 새로운 연결은 향상된 리소스 가용성을 보장하지 않습니다. 새 연결이 이전에 사용할 수 있던 리소스에 대한 액세스를 차단하는 경우도 있습니다(예: VPN에 연결하는 경우).

일반적으로 응용 프로그램에서 원격 리소스에 연결할 수 있는지 확인하는 유일한 방법은 연결을 시도하는 것입니다. 서비스 모니터링 프레임워크는 지정된 호스트에 대한 네트워크 연결이 변경되는 경우 적용할 수 있는 이벤트 기반 응답 방법을 제공합니다.

참고: 서비스 모니터링 프레임워크에서는 서버가 요청에 적절하게 응답하는지 여부를 감지합니다. 검사가 성공하더라도 연결이 완전히 보장되는 것은 아닙니다. 확장 가능한 웹 서비스에서는 흔히 캐싱 및 로드 균형 조정 도구를 사용하여 트래픽을 웹 서버의 클러스터에 리디렉션합니다. 이 상황에서 서비스 공급자는 네트워크 연결의 부분적 진단만 제공합니다.

서비스 모니터링

Adobe AIR 1.0 이상

AIR 프레임워크와는 별도의 서비스 모니터링 프레임워크가 aircore.swc 파일에 있습니다. 이 프레임워크를 사용하려면 aircore.swc 파일이 제작 과정에 포함되어야 합니다.

Adobe® Flash® Builder에서는 이 라이브러리를 자동으로 포함합니다.

ServiceMonitor 클래스는 네트워크 서비스를 모니터링하기 위한 프레임워크를 구현하고 서비스 모니터에 대한 기본 기능을 제공합니다. 기본적으로 ServiceMonitor 클래스의 인스턴스는 네트워크 연결과 관련된 이벤트를 전달합니다. ServiceMonitor 객체는 인스턴스가 만들어질 때 그리고 런타임에서 네트워크 변경이 감지될 때 이러한 이벤트를 전달합니다. 또한 ServiceMonitor 인스턴스의 pollInterval 속성을 설정하여 일반적인 네트워크 연결 이벤트와 관계없이 지정된 간격(밀리초)으로 연결을 확인할 수 있습니다. ServiceMonitor 객체는 start() 메서드가 호출될 때까지 네트워크 연결을 확인하지 않습니다.

ServiceMonitor 클래스의 하위 클래스인 URLMonitor 클래스는 지정된 URLRequest의 HTTP 연결의 변경을 감지합니다.

역시 ServiceMonitor 클래스의 하위 클래스인 SocketMonitor 클래스는 지정된 포트에서 지정된 호스트에 대한 연결의 변경을 감지합니다.

참고: AIR 2 이전 버전에서는 서비스 모니터링 프레임워크가 servicemonitor.swc 라이브러리에서 제작되었습니다. 이 라이브러리는 현재 사용되지 않습니다. 대신 aircore.swc 라이브러리를 사용합니다.

Flash CS4 및 CS5 Professional

Adobe® Flash® CS4 또는 CS5 Professional에서 이러한 클래스를 사용하려면 다음 작업을 수행합니다.

1. [파일] > [제작 설정] 명령을 선택합니다.
2. ActionScript 3.0에 대한 [설정] 버튼을 클릭합니다. [라이브러리 경로]를 선택합니다.
3. [SWC 파일 찾아보기] 버튼을 클릭하고 Flash Professional 설치 폴더에서 AIK 폴더를 찾습니다.
4. 이 폴더에서 /frameworks/libs/air/aircore.swc(AIR 2의 경우) 또는 /frameworks/libs/air/servicemonitor.swc(AIR 1.5의 경우)를 찾습니다.
5. [확인] 버튼을 클릭합니다.
6. 다음 import 문을 ActionScript 3.0 코드에 추가합니다.

```
import air.net.*;
```

Flash CS3 Professional

Adobe® Flash® CS3 Professional에서 이러한 클래스를 사용하려면 ServiceMonitorShim 구성 요소를 [구성 요소] 패널에서 [라이브러리]로 드래그하고 다음 import 문을 ActionScript 3.0 코드에 추가합니다.

```
import air.net.*;
```

HTTP 모니터링

Adobe AIR 1.0 이상

URLMonitor 클래스는 포트 80(HTTP 통신을 위한 일반적인 포트)에서 지정된 주소에 대한 HTTP 요청을 수행할 수 있는지 확인합니다. 다음 코드에서는 URLMonitor 클래스의 인스턴스를 사용하여 Adobe 웹 사이트에 대한 연결 변경을 감지합니다.

```
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);
}
```

소켓 모니터링

Adobe AIR 1.0 이상

AIR 응용 프로그램에서는 푸쉬 모델 연결을 위해 소켓 연결도 사용할 수 있습니다. 일반적으로 방화벽과 네트워크 라우터는 보안상의 이유로 권한이 없는 포트에서 네트워크 통신을 제한합니다. 이 때문에 개발자는 사용자가 소켓 연결을 설정할 수 없는 경우를 고려해야 합니다.

다음 코드에서는 SocketMonitor 클래스의 인스턴스를 사용하여 소켓 연결에 대한 연결 변경을 감지합니다. 모니터링되는 포트는 IRC에 대한 일반 포트인 6667입니다.

```
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);
}
```

소켓 서버에 보안 연결이 필요한 경우 SocketMonitor 대신에 SecureSocketMonitor 클래스를 사용할 수 있습니다.

DNS(Domain Name System) 레코드

Adobe AIR 2.0 이상

DNSResolver 클래스를 사용하여 DNS 리소스 레코드를 조회할 수 있습니다. DNS 리소스 레코드는 도메인 이름의 IP 주소 및 IP 주소의 도메인 이름과 같은 정보를 제공합니다. 다음과 같은 DNS 리소스 레코드의 유형을 조회할 수 있습니다.

- ARecord - 호스트의 IPv4 주소입니다.
- AAAARecord - 호스트의 IPv6 주소입니다.
- MXRecord - 호스트의 메일 교환 레코드입니다.
- PTRRecord - IP 주소의 호스트 이름입니다.
- SRVRecord - 서비스에 대한 서비스 레코드입니다.

레코드를 조회하려면 쿼리 문자열과 레코드 유형을 나타내는 클래스 객체를 DNSResolver 객체의 lookup() 메서드에 전달합니다. 사용할 쿼리 문자열은 레코드 유형에 따라 달라집니다.

레코드 클래스	쿼리 문자열	예제 쿼리 문자열
ARecord	호스트 이름	“example.com”
AAAARecord	호스트 이름	“example.com”
MXRecord	호스트 이름	“example.com”
PTRRecord	IP 주소	“208.77.188.166”
SRVRecord	Service 식별자: _service._protocol.host	“_sip._tcp.example.com”

다음 코드 예제에서는 “example.com” 호스트의 IP 주소를 조회합니다.

```
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 );
        }
    }
}
```

자세한 내용은 다음 항목을 참조하십시오.

- DNSResolver
- DNSResolverEvent
- ARecord
- AAAARecord
- MXRecord
- PTRRecord
- SRVRecord

43장: 소켓

Flash Player 9 이상, Adobe AIR 1.0 이상

소켓은 두 컴퓨터 프로세스 간에 설정되는 네트워크 연결 유형입니다. 일반적으로 동일한 IP(인터넷 프로토콜) 네트워크에 연결된 서로 다른 두 컴퓨터에서 프로세스가 실행됩니다. 그러나 특수한 "로컬 호스트" IP 주소를 사용하면 동일한 컴퓨터에서도 연결된 프로세스를 실행할 수 있습니다.

Adobe Flash Player는 클라이언트측 TCP(전송 제어 프로토콜) 소켓을 지원합니다. Flash Player 응용 프로그램은 소켓 서버 역할을 하는 다른 프로세스에 연결할 수 있지만 다른 프로세스에서 들어오는 연결 요청은 받을 수 없습니다. 즉, Flash Player 응용 프로그램은 TCP 서버에 연결할 수 있지만 TCP 서버 역할은 수행할 수 없습니다.

또한 Flash Player API에도 XMLSocket 클래스가 포함되어 있습니다. XMLSocket 클래스는 Flash Player 전용 프로토콜을 사용하여 해당 프로토콜을 인식하는 서버와 XML 메시지를 교환할 수 있도록 합니다. XMLSocket 클래스는 ActionScript 1에서 도입된 이후 계속 지원되어 이전 버전과의 호환성을 제공합니다. Flash XMLSocket과 통신하기 위해 특별히 만든 서버에 연결하는 경우가 아니라면 Socket 클래스는 일반적으로 새 응용 프로그램에 사용되어야 합니다.

Adobe AIR에는 소켓 기반 네트워크 프로그래밍을 위한 몇 가지 클래스가 추가되었습니다. AIR 응용 프로그램은 ServerSocket 클래스를 사용하여 TCP 소켓 서버 역할을 수행할 수 있으며 SecureSocket 클래스를 사용하여 SSL 또는 TLS 보안이 필요한 소켓 서버에 연결할 수 있습니다. 또한 AIR 응용 프로그램은 DatagramSocket 클래스를 사용하여 UDP(Universal Datagram Protocol) 메시지를 주고 받을 수 있습니다.

기타 도움말 항목

flash.net 패키지

972페이지의 “소켓 연결”

TCP 소켓

Flash Player 9 이상, Adobe AIR 1.0 이상

TCP(Transmission Control Protocol)는 영구 네트워크 연결에서 메시지를 교환할 수 있는 방법을 제공합니다. TCP를 사용할 경우, 심각한 네트워크 문제만 없다면 전송된 모든 메시지가 올바른 순서로 도착합니다. TCP 연결에는 "클라이언트"와 "서버"가 필요합니다. Flash Player에서 클라이언트 소켓을 만들 수 있습니다. 또한 Adobe AIR에서는 서버 소켓을 만들 수 있습니다.

다음 ActionScript API는 TCP 연결을 제공합니다.

- Socket - 클라이언트 응용 프로그램이 서버에 연결할 수 있도록 해 줍니다. Socket 클래스는 들어오는 연결을 수신할 수 없습니다.
- SecureSocket(AIR) - 클라이언트 응용 프로그램이 신뢰할 수 있는 서버에 연결하고 암호화된 통신에 참여할 수 있도록 해 줍니다.
- ServerSocket(AIR) - 응용 프로그램이 들어오는 연결을 수신하고 서버 역할을 할 수 있도록 해 줍니다.
- XMLSocket - 클라이언트 응용 프로그램이 XMLSocket 서버에 연결할 수 있도록 해 줍니다.

이진 클라이언트 소켓

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 메시지 교환이 클라이언트와 서버로 제한되지 않는다는 점을 제외하면 이진 소켓 연결도 XML 소켓과 비슷합니다. 대신 연결할 때 이진 정보로 데이터를 전송할 수 있으므로 메일 서버(POP3, SMTP 및 IMAP) 및 뉴스 서버(NNTP)를 비롯한 다양한 서비스에 연결할 수 있습니다.

Socket 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

Socket 클래스는 소켓 연결을 만들고 원시 이진 데이터를 읽고 쓸 수 있도록 하며 이진 프로토콜을 사용하는 서버와 상호 운용하는 데 유용합니다. 이진 소켓 연결을 사용하여 POP3, SMTP, IMAP 및 NNTP 같은 여러 가지 인터넷 프로토콜과 상호 작용하는 코드를 작성할 수 있습니다. 이러한 상호 작용으로 응용 프로그램에서 메일 및 뉴스 서버에 연결할 수 있습니다.

Flash Player에서는 서버의 이진 프로토콜을 사용하여 해당 서버와 직접 통신할 수 있습니다. 일부 서버에서는 big-endian 바이트 순서를 사용하는 반면 다른 서버에서는 little-endian 바이트 순서를 사용합니다. "네트워크 바이트 순서"가 big-endian이므로 인터넷에서 대부분의 서버는 big-endian 바이트 순서를 사용합니다. little-endian 바이트 순서가 널리 사용되는 이유는 Intel® x86 아키텍처에서 사용되기 때문입니다. endian 바이트 순서를 선택할 때는 데이터를 보내거나 받는 서버의 바이트 순서와 일치하는 endian 바이트 순서를 사용해야 합니다. IDataInput 및 IDataOutput 인터페이스에서 수행되는 모든 연산과 그러한 인터페이스(ByteArray, Socket 및 URLStream)를 구현하는 클래스는 기본적으로 big-endian 형식으로 인코딩됩니다. 즉, 가장 중요한 바이트가 앞에 옵니다. 이 기본 바이트 순서를 선택한 것은 Java와 공식 네트워크 바이트 순서를 일치시키기 위한 것입니다. big-endian 바이트 순서를 사용할지 또는 little-endian 바이트 순서를 사용할지 변경하려면 endian 속성을 Endian.BIG_ENDIAN 또는 Endian.LITTLE_ENDIAN으로 설정할 수 있습니다.

Socket 클래스는 IDataInput 및 IDataOutput 인터페이스(flash.utils 패키지에 있음)에서 정의된 모든 메서드를 상속합니다. 소켓에서 쓰기 및 읽기를 수행하려면 이러한 메서드를 사용해야 합니다.

자세한 내용은 다음 항목을 참조하십시오.

- Socket
- IDataInput
- IDataOutput
- socketData 이벤트

보안 클라이언트 소켓(AIR)

Adobe AIR 2 이상

SecureSocket 클래스를 사용하여 SSLv4(Secure Sockets Layer 버전 4) 또는 TLSv1(Transport Layer Security 버전 1)을 사용하는 소켓 서버에 연결할 수 있습니다. 보안 소켓은 서버 인증, 데이터 무결성 및 메시지 기밀성의 세 가지 이점을 제공합니다. 런타임은 서버 인증서와 루트에 대한 관계 또는 사용자의 신뢰 저장소에 있는 중간 인증 기관 인증서를 사용하여 서버를 인증합니다. 런타임은 SSL 및 TLS 프로토콜 구현에서 사용되는 암호화 알고리즘을 사용하여 데이터 무결성 및 메시지 기밀성을 제공합니다.

SecureSocket 객체를 사용하여 서버에 연결할 때 런타임은 인증서 신뢰 저장소를 사용하여 서버 인증서의 유효성을 검사합니다. Windows와 Mac의 경우 운영 체제에서 신뢰 저장소를 제공합니다. Linux의 경우 런타임에서 자체 신뢰 저장소를 제공합니다.

서버 인증서가 유효하지 않거나 신뢰할 수 없는 경우 런타임이 ioError 이벤트를 전달합니다. SecureSocket 객체의 serverCertificateStatus 속성을 보면 유효성 검사의 실패 원인을 확인할 수 있습니다. 신뢰할 수 있는 유효한 인증서가 없는 서버와 통신할 때는 저장소가 제공되지 않습니다.

CertificateStatus 클래스는 다음과 같이 가능한 유효성 검사 결과를 나타내는 문자열 상수를 정의합니다.

- Expired - 인증서 만료 날짜가 지났습니다.
- Invalid - 여러 가지 이유로 인증서가 유효하지 않을 수 있습니다. 예를 들어 인증서가 수정 또는 손상되었거나 인증서 유형이 올바르지 않을 수 있습니다.
- Invalid chain - 서버의 인증서 체인에 있는 하나 이상의 인증서가 유효하지 않습니다.
- Principal mismatch - 서버의 호스트 이름 및 인증서 공통 이름이 일치하지 않습니다. 즉, 서버에서 잘못된 인증서를 사용하고 있습니다.
- Revoked - 인증서 발행 기관이 인증서를 취소했습니다.
- Trusted - 인증서가 유효하고 신뢰할 수 있습니다. SecureSocket 객체는 신뢰할 수 있는 유효한 인증서를 사용하는 서버에만 연결할 수 있습니다.
- Unknown - 아직 SecureSocket 객체의 인증서 유효성을 검사하지 않았습니다. connect()를 호출하기 전이나 connect 또는 ioError 이벤트가 전달되기 전에는 serverCertificateStatus 속성이 이 상태 값을 가집니다.
- Untrusted signers - 인증서가 클라이언트 컴퓨터의 신뢰 저장소에 있는 신뢰할 수 있는 루트 인증서에 "연결"되지 않았습니다.

SecureSocket 객체와 통신하려면 서버에서 보안 프로토콜을 사용하고, 서버에 신뢰할 수 있는 유효한 인증서가 있어야 합니다. 이 점 외에는 SecureSocket 객체를 사용하는 것과 Socket 객체를 사용하는 것이 동일합니다.

SecureSocket 객체는 일부 플랫폼에서 지원되지 않습니다. 런타임이 현재 클라이언트 컴퓨터에서 SecureSocket 객체 사용을 지원하는지 여부를 테스트하려면 SecureSocket 클래스의 isSupported 속성을 사용하십시오.

자세한 내용은 다음 항목을 참조하십시오.

- SecureSocket
- CertificateStatus
- IDataInput
- IDataOutput
- socketData 이벤트

TCP 소켓 예제: Telnet 클라이언트 구축

Flash Player 9 이상, Adobe AIR 1.0 이상

이 Telnet 예제에서는 Socket 클래스를 사용하여 원격 서버와 연결하고 데이터를 전송하는 기술을 보여 줍니다. 이 예제는 다음과 같은 기술에 대해 설명합니다.

- Socket 클래스를 사용하여 사용자 정의 Telnet 클라이언트 만들기
- ByteArray 객체를 사용하여 원격 서버로 텍스트 보내기
- 원격 서버에서 수신한 데이터 처리

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. Telnet 응용 프로그램 파일은 Samples/Telnet 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
TelnetSocket.fla 또는 TelnetSocket.mxml	Flex(MXML) 또는 Flash(FLA)용 사용자 인터페이스로 구성된 기본 응용 프로그램 파일입니다.
TelnetSocket.as	사용자 인터페이스 논리를 제공하는 문서 클래스입니다(Flash 전용).
com/example/programmingas3/Telnet/Telnet.as	원격 서버에 연결, 데이터 송수신 및 표시 등 응용 프로그램에 대한 Telnet 클라이언트 기능을 제공합니다.

Telnet 소켓 응용 프로그램 개요

Flash Player 9 이상, Adobe AIR 1.0 이상

기본 TelnetSocket.mxml 파일은 전체 응용 프로그램의 UI(사용자 인터페이스)를 만드는 역할을 합니다.

이 파일은 UI 외에도 login()과 sendCommand()라는 두 가지 메서드를 정의하여 지정한 서버에 사용자를 연결합니다.

다음 코드는 기본 응용 프로그램 파일의 ActionScript를 보여 줍니다.

```
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 = "";
}
```

코드의 첫 번째 줄은 사용자 정의 com.example.programmingas.socket 패키지에서 Telnet 클래스를 가져옵니다. 코드의 두 번째 줄은 Telnet 클래스의 인스턴스(telnetClient)를 선언하며 이는 이후 connect() 메서드를 통해 초기화됩니다. 그 다음에 connect() 메서드를 선언하고 앞에서 선언한 telnetClient 변수를 초기화합니다. 이 메서드는 사용자가 지정한 Telnet 서버 이름, Telnet 서버 포트 및 표시 목록의 TextArea 구성 요소에 대한 참조를 전달합니다. 표시 목록은 소켓 서버에서 텍스트 응답을 표시하는 데 사용됩니다. connect() 메서드의 마지막 두 줄은 패널의 title 속성을 설정하고 패널 구성 요소를 사용하도록 설정하여 사용자가 원격 서버에 데이터를 전송할 수 있도록 합니다. 기본 응용 프로그램 파일의 마지막 메서드인 sendCommand()는 사용자의 명령을 원격 서버에 ByteArray 객체로 전송하는 데 사용됩니다.

Telnet 클래스 개요

Flash Player 9 이상, Adobe AIR 1.0 이상

Telnet 클래스는 원격 Telnet 서버에 연결하고 데이터를 송수신하는 작업을 담당합니다.

Telnet 클래스는 다음과 같은 전용 변수를 선언합니다.

```
private var serverURL:String;
private var portNumber:int;
private var socket:Socket;
private var ta:TextArea;
private var state:int = 0;
```

첫 번째 변수인 serverURL에는 연결할 사용자 지정 서버 주소가 포함되어 있습니다.

두 번째 변수인 portNumber는 현재 실행 중인 Telnet 서버의 포트 번호입니다. 기본적으로 Telnet 서비스는 포트 23에서 실행됩니다.

세 번째 변수인 socket은 serverURL과 portNumber 변수에서 정의한 서버에 연결을 시도할 Socket 인스턴스입니다.

네 번째 변수인 ta는 스테이지의 TextArea 구성 요소 인스턴스에 대한 참조입니다. 이 구성 요소는 원격 Telnet 서버의 응답이나 오류 메시지를 표시하는 데 사용됩니다.

마지막 변수인 state는 Telnet 클라이언트에서 지원하는 옵션을 결정하는 데 사용되는 숫자 값입니다.

앞에서 보듯이 Telnet 클래스의 생성자 함수는 기본 응용 프로그램 파일의 connect() 메서드에서 호출합니다.

Telnet 생성자는 server, port 및 output이라는 세 가지 매개 변수를 사용합니다. server 및 port 매개 변수는 Telnet 서버를 실행하는 서버 이름과 포트 번호를 지정합니다. 마지막 매개 변수인 output은 서버 출력을 사용자에게 표시하는 스테이지의 TextArea 구성 요소 인스턴스에 대한 참조입니다.

```
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();
    }
}
```

소켓에 데이터 기록

Flash Player 9 이상, Adobe AIR 1.0 이상

소켓 연결에 데이터를 쓰려면 Socket 클래스에서 아무 쓰기 메서드나 호출합니다. 이러한 쓰기 메서드에는 writeBoolean(), writeByte(), writeBytes(), writeDouble() 등이 있습니다. 그런 다음 flush() 메서드를 사용하여 출력 버퍼의 데이터를 플러시합니다. Telnet 서버에서는 바이트 배열을 매개 변수로 가져와 출력 버퍼로 전송하는 writeBytes() 메서드를 사용하여 소켓 연결에 데이터를 기록합니다. writeBytesToSocket() 메서드는 다음과 같습니다.

```
public function writeBytesToSocket(ba:ByteArray):void
{
    socket.writeBytes(ba);
    socket.flush();
}
```

이 메서드는 기본 응용 프로그램 파일의 sendCommand() 메서드에 의해 호출됩니다.

소켓 서버의 메시지 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

소켓 서버에서 메시지를 수신하거나 이벤트가 발생할 때마다 사용자 정의 msg() 메서드가 호출됩니다. 이 메서드는 문자열을 스테이지의 TextArea에 추가하고 사용자 정의 setScroll() 메서드를 호출하여 TextArea 구성 요소가 아래쪽까지 스크롤되도록 합니다. msg() 메서드는 다음과 같습니다.

```
private function msg(value:String):void
{
    ta.text += value;
    setScroll();
}
```

TextArea 구성 요소의 내용을 자동으로 스크롤하지 않은 경우 사용자가 서버의 최신 응답을 보려면 텍스트 영역에서 스크롤 막대를 수동으로 드래그해야 합니다.

TextArea 구성 요소 스크롤

Flash Player 9 이상, Adobe AIR 1.0 이상

setScroll() 메서드에는 TextArea 구성 요소의 내용을 세로로 스크롤하는 ActionScript의 한 행이 포함되어 있으므로 사용자는 반환된 텍스트의 마지막 행을 볼 수 있습니다. 다음 코드 예제는 setScroll() 메서드를 보여 줍니다.

```
public function setScroll():void
{
    ta.verticalScrollPosition = ta.maxVerticalScrollPosition;
}
```

이 메서드는 verticalScrollPosition 속성을 설정합니다. 이 속성은 현재 표시된 문자의 맨 위 행에 대한 행 번호이며 이 번호를 maxVerticalScrollPosition 속성 값으로 설정합니다.

XML 소켓

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 소켓을 사용하면 명시적으로 닫힐 때까지 열린 상태로 유지되는 원격 서버에 대한 연결을 만들 수 있습니다. 서버와 클라이언트 간에 XML과 같은 문자열 데이터를 교환할 수 있습니다. XML 소켓 서버 사용 시 얻을 수 있는 이점은 클라이언트가 명시적으로 데이터를 요청할 필요가 없다는 것입니다. 서버가 요청을 기다리지 않고 데이터를 전송할 수 있으며, 연결된 모든 클라이언트에 데이터를 전송할 수 있습니다.

Flash Player에서, 그리고 응용 프로그램 샌드박스 외부의 Adobe AIR 내용에서 XML 소켓 연결을 사용하려면 대상 서버에 소켓 정책 파일이 있어야 합니다. 자세한 내용은 956페이지의 “웹 사이트 컨트롤(정책 파일)” 및 972페이지의 “소켓 연결”을 참조하십시오.

RTMP(Real-Time Messaging Protocol) 프로토콜과 달리, XMLSocket에는 HTTP 터널링 기능이 없기 때문에 XMLSocket 클래스는 자동으로 방화벽을 통해 터널링할 수 없습니다. HTTP 터널링을 사용해야 할 경우 Flash Remoting 또는 Flash Media Server(RTMP 지원)를 대신 사용하십시오.

Flash Player의 내용 또는 응용 프로그램 보안 샌드박스 외부의 AIR 응용 프로그램 내용에서 XMLSocket 객체를 사용하여 서버에 연결할 수 있는 방법과 경우에는 다음과 같은 제한 사항이 적용됩니다.

- 응용 프로그램 보안 샌드박스 외부에 있는 내용의 경우 XMLSocket.connect() 메서드는 1024 이상의 TCP 포트 번호에만 연결할 수 있습니다. 이 제한 사항 때문에 XMLSocket 객체와 통신하는 서버 데몬 역시 1024 이상의 포트 번호에 할당되어야 합니다. 1024보다 작은 포트 번호는 주로 FTP (21), Telnet (23), SMTP (25), HTTP (80), POP3 (110) 등의 시스템 서비스에 사용되므로 보안 문제 때문에 XMLSocket 객체가 이러한 포트를 사용하지 못하도록 차단됩니다. 이렇게 포트 번호를 제한하면 포트 번호가 잘못 사용될 가능성이 줄어듭니다.

- 응용 프로그램 보안 샌드박스 외부에 있는 내용의 경우 XMLSocket.connect() 메서드는 내용이 있는 동일한 도메인의 컴퓨터에만 연결할 수 있습니다. 이 제한 사항은 URLLoader.load()의 보안 규칙과 동일합니다. 내용이 있는 도메인 이외의 도메인에서 실행되는 서버 데몬에 연결하려면 해당 서버에 특정 도메인에서의 액세스를 허용하는 크로스 도메인 정책 파일을 만들면 됩니다. 크로스 도메인 정책 파일에 대한 자세한 내용은 979페이지의 "AIR 보안"을 참조하십시오.

참고: XMLSocket 객체와 통신할 서버를 설정하는 작업은 그리 간단하지 않습니다. 응용 프로그램에서 실시간 상호 작용이 필요하지 않은 경우에는 XMLSocket 클래스 대신 URLLoader 클래스를 사용합니다.

XMLSocket 클래스의 XMLSocket.connect() 및 XMLSocket.send() 메서드를 사용하여 소켓 연결을 통해 서버와 XML을 주고받을 수 있습니다. XMLSocket.connect() 메서드는 웹 서버 포트와 소켓 연결을 설정하고, XMLSocket.send() 메서드는 소켓 연결에 지정된 서버에 XML 객체를 전달합니다.

XMLSocket.connect() 메서드를 호출할 때 응용 프로그램은 서버에 대한 TCP/IP 연결을 열고 다음 중 하나가 발생할 때까지 해당 연결을 열린 상태로 유지합니다.

- XMLSocket 클래스의 XMLSocket.close() 메서드가 호출됩니다.
- XMLSocket 객체에 대한 참조가 더 이상 존재하지 않습니다.
- Flash Player가 종료됩니다.
- 연결이 끊어집니다(예: 모뎀 연결이 끊어짐).

XMLSocket 클래스를 사용하여 서버에 연결

Flash Player 9 이상, Adobe AIR 1.0 이상

소켓 연결을 만들려면 소켓 연결 요청을 기다리고 Flash Player 또는 AIR 응용 프로그램에 응답을 전송하는 서버측 응용 프로그램을 만들어야 합니다. 이 유형의 서버측 응용 프로그램은 AIR 또는 Java, Python, Perl과 같은 다른 프로그래밍 언어로 작성할 수 있습니다. XMLSocket 클래스를 사용하려면 서버 컴퓨터에서 XMLSocket 클래스가 사용하는 단순 프로토콜을 이해하는 데몬을 실행해야 합니다.

- XML 메시지는 전이중 TCP/IP 스트림 소켓 연결을 통하여 전송됩니다.
- 각 XML 메시지는 완전한 XML 문서이며 0바이트로 끝납니다.
- 단일 XMLSocket 연결에서 송수신할 수 있는 XML 메시지의 수에는 제한이 없습니다.

Java XML 소켓 서버 만들기 및 연결

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 코드에서는 들어오는 연결을 허용하고 명령 프롬프트 윈도우에 받은 메시지를 표시하는, Java로 작성된 간단한 XMLSocket 서버를 보여 줍니다. 명령줄에서 서버를 시작할 때 다른 포트 번호를 지정할 수 있지만 기본적으로 새로운 서버는 로컬 컴퓨터의 포트 8080에서 만들어집니다.

새 텍스트 문서를 만들고 다음 코드를 추가합니다.

```
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);
    }
}
```

문서를 하드 디스크에 SimpleServer.java로 저장하고 Java 컴파일러를 사용하여 컴파일하면 SimpleServer.class라는 Java 클래스 파일이 만들어집니다.

명령 프롬프트를 열고 java SimpleServer를 입력하여 XMLSocket 서버를 시작할 수 있습니다. SimpleServer.class 파일은 로컬 컴퓨터나 네트워크의 어디에나 보관할 수 있으므로 반드시 웹 서버의 루트 디렉토리에 넣지 않아도 됩니다.

파일이 Java 클래스 경로에 없어 서버를 시작할 수 없는 경우 java -classpath . SimpleServer로 서버를 시작하십시오.

응용 프로그램에서 XMLSocket에 연결하려면 다음과 같이 호스트 이름과 포트 번호를 전달할 때 XMLSocket 클래스의 새 인스턴스를 만들고 XMLSocket.connect() 메서드를 호출해야 합니다.

```
var xmlsock:XMLSocket = new XMLSocket();
xmlsock.connect("127.0.0.1", 8080);
```

서버에서 데이터를 수신할 때마다 data 이벤트(flash.events.DataEvent.DATA)가 전달됩니다.

```
xmlsock.addEventListener(DataEvent.DATA, onData);
private function onData(event:DataEvent):void
{
    trace("[" + event.type + "] " + event.data);
}
```

XMLSocket 서버에 데이터를 전송하려면 XMLSocket.send() 메서드를 사용하고 XML 객체나 문자열을 전달합니다. Flash Player가 제공된 매개 변수를 String 객체로 변환하고 XMLSocket 서버에 내용을 전송하고 뒤에 0바이트를 붙입니다.

```
xmlsock.send(xmlFormattedData);
```

XMLSocket.send() 메서드는 데이터가 성공적으로 전송되었는지 여부를 나타내는 값을 반환하지 않습니다. 데이터를 보내려고 할 때 오류가 발생하면 IOError 오류가 발생합니다.

XML 소켓 서버에 전송하는 각 메시지는 개행(\n) 문자로 끝나야 합니다.

자세한 내용은 XMLSocket을 참조하십시오.

서버 소켓

Adobe AIR 2 이상

다른 프로세스가 TCP(Transport Control Protocol) 소켓을 사용하여 응용 프로그램에 연결할 수 있도록 허용하려면 ServerSocket 클래스를 사용합니다. 연결 프로세스는 로컬 컴퓨터 또는 다른 네트워크에 연결된 컴퓨터에서 실행될 수 있습니다 . ServerSocket 객체가 연결 요청을 받으면 connect 이벤트를 전달합니다. 이벤트와 함께 전달된 ServerSocketConnectEvent 객체에는 Socket 객체가 포함됩니다. 이 Socket 객체는 이후 다른 프로세스와의 통신에 사용할 수 있습니다.

들어오는 소켓 연결을 수신하려면

1. ServerSocket 객체를 만들고 로컬 포트에 바인딩합니다.
2. connect 이벤트에 대한 이벤트 리스너를 추가합니다.
3. listen() 메서드를 호출합니다.
4. 들어오는 각 연결에 대한 Socket 객체를 제공공하는 connect 이벤트에 응답합니다.

ServerSocket 객체는 close() 메서드가 호출될 때까지 새 연결을 계속해서 수신합니다.

다음 코드 예제에서는 소켓 서버 응용 프로그램을 만드는 방법을 보여 줍니다. 이 예제에서는 포트 8087에서 들어오는 연결을 수신합니다. 연결이 수신되면 예제에서 메시지(“Connected.” 문자열)가 클라이언트 소켓에 전달됩니다. 그런 다음 서버는 수신되는 모든 메시지를 클라이언트에 다시 전달합니다.

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

자세한 내용은 다음 항목을 참조하십시오.

- ServerSocket
- ServerSocketConnectEvent
- Socket

UDP 소켓(AIR)

Adobe AIR 2 이상

UDP(Universal Datagram Protocol)는 상태 비저장 네트워크 연결에서 메시지를 교환할 수 있는 방법을 제공합니다. UDP에서는 메시지가 순서대로 전달되지 않거나 심지어 일부 메시지가 누락될 수도 있습니다. UDP를 사용할 경우 운영 체제의 네트워크 코드에서는 일반적으로 메시지를 마샬링, 추적 및 인식하는 데 시간이 적게 소비됩니다. 따라서 일반적으로 UDP 메시지는 TCP 메시지의 경우보다 더 빨리 대상 응용 프로그램에 도착합니다.

UDP 소켓 통신은 게임에서 위치를 업데이트하거나 음성 채팅 응용 프로그램에서 사운드 패킷을 전달할 때와 같이 실시간 정보를 전송해야 하는 경우에 유용합니다. 이러한 응용 프로그램에서는 일부 데이터 손실도 허용 가능하며, 모든 데이터가 도착하도록 하는 것보다 전송 지연 시간을 짧게 유지하는 것이 중요합니다. 이 외의 거의 모든 다른 용도의 경우 TCP 소켓을 사용하는 것이 더 좋습니다.

AIR 응용 프로그램은 DatagramSocket 및 DatagramSocketDataEvent 클래스에서 UDP 메시지를 보내고 받을 수 있습니다. UDP 메시지를 보내거나 받으려면

1. DatagramSocket 객체를 만듭니다.
2. data 이벤트에 대해 이벤트 리스너를 추가합니다.
3. bind() 메서드를 사용하여 소켓을 로컬 IP 주소 및 포트에 바인딩합니다.
4. send() 메서드를 호출하고 대상 컴퓨터의 IP 주소 및 포트를 전달하여 메시지를 보냅니다.
5. data 이벤트에 응답하여 메시지를 받습니다. 이 이벤트에 대해 전달된 DatagramSocketDataEvent 객체에는 메시지 데이터가 들어 있는 ByteArray 객체가 포함됩니다.

다음 코드 예제에서는 응용 프로그램에서 UDP 메시지를 보내고 받는 방법을 보여 줍니다. 이 예제에서는 “Hello.”라는 문자열이 포함된 단일 메시지를 대상 컴퓨터에 보냅니다. 또한 수신되는 모든 메시지의 내용을 추적합니다.

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

UDP 소켓을 사용할 때는 다음과 같은 사항들을 고려하십시오.

- 데이터의 단일 패킷은 전송자 및 수신자 간의 네트워크 노드 또는 네트워크 인터페이스의 최소 MTU(최대 전송 단위)보다 클 수 없습니다. send() 메서드로 전달된 ByteArray 객체의 모든 데이터는 단일 패킷으로 전송됩니다. TCP의 경우 크기가 큰 메시지는 여러 패킷으로 분할됩니다.
- 전송자와 대상 간 핸드셰이킹은 지원되지 않습니다. 대상이 존재하지 않거나 지정된 포트에 활성 리스너가 없는 경우 오류 없이 메시지가 삭제됩니다.
- connect() 메서드를 사용할 경우 다른 소스에서 보낸 메시지는 무시됩니다. UDP 연결은 간단한 패킷 필터링만 제공합니다. 즉, 대상 주소 및 포트에 수신되고 있는 유효한 프로세스가 없을 수도 있습니다.
- UDP 트래픽은 네트워크 성능을 저하시킬 수 있습니다. 네트워크 정체가 발생할 경우 네트워크 관리자가 서비스 품질 제어 기능을 구현해야 할 수도 있습니다. TCP에는 기본적으로 네트워크 정체 영향을 줄이기 위한 제어 기능이 포함되어 있습니다.

자세한 내용은 다음 항목을 참조하십시오.

- DatagramSocket
- DatagramSocketDataEvent
- ByteArray

IPv6 주소

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 9.0.115.0 이상에서는 IPv6(Internet Protocol version 6)을 지원합니다. IPv6은 128비트 주소를 지원하는 인터넷 프로토콜입니다(이전에 32비트 주소 체계를 지원하는 IPv4를 개선한 새 버전). 네트워크 인터페이스에서 IPv6를 활성화시켜야 합니다. 자세한 내용은 데이터를 호스팅하는 운영 체제의 도움말을 참조하십시오.

호스팅 시스템이 IPv6을 지원하면 다음과 같이 대괄호([])로 묶은 URL에 IPv6 리터럴 주소를 지정할 수 있습니다.

```
[2001:db8:ccc3:ffff:0:444d:555e:666f]
```

Flash Player에서는 다음 규칙에 따라 리터럴 IPv6 값을 반환합니다.

- Flash Player에서는 IPv6 주소에 대해 긴 형식의 문자열을 반환합니다.
- IP 값에는 콜론이 두 개인 약어가 없습니다.

- 16진수는 소문자로만 사용됩니다.
- IPv6 주소는 대괄호([])로 묶습니다.
- 각 주소는 0~4개의 16진수로 출력됩니다. 이때 선행 0은 생략됩니다.
- 다음 예외를 제외하고 모든 0의 주소는 이중 콜론이 아닌 단일 0으로 출력됩니다.

Flash Player에서 반환되는 IPv6 값에는 다음과 같은 예외가 있습니다.

- 지정되지 않은 IPv6 주소(모든 0)는 [::]으로 출력됩니다.
- 루프백 또는 로컬 호스트 IPv6 주소는 [::1]로 출력됩니다.
- IPv4 매핑 주소(IPv6으로 변환된 주소)는 [::ffff:a.b.c.d]로 출력됩니다. 여기서 a.b.c.d는 점으로 구분된 일반적인 IPv4 값입니다.
- IPv4 호환 주소는 [::a.b.c.d]로 출력됩니다. 여기서 a.b.c.d는 점으로 구분된 일반적인 IPv4 값입니다.

44장: HTTP 통신

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe® AIR® 및 Adobe® Flash® Player 응용 프로그램은 데이터, 이미지, 비디오를 로드하고 메시지를 교환하기 위해 HTTP 기반 서버와 통신할 수 있습니다.

기타 도움말 항목

flash.net.URLLoader

flash.net.URLStream

flash.net.URLRequest

flash.net.URLRequestDefaults

flash.net.URLRequestHeader

flash.net.URLRequestMethod

flash.net.URLVariables

외부 데이터 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에는 외부 소스에서 데이터를 로드하기 위한 메커니즘이 있습니다. 외부 소스는 텍스트 파일과 같은 정적 내용 또는 웹 스크립트에서 생성한 동적 내용을 제공할 수 있습니다. 데이터의 형식은 다양한 방법으로 지정할 수 있으며 ActionScript는 데이터의 암호를 해독하고 데이터에 액세스하는 기능을 제공합니다. 데이터를 가져오는 프로세스의 일환으로 데이터를 외부 서버에 보낼 수도 있습니다.

URLRequest 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 데이터를 로드하는 많은 API는 URLRequest 클래스를 사용하여 필요한 네트워크 요청 속성을 정의합니다.

URLRequest 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

보안 샌드박스에 URLRequest 객체의 다음 속성을 설정할 수 있습니다.

속성	설명
contentType	URL 요청과 함께 전송되는 데이터의 MIME 내용 유형입니다. contentType에 아무 값도 설정되어 있지 않으면 값이 application/x-www-form-urlencoded로 전송됩니다.
data	URL 요청과 함께 전송될 데이터를 포함하는 객체입니다.
digest	Adobe® Flash® Player 캐시에 저장(또는 캐시에서 검색)할 서명된 Adobe 플랫폼 구성 요소를 고유하게 식별하는 문자열입니다.
method	GET 또는 POST 작업과 같은 HTTP 요청 메서드입니다. AIR 응용 프로그램 보안 도메인에서 실행되는 내용은 "GET" 또는 "POST" 이외의 문자열을 method 속성으로 지정할 수 있습니다. 모든 HTTP 동사가 허용되며 "GET"이 기본 메서드입니다. 자세한 내용은 979페이지의 "AIR 보안"을 참조하십시오.
requestHeaders	HTTP 요청에 추가할 HTTP 요청 헤더의 배열입니다. Flash Player 및 응용 프로그램 보안 샌드박스 외부에서 실행되는 AIR 내용에서는 일부 헤더를 설정하는 권한이 제한되어 있습니다.
url	요청할 URL을 지정합니다.

AIR에서는 URLRequest 클래스의 추가 속성을 설정할 수 있습니다. 이러한 속성은 응용 프로그램 보안 샌드박스에서 실행되는 AIR 내용에만 사용할 수 있습니다. 응용 프로그램 샌드박스의 내용은 file 및 http와 같은 표준 스킴 외에도 새로운 URL 스킴을 사용하여 URL을 정의할 수 있습니다.

속성	설명
followRedirects	리디렉션을 수행할지(true, 기본값) 아니면 수행하지 않을지(false)를 지정합니다. 이 속성은 AIR 응용 프로그램 샌드박스에서만 지원됩니다.
manageCookies	HTTP 프로토콜 스택에서 이 요청에 대한 쿠키를 관리할지(true, 기본값) 아니면 관리하지 않을지(false)를 지정합니다. 이 속성은 AIR 응용 프로그램 샌드박스에서만 설정할 수 있습니다.
authenticate	이 요청에 대한 인증 요청을 처리할지(true) 여부를 지정합니다. 이 속성은 AIR 응용 프로그램 샌드박스에서만 설정할 수 있습니다. 기본값은 요청을 인증하는 것입니다. 이 경우 서버에서 자격 증명 표시를 요구하면 인증 대화 상자가 표시될 수 있습니다. 또한 URLRequestDefaults 클래스를 사용하여 사용자 이름과 암호를 설정할 수도 있습니다. 자세한 내용은 740페이지의 "URLRequest 기본값 설정(AIR만 해당)"을 참조하십시오.
cacheResponse	이 요청에 대해 응답 데이터를 캐시할지 여부를 지정합니다. 이 속성은 AIR 응용 프로그램 샌드박스에서만 설정할 수 있습니다. 기본값(true)은 응답을 캐시하는 것입니다.
useCache	이 URLRequest에서 데이터를 페치하기 전에 로컬 캐시를 참조할지 여부를 지정합니다. 이 속성은 AIR 응용 프로그램 샌드박스에서만 설정할 수 있습니다. 기본값(true)은 사용 가능한 경우 캐시된 로컬 버전을 사용하는 것입니다.
userAgent	HTTP 요청에 사용할 사용자 에이전트 문자열을 지정합니다.

참고: HTMLLoader 클래스에는 HTMLLoader 객체가 로드한 내용과 관련된 설정에 대한 관련 속성이 있습니다. 자세한 내용은 892페이지의 "HTMLLoader 클래스"를 참조하십시오.

URLRequest 기본값 설정(AIR만 해당)

Adobe AIR 1.0 이상

URLRequestDefaults 클래스를 사용하여 URLRequest 객체의 응용 프로그램별 기본 설정을 정의할 수 있습니다. 예를 들어 다음 코드에서는 manageCookies 및 useCache 속성의 기본값을 설정합니다. 모든 새 URLRequest 객체는 일반적인 기본값 대신 이러한 속성에 지정된 값을 사용합니다.

```
URLRequestDefaults.manageCookies = false;
URLRequestDefaults.useCache = false;
```

참고: URLRequestDefaults 클래스는 Adobe AIR에서 실행되는 내용에 대해서만 정의되며 Flash Player에서는 지원되지 않습니다.

URLRequestDefaults 클래스에는 특정 호스트에 사용할 기본 사용자 이름과 암호를 지정할 수 있는 setLoginCredentialsForHost() 메서드가 포함되어 있습니다. 메서드의 hostname 매개 변수에 정의된 호스트는 "www.example.com"과 같은 도메인이거나 "www.example.com:80"과 같은 도메인과 포트 번호일 수 있습니다. "example.com", "www.example.com" 및 "sales.example.com"은 각각 고유한 호스트로 간주됩니다.

이러한 자격 증명은 서버에서 요구하는 경우에만 사용됩니다. 사용자가 인증 대화 상자 등을 통해 이미 인증한 경우 setLoginCredentialsForHost() 메서드를 호출해도 인증된 사용자가 변경되지 않습니다.

다음 코드에서는 www.example.com에 보내는 요청에 사용되는 기본 사용자 이름 및 암호를 설정합니다.

```
URLRequestDefaults.setLoginCredentialsForHost("www.example.com", "Ada", "love1816$X");
```

URLRequestDefaults 설정은 현재 응용 프로그램 도메인에만 적용되지만, 한 가지 예외가 있습니다. setLoginCredentialsForHost() 메서드에 전달된 자격 증명은 AIR 응용 프로그램 내의 모든 응용 프로그램 도메인에서 발생하는 요청에 사용됩니다.

자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에서 **URLRequestDefaults** 클래스를 참조하십시오.

URI 스킴

Flash Player 9 이상, Adobe AIR 1.0 이상

모든 보안 샌드박스에서 이루어지는 요청에는 다음과 같은 표준 URI 스킴을 사용할 수 있습니다.

http: 및 https:

웹 브라우저에 사용되는 것과 같은 방식으로 표준 인터넷 URL에 사용합니다.

file:

file:은 로컬 파일 시스템에 있는 파일의 URL을 지정하는 데 사용합니다. 예를 들면 다음과 같습니다.

```
file:///c:/AIR Test/test.txt
```

AIR에서는 응용 프로그램 보안 샌드박스에서 실행되는 내용에 대한 URL을 정의할 때 다음과 같은 스킴도 사용할 수 있습니다.

app:

app:는 설치된 응용 프로그램의 루트 디렉토리에 상대적인 경로를 지정하는 데 사용합니다. 예를 들어 다음 경로는 설치된 응용 프로그램 디렉토리의 resources 하위 디렉토리를 가리킵니다.

```
app:/resources
```

ADL(AIR Debug Launcher)을 통해 AIR 응용 프로그램을 실행하는 경우 응용 프로그램 설명자 파일이 포함된 디렉토리가 응용 프로그램 디렉토리입니다.

File.applicationDirectory를 사용하여 만든 File 객체의 URL 및 url 속성은 다음과 같이 app URI 스킴을 사용합니다.

```
var dir:File = File.applicationDirectory;
dir = dir.resolvePath("assets");
trace(dir.url); // app:/assets
```

app-storage:

app-storage:는 응용 프로그램의 데이터 저장소 디렉토리에 상대적인 경로를 지정하는 데 사용합니다. 설치된 각 응용 프로그램 및 사용자에 대해 AIR은 해당 응용 프로그램 관련 데이터를 저장하기에 적절한 장소인 고유한 응용 프로그램 저장소 디렉토리를 만듭니다. 예를 들어 다음 경로는 응용 프로그램 저장소 디렉토리의 settings 하위 디렉토리에 있는 prefs.xml 파일을 가리킵니다.

```
app-storage:/settings/prefs.xml
```

File.applicationStorageDirectory를 사용하여 만든 File 객체의 URL 및 url 속성은 다음과 같이 app-storage URI 스킴을 사용합니다.

```
var prefsFile:File = File.applicationStorageDirectory;
prefsFile = prefsFile.resolvePath("prefs.xml");
trace(dir.prefsFile); // app-storage:/prefs.xml
```

mailto:

navigateToURL() 함수에 전달된 URLRequest 객체에 mailto 스킴을 사용할 수 있습니다. 754페이지의 “다른 응용 프로그램에서 URL 열기”를 참조하십시오.

이러한 URI 스킴 중 하나를 사용하는 URLRequest 객체를 사용하여 FileStream 또는 Sound 객체와 같은 다양한 객체에 대한 URL 요청을 정의할 수 있습니다. AIR에서 실행되는 HTML 내용에서도 이러한 스킴을 사용할 수 있습니다. 예를 들어 img 태그의 src 특성에서 이러한 스킴을 사용할 수 있습니다.

그러나 응용 프로그램 보안 샌드박스에 있는 내용에서만 이러한 AIR 관련 URI 스킴(app: 및 app-storage:)을 사용할 수 있습니다. 자세한 내용은 979페이지의 “AIR 보안”을 참조하십시오.

URL 변수 설정

URL 문자열에 변수를 직접 추가할 수도 있지만 URLVariables 클래스를 사용하여 요청에 필요한 모든 변수를 정의하는 것이 더 쉽습니다.

다음 세 가지 방법으로 URLVariables 객체에 매개 변수를 추가할 수 있습니다.

- URLVariables 생성자 내에서 지정
- URLVariables.decode() 메서드를 사용하여 지정
- URLVariables 객체 자체에 포함된 동적 속성에 따라 지정

다음 예제에서는 세 가지 방법 모두와 URLRequest 객체에 변수를 할당하는 방법을 보여 줍니다.

```
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;
```

URLVariables 생성자 또는 URLVariables.decode() 메서드 내에 변수를 정의할 때는 URI 문자열에서 특별한 의미를 갖는 문자를 URL 인코딩해야 합니다. 예를 들어 매개 변수 이름 또는 값에 앰퍼샌드를 사용하는 경우 앰퍼샌드를 &에서 %26으로 인코딩해야 합니다. 이는 앰퍼샌드가 매개 변수 구분자 역할을 하기 때문입니다. 이를 위해 최상위 수준encodeURIComponent() 함수를 사용할 수 있습니다.

URLLoader 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

URLLoader 클래스를 사용하면 서버에 요청을 보내고 반환된 정보에 액세스할 수 있습니다. 또한 URLLoader 클래스를 사용하면 로컬 파일 액세스가 허용되는 상황(예: Flash Player local-with-filesystem 샌드박스 및 AIR 응용 프로그램 샌드박스)에서 로컬 파일 시스템에 있는 파일에 액세스할 수 있습니다. URLLoader 클래스는 URL에서 텍스트, 이진 데이터 또는 URL 인코딩된 변수 형식으로 데이터를 다운로드합니다. URLLoader 클래스는 complete, httpStatus, ioError, open, progress, securityError 등의 이벤트를 전달합니다.

ActionScript 3.0 이벤트 처리 모델은 ActionScript 2.0 모델과 많은 차이가 있습니다. 2.0 모델의 경우 LoadVars.onData, LoadVars.onHTTPStatus 및 LoadVars.onLoad 이벤트 핸들러를 사용했습니다. ActionScript 3.0에서의 이벤트 처리에 대한 자세한 내용은 113페이지의 “이벤트 처리”를 참조하십시오.

다운로드되는 데이터는 다운로드가 완료될 때까지 사용할 수 없습니다. progress 이벤트를 수신하여 전달되는 다운로드의 진행률(로드된 바이트 및 총 바이트)을 모니터링할 수 있습니다. 그러나 파일이 너무 빠르게 로드되면 progress 이벤트가 전달되지 않을 수 있습니다. 파일이 성공적으로 다운로드되면 complete 이벤트가 전달됩니다. URLLoader dataFormat 속성을 설정하면 데이터를 텍스트, 원시 이진 데이터 또는 URLVariables 객체로 받을 수 있습니다.

URLLoader.load() 메서드 및 URLLoader 클래스 생성자(선택 항목)는 URLRequest 객체인 request 하나만 매개 변수로 사용합니다. URLRequest 객체에는 대상 URL, 요청 메서드(GET 또는 POST), 추가 헤더 정보 및 MIME 유형과 같은 단일 HTTP 요청의 모든 정보가 들어 있습니다.

예를 들어 XML 패킷을 서버측 스크립트에 업로드할 때 다음과 같은 코드를 사용할 수 있습니다.

```
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);
```

앞의 코드 예제에서는 dataXML이라는 XML 문서(서버에 보낼 XML 패킷 포함)를 만듭니다. 예제에서는 먼저 URLRequest contentType 속성을 "text/xml"로 설정하고 XML 문서를 URLRequest data 속성에 할당합니다. 마지막으로, URLLoader 객체를 생성하고 load() 메서드를 사용하여 원격 스크립트로 요청을 보냅니다.

URLStream 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

URLStream 클래스는 데이터가 도착하면 다운로드 중인 데이터에 대한 액세스를 제공합니다. URLStream 클래스를 사용하면 다운로드가 완료되기 전에 스트림을 닫을 수도 있습니다. 다운로드된 데이터는 원시 이진 데이터로 사용할 수 있습니다.

URLStream 객체에서 데이터를 읽을 때는 먼저 bytesAvailable 속성을 통해 충분한 데이터를 사용할 수 있는지 확인합니다. 사용할 수 있는 데이터 이상으로 데이터를 읽으려고 하면 EOFError 예외가 발생합니다.

httpResponseStatus 이벤트(AIR)

Adobe AIR에서 URLStream 클래스는 httpStatus 이벤트 이외에 httpResponseStatus 이벤트를 전달합니다. httpResponseStatus 이벤트가 전달된 후 응답 데이터(있는 경우)가 전달됩니다. HTTPStatusEvent 클래스로 표현된 httpResponseStatus 이벤트에는 응답이 반환된 URL인 responseURL 속성과 응답을 반환한 응답 헤더를 나타내는 URLRequestHeader 객체의 배열인 responseHeaders 속성이 포함됩니다.

외부 문서에서 데이터 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

동적 응용 프로그램을 작성하는 경우 외부 파일 또는 서버 측 스크립트에서 데이터를 로드하는 것이 좋을 수 있습니다. 그러면 응용 프로그램을 편집하거나 다시 컴파일하지 않고도 동적 응용 프로그램을 작성할 수 있습니다. 예를 들어 "오늘의 팁"을 표시하는 응용 프로그램을 작성하는 경우 하루에 한 번씩 데이터베이스에서 임의로 하나의 팁을 가져와 텍스트 파일로 저장하는 서버 측 스크립트를 작성할 수 있습니다. 그러면 응용 프로그램에서 매번 데이터베이스를 쿼리하지 않고 정적 텍스트 파일의 내용을 로드할 수 있습니다.

다음 코드 예제에서는 외부 텍스트 파일인 params.txt의 내용을 로드하는 URLRequest 및 URLLoader 객체를 만듭니다.

```
var request:URLRequest = new URLRequest("params.txt");
var loader:URLLoader = new URLLoader();
loader.load(request);
```

기본적으로 요청 메서드를 정의하지 않으면 Flash Player 및 Adobe AIR에서 HTTP GET 메서드를 사용하여 내용을 로드합니다. POST 메서드를 사용하여 요청을 보내려면 다음 코드와 같이 정적 상수를 사용하여 request.method 속성을 POSTURLRequestMethod.POST로 설정합니다.

```
var request:URLRequest = new URLRequest("sendfeedback.cfm");
request.method = URLRequestMethod.POST;
```

런타임에 로드되는 외부 문서 params.txt에는 다음 데이터가 포함됩니다.

```
monthNames=January,February,March,April,May,June,July,August,September,October,November,December&dayNames
=Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
```

이 파일에는 monthNames 및 dayNames라는 두 개의 매개 변수가 포함됩니다. 각 매개 변수에는 쉼표로 구분되어 문자열로 파싱된 목록이 들어 있습니다. String.split() 메서드를 사용하여 이 목록을 배열로 나눌 수 있습니다.


코드를 읽거나 디버깅하기가 어려워지므로 외부 데이터 파일에서 예약어나 언어 구문을 변수 이름으로 사용하지 마십시오.

데이터가 로드되면 complete 이벤트가 전달되고 다음 코드에서 보듯이 URLLoader의 data 속성에서 외부 문서의 내용을 사용할 수 있게 됩니다.

```
function completeHandler(event)
{
    var loader2 = event.target;
    air.trace(loader2.data);
}
```

원격 문서에 이름-값 쌍이 있으면 다음과 같이 로드된 파일의 내용에서 전달하는 방식으로 URLVariables 클래스를 사용하여 데이터를 파싱할 수 있습니다.

```
private function completeHandler(event:Event):void
{
    var loader2:URLLoader = URLLoader(event.target);
    var variables:URLVariables = new URLVariables(loader2.data);
    trace(variables.dayNames);
}
```

외부 파일의 각 이름-값 쌍은 URLVariables 객체에서 속성으로 만들어집니다. 앞의 코드 샘플에서 변수 객체의 각 속성은 문자열로 처리됩니다. 이름-값 쌍의 값이 항목의 목록이면 다음과 같이 String.split() 메서드를 호출하여 문자열을 배열로 변환할 수 있습니다.

```
var dayNameArray:Array = variables.dayNames.split(",");
```


외부 텍스트 파일에서 숫자 데이터를 로드하는 경우 int(), uint() 또는 Number() 등 최상위 함수를 사용하여 값을 숫자 값으로 변환합니다.

원격 파일의 내용을 문자열로 로드하여 새 URLVariables 객체를 만드는 대신 URLLoader.dataFormat 속성을 URLLoaderDataFormat 클래스에 있는 정적 속성 중 하나로 설정할 수 있습니다. URLLoader.dataFormat 속성에 사용할 수 있는 값은 다음 세 가지입니다.

- URLLoaderDataFormat.BINARY - ByteArray 객체에 저장된 이진 데이터가 URLLoader.data 속성에 포함됩니다.
- URLLoaderDataFormat.TEXT - 문자열 객체의 텍스트가 URLLoader.data 속성에 포함됩니다.
- URLLoaderDataFormat.VARIABLES - URLVariables 객체에 저장된 URL 인코딩된 변수가 URLLoader.data 속성에 포함됩니다.

다음 코드는 URLLoader.dataFormat 속성을 URLLoaderDataFormat.VARIABLES로 설정하는 경우 로드된 데이터가 자동으로 URLVariables 객체로 파싱되는 방법을 보여 줍니다.

```
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);
        }
    }
}
```

참고: URLLoader.dataFormat의 기본값은 URLLoaderDataFormat.TEXT입니다.

다음 예제에 나와 있듯이 외부 파일에서 XML을 로드하는 것은 URLVariables를 로드하는 것과 같습니다. URLRequest 인스턴스와 URLLoader 인스턴스를 만들고 이러한 인스턴스를 사용하여 원격 XML 문서를 다운로드할 수 있습니다. 파일이 완전히 다운로드되면 Event.COMPLETE 이벤트가 전달되고 외부 파일의 내용이 XML 인스턴스로 변환됩니다. 이 인스턴스는 XML 메서드와 속성을 사용하여 파싱할 수 있습니다.

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

외부 스크립트와 통신

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 데이터 파일 로드 외에 URLVariables 클래스를 사용하여 변수를 서버측 스크립트로 전송하고 서버의 응답을 처리할 수도 있습니다. 예를 들어 이 방법은 게임을 프로그래밍하는 경우 사용자의 점수를 서버에 전송하여 고득점 목록에 추가할지 여부를 계산하거나 사용자 로그인 정보를 서버에 전송하여 확인하려는 경우에 유용합니다. 서버측 스크립트는 사용자 이름과 암호를 처리하고, 데이터베이스에서 해당 이름과 암호의 유효성을 검사하며, 사용자가 제공한 자격 증명이 유효한지 여부를 확인하여 반환할 수 있습니다.

다음 코드 예제에서는 variables라는 URLVariables 객체를 만듭니다. 이 객체는 name이라는 새 변수를 만듭니다. 다음으로, 변수를 보낼 서버측 스크립트의 URL을 지정하는 URLRequest 객체를 만듭니다. 그런 다음 URLRequest 객체의 method 속성을 설정하여 변수를 HTTP POST 요청으로 보냅니다. URLVariables 객체를 URL 요청에 추가하려면 URLRequest 객체의 data 속성을 이전에 만든 URLVariables 객체로 설정합니다. 마지막으로 URLLoader 인스턴스가 만들고 URLLoader.load() 메서드를 호출하여 요청을 시작합니다.

```
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);
}
```

다음 코드에는 앞의 예제에서 사용된 Adobe ColdFusion® greeting.cfm 문서의 내용이 포함되어 있습니다.

```
<cfif NOT IsDefined("Form.name") OR Len(Trim(Form.Name)) EQ 0>
    <cfset Form.Name = "Stranger" />
</cfif>
<cfoutput>welcomeMessage=#UrlEncodedFormat("Welcome, " & Form.name)#
</cfoutput>
```

웹 서비스 요청

Flash Player 9 이상, Adobe AIR 1.0 이상

HTTP 기반 웹 서비스에는 여러 가지가 있습니다. 기본적인 종류는 다음과 같습니다.

- REST
- XML-RPC
- SOAP

ActionScript 3에서 웹 서비스를 사용하려면 URLRequest 객체를 만들고 URL 변수 또는 XML 문서를 사용하여 웹 서비스 호출을 만든 후 URLLoader 객체를 통해 서비스로 호출을 보냅니다. Flex 프레임워크에는 웹 서비스를 쉽게 사용할 수 있도록 하는 여러 클래스가 들어 있습니다. 특히 복잡한 SOAP 서비스를 액세스하는 데 유용합니다. Flash Professional CS3부터 Flash Professional로 개발된 응용 프로그램과 Flash Builder로 개발된 응용 프로그램에서 Flex 클래스를 사용할 수 있습니다.

HTML 기반 AIR 응용 프로그램에서는 URLRequest 및 URLLoader 클래스 또는 JavaScript XMLHttpRequest 클래스를 사용할 수 있습니다. 필요한 경우 SWF 라이브러리를 생성하여 Flex 프레임워크의 웹 서비스 구성 요소를 JavaScript 코드에서 사용할 수도 있습니다.

응용 프로그램을 브라우저에서 실행하는 경우 호출 SWF와 동일한 인터넷 도메인에서만 웹 서비스를 사용할 수 있습니다. 단, 웹 서비스를 호스팅하는 서버가 다른 도메인에서의 액세스를 허용하는 크로스 도메인 정책 파일도 호스팅하는 경우는 예외입니다. 크로스 도메인 정책 파일을 사용할 수 없는 경우 자체 서버를 통해 요청을 프록싱하는 방법이 주로 사용됩니다. Adobe Blaze DS와 Adobe LiveCycle은 웹 서비스 프록싱을 지원합니다.

AIR 응용 프로그램에서는 웹 서비스 호출이 응용 프로그램 보안 샌드박스에서 발생한 경우 크로스 도메인 정책 파일이 필요 없습니다. AIR 응용 프로그램 내용은 원격 도메인에서 제공되지 않으므로 크로스 도메인 정책에서 금지하도록 설정한 유형의 공격에 참여할 수 없습니다. HTML 기반 AIR 응용 프로그램에서 응용 프로그램 보안 샌드박스의 내용은 크로스 도메인 XMLHttpRequests를 만들 수 있습니다. 다른 보안 샌드박스의 내용이 iframe에 로드되는 한 해당 내용이 크로스 도메인 XMLHttpRequests를 만들도록 허용할 수 있습니다.

기타 도움말 항목

956페이지의 “웹 사이트 컨트롤(정책 파일)”

Adobe BlazeDS

Adobe LiveCycle ES2

REST 아키텍처

XML-RPC

SOAP 프로토콜

REST 스타일 웹 서비스 요청

Flash Player 9 이상, Adobe AIR 1.0 이상

REST 스타일 웹 서비스는 HTTP 메서드 동사를 사용하여 기본 액션을 지정하고 URL 변수를 통해 액션의 세부 사항을 지정할 수 있습니다. 예를 들어 항목에 대한 데이터를 가져오는 요청은 GET 동사와 URL 변수를 사용하여 메서드 이름과 항목 ID를 지정할 수 있습니다. 결과 URL 문자열은 다음 예처럼 나타날 수 있습니다.

```
http://service.example.com/?method=getItem&id=d3452
```

ActionScript로 REST 스타일 웹 서비스에 액세스하려면 URLRequest, URLVariables 및 URLLoader 클래스를 사용할 수 있습니다. AIR 응용 프로그램 내의 JavaScript 코드에서 XMLHttpRequest를 사용할 수도 있습니다.

ActionScript에서 REST 스타일 웹 서비스 호출을 프로그래밍하는 단계는 일반적으로 다음과 같습니다.

1. URLRequest 객체를 만듭니다.
2. 요청 객체에 대해 서비스 URL 및 HTTP 메서드를 설정합니다.
3. URLVariables 객체를 만듭니다.
4. 서비스 호출 매개 변수를 변수 객체의 동적 속성으로 설정합니다.
5. 변수 객체를 요청 객체의 데이터 속성에 할당합니다.
6. 요청을 URLLoader 객체와 함께 서비스로 보냅니다.
7. 서비스 호출이 완료되었음을 나타내기 위해 URLLoader에서 전달한 complete 이벤트를 처리합니다. 또한 URLLoader 객체에서 전달할 수 있는 다양한 오류 이벤트를 수신하는 것이 좋습니다.

예를 들어 웹 서비스가 호출 매개 변수를 요청자에게 되돌려주는 테스트 메서드를 제공하는 경우 다음과 같은 ActionScript 코드를 사용하여 서비스를 호출할 수 있습니다.

```
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 );
}
```

AIR 응용 프로그램 내의 JavaScript에서는 XMLHttpRequest 객체를 사용하여 동일한 요청을 수행할 수 있습니다.

```
<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 웹 서비스 요청

Flash Player 9 이상, Adobe AIR 1.0 이상

XML-RPC 웹 서비스는 호출 매개 변수를 URL 변수 집합이 아닌 XML 문서로 취합니다. XML-RPC 웹 서비스를 통한 트랜잭션을 수행하려면 올바르게 서식 지정된 XML 메시지를 만들어 HTTP POST 메서드를 사용하여 웹 서비스로 보냅니다. 또한 서버가 요청 데이터를 XML로 취급하도록 요청에 대한 Content-Type 헤더를 설정해야 합니다.

다음 예제에서는 REST 예제에서 보여준 것과 동일한 웹 서비스를 XML-RPC 서비스로 사용하는 방법을 보여 줍니다.

```
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 );
}
```

AIR에서의 WebKit는 E4X 구문을 지원하지 않으므로 앞의 예제에서 XML 문서를 만드는 데 사용한 메서드는 JavaScript 코드에서 작동하지 않습니다. 대신 DOM 메서드를 사용하여 XML 문서를 만들거나 문서를 문자열로 만든 후 JavaScript DOMParser 클래스를 사용하여 해당 문자열을 XML로 변환해야 합니다.

다음 예제에서는 DOM 메서드를 사용하여 XML-RPC 메시지와 XMLHttpRequest를 만들고 웹 서비스 트랜잭션을 수행합니다.

```
<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 웹 서비스 요청

Flash Player 9 이상, Adobe AIR 1.0 이상

SOAP는 일반 XML-RPC 웹 서비스 개념을 기반으로 제작되어 유형이 지정된 데이터를 전송하는 더 복잡하고 풍부한 수단입니다. SOAP 웹 서비스는 일반적으로 통해 웹 서비스 호출, 데이터 유형, 서비스 URL을 지정하는 WSDL(Web Service Description Language) 파일을 제공합니다. ActionScript 3은 SOAP를 직접 지원하지 않지만 SOAP XML 메시지를 수동으로 생성하여 서버에 게시한 후 결과를 파싱할 수 있습니다. 그러나 가장 간단한 SOAP 웹 서비스가 아닌 이상 기존 SOAP 라이브러리를 사용할 때 상당한 개발 시간을 절약할 수 있습니다.

Flex 프레임워크에는 SOAP 웹 서비스에 액세스하기 위한 라이브러리가 들어 있습니다. Flash Builder는 Flex 프레임워크의 일부이므로 해당 라이브러리 rpc.swc를 Flex 프로젝트에 자동으로 포함하고 있습니다. Flash Professional에서는 Flex framework.swc 및 rpc.swc를 프로젝트의 라이브러리 경로에 추가하여 ActionScript로 Flex 클래스에 액세스할 수 있습니다.

기타 도움말 항목

Using the Flex web service component in Flash Professional

Cristophe Coenraets: Real-time Trader Desktop for Android

다른 응용 프로그램에서 URL 열기

Flash Player 9 이상, Adobe AIR 1.0 이상

navigateToURL() 함수를 사용하여 웹 브라우저나 다른 응용 프로그램에서 URL을 열 수 있습니다. AIR에서 실행되는 내용의 경우 navigateToURL() 함수가 기본 시스템 웹 브라우저에서 페이지를 엽니다.

이 함수의 request 매개 변수로 전달하는 URLRequest 객체의 경우 url 속성만 사용됩니다.

navigateToURL() 함수의 첫 번째 매개 변수인 navigate 매개 변수는 URLRequest 객체입니다(739페이지의 "URLRequest 클래스 사용" 참조). 두 번째 매개 변수는 윈도우 이름을 지정할 수 있는 선택적 window 매개 변수입니다. 예를 들어 다음 코드는 www.adobe.com 웹 페이지를 엽니다.

```
var url:String = "http://www.adobe.com";
var urlReq:URLRequest = new URLRequest(url);
navigateToURL(urlReq);
```

참고: navigateToURL() 함수를 사용할 때 런타임은 POST 메서드(해당 method 속성이 URLRequestMethod.POST로 설정된 메서드)를 사용하는 URLRequest 객체가 GET 메서드를 사용하는 것으로 처리합니다.

navigateToURL() 함수를 사용할 때 navigateToURL() 함수를 호출하는 코드의 보안 샌드박스에 따라 URI 스킴이 허용됩니다.

일부 API를 사용하여 웹 브라우저에서 내용을 실행할 수 있습니다. 보안상의 이유로 일부 URI 스킴은 AIR에서 이러한 API를 사용할 때 금지됩니다. 금지된 스킴의 목록은 API를 사용하는 코드의 보안 샌드박스에 따라 달라집니다. 보안 샌드박스에 대한 자세한 내용은 979페이지의 "AIR 보안"을 참조하십시오.

응용 프로그램 샌드박스 (AIR에만 해당)

AIR 응용 프로그램 샌드박스에서 실행 중인 내용에 의해 시작된 URL에서는 모든 URI 스킴을 사용할 수 있습니다. URI 스킴을 처리하려면 응용 프로그램을 등록해야 하고 그렇지 않으면 요청이 이무것도 수행하지 않습니다. 대부분의 컴퓨터와 장치에서는 다음 스킴이 지원됩니다.

- http:
- https:
- file:
- mailto: - AIR에서는 이러한 요청을 등록된 시스템 메일 응용 프로그램에 보냅니다.
- sms: — AIR는 sms: 요청을 기본 텍스트 메시지 응용 프로그램으로 보냅니다. URL 형식은 응용 프로그램이 실행되는 시스템 규칙을 따라야 합니다. 예를 들어 Android에서는 URI 스킴이 소문자여야 합니다.

  ```
  navigateToURL( new URLRequest( "sms:+15555550101") );
  ```

- tel: — AIR는 tel: 요청을 기본 전화 걸기 응용 프로그램으로 보냅니다. URL 형식은 응용 프로그램이 실행되는 시스템 규칙을 따라야 합니다. 예를 들어 Android에서는 URI 스킴이 소문자여야 합니다.

  ```
  navigateToURL( new URLRequest( "tel:5555555555") );
  ```

- market: — AIR는 market: 요청을 일반적으로 Android 장치에서 지원되는 Market 응용 프로그램으로 보냅니다.

```
navigateToURL( new URLRequest( "market://search?q=Adobe Flash") );
navigateToURL( new URLRequest( "market://search?q=pname:com.adobe.flashplayer") );
```

운영 체제에서 허용하는 경우 응용 프로그램은 사용자 정의 URI 스킴을 정의 및 등록할 수 있습니다. AIR에서 응용 프로그램을 시작하기 위해 스킴을 사용하여 URL을 만들 수 있습니다.

원격 샌드박스

다음과 같은 스킴이 허용됩니다. 웹 브라우저에서 사용하듯이 이러한 URL 스킴을 사용합니다.

- http:
- https:
- mailto: - AIR에서는 이러한 요청을 등록된 시스템 메일 응용 프로그램에 보냅니다.

다른 모든 URI 스킴은 금지됩니다.

Local-with-file 샌드박스

다음과 같은 스킴이 허용됩니다. 웹 브라우저에서 사용하듯이 이러한 URL 스킴을 사용합니다.

- file:
- mailto: - AIR에서는 이러한 요청을 등록된 시스템 메일 응용 프로그램에 보냅니다.

다른 모든 URI 스킴은 금지됩니다.

Local-with-networking 샌드박스

다음과 같은 스킴이 허용됩니다. 웹 브라우저에서 사용하듯이 이러한 URL 스킴을 사용합니다.

- http:
- https:
- mailto: - AIR에서는 이러한 요청을 등록된 시스템 메일 응용 프로그램에 보냅니다.

다른 모든 URI 스킴은 금지됩니다.

Local-trusted 샌드박스

다음과 같은 스킴이 허용됩니다. 웹 브라우저에서 사용하듯이 이러한 URL 스킴을 사용합니다.

- file:
- http:
- https:
- mailto: - AIR에서는 이러한 요청을 등록된 시스템 메일 응용 프로그램에 보냅니다.

다른 모든 URI 스킴은 금지됩니다.

45장: 다른 Flash Player 및 AIR 인스턴스와 통신

Flash Player 9 이상, Adobe AIR 1.0 이상

LocalConnection 클래스를 사용하면 Adobe® AIR® 응용 프로그램 간은 물론 브라우저에서 실행되는 SWF 내용 간에도 통신할 수 있습니다. 또한 브라우저에서 실행하는 AIR 응용 프로그램 및 SWF 내용 간에도 통신할 수 있습니다. LocalConnection 클래스를 사용하면 Flash Player 및 AIR 인스턴스 사이에 데이터 공유가 가능한 다양한 용도의 응용 프로그램을 작성할 수 있습니다.

LocalConnection 클래스

Flash Player 9 이상, Adobe AIR 1.0 이상

LocalConnection 클래스를 사용하면 fscommand() 메서드나 JavaScript를 사용하지 않고 다른 SWF 파일에 명령을 보낼 수 있는 SWF 파일을 개발할 수 있습니다. LocalConnection 객체는 동일한 클라이언트 시스템에서 실행되는 SWF 파일 간에만 통신할 수 있습니다. 하지만 서로 다른 응용 프로그램에서 실행되는 경우는 가능합니다. 예를 들어 프로젝터에서 로컬 정보를 유지하고 원격으로 브라우저 기반 SWF 파일에 연결한 상태로 파일 브라우저에서 실행되는 SWF 파일과 프로젝터에서 실행되는 SWF 파일이 정보를 공유할 수 있습니다. 프로젝터는 독립 실행형 응용 프로그램으로 실행할 수 있는 형식으로 저장된 SWF 파일로, 실행 파일 내에 Flash Player가 포함되어 있으므로 Flash Player를 설치할 필요가 없습니다.

LocalConnection 객체는 다음과 같이 여러 ActionScript 버전을 사용하여 SWF 간에 통신하는 데 사용할 수 있습니다.

- ActionScript 3.0 LocalConnection 객체는 ActionScript 1.0 또는 2.0으로 만들어진 LocalConnection 객체와 통신할 수 있습니다.
- ActionScript 1.0 또는 2.0 LocalConnection 객체는 ActionScript 3.0으로 만들어진 LocalConnection 객체와 통신할 수 있습니다.

Flash Player는 다른 버전의 LocalConnection 객체 간의 통신을 자동으로 처리합니다.

LocalConnection 객체를 가장 간단하게 사용할 수 있는 방법은 동일한 도메인 또는 동일한 AIR 응용 프로그램에 위치한 LocalConnection 객체 간 통신만 허용하는 것입니다. 그러면 보안 문제에 대해 걱정하지 않아도 됩니다. 서로 다른 도메인 사이에서 통신해야 하는 경우, 보안 문제를 해결할 수 있는 몇 가지 방법이 있습니다. 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서에 나열된 LocalConnection 클래스의 allowDomain() 및 domain 항목과 send() 메서드의 connectionName 매개 변수에 대한 설명을 참조하십시오.

LocalConnection 객체를 사용하여 하나의 SWF 파일에 포함된 데이터를 송수신할 수 있으나 좋은 방법은 아닙니다. 대신 공유 객체를 사용하십시오.

LocalConnection 객체에 콜백 메서드를 추가하는 방법은 다음 세 가지입니다.

- LocalConnection 클래스를 하위 클래스로 사용하고 메서드 추가
- LocalConnection.client 속성을 이 메서드를 구현하는 객체로 설정
- LocalConnection을 확장하는 동적 클래스를 만들고 동적으로 메서드 추가

콜백 메서드를 추가하는 첫 번째 방법은 LocalConnection 클래스를 확장하는 것입니다. 동적으로 사용자 정의 클래스를 LocalConnection 인스턴스에 추가하지 않고 사용자 정의 클래스 내에 메서드를 정의합니다. 이 방법은 다음 코드에서 확인할 수 있습니다.

```
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);
        }
    }
}
```

CustomLocalConnection 클래스의 새 인스턴스를 만들려면 다음 코드를 사용할 수 있습니다.

```
var serverLC:CustomLocalConnection;
serverLC = new CustomLocalConnection("serverName");
```

콜백 메서드를 추가하는 두 번째 방법은 LocalConnection.client 속성을 사용하는 것입니다. 다음 코드와 같이 사용자 정의 클래스를 만들고 새 인스턴스를 client 속성에 할당하는 것입니다.

```
var lc:LocalConnection = new LocalConnection();
lc.client = new CustomClient();
```

LocalConnection.client 속성은 호출하면 안 되는 객체 콜백 메서드를 나타냅니다. 앞의 코드에서 client 속성은 사용자 정의 클래스 CustomClient의 새 인스턴스로 설정되었습니다. client 속성의 기본값은 현재 LocalConnection 인스턴스입니다. 윈도우 하나의 버튼이 두 번째 윈도우의 보기를 전환하는 응용 프로그램에서 메서드 설정은 같지만 다르게 작동하는 두 개의 데이터 핸들러가 있는 경우 client 속성을 사용할 수 있습니다.

CustomClient 클래스를 만들려면 다음 코드를 사용할 수 있습니다.

```
package
{
    public class CustomClient extends Object
    {
        public function onMethod(timeString:String):void
        {
            trace("onMethod called at: " + timeString);
        }
    }
}
```

동적 클래스를 만들고 동적으로 메서드를 추가하는 등 콜백 메서드를 추가하는 세 번째 방법은 다음 코드에서 보듯이 이전 버전의 ActionScript에서 LocalConnection 클래스를 사용하는 것과 매우 비슷합니다.

```
import flash.net.LocalConnection;
dynamic class DynamicLocalConnection extends LocalConnection {}
```

다음 코드를 사용하여 이 클래스에 콜백 메서드를 동적으로 추가할 수 있습니다.

```
var connection:DynamicLocalConnection = new DynamicLocalConnection();
connection.onMethod = this.onMethod;
// Add your code here.
public function onMethod(timeString:String):void
{
    trace("onMethod called at: " + timeString);
}
```

앞에서 설명한 콜백 메서드 추가 방법은 코드를 옮기기가 매우 어려우므로 사용하지 않는 것이 좋습니다. 뿐만 아니라 동적 속성에 액세스하는 경우 봉인된 속성에 액세스하는 경우보다 속도가 크게 느리므로 이 방법을 사용하여 로컬 연결을 만들면 성능 문제가 발생할 수 있습니다.

isPerUser 속성

Mac 컴퓨터에 둘 이상의 사용자가 로그인한 경우 발생하는 충돌을 해결할 수 있도록 isPerUser 속성이 Flash Player(10.0.32)와 AIR(1.5.2)에 추가되었습니다. 다른 운영 체제에서는 로컬 연결의 범위가 항상 개별 사용자로 제한되어 왔으므로 이 속성이 무시됩니다. isPerUser 속성은 새로운 코드에서 true로 설정되어야 합니다. 그러나 이전 버전과의 호환성을 위해 기본값은 현재 false입니다. 이후 버전의 런타임에서는 기본값이 변경될 수 있습니다.

두 응용 프로그램 간 메시지 송신

Flash Player 9 이상, Adobe AIR 1.0 이상

LocalConnection 클래스를 사용하면 서로 다른 AIR 응용 프로그램 간은 물론 브라우저에서 실행되는 서로 다른 Adobe® Flash® Player(SWF) 응용 프로그램 간에 통신할 수 있습니다. 또한 브라우저에서 실행하는 AIR 응용 프로그램 및 SWF 응용 프로그램 간에도 통신할 수 있습니다.

예를 들어 웹 페이지에 여러 Flash Player 인스턴스가 있거나 Flash Player 인스턴스가 팝업 윈도우의 Flash Player 인스턴스에서 데이터를 가져오도록 할 수 있습니다.

다음 코드에서는 서버처럼 작동하고 다른 응용 프로그램에서의 수신 LocalConnection 호출을 허용하는 LocalConnection 객체를 정의합니다.

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

우선 이 코드는 lc라는 LocalConnection 객체를 만들고 client 속성을 clientObject 객체로 설정합니다. 다른 응용 프로그램이 이 LocalConnection 인스턴스에서 메서드를 호출하면 런타임이 clientObject 객체에서 해당 메서드를 찾습니다.

지정한 이름의 연결이 이미 있는 경우 객체가 이미 연결되어 있어 해당 연결이 실패했음을 나타내는 인수 오류 예외가 발생합니다.

Flash Player 인스턴스가 이 SWF 파일에 연결하고 지정된 로컬 연결의 메서드를 호출하려 할 때마다 CustomClient1 클래스로 설정된 client 속성에서 지정한 클래스에 요청이 전송됩니다.

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

LocalConnection 서버를 만들려면 LocalConnection.connect() 메서드를 호출하고 고유한 연결 이름을 입력하십시오. 지정한 이름의 연결이 이미 있는 경우 객체가 이미 연결되어 있어 연결하지 못했음을 나타내는 ArgumentError 오류가 생성됩니다.

다음 코드에서는 conn1이라는 이름으로 LocalConnection을 만드는 방법을 보여 줍니다.

```
try
{
    connection.connect("conn1");
}
catch (error:ArgumentError)
{
    trace("Error! Server already exists\n");
}
```

보조 응용 프로그램에서 기본 응용 프로그램에 연결하려면 송신 LocalConnection 객체에서 새 LocalConnection 객체를 만든 다음 연결 이름 및 실행할 메서드 이름으로 LocalConnection.send() 메서드를 호출해야 합니다. 예를 들어 doQuit 메서드를 이전에 만든 LocalConnection 객체에 전송하려면 다음 코드를 사용합니다.

```
sendingConnection.send("conn1", "doQuit");
```

다음 코드는 기존 LocalConnection 객체를 conn1이라는 연결 이름으로 연결하고 원격 응용 프로그램에서 doMessage() 메서드를 호출합니다. 원격 응용 프로그램에 매개 변수를 보내려면 다음 코드에 나와 있는 대로 send() 메서드의 메서드 이름 뒤에 추가 인수를 지정합니다.

```
sendingConnection.send("conn1", "doMessage", "Hello world");
```

서로 다른 도메인의 내용 및 AIR 응용 프로그램에 연결

Flash Player 9 이상, Adobe AIR 1.0 이상

특정 도메인에서만 통신할 수 있도록 하려면 LocalConnection 클래스의 allowDomain() 또는 allowInsecureDomain() 메서드를 호출하고 이 LocalConnection 객체에 액세스할 수 있는 하나 이상의 도메인 목록을 전달하여 허용되는 하나 이상의 도메인 이름을 전달합니다.

이전 버전의 ActionScript에서 LocalConnection.allowDomain() 및 LocalConnection.allowInsecureDomain()은 개발자에 의해 구현되어야 하고 부울 값을 반환해야 하는 콜백 메서드였습니다. ActionScript 3.0에서 LocalConnection.allowDomain() 및 LocalConnection.allowInsecureDomain()은 기본 제공 메서드입니다. 따라서 개발자는 허용할 도메인 이름 하나 이상을 전달하여 Security.allowDomain() 및 Security.allowInsecureDomain()처럼 호출만 하면 됩니다.

Flash Player 8은 로컬 SWF 파일에 대한 보안 제한 사항을 추가했습니다. 인터넷 액세스가 허용된 SWF 파일이라도 로컬 파일 시스템에 액세스할 수 있는 권한은 없습니다. localhost를 지정하면 임의의 로컬 SWF 파일이 SWF 파일에 액세스할 수 있습니다. LocalConnection.send() 메서드가 호출 코드가 액세스 권한을 가지고 있지 않은 보안 샌드박스로부터 SWF 파일과 통신하려고 하면 securityError 이벤트(SecurityErrorEvent.SECURITY_ERROR)가 전달됩니다. 이 오류를 해결하려면 수신자의 LocalConnection.allowDomain() 메서드에서 호출자의 도메인을 지정합니다.

LocalConnection.allowDomain() 및 LocalConnection.allowInsecureDomain() 메서드에 전달할 수 있는 특수한 값은 *와 localhost 두 가지입니다. 별표 값(*)은 모든 도메인에서 액세스를 허용합니다. 문자열 localhost는 로컬로 설치되지만 응용 프로그램 리소스 디렉토리 외부의 내용에서 응용 프로그램에 대한 호출을 허용합니다.

LocalConnection.send() 메서드가 호출 코드에 액세스 권한이 없는 보안 샌드박스로부터 응용 프로그램과 통신하려고 하면 securityError 이벤트(SecurityErrorEvent.SECURITY_ERROR)가 전달됩니다. 이 오류를 해결하려면 수신자의 LocalConnection.allowDomain() 메서드에서 호출자의 도메인을 지정합니다.

같은 도메인의 SWF 내용 간 통신만 구현하는 경우 밑줄(_)로 시작하지 않고 도메인 이름(예: myDomain:connectionName)을 지정하지 않는 connectionName 매개 변수를 지정할 수 있습니다. LocalConnection.connect(connectionName) 명령에서 같은 문자열을 사용합니다.

서로 다른 도메인의 내용 간 통신을 구현하는 경우 밑줄로 시작하는 connectionName 매개 변수를 지정합니다. 밑줄을 지정하면 수신 LocalConnection 객체가 있는 내용을 도메인 간에 보다 자유롭게 이동할 수 있습니다. 다음과 같은 두 가지 경우가 가능합니다.

- connectionName의 문자열이 밑줄로 시작하지 않을 경우 런타임은 상위 도메인 이름과 콜론이 포함된 접두어(예: myDomain:connectionName)를 추가합니다. 이렇게 하면 이름이 같은 다른 도메인의 연결과 충돌하지 않지만 모든 송신 LocalConnection 객체는 이 상위 도메인(예: myDomain:connectionName)을 지정해야 합니다. 수신 LocalConnection 객체가 있는 HTML 또는 SWF 파일을 다른 도메인으로 이동하면 런타임이 새 상위 도메인이 반영되도록 접두어를 변경합니다(예: anotherDomain:connectionName). 모든 송신 LocalConnection 객체는 새 상위 도메인을 가리키도록 수동으로 편집해야 합니다.
- connectionName의 문자열이 밑줄(_)로 시작할 경우(예: _connectionName) 런타임은 접두어를 문자열에 추가하지 않습니다. 즉, 수신 및 송신 LocalConnection 객체가 connectionName에 동일한 문자열을 사용한다는 것을 의미합니다. 수신 객체에서 LocalConnection.allowDomain()을 사용하여 모든 도메인의 연결을 승인하도록 지정할 경우 송신 LocalConnection 객체를 변경하지 않고 수신 LocalConnection 객체가 있는 HTML 또는 SWF 파일을 다른 도메인으로 이동할 수 있습니다.

  connectionName에 밑줄 이름을 사용할 때의 단점은 두 응용 프로그램 모두 동일한 connectionName을 사용하여 연결을 시도하는 경우와 같이 충돌 가능성이 있다는 것입니다. 두 번째 단점은 보안 문제입니다. 밑줄 구문을 사용하는 연결 이름은 수신 응용 프로그램의 도메인을 식별하지 않습니다. 이러한 이유 때문에 도메인 정규화된 이름을 선호합니다.

Adobe AIR

AIR 응용 프로그램 보안 샌드박스에서 실행 중인 내용(AIR 응용 프로그램과 함께 설치되는 내용)과 통신하려면 연결 이름의 앞에 AIR 응용 프로그램을 식별하는 수퍼 도메인을 추가해야 합니다. 수퍼 도메인 문자열은 app#으로 시작하며, 응용 프로그램 ID, 점(.) 문자, 제작자 ID(정의된 경우)가 차례로 뒤에 이어집니다. 예를 들어 응용 프로그램 ID가 com.example.air.MyApp이고

제작자 ID가 없는 응용 프로그램의 connectionName 매개 변수에서 사용할 올바른 수퍼 도메인은 "app#com.example.air.MyApp"입니다. 따라서 기본 연결 이름이 "appConnection"인 경우 connectionName 매개 변수에서 사용할 전체 문자열은 "app#com.example.air.MyApp:appConnection"입니다. 응용 프로그램에 제작자 ID가 있는 경우에는 "app#com.example.air.MyApp.B146A943FBD637B68C334022D304CEA226D129B4.1"과 같이 해당 ID도 수퍼 도메인 문자열에 포함되어야 합니다.

다른 AIR 응용 프로그램이 로컬 연결을 통해 응용 프로그램과 통신할 수 있도록 허용하는 경우 LocalConnection 객체의 allowDomain()을 호출하여 로컬 연결 도메인 이름으로 전달해야 합니다. AIR 응용 프로그램의 경우 이 도메인 이름은 연결 문자열과 동일한 방식으로 응용 프로그램 및 제작자 ID를 사용하여 형성됩니다. 예를 들어 보내는 AIR 응용 프로그램의 응용 프로그램 ID가 com.example.air.FriendlyApp이고 제작자 ID가 214649436BD677B62C33D02233043EA236D13934.1인 경우 이 응용 프로그램에서 연결할 수 있도록 허용하는 데 사용할 도메인 문자열은 app#com.example.air.FriendlyApp.214649436BD677B62C33D02233043EA236D13934.1입니다. AIR 1.5.3 현재, 일부 AIR 응용 프로그램에는 제작자 ID가 없습니다.

46장: AIR의 기본 프로세스와 통신

Adobe AIR 2 이상

Adobe AIR 2에서는 명령줄을 통해 AIR 응용 프로그램이 실행되고 다른 기본 프로세스와 통신할 수 있습니다. 예를 들어 AIR 응용 프로그램에서 프로세스를 실행하고 표준 입력 및 출력 스트림을 통해 해당 프로세스와 통신할 수 있습니다.

기본 프로세스와 통신하려면 기본 설치 프로그램을 통해 설치되도록 AIR 응용 프로그램을 패키지화해야 합니다. 기본 설치 프로그램의 파일 유형은 설치되는 운영 체제에 따라 다릅니다.

- Mac OS의 경우 DMG 파일
- Windows의 경우 EXE 파일
- Linux의 경우 RPM 또는 DEB 패키지

이러한 응용 프로그램을 확장 데스크톱 프로파일 응용 프로그램이라고 합니다. ADT를 사용하여 -target native 옵션(-package 명령 호출 시)을 지정하여 기본 설치 프로그램 파일을 만들 수 있습니다.

기타 도움말 항목

flash.filesystem.File.openWithDefaultApplication()

flash.desktop.NativeProcess

기본 프로세스 통신 개요

Adobe AIR 2 이상

확장 데스크톱 프로파일의 AIR 응용 프로그램은 명령줄에서 호출된 것처럼 파일을 실행할 수 있으며 기본 프로세스의 표준 스트림과 통신할 수 있습니다. 표준 스트림에는 표준 입력 스트림(stdin), 출력 스트림(stdout), 표준 오류 스트림(stderr)이 포함됩니다.

참고: 확장 데스크톱 프로파일의 응용 프로그램은 File.openWithDefaultApplication() 메서드를 사용하여 파일 및 응용 프로그램을 실행할 수도 있습니다. 그러나 이 메서드를 사용하면 AIR 응용 프로그램에 표준 스트림에 대한 액세스 권한이 제공되지 않습니다. 자세한 내용은 621페이지의 “기본 시스템 응용 프로그램으로 파일 열기”를 참조하십시오.

다음 코드 샘플에서는 응용 프로그램 디렉토리에서 test.exe 응용 프로그램을 실행하는 방법을 보여 줍니다. 응용 프로그램은 인수 "hello"를 명령줄 인수로 전달하고 프로세스의 표준 출력 스트림에 이벤트 리스너를 추가합니다.

```
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);
}
```

기본 프로세스 실행 및 닫기

Adobe AIR 2 이상

기본 프로세스를 실행하려면 다음을 수행하도록 NativeProcessInfo 객체를 설정합니다.

- 실행하려는 파일 가리키기
- 시작 시 프로세스에 전달할 명령줄 인수 저장 (선택 사항)
- 프로세스의 작업 디렉토리 설정(선택 사항)

기본 프로세스를 시작하려면 NativeProcessInfo 객체를 NativeProcess 객체의 start() 메서드에 대한 매개 변수로 전달합니다.

예를 들어 다음 코드에서는 응용 프로그램 디렉토리에서 test.exe 응용 프로그램을 실행하는 방법을 보여 줍니다. 응용 프로그램은 인수 "hello"를 전달하고 사용자의 문서 디렉토리를 작업 디렉토리로 설정합니다.

```
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);
```

프로세스를 종료하려면 NativeProcess 객체의 exit() 메서드를 호출합니다.

설치된 응용 프로그램에서 파일이 실행 가능하도록 하려면 해당 응용 프로그램을 패키지화할 때 파일 시스템에서 해당 파일을 실행할 수 있어야 합니다. Mac 및 Linux에서는 필요한 경우 chmod를 사용하여 실행 가능 플래그를 설정할 수 있습니다.

기본 프로세스와 통신

Adobe AIR 2 이상

AIR 응용 프로그램이 기본 프로세스를 시작한 후에는 프로세스의 표준 입력, 표준 출력 및 표준 오류 스트림과 통신할 수 있습니다.

NativeProcess 객체의 다음 속성을 사용하여 스트림에서 데이터를 읽고 쓸 수 있습니다.

- standardInput - 표준 입력 스트림 데이터에 대한 액세스를 포함합니다.
- standardOutput - 표준 출력 스트림 데이터에 대한 액세스를 포함합니다.
- standardError - 표준 오류 스트림 데이터에 대한 액세스를 포함합니다.

표준 입력 스트림에 쓰기

NativeProcess 객체의 standardInput 속성에 대한 쓰기 메서드를 사용하여 표준 입력 스트림에 데이터를 쓸 수 있습니다. AIR 응용 프로그램이 프로세스에 데이터를 쓰면 NativeProcess 객체가 standardInputProgress 이벤트를 전달합니다.

표준 입력 스트림에 쓰는 동안 오류가 발생하면 NativeProcess 객체가 ioErrorStandardInput 이벤트를 전달합니다.

NativeProcess 객체의 closeInput() 메서드를 호출하여 입력 스트림을 닫을 수 있습니다. 입력 스트림이 닫히면 NativeProcess 객체가 standardInputClose 이벤트를 전달합니다.

```
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();
}
```

표준 출력 스트림에서 읽기

이 속성의 읽기 메서드를 사용하여 표준 출력 스트림에서 데이터를 읽을 수 있습니다. AIR 응용 프로그램이 프로세스에서 출력 스트림 데이터를 가져오면 NativeProcess 객체가 standardOutputData 이벤트를 전달합니다.

표준 출력 스트림에 쓰는 동안 오류가 발생하면 NativeProcess 객체가 standardOutputError 이벤트를 전달합니다.

프로세스에서 출력 스트림을 닫으면 NativeProcess 객체가 standardOutputClose 이벤트를 전달합니다.

표준 입력 스트림에서 데이터를 읽을 때는 생성된 대로 데이터를 읽어야 합니다. 즉, standardOutputData 이벤트에 대한 이벤트 리스너를 추가한 후 standardOutputData 이벤트 리스너에서 NativeProcess 객체의 standardOutput 속성으로부터 데이터를 읽어야 합니다. 단순히 standardOutputClose 이벤트 또는 exit 이벤트에서 모든 데이터를 읽을 때까지 기다리면 안 됩니다. 기본 프로세스에서 데이터가 생성된 대로 데이터를 읽지 않을 경우 버퍼가 가득 차서 데이터가 손실될 수 있습니다. 버퍼가 차면 데이터를 더 쓰려고 시도할 때 기본 프로세스가 멈출 수 있습니다. 반면 standardOutputData 이벤트에 대한 이벤트 리스너를 등록하지 않을 경우 버퍼가 차지 않아 프로세스가 멈추지 않습니다. 이 경우 데이터에 액세스할 수 없습니다.

```
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);
}
```

표준 오류 스트림에서 읽기

이 속성의 읽기 메서드를 사용하여 표준 오류 스트림에서 데이터를 읽을 수 있습니다. AIR 응용 프로그램이 프로세스에서 오류 스트림 데이터를 읽으면 NativeProcess 객체가 standardErrorData 이벤트를 전달합니다.

표준 오류 스트림에 쓰는 동안 오류가 발생하면 NativeProcess 객체가 standardErrorIoError 이벤트를 전달합니다.

프로세스에서 오류 스트림을 닫으면 NativeProcess 객체가 standardErrorClose 이벤트를 전달합니다.

표준 오류 스트림에서 데이터를 읽을 때는 생성된 대로 데이터를 읽어야 합니다. 즉, standardErrorData 이벤트에 대한 이벤트 리스너를 추가한 후 standardErrorData 이벤트 리스너에서 NativeProcess 객체의 standardError 속성으로부터 데이터를 읽어야 합니다. 단순히 standardErrorClose 이벤트 또는 exit 이벤트에서 모든 데이터를 읽을 때까지 기다리면 안 됩니다. 기본 프로세스에서 데이터가 생성된 대로 데이터를 읽지 않을 경우 버퍼가 가득 차서 데이터가 손실될 수 있습니다. 버퍼가 차면 데이터를 더 쓰려고 시도할 때 기본 프로세스가 멈출 수 있습니다. 반면 standardErrorData 이벤트에 대한 이벤트 리스너를 등록하지 않을 경우 버퍼가 차지 않아 프로세스가 멈추지 않습니다. 이 경우 데이터에 액세스할 수 없습니다.

```
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);
}
```

기본 프로세스 통신에 대한 보안 고려 사항

Adobe AIR 2 이상

기본 프로세스 API는 사용자 시스템의 모든 실행 파일을 실행할 수 있습니다. 명령을 생성하고 실행할 때는 각별한 주의가 필요합니다. 실행할 명령의 일부를 외부 소스에서 가져오는 경우 해당 명령이 실행하기에 안전한지 신중하게 유효성을 검사해야 합니다. 마찬가지로 AIR 응용 프로그램은 실행 프로세스로 전달되는 모든 데이터의 유효성을 검사해야 합니다.

그러나 입력의 유효성을 검사하는 것은 어려운 작업일 수 있습니다. 이와 같은 어려움을 없애려면 특정 API가 포함된 기본 응용 프로그램(예: Windows의 EXE 파일)을 작성하는 것이 가장 좋습니다. 이러한 API는 해당 응용 프로그램에서 정의한 명령만 처리해야 합니다. 예를 들어 응용 프로그램에서 표준 입력 스트림을 통해 제한된 명령 세트만 허용할 수 있습니다.

Windows에서 AIR를 사용하는 경우 .bat 파일을 직접 실행할 수 없습니다. 대신 명령 인터프리터 응용 프로그램(cmd.exe)을 사용하여 Windows .bat 파일을 실행합니다. .bat 파일을 호출하면 이 명령 응용 프로그램에서 명령에 전달된 인수를 추가로 실행해야 할 응용 프로그램으로 해석할 수 있습니다. 인수 문자열에 추가 문자를 악의적으로 삽입할 경우 cmd.exe가 유해하거나 안전하지 않은 응용 프로그램을 실행할 수 있습니다. 예를 들어 올바른 데이터 유효성 검사를 수행하지 않으면 AIR 응용 프로그램이 myBat.bat myArguments c:/evil.exe를 호출할 수 있습니다. 명령 응용 프로그램에서 일괄 처리 파일을 실행할 뿐 아니라 evil.exe 응용 프로그램도 실행하게 됩니다.

47장: 외부 API 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0 외부 API(flash.external.ExternalInterface)를 사용하면 Adobe Flash Player가 실행되고 있는 컨테이너 응용 프로그램과 ActionScript 간에 간단하게 통신할 수 있습니다. SWF 문서와 HTML 페이지의 JavaScript 간에 상호 작용을 생성하려면 ExternalInterface API를 사용합니다.

외부 API를 사용하여 컨테이너 응용 프로그램과 상호 작용하고 HTML 페이지의 JavaScript와 ActionScript 간에 데이터를 전달할 수 있습니다.

몇 가지 일반적인 외부 API 작업은 다음과 같습니다.

- 컨테이너 응용 프로그램에 대한 정보 얻기
- 브라우저 또는 AIR 데스크톱 응용 프로그램에 표시된 웹 페이지에서 ActionScript를 사용하여 코드 호출
- 웹 페이지에서 ActionScript 코드 호출
- 프록시를 통해 웹 페이지에서의 ActionScript 코드 호출 간소화

참고: 외부 인터페이스에 대한 이 논의에서는 SWF 파일의 ActionScript와 Flash Player 또는 해당 SWF가 로드된 인스턴스를 참조하는 컨테이너 응용 프로그램 간의 통신에 대해서만 다룹니다. 응용 프로그램 내에서 Flash Player를 사용하는 방법은 여기서 설명하지 않습니다. Flash Player는 브라우저 플러그인 또는 프로젝터(독립 실행형 응용 프로그램)용으로 만들어졌습니다. 그 밖의 사용은 지원이 제한될 수 있습니다.

AIR에서 외부 API 사용

AIR 응용 프로그램에는 외부 컨테이너가 없으므로 이 외부 인터페이스가 일반적으로 적용되지 않으며 일반적으로 필요하지도 않습니다. AIR 응용 프로그램에서 SWF 파일을 직접 로드하는 경우 응용 프로그램 코드가 SWF의 ActionScript 코드와 직접 통신할 수 있습니다(보안 샌드박스 제한 사항이 적용됨).

그러나 AIR 응용 프로그램에서 HTMLLoader 객체(또는 Flex의 HTML 구성 요소)를 사용하여 HTML 페이지에서 SWF 파일을 로드하는 경우 HTMLLoader 객체가 외부 컨테이너 역할을 합니다. 따라서 외부 인터페이스를 사용하여 로드된 SWF의 ActionScript 코드와 HTMLLoader에 로드된 HTML 페이지의 JavaScript 코드 간에 통신할 수 있습니다.

외부 API 사용의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

경우에 따라 자체적으로 SWF 파일을 실행(예: Adobe® Flash® Professional을 사용하여 SWF 프로젝터를 만든 경우)할 수도 있지만, 대부분의 경우 SWF 응용 프로그램은 다른 응용 프로그램 내부의 요소로 실행됩니다. SWF를 포함하는 컨테이너는 대개 HTML 파일입니다. 그러나 데스크톱 응용 프로그램의 사용자 인터페이스 전체 또는 일부에 SWF 파일을 사용하기도 합니다.

고급 기능을 갖춘 응용 프로그램을 사용하다 보면 SWF 파일과 컨테이너 응용 프로그램 간의 통신 설정이 필요할 수 있습니다. 예를 들어 웹 페이지에서는 텍스트 또는 기타 정보를 HTML로 표시하고 SWF 파일을 포함시켜 차트나 비디오 등의 동적인 시각적 정보를 표시할 수 있습니다. 이 경우 사용자가 웹 페이지의 버튼을 클릭할 때 SWF 파일의 내용이 변경되도록 할 수 있습니다. ActionScript에는 SWF 파일의 ActionScript와 컨테이너 응용 프로그램의 기타 코드 사이에 이러한 통신을 가능하게 하는 외부 API라는 메커니즘이 포함되어 있습니다.

중요한 개념 및 용어

이 기능과 관련된 중요한 용어가 아래 참조 목록에 정리되어 있습니다.

컨테이너 응용 프로그램 Flash Player가 SWF 파일을 실행하는 응용 프로그램(예: Flash Player 내용이 포함된 HTML 페이지 및 웹 브라우저 또는 웹 페이지에 SWF를 로드하는 AIR 응용 프로그램)

프로젝터 SWF 내용과 Flash Player가 포함된 실행 파일입니다. 프로젝터 파일은 Flash Professional이나 독립 실행형 Flash Player를 사용하여 만들 수 있습니다. 프로젝터는 CD-ROM을 통해 SWF 파일을 배포하거나, 다운로드 크기가 문제되지 않고 SWF 제작자가 Flash Player의 설치 여부와 관계없이 사용자가 SWF 파일을 실행할 수 있도록 하려는 유사 환경에서 SWF 파일을 배포하는 데 일반적으로 사용됩니다.

프록시 특정 응용 프로그램("외부 응용 프로그램")에서 다른 응용 프로그램("호출 응용 프로그램") 대신 코드를 호출하여 호출 응용 프로그램으로 값을 반환하는 중개 응용 프로그램 또는 코드입니다. 프록시는 다음과 같은 여러 가지 용도에 사용될 수 있습니다.

- 호출 응용 프로그램의 네이티브 함수 호출을 외부 응용 프로그램이 이해할 수 있는 형식으로 변환하여 외부 함수 호출 프로세스 간소화.
- 호출자가 외부 응용 프로그램과 직접 통신하는 것을 막는 보안 또는 기타 제한 사항 문제 해결

직렬화 인터넷 환경이나 단일 컴퓨터에서 실행되는 서로 다른 두 응용 프로그램과 같은 두 프로그래밍 시스템 간 메시지를 통한 값 전달에 사용할 수 있는 형식으로 객체 또는 데이터 값을 변환하는 작업입니다.

예제를 사용하여 작업

제공된 코드 예제 중 상당수는 전체 작업 예제 또는 값 확인용 코드가 아니라 데모용 소형 코드 목록입니다. 외부 API를 사용하려면 컨테이너 응용 프로그램 코드와 함께 ActionScript 코드를 작성해야 하므로, 컨테이너(예: SWF 파일이 포함된 웹 페이지)를 만들고 코드 목록을 사용하여 해당 컨테이너와 상호 작용하는 과정이 예제 테스트에 포함됩니다.

ActionScript와 JavaScript 간의 통신 예제를 테스트하려면

1. Flash Professional을 사용하여 새 문서를 만들고 컴퓨터에 저장합니다.
2. 주 메뉴에서 [파일] > [제작 설정]을 선택합니다.
3. [제작 설정] 대화 상자의 [포맷] 탭에서 HTML 및 Flash 체크 상자가 선택되어 있는지 확인합니다.
4. [제작] 버튼을 클릭합니다. 이렇게 하면 문서 저장 시 사용한 것과 같은 이름의 SWF 파일과 HTML 파일이 같은 폴더에 생성됩니다. [확인]을 클릭하여 [제작 설정] 대화 상자를 닫습니다.
5. HTML 체크 상자의 선택을 취소합니다. 이제 HTML 페이지가 생성되었으므로 이를 수정하여 원하는 JavaScript 코드를 추가할 수 있습니다. HTML 체크 상자의 선택을 취소하면 HTML 페이지를 수정하더라도 Flash에서 SWF 파일을 제작할 때 새 HTML 문서로 사용자의 변경 사항을 덮어쓰는 일이 없습니다.
6. [확인]을 클릭하여 [제작 설정] 대화 상자를 닫습니다.
7. HTML 또는 텍스트 편집기 응용 프로그램을 사용하여 SWF 파일 제작 시 Flash에서 만든 HTML 파일을 엽니다. HTML 소스 코드에서 열기 및 닫기 script 태그를 추가한 다음 예제 코드 샘플의 JavaScript 코드를 해당 태그에 복사합니다.

```
<script>
// add the sample JavaScript code here
</script>
```

8. HTML 파일을 저장하고 Flash로 돌아갑니다.
9. 타임라인의 프레임 1에서 키프레임을 선택하고 [액션] 패널을 엽니다.
10. [스크립트] 창에 ActionScript 코드 샘플을 복사합니다.
11. 주 메뉴에서 [파일] > [제작]을 선택하여 SWF 파일에 변경 사항을 업데이트합니다.
12. 웹 브라우저에서 편집한 HTML 페이지를 열어 확인한 후 ActionScript와 HTML 간의 통신을 테스트합니다.

ActionScript와 ActiveX 컨테이너 간의 통신 예제를 테스트하려면

1. Flash Professional을 사용하여 새 문서를 만들고 컴퓨터에 저장합니다. 해당 문서를 컨테이너 응용 프로그램의 SWF 파일 폴더에 저장하려면
2. 주 메뉴에서 [파일] > [제작 설정]을 선택합니다.
3. [제작 설정] 대화 상자의 [포맷] 탭에서 Flash 체크 상자만 선택되어 있는지 확인합니다.
4. Flash 체크 상자 옆에 있는 [파일] 필드에서 폴더 아이콘을 클릭하여 SWF 파일을 제작할 폴더를 선택합니다. SWF 파일 위치를 설정하여 문서를 한 폴더에 저장할 수도 있지만, 제작된 SWF 파일은 컨테이너 응용 프로그램의 소스 코드가 들어 있는 폴더 등 다른 폴더에 저장하는 것이 좋습니다.
5. 타임라인의 프레임 1에서 키프레임을 선택하고 [액션] 패널을 엽니다.
6. 예제의 ActionScript 코드를 [스크립트] 창으로 복사합니다.
7. 주 메뉴에서 [파일] > [제작]을 선택하여 SWF 파일을 다시 제작합니다.
8. 컨테이너 응용 프로그램을 만들고 실행하여 ActionScript와 컨테이너 응용 프로그램 간의 통신을 테스트합니다.

외부 API를 사용하여 HTML 페이지와 통신하는 방법에 대한 전체 예제를 확인하려면 다음 항목을 참조하십시오.

- 772페이지의 “외부 API 예제: 웹 브라우저에서 ActionScript와 JavaScript 간의 통신”

이러한 예제에는 ActionScript 및 컨테이너 오류 검사 코드를 비롯한 전체 코드가 포함되어 있으며, 외부 API로 코드를 작성할 때 이 코드를 사용해야 합니다. 외부 API를 사용하는 기타 전체 예제는 ActionScript 3.0 참조 설명서에서 ExternalInterface 클래스에 대한 클래스 예제를 참조하십시오.

외부 API 요구 사항 및 장점

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 API는 ActionScript의 일부로서, Flash Player의 컨테이너 역할을 하는 "외부 응용 프로그램"(일반적으로 웹 브라우저 또는 독립 실행형 프로젝터 응용 프로그램)에서 실행되는 코드와 ActionScript 간의 통신 메커니즘을 제공합니다. ActionScript 3.0에서는 ExternalInterface 클래스에서 외부 API의 기능을 제공합니다. Flash Player 8의 이전 버전에서는 fscommand() 액션을 사용하여 컨테이너 응용 프로그램과 통신합니다. ExternalInterface 클래스는 fscommand()를 대체합니다.

참고: 예를 들어 이전 응용 프로그램과 호환성을 유지해야 하거나 타사 SWF 컨테이너 응용 프로그램 또는 독립 실행형 Flash Player와의 호환을 위해 이전 fscommand() 함수를 사용해야 하는 경우, flash.system 패키지에서 패키지 수준 함수로 사용할 수 있습니다.

ExternalInterface 클래스는 ActionScript 및 Flash Player에서 HTML 페이지 JavaScript로의 통신을 손쉽게 수행할 수 있도록 하는 하위 시스템입니다.

ExternalInterface 클래스는 다음 경우에만 사용할 수 있습니다.

- 지원되는 모든 버전의 Internet Explorer for Windows(5.0 이상)
- NPRuntime 인터페이스를 지원하는 브라우저(현재 Firefox 1.0 이상, Mozilla 1.7.5 이상, Netscape 8.0 이상 및 Safari 1.3 이상이 포함됨)
- AIR 응용 프로그램(HTMLLoader 컨트롤을 통해 표시되는 HTML 페이지에 SWF가 포함된 경우)

그 밖의 모든 경우(예: 독립 실행형 플레이어에서 실행) ExternalInterface.available 속성은 false를 반환합니다.

ActionScript에서는 HTML 페이지에서 JavaScript 함수를 호출할 수 있습니다. 외부 API는 fscommand()와 비교할 때 다음과 같은 향상된 기능을 제공합니다.

- fscommand() 함수와 함께 사용되는 함수는 물론 모든 JavaScript 함수를 사용할 수 있습니다.

- 이름과 개수에 상관없이 인수를 전달할 수 있습니다. 명령과 단일 문자열 인수를 전달하는 데 제한이 없습니다. 따라서 외부 API는 fscommand()보다 유연성이 뛰어납니다.
- Boolean, Number 및 String 등 다양한 유형의 데이터를 전달할 수 있으며 String 매개 변수에 국한되지 않습니다.
- 호출 값을 수신할 수 있으며 그 값은 즉시 사용자 호출의 반환 값으로 ActionScript에 반환됩니다.

중요: HTML 페이지의 Flash Player 인스턴스에 제공된 이름(object 태그의 id 특성)에 하이픈(-)이나 JavaScript에서 연산자로 정의된 다른 문자(예: +, *, /, \, . 등)가 포함되어 있으면, Internet Explorer에서 컨테이너 웹 페이지를 볼 때 ActionScript가 ExternalInterface를 호출할 수 없습니다. 또한 Flash Player 인스턴스를 정의하는 HTML 태그(object 및 embed 태그)가 HTML form 태그에 중첩되어 있는 경우에도 ActionScript에서 ExternalInterface가 호출되지 않습니다.

ExternalInterface 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript와 컨테이너 응용 프로그램 간의 통신은 다음 두 가지 양식 중 하나를 사용할 수 있습니다. 즉, ActionScript가 컨테이너에 정의된 코드(예: JavaScript 함수)를 호출하거나 컨테이너의 코드가 호출 가능한 것으로 지정된 ActionScript 함수를 호출할 수 있습니다. 두 경우에 호출되는 코드에 정보를 전송하고 호출을 만드는 코드에 결과를 반환할 수 있습니다.

ExternalInterface 클래스에는 이러한 통신을 쉽게 하기 위한 두 개의 정적 속성과 두 개의 정적 메서드가 포함되어 있습니다. 이러한 속성과 메서드를 사용하여 외부 인터페이스 연결에 대한 정보를 가져오고, ActionScript에서 컨테이너의 코드를 실행하고, 컨테이너에서 ActionScript 함수를 호출할 수 있는 상태로 만듭니다.

외부 컨테이너에 대한 정보 얻기

Flash Player 9 이상, Adobe AIR 1.0 이상

ExternalInterface.available 속성은 현재 Flash Player가 외부 인터페이스를 제공하는 컨테이너에 있는지 여부를 나타냅니다. 외부 인터페이스를 사용할 수 있는 경우 이 속성은 true이고, 그렇지 않으면 false입니다. ExternalInterface 클래스에서 다른 함수를 사용하기 전에 항상 다음과 같이 현재 컨테이너에서 외부 인터페이스 통신을 지원하는지 확인해야 합니다.

```
if (ExternalInterface.available)
{
    // Perform ExternalInterface method calls here.
}
```

참고: ExternalInterface.available 속성은 현재 컨테이너가 ExternalInterface 연결을 지원하는 유형인지 여부를 보고합니다. 현재 브라우저에서 JavaScript가 활성화되어 있는지 여부는 표시되지 않습니다.

ExternalInterface.objectID 속성을 사용하면 Flash Player 인스턴스의 고유한 ID(특히 Internet Explorer에서 object 태그의 id 특성 또는 NPRuntime 인터페이스를 사용하는 브라우저에서 embed 태그의 name 특성)를 확인할 수 있습니다. 이 고유한 ID는 브라우저에 있는 현재 SWF 문서를 나타내고 SWF 문서에 대한 참조를 만드는 데 사용할 수 있습니다. 예를 들어 컨테이너 HTML 페이지에서 JavaScript 함수를 호출할 때 사용할 수 있습니다. Flash Player 컨테이너가 웹 브라우저가 아니면 이 속성은 null입니다.

ActionScript에서 외부 코드 호출

Flash Player 9 이상, Adobe AIR 1.0 이상

ExternalInterface.call() 메서드는 컨테이너 응용 프로그램에서 코드를 실행합니다. 이때 컨테이너 응용 프로그램에서 호출할 함수 이름이 포함된 문자열이 매개 변수로 반드시 필요합니다. ExternalInterface.call() 메서드에 전달된 그 밖의 매개 변수는 함수 호출의 매개 변수로 컨테이너에 전달됩니다.

```
// 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);
```

컨테이너가 HTML 페이지이면 이 메서드는 해당 HTML 페이지의 script 요소에 정의된 이름으로 JavaScript 함수를 호출합니다. JavaScript 함수의 반환 값은 ActionScript로 다시 전달됩니다.

```
<script language="JavaScript">
    // adds two numbers, and sends the result back to ActionScript
    function addNumbers(num1, num2)
    {
        return (num1 + num2);
    }
</script>
```

컨테이너가 다른 ActiveX 컨테이너인 경우 이 메서드로 인해 Flash Player ActiveX 컨트롤이 해당 FlashCall 이벤트를 전달합니다. 지정된 함수 이름 및 매개 변수는 Flash Player에서 XML 문자열로 직렬화합니다. 컨테이너는 event 객체의 request 속성에서 이 정보에 액세스할 수 있으며 이 정보를 통해 자신의 코드를 어떻게 실행할지 결정합니다. 컨테이너 코드는 ActionScript에 값을 반환하기 위해 ActiveX 객체의 SetReturnValue() 메서드를 호출하여 그 결과(XML 문자열로 직렬화된 결과)를 해당 메서드의 매개 변수로 전달합니다. 이 통신에 사용되는 XML 형식에 대한 자세한 내용은 771페이지의 “외부 API의 XML 형식”을 참조하십시오.

컨테이너가 웹 브라우저인지 또는 다른 ActiveX 컨테이너인지 관계없이 호출이 실패하거나 컨테이너 메서드에서 반환 값을 지정하지 않으면 null이 반환됩니다. 포함 환경이 호출 코드가 액세스할 수 없는 보안 샌드박스에 속하는 경우 ExternalInterface.call() 메서드가 SecurityError 예외를 발생시킵니다. 컨테이너 환경에서 allowScriptAccess에 적절한 값을 설정하면 이 문제를 해결할 수 있습니다. 예를 들어 HTML 페이지에서 allowScriptAccess의 값을 변경하려면 object 및 embed 태그에서 해당 특성을 편집합니다.

컨테이너에서 ActionScript 코드 호출

Flash Player 9 이상, Adobe AIR 1.0 이상

컨테이너는 함수에 있는 ActionScript 코드만 호출할 수 있고 다른 ActionScript 코드는 호출할 수 없습니다. 컨테이너 응용 프로그램에서 ActionScript 함수를 호출하려면 ExternalInterface 클래스로 함수를 등록한 다음 컨테이너의 코드에서 호출해야 합니다.

먼저 ActionScript 함수를 등록하여 컨테이너에서 사용할 수 있도록 나타내야 합니다. 다음과 같이 ExternalInterface.addCallback() 메서드를 사용합니다.

```
function callMe(name:String):String
{
    return "busy signal";
}
ExternalInterface.addCallback("myFunction", callMe);
```

addCallback() 메서드는 두 가지 매개 변수를 사용합니다. 먼저 문자열로 된 함수 이름에 의해 컨테이너에서 함수를 인식하게 됩니다. 두 번째 매개 변수는 컨테이너가 정의된 함수 이름을 호출할 때 실행될 실제 ActionScript 함수입니다. 이러한 이름은 고유하므로 실제 ActionScript 함수에 다른 이름이 있더라도 컨테이너에서 사용할 함수 이름을 지정할 수 있습니다. 이것은 함수 이름을 모르는 경우, 예를 들어 익명 함수가 지정되거나 호출할 함수가 런타임에 확인되는 경우에 특히 유용합니다.

ActionScript 함수는 ExternalInterface 클래스로 등록되었으므로 컨테이너에서 실제로 이 함수를 호출할 수 있습니다. 실행 방법은 컨테이너 유형에 따라 다양합니다. 예를 들어 웹 브라우저의 JavaScript 코드에서 ActionScript 함수는 Flash Player 브라우저 객체의 메서드(object 또는 embed 태그를 나타내는 JavaScript 객체의 메서드)와 마찬가지로 등록된 함수 이름을 사용하여 호출됩니다. 즉, 로컬 함수가 호출되는 것처럼 매개 변수가 전달되고 결과가 반환됩니다.

```
<script language="JavaScript">
    // callResult gets the value "busy signal"
    var callResult = flashObject.myFunction("my name");
</script>
...
<object id="flashObject"...>
    ...
    <embed name="flashObject".../>
</object>
```

또는 데스크톱 응용 프로그램에서 실행되는 SWF 파일에서 ActionScript 함수를 호출할 때 등록된 함수 이름 및 매개 변수를 XML 형식 문자열로 직렬화해야 합니다. 그러면 XML 문자열을 매개 변수로 ActiveX 컨트롤의 CallFunction() 메서드를 호출하는 방식으로 호출이 실제로 수행됩니다. 이 통신에 사용되는 XML 형식에 대한 자세한 내용은 771페이지의 "외부 API의 XML 형식"을 참조하십시오.

두 경우 모두 ActionScript 함수의 반환 값은 다시 컨테이너 코드로 전달됩니다. 호출자가 브라우저의 JavaScript인 경우에는 직접 값으로 전달되고 호출자가 ActiveX 컨테이너인 경우에는 XML 형식 문자열로 직렬화되어 전달됩니다.

외부 API의 XML 형식

Flash Player 9 이상, Adobe AIR 1.0 이상

Shockwave Flash ActiveX 컨트롤을 호스팅하는 ActionScript와 응용 프로그램 간의 통신은 특정 XML 형식을 사용하여 함수 호출과 값을 인코딩합니다. 외부 API에서 사용하는 XML 형식에는 두 부분이 있습니다. 한 가지 형식은 함수 호출을 나타내는 데 사용됩니다. 다른 형식은 개별 값을 나타내는 데 사용됩니다. 이 형식은 함수의 매개 변수와 함수 반환 값에 사용됩니다. 함수 호출의 XML 형식은 ActionScript에서 송수신되는 호출에 사용됩니다. ActionScript에서 함수를 호출하는 경우에는 Flash Player가 XML을 컨테이너에 전달하고, 컨테이너에서 호출하는 경우에는 Flash Player에서 컨테이너 응용 프로그램이 이 형식으로 XML 문자열을 전달해야 합니다. 다음 XML 프래그먼트는 XML 형식의 함수 호출을 보여 줍니다.

```
<invoke name="functionName" returntype="xml">
    <arguments>
        ... (individual argument values)
    </arguments>
</invoke>
```

루트 노드는 invoke 노드입니다. 이 노드에는 호출할 함수 이름을 나타내는 name과 항상 xml인 returntype의 두 가지 특성이 있습니다. 함수 호출에 매개 변수가 포함되는 경우 invoke 노드에 자식 arguments 노드가 포함되고 그 노드의 자식 노드는 아래 설명하는 개별 값 형식의 매개 변수 값이 됩니다.

함수 매개 변수와 함수 반환 값 등의 개별 값은 실제 값뿐 아니라 데이터 형식 정보가 포함된 형식 지정 스키마를 사용합니다. 다음 표에는 ActionScript 클래스 및 해당 데이터 유형의 값을 인코딩하는 데 사용되는 XML 형식이 나와 있습니다.

ActionScript 클래스/값	C# 클래스/값	형식	설명
null	null	<null/>	
Boolean true	bool true	<true/>	
Boolean false	bool false	<false/>	
String	string	<string>문자열 값</string>	
Number, int, uint	single, double, int, uint	<number>27.5</number> <number>-12</number>	

ActionScript 클래스/값	C# 클래스/값	형식	설명
Array(요소는 혼합 유형일 수 있습니다.)	ArrayList 또는 object[] 등 혼합 유형 요소를 허용하는 컬렉션	<array> <property id="0"> <number>27.5</number> </property> <property id="1"> <string>Hello there!</string> </property> ... </array>	property 노드는 개별 요소를 정의하고 id 특성은 숫자 0부터 시작하는 인덱스입니다.
Object	문자열 키가 있는 HashTable처럼 문자열 키와 객체 값이 있는 사전	<object> <property id="name"> <string>John Doe</string> </property> <property id="age"> <string>33</string> </property> ... </object>	property 노드는 개별 속성을 정의하고 id 특성은 속성 이름(문자열)입니다.
기타 기본 제공 또는 사용자 정의 클래스		<null/> or <object></object>	ActionScript에서는 다른 객체를 null 또는 빈 객체로 인코딩합니다. 두 경우 모두 속성 값이 사라집니다.

참고: 이 표에서는 예제로 ActionScript 클래스와 함께 해당 C# 클래스를 보여 줍니다. 그러나 외부 API는 ActiveX 컨트롤을 지원하는 모든 프로그래밍 언어 또는 런타임 통신에 사용될 수 있으며 C# 응용 프로그램에만 사용하는 것이 아닙니다.

외부 API와 ActiveX 컨테이너 응용 프로그램을 사용하여 직접 응용 프로그램을 작성할 때는 네이티브 함수 호출을 직렬화된 XML 형식으로 변환하는 작업을 수행할 프록시를 작성하는 것이 편하다는 것을 알게 될 것입니다. C#으로 작성된 proxy 클래스 예제는 ExternalInterfaceProxy 클래스 내부를 참조하십시오.

외부 API 예제: 웹 브라우저에서 ActionScript와 JavaScript 간의 통신

Flash Player 9 이상, Adobe AIR 1.0 이상

이 샘플 응용 프로그램은 다른 사람과 채팅이 가능한 인스턴트 메시징 응용 프로그램(응용 프로그램 이름은 Introvert IM)을 사용할 때 웹 브라우저의 ActionScript와 JavaScript 간의 적절한 통신 기법을 보여 줍니다. 외부 API를 사용하여 웹 페이지와 SWF 인터페이스의 HTML 양식 간에 메시지가 전송됩니다. 이 예제에서 보여 주는 기술은 다음과 같습니다.

- 통신을 설정하기 전에 브라우저가 통신 가능한 상태인지 확인하여 제대로 통신 초기화
- 컨테이너의 외부 API 지원 여부 확인
- ActionScript에서 JavaScript 함수 호출, 매개 변수 전달 및 응답으로 값 수신
- ActionScript 메서드를 JavaScript에서 호출할 수 있는 상태로 만들고 이러한 호출 수행

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. Introvert IM 응용 프로그램 파일은 Samples/IntrovertIM_HTML 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
IntrovertIMApp.fla 또는 IntrovertIMApp.mxml	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/introvertIM/IMManager.as	ActionScript 및 컨테이너 간에 통신을 구축하고 관리하는 클래스입니다.
com/example/programmingas3/introvertIM/IMMessageEvent.as	컨테이너에서 메시지를 수신할 때 IMManager 클래스에서 전달되는 사용자 정의 이벤트 유형입니다.
com/example/programmingas3/introvertIM/IMStatus.as	값이 응용 프로그램에서 선택할 수 있는 다른 "가용성" 상태 값을 나타내는 열거입니다.
html-flash/IntrovertIMApp.html 또는 html-template/index.template.html	Flash 응용 프로그램의 HTML 페이지(html-flash/IntrovertIMApp.html)이거나 Adobe Flex 응용 프로그램의 컨테이너 HTML 페이지(html-template/index.template.html) 작성에 사용된 템플릿입니다. 이 파일에는 응용 프로그램의 컨테이너 부분을 구성하는 모든 JavaScript 함수가 포함되어 있습니다.

ActionScript 브라우저 통신 준비

Flash Player 9 이상, Adobe AIR 1.0 이상

외부 API는 주로 ActionScript 응용 프로그램이 웹 브라우저와 통신할 수 있도록 허용하는 데 사용됩니다. 외부 API를 사용하면 ActionScript 메서드가 JavaScript에서 작성한 코드를 호출할 수 있으며 그 반대도 마찬가지입니다. 브라우저의 복잡성과 내부에서 페이지를 렌더링하는 방법으로 인해 HTML 페이지의 첫 번째 JavaScript를 실행하기 전에 SWF 문서에서 콜백을 등록하도록 보장할 수 있는 방법은 없습니다. 이러한 이유로 JavaScript에서 SWF 문서의 함수를 호출하기 전에 항상 SWF 문서에서 HTML 페이지를 호출하여 SWF 문서가 연결을 허용할 수 있는 상태임을 알려야 합니다.

예를 들어 IMManager 클래스에서 수행한 몇 가지 단계를 통해 Introvert IM에서 브라우저가 통신할 수 있는 상태인지 확인하고 통신을 위해 SWF 파일을 준비합니다. 브라우저가 통신 가능한 상태인지 확인하는 첫 번째 단계는 다음과 같이 IMManager 생성자에서 이루어집니다.

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

우선 이 코드는 ExternalInterface.available 속성을 사용하여 현재 컨테이너에서도 외부 API를 사용할 수 있는지 여부를 확인합니다. 가용성 상태인 경우 통신을 설정하는 프로세스가 시작됩니다. 외부 응용 프로그램과 통신을 시도하면 보안 예외 및 기타 오류가 발생할 수 있으므로 이 코드는 try 블록에서 래핑됩니다(편의상 해당 catch 블록은 샘플에서 생략).

그런 다음 이 코드는 여기에 나열된 isContainerReady() 메서드를 호출합니다.

```
private function isContainerReady():Boolean
{
    var result:Boolean = ExternalInterface.call("isReady");
    return result;
}
```

이제 isContainerReady() 메서드가 다시 ExternalInterface.call() 메서드를 사용하여 JavaScript 함수 isReady()를 호출합니다.

```
<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>
```

isReady() 함수는 jsReady 변수 값만 반환합니다. 처음에 이 변수는 false이며, 웹 페이지의 onload 이벤트가 트리거되면 변수 값이 true로 바뀝니다. 즉, 페이지가 로드되기 전에 ActionScript가 isReady() 함수를 호출하면 JavaScript가 ExternalInterface.call("isReady")에 false를 반환하고, 그 결과 ActionScript isContainerReady() 메서드가 false를 반환합니다. 페이지가 로드되면 JavaScript isReady() 함수가 true를 반환하므로 ActionScript isContainerReady() 메서드도 true를 반환합니다.

IMManager 생성자로 돌아가면 컨테이너의 준비 상태에 따라 두 가지 중 하나가 발생합니다. isContainerReady()가 true를 반환하면 코드가 setupCallbacks() 메서드만 호출하여 JavaScript와 통신을 설정하는 프로세스를 완료합니다. 그와 반대로 isContainerReady()가 false를 반환하면 프로세스는 기본적으로 보류됩니다. Timer 객체가 만들어지고 100밀리초마다 timerHandler() 메서드를 호출하도록 지시됩니다.

```
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();
    }
}
```

timerHandler() 메서드가 호출될 때마다 isContainerReady() 메서드의 결과를 다시 확인합니다. 컨테이너가 초기화되면 이 메서드는 true를 반환합니다. 그런 다음 이 코드는 Timer를 중지하고 setupCallbacks() 메서드를 호출하여 브라우저와의 통신 설정 프로세스를 마칩니다.

JavaScript에 ActionScript 메서드 표시

Flash Player 9 이상, Adobe AIR 1.0 이상

앞의 예제와 같이, 브라우저가 대기 상태임을 확인한 코드는 setupCallbacks() 메서드를 호출합니다. 이 메서드는 ActionScript를 준비하여 다음과 같이 JavaScript에서 호출을 수신합니다.

```
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");
}
```

setCallBacks() 메서드는 ExternalInterface.addCallback()을 호출하여 JavaScript에서 호출 가능한 두 개의 메서드를 등록함으로써 컨테이너와 통신할 준비를 마칩니다. 이 코드의 첫 번째 매개 변수는 ActionScript의 메서드 이름과 같습니다. 이 메서드는 JavaScript에 이 이름으로 인식됩니다("newMessage" 및 "getStatus"). 이 경우 다른 이름을 사용해도 별 다른 장점이 없으므로 간편하게 같은 이름을 다시 사용합니다. 마지막으로 ExternalInterface.call() 메서드를 사용하여 JavaScript 함수 setSWFIsReady()를 호출합니다. 이 함수는 컨테이너에 ActionScript 함수가 등록되었음을 알려 줍니다.

ActionScript에서 브라우저로 통신

Flash Player 9 이상, Adobe AIR 1.0 이상

Introvert IM 응용 프로그램은 컨테이너 페이지에서 JavaScript 함수를 호출하는 다양한 예를 보여 줍니다. 가장 간단한 경우(setupCallbacks() 메서드의 예제)는 매개 변수를 전달하거나 반환값을 수신하지 않고 JavaScript 함수 setSWFIsReady()를 호출하는 것입니다.

```
ExternalInterface.call("setSWFIsReady");
```

isContainerReady()의 다른 예제에서 ActionScript는 isReady() 함수를 호출하고 그 응답으로 부울 값을 수신합니다.

```
var result:Boolean = ExternalInterface.call("isReady");
```

외부 API를 사용하여 매개 변수를 JavaScript 함수로 전달할 수도 있습니다. 예를 들어 IMManager 클래스의 sendMessage() 메서드를 생각해 보십시오. 이 메서드는 사용자가 "대화 상대"에게 새 메시지를 보낼 때 호출됩니다.

```
public function sendMessage(message:String):void
{
    ExternalInterface.call("newMessage", message);
}
```

여기에서도 지정된 JavaScript 함수를 호출하고 브라우저에 새 메시지를 알리는 데 ExternalInterface.call()이 사용됩니다. 뿐만 아니라 메시지 자체가 추가 매개 변수로 ExternalInterface.call()에 전달되고, 그 결과 JavaScript 함수 newMessage()에 매개 변수로 전달됩니다.

JavaScript에서 ActionScript 코드 호출

Flash Player 9 이상, Adobe AIR 1.0 이상

통신은 두 갈래 길이라고 할 수 있으며 Introvert IM 응용 프로그램도 예외는 아닙니다. Flash Player IM 클라이언트만 JavaScript를 호출하여 메시지를 보내는 게 아니라 HTML 양식도 JavaScript 코드를 호출하여 SWF 파일에 메시지를 보내고 SWF 파일에서도 정보를 요구합니다. 예를 들어 SWF 파일이 컨테이너에 연락처 구축이 끝나 통신할 수 있는 상태가 되었음을 알리면 브라우저에서 처음 하는 일은 IMManager 클래스의 getStatus() 메서드를 호출하여 SWF IM 클라이언트에서 첫 사용자의 가용성 상태를 검색하는 것입니다. 웹 페이지에서 updateStatus() 함수로 다음과 같이 수행하면 됩니다.

```
<script language="JavaScript">
...
function updateStatus()
{
    if (swfReady)
    {
        var currentStatus = getSWF("IntrovertIMApp").getStatus();
        document.forms["imForm"].status.value = currentStatus;
    }
}
...
</script>
```

이 코드는 swfReady 변수의 값을 확인합니다. 이 값은 SWF 파일이 ExternalInterface 클래스로 메서드를 등록했음을 브라우저에 알렸는지 여부를 추적합니다. SWF 파일이 통신을 수신할 수 있는 상태가 되면 다음 줄(var currentStatus = ...)이 IMManager 클래스에서 실제로 getStatus() 메서드를 호출합니다. 코드의 이 줄에서는 다음 세 가지가 발생합니다.

- getSWF() JavaScript 함수가 호출되어 SWF 파일을 나타내는 JavaScript 객체에 대한 참조를 반환합니다. getSWF()에 전달된 매개 변수는 HTML 페이지에 둘 이상의 SWF 파일이 있는 경우 브라우저 객체가 반환되는지 여부를 결정합니다. 이 매개 변수에 전달된 값은 SWF 파일을 포함하는 데 사용되는 object 태그의 id 특성 및 embed 태그의 name 특성과 일치해야 합니다.
- SWF 파일에 대한 참조를 사용하면 getStatus() 메서드가 SWF 객체의 메서드인 것처럼 호출됩니다. 이 경우 함수 이름 "getStatus"가 사용됩니다. ExternalInterface.addCallback()을 사용하여 ActionScript 함수를 등록할 때 이 이름을 사용합니다.
- getStatus() ActionScript 메서드는 값을 반환하고 이 값이 currentStatus 변수에 지정됩니다. 그 다음 이 변수는 status 텍스트 필드의 내용(value 속성)으로 지정됩니다.

참고: 코드를 따라 실행하다 보면 updateStatus() 함수에 대한 소스 코드에서 getSWF() 함수를 호출하는 코드 행이 실제로는 var currentStatus = getSWF("${application}").getStatus()로 작성되어 있음을 알 수 있습니다. ${application} 텍스트는 HTML 페이지 템플릿 내의 자리 표시자이며, Adobe Flash Builder에서 응용 프로그램의 실제 HTML 페이지를 생성할 때 이 자리 표시자 텍스트는 object 태그의 id 특성과 embed 태그의 name 특성으로 사용되는 것과 동일한 텍스트(이 예제의 경우 IntrovertIMApp)로 대체됩니다. 이 값은 getSWF() 함수에 필요한 값입니다.

sendMessage() JavaScript 함수는 ActionScript 함수에 매개 변수를 전달하는 방법을 보여 줍니다. sendMessage()는 HTML 페이지에서 사용자가 [전송] 버튼을 누를 때 호출되는 함수입니다.

```
<script language="JavaScript">
...
function sendMessage(message)
{
    if (swfReady)
    {
        ...
        getSWF("IntrovertIMApp").newMessage(message);
    }
}
...
</script>
```

newMessage() ActionScript 메서드에는 매개 변수가 하나 필요하므로 JavaScript 코드의 newMessage() 메서드를 호출할 때 JavaScript message 변수를 매개 변수로 사용하여 ActionScript에 전달합니다.

브라우저 유형 검색

Flash Player 9 이상, Adobe AIR 1.0 이상

브라우저마다 내용에 액세스하는 방법이 다르므로, 다음 예제의 getSWF() JavaScript 함수에서 보듯이 항상 JavaScript로 사용자가 실행하고 있는 브라우저가 무엇인지 검색하고 해당 브라우저의 구문에 따라 Window 객체나 Document 객체를 사용하여 동영상에 액세스해야 합니다.

```
<script language="JavaScript">
...
function getSWF(movieName)
{
    if (navigator.appName.indexOf("Microsoft") != -1)
    {
        return window[movieName];
    }
    else
    {
        return document[movieName];
    }
}
...
</script>
```

스크립트에서 사용자의 브라우저 종류를 검색하지 않으면 HTML 컨테이너에서 SWF 파일을 재생할 때 예상치 못한 비헤이비어가 표시될 수도 있습니다.

48장: AIR의 XML 서명 유효성 검사

Adobe AIR 1.5 이상

Adobe® AIR® XMLSignatureValidator API의 클래스를 사용하면 XML 서명 구문 및 처리를 위한 W3C 권장 사항의 하위 집합(http://http://www.w3.org/TR/xmldsig-core/)을 따르는 디지털 서명의 유효성을 검사할 수 있습니다. XML 서명을 사용하면 데이터 또는 정보의 무결성 및 서명자 신원을 확인할 수 있습니다.

XML 서명을 사용하면 응용 프로그램에서 다운로드한 메시지 또는 리소스의 유효성을 검사할 수 있습니다. 예를 들어 구독 서비스를 제공하는 응용 프로그램의 경우 구독 계약을 서명된 XML 문서에 캡슐화할 수 있습니다. 누군가가 구독 문서를 변경하려고 할 경우 유효성 검사에 실패하게 됩니다.

또한 XML 서명을 사용하면 리소스 다이제스트가 들어 있는 서명된 목록을 포함시켜 응용 프로그램에서 사용하는 다운로드한 리소스의 유효성을 검사할 수 있습니다. 응용 프로그램에서는 서명된 파일의 목록과 로드된 바이트에서 계산된 목록을 비교하여 리소스가 변경되지 않았는지 확인할 수 있습니다. 이것은 다운로드한 리소스가 응용 프로그램 보안 샌드박스에서 실행할 SWF 파일이거나 기타 활성 내용인 경우에 특히 유용합니다.

XML 서명 유효성 검사의 기초

Adobe AIR 1.5 이상

XML 서명의 유효성 검사에 대한 간략한 설명과 코드 예제를 보려면 Adobe Developer Connection의 다음 퀵 스타트 문서를 참조하십시오.

- XML 서명 만들기 및 유효성 검사(Flex)
- XML 서명 만들기 및 유효성 검사(Flash)

Adobe® AIR®는 XML 서명의 유효성을 검사하는 데 사용되는 XMLSignatureValidator 클래스 및 IURIDereferencer 인터페이스를 제공합니다. XMLSignatureValidator 클래스에서 사용할 수 있는 XML 구문은 XML 서명 구문 및 처리를 위한 W3C 권장 사항의 하위 집합입니다. (권장 사항의 하위 집합만 지원되므로 합법적인 서명을 모두 유효성 검사할 수 있는 것은 아닙니다.) AIR에서는 XML 서명을 만들기 위한 API를 제공하지 않습니다.

XML 서명 유효성 검사 클래스

Adobe AIR 1.5 이상

XML 서명 유효성 검사 API에는 다음과 같은 클래스가 포함되어 있습니다.

패키지	클래스
flash.security	• XMLSignatureValidator • IURIDereferencer(인터페이스) XMLSignatureValidator 문자열 상수는 다음과 같은 클래스에서 정의됩니다. • ReferencesValidationSetting • RevocationCheckSettings • SignatureStatus • SignerTrustSettings
flash.events	• Event • ErrorEvent

XML 서명 유효성 검사 클래스 사용

Adobe AIR 1.5 이상

XMLSignatureValidator 클래스를 사용하여 XML 서명의 유효성을 검사하려면 다음과 같은 작업을 수행해야 합니다.

- XMLSignatureValidator 객체를 만듭니다.
- IURIDereferencer 인터페이스의 구현을 제공합니다. XMLSignatureValidator 객체에서 IURIDereferencer dereference() 메서드를 호출하여 서명의 각 참조에 대한 URI를 전달합니다. dereference() 메서드는 URI를 확인하고 참조된 데이터(서명과 동일한 문서 또는 외부 리소스에 포함되어 있을 수 있음)를 반환해야 합니다.
- 응용 프로그램에 적합한 XMLSignatureValidator 객체의 인증서 신뢰, 해지 확인 및 참조 유효성 검사 설정을 설정합니다.
- complete 및 error 이벤트에 대한 이벤트 리스너를 추가합니다.
- verify() 메서드를 호출하여 유효성을 검사할 서명을 전달합니다.
- complete 및 error 이벤트를 처리하고 결과를 해석합니다.

다음 예제에서는 XML 서명의 유효성을 검사하는 validate() 함수를 구현합니다. 서명 인증서가 시스템 신뢰 저장소에 있거나 신뢰 저장소의 인증서에 체인으로 연결되도록 XMLSignatureValidator 속성을 설정합니다. 또한 예제에서는 이름이 **XMLDereferencer**인 적합한 IURIDereferencer 클래스가 있다고 가정합니다.

```
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);
}
```

XML 서명 유효성 검사 프로세스

Adobe AIR 1.5 이상

XMLSignatureValidator verify() 메서드를 호출할 때 AIR에서는 다음 단계를 수행합니다.

- 서명 인증서의 공개 키를 사용하여 서명의 암호화 무결성을 확인합니다.
- XMLSignatureValidator 객체의 현재 설정을 기반으로 하여 인증서의 암호화 무결성, ID 및 신뢰성을 설정합니다.

  서명 인증서에 설정된 신뢰는 유효성 검사 프로세스 무결성의 핵심 요소입니다. 서명 유효성 검사는 올바르게 정의된 암호화 프로세스를 사용하여 수행되지만 서명 인증서의 신뢰성은 알고리즘 방식으로 내릴 수 없는 판단입니다.

  일반적으로 다음과 같은 세 가지 방법으로 인증서를 신뢰할 수 있는지 여부를 판단할 수 있습니다.

  - 인증 기관 및 운영 체제 신뢰 저장소 사용
  - 서명자에게서 직접 인증서 복사본, 인증서의 신뢰 앵커로 사용되는 다른 인증서 또는 공개 키와 같이 인증서를 안정적으로 식별할 수 있는 정보 가져오기
  - 응용 프로그램의 최종 사용자에게 인증서를 신뢰하는지 여부 문의 인증서에 있는 식별 정보를 원래 신뢰할 수 없으므로 이러한 쿼리는 자체 서명된 인증서에 유효하지 않습니다.
- 서명된 데이터의 암호화 무결성을 확인합니다.

서명된 데이터는 IURIDereferencer 구현을 통해 확인합니다. 서명 문서의 각 참조에 대해 IURIDereferencer 구현 dereference() 메서드가 호출됩니다. dereference() 메서드에서 반환한 데이터를 사용하여 참조 다이제스트를 계산할 수 있습니다. 이 다이제스트 값은 서명 문서에 기록된 다이제스트와 비교됩니다. 다이제스트가 일치하는 경우 데이터는 서명된 이후 변경되지 않았습니다.

XML 서명의 유효성 검사 결과를 사용할 때 중요하게 고려해야 할 하나의 사항은 서명한 항목만 안전하다는 것입니다. 예를 들어 패키지에 있는 파일을 나열하는 서명된 목록을 살펴보십시오. XMLSignatureValidator에서 서명을 확인할 때는 목록 자체가 변경되지 않았는지만 확인합니다. 파일에 있는 데이터에 서명하지 않았기 때문에 목록에서 참조하는 파일을 변경하거나 삭제하는 경우 계속 서명의 유효성을 검사합니다.

참고: 이러한 목록에 있는 파일을 확인하기 위해 표시에서 사용하는 동일한 해시 알고리즘을 사용하여 파일 데이터의 다이제스트를 계산하고 결과를 서명된 표시에 저장된 다이제스트와 비교할 수 있습니다. 추가 파일의 존재 여부를 확인해야 하는 경우도 있습니다.

유효성 검사 결과 해석

Adobe AIR 1.5 이상

유효성 검사 결과는 XMLSignatureValidator 객체의 상태 속성에 의해 보고됩니다. 이러한 속성은 유효성 검사기 객체에서 **complete** 이벤트를 전달한 후 읽을 수 있습니다. 네 가지 상태 속성에는 validityStatus, digestStatus, identityStatus 및 referencesStatus가 있습니다.

validityStatus 속성

Adobe AIR 1.5 이상

validityStatus 속성은 서명의 전체 유효성을 보고합니다. validityStatus는 기타 세 가지 상태 속성의 상태에 따라 다르며 다음 값 중 하나를 가질 수 있습니다.

- valid - digestStatus, identityStatus 및 referencesStatus 모두가 valid인 경우
- invalid - 개별 상태 속성 중 하나 이상이 invalid인 경우
- unknown — 개별 상태 속성 중 하나 이상이 unknown이고 각 상태 중 하나도 invalid가 아닌 경우

digestStatus 속성

Adobe AIR 1.5 이상

digestStatus 속성은 메시지 다이제스트의 암호화 확인 결과를 보고합니다. digestStatus 속성은 다음 값 중 하나를 가질 수 있습니다.

- valid - 서명 문서 자체가 서명 이후 변경되지 않은 경우
- invalid — 서명 문서가 변경되거나 잘못된 경우
- unknown - verify() 메서드가 오류 없이 완료되지 않은 경우

identityStatus 속성

Adobe AIR 1.5 이상

identityStatus 속성은 서명 인증서의 상태를 보고합니다. 이 속성의 값은 다음과 같은 여러 요인에 따라 다릅니다.

- 인증서의 암호화 무결성
- 인증서가 만료 또는 해지되었는지 여부
- 현재 시스템에서 인증서를 신뢰할 수 있는지 여부

- XMLSignatureValidator 객체의 상태(예: 신뢰 체인을 구축하는 데 추가 인증서가 추가되었는지 여부, 이러한 인증서를 신뢰할 수 있는지 여부, useSystemTrustStore 및 revocationCheckSettings 속성의 값)

identityStatus는 다음과 같은 값을 가질 수 있습니다.

- valid - 서명 인증서가 유효한 것으로 간주되려면 다음 조건을 충족해야 합니다.
  - 서명 인증서를 변경하지 않아야 합니다.
  - 서명 인증서가 만료되거나 해지되지 않아야 합니다. 단, 서명에 유효한 타임스탬프가 있는 경우는 예외입니다. 서명에 타임스탬프가 있는 경우 인증서는 문서 서명 시 유효했던 경우 유효한 것으로 간주됩니다. 타임스탬프에 서명하기 위해 타임스탬프 서비스에서 사용하는 인증서는 사용자 컴퓨터에 있는 신뢰할 수 있는 루트 인증서에 연결되어야 합니다.
  - 서명 인증서를 신뢰합니다. 인증서가 시스템 신뢰 저장소에 있거나 시스템 신뢰 저장소의 다른 인증서에 체인으로 연결되어 있으며 useSystemTrustStore 속성을 true로 설정하는 경우 인증서를 신뢰합니다. 또한 XMLSignatureValidator 객체의 addCertificate() 메서드를 사용하여 인증서를 신뢰한 것으로 지정할 수 있습니다.
  - 사실 인증서는 서명 인증서입니다.
- invalid - 인증서가 만료되었거나 해지되었습니다. 서명 시 유효성을 제공하는 타임스탬프가 없거나 인증서가 변경되었습니다.
- unknown - 인증서가 잘못된 것은 아니지만 신뢰할 수 없습니다. 예를 들어 자체 서명된 인증서는 unknown으로 보고됩니다(명시적으로 신뢰하지 않는 경우). verify() 메서드를 오류 없이 완료하지 않았거나 서명 다이제스트가 잘못되어 ID를 확인하지 않은 경우 identityStatus도 unknown으로 보고됩니다.

referencesStatus 속성
Adobe AIR 1.5 이상

referencesStatus 속성은 서명의 SignedData 요소에 있는 참조의 암호화 무결성을 보고합니다.

- valid - 서명에 있는 모든 참조의 계산된 다이제스트가 XML 서명에 기록된 해당 다이제스트와 일치하는 경우 valid 상태는 서명된 데이터가 변경되지 않았음을 나타냅니다.
- invalid - 계산된 다이제스트가 서명에 있는 해당 다이제스트와 일치하지 않는 경우
- unknown - 참조 다이제스트를 확인하지 않은 경우 전체 서명 다이제스트가 invalid이거나 서명 인증서가 잘못된 경우 참조를 확인하지 않습니다. identityStatus가 unknown인 경우 참조는 referencesValidationSetting이 validOrUnknown인 경우에만 확인됩니다.

XML 서명

Adobe AIR 1.5 이상

XML 서명은 XML 구문으로 표시되는 디지털 서명입니다. XML 서명의 데이터를 사용하면 서명된 정보를 서명한 이후 변경하지 않았음을 확인할 수 있습니다. 또한 신뢰할 수 있는 인증 기관에서 서명 인증서를 발급한 경우 서명자의 신원은 공용 키 인프라를 통해 확인할 수 있습니다.

XML 서명은 모든 형식의 디지털 데이터(이진 또는 XML 형식)에 적용할 수 있습니다. XML 서명은 일반적으로 다음과 같은 용도로 사용됩니다.

- 외부 또는 다운로드한 리소스를 수정했는지 여부 확인
- 알 수 있는 소스에서 가져온 메시지인지 확인
- 응용 프로그램 사용권 또는 구독 권한의 유효성 검사

지원되는 XML 서명 구문

Adobe AIR 1.5 이상

AIR에서는 XML 서명 구문 및 처리를 위한 W3C 권장 사항의 다음과 같은 요소를 지원합니다.

- 모든 핵심 서명 구문 요소(W3C 권장 사항 문서의 4항) - 단, KeyInfo 요소는 완전히 지원되지 않습니다.
- KeyInfo 요소는 X509Data 요소만 포함해야 합니다.
- X509Data 요소는 X509Certificate 요소만 포함해야 합니다.
- SHA256 다이제스트 메서드
- RSA-SHA1(PKCS1) 서명 알고리즘
- "주석 없는 정규 XML" 정규화 메서드 및 변형
- 엔벌로프된 서명 변형
- 타임스탬프

다음 문서에서는 일반적인 XML 서명을 보여 줍니다. 대부분의 암호화 데이터는 간단한 예제를 위해 제거되었습니다.

```
<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>
```

서명의 주요 요소는 다음과 같습니다.

- SignedInfo - 서명한 때의 서명된 데이터 및 계산된 다이제스트 값에 대한 참조를 포함합니다. 서명된 데이터 자체는 XML 서명과 동일한 문서에 포함될 수 있으며 외부 요소일 수 있습니다.
- SignatureValue - 서명자의 개인 키를 사용하여 암호화된 SignedInfo 요소의 다이제스트를 포함합니다.
- KeyInfo - 서명 인증서는 물론 신뢰 체인을 설정하는 데 필요한 추가 인증서도 포함합니다. 기술적으로는 KeyInfo 요소는 선택 사항이지만 이 요소가 포함되어 있지 않으면 서명의 유효성을 검사할 수 없습니다.

다음과 같은 세 가지 일반적인 XML 서명 유형이 있습니다.

- 엔벌로프된 서명 - 서명이 서명하는 XML 데이터 내부에 삽입됩니다.
- 엔벌로프 중인 서명 - 서명된 XML 데이터가 Signature 요소 내의 Object 요소 내에 포함됩니다.
- 분리된 서명 - 서명된 데이터가 XML 서명의 외부 요소입니다. 서명된 데이터가 외부 파일에 있을 수 있습니다. 또는 Signature 요소의 부모 또는 자식 요소가 아닌 서명과 동일한 XML 문서에 있을 수 있습니다.

XML 서명은 URI를 사용하여 서명된 데이터를 참조합니다. 서명 및 유효성 검사 응용 프로그램은 이러한 URI를 확인하는 데 동일한 규칙을 사용해야 합니다. XMLSignatureValidator 클래스를 사용할 때 IURIDereferencer 인터페이스의 구현을 제공해야 합니다. 이러한 구현이 URI를 확인하고 서명된 데이터를 ByteArray 객체로 반환합니다. 반환된 ByteArray 객체는 서명의 다이제스트를 생성한 동일한 알고리즘을 사용하여 다이제스트됩니다.

인증서 및 신뢰

Adobe AIR 1.5 이상

인증서는 공개 키, 식별 정보, 인증 발급 기관에 속한 하나 이상의 인증서로 구성되어 있습니다.

두 가지 방법을 사용하여 인증서의 신뢰를 설정할 수 있습니다. 물리적 미디어에서 또는 SSL 트랜잭션과 같은 안전한 디지털 전송을 통해 서명자에게서 직접 인증서 복사본을 얻어 신뢰를 설정할 수 있습니다. 또한 인증 기관을 통해 서명 인증서의 신뢰 여부를 확인할 수 있습니다.

인증 기관을 이용하려면 서명의 유효성을 검사하는 컴퓨터에서 신뢰할 수 있는 기관에서 서명 인증서를 발급해야 합니다. 대부분의 운영 체제 제조업체는 많은 인증 기관의 루트 인증서를 운영 체제 신뢰 저장소에 보관합니다. 또한 사용자는 저장소에서 인증서를 추가 및 제거할 수 있습니다.

신뢰할 수 있는 인증 기관에서 인증서를 발급한 경우에도 인증서가 신뢰할 수 있는 사람에 속하는지 여부를 결정해야 합니다. 대부분의 경우 최종 사용자가 이러한 결정을 내려야 합니다. 예를 들어 AIR 응용 프로그램이 설치되어 있는 경우 AIR 설치 프로그램에서는 응용 프로그램 설치를 원하는지 여부를 사용자가 확인하도록 요청할 때 게시자 인증서의 식별 정보를 표시합니다. 다른 경우 공개 키 또는 기타 인증서 정보를 허용되는 키 목록과 비교해야 할 수 있습니다. 이러한 목록은 변경되지 않도록 자체 서명을 사용하거나 AIR로 암호화된 로컬 저장소에 저장하여 보안을 유지해야 합니다.

참고: 서명이 "자체 서명"된 경우와 같이 독립적인 확인 없이 서명 인증서를 신뢰하도록 선택할 수 있는 경우에는 서명을 확인하여 중요한 사항을 많이 확인할 수는 없습니다. 서명을 만든 사람을 모르는 경우 서명이 변경되지 않았다는 확인 자체는 그리 중요하지 않습니다. 서명이 유효하게 서명된 위조일 가능성이 있습니다.

인증서 만료 및 해지

Adobe AIR 1.5 이상

모든 인증서는 만료됩니다. 또한 인증서는 인증서와 관련된 개인 키가 손상되거나 도난당한 경우 인증 발급 기관에 의해 해지될 수 있습니다. 만료되거나 해지된 인증서를 사용하여 서명을 서명한 경우 해당 서명은 타임스탬프가 서명의 일부로 포함되지 않으면 유효하지 않은 서명으로 보고됩니다. 타임스탬프가 있는 경우 XMLSignatureValidator 클래스는 인증서가 서명 시 유효한 경우 서명의 유효성을 검사합니다.

타임스탬프는 데이터가 특정 시간 및 날짜에 서명되었음을 인증하는 타임스탬프 서비스의 서명된 디지털 메시지입니다. 타임스탬프는 타임스탬프 기관에 의해 발급되고 타임스탬프 기관의 자체 인증서에 의해 서명됩니다. 타임스탬프가 유효한 것으로 간주되려면 현재 시스템에서 타임스탬프에 포함된 타임스탬프 기관 인증서를 신뢰해야 합니다. XMLSignatureValidator는 타임스탬프의 유효성을 검사할 때 사용할 다른 인증서를 지정하는 데 API를 제공하지 않습니다.

IURIDereferencer 인터페이스 구현

Adobe AIR 1.5 이상

XML 서명의 유효성을 검사하려면 IURIDereferencer 인터페이스의 구현을 제공해야 합니다. 구현은 XML 서명 문서의 Reference 요소 내에 있는 URI를 확인하며 다이제스트를 계산할 수 있도록 데이터를 반환합니다. 계산된 다이제스트는 서명을 만든 이후 참조된 데이터를 변경했는지 여부를 확인하기 위해 서명의 다이제스트와 비교합니다.

참고: HTML 기반 AIR 응용 프로그램은 XML 서명의 유효성을 검사하기 위해 ActionScript 구현을 포함하는 SWF 라이브러리를 가져와야 합니다. IURIDereferencer 인터페이스는 JavaScript에서 구현할 수 없습니다.

IURIDerefencer 인터페이스는 구현해야 할 단일 메서드인 dereference(uri:String)를 포함합니다. XMLSignatureValidator 객체는 서명의 각 참조에 대해 이 메서드를 호출합니다. 메서드는 ByteArray 객체의 데이터를 반환해야 합니다.

대부분의 경우 역참조자 객체가 참조된 데이터를 찾을 수 있도록 속성 또는 메서드도 추가해야 합니다. 예를 들어 서명된 데이터가 서명과 동일한 문서에 있는 경우 XML 문서에 대한 참조를 제공하는 멤버 변수를 추가할 수 있습니다. 그렇게 하면 dereference() 메서드가 이 변수를 URI와 함께 사용하여 참조된 데이터를 찾을 수 있습니다. 마찬가지로 서명된 데이터가 로컬 파일 시스템의 디렉토리에 있는 경우 dereference() 메서드에는 참조된 파일을 확인하기 위해 해당 디렉토리에 대한 경로를 제공하는 속성이 필요할 수 있습니다.

XMLSignatureValidator는 URI 문자열을 해석할 때 전적으로 역참조자를 이용합니다. URI 역참조에 대한 일반 규칙은 XML 서명 구문 및 처리에 대한 W3C 권장 사항의 4.3.3항에 나와 있습니다.

엔벌로프된 서명에서 URI 역참조

Adobe AIR 1.5 이상

엔벌로프된 XML 서명을 생성하면 서명 요소가 서명된 데이터에 삽입됩니다. 예를 들어 엔벌로프된 서명 구조를 사용하여 다음 메시지에 서명한 경우

```
<message>
    <data>...</data>
</message>
```

서명된 문서 결과는 다음과 같습니다.

```
<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>
```

서명에 해당 URI로 빈 문자열이 있는 단일 Reference 요소가 포함되어 있습니다. 이 컨텍스트에서 빈 문자열은 문서의 루트를 나타냅니다.

또한 변형 알고리즘은 엔벌로프된 서명 변형이 적용되었음을 지정합니다. 엔벌로프된 서명 변형이 적용된 경우 XMLSignatureValidator는 다이제스트를 계산하기 전에 자동으로 문서에서 서명을 제거합니다. 즉 역참조자는 데이터를 반환할 때 Signature 요소를 제거할 필요가 없습니다.

다음 예제에서는 엔벌로프된 서명에 대한 역참조자를 보여 줍니다.

```
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;
            }
        }
    }
}
```

이 역참조자 클래스는 signedMessage 매개 변수와 함께 생성자 함수를 사용하여 dereference() 메서드에서 사용할 수 있는 엔벌로프된 서명 문서를 만들 수 있습니다. 엔벌로프된 서명의 참조는 항상 서명된 데이터의 루트를 참조하므로 dereferencer() 메서드는 문서를 바이트 배열에 작성하고 반환합니다.

엔벌로프 중인 서명 및 분리된 서명에서 URI 역참조

Adobe AIR 1.5 이상

서명된 데이터가 서명 자체와 동일한 문서에 있는 경우 참조의 URI는 일반적으로 XPath 또는 XPointer 구문을 사용하여 서명된 요소를 처리합니다. XML 서명 구문 및 처리를 위한 W3C 권장 사항에서는 이 구문만 권장하므로 발견할 수 있는 서명을 기반으로 구현하고 지원되지 않는 구문을 적절히 처리할 수 있는 오류 확인을 추가해야 합니다.

AIR 응용 프로그램의 서명은 엔벌로프 중인 서명의 예입니다. 응용 프로그램의 파일은 Manifest 요소에 나열됩니다. Manifest 요소는 Manifest 요소의 ID를 나타내는 “#PackageContents” 문자열을 사용하여 Reference URI 특성에서 처리합니다.

```
<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>
```

이 서명의 유효성을 검사하는 데 사용되는 역참조자는 Reference 요소의 "#PackageContents"를 포함하는 URI 문자열을 사용하고 ByteArray 객체의 Manifest 요소를 반환해야 합니다. “#” 심볼은 요소 ID 특성의 값을 나타냅니다.

다음 예제에서는 AIR 응용 프로그램 서명의 유효성을 검사하는 데 사용되는 역참조자를 구현합니다. 구현은 AIR 서명의 알려진 구조를 이용하여 간단하게 유지됩니다. 일반적인 용도의 역참조자는 매우 복잡할 수 있습니다.

```
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;
            }

        }
    }
}
```

이러한 유형의 서명을 확인할 때 Manifest 요소에 있는 데이터의 유효성만 검사합니다. 패키지의 실제 파일은 확인하지 않습니다. 패키지 파일의 변경 여부를 확인하려면 파일을 읽고, SHA256 다이제스트를 계산하고, 결과를 목록에 기록된 다이제스트와 비교해야 합니다. XMLSignatureValidator는 이러한 2차 참조를 자동으로 확인하지 않습니다.

참고: 이 예제는 서명 유효성 검사 프로세스를 보여 주기 위해서만 제공됩니다. 자체 서명의 유효성을 검사하기 위해 AIR 응용 프로그램에서 사용되는 경우는 거의 없습니다. 응용 프로그램이 이미 변경된 경우 변경하는 에이전트는 단순히 유효성 검사 확인을 제거할 수 있습니다.

외부 리소스에 대한 다이제스트 값 계산

Adobe AIR 1.5 이상

AIR에는 SHA256 다이제스트 계산을 위한 내장 함수가 없지만 Flex SDK에는 SHA256 유틸리티 클래스가 있습니다. SDK는 계산된 다이제스트를 서명에 저장된 다이제스트와 비교하는 데 유용한 Base64 인코더 유틸리티 클래스도 포함할 수 있습니다.

다음 예제 함수에서는 AIR 패키지 목록의 파일을 읽고 유효성을 검사합니다.

```
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;
}
```

이 함수는 Manifest 요소에 있는 모든 참조를 반복 탐색합니다. 각 참조에 대해 SHA256 다이제스트가 계산되고, base64 형식으로 인코딩되고, 목록의 다이제스트와 비교됩니다. AIR 패키지의 URI는 응용 프로그램 디렉토리에 상대적인 경로를 나타냅니다. 경로는 응용 프로그램 디렉토리 내의 META-INF 하위 디렉토리에 항상 있는 서명 파일의 위치를 기반으로 확인됩니다. Flex SHA256 클래스는 다이제스트를 16진수 문자의 문자열로 반환합니다. 이 문자열은 16진수 문자열이 나타내는 바이트를 포함하는 ByteArray로 변환되어야 합니다.

Flash CS4에서 mx.utils.SHA256 및 Base64Encoder 클래스를 사용하려면 이러한 클래스를 찾아서 응용 프로그램 개발 디렉토리에 복사하거나 Flex SDK를 사용하여 이 클래스를 포함하는 라이브러리 SWF를 컴파일합니다.

외부 데이터를 참조하는 분리된 서명에서 URI 역참조

Adobe AIR 1.5 이상

URI가 외부 리소스를 나타내는 경우 데이터에 액세스하여 ByteArray 객체로 로드해야 합니다. URI에 절대 URL이 포함되어 있는 경우 단순히 파일을 읽거나 URL을 요청하면 됩니다. 보다 일반적으로 URI에 절대 경로가 포함되어 있는 경우 IURIDereferencer 구현에는 서명된 파일에 대한 경로를 확인하는 방법이 포함되어 있어야 합니다.

다음 예제에서는 역참조자 인스턴스가 서명된 파일 확인을 위한 기반으로 생성된 경우 초기화된 File 객체를 사용합니다.

```
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;
            }
        }
    }
}
```

dereference() 함수는 간단하게 참조 URI에서 처리하는 파일을 찾고, 파일 내용을 바이트 배열로 로드하고, ByteArray 객체를 반환합니다.

참고: 원격 외부 참조의 유효성을 검사하기 전에 응용 프로그램이 “phone home” 또는 악의적으로 생성된 서명 문서에 의한 비슷한 유형의 공격에 취약하지 않은지 확인하십시오.

49장: 클라이언트 시스템 환경

Flash Player 9 이상, Adobe AIR 1.0 이상

이 장에서는 사용자의 시스템과 상호 작용하는 방법에 대해 설명하고 지원되는 기능을 확인하는 방법과 사용 가능한 경우 사용자 시스템에 설치된 IME(Input Method Editor)를 사용하여 다국어 응용 프로그램을 만드는 방법을 보여 줍니다. 또한 응용 프로그램 도메인의 일반적인 사용 방법을 보여 줍니다.

기타 도움말 항목

flash.system.System

flash.system.Capabilities

클라이언트 시스템 환경의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

다양한 기능을 갖춘 고급 응용 프로그램을 개발할수록 사용자 운영 체제 정보에 대해 보다 자세히 알고 그 기능에 액세스해야 할 필요성을 느끼게 될 것입니다. flash.system 패키지에는 다음과 같은 시스템 수준 기능에 액세스하는 데 사용되는 클래스 컬렉션이 포함되어 있습니다.

- 실행 중인 응용 프로그램 및 보안 도메인 코드 확인
- 화면 크기(해상도) 및 mp3 오디오 등의 일부 기능 사용 가능 여부와 같은 사용자 Flash 런타임(예: Flash® Player 또는 Adobe® AIR™)의 기능 확인
- IME를 통한 다국어 사이트 구축
- Flash 런타임 컨테이너(예: HTML 페이지 또는 컨테이너 응용 프로그램)와의 상호 작용
- 클립보드에 정보 저장

flash.system 패키지에는 IMEConversionMode 및 SecurityPanel 클래스도 포함되어 있습니다. 이러한 클래스에는 각각 IME 및 Security 클래스를 사용하는 정적 상수가 들어 있습니다.

중요한 개념 및 용어

다음 참조 목록에는 중요한 용어가 정리되어 있습니다.

운영 체제 컴퓨터에서 기타 모든 응용 프로그램이 실행되는 기본 프로그램으로, Microsoft Windows, Mac OS X, Linux® 등이 있습니다.

Clipboard 복사하거나 잘라낸 텍스트 또는 항목이 일시적으로 저장되는 운영 체제 내 컨테이너로, 이 컨테이너에서 응용 프로그램으로 해당 텍스트 또는 항목을 붙여넣습니다.

응용 프로그램 도메인 서로 다른 여러 SWF 파일에서 사용되는 클래스를 분류하는 메커니즘으로, SWF 파일에 이름이 동일한 여러 클래스가 포함될 경우 해당 클래스가 서로를 덮어쓰지 않도록 합니다.

IME(input method editor) 표준 키보드를 통해 복잡한 문자 또는 심볼을 입력하도록 지원하는 프로그램 또는 운영 체제 도구입니다.

클라이언트 시스템 프로그래밍 용어에서 클라이언트는 개인 컴퓨터에서 실행되고 단일 사용자가 사용하는 응용 프로그램의 부분 또는 전체를 의미합니다. 클라이언트 시스템은 사용자 컴퓨터의 기본 운영 체제입니다.

System 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

System 클래스에는 사용자의 운영 체제와 상호 작용하고 런타임의 현재 메모리 사용 현황을 검색하는 데 사용되는 메서드와 속성이 포함되어 있습니다. System 클래스의 메서드와 속성을 사용하는 경우에도 imeComposition 이벤트를 수신하거나, 사용자의 현재 코드 페이지를 사용하여 외부 텍스트 파일을 로드하거나 이러한 파일을 유니코드로 로드하도록 런타임에 명령하거나, 사용자 클립보드의 내용을 설정할 수 있습니다.

런타임에 사용자의 시스템에 대한 데이터 가져오기

Flash Player 9 이상, Adobe AIR 1.0 이상

System.totalMemory 속성을 확인하면 런타임에서 현재 사용 중인 메모리 양(바이트 단위)을 알 수 있습니다. 이 속성을 사용하면 메모리 사용 현황을 모니터링하고 메모리 레벨 변경 방식에 따라 응용 프로그램을 최적화할 수 있습니다. 예를 들어 특정 시각 효과로 인해 메모리 사용량이 크게 증가하는 경우 이 효과를 수정하거나 모두 제거할 수 있습니다.

System.ime 속성은 현재 설치되어 있는 IME(Input Method Editor)에 대한 참조입니다. 이 속성을 사용하면 addEventListener() 메서드를 사용하여 imeComposition 이벤트(flash.events.IMEEvent.IME_COMPOSITION)를 수신할 수 있습니다.

System 클래스의 세 번째 속성은 useCodePage입니다. useCodePage를 true로 설정하면 런타임은 운영 체제의 기존 코드 페이지를 사용하여 외부 텍스트 파일을 로드합니다. 이 속성을 false로 설정하면 런타임에 외부 파일을 유니코드로 해석하도록 지시합니다.

System.useCodePage를 true로 설정하는 경우 운영 체제의 기존 코드 페이지에 외부 텍스트 파일에 사용된 문자가 포함되어 있어야 텍스트를 표시할 수 있습니다. 예를 들어 중국어 문자가 포함된 외부 텍스트 파일을 로드하는 경우 영문 Windows 코드 페이지에는 중국어 문자가 없으므로 이 코드 페이지를 사용하는 시스템에는 중국어 문자가 표시되지 않습니다.

모든 플랫폼에서 사용자가 응용 프로그램에 사용된 외부 텍스트 파일을 볼 수 있도록 하려면 모든 외부 텍스트 파일을 유니코드로 인코딩하고 System.useCodePage를 기본값인 false로 설정해야 합니다. 이렇게 하면 런타임이 텍스트를 유니코드로 해석합니다.

클립보드에 텍스트 저장

Flash Player 9 이상, Adobe AIR 1.0 이상

System 클래스에는 Flash 런타임에서 지정된 문자열로 사용자 클립보드의 내용을 설정할 수 있는 setClipboard()라는 메서드가 포함되어 있습니다. Security.getClipboard() 메서드를 사용하면 악의적인 사이트에서 사용자의 클립보드에 마지막으로 복사된 데이터에 액세스할 수 있으므로 보안상의 이유로 이 메서드는 포함되지 않습니다.

다음 코드는 보안 오류가 발생하는 경우 오류 메시지가 사용자의 클립보드에 어떤 방식으로 복사될 수 있는지 보여 줍니다. 오류 메시지는 사용자가 응용 프로그램에 잠재적인 버그를 보고하려는 경우에 유용할 수 있습니다.

```
private function securityErrorHandler(event:SecurityErrorEvent):void
{
    var errorString:String = "[" + event.type + "] " + event.text;
    trace(errorString);
    System.setClipboard(errorString);
}
```

Flash Player 10 및 AIR 1.0
Clipboard 클래스를 사용하여 사용자 이벤트에 대한 응답으로 클립보드 데이터를 읽고 쓸 수 있습니다. AIR에서는 사용자 이벤트가 없어도 응용 프로그램 샌드박스에서 실행 중인 코드가 클립보드에 액세스할 수 있습니다.

Capabilities 클래스 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

Capabilities 클래스를 사용하면 개발자가 응용 프로그램을 실행 중인 환경을 확인할 수 있습니다. Capabilities 클래스의 다양한 속성을 사용하면 현재 설치된 Flash 런타임의 버전뿐 아니라 사용자 시스템의 해상도, 사용자 시스템에서 액세스 가능성 소프트웨어를 지원하는지 여부 및 사용자 운영 체제의 언어를 확인할 수 있습니다.

Capabilities 클래스에서 속성을 확인하면 특정한 사용자 환경에 가장 적합하게 응용 프로그램을 사용자 정의할 수 있습니다. 예를 들어 Capabilities.screenResolutionX 및 Capabilities.screenResolutionY 속성을 확인하여 사용자 시스템에서 사용 중인 화면 해상도를 확인하고 가장 적합할 것 같은 비디오 크기를 결정할 수 있습니다. 또는 외부 mp3 파일을 로드하기 전에 Capabilities.hasMP3 속성을 확인하여 사용자 시스템에서 mp3 재생을 지원하는지 여부를 확인할 수 있습니다.

다음 코드는 클라이언트에서 사용 중인 Flash 런타임 버전을 파싱하는 데 일반 표현식을 사용합니다.

```
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.");
}
```

데이터베이스에 정보를 저장할 수 있도록 사용자의 시스템 기능을 서버측 스크립트로 전송하려면 다음 ActionScript 코드를 사용할 수 있습니다.

```
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);
```

기능 예제: 시스템 기능 검색

Flash Player 9 이상

CapabilitiesExplorer 예제에서는 flash.system.Capabilities 클래스를 사용하여 사용자의 Flash 런타임 버전에서 지원하는 기능을 확인할 수 있는 방법을 보여 줍니다. 이 예제는 다음과 같은 기술에 대해 설명합니다.

- Capabilities 클래스를 사용하여 사용자의 Flash 런타임 버전에서 지원하는 기능 검색

- ExternalInterface 클래스를 사용하여 사용자의 브라우저에서 지원하는 브라우저 설정 검색

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. CapabilitiesExplorer 응용 프로그램 파일은 Samples/CapabilitiesExplorer 폴더에 있습니다. 이 응용 프로그램은 다음 파일로 구성되어 있습니다.

파일	설명
CapabilitiesExplorer.fla 또는 CapabilitiesExplorer.mxml	Flash(FLA) 또는 Flex(MXML) 형식의 기본 응용 프로그램 파일입니다.
com/example/programmingas3/capabilities/CapabilitiesGrabber.as	시스템 Capabilities를 배열에 추가하고, 항목을 정렬하며, ExternalInterface 클래스를 사용하여 브라우저 기능을 검색하는 등 응용 프로그램의 기본 기능을 제공하는 클래스입니다.
capabilities.html	외부 API와 통신할 수 있도록 필수 JavaScript를 포함하는 HTML 컨테이너입니다.

CapabilitiesExplorer 개요

Flash Player 9 이상

CapabilitiesExplorer.mxml 파일은 CapabilitiesExplorer 응용 프로그램의 사용자 인터페이스를 설정하는 작업을 담당합니다. 사용자 Flash 런타임 버전의 기능은 스테이지의 DataGrid 구성 요소 인스턴스 내에 표시됩니다. HTML 컨테이너에서 응용 프로그램을 실행하는 경우와 외부 API를 사용할 수 있는 경우에는 브라우저 기능도 표시됩니다.

기본 응용 프로그램 파일의 creationComplete 이벤트가 전달되면 initApp() 메서드가 호출됩니다. initApp() 메서드는 com.example.programmingas3.capabilities.CapabilitiesGrabber 클래스에서 getCapabilities() 메서드를 호출합니다. initApp() 메서드에 대한 코드는 다음과 같습니다.

```
private function initApp():void
{
    var dp:Array = CapabilitiesGrabber.getCapabilities();
    capabilitiesGrid.dataProvider = dp;
}
```

CapabilitiesGrabber.getCapabilities() 메서드는 Flash 런타임 및 브라우저 기능의 정렬된 배열을 반환합니다. 그런 다음 이 배열이 스테이지에 있는 capabilitiesGrid DataGrid 구성 요소 인스턴스의 dataProvider 속성으로 설정됩니다.

CapabilitiesGrabber 클래스 개요

Flash Player 9 이상

CapabilitiesGrabber 클래스의 정적 getCapabilities() 메서드는 flash.system.Capabilities 클래스에서 배열(capDP)에 각 속성을 추가합니다. 그런 다음 CapabilitiesGrabber 클래스에서 정적 getBrowserObjects() 메서드를 호출합니다. getBrowserObjects() 메서드는 외부 API를 사용하여 브라우저의 기능이 포함된 브라우저의 navigator 객체를 반복합니다. getCapabilities() 메서드는 다음과 같습니다.

```
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;
}
```

getBrowserObjects() 메서드는 브라우저의 navigator 객체에서 각 속성의 배열을 반환합니다. 이 배열에 항목 하나 이상의 길이가 포함된 경우 브라우저 기능의 배열(navArr)이 Flash Player 기능(capDP)에 추가되고 전체 배열이 알파벳 순서로 정렬됩니다. 마지막으로 정렬된 배열이 기본 응용 프로그램 파일에 반환된 다음 데이터 격자를 채웁니다. getBrowserObjects() 메서드에 대한 코드는 다음과 같습니다.

```
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;
}
```

현재 사용자 환경에서 외부 API를 사용할 수 있는 경우 Flash 런타임에서 JavaScript JS_getBrowserObjects() 메서드를 호출하여 브라우저의 navigator 객체를 반복하고 URL 인코딩된 값의 문자열을 ActionScript로 반환합니다. 그러면 이 문자열이 URLVariables 객체(itemVars)로 변환되고 호출 스크립트로 반환되는 itemArr 배열에 추가됩니다.

JavaScript로 통신

Flash Player 9 이상

CapabilitiesExplorer 응용 프로그램을 구축하는 마지막 단계는 필요한 JavaScript를 작성하여 브라우저의 navigator 객체에서 각 항목을 반복하고 이름-값 쌍을 임시 배열에 추가하는 것입니다. container.html 파일의 JavaScript JS_getBrowserObjects() 메서드에 대한 코드는 다음과 같습니다.

```
<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>
```

이 코드는 navigator 객체의 이름-값 쌍을 모두 포함할 임시 배열을 만드는 것으로 시작합니다. 그런 다음 for..in 루프를 사용하여 navigator 객체를 반복하고 현재 값의 데이터 유형을 평가하여 원치 않는 값을 제외합니다. 이 응용 프로그램에서는 문자열 또는 부울 값만 사용하고 다른 데이터 유형(함수 또는 배열)은 무시합니다. navigator 객체의 각 문자열 또는 부울 값은 tempArr 배열에 추가됩니다. 그 다음에는 for..in 루프를 사용하여 브라우저의 screen 객체가 반복되고 각 숫자 값이 tempArr 배열에 추가됩니다. 마지막으로 임시 배열은 Array.join() 메서드를 사용하여 문자열로 변환됩니다. 배열은 앰퍼샌드(&)를 구분 기호로 사용합니다. 그러므로 ActionScript에서 URLVariables 클래스를 사용하여 쉽게 데이터를 파싱할 수 있습니다.

50장: AIR 응용 프로그램 호출 및 종료

Adobe AIR 1.0 이상

이 단원에서는 설치되어 있는 Adobe® AIR® 응용 프로그램을 호출하는 방법과, 실행 중인 응용 프로그램을 닫는 데 사용되는 옵션 및 고려해야 할 사항에 대해 살펴봅니다.

참고: NativeApplication, InvokeEvent, BrowserInvokeEvent 객체는 AIR 응용 프로그램 샌드박스에서 실행되는 SWF 내용에만 사용할 수 있습니다. Flash Player 런타임에서 실행되거나, 브라우저 또는 독립 실행형 플레이어(프로젝터)에서 실행되거나, AIR 응용 프로그램 샌드박스 외부에서 실행되는 SWF 내용은 이러한 클래스에 액세스할 수 없습니다.

AIR 응용 프로그램의 호출 및 종료에 대한 간략한 설명과 코드 예제를 보려면 Adobe Developer Connection의 다음 퀵 스타트 문서를 참조하십시오.

- 시작 옵션

기타 도움말 항목

flash.desktop.NativeApplication

flash.events.InvokeEvent

flash.events.BrowserInvokeEvent

응용 프로그램 호출

Adobe AIR 1.0 이상

사용자 또는 운영 체제에서 다음 작업을 수행할 때 AIR 응용 프로그램이 호출됩니다.

- 데스크톱 셸에서 응용 프로그램을 시작합니다.
- 응용 프로그램을 명령줄 셸에서 명령으로 사용합니다.
- 응용 프로그램이 기본적으로 열리는 응용 프로그램인 파일 유형을 엽니다.
- (Mac OS X) 응용 프로그램이 현재 실행 중인지 여부에 상관없이 고정 작업 표시줄에서 응용 프로그램 아이콘을 클릭합니다.
- 이미 설치된 응용 프로그램에 대한 AIR 파일을 두 번 클릭한 후 또는 새 설치 프로세스를 끝낸 후 설치 프로그램에서 응용 프로그램을 시작하도록 선택합니다.
- 응용 프로그램 설명자 파일에 <customUpdateUI>true</customUpdateUI> 선언을 포함하여 설치된 버전에서 응용 프로그램 업데이트를 자체적으로 처리하고 있음을 신호로 표시한 경우 AIR 응용 프로그램 업데이트를 시작합니다.
- (iOS) APNs(Apple Push Notification service)로부터 알림을 수신합니다.
- URL을 통해 응용 프로그램을 호출합니다.
- AIR 응용 프로그램에 대한 식별 정보를 지정하는 com.adobe.air.AIR launchApplication() 메서드를 호출하는 Flash 배지 또는 응용 프로그램을 호스팅하는 웹 페이지를 방문합니다. 응용 프로그램 설명자에 브라우저 호출 성공을 위한 <allowBrowserInvocation>true</allowBrowserInvocation> 선언이 있어야 합니다.

AIR 응용 프로그램을 호출할 때마다 AIR이 단일 NativeApplication 객체를 통해 invoke 유형의 InvokeEvent 객체를 전달합니다. 응용 프로그램 시간에서 자체적으로 초기화하고 이벤트 리스너를 등록하도록 하기 위해 invoke 이벤트는 버리지 않고 대기열에 있습니다. 리스너를 등록하면 대기열에 있는 모든 이벤트가 전송됩니다.

참고: 브라우저 호출 기능을 사용하여 응용 프로그램을 호출할 때 응용 프로그램이 이미 실행 중이 아닌 경우에도 NativeApplication 객체는 invoke 이벤트를 전달하기만 합니다.

invoke 이벤트를 받으려면 NativeApplication 객체(NativeApplication.nativeApplication)의 addEventListener() 메서드를 호출합니다. 이벤트 리스너가 invoke 이벤트를 등록하면 등록 이전에 발생한 모든 invoke 이벤트도 수신됩니다. addEventListener()에 대한 호출을 반환한 후 대기열에 있는 invoke 이벤트를 짧은 간격으로 한 번에 하나씩 전달합니다. 이 프로세스를 진행하는 동안 새 invoke 이벤트가 발생하는 경우 대기열에 있는 하나 이상의 이벤트 이전에 해당 이벤트를 전달할 수 있습니다. 이러한 이벤트 대기열 배치 기능을 통해 초기화 코드를 실행하기 전에 발생한 모든 invoke 이벤트를 처리할 수 있습니다. 응용 프로그램 초기화 이후 이벤트 리스너를 실행 후반부에 추가하는 경우, 응용 프로그램이 시작된 후 발생한 모든 invoke 이벤트가 계속 수신됩니다.

하나의 AIR 응용 프로그램 인스턴스만 시작됩니다. 이미 실행 중인 응용 프로그램을 다시 호출하면 AIR이 실행 중인 인스턴스에 새 invoke 이벤트를 전달합니다. AIR 응용 프로그램은 invoke 이벤트에 응답하고 새 문서 윈도우 열기와 같은 적합한 액션을 수행합니다.

InvokeEvent 객체에는 응용 프로그램에 전달된 인수는 물론 응용 프로그램이 호출된 디렉토리도 포함됩니다. 파일 유형 연결 때문에 응용 프로그램이 호출된 경우 파일의 전체 경로가 명령줄 인수에 포함됩니다. 마찬가지로 응용 프로그램 업데이트 때문에 응용 프로그램이 호출된 경우 업데이트 AIR 파일의 전체 경로가 제공됩니다.

한 작업에서 여러 파일을 여는 경우 Mac OS X에서는 단일 InvokeEvent 객체가 전달되며 각 파일이 arguments 배열에 포함됩니다. Windows 및 Linux의 경우 각 파일별로 InvokeEvent 객체가 전달됩니다.

응용 프로그램은 invoke 이벤트의 NativeApplication 객체를 통해 다음과 같이 리스너를 등록하고

```
NativeApplication.nativeApplication.addEventListener(InvokeEvent.INVOKE, onInvokeEvent);
```

다음과 같이 이벤트 리스너를 정의하여 해당 이벤트를 처리할 수 있습니다.

```
 var arguments:Array;
var currentDir:File;
public function onInvokeEvent(invocation:InvokeEvent):void {
    arguments = invocation.arguments;
    currentDir = invocation.currentDirectory;
}
```

명령줄 인수 캡처

Adobe AIR 1.0 이상

NativeApplication 객체에 의해 전달된 InvokeEvent 객체에서 AIR 응용 프로그램 호출과 관련된 명령줄 인수가 전달됩니다. InvokeEvent arguments 속성에는 AIR 응용 프로그램이 호출될 때 운영 체제에 의해 전달된 인수 배열이 포함됩니다. 인수에 관련 파일 경로가 포함된 경우 일반적으로 currentDirectory 속성을 사용하여 경로를 확인할 수 있습니다.

AIR 프로그램에 전달된 인수는 큰따옴표로 묶지 않는 한 공백으로 구분된 문자열로 처리됩니다.

인수	배열
tick tock	{tick,tock}
tick "tick tock"	{tick,tick tock}
"tick" “tock”	{tick,tock}
\"tick\" \"tock\"	{"tick","tock"}

InvokeEvent 객체의 currentDirectory 속성에는 응용 프로그램이 시작된 디렉토리를 나타내는 File 객체가 포함됩니다.

응용 프로그램에서 등록된 파일 유형을 열어 응용 프로그램이 호출된 경우 파일의 기본 경로가 명령줄 인수에 문자열로 포함됩니다. 응용 프로그램에서는 파일에서 의도된 작업을 열거나 수행합니다. 마찬가지로 응용 프로그램이 표준 AIR 업데이트 사용자 인터페이스를 사용하지 않고 자체적으로 업데이트하도록 프로그래밍된 경우 사용자가 일치하는 응용 프로그램 ID의 응용 프로그램을 포함하는 AIR 파일을 두 번 클릭할 때 AIR 파일에 대한 기본 경로가 포함됩니다.

currentDirectory File 객체의 resolve() 메서드를 사용하여 파일에 액세스할 수 있습니다.

```
if((invokeEvent.currentDirectory != null)&&(invokeEvent.arguments.length > 0)){
    dir = invokeEvent.currentDirectory;
    fileToOpen = dir.resolvePath(invokeEvent.arguments[0]);
}
```

또한 인수가 사실상 파일 경로임을 확인해야 합니다.

예제: 호출 이벤트 로그

Adobe AIR 1.0 이상

다음 예제에서는 리스너를 등록하고 invoke 이벤트를 처리하는 방법을 보여 줍니다. 예제에서는 수신한 모든 호출 이벤트를 기록하고 현재 디렉토리 및 명령줄 인수를 표시합니다.

ActionScript 예제

```
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 예제

```
<?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>
```

사용자 로그인 시 AIR 응용 프로그램 호출

Adobe AIR 1.0 이상

NativeApplication startAtLogin 속성을 true로 설정하여 현재 사용자가 로그인할 때 AIR 응용 프로그램이 자동으로 시작하도록 설정할 수 있습니다. 이렇게 설정하면 사용자가 로그인할 때마다 응용 프로그램이 자동으로 시작합니다. 설정을 false로 변경하거나, 사용자가 운영 체제를 통해 설정을 수동으로 변경하거나, 응용 프로그램을 제거하지 않는 한 응용 프로그램은 사용자 로그인 시 계속 시작합니다. 로그인 시 시작은 런타임 설정입니다. 설정은 현재 사용자에게만 적용됩니다. 또한 성공적으로 startAtLogin 속성을 true로 설정하려면 응용 프로그램을 설치해야 합니다. 응용 프로그램이 설치되어 있지 않은 경우(예: 응용 프로그램이 ADL로 시작되는 경우) 속성이 설정되면 오류가 발생합니다.

참고: 컴퓨터 시스템을 시작할 때는 응용 프로그램이 시작되지 않습니다. 사용자가 로그인할 때 응용 프로그램이 시작됩니다.

응용 프로그램이 자동으로 시작되었는지 아니면 사용자 작업으로 시작되었는지 확인하려면 InvokeEvent 객체의 reason 속성을 검토하면 됩니다. 속성이 InvokeEventReason.LOGIN과 같으면 응용 프로그램이 자동으로 시작된 것입니다. 다른 호출 경로의 경우 reason 속성을 다음과 같이 설정합니다.

- InvokeEventReason.NOTIFICATION(iOS에만 해당) - 응용 프로그램이 APNs를 통해 호출된 것입니다. APNs에 대한 자세한 내용은 푸시 알림 사용을 참조하십시오.
- InvokeEventReason.OPEN_URL - 응용 프로그램이 다른 응용 프로그램 또는 시스템에 의해 호출된 것입니다.
- InvokeEventReason.Standard - 다른 모든 경우입니다.

reason 속성에 액세스하려면 응용 프로그램이 AIR 1.5.1 이상 버전이어야 합니다(응용 프로그램 설명자 파일에서 올바른 네임스페이스 값 설정).

다음의 간단한 응용 프로그램에서는 InvokeEvent reason 속성을 사용하여 invoke 이벤트 발생 시 동작 방식을 결정합니다. reason 속성이 "login"이면 응용 프로그램이 백그라운드에 유지되고, 그렇지 않으면 기본 응용 프로그램이 볼 수 있게 나타납니다. 이 패턴을 사용하는 응용 프로그램은 일반적으로 로그인 시 시작되므로 백그라운드에서 이루어지는 작업을 처리하거나 이벤트 모니터링을 수행할 수 있으며 사용자가 실행한 invoke 이벤트에 응답하여 윈도우를 엽니다.

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

참고: 비헤이비어의 차이점을 확인하려면 응용 프로그램을 패키지하여 설치하십시오. startAtLogin 속성은 설치되어 있는 응용 프로그램에 대해서만 설정할 수 있습니다.

브라우저에서 AIR 응용 프로그램 호출

Adobe AIR 1.0 이상

브라우저 호출 기능을 사용하면 브라우저에서 시작할 설치된 AIR 응용 프로그램을 웹 사이트에서 시작할 수 있습니다. 응용 프로그램 설명자 파일에서 allowBrowserInvocation을 true로 설정한 경우에만 브라우저 호출이 허용됩니다.

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

브라우저를 통해 응용 프로그램이 호출되면 응용 프로그램의 NativeApplication 객체가 BrowserInvokeEvent 객체를 전달합니다.

BrowserInvokeEvent 이벤트를 받으려면 AIR 응용 프로그램에서 NativeApplication 객체 (NativeApplication.nativeApplication)의 addEventListener() 메서드를 호출합니다. 이벤트 리스너가 BrowserInvokeEvent 이벤트를 등록하면 등록 이전에 발생한 모든 BrowserInvokeEvent 이벤트도 수신됩니다. 이러한 이벤트는 addEventListener() 호출이 반환된 이후에 전달되지만 등록 이후에 수신될 수 있는 다른 BrowserInvokeEvent 이벤트 이전이 아닐 수도 있습니다. 따라서

응용 프로그램이 브라우저에서 처음 호출되는 경우와 같이 초기화 코드가 실행되기 전에 발생한 BrowserInvokeEvent 이벤트를 처리할 수 있습니다. 응용 프로그램 초기화 이후 이벤트 리스너를 실행 후반부에 추가하는 경우, 응용 프로그램이 시작된 후 발생한 모든 BrowserInvokeEvent 이벤트가 계속 수신됩니다.

BrowserInvokeEvent 객체에는 다음과 같은 속성이 있습니다.

속성	설명
arguments	응용 프로그램에 전달할 인수(문자열)의 배열입니다.
isHTTPS	브라우저의 내용에서 https URL 스킴을 사용하는지(true), 사용하지 않는지(false) 여부를 나타냅니다.
isUserEvent	브라우저 호출이 사용자 이벤트(예: 마우스 클릭)의 결과인지 여부를 나타냅니다. AIR 1.0에서 이 값은 항상 true로 설정됩니다. AIR에서는 브라우저 호출 기능에 사용자 이벤트가 필요합니다.
sandboxType	브라우저의 내용에 대한 샌드박스 유형입니다. 유효한 값은 Security.sandboxType 속성에 사용할 수 있는 값과 동일하게 정의되며 다음 중 하나가 될 수 있습니다. • Security.APPLICATION - 내용이 응용 프로그램 보안 샌드박스에 있습니다. • Security.LOCAL_TRUSTED - 내용이 local-with-filesystem 보안 샌드박스에 있습니다. • Security.LOCAL_WITH_FILE - 내용이 local-with-filesystem 보안 샌드박스에 있습니다. • Security.LOCAL_WITH_NETWORK - 내용이 local-with-networking 보안 샌드박스에 있습니다. • Security.REMOTE - 내용이 원격(네트워크) 도메인에 있습니다.
securityDomain	브라우저의 내용에 대한 보안 도메인입니다(예: "www.adobe.com" 또는 "www.example.org"). 이 속성은 원격 보안 샌드박스의 내용(네트워크 도메인의 내용)에 대해서만 설정되며 로컬 또는 응용 프로그램 보안 샌드박스의 내용에 대해서는 설정되지 않습니다.

브라우저 호출 기능을 사용하는 경우 보안 영향을 고려해야 합니다. 웹 사이트에서 AIR 응용 프로그램을 시작하는 경우 BrowserInvokeEvent 객체의 arguments 속성을 통해 데이터를 보낼 수 있습니다. API를 로딩하는 파일 또는 코드와 같이 민감한 작업에 이러한 데이터를 사용할 때는 주의해야 합니다. 위험 수준은 응용 프로그램이 데이터를 사용하여 수행하는 작업에 따라 다릅니다. 특정 웹 사이트에서만 응용 프로그램을 호출하려는 경우 응용 프로그램은 BrowserInvokeEvent 객체의 securityDomain 속성을 확인해야 합니다. 또한 응용 프로그램을 호출하는 웹 사이트에서 HTTPs를 사용하도록 지정할 수 있습니다. 이는 BrowserInvokeEvent 객체의 isHTTPS 속성을 사용하여 확인할 수 있습니다.

응용 프로그램은 전달된 데이터의 유효성을 검사해야 합니다. 예를 들어 응용 프로그램에 특정 도메인에 대한 URL을 전달하려는 경우 응용 프로그램은 URL이 실제로 해당 도메인을 가리키는지 확인해야 합니다. 따라서 공격자가 응용 프로그램이 중요한 데이터를 보내도록 위장할 수 없습니다.

로컬 리소스를 가리킬 수 있는 BrowserInvokeEvent 인수를 사용해야 하는 응용 프로그램은 없습니다. 예를 들어 응용 프로그램은 브라우저에서 전달된 경로를 기반으로 하여 File 객체를 만들지 않아야 합니다. 브라우저에서 원격 경로를 전달해야 하는 경우 응용 프로그램은 경로에서 원격 프로토콜 대신 file:// 프로토콜을 사용하지 않는지 확인해야 합니다.

응용 프로그램 종료

Adobe AIR 1.0 이상

응용 프로그램을 종료하는 가장 빠른 방법은 NativeApplication exit() 메서드를 호출하는 것으로, 응용 프로그램에 저장할 데이터가 없거나 정리할 외부 리소스가 없을 때 효과적으로 사용할 수 있는 방법입니다. exit()를 호출하면 모든 윈도우가 닫힌 다음 응용 프로그램이 종료됩니다. 그러나 중요한 데이터를 저장하기 위해 응용 프로그램의 윈도우 또는 다른 구성 요소에서 종료 프로세스를 중단하려면 exit()를 호출하기 전에 적합한 경고 이벤트를 전달합니다.

응용 프로그램을 적절하게 종료할 때의 다른 고려 사항은 종료 프로세스를 시작하는 방법에 상관없이 단일 실행 경로를 제공하는 것입니다. 사용자 또는 운영 체제는 다음과 같은 방식으로 응용 프로그램 종료를 트리거할 수 있습니다.

- NativeApplication.nativeApplication.autoExit가 true인 경우 마지막 응용 프로그램 윈도우를 닫습니다.
- 사용자가 기본 메뉴에서 응용 프로그램 종료 명령을 선택하는 경우와 같이 운영 체제에서 응용 프로그램 종료 명령을 선택합니다. 이는 Mac OS에서만 가능합니다. Windows 및 Linux에서는 시스템 크롬을 통해 응용 프로그램 종료 명령을 제공하지 않습니다.
- 컴퓨터를 종료합니다.

종료 명령이 이러한 경로 중 하나로 운영 체제를 통해 조정되는 경우 NativeApplication은 exiting 이벤트를 전달합니다. exiting 이벤트를 취소할 리스너가 없는 경우 열려 있는 모든 윈도우가 닫힙니다. 각 윈도우는 closing 이벤트를 전달한 다음 close 이벤트를 전달합니다. closing 이벤트를 취소할 윈도우가 없는 경우 종료 프로세스가 중단됩니다.

윈도우 종료 순서가 응용 프로그램에서 중요한 문제가 되는 경우 NativeApplication에서 exiting 이벤트를 수신하고 자체적으로 적합한 순서로 윈도우를 닫습니다. 예를 들어 도구 팔레트가 포함된 문서 윈도우를 사용하는 경우 이 방법이 필요할 수 있습니다. 시스템에서 팔레트를 닫았는데 사용자가 일부 데이터를 저장하기 위해 종료 명령을 취소하려는 경우 이러한 작업은 불편하거나 문제를 발생시킬 수 있습니다. Windows에서 exiting 이벤트를 받을 유일한 시간은 마지막 윈도우를 닫은 후입니다 (NativeApplication 객체의 autoExit 속성이 true로 설정된 경우).

모든 플랫폼에서의 일관된 비헤이비어를 제공하려면 종료 시퀀스를 운영 체제 크롬, 메뉴 명령 또는 응용 프로그램 논리를 통해 시작했는지에 상관없이 응용 프로그램을 종료하기 위한 다음과 같은 권장 방식을 관찰합니다.

1. 항상 응용 프로그램 코드에서 exit()를 호출하기 전에 NativeApplication 객체를 통해 exiting 이벤트를 전달하고 응용 프로그램의 다른 구성 요소에서 이벤트를 취소하지 않는지 확인합니다.

```
public function applicationExit():void {
    var exitingEvent:Event = new Event(Event.EXITING, false, true);
    NativeApplication.nativeApplication.dispatchEvent(exitingEvent);
    if (!exitingEvent.isDefaultPrevented()) {
        NativeApplication.nativeApplication.exit();
    }
}
```

2. NativeApplication.nativeApplication 객체에서 응용 프로그램 exiting 이벤트를 수신하고 핸들러에서 먼저 closing 이벤트를 전달하는 모든 윈도우를 닫습니다. 모든 윈도우가 닫힌 후에는 응용 프로그램 데이터 저장 또는 임시 파일 삭제와 같은 필요한 정리 작업을 수행합니다. 정리 작업을 수행하는 동안 동기적 메서드만 사용하여 응용 프로그램이 끝나기 전에 이러한 메서드가 종료되는지 확인합니다.

   윈도우가 닫히는 순서가 그리 중요하지 않는 경우 NativeApplication.nativeApplication.openedWindows 배열을 반복하고 각 윈도우를 차례로 닫을 수 있습니다. 순서가 중요한 경우에는 윈도우를 올바른 시퀀스로 닫는 방법을 제공합니다.

```
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 Windows는 항상 자체 closing 이벤트를 수신하여 자체 정리 작업을 처리합니다.

4 이전에 호출된 핸들러에서 후속 핸들러가 exiting 이벤트를 처리할지 여부를 알 수 없기 때문에(그리고 실행 순서에 의존하는 것은 그리 권장할 만한 것이 아님) 응용 프로그램에서 하나의 exiting 리스너만 사용합니다.

51장: AIR 런타임 및 운영 체제 정보 작업

Adobe AIR 1.0 이상

이 단원에서는 AIR 응용 프로그램에서 운영 체제 파일 연결을 관리하고, 사용자 작업을 감지하고, Adobe® AIR® 런타임에 대한 정보를 얻는 방법에 대해 설명합니다.

기타 도움말 항목
flash.desktop.NativeApplication

파일 연결 관리

Adobe AIR 1.0 이상

응용 프로그램과 파일 유형 간의 연결은 응용 프로그램 설명자에 선언되어야 합니다. 설치 프로세스 동안 AIR 응용 프로그램 설치 프로그램은 다른 응용 프로그램이 이미 기본값이 아닌 경우 AIR 응용 프로그램을 선언된 각각의 파일 유형에 대해 기본적으로 열리는 응용 프로그램으로 연결합니다. AIR 응용 프로그램 설치 프로세스는 기존 파일 유형 연결을 재정의하지 않습니다. 다른 응용 프로그램의 연결을 가져오려면 런타임에 NativeApplication.setAsDefaultApplication() 메서드를 호출합니다.

응용 프로그램이 시작될 때 예상 파일 연결이 있는지 확인하는 것이 좋습니다. 이는 AIR 응용 프로그램 설치 프로그램이 기존 파일 연결을 재정의하지 않을 뿐만 아니라 사용자 시스템의 파일 연결을 언제든지 변경할 수 있기 때문입니다. 다른 응용 프로그램에 현재 파일 연결이 있는 경우 기존 연결을 가져오기 전에 사용자에게 물어보는 것이 좋습니다.

NativeApplication 클래스의 다음 메서드를 통해 응용 프로그램이 파일 연결을 관리할 수 있습니다. 각각의 메서드는 파일 유형 확장명을 매개 변수로 사용합니다.

메서드	설명
isSetAsDefaultApplication()	AIR 응용 프로그램이 현재 지정된 파일 유형과 연결되어 있으면 true를 반환합니다.
setAsDefaultApplication()	AIR 응용 프로그램과 파일 유형의 열기 액션 간 연결을 만듭니다.
removeAsDefaultApplication()	AIR 응용 프로그램과 파일 유형 간 연결을 제거합니다.
getDefaultApplication()	파일 유형에 현재 연결되어 있는 응용 프로그램의 경로를 보고합니다.

AIR은 응용 프로그램 설명자에 원래 선언된 파일 유형에 대한 연결만 관리할 수 있습니다. 사용자가 파일 유형과 응용 프로그램 간 연결을 수동으로 만든 경우에도 선언되지 않은 파일 유형의 연결에 대한 정보를 얻을 수 없습니다. 응용 프로그램 설명자에 선언되지 않은 파일 유형에 대한 파일 확장명의 파일 연결 관리 메서드를 호출하면 응용 프로그램에서 런타임 예외가 발생합니다.

런타임 버전 및 패치 수준 가져오기

Adobe AIR 1.0 이상

NativeApplication 객체에는 응용 프로그램이 실행 중인 런타임 버전("1.0.5"와 같은 문자열)인 runtimeVersion 속성이 있습니다. NativeApplication 객체에는 런타임의 패치 수준(2960과 같은 숫자)인 runtimePatchLevel 속성도 있습니다. 다음 코드는 이러한 속성을 사용합니다.

```
trace(NativeApplication.nativeApplication.runtimeVersion);
trace(NativeApplication.nativeApplication.runtimePatchLevel);
```

AIR 기능 감지

Adobe AIR 1.0 이상

Adobe AIR 응용 프로그램과 함께 제공되는 파일의 경우 Security.sandboxType 속성이 Security.APPLICATION 상수에 의해 정의된 값으로 설정됩니다. 다음 코드에 나오는 대로 파일이 Adobe AIR 보안 샌드박스에 있는지 여부에 따라 AIR 관련 API를 포함하거나 포함하지 않는 내용을 로드할 수 있습니다.

```
if (Security.sandboxType == Security.APPLICATION)
{
    // Load SWF that contains AIR APIs
}
else
{
    // Load SWF that does not contain AIR APIs
}
```

AIR 응용 프로그램과 함께 설치되지 않은 모든 리소스는 웹 브라우저에서 Adobe® Flash® Player에 의해 할당되는 동일한 보안 샌드박스에 할당됩니다. 원격 리소스는 해당 소스 도메인에 따라 샌드박스에 배치되고 로컬 리소스는 local-with-networking, local-with-filesystem 또는 local-trusted sandbox에 배치됩니다.

내용이 런타임에서 실행 중이며 브라우저에서 실행하는 Flash Player에서는 실행되고 있지 않은지 확인할 수 있도록 Capabilities.playerType 정적 속성이 "Desktop"으로 설정되었는지 검사할 수 있습니다.

자세한 내용은 979페이지의 “AIR 보안”을 참조하십시오.

사용자 현재 상태 추적

Adobe AIR 1.0 이상

NativeApplication 객체는 사용자가 컴퓨터를 실제로 사용 중인 경우를 감지하는 두 개의 이벤트를 전달합니다. NativeApplication.idleThreshold 속성에 의해 결정된 간격으로 마우스 또는 키보드 작업이 감지되지 않는 경우 NativeApplication은 userIdle 이벤트를 전달합니다. 다음 키보드 또는 마우스 입력이 발생할 때 NativeApplication 객체는 userPresent 이벤트를 전달합니다. idleThreshold 간격은 초 단위로 측정되고 기본값이 300(5분)입니다. 또한 NativeApplication.nativeApplication.lastUserInput 속성에서 마지막 사용자 입력 이후 초 수를 얻을 수 있습니다.

다음 코드 줄은 유휴 임계값을 2분으로 설정하고 userIdle 이벤트와 userPresent 이벤트를 모두 수신합니다.

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

참고: 두 userPresent 이벤트 사이에는 하나의 userIdle 이벤트만 전달됩니다.

52장: AIR 기본 윈도우를 사용한 작업

Adobe AIR 1.0 이상

Adobe® AIR® 기본 윈도우 API에서 제공하는 클래스를 사용하여 데스크톱 윈도우를 만들고 관리합니다.

AIR 기본 윈도우의 기초

Adobe AIR 1.0 이상

AIR의 기본 윈도우를 사용한 작업에 대한 빠른 설명 및 코드 예제를 보려면 Adobe Developer Connection에서 다음과 같은 퀵 스타트 문서를 참조하십시오.

- 투명 윈도우 응용 프로그램 만들기(Flex)
- 윈도우와 상호 작용(Flex)
- 기본 윈도우의 모양 및 느낌 사용자 정의(Flex)
- 윈도우 시작(Flex)
- 토스트 스타일 윈도우 만들기(Flex)
- 윈도우의 표시 순서 제어(Flex)
- 사각형이 아닌 크기 조절이 가능한 윈도우 만들기(Flex)
- 윈도우와 상호 작용(Flash)
- 기본 윈도우의 모양 및 느낌 사용자 정의(Flash)
- 토스트 스타일 윈도우 만들기(Flash)
- 윈도우의 표시 순서 제어(Flash)
- 사각형이 아닌 크기 조절이 가능한 윈도우 만들기(Flash)

AIR은 Flash®, Flex™ 및 HTML 프로그래밍 기술을 사용하여 기본 운영 체제 윈도우를 만드는 데 쉽게 사용할 수 있으며 플랫폼의 영향을 받지 않는 윈도우 API를 제공합니다.

AIR을 사용하면 응용 프로그램의 모양을 개발할 때 융통성이 커집니다. Mac에서 실행될 때는 Apple 스타일과 일치하고 Windows에서 실행될 때는 Microsoft 규칙을 따르며 Linux에서는 윈도우 관리자와 조화를 이루도록 윈도우의 모양을 표준 데스크톱 응용 프로그램과 비슷하게 만들 수 있습니다. 이러한 작업은 모두 플랫폼별 코드 행을 포함하지 않고 수행할 수 있습니다. 또는 Flex 프레임워크에서 제공하며 스킨을 사용할 수 있고 확장 가능한 크롬을 사용하여 응용 프로그램이 실행되는 곳에 관계없이 고유의 스타일을 설정할 수도 있습니다. 데스크톱에 대한 알파 블렌딩과 투명도를 완전히 지원하는 벡터 및 비트맵 아트웍을 사용하여 직접 윈도우 크롬을 그릴 수도 있습니다. 사각형 윈도우가 지겨우면 원형 윈도우를 그려보십시오.

AIR의 윈도우

Adobe AIR 1.0 이상

AIR은 윈도우 작업을 위한 세 가지 고유한 API를 지원합니다.

- ActionScript 지향 NativeWindow 클래스는 최하위 수준의 윈도우 API를 제공합니다. ActionScript 및 Flash Professional 제작 응용 프로그램에서는 NativeWindows를 사용합니다. 응용 프로그램에 사용된 윈도우를 구체화하려면 NativeWindow 클래스를 확장하십시오.
- HTML 환경에서는 브라우저 기반 웹 응용 프로그램에서와 같이 JavaScript Window 클래스를 사용할 수 있습니다. JavaScript Window 메서드에 대한 호출은 기본 윈도우 객체로 이동됩니다.
- Flex 프레임워크인 mx:WindowedApplication 및 mx:Window 클래스는 NativeWindow 클래스를 위한 Flex "래퍼"를 제공합니다. WindowedApplication 구성 요소는 Flex로 AIR 응용 프로그램을 만들 때 응용 프로그램 구성 요소를 바꾸므로 Flex 응용 프로그램에서 항상 초기 윈도우로 사용되어야 합니다.

ActionScript 윈도우

NativeWindow 클래스를 사용하여 윈도우를 만드는 경우 Flash Player 스테이지를 사용하고 목록을 직접 표시합니다. 시각적 객체를 NativeWindow에 추가하려면 윈도우 스테이지의 표시 목록이나 스테이지의 다른 표시 객체 컨테이너에 객체를 추가합니다.

HTML 윈도우

HTML 윈도우를 만들 때 HTML, CSS 및 JavaScript를 사용하여 내용을 표시합니다. 시각적 객체를 HTML 윈도우에 추가하려면 해당 내용을 HTML DOM에 추가합니다. HTML 윈도우는 NativeWindow의 특수한 범주입니다. AIR 호스트는 HTML 윈도우에서 기본 NativeWindow 인스턴스에 액세스할 수 있도록 하는 nativeWindow 속성을 정의합니다. 이 속성을 사용하여 여기에서 설명하는 NativeWindow 속성, 메서드 및 이벤트에 액세스할 수 있습니다.

참고: JavaScript Window 객체에는 moveTo() 및 close()와 같이 포함하는 윈도우를 스크립팅하기 위한 메서드도 있습니다. 사용할 수 있는 메서드가 겹치는 경우 편리한 메서드를 사용하면 됩니다.

Flex 프레임워크 윈도우

Flex 프레임워크를 사용하여 윈도우를 만드는 경우 일반적으로 MXML 구성 요소를 사용하여 윈도우를 채웁니다. Flex 구성 요소를 윈도우에 추가하려면 해당 구성 요소를 윈도우 MXML 정의에 추가합니다. ActionScript를 사용하여 내용을 동적으로 추가할 수도 있습니다. mx:WindowedApplication 및 mx:Window 구성 요소는 Flex 컨테이너로 설계되었으므로 Flex 구성 요소를 직접 받아들일 수 있지만 NativeWindow 객체는 그럴 수 없습니다. 필요한 경우 NativeWindow 속성 및 메서드는 nativeWindow 속성을 사용하여 WindowedApplication 및 Window 객체를 통해 액세스할 수 있습니다.

초기 응용 프로그램 윈도우

응용 프로그램의 첫 번째 윈도우는 AIR에서 자동으로 만듭니다. AIR은 응용 프로그램 설명자 파일의 initialWindow 요소에 지정된 매개 변수를 사용하여 윈도우의 속성과 내용을 설정합니다.

루트 내용이 SWF 파일이면 AIR은 NativeWindow 인스턴스를 만든 다음 SWF 파일을 로드하여 윈도우 스테이지에 추가합니다. 루트 내용이 HTML 파일이면 AIR은 HTML 윈도우를 만들고 HTML을 로드합니다.

기본 윈도우 클래스

Adobe AIR 1.0 이상

기본 윈도우 API에는 다음과 같은 클래스가 포함되어 있습니다.

패키지	클래스
flash.display	• NativeWindow • NativeWindowInitOptions • NativeWindowDisplayState • NativeWindowResize • NativeWindowSystemChrome • NativeWindowType
flash.events	• NativeWindowBoundsEvent • NativeWindowDisplayStateEvent

기본 윈도우 이벤트 흐름

Adobe AIR 1.0 이상

기본 윈도우는 중요한 변경이 발생하려고 하거나 이미 발생했음을 관련 구성 요소에 알리기 위해 이벤트를 전달합니다. 많은 윈도우 관련 이벤트가 쌍으로 전달됩니다. 첫 번째 이벤트는 변경이 발생하려고 한다고 경고합니다. 두 번째 이벤트는 변경이 수행되었다고 알립니다. 경고 이벤트를 취소할 수 있지만 알림 이벤트는 취소할 수 없습니다. 사용자가 윈도우의 최대화 버튼을 클릭할 때 발생하는 이벤트의 흐름은 다음과 같습니다.

1. NativeWindow 객체가 displayStateChanging 이벤트를 전달합니다.
2. 등록된 리스너가 이벤트를 취소하지 않으면 윈도우가 최대화됩니다.
3. NativeWindow 객체가 displayStateChange 이벤트를 전달합니다.

   또한 NativeWindow 객체도 윈도우 크기 및 위치의 관련 변경에 대한 이벤트를 전달합니다. 윈도우는 이러한 관련 변경에 대한 경고 이벤트를 전달하지 않습니다. 관련 이벤트는 다음과 같습니다.

   a. move 이벤트는 최대화 작업 때문에 윈도우의 왼쪽 맨 위 모서리가 이동한 경우 전달됩니다.

   b. resize 이벤트는 최대화 작업 때문에 윈도우 크기가 변경된 경우 전달됩니다.

   NativeWindow 객체는 윈도우의 최소화, 복원, 닫기, 이동 및 크기 조절 시 유사한 이벤트 시퀀스를 전달합니다.

   경고 이벤트는 윈도우 크롬이나 운영 체제에서 제어하는 다른 메커니즘을 통해 변경이 시작될 때만 전달됩니다. 윈도우 메서드를 호출하여 윈도우 크기, 위치 또는 표시 상태를 변경하는 경우 윈도우는 이러한 변경을 알리기 위해서만 이벤트를 전달합니다. 윈도우 dispatchEvent() 메서드를 사용하여 경고 이벤트를 전달한 다음 변경을 계속하기 전에 경고 이벤트가 취소되었는지 확인할 수 있습니다.

   윈도우 API 클래스, 메서드, 속성 및 이벤트에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

기본 윈도우 스타일 및 비헤이비어를 제어하는 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 속성은 윈도우의 기본적인 모양과 비헤이비어를 제어합니다.

- type
- systemChrome
- transparent

- owner

윈도우를 만들 때 윈도우 생성자에 전달된 NativeWindowInitOptions 객체에서 이러한 속성을 설정합니다. AIR은 응용 프로그램 설명자에서 초기 응용 프로그램 윈도우의 속성을 읽습니다(응용 프로그램 설명자에서 설정할 수 없고 항상 normal로 설정되는 type 속성 제외). 이러한 속성은 윈도우를 만든 후 변경할 수 없습니다.

이러한 속성의 일부 설정은 상호 호환되지 않습니다. transparent가 true이거나 type이 lightweight인 경우 systemChrome을 standard로 설정할 수 없습니다.

윈도우 유형

Adobe AIR 1.0 이상

AIR 윈도우 유형의 경우 기본 운영 체제의 가시성 특성과 크롬을 결합하여 세 가지 기능의 윈도우 유형을 만듭니다. NativeWindowType 클래스에 정의된 상수를 사용하여 코드에서 유형 이름을 참조합니다. AIR은 다음과 같은 윈도우 유형을 제공합니다.

유형	설명
일반	일반적인 윈도우입니다. 일반 윈도우는 전체 크기 스타일의 크롬을 사용하고 Windows 작업 표시줄 및 Mac OS X 윈도우 메뉴에 표시됩니다.
유틸리티	도구 팔레트입니다. 유틸리티 윈도우는 보다 간단한 시스템 크롬 버전을 사용하며 Windows 작업 표시줄 및 Mac OS X 윈도우 메뉴에 표시되지 않습니다.
경량	경량 윈도우에는 크롬이 없으며 Windows 작업 표시줄이나 Mac OS X 윈도우 메뉴에 표시되지 않습니다. 또한 경량 윈도우에는 Windows의 시스템 메뉴(Alt+Space)가 없습니다. 경량 윈도우는 알림 버블링 및 잠시 사용되는 표시 영역을 여는 콤보 상자와 같은 컨트롤에 적합합니다. 경량 유형을 사용할 때는 systemChrome을 none으로 설정해야 합니다.

윈도우 크롬

Adobe AIR 1.0 이상

윈도우 크롬은 데스크톱 환경에서 윈도우를 조작하는 데 사용할 수 있는 컨트롤의 집합입니다. 크롬 요소에는 제목 표시줄, 제목 표시줄 버튼, 테두리, 크기 조절 그리퍼 등이 있습니다.

시스템 크롬

systemChrome 속성을 standard 또는 none으로 설정할 수 있습니다. 사용자의 운영 체제에서 만들어지고 스타일이 지정된 표준 컨트롤의 집합을 윈도우에 제공하려면 standard 시스템 크롬을 선택하고, 윈도우에 사용자 고유의 크롬을 제공하려면 none을 선택합니다. 코드에서 시스템 크롬 설정을 참조하려면 NativeWindowSystemChrome 클래스에 정의된 상수를 사용합니다.

시스템 크롬은 시스템에서 관리됩니다. 응용 프로그램에서는 컨트롤 자체에 직접 액세스할 수 없지만 컨트롤이 사용될 때 전달된 이벤트에 응답할 수 있습니다. 윈도우에 표준 크롬을 사용하는 경우 transparent 속성을 false로 설정해야 하고 type 속성이 normal 또는 utility여야 합니다.

Flex 크롬

Flex WindowedApplication 또는 Window 구성 요소를 사용하는 경우 윈도우에서 시스템 크롬이나 Flex 프레임워크에서 제공하는 크롬을 사용할 수 있습니다. Flex 크롬을 사용하려면 윈도우를 만드는 데 사용된 systemChrome 속성을 none으로 설정합니다. mx 구성 요소가 아닌 Flex 4 spark 구성 요소를 사용하는 경우 Flex 크롬을 사용하려면 스킨 클래스를 지정해야 합니다. 기본 제공 스킨을 사용하거나 사용자의 스킨을 제공할 수 있습니다. 다음 예에서는 기본 제공 spark WindowedApplication 스킨 클래스를 사용하여 윈도우 크롬을 제공하는 방법을 보여 줍니다.

```
<?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>
```

자세한 내용은 Flex 4 사용: AIR 윈도우 컨테이너 정보: 윈도우 크롬 제어를 참조하십시오.

사용자 정의 크롬

시스템 크롬을 사용하지 않고 윈도우를 만드는 경우 사용자와 윈도우의 상호 작용을 처리하기 위해 사용자 고유의 크롬 컨트롤을 추가해야 합니다. 또한 사각형이 아닌 투명 윈도우를 만들 수 있습니다.

mx:WindowedApplication 또는 mx:Window 구성 요소와 함께 사용자 정의 크롬을 사용하려면 showFlexChrome 스타일을 false로 설정해야 합니다. 그렇지 않으면 Flex에서 자체의 고유한 크롬을 윈도우에 추가합니다.

윈도우 투명도

Adobe AIR 1.0 이상

바탕 화면이나 다른 윈도우와 윈도우의 알파 블렌딩을 허용하려면 윈도우 transparent 속성을 true로 설정합니다. transparent 속성은 윈도우를 만들기 전에 설정해야 하며 변경할 수 없습니다.

투명 윈도우에는 기본 배경이 없습니다. 응용 프로그램에서 그린 객체가 포함되지 않은 윈도우 영역은 표시되지 않습니다. 표시된 객체의 알파가 1보다 작게 설정되면 동일한 윈도우, 다른 윈도우 및 바탕 화면에 있는 다른 표시 객체를 비롯하여 객체 아래에 있는 모든 것이 표시됩니다.

투명 윈도우는 모양이 불규칙하거나 "페이드 아웃"되거나 표시되지 않는 테두리를 사용하여 응용 프로그램을 만들려는 경우 유용합니다. 그러나 알파 블렌딩된 큰 영역을 렌더링하는 경우 속도가 느릴 수 있으므로 이 효과는 반드시 필요한 경우에만 사용해야 합니다.

중요: Linux에서는 마우스 이벤트가 완전히 투명한 픽셀을 통과하지 않습니다. 따라서 보이지 않는 방식으로 데스크톱의 다른 윈도우 또는 항목에 대한 사용자의 액세스를 차단할 수 있으므로 완전히 투명한 영역이 큰 윈도우를 만들어서는 안 됩니다. Mac OS X 및 Windows의 경우에는 마우스 이벤트가 완전히 투명한 픽셀을 통과합니다.

시스템 크롬이 있는 윈도우에는 투명도를 사용할 수 없습니다. 또한 HTML의 SWF 및 PDF 내용은 투명 윈도우에 표시되지 않을 수 있습니다. 자세한 내용은 913페이지의 “SWF 또는 PDF 내용을 HTML 페이지에 로드하는 경우의 고려 사항”을 참조하십시오.

정적 NativeWindow.supportsTransparency 속성은 윈도우 투명도를 사용할 수 있는지 여부를 보고합니다. 투명도가 지원되지 않으면 응용 프로그램이 검정 배경에 합성됩니다. 이러한 경우 응용 프로그램의 투명한 영역은 모두 불투명한 검정으로 표시됩니다. 따라서 이 속성이 false로 판단되는 경우에 대비하여 대체 동작을 제공하는 것이 좋습니다. 예를 들어 경고 대화 상자를 사용자에게 표시하거나 투명하지 않은 사각형 사용자 인터페이스를 표시할 수 있습니다.

Mac 및 Windows 운영 체제에서는 투명도가 항상 지원됩니다. Linux 운영 체제에서 투명도를 사용하려면 합성 윈도우 관리자가 필요하지만 합성 윈도우 관리자가 활성 상태인 경우에도 사용자 표시 옵션 또는 하드웨어 구성으로 인해 투명도를 사용하지 못할 수 있습니다.

MXML 응용 프로그램 윈도우의 투명도

Adobe AIR 1.0 이상

투명 윈도우를 만드는 경우에도 MXML 윈도우의 배경은 기본적으로 불투명합니다. 투명도 효과는 윈도우의 모서리에서 확인할 수 있습니다. 윈도우의 배경을 투명하게 표시하려면 응용 프로그램 MXML 파일에 포함된 <mx:Style> 요소나 스타일 시트의 배경색과 알파 값을 설정합니다. 예를 들어 다음과 같은 스타일 선언은 배경에 약간 투명한 녹색 음영을 제공합니다.

```
WindowedApplication
{
    background-alpha:".8";
    background-color:"0x448234";
}
```

HTML 응용 프로그램 윈도우의 투명도

Adobe AIR 1.0 이상

포함하는 윈도우가 투명한 경우에도 HTML 윈도우나 HTMLLoader 객체에 표시되는 HTML 내용의 배경은 기본적으로 불투명합니다. HTML 내용에 대해 표시되는 기본 배경을 해제하려면 paintsDefaultBackground 속성을 false로 설정합니다. 다음 예제에서는 HTMLLoader를 만들고 기본 배경을 해제합니다.

```
var htmlView:HTMLLoader = new HTMLLoader();
htmlView.paintsDefaultBackground = false;
```

이 예제에서는 JavaScript를 사용하여 HTML 윈도우의 기본 배경을 해제합니다.

```
window.htmlLoader.paintsDefaultBackground = false;
```

HTML 문서의 요소가 배경색을 설정하는 경우 해당 요소의 배경은 투명하지 않습니다. 부분 투명도(또는 불투명도) 값 설정은 지원되지 않습니다. 그러나 투명한 PNG 형식 그래픽을 페이지나 페이지 요소의 배경으로 사용하여 유사한 시각적 효과를 얻을 수 있습니다.

윈도우 소유권

한 개의 윈도우는 다른 윈도우를 하나 이상 **소유**할 수 있습니다. 소유된 윈도우는 항상 마스터 윈도우 앞에 나타나며, 마스터 윈도우와 함께 최소화 및 복원되고 마스터 윈도우가 닫히면 함께 닫힙니다. 윈도우 소유권은 다른 윈도우로 이동하거나 제거할 수 없습니다. 윈도우는 하나의 마스터 윈도우에서만 소유될 수 있지만 임의의 개수의 다른 윈도우를 소유할 수 있습니다.

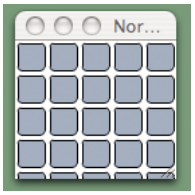
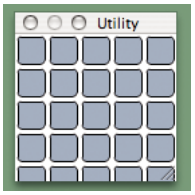
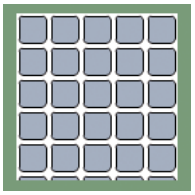
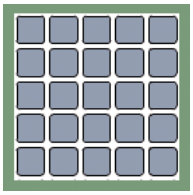
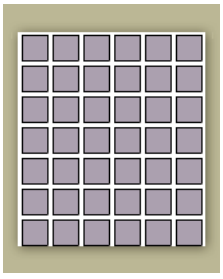
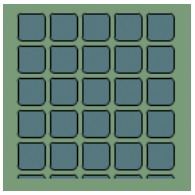
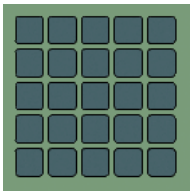
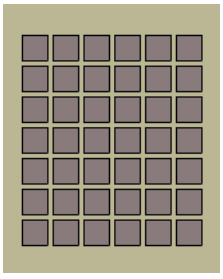
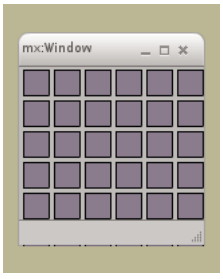
윈도우 소유권을 활용하면 도구 팔레트 및 대화 상자에 사용되는 윈도우를 더욱 쉽게 관리할 수 있습니다. 예를 들어 [저장] 대화 상자를 문서 윈도우와 연결해 표시한 경우 문서 윈도우가 대화 상자를 소유하도록 하면 대화 상자가 문서 윈도우 앞에 자동으로 유지됩니다.

- NativeWindow.owner
- Christian Cantrell: AIR 2.6의 소속 윈도우

시각적 윈도우 카탈로그

Adobe AIR 1.0 이상

다음 표에서는 Mac OS X, Windows 및 Linux 운영 체제에서 윈도우 속성 설정의 여러 가지 조합에 대한 시각적 효과를 보여 줍니다.

윈도우 설정	Mac OS X	Microsoft Windows	Linux*
Type: normal SystemChrome: standard Transparent: false			
Type: utility SystemChrome: standard Transparent: false			
Type: Any SystemChrome: none Transparent: false			
Type: Any SystemChrome: none Transparent: true			
mx:WindowedApplication 또는 mx:Window Type: Any SystemChrome: none Transparent: true			

*Compiz 윈도우 관리자가 포함된 Ubuntu

참고: Mac OS X 툴바, Mac OS X 프록시 아이콘, Windows 제목 표시줄 아이콘 및 다른 시스템 크롬과 같은 시스템 크롬 요소는 AIR에서 지원되지 않습니다.

윈도우 만들기

Adobe AIR 1.0 이상

AIR에서는 응용 프로그램의 첫 번째 윈도우를 자동으로 만들지만 사용자가 필요한 윈도우를 추가로 만들 수 있습니다. 기본 윈도우를 만들려면 NativeWindow 생성자 메서드를 사용합니다.

HTML 윈도우를 만들려면 HTMLLoader createRootWindow() 메서드를 사용하거나 HTML 문서에서 JavaScript window.open() 메서드를 호출합니다. 만들어진 윈도우는 표시 목록에 HTMLLoader 객체가 포함된 NativeWindow 객체입니다. HTMLLoader 객체는 윈도우의 HTML 및 JavaScript 내용을 해석하고 표시합니다. window.nativeWindow 속성을 사용하여 JavaScript의 기본 NativeWindow 객체의 속성에 액세스할 수 있습니다. 이 속성은 AIR 응용 프로그램 샌드박스에서 실행되는 코드에서만 액세스할 수 있습니다.

응용 프로그램 초기 윈도우를 비롯한 윈도우를 초기화할 때는 내용을 보이지 않는 상태로 만들고 내용을 로드하거나 그래픽 업데이트를 실행한 후 윈도우를 보이게 만드는 것이 좋습니다. 이 순서를 사용하면 사용자에게 거슬릴 수 있는 시각적 변경 사항이 표시되지 않습니다. 응용 프로그램의 초기 윈도우가 보이지 않는 상태로 만들려면 응용 프로그램 설명자에서 <visible>false</visible> 태그를 지정하거나 false가 기본값이므로 이 태그를 아예 빼면 됩니다. 새 NativeWindows는 기본적으로 보이지 않습니다. HTMLLoader createRootWindow() 메서드로 HTML 윈도우를 만드는 경우 visible 인수를 false로 설정할 수 있습니다. 윈도우를 보이게 만들려면 NativeWindow activate() 메서드를 호출하거나 visible 속성을 true로 설정합니다.

윈도우 초기화 속성 지정

Adobe AIR 1.0 이상

기본 윈도우의 초기화 속성은 데스크톱 윈도우를 만든 다음에는 변경할 수 없습니다. 이러한 변경할 수 없는 속성과 기본값은 다음과 같습니다.

속성	기본값
systemChrome	standard
type	normal
transparent	false
owner	null
maximizable	true
minimizable	true
resizable	true

응용 프로그램 설명자 파일에서 AIR을 통해 만들어진 초기 윈도우의 속성을 설정합니다. AIR 응용 프로그램의 기본 윈도우는 항상 type, **normal**입니다. visible, width 및 height와 같은 추가 윈도우 속성을 지정할 수 있지만 이러한 속성은 언제든지 변경할 수 있습니다.

NativeWindowInitOptions 클래스를 사용하여 응용 프로그램에서 만든 다른 기본 윈도우와 HTML 윈도우의 속성을 설정합니다. 윈도우를 만들 때 윈도우 속성을 지정하는 NativeWindowInitOptions 객체를 NativeWindow 생성자 함수나 HTMLLoader createRootWindow() 메서드에 전달해야 합니다.

다음 코드에서는 유틸리티 윈도우에 대한 NativeWindowInitOptions 객체를 만듭니다.

```
var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.type = NativeWindowType.UTILITY
options.transparent = false;
options.resizable = false;
options.maximizable = false;
```

transparent가 true이거나 type이 lightweight인 경우 **systemChrome**을 **standard**로 설정할 수 없습니다.

참고: JavaScript window.open() 함수로 만든 윈도우에 대한 초기화 속성을 설정할 수는 없지만, HTMLHost 클래스를 직접 구현하여 이러한 윈도우를 만드는 방법을 재정의할 수 있습니다. 자세한 내용은 923페이지의 "window.open()에 대한 JavaScript 호출 처리"를 참조하십시오.

Flex mx:Window 클래스를 사용하여 윈도우를 만드는 경우 윈도우에 대한 MXML 선언이나 윈도우를 만드는 코드에서 윈도우 객체 자체에 대한 초기화 속성을 지정합니다. 기본 NativeWindow 객체는 open() 메서드를 호출할 때까지 만들어지지 않습니다. 윈도우가 열리면 이러한 초기화 속성을 변경할 수 없습니다.

초기 응용 프로그램 윈도우 만들기

Adobe AIR 1.0 이상

AIR에서는 응용 프로그램 설명자에 지정된 속성을 기반으로 초기 응용 프로그램 윈도우를 만들고 내용 요소에서 참조된 파일을 로드합니다. 내용 요소는 SWF 파일이나 HTML 파일을 참조해야 합니다.

초기 윈도우는 응용 프로그램의 기본 윈도우이거나 하나 이상의 다른 윈도우를 시작하는 역할만 수행할 수 있습니다. 초기 윈도우를 표시할 필요는 전혀 없습니다.

ActionScript를 사용하여 초기 윈도우 만들기

Adobe AIR 1.0 이상

ActionScript를 사용하여 AIR 응용 프로그램을 만들 때 응용 프로그램의 기본 클래스는 Sprite 클래스(또는 Sprite의 하위 클래스)를 확장해야 합니다. 이 클래스는 응용 프로그램의 기본 진입점 역할을 합니다.

응용 프로그램이 시작되면 AIR에서는 윈도우를 만들고 기본 클래스의 인스턴스를 만들어 윈도우 스테이지에 추가합니다. 윈도우에 액세스하려면 addedToStage 이벤트를 수신한 다음 Stage 객체의 nativeWindow 속성을 사용하여 NativeWindow 객체에 대한 참조를 가져올 수 있습니다.

다음 예제에서는 ActionScript를 사용하여 빌드된 AIR 응용 프로그램의 기본 클래스에 대한 기본적인 구조를 보여 줍니다.

```
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
        }
    }
}
```

참고: 일반적으로 기본 클래스의 생성자 함수에 있는 nativeWindow 속성에 액세스할 수 있습니다. 그러나 이러한 경우는 초기 응용 프로그램 윈도우에만 적용되는 특수한 경우입니다.

Flash Professional에서 응용 프로그램을 만드는 경우 별도의 ActionScript 파일에 고유한 문서 클래스를 만들지 않아도 기본 문서 클래스가 자동으로 만들어집니다. 스테이지 nativeWindow 속성을 사용하여 초기 윈도우의 NativeWindow 객체에 액세스할 수 있습니다. 예를 들어 다음 코드는 타임라인에서 최대화된 상태로 기본 윈도우를 활성화합니다.

```
import flash.display.NativeWindow;

var mainWindow:NativeWindow = this.stage.nativeWindow;
mainWindow.maximize();
mainWindow.activate();
```

Flex를 사용하여 초기 윈도우 만들기

Adobe AIR 1.0 이상

Flex 프레임워크를 사용하여 AIR 응용 프로그램을 만드는 경우 mx:WindowedApplication을 기본 MXML 파일의 루트 요소로 사용합니다. mx:Application 구성 요소를 사용할 수 있지만 이 구성 요소는 AIR에서 사용할 수 있는 기능 중 일부를 지원하지 않습니다. WindowedApplication 구성 요소는 응용 프로그램의 초기 진입점 역할을 합니다.

응용 프로그램을 시작할 때 AIR에서는 기본 윈도우를 만들고 Flex 프레임워크를 초기화한 다음 WindowedApplication 객체를 윈도우 스테이지에 추가합니다. 시작 시퀀스가 완료되면 WindowedApplication은 applicationComplete 이벤트를 전달합니다. WindowedApplication 인스턴스의 nativeWindow 속성을 사용하여 데스크톱 윈도우 객체에 액세스합니다.

다음 예제에서는 x 및 y 좌표를 설정하는 간단한 WindowedApplication 구성 요소를 만듭니다.

```
 <?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>
```

NativeWindow 만들기

Adobe AIR 1.0 이상

NativeWindow를 만들려면 NativeWindowInitOptions 객체를 NativeWindow 생성자에 전달합니다.

```
 var options:NativeWindowInitOptions = new NativeWindowInitOptions();
options.systemChrome = NativeWindowSystemChrome.STANDARD;
options.transparent = false;
var newWindow:NativeWindow = new NativeWindow(options);
```

visible 속성을 true로 설정하거나 activate() 메서드를 호출할 때까지 윈도우가 표시되지 않습니다.

윈도우가 만들어지면 윈도우의 속성을 초기화하고 스테이지 속성과 Flash 표시 목록 방법을 사용하여 내용을 윈도우에 로드합니다.

거의 모든 경우 새로운 기본 윈도우의 스테이지 scaleMode 속성을 noScale로 설정해야 합니다(StageScaleMode.NO_SCALE 상수 사용). Flash 크기 조절 모드는 응용 프로그램 작성자가 응용 프로그램 표시 공간의 종횡비를 미리 알 수 없는 경우를 위해 설계되었습니다. 크기 조절 모드를 사용하면 작성자가 내용을 자르거나, 내용을 잡아 늘이거나 납작하게 하거나, 빈 공간으로 채우는 등의 작업 중에서 손상이 가장 적은 방법을 선택할 수 있습니다. AIR에서 표시 공간(윈도우 프레임)을 제어하므로 손상 없이 윈도우의 크기를 내용에 맞게 조절하거나 내용의 크기를 윈도우에 맞게 조절할 수 있습니다.

Flex 및 HTML 윈도우의 크기 조절 모드는 noScale로 자동으로 설정됩니다.

참고: 현재 운영 체제에서 허용되는 최대 및 최소 윈도우 크기를 확인하려면 다음과 같은 정적 NativeWindow 속성을 사용합니다.

```
var maxOSSize:Point = NativeWindow.systemMaxSize;
var minOSSize:Point = NativeWindow.systemMinSize;
```

HTML 윈도우 만들기

Adobe AIR 1.0 이상

HTML 윈도우를 만들려면 JavaScript Window.open() 메서드를 호출하거나 AIR HTMLLoader 클래스 createRootWindow() 메서드를 호출할 수 있습니다.

보안 샌드박스의 HTML 내용은 표준 JavaScript Window.open() 메서드를 사용할 수 있습니다. 내용이 응용 프로그램 샌드박스 밖에서 실행되는 경우에는 마우스 클릭이나 키 누르기와 같은 사용자 상호 작용에 대한 응답으로만 open() 메서드를 호출할 수 있습니다. open()이 호출되면 지정된 URL의 내용을 표시하기 위해 시스템 크롬을 사용하여 윈도우가 만들어집니다. 예를 들면 다음과 같습니다.

```
newWindow = window.open("xmpl.html", "logWindow", "height=600, width=400, top=10, left=10");
```

참고: ActionScript에서 HTMLHost 클래스를 확장하여 JavaScript window.open() 함수로 만든 윈도우를 사용자 정의할 수 있습니다. 자세한 내용은 917페이지의 “HTMLHost 클래스 확장”을 참조하십시오.

응용 프로그램 보안 샌드박스의 내용은 윈도우를 만드는 더욱 강력한 메서드인 HTMLLoader.createRootWindow()에 액세스할 수 있습니다. 이 메서드를 사용하면 새 윈도우를 만드는 모든 옵션을 지정할 수 있습니다. 예를 들어 다음 JavaScript 코드에서는 크기가 300x400 픽셀이고 시스템 크롬을 사용하지 않는 경량 유형의 윈도우를 만듭니다.

```
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"));
```

참고: 새 윈도우에서 로드된 내용이 응용 프로그램 보안 샌드박스 밖에 있는 경우 window 객체에는 AIR 속성 runtime, nativeWindow 또는 htmlLoader가 없습니다.

투명 윈도우를 만들면 해당 윈도우에 로드되는 HTML에 포함된 SWF 내용이 표시되지 않는 경우도 있습니다. 객체의 wmode 매개 변수나 SWF 파일을 참조하는 데 사용되는 embed 태그를 opaque 또는 transparent로 설정해야 합니다. wmode의 기본값은 window이므로 기본적으로 SWF 내용은 투명 윈도우에 표시되지 않습니다. 어떤 wmode 값을 설정하더라도 PDF 내용은 투명 윈도우에 표시할 수 없습니다. AIR 1.5.2 이전에는 SWF 내용도 투명 윈도우에 표시할 수 없었습니다.

createRootWindow() 메서드로 만든 윈도우는 여는 윈도우와 독립적으로 유지됩니다. JavaScript Window 객체의 parent 및 opener 속성은 null입니다. 여는 윈도우는 createRootWindow() 함수에서 반환되는 HTMLLoader 참조를 사용하여 새 윈도우의 Window 객체에 액세스할 수 있습니다. 이전 예제의 맥락에서 newHTMLLoader.window 문은 만들어진 윈도우의 JavaScript Window 객체를 참조합니다.

참고: createRootWindow() 함수는 JavaScript와 ActionScript에서 모두 호출할 수 있습니다.

mx:Window 만들기

Adobe AIR 1.0 이상

mx:Window를 만들려면 mx:Window를 루트 태그로 사용하여 MXML 파일을 만들거나 Window 클래스 생성자를 직접 호출할 수 있습니다.

다음 예제에서는 Window 생성자를 호출하여 mx:Window를 만들고 표시합니다.

```
 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);
```

윈도우에 내용 추가

Adobe AIR 1.0 이상

AIR 윈도우에 내용을 추가하는 방법은 윈도우 유형에 따라 달라집니다. 예를 들어 MXML 및 HTML을 사용하면 윈도우의 기본 내용을 선언적으로 정의할 수 있습니다. 응용 프로그램 SWF 파일에 리소스를 포함하거나 별도의 응용 프로그램 파일에서 리소스를 로드할 수 있습니다. Flex, Flash 및 HTML 내용은 모두 즉석에서 만들어 윈도우에 동적으로 추가할 수 있습니다.

SWF 내용이나 JavaScript가 포함된 HTML 내용을 로드할 때 AIR 보안 모델을 고려해야 합니다. 응용 프로그램 보안 샌드박스의 내용, 즉 응용 프로그램과 함께 설치되고 app: URL 스킴으로 로드 가능한 내용은 모든 AIR API에 액세스할 수 있는 전체 권한을 가집니다. 이 샌드박스 밖에서 로드된 내용은 AIR API에 액세스할 수 없습니다. 응용 프로그램 샌드박스 밖의 JavaScript 내용은 JavaScript Window 객체의 runtime, nativeWindow 또는 htmlLoader 속성을 사용할 수 없습니다.

안전한 크로스 스크립팅을 허용하려면 샌드박스 브리지를 사용하여 응용 프로그램 내용과 비 응용 프로그램 내용 간의 제한된 인터페이스를 제공할 수 있습니다. HTML 내용에서 응용 프로그램의 페이지를 비 응용 프로그램 샌드박스에 매핑하여 해당 페이지의 코드에서 외부 내용의 크로스 스크립팅을 허용할 수도 있습니다. 자세한 내용은 979페이지의 “AIR 보안”을 참조하십시오.

SWF 파일 또는 이미지 로드

flash.display.Loader 클래스를 사용하여 Flash SWF 파일 또는 이미지를 기본 윈도우의 표시 목록에 로드할 수 있습니다.

```
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);
        }
    }
}
```

참고: ActionScript 1 또는 2를 사용하여 만든 이전 SWF 파일은 동일한 윈도우에 로드된 경우 클래스 정의, singleton 및 전역 변수와 같은 전역 상태를 공유합니다. 이러한 SWF 파일이 제대로 작동하려면 전역 상태가 변경되지 않아야 하는 경우 이러한 파일을 동일한 윈도우에 두 번 이상 로드하거나 겹치는 클래스 정의와 변수를 사용하여 다른 SWF 파일과 동일한 윈도우에 로드할 수 없습니다. 이 내용은 별도의 윈도우에 로드할 수 있습니다.

NativeWindow에 HTML 내용 로드

HTML 내용을 NativeWindow에 로드하려면 HTMLLoader 객체를 윈도우 스테이지에 추가하고 HTML 내용을 HTMLLoader에 로드하거나, HTMLLoader.createRootWindow() 메서드를 사용하여 HTMLLoader 객체가 이미 포함된 윈도우를 만들 수 있습니다. 다음 예제에서는 기본 윈도우 스테이지의 300x500 픽셀 표시 영역에서 HTML 내용을 표시합니다.

```
 //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) );
```

HTML 페이지를 Flex 응용 프로그램에 로드하려면 Flex HTML 구성 요소를 사용할 수 있습니다.

윈도우에서 투명도를 사용하는 경우(즉, 윈도우의 transparent 속성이 true인 경우) 객체의 wmode 매개 변수나 SWF 파일을 참조하는 데 사용되는 embed 태그를 opaque 또는 transparent로 설정하지 않으면 HTML 파일의 SWF 내용이 표시되지 않습니다. 기본 wmode 값은 window이므로 기본적으로 SWF 내용은 투명 윈도우에 표시되지 않습니다. PDF 내용은 어떤 wmode 값을 사용하더라도 투명 윈도우에 표시되지 않습니다.

또한 HTMLLoader 컨트롤이 크기 조절 또는 회전되거나 HTMLLoader alpha 속성이 1.0 이외의 값으로 설정되는 경우에는 SWF 내용과 PDF 내용이 모두 표시되지 않습니다.

HTML 윈도우에서 오버레이로 SWF 내용 추가

HTML 윈도우가 NativeWindow 인스턴스에 포함되기 때문에 Flash 표시 객체를 표시 목록에서 HTML 레이어 위와 아래에 모두 추가할 수 있습니다.

표시 객체를 HTML 레이어 위에 추가하려면 window.nativeWindow.stage 속성의 addChild() 메서드를 사용합니다. addChild() 메서드는 윈도우에서 기존 내용 위에 레이어로 내용을 추가합니다.

HTML 레이어 아래에 표시 객체를 추가하려면 window.nativeWindow.stage 속성의 addChildAt() 메서드를 사용하여 index 매개 변수에 0 값을 전달합니다. 객체를 인덱스 0에 배치하면 HTML 표시가 포함된 기존 내용이 한 레이어 위로 이동하고 새 내용이 맨 아래에 삽입됩니다. HTML 페이지 아래에 있는 레이어의 내용이 표시되려면 HTMLlLoader 객체의 paintsDefaultBackground 속성을 false로 설정해야 합니다. 또한 배경색을 설정하는 페이지의 모든 요소가 투명하게 표시되지 않습니다. 예를 들어 페이지의 body 요소에 대한 배경색을 설정하는 경우 페이지가 모두 투명하게 표시되지 않습니다.

다음 예제에서는 Flash 표시 객체를 오버레이로 추가하고 HTML 페이지 아래에 삽입하는 방법을 보여 줍니다. 이 예제에서는 두 가지 간단한 모양 객체를 만들어 그 중 하나를 HTML 내용 아래에 추가하고 다른 하나를 위에 추가합니다. 또한 enterFrame 이벤트에 따라 모양 위치도 업데이트합니다.

```
 <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>
```

이 예제에서는 AIR의 ActionScript와 JavaScript 간 경계를 넘는 고급 방법에 대한 기본적인 내용을 소개합니다. ActionScript 표시 객체 사용 방법에 익숙하지 않은 경우 135페이지의 "디스플레이 프로그래밍"(**ActionScript 3.0** 개발자 안내서)을 참조하십시오.

예제: 기본 윈도우 만들기

Adobe AIR 1.0 이상

다음 예제에서는 기본 윈도우를 만드는 방법을 보여 줍니다.

```
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();
}
```

윈도우 관리

Adobe AIR 1.0 이상

NativeWindow 클래스의 속성과 메서드를 사용하여 데스크톱 윈도우의 모양, 비헤이비어 및 수명 주기를 관리합니다.

참고: Flex 프레임워크를 사용하는 경우 일반적으로 프레임워크 클래스를 사용하여 윈도우 비헤이비어를 관리하는 것이 좋습니다. 대부분의 NativeWindow 속성 및 메서드는 mx:WindowedApplication 및 mx:Window 클래스를 통해 액세스할 수 있습니다.

NativeWindow 인스턴스 얻기

Adobe AIR 1.0 이상

윈도우를 조작하려면 먼저 윈도우 인스턴스를 얻어야 합니다. 다음 중 하나에서 윈도우 인스턴스를 얻을 수 있습니다.

- 윈도우를 만드는 데 사용되는 기본 윈도우 생성자:

```
var win:NativeWindow = new NativeWindow(initOptions);
```

- 윈도우 스테이지의 nativeWindow 속성:

```
var win:NativeWindow = stage.nativeWindow;
```

- 윈도우에 있는 표시 객체의 stage 속성:

```
var win:NativeWindow = displayObject.stage.nativeWindow;
```

- 윈도우에서 전달하는 기본 윈도우 이벤트의 target 속성:

```
private function onNativeWindowEvent(event:NativeWindowBoundsEvent):void
{
    var win:NativeWindow = event.target as NativeWindow;
}
```

- 윈도우에 표시되는 HTML 페이지의 nativeWindow 속성:

```
var win:NativeWindow = htmlLoader.window.nativeWindow;
```

- NativeApplication 객체의 activeWindow 및 openedWindows 속성:

```
var nativeWin:NativeWindow = NativeApplication.nativeApplication.activeWindow;
var firstWindow:NativeWindow = NativeApplication.nativeApplication.openedWindows[0];
```

  NativeApplication.nativeApplication.activeWindow는 응용 프로그램의 활성 윈도우를 참조합니다. 그러나 활성 윈도우가 이 AIR 응용 프로그램의 윈도우가 아닌 경우 null을 반환합니다. NativeApplication.nativeApplication.openedWindows 배열에는 AIR 응용 프로그램에서 닫히지 않은 모든 윈도우가 포함됩니다.

Flex mx:WindowedApplication 및 mx:Window 객체가 표시 객체이기 때문에 다음과 같이 stage 속성을 사용하여 MXML 파일에서 응용 프로그램 윈도우를 쉽게 참조할 수 있습니다.

```
<?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
```

참고: Flex 프레임워크에서 WindowedApplication 또는 Window 구성 요소를 윈도우 스테이지에 추가할 때까지 구성 요소의 stage 속성은 null입니다. 이 비헤이비어는 Flex 응용 프로그램 구성 요소의 비헤이비어와 일치하지만 creationComplete와 같이 WindowedApplication 및 Window 구성 요소의 초기화 주기에서 초기에 발생하는 이벤트의 리스너에서 스테이지나 NativeWindow 인스턴스에 액세스할 수 없습니다. applicationComplete 이벤트가 전달될 때 스테이지와 NativeWindow 인스턴스에 액세스하는 것이 안전합니다.

윈도우 활성화, 표시 및 숨기기

Adobe AIR 1.0 이상

윈도우를 활성화하려면 NativeWindow activate() 메서드를 호출합니다. 윈도우를 활성화하면 윈도우가 맨 앞에 배치되고 키보드 및 마우스 포커스를 갖게 되며 필요한 경우 윈도우를 복원하거나 visible 속성을 true로 설정하여 윈도우를 표시할 수 있습니다. 윈도우를 활성화할 때 응용 프로그램에서 다른 윈도우의 순서는 변경되지 않습니다. activate() 메서드를 호출하면 윈도우에서 activate 이벤트를 전달합니다.

활성화하지 않고 숨겨진 윈도우를 표시하려면 visible 속성을 true로 설정합니다. 이렇게 하면 윈도우가 맨 앞에 배치되지만 윈도우에 포커스가 할당되지 않습니다.

윈도우를 숨기려면 visible 속성을 false로 설정합니다. 윈도우를 숨기면 윈도우, 관련 작업 표시줄 아이콘 및 Mac OS X에서는 Windows 메뉴의 항목이 표시되지 않습니다.

윈도우의 가시성을 변경하면 해당 윈도우가 소유한 모든 윈도우의 가시성 또한 변경됩니다. 예를 들어 특정 윈도우를 숨기는 경우 이 윈도우가 소유한 모든 윈도우도 숨겨집니다.

참고: Mac OS X에서는 도크의 윈도우 부분에 아이콘이 있는 최소화된 윈도우를 완전히 숨길 수 없습니다. 최소화된 윈도우에서 visible 속성이 false로 설정되어 있는 경우 윈도우의 도크 아이콘이 여전히 표시됩니다. 사용자가 아이콘을 클릭하면 윈도우가 표시 상태로 복원되고 표시됩니다.

윈도우 표시 순서 변경

Adobe AIR 1.0 이상

AIR에서는 윈도우의 표시 순서를 직접 변경하는 몇 가지 방법을 제공합니다. 윈도우를 표시 순서의 앞이나 뒤로 이동하거나 다른 윈도우의 위나 아래로 이동할 수 있습니다. 이와 동시에 사용자가 윈도우를 활성화하여 윈도우의 순서를 재배열할 수 있습니다.

alwaysInFront 속성을 true로 설정하여 다른 윈도우 앞에 윈도우를 유지할 수 있습니다. 두 개 이상의 윈도우에 이 설정이 있으면 이러한 윈도우의 표시 순서가 서로 정렬되지만 이러한 윈도우는 alwaysInFront가 false로 설정된 윈도우 위에 항상 정렬됩니다.

또한 AIR 응용 프로그램이 활성화되어 있지 않은 경우에도 최상위 그룹의 윈도우가 다른 응용 프로그램의 윈도우 위에 표시됩니다. 이 비헤이비어는 사용자에게 방해가 될 수 있기 때문에 필요하고 적절한 경우에만 alwaysInFront를 true로 설정해야 합니다. 이러한 설정이 적합한 예는 다음과 같습니다.

- 도구 설명, 팝업 목록, 사용자 정의 메뉴 또는 콤보 상자와 같은 컨트롤의 임시 팝업 윈도우. 이러한 윈도우는 포커스를 잃으면 닫혀야 하기 때문에 사용자가 다른 윈도우를 보지 못하게 차단하는 불편을 방지할 수 있습니다.
- 매우 긴급한 오류 메시지 및 경고. 사용자가 시기 적절하게 응답하지 않으면 취소할 수 없는 변경이 발생할 수 있는 경우 경고 윈도우를 맨 앞에 배치하는 것이 적절할 수 있습니다. 그러나 대부분의 오류와 경고는 일반 윈도우 표시 순서로 처리될 수 있습니다.
- 잠시 사용되는 토스트 스타일 윈도우

참고: AIR에서는 alwaysInFront 속성을 적절히 사용하도록 강제하지 않습니다. 그러나 응용 프로그램이 사용자의 작업 과정을 방해하는 경우 해당 사용자의 휴지통에 들어가게 될 수 있습니다.

특정 윈도우가 다른 여러 윈도우를 소유한 경우 이러한 윈도우는 항상 해당 윈도우 앞에 정렬됩니다. 다른 윈도우를 소유한 윈도우에서 orderToFront()를 호출하거나 alwaysInFront를 true로 설정하면 소유된 윈도우가 소유 윈도우와 함께 다른 윈도우들 앞에 다시 정렬되지만 소유된 윈도우는 여전히 소유 윈도우 앞에 표시됩니다.

소유된 윈도우에서 윈도우 정렬 메서드를 호출하는 작업은 동일한 윈도우에 소유된 윈도우 간에 정상적으로 작동하지만 전체 소유된 윈도우 그룹의 순서 또한 해당 그룹 외부의 윈도우와 비교해 변경될 수 있습니다. 예를 들여 소유된 윈도우에서 orderToFront()를 호출하면 해당 윈도우와 소유 윈도우, 그리고 이 윈도우에서 소유한 다른 모든 윈도우가 윈도우 표시 순서의 맨 앞으로 이동합니다.

NativeWindow 클래스는 다른 윈도우에 상대적으로 윈도우의 표시 순서를 설정하기 위한 다음 속성과 메서드를 제공합니다.

멤버	설명
alwaysInFront 속성	윈도우의 최상위 그룹에서 윈도우가 표시되는지 여부를 지정합니다. 거의 모든 경우에 false가 최상의 설정입니다. 값을 false에서 true로 변경하면 윈도우가 모든 윈도우 앞에 배치되지만 활성화되지는 않습니다. 값을 true에서 false로 변경하면 최상위 그룹에 남아 있는 윈도우 뒤에 윈도우가 배치되지만 여전히 다른 윈도우 앞에 있습니다. 이 속성을 윈도우의 현재 값으로 설정하면 윈도우 표시 순서가 변경되지 않습니다. alwaysInFront 설정은 다른 윈도우에 소유된 윈도우에 아무런 영향을 미치지 않습니다.
orderToFront()	윈도우를 맨 앞으로 가져옵니다.
orderInFrontOf()	윈도우를 특정 윈도우의 바로 앞으로 가져옵니다.
orderToBack()	윈도우를 다른 윈도우 뒤로 보냅니다.
orderBehind()	윈도우를 특정 윈도우의 바로 뒤로 보냅니다.
activate()	윈도우를 맨 앞으로 가져옵니다(윈도우 표시 및 포커스 할당과 함께).

참고: 윈도우가 숨겨져 있거나(visible이 false인 경우) 최소화된 경우 표시 순서 메서드를 호출해도 아무 효과가 없습니다.

Linux 운영 체제에서는 윈도우 표시 순서와 관련하여 윈도우 관리자마다 서로 다른 규칙이 적용됩니다.

- 일부 윈도우 관리자에서 유틸리티 윈도우는 항상 일반 윈도우 앞에 표시됩니다.
- 일부 윈도우 관리자에서 alwaysInFront가 true로 설정된 전체 화면 윈도우는 항상 alwaysInFront가 동일하게 true로 설정된 다른 윈도우 앞에 표시됩니다.

윈도우 닫기

Adobe AIR 1.0 이상

윈도우를 닫으려면 NativeWindow.close() 메서드를 사용합니다.

윈도우를 닫으면 윈도우의 내용이 언로드되지만 다른 객체가 이 내용에 대한 참조를 가진 경우 내용 객체가 삭제되지 않습니다. NativeWindow.close() 메서드는 비동기적으로 실행되며 윈도우에 포함된 응용 프로그램이 닫는 과정 중에 계속 실행됩니다. close 메서드는 닫기 작업이 완료되면 close 이벤트를 전달합니다. NativeWindow 객체는 기술적으로 유효하지만 닫힌 윈도우에 대한 속성과 메서드에 액세스하는 경우 대부분 IllegalOperationError가 생성됩니다. 닫힌 윈도우는 다시 열 수 없습니다. 윈도우의 closed 속성을 확인하여 윈도우가 닫혔는지 테스트할 수 있습니다. 윈도우를 숨기려면 NativeWindow.visible 속성을 false로 설정합니다.

Nativeapplication.autoExit 속성이 기본값인 true이면 마지막 윈도우가 닫힐 때 응용 프로그램이 종료됩니다.

소유 윈도우가 있는 모든 윈도우는 해당 소유 윈도우가 닫히면 모두 닫힙니다. 소유된 윈도우는 closing 이벤트를 전달하지 않기 때문에 클로저를 방지할 수 없습니다. close 이벤트가 전달됩니다.

윈도우 작업의 취소 허용

Adobe AIR 1.0 이상

윈도우에서 시스템 크롬을 사용하는 경우 적절한 이벤트를 수신하고 해당 이벤트의 기본 비헤이비어를 취소하여 윈도우와 사용자의 상호 작용을 취소할 수 있습니다. 예를 들어 사용자가 시스템 크롬 닫기 버튼을 클릭하면 closing 이벤트가 전달됩니다. 등록된 리스너에서 이벤트의 preventDefault() 메서드를 호출하면 윈도우가 닫히지 않습니다.

윈도우에서 시스템 크롬을 사용하지 않는 경우에는 의도한 변경의 알림 이벤트가 변경이 수행되기 전에 자동으로 전달되지 않습니다. 따라서 윈도우를 닫기 위한 메서드를 호출하여 윈도우 상태를 변경하거나 윈도우 경계 속성을 설정하면 변경을 취소할 수 없습니다. 윈도우를 변경하기 전에 응용 프로그램의 구성 요소에 알리려면 응용 프로그램 논리에서는 윈도우의 dispatchEvent() 메서드를 사용하여 관련 알림 이벤트를 전달할 수 있습니다.

예를 들어 다음 논리에서는 윈도우 닫기 버튼에 대한 취소 가능한 이벤트 핸들러를 구현합니다.

```
public function onCloseCommand(event:MouseEvent):void{
    var closingEvent:Event = new Event(Event.CLOSING,true,true);
    dispatchEvent(closing);
    if(!closingEvent.isDefaultPrevented()){
        win.close();
    }
}
```

리스너에서 이벤트 preventDefault() 메서드를 호출하는 경우 dispatchEvent() 메서드는 false를 반환합니다. 그러나 다른 이유로도 false를 반환할 수 있으므로 isDefaultPrevented() 메서드를 명시적으로 사용하여 변경을 취소해야 할지 여부를 테스트하는 것이 좋습니다.

윈도우 최대화, 최소화 및 복원

Adobe AIR 1.0 이상

윈도우를 최대화하려면 NativeWindow maximize() 메서드를 사용합니다.

```
myWindow.maximize();
```

윈도우를 최소화하려면 NativeWindow minimize() 메서드를 사용합니다.

```
myWindow.minimize();
```

윈도우를 복원하려면(즉, 이전의 최소화되었거나 최대화된 크기로 되돌리려면) NativeWindow restore() 메서드를 사용합니다.

```
myWindow.restore();
```

소유 윈도우가 있는 윈도우는 소유 윈도우가 최소화되거나 복원되면 함께 최소화되고 복원됩니다. 소유된 윈도우는 소유 윈도우가 최소화되어 함께 최소화된 경우에 아무런 이벤트도 전달하지 않습니다.

참고: AIR 윈도우 최대화로 인한 비헤이비어는 Mac OS X 표준 비헤이비어와 다릅니다. 응용 프로그램에서 정의한 "표준" 크기와 사용자가 설정한 마지막 크기 간을 전환하는 대신 AIR 윈도우는 응용 프로그램이나 사용자가 마지막으로 설정한 크기와 화면의 사용 가능한 전체 영역 간을 전환합니다.

Linux 운영 체제에서는 윈도우 표시 상태 설정과 관련하여 윈도우 관리자마다 서로 다른 규칙이 적용됩니다.

- 일부 윈도우 관리자에서는 유틸리티 윈도우를 최대화할 수 없습니다.
- 윈도우 크기가 최대로 설정된 경우에는 일부 윈도우에서 윈도우를 최대화할 수 없습니다. 일부 다른 윈도우 관리자는 표시 상태를 최대화된 상태로 설정하지만 윈도우 크기를 조절하지 않습니다. 이러한 두 경우 모두 표시 상태 변경 이벤트가 전달되지 않습니다.
- 일부 윈도우 관리자에서는 윈도우 maximizable 또는 minimizable 설정이 유지되지 않습니다.

참고: Linux에서 윈도우 속성은 비동기적으로 변경됩니다. 프로그램의 한 행에서 표시 상태를 변경하고 다음 행에서 값을 읽으면 읽는 값에 여전히 이전 설정이 반영됩니다. 모든 플랫폼에서 NativeWindow 객체는 표시 상태가 변경되면 displayStateChange 이벤트를 전달합니다. 윈도우의 새로운 상태에 따라 어떤 액션을 취해야 하는 경우에는 항상 displayStateChange 이벤트 핸들러를 사용합니다. 자세한 내용은 832페이지의 "윈도우 이벤트 수신"을 참조하십시오.

예제: 윈도우 최소화, 최대화, 복원 및 닫기

Adobe AIR 1.0 이상

아래의 간단한 MXML 응용 프로그램에서는 Window maximize(), minimize(), restore() 및 close() 메서드를 보여 줍니다.

```
<?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>
```

아래의 Flash용 ActionScript 예제에서는 NativeWindow minimize(), maximize(), restore() 및 close() 메서드를 트리거하는 클릭 가능한 네 가지 텍스트 필드를 만듭니다.

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

윈도우 크기 조절 및 이동

Adobe AIR 1.0 이상

윈도우에서 시스템 크롬을 사용하는 경우 시스템 크롬은 윈도우 크기 조절과 바탕 화면 위의 이동을 위한 드래그 컨트롤을 제공합니다. 윈도우에서 시스템 크롬을 사용하지 않는 경우에는 사용자가 윈도우 크기를 조절하고 이동할 수 있도록 사용자 고유의 컨트롤을 추가해야 합니다.

참고: 윈도우 크기를 조절하고 이동하려면 먼저 NativeWindow 인스턴스에 대한 참조를 얻어야 합니다. 윈도우 참조를 얻는 방법에 대한 자세한 내용은 824페이지의 “NativeWindow 인스턴스 얻기”를 참조하십시오.

윈도우 크기 조절

사용자가 대화식으로 윈도우 크기를 조절할 수 있도록 하려면 NativeWindow startResize() 메서드를 사용합니다. 이 메서드가 mouseDown 이벤트에서 호출되면 크기 조절 작업이 마우스로 작동되고 운영 체제에서 mouseUp 이벤트를 받으면 완료됩니다. startResize()를 호출할 때 윈도우 크기를 조절할 가장자리나 모서리를 지정하는 인수를 전달합니다.

윈도우 크기를 프로그래밍 방식으로 설정하려면 윈도우의 width, height 또는 bounds 속성을 원하는 크기로 설정합니다. 경계를 설정하면 윈도우 크기 및 위치를 동시에 변경할 수 있습니다. 그러나 변경되는 순서는 확실하지 않습니다. 일부 Linux 윈도우 관리자에서는 윈도우를 데스크톱 화면의 경계 밖으로 확장할 수 없습니다. 이러한 경우 변경 사항을 적용한 결과 유효한 윈도우가 생성되더라도 속성이 설정되는 순서 때문에 최종 윈도우 크기가 제한될 수 있습니다. 예를 들어 화면 아래쪽 근처에 있는 윈도우의 높이와 y 위치를 모두 변경하는 경우 높이 변경 사항이 적용된 후에 y 위치가 변경되면 전체 높이가 변경되지 않을 수 있습니다.

참고: Linux에서 윈도우 속성은 비동기적으로 변경됩니다. 프로그램의 한 행에서 윈도우의 크기를 변경하고 다음 행에서 크기를 읽으면 읽는 값에 여전히 이전 설정이 반영됩니다. 모든 플랫폼에서 NativeWindow 객체는 윈도우 크기가 변경되면 resize 이벤트를 전달합니다. 윈도우의 새로운 상태 또는 크기에 따라 윈도우에서 컨트롤을 배치하는 등의 어떤 액션을 취해야 하는 경우에는 항상 resize 이벤트 핸들러를 사용합니다. 자세한 내용은 832페이지의 “윈도우 이벤트 수신”을 참조하십시오.

스테이지의 크기 조절 모드는 윈도우 크기가 조절될 때 윈도우 스테이지와 그 내용이 동작하는 방식을 결정합니다. 스테이지 크기 조절 모드가 응용 프로그램이 표시 공간의 크기나 종횡비를 제어하지 않는 웹 브라우저와 같은 경우를 위해 설계되었음을 명심하십시오. 일반적으로 스테이지 scaleMode 속성을 StageScaleMode.NO_SCALE로 설정하여 최상의 결과를 얻을 수 있습니다. 윈도우의 내용 크기를 조절하려는 경우 여전히 윈도우 경계 변경에 대한 응답으로 내용의 scaleX 및 scaleY 매개 변수를 설정할 수 있습니다.

윈도우 이동

크기를 조절하지 않고 윈도우를 이동하려면 NativeWindow startMove() 메서드를 사용합니다. startResize() 메서드와 마찬가지로, startMove() 메서드가 mouseDown 이벤트에서 호출되면 이동 프로세스가 마우스로 작동되고 운영 체제에서 mouseUp 이벤트를 받으면 완료됩니다.

startResize() 및 startMove() 메서드에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

윈도우를 프로그래밍 방식으로 이동하려면 윈도우의 x, y 또는 bounds 속성을 원하는 위치로 설정합니다. 경계를 설정하면 윈도우 크기 및 위치를 동시에 변경할 수 있습니다.

참고: Linux에서 윈도우 속성은 비동기적으로 변경됩니다. 프로그램의 한 행에서 윈도우를 이동하고 다음 행에서 위치를 읽으면 읽는 값에 여전히 이전 설정이 반영됩니다. 모든 플랫폼에서 NativeWindow 객체는 위치가 변경되면 move 이벤트를 전달합니다. 윈도우의 새로운 위치에 따라 어떤 액션을 취해야 하는 경우에는 항상 move 이벤트 핸들러를 사용합니다. 자세한 내용은 832페이지의 “윈도우 이벤트 수신”을 참조하십시오.

예제: 윈도우 크기 조절 및 이동

Adobe AIR 1.0 이상

다음 예제에서는 윈도우에서 크기 조절 및 이동 작업을 시작하는 방법을 보여 줍니다.

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

윈도우 이벤트 수신

Adobe AIR 1.0 이상

윈도우에서 전달하는 이벤트를 수신하려면 윈도우 인스턴스에 리스너를 등록합니다. 예를 들어 closing 이벤트를 수신하려면 다음과 같이 윈도우에 리스너를 등록합니다.

```
myWindow.addEventListener(Event.CLOSING, onClosingEvent);
```

이벤트가 전달되면 target 속성이 이벤트를 보내는 윈도우를 참조합니다.

대부분의 윈도우 이벤트에는 두 가지 관련 메시지가 있습니다. 첫 번째 메시지는 윈도우 변경이 발생하려고 하고 취소할 수 있음을 알리고, 두 번째 메시지는 변경이 발생했음을 알립니다. 예를 들어 윈도우의 닫기 버튼을 클릭하면 closing 이벤트 메시지가 전달됩니다. 리스너가 이 이벤트를 취소하지 않으면 윈도우가 닫히고 close 이벤트가 모든 리스너에 전달됩니다.

일반적으로 closing과 같은 경고 이벤트는 시스템 크롬이 이벤트를 트리거하는 데 사용된 경우에만 전달됩니다. 예를 들어 윈도우 close() 메서드를 호출하면 closing 이벤트가 자동으로 전달되지 않고 close 이벤트만 전달됩니다. 그러나 closing 이벤트 객체를 생성하고 윈도우 dispatchEvent() 메서드를 사용하여 전달할 수 있습니다.

Event 객체를 전달하는 윈도우 이벤트는 다음과 같습니다.

이벤트	설명
activate	윈도우가 포커스를 받으면 전달됩니다.
deactivate	윈도우가 포커스를 잃으면 전달됩니다.
closing	윈도우가 닫히려고 하면 전달됩니다. 이 이벤트는 시스템 크롬 닫기 버튼을 누르거나 Mac OS X에서 종료 명령을 호출하는 경우에만 자동으로 전달됩니다.
close	윈도우가 닫히면 전달됩니다.

NativeWindowBoundsEvent 객체를 전달하는 윈도우 이벤트는 다음과 같습니다.

이벤트	설명
moving	윈도우 이동, 크기 조절 또는 표시 상태 변경의 결과로 윈도우의 왼쪽 위 모서리 위치가 변경되기 직전에 전달됩니다.
move	왼쪽 위 모서리 위치가 변경되면 전달됩니다.
resizing	크기 조절 또는 표시 상태 변경의 결과로 윈도우 폭이나 높이가 변경되기 직전에 전달됩니다.
resize	윈도우 크기가 변경된 후 전달됩니다.

NativeWindowBoundsEvent 이벤트의 경우 beforeBounds 및 afterBounds 속성을 사용하여 임박한 변경이나 완료된 변경 전과 후의 윈도우 경계를 확인할 수 있습니다.

NativeWindowDisplayStateEvent 객체를 전달하는 윈도우 이벤트는 다음과 같습니다.

이벤트	설명
displayStateChanging	윈도우 표시 상태가 변경되기 직전에 전달됩니다.
displayStateChange	윈도우 표시 상태가 변경된 후 전달됩니다.

NativeWindowDisplayStateEvent 이벤트의 경우 beforeDisplayState 및 afterDisplayState 속성을 사용하여 임박한 변경이나 완료된 변경 전과 후의 윈도우 표시 상태를 확인할 수 있습니다.

일부 Linux 윈도우 관리자에서는 최대 크기 설정이 지정된 윈도우가 최대화되면 표시 상태 변경 이벤트가 전달되지 않습니다. 이때 윈도우가 최대화된 표시 상태로 설정되지만 크기는 조절되지 않습니다.

전체 화면 윈도우 표시

Adobe AIR 1.0 이상

Stage의 displayState 속성을 StageDisplayState.FULL_SCREEN_INTERACTIVE로 설정하면 윈도우가 전체 화면 모드로 전환되며 이 모드에서는 키보드 입력이 **허용됩니다**. 브라우저에서 실행되는 SWF 내용에서는 키보드 입력이 허용되지 않습니다. 전체 화면 모드를 종료하려면 Esc 키를 누릅니다.

참고: 일부 Linux 윈도우 관리자에서는 윈도우의 최대 크기가 설정된 경우 화면을 채우도록 윈도우 크기가 변경되지 않습니다(윈도우 시스템 크롬은 제거되지 않음).

예를 들어 다음 Flex 코드에서는 간단한 전체 화면 터미널을 설정하는 간략한 AIR 응용 프로그램을 정의합니다.

```
<?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>
```

아래의 Flash용 ActionScript 예제에서는 간단한 전체 화면 텍스트 터미널을 시뮬레이션합니다.

```
    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);
        }
    }
```

53장: AIR의 표시 화면

Adobe AIR 1.0 이상

Adobe® AIR® Screen 클래스를 사용하여 컴퓨터 또는 장치에 연결된 표시 화면 정보에 액세스할 수 있습니다.

기타 도움말 항목
flash.display.Screen

AIR 표시 화면의 기초

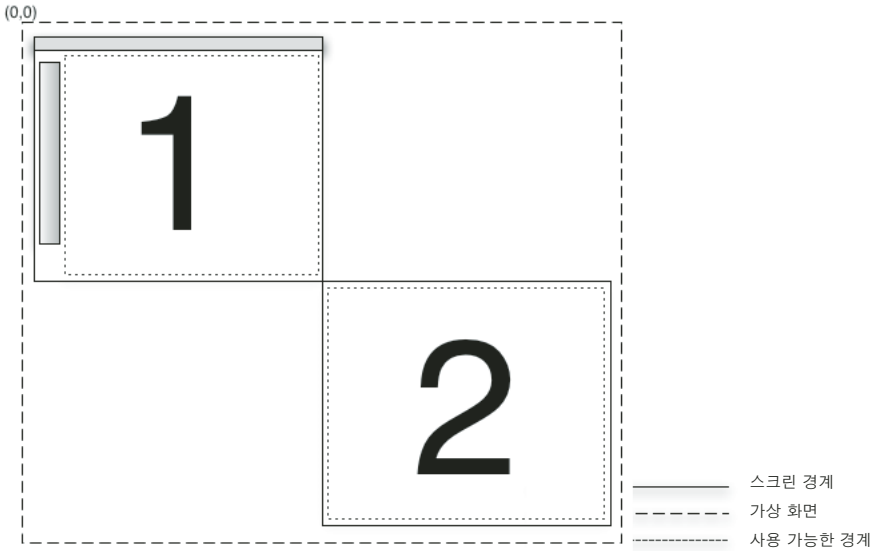
Adobe AIR 1.0 이상

- 가상 데스크톱 측정(Flex)
- 가상 데스크톱 측정(Flash)

스크린 API에는 Screen이라는 단일 클래스가 있으며 이는 시스템 스크린 정보를 가져오기 위한 정적 멤버와 특정 스크린을 기술하기 위한 인스턴스 멤버를 제공합니다.

하나의 컴퓨터 시스템은 여러 개의 모니터 또는 디스플레이와 연결될 수 있으며, 이는 가상 공간에 배치된 여러 개의 데스크톱 스크린에 대응합니다. AIR Screen 클래스는 스크린, 스크린의 상대적인 배치 및 사용 가능한 공간에 대한 정보를 제공합니다. 둘 이상의 모니터가 동일한 스크린에 매핑될 경우 단 하나의 스크린만 존재합니다. 스크린 크기가 모니터의 표시 영역보다 클 경우 현재 스크린의 어느 부분이 보이는지 확인할 방법이 없습니다.

스크린은 독립된 데스크톱 표시 영역을 나타냅니다. 스크린은 가상 데스크톱 내의 사각형으로 기술됩니다. 기본 표시로 지정된 화면의 왼쪽 위 모서리가 가상 데스크톱 좌표 시스템의 시작점입니다. 스크린을 기술하는 데 사용되는 모든 값은 픽셀 단위로 제공됩니다.



이 스크린 배치에서는 가상 데스크톱에 두 개의 스크린이 존재합니다. 기본 스크린(#1)의 왼쪽 위 모서리의 좌표는 항상 (0,0)입니다. 스크린 #2를 기본 스크린으로 지정하기 위해 스크린 배치를 변경할 경우 스크린 #1의 좌표는 음수가 됩니다. 화면에 사용할 수 있는 경계를 보고할 때 메뉴 모음, 작업 표시줄 및 도크는 제외됩니다.

화면 API 클래스, 메서드, 속성 및 이벤트에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

화면 열거

Adobe AIR 1.0 이상

다음과 같은 스크린 메서드 및 속성을 사용하여 가상 데스크톱의 스크린을 열거할 수 있습니다.

메서드 또는 속성	설명
Screen.screens	사용 가능한 스크린을 기술하는 Screen 객체의 배열을 제공합니다. 배열 순서는 중요하지 않습니다.
Screen.mainScreen	기본 스크린의 Screen 객체를 제공합니다. Mac OS X에서 기본 스크린은 메뉴 모음이 표시된 스크린입니다. Windows에서 기본 스크린은 시스템에서 지정한 기본 스크린입니다.
Screen.getScreensForRectangle()	주어진 사각형과 교차하는 스크린을 기술하는 Screen 객체의 배열을 제공합니다. 이 메서드에 전달되는 사각형은 가상 데스크톱에서의 픽셀 좌표로 표시됩니다. 스크린이 사각형과 교차하지 않을 경우 배열은 빕니다. 이 메서드를 사용하여 어떤 스크린에 윈도우가 표시될지 찾아낼 수 있습니다.

Screen 클래스 메서드 및 속성에 의해 반환되는 값은 저장하지 마십시오. 사용자 또는 운영 체제는 언제든지 사용 가능한 스크린 및 배치를 변경할 수 있습니다.

다음 예제에서는 스크린 API를 사용하여 화살표 키를 눌러 여러 화면 간에 윈도우를 이동하는 것을 보여 줍니다. 다음 스크린으로 윈도우를 이동하기 위해 이 예제에서는 screens 배열을 가져와서 이를 수직 또는 수평으로(누른 화살표 키에 따라) 정렬합니다. 이 코드는 정렬된 배열을 검토하여 각 화면을 현재 화면의 좌표를 비교합니다. 윈도우의 현재 스크린을 식별하기 위해 이 예제에서는 Screen.getScreensForRectangle()을 호출하여 윈도우 경계를 전달합니다.

```
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;
        }
    }
}
```

54장: 인쇄

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe® Flash® Player 및 Adobe® AIR™ 같은 Flash 런타임은 운영 체제의 인쇄 인터페이스와 통신할 수 있으므로 페이지를 인쇄 스풀러로 전달할 수 있습니다. 스풀러에 전송되는 각 페이지에는 데이터베이스 값 및 동적 텍스트를 비롯하여 사용자에게 보이거나, 동적이거나, 화면 밖에 표시되는 내용이 포함될 수 있습니다. 또한 사용자의 프린터 설정에 따라 flash.printing.PrintJob 클래스의 속성에 값이 포함되므로 이에 맞게 페이지 형식을 지정할 수 있습니다.

다음 내용에서는 flash.printing.PrintJob 클래스 메서드와 속성을 사용하여 인쇄 작업을 만들고, 사용자의 인쇄 설정을 읽고, Flash 런타임 및 사용자 운영 체제의 피드백에 따라 인쇄 작업을 조정하는 전략에 대해 설명합니다.

인쇄의 기초

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 PrintJob 클래스를 사용하여 표시 내용의 스냅샷을 만들고 출력에 잉크 용지 품질의 표현으로 변환합니다. 인쇄 내용 설정은 여러 요소를 배치하고 크기를 조정하며 원하는 레이아웃을 만드는 등 몇 가지 면에서 화면 표시를 설정하는 것과 동일합니다. 하지만 인쇄가 화면 레이아웃과 구별되는 몇 가지 특성이 있습니다. 예를 들어 프린터와 컴퓨터 모니터에서 사용되는 해상도가 다르고, 컴퓨터 화면 내용은 동적이고 변화 가능한 반면 인쇄 내용은 본질적으로 정적이며, 인쇄 계획 시에는 페이지 크기가 고정되어 있다는 제약 및 여러 페이지 인쇄 기능을 고려해야 합니다.

이러한 차이는 명백해 보이기는 하지만 ActionScript에서 인쇄 설정 시 반드시 유념해야 합니다. 정확한 인쇄를 위해서는 사용자가 지정하는 값과 사용자 프린터의 특성이 적절하게 조합되어야 하므로 PrintJob 클래스에는 사용자 프린터의 중요 특성을 사용자가 지정하도록 하는 속성이 포함됩니다.

중요한 개념 및 용어

다음 참조 목록에는 인쇄와 관련된 중요 용어가 정리되어 있습니다.

스풀러 인쇄 대기 중인 페이지를 추적하여 프린터가 사용 가능할 때 프린터로 작업을 전송하는 프린터 드라이버 소프트웨어 또는 운영 체제의 일부입니다.

페이지 방향 인쇄할 내용을 용지의 가로 또는 세로 방향으로 회전하는 것입니다.

인쇄 작업 1회의 출력을 구성하는 페이지 또는 페이지 집합입니다.

페이지 인쇄

Flash Player 9 이상, Adobe AIR 1.0 이상

인쇄 작업을 처리하려면 PrintJob 클래스의 인스턴스를 사용합니다. Flash Player 또는 AIR를 통해 기본 페이지를 인쇄하려면 다음 4개의 명령문을 차례로 사용하십시오.

- new PrintJob(): 지정한 이름의 새로운 인쇄 작업 인스턴스를 만듭니다.
- PrintJob.start(): 운영 체제의 인쇄 프로세스를 초기화하여 사용자에게 [인쇄] 대화 상자를 호출하고 인쇄 작업의 읽기 전용 속성이 선택되도록 합니다.

- PrintJob.addPage(): Sprite 객체 및 그 안에 포함된 자식, 인쇄 영역의 크기, 프린터에서 이미지를 벡터로 인쇄할지 또는 비트맵으로 인쇄할지 여부 등 인쇄 작업 내용에 대한 정보가 들어 있습니다. addPage()에 대해 연속 호출을 사용하여 여러 페이지에 걸쳐 여러 스프라이트를 인쇄할 수 있습니다.
- PrintJob.send(): 페이지를 운영 체제의 프린터로 전송합니다.

따라서, 컴파일을 위해 package, import 및 class 명령문을 포함하는 간단한 인쇄 작업 스크립트는 다음과 같습니다.

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

참고: 이 예는 인쇄 작업 스크립트의 기본 요소를 보여 주기 위한 것으로 오류 처리는 포함되어 있지 않습니다. 인쇄 작업을 취소하는 사용자에게 적절히 응답하는 스크립트를 작성하려면 840페이지의 “예외 및 반환 처리”를 참조하십시오.

특정 이유로 인해 PrintJob 객체의 속성을 제거하려는 경우 PrintJob 변수를 null로 설정하십시오(예: myPrintJob = null).

Flash 런타임 작업 및 시스템 인쇄

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash 런타임은 운영 체제의 인쇄 인터페이스에 페이지를 전달하므로 Flash 런타임에서 관리하는 작업과 운영 체제의 자체 인쇄 인터페이스에서 관리하는 작업에 주의해야 합니다. Flash 런타임은 인쇄 작업을 초기화하고, 프린터의 페이지 설정 일부를 읽고, 인쇄 작업의 내용을 운영 체제에 전달하고, 사용자나 시스템에서 인쇄 작업을 취소했는지 여부를 확인합니다. 프린터별 대화상자를 표시하거나, 스풀링된 인쇄 작업을 취소하거나, 프린터의 상태를 보고하는 등 그 밖의 프로세스는 모두 운영 체제에서 처리됩니다. Flash 런타임에서는 인쇄 작업을 초기화하거나 형식을 지정하는 데 문제가 있는 경우에 응답은 할 수 있지만 운영 체제의 인쇄 인터페이스에서 특정 속성이나 조건에 대해서만 보고할 수 있습니다. 개발자의 경우 코드에서 이러한 속성이나 조건에 응답해야 합니다.

예외 및 반환 처리

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 인쇄 작업을 취소한 경우 addPage() 및 send() 호출을 실행하기 전에 PrintJob.start() 메서드가 true를 반환하는지 확인하십시오. 계속 진행하기에 앞서 이러한 메서드가 취소되었는지 확인하는 간단한 방법은 if 문에서 해당 메서드를 다음과 같이 래핑하는 것입니다.

```
if (myPrintJob.start())
{
    // addPage() and send() statements here
}
```

PrintJob.start()가 true인 경우 사용자가 [인쇄]를 선택했거나 Flash Player나 AIR와 같은 Flash 런타임에서 인쇄 명령을 시작한 것입니다. 따라서 addPage() 및 send() 메서드를 호출할 수 있습니다.

또한 인쇄 프로세스를 관리할 수 있도록 Flash 런타임에서 PrintJob.addPage() 메서드에 대한 예외를 발생시키므로 오류를 catch하고 사용자에게 정보와 옵션을 제공할 수 있습니다. PrintJob.addPage() 메서드가 실패하면 다른 함수를 호출하거나 현재 인쇄 작업을 중지할 수도 있습니다. 다음 예제와 같이 addPage() 호출을 try..catch 문 내에 포함하여 이러한 예외를 catch합니다. 예제에서 [params]는 실제 인쇄할 내용을 지정하는 매개 변수의 자리 표시자입니다.

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error,
    }
    myPrintJob.send();
}
```

인쇄 작업이 시작되면 PrintJob.addPage()를 사용하여 내용을 추가할 수 있으며 그로 인해 예외가 생성되는지 여부(예: 사용자가 인쇄 작업을 취소했는지 여부)를 확인할 수 있습니다. 예외가 발생한 경우 catch 문에 논리를 추가하여 사용자 또는 Flash 런타임에 정보와 옵션을 제공하거나 현재 인쇄 작업을 중지할 수 있습니다. 페이지를 제대로 추가한 경우에는 계속해서 PrintJob.send()를 사용하여 페이지를 프린터로 전송할 수 있습니다.

Flash 런타임이 프린터에 인쇄 작업을 보내는 데 문제가 발생하는 경우, 예를 들어 프린터가 오프라인인 경우 이 예외도 catch하여 추가 정보나 추가 옵션(예: 메시지 텍스트 표시 또는 애니메이션 내에 경고 표시)을 제공할 수 있습니다. 예를 들어 다음 코드와 같이 if..else 문의 텍스트 필드에 새 텍스트를 지정할 수 있습니다.

```
if (myPrintJob.start())
{
    try
    {
        myPrintJob.addPage([params]);
    }
    catch (error:Error)
    {
        // Handle error.
    }
    myPrintJob.send();
}
else
{
    myAlert.text = "Print job canceled";
}
```

작업 예제는 848페이지의 “인쇄 예제: 배율 조절, 자르기 및 자동 맞춤”을 참조하십시오.

페이지 속성을 사용한 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자가 [인쇄] 대화 상자에서 [확인]을 클릭하고 PrintJob.start()가 true를 반환하면 프린터 설정에서 정의한 속성에 액세스할 수 있습니다. 이러한 설정에는 용지 폭, 용지 높이(pageHeight 및 pageWidth) 및 용지의 내용 방향 등이 있습니다. 이러한 속성은 프린터 설정이므로 Flash 런타임에서 조절할 수 없으며 이러한 설정은 변경할 수 없습니다. 그러나 이러한 속성을 사용하여 프린터로 전송할 내용을 현재 설정에 맞게 조정할 수 있습니다. 자세한 내용은 843페이지의 “크기, 배율 및 방향 설정”을 참조하십시오.

벡터 또는 비트맵 렌더링 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

각 페이지를 벡터 그래픽이나 비트맵 이미지로 스풀링하도록 인쇄 작업을 수동으로 설정할 수 있습니다. 경우에 따라 벡터 인쇄는 비트맵 인쇄보다 더 작은 스풀 파일을 만들고 더 나은 이미지를 만듭니다. 그러나 내용에 비트맵 이미지가 있는 경우 알파 투명도나 색상 효과를 보존하려면 페이지를 비트맵 이미지로 인쇄합니다. 또한 PostScript 프린터가 아닌 프린터에서는 자동으로 벡터 그래픽을 비트맵 이미지로 변환합니다.

PrintJobOptions 객체를 PrintJob.addPage()의 세 번째 매개 변수로 전달하여 비트맵 인쇄를 지정할 수 있습니다.

Flash Player 및 AIR 2 이전 버전의 경우 다음과 같이 PrintJobOptions 객체의 printAsBitmap 매개 변수를 true로 설정합니다.

```
var options:PrintJobOptions = new PrintJobOptions();
options.printAsBitmap = true;
myPrintJob.addPage(mySprite, null, options);
```

세 번째 매개 변수의 값을 지정하지 않으면 인쇄 작업에서 기본값(벡터 인쇄)을 사용합니다.

AIR 2 이상에서는 PrintJobOptions 객체의 printMethod 속성을 사용하여 인쇄 방법을 지정합니다. 이 속성은 세 가지 값을 사용합니다. 이러한 값은 PrintMethod 클래스에 상수로 정의되어 있습니다.

- PrintMethod.AUTO: 인쇄되는 내용에 따라 가장 적합한 인쇄 방법을 자동으로 선택합니다. 예를 들어 텍스트로 이루어진 페이지의 경우 벡터 인쇄 방법이 선택됩니다. 한편 알파 투명도가 있는 워터마크 이미지가 텍스트에 겹쳐 있는 경우 투명도를 유지하기 위해 비트맵 인쇄가 선택됩니다.
- PrintMethod.BITMAP: 내용에 관계없이 비트맵 인쇄를 적용합니다.
- PrintMethod.VECTOR: 내용에 관계없이 벡터 인쇄를 적용합니다.

인쇄 작업 시간 제한 명령문

Flash Player 9 이상, Adobe AIR 1.0 이상

ActionScript 3.0에서는 이전 버전의 ActionScript와 달리 PrintJob 객체를 단일 프레임으로 제한하지 않습니다. 그러나 사용자가 [인쇄] 대화 상자에서 [확인] 버튼을 클릭하면 운영 체제에서 사용자에게 인쇄 상태 정보를 표시하므로 PrintJob.addPage() 및 PrintJob.send()를 최대한 빠르게 호출하여 페이지를 스풀러로 보냅니다. PrintJob.send() 호출이 들어 있는 프레임에 늦게 도달하면 인쇄 과정이 지연될 수 있습니다.

ActionScript 3.0에서는 15초의 스크립트 시간 초과 제한이 있습니다. 그러므로 연속된 인쇄 작업에서 각 기본 명령문 사이의 시간은 15초를 초과할 수 없습니다. 즉, 다음 간격에 15초 스크립트 시간 초과 제한이 적용됩니다.

- PrintJob.start()와 첫 번째 PrintJob.addPage() 사이
- PrintJob.addPage()와 다음 PrintJob.addPage() 사이
- 마지막 PrintJob.addPage()와 PrintJob.send() 사이

이러한 간격 중 하나가 15초를 초과하는 경우 다음에 PrintJob 인스턴스에서 PrintJob.start()를 호출하면 false가 반환되고 다음에 PrintJob 인스턴스에서 PrintJob.addPage()를 반환하면 Flash Player 또는 AIR에서 런타임 예외가 발생합니다.

크기, 배율 및 방향 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

839페이지의 "페이지 인쇄" 단원에는 기본 인쇄 작업 단계가 자세히 설명되어 있습니다. 지정한 스프라이트의 화면 크기와 위치에 해당하는 인쇄 내용이 출력에 직접 반영됩니다. 그러나 프린터에서 인쇄할 때 다른 해상도를 사용하고, 인쇄된 스프라이트의 모양에 영향을 주는 설정이 있을 수 있습니다.

Flash 런타임은 운영 체제의 인쇄 설정을 읽을 수 있지만 이러한 속성은 읽기 전용이므로 인쇄 설정 값에 응답할 수는 있지만 설정할 수는 없습니다. 그러므로 예를 들어 프린터의 용지 크기 설정을 알아 보고 그 크기에 맞게 내용을 조정할 수 있습니다. 또한 프린터의 여백 설정과 페이지 방향도 확인할 수 있습니다. 프린터 설정에 응답하려면 인쇄 영역을 지정하거나, 화면 해상도와 프린터의 포인트 측정 단위 간에 차이를 조정하거나, 내용을 사용자 프린터의 크기나 방향 설정에 맞게 변형해야 합니다.

인쇄 영역에 사각형 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

PrintJob.addPage() 메서드를 사용하면 인쇄할 스프라이트의 영역을 지정할 수 있습니다. 두 번째 매개 변수인 printArea는 Rectangle 객체의 양식에 있습니다. 이 매개 변수에 값을 제공하는 옵션은 다음 세 가지입니다.

- 다음 예제와 같이 특정 속성이 있는 Rectangle 객체를 만든 다음 addPage() 호출에서 해당 사각형을 사용합니다.

```
private var rect1:Rectangle = new Rectangle(0, 0, 400, 200);
myPrintJob.addPage(sheet, rect1);
```

- 아직 Rectangle 객체를 지정하지 않은 경우 다음 예제와 같이 호출 자체 내에서 지정할 수 있습니다.

```
myPrintJob.addPage(sheet, new Rectangle(0, 0, 100, 100));
```

- addPage() 호출에서 세 번째 매개 변수의 값은 제공할 계획이지만 사각형을 지정하지 않으려면 다음과 같이 두 번째 매개 변수에 null을 사용할 수 있습니다.

```
myPrintJob.addPage(sheet, null, options);
```

포인트와 픽셀 비교

Flash Player 9 이상, Adobe AIR 1.0 이상

사각형의 폭과 높이는 픽셀 값입니다. 프린터에서는 인쇄 측정 단위로 포인트를 사용합니다. 포인트는 고정된 실제 크기(1/72인치)이지만 화면상 픽셀의 크기는 특정 화면의 해상도에 따라 다릅니다. 픽셀과 포인트 사이의 변환 비율은 프린터 설정 및 스프라이트의 배율 조절 여부에 따라 다릅니다. 폭이 72픽셀이고 배율이 조절되지 않은 Sprite는 폭이 1인치로 인쇄되며 이때 1포인트는 화면 해상도에 관계없이 1픽셀과 같습니다.

인치나 센티미터를 트윕 또는 포인트(1트윕은 1/20포인트)로 변환할 때 다음 등식을 사용할 수 있습니다.

- 1포인트 = 1/72인치 = 20트윕
- 1인치 = 72포인트 = 1440트윕
- 1센티미터 = 567트윕

printArea 매개 변수가 생략되거나 잘못 전달되면 스프라이트의 전체 영역이 인쇄됩니다.

크기 조절

Flash Player 9 이상, Adobe AIR 1.0 이상

인쇄하기 전에 Sprite 객체의 배율을 조절하려면 PrintJob.addPage() 메서드를 호출하기 전에 배율 속성(162페이지의 "객체 크기 조작 및 크기 조절" 참조)을 설정하고 인쇄 후 속성을 원래 값으로 다시 설정합니다. Sprite 객체의 배율은 printArea 속성과 관계가 없습니다. 즉, 인쇄 영역을 50픽셀 x 50픽셀로 지정하면 2500픽셀이 인쇄됩니다. Sprite 객체의 배율을 조절하면 동일하게 2500픽셀이 인쇄되지만 Sprite 객체는 배율이 조절된 크기로 인쇄됩니다.

예제는 848페이지의 "인쇄 예제: 배율 조절, 자르기 및 자동 맞춤"을 참조하십시오.

가로 또는 세로 방향으로 인쇄

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 AIR에서 방향 설정을 감지할 수 있으므로 다음 예제와 같이 ActionScript에 논리를 작성하여 프린터 설정에 맞게 현재 크기나 회전을 조절할 수 있습니다.

```
if (myPrintJob.orientation == PrintJobOrientation.LANDSCAPE)
{
    mySprite.rotation = 90;
}
```

참고: 용지에서 내용 방향에 대한 시스템 설정을 읽으려는 경우 PrintJobOrientation 클래스를 가져와야 합니다. PrintJobOrientation 클래스는 페이지에서 내용 방향을 정의하는 상수 값을 제공합니다. 이 클래스를 가져오려면 다음 명령문을 사용합니다.

```
import flash.printing.PrintJobOrientation;
```

페이지 높이와 폭에 자동 맞춤

Flash Player 9 이상, Adobe AIR 1.0 이상

프린터 방향 설정을 처리하는 것과 비슷한 방법을 사용하면 if 문에 일부 논리를 포함함으로써 페이지 높이와 폭 설정을 읽은 후 그에 맞게 조정할 수 있습니다. 다음은 예제 코드입니다.

```
if (mySprite.height > myPrintJob.pageHeight)
{
    mySprite.scaleY = .75;
}
```

뿐만 아니라 다음 예제와 같이 페이지와 용지의 크기를 비교하여 페이지의 여백 설정을 확인할 수 있습니다.

```
margin_height = (myPrintJob.paperHeight - myPrintJob.pageHeight) / 2;
margin_width = (myPrintJob.paperWidth - myPrintJob.pageWidth) / 2;
```

고급 인쇄 기법

Adobe AIR 2 이상

Adobe AIR 2부터 PrintJob 클래스에 속성 및 메서드가 추가되었습니다. 또한 PrintUIOptions, PaperSize, PrintMethod의 세 가지 클래스가 추가로 지원됩니다. 이러한 변경으로 추가 인쇄 작업 과정을 사용할 수 있으며 인쇄 프로세스에 대한 제작자의 제어력이 강화됩니다. 변경된 사항은 다음과 같습니다.

- 페이지 설정 대화 상자: 표준 및 사용자 정의 페이지 설정 대화 상자가 모두 표시될 수 있습니다. 사용자가 인쇄하기 전에 페이지 범위, 용지 크기, 방향 및 비율을 설정할 수 있습니다.
- 인쇄 보기: 용지 크기, 여백, 페이지 내용의 위치를 정확하게 보여 주는 보기 모드를 만들 수 있습니다.
- 제한 인쇄: 제작자가 인쇄 가능한 페이지 범위 등의 인쇄 옵션을 제한할 수 있습니다.
- 품질 옵션: 제작자가 문서의 인쇄 품질을 조정하고 사용자가 해상도 및 색상 옵션을 선택하도록 허용할 수 있습니다.
- 다중 인쇄 세션: 이제 여러 인쇄 세션에 대해 하나의 PrintJob 인스턴스를 사용할 수 있습니다. 따라서 페이지 설정 및 인쇄 대화 상자가 표시될 때마다 일관된 설정을 제공할 수 있습니다.

인쇄 작업 과정 변경 사항

새로운 인쇄 작업 과정은 다음과 같은 단계로 구성됩니다.

- new PrintJob(): PrintJob 인스턴스를 만들거나 기존 인스턴스를 다시 사용합니다. 인쇄 작업이 시작되기 전 또는 인쇄가 진행되는 동안 selectPaperSize()와 같은 여러 가지 새로운 PrintJob 속성 및 메서드를 사용할 수 있습니다.
- PrintJob.showPageSetupDialog(): (선택 사항) 인쇄 작업을 시작하지 않고 페이지 설정 대화 상자를 표시합니다.
- PrintJob.start() 또는 PrintJob.start2(): start() 메서드 외에 start2() 메서드를 사용하여 인쇄 스풀링 프로세스를 시작할 수 있습니다. start2() 메서드를 사용하면 [인쇄] 대화 상자를 표시할지 여부를 선택하고 표시되는 경우 대화 상자를 사용자 정의할 수 있습니다.
- PrintJob.addPage(): 인쇄 작업에 내용을 추가합니다. 기존 프로세스에서 변경된 사항이 없습니다.
- PrintJob.send() 또는 PrintJob.terminate(): 선택한 프린터로 페이지를 전송하거나, 페이지 전송 없이 인쇄 작업을 종료합니다. 오류가 발생할 경우 인쇄 작업이 종료됩니다. PrintJob 인스턴스가 종료되어도 이를 다시 사용할 수 있습니다. 인쇄 작업을 프린터로 보내든지 종료하든지 관계없이 PrintJob 인스턴스를 다시 사용할 때 현재 인쇄 설정이 유지됩니다.

페이지 설정 대화 상자

현재 환경에서 지원되는 경우 showPageSetupDialog() 메서드가 운영 체제의 페이지 설정 대화 상자를 표시합니다. 이 메서드를 호출하기 전에 항상 supportsPageSetupDialog 속성을 확인하십시오. 다음은 간단한 예입니다.

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
//check for static property supportsPageSetupDialog of PrintJob class
if (PrintJob.supportsPageSetupDialog) {
    myPrintJob.showPageSetupDialog();
}
```

PrintUIOptions 클래스 속성으로 메서드를 호출하여 페이지 설정 대화 상자에 표시될 옵션을 제어할 수 있습니다(선택 사항). 최소/최대 페이지 번호를 설정할 수 있습니다. 다음 예에서는 처음 3개 페이지로 인쇄를 제한합니다.

```
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);
}
```

인쇄 설정 변경

PrintJob 인스턴스는 생성 후 언제든지 해당 설정을 변경할 수 있습니다. 이 기능에는 addPage() 호출 사이에 설정을 변경하는 것과 인쇄 작업이 전송 또는 종료된 후 설정을 변경하는 것이 포함됩니다. printer 속성과 같은 일부 설정은 개별 페이지가 아닌 전체 인쇄 작업에 적용됩니다. 이러한 설정은 start() 또는 start2()를 호출하기 전에 지정해야 합니다.

selectPaperSize() 메서드를 호출하여 페이지 설정 대화 상자와 인쇄 대화 상자의 기본 용지 크기를 설정할 수 있습니다. 또한 인쇄 작업이 진행되는 동안 호출하여 일정 페이지 범위에 대해 용지 크기를 설정할 수도 있습니다. PaperSize 클래스에 정의된 상수를 사용하여 호출합니다. 아래 예에서는 봉투 크기 10을 선택합니다.

```
import flash.printing.PrintJob;
import flash.printing.PaperSize;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.selectPaperSize(PaperSize.ENV_10);
```

printer 속성을 사용하여 현재 인쇄 작업의 프린터 이름을 가져오거나 설정합니다. 기본적으로 기본 프린터 이름으로 설정됩니다. 사용할 수 있는 프린터가 없거나 시스템에서 인쇄를 지원하지 않는 경우에는 printer 속성 값이 null입니다. 프린터를 변경하려면 먼저 printers 속성을 사용하여 사용 가능한 프린터 목록을 가져옵니다. 이 속성은 Vector로서 해당 String 요소가 사용 가능한 프린터 이름입니다. 프린터를 활성 프린터로 설정하려면 printer 속성을 해당 String 값 중 하나로 설정합니다. 활성 인쇄 작업의 printer 속성은 변경할 수 없습니다. start() 또는 start2()를 성공적으로 호출한 후 작업이 전송되거나 종료되기 전에 이 속성을 변경하려고 하면 실패합니다. 다음은 이 속성을 설정하는 방법의 예입니다.

```
import flash.printing.PrintJob;

var myPrintJob:PrintJob = new PrintJob();
myPrintJob.printer = "HP_LaserJet_1";
myPrintJob.start();
```

copies 속성은 운영 체제의 인쇄 대화 상자에 설정된 매수 값을 가져옵니다. firstPage 및 lastPage 속성은 페이지 범위를 가져옵니다. orientation 속성은 페이지 방향 설정을 가져옵니다. 이러한 속성을 설정하여 인쇄 대화 상자의 값을 재정의할 수 있습니다. 다음 예에서는 이러한 속성을 설정합니다.

```
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;
```

PrintJob과 관련된 다음 읽기 전용 설정은 현재 프린터 설정에 대한 유용한 정보를 제공합니다.

- paperArea: 프린터 미디어의 사각형 경계(포인트 단위)입니다.
- printableArea: 인쇄 가능 영역의 사각형 경계(포인트 단위)입니다.
- maxPixelsPerInch: 현재 프린터의 실제 해상도(인치당 픽셀 수 단위)입니다.
- isColor: 현재 프린터가 컬러 인쇄를 할 수 있는지 여부입니다. 현재 프린터가 컬러 인쇄를 할 수 있는 경우 true를 반환합니다.

850페이지의 “인쇄 예제: 페이지 설정 및 인쇄 옵션”을 참조하십시오.

인쇄 예제: 여러 페이지 인쇄

Flash Player 9 이상, Adobe AIR 1.0 이상

두 페이지 이상의 내용을 인쇄할 때 각 페이지의 내용을 다른 스프라이트(이 예제에서는 sheet1과 sheet2)와 연결한 다음 각 스프라이트에 대해 PrintJob.addPage()를 사용할 수 있습니다. 다음 코드는 이 기술을 보여 줍니다.

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

인쇄 예제: 배율 조절, 자르기 및 자동 맞춤

Flash Player 9 이상, Adobe AIR 1.0 이상

화면에 나타나는 방식과 용지에 인쇄되어 나타나는 방식의 차이에 따라 인쇄할 때 표시 객체의 크기나 다른 속성을 조정해야 하는 경우가 있습니다. scaleX 및 scaleY 속성을 사용하는 등 인쇄하기 전에 표시 객체의 속성을 조절할 때 객체 크기가 인쇄 영역에 정의된 사각형보다 크면 객체가 잘립니다. 페이지를 인쇄한 후에는 속성을 원래대로 설정할 수도 있습니다.

다음 코드에서는 txt 표시 객체(녹색 상자 배경 제외)의 크기를 조절하고 지정된 사각형의 크기에 따라 텍스트 필드가 잘립니다. 인쇄한 후에는 텍스트 필드가 화면상에 표시되는 원래 크기로 돌아갑니다. 사용자가 운영 체제의 [인쇄] 대화 상자에서 인쇄 작업을 취소하면 Flash 런타임의 내용이 변경되어 작업을 취소한 사용자에게 경고를 표시합니다.

```
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;
        }
    }
}
```

인쇄 예제: 페이지 설정 및 인쇄 옵션

Adobe AIR 2 이상

다음 예에서는 매수, 용지 크기(Legal), 페이지 방향(가로)에 대한 PrintJob 설정을 초기화합니다. 먼저 페이지 설정 대화 상자를 강제로 표시한 다음 인쇄 대화 상자를 표시하여 인쇄 작업을 시작합니다.

```
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;
        }
    }
}
```

55장: Geolocation

장치가 지리적 위치 정보를 지원하면 지리적 위치 정보 API를 사용하여 장치의 현재 지리적 위치를 가져올 수 있습니다. 장치가 이 기능을 지원하는 경우 마지막 위치 변경 내용의 고도, 정확도, 방향, 속도 및 타임스탬프에 대한 정보를 가져올 수 있습니다.

Geolocation 클래스는 장치의 위치 센서에 대한 응답으로 update 이벤트를 전달합니다. update 이벤트는 GeolocationEvent 객체입니다.

기타 도움말 항목

flash.sensors.Geolocation

flash.events.GeolocationEvent

지리적 위치 정보 변경 감지

지리적 위치 센서를 사용하려면 Geolocation 객체를 인스턴스화하고 이 객체가 전달하는 update 이벤트에 대해 등록합니다. update 이벤트는 Geolocation 이벤트 객체로서 8가지 속성을 가집니다.

- altitude - 미터 단위의 고도입니다.
- heading - 정북 방향으로 이동한 각도입니다.
- horizontalAccuracy - 미터 단위의 수평 정확도입니다.
- latitude - 도 단위의 위도입니다.
- longitude - 도 단위의 경도입니다.
- speed - 미터/초 단위의 속도입니다.
- timestamp - 런타임이 초기화된 이후 이벤트가 발생하는 시점의 밀리초입니다.
- verticalAccuracy - 미터 단위의 수직 정확도입니다.

timestamp 속성은 int 객체이고 나머지는 Number 객체입니다.

다음은 텍스트 필드에 지리적 위치 데이터를 표시하는 기본적인 예제입니다.

```
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()
}
```

이 예제 코드를 사용하려면 먼저 geoTextField 텍스트 필드를 만들어 표시 목록에 추가해야 합니다.

Geolocation 객체의 setRequestedUpdateInterval() 메서드를 호출하여 지리적 위치 정보 이벤트의 원하는 시간 간격을 조정할 수 있습니다. 이 메서드는 interval 매개 변수 하나를 사용합니다. 이 매개 변수는 밀리초 단위의 요청된 업데이트 간격입니다.

```
var geo:Geolocation = new Geolocation();
geo.setRequestedUpdateInterval(10000);
```

지리적 위치 정보 업데이트의 실제 시간 간격은 이 값보다 크거나 작을 수 있습니다. 업데이트 간격을 변경하면 등록된 모든 리스너에 영향을 줍니다. setRequestedUpdateInterval() 메서드를 호출하지 않으면 응용 프로그램에서 장치의 기본 간격에 따라 업데이트를 받습니다.

사용자는 응용 프로그램에서 지리적 위치 데이터에 액세스하는 것을 막을 수 있습니다. 예를 들어 iPhone은 응용 프로그램에서 지리적 위치 데이터를 얻으려고 할 때 사용자에게 허용할 것인지 묻는 메시지를 표시합니다. 사용자는 해당 메시지에 대한 반응으로 응용 프로그램이 지리적 위치 데이터에 액세스하는 것을 거부할 수 있습니다. Geolocation 객체는 사용자가 지리적 위치 데이터를 사용할 수 없게 만드는 경우 status 이벤트를 전달합니다. 또한 지리적 위치 센서를 사용할 수 없게 될 때 Geolocation 객체의 muted 속성이 true로 설정됩니다. Geolocation 객체는 muted 속성이 변경될 때 status 이벤트를 전달합니다. 다음 코드에서는 지리적 위치 데이터를 사용할 수 없을 때 이를 감지하는 방법을 보여 줍니다.

```
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 = "";
        }
    }
}
```

참고: GPS 장치를 포함하지 않는 1세대 iPhone은 일부의 경우에만 update 이벤트를 전달합니다. 이러한 장치에서 Geolocation 객체는 처음에 update 이벤트를 한두 개 전달합니다. 그런 다음 정보가 눈에 띄게 변경되는 경우 update 이벤트를 전달합니다.

지리적 위치 정보 지원 확인

Geolocation.isSupported 속성을 통해 런타임 환경에서 이 기능을 사용할 수 있는지 여부를 테스트합니다.

```
if (Geolocation.isSupported)
{
    // Set up geolocation event listeners and code.
}
```

현재 지리적 위치 정보는 iPhone의 ActionScript 기반 응용 프로그램과 Flash Lite 4에서만 지원됩니다. Geolocation.isSupported가 런타임에 true 값을 반환하는 경우 지리적 위치 정보가 지원되는 것입니다.

일부 iPhone 모델에는 GPS 장치가 포함되어 있지 않습니다. 이러한 모델은 다른 수단(예: 휴대폰 삼각 측량)을 통해 지리적 위치 데이터를 얻습니다. 이러한 모델 또는 GPS가 지원되지 않는 iPhone의 경우 Geolocation 객체가 한두 개의 초기 update 이벤트만 가져올 수 있습니다.

56장: 응응 프로그램 국제화

Flash Player 10.1 이상, Adobe AIR 2.0 이상

flash.globalization 패키지를 사용하면 다양한 언어 및 국가/지역의 규칙에 부합하는 국제적인 소프트웨어를 더욱 쉽게 만들 수 있습니다.

기타 도움말 항목

flash.globalization 패키지

870페이지의 "응용 프로그램 지역화"

Charles Bihis: Flex/AIR 응용 프로그램을 지역화하려고 합니까?

응용 프로그램 국제화의 기초

세계화라는 용어와 국제화라는 용어는 경우에 따라 서로 통용될 수 있는 의미로 사용되기도 합니다. 그러나 두 용어의 가장 보편적인 정의에 따르면 세계화는 비즈니스 프로세스와 엔지니어링 프로세스의 결합을 가리키는 반면 국제화는 엔지니어링 프로세스만을 가리킵니다.

다음은 몇 가지 중요한 용어에 대한 정의입니다.

세계화 전 세계적으로 제품 및 회사 활동을 준비하고 실행하는 데 필요한 광범위한 엔지니어링 및 비즈니스 프로세스입니다. 세계화는 국제화, 지역화, 문화화 등의 엔지니어링 활동과 제품 관리, 재무 계획, 마케팅, 법률 관련 업무 등의 비즈니스 활동으로 구성됩니다. 세계화는 경우에 따라 **G11n**(첫 문자 G, 중간 11개 문자, 마지막 문자 N)으로 줄여 부르기도 합니다. **"세계화란 일련의 비즈니스 프로세스입니다."**

국제화 다시 설계하거나 다시 컴파일할 필요 없이 여러 언어, 스크립트 및 문화적 규칙(통화, 정렬 규칙, 숫자 및 날짜 서식 등)을 처리할 수 있도록 제품을 일반화하기 위한 엔지니어링 프로세스입니다. 이 프로세스는 실행 가능화(enablement)와 지역화의 두 가지 활동으로 나뉠 수 있습니다. 국제화는 경우에 따라 월드 레디니스(**world-readiness**) 또는 **I18n**으로 줄여 부르기도 합니다. **"국제화란 일련의 엔지니어링 프로세스입니다."**

지역화 특정 언어, 문화 및 해당 지역적 특성 맞춰 제품 또는 서비스를 조정하는 프로세스입니다. 지역화는 경우에 따라 **L10n**으로 줄여 부르기도 합니다. **"지역화란 일련의 번역 프로세스입니다."**

문화화 한 문화의 고유한 요구 사항에 맞춰 특정 기능을 개발하거나 조정하기 위한 엔지니어링 프로세스입니다. Adobe InDesign에서 사용할 수 있는 일본어 출판 기능과 Adobe Acrobat의 도장(Hanko) 지원 기능을 예로 들 수 있습니다.

기타 몇 가지 중요한 국제화 용어는 다음과 같이 정의할 수 있습니다.

문자 세트 하나의 언어 또는 언어 그룹에서 사용되는 문자입니다. 문자 세트에는 국가 문자, 특수 문자(예: 문장 부호, 수학 심볼), 숫자, 컴퓨터 제어 문자 등이 포함됩니다.

데이터 정렬 특정 로캘에 대해 올바른 순서로 텍스트를 정렬하는 것입니다.

로캘 하나의 지리적, 정치적 또는 문화적 지역(대부분의 경우 하나의 국가를 가리킴)에서 사용되는 언어 및 문화적 규칙을 반영하는 값입니다. 이 값은 고유한 로캘 식별자(로캘 ID)로 나타냅니다. 로캘 ID는 로캘별 지원을 제공하는 로캘 데이터 세트를 조회하는 데 사용됩니다. 이 지원은 측정 단위, 숫자/날짜 파싱 및 서식 지정 등에 적용됩니다.

리소스 번들 응용 프로그램이 사용될 로캘용으로 생성된 로캘별 요소를 모아 저장해 놓은 것입니다. 대개 응용 프로그램의 사용자 인터페이스에 있는 모든 텍스트 요소가 리소스 번들에 포함됩니다. 번들 내에서 이러한 요소는 지정된 로캘에 적합한 언어로 변환됩니다. 특정 로캘에 맞게 사용자 인터페이스의 레이아웃 또는 비헤이비어를 변경하는 기타 설정도 포함될 수 있습니다. 특정 로캘과 관련된 다른 미디어 유형이나 해당 미디어 유형에 대한 참조가 리소스 번들에 포함될 수 있습니다.

flash.globalization 패키지 개요

Flash Player 10.1 이상, Adobe AIR 2.0 이상

flash.globalization 패키지는 기본 운영 체제의 문화적 지원 기능을 활용합니다. 개별 사용자의 문화적 규칙을 따르는 응용 프로그램을 더욱 쉽게 작성할 수 있도록 해 줍니다.

이 패키지의 기본 클래스는 다음과 같습니다.

- 문자열의 정렬 및 일치를 제어하는 Collator 클래스
- 통화 금액 문자열로 숫자에 서식을 지정하고 입력 문자열의 통화 금액 및 심볼을 파싱하는 CurrencyFormatter 클래스
- 날짜 값의 서식을 지정하는 DateTimeFormatter 클래스
- 특정 로캘에 대한 정보를 검색하기 위한 LocaleID 클래스
- 숫자 값을 서식 지정 및 파싱하는 NumberFormatter 클래스
- 문자열의 로캘별 대/소문자 변환을 처리하는 StringTools 클래스

flash.globalization 패키지 및 리소스 지역화

flash.globalization 패키지는 리소스 지역화를 처리하지 않습니다. 그러나 flash.globalization 로캘 ID를 키 값으로 사용하여 다른 방법을 통해 지역화된 리소스를 검색할 수 있습니다. 예를 들어 ResourceManager 및 ResourceBundle 클래스를 사용하여 Flex로 만든 응용 프로그램 리소스를 지역화할 수 있습니다. 자세한 내용은 Flex 응용 프로그램 지역화를 참조하십시오.

또한 Adobe AIR 1.1에는 AIR 응용 프로그램 지역화에 도움이 되는 몇 가지 기능이 포함되어 있습니다. 이에 대해서는 871페이지의 “AIR 응용 프로그램 지역화”에서 설명합니다.

응용 프로그램을 국제화하는 일반적인 방식

다음 단계에서는 flash.globalization 패키지를 사용하여 응용 프로그램을 국제화하는 상위 수준의 일반적인 방식을 설명합니다.

1. 로캘을 결정 또는 설정합니다.
2. 서비스 클래스의 인스턴스(Collator, CurrencyFormatter, DateTimeFormatter, NumberFormatter 또는 StringTools)를 생성합니다.
3. lastOperationStatus 속성을 사용하여 오류 및 폴백을 확인합니다
4. 로캘별 설정을 사용하여 정보를 서식 지정 및 표시합니다.

다음 단계는 특정 로캘과 관련된 문자열 및 사용자 인터페이스 리소스를 로드하고 표시하는 것입니다. 이 단계에는 다음과 같은 작업이 포함될 수 있습니다.

- 자동 배치 기능을 사용하여 문자열 길이에 맞게 UI 크기조정
- 적절한 글꼴 선택 및 글꼴 대체 지원
- FTE 텍스트 엔진을 사용하여 다른 쓰기 시스템 지원
- IME(Input Method Editor)가 적절히 처리되는지 확인

오류 및 폴백 확인

flash.globalization 서비스 클래스는 모두 유사한 오류 식별 패턴을 따릅니다. 또한 사용할 수 없는 요청된 로캘에서 사용자의 운영 체제가 지원하는 로캘로의 폴백 패턴을 공유합니다.

다음 예제에서는 서비스 클래스를 인스턴스화할 때 오류 및 폴백을 확인하는 방법을 보여 줍니다. 각 서비스 클래스에는 최근 메서드 호출이 오류 또는 경고를 트리거했는지 여부를 나타내는 lastOperationStatus 속성이 있습니다.

```
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);
    }
}
```

이 예제는 폴백 로캘 ID가 사용된 경우 또는 오류가 있는 경우 메시지를 추적하는 것입니다. 필요한 경우 응용 프로그램에서 추가 오류 처리 논리를 수행할 수 있습니다. 예를 들어 사용자에게 특정 메시지를 표시하거나 응용 프로그램이 지원되는 특정 로캘을 사용하도록 할 수 있습니다.

로캘 결정

Flash Player 10.1 이상, Adobe AIR 2.0 이상

로캘은 해당 국가 또는 지역의 특정 언어 및 문화적 규칙 조합을 식별합니다.

로캘 식별자는 문자열로 관리하는 것이 안전합니다. 그러나 로캘과 관련된 추가 정보를 가져오도록 LocaleID 클래스를 사용할 수 있습니다.

LocaleID 객체를 만드는 방법은 다음과 같습니다.

```
var locale:LocaleID = new LocaleID("es-MX");
```

LocaleID 객체가 생성되었으면 로캘 ID에 대한 데이터를 검색할 수 있습니다. getKeysAndValues(), getLanguage(), getRegion(), getScript(), getVariant() 및 isRightToLeft() 메서드와 name 속성을 사용합니다.

이러한 메서드 및 속성으로 검색된 값은 해당 로캘에 대한 로캘 식별자를 통해 직접 추출할 수 없는 추가 정보를 나타낼 수 있습니다.

응용 프로그램에서 날짜 포맷터와 같은 로캘 인식 서비스를 만드는 경우에는 사용할 로캘을 지정해야 합니다. 지원되는 로캘 목록이 운영 체제마다 다르므로 요청된 로캘을 사용하지 못할 수 있습니다.

Flash Player는 먼저 요청된 로캘의 언어 코드와 일치시키려고 시도합니다. 그런 다음 일치하는 쓰기 시스템(스크립트) 및 지역을 검색하여 로캘을 세부적으로 지정합니다. 예를 들면 다음과 같습니다.

```
var loc:LocaleID = new LocaleID("es");
trace(loc.getLanguage()); // es
trace(loc.getScript()); // Latn
trace(loc.getRegion()); // ES
```

이 예제에서 LocaleID() 생성자는 사용자의 언어 코드 "es"와 가장 일치하는 로캘에 대한 정보를 검색합니다.

로캘 ID 설정

다음과 같은 여러 가지 방법으로 응용 프로그램의 현재 로캘을 설정할 수 있습니다.

- 단일 로캘 ID를 응용 프로그램에 하드 코딩합니다. 이것은 일반적인 방식이지만 응용 프로그램의 국제화를 지원하지 않습니다.
- 사용자의 운영 체제, 브라우저 또는 기타 사용자 환경 설정의 기본 로캘 ID를 사용합니다. 이렇게 하면 일반적으로 사용자에게 가장 적합한 로캘 설정이 구성되지만 정확하지 않은 경우도 있습니다. 운영 체제 설정이 사용자가 실제로 원하는 설정을 반영하지 않을 위험이 있습니다. 예를 들어 사용자가 공용 컴퓨터를 사용하므로 운영 체제의 기본 로캘을 변경하지 못할 수 있습니다.
- 사용자의 환경 설정에 따라 로캘 ID를 설정한 후 사용자가 지원되는 로캘 목록에서 선택할 수 있게 합니다. 이 전략은 대개 응용 프로그램에서 둘 이상의 로캘을 지원할 수 있는 경우에 가장 적합한 선택이 될 수 있습니다.

이 세 번째 옵션을 구현하는 방법은 다음과 같습니다.

1. 사용자 프로파일, 브라우저 설정, 운영 체제 설정 또는 쿠키에서 사용자의 기본 로캘 또는 언어 목록을 검색합니다. 이 경우 응용 프로그램 자체가 이 논리를 구현해야 합니다. flash.globalization 라이브러리로는 이러한 기본 설정을 직접 읽을 수 없습니다.
2. 응용 프로그램에서 지원하는 로캘을 결정하고 가장 적합한 로캘을 기본값으로 선택합니다. LocaleID.determinePreferredLocales() 메서드를 사용하여 사용자의 기본 로캘과 운영 체제에서 지원되는 로캘에 따라 가장 적합한 로캘을 찾습니다.
3. 기본 로캘이 만족스럽지 않은 경우 사용자가 기본 로캘 설정을 변경할 수 있는 방법을 제공합니다.

기타 로캘 및 언어 클래스의 제한 사항

fl.lang.Locale 클래스는 문자열 값을 포함한 리소스 번들을 사용하여 로캘을 기반으로 텍스트 문자열을 바꿀 수 있도록 합니다. 그러나 이 클래스는 숫자, 통화, 날짜 서식 지정, 정렬 및 일치 등의 기타 국제화 기능을 지원하지 않습니다. 또한 이 클래스는 Flash Professional에서만 사용할 수 있습니다.

flash.system.Capabilities.language 속성을 사용하여 운영 체제에 대한 현재 언어 코드 설정을 검색할 수도 있습니다. 그러나 이 속성은 전체 로캘 ID가 아닌 2문자 ISO 639-1 언어 코드만을 검색하고 특정 로캘 세트만을 지원합니다.

AIR 1.5에서 flash.system.Capabilities.languages 속성을 사용할 수 있습니다. 이 속성은 기본 사용자 인터페이스 언어 배열을 제공합니다. 따라서 Capabilities.language의 제한 사항이 없습니다.

숫자 서식 지정

Flash Player 10.1 이상, Adobe AIR 2.0 이상

숫자 값의 표시 서식은 지역에 따라 크게 달라집니다. 예를 들어 특정 로캘에서 숫자 123456.78의 서식이 지정되는 방법은 다음과 같습니다.

로캘	숫자 서식
en-US(영어, 미국)	-123,456.78
de-DE(독일어, 독일)	-123.456,78
fr-FR(프랑스어, 프랑스)	-123 456,78

로캘	숫자 서식
de-CH(독일어, 스위스)	-123'456.78
en-IN(영어, 인도)	-1,23,456.78
대부분의 아랍 로캘	123,456.78-

숫자 서식에 영향을 주는 요소는 다음과 같은 여러 가지가 있습니다.

- 구분 기호. 소수점 구분 기호는 정수와 숫자의 소수 부분 사이에 오며 마침표, 쉼표 또는 기타 문자가 될 수 있습니다. 그룹화 구분 기호 또는 천 단위 구분 기호는 마침표, 쉼표, 단어 잘림 방지 공백 또는 기타 문자가 될 수 있습니다.
- 그룹화 패턴. 소수점 왼쪽에 있는 각 그룹화 구분 기호 사이의 자릿수는 2-3개 또는 기타 값이 될 수 있습니다.
- 음수 표시. 음수는 숫자 왼쪽 또는 오른쪽에 빼기 기호로 표시되거나 재무 응용 프로그램의 경우 괄호 안에 표시될 수 있습니다. 예를 들어 음수 19는 -19, 19- 또는 (19)로 표시될 수 있습니다.
- 선행 및 후행 0. 일부 문화적 규칙에서는 선행 또는 후행 0을 표시되는 숫자로 추가합니다. 예를 들어 값 0.17은 .17, 0.17, 0.170 등으로 표시될 수 있습니다.
- 정수 문자 세트. 힌디어, 아랍어, 일본어를 비롯한 여러 언어는 서로 다른 정수 문자 세트를 사용합니다. flash.globalization 패키지는 숫자 0-9에 매핑되는 모든 정수 문자 세트를 지원합니다.

NumberFormatter 클래스는 숫자 값의 서식을 지정할 때 이러한 모든 요인을 고려합니다.

NumberFormatter 클래스 사용

NumberFormatter 클래스는 특정 로캘의 규칙에 따라 숫자 값(int, uint 또는 Number 유형) 서식을 지정합니다.

다음 예제에서는 사용자의 운영 체제에서 제공되는 기본 서식 지정 속성을 사용하여 숫자 서식을 지정하는 가장 간단한 방법을 보여 줍니다.

```
var nf:NumberFormatter = new NumberFormatter(LocaleID.DEFAULT);
trace(nf.formatNumber(-123456.789))
```

결과는 사용자의 로캘 설정 및 사용자 기본 설정에 따라 달라집니다. 예를 들어 사용자의 로캘이 fr-FR인 경우 값은 다음과 같이 서식 지정됩니다.

-123.456,789

사용자 설정에 관계없이 특정 로캘에 맞춰 숫자의 서식을 지정하려면 로캘 이름을 구체적으로 설정합니다. 예를 들면 다음과 같습니다.

```
var nf:NumberFormatter = new NumberFormatter("de-CH");
trace(nf.formatNumber(-123456.789));
```

이 경우 결과는 다음과 같습니다.

-123'456.789

formatNumber() 메서드는 매개 변수로 Number를 사용합니다. 또한 NumberFormatter 클래스는 입력 값으로 int를 사용하는 formatInt() 메서드와 unit를 사용하는 formatUint() 메서드를 가집니다.

다음 예와 같이 NumberFormatter 클래스의 속성을 설정하여 서식 지정 논리를 명시적으로 제어할 수 있습니다.

```
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)
```

이 예제에서는 먼저 NumberFormatter 객체를 만든 후 다음 작업을 수행합니다.

- 재무 응용 프로그램에서와 같이 괄호를 사용하도록 음수 서식을 설정합니다.
- 소수점 구분 기호 이후의 자릿수를 5로 설정합니다.
- 소수 자릿수를 항상 5자리로 유지하기 위해 후행 0을 사용하도록 지정합니다.
- 소수점 구분 기호를 쉼표로 설정합니다.
- 포맷터에서 그룹화 구분 기호를 사용하지 않도록 지정합니다.

참고: 이러한 속성 중 일부가 변경되면 결과 숫자 서식이 더 이상 지정된 로캘의 기본 서식과 일치하지 않게 됩니다. 로캘 인식이 중요하지 않은 경우, 서식 부분(예: 후행 0 수)을 세부적으로 제어해야 하는 경우 또는 Windows 제어판 등을 통해 사용자가 직접 변경을 요청하는 경우에만 이러한 속성 중 일부를 사용합니다.

숫자 값이 포함된 문자열 파싱

또한 NumberFormatter 클래스는 로캘별 서식 지정 요구 사항에 맞게 문자열에서 숫자 값을 추출할 수 있습니다. NumberFormatter.parseNumber() 메서드는 문자열에서 단일 숫자 값을 추출합니다. 예를 들면 다음과 같습니다.

```
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
```

parseNumber() 메서드는 음수 부호 및 구분 기호와 같은 자릿수 및 숫자 서식 지정 문자만 포함하는 문자열을 처리합니다. 문자열에 다른 문자가 포함된 경우 오류 코드가 설정됩니다.

```
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
```

영문자도 함께 포함된 문자열에서 숫자를 추출하려면 다음과 같이 NumberFormatter.parse() 메서드를 사용합니다.

```
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
```

parse() 메서드는 value 속성에 파싱된 숫자 값을 포함하는 NumberParseResult 객체를 반환합니다. startIndex 속성은 발견된 첫 번째 숫자의 인덱스를 나타냅니다. startIndex 및 endIndex 속성을 사용하여 숫자 앞과 뒤에 나오는 문자열 부분을 추출할 수 있습니다.

통화 값 서식 지정

Flash Player 10.1 이상, Adobe AIR 2.0 이상

통화 값 표시 서식도 숫자 서식만큼이나 다양합니다. 예를 들어 특정 로캘에서 미국 달러 값 $123456.78의 서식이 지정되는 방법은 다음과 같습니다.

로캘	숫자 서식
en-US(영어, 미국)	$123,456.78
de-DE(독일어, 독일)	123.456,78 $
en-IN(영어, 인도)	$ 1,23,456.78

통화 서식 지정에 영향을 주는 요소는 숫자 서식의 경우와 모두 동일하며 추가로 다음 요소가 적용됩니다.

- 통화 ISO 코드. 사용되는 실제 로캘에 대한 3자리 문자 ISO 4217 통화 코드입니다(예: USD 또는 EUR).
- 통화 심볼. 사용되는 실제 로캘에 대한 통화 심볼 또는 문자열입니다(예: $ 또는 €).
- 음수 통화 서식. 통화 금액의 숫자 부분과 관련하여 통화 심볼 및 음수 심볼 또는 괄호의 위치를 정의합니다.
- 양수 통화 서식 통화 금액의 숫자 부분과 관련하여 통화 심볼의 위치를 정의합니다.

CurrencyFormatter 클래스 사용

CurrencyFormatter 클래스는 특정 로캘 규칙에 따라 숫자 값을 통화 문자열 및 서식 지정된 숫자를 포함하는 문자열 형식으로 지정합니다.

새 CurrencyFormatter 객체를 인스턴스화하면 이 객체가 지정된 로캘에 대한 기본 통화로 통화를 설정합니다.

다음 예제에서는 독일어 로캘을 사용하여 생성된 CurrencyFormatter 객체가 통화 금액을 유로로 간주함을 보여 줍니다.

```
var cf:CurrencyFormatter = new CurrencyFormatter( "de-DE" );
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

대부분의 경우 로캘의 기본 통화에 의존하는 것은 바람직하지 않습니다. 사용자의 기본 로캘이 지원되지 않는 경우 CurrencyFormatter 클래스가 폴백 로캘을 할당하는데, 폴백 로캘의 기본 통화가 다를 수 있습니다. 또한 금액을 사용자의 현지 통화로 표시하지 않는 경우에도 일반적으로 통화 서식은 사용자에게 적합하게 나타내야 합니다. 예를 들어 캐나다 사용자가 독일 회사의 가격을 유로 단위로 보지만 캐나다에서 사용하는 통화 서식으로 표시하길 원할 수 있습니다.

CurrencyFormatter.setCurrency() 메서드는 사용할 정확한 통화 문자열과 통화 심볼을 지정합니다.

다음 예제에서는 캐나다의 프랑스어 사용 지역에서 통화 금액을 유로로 표시하는 경우를 보여 줍니다.

```
var cf:CurrencyFormatter = new CurrencyFormatter( "fr-CA" );
cf.setCurrency("EUR", "€");
trace(cf.format(1234567.89)); // 1.234.567,89 EUR
```

또한 setCurrency() 메서드를 사용하면 명확한 통화 심볼을 설정하여 혼동을 줄일 수 있습니다. 예를 들면 다음과 같습니다.

```
cf.setCurrency("USD","US$");
```

format() 메서드는 기본적으로 통화 심볼 대신에 3문자 ISO 4217 통화 코드를 표시합니다. ISO 4217 코드는 명확하며 지역화할 필요가 없습니다. 하지만 많은 사용자는 ISO 코드보다는 통화 심볼 표시를 선호합니다.

CurrencyFormatter 클래스는 서식이 지정된 통화 문자열에 어떤 통화 심볼(예: 달러 기호 또는 유로 기호)을 사용할지 또는 3문자 ISO 통화 문자열(예: USD 또는 EUR)을 사용할지를 결정하는 데 도움이 됩니다. 예를 들어 캐나다에 있는 사용자에게는 캐나다 달러 금액을 $200으로, 미국에 있는 사용자에게는 CAD 200으로 표시할 수 있습니다. formattingWithCurrencySymbolIsSafe() 메서드를 사용하면 해당 금액의 통화 심볼이 사용자의 로캘 설정에 따라 모호하거나 부정확한지 확인할 수 있습니다.

다음 예에서는 금액을 유로로 표시하지만 en-US 로캘에 적합한 서식으로 지정하는 방법을 보여 줍니다. 사용자의 로캘에 따라 ISO 통화 코드 또는 통화 심볼이 출력 문자열에 사용됩니다.

```
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
}
```

통화 값이 포함된 문자열 파싱

CurrencyFormatter 클래스는 로캘별 서식 지정 요구 사항에 맞게 입력 문자열에서 통화 금액과 통화 문자열을 추출할 수 있습니다. 다음과 같이 CurrencyFormatter.parse() 메서드는 파싱된 금액 및 통화 문자열을 CurrencyParseResult 객체에 저장합니다.

```
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
```

입력 문자열의 통화 문자열 부분에는 통화 심볼, 통화 ISO 코드 및 추가 텍스트 문자가 포함될 수 있습니다. 통화 문자열, 음수 표시 및 숫자 값의 위치는 negativeCurrencyFormat 및 positiveCurrencyFormat 속성으로 지정된 서식과 일치합니다. 예를 들면 다음과 같습니다.

```
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 $
```

이 예제에서는 입력 문자열에서 통화 문자열 다음에 빼기 부호와 숫자가 포함되어 있습니다. 그러나 en-US 로캘의 기본 negativeCurrencyFormat 값에 따라 음수 표시가 먼저 와야 하므로 parse() 메서드에서 오류가 발생하고 파싱된 값이 NaN이 됩니다.

통화 문자열이 먼저 오도록 negativeCurrencyFormat 값을 2로 설정하면 parse() 메서드가 성공합니다.

날짜 및 시간 서식 지정

Flash Player 10.1 이상, Adobe AIR 2.0 이상

날짜 및 시간 값의 표시 서식 역시 지역에 따라 크게 다릅니다. 예를 들어 특정 로캘에서 1962년 7월 2일 오후 1시 1분이 약식으로 표시되는 방법은 다음과 같습니다.

로캘	날짜 및 시간 서식
en-US(영어, 미국)	1/2/62 1:01pm
fr-FR(프랑스어, 프랑스)	2/1/62 13:01
ja-JP(일본, 일본어)	1962/2/1 13:01

DateTimeFormatter 클래스 사용

DateTimeFormatter 클래스는 특정 로캘 규칙에 따라 Date 값을 날짜 및 시간 문자열로 서식 지정합니다.

서식 지정은 날짜 또는 시간 값으로 대체되는 글자 시퀀스가 포함되어 있는 패턴 문자열을 따릅니다. 예를 들어 "yyyy/MM" 패턴에서는 "yyyy" 문자가 4자리 연도로 대체되고 그 뒤에 "/" 문자와 2자리 월이 차례로 표시됩니다.

패턴 문자열은 setDateTimePattern() 메서드를 사용하여 명시적으로 설정할 수 있습니다. 그러나 사용자의 로캘과 운영 체제 기본 설정에 따라 패턴을 자동으로 설정하는 것이 가장 좋습니다. 이렇게 하면 해당 문화에 적합한 결과를 얻는 데 도움이 됩니다.

DateTimeFormatter는 세 가지 표준 스타일(LONG, MEDIUM, SHORT)로 날짜 및 시간을 나타내고 CUSTOM 패턴을 사용할 수도 있습니다. 날짜와 시간에 서로 다른 스타일을 사용할 수 있습니다. 각 스타일에 사용되는 실제 패턴은 운영 체제에 따라 다소 달라집니다.

DateTimeFormatter 객체를 만들 때 스타일을 지정할 수 있습니다. 스타일 매개 변수를 지정하지 않으면 기본적으로 DateTimeStyle.LONG으로 설정되어 있습니다. 다음 예와 같이, 나중에 setDateTimeStyles() 메서드를 사용하여 스타일을 변경할 수 있습니다.

```
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
```

월 이름 및 요일 이름 지역화

많은 응용 프로그램에서 달력 표시에 월 이름 및 요일 이름 풀다운 목록을 사용합니다.

DateTimeFormatter.getMonthNames() 메서드를 사용하면 월 이름의 지역화된 목록을 가져올 수 있습니다. 운영 체제에 따라 정식 및 약식 목록을 얻을 수 있습니다. 정식 월 이름을 얻으려면 DateTimeNameStyle.FULL 값을 전달하고 약식 이름을 얻으려면 DateTimeNameStyle.LONG_ABBREVIATION 또는 DateTimeNameStyle.SHORT_ABBREVIATION 값을 전달합니다.

일부 언어에서는 월 이름이 날짜 서식에서 일 값 다음에 올 경우 소유격 형태로 바뀝니다. 월 이름을 단독으로 사용하려면 getMonthNames() 메서드에 DateTimeNameContext.STANDALONE 값을 전달합니다. 그러나 서식이 지정된 날짜에 월 이름을 사용하려면 DateTimeNameContext.FORMAT 값을 전달합니다.

```
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.
```

DateTimeFormatter.getWeekdayNames() 메서드는 요일 이름의 지역화된 목록을 제공합니다. getWeekdayNames() 메서드는 getMonthNames() 메서드와 동일한 nameStyle 및 context 매개 변수를 사용합니다.

```
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.
```

또한 getFirstWeekday() 메서드는 선택된 로캘에서 한 주의 시작을 나타내는 요일의 인덱스 값을 반환합니다.

문자열 정렬 및 비교

Flash Player 10.1 이상, Adobe AIR 2.0 이상

정렬은 적절한 순서로 항목을 배열하는 프로세스입니다. 정렬 규칙은 로캘에 따라 크게 달라집니다. 목록을 정렬하거나 텍스트 검색 알고리즘에서와 같이 유사한 항목을 일치시키는 경우에도 규칙이 달라집니다.

정렬 시에는 대/소문자, 분음 기호(예: 액센트) 등의 사소한 차이도 중요한 경우가 많습니다. 예를 들어 프랑스어나 영어에서는 ö(분음이 포함된 o)가 대부분 일반 글자 o에 해당하지만 스웨덴어에서는 그 뒤에 z 문자가 옵니다. 또한 프랑스어를 비롯한 일부 언어에서는 단어의 마지막 액센트 차이에 따라 정렬 목록에서의 순서가 결정됩니다.

검색 시에는 일치하는 값을 찾을 확률을 높이기 위해 대/소문자 또는 분음 기호 차이는 무시하는 경우가 많습니다. 예를 들어 프랑스어 문서에서 "cote"라는 문자를 검색하면 검색 결과에 "cote", "côte", "coté"가 포함됩니다.

Collator 클래스 사용

Collator 클래스의 기본 메서드는 주로 정렬에 사용되는 compare() 메서드와 값 일치에 사용되는 equals() 메서드입니다.

다음 예제에서는 compare() 메서드와 equals() 메서드의 서로 다른 비헤이비어를 보여 줍니다.

```
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
}
```

이 예제에서는 프랑스어-프랑스 로캘에 대해 먼저 SORTING 모드로 Collator 객체를 생성합니다. 그런 다음 분음 기호만 다른 두 단어를 정렬합니다. 이를 통해 SORTING 비교가 액센트 있는 문자와 없는 문자를 구별한다는 점을 알 수 있습니다.

정렬은 Collator 객체의 sort() 메서드에 대한 참조를 Array.sort() 메서드에 대한 매개 변수로 전달하여 수행됩니다. 이는 Collator 객체를 사용하여 정렬 순서를 제어하는 가장 효율적인 방식 중 하나입니다.

그 후에 이 예제에서는 MATCHING 모드로 Collator 객체를 만듭니다. 이때 Collator 객체는 두 단어를 비교하여 동일한 것으로 간주합니다. 이를 통해 MATCHING 비교는 액센트 있는 문자와 없는 문자를 동일한 값으로 간주한다는 점을 알 수 있습니다.

Collator 클래스의 비헤이비어 사용자 지정

기본적으로 Collator 클래스는 해당 로캘 및 사용자의 시스템 기본 설정에 따라 운영 체제에서 가져온 문자열 비교 규칙을 사용합니다. 다양한 속성을 명시적으로 설정하여 compare() 및 equals() 메서드의 비헤이비어를 사용자 지정할 수 있습니다. 다음 표에서는 다양한 Collator 속성과 해당 속성이 비교에 미치는 효과를 보여 줍니다.

Collator 속성	효과
numericComparison	숫자 문자를 숫자로 취급할지 텍스트로 취급할지 제어합니다.
ignoreCase	대문자와 소문자의 차이를 무시할지 여부를 제어합니다.
ignoreCharacterWidth	일부 중국어 및 일본어 문자의 전자와 반자 형태를 동일한 것으로 평가할지 여부를 제어합니다.
ignoreDiacritics	같은 기본 문자를 사용하지만 액센트 및 기타 분음 기호가 다른 문자열을 동일한 것으로 평가할지 여부를 제어합니다.
ignoreKanaType	사용 중인 가나 문자 형식만 다른 문자열을 동일한 것으로 취급할지 여부를 제어합니다.
ignoreSymbols	공백, 통화 심볼, 수학 심볼 및 기타 심볼 유형과 같은 심볼 문자를 무시할지 여부를 제어합니다.

다음 코드에서는 ignoreDiacritics 속성을 true로 설정하면 프랑스어 단어 목록의 정렬 순서가 바뀌는 것을 보여 줍니다.

```
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
```

대/소문자 변환

Flash Player 10.1 이상, Adobe AIR 2.0 이상

언어에 따라 대문자 형태와 소문자 형태 간의 문자 변환 규칙도 달라집니다.

예를 들어 라틴 알파벳을 사용하는 대부분의 언어에서는 대문자 "I"의 소문자 형태가 "i"입니다. 하지만 터키어 및 아제르바이잔어와 같은 일부 언어에서는 점이 없는 문자 "ı"가 추가로 있습니다. 따라서 이러한 언어에서 점이 없는 소문자 "ı"는 대문자 "I"로, 소문자 "i"는 점이 있는 대문자 "İ"로 변환됩니다.

StringTools 클래스는 언어별 규칙을 사용하여 이러한 변환을 수행하는 메서드를 제공합니다.

StringTools 클래스 사용

StringTools 클래스는 대/소문자 변환을 수행하는 두 가지 메서드, toLowerCase() 및 toUpperCase()를 제공합니다. StringTools 객체는 로캘 ID로 생성자를 호출하여 만듭니다. StringTools 클래스는 운영 체제에서 해당 로캘(또는 폴백 로캘)에 대한 대/소문자 변환 규칙을 검색합니다. 대/소문자 변환 알고리즘은 추가로 사용자 정의할 수 없습니다.

다음 예제에서는 toUpperCase() 및 toLowerCase() 메서드를 사용하여 문자 "ß"(샤프 S)를 포함하는 독일어 구절을 변환합니다.

```
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
```

toUpperCase() 메서드는 소문자 "ß"를 대문자 "SS"로 변환합니다. 이 변환은 한 방향으로만 작동합니다. 즉, 문자 "SS"를 다시 소문자로 변환한 결과는 "ß"가 아니라 "ss"입니다.

예제: 주식 시세 표시 응용 프로그램 국제화

Flash Player 10.1 이상, Adobe AIR 2.0 이상

Global Stock Ticker 응용 프로그램은 미국, 일본 및 유럽 주식 시장의 주식에 대한 가상 데이터를 검색 및 표시하며 다양한 로캘 규칙에 따라 데이터의 서식을 지정합니다.

이 예제에서는 flash.globalization 패키지의 다음 기능을 보여 줍니다.

- 로캘 인식 숫자 서식 지정
- 로캘 인식 통화 서식 지정
- 통화 ISO 코드 및 통화 기호 설정
- 로캘 인식 날짜 서식 지정
- 적절한 약식 월 이름 검색 및 표시

이 샘플에 대한 응용 프로그램 파일을 가져오려면 www.adobe.com/go/learn_programmingAS3samples_flash_kr을 참조하십시오. Global Stock Ticker 응용 프로그램 파일은 Samples/GlobalStockTicker 폴더에 있습니다. 이 응용 프로그램은 다음과 같은 파일로 구성됩니다.

파일	설명
GlobalStockTicker.mxml 또는 GlobalStockTicker.fla	Flex(MXML) 또는 Flash(FLA)용 응용 프로그램의 사용자 인터페이스입니다.
styles.css	응용 프로그램 사용자 인터페이스의 스타일입니다(Flex 전용).
com/example/programmingas3/stockticker/flex/FinGraph.mxml	시뮬레이션된 주식 데이터 차트를 표시하는 MXML 구성 요소입니다(Flex 전용).
com/example/programmingas3/stockticker/flash/GlobalStockTicker.as	이 응용 프로그램용 사용자 인터페이스 논리를 포함하는 문서 클래스입니다(Flash 전용).
comp/example/programmingas3/stockticker/flash/RightAlignedColumn.as	Flash DataGrid 구성 요소에 대한 사용자 지정 셀 렌더러입니다(Flash 전용).

파일	설명
com/example/programmingas3/stockticker/FinancialGraph.as	시뮬레이션된 주식 데이터 차트를 작성하는 ActionScript 클래스입니다.
com/example/programmingas3/stockticker/Localizer.as	로캘 및 통화를 관리하고 숫자, 통화 금액 및 날짜의 지역화된 서식 지정을 처리하는 ActionScript 클래스입니다.
com/example/programmingas3/stockticker/StockDataModel.as	Global Stock Ticker 예제의 모든 샘플 데이터를 저장하는 ActionScript 클래스입니다.

사용자 인터페이스 및 샘플 데이터 이해

이 응용 프로그램의 기본 사용자 인터페이스 요소는 다음과 같습니다.

- 로캘 선택을 위한 콤보 상자
- 시장 선택을 위한 콤보 상자
- 각 시장의 6개 회사에 대한 데이터를 표시하는 DataGrid 컨트롤
- 선택된 회사의 주식에 대한 시뮬레이션된 기간별 데이터를 보여주는 차트

이 응용 프로그램은 로캘, 시장 및 회사 주식에 대한 모든 샘플 데이터를StockDataModel 클래스에 저장합니다. 실제 응용 프로그램은 서버에서 데이터를 검색한 후 StockDataModel과 같은 클래스에 저장합니다. 이 예제에서는 모든 데이터가 StockDataModel 클래스에 하드 코딩되어 있습니다.

참고: 재무 차트에 표시되는 데이터는 DataGrid 컨트롤에 표시되는 데이터와 일치하지 않을 수 있습니다. 차트는 다른 회사를 선택할 때마다 임의로 다시 그려집니다. 이는 단지 예시용입니다.

로캘 설정

이 응용 프로그램은 일부 초기 설정 작업 후에 Localizer.setLocale() 메서드를 호출하여 기본 로캘에 대한 포맷터 객체를 생성합니다. 또한 setLocale() 메서드는 사용자가 Locale 콤보 상자에서 새 값을 선택할 때마다 호출됩니다.

```
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);
    }
}
```

setLocale() 메서드는 먼저 LocaleID 객체를 생성합니다. 이 객체를 통해 나중에 필요한 경우 실제 로캘에 대한 세부 사항을 쉽게 얻을 수 있습니다.

그런 다음 이 메서드는 해당 로캘에 대한 새 NumberFormatter, CurrencyFormatter 및 DateTimeFormatter 객체를 생성합니다. 각 포맷터 객체를 생성한 후 traceError() 메서드가 호출됩니다. 이 메서드는 요청된 로캘에 문제가 있는 경우 콘솔에 오류 및 경고 메시지를 표시합니다. 실제 응용 프로그램은 이러한 오류를 추적할 뿐만 아니라 적절히 반응해야 합니다.

CurrencyFormatter 객체를 생성한 후 setLocale() 메서드는 포맷터의 통화 ISO 코드, 통화 기호 및 fractionalDigits 속성을 사전 지정된 값으로 설정합니다. 이러한 값은 사용자가 Markets 콤보 상자에서 새 시장을 선택할 때마다 설정됩니다.

DateTimeFormatter 객체를 생성한 후 setLocale() 메서드는 지역화된 약식 월 이름의 배열을 가져옵니다.

데이터 서식 지정

서식이 지정된 주식 데이터는 DataGrid 컨트롤에 표시됩니다. DataGrid 열은 각각 적절한 포맷터 객체를 사용하여 열 값의 서식을 지정하는 레이블 함수를 호출합니다.

예를 들어 Flash 버전에서는 다음 코드가 DataGrid 열을 설정합니다.

```
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;
```

이 예제의 Flex 버전의 경우 MXML에 DataGrid를 선언합니다. 또한 각 열에 대해 유사한 레이블 함수를 정의합니다. labelFunction 속성은 Localizer 클래스의 서식 지정 메서드를 호출하는 다음 함수를 참조합니다.

```
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);
}
```

그런 다음 Localizer 메서드는 적절한 포맷터를 설정 및 호출합니다.

```
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);
}
|
```

57장: 응용 프로그램 지역화

Flash Player 9 이상, Adobe AIR 1.0 이상

지역화는 여러 로캘을 지원하기 위한 에셋을 포함하는 프로세스입니다. 로캘은 언어 및 국가 코드의 조합입니다. 예를 들어 en_US는 미국에서 사용되는 영어를 나타내고 fr_FR은 프랑스에서 사용되는 프랑스어를 나타냅니다. 이러한 로캘용으로 응용 프로그램을 지역화하려면 en_US 로캘용과 fr_FR 로캘용의 두 에셋 세트를 제공합니다.

로캘 간에 언어를 공유할 수 있습니다. 예를 들어 en_US 및 en_GB(영국)는 다른 로캘입니다. 이 경우 두 로캘은 모두 영어를 사용하지만 국가 코드는 이 둘이 서로 다른 로캘이고 따라서 다른 에셋을 사용할 수 있음을 나타냅니다. 예를 들어 en_US 로캘의 "color"라는 단어가 en_GB 로캘에서는 "colour"입니다. 또한 통화 단위가 로캘에 따라 달러 또는 파운드로 표시되고 날짜 및 시간 서식도 다를 수 있습니다.

국가 코드를 지정하지 않고 언어에 대한 에셋 세트를 제공할 수도 있습니다. 예를 들어 영어용 에셋을 제공하고 영어(미국)에 한정된 en_US 로캘용 추가 에셋을 제공할 수 있습니다.

지역화는 단순히 응용 프로그램에서 사용되는 문자열을 번역하는 것만을 의미하지 않습니다. 오디오 파일, 이미지 및 비디오를 비롯한 모든 유형의 에셋이 포함될 수도 있습니다.

로캘 선택

Flash Player 9 이상, Adobe AIR 1.0 이상

내용 또는 응용 프로그램에서 사용할 로캘을 결정하려면 다음 방법 중 하나를 사용할 수 있습니다.

- flash.globalization 패키지 - 운영 체제 및 사용자 기본 설정에 따라 기본 로캘을 검색하려면 flash.globalization 패키지의 로캘 인식 클래스를 사용합니다. Flash Player 10.1 이상 또는 AIR 2.0 이상 런타임에서 실행되는 응용 프로그램의 경우 이 방법이 권장됩니다. 자세한 내용은 857페이지의 “로캘 결정”을 참조하십시오.
- 사용자 프롬프트 - 응용 프로그램을 기본 로캘로 시작한 다음 사용자에게 원하는 기본 로캘을 선택하게 할 수 있습니다.
- (AIR에만 해당)Capabilities.languages - Capabilities.languages 속성은 운영 체제를 통해 설정된 사용자의 기본 언어에서 사용할 수 있는 언어의 배열을 나열합니다. 이 문자열에는 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 태그 및 해당되는 경우 스크립트 및 지역 정보가 포함됩니다. 이 문자열에서는 하이픈을 구분 기호로 사용합니다(예: "en-US" 또는 "ja-JP"). 반환된 배열의 첫 번째 항목은 language 속성과 기본 언어 ID가 동일합니다. 예를 들어 languages[0]을 "en-US"로 설정하면 language 속성이 "en"으로 설정됩니다. 그러나 language 속성을 알 수 없는 언어를 지정하는 "xu"로 설정하면 languages 배열의 첫 번째 요소가 달라집니다.
- Capabilities.language - Capabilities.language 속성은 운영 체제의 사용자 인터페이스 언어 코드를 제공합니다. 그러나 이 속성은 20개의 알려진 언어로 제한됩니다. 그리고 영어 시스템에서 이 속성은 국가 코드가 아닌 언어 코드만 반환합니다. 따라서 Capabilities.languages 배열의 첫 번째 요소를 사용하는 것이 좋습니다.

Flex 내용 지역화

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe Flex에는 Flex 내용을 지역화하기 위한 프레임워크가 포함되어 있습니다. 이 프레임워크에는 Locale, ResourceBundle 및 ResourceManagerImpl 클래스뿐 아니라 IResourceBundle 및 IResourceManagerImpl 인터페이스가 포함되어 있습니다 .

응용 프로그램 로캘 정렬용 유틸리티 클래스가 포함되어 있는 Flex 지역화 라이브러리는 Google 코드 (http://code.google.com/p/as3localelib/)에서 제공됩니다.

기타 도움말 항목

http://code.google.com/p/as3localelib/

Flash 내용 지역화

Flash Player 9 이상, Adobe AIR 1.0 이상

Adobe Flash Professional에는 ActionScript 3.0 구성 요소에 Locale 클래스가 포함되어 있습니다. Locale 클래스를 사용하면 SWF 파일에서 다국어 텍스트를 표시하는 방법을 제어할 수 있습니다. Flash [문자열] 패널에서는 동적 텍스트 필드에 문자열 리터럴 대신 문자열 ID를 사용할 수 있습니다. 따라서 언어별 XML 파일에서 로드된 텍스트를 표시하는 SWF 파일을 만들 수 있습니다. Locale 클래스 사용에 대한 자세한 내용은 Adobe Flash Platform용 ActionScript 3.0 참조 설명서를 참조하십시오.

AIR 응용 프로그램 지역화

Adobe AIR 1.0 이상

AIR SDK에서는 AIRLocalizer.js 파일에 포함된 HTML Localization Framework를 제공합니다. 이 프레임워크에는 HTML 기반 응용 프로그램에서 여러 로캘을 사용하여 작업하는 데 도움이 되는 API가 포함되어 있습니다. 로캘 정렬용 ActionScript 라이브러리는 http://code.google.com/p/as3localelib/에서 제공됩니다.

기타 도움말 항목

http://code.google.com/p/as3localelib/

날짜, 시간 및 통화 지역화

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램에서 날짜, 시간 및 통화를 표시하는 방법은 로캘마다 크게 다릅니다. 예를 들어 날짜를 나타내는 미국 표준은 월/일/년인 반면 유럽 표준은 일/월/년입니다.

코드를 작성하여 날짜, 시간 및 통화의 서식을 지정할 수 있습니다. 예를 들어 다음 코드에서는 Date 객체를 월/일/년 형식이나 일/월/년 서식으로 변환합니다. 로캘을 나타내는 locale 변수가 "en_US"로 설정된 경우 이 함수는 월/일/년 서식을 반환합니다. 다음 예제에서는 다른 모든 로캘에 대해 Date 객체를 일/월/년 서식으로 변환합니다.

```
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

Flex 프레임워크에는 날짜, 시간 및 통화의 서식을 지정하는 컨트롤이 포함되어 있습니다. DateFormatter 및 CurrencyFormatter 컨트롤을 예로 들 수 있습니다.

- mx:DateFormatter
- mx:CurrencyFormatter

58장: HTML 환경

Adobe AIR 1.0 이상

Adobe® AIR®에서는 Safari 웹 브라우저에서도 사용되는 WebKit(**www.webkit.org**)를 사용하여 HTML 및 JavaScript 내용을 파싱, 레이아웃 지정 및 렌더링합니다. HTML 내용에서 AIR API 사용은 선택 사항입니다. HTML 및 JavaScript만 사용하여 HTMLLoader 객체 또는 HTML 윈도우의 내용을 완전히 프로그래밍할 수 있습니다. 대부분의 기존 HTML 페이지 및 응용 프로그램은 WebKit와 호환되는 HTML, CSS, DOM 및 JavaScript 기능을 사용한다고 가정할 때 약간만 변경하여 실행할 수 있어야 합니다.

중요: Adobe AIR 런타임의 새 버전에는 업데이트된 버전의 WebKit가 포함되어 있습니다. 새 AIR 버전의 업데이트된 WebKit는 배포된 AIR 응용 프로그램에 **예기치 못한** 변경을 일으킬 수 있습니다. 이러한 변경은 응용 프로그램에서 HTML 내용의 동작이나 모양에 영향을 줄 수 있습니다. 예를 들어 WebKit 렌더링에서 향상되었거나 수정된 사항으로 인해 응용 프로그램의 사용자 인터페이스에 있는 요소의 레이아웃이 변경될 수 있습니다. 따라서 응용 프로그램에 업데이트 메커니즘을 추가하는 것이 좋습니다. AIR에 포함된 WebKit 버전이 변경됨으로 인해 응용 프로그램을 업데이트해야 하는 경우가 생기면 AIR 업데이트 메커니즘에서 사용자에게 새 버전의 응용 프로그램을 설치하라는 메시지를 표시할 수 있습니다.

다음 표에는 AIR에서 사용되는 것과 동일한 버전의 WebKit를 사용하는 Safari 웹 브라우저 버전이 나와 있습니다.

AIR 버전	Safari 버전
1.0	2.04
1.1	3.04
1.5	4.0 베타
2.0	4.03
2.5	4.03
2.6	4.03
2.7	4.03
3	5.0.3

다음과 같이 HTMLLoader 객체에 의해 반환되는 기본 사용자 에이전트 문자열을 확인하여 설치된 Webkit 버전을 파악할 수 있습니다.

```
var htmlLoader:HTMLLoader = new HTMLLoader();
trace( htmlLoader.userAgent );
```

AIR에 사용된 WebKit 버전은 오픈 소스 버전과 다르다는 점에 주의하십시오. 일부 기능은 AIR에서 지원되지 않으며 AIR 버전에는 해당 WebKit 버전에 아직 제공되지 않은 보안 및 버그 해결이 포함될 수 있습니다. 자세한 내용은 887페이지의 "AIR에서 지원되지 않는 WebKit 기능"을 참조하십시오.

AIR 응용 프로그램은 파일 시스템에 대한 전체 액세스 권한을 가지고 데스크톱에서 직접 실행되므로 HTML 내용에 대한 보안 모델이 일반 웹 브라우저의 보안 모델보다 엄격합니다. AIR에서는 응용 프로그램 설치 디렉토리에서 로드된 내용만 응용 프로그램 샌드박스에 저장됩니다. 응용 프로그램 샌드박스는 가장 높은 수준의 권한이 있으며 AIR API에 대한 액세스를 허용합니다. AIR에서는 해당 내용을 가져온 위치에 따라 격리된 샌드박스에 다른 내용을 저장합니다. 파일 시스템에서 로드된 파일은 로컬 샌드박스로 이동하고, 네트워크에서 http: 또는 https: 프로토콜을 사용하여 로드된 파일은 원격 서버의 도메인에 따라 다른 샌드박스로 이동합니다. 이러한 비 응용 프로그램 샌드박스의 내용은 AIR API에 액세스할 수 없으며 일반 웹 브라우저에서와 같이 실행됩니다.

AIR의 HTML 내용에서는 알파, 크기 조절 또는 투명도 설정이 적용되는 경우 SWF 또는 PDF 내용을 표시하지 않습니다. 자세한 내용은 913페이지의 "SWF 또는 PDF 내용을 HTML 페이지에 로드하는 경우의 고려 사항" 및 813페이지의 "윈도우 투명도"를 참조하십시오.

기타 도움말 항목

Webkit DOM 참조

Safari HTML 참조

Safari CSS 참조

www.webkit.org

HTML 환경 개요

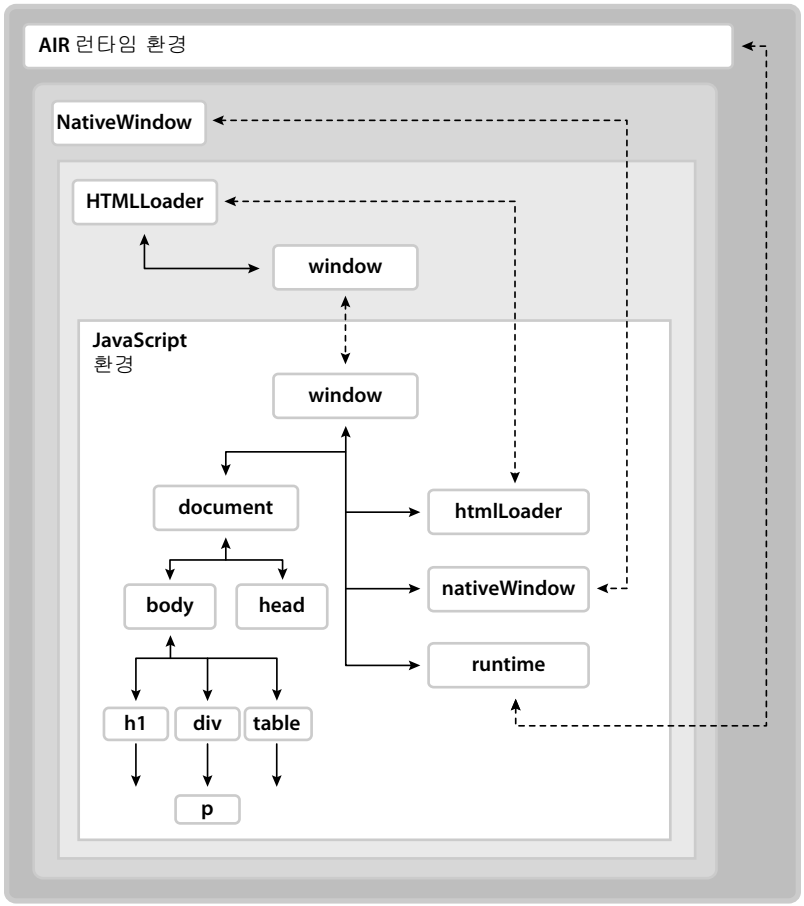
Adobe AIR 1.0 이상

Adobe AIR에서는 HTML 렌더러, 문서 객체 모델 및 JavaScript 인터프리터가 포함된 완전히 브라우저와 동일한 JavaScript 환경을 제공합니다. JavaScript 환경은 AIR HTMLLoader 클래스로 표현됩니다. HTML 윈도우에서 HTMLLoader 객체는 모든 HTML 내용을 포함하고 다시 이 객체는 NativeWindow 객체 내에 포함됩니다. SWF 내용에서는 Sprite 클래스를 확장하는 HTMLLoader 클래스를 다른 표시 객체처럼 스테이지 표시 목록에 추가할 수 있습니다. 해당 클래스의 ® ActionScript® 3.0 속성에 대한 자세한 내용은 912페이지의 “AIR HTML 컨테이너 스크립팅” 및 **Adobe Flash Platform용 ActionScript 3.0 참조 설명서**를 참조하십시오. Flex 프레임워크에서 AIR HTMLLoader 클래스는 mx:HTML 구성 요소에서 래핑됩니다. mx:HTML 구성 요소는 UIComponent 클래스를 확장하므로 다른 Flex 컨테이너에서 직접 사용할 수 있습니다. mx:HTML 구성 요소 내의 JavaScript 환경의 기타 부분은 동일합니다.

JavaScript 환경 및 AIR 호스트와의 관계

Adobe AIR 1.0 이상

다음 다이어그램에서는 JavaScript 환경과 AIR 런타임 환경 간의 관계를 보여 줍니다. 여기에는 하나의 기본 윈도우만 표시되어 있지만 AIR 응용 프로그램에는 여러 윈도우가 포함될 수 있습니다. 그리고 단일 윈도우에 여러 HTMLLoader 객체가 포함될 수 있습니다.



JavaScript 환경에는 고유의 Document 및 Window 객체가 있습니다. JavaScript 코드는 runtime, nativeWindow 및 htmlLoader 속성을 통해 AIR 런타임 환경과 상호 작용할 수 있습니다. ActionScript 코드는 JavaScript Window 객체를 참조하는 HTMLLoader 객체의 window 속성을 통해 **JavaScript** 환경과 상호 작용할 수 있습니다. 또한 **ActionScript** 및 **JavaScript** 객체 모두 **AIR** 및 **JavaScript** 객체를 통해 전달된 이벤트를 수신할 수 있습니다.

runtime 속성을 사용하여 AIR API 클래스에 액세스할 수 있으므로 새 AIR 객체 및 액세스 클래스(정적이라고도 함) 멤버를 만들 수 있습니다. AIR API에 액세스하려면 클래스 이름을 패키지와 함께 runtime 속성에 추가합니다. 예를 들어 File 객체를 만들려면 다음 명령문을 사용합니다.

```
var file = new window.runtime.filesystem.File();
```

참고: AIR SDK에서는 가장 일반적으로 사용되는 AIR 클래스에 대한 더욱 간편한 별칭을 정의하는 JavaScript 파일 AIRAliases.js를 제공합니다. 이 파일을 가져올 때 window.runtime.package.Class 대신 더 짧은 형식인 air.Class를 사용할 수 있습니다. 예를 들어 new air.File()을 사용하여 File 객체를 만들 수 있습니다.

NativeWindow 객체는 데스크톱 윈도우를 제어하기 위한 속성을 제공합니다. HTML 페이지 내에서 window.nativeWindow 속성을 사용하여 포함하는 NativeWindow 객체에 액세스할 수 있습니다.

HTMLLoader 객체는 내용을 로드하고 렌더링하는 방법을 제어하기 위한 속성, 메서드 및 이벤트를 제공합니다. HTML 페이지 내에서 window.htmlLoader 속성을 사용하여 부모 HTMLLoader 객체에 액세스할 수 있습니다.

중요: 응용 프로그램의 일부로 설치된 페이지에만 htmlLoader, nativeWindow 또는 runtime 속성이 있으며 최상위 문서로 로드된 경우에만 이러한 속성이 포함됩니다. 문서가 프레임 또는 iframe으로 로드되는 경우에는 이러한 속성이 추가되지 않습니다. 자식 문서는 동일한 보안 샌드박스에 있는 경우에만 부모 문서에 있는 이러한 속성에 액세스할 수 있습니다. 예를 들어 프레임에 로드된 문서는 parent.runtime을 사용하여 부모의 runtime 속성에 액세스할 수 있습니다.

보안

Adobe AIR 1.0 이상

AIR에서는 원래 도메인에 따라 다른 보안 샌드박스 내에서 모든 코드를 실행합니다. 응용 프로그램 설치 디렉토리에서 로드된 내용으로 제한되는 응용 프로그램 내용은 **응용 프로그램** 샌드박스에 저장됩니다. 이 샌드박스 내에서 실행되는 HTML 및 JavaScript에서만 런타임 환경 및 AIR API에 액세스할 수 있습니다. 동시에, 페이지 load 이벤트에 대한 핸들러가 모두 반환된 후 JavaScript에 대한 가장 동적인 평가 및 실행이 응용 프로그램 샌드박스에서 차단됩니다.

페이지를 프레임 또는 iframe으로 로드하고 프레임의 AIR 관련 sandboxRoot 및 documentRoot 특성을 설정하여 응용 프로그램 페이지를 비 응용 프로그램 샌드박스에 매핑할 수 있습니다. sandboxRoot 값을 실제 원격 도메인으로 설정하면 샌드박스 내용에서 해당 도메인의 내용을 크로스 스크립팅할 수 있습니다. **mash-up** 응용 프로그램에서처럼 원격 내용을 로드 및 스크립팅할 때 이와 같이 페이지를 매핑하면 유용합니다.

응용 프로그램 및 비 응용 프로그램 내용이 서로 크로스 스크립팅할 수 있도록 하는 또 다른 방법이면서 비 응용 프로그램 내용에서 AIR API에 액세스할 수 있게 하는 유일한 방법은 **샌드박스 브리지**를 만드는 것입니다. **부모 대 자식** 브리지를 사용하면 자식 프레임, iframe 또는 윈도우의 내용에서 응용 프로그램 샌드박스에 정의된 지정 메서드 및 속성에 액세스할 수 있습니다. 반대로 **자식 대 부모** 브리지를 사용하면 응용 프로그램 내용에서 자식의 샌드박스에 정의된 지정 메서드 및 속성에 액세스할 수 있습니다. 샌드박스 브리지는 window 객체의 parentSandboxBridge 및 childSandboxBridge 속성을 설정하여 지정합니다. 자세한 내용은 983페이지의 “Adobe AIR의 HTML 보안” 및 883페이지의 “HTML 프레임 및 iframe 요소”를 참조하십시오.

플러그인 및 포함된 객체

Adobe AIR 1.0 이상

AIR에서는 Adobe® Acrobat® 플러그인을 지원합니다. PDF 내용을 표시하려면 Acrobat 또는 Adobe® Reader® 8.1 이상이 있어야 합니다. HTMLLoader 객체에는 사용자 시스템에서 PDF를 표시할 수 있는지 여부를 확인하기 위한 속성이 있습니다. SWF 파일 내용도 HTML 환경 내에서 표시할 수 있지만 이 기능은 AIR에 내장되어 있으므로 외부 플러그인을 사용하지 않습니다.

다른 Webkit 플러그인은 AIR에서 지원되지 않습니다.

기타 도움말 항목

983페이지의 “Adobe AIR의 HTML 보안”

877페이지의 “HTML 샌드박스”

883페이지의 “HTML 프레임 및 iframe 요소”

882페이지의 “JavaScript Window 객체”

878페이지의 “XMLHttpRequest 객체”

501페이지의 “AIR에서 PDF 내용 추가”

AIR 및 WebKit

Adobe AIR 1.0 이상

Adobe AIR에서는 Safari 웹 브라우저에서도 사용되는 오픈 소스 WebKit 엔진을 사용합니다. AIR에서는 보안을 위해서 뿐만 아니라 런타임 클래스 및 객체에 액세스할 수 있도록 여러 확장을 추가합니다. 또한 Webkit 자체도 HTML, CSS 및 JavaScript에 대한 W3C 표준에 포함되어 있지 않은 기능을 추가합니다.

여기서는 AIR 추가 및 가장 주목할 만한 Webkit 확장에 대해서만 다룹니다. 비표준 HTML, CSS 및 JavaScript에 대한 추가 설명은 www.webkit.org 및 developer.apple.com을 참조하십시오. 표준에 대한 정보는 W3C 웹 사이트를 참조하십시오. Mozilla에서도 HTML, CSS 및 DOM 항목에 대한 유용한 일반 참조 설명서를 제공합니다. 물론 WebKit와 Mozilla 엔진은 동일하지 않습니다.

AIR의 JavaScript

Flash Player 9 이상, Adobe AIR 1.0 이상

AIR에서는 공통 JavaScript 객체의 일반 비헤이비어가 일부 변경되었습니다. 이러한 변경 사항은 대부분 AIR에서 보안 응용 프로그램을 더 쉽게 작성하기 위한 것입니다. 동시에, 비헤이비어에서의 이러한 차이는 일부 공통 JavaScript 코딩 패턴과 해당 패턴을 사용하는 기존 웹 응용 프로그램이 AIR에서 예상한 대로 실행되지 않을 수도 있다는 것을 나타냅니다. 이러한 유형의 문제를 수정하는 방법에 대한 자세한 내용은 894페이지의 "보안 관련 JavaScript 오류 방지"를 참조하십시오.

HTML 샌드박스

Adobe AIR 1.0 이상

AIR에서는 내용의 원본에 따라 격리된 샌드박스에 내용을 저장합니다. 샌드박스 규칙은 Adobe Flash Player에서 구현되는 샌드박스에 대한 규칙뿐 아니라 대부분의 웹 브라우저에서 구현되는 같은 원본의 정책과 일치합니다. 또한, AIR에서는 응용 프로그램 내용을 포함하고 보호하기 위해 새로운 **응용 프로그램** 샌드박스 유형을 제공합니다. AIR 응용 프로그램을 개발할 때 다루게 되는 샌드박스 유형에 대한 자세한 내용은 950페이지의 "보안 샌드박스"를 참조하십시오.

응용 프로그램 샌드박스 내에서 실행되는 HTML 및 JavaScript에서만 런타임 환경 및 AIR API에 액세스할 수 있습니다. 그러나 이와 동시에 다양한 형식의 JavaScript에 대한 동적 평가 및 실행은 보안상의 이유로 응용 프로그램 샌드박스 내에서 대부분 제한됩니다. 이러한 제한은 실제로 응용 프로그램이 서버에서 직접 정보를 로드하는지 여부와 상관없이 적용됩니다. 파일 내용, 붙여넣은 문자열 및 사용자가 직접 입력한 내용도 신뢰할 수 없을 수 있습니다.

페이지에 있는 내용의 원본에 따라 내용이 전달되는 샌드박스가 달라집니다. 응용 프로그램 디렉토리(app: URL 스킴으로 참조되는 설치 디렉토리)에서 로드된 내용만 응용 프로그램 샌드박스에 저장됩니다. 파일 시스템에서 로드된 내용은 **local-with-file system** 또는 **local-trusted** 샌드박스에 저장됩니다. 이 샌드박스에서는 로컬 파일 시스템의 내용에 액세스하고 상호 작용할 수는 있지만 원격 내용에는 액세스하거나 상호 작용할 수 없습니다. 네트워크에서 로드된 내용은 원래 도메인에 해당하는 원격 샌드박스에 저장됩니다.

응용 프로그램 페이지에서 원격 샌드박스의 내용과 자유롭게 상호 작용할 수 있게 하려면 페이지를 원격 내용과 동일한 도메인에 매핑할 수 있습니다. 예를 들어 인터넷 서비스에서 맵 데이터를 표시하는 응용 프로그램을 작성하는 경우 서비스의 내용을 로드 및 표시하는 응용 프로그램 페이지를 서비스 도메인에 매핑할 수 있습니다. 원격 샌드박스 및 도메인에 페이지를 매핑하기 위한 특성은 프레임 및 iframe HTML 요소에 추가된 새로운 특성입니다.

비 응용 프로그램 샌드박스의 내용에서 AIR 기능을 안전하게 사용할 수 있게 하려면 부모 샌드박스 브리지를 설정할 수 있습니다. 응용 프로그램 내용에서 안전하게 다른 샌드박스의 내용에 대한 메서드를 호출하고 속성에 액세스할 수 있게 하려면 자식 샌드박스 브리지를 설정할 수 있습니다. 여기서 안전이란 명시적으로 표시되지 않은 객체, 속성 또는 메서드에 대한 참조를 원격 내용에서 실수로 가져올 수 없음을 의미합니다. 간단한 데이터 유형, 함수 및 익명 객체만 브리지를 통해 전달될 수 있습니다. 그러나, 잠재적으로 위험한 함수는 계속 명시적으로 표시되지 않아야 합니다. 예를 들어 원격 내용이 사용자 시스템의 임의의 위치에 있는 파일을 읽고 쓰도록 허용한 인터페이스를 노출한 경우 원격 내용으로 인해 사용자가 상당한 피해를 입을 수 있습니다.

JavaScript eval() 함수

Adobe AIR 1.0 이상

페이지 로드가 완료되면 응용 프로그램 샌드박스 내에서 eval() 함수의 사용이 제한됩니다. JSON 형식 데이터를 안전하게 파싱할 수 있도록 몇 가지 사용 방법이 허용되지만 실행 가능한 명령문을 생성하는 평가에서는 오류가 발생합니다. 985페이지의 "서로 다른 샌드박스의 내용에 대한 코드 제한 사항"에서는 eval() 함수의 허용되는 사용 방법에 대해 설명합니다.

함수 생성자

Adobe AIR 1.0 이상

응용 프로그램 샌드박스에서 페이지 로드가 완료되기 전에 함수 생성자를 사용할 수 있습니다. 모든 페이지 load 이벤트 핸들러가 완료된 후에는 새 함수를 만들 수 없습니다.

외부 스크립트 로드

Adobe AIR 1.0 이상

응용 프로그램 샌드박스의 HTML 페이지에서는 스크립트 태그를 사용하여 응용 프로그램 디렉토리 외부에서 JavaScript 파일을 로드할 수 없습니다. 응용 프로그램의 페이지에서 응용 프로그램 디렉토리 외부의 스크립트를 로드하려면 페이지를 비 응용 프로그램 샌드박스에 매핑해야 합니다.

XMLHttpRequest 객체

Adobe AIR 1.0 이상

AIR에서는 응용 프로그램이 데이터를 요청할 때 사용할 수 있는 XMLHttpRequest(XHR) 객체를 제공합니다. 다음 예에서는 간단한 데이터 요청을 보여 줍니다.

```
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);
```

브라우저와 달리 AIR에서는 응용 프로그램 샌드박스에서 실행 중인 내용이 모든 도메인의 데이터를 요청할 수 있습니다. JSON 문자열이 포함된 XHR의 결과는 결과에 실행 가능한 코드도 포함된 경우 외에는 데이터 객체로 평가될 수 있습니다. XHR 결과에 실행 가능한 명령문이 있는 경우 오류가 발생하고 평가 시도가 실패합니다.

원격 소스에서 코드가 실수로 삽입되지 않도록 하기 위해 동기식 XHR은 페이지 로드가 완료되기 전에 만들어진 경우 빈 결과를 반환합니다. 비동기 XHR은 항상 페이지가 로드된 후 반환됩니다.

기본적으로 AIR은 비 응용 프로그램 샌드박스에서 크로스 도메인 XMLHttpRequests를 차단합니다. 응용 프로그램 샌드박스의 부모 윈도우에서는 다음과 같이 포함하는 프레임 또는 iframe 요소에서 AIR에 추가된 특성인 allowCrossDomainXHR을 true로 설정하여 비 응용 프로그램 샌드박스의 내용을 포함하는 자식 프레임에서 크로스 도메인 요청을 허용할 수 있습니다.

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    allowCrossDomainXHR="true"
</iframe>
```

참고: 편리하다면 AIR URLStream 클래스를 사용하여 데이터를 다운로드할 수도 있습니다.

원격 샌드박스에 매핑된 응용 프로그램 내용을 포함하는 프레임 또는 iframe 요소에서 원격 서버에 XMLHttpRequest를 전달하는 경우 매핑 URL이 XHR에서 사용된 서버 주소에 마스크를 적용하지 않는지 확인합니다. 예를 들어 응용 프로그램 내용을 example.com 도메인의 원격 샌드박스에 매핑하는 다음과 같은 iframe 정의를 생각해 볼 수 있습니다.

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/"
    allowCrossDomainXHR="true"
</iframe>
```

sandboxRoot 특성은 www.example.com 주소의 루트 URL을 다시 매핑하므로 모든 요청이 원격 서버가 아닌 응용 프로그램 디렉토리에서 로드됩니다. 요청이 페이지 탐색에서 파생되는지 또는 XMLHttpRequest에서 파생되는지 여부에 상관없이 요청이 다시 매핑됩니다.

원격 서버에 대한 데이터 요청을 실수로 차단하지 않게 하려면 sandboxRoot를 루트가 아닌 원격 URL의 하위 디렉토리에 매핑합니다. 이 디렉토리는 존재하지 않아도 됩니다. 예를 들어 www.example.com에 대한 요청이 응용 프로그램 디렉토리가 아닌 원격 서버에서 로드되도록 하려면 앞서 설명한 iframe을 다음과 같이 변경합니다.

```
<iframe id="mashup"
    src="http://www.example.com/map.html"
    documentRoot="app:/sandbox/"
    sandboxRoot="http://www.example.com/air/"
    allowCrossDomainXHR="true"
</iframe>
```

이 경우 air 하위 디렉토리의 내용만 로컬로 로드됩니다.

샌드박스 매핑에 대한 자세한 내용은 883페이지의 “HTML 프레임 및 iframe 요소” 및 983페이지의 “Adobe AIR의 HTML 보안”을 참조하십시오.

쿠키

Adobe AIR 1.0 이상

AIR 응용 프로그램에서는 원격 샌드박스의 내용(http: 및 https: 소스에서 로드된 내용)에서만 쿠키(document.cookie 속성)를 사용할 수 있습니다. 응용 프로그램 샌드박스에서는 EncryptedLocalStore, SharedObject, FileStream 클래스와 같이, 영구 데이터를 저장하는 데 사용할 수 있는 다른 방법들이 지원됩니다.

Clipboard 객체

Adobe AIR 1.0 이상

WebKit Clipboard API는 copy, cut 및 paste 이벤트에 의해 작동됩니다. 이러한 이벤트에서 전달되는 이벤트 객체의 clipboardData 속성을 통해 클립보드에 액세스할 수 있습니다. 클립보드 데이터를 읽거나 쓰려면 clipboardData 객체의 다음 메서드를 사용합니다.

메서드	설명
clearData(mimeType)	클립보드 데이터를 지웁니다. mimeType 매개 변수를 지울 데이터의 MIME 유형으로 설정합니다.
getData(mimeType)	클립보드 데이터를 가져옵니다. 이 메서드는 paste 이벤트의 핸들러에서만 호출할 수 있습니다. mimeType 매개 변수를 반환할 데이터의 MIME 유형으로 설정합니다.
setData(mimeType, data)	데이터를 클립보드에 복사합니다. mimeType 매개 변수를 데이터의 MIME 유형으로 설정합니다.

응용 프로그램 샌드박스 외부의 JavaScript 코드에서만 이러한 이벤트를 통해 클립보드에 액세스할 수 있습니다. 그러나 응용 프로그램 샌드박스의 내용에서는 AIR Clipboard 클래스를 사용하여 시스템 클립보드에 직접 액세스할 수 있습니다. 예를 들어 다음 명령문을 사용하여 클립보드의 텍스트 서식 데이터를 가져올 수 있습니다.

```
var clipping = air.Clipboard.generalClipboard.getData("text/plain",
                                    air.ClipboardTransferMode.ORIGINAL_ONLY);
```

유효한 데이터 MIME 유형은 다음과 같습니다.

MIME 유형	값
텍스트	"text/plain"
HTML	"text/html"
URL	"text/uri-list"
비트맵	"image/x-vnd.adobe.air.bitmap"
파일 목록	"application/x-vnd.adobe.air.file-list"

중요: 응용 프로그램 샌드박스의 내용에서만 클립보드에 있는 파일 데이터에 액세스할 수 있습니다. 비 응용 프로그램 내용에서 클립보드의 File 객체에 액세스하려고 하면 보안 오류가 발생합니다.

클립보드 사용에 대한 자세한 내용은 542페이지의 "복사하여 붙여넣기" 및 JavaScript에서 임시 보드 사용(Apple 개발자 센터)을 참조하십시오.

드래그 앤 드롭

Adobe AIR 1.0 이상

HTML 내부 및 외부로의 드래그 앤 드롭 동작은 dragstart, drag, dragend, dragenter, dragover, dragleave 및 drop과 같은 DOM 이벤트를 발생시킵니다. 이러한 이벤트에서 전달되는 이벤트 객체의 dataTransfer 속성을 통해 드래그한 데이터에 액세스할 수 있습니다. dataTransfer 속성은 클립보드 이벤트와 관련된 clipboardData 객체와 동일한 메서드를 제공하는 객체를 참조합니다. 예를 들어 다음 함수를 사용하여 drop 이벤트에서 텍스트 서식 데이터를 가져올 수 있습니다.

```
function onDrop(dragEvent){
    return dragEvent.dataTransfer.getData("text/plain",
            air.ClipboardTransferMode.ORIGINAL_ONLY);
}
```

dataTransfer 객체에는 다음과 같은 중요한 멤버가 있습니다.

멤버	설명
clearData(mimeType)	데이터를 지웁니다. mimeType 매개 변수를 지울 데이터 표현의 MIME 유형으로 설정합니다.
getData(mimeType)	드래그한 데이터를 가져옵니다. 이 메서드는 drop 이벤트의 핸들러에서만 호출할 수 있습니다. mimeType 매개 변수를 가져올 데이터의 MIME 유형으로 설정합니다.
setData(mimeType, data)	드래그할 데이터를 설정합니다. mimeType 매개 변수를 데이터의 MIME 유형으로 설정합니다.
types	현재 dataTransfer 객체에서 사용할 수 있는 모든 데이터 표현의 MIME 유형을 포함하는 문자열의 배열입니다.
effectsAllowed	드래그되는 데이터를 복사, 이동, 연결할 수 있는지 여부 또는 일부 관련 조합을 지정합니다. dragstart 이벤트의 핸들러에서 effectsAllowed 속성을 설정합니다.
dropEffect	허용되는 드롭 효과 중 드래그 대상에서 지원되는 효과를 지정합니다. dragEnter 이벤트의 핸들러에서 dropEffect 속성을 설정합니다. 드래그하는 동안 사용자가 마우스를 놓을 경우 발생할 효과를 표시하기 위해 커서가 변경됩니다. dropEffect를 지정하지 않은 경우 effectsAllowed 속성 효과가 선택됩니다. 복사 효과가 이동 효과보다 우선하고 이동 효과는 연결 효과보다 우선합니다. 사용자가 키보드를 사용하여 기본 우선 순위를 수정할 수 있습니다.

AIR 응용 프로그램으로 드래그 앤 드롭에 대한 지원을 추가하는 방법에 대한 자세한 내용은 553페이지의 "AIR에서 드래그 앤 드롭" 및 JavaScript에서 드래그 앤 드롭 사용(Apple 개발자 센터)을 참조하십시오.

innerHTML 및 outerHTML 속성

Adobe AIR 1.0 이상

AIR에서는 보안상의 이유로 응용 프로그램 샌드박스에서 실행 중인 내용에 대해 innerHTML 및 outerHTML 속성의 사용을 제한합니다. 페이지 load 이벤트 이전 및 load 이벤트 핸들러 실행 중에는 innerHTML 및 outerHTML 속성을 제약 없이 사용할 수 있습니다. 그러나 페이지가 로드된 후에는 문서에 정적 내용을 추가하는 데에만 innerHTML 또는 outerHTML 속성을 사용할 수 있습니다. innerHTML 또는 outerHTML에 할당된 문자열에서 실행 코드로 평가되는 모든 명령문은 무시됩니다. 예를 들어 요소 정의에 이벤트 콜백 특성을 포함한 경우 이벤트 리스너가 추가되지 않습니다. 마찬가지로 포함된 <script> 태그는 평가되지 않습니다. 자세한 내용은 983페이지의 “Adobe AIR의 HTML 보안”을 참조하십시오.

Document.write() 및 Document.writeln() 메서드

Adobe AIR 1.0 이상

페이지의 load 이벤트 전에는 응용 프로그램 샌드박스에서 write() 및 writeln() 메서드를 제약 없이 사용할 수 있습니다. 그러나 페이지가 로드된 후에는 이러한 메서드를 호출해도 페이지가 지워지거나 새 페이지가 만들어지지 않습니다. 비 응용 프로그램 샌드박스에서는 대부분의 웹 브라우저에서처럼 페이지 로드가 완료된 후 document.write() 또는 writeln()을 호출하면 현재 페이지가 지워지고 비어 있는 새 페이지가 열립니다.

Document.designMode 속성

Adobe AIR 1.0 이상

문서의 모든 요소를 편집할 수 있도록 만들려면 document.designMode 속성을 on 값으로 설정합니다. 내장 편집기에서는 텍스트 편집, 복사, 붙여넣기 및 드래그 앤 드롭이 지원됩니다. designMode를 on으로 설정하는 것은 body 요소의 contentEditable 속성을 true로 설정하는 것과 같습니다. 대부분의 HTML 요소에서 contentEditable 속성을 사용하여 편집할 수 있는 문서 섹션을 정의할 수 있습니다. 자세한 내용은 886페이지의 “HTML contentEditable 특성”을 참조하십시오.

unload 이벤트(body 및 frameset 객체)

Adobe AIR 1.0 이상

윈도우(응용 프로그램의 기본 윈도우 포함)의 최상위 frameset 또는 body 태그에서는 닫히는 윈도우 또는 응용 프로그램에 unload 이벤트를 사용하여 응답하지 마십시오. 대신 NativeApplication 객체의 exiting 이벤트를 사용하여 응용 프로그램이 닫히는 시간을 감지합니다. 또는 NativeWindow 객체의 closing 이벤트를 사용하여 윈도우가 닫히는 시간을 감지합니다. 예를 들어 다음 JavaScript 코드에서는 사용자가 응용 프로그램을 닫을 때 메시지("Goodbye.")를 표시합니다.

```
var app = air.NativeApplication.nativeApplication;
app.addEventListener(air.Event.EXITING, closeHandler);
function closeHandler(event)
{
    alert("Goodbye.");
}
```

그러나 스크립트는 프레임, iframe 또는 최상위 윈도우 내용의 탐색으로 발생한 unload 이벤트에 성공적으로 응답할 수 있습니다.

참고: 이러한 제한 사항은 이후 버전의 Adobe AIR에서 제거됩니다.

JavaScript Window 객체

Adobe AIR 1.0 이상

Window 객체는 JavaScript 실행 컨텍스트에서 전역 객체로 유지됩니다. 응용 프로그램 샌드박스에서 AIR은 AIR의 내장 클래스 및 중요한 호스트 객체에 액세스하는 데 사용할 수 있는 새 속성을 JavaScript Window 객체에 추가합니다. 또한 일부 메서드 및 속성은 응용 프로그램 샌드박스 내에 있는지 여부에 따라 다르게 동작합니다.

Window.runtime 속성 runtime 속성을 사용하면 응용 프로그램 샌드박스 내에서 내장 런타임 클래스를 인스턴스화하고 사용할 수 있습니다. 이러한 클래스에는 AIR 및 Flash Player API가 포함되지만 Flex 프레임워크 등은 포함되지 않습니다. 예를 들어 다음 명령문은 AIR File 객체를 만듭니다.

```
var preferencesFile = new window.runtime.flash.filesystem.File();
```

AIR SDK에 제공되는 AIRAliases.js 파일에는 이러한 참조를 줄이는 데 사용할 수 있는 별칭 정의가 포함되어 있습니다. 예를 들어 AIRAliases.js를 페이지로 가져오면 다음 명령문을 사용하여 File 객체를 만들 수 있습니다.

```
var preferencesFile = new air.File();
```

window.runtime 속성은 응용 프로그램 샌드박스 내의 내용 및 프레임 또는 iframe이 있는 페이지의 부모 문서에 대해서만 정의됩니다.

899페이지의 “AIRAliases.js 파일 사용”을 참조하십시오.

Window.nativeWindow 속성 nativeWindow 속성은 기본 window 객체에 대한 참조를 제공합니다. 이 속성을 사용하면 화면 위치, 크기 및 가시성과 같은 윈도우 함수 및 속성을 스크립팅하고 닫기, 크기 조정 및 이동과 같은 윈도우 이벤트를 처리할 수 있습니다. 예를 들어 다음 명령문은 윈도우를 닫습니다.

```
window.nativeWindow.close();
```

참고: NativeWindow 객체에서 제공하는 윈도우 제어 기능은 JavaScript Window 객체에서 제공하는 기능과 겹칩니다. 이러한 경우 가장 간편한 메서드를 사용할 수 있습니다.

window.nativeWindow 속성은 응용 프로그램 샌드박스 내의 내용 및 프레임 또는 iframe이 있는 페이지의 부모 문서에 대해서만 정의됩니다.

Window.htmlLoader 속성 htmlLoader 속성은 HTML 내용을 포함하는 AIR HTMLLoader 객체에 대한 참조를 제공합니다. 이 속성을 사용하면 HTML 환경의 모양과 비헤이비어를 스크립팅할 수 있습니다. 예를 들어 htmlLoader.paintsDefaultBackground 속성을 사용하여 컨트롤을 기본 흰색 배경으로 칠할지 여부를 결정할 수 있습니다.

```
window.htmlLoader.paintsDefaultBackground = false;
```

참고: HTMLLoader 객체 자체에는 포함된 HTML 내용의 JavaScript Window 객체를 참조하는 window 속성이 있습니다. 이 속성을 사용하면 포함하는 HTMLLoader에 대한 참조를 통해 JavaScript 환경에 액세스할 수 있습니다.

window.htmlLoader 속성은 응용 프로그램 샌드박스 내의 내용 및 프레임 또는 iframe이 있는 페이지의 부모 문서에 대해서만 정의됩니다.

Window.parentSandboxBridge 및 Window.childSandboxBridge 속성 parentSandboxBridge 및 childSandboxBridge 속성을 사용하면 부모 및 자식 프레임 간의 인터페이스를 정의할 수 있습니다. 자세한 내용은 908페이지의 “서로 다른 보안 샌드박스의 내용 크로스 스크립팅”을 참조하십시오.

Window.setTimeout() 및 Window.setInterval() 함수 AIR에서는 보안상의 이유로 응용 프로그램 샌드박스 내에서 setTimeout() 및 setInterval() 함수의 사용을 제한합니다. setTimeout() 또는 setInterval()을 호출할 때 문자열로 실행할 코드를 정의할 수 없습니다. 함수 참조를 사용해야 합니다. 자세한 내용은 896페이지의 “setTimeout() 및 setInterval()”을 참조하십시오.

Window.open() 함수 비 응용 프로그램 샌드박스에서 실행 중인 코드에서 open() 메서드를 호출하는 경우에는 마우스 클릭이나 키 누르기와 같은 사용자 상호 작용의 결과로 호출될 때만 윈도우를 엽니다. 또한 원격 내용에서 여는 윈도우가 응용 프로그램에서 여는 윈도우로 가장하지 못하도록 응용 프로그램 제목 앞에 윈도우 제목을 붙입니다. 자세한 내용은 988페이지의 “JavaScript window.open() 메서드 호출에 대한 제한 사항”을 참조하십시오.

air.NativeApplication 객체

Adobe AIR 1.0 이상

NativeApplication 객체는 응용 프로그램 상태에 대한 정보를 제공하고 여러 중요한 응용 프로그램 수준의 이벤트를 전달하여 응용 프로그램 비헤이비어를 제어하는 유용한 함수를 제공합니다. NativeApplication 객체의 단일 인스턴스가 자동으로 생성되며 클래스 정의 NativeApplication.nativeApplication 속성을 통해 이 인스턴스에 액세스할 수 있습니다.

JavaScript 코드에서 객체에 액세스하려면 다음을 사용할 수 있습니다.

```
var app = window.runtime.flash.desktop.NativeApplication.nativeApplication;
```

또는 AIRAliases.js 스크립트를 가져온 경우 다음과 같은 짧은 형식을 사용할 수 있습니다.

```
var app = air.NativeApplication.nativeApplication;
```

NativeApplication 객체는 응용 프로그램 샌드박스 내에서만 액세스할 수 있습니다. NativeApplication 객체에 대한 자세한 내용은 807페이지의 “AIR 런타임 및 운영 체제 정보 작업”을 참조하십시오.

JavaScript URL 스킴

Adobe AIR 1.0 이상

JavaScript URL 스킴 내에 정의된 코드 실행(예: href="javascript:alert('Test')")은 응용 프로그램 샌드박스 내에서 차단됩니다. 오류는 발생하지 않습니다.

AIR에서의 HTML

Adobe AIR 1.0 이상

AIR 및 WebKit에서는 다음을 포함한 몇 가지의 비 표준 HTML 요소 및 특성을 정의합니다.

883페이지의 “HTML 프레임 및 iframe 요소”

885페이지의 “HTML 요소 이벤트 핸들러”

HTML 프레임 및 iframe 요소

Adobe AIR 1.0 이상

AIR에서는 응용 프로그램 샌드박스에서 내용의 프레임 및 iframe 요소에 다음과 같은 새 특성을 추가합니다.

sandboxRoot 특성 sandboxRoot 특성은 프레임 src 특성에서 지정한 파일에 대해 원본의 비 응용 프로그램 대체 도메인을 지정합니다. 파일은 지정된 도메인에 해당하는 비 응용 프로그램 샌드박스로 로드됩니다. 파일의 내용과 지정된 도메인에서 로드되는 내용은 서로 크로스 스크립팅할 수 있습니다.

중요: sandboxRoot 값을 도메인의 기준 URL로 설정하는 경우 해당 도메인의 내용에 대한 모든 요청은 해당 요청이 페이지 탐색, XMLHttpRequest 또는 내용을 로드하는 다른 방법 중 무엇을 통해 발생했는지에 관계없이 원격 서버가 아닌 응용 프로그램 디렉토리에서 로드됩니다.

documentRoot 특성 documentRoot 특성은 sandboxRoot에서 지정한 위치 내의 파일로 확인되는 URL을 로드할 로컬 디렉토리를 지정합니다.

프레임 src 특성 또는 프레임에 로드된 내용에서 URL을 확인할 때 sandboxRoot에 지정한 값과 일치하는 URL 부분이 documentRoot에 지정한 값으로 대체됩니다. 따라서 다음 프레임 태그에서

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

child.html은 응용 프로그램 설치 폴더의 sandbox 하위 디렉토리에서 로드됩니다. child.html의 상대 URL은 sandbox 디렉토리를 기준으로 확인됩니다. www.example.com/air의 원격 서버에 있는 파일은 프레임에서 액세스할 수 없습니다. AIR이 이러한 파일을 app:/sandbox/ 디렉토리에서 로드하려고 하기 때문입니다.

allowCrossDomainXHR 특성 프레임의 내용에서 원격 도메인에 대한 XMLHttpRequests를 만들 수 있게 하려면 여는 프레임 태그에 allowCrossDomainXHR="allowCrossDomainXHR"을 포함합니다. 기본적으로 비 응용 프로그램 내용은 자신의 원래 도메인에 대해서만 이러한 요청을 할 수 있습니다. 크로스 도메인 XHR을 허용할 경우 심각한 보안 영향이 발생할 수 있습니다. 페이지의 코드에서 도메인과 데이터를 교환할 수 있으므로, 악의적인 내용이 페이지에 삽입되는 경우 현재 샌드박스의 코드에 액세스할 수 있는 데이터가 손상될 수 있습니다. 사용자가 만들고 제어하는 페이지에 대해서만, 그리고 크로스 도메인 데이터 로딩이 정말로 필요한 경우에만 크로스 도메인 XHR을 사용합니다. 또한 페이지에서 로드한 모든 외부 데이터의 유효성을 세밀히 검사하여 코드 삽입이나 다른 형태의 공격을 예방합니다.

중요: allowCrossDomainXHR 특성이 프레임 또는 iframe 요소에 포함되어 있는 경우 할당된 값이 "0"이거나 문자 "f" 또는 "n"으로 시작하지 않는 한 크로스 도메인 XHR이 사용됩니다. 예를 들어 allowCrossDomainXHR을 "deny"로 설정해도 크로스 도메인 XHR이 계속 사용됩니다. 크로스 도메인 요청을 사용하지 않으려면 이 특성을 완전히 요소 선언 밖에 둡니다.

ondominitialize 특성 프레임의 dominitialize 이벤트에 대한 이벤트 핸들러를 지정합니다. 이 이벤트는 프레임의 Window 및 Document 객체가 만들어진 후 스크립트가 파싱되거나 문서 요소가 만들어지기 전에 발생하는 AIR 관련 이벤트입니다.

dominitialize 핸들러에서 자식 문서에 추가한 객체, 변수 및 함수를 자식 페이지의 스크립트에서 참조할 수 있도록 프레임에서는 dominitialize 이벤트를 로드 시퀀스에서 초기에 전달합니다. 부모 페이지가 자식과 동일한 샌드박스에 있어야만 자식 문서의 객체를 직접 추가하거나 액세스할 수 있습니다. 그러나 응용 프로그램 샌드박스의 부모가 샌드박스 브리지를 설정하여 비 응용 프로그램 샌드박스의 내용과 통신할 수 있습니다.

다음 예에서는 AIR에서의 iframe 태그 사용을 보여 줍니다.

child.html을 원격 서버의 실제 도메인에 매핑하지 않고 원격 샌드박스에 놓습니다.

```
<iframe src="http://localhost/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://localhost/air/"/>
```

child.html을 원격 샌드박스에 놓아 www.example.com에 대한 XMLHttpRequests만 허용합니다.

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"/>
```

child.html을 원격 샌드박스에 놓아 모든 원격 도메인에 대한 XMLHttpRequests를 허용합니다.

```
<iframe src="http://www.example.com/air/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="http://www.example.com/air/"
            allowCrossDomainXHR="allowCrossDomainXHR"/>
```

child.html을 local-with-file-system 샌드박스에 놓습니다.

```
<iframe     src="file:///templates/child.html"
            documentRoot="app:/sandbox/"
            sandboxRoot="app-storage:/templates/"/>
```

dominitialize 이벤트를 통해 샌드박스 브리지를 설정하여 child.html을 원격 샌드박스에 놓습니다.

```
<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>
```

다음 child.html 문서에서는 자식 내용이 부모 샌드박스 브리지에 액세스하는 방법을 보여 줍니다.

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

자세한 내용은 908페이지의 “서로 다른 보안 샌드박스의 내용 크로스 스크립팅” 및 983페이지의 “Adobe AIR의 HTML 보안”을 참조하십시오.

HTML 요소 이벤트 핸들러

Adobe AIR 1.0 이상

AIR 및 Webkit의 DOM 객체는 표준 DOM 이벤트 모델에는 없는 몇 가지 이벤트를 전달합니다. 다음 표에서는 이러한 이벤트의 핸들러를 지정하는 데 사용할 수 있는 관련 이벤트 특성을 나열합니다.

콜백 특성 이름	설명
oncontextmenu	선택한 텍스트에서 명령 클릭이나 마우스 오른쪽 버튼 클릭과 같은 동작을 통해 컨텍스트 메뉴를 호출할 때 호출됩니다.
oncopy	요소에서 선택 내용을 복사할 때 호출됩니다.
oncut	요소에서 선택 내용을 잘라낼 때 호출됩니다.
ondominitialize	프레임 또는 iframe에 로드된 문서의 DOM을 만들 때 DOM 요소가 만들어지거나 스크립트가 파싱되기 전에 호출됩니다.
ondrag	요소를 드래그할 때 호출됩니다.
ondragend	드래그하고 마우스 버튼을 놓을 때 호출됩니다.
ondragenter	드래그 동작이 요소 경계로 들어갈 때 호출됩니다.
ondragleave	드래그 동작이 요소 경계 밖으로 나갈 때 호출됩니다.
ondragover	드래그 동작이 요소 경계 내에 있을 때 연속적으로 호출됩니다.
ondragstart	드래그 동작이 시작될 때 호출됩니다.

콜백 특성 이름	설명
ondrop	드래그 동작이 요소 위에 있는 동안 마우스 버튼을 놓을 때 호출됩니다.
onerror	요소를 로드하는 동안 오류가 발생하면 호출됩니다.
oninput	양식 요소에 텍스트를 입력할 때 호출됩니다.
onpaste	요소에 항목을 붙여넣을 때 호출됩니다.
onscroll	스크롤할 수 있는 요소의 내용을 스크롤할 때 호출됩니다.
onselectstart	선택이 시작될 때 호출됩니다.

HTML contentEditable 특성

Adobe AIR 1.0 이상

HTML 요소에 contentEditable 특성을 추가하여 사용자가 요소의 내용을 편집할 수 있도록 할 수 있습니다. 예를 들어 다음 예제 HTML 코드에서는 첫 번째 p 요소를 제외하고 전체 문서를 편집할 수 있도록 설정합니다.

```
<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>
```

참고: document.designMode 속성을 on으로 설정하면 개별 요소의 contentEditable 설정에 관계없이 문서의 모든 요소를 편집할 수 있게 됩니다. 그러나 designMode를 off로 설정해도 contentEditable이 true인 요소를 편집할 수 없게 되지는 않습니다. 자세한 내용은 881페이지의 “Document.designMode 속성”을 참조하십시오.

data: URL

Adobe AIR 2 이상

AIR는 다음 요소에 대해 data: URL을 지원합니다.

- img
- input type="image"
- 이미지(예: 배경 이미지)를 허용하는 CSS 규칙

data: URL을 사용하면 이진 이미지 데이터를 base64 인코딩 문자열로 CSS 또는 HTML 문서에 직접 삽입할 수 있습니다. 다음 예에서는 data: URL을 반복되는 배경으로 사용합니다.

```
<html>
<head>
<style>
body {
background-
image:url('
ZSBJbWFnZVJlYWR5ccllPAAAAAZQTFRF%2F6cA%2F%2F%2F%2Fgxp3lwAAAAJ0Uk5T%2FwDltzBKAAABF0lEQVR42uzZQQ7CMAxE0e%2F
7X5oNCyRocWzPiJbMBZ6qpIljE%2BnwklgKG7kwUjc2IkIaxkY0CPdEsCCasws6ShXBgmBBmEagpXQQLAgWBAuSY2gaKaWPYEGwIEwg0F
RmECwIFoQeQjJlhJWUEFazjFDJCkI5WYRWMgjtfEGYyQnCXD4jTCdm1zmngFpBFznwVNi5RPSbwbWnpYr%2BBHi%2FtCTfgPLEPL7jBct
AKBRptXJ8M%2BprIuZKu%2BUKcg4YK1PLz7kx4bSqHyPaT4d%2B28OCJJiRBo4FCQsSA0bziT3XubMgYUG6fc5fatmGBQkL0hoJ1IaZMi
QsSFiQ8vRscTjlQOI2iHZwtpHuf%2BJAYiOiJSkj8Z%2FIQ4ABANvXGLd3%2BZMrAAAAAElFTkSuQmCC');
background-repeat:repeat;
}
</style>
</head>
<body>
</body>
</html>
```

data: URL을 사용할 때는 추가 공백에 유의해야 합니다. 즉, 데이터 문자열을 분리되지 않은 단일 행으로 입력해야 합니다. 그렇지 않으면 행 분리가 데이터의 일부로 간주되어 이미지를 디코딩할 수 없게 됩니다.

AIR에서의 CSS

Adobe AIR 1.0 이상

WebKit에서는 여러 확장 CSS 속성을 지원합니다. 이러한 대부분의 확장은 접두어 -webkit를 사용합니다. 이러한 확장 중 일부는 시험용이므로 이후 WebKit 버전에서 제거될 수도 있습니다. WebKit의 CSS 및 CSS 확장 지원에 대한 자세한 내용은 Safari CSS 참조를 참조하십시오.

AIR에서 지원되지 않는 WebKit 기능

Adobe AIR 1.0 이상

AIR에서는 다음과 같은 WebKit 또는 Safari 4 기능이 지원되지 않습니다.

- window.postMessage를 통한 크로스 도메인 메시징(AIR에서 자체 크로스 도메인 통신 API 제공)
- CSS 변수
- WOFF(Web Open Font Format) 및 SVG 글꼴
- HTML video 및 audio 태그
- 미디어 장치 쿼리
- 오프라인 응용 프로그램 캐시
- 인쇄(AIR에서 자체 PrintJob API 제공)
- 맞춤법 및 문법 검사
- SVG
- WAI-ARIA
- WebSockets(AIR에서 자체 소켓 API 제공)
- Web workers
- WebKit SQL API(AIR에서 자체 API 제공)
- WebKit 지리적 위치 정보 API(AIR에서 자체 지리적 위치 정보 API를 지원되는 장치에 제공)

- WebKit 다중 파일 업로드 API
- WebKit 터치 이벤트(AIR에서 자체 터치 이벤트 제공)
- WML(Wireless Markup Language)

다음 목록은 AIR에서 지원되지 않는 특정 JavaScript API, HTML 요소, CSS 속성 및 값을 보여 줍니다.

지원되지 않는 JavaScript Window 객체 멤버:

- applicationCache()
- console
- openDatabase()
- postMessage()
- document.print()

지원되지 않는 HTML 태그:

- audio
- video

지원되지 않는 HTML 특성:

- aria-*
- draggable
- formnovalidate
- list
- novalidate
- onbeforeload
- onhashchange
- onorientationchange
- onpagehide
- onpageshow
- onpopstate
- ontouchstart
- ontouchmove
- ontouchend
- ontouchcancel
- onwebkitbeginfullscreen
- onwebkitendfullscreen
- pattern
- required
- sandbox

지원되지 않는 JavaScript 이벤트:

- beforeload
- hashchange
- orientationchange
- pagehide
- pageshow
- popstate
- touchstart
- touchmove
- touchend
- touchcancel
- webkitbeginfullscreen
- webkitendfullscreen

지원되지 않는 CSS 속성:

- background-clip
- background-origin(-webkit-background-origin 사용)
- background-repeat-x
- background-repeat-y
- background-size(-webkit-background-size 사용)
- 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

지원되지 않는 CSS 값:

- 모양 속성 값:
  - media-volume-slider-container
  - media-volume-slider
  - media-volume-sliderthumb
  - outer-spin-button

- border-box(background-clip 및 background-origin)
- contain(background-size)
- content-box(background-clip 및 background-origin)
- cover(background-size)
- 목록 속성 값:
  - 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(표시 속성)

59장: AIR에서 HTML 및 JavaScript 프로그래밍

Adobe AIR 1.0 이상

상당수의 프로그래밍 항목은 HTML 및 JavaScript를 사용한 Adobe® AIR® 응용 프로그램 개발에 초점을 맞추고 있습니다. HTML 기반 AIR 응용 프로그램을 프로그래밍하는지, 아니면 HTMLLoader 클래스 또는 mx:HTML Flex™ 구성 요소를 사용하여 HTML 및 JavaScript를 실행하는 SWF 기반 AIR 응용 프로그램을 프로그래밍하는지에 상관없이 다음 정보가 중요합니다.

HTMLLoader 클래스

Adobe AIR 1.0 이상

Adobe AIR의 HTMLLoader 클래스는 AIR 응용 프로그램에 HTML 내용을 표시할 수 있는 표시 객체를 정의합니다. SWF 기반 응용 프로그램은 HTMLLoader 컨트롤을 기존 윈도우에 추가하거나 HTMLLoader.createRootWindow()를 사용하여 HTMLLoader 객체를 자동으로 포함하는 HTML 윈도우를 만들 수 있습니다. HTMLLoader 객체는 로드된 HTML 페이지 내에서 JavaScript window.htmlLoader 속성을 통해 액세스할 수 있습니다.

URL에서 HTML 내용 로드

Adobe AIR 1.0 이상

다음 코드에서는 URL을 HTMLLoader 객체로 로드(응용 프로그램의 HTML 내용을 표시하기 위한 스테이지 또는 기타 표시 객체 컨테이너의 자식으로 HTMLLoader 추가)합니다.

```
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);
```

HTMLLoader 객체의 width 및 height 속성을 모두 기본적으로 0으로 설정합니다. HTMLLoader 객체를 스테이지에 추가할 때 이러한 크기를 설정할 수 있습니다. HTMLLoader는 페이지가 로드될 때 여러 이벤트를 전달합니다. 이러한 이벤트를 사용하여 로드된 페이지와 상호 작용해도 되는 경우를 결정할 수 있습니다. 이러한 이벤트에 대해서는 927페이지의 “AIR에서 HTML 관련 이벤트 처리”에서 자세하게 설명합니다.

참고: Flex 프레임워크에서 UIComponent 클래스를 확장하는 클래스만 Flex 컨테이너 구성 요소의 자식으로 추가할 수 있습니다. 따라서 HTMLLoader를 Flex 컨테이너 구성 요소의 자식으로 직접 추가할 수 없습니다. 그러나 Flex mx:HTML 컨트롤을 사용하거나, UIComponent를 확장하고 HTMLLoader를 UIComponent의 자식으로 포함하는 사용자 정의 클래스를 작성하거나, HTMLLoader를 UIComponent의 자식으로 추가하고 UIComponent를 Flex 컨테이너에 추가할 수 있습니다.

또한 TextField 클래스를 사용하여 HTML 텍스트를 렌더링할 수 있지만 그 기능은 제한적입니다. Adobe® Flash® Player의 TextField 클래스는 HTML 마크업의 하위 집합을 지원하지만 크기 제한 때문에 그 기능은 제한적입니다. Adobe AIR에 포함된 HTMLLoader 클래스는 Flash Player에서 사용할 수 없습니다.

문자열에서 HTML 내용 로드

Adobe AIR 1.0 이상

HTMLLoader 객체의 loadString() 메서드는 HTML 내용의 문자열을 HTMLLoader 객체로 로드합니다.

```
var html:HTMLLoader = new HTMLLoader();
var htmlStr:String = "<html><body>Hello <b>world</b>.</body></html>";
html.loadString(htmlStr);
```

기본적으로 loadString() 메서드를 통해 로드된 내용은 비 응용 프로그램 샌드박스에 배치되며 다음 특성을 갖게 됩니다.

- 파일 시스템이 아니라 네트워크에서 내용을 로드할 수 있게 됩니다.
- XMLHttpRequest를 사용하여 데이터를 로드할 수 없습니다.
- window.location 속성이 "about:blank"로 설정됩니다.
- 내용이 모든 비 응용 프로그램 샌드박스의 내용과 마찬가지로 window.runtime 속성에 액세스할 수 없습니다.

AIR 1.5에서는 HTMLLoader 클래스에 placeLoadStringContentInApplicationSandbox 속성이 포함됩니다. 이 속성을 HTMLLoader 객체에 대해 true로 설정하면 loadString() 메서드를 통해 로드된 내용이 응용 프로그램 샌드박스에 배치됩니다(기본값: false). 이렇게 하면 loadString() 메서드를 통해 로드된 내용이 window.runtime 속성 및 모든 AIR API에 액세스할 수 있게 됩니다. 이 속성을 true로 설정하는 경우 loadString() 메서드 호출에 사용된 문자열의 데이터 소스를 신뢰할 수 있는지 확인해야 합니다. 이 속성을 true로 설정하면 HTML 문자열의 코드 문이 전체 응용 프로그램 권한으로 실행됩니다. 문자열에 유해한 코드가 포함되어 있지 않은 경우에만 이 속성을 true로 설정하십시오.

AIR 1.0 또는 AIR 1.1 SDK로 컴파일된 응용 프로그램에서는 loadString() 메서드를 통해 로드된 내용이 응용 프로그램 샌드박스에 배치됩니다.

AIR 응용 프로그램에서 HTML 사용 시 중요한 보안 규칙

Adobe AIR 1.0 이상

AIR 응용 프로그램과 함께 설치한 파일은 AIR API에 액세스할 수 있습니다. 보안상의 이유로 다른 소스의 내용은 이에 해당하지 않습니다. 예를 들어 이러한 제한 사항으로 인해 원격 도메인(예: http://example.com)의 내용이 사용자의 데스크톱 디렉토리를 읽을 수 없거나 더 많은 제약을 받게 됩니다.

eval() 함수 및 관련 API 호출을 통해 노출될 수 있는 보안 취약성이 존재하기 때문에 응용 프로그램과 함께 설치되는 내용은 기본적으로 이러한 메서드를 사용할 수 없습니다. 그러나 일부 Ajax 프레임워크는 eval() 함수 및 관련 API를 호출하는 기능을 사용합니다.

내용이 AIR 응용 프로그램에서 작동하도록 올바르게 구조화하려면 서로 다른 소스의 내용에 대한 보안 제한 사항의 규칙을 고려해야 합니다. 서로 다른 소스의 내용은 샌드박스(950페이지의 “보안 샌드박스” 참조)라고 하는 개별 보안 분류에 배치됩니다. 기본적으로 응용 프로그램과 함께 설치되는 내용은 **응용 프로그램** 샌드박스라고 하는 샌드박스에 설치되며 AIR API에 대한 액세스 권한이 부여됩니다. 응용 프로그램 샌드박스는 일반적으로 가장 안전한 샌드박스로, 신뢰할 수 없는 코드 실행을 금지하도록 제한합니다.

런타임을 통해 응용 프로그램과 함께 설치된 내용을 응용 프로그램 샌드박스가 아닌 다른 샌드박스에 로드할 수 있습니다. 비 응용 프로그램 샌드박스의 내용은 일반적인 웹 브라우저의 경우와 유사한 보안 환경에서 작동합니다. 예를 들어 비 응용 프로그램 샌드박스의 코드는 eval() 및 관련 메서드를 사용할 수 있지만 동시에 AIR API에 액세스할 수는 없습니다. 런타임에는 AIR API를 비 응용 프로그램 내용에 표시하지 않고 서로 다른 샌드박스의 내용이 안전하게 통신할 수 있도록 하는 몇 가지 방법이 있습니다. 자세한 내용은 908페이지의 “서로 다른 보안 샌드박스의 내용 크로스 스크립팅”을 참조하십시오.

보안상의 이유로 샌드박스에서의 사용이 제한되는 코드를 호출하면 런타임에서 JavaScript 오류: "응용 프로그램 보안 샌드박스에서의 JavaScript 코드에 대한 Adobe AIR 런타임 보안 위반"을 전달합니다.

이 오류를 방지하려면 다음 단원 894페이지의 “보안 관련 JavaScript 오류 방지”에서 설명하는 코딩 방법을 따르십시오.

자세한 내용은 983페이지의 “Adobe AIR의 HTML 보안”을 참조하십시오.

보안 관련 JavaScript 오류 방지

Adobe AIR 1.0 이상

이러한 보안 제한으로 인해 샌드박스에서의 사용이 제한되는 코드를 호출하면 런타임에서 JavaScript 오류: "응용 프로그램 보안 샌드박스에서의 JavaScript 코드에 대한 Adobe AIR 런타임 보안 위반"을 전달합니다. 이 오류를 방지하려면 이러한 코딩 방법을 따르십시오.

보안 관련 JavaScript 오류의 원인

Adobe AIR 1.0 이상

문서 load 이벤트가 발생하고 load 이벤트 핸들러가 종료된 경우 문자열을 평가 및 실행하는 대부분의 작업에서는 응용 프로그램 샌드박스에서 코드를 실행할 수 없습니다. 잠재적으로 안전하지 않은 문자열을 평가 및 실행하는 다음 유형의 JavaScript 명령문을 사용하려고 하면 JavaScript 오류가 생성됩니다.

- eval() 함수
- setTimeout() 및 setInterval()
- 함수 생성자

  또한 다음 유형의 JavaScript 명령문이 안전하지 않은 JavaScript 오류를 생성하지 않고 실패할 수 있습니다.

- javascript: URL
- innerHTML 및 outerHTML 문의 onevent 특성을 통해 할당된 이벤트 콜백
- 응용 프로그램 설치 디렉토리 외부에서 JavaScript 파일 로드
- document.write() 및 document.writeln()
- load 이벤트 이전 또는 load 이벤트 핸들러 동안의 동기 XMLHttpRequests
- 동적으로 만든 스크립트 요소

  참고: 일부 제한적인 경우에 문자열을 평가할 수 있습니다. 자세한 내용은 985페이지의 “서로 다른 샌드박스의 내용에 대한 코드 제한 사항”을 참조하십시오.

  Adobe에서는 http://www.adobe.com/go/airappsandboxframeworks_kr에서 응용 프로그램 보안 샌드박스를 지원하는 것으로 알려진 Ajax 프레임워크 목록을 유지합니다.

  다음 단원에서는 응용 프로그램 샌드박스에서 실행 중인 코드에 대한 이러한 안전하지 않은 JavaScript 오류 및 표시되지 않는 오류를 방지하기 위해 스크립트를 다시 작성하는 방법에 대해 설명합니다.

다른 샌드박스에 응용 프로그램 매핑

Adobe AIR 1.0 이상

대부분의 경우 보안 관련 JavaScript 오류를 방지하기 위해 응용 프로그램을 다시 작성하거나 다시 구조화할 수 있습니다. 그러나 다시 작성 또는 다시 구조화를 수행할 수 없는 경우에는 909페이지의 “비 응용 프로그램 샌드박스로 응용 프로그램 내용 로드”에 설명되어 있는 기술을 사용하여 응용 프로그램 내용을 다른 샌드박스로 로드할 수 있습니다. 해당 내용이 AIR API에 액세스할 수 있는 경우 909페이지의 “샌드박스 브리지 인터페이스 설정”에 설명되어 있는 대로 샌드박스 브리지를 만들 수 있습니다.

eval() 함수

Flash Player 9 이상, Adobe AIR 1.0 이상

응용 프로그램 샌드박스에서 eval() 함수는 페이지 load 이벤트 이전 또는 load 이벤트 핸들러 동안에만 사용할 수 있습니다. 페이지를 로드한 후 eval()을 호출하면 코드가 실행되지 않습니다. 그러나 다음 경우 eval() 사용을 방지하도록 코드를 다시 작성할 수 있습니다.

객체에 속성 할당

Adobe AIR 1.0 이상

다음과 같이 속성 접근자를 작성하도록 문자열을 파싱하는 대신

```
eval("obj." + propName + " = " + val);
```

다음과 같이 대괄호 표기법을 사용하여 속성에 액세스합니다.

```
obj[propName] = val;
```

컨텍스트에서 사용할 수 있는 변수를 포함하는 함수 만들기

Adobe AIR 1.0 이상

다음과 같은 명령문을

```
function compile(var1, var2){
    eval("var fn = function(){ this."+var1+"(var2) }");
    return fn;
}
```

다음으로 바꿀 수 있습니다.

```
function compile(var1, var2){
    var self = this;
    return function(){ self[var1](var2) };
}
```

클래스 이름을 문자열 매개 변수로 사용하여 객체 만들기

Adobe AIR 1.0 이상

다음 코드로 정의된 가상 JavaScript 클래스를 검토합니다.

```
var CustomClass =
    {
        Utils:
        {
            Parser: function(){ alert('constructor') }
        },
        Data:
        {

        }
    };
var constructorClassName = "CustomClass.Utils.Parser";
```

인스턴스를 만드는 가장 간단한 방법은 eval()을 사용하는 것입니다.

```
var myObj;
eval('myObj=new ' + constructorClassName +'()')
```

그러나 클래스 이름의 각 구성 요소를 파싱하고 대괄호 표기법을 통해 새 객체를 작성하여 eval()에 대한 호출을 방지할 수 있습니다.

```
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;
}
```

이 인스턴스를 만들려면 다음을 사용합니다.

```
try{
    var Parser = getter(constructorClassName);
    var a = new Parser();
    }catch(e){
        alert(e);
}
```

setTimeout() 및 setInterval()

Adobe AIR 1.0 이상

핸들러 함수로 전달한 문자열을 함수 참조 또는 객체로 바꿉니다. 예를 들어 다음과 같은 명령문을

```
setTimeout("alert('Timeout')", 100);
```

다음으로 바꿀 수 있습니다.

```
setTimeout(function(){alert('Timeout')}, 100);
```

또는 함수를 사용하기 위해 호출자가 this 객체를 설정해야 하는 경우 다음과 같은 명령문을

```
this.appTimer = setInterval("obj.customFunction();", 100);
```

다음으로 바꿉니다.

```
var _self = this;
this.appTimer = setInterval(function(){obj.customFunction.apply(_self);}, 100);
```

함수 생성자

Adobe AIR 1.0 이상

new Function(param, body)에 대한 호출을 인라인 함수 선언으로 바꾸거나 페이지 load 이벤트를 처리하기 이전에만 사용할 수 있습니다.

javascript: URL

Adobe AIR 1.0 이상

javascript: URL 스킴을 사용하여 링크에 정의된 코드는 응용 프로그램 샌드박스에서 무시됩니다. 안전하지 않은 JavaScript 오류는 생성되지 않습니다. 다음과 같이 javascript: URL을 사용하는 링크를

```
<a href="javascript:code()">Click Me</a>
```

다음으로 바꿀 수 있습니다.

```
<a href="#" onclick="code()">Click Me</a>
```

innerHTML 및 outerHTML 문의 onevent 특성을 통해 할당된 이벤트 콜백

Adobe AIR 1.0 이상

innerHTML 또는 outerHTML을 사용하여 문서의 DOM에 요소를 추가할 때 onclick 또는 onmouseover와 같이 명령문 내에 할당된 모든 이벤트 콜백은 무시됩니다. 보안 오류는 생성되지 않습니다. 대신 addEventListener() 메서드를 사용하여 새 요소에 id 특성을 할당하고 이벤트 핸들러 콜백 함수를 설정할 수 있습니다.

예를 들어 다음과 같이 문서에 대상 요소가 있는 경우

```
<div id="container"></div>
```

다음과 같은 명령문을

```
document.getElementById('container').innerHTML =
    '<a href="#" onclick="code()">Click Me.</a>';
```

다음으로 바꿀 수 있습니다.

```
document.getElementById('container').innerHTML = '<a href="#" id="smith">Click Me.</a>';
document.getElementById('smith').addEventListener("click", function() { code(); });
```

응용 프로그램 설치 디렉토리 외부에서 JavaScript 파일 로드

Adobe AIR 1.0 이상

응용 프로그램 샌드박스 외부에서 스크립트 파일을 로드할 수 없습니다. 보안 오류는 생성되지 않습니다. 응용 프로그램 샌드박스에서 실행되는 모든 스크립트 파일은 응용 프로그램 디렉토리에 설치되어야 합니다. 페이지의 외부 스크립트를 사용하려면 해당 페이지를 다른 샌드박스로 매핑해야 합니다. 909페이지의 “비 응용 프로그램 샌드박스로 응용 프로그램 내용 로드”를 참조하십시오.

document.write() 및 document.writeln()

Adobe AIR 1.0 이상

document.write() 또는 document.writeln()에 대한 호출은 페이지 load 이벤트를 처리한 이후 무시됩니다. 보안 오류는 생성되지 않습니다. 대신 DOM 조작 기술을 사용하여 새 파일을 로드하거나 문서의 본문을 바꿀 수 있습니다.

load 이벤트 이전 또는 load 이벤트 핸들러 동안의 동기 XMLHttpRequests

Adobe AIR 1.0 이상

페이지 load 이벤트 이전 또는 load 이벤트 핸들러 동안에 시작된 동기 XMLHttpRequests는 내용을 반환하지 않습니다. 비동기 XMLHttpRequests는 시작될 수 있지만 load 이벤트 이후에만 반환됩니다. load 이벤트가 처리되어야 동기 XMLHttpRequests가 정상적으로 작동합니다.

동적으로 만든 스크립트 요소

Adobe AIR 1.0 이상

innerHTML 또는 document.createElement() 메서드를 사용하여 만든 스크립트 요소와 같이 동적으로 만든 스크립트 요소는 무시됩니다.

JavaScript에서 AIR API 클래스에 액세스

Adobe AIR 1.0 이상

HTML 및 JavaScript 코드에서는 Webkit의 표준 및 확장 요소 이외에도 런타임에서 제공하는 호스트 클래스에 액세스할 수 있습니다. 이러한 클래스를 사용하면 다음과 같이 AIR에서 제공하는 고급 기능에 액세스할 수 있습니다.

- 파일 시스템에 액세스
- 로컬 SQL 데이터베이스 사용
- 응용 프로그램 및 윈도우 메뉴 제어
- 네트워킹을 위해 소켓에 액세스
- 사용자 정의 클래스 및 객체 사용
- 사운드 기능

  예를 들어 AIR 파일 API는 flash.filesystem 패키지에 들어 있는 File 클래스를 포함합니다. 다음과 같이 JavaScript에서 File 객체를 만들 수 있습니다.

  ```
  var myFile = new window.runtime.flash.filesystem.File();
  ```

  runtime 객체는 응용 프로그램 샌드박스의 AIR에서 실행 중인 HTML 내용에서 사용할 수 있는 JavaScript 객체입니다. 이 객체를 통해 JavaScript에서 런타임 클래스에 액세스할 수 있습니다. runtime 객체의 flash 속성은 flash 패키지에 대한 액세스를 제공합니다. 또한 runtime 객체의flash.filesystem 속성은 flash.filesystem 패키지에 대한 액세스를 제공합니다. 이 패키지는 File 클래스를 포함합니다. 패키지는 ActionScript에 사용된 클래스를 구성하는 한 방법입니다.

  참고: runtime 속성은 프레임 또는 iframe에 로드된 페이지의 window 객체에 자동으로 추가되지 않습니다. 그러나 자식 문서가 응용 프로그램 샌드박스에 있는 동안에는 자식이 부모의 runtime 속성에 액세스할 수 있습니다.

  런타임 클래스의 패키지 구조의 경우 개발자가 window.runtime.flash.desktop.NativeApplication과 같이 각 클래스에 액세스하기 위해 JavaScript 코드 문자열의 긴 문자열을 입력해야 하기 때문에 AIR SDK에는 air.NativeApplication을 간단하게 입력하는 작업 등을 통해 런타임 클래스에 보다 쉽게 액세스할 수 있게 해주는 AIRAliases.js 파일이 포함되어 있습니다.

  AIR API 클래스는 이 안내서 전체에 나옵니다. HTML 개발자가 관심을 가질만한 Flash Player API의 다른 클래스에 대한 내용은 **HTML 개발자를 위한 Adobe AIR API 참조 설명서**를 참조하십시오. ActionScript는 SWF(Flash Player) 내용에서 사용하는 언어입니다. 그러나 JavaScript 구문과 ActionScript 구문은 서로 유사합니다. 이 두 구문은 모두 ECMAScript 언어 버전을 기반으로 합니다. 모든 기본 제공 클래스를 JavaScript(HTML 내용의 경우)와 ActionScript(SWF 내용의 경우) 모두에서 사용할 수 있습니다.

참고: JavaScript 코드는 ActionScript에서 이용할 수 있는 Dictionary, XML 및 XMLList 클래스를 사용할 수 없습니다.

AIRAliases.js 파일 사용

Adobe AIR 1.0 이상

런타임 클래스는 다음과 같이 패키지 구조로 구성됩니다.

- window.runtime.flash.desktop.NativeApplication
- window.runtime.flash.desktop.ClipboardManager
- window.runtime.flash.filesystem.FileStream
- window.runtime.flash.data.SQLDatabase

AIR SDK에는 적은 입력으로 런타임 클래스에 액세스할 수 있게 해주는 "별칭" 정의를 제공하는 AIRAliases.js 파일이 포함됩니다. 예를 들어 다음과 같이 간단하게 입력하여 위에 나열된 클래스에 액세스할 수 있습니다.

- air.NativeApplication
- air.Clipboard
- air.FileStream
- air.SQLDatabase

이 목록은 AIRAliases.js 파일에 있는 클래스의 간단한 하위 집합일 뿐입니다. 클래스 및 패키지 수준 함수의 전체 목록은 **HTML 개발자를 위한 Adobe AIR API 참조 설명서**에 정리되어 있습니다.

AIRAliases.js 파일에는 일반적으로 사용되는 런타임 클래스 이외에도 일반적으로 사용되는 패키지 수준 함수에 대한 별칭이 포함되어 있습니다. 예를 들어 window.runtime.trace(), window.runtime.flash.net.navigateToURL() 및 window.runtime.flash.net.sendToURL()이 있으며 이들은 air.trace(), air.navigateToURL() 및 air.sendToURL()로 앨리어싱됩니다.

AIRAliases.js 파일을 사용하려면 HTML 페이지에 다음과 같은 script 참조를 포함합니다.

```
<script src="AIRAliases.js"></script>
```

필요한 경우 src 참조의 경로를 조정합니다.

중요: 명시된 경우는 제외하고 이 설명서의 JavaScript 예제 코드에서는 HTML 페이지에 AIRAliases.js 파일을 포함했다고 가정합니다.

AIR의 URL

Adobe AIR 1.0 이상

AIR에서 실행하는 HTML 내용에서는 link 태그의 href 특성 또는 URL을 제공할 수 있는 다른 위치에서 img, frame, iframe 및 script 태그에 대한 src 특성을 정의하는 데 다음과 같은 URL 스킴을 사용할 수 있습니다.

URL 스킴	설명	예제
file	파일 시스템 루트에 상대적인 경로입니다.	`file:///c:/AIR Test/test.txt`
app	설치된 응용 프로그램의 루트 디렉토리에 상대적인 경로입니다.	`app:/images`
app-storage	응용 프로그램 저장소 디렉토리에 상대적인 경로입니다. 설치된 각 응용 프로그램에 대해 AIR은 해당 응용 프로그램 관련 데이터를 저장하기에 좋은 장소인 고유한 응용 프로그램 저장소 디렉토리를 정의합니다.	`app-storage:/settings/prefs.xml`
http	표준 HTTP 요청입니다.	`http://www.adobe.com`
https	표준 HTTPS 요청입니다.	`https://secure.example.com`

AIR에서의 URL 스킴 사용에 대한 자세한 내용은 741페이지의 "URI 스킴"을 참조하십시오.

File, Loader, URLStream 및 Sound 클래스와 같은 많은 AIR API는 URL을 포함하는 문자열이 아닌 URLRequest 객체를 사용합니다. URLRequest 객체 자체는 동일한 URL 스킴을 사용할 수 있는 문자열로 초기화됩니다. 예를 들어 다음 명령문에서는 Adobe 홈 페이지를 요청하는 데 사용할 수 있는 URLRequest 객체를 만듭니다.

```
var urlReq = new air.URLRequest("http://www.adobe.com/");
```

URLRequest 객체에 대한 자세한 내용은 739페이지의 "HTTP 통신"을 참조하십시오.

ActionScript 객체를 JavaScript에서 사용할 수 있도록 지정

Adobe AIR 1.0 이상

HTMLLoader 객체에서 로드하는 HTML 페이지의 JavaScript는 HTML 페이지의 window.runtime, window.htmlLoader, 및 window.nativeWindow 속성을 사용하여 ActionScript 실행 컨텍스트에 정의된 클래스, 객체 및 함수를 호출할 수 있습니다. 또한 JavaScript 실행 컨텍스트 내에서 ActionScript 객체 및 함수에 대한 참조를 만들어 이러한 객체 및 함수를 JavaScript 코드에서 사용할 수 있도록 지정할 수 있습니다.

ActionScript에서 JavaScript 객체에 액세스하는 기본 예제

Adobe AIR 1.0 이상

다음 예제에서는 ActionScript 객체를 참조하는 속성을 HTML 페이지의 전역 window 객체에 추가하는 방법을 보여 줍니다.

```
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;
}
```

이전 예제에 나와 있는 HTMLLoader 객체로 로드된 HTML 내용(test.html이라는 파일에 포함)은 부모 SWF 파일에 정의되어 있는 foo 속성 및 helloFromJS() 메서드에 액세스할 수 있습니다.

```
<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>
```

로드 중인 문서의 JavaScript 컨텍스트에 액세스할 때 htmlDOMInitialize 이벤트를 사용하면 페이지에 정의되어 있는 스크립트에서 액세스할 수 있도록 페이지 생성 시퀀스에서 충분히 초기에 객체를 만들 수 있습니다. complete 이벤트를 기다릴 때 페이지 load 이벤트 이후에 실행하는 페이지의 스크립트만 추가된 객체에 액세스할 수 있습니다.

클래스 정의를 JavaScript에서 사용할 수 있도록 지정

Adobe AIR 1.0 이상

응용 프로그램의 ActionScript 클래스를 JavaScript에서 사용할 수 있도록 지정하려면 로드된 HTML 내용을 클래스 정의를 포함하는 응용 프로그램 도메인에 할당합니다. JavaScript 실행 컨텍스트의 응용 프로그램 도메인은 HTMLLoader 객체의 runtimeApplicationDomain 속성을 사용하여 설정할 수 있습니다. 예를 들어 응용 프로그램 도메인을 기본 응용 프로그램 도메인으로 설정하려면 다음 코드에 나와 있는 대로 runtimeApplicationDomain을 ApplicationDomain.currentDomain으로 설정합니다.

```
html.runtimeApplicationDomain = ApplicationDomain.currentDomain;
```

runtimeApplicationDomain 속성이 설정되면 JavaScript 컨텍스트가 클래스 정의를 할당된 도메인과 공유합니다. JavaScript에서 사용자 정의 클래스의 인스턴스를 만들려면 window.runtime 속성을 통해 클래스 정의를 참조하고 new 연산자를 사용합니다.

```
var customClassObject = new window.runtime.CustomClass();
```

HTML 내용은 호환되는 보안 도메인에서 가져와야 합니다. HTML 내용을 사용자가 할당하는 응용 프로그램 도메인의 보안 도메인이 아닌 다른 보안 도메인에서 가져온 경우 페이지는 대신 기본 응용 프로그램 도메인을 사용합니다. 예를 들어 인터넷에서 원격 페이지를 로드하는 경우 ApplicationDomain.currentDomain을 페이지의 응용 프로그램 도메인으로 할당할 수 없습니다.

이벤트 리스너 제거

Adobe AIR 1.0 이상

JavaScript 이벤트 리스너를 현재 페이지 외부에 있는 객체(런타임 객체, 로드된 SWF 내용의 객체, 다른 페이지에서 실행하는 JavaScript 객체까지 포함)에 추가하는 경우 페이지가 언로드할 때 항상 이러한 이벤트 리스너를 제거해야 합니다. 그렇지 않으면 이벤트 리스너가 이벤트를 더 이상 존재하지 않는 핸들러 함수에 전달합니다. 이러한 작업이 수행되면 "응용 프로그램에서 더 이상 로드되지 않는 HTML 페이지의 JavaScript 객체를 참조하려고 합니다."라는 오류 메시지가 표시됩니다. 필요 없는 이벤트 리스너를 제거해도 AIR이 관련된 메모리를 회수합니다. 자세한 내용은 931페이지의 "탐색하는 HTML 페이지의 이벤트 리스너 제거"를 참조하십시오.

ActionScript에서 HTML DOM 및 JavaScript 객체에 액세스

Adobe AIR 1.0 이상

HTMLLoader 객체에서 complete 이벤트를 전달하면 페이지에 대한 HTML DOM(문서 객체 모델)의 모든 객체에 액세스할 수 있습니다. 액세스할 수 있는 객체에는 표시 객체(예: 페이지의 div 및 p 객체)는 물론 JavaScript 변수 및 함수도 포함됩니다. complete 이벤트는 JavaScript 페이지 load 이벤트에 해당합니다. complete를 전달하기 전에 DOM 요소, 변수 및 함수를 파싱 또는 만들지 못했을 수 있습니다. 가능하면 HTML DOM에 액세스하기 전에 complete 이벤트를 기다립니다.

예를 들어 다음과 같은 HTML 페이지를 살펴봅니다.

```
<html>
    <script>
        foo = 333;
        function test() {
            return "OK.";
        }
    </script>
    <body>
        <p id="p1">Hi.</p>
    </body>
</html>
```

이 간단한 HTML 페이지는 **foo**라고 하는 JavaScript 변수와 **test()**라고 하는 JavaScript 함수를 정의합니다. 이 두 항목 모두 페이지 전역 window 객체의 속성입니다. 또한 window.document 객체에는 getElementById() 메서드를 사용하여 액세스할 수 있는 ID **p1**의 이름이 지정된 P 요소가 포함됩니다. 페이지를 로드하면(HTMLLoader 객체에서 complete 이벤트를 전달하는 경우) 다음 ActionScript 코드에 나와 있는 대로 ActionScript에서 이러한 객체 각각에 액세스할 수 있습니다.

```
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.
}
```

HTML 요소의 내용에 액세스하려면 innerHTML 속성을 사용합니다. 예를 들어 이전 코드에서 html.window.document.getElementById("p1").innerHTML을 사용하여 **p1**이라고 하는 HTML 요소의 내용을 가져옵니다.

또한 ActionScript에서 HTML 페이지의 속성을 설정할 수 있습니다. 예를 들어 다음은 포함하는 HTMLLoader 객체에 대한 참조를 사용하여 페이지에서 foo JavaScript 변수의 값 및 p1 요소의 내용을 설정하는 예제입니다.

```
 html.window.document.getElementById("p1").innerHTML = "Goodbye";
html.window.foo = 66;
```

HTML에 SWF 내용 포함

Adobe AIR 1.0 이상

브라우저의 경우와 같이 AIR 응용 프로그램 내에서 HTML 내용에 SWF 내용을 포함할 수 있습니다. object 태그와 embed 태그 중 하나 또는 둘 다를 사용하여 SWF 내용을 포함합니다.

참고: 일반적인 웹 개발 방법은 object 태그와 embed 태그를 모두 사용하여 HTML 페이지에 SWF 내용을 표시하는 것입니다. 이러한 방법은 AIR에 아무런 이점이 없습니다. AIR에 표시할 내용에 W3C 표준 object 태그를 단독으로 사용할 수 있습니다. 동시에 브라우저에도 표시되는 HTML 내용에 대해 필요한 경우 object 태그와 embed 태그를 계속 함께 사용할 수 있습니다.

HTML 및 SWF 내용을 표시하는 NativeWindow 객체에서 투명도를 설정한 경우, 내용을 포함하는 데 사용되는 윈도우 모드(wmode)가 window 값으로 설정되면 SWF 내용이 표시되지 않습니다. 투명 윈도우의 HTML 페이지에서 SWF 내용을 표시하려면 wmode 매개 변수를 opaque 또는 transparent로 설정하십시오. window가 wmode의 기본값이므로 값을 지정하지 않으면 내용이 표시되지 않을 수 있습니다.

다음 예제에서는 HTML object 태그를 사용하여 HTML 내용 내에 SWF 파일을 표시하는 방법을 보여 줍니다. 기본 NativeWindow 객체가 투명한 경우에도 내용이 표시되도록 wmode 매개 변수가 opaque로 설정됩니다. SWF 파일은 응용 프로그램 디렉토리에서 로드되지만 사용자는 AIR에서 지원하는 모든 URL 스킴을 사용할 수 있습니다. SWF 파일이 로드되는 위치에 따라 AIR이 내용을 배치하는 보안 샌드박스가 결정됩니다.

```
<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>
```

또한 스크립트를 사용하여 내용을 동적으로 로드할 수 있습니다. 다음은 urlString 매개 변수에 지정된 SWF 파일을 표시하기 위해 object 노드를 만드는 예제입니다. 이 예제에서는 노드를 elementID 매개 변수에서 지정한 ID의 페이지 요소의 자식으로 추가합니다.

```
<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>
```

HTMLLoader 객체가 크기 조절 또는 회전되거나 alpha 속성이 1.0 이외의 값으로 설정되는 경우에는 SWF 내용이 표시되지 않습니다. AIR 1.5.2 이전에는 어떤 wmode 값을 설정하더라도 SWF 내용이 투명 윈도우에 표시되지 않았습니다.

참고: 포함된 SWF 객체에서 비디오 파일 같은 외부 에셋을 로드하려는 경우 HTML 파일에 비디오 파일의 절대 경로가 제공되어 있지 않으면 SWF 내용이 올바르게 렌더링되지 않을 수 있습니다. 하지만 포함된 SWF 객체에서는 상대 경로를 사용하여 외부 이미지 파일을 로드할 수 있습니다.

다음 예제에서는 HTML 내용에 포함된 SWF 객체를 통해 외부 에셋을 로드하는 방법을 보여 줍니다.

```
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");

}
```

다음 ActionScript 예제에서는 HTML 파일을 통해 전달된 이미지 경로를 읽고 이미지를 스테이지로 로드합니다.

```
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;
                  }
            }
      }
}
```

HTML 페이지 내에서 ActionScript 라이브러리 사용

Adobe AIR 1.0 이상

AIR은 페이지가 컴파일된 SWF 파일에서 ActionScript 클래스를 가져올 수 있도록 HTML 스크립트 요소를 확장합니다. 예를 들어 루트 응용 프로그램 폴더의 lib 하위 디렉토리에 있는 **myClasses.swf**라는 라이브러리를 가져오려면 HTML 파일 내에 다음과 같은 script 태그를 포함합니다.

```
<script src="lib/myClasses.swf" type="application/x-shockwave-flash"></script>
```

중요: 라이브러리를 올바르게 로드하기 위해 유형 특성이 type="application/x-shockwave-flash"이어야 합니다.

SWF 내용이 Flash Player 10 또는 AIR 1.5 SWF로 컴파일되는 경우 응용 프로그램 설명자 파일의 XML 네임스페이스를 AIR 1.5 네임스페이스로 설정해야 합니다.

AIR 파일을 패키지할 때 lib 디렉토리 및 myClasses.swf 파일도 포함되어야 합니다.

가져온 클래스를 JavaScript Window 객체의 runtime 속성을 통해 액세스합니다.

```
var libraryObject = new window.runtime.LibraryClass();
```

SWF 파일의 클래스가 패키지로 구성되어 있으면 패키지 이름도 포함해야 합니다. 예를 들어 LibraryClass 정의가 **utilities**라는 패키지에 포함되어 있으면 다음 명령문을 사용하여 클래스 인스턴스를 만듭니다.

```
var libraryObject = new window.runtime.utilities.LibraryClass();
```

참고: AIR의 HTML 페이지 일부로 사용할 ActionScript SWF 라이브러리를 컴파일하려면 acompc 컴파일러를 사용합니다. acompc 유틸리티는 Flex SDK의 일부이며 Flex SDK 설명서에 나와 있습니다.

가져온 ActionScript 파일에서 HTML DOM 및 JavaScript 객체에 액세스

Adobe AIR 1.0 이상

<script> 태그를 사용하여 페이지로 가져온 SWF 파일의 ActionScript에서 HTML 페이지의 객체에 액세스하려면 window 또는 document와 같은 JavaScript 객체에 대한 참조를 ActionScript 코드에 정의되어 있는 함수에 전달합니다. 함수 내의 참조를 사용하여 JavaScript 객체 또는 전달된 참조를 통해 액세스할 수 있는 다른 객체에 액세스합니다.

예를 들어 다음과 같은 HTML 페이지를 살펴봅니다.

```
<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>
```

이 간단한 HTML 페이지는 **num**이라고 하는 JavaScript 변수와 **getStatus()**라고 하는 JavaScript 함수를 포함합니다. 이 두 항목 모두 페이지 window 객체의 속성입니다. 또한 window.document 객체에는 ID **p1**의 이름이 지정된 P 요소가 포함됩니다.

페이지는 ASClass 클래스를 포함하는 "ASLibrary.swf"라는 ActionScript 파일을 로드합니다. ASClass는 이러한 JavaScript 객체의 값을 추적하기만 하는 accessDOM()이라는 함수를 정의합니다. accessDOM() 메서드는 JavaScript Window 객체를 인수로 사용합니다. 이러한 Window 참조를 사용하여 다음 정의에서 설명한 대로 페이지에서 변수, 함수 및 DOM 요소를 포함하는 기타 객체에 액세스할 수 있습니다.

```
public class ASClass{
    public function accessDOM(window:*):void {
        trace(window.num); // 254
        trace(window.document.getElementById("p1").innerHTML); // Body text..
        trace(window.getStatus()); // OK.
    }
}
```

가져온 ActionScript 클래스에서 HTML 페이지의 속성을 가져오고 설정할 수 있습니다. 예를 들어 다음 함수는 페이지에서 p1 요소의 내용을 설정하고 페이지에서 foo JavaScript 변수의 값을 설정합니다.

```
public function modifyDOM(window:*):void {
    window.document.getElementById("p1").innerHTML = "Bye";
    window.foo = 66;
```

Date 및 RegExp 객체 변환

Adobe AIR 1.0 이상

JavaScript 언어와 ActionScript 언어는 모두 Date 클래스와 RegExp 클래스를 정의하지만 이러한 유형의 객체는 두 실행 컨텍스트 사이에서 자동으로 변환되지 않습니다. Date 및 RegExp 객체를 사용하여 대체 실행 컨텍스트의 속성 또는 함수 매개 변수를 설정하기 전에 이러한 객체를 해당되는 유형으로 변환해야 합니다.

예를 들어 다음 ActionScript 코드는 jsDate라는 JavaScript Date 객체를 ActionScript Date 객체로 변환합니다.

```
var asDate:Date = new Date(jsDate.getMilliseconds());
```

다음 ActionScript 코드는 jsRegExp라는 JavaScript RegExp 객체를 ActionScript RegExp 객체로 변환합니다.

```
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);
```

ActionScript에서 HTML 스타일 시트 조작

Adobe AIR 1.0 이상

HTMLLoader 객체에서 complete 이벤트를 전달하면 페이지에서 CSS 스타일을 검토하고 조작할 수 있습니다.

예를 들어 다음과 같은 간단한 HTML 문서를 살펴봅니다.

```
<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>
```

HTMLLoader 객체가 이 내용을 로드하면 여기에 나와 있는 대로 window.document.styleSheets 배열의 cssRules 배열을 통해 페이지에서 CSS 스타일을 조작할 수 있습니다.

```
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";
}
```

이 코드는 결과 HTML 문서가 다음과 같이 표시되도록 CSS 스타일을 조정합니다.

Style 1A

Style 1B

Style 2

HTMLLoader 객체가 complete 이벤트를 전달한 이후에만 코드가 페이지에 스타일을 추가할 수 있다는 사실을 기억하십시오.

서로 다른 보안 샌드박스의 내용 크로스 스크립팅

Adobe AIR 1.0 이상

런타임 보안 모델은 서로 다른 원본에서 코드를 분리합니다. 서로 다른 보안 샌드박스의 내용을 크로스 스크립팅하면 한 보안 샌드박스의 내용이 다른 샌드박스의 선택된 속성 및 메서드에 액세스할 수 있도록 지정할 수 있습니다.

AIR 보안 샌드박스 및 JavaScript 코드

Adobe AIR 1.0 이상

AIR은 한 도메인의 코드가 다른 도메인의 내용과 상호 작용할 수 없는 동일 원본 정책을 적용합니다. 모든 파일은 해당 원본을 기반으로 하여 샌드박스에 배치됩니다. 원래 응용 프로그램 샌드박스의 내용은 동일 원본 원칙을 위반하고 응용 프로그램 설치 디렉토리 외부에서 로드한 내용을 크로스 스크립팅할 수 없습니다. 그러나 AIR은 비 응용 프로그램 내용을 크로스 스크립팅할 수 있는 몇 가지 기술을 제공합니다.

한 기술은 프레임 또는 iframe을 사용하여 응용 프로그램 내용을 다른 보안 샌드박스로 매핑하는 것입니다. 응용 프로그램의 샌드박싱된 영역에서 로드한 페이지는 원격 도메인에서 로드한 것처럼 동작합니다. 예를 들어 응용 프로그램 내용을 **example.com** 도메인으로 매핑하면 해당 내용이 example.com에서 로드된 페이지를 크로스 스크립팅할 수 있습니다.

이 기술은 응용 프로그램 내용을 다른 샌드박스에 배치하기 때문에 해당 내용 내에 있는 코드가 더 이상 평가된 문자열의 코드 실행에 대한 제한을 적용받지 않습니다. 이 샌드박스 매핑 기술을 사용하면 원격 내용을 크로스 스크립팅할 필요가 없는 경우에도 이러한 제한을 완화할 수 있습니다. 이러한 방식으로 내용을 매핑하는 기술은 많은 JavaScript 프레임워크 중 하나를 사용하거나 문자열 평가에 의존하는 기존 코드를 사용할 때 특히 유용합니다. 그러나 내용이 응용 프로그램 샌드박스 외부에서 실행될 때 신뢰할 수 없는 내용을 삽입하고 실행할 수 있는 추가 위험에 주의하고 미연에 방지해야 합니다.

동시에 다른 샌드박스에 매핑된 응용 프로그램 내용이 AIR API에 대한 액세스 권한을 상실하기 때문에 응용 프로그램 샌드박스 외부에서 실행된 코드에 AIR 기능을 표시하는 데 샌드박스 매핑 기술을 사용할 수 없습니다.

다른 크로스 스크립팅 기술을 사용하면 비 응용 프로그램 샌드박스의 내용과 응용 프로그램 샌드박스에 있는 해당 부모 문서 간의 **샌드박스 브리지**라고 하는 인터페이스를 만들 수 있습니다. 브리지를 통해 자식 내용이 부모가 정의한 속성 및 메서드에 액세스하거나 부모가 자식이 정의한 속성 및 메서드에 액세스할 수 있습니다. 또는 이 두 작업을 모두 수행할 수 있습니다.

마지막으로 응용 프로그램 샌드박스 및 기타 샌드박스(선택 사항)에서 크로스 도메인 XMLHttpRequests를 수행할 수도 있습니다.

자세한 내용은 883페이지의 “HTML 프레임 및 iframe 요소”, 983페이지의 “Adobe AIR의 HTML 보안” 및 878페이지의 “XMLHttpRequest 객체”를 참조하십시오.

비 응용 프로그램 샌드박스로 응용 프로그램 내용 로드

Adobe AIR 1.0 이상

응용 프로그램 내용이 응용 프로그램 설치 디렉토리 외부에서 로드한 내용을 안전하게 크로스 스크립팅하려면 frame 또는 iframe 요소를 사용하여 응용 프로그램 내용을 외부 내용과 동일한 보안 샌드박스로 로드합니다. 원격 내용을 크로스 스크립팅할 필요가 없지만 응용 프로그램 페이지를 응용 프로그램 샌드박스 외부에 로드하려는 경우 원본 도메인과 동일하게 http://localhost/ 또는 다른 몇 가지 안전한 값을 지정하는 방법을 사용할 수 있습니다.

AIR은 프레임에 로드한 응용 프로그램 파일을 비 응용 프로그램 샌드박스로 매핑해야 할지 여부를 지정할 수 있는 프레임 요소에 새 특성인 sandboxRoot 및 documentRoot를 추가합니다. sandboxRoot URL 아래의 경로로 확인되는 파일은 대신 documentRoot 디렉토리에서 로드됩니다. 보안상의 이유로 이러한 방식으로 로드한 응용 프로그램 내용은 실제로 sandboxRoot URL에서 로드한 것처럼 처리됩니다.

sandboxRoot 속성은 프레임 내용을 배치할 샌드박스 및 도메인을 결정하는 데 사용할 URL을 지정합니다. file:, http: 또는 https: URL 스킴을 사용해야 합니다. 상대 URL을 지정하는 경우 내용이 응용 프로그램 샌드박스에 남습니다.

documentRoot 속성은 프레임 내용을 로드할 디렉토리를 지정합니다. file:, app: 또는 app-storage: URL 스킴을 사용해야 합니다.

다음은 원격 샌드박스 및 www.example.com 도메인에서 실행할 응용 프로그램의 sandbox 하위 디렉토리에 설치된 내용을 매핑하는 예제입니다.

```
<iframe
    src="http://www.example.com/local/ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

ui.html 페이지는 다음 script 태그를 사용하여 로컬 sandbox 폴더에서 javascript 파일을 로드할 수 있습니다.

```
<script src="http://www.example.com/local/ui.js"></script>
```

또한 다음과 같이 script 태그를 사용하여 원격 서버의 디렉토리에서 내용을 로드할 수 있습니다.

```
<script src="http://www.example.com/remote/remote.js"></script>
```

sandboxRoot URL는 원격 서버의 동일한 URL에서 모든 내용을 마스크 처리합니다. 위의 예제에서는 AIR이 요청을 로컬 응용 프로그램 디렉토리로 다시 매핑하기 때문에 www.example.com/local/ 또는 해당 하위 디렉토리의 원격 내용에 액세스할 수 없습니다. 요청은 페이지 탐색, XMLHttpRequest 또는 다른 내용 로드 방법에서 파생되었는지 여부에 상관없이 다시 매핑됩니다.

샌드박스 브리지 인터페이스 설정

Adobe AIR 1.0 이상

응용 프로그램 샌드박스의 내용이 비 응용 프로그램 샌드박스의 내용에서 정의한 속성 또는 메서드에 액세스해야 하는 경우 또는 비 응용 프로그램 내용이 응용 프로그램 샌드박스의 내용에서 정의한 속성 및 메서드에 액세스해야 하는 경우 샌드박스 브리지를 사용할 수 있습니다. 자식 문서의 window 객체에 대한 childSandboxBridge 및 parentSandboxBridge 속성을 사용하여 브리지를 만들 수 있습니다.

자식 샌드박스 브리지 구축

Adobe AIR 1.0 이상

childSandboxBridge 속성을 통해 자식 문서가 인터페이스를 부모 문서의 내용에 표시할 수 있습니다. 인터페이스를 표시하려면 childSandbox 속성을 자식 문서의 함수 또는 객체로 설정합니다. 그렇게 하면 부모 문서의 내용에서 객체 또는 함수에 액세스할 수 있습니다. 다음 예제에서는 자식 문서에서 실행하는 스크립트에서 함수 및 속성을 포함하는 객체를 해당 부모에 표시하는 방법을 보여 줍니다.

```
var interface = {};
interface.calculatePrice = function(){
    return ".45 cents";
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

이 자식 내용이 "child"의 id가 할당된 iframe에 로드된 경우 프레임의 childSandboxBridge 속성을 읽어 부모 내용에서 인터페이스에 액세스할 수 있습니다.

```
var childInterface = document.getElementById("child").contentWindow.childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces ".45 cents"
air.trace(childInterface.storeID)); //traces "abc"
```

부모 샌드박스 브리지 구축

Adobe AIR 1.0 이상

parentSandboxBridge 속성을 통해 부모 문서가 인터페이스를 자식 문서의 내용에 표시할 수 있습니다. 인터페이스를 표시하려면 부모 문서가 자식 문서의 parentSandbox 속성을 부모 문서에서 정의한 함수 또는 객체로 설정합니다. 그렇게 하면 자식의 내용에서 객체 또는 함수에 액세스할 수 있습니다. 다음 예제에서는 부모 프레임에서 실행하는 스크립트에서 함수를 포함하는 객체를 자식 문서에 표시하는 방법을 보여 줍니다.

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").contentWindow.parentSandboxBridge = interface;
```

이 인터페이스를 사용하면 자식 프레임의 내용이 save.txt라는 파일에 텍스트를 저장할 수 있지만 파일 시스템에 대한 기타 액세스 권한이 없습니다. 자식 내용은 다음과 같이 save 함수를 호출할 수 있습니다.

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

응용 프로그램 내용은 가능한 가장 좁은 인터페이스를 다른 샌드박스에 표시해야 합니다. 비 응용 프로그램 내용은 실수 또는 악의적인 코드 삽입의 피해를 입을 수 있기 때문에 본질적으로 신뢰할 수 없는 것으로 간주해야 합니다. 적합한 보호 조치를 마련하여 부모 샌드박스 브리지를 통해 표시하는 인터페이스가 잘못 사용되지 않도록 해야 합니다.

페이지를 로드하는 동안 부모 샌드박스 브리지에 액세스

Adobe AIR 1.0 이상

자식 문서의 스크립트에서 부모 샌드박스 브리지에 액세스하기 위해서는 스크립트를 실행하기 전에 브리지를 설정해야 합니다. Window, 프레임 및 iframe 객체는 새 페이지 DOM이 생성되지만 스크립트가 파싱되거나 DOM 요소가 추가되기 전에 dominitialize 이벤트를 전달합니다. dominitialize 이벤트를 사용하여 자식 문서의 모든 스크립트가 액세스할 수 있도록 페이지 생성 시퀀스에서 충분히 초기에 브리지를 구축할 수 있습니다.

다음 예제에서는 자식 프레임에서 전달한 dominitialize 이벤트에 응답하여 부모 샌드박스 브리지를 만드는 방법을 보여 줍니다.

```
<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>
```

다음 child.html 문서에서는 자식 내용이 부모 샌드박스 브리지에 액세스하는 방법을 보여 줍니다.

```
<html>
    <head>
        <script>
            document.write(window.parentSandboxBridge.testProperty);
        </script>
    </head>
    <body></body>
</html>
```

프레임이 아닌 자식 윈도우의 dominitialize 이벤트를 수신하려면 window.open() 함수에서 만든 새 자식 window 객체에 리스너를 추가해야 합니다.

```
var childWindow = window.open();
childWindow.addEventListener("dominitialize", engageBridge());
childWindow.document.location = "http://www.example.com/air/child.html";
```

이 경우 응용 프로그램 내용을 비 응용 프로그램 샌드박스로 매핑할 방법은 없습니다. 이 기술은 응용 프로그램 디렉토리 외부에서 child.html을 로드한 경우에만 유용합니다. 윈도우의 응용 프로그램 내용을 비 응용 프로그램 샌드박스에 매핑할 수 있지만 먼저 자체적으로 프레임을 사용하여 자식 문서를 로드하고 해당 문서를 원하는 샌드박스로 매핑하는 중간 페이지를 로드해야 합니다.

HTMLLoader 클래스 createRootWindow() 함수를 사용하여 윈도우를 만드는 경우 새 윈도우는 createRootWindow()가 호출되는 문서의 자식이 아닙니다. 따라서 새 윈도우로 로드된 비 응용 프로그램 내용에 대한 호출 윈도우에서 샌드박스 브리지를 만들 수 없습니다. 대신 자체적으로 프레임을 사용하여 자식 문서를 로드하는 새 윈도우에서 중간 페이지를 로드해야 합니다. 그러고 나서 새 윈도우의 부모 문서에서 프레임에 로드된 자식 문서로의 브리지를 구축할 수 있습니다.

60장: AIR HTML 컨테이너 스크립팅

Adobe AIR 1.0 이상

HTMLLoader 클래스는 Adobe® AIR®에서 HTML 내용의 컨테이너 역할을 하며, ActionScript® 3.0 표시 목록에서 객체의 비헤이비어와 모양을 제어하기 위한 많은 속성과 메서드를 제공합니다. 이러한 속성과 메서드는 Sprite 클래스에서 상속한 것입니다. 또한 HTML 내용을 로드하여 상호 작용하고 작업 내역을 관리하는 등의 작업을 위한 속성과 메서드를 정의합니다.

HTMLHost 클래스는 HTMLLoader에 대한 기본 비헤이비어 세트를 정의합니다. HTMLLoader 객체를 만들 때 HTMLHost 구현은 제공되지 않으므로 HTML 내용이 윈도우 위치나 윈도우 제목의 변경과 같은 기본 비헤이비어 중 하나를 트리거하는 경우 어떤 동작도 발생하지 않습니다. HTMLHost 클래스를 확장하여 응용 프로그램에 적합한 비헤이비어를 정의할 수 있습니다.

HTMLHost의 기본 구현은 AIR에서 만들어진 HTML 윈도우에 제공됩니다. defaultBehavior 매개 변수를 true로 설정하여 만든 새 HTMLHost 객체를 사용하여 객체의 htmlHost 속성을 설정하는 방법으로 기본 HTMLHost 구현을 다른 HTMLLoader 객체에 할당할 수 있습니다.

참고: Adobe® Flex™ Framework에서는 HTMLLoader 객체가 mx:HTML 구성 요소로 래핑됩니다. Flex를 사용하는 경우 HTML 구성 요소를 사용합니다.

HTMLLoader 객체의 표시 속성

Adobe AIR 1.0 이상

HTMLLoader 객체는 Adobe® Flash® Player Sprite 클래스의 표시 속성을 상속합니다. 예를 들어, 배경색의 크기 조절, 이동, 숨기기 및 변경을 수행할 수 있습니다. 또는 필터, 마스크, 크기 조절 및 회전과 같은 고급 효과를 적용할 수 있습니다. 효과를 적용할 때는 가독성에 미치는 영향을 고려해야 합니다. 일부 효과가 적용된 경우 HTML 페이지에 로드된 SWF 및 PDF 내용을 표시할 수 없습니다.

HTML 윈도우에는 HTML 내용을 렌더링하는 HTMLLoader 객체가 포함되어 있습니다. 이 객체는 윈도우의 영역 안으로 제한되므로 크기, 위치, 회전 또는 비율 인수를 변경할 때 항상 원하는 결과가 생성되지는 않습니다.

기본 표시 속성

Adobe AIR 1.0 이상

HTMLLoader의 기본 표시 속성을 사용하면 부모 표시 객체 안에 컨트롤을 배치한 후 크기를 설정하고 표시하거나 숨길 수 있습니다. HTML 윈도우의 HTMLLoader 객체에 대한 이러한 속성은 변경하면 안 됩니다.

기본 속성은 다음과 같습니다.

속성	참고 사항
x, y	부모 컨테이너 안에 객체를 배치합니다.
width, height	표시 영역의 크기를 변경합니다.
visible	객체와 객체에 포함된 내용의 가시성을 제어합니다.

HTML 윈도우 밖에서 HTMLLoader 객체의 width 및 height 속성에 대한 기본값은 0입니다. 로드된 HTML 내용을 표시하려면 폭과 높이를 설정해야 합니다. HTML 내용은 HTMLLoader 크기로 그려지며 내용의 HTML 및 CSS 속성에 따라 배치됩니다. HTMLLoader 크기를 변경하면 내용이 다시 전개됩니다.

width가 여전히 0으로 설정된 상태로 내용을 새 HTMLLoader 객체에 로드하는 경우 contentWidth 및 contentHeight 속성을 사용하여 HTMLLoader의 표시 폭과 높이를 설정하기가 쉽습니다. 이 방법은 HTML 및 CSS 흐름 규칙에 따라 배치될 때 최소 폭이 적당한 페이지에는 효과가 있지만 일부 페이지는 HTMLLoader에서 제공하는 적당한 폭이 없으면 길고 좁은 레이아웃으로 전개됩니다.

참고: HTMLLoader 객체의 폭과 높이를 변경하면 대부분의 다른 표시 객체 유형에서 그렇듯이 scaleX 및 scaleY 값이 변경되지 않습니다.

HTMLLoader 내용의 투명도

Adobe AIR 1.0 이상

기본적으로 true인 HTMLLoader 객체의 paintsDefaultBackground 속성은 HTMLLoader 객체가 불투명한 배경을 그리는지 여부를 결정합니다. paintsDefaultBackground가 false이면 배경이 지워집니다. HTMLLoader 객체 아래의 다른 표시 객체나 표시 객체 컨테이너는 HTML 내용의 전경 요소 뒤에 표시됩니다.

HTML 문서의 body 요소나 다른 요소가 style="background-color:gray" 등을 사용하여 배경색을 지정하는 경우 해당 HTML 부분의 배경은 불투명하며 지정된 배경색으로 렌더링됩니다. HTMLLoader 객체의 opaqueBackground 속성을 설정하고 paintsDefaultBackground가 false인 경우 opaqueBackground에 설정된 색상이 표시됩니다.

참고: 투명한 PNG 형식의 그래픽을 사용하여 HTML 문서의 요소에 알파 블렌딩된 배경을 제공할 수 있습니다. HTML 요소의 불투명도 스타일은 설정할 수 없습니다.

HTMLLoader 내용 크기 조절

Adobe AIR 1.0 이상

HTMLLoader 객체의 크기를 비율 인수 1.0을 초과하게 조절하지 마십시오. HTMLLoader 내용의 텍스트는 특정 해상도로 렌더링되며 HTMLLoader 객체의 크기를 크게 조절하는 경우 픽셀화되어 나타납니다. 윈도우 크기가 조절될 때 HTMLLoader와 그 내용의 크기가 조절되지 않게 하려면 Stage의 scaleMode 속성을 StageScaleMode.NO_SCALE로 설정합니다.

SWF 또는 PDF 내용을 HTML 페이지에 로드하는 경우의 고려 사항

Adobe AIR 1.0 이상

HTMLLoader 객체에 로드된 SWF 및 PDF 내용은 다음과 같은 경우에 사라집니다.

- HTMLLoader 객체의 크기를 1.0이 아닌 인수로 조절하는 경우
- HTMLLoader 객체의 알파 속성을 1.0이 아닌 값으로 설정하는 경우
- HTMLLoader 내용을 회전하는 경우

잘못된 속성 설정을 제거하고 활성 필터를 제거하면 내용이 다시 나타납니다.

또한 런타임은 투명 윈도우에 PDF 내용을 표시할 수 없습니다. 객체의 wmode 매개 변수나 embed 태그가 opaque 또는 transparent로 설정된 경우에만 HTML 페이지에 포함된 SWF 내용이 표시됩니다. wmode의 기본값은 window이므로 wmode 매개 변수를 명시적으로 설정하지 않으면 투명 윈도우에 SWF 내용이 표시되지 않습니다.

참고: AIR 1.5.2 이전에는 어떤 wmode 값을 사용하더라도 HTML에 포함된 SWF를 표시할 수 없었습니다.

HTMLLoader에서 이러한 유형의 미디어를 로드하는 방법에 대한 자세한 내용은 903페이지의 “HTML에 SWF 내용 포함” 및 501페이지의 “AIR에서 PDF 내용 추가”를 참조하십시오.

고급 표시 속성

Adobe AIR 1.0 이상

HTMLLoader 클래스는 특수 효과에 사용할 수 있는 몇 가지 메서드를 상속합니다. 일반적으로 이러한 효과는 HTMLLoader 표시에서 사용될 때 제한이 따르지만 전환이나 다른 임시 효과에 유용할 수 있습니다. 예를 들어, 사용자 입력을 수집하기 위해 대화 상자 윈도우를 표시하는 경우 사용자가 대화 상자를 닫을 때까지 기본 윈도우의 표시를 흐리게 나타낼 수 있습니다. 이와 마찬가지로 윈도우를 닫을 때 표시를 점차 사라지게 할 수도 있습니다.

고급 표시 속성은 다음과 같습니다.

속성	제한
alpha	HTML 내용의 가독성을 줄일 수 있습니다.
filters	HTML 윈도우에서 외부 효과가 윈도우 가장자리로 잘립니다.
graphics	그래픽 명령으로 그려진 모양이 기본 배경을 비롯한 HTML 내용 아래에 나타납니다. 그려진 모양이 표시되려면 paintsDefaultBackground 속성이 false여야 합니다.
opaqueBackground	기본 배경색을 변경하지 않습니다. 이 색상 레이어가 표시되려면 paintsDefaultBackground 속성이 false여야 합니다.
rotation	사각형 HTMLLoader 영역의 모서리가 윈도우 가장자리로 잘릴 수 있습니다. HTML 내용에 로드된 SWF 및 PDF 내용이 표시되지 않습니다.
scaleX, scaleY	비율 인수가 1보다 크면 렌더링된 표시가 픽셀화되어 나타날 수 있습니다. HTML 내용에 로드된 SWF 및 PDF 내용이 표시되지 않습니다.
transform	HTML 내용의 가독성을 줄일 수 있습니다. HTML 표시가 윈도우 가장자리로 잘릴 수 있습니다. 변환에 회전, 크기 조절 또는 기울기가 포함된 경우 HTML 내용에 로드된 SWF 및 PDF 내용이 표시되지 않습니다.

다음 예제에서는 전체 HTML 표시를 흐리게 나타내도록 filters 배열을 설정하는 방법을 보여 줍니다.

```
 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 내용 스크롤

Adobe AIR 1.0 이상

HTMLLoader 클래스에는 HTML 내용의 스크롤을 제어하는 데 사용할 수 있는 다음과 같은 속성이 포함되어 있습니다.

속성	설명
contentHeight	HTML 내용의 높이(픽셀 단위)입니다.
contentWidth	HTML 내용의 폭(픽셀 단위)입니다.
scrollH	HTMLLoader 객체에 있는 HTML 내용의 가로 스크롤 위치입니다.
scrollV	HTMLLoader 객체에 있는 HTML 내용의 세로 스크롤 위치입니다.

다음 코드에서는 HTML 내용이 페이지 맨 아래로 스크롤되도록 scrollV 속성을 설정합니다.

```
 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;
}
```

HTMLLoader에는 가로 및 세로 스크롤 막대가 포함되어 있지 않습니다. Flex 구성 요소를 사용하거나 ActionScript에서 스크롤 막대를 구현할 수 있습니다. Flex HTML 구성 요소에는 HTML 내용에 대한 스크롤 막대가 자동으로 포함됩니다. 또한 HTMLLoader.createRootWindow() 메서드를 사용하여 스크롤 막대와 함께 HTMLLoader 객체가 포함된 윈도우를 만들 수 있습니다(925페이지의 “스크롤하는 HTML 내용으로 윈도우 만들기” 참조).

HTML 작업 내역 목록 액세스

Adobe AIR 1.0 이상

새 페이지가 HTMLLoader 객체에 로드되면 런타임은 객체에 대한 작업 내역 목록을 유지 관리합니다. 작업 내역 목록은 HTML 페이지의 window.history 객체에 해당합니다. HTMLLoader 클래스에는 HTML 작업 내역 목록으로 작업하는 데 사용할 수 있는 다음과 같은 속성과 메서드가 포함되어 있습니다.

클래스 멤버	설명
historyLength	앞뒤 항목을 비롯한 작업 내역 목록의 전체 길이입니다.
historyPosition	작업 내역 목록에서의 현재 위치입니다. 이 위치 앞의 작업 내역 항목은 "뒤로" 이동을 나타내고, 이 위치 뒤의 항목은 "앞으로" 이동을 나타냅니다.
getHistoryAt()	작업 내역 목록의 지정된 위치에서 작업 내역 항목에 해당하는 URLRequest 객체를 반환합니다.
historyBack()	가능한 경우 작업 내역 목록에서 뒤로 이동합니다.
historyForward()	가능한 경우 작업 내역 목록에서 앞으로 이동합니다.
historyGo()	브라우저 작업 내역에서 지정된 수의 단계를 이동합니다. 양수이면 앞으로 이동하고 음수이면 뒤로 이동합니다. 0으로 이동하면 페이지가 다시 로드됩니다. 끝을 넘는 위치를 지정하면 목록의 끝으로 이동합니다.

작업 내역 목록에 있는 항목은 HTMLHistoryItem 유형의 객체로 저장됩니다. HTMLHistoryItem 클래스에는 다음과 같은 속성이 있습니다.

속성	설명
isPost	HTML 페이지에 POST 데이터가 포함되어 있으면 true로 설정합니다.
originalUrl	리디렉션 전 HTML 페이지의 원래 URL입니다.
title	HTML 페이지의 제목입니다.
url	HTML 페이지의 URL입니다.

HTML 내용을 로드할 때 사용된 사용자 에이전트 설정

Adobe AIR 1.0 이상

HTMLLoader 클래스에는 HTMLLoader에서 사용하는 사용자 에이전트 문자열을 설정하는 데 사용할 수 있는 userAgent 속성이 있습니다. load() 메서드를 호출하기 전에 HTMLLoader 객체의 userAgent 속성을 설정합니다. HTMLLoader 인스턴스에서 이 속성을 설정하면 load() 메서드에 전달되는 URLRequest의 userAgent 속성이 사용되지 않습니다.

URLRequestDefaults.userAgent 속성을 설정하여 응용 프로그램 도메인에 있는 모든 HTMLLoader 객체가 사용하는 기본 사용자 에이전트 문자열을 설정할 수 있습니다. 정적 URLRequestDefaults 속성은 HTMLLoader 객체의 load() 메서드에 사용되는 URLRequest뿐만 아니라 모든 URLRequest 객체에 대한 기본값으로 적용됩니다. HTMLLoader의 userAgent 속성을 설정하면 기본 URLRequestDefaults.userAgent 설정이 재정의됩니다.

HTMLLoader 객체의 userAgent 속성이나 URLRequestDefaults.userAgent에 대한 사용자 에이전트 값을 설정하지 않으면 기본 AIR 사용자 에이전트 값이 사용됩니다. 이 기본값은 다음 두 예제와 같이 런타임 운영 체제(예: Mac OS 또는 Windows), 런타임 언어 및 런타임 버전에 따라 다릅니다.

- "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"

HTML 내용에 사용할 문자 인코딩 설정

Adobe AIR 1.0 이상

HTML 페이지에서는 다음과 같이 meta 태그를 포함하여 사용할 문자 인코딩을 지정할 수 있습니다.

```
meta http-equiv="content-type" content="text/html" charset="ISO-8859-1";
```

HTMLLoader 객체의 textEncodingOverride 속성을 설정하여 특정 문자 인코딩이 사용되도록 페이지 설정을 재정의합니다.

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingOverride = "ISO-8859-1";
```

HTML 페이지에서 HTMLLoader 객체의 textEncodingFallback 속성을 사용하여 설정을 지정하지 않는 경우 사용할 HTMLLoader 내용의 문자 인코딩을 지정합니다.

```
var html:HTMLLoader = new HTMLLoader();
html.textEncodingFallback = "ISO-8859-1";
```

textEncodingOverride 속성은 HTML 페이지의 설정을 재정의합니다. 또한 textEncodingOverride 속성과 HTML 페이지의 설정은 textEncodingFallback 속성을 재정의합니다.

HTML 내용을 로드하기 전에 textEncodingOverride 속성이나 textEncodingFallback 속성을 설정합니다.

HTML 내용에 대해 브라우저와 유사한 사용자 인터페이스 정의

Adobe AIR 1.0 이상

JavaScript에서는 HTML 내용을 표시하는 윈도우를 제어하기 위한 몇 가지 API를 제공합니다. AIR에서 사용자 정의 HTMLHost 클래스를 구현하여 이러한 API를 재정의할 수 있습니다.

HTMLHost 클래스 확장

Adobe AIR 1.0 이상

예를 들어, 응용 프로그램이 탭 인터페이스에서 여러 HTMLLoader 객체를 제공하는 경우 로드된 HTML 페이지에서 기본 윈도우의 제목이 아니라 탭의 레이블을 변경하게 할 수 있습니다. 마찬가지로 코드에서 부모 표시 객체 컨테이너에 있는 HTMLLoader 객체의 위치를 변경하거나, HTMLLoader 객체가 포함된 윈도우를 이동하거나, 아무 작업도 수행하지 않거나, 다른 작업을 전체적으로 수행하여 window.moveTo() 호출에 응답할 수 있습니다.

AIR HTMLHost 클래스는 다음과 같은 JavaScript 속성과 메서드를 제어합니다.

- window.status
- window.document.title
- window.location
- window.blur()
- window.close()
- window.focus()
- window.moveBy()
- window.moveTo()
- window.open()
- window.resizeBy()
- window.resizeTo()

new HTMLLoader()를 사용하여 HTMLLoader 객체를 만드는 경우 나열된 JavaScript 속성이나 메서드가 사용되지 않습니다. HTMLHost 클래스는 브라우저와 유사한 이러한 JavaScript API의 기본 구현을 제공합니다. HTMLHost 클래스를 확장하여 비헤이비어를 사용자 정의할 수도 있습니다. 기본 비헤이비어를 지원하는 HTMLHost 객체를 만들려면 HTMLHost 생성자에서 defaultBehaviors 매개 변수를 true로 설정합니다.

```
var defaultHost:HTMLHost = new HTMLHost(true);
```

HTMLLoader 클래스의 createRootWindow() 메서드를 사용하여 AIR에서 HTML 윈도우를 만드는 경우 기본 비헤이비어를 지원하는 HTMLHost 인스턴스가 자동으로 할당됩니다. 다른 HTMLHost 구현을 HTMLLoader의 htmlHost 속성에 할당하여 호스트 객체 비헤이비어를 변경하거나, null을 할당하여 기능을 전체적으로 비활성화할 수 있습니다.

참고: AIR에서는 HTML 기반 AIR 응용 프로그램용으로 만들어진 초기 윈도우와 JavaScript window.open() 메서드의 기본 구현으로 만들어진 모든 윈도우에 기본 HTMLHost 객체를 할당합니다.

예제: HTMLHost 클래스 확장

Adobe AIR 1.0 이상

다음 예제에서는 HTMLHost 클래스를 확장하여 HTMLLoader 객체가 사용자 인터페이스에 영향을 주는 방식을 사용자 정의하는 방법을 보여 줍니다.

Flex 예제:

1. HTMLHost 클래스를 확장하는 클래스(하위 클래스)를 만듭니다.
2. 사용자 인터페이스 관련 설정의 변경을 처리하도록 새 클래스의 메서드를 재정의합니다. 예를 들어, 아래의 클래스 CustomHost는 window.open() 호출과 window.document.title 변경에 대한 비헤이비어를 정의합니다. window.open()을 호출하면 새 윈도우에서 HTML 페이지가 열리고 window.document.title(HTML 페이지의 <title> 요소 설정 포함)이 변경되면 해당 윈도우의 제목이 설정됩니다.

```
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. HTMLLoader가 포함된 코드(HTMLHost의 새 하위 클래스에 대한 코드 제외)에서 새 클래스의 객체를 만듭니다. 새 객체를 HTMLLoader의 htmlHost 속성에 할당합니다. 다음 Flex 코드에서는 이전 단계에서 정의된 CustomHost 클래스를 사용합니다.

```
<?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>
```

여기에서 설명하는 코드를 테스트하려면 응용 프로그램 디렉토리에 다음 내용이 포함된 HTML 파일을 넣습니다.

```
<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 예제:

1. AIR용 Flash 파일을 만듭니다. 문서 클래스를 CustomHostExample로 설정한 다음 파일을 CustomHostExample.fla로 저장합니다.
2. HTMLHost 클래스를 확장하는 클래스(하위 클래스)가 포함된 CustomHost.as라는 ActionScript 파일을 만듭니다. 이 클래스는 사용자 인터페이스 관련 설정의 변경을 처리하도록 새 클래스의 특정 메서드를 재정의합니다. 예를 들어 아래의 클래스 CustomHost는 window.open() 호출과 window.document.title 변경에 대한 비헤이비어를 정의합니다. window.open() 메서드를 호출하면 새 윈도우에서 HTML 페이지가 열리고 window.document.title 속성(HTML 페이지의 <title> 요소 설정 포함)이 변경되면 해당 윈도우의 제목이 설정됩니다.

```
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 응용 프로그램의 문서 클래스를 포함할 CustomHostExample.as라는 ActionScript 파일을 추가로 만듭니다. 이 클래스는 HTMLLoader 객체를 만들고 호스트 속성을 이전 단계에서 정의된 CustomHost 클래스의 인스턴스로 설정합니다.

```
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;
        }
    }
}
```

여기에서 설명하는 코드를 테스트하려면 응용 프로그램 디렉토리에 다음 내용이 포함된 HTML 파일을 넣습니다.

```
<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>
```

window.location 속성의 변경 처리

Adobe AIR 1.0 이상

HTML 페이지의 URL 변경을 처리하도록 locationChange() 메서드를 재정의합니다. locationChange() 메서드는 페이지의 JavaScript에서 window.location의 값을 변경할 때 호출됩니다. 다음 예제에서는 요청된 URL을 로드합니다.

```
override public function updateLocation(locationURL:String):void
{
    htmlLoader.load(new URLRequest(locationURL));
}
```

참고: HTMLHost 객체의 htmlLoader 속성을 사용하여 현재 HTMLLoader 객체를 참조할 수 있습니다.

window.moveBy(), window.moveTo(), window.resizeTo(), window.resizeBy()에 대한 JavaScript 호출 처리

Adobe AIR 1.0 이상

HTML 내용 경계의 변경을 처리하도록 set windowRect() 메서드를 재정의합니다. set windowRect() 메서드는 페이지의 JavaScript에서 window.moveBy(), window.moveTo(), window.resizeTo() 또는 window.resizeBy()를 호출할 때 호출됩니다. 다음 예제에서는 데스크톱 윈도우의 경계를 업데이트합니다.

```
override public function set windowRect(value:Rectangle):void
{
    htmlLoader.stage.nativeWindow.bounds = value;
}
```

window.open()에 대한 JavaScript 호출 처리

Adobe AIR 1.0 이상

window.open()에 대한 JavaScript 호출을 처리하도록 createWindow() 메서드를 재정의합니다. createWindow() 메서드의 구현은 새 HTMLLoader 객체를 만들고 반환하는 작업을 담당합니다. 일반적으로 HTMLLoader를 새 윈도우에 표시하지만 윈도우를 만들 필요는 없습니다.

다음 예제에서는 HTMLLoader.createRootWindow()를 사용하여 윈도우와 HTMLLoader 객체를 만드는 createWindow() 함수를 구현하는 방법을 보여 줍니다. NativeWindow 객체를 별도로 만들고 HTMLLoader를 윈도우 스테이지에 추가할 수도 있습니다.

```
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;
}
```

참고: 이 예제에서는 window.open()을 사용하여 만든 새 윈도우에 사용자 정의 HTMLHost 구현을 할당합니다. 다른 구현을 사용하거나 새 윈도우의 htmlHost 속성을 null로 설정할 수도 있습니다.

createWindow() 메서드에 매개 변수로 전달되는 객체는 HTMLWindowCreateOptions 객체입니다. HTMLWindowCreateOptions 클래스에는 window.open() 호출에서 features 매개 변수 문자열에 설정된 값을 보고하는 속성이 포함되어 있습니다.

HTMLWindowCreateOptions 속성	window.open()에 대한 JavaScript 호출에서 기능 문자열의 해당 설정
fullscreen	fullscreen
height	height
locationBarVisible	location
menuBarVisible	menubar
resizeable	resizable
scrollBarsVisible	scrollbars
statusBarVisible	status
toolBarVisible	toolbar
width	width
x	left 또는 screenX
y	top 또는 screenY

HTMLLoader 클래스가 기능 문자열에 지정할 수 있는 기능을 모두 구현하는 것은 아닙니다. 적절한 경우 응용 프로그램에서 스크롤 막대, 위치 막대, 메뉴 모음, 상태 표시줄 및 툴바를 제공해야 합니다.

JavaScript window.open() 메서드의 다른 인수는 시스템에서 처리됩니다. createWindow() 구현은 HTMLLoader 객체에 내용을 로드하거나 윈도우 제목을 설정하면 안 됩니다.

window.close()에 대한 JavaScript 호출 처리

Adobe AIR 1.0 이상

window.close() 메서드에 대한 JavaScript 호출을 처리하도록 windowClose()를 재정의합니다. 다음 예제에서는 window.close() 메서드가 호출될 때 데스크톱 윈도우를 닫습니다.

```
override public function windowClose():void
{
    htmlLoader.stage.nativeWindow.close();
}
```

window.close()에 대한 JavaScript 호출에서는 포함하는 윈도우를 닫을 필요가 없습니다. 예를 들어, 다음 코드와 같이 HTMLLoader를 표시 목록에서 제거하여 다른 내용이 있을 수 있는 윈도우를 열어 둘 수 있습니다.

```
override public function windowClose():void
{
    htmlLoader.parent.removeChild(htmlLoader);
}
```

window.status 속성의 변경 처리

Adobe AIR 1.0 이상

window.status의 값에 대한 JavaScript 변경을 처리하도록 updateStatus() 메서드를 재정의합니다. 다음 예제에서는 상태 값을 추적합니다.

```
override public function updateStatus(status:String):void
{
    trace(status);
}
```

요청된 상태는 updateStatus() 메서드에 문자열로 전달됩니다.

HTMLLoader 객체는 상태 표시줄을 제공하지 않습니다.

window.document.title 속성의 변경 처리

Adobe AIR 1.0 이상

window.document.title 값의 JavaScript 변경을 처리하도록 updateTitle() 메서드를 재정의합니다. 다음 예제에서는 윈도우 제목을 변경하고 "Sample"이라는 문자열을 제목에 추가합니다.

```
override public function updateTitle(title:String):void
{
    htmlLoader.stage.nativeWindow.title = title + " - Sample";
}
```

document.title이 HTML 페이지에서 설정되면 요청된 제목이 updateTitle() 메서드에 문자열로 전달됩니다.

document.title이 변경될 때 HTMLLoader 객체가 포함된 윈도우의 제목이 반드시 변경되지는 않습니다. 예를 들어, 텍스트 필드 등의 다른 인터페이스 요소를 변경할 수 있습니다.

window.blur() 및 window.focus()에 대한 JavaScript 호출 처리

Adobe AIR 1.0 이상

다음 예제와 같이 window.blur() 및 window.focus()에 대한 JavaScript 호출을 처리하도록 windowBlur() 및 windowFocus() 메서드를 재정의합니다.

```
 override public function windowBlur():void
{
    htmlLoader.alpha = 0.5;
}
override public function windowFocus():void
{
    htmlLoader.alpha = 1.0;
    NativeApplication.nativeApplication.activate(htmlLoader.stage.nativeWindow);
}
```

참고: AIR에서는 윈도우나 응용 프로그램을 비활성화하기 위한 API를 제공하지 않습니다.

스크롤하는 HTML 내용으로 윈도우 만들기

Adobe AIR 1.0 이상

HTMLLoader 클래스에는 정적 메서드 HTMLLoader.createRootWindow()가 포함되어 있습니다. 이 메서드를 사용하면 HTMLLoader 객체가 포함된 새 윈도우(NativeWindow 객체로 나타냄)를 열고 이 윈도우에 대한 일부 사용자 인터페이스 설정을 정의할 수 있습니다. 이 메서드는 사용자 인터페이스를 정의할 수 있는 네 가지 매개 변수를 사용합니다.

매개 변수	설명
visible	윈도우가 처음에 표시되는지(true) 아니면 표시되지 않는지(false)를 지정하는 부울 값
windowInitOptions	NativeWindowInitOptions 객체입니다. NativeWindowInitOptions 클래스는 윈도우의 최소화, 최대화 또는 크기 조절 가능 여부, 윈도우에 시스템 크롬이 있는지, 아니면 사용자 정의 크롬이 있는지 여부, 윈도우가 투명한지 여부(시스템 크롬을 사용하지 않는 윈도우의 경우) 및 윈도우 유형과 같은 NativeWindow 객체에 대한 초기화 옵션을 정의합니다.
scrollBarsVisible	스크롤 막대가 있는지(true) 아니면 없는지(false)를 지정하는 부울 값
bounds	새 윈도우의 위치와 크기를 정의하는 Rectangle 객체입니다.

예를 들어 다음 코드에서는 HTMLLoader.createRootWindow() 메서드를 사용하여 스크롤 막대를 사용하는 HTMLLoader 내용으로 윈도우를 만듭니다.

```
 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();
```

참고: JavaScript에서 createRootWindow()를 직접 호출하여 만든 윈도우는 여는 HTML 윈도우와 독립적으로 유지됩니다. 예를 들어, JavaScript Window opener 및 parent 속성은 null입니다. 그러나 createRootWindow()를 호출하도록 HTMLHost createWindow() 메서드를 재정의하여 createRootWindow()를 간접적으로 호출하면 opener 및 parent가 여는 HTML 윈도우를 참조합니다.

HTMLLoader 클래스의 하위 클래스 만들기

Adobe AIR 1.0 이상

HTMLLoader 클래스의 하위 클래스를 만들어 새 비헤이비어를 만들 수 있습니다. 예를 들어, HTML이 렌더링되거나 사용자가 링크를 클릭할 때 전달되는 이벤트와 같은 HTMLLoader 이벤트에 대한 기본 이벤트 리스너를 정의하는 하위 클래스를 만들 수 있습니다.

다음 예제에서는 JavaScript window.open() 메서드가 호출될 때 일반 비헤이비어를 제공하도록 HTMLHost 클래스를 확장합니다. 그런 다음 사용자 정의 HTMLHost 구현 클래스를 사용하는 HTMLLoader의 하위 클래스를 정의합니다.

```
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
       }
}
```

다음 예제에서는 MyHTMLHost 객체를 htmlHost 속성에 할당하는 HTMLLoader 클래스의 하위 클래스를 정의합니다.

```
package
{
       import flash.html.HTMLLoader;
    import MyHTMLHost;
    import HTMLLoader;
       public class MyHTML extends HTMLLoader
       {
           public function MyHTML()
           {
               super();
               htmlHost = new MyHTMLHost();
           }
       }
}
```

이 예제에서 사용된 HTMLHost 클래스와 HTMLLoader.createRootWindow() 메서드에 대한 자세한 내용은 917페이지의 “HTML 내용에 대해 브라우저와 유사한 사용자 인터페이스 정의”를 참조하십시오.

61장: AIR에서 HTML 관련 이벤트 처리

Adobe AIR 1.0 이상

이벤트 처리 시스템을 사용하면 프로그래머가 편리한 방식으로 사용자 입력 및 시스템 이벤트에 응답할 수 있습니다. Adobe® AIR® 이벤트 모델은 편리할 뿐만 아니라 표준과 호환됩니다. 이벤트 모델은 산업 표준 이벤트 처리 아키텍처인 DOM(Document Object Model) 레벨 3 이벤트 사양을 기초로 프로그래머를 위해 강력하고 직관적인 이벤트 처리 도구를 제공합니다.

HTMLLoader 이벤트

Adobe AIR 1.0 이상

HTMLLoader 객체는 다음 Adobe® ActionScript® 3.0 이벤트를 전달합니다.

이벤트	설명
htmlDOMInitialize	스크립트가 파싱되기 전이나 페이지에 DOM 노드가 추가되기 전에 HTML 문서가 생성될 때 전달됩니다.
complete	HTML 페이지의 onload 이벤트 직후 로드 작업에 대한 응답으로 HTML DOM이 생성되었을 때 전달됩니다.
htmlBoundsChanged	contentWidth 및 contentHeight 속성 중 하나 또는 둘 다 변경된 경우 전달됩니다.
locationChange	HTMLLoader의 location 속성이 변경되었을 때 전달됩니다.
locationChanging	사용자 탐색, JavaScript 호출 또는 리디렉션으로 인해 HTMLLoader의 위치가 변경되기 전에 전달됩니다. load(), loadString(), reload(), historyGo(), historyForward() 또는 historyBack() 메서드를 호출하는 경우에는 locationChanging 이벤트가 전달되지 않습니다. 전달된 이벤트 객체의 preventDefault() 메서드를 호출하면 탐색이 취소됩니다. 링크가 시스템 브라우저에서 열려 있으면 HTMLLoader가 위치를 변경하지 않기 때문에 locationChanging 이벤트가 전달되지 않습니다.
scroll	HTML 엔진이 스크롤 위치를 변경할 때 언제든지 전달됩니다. Scroll 이벤트는 페이지의 앵커 링크(# 링크)로의 이동이나 window.scrollTo() 메서드에 대한 호출로 발생할 수 있습니다. 텍스트 입력이나 텍스트 영역에 텍스트를 입력해도 scroll 이벤트가 발생할 수 있습니다.
uncaughtScriptException	HTMLLoader에서 JavaScript 예외가 발생하였는데 해당 예외가 JavaScript 코드에서 catch되지 않은 경우 전달됩니다.

JavaScript 이벤트에 대한 ActionScript 함수(예: onClick)를 등록할 수도 있습니다. 자세한 내용은 927페이지의 “ActionScript를 사용한 DOM 이벤트 처리”를 참조하십시오.

ActionScript를 사용한 DOM 이벤트 처리

Adobe AIR 1.0 이상

ActionScript 함수를 등록하여 JavaScript 이벤트에 응답할 수 있습니다. 예를 들어, 다음과 같은 HTML 내용을 가정합니다.

```
<html>
<body>
    <a href="#" id="testLink">Click me.</a>
</html>
```

페이지의 모든 이벤트에 대한 핸들러로서 ActionScript 함수를 등록할 수 있습니다. 예를 들어, 다음 코드는 HTML 페이지에 있는 testLink 요소의 onclick 이벤트에 대한 수신기로서 clickHandler() 함수를 추가합니다.

```
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 );
}
```

전달된 이벤트 객체가 flash.events.Event 유형이나 Event 하위 클래스 중 하나가 아닙니다. Object 클래스를 사용하여 이벤트 핸들러 함수 인수의 유형을 선언하십시오.

addEventListener() 메서드를 사용하여 이러한 이벤트에 대해 등록할 수도 있습니다. 예를 들어, 이전 예제의 completeHandler() 메서드를 다음 코드로 바꿀 수 있습니다.

```
function completeHandler(event:Event):void {
    var testLink:Object = html.window.document.getElementById("testLink");
    testLink.addEventListener("click", clickHandler);
}
```

수신기가 특정 DOM 요소를 참조하는 경우 이벤트 수신기를 추가하기 전에 HTMLLoader가 complete 이벤트를 전달하기를 대기하는 것이 좋습니다. HTML 페이지는 여러 파일을 로드하는 경우가 많으며 모든 파일이 로드되어 파싱되어야 HTML DOM이 완전히 만들어집니다. HTMLLoader는 모든 요소가 만들어졌을 때 complete 이벤트를 전달합니다.

catch되지 않은 JavaScript 예외에 대한 응답

Adobe AIR 1.0 이상

다음과 같은 HTML을 가정합니다.

```
<html>
<head>
    <script>
        function throwError() {
            var x = 400 * melbaToast;
        }
    </script>
</head>
<body>
    <a href="#" onclick="throwError()">Click me.</a>
</html>
```

알 수 없는 변수인 melbaToast를 참조하는 JavaScript 함수인 throwError()가 포함되어 있습니다.

```
var x = 400 * melbaToast;
```

JavaScript 작업에서 try/catch 구조를 사용하여 JavaScript 코드에서 catch하지 않은 잘못된 작업을 만나면 해당 페이지가 포함된 HTMLLoader 객체가 HTMLUncaughtScriptExceptionEvent 이벤트를 전달합니다. 다음 코드와 같이 이 이벤트에 대한 핸들러를 등록할 수 있습니다.

```
 var html:HTMLLoader = new HTMLLoader();
var urlReq:URLRequest = new URLRequest("test.html");
html.load(urlReq);
html.width = container.width;
html.height = container.height;
container.addChild(html);
html.addEventListener(HTMLUncaughtScriptExceptionEvent.UNCAUGHT_SCRIPT_EXCEPTION,
                          htmlErrorHandler);
function htmlErrorHandler(event:HTMLUncaughtJavaScriptExceptionEvent):void
{
    event.preventDefault();
    trace("exceptionValue:", event.exceptionValue)
    for (var i:int = 0; i < event.stackTrace.length; i++)
    {
        trace("sourceURL:", event.stackTrace[i].sourceURL);
        trace("line:", event.stackTrace[i].line);
        trace("function:", event.stackTrace[i].functionName);
    }
}
```

JavaScript 내에서 window.htmlLoader 속성을 사용하여 같은 이벤트를 처리할 수 있습니다.

```
 <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>
```

htmlErrorHandler() 이벤트 핸들러는 JavaScript 오류 메시지를 AIR 추적 출력으로 보내는 이벤트의 기본 비헤이비어를 취소하고 고유한 출력 메시지를 생성합니다. HTMLUncaughtScriptExceptionEvent 객체의 exceptionValue 값을 출력합니다. 그리고 stackTrace 배열에 있는 각 객체의 속성을 출력합니다.

```
exceptionValue: ReferenceError: Can't find variable: melbaToast
sourceURL: app:/test.html
line: 5
function: throwError
sourceURL: app:/test.html
line: 10
function: onclick
```

JavaScript에서 런타임 이벤트 처리

Adobe AIR 1.0 이상

런타임 클래스는 addEventListener() 메서드를 사용한 이벤트 핸들러 추가를 지원합니다. 한 이벤트에 대한 핸들러 함수를 추가하려면 이벤트를 전달하는 객체의 addEventListener() 메서드를 호출하며 이벤트 유형과 처리 함수를 제공합니다. 예를 들어, 사용자가 제목 표시줄에서 윈도우 닫기 버튼을 클릭할 때 전달되는 closing 이벤트를 수신하려면 다음 문을 사용합니다.

```
window.nativeWindow.addEventListener(air.NativeWindow.CLOSING, handleWindowClosing);
```

이벤트 핸들러 함수 만들기

Adobe AIR 1.0 이상

다음 코드에서는 기본 윈도우의 위치에 대한 정보를 표시하는 간단한 HTML 파일을 만듭니다. moveHandler()라는 핸들러 함수는 기본 윈도우의 NativeWindowBoundsEvent 클래스에서 정의한 move 이벤트를 수신합니다.

```
<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>
```

사용자가 윈도우를 이동할 때 textarea 요소는 윈도우의 업데이트된 X 및 Y 위치를 표시합니다.

이벤트 객체는 moveHandler() 메서드에 인수로 전달됩니다. 이벤트 매개 변수를 통해 핸들러 함수에서 이벤트 객체를 검토할 수 있습니다. 이 예제에서는 이벤트 객체의 type 속성을 사용하여 이벤트가 move 이벤트인지 확인합니다.

이벤트 리스너 제거

Adobe AIR 1.0 이상

removeEventListener() 메서드를 사용하면 더 이상 필요하지 않은 이벤트 리스너를 제거할 수 있습니다. 더 이상 사용하지 않을 모든 리스너를 제거하는 것이 좋습니다. 이 메서드를 사용하는 경우에는 addEventListener() 메서드를 사용할 때와 마찬가지로 eventName 및 listener 매개 변수가 반드시 필요합니다.

탐색하는 HTML 페이지의 이벤트 리스너 제거

Adobe AIR 1.0 이상

HTML 내용이 이동하거나 HTML 내용이 포함하는 윈도우가 닫히기 때문에 해당 내용이 버려지는 경우 언로드된 페이지의 객체를 참조하는 이벤트 리스너는 자동으로 제거되지 않습니다. 한 객체가 이미 언로드된 핸들러에 이벤트를 전달하는 경우 "응용 프로그램에서 더 이상 로드되지 않는 HTML 페이지의 JavaScript 객체를 참조하려고 시도했습니다."라는 오류 메시지가 표시됩니다.

이 오류를 피하려면 HTML 페이지가 없어지기 전에 해당 페이지에서 JavaScript 이벤트 리스너를 제거합니다. HTMLLoader 객체 내에서 페이지를 이동하는 경우 window 객체의 unload 이벤트 중에 이벤트 리스너를 제거합니다.

예를 들어, 다음 JavaScript 코드는 uncaughtScriptException 이벤트에 대한 이벤트 리스너를 제거합니다.

```
window.onunload = cleanup;
window.htmlLoader.addEventListener('uncaughtScriptException', uncaughtScriptException);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

HTML 내용이 포함된 윈도우를 닫을 때 오류가 발생하지 않게 하려면 NativeWindow 객체(window.nativeWindow)의 closing 이벤트에 대한 응답으로 cleanup 함수를 호출합니다. 예를 들어 다음 JavaScript 코드는 uncaughtScriptException 이벤트에 대한 이벤트 리스너를 제거합니다.

```
 window.nativeWindow.addEventListener(air.Event.CLOSING, cleanup);
function cleanup()
{
    window.htmlLoader.removeEventListener('uncaughtScriptException',
                            uncaughtScriptExceptionHandler);
}
```

이벤트 리스너가 실행되는 즉시 제거함으로써 이 오류가 발생하는 것을 막을 수도 있습니다(이벤트를 한 번만 처리하면 되는 경우). 예를 들어, 다음 JavaScript 코드는 HTMLLoader 클래스의 createRootWindow() 메서드를 호출하여 html 윈도우를 만들고 complete 이벤트에 대한 이벤트 리스너를 추가합니다. complete 이벤트 리스너를 호출하면 removeEventListener() 함수를 사용하여 고유 이벤트 리스너를 제거합니다.

```
 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"));
```

사용되지 않는 이벤트 리스너를 제거하면 시스템 가비지 수집기에서 해당 리스너와 연결된 메모리를 모두 회수할 수 있습니다.

기존 이벤트 리스너 확인

Adobe AIR 1.0 이상

hasEventListener() 메서드를 사용하여 객체의 이벤트 리스너에 대한 존재를 확인할 수 있습니다.

62장: 모바일 응용 프로그램에서 HTML 내용 표시

Adobe AIR 2.5 이상

StageWebView 클래스는 휴대 장치에서는 시스템 브라우저 컨트롤을, 데스크톱 컴퓨터에서는 표준 Adobe® AIR® HTMLLoader 컨트롤을 사용하여 HTML 내용을 표시합니다. StageWebView.isSupported 속성을 검사하여 이 클래스가 현재의 장치에서 지원되는지 확인할 수 있습니다. 휴대 장치 프로파일의 모든 장치에서 이 클래스가 지원되는 것은 아닙니다.

모든 프로파일에서 StageWebView 클래스는 응용 프로그램의 HTML 내용과 나머지 내용 간의 제한된 상호 작용만 지원합니다. 탐색을 제어할 수는 있지만 데이터의 크로스 스크립팅 또는 직접 교환은 허용되지 않습니다. 내용을 로컬 또는 원격 URL로부터 로드하거나 HTML 문자열로 전달할 수 있습니다.

StageWebView 객체

StageWebView 객체는 표시 객체가 아니며 표시 목록에 추가될 수 없습니다. 대신 이 객체는 스테이지에 직접 연결된 뷰포트로 작동합니다. StageWebView 내용은 표시 목록 내용 위에 그려지며, 여러 개의 StageWebView 객체가 그려질 때 그 순서를 제어할 수는 없습니다.

StageWebView 객체를 표시하려면 객체가 나타나는 스테이지를 StageWebView의 stage 속성에 할당해야 합니다. viewPort 속성을 사용하면 표시 크기를 설정할 수 있습니다.

viewPort 속성의 x 및 y 좌표의 범위는 -8192~8191이며, 스테이지의 폭 및 높이의 최대 값은 8191입니다. 설정된 크기가 최대 값을 초과하면 예외가 발생합니다.

다음 예제에서는 StageWebView 객체를 만들고 stage 및 viewPort 속성을 설정하여 HTML 문자열을 표시합니다.

```
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 );
```

StageWebView 객체를 숨기려면 stage 속성을 null로 설정하고, 객체를 완전히 제거하려면 dispose() 메서드를 호출하십시오. dispose()를 호출하는 것은 선택 사항이지만 이 메서드를 호출하면 가비지 수집기가 객체에 사용된 메모리를 더 빠르게 회수할 수 있습니다.

내용

loadURL() 및 loadString() 메서드를 사용하면 내용을 StageWebView 객체 안으로 로드할 수 있습니다.

loadURL() 메서드는 지정된 URL에 있는 리소스를 로드합니다. data:, file:, http:, https:, javascript: 등 시스템 웹 브라우저 컨트롤에서 지원하는 모든 URI 스킴을 사용할 수 있으며, app: 및 app-storage: 스킴은 지원되지 않습니다. AIR는 URL 문자열의 유효성을 검사하지 않습니다.

loadString() 메서드는 HTML 내용이 포함된 리터럴 문자열을 로드합니다. 이 메서드로 로드된 페이지의 위치는 다음과 같이 표현됩니다.

- 데스크톱의 경우: about:blank

- iOS의 경우: **htmlString**
- Android의 경우: 인코딩된 **htmlString**의 데이터 URI 포맷

URI 스킴은 포함된 내용 또는 데이터를 로드하기 위한 규칙을 결정합니다.

URI 스킴	로컬 리소스 로드	원격 리소스 로드	로컬 XMLHttpRequest	원격 XMLHttpRequest
data:	아니요	예	아니요	아니요
file:	예	예	예	예
http:, https:	아니요	예	아니요	동일한 도메인의 경우
about: (loadString() method)	아니요	예	아니요	아니요

참고: 스테이지의 displayState 속성이 FULL_SCREEN으로 설정된 경우 데스크톱에서 StageWebView에 표시된 텍스트 필드에 값을 입력할 수 없습니다. 그러나 iOS 및 Android에서는 스테이지의 displayState가 FULL_SCREEN으로 설정되어 있더라도 StageWebView의 텍스트 필드에 값을 입력할 수 있습니다.

다음 예제에서는 StageWebView 객체를 사용하여 Adobe 웹 사이트를 표시합니다.

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

Android 장치의 응용 프로그램에서 원격 리소스를 성공적으로 로드하려면 Android INTERNET 권한을 지정해야 합니다.

Android 3.0 이상의 경우 StageWebView 객체의 플러그인 내용을 표시하려면 응용 프로그램이 AIR 응용 프로그램 설명자의 Android manifestAdditions 요소에서 하드웨어 가속화를 활성화해야 합니다. Flash Player 및 StageWebView 객체의 다른 플러그인 활성화를 참조하십시오.

JavaScript URI

JavaScript URI를 사용하면 StageWebView 객체에서 로드하는 HTML 페이지에 정의된 함수를 호출할 수 있습니다. JavaScript URI를 사용하여 호출한 함수는 로드되는 웹 페이지의 컨텍스트에서 실행됩니다. 다음 예제에서는 StageWebView 객체를 사용하여 JavaScript 함수를 호출합니다.

```
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 );
        }
    }
}
```

기타 도움말 항목

Sean Voisen: StageWebView의 효과적인 활용

탐색 이벤트

사용자가 HTML의 링크를 클릭하면 StageWebView 객체는 locationChanging 이벤트를 전달합니다. 이벤트 객체의 preventDefault() 메서드를 호출하면 탐색을 중단할 수 있습니다. 그러지 않으면 StageWebView 객체는 새 페이지로 이동하여 locationChange 이벤트를 전달합니다. 페이지 로드가 끝나면 StageWebView는 complete 이벤트를 전달합니다.

locationChanging 이벤트는 모든 HTML 리디렉션에서 전달되며, locationChange 및 complete 이벤트는 적절한 시기에 전달됩니다.

iOS에서 locationChanging 이벤트는 첫 번째 loadURL() 또는 loadString() 메서드를 제외하고 locationChange 이벤트에 앞서 전달됩니다. 또한 locationChange 이벤트는 iFrames 및 Frames에서 탐색 변경이 발생하는 경우에도 전달됩니다.

다음 예제에서는 위치 변경을 방지하고 대신 시스템 브라우저에서 새 페이지를 여는 방법을 보여 줍니다.

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

작업 내역

사용자가 StageWebView 객체에 표시된 내용에 있는 링크를 클릭하면 컨트롤에서 뒤로 이동 및 앞으로 이동 내역 스택을 저장합니다. 다음 예제에서는 이러한 두 개의 내역 스택을 탐색하는 방법을 보여 주며, 뒤로 소프트 키와 검색 소프트 키를 활용합니다.

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

포커스

StageWebView 클래스는 표시 객체는 아니지만 컨트롤 내부 또는 외부로의 포커스 전환을 관리할 수 있는 멤버를 포함하고 있습니다.

StageWebView 객체가 포커스를 얻으면 focusIn 이벤트가 전달됩니다. 이 이벤트를 활용하면 필요한 경우 응용 프로그램에서 포커스 요소를 관리할 수 있습니다.

StageWebView가 포커스를 잃으면 focusOut 이벤트가 전달됩니다. 사용자가 장치 트랙볼 또는 방향 화살표를 눌러 웹 페이지의 첫 번째 또는 마지막 컨트롤을 벗어날 경우 StageWebView 인스턴스는 포커스를 잃을 수 있습니다. 이벤트 객체의 direction 속성을 통해 포커스 흐름이 페이지 맨 위를 지나 올라가는지 또는 페이지 맨 아래를 지나 내려가는지 알아낼 수 있습니다. 이 정보를 활용하면 StageWebView 위쪽 또는 아래쪽에 있는 적절한 표시 객체에 포커스를 할당할 수 있습니다.

iOS에서 포커스는 프로그래밍 방식으로 설정할 수 없습니다. StageWebView는 FocusEvent의 direction 속성이 none으로 설정된 상태에서 focusIn 및 focusOut 이벤트를 전달합니다. 사용자가 StageWebView의 내부를 터치하면 focusIn 이벤트가 전달되며, StageWebView의 외부를 터치하면 focusOut 이벤트가 전달됩니다.

다음 예제에서는 StageWebView 객체에서 Flash 표시 객체로 포커스가 전달되는 방식을 보여 줍니다.

```
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;
        }

    }
}
```

비트맵 캡처

StageWebView 객체는 모든 표시 목록 내용 위에 렌더링되며, StageWebView 객체 위에는 내용을 추가할 수 없습니다. 예를 들어 StageWebView 내용 위에서 드롭 다운을 확장할 수 없습니다. 이 문제를 해결하려면 StageWebView의 스냅샷을 캡처한 다음 StageWebView를 숨기고 대신 비트맵 스냅샷을 추가합니다.

다음 예제에서는 drawViewPortToBitmapData 메서드를 사용하여 StageWebView 객체의 스냅샷을 캡처하는 방법을 보여 줍니다. 이 예제에서는 스테이지를 null로 설정하여 StageWebView 객체를 숨깁니다. 웹 페이지가 완전히 로드되면 비트맵을 캡처 및 표시하는 함수가 호출됩니다. 코드를 실행하면 Google 및 Facebook 두 개의 레이블이 표시됩니다. 각 레이블을 클릭하면 해당 웹 페이지가 캡처되어 스테이지에 스냅샷으로 표시됩니다.

```
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);
        }
    }
}
```

63장: 워커를 활용한 동시성 구현

데스크톱 플랫폼용 Flash Player 11.4 이상, Adobe AIR 13.4 이상

ActionScript 워커를 사용하면 코드를 동시에 실행할 수 있습니다. 즉, 기본 코드의 실행을 중단하지 않고 백그라운드에서 특정 코드를 실행할 수 있습니다.

ActionScript 동시성 API는 Flash Player 11.4 이상 및 AIR 3.4 이상이 설치된 데스크톱 플랫폼에서만 사용할 수 있습니다. ActionScript 동시성 API는 데스크톱 플랫폼에 설치된 Flash Player 11.4 이상 및 AIR 3.4 이상에서만 사용할 수 있습니다. 동시성 기능은 모바일 플랫폼용 AIR에서는 지원되지 않습니다.

워커 및 동시성에 대한 이해

데스크톱 플랫폼용 Flash Player 11.4 이상, Adobe AIR 13.4 이상

응용 프로그램에서 워커를 사용하지 않는 경우, 응용 프로그램의 코드는 실행 스레드라고 하는 단일 실행 단계 선형 블록에서 실행됩니다. 이 스레드에서는 개발자가 작성하는 코드를 실행합니다. 또한 이 스레드에서는 런타임에 속하는 많은 코드를 실행하며, 특히 표시 객체의 속성이 변경될 때 화면을 업데이트하는 코드를 주로 실행합니다. 코드는 메서드 및 클래스로서 일괄 작성되지만 런타임이 되면 마치 하나의 긴 일련의 단계로 작성된 것처럼 한 번에 한 행씩 실행됩니다. 다음은 응용 프로그램이 실행되는 단계를 보여 주는 가상의 예입니다.

1. 프레임 진입: 런타임에서 enterFrame 이벤트 핸들러를 호출하여 해당 코드를 한 번에 하나씩 실행합니다.
2. 마우스 이벤트: 사용자가 마우스를 움직이고 여러 rollover 및 rollout 이벤트가 발생할 때 런타임에서 마우스 이벤트 핸들러를 호출합니다.
3. complete 이벤트 로드: url에서 xml 파일을 로드하는 요청으로 인해 로드된 파일 데이터가 반환됩니다. 이벤트 핸들러가 호출되어 해당 단계를 실행하면서 xml 데이터에서 xml 내용을 읽고 객체 세트를 만듭니다.
4. 마우스 이벤트: 마우스가 다시 움직였으므로 런타임에서 관련 마우스 이벤트 핸들러를 호출합니다.
5. 렌더링: 대기 중인 이벤트가 더 이상 없으므로 런타임에서 표시 객체의 변경 내용에 따라 화면을 업데이트합니다.
6. 프레임 진입: 주기가 다시 시작됩니다.

위의 예에서 설명했듯이 가상의 단계 1~5는 프레임이라고 하는 단일 시간 블록 안에서 순서대로 실행됩니다. 전체 단계는 단일 스레드에서 순서대로 실행되므로 런타임에서는 프로세스 중 임의의 단계 하나를 실행하기 위해 다른 단계 하나를 중지할 수 없습니다. 초당 30프레임의 프레임 속도에서 런타임이 모든 작업을 실행하는 데 할당된 시간은 1/30초 미만입니다. 이는 대개 코드 실행에 충분한 시간이며 시간이 남아 런타임이 대기하는 것이 일반적입니다. 그러나 3단계에서 로드되는 xml 데이터가 용량이 매우 크며 깊이 중첩된 xml 구조로 된 경우도 있습니다. 이때, 코드가 xml에서 반복 실행되고 객체를 만들면서 모든 작업을 완료하려면 1/30초보다 더 긴 시간이 걸릴 수도 있습니다. 이 경우에는 마우스에 응답하고 화면을 다시 그리는 후반 단계는 정상 속도보다 더 느리게 실행됩니다. 이로 인해 화면은 마우스를 움직이는 사용자에 응답해서 충분한 속도로 다시 그려지지 않으므로 화면이 멈추는 현상이 발생할 수 있습니다.

모든 코드가 동일한 스레드에서 실행될 경우 이따금씩 발생하는 멈춤 현상을 피하는 방법은 단 한 가지입니다. 즉, 대용량 데이터 세트를 반복하는 것처럼 장시간 실행되는 작업을 수행하지 않는 것입니다. ActionScript 워커는 다른 해결책을 제공합니다. 별도의 워커에서 장시간 실행되는 코드를 실행할 수 있습니다. 각 워커는 개별 스레드에서 실행되므로 백그라운드 워커는 자체의 고유 스레드에서 장시간 실행되는 작업을 수행합니다. 따라서 기본 워커의 실행 드레드가 해제되어 화면이 다시 그려지면서 각 프레임은 다른 작업에 의해 차단되지 않습니다.

이러한 방식으로 다중 코드 작업을 동시에 실행하는 기능을 **동시성**이라고 합니다. 백그라운드 워커가 작업을 종료하거나 작업 중 "진행" 지점에 위치한 경우 기본 워커 알림 및 데이터를 전송할 수 있습니다. 이와 같은 방식으로, 복잡하거나 오랜 시간이 걸리는 작업을 수행하는 코드를 작성하고 사용자 환경에서 화면 멈춤이 발생하는 문제를 방지할 수도 있습니다.

프레임 속도는 기본 렌더링 스레드가 다른 코드에 의해 차단될 때 감소할 수 있으며, 워커를 사용하면 이러한 문제를 최소화할 수 있습니다. 그러나 워커는 시스템 메모리 및 CPU를 추가로 점유하며 이는 곧 전반적인 응용 프로그램 성능의 저하로 이어질 수 있습니다. 각 워커에서는 자체의 런타임 가상 머신 인스턴스를 사용하므로 작은 워커도 큰 오버헤드를 일으킬 수 있습니다. 워커를 사용할 경우 모든 대상 플랫폼에서 해당 코드를 테스트하여 시스템에 추가되는 부담이 너무 크지 않은지 확인해야 합니다. 일반적인 상황에서 백그라운드 워커는 3개 이상 사용하지 않는 것이 좋습니다.

워커 만들기 및 관리

데스크톱 플랫폼용 Flash Player 11.4 이상, Adobe AIR 13.4 이상

동시성 구현을 위해 워커를 사용하기 위한 첫 번째 단계는 백그라운드 워커를 만드는 것입니다. 두 가지 유형의 객체를 사용하여 워커를 만들 수 있습니다. 그중 하나는 Worker 인스턴스로서 이는 사용자가 직접 만듭니다. 다른 하나는 WorkerDomain 객체로서 이 객체에서는 Worker를 생성하고 응용 프로그램에서 실행 중인 Worker 객체를 관리합니다.

런타임은 로드된 후 자동으로 WorkerDomain 객체를 만듭니다. 또한 런타임에서는 응용 프로그램의 기본 swf에 대한 워커를 자동으로 만듭니다. 첫 번째 워커는 **초기 워커**라고 합니다.

한 응용 프로그램에 대해 하나의 WorkerDomain 객체만 존재하므로 정적 WorkerDomain.current 속성을 사용하여 WorkerDomain 인스턴스에 액세스할 수 있습니다.

언제든 정적 Worker.current 속성을 사용하면 현재의 코드가 실행되고 있는 워커인 현재의 Worker 인스턴스에 액세스할 수 있습니다.

swf에서 Worker 객체 만들기

기본 swf가 초기 워커 안에서 실행되는 것처럼 백그라운드 워커는 단일 swf 파일의 코드를 실행합니다. 백그라운드 워커를 사용하려면 워커의 코드를 swf 파일로 작성 및 컴파일해야 합니다. 백그라운드 워커를 만들려면 부모 워커는 해당 swf 파일의 바이트를 ByteArray 객체로 액세스할 수 있어야 합니다. 이 ByteArray를 WorkerDomain 객체의 createWorker() 메서드에 전달하여 실제로 워커를 만듭니다.

백그라운드 워커 swf를 ByteArray 객체로 가져올 수 있는 세 가지 주요 방법은 다음과 같습니다.

워커 swf 포함

[Embed] 메타태그를 사용하여 워커 swf를 기본 swf 안에 ByteArray로 포함시킵니다.

```
[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
}
```

워커 swf는 기본 swf 내에 BgWorker_ByteClass라는 이름의 ByteArray 하위 클래스로 컴파일됩니다. 이 클래스의 인스턴스를 만들면 워커 swf의 바이트로 미리 채워진 ByteArray가 제공됩니다.

외부 워커 swf 로드

URLLoader 객체를 사용하여 외부 swf 파일을 로드합니다. 이 swf 파일은 동일한 보안 도메인에서 가져와야 합니다. 예를 들면 기본 swf와 같은 인터넷 도메인에서 로드되거나 AIR 응용 프로그램 패키지에 포함되어 있는 swf 파일을 사용해야 합니다.

```
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
}
```

URLLoader에서 swf 파일의 로드를 마치면 URLLoader 객체의 data 속성(위의 예에서 event.target.data 참조)에서 해당 swf의 바이트를 사용할 수 있습니다.

기본 swf를 워커 swf로 사용하기

하나의 swf를 기본 swf 및 워커 swf로 모두 사용할 수 있습니다. 기본 표시 클래스의 loaderInfo.bytes 속성을 사용하여 swf의 바이트에 액세스합니다.

```
// 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
    }
}
```

이 기술을 사용할 경우 if 문을 활용하여 기본 클래스의 생성자 또는 이 생성자가 호출하는 메서드 안에서 swf 파일 코드를 분기하십시오. 해당 코드가 기본 워커 또는 백그라운드 워커 중 어디에서 실행되고 있는지 알려면 위의 예처럼 현재 Worker 객체의 isPrimordial 속성을 확인합니다.

워커 실행의 시작

워커를 만든 후에는 Worker 객체의 start() 메서드를 호출하여 코드 실행을 시작합니다. start() 작업은 즉시 발생하지 않습니다. 워커가 언제 실행되는지 알려면 Worker 객체의 workerState 이벤트에 대해 리스너를 등록하십시오. 이 이벤트는 Worker 객체가 코드 실행을 시작하는 경우처럼 수명 주기에서 상태를 전환할 때 전달됩니다. workerState 이벤트 핸들러에서 Worker 객체의 state 속성이 WorkerState.RUNNING인지 확인하십시오. 이 지점에서 워커는 실행 중이며 기본 클래스의 생성자가 실행되었습니다. 다음 코드 목록에서는 workerState 이벤트에 대해 등록하고 start() 메서드를 호출하는 예를 보여 줍니다.

```
// 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.
    }
}
```

워커 실행 관리

WorkerDomain 클래스의 listWorkers() 메서드를 사용하면 언제든 응용 프로그램에서 실행 중인 워커 세트에 액세스할 수 있습니다. 이 메서드에서는 초기 워커를 포함해 state 속성이 WorkerState.RUNNING인 워커 세트를 반환합니다. 시작되지 않았거나 실행이 이미 중지된 워커는 포함되지 않습니다.

워커가 더 이상 필요하지 않으면 Worker 객체의 terminate() 메서드를 호출하여 해당 워커를 종료하고 메모리와 기타 시스템 리소스를 해제할 수 있습니다.

워커 간 통신

데스크톱 플랫폼용 Flash Player 11.4 이상, Adobe AIR 13.4 이상

각 워커는 별개의 실행 스레드에서 자체의 코드를 실행하지만 서로 완전히 격리된 경우에는 아무런 이점도 제공하지 않습니다. 워커 간 통신은 결국 워커 간 데이터 전달을 의미합니다. 다음과 같이 세 가지 기본 메커니즘으로 한 워커에서 다른 워커로 데이터를 가져올 수 있습니다.

특정 데이터 전달 요구 사항에 이러한 데이터 공유 기법 중 어떤 것이 적합한지 결정할 때는 서로 다른 두 가지 주요 방식을 고려합니다. 이 둘 간의 한 가지 차이점은 새 데이터를 사용할 수 있음을 수신자에게 알리는 이벤트가 있는지 여부 또는 수신 워커가 업데이트를 확인해야 하는지 여부와 관련이 있습니다. 이러한 데이터 공유 기법 간의 또 다른 차이점은 데이터를 실제로 전달하는 방식과 관련이 있습니다. 경우에 따라 수신 워커는 공유 데이터의 복제본을 받으며, 이렇게 되면 더 많은 객체가 생성되어 메모리 및 CPU 주기가 늘어납니다. 또 어떤 경우 워커는 동일한 기본 시스템 메모리를 참조하는 객체에 액세스하므로 더 적은 객체가 생성되어 전반적으로 사용되는 메모리가 줄어듭니다. 이러한 차이점은 아래에 요약되어 있습니다.

통신 기법	데이터 수신 시 이벤트 전달	워커 간에 메모리 공유
워커 공유 속성	아니요	아니요. 객체가 참조가 아닌 복제본임
MessageChannel	예	아니요. 객체가 참조가 아닌 복제본임
공유 가능한 ByteArray	아니요	예. 메모리가 공유됨

공유 속성을 사용한 데이터 전달

워커 간에 데이터를 공유하는 가장 기본적인 방법은 공유 속성을 사용하는 것입니다. 각 워커에서는 공유 속성 값의 내부 사전을 유지 관리합니다. 각 속성은 String 키 이름과 함께 저장되어 다른 속성과 구분됩니다. 워커에 공유 속성으로 객체를 저장하려면 두 인수, 즉 저장할 키 이름 및 값과 함께 Worker 객체의 setSharedProperty() 메서드를 호출하십시오.

```
// code running in the parent worker
bgWorker.setSharedProperty("sharedPropertyName", someObject);
```

공유 속성이 설정된 후에 해당 값은 Worker 객체의 getSharedProperty() 메서드를 호출하여 키 이름을 전달하면 읽을 수 있습니다.

```
// code running in the background worker
receivedProperty = Worker.current.getSharedProperty("sharedPropertyName");
```

어떤 워커에서 속성 값을 읽거나 설정하는지에 대한 제한은 없습니다. 예를 들어 백그라운드 워커의 코드에서 setSharedProperty() 메서드를 호출하여 값을 저장할 수 있습니다. 그런 다음, 부모 워커에서 실행 중인 코드는 getSharedProperty()를 사용하여 데이터를 수신할 수 있습니다.

setSharedProperty() 메서드로 전달되는 값은 거의 모든 유형의 객체가 될 수 있습니다. getSharedProperty() 메서드를 호출하면 반환되는 객체는 몇몇 특수한 경우를 제외하고는 setSharedProperty()에 전달된 객체의 사본이며 동일한 객체에 대한 참조가 아닙니다. 데이터가 공유되는 방식에 대한 자세한 내용은 947페이지의 “공유 참조 및 복제 값”에 설명되어 있습니다.

공유 속성을 사용하여 워커 간에 데이터를 전달하는 방식의 가장 큰 이점은 워커가 실행되기 전에도 해당 방식을 활용할 수 있다는 것입니다. 백그라운드 Worker 객체의 setSharedProperty() 메서드를 호출하면 워커가 실행되기 전에도 공유 속성을 설정할 수 있습니다. 부모 워커에서 Worker의 start() 메서드를 호출하면 런타임에서는 자식 워커의 기본 클래스의 생성자를 호출합니다. start()가 호출되기 전에 설정된 모든 공유 속성은 자식 워커에서 코드를 통해 읽는 것이 가능합니다.

MessageChannel을 사용한 데이터 전달

메시지 채널은 두 워커 간에 단방향 데이터 전달 링크를 제공합니다. 워커 간 데이터 전달을 위해 MessageChannel 객체를 사용하는 방식에는 핵심적인 이점 하나가 있습니다. 메시지 채널을 사용하여 메시지(객체)를 전송할 경우, MessageChannel 객체에서는 channelMessage 이벤트를 전달합니다. 수신 워커의 코드에서는 해당 이벤트를 수신하여 데이터의 사용 가능 시기를 파악할 수 있습니다. 따라서 수신 워커에서는 데이터의 업데이트 여부를 상시적으로 확인할 필요가 없습니다.

메시지 채널 하나는 두 개의 워커, 즉 전송자 및 수신자와만 연결되어 있습니다. MessageChannel 객체를 만들려면 전송자 Worker 객체의 createMessageChannel() 메서드를 호출하고 수신 워커를 인수로 전달하십시오.

```
// In the sending worker swf
var sendChannel:MessageChannel;
sendChannel = Worker.current.createMessageChannel(receivingWorker);
```

두 워커 모두 MessageChannel 객체에 액세스할 수 있어야 합니다. 이를 위한 가장 간단한 방법은 setSharedProperty() 메서드를 사용하여 MessageChannel 객체를 전달하는 것입니다.

```
receivingWorker.setSharedProperty("incomingChannel", sendChannel);
```

수신 워커에서 MessageChannel 객체의 channelMessage 이벤트에 대해 리스너를 등록합니다. 이 이벤트는 전송 워커에서 메시지 채널을 통해 데이터를 전송할 때 전달됩니다.

```
// In the receiving worker swf
var incomingChannel:MessageChannel;
incomingChannel = Worker.current.getSharedProperty("incomingChannel");
incomingChannel.addEventListener(Event.CHANNEL_MESSAGE, handleIncomingMessage);
```

실제로 데이터를 전송하려면 전송 워커에서 MessageChannel 객체의 send() 메서드를 호출하십시오.

```
// In the sending worker swf
sendChannel.send("This is a message");
```

수신 워커에서 MessageChannel은 channelMessage 이벤트 핸들러를 호출합니다. 그런 다음 수신 워커에서는 MessageChannel 객체의 receive() 메서드를 호출하여 데이터를 가져올 수 있습니다.

```
private function handleIncomingMessage(event:Event):void
{
    var message:String = incomingChannel.receive() as String;
}
```

receive 메서드에 의해 반환된 객체는 send() 메서드에 전달된 객체와 동일한 데이터 유형을 갖습니다. 수신된 객체는 전송자에 의해 전달된 객체의 사본이며 전송 워커에 있는 객체의 참조가 아닙니다. 단, 해당 객체가 947페이지의 "공유 참조 및 복제 값"에 설명되어 있는 몇몇 데이터 유형 중 하나에 속하는 경우는 예외입니다.

공유 가능한 ByteArray를 사용하여 데이터 공유

두 워커 간에 객체가 전달되면 수신 워커는 원본의 복제본에 해당하는 새 객체를 받습니다. 두 객체는 시스템 메모리에서 서로 다른 위치에 저장됩니다. 따라서 객체의 복제본이 수신될 때마다 런타이멩서 사용되는 총 메모리가 증가합니다. 또한 한 워커의 객체에서 변경하는 사항이 다른 워커의 복제본에 영향을 주지 않습니다. 객체가 복제되는 방식에 대한 자세한 내용은 947페이지의 "공유 참조 및 복제 값"을 참조하십시오.

기본적으로 ByteArray 객체는 동일한 비헤이비어를 사용합니다. ByteArray 인스턴스를 Worker 객체의 setSharedProperty() 메서드 또는 MessageChannel 객체의 send() 메서드에 전달하는 경우 런타임에서는 컴퓨터의 메모리에 새 ByteArray를 생성하며 수신 워커는 새 ByteArray에 대한 참조에 해당하는 ByteArray 인스턴스를 가져옵니다. 하지만 해당 shareable 속성을 true로 설정하여 ByteArray 객체에 대한 이 비헤이비어를 변경할 수 있습니다.

워커 간에 공유 가능한 ByteArray 객체를 전달하는 경우 수신 워커의 ByteArray 인스턴스는 송신 워커의 ByteArray 인스턴스에서 사용하는 것과 동일한 기본 운영 체제 메모리에 대한 참조입니다. 한 워커의 코드로 인해 바이트 배열의 내용이 변경되는 경우 이러한 변경 사항은 공유되는 바이트 배열에 액세스할 수 있는 다른 워커에서 바로 사용할 수 있습니다.

두 워커는 각자의 코드를 동시에 실행하기 때문에 바이트 배열에 있는 동일한 바이트에 동시에 액세스하려고 할 수 있습니다. 이렇게 되면 데이터가 손실되거나 손상될 수 있습니다. 공유 리소스에 대한 액세스를 관리하고 이러한 문제를 방지하기 위해 여러 가지 API를 사용할 수 있습니다.

ByteArray 클래스에는 다음과 같이 단일 작업에서 바이트 배열의 내용을 검증하고 변경하는 데 사용할 수 있는 메서드가 있습니다.

- atomicCompareAndSwapIntAt() 메서드
- atomicCompareAndSwapLength() 메서드

또한 flash.concurrent 패키지에는 공유 리소스 작업에 필요한 액세스 제어 기능을 제공하는 다음과 같은 클래스가 포함되어 있습니다.

- Mutex 클래스
- Condition 클래스

공유 참조 및 복제 값

정상적인 상황에서 Worker.setSharedProperty() 또는 MessageChannel.send()를 호출할 경우 수신 워커에 전달되는 객체는 AMF 형식으로 직렬화되어 전달됩니다. 그 결과는 다음과 같습니다.

- getSharedProperty() 메서드가 호출될 때 수신 워커에 만들어지는 객체는 AMF 바이트로부터 비직렬화됩니다. 이 객체는 원본 객체의 참조가 아니라 사본입니다. 양쪽 워커의 객체에서 변경된 내용은 다른 워커에 있는 사본에는 적용되지 않습니다.
- 표시 객체와 같이 AMF 형식으로 직렬화될 수 없는 객체는 Worker.setSharedProperty() 또는 MessageChannel.send()를 사용하여 워커에 전달할 수 없습니다.
- 사용자 정의 클래스를 올바르게 비직렬화하려면 flash.net.registerClassAlias() 함수 또는 [RemoteClass] 메타데이터를 사용하여 클래스 정의를 등록해야 합니다. 해당 클래스의 두 워커 버전 모두에 같은 별칭을 사용해야 합니다.

다음 객체는 워커 간에 복제되는 대신 실제로 공유되는 다섯 가지 특수한 객체입니다.

- Worker 객체
- MessageChannel 객체
- 공유 가능한 바이트 배열(해당 shareable 속성이 true인 ByteArray 객체)
- Mutex 객체
- Condition 객체

Worker.setSharedProperty() 메서드 또는 MessageChannel.send() 메서드를 사용하여 두 객체 중 하나의 인스턴스를 전달하면 각 워커는 동일한 기본 객체에 대한 참조를 갖게 됩니다. 한 워커에 있는 인스턴스에서 변경된 내용은 다른 워커에서 즉시 사용 가능합니다. 또한 이 객체 중 하나의 동일한 인스턴스를 한 워커에 두 번 이상 전달할 경우 런타임에서는 수신 워커에 해당 객체의 새로운 사본을 만들지 않습니다. 그 대신 같은 참조가 다시 사용됩니다.

그 밖의 데이터 공유 기술

데이터 전달을 위한 워커별 메커니즘 외에도, 다음과 같은 두 swf 응용 프로그램 간 데이터 공유를 지원하는 기존 API를 사용하면 데이터를 교환할 수 있습니다.

- 로컬 공유 객체
- 데이터를 한 워커의 파일에 쓴 후 다른 워커의 파일에서 읽기
- SQLite 데이터베이스를 통해 데이터를 저장하고 읽기

둘 이상의 워커 간에 리소스를 공유할 경우에는 일반적으로 여러 워커가 동시에 해당 리소스에 액세스하지 않도록 방지해야 합니다. 예를 들어 여러 워커가 로컬 파일 시스템에 있는 하나의 파일에 액세스하면 데이터가 손실되거나 손상될 수 있으므로 이러한 액세스는 운영 체제에서 지원되지 않을 수도 있습니다.

동시 액세스 문제를 방지하려면 flash.concurrent 패키지의 Mutex 및 Condition 클래스를 사용하여 공유 리소스 작업에 필요한 액세스 제어 기능을 제공합니다.

다른 데이터 공유 메커니즘과 달리, SQLite 데이터베이스 엔진은 동시 액세스용으로 설계되었으며 자체적인 트랜잭션 지원 내장 기능을 갖추고 있습니다. 다중 워커는 데이터 손상의 위험 없이 SQLite 데이터베이스에 동시에 액세스할 수 있습니다. 각 워커는 서로 다른 SQLConnection 인스턴스를 사용하므로 개별 트랜잭션을 통해 데이터베이스에 액세스합니다. 따라서 동시적인 데이터 조작 작업은 데이터의 무결성에 영향을 주지 않습니다.

참고 사항

651페이지의 “AIR에서 로컬 SQL 데이터베이스를 사용한 작업”
flash.concurrent 패키지

64장: 보안

Flash Player 9 이상, Adobe AIR 1.0 이상

보안 문제는 Adobe, 사용자, 웹 사이트 소유자 및 내용 개발자의 주요 관심사입니다. 이에 따라 Adobe® Flash® Player 및 Adobe® AIR™에는 이러한 사용자, 웹 사이트 소유자 및 내용 개발자를 보호하기 위해 일련의 보안 규칙과 컨트롤이 포함되어 있습니다. 이 단원에서는 ActionScript 3.0으로 작성되어 Flash Player 9.0.124.0 이상에서 실행되는 SWF 파일에 대한 보안 모델과 특별히 언급되지 않는 한 AIR 1.0 이상에서 실행되는 SWF, HTML 및 JavaScript 파일에 대해 설명합니다.

이 장은 보안에 대한 개괄적 설명을 포함하며 특정 API를 사용한 상세 구현 정보, 사용 시나리오 또는 기타 세부 사항에 대해 포괄적으로 설명하기 위한 것은 아닙니다. Flash Player 보안 개념에 대한 자세한 내용은 www.adobe.com/go/devnet_security_kr에서 Flash Player 개발자 센터 항목 "보안"을 참조하십시오.

Flash Platform 보안 개요

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 AIR 런타임에서 사용되는 대부분의 보안 모델은 로드된 SWF 파일, HTML, 미디어 및 기타 에셋에 대한 원래 도메인을 기준으로 합니다. www.example.com과 같은 특정 인터넷 도메인을 기반으로 하는 파일의 실행 가능 코드는 언제든지 해당 도메인의 모든 데이터에 액세스할 수 있습니다. 이러한 에셋은 **보안 샌드박스**라는 동일한 보안 그룹에 포함됩니다. 자세한 내용은 950페이지의 “보안 샌드박스”를 참조하십시오.

예를 들어 SWF 파일의 ActionScript 코드는 SWF 파일, 비트맵, 오디오, 텍스트 파일 및 자체 도메인의 모든 에셋을 로드할 수 있습니다. 또한 동일한 도메인에서 ActionScript 3.0으로 작성된 두 파일 간의 크로스 스크립팅은 항상 허용됩니다. **크로스 스크립팅**은 한 파일의 코드가 다른 파일의 코드로 정의된 속성, 메서드 및 객체에 액세스하는 기능입니다.

ActionScript 3.0을 사용하여 작성된 SWF 파일과 ActionScript 3.0 이전 버전을 사용하여 작성된 SWF 파일 간의 크로스 스크립팅은 지원되지 않습니다. 단, LocalConnection 클래스를 사용하여 이러한 파일 간의 통신은 가능합니다. 또한 다른 도메인의 ActionScript 3.0 SWF 파일을 크로스 스크립팅하고 다른 도메인의 데이터를 로드하는 SWF 파일의 기능은 기본적으로 차단됩니다. 그러나 로드된 SWF 파일에서 Security.allowDomain() 메서드를 호출하여 이러한 액세스를 허용할 수 있습니다. 자세한 내용은 967페이지의 “크로스 스크립팅”을 참조하십시오.

기본적으로 다음과 같은 기본 보안 규칙이 적용됩니다.

- 동일한 보안 샌드박스의 리소스는 항상 서로 액세스할 수 있습니다.
- 원격 샌드박스의 파일에 있는 실행 가능 코드에서는 로컬 파일과 데이터에 액세스할 수 없습니다.

Flash Player 및 AIR 런타임은 다음과 같은 도메인을 개별 도메인으로 간주하고 각각에 대해 개별 보안 샌드박스를 설정합니다.

- http://example.com
- http://www.example.com
- http://store.example.com
- https://www.example.com
- http://192.0.34.166

http://example.com과 같은 이름을 가진 도메인이 http://192.0.34.166 등의 특정 IP 주소로 매핑되는 경우에도 런타임은 각각에 대해 개별 보안 샌드박스를 설정합니다.

SWF 파일에서 해당 샌드박스가 아닌 다른 샌드박스의 에셋에 액세스할 수 있도록 하기 위해 개발자는 다음과 같은 기본적인 두 가지 메서드를 사용할 수 있습니다.

- Security.allowDomain() 메서드(959페이지의 "제작자(개발자) 컨트롤" 참조)
- URL 정책 파일(956페이지의 "웹 사이트 컨트롤(정책 파일)" 참조)

Flash Player 및 AIR 런타임 보안 모델에서는 내용을 로드하는 것과 데이터를 추출하거나 액세스하는 것을 구분합니다. 내용이 시각적 미디어를 포함한 미디어로 정의된 경우 런타임은 오디오, 비디오 또는 표시된 미디어를 포함하는 SWF 파일 또는 HTML을 표시할 수 있습니다. 데이터는 코드에만 액세스할 수 있는 항목으로 정의됩니다. 내용과 데이터는 서로 다른 방법으로 로드됩니다.

- 내용 로드 - Flex를 사용할 경우 MXML 태그를 통해, AIR 응용 프로그램의 경우 HTML 태그를 통해 Loader, Sound 및 NetStream과 같은 클래스를 사용하여 내용을 로드할 수 있습니다.
- 데이터 추출 - Bitmap 객체, BitmapData.draw() 및 BitmapData.drawWithQuality() 메서드, Sound.id3 속성 또는 SoundMixer.computeSpectrum() 메서드를 사용하여 로드된 미디어 내용에서 데이터를 추출할 수 있습니다. drawWithQuality 메서드는 Flash Player 11.3 이상, AIR 3.3 이상에서 사용할 수 있습니다.
- 데이터 액세스 - URLStream, URLLoader, FileReference, Socket, XMLSocket 클래스 등의 클래스를 사용하여 XML 파일과 같은 외부 파일에서 로드하여 직접 데이터에 액세스할 수 있습니다. AIR는 FileStream 및 XMLHttpRequest와 같은 로드 중인 데이터를 위한 추가 클래스를 제공합니다.

Flash Player 보안 모델은 내용 로드와 데이터 액세스에 대해 서로 다른 규칙을 정의합니다. 일반적으로 데이터에 액세스하는 것보다 내용을 로드하는 데 적용되는 제한이 적습니다.

일반적으로 SWF 파일, 비트맵, mp3 파일, 비디오 등의 내용은 모든 위치에서 로드될 수 있지만 로드하는 코드 또는 내용의 도메인이 아닌 다른 도메인의 내용은 별도의 보안 샌드박스로 구분됩니다.

내용을 로드하는 데는 다음과 같은 몇 가지 제한이 있습니다.

- 기본적으로 사용자의 하드 드라이브와 같이 네트워크 주소가 아닌 위치에서 로드된 로컬 SWF 파일은 local-with-filesystem 샌드박스로 분류됩니다. 이러한 파일은 네트워크에서 내용을 로드할 수 없습니다. 자세한 내용은 950페이지의 "로컬 샌드박스"를 참조하십시오.
- RTMP(Real-Time Messaging Protocol) 서버로 내용에 대한 액세스를 제한할 수 있습니다. 자세한 내용은 967페이지의 "RTMP 서버를 사용하여 제공된 내용"을 참조하십시오.

로드된 미디어가 이미지, 오디오 또는 비디오인 경우, SWF 파일의 도메인이 미디어의 원래 도메인에 있는 URL 정책 파일에 포함된 경우에만 해당 보안 샌드박스 외부에 있는 SWF 파일에서 로드된 미디어의 데이터(예: 픽셀 데이터 및 사운드 데이터)에 액세스할 수 있습니다. 자세한 내용은 970페이지의 "데이터로 로드된 미디어 액세스"를 참조하십시오.

다른 형식의 로드된 데이터에는 URLLoader 객체를 통해 로드되는 텍스트나 XML 파일이 있습니다. 이 경우에도 다른 보안 샌드박스에서 모든 데이터에 액세스하려면 원래 도메인에 있는 URL 정책 파일을 통해 액세스 권한이 부여되어야 합니다. 자세한 내용은 972페이지의 "URLLoader 및 URLStream 사용"을 참조하십시오.

참고: AIR 응용 프로그램 샌드박스에서 원격 내용 또는 데이터를 로드하기 위해 실행되는 코드에서는 정책 파일이 필요하지 않습니다.

보안 샌드박스

Flash Player 9 이상, Adobe AIR 1.0 이상

클라이언트 컴퓨터는 외부 웹 사이트, 로컬 파일 시스템 또는 설치된 AIR 응용 프로그램 등 다양한 소스로부터 코드, 내용 및 데이터를 포함하는 개별 파일을 가져올 수 있습니다. Flash Player 및 AIR 런타임은 코드 파일과 공유 객체, 비트맵, 사운드, 비디오 및 데이터 파일 등의 기타 리소스를, 이들이 로드될 때의 원래 위치를 기준으로 하여 개별적으로 보안 샌드박스에 할당합니다. 다음 단원에서는 지정된 샌드박스 내에서 실행되는 코드 또는 내용이 액세스할 수 있는 항목을 제어하는 런타임의 규칙에 대해 설명합니다.

Flash Player 보안에 대한 자세한 내용은 www.adobe.com/go/devnet_security_kr에서 Flash Player 개발자 센터 항목 "보안"을 참조하십시오.

원격 샌드박스

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 AIR 런타임은 인터넷에서 가져온 에셋(SWF 파일 포함)을 원래 도메인에 해당하는 별개의 샌드박스로 분류합니다. 예를 들어 **example.com**에서 로드된 에셋은 **foo.org**에서 로드된 에셋과 다른 보안 샌드박스에 배치됩니다. 기본적으로 이러한 파일은 해당 서버의 모든 리소스에 대한 액세스가 허용됩니다. 원격 SWF 파일은 URL 정책 파일 및 Security.allowDomain() 메서드와 같은 명시적인 웹 사이트 및 제작자 권한을 사용하여 다른 도메인의 데이터에 추가로 액세스할 수 있습니다. 자세한 내용은 956페이지의 “웹 사이트 컨트롤(정책 파일)” 및 959페이지의 “제작자(개발자) 컨트롤”을 참조하십시오.

원격 SWF 파일에서는 로컬 파일이나 리소스를 로드할 수 없습니다.

Flash Player 보안에 대한 자세한 내용은 www.adobe.com/go/devnet_security_kr에서 Flash Player 개발자 센터 항목 "보안"을 참조하십시오.

로컬 샌드박스

Flash Player 9 이상, Adobe AIR 1.0 이상

로컬 파일은 file: 프로토콜이나 UNC(Universal Naming Convention) 경로를 사용하여 참조되는 모든 파일을 의미합니다. 로컬 SWF 파일은 다음 네 개의 로컬 샌드박스 중 하나에 배치됩니다.

- local-with-filesystem 샌드박스 - 보안을 위해 Flash Player 및 AIR 런타임은 모든 로컬 파일을 기본적으로 local-with-file-system 샌드박스에 배치합니다. 이 샌드박스에서 실행 가능 코드는 URLLoader 클래스 등을 사용하여 로컬 파일을 읽을 수 있지만, 네트워크와는 어떤 방식으로도 통신할 수 없습니다. 사용자는 로컬 데이터가 네트워크로 누출되거나 다른 방식으로 부적절하게 공유되지 않는다는 확신을 가질 수 있습니다.
- local-with-networking 샌드박스 - SWF 파일을 컴파일할 때, 로컬 파일로 실행되지만 네트워크 액세스가 가능하도록 지정할 수 있습니다(952페이지의 “로컬 SWF 파일의 샌드박스 유형 설정” 참조). 이러한 파일은 local-with-networking 샌드박스에 배치됩니다. local-with-networking 샌드박스에 할당된 SWF 파일에서는 해당 로컬 파일에 액세스할 수 없습니다. 대신, SWF 파일은 네트워크의 데이터에 액세스할 수 있습니다. 하지만 URL 정책 파일이나 Security.allowDomain() 메서드에 대한 호출을 통해 권한이 부여되지 않는 한 local-with-networking SWF 파일은 여전히 네트워크의 데이터를 읽을 수 없습니다. 이러한 권한을 부여하려면 URL 정책 파일에서 <allow-access-from domain="*"/> 또는 Security.allowDomain("*")을 사용하여 모든 도메인에 권한을 부여해야 합니다. 자세한 내용은 956페이지의 “웹 사이트 컨트롤(정책 파일)” 및 959페이지의 “제작자(개발자) 컨트롤”을 참조하십시오.
- local-trusted 샌드박스 - 사용자나 설치 프로그램에 의해 신뢰할 수 있는 파일로 등록된 로컬 SWF 파일은 local-trusted 샌드박스에 배치됩니다. 시스템 관리자와 사용자는 보안 고려 사항에 따라 local-trusted 샌드박스에 또는 해당 샌드박스로부터 로컬 SWF 파일을 재할당하거나 이동할 수도 있습니다(954페이지의 “관리자 컨트롤” 및 955페이지의 “사용자 컨트롤” 참조

). local-trusted 샌드박스에 할당된 SWF 파일은 다른 SWF 파일과 상호 작용하고 원격이나 로컬의 모든 위치에서 데이터를 로드할 수 있습니다.

- AIR 응용 프로그램 샌드박스 - 이 샌드박스에는 실행 중인 AIR 응용 프로그램과 함께 설치된 내용이 들어 있습니다. 기본적으로 AIR 응용 프로그램 샌드박스에서 실행되는 코드는 모든 도메인의 코드를 크로스 스크립팅할 수 있습니다. 그러나 AIR 응용 프로그램 샌드박스 외부의 파일은 응용 프로그램 샌드박스의 코드를 크로스 스크립팅할 수 없습니다. 기본적으로 AIR 응용 프로그램 샌드박스의 코드 및 내용은 도메인에서 내용과 데이터를 로드할 수 있습니다.

local-with-networking과 local-with-filesystem 샌드박스 간의 통신은 물론 local-with-filesystem과 원격 샌드박스 간의 통신이 엄격하게 금지됩니다. Flash Player에서 실행되는 응용 프로그램이나 사용자 또는 관리자는 이러한 통신을 허용하는 권한을 부여할 수 없습니다.

어느 방향으로든 로컬 HTML 파일과 로컬 SWF 파일 간에 스크립팅하려면(예: ExternalInterface 클래스 사용) 관련 HTML 파일과 로컬 SWF 파일이 모두 local-trusted 샌드박스에 있어야 합니다. 이것은 브라우저의 로컬 보안 모델이 Flash Player 로컬 보안 모델과 다르기 때문입니다.

local-with-networking 샌드박스의 SWF 파일은 local-with-filesystem 샌드박스의 SWF 파일을 로드할 수 없습니다. local-with-filesystem 샌드박스의 SWF 파일은 local-with-networking 샌드박스의 SWF 파일을 로드할 수 없습니다.

AIR 응용 프로그램 샌드박스

Adobe AIR 1.0 이상

Adobe AIR 런타임은 **application** 샌드박스라는 추가 샌드박스를 Flash Player 보안 샌드박스 모델에 추가합니다. AIR 응용 프로그램의 일부로 설치된 파일은 응용 프로그램 샌드박스에 로드됩니다. 응용 프로그램에서 로드된 다른 모든 파일의 보안 제한은 일반적인 Flash Player 보안 모델에서 지정한 제한 사항을 따릅니다.

응용 프로그램이 설치되면 AIR 패키지 내에 포함된 모든 파일이 사용자 컴퓨터의 응용 프로그램 디렉토리에 설치됩니다. 개발자는 app:/ URL 스킴을 통해 코드에서 이 디렉토리를 참조할 수 있습니다(741페이지의 “URI 스킴” 참조). 응용 프로그램이 실행하면 응용 프로그램 디렉토리 트리 내의 모든 파일이 응용 프로그램 샌드박스에 할당됩니다. 응용 프로그램 샌드박스의 내용은 로컬 파일 시스템과의 상호 작용을 포함하여 AIR 응용 프로그램에서 사용할 수 있는 전체 권한으로 보호됩니다.

많은 AIR 응용 프로그램에서는 로컬로 설치된 이러한 파일만 사용하여 응용 프로그램을 실행합니다. 그러나 AIR 응용 프로그램은 응용 프로그램 디렉토리 내의 파일만 제한하지 않고 모든 소스에서 파일 유형을 로드할 수 있습니다. 이러한 파일에는 사용자 컴퓨터에 대한 로컬 파일은 물론 로컬 네트워크 또는 인터넷에서의 소스와 같이 사용 가능한 외부 소스에서 가져온 파일도 포함됩니다. 파일 유형은 보안 제한 사항에는 영향을 주지 않습니다. 로드된 HTML 파일의 보안 권한은 동일한 소스에서 로드한 SWF 파일의 보안 권한과 동일합니다.

응용 프로그램 보안 샌드박스의 내용은 다른 샌드박스의 내용 사용을 금지하는 AIR API에 액세스할 수 있습니다. 예를 들어 응용 프로그램에 대한 응용 프로그램 설명자 파일의 내용을 반환하는 air.NativeApplication.nativeApplication.applicationDescriptor 속성은 응용 프로그램 보안 샌드박스의 내용에만 적용됩니다. 제한된 API의 또 다른 예는 로컬 파일 시스템을 읽고 쓰는 메서드를 포함하는 FileStream 클래스입니다.

응용 프로그램 보안 샌드박스의 내용에만 사용할 수 있는 ActionScript API는 **ActionScript 3.0 Reference for Adobe Flash Platform**에서 AIR 로고로 표시됩니다. 기타 샌드박스에서 이러한 API를 사용하면 런타임에서 SecurityError 예외가 발생합니다.

HTMLLoader 객체에 있는 HTML 내용의 경우 모든 AIR JavaScript API(AIRAliases.js 파일을 사용할 때 air 객체를 통해 또는 window.runtime 속성을 통해 사용 가능)를 응용 프로그램 보안 샌드박스의 내용에서 사용할 수 있습니다. 다른 샌드박스의 HTML 내용이 window.runtime 속성에 액세스할 수 없으므로 이 내용은 AIR 또는 Flash Player API에 액세스할 수 없습니다.

AIR 응용 프로그램 내에서 실행되는 내용은 다음과 같은 추가 제한을 가집니다.

- 응용 프로그램 보안 샌드박스에 있는 HTML 내용의 경우 코드를 로드한 후 문자열을 실행 코드로 동적으로 변환할 수 있는 API를 사용할 때 제한 사항이 있습니다. 이는 응용 프로그램이 잠재적으로 안전하지 않은 네트워크 도메인과 같은 비 응용 프

로그램 소스의 코드를 실수로 삽입 및 실행하지 않도록 하는 것입니다. 예를 들어 eval() 함수를 사용할 때 이러한 제한 사항이 적용됩니다. 자세한 내용은 985페이지의 "서로 다른 샌드박스의 내용에 대한 코드 제한 사항"을 참조하십시오.

- 가능한 피싱 공격을 막기 위해 ActionScript TextField 객체에 있는 HTML 내용의 img 태그가 응용 프로그램 보안 샌드박스의 SWF 내용에서 무시됩니다.
- 응용 프로그램 샌드박스의 내용은 ActionScript 2.0 텍스트 필드의 HTML 내용에 asfunction 프로토콜을 사용할 수 없습니다.
- 응용 프로그램 샌드박스의 SWF 내용은 Flash Player 9 업데이트 3에 추가된 기능인 크로스 도메인 캐시를 사용할 수 없습니다. 이 기능을 통해 Flash Player에서는 Adobe 플랫폼 구성 요소 내용을 영구적으로 캐시하고 필요 시 로드된 SWF 내용에서 다시 사용할 수 있습니다. 이 경우 내용을 여러 번 다시 로드할 필요가 없습니다.

AIR 내부의 JavaScript에 대한 제한

Adobe AIR 1.0 이상

응용 프로그램 보안 샌드박스의 내용과 달리 비 응용 프로그램 보안 샌드박스의 JavaScript 내용은 eval() 함수를 호출하여 동적으로 생성된 코드를 언제든지 실행할 수 있습니다. 하지만 AIR 내의 비 응용 프로그램 보안 샌드박스에서 실행되는 JavaScript에는 제한이 있습니다. 그 중 일부는 다음과 같습니다.

- 비 응용 프로그램 샌드박스의 JavaScript 코드에서는 window.runtime 객체에 액세스할 수 없으며 그러므로 이 코드에서 AIR API를 실행할 수 없습니다.
- 기본적으로 비 응용 프로그램 보안 샌드박스의 내용은 XMLHttpRequest 호출을 사용하여 요청을 호출하는 도메인 이외의 다른 도메인에서 데이터를 로드할 수 없습니다. 그러나 응용 프로그램 코드는 포함하는 프레임 또는 iframe에서 allowCrossdomainXHR 특성을 설정하여 이를 수행하도록 비 응용 프로그램 내용 권한을 부여할 수 있습니다. 자세한 내용은 985페이지의 "서로 다른 샌드박스의 내용에 대한 코드 제한 사항"을 참조하십시오.
- JavaScript window.open() 메서드 호출에 대한 제한 사항이 있습니다. 자세한 내용은 988페이지의 "JavaScript window.open() 메서드 호출에 대한 제한 사항"을 참조하십시오.
- 원격(네트워크) 보안 샌드박스의 HTML 내용은 원격 도메인(네트워크 URL)에서 CSS, frame, iframe 및 img 내용을 로드할 수만 있습니다.
- local-with-filesystem, local-with-networking 또는 local-trusted 샌드박스의 HTML 내용은 응용 프로그램 또는 네트워크 URL이 아닌 로컬 샌드박스에서 CSS, frame, iframe 및 img 내용을 로드할 수만 있습니다.

자세한 내용은 985페이지의 "서로 다른 샌드박스의 내용에 대한 코드 제한 사항"을 참조하십시오.

로컬 SWF 파일의 샌드박스 유형 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

컴퓨터의 최종 사용자나 관리자는 로컬 SWF 파일을 신뢰할 수 있는 파일로 지정하여 해당 파일에서 로컬과 네트워크의 모든 도메인에 있는 데이터를 로드하도록 할 수 있습니다. 이것은 전역 Flash Player Trust 및 사용자 Flash Player Trust 디렉토리에 지정됩니다. 자세한 내용은 954페이지의 "관리자 컨트롤" 및 955페이지의 "사용자 컨트롤"을 참조하십시오.

로컬 샌드박스에 대한 자세한 내용은 950페이지의 "로컬 샌드박스"를 참조하십시오.

Adobe Flash Professional

제작 도구에서 문서의 제작 설정을 설정함으로써 local-with-filesystem 샌드박스 또는 local-with-networking 샌드박스에 대해 SWF 파일을 구성할 수 있습니다.

Adobe Flex

Adobe Flex 컴파일러에서 use-network 플래그를 설정하여 local-with-filesystem 샌드박스 또는 local-with-networking 샌드박스에 대해 SWF 파일을 구성할 수 있습니다. 자세한 내용은 **Adobe Flex 3 응용 프로그램 작성 및 배포**의 "응용 프로그램 컴파일러 옵션 정보"를 참조하십시오.

Security.sandboxType 속성

Flash Player 9 이상, Adobe AIR 1.0 이상

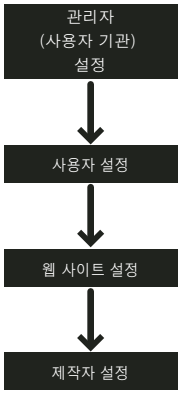
SWF 파일의 제작자는 읽기 전용의 정적 Security.sandboxType 속성을 사용하여 Flash Player 또는 AIR 런타임이 SWF 파일에 할당한 샌드박스의 유형을 확인할 수 있습니다. Security 클래스에는 다음과 같이 Security.sandboxType 속성의 가능한 값을 나타내는 상수가 포함되어 있습니다.

- Security.REMOTE - 인터넷 URL에서 가져온 SWF 파일이며 도메인 기반 샌드박스 규칙에 따라 작동합니다.
- Security.LOCAL_WITH_FILE - 로컬 SWF 파일이지만 사용자가 신뢰하지 않았고 네트워킹이 지정되지 않았습니다. 이 SWF 파일은 로컬 데이터 소스에서 읽을 수 있지만 인터넷과 통신할 수 없습니다.
- Security.LOCAL_WITH_NETWORK - 로컬 SWF 파일이고 사용자가 신뢰하지 않았지만 네트워킹이 지정되었습니다. 이 SWF 파일은 인터넷과 통신할 수 있지만 로컬 데이터 소스에서 읽을 수는 없습니다.
- Security.LOCAL_TRUSTED - 로컬 SWF 파일이고 설정 관리자나 Flash Player 신뢰 구성 파일을 사용하여 사용자가 신뢰한 파일입니다. 이 SWF 파일은 로컬 데이터 소스에서 읽을 수 있고 인터넷과 통신할 수 있습니다.
- Security.APPLICATION - SWF 파일은 AIR 응용 프로그램에서 실행되고 있고 해당 응용 프로그램용 패키지(AIR 파일)와 함께 설치되었습니다. 기본적으로 AIR 응용 프로그램 샌드박스의 파일은 모든 도메인의 임의 파일을 크로스 스크립팅할 수 있습니다. 그러나 AIR 응용 프로그램 샌드박스 외부의 파일은 AIR 파일을 크로스 스크립팅할 수 없습니다. 기본적으로 AIR 응용 프로그램 샌드박스의 파일은 도메인에서 내용과 데이터를 로드할 수 있습니다.

권한 컨트롤

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 클라이언트 런타임 보안 모델은 리소스를 중심으로 설계되어 있으며, 이러한 리소스는 SWF 파일, 로컬 데이터 및 인터넷 URL 등의 객체입니다. 이러한 리소스를 소유하거나 사용하는 **관계자**는 자신이 소유한 리소스에 대해 컨트롤(보안 설정)을 실행할 수 있습니다. 각 리소스에는 네 명의 관계자가 있습니다. Flash Player에서는 다음 그림과 같이 컨트롤에 대해 권한 계층 구조를 엄격하게 적용합니다.



보안 컨트롤 계층 구조

예를 들어 관리자가 리소스에 대한 액세스를 제한하면 다른 관계자가 해당 제한을 무시할 수 없습니다.

AIR 응용 프로그램의 경우 이러한 권한 제어는 AIR 응용 프로그램 샌드박스 외부에서 실행되는 내용에만 적용됩니다.

관리자 컨트롤

Flash Player 9 이상, Adobe AIR 1.0 이상

컴퓨터에 관리자 권한으로 로그인한 관리자의 경우 해당 컴퓨터의 다른 모든 사용자에게 영향을 주는 Flash Player 보안 설정을 적용할 수 있습니다. 가정용 컴퓨터와 같이 엔터프라이즈 환경이 아닌 컴퓨터에는 보통 한 명의 사용자가 관리자 권한도 가지고 있습니다. 엔터프라이즈 환경에서도 개별 사용자가 컴퓨터에 대해 관리자 권한을 가질 수 있습니다.

관리자 컨트롤에는 다음 두 가지 유형이 있습니다.

- mms.cfg 파일
- 전역 Flash Player Trust 디렉토리

mms.cfg 파일

Flash Player 9 이상, Adobe AIR 1.0 이상

mms.cfg 파일은 다양한 기능에 대한 관리자 액세스를 허용하거나 제한할 수 있는 텍스트 파일입니다. Flash Player를 시작하면 이 파일의 보안 설정을 읽어 기능을 제한하는 데 사용합니다. mms.cfg 파일에는 관리자가 개인 정보 제어, 로컬 파일 보안, 소켓 연결 등의 기능을 관리하는 데 사용하는 설정이 포함되어 있습니다.

SWF 파일은 Capabilities.avHardwareDisable 및 Capabilities.localFileReadDisable 속성을 호출하여 비활성화된 기능에 대한 일부 정보를 액세스할 수 있습니다. 그러나 mms.cfg 파일의 설정 대부분은 ActionScript에서 쿼리할 수 없습니다.

컴퓨터에 응용 프로그램과 무관한 보안 및 개인 정보 보호 정책을 적용하려면 시스템 관리자가 mms.cfg 파일을 수정해야 합니다. 응용 프로그램의 설치 관리자에서는 mms.cfg 파일을 사용할 수 없습니다. 관리자 권한으로 실행되는 설치 관리자는 mms.cfg 파일의 내용을 수정할 수 있지만, Adobe는 이러한 사용을 사용자의 신뢰 위반으로 간주하고 설치 프로그램 작성자에게 mms.cfg 파일을 수정하지 않도록 권고합니다.

mms.cfg 파일은 다음 위치에 저장됩니다.

- Windows: **system**\Macromed\Flash\mms.cfg

  (예: C:\WINDOWS\system32\Macromed\Flash\mms.cfg)
- Mac: **app support**/Macromedia/mms.cfg

  (예: /Library/Application Support/Macromedia/mms.cfg)

mms.cfg 파일에 대한 자세한 내용은 www.adobe.com/go/flash_player_admin_kr에서 Flash Player 관리 안내서를 참조하십시오.

전역 Flash Player Trust 디렉토리

Flash Player 9 이상, Adobe AIR 1.0 이상

관리자 및 설치 프로그램은 지정된 로컬 SWF 파일을 모든 사용자에 대해 신뢰할 수 있는 파일로 등록할 수 있습니다. 이러한 SWF 파일은 local-trusted 샌드박스에 할당되며, 다른 SWF 파일과 상호 작용하고 원격이나 로컬의 모든 위치에서 데이터를 로드할 수 있습니다. 파일은 다음 위치의 전역 Flash Player Trust 디렉토리에 신뢰할 수 있는 파일로 지정됩니다.

- Windows: **system**\Macromed\Flash\FlashPlayerTrust

  (예: C:\WINDOWS\system32\Macromed\Flash\FlashPlayerTrust)

- Mac: **app support**/Macromedia/FlashPlayerTrust

  (예: /Library/Application Support/Macromedia/FlashPlayerTrust)

Flash Player Trust 디렉토리에는 텍스트 파일을 무제한으로 포함할 수 있으며 각 텍스트 파일에는 신뢰할 수 있는 경로가 한 줄에 하나씩 나열되어 있습니다. 각 경로는 개별 SWF 파일, HTML 파일 또는 디렉토리일 수 있습니다. 주석 행은 # 심볼로 시작됩니다. 예를 들어 다음과 같은 텍스트가 포함된 Flash Player 신뢰 구성 파일은 지정된 디렉토리 및 모든 하위 디렉토리의 모든 파일에 대해 신뢰 상태를 부여합니다.

```
# Trust files in the following directories:
C:\Documents and Settings\All Users\Documents\SampleApp
```

신뢰 구성 파일에 나열된 경로는 항상 로컬 경로나 SMB 네트워크 경로여야 합니다. 신뢰 구성 파일의 모든 HTTP 경로는 무시되며, 로컬 파일만 신뢰할 수 있습니다.

충돌을 피하려면 각 신뢰 구성 파일에 설치 응용 프로그램에 해당하는 파일 이름을 부여하고 .cfg 파일 확장명을 사용합니다.

설치 프로그램을 통해 로컬로 실행되는 SWF 파일을 배포하려는 개발자의 경우, 설치 프로그램에서 전역 Flash Player Trust 디렉토리에 구성 파일을 추가하도록 함으로써, 배포하는 파일에 전체 권한을 부여할 수 있습니다. 이때, 설치 프로그램은 관리자 권한이 있는 사용자가 실행해야 합니다. mms.cfg 파일과 달리 전역 Flash Player Trust 디렉토리는 신뢰 권한을 부여하기 위한 목적으로 설치 프로그램에 포함될 수 있습니다. 관리자와 설치 프로그램 모두 전역 Flash Player Trust 디렉토리를 사용하여 신뢰할 수 있는 로컬 응용 프로그램을 지정할 수 있습니다.

또한 개별 사용자를 위한 Flash Player Trust 디렉토리도 있습니다(955페이지의 “사용자 컨트롤” 참조).

사용자 컨트롤

Flash Player 9 이상

Flash Player에서는 권한 설정에 대해 설정 UI, 설정 관리자 및 사용자 Flash Player Trust 디렉토리의 서로 다른 세 가지의 사용자 수준 메커니즘을 제공합니다.

설정 UI 및 설정 관리자

Flash Player 9 이상

설정 UI는 특정 도메인의 설정을 구성하는 빠른 대화형 메커니즘입니다. 설정 관리자는 보다 자세한 인터페이스와 많은 도메인이나 전체 도메인의 권한에 영향을 주는 글로벌 변경 기능을 제공합니다. 또한 SWF 파일에서 새로운 권한을 요청할 때 보안 또는 개인 정보에 대해 런타임 의사 결정이 필요한 경우, 사용자가 일부 Flash Player 설정을 조정할 수 있는 대화 상자가 표시됩니다.

설정 관리자 및 설정 UI에서는 카메라 및 마이크 설정, 공유 객체 저장 설정, 이전 내용과 관련된 설정 관련 등의 보안 관련 옵션을 제공합니다. AIR 응용 프로그램에서는 설정 관리자나 설정 UI를 사용할 수 없습니다.

참고: mms.cfg 파일에 지정된 모든 설정(954페이지의 "관리자 컨트롤" 참조)은 설정 관리자에 반영되지 않습니다.

설정 관리자에 대한 자세한 내용은 www.adobe.com/go/settingsmanager_kr를 참조하십시오.

사용자 Flash Player Trust 디렉토리

Flash Player 9 이상, Adobe AIR 1.0 이상

사용자 및 설치 프로그램은 지정된 로컬 SWF 파일을 신뢰할 수 있는 파일로 등록할 수 있습니다. 이러한 SWF 파일은 local-trusted 샌드박스에 할당되며, 다른 SWF 파일과 상호 작용하고 원격이나 로컬의 모든 위치에서 데이터를 로드할 수 있습니다. 사용자는 다음 위치에서 공유 객체 저장 영역과 동일한 디렉토리에 있는 사용자 Flash Player Trust 디렉토리에 신뢰할 수 있는 파일로 지정합니다(위치는 현재 사용자에 따라 다름).

- Windows: app data\Macromedia\Flash Player\#Security\FlashPlayerTrust

  (예: Windows XP의 경우 C:\Documents and Settings\JohnD\Application Data\Macromedia\Flash Player\#Security\FlashPlayerTrust 또는 Windows Vista의 경우 C:\Users\JohnD\AppData\Roaming\Macromedia\Flash Player\#Security\FlashPlayerTrust)

  Windows에서 Application Data 폴더는 기본적으로 숨겨져 있습니다. 숨겨진 폴더와 파일을 표시하려면 [내 컴퓨터]를 선택하여 Windows 탐색기를 열고 [도구] > [폴더 옵션]을 선택한 다음 [보기] 탭을 선택합니다. [보기] 탭에서 [숨김 파일 및 폴더 표시] 라디오 버튼을 선택합니다.

- Mac: app data/Macromedia/Flash Player/#Security/FlashPlayerTrust

  (예: /Users/JohnD/Library/Preferences/Macromedia/Flash Player/#Security/FlashPlayerTrust)

  이러한 설정은 현재 사용자에게만 영향을 주며 컴퓨터에 로그인한 다른 사용자에게는 적용되지 않습니다. 관리자 권한이 없는 사용자가 시스템의 자체 소유 부분에 응용 프로그램을 설치한 경우, 설치 관리자에서 사용자 Flash Player Trust 디렉토리에 이 응용 프로그램을 해당 사용자에 대해 신뢰할 수 있는 것으로 등록할 수 있습니다.

  설치 프로그램을 통해 로컬로 실행되는 SWF 파일을 배포하는 개발자의 경우, 설치 프로그램에서 사용자 Flash Player Trust 디렉토리에 구성 파일을 추가하도록 함으로써 배포하는 파일에 전체 권한을 부여할 수 있습니다. 이런 경우에도 사용자 Flash Player Trust 디렉토리 파일이 사용자 컨트롤로 간주되며, 이는 사용자 액션(설치)으로 시작되기 때문입니다.

  또한 관리자나 설치 프로그램에서 전역 Flash Player Trust 디렉토리를 사용하여 컴퓨터의 모든 사용자에 대해 응용 프로그램을 등록할 수 있습니다(954페이지의 "관리자 컨트롤" 참조).

웹 사이트 컨트롤(정책 파일)

Flash Player 9 이상, Adobe AIR 1.0 이상

웹 서버의 데이터를 다른 도메인의 SWF 파일에서 사용할 수 있도록 하기 위해 사용자 서버에 정책 파일을 작성할 수 있습니다. 정책 파일은 사용자 서버의 특정 위치에 배치되는 XML 파일입니다.

정책 파일은 다음을 포함하여 많은 에셋에 대한 액세스에 영향을 줍니다.

- 비트맵, 사운드 및 비디오 데이터
- XML 및 텍스트 파일 로드
- 다른 보안 도메인의 SWF 파일을 로드하는 SWF 파일의 보안 도메인으로 가져오기
- 소켓 및 XML 소켓 연결에 대한 액세스

ActionScript 객체는 문서 기반 서버 연결 및 소켓 연결의 서로 다른 두 종류의 서버 연결을 인스턴스화합니다. Loader, Sound, URLLoader 및 URLStream과 같은 ActionScript 객체는 문서 기반 연결을 인스턴스화하며 이러한 객체는 URL에서 파일을 로드합니다. ActionScript Socket 및 XMLSocket 객체는 소켓 연결을 만들어 로드된 문서가 아니라 스트리밍 데이터로 작업합니다.

Flash Player에서는 두 종류의 서버 연결을 지원하기 때문에 URL 정책 파일과 소켓 정책 파일의 두 가지 정책 파일 유형이 있습니다.

- 문서 기반 연결에는 **URL 정책 파일**이 필요합니다. 이러한 파일을 통해 서버는 특정 도메인이나 모든 도메인에서 제공된 SWF 파일에 해당 데이터와 문서를 사용할 수 있음을 나타낼 수 있습니다.
- 소켓 연결에는 Socket 및 XMLSocket 클래스를 사용하여 하위 TCP 소켓 수준에서 직접 네트워킹을 가능하게 하는 **소켓 정책 파일**이 필요합니다.

Flash Player에서는 정책 파일을 전송할 때 연결 시도에서 사용하려는 프로토콜과 동일한 프로토콜을 사용해야 합니다. 예를 들어 정책 파일이 HTTP 서버에 있는 경우 다른 도메인의 SWF 파일은 HTTP 서버로 데이터를 로드할 수 있습니다. 하지만 동일한 서버에 소켓 정책 파일을 제공하지 않으면 다른 도메인의 SWF 파일에서 소켓 수준의 서버에 연결할 수 없게 됩니다. 즉, 정책 파일을 가져올 때는 연결할 때와 같은 방법을 사용해야 합니다.

정책 파일 사용 및 구문은 Flash Player 10용으로 제작된 SWF 파일에 적용되기 때문에 이 단원의 나머지 부분에서 간략하게 설명합니다. 연속 릴리스에서 Flash Player 보안이 강화되었으므로 정책 파일 구현은 이전 버전의 Flash Player와 약간 다릅니다. 정책 파일에 대한 자세한 내용은 www.adobe.com/go/devnet_security_kr에서 Flash Player 개발자 센터 항목 "Flash Player 9의 정책 파일 변경 내용"을 참조하십시오.

AIR 응용 프로그램 샌드박스에서 실행되는 코드에는 URL 또는 소켓에서 데이터를 액세스하는 데 정책 파일이 필요하지 않습니다. 비 응용 프로그램 샌드박스에서 실행되는 AIR 응용 프로그램의 코드에는 정책 파일이 필요합니다.

마스터 정책 파일

Flash Player 9 이상, Adobe AIR 1.0 이상

기본적으로 Flash Player와 AIR 응용 프로그램 샌드박스에 있지 않은 AIR 내용은 먼저 서버의 루트 디렉토리에서 crossdomain.xml이라는 URL 정책 파일을 찾고 포트 843에서 소켓 정책 파일을 찾습니다. 이러한 위치 중 하나에 있는 파일을 **마스터 정책 파일**이라고 합니다. 소켓 연결의 경우 Flash Player는 기본 연결과 동일한 포트에서 소켓 정책 파일을 찾습니다. 그러나 해당 포트에 있는 정책 파일은 마스터 정책 파일로 간주되지 않습니다.

액세스 권한을 지정하는 것 외에도 마스터 정책 파일에는 **메타 정책** 문이 포함될 수 있습니다. 메타 정책은 정책 파일이 포함될 수 있는 위치를 지정합니다. URL 정책 파일에 대한 기본 메타 정책은 "master-only"이므로 서버에서 허용되는 정책 파일은 /crossdomain.xml뿐입니다. 소켓 정책 파일에 대한 기본 메타 정책은 "all"이므로 호스트의 모든 소켓에서 소켓 정책 파일을 제공할 수 있습니다.

참고: Flash Player 9 이전 버전에서 URL 정책 파일에 대한 기본 메타 정책은 "all"이었으므로 모든 디렉토리에 정책 파일이 포함될 수 있습니다. 기본 /crossdomain.xml 파일이 아닌 위치에서 정책 파일을 로드하는 응용 프로그램을 배포했으며 해당 응용 프로그램이 현재 Flash Player 10에서 실행 중일 수 있는 경우 사용자(또는 서버 관리자)가 추가 정책 파일을 허용하도록 마스터 정책 파일을 수정해야 합니다. 다른 메타 정책을 지정하는 방법에 대한 자세한 내용은 www.adobe.com/go/devnet_security_kr에서 Flash Player 개발자 센터 항목 "Flash Player 9의 정책 파일 변경 내용"을 참조하십시오.

SWF 파일에서 Security.loadPolicyFile() 메서드를 호출하여 다른 정책 파일 이름이나 다른 디렉토리 위치를 확인할 수 있습니다. 그러나 대상 위치에서 정책 파일을 제공할 수 있도록 마스터 정책 파일에 지정되어 있지 않으면 해당 위치에 정책 파일이 있는 경우에도 loadPolicyFile() 호출 시 아무 영향도 주지 않습니다. 정책 파일이 필요한 네트워크 작업을 시도하기 전에 loadPolicyFile()을 호출합니다. Flash Player는 자동으로 네트워킹 요청을 해당 정책 파일 시도 뒤에 대기열에 보관합니다. 따라서 예를 들어 네트워킹 작업을 시작하기 직전에 Security.loadPolicyFile()을 호출할 수 있습니다.

마스터 정책 파일을 확인할 때 Flash Player는 3초 동안 서버 응답을 기다립니다. 응답이 수신되지 않으면 Flash Player는 마스터 정책 파일이 없다고 가정합니다. 그러나 loadPolicyFile() 호출에 대한 기본 제한 시간 값은 없습니다. Flash Player는 호출되는 파일이 있다고 가정하고 파일을 로드하는 데 필요한 시간 동안 기다립니다. 따라서 마스터 정책 파일이 로드되도록 하려면 loadPolicyFile()을 사용하여 명시적으로 파일을 호출합니다.

메서드에 Security.loadPolicyFile()이란 이름이 지정된 경우에도 정책 파일이 필요한 네트워크 호출을 실행해야 정책 파일이 로드됩니다. loadPolicyFile() 호출은 단순히 정책 파일이 필요할 때 찾을 위치를 Flash Player에 알리는 기능만 합니다.

정책 파일 요청이 시작 또는 완료된 시기에 대한 알림은 받을 수 없으며 알림을 받을 이유도 없습니다. Flash Player는 비동기적으로 정책을 확인하며, 정책 파일 확인이 성공한 후 자동으로 연결을 시작합니다.

다음 단원에는 URL 정책 파일에만 적용되는 정보가 포함되어 있습니다. 소켓 정책 파일에 대한 자세한 내용은 972페이지의 "소켓 연결"을 참조하십시오.

URL 정책 파일 범위

Flash Player 9 이상, Adobe AIR 1.0 이상

URL 정책 파일은 정책 파일이 로드되어 있는 디렉토리와 해당 하위 디렉토리에만 적용됩니다. 루트 디렉토리에 있는 정책 파일은 전체 서버에 적용되고, 임의의 하위 디렉토리에서 로드된 정책 파일은 해당 디렉토리와 하위 디렉토리에만 적용됩니다.

정책 파일은 해당 파일이 상주하는 특정 서버에 대한 액세스에만 영향을 줍니다. 예를 들어 https://www.adobe.com:8080/crossdomain.xml에 있는 정책 파일은 HTTPS를 통해 www.adobe.com의 8080 포트에 대한 데이터 로드 호출에만 적용됩니다.

URL 정책 파일에 액세스 권한 지정

Flash Player 9 이상, Adobe AIR 1.0 이상

정책 파일에는 하나의 <cross-domain-policy> 태그가 있습니다. 그리고 이 태그에는 0개 이상의 <allow-access-from> 태그가 포함됩니다. 각 <allow-access-from> 태그에는 domain이라는 속성이 있습니다. 이 속성은 정확한 IP 주소, 정확한 도메인 또는 와일드카드 도메인(임의의 도메인)을 지정합니다. 와일드카드 도메인은 다음 두 가지 방법 중 하나로 표시됩니다.

- 모든 도메인과 IP 주소를 나타내는 경우 단일 별표(*)
- 특정 접미어로 끝나는 도메인을 나타내는 경우 접미어가 뒤에 붙은 별표

접미어는 점으로 시작해야 합니다. 그러나, 접미어를 가진 별표 문자는 앞의 점을 제외한 접미어만으로 구성되는 도메인과 일치할 수 있습니다. 예를 들어 xyz.com은 *.xyz.com에 속하는 것으로 간주됩니다. IP 도메인 형식에는 와일드카드를 사용할 수 없습니다.

다음 예제에서는 *.example.com, www.friendOfExample.com 및 192.0.34.166에서 시작되는 SWF 파일에 대한 액세스를 허용하는 URL 정책 파일을 보여 줍니다.

```
<?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>
```

IP 주소를 지정하면 IP 구문(예: http://65.57.83.12/flashmovie.swf)을 사용하여 해당 IP 주소에서 로드된 SWF에만 액세스 권한이 부여됩니다. 도메인 이름 구문을 사용하여 로드된 SWF에는 액세스 권한이 부여되지 않습니다. Flash Player는 DNS 이름 확인을 수행하지 않습니다.

다음 예제와 같이 모든 도메인의 문서에 액세스할 수 있도록 허용할 수 있습니다.

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*" />
</cross-domain-policy>
```

각 <allow-access-from> 태그에는 선택적인 secure 속성이 포함될 수 있으며 기본값은 true입니다. 정책 파일이 HTTPS 서버에 있을 경우 HTTPS가 아닌 서버의 SWF 파일이 HTTPS 서버에서 데이터를 로드할 수 있도록 허용하려면 이 속성을 false로 설정합니다.

secure 속성을 false로 설정하면 HTTPS에서 제공하는 보안 기능이 손상될 수 있습니다. 특히 이 속성을 false로 설정하면 보안 내용이 열려서 스누핑(snooping) 및 스푸핑(spoofing) 공격에 취약해집니다. 따라서 secure 속성을 false로 설정하지 않아야 합니다.

로드할 데이터가 HTTPS 서버에 있고, 해당 데이터를 로드하는 SWF 파일이 HTTP 서버에 있는 경우에는 로드하는 SWF 파일을 HTTPS 서버로 이동하여 HTTPS의 보호를 받아 보안 데이터의 모든 복사본을 보관하는 것이 좋습니다. 그러나 로드하는 SWF 파일을 HTTP 서버에 두기로 결정한 경우에는 다음 코드에 표시된 대로 secure="false" 속성을 <allow-access-from> 태그에 추가합니다.

```
<allow-access-from domain="www.example.com" secure="false" />
```

액세스를 허용하는 데 사용할 수 있는 다른 요소는 allow-http-request-headers-from 태그입니다. 이 요소는 다른 권한 도메인의 내용을 호스팅하는 클라이언트가 사용자 정의 머리글을 사용자 도메인으로 보낼 수 있도록 허용합니다. <allow-access-from> 태그는 사용자 도메인에서 데이터를 가져오는 권한을 다른 도메인에 부여하는 반면, allow-http-request-headers-from 태그는 사용자 도메인에 머리글 형태로 데이터를 밀어넣는 권한을 다른 도메인에 부여합니다. 다음 예제에서는 모든 도메인이 현재 도메인에 SOAPAction 머리글을 보낼 수 있습니다.

```
<cross-domain-policy>
    <allow-http-request-headers-from domain="*" headers="SOAPAction"/>
</cross-domain-policy>
```

allow-http-request-headers-from 문이 마스터 정책 파일에 있으면 호스트에 있는 모든 디렉토리에 적용됩니다. 그렇지 않으면 해당 문이 포함된 정책 파일의 디렉토리와 하위 디렉토리에만 적용됩니다.

정책 파일 미리 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

서버에서 데이터를 로드하거나 소켓에 연결하는 작업은 비동기 작업입니다. Flash Player는 기본 작업을 시작하기 전에 정책 파일의 다운로드가 완료될 때까지 기다립니다. 그러나 이미지에서 픽셀 데이터를 추출하거나 사운드에서 샘플 데이터를 추출하는 작업은 동기 작업입니다. 데이터를 추출하려면 먼저 정책 파일이 로드되어야 합니다. 미디어를 로드하는 경우에는 다음과 같이 미디어에서 정책 파일을 확인하도록 지정합니다.

- Loader.load() 메서드를 사용하는 경우에는 LoaderContext 객체인 context 매개 변수의 checkPolicyFile 속성을 설정합니다.
- <img> 태그를 사용하여 텍스트 필드에 이미지를 포함하는 경우, 다음과 같이 <img> 태그의 checkPolicyFile 속성을 "true"로 설정합니다.

  ```
  <img checkPolicyFile = "true" src = "example.jpg">
  ```

- Sound.load() 메서드를 사용하는 경우에는 SoundLoaderContext 객체인 context 매개 변수의 checkPolicyFile 속성을 설정합니다.
- NetStream 클래스를 사용하는 경우, NetStream 객체의 checkPolicyFile 속성을 설정합니다.

이 매개 변수 중 하나를 설정하는 경우에는 Flash Player에서 먼저 해당 도메인에서 이미 다운로드한 모든 정책 파일을 확인합니다. 그런 다음 서버의 기본 위치에서 정책 파일을 찾고 <allow-access-from> 문과 메타 정책이 있는지 모두 확인합니다. 마지막으로, Security.loadPolicyFile() 메서드에 대해 대기 중인 모든 호출을 검토하여 해당 범위 내에 있는지 확인합니다.

제작자(개발자) 컨트롤

Flash Player 9 이상, Adobe AIR 1.0 이상

보안 권한을 부여하는 데 사용되는 기본 ActionScript API는 Security.allowDomain() 메서드이며, 이 메서드를 사용하여 지정한 도메인의 SWF 파일에 권한을 부여합니다. 다음 예제에서는 SWF 파일이 www.example.com 도메인에서 제공되는 SWF 파일에 대한 액세스 권한을 부여합니다.

```
Security.allowDomain("www.example.com")
```

이 메서드는 다음과 같은 권한을 부여합니다.

- SWF 파일 간 크로스 스크립팅(967페이지의 “크로스 스크립팅” 참조)
- 표시 목록 액세스(969페이지의 “표시 목록 순회” 참조)
- 이벤트 감지(970페이지의 “이벤트 보안” 참조)
- Stage 객체의 속성 및 메서드에 대한 전체 액세스(968페이지의 “스테이지 보안” 참조)

Security.allowDomain() 메서드를 호출하는 주요 목적은 도메인 외부에 있는 SWF 파일에 Security.allowDomain() 메서드를 호출하는 SWF 파일을 스크립팅할 수 있는 권한을 부여하는 것입니다. 자세한 내용은 967페이지의 “크로스 스크립팅”을 참조하십시오.

Security.allowDomain() 메서드에 매개 변수로 IP 주소를 지정하더라도 지정된 IP 주소에 있는 모든 항목에 액세스가 허용되는 것은 아닙니다. 이때 해당 IP 주소에 매핑되는 도메인 이름이 아닌 URL에 지정한 IP 주소가 들어 있는 항목에만 액세스가 허용됩니다. 예를 들어 도메인 이름 www.example.com이 IP 주소 192.0.34.166으로 매핑되는 경우 Security.allowDomain("192.0.34.166")을 호출해도 www.example.com에 액세스가 허용되지 않습니다.

"*" 와일드카드를 Security.allowDomain() 메서드로 전달하여 모든 도메인에서의 액세스를 허용할 수 있습니다. 이렇게 하면 모든 도메인의 SWF 파일에 권한이 부여되어 호출하는 SWF 파일을 스크립팅하므로 "*" 와일드카드는 주의해서 사용해야 합니다.

ActionScript에는 Security.allowInsecureDomain()이라는 두 번째 권한 부여 API가 포함되어 있습니다. 이 메서드는 Security.allowDomain() 메서드와 동일한 작용을 하지만, 보안 HTTPS 연결에 의해 제공되는 SWF 파일에서 이 메서드를 호출하는 경우에 HTTP와 같이 비보안 프로토콜에서 제공되는 다른 SWF 파일에 대해서도 호출하는 SWF 파일에 대한 액세스를 추가로 허용한다는 것이 다릅니다. 그러나 보안 프로토콜(HTTPS)의 파일과 비보안 프로토콜(HTTP 등)의 파일 간에 스크립팅을 허용하는 것은 보안상 안전하지 않습니다. 이렇게 하면 보안 내용이 열려 스누핑 및 스푸핑 공격에 취약해집니다. 이러한 공격의 작동 방식은 다음과 같습니다. Security.allowInsecureDomain() 메서드는 HTTP 연결을 통해 제공되는 SWF 파일에 대해 보안 HTTPS 데이터에 대한 액세스를 허용하므로 공격자가 HTTP 서버와 사용자 간에 끼어 들어 HTTP SWF 파일을 자신의 파일로 교체함으로써 HTTPS 데이터에 액세스할 수 있게 됩니다.

중요: AIR 응용 프로그램 샌드박스에서 실행되는 코드는 Security 클래스의 allowDomain() 또는 allowInsecureDomain() 메서드를 호출하도록 허용되지 않습니다.

또 다른 중요한 보안 관련 메서드에는 Security.loadPolicyFile() 메서드가 있으며, 이를 통해 Flash Player는 비표준 위치에서 정책 파일을 확인합니다. 자세한 내용은 956페이지의 “웹 사이트 컨트롤(정책 파일)”을 참조하십시오.

제한적 네트워킹 API

Flash Player 9 이상, Adobe AIR 1.0 이상

네트워킹 API는 다음 두 가지 방법으로 제한될 수 있습니다. 악의적인 동작을 방지하기 위해 일반적으로 예약된 포트에 대한 액세스는 차단됩니다. 사용자 코드에서 이러한 블록을 재정의할 수 없습니다. 다른 포트와 관련해서 네트워크 기능에 대한 SWF 파일의 액세스를 제어하기 위해 allowNetworking 설정을 사용할 수 있습니다.

차단된 포트

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 Adobe AIR는 브라우저와 마찬가지로 특정 포트에 대한 HTTP 액세스를 제한합니다. 일반적으로 HTTP가 아닌 서버 유형에 사용되는 특정 표준 포트에 대한 HTTP 요청은 허용되지 않습니다.

네트워크 URL에 액세스하는 모든 API에 이러한 포트 차단 제한이 적용됩니다. 단, Socket.connect() 및 XMLSocket.connect()와 같이 직접 소켓을 호출하는 API나 소켓 정책 파일이 로드되는 Security.loadPolicyFile() 호출은 예외입니다. 소켓 연결은 대상 서버의 소켓 정책 파일을 사용하여 허용되거나 거부됩니다.

다음 목록에서는 포트 차단이 적용되는 ActionScript 3.0 API를 보여 줍니다.

FileReference.download(), FileReference.upload(), Loader.load(), Loader.loadBytes(), navigateToURL(), NetConnection.call(), NetConnection.connect(), NetStream.play(), Security.loadPolicyFile(), sendToURL(), Sound.load(), URLLoader.load(), URLStream.load()

포트 차단은 공유 라이브러리 가져오기, 텍스트 필드에 <img> 태그 사용, <object> 및 <embed> 태그를 사용하여 HTML 페이지에 SWF 파일을 로드하는 경우에도 적용됩니다.

포트 차단은 텍스트 필드에 <img> 태그 사용, <object> 및 <embed> 태그를 사용하여 HTML 페이지에 SWF 파일을 로드하는 경우에도 적용됩니다.

다음 목록에서는 차단되는 포트를 보여 줍니다.

HTTP: 20(ftp 데이터), 21(ftp 제어)

HTTP 및 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)

allowNetworking 매개 변수 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

SWF 파일이 포함된 HTML 페이지에 있는 <object> 및 <embed> 태그의 allowNetworking 매개 변수를 설정하여 SWF 파일이 네트워크 기능에 액세스하는 것을 제어할 수 있습니다.

allowNetworking은 다음과 같은 값을 가질 수 있습니다.

- "all"(기본값) - SWF 파일에서 모든 네트워킹 API를 사용할 수 있습니다.
- "internal" - SWF 파일에서 이 단원의 뒷부분에 나열되어 있는 브라우저 내비게이션이나 브라우저 상호 작용 API를 호출할 수 없지만 다른 네트워킹 API를 호출할 수 있습니다.
- "none" - SWF 파일에서 이 단원의 뒷부분에 나열되어 있는 브라우저 내비게이션이나 브라우저 상호 작용 API를 호출할 수 없고 마찬가지로, 뒷부분에 나열되어 있는 모든 SWF 간 통신 API를 사용할 수 없습니다.

allowNetworking 매개 변수는 주로 SWF 파일과 포함하는 HTML 페이지가 다른 도메인에 속해 있을 때 사용됩니다. 로드되는 SWF 파일이 포함하는 HTML 페이지와 동일한 도메인에 속해 있는 경우에는 "internal" 또는 "none" 값을 사용하지 않는 것이 좋습니다. SWF 파일이 원하는 HTML 페이지와 함께 로드되지 않을 수도 있기 때문입니다. 신뢰할 수 없는 사람이 포함하는 HTML 없이 사용자 도메인에서 SWF 파일을 로드할 수 있으며, 이런 경우에는 allowNetworking 제한이 예상대로 작동하지 않습니다.

금지된 API를 호출하면 SecurityError 예외가 발생합니다.

다음 예제에 표시된 대로 allowNetworking 매개 변수를 추가하고 SWF 파일에 대한 참조가 포함된 HTML 페이지의 <object> 및 <embed> 태그에 해당 값을 설정합니다.

```
<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>
```

또한 HTML 페이지는 스크립트를 사용하여 SWF를 포함하는 태그를 생성할 수 있습니다. 해당 스크립트를 변경하여 올바른 allowNetworking 설정을 삽입해야 합니다. Adobe Flash Professional 및 Adobe Flash Builder에서 생성된 HTML은 AC_FL_RunContent() 함수를 사용하여 SWF 파일에 대한 참조를 포함합니다. 다음과 같이 allowNetworking 매개 변수 설정을 스크립트에 추가합니다.

```
AC_FL_RunContent( ... "allowNetworking", "none", ...)
```

다음 API는 allowNetworking이 "internal"로 설정된 경우 사용할 수 없습니다.

navigateToURL(), fscommand(), ExternalInterface.call()

이전 목록의 API뿐 아니라 allowNetworking이 "none"으로 설정된 경우 다음 API도 사용할 수 없습니다.

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()

선택한 allowNetworking 설정에서 SWF 파일이 네트워킹 API를 사용할 수 있도록 허용한 경우에도 보안 샌드박스 제한에 따른 다른 유형의 제한이 있을 수 있습니다(950페이지의 “보안 샌드박스” 참조).

allowNetworking이 "none"으로 설정된 경우에는 TextField 객체의 htmlText 속성에 있는 <img> 태그에서 외부 미디어를 참조할 수 없습니다(SecurityError 예외 발생).

allowNetworking이 "none"으로 설정된 경우 가져와서 Flash Professional(ActionScript가 아님)에 추가한 공유 라이브러리의 심볼이 런타임 시 차단됩니다.

전체 화면 모드 보안

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 9.0.27.0 이상 버전에서는 Flash Player에서 실행되는 내용으로 전체 화면을 채울 수 있는 전체 화면 모드를 지원합니다. 전체 화면 모드를 사용하려면 스테이지의 displayState 속성을 StageDisplayState.FULL_SCREEN 상수로 설정합니다. 자세한 내용은 151페이지의 “전체 화면 모드 작업”을 참조하십시오.

원격 샌드박스에서 실행되는 SWF 파일에는 몇 가지의 보안 고려 사항이 있습니다.

전체 화면 모드를 사용하려면 다음 예제에 표시된 대로 SWF 파일에 대한 참조를 포함하는 HTML 페이지의 <object> 및 <embed> 태그에 값이 "true"(기본값은 "false")로 설정된 allowFullScreen 매개 변수를 추가합니다.

```
<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>
```

또한 HTML 페이지는 스크립트를 사용하여 SWF를 포함하는 태그를 생성할 수 있습니다. 해당 스크립트를 변경하여 올바른 allowFullScreen 설정을 삽입해야 합니다. Flash Professional 및 Flash Builder에서 생성된 HTML 페이지는 AC_FL_RunContent() 함수를 사용하여 SWF 파일에 대한 참조를 포함하므로 사용자는 다음과 같이 allowFullScreen 매개 변수 설정을 추가해야 합니다.

```
AC_FL_RunContent( ... "allowFullScreen", "true", ...)
```

전체 화면 모드를 시작하는 ActionScript는 마우스 이벤트나 키보드 이벤트에 대한 응답으로만 호출할 수 있습니다. 기타의 경우에 호출되면 예외가 발생합니다.

내용이 전체 화면 모드로 변경되면 사용자에게 종료 방법과 일반 모드로 돌아가는 방법을 알리는 메시지가 나타납니다. 이 메시지는 몇 초 동안 표시된 후 사라집니다.

브라우저에서 실행되는 내용의 경우 전체 화면 모드에서는 키보드 사용이 제한됩니다. Flash Player 9에서는 Esc 키와 같이 응용 프로그램을 보통 모드로 되돌리는 키보드 단축키만 지원됩니다. 사용자는 텍스트 필드에 텍스트를 입력하거나 화면을 탐색할 수 없습니다. Flash Player 10 이상에서는 일부 인쇄되지 않는 키(예: 화살표 키, 스페이스바, Tab 키)가 지원됩니다. 그러나 텍스트는 여전히 입력할 수 없습니다.

독립 실행형 플레이어나 프로젝터 파일에서는 전체 화면 모드가 항상 허용됩니다. 또한 이러한 환경에서는 텍스트 입력을 포함한 키보드 사용이 완전히 지원됩니다.

Stage 객체의 displayState 속성을 호출하면 스테이지 소유자(기본 SWF 파일)와 다른 보안 샌드박스에 있는 호출자에 대해서는 예외가 발생합니다. 자세한 내용은 968페이지의 “스테이지 보안”을 참조하십시오.

관리자는 mms.cfg 파일에서 FullScreenDisable = 1을 설정하여 브라우저에 실행 중인 SWF 파일에 대한 전체 화면 모드를 비활성화할 수 있습니다. 자세한 내용은 954페이지의 “관리자 컨트롤”을 참조하십시오.

브라우저에서 전체 화면 모드를 사용하려면 HTML 페이지에 SWF 파일이 포함되어 있어야 합니다.

전체 화면 상호 작용 모드 보안

Flash Player 11.3 이상, Adobe AIR 1.0 이상

Flash Player 11.3 이상에서는 전체 화면 상호 작용 모드를 지원합니다. 이 모드에서는 Flash Player에서 실행되는 내용으로 전체 화면을 채우고 텍스트 입력을 허용할 수 있습니다. 전체 화면 상호 작용 모드를 사용하려면 스테이지의 displayState 속성을 StageDisplayState.FULL_SCREEN_INTERACTIVE 상수로 설정합니다. 자세한 내용은 151페이지의 “전체 화면 모드 작업”을 참조하십시오.

원격 샌드박스에서 실행되는 SWF 파일에는 몇 가지의 보안 고려 사항이 있습니다.

전체 화면 모드를 사용하려면 다음 예제에 표시된 대로 SWF 파일에 대한 참조를 포함하는 HTML 페이지의 <object> 및 <embed> 태그에 allowFullScreenInteractive 매개 변수(값을 "true"로 설정(기본값은 "false"))를 추가합니다.

```
<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>
```

또한 HTML 페이지는 스크립트를 사용하여 SWF를 포함하는 태그를 생성할 수 있습니다. 올바른 allowFullScreenInteractive 설정이 삽입되도록 스크립트를 변경해야 합니다. Flash Professional 및 Flash Builder에서 생성한 HTML 페이지는 AC_FL_RunContent() 함수를 사용하여 SWF 파일에 대한 참조를 포함하므로 다음과 같이 allowFullScreenInteractive 매개 변수 설정을 추가해야 합니다.

```
AC_FL_RunContent( ... "allowFullScreenInteractive", "true", ...)
```

전체 화면 상호 작용 모드를 시작하는 ActionScript는 마우스 이벤트나 키보드 이벤트에 대한 응답으로만 호출할 수 있습니다. 기타의 경우에 호출되면 예외가 발생합니다.

내용에 전체 화면 상호 작용 모드가 적용되면 오버레이 메시지가 표시됩니다. 메시지에는 전체 화면 페이지의 도메인, 전체 화면 모드를 종료하는 방법에 대한 지침, **허용** 버튼이 표시됩니다. 오버레이는 사용자가 **허용**을 클릭하여 전체 화면 상호 작용 모드 사용을 승인할 때까지 계속 유지됩니다.

관리자는 mms.cfg 파일에서 FullScreenInteractiveDisable = 1을 설정하여 브라우저에서 실행되는 SWF 파일에 대해 전체 화면 상호 작용 모드를 비활성화할 수 있습니다. 자세한 내용은 954페이지의 “관리자 컨트롤”을 참조하십시오.

브라우저에서 전체 화면 상호 작용 모드를 허용하려면 HTML 페이지에 SWF 파일이 포함되어 있어야 합니다.

내용 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 AIR 내용은 다음과 같은 여러 유형의 내용을 로드할 수 있습니다.

- SWF 파일
- 이미지
- 사운드
- 비디오
- HTML 파일(AIR에만 해당)
- JavaScript(AIR에만 해당)

Loader 클래스로 SWF 파일 및 이미지 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

Loader 클래스를 사용하여 SWF 파일 및 이미지(JPG, GIF, 또는 PNG 파일)를 로드할 수 있습니다. local-with-filesystem 샌드박스에 있는 SWF 파일을 제외한 모든 SWF 파일에서는 모든 네트워크 도메인에 있는 SWF 파일과 이미지를 로드할 수 있습니다. 로컬 샌드박스의 SWF 파일만 로컬 파일 시스템의 SWF 파일과 이미지를 로드할 수 있습니다. 하지만 local-with-

networking 샌드박스에 있는 파일은 local-trusted 또는 local-with-networking 샌드박스에 있는 로컬 SWF 파일만 로드할 수 있습니다. local-with-networking 샌드박스에 있는 SWF 파일은 이미지와 같이 SWF 파일이 아닌 로컬 내용을 로드할 수 있지만 로드된 내용의 데이터에는 액세스할 수 없습니다.

신뢰할 수 없는 소스(예: Loader 객체의 루트 SWF 파일에 대한 도메인이 아닌 도메인)에서 SWF 파일을 로드하는 경우, 다음 코드와 같이 Loader 객체의 마스크를 정의하여 해당 마스크의 바깥쪽 스테이지 부분에 로드된 내용(Loader 객체의 자식)을 그릴 수 없도록 할 수 있습니다.

```
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);
```

Loader 객체의 load() 메서드를 호출하는 경우에는 LoaderContext 객체인 context 매개 변수를 지정할 수 있습니다. LoaderContext 클래스에는 로드된 내용을 사용하는 방법에 대한 컨텍스트를 정의할 수 있는 세 가지 속성이 포함되어 있습니다.

- checkPolicyFile: 이 속성은 SWF 파일이 아닌 이미지 파일을 로드할 때만 사용합니다. Loader 객체를 포함하는 파일의 도메인을 제외한 도메인의 이미지 파일에 대해 이 속성을 정의합니다. 이 속성을 true로 설정하면 Loader는 원래 서버에서 URL 정책 파일을 확인합니다(956페이지의 "웹 사이트 컨트롤(정책 파일)" 참조). 서버에서 Loader 도메인에 대한 권한을 부여하면 Loader 도메인에 있는 SWF 파일의 ActionScript에서 로드된 이미지의 데이터에 액세스할 수 있습니다. 즉, Loader.content 속성을 사용하여 로드된 이미지를 나타내는 Bitmap 객체나, 로드된 이미지의 픽셀에 액세스하는 BitmapData.draw() 또는 BitmapData.drawWithQuality() 메서드에 대한 참조를 가져올 수 있습니다. drawWithQuality 메서드는 Flash Player 11.3 이상, AIR 3.3 이상에서 사용할 수 있습니다.
- securityDomain: 이 속성은 이미지가 아닌 SWF 파일을 로드할 때만 사용합니다. Loader 객체를 포함하는 파일의 도메인과 다른 도메인에서의 SWF 파일에 대해 이 속성을 지정합니다. securityDomain 속성의 경우 현재 null(기본값) 및 SecurityDomain.currentDomain의 두 값만 지원됩니다. SecurityDomain.currentDomain을 지정하면 로드된 SWF 파일이 로드하는 SWF 파일의 샌드박스로 가져와서 로드하는 SWF 파일의 자체 서버에서 로드된 것처럼 작동합니다. 이 기능은 로드된 SWF 파일 서버에 URL 정책 파일이 있고 로드하는 SWF 파일의 도메인에서 액세스할 수 있는 경우에만 허용됩니다. 필요한 정책 파일이 있는 경우에는 로드가 시작될 때 로더와 로드되는 파일에서 자유롭게 서로 스크립팅할 수 있는데, 이는 이 두 항목이 동일한 샌드박스에 있기 때문입니다. 대부분의 샌드박스 가져오기 작업은 일반적인 로드를 수행한 다음 로드된 SWF 파일에서 Security.allowDomain() 메서드를 호출하는 작업으로 바꿀 수 있습니다. 로드된 SWF 파일이 자체의 고유 샌드박스에 있게 되어 SWF 파일의 실제 서버에 있는 리소스에 액세스할 수 있기 때문에 두 번째 방법이 더욱 사용하기 쉽습니다.
- applicationDomain: 이 속성은 ActionScript 3.0으로 작성된 SWF 파일을 로드할 때만 사용합니다. 이미지 파일이나 ActionScript 1.0 또는 2.0으로 작성된 SWF 파일은 해당하지 않습니다. 파일을 로드할 때 사용자는 해당 파일을 로드하는 SWF 파일 응용 프로그램 도메인의 자식인 새 응용 프로그램 도메인(기본 위치)이 아닌 특정 응용 프로그램 도메인에 배치되도록 지정할 수 있습니다. 이때, 응용 프로그램 도메인은 보안 도메인의 하위 개념이며, 따라서 이는 로드하는 SWF 파일을 자체 서버에서 가져왔거나, 또는 securityDomain 속성을 사용하여 자체 보안 도메인으로 성공적으로 가져온 경우 즉, 로드하는 SWF 파일이 자체 보안 도메인에 있는 경우에만 대상 응용 프로그램 도메인을 지정할 수 있습니다. 응용 프로그램 도메인을 지정했으나 로드된 SWF 파일이 다른 보안 도메인에 속한 경우에는 applicationDomain에 지정된 도메인은 무시됩니다. 자세한 내용은 132페이지의 "응용 프로그램 도메인 작업"을 참조하십시오.

자세한 내용은 180페이지의 "로드 컨텍스트 지정"을 참조하십시오.

Loader 객체의 중요한 속성은 LoaderInfo 객체인 contentLoaderInfo 속성입니다. 다른 대부분의 객체와 달리 LoaderInfo 객체는 로드하는 SWF 파일과 로드된 내용 간에 공유되며 항상 양쪽에서 모두 액세스할 수 있습니다. 로드된 내용이 SWF 파일인 경우 DisplayObject.loaderInfo 속성을 통해 LoaderInfo 객체에 액세스할 수 있습니다. LoaderInfo 객체에는 로드 진행률, 로더 및 로드된 내용의 URL, 로더와 로드된 내용 간의 신뢰 관계 및 기타 정보가 포함됩니다. 자세한 내용은 179페이지의 "로드 진행률 모니터링"을 참조하십시오.

사운드 및 비디오 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

local-with-filesystem 샌드박스에 있는 내용을 제외한 모든 내용은 Sound.load(), NetConnection.connect() 및 NetStream.play() 메서드를 사용하여 네트워크에서 사운드와 비디오를 로드할 수 있습니다.

local-with-filesystem 및 AIR 응용 프로그램 샌드박스의 내용만 로컬 파일 시스템에서 미디어를 로드할 수 있습니다. local-with-filesystem 샌드박스, AIR 응용 프로그램 샌드박스 또는 local-trusted 샌드박스에 있는 내용만 해당 로드된 파일의 데이터에 액세스할 수 있습니다.

로드된 미디어의 데이터에 액세스하는 데는 몇 가지 제한 사항이 있습니다. 자세한 내용은 970페이지의 "데이터로 로드된 미디어 액세스"를 참조하십시오.

텍스트 필드에서 <img> 태그를 사용하여 SWF 파일 및 이미지 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

다음 코드와 같이 <img> 태그를 사용하여 텍스트 필드로 SWF 파일과 비트맵을 로드할 수 있습니다.

```
<img src = 'filename.jpg' id = 'instanceName' >
```

다음 코드와 같이 TextField 인스턴스의 getImageReference() 메서드를 사용하여 위와 같은 방식으로 로드된 내용에 액세스할 수 있습니다.

```
var loadedObject:DisplayObject = myTextField.getImageReference('instanceName');
```

하지만 이렇게 로드된 SWF 파일과 이미지는 원래 위치에 있는 샌드박스에 배치됩니다.

텍스트 필드에서 <img> 태그를 사용하여 이미지를 로드할 때, 이미지의 데이터에 대한 액세스는 URL 정책 파일에서 허용됩니다. 다음 코드와 같이 <img> 태그에 checkPolicyFile 속성을 추가하여 정책 파일을 확인할 수 있습니다.

```
<img src = 'filename.jpg' checkPolicyFile = 'true' id = 'instanceName' >
```

텍스트 필드에서 <img> 태그를 사용하여 SWF를 로드할 때 Security.allowDomain() 메서드를 호출하여 SWF 파일의 데이터에 대한 액세스를 허용할 수 있습니다.

텍스트 필드의 <img> 태그를 사용하여 외부 파일을 로드하는 경우(SWF 파일에 포함된 Bitmap 클래스를 사용하는 경우와는 반대), Loader 객체가 자동으로 TextField 객체의 자식으로 생성되고 ActionScript의 Loader 객체를 사용하여 파일을 로드한 것과 마찬가지로 외부 파일이 해당 Loader에 로드됩니다. 이 경우 getImageReference() 메서드에서 자동으로 생성된 Loader를 반환합니다. 호출하는 코드와 동일한 보안 샌드박스에 있기 때문에 이 Loader 객체에 액세스할 때는 별도의 보안 확인이 필요하지 않습니다.

하지만 Loader 객체의 content 속성을 참조하여 로드된 미디어에 액세스하는 경우에는 보안 규칙이 적용됩니다. 내용이 이미지일 경우에는 URL 정책 파일을 구현해야 하고, 내용이 SWF 파일인 경우에는 SWF 파일에 allowDomain() 메서드를 호출하는 코드가 있어야 합니다.

Adobe AIR

응용 프로그램 샌드박스에서 텍스트 필드에 있는 <img> 태그는 피싱 공격을 방지하기 위해 무시됩니다. 또한 응용 프로그램 샌드박스에서 실행되는 코드는 Security allowDomain() 메서드를 호출하도록 허용되지 않습니다.

RTMP 서버를 사용하여 제공된 내용

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Media Server는 RTMP(Real-Time Media Protocol)를 사용하여 데이터, 오디오 및 비디오를 제공합니다. RTMP URL을 매개 변수로 전달하고 NetConnection 클래스의 connect() 메서드를 사용하여 이 미디어를 로드할 수 있습니다. Flash Media Server는 요청하는 파일의 도메인을 기준으로 연결을 제한하고 내용 다운로드를 금지할 수 있습니다. 자세한 내용은 www.adobe.com/go/learn_fms_docs_kr에서 온라인으로 제공되는 Flash Media Server 설명서를 참조하십시오.

BitmapData.draw(), BitmapData.drawWithQuality() 및 SoundMixer.computeSpectrum() 메서드를 사용하여 RTMP 스트림에서 런타임 그래픽과 사운드 데이터를 추출하려면 해당 서버에서 액세스를 허용해야 합니다. 서버측 ActionScript Client.videoSampleAccess 및 Client.audioSampleAccess 속성을 사용하여 Flash Media Server에 있는 특정 디렉토리에 대한 액세스를 허용합니다. 자세한 내용은 서버측 ActionScript 언어 참조 설명서를 참조하십시오. (drawWithQuality 메서드는 Flash Player 11.3 이상, AIR 3.3 이상에서 사용할 수 있습니다.)

크로스 스크립팅

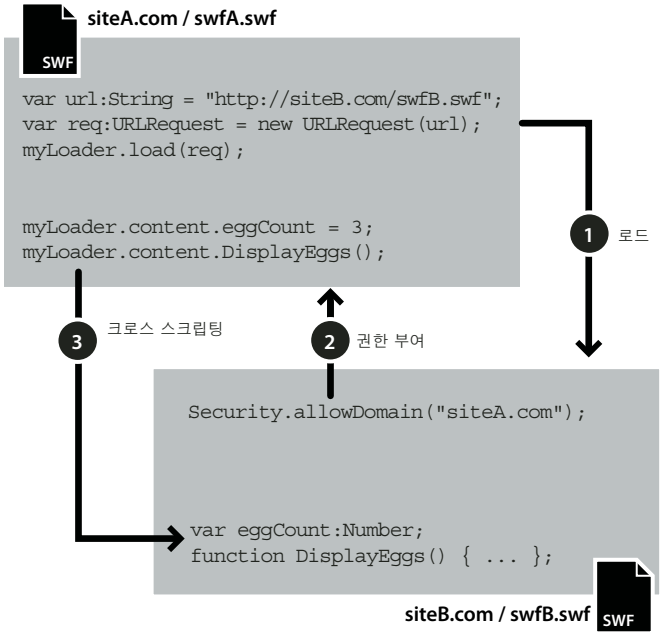
Flash Player 9 이상, Adobe AIR 1.0 이상

동일한 도메인에서 ActionScript 3.0으로 작성된 두 개의 SWF 파일 또는 AIR에서 실행되는 두 개의 HTML 파일을 제공하는 경우에는(예: 한 SWF 파일의 URL은 http://www.example.com/swfA.swf이고 다른 SWF 파일의 URL은 http://www.example.com/swfB.swf인 경우) 한 파일에 정의된 코드에서 다른 파일에 정의된 코드의 변수, 객체, 속성, 메서드 및 기타 항목을 검사 및 수정할 수 있습니다. 이를 **크로스 스크립팅**이라고 합니다.

두 파일이 서로 다른 도메인에서 제공되는 경우(예: http://siteA.com/swfA.swf 및 http://siteB.com/swfB.swf) 기본적으로 Flash Player 및 AIR에서는 swfA.swf가 swfB.swf를 스크립팅하거나 swfB.swf가 swfA.swf를 스크립팅하도록 허용하지 않습니다. SWF 파일은 Security.allowDomain()을 호출하여 다른 도메인의 SWF 파일에 대한 권한을 제공합니다. Security.allowDomain("siteA.com")을 호출하여 swfB.swf는 siteA.com의 SWF 파일에 스크립팅 권한을 부여합니다.

크로스 스크립팅은 AVM1 SWF 파일과 AVM2 SWF 파일 간에는 지원되지 않습니다. AVM1 SWF 파일은 ActionScript 1.0 또는 ActionScript 2.0을 사용하여 작성됩니다. (AVM1 및 AVM2는 ActionScript Virtual Machine을 나타냅니다.) 하지만 LocalConnection 클래스를 사용하여 AVM1과 AVM2 간에 데이터를 보낼 수 있습니다.

크로스 도메인 상황에서는 관련되는 두 도메인을 명확하게 구별하는 것이 중요합니다. 이 설명서에서는 크로스 스크립팅을 수행하는 쪽을 액세스하는 항목(일반적으로 액세스하는 SWF)이라고 하고, 다른 한 쪽을 액세스되는 항목(일반적으로 액세스되는 SWF)이라고 합니다. 다음 그림에 표시된 대로 siteA.swf에서 siteB.swf를 스크립팅할 때 siteA.swf는 액세스하는 항목이고 siteB.swf는 액세스되는 항목입니다.



Security.allowDomain() 메서드로 설정된 크로스 도메인 권한은 비대칭적입니다. 앞의 예제에서 siteA.swf는 siteB.swf를 스크립팅할 수 있지만 siteB.swf는 siteA.swf를 스크립팅할 수 없습니다. 이는 siteA.swf에서 Security.allowDomain() 메서드를 호출하여 siteB.com에 있는 SWF 파일에 스크립팅 권한을 부여하지 않았기 때문입니다. 양쪽 SWF 파일이 Security.allowDomain() 메서드를 호출하도록 하여 대칭적 권한을 설정할 수 있습니다.

Flash Player에서 SWF 파일은 다른 SWF 파일의 크로스 도메인 스크립팅으로부터 보호될 뿐 아니라 HTML 파일의 크로스 도메인 스크립팅으로부터도 보호됩니다. HTML에서 SWF로의 스크립팅은 ExternalInterface.addCallback() 메서드를 통해 설정된 콜백에서 발생할 수 있습니다. HTML에서 SWF로의 스크립팅이 다른 도메인 간에 발생하는 경우 액세스하는 항목이 SWF 파일일 때와 마찬가지로 액세스되는 SWF 파일에서 Security.allowDomain() 메서드를 호출해야 하며, 그렇지 않으면 작업이 실패합니다. 자세한 내용은 959페이지의 "제작자(개발자) 컨트롤"을 참조하십시오.

또한 Flash Player는 SWF에서 HTML로의 스크립팅에 대해 보안 컨트롤을 제공합니다. 자세한 내용은 976페이지의 "아웃바운드 URL 액세스 제어"를 참조하십시오.

스테이지 보안

Flash Player 9 이상, Adobe AIR 1.0 이상

Stage 객체의 일부 속성 및 메서드를 표시 목록에 있는 모든 sprite 또는 동영상 클립에서 사용할 수 있습니다.

그러나 Stage 객체는 첫 번째 SWF 파일이 로드된 경우에 소유자가 있는 것으로 간주됩니다. 기본적으로 다음과 같은 Stage 객체의 속성 및 메서드는 스테이지 소유자와 동일한 보안 샌드박스에 있는 SWF 파일에만 사용할 수 있습니다.

속성	메서드
align	addChild()
displayState	addChildAt()
frameRate	addEventListener()
height	dispatchEvent()
mouseChildren	hasEventListener()
numChildren	setChildIndex()
quality	willTrigger()
scaleMode	
showDefaultContextMenu	
stageFocusRect	
stageHeight	
stageWidth	
tabChildren	
textSnapshot	
width	

스테이지 소유자의 샌드박스가 아닌 샌드박스에 있는 SWF 파일에서 위의 속성과 메서드에 액세스하려면 스테이지 소유자 SWF 파일에서 Security.allowDomain() 메서드를 호출하여 외부 샌드박스의 도메인을 허용해야 합니다. 자세한 내용은 959페이지의 "제작자(개발자) 컨트롤"을 참조하십시오.

frameRate 속성은 특별한 경우로 모든 SWF 파일에서 frameRate 속성을 읽을 수 있습니다. 그러나 스테이지 소유자의 보안 샌드박스에 있거나 Security.allowDomain() 메서드를 호출하여 권한을 부여한 SWF 파일에서만 이 속성을 변경할 수 있습니다.

Stage 객체의 removeChildAt() 및 swapChildrenAt() 메서드에도 제한 사항이 있지만 다른 제한 사항과는 다릅니다. 이 메서드를 호출하려면 코드가 스테이지 소유자와 동일한 도메인이 아니라 영향 받는 자식 객체 또는 Security.allowDomain() 메서드를 호출할 수 있는 자식 객체의 소유자와 동일한 도메인에 있어야 합니다.

표시 목록 순회

Flash Player 9 이상, Adobe AIR 1.0 이상

다른 샌드박스에서 로드된 표시 객체에 액세스하는 SWF 파일의 기능은 제한적입니다. SWF 파일에서 다른 샌드박스에 있는 다른 SWF 파일에서 생성된 표시 객체에 액세스하려면 액세스되는 SWF 파일에서 Security.allowDomain() 메서드를 호출하여 액세스하는 SWF 파일의 도메인에 액세스를 허용해야 합니다. 자세한 내용은 959페이지의 "제작자(개발자) 컨트롤"을 참조하십시오.

Loader 객체에서 로드한 Bitmap 객체에 액세스하려면 해당 이미지 파일의 원래 서버에 URL 정책 파일이 있어야 하고 해당 크로스 도메인 정책 파일에서 Bitmap 객체에 액세스를 시도하는 SWF 파일의 도메인에 액세스 권한을 부여해야 합니다(956페이지의 "웹 사이트 컨트롤(정책 파일)" 참조).

로드된 파일 및 Loader 객체에 해당하는 LoaderInfo 객체에는 로드된 객체와 Loader 객체 간의 관계를 정의하는 childAllowsParent, parentAllowsChild 및 sameDomain의 세 가지 속성이 있습니다.

이벤트 보안

Flash Player 9 이상, Adobe AIR 1.0 이상

표시 목록과 연관된 이벤트에는 해당 이벤트를 전달하는 표시 객체의 샌드박스를 기준으로 보안 액세스 제한이 적용됩니다. 표시 목록의 이벤트에는 버블링과 캡처 단계가 있습니다(113페이지의 "이벤트 처리" 참조). 버블링 및 캡처 단계에서 이벤트는 소스 표시 객체에서 표시 목록의 부모 표시 객체를 통해 마이그레이션됩니다. 부모 객체가 소스 표시 객체와 다른 보안 샌드박스에 있는 경우에는 부모 객체 소유자와 소스 객체 소유자 간에 상호 신뢰가 없는 한 해당 부모 객체 아래에서 캡처 및 버블링 단계가 중단됩니다. 상호 신뢰는 다음 작업을 통해 얻을 수 있습니다.

1. 부모 객체를 소유한 SWF 파일에서 Security.allowDomain() 메서드를 호출하여 소스 객체를 소유한 SWF 파일의 도메인을 신뢰해야 합니다.
2. 소스 객체를 소유한 SWF 파일에서 Security.allowDomain() 메서드를 호출하여 부모 객체를 소유한 SWF 파일의 도메인을 신뢰해야 합니다.

로드된 파일 및 Loader 객체에 해당하는 LoaderInfo 객체에는 로드된 객체와 Loader 객체 간의 관계를 정의하는 childAllowsParent 및 parentAllowsChild의 두 가지 속성이 있습니다.

표시 객체 이외의 객체에서 전달된 이벤트에 대해서는 보안 확인이나 보안 관련 지정이 없습니다.

데이터로 로드된 미디어 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

로드 데이터에 액세스하려면 BitmapData.draw(), BitmapData.drawWithQuality() 및 SoundMixer.computeSpectrum() 메서드를 사용합니다. 기본적으로 다른 샌드박스에서 로드된 미디어로 렌더링 또는 재생되는 그래픽 또는 오디오 객체에서는 픽셀 데이터 또는 오디오 데이터를 가져올 수 없습니다. 그러나 다음 메서드를 사용하면 샌드박스 경계에 걸쳐 해당 데이터에 대한 액세스 권한을 부여할 수 있습니다.

- 내용 렌더링 또는 액세스할 데이터 재생 시 다른 도메인의 내용에 대한 데이터 액세스 권한을 부여하려면 Security.allowDomain() 메서드를 호출합니다.
- 로드된 이미지, 사운드 또는 비디오의 경우 로드된 파일의 서버에 URL 정책 파일을 추가합니다. 이 정책 파일에서 BitmapData.draw(), BitmapData.drawWithQuality() 또는 SoundMixer.computeSpectrum() 메서드 호출을 시도하는 SWF 파일의 도메인에 대한 액세스 권한을 부여해야 해당 파일에서 데이터를 추출할 수 있습니다. drawWithQuality 메서드는 Flash Player 11.3 이상, AIR 3.3 이상에서 사용할 수 있습니다.

다음 단원에서는 비트맵, 사운드 및 비디오 데이터 액세스에 대해 자세히 설명합니다.

비트맵 데이터 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

BitmapData 객체의 draw() 및 drawWithQuality()(Flash Player 11.3, AIR 3.3) 메서드를 사용하면 모든 표시 객체의 현재 표시된 픽셀을 BitmapData 객체에 그릴 수 있습니다. 여기에는 MovieClip 객체, Bitmap 객체 또는 모든 표시 객체의 픽셀이 포함될 수 있습니다. 다음 조건이 충족되어야만 이러한 메서드로 BitmapData 객체에 픽셀을 그릴 수 있습니다.

- 로드된 비트맵이 아닌 소스 객체의 경우, 소스 객체와 (Sprite 또는 MovieClip 객체의 경우) 모든 자식 객체가 draw 메서드를 호출하는 객체와 동일한 도메인에 있거나, Security.allowDomain() 메서드를 호출하여 해당 호출자에 액세스할 수 있는 SWF 파일에 있어야 합니다.
- 로드된 비트맵 소스 객체의 경우에는 소스 객체가 draw 메서드를 호출하는 객체와 동일한 도메인에 있거나, 해당 소스 서버에 호출하는 도메인에 권한을 부여하는 URL 정책 파일이 포함되어 있어야 합니다.

이러한 조건이 충족되지 않으면 SecurityError 예외가 발생합니다.

Loader 클래스의 load() 메서드를 사용하여 이미지를 로드하는 경우 LoaderContext 객체인 context 매개 변수를 지정할 수 있습니다. LoaderContext 객체의 checkPolicyFile 속성을 true로 설정하면 Flash Player에서 이미지가 로드되는 서버의 URL 정책 파일을 확인합니다. 정책 파일이 있고 이 파일에서 로드하는 SWF 파일의 도메인을 허용할 경우, 해당 SWF 파일은 Bitmap 객체의 데이터에 액세스할 수 있고, 그렇지 않으면 액세스할 수 없습니다.

또한 텍스트 필드의 <img> 태그를 통해 로드된 이미지의 checkPolicyFile 속성을 지정할 수 있습니다. 자세한 내용은 966페이지의 "텍스트 필드에서 <img> 태그를 사용하여 SWF 파일 및 이미지 로드"를 참조하십시오.

사운드 데이터 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

다음과 같은 사운드 관련 ActionScript 3.0 API에는 보안 제한 사항이 있습니다.

- SoundMixer.computeSpectrum() 메서드 - 사운드 파일과 동일한 보안 샌드박스에서 실행되는 코드를 항상 허용합니다. 다른 샌드박스에서 실행되는 코드에 대해서는 보안 확인을 실행합니다.
- SoundMixer.stopAll() 메서드 - 사운드 파일과 동일한 보안 샌드박스에서 실행되는 코드를 항상 허용합니다. 다른 샌드박스의 파일에 대해서는 보안 확인을 실행합니다.
- Sound 클래스의 id3 속성 - 사운드 파일과 동일한 보안 샌드박스에 있는 SWF 파일을 항상 허용합니다. 다른 샌드박스에서 실행되는 코드에 대해서는 보안 확인을 실행합니다.

모든 사운드에는 내용 샌드박스와 소유자 샌드박스의 두 가지 관련 샌드박스가 있습니다.

- 사운드의 원래 도메인에 의해 내용 샌드박스가 결정되며 여기에서 사운드의 id3 속성과 SoundMixer.computeSpectrum() 메서드를 통해 사운드에서 데이터를 추출할 수 있는지 여부가 결정됩니다.
- 사운드 재생을 시작한 객체에 의해 소유자 샌드박스가 결정되며 여기에서 SoundMixer.stopAll() 메서드를 사용하여 사운드를 중단할 수 있는지 여부가 결정됩니다.

Sound 클래스의 load() 메서드를 사용하여 사운드를 로드하는 경우 SoundLoaderContext 객체인 context 매개 변수를 지정할 수 있습니다. SoundLoaderContext 객체의 checkPolicyFile 속성을 true로 설정하면 런타임에서 사운드가 로드되는 서버에 URL 정책 파일이 있는지 확인합니다. 정책 파일이 있고 이 파일이 로드하는 코드의 도메인을 허용할 경우, 해당 코드는 Sound 객체의 id 속성에 액세스할 수 있고, 그렇지 않으면 액세스할 수 없습니다. 또한 checkPolicyFile 속성을 설정하여 로드된 사운드에 SoundMixer.computeSpectrum() 메서드를 사용할 수 있습니다.

하나 이상의 사운드 소유자 샌드박스를 호출자에서 액세스할 수 없어 SoundMixer.stopAll() 메서드 호출 시 모든 사운드가 중단되지 않는지 여부를 SoundMixer.areSoundsInaccessible() 메서드를 사용하여 확인할 수 있습니다.

SoundMixer.stopAll() 메서드를 호출하면 stopAll()의 호출자와 동일한 소유자 샌드박스에 있는 사운드를 중단할 수 있습니다. 또한 Security.allowDomain() 메서드를 호출한 SWF 파일에서 재생을 시작한 사운드를 중단하여 stopAll() 메서드를 호출하는 SWF 파일의 도메인에 액세스를 허용할 수 있습니다. 다른 사운드는 중단되지 않으며, 이 사운드는 SoundMixer.areSoundsInaccessible() 메서드를 호출하여 확인할 수 있습니다.

computeSpectrum() 메서드를 호출하려면 재생되는 모든 사운드가 해당 메서드를 호출하는 객체와 동일한 샌드박스에 있거나 또는 호출자의 샌드박스에 액세스 권한을 부여한 소스에 있어야 합니다. 그렇지 않으면 SecurityError 예외가 발생합니다. SWF 파일의 라이브러리에 포함된 사운드에서 로드된 사운드의 경우, 로드된 SWF 파일에서 Security.allowDomain() 메서드를 호출하면 권한이 부여됩니다. SWF 파일이 아닌 소스에서 로드된 사운드의 경우(로드된 mp3 파일 또는 비디오 파일에서 시작), 소스 서버의 URL 정책 파일에서 로드된 미디어의 데이터에 대한 액세스 권한을 부여합니다.

자세한 내용은 959페이지의 "제작자(개발자) 컨트롤" 및 956페이지의 "웹 사이트 컨트롤(정책 파일)"을 참조하십시오.

RTMP 스트림에서 사운드 데이터에 액세스하려면 서버에 대한 액세스를 허용해야 합니다. 서버측 ActionScript Client.audioSampleAccess 속성을 사용하여 Flash Media Server에 있는 특정 디렉토리에 대한 액세스를 허용합니다. 자세한 내용은 서버측 ActionScript 언어 참조 설명서를 참조하십시오.

비디오 데이터 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

BitmapData.draw() 또는 BitmapData.drawWithQuality() 메서드를 사용하여 현재 비디오 프레임의 픽셀 데이터를 캡처할 수 있습니다. (drawWithQuality 메서드는 Flash Player 11.3 이상, AIR 3.3 이상에서 사용할 수 있습니다.)

다음과 같은 두 종류의 비디오가 있습니다.

- Flash Media Server에서 RTMP를 통해 스트리밍되는 비디오
- FLV 또는 F4V 파일에서 로드되는 점진적 비디오

draw 메서드를 사용하여 RTMP 스트림에서 런타임 그래픽을 추출하려면 해당 서버에서 액세스를 허용해야 합니다. 서버측 ActionScript Client.videoSampleAccess 속성을 사용하여 Flash Media Server에 있는 특정 디렉토리에 대한 액세스를 허용합니다. 자세한 내용은 서버측 ActionScript 언어 참조 설명서를 참조하십시오.

점진적 비디오를 source 매개 변수로 하여 draw 메서드를 호출하는 경우 해당 메서드 호출자는 FLV 파일과 동일한 샌드박스에 있거나, FLV 파일의 서버에 호출하는 SWF 파일의 도메인에 권한을 부여하는 정책 파일이 있어야 합니다. NetStream 객체의 checkPolicyFile 속성을 true로 설정하여 해당 정책 파일의 다운로드를 요청할 수 있습니다.

데이터 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 및 AIR 내용은 서버와 데이터를 교환할 수 있습니다. 로드된 데이터가 미디어로 표시되지 않고 프로그램 객체로 직접 표시되므로, 데이터 로드 작업은 미디어 로드 작업과는 다릅니다. 일반적으로 내용은 내용이 시작된 도메인에서 데이터를 로드할 수 있습니다. 그러나 일반적으로 다른 도메인에서 데이터를 로드하려면 내용에 정책 파일이 필요합니다. 자세한 내용은 956 페이지의 "웹 사이트 컨트롤(정책 파일)"을 참조하십시오.

참고: AIR 응용 프로그램 샌드박스에서 실행되는 내용은 개발자가 의도적으로 원격 내용을 응용 프로그램 샌드박스로 가져오지 않는 한 원격 도메인에서 제공되지 않습니다. 따라서 정책 파일에서 방지하도록 설정된 공격 유형에 참여할 수 없습니다. 응용 프로그램 샌드박스의 AIR 내용은 정책 파일에서 데이터를 로드하는 데 제한이 없습니다. 그러나 다른 샌드박스에 있는 AIR 내용은 여기에 설명된 제한이 적용됩니다.

URLLoader 및 URLStream 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

XML 파일이나 텍스트 파일과 같은 데이터를 로드할 수 있습니다. URLLoader 및 URLStream 클래스의 load() 메서드는 URL 정책 파일 권한으로 제어합니다.

load() 메서드를 사용하여 해당 메서드를 호출하는 코드의 도메인이 아닌 다른 도메인의 내용을 로드하는 경우, 런타임은 로드되는 에셋의 서버에서 URL 정책 파일을 확인합니다. 정책 파일이 있으면 로드하는 내용의 도메인에 대한 액세스가 허용되고 데이터를 로드할 수 있습니다.

소켓 연결

Flash Player 9 이상, Adobe AIR 1.0 이상

기본적으로 런타임에서는 포트 843에서 제공되는 소켓 정책 파일을 찾습니다. URL 정책 파일과 마찬가지로 이 파일을 마스터 정책 파일이라고 합니다.

정책 파일이 처음으로 Flash Player 6에 도입되었을 때는 소켓 정책 파일이 지원되지 않았습니다. 따라서 소켓 서버에 대한 연결은 소켓 서버와 동일한 호스트의 포트 80에 있는 HTTP 서버의 기본 위치에 있는 정책 파일에 의해 허가되었습니다. Flash Player 9는 이 기능을 계속 지원하지만 Flash Player 10에서는 지원되지 않습니다. Flash Player 10에서는 소켓 정책 파일만 소켓 연결에 권한을 부여할 수 있습니다.

URL 정책 파일과 마찬가지로 소켓 정책 파일은 정책 파일을 서비스할 수 있는 포트를 지정하는 메타 정책 문을 지원합니다. 그러나 소켓 정책 파일에 대한 기본 메타 정책은 "master-only" 대신 "all"입니다. 즉, 마스터 정책 파일에서 보다 제한적인 설정을 지정하지 않는 한 Flash Player는 호스트의 모든 소켓에서 소켓 정책 파일을 제공할 수 있다고 가정합니다.

연결하는 소켓이 SWF 파일과 동일한 도메인에 있는 경우에도 소켓 및 XML 소켓 연결에 대한 액세스는 기본적으로 사용되지 않습니다. 다음 위치 중 하나에서 소켓 정책 파일을 제공하여 소켓 수준 액세스를 허용할 수 있습니다.

- 포트 843(마스터 정책 파일의 위치)
- 기본 소켓 연결과 동일한 포트
- 기본 소켓 연결과 다른 포트

기본적으로 Flash Player는 포트 843 및 기본 소켓 연결과 동일한 포트에서 소켓 정책 파일을 찾습니다. 다른 포트에서 소켓 정책 파일을 제공하려면 SWF 파일에서 Security.loadPolicyFile()을 호출해야 합니다.

소켓 정책 파일은 액세스가 허용되는 포트를 지정해야 한다는 점을 제외하고 URL 정책 파일과 동일한 구문을 사용합니다. 1024 보다 낮은 포트에서 제공되는 소켓 정책 파일에서는 모든 포트에 액세스를 허용할 수 있고, 1024 이상의 포트에서 제공되는 정책 파일에서는 1024 이상의 포트에만 액세스를 허용할 수 있습니다. 허용되는 포트는 <allow-access-from> 태그의 to-ports 특성에 지정됩니다. 단일 포트 번호, 포트 범위 및 와일드카드가 모두 허용되는 값입니다.

소켓 정책 파일의 예는 다음과 같습니다.

```
<?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>
```

포트 843이나 기본 소켓 연결과 동일한 포트에서 소켓 정책 파일을 검색하려면 Socket.connect() 또는 XMLSocket.connect() 메서드를 호출합니다. Flash Player는 먼저 포트 843에서 마스터 정책 파일을 확인합니다. 파일이 있으면 대상 포트의 소켓 정책 파일을 차단하는 메타 정책 문이 파일에 포함되어 있는지 확인합니다. 액세스가 차단되지 않은 경우 Flash Player는 먼저 마스터 정책 파일에서 적합한 allow-access-from 문을 찾습니다. 파일이 없으면 Flash Player는 주 소켓 연결과 동일한 포트에서 소켓 정책 파일을 찾습니다.

다른 위치에서 소켓 정책 파일을 검색하려면 먼저 다음과 같이 특수 "xmlsocket" 구문을 사용하여 Security.loadPolicyFile() 메서드를 호출합니다.

```
Security.loadPolicyFile("xmlsocket://server.com:2525");
```

Security.loadPolicyFile() 메서드를 Socket.connect() 또는 XMLSocket.connect() 메서드보다 먼저 호출합니다. 그러면 Flash Player은 정책 파일 요청이 수행될 때까지 기다렸다가 기본 연결을 허용할지 여부를 결정합니다. 그러나 대상 위치에서 정책 파일을 제공할 수 없도록 마스터 정책 파일에 지정되어 있으면 해당 위치에 정책 파일이 있는 경우에도 loadPolicyFile() 호출 시 아무 영향도 주지 않습니다.

소켓 서버를 구현하고 소켓 정책 파일을 제공해야 할 필요가 있는 경우에는 기본 연결을 허용하는 포트와 동일한 포트를 사용하여 정책 파일을 제공할지 아니면 다른 포트에서 정책 파일을 제공할지 여부를 결정합니다. 두 경우 모두, 서버에서 응답을 보내기 전에 클라이언트의 첫 번째 전송을 기다려야 합니다.

Flash Player에서 정책 파일을 요청할 때는 항상 연결이 설정된 직후에 다음 문자열을 전송합니다.

```
<policy-file-request/>
```

서버는 이 문자열을 수신한 후 정책 파일을 전송할 수 있습니다. Flash Player의 요청은 항상 null 바이트로 종료되며, 서버의 응답도 null 바이트로 종료되어야 합니다.

정책 파일 요청과 기본 연결 모두에 동일한 연결을 다시 사용하지 마십시오. 정책 파일을 전송한 후 해당 연결을 닫습니다. 연결을 닫지 않으면, Flash Player에서 기본 연결을 설정하기 위해 다시 연결하기 전에 정책 파일 연결을 닫습니다.

데이터 보호

Flash Player 9 이상, Adobe AIR 1.0 이상

인터넷을 통해 전송하는 동안 데이터가 도용되거나 변경되지 않도록 보호하기 위해 원본 데이터가 있는 서버에서 TLS(Transport Layer Security) 또는 SSL(Socket Layer Security)을 사용할 수 있습니다. 그런 다음 HTTPS 프로토콜을 사용하여 서버에 연결할 수 있습니다.

또한 AIR 2 이상용으로 작성된 응용 프로그램에서는 TCP 소켓 통신을 보호할 수 있습니다. SecureSocket 클래스를 통해 TLS 버전 1 또는 SSL 버전 4를 사용하는 소켓 서버에 대한 소켓 연결을 시작할 수 있습니다.

데이터 보내기

Flash Player 9 이상, Adobe AIR 1.0 이상

코드에서 데이터를 서버 또는 리소스에 보내면 데이터 보내기가 발생합니다. 네트워크 도메인의 내용에 대해서는 데이터 보내기가 항상 허용됩니다. 로컬 SWF 파일은 local-trusted, local-with-networking 또는 AIR 응용 프로그램 샌드박스에 있는 경우에만 네트워크 주소로 데이터를 보낼 수 있습니다. 자세한 내용은 950페이지의 "로컬 샌드박스"를 참조하십시오.

flash.net.sendToURL() 함수를 사용하여 URL로 데이터를 보낼 수 있습니다. 기타 메서드를 사용하여 URL로 요청을 보낼 수도 있습니다. 여기에는 Loader.load() 및 Sound.load()와 같은 로드 메서드, URLLoader.load() 및 URLStream.load()와 같은 데이터 로드 메서드가 포함됩니다.

파일 업로드 및 다운로드

Flash Player 9 이상, Adobe AIR 1.0 이상

FileReference.upload() 메서드는 원격 서버로 사용자가 선택한 파일의 업로드를 시작합니다. FileReference.browse() 또는 FileReferenceList.browse() 메서드는 FileReference.upload() 메서드보다 먼저 호출해야 합니다.

FileReference.browse() 또는 FileReferenceList.browse() 메서드를 초기화하는 코드는 마우스 이벤트나 키보드 이벤트에 대한 응답으로만 호출될 수 있습니다. 그 밖의 경우에 ActionScript가 호출되면 Flash Player 10 이상에서는 예외가 발생합니다. 그러나 사용자가 초기화한 이벤트는 AIR 응용 프로그램 샌드박스에서 이러한 메서드를 호출할 필요가 없습니다.

FileReference.download() 메서드를 호출하면 원격 서버에서 파일을 다운로드할 수 있는 대화 상자가 열립니다.

참고: 서버에 사용자 인증이 필요한 경우 브라우저에서 실행 중인, 즉 브라우저 플러그 인이나 ActiveX 컨트롤을 사용하는 SWF 파일만이 인증 및 다운로드에 필요한 사용자 이름과 암호를 요청하는 대화 상자를 제공할 수 있습니다. Flash Player는 사용자 인증을 요청하는 서버에 대한 업로드를 허용하지 않습니다.

호출하는 SWF 파일이 local-with-filesystem 샌드박스에 있는 경우에는 업로드 및 다운로드가 허용되지 않습니다.

기본적으로 SWF 파일은 자체 서버가 아닌 서버에서의 업로드나 다운로드를 시작하지 않습니다. 호출하는 SWF 파일의 도메인에 권한을 부여하는 정책 파일이 서버에서 제공되는 경우, SWF 파일은 다른 서버로부터의 업로드 및 다운로드를 허용합니다.

보안 도메인으로 가져온 SWF 파일에서 포함된 내용 로드

Flash Player 9 이상, Adobe AIR 1.0 이상

SWF 파일을 로드하는 경우 해당 파일을 로드하는 데 사용되는 Loader 객체의 load() 메서드에 대한 context 매개 변수를 설정할 수 있습니다. 이 매개 변수는 LoaderContext 객체를 사용합니다. LoaderContext 객체의 securityDomain 속성을 Security.currentDomain으로 설정하면 Flash Player에서 로드된 SWF 파일의 서버에서 URL 정책 파일을 확인합니다. 정책 파일이 있으면 로드하는 SWF 파일의 도메인에 대한 액세스가 허용되고 SWF 파일을 가져온 미디어로 로드할 수 있습니다. 이렇게 하여 로드하는 파일이 SWF 파일의 라이브러리에 있는 객체에 액세스할 수 있습니다.

SWF 파일에서 다른 보안 샌드박스에 로드된 SWF 파일의 클래스에 액세스하는 다른 방법은, 로드된 SWF 파일에서 Security.allowDomain() 메서드를 호출하여 호출하는 SWF 파일의 도메인에 대한 액세스를 허용하는 것입니다. 로드된 SWF 파일의 기본 클래스 생성자 메서드에 Security.allowDomain() 메서드에 대한 호출을 추가한 다음, 로드하는 SWF 파일에서 Loader 객체의 contentLoaderInfo 속성으로부터 전달되는 init 이벤트에 응답하는 이벤트 리스너를 추가하도록 할 수 있습니다. 이 이벤트가 전달되면 생성자 메서드에서 Security.allowDomain() 메서드를 호출했던 로드된 SWF 파일과 로드된 SWF 파일의 클래스를 로드하는 SWF 파일에서 사용할 수 있습니다. 로드하는 SWF 파일은 Loader.contentLoaderInfo.applicationDomain.getDefinition() 또는 Loader.contentLoaderInfo.applicationDomain.getQualifiedDefinitionNames()(Flash Player 11.3 이상, AIR 3.3 이상)를 호출하여, 로드되는 SWF 파일에서 클래스를 가져올 수 있습니다.

이전 내용으로 작업

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player 6에서 특정 Flash Player 설정에 사용되는 도메인은 SWF 파일 도메인의 뒷부분을 기준으로 합니다. 이러한 설정에는 카메라 및 마이크 권한, 저장소 할당량, 영구 공유 객체 저장소 설정이 포함됩니다.

예를 들어 SWF 파일의 도메인이 www.example.com과 같이 세 부분 이상으로 구성된 경우 도메인의 첫 번째 부분(www)이 제거되고 도메인의 나머지 부분이 사용됩니다. 따라서 Flash Player 6에서는 www.example.com과 store.example.com 모두에서 설정에 example.com 도메인을 사용합니다. 마찬가지로, www.example.co.uk와 store.example.co.uk 모두에서 설정에 example.co.uk 도메인을 사용합니다. 이렇게 하면 example1.co.uk 및 example2.co.uk와 같은 관련되지 않은 도메인의 SWF 파일에서 동일한 공유 객체에 액세스하는 문제가 발생할 수 있습니다.

Flash Player 7 이상 버전에서는 기본적으로 SWF 파일의 정확한 도메인에 따라 플레이어 설정이 선택됩니다. 예를 들어 www.example.com의 SWF 파일은 www.example.com의 플레이어 설정을 사용하고, store.example.com의 SWF 파일은 store.example.com에 별도로 설정된 플레이어 설정을 사용합니다.

ActionScript 3.0을 사용하여 작성된 SWF 파일에서 Security.exactSettings가 true(기본값)로 설정된 경우 Flash Player는 플레이어 설정에 정확히 일치하는 도메인을 사용합니다. false로 설정된 경우에는 Flash Player 6에서 사용되는 도메인 설정을 사용합니다. 설정 기본값에서 exactSettings를 변경하려면 Flash Player에서 카메라 또는 마이크를 사용하거나 영구 공유 객체를 가져오는 등 플레이어 설정을 선택하는 이벤트가 발생하기 전에 변경해야 합니다.

버전 6 SWF 파일을 제작하고 여기에서 영구 공유 객체를 작성한 경우, ActionScript 3.0을 사용하는 SWF에서 영구 공유 객체를 가져오려면 SharedObject.getLocal()을 호출하기 전에 Security.exactSettings를 false로 설정해야 합니다.

LocalConnection 권한 설정

Flash Player 9 이상, Adobe AIR 1.0 이상

LocalConnection 클래스를 사용하면 Flash Player 또는 AIR 응용 프로그램 간에 메시지를 보낼 수 있습니다. LocalConnection 객체는 동일한 클라이언트 컴퓨터에서 실행되는 Flash Player 또는 AIR 내용 사이에서만 통신할 수 있지만 서로 다른 응용 프로그램에서는 실행할 수 있습니다. 예를 들어 브라우저에서 실행되는 SWF, 프로젝터에서 실행되는 SWF 파일, AIR 응용 프로그램은 모두 LocalConnection 클래스를 사용하여 통신할 수 있습니다.

모든 LocalConnection 통신에는 전송자와 리스너가 있습니다. 기본적으로 Flash Player는 동일한 도메인에서 실행되는 코드 간에 LocalConnection 통신을 허용합니다. 서로 다른 샌드박스에서 실행되는 코드의 경우에는 리스너에서 LocalConnection.allowDomain() 메서드를 사용하여 전송자 권한을 허용해야 합니다. LocalConnection.allowDomain() 메서드에 인수로 전달하는 문자열에는 정확한 도메인 이름, IP 주소 및 * 와일드카드를 포함할 수 있습니다.

allowDomain() 메서드는 ActionScript 1.0 및 2.0에서의 형식과는 다르게 변경되었습니다. 이전 버전에서 allowDomain()은 구현되는 콜백 메서드였습니다. ActionScript 3.0에서 allowDomain()은 호출되는 LocalConnection 클래스의 내장 메서드입니다. 이러한 변경으로 인해 allowDomain()은 Security.allowDomain()과 거의 동일한 방식으로 작동됩니다.

SWF 파일은 LocalConnection 클래스의 domain 속성을 사용하여 도메인을 확인합니다.

아웃바운드 URL 액세스 제어

Flash Player 9 이상, Adobe AIR 1.0 이상

아웃바운드 스크립팅 및 URL 액세스(HTTP URL, mailto: 등 사용)는 다음 API를 사용하여 수행됩니다.

- flash.system.fscommand() 함수
- ExternalInterface.call() 메서드
- flash.net.navigateToURL() 함수

로컬 파일 시스템에서 로드된 내용의 경우 코드 및 포함 웹 페이지(있는 경우)가 local-trusted 또는 AIR 응용 프로그램 보안 샌드박스에 있는 이러한 메서드 호출이 성공합니다. 내용이 local-with-networking 또는 local-with-filesystem 샌드박스에 있는 경우에는 해당 메서드에 대한 호출이 실패합니다.

로컬에서 로드되지 않은 내용의 경우 아래에 설명된 AllowScriptAccess 매개 변수의 값에 따라 이러한 모든 API가 포함된 웹 페이지와 통신할 수 있습니다. flash.net.navigateToURL() 함수에는 SWF 파일이 포함된 페이지뿐 아니라 열려 있는 모든 브라우저 윈도우나 프레임과 통신할 수 있는 추가 기능이 있습니다. 이 기능에 대한 자세한 내용은 977페이지의 “navigateToURL() 함수 사용”을 참조하십시오.

SWF 파일을 로드하는 HTML 코드의 AllowScriptAccess 매개 변수는 SWF 파일 내에서 URL 액세스를 수행하는 기능을 제어합니다. PARAM 또는 EMBED 태그 내부에 이 매개 변수를 설정합니다. AllowScriptAccess에 대해 값이 설정되지 않은 경우 SWF 파일과 HTML 페이지가 동일한 도메인에 속해 있는 경우에만 서로 통신할 수 있습니다.

AllowScriptAccess 매개 변수는 가능한 세 가지 값, "always", "sameDomain" 또는 "never" 중 하나를 가질 수 있습니다.

- AllowScriptAccess가 "always"이면 SWF 파일은 HTML 페이지와 다른 도메인에 있는 경우에도 SWF 파일이 포함된 HTML 페이지와 통신할 수 있습니다.
- AllowScriptAccess가 "sameDomain"이면 SWF 파일은 HTML 페이지와 동일한 도메인에 속해 있는 경우에만 SWF 파일이 포함된 HTML 페이지와 통신할 수 있습니다. 이 값이 AllowScriptAccess의 기본값입니다. 한 도메인에 호스팅된 SWF 파일이 다른 도메인에 속해 있는 HTML 페이지의 스크립트에 액세스할 수 없도록 하려면 이 설정을 사용하거나 AllowScriptAccess 값을 설정하지 마십시오.

- AllowScriptAccess가 "never"이면 SWF 파일에서 어떠한 HTML 페이지와도 통신할 수 없습니다. Adobe Flash CS4 Professional 릴리스 이후부터 이 값은 사용되지 않습니다. 따라서 이 값은 더 이상 사용하지 않는 것이 좋으며 사용자의 도메인에서 신뢰할 수 없는 SWF 파일을 제공하지 않는 한 이 값은 필요하지 않습니다. 신뢰할 수 없는 SWF 파일을 불가피하게 제공해야 하는 경우 신뢰할 수 없는 모든 내용을 배치할 별도의 하위 도메인을 만드는 것이 좋습니다.

다음은 HTML 페이지에 AllowScriptAccess 태그를 설정하여 다른 도메인에 대한 아웃바운드 URL 액세스를 허용하는 예제입니다.

```
<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>
```

navigateToURL() 함수 사용

Flash Player 9 이상, Adobe AIR 1.0 이상

위에서 설명한 allowScriptAccess 매개 변수로 지정된 보안 설정 외에도 navigateToURL() 함수에는 선택적인 두 번째 매개 변수 target이 있습니다. target 매개 변수를 사용하여 URL 요청을 보낼 HTML 윈도우 또는 프레임의 이름을 지정할 수 있습니다. 이러한 요청에는 추가 보안 제한이 적용되며, 제한 사항은 navigateToURL()이 스크립팅 또는 비스크립팅 문으로 사용되는지에 따라 달라집니다.

navigateToURL("javascript: alert('Hello from Flash Player.')")과 같은 스크립팅 문의 경우 다음 규칙이 적용됩니다.

- SWF 파일이 로컬에서 신뢰할 수 있는 파일인 경우 요청이 성공합니다.
- SWF 파일이 포함된 HTML 페이지가 대상인 경우 위에서 설명한 allowScriptAccess 규칙이 적용됩니다.
- 대상이 SWF 파일과 동일한 도메인에서 로드된 내용을 포함하는 경우 요청이 성공합니다.
- 대상이 SWF 파일과 다른 도메인에서 로드된 내용을 포함하며 앞의 두 조건이 하나도 충족되지 않을 경우 요청이 실패합니다.

비스크립팅 문(예: HTTP, HTTPS 및 mailto:)의 경우 다음 조건이 모두 충족되면 요청이 실패합니다.

- 대상이 특수 키워드 "_top" 또는 "_parent" 중 하나인 경우
- SWF 파일이 다른 도메인에서 호스팅된 웹 페이지에 있는 경우
- allowScriptAccess의 값이 "always"가 아닌 상태로 SWF 파일이 포함된 경우

자세한 내용

Flash Player 9 이상, Adobe AIR 1.0 이상

아웃바운드 URL 액세스에 대한 자세한 내용은 **Adobe Flash Platform**용 **ActionScript 3.0 참조** 설명서의 다음 항목을 참조하십시오.

- flash.system.fscommand() 함수
- ExternalInterface 클래스의 call() 메서드
- flash.net.navigateToURL() 함수

공유 객체

Flash Player 9 이상, Adobe AIR 1.0 이상

Flash Player는 공유 객체를 사용하는 기능을 제공합니다. 공유 객체는 SWF 파일의 외부에 즉, 사용자의 파일 시스템에 로컬로 또는 RTMP 서버에 원격으로 존속하는 ActionScript 객체입니다. 공유 객체는 Flash Player의 다른 미디어와 같이 보안 샌드박스로 구분됩니다. 하지만 공유 객체는 도메인 경계를 넘어 액세스할 수 있는 리소스가 아니므로 공유 객체에 대한 샌드박스 모델은 약간 다릅니다. 대신 공유 객체는 항상 SharedObject 클래스의 메서드를 호출하는 각 SWF 파일의 도메인에 대한 특정 공유 객체 저장소에서 가져옵니다. 보통 공유 객체 저장소는 SWF 파일의 도메인보다 세부적으로 구분되어 있으며, 기본적으로 각 SWF 파일은 전체 원래 URL에 대한 특정 공유 객체 저장소를 사용합니다. 공유 객체에 대한 자세한 내용은 639페이지의 "공유 객체"를 참조하십시오.

SWF 파일은 SharedObject.getLocal() 및 SharedObject.getRemote() 메서드의 localPath 매개 변수를 사용하여 해당 URL의 일부와 연관된 공유 객체 저장소를 사용할 수 있습니다. 이렇게 하여 SWF 파일은 다른 URL의 다른 SWF 파일과 공유 객체 저장소를 공유할 수 있습니다. localPath 매개 변수로 '/' 를 전달하는 경우에도 해당 도메인에 대한 특정 공유 객체 저장소가 지정됩니다.

사용자는 [Flash Player 설정] 대화 상자 또는 설정 관리자를 사용하여 공유 객체 액세스를 제한할 수 있습니다. 기본적으로 도메인당 최대 100KB의 공유 객체 데이터를 만들 수 있습니다. 관리자와 사용자는 또한 파일 시스템에 대한 쓰기 권한을 제한할 수도 있습니다. 자세한 내용은 954페이지의 "관리자 컨트롤" 및 955페이지의 "사용자 컨트롤"을 참조하십시오.

SharedObject.getLocal() 메서드 또는 SharedObject.getRemote() 메서드의 secure 매개 변수에 true를 지정하여 보안 공유 객체를 지정할 수 있습니다. secure 매개 변수의 다음 사항에 주의하십시오.

- 이 매개 변수를 true로 설정하면 Flash Player가 보안 공유 객체를 새로 만들거나 기존 보안 공유 객체에 대한 참조를 가져옵니다. 이 보안 공유 객체는 secure 매개 변수가 true로 설정된 SharedObject.getLocal()을 호출하는 HTTPS를 통해 제공되는 SWF 파일에서만 읽거나 쓸 수 있습니다.
- 이 매개 변수가 false로 설정되면 Flash Player는 비 HTTPS 연결을 통해 제공되는 SWF 파일에서 읽거나 쓸 수 있는 새 공유 객체를 만들거나 기존 공유 객체에 대한 참조를 가져옵니다.

호출하는 SWF 파일이 HTTPS URL에 없는 경우, SharedObject.getLocal() 메서드 또는 SharedObject.getRemote() 메서드의 secure 매개 변수를 true로 지정하면 SecurityError 예외가 발생합니다.

공유 객체 저장소의 선택은 SWF 파일의 원래 URL을 기반으로 합니다. 이것은 SWF 파일이 단순 URL에서 시작되지 않은 경우 즉, 가져오기 로드 및 동적 로드의 두 상황에도 적용됩니다. 가져오기 로드는 LoaderContext.securityDomain 속성이 SecurityDomain.currentDomain으로 설정된 SWF 파일을 로드하는 경우입니다. 이 경우 로드된 SWF 파일은 로드하는 SWF 파일의 도메인에서 시작하여 실제 원래 URL을 지정하는 의사 URL을 가집니다. 동적 로드는 Loader.loadBytes() 메서드를 사용하여 SWF 파일을 로드하는 것을 말합니다. 이 경우 로드된 SWF 파일은 로드하는 SWF 파일의 전체 URL로 시작하고 뒤에 정수 ID가 있는 의사 URL을 가집니다. 가져오기 로드 및 동적 로드의 경우에서 모두, LoaderInfo.url 속성을 사용하여 SWF 파일의 의사 URL을 검사할 수 있습니다. 의사 URL은 공유 객체 저장소 선택 시 실제 URL과 동일하게 취급됩니다. 의사 URL의 일부 또는 전체를 사용하는 공유 객체 localPath 매개 변수를 지정할 수 있습니다.

사용자 및 관리자는 **타사 공유 객체**를 사용하지 못하도록 지정할 수 있습니다. SWF 파일의 원래 URL이 브라우저의 주소 표시줄에 표시된 URL과 다른 도메인인 경우에는 웹 브라우저에서 실행 중인 모든 SWF 파일에서 공유 객체를 사용합니다. 사용자 및 관리자는 크로스 도메인 추적을 방지하고 개인 정보를 보호하기 위해 타사 공유 객체를 사용하지 못하도록 지정할 수 있습니다. 이 제한을 피하려면 SWF 파일이 브라우저의 주소 표시줄에 표시된 도메인과 동일한 도메인에서 제공되도록 하는 HTML 페이지 구조 내에서만 공유 객체를 사용하는 SWF 파일이 로드되도록 해야 합니다. 타사 SWF 파일의 공유 객체를 사용하려고 시도하면 타사 공유 객체 사용이 비활성화되고 SharedObject.getLocal() 및 SharedObject.getRemote() 메서드에서 null을 반환합니다. 자세한 내용은 www.adobe.com/products/flashplayer/articles/thirdpartylso을 참조하십시오.

카메라, 마이크, 클립보드, 마우스 및 키보드 액세스

Flash Player 9 이상, Adobe AIR 1.0 이상

SWF 파일에서 Camera.get() 또는 Microphone.get() 메서드를 사용하여 사용자의 카메라나 마이크에 액세스하려고 시도하면 Flash Player에서 사용자가 카메라와 마이크에 대한 액세스를 허용하거나 거부할 수 있는 [개인 정보] 대화 상자를 표시합니다. 또한 사용자와 관리자는 mms.cfg 파일의 컨트롤, 설정 UI 및 설정 관리자를 통해 사이트별로 또는 전체 사이트에 대해 카메라 액세스를 비활성화할 수 있습니다(954페이지의 "관리자 컨트롤" 및 955페이지의 "사용자 컨트롤" 참조). 사용자 제한이 있는 Camera.get() 및 Microphone.get() 메서드 각각은 null 값을 반환합니다. Capabilities.avHardwareDisable 속성을 사용하여 카메라 및 마이크가 관리상의 이유로 금지되었는지(true) 또는 허용되었는지(false) 여부를 확인할 수 있습니다.

System.setClipboard() 메서드를 사용하면 SWF 파일에서 클립보드의 내용을 일반 텍스트 문자열로 바꿀 수 있습니다. 이로 인한 보안상의 위험은 없습니다. 암호 및 기타 중요한 데이터를 클립보드로 잘라내거나 복사하는 경우 이에 따른 위험을 방지하기 위해, 클립보드의 내용을 읽을 수 있는 getClipboard() 메서드는 제공되지 않습니다.

Flash Player에서 실행되는 응용 프로그램은 응용 프로그램 포커스 안에서 발생하는 키보드 및 마우스 이벤트만 모니터링할 수 있으며, Flash Player에서 실행되는 내용은 다른 응용 프로그램의 키보드 또는 마우스 이벤트를 감지할 수 없습니다.

AIR 보안

Adobe AIR 1.0 이상

AIR 보안 기본

Adobe AIR 1.0 이상

AIR 응용 프로그램은 기본 응용 프로그램과 동일한 보안 제한 사항을 갖고 실행됩니다. 일반적으로 AIR 응용 프로그램은 기본 응용 프로그램과 같이 파일 읽기 및 쓰기, 응용 프로그램 시작, 화면에 그리기, 네트워크와의 통신과 같은 운영 체제 기능에 대한 광범위한 액세스가 가능합니다. 사용자별 권한과 같이 기본 응용 프로그램에 적용되는 운영 체제 제한은 AIR 응용 프로그램에 동일하게 적용됩니다.

Adobe® AIR® 보안 모델은 Adobe® Flash® Player 보안 모델이 발전한 것이지만 보안 계약은 브라우저의 내용에 적용된 보안 계약과 다릅니다. 이 계약은 개발자에게 브라우저 기반 응용 프로그램에 적합하지 않은 자유롭고 다양한 경험에 대한 보다 광범위한 기능의 보안 수단을 제공합니다.

런타임에서 메모리 관리를 제공하도록 AIR 응용 프로그램은 컴파일된 바이트코드(SWF 내용) 또는 해석된 스크립트(JavaScript, HTML)를 사용하여 작성됩니다. 이로 인해 AIR 응용 프로그램이 버퍼 오버플로 및 메모리 손상과 같은 메모리 관리 관련 취약성에 의한 영향을 덜 받게 됩니다. 이는 기본 코드로 작성된 데스크톱 응용 프로그램에 영향을 주는 가장 일반적인 취약성 중 일부입니다.

설치 및 업데이트

Adobe AIR 1.0 이상

AIR 응용 프로그램은 air 확장자를 갖는 AIR 설치 프로그램 파일이나 기본 플랫폼의 파일 형식 및 확장자를 갖는 기본 설치 프로그램을 통해 배포됩니다. 예를 들어 Windows의 기본 설치 프로그램 형식은 EXE 파일이고 Android의 기본 형식은 APK 파일입니다.

Adobe AIR가 설치되고 AIR 설치 프로그램 파일이 열리면 AIR 런타임에서 설치 프로세스를 관리합니다. 기본 설치 프로그램이 사용되는 경우에는 운영 체제에서 설치 프로세스를 관리합니다.

참고: 개발자는 AIR 파일 형식을 사용할 경우 버전, 응용 프로그램 이름 및 제작자 소스를 지정할 수 있지만 초기 응용 프로그램 설치 워크플로 자체는 수정할 수 없습니다. 이러한 제한 사항은 모든 AIR 응용 프로그램이 런타임에 의해 관리되는 안전하고 효율적이며 일관된 설치 절차를 공유하기 때문에 사용자에게 유용합니다. 응용 프로그램 사용자 정의가 필요하면 응용 프로그램이 처음에 실행될 때 제공될 수 있습니다.

런타임 설치 위치

Adobe AIR 1.0 이상

SWF 파일을 사용하기 위해 먼저 Flash Player 브라우저 플러그인을 설치해야 하는 것처럼 AIR 파일 형식을 사용하는 AIR 응용 프로그램을 사용하려면 먼저 사용자 컴퓨터에 런타임을 설치해야 합니다.

런타임은 데스크톱 컴퓨터의 다음 위치에 설치됩니다.

- Mac OS: /Library/Frameworks/
- Windows: C:\Program Files\Common Files\Adobe AIR
- Linux: /opt/Adobe AIR/

Mac OS에서는 업데이트된 버전의 응용 프로그램을 설치하려면 사용자에게 응용 프로그램 디렉토리에 설치할 수 있는 적절한 시스템 권한이 필요합니다. Windows 및 Linux에서는 사용자에게 관리 권한이 있어야 합니다.

참고: iOS에서 AIR 런타임은 개별적으로 설치되지 않으며 모든 AIR 응용 프로그램은 자체 포함 응용 프로그램입니다.

런타임은 웹 브라우저에서 바로 설치하는 연속 설치 기능을 사용하거나 수동 설치를 통해 설치할 수 있습니다. 또한 기본 설치 프로그램으로 패키지화된 AIR 응용 프로그램은 일반 응용 프로그램 설치 프로세스의 일부로 AIR 런타임을 설치할 수 있습니다. 이 방법으로 AIR 런타임을 배포하려면 Adobe와 재배포 계약을 맺어야 합니다.

연속 설치(런타임 및 응용 프로그램)

Adobe AIR 1.0 이상

연속 설치 기능은 개발자에게 Adobe AIR이 아직 설치되지 않은 사용자에 대한 효율적인 설치 경험을 제공합니다. 연속 설치 방법에서 개발자는 설치를 위한 응용 프로그램을 제공하는 SWF 파일을 만듭니다. 사용자가 SWF 파일에서 클릭하여 응용 프로그램을 설치하면 SWF 파일에서 런타임을 감지하려고 합니다. 런타임 설치를 감지할 수 없는 경우 런타임은 개발자 응용 프로그램에 대한 설치 프로세스로 즉시 활성화됩니다.

수동 설치

Adobe AIR 1.0 이상

다른 방법으로 사용자가 AIR 파일을 열기 전에 수동으로 런타임을 다운로드하고 설치할 수 있습니다. 그러면 개발자가 전자 메일 또는 웹 사이트의 HTML 링크 등과 같이 서로 다른 방법으로 AIR 파일을 배포할 수 있습니다. AIR 파일이 열리면 런타임에서 응용 프로그램 설치 처리를 시작합니다.

응용 프로그램 설치 흐름

Adobe AIR 1.0 이상

AIR 보안 모델을 통해 사용자는 AIR 응용 프로그램 설치 여부를 결정할 수 있습니다. AIR 설치 경험에서는 기본 응용 프로그램 설치 기술의 여러 사항을 개선했으므로 사용자에 대해 이러한 신뢰하는 결정을 보다 쉽게 내릴 수 있습니다.

- 런타임은 AIR 응용 프로그램이 웹 브라우저의 링크에서 설치되어도 모든 운영 체제에 일관된 설치 경험을 제공합니다. 대부분의 기본 응용 프로그램 설치 경험은 브라우저 또는 보안 정보를 제공(제공되는 경우)하는 다른 응용 프로그램에 따라 다릅니다.

- AIR 응용 프로그램 설치 경험은 응용 프로그램 소스 및 응용 프로그램에 사용할 수 있는 권한(사용자가 설치 진행을 허용하는 경우)에 대한 정보를 식별합니다.
- 런타임은 AIR 응용 프로그램의 설치 프로세스를 관리합니다. AIR 응용 프로그램은 런타임에서 사용하는 설치 프로세스를 조작할 수 없습니다.

일반적으로 사용자는 신뢰하지 않거나 확인할 수 없는 소스에서 가져온 데스크톱 응용 프로그램을 설치할 수 없습니다. 다른 설치 가능 응용 프로그램의 경우와 마찬가지로 AIR 응용 프로그램에도 기본 응용 프로그램에 대한 보안을 입증해야 하는 부담이 동일하게 존재합니다.

응용 프로그램 대상

Adobe AIR 1.0 이상

다음과 같은 두 옵션 중 하나를 사용하여 설치 디렉토리를 설정할 수 있습니다.

1. 설치하는 동안 사용자가 대상을 사용자 정의합니다. 사용자가 지정하는 위치로 응용 프로그램이 설치됩니다.
2. 사용자가 설치 대상을 변경하지 않으면 응용 프로그램은 런타임에 의해 결정된 다음과 같은 기본 경로에 설치됩니다.
   - Mac OS: ~/Applications/
   - Windows XP 이전: C:\Program Files\
   - Windows Vista: ~/Apps/
   - Linux: /opt/

   개발자가 응용 프로그램 설명자 파일에서 installFolder 설정을 지정하면 응용 프로그램은 이 디렉토리의 하위 경로에 설치됩니다.

AIR 파일 시스템

Adobe AIR 1.0 이상

AIR 응용 프로그램에 대한 설치 프로세스는 개발자가 AIR 설치 프로그램 파일 내에 포함한 모든 파일을 사용자의 로컬 컴퓨터에 복사합니다. 설치된 응용 프로그램은 다음으로 구성되어 있습니다.

- Windows: AIR 설치 프로그램 파일에 들어 있는 모든 파일을 포함하는 디렉토리. 런타임은 AIR 응용 프로그램을 설치하는 동안 exe 파일도 만듭니다.
- Linux: AIR 설치 프로그램 파일에 들어 있는 모든 파일을 포함하는 디렉토리. 런타임은 AIR 응용 프로그램을 설치하는 동안 bin 파일도 만듭니다.
- Mac OS: AIR 설치 프로그램 파일의 모든 내용을 포함하는 app 파일. 이 파일은 Finder의 "패키지 내용 보기" 옵션을 사용하여 관리할 수 있습니다. 런타임은 이 app 파일을 AIR 응용 프로그램 설치의 일부로 만듭니다.

AIR 응용 프로그램은 다음 방법을 사용하여 실행됩니다.

- Windows: 설치 폴더 또는 이 파일에 해당하는 바로 가기(예: [시작] 메뉴 또는 바탕 화면에 있는 바로 가기)에서 .exe 파일을 실행합니다.
- Linux: 설치 폴더에서 .bin 파일을 실행하거나 [응용 프로그램] 메뉴에서 AIR 응용 프로그램을 선택하거나 AIR 응용 프로그램의 별칭 또는 바탕 화면 바로 가기를 실행합니다.
- Mac OS: .app 파일 또는 이 파일을 가리키는 별칭을 실행합니다.

응용 프로그램 파일 시스템에는 응용 프로그램 기능과 관련된 하위 디렉토리도 포함되어 있습니다. 예를 들어 암호화된 로컬 저장소에 쓴 정보가 응용 프로그램의 응용 프로그램 식별자의 이름을 딴 하위 디렉토리에 저장됩니다.

AIR 응용 프로그램 저장소

Adobe AIR 1.0 이상

AIR 응용 프로그램에 사용자 하드 드라이브의 모든 위치에 쓸 수 있는 권한이 있지만 개발자는 해당 응용 프로그램에 대한 로컬 저장소의 app-storage:/ 경로를 사용해야 합니다. 응용 프로그램에서 app-storage:/에 쓴 파일은 다음과 같이 사용자 운영 체제에 따라 표준 위치에 배치됩니다.

- Mac·OS의 경우: 응용 프로그램의 저장소 디렉토리는 다음과 같이 AIR 버전에 따라 다릅니다.
  - **AIR 3.2 이하** - <appData>/<appId>/Local Store/. 여기서 <appData>는 사용자의 "환경 설정 폴더"이며, 일반적으로 /Users/<user>/Library/Preferences
  - **AIR 3.3 이상** - <path>/Library/Application Support/<appID>/Local Store. 여기서 <path>는 /Users/<user>/Library/Containers/<bundle-id>/Data(샌드박스 환경) 또는 /Users/<user>(샌드박스 환경 외부에서 실행되는 경우)
- Windows의 경우: 응용 프로그램의 저장소 디렉토리는 <appData>\<appId>\Local Store\이며 여기서 <appData>는 사용자의 CSIDL_APPDATA "특수 폴더"(일반적으로 C:\Documents and Settings\<user>\Application Data)입니다.
- Linux의 경우: <appData>/<appID>/Local Store/이며 여기서 <appData>는 /home/<user>/.appdata입니다.

air.File.applicationStorageDirectory 속성을 통해 응용 프로그램 저장소 디렉토리에 액세스할 수 있습니다. File 클래스의 resolvePath() 메서드를 사용하여 이러한 디렉토리 내용에 액세스할 수 있습니다. 자세한 내용은 595페이지의 "파일 시스템 작업"을 참조하십시오.

Adobe AIR 업데이트

Adobe AIR 1.0 이상

사용자가 업데이트된 버전의 런타임이 필요한 AIR 응용 프로그램을 설치하면 런타임은 필수 런타임 업데이트를 자동으로 설치합니다.

런타임을 업데이트하려면 사용자에게 컴퓨터에 대한 관리 권한이 필요합니다.

AIR 응용 프로그램 업데이트

Adobe AIR 1.0 이상

소프트웨어 업데이트의 개발 및 배포는 기본 코드 응용 프로그램이 직면하는 가장 큰 보안 문제 중 하나입니다. AIR API는 이를 개선하는 메커니즘을 제공합니다. AIR 파일에 대한 원격 위치를 확인하기 위해 시작 시 Updater.update() 메서드를 호출할 수 있습니다. 업데이트가 적합하면 AIR 파일이 다운로드되고 설치되며 응용 프로그램이 다시 시작됩니다. 개발자는 이 클래스를 사용하여 새 기능을 제공할 뿐만 아니라 잠재적 보안 취약성에 대응할 수도 있습니다.

Updater 클래스는 AIR 파일로 배포된 응용 프로그램을 업데이트하는 데만 사용할 수 있습니다. 기본 응용 프로그램으로 배포된 응용 프로그램에서는 기본 운영 체제의 업데이트 기능이 있는 경우 해당 기능을 활용해야 합니다.

참고: 개발자는 응용 프로그램 설명자 파일의 versionNumber 속성을 설정하여 응용 프로그램 버전을 지정할 수 있습니다.

AIR 응용 프로그램 제거

Adobe AIR 1.0 이상

AIR 응용 프로그램을 제거하면 응용 프로그램 디렉토리의 모든 파일이 제거됩니다. 그러나 응용 프로그램 디렉토리 외부에 작성했을 수도 있는 모든 파일은 제거하지 않습니다. AIR 응용 프로그램을 제거하는 경우 AIR 응용 프로그램이 응용 프로그램 디렉토리 외부에 있는 파일에 대해 변경한 내용은 되돌리지 않습니다.

관리자를 위한 Windows 레지스트리 설정

Adobe AIR 1.0 이상

Windows의 경우 관리자는 AIR 응용 프로그램 설치 및 런타임 업데이트를 방지하거나 허용하도록 컴퓨터를 구성할 수 있습니다. 이러한 설정은 Windows 레지스트리의 HKLM\Software\Policies\Adobe\AIR 키 아래에 포함되어 있습니다. 이 설정에는 다음이 포함됩니다.

레지스트리 설정	설명
AppInstallDisabled	AIR 응용 프로그램 설치 및 제거를 허용하도록 지정합니다. "허용하는" 경우 0으로 설정하고, "허용하지 않는" 경우 1로 설정합니다.
UntrustedAppInstallDisabled	신뢰할 수 없는 AIR 응용 프로그램(신뢰할 수 있는 인증서를 포함하지 않는 응용 프로그램)을 설치할 수 있도록 지정합니다. "허용하는" 경우 0으로 설정하고, "허용하지 않는" 경우 1로 설정합니다.
UpdateDisabled	런타임 업데이트를 배경 작업 또는 명시적 설치의 일부로 허용하도록 지정합니다. "허용하는" 경우 0으로 설정하고, "허용하지 않는" 경우 1로 설정합니다.

Adobe AIR의 HTML 보안

Adobe AIR 1.0 이상

이 항목에서는 AIR HTML 보안 아키텍처에 대해 설명하고 iframe, 프레임 및 샌드박스 브리지를 활용하여 HTML 기반 응용 프로그램을 설치하고 HTML 내용을 SWF 기반 응용 프로그램으로 안전하게 통합하는 방법에 대해 설명합니다.

런타임은 HTML 및 JavaScript에서 규칙을 적용하고 가능한 보안 취약성을 극복할 수 있는 메커니즘을 제공합니다. 응용 프로그램이 주로 JavaScript로 작성되었는지 또는 HTML 및 JavaScript 내용을 SWF 기반 응용 프로그램으로 로드할지 여부에 상관없이 동일한 규칙이 적용됩니다. 응용 프로그램 샌드박스 내용과 비 응용 프로그램 보안 샌드박스 내용의 권한은 서로 다릅니다. 내용을 iframe 또는 프레임으로 로드할 때 런타임은 프레임 또는 iframe의 내용을 응용 프로그램 보안 샌드박스의 내용과 안전하게 통신할 수 있는 안전한 **샌드박스 브리지** 메커니즘을 제공합니다.

AIR SDK에서는 HTML 내용을 렌더링하기 위한 세 개의 클래스를 제공합니다.

HTMLLoader 클래스는 JavaScript 코드 및 AIR API 간의 긴밀한 통합을 지원합니다.

StageWebView 클래스는 HTML 렌더링 클래스이며 호스트 AIR 응용 프로그램과 매우 제한된 상태로 통합되어 있습니다. StageWebView 클래스에서 로드한 내용은 응용 프로그램 보안 샌드박스에 배치되지 않으며 호스트 AIR 응용 프로그램에 있는 데이터에 액세스하거나 함수를 호출할 수 없습니다. 데스크톱 플랫폼에서 StageWebView 클래스는 Webkit에 기반한 내장 AIR HTML 엔진을 사용하며, 이 엔진은 HTMLLoader 클래스에서도 사용됩니다. 모바일 플랫폼에서 StageWebView 클래스는 운영 체제에서 제공하는 HTML 컨트롤을 사용합니다. 따라서 모바일 플랫폼에서 StageWebView 클래스는 시스템 웹 브라우저와 동일한 보안 고려 사항 및 취약성을 갖고 있습니다.

TextField 클래스는 HTML 텍스트의 문자열을 표시할 수 있으며, JavaScript는 실행할 수 없지만 텍스트에 링크 및 외부에서 로드된 이미지를 포함할 수 있습니다.

자세한 내용은 894페이지의 “보안 관련 JavaScript 오류 방지”를 참조하십시오.

HTML 기반 응용 프로그램 구성에 대한 개요

Adobe AIR 1.0 이상

프레임 및 iframe은 AIR에서 HTML 내용을 구성하는 편리한 구조를 제공합니다. 프레임은 데이터 지속성 유지와 원격 내용의 안전한 작업을 위한 수단을 제공합니다.

AIR의 HTML이 해당 페이지 기반의 일반 조직을 보유하고 있으므로 HTML 환경은 HTML 내용의 맨 위 프레임이 다른 페이지로 "이동하는" 경우 완전히 새로 고쳐집니다. 브라우저에서 실행 중인 웹 응용 프로그램에 대해 수행하는 작업과 거의 마찬가지로 프레임 및 iframe을 사용하여 AIR에서 데이터 지속성을 유지할 수 있습니다. 맨 위 프레임에서 주 응용 프로그램 객체를 정의합니다. 이러한 객체는 프레임이 새 페이지로 이동하도록 허용하지 않는 한 계속 유지됩니다. 자식 프레임 또는 iframe을 사용하여 응용 프로그램의 일시적인 부분을 로드 및 표시할 수 있습니다. 프레임 이외에 또는 프레임 대신 사용할 수 있는 데이터 지속성을 유지하는 여러 방법이 있습니다. 이러한 방법에는 쿠키, 로컬 공유 객체, 로컬 파일 저장소, 암호화된 파일 저장소, 로컬 데이터베이스 저장소가 있습니다.

AIR의 HTML은 실행 코드와 데이터 간의 일반적인 모호한 경계를 유지하므로 AIR은 HTML 환경의 맨 위 프레임에 있는 내용을 응용 프로그램 샌드박스에 배치합니다. 페이지 load 이벤트 후 AIR은 텍스트 문자열을 실행 객체로 변환할 수 있는 eval()과 같은 모든 작업을 제한합니다. 응용 프로그램에서 원격 내용을 로드하지 않는 경우에도 이러한 제한이 적용됩니다. HTML 내용에서 이러한 제한된 작업을 실행할 수 있도록 하려면 프레임 또는 iframe을 사용하여 비 응용 프로그램 샌드박스에 내용을 배치해야 합니다. eval() 함수에 의존하는 일부 JavaScript 응용 프로그램 프레임워크를 사용할 경우 샌드박스된 자식 프레임에서 내용을 실행해야 할 수 있습니다. 응용 프로그램 샌드박스의 JavaScript에 대한 전체 제한 사항 목록은 985페이지의 "서로 다른 샌드박스의 내용에 대한 코드 제한 사항"을 참조하십시오.

AIR의 HTML이 원격의 안전하지 않을 수 있는 내용을 로드할 수 있으므로 AIR은 한 도메인의 내용이 다른 도메인의 내용과 상호 작용하지 않도록 하는 동일 원본 정책을 적용합니다. 응용 프로그램 도메인과 기타 도메인의 내용 간 상호 작용을 허용하려면 부모 프레임과 자식 프레임 간의 인터페이스로 사용할 브리지를 설정합니다.

부모와 자식 간 샌드박스 관계 설정

Adobe AIR 1.0 이상

AIR은 sandboxRoot 및 documentRoot 특성을 HTML 프레임 및 iframe 요소에 추가합니다. 이러한 특성을 통해 응용 프로그램 내용을 다른 도메인에서 가져온 것처럼 다룰 수 있습니다.

특성	설명
sandboxRoot	프레임 내용을 배치할 샌드박스 및 도메인을 결정하는 데 사용할 URL입니다. file:, http: 또는 https: URL 스킴을 사용해야 합니다.
documentRoot	프레임 내용을 로드할 URL입니다. file:, app: 또는 app-storage: URL 스킴을 사용해야 합니다.

다음은 원격 샌드박스 및 www.example.com 도메인에서 실행할 응용 프로그램의 sandbox 하위 디렉토리에 설치된 내용을 매핑하는 예제입니다.

```
<iframe
    src="ui.html"
    sandboxRoot="http://www.example.com/local/"
    documentRoot="app:/sandbox/">
</iframe>
```

다른 샌드박스 또는 도메인의 부모 프레임과 자식 프레임 간 브리지 설정

Adobe AIR 1.0 이상

AIR은 childSandboxBridge 및 parentSandboxBridge 속성을 자식 프레임의 window 객체에 추가합니다. 이러한 속성을 통해 부모 프레임과 자식 프레임 간의 인터페이스로 사용할 브리지를 정의할 수 있습니다. 각 브리지는 한 방향으로 이동합니다.

childSandboxBridge - childSandboxBridge 속성을 통해 자식 프레임이 인터페이스를 부모 프레임의 내용에 표시할 수 있습니다. 인터페이스를 표시하려면 childSandbox 속성을 자식 프레임의 함수 또는 객체로 설정합니다. 그렇게 하면 부모 프레임의 내용에서 객체 또는 함수에 액세스할 수 있습니다. 다음 예제에서는 자식 프레임에서 실행하는 스크립트에서 함수 및 속성을 포함하는 객체를 해당 부모에 표시하는 방법을 보여 줍니다.

```
var interface = {};
interface.calculatePrice = function(){
    return .45 + 1.20;
}
interface.storeID = "abc"
window.childSandboxBridge = interface;
```

이 자식 내용이 "child"의 id가 할당된 iframe에 있는 경우 프레임의 childSandboxBridge 속성을 읽어 부모 내용에서 인터페이스에 액세스할 수 있습니다.

```
var childInterface = document.getElementById("child").childSandboxBridge;
air.trace(childInterface.calculatePrice()); //traces "1.65"
air.trace(childInterface.storeID)); //traces "abc"
```

parentSandboxBridge - parentSandboxBridge 속성을 통해 부모 프레임은 인터페이스를 자식 프레임의 내용에 표시할 수 있습니다. 인터페이스를 표시하려면 자식 프레임의 parentSandbox 속성을 부모 프레임의 함수 또는 객체로 설정합니다. 그렇게 하면 자식 프레임의 내용에서 객체 또는 함수에 액세스할 수 있습니다. 다음 예제에서는 부모 프레임에서 실행하는 스크립트에서 save 함수를 포함하는 객체를 해당 자식에 표시하는 방법을 보여 줍니다.

```
var interface = {};
interface.save = function(text){
    var saveFile = air.File("app-storage:/save.txt");
    //write text to file
}
document.getElementById("child").parentSandboxBridge = interface;
```

이 인터페이스를 사용하면 자식 프레임의 내용이 save.txt라는 파일에 텍스트를 저장할 수 있습니다. 그러나 파일 시스템에 대한 어떠한 액세스 권한도 없습니다. 일반적으로 응용 프로그램 내용은 가능한 가장 좁은 인터페이스를 다른 샌드박스에 표시해야 합니다. 자식 내용은 다음과 같이 save 함수를 호출할 수 있습니다.

```
var textToSave = "A string.";
window.parentSandboxBridge.save(textToSave);
```

자식 내용이 parentSandboxBridge 객체의 속성을 설정하려는 경우 런타임에서 SecurityError 예외가 발생합니다. 부모 내용이 childSandboxBridge 객체의 속성을 설정하려는 경우 런타임에서 SecurityError 예외가 발생합니다.

서로 다른 샌드박스의 내용에 대한 코드 제한 사항

Adobe AIR 1.0 이상

이 항목의 소개 부분인 983페이지의 “Adobe AIR의 HTML 보안”에서 설명한 대로 런타임은 규칙을 적용하고 HTML 및 JavaScript에서의 가능한 보안 취약성을 극복할 수 있는 메커니즘을 제공합니다. 이 항목에서는 그러한 제한 사항을 보여 줍니다. 코드가 이러한 제한된 API를 호출하려고 하면 런타임에서 "응용 프로그램 보안 샌드박스에서의 JavaScript 코드에 대한 Adobe AIR 런타임 보안 위반” 메시지의 오류가 발생합니다.

자세한 내용은 894페이지의 “보안 관련 JavaScript 오류 방지”를 참조하십시오.

JavaScript eval() 함수 및 유사 기술 사용에 대한 제한 사항

Adobe AIR 1.0 이상

응용 프로그램 보안 샌드박스의 HTML 내용의 경우 코드를 로드한 후(body 요소의 onload 이벤트를 전달하고 onload 핸들러 함수가 실행을 완료한 후) 문자열을 실행 코드로 동적으로 변환할 수 있는 API를 사용할 때 제한 사항이 있습니다. 이는 응용 프로그램이 잠재적으로 안전하지 않은 네트워크 도메인과 같은 비 응용 프로그램 소스의 코드를 실수로 삽입 및 실행하지 않도록 하는 것입니다.

예를 들어 응용 프로그램에서 DOM 요소의 innerHTML 속성에 쓸 원격 소스의 문자열 데이터를 사용하는 경우 문자열에 안전하지 않은 작업을 수행할 수 있는 실행(JavaScript) 코드가 포함될 수 있습니다. 그러나 내용을 로드하는 동안 원격 문자열을 DOM에 삽입할 위험은 없습니다.

하나의 제한 사항은 JavaScript eval() 함수를 사용하는 경우에 해당됩니다. 응용 프로그램 샌드박스의 코드가 로드되고 onload 이벤트 핸들러가 처리된 후에는 eval() 함수를 제한적인 방법으로만 사용할 수 있습니다. 다음 규칙은 응용 프로그램 보안 샌드박스에서 코드가 로드된 **이후**eval() 함수를 사용할 때 적용됩니다.

- 리터럴을 포함하는 식이 허용됩니다. 예를 들면 다음과 같습니다.

```
eval("null");
eval("3 + .14");
eval("'foo'");
```

- 다음과 같이 객체 리터럴이 허용됩니다.

```
{ prop1: val1, prop2: val2 }
```

- 다음과 같이 객체 리터럴 setter/getter가 **금지됩니다**.

```
{ get prop1() { ... }, set prop1(v) { ... } }
```

- 다음과 같이 배열 리터럴이 허용됩니다.

```
[ val1, val2, val3 ]
```

- 다음과 같이 속성 읽기를 포함하는 식이 **금지됩니다**.

```
a.b.c
```

- 함수 호출이 **금지됩니다**.
- 함수 정의가 **금지됩니다**.
- 속성 설정이 **금지됩니다**.
- 함수 리터럴이 **금지됩니다**.

그러나 코드를 로드하는 동안, onload 이벤트 이전 및 onload 이벤트 핸들러 함수를 실행하는 동안 이러한 제한 사항이 응용 프로그램 보안 샌드박스의 내용에 적용되지 않습니다.

예를 들어 코드를 로드한 후 다음 코드를 실행하면 런타임에서 예외가 발생합니다.

```
eval("alert(44)");
eval("myFunction(44)");
eval("NativeApplication.applicationID");
```

eval() 함수를 호출할 때 생성된 코드와 같이 동적으로 생성된 코드는 응용 프로그램 샌드박스 내에 허용되는 경우 보안 위험을 유발할 수 있습니다. 예를 들어 응용 프로그램에서 네트워크 도메인에서 로드한 문자열을 실수로 실행하고 해당 문자열이 악의적인 코드를 포함할 수 있습니다. 예를 들어 이 코드가 사용자 컴퓨터의 파일을 삭제하거나 변경하는 코드일 수 있습니다. 또는 로컬 파일의 내용을 신뢰할 수 없는 네트워크 도메인으로 다시 보고하는 코드일 수 있습니다.

동적 코드를 생성하는 방법은 다음과 같습니다.

- eval() 함수를 호출합니다.
- innerHTML 속성 또는 DOM 함수를 사용하여 응용 프로그램 디렉토리 외부에 있는 스크립트를 로드하는 script 태그를 삽입합니다.
- innerHTML 속성 또는 DOM 함수를 사용하여 src 특성을 통해 스크립트를 로드하지 않고 인라인 코드를 포함하는 script 태그를 삽입합니다.
- script 태그에 대한 src 특성을 설정하여 응용 프로그램 디렉토리 외부에 있는 JavaScript 파일을 로드합니다.
- href="javascript:alert('Test')"의 경우처럼 javascript URL 스킴을 사용합니다.
- setInterval() 또는 setTimout() 함수를 사용합니다. 여기서 첫 번째 매개 변수(함수를 비동기적으로 실행하도록 정의)는 함수 이름이 아닌 평가할 문자열입니다(setTimeout('x = 4', 1000)과 유사).
- document.write() 또는 document.writeln()을 호출합니다.

응용 프로그램 보안 샌드박스의 코드는 내용을 로드하는 동안 이러한 메서드만 사용할 수 있습니다.

이러한 제한 사항은 JSON 객체 리터럴과 함께 eval()을 사용하지 않도록 지정하지 않습니다. 이를 통해 응용 프로그램 내용을 JSON JavaScript 라이브러리에서 사용할 수 있습니다. 그러나 오버로드된 JSON 코드(이벤트 핸들러 사용)를 사용하는 데 제한을 받습니다.

다른 Ajax 프레임워크 및 JavaScript 코드 라이브러리의 경우 프레임워크 또는 라이브러리의 코드가 동적으로 생성된 코드에 대한 이러한 제한 사항 내에서 작동하는지 확인합니다. 그렇게 작동하지 않는 경우 비 응용 프로그램 보안 샌드박스에서 프레임워크 또는 라이브러리를 사용하는 내용을 포함합니다. 자세한 내용은 952페이지의 "AIR 내부의 JavaScript에 대한 제한" 및 993페이지의 "응용 프로그램 내용과 비 응용 프로그램 내용 간 스크립팅"을 참조하십시오. Adobe에서는 http://www.adobe.com/products/air/develop/ajax/features/에서 응용 프로그램 보안 샌드박스를 지원하는 것으로 알려진 Ajax 프레임워크 목록을 유지합니다.

응용 프로그램 보안 샌드박스의 내용과 달리 비 응용 프로그램 보안 샌드박스의 JavaScript 내용은 eval() 함수를 호출하여 동적으로 생성된 코드를 언제든지 실행할 수 있습니다.

AIR API 액세스에 대한 제한 사항(비 응용 프로그램 샌드박스의 경우)

Adobe AIR 1.0 이상

비 응용 프로그램 샌드박스의 JavaScript 코드에서는 window.runtime 객체에 액세스할 수 없으며 그러므로 이 코드에서 AIR API를 실행할 수 없습니다. 비 응용 프로그램 보안 샌드박스의 내용이 다음 코드를 호출하는 경우 응용 프로그램에서 TypeError 예외가 발생합니다.

```
try {
    window.runtime.flash.system.NativeApplication.nativeApplication.exit();
}
catch (e)
{
    alert(e);
}
```

비 응용 프로그램 샌드박스의 내용이 window.runtime 객체를 인식하지 않아 해당 객체를 정의되지 않은 값으로 식별하기 때문에 예외 유형은 TypeError(정의되지 않은 값)입니다.

스크립트 브리지를 사용하여 비 응용 프로그램 샌드박스의 내용에 런타임 기능을 표시할 수 있습니다. 자세한 내용은 993페이지의 "응용 프로그램 내용과 비 응용 프로그램 내용 간 스크립팅"을 참조하십시오.

XMLHttpRequest 호출 사용에 대한 제한 사항

Adobe AIR 1.0 이상

응용 프로그램 보안 샌드박스의 HTML 내용은 동기 XMLHttpRequest 메서드를 사용하여 onLoad 이벤트 동안 HTML 내용이 로드될 때 응용 프로그램 샌드박스 외부에서 데이터를 로드할 수 없습니다.

기본적으로 비 응용 프로그램 보안 샌드박스의 HTML 내용은 JavaScript XMLHttpRequest 객체를 사용하여 요청을 호출하는 도메인 이외의 다른 도메인에서 데이터를 로드할 수 없습니다. frame 또는 iframe 태그에 allowcrosscomainxhr 특성이 포함될 수 있습니다. 이 특성을 null이 아닌 값으로 설정하면 프레임 또는 iframe의 내용이 Javascript XMLHttpRequest 객체를 사용하여 요청을 호출하는 코드의 도메인 이외의 다른 도메인에서 데이터를 로드할 수 있습니다.

```
<iframe id="UI"
    src="http://example.com/ui.html"
    sandboxRoot="http://example.com/"
    allowcrossDomainxhr="true"
    documentRoot="app:/">
</iframe>
```

자세한 내용은 989페이지의 "다른 도메인의 내용 간 스크립팅"을 참조하십시오.

CSS, 프레임, iframe 및 img 요소 로드에 대한 제한 사항(비 응용 프로그램 샌드박스 내용의 경우)
Adobe AIR 1.0 이상

원격(네트워크) 보안 샌드박스의 HTML 내용은 원격 샌드박스(네트워크 URL)에서 CSS, frame, iframe 및 img 내용을 로드할 수만 있습니다.

local-with-filesystem, local-with-networking 또는 local-trusted 샌드박스의 HTML 내용은 응용 프로그램 또는 네트워크 샌드박스가 아닌 로컬 샌드박스에서 CSS, frame, iframe 및 img 내용을 로드할 수만 있습니다.

JavaScript window.open() 메서드 호출에 대한 제한 사항
Adobe AIR 1.0 이상

JavaScript window.open() 메서드에 대한 호출을 통해 생성된 윈도우에 비 응용 프로그램 보안 샌드박스의 내용이 표시되는 경우 윈도우 제목은 주(시작) 윈도우 제목으로 시작하고 뒤에 콜론이 나옵니다. 코드를 사용하여 윈도우 제목의 해당 부분을 화면 밖으로 이동할 수 없습니다.

비 응용 프로그램 보안 샌드박스의 내용은 사용자 마우스 또는 키보드 상호 작용에 의해 트리거된 이벤트에 응답하여 JavaScript window.open() 메서드만 성공적으로 호출할 수 있습니다. 따라서 비 응용 프로그램 내용이 피싱 공격 등 위장하여 사용될 수 있는 윈도우를 만들 수 없습니다. 또한 마우스 또는 키보드 이벤트에 대한 이벤트 핸들러는 setTimeout() 함수 호출 등의 작업을 통해 window.open() 메서드가 지연 후 실행되도록 설정할 수 없습니다.

원격(네트워크) 샌드박스의 내용은 window.open() 메서드를 사용하여 원격 네트워크 샌드박스의 내용을 열 수만 있습니다. window.open() 메서드를 사용하여 응용 프로그램 또는 로컬 샌드박스의 내용을 열 수 없습니다.

local-with-filesystem, local-with-networking 또는 local-trusted 샌드박스(950페이지의 “보안 샌드박스” 참조)의 내용은 window.open() 메서드를 사용하여 로컬 샌드박스의 내용을 열 수만 있습니다. window.open()을 사용하여 응용 프로그램 또는 원격 샌드박스의 내용을 열 수 없습니다.

제한된 코드 호출 시 오류
Adobe AIR 1.0 이상

이러한 보안 제한으로 인해 샌드박스에서의 사용이 제한되는 코드를 호출하면 런타임에서 JavaScript 오류: "응용 프로그램 보안 샌드박스에서의 JavaScript 코드에 대한 Adobe AIR 런타임 보안 위반"을 전달합니다.

자세한 내용은 894페이지의 “보안 관련 JavaScript 오류 방지”를 참조하십시오.

문자열에서 HTML 내용을 로드할 때의 샌드박스 보호
Adobe AIR 1.0 이상

HTMLLoader 클래스의 loadString() 메서드를 사용하면 런타임에 HTML 내용을 만들 수 있습니다. 그러나 안전하지 않은 인터넷 소스에서 데이터를 로드할 경우 HTML 내용으로 사용하는 데이터가 손상될 수 있습니다. 따라서 loadString() 메서드를 사용하여 만들어진 HTML은 기본적으로 응용 프로그램 샌드박스에 배치되지 않으며 AIR API에 액세스할 수 없습니다. 그러나 HTMLLoader 객체의 placeLoadStringContentInApplicationSandbox 속성을 true로 설정하면 loadString() 메서드를 사용하여 만든 HTML을 응용 프로그램 샌드박스에 배치할 수 있습니다. 자세한 내용은 893페이지의 “문자열에서 HTML 내용 로드”를 참조하십시오.

다른 도메인의 내용 간 스크립팅

Adobe AIR 1.0 이상

AIR 응용 프로그램이 설치되면 특수 권한이 부여됩니다. 동일한 권한이 응용 프로그램에 포함되지 않은 원격 파일 및 로컬 파일을 포함하여 기타 내용에 누출되지 않아야 합니다.

AIR 샌드박스 브리지

Adobe AIR 1.0 이상

대개 다른 도메인의 내용은 다른 도메인의 스크립트를 호출할 수 없습니다. 권한이 부여된 정보 또는 제어가 실수로 누출되지 않도록 AIR 응용 프로그램을 보호하기 위해 응용 프로그램 보안 샌드박스의 내용(응용 프로그램과 함께 설치된 내용)에 다음과 같은 제한이 적용됩니다.

- Security.allowDomain() 메서드를 호출하여 응용 프로그램 보안 샌드박스의 코드를 다른 샌드박스에 허용할 수 없습니다. 응용 프로그램 보안 샌드박스에서 이 메서드를 호출하면 오류가 발생합니다.
- LoaderContext.securityDomain 또는 LoaderContext.applicationDomain 속성을 설정하여 비 응용 프로그램 내용을 응용 프로그램 샌드박스로 가져올 수 없습니다.

주 AIR 응용 프로그램에서 원격 도메인의 내용이 주 AIR 응용 프로그램의 스크립트에 대한 액세스를 제어해야 하거나 그 반대의 경우가 여전히 존재합니다. 이를 위해서 런타임에서 두 샌드박스 간 게이트웨이로 사용되는 **샌드박스 브리지** 메커니즘을 제공합니다. 샌드박스 브리지는 원격 응용 프로그램 보안 샌드박스와 응용 프로그램 보안 샌드박스 간의 명시적인 상호 작용을 제공할 수 있습니다.

샌드박스 브리지는 로드된 스크립트와 로드 중인 스크립트 모두 액세스할 수 있는 두 객체를 표시합니다.

- parentSandboxBridge 객체를 사용하면 로드 중인 내용이 로드된 내용의 스크립트에 속성 및 함수를 표시할 수 있습니다.
- childSandboxBridge 객체를 사용하면 로드된 내용이 로드 중인 내용의 스크립트에 속성 및 함수를 표시할 수 있습니다.

샌드박스 브리지를 통해 표시된 객체는 참조가 아닌 값에 따라 전달됩니다. 모든 데이터가 직렬화됩니다. 즉, 브리지의 한 면에서 표시한 객체가 다른 면에 의해 설정될 수 없으며 표시된 모든 객체의 유형이 지정됩니다. 또한 간단한 객체 및 함수만 표시할 수 있으며 복잡한 객체는 표시할 수 없습니다.

자식 내용이 parentSandboxBridge 객체의 속성을 설정하려는 경우 런타임에서 SecurityError 예외가 발생합니다. 마찬가지로, 부모 내용이 childSandboxBridge 객체의 속성을 설정하려는 경우 런타임에서 SecurityError 예외가 발생합니다.

샌드박스 브리지 예(SWF)

Adobe AIR 1.0 이상

AIR music store 응용 프로그램에서 원격 SWF 파일이 앨범 가격을 브로드캐스트할 수 있기를 원하지만 원격 SWF 파일에서 가격이 할인 가격인지 여부를 공개하길 원하지 않을 수 있습니다. 이를 위해서 StoreAPI 클래스에서 가격을 구하는 메서드를 제공하지만 할인 가격은 숨깁니다. 그러고 나서 이 StoreAPI 클래스의 인스턴스가 원격 SWF를 로드하는 Loader 객체의 LoaderInfo 객체에 대한 parentSandboxBridge 속성에 할당됩니다.

다음은 AIR music store에 대한 코드입니다.

```
<?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>
```

StoreAPI 객체는 주 응용 프로그램을 호출하여 일반 앨범 가격을 가져오지만 getSaleAlbumPrice() 메서드가 호출될 때 “사용할 수 없음”을 반환합니다. 다음 코드에서는 StoreAPI 클래스를 정의합니다.

```
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();
    }
}
```

다음 코드에서는 저장소의 가격을 보고하지만 할인 가격은 보고할 수 없는 PriceQuoter SWF 파일의 예를 보여 줍니다.

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

샌드박스 브리지 예(HTML)

Adobe AIR 1.0 이상

HTML 내용에서 parentSandboxBridge 및 childSandboxBridge 속성이 자식 문서의 JavaScript window 객체에 추가됩니다. HTML 내용의 브리지 함수를 설정하는 방법에 대한 예는 909페이지의 “샌드박스 브리지 인터페이스 설정”을 참조하십시오.

API 노출 제한

Adobe AIR 1.0 이상

샌드박스 브리지를 표시할 때 남용 정도를 제한하는 높은 수준의 API를 표시하는 것이 중요합니다. 브리지 구현을 호출하는 내용이 코드 삽입 등을 통해 손상될 수 있습니다. 따라서 예를 들어 브리지를 통해 임의 파일의 내용을 읽는 readFile(path:String) 메서드 표시가 쉽게 남용될 수 있습니다. 경로를 사용하지 않고 특정 파일을 읽는 readApplicationSetting() API를 표시하는 것이 더 좋습니다. 보다 의미적인 접근 방식은 응용 프로그램의 일부가 손상된 경우 해당 응용 프로그램에서 유발할 수 있는 손상을 제한합니다.

기타 도움말 항목

908페이지의 “서로 다른 보안 샌드박스의 내용 크로스 스크립팅”

951페이지의 “AIR 응용 프로그램 샌드박스”

디스크에 쓰기

Adobe AIR 1.0 이상

웹 브라우저에서 실행 중인 응용 프로그램은 사용자의 로컬 파일 시스템과 제한적으로만 상호 작용합니다. 웹 브라우저는 웹 내용 로드 결과로 사용자의 컴퓨터가 손상되지 않도록 하는 보안 정책을 구현합니다. 예를 들어 브라우저에서 Flash Player를 통해 실행하는 SWF 파일은 사용자 컴퓨터에 이미 있는 파일과 직접 상호 작용할 수 없습니다. 공유 객체 및 쿠키는 사용자 환경 설정 및 기타 데이터를 유지하기 위해 사용자 컴퓨터에 기록될 수 있지만 이는 파일 시스템 상호 작용의 한계입니다. AIR 응용 프로그램은 기본적으로 설치되므로 로컬 파일 시스템 전체에서 읽고 쓰는 기능을 포함하는 서로 다른 보안 계약을 가지고 있습니다.

이러한 자유로운 환경이 제공되는 반면 개발자의 책임도 강화됩니다. 갑작스러운 응용 프로그램 보안 문제는 응용 프로그램의 기능뿐만 아니라 사용자 컴퓨터의 무결성도 저해합니다. 따라서 개발자는 994페이지의 "개발자에 대한 최선의 보안 방법"을 읽어야 합니다.

AIR 개발자는 여러 URL 스킴 규칙을 사용하여 파일에 액세스하고 파일을 로컬 파일 시스템에 쓸 수 있습니다.

URL 스킴	설명
app:/	응용 프로그램 디렉토리에 대한 별칭입니다. 이 경로에서 액세스한 파일에 응용 프로그램 샌드박스가 할당되며 런타임에 의해 부여된 전체 권한이 제공됩니다.
app-storage:/	로컬 저장소 디렉토리에 대한 별칭으로, 런타임에 의해 표준화됩니다. 이 경로에서 액세스한 파일에 비 응용 프로그램 샌드박스가 할당됩니다.
file:///	사용자 하드 디스크의 루트를 나타내는 별칭입니다. 이 경로에서 액세스한 파일에는 파일이 응용 프로그램 디렉토리에 있으면 응용 프로그램 샌드박스가 할당되고, 그렇지 않으면 비 응용 프로그램 샌드박스가 할당됩니다.

참고: AIR 응용 프로그램은 app: URL 스킴을 사용하여 내용을 수정할 수 없습니다. 또한 응용 프로그램 디렉토리는 관리자 설정에 의해서만 읽을 수 있습니다.

사용자 컴퓨터에 대한 관리자 제한 사항이 없으면 AIR 응용 프로그램에서 사용자 하드 드라이브의 어떤 위치에도 쓸 수 있습니다. 개발자는 해당 응용 프로그램 관련 로컬 저장소에 대해 app-storage:/ 경로를 사용하는 것이 좋습니다. 응용 프로그램에서 app-storage:/로 작성된 파일은 다음과 같이 표준 위치에 배치됩니다.

- Mac OS의 경우: 응용 프로그램의 저장소 디렉토리는 <appData>/<appId>/Local Store/이며 여기서 <appData>는 사용자의 환경 설정 폴더입니다. 일반적으로 /Users/<user>/Library/Preferences입니다.
- Windows의 경우: 응용 프로그램의 저장소 디렉토리는 <appData>\<appId>\Local Store\이며 여기서 <appData>는 사용자의 CSIDL_APPDATA Special Folder입니다. 일반적으로 C:\Documents and Settings\<userName>\Application Data입니다.
- Linux의 경우: <appData>/<appID>/Local Store/이며 여기서 <appData>는 /home/<user>/.appdata입니다.

응용 프로그램이 사용자 파일 시스템의 기존 파일과 상호 작용하도록 설계된 경우 994페이지의 "개발자에 대한 최선의 보안 방법"을 읽어야 합니다.

신뢰할 수 없는 내용과 안전하게 작업

Adobe AIR 1.0 이상

응용 프로그램 샌드박스에 할당되지 않은 내용은 런타임의 보안 기준에 맞는 경우에만 응용 프로그램에 추가 스크립팅 기능을 제공할 수 있습니다. 이 항목에서는 비 응용 프로그램 내용을 포함하는 AIR 보안 계약에 대해 설명합니다.

Security.allowDomain()

Adobe AIR 1.0 이상

AIR 응용 프로그램은 Flash Player 브라우저 플러그인에서 신뢰할 수 없는 내용에 대한 스크립팅 액세스를 제한하는 것보다 더 엄격하게 비 응용 프로그램 내용에 대한 스크립팅 액세스를 제한합니다. 예를 들어 브라우저의 Flash Player에서 local-trusted 샌드박스에 할당된 SWF 파일이 System.allowDomain() 메서드를 호출하는 경우 지정된 도메인에서 로드된 모든 SWF에 스크립팅 액세스 권한이 부여됩니다. AIR 응용 프로그램의 응용 프로그램 내용에서 유사한 접근 방식이 허용되지 않습니다. 그 이유는 사용자 파일 시스템에 비 응용 프로그램 파일에 대한 비정상적인 액세스 권한을 부여하기 때문입니다. 원격 파일은 Security.allowDomain() 메서드에 대한 호출에 상관없이 응용 프로그램 샌드박스에 직접 액세스할 수 없습니다.

응용 프로그램 내용과 비 응용 프로그램 내용 간 스크립팅

Adobe AIR 1.0 이상

응용 프로그램 내용과 비 응용 프로그램 내용 간을 스크립팅하는 AIR 응용 프로그램의 보안 배열은 다소 복잡합니다. 응용 프로그램 샌드박스에 없는 파일만 샌드박스 브리지를 사용하여 응용 프로그램 샌드박스에 있는 파일의 속성 및 메서드에 액세스할 수 있습니다. 샌드박스 브리지는 응용 프로그램 내용과 비 응용 프로그램 내용 간 게이트웨이로 사용되어 두 파일 간의 명시적인 상호 작용을 제공합니다. 샌드박스 브리지는 올바르게 사용되는 경우 추가 보안 계층을 제공하여 비 응용 프로그램 내용이 응용 프로그램 내용에 포함된 객체 참조에 액세스하지 못하도록 지정합니다.

샌드박스 브리지의 장점은 예제에 가장 잘 설명되어 있습니다. AIR music store 응용 프로그램이 저장소 응용 프로그램이 통신할 수 있는 자체 SWF 파일을 만들려고 하는 광고자에게 API를 제공하려고 합니다. 저장소는 광고자에게 저장소에서 아티스트와 CD를 조회하는 메서드를 제공할 뿐만 아니라 보안을 위해 타사 SWF 파일의 일부 메서드 및 속성을 격리하려고 합니다.

샌드박스 브리지가 이러한 기능을 제공할 수 있습니다. 기본적으로 런타임에 AIR 응용 프로그램에 외부적으로 로드된 내용은 주 응용 프로그램의 메서드 또는 속성에 액세스할 수 없습니다. 개발자는 사용자 정의 샌드박스 브리지 구현을 통해 이러한 메서드 또는 속성을 표시하지 않고 원격 내용에 서비스를 제공할 수 있습니다. 샌드박스 브리지를 신뢰할 수 있는 내용과 신뢰할 수 없는 내용 간 통로로 사용하여 객체 참조를 표시하지 않고 로더와 로드되는 내용 간 통신을 제공할 수 있도록 합니다.

샌드박스 브리지를 안전하게 사용하는 방법에 대한 자세한 내용은 989페이지의 “다른 도메인의 내용 간 스크립팅”을 참조하십시오.

동적으로 생성되는 안전하지 않은 SWF 내용 보호

Adobe AIR 1.0 이상

Loader.loadBytes() 메서드는 응용 프로그램이 바이트 배열에서 SWF 내용을 생성하는 방법을 제공합니다. 그러나 원격 소스에서 로드한 데이터에 대한 삽입 공격은 내용을 로드할 때 심각한 손상을 일으킬 수 있습니다. 이러한 손상은 특히 생성된 SWF 내용이 전체 AIR API 집합에 액세스할 수 있는 응용 프로그램 샌드박스로 데이터를 로드할 때 발생합니다.

실행 SWF 코드를 생성하지 않고 loadBytes() 메서드를 이용하는 합법적인 사용이 있습니다. 예를 들어 loadBytes() 메서드를 사용하여 이미지 표시의 타이밍을 제어하는 이미지 데이터를 생성할 수 있습니다. 오디오 재생을 위한 SWF 내용의 동적 생성과 같이 실행 중인 코드를 **이용하는** 합법적인 사용도 있습니다. AIR에서 기본적으로 loadBytes() 메서드는 사용자가 SWF 내용을 로드하도록 **허용하지 않고** 사용자가 이미지 내용만 로드하도록 허용합니다. AIR에서 loadBytes() 메서드의 loaderContext 속성은 allowLoadBytesCodeExecution 속성을 포함합니다. 사용자는 이 속성을 true로 설정하여 응용 프로그램이 loadBytes()를 사용하여 실행 SWF 내용을 로드하도록 명시적으로 허용할 수 있습니다. 다음 코드에서는 이 기능을 사용하는 방법을 보여 줍니다.

```
var loader:Loader = new Loader();
var loaderContext:LoaderContext = new LoaderContext();
loaderContext.allowLoadBytesCodeExecution = true;
loader.loadBytes(bytes, loaderContext);
```

loadBytes()를 호출하여 SWF 내용을 로드하고 LoaderContext 객체의 allowLoadBytesCodeExecution 속성을 false(기본값)로 설정하면 Loader 객체에서 SecurityError 예외가 발생합니다.

참고: Adobe AIR의 이후 릴리스에서는 이 API가 변경될 수 있습니다. 그 경우 사용자는 LoaderContext 클래스의 allowLoadBytesCodeExecution 속성을 사용하는 내용을 다시 컴파일할 수 있습니다.

개발자에 대한 최선의 보안 방법

Adobe AIR 1.0 이상

웹 기술을 사용하여 AIR 응용 프로그램을 작성하는 경우에도 개발자는 브라우저 보안 샌드박스 내에서 작업하고 있지 않음을 인식해야 합니다. 즉, 로컬 시스템에 의도적으로 또는 예기치 않게 손상을 줄 수 있는 AIR 응용 프로그램을 작성할 수 있습니다. AIR은 이러한 위험을 최소화하려고 하지만 취약성이 발생할 수 있는 경우가 여전히 존재합니다. 이 항목에서는 중요한 잠재적 보안 문제를 다룹니다.

파일을 응용 프로그램 보안 샌드박스로 가져올 위험

Adobe AIR 1.0 이상

응용 프로그램 디렉토리에 있는 파일은 응용 프로그램 샌드박스에 할당되고 런타임의 전체 권한을 가집니다. 로컬 파일 시스템에 쓰는 응용 프로그램은 app-storage:/에 쓰는 것이 좋습니다. 이 디렉토리는 사용자 컴퓨터에 있는 응용 프로그램 파일과 별개로 존재하므로 파일이 응용 프로그램 샌드박스에 할당되지 않고 다소 약화된 보안 위험을 제공합니다. 개발자는 다음 사항을 고려하는 것이 좋습니다.

- 필요한 경우에만 설치된 응용 프로그램의 AIR 파일에 파일을 포함합니다.
- 비헤이비어를 완전히 이해하고 신뢰하는 경우에만 설치된 응용 프로그램의 AIR 파일에 스크립팅 파일을 포함합니다.
- 응용 프로그램 디렉토리의 내용에 쓰거나 내용을 수정하지 않습니다. 런타임은 SecurityError 예외를 발생하여 app:/ URL 스킴을 통해 응용 프로그램이 파일 및 디렉토리를 쓰거나 수정할 수 없도록 지정합니다.
- 네트워크 소스의 데이터를 코드 실행을 유발할 수 있는 AIR API의 메서드에 대한 매개 변수로 사용하지 않습니다. 여기에는 Loader.loadBytes() 메서드 및 JavaScript eval() 함수의 사용이 포함됩니다.

외부 소스를 사용하여 경로를 결정할 위험

Adobe AIR 1.0 이상

AIR 응용 프로그램은 외부 데이터 또는 내용을 사용할 때 손상될 수 있습니다. 따라서 네트워크 또는 파일 시스템의 데이터를 사용할 때 특별한 주의를 기울여야 합니다. 신뢰의 책임은 전적으로 개발자 및 개발자가 만드는 네트워크 연결에 있지만 외부 데이터를 로드하는 것은 본질적으로 위험하므로 중요한 작업에 대한 입력에는 사용하지 않습니다. 개발자는 다음 작업은 수행하지 않는 것이 좋습니다.

- 네트워크 소스의 데이터를 사용하여 파일 이름을 결정합니다.
- 네트워크 소스의 데이터를 사용하여 응용 프로그램이 개인 정보를 보내는 데 사용하는 URL을 생성합니다.

안전하지 않은 자격 증명을 사용, 저장 또는 전송할 위험

Adobe AIR 1.0 이상

사용자의 로컬 파일 시스템에 사용자 자격 증명을 저장하면 본질적으로 이러한 자격 증명이 손상될 수 있는 위험이 발생합니다. 개발자는 다음 사항을 고려하는 것이 좋습니다.

- 자격 증명을 로컬로 저장해야 하는 경우 로컬 파일 시스템에 쓸 때 자격 증명을 암호화합니다. 런타임은 EncryptedLocalStore 클래스를 통해 각각의 설치된 응용 프로그램에 고유한 암호화된 저장소를 제공합니다. 자세한 내용은 647페이지의 “암호화된 로컬 저장소”를 참조하십시오.

- 네트워크 소스를 신뢰할 수 없고 HTTPS: 또는 TLS(Transport Layer Security) 프로토콜이 전송에 사용되지 않을 경우 암호화되지 않은 사용자 자격 증명을 해당 소스로 전송하지 마십시오.
- 자격 증명을 만들 때 기본 암호를 지정하지 마십시오. 사용자가 직접 자신의 암호를 지정해야 합니다. 기본값을 그대로 두는 사용자는 기본 암호를 이미 알고 있는 공격자에게 자신의 자격 증명을 노출하게 됩니다.

다운그레이드 공격 위험

Adobe AIR 1.0 이상

응용 프로그램을 설치하는 동안 런타임은 응용 프로그램 버전이 현재 설치되어 있지 않은지 확인합니다. 응용 프로그램이 이미 설치되어 있으면 런타임은 설치 중인 버전에 대해 버전 문자열을 비교합니다. 이 문자열이 다른 경우 사용자는 설치를 업그레이드하도록 선택할 수 있습니다. 런타임은 새로 설치된 버전이 기존 버전보다 새 버전임을 보장하지 않고 단지 두 버전이 서로 다른지 여부만 확인합니다. 공격자는 사용자에게 이전 버전을 배포하여 보안 약점을 피할 수 있습니다. 따라서 개발자는 응용 프로그램이 실행될 때 버전 감지를 수행하는 것이 좋습니다. 응용 프로그램이 네트워크에서 필수 업데이트를 검사하도록 지정하는 것이 좋습니다. 그렇게 하면 공격자로 인해 사용자가 기존 버전을 실행하게 된 경우에도 해당 기존 버전은 업데이트해야 함을 인식합니다. 또한 응용 프로그램에 대해 분명한 버전 지정 스킴을 사용하면 사용자에게 다운그레이드된 버전을 설치하도록 속이는 것이 보다 어려워집니다.

코드 서명

Adobe AIR 1.0 이상

모든 AIR 설치 프로그램 파일은 코드 서명되어야 합니다. 코드 서명은 소프트웨어의 지정된 원본이 정확한지 확인하는 암호화 프로세스입니다. AIR 응용 프로그램은 외부 인증 기관(CA)에서 발행된 인증서 또는 사용자가 직접 만든 자체 서명된 인증서를 사용하여 서명할 수 있습니다. 잘 알려진 CA의 상용 인증서를 사용하는 것이 좋으며 이러한 인증서를 사용하면 사용자가 현재 위조된 대상이 아닌 자신의 응용 프로그램에 설치하고 있다는 확신을 가질 수 있습니다. 그러나 자체 서명된 인증서는 SDK의 adt를 사용하여 만들거나 인증서 생성에 adt를 사용하는 Flash, Flash Builder 또는 기타 응용 프로그램을 사용하여 만들 수 있습니다. 자체 서명된 인증서는 설치하려는 응용 프로그램이 정품이고 공개 릴리스 전 응용 프로그램을 테스트하는 목적으로만 사용할 수 있다는 것에 대해 어떠한 보증도 제공하지 않습니다.

Android 장치의 보안

Adobe AIR 2.5 이상

모든 컴퓨팅 장치에서와 마찬가지로 Android에서도 AIR는 기본 보안 모델을 따릅니다. 동시에 AIR는 자체 보안 규칙을 유지하므로 이를 통해 개발자가 안전한 인터넷 연결 응용 프로그램을 더욱 쉽게 작성할 수 있습니다.

Android에서 AIR 응용 프로그램은 Android 패키지 형식을 사용하므로 설치는 Android 보안 모델에 따릅니다. AIR 응용 프로그램 설치 프로그램은 사용되지 않습니다.

Android 보안 모델의 주요 세 가지 측면은 다음과 같습니다.

- 권한
- 응용 프로그램 서명
- 응용 프로그램 사용자 ID

Android 권한

Android의 여러 기능은 운영 체제 권한 메커니즘에 의해 보호됩니다. 보호되는 기능을 사용하려면 AIR 응용 프로그램 설명자에서 응용 프로그램이 필요한 권한을 요구함을 선언해야 합니다. 사용자가 응용 프로그램을 설치하려고 하면 Android 운영 체제는 설치가 진행되기 전에 요청된 모든 권한을 사용자에게 표시합니다.

대부분의 AIR 응용 프로그램은 Android 권한을 응용 프로그램 설명자에 지정해야 합니다. 기본적으로 아무런 권한도 포함되지 않습니다. 다음은 AIR 런타임에서 노출되는 보호된 Android 기능에 필요한 권한입니다.

ACCESS_COARSE_LOCATION 응용 프로그램에서 Geolocation 클래스를 통해 WIFI 및 셀룰러 네트워크 위치 데이터에 액세스할 수 있도록 허용합니다.

ACCESS_FINE_LOCATION 응용 프로그램에서 Geolocation 클래스를 통해 GPS 데이터에 액세스할 수 있도록 허용합니다.

ACCESS_NETWORK_STATE 및 ACCESS_WIFI_STATE 응용 프로그램에서 NetworkInfo 클래스를 통해 네트워크 정보에 액세스할 수 있도록 허용합니다.

CAMERA 응용 프로그램에서 카메라에 액세스할 수 있도록 허용합니다.

INTERNET 응용 프로그램에서 네트워크 요청을 수행할 수 있도록 허용하고 원격 디버깅도 허용합니다.

READ_PHONE_STATE 수신 호출이 발생하면 AIR 런타임에 오디오를 음소거할 수 있도록 허용합니다.

RECORD_AUDIO 응용 프로그램에서 마이크에 액세스할 수 있도록 허용합니다.

WAKE_LOCK 및 DISABLE_KEYGUARD 응용 프로그램에서 장치가 SystemIdleMode 클래스 설정을 사용하여 대기 모드로 들어가지 못하게 할 수 있도록 허용합니다.

WRITE_EXTERNAL_STORAGE 응용 프로그램에서 장치의 외부 메모리 카드에 쓸 수 있도록 허용합니다.

응용 프로그램 서명

Android 플랫폼용으로 만들어진 모든 응용 프로그램 패키지는 반드시 서명되어야 합니다. Android에서 AIR 응용 프로그램은 기본 Android APK 형식으로 패키지되어 있기 때문에 AIR 규칙이 아니라 Android 규칙에 따라 서명됩니다. Android와 AIR는 서로 비슷한 방식으로 코드 서명을 사용하지만 다음과 같은 중요한 차이점이 있습니다.

- Android에서 서명을 통해 개인 키가 개발자의 소유임을 확인할 수 있지만 해당 개발자의 ID를 확인할 수는 없습니다.
- Android 마켓에 제공된 응용 프로그램의 경우 인증서는 최소한 25년 동안 유효해야 합니다.
- Android에서 패키지 서명을 다른 인증서로 마이그레이션하는 것은 지원되지 않습니다. 업데이트가 다른 인증서에 의해 서명된 경우 사용자는 원래의 응용 프로그램을 제거해야 업데이트된 응용 프로그램을 설치할 수 있습니다.
- 동일한 인증서로 서명된 두 개의 응용 프로그램은 서로의 캐시 및 데이터 파일에 액세스할 수 있도록 허용하는 공유 ID를 지정할 수 있습니다. 이러한 공유는 AIR에서 지원되지 않습니다.)

응용 프로그램 사용자 ID

Android는 Linux 커널을 사용합니다. 설치된 모든 응용 프로그램에는 파일 액세스와 같은 작업에 대한 권한을 결정하는 Linux 형식의 사용자 ID가 할당됩니다. 응용 프로그램, 응용 프로그램 저장소 및 임시 디렉토리의 파일은 파일 시스템 권한에 따라 액세스로부터 보호됩니다. SD 카드와 같은 외부 저장소에 작성된 파일은 해당 SD 카드가 컴퓨터에 대용량 저장 장치로 마운트된 경우 다른 응용 프로그램 또는 사용자가 읽고 수정하고 삭제할 수 있습니다.

인터넷 요청을 통해 수신된 쿠키는 AIR 응용 프로그램 간에 공유되지 않습니다.

배경 이미지의 개인 정보 보호

사용자가 응용 프로그램을 백그라운드로 전환할 경우 일부 Android 버전은 최근 응용 프로그램 목록에서 축소판을 사용하는 스크린 샷을 캡처합니다. 이 스크린 샷은 장치 메모리에 저장되므로 장치를 물리적으로 장악한 공격자가 이 정보에 액세스할 수 있습니다.

응용 프로그램이 중요한 정보를 표시할 경우 배경 스크린 샷이 해당 정보를 캡처하지 않도록 보호해야 합니다.
NativeApplication 객체가 전달하는 deactivate 이벤트는 응용 프로그램이 배경으로 전환될 때 이를 알리는 역할을 합니다. 이 이벤트를 사용하여 중요한 정보를 지우거나 숨기십시오.

기타 도움말 항목

Android: Security and Permissions

Android의 암호화된 데이터

Android의 AIR 응용 프로그램에서는 내장 SQL 데이터베이스에서 사용 가능한 암호화 옵션을 통해 암호화된 데이터를 저장할 수 있습니다. 최적의 보안을 위해 응용 프로그램이 실행될 때마다 사용자가 입력하는 암호에 기초한 암호화 키를 사용합니다. 로컬로 저장된 암호화 키 또는 암호는 응용 프로그램 파일에 액세스할 수 있는 공격자로부터 숨기는 것이 힘들거나 불가능합니다. 공격자가 키를 가져올 수 있는 경우 데이터를 암호화해도 Android 시스템이 제공하는 사용자 ID 기반 파일 시스템 보안 이상의 추가 보호가 제공되지 않습니다.

EncryptedLocalStore 클래스를 사용하여 데이터를 저장할 수 있지만 이러한 데이터는 Android 장치에서 암호화되지 않습니다. 대신 Android 보안 모델은 응용 프로그램 사용자 ID를 사용하여 다른 응용 프로그램으로부터 데이터를 보호합니다. 공유된 사용자 ID를 사용하고 동일한 코드 서명 인증서로 서명된 응용 프로그램은 암호화된 동일한 로컬 저장소를 사용합니다.

중요: 루트로 지정된 전화에서 루트 권한으로 실행 중인 모든 응용 프로그램은 다른 응용 프로그램의 파일에 액세스할 수 있습니다. 따라서 암호화된 로컬 저장소를 사용하여 저장된 데이터는 루트로 지정된 장치에서 안전하지 않습니다.

iOS 장치의 보안

iOS에서 AIR는 기본 보안 모델을 따릅니다. 동시에 AIR는 자체 보안 규칙을 유지하므로 이를 통해 개발자가 안전한 인터넷 연결 응용 프로그램을 더욱 쉽게 작성할 수 있습니다.

iOS에서 AIR 응용 프로그램은 iOS 패키지 형식을 사용하므로 설치는 iOS 보안 모델을 따릅니다. AIR 응용 프로그램 설치 프로그램은 사용되지 않습니다. 또한 iOS 장치에서 별도의 AIR 런타임이 사용되지 않습니다. 모든 AIR 응용 프로그램에는 올바르게 작동하는 데 필요한 모든 코드가 포함되어 있습니다.

응용 프로그램 서명

iOS 플랫폼용으로 만들어진 모든 응용 프로그램 패키지는 반드시 서명되어야 합니다. iOS의 AIR 응용 프로그램은 기본 iOS IPA 형식으로 패키지화되기 때문에 AIR 요구 사항이 아닌 iOS 요구 사항에 따라 서명됩니다. iOS와 AIR은 서로 비슷한 방식으로 코드 서명을 사용하지만 다음과 같은 중요한 차이점이 있습니다.

- iOS에서는 응용 프로그램 서명에 사용되는 인증서가 Apple에서 발행한 것이어야 하며, 다른 인증 기관의 인증서는 사용할 수 없습니다.
- iOS에서는 Apple에서 발행한 배포 인증서의 유효 기간이 대개 1년입니다.

배경 이미지의 개인 정보 보호

사용자가 iOS에서 응용 프로그램을 배경으로 전환하면 운영 체제가 전환 효과에 애니메이션을 적용할 때 사용하는 스크린 샷을 캡처합니다. 이 스크린 샷은 장치 메모리에 저장되므로 장치를 물리적으로 장악한 공격자가 이 정보에 액세스할 수 있습니다.

응용 프로그램이 중요한 정보를 표시할 경우 배경 스크린 샷이 해당 정보를 캡처하지 않도록 보호해야 합니다.
NativeApplication 객체가 전달하는 deactivate 이벤트는 응용 프로그램이 배경으로 전환될 때 이를 알리는 역할을 합니다. 이 이벤트를 사용하여 중요한 정보를 지우거나 숨기십시오.

65장: ActionScript 예제 사용 방법

특정 클래스 및 메서드의 작동 방법을 익히는 가장 효과적인 방법 중 한 가지는 ActionScript 3.0 코드 예제를 실행하는 것입니다. 예제는 사용 중이거나 대상으로 지정한 장치에 따라 다양한 방법으로 사용할 수 있습니다.

Flash Professional 또는 Flash Builder를 실행하는 컴퓨터 이러한 개발 환경에서 ActionScript 3.0 예제를 실행하는 방법에 대한 자세한 내용은 1000페이지의 "Flash Professional에서 ActionScript 3.0 예제 실행" 또는 1001페이지의 "Flash Builder에서 ActionScript 3.0 예제 실행"을 참조하십시오. 코드 예제의 작동 방식을 보다 확실하게 이해하려면 추적 명령문 및 기타 디버깅 도구를 사용하십시오.

모바일 장치 Flash Player 10.1 이상 버전을 지원하는 모바일 장치에서 ActionScript 3.0 코드 예제를 실행할 수 있습니다. 자세한 내용은 1002페이지의 "모바일 장치에서 ActionScript 3.0 예제 실행"을 참조하십시오. 또한 Flash Professional 또는 Flash Builder를 사용하는 컴퓨터에서도 이러한 예제를 실행할 수 있습니다.

TV 장치 TV 장치에서는 이러한 예제를 실행할 수 없지만 이를 컴퓨터에서 실행하는 방식으로 여전히 예제를 통해 학습할 수 있습니다. TV 장치용 응용 프로그램을 개발하는 방법에 대한 자세한 내용은 Adobe Developer Connection 웹 사이트에서 Flash Platform for TV를 참조하십시오.

예제 유형

ActionScript 3.0 코드 예제의 유형은 다음과 같습니다.

- 코드 조각 예제(ActionScript 3.0 설명서 모음 전반에 걸쳐 나와 있음)
- 클래스 기반 예제(주로 ActionScript 3.0 언어 참조에서 확인 가능)
- 여러 소스 파일이 포함된 실습 예제(www.adobe.com/go/learn_programmingAS3samples_flash_kr에서 소스 ZIP 파일 다운로드)

코드 조각 예제

코드 조각 예제는 다음과 같은 형식으로 되어 있습니다.

```
var x:int = 5;
trace(x); // 5
```

코드 조각에는 단일 개념을 표현하는 데 필요한 코드만 포함됩니다. 패키지 또는 클래스 문은 일반적으로 포함되지 않습니다.

클래스 기반 예제

많은 예제에서 완전한 ActionScript 클래스의 소스 코드를 보여 줍니다. 클래스 기반 예제는 다음과 같은 형식으로 되어 있습니다.

```
package {
    public class Example1 {
        public function Example1():void {
            var x:int = 5;
            trace(x); //5
        }
    }
}
```

클래스 기반 예제의 코드에는 패키지 문, 클래스 선언 및 생성자 함수가 포함됩니다.

여러 소스 파일을 포함하는 실습 예제

ActionScript 3.0 개발자 안내서의 많은 항목에서는 마지막 부분에 실제 환경에서 특정 ActionScript 기능들을 사용하는 방법을 보여 주는 실습 예제를 싣고 있습니다. 이러한 예제에는 일반적으로 다음을 비롯한 여러 파일이 포함됩니다.

- 하나 이상의 ActionScript 소스 파일
- Flash Professional에 사용하기 위한 .FLA 파일
- Flash Builder에 사용하기 위한 하나 이상의 MXML 파일
- 데이터 파일, 이미지 파일, 사운드 파일 또는 예제 응용 프로그램에서 사용되는 기타 에셋(선택 사항)

실습 예제는 일반적으로 ZIP 아카이브 파일로 제공됩니다.

ZIP 파일로 제공되는 개발자 안내서 예제 목록

Flash Professional CS5 및 Flex 4용 ZIP 파일(www.adobe.com/go/learn_programmingAS3samples_flash_kr에서 다운로드)에는 다음 예제가 포함되어 있습니다.

- AlarmClock(127페이지의 "이벤트 처리 예제: 알람 시계")
- AlgorithmicVisualGenerator(210페이지의 "드로잉 API 예제: Algorithmic Visual Generator")
- ASCIIArt(17페이지의 "문자열 예제: ASCII Art")
- CapabilitiesExplorer(794페이지의 "기능 예제: 시스템 기능 검색")
- CustomErrors(62페이지의 "오류 처리 예제: CustomErrors 응용 프로그램")
- DisplayObjectTransformer(196페이지의 "기하 도형 예제: 표시 객체에 행렬 변환 적용")
- FilterWorkbench(264페이지의 "표시 객체 필터링 예제: Filter Workbench")
- GlobalStockTicker(866페이지의 "예제: 주식 시세 표시 응용 프로그램 국제화")
- IntrovertIM_HTML(772페이지의 "외부 API 예제: 웹 브라우저에서 ActionScript와 JavaScript 간의 통신")
- NewsLayout(349페이지의 "TextField 예제: 신문 스타일 텍스트 서식")
- PlayList(41페이지의 "배열 예제: PlayList")
- PodcastPlayer(422페이지의 "사운드 예제: 포드캐스트 플레이어")
- ProjectionDragger(322페이지의 "예제: 원근 투영")
- ReorderByZ(326페이지의 "Matrix3D 객체를 사용하여 표시 순서 재지정")
- RSSViewer(103페이지의 "ActionScript의 XML 예제: 인터넷에서 RSS 데이터 로드")
- RuntimeAssetsExplorer(299페이지의 "동영상 클립 예제: RuntimeAssetsExplorer")
- SimpleClock(6페이지의 "날짜 및 시간 예제: 간단한 아날로그 시계")
- SpinningMoon(230페이지의 "비트맵 예제: 회전하는 달 애니메이션")
- SpriteArranger(183페이지의 "표시 객체 예제: SpriteArranger")
- TelnetSocket(727페이지의 "TCP 소켓 예제: Telnet 클라이언트 구축")
- VideoJukebox(458페이지의 "비디오 예제: 비디오 주크박스")
- WikiEditor(81페이지의 "일반 표현식 예제: Wiki 파서")
- WordSearch(524페이지의 "마우스 입력 예제: WordSearch")

실습 예제는 Flash 개발자 센터 및 Flex 개발자 센터의 여러 퀵 스타트 문서에서도 찾아볼 수 있습니다.

Flash Professional에서 ActionScript 3.0 예제 실행

예제 유형에 따라 다음 절차 중 하나를 사용하여 Flash Professional을 통해 예제를 실행합니다.

Flash Professional에서 코드 조각 예제 실행

Flash Professional에서 코드 조각 예제를 실행하려면

1. [파일] > [새로 만들기]를 선택합니다.
2. [새 문서] 대화 상자에서 [Flash 문서]를 선택하고 [확인]을 클릭합니다.

   새 Flash 윈도우가 나타납니다.
3. [타임라인] 패널의 첫 번째 레이어에서 첫 번째 프레임을 클릭합니다.
4. [액션] 패널에서 코드 조각 예제를 입력하거나 붙여 넣습니다.
5. [파일] > [저장]을 선택합니다. 파일에 이름을 지정한 다음 [확인]을 클릭합니다.
6. 예제를 테스트하려면 [컨트롤] > [동영상 테스트]를 선택합니다.

Flash Professional에서 클래스 기반 예제 실행

Flash Professional에서 클래스 기반 예제를 실행하려면

1. [파일] > [새로 만들기]를 선택합니다.
2. [새 문서] 대화 상자에서 [ActionScript 파일]을 선택하고 [확인]을 클릭합니다. 새 편집기 윈도우가 나타납니다.
3. 클래스 기반 예제 코드를 복사하여 편집기 윈도우에 붙여 넣습니다.

   해당 클래스가 프로그램의 기본 문서 클래스인 경우 다음과 같이 MovieClip 클래스를 확장해야 합니다.

   ```
   import flash.display.MovieClip;
   public class Example1 extends MovieClip{
   //...
   }
   ```

   또한 예제에서 참조되는 모든 클래스는 import 문을 사용하여 선언해야 합니다.
4. [파일] > [저장]을 선택합니다. 파일 이름을 예제의 클래스 이름과 동일하게 지정합니다(예: ContextMenuExample.as).

   참고: flashx.textLayout.container.ContainerController 클래스 예제와 같은 일부 클래스 기반 예제는 패키지 선언(package flashx.textLayout.container.examples {)에 여러 수준을 포함합니다. 이러한 예제의 경우 파일을 패키지 선언과 일치하는 하위 폴더(flashx/textLayout/container/examples)에 저장하거나, 패키지 이름을 제거하여 ActionScript가 package { 만으로 시작하도록 하면 모든 위치에서 파일을 테스트할 수 있습니다.
5. [파일] > [새로 만들기]를 선택합니다.
6. [새 문서] 대화 상자에서 [Flash 문서(ActionScript 3.0)]를 선택하고 [확인]을 클릭합니다. 새 Flash 윈도우가 나타납니다.
7. [속성] 패널에서 [문서 클래스] 필드에 예제 클래스의 이름을 입력합니다. 이 이름은 바로 전에 저장한 ActionScript 소스 파일의 이름(예: ContextMenuExample)과 일치해야 합니다.
8. [파일] > [저장]을 선택합니다. FLA 파일 이름을 예제의 클래스 이름과 동일하게 지정합니다(예: ContextMenuExample.fla).
9. 예제를 테스트하려면 [컨트롤] > [동영상 테스트]를 선택합니다.

Flash Professional에서 실습 예제 실행

실습 예제는 일반적으로 ZIP 아카이브 파일로 제공됩니다. Flash Professional에서 실습 예제를 실행하려면

1. 원하는 폴더에 아카이브 파일의 압축을 풉니다.

2 Flash Professional에서 [파일] > [열기]를 선택합니다.

3 아카이브 파일의 압축을 푼 폴더로 이동합니다. 폴더에서 FLA 파일을 선택하고 [열기]를 클릭합니다.

4 예제를 테스트하려면 [컨트롤] > [동영상 테스트]를 선택합니다.

Flash Builder에서 ActionScript 3.0 예제 실행

예제 유형에 따라 다음 절차 중 하나를 사용하여 Flash Builder를 통해 예제를 실행합니다.

Flash Builder에서 코드 조각 예제 실행

Flash Builder에서 코드 조각 예제를 실행하려면

1 [File] > [New] > [Flex Project]를 선택하여 새 Flex 프로젝트를 만들거나, [File] > [New] > [MXML Application]을 선택하여 기존 Flex 프로젝트 내에서 새 MXML 응용 프로그램을 만듭니다. 프로젝트 또는 응용 프로그램의 이름을 알기 쉽게(예: ContextMenuExample) 지정합니다.

2 생성된 MXML 파일 내에 <mx:Script> 태그를 추가합니다.

3 <mx:Script>와 </mx:Script> 태그 사이에 코드 조각 예제의 내용을 붙여 넣습니다. MXML 파일을 저장합니다.

4 예제를 실행하려면 [Run] > 기본 MXML 파일의 메뉴 옵션 실행(예: [Run] > [Run ContextMenuExample])을 선택합니다.

Flash Builder에서 클래스 기반 예제 실행

Flash Builder에서 클래스 기반 예제를 실행하려면

1 [File] > [New] > [ActionScript Project]를 선택합니다.

2 기본 클래스의 이름(예: ContextMenuExample)을 [Project Name] 필드에 입력합니다. 다른 필드는 기본값을 그대로 사용합니다. 아니면, 특정 환경에 맞게 변경할 수도 있습니다. [Finish]를 클릭하여 프로젝트와 기본 ActionScript 파일을 만듭니다.

3 ActionScript 파일에서 생성된 내용을 지웁니다. 패키지 및 가져오기 설명을 포함하여 예제 코드를 ActionScript 파일에 붙여 넣고 파일을 저장합니다.

참고: flashx.textLayout.container.ContainerController 클래스 예제와 같은 일부 클래스 기반 예제는 패키지 선언(package flashx.textLayout.container.examples {)에 여러 수준을 포함합니다. 이러한 예제의 경우 파일을 패키지 선언과 일치하는 하위 폴더(flashx/textLayout/container/examples)에 저장하거나, 패키지 이름을 제거하여 ActionScript가 package { 만으로 시작하도록 하면 모든 위치에서 파일을 테스트할 수 있습니다.

4 예제를 실행하려면 [Run] > 기본 ActionScript 클래스 이름의 메뉴 옵션 실행(예: [Run] > [Run ContextMenuExample])을 선택합니다.

Flash Builder에서 실습 예제 실행

실습 예제는 일반적으로 ZIP 아카이브 파일로 제공됩니다. Flash Builder에서 실습 예제를 실행하려면

1 원하는 폴더에 아카이브 파일의 압축을 풉니다. 폴더의 이름을 알기 쉽게(예: ContextMenuExample) 지정합니다.

2 Flash Builder에서 [File] > [New] > [Flex Project]를 선택합니다. [Project Location] 섹션에서 [Browse]를 클릭하고 예제 파일이 포함된 폴더를 선택합니다. [Project Name] 필드에서 폴더 이름(예: ContextMenuExample)을 입력합니다. 다른 필드는 기본값을 그대로 사용합니다. 아니면, 특정 환경에 맞게 변경할 수도 있습니다. [Next]를 클릭하여 계속합니다.

3 [Output] 패널에서 [Next]를 클릭하여 기본값을 그대로 사용합니다.

4 [Source Paths] 패널에서 [Main Application File] 필드 옆에 있는 [Browse] 버튼을 클릭합니다. 예제 폴더에서 기본 MXML 예제 파일을 선택합니다. [Finish]를 클릭하여 프로젝트 파일을 만듭니다.

5 예제를 실행하려면 [Run] > 기본 MXML 파일의 메뉴 옵션 실행(예: [Run] > [Run ContextMenuExample])을 선택합니다.

모바일 장치에서 ActionScript 3.0 예제 실행

Flash Player 10.1을 지원하는 모바일 장치에서 ActionScript 3.0 코드 예제를 실행할 수도 있지만 일반적으로 코드 예제는 특정 클래스 및 메서드의 작동 방식을 학습하기 위해 실행됩니다. 그러한 경우 데스크톱 컴퓨터와 같은 비모바일 장치에서 예제를 실행하십시오. 데스크톱 컴퓨터에서는 추적 명령문 및 Flash Professional 또는 Flash Builder의 기타 디버깅 도구를 사용하여 코드 예제에 대한 이해도를 높일 수 있습니다.

모바일 장치에서 예제를 실행하려는 경우 파일을 장치 또는 웹 서버에 복사할 수 있습니다. 파일을 장치에 복사하고 브라우저에서 예제를 실행하려면 다음을 수행합니다.

1 1000페이지의 "Flash Professional에서 ActionScript 3.0 예제 실행" 또는 1001페이지의 "Flash Builder에서 ActionScript 3.0 예제 실행"의 지침에 따라 SWF 파일을 만듭니다. Flash Professional에서는 [컨트롤] > [동영상 테스트]를 선택하면 SWF 파일이 만들어집니다. Flash Builder에서는 Flash Builder 프로젝트를 실행, 디버깅 또는 작성하면 SWF 파일이 만들어집니다.

2 SWF 파일을 모바일 장치의 디렉토리에 복사합니다. 장치에 제공된 소프트웨어를 사용하여 파일을 복사합니다.

3 모바일 장치의 브라우저에 있는 주소 표시줄에 file://SWF 파일의 URL을 입력합니다. 예를 들어 file:://applications/myExample.swf를 입력합니다.

파일을 웹 서버에 복사하고 장치의 브라우저에서 예제를 실행하려면 다음을 수행합니다.

1 SWF 파일과 HTML 파일을 만듭니다. 먼저 1000페이지의 "Flash Professional에서 ActionScript 3.0 예제 실행" 또는 1001페이지의 "Flash Builder에서 ActionScript 3.0 예제 실행"의 지침을 따릅니다. Flash Professional에서는 [컨트롤] > [동영상 테스트]를 선택하면 SWF 파일만 만들어집니다. 두 파일을 모두 만들려면 먼저 [제작 설정] 대화 상자의 [형식] 탭에서 Flash와 HTML을 모두 선택해야 합니다. 그런 다음 [파일] > [제작]을 선택하여 HTML과 SWF 파일을 모두 만듭니다. Flash Builder에서는 Flash Builder 프로젝트를 실행, 디버깅 또는 작성할 때 SWF 파일과 HTML 파일이 모두 만들어집니다.

2 SWF 파일과 HTML 파일을 웹 서버에 있는 디렉토리에 복사합니다.

3 모바일 장치의 브라우저에 있는 주소 표시줄에 HTML 파일의 HTTP 주소를 입력합니다. 예를 들어 http://www.myWebServer/examples/myExample.html을 입력합니다.

모바일 장치에서 예제를 실행할 때는 먼저 다음과 같은 문제들을 고려해보십시오.

스테이지 크기

모바일 장치에서 예제를 실행할 때 사용하는 스테이지 크기는 비모바일 장치를 사용할 때보다 매우 작습니다. 많은 예제의 경우 특정한 스테이지 크기를 요구하지 않습니다. 그러나 SWF 파일을 만들 때는 장치에 적합한 스테이지 크기를 지정하십시오. 예를 들어 176 x 208 픽셀로 지정하십시오.

ActionScript 3.0 개발자 안내서에 있는 실습 예제의 목적은 여러 가지 ActionScript 3.0의 개념과 클래스를 설명하는 것입니다. 이러한 실습 예제의 사용자 인터페이스는 데스크톱 또는 랩톱 컴퓨터에서 올바르게 표시되고 작동할 수 있도록 설계되었습니다. 예제가 모바일 장치에서 작동하더라도 스테이지 크기 및 사용자 인터페이스 설계는 작은 화면에 적합하지 않습니다. Adobe에서는 컴퓨터에서 실습 예제를 실행하여 ActionScript를 학습한 후에 모바일 응용 프로그램에서 적절한 코드 조각을 사용할 것을 권장합니다.

추적 명령문 대신 텍스트 필드

모바일 장치에서 예제를 실행할 때는 예제의 추적 명령문에서 출력을 확인할 수 없습니다. 출력을 보려면 TextField 클래스의 인스턴스를 만듭니다. 그런 다음 추적 명령문에서 텍스트를 텍스트 필드의 text 속성에 추가합니다.

다음 함수를 사용하여 추적에 사용할 텍스트 필드를 설정할 수 있습니다.

```
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;
}
```

예를 들어 이 함수를 문서 클래스에 전용 함수로 추가합니다. 그런 다음 문서 클래스의 다른 메서드에서 다음과 같이 코드를 사용하여 데이터를 추적합니다.

```
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("");
```

appendText() 메서드에서는 매개 변수로 하나의 값만 사용합니다. 이 값은 문자열입니다(문자열 인스턴스 또는 문자열 리터럴). 문자열이 아닌 변수 값을 출력하려면 먼저 값을 문자열로 변환합니다. 가장 쉬운 변환 방법은 객체의 toString() 메서드를 호출하는 것입니다.

```
var albumYear:int = 1999;
traceField.appendText("albumYear = ");
traceField.appendText(albumYear.toString());
```

텍스트 크기

많은 예제에서 개념 설명을 돕기 위해 텍스트 필드가 사용됩니다. 때때로 텍스트 필드에서 텍스트 크기를 조정할 경우 모바일 장치에서의 가독성을 향상시키는 데 도움이 될 수 있습니다. 예를 들어 예제에 myTextField라는 텍스트 필드 인스턴스가 사용될 경우 다음과 같은 코드로 텍스트의 크기를 변경합니다.

```
// Use a text size that works well on the device.
var myFormat:TextFormat = new TextFormat();
myFormat.size = 18;
myTextField.defaultTextFormat = myFormat
```

사용자 입력 캡처

모바일 운영 체제 및 브라우저에서는 SWF 내용에서 수신되지 않는 일부 사용자 입력 이벤트가 캡처됩니다. 특정 동작은 운영 체제 및 브라우저에 따라 다르지만, 예제를 모바일 장치에서 실행할 때 예기치 못한 동작이 발생할 수도 있습니다. 자세한 내용은 510페이지의 “KeyboardEvent 우선 순위”를 참조하십시오.

또한 많은 예제의 사용자 인터페이스가 데스크톱 또는 랩톱 컴퓨터용으로 설계됩니다. 예를 들어 ActionScript 3.0 개발자 안내서에 있는 대부분의 실습 예제는 데스크톱에서 표시하는 데 적합합니다. 따라서 모바일 장치의 화면에서는 전체 스테이지가 표시되지 않는 경우가 있습니다. 브라우저에 따라 브라우저 내용에 대한 패닝 기능이 지원되지 않을 수 있습니다. 또한 스크롤 또는 패닝 이벤트를 포착 및 처리하도록 예제가 설계되지 않습니다. 따라서 일부 예제의 사용자 인터페이스는 작은 화면에서 실행하는 데 적합하지 않을 수 있습니다. Adobe에서는 컴퓨터에서 예제를 실행하여 ActionScript를 학습한 후에 모바일 응용 프로그램에서 적절한 코드 조각을 사용할 것을 권장합니다.

자세한 내용은 161페이지의 "표시 객체 패닝 및 스크롤"을 참조하십시오.

포커스 처리

일부 예제에서는 사용자가 필드에 포커스를 두어야 합니다. 예를 들어 필드에 포커스를 두어야 텍스트를 입력하거나 버튼을 선택할 수 있습니다. 필드에 포커스를 두려면 스타일러스 또는 손가락과 같이 모바일 장치의 포인터 장치를 사용하십시오. 또는 모바일 장치의 탐색 키를 사용하여 필드에 포커스를 둡니다. 포커스가 있는 버튼을 선택하려면 컴퓨터에서 Enter 키를 사용하는 것처럼 모바일 장치의 Select 키를 사용합니다. 일부 장치에서는 버튼을 두 번 두드리면 버튼이 선택됩니다.

포커스에 대한 자세한 내용은 506페이지의 "포커스 관리"를 참조하십시오.

마우스 이벤트 처리

많은 예제에서 마우스 이벤트가 수신됩니다. 예를 들어 컴퓨터에서 이러한 마우스 이벤트는 사용자가 마우스를 표시 객체 위로 이동하거나 표시 객체에서 마우스 버튼을 클릭할 때 발생할 수 있습니다. 모바일 장치에서는 스타일러스 또는 손가락과 같은 포인터 장치를 사용하여 발생하는 이벤트를 터치 이벤트라고 부릅니다. Flash Player 10.1은 이러한 터치 이벤트를 마우스 이벤트로 매핑합니다. 이러한 매핑을 통해 Flash Player 10.1 이전에 개발된 SWF 내용도 계속해서 작동할 수 있습니다. 따라서 포인터 장치를 사용하여 표시 객체를 선택하거나 드래그할 때에도 예제가 작동합니다.

성능

모바일 장치는 데스크톱 장치보다 처리 능력이 낮습니다. 일부 CPU 사용량이 높은 예제는 모바일 장치에서 성능이 느려질 수 있습니다. 예를 들어 210페이지의 "드로잉 API 예제: Algorithmic Visual Generator"의 예제는 계산 작업이 많고 모든 프레임에 들어갈 때마다 그리기를 수행합니다. 이 예제를 컴퓨터에서 실행하면 다양한 그리기 API가 표시됩니다. 하지만 이 예제는 성능상의 제한으로 인해 일부 모바일 장치에 적합하지 않습니다.

휴대 장치의 성능에 대한 자세한 내용은 **Flash Platform**의 성능 최적화를 참조하십시오.

유용한 방법

예제에서는 모바일 장치에 맞는 응용 프로그램 개발에 대한 유용한 방법들이 고려되지 않습니다. 모바일 장치는 메모리 및 처리 능력에 대한 제한으로 인해 특별한 주의가 필요합니다. 마찬가지로, 작은 화면용 사용자 인터페이스는 데스크톱 디스플레이와 그 요구 사항이 다릅니다. 휴대 장치용 응용 프로그램 개발에 대한 자세한 내용은 **Flash Platform**의 성능 최적화를 참조하십시오.

66장: 로컬 데이터베이스의 SQL 지원

Adobe AIR에는 오픈 소스 SQLite 데이터베이스 시스템을 사용하여 다양한 표준 SQL 기능과 함께 로컬 SQL 데이터베이스를 지원하는 SQL 데이터베이스 엔진이 포함되어 있습니다. 런타임은 파일 시스템에서 데이터베이스 데이터가 저장되는 방법이나 위치를 지정하지 않습니다. 각 데이터베이스는 단일 파일에 완전히 저장됩니다. 개발자가 파일 시스템 내의 데이터베이스 파일 저장 위치를 지정할 수 있으며, 단일 AIR 응용 프로그램에서 하나 이상의 독립적인 데이터베이스(예: 독립적인 여러 데이터베이스 파일)에 액세스할 수 있습니다. 이 문서에서는 Adobe AIR 로컬 SQL 데이터베이스에 대한 SQL 구문 및 데이터 유형 지원에 대해 설명합니다. 이 문서는 포괄적인 SQL 참조가 아니며, Adobe AIR에서 지원하는 SQL 언어에 대해 자세히 설명합니다. 런타임에서는 대부분의 SQL-92 표준 SQL 언어를 지원합니다. 수많은 SQL 학습용 참고 자료, 웹 사이트, 서적 및 교육 자료가 제공되고 있으므로 이 문서는 포괄적인 SQL 참조 또는 자습서로 사용하도록 작성되지 않았습니다. 대신 이 문서는 AIR 지원 SQL 구문 및 SQL-92와 지원되는 SQL 언어 사이의 차이점에 대해 중점적으로 설명합니다.

SQL 명령문 정의 규칙

이 문서의 명령문 정의에는 다음과 같은 규칙이 사용되었습니다.

- 텍스트 대/소문자
  - 대문자 - 리터럴 SQL 키워드는 모두 대문자입니다.
  - 소문자 - 자리 표시자 항목 또는 절 이름은 모두 소문자입니다.
- 정의 문자
  - ::= 절 또는 명령문 정의를 나타냅니다.
- 그룹화 및 대체 문자
  - | 파이프 문자는 대체 옵션을 구분하는 데 사용되었으며 "또는"으로 읽을 수 있습니다.
  - [] 대괄호로 묶은 항목은 선택적 항목이며 대괄호에는 단일 항목이나 대체 항목 집합이 들어 있을 수 있습니다.
  - () 대체 항목 집합(파이프 문자로 구분된 항목 집합)을 묶는 괄호는 필수 항목 그룹(단일 필수 항목에 사용 가능한 값인 항목 집합)을 나타냅니다.
- 한정 기호
  - + 괄호로 묶은 항목 뒤에 더하기 문자가 있으면 이전 항목이 한 번 이상 나타날 수 있다는 의미입니다.
  - * 대괄호로 묶은 항목 뒤에 별표 문자가 있으면 대괄호로 묶은 이전 항목이 0번 이상 나타날 수 있다는 의미입니다.
- 리터럴 문자
  - * 열 이름에 사용되거나 함수 이름 뒤의 괄호 사이에 사용된 별표 문자는 "0번 이상" 한정 기호가 아닌 리터럴 별표 문자를 나타냅니다.
  - . 마침표 문자는 리터럴 마침표를 나타냅니다.
  - , 쉼표 문자는 리터럴 쉼표를 나타냅니다.
  - () 단일 절이나 항목을 묶는 괄호 쌍은 괄호가 필수적인 리터럴 괄호 문자임을 나타냅니다.
  - 별도로 명시되지 않은 경우 기타 문자는 해당 리터럴 문자를 나타냅니다.

지원되는 SQL 구문

다음은 Adobe AIR SQL 데이터베이스 엔진에서 지원되는 SQL 구문 목록입니다. 이러한 목록은 다양한 명령문 및 절 유형, 표현식, 내장 함수 및 연산자에 대한 설명으로 구분되어 있습니다. 여기에서 다루는 항목은 다음과 같습니다.

- 일반적인 SQL 구문
- 데이터 조작 명령문(SELECT, INSERT, UPDATE, DELETE)
- 데이터 정의 명령문(테이블, 인덱스, 뷰, 트리거에 대한 CREATE, ALTER 및 DROP 문)
- 특수 명령문 및 절
- 내장 함수(집계, 스칼라 및 날짜/시간 형식 함수)
- 연산자
- 매개 변수
- 지원되지 않는 SQL 기능
- 추가 SQL 기능

일반적인 SQL 구문

다양한 명령문 및 표현식에 대한 구체적인 구문 이외에 SQL 구문의 일반적인 규칙은 다음과 같습니다.

대/소문자 구분 객체 이름을 포함하여 SQL 명령문은 대/소문자를 구분하지 않습니다. 그러나 SQL 명령문의 SQL 키워드는 대문자로 작성하는 경우가 많으며, 이 문서에서는 이러한 규칙을 따릅니다. SQL 구문에서는 대/소문자가 구분되지 않지만 SQL의 리터럴 텍스트 값에서는 대/소문자가 구분되며, 비교 및 정렬 작업에서도 열 또는 작업에 정의된 데이터 정렬 시퀀스에 따라 대/소문자가 구분될 수 있습니다. 자세한 내용은 COLLATE를 참조하십시오.

공백 빈 칸, 탭, 개행 문자 등의 공백 문자를 사용하여 SQL 명령문의 개별 단어를 분리해야 합니다. 그러나 단어와 심볼 사이에서는 공백 문자를 생략할 수 있습니다. SQL 명령문에서 공백 문자의 종류와 개수는 중요하지 않습니다. 들여쓰기 및 줄 바꿈 등의 공백을 사용하여 SQL 명령문을 읽기 쉽도록 서식을 지정할 수 있으며, 이때 명령문의 의미에는 영향이 없습니다.

데이터 조작 명령문

데이터 조작 명령문은 가장 자주 사용되는 SQL 명령문입니다. 이러한 명령문을 사용하여 데이터베이스 테이블에서 데이터를 검색, 추가, 수정 및 제거할 수 있습니다. SELECT, INSERT, UPDATE, DELETE 등의 데이터 조작 명령문이 지원됩니다.

SELECT

SELECT 문은 데이터베이스를 쿼리하는 데 사용됩니다. SELECT의 결과는 0개 이상의 데이터 행이며 각 행에 고정된 개수의 열이 있습니다. 결과의 열 수는 SELECT와 선택적 키워드인 FROM 사이에서 결과 열 이름이나 표현식 목록으로 지정됩니다.

```
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
```

임의의 표현식을 결과로 사용할 수 있습니다. 결과 표현식이 *이면 모든 테이블의 모든 열이 해당 표현식 하나로 대체됩니다. 표현식으로 테이블 이름 뒤에 .*를 사용하면 결과는 해당 테이블 하나의 모든 열입니다.

DISTINCT 키워드를 사용하면 결과 행 중 서로 다른 행만 반환됩니다. NULL 값은 서로 같은 것으로 간주됩니다. 기본 비헤이비어는 모든 결과 행을 반환하는 것이며, ALL 키워드를 사용하여 이를 명시적으로 지정할 수 있습니다.

쿼리는 FROM 키워드 뒤에 지정된 하나 이상의 테이블에 대해 실행됩니다. 여러 테이블 이름을 쉼표로 구분하여 지정하면 쿼리에서 여러 테이블의 크로스 조인을 사용합니다. JOIN 구문을 사용하여 테이블을 조인하는 방법을 지정할 수도 있습니다. 지원되는 유일한 외부 조인 유형은 LEFT OUTER JOIN입니다. join-args의 ON 절 표현식은 부울 값으로 확인되어야 합니다. 괄호로 묶은 하위 쿼리를 FROM 절에서 테이블로 사용할 수 있습니다. 전체 FROM 절을 생략할 수도 있으며 이 경우 결과는 결과 표현식 목록의 값으로 구성된 단일 행입니다.

WHERE 절은 쿼리에서 검색하는 행 수를 제한하는 데 사용됩니다. WHERE 절 표현식은 부울 값으로 확인되어야 합니다. WHERE 절 필터링이 모든 그룹화 이전에 수행되므로 WHERE 절 표현식에는 집계 함수가 포함될 수 없습니다.

GROUP BY 절을 사용하면 하나 이상의 결과 행이 단일 출력 행으로 결합됩니다. GROUP BY 절은 결과에 집계 함수가 들어 있는 경우에 특히 유용합니다. GROUP BY 절의 표현식은 SELECT 표현식 목록에 나타나는 표현식이 아니어도 됩니다.

HAVING 절은 명령문에서 반환하는 행을 제한한다는 점에서 WHERE와 비슷합니다. 그러나 HAVING 절은 GROUP BY 절에 지정된 그룹화가 모두 수행된 후에 적용됩니다. 따라서 HAVING 표현식은 집계 함수가 포함된 값을 참조할 수 있습니다. HAVING 절 표현식은 SELECT 목록에 나타나지 않아도 됩니다. WHERE 표현식과 마찬가지로 HAVING 표현식도 부울 값으로 확인되어야 합니다.

ORDER BY 절을 사용하면 출력 행이 정렬됩니다. ORDER BY 절의 sort-expr-list 인수는 정렬 키로 사용되는 표현식 목록입니다. 단순 SELECT에서는 표현식이 결과에 포함되지 않을 수도 있지만 복합 SELECT(compound-op 연산자 중 하나를 사용하는 SELECT)에서는 각 정렬 표현식이 이러한 결과 열 중 하나와 정확히 일치해야 합니다. 각 정렬 표현식 뒤에 COLLATE 키워드와 텍스트 정렬에 사용되는 데이터 정렬 함수의 이름 및/또는 정렬 순서(오름차순 또는 내림차순)를 지정하는 ASC 또는 DESC 키워드로 구성된 sort-order 절을 추가할 수 있습니다. sort-order를 생략하고 기본값(오름차순)을 사용할 수도 있습니다. COLLATE 절의 정의 및 데이터 정렬 함수에 대한 자세한 내용은 COLLATE를 참조하십시오.

LIMIT 절은 결과에서 반환되는 행 수의 상한을 설정합니다. LIMIT가 음수이면 상한이 없는 것입니다. LIMIT 뒤에 오는 선택적 OFFSET은 결과 집합의 시작 부분에서 건너뛸 행 수를 지정합니다. 복합 SELECT 쿼리에서 LIMIT 절은 마지막 SELECT 문 뒤에만 올 수 있으며 전체 쿼리에 제한이 적용됩니다. LIMIT 절에 OFFSET 키워드를 사용한 경우 첫 번째 정수는 제한으로, 두 번째 정수는 오프셋으로 사용됩니다. OFFSET 키워드 대신 쉼표를 사용한 경우 첫 번째 숫자는 오프셋으로, 두 번째 숫자는 제한으로 사용됩니다. 이는 모순처럼 보이지만 레거시 SQL 데이터베이스 시스템과의 호환성을 최대화하기 위한 의도적인 조치입니다.

복합 SELECT는 UNION, UNION ALL, INTERSECT 또는 EXCEPT 연산자 중 하나로 둘 이상의 단순 SELECT 문을 연결하여 구성됩니다. 복합 SELECT를 구성하는 모든 SELECT 문은 동일한 개수의 결과 열을 지정해야 합니다. ORDER BY 절은 마지막 SELECT 문 뒤에 하나만 사용할 수 있으며 단일 LIMIT 절(지정된 경우) 앞에 와야 합니다. UNION 및 UNION ALL 연산자는 앞뒤 SELECT 문의 결과를 단일 테이블로 결합합니다. UNION에서는 모든 결과 행이 고유하지만 UNION ALL에서는 중복이 나타날 수 있다는 차이점이 있습니다. INTERSECT 연산자는 앞뒤 SELECT 문에서 나타난 결과의 교집합을 취합니다. EXCEPT는 앞에 있는 SELECT의 결과에서 뒤에 있는 SELECT의 결과를 제거합니다. 세 개 이상의 SELECT 문을 복합 명령문으로 연결하는 경우 앞에서 뒤로 그룹화됩니다.

허용되는 표현식의 정의는 표현식을 참조하십시오.

AIR 2.5부터 SQL CAST 연산자는 읽기 작업으로 BLOB 데이터를 ActionScript ByteArray 객체로 변환할 때 지원됩니다. 예를 들어 다음 코드에서는 AMF 형식으로 저장되지 않은 원시 데이터를 읽고 ByteArray 객체로 저장합니다.

```
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

테이블을 데이터로 채우는 데 사용되는 INSERT 문에는 두 가지 기본 형태가 있습니다.

```
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
```

VALUES 키워드가 사용된 첫 번째 형태는 기존 테이블에 새 행을 하나 만듭니다. column-list를 지정하지 않은 경우 값의 개수는 테이블의 열 수와 같아야 합니다. column-list를 지정한 경우에는 값의 개수가 지정된 열 수와 일치해야 합니다. 열 목록에 나타나지 않는 테이블의 열은 테이블이 만들어질 때 정의된 기본값으로 채워지며, 기본값이 정의되지 않은 경우 NULL로 채워집니다.

INSERT 문의 두 번째 형태는 SELECT 문에서 데이터를 가져옵니다. SELECT의 결과에 있는 열 수는 테이블의 열 수(column-list를 지정하지 않은 경우) 또는 column-list에 지정된 열 수와 정확히 일치해야 합니다. SELECT 결과에 있는 모든 행에 대해 테이블에 새 항목이 만들어집니다. SELECT는 단순 명령문일 수도 있고 복합 명령문일 수도 있습니다. 사용 가능한 SELECT 문의 정의는 SELECT를 참조하십시오.

선택적 conflict-algorithm을 사용하면 이 명령 하나를 실행할 때 사용할 대체 제약 조건 충돌 해결 알고리즘을 지정할 수 있습니다. 충돌 알고리즘에 대한 설명 및 정의는 1014페이지의 "특수 명령문 및 절"을 참조하십시오.

명령문의 두 가지 REPLACE INTO 형태는 REPLACE 충돌 알고리즘과 함께 표준 INSERT [OR conflict-algorithm] 형태(예: INSERT OR REPLACE... 형태)를 사용하는 것과 같습니다.

명령문의 두 가지 REPLACE INTO 형태는 REPLACE 충돌 알고리즘과 함께 표준 INSERT [OR conflict-algorithm] 형태(예: INSERT OR REPLACE... 형태)를 사용하는 것과 같습니다.

UPDATE

update 명령은 테이블의 기존 레코드를 변경합니다.

```
sql-statement  ::=  UPDATE [database-name.] table-name SET column1=value1, column2=value2,... [WHERE expr]
```

이 명령은 UPDATE 키워드와 레코드를 업데이트할 테이블의 이름으로 구성됩니다. SET 키워드 다음에 열의 이름과 해당 열이 변경될 값을 쉼표로 구분해 나열합니다. WHERE 절 표현식은 레코드가 업데이트될 행 또는 행들을 제공합니다.

DELETE

DELETE 명령은 테이블에서 레코드를 제거하는 데 사용됩니다.

```
sql-statement  ::=  DELETE FROM [database-name.] table-name [WHERE expr]
```

이 명령은 DELETE FROM 키워드와 레코드를 제거할 테이블의 이름으로 구성됩니다.

WHERE 절이 없으면 테이블의 모든 행이 제거됩니다. WHERE 절을 제공한 경우에는 표현식과 일치하는 행만 제거됩니다. WHERE 절 표현식은 부울 값으로 확인되어야 합니다. 허용되는 표현식의 정의는 표현식을 참조하십시오.

데이터 정의 명령문

데이터 정의 명령문은 테이블, 뷰, 인덱스 및 트리거 등의 데이터베이스 객체를 만들고 수정하고 제거하는 데 사용됩니다. 다음과 같은 데이터 정의 명령문이 지원됩니다.

- 테이블:
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
- 인덱스:
  - CREATE INDEX
  - DROP INDEX
- 뷰:
  - CREATE VIEWS
  - DROP VIEWS
- 트리거:
  - CREATE TRIGGERS
  - DROP TRIGGERS

CREATE TABLE

CREATE TABLE 문은 CREATE TABLE 키워드, 새 테이블 이름, 괄호로 묶은 열 정의 및 제약 조건 목록으로 구성됩니다. 테이블 이름은 식별자 또는 문자열일 수 있습니다.

```
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]*
```

각 열 정의에는 열 이름, 해당 열의 데이터 유형, 하나 이상의 선택적 열 제약 조건이 차례로 나옵니다. 열의 데이터 유형은 해당 열에 저장할 수 있는 데이터를 제한합니다. 열에 다른 데이터 유형의 값을 저장하려고 하면 런타임에서 가능한 경우 값을 적절한 유형으로 변환합니다. 변환할 수 없으면 오류가 발생합니다. 자세한 내용은 "데이터 유형 지원" 단원을 참조하십시오.

NOT NULL 열 제약 조건은 열에 NULL 값이 포함될 수 없음을 나타냅니다.

UNIQUE 제약 조건을 사용하면 지정된 하나 이상의 열에 인덱스가 만들어집니다. 이 인덱스에는 고유 키가 있어야 하며, 두 행에서 지정된 하나 이상의 열에는 중복된 값이나 값 조합이 포함될 수 없습니다. CREATE TABLE 문에는 여러 UNIQUE 제약 조건이 있을 수 있습니다. 여러 열의 정의에 UNIQUE 제약 조건이 있을 수도 있고 여러 테이블 수준 UNIQUE 제약 조건이 있을 수도 있습니다.

CHECK 제약 조건은 true로 평가되어야 행의 데이터를 삽입하거나 업데이트할 수 있는 표현식을 정의합니다. CHECK 표현식은 부울 값으로 확인되어야 합니다.

열 정의의 COLLATE 절은 열의 텍스트 항목을 비교할 때 사용할 텍스트 데이터 정렬 함수를 지정합니다. 기본적으로 BINARY 데이터 정렬 함수가 사용됩니다. COLLATE 절 및 데이터 정렬 함수에 대한 자세한 내용은 COLLATE를 참조하십시오.

DEFAULT 제약 조건은 INSERT를 수행할 때 사용할 기본값을 지정합니다. 이 값은 NULL, 문자열 상수 또는 숫자일 수 있습니다. 기본값은 대/소문자를 구분하지 않는 특수 키워드인 CURRENT_TIME, CURRENT_DATE 또는 CURRENT_TIMESTAMP 중 하나일 수도 있습니다. 값이 NULL, 문자열 상수 또는 숫자이면 INSERT 문에 열 값이 지정되지 않은 경우 해당 값이 열에 그대로 삽입됩니다. 값이 CURRENT_TIME, CURRENT_DATE 또는 CURRENT_TIMESTAMP이면 현재 UTC 날짜 및/또는 시간이 열에 삽입됩니다. CURRENT_TIME의 서식은 HH:MM:SS입니다. CURRENT_DATE의 서식은 YYYY-MM-DD입니다. CURRENT_TIMESTAMP의 서식은 YYYY-MM-DD HH:MM:SS입니다.

PRIMARY KEY를 지정하면 일반적으로 하나 이상의 해당 열에 단순히 UNIQUE 인덱스가 만들어집니다. 그러나 데이터 유형이 INTEGER(또는 int와 같은 동의어 중 하나)인 단일 열에 PRIMARY KEY 제약 조건을 지정하면 해당 열이 데이터베이스에서 테이블의 실제 기본 키로 사용됩니다. 즉, 해당 열은 고유 정수 값만 보유할 수 있습니다. 여러 SQLite 구현에서는 열 유형 INTEGER를 사용해야만 열이 내부 기본 키 역할을 하지만, Adobe AIR에서는 INTEGER의 동의어(예: int)도 해당 비헤이비어를 지정합니다.

테이블에 INTEGER PRIMARY KEY 열이 없으면 행이 삽입될 때 정수 키가 자동으로 생성됩니다. 특수 이름인 ROWID, OID 또는 _ROWID_ 중 하나를 사용하면 언제든지 행의 기본 키에 액세스할 수 있습니다. INTEGER PRIMARY KEY로 명시적으로 선언되었는지 또는 값이 내부적으로 생성되었는지 여부에 관계없이 이러한 이름을 사용할 수 있습니다. 그러나 테이블에 명시적인 INTEGER PRIMARY KEY가 있는 경우 결과 데이터의 열 이름은 특수 이름이 아니라 실제 열 이름입니다.

INTEGER PRIMARY KEY 열에 AUTOINCREMENT 키워드가 포함될 수도 있습니다. AUTOINCREMENT 키워드를 사용하면 열의 명시적인 값을 지정하지 않는 INSERT 문이 실행될 때 데이터베이스에서 순차적으로 증가하는 정수 키를 자동으로 생성하여 INTEGER PRIMARY KEY 열에 삽입합니다.

CREATE TABLE 문에는 PRIMARY KEY 제약 조건이 하나만 있을 수 있습니다. 이 제약 조건은 열 정의에 포함될 수도 있고 단일 테이블 수준 PRIMARY KEY 제약 조건일 수도 있습니다. 기본 키 열은 암시적으로 NOT NULL입니다.

제약 조건 뒤에 선택적 conflict-clause를 사용하면 해당 제약 조건의 대체 기본 제약 조건 충돌 해결 알고리즘을 지정할 수 있습니다. 기본값은 ABORT입니다. 같은 테이블 내의 여러 제약 조건에 서로 다른 기본 충돌 해결 알고리즘을 지정할 수 있습니다. INSERT 또는 UPDATE 문이 다른 충돌 해결 알고리즘을 지정하는 경우 해당 알고리즘이 CREATE TABLE 문에 지정된 알고리즘 대신 사용됩니다. 자세한 내용은 1014페이지의 "특수 명령문 및 절"의 ON CONFLICT 단원을 참조하십시오.

FOREIGN KEY 제약 조건 등의 추가 제약 조건은 오류를 발생시키지 않지만 런타임에서 무시됩니다.

CREATE와 TABLE 사이에 TEMP 또는 TEMPORARY 키워드를 사용하면 작성되는 테이블을 같은 데이터베이스 연결(SQLConnection 인스턴스)에서만 볼 수 있습니다. 데이터베이스 연결이 닫히면 테이블이 자동으로 삭제됩니다. 임시 테이블에 만들어진 인덱스도 임시적입니다. 임시 테이블과 인덱스는 주 데이터베이스 파일과 다른 별도의 파일에 저장됩니다.

선택적 접두어인 database-name을 지정하면 명명된 데이터베이스(지정된 데이터베이스 이름으로 attach() 메서드를 호출하여 SQLConnection 인스턴스에 연결된 데이터베이스)에 테이블이 만들어집니다. database-name 접두어가 temp가 아닌 경우 database-name 접두어와 TEMP 키워드를 모두 지정하면 오류가 발생합니다. 데이터베이스 이름이 지정되지 않았고 TEMP 키워드가 없으면 주 데이터베이스(open() 또는 openAsync()method 메서드를 사용하여 SQLConnection 인스턴스에 연결된 데이터베이스)에 테이블이 만들어집니다.

테이블의 열 수나 제약 조건 수에는 임의적인 제한이 없습니다. 행의 데이터 양에도 임의적인 제한이 없습니다.

CREATE TABLE AS 형태는 테이블을 쿼리의 결과 집합으로 정의합니다. 테이블 열의 이름은 결과 열의 이름입니다.

선택적 절인 IF NOT EXISTS가 있고 이름이 동일한 다른 테이블이 이미 있으면 데이터베이스에서 CREATE TABLE 명령을 무시합니다.

DROP TABLE 문을 사용하여 테이블을 제거할 수 있고, ALTER TABLE 문을 사용하여 제한적인 변경을 수행할 수 있습니다.

ALTER TABLE

ALTER TABLE 명령을 사용하여 기존 테이블의 이름을 바꾸거나 새 열을 추가할 수 있습니다. 테이블에서 열을 제거할 수는 없습니다.

```
sql-statement ::= ALTER TABLE [database-name.] table-name alteration
alteration    ::= RENAME TO new-table-name
alteration    ::= ADD [COLUMN] column-def
```

RENAME TO 구문은 [database-name.] table-name으로 식별된 테이블 이름을 new-table-name으로 바꾸는 데 사용됩니다. 이 명령을 사용하여 연결된 데이터베이스 간에 테이블을 이동할 수는 없으며, 같은 데이터베이스 내에서만 테이블의 이름을 바꿀 수 있습니다.

이름을 바꾸는 테이블에 트리거나 인덱스가 있으면 이름을 바꾼 후에도 트리거나 인덱스가 테이블에 연결된 상태로 유지됩니다. 그러나 이름을 바꾸는 테이블을 참조하는 트리거에서 실행하는 뷰 정의나 명령문은 새 테이블 이름을 사용하도록 자동으로 수정되지 않습니다. 이름을 바꾼 테이블에 뷰나 트리거가 연결되어 있는 경우 뷰나 트리거를 수동으로 삭제한 다음 새 테이블 이름을 사용하여 다시 만들어야 합니다.

ADD [COLUMN] 구문은 기존 테이블에 새 열을 추가하는 데 사용됩니다. 새 열은 항상 기존 열 목록의 끝에 추가됩니다. column-def 절은 CREATE TABLE 문에 허용되는 모든 형태를 취할 수 있지만 다음과 같은 제한 사항이 있습니다.

- 열에 PRIMARY KEY 또는 UNIQUE 제약 조건이 없어야 합니다.
- 열의 기본값이 CURRENT_TIME, CURRENT_DATE 또는 CURRENT_TIMESTAMP가 아니어야 합니다.
- NOT NULL 제약 조건이 지정된 경우 열의 기본값은 NULL이 아니어야 합니다.

ALTER TABLE 문의 실행 시간은 테이블의 데이터 양과 관계가 없습니다.

DROP TABLE

DROP TABLE 문은 CREATE TABLE 문으로 추가된 테이블을 제거합니다. 지정된 table-name의 테이블이 삭제됩니다. 이 테이블은 데이터베이스와 디스크 파일에서 완전히 제거되며 복구할 수 없습니다. 테이블에 연결된 모든 인덱스도 삭제됩니다.

```
sql-statement  ::=  DROP TABLE [IF EXISTS] [database-name.] table-name
```

기본적으로 DROP TABLE 문은 데이터베이스 파일의 크기를 줄이지 않습니다. 빈 공간이 데이터베이스에 유지되며 이후 INSERT 작업에 사용됩니다. 데이터베이스의 여유 공간을 제거하려면 SQLConnection.clean() 메서드를 사용합니다. 초기에 데이터베이스를 만들 때 autoClean 매개 변수를 true로 설정하면 공간이 자동으로 해제됩니다.

선택적 절인 IF EXISTS를 사용하면 일반적으로 해당 테이블이 없을 때 발생하는 오류가 숨겨집니다.

CREATE INDEX

CREATE INDEX 명령은 CREATE INDEX 키워드, 새 인덱스의 이름, ON 키워드, 이전에 만든 인덱싱할 테이블의 이름, 해당 값을 인덱스 키로 사용할 테이블 열 이름을 괄호로 묶은 목록으로 구성됩니다.

```
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]
```

각 열 이름 뒤에 정렬 순서를 지정하는 ASC 또는 DESC 키워드를 사용할 수 있지만 런타임은 지정된 정렬 순서를 무시하고 항상 오름차순으로 정렬합니다.

각 열 이름 뒤에 오는 COLLATE 절은 해당 열의 텍스트 값에 사용되는 데이터 정렬 시퀀스를 정의합니다. 기본 데이터 정렬 시퀀스는 CREATE TABLE 문에서 해당 열에 정의된 데이터 정렬 시퀀스입니다. 데이터 정렬 시퀀스를 지정하지 않은 경우 BINARY 데이터 정렬 시퀀스가 사용됩니다. COLLATE 절의 정의 및 데이터 정렬 함수에 대한 자세한 내용은 COLLATE를 참조하십시오.

단일 테이블에 연결할 수 있는 인덱스 수에는 임의적인 제한이 없습니다. 인덱스의 열 수에도 제한이 없습니다.

DROP INDEX

DROP INDEX 문은 CREATE INDEX 문으로 추가된 인덱스를 제거합니다. 지정된 인덱스는 데이터베이스 파일에서 완전히 제거됩니다. 인덱스를 복구하려면 적절한 CREATE INDEX 명령을 다시 입력해야 합니다.

```
sql-statement ::= DROP INDEX [IF EXISTS] [database-name.] index-name
```

기본적으로 DROP INDEX 문은 데이터베이스 파일의 크기를 줄이지 않습니다. 빈 공간이 데이터베이스에 유지되며 이후 INSERT 작업에 사용됩니다. 데이터베이스의 여유 공간을 제거하려면 SQLConnection.clean() 메서드를 사용합니다. 초기에 데이터베이스를 만들 때 autoClean 매개 변수를 true로 설정하면 공간이 자동으로 해제됩니다.

CREATE VIEW

CREATE VIEW 명령은 미리 정의된 SELECT 문에 이름을 할당합니다. 그런 다음 다른 SELECT 문의 FROM 절에서 새 이름을 테이블 이름 대신 사용할 수 있습니다. 뷰는 복잡하고 자주 사용되는 데이터 집합을 다른 작업에 사용할 수 있는 구조체로 결합하여 쿼리를 단순화하는 데 널리 사용됩니다.

```
sql-statement ::= CREATE [TEMP | TEMPORARY] VIEW [IF NOT EXISTS] [database-name.] view-name AS select-
statement
```

CREATE와 VIEW 사이에 TEMP 또는 TEMPORARY 키워드를 사용하면 데이터베이스를 연 SQLConnection 인스턴스에서만 작성된 뷰를 볼 수 있고, 데이터베이스가 닫히면 뷰가 자동으로 삭제됩니다.

[database-name]을 지정하면 해당 데이터베이스(지정된 name 인수와 함께 attach() 메서드를 사용하여 SQLConnection 인스턴스에 연결된 데이터베이스)에 뷰가 만들어집니다. [database-name]이 temp가 아닌 경우 [database-name]과 TEMP 키워드를 모두 지정하면 오류가 발생합니다. 데이터베이스 이름이 지정되지 않았고 TEMP 키워드가 없으면 주 데이터베이스(open() 또는 openAsync() 메서드를 사용하여 SQLConnection 인스턴스에 연결된 데이터베이스)에 뷰가 만들어집니다.

뷰는 읽기 전용입니다. 뷰에 DELETE, INSERT 또는 UPDATE 문을 사용하려면 연결된 유형(INSTEAD OF DELETE, INSTEAD OF INSERT, INSTEAD OF UPDATE)의 트리거를 최소한 하나 이상 정의해야 합니다. 뷰에 대한 트리거를 만드는 방법은 CREATE TRIGGER를 참조하십시오.

뷰를 데이터베이스에서 제거하려면 DROP VIEW 문을 사용합니다.

DROP VIEW

DROP VIEW 문은 CREATE VIEW 문으로 만든 뷰를 제거합니다.

```
sql-statement ::= DROP VIEW [IF EXISTS] view-name
```

지정된 view-name은 삭제할 뷰의 이름입니다. 뷰가 데이터베이스에서 제거되지만 내부 테이블의 데이터는 수정되지 않습니다.

CREATE TRIGGER

CREATE TRIGGER 문은 데이터베이스 스키마에 트리거를 추가하는 데 사용됩니다. 트리거는 지정된 데이터베이스 이벤트(database-event)가 발생할 때 자동으로 수행되는 데이터베이스 작업(trigger-action)입니다.

```
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]*
```

트리거는 특정 데이터베이스 테이블에서 DELETE, INSERT 또는 UPDATE 문이 실행되거나 지정된 하나 이상의 테이블 열이 UPDATE 문으로 인해 업데이트될 때 발생하도록 지정됩니다. 트리거는 TEMP 또는 TEMPORARY 키워드가 사용되지 않은 경우 영구적입니다. 이러한 키워드가 사용된 경우에는 SQLConnection 인스턴스의 주 데이터베이스 연결이 닫힐 때 트리거가 제거됩니다. 타이밍(BEFORE 또는 AFTER)이 지정되지 않은 경우 트리거의 기본값은 BEFORE입니다.

FOR EACH ROW 트리거만 지원되므로 FOR EACH ROW 텍스트는 선택적입니다. FOR EACH ROW 트리거를 사용하면 WHEN 절 표현식이 true로 평가되는 경우 트리거를 발생시킨 명령문에서 삽입, 업데이트 또는 삭제한 각 데이터베이스 행에 대해 trigger-step 명령문이 실행됩니다.

WHEN 절을 지정하면 WHEN 절이 true인 행에 대해서만 trigger-steps로 지정된 SQL 명령문이 실행됩니다. WHEN 절을 지정하지 않으면 모든 행에 대해 SQL 명령문이 실행됩니다.

트리거 본문(trigger-action 절) 내에서는 특수 테이블 이름인 OLD 및 NEW를 사용하여 영향을 받는 테이블의 변경 전 및 변경 후 값을 참조할 수 있습니다. OLD 및 NEW 테이블의 구조는 트리거가 작성된 테이블의 구조와 일치합니다. OLD 테이블에는 트리거하는 명령문에서 수정하거나 삭제한 모든 행이 해당 명령문의 작업 이전 상태로 들어 있습니다. NEW 테이블에는 트리거하는 명령문에서 수정하거나 작성한 모든 행이 해당 명령문의 작업 이후 상태로 들어 있습니다. WHEN 절과 trigger-step 명령문 모두에서 NEW.column-name 및 OLD.column-name 형태의 참조를 사용하여 삽입, 삭제 또는 업데이트되는 행의 값에 액세스할 수 있습니다. 여기에서 column-name은 트리거가 연결된 테이블에 있는 열의 이름입니다. OLD 및 NEW 테이블 참조를 사용할 수 있는지 여부는 트리거에서 처리하는 database-event의 유형에 따라 다릅니다.

- INSERT – NEW 참조를 사용할 수 있습니다.
- UPDATE – NEW 및 OLD 참조를 사용할 수 있습니다.
- DELETE – OLD 참조를 사용할 수 있습니다.

지정된 타이밍(BEFORE, AFTER 또는 INSTEAD OF)은 연결된 행의 삽입, 수정 또는 제거를 기준으로 trigger-step 명령문이 실행되는 시점을 결정합니다. trigger-step에서 UPDATE 또는 INSERT 문의 일부로 ON CONFLICT 절을 지정할 수 있습니다. 그러나 트리거를 발생시키는 명령문의 일부로 ON CONFLICT 절을 지정하면 해당 충돌 처리 정책이 대신 사용됩니다.

테이블 트리거뿐 아니라 INSTEAD OF 트리거도 뷰에 만들 수 있습니다. 뷰에 하나 이상의 INSTEAD OF INSERT, INSTEAD OF DELETE 또는 INSTEAD OF UPDATE 트리거를 정의하면 뷰에서 연결된 명령문 유형(INSERT, DELETE 또는 UPDATE)을 실행해도 오류로 간주되지 않습니다. 이러한 경우 뷰에서 INSERT, DELETE 또는 UPDATE를 실행하면 연결된 트리거가 발생합니다. 트리거가 INSTEAD OF 트리거이므로 트리거를 발생시킨 명령문에 의해 뷰의 내부 테이블이 수정되지 않습니다. 그러나 트리거를 사용하여 내부 테이블에서 수정 작업을 수행할 수는 있습니다.

INTEGER PRIMARY KEY 열이 있는 테이블에 트리거를 만들 때 고려해야 하는 중요한 문제가 있습니다. 트리거를 발생시키는 명령문으로 업데이트할 행의 INTEGER PRIMARY KEY 열이 BEFORE 트리거로 수정되면 업데이트가 발생하지 않습니다. 이 문제를 해결하려면 INTEGER PRIMARY KEY 열 대신 PRIMARY KEY 열이 있는 테이블을 만듭니다.

트리거를 제거하려면 DROP TRIGGER 문을 사용합니다. 테이블이나 뷰를 삭제하면 해당 테이블이나 뷰에 연결된 모든 트리거가 자동으로 함께 삭제됩니다.

RAISE() 함수

트리거의 trigger-step 명령문에서 특수 SQL 함수인 RAISE()를 사용할 수 있습니다. 이 함수의 구문은 다음과 같습니다.

```
raise-function  ::=  RAISE ( ABORT, error-message ) |
                     RAISE ( FAIL, error-message ) |
                     RAISE ( ROLLBACK, error-message ) |
                     RAISE ( IGNORE )
```

트리거 실행 도중 처음 세 가지 형태 중 하나를 호출하면 지정된 ON CONFLICT 처리 작업(ABORT, FAIL 또는 ROLLBACK)이 수행되고 현재 명령문의 실행이 종료됩니다. ROLLBACK은 명령문 실행 실패로 간주되므로 execute() 메서드를 실행 중인 SQLStatement 인스턴스에서 오류(SQLErrorEvent.ERROR) 이벤트를 전달합니다. 전달된 이벤트 객체의 error 속성에 있는 SQLError 객체의 details 속성은 RAISE() 함수에 지정된 error-message로 설정됩니다.

RAISE(IGNORE)를 호출하면 현재 트리거의 나머지 부분, 트리거를 실행한 명령문, 실행될 예정인 후속 트리거가 모두 실행되지 않습니다. 데이터베이스 변경 내용은 롤백되지 않습니다. 트리거를 실행한 명령문 자체가 트리거의 일부이면 해당 트리거 프로그램의 실행이 다음 단계의 시작 부분부터 재개됩니다. 충돌 해결 알고리즘에 대한 자세한 내용은 "ON CONFLICT(충돌 알고리즘)" 단원을 참조하십시오.

DROP TRIGGER

DROP TRIGGER 문은 CREATE TRIGGER 문으로 만든 트리거를 제거합니다.

```
sql-statement  ::=  DROP TRIGGER [IF EXISTS] [database-name.] trigger-name
```

트리거가 데이터베이스에서 삭제됩니다. 연결된 테이블이 삭제되면 트리거가 자동으로 삭제됩니다.

특수 명령문 및 절

이 단원에서는 런타임에서 제공하는 SQL을 확장하는 몇 가지 절과 여러 명령문, 주석 및 표현식에 사용할 수 있는 두 가지 언어 요소에 대해 설명합니다.

COLLATE

COLLATE 절은 SELECT, CREATE TABLE 및 CREATE INDEX 문에 사용되며 값을 비교하거나 정렬할 때 사용되는 비교 알고리즘을 지정합니다.

```
sql-statement   ::=  COLLATE collation-name
collation-name  ::=  BINARY | NOCASE
```

열의 기본 데이터 정렬 유형은 BINARY입니다. TEXT 저장소 클래스의 값에 BINARY 데이터 정렬을 사용하면 텍스트 인코딩에 관계없이 값을 나타내는 메모리 내 바이트를 비교하여 이진 데이터 정렬을 수행합니다.

NOCASE 데이터 정렬 시퀀스는 TEXT 저장소 클래스의 값에만 적용됩니다. NOCASE 데이터 정렬을 사용하면 대/소문자를 구분하지 않는 비교가 수행됩니다.

NULL, BLOB, INTEGER 또는 REAL 유형의 저장소 클래스에는 데이터 정렬 시퀀스가 사용되지 않습니다.

열에서 BINARY 이외의 데이터 정렬 유형을 사용하려면 CREATE TABLE 문에서 열 정의의 일부로 COLLATE 절을 지정해야 합니다. 두 TEXT 값을 비교할 때마다 다음 규칙에 따라 데이터 정렬 시퀀스를 사용하여 비교 결과를 결정합니다.

- 이항 비교 연사자의 경우 피연산자 중 하나가 열이면 해당 열의 기본 데이터 정렬 유형에 따라 비교에 사용되는 데이터 정렬 시퀀스가 결정됩니다. 두 피연산자가 모두 열이면 왼쪽 피연산자의 데이터 정렬 유형에 따라 사용되는 데이터 정렬 시퀀스가 결정됩니다. 두 피연산자가 모두 열이 아니면 BINARY 데이터 정렬 시퀀스가 사용됩니다.

- BETWEEN...AND 연산자는 >= 및 <= 연산자가 있는 두 표현식을 사용하는 것과 같습니다. 예를 들어 x BETWEEN y AND z 표현식은 x >= y AND x <= z와 동일합니다. 따라서 BETWEEN...AND 연산자는 위와 같은 규칙에 따라 데이터 정렬 시퀀스를 결정합니다.
- IN 연산자는 =연산자와 같은 방법으로 사용할 데이터 정렬 시퀀스를 결정합니다. 예를 들어 IN (y, z) 표현식에 사용되는 데이터 정렬 시퀀스는 x가 열인 경우 x의 기본 데이터 정렬 유형입니다. 그렇지 않은 경우에는 BINARY 데이터 정렬이 사용됩니다.
- SELECT 문에 포함된 ORDER BY 절에는 정렬 작업에 사용할 데이터 정렬 시퀀스를 명시적으로 할당할 수 있습니다. 이렇게 하면 명시적인 데이터 정렬 시퀀스가 항상 사용됩니다. 그렇지 않으면 ORDER BY 절에서 정렬하는 표현식이 열인 경우 해당 열의 기본 데이터 정렬 유형에 따라 정렬 순서가 결정됩니다. 표현식이 열이 아니면 BINARY 데이터 정렬 시퀀스가 사용됩니다.

EXPLAIN

EXPLAIN 명령 수정자는 표준이 아닌 SQL 확장입니다.

```
sql-statement  ::=  EXPLAIN sql-statement
```

다른 SQL 명령문 앞에 EXPLAIN 키워드를 사용하면 명령이 실제로 실행되는 대신 EXPLAIN 키워드가 없을 때 명령을 실행하는 데 사용될 가상 머신 명령의 시퀀스가 결과로 보고됩니다. EXPLAIN 기능은 개발자가 성능을 최적화하거나 제대로 작동하지 않는 명령문을 디버깅하기 위해 SQL 명령문 텍스트를 변경하는 데 사용할 수 있는 고급 기능입니다.

ON CONFLICT(충돌 알고리즘)

ON CONFLICT 절은 독립적인 SQL 명령이 아니며 다른 SQL 명령에 나타날 수 있는 표준이 아닌 절입니다.

```
conflict-clause     ::=  ON CONFLICT conflict-algorithm
conflict-clause     ::=  OR conflict-algorithm
conflict-algorithm  ::=  ROLLBACK |
                         ABORT |
                         FAIL |
                         IGNORE |
                         REPLACE
```

ON CONFLICT 키워드를 사용하는 ON CONFLICT 절의 첫 번째 형태는 CREATE TABLE 문에 사용됩니다. INSERT 또는 UPDATE 문에는 OR 대신 ON CONFLICT를 사용하여 구문을 자연스럽게 하는 두 번째 형태가 사용됩니다. 예를 들어 INSERT ON CONFLICT IGNORE 대신 INSERT OR IGNORE 문을 사용합니다. 두 형태의 키워드는 다르지만 절의 의미는 같습니다.

ON CONFLICT 절은 제약 조건 충돌을 해결하는 데 사용되는 알고리즘을 지정합니다. ROLLBACK, ABORT, FAIL, IGNORE 및 REPLACE라는 다섯 가지 알고리즘이 있으며 기본 알고리즘은 ABORT입니다. 다섯 가지 충돌 알고리즘에 대한 설명은 다음과 같습니다.

ROLLBACK 제약 조건 위반이 발생하면 즉시 ROLLBACK이 발생하고 현재 트랜잭션이 종료됩니다. 명령이 중단되고 SQLStatement 인스턴스에서 error 이벤트를 전달합니다. 모든 명령에서 만들어지는 암시적인 트랜잭션 이외에 활성 상태인 트랜잭션이 없으면 이 알고리즘은 ABORT와 동일하게 작동합니다.

ABORT 제약 조건 위반이 발생하면 명령에서 이전에 수행한 모든 변경 내용을 취소하고 SQLStatement 인스턴스에서 error 이벤트를 전달합니다. ROLLBACK은 실행되지 않으므로 트랜잭션 내에서 이전 명령의 변경 내용이 그대로 유지됩니다. ABORT가 기본 비헤이비어입니다.

FAIL 제약 조건 위반이 발생하면 명령이 중단되고 SQLStatement에서 error 이벤트를 전달합니다. 그러나 제약 조건 위반이 발생하기 전에 명령문에서 데이터베이스를 변경한 내용은 취소되지 않고 모두 유지됩니다. 예를 들어 UPDATE 문에서 업데이트를 시도한 100번째 행에서 제약 조건 위반이 발생한 경우 처음 99개 행의 변경 내용은 그대로 유지되고 100번째 행과 이후 행은 변경되지 않습니다.

IGNORE 제약 조건 위반이 발생하면 제약 조건 위반이 들어 있는 해당 행은 삽입되거나 변경되지 않습니다. 이 행을 무시하고 명령이 계속 정상적으로 실행됩니다. 제약 조건 위반이 들어 있는 행의 앞뒤에 있는 다른 행은 계속 정상적으로 삽입되거나 업데이트됩니다. 오류는 반환되지 않습니다.

REPLACE UNIQUE 제약 조건 위반이 발생하면 제약 조건을 위반한 기존 행을 제거한 후 현재 행을 삽입하거나 업데이트합니다. 따라서 삽입이나 업데이트가 항상 수행되고 명령이 계속 정상적으로 실행됩니다. 오류는 반환되지 않습니다. NOT NULL 제약 조건 위반이 발생하면 NULL 값이 해당 행의 기본값으로 바뀝니다. 열에 기본값이 없으면 ABORT 알고리즘이 사용됩니다. CHECK 제약 조건 위반이 발생하면 IGNORE 알고리즘이 사용됩니다. 이 충돌 해결 전략에 따라 제약 조건을 만족하기 위해 행을 삭제하는 경우에는 해당 행에 대한 삭제 트리거가 발생하지 않습니다.

INSERT 또는 UPDATE 문의 OR 절에 지정된 알고리즘은 CREATE TABLE 문에 지정된 모든 알고리즘을 재정의합니다. CREATE TABLE 문이나 실행 중인 INSERT 또는 UPDATE 문에 지정된 알고리즘이 없으면 ABORT 알고리즘이 사용됩니다.

REINDEX

REINDEX 명령은 하나 이상의 인덱스를 삭제하고 다시 만드는 데 사용됩니다. 이 명령은 데이터 정렬 시퀀스의 정의가 변경되었을 때 유용합니다.

```
sql-statement  ::=  REINDEX collation-name
sql-statement  ::=  REINDEX [database-name .] ( table-name | index-name )
```

첫 번째 형태에서는 연결된 모든 데이터베이스에서 지정된 데이터 정렬 시퀀스를 사용하는 모든 인덱스를 다시 만듭니다. table-name이 지정된 두 번째 형태에서는 특정 테이블에 연결된 모든 인덱스를 다시 만듭니다. index-name을 지정하면 해당 인덱스만 삭제하고 다시 만듭니다.

COMMENTS

주석은 SQL 명령이 아니지만 SQL 쿼리에 나타날 수 있습니다. 주석은 런타임에서 공백으로 간주하며, 여러 줄에 걸친 표현식 내부를 포함하여 공백을 사용할 수 있는 모든 위치에서 시작될 수 있습니다.

```
comment             ::=  single-line-comment |
                         block-comment
single-line-comment ::=  -- single-line
block-comment       ::=  /* multiple-lines or block [*/]
```

한 줄 주석은 대시 두 개로 나타냅니다. 한 줄 주석은 현재 행의 끝까지만 확장될 수 있습니다.

블록 주석은 여러 행으로 확장되거나 한 줄에 포함될 수 있습니다. 종결 구분 기호가 없으면 블록 주석이 입력의 끝까지 확장됩니다. 이러한 경우는 오류로 간주되지 않습니다. 행에서 블록 주석이 끝난 위치에 새 SQL 명령문을 시작할 수 있습니다. 블록 주석은 표현식 내부 및 다른 SQL 명령문 중간을 비롯하여 공백을 사용할 수 있는 모든 위치에 포함될 수 있습니다. 블록 주석은 중첩되지 않습니다. 블록 주석 내에 있는 한 줄 주석은 무시됩니다.

EXPRESSIONS

표현식은 다른 SQL 블록 내의 하위 명령입니다. SQL 명령문 내에서 표현식의 유효한 구문은 다음과 같습니다.

```
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
```

표현식은 단일 값으로 확인될 수 있는 값과 연산자의 모든 조합입니다. 부울(true 또는 false) 값으로 확인되는지 아니면 부울이 아닌 값으로 확인되는지에 따라 표현식을 두 가지 일반적인 유형으로 나눌 수 있습니다.

WHERE 절, HAVING 절, JOIN 절의 ON 표현식, CHECK 표현식을 비롯한 몇 가지 일반적인 경우 표현식은 부울 값으로 확인되어야 합니다. 이 조건에 맞는 표현식 유형은 다음과 같습니다.

- ISNULL
- NOTNULL
- IN ()
- EXISTS ()
- LIKE
- GLOB
- 특정 함수
- 특정 연산자(특히 비교 연산자)

리터럴 값

리터럴 숫자 값은 정수 또는 부동 소수점 숫자로 작성됩니다. 공학 표기법도 지원됩니다. 마침표(.) 문자가 항상 소수점으로 사용됩니다.

문자열 리터럴은 문자열을 작은따옴표(')로 묶어 나타냅니다. 문자열 안에 작은따옴표 하나를 포함하려면 ''와 같이 작은따옴표 두 개를 연속으로 추가합니다.

부울 리터럴은 true 또는 false 값으로 나타냅니다. 리터럴 부울 값은 부울 열 데이터 유형에 사용됩니다.

BLOB 리터럴은 16진수 데이터가 들어 있는 문자열 리터럴이며 X'53514697465'와 같이 앞에 x 또는 X 문자 하나가 옵니다.

리터럴 값이 NULL 토큰일 수도 있습니다.

열 이름

열 이름은 CREATE TABLE 문에 정의된 이름이거나 특수 식별자인 ROWID, OID 또는 _ROWID_ 중 하나일 수 있습니다. 이러한 특수 식별자는 모든 테이블의 모든 행에 연결된 고유 난수 정수 키("행 키")를 나타냅니다. 특수 식별자는 CREATE TABLE 문에 이름이 같은 실제 열이 정의되지 않은 경우에만 행 키를 참조합니다. 행 키는 읽기 전용 열처럼 작동합니다. 일반 열을 사용할 수 있는 모든 위치에 행 키를 사용할 수 있지만 UPDATE 또는 INSERT 문에서는 행 키의 값을 변경할 수 없습니다. SELECT * FROM table 문의 결과 집합에는 행 키가 포함되지 않습니다.

SELECT 문

표현식에서 SELECT 문은 IN 연산자의 오른쪽 피연산자, 스칼라 양(단일 결과 값) 또는 EXISTS 연산자의 피연산자로 나타날 수 있습니다. 스칼라 양이나 IN 연산자의 피연산자로 사용된 경우 SELECT의 결과에는 단일 열만 있을 수 있습니다. UNION 또는 EXCEPT 등의 키워드로 연결된 복합 SELECT 문도 사용할 수 있습니다. EXISTS 연산자를 사용하면 SELECT의 결과 집합에서 열이 무시되고 행이 하나 이상 있으면 표현식에서 TRUE를, 결과 집합이 비어 있으면 FALSE를 반환합니다. SELECT 표현식의 항목 중 포함하는 쿼리의 값을 참조하는 항목이 없으면 다른 내용이 처리되기 전에 해당 표현식이 가장 먼저 평가되고 결과가 필요에 따라 다시 사용됩니다. SELECT 표현식에 외부 쿼리(상관 하위 쿼리)의 변수가 들어 있으면 필요할 때마다 SELECT가 다시 평가됩니다.

SELECT가 IN 연산자의 오른쪽 피연산자인 경우 IN 연산자는 왼쪽 피연산자의 결과가 SELECT 문의 결과 집합에 있는 값 중 하나와 같으면 TRUE를 반환합니다. IN 연산자 앞에 NOT 키워드를 사용하면 정반대의 논리로 테스트할 수 있습니다.

SELECT가 표현식 내에 나오지만 IN 연산자의 오른쪽 피연산자가 아닌 경우 SELECT의 첫 번째 결과 행이 표현식에서 값으로 사용됩니다. SELECT가 결과 행을 둘 이상 반환하면 첫 번째 행 이후의 모든 행이 무시됩니다. SELECT에서 행을 반환하지 않으면 SELECT의 값이 NULL입니다.

CAST 표현식

CAST 표현식은 지정된 값의 데이터 유형을 다른 유형으로 변경합니다. 지정된 유형은 CREATE TABLE 문의 열 정의에서 유형으로 사용할 수 있는 비어 있지 않은 모든 유형 이름일 수 있습니다. 자세한 내용은 데이터 유형 지원을 참조하십시오.

추가 표현식 요소

다음 SQL 요소도 표현식에 사용할 수 있습니다.

- 내장 함수: 집계 함수, 스칼라 함수, 날짜 및 시간 서식 지정 함수
- 연산자
- 매개 변수

내장 함수

내장 함수에는 세 가지 주요 범주가 있습니다.

- 집계 함수
- 스칼라 함수
- 날짜 및 시간 함수

이러한 함수 외에도 트리거를 실행할 때 오류를 알리는 데 사용되는 RAISE()라는 특수 함수가 있습니다. 이 함수는 CREATE TRIGGER 문의 본문 내에서만 사용할 수 있습니다. RAISE() 함수에 대한 자세한 내용은 CREATE TRIGGER > RAISE()를 참조하십시오.

SQL의 모든 키워드와 마찬가지로 함수 이름에는 대/소문자가 구분되지 않습니다.

집계 함수

집계 함수는 여러 행의 값에 대해 작업을 수행합니다. 이러한 함수는 주로 SELECT 문에서 GROUP BY 절과 함께 사용됩니다.

AVG(X)	그룹 내에서 NULL이 아닌 모든 X의 평균 값을 반환합니다. 숫자로 인식할 수 없는 문자열 및 BLOB 값은 0으로 해석됩니다. 입력이 모두 정수인 경우에도 AVG()의 결과는 항상 부동 소수점 값입니다.
COUNT(X) COUNT(*)	첫 번째 형태는 그룹에서 X가 NULL이 아닌 횟수를 세어 반환합니다. * 인수가 있는 두 번째 형태는 그룹에 있는 행의 총 개수를 반환합니다.
MAX(X)	그룹에 있는 모든 값 중 최대값을 반환합니다. 일반적인 정렬 순서를 사용하여 최대값을 확인합니다.
MIN(X)	그룹에 있는 모든 값 중 NULL이 아닌 최소값을 반환합니다. 일반적인 정렬 순서를 사용하여 최소값을 확인합니다. 그룹의 모든 값이 NULL이면 NULL이 반환됩니다.
SUM(X)	그룹에 있는 NULL이 아닌 모든 값의 숫자 합계를 반환합니다. 모든 값이 NULL이면 SUM()에서 NULL을, TOTAL()에서 0.0을 반환합니다. TOTAL()의 결과는 항상 부동 소수점 값입니다. NULL이 아닌 모든 입력이 정수이면 SUM()의 결과는 정수 값입니다. SUM()에 NULL이 아니며 정수가 아닌 입력이 하나라도 있으면 SUM()에서 부동 소수점 값을 반환합니다. 이 값은 실제 합계의 근사값일 수 있습니다.
TOTAL(X)	

인수 하나를 받는 위와 같은 집계 함수에서는 인수 앞에 DISTINCT 키워드를 사용할 수 있습니다. 이렇게 하면 중복 요소가 필터링된 후 집계 함수에 전달됩니다. 예를 들어 COUNT(DISTINCT x) 함수를 호출하면 x 열에 있는 NULL이 아닌 값의 총 개수 대신 X 열의 고유 값 개수가 반환됩니다.

스칼라 함수

스칼라 함수는 한 번에 한 행의 값을 계산합니다.

ABS(X)	인수 X의 절대값을 반환합니다.
COALESCE(X, Y, ...)	NULL이 아닌 첫 번째 인수의 복사본을 반환합니다. 모든 인수가 NULL이면 NULL이 반환됩니다. 인수는 최소한 두 개 이상이어야 합니다.
GLOB(X, Y)	이 함수는 X GLOB Y 구문을 구현하는 데 사용됩니다.
IFNULL(X, Y)	NULL이 아닌 첫 번째 인수의 복사본을 반환합니다. 두 인수가 모두 NULL이면 NULL이 반환됩니다. 이 함수의 비헤이비어는 COALESCE()와 같습니다.
HEX(X)	인수를 BLOB 저장소 유형 값으로 해석합니다. 결과는 이 값의 내용을 16진수로 렌더링한 것입니다.
LAST_INSERT_ROWID()	현재 SQLConnection을 통해 데이터베이스에 삽입된 마지막 행의 행 식별자(생성된 기본 키)를 반환합니다. 이 값은 SQLConnection.lastInsertRowID 속성이 반환하는 값과 동일합니다.
LENGTH(X)	X의 문자열 길이를 문자 수 단위로 반환합니다.
LIKE(X, Y [, Z])	이 함수는 SQL의 X LIKE Y [ESCAPE Z] 구문을 구현하는 데 사용됩니다. 선택적 절인 ESCAPE가 있으면 세 개의 인수를 사용하여 함수가 호출됩니다. 그렇지 않으면 두 인수만 사용하여 호출됩니다.
LOWER(X)	X 문자열의 모든 문자를 소문자로 변환한 복사본을 반환합니다.
LTRIM(X) LTRIM(X, Y)	X의 왼쪽에서 공백을 제거하여 구성한 문자열을 반환합니다. Y 인수를 지정하면 X의 왼쪽에서 Y에 있는 모든 문자를 제거합니다.
MAX(X, Y, ...)	최대값을 갖는 인수를 반환합니다. 인수는 숫자뿐 아니라 문자열일 수 있습니다. 정의된 정렬 순서에 따라 최대값을 결정합니다. MAX()는 인수가 2개 이상이면 단순 함수이지만 인수가 하나뿐이면 집계 함수입니다.
MIN(X, Y, ...)	최소값을 갖는 인수를 반환합니다. 인수는 숫자뿐 아니라 문자열일 수 있습니다. 정의된 정렬 순서에 따라 최소값을 결정합니다. MIN()은 인수가 2개 이상이면 단순 함수이지만 인수가 하나뿐이면 집계 함수입니다.
NULLIF(X, Y)	인수가 서로 다르면 첫 번째 인수를, 그렇지 않으면 NULL을 반환합니다.
QUOTE(X)	이 루틴은 다른 SQL 명령문에 포함할 수 있는 인수 값인 문자열을 반환합니다. 문자열은 작은따옴표로 묶이며 필요에 따라 내부에 있는 따옴표에 이스케이프가 추가됩니다. BLOB 저장소 클래스는 16진수 리터럴로 인코딩됩니다. 이 함수는 실행 취소/다시 실행 기능을 구현하는 트리거를 작성할 때 유용합니다.
RANDOM(*)	-9223372036854775808에서 9223372036854775807 사이의 의사 난수 정수를 반환합니다. 이 난수 값은 crypto-strong이 아닙니다.
RANDOMBLOB(N)	의사 난수 바이트가 들어 있는 N바이트 BLOB을 반환합니다. N은 양의 정수여야 합니다. 이 난수 값은 crypto-strong이 아닙니다. N 값이 음수이면 단일 바이트가 반환됩니다.
ROUND(X) ROUND(X, Y)	숫자 X를 소수점 오른쪽 Y자리로 반올림합니다. Y 인수를 생략하면 0이 사용됩니다.
RTRIM(X) RTRIM(X, Y)	X의 오른쪽에서 공백을 제거하여 구성한 문자열을 반환합니다. Y 인수를 지정하면 X의 오른쪽에서 Y에서 있는 모든 문자를 제거합니다.

SUBSTR(X, Y, Z)	입력 문자열 X에서 Y번째 문자로 시작되고 길이가 Z자인 부분 문자열을 반환합니다. X에서 가장 왼쪽에 있는 문자의 인덱스 위치는 1입니다. Y가 음수이면 왼쪽이 아닌 오른쪽부터 계산하여 부분 문자열의 첫 번째 문자를 찾습니다.
TRIM(X) TRIM(X, Y)	X의 오른쪽에서 공백을 제거하여 구성한 문자열을 반환합니다. Y 인수를 지정하면 X의 오른쪽에서 Y에 서 있는 모든 문자를 제거합니다.
TYPEOF(X)	X 표현식의 유형을 반환합니다. 가능한 반환 값은 'null', 'integer', 'real', 'text' 및 'blob'입니다. 데이터 유형에 대한 자세한 내용은 데이터 유형 지원을 참조하십시오.
UPPER(X)	입력 문자열 X를 모두 대문자로 변환한 복사본을 반환합니다.
ZEROBLOB(N)	N바이트의 0x00이 들어 있는 BLOB을 반환합니다.

날짜 및 시간 형식 함수

날짜 및 시간 형식 함수는 형식이 지정된 날짜 및 시간 데이터를 만드는 데 사용되는 스칼라 함수 그룹입니다. 이러한 함수는 문자열 및 숫자 값을 계산하고 반환하며, DATE 데이터 유형에는 사용할 수 없습니다. 선언된 데이터 유형이 DATE인 열의 데이터에 대해 이러한 함수를 사용하면 정상적으로 작동하지 않습니다.

DATE(T, ...)	DATE() 함수는 YYYY-MM-DD 서식의 날짜가 들어 있는 문자열을 반환합니다. 첫 번째 매개 변수(T)는 시간 서식에 있는 서식의 시간 문자열을 지정합니다. 시간 문자열 뒤에 수정자를 개수에 제한 없이 지정할 수 있습니다. 수정자에 다양한 수정자가 나와 있습니다.
TIME(T, ...)	TIME() 함수는 HH:MM:SS 서식의 시간이 들어 있는 문자열을 반환합니다. 첫 번째 매개 변수(T)는 시간 서식에 있는 서식의 시간 문자열을 지정합니다. 시간 문자열 뒤에 수정자를 개수에 제한 없이 지정할 수 있습니다. 수정자에 다양한 수정자가 나와 있습니다.
DATETIME(T, ...)	DATETIME() 함수는 YYYY-MM-DD HH:MM:SS 서식의 날짜 및 시간이 들어 있는 문자열을 반환합니다. 첫 번째 매개 변수(T)는 시간 서식에 있는 서식의 시간 문자열을 지정합니다. 시간 문자열 뒤에 수정자를 개수에 제한 없이 지정할 수 있습니다. 수정자에 다양한 수정자가 나와 있습니다.
JULIANDAY(T, ...)	JULIANDAY() 함수는 기원전 4714년 11월 24일 그리니치 정오와 제공된 날짜 사이에 경과한 날짜 수를 나타내는 숫자를 반환합니다. 첫 번째 매개 변수(T)는 시간 서식에 있는 서식의 시간 문자열을 지정합니다. 시간 문자열 뒤에 수정자를 개수에 제한 없이 지정할 수 있습니다. 수정자에 다양한 수정자가 나와 있습니다.
STRFTIME(F, T, ...)	STRFTIME() 루틴은 첫 번째 인수인 F에 지정된 서식 문자열에 따라 서식이 지정된 날짜를 반환합니다. 서식 문자열에 다음과 같은 대체 문자를 사용할 수 있습니다. %d - 날짜 %f - 분수 초 SS.SSS %H - 시간 00-24 %j - 연중 날짜 001-366 %J - Julian 날짜 번호 %m -월 01-12 %M - 분 00-59 %s - 1970-01-01 이후 경과한 초 %S - 초 00-59 %w - 요일 0-6(일요일 = 0) %W - 연중 주 00-53 %Y - 연도 0000-9999 %% - % 두 번째 매개 변수(T)는 시간 서식에 있는 서식의 시간 문자열을 지정합니다. 시간 문자열 뒤에 수정자를 개수에 제한 없이 지정할 수 있습니다. 수정자에 다양한 수정자가 나와 있습니다.

시간 형식

시간 문자열은 다음 형식 중 하나일 수 있습니다.

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(날짜는 2000-01-01)
HH:MM:SS	07:30:59(날짜는 2000-01-01)
HH:MM:SS.SSS	07:30:59:152(날짜는 2000-01-01)
now	현재 날짜 및 시간(협정 세계시)
DDDD.DDDD	Julian 날짜 번호(부동 소수점 숫자)

이러한 형식의 T 문자는 날짜와 시간을 구분하는 리터럴 문자 "T"입니다. 시간만 포함되어 있는 형식에서는 날짜를 2001-01-01로 가정합니다.

수정자

시간 문자열 뒤에 날짜를 수정하거나 날짜를 다르게 해석하는 0개 이상의 수정자를 추가할 수 있습니다. 사용 가능한 수정자는 다음과 같습니다.

NNN days	시간에 추가할 날짜 수입니다.
NNN hours	시간에 추가할 시간 수입니다.
NNN minutes	시간에 추가할 분 수입니다.
NNN.NNNN seconds	시간에 추가할 초 및 밀리초 수입니다.
NNN months	시간에 추가할 개월 수입니다.
NNN years	시간에 추가할 연도 수입니다.
start of month	시간을 월초로 되돌립니다.
start of year	시간을 연초로 되돌립니다.
start of day	시간을 해당 날짜의 자정으로 되돌립니다.
weekday N	시간을 지정된 요일로 옮깁니다. 0 = 일요일, 1 = 월요일 등입니다.
localtime	날짜를 현지 시간으로 변환합니다.
utc	날짜를 협정 세계시로 변환합니다.

연산자

SQL에서는 대부분의 프로그래밍 언어에 있는 일반적인 연산자 및 SQL에만 사용되는 몇 가지 연산자를 비롯하여 매우 다양한 연산자를 지원합니다.

일반적인 연산자

SQL 블록에서 다음과 같은 이항 연산자를 사용할 수 있으며, 우선 순위가 높은 연산자부터 나열되어 있습니다.

```
*    /    %
+    -
<< >> & |
< >=   > >=
=    ==   !=   <> IN
AND
OR
```

지원되는 단항 접두어 연산자는 다음과 같습니다.

```
 !    ~    NOT
```

COLLATE 연산자는 단항 접미어 연산자로 간주할 수 있습니다. COLLATE 연산자는 우선 순위가 가장 높습니다. 이 연산자는 항상 모든 접두어 단항 연산자 또는 이항 연산자보다 우선적으로 결합됩니다.

같음 연산자와 같지 않음 연산자에는 두 가지 변형이 있습니다. 같음 연산자는 = 또는 ==일 수 있습니다. 같지 않음 연산자는 != 또는 <>입니다.

|| 연산자는 피연산자인 두 문자열을 연결하는 문자열 결합 연산자입니다.

% 연산자는 왼쪽 피연산자를 오른쪽 피연산자로 나눈 나머지를 출력합니다.

결과가 문자열인 || 결합 연산자를 제외하고 모든 이항 연산자의 결과는 숫자 값입니다.

SQL 연산자
LIKE

LIKE 연산자는 패턴 일치 비교를 수행합니다.

```
expr     ::=  (column-name | expr) LIKE pattern
pattern  ::=  '[ string | % | _ ]'
```

LIKE 연산자의 오른쪽 피연산자에는 패턴이 들어 있고 오른쪽 피연산자에는 패턴과 비교할 문자열이 들어 있습니다. 패턴의 백분율 심볼(%)은 문자열에서 0개 이상의 모든 문자 시퀀스와 일치하는 와일드카드 문자입니다. 패턴의 밑줄(_)은 문자열에서 모든 단일 문자와 일치합니다. 다른 문자는 모두 자신과 일치하거나 해당 소/대문자와 일치합니다. 즉, 대/소문자를 구분하지 않고 비교됩니다. 데이터베이스 엔진에서는 7비트 라틴 문자의 대/소문자만 인식합니다. 따라서 8비트 iso8859 문자나 UTF-8 문자의 경우 LIKE 연산자에서 대/소문자를 구분합니다. 예를 들어 'a' LIKE 'A' 표현식은 TRUE이지만 'æ' LIKE 'Æ' 표현식은 FALSE입니다. 라틴 문자의 대/소문자 구분 여부를 변경하려면 SQLConnection.caseSensitiveLike 속성을 사용합니다.

선택적 절인 ESCAPE가 있으면 ESCAPE 키워드 뒤에 오는 표현식이 단일 문자로 구성된 문자열로 평가되어야 합니다. LIKE 패턴에서 이 문자를 사용하여 리터럴 백분율 또는 밑줄 문자와 비교할 수 있습니다. 이스케이프 문자 뒤에 백분율 심볼, 밑줄 또는 이스케이프 문자 자체가 나오면 문자열에서 리터럴 백분율 심볼, 밑줄 또는 이스케이프 문자와 각각 일치합니다.

GLOB

GLOB 연산자는 LIKE와 비슷하지만 와일드카드에 Unix 파일 글로빙 구문이 사용됩니다. LIKE와는 달리 GLOB은 대/소문자를 구분합니다.

IN

IN 연산자는 왼쪽 피연산자가 오른쪽 피연산자(괄호로 묶은 값 집합)의 값 중 하나와 같은지 여부를 계산합니다.

```
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]*
```

오른쪽 피연산자는 쉼표로 구분된 리터럴 값 집합이거나 SELECT 문의 결과일 수 있습니다. SELECT 문을 IN 연산자의 오른쪽 피연산자로 사용하는 방법 및 제한 사항은 SELECT 문을 참조하십시오.

BETWEEN...AND

BETWEEN...AND 연산자는 >= 및 <= 연산자가 있는 두 표현식을 사용하는 것과 같습니다. 예를 들어 x BETWEEN y AND z 표현식은 x >= y AND x <= z와 같습니다.

NOT

NOT 연산자는 부정 연산자입니다. GLOB, LIKE 및 IN 연산자 앞에 NOT 키워드를 추가하여 정반대의 논리로 테스트할 수 있습니다. 즉, 값이 지정된 패턴과 일치하지 않는지 확인할 수 있습니다.

매개 변수

매개 변수는 표현식에서 리터럴 값에 대한 자리 표시자를 지정하며, 이러한 자리 표시자는 런타임에 SQLStatement.parameters 연결 배열에 값을 할당하여 채워집니다. 매개 변수에는 다음과 같은 세 가지 형태가 있습니다.

?	물음표는 인덱싱된 매개 변수를 나타냅니다. 명령문에서 매개 변수의 순서에 따라 매개 변수에 0부터 시작하는 숫자 인덱스 값이 할당됩니다.

:AAAA 콜론 뒤에 식별자 이름을 추가하면 AAAA라는 이름으로 명명된 매개 변수가 사용됩니다. SQL 명령문에서의 순서에 따라 명명된 매개 변수에 번호를 매길 수도 있습니다. 혼동을 피하려면 명명된 매개 변수와 번호가 매겨진 매개 변수 중 하나만 사용하는 것이 좋습니다.

@AAAA "at 기호"는 콜론과 같습니다.

지원되지 않는 SQL 기능

다음은 Adobe AIR에서 지원되지 않는 표준 SQL 요소의 목록입니다.

FOREIGN KEY 제약 조건 FOREIGN KEY 제약 조건은 파싱되지만 적용되지 않습니다.

트리거 FOR EACH STATEMENT 트리거는 지원되지 않으며 모든 트리거는 FOR EACH ROW여야 합니다. INSTEAD OF 트리거는 테이블에서 지원되지 않습니다. INSTEAD OF 트리거는 뷰에만 사용할 수 있습니다. 재귀 트리거(자신을 트리거하는 트리거)는 지원되지 않습니다.

ALTER TABLE ALTER TABLE 명령의 RENAME TABLE 및 ADD COLUMN 변형만 지원됩니다. 다른 종류의 ALTER TABLE 작업(예: DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT 등)은 무시됩니다.

중첩된 트랜잭션 단일 활성 트랜잭션만 사용할 수 있습니다.

RIGHT 및 FULL OUTER JOIN RIGHT OUTER JOIN 또는 FULL OUTER JOIN은 지원되지 않습니다.

업데이트 가능한 뷰 뷰는 읽기 전용입니다. 뷰에 대해서는 DELETE, INSERT 또는 UPDATE 문을 실행할 수 없습니다. 뷰에 대해 DELETE, INSERT 또는 UPDATE를 시도할 때 발생하는 INSTEAD OF 트리거는 지원되며, 이를 통해 지원하는 테이블을 트리거 본문에서 업데이트할 수 있습니다.

GRANT 및 REVOKE 데이터베이스는 일반적인 디스크 파일이며 내부 운영 체제의 기본적인 파일 액세스 권한만 적용할 수 있습니다. 클라이언트/서버 RDBMS에서 자주 사용되는 GRANT 및 REVOKE 명령은 구현되지 않습니다.

다음과 같은 SQL 요소 및 SQLite 기능은 일부 SQLite 구현에서 지원되지만 Adobe AIR에서는 지원되지 않습니다. 이러한 기능은 대부분 SQLConnection 클래스의 메서드를 통해 사용할 수 있습니다.

트랜잭션 관련 SQL 요소(BEGIN, END, COMMIT, ROLLBACK) 이 기능은 SQLConnection 클래스의 트랜잭션 관련 메서드인 SQLConnection.begin(), SQLConnection.commit() 및 SQLConnection.rollback()을 통해 사용할 수 있습니다.

ANALYZE 이 기능은 SQLConnection.analyze() 메서드를 통해 사용할 수 있습니다.

ATTACH 이 기능은 SQLConnection.attach() 메서드를 통해 사용할 수 있습니다.

COPY 이 명령문은 지원되지 않습니다.

CREATE VIRTUAL TABLE 이 명령문은 지원되지 않습니다.

DETACH 이 기능은 SQLConnection.detach() 메서드를 통해 사용할 수 있습니다.

PRAGMA 이 명령문은 지원되지 않습니다.

VACUUM 이 기능은 SQLConnection.compact() 메서드를 통해 사용할 수 있습니다.

시스템 테이블에 액세스할 수 없음 SQL 명령문에서 "sqlite_" 접두어가 있는 sqlite_master 등의 시스템 테이블을 사용할 수 없습니다. 런타임에는 객체 지향 방식으로 스키마 데이터에 액세스할 수 있는 스키마 API가 포함되어 있습니다. 자세한 내용은 SQLConnection.loadSchema() 메서드를 참조하십시오.

일반 표현식 함수(MATCH() 및 REGEX()) 이러한 함수는 SQL 명령문에서 사용할 수 없습니다.

다음과 같은 기능은 여러 SQLite 구현과 Adobe AIR 간에 차이점이 있습니다.

인덱싱된 명령문 매개 변수 여러 구현에서는 인덱싱된 명령문 매개 변수가 1부터 시작됩니다. 그러나 Adobe AIR의 인덱싱된 명령문 매개 변수는 0부터 시작됩니다. 즉, 첫 번째 매개 변수에는 인덱스 0이 지정되고 두 번째 매개 변수에는 인덱스 1이 지정되는 방식입니다.

INTEGER PRIMARY KEY 열 정의 여러 구현에서 정확하게 INTEGER PRIMARY KEY로 정의된 열만 테이블에 대한 실제 기본 키 열로 사용됩니다. 이러한 구현에서는 일반적으로 INTEGER의 동의어(예: int)인 다른 데이터 유형을 사용해도 열이 내부 기본 키로 사용되지 않습니다. 그러나 Adobe AIR에서는 int 데이터 유형과 INTEGER의 다른 동의어가 정확하게 INTEGER와

같은 것으로 간주됩니다. 따라서 int PRIMARY KEY로 정의된 열이 테이블에 대한 내부 기본 키로 사용됩니다. 자세한 내용은 "CREATE TABLE" 및 "열 선호도" 단원을 참조하십시오.

추가 SQL 기능

다음과 같은 열 선호도 유형은 SQLite에서 기본적으로 지원되지 않지만 Adobe AIR에서는 지원됩니다. SQL의 모든 키워드와 마찬가지로 이러한 데이터 유형 이름에는 대/소문자가 구분되지 않습니다.

부울 Boolean 클래스에 해당합니다.

DATE Date 클래스에 해당합니다.

int int 클래스에 해당합니다(INTEGER 열 선호도와 동일).

Number Number 클래스에 해당합니다(REAL 열 선호도와 동일).

Object Object 클래스 또는 AMF3을 사용하여 직렬화 및 비직렬화할 수 있는 모든 하위 클래스에 해당합니다. 여기에는 사용자 정의 클래스를 비롯한 대부분의 클래스가 포함되지만 표시 객체 및 표시 객체가 속성으로 포함되어 있는 일부 클래스는 제외됩니다.

String String 클래스에 해당합니다(TEXT 열 선호도와 동일).

XML ActionScript(E4X) XML 클래스에 해당합니다.

XMLList ActionScript(E4X) XMLList 클래스에 해당합니다.

다음과 같은 리터럴 값은 SQLite에서 기본적으로 지원되지 않지만 Adobe AIR에서는 지원됩니다.

true BOOLEAN 열에서 리터럴 부울 값인 true를 나타내는 데 사용됩니다.

false BOOLEAN 열에서 리터럴 부울 값인 false를 나타내는 데 사용됩니다.

데이터 유형 지원

대부분의 SQL 데이터베이스와 달리 Adobe AIR SQL 데이터베이스 엔진에는 테이블 열에 특정 유형의 값이 들어 있어야 한다는 제한 사항이 없습니다. 대신 런타임에서 저장소 클래스와 열 선호도라는 두 가지 개념을 사용하여 데이터 유형을 제어합니다. 이 단원에서는 저장소 클래스와 열 선호도에 대해 설명하고 다양한 조건에서 데이터 유형의 차이가 해결되는 방식을 보여 줍니다.

- 1024페이지의 “저장소 클래스”
- 1025페이지의 “열 선호도”
- 1027페이지의 “데이터 유형과 비교 연산자”
- 1027페이지의 “데이터 유형과 수학 연산자”
- 1028페이지의 “데이터 유형과 정렬”
- 1028페이지의 “데이터 유형과 그룹화”
- 1028페이지의 “데이터 유형과 복합 SELECT 문”

저장소 클래스

저장소 클래스는 데이터베이스에서 값을 저장하는 데 사용되는 실제 데이터 유형을 나타냅니다. 다음은 데이터베이스에서 사용되는 저장소 클래스입니다.

NULL 값이 NULL 값입니다.

INTEGER 값이 부호 있는 정수입니다.

REAL 값이 부동 소수점 숫자 값입니다.

TEXT 값이 텍스트 문자열(256MB로 제한)입니다.

BLOB 값이 BLOB(Binary Large Object)이라는 원시 이진 데이터(256MB로 제한)입니다.

SQL 명령문에 리터럴로 포함되어 데이터베이스에 제공되는 모든 값이나 매개 변수를 사용하여 미리 준비된 SQL 명령문에 바인딩된 값에 저장소 클래스가 할당된 후 SQL 명령문이 실행됩니다.

SQL 명령문에 포함된 리터럴에 할당되는 저장소 클래스는 작은따옴표나 큰따옴표로 묶인 경우 TEXT, 리터럴이 따옴표로 묶이지 않았으며 소수점 또는 지수가 없는 숫자로 지정된 경우 INTEGER, 리터럴이 따옴표로 묶이지 않았으며 소수점 또는 지수가 있는 숫자이면 REAL, 값이 NULL이면 NULL입니다. 저장소 클래스가 BLOB인 리터럴은 X'ABCD' 표기법을 사용하여 지정합니다. 자세한 내용은 표현식의 리터럴 값을 참조하십시오.

SQLStatement.parameters 연결 배열을 사용하여 매개 변수로 제공된 값에는 바인딩된 고유 데이터 유형에 가장 가까운 저장소 클래스가 할당됩니다. 예를 들어 int 값은 INTEGER 저장소 클래스로 바인딩되고, Number 값에는 REAL 저장소 클래스가 지정되고, String 값에는 TEXT 저장소 클래스가 지정되고, ByteArray 객체에는 BLOB 저장소 클래스가 지정됩니다.

열 선호도

열의 **선호도**는 해당 열에 저장되는 데이터의 권장 유형입니다. INSERT 또는 UPDATE 문을 통해 값이 열에 저장되면 런타임에서 값을 원래 데이터 유형에서 지정된 선호도로 변환하려고 합니다. 예를 들어 ActionScript 또는 JavaScript Date 인스턴스인 Date 값을 선호도가 TEXT인 열에 삽입하면 Date 값이 String 표현으로 변환된 후 데이터베이스에 저장됩니다. 이는 객체의 toString() 메서드를 호출하는 것과 같습니다. 값을 지정된 선호도로 변환할 수 없으면 오류가 발생하고 작업이 수행되지 않습니다. SELECT 문을 사용하여 데이터베이스에서 검색한 값은 선호도에 해당하는 클래스의 인스턴스로 반환됩니다. 이는 해당 값이 다른 데이터 유형에서 변환되어 저장되었는지 여부와 관계가 없습니다.

열에 NULL 값을 사용할 수 있으면 ActionScript 또는 JavaScript 값인 null을 매개 변수 값으로 사용하여 열에 NULL을 저장할 수 있습니다. SELECT 문에서 NULL 저장소 클래스 값이 검색되면 이 값은 열의 선호도에 관계없이 항상 ActionScript 또는 JavaScript 값인 null로 반환됩니다. 열에 NULL 값을 사용할 수 있는 경우 항상 해당 열에서 검색된 값이 null인지 확인한 후 값을 Number 또는 Boolean 등의 nullable이 아닌 유형으로 변환해야 합니다.

데이터베이스의 각 열에는 다음과 같은 유형 선호도 중 하나가 할당됩니다.

- TEXT 또는 STRING
- NUMERIC
- INTEGER 또는 int
- REAL 또는 NUMBER
- 부울
- DATE
- XML
- XMLLIST
- Object
- NONE

TEXT 또는 STRING

선호도가 TEXT 또는 STRING인 열은 저장소 클래스 NULL, TEXT 또는 BLOB을 사용하여 모든 데이터를 저장합니다. 선호도가 TEXT인 열에 숫자 데이터를 삽입하면 숫자 데이터가 텍스트로 변환된 후 저장됩니다.

NUMERIC

선호도가 NUMERIC인 열은 저장소 클래스 NULL, REAL 또는 INTEGER를 사용하여 값을 저장합니다. NUMERIC 열에 텍스트 데이터를 삽입하면 텍스트 데이터를 정수나 실수로 변환한 후 저장합니다. 변환에 성공하면 INTEGER 또는 REAL 저장소 클래스를 사용하여 값이 저장됩니다. 예를 들어 '10.05' 값은 REAL 저장소 클래스로 변환된 후 저장됩니다. 변환할 수 없으면

오류가 발생하며 NULL 값으로 변환되지는 않습니다. NUMERIC 열에서 검색된 값은 해당 값에 맞는 가장 구체적인 숫자 유형의 인스턴스로 반환됩니다. 즉, 값이 양의 정수이거나 0이면 uint 인스턴스로 반환됩니다. 음의 정수이면 int 인스턴스로 반환됩니다. 마지막으로 값이 정수가 아니고 부동 소수점 구성 요소가 있으면 Number 인스턴스로 반환됩니다.

INTEGER 또는 int

INTEGER 선호도를 사용하는 열의 비헤이비어는 선호도가 NUMERIC인 열과 같지만 한 가지 예외가 있습니다. 저장할 값이 부동 소수점 구성 요소가 없는 Number 인스턴스 등의 실수이거나 부동 소수점 구성 요소가 없는 실수로 변환될 수 있는 텍스트 값이면 값이 정수로 변환되고 INTEGER 저장소 클래스를 사용하여 저장됩니다. 부동 소수점 구성 요소가 있는 실수를 저장하려고 하면 오류가 발생합니다.

REAL 또는 NUMBER

선호도가 REAL 또는 NUMBER인 열의 비헤이비어는 선호도가 NUMERIC인 열과 같지만 정수 값을 부동 소수점 표현으로 나타낸다는 점이 다릅니다. REAL 열의 값은 데이터베이스에서 항상 Number 인스턴스로 반환됩니다.

BOOLEAN

선호도가 BOOLEAN인 열은 true 또는 false 값을 저장합니다. BOOLEAN 열은 ActionScript 또는 JavaScript Boolean 인스턴스인 값을 받아들입니다. 코드에서 String 값을 저장하려고 하면 길이가 0보다 큰 String은 true로, 빈 String은 false로 간주됩니다. 코드에서 숫자 데이터를 저장하려고 하면 0이 아닌 값은 true로, 0은 false로 간주됩니다. SELECT 문을 사용하여 검색한 부울 값은 Boolean 인스턴스로 반환됩니다. NULL이 아닌 값은 INTEGER 저장소 클래스(false는 0, true는 1)를 사용하여 저장되고 데이터를 검색할 때 Boolean 객체로 변환됩니다.

DATE

선호도가 DATE인 열은 날짜 및 시간 값을 저장합니다. DATE 열은 ActionScript 또는 JavaScript Date 인스턴스인 값을 받아들이도록 설계되었습니다. DATE 열에 String 값을 저장하려고 하면 런타임에서 값을 Julian 날짜로 변환하려고 합니다. 변환에 실패하면 오류가 발생합니다. 코드에서 Number, int 또는 uint 값을 저장하려고 하면 데이터의 유효성이 검사되지 않고 값이 유효한 Julian 날짜 값으로 간주됩니다. SELECT 문을 사용하여 검색한 DATE 값은 자동으로 Date 인스턴스로 변환됩니다. DATE 값은 REAL 저장소 클래스를 사용하여 Julian 날짜 값으로 저장되므로 정상적으로 정렬 및 비교할 수 있습니다.

XML 또는 XMLList

XML 또는 XMLLIST 선호도를 사용하는 열은 XML 구조를 저장합니다. 코드에서 SQLStatement 매개 변수를 사용하여 XML 열에 데이터를 저장하려고 하면 런타임에서 ActionScript XML() 또는 XMLList() 함수를 사용하여 값을 변환하고 유효성을 검사합니다. 값을 유효한 XML로 변환할 수 없으면 오류가 발생합니다. INSERT INTO (col1) VALUES ('Invalid XML (no closing tag)')와 같이 리터럴 SQL 텍스트 값이 사용된 데이터를 저장하려고 하면 값이 파싱되거나 유효성이 검사되지 않고 올바른 형식으로 간주됩니다. 잘못된 값이 저장된 경우 값을 검색하면 빈 XML 객체가 반환됩니다. XML 및 XMLList 데이터는 TEXT 저장소 클래스나 NULL 저장소 클래스를 사용하여 저장됩니다.

Object

선호도가 OBJECT인 열은 ActionScript 또는 JavaScript의 복잡한 객체를 저장합니다. 여기에는 Object 클래스 인스턴스뿐 아니라 Array 인스턴스 등의 Object 하위 클래스 인스턴스 및 사용자 정의 클래스 인스턴스가 포함됩니다. OBJECT 열 데이터는 AMF3 형식으로 직렬화되며 BLOB 저장소 클래스를 사용하여 저장됩니다. 값을 검색하면 값이 AMF3에서 비직렬화되어 저장된 클래스의 인스턴스로 반환됩니다. 특히 표시 객체를 비롯한 일부 ActionScript 클래스는 원래 데이터 유형의 인스턴스로 비직렬화할 수 없습니다. 사용자 정의 클래스 인스턴스를 저장하기 전에 flash.net.registerClassAlias() 메서드를 사용하거나 Flex에서 클래스 선언에 [RemoteObject] 메타데이터를 추가하여 클래스의 별칭을 등록해야 합니다. 또한 데이터를 검색하기 전에 클래스의 같은 별칭을 등록해야 합니다. 클래스를 근본적으로 비직렬화할 수 없거나 클래스 별칭이 누락되었거나 불일치하여 데이터를 제대로 비직렬화할 수 없으면 데이터가 익명 객체(Object 클래스 인스턴스)로 반환되며 해당 속성 및 값은 저장된 원래 인스턴스와 일치합니다.

NONE

선호도가 NONE인 열은 특정 저장소 클래스를 선호하지 않습니다. 데이터는 변환되지 않고 그대로 삽입됩니다.

선호도 결정

열의 선호도는 CREATE TABLE 문에서 열에 선언된 유형에 따라 결정됩니다. 유형을 결정할 때는 대/소문자를 구별하지 않으며 다음과 같은 규칙이 적용됩니다.

- 열의 데이터 유형에 "CHAR", "CLOB", "STRI" 또는 "TEXT" 문자열이 들어 있으면 해당 열의 선호도는 TEXT/STRING입니다. VARCHAR 유형에는 "CHAR" 문자열이 들어 있으므로 TEXT 선호도가 할당됩니다.
- 열의 데이터 유형에 "BLOB" 문자열이 들어 있거나 데이터 유형이 지정되지 않았으면 열의 선호도가 NONE입니다.
- 열의 데이터 유형에 "XMLL" 문자열이 들어 있으면 열의 선호도가 XMLLIST입니다.
- 데이터 유형이 "XML" 문자열이면 열의 선호도가 XML입니다.
- 데이터 유형에 "OBJE" 문자열이 들어 있으면 열의 선호도가 OBJECT입니다.
- 데이터 유형에 "BOOL" 문자열이 들어 있으면 열의 선호도가 BOOLEAN입니다.
- 데이터 유형에 "DATE" 문자열이 들어 있으면 열의 선호도가 DATE입니다.
- 데이터 유형에 "INT" 문자열("UINT" 포함)이 들어 있으면 INTEGER 선호도가 할당됩니다.
- 열의 데이터 유형에 "REAL", "NUMB", "FLOA" 또는 "DOUB" 문자열이 들어 있으면 열의 선호도가 REAL/NUMBER입니다.
- 어떠한 조건에도 맞지 않으면 선호도가 NUMERIC입니다.
- CREATE TABLE t AS SELECT... 문을 사용하여 테이블을 만든 경우 모든 열에 데이터 유형이 지정되지 않으며 NONE 선호도가 지정됩니다.

데이터 유형과 비교 연산자

이항 비교 연산자인 =, <, <=, >= 및 !=, 집합 멤버 테스트 연산자인 IN, 삼항 비교 연산자인 BETWEEN이 지원됩니다. 이러한 연산자에 대한 자세한 내용은 연산자를 참조하십시오.

비교 결과는 비교되는 두 값의 저장소 클래스에 따라 다릅니다. 두 값을 비교할 때는 다음과 같은 규칙이 적용됩니다.

- 저장소 클래스가 NULL인 값은 저장소 클래스가 NULL인 다른 값을 포함하여 다른 모든 값보다 작은 것으로 간주됩니다.
- INTEGER 또는 REAL 값은 모든 TEXT 또는 BLOB 값보다 작습니다. INTEGER 또는 REAL을 다른 INTEGER 또는 REAL과 비교하면 숫자 비교가 수행됩니다.
- TEXT 값은 BLOB 값보다 작습니다. 두 TEXT 값을 비교하면 이진 비교가 수행됩니다.
- 두 BLOB 값을 비교하면 항상 이진 비교를 사용하여 결과가 판단됩니다.

삼항 연산자 BETWEEN은 항상 해당 이항 표현식으로 다시 변환됩니다. 예를 들어 BETWEEN b AND c는 >= b AND a <= c로 다시 변환되며, 이때 표현식을 평가하는 데 필요한 각 비교에서 a에 서로 다른 선호도가 적용될 수도 있습니다.

a IN (SELECT b ....)와 같은 유형의 표현식은 앞에서 이항 비교에 대해 설명한 세 가지 규칙에 따라 a = b와 비슷한 방식으로 처리됩니다. 예를 들어 b가 열 값이고 a가 표현식이면 b의 선호도가 a에 적용된 후 비교됩니다. a IN (x, y, z) 표현식은 a = +x OR a = +y OR a = +z로 다시 변환됩니다. IN 연산자 오른쪽에 있는 값(이 예제에서 x, y 및 z 값)은 열 값인 경우에도 표현식으로 간주됩니다. IN 연산자의 왼쪽에 있는 값이 열이면 해당 열의 선호도가 사용됩니다. 값이 표현식이면 변환이 수행되지 않습니다.

COLLATE 절을 사용하는 방법도 비교에 영향을 줍니다. 자세한 내용은 COLLATE를 참조하십시오.

데이터 유형과 수학 연산자

지원되는 각 수학 연산자인 *, /, %, + 및 -에서 각 피연산자에 숫자 선호도가 적용된 후 표현식이 평가됩니다. NUMERIC 저장소 클래스로 변환할 수 없는 피연산자가 있으면 표현식이 NULL로 평가됩니다.

|| 결합 연산자를 사용하면 각 피연산자가 TEXT 저장소 클래스로 변환된 후 표현식이 평가됩니다. TEXT 저장소 클래스로 변환할 수 없는 피연산자가 있으면 표현식의 결과가 NULL입니다. 피연산자의 값이 NULL이거나 TEXT가 아닌 저장소 클래스가 들어 있는 BLOB이면 값을 변환할 수 없는 이러한 상황이 나타날 수 있습니다.

데이터 유형과 정렬

ORDER BY 절을 사용하여 값을 정렬하면 저장소 클래스가 NULL인 값이 가장 먼저 옵니다. 이후에는 숫자 순서에 따라 정렬된 INTEGER 및 REAL 값이 온 다음 이진 순서나 지정된 데이터 정렬(BINARY 또는 NOCASE)에 따른 TEXT 값이 옵니다. 마지막으로 이진 순서에 따라 BLOB 값이 나옵니다. 정렬하기 전에 저장소 클래스가 변환되지 않습니다.

데이터 유형과 그룹화

GROUP BY 절을 사용하여 값을 그룹화할 때 저장소 클래스가 서로 다른 값은 고유한 것으로 간주됩니다. 예외적으로 INTEGER와 REAL 값은 숫자가 같으면 동일한 것으로 간주됩니다. GROUP BY 절의 결과로 값에 선호도가 적용되지 않습니다.

데이터 유형과 복합 SELECT 문

복합 SELECT 연산자인 UNION, INTERSECT 및 EXCEPT는 값을 암시적으로 비교합니다. 이러한 비교를 수행하기 전에 각 값에 선호도가 적용될 수 있습니다. 이 선호도(적용된 경우)는 복합 SELECT 결과 집합의 단일 열에 반환될 수 있는 모든 값에 적용됩니다. 해당 위치에 다른 종류의 표현식이 아닌 열 값이 있는 첫 번째 구성 요소 SELECT 문에서 반환하는 열의 선호도가 적용됩니다. 지정된 복합 SELECT 열에서 열 값을 반환하는 구성 요소 SELECT 문이 없으면 해당 열의 값을 비교하기 전에 선호도가 적용되지 않습니다.

67장: SQL 자세한 오류 메시지, ID 및 인수

SQLError 클래스는 Adobe AIR 로컬 SQL 데이터베이스를 사용한 작업을 하는 동안에 발생할 수 있는 다양한 오류를 나타냅니다. 모든 예외에 대해 SQLError 인스턴스에는 영어 오류 메시지를 포함하는 details 속성이 있습니다. 또한 각 오류 메시지에는 해당 고유 식별자가 있으며 이는 SQLError 객체의 detailID 속성에서 제공됩니다. 응용 프로그램에서는 detailID 속성을 사용하여 특정 details 오류 메시지를 확인할 수 있습니다. 그런 다음 최종 사용자의 로캘 언어로 대체 텍스트를 제공할 수 있습니다. detailArguments 배열의 인수 값을 오류 메시지 문자열에서 적절한 위치에 대체할 수 있습니다. 이 기능은 오류의 details 속성 오류 메시지를 최종 사용자에게 특정 로캘로 직접 표시하는 응용 프로그램에 유용합니다.

다음 표에는 detailID 값과 해당 값에 연결된 영어 오류 메시지 텍스트를 나열하는 목록이 포함되어 있습니다. 메시지의 자리 표시자 텍스트는 런타임에서 detailArguments 값을 대체하는 위치를 나타냅니다. 이 목록은 SQL 데이터베이스 작업에서 발생할 수 있는 오류 메시지를 지역화하기 위한 소스로 사용할 수 있습니다.

SQLError detailID	자세한 영어 오류 메시지 및 매개 변수
1001	Connection closed.(연결이 닫혔습니다.)
1102	Database must be open to perform this operation.(이 작업을 수행하려면 데이터베이스가 열려 있어야 합니다.)
1003	%s [,|and %s] parameter name(s) found in parameters property but not in the SQL specified.(%s [|및 %s] 매개 변수 이름이 매개 변수 속성에 있지만 지정된 SQL에는 없습니다.)
1004	Mismatch in parameter count.(매개 변수 개수가 일치하지 않습니다.) Found %d in SQL specified and %d value(s) set in parameters property.(지정된 SQL에는 %d개가 있는데 매개 변수 속성에는 값 %d개가 설정되어 있습니다.) Expecting values for %s [,|and %s].(%s [|및 %s]에 대한 값이 필요합니다.)
1005	Auto compact could not be turned on.(자동 압축을 사용할 수 없습니다.)
1006	The pageSize value could not be set.(pageSize 값을 설정할 수 없습니다.)
1007	The schema object with name '%s' of type '%s' in database '%s' was not found.(스키마 객체(이름: '%s', 유형: '%s', 데이터베이스: '%s')를 찾을 수 없습니다.)
1008	The schema object with name '%s' in database '%s' was not found.(스키마 객체(이름: '%s', 데이터베이스: '%s')를 찾을 수 없습니다.)
1009	No schema objects with type '%s' in database '%s' were found.(스키마 객체(유형: '%s', 데이터베이스: '%s')를 찾을 수 없습니다.)
1010	No schema objects in database '%s' were found.(데이터베이스 '%s'에 스키마 객체가 없습니다.)
2001	Parser stack overflow.(파서 스택 오버플로)
2002	Too many arguments on function '%s'('%s' 함수에 인수가 너무 많습니다.)
2003	near '%s': syntax error('%s' 근처에 구문 오류가 있음)
2004	there is already another table or index with this name: '%s'(이름이 '%s'인 테이블 또는 인덱스가 이미 있습니다.)
2005	PRAGMA is not allowed in SQL.(SQL에서 PRAGMA를 사용할 수 없습니다.)
2006	Not a writable directory.(쓰기 가능한 디렉토리가 아닙니다.)
2007	Unknown or unsupported join type: '%s %s %s'(알 수 없거나 지원되지 않는 조인 유형: '%s %s %s')
2008	RIGHT and FULL OUTER JOINs are not currently supported.(RIGHT 및 FULL OUTER JOIN은 현재 지원되지 않습니다.)
2009	A NATURAL join may not have an ON or USING clause.(NATURAL 조인에는 ON 또는 USING 절이 있을 수 없습니다.)
2010	Cannot have both ON and USING clauses in the same join.(같은 조인에 ON과 USING 절이 모두 있을 수는 없습니다.)
2011	Cannot join using column '%s' - column not present in both tables.('%s' 열을 사용하여 조인할 수 없습니다. 두 테이블 중 최소한 하나에 해당 열이 없습니다.)
2012	Only a single result allowed for a SELECT that is part of an expression.(표현식의 일부인 SELECT에는 결과가 하나만 허용됩니다.)

2013	No such table: '[%s.]%s'('[%s.]%s' 테이블이 없습니다.)
2014	No tables specified.(테이블이 지정되지 않았습니다.)
2015	Too many columns in result set|too many columns on '%s'.(결과 집합에 열이 너무 많습니다. |'%s'에 열이 너무 많습니다.)
2016	%s ORDER|GROUP BY term out of range - should be between 1 and %d(%s ORDER|GROUP BY 항목이 범위를 벗어납니다. 1에서 %d 사이여야 합니다.)
2017	Too many terms in ORDER BY clause.(ORDER BY 절에 항목이 너무 많습니다.)
2018	%s ORDER BY term out of range - should be between 1 and %d.(%s ORDER BY 항목이 범위를 벗어납니다. 1에서 %d 사이여야 합니다.)
2019	%r ORDER BY term does not match any column in the result set.(%r ORDER BY 항목이 결과 집합의 어떠한 열과도 일치하지 않습니다.)
2020	ORDER BY clause should come after '%s' not before.(ORDER BY 절은 '%s'보다 앞이 아닌 뒤에 나와야 합니다.)
2021	LIMIT clause should come after '%s' not before.(LIMIT 절은 '%s'보다 앞이 아닌 뒤에 나와야 합니다.)
2022	SELECTs to the left and right of '%s' do not have the same number of result columns.('%s'의 왼쪽과 오른쪽에 있는 SELECT에서 결과 열 수가 다릅니다.)
2023	A GROUP BY clause is required before HAVING.(HAVING 앞에 GROUP BY 절이 필요합니다.)
2024	Aggregate functions are not allowed in the GROUP BY clause.(GROUP BY 절에는 집계 함수를 사용할 수 없습니다.)
2025	DISTINCT in aggregate must be followed by an expression.(집계에서 DISTINCT 뒤에 표현식이 나와야 합니다.)
2026	Too many terms in compound SELECT.(복합 SELECT에 항목이 너무 많습니다.)
2027	Too many terms in ORDER|GROUP BY clause(ORDER|GROUP BY 절에 항목이 너무 많습니다.)
2028	Temporary trigger may not have qualified name(임시 트리거는 정규화된 이름을 가질 수 없습니다.)
2030	Trigger '%s' already exists('%s' 트리거가 이미 있습니다.)
2032	Cannot create BEFORE|AFTER trigger on view: '%s'.('%s' 뷰에 BEFORE|AFTER 트리거를 만들 수 없습니다.)
2033	Cannot create INSTEAD OF trigger on table: '%s'.('%s' 테이블에 INSTEAD OF 트리거를 만들 수 없습니다.)
2034	No such trigger: '%s'('%s' 트리거가 없습니다.)
2035	Recursive triggers not supported ('%s').(재귀 트리거('%s')는 지원되지 않습니다.)
2036	No such column: %s[.(%s[열이 없습니다.)%s[.%s]]
2037	VACUUM is not allowed from SQL.(SQL에서 VACUUM을 사용할 수 없습니다.)
2043	Table '%s': indexing function returned an invalid plan.(테이블 '%s': 인덱싱 함수에서 잘못된 계획을 반환했습니다.)
2044	At most %d tables in a join.(조인의 테이블은 최대 %d개입니다.)
2046	Cannot add a PRIMARY KEY column.(PRIMARY KEY 열을 추가할 수 없습니다.)
2047	Cannot add a UNIQUE column.(UNIQUE 열을 추가할 수 없습니다.)
2048	Cannot add a NOT NULL column with default value NULL.(기본값이 NULL인 NOT NULL 열을 추가할 수 없습니다.)
2049	Cannot add a column with non-constant default.(기본값이 상수가 아닌 열을 추가할 수 없습니다.)
2050	Cannot add a column to a view.(뷰에 열을 추가할 수 없습니다.)
2051	ANALYZE is not allowed in SQL.(SQL에서 ANALYZE를 사용할 수 없습니다.)
2052	Invalid name: '%s'(잘못된 이름: '%s')
2053	ATTACH is not allowed from SQL.(SQL에서 ATTACH를 사용할 수 없습니다.)
2054	%s '%s' cannot reference objects in database '%s'(%s '%s'에서 '%s' 데이터베이스의 객체를 참조할 수 없습니다.)
2055	Access to '[%s.]%s.%s' is prohibited.('[%s.]%s.%s'에 대한 액세스가 금지되었습니다.)
2056	Not authorized.(권한이 없습니다.)
2058	No such view: '[%s.]%s'('[%s.]%s' 뷰가 없습니다.)
2060	Temporary table name must be unqualified.(임시 테이블 이름은 정규화되지 않아야 합니다.)

2061	Table '%s' already exists.('%s' 테이블이 이미 있습니다.)
2062	There is already an index named: '%s'(이름이 '%s'인 인덱스가 이미 있습니다.)
2064	Duplicate column name: '%s'(중복된 열 이름: '%s')
2065	Table '%s' has more than one primary key.(테이블 '%s'에 기본 키가 둘 이상 있습니다.)
2066	AUTOINCREMENT is only allowed on an INTEGER PRIMARY KEY(AUTOINCREMENT는 INTEGER PRIMARY KEY에만 사용할 수 있습니다.)
2067	No such collation sequence: '%s'('%s' 데이터 정렬 시퀀스가 없습니다.)
2068	Parameters are not allowed in views.(뷰에서 매개 변수를 사용할 수 없습니다.)
2069	View '%s' is circularly defined.('%s' 뷰가 순환 정의되었습니다.)
2070	Table '%s' may not be dropped.('%s' 테이블을 삭제할 수 없습니다.)
2071	Use DROP VIEW to delete view '%s'(DROP VIEW를 사용하여 '%s' 뷰를 삭제해야 합니다.)
2072	Use DROP TABLE to delete table '%s'(DROP TABLE을 사용하여 '%s' 테이블을 삭제해야 합니다.)
2073	Foreign key on '%s' should reference only one column of table '%s'('%s'의 외래 키는 '%s' 테이블의 열 중 하나만 참조해야 합니다.)
2074	Number of columns in foreign key does not match the number of columns in the referenced table.(외래 키의 열 수가 참조된 테이블의 열 수와 일치하지 않습니다.)
2075	Unknown column '%s' in foreign key definition.(외래 키 정의에서는 알 수 없는 '%s' 열입니다.)
2076	Table '%s' may not be indexed.('%s' 테이블을 인덱싱할 수 없습니다.)
2077	Views may not be indexed.(뷰를 인덱싱할 수 없습니다.)
2080	Conflicting ON CONFLICT clauses specified.(충돌하는 ON CONFLICT 절이 지정되었습니다.)
2081	No such index: '%s'('%s' 인덱스가 없습니다.)
2082	Index associated with UNIQUE or PRIMARY KEY constraint cannot be dropped.(UNIQUE 또는 PRIMARY KEY 제약 조건이 연결된 인덱스를 삭제할 수 없습니다.)
2083	BEGIN is not allowed in SQL.(SQL에서 BEGIN을 사용할 수 없습니다.)
2084	COMMIT is not allowed in SQL.(SQL에서 COMMIT을 사용할 수 없습니다.)
2085	ROLLBACK is not allowed in SQL.(SQL에서 ROLLBACK을 사용할 수 없습니다.)
2086	Unable to open a temporary database file for storing temporary tables.(임시 테이블을 저장하는 임시 데이터베이스 파일을 열 수 없습니다.)
2087	Unable to identify the object to be reindexed.(다시 인덱싱할 객체를 식별할 수 없습니다.)
2088	Table '%s' may not be modified.('%s' 테이블을 수정할 수 없습니다.)
2089	Cannot modify '%s' because it is a view.('%s'은(는) 뷰이므로 수정할 수 없습니다.)
2090	Variable number must be between ?0 and ?%d<(변수 번호는 ?0에서 ?%d< 사이여야 합니다.)
2092	Misuse of aliased aggregate '%s'(앨리어스된 집계 '%s'을(를) 잘못 사용했습니다.)
2093	Ambiguous column name: '[%s.[%s.]]%s'(모호한 열 이름: '[%s.[%s.]]%s')
2094	No such function: '%s'('%s' 함수가 없습니다.)
2095	Wrong number of arguments to function '%s'('%s' 함수에 대한 인수 개수가 잘못되었습니다.)
2096	Subqueries prohibited in CHECK constraints.(CHECK 제약 조건에 하위 쿼리를 사용할 수 없습니다.)
2097	Parameters prohibited in CHECK constraints.(CHECK 제약 조건에 매개 변수를 사용할 수 없습니다.)
2098	Expression tree is too large (maximum depth %d)(표현식 트리가 너무 큽니다(최대 깊이 %d).)
2099	RAISE() may only be used within a trigger-program(RAISE()는 트리거 프로그램 내에서만 사용할 수 있습니다.)
2100	Table '%s' has %d columns but %d values were supplied('%s' 테이블에 열이 %d개 있지만 값이 %d개 제공되었습니다.)
2101	Database schema is locked: '%s'('%s' 데이터베이스 스키마가 잠겼습니다.)
2102	Statement too long.(명령문이 너무 깁니다.)
2103	Unable to delete/modify collation sequence due to active statements(활성 명령문으로 인해 데이터 정렬 시퀀스를 삭제/수정할 수 없습니다.)

2104	Too many attached databases - max %d(연결된 데이터베이스가 너무 많습니다(최대 %d개).)
2105	Cannot ATTACH database within transaction.(트랜잭션 내에서 데이터베이스를 ATTACH할 수 없습니다.)
2106	Database '%s' is already in use.('%s' 데이터베이스가 이미 사용 중입니다.)
2108	Attached databases must use the same text encoding as main database.(연결된 데이터베이스에서는 주 데이터베이스와 같은 텍스트 인코딩을 사용해야 합니다.)
2200	Out of memory.(메모리가 부족합니다.)
2201	Unable to open database.(데이터베이스를 열 수 없습니다.)
2202	Cannot DETACH database within transaction.(트랜잭션 내에서 데이터베이스를 DETACH할 수 없습니다.)
2203	Cannot detach database: '%s'('%s' 데이터베이스를 분리할 수 없습니다.)
2204	Database '%s' is locked.('%s' 데이터베이스가 잠겼습니다.)
2205	Unable to acquire a read lock on the database.(데이터베이스에 대한 읽기 잠금을 수행할 수 없습니다.)
2206	[column|columns] '%s'[,'%s'] are not [unique|is] not unique.('%s'[,'%s'] [열]이 고유하지 않습니다.)
2207	Malformed database schema.(데이터베이스 스키마의 형식이 잘못되었습니다.)
2208	Unsupported file format.(지원되지 않는 파일 형식입니다.)
2209	Unrecognized token: '%s'(인식할 수 없는 토큰: '%s')
2300	Could not convert text value to numeric value.(텍스트 값을 숫자 값으로 변환할 수 없습니다.)
2301	Could not convert string value to date.(문자열 값을 날짜로 변환할 수 없습니다.)
2302	Could not convert floating point value to integer without loss of data.(부동 소수점 값을 데이터 손실 없이 정수로 변환할 수 없습니다.)
2303	Cannot rollback transaction - SQL statements in progress.(트랜잭션을 롤백할 수 없습니다. SQL 명령문이 실행 중입니다.)
2304	Cannot commit transaction - SQL statements in progress.(트랜잭션을 커밋할 수 없습니다. SQL 명령문이 실행 중입니다.)
2305	Database table is locked: '%s'('%s' 데이터베이스 테이블이 잠겼습니다.)
2306	Read-only table.(읽기 전용 테이블입니다.)
2307	String or blob too big.(문자열 또는 BLOB이 너무 큽니다.)
2309	Cannot open indexed column for writing.(인덱싱된 열을 쓰기용으로 열 수 없습니다.)
2400	Cannot open value of type %s.(%s 유형의 값을 열 수 없습니다.)
2401	No such rowid: %s<(%s< rowid가 없습니다.)
2402	Object name reserved for internal use: '%s'('%s' 객체 이름은 내부적으로 사용하도록 예약되었습니다.)
2403	View '%s' may not be altered.('%s' 뷰를 변경할 수 없습니다.)
2404	Default value of column '%s' is not constant.('%s' 열의 기본값이 상수가 아닙니다.)
2405	Not authorized to use function '%s'('%s' 함수를 사용할 권한이 없습니다.)
2406	Misuse of aggregate function '%s'('%s' 집계 함수를 잘못 사용했습니다.)
2407	Misuse of aggregate: '%s'('%s' 집계를 잘못 사용했습니다.)
2408	No such database: '%s'('%s' 데이터베이스가 없습니다.)
2409	Table '%s' has no column named '%s'('%s' 테이블에 이름이 '%s'인 열이 없습니다.)
2501	No such module: '%s'('%s' 모듈이 없습니다.)
2508	No such savepoint: '%s'('%s' 저장점이 없습니다.)
2510	Cannot rollback - no transaction is active.(롤백할 수 없습니다. 활성 트랜잭션이 없습니다.)
2511	Cannot commit - no transaction is active.(커밋할 수 없습니다. 활성 트랜잭션이 없습니다.)

68장: AGAL(Adobe Graphics Assembly Language)

AGAL(Adobe Graphics Assembly Language)은 정점 및 조각 렌더링 프로그램을 정의하기 위한 세이더 언어입니다. AGAL 프로그램은 이 문서에 설명된 이진 바이트코드 형식으로 렌더링 컨텍스트에 업로드해야 합니다.

AGAL 바이트코드 형식

AGAL 바이트코드는 Endian.LITTLE_ENDIAN 형식을 사용해야 합니다.

바이트코드 헤더

AGAL 바이트코드는 다음과 같이 7바이트 헤더로 시작해야 합니다.

```
A0 01000000 A1 00 -- for a vertex program
A0 01000000 A1 01 -- for a fragment program
```

오프셋(바이트)	크기(바이트)	이름	설명
0	1	magic	0xa0이어야 함
1	4	version	1이어야 함
5	1	shader type ID	0xa1이어야 함
6	1	shader type	정점 프로그램의 경우 0, 조각 프로그램의 경우 1

토큰

헤더 바로 다음에 개수에 상관없이 토큰이 이어집니다. 모든 토큰의 크기는 192비트(24바이트)이고 형식은 항상 다음과 같습니다.

[opcode][destination][source1][source2 또는 sampler]

특정 opcode에서는 이러한 필드 중 일부를 사용하지 않습니다. 사용되지 않는 필드는 0으로 설정되어야 합니다.

작업 코드

[opcode] 필드의 크기는 32비트이고 다음 값 중 하나일 수 있습니다.

이름	Opcode	작업	설명
mov	0x00	이동	데이터를 source1에서 destination으로 이동, 구성 요소 전체
add	0x01	더하기	destination = source1 + source2, 구성 요소 전체
sub	0x02	빼기	destination = source1 - source2, 구성 요소 전체
mul	0x03	곱하기	destination = source1 * source2, 구성 요소 전체
div	0x04	나누기	destination = source1 / source2, 구성 요소 전체
rcp	0x05	역	destination = 1/source1, 구성 요소 전체
min	0x06	최소	destination = minimum(source1,source2), 구성 요소 전체

이름	Opcode	작업	설명
max	0x07	최대	destination = maximum(source1,source2), 구성 요소 전체
frc	0x08	소수	destination = source1 - (float)floor(source1), 구성 요소 전체
sqt	0x09	제곱근	destination = sqrt(source1), 구성 요소 전체
rsq	0x0a	역의 근	destination = 1/sqrt(source1), 구성 요소 전체
pow	0x0b	멱	destination = pow(source1,source2), 구성 요소 전체
log	0x0c	로그	destination = log_2(source1), 구성 요소 전체
exp	0x0d	지수	destination = 2^source1, 구성 요소 전체
nrm	0x0e	표준화	destination = normalize(source1), 구성 요소 전체(3요소 결과만 산출하며, 대상은 .xyz 이하로 마스크 처리되어야 함)
sin	0x0f	사인	destination = sin(source1), 구성 요소 전체
cos	0x10	코사인	destination = cos(source1), 구성 요소 전체
crs	0x11	외적	destination.x = source1.y * source2.z - source1.z * source2.y destination.y = source1.z * source2.x - source1.x * source2.z destination.z = source1.x * source2.y - source1.y * source2.x (3요소 결과만 산출하며, 대상은 .xyz 이하로 마스크 처리되어야 함)
dp3	0x12	내적	destination = source1.x*source2.x + source1.y*source2.y + source1.z*source2.z
dp4	0x13	내적	destination = source1.x*source2.x + source1.y*source2.y + source1.z*source2.z + source1.w*source2.w
abs	0x14	절대	destination = abs(source1), 구성 요소 전체
neg	0x15	무효화	destination = -source1, 구성 요소 전체
sat	0x16	포화	destination = maximum(minimum(source1,1),0), 구성 요소 전체
m33	0x17	곱하기 행렬 3x3	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) (3요소 결과만 산출하며, 대상은 .xyz 이하로 마스크 처리되어야 함)

이름	Opcode	작업	설명
m44	0x18	곱하기 행렬 4x4	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) + (source1.w * source2[0].w) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) + (source1.w * source2[1].w) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) + (source1.w * source2[2].w) destination.w = (source1.x * source2[3].x) + (source1.y * source2[3].y) + (source1.z * source2[3].z) + (source1.w * source2[3].w)
m34	0x19	곱하기 행렬 3x4	destination.x = (source1.x * source2[0].x) + (source1.y * source2[0].y) + (source1.z * source2[0].z) + (source1.w * source2[0].w) destination.y = (source1.x * source2[1].x) + (source1.y * source2[1].y) + (source1.z * source2[1].z) + (source1.w * source2[1].w) destination.z = (source1.x * source2[2].x) + (source1.y * source2[2].y) + (source1.z * source2[2].z) + (source1.w * source2[2].w) (3요소 결과만 산출하며, 대상은 .xyz 이하로 마스크 처리되어야 함)
kil	0x27	삭제(조각 셰이더만 해당)	단일 스칼라 값의 소스 구성 요소가 0보다 작은 경우 조각은 삭제되고 프레임 버퍼에 그려지지 않습니다. (대상 레지스터는 모두 0으로 설정되어야 함)
tex	0x28	텍스처 샘플(조각 셰이더만 해당)	destination은 source1 좌표에서 source2 텍스처의 로드와 동일합니다. 이 경우 source2는 샘플러 형식이어야 합니다.
sge	0x29	같거나 큰 경우 설정	destination = source1 >= source2 ? 1 : 0, 구성 요소 전체
slt	0x2a	작은 경우 설정	destination = source1 < source2 ? 1 : 0, 구성 요소 전체
seq	0x2c	같은 경우 설정	destination = source1 == source2 ? 1 : 0, 구성 요소 전체
sne	0x2d	같지 않은 경우 설정	destination = source1 != source2 ? 1 : 0, 구성 요소 전체

Destination 필드 형식

[destination] 필드의 크기는 32비트:

```
31.............................0
----TTTT----MMMMNNNNNNNNNNNNNNNN
```

T = 레지스터 유형(4비트)

M = 쓰기 마스크(4비트)

N = 레지스터 번호(16비트)

- = 정의되지 않음, 0이어야 함

Source 필드 형식

[source] 필드의 크기는 64비트:

```
63.........................................................0
D------------QQ----IIII----TTTTSSSSSSSSOOOOOOOONNNNNNNNNNNNNNNN
```

D = 직접=0/간접=1(직접 Q 및 I는 무시됨), 1비트

Q = 인덱스 레지스터 구성 요소 선택(2비트)

I = 인덱스 레지스터 유형(4비트)

T = 레지스터 유형(4비트)

S = Swizzle(8비트, 구성 요소당 2비트)

O = 간접 오프셋(8비트)

N = 레지스터 번호(16비트)

- = 정의되지 않음, 0이어야 함

Sampler 필드 형식

tex opcode의 두 번째 source 필드는 64비트 크기의 [sampler] 형식이어야 함:

```
63.........................................................0
FFFFMMMMWWWWSSSSDDDD--------TTTT--------BBBBBBBBNNNNNNNNNNNNNNNN
```

N = Sampler 레지스터 번호(16비트)

B = 텍스처 LOD(Level-of-Detail) 편차, 부호 있는 정수, 배율: 8. 사용되는 부동 소수점 값은 b/8.0(8비트)입니다.

T = 레지스터 유형, 5여야 함, Sampler(4비트)

F = 필터(0=가장 가까움, 1=선형)(4비트)

M = 밉맵(0=사용 안 함, 1=가장 가까움, 2=선형)

W = 래핑(0=클램프, 1=반복)

S = 특수 플래그 비트(0이어야 함)

D = 차원(0=2D, 1=정육면체)

프로그램 레지스터

다음과 같은 레지스터 유형이 정의되어 있습니다. 나열된 값을 사용하여 토큰의 필드에 레지스터 유형을 지정하십시오.

이름	값	조각 프로그램당 수	정점 프로그램당 수	사용
특성	0	해당 사항 없음	8	정점 셰이더 입력, Context3D.setVertexBufferAt()을 사용하여 지정된 정점 버퍼에서 읽어옴
상수	1	28	128	셰이더 입력, Context3D.setProgramConstants() 함수 모음을 사용하여 설정
임시	2	8	8	계산을 위한 임시 레지스터, 프로그램 외부에서 액세스할 수 없음

이름	값	조각 프로그램당 수	정점 프로그램당 수	사용
출력	3	1	1	셰이더 출력: 정점 프로그램에서는 출력이 클립 공간 위치에 있고, 조각 프로그램에서는 출력이 색상임
변형	4	8	8	정점 셰이더와 조각 셰이더 간에 보간 데이터를 전송합니다. 정점 프로그램의 가변 레지스터는 조각 프로그램에 입력으로 적용됩니다. 값은 삼각형 꼭지점으로부터의 거리에 따라 보간됩니다.
샘플러	5	8	해당 사항 없음	조각 셰이더 입력, Context3D.setTextureAt()을 사용하여 지정된 텍스처에서 읽어옴