ADOBE® AIR® 응용 프로그램 만들기

마지막 업데이트 2010 년 7 월 12 일

Adobe® AIR® 응용 프로그램 만들기

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

컨텐츠

1장: Adobe AIR 소개

Adboe® AIR®은 크로스 운영 체제 런타임으로, 기존 웹 개발 기술을 활용하여 RIA(Rich Internet Application)를 구축하고 데스크톱 및 휴대 장치에 배포할 수 있도록 합니다. AIR 응용 프로그램은 Adobe® Flex 및 Adobe® Flash®(SWF 기반)를 사용하는 ActionScript 3.0으로 작성할 수 있으며 HTML, JavaScript® 및 Ajax(HTML 기반)로도 작성할 수 있습니다. AIR 응용 프로그램을 작성하는 데 사용할 수 있는 Adobe Flash Platform 도구에 대한 자세한 내용은 16페이지의 "AIR 개발용 Adobe Flash Platform 도구"를 참조하십시오.

Adobe AIR Developer Connection(http://www.adobe.com/devnet/air/)에서 Adobe AIR를 시작하고 사용하는 방법에 대한 자세한 내용을 참조할 수 있습니다.

AIR를 사용하면 친숙한 환경에서 도구 및 응용 프로그램을 가장 편안한 방식으로 활용하면서 작업할 수 있습니다. Flash, Flex, HTML, JavaScript 및 Ajax를 지원하므로 사용자 요구에 가장 적합한 환경을 구축할 수 있습니다.

예를 들어, 다음 기술 중 하나 이상을 사용하여 응용 프로그램을 개발할 수 있습니다.

- Flash / Flex / ActionScript
- HTML / JavaScript / CSS / Ajax
- 모든 응용 프로그램에서 활용할 수 있는 PDF

따라서 AIR 응용 프로그램은 다음과 같을 수 있습니다.

- Flash 또는 Flex 기반: Flash/Flex(SWF)가 루트 내용인 응용 프로그램
- HTML 또는 PDF가 포함된 Flash 또는 Flex 기반, HTML(HTML, JS, CSS) 또는 PDF 내용이 포함된 Flash/Flex(SWF)가 루트 내용인 응용 프로그램
- HTML 기반, HTML, JS, CSS가 루트 내용인 응용 프로그램
- Flash/Flex 또는 PDF가 포함된 HTML 기반, Flash/Flex(SWF) 또는 PDF 내용이 포함된 HTML이 루트 내용인 응용 프로그램

사용자는 기본 데스크톱 응용 프로그램과 상호 작용하는 방식과 동일하게 AIR 응용 프로그램과 상호 작용합니다. 런타임이 사용자의 컴퓨터에 설치되면 AIR 응용 프로그램이 설치되고 다른 데스크톱 응용 프로그램과 마찬가지로 실행됩니다.

런타임은 응용 프로그램을 배포하기 위한 일관된 크로스 운영 체제 플랫폼과 프레임워크를 제공하므로 데스크톱 전반에서 일관된 기능과 상호 작용을 보장하여 크로스 브라우저 테스트를 제거합니다. 특정 운영 체제용으로 개발하는 대신 런타임을 대상으로 개발할 수 있으므로 다음과 같은 이점이 있습니다.

- AIR용으로 개발된 응용 프로그램이 추가 작업 없이 여러 운영 체제에서 실행됩니다. 런타임은 AIR에서 지원하는 모든 운영 체제에서 일관되고 예측 가능한 프리젠테이션과 상호 작용을 보장합니다.
- 기존 웹 기술과 디자인 패턴을 활용할 수 있어 응용 프로그램을 빠르게 작성할 수 있습니다. 기존 데스크톱 개발 기술이나 복잡한 기본 코드에 대해 모르더라도 웹 기반 응용 프로그램을 데스크톱으로 확장할 수 있습니다.
- 응용 프로그램 개발이 C 및 C++와 같은 하위 수준 언어를 사용하는 경우보다 쉽습니다. 각 운영 체제와 관련된 복잡한 하위 수준 AP를 관리할 필요가 없습니다.

AIR용 응용 프로그램을 개발할 때 프레임워크와 API의 다양한 집합을 활용할 수 있습니다.

- AIR 프레임워크와 런타임에서 제공하는 AIR와 관련된 API
- Flex 프레임워크와 SWF 파일에서 사용되는 ActionScript API 및 다른 ActionScript 기반 라이브러리와 프레임워크
- HTML, CSS 및 JavaScript
- 대부분의 Ajax 프레임워크

AIR는 응용 프로그램을 만들고 배포하고 사용하는 방법을 동적으로 변경합니다. 더욱 창의적으로 제어할 수 있으며 기존 데스크톱 개발 기술을 배우지 않고도 Flash, Flex, HTML 및 Ajax 기반 응용 프로그램을 데스크톱으로 확장할 수 있습니다.

AIR 1.1의 새로운 기능

Adobe AIR 1.1에 도입된 새로운 기능은 다음과 같습니다.

- 설치 및 기타 런타임 대화 상자가 다음 언어로 번역되었습니다.
  - 포르투갈어(브라질)
  - 중국어(번체 및 간체)
  - 프랑스어
  - 독일어
  - 이탈리아어
  - 일본어
  - 한국어
  - 러시아어
  - 프랑스어
  - 스페인어
- 2바이트 언어의 키보드 입력을 비롯한 국가별 응용 프로그램 구축이 지원됩니다. 124페이지의 “AIR 응용 프로그램 지역화”를 참조하십시오.
  - 응용 프로그램 설명자 파일의 이름 및 설명자 특성 지역화가 지원됩니다.
  - SQLite 데이터베이스의 SQLError.detailID 및 SQLError.detailArguments 같은 오류 메시지 지역화가 지원됩니다.
  - 운영 체제에 설정한 기본 설정 UI 언어 배열을 가져오기 위한 Capabilities.languages 속성이 추가되었습니다.
  - HTML 버튼 레이블과 기본 메뉴(컨텍스트 메뉴 및 Mac 메뉴 모음 등)가 지원되는 모든 언어로 지역화되었습니다.
- 자체 서명된 응용 프로그램에서 인증 기관(CA) 인증서에 연결된 응용 프로그램으로 인증서 마이그레이션이 지원됩니다.
- Microsoft Windows XP Tablet PC Edition과 Windows Vista® Home Premium, Business, Ultimate, Enterprise 64비트 버전이 지원됩니다.
- 디스크에서 사용할 수 있는 디스크 공간을 가져오기 위한 File.spaceAvailable API가 추가되었습니다.
- 현재 운영 체제에서 투명 윈도우를 지원하는지 여부를 확인하기 위한 NativeWindow.supportsTransparency 속성이 추가되었습니다.

AIR 1.1 릴리스에 대한 자세한 내용은 Adobe AIR 1.1 릴리스 정보(http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

AIR 1.5의 새로운 기능

Adobe AIR 1.5에 도입된 새로운 기능은 다음과 같습니다.

- Flash Player 10의 다음 기능이 지원됩니다.
  - 사용자 정의 필터 및 효과
  - 향상된 드로잉 API

- 동적 사운드 생성
- 벡터 데이터 유형
- 향상된 파일 업로드 및 다운로드 API
- RTMFP(Real Time Media Flow Protocol)
- 3D 효과
- 고급 텍스트 지원
- 색상 관리
- 텍스트 엔진
- 동적 스트리밍
- Speex 오디오 코덱

이러한 기능에 대한 자세한 내용은 http://www.adobe.com/kr/products/flashplayer/features/를 참조하십시오.

새 Flash Player 10 ActionScript 3.0 API 사용법에 대한 자세한 내용은 Adobe ActionScript 3.0 개발자 안내서 및 Adobe ActionScript 3.0 Reference for the Adobe Flash Platform을 참조하십시오.

- AIR 1.5 설치 프로그램 및 기타 런타임 대화 상자에서 지원하는 추가 언어:
  - 체코어
  - 네덜란드어
  - 스웨덴어
  - 터키어
  - 폴란드어
- 데이터베이스 암호화

  AIR 1.5에서는 데이터베이스 파일을 암호화할 수 있습니다. 메타데이터를 포함한 모든 데이터베이스 내용을 암호화하여 암호화한 AIR 응용 프로그램에서만 읽을 수 있습니다. 이 기능으로 개발자는 데이터베이스 파일을 암호화, 해독 및 재암호화할 수 있습니다. 암호화된 로컬 저장소(ActionScript 개발자용) 또는 Encrypted local storage(HTML 개발자용)를 참조하십시오.
- Adobe AIR에서 사용하는 WebKit 버전이 업데이트되어 SquirrelFish JavaScript 인터프리터가 지원됩니다.
- 새로운 XML 서명 유효성 검사 API를 사용하면 데이터 또는 정보의 무결성 및 서명자 신원을 확인할 수 있습니다. 자세한 내용은 AIR의 XML 서명 유효성 검사(ActionScript 개발자용) 또는 XML signature validation in AIR(HTML 개발자용)을 참조하십시오.

AIR 1.5 릴리스에 대한 자세한 내용은 Adobe AIR 1.5 릴리스 정보(http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

AIR 1.5.1의 새로운 기능

Adobe AIR 1.5.1에 도입된 새로운 기능은 다음과 같습니다.

최신 Flash Player 플러그인

AIR 1.5.1에는 HTML 내부에서 SWF 파일을 표시하는 데 사용되는 Flash Player 플러그인의 업데이트된 버전(10.0.22)이 포함되어 있습니다. 자세한 내용은 http://www.adobe.com/support/documentation/kr/flashplayer/releasenotes.html을 참조하십시오.

새 API

InvokeEvent.reason

InvokeEvent 이벤트의 이 새 속성은 응용 프로그램을 사용자가 시작했는지 아니면 로그인 시에 자동으로 시작되었는지를 나타냅니다. InvokeEventReason 클래스(flash.desktop 패키지에 있음)는 InvokeEvent.reason 속성에 사용할 수 있는 두 가지 문자열 값을 정의합니다. InvokeEventReason.LOGIN은 로그인 시에 실행된 경우를 정의하고, InvokeEventReason.STANDARD는 표준 시작을 정의합니다.

Capabilities.cpuArchitecture

이 새 속성은 컴퓨터의 프로세서 아키텍처를 문자열 반환합니다(예: "PowerPC", "x86").

새 AIR 1.5.1 API의 이점을 활용하려면 응용 프로그램 설명자를 1.5.1 네임스페이스를 사용하도록 업데이트하십시오.

xmlns="http://ns.adobe.com/air/application/1.5.1"

새 API를 사용할 필요가 없는 경우에는 응용 프로그램 설명자를 업데이트하지 않아도 됩니다. 사용자가 시스템에 설치된 런타임 버전을 업데이트한 경우 응용 프로그램이 AIR 1.5.1로 실행될 수 있습니다.

AIR 1.5.1 릴리스에 대한 자세한 내용은 Adobe AIR 1.5.1 릴리스 정보(http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

AIR 1.5.2의 새로운 기능

Adobe AIR 1.5.2에 도입된 새로운 기능은 다음과 같습니다.

새 API

Capabilities.supports32BitProcesses 및 Capabilities.supports64BitProcesses

이 속성은 시스템이 64비트 프로세스를 지원하는지 아니면 32비트 프로세스를 지원하는지를 나타냅니다.

LocalConnection.isPerUser

이 속성은 LocalConnection 객체의 범위가 현재 사용자(true)인지 아니면 컴퓨터의 모든 사용자가 액세스할 수 있는 전역(false)인지를 나타냅니다. 이 속성은 Mac OS에서 실행되는 내용에만 해당되며 다른 플랫폼에서는 이 매개 변수가 무시됩니다. 예를 들어, Windows와 Linux의 로컬 연결은 항상 사용자 단위입니다. 이전 버전에서는 Mac OS의 모든 LocalConnection 객체가 전역 범위를 가졌습니다. 이전 버전과의 호환성을 유지해야 하는 경우가 아니라면 보안을 위해 이 속성을 항상 true로 설정해야 합니다. 이후 버전에서는 이 속성의 기본값이 true가 될 것입니다.

System.disposeXML()

이 정적 메서드는 ActionScript XML 객체를 가비지 수집에 즉시 사용할 수 있게 만듭니다. 이 메서드는 지정된 XML 객체의 모든 노드에서 부모와 자식 간의 연결을 제거합니다. 이 메서드에는 XML 객체를 가비지 수집에 사용할 수 있게 만들 것이지 여부를 지정하는 매개 변수 하나만 사용됩니다. 이 메서드를 사용하면 XML 객체와 연결된 메모리를 효율적으로 처리할 수 있습니다.

응용 프로그램 설명자 파일 업데이트

새 AIR 1.5.2 API와 비헤이비어에 액세스하려면 응용 프로그램 설명자 파일을 1.5.2 네임스페이스로 업데이트하십시오. 네임스페이스를 업데이트하려면 xmlns 특성을 다음과 같이 변경합니다.

xmlns="http://ns.adobe.com/air/application/1.5.2"

AIR 1.5.2 릴리스에 대한 자세한 내용은 Adobe AIR 1.5.2 릴리스 정보(http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

AIR 1.5.3의 새로운 기능

이 버전에는 업데이트된 버전의 Flash Player, 보안 업데이트 및 몇 가지 버그 해결이 포함되어 있습니다. 개발자 릴리스 정보에는 인증서 갱신을 위한 새 프로세스를 비롯하여 AIR 응용 프로그램을 제작하는 모든 개발자가 읽어야 하는 중요 정보가 포함되어 있습니다.

AIR 1.5.3 릴리스에 대한 자세한 내용은 Adobe AIR 1.5.3 릴리스 정보(http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

AIR 2의 새로운 기능

AIR 2 릴리스에 대한 자세한 내용은 Adobe AIR 2 릴리스 정보(http://www.adobe.com/go/learn_air_relnotes_kr)를 참조하십시오.

2장: Adobe AIR 설치

Adobe® AIR®을 통해 데스크톱에서 AIR 응용 프로그램을 실행할 수 있습니다. 다음과 같은 방법으로 런타임을 설치할 수 있습니다.

- AIR 응용 프로그램 설치 없이 런타임을 개별적으로 설치합니다.
- 처음으로 웹 페이지 설치 "badge"를 통해 AIR 응용 프로그램을 설치합니다(런타임을 설치하라는 메시지가 표시됨).
- AIR SDK, Adobe® Flash® Builder™ 또는 Adobe Flex® SDK(AIR 명령줄 개발 도구 포함)와 같은 AIR 개발 환경을 설정합니다. SDK에 포함된 런타임은 응용 프로그램을 디버깅할 때만 사용되며, 설치된 AIR 응용 프로그램을 실행하는 데에는 사용되지 않습니다.

AIR 설치 및 AIR 응용 프로그램 실행을 위한 시스템 요구 사항은 Adobe AIR: 시스템 요구 사항 (http://www.adobe.com/kr/products/air/systemreqs/)을 참조하십시오.

런타임 설치 프로그램 및 AIR 응용 프로그램 설치 프로그램은 모두 AIR 응용 프로그램 또는 AIR 런타임 자체를 설치, 업데이트 또는 제거할 때 로그 파일을 만듭니다. 이러한 로그를 참조하여 설치 문제의 원인을 확인할 수 있습니다. 83페이지의 “설치 로그”를 참조하십시오.

Adobe AIR 설치

다음 지침에 따라 Windows, Mac OS X 및 Linux 버전의 AIR를 다운로드하고 설치합니다.

런타임을 설치하거나 업데이트하려면 사용자에게 컴퓨터에 대한 관리 권한이 필요합니다.

Windows 컴퓨터에 런타임 설치

1. http://get.adobe.com/kr/air/에서 런타임 설치 파일을 다운로드합니다.
2. 런타임 설치 파일을 두 번 클릭합니다.
3. 설치 윈도우에서 프롬프트에 따라 설치를 완료합니다.

Mac 컴퓨터에 런타임 설치

1. http://get.adobe.com/kr/air/에서 런타임 설치 파일을 다운로드합니다.
2. 런타임 설치 파일을 두 번 클릭합니다.
3. 설치 윈도우에서 프롬프트에 따라 설치를 완료합니다.
4. 설치 프로그램에서 인증 윈도우를 표시하면 해당 Mac OS 사용자 이름과 암호를 입력합니다.

Linux 컴퓨터에서 런타임 설치

이진 설치 프로그램 사용:

1. http://get.adobe.com/kr/air/에서 설치 이진 파일을 다운로드합니다.
2. 설치 프로그램을 실행할 수 있도록 파일 권한을 설정합니다. 명령줄에서 다음을 사용하여 파일 사용 권한을 설정할 수 있습니다.

   ```
   chmod +x AdobeAIRInstaller.bin
   ```

   일부 버전의 Linux에서는 컨텍스트 메뉴를 통해 여는 [속성] 대화 상자에서 파일 권한을 설정할 수 있습니다.
3. 런타임 설치 파일을 두 번 클릭하거나 명령줄에서 설치 프로그램을 실행합니다.

4 설치 윈도우에서 프롬프트에 따라 설치를 완료합니다.

Adobe AIR이 기본 패키지로 설치됩니다. 즉, RPM 기반 배포에서는 rpm으로, Debian 배포에서는 deb로 설치됩니다. 현재 AIR에서 다른 패키지 형식은 지원하지 않습니다.

패키지 설치 프로그램 사용:

1 http://get.adobe.com/kr/air/에서 AIR 패키지 파일을 다운로드합니다. 시스템에서 지원하는 패키지 형식에 따라 RPM 또는 Debian 패키지를 선택합니다.

2 필요한 경우 AIR 패키지 파일을 두 번 클릭하여 패키지를 설치합니다.

또한 명령줄에서 다음과 같이 설치할 수 있습니다.

a Debian 시스템:

```
sudo dpkg -i <path to the package>/adobeair-2.0.0.xxxxx.deb
```

b RPM 기반 시스템:

```
sudo rpm -i <path to the package>/adobeair-2.0.0-xxxxx.i386.rpm
```

또는 기존 버전(AIR 1.5.3 이상)을 업데이트하는 경우:

```
sudo rpm -U <path to the package>/adobeair-2.0.0-xxxxx.i386.rpm
```

AIR 2 및 AIR 응용 프로그램을 설치하려면 컴퓨터에 대한 관리자 권한이 있어야 합니다.

Adobe AIR은 /opt/Adobe AIR/Versions/1.0에 설치됩니다.

AIR은 MIME 형식 "application/vnd.adobe.air-application-installer-package+zip"을 등록하는데, 이는 .air 파일이 MIME 형식이며 AIR 런타임에 등록된다는 것을 나타냅니다.

Adobe AIR 제거

런타임을 설치하면 다음 절차에 따라 제거할 수 있습니다.

Windows 컴퓨터에서 런타임 제거

1 Windows [시작] 메뉴에서 [설정] > [제어판]을 선택합니다.

2 [프로그램 추가/제거] 제어판을 선택합니다.

3 "Adobe AIR"을 선택하여 런타임을 제거합니다.

4 [변경/제거] 버튼을 클릭합니다.

Mac 컴퓨터에서 런타임 제거

- 응용 프로그램/유틸리티 폴더에서 "Adobe AIR Uninstaller"를 두 번 클릭합니다.

Linux 컴퓨터에서 런타임 제거

다음 중 하나를 수행합니다.

- [응용 프로그램] 메뉴에서 "Adobe AIR Uninstaller" 명령을 선택합니다.
- -uninstall 옵션을 사용하여 AIR 설치 프로그램 이진을 실행합니다.
- 패키지 관리자를 사용하여 AIR 패키지(adobeair***v.n*** 및 adobecerts)를 제거합니다.

AIR 샘플 응용 프로그램 설치 및 실행

AIR 응용 프로그램을 설치하거나 업데이트하려면 컴퓨터에 대한 관리자 권한이 있어야 합니다.

AIR 기능을 보여 주는 일부 샘플 응용 프로그램을 사용할 수 있습니다. 다음 지침에 따라 이러한 응용 프로그램에 액세스하고 설치할 수 있습니다.

1 AIR 샘플 응용 프로그램을 다운로드하고 실행합니다. 컴파일된 응용 프로그램과 소스 코드를 사용할 수 있습니다.

2 샘플 응용 프로그램을 다운로드하고 실행하려면 샘플 응용 프로그램의 "Install Now" 버튼을 클릭합니다. 응용 프로그램을 설치하고 실행하라는 프롬프트가 표시됩니다.

3 샘플 응용 프로그램을 다운로드하고 나중에 실행하도록 선택하는 경우 다운로드 링크를 선택합니다. 다음 방법을 사용하여 언제든지 AIR 응용 프로그램을 실행할 수 있습니다.

- Windows에서는 데스크톱에서 응용 프로그램 아이콘을 두 번 클릭하거나 Windows [시작] 메뉴에서 해당 응용 프로그램을 선택합니다.
- Mac OS에서는 기본적으로 사용자 디렉토리의 Applications 폴더(예: Macintosh HD/Users/JoeUser/Applications/)에 설치되어 있는 응용 프로그램 아이콘을 두 번 클릭합니다.
- Linux에서는 데스크톱에서 응용 프로그램 아이콘을 두 번 클릭하거나 [응용 프로그램] 메뉴에서 해당 응용 프로그램을 선택합니다. AIR 응용 프로그램은 /opt 디렉토리 아래의 해당 폴더에 설치됩니다.

참고: 이러한 지침의 업데이트는 AIR 릴리스 정보(위치: http://www.adobe.com/go/learn_air_relnotes_kr)를 확인하십시오.

Adobe AIR 업데이트

Adobe는 새로운 기능 또는 문제 해결을 포함하도록 Adobe AIR를 정기적으로 업데이트합니다. 자동 알림 및 업데이트 기능을 사용하면 Adobe AIR의 업데이트된 버전이 제공될 때 Adobe에서 사용자에게 자동으로 알려 줍니다.

Adobe AIR 업데이트는 Adobe AIR가 올바르게 작동하도록 보장하며 중요한 보안 변경 사항을 포함할 수 있습니다. 사용자는 새로운 버전이 제공될 때마다 최신 버전의 Adobe AIR로 업데이트하는 것이 좋습니다. 특히 보안 업데이트가 포함된 경우에는 반드시 업데이트해야 합니다.

기본적으로 AIR 응용 프로그램이 실행되면 런타임에서 업데이트를 사용할 수 있는지 확인합니다. 런타임에서는 마지막 업데이트 확인 후 2주 이상 경과한 경우 업데이트가 있는지 확인합니다. 업데이트를 사용할 수 있는 경우 AIR는 업데이트를 백그라운드에서 다운로드합니다.

사용자는 AIR SettingsManager 응용 프로그램을 사용하여 자동 업데이트 기능을 비활성화할 수 있습니다. AIR SettingsManager 응용 프로그램은 http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air에서 다운로드할 수 있습니다.

일반적으로 Adobe AIR 설치 프로세스 동안 http://airinstall.adobe.com에 연결하면 운영 체제 버전, 언어 등의 설치 환경에 대한 기본적인 정보가 전송됩니다. 이 정보는 설치할 때마다 한 번만 전송되며 Adobe는 이를 통해 설치가 성공적으로 수행되었는지 확인할 수 있습니다. 이때 어떠한 개인 정보도 수집 또는 전송되지 않습니다.

3장: AIR API를 사용한 작업

Adobe® AIR®의 일부 기능은 Adobe® Flash® Player에서 실행되는 SWF 내용에 사용할 수 없습니다.

ActionScript 3.0 개발자
Adobe AIR API는 다음 두 문서에 설명되어 있습니다.

- ActionScript 3.0 개발자 안내서
- ActionScript 3.0 Reference for the Adobe Flash Platform

HTML 개발자
HTML 기반 AIR 응용 프로그램을 만드는 경우 AIRAliases.js 파일을 통해 JavaScript에서 사용할 수 있는 API(Accessing AIR API classes from JavaScript 참조)가 다음 두 문서에 설명되어 있습니다.

- HTML Developer's Guide for Adobe AIR
- Adobe AIR API Reference for HTML Developers

AIR에 고유한 ActionScript 3.0 클래스

다음 표에는 Adobe AIR에 고유한 런타임 클래스가 포함되어 있습니다. 브라우저의 Adobe® Flash® Player에서 실행되는 SWF 내용에는 해당 클래스를 사용할 수 없습니다.

HTML 개발자
AIRAliases.js 파일을 통해 JavaScript에서 사용할 수 있는 클래스는 Adobe AIR API Reference for HTML Developers에 설명되어 있습니다.

클래스	ActionScript 3.0 패키지
AAAARecord	flash.net.dns
ApplicationUpdater	air.update
ApplicationUpdaterUI	air.update
ARecord	flash.net.dns
BrowserInvokeEvent	flash.events
CertificateStatus	flash.security
CompressionAlgorithm	flash.utils
DatagramSocket	flash.net
DatagramSocketDataEvent	flash.events
DNSResolver	flash.net.dns
DNSResolverEvent	flash.events
DockIcon	flash.desktop

클래스	ActionScript 3.0 패키지
DownloadErrorEvent	air.update.events
DRMAuthenticateEvent	flash.events
DRMManagerError	flash.errors
EncryptedLocalStore	flash.data
File	flash.filesystem
FileListEvent	flash.events
FileMode	flash.filesystem
FileStream	flash.filesystem
FocusDirection	flash.display
Geolocation	flash.sensors
GeolocationEvent	flash.events
HTMLHistoryItem	flash.html
HTMLHost	flash.html
HTMLLoader	flash.html
HTMLPDFCapability	flash.html
HTMLUncaughtScriptExceptionEvent	flash.events
HTMLWindowCreateOptions	flash.html
Icon	flash.desktop
InteractiveIcon	flash.desktop
InterfaceAddress	flash.net
InvokeEvent	flash.events
InvokeEventReason	flash.desktop
MXRecord	flash.net.dns
NativeApplication	flash.desktop
NativeDragActions	flash.desktop
NativeDragEvent	flash.events
NativeDragManager	flash.desktop
NativeDragOptions	flash.desktop
NativeMenu	flash.display
NativeMenuItem	flash.display
NativeProcess	flash.desktop
NativeProcessExitEvent	flash.events
NativeProcessStartupInfo	flash.desktop
NativeWindow	flash.display

클래스	ActionScript 3.0 패키지
NativeWindowBoundsEvent	flash.events
NativeWindowDisplayState	flash.display
NativeWindowDisplayStateEvent	flash.events
NativeWindowInitOptions	flash.display
NativeWindowResize	flash.display
NativeWindowSystemChrome	flash.display
NativeWindowType	flash.display
NetworkInfo	flash.net
NetworkInterface	flash.net
NotificationType	flash.desktop
OutputProgressEvent	flash.events
PaperSize	flash.printing
PrintMethod	flash.printing
PrintUIOptions	flash.printing
PTRRecord	flash.net.dns
ReferencesValidationSetting	flash.security
ResourceRecord	flash.net.dns
RevocationCheckSettings	flash.security
Screen	flash.display
ScreenMouseEvent	flash.events
SecureSocket	flash.net
SecureSocketMonitor	air.net
SecureSocketConnectEvent	flash.net
ServiceMonitor	air.net
SignatureStatus	flash.security
SignerTrustSettings	flash.security
SocketMonitor	air.net
SQLCollationType	flash.data
SQLColumnNameStyle	flash.data
SQLColumnSchema	flash.data
SQLConnection	flash.data
SQLError	flash.errors
SQLErrorEvent	flash.events
SQLErrorOperation	flash.errors

클래스	ActionScript 3.0 패키지
SQLEvent	flash.events
SQLIndexSchema	flash.data
SQLMode	flash.data
SQLResult	flash.data
SQLSchema	flash.data
SQLSchemaResult	flash.data
SQLStatement	flash.data
SQLTableSchema	flash.data
SQLTransactionLockType	flash.data
SQLTriggerSchema	flash.data
SQLUpdateEvent	flash.events
SQLViewSchema	flash.data
SRVRecord	flash.net.dns
StageAspectRatio	flash.display
StageOrientation	flash.display
StageOrientationEvent	flash.events
StatusFileUpdateErrorEvent	air.update.events
StatusFileUpdateEvent	air.update.events
StatusUpdateErrorEvent	air.update.events
StatusUpdateEvent	air.update.events
StorageVolume	flash.filesystem
StorageVolumeChangeEvent	flash.events
StorageVolumeInfo	flash.filesystem
SystemTrayIcon	flash.desktop
UpdateEvent	air.update.events
Updater	flash.desktop
URLFilePromise	air.desktop
URLMonitor	air.net
URLRequestDefaults	flash.net
XMLSignatureValidator	flash.security

또한 다음 두 가지 AIR 관련 인터페이스가 있습니다.

- flash.desktop.IFilePromise
- flash.security.IURIDereferencer

AIR 고유 기능이 포함된 Flash Player 클래스

브라우저에서 실행되는 SWF 내용에서 다음과 같은 클래스를 사용할 수 있지만 AIR는 추가 속성 또는 메서드를 제공합니다.

클래스	속성 또는 메서드
Capabilities	languages
Clipboard	supportsFilePromise
ClipboardFormats	BITMAP_FORMAT FILE_LIST_FORMAT FILE_PROMISE_LIST_FORMAT URL_FORMAT
Event	DISPLAYING EXITING HTML_BOUNDS_CHANGE HTML_DOM_INITIALIZE HTML_RENDER LOCATION_CHANGE NETWORK_CHANGE USER_IDLE USER_PRESENT
FileReference	uploadUnencoded()
HTTPStatusEvent	HTTP_RESPONSE_STATUS responseURL responseHeaders
KeyboardEvent	commandKey controlKey
LoaderContext	allowLoadBytesCodeExecution
LoaderInfo	parentSandboxBridge childSandboxBridge
NetStream	resetDRMVouchers() setDRMAuthenticationCredentials()

클래스	속성 또는 메서드
PrintJob	active copies firstPage isColor jobName lastPage maxPixelsPerInch paperArea printableArea printer printers supportsPageSetupDialog maxPixelsPerInch
URLRequest	followRedirects manageCookies shouldAuthenticate shouldCacheResponse userAgent userCache selectPaperSize() showPageSetupDialog() start2() terminate()
PrintJobOptions	resolution
URLStream	httpResponseStatus 이벤트
Stage	nativeWindow
Security	APPLICATION

이러한 새 속성 및 메서드의 대부분은 AIR 응용 프로그램 보안 샌드박스의 내용에서만 사용할 수 있습니다. 그러나 기타 샌드박스에서 실행되는 내용에서는 URLRequest 클래스의 새 멤버도 사용할 수 있습니다.

ByteArray.compress() 및 ByteArray.uncompress() 메서드에는 각각 algorithm 매개 변수가 포함되어 있으므로 deflate 압축과 zlib 압축 중에서 선택할 수 있습니다. 이 매개 변수는 AIR에서 실행되는 내용에만 사용할 수 있습니다.

AIR 고유 Flex 구성 요소

Adobe AIR에 대한 내용을 개발할 때 다음과 같은 Adobe® Flex™ MX 구성 요소를 사용할 수 있습니다.

- FileEvent

- FileSystemComboBox
- FileSystemDataGrid
- FileSystemEnumerationMode
- FileSystemHistoryButton
- FileSystemList
- FileSystemSizeDisplayMode
- FileSystemTree
- FlexNativeMenu
- HTML
- Window
- WindowedApplication
- WindowedSystemManager

또한 Flex 4에는 다음과 같은 Spark AIR 구성 요소가 포함되어 있습니다.

- Window
- WindowedApplication

AIR Flex 구성 요소에 대한 자세한 내용은 Using the Flex AIR components를 참조하십시오.

4장: AIR 개발용 Adobe Flash Platform 도구

다음과 같은 Adobe Flash Platform 개발 도구를 사용하여 AIR 응용 프로그램을 개발할 수 있습니다.

ActionScript 3.0(Flash 및 Flex) 개발자용:

- Adobe Flash Professional CS3(16페이지의 "Adobe AIR용 Flash CS3 설정" 참조)
- Adobe Flash Professional CS4(AIR용으로 제작 참조)
- Adobe Flash Professional CS5(AIR용으로 제작 참조)
- Adobe Flex 3.x 및 4.0 SDK(22페이지의 "Flex SDK 설정" 및 42페이지의 "명령줄 도구를 사용하여 AIR 응용 프로그램 만들기" 참조)
- Adobe Flex Builder 3.x(18페이지의 "Flex Builder 3으로 AIR 응용 프로그램 개발" 참조)
- Adobe Flash Builder 4(Developing AIR Applications with Flash Builder 참조)

HTML 및 Ajax 개발자용:

- Adobe AIR SDK (21페이지의 "AIR SDK 설치" 및 42페이지의 "명령줄 도구를 사용하여 AIR 응용 프로그램 만들기" 참조)
- Adobe Dreamweaver CS3, CS4, CS5(AIR Extension for Dreamweaver 참조)

Adobe AIR용 Flash CS3 설정

Adobe® Flash® CS3 Professional용 Adobe® AIR® 업데이트는 Flash를 사용하여 AIR 응용 프로그램을 만들 수 있도록 해 주는 요소를 Flash 개발 환경에 추가합니다. 그 결과 사용자는 Flash에서 AIR 응용 프로그램을 만들고 테스트하며 디버깅할 수 있습니다.

Adobe® Flash® CS4 Professional 및 Adobe® Flash® CS5 Professional에는 AIR 응용 프로그램을 만들 수 있는 기본 지원이 포함되어 있습니다. 자세한 내용은 Adobe AIR용으로 제작을 참조하십시오.

참고: Flash CS3용 Adobe AIR 업데이트에서는 AIR 1.0, 1.1 및 Flash Player 9.x를 지원합니다. AIR 1.5.x 및 Flash Player 10을 사용하여 응용 프로그램을 개발하려면 Flash CS4가 있어야 합니다. Flash CS5는 AIR 2 및 Flash Player 10.1로 응용 프로그램을 개발하는 데 필요합니다.

Adobe Flash Professional CS4 및 CS5

최신 버전의 Flash Professional로 AIR 응용 프로그램을 제작하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Adobe Flash Professional CS4(AIR용으로 제작 참조)
- Adobe Flash Professional CS5(AIR용으로 제작 참조)

Flex Professional로 AIR 응용 프로그램을 만드는 방법에 대한 간단한 개요를 보려면 33페이지의 "Flash Professional을 사용하여 첫 번째 AIR 응용 프로그램 만들기"를 참조하십시오.

Flash CS3용 Adobe AIR 업데이트를 위한 시스템 요구 사항

Flash CS3를 사용하여 AIR 응용 프로그램을 개발하고 실행하려면 다음 소프트웨어를 설치해야 합니다.

- Flash CS3 Professional

  Flash CS3 Professional의 사본이 없는 경우 Adobe 웹 사이트(http://www.adobe.com/kr/products/flash/)에서 구입할 수 있습니다.

- Adobe AIR

  Adobe AIR 설치에 대한 자세한 내용은 6페이지의 "Adobe AIR 설치"를 참조하십시오.

- Flash CS3용 Adobe AIR 업데이트

Flash CS 3용 Adobe AIR 1.0 업데이트 설치

Flash CS3용 Adobe AIR 업데이트를 설치하려면 먼저 Flash를 종료하고 열려 있는 브라우저를 모두 닫아야 합니다.

- Flash CS3용 Adobe AIR 1.0 업데이트 다운로드(http://www.adobe.com/support/flash/downloads.html#flashCS3)
- 업데이트의 다운로드가 완료되면 업데이트 패치 파일을 두 번 클릭하여 설치합니다.

Flash CS 3용 Adobe AIR 1.1 업데이트 설치

이 업데이트를 설치하려면 먼저 Adobe Flash CS3 Professional용 원본 Adobe AIR 1.0 업데이트를 설치해야 합니다. 이전 업데이트를 설치하지 않으면 이 업데이트를 설치할 수 없습니다.

1. 이름이 "AIK"인 폴더를 찾습니다.

   Windows: C:/Program Files/Adobe/Adobe Flash CS3

   Mac OS X: Macintosh HD: Applications: Adobe Flash CS3

2. AIK 폴더에서 모든 파일을 제거합니다. 폴더 자체는 제거하지 마십시오.
3. 다음 위치에서 AIRSDKIntegrationKit를 다운로드합니다.

   Windows: http://www.adobe.com/go/getaikwin_kr

   Mac OS X: http://www.adobe.com/go/getaikmac_kr

4. AIRSDKIntegrationKit의 내용을 압축 해제하고 AIK 폴더 안에 저장합니다.
5. http://get.adobe.com/kr/air/에서 Adobe AIR 1.1 런타임을 다운로드합니다.
6. 런타임을 사용자의 컴퓨터에 설치합니다.

Flash CS3용 Adobe AIR 업데이트 제거

Flash CS3용 Adobe AIR 업데이트를 제거하려면 다음 단계를 수행합니다.

1. 다음 폴더를 삭제합니다.

   (Windows) 하드 드라이브:\Program Files\Adobe\Adobe Flash CS3\AIK

   (Mac) 하드 드라이브:/Applications/Adobe Flash CS3/AIK

2. 다음 위치를 찾습니다.

   (Windows) 하드 드라이브:\Program Files\Adobe\Adobe Flash CS3\<언어>\First Run\Commands\

   (Mac) 하드 드라이브:/Applications/Adobe Flash CS3/First Run/Commands

다음 파일/폴더를 삭제합니다.

- AIR 폴더
- AIR - Application and Installer Settings.jsfl
- AIR - Create AIR File.jsfl

3 다음 파일을 삭제합니다.

(Windows) 하드 드라이브:\Program Files\Adobe\Adobe Flash CS3\<언어>\Configuration\External Libraries\FLAir.dll

(Mac) 하드 드라이브:/Applications/Adobe Flash CS3/Configuration/External Libraries/FLAir.bundle

4 다음 파일을 삭제합니다.

(Windows) 하드 드라이브:\Program Files\Adobe\Adobe Flash CS3\<언어>\Configuration\Players\AdobeAIR1_0.xml

(Mac) 하드 드라이브:/Applications/Adobe Flash CS3/Configuration/Players/ AdobeAIR1_0.xml

5 다음 위치를 찾습니다.

(Windows) 하드 드라이브:\Document and Settings\<사용자 이름>\Local Settings\Application Data\Adobe\Flash CS3\<언어>\Configuration\Commands\

(Mac) 하드 드라이브:/Users/<사용자 이름>/Library/Application Support/Adobe/Flash CS3/<언어>/Configuration/Commands/

다음 파일/폴더를 삭제합니다.

- AIR 폴더
- AIR - Application and Installer Settings.jsfl
- AIR - Create AIR File.jsfl

참고: Windows에서 지정된 위치를 찾을 수 없는 경우 폴더 옵션의 "숨김 파일 및 폴더 표시"를 선택하십시오.

Flex Builder 3으로 AIR 응용 프로그램 개발

Adobe® Flex™ Builder™ 3은 Adobe® AIR® 응용 프로그램을 만들기 위한 다음과 같은 도구를 제공합니다. Flex Builder에는 Adobe AIR 응용 프로그램용 Flex 구성 요소가 포함됩니다. Flex Builder에서 AIR 응용 프로그램을 개발하기 위한 작업 과정은 대부분의 Flex 기반 응용 프로그램을 개발할 때와 비슷합니다.

Adobe Flash Builder

Flash Builder로 AIR 응용 프로그램을 만드는 방법에 대한 자세한 내용은 Developing AIR applications with Flash Builder를 참조하십시오.

Flex Builder 3 또는 Flash Builder 4로 AIR 응용 프로그램을 만드는 방법에 대한 간단한 개요를 보려면 25페이지의 “Flash Builder 또는 Flex Builder에서 첫 번째 Flex AIR 응용 프로그램 만들기”를 참조하십시오.

Flex Builder를 사용하여 AIR 프로젝트 만들기

아직 설치되지 않았으면 AIR와 Flex Builder 3을 설치합니다.

- Flex Builder 3을 엽니다.

- [File] > [New] > [Flex Project]를 선택합니다.
- 프로젝트 이름을 입력합니다.
- Flex에서 AIR 응용 프로그램은 응용 프로그램 유형으로 간주됩니다. 두 가지 방법이 있습니다. 한 가지 방법은 Adobe® Flash® Player에서 웹으로 실행되는 Flex 기반 응용 프로그램을 만드는 것입니다. 다른 방법은 Adobe AIR에서 데스크톱으로 실행되는 AIR 응용 프로그램을 만드는 것입니다. 응용 프로그램 유형으로 [Desktop Application]을 선택합니다.
- AIR 응용 프로그램에서 사용하려는 서버 기술(있는 경우)을 선택합니다. 서버 기술을 사용 중이 아니면 [None]을 선택하고 [Next]를 클릭합니다.
- 응용 프로그램을 저장할 폴더를 선택합니다. 기본값은 bin 폴더입니다. [Next]를 클릭합니다.
- 필요에 따라 소스 및 라이브러리 경로를 수정한 다음 [Finish]를 클릭하여 AIR 프로젝트를 만듭니다.

Flex Builder를 사용하여 AIR 응용 프로그램 디버깅

Flex Builder는 AIR 응용 프로그램에 대한 전체 디버깅 지원을 제공합니다. Flex Builder의 디버깅 기능에 대한 자세한 내용은 Flex Builder 도움말을 참조하십시오.

1. Flex Builder에서 응용 프로그램(예: MXML 파일)에 대한 소스 파일을 엽니다.
2. 기본 툴바에서 [Debug] 버튼을 클릭합니다.

   또는 [Run] > [Debug]를 선택합니다.

응용 프로그램이 시작되어 ADL 응용 프로그램에서 실행됩니다(AIR Debug Launcher) Flex Builder 디버거는 모든 중단점 또는 런타임 오류를 catch하고 사용자는 다른 Flex 응용 프로그램에서와 같이 응용 프로그램을 디버깅할 수 있습니다.

AIR Debug Launcher 명령줄 도구를 사용하여 명령줄에서 응용 프로그램을 디버깅할 수도 있습니다. 자세한 내용은 45페이지의 “ADL(AIR Debug Launcher) 사용”을 참조하십시오.

Flex Builder를 사용하여 AIR 응용 프로그램 패키지

응용 프로그램이 완료되고 배포할 준비가 되면(또는 데스크톱에서 실행이 테스트되면) 이를 AIR 파일로 패키지화합니다. 패키징은 다음 단계로 구성됩니다.

- 제작할 AIR 응용 프로그램 선택
- 선택적으로 사용자가 소스 코드를 볼 수 있도록 허용하고 포함할 응용 프로그램 파일 선택
- 상용 코드 서명 인증서를 사용하거나 자체 서명 인증서를 만들고 적용하여 AIR 응용 프로그램에 디지털 서명 적용
- 선택적으로 나중에 서명할 수 있도록 중간 AIR 파일을 만들도록 선택

AIR 응용 프로그램 패키지

1. 프로젝트를 열고 응용 프로그램에 컴파일 오류가 없고 예상한 대로 실행되는지 확인합니다.
2. [Project] > [Export Release Build]를 선택합니다.
3. Flex Builder에 여러 개의 프로젝트와 응용 프로그램이 열려 있다면 패키지화할 특정 AIR 프로젝트를 선택해야 합니다.
4. 사용자가 응용 프로그램을 실행할 때 소스 코드를 볼 수 있도록 하려면 선택적으로 [Enable View Source]를 선택합니다. [Choose Source Files]를 선택하여 제외할 개별 파일을 선택할 수 있습니다. 기본적으로 모든 소스 파일이 선택됩니다. Flex Builder에서 소스 파일 제작에 대한 자세한 내용은 Flex Builder 도움말을 참조하십시오.
5. 생성된 AIR 파일의 이름을 선택적으로 변경할 수도 있습니다. 계속할 준비가 되면 [Next]를 클릭하여 응용 프로그램을 디지털로 서명합니다.

AIR 응용 프로그램에 디지털 서명

[Export Release Version]를 계속 실행하기 전에 AIR 응용 프로그램의 디지털 서명 방법을 결정합니다. 몇 가지 방법이 있습니다. 상용 코드 서명 인증서를 사용하여 응용 프로그램을 서명하거나, 자체 서명 디지털 인증서를 만들어서 사용하거나, 일단 응용 프로그램을 패키지화하고 나중에 서명하도록 선택할 수 있습니다.

인증 기관(예: VeriSign, Thawte, GlobalSign 및 ChosenSecurity)에서 발급한 디지털 인증서를 사용하면 사용자가 제작자의 ID를 확인시키고 설치 파일이 서명된 이후로 변경되지 않았음을 보증합니다. 자체 서명된 디지털 인증서는 동일한 기능을 수행하지만 제삼자의 유효성 검사는 제공하지 않습니다.

또한 중간 AIR 파일(.airi)을 만들어서 디지털 서명 없이 AIR 응용 프로그램을 패키지화하는 방법도 있습니다. 중간 AIR 파일은 설치가 불가능하기 때문에 유효하지 않습니다. 대신 테스트용으로(개발자) 사용되며 AIR ADT 명령줄 도구를 사용하여 실행할 수 있습니다. AIR에서 이 기능이 제공되는 이유는 일부 개발 환경의 경우 특정 개발자 또는 팀에서 서명을 처리하기 때문입니다. 이렇게 하면 디지털 인증서를 관리하는 데 추가적인 보안 수준을 얻을 수 있습니다.

응용 프로그램에 서명하는 방법에 대한 자세한 내용은 84페이지의 “AIR 파일에 디지털 서명”을 참조하십시오.

AIR 응용 프로그램을 디지털로 서명

1. 기존 디지털 인증서를 선택하거나 새 자체 서명 인증서를 만들어서 AIR 응용 프로그램을 디지털로 서명할 수 있습니다. [Export and Sign an AIR File with a Digital Certificate] 옵션을 선택합니다.
2. 사용할 수 있는 기존 디지털 인증서가 있다면 [Browse]를 클릭하여 인증서를 선택합니다.
3. 새 자체 서명 디지털 인증서를 만들려면 [Create]를 선택합니다.
4. 필요한 정보를 입력하고 [OK]를 클릭합니다.
5. [Next]를 클릭하여 내보낸 AIR 파일에서 선택적으로 제외할 파일을 선택합니다. 기본적으로 모든 파일이 포함됩니다.
6. [Finish]를 클릭하여 AIR 파일을 생성합니다.

중간 AIR 파일 만들기

❖ [Export an Intermediate AIRI File that will be Exported Later] 옵션을 선택합니다. [Finish]를 클릭하여 중간 파일을 생성합니다.

중간 AIR 파일을 생성한 다음에는 ADT 명령줄 도구를 사용하여 서명할 수 있습니다(59페이지의 “ADT를 사용하여 AIR 중간 파일 서명” 참조).

AIR 라이브러리 프로젝트 만들기

여러 AIR 프로젝트에 대한 AIR 코드 라이브러리를 만들려면 표준 Flex 라이브러리 프로젝트 마법사를 사용하여 AIR 라이브러리 프로젝트를 만듭니다.

1. [File] > [New] > [Flex Library Project]를 선택합니다.
2. 프로젝트를 지정하십시오.
3. [Add Adobe AIR Libraries]를 선택한 후 [Next]를 클릭합니다.

   참고: 선택한 Flex SDK 버전이 AIR를 지원해야 합니다. Flex 2.0.1 SDK는 지원하지 않습니다.
4. 필요에 따라 빌드 경로를 수정하고 [Finish]를 클릭합니다. 라이브러리 프로젝트를 만드는 방법에 대한 자세한 내용은 Flex Builder 도움말에서 “라이브러리 프로젝트 정보”를 참조하십시오.

AIR SDK 설치

Adobe AIR SDK에는 응용 프로그램 실행 및 패키지를 위해 사용하는 다음과 같은 명령줄 도구가 포함됩니다.

ADL(AIR Debug Launcher) AIR 응용 프로그램을 먼저 설치하지 않고도 실행할 수 있게 해줍니다. 45페이지의 "ADL(AIR Debug Launcher) 사용"을 참조하십시오.

ADT(AIR Development Tool) AIR 응용 프로그램을 배포 가능한 설치 패키지로 패키지화합니다. 48페이지의 "ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지"를 참조하십시오.

AIR 명령줄 도구를 사용하려면 Java가 컴퓨터에 설치되어 있어야 합니다. JRE 또는 JDK(버전 1.5 이상)에서 Java 가상 시스템을 사용할 수 있습니다. Java JRE 및 Java JDK는 http://java.sun.com/에서 제공됩니다.

참고: 최종 사용자가 AIR 응용 프로그램을 실행할 때는 Java가 필요하지 않습니다.

AIR SDK로 AIR 응용 프로그램을 만드는 방법에 대한 간단한 개요를 보려면 38페이지의 "AIR SDK를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기"를 참조하십시오.

AIR SDK 다운로드 및 설치

다음 지침에 따라 AIR SDK를 다운로드하고 설치할 수 있습니다.

Windows에 AIR SDK 설치

- AIR SDK 설치 파일을 다운로드합니다.
- AIR SDK는 표준 파일 아카이브로 배포됩니다. AIR를 설치하려면 SDK의 내용을 컴퓨터 폴더에 추출합니다(예: C:\Program Files\Adobe\AIRSDK 또는 C:\AIRSDK)
- ADL 및 ADT 도구는 AIR SDK의 bin 폴더에 포함됩니다. 이 폴더에 대한 경로를 PATH 환경 변수에 추가하십시오.

Mac OS X에 AIR SDK 설치

- AIR SDK 설치 파일을 다운로드합니다.
- AIR SDK는 표준 파일 아카이브로 배포됩니다. AIR를 설치하려면 SDK의 내용을 컴퓨터 폴더에 추출합니다(예: /Users/<사용자 이름>/Applications/AIRSDK).
- ADL 및 ADT 도구는 AIR SDK의 bin 폴더에 포함됩니다. 이 폴더에 대한 경로를 PATH 환경 변수에 추가하십시오.

Linux에 AIR SDK 설치

- SDK는 tbz2 형식으로 제공됩니다.
- SDK를 설치하려면 SDK의 압축을 해제하려는 폴더를 만든 후 tar -jxvf <path to AIR-SDK.tbz2> 명령을 사용합니다.

AIR SDK 도구 사용 시작에 대한 자세한 내용은 명령줄 도구를 사용하여 AIR 응용 프로그램 만들기를 참조하십시오.

AIR SDK에 포함된 내용

다음 표에서는 AIR SDK에 포함된 파일의 목적에 대해 설명합니다.

SDK 폴더	파일/도구 설명
BIN	adl.exe - ADL(AIR Debug Launcher)은 AIR 응용 프로그램을 먼저 패키지화하거나 설치하지 않고도 실행할 수 있게 해줍니다. 이 도구 사용에 대한 자세한 내용은 45페이지의 "ADL(AIR Debug Launcher) 사용"을 참조하십시오. adt.bat - ADT(AIR Developer Tool)는 응용 프로그램을 배포할 수 있도록 AIR 파일로 패키지화합니다. 이 도구 사용에 대한 자세한 내용은 48페이지의 "ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지"를 참조하십시오.
FRAMEWORKS	AIRAliases.js - Javascript 코드에서 AIR 런타임 클래스를 쉽게 참조하도록 "별칭" 정의를 제공합니다. aircore 라이브러리 - 파일 프로미스, 네트워크 모니터링과 같은 확장된 응용 프로그램 기능을 제공합니다. applicationupdator 라이브러리 - AIR 응용 프로그램에서 사용할 업데이트 프레임워크를 구현합니다. airglobal 라이브러리 - 기본 AIR 런타임 클래스를 정의합니다.
LIB	adt.jar - adt.bat 파일로 호출되는 adt 실행 파일입니다. Descriptor.1.0.xsd - 응용 프로그램 스키마 파일입니다.
RUNTIMES	AIR 런타임 - 이 런타임은 ADL이 AIR 응용 프로그램을 패키지 또는 설치하기 전에 실행하기 위해 사용됩니다.
SAMPLES	이 폴더에는 샘플 응용 프로그램 설명자 파일, 간편한 설치 기능 샘플(badge.swf) 및 기본 AIR 응용 프로그램 아이콘이 포함됩니다. 75페이지의 "AIR 응용 프로그램 배포, 설치 및 실행"을 참조하십시오.
TEMPLATES	descriptor-template.xml - 각 AIR 응용 프로그램에 필요한 응용 프로그램 설명자 파일의 템플릿입니다. 응용 프로그램 설명자 파일에 대한 자세한 설명은 62페이지의 "AIR 응용 프로그램 속성 설정"을 참조하십시오.

Flex SDK 설정

Adobe® Flex™에서 Adobe® AIR® 응용 프로그램을 개발하기 위해서는 다음과 같은 옵션이 있습니다.

- Adobe AIR 프로젝트를 만들고 AIR 응용 프로그램을 테스트, 디버깅 및 패키지화하기 위한 통합 도구를 제공하는 Adobe® Flash® Builder™를 다운로드하고 설치할 수 있습니다. 25페이지의 "Flash Builder 또는 Flex Builder에서 첫 번째 Flex AIR 응용 프로그램 만들기"를 참조하십시오.
- Adobe® Flex™ SDK를 다운로드하고 자주 사용하는 텍스트 편집기 및 명령줄 도구를 사용하여 Flex AIR 응용 프로그램을 개발할 수 있습니다.

Flex SDK로 AIR 응용 프로그램을 만드는 방법에 대한 간단한 개요를 보려면 29페이지의 "Flex SDK를 사용하여 첫 번째 AIR 응용 프로그램 만들기"를 참조하십시오.

Flex SDK의 AIR 명령줄 도구

Adobe AIR 응용 프로그램을 만들기 위해 사용하는 각 명령줄 도구에서는 Flex 응용 프로그램을 만들기 위해 사용되는 해당 도구를 호출할 수 있습니다.

- amxmlc는 mxmlc를 호출하여 응용 프로그램 클래스를 컴파일합니다.
- acompc는 compc를 호출하여 라이브러리 및 구성 요소 클래스를 컴파일합니다.
- aasdoc는 asdoc를 호출하여 소스 코드 설명에서 문서 파일을 생성합니다.

이러한 유틸리티의 Flex 및 AIR 버전 간의 유일한 차이점은 AIR 버전의 경우 flex-config.xml 파일이 아닌 air-config.xml 파일에서 구성 옵션을 로드한다는 점입니다.

Flex SDK 도구 및 해당 명령줄 옵션은 Flex 설명서 라이브러리의 Flex 응용 프로그램 만들기 및 배포에서 자세히 설명됩니다. 여기에서는 Flex SDK 도구에 대해 도구 사용을 시작하고 Flex 응용 프로그램 만들기와 AIR 응용 프로그램 만들기 사이의 차이점을 이해할 수 있도록 기본적인 내용이 설명됩니다.

기타 도움말 항목

29페이지의 “Flex SDK를 사용하여 첫 번째 AIR 응용 프로그램 만들기”

Flex compilers

Flex SDK 설치

명령줄 도구를 사용하여 AIR 응용 프로그램을 만들려면 컴퓨터에 Java가 설치되어 있어야 합니다. JRE 또는 JDK(버전 1.5 이상)에서 Java 가상 시스템을 사용할 수 있습니다. Java JRE 및 JDK는 http://java.sun.com/에서 제공됩니다.

참고: 최종 사용자가 AIR 응용 프로그램을 실행할 때는 Java가 필요하지 않습니다.

Flex SDK는 AIR 응용 프로그램을 패키지, 컴파일 및 디버그하는 데 사용하는 AIR API 및 명령줄 도구를 제공합니다.

1. 아직 Flex SDK를 설치하지 않은 경우 http://opensource.adobe.com/wiki/display/flexsdk/Downloads에서 다운로드하십시오.
2. SDK의 내용을 폴더(예: Flex SDK)에 저장합니다.
3. 명령줄 유틸리티는 bin 폴더에 있습니다.

컴파일러 설정

일반적으로 명령줄 및 하나 이상의 구성 파일에서 컴파일 옵션을 지정합니다. 전역 Flex SDK 구성 파일에는 컴파일러가 실행될 때마다 사용되는 기본 값이 포함됩니다. 사용자의 고유 개발 환경에 적합하도록 이 파일을 편집할 수 있습니다. 설치된 Flex SDK의 frameworks 디렉토리에는 두 개의 전역 Flex 구성 파일이 있습니다. air-config.xml 파일은 amxmlc 컴파일러를 실행할 때 사용됩니다. 이 파일은 AIR 라이브러리를 포함하여 AIR에 대한 컴파일러를 구성합니다. flex-config.xml 파일은 mxmlc를 실행할 때 사용됩니다.

기본 구성 값은 Flex 및 AIR의 작동 방식을 확인할 때 적합하지만 전체 프로젝트를 시작할 때는 가능한 옵션들을 보다 세밀하게 검사해야 합니다. 특정 프로젝트에 대해 전역 값보다 우선 적용되는 로컬 구성 파일의 컴파일러 옵션에 프로젝트별 값을 제공할 수 있습니다. 컴파일 옵션 및 구성 파일의 구문에 대한 전체 목록은 Flex 설명서 라이브러리에서 Flex 응용 프로그램 만들기 및 배포의 Flex SDK 구성을 참조하십시오.

참고: 컴파일 옵션은 특히 AIR 응용 프로그램에 대해 사용되지 않지만 AIR 응용 프로그램을 컴파일할 때는 AIR 라이브러리를 참조해야 합니다. 일반적으로 이러한 라이브러리는 프로젝트 레벨 구성, Ant와 같은 빌드 도구에 대한 파일 또는 명령줄에서 직접 참조됩니다.

디버거 설정

AIR는 디버깅을 직접 지원하므로 Adobe® Flash® Player에서와 같이 런타임의 디버그 버전이 필요하지 않습니다. 명령줄 디버깅을 수행하려면 Flash Debugger 및 ADL(AIR Debug Launcher)을 사용합니다.

Flash Debugger는 Flex SDK 디렉토리에 배포됩니다. Windows의 fdb.exe와 같은 기본 버전은 bin 하위 디렉토리에 있습니다. Java 버전은 lib 하위 디렉토리에 있습니다. AIR Debug Launcher인 adl.exe는 Flex SDK 설치의 bin 디렉토리에 있습니다. 별개의 Java 버전은 없습니다.

참고: fdb는 Flash Player로 시작하려고 시도하기 때문에 fdb로 직접 AIR 응용 프로그램을 시작할 수 없습니다. 대신 AIR 응용 프로그램이 실행 중인 fdb 세션에 연결하도록 하십시오.

자세한 내용은 45페이지의 “ADL(AIR Debug Launcher) 사용”을 참조하십시오.

응용 프로그램 패키지 프로그램 설정

응용 프로그램을 설치 가능한 AIR 파일로 패키지화하는 ADT(AIR Developer Tool)는 Java 프로그램입니다. 유틸리티를 편리하게 실행할 수 있도록 환경을 설정하는 것 이외의 설정은 필요하지 않습니다.

SDK에는 ADT를 명령으로 실행하기 위해 SDK bin 디렉토리에 스크립트 파일이 포함됩니다. 또한 Apache Ant와 같은 빌드 도구를 사용할 때 편리하도록 ADT를 Java 프로그램으로 실행할 수도 있습니다.

자세한 내용은 48페이지의 “ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지”를 참조하십시오.

5장: Flash Builder 또는 Flex Builder에서 첫 번째 Flex AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 빠르게 살펴보려면 다음 지침에 따라 Adobe® Flash® Builder를 사용하여 간단한 SWF 파일 기반 AIR "Hello World" 응용 프로그램을 만들어 패키지해 보십시오.

아직 Flex Builder 3 또는 Flex Builder 4를 설치하지 않았다면 지금 다운로드하여 설치하십시오.

AIR 프로젝트 만들기

Flash Builder 에는 AIR 응용 프로그램을 개발하고 패키지할 수 있는 도구가 포함되어 있습니다.

다른 Flex 기반 응용 프로그램 프로젝트를 만들 때와 마찬가지로 새 프로젝트를 정의하여 Flash Builder 또는 Flex Builder에서 AIR 응용 프로그램을 만들 수 있습니다.

참고: 다음 지침은 Flash Builder를 사용하는 경우의 예입니다. 이 지침은 Flex Builder 3에도 동일하게 적용됩니다.

1. Flash Builder를 엽니다.
2. [File] > [New] > [Flex Project]를 선택합니다.
3. 프로젝트 이름을 AIRHelloWorld로 입력합니다.
4. Flex에서 AIR 응용 프로그램은 응용 프로그램 유형으로 간주됩니다. 두 가지 방법이 있습니다.
   - Adobe® Flash® Player에서 웹으로 실행되는 Flex 응용 프로그램
   - Adobe AIR에서 데스크톱으로 실행되는 AIR 응용 프로그램

   응용 프로그램 유형으로 [Desktop Application]을 선택합니다.
5. 서버 기술을 사용하지 않을 것이므로 해당 옵션에 [None]을 선택하고 [Next]를 클릭합니다.
6. 컴파일된 응용 프로그램을 저장할 폴더를 선택합니다. 기본값은 bin 폴더입니다. [Finish]를 클릭하여 프로젝트를 만듭니다.

초기 AIR 프로젝트는 기본 MXML 파일과 응용 프로그램 XML 파일(응용 프로그램 설명자 파일이라고 함)의 두 가지 파일로 구성됩니다. 응용 프로그램 설명자 파일은 AIR 응용 프로그램을 식별, 설치 및 시작하는 매개 변수를 지정합니다. 이 파일을 직접 편집해야 하는 경우가 있지만 지금은 이러한 파일이 있다는 것만 알아 두십시오.

자세한 내용은 18페이지의 “Flex Builder 3으로 AIR 응용 프로그램 개발” 및 Developing AIR applications with Flash Builder를 참조하십시오.

AIR 응용 프로그램 코드 작성

"Hello World" 응용 프로그램 코드를 작성하려면 응용 프로그램 MXML 파일(AIRHelloWorld.mxml)을 편집기에서 열어 편집해야 합니다. 파일이 열려 있지 않다면 Project Navigator를 사용하여 여십시오.

Flex AIR 응용 프로그램은 MXML WindowedApplication 태그 내에 포함됩니다. MXML WindowedApplication 태그는 제목 표시줄, 닫기 버튼 같은 기본 윈도우 컨트롤이 포함된 간단한 윈도우를 만듭니다.

1. WindowedApplication 구성 요소에 title 특성을 추가하고 값으로 "Hello World"를 할당합니다.

```
?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"
                       title="Hello World">
</s:WindowedApplication>
```

2 응용 프로그램에 Label 구성 요소를 추가(WindowedApplication 태그 안에 배치)합니다. Label 구성 요소의 text 속성을 "Hello AIR"로 설정한 후 다음과 같이 텍스트를 가운데에 맞추는 레이아웃 제한을 설정합니다.

```
<?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"
                       title="Hello World">

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

3 여는 WindowedApplication 태그와 앞서 입력한 레이블 구성 요소 태그 사이에 다음 스타일 블록을 추가합니다.

```
<fx:Style>
    @namespace s "library://ns.adobe.com/flex/spark";
    s|WindowedApplication
    {

skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
        background-color:#999999;
        background-alpha:"0.7";
    }
</fx:Style>
```

이 스타일 설정은 전체 응용 프로그램에 적용되어 윈도우 배경을 조금 투명한 회색으로 렌더링합니다.

이제 응용 프로그램 코드가 다음과 같아야 합니다.

```
<?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"
                       title="Hello World">

    <fx:Style>
        @namespace s "library://ns.adobe.com/flex/spark";
        s|WindowedApplication
        {

skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
            background-color:#999999;
            background-alpha:"0.7";
        }
    </fx:Style>

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

다음으로, 응용 프로그램을 투명하게 만드는 몇 가지 설정을 변경합니다.

1 Flex Navigator 윈도우에서 프로젝트의 소스 디렉토리에 있는 응용 프로그램 설명자 파일을 찾습니다. 프로젝트 이름을 AIRHelloWorld로 지정했으므로 이 파일의 이름은 AIRHelloWorld-app.xml입니다.

2 응용 프로그램 설명자 파일을 두 번 클릭하여 Flash Builder에서 편집합니다.

3 XML 코드에서 systemChrome 및 transparent 속성에 대한 주석 줄을 찾습니다. 이 줄은 initialWindow 속성 안에 있습니다. 주석을 제거합니다. 즉, "<!--" 및 "-->" 주석 구분 기호를 제거하십시오.

4 다음과 같이 systemChrome 속성의 텍스트 값을 none으로 설정합니다.

```
<systemChrome>none</systemChrome>
```

5 다음과 같이 transparent 속성의 텍스트 값을 true로 설정합니다.

```
<transparent>true</transparent>
```

6 파일을 저장합니다.

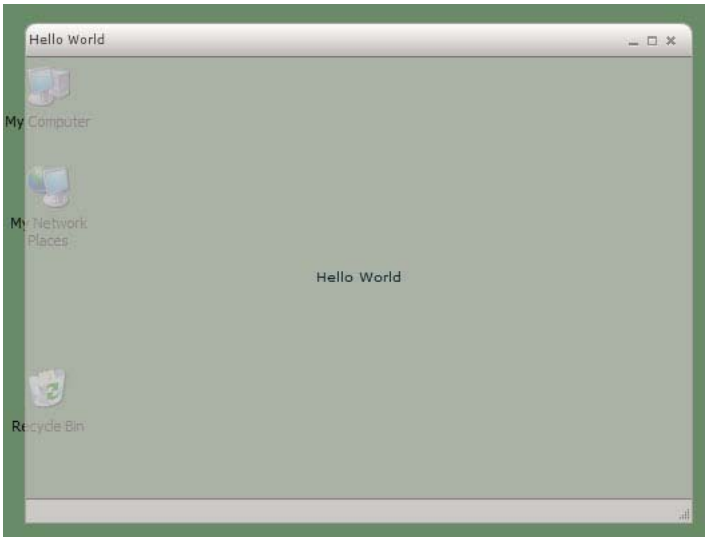
AIR 응용 프로그램 테스트

작성한 응용 프로그램 코드를 테스트하려면 디버그 모드에서 응용 프로그램을 실행합니다.

1 기본 툴바에서 [Debug] 버튼 을 클릭합니다.

또는 [Run] > [Debug] > [AIRHelloWorld] 명령을 선택할 수도 있습니다.

결과 AIR 응용 프로그램은 다음 예제와 유사해야 합니다(녹색 배경은 바탕 화면임).



2 Label 컨트롤의 horizontalCenter 및 verrticalCenter 속성을 사용하여 텍스트를 윈도우 가운데에 배치합니다. 다른 모든 데스크톱 응용 프로그램과 마찬가지로 윈도우를 이동하거나 윈도우의 크기를 조정할 수 있습니다.

참고: 응용 프로그램이 컴파일되지 않으면 코드에서 잘못 입력한 구문 오류나 맞춤법 오류가 없는지 확인하고 해당 오류를 수정합니다. 오류와 경고는 Flash Builder의 Problems 보기에 표시됩니다.

AIR 응용 프로그램 패키지, 서명 및 실행

이제 "Hello World" 응용 프로그램을 AIR 파일로 패키지화하여 배포할 수 있습니다. AIR 파일은 프로젝트의 bin 폴더에 들어 있는 모든 파일(응용 프로그램 파일)이 포함된 보관 파일입니다. 이 간단한 예제에서는 SWF 파일과 응용 프로그램 XML 파일이 보관 파일에 포함됩니다. 이제 AIR 패키지를 배포하면 사용자가 응용 프로그램을 설치하고 사용할 수 있습니다. 이 과정의 필수 단계는 패키지에 디지털 서명을 하는 것입니다.

1 응용 프로그램에서 컴파일 오류가 발생하지 않고 예상대로 실행되는지 확인합니다.

2 [Project] > [Export Release Build]를 선택합니다.

3 여러 프로젝트와 응용 프로그램이 열려 있는 경우 패키지화할 특정 AIR 프로젝트를 선택합니다. 그런 다음 [Next] 버튼을 클릭합니다.

4 [Export and Sign an AIR File with a Digital Certificate] 옵션을 선택합니다.

5 사용할 수 있는 기존 디지털 인증서가 있다면 [Browse]를 클릭하여 인증서를 선택합니다.

6 새 자체 서명 디지털 인증서를 만들려면 [Create]를 선택합니다.

7 필요한 정보를 입력하고 [OK]를 클릭합니다.

8 [Finish]를 클릭하여 AIR 패키지를 생성합니다. 생성된 패키지의 이름은 AIRHelloWorld.air입니다.

이제 Flash Builder의 Project Navigator나 파일 시스템에서 AIR 파일을 두 번 클릭하여 응용 프로그램을 실행할 수 있습니다.

6장: Flex SDK를 사용하여 첫 번째 AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 직접 빠르게 알아보려면 다음에 나와 있는 지침에 따라 Flex SDK를 사용하여 간단한 SWF 기반 AIR "Hello World" 응용 프로그램을 만드십시오. 이 자습서에서는 SDK에서 제공되는 명령줄 도구를 사용하여 AIR 응용 프로그램을 컴파일, 테스트 및 패키지화하는 방법을 보여 줍니다.

시작하려면 런타임을 설치하고 Adobe® Flex™를 설정해야 합니다. 이 자습서에서는 **AMXMLC** 컴파일러, ADL(**AIR Debug Launcher**) 및 ADT(**AIR Developer Tool**)를 사용합니다. 이러한 프로그램은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다(22페이지의 “Flex SDK 설정” 참조).

AIR 응용 프로그램 설명자 파일 만들기

이 단원에서는 응용 프로그램 설명자를 만드는 방법을 설명합니다. 응용 프로그램 설명자는 다음과 같은 구조를 갖는 XML 파일입니다.

```
<application>
    <id>...</id>
    <version>...</version>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
        <visible>…</visible>
        <width>…</width>
        <height>…</height>
    </initialWindow>
</application>
```

1 HelloWorld-app.xml이라는 XML 파일을 만들어 프로젝트 디렉토리에 저장합니다.

2 AIR 네임스페이스 특성을 포함하여 <application> 요소를 추가합니다.

<application xmlns="http://ns.adobe.com/air/application/2.0"> 네임스페이스의 마지막 세그먼트인 "2.0"은 응용 프로그램에 필요한 런타임의 버전을 지정합니다.

3 <id> 요소를 추가합니다.

<id>samples.flex.HelloWorld</id> 응용 프로그램 ID는 제작자 ID(응용 프로그램 패키지에 서명하는 데 사용된 인증서에서 파생됨)와 함께 응용 프로그램을 고유하게 식별합니다. 권장되는 형식은 "com.company.AppName"과 같이 DNS 스타일과 반대 방향인, 점으로 구분된 문자열입니다. 응용 프로그램 ID는 설치, 개인 응용 프로그램 파일 시스템 저장소 디렉토리 액세스, 개인 암호화 저장소 액세스 및 응용 프로그램 간 통신에 사용됩니다.

4 <version> 요소를 추가합니다.

<version>1.0</version> 사용자에게 설치하는 응용 프로그램의 버전을 알려줍니다.

5 <filename> 요소를 추가합니다.

<filename>HelloWorld</filename> 응용 프로그램 실행 파일, 설치 디렉토리 및 운영 체제의 참조와 유사한 항목에 사용되는 이름입니다.

6 초기 응용 프로그램 윈도우의 속성을 지정하는 다음과 같은 자식 요소를 포함하는 <initialWindow> 요소를 추가합니다.

<content>HelloWorld.swf</content> AIR에서 로드할 루트 HTML 파일을 식별합니다.

<visible>true</visible> 윈도우를 즉시 보이게 만듭니다.

<width>400</width> 윈도우 폭을 픽셀 단위로 설정합니다.

<height>200</height> 윈도우 높이를 설정합니다.

7 파일을 저장합니다. 전체 응용 프로그램 설명자 파일은 다음과 같은 형태가 됩니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.0">
    <id>samples.flex.HelloWorld</id>
    <version>0.1</version>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.swf</content>
        <visible>true</visible>
        <width>400</width>
        <height>200</height>
    </initialWindow>
</application>
```

이 예제에서는 사용 가능한 응용 프로그램 속성 중 일부만 설정합니다. 윈도우 크롬, 윈도우 크기, 투명도, 기본 설치 디렉토리, 연결된 파일 형식, 응용 프로그램 아이콘 등을 지정할 수 있는 전체 응용 프로그램 속성 세트는 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오.

응용 프로그램 코드 작성

참고: SWF 기반 AIR 응용 프로그램은 MXML 또는 Adobe® ActionScript® 3.0을 사용하여 정의된 기본 클래스를 사용할 수 있습니다. 아래 예제에서는 MXML 파일을 사용하여 기본 클래스를 정의합니다. 기본 ActionScript 클래스를 사용하여 AIR 응용 프로그램을 만드는 과정도 이와 비슷합니다. 단, MXML 파일을 SWF 파일로 컴파일하는 대신 ActionScript 클래스 파일을 컴파일합니다. ActionScript를 사용할 때는 기본 클래스가 flash.display.Sprite를 확장해야 합니다.

모든 Flex 기반 응용 프로그램과 마찬가지로, Flex 프레임워크로 구축된 AIR 응용 프로그램에는 기본 MXML 파일이 포함되어 있습니다. 그러나 AIR 응용 프로그램은 Application 구성 요소 대신 **WindowedApplication** 구성 요소를 루트 요소로 사용합니다. WindowedApplication 구성 요소는 응용 프로그램과 해당 초기 윈도우를 제어하기 위한 속성, 메서드 및 이벤트를 제공합니다.

다음 절차에 따라 Hello World 응용 프로그램을 만듭니다.

1 텍스트 편집기를 사용하여 HelloWorld.mxml이라는 파일을 만들고 다음 MXML 코드를 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" title="Hello World">
</mx:WindowedApplication>
```

2 그런 다음 응용 프로그램에 Label 구성 요소를 추가합니다(WindowedApplication 태그 안에 배치).

3 Label 구성 요소의 text 속성을 "Hello AIR"로 설정합니다.

4 텍스트를 항상 가운데에 맞추도록 레이아웃 제한 사항을 설정합니다.

다음 예제는 지금까지 작성한 코드를 보여 줍니다.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
    title="Hello World">
        <mx:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</mx:WindowedApplication>
```

이제 전체 응용 프로그램 코드는 다음과 같은 형태가 됩니다.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" title="Hello World">
    <mx:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</mx:WindowedApplication>
```

응용 프로그램 컴파일

응용 프로그램을 실행하고 디버깅하려면 먼저 amxmlc 컴파일러를 사용하여 MXML 코드를 SWF 파일로 컴파일하십시오. amxmlc 컴파일러는 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다. 필요한 경우 컴퓨터의 경로 환경을 Flex SDK bin 디렉토리를 포함하도록 설정할 수 있습니다. 경로를 설정하면 명령줄에서 유틸리티를 더 쉽게 실행할 수 있습니다.

1 명령 셸 또는 터미널을 열고 AIR 응용 프로그램의 프로젝트 폴더로 이동합니다.

2 다음 명령을 입력합니다.

```
amxmlc HelloWorld.mxml
```

amxmlc를 실행하면 HelloWorld.swf가 생성되며, 응용 프로그램의 컴파일된 코드가 이 파일에 포함됩니다.

참고: 응용 프로그램이 컴파일되지 않을 경우 구문 오류나 맞춤법 오류를 수정하십시오. amxmlc 컴파일러를 실행하는 데 사용되는 콘솔 윈도우에 오류 및 경고가 표시됩니다.

자세한 내용은 42페이지의 “AIR용 MXML 및 ActionScript 소스 파일 컴파일”을 참조하십시오.

응용 프로그램 테스트

명령줄에서 응용 프로그램을 실행하고 테스트하려면 ADL(AIR Debug Launcher)을 사용하여 해당 응용 프로그램 설명자 파일을 통해 응용 프로그램을 시작합니다. ADL은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

❖ 명령 프롬프트에서 다음 명령을 입력합니다.

```
adl HelloWorld-app.xml
```

그러면 다음 그림과 비슷한 AIR 응용 프로그램이 표시됩니다.

Label 컨트롤의 horizontalCenter 및 verticalCenter 속성을 사용하여 텍스트를 윈도우 가운데에 배치합니다. 다른 모든 데스크톱 응용 프로그램과 마찬가지로 윈도우를 이동하거나 윈도우의 크기를 조정할 수 있습니다.

자세한 내용은 45페이지의 “ADL(AIR Debug Launcher) 사용”을 참조하십시오.

AIR 설치 파일 만들기

응용 프로그램이 성공적으로 실행되면 ADT 유틸리티를 사용하여 응용 프로그램을 AIR 설치 파일로 패키지화할 수 있습니다. AIR 설치 파일은 사용자에게 응용 프로그램을 배포할 수 있도록 응용 프로그램의 모든 파일을 포함하는 보관 파일입니다. 패키지된 AIR 파일을 설치하려면 먼저 Adobe AIR를 설치해야 합니다.

응용 프로그램 보안을 유지하기 위해 모든 AIR 설치 파일은 디지털 서명이 되어야 합니다. ADT나 다른 인증서 생성 도구에서는 개발 용도로 사용할 기본적인 자체 서명 인증서를 생성할 수 있습니다. 상업용 인증 기관에서 상업용 코드 서명 인증서를 구입할 수도 있습니다. 사용자가 자체 서명된 AIR 파일을 설치할 경우 설치 과정에서 제작자가 "알 수 없음"으로 표시됩니다. 이것은 자체 서명된 인증서가 AIR 파일이 만들어진 이후로 변경되지 않았다는 것만 보장하기 때문입니다. 악의적인 AIR 파일을 자체 서명하여 응용 프로그램으로 제공하는 것 자체를 막을 방법은 없습니다. 따라서, 공개적으로 배포되는 AIR 파일에는 검증 가능한 상용 인증서를 사용하는 것이 좋습니다. AIR 보안 문제에 대한 개요는 AIR 보안(ActionScript 개발자용) 또는 AIR security(HTML 개발자용)을 참조하십시오.

자체 서명 인증서 및 키 쌍 생성

❖ 명령 프롬프트에서 다음 명령을 입력합니다. ADT 실행 파일은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

```
adt -certificate -cn SelfSigned 1024-RSA sampleCert.pfxsamplePassword
```

이 예제에서는 인증서에 설정할 수 있는 최소한의 특성만 사용합니다. **기울임체**로 표시된 매개 변수에는 원하는 값을 사용할 수 있습니다. 키 유형은 **1024-RSA** 또는 **2048-RSA**여야 합니다(84페이지의 “AIR 파일에 디지털 서명” 참조).

AIR 설치 파일 만들기

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -package -storetype pkcs12 -keystore sampleCert.pfx HelloWorld.air
HelloWorld-app.xml HelloWorld.swf
```

키 저장소 파일의 암호를 입력하라는 메시지가 나타납니다. 암호를 입력하고 Enter 키를 누릅니다. 보안상의 이유로 암호 문자는 표시되지 않습니다.

HelloWorld.air 인수는 ADT가 생성한 AIR 파일입니다. HelloWorld-app.xml은 응용 프로그램 설명자 파일입니다. 그 이후의 인수는 응용 프로그램에 사용되는 파일입니다. 이 예제에서는 파일 세 개만 사용하지만 원하는 수의 파일 및 디렉토리를 포함할 수 있습니다.

AIR 패키지가 만들어지면 해당 패키지 파일을 두 번 클릭하여 응용 프로그램을 설치하고 실행할 수 있습니다. 셸이나 명령 윈도우에서 명령으로 AIR 파일 이름을 입력할 수도 있습니다.

자세한 내용은 48페이지의 “ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지”를 참조하십시오.

7장: Flash Professional을 사용하여 첫 번째 AIR 응용 프로그램 만들기

이 장의 지시에 따라 Adobe® Flash® Professional을 사용하여 간단한 “Hello World” AIR 응용 프로그램을 직접 만들고 패키지화하면 Adobe® AIR®의 작동 방식을 빠르고 쉽게 확인해 볼 수 있습니다.

Flash에서 Hello World 응용 프로그램 만들기

Flash에서 Adobe AIR 응용 프로그램을 만드는 것은 다른 FLA 파일을 만드는 것과 비슷합니다. 단, 시작 화면에서 먼저 Flash 파일(Adobe AIR)을 만드는 것으로 작업을 시작합니다. 그런 다음 응용 프로그램 및 설치 프로그램 설정을 지정하고 AIR 응용 프로그램을 설치하면 됩니다. 다음 절차를 따라 Flash Professional로 간단한 Hello World 응용 프로그램을 만들어 보십시오.

Hello World 응용 프로그램을 만들려면

1. Flash를 시작합니다.
2. 시작 화면에서 [Flash 파일(Adobe AIR)]을 클릭하고 Adobe AIR 제작 설정을 지정하여 빈 FLA 파일을 만듭니다.
3. [도구] 패널에서 [텍스트] 도구를 선택하여 스테이지 가운데에 정적 텍스트 필드(기본값)를 만듭니다. 15-20자가 들어갈 정도로 넓게 만드십시오.
4. 텍스트 필드에 “Hello World”를 입력합니다.
5. 파일을 저장하고 이름(예: helloAIR)을 지정합니다.

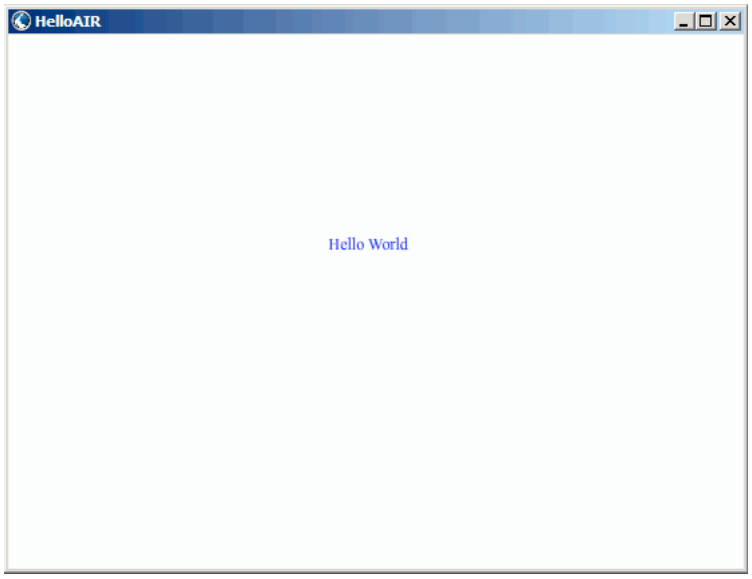
응용 프로그램 테스트

1. Ctrl+Enter를 누르거나 [컨트롤] -> [동영상 테스트]를 선택하여 Adobe AIR에서 응용 프로그램을 테스트합니다.
2. [동영상 디버그] 기능을 사용하려면 먼저 응용 프로그램에 ActionScript 코드를 추가해야 합니다. 다음과 같은 trace 문을 간단히 추가해 볼 수 있습니다.

   ```
   trace("Running AIR application using Debug Movie");
   ```

3. Ctrl+Shift+Enter를 누르거나 [컨트롤] -> [동영상 디버그]를 선택하여 [동영상 디버그]로 응용 프로그램을 실행합니다.

Hello World 응용 프로그램은 다음 그림과 같습니다.



응용 프로그램 패키지화

1 [파일] > [제작]을 선택합니다.

2 기존 디지털 인증서로 Adobe AIR 패키지를 서명하거나, 다음 단계를 따라 자체 서명 인증서를 만듭니다.

   a [만들기] 버튼을 클릭하여 [자체 서명된 디지털 인증서 만들기] 대화 상자를 엽니다.

   b [제작자 이름], [조직 구성 단위], [조직 이름], [전자 메일], [국가], [암호] 및 [암호 확인] 필드에 값을 입력합니다.

   c 인증서 유형을 지정합니다. 인증서 [유형] 옵션은 보안 수준을 가리킵니다. [1024-RSA]는 1024비트 키(보안 수준 낮음)를 사용하고 [2048-RSA]는 2048비트 키(보안 수준 높음)를 사용합니다.

   d [다른 이름으로 저장] 필드에 값을 입력하거나 [찾아보기...] 버튼을 클릭하고 폴더 위치를 찾아 인증서 파일의 정보를 저장합니다(예: **C:/Temp/mycert.pfx**). 작업을 마치면 [확인]을 클릭합니다.

   e [디지털 서명] 대화 상자가 다시 나타납니다. 만든 자체 서명된 인증서의 경로 및 파일 이름이 [인증서] 텍스트 상자에 표시됩니다. 표시되지 않는 경우, 경로 및 파일 이름을 입력하거나 [찾아보기] 버튼을 클릭하여 파일을 찾아 선택합니다.

   f c 단계에서 지정한 암호와 동일한 암호를 [디지털 서명] 대화 상자의 [암호] 텍스트 필드에 입력합니다. Adobe AIR 응용 프로그램 서명에 대한 자세한 내용은 84페이지의 “AIR 파일에 디지털 서명”을 참조하십시오.

3 응용 프로그램 및 설치 프로그램 파일을 만들려면 [파일 제작] 버튼을 클릭합니다. Flash CS4 및 CS5의 경우 [확인] 버튼을 클릭합니다. AIR 파일을 만들기 전에 [동영상 테스트] 또는 [동영상 디버그]를 실행하여 SWF 파일 및 application.xml 파일을 만들어야 합니다.

4 응용 프로그램을 설치하려면 응용 프로그램을 저장한 폴더에서 AIR 파일(**application**.air)을 두 번 클릭합니다.

5 [응용 프로그램 설치] 대화 상자에서 [설치] 버튼을 클릭합니다.

6 [설치 환경 설정] 및 [위치] 설정을 검토하고 '설치 후 응용 프로그램 시작' 체크 상자를 선택합니다. 그런 다음 [계속]을 클릭합니다.

7 '설치가 완료되었습니다.' 메시지가 나타나면 [완료]를 클릭합니다.

8장: Dreamweaver를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 빠르게 살펴보려면 다음 지침에 따라 Adobe® AIR® Extension for Dreamweaver®를 사용하여 간단한 HTML 기반 AIR "Hello World" 응용 프로그램을 만들고 패키지해 보십시오.

아직 Adobe AIR를 설치하지 않았다면 www.adobe.com/go/air_kr에서 다운로드하여 설치하십시오.

Adobe AIR Extension for Dreamweaver를 설치하는 방법은 Adobe AIR Extension for Dreamweaver 설치를 참조하십시오.

시스템 요구 사항을 비롯한 Extension에 대한 개요는 AIR Extension for Dreamweaver를 참조하십시오.

응용 프로그램 파일 준비

Adobe AIR 응용 프로그램의 시작 페이지와 모든 관련 페이지가 Dreamweaver 사이트에 정의되어 있어야 합니다.

1 Dreamweaver를 시작하고 사이트를 정의했는지 확인합니다.

2 [파일] > [새로 만들기]를 선택한 다음 [페이지 유형] 열에서 HTML을 선택하고 [레이아웃] 열에서 [없음]을 선택한 후 [만들기]를 클릭하여 새 HTML 페이지를 엽니다.

3 새 페이지에서 **Hello World!**를 입력합니다.

   이 예제는 매우 단순합니다. 원한다면 텍스트에 좋아하는 스타일을 지정하고, 페이지에 더 많은 내용을 추가하고, 다른 페이지를 이 시작 페이지에 링크할 수 있습니다.

4 [파일] > [저장]을 차례로 클릭한 다음 페이지를 hello_world.html로 저장합니다. 파일을 Dreamweaver 사이트에 저장했는지 확인합니다.

Dreamweaver 사이트에 대한 자세한 내용은 Dreamweaver 도움말을 참조하십시오.

Adobe AIR 응용 프로그램 만들기

1 hello_world.html 페이지가 Dreamweaver 문서 윈도우에 열려 있어야 합니다. 만드는 방법은 이전 단원을 참조하십시오.

2 [사이트] > [Air 응용 프로그램 설정]을 선택합니다.

   [AIR 응용 프로그램 및 설정] 대화 상자에 있는 대부분의 필수 설정은 자동으로 입력됩니다. 하지만 응용 프로그램의 초기 내용(또는 시작 페이지)은 직접 선택해야 합니다.

3 [초기 내용] 옵션 옆에 있는 [찾아보기] 버튼을 클릭하고 hello_world.html 페이지를 찾아 선택합니다.

4 [디지털 서명] 옵션 옆에 있는 [설정] 버튼을 클릭합니다.

   디지털 서명은 소프트웨어 작성자가 만든 이후로 응용 프로그램의 코드가 변경되거나 손상되지 않았음을 보장하며 모든 Adobe AIR 응용 프로그램에 필요합니다.

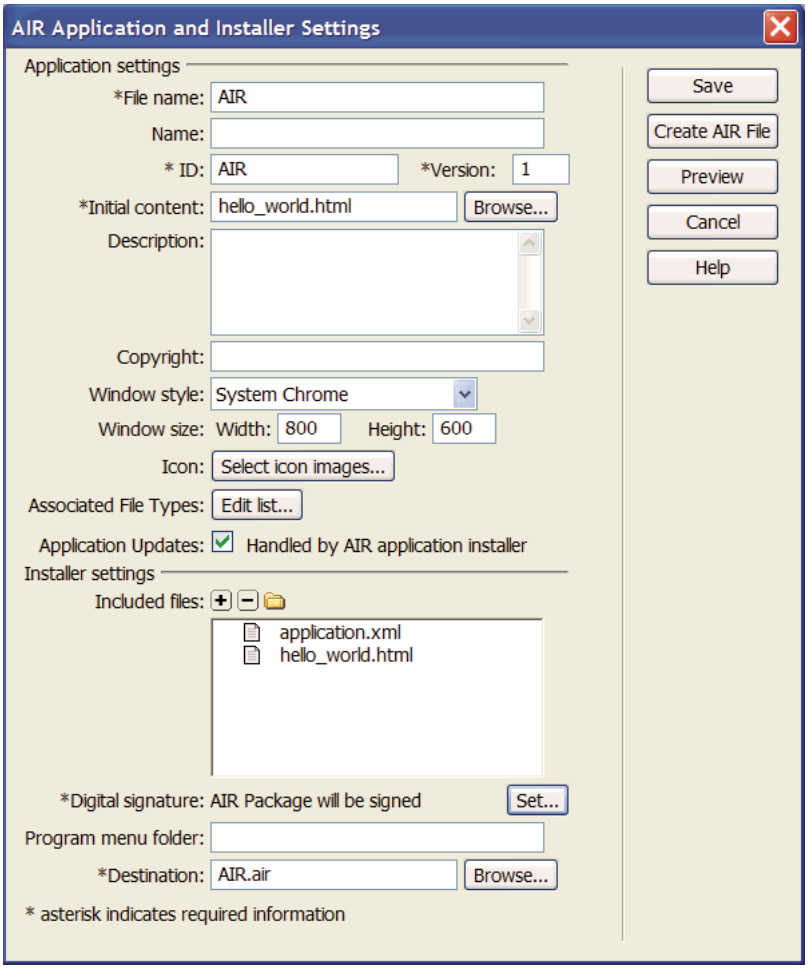
5 [디지털 서명] 대화 상자에서 [디지털 인증서로 AIR 패키지 서명]을 선택한 다음 [만들기] 버튼을 클릭합니다. 디지털 인증서에 액세스할 수 있는 경우에는 [찾아보기] 버튼을 클릭하여 선택할 수도 있습니다.

6 [자체 서명된 디지털 인증서] 대화 상자의 필수 필드에 필요한 정보를 모두 입력합니다. 이름, 암호, 암호 확인을 입력한 다음 디지털 인증서 파일의 이름을 입력해야 합니다. 디지털 인증서는 사이트 루트에 저장됩니다.

7 [확인]을 클릭하여 [디지털 서명] 대화 상자로 돌아갑니다.

8 [디지털 서명] 대화 상자에서 디지털 인증서에 지정한 암호를 입력하고 [확인]을 클릭합니다.

작업을 마치면 [AIR 응용 프로그램 및 설치 프로그램 설정] 대화 상자가 다음과 같이 표시됩니다.



모든 대화 상자 옵션 및 해당 옵션을 편집하는 방법에 대해서는 Dreamweaver에서 AIR 응용 프로그램 만들기를 참조하십시오.

9 [AIR 파일 만들기] 버튼을 클릭합니다.

Dreamweaver에서 Adobe AIR 응용 프로그램을 만들어 사이트 루트 폴더에 저장합니다. 또한 application.xml 파일을 만들어 같은 위치에 저장합니다. 이 파일은 응용 프로그램의 다양한 속성을 정의하는 매니페스트로 사용됩니다.

데스크톱에 응용 프로그램 설치

이제 응용 프로그램 파일을 만들었으므로 원하는 데스크톱에 설치할 수 있습니다.

1 Adobe AIR 응용 프로그램 파일을 Dreamweaver 사이트에서 자신의 데스크톱으로 이동하거나 다른 데스크톱으로 이동합니다.

이 단계는 선택 사항입니다. 원하는 경우 Dreamweaver 사이트 디렉토리에서 사용자 컴퓨터에 직접 새 응용 프로그램을 설치할 수 있습니다.

2 응용 프로그램 실행 파일(.air 파일)을 두 번 클릭하여 응용 프로그램을 설치합니다.

Adobe AIR 응용 프로그램 미리 보기

AIR 응용 프로그램의 일부가 되는 페이지는 언제라도 미리 볼 수 있습니다. 즉, 응용 프로그램을 설치했을 때 어떻게 보이는지 확인하기 위해 응용 프로그램을 패키지화할 필요가 없습니다.

1 Dreamweaver 문서 윈도우에 hello_world.html 페이지가 열려 있어야 합니다.

2 [문서] 툴바에서 [미리 보기]/[브라우저에서 디버그] 버튼을 클릭한 다음 [AIR에서 미리 보기]를 선택합니다.

Ctrl+Shift+F12(Windows) 또는 Cmd+Shift+F12(Macintosh)를 눌러 이 작업을 수행할 수도 있습니다.

이 페이지를 미리 보면 사용자가 데스크톱에 응용 프로그램을 설치한 후 응용 프로그램의 시작 페이지로 표시되는 내용을 확인할 수 있습니다.

9장: AIR SDK를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기

Adobe® AIR® 작동 방식을 빠르게 살펴보려면 다음 지침에 따라 간단한 HTML 기반 AIR "Hello World" 응용 프로그램을 만들고 패키지 해보십시오.

시작하려면 런타임을 설치하고 AIR SDK를 설정해야 합니다. 이 자습서에서는 ADL(**AIR Debug Launcher**)과 ADT(**AIR Developer Tool**)를 사용합니다. ADL 및 ADT는 명령줄 유틸리티 프로그램이며 AIR SDK의 bin 디렉토리에서 찾을 수 있습니다(21페이지의 "AIR SDK 설치" 참조). 이 자습서에서는 학습자가 명령줄에서 프로그램을 실행하는 데 익숙하며 운영 체제에 필요한 경로 환경 변수를 설정하는 방법을 알고 있다고 가정합니다.

참고: Adobe® Dreamweaver® 사용자는 35페이지의 "Dreamweaver를 사용하여 첫 번째 HTML 기반 AIR 응용 프로그램 만들기"를 읽어보십시오.

프로젝트 파일 만들기

모든 HTML 기반 AIR 프로젝트에는 응용 프로그램 메타데이터를 지정하는 응용 프로그램 설명자 파일과 최상위 HTML 페이지의 두 가지 파일이 포함되어야 합니다. 이러한 필수 파일과 함께 이 프로젝트에는 AIR API 클래스에서 편리하게 사용할 수 있는 별칭 변수를 정의하는 JavaScript 코드 파일인 AIRAliases.js가 포함되어 있습니다.

1. 프로젝트 파일을 저장할 HelloWorld 디렉토리를 만듭니다.
2. HelloWorld-app.xml이라는 XML 파일을 만듭니다.
3. HelloWorld.html이라는 HTML 파일을 만듭니다.
4. AIR SDK의 frameworks 폴더에 있는 AIRAliases.js를 프로젝트 디렉토리에 복사합니다.

AIR 응용 프로그램 설명자 파일 만들기

AIR 응용 프로그램을 작성하려면 다음과 같은 구조로 XML 응용 프로그램 설명자 파일을 만들어야 합니다.

```
<application>
    <id>…</id>
    <version>…</version>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
        <visible>…</visible>
        <width>…</width>
        <height>…</height>
    </initialWindow>
</application>
```

1. HelloWorld-app.xml을 편집할 수 있도록 엽니다.
2. AIR 네임스페이스 특성을 포함하는 루트 <application> 요소를 추가합니다.

   <application xmlns="http://ns.adobe.com/air/application/2.0"> 네임스페이스의 마지막 세그먼트인 "2.0"은 응용 프로그램에 필요한 런타임 버전을 지정합니다.
3. <id> 요소를 추가합니다.

<id>examples.html.HelloWorld</id> 응용 프로그램 ID는 제작자 ID(응용 프로그램 패키지에 서명하는 데 사용된 인증서에서 파생됨)와 함께 응용 프로그램을 고유하게 식별합니다. 응용 프로그램 ID는 설치, 개인 응용 프로그램 파일 시스템 저장소 디렉토리 액세스, 개인 암호화 저장소 액세스 및 응용 프로그램 간 통신에 사용됩니다.

4 <version> 요소를 추가합니다.

<version>0.1</version> 사용자에게 설치하는 응용 프로그램의 버전을 알려줍니다.

5 <filename> 요소를 추가합니다.

<filename>HelloWorld</filename> 응용 프로그램 실행 파일, 설치 디렉토리 및 운영 체제에서 응용 프로그램에 대한 기타 참조로 사용되는 이름입니다.

6 초기 응용 프로그램 윈도우의 속성을 지정하는 다음과 같은 자식 요소를 포함하는 <initialWindow> 요소를 추가합니다.

<content>HelloWorld.html</content> AIR에서 로드할 루트 HTML 파일을 식별합니다.

<visible>true</visible> 윈도우를 즉시 보이게 만듭니다.

<width>400</width> 윈도우 폭을 픽셀 단위로 설정합니다.

<height>200</height> 윈도우 높이를 설정합니다.

7 파일을 저장합니다. 완성된 응용 프로그램 설명자 파일은 다음과 같아야 합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.0">
    <id>examples.html.HelloWorld</id>
    <version>0.1</version>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.html</content>
        <visible>true</visible>
        <width>400</width>
        <height>200</height>
    </initialWindow>
</application>
```

이 예제에서는 사용 가능한 응용 프로그램 속성 중 일부만 설정합니다. 윈도우 크롬, 윈도우 크기, 투명도, 기본 설치 디렉토리, 연결된 파일 유형, 응용 프로그램 아이콘 등을 지정할 수 있는 전체 응용 프로그램 속성을 보려면 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오.

응용 프로그램 HTML 페이지 만들기

AIR 응용 프로그램의 기본 파일로 사용할 간단한 HTML 페이지를 만들어야 합니다.

1 HelloWorld.html 파일을 편집할 수 있도록 엽니다. 다음 HTML 코드를 추가합니다.

```
<html>
<head>
    <title>Hello World</title>
</head>
<body onLoad="appLoad()">
    <h1>Hello World</h1>
</body>
</html>
```

2 HTML의 <head> 섹션에서 AIRAliases.js 파일을 가져옵니다.

```
<script src="AIRAliases.js" type="text/javascript"></script>
```

AIR가 HTML 윈도우 객체에 runtime이라는 속성을 정의합니다. 이 runtime 속성은 클래스의 정규화된 패키지 이름을 사용하여 기본 제공 AIR 클래스에 액세스하는 방법을 제공합니다. 예를 들어, AIR File 객체를 만들려면 JavaScript에 다음과 같은 문을 추가합니다.

```
var textFile = new runtime.flash.filesystem.File("app:/textfile.txt");
```

AIRAliases.js 파일은 자주 사용하는 AIR API를 편리하게 사용할 수 있도록 별칭을 정의합니다. AIRAliases.js를 사용하면 File 클래스에 대한 참조를 다음과 같이 줄일 수 있습니다.

```
var textFile = new air.File("app:/textfile.txt");
```

3 AIRAliases 스크립트 태그 아래에 onLoad 이벤트를 처리하는 JavaScript 함수가 포함된 다른 스크립트 태그를 추가합니다.

```
<script type="text/javascript">
function appLoad(){
    air.trace("Hello World");
}
</script>
```

appLoad() 함수는 단순히 air.trace() 함수를 호출합니다. 이제 ADL을 사용하여 응용 프로그램을 실행하면 명령 콘솔에 추적 메시지가 출력됩니다. Trace 문은 디버깅 시 매우 유용하게 사용할 수 있습니다.

4 파일을 저장합니다.

이제 HelloWorld.html 파일이 다음과 같아야 합니다.

```
<html>
<head>
    <title>Hello World</title>
    <script type="text/javascript" src="AIRAliases.js"></script>
    <script type="text/javascript">
        function appLoad(){
            air.trace("Hello World");
        }
    </script>
</head>
<body onLoad="appLoad()">
    <h1>Hello World</h1>
</body>
</html>
```

응용 프로그램 테스트

명령줄에서 응용 프로그램을 실행하고 테스트하려면 ADL(AIR Debug Launcher) 유틸리티를 사용합니다. ADL 실행 파일은 AIR SDK의 bin 디렉토리에서 찾을 수 있습니다. AIR SDK를 아직 설치하지 않은 경우 21페이지의 “AIR SDK 설치”를 참조하십시오.

1 명령 콘솔이나 셸을 엽니다. 이 프로젝트용으로 만든 디렉토리로 변경합니다.

2 다음 명령을 실행합니다.

```
adl HelloWorld-app.xml
```

응용 프로그램을 표시하는 AIR 윈도우가 열립니다. 또한 콘솔 윈도우에 air.trace() 호출의 결과 메시지가 표시됩니다.

자세한 내용은 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오.

AIR 설치 파일 만들기

응용 프로그램이 성공적으로 실행되면 ADT 유틸리티를 사용하여 응용 프로그램을 AIR 설치 파일로 패키지화할 수 있습니다. AIR 설치 파일은 사용자에게 응용 프로그램을 배포할 수 있도록 응용 프로그램의 모든 파일을 포함하는 보관 파일입니다. 패키지된 AIR 파일을 설치하려면 먼저 Adobe AIR를 설치해야 합니다.

응용 프로그램 보안을 유지하기 위해 모든 AIR 설치 파일은 디지털 서명이 되어야 합니다. ADT나 다른 인증서 생성 도구에서는 개발 용도로 사용할 기본적인 자체 서명 인증서를 생성할 수 있습니다. 또한 VeriSign이나 Thawte 같은 기업 인증 기관에서 상업적인 코드로 서명하는 인증서를 구매할 수도 있습니다. 사용자가 자체 서명된 AIR 파일을 설치할 경우 설치 과정에서 제작자가 "알 수 없음"으로 표시됩니다. 이것은 자체 서명된 인증서가 AIR 파일이 만들어진 이후로 변경되지 않았다는 것만 보장하기 때문입니다. 악의적인 AIR 파일을 자체 서명하여 응용 프로그램으로 제공하는 것 자체를 막을 방법은 없습니다. 따라서, 공개적으로 배포되는 AIR 파일에는 검증 가능한 상용 인증서를 사용하는 것이 좋습니다. AIR 보안 문제에 대한 개요는 AIR 보안 (ActionScript 개발자용) 또는 AIR security(HTML 개발자용)을 참조하십시오.

자체 서명 인증서 및 키 쌍 생성

❖ 명령 프롬프트에서 다음 명령을 입력합니다. ADT 실행 파일은 AIR SDK의 bin 디렉토리에 있습니다.

```
adt -certificate -cn SelfSigned 1024-RSA sampleCert.pfx samplePassword
```

ADT가 인증서와 관련 개인 키가 들어 있는 **sampleCert.pfx**라는 키 저장소 파일을 생성합니다.

이 예제에서는 인증서에 설정할 수 있는 최소한의 특성만 사용합니다. **기울임체**로 표시된 매개 변수에는 원하는 값을 사용할 수 있습니다. 키 유형은 **1024-RSA** 또는 **2048-RSA**여야 합니다(84페이지의 “AIR 파일에 디지털 서명” 참조).

AIR 설치 파일 만들기

❖ 명령 프롬프트에서 다음 명령을 한 줄에 입력합니다.

```
adt -package -storetype pkcs12 -keystore sampleCert.pfx HelloWorld.air
HelloWorld-app.xml HelloWorld.html AIRAliases.js
```

키 저장소 파일의 암호를 입력하라는 메시지가 나타납니다.

HelloWorld.air 인수는 ADT가 생성한 AIR 파일입니다. HelloWorld-app.xml은 응용 프로그램 설명자 파일입니다. 그 이후의 인수는 응용 프로그램에 사용되는 파일입니다. 이 예제에서는 파일 두 개만 사용하지만 원하는 모든 파일과 디렉토리를 포함할 수 있습니다.

AIR 패키지가 만들어지면 해당 패키지 파일을 두 번 클릭하여 응용 프로그램을 설치하고 실행할 수 있습니다. 셸이나 명령 윈도우에서 명령으로 AIR 파일 이름을 입력할 수도 있습니다.

다음 단계

AIR에서 HTML 및 JavaScript 코드는 대개 일반적인 웹 브라우저에서와 동일하게 동작합니다. 실제로 AIR는 Safari 웹 브라우저에 사용되는 것과 동일한 WebKit 렌더링 엔진을 사용합니다. 하지만 AIR에서 HTML 응용 프로그램을 개발할 때 반드시 알아야 할 중요한 차이점이 있습니다. 이러한 차이점 및 기타 중요한 주제에 대한 자세한 내용은 Programming HTML and JavaScript를 참조하십시오.

10장: 명령줄 도구를 사용하여 AIR 응용 프로그램 만들기

Adobe® AIR® 명령줄 도구를 사용하여 Adobe AIR 응용 프로그램을 컴파일, 테스트, 패키지 및 서명할 수 있습니다. AIR 명령줄 도구는 Adobe® Flex™와 AIR SDK(http://www.adobe.com/go/learn_air_download_AIRSDK_kr)에 모두 포함되어 있습니다.

AIR 및 Flex SDK는 AIR 응용 프로그램을 개발하는 데 사용할 수 있는 다음과 같은 명령줄 도구를 제공합니다.

컴파일 Flex MXML 및 ActionScript 소스 파일을 컴파일하려면 **amxmlc** 또는 **acompc** 컴파일러를 사용합니다. Flash Professional 프로젝트의 경우 동영상을 제작하여 프로젝트를 컴파일합니다. HTML 기반 AIR 응용 프로그램은 컴파일할 필요가 없습니다.

참고: amxmlc 및 acompc 도구는 AIR SDK에는 포함되지 않으며 Flex SDK에만 포함됩니다.

테스트 및 디버그 **ADL(AIR Debug Launcher)**을 사용하여 AIR 응용 프로그램을 테스트하고 디버깅합니다. ADL을 사용하면 응용 프로그램을 패키지화하여 설치하지 않고도 AIR 프로젝트를 실행할 수 있습니다. ADL은 추적 메시지와 오류 출력을 명령줄 콘솔에 출력합니다. 또한 ADL을 사용하여 **FDB(Flash Debugger)** 도구에 연결하면 좀 더 복잡한 문제를 디버깅할 수 있습니다.

참고: FDB 도구는 AIR SDK에는 포함되지 않으며 Flex SDK에만 포함됩니다. Flex SDK는 http://opensource.adobe.com에서 사용할 수 있습니다.

패키지 **ADT(AIR Developer Tool)**를 사용하면 완성된 AIR 응용 프로그램을 지원되는 모든 컴퓨터 시스템에 설치할 수 있는 AIR 설치 프로그램 파일(.air)이나 ADT 도구를 실행 중인 컴퓨터와 동일한 유형의 컴퓨터에만 설치할 수 있는 기본 설치 프로그램 파일로 패키지화할 수 있습니다.

서명 ADT를 사용하면 AIR 응용 프로그램에 제작 인증서로 서명을 할 수 있습니다. 응용 프로그램 패키지를 만들면서 동시에 응용 프로그램에 서명할 수 있습니다. 또한 패키징을 두 단계로 나눌 수 있는데, 이렇게 나누면 코드 서명 인증서에 대한 액세스와 개인 키에 대한 액세스를 철저하게 관리하는 경우에 유용합니다. 모든 AIR 응용 프로그램에는 서명이 필요합니다.

참고: Adobe Flash Builder, Adobe Flash Professional 및 Aptana Studio 등을 비롯한 대부분의 통합 개발 환경에서 AIR 응용 프로그램을 컴파일, 디버깅 및 패키지화할 수 있습니다. 이와 같은 개발 환경을 사용하고 있다면 이러한 공통적인 작업에 명령줄 도구를 사용할 필요가 없습니다. 하지만 통합 개발 환경에서 지원하지 않는 기능의 경우에는 여전히 명령줄 도구를 사용해야 합니다. 또한 명령줄 도구를 사용하여 자동 빌드 프로세스를 만들 수 있습니다.

AIR용 MXML 및 ActionScript 소스 파일 컴파일

다음과 같이 명령줄 MXML 컴파일러(amxmlc)를 사용하여 AIR 응용 프로그램의 Adobe® ActionScript® 3.0 및 MXML 에셋을 컴파일할 수 있습니다. HTML 기반 응용 프로그램은 컴파일할 필요가 없습니다. Flash Professional에서 SWF를 컴파일하려면 동영상을 SWF 파일로 제작하기만 하면 됩니다.

amxmlc를 사용하는 기본 명령줄 패턴은 다음과 같습니다.

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

여기서 **[compiler options]**은 AIR 응용 프로그램을 컴파일하는 데 사용하는 명령줄 옵션을 지정합니다.

amxmlc 명령은 표준 Flex mxmlc 컴파일러와 추가 매개변수 +configname=air을 호출합니다. 이 매개변수는 컴파일러에게 flex-config.xml 파일 대신 air-config.xml 파일을 사용하도록 지시합니다. amxmlc를 사용하는 것은 mxmlc를 사용하는 것과 동일합니다. mxmlc 컴파일러 및 구성 파일 형식에 대해서는 Flex 3 문서 라이브러리의 Flex 3 응용 프로그램 만들기 및 배포에서 설명합니다.

컴파일러는 일반적으로 AIR 응용 프로그램을 컴파일하는 데 필요한 AIR 및 Flex 라이브러리를 지정하는 air-config.xml 구성 파일을 로드합니다. 또한 전역 구성에 대한 추가 옵션을 재정의하거나 추가하는 데 프로젝트 레벨의 로컬 구성 파일을 사용할 수도 있습니다. 일반적으로 로컬 구성 파일을 만드는 가장 쉬운 방법은 전역 버전의 사본을 편집하는 것입니다. -load-config 옵션을 사용하여 로컬 파일을 로드할 수 있습니다.

-load-config=project-config.xml 전역 옵션을 재정의합니다.

-load-config+=project-config.xml -library-path 옵션과 같이 값 이외의 것을 사용하는 전역 옵션에 추가 값을 추가합니다. 단일 값만을 사용하는 전역 옵션이 재정의됩니다.

로컬 구성 파일에 대한 특별한 이름 지정 규칙을 사용할 경우 amxmlc 컴파일러는 로컬 파일을 자동으로 로드합니다. 예를 들어 기본 MXML 파일이 RunningMan.mxml인 경우 로컬 구성 파일의 이름을 RunningMan-config.xml로 지정합니다. 이제 응용 프로그램을 컴파일하려면 다음을 입력하기만 하면 됩니다.

```
amxmlc RunningMan.mxml
```

파일 이름이 컴파일된 MXML 파일 이름과 일치하므로 RunningMan-config.xml이 자동으로 로드됩니다.

amxmlc 예제

다음 예제에서는 amxmlc 컴파일러의 사용을 보여 줍니다. 여기서는 응용 프로그램의 ActionScript 및 MXML 에셋만 컴파일됩니다.

AIR MXML 파일 컴파일:

```
amxmlc myApp.mxml
```

컴파일 및 출력 이름 설정:

```
amxmlc –output anApp.swf -- myApp.mxml
```

AIR ActionScript 파일 컴파일:

```
amxmlc myApp.as
```

컴파일러 구성 파일 지정:

```
amxmlc –load-config config.xml -- myApp.mxml
```

다른 구성 파일에서 추가 옵션 추가:

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

명령줄에서 이미 구성 파일에 있는 라이브러리에 라이브러리 추가:

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

구성 파일을 사용하지 않고 AIR MXML 파일 컴파일(Win):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, ^
[AIR SDK]/frameworks/libs/air/airframework.swc, ^
-library-path [Flex 3 SDK]/frameworks/libs/framework.swc ^
-- myApp.mxml
```

구성 파일을 사용하지 않고 AIR MXML 파일 컴파일(Mac OS X 또는 Linux):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, \
[AIR SDK]/frameworks/libs/air/airframework.swc, \
-library-path [Flex 3 SDK]/frameworks/libs/framework.swc \
-- myApp.mxml
```

런타임 공유 라이브러리를 사용하기 위한 AIR MXML 파일 컴파일:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf -- myApp.mxml
```

mxmlc.jar을 포함하도록 클래스 경로를 설정하여 Java에서 컴파일:

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional compiler options] -
- myApp.mxml
```

flexlib 옵션은 컴파일러에서 flex_config.xml 파일의 위치를 찾을 수 있도록 활성화시킴으로써 Flex SDK 프레임워크 디렉토리의 위치를 식별합니다.

클래스 경로를 설정하지 않고 Java에서 컴파일:

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air [additional compiler options] -- myApp.mxml
```

Apache Ant를 사용하는 컴파일러를 호출하려면(예제에서는 Java 작업을 사용하여 mxmlc.jar 실행)

```
<property name="SDK_HOME" value="C:/Flex3SDK"/>
<property name="MAIN_SOURCE_FILE" value="src/myApp.mxml"/>
<property name="DEBUG" value="true"/>
<target name="compile">
    <java jar="${MXMLC.JAR}" fork="true" failonerror="true">
        <arg value="-debug=${DEBUG}"/>
        <arg value="+flexlib=${SDK_HOME}/frameworks"/>
        <arg value="+configname=air"/>
        <arg value="-file-specs=${MAIN_SOURCE_FILE}"/>
    </java>
</target>
```

AIR 구성 요소 또는 라이브러리 컴파일 (Flex)

구성 요소 컴파일러인 acompc를 사용하여 AIR 라이브러리 및 독립 구성 요소를 컴파일합니다. acompc 구성 요소 컴파일러는 amxmlc 컴파일러처럼 동작하지만 다음과 같은 예외가 있습니다.

- 라이브러리 또는 구성 요소에 포함시키려면 코드 베이스 내에 어떤 클래스인지를 지정해야 합니다.
- acompc는 로컬 구성 파일을 자동으로 검색하지 않습니다. 프로젝트 구성 파일을 사용하려면 –load-config 옵션을 사용해야 합니다.

acompc 명령은 표준 Flex compc 구성 요소 컴파일러를 호출하지만 해당 구성 옵션은 air-config.xml 파일에서 로드하며, flex-config.xml 파일에서 로드하지는 않습니다.

구성 요소 컴파일러 구성 파일

로컬 구성 파일을 사용하면 명령줄에서 소스 경로 및 클래스 이름을 입력하지 않아도 됩니다. 이는 잘못 입력될 가능성을 방지합니다. 로컬 구성 파일을 로드하려면 -load-config 옵션을 acompc 명령줄에 추가합니다.

다음 예제에서는 com.adobe.samples.particles 패키지의 두 개의 클래스인 ParticleManager 및 Particle을 사용하여 라이브러리를 구축하기 위한 구성을 보여 줍니다. 클래스 파일은 source/com/adobe/samples/particles 폴더에 들어 있습니다.

```
<flex-config>
    <compiler>
        <source-path>
            <path-element>source</path-element>
        </source-path>
    </compiler>
    <include-classes>
        <class>com.adobe.samples.particles.ParticleManager</class>
        <class>com.adobe.samples.particles.Particle</class>
    </include-classes>
</flex-config>
```

ParticleLib-config.xml이라는 구성 파일을 사용하여 라이브러리를 컴파일하려면 다음과 같이 입력합니다.

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

명령줄에서 동일한 명령을 실행하려면 다음과 같이 입력합니다.

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

한 줄에 명령 전체를 입력하거나 해당 명령 셸의 줄 연결 문자를 사용합니다.

acompc 예제

이 예제에서는 myLib-config.xml이라는 구성 파일을 사용한다고 가정합니다.

AIR 구성 요소 또는 라이브러리 컴파일:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

런타임 공유 라이브러리 컴파일:

```
acompc -load-config myLib-config.xml -directory -output lib
```

이 명령을 실행하려면 lib 폴더가 존재해야 하며 비어 있어야 합니다.

ADL(AIR Debug Launcher) 사용

ADL(AIR Debug Launcher)을 사용하여 개발 중에 SWF 기반 응용 프로그램과 HTML 기반 응용 프로그램을 모두 실행합니다. ADL을 사용하면 먼저 응용 프로그램을 패키지 및 설치하지 않고도 실행할 수 있습니다. 기본적으로 ADL은 SDK에 포함된 런타임을 사용하므로 ADL을 사용하기 위해 런타임을 별도로 설치할 필요가 없습니다.

ADL은 trace 문 및 런타임 오류를 표준 출력으로 인쇄하지만 중단점 또는 기타 디버깅 기능은 지원하지 않습니다. 복잡한 디버깅 문제에는 Flash Debugger 또는 Flash Builder나 Aptana Studio 같은 통합 개발 환경을 사용합니다.

ADL을 사용하여 응용 프로그램 시작

ADL로 응용 프로그램을 실행하려면 다음과 같은 패턴을 사용하십시오.

```
adl application.xml
```

여기서 application.xml은 응용 프로그램의 응용 프로그램 설명자 파일입니다.

ADL의 전체 구문은 다음과 같습니다.

```
adl [-runtime runtime-directory] [-pubid publisher-id] [-nodebug] [-profile profileName] application.xml
[root-directory] [-- arguments]
```

-runtime runtime-directory 사용할 런타임이 포함된 디렉토리를 지정합니다. 지정하지 않을 경우 ADL 프로그램과 같은 SDK의 런타임 디렉토리가 사용됩니다. SDK 폴더에서 ADL을 다른 위치로 이동할 경우 런타임 디렉토리를 지정해야 합니다. Windows 및 Linux에서는 Adobe AIR 디렉토리가 포함된 디렉토리를 지정하십시오. Mac OS X에서는 Adobe AIR.framework가 포함된 디렉토리를 지정하십시오.

-pubid publisher-id 이 실행에 대한 AIR 응용 프로그램의 제작자 ID로 지정된 값을 지정합니다. 임시 제작자 ID를 지정함으로써 로컬 연결을 통한 통신과 같이 제작자 ID를 사용하여 응용 프로그램을 고유하게 식별하는 AIR 응용 프로그램의 기능을 테스트할 수 있습니다. AIR 1.5.3 현재, 응용 프로그램 설명자 파일에서 제작자 ID를 지정할 수도 있습니다(이 매개 변수를 사용해서는 안 됨).

참고: AIR 1.5.3 현재, 제작자 ID가 더 이상 자동으로 계산되어 AIR 응용 프로그램에 할당되지 않습니다. 기존 AIR 응용 프로그램에 대한 업데이트를 만들 때 제작자 ID를 지정할 수 있지만, 새 응용 프로그램에서는 제작자 ID를 지정할 필요가 없으며 지정해서도 안 됩니다.

-nodebug 디버깅 지원 기능을 해제합니다. 이를 사용할 경우 응용 프로그램 프로세스는 Flash 디버거에 연결되지 못하고 처리되지 않은 예외에 대한 대화 상자가 표시되지 않습니다. 그러나 trace 문은 계속 콘솔 윈도우에 인쇄됩니다. 디버깅 기능을 해제하면 응용 프로그램을 좀 더 빠르게 실행할 수 있으며 설치된 응용 프로그램의 실행 모드를 보다 근접하게 에뮬레이션할 수 있습니다.

-atlogin 로그인 시 응용 프로그램 실행을 시뮬레이션합니다. 이 플래그를 통해 사용자가 로그인하면 응용 프로그램이 실행되도록 설정된 경우에만 실행되는 응용 프로그램 논리를 테스트할 수 있습니다. -atlogin이 사용되는 경우 응용 프로그램에 전달되는 InvokeEvent 객체의 reason 속성은 standard가 아닌 login입니다(응용 프로그램이 아직 실행되지 않고 있는 경우).

-profile profileName ADL은 지정된 프로파일을 사용하여 응용 프로그램을 디버깅합니다. profileName은 desktop, extendedDesktop, mobileDevice 및 extendedMobileDevice가 될 수 있습니다. 자세한 내용은 67페이지의 “대상 응용 프로그램 프로파일 제한” 및 72페이지의 “응용 프로그램 프로파일”을 참조하십시오.

application.xml 응용 프로그램 설명자 파일입니다. 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오. 응용 프로그램 설명자는 ADL의 유일한 필수 매개 변수이고, 대부분의 경우 이 매개 변수 외의 다른 매개 변수는 필요하지 않습니다.

root-directory 실행할 응용 프로그램의 루트 디렉토리를 지정합니다. 지정하지 않으면 응용 프로그램 설명자 파일이 포함된 디렉토리가 사용됩니다.

-- arguments "--" 다음에 추가되는 모든 문자열은 명령줄 인수로 응용 프로그램에 전달됩니다.

참고: 이미 실행 중인 AIR 응용 프로그램을 시작할 때 해당 응용 프로그램의 새 인스턴스가 시작되지 않습니다. 그 대신 invoke 이벤트가 실행 중인 인스턴스로 전달됩니다.

trace 문 인쇄

trace 문을 ADL 실행에 사용되는 콘솔로 인쇄하려면 trace() 함수를 사용하여 코드에 trace 문을 추가합니다.

ActionScript 예제:

```
//ActionScript
trace("debug message");
```

JavaScript 예제:

```
//JavaScript
air.trace("debug message");
```

JavaScript에서는 alert() 및 confirm() 함수를 사용하여 응용 프로그램에서 보내는 디버깅 메시지를 표시할 수 있습니다. 또한 catch되지 않은 JavaScript 예외와 구문 오류의 줄 번호가 콘솔로 인쇄됩니다.

참고: JavaScript 예제에 나오는 air 접두어를 사용하려면 AIRAliases.js 파일을 페이지로 가져와야 합니다. 이 파일은 AIR SDK의 frameworks 디렉토리에 있습니다.

ADL 예제

현재 디렉토리에서 응용 프로그램 실행:

```
adl myApp-app.xml
```

현재 디렉토리의 하위 디렉토리에서 응용 프로그램 실행:

```
adl source/myApp-app.xml release
```

응용 프로그램을 실행하고 두 개의 명령줄 인수인 "tick"과 "tock"을 전달:

```
adl myApp-app.xml -- tick tock
```

특정 런타임을 사용하여 응용 프로그램 실행:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

디버깅 지원 없이 응용 프로그램 실행:

```
adl myApp-app.xml -nodebug
```

응용 프로그램을 실행하는 Apache Ant를 사용하여 응용 프로그램 실행:

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADL" value="${SDK_HOME}/bin/adl.exe"/>
<property name="APP_DESCRIPTOR" value="$src/myApp-app.xml"/>

<target name="test">
    <exec executable="${ADL}">
        <arg value="${APP_DESCRIPTOR}"/>
    </exec>
</target>
```

Flash Debugger(FDB)에 연결

Flash Debugger를 사용하여 AIR 응용 프로그램을 디버깅하려면 FDB 세션을 시작한 다음 ADL을 사용하여 응용 프로그램을 시작합니다.

참고: SWF 기반 AIR 응용 프로그램에서는 ActionScript 소스 파일을 -debug 플래그로 컴파일해야 합니다. Flash Professional에서는 [제작 설정] 대화 상자에서 [디버깅 허용] 옵션을 선택합니다.

1. FDB를 시작합니다. FDB 프로그램은 Flex SDK의 bin 디렉토리에서 찾을 수 있습니다.

   콘솔은 FDB 프롬프트인 <fdb>를 표시합니다.

2. run 명령 <fdb>run [Enter]를 실행합니다.
3. 다른 명령 또는 셸 콘솔에서 해당 응용 프로그램의 디버그 버전을 시작합니다.

   ```
   adl myApp.xml
   ```

4. FDB 명령을 사용하여 원하는 중단점을 설정합니다.
5. continue [Enter]를 입력합니다.

AIR 응용 프로그램이 SWF 기반인 경우 디버거는 ActionScript 코드의 실행만 제어합니다. AIR 응용 프로그램이 HTML 기반인 경우에는 디버거가 JavaScript 코드의 실행만 제어합니다.

디버거에 연결하지 않고 ADL을 실행하려면 -nodebug 옵션을 포함합니다.

```
adl myApp.xml -nodebug
```

FDB 명령에 대한 기본 정보를 보려면 help 명령을 실행합니다.

```
<fdb>help [Enter]
```

FDB 명령에 대한 자세한 내용은 Flex 설명서의 명령줄 디버거 명령 사용을 참조하십시오.

ADL 종료 및 오류 코드

다음 표에서는 ADL이 인쇄하는 종료 코드에 대해 설명합니다.

종료 코드	설명
0	성공적으로 시작했습니다. AIR 응용 프로그램 종료 후 ADL이 종료됩니다.
1	이미 실행 중인 AIR 응용 프로그램을 성공적으로 호출하였습니다. ADL이 즉시 종료됩니다.
2	사용 오류입니다. ADL에 잘못된 인수가 제공되었습니다.
3	런타임을 찾을 수 없습니다.
4	런타임을 시작할 수 없습니다. 이는 종종 응용 프로그램에 지정된 버전이 런타임의 버전과 일치하지 않기 때문에 발생합니다.
5	원인을 알 수 없는 오류가 발생했습니다.

종료 코드	설명
6	응용 프로그램 설명자 파일을 찾을 수 없습니다.
7	응용 프로그램 설명자의 내용이 유효하지 않습니다. 이 오류는 일반적으로 xml 형식이 올바르지 않음을 나타냅니다.
8	응용 프로그램 설명자 파일의 <content> 요소에 지정된 기본 응용 프로그램 내용 파일을 찾을 수 없습니다.
9	기본 응용 프로그램 내용 파일이 올바른 SWF 또는 HTML 파일이 아닙니다.
10	응용 프로그램에서 -profile 옵션으로 지정된 프로필을 지원하지 않습니다.

ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지

ADT(AIR Developer Tool)를 사용하여 SWF 기반 및 HTML 기반 AIR 응용 프로그램의 AIR 설치 파일을 만듭니다.

ADT는 명령줄 또는 Ant와 같은 빌드 도구에서 실행할 수 있는 Java 프로그램 입니다. SDK에는 Java 프로그램을 실행하는 명령줄 스크립트가 포함됩니다.

Flash Builder를 사용하여 응용 프로그램을 만들 경우 Flash Builder 내보내기 마법사를 사용하여 AIR 파일 패키지를 만들 수도 있습니다. Developing AIR applications with Flash Builder를 참조하십시오.

Flash Professional CS5를 사용할 경우 [파일] > [제작] 명령을 통해 AIR 패키지를 만들 수 있습니다. 자세한 내용은 Flash 사용(CS5)의 Adobe AIR용으로 제작을 참조하십시오. Adobe Flash CS3 또는 CS4 Professional을 사용할 경우 [명령] > [AIR 파일 만들기] 명령을 통해 AIR 패키지를 만들 수 있습니다. 자세한 내용은 Flash 사용(CS4)의 Adobe AIR용으로 제작을 참조하십시오.

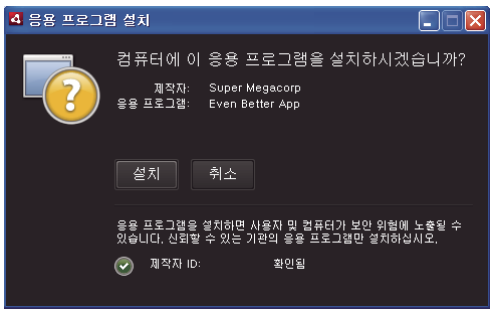
Adobe® AIR® Extension for Dreamweaver®를 사용하여 응용 프로그램을 만들 경우 [AIR 파일 만들기] 명령을 사용하여 AIR 패키지를 만들 수 있습니다. 이 명령은 [AIR 응용 프로그램 및 설치 프로그램 설정] 대화 상자에서 사용할 수 있습니다.

AIR 설치 파일 패키지

모든 AIR 응용 프로그램은 최소한 응용 프로그램 설명자 파일 및 기본 SWF 또는 HTML 파일이 있어야 합니다. 응용 프로그램에 설치할 다른 에셋들도 AIR 파일로 패키지해야 합니다.

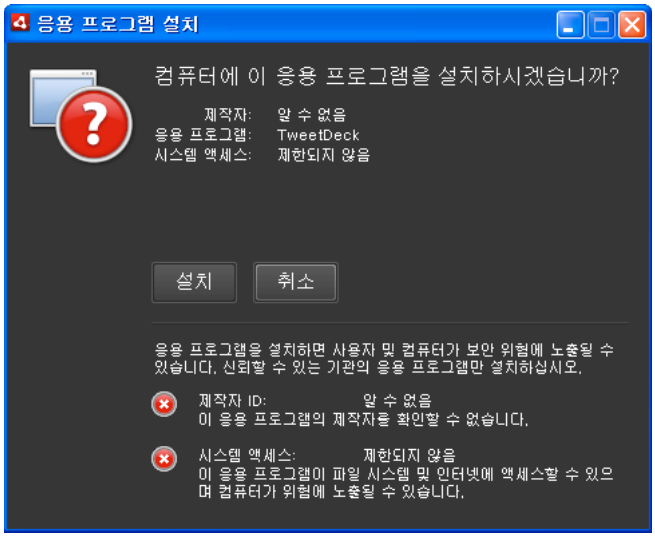
모든 AIR 설치 프로그램 파일은 디지털 인증서를 사용하여 서명해야 합니다. AIR 설치 프로그램은 서명을 사용하여 사용자가 응용 프로그램 파일에 서명한 이후 해당 파일이 변경되지 않았는지 확인합니다. 인증 기관의 코드 서명 인증서를 사용하거나 자체 서명된 인증서를 사용할 수 있습니다.

신뢰할 수 있는 인증 기관에서 발행된 인증서를 사용할 경우에는 응용 프로그램 사용자에게 제작자로서 자신의 ID에 대한 약간의 보증을 제공할 수 있습니다. 설치 대화 상자에는 자신의 ID가 인증 기관에 의해 확인되었다는 메시지가 표시됩니다.



신뢰할 수 있는 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

자체 서명된 인증서를 사용할 경우에는 사용자가 자신의 ID를 서명자로서 확인할 수 없습니다. 자체 서명된 인증서는 또한 패키지가 수정되지 않았음에 대한 보증도 약합니다. 사용자가 응용 프로그램을 받아 보기 전에 올바른 설치 파일이 위조된 파일로 대체될 수 있기 때문입니다. 설치 대화 상자에는 제작자의 ID를 확인할 수 없다는 메시지가 표시됩니다. 사용자는 응용 프로그램을 설치할 때 큰 보안 위험을 감수합니다.



자체 서명 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

ADT -package 명령을 사용하여 한 단계로 AIR 파일을 패키지 및 서명할 수 있습니다. -prepare 명령을 사용하여 서명되지 않은 중간 패키지를 만들고 별도의 단계에서 -sign 명령을 사용하여 중간 패키지에 서명할 수 있습니다.

참고: Java 버전 1.5 이상의 경우 PKCS12 인증서 파일을 보호하는 데 사용되는 암호에 상위 ASCII 문자를 사용할 수 없습니다. 코드 서명 인증서 파일을 만들거나 내보낼 경우에는 암호로 일반 ASCII 문자만 사용하십시오.

설치 패키지에 서명할 때 ADT는 자동으로 타임스탬프 기관 서버에 연결하여 시간을 확인합니다. 타임스탬프 정보는 AIR 파일에 포함됩니다. 확인된 타임스탬프가 들어 있는 AIR 파일은 나중에 언제든지 설치할 수 있습니다. ADT가 타임스탬프 서버에 연결할 수 없을 경우 패키지가 취소됩니다. 타임스탬프 옵션은 재정의가 가능하지만 타임스탬프가 없으면 설치 파일 서명에 사용된 인증서가 만료된 후에 AIR 응용 프로그램을 더 이상 설치할 수 없습니다.

기존 AIR 응용 프로그램을 업데이트하는 패키지를 만드는 중인 경우 원래 응용 프로그램과 동일한 인증서로 패키지를 서명해야 합니다. 원본 인증서가 갱신되었거나, 만료된 후 180일이 지나지 않았거나, 새 인증서로 바꾸려는 경우 마이그레이션 서명을 적용할 수 있습니다. 마이그레이션 서명은 응용 프로그램 AIR 파일을 새 인증서 및 이전 인증서 모두로 서명합니다. 60페이지의 “AIR 파일에 서명하여 응용 프로그램 인증서 변경”에 설명된 대로 -migrate 명령을 사용하여 마이그레이션 서명을 적용하십시오.

중요: 원본 인증서가 만료된 후 마이그레이션 서명을 적용할 수 있는 유예 기간은 정확히 180일입니다. 마이그레이션 서명을 사용하지 않으면 기존 사용자가 새 버전을 설치하기 전에 기존 응용 프로그램을 제거해야 합니다. 유예 기간은 응용 프로그램 설명자 네임스페이스에서 AIR 버전 1.5.3 이상이 지정된 응용 프로그램에만 적용됩니다. 이전 AIR 런타임 버전을 대상으로 할 경우에는 유예 기간이 없습니다.

AIR 1.1 이전에는 마이그레이션 서명이 지원되지 않습니다. 마이그레이션 서명을 적용하려면 SDK 버전 1.1 이상으로 응용 프로그램을 패키지해야 합니다.

AIR 파일을 사용하여 배포된 응용 프로그램은 데스크톱 프로파일 응용 프로그램으로 알려져 있습니다. 응용 프로그램 설명자 파일이 데스크톱 프로파일을 지원하지 않을 경우 ADT를 사용하여 AIR 응용 프로그램에 대한 기본 설치 프로그램을 패키지화할 수 없습니다. 응용 프로그램 설명자 파일에서 supportedProfiles 요소를 사용하여 프로파일을 제한할 수 있습니다. 67페이지의 “대상 응용 프로그램 프로파일 제한”을 참조하십시오.

참고: 응용 프로그램 설명자 파일의 설정은 AIR 응용 프로그램의 ID 및 기본 설치 경로를 결정합니다. 62페이지의 "응용 프로그램 설명자 파일 구조"를 참조하십시오.

제작자 ID

AIR 1.5.3부터는 제작자 ID가 더 이상 사용되지 않습니다. 원래 AIR 1.5.3 이상 버전에서 제작된 새 응용 프로그램에는 제작자 ID가 필요하지 않으며 이를 지정하지 않아야 합니다.

이전 버전의 AIR를 사용하여 제작된 응용 프로그램을 업데이트할 경우에는 응용 프로그램 설명자 파일에 원래 제작자 ID를 지정해야 합니다. 그렇지 않으면 응용 프로그램의 설치된 버전과 업데이트 버전이 다른 응용 프로그램으로 취급됩니다. 다른 ID를 사용하거나 publisherID 태그를 생략할 경우 사용자가 새 버전을 설치하기 전에 이전 버전을 제거해야 합니다.

원래 제작자 ID를 확인하려면 원래 응용 프로그램이 설치된 META-INF/AIR 하위 디렉토리에서 publisherid 파일을 찾아보십시오. 이 파일 내에 들어 있는 문자열이 제작자 ID입니다. 제작자 ID를 수동으로 지정하려면 응용 프로그램 설명자가 응용 프로그램 설명자 파일의 네임스페이스 선언에서 AIR 1.5.3 런타임 이상을 지정해야 합니다.

AIR 1.5.3 이전에 제작되었거나 AIR 1.5.3 SDK로 제작되었지만 응용 프로그램 설명자 네임스페이스에서 이전 AIR 버전이 지정된 응용 프로그램의 경우 제작자 ID가 서명 인증서를 기준으로 계산됩니다. 이 ID는 응용 프로그램 ID와 함께 응용 프로그램의 ID를 식별하는 데 사용됩니다. 제작자 ID(있을 경우)는 다음과 같은 용도로 사용됩니다.

- AIR 파일이 설치할 새 응용 프로그램보다 최신 파일인지 확인
- 암호화된 로컬 저장소를 위한 암호화 키의 일부
- 응용 프로그램 저장소 디렉토리 경로의 일부
- 로컬 연결을 위한 연결 문자열의 일부
- AIR 인 브라우저(in-browser) API를 통해 응용 프로그램을 호출하는 데 사용되는 ID 문자열의 일부
- OSID의 일부(사용자 정의 설치/제거 프로그램을 만들 때 사용됨)

AIR 1.5.3 이전에는 신규 또는 갱신된 인증서를 사용하는 마이그레이션 서명으로 응용 프로그램 업데이트를 서명할 경우 응용 프로그램의 제작자 ID가 변경될 수 있었습니다. 제작자 ID가 변경되면 해당 ID를 사용하는 모든 AIR 기능의 비헤이비어도 변경됩니다. 예를 들어 기존 암호화된 로컬 저장소에 포함된 데이터에 더 이상 액세스할 수 없으며, 응용 프로그램에 대한 로컬 연결을 만드는 모든 Flash 또는 AIR 인스턴스는 연결 문자열에 새로운 ID를 사용해야 합니다.

AIR 1.5.3 또는 그 이상에서는 제작자 ID가 서명 인증서를 기반으로 하지 않으며 응용 프로그램 설명자에 publisherID 태그가 포함된 경우에만 지정됩니다. 업데이트 AIR 패키지에 대해 지정된 제작자 ID가 현재 제작자 ID와 일치하지 않을 경우 응용 프로그램을 업데이트할 수 없습니다.

한 단계로 AIR 파일 패키지 및 서명

❖ -package 명령을 다음과 같은 구문으로 사용합니다(단일 명령줄에서).

```
adt -package SIGNING_OPTIONS air_file app_xml [file_or_dir | -C dir file_or_dir | -e file dir ...] ...
```

SIGNING_OPTIONS 서명 옵션은 AIR 파일 서명에 사용되는 개인 키 및 인증서가 포함된 키 저장소를 식별합니다. ADT에서 생성된 자체 서명 인증서를 사용하여 AIR 응용 프로그램에 서명하려면 다음과 같은 옵션을 사용합니다.

```
-storetype pkcs12 -keystore certificate.p12
```

이 예제에서 **certificate.p12**는 키 저장소 파일의 이름입니다. 명령줄에 암호가 제공되지 않으므로 ADT가 암호를 입력하도록 요청합니다. 서명 옵션은 55페이지의 “ADT 명령줄 서명 옵션”에서 자세히 설명합니다.

air_file 만들어지는 AIR 파일의 이름입니다.

app_xml 응용 프로그램 설명자 파일에 대한 경로입니다. 경로는 현대 디렉토리에 대한 상대 경로로 지정하거나 절대 경로로 지정할 수 있습니다. 응용 프로그램 설명자 파일은 AIR 파일에서 “application.xml”로 이름이 변경됩니다.

file_or_dir AIR 파일에서 패키지화할 파일 및 디렉토리입니다. 파일 및 디렉토리는 개수에 관계없이 지정할 수 있으며 공백으로 구분합니다. 디렉토리를 지정할 경우 해당 디렉토리 내의 숨겨진 파일을 제외한 모든 파일 및 하위 디렉토리가 패키지에 추가됩니다. 또한 직접 지정하거나 와일드 카드 또는 디렉토리 확장을 통해 응용 프로그램 설명자 파일을 지정할 경우 두 번째는 무시되고 패키지에 추가되지 않습니다. 지정된 파일 및 디렉토리는 현재 디렉토리에 있거나 하위 디렉토리 중 하나에 있어야 합니다. 현재 디렉토리를 변경하려면 -C 옵션을 사용합니다.

중요: –C 옵션 다음에 나오는 file_or_dir 인수에는 와일드 카드를 사용할 수 없습니다. 인수를 ADT에 전달하기 전에 명령 셸이 와일드 카드를 확장하므로, 이로 인해 ADT가 잘못된 위치에서 파일을 검색할 수 있기 때문입니다. 그러나 현재 디렉토리를 나타내는 도트 문자(".")는 계속 사용할 수 있습니다. 예를 들면 "-C assets ."은 하위 디렉토리를 포함하여 에셋 디렉토리의 모든 파일을 응용 프로그램 패키지의 루트 레벨로 복사합니다.

-C dir 응용 프로그램 패키지에 추가된 후속 파일 및 디렉토리를 처리하기 전에 작업 디렉토리를 **dir** 값으로 변경합니다. 파일 또는 디렉토리가 응용 프로그램 패키지의 루트에 추가됩니다. –C 옵션을 여러 번 사용하여 파일 시스템의 여러 지점에 있는 파일을 포함시킬 수 있습니다. dir에 상대 경로를 지정할 경우 경로는 항상 원래 작업 디렉토리에 대한 상대 경로로 해석됩니다.

ADT는 패키지에 포함된 파일 및 디렉토리를 처리하므로 현재 디렉토리와 대상 파일 간의 상대 경로가 저장됩니다. 이 경로는 패키지가 설치될 때 응용 프로그램 디렉토리 구조로 확장됩니다. 따라서 -C release/bin lib/feature.swf를 지정하면 루트 응용 프로그램 폴더의 lib 하위 디렉토리에 release/bin/lib/feature.swf 파일이 저장됩니다.

-e file dir 지정된 파일을 지정된 패키지 디렉토리에 저장합니다.

참고: 응용 프로그램 설명자 파일의 <content> 요소에서는 응용 프로그램 패키지 디렉토리 트리 내에서 기본 응용 프로그램 파일의 최종 위치를 지정해야 합니다.

또한 기본 설치 프로그램(Windows의 경우 EXE 파일, Mac OS의 경우 DMG 파일)을 사용하여 AIR 응용 프로그램을 패키지화하고 배포할 수도 있습니다. 자세한 내용은 57페이지의 “기본 설치 프로그램의 AIR 응용 프로그램 패키지”를 참조하십시오.

ADT -package 명령 예제

SWF 기반 AIR 응용 프로그램에 대해 현재 디렉토리에 있는 특정 응용 프로그램 파일을 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

HTML 기반 AIR 응용 프로그램에 대해 현재 디렉토리에 있는 특정 응용 프로그램 파일을 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js image.gif
```

현재 작업 디렉토리의 모든 파일 및 하위 디렉토리 패키지:

```
adt –package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

참고: 키 저장소 파일에는 응용 프로그램 서명에 사용되는 개인 키가 들어 있습니다. AIR 패키지 안에 서명 인증서를 포함시키지 마십시오. ADT 명령에 와일드카드를 사용할 경우 키 저장소 파일을 다른 위치에 보관하여 패키지에 포함되지 않도록 하십시오. 이 예제에서 키 저장소 파일 cert.p12는 상위 디렉토리에 있습니다.

기본 파일 및 이미지 하위 디렉토리만 패키지:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

HTML 기반 응용 프로그램 및 HTML, 스크립트 및 이미지 하위 디렉토리의 모든 파일을 패키지:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js html scripts
images
```

작업 디렉토리(release/bin)에 있는 application.xml 파일 및 기본 SWF 패키지:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml -C release/bin myApp.swf
```

빌드 파일 시스템의 두 곳 이상에서 에셋을 패키지화합니다. 이 예제에서 응용 프로그램 에셋은 패키지되기 전에 다음과 같은 폴더에 들어 있습니다.

```
/devRoot
    /myApp
        /release
            /bin
                myApp.xml
                myApp.swf or myApp.html
    /artwork
        /myApp
            /images
                image-1.png
                ...
                image-n.png
    /libraries
        /release
            /libs
                lib-1.swf
                lib-2.swf
                lib-a.js
                AIRAliases.js
```

/devRoot/myApp 디렉토리에서 다음과 같은 ADT 명령 실행:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml
    -C release/bin myApp.swf (or myApp.html)
    -C ../artwork/myApp images
    -C ../libraries/release libs
```

다음 패키지 구조 생성:

```
/myAppRoot
    /META-INF
        /AIR
            application.xml
            hash
    myApp.swf
    mimetype
    /images
        image-1.png
        ...
        image-n.png
    /libs
        lib-1.swf
        lib-2.swf
        lib-a.js
        AIRAliases.js
```

단순 SWF 기반 응용 프로그램에 대해 Java 프로그램으로 ADT 실행(클래스 경로 설정 안 함):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf
```

단순 HTML 기반 응용 프로그램에 대해 Java 프로그램으로 ADT 실행(클래스 경로 설정 안 함):

```
java -jar {AIRSDK}/lib/ADT.jar -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html
AIRAliases.js
```

ADT.jar 패키지를 포함하도록 Java 클래스 경로를 설정하여 Java 프로그램으로 ADT 실행:

```
java com.adobe.air.ADT -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf
```

Apache Ant에서 Java 작업으로 ADT 실행:

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADT.JAR" value="${SDK_HOME}/lib/adt.jar"/>

target name="package">
    <java jar="${ADT.JAR}" fork="true" failonerror="true">
        <arg value="-package"/>
        <arg value="-storetype"/>
        <arg value="pkcs12"/>
        <arg value="-keystore"/>
        <arg value="../../ExampleCert.p12"/>
        <arg value="myApp.air"/>
        <arg value="myApp.xml"/>
        <arg value="myApp.xml"/>
        <arg value="icons/*.png"/>
    </java>
</target>
```

ADT 오류 메시지

다음 표에서는 ADT 프로그램에서 보고할 수 있는 오류 및 가능한 원인의 목록을 보여 줍니다.

응용 프로그램 설명자 유효성 검사 오류

오류 코드	설명	참고 사항
100	응용 프로그램 설명자를 파싱할 수 없습니다.	응용 프로그램 설명자 파일에 닫히지 않은 태그와 같은 XML 구문 오류가 있는지 확인합니다.
101	네임스페이스가 없습니다.	누락된 네임스페이스를 추가합니다.
102	네임스페이스가 잘못되었습니다.	네임스페이스의 맞춤법을 확인합니다.
103	예기치 못한 요소 또는 특성입니다.	잘못된 요소 및 특성을 제거합니다. 사용자 정의 값은 설명자 파일에 사용할 수 없습니다. 요소 및 특성 이름의 맞춤법을 확인합니다. 요소가 올바른 부모 요소 내에 배치되었는지, 특성이 올바른 요소에 사용되었는지 확인합니다.
104	요소 또는 특성이 없습니다.	필요한 요소 또는 특성을 추가합니다.
105	요소 또는 특성에 잘못된 값이 포함되어 있습니다.	잘못된 값을 수정합니다.
106	잘못된 윈도우 특성 조합입니다.	transparency = true 및 systemChrome = standard와 같은 일부 윈도우 설정은 함께 사용할 수 없습니다. 호환되지 않는 설정 중 하나를 변경합니다.
107	윈도우 최소 크기가 윈도우 최대 크기보다 큽니다.	최소 또는 최대 크기 설정을 변경합니다.

네임스페이스, 요소, 특성 및 유효한 해당 값에 대한 자세한 내용은 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오.

응용 프로그램 아이콘 오류

오류 코드	설명	참고 사항
200	아이콘 파일을 열 수 없습니다.	파일이 지정된 경로에 있는지 확인합니다. 다른 응용 프로그램을 사용하여 파일을 열 수 있는지 확인합니다.
201	아이콘의 크기가 잘못되었습니다.	아이콘 크기(픽셀)는 XML 태그와 일치해야 합니다. 예를 들어 다음 응용 프로그램 설명자 요소가 있을 경우 <image32x32>icon.png</image32x32> icon.png의 이미지 크기는 정확히 32x32픽셀이어야 합니다.
202	아이콘 파일에 지원되지 않는 이미지 형식이 포함되어 있습니다.	PNG 형식만 지원됩니다. 응용 프로그램을 패키지화하기 전에 이미지를 다른 형식으로 변환합니다.

응용 프로그램 파일 오류

오류 코드	설명	참고 사항
300	파일이 없거나 파일을 열 수 없습니다.	명령줄에 지정된 파일을 찾을 수 없거나 열 수 없습니다.
301	응용 프로그램 설명자 파일이 없거나 파일을 열 수 없습니다.	응용 프로그램 설명자 파일이 지정된 경로에 없거나 파일을 열 수 없습니다.
302	루트 내용 파일이 패키지에 없습니다.	응용 프로그램 설명자의 <content> 요소에서 참조되는 SWF 또는 HTML 파일을 ADT 명령줄에 나열된 파일에 포함하여 패키지에 추가해야 합니다.
303	패키지에 아이콘 파일이 없습니다.	응용 프로그램 설명자에 지정된 아이콘 파일을 ADT 명령줄에 나열된 파일에 포함하여 패키지에 추가해야 합니다. 아이콘 파일은 자동으로 추가되지 않습니다.
304	초기 윈도우 내용이 잘못되었습니다.	응용 프로그램 설명자의 <content> 요소에서 참조되는 파일은 올바른 HTML 또는 SWF 파일로 인식되지 않습니다.
305	초기 윈도우 내용 SWF 버전이 네임스페이스 버전을 초과합니다.	응용 프로그램 설명자의 <content> 요소에서 참조되는 파일의 SWF 버전은 설명자 네임스페이스에 지정된 버전의 AIR에서 지원되지 않습니다. 예를 들어 SWF10(Flash Player 10) 파일을 AIR 1.1 응용 프로그램의 초기 내용으로 패키지화하려고 하면 이 오류가 발생합니다.
306	프로파일이 지원되지 않습니다.	응용 프로그램 설명자 파일에서 지정한 프로파일이 지원되지 않습니다. 67페이지의 "대상 응용 프로그램 프로파일 제한"을 참조하십시오.
	네임스페이스가 기능을 지원하지 않습니다.	해당 기능에 적절한 네임스페이스(예: 2.0 네임스페이스)를 사용하십시오.

기타 오류에 대한 종료 코드

종료 코드	설명	참고 사항
2	사용 오류입니다.	명령줄 인수에 오류가 없는지 확인합니다.
5	알 수 없는 오류입니다.	이 오류는 일반적인 오류 상황으로 설명할 수 없는 상황을 나타냅니다. ADT와 Java 런타임 환경이 호환되지 않거나, ADT 또는 JRE 설치가 손상되었거나, ADT 내에 프로그래밍 오류가 발생하는 등의 근본적인 원인입니다.
6	출력 디렉토리에 쓸 수 없습니다.	지정되거나 암시된 출력 디렉토리에 액세스할 수 있는지, 포함된 드라이브의 디스크 공간이 충분한지 확인합니다.
7	인증서에 액세스할 수 없습니다.	키 저장소에 대한 경로가 올바르게 지정되었는지 확인합니다. 키 저장소 내의 인증서에 액세스할 수 있는지 확인합니다. Java 1.6 Keytool 유틸리티를 사용하면 인증서 액세스 문제를 해결하는 데 도움이 됩니다.
8	인증서가 잘못되었습니다.	인증서 파일의 형식이 잘못되었거나 인증서 파일이 수정, 만료 또는 해지되었습니다.
9	AIR 파일을 서명할 수 없습니다.	ADT에 전달된 서명 옵션을 확인합니다.
10	타임스탬프를 만들 수 없습니다.	ADT에서 타임스탬프 서버에 대한 연결을 설정할 수 없습니다. 프록시 서버를 통해 인터넷에 연결하는 경우 JRE 프록시 설정을 구성해야 할 수 있습니다.
11	인증서 만들기 오류가 발생했습니다.	서명을 만드는 데 사용된 명령줄 인수를 확인합니다.
12	입력이 잘못되었습니다.	명령줄에서 ADT에 전달된 파일 경로 및 기타 인수를 확인합니다.

ADT 명령줄 서명 옵션

ADT는 JCA(Java Cryptography Architecture)를 사용하여 AIR 응용 프로그램에 서명할 개인 키 및 인증서에 액세스합니다. 서명 옵션은 키 저장소 및 키 저장소 내의 개인 키와 인증서를 식별합니다.

키 저장소에는 개인 키 및 연관된 인증서 체인이 모두 들어 있어야 합니다. 서명 인증서가 사용자 컴퓨터에 있는 신뢰할 수 있는 인증서에 체인으로 연결되는 경우 AIR 설치 대화 상자에 인증서의 공용 이름이 제작자 이름으로 표시됩니다.

ADT에서는 x509v3 표준(RFC3280)을 따르는 인증서를 필요로 하며 적절한 코드 서명 값을 사용하여 확장 키 사용 확장을 포함시킵니다. 인증서 내의 제약 조건을 준수하고 AIR 응용 프로그램에 서명할 일부 인증서 사용을 제외시킬 수 있습니다.

참고: ADT는 필요한 경우 인증서 해지 목록을 확인하고 타임스탬프를 가져오기 위해 Java 런타임 환경 프록시 설정을 사용하여 인터넷 리소스에 연결합니다. ADT 사용 시 인터넷 리소스 연결에 문제가 발생하여 네트워크에 특정 프록시 설정이 필요한 경우 JRE 프록시 설정을 구성해야 합니다.

AIR 서명 옵션 지정

❖ -package 및 -prepare 명령에 ADT 서명 옵션을 지정하려면 다음 구문을 사용합니다.

```
[-alias aliasName] [-storetype type] [-keystore path] [-storepass password1] [-keypass password2] [-
providerName className] [-tsa url]
```

-alias aliasName 키 저장소에 있는 키의 별칭입니다. 키 저장소에 단 하나의 인증서만 들어 있을 때는 별칭을 지정할 필요가 없습니다. 별칭을 지정하지 않을 경우 ADT는 키 저장소의 첫 번째 키를 사용합니다.

모든 키 저장소 관리 응용 프로그램이 인증서에 대한 별칭 지정을 허용하는 것은 아닙니다. 예를 들어 Windows 시스템 키 저장소를 사용할 경우 인증서의 고유 이름을 별칭으로 사용합니다. 별칭을 확인할 수 있도록 Java Keytool 유틸리티를 사용하여 사용 가능한 인증서를 나열할 수 있습니다. 예를 들어 다음과 같은 명령을 실행합니다.

```
keytool -list -storetype Windows-MY
```

그러면 인증서에 대해 다음과 같은 출력이 생성됩니다.

```
CN=TestingCert,OU=QE,O=Adobe,C=US, PrivateKeyEntry,
Certificate fingerprint (MD5): 73:D5:21:E9:8A:28:0A:AB:FD:1D:11:EA:BB:A7:55:88
```

ADT 명령줄에서 이 인증서를 참조하려면 별칭을 다음과 같이 설정합니다.

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

MaC Os X에서 키체인의 인증서 별칭은 키체인 접근 응용 프로그램에 표시된 이름입니다.

-storetype type 키 저장소의 유형으로 키 저장소 구현 시 결정됩니다. 대부분의 Java 설치 시 포함되는 기본 키 저장소 구현은 JKS 및 PKCS12 유형이 지원됩니다. Java 5.0에는 하드웨어 토큰에서 키 저장소에 액세스하기 위한 PKCS11 유형 및 MaC Os X 키체인에 액세스하기 위한 Keychain 유형에 대한 지원이 포함됩니다. Java 6.0에는 MSCAPI 유형에 대한 지원이 포함됩니다(Windows에서). 다른 JCA 공급자가 설치 및 구성되어 있는 경우 추가적인 키 저장소 유형을 사용할 수도 있습니다. 키 저장소 유형을 지정하지 않을 경우 기본 JCA 공급자에 대한 기본 유형이 사용됩니다.

저장소 유형	키 저장소 형식	최소 Java 버전
JKS	Java 키 저장소 파일 (.keystore)	1.2
PKCS12	PKCS12 파일(.p12 또는 .pfx)	1.4
PKCS11	하드웨어 토큰	1.5
KeychainStore	MaC Os X 키체인	1.5
Windows-MY 또는 Windows-ROOT	MSCAPI	1.6

-keystore path 파일 기반 저장소 유형을 위한 키 저장소 파일의 경로입니다.

-storepass password1 키 저장소에 액세스하는 데 필요한 암호입니다. 지정하지 않는 경우 ADT가 암호 입력을 요구합니다.

-keypass password2 AIR 응용 프로그램 서명에 사용되는 개인 키에 액세스하는 데 필요한 암호입니다. 지정하지 않는 경우 ADT가 암호 입력을 요구합니다.

-providerName className 지정된 키 저장소 유형의 JCA 공급자입니다. 지정하지 않는 경우 ADT는 해당 유형의 키 저장소에 대한 기본 공급자를 사용합니다.

-tsa urlRFC3161 호환 타임스탬프 서버의 URL을 지정하여 디지털 서명에 타임스탬프를 포함시킵니다. URL을 지정하지 않으면 Geotrust가 제공한 기본 타임스탬프 서버가 사용됩니다. AIR 응용 프로그램의 서명에 타임스탬프가 포함되어 있을 경우 타임스탬프가 서명 시점에 인증서가 유효함을 확인한 것이므로 서명 인증서가 만료된 후에도 응용 프로그램을 설치할 수 있습니다.

ADT가 타임스탬프 서버에 연결되지 않으면 서명이 취소되고 패키지가 만들어지지 않습니다. 타임스탬프를 포함시키지 않으려면 -tsa none을 지정합니다. 그러나 타임스탬프 없이 패키지된 AIR 응용 프로그램은 서명 인증서가 만료된 후에는 더 이상 설치할 수 없습니다.

참고: 서명 옵션은 Java Keytool 유틸리티의 해당 옵션과 유사합니다. Windows에서는 Keytool 유틸리티를 사용하여 키 저장소를 검사 및 관리할 수 있습니다. MaC Os X의 경우 Apple® 보안 유틸리티를 사용할 수도 있습니다.

서명 옵션 예제

.p12 파일을 사용한 서명:

```
-storetype pkcs12 -keystore cert.p12
```

기본 Java 키 저장소를 사용한 서명:

```
-alias AIRcert -storetype jks
```

특정 Java 키 저장소를 사용한 서명:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

MaC Os X 키체인을 사용한 서명:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Windows 시스템 키 저장소를 사용한 서명:

```
-alias cn=AIRCert -storeype Windows-MY
```

하드웨어 토큰을 사용한 서명(토큰을 사용하도록 Java를 구성하는 방법 및 올바른 providerName 값은 토큰 제조업체의 지침 참조):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

타임스탬프를 포함하지 않는 서명:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

기본 설치 프로그램의 AIR 응용 프로그램 패키지

AIR 2에서는 ADT를 사용하여 AIR 응용 프로그램 배포에 대한 기본 응용 프로그램 설치 프로그램을 만들 수 있습니다. 예를 들어, Windows에서 AIR 응용 프로그램을 배포하기 위한 EXE 설치 프로그램 파일을 만들 수 있습니다. Mac OS에서는 AIR 응용 프로그램의 배포를 위한 DMG 설치 프로그램 파일을 만들 수 있습니다. Linux에서는 AIR 응용 프로그램의 배포를 위한 DEB 또는 RPM 설치 프로그램 파일을 만들 수 있습니다.

기본 응용 프로그램 설치 프로그램으로 설치된 응용 프로그램은 확장 데스크톱 프로파일 응용 프로그램이라고 합니다. 응용 프로그램 설명자 파일이 데스크톱 확장 프로파일을 지원하지 않을 경우 ADT를 사용하여 AIR 응용 프로그램에 대한 기본 설치 프로그램을 패키지화할 수 없습니다. 응용 프로그램 설명자 파일에서 supportedProfiles 요소를 사용하여 프로파일을 제한할 수 있습니다. 67페이지의 “대상 응용 프로그램 프로파일 제한”을 참조하십시오.

AIR 응용 프로그램의 기본 설치 프로그램 버전은 다음 두 가지 방식으로 만들 수 있습니다.

- 응용 프로그램 설명자 파일 및 기타 소스 파일을 기반으로 기본 설치 프로그램을 만들 수 있습니다. 기타 소스 파일에는 SWF 파일, HTML 파일 및 기타 에셋이 포함될 수 있습니다.
- AIR 파일 기반 또는 AIRI 파일 기반으로 기본 설치 프로그램을 만들 수 있습니다.

생성하려는 기본 설치 프로그램 파일과 동일한 운영 체제에서 ADT를 사용해야 합니다. 따라서 Windows용 EXE 파일을 만들려면 Windows에서 ADT를 실행하십시오. Mac OS용 DMG 파일을 만들려면 Mac OS에서 ADT를 실행하십시오. Linux용 DEB 또는 RPG 파일을 만들려면 Linux에서 ADT를 실행하십시오.

AIR 응용 프로그램을 배포하기 위해 기본 설치 프로그램을 만들 경우 이러한 응용 프로그램에는 다음과 같은 기능이 포함됩니다.

- NativeProcess 클래스를 이용하여 기본 프로세스를 실행하고 상호 작용할 수 있습니다. 자세한 내용은 다음 중 하나를 참조하십시오.
  - AIR의 기본 프로세스와 통신(ActionScript 개발자용)

- Communicating with native processes in AIR(HTML 개발자용)

- 이 클래스는 파일 유형에 관계없이, 기본 시스템 응용 프로그램에서 열리도록 정의된 모든 파일에 대해 File.openWithDefaultApplication() 메서드를 호출할 수 있습니다. 기본 설치 프로그램으로 설치되지 않은 응용 프로그램에서는 사용할 수 없습니다. 자세한 내용은 언어 참조 설명서에서 File.openWithDefaultApplication() 항목에 대한 항목을 참조하십시오.

사용자가 기본 설치 프로그램 파일을 두 번 클릭하면 AIR 응용 프로그램이 설치됩니다. 필요한 Adobe AIR 버전이 시스템에 아직 설치되지 않은 경우 설치 프로그램이 네트워크에서 Adobe AIR 버전을 다운로드하여 먼저 설치합니다. 올바른 Adobe AIR 버전(필요한 경우)을 가져올 수 있는 네트워크 연결이 없는 경우 설치가 실패합니다. 또한 운영 체제가 Adobe AIR 2에서 지원되지 않으면 설치가 실패합니다.

참고: 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.

응용 프로그램 소스 파일에서 기본 설치 프로그램 만들기

응용 프로그램에 대한 소스 파일에서 기본 설치 프로그램을 만들려면 단일 명령줄에서 다음 구문으로 -package 명령을 사용하십시오.

```
adt -package AIR_SIGNING_OPTIONS-target native [WINDOWS_INSTALLER_SIGNING_OPTIONS]
installer_fileapp_xml [file_or_dir | -C dir file_or_dir | -e file dir ...] ...
```

이 구문은 AIR 파일을 패키지화하는 구문과 비슷합니다(기본 설치 프로그램 사용 안 함) 하지만 다음과 같은 몇 가지 차이점이 있습니다.

- -target native 옵션을 명령에 추가합니다. -target air를 지정한 경우 ADT는 기본 설치 프로그램 파일 대신 AIR 파일을 생성합니다.
- 대상 DMG 또는 EXE 파일을 installer_file로 지정합니다.
- 선택적으로 Windows에서는 구문 목록에 [WINDOWS_INSTALLER_SIGNING_OPTIONS]로 표시되는 두 번째 서명 옵션 집합을 추가할 수 있습니다. Windows에서는 AIR 파일 서명 외에도 Windows 설치 프로그램 파일을 서명할 수 있습니다. AIR 파일 서명에서와 동일한 인증서 유형 및 서명 옵션 구문을 사용합니다(55페이지의 "ADT 명령줄 서명 옵션"). AIR 파일 및 설치 프로그램 파일을 서명하는 데 동일한 인증서를 사용하거나 다른 인증서를 지정할 수 있습니다. 사용자가 웹에서 서명된 Windows 설치 프로그램 파일을 다운로드한 경우 Windows는 인증서를 기반으로 파일 소스를 식별합니다.

-target 옵션 이외의 ADT 옵션에 대한 자세한 내용은 48페이지의 "AIR 설치 파일 패키지"를 참조하십시오.

다음 예에서는 DMG 파일(Mac OS의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native myApp.dmg application.xml index.html
resources
```

다음 예에서는 EXE 파일(Windows의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native myApp.exe application.xml index.html
resources
```

다음 예에서는 EXE 파일을 만들고 서명합니다.

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native -storetype pkcs12 -keystore myCert.pfx
myApp.exe application.xml index.html resources
```

AIR 파일 또는 AIRI 파일에서 기본 설치 프로그램 만들기

ADT를 사용하여 AIR 파일 또는 AIRI 파일을 기반으로 기본 설치 프로그램 파일을 생성할 수 있습니다. AIR 파일을 기반으로 기본 설치 프로그램을 만들려면 단일 명령줄에 다음 구문으로 ADT -package 명령을 사용하십시오.

```
adt -package -target native [WINDOWS_INSTALLER_SIGNING_OPTIONS] installer_file air_file
```

이 구문은 AIR 응용 프로그램에 대한 소스 파일을 기반으로 기본 설치 프로그램을 만들 때의 구문과 비슷합니다. 하지만 다음과 같은 몇 가지 차이점이 있습니다.

- 응용 프로그램 설명자 파일 및 AIR 응용 프로그램에 대한 기타 소스 파일 대신 AIR 파일을 소스로 지정합니다.
- AIR 파일은 이미 서명되어 있으므로 AIR 파일에 대해 서명 옵션을 지정하지 않습니다.

AIRI 파일을 기반으로 기본 설치 프로그램을 만들려면 단일 명령줄에 다음 구문으로 ADT -package 명령을 사용하십시오.

```
adt AIR_SIGNING_OPTIONS -package -target native [WINDOWS_INSTALLER_SIGNING_OPTIONS] installer_file
airi_file
```

이 구문은 AIR 파일을 기반으로 기본 설치 프로그램을 만들 때의 구문과 비슷합니다. 하지만 다음과 같은 몇 가지 차이점이 있습니다.

- AIRI 파일을 소스로 지정합니다.
- 대상 AIR 응용 프로그램에 대한 서명 옵션을 지정합니다.

다음 예에서는 AIR 파일을 기반으로 DMG 파일(Mac OS의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package -target native myApp.dmg myApp.air
```

다음 예에서는 AIR 파일을 기반으로 EXE 파일(Windows의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -package -target native myApp.exe myApp.air
```

다음 예에서는 EXE 파일(AIR 파일 기반)을 만들고 서명합니다.

```
adt -package -target native -storetype pkcs12 -keystore myCert.pfx myApp.exe myApp.air
```

다음 예에서는 AIRI 파일을 기반으로 DMG 파일(Mac OS의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -storetype pkcs12 -keystore myCert.pfx -packag e-target native myApp.dmg myApp.airi
```

다음 예에서는 AIRI 파일을 기반으로 EXE 파일(Windows의 기본 설치 프로그램 파일)을 만듭니다.

```
adt -storetype pkcs12 -keystore myCert.pfx -package -target native myApp.exe myApp.airi
```

다음 예에서는 EXE 파일(AIRI 파일 기반)을 만들고 서명합니다.

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native -storetype pkcs12 -keystore myCert.pfx
myApp.exe myApp.airi
```

ADT를 사용하여 서명되지 않은 AIR 중간 파일 만들기

-prepare 명령을 사용하여 서명되지 않은 AIR 중간 파일을 만듭니다. AIR 중간 파일은 ADT -sign 명령을 사용하여 서명되어야만 유효한 AIR 설치 파일이 만들어집니다.

-prepare 명령은 -package 명령과 동일한 플래그 및 매개변수를 사용합니다(서명 옵션은 예외). 유일한 차이점은 출력 파일이 서명되지 않는다는 점입니다. 생성되는 중간 파일의 파일 확장명은 airi입니다.

AIR 중간 파일에 서명하려면 ADT -sign 명령을 사용합니다. 59페이지의 “ADT를 사용하여 AIR 중간 파일 서명”을 참조하십시오.

ADT -prepare 명령 예

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

ADT를 사용하여 AIR 중간 파일 서명

ADT를 사용하여 AIR 중간 파일에 서명하려면 -sign 명령을 사용합니다. 서명 명령은 AIR 중간 파일(확장명은 airi)에 대해서만 실행됩니다. AIR 파일에는 두 번 서명할 수 없습니다.

AIR 중간 파일을 만들려면 ADT -prepare 명령을 사용합니다. 자세한 내용은 59페이지의 “ADT를 사용하여 서명되지 않은 AIR 중간 파일 만들기”를 참조하십시오.

AIRI 파일 서명

❖ ADT -sign 명령을 다음과 같은 구문으로 사용합니다.

```
adt -sign SIGNING_OPTIONSairi_fileair_file
```

SIGNING_OPTIONSs 서명 옵션은 AIR 파일 서명에 사용할 개인 키 및 인증서를 식별합니다. 이 옵션에 대한 자세한 내용은 55페이지의 “ADT 명령줄 서명 옵션”에서 설명합니다.

airi_file 서명할 서명되지 않은 AIR 중간 파일 경로입니다.

air_file 만들 AIR 파일의 이름입니다.

ADT -sign 명령 예

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

자세한 내용은 84페이지의 “AIR 파일에 디지털 서명”을 참조하십시오.

AIR 파일에 서명하여 응용 프로그램 인증서 변경

새 서명 인증서나 갱신된 서명 인증서를 사용하는 동안 기존 AIR 응용 프로그램에 대한 업데이트를 제작하려면 ADT -migrate 명령을 사용하여 인증서 마이그레이션 서명을 적용합니다. 마이그레이션 서명은 원본 인증서를 사용하여 AIR 파일에 적용되는 2차 서명으로, 응용 프로그램 업데이트가 원본 인증서의 소유자에 의해 제작되었는지 유효성을 검사합니다.

마이그레이션 서명을 적용하려면 원본 인증서가 아직 유효하거나, 만료된 후 180일이 지나지 않은 상태여야 합니다. 인증서가 만료되고 180일의 유예 기간이 경과한 후에는 마이그레이션 서명을 적용할 수 없습니다. 응용 프로그램 사용자는 기존 버전을 제거한 후에 업데이트 버전을 설치할 수 있습니다. 기본적으로 마이그레이션 서명에는 타임스탬프가 포함되어 있으므로 마이그레이션 서명으로 서명된 AIR 업데이트는 인증서가 만료된 후에도 유효합니다.

참고: 180일의 유예 기간은 응용 프로그램 설명자 네임스페이스에서 AIR 버전 1.5.3 이상을 지정하는 응용 프로그램에만 적용됩니다.

새 인증서나 갱신된 인증서를 사용하기 위해 응용 프로그램을 마이그레이션하려면

1. 응용 프로그램의 업데이트를 만듭니다.
2. **새** 인증서를 사용하여 업데이트 AIR 파일을 패키지화하고 서명합니다.
3. -migrate 명령을 사용하여 **원본** 인증서로 AIR 파일을 다시 서명합니다.

-migrate 명령으로 서명된 AIR 파일을 사용하여 새 버전의 응용 프로그램을 설치하고 이전 인증서로 서명된 버전을 비롯한 모든 이전 버전을 업데이트할 수 있습니다.

참고: 1.5.3 이전 버전의 AIR용으로 제작된 응용 프로그램을 업데이트하는 경우 응용 프로그램 설명자에서 원래 제작자 ID를 지정해야 합니다. 그렇지 않으면 응용 프로그램 사용자가 이전 버전을 제거한 후 업데이트를 설치해야 합니다.

새 인증서 사용을 위한 AIR 응용 프로그램 마이그레이션

❖ ADT -migrate 명령을 다음과 같은 구문으로 사용합니다.

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

SIGNING_OPTIONSs 서명 옵션은 AIR 파일 서명에 사용할 개인 키 및 인증서를 식별합니다. 이 옵션으로 **원본** 서명 인증서를 식별해야 합니다. 이러한 옵션에 대한 자세한 내용은 55페이지의 “ADT 명령줄 서명 옵션”에서 설명합니다.

air_file_in 새 인증서를 사용하여 서명한 업데이트의 AIR 파일입니다.

air_file_out 만들 AIR 파일입니다.

ADT 예제

```
adt -migrate -storetype pkcs12 -keystore cert.p12 myApp.air myApp.air
```

자세한 내용은 84페이지의 “AIR 파일에 디지털 서명”을 참조하십시오.

참고: -migrate 명령은 AIR 1.1 릴리스의 ADT에 추가되었습니다.

ADT를 사용하여 자체 서명된 인증서 만들기

자체 서명된 인증서를 사용하여 유효한 AIR 설치 파일을 만들 수 있습니다. 하지만 자체 서명된 인증서는 사용자의 보안을 제한적으로만 보장하고, 자체 서명된 인증서의 신뢰성은 확인할 수 없습니다. 자체 서명된 AIR 파일을 설치하면 사용자에게 제작자 정보가 [알 수 없음]으로 표시됩니다. ADT에 의해 생성되는 인증서는 5년 동안 유효합니다.

자체 서명된 인증서로 서명한 AIR 응용 프로그램의 업데이트를 만들 경우 동일한 인증서를 사용하여 원본 및 업데이트 AIR 파일을 모두 서명해야 합니다. ADT에서 생성된 인증서는 동일한 매개변수를 사용한 경우에도 항상 고유합니다. 따라서 ADT에서 생성된 인증서로 업데이트를 자체 서명하려는 경우 원본 인증서를 안전한 위치에 보관하십시오. 또한 ADT에서 생성된 원본 인증서가 만료되면 업데이트된 AIR 파일을 만들 수 없습니다. 다른 인증서로 새 응용 프로그램을 제작할 수 있으나 동일한 응용 프로그램의 새 버전은 제작할 수 없습니다.

중요: 자체 서명된 인증서의 제한으로 인해 공개적으로 출시되는 AIR 응용 프로그램에 서명할 경우 신뢰할 수 있는 인증 기관의 상업용 인증서를 사용하는 것이 좋습니다.

ADT에서 생성된 인증서 및 연관된 개인 키는 PKCS12 유형의 키 저장소 파일에 저장됩니다. 지정된 암호는 키 저장소가 아닌 키 자체에 설정됩니다.

자체 서명 AIR 파일의 디지털 ID 인증서 생성

❖ ADT -certificate 명령 사용(단일 명령줄에서):

```
adt -certificate -cn name [-ou org_unit][-o org_name][-c country] key_type pfx_file password
```

-cn name 새 인증서의 공용 이름으로 지정된 문자열입니다.

-ou org_unit 인증서를 발급한 기관 구성 단위로 지정된 문자열입니다. 이는 선택 사항입니다.

-o org_name 인증서를 발급한 기관으로 지정된 문자열입니다. 이는 선택 사항입니다.

-c country 2자로 구성된 ISO-3166 국가 코드입니다. 코드가 유효하지 않을 경우 인증서가 생성되지 않습니다. 이는 선택 사항입니다.

key_type 인증서에 사용할 키 유형으로 "1024-RSA" 또는 "2048-RSA"입니다.

pfx_file 생성할 인증서 파일의 경로입니다.

password 새 인증서에 액세스하는 데 사용할 암호입니다. 이 인증서로 AIR 파일에 서명할 때 암호가 필요합니다.

인증서 생성 예제

```
adt -certificate -cn SelfSign -ou QE -o "Example, Co" -c US 2048-RSA newcert.p12 39#wnetx3tl
adt -certificate -cn ADigitalID 1024-RSA SigningCert.p12 39#wnetx3tl
```

이 인증서를 사용하여 AIR 파일에 서명하려면 ADT -package 또는 -prepare 명령과 함께 다음과 같은 서명 옵션을 사용합니다.

```
-storetype pkcs12 -keystore newcert.p12 -keypass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -keypass 39#wnetx3tl
```

참고: Java 버전 1.5 이상의 경우 PKCS12 인증서 파일을 보호하는 데 사용되는 암호에 상위 ASCII 문자를 사용할 수 없습니다. 암호에는 일반 ASCII 문자만 사용하십시오.

11장: AIR 응용 프로그램 속성 설정

AIR 응용 프로그램을 구성하는 모든 파일 및 기타 에셋 이외에도 각 AIR 응용 프로그램에는 응용 프로그램 설명자 파일이 필요합니다. 응용 프로그램 설명자 파일은 응용 프로그램의 기본 속성을 정의하는 XML 파일입니다.

AIR을 지원하는 대부분의 개발 환경은 프로젝트를 만들 때 자동으로 응용 프로그램 설명자를 생성합니다. 그렇지 않은 경우 직접 설명자 파일을 생성해야 합니다. 샘플 설명자 파일(descriptor-sample.xml)은 AIR 및 Flex SDK의 samples 디렉토리에 있습니다.

중요: Flash Professional CS5 [AIR 설정] 대화 상자가 열린 상태에서는 응용 프로그램 설명자 파일을 편집하지 마십시오. [iPhone 설정] 대화 상자를 열기 전에 응용 프로그램 설명자 파일에 변경 내용을 저장하십시오.

이 응용 프로그램 설명자 파일을 사용하여 ActionScript 기반 iPhone 응용 프로그램에 대한 설정을 정의할 수도 있습니다. 자세한 내용은 응용 프로그램 설명자 파일의 iPhone 응용 프로그램 속성 설정을 참조하십시오.

응용 프로그램 설명자 파일 구조

응용 프로그램 설명자 파일에는 이름, 버전, 저작권 등과 같이 전체 응용 프로그램에 영향을 주는 속성이 포함되어 있습니다. 응용 프로그램 설명자 파일에는 모든 파일 이름을 사용할 수 있습니다. 응용 프로그램을 패키지화하면 응용 프로그램 설명자 파일이 application.xml로 이름이 변경되고 AIR 패키지의 특수 디렉토리 내에 배치됩니다.

다음은 응용 프로그램 설명자 파일의 예제입니다.

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/2.0">
    <id>HelloWorld</id>
    <version>2.0</version>
    <filename>Hello World</filename>
    <name>Example Co. AIR Hello World</name>
     <description>
        <text xml:lang="en">This is a example.</text>
        <text xml:lang="fr">C'est un exemple.</text>
        <text xml:lang="es">Esto es un ejemplo.</text>
    </description>
    <copyright>Copyright (c) 2006 Example Co.</copyright>
    <initialWindow>
        <title>Hello World</title>
        <content>
            HelloWorld.swf
        </content>
        <systemChrome>none</systemChrome>
        <transparent>true</transparent>
        <visible>true</visible>
        <minimizable>true</minimizable>
        <maximizable>false</maximizable>
        <resizable>false</resizable>
        <width>640</width>
        <height>480</height>
        <minSize>320 240</minSize>
        <maxSize>1280 960</maxSize>
    </initialWindow>
    <installFolder>Example Co/Hello World</installFolder>
    <programMenuFolder>Example Co</programMenuFolder>
    <icon>
        <image16x16>icons/smallIcon.png</image16x16>
```

```
        <image32x32>icons/mediumIcon.png</image32x32>
        <image48x48>icons/bigIcon.png</image48x48>
        <image128x128>icons/biggestIcon.png</image128x128>
    </icon>
    <customUpdateUI>true</customUpdateUI>
    <allowBrowserInvocation>false</allowBrowserInvocation>
    <fileTypes>
        <fileType>
            <name>adobe.VideoFile</name>
            <extension>avf</extension>
            <description>Adobe Video File</description>
            <contentType>application/vnd.adobe.video-file</contentType>
            <icon>
                <image16x16>icons/avfIcon_16.png</image16x16>
                <image32x32>icons/avfIcon_32.png</image32x32>
                <image48x48>icons/avfIcon_48.png</image48x48>
                <image128x128>icons/avfIcon_128.png</image128x128>
            </icon>
        </fileType>
    </fileTypes>
</application>
```

응용 프로그램이 SWF 파일이 아닌 HTML 파일을 루트 내용으로 사용하는 경우 <content> 요소만 달라집니다.

```
<content>
    HelloWorld.html
</content>
```

응용 프로그램 설명자 파일의 속성 정의

응용 프로그램 설명자의 XML 요소 및 특성을 사용하여 AIR 응용 프로그램에 다음 유형의 속성을 정의할 수 있습니다.

- 필요한 AIR 런타임 버전
- 응용 프로그램 ID
- 설치 및 프로그램 메뉴 폴더
- 초기 내용 및 윈도우 속성
- 응용 프로그램 아이콘 파일
- 응용 프로그램에서 사용자 정의 업데이트 사용자 인터페이스를 제공하는지 여부
- 사용자의 브라우저에서 실행되는 SWF 내용이 응용 프로그램을 호출할 수 있는지 여부
- 파일 유형 연결

필요한 AIR 버전 지정

응용 프로그램 설명자 파일의 루트 요소 특성 application은 필요한 AIR 런타임 버전을 지정합니다.

```
<application xmlns="http://ns.adobe.com/air/application/2.0">
```

xmlns 기본 XML 네임스페이스로 정의해야 하는 AIR 네임스페이스입니다. 이 네임스페이스는 보조 패치가 아니라 AIR의 주요 릴리스가 출시될 때마다 변경됩니다. "2.0"과 같은 네임스페이스의 마지막 세그먼트는 응용 프로그램에 필요한 런타임 버전을 나타냅니다. 응용 프로그램에서 AIR 2의 새로운 기능을 사용하는 경우 네임스페이스를 AIR 2("http://ns.adobe.com/air/application/2.0")로 설정하십시오.

SWF 기반 응용 프로그램의 경우 응용 프로그램 설명자에 지정된 AIR 런타임 버전에 따라 응용 프로그램의 초기 내용으로 로드할 수 있는 최대 SWF 버전이 결정됩니다. AIR 1.0 또는 AIR 1.1이 지정된 응용 프로그램에서는 AIR 2 런타임을 사용하여 실행하는 경우에도 초기 내용으로 SWF9(Flash Player 9) 파일만 사용할 수 있습니다. AIR 1.5(또는 그 이상)가 지정된 응용 프로그램에서는 초기 내용으로 SWF9 또는 SWF10(Flash Player 10) 파일을 사용할 수 있습니다.

SWF 버전에 따라 사용할 수 있는 AIR 및 Flash Player API 버전이 달라집니다. SWF9 파일이 AIR 1.5 응용 프로그램의 초기 내용으로 사용되는 경우 응용 프로그램에서 AIR 1.1 및 Flash Player 9 API에만 액세스할 수 있습니다. 이 경우 AIR 2.0 또는 Flash Player 10.1에서 기존 API에 이루어진 비헤이비어 변경 사항도 적용되지 않습니다. 단, API에 대한 중요한 보안 관련 변경 사항은 이 원칙에 대한 예외로서 현재 또는 향후 런타임 패치에서 소급 적용할 수 있습니다.

HTML 기반 응용 프로그램의 경우 응용 프로그램 설명자에 지정된 런타임 버전으로 응용 프로그램에 사용할 수 있는 AIR 및 Flash Player API 버전이 결정됩니다. HTML, CSS 및 JavaScript 비헤이비어는 응용 프로그램 설명자에 의해 결정되는 것이 아니라 항상 설치된 AIR 런타임에 사용된 Webkit 버전에 의해 결정됩니다.

AIR 응용 프로그램에서 SWF 내용을 로드할 때 해당 내용에 사용할 수 있는 AIR 및 Flash Player API 버전은 내용을 로드하는 방법에 따라 달라집니다. 유효한 버전은 응용 프로그램 설명자 네임스페이스로 결정되거나, 로딩하는 내용의 버전에 따라 결정되거나, 로드된 내용의 버전에 따라 결정됩니다. 다음 표에서는 내용을 로드하는 방법에 따라 API 버전이 어떻게 결정되는지 보여 줍니다.

내용 로드 방법	**API 버전 결정 방법**
초기 내용, SWF 기반 응용 프로그램	**로드되는** SWF 파일 버전
초기 내용, HTML 기반 응용 프로그램	응용 프로그램 설명자 네임스페이스
SWF 내용에서 로드되는 SWF	**로드하는** 내용의 버전
<script> 태그를 사용하여 HTML 내용에서 로드되는 SWF 라이브러리	응용 프로그램 설명자 네임스페이스
AIR 또는 Flash Player API(예: flash.display.Loader)를 사용하여 HTML에서 로드되는 SWF	응용 프로그램 설명자 네임스페이스
<object> 또는 <embed> 태그(또는 이에 해당하는 JavaScript API)를 사용하여 HTML 내용에서 로드되는 SWF	**로드되는** SWF 파일 버전

로드하는 내용과 다른 버전의 SWF 파일을 로드할 경우 두 가지 문제가 발생할 수 있습니다.

- SWF9 이하 버전을 사용하여 SWF10 내용을 로드하는 경우 - 로드된 내용에서 AIR 1.5 이상 및 Flash Player 10 이상 API에 대한 참조를 확인할 수 없습니다.
- SWF 10을 사용하여 SWF9 이하 버전의 내용을 로드하는 경우 - AIR 1.5 및 Flash Player 10에서 변경된 API가 로드된 내용에서 예상되는 것과 다르게 동작할 수 있습니다.

응용 프로그램 ID 정의

다음 요소는 응용 프로그램 ID, 제작자 ID, 버전, 이름, 파일 이름, 설명 및 저작권 정보를 정의합니다.

```
<id>TestApp</id>
<publisherID>48B5E02D9FB21E6389F73B8D17023320485DF8CE.1</publisherID>
<version>2.0</version>
<filename>TestApp</filename>
<name>
    <text xml:lang="en">Hello AIR</text>
    <text xml:lang="fr">Bonjour AIR</text>
    <text xml:lang="es">Hola AIR</text>
</name>
<description>An MP3 player.</description>
<copyright>Copyright (c) 2008 YourCompany, Inc.</copyright>
```

id 응용 프로그램에 대한 식별자 문자열로, 응용 프로그램 ID라고 합니다. 특성 값은 다음 문자로 제한됩니다.

- 0–9
- a–z
- A–Z
- . (도트)
- -(하이픈)

이 값에는 1-212개의 문자를 포함해야 합니다. 이 요소는 필수 사항입니다.

publisherID(선택 사항) 1.5.2 이하 버전의 AIR 응용 프로그램을 AIR 1.5.3 이상 버전으로 업데이트할 때 사용됩니다. 이 요소에는 이전 버전 AIR 응용 프로그램의 제작자 ID가 포함되어 있습니다. AIR 1.5.3 이상을 위한 응용 프로그램을 개발 중이며 AIR 1.5.2 이하용 기존 버전이 있는 경우에만 이 요소를 지정하십시오. 자세한 내용은 85페이지의 “AIR 제작자 ID”를 참조하십시오.

AIR 1.5.3 이상을 위한 AIR 응용 프로그램을 개발하는데 AIR 1.5.2 이하 버전을 사용하는 응용 프로그램 버전이 있는 경우 다음과 같이 합니다.

1. 응용 프로그램의 현재 제작자 ID를 확인합니다. 설치된 응용 프로그램의 META-INF/AIR/publisherid 파일에서 현재 제작자 ID를 찾을 수 있습니다.
2. 새 응용 프로그램에 대한 응용 프로그램 설명자 파일의 <publisherID> 요소에 제작자 ID를 추가합니다.

version 응용 프로그램에 대한 버전 정보를 지정합니다. 이 정보는 런타임 버전과 관련이 없습니다. 버전 문자열은 응용 프로그램이 정의하는 지정자입니다. AIR는 어떤 방식으로도 버전 문자열을 해석하지 않습니다. 따라서 "3.0" 버전이 "2.0" 버전보다 최신 버전이라고 가정할 수 없습니다. 몇 가지 예로는 "1.0", ".4", "0.5", "4.9", "1.3.4a" 등을 들 수 있습니다. 이 요소는 필수 사항입니다.

filename 응용 프로그램을 설치할 때 응용 프로그램의 파일 이름(확장명 제외)으로 사용할 문자열입니다. 이 응용 프로그램 파일은 런타임에 AIR 응용 프로그램을 시작합니다. name 값을 제공하지 않으면 filename이 설치 폴더의 이름으로도 사용됩니다. 이 요소는 필수 사항입니다.

filename 속성에는 다양한 파일 시스템에서 파일 이름으로 사용할 수 없는 다음 문자를 제외한 모든 유니코드(UTF-8) 문자를 포함할 수 있습니다.

문자	16진수 코드
다양함	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E

문자	16진수 코드
?	x3F
\	x5C
|	x7C

filename 값은 마침표로 끝낼 수 없습니다.

name(선택 사항이지만 사용하는 것이 좋음) AIR 응용 프로그램 설치 프로그램에 표시되는 제목입니다.

다중 text 요소 대신 단일 텍스트 노드를 지정하는 경우 AIR 응용 프로그램 설치 프로그램은 시스템 언어와 상관없이 이 이름을 사용합니다.

```
<name>Test Application</name>
```

AIR 1.0 응용 프로그램 설명자 스키마를 사용하면 이름에 대해 여러 text 요소가 아닌 하나의 간단한 텍스트 노드만 정의할 수 있습니다.

AIR 1.1 이상에서는 name 요소에 다중 언어를 지정할 수 있습니다. 예를 들어 다음에서는 이름을 영어, 프랑스어, 스페인어 등 세 가지 언어로 지정합니다.

```
<name>
    <text xml:lang="en">Hello AIR</text>
    <text xml:lang="fr">Bonjour AIR</text>
    <text xml:lang="es">Hola AIR</text>
</name>
```

각 text 요소에 대한 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

AIR 응용 프로그램 설치 프로그램은 사용되는 운영 체제의 사용자 인터페이스 언어에 가장 가까운 이름을 사용합니다. 예를 들어 응용 프로그램 설명자 파일의 name 요소에 en(영어) 로캘의 값이 포함된 설치를 살펴보면 운영 체제가 en(영어)을 사용자 인터페이스 언어로 식별하는 경우 AIR 응용 프로그램 설치 프로그램은 en 이름을 사용합니다. 시스템 사용자 인터페이스 언어가 en-US(미국 영어)인 경우에도 en 이름이 사용됩니다. 그러나 사용자 인터페이스 언어가 en-US이며 응용 프로그램 설명자 파일이 en-US 이름과 en-GB 이름을 모두 정의하는 경우 AIR 응용 프로그램 설치 프로그램은 en-US 값을 사용합니다. 응용 프로그램이 시스템 사용자 인터페이스 언어와 일치하는 이름을 정의하지 않는 경우 AIR 응용 프로그램 설치 프로그램은 응용 프로그램 설명자 파일에 정의된 첫 번째 name 값을 사용합니다.

name 요소가 지정되어 있지 않은 경우 AIR 응용 프로그램 설치 프로그램은 filename을 응용 프로그램 이름으로 표시합니다.

name 요소는 AIR 응용 프로그램 설치 프로그램에 사용되는 응용 프로그램 제목만 정의합니다. AIR 응용 프로그램 설치 프로그램은 중국어 번체, 중국어 간체, 체코어, 네덜란드어, 영어, 프랑스어, 독일어, 이탈리아어, 일본어, 한국어, 폴란드어, 포르투갈어(브라질), 러시아어, 스페인어, 스웨덴어, 터키어 등 다양한 언어를 지원합니다. AIR 응용 프로그램 설치 프로그램은 시스템 사용자 인터페이스 언어를 기반으로 응용 프로그램 제목 및 설명 외 텍스트에 대한 표시 언어를 선택합니다. 이러한 언어 선택 작업은 응용 프로그램 설명자 파일의 설정과는 별개로 수행됩니다.

name 요소는 설치되어 실행 중인 응용 프로그램에 대해 사용 가능한 로캘을 정의하지 **않습니다**. 다중 언어 응용 프로그램을 개발하는 방법에 대한 자세한 내용은 124페이지의 “AIR 응용 프로그램 지역화”를 참조하십시오.

description(선택 사항) AIR 응용 프로그램 설치 프로그램에 표시되는 응용 프로그램에 대한 설명입니다.

다중 text 요소 대신 단일 텍스트 노드를 지정하는 경우 AIR 응용 프로그램 설치 프로그램은 시스템 언어와 상관없이 이 설명을 사용합니다.

```
<description>This is a sample AIR application.</description>
```

AIR 1.0 응용 프로그램 설명자 스키마를 사용하면 이름에 대해 여러 text 요소가 아닌 하나의 간단한 텍스트 노드만 정의할 수 있습니다.

AIR 1.1 이상에서는 description 요소에 다중 언어를 지정할 수 있습니다. 예를 들어 다음에서는 설명을 영어, 프랑스어, 스페인어 등 세 가지 언어로 지정합니다.

```
<description>
    <text xml:lang="en">This is a example.</text>
    <text xml:lang="fr">C'est un exemple.</text>
    <text xml:lang="es">Esto es un ejemplo.</text>
</description>
```

각 text 요소에 대한 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

AIR 응용 프로그램 설치 프로그램은 사용되는 운영 체제의 사용자 인터페이스 언어에 가장 가까운 설명을 사용합니다. 예를 들어 응용 프로그램 설명자 파일의 description 요소에 en(영어) 로캘의 값이 포함된 설치를 살펴보면 사용자 시스템이 en(영어)을 사용자 인터페이스 언어로 식별하는 경우 AIR 응용 프로그램 설치 프로그램은 en 이름을 사용합니다. 시스템 사용자 인터페이스 언어가 en-US(미국 영어)인 경우에도 en 이름이 사용됩니다. 그러나 시스템 사용자 인터페이스 언어가 en-US이며 응용 프로그램 설명자 파일이 en-US 이름과 en-GB 이름을 모두 정의하는 경우 AIR 응용 프로그램 설치 프로그램은 en-US 값을 사용합니다. 응용 프로그램이 시스템 사용자 인터페이스 언어와 일치하는 이름을 정의하지 않는 경우 AIR 응용 프로그램 설치 프로그램은 응용 프로그램 설명자 파일에 정의된 첫 번째 description 값을 사용합니다.

다중 언어 응용 프로그램을 개발하는 방법에 대한 자세한 내용은 124페이지의 “AIR 응용 프로그램 지역화”를 참조하십시오.

copyright(선택 사항) AIR 응용 프로그램에 대한 저작권 정보입니다. Mac OS에서 저작권 텍스트는 설치된 응용 프로그램의 [정보] 대화 상자에 나타납니다. Mac OS에서는 저작권 정보가 응용 프로그램에 대한 info.plist 파일의 NSHumanReadableCopyright 필드에도 사용됩니다.

대상 응용 프로그램 프로파일 제한

ActionScript 3.0을 사용하여 구축된 AIR 2 및 iPhone 응용 프로그램을 사용하면 컴파일된 응용 프로그램의 대상 프로파일을 제한할 수 있습니다.

```
<supportedProfiles>extendedDesktop</supportedProfiles>
```

supportedProfiles(선택 사항) 응용 프로그램에서 지원하는 프로파일을 식별합니다.

supportedProfiles 요소에는 다음 세 값 중 하나를 사용할 수 있습니다.

- desktop - 데스크톱 프로파일은 AIR 파일을 사용하여 데스크톱 컴퓨터에 설치되는 AIR 응용 프로그램을 정의합니다. 이러한 응용 프로그램은 기본 응용 프로그램과의 통신을 제공하는 NativeProcess 클래스에 액세스할 수 없습니다.
- extendedDesktop - 확장 데스크톱 프로파일은 기본 응용 프로그램 설치 프로그램을 사용하여 데스크톱 컴퓨터에 설치되는 AIR 응용 프로그램을 정의합니다. 이러한 응용 프로그램은 기본 응용 프로그램과의 통신을 제공하는 NativeProcess 클래스에 액세스할 수 있습니다. 자세한 내용은 57페이지의 “기본 설치 프로그램의 AIR 응용 프로그램 패키지”를 참조하십시오. 또한 AIR의 기본 프로세스와 통신(ActionScript 개발자용) 및 Communicating with native processes in AIR(HTML 개발자용)을 참조하십시오.
- mobileDevice - 휴대 장치 프로파일은 ActionScript 3.0을 사용하여 제작된 iPhone 응용 프로그램을 정의합니다.
- extendedMobileDevice - 확장 휴대 장치 프로파일이 현재 사용되고 있지 않습니다.

supportedProfiles 속성은 선택 사항입니다. 응용 프로그램 설명자 파일에 이 요소를 포함하지 않으면 모든 프로파일에서 응용 프로그램을 컴파일하고 배포할 수 있습니다.

프로파일을 여러 개 지정하려면 공백 문자로 프로파일을 구분하십시오. 예를 들어, 다음 설정은 응용 프로그램을 데스크톱 및 확장된 프로파일에서만 사용할 수 있도록 지정합니다.

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

자세한 내용은 72페이지의 “응용 프로그램 프로파일”을 참조하십시오.

설치 폴더 및 프로그램 메뉴 폴더 정의

설치 및 프로그램 메뉴 폴더는 다음 속성 설정을 사용하여 정의됩니다.

```
<installFolder>Acme</installFolder>
<programMenuFolder>Acme/Applications</programMenuFolder>
```

installFolder(선택 사항) 기본 설치 디렉토리의 하위 디렉토리를 식별합니다.

기본 설치 하위 디렉토리는 Windows에서 Program Files 디렉토리이고 Mac OS에서는 /Applications 디렉토리입니다. Linux에서는 /opt/입니다. 예를 들어 installFolder 속성이 "Acme"로 설정되어 있으며 응용 프로그램의 이름이 "ExampleApp"로 지정되어 있는 경우 응용 프로그램은 Windows에서 C:\Program Files\Acme\ExampleApp에 설치되고 Mac OS에서는 /Applications/Acme/Example.app에 설치되며 Linux에서는 /opt/Acme/ExampleApp에 설치됩니다.

중첩된 하위 디렉토리를 지정하려면 다음과 같이 슬래시(/) 문자를 디렉토리 분리 기호 문자로 사용합니다.

```
<installFolder>Acme/Power Tools</installFolder>
```

installFolder 속성에는 다양한 파일 시스템에서 폴더 이름으로 사용할 수 없는 문자를 제외한 모든 유니코드(UTF-8) 문자를 포함할 수 있습니다. 예외 목록은 위의 filename 속성을 참조하십시오.

installFolder 속성은 선택 사항입니다. installFolder 속성을 지정하지 않으면 응용 프로그램이 name 속성을 기반으로 기본 설치 디렉토리의 하위 디렉토리에 설치됩니다.

programMenuFolder(선택 사항) Windows 운영 체제의 [모든 프로그램] 메뉴 또는 Linux의 [응응 프로그램] 메뉴에서 응용 프로그램에 대한 단축키를 배치할 위치를 식별합니다. 현재 다른 운영 체제에서는 이 설정이 무시됩니다. 속성의 값으로 허용되는 문자에 대한 제한 사항은 installFolder 속성에 대한 제한 사항과 같습니다. 이 값의 마지막 문자로 슬래시(/) 문자를 사용하지 마십시오.

초기 응용 프로그램 윈도우의 속성 정의

AIR 응용 프로그램이 로드되면 런타임은 initialWindow 요소의 값을 사용하여 응용 프로그램에 대한 초기 윈도우를 만듭니다. 그런 다음 런타임은 content 요소에 지정된 SWF 또는 HTML 파일을 윈도우로 로드합니다.

다음은 initialWindow 요소의 예제입니다.

```
<initialWindow>
    <content>AIRTunes.swf</content>
    <title>AIR Tunes</title>
    <systemChrome>none</systemChrome>
    <transparent>true</transparent>
    <visible>true</visible>
    <minimizable>true</minimizable>
    <maximizable>true</maximizable>
    <resizable>true</resizable>
    <width>400</width>
    <height>600</height>
    <x>150</x>
    <y>150</y>
    <minSize>300 300</minSize>
    <maxSize>800 800</maxSize>
</initialWindow>
```

응용 프로그램의 루트 내용이 SWF 파일이 아닌 HTML 파일인 경우 content 요소가 HTML 파일을 참조합니다.

```
<content>AIRTunes.html</content>
```

initialWindow 요소의 자식 요소는 루트 내용 파일이 로드되는 윈도우의 속성을 설정합니다.

content content 요소에 대해 지정되는 값은 응용 프로그램의 기본 내용 파일에 대한 URL입니다. 이는 SWF 파일 또는 HTML 파일일 수 있습니다. URL은 응용 프로그램 설치 폴더의 루트를 기준으로 지정됩니다. ADL을 사용하여 AIR 응용 프로그램을 실행하는 경우 URL은 응용 프로그램 설명자 파일이 포함된 폴더를 기준으로 합니다. ADL의 root-dir 매개 변수를 사용하여 다른 루트 디렉토리를 지정할 수 있습니다.

참고: content 요소의 값이 URL로 처리되기 때문에 내용 파일의 이름에 사용된 문자는 RFC 1738에 정의된 규칙에 따라 URL로 인코딩해야 합니다. 예를 들어 공백 문자는 %20으로 인코딩해야 합니다.

title(선택 사항) 윈도우 제목입니다.

systemChrome(선택 사항) 이 특성을 standard로 설정하면 운영 체제가 제공하는 표준 시스템 크롬이 표시됩니다. 이 특성을 none으로 설정하면 시스템 크롬이 표시되지 않습니다.

Flex mx:WindowedApplication 구성 요소를 사용할 경우 표준 시스템 크롬이 설정되어 있지 않으면 구성 요소가 해당 사용자 정의 크롬을 적용합니다. 런타임에는 시스템 크롬 설정을 변경할 수 없습니다.

transparent(선택 사항) 응용 프로그램 윈도우가 알파 블렌딩을 지원하도록 하려면 "true"로 설정합니다. 투명도를 갖춘 윈도우는 보다 느리게 그려지며 더 많은 메모리를 요구할 수 있습니다. 런타임에는 투명 설정을 변경할 수 없습니다.

중요: systemChrome이 none인 경우에만 transparent를 true로 설정할 수 있습니다.

visible(선택 사항) 기본값은 false입니다. 기본 윈도우가 만들어지자마자 표시되도록 하려는 경우에만 true로 설정합니다. 이때 윈도우 구성 요소가 배치되면서 윈도우에 적용된 변경 사항도 표시됩니다.

MXML 정의에서 visible 특성이 false로 설정되어 있지 않은 한 Flex mx:WindowedApplication 구성 요소는 applicationComplete 이벤트가 전달되기 직전에 윈도우를 자동으로 표시 및 활성화합니다.

윈도우 위치, 윈도우 크기 및 윈도우 내용의 레이아웃에 대한 변경 사항이 표시되지 않도록 처음에는 기본 윈도우를 숨겨 둘 수도 있습니다. 그런 다음 윈도우의 activate() 메서드를 호출하거나 visible 속성을 true로 설정하여 윈도우를 표시할 수 있습니다.

x, y, width, height(선택 사항) 응용 프로그램 기본 윈도우의 초기 경계입니다. 이러한 값을 설정하지 않으면 루트 SWF 파일(HTML의 경우 운영 체제)의 설정에 따라 윈도우 크기가 결정됩니다. width 및 height의 최대값은 2880입니다.

minSize, maxSize(선택 사항) 윈도우의 최소 및 최대 크기입니다. 이러한 값을 설정하지 않으면 운영 체제에서 해당 크기가 결정됩니다.

minimizable, maximizable, resizable(선택 사항) 윈도우에 대해 최소화, 최대화 및 크기 조절 작업을 수행할 수 있는지 여부를 지정합니다. 기본적으로 이러한 설정에는 기본값인 true가 사용됩니다.

참고: 윈도우 최대화가 크기 조절 작업인 Mac OS X와 같은 운영 체제에서 윈도우가 확대/축소 또는 크기 조절되지 않도록 하려면 maximizable과 resizable을 모두 false로 설정해야 합니다.

기타 도움말 항목

HTML에 SWF 내용 포함(ActionScript 개발자용)

Embedding SWF content in HTML(HTML 개발자용)

AIR의 PDF 내용 추가(ActionScript 개발자용)

Adding PDF content in AIR(HTML 개발자용)

AIR 기본 윈도우를 사용한 작업(ActionScript 개발자용)

Working with AIR native windows(HTML 개발자용)

아이콘 파일 지정

icon 속성은 응용 프로그램에 사용될 하나 이상의 아이콘 파일을 지정합니다. 아이콘 포함은 선택 사항입니다. icon 속성을 지정하지 않으면 운영 체제가 기본 아이콘을 표시합니다.

지정된 경로는 응용 프로그램 루트 디렉토리를 기준으로 합니다. 아이콘 파일은 PNG 형식이어야 합니다. 모든 다음 아이콘 크기를 지정할 수 있습니다.

```
<icon>
    <image16x16>icons/smallIcon.png</image16x16>
    <image32x32>icons/mediumIcon.png</image32x32>
    <image48x48>icons/bigIcon.png</image48x48>
    <image128x128>icons/biggestIcon.png</image128x128>
</icon>
```

지정된 크기에 대한 요소가 있는 경우 파일의 이미지는 지정된 크기와 일치해야 합니다. 아무 크기도 제공되지 않은 경우에는 운영 체제에서 가장 근접한 크기가 아이콘의 지정된 용도에 맞게 조절됩니다.

참고: 지정된 아이콘은 AIR 패키지에 자동으로 추가되지 않으며 응용 프로그램이 패키지될 때 아이콘 파일을 정확한 해당 상대적 위치에 포함해야 합니다.

최적의 결과를 얻으려면 사용 가능한 크기 각각에 대한 이미지를 제공하십시오. 또한 아이콘이 16비트 색상 모드와 32비트 색상 모드에서 제대로 표시되는지 확인하십시오.

응용 프로그램 업데이트를 위한 사용자 정의 사용자 인터페이스 제공

AIR는 기본 설치 대화 상자를 사용하여 응용 프로그램을 설치 및 업데이트합니다. 그러나 응용 프로그램을 업데이트할 때 사용할 고유한 사용자 인터페이스를 제공할 수 있습니다. 응용 프로그램이 업데이트 프로세스를 자체적으로 처리해야 함을 나타내려면 customUpdateUI 요소를 true로 설정합니다.

```
<customUpdateUI>true</customUpdateUI>
```

설치된 응용 프로그램 버전의 customUpdateUI 요소가 true로 설정된 상태에서 사용자가 새 버전에 대한 AIR 파일을 두 번 클릭하거나 연속 설치 기능을 사용하여 응용 프로그램의 업데이트를 설치하면 런타임은 설치된 버전의 응용 프로그램을 열며, 기본 AIR 설치 프로그램은 열지 않습니다. 그러면 응용 프로그램 논리에서 업데이트 작업의 진행 방법을 결정할 수 있습니다. AIR 파일의 응용 프로그램 ID 및 제작자 ID가 설치된 응용 프로그램의 해당 값과 일치해야 업그레이드를 진행할 수 있습니다.

참고: 응용 프로그램이 이미 설치되어 있으며 사용자가 업데이트가 포함된 AIR 설치 파일을 두 번 클릭하거나 연속 설치 기능을 사용하여 응용 프로그램의 업데이트를 설치하는 경우에만 customUpdateUI 메커니즘이 작동합니다. customUpdateUI가 true인지 여부에 관계없이 필요에 따라 사용자 정의 UI를 표시하며 고유한 응용 프로그램 논리를 통해 업데이트를 다운로드 및 시작할 수 있습니다.

자세한 내용은 91페이지의 “AIR 응용 프로그램 업데이트”를 참조하십시오.

응용 프로그램의 브라우저 호출 허용

다음 설정을 지정하면 설치된 AIR 응용 프로그램을 브라우저 호출 기능을 통해 시작할 수 있습니다(사용자가 웹 브라우저의 페이지에서 링크 클릭).

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

기본값은 false입니다.

이 값을 true로 설정하는 경우 보안 영향을 고려해야 합니다. 이러한 보안 영향은 브라우저에서 AIR 응용 프로그램 호출(ActionScript 개발자용) 및 Invoking an AIR application from the browser(HTML 개발자용)에 설명되어 있습니다.

자세한 내용은 76페이지의 “웹 페이지에서 AIR 응용 프로그램 설치 및 실행”을 참조하십시오.

파일 유형 연결 선언

fileTypes 요소를 사용하면 AIR 응용 프로그램을 연결할 수 있는 파일 유형을 선언할 수 있습니다. AIR 응용 프로그램이 설치되면 선언된 모든 파일 유형이 운영 체제에 등록됩니다. 이러한 파일 유형이 아직 다른 응용 프로그램과 연결되어 있지 않은 경우 AIR 응용 프로그램과 연결됩니다. 파일 유형과 다른 응용 프로그램 간의 기존 연결을 재정의하려면 런타임에 NativeApplication.setAsDefaultApplication() 메서드를 사용합니다. 이때 사용자의 허락을 받는 것이 좋습니다.

참고: 런타임 메서드는 응용 프로그램 설명자에 선언된 파일 유형에 대한 연결만 관리할 수 있습니다.

```
<fileTypes>
    <fileType>
        <name>adobe.VideoFile</name>
        <extension>avf</extension>
        <description>Adobe Video File</description>
        <contentType>application/vnd.adobe.video-file</contentType>
        <icon>
            <image16x16>icons/AIRApp_16.png</image16x16>
            <image32x32>icons/AIRApp_32.png</image32x32>
            <image48x48>icons/AIRApp_48.png</image48x48>
            <image128x128>icons/AIRApp_128.png</image128x128>
        </icon>
    </fileType>
</fileTypes>
```

fileTypes 요소는 선택 사항입니다. 이 요소에는 원하는 수의 fileType 요소를 포함할 수 있습니다.

포함하는 각 fileType 선언에 name 및 extension 요소가 필요합니다. 여러 확장명에 동일한 이름을 사용할 수 있습니다. 확장명은 파일 유형을 고유하게 식별합니다. 확장명을 지정할 때는 마침표를 앞에 넣지 마십시오. description 요소는 선택 사항이며 운영 체제 사용자 인터페이스를 통해 사용자에게 표시됩니다. contentType은 AIR 1.0 및 1.1에서 선택 사항이지만 AIR 1.5에서는 필수입니다. 속성은 운영 체제가 특정 상황에서 파일을 열기 위한 최적의 응용 프로그램을 찾을 수 있도록 도움을 줍니다. 값은 파일 내용의 MIME 유형이어야 합니다. 해당 파일 형식이 이미 등록되어 있고 지정된 MIME 유형을 갖는 경우 Linux에서 값이 무시됩니다.

응용 프로그램 아이콘 요소와 동일한 형식을 사용하여 파일 확장명에 아이콘을 지정할 수 있습니다. 아이콘 파일도 자동으로 패키지되지 않기 때문에 AIR 설치 파일에 포함해야 합니다.

파일 유형이 AIR 응용 프로그램과 연결되면 사용자가 해당 유형의 파일을 열 때마다 AIR 응용 프로그램이 호출됩니다. 해당 응용 프로그램이 이미 실행되고 있는 경우 AIR는 실행 중인 인스턴스에 InvokeEvent 객체를 전달합니다. 그렇지 않으면 AIR는 먼저 해당 응용 프로그램을 시작합니다. 두 경우 모두 NativeApplication 객체를 통해 전달된 InvokeEvent 객체에서 파일 경로를 검색할 수 있습니다. 이 경로를 사용하여 파일을 열 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- 파일 연결 관리(ActionScript 개발자용)
- Managing file associations(HTML 개발자용)
- 명령줄 인수 캡처(ActionScript 개발자용)
- Capturing command line arguments(HTML 개발자용)

12장: 응용 프로그램 프로파일

Adobe AIR 2 이상

Adobe AIR 2에는 프로파일의 개념이 도입되었습니다. 프로파일은 지원되는 API 및 기능 세트를 정의합니다. 각 프로파일은 지정된 컴퓨터 또는 장치 세트에 설치될 수 있습니다.

예를 들어 데스크톱 프로파일과 확장 데스크톱 프로파일은 모두 데스크톱 AIR 응용 프로그램에 사용됩니다. 그러나 확장 데스크톱 프로파일 응용 프로그램은 기본 프로세스와 통신할 수 있지만 데스크톱 프로파일 응용 프로그램은 기본 프로세스와 통신할 수 없습니다. 마찬가지로 휴대 장치 프로파일은 다른 프로파일에서 사용할 수 있는 기능과는 다른 기능 세트를 정의합니다.

다음과 같은 네 가지 프로파일이 있습니다.

데스크톱 데스크톱 프로파일은 데스크톱 컴퓨터에 AIR 파일로 설치되는 AIR 응용 프로그램의 기능 세트를 정의합니다. 이러한 응용 프로그램은 지원되는 데스크톱 플랫폼(Mac OS, Windows 및 Linux)에서 설치되어 실행됩니다. AIR 2 이전 버전에서 개발된 AIR 응용 프로그램은 데스크톱 프로파일로 간주될 수 있습니다. 일부 API는 이 프로파일에서 작동하지 않습니다. 예를 들어, 데스크톱 응용 프로그램이 기본 프로세스와 통신할 수 없습니다.

확장 데스크톱 확장 데스크톱 프로파일은 기본 설치 프로그램으로 패키지화되어 설치된 AIR 응용 프로그램의 기능 세트를 정의합니다. 이러한 기본 설치 프로그램은 Windows의 경우 EXE 파일, Mac OS의 경우 DMG 파일, Linux의 경우 DEB 또는 RPM 파일입니다. 확장 데스크톱 응용 프로그램에는 데스크톱 프로파일 응용 프로그램에 제공되지 않는 추가 기능이 있습니다. 기본 설치 프로그램으로 AIR 응용 프로그램을 패키지화하고 배포하는 기능은 AIR 2의 새 기능입니다. 자세한 내용은 57페이지의 “기본 설치 프로그램의 AIR 응용 프로그램 패키지”를 참조하십시오.

휴대 장치 휴대 장치 프로파일은 휴대 장치에 설치된 응용 프로그램의 기능 세트를 정의합니다. ActionScript 3.0 및 AIR API를 사용하여 iPhone, iPod touch 및 iPad용 응용 프로그램을 만들 수 있습니다. 현재까지는 휴대 장치 프로파일 응용 프로그램을 지원하는 장치입니다.

확장 휴대 장치 확장 휴대 장치 프로파일은 휴대 장치의 하위 세트에 설치된 응용 프로그램의 기능 세트를 정의합니다. 이 휴대 장치의 하위 세트는 휴대 장치 프로파일에 정의된 기능 외에도 HTMLLoader 클래스를 사용할 수 있습니다. 현재로서 이 프로파일을 지원하는 장치는 없습니다.

응용 프로그램 설명자 파일에 대상 프로파일 제한

AIR 2의 경우 응용 프로그램 설명자 파일에는 대상 프로파일을 제한할 수 있는 supportedProfiles 요소가 포함됩니다. 예를 들어, 다음 설정은 응용 프로그램을 데스크톱 프로파일에서만 사용할 수 있도록 지정합니다.

```
<supportedProfiles>desktop</supportedProfiles>
```

이 요소가 설정되면 사용자가 나열한 프로파일에서만 응용 프로그램을 패키지화할 수 있습니다. 다음 값을 사용합니다.

- desktop - 데스크톱 프로파일
- extendedDesktop - 확장 데스크톱 프로파일
- mobileDevice - 휴대 장치 프로파일

supportedProfiles 요소는 선택 사항입니다. 응용 프로그램 설명자 파일에 이 요소를 포함하지 않으면 모든 프로파일에서 응용 프로그램을 패키지화하고 배포할 수 있습니다.

supportedProfiles 요소에서 여러 프로파일을 지정하려면 다음과 같이 공백 문자를 사용하여 구분하십시오.

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

각 프로파일의 기능

일부 ActionScript API는 일부 프로파일에서 작동하지 않습니다.

데스크톱

데스크톱 프로파일 응용 프로그램은 모든 ActionScript API에 액세스할 수 있지만 다음과 같은 예외가 있습니다.

API	지원 테스트	설명
Accelerometer	Accelerometer.isSupported	Accelerometer 지원은 휴대 장치 프로파일에서만 제공됩니다.
CameraRoll	CameraRoll.isSupported	이 클래스는 iPhone에서 카메라 롤에 대한 액세스를 제공합니다.
File.openWithDefaultApplication()	-	데스크톱 프로파일 응용 프로그램은 이 메서드를 호출할 수 있습니다. 하지만 데스크톱 프로파일 응용 프로그램은 일부 유형의 파일을 열 수 없습니다. Adobe® Flash® Professional CS5용 ActionScript® 3.0용 언어 참조 설명서의 설명에는 제한된 파일 유형이 나열되어 있습니다.
File.cacheAsBitmapMatrix	-	이 속성은 iPhone의 ActionScript 3.0 응용 프로그램에서만 지원됩니다.
Geolocation	Geolocation.isSupported	Geolocation 지원은 휴대 장치 프로파일에서만 제공됩니다.
NativeApplication.systemIdleMode	없음	데스크톱 프로파일로 실행되는 응용 프로그램에서는 이 속성을 설정해도 효과가 없습니다.
NativeProcess	NativeProcess.isSupported	기본 프로세스 상호 작용은 확장 데스크톱 프로파일 응용 프로그램에서만 제공됩니다.
Stage.orientation	Stage.supportsOrientationChange	스테이지 방향 API는 휴대 장치 프로파일에서만 제공됩니다.

확장 데스크톱

확장 데스크톱 프로파일 응용 프로그램은 데스크톱 프로파일 응용 프로그램과 동일한 ActionScript API에 액세스할 수 있습니다. 하지만 두 가지 추가 기능이 있습니다.

- 확장 데스크톱 프로파일 응용 프로그램은 기본 프로세스를 실행하고 이와 상호 작용할 수 있습니다. 자세한 내용은 다음 중 하나를 참조하십시오.
  - AIR의 기본 프로세스와 통신(ActionScript 개발자용)
  - Communicating with native processes in AIR(HTML 개발자용)
- 확장 데스크톱 프로파일 응용 프로그램은 모든 유형의 파일에 대해 File.openWithDefaultApplication()을 호출할 수 있습니다. 데스크톱 프로파일에 적용되는 제한은 확장 데스크톱 프로파일에는 적용되지 않습니다. File.openWithDefaultApplication() 메서드를 사용하면 운영 체제에 등록된 기본 응용 프로그램을 사용하여 파일을 열 수 있습니다.

휴대 장치

휴대 장치 프로파일 응용 프로그램은 대부분의 ActionScript 3.0 API에 액세스할 수 있지만 예외 및 제한 사항이 있습니다. 또한 휴대 장치 프로파일 응용 프로그램에서만 작동하는 API도 있습니다. 자세한 내용은 iPhone 응용 프로그램에 대한 ActionScript 3.0 API 지원을 참조하십시오.

ADL로 디버깅할 때 프로파일 지정

ADL은 응용 프로그램 설명자 파일의 supportedProfiles 요소에서 지원되는 프로파일이 지정되었는지 확인합니다. 파일이 지정되었으면 디버그할 때 기본적으로 ADL은 프로파일로 나열된 것 중 지원되는 첫 번째 프로파일을 사용합니다.

-profile 명령줄 인수를 사용하여 ADL 디버그 세션에 대한 프로파일을 지정할 수 있습니다. 67페이지의 “대상 응용 프로그램 프로파일 제한”을 참조하십시오. 응용 프로그램 설명자 파일에서 supportedProfiles 요소에 프로파일을 지정하는지 여부에 관계없이 이 인수를 사용할 수 있습니다. 하지만 supportedProfiles 요소를 지정할 경우 명령줄에 지정한 프로파일을 포함해야 합니다. 그렇지 않으면 ADL에서 오류가 발생합니다.

13장: AIR 응용 프로그램 배포, 설치 및 실행

AIR 응용 프로그램은 응용 프로그램 코드와 모든 에셋이 포함된 단일 AIR 설치 파일로 배포됩니다. 다운로드, 전자 메일 또는 물리적 미디어(예: CD-ROM)와 같은 일반적인 수단을 통해 이 파일을 배포할 수 있습니다. 사용자는 AIR 파일을 두 번 클릭하여 응용 프로그램을 설치할 수 있습니다. 웹 페이지에서 사용자가 단일 링크를 클릭하여 AIR 응용 프로그램을 설치하고 필요한 경우 Adobe® AIR®도 설치할 수 있도록 하는 **연속 설치** 기능을 사용할 수 있습니다.

AIR 설치 파일을 배포하려면 먼저 패키지로 만들고 코드 서명 인증서 및 개인 키로 서명해야 합니다. 설치 파일에 디지털 서명하면 서명된 이후 응용 프로그램이 변경되지 않았음이 보장됩니다. 또한 신뢰할 수 있는 인증 기관에서 디지털 인증서를 발급한 경우 사용자가 제작자와 서명자 신원을 확인할 수 있습니다. 응용 프로그램이 ADT(AIR Developer Tool)로 패키지되면 AIR 파일이 서명됩니다.

AIR 파일에 응용 프로그램을 패키지화하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Adobe® Flex® Builder™: Flex Builder를 사용하여 AIR 응용 프로그램 패키지 참조
- Adobe® Flash® Builder™: Packaging AIR applications with Flash Builder 참조
- Adobe® Flash® Professional: Adobe AIR용으로 제작 참조
- Adobe® Dreamweaver®: Dreamweaver에서 AIR 응용 프로그램 만들기 참조
- Adobe® AIR® SDK의 경우 48페이지의 "ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지" 참조

데스크톱에서 AIR 응용 프로그램 설치 및 실행

AIR 파일은 단순하게 수신자에게 보내면 됩니다. 예를 들어 AIR 파일을 전자 메일 첨부 파일이나 웹 페이지의 링크로 보낼 수 있습니다.

사용자는 AIR 응용 프로그램을 다운로드한 후 다음 지침을 따라 설치합니다.

1 AIR 파일을 두 번 클릭합니다.

Adobe AIR가 컴퓨터에 이미 설치되어 있어야 합니다.

2 [설치] 윈도우에서 기본 설정을 선택한 상태로 두고 [계속]을 클릭합니다.

Windows에서 AIR는 자동으로 다음을 수행합니다.

- 응용 프로그램을 Program Files 디렉토리에 설치
- 응용 프로그램에 대한 바탕 화면 단축키 만들기
- [시작] 메뉴 단축키 만들기
- [제어판]의 [프로그램 추가/제거]에서 응용 프로그램에 대한 항목 추가

Mac OS에서 기본적으로 응용 프로그램은 Applications 디렉토리에 추가됩니다.

응용 프로그램이 이미 설치된 경우 설치 프로그램은 응용 프로그램의 기존 버전을 열 것인지, 아니면 다운로드한 AIR 파일의 버전으로 업데이트할 것인지를 사용자가 선택할 수 있게 합니다. 설치 프로그램은 AIR 파일의 응용 프로그램 ID 및 제작자 ID를 사용하여 응용 프로그램을 식별합니다.

3 설치가 완료되면 [완료]를 클릭합니다.

Mac OS에서는 업데이트된 버전의 응용 프로그램을 설치하려면 사용자에게 응용 프로그램 디렉토리에 설치할 수 있는 적절한 시스템 권한이 필요합니다. Windows 및 Linux에서는 사용자에게 관리 권한이 필요합니다.

응용 프로그램은 ActionScript 또는 JavaScript를 통해 새 버전을 설치할 수도 있습니다. 자세한 내용은 91페이지의 "AIR 응용 프로그램 업데이트"를 참조하십시오.

AIR 응용 프로그램이 설치되면 사용자는 다른 데스크톱 응용 프로그램과 마찬가지로 단순히 응용 프로그램 아이콘을 두 번 클릭하여 응용 프로그램을 실행합니다.

- Windows에서는 바탕 화면이나 폴더에 설치된 응용 프로그램 아이콘을 두 번 클릭하거나 [시작] 메뉴에서 응용 프로그램을 선택합니다.
- Linux에서는 바탕 화면이나 폴더에 설치된 응용 프로그램 아이콘을 두 번 클릭하거나 응용 프로그램 메뉴에서 응용 프로그램을 선택합니다.
- Mac OS에서는 응용 프로그램이 설치된 폴더에서 해당 응용 프로그램을 두 번 클릭합니다. 기본 설치 디렉토리는 /Applications 디렉토리입니다.

AIR 연속 설치 기능을 사용하면 사용자는 웹 페이지의 링크를 클릭하여 AIR 응용 프로그램을 설치할 수 있습니다. AIR 브라우저 호출 기능을 사용하면 사용자는 웹 페이지의 링크를 클릭하여 설치된 AIR 응용 프로그램을 실행할 수 있습니다. 이러한 기능에 대해서는 다음 단원에서 설명합니다.

웹 페이지에서 AIR 응용 프로그램 설치 및 실행

연속 설치 기능을 사용하면 웹 페이지에 SWF 파일을 포함할 수 있습니다. 이렇게 하면 사용자가 브라우저에서 AIR 응용 프로그램을 설치할 수 있습니다. 런타임이 설치되지 않은 경우 연속 설치 기능이 런타임을 설치합니다. 연속 설치 기능을 사용하여 사용자는 AIR 파일을 자신의 컴퓨터에 저장하지 않고 AIR 응용 프로그램을 설치할 수 있습니다.

연속 설치 기능을 쉽게 사용할 수 있게 해주는 badge.swf 파일이 AIR SDK 및 Flex SDK에 포함되어 있습니다. 자세한 내용은 77페이지의 "badge.swf 파일을 사용하여 AIR 응용 프로그램 설치"를 참조하십시오.

연속 설치 기능을 사용하는 방법에 대한 데모는 웹을 통한 AIR 응용 프로그램 배포 (http://www.adobe.com/go/learn_air_qs_seamless_install_kr) 퀵 스타트 샘플 문서를 참조하십시오.

연속 설치 badge.swf 사용자 정의

SDK와 함께 제공되는 badge.swf 파일을 사용하지 않고 고유 SWF 파일을 만들어 브라우저 페이지에 사용할 수도 있습니다. 사용자 정의 SWF 파일은 다음과 같은 방법으로 런타임과 상호 작용할 수 있습니다.

- AIR 응용 프로그램 설치. 81페이지의 "브라우저에서 AIR 응용 프로그램 설치"를 참조하십시오.
- 특정 AIR 응용 프로그램이 설치되어 있는지 확인. 80페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오.
- 런타임이 설치되어 있는지 확인. 80페이지의 "런타임이 설치되어 있는지 확인"을 참조하십시오.
- 사용자 시스템에 설치된 AIR 응용 프로그램 시작. 자세한 내용은 82페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

이러한 모든 기능은 adobe.com에 있는 SWF 파일(air.swf)의 API를 호출하여 제공됩니다. badge.swf 파일을 사용자 정의하여 사용자의 SWF 파일에서 air.swf API를 호출할 수 있습니다.

브라우저에서 실행되는 SWF 파일은 LocalConnection 클래스를 사용하여 실행 중인 AIR 응용 프로그램과 통신할 수도 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML·개발자용)을·참조하십시오.

중요: 이 단원에 설명된 기능 및 air.swf 파일의 API를 사용하려면 최종 사용자의 Windows 또는 Mac OS 웹 브라우저에 Adobe® Flash® Player 9 업데이트 3이 설치되어 있어야 합니다. Linux에서 연속 설치 기능을 사용하려면 Flash Player 10 버전 10,0,12,36 이상이 필요합니다. 코드를 작성하여 Flash Player의 설치된 버전을 확인하고 필요한 버전의 Flash Player가 설치되어 있지 않은 경우 사용자에게 대체 인터페이스를 제공할 수 있습니다. 예를 들어 이전 버전의 Flash Player가 설치되어 있는 경우 badge.swf 파일이나 air.swf API를 사용하여 응용 프로그램을 설치하는 대신 다운로드 버전의 AIR 파일에 대한 링크를 제공할 수 있습니다.

badge.swf 파일을 사용하여 AIR 응용 프로그램 설치

연속 설치 기능을 쉽게 사용할 수 있게 해주는 badge.swf 파일이 AIR SDK 및 Flex SDK에 포함되어 있습니다. badge.swf는 웹 페이지의 링크에서 런타임 및 AIR 응용 프로그램을 설치할 수 있습니다. 웹 사이트에 배포할 수 있도록 badge.swf 파일 및 소스 코드가 제공됩니다.

웹 페이지에 badge.swf 파일 포함

1 AIR SDK 또는 Flex SDK의 samples/badge 디렉토리에 제공되는 다음 파일을 찾아 웹 서버에 추가합니다.

- badge.swf
- default_badge.html
- AC_RunActiveContent.js

2 텍스트 편집기에서 default_badge.html 페이지를 엽니다.

3 default_badge.html 페이지의 AC_FL_RunContent() JavaScript 함수에서 다음에 대해 FlashVars 매개 변수 정의를 조정합니다.

매개 변수	설명
appname	런타임이 설치되어 있지 않을 때 SWF 파일에서 표시하는 응용 프로그램의 이름입니다.
appurl	(필수) 다운로드할 AIR 파일의 URL입니다. 상대 URL이 아닌 절대 URL을 사용해야 합니다.
airversion	(필수) 런타임 버전 1.0의 경우 이 값을 1.0으로 설정합니다.
imageurl	배지에 표시할 이미지의 URL(선택 사항)입니다.
buttoncolor	다운로드 버튼의 색입니다(FFCC00과 같은 16진수 값으로 지정).
messagecolor	런타임이 설치되어 있지 않을 때 버튼 아래에 표시되는 텍스트 메시지의 색입니다(FFCC00과 같은 16진수 값으로 지정).

4 badge.swf 파일의 최소 크기는 217 x 180픽셀입니다. 필요에 맞게 AC_FL_RunContent() 함수의 width 및 height 매개 변수의 값을 조정합니다.

5 default_badge.html 파일의 이름을 변경하고 필요에 맞게 코드를 조정하거나 다른 HTML 페이지에 포함합니다.

참고: badge.swf 파일을 로드하는 HTML embed 태그에서 wmode 특성을 설정하면 안 됩니다. 기본 설정인 "window"를 유지하십시오. 다른 wmode 설정을 사용하면 일부 시스템에서 설치가 되지 않습니다. 또한 다른 wmode 설정을 사용하면 "오류 #2044: 처리되지 않은 ErrorEvent:. text=Error #2074: 스테이지가 너무 작아 다운로드 UI에 맞지 않습니다." 오류가 생성됩니다.

badge.swf 파일을 편집하여 다시 컴파일할 수도 있습니다. 자세한 내용은 78페이지의 “badge.swf 파일 수정”을 참조하십시오.

웹 페이지의 연속 설치 링크에서 AIR 응용 프로그램 설치

페이지에 연속 설치 링크가 추가되면 사용자는 SWF 파일의 링크를 클릭하여 AIR 응용 프로그램을 설치할 수 있습니다.

1 Flash Player 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 설치된 웹 브라우저에서 HTML 페이지로 이동합니다.

2 웹 페이지에서 badge.swf 파일의 링크를 클릭합니다.

- 런타임을 설치한 경우 다음 단계로 건너뜁니다.
- 런타임을 설치하지 않은 경우 런타임을 설치할 것인지 여부를 묻는 대화 상자가 표시됩니다. 런타임을 설치(6페이지의 "Adobe AIR 설치" 참조)하고 다음 단계를 진행합니다.

3 [설치] 윈도우에서 기본 설정을 선택한 상태로 두고 [계속]을 클릭합니다.

Windows 컴퓨터에서 AIR는 자동으로 다음을 수행합니다.

- 응용 프로그램을 c:\Program Files\에 설치
- 응용 프로그램에 대한 바탕 화면 단축키 만들기
- [시작] 메뉴 단축키 만들기
- [제어판]의 [프로그램 추가/제거]에서 응용 프로그램에 대한 항목 추가

Mac OS에서 설치 프로그램은 응용 프로그램을 Applications 디렉토리(예: Mac OS의 /Applications 디렉토리)에 추가합니다.

Linux 컴퓨터에서 AIR는 자동으로 다음을 수행합니다.

- /opt에 응용 프로그램을 설치합니다.
- 응용 프로그램에 대한 바탕 화면 단축키 만들기
- [시작] 메뉴 단축키 만들기
- 시스템 패키지 관리자에서 응용 프로그램에 대한 항목을 추가합니다.

4 원하는 옵션을 선택한 다음 [설치] 버튼을 클릭합니다.

5 설치가 완료되면 [완료]를 클릭합니다.

badge.swf 파일 수정

Flex SDK 및 AIR SDK에서 badge.swf 파일에 소스 파일을 제공해 줍니다. 이러한 파일은 SDK의 samples/badge 폴더에 포함되어 있습니다.

소스 파일	설명
badge.fla	badge.swf 파일을 컴파일하는 데 사용되는 소스 Flash 파일입니다. badge.fla 파일은 SWF 9 파일(Flash Player에 로드할 수 있는 파일)로 컴파일됩니다.
AIRBadge.as	basdge.fla 파일에 사용되는 기본 클래스를 정의하는 ActionScript 3.0 클래스입니다.

Flash Professional을 사용하여 badge.fla 파일의 시각 인터페이스를 다시 설계할 수 있습니다.

AIRBadge 클래스에 정의되어 있는 AIRBadge() 생성자 함수는 http://airdownload.adobe.com/air/browserapi/air.swf에 있는 air.swf 파일을 로드합니다. air.swf 파일에는 연속 설치 기능을 사용하기 위한 코드가 포함됩니다.

air.swf 파일이 성공적으로 로드되면 AIRBadge 클래스의 onInit() 메서드가 호출됩니다.

```
private function onInit(e:Event):void {
    _air = e.target.content;
    switch (_air.getStatus()) {
        case "installed" :
            root.statusMessage.text = "";
            break;
        case "available" :
            if (_appName && _appName.length > 0) {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor + "'>In order to run " + _appName +
                        ", this installer will also set up Adobe® AIR®.</font></p>";
            } else {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor + "'>In order to run this application, "
                        + "this installer will also set up Adobe® AIR®.</font></p>";
            }
            break;
        case "unavailable" :
            root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor
                        + "'>Adobe® AIR® is not available for your system.</font></p>";
            root.buttonBg_mc.enabled = false;
            break;
    }
}
```

이 코드는 로드된 air.swf 파일의 기본 클래스에 전역 _air 변수를 설정합니다. 이 클래스에는 badge.swf 파일이 연속 설치 기능을 호출하기 위해 액세스하는 다음 공용 메서드가 포함됩니다.

메서드	설명
getStatus()	컴퓨터에 런타임이 설치되어 있는지 또는 런타임을 설치할 수 있는지 여부를 확인합니다. 자세한 내용은 80페이지의 "런타임이 설치되어 있는지 확인"을 참조하십시오.
installApplication()	사용자의 컴퓨터에 지정한 응용 프로그램을 설치합니다. 자세한 내용은 81페이지의 "브라우저에서 AIR 응용 프로그램 설치"를 참조하십시오. • url - URL을 정의하는 문자열입니다. 상대 URL 경로가 아닌 절대 URL 경로를 사용해야 합니다. • runtimeVersion - 설치할 응용 프로그램에 필요한 런타임의 버전을 나타내는 문자열(예: "1.0.M6")입니다. • arguments - 설치 시 시작되는 경우 응용 프로그램에 전달할 인수입니다. 응용 프로그램 설명자 파일에서 allowBrowserInvocation 요소가 true로 설정되어 있는 경우 설치 시 응용 프로그램이 시작됩니다. 응용 프로그램 설명자 파일에 대한 자세한 내용은 62페이지의 "AIR 응용 프로그램 속성 설정"을 참조하십시오. 사용자가 설치 시 시작 기능을 선택하여 브라우저에서 연속 설치의 결과로 응용 프로그램이 시작된 경우 응용 프로그램의 NativeApplication 객체는 인수가 전달된 경우에만 BrowserInvokeEvent 객체를 전달합니다. 응용 프로그램에 전달하는 데이터의 보안 문제를 고려하십시오. 자세한 내용은 82페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

url 및 runtimeVersion의 설정은 컨테이너 HTML 페이지의 FlashVars 설정을 통해 SWF 파일에 전달됩니다.

설치 시 응용 프로그램이 자동으로 시작되는 경우 LocalConnection 통신을 사용하여 badge.swf 파일 호출 시 설치된 응용 프로그램이 이 파일과 통신하도록 만들 수 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

응용 프로그램이 설치되어 있는지 여부를 확인하기 위해 air.swf 파일의 getApplicationVersion() 메서드를 호출할 수도 있습니다. 응용 프로그램 설치 프로세스가 시작되기 전이나 설치가 시작된 후에 이 메서드를 호출할 수 있습니다. 자세한 내용은 80페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오.

air.swf 파일 로드

브라우저의 웹 페이지에서 런타임 및 AIR 응용 프로그램과 상호 작용하기 위해 air.swf 파일의 API를 사용하는 고유 SWF 파일을 만들 수 있습니다. air.swf 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에 있습니다. SWF 파일에서 air.swf API를 참조하려면 SWF 파일과 같은 응용 프로그램 도메인에 air.swf 파일을 로드합니다. 다음 코드에서는 로드하는 SWF 파일의 응용 프로그램 도메인에 air.swf 파일을 로드하는 예제를 보여 줍니다.

```
var airSWF:Object; // This is the reference to the main class of air.swf
var airSWFLoader:Loader = new Loader(); // Used to load the SWF
var loaderContext:LoaderContext = new LoaderContext();
                                  // Used to set the application domain

loaderContext.applicationDomain = ApplicationDomain.currentDomain;

airSWFLoader.contentLoaderInfo.addEventListener(Event.INIT, onInit);
airSWFLoader.load(new URLRequest("http://airdownload.adobe.com/air/browserapi/air.swf"),
                    loaderContext);

function onInit(e:Event):void
{
    airSWF = e.target.content;
}
```

air.swf 파일이 로드되면(Loader 객체의 contentLoaderInfo 객체가 init 이벤트를 전달하면) 이후 단원에서 설명하는 모든 air.swf API를 호출할 수 있습니다.

참고: AIR SDK 및 Flex SDK와 함께 제공되는 badge.swf 파일은 air.swf 파일을 자동으로 로드합니다. 77페이지의 “badge.swf 파일을 사용하여 AIR 응용 프로그램 설치”를 참조하십시오. 이 단원의 지침은 air.swf 파일을 로드하는 고유 SWF 파일을 만드는 데 적용됩니다.

런타임이 설치되어 있는지 확인

SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 getStatus() 메서드를 호출하여 런타임이 설치되어 있는지 여부를 확인할 수 있습니다. 자세한 내용은 80페이지의 “air.swf 파일 로드”를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 getStatus() 메서드를 다음과 같이 호출할 수 있습니다.

```
var status:String = airSWF.getStatus();
```

getStatus() 메서드는 컴퓨터에 있는 런타임의 상태에 따라 다음 문자열 값 중 하나를 반환합니다.

문자열 값	설명
"available"	런타임을 이 컴퓨터에 설치할 수 있지만 현재 설치되어 있지 않습니다.
"unavailable"	이 컴퓨터에 런타임을 설치할 수 없습니다.
"installed"	이 컴퓨터에 런타임이 설치되어 있습니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 getStatus() 메서드에서 오류가 발생합니다.

AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인

SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 getApplicationVersion() 메서드를 호출하여 응용 프로그램 ID와 제작자 ID가 일치하는 AIR 응용 프로그램이 설치되어 있는지 여부를 확인할 수 있습니다. 자세한 내용은 80페이지의 “air.swf 파일 로드”를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 getApplicationVersion() 메서드를 다음과 같이 호출할 수 있습니다.

```
 var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
airSWF.getApplicationVersion(appID, pubID, versionDetectCallback);

function versionDetectCallback(version:String):void
{
    if (version == null)
    {
        trace("Not installed.");
        // Take appropriate actions. For instance, present the user with
        // an option to install the application.
    }
    else
    {
        trace("Version", version, "installed.");
        // Take appropriate actions. For instance, enable the
        // user interface to launch the application.
    }
}
```

getApplicationVersion() 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
appID	응용 프로그램의 응용 프로그램 ID입니다. 자세한 내용은 64페이지의 "응용 프로그램 ID 정의"를 참조하십시오.
pubID	응용 프로그램의 제작자 ID입니다. 자세한 내용은 85페이지의 "AIR 제작자 ID"를 참조하십시오. 해당 응용 프로그램에 제작자 ID가 없는 경우 pubID 매개 변수를 빈 문자열("")로 설정하십시오.
콜백	핸들러 함수의 역할을 하는 콜백 함수입니다. getApplicationVersion() 메서드는 비동기적으로 작동하며 설치된 버전을 검색하거나 설치된 버전이 없음을 확인하면 이 콜백 메서드가 호출됩니다. 콜백 메서드 정의에는 설치된 응용 프로그램의 버전 문자열로 설정된 문자열인 매개 변수가 하나 포함되어야 합니다. 응용 프로그램이 설치되지 않은 경우 이전 코드 샘플에 설명된 대로 null 값이 함수에 전달됩니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 getApplicationVersion() 메서드에서 오류가 발생합니다.

참고: AIR 1.5.3 현재, 제작자 ID는 더 이상 사용되지 않습니다. 더 이상 응용 프로그램에 제작자 ID가 자동으로 할당되지 않습니다. 이전 버전과의 호환성을 위해 응용 프로그램에서 제작자 ID를 지정하는 것은 가능합니다.

브라우저에서 AIR 응용 프로그램 설치

SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 installApplication() 메서드를 호출하여 AIR 응용 프로그램을 설치할 수 있습니다. 자세한 내용은 80페이지의 "air.swf 파일 로드"를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 installApplication() 메서드를 다음과 같이 호출할 수 있습니다.

```
 var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

installApplication() 메서드는 사용자의 컴퓨터에 지정한 응용 프로그램을 설치합니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
url	설치할 AIR 파일의 URL을 정의하는 문자열입니다. 상대 URL 경로가 아닌 절대 URL 경로를 사용해야 합니다.
runtimeVersion	설치할 응용 프로그램에 필요한 런타임의 버전을 나타내는 문자열(예: "1.0")입니다.
arguments	설치 시 시작되는 경우 응용 프로그램에 전달할 인수의 배열입니다. 영숫자 문자만 인수로 인식됩니다. 다른 값을 전달해야 할 경우 인코딩 스킴을 사용하십시오. 응용 프로그램 설명자 파일에서 allowBrowserInvocation 요소가 true로 설정되어 있는 경우 설치 시 응용 프로그램이 시작됩니다. 응용 프로그램 설명자 파일에 대한 자세한 내용은 62페이지의 "AIR 응용 프로그램 속성 설정"을 참조하십시오. 사용자가 설치 시 시작 기능을 선택하여 브라우저에서 연속 설치의 결과로 응용 프로그램이 시작된 경우 응용 프로그램의 NativeApplication 객체는 인수가 전달된 경우에만 BrowserInvokeEvent 객체를 전달합니다. 자세한 내용은 82페이지의 "브라우저에서 설치된 AIR 응용 프로그램 시작"을 참조하십시오.

마우스 클릭과 같은 사용자 이벤트에 대한 이벤트 핸들러에서 호출될 때만 installApplication() 메서드가 작동할 수 있습니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 installApplication() 메서드에서 오류가 발생합니다.

Mac OS에서는 업데이트된 버전의 응용 프로그램을 설치하려면 사용자에게 응용 프로그램 디렉토리에 설치할 수 있는 적절한 시스템 권한이 필요하며 응용 프로그램이 런타임을 업데이트하는 경우 관리 권한이 필요합니다. Windows에서는 사용자에게 관리 권한이 필요합니다.

응용 프로그램이 설치되어 있는지 여부를 확인하기 위해 air.swf 파일의 getApplicationVersion() 메서드를 호출할 수도 있습니다. 응용 프로그램 설치 프로세스가 시작되기 전이나 설치가 시작된 후에 이 메서드를 호출할 수 있습니다. 자세한 내용은 80페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오. 응용 프로그램이 실행되면 LocalConnection 클래스를 사용하여 브라우저의 SWF 내용과 통신할 수 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

브라우저에서 설치된 AIR 응용 프로그램 시작

브라우저 호출 기능을 사용하여 대상 응용 프로그램이 브라우저에서 시작되게 하려면 해당 응용 프로그램 설명자 파일에 다음 설정이 포함되어야 합니다.

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

응용 프로그램 설명자 파일에 대한 자세한 내용은 62페이지의 "AIR 응용 프로그램 속성 설정"을 참조하십시오.

브라우저의 SWF 파일은 http://airdownload.adobe.com/air/browserapi/air.swf에서 로드된 air.swf 파일의 launchApplication() 메서드를 호출하여 AIR 응용 프로그램을 시작할 수 있습니다. 자세한 내용은 80페이지의 "air.swf 파일 로드"를 참조하십시오.

air.swf 파일이 로드되면 SWF 파일은 air.swf 파일의 launchApplication() 메서드를 다음과 같이 호출할 수 있습니다.

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

launchApplication() 메서드는 사용자 인터페이스 SWF 파일의 응용 프로그램 도메인에 로드되는 air.swf 파일의 최상위 수준에 정의되어 있습니다. 이 메서드를 호출하면 응용 프로그램이 설치되어 있으며 응용 프로그램 설명자 파일의 allowBrowserInvocation 설정을 통해 브라우저 호출이 허용된 경우 AIR가 지정한 응용 프로그램을 시작합니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
appID	시작할 응용 프로그램의 응용 프로그램 ID입니다. 자세한 내용은 64페이지의 "응용 프로그램 ID 정의"를 참조하십시오.
pubID	시작할 응용 프로그램의 제작자 ID입니다. 자세한 내용은 85페이지의 "AIR 제작자 ID"를 참조하십시오. 해당 응용 프로그램에 제작자 ID가 없는 경우 pubID 매개 변수를 빈 문자열("")로 설정하십시오.
arguments	응용 프로그램에 전달할 인수의 배열입니다. 응용 프로그램의 NativeApplication 객체는 arguments 속성이 이 배열로 설정된 BrowserInvokeEvent 이벤트를 전달합니다. 영숫자 문자만 인수로 인식됩니다. 다른 값을 전달해야 할 경우 인코딩 스킴을 사용하십시오.

마우스 클릭과 같은 사용자 이벤트에 대한 이벤트 핸들러에서 호출될 때만 launchApplication() 메서드가 작동할 수 있습니다.

필요한 버전의 Flash Player, 즉 버전 9 업데이트 3 이상(Windows 및 Mac OS) 또는 버전 10(Linux)이 브라우저에 설치되어 있지 않은 경우 launchApplication() 메서드에서 오류가 발생합니다.

응용 프로그램 설명자 파일에서 allowBrowserInvocation 요소가 false로 설정된 경우 launchApplication() 메서드를 호출해도 효과가 없습니다.

응용 프로그램을 시작하는 사용자 인터페이스를 표시하기 전에 air.swf 파일에서 getApplicationVersion() 메서드를 호출할 수 있습니다. 자세한 내용은 80페이지의 "AIR 응용 프로그램이 설치되어 있는지 웹 페이지에서 확인"을 참조하십시오.

브라우저 호출 기능을 통해 응용 프로그램을 호출하면 응용 프로그램의 NativeApplication 객체가 BrowserInvokeEvent 객체를 전달합니다. 자세한 내용은 브라우저에서 AIR 응용 프로그램 호출(ActionScript 개발자용) 또는 Invoking an AIR application from the browser(HTML 개발자용)을 참조하십시오.

브라우저 호출 기능을 사용하는 경우 보안 영향을 고려해야 합니다. 이러한 보안 영향은 브라우저에서 AIR 응용 프로그램 호출(ActionScript 개발자용) 및 Invoking an AIR application from the browser(HTML 개발자용)에 설명되어 있습니다.

응용 프로그램이 실행되면 LocalConnection 클래스를 사용하여 브라우저의 SWF 내용과 통신할 수 있습니다. 자세한 내용은 다른 Flash Player 및 AIR 인스턴스와의 통신(ActionScript 개발자용) 또는 Communicating with other Flash Player and AIR instances(HTML 개발자용)을 참조하십시오.

참고: AIR 1.5.3 현재, 제작자 ID는 더 이상 사용되지 않습니다. 더 이상 응용 프로그램에 제작자 ID가 자동으로 할당되지 않습니다. 이전 버전과의 호환성을 위해 응용 프로그램에서 제작자 ID를 지정하는 것은 가능합니다.

엔터프라이즈 배포

IT 관리자는 Adobe AIR 런타임 및 AIR 응용 프로그램을 표준 데스크톱 배포 도구를 사용하여 자동으로 설치할 수 있습니다. IT 관리자는 다음을 수행할 수 있습니다.

- Microsoft SMS, IBM Tivoli 또는 부트스트래퍼 사용 자동 설치를 허용하는 기타 배포 도구와 같은 도구를 사용하여 Adobe AIR 런타임 자동 설치
- 런타임을 배포하는 데 사용된 것과 같은 도구를 사용하여 AIR 응용 프로그램 자동 설치

자세한 내용은 **Adobe AIR 관리자 안내서(http://www.adobe.com/go/learn_air_admin_guide_kr)**를 참조하십시오.

설치 로그

설치 로그는 AIR 런타임 자체 또는 AIR 응용 프로그램이 설치될 때 기록됩니다. 로그 파일을 검사하면 발생하는 모든 설치 또는 업데이트 문제의 원인을 확인하는 데 도움이 됩니다.

로그 파일은 다음 위치에 생성됩니다.

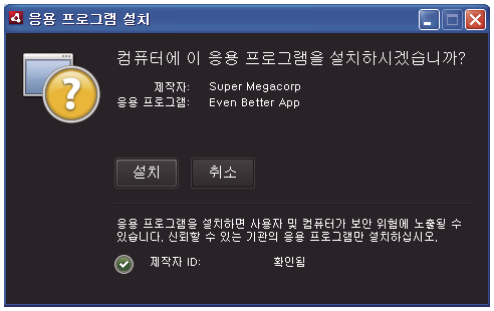
- Mac: 표준 시스템 로그(/private/var/log/system.log)

- Windows XP: C:\Documents and Settings\<사용자 이름>\Local Settings\Application Data\Adobe\AIR\logs\Install.log
- Windows Vista, Windows 7: C:\Users\<사용자 이름>\AppData\Local\Adobe\AIR\logs\Install.log
- Linux: /home/<사용자 이름>/.appdata/Adobe/AIR/Logs/Install.log

참고: 이러한 로그 파일은 AIR 2 이전의 AIR 버전에서는 생성되지 않습니다.

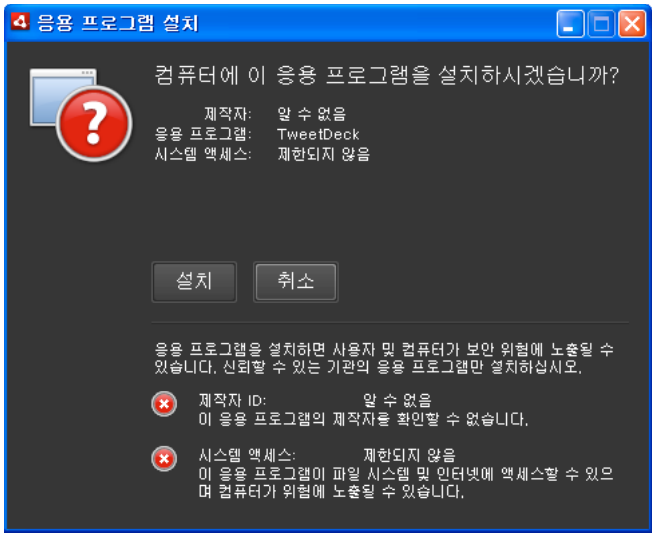
AIR 파일에 디지털 서명

공인된 CA(인증 기관)에서 발급한 인증서로 AIR 설치 파일에 디지털 서명하면 설치하는 응용 프로그램이 실수로 또는 악의적으로 변경되지 않았음이 확실히 보장되며 사용자가 서명자(제작자)의 신원을 확인할 수 있습니다. AIR는 AIR 응용 프로그램이 신뢰할 수 있는 인증서 또는 설치 컴퓨터에서 신뢰할 수 있는 인증서에 **체인으로 연결**된 인증서로 서명된 경우 설치하는 동안 제작자 이름을 표시합니다.



신뢰할 수 있는 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

자체 서명 인증서(또는 신뢰할 수 있는 인증서에 체인으로 연결되지 않은 인증서)로 응용 프로그램을 서명할 경우 사용자가 응용 프로그램 설치 시 더 큰 보안 위험을 감수해야 합니다. 설치 대화 상자에는 이러한 추가 위험에 대한 설명이 표시됩니다.



자체 서명 인증서로 서명된 응용 프로그램의 설치 확인 대화 상자

중요: 악의적인 사용자가 제작자의 서명 키 저장소 파일 또는 개인 키를 획득하게 될 경우 ID를 도용하여 AIR 파일을 위조할 수 있습니다.

코드 서명 인증서

코드 서명 인증서 사용과 관련된 보안 보증, 제한 사항 및 법적 규제는 인증 발급 기관에서 발행하는 CPS(Certificate Practice Statements) 및 구독자 계약서에 약술되어 있습니다. 현재 AIR 코드 서명 인증서를 발급하는 인증 기관의 계약서에 대한 자세한 내용은 다음을 참조하십시오.

ChosenSecurity(http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm)

ChosenSecurity CPS(http://www.chosensecurity.com/resource_center/repository.htm)

GlobalSign(http://www.globalsign.com/developer/code-signing-certificate/index.htm)

GlobalSign CPS(http://www.globalsign.com/repository/index.htm)

Thawte CPS(http://www.thawte.com/cps/index.html)

Thawte Code Signing Developer's Agreement(http://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/develcertsign.pdf)

VeriSign CPS(http://www.verisign.com/repository/CPS/)

VeriSign Subscriber's Agreement(https://www.verisign.com/repository/subscriber/SUBAGR.html)

AIR 코드 서명

AIR 파일에 서명하면 디지털 서명이 설치 파일에 포함됩니다. 서명에는 AIR 파일이 서명된 이후 변경되지 않았음을 확인하는 데 사용되는 패키지의 다이제스트와 제작자 ID를 확인하는 데 사용되는 서명 인증서에 대한 정보가 포함됩니다.

AIR는 운영 체제의 인증서 저장소를 통해 지원되는 PKI(공개 키 인프라)를 사용하여 인증서를 신뢰할 수 있는지 여부를 증명합니다. 제작자 정보를 확인하려면 AIR 응용 프로그램이 설치되어 있는 컴퓨터가 AIR 응용 프로그램에 서명하는 데 사용된 인증서를 직접 신뢰하거나 인증서를 신뢰할 수 있는 인증 기관에 연결하는 인증서 체인을 신뢰해야 합니다.

AIR 파일이 신뢰할 수 있는 루트 인증서(일반적으로 여기에는 자체 서명된 모든 인증서가 포함됨) 중 하나에 체인으로 연결되지 않은 인증서로 서명된 경우 제작자 정보를 확인할 수 없습니다. AIR는 AIR 패키지가 서명된 이후 변경되지 않았음을 확인할 수 있지만 해당 파일을 실제로 누가 만들고 서명했는지는 알 수 없습니다.

참고: 사용자는 자체 서명된 인증서를 신뢰하도록 선택할 수 있으며 이 경우 인증서로 서명된 AIR 응용 프로그램에서 인증서의 공용 이름 필드 값이 제작자 이름으로 표시됩니다. AIR에는 사용자가 인증서를 신뢰할 수 있는 것으로 지정할 수 있는 방법이 없습니다. 개인 키가 포함되지 않은 인증서를 사용자에게 개별적으로 제공해야 하며 사용자는 운영 체제에서 제공하는 메커니즘 중 하나나 적절한 도구를 사용하여 인증서를 시스템 인증서 저장소의 적절한 위치로 가져와야 합니다.

AIR 제작자 ID

중요: AIR 1.5.3 현재, 제작자 ID는 사용되지 않으며 더 이상 코드 서명 인증서를 기반으로 계산되지 않습니다. 새로운 응용 프로그램에서는 제작자 ID를 사용할 필요가 없으며 사용해서도 안 됩니다. 기존 응용 프로그램을 업데이트하는 경우 응용 프로그램 설명자 파일에서 원래 제작자 ID를 지정해야 합니다.

AIR 1.5.3 이전에는 AIR 응용 프로그램 설치 프로그램이 AIR 파일 설치 중에 제작자 ID를 생성했습니다. 이 ID는 AIR 파일에 서명하는 데 사용된 인증서에 고유한 ID였습니다. 여러 AIR 응용 프로그램에 대해 동일한 인증서를 다시 사용하는 경우 해당 응용 프로그램은 모두 동일한 제작자 ID를 받았습니다. 다른 인증서로 응용 프로그램 업데이트에 서명하거나 원본 인증서의 인스턴스가 갱신되는 경우에도 제작자 ID가 변경되었습니다.

AIR 1.5.3 이상에서는 AIR에서 제작자 ID를 할당하지 않습니다. AIR 1.5.3을 사용하여 제작된 응용 프로그램은 응용 프로그램 설명자에서 제작자 ID 문자열을 지정할 수 있습니다. 원래 1.5.3 이전 버전의 AIR에 대해 제작된 응용 프로그램의 업데이트를 제작하는 경우에만 제작자 ID를 지정해야 합니다. 응용 프로그램 설명자에서 원래 ID를 지정하지 않으면 새로운 AIR 패키지가 기존 응용 프로그램의 업데이트로 간주되지 않습니다.

원래 제작자 ID를 확인하려면 원래 응용 프로그램이 설치된 META-INF/AIR 하위 디렉토리에서 publisherid 파일을 찾아보십시오. 이 파일 내에 들어 있는 문자열이 제작자 ID입니다. 제작자 ID를 수동으로 지정하려면 응용 프로그램 설명자가 응용 프로그램 설명자 파일의 네임스페이스 선언에서 AIR 1.5.3 런타임 이상을 지정해야 합니다.

제작자 ID(있을 경우)는 다음과 같은 용도로 사용됩니다.

- 암호화된 로컬 저장소를 위한 암호화 키의 일부
- 응용 프로그램 저장소 디렉토리 경로의 일부
- 로컬 연결을 위한 연결 문자열의 일부
- AIR 인 브라우저(in-browser) API를 통해 응용 프로그램을 호출하는 데 사용되는 ID 문자열의 일부
- OSID의 일부(사용자 정의 설치/제거 프로그램을 만들 때 사용됨)

제작자 ID가 변경되면 해당 ID를 사용하는 모든 AIR 기능의 비헤이비어도 변경됩니다. 예를 들어 기존 암호화된 로컬 저장소에 포함된 데이터에 더 이상 액세스할 수 없으며, 응용 프로그램에 대한 로컬 연결을 만드는 모든 Flash 또는 AIR 인스턴스는 연결 문자열에 새로운 ID를 사용해야 합니다. AIR 1.5.3 이상에서는 설치된 응용 프로그램의 제작자 ID를 변경할 수 없습니다. AIR 패키지를 제작할 때 다른 제작자 ID를 사용하면 설치 프로그램이 새 패키지를 업데이트가 아니라 다른 응용 프로그램으로 간주합니다.

인증서 형식

AIR 서명 도구는 JCA(Java Cryptography Architecture)를 통해 액세스할 수 있는 모든 키 저장소를 허용합니다. 여기에는 PKCS12 형식 파일(일반적으로 .pfx 또는 .p12 파일 확장명 사용)과 같은 파일 기반 키 저장소, Java .keystore 파일, PKCS11 하드웨어 키 저장소 및 시스템 키 저장소가 포함됩니다. ADT에서 액세스할 수 있는 키 저장소 형식은 ADT를 실행하는 데 사용된 Java 런타임의 버전과 구성에 따라 달라집니다. PKCS11 하드웨어 토큰과 같은 일부 키 저장소 유형에 액세스하려면 추가 소프트웨어 드라이버와 JCA 플러그인을 설치 및 구성해야 할 수 있습니다.

AIR 파일에 서명하기 위해 대부분의 기존 코드 서명 인증서를 사용하거나 AIR 응용 프로그램에 서명하기 위해 명시적으로 발급된 새 인증서를 얻을 수 있습니다. 예를 들어 다음과 같은 유형의 VeriSign, Thawte, GlobalSign 또는 ChosenSecurity 인증서 중 하나를 사용할 수 있습니다.

- ChosenSecurity:
  - TC Publisher ID for Adobe AIR
- GlobalSign:
  - ObjectSign Code Signing Certificate
- Thawte:
  - AIR Developer Certificate
  - Apple Developer Certificate
  - JavaSoft Developer Certificate
  - Microsoft Authenticode Certificate
- VeriSign:
  - Adobe AIR Digital ID
  - Microsoft Authenticode Digital ID
  - Sun Java Signing Digital ID

참고: 코드 서명용으로 인증서를 만들어야 합니다. SSL 또는 기타 유형의 인증서를 AIR 파일에 서명하는 데 사용할 수 없습니다.

타임스탬프

AIR 파일에 서명하면 패키지 도구가 독립적으로 확인 가능한 서명 날짜와 시간을 가져오기 위해 타임스탬프 기관의 서버에 쿼리를 보냅니다. 가져온 타임스탬프는 AIR 파일에 포함됩니다. 서명 인증서가 서명 당시 유효하기만 하면 인증서가 만료된 후에도 AIR 파일을 설치할 수 있습니다. 반대로 타임스탬프를 가져오지 못하면 인증서가 만료되거나 해지될 경우 AIR 파일을 더 이상 설치할 수 없게 됩니다.

기본적으로 AIR 패키지 도구는 타임스탬프를 가져옵니다. 하지만 타임스탬프 서비스를 사용할 수 없을 때 응용 프로그램을 패키지화할 수 있도록 타임스탬프 기능을 해제할 수 있습니다. 공개적으로 배포되는 모든 AIR 파일에는 타임스탬프가 포함되는 것이 좋습니다.

AIR 패키지 도구에서 사용되는 기본 타임스탬프 기관은 Geotrust입니다.

인증서 얻기

인증서를 얻으려면 일반적으로 인증 기관 웹 사이트를 방문하여 해당 회사의 구매 프로세스를 완료해야 합니다. AIR 도구에 필요한 키 저장소 파일을 생성하는 데 사용되는 도구는 구매한 인증서의 유형, 수신 컴퓨터에서 인증서가 저장되는 방법 및 인증서를 얻는 데 사용한 브라우저(일부 경우)에 따라 달라집니다. 예를 들어 Thawte에서 Adobe Developer Certificate를 얻어 내보내려면 Mozilla Firefox를 사용해야 합니다. 그러면 Firefox 사용자 인터페이스에서 직접 인증서를 .p12 또는 .pfx 파일로 내보낼 수 있습니다.

참고: Java 버전 1.5 이상의 경우 PKCS12 인증서 파일을 보호하는 데 사용되는 암호에 상위 ASCII 문자를 사용할 수 없습니다. Java는 AIR 개발 도구에서 서명된 AIR 패키지를 만드는 데 사용됩니다. 인증서를 .p12 또는 .pfx 파일로 내보낼 경우 암호에 일반 ASCII 문자만 사용하십시오.

AIR 설치 파일을 패키지화하는 데 사용된 ADT(Air Development Tool)를 사용하여 자체 서명된 인증서를 생성할 수 있습니다. 일부 타사 도구도 사용할 수 있습니다.

자체 서명된 인증서를 생성하는 방법 및 AIR 파일 서명 방법에 대한 자세한 내용은 48페이지의 “ADT(AIR Developer Tool)를 사용하여 AIR 설치 파일 패키지”를 참조하십시오. Flash Builder, Dreamweaver 및 Flash용 AIR 업데이트를 사용하여 AIR 파일을 내보내고 서명할 수도 있습니다.

다음 예에서는 Thawte 인증 기관으로부터 AIR Developer Certificate를 얻는 방법과 ADT에 사용하기 위해 해당 인증서를 준비하는 방법을 설명합니다.

예제: Thawte로부터 AIR Developer Certificate 얻기

참고: 이 예는 사용할 코드 서명 인증서를 얻고 준비하는 많은 방법 중 하나를 보여 줄 뿐입니다. 인증 기관별로 자체적인 정책과 절차가 있습니다.

Thawte 웹 사이트에서 AIR Developer Certificate를 구매하려면 Mozilla Firefox 브라우저를 사용해야 합니다. 인증서의 개인 키는 브라우저의 키 저장소에 저장됩니다. Firefox 키 저장소가 마스터 암호로 보호되는지 그리고 컴퓨터 자체가 물리적으로 안전한지 확인하십시오. 구매 프로세스가 완료되면 브라우저 키 저장소에서 인증서와 개인 키를 내보내고 제거할 수 있습니다.

인증서 등록 프로세스의 일부로 개인/공개 키 쌍이 생성됩니다. 개인 키는 Firefox 키 저장소 내에 자동으로 저장됩니다. Thawte 웹 사이트에서 인증서를 요청하거나 검색할 때 항상 같은 컴퓨터와 브라우저를 사용해야 합니다.

1 Thawte 웹 사이트를 방문하여 Product 페이지의 Code Signing Certificates로 이동합니다.

2 Code Signing Certificates 목록에서 [Adobe AIR Developer Certificate]를 선택합니다.

3 세 단계로 구성된 등록 프로세스를 완료합니다. 조직 및 연락처 정보를 제공해야 합니다. 그러면 Thawte에서 ID 확인 프로세스를 수행하고 추가 정보를 요청할 수 있습니다. 확인이 완료되면 Thawte에서 인증서를 검색하는 방법에 대한 지침을 전자 메일로 보냅니다.

   참고: 필요한 설명서 유형에 대한 자세한 내용은 https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf를 참조하십시오.

4 Thawte 사이트에서 발급된 인증서를 검색합니다. 인증서는 자동으로 Firefox 키 저장소에 저장됩니다.

5 다음 단계를 사용하여 Firefox 키 저장소에서 개인 키 및 인증서가 포함된 키 저장소 파일을 내보냅니다.

참고: Firefox에서 개인 키/인증서를 내보낼 때 해당 파일은 ADT, Flex, Flash 및 Dreamweaver에서 사용할 수 있는 .p12(pfx) 형식으로 내보내집니다.

a [Firefox 인증서 관리자] 대화 상자를 엽니다.

b Windows의 경우: [도구] -> [옵션] -> [고급] -> [암호화] -> [인증서 보기]를 엽니다.

c Mac OS의 경우: [Firefox] -> [환경 설정] -> [고급] -> [암호화] -> [인증서 보기]를 엽니다.

d Linux의 경우: [편집] -> [환경 설정] -> [고급] -> [암호화] -> [인증서 보기]를 엽니다.

e 인증서 목록에서 [AIR Code Signing Certificate]를 선택하고 **[백업]** 버튼을 클릭합니다.

f 파일 이름과 키 저장소 파일을 내보낼 위치를 입력하고 **저장**을 클릭합니다.

g Firefox 마스터 암호를 사용 중인 경우 파일을 내보내려면 소프트웨어 보안 장치의 암호를 입력하라는 메시지가 표시됩니다. 이 암호는 Firefox에서만 사용됩니다.

h **인증서 백업 암호 선택** 대화 상자에서 키 저장소 파일의 암호를 만듭니다.

중요: 이 암호는 키 저장소 파일을 보호하며 AIR 응용 프로그램 서명에 파일을 사용할 때 필요합니다. 보안 암호를 선택해야 합니다.

i [확인]을 클릭합니다. 백업 암호가 만들어졌다는 메시지가 표시됩니다. 개인 키와 인증서가 포함된 키 저장소 파일은 .p12 파일 확장명(PKCS12 형식)으로 저장됩니다.

6 내보낸 키 저장소 파일을 ADT, Flash Builder, Flash Professional 또는 Dreamweaver에서 사용합니다. AIR 응용 프로그램에 서명할 때마다 파일에 대해 생성된 암호가 필요합니다.

중요: 개인 키와 인증서는 계속 Firefox 키 저장소 내에 저장되어 있습니다. 이로 인해 인증서 파일의 추가 복사본을 내보낼 수 있지만 인증서와 개인 키의 보안을 유지하기 위해 보호해야 하는 다른 액세스 지점도 노출됩니다.

인증서 변경

일부 경우, AIR 응용 프로그램의 업데이트에 서명하는 데 사용하는 인증서를 변경해야 합니다. 이러한 경우는 다음과 같습니다.

- 원본 서명 인증서를 갱신하는 경우
- 자체 서명된 인증서에서 인증 기관에서 발급한 인증서로 업그레이드하는 경우
- 만료 예정인 자체 서명된 인증서에서 다른 인증서로 변경하는 경우
- CI(Corporate Identity)가 변경되는 등의 이유로 인해 한 상업용 인증서에서 다른 상업용 인증서로 변경하는 경우

AIR에서 AIR 파일을 업데이트로 인식하도록 하려면 원본 AIR 파일과 업데이트 AIR 파일에 동일한 인증서로 서명하거나 업데이트에 인증서 마이그레이션 서명을 적용해야 합니다. 마이그레이션 서명은 원본 인증서를 사용하여 업데이트 AIR 패키지에 적용되는 2차 서명입니다. 마이그레이션 서명은 원본 인증서를 사용하여 서명자가 응용 프로그램의 원래 제작자임을 입증합니다.

마이그레이션 서명이 적용된 AIR 파일이 설치된 후에는 새 인증서가 기본 인증서가 됩니다. 이후 업데이트에는 마이그레이션 서명이 필요하지 않습니다. 그러나 업데이트를 건너뛰는 사용자도 수용할 수 있도록 가능한 한 오랫동안 마이그레이션 서명을 적용해야 합니다.

중요: 원본 인증서가 만료되기 전에 인증서를 변경해야 합니다. 인증서가 만료되기 전에 마이그레이션 서명으로 서명된 업데이트를 만들지 않으면 기존 버전의 응용 프로그램을 제거하고 새 버전을 설치해야 합니다. AIR 1.5.3 현재, 인증서가 만료된 후 180일의 유예 기간 안에 만료된 인증서를 사용하여 마이그레이션 서명을 적용할 수 있습니다. 만료된 인증서를 사용하여 기본 응용 프로그램 서명을 적용할 수는 없습니다.

인증서를 변경하려면

1 응용 프로그램의 업데이트를 만듭니다.

2 **새** 인증서를 사용하여 업데이트 AIR 파일을 패키지화하고 서명합니다.

3 **원본** 인증서(ADT -migrate 명령 사용)를 사용하여 다시 AIR 파일에 서명합니다.

다른 측면에서 보면 마이그레이션 서명이 있는 AIR 파일은 정상 AIR 파일입니다. 원본 버전이 없는 시스템에 응용 프로그램을 설치한 경우 AIR는 일반적인 방식으로 새 버전을 설치합니다.

참고: AIR 1.5.3 이전에는 갱신된 인증서로 AIR 응용 프로그램에 서명할 때 마이그레이션 서명이 필요하지 않은 경우도 있었습니다. AIR 1.5.3 버전부터는 갱신된 인증서를 사용할 경우 항상 마이그레이션 서명이 필요합니다.

마이그레이션 서명을 적용하는 절차는 60페이지의 “AIR 파일에 서명하여 응용 프로그램 인증서 변경”에서 설명합니다.

응용 프로그램 ID 변경

AIR 1.5.3 이전에는 마이그레이션 서명으로 서명된 업데이트가 설치될 경우 AIR 응용 프로그램의 ID가 변경되었습니다. 응용 프로그램의 ID를 변경하면 다음과 같은 몇 가지 부정적인 영향을 줍니다.

- 새 응용 프로그램 버전이 암호화된 기존 로컬 저장소의 데이터에 액세스할 수 없습니다.
- 응용 프로그램 저장소 디렉토리의 위치가 변경됩니다. 이전 위치의 데이터가 새 디렉토리로 복사되지 않습니다. 하지만 새 응용 프로그램은 이전 제작자 ID를 기반으로 원본 디렉토리를 찾을 수 있습니다.
- 응용 프로그램이 이전 제작자 ID를 사용하여 로컬 연결을 더 이상 열 수 없습니다.
- 웹 페이지에서 응용 프로그램에 액세스하는 데 사용되는 ID 문자열이 변경됩니다.
- 응용 프로그램의 OSID가 변경됩니다. OSID는 사용자 정의 설치/제거 프로그램을 작성할 때 사용됩니다.

AIR 1.5.3을 사용하여 업데이트를 제작하는 경우에는 응용 프로그램 ID를 변경할 수 없습니다. 업데이트 AIR 파일의 응용 프로그램 설명자에 원래 응용 프로그램 및 제작자 ID가 지정되어야 합니다. 그렇지 않으면 새 패키지가 업데이트로 인식되지 않습니다.

참고: AIR 1.5.3 이상을 사용하여 새 AIR 응용 프로그램을 제작하는 경우에는 제작자 ID를 지정하면 안 됩니다.

만료된 인증서

AIR 1.5.3 현재, 만료된 후 180일이 지나지 않은 인증서는 응용 프로그램 업데이트에 마이그레이션 서명을 적용하는 데 계속 사용할 수 있습니다. 이러한 유예 기간을 활용하려면 업데이트 응용 프로그램 설명자가 네임스페이스 특성에서 1.5.3을 지정해야 합니다. 유예 기간이 지난 후에는 더 이상 인증서를 사용할 수 없습니다. 새 버전의 응용 프로그램으로 업데이트하는 사용자는 기존 버전을 제거해야 합니다. 1.5.3 이전 버전의 AIR에는 유예 기간이 없습니다.

용어

이 단원에서는 공개 배포를 위해 응용 프로그램에 서명하는 방법을 결정할 때 이해해야 할 핵심 용어 중 몇 가지에 대한 용어 사전을 제공합니다.

용어	설명
CA(인증 기관)	신뢰할 수 있는 타사 역할을 하며 궁극적으로 공개 키 소유자의 ID를 인증하는 공개 키 인프라 네트워크의 주체입니다. CA는 일반적으로 자체 개인 키로 서명된 디지털 인증서를 발급하여 인증서 소유자의 ID를 확인했음을 증명합니다.
CPS(Certificate Practice Statement)	인증서를 발급하고 확인하는 인증 기관의 사용 약관과 정책을 설명합니다. CPS는 CA와 해당 구독자 및 관련 당사자 간 계약의 일부입니다. 또한 ID 확인에 대한 정책과 해당 기관에서 제공하는 인증서의 보증 수준에 대해 간략하게 설명합니다.
CRL(인증서 해지 목록)	이미 해지되었으므로 더 이상 신뢰할 수 없게 된 인증서 목록입니다. AIR는 AIR 응용 프로그램이 서명될 때 CRL을 확인하고 타임스탬프가 없으면 응용 프로그램을 설치할 때 다시 CRL을 확인합니다.
인증서 체인	인증서 체인은 체인의 각 인증서가 다음 인증서에 의해 서명된 인증서의 시퀀스입니다.

용어	설명
디지털 인증서	소유자의 ID, 소유자의 공개 키 및 인증서 자체의 ID에 대한 정보가 포함된 디지털 문서입니다. 인증 기관에서 발급한 인증서 자체는 발급 주체 CA에 속하는 인증서에 의해 서명됩니다.
디지털 서명	공개-개인 키 쌍의 절반인 공개 키로만 해독할 수 있는 암호화된 메시지 또는 다이제스트입니다. PKI에서 디지털 서명에는 결국은 인증 기관에서 추적할 수 있는 하나 이상의 디지털 인증서가 포함됩니다. 메시지 또는 컴퓨터 파일에 서명한 후 변경되지 않았음(사용된 암호화 알고리즘에서 제공하는 보증 제한 내에서)을 검증하는 데 사용될 수 있습니다. 이 경우 발급 주체 인증 기관, 서명자의 ID를 신뢰한다고 가정합니다.
키 저장소	디지털 인증과 경우에 따라서는 관련 개인 키를 포함하는 데이터베이스입니다.
JCA(Java Cryptography Architecture)	키 저장소를 관리하고 액세스하는 확장 가능한 아키텍처입니다. 자세한 내용은 Java Cryptography Architecture Reference Guide를 참조하십시오.
PKCS #11	RSA Laboratories의 암호화 토큰 인터페이스 표준입니다. 하드웨어 토큰 기반 키 저장소입니다.
PKCS #12	RSA Laboratories의 개인 정보 교환 구문 표준입니다. 일반적으로 개인 키 및 연결된 디지털 인증서가 포함된 파일 기반 키 저장소입니다.
개인 키	두 부분으로 구성된 공개-개인 키 비대칭 암호화 시스템 중 절반인 개인 부분입니다. 개인 키는 비밀로 보관되어야 하며 네트워크를 통해 전송하면 안 됩니다. 디지털로 서명된 메시지는 서명자에 의해 개인 키로 암호화됩니다.
공개 키	두 부분으로 구성된 공개-개인 키 비대칭 암호화 시스템 중 절반인 공개 부분입니다. 공개 키는 공개적으로 사용할 수 있으며 개인 키로 암호화된 메시지를 해독하는 데 사용됩니다.
PKI(공개 키 인프라)	인증 기관이 공개 키 소유주의 ID에 증명하는 신뢰 시스템입니다. 네트워크의 클라이언트는 디지털 메시지 또는 파일 서명자의 ID를 확인하기 위해 신뢰할 수 있는 CA에서 발급한 디지털 인증서에 의존합니다.
타임스탬프	이벤트가 발생한 날짜 및 시간이 포함된 디지털로 서명된 데이터입니다. ADT는 AIR 패키지의 RFC 3161 호환 시간 서버에서 타임스탬프를 포함할 수 있습니다. 타임스탬프가 있는 경우 AIR는 타임스탬프를 사용하여 서명 시간의 인증서 유효성을 설정합니다. 이를 통해 AIR 응용 프로그램의 서명 인증서가 만료된 이후 해당 응용 프로그램이 설치될 수 있습니다.
타임스탬프 기관	타임스탬프를 발급하는 기관입니다. AIR가 인식할 수 있도록 타임스탬프는 RFC 3161을 따라야 하고 타임스탬프 서명은 설치 컴퓨터에서 신뢰할 수 있는 인증 기관에 연결되어야 합니다.

14장: AIR 응용 프로그램 업데이트

사용자는 컴퓨터 또는 브라우저(간편한 설치 기능 사용)에서 AIR 파일을 두 번 클릭하여 AIR 응용 프로그램을 설치하거나 업데이트할 수 있습니다. Adobe® AIR® 설치 응용 프로그램은 설치를 관리하고 사용자가 기존 응용 프로그램을 업데이트하려고 할 경우 경고를 표시합니다. 자세한 내용은 75페이지의 "AIR 응용 프로그램 배포, 설치 및 실행"을 참조하십시오.

그러나 Updater 클래스를 사용하여 설치된 응용 프로그램이 새 버전으로 업데이트되게 할 수도 있습니다. 설치된 응용 프로그램은 새 버전을 다운로드하고 설치할 수 있음을 감지할 수 있습니다. Updater 클래스에는 사용자 컴퓨터의 AIR 파일을 가리키고 해당 버전으로 업데이트할 수 있게 하는 update() 메서드가 포함되어 있습니다.

업데이트 AIR 파일의 응용 프로그램 ID와 제작자 ID가 모두 일치해야 응용 프로그램이 업데이트됩니다. 제작자 ID는 서명 인증서에서 파생됩니다. 업데이트와 업데이트할 응용 프로그램이 모두 동일 인증서로 서명되어야 합니다.

AIR 1.5.3 이상의 경우 응용 프로그램 설명자 파일에는 <publisherID> 요소가 포함됩니다. AIR 1.5.2 이하를 사용하여 개발된 응용 프로그램 버전이 있으면 이 요소를 사용해야 합니다. 자세한 내용은 64페이지의 "응용 프로그램 ID 정의"를 참조하십시오.

AIR 1.1 이상부터 응용 프로그램을 마이그레이션하여 새 코드 서명 인증서를 사용할 수 있습니다. 응용 프로그램을 마이그레이션하여 새 서명을 사용하려면 업데이트 AIR 파일을 새 인증서와 원래 인증서를 둘 다 사용하여 서명해야 합니다. 인증서 마이그레이션은 단방향 프로세스입니다. 마이그레이션 후에는 새 인증서(또는 두 인증서)로 서명된 AIR 파일만 기존 설치에 대한 업데이트로 인식됩니다.

응용 프로그램의 업데이트 관리는 복잡할 수 있습니다. AIR 1.5에는 **AdobeAIR 응용 프로그램에 대한 새로운업데이트 프레임워크**가 포함됩니다. 이 프레임워크에서는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 API를 제공합니다.

인증서 마이그레이션을 사용하여 자체 서명된 인증서를 상용 코드 서명 인증서로 변경하거나 자체 서명된 인증서 또는 상용 인증서를 다른 자체 서명된 인증서 또는 상용 인증서로 변경할 수 있습니다. 인증서를 마이그레이션하지 않으면 새 버전을 설치하기 전에 기존 사용자가 현재 응용 프로그램 버전을 제거해야 합니다. 자세한 내용은 88페이지의 "인증서 변경"을 참조하십시오.

응용 프로그램에는 업데이트 메커니즘을 포함하는 것이 좋습니다. 응용 프로그램의 새 버전을 만들 경우 업데이트 메커니즘을 사용하면 사용자에게 새 버전을 설치하도록 알릴 수 있습니다.

AIR 응용 프로그램 설치 프로그램은 AIR 응용 프로그램이 설치, 업데이트 또는 제거될 때 로그 파일을 생성합니다. 이러한 로그를 참조하여 설치 문제의 원인을 확인할 수 있습니다. 83페이지의 "설치 로그"를 참조하십시오.

참고: 새 버전의 Adobe AIR 런타임에는 업데이트된 WebKit 버전이 포함될 수 있습니다. 업데이트된 WebKit 버전은 배포된 AIR 응용 프로그램의 HTML 내용을 예상치 않게 변경시킬 수 있습니다. 이러한 변경을 위해서는 응용 프로그램을 업데이트해야 할 수 있습니다. 업데이트 메커니즘을 사용하면 응용 프로그램의 새 버전을 사용자에게 알릴 수 있습니다. 자세한 내용은 HTML 환경(ActionScript 개발자용) 또는 About the HTML environment(HTML 개발자용)을 참조하십시오.

응용 프로그램 업데이트

Updater 클래스(flash.desktop 패키지에 포함됨)에 있는 update() 메서드는 현재 실행되는 응용 프로그램을 다른 버전으로 업데이트하는 데 사용할 수 있습니다. 예를 들어, 데스크톱에 특정 버전의 AIR 파일("Sample_App_v2.air")이 있는 경우 다음 코드는 응용 프로그램을 업데이트합니다.

ActionScript 예제:

```
var updater:Updater = new Updater();
var airFile:File = File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version:String = "2.01";
updater.update(airFile, version);
```

JavaScript 예제:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

응용 프로그램에서 Updater 클래스를 사용하기 전에 사용자나 응용 프로그램은 업데이트된 버전의 AIR 파일을 컴퓨터에 다운로드해야 합니다. 자세한 내용은 93페이지의 “사용자 컴퓨터에 AIR 파일 다운로드”를 참조하십시오.

Updater.update() 메서드 호출의 결과

런타임의 응용 프로그램에서 update() 메서드를 호출하면 런타임은 응용 프로그램을 닫은 다음 AIR 파일에서 새 버전을 설치하려고 합니다. 런타임은 AIR 파일에 지정된 응용 프로그램 ID 및 제작자 ID가 update() 메서드를 호출하는 응용 프로그램의 응용 프로그램 ID 및 제작자 ID와 일치하는지 확인합니다. 응용 프로그램 ID 및 제작자 ID에 대한 자세한 내용은 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오. 또한 버전 문자열이 update() 메서드에 전달된 version 문자열과 일치하는지도 확인합니다. 런타임은 설치가 성공적으로 완료되면 새 버전의 응용 프로그램을 열고, 그렇지 않으면(설치가 완료될 수 없으면) 기존(설치 전) 버전의 응용 프로그램을 다시 엽니다.

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.

업데이트된 버전의 응용 프로그램에 업데이트된 버전의 런타임이 필요한 경우에는 새 런타임 버전이 설치됩니다. 런타임을 업데이트하려면 사용자에게 컴퓨터에 대한 관리 권한이 필요합니다.

ADL을 사용하여 응용 프로그램을 테스트할 때 update() 메서드를 호출하면 런타임 예외가 발생합니다.

버전 문자열

update() 메서드의 version 매개 변수로 지정된 문자열이 응용 프로그램 설명자 파일의 기본 application 요소에 대한 version 특성에 있는 문자열과 일치해야 AIR 파일이 설치됩니다. version 매개 변수를 지정하는 것은 보안상의 이유로 필요합니다. 응용 프로그램에서 AIR 파일의 버전 번호를 확인하도록 설정하면 응용 프로그램에서 의도하지 않게 이전 버전을 설치하는 일은 없어집니다. 이전 버전의 응용 프로그램에는 현재 설치된 응용 프로그램에서 수정된 보안 취약점이 있을 수 있습니다. 또한 응용 프로그램에서는 AIR 파일의 버전 문자열을 설치된 응용 프로그램의 버전 문자열과 비교하여 다운그레이드 공격을 방지해야 합니다.

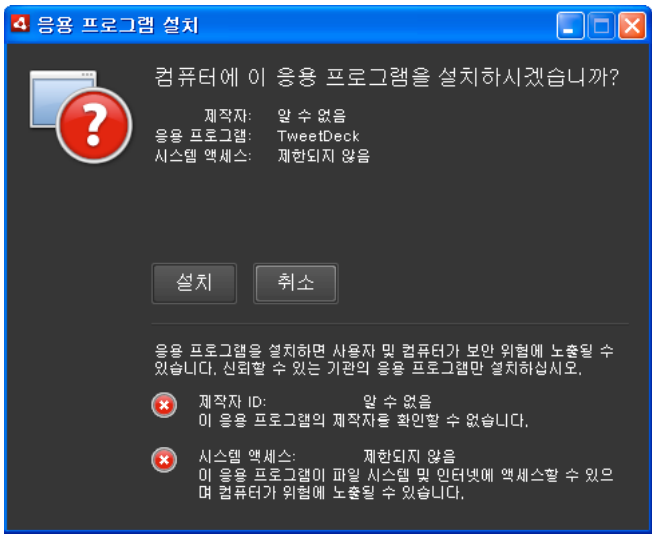
버전 문자열은 임의의 형식일 수 있습니다. 예를 들어, "2.01" 또는 "version 2"일 수 있습니다. 이 문자열의 형식은 응용 프로그램 개발자가 디코딩해야 합니다. 런타임은 버전 문자열의 유효성을 검사하지 않으며 응용 프로그램 코드에서 응용 프로그램을 업데이트하기 전에 이 문자열의 유효성을 검사해야 합니다.

Adobe AIR 응용 프로그램에서 AIR 파일을 웹을 통해 다운로드하는 경우 웹 서비스에서 Adobe AIR 응용 프로그램에 다운로드할 버전을 알릴 수 있는 메커니즘을 마련하는 것이 좋습니다. 이렇게 하면 응용 프로그램에서 이 문자열을 update() 메서드의 version 매개 변수로 사용할 수 있습니다. AIR 파일의 버전을 알 수 없는 다른 방법으로 AIR 파일을 얻은 경우 AIR 응용 프로그램에서 AIR 파일을 검사하여 버전 정보를 확인할 수 있습니다. AIR 파일은 ZIP 압축 파일이고 응용 프로그램 설명자 파일은 압축 파일의 두 번째 레코드입니다.

응용 프로그램 설명자 파일에 대한 자세한 내용은 62페이지의 “AIR 응용 프로그램 속성 설정”을 참조하십시오.

사용자 정의 응용 프로그램 업데이트 사용자 인터페이스 제공

AIR에는 기본 업데이트 인터페이스가 포함되어 있습니다.



이 인터페이스는 사용자가 컴퓨터에 응용 프로그램의 버전을 처음 설치할 때 항상 사용됩니다. 그러나 이후 인스턴스의 경우 사용자가 고유의 인터페이스를 정의할 수 있습니다. 응용 프로그램에서 사용자 정의 업데이트 인터페이스를 정의하는 경우 현재 설치된 응용 프로그램에 대한 응용 프로그램 설명자 파일에서 customUpdateUI 요소를 지정합니다.

```
<customUpdateUI>true</customUpdateUI>
```

응용 프로그램이 설치되고 사용자가 설치된 응용 프로그램과 일치하는 응용 프로그램 ID 및 제작자 ID가 포함된 AIR 파일을 열면 런타임은 기본 AIR 응용 프로그램 설치 프로그램이 아니라 응용 프로그램을 엽니다. 자세한 내용은 70페이지의 “응용 프로그램 업데이트를 위한 사용자 정의 사용자 인터페이스 제공”을 참조하십시오.

응용 프로그램에서 실행된 시기(NativeApplication.nativeApplication 객체가 load 이벤트를 전달한 시기)와 응용 프로그램을 업데이트할지 여부(Updater 클래스 사용)를 확인할 수 있습니다. 응용 프로그램에서 업데이트하도록 결정하는 경우 표준 실행 인터페이스와 다른 고유의 설치 인터페이스를 사용자에게 제공할 수 있습니다.

사용자 컴퓨터에 AIR 파일 다운로드

Updater 클래스를 사용하려면 먼저 사용자나 응용 프로그램이 AIR 파일을 사용자 컴퓨터에 로컬로 저장해야 합니다.

참고: AIR 1.5에는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 업데이트 프레임워크가 포함되어 있습니다. Update 클래스의 update() 메서드를 직접 사용하는 것보다 이 프레임워크를 사용하는 것이 훨씬 쉽습니다. 자세한 내용은 96페이지의 “업데이트 프레임워크 사용”을 참조하십시오.

다음 코드는 AIR 파일을 URL(http://example.com/air/updates/Sample_App_v2.air)에서 읽고 AIR 파일을 응용 프로그램 저장소 디렉토리에 저장합니다.

ActionScript 예제:

```
var urlString:String = "http://example.com/air/updates/Sample_App_v2.air";
var urlReq:URLRequest = new URLRequest(urlString);
var urlStream:URLStream = new URLStream();
var fileData:ByteArray = new ByteArray();
urlStream.addEventListener(Event.COMPLETE, loaded);
urlStream.load(urlReq);

function loaded(event:Event):void {
    urlStream.readBytes(fileData, 0, urlStream.bytesAvailable);
    writeAirFile();
}

function writeAirFile():void {
    var file:File = File.applicationStorageDirectory.resolvePath("My App v2.air");
    var fileStream:FileStream = new FileStream();
    fileStream.open(file, FileMode.WRITE);
    fileStream.writeBytes(fileData, 0, fileData.length);
    fileStream.close();
    trace("The AIR file is written.");
}
```

JavaScript 예제:

```
var urlString = "http://example.com/air/updates/Sample_App_v2.air";
var urlReq = new air.URLRequest(urlString);
var urlStream = new air.URLStream();
var fileData = new air.ByteArray();
urlStream.addEventListener(air.Event.COMPLETE, loaded);
urlStream.load(urlReq);

function loaded(event) {
    urlStream.readBytes(fileData, 0, urlStream.bytesAvailable);
    writeAirFile();
}

function writeAirFile() {
    var file = air.File.desktopDirectory.resolvePath("My App v2.air");
    var fileStream = new air.FileStream();
    fileStream.open(file, air.FileMode.WRITE);
    fileStream.writeBytes(fileData, 0, fileData.length);
    fileStream.close();
    trace("The AIR file is written.");
}
```

자세한 내용은 다음 항목을 참조하십시오.

- 파일 읽기 및 쓰기 작업 과정(ActionScript 개발자용)
- Workflow for reading and writing files(HTML 개발자용)

응용 프로그램이 처음 실행되는지 확인

응용 프로그램을 업데이트한 후 사용자에게 "시작" 또는 "환영" 메시지를 제공할 수 있습니다. 응용 프로그램은 시작될 때 처음 실행되는지 여부를 확인하므로 이러한 메시지를 표시할지 여부를 결정할 수 있습니다.

참고: AIR 1.5에는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 업데이트 프레임워크가 포함되어 있습니다. 이 프레임워크에서는 응용 프로그램 버전이 처음으로 실행되는지 여부를 확인하는 간편한 방법을 제공합니다. 자세한 내용은 96페이지의 “업데이트 프레임워크 사용”을 참조하십시오.

이렇게 하는 한 가지 방법은 응용 프로그램을 초기화할 때 응용 프로그램 저장소 디렉토리에 파일을 저장하는 것입니다. 응용 프로그램은 시작될 때마다 해당 파일이 있는지 확인해야 합니다. 파일이 없으면 현재 사용자의 경우 응용 프로그램이 처음 실행되는 것이고, 파일이 있으면 응용 프로그램이 이미 한 번 이상 실행된 것입니다. 파일이 있고 현재 버전 번호보다 이전의 버전 번호를 포함하는 경우 사용자가 새 버전을 처음 실행하는 것입니다.

다음 Flex 예제에서는 이 개념을 보여 줍니다.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    title="Sample Version Checker Application"
    applicationComplete="system extension()">
    <mx:Script>
        <![CDATA[
            import flash.filesystem.*;
            public var file:File;
            public var currentVersion:String = "1.2";
            public function system extension():void {
                file = File.applicationStorageDirectory;
                file = file.resolvePath("Preferences/version.txt");
                trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            private function checkVersion():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.READ);
                var reversion:String = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    log.text = "You have updated to version " + currentVersion + ".\n";
                } else {
                    saveFile();
                }
                log.text += "Welcome to the application.";
            }
            private function firstRun():void {
                log.text = "Thank you for installing the application. \n"
                    + "This is the first time you have run it.";
                saveFile();
            }
            private function saveFile():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        ]]>
    </mx:Script>
    <mx:TextArea ID="log" width="100%" height="100%" />
</mx:WindowedApplication>
```

다음 예제에서는 JavaScript에서의 이 개념을 보여 줍니다.

```
<html>
    <head>
        <script src="AIRAliases.js" />
        <script>
            var file;
            var currentVersion = "1.2";
            function system extension() {
                file = air.File.appStorageDirectory.resolvePath("Preferences/version.txt");
                air.trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            function checkVersion() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.READ);
                var reversion = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    window.document.getElementById("log").innerHTML
                            = "You have updated to version " + currentVersion + ".\n";
                } else {
                    saveFile();
                }
                window.document.getElementById("log").innerHTML
                                += "Welcome to the application.";
            }
            function firstRun() {
                window.document.getElementById("log").innerHTML
                            = "Thank you for installing the application. \n"
                            + "This is the first time you have run it.";
                saveFile();
            }
            function saveFile() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        </script>
    </head>
    <body onLoad="system extension()">
        <textarea ID="log" rows="100%" cols="100%" />
    </body>
</html>
```

응용 프로그램에서 데이터를 응용 프로그램 저장소 디렉토리 등에 로컬로 저장하는 경우 처음 실행될 때 이전 버전에서 이전에 저장된 데이터를 확인할 수 있습니다.

업데이트 프레임워크 사용

응용 프로그램의 업데이트 관리는 복잡할 수 있습니다. **AdobeAIR** 응용 프로그램의 업데이트 프레임워크에서는 개발자가 AIR 응용 프로그램에서 적절한 업데이트 기능을 제공하는 데 도움이 되는 API를 제공합니다. AIR 업데이트 프레임워크의 기능은 개발자의 다음 작업을 지원합니다.

- 일정 간격에 따라 또는 사용자 요청 시 업데이트가 있는지 정기적으로 확인

- 웹 소스에서 AIR 파일(업데이트) 다운로드
- 새로 설치된 버전의 첫 실행을 사용자에게 알림
- 사용자에게 업데이트를 확인할지 묻기
- 새 업데이트 버전에 대한 정보를 사용자에게 표시
- 다운로드 진행률 및 오류 정보를 사용자에게 표시

AIR 업데이트 프레임워크에서는 응용 프로그램에서 사용할 수 있는 샘플 사용자 인터페이스를 제공하며, 응용 프로그램 업데이트와 관련된 기본 정보와 옵션을 제공합니다. 응용 프로그램에서 업데이트 프레임워크에 사용할 고유한 사용자 정의 사용자 인터페이스를 정의할 수도 있습니다.

AIR 업데이트 프레임워크에서는 AIR 응용 프로그램의 업데이트 버전에 대한 정보를 간단한 XML 구성 파일에 저장할 수 있습니다. 대부분의 응용 프로그램에서 이러한 구성 파일을 설정하고 일부 기본적인 코드를 포함하면 최종 사용자에게 적절한 업데이트 기능이 제공됩니다.

Adobe AIR에는 업데이트 프레임워크 외에도 AIR 응용 프로그램이 새 버전으로 업그레이드하는 데 사용할 수 있는 Updater 클래스가 포함되어 있습니다. 이 클래스를 사용하여 응용 프로그램을 사용자 컴퓨터의 AIR 파일에 포함된 버전으로 업그레이드할 수 있습니다. 그러나 업그레이드 관리에는 단순히 응용 프로그램이 로컬에 저장한 AIR 파일을 기반으로 업데이트되는 것 이상의 작업이 포함될 수 있습니다.

AIR 업데이트 프레임워크 파일

AIR 업데이트 프레임워크는 AIR 2 SDK의 frameworks/libs/air 디렉토리에 있으며 다음 파일을 포함합니다.

- applicationupdater.swc - ActionScript에서 사용되는 업데이트 라이브러리의 기본적인 기능을 정의합니다. 이 버전은 사용자 인터페이스를 포함하지 않습니다.
- applicationupdater.swf - JavaScript에서 사용되는 업데이트 라이브러리의 기본적인 기능을 정의합니다. 이 버전은 사용자 인터페이스를 포함하지 않습니다.
- applicationupdater_ui.swc - 응용 프로그램이 업데이트 옵션을 표시하는 데 사용할 수 있는 사용자 인터페이스를 포함하여 Flex 4 버전 업데이트 라이브러리의 기본적인 기능을 정의합니다.
- applicationupdater_ui.swf - 응용 프로그램이 업데이트 옵션을 표시하는 데 사용할 수 있는 사용자 인터페이스를 포함하여 JavaScript 버전 업데이트 라이브러리의 기본적인 기능을 정의합니다.
- flex3/applicationupdater_ui.swc - Flex 3과 함께 사용되는 applicationupdater_ui 파일의 버전입니다.

  중요: AIR 2 SDK를 Flex 3과 함께 사용할 경우 flex3 하위 디렉토리에 있는 applicationupdater_ui.swc 파일의 버전을 참조해야 합니다. 또는 frameworks/libs/air 디렉토리에 있는 버전을 flex3 하위 디렉토리에 있는 버전으로 바꿀 수도 있습니다.

자세한 내용은 다음 단원을 참조하십시오.

- 97페이지의 “Flex 개발 환경 설정”
- 98페이지의 “HTML 기반 AIR 응용 프로그램에 프레임워크 파일 포함”
- 98페이지의 “기본 예제: ApplicationUpdaterUI 버전 사용”

Flex 개발 환경 설정

AIR 2 SDK의 frameworks/libs/air 디렉토리에 있는 SWC 파일은 Flex 및 Flash 개발에 사용할 수 있는 클래스를 정의합니다.

Flex SDK를 사용하여 컴파일할 때 업데이트 프레임워크를 사용하려면 amxmlc 컴파일러에 대한 호출에 ApplicationUpdater.swc 또는 ApplicationUpdater_UI.swc 파일을 포함합니다. 다음 예제에서는 컴파일러가 Flex SDK 디렉토리의 lib 하위 디렉토리에 있는 ApplicationUpdater.swc 파일을 로드합니다.

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

다음 예제에서는 컴파일러가 Flex SDK 디렉토리의 lib 하위 디렉토리에 있는 ApplicationUpdater_UI.swc 파일을 로드합니다.

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Flash Builder를 사용하여 개발할 경우 [Properties] 대화 상자에서 [Flex Build Path settings]의 [Library Path] 탭에 SWC 파일을 추가합니다.

SWC 파일을 amxmlc 컴파일러(Flex SDK 사용) 또는 Flash Builder에서 참조할 디렉토리로 복사해야 합니다.

HTML 기반 AIR 응용 프로그램에 프레임워크 파일 포함

업데이트 프레임워크의 frameworks/html 디렉토리에는 다음 SWF 파일이 포함되어 있습니다.

- ApplicationUpdater.swf - 사용자 인터페이스 없이 업데이트 라이브러리의 기본 기능을 정의합니다.
- ApplicationUpdater_UI.swf - 응용 프로그램이 업데이트 옵션을 표시하는 데 사용할 수 있는 사용자 인터페이스를 포함하여 업데이트 라이브러리의 기본 기능을 정의합니다.

AIR 응용 프로그램의 JavaScript 코드에서는 SWF 파일에 정의된 클래스를 사용할 수 있습니다.

업데이트 프레임워크를 사용하려면 ApplicationUpdater.swf 또는 ApplicationUpdater_UI.swf 파일을 응용 프로그램 디렉토리 또는 하위 디렉토리에 포함합니다. 그런 다음 JavaScript 코드에서 프레임워크를 사용할 HTML 파일에 파일을 로드하는 script 태그를 포함합니다.

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

또는 이 script 태그를 사용하여 ApplicationUpdater_UI.swf 파일을 로드합니다.

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

이러한 두 파일에서 정의되는 API는 이 문서의 나머지 부분에서 설명합니다.

기본 예제: ApplicationUpdaterUI 버전 사용

업데이트 프레임워크의 ApplicationUpdaterUI 버전은 응용 프로그램에서 쉽게 사용할 수 있는 기본 인터페이스를 제공합니다. 다음은 기본 예제입니다.

먼저 업데이트 프레임워크를 호출하는 AIR 응용 프로그램을 만듭니다.

1 응용 프로그램이 HTML 기반 AIR 응용 프로그램인 경우 ApplicationUpdaterUI.js 파일을 로드합니다.

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 AIR 응용 프로그램 논리에서 ApplicationUpdaterUI 객체를 인스턴스화합니다.

ActionScript에서는 다음 코드를 사용합니다.

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript에서는 다음 코드를 사용합니다.

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

응용 프로그램이 로드될 때 실행되는 초기화 기능에 이 코드를 추가할 수 있습니다.

3 updateConfig.xml이라는 텍스트 파일을 만들어 다음 내용을 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
    <url>http://example.com/updates/update.xml</url>
    <delay>1</delay>
</configuration>
```

updateConfig.xml 파일의 URL 요소가 웹 서버에 있는 업데이트 설명자 파일의 최종 위치와 일치하도록 편집합니다(다음 절차 참조).

delay는 응용 프로그램이 다음 업데이트 확인 때까지 대기하는 기간(일)입니다.

4 updateConfig.xml 파일을 AIR 응용 프로그램의 project 디렉토리에 추가합니다.

5 업데이트 프로그램 객체가 updateConfig.xml 파일을 참조하게 하도록 객체의 initialize() 메서드를 호출합니다.

ActionScript에서는 다음 코드를 사용합니다.

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

JavaScript에서는 다음 코드를 사용합니다.

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 첫 번째 응용 프로그램과 버전이 다른 두 번째 AIR 응용 프로그램 버전을 만듭니다. 버전은 응용 프로그램 설명자 파일의 version 요소에 지정됩니다.

그런 다음 AIR 응용 프로그램의 업데이트 버전을 웹 서버에 추가합니다.

1 AIR 파일의 업데이트 버전을 웹 서버에 배치합니다.

2 updateDescriptor.xml이라는 텍스트 파일을 만들어 다음 내용을 추가합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1</version>
       <url>http://example.com/updates/sample_1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

updateDescriptor.xml 파일의 version, URL 및 description을 업데이트 AIR 파일과 일치하도록 편집합니다.

3 업데이트 AIR 파일이 포함된 동일한 웹 서버 디렉토리에 updateDescriptor.xml 파일을 추가합니다.

이것은 기본적인 예제이지만 많은 응용 프로그램에 충분한 업데이트 기능을 제공합니다. 이 문서의 나머지 부분에서는 업데이트 프레임워크를 사용자 요구에 가장 잘 맞게 사용하는 방법을 설명합니다.

업데이트 프레임워크 사용과 관련된 다른 예제를 보려면 Adobe AIR 개발자 센터의 다른 샘플 응용 프로그램을 참조하십시오.

- Flex 기반 응용 프로그램의 업데이트 프레임워크
  (http://www.adobe.com/go/learn_air_qs_update_framework_flex_kr)
- Flsh 기반 응용 프로그램의 업데이트 프레임워크
  (http://www.adobe.com/go/learn_air_qs_update_framework_flash_kr)
- HTML 기반 응용 프로그램의 업데이트 프레임워크
  (http://www.adobe.com/go/learn_air_qs_update_framework_html_kr)

업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가

AIR 업데이트 프레임워크를 사용할 때는 웹 서버에 저장된 업데이트 설명자 파일에 사용 가능한 업데이트에 대한 기본 정보를 정의합니다. 업데이트 설명자 파일은 간단한 XML 파일입니다. 응용 프로그램에 포함된 업데이트 프레임워크는 이 파일을 확인하여 새 버전이 업로드되었는지 확인합니다.

업데이트 설명자 파일에는 다음 데이터가 포함됩니다.

- version - AIR 응용 프로그램의 새 버전입니다. 새 AIR 응용 프로그램 설명자 파일에서 버전으로 사용되는 문자열과 같아야 합니다. 업데이트 설명자 파일의 버전이 업데이트 AIR 파일의 버전과 일치하지 않으면 업데이트 프레임워크에서 예외가 발생합니다.
- url - 업데이트 AIR 파일의 위치입니다. 이 파일은 AIR 응용 프로그램의 업데이트 버전을 포함합니다.
- description - 새 버전과 관련된 세부 사항입니다. 이 정보는 업데이트 프로세스 중에 사용자에게 표시될 수 있습니다.

version 및 url 요소는 필수이고, description 요소는 선택 사항입니다.

다음은 샘플 업데이트 설명자 파일입니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

여러 언어를 사용하여 description 태그를 정의하려면 lang 특성을 정의하는 여러 text 요소를 사용합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>
           <text xml:lang="en">English description</text>
           <text xml:lang="fr">French description</text>
           <text xml:lang="ro">Romanian description</text>
       </description>
    </update>
```

업데이트 설명자 파일을 업데이트 AIR 파일과 함께 웹 서버에 배치합니다.

업데이트 설명자에 포함된 templates 디렉토리에는 샘플 업데이트 설명자 파일이 포함됩니다. 여기에는 단일 언어 버전과 여러 언어 버전이 모두 포함됩니다.

업데이트 프로그램 객체 인스턴스화

코드에서 AIR 업데이트 프레임워크를 로드한 후(97페이지의 “Flex 개발 환경 설정” 및 98페이지의 “HTML 기반 AIR 응용 프로그램에 프레임워크 파일 포함” 참조) 다음과 같이 업데이트 프로그램 객체를 인스턴스화해야 합니다

ActionScript 예제:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

JavaScript 예제:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

앞의 코드에서는 사용자 인터페이스를 제공하지 않는 ApplicationUpdater 클래스를 사용합니다. 사용자 인터페이스를 제공하는 ApplicationUpdaterUI 클래스를 사용하려면 다음을 사용합니다.

ActionScript 예제:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

JavaScript 예제:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

이 문서의 나머지 코드 샘플에서는 appUpdater라는 업데이트 프로그램 객체를 인스턴스화한 것으로 가정합니다.

업데이트 설정 구성

ApplicationUpdater 및 ApplicationUpdaterUI는 모두 응용 프로그램과 함께 제공되는 구성 파일을 통해 또는 응용 프로그램의 ActionScript나 JavaScript를 통해 구성할 수 있습니다.

XML 구성 파일에서 업데이트 설정 정의

업데이트 구성 파일은 XML 파일이며, 다음 요소를 포함할 수 있습니다.

- updateURL - 문자열입니다. 원격 서버에 있는 업데이트 설명자의 위치를 나타냅니다. 유효한 URLRequest 위치를 사용할 수 있습니다. 구성 파일이나 스크립트를 통해 updateURL 속성을 정의해야 합니다(99페이지의 “업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가” 참조). 업데이트 프로그램을 사용하기 전에 즉, 103페이지의 “업데이트 프로그램 초기화”에서 설명한 대로 업데이트 프로그램 객체의 initialize() 메서드를 호출하기 전에 이 속성을 정의해야 합니다.
- delay - 숫자입니다. 업데이트를 확인하는 시간 간격(일)입니다. 예를 들어 0.25와 같은 값을 사용할 수 있습니다. 값이 0(기본값)이면 업데이트 프로그램이 자동 정기 확인을 수행하지 않습니다.

ApplicationUpdaterUI의 구성 파일에는 updateURL 및 delay 요소와 함께 다음 요소가 포함될 수 있습니다.

- defaultUI: dialog 요소 목록입니다. 각 dialog 요소에는 사용자 인터페이스의 대화 상자에 해당하는 name 특성이 있습니다. 각 dialog 요소에는 대화 상자가 표시되는지 여부를 정의하는 visible 특성이 있습니다. 기본값은 true입니다. name 특성에 사용할 수 있는 값은 다음과 같습니다.
  - "checkForUpdate" - 업데이트 확인, 업데이트 없음 및 업데이트 오류 대화 상자에 해당합니다.
  - "downloadUpdate" - 업데이트 다운로드 대화 상자에 해당합니다.
  - "downloadProgress" - 다운로드 진행률 및 다운로드 오류 대화 상자에 해당합니다.
  - "installUpdate" - 업데이트 설치 대화 상자에 해당합니다.
  - "fileUpdate" - 파일 업데이트, 파일 업데이트 없음 및 파일 오류 대화 상자에 해당합니다.
- "unexpectedError" - 예기치 않은 오류 대화 상자에 해당합니다.

  false로 설정할 경우 해당하는 대화 상자가 업데이트 절차의 일부로 표시되지 않습니다.

다음은 ApplicationUpdater 프레임워크의 구성 파일 예제입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
</configuration>
```

다음은 defaultUI 요소에 대한 정의를 포함하는 ApplicationUpdaterUI 프레임워크의 구성 파일 예제입니다.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
      <defaultUI>
         <dialog name="checkForUpdate" visible="false" />
         <dialog name="downloadUpdate" visible="false" />
         <dialog name="downloadProgress" visible="false" />
      </defaultUI>
</configuration>
```

configurationFile 속성을 해당 파일의 위치로 지정합니다.

ActionScript 예제:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

JavaScript 예제:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

업데이트 프레임워크의 templates 디렉토리에는 샘플 구성 파일 config-template.xml이 포함됩니다.

업데이트 설정 ActionScript 또는 JavaScript 코드 정의

이러한 구성 매개 변수를 다음과 같이 응용 프로그램의 코드를 사용하여 설정할 수도 있습니다.

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

업데이트 프로그램 객체의 속성은 updateURL 및 delay입니다. 이러한 속성은 구성 파일의 updateURL 및 delay 요소와 동일한 설정 즉, 업데이트 설명자 파일의 URL과 업데이트 확인 간격을 정의합니다. 구성 파일 및 코드의 설정을 지정하는 경우 코드를 사용하여 설정한 속성이 구성 파일의 해당하는 설정보다 우선합니다.

업데이트 프로그램을 사용하기 전에 즉, 103페이지의 “업데이트 프로그램 초기화”에서 설명한 대로 업데이트 프로그램 객체의 initialize() 메서드를 호출하기 전에 구성 파일이나 스크립트를 통해 updateURL 속성을 정의해야 합니다(99페이지의 “업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가” 참조).

ApplicationUpdaterUI 프레임워크는 업데이트 프로그램 객체에 대한 다음과 같은 추가 속성을 정의합니다.

- isCheckForUpdateVisible - 업데이트 확인, 업데이트 없음 및 업데이트 오류 대화 상자에 해당합니다.
- isDownloadUpdateVisible - 업데이트 다운로드 대화 상자에 해당합니다.
- isDownloadProgressVisible - 다운로드 진행률 및 다운로드 오류 대화 상자에 해당합니다.
- isInstallUpdateVisible - 업데이트 설치 대화 상자에 해당합니다.
- isFileUpdateVisible - 파일 업데이트, 파일 업데이트 없음 및 파일 오류 대화 상자에 해당합니다.
- isUnexpectedErrorVisible - 예기치 않은 오류 대화 상자에 해당합니다.

각 속성은 ApplicationUpdaterUI 사용자 인터페이스에 있는 하나 이상의 대화 상자에 해당합니다. 각 속성은 기본값이 true인 부울 값입니다. false로 설정할 경우 해당하는 대화 상자가 업데이트 절차의 일부로 표시되지 않습니다.

이러한 대화 상자 속성은 업데이트 구성 파일의 설정을 재정의합니다.

업데이트 프로세스

AIR 업데이트 프레임워크에서는 다음 단계에 따라 업데이트 프로세스를 완료합니다.

1. 업데이트 프로그램 초기화에서 업데이트 확인이 정의된 지연 시간 간격 내에 수행되었는지 확인합니다(100페이지의 “업데이트 설정 구성” 참조). 업데이트 확인이 예정되어 있는 경우 업데이트 프로세스가 계속됩니다.
2. 업데이트 프로그램에서 업데이트 설명자 파일을 다운로드하고 해석합니다.
3. 업데이트 프로그램이 업데이트 AIR 파일을 다운로드합니다.
4. 업데이트 프로그램이 업데이트된 응용 프로그램 버전을 설치합니다.

업데이트 프로그램 객체는 이러한 각 단계 완료 시 이벤트를 전달합니다. ApplicationUpdater 버전에서 프로세스 단계의 성공적인 완료를 알려 주는 이벤트를 취소할 수 있습니다. 이러한 이벤트 중 하나를 취소하려면 프로세스의 다음 단계는 취소됩니다. ApplicationUpdaterUI 버전에서 업데이트 프로그램은 사용자가 프로세스의 각 단계에서 취소 또는 진행 여부를 선택할 수 있는 대화 상자를 표시합니다.

이벤트를 취소하는 경우 업데이트 프로그램 객체의 메서드를 호출하여 프로세스를 다시 시작할 수 있습니다.

업데이트 프로그램의 ApplicationUpdater 버전에서 업데이트 프로세스를 진행하는 동안 현재 상태를 currentState 속성에 기록합니다. 이 속성은 다음과 같은 값의 문자열로 설정됩니다.

- "UNINITIALIZED" - 업데이트 프로그램이 초기화되지 않았습니다.
- "INITIALIZING" - 업데이트 프로그램이 초기화되고 있습니다.
- "READY" - 업데이트 프로그램이 초기화되었습니다
- "BEFORE_CHECKING" - 업데이트 프로그램이 업데이트 설명자 파일을 아직 확인하지 않았습니다.
- "CHECKING" - 업데이트 프로그램이 업데이트 설명자 파일을 확인하고 있습니다.

- "AVAILABLE" - 업데이트 설명자 파일을 사용할 수 있습니다.
- "DOWNLOADING" - 업데이트 프로그램이 AIR 파일을 다운로드하고 있습니다.
- "DOWNLOADED" - 업데이트 프로그램이 AIR 파일을 다운로드했습니다.
- "INSTALLING" - 업데이트 프로그램이 AIR 파일을 설치하고 있습니다.
- "PENDING_INSTALLING" - 업데이트 프로그램이 초기화되었고 대기 중인 업데이트가 있습니다.

업데이트 프로그램 객체의 일부 메서드는 업데이트 프로그램이 특정 상태에 있는 경우 경우에만 실행됩니다.

업데이트 프로그램 초기화

구성 속성을 설정한 후(98페이지의 "기본 예제: ApplicationUpdaterUI 버전 사용") initialize() 메서드를 사용하여 업데이트를 초기화합니다.

```
appUpdater.initialize();
```

이 메서드는 다음과 같은 작업을 수행합니다.

- 대기 중인 업데이트를 동기적으로 자동 설치하여 업데이트 프레임워크를 초기화합니다. 이 메서드를 호출할 때 응용 프로그램을 다시 시작할 수 있으므로 응용 프로그램 시작 시 이 메서드를 호출해야 합니다.
- 연기된 업데이트가 있는지 확인한 다음 설치합니다.
- 업데이트 프로세스 동안 오류가 발생하는 경우 응용 프로그램 저장소 영역에서 업데이트 파일 및 버전 정보를 지웁니다.
- 지연 시간이 만료되는 경우 업데이트 프로세스를 시작합니다. 그렇지 않은 경우 타이머를 다시 시작합니다.

이 메서드를 호출하면 업데이트 프로그램 객체에서 다음 이벤트를 전달합니다.

- UpdateEvent.INITIALIZED - 초기화가 완료될 때 전달됩니다.
- ErrorEvent.ERROR - 초기화 동안 오류가 발생하는 경우 전달됩니다.

UpdateEvent.INITIALIZED 이벤트가 전달되면 업데이트 프로세스가 완료됩니다.

initialize() 메서드를 호출하면 업데이트 프로그램이 업데이트 프로세스를 시작하고 타이머 지연 시간 설정에 따라 모든 단계를 완료합니다. 그러나 언제든 업데이트 프로그램 객체의 checkNow() 메서드를 호출하여 업데이트 프로세스를 시작할 수 있습니다.

```
appUpdater.checkNow();
```

업데이트 프로세스가 이미 실행되고 있는 경우 이 메서드는 아무 작업도 수행하지 않습니다. 그렇지 않으면 업데이트 프로세스를 시작합니다.

업데이트 프로그램 객체는 checkNow() 메서드 호출의 결과로 다음 이벤트를 전달합니다.

- UpdateEvent.CHECK_FOR_UPDATE 이벤트 - 업데이트 설명자 파일을 다운로드하기 바로 전에 전달됩니다.

checkForUpdate 이벤트를 취소하는 경우 업데이트 프로그램 객체의 checkForUpdate() 메서드를 호출할 수 있습니다. 다음 단원을 참조하십시오. 이벤트를 취소하지 않으면 업데이트 프로세스에서 업데이트 설명자 파일을 확인하는 과정을 계속합니다.

ApplicationUpdaterUI 버전에서 업데이트 프로세스 관리

ApplicationUpdaterUI 버전에서 사용자는 사용자 인터페이스의 대화 상자에서 취소 버튼을 통해 프로세스를 취소할 수 있습니다. 또한 ApplicationUpdaterUI 객체의 cancelUpdate() 메서드를 호출하여 업데이트 프로세스를 프로그래밍 방식으로 취소할 수 있습니다.

ApplicationUpdaterUI 객체의 속성을 설정하거나 업데이트 구성 파일의 요소를 정의하여 업데이트 프로그램에서 표시하는 대화 상자 확인을 지정할 수 있습니다. 자세한 내용은 100페이지의 "업데이트 설정 구성"을 참조하십시오.

ApplicationUpdater 버전에서 업데이트 프로세스 관리

ApplicationUpdater 객체에서 전달한 이벤트의 preventDefault() 메서드를 호출하여 업데이트 프로세스의 단계를 취소할 수 있습니다(102페이지의 “업데이트 프로세스” 참조). 기본 비헤이비어를 취소하면 응용 프로그램에서 사용자에게 계속할지 묻는 메시지를 표시할 수 있게 됩니다.

다음 단원에서는 프로세스 단계가 취소된 경우 업데이트 프로세스를 계속하는 방법에 대해 설명합니다.

업데이트 설명자 파일 다운로드 및 해석

업데이트 프로그램에서 업데이트 설명자 파일을 다운로드하기 직전에 업데이트 프로세스가 시작되기 전에 ApplicationUpdater 객체에서 checkForUpdate 이벤트를 전달합니다. checkForUpdate 이벤트의 기본 비헤이비어를 취소하는 경우 업데이트 프로그램은 업데이트 설명자 파일을 다운로드하지 않습니다. 다음과 같이 checkForUpdate() 메서드를 호출하여 업데이트 프로세스를 다시 시작할 수 있습니다.

```
appUpdater.checkForUpdate();
```

checkForUpdate() 메서드를 호출하면 업데이트 프로그램이 업데이트 설명자 파일을 비동기적으로 다운로드하여 해석합니다. checkForUpdate() 메서드 호출의 결과로 업데이트 프로그램 객체에서 다음 이벤트를 전달할 수 있습니다.

- StatusUpdateEvent.UPDATE_STATUS - 업데이트 프로그램이 업데이트 설명자 파일을 성공적으로 다운로드하고 해석했습니다. 이 이벤트에는 다음과 같은 속성이 있습니다.
  - always - 부울 값입니다. 현재 응용 프로그램의 버전과 다른 버전을 사용할 수 있는 경우 true이고, 그렇지 않은 경우(버전이 동일한 경우) false입니다.
  - version - 문자열입니다. 업데이트 파일의 응용 프로그램 설명자 파일의 버전입니다.
  - details - 배열입니다. 설명에 대한 지역화된 버전이 없는 경우 이 배열은 빈 문자열("")을 첫 번째 요소로, 설명을 두 번째 요소로 반환합니다.

    업데이트 설명자 파일에 여러 버전의 설명이 있는 경우 이 배열에는 여러 하위 배열이 포함됩니다. 각 배열에는 두 요소가 포함되며, 첫째는 언어 코드(예: "en")이고 둘째는 언어에 해당하는 설명(문자열)입니다. 99페이지의 “업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가”를 참조하십시오.
- StatusUpdateErrorEvent.UPDATE_ERROR - 오류가 발생했고 업데이트 프로그램에서 업데이트 설명자 파일을 다운로드 또는 해석하지 못했습니다.

업데이트 AIR 파일 다운로드

업데이트 프로그램에서 업데이트 설명자 파일을 성공적으로 다운로드 및 해석한 후 ApplicationUpdater 객체는 updateStatus 이벤트를 전달합니다. 기본 비헤이비어는 사용 가능한 업데이트가 있는 경우 다운로드를 시작하는 것입니다. 기본 비헤이비어를 취소하는 경우 downloadUpdate() 메서드를 호출하여 업데이트 프로세스를 다시 시작할 수 있습니다.

```
appUpdater.downloadUpdate();
```

이 메서드를 호출하면 업데이트 프로그램에서는 AIR 파일의 업데이트 버전을 비동기적으로 다운로드합니다.

downloadUpdate() 메서드는 다음 이벤트를 전달할 수 있습니다.

- UpdateEvent.DOWNLOAD_START - 서버에 대한 연결이 설정되었습니다. ApplicationUpdaterUI 라이브러리를 사용할 경우 이 이벤트는 다운로드 진행률을 추적하는 진행률 막대가 있는 대화 상자를 표시합니다.
- ProgressEvent.PROGRESS - 파일 다운로드가 진행될 때 주기적으로 전달됩니다.
- DownloadErrorEvent.DOWNLOAD_ERROR - 업데이트 파일을 연결하거나 다운로드하는 동안 오류가 발생하는 경우 전달됩니다. HTTP 상태가 잘못된 경우에도 전달됩니다(예: "404 - 파일을 찾을 수 없습니다”). 이 이벤트에는 추가 오류 정보를 정의하는 정수인 errorID 속성이 있습니다. 추가 subErrorID 속성에는 추가 오류 정보가 포함될 수 있습니다.
- UpdateEvent.DOWNLOAD_COMPLETE - 업데이트 프로그램이 업데이트 설명자 파일을 성공적으로 다운로드하고 해석했습니다. 이 이벤트를 취소하지 않으면 ApplicationUpdater 버전이 업데이트 버전 설치를 진행합니다. ApplicationUpdaterUI 버전에서는 계속하기 위한 옵션을 제공하는 대화 상자를 표시합니다.

응용 프로그램 업데이트

업데이트 다운로드가 완료되면 ApplicationUpdater 객체는 downloadComplete 이벤트를 전달합니다. 기본 비헤이비어를 취소하는 경우 installUpdate() 메서드를 호출하여 업데이트 프로세스를 다시 시작할 수 있습니다.

```
appUpdater.installUpdate(file);
```

이 메서드를 호출하면 업데이트 프로그램은 AIR 파일의 업데이트 버전을 설치합니다. 이 메서드에는 업데이트 사용할 AIR 파일을 참조하는 File 객체인 file 매개 변수 하나가 포함되어 있습니다.

installUpdate() 메서드 호출의 결과로 ApplicationUpdater 객체가 beforeInstall 이벤트를 전달할 수 있습니다.

- UpdateEvent.BEFORE_INSTALL - 업데이트를 설치하기 직전에 전달됩니다. 경우에 따라 사용자가 업데이트를 계속하기 전에 현재 작업을 완료할 수 있도록 지금 업데이트를 설치하지 않도록 하는 데 이 이벤트를 사용할 수 있습니다. Event 객체의 preventDefault() 메서드를 호출하면 다음 재시작 시까지 설치가 연기되고 추가 업데이트 프로세스를 시작할 수 없습니다. 여기에는 checkNow() 메서드 호출의 결과나 정기적인 확인으로 인해 발생하는 업데이트도 포함됩니다.

임의의 AIR 파일에서 설치

installFromAIRFile() 메서드를 호출하여 사용자 컴퓨터에 있는 AIR 파일의 업데이트 버전을 설치할 수 있습니다.

```
appUpdater.installFromAIRFile();
```

이 메서드를 호출하면 업데이트 프로그램은 AIR 파일의 업데이트 버전을 설치합니다.

installFromAIRFile() 메서드는 다음 이벤트를 전달할 수 있습니다.

- StatusFileUpdateEvent.FILE_UPDATE_STATUS - ApplicationUpdater에서 installFromAIRFile() 메서드를 사용하여 전송한 파일의 유효성을 검사한 후 전달됩니다. 이 이벤트에는 다음과 같은 속성이 있습니다.
  - available - 현재 응용 프로그램 버전과 다른 버전을 사용할 수 있는 경우 true이고 그렇지 않은 경우(버전이 동일한 경우) false입니다.
  - version - 사용할 수 있는 새 버전을 나타내는 문자열입니다.
  - path - 업데이트 파일의 기본 경로를 나타냅니다.

  StatusFileUpdateEvent 객체의 available 속성을 true로 설정한 경우 이 이벤트를 취소할 수 있습니다. 이 이벤트를 취소하면 업데이트 진행이 취소됩니다. 취소된 업데이트를 계속하려면 installUpdate() 메서드를 호출합니다.
- StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR - 오류가 발생하여 업데이트 프로그램이 AIR 응용 프로그램을 설치하지 못했습니다.

업데이트 프로세스 취소

cancelUpdate() 메서드를 호출하여 업데이트 프로세스를 취소할 수 있습니다.

```
appUpdater.cancelUpdate();
```

이 메서드는 완료되지 않은 다운로드 파일을 삭제하여 대기 중인 다운로드를 취소하고 정기 확인 타이머를 다시 시작합니다.

업데이트 프로그램 객체가 초기화되고 있는 경우 이 메서드는 아무 작업도 수행하지 않습니다.

ApplicationUpdaterUI 인터페이스 지역화

ApplicationUpdaterUI 클래스에서는 업데이트 프로세스를 위한 기본 사용자 인터페이스를 제공합니다. 여기에는 사용자가 프로세스를 시작하고, 프로세스를 취소하고, 기타 관련 작업을 수행할 수 있는 대화 상자가 포함됩니다.

업데이트 설명자 파일의 description 요소를 사용하면 응용 프로그램에 대한 설명을 여러 언어로 정의할 수 있습니다. 다음과 같이 lang 특성을 정의하는 여러 text 요소를 사용합니다.

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>
           <text xml:lang="en">English description</text>
           <text xml:lang="fr">French description</text>
           <text xml:lang="ro">Romanian description</text>
       </description>
    </update>
```

업데이트 프레임워크에서는 최종 사용자의 지역화 체인에 가장 잘 맞는 설명을 사용합니다. 자세한 내용은 업데이트 설명자 파일 정의 및 웹 서버에 AIR 파일 추가를 참조하십시오.

Flex 개발자는 "ApplicationUpdaterDialogs" 번들에 새 언어를 직접 추가할 수 있습니다.

JavaScript 개발자는 업데이트 프로그램 객체의 addResources() 메서드를 호출할 수 있습니다. 이 메서드는 언어에 대한 새 리소스 번들을 동적으로 추가합니다. 리소스 번들은 언어에 대한 지역화된 문자열을 정의합니다. 이러한 문자열은 다양한 대화 상자 텍스트 필드에서 사용됩니다.

JavaScript 개발자는 ApplicationUpdaterUI 클래스의 localeChain 속성을 사용하여 사용자 인터페이스에서 사용되는 로캘 체인을 정의할 수 있습니다. 일반적으로 JavaScript(HTML) 개발자만 이 속성을 사용합니다. Flex 개발자는 ResourceManager를 사용하여 로캘 체인을 관리합니다.

예를 들어 다음 JavaScript 코드는 루마니아어와 헝가리어의 리소스 번들을 정의합니다.

```
appUpdater.addResources("ro_RO",
                {titleCheck: "Titlu", msgCheck: "Mesaj", btnCheck: "Buton"});
appUpdater.addResources("hu", {titleCheck: "Cím", msgCheck: "Üzenet"});
var languages = ["ro", "hu"];
languages = languages.concat(air.Capabilities.languages);
var sortedLanguages = air.Localizer.sortLanguagesByPreference(languages,
                     air.Capabilities.language,
                     "en-US");
sortedLanguages.push("en-US");
appUpdater.localeChain = sortedLanguages;
```

자세한 내용은 언어 참조 설명서에서 ApplicationUpdaterUI 클래스의 addResources() 메서드에 대한 설명을 참조하십시오.

15장: 응용 프로그램 설정 읽기

런타임에 응용 프로그램 설명자 파일의 속성과 응용 프로그램의 제작자 ID를 가져올 수 있습니다. 이러한 항목은 NativeApplication 객체의 applicationDescriptor 및 publisherID 속성에서 설정됩니다.

응용 프로그램 설명자 파일 읽기

NativeApplication 객체의 applicationDescriptor 속성을 가져와서 현재 실행되는 응용 프로그램의 응용 프로그램 설명자 파일을 XML 객체로 읽을 수 있습니다.

ActionScript 3.0 예제:

```
var appXml:XML = NativeApplication.nativeApplication.applicationDescriptor;
```

JavaScript 예제:

```
var appXml:XML = air.ativeApplication.nativeApplication.applicationDescriptor;
```

ActionScript 3.0에서 다음과 같이 XML(E4X) 객체로 응용 프로그램 설명자 데이터에 액세스할 수 있습니다.

```
var appXml:XML = NativeApplication.nativeApplication.applicationDescriptor;
var ns:Namespace = appXml.namespace();
var appId = appXml.ns::id[0];
var appVersion = appXml.ns::version[0];
var appName = appXml.ns::filename[0];
air.trace("appId:", appId);
air.trace("version:", appVersion);
air.trace("filename:", appName);
var xmlString = air.NativeApplication.nativeApplication.applicationDescriptor;
```

JavaScript에서 다음과 같이 DOMParser 객체를 사용하여 데이터를 파싱할 수 있습니다.

```
var xmlString = air.NativeApplication.nativeApplication.applicationDescriptor;
var appXml = new DOMParser();
var xmlobject = appXml.parseFromString(xmlString, "text/xml");
var root = xmlobject.getElementsByTagName('application')[0];
var appId = root.getElementsByTagName("id")[0].firstChild.data;
var appVersion = root.getElementsByTagName("version")[0].firstChild.data;
var appName = root.getElementsByTagName("filename")[0].firstChild.data;
air.trace("appId:", appId);
air.trace("version:", appVersion);
air.trace("filename:", appName);
```

자세한 내용은 62페이지의 “응용 프로그램 설명자 파일 구조”를 참조하십시오.

응용 프로그램 및 제작자 ID 가져오기

응용 프로그램 및 제작자 ID가 함께 AIR 응용 프로그램을 고유하게 식별합니다. 응용 프로그램 설명자의 <id> 요소에서 응용 프로그램 ID를 지정합니다. 제작자 ID는 AIR 설치 패키지에 서명하는 데 사용되는 인증서에서 파생됩니다.

다음 코드와 같이 응용 프로그램 ID를 NativeApplication 객체의 id 속성에서 읽을 수 있습니다.

ActionScript 3.0 예제:

```
trace(NativeApplication.nativeApplication.applicationID);
```

JavaScript 예제:

```
air.trace(air.NativeApplication.nativeApplication.applicationID);
```

제작자 ID는 NativeApplication 객체의 publisherID 속성에서 읽을 수 있습니다.

ActionScript 3.0 예제:

```
trace(NativeApplication.nativeApplication.publisherID);
```

ActionScript 3.0 예제:

```
air.trace(air.NativeApplication.nativeApplication.publisherID);
```

참고: AIR 응용 프로그램이 ADL과 함께 실행되는 경우 ADL 명령줄에서 -pubID 플래그를 사용하여 제작자 ID를 임시로 할당하지 않는 한 AIR 응용 프로그램에 제작자 ID가 없습니다.

설치된 응용 프로그램의 제작자 ID는 응용 프로그램 설치 디렉토리에 있는 META-INF/AIR/publisherid 파일에서도 찾을 수 있습니다.

자세한 내용은 85페이지의 “AIR 제작자 ID”를 참조하십시오.

16장: 소스 코드 보기

웹 브라우저에서 HTML 페이지의 소스 코드를 볼 수 있는 것과 마찬가지로 HTML 기반 AIR 응용 프로그램의 소스 코드를 볼 수 있습니다. Adobe® AIR® SDK에는 응용 프로그램에서 최종 사용자에게 소스 코드를 보여 줄 때 사용할 수 있는 AIRSourceViewer.js라는 JavaScript 파일이 포함되어 있습니다.

소스 뷰어 로드, 구성 및 열기

소스 뷰어 코드는 JavaScript 파일인 AIRSourceViewer.js에 포함되어 있으며, 이 파일은 AIR SDK의 frameworks 디렉토리에 있습니다. 응용 프로그램에서 소스 뷰어를 사용하려면 AIRSourceViewer.js를 응용 프로그램 프로젝트 디렉토리에 복사하고 응용 프로그램의 기본 HTML 파일에서 script 태그를 사용하여 로드합니다.

```
<script type="text/javascript" src="AIRSourceViewer.js"></script>
```

AIRSourceViewer.js 파일은 SourceViewer 클래스를 정의합니다. JavaScript 코드에서는 air.SourceViewer를 호출하여 이 클래스에 액세스할 수 있습니다.

SourceViewer 클래스는 getDefault(), setup() 및 viewSource()라는 세 가지 메서드를 정의합니다.

메서드	설명
getDefault()	정적 메서드입니다. 다른 메서드를 호출하는 데 사용할 수 있는 SourceViewer 인스턴스를 반환합니다.
setup()	소스 뷰어에 구성 설정을 적용합니다. 자세한 내용은 109페이지의 "소스 뷰어 구성"을 참조하십시오.
viewSource()	사용자가 호스트 응용 프로그램의 소스 파일을 찾아보고 열 수 있는 새 윈도우를 엽니다.

참고: 소스 뷰어를 사용하는 코드는 응용 프로그램 보안 샌드박스(응용 프로그램 디렉토리의 파일)에 있어야 합니다.

예를 들어, 다음 JavaScript 코드는 소스 뷰어 객체를 인스턴스화하고 모든 소스 파일을 나열하는 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

소스 뷰어 구성

config() 메서드는 지정된 설정을 소스 뷰어에 적용합니다. 이 메서드는 configObject 매개 변수 하나만 사용합니다. configObject 객체에는 소스 뷰어에 대한 구성 설정을 정의하는 속성이 있습니다. 속성은 default, exclude, initialPosition, modal, typesToRemove 및 typesToAdd입니다.

default

소스 뷰어에 표시할 초기 파일에 대한 상대 경로를 지정하는 문자열입니다.

예를 들어, 다음 JavaScript 코드는 index.html 파일이 초기 파일로 표시된 상태로 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude

소스 뷰어 목록에서 제외할 파일이나 디렉토리를 지정하는 문자열 배열입니다. 경로는 응용 프로그램 디렉토리에 대해 상대적입니다. 와일드카드 문자는 사용할 수 없습니다.

예를 들어, 다음 JavaScript 코드는 소스 뷰어 윈도우를 열고 AIRSourceViewer.js 파일과 Images 및 Sounds 하위 디렉토리의 파일을 제외한 모든 소스 파일을 나열합니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition

소스 뷰어 윈도우의 초기 x 및 y 좌표를 지정하는 숫자 두 개를 포함하는 배열입니다.

예를 들어, 다음 JavaScript 코드는 화면 좌표 [40, 60](X = 40, Y = 60)에 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal

소스 뷰어 윈도우가 모달 윈도우(true)인지 비 모달 윈도우(false)인지를 지정하는 부울 값입니다. 기본적으로 소스 뷰어 윈도우는 모달 윈도우입니다.

예를 들어, 다음 JavaScript 코드는 사용자가 소스 뷰어 윈도우와 다른 응용 프로그램 윈도우를 동시에 사용할 수 있도록 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd

기본 유형과 함께 소스 뷰어 목록에 포함될 파일 유형을 지정하는 문자열 배열입니다.

기본적으로 소스 뷰어 윈도우는 다음 파일 유형을 나열합니다.

- 텍스트 파일 - TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- 이미지 파일 - JPG, JPEG, PNG, GIF

  값을 지정하지 않으면 typesToExclude 속성에 지정된 유형을 제외한 모든 기본 유형이 포함됩니다.

  예를 들어, 다음 JavaScript 코드는 VCF와 VCARD 파일이 포함된 소스 뷰어 윈도우를 엽니다.

  ```
  var viewer = air.SourceViewer.getDefault();
  var configObj = {};
  configObj.typesToAdd = ["text.vcf", "text.vcard"];
  viewer.viewSource(configObj);
  ```

  나열하는 각 파일 유형에 대해 "text"(텍스트 파일 유형) 또는 "image"(이미지 파일 유형)를 지정해야 합니다.

typesToExclude

소스 뷰어에서 제외할 파일 유형을 지정하는 문자열 배열입니다.

기본적으로 소스 뷰어 윈도우는 다음 파일 유형을 나열합니다.

- 텍스트 파일 - TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG
- 이미지 파일 - JPG, JPEG, PNG, GIF

  예를 들어, 다음 JavaScript 코드는 GIF나 XML 파일을 나열하지 않도록 소스 뷰어 윈도우를 엽니다.

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

나열하는 각 파일 유형에 대해 "text"(텍스트 파일 유형) 또는 "image"(이미지 파일 유형)를 지정해야 합니다.

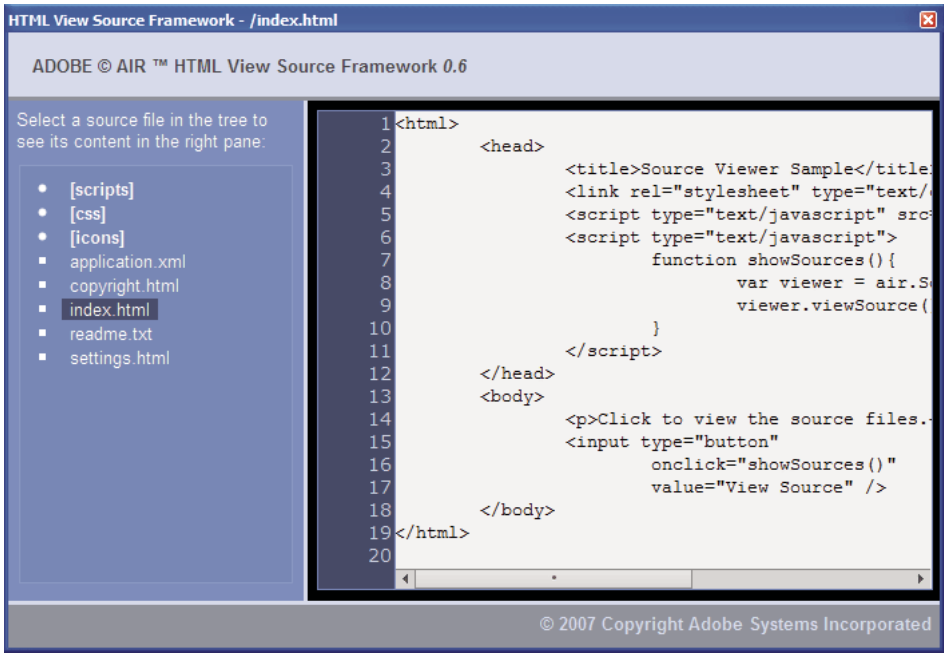
소스 뷰어 열기

사용자가 선택하여 소스 뷰어 코드를 호출할 수 있는 링크, 버튼, 메뉴 명령 등과 같은 사용자 인터페이스 요소가 있어야 합니다. 예를 들어, 다음은 사용자가 링크를 클릭하면 소스 뷰어를 여는 간단한 응용 프로그램입니다.

```
<html>
    <head>
        <title>Source Viewer Sample</title>
        <script type="text/javascript" src="AIRSourceViewer.js"></script>
        <script type="text/javascript">
            function showSources(){
                var viewer = air.SourceViewer.getDefault();
                viewer.viewSource()
            }
        </script>
    </head>
    <body>
        <p>Click to view the source files.</p>
        <input type="button"
            onclick="showSources()"
            value="View Source" />
    </body>
</html>
```

소스 뷰어 사용자 인터페이스

응용 프로그램에서 SourceViewer 객체의 viewSource() 메서드를 호출하면 AIR 응용 프로그램이 소스 뷰어 윈도우를 엽니다. 이 윈도우 왼쪽에는 소스 파일 및 디렉토리 목록이 있고 오른쪽에는 선택한 파일의 소스 코드를 표시하는 디스플레이 영역이 있습니다.



디렉토리는 대괄호 안에 나열됩니다. 사용자는 대괄호를 클릭하여 디렉토리 목록을 확장하거나 축소할 수 있습니다.

소스 뷰어는 인식된 확장명을 가진 텍스트 파일(HTML, HTML, JS, TXT, XML 등)이나 인식된 이미지 확장명을 가진 이미지 파일(JPG, JPEG, PNG 및 GIF)의 소스를 표시할 수 있습니다. 사용자가 인식된 파일 확장명이 없는 파일을 선택하면 "이 파일 유형에서 텍스트 내용을 검색할 수 없습니다."라는 오류 메시지가 표시됩니다.

setup() 메서드를 통해 제외된 모든 소스 파일은 나열되지 않습니다(109페이지의 “소스 뷰어 로드, 구성 및 열기” 참조).

17장: AIR HTML Introspector를 사용한 디버깅

Adobe® AIR® SDK에는 응용 프로그램에 포함시켜 HTML 기반 응용 프로그램을 디버깅하는 데 사용할 수 있는 AIRIntrospector.js JavaScript 파일이 포함되어 있습니다.

AIR Introspector

Adobe AIR HTML/JavaScript Application Introspector(AIR HTML Introspector)는 HTML 기반 응용 프로그램을 개발하고 디버깅하는 데 도움을 주는 다음과 같은 유용한 기능을 제공합니다.

- 응용 프로그램의 사용자 인터페이스 요소를 가리켜 해당 마크업과 DOM 속성을 확인할 수 있는 검사 도구가 포함되어 있습니다.
- 검사를 위해 객체 참조를 보내거나, 속성 값을 조정하고 JavaScript 코드를 실행할 수 있는 콘솔이 포함되어 있습니다. 또한 객체를 콘솔과 직렬화하여 데이터를 편집하지 못하도록 제한할 수 있습니다. 콘솔에서 텍스트를 복사하고 저장할 수 있습니다.
- DOM 속성 및 함수의 트리 보기가 포함되어 있습니다.
- DOM 요소의 특성과 텍스트 노드를 편집할 수 있습니다.
- 응용 프로그램에 로드된 링크, CSS 스타일, 이미지 및 JavaScript 파일을 목록으로 표시합니다.
- 사용자 인터페이스의 초기 HTML 소스와 현재 마크업 소스를 확인할 수 있습니다.
- 응용 프로그램 디렉토리에 있는 파일에 액세스할 수 있습니다. 이 기능은 응용 프로그램 샌드박스에 대해 열린 AIR HTML Introspector 콘솔에서만 사용할 수 있습니다. 응용 프로그램 샌드박스 내용에서 열린 콘솔이 아닌 경우 이 기능을 사용할 수 없습니다.
- XMLHttpRequest 객체와 responseText, responseXML 등을 비롯한 해당 속성(사용할 수 있는 경우)을 볼 수 있는 뷰어가 포함되어 있습니다.
- 소스 코드와 파일에서 일치하는 텍스트를 검색할 수 있습니다.

AIR Introspector 코드 로드

AIR Introspector 코드는 JavaScript 파일인 AIRIntrospector.js에 포함되어 있으며, 이 파일은 AIR SDK의 frameworks 디렉토리에 있습니다. 응용 프로그램에서 AIR Introspector를 사용하려면 AIRIntrospector.js를 응용 프로그램 프로젝트 디렉토리에 복사하고 응용 프로그램의 기본 HTML 파일에서 script 태그를 사용하여 로드합니다.

```
<script type="text/javascript" src="AIRIntrospector.js"></script>
```

또한 응용 프로그램의 다양한 기본 윈도우에 해당하는 모든 HTML 파일을 파일에 포함시킵니다.

중요: AIRIntrospector.js 파일은 응용 프로그램을 개발하고 디버깅할 때만 포함시키십시오. 배포하려고 패키지화한 AIR 응용 프로그램에서는 이 파일을 제거해야 합니다.

AIRIntrospector.js 파일은 Console 클래스를 정의합니다. JavaScript 코드에서는 air.Introspector.Console을 호출하여 이 클래스에 액세스할 수 있습니다.

참고: AIR Introspector를 사용하는 코드는 응용 프로그램 보안 샌드박스(응용 프로그램 디렉토리의 파일)에 있어야 합니다.

콘솔 탭에서 객체 검사

Console 클래스에는 log(), warn(), info(), error() 및 dump()의 다섯 가지 메서드가 정의되어 있습니다.

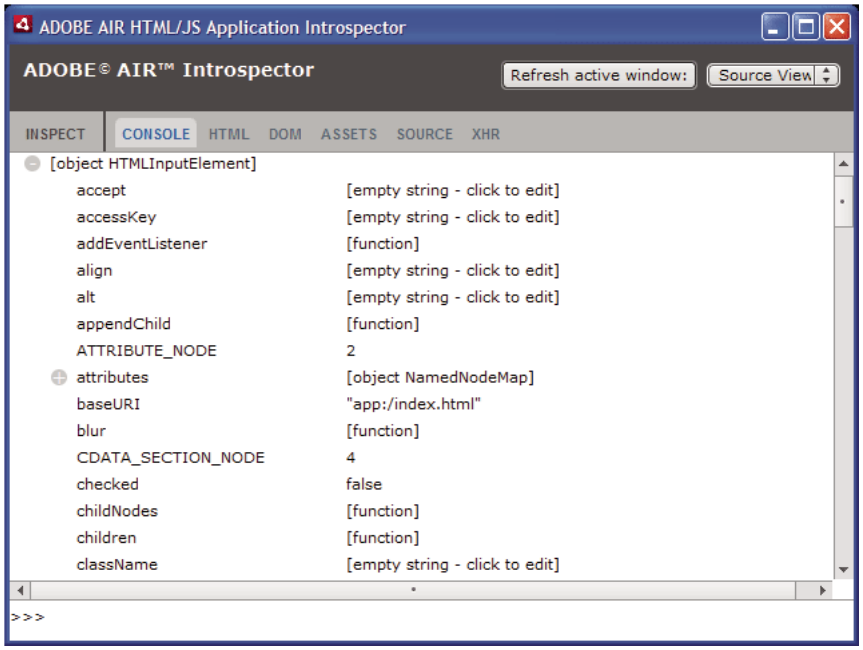
log(), warn(), info() 및 error() 메서드 모두 객체를 콘솔 탭으로 보내는 데 사용할 수 있습니다. 이러한 메서드 중 가장 기본적인 메서드는 log() 메서드입니다. 다음 코드에서는 test 변수로 표현된 단순 객체를 콘솔 탭으로 보냅니다.

```
var test = "hello";
air.Introspector.Console.log(test);
```

하지만 복합 객체를 콘솔 탭으로 보내는 것이 유용한 경우가 많습니다. 예를 들어, 다음 HTML 페이지에는 버튼 객체 자체를 콘솔 탭으로 보내는 함수를 호출한 버튼(btn1)이 포함되어 있습니다.

```
<html>
    <head>
        <title>Source Viewer Sample</title>
        <script type="text/javascript" src="scripts/AIRIntrospector.js"></script>
        <script type="text/javascript">
            function logBtn()
            {
                var button1 = document.getElementById("btn1");
                air.Introspector.Console.log(button1);
            }
        </script>
    </head>
    <body>
        <p>Click to view the button object in the Console.</p>
        <input type="button" id="btn1"
            onclick="logBtn()"
            value="Log" />
    </body>
</html>
```

이 버튼을 클릭하면 콘솔 탭에 btn1 객체가 표시되고, 객체의 트리 보기를 확장하여 속성을 검사할 수 있습니다.



속성 이름 오른쪽에 있는 목록을 클릭하고 텍스트 목록을 수정하여 객체의 속성을 편집할 수 있습니다.

info(), error() 및 warn() 메서드는 log() 메서드와 동일하지만 호출될 때 콘솔 탭의 줄 시작 부분에 다음과 같은 아이콘이 표시된다는 것이 다릅니다.

메서드	아이콘
info()	
error()	
warn()	

log(), warn(), info() 및 error() 메서드는 실제 객체에 대한 참조만 보내므로 사용할 수 있는 속성은 보는 시점의 속성입니다. 실제 객체와 직렬화하고 싶은 경우에는 dump() 메서드를 사용하십시오. 이 메서드에는 매개 변수가 두 개 있습니다.

매개 변수	설명
dumpObject	직렬화할 객체입니다.
levels	객체 트리에서 루트 수준과 함께 검사할 수준의 최대 수입니다. 기본값은 1로, 트리의 루트 수준에서 한 수준 깊은 수준이 표시됩니다. 이 매개 변수는 선택적입니다.

dump() 메서드를 호출하면 객체를 콘솔 탭으로 보내기 전에 직렬화하므로 객체의 속성을 편집할 수 없게 됩니다. 예를 들어, 다음과 같은 코드를 살펴봅니다.

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

이 코드를 실행하면 콘솔에 testObject 객체와 해당 속성이 나타나지만 콘솔에서 속성 값을 편집할 수 없습니다.

AIR Introspector 구성

AIRIntrospectorConfig 전역 변수를 설정하여 콘솔을 구성할 수 있습니다. 예를 들어, 다음 JavaScript 코드는 100자 단위로 열을 줄 바꿈하도록 AIR Introspector를 구성합니다.

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

AIRIntrospectorConfig 변수의 속성은 script 태그를 통해 AIRIntrospector.js 파일을 로드하기 전에 설정해야 합니다.

AIRIntrospectorConfig 변수에는 다음과 같은 8가지 속성이 있습니다.

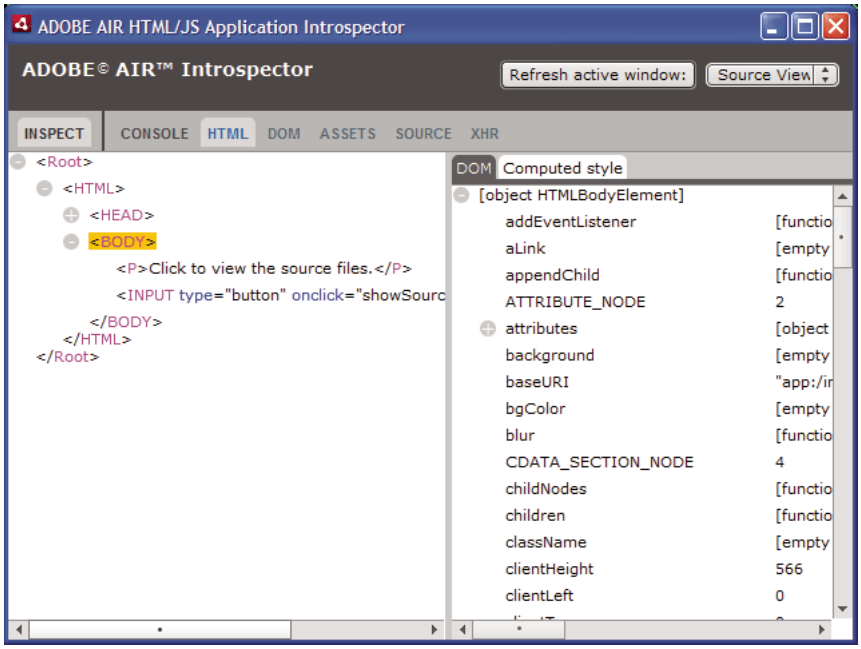
속성	기본값	설명
closeIntrospectorOnExit	true	응용 프로그램의 다른 모든 윈도우가 닫히면 관리자 윈도우가 닫히도록 설정합니다.
debuggerKey	123(F12 키)	AIR Introspector 윈도우를 표시하거나 숨기는 키보드 단축키에 대한 키 코드입니다.
debugRuntimeObjects	true	JavaScript에 정의되어 있는 객체와 함께 런타임 객체를 확장하도록 설정합니다.
flashTabLabels	true	콘솔 및 XMLHttpRequest 탭이 변경되면 깜빡이도록 설정합니다(예: 이러한 탭에서 텍스트가 기록된 경우).
introspectorKey	122(F11 키)	검사 패널을 여는 키보드 단축키에 대한 키 코드입니다.

속성	기본값	설명
showTimestamp	true	콘솔 탭에서 각 줄의 시작 부분에 타임스탬프를 표시하도록 설정합니다.
showSender	true	콘솔 탭에서 각 줄의 시작 부분에 메시지를 보내는 객체에 대한 정보를 표시하도록 설정합니다.
wrapColumns	2000	소스 파일에서 줄 바꿈을 할 열 번호입니다.

AIR Introspector 인터페이스

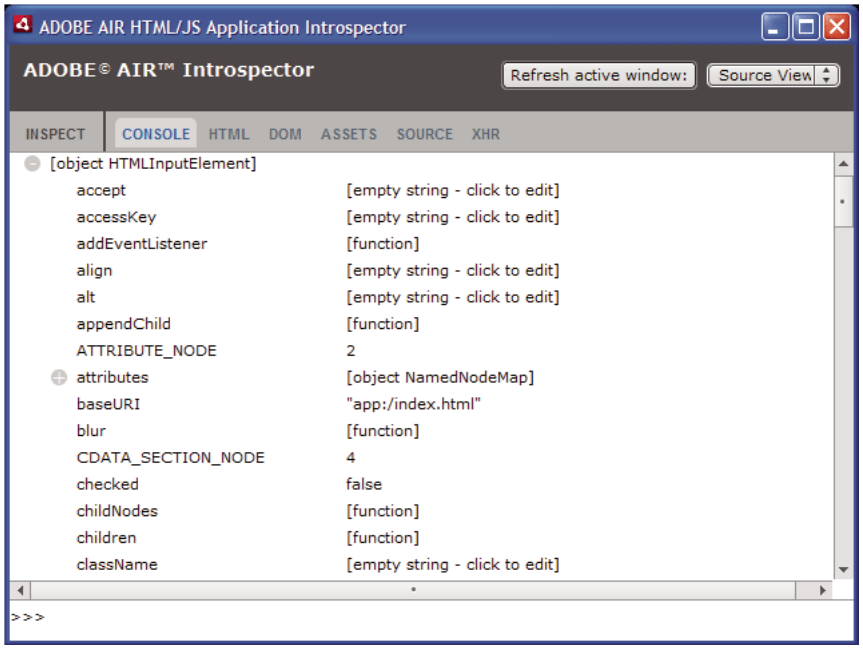
응용 프로그램을 디버깅할 때 AIR introspector 윈도우를 열려면 F12 키를 누르거나 Console 클래스의 메서드 중 하나를 호출합니다(114페이지의 “콘솔 탭에서 객체 검사” 참조). F12 키가 아닌 키를 바로 가기 키로 구성할 수 있습니다. 자세한 내용은 115 페이지의 “AIR Introspector 구성”을 참조하십시오.

다음 그림에서 볼 수 있는 것처럼 AIR Introspector 윈도우에는 콘솔, HTML, DOM, 에셋, 소스 및 XHR의 6개 탭이 있습니다.



콘솔 탭

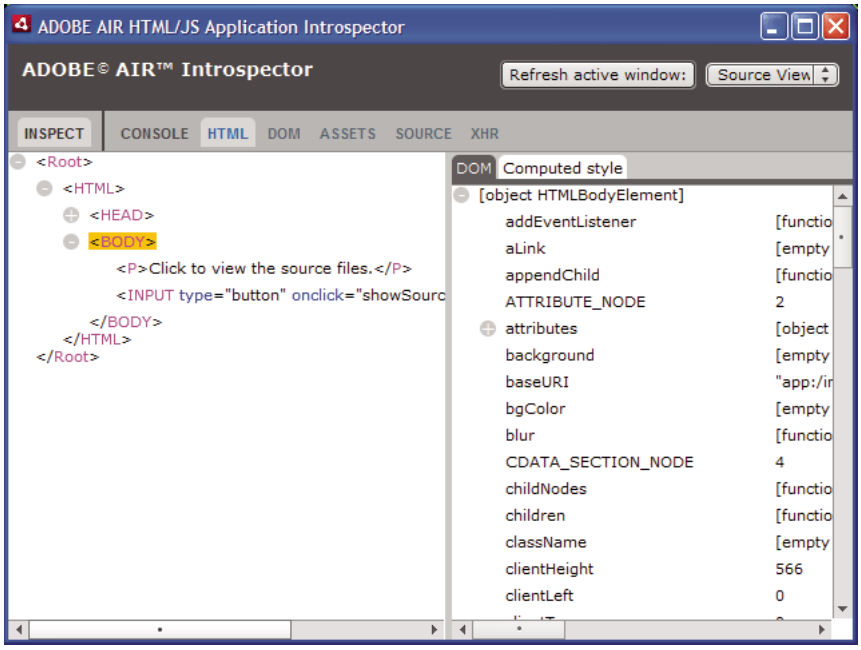
[콘솔] 탭에는 air.Introspector.Console 클래스의 메서드 중 하나에 매개 변수로 전달된 속성 값이 표시됩니다. 자세한 내용은 114페이지의 “콘솔 탭에서 객체 검사”를 참조하십시오.



- 콘솔을 지우려면 텍스트를 마우스 오른쪽 버튼으로 클릭하고 [콘솔 지우기]를 선택합니다.
- [콘솔] 탭에 있는 텍스트를 파일에 저장하려면 콘솔 탭을 마우스 오른쪽 버튼으로 클릭하고 [파일에 콘솔 저장]을 선택합니다.
- [콘솔] 탭에 있는 텍스트를 클립보드에 저장하려면 콘솔 탭을 마우스 오른쪽 버튼으로 클릭하고 [클립보드에 콘솔 저장]을 선택합니다. 선택한 텍스트를 클립보드에 복사하려면 텍스트를 마우스 오른쪽 버튼으로 클릭하고 [복사]를 선택합니다.
- Console 클래스에 있는 텍스트를 파일에 저장하려면 콘솔 탭을 마우스 오른쪽 버튼으로 클릭하고 [파일에 콘솔 저장]을 선택합니다.
- 탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

HTML 탭

[HTML] 탭을 사용하면 전체 HTML DOM을 트리 구조로 볼 수 있습니다. 요소를 클릭하면 탭 오른쪽에 해당 속성이 표시됩니다. 트리의 노드를 확장하거나 축소하려면 + 및 - 아이콘을 클릭합니다.



[HTML] 탭에서는 모든 특성과 텍스트 요소를 편집할 수 있으며, 편집한 값은 응용 프로그램에 반영됩니다.

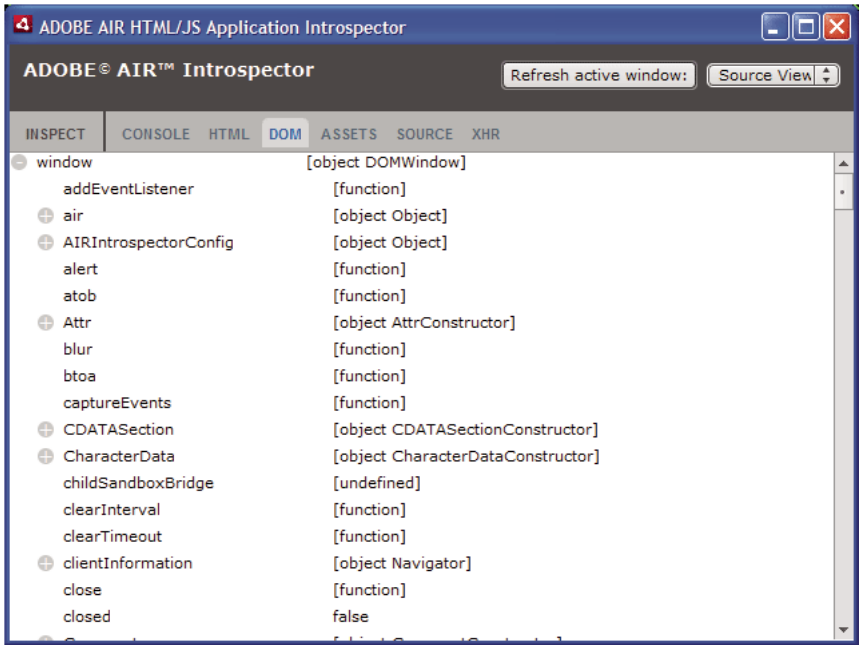
AIR Introspector 윈도우의 탭 목록 왼쪽에 있는 [검사] 버튼을 클릭합니다. 기본 윈도우의 HTML 페이지에서 원하는 요소를 클릭하면 [HTML] 탭에 연결된 DOM 객체가 표시됩니다. 기본 윈도우에 포커스가 있는 경우 키보드 단축키를 눌러 [검사] 버튼을 설정하거나 해제할 수 있습니다. 키보드 단축키는 기본적으로 F11입니다. F11 키가 아닌 키를 키보드 단축키로 구성할 수 있습니다. 자세한 내용은 115페이지의 “AIR Introspector 구성”을 참조하십시오.

[HTML] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

DOM 탭

[DOM] 탭에는 윈도우 객체가 트리 구조로 표시됩니다. 모든 문자열 및 숫자 속성을 편집할 수 있으며, 편집한 값은 응용 프로그램에 반영됩니다.

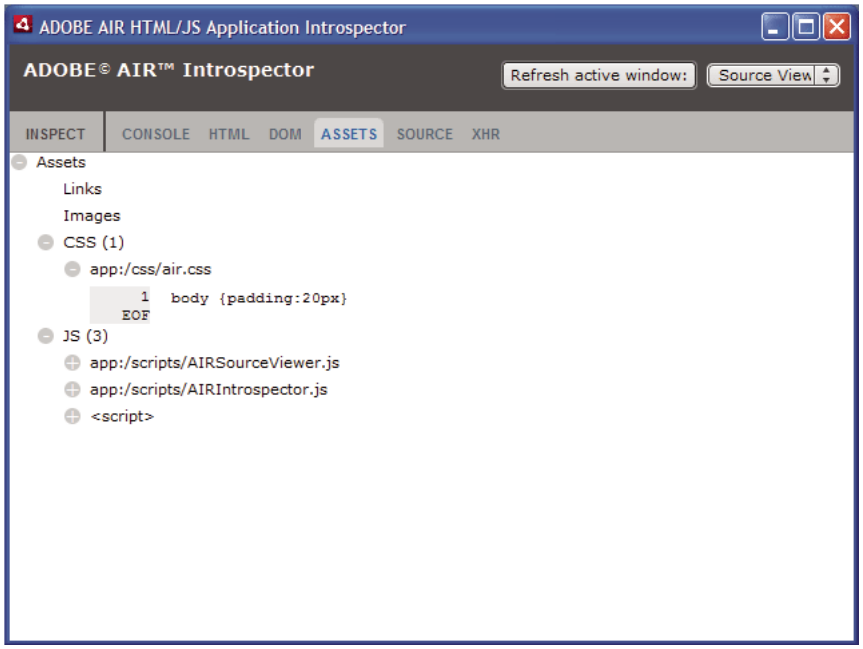


[DOM] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

에셋 탭

[에셋] 탭을 사용하면 기본 윈도우에 로드된 링크, 이미지, CSS 및 JavaScript 파일을 확인할 수 있습니다. 이러한 노드 중 하나를 확장하면 파일의 내용이나 실제 사용된 이미지가 표시됩니다.



[에셋] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

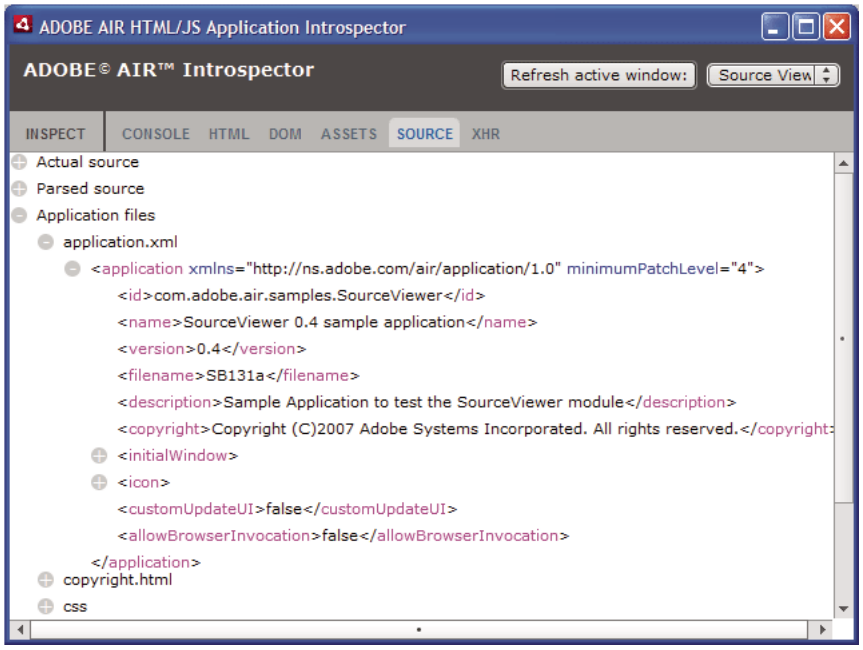
탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

소스 탭

[소스] 탭에는 섹션 세 개가 있습니다.

- 실제 소스 - 응용 프로그램을 시작할 때 루트 내용으로 로드된 페이지의 HTML 소스를 표시합니다.
- 파싱된 소스 - 응용 프로그램 UI를 구성하는 현재 마크업을 표시합니다. 응용 프로그램은 Ajax 기술을 사용하여 즉석에서 마크업 코드를 생성하므로 마크업 코드가 실제 소스와 다를 수 있습니다.

- 응용 프로그램 파일 - 응용 프로그램 디렉토리에 있는 파일을 나열합니다. 이 목록은 응용 프로그램 보안 샌드박스의 내용에서 실행된 AIR Introspector에서만 사용할 수 있습니다. 이 섹션에서 텍스트 파일의 내용이나 이미지를 볼 수 있습니다.

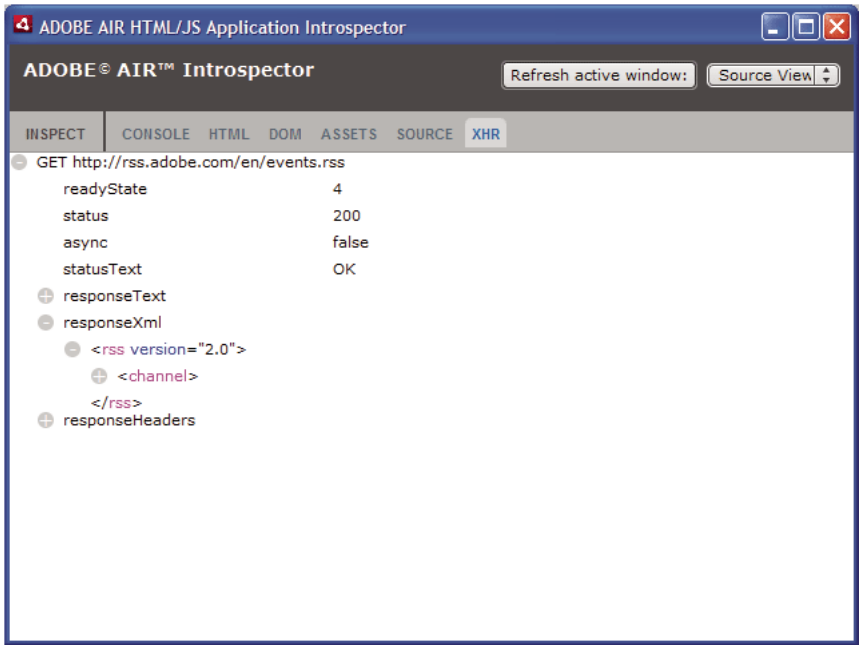


[소스] 탭에 표시된 데이터를 새로 고치려면 AIR Introspector 윈도우 맨 위에 있는 [활성 윈도우 새로 고침] 버튼을 클릭합니다.

탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

XHR 탭

[XHR] 탭은 응용 프로그램의 모든 XMLHttpRequest 통신을 가로채 정보를 기록합니다. 이 탭을 사용하면 responseText, responseXML 등을 비롯한 XMLHttpRequest 속성(사용 가능한 경우)을 트리 보기로 볼 수 있습니다.



탭에 표시된 텍스트에서 일치하는 텍스트를 검색하려면 Ctrl+F(Windows) 또는 Command+F(Mac OS)를 누릅니다. 표시되지 않은 트리 노드는 검색되지 않는다는 것에 주의하십시오.

비 응용 프로그램 샌드박스의 내용으로 AIR Introspector 사용

응용 프로그램 디렉토리의 내용을 비 응용 프로그램 샌드박스에 매핑된 iframe이나 frame으로 로드할 수 있습니다. Adobe AIR의 HTML 보안(ActionScript 개발자용) 또는 HTML security in Adobe AIR(ActionScript 개발자용)을 참조하십시오. 이러한 내용을 포함한 AIR introspector를 사용할 수 있지만 다음과 같은 규칙을 따라야 합니다.

- AIRIntrospector.js 파일이 응용 프로그램 샌드박스 내용과 비 응용 프로그램 샌드박스(iframe) 내용에 모두 포함되어야 합니다.
- parentSandboxBridge 속성은 덮어쓰지 마십시오. AIR Introspector 코드에서 이 속성을 사용합니다. 필요한 경우에는 속성을 추가합니다. 예를 들어, 다음과 같이 하는 대신

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)} ;
  ```

  다음과 같은 구문을 사용하십시오.

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- 비 응용 프로그램 샌드박스 내용에서는 F12 키를 누르거나 air.Introspector.Console 클래스의 메서드 중 하나를 호출하여 AIR Introspector를 열 수 없습니다. 이 경우 Introspector 윈도우를 열려면 [Introspector 열기] 버튼을 클릭해야 합니다. 이 버튼은 기본적으로 iframe 또는 frame의 맨 오른쪽 위에 추가됩니다. 비 응용 프로그램 샌드박스 내용에 적용되는 보안 제한 때문에 버튼 클릭과 같은 사용자 동작의 결과로만 새 윈도우를 열 수 있습니다.

- 응용 프로그램 샌드박스와 비 응용 프로그램 샌드박스 각각에 대해 별도의 AIR Introspector 윈도우를 열 수 있습니다. 두 윈도우는 AIR Introspector 윈도우에 표시된 제목으로 구분할 수 있습니다.
- AIR Introspector를 비 응용 프로그램 샌드박스에서 실행할 때는 [소스] 탭에 응용 프로그램 파일이 표시되지 않습니다.
- AIR Introspector는 자신을 연 샌드박스의 코드만 검사할 수 있습니다.

18장: AIR 응용 프로그램 지역화

Adobe AIR 1.1 이상

Adobe® AIR®에서는 여러 언어를 지원합니다.

ActionScript 3.0 및 Flex 프레임워크의 내용을 지역화하는 방법에 대한 개요는 ActionScript 3.0 개발자 안내서에서 "응용 프로그램 지역화"를 참조하십시오.

AIR에서 지원되는 언어

AIR 1.1 릴리스에는 다음 언어에 대한 AIR 응용 프로그램 지역화 지원이 포함되었습니다.

- 중국어 간체
- 중국어 번체
- 프랑스어
- 독일어
- 이탈리아어
- 일본어
- 한국어
- 포르투갈어(브라질)
- 러시아어
- 스페인어

AIR 1.5 릴리스에는 다음 언어에 대한 지원이 추가되었습니다.

- 체코어
- 네덜란드어
- 폴란드어
- 스웨덴어
- 터키어

기타 도움말 항목

Building multilingual Flex applications on Adobe AIR

Building a multilingual HTML-based application

AIR 응용 프로그램 설치 프로그램에서 응용 프로그램 이름 및 설명 지역화

Adobe AIR 1.1 이상

응용 프로그램 설명자 파일의 name 및 description 요소에 여러 언어를 지정할 수 있습니다. 예를 들어, 다음에서는 3개 언어(영어, 프랑스어 및 독일어)로 응용 프로그램 이름을 지정합니다.

```
<name>
    <text xml:lang="en">Sample 1.0</text>
    <text xml:lang="fr">Échantillon 1.0</text>
    <text xml:lang="de">Stichprobe 1.0</text>
</name>
```

각 text 요소에 대한 xml:lang 특성은 RFC4646(http://www.ietf.org/rfc/rfc4646.txt)에 정의된 언어 코드를 지정합니다.

name 요소는 AIR 설치 프로그램에서 표시하는 응용 프로그램 이름을 정의합니다. AIR 설치 프로그램에서는 운영 체제 설정에 정의된 사용자 인터페이스 언어에 가장 잘 맞는 지역화된 값을 사용합니다.

마찬가지로 응용 프로그램 설명자 파일의 description 요소에도 여러 언어 버전을 지정할 수 있습니다. 이 요소에서는 AIR 설치 프로그램에서 표시하는 설명 텍스트를 정의합니다.

이러한 설정은 AIR 설치 프로그램에서 사용할 수 있는 언어에만 적용되며, 설치되어 실행 중인 응용 프로그램에 사용할 수 있는 로캘을 정의하지는 않습니다. AIR 응용 프로그램에서는 AIR 설치 프로그램에 사용할 수 있는 언어뿐 아니라 여러 언어를 지원하는 사용자 인터페이스를 제공할 수 있습니다.

자세한 내용은 63페이지의 "응용 프로그램 설명자 파일의 속성 정의"를 참조하십시오.

기타 도움말 항목

Building multilingual Flex applications on Adobe AIR

Building a multilingual HTML-based application

AIR HTML 지역화 프레임워크를 사용하여 HTML 내용 지역화

Adobe AIR 1.1 이상

AIR 1.1 SDK에는 HTML 지역화 프레임워크가 포함되어 있습니다. AIRLocalizer.js JavaScript file 파일에서 이 프레임워크를 정의합니다. AIR SDK의 프레임워크 디렉토리에는 AIRLocalizer.js 파일이 포함되어 있습니다. 이 파일에는 여러 지역화된 버전을 지원하는 응용 프로그램을 만들 때 도움이 되는 기능을 제공하는 air.Localizer 클래스가 포함되어 있습니다.

AIR HTML 지역화 프레임워크 코드 로드

지역화 프레임워크를 사용하려면 AIRLocalizer.js 파일을 프로젝트에 복사합니다. 그런 다음 스크립트 태그를 사용하여 이 파일을 응용 프로그램의 주 HTML 파일에 포함합니다.

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

이후 JavaScript에서는 다음과 같이 air.Localizer.localizer 객체를 호출할 수 있습니다.

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

air.Localizer.localizer 객체는 지역화된 리소스를 사용 및 관리하기 위한 메서드와 속성을 정의하는 단일 객체입니다. Localizer 클래스에는 다음과 같은 메서드가 포함되어 있습니다.

메서드	설명
getFile()	특정 로캘에 대해 지정된 리소스 번들의 텍스트를 가져옵니다. 130페이지의 "특정 로캘의 리소스 가져오기"를 참조하십시오.
getLocaleChain()	로캘 체인의 언어를 반환합니다. 130페이지의 "로캘 체인 정의"를 참조하십시오.
getResourceBundle()	번들 키 및 해당 값을 객체로 반환합니다. 130페이지의 "특정 로캘의 리소스 가져오기"를 참조하십시오.

메서드	설명
getString()	리소스에 대해 정의된 문자열을 가져옵니다. 130페이지의 "특정 로캘의 리소스 가져오기"를 참조하십시오.
setBundlesDirectory()	번들 디렉토리 위치를 설정합니다. 129페이지의 "AIR HTML Localizer 설정 사용자 정의"를 참조하십시오.
setLocalAttributePrefix()	HTML DOM 요소에 사용된 localizer 특성에서 사용하는 접두어를 설정합니다. 129페이지의 "AIR HTML Localizer 설정 사용자 정의"를 참조하십시오.
setLocaleChain()	로캘 체인의 언어 순서를 설정합니다. 130페이지의 "로캘 체인 정의"를 참조하십시오.
sortLanguagesByPreference()	운영 체제 설정의 로캘 순서를 기준으로 로캘 체인의 로캘을 정렬합니다. 130페이지의 "로캘 체인 정의"를 참조하십시오.
update()	현재 로캘 체인의 지역화된 문자열을 사용하여 HTML DOM(또는 DOM 요소)을 업데이트합니다. 로캘 체인에 대한 자세한 내용은 127페이지의 "로캘 체인 관리"를 참조하십시오. update() 메서드에 대한 자세한 내용은 128페이지의 "현재 로캘을 사용하도록 DOM 요소 업데이트"를 참조하십시오.

Localizer 클래스에는 다음과 같은 정적 속성이 포함됩니다.

속성	설명
localizer	응용 프로그램의 단일 Localizer 객체에 대한 참조를 반환합니다.
ultimateFallbackLocale	응용 프로그램에서 사용자 환경 설정을 지원하지 않는 경우 사용되는 로캘입니다. 130페이지의 "로캘 체인 정의"를 참조하십시오.

리소스 번들 정의

HTML 지역화 프레임워크는 **지역화** 파일에서 지역화된 버전의 문자열을 읽습니다. 지역화 파일은 텍스트 파일로 직렬화된 키 기반 값의 컬렉션입니다. 지역화 파일은 번들이라고도 합니다.

응용 프로그램 프로젝트 디렉토리의 하위 디렉토리를 locale이라는 이름으로 만듭니다. 다른 이름을 사용할 수도 있습니다. 129페이지의 "AIR HTML Localizer 설정 사용자 정의"를 참조하십시오. 이 디렉토리에는 지역화 파일이 포함됩니다. 이 디렉토리를 **번들 디렉토리**라고 합니다.

응용 프로그램에서 지원하는 각 로캘에 대해 번들 디렉토리의 하위 디렉토리를 만듭니다. 각 하위 디렉토리의 이름을 로캘 코드와 같게 지정합니다. 예를 들어 프랑스어 디렉토리는 "fr"로, 영어 디렉토리는 "en"으로 이름을 지정합니다. 밑줄(_) 문자를 사용하여 언어 및 국가 코드가 있는 로캘을 정의할 수 있습니다. 예를 들어 영어(미국) 디렉토리는 "en_us"로 이름을 지정합니다. 또한 "en-us"와 밑줄 대신 하이픈을 사용할 수도 있습니다. HTML 지역화 프레임워크에서는 이 둘을 모두 지원합니다.

로캘 하위 디렉토리에 원하는 수의 리소스 파일을 추가할 수 있습니다. 일반적으로 각 언어에 대해 지역화 파일을 만들어 해당 응용 프로그램의 디렉토리에 저장합니다. HTML 지역화 프레임워크에는 파일의 내용을 읽는 데 사용할 수 있는 getFile() 메서드가 포함되어 있습니다(130페이지의 "특정 로캘의 리소스 가져오기" 참조).

.properties 파일 확장명을 갖는 파일을 지역화 속성 파일이라고 합니다. 이 파일을 사용하여 로캘의 키 값 쌍을 정의할 수 있습니다. 속성 파일에서는 각 행에 하나의 문자열을 정의합니다. 예를 들어 다음은 문자열 값 "Hello in English."(greeting이라는 키에 대해)를 정의합니다.

```
greeting=Hello in English.
```

다음 텍스트가 포함된 속성 파일은 6개의 키-값 쌍을 정의합니다.

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

이 예제에서는 en 디렉토리에 저장될 영어 버전의 속성 파일을 보여 줍니다.

이 속성 파일의 프랑스어 버전은 fr 디렉토리에 저장됩니다.

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

서로 다른 정보 유형에 대해 여러 리소스 파일을 정의할 수 있습니다. 예를 들어 legal.properties 파일에는 저작권 정보와 같은 공통 조항 법률 텍스트가 포함될 수 있습니다. 또한 여러 응용 프로그램에서 해당 리소스를 다시 사용할 수 있습니다. 마찬가지로 사용자 인터페이스의 서로 다른 부분에 대해 지역화된 내용을 정의하는 별도의 파일을 정의할 수 있습니다.

여러 언어를 지원하려면 이러한 파일에 UTF-8 인코딩을 사용합니다.

로캘 체인 관리

응용 프로그램에서 AIRLocalizer.js 파일을 로드할 때 응용 프로그램에 정의된 로캘을 검사합니다. 이러한 로캘은 번들 디렉토리의 하위 디렉토리에 해당합니다(126페이지의 “리소스 번들 정의” 참조). 이러한 사용 가능한 로캘 목록을 **로캘 체인**이라고 합니다. AIRLocalizer.js 파일에서는 운영 체제 설정에 정의된 기본 순서에 따라 로캘 체인을 자동으로 정렬합니다. Capabilities.languages 속성에서는 운영 체제 사용자 인터페이스 언어를 기본 순서로 나열합니다.

따라서 응용 프로그램에서 "en", "en_US" 및 "en_UK" 로캘에 대한 리소스를 정의하면 AIR HTML Localizer 프레임워크는 로캘 체인을 적절하게 정렬합니다. "en"을 기본 로캘로 보고하는 시스템에서 응용 프로그램이 시작되는 경우 로캘 체인은 ["en", "en_US", "en_UK"]로 정렬됩니다. 이 경우 응용 프로그램에서는 먼저 "en" 번들의 리소스를 찾은 다음 "en_US" 번들의 리소스를 찾습니다.

그러나 시스템에서 "en-US"를 기본 로캘로 보고하는 경우에는 ["en_US", "en", en_UK"]로 정렬됩니다. 이 경우 응용 프로그램에서는 먼저 "en_US" 번들에서, 그 다음 "en" 번들에서 리소스를 찾습니다.

기본적으로 응용 프로그램에서는 로캘 체인의 첫 번째 로캘을 사용할 기본 로캘로 정의합니다. 응용 프로그램을 처음 실행할 때 사용자에게 로캘을 선택하도록 요청할 수 있습니다. 그런 다음 선택한 내용을 환경 설정 파일에 저장하고 이후 응용 프로그램을 시작할 때 해당 로캘을 사용할 수 있습니다.

응용 프로그램에서는 로캘 체인에 있는 모든 로캘의 리소스 문자열을 사용할 수 있습니다. 특정 로캘에 리소스 문자열이 정의되어 있지 않으면 응용 프로그램은 로캘 체인에 정의된 다른 로캘에서 일치하는 다음 리소스 문자열을 사용합니다.

Localizer 객체의 setLocaleChain() 메서드를 호출하여 로캘 체인을 사용자 정의할 수 있습니다. 130페이지의 “로캘 체인 정의”를 참조하십시오.

지역화된 내용으로 DOM 요소 업데이트

응용 프로그램의 요소에서 지역화 속성 파일을 키 값을 참조할 수 있습니다. 예를 들어 다음 예제의 title 요소는 local_innerHTML 특성을 지정합니다. 지역화 프레임워크에서는 이 특성을 사용하여 지역화된 값을 조회합니다. 기본적으로 프레임워크에서는 "local_"로 시작하는 특성 이름을 찾습니다. 그리고 "local_" 다음의 텍스트와 일치하는 이름의 특성을 업데이트합니다. 이 경우 프레임워크에서는 title 요소의 innerHTML 특성을 설정합니다. innerHTML 특성은 기본 속성 파일(default.properties)의 mainWindowTitle 키에 정의된 값을 사용합니다.

```
<title local_innerHTML="default.mainWindowTitle"/>
```

현재 로캘에 일치하는 값이 정의되어 있지 않으면 localizer 프레임워크는 나머지 로캘 체인을 검색합니다. 로캘 체인에서 값이 정의된 다음 로캘을 사용합니다.

다음 예제에서 p 요소의 텍스트(innerHTML 특성)는 기본 속성 파일에 정의된 greeting 키 값을 사용합니다.

```
<p local_innerHTML="default.greeting" />
```

다음 예제에서 input 요소의 값 특성 및 표시된 텍스트는 기본 속성 파일에 정의된 btnBlue 키의 값을 사용합니다.

```
<input type="button" local_value="default.btnBlue" />
```

현재 로캘 체인에 정의된 문자열을 사용하도록 HTML DOM을 업데이트하려면 Localizer 객체의 update() 메서드를 호출합니다. update() 메서드를 호출하면 Localizer 객체가 DOM을 파싱하고 지역화("local_...") 특성을 찾는 경우 조작을 적용합니다.

```
air.Localizer.localizer.update();
```

특성(예: "innerHTML")과 해당하는 지역화 특성(예: "local_innerHTML") 모두에 대해 값을 정의할 수 있습니다. 이 경우 지역화 프레임워크는 지역화 체인에서 일치하는 값을 찾는 경우에만 특성 값을 덮어씁니다. 예를 들어 다음 요소는 value 및 local_value 특성을 모두 정의합니다.

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

특정 DOM 요소만 업데이트할 수도 있습니다. 다음 단원 128페이지의 “현재 로캘을 사용하도록 DOM 요소 업데이트”를 참조하십시오.

기본적으로 AIR HTML Localizer에서는 "local_"을 요소에 대한 지역화 설정을 정의하는 특성의 접두어로 사용합니다. 예를 들어 기본적으로 local_innerHTML 특성은 요소의 innerHTML 값에 사용되는 번들 및 리소스 이름을 정의합니다. 또한 기본적으로 local_value 특성은 요소의 value 특성에 사용되는 번들 및 리소스 이름을 정의합니다. "local_"이 아닌 다른 특성 접두어를 사용하도록 Localizer를 구성할 수 있습니다. 129페이지의 “AIR HTML Localizer 설정 사용자 정의”를 참조하십시오.

현재 로캘을 사용하도록 DOM 요소 업데이트

Localizer 객체가 HTML DOM을 업데이트하면 표시된 요소는 현재 로캘 체인에 정의된 문자열을 기준으로 특성 값을 사용하게 됩니다. HTML localizer가 HTML DOM을 업데이트하게 하려면 Localizer 객체의 update() 메서드를 호출합니다.

```
air.Localizer.localizer.update();
```

지정한 DOM 요소만 업데이트하려면 해당 요소를 update() 메서드에 매개 변수로 전달합니다. update() 메서드에는 선택 사항인 한 개의 메서드 parentNode만 있습니다. parentNode 매개 변수는 지정하는 경우 지역화할 DOM 요소를 정의합니다. update() 메서드를 호출하고 parentNode 매개 변수를 지정하면 지역화 특성을 지정하는 모든 자식 요소에 대한 지역화된 값이 설정됩니다.

예를 들어, 다음 div 요소를 살펴봅니다.

```
<div id="colorsDiv">
    <h1 local_innerHTML="default.lblColors" ></h1>
    <p><input type="button" local_value="default.btnBlue" /></p>
    <p><input type="button" local_value="default.btnRed" /></p>
    <p><input type="button" local_value="default.btnGreen" /></p>
</div>
```

현재 로캘 체인에 정의된 지역화된 문자열을 사용하도록 이 요소를 업데이트하려면 다음 JavaScript 코드를 사용합니다.

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

로캘 체인에 키 값이 없는 경우 지역화 프레임워크에서는 특성 값을 "local_" 특성의 값으로 설정합니다. 예를 들어 앞의 예제에서 지역화 프레임워크가 로캘 체인의 모든 default.properties 파일에서 lblColors 키 값을 찾을 수 없다고 가정합니다. 이 경우 프레임워크는 "default.lblColors"를 innerHTML 값으로 사용합니다. 이 값을 사용하면 개발자에게 누락된 리소스를 표시할 수 있습니다.

update() 메서드는 로캘 체인에서 리소스를 찾을 수 없는 경우 resourceNotFound 이벤트를 전달합니다. air.Localizer.RESOURCE_NOT_FOUND 상수는 문자열 "resourceNotFound"를 정의합니다. 이 이벤트에는 bundleName, resourceName 및 locale의 세 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. resourceName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

update() 메서드는 지정한 번들을 찾을 수 없는 경우 bundleNotFound 이벤트를 전달합니다. air.Localizer.BUNDLE_NOT_FOUND 상수는 문자열 "bundleNotFound"를 정의합니다. 이 이벤트에는 bundleName 및 locale의 두 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

update() 메서드는 비동기적으로 작동하며 resourceNotFound 및 bundleNotFound 이벤트를 비동기적으로 전달합니다. 다음 코드에서는 resourceNotFound 및 bundleNotFound 이벤트에 대한 이벤트 리스너를 설정합니다.

```
air.Localizer.localizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.update();
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

AIR HTML Localizer 설정 사용자 정의

Localizer 객체의 setBundlesDirectory() 메서드를 사용하면 번들 디렉토리 경로를 사용자 정의할 수 있습니다. Localizer 객체의 setLocalAttributePrefix() 메서드를 사용하면 번들 디렉토리 경로를 사용자 정의하고 Localizer에서 사용하는 특성 값을 사용자 정의할 수 있습니다.

기본 번들 디렉토리는 응용 프로그램 디렉토리의 로캘 하위 디렉토리로 정의됩니다. Localizer 객체의 setBundlesDirectory() 메서드를 호출하여 다른 디렉토리를 지정할 수 있습니다. 이 메서드에서는 하나의 매개 변수 path를 사용하며 이 매개 변수는 원하는 번들 디렉토리의 경로를 문자열로 지정합니다. path 매개 변수 값은 다음 중 하나가 될 수 있습니다.

- 응용 프로그램 디렉토리에 상대적인 경로를 정의하는 문자열(예: "locales")
- app, app-storage 또는 file URL 스킴을 사용하는 유효한 URL을 정의하는 문자열(예: "app://languages"). http URL 스킴은 사용하지 마십시오.
- File 객체

URL 및 디렉토리 경로에 대한 자세한 내용은 다음을 참조하십시오.

- File 객체의 경로(ActionScript 개발자용)
- Paths of File objects(HTML 개발자용)

예를 들어 다음 코드에서는 번들 디렉토리를 응용 프로그램 디렉토리가 아닌 응용 프로그램 저장소 디렉토리의 languages 하위 디렉토리로 설정합니다.

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

유효한 경로를 path 매개 변수로 전달합니다. 그렇지 않으면 메서드에서 BundlePathNotFoundError 예외가 발생합니다. 이 오류의 name 속성은 "BundlePathNotFoundError"이고 message 속성은 잘못된 경로를 지정합니다.

기본적으로 AIR HTML Localizer에서는 "local_"을 요소에 대한 지역화 설정을 정의하는 특성의 접두어로 사용합니다. 예를 들어 local_innerHTML 특성은 다음 input 요소의 innerHTML 값에 사용되는 번들 및 리소스 이름을 정의합니다.

```
<p local_innerHTML="default.greeting" />
```

Localizer 객체의 setLocalAttributePrefix() 메서드를 사용하면 "local_"이 아닌 특성 접두어를 사용할 수 있습니다. 이 정적 메서드에서는 특성 접두어로 사용할 문자열인 하나의 매개 변수를 사용합니다. 예를 들어 다음 코드에서는 "loc_"를 특성 접두어로 사용하도록 지역화 프레임워크를 설정합니다.

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

지역화 프레임워크에서 사용하는 특성 접두어를 사용자 정의할 수 있습니다. 기본값("local_")이 코드에 사용되는 다른 특성의 이름과 충돌하는 경우 접두어를 사용자 정의할 수 있습니다. 이 메서드를 호출할 때는 HTML 특성에 유효한 문자를 사용해야 합니다. 예를 들어 값에 공백 문자가 포함될 수 없습니다.

HTML 요소에서 지역화 특성 사용에 대한 자세한 내용은 127페이지의 “지역화된 내용으로 DOM 요소 업데이트”를 참조하십시오.

번들 디렉토리 및 특성 접두어 설정은 응용 프로그램 세션이 바뀌면 유지되지 않습니다. 사용자 정의 번들 디렉토리 또는 특성 접두어 설정을 사용하려면 응용 프로그램이 시작할 때마다 이를 설정해야 합니다.

로캘 체인 정의

기본적으로 AIRLocalizer.js 코드를 로드할 때 기본 로캘 체인이 설정됩니다. 로캘은 번들 디렉토리에서 사용할 수 있고 운영 체제 언어 설정에서 이 로캘 체인을 정의합니다. 자세한 내용은 127페이지의 “로캘 체인 관리”를 참조하십시오.

Localizer 객체의 정적 setLocaleChain() 메서드를 호출하여 로캘 체인을 수정할 수 있습니다. 예를 들어 특정 언어를 선호하는 경우 이 메서드를 호출할 수 있습니다. setLocaleChain() 메서드는 하나의 매개 변수 chain을 사용합니다. 이 매개 변수는 ["fr_FR","fr","fr_CA"]과 같은 로캘 배열입니다. 배열의 로캘 순서에 따라 후속 작업에서 프레임워크가 리소스를 찾는 순서가 결정됩니다. 체인의 첫 번째 로캘에서 리소스를 찾지 못하면 계속 다른 로캘의 리소스를 찾습니다. chain 인수가 없거나 배열이 아니거나 빈 배열이면 함수가 실패하고 IllegalArgumentsError 예외가 발생합니다.

Localizer 객체의 정적 getLocaleChain() 메서드는 현재 로캘 체인의 로캘을 나열하는 배열을 반환합니다.

다음 코드에서는 현재 로캘 체인을 읽고 체인의 머리 부분에 두 개의 프랑스어 로캘을 추가합니다.

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

setLocaleChain() 메서드는 로캘 체인을 업데이트할 때 "change" 이벤트를 전달합니다. air.Localizer.LOCALE_CHANGE 상수는 문자열 "change"를 정의합니다. 이 이벤트에서는 새 로캘 체인의 로캘 코드 배열인 localeChain이라는 하나의 속성을 사용합니다. 다음 코드에서는 이 이벤트에 대한 이벤트 리스너를 설정합니다.

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
localizer.addEventListener(air.Localizer.LOCALE_CHANGE, changeHandler);
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
function changeHandler(event)
{
    alert(event.localeChain);
}
```

정적 air.Localizer.ultimateFallbackLocale 속성은 응용 프로그램이 사용자 환경 설정을 지원하지 않는 경우 사용되는 로캘을 나타냅니다. 기본값은 "en"입니다. 다음 코드와 같이 이를 다른 로캘로 설정할 수 있습니다.

```
air.Localizer.ultimateFallbackLocale = "fr";
```

특정 로캘의 리소스 가져오기

Localizer 객체의 getString() 메서드는 특정 로캘의 리소스에 대해 정의된 문자열을 반환합니다. 이 메서드를 호출할 때 locale 값을 지정할 필요가 없습니다. 이 경우 메서드에서는 전체 로캘 체인을 확인하고 지정된 리소스 이름을 제공하는 첫 번째 로캘의 문자열을 반환합니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
bundleName	리소스를 포함하는 번들입니다. .properties 확장명을 제외한 속성 파일의 파일 이름입니다. 예를 들어 이 매개 변수를 "alerts"로 설정하면 Localizer 코드에서는 alerts.properties라는 지역화 파일을 확인합니다.
resourceName	리소스 이름입니다.
templateArgs	선택 사항입니다. 대체 문자열에서 번호 지정된 태그를 대체할 문자열 배열입니다. 예를 들어 templateArgs 매개 변수가 ["Raúl", "4"]이고 일치하는 리소스 문자열이 "Hello, {0}. You have {1} new messages."인 함수를 호출한다고 가정합니다. 이 경우 함수는 "Hello, Raúl. You have 4 new messages."를 반환합니다. 이 설정을 무시하려면 null 값을 전달합니다.
locale	선택 사항입니다. 사용할 로캘 코드입니다(예: "en", "en_us" 또는 "fr"). 로캘을 지정하고 일치하는 값이 없으면 메서드에서는 로캘 체인의 다른 로캘에서 값을 검색하지 않습니다. 로캘 코드를 지정하지 않으면 함수에서는 지정된 리소스 이름에 대한 값을 제공하는 로캘 체인의 첫 번째 로컬에 있는 문자열을 반환합니다.

지역화 프레임워크에서 표시된 HTML DOM 특성을 업데이트할 수 있습니다. 그러나 지역화된 문자열을 다른 방법으로 사용할 수 있습니다. 예를 들어 문자열을 동적으로 생성된 HTML에서 사용하거나 함수 호출의 매개 변수로 사용할 수 있습니다. 예를 들어 다음 코드에서는 fr_FR 로캘의 기본 속성 파일에서 error114 리소스에 정의된 문자열을 사용하여 alert() 함수를 호출합니다.

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

getString() 메서드는 지정된 번들에서 리소스를 찾을 수 없는 경우 resourceNotFound 이벤트를 전달합니다. air.Localizer.RESOURCE_NOT_FOUND 상수는 문자열 "resourceNotFound"를 정의합니다. 이 이벤트에는 bundleName, resourceName 및 locale의 세 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. resourceName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

getString() 메서드는 지정된 번들을 찾을 수 없는 경우 bundleNotFound 이벤트를 전달합니다. air.Localizer.BUNDLE_NOT_FOUND 상수는 문자열 "bundleNotFound"를 정의합니다. 이 이벤트에는 bundleName 및 locale의 두 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

getString() 메서드는 비동기적으로 작동하며 resourceNotFound 및 resourceNotFound 이벤트를 비동기적으로 전달합니다. 다음 코드에서는 resourceNotFound 및 bundleNotFound 이벤트에 대한 이벤트 리스너를 설정합니다.

```
air.Localizerlocalizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizerlocalizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, bnfHandler);
var str = air.Localizer.localizer.getString("default", "error114", null, "fr_FR");
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

Localizer 객체의 getResourceBundle() 메서드는 제공된 로캘에 대해 지정된 번들을 반환합니다. 이 메서드의 반환 값은 번들의 키와 일치하는 속성이 있는 객체입니다. 응용 프로그램에서 지정된 번들을 찾을 수 없는 경우 이 메서드는 null을 반환합니다.

이 메서드는 locale 및 bundleName이라는 두 가지 매개 변수를 사용합니다.

매개 변수	설명
locale	로캘(예: "fr")입니다.
bundleName	번들 이름입니다.

예를 들어 다음 코드에서는 document.write() 메서드를 호출하여 fr 로캘의 기본 번들을 로드합니다. 그런 다음 document.write() 메서드를 호출하여 해당 번들에 있는 str1 및 str2 키의 값을 작성합니다.

```
var aboutWin = window.open();
var bundle = localizer.getResourceBundle("fr", "default");
aboutWin.document.write(bundle.str1);
aboutWin.document.write("<br/>");
aboutWin.document.write(bundle.str2);
aboutWin.document.write("<br/>");
```

getResourceBundle() 메서드는 지정된 번들을 찾을 수 없는 경우 bundleNotFound 이벤트를 전달합니다. air.Localizer.BUNDLE_NOT_FOUND 상수는 문자열 "bundleNotFound"를 정의합니다. 이 이벤트에는 bundleName 및 locale의 두 가지 속성이 있습니다. bundleName 속성은 리소스가 없는 번들의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다.

Localizer 객체의 getFile() 메서드는 지정된 로캘에 대해 번들의 내용을 문자열로 반환합니다. 번들 파일은 UTF-8 파일로 읽습니다. 이 메서드의 매개 변수는 다음과 같습니다.

매개 변수	설명
resourceFileName	리소스 파일의 파일 이름입니다(예: "about.html").

templateArgs	선택 사항입니다. 대체 문자열에서 번호 지정된 태그를 대체할 문자열 배열입니다. 예를 들어 templateArgs 매개 변수가 ["Raúl", "4"]이고 일치하는 리소스 파일에 두 줄이 포함된 함수를 호출한다고 가정합니다.
	
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	
	이 경우 함수는 다음과 같이 두 줄이 포함된 문자열을 반환합니다.
	
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

locale	사용할 로캘 코드입니다(예: "en_GB"). 로캘을 지정하고 일치하는 파일이 없으면 메서드에서는 로캘 체인의 다른 로캘에서 검색을 계속하지 않습니다. 로캘 코드를 지정하지 **않으면** 함수에서는 resourceFileName과 일치하는 파일이 있는 로캘 체인의 첫 번째 로캘에 있는 텍스트를 반환합니다.

예를 들어 다음 코드에서는 fr 로캘의 about.html 파일의 내용을 사용하여 document.write() 메서드를 호출합니다.

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

getFile() 메서드는 로캘 체인에서 리소스를 찾을 수 없는 경우 fileNotFound 이벤트를 전달합니다. air.Localizer.FILE_NOT_FOUND 상수는 문자열 "resourceNotFound"를 정의합니다. getFile() 메서드는 비동기적으로 작동하며 fileNotFound 이벤트를 비동기적으로 전달합니다. 이 이벤트에는 fileName 및 locale의 두 가지 속성이 있습니다. fileName 속성은 찾지 못한 파일의 이름입니다. locale 속성은 리소스가 없는 로캘의 이름입니다. 다음 코드에서는 이 이벤트에 대한 이벤트 리스너를 설정합니다.

```
air.Localizer.localizer.addEventListener(air.Localizer.FILE_NOT_FOUND, fnfHandler);
air.Localizer.localizer.getFile("missing.html", null, "fr");
function fnfHandler(event)
{
    alert(event.fileName + ": " + event.locale);
}
```

기타 도움말 항목

Building a multilingual HTML-based application