Introduction to Adobe AIR

Alin-Cristian JOIȚA
Faculty of Computer Science for Business Management,
Romanian – American University, Bucharest, Romania

ABSTRACT

Rich Internet applications *(RIA) are web applications that have the features and functionality of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data (i.e. maintaining the state of the program, the data etc) back on the application server.*

AIR *(Adobe Integrated Runtime), which was originally code-named Apollo, is a cross-operating system runtime, that allows developers to build and deploy rich Internet applications to the desktop using their existing skill sets. AIR applications can be built using HTML, AJAX, Adobe Flash, and Adobe Flex. The version 1.0 release supports both Mac OSX and WindowsXP and Vista. Adobe has also confirmed that a Linux version of the runtime is on its radar.*

1. INTRODUCTION

What exactly does this AIR mean to developers?

First and foremost, it means that if your current skill set includes HTML/JavaScript (AJAX), Flex, or Flash, you already have everything you need to create a desktop application that will install on a Windows or Mac computer. Therefore, you don't need to know Java, C++, C, or any of the other traditional desktop languages to create and distribute full-fledged desktop applications.

It also means that, since Adobe has created these runtimes, Adobe is the one responsible for ensuring that your application performs the same on any of the operating systems that AIR supports. If you are coming from a Flash or Flex background, you already know how nice it is to write an application that performs the same within the Flash player in a traditional web browser. If you are coming from an HTML/JavaScript background, you have undoubtedly suffered through countless frustrations and hacks to get your web page to show up the same in many different browsers. You will suffer no longer when your HTML/JavaScript application is deployed as an AIR application, since it will be running within the Adobe Integrated Runtime.

Online versus Desktop Applications

The traditional definition of an *online application* is one that runs within a web browser while connected to the Internet. A *desktop application* has traditionally been one that runs on the local computer whether there is an Internet connection or not. New programming models, like AIR have begun to blend these ideas and create hybrid applications in which some of the data are stored locally, while additional data may be loaded into the application when an Internet connection exists. Or the application can synchronize its data or files when an Internet connection exists for use later when an Internet connection no longer exists. Google via its Gears API has also begun to create browser-based applications that can cache data within an embedded database for offline use.

There is no doubt that the convergence of online and desktop applications into occasionally connected applications will continue. With tools such as AIR, it is easier than ever to create applications that can perform well whether an Internet connection exists or not.

2. THE RUNTIME ENVIRONMENT

The runtime environment is what guarantees the consistent experience across different operating systems and the versions of each. For example, there is an API within an AIR application that handles the creation of a new file or folder. The AIR developer writes the application using this API. The runtime then translates this request into an operating-system-specific call, ensuring that the file or folder is created in the native way that the operating system normally handles the creation of a file or folder.

The runtime itself is distributed either as a stand-alone install package catered to each operating system, or it can be packaged and distributed along with an AIR application. Once the runtime exists on a user's machine, it will handle the responsibility of installing new AIR applications as well as maintaining a version history of itself, which is fully backward-compatible. The backward compatibility will allow developers to build applications that target a specific release of the runtime but also ensure that a future runtime release doesn't break an existing application.

A few of the unique features of the Adobe Integrated Runtime include an integrated web browser based on the Safari WebKit as well as an integrated database based on the SQLite embedded database distribution.

3. THE AIR FILE

The .air file extension is a new file extension from Adobe that's used to signify an application built to run on the Adobe Integrated Runtime. When a developer creates a new AIR application, it is compiled to an AIR package with the .air extension, for example, HelloWorld.air. When a user downloads the HelloWorld.air package, the runtime recognizes this as an installer package and will then install the application to the operating system. The application itself will either be an executable .exe file extension on Windows or an .app file extension on Mac. Once the application is installed, the original HelloWorld.air file is no longer needed and can be deleted, as it is needed for distribution and installation.

4. THE TOOLS

Adobe has made it possible to create and distribute AIR applications with absolutely no cost at all. It again (as it did with Flex 2) offers a free SDK that can be used to package AIR applications using a command window with the ADT library. The SDK also allows the compilation of the application to a temporary file for testing purposes. The temporary file is created using the ADL library and runs the application without the need of installing it to the operating system.

Flex Builder has also been updated to version 3 and now includes the AIR tools. Flex Builder 3 makes it easy to create, test, debug, and package AIR applications. Adobe has also released extensions for both Dreamweaver CS3 and Flash CS3 to integrate the creation and testing of AIR applications within these tools.

5. THE AIR SDK

The AIR SDK is a free library offered by Adobe, which allows you to test an AIR application using the adl command and also compile the application to a distributable AIR package using the adt command. There is no GUI (graphical user interface) offered, although it is not very difficult to set up an ant task to call these commands from an editing tool like Eclipse.

- **ADL**

  The adl command is part of the free AIR SDK and allows for the testing of AIR applications without the need to package and install the AIR application. After navigating to the directory that contains your application through a terminal window, the sample syntax for adl would look like this:

adl HelloWorld-app.xml

- **ADT**

  The adt command is also part of the free AIR SDK and allows for the compiling and packaging of an AIR application to an AIR package for distribution and installation. After navigating to the directory that contains your application through a terminal window, the sample syntax for adt would look like this:

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

6. SECURITY

A desktop application has certain characteristics. On the one hand, desktop applications generally have a lot more privileges than a similar web application, as they have been installed by the user to a specific desktop machine, implying a degree of trust that is greater than that of arbitrary web content. On the other hand, the privileges inherent in a desktop application require a greater degree of caution as certain coding practices and patterns that may be common in web applications may never be acceptable in a desktop application.

Regardless of whether an application is built primarily in Flash or HTML, all AIR applications have some characteristics in common. Within a given AIR application, there is a set of AIR specific APIs that are available to provide access to local system and network resources that would not be normally available in a web application contained in a browser. Each AIR application also contains a number of different sandboxes, depending on what type of content is being loaded, and for what purpose:

- **Application sandbox** is the root of every AIR application. This sandbox permits access to the privileged AIR specific system APIs. In return for access to these powerful APIs, some common dangerous APIs and patterns are restricted. For example, dynamic importing of remote content is generally prohibited and dynamic code generation techniques are heavily restricted. Only content loaded directly from the application home directory (via the app:/ URI scheme) can be placed in the application sandbox.
- **Non-application sandbox** contains all other content that is not loaded directly into the application sandbox. This includes local and remote content. Such content does not have direct access to AIR APIs and obeys the same rules that it would have to obey in the browser when loaded from the same location (for example, a local SWF file behaves the same way a local SWF file would in the browser, and HTML from a remote domain behaves like it would behave in the browser).

Due to the restrictions placed upon dynamic coding and script importing, the application sandbox is generally the safest sandbox to place your application code into as the risk from injection attacks is greatly diminished compared to the typical browser sandbox. However, there may be cases where developers still need to use these risky techniques in their applications—for example, when interacting with web services that only support JSON non-compliant JavaScript APIs.

The recommended technique in these cases is to create a non-application sandbox to perform the risky operations, and then interact with that sandbox via the SandboxBridge API. The SandboxBridge is a bi-directional serialization API designed to allow domains/sandboxes that otherwise cannot trust each other entirely to interact.

7. SIGNING OF AIR APPLICATIONS

All AIR applications must be signed by a code-signing certificate. The only question is

whether the certificate in question is what is commonly known as a *self-signed* certificate, which means that it is not recognized as being trustworthy by a typical user's machine (unless the user chooses to import that specific certificate into his or her certificate trust store), or a commercial code-signing certificate purchased from a major certification authority (CA).

The recommended approach is to use a commercially obtained code-signing certificate, as that will be recognized by the AIR installer on almost all user machines. This permits the name of the publisher to be recognized and provides a better installation experience for the user.

REFERENCE

[1] Rich Tretola, **"Beginning Adobe® AIR™, Building Applications for the Adobe Integrated Runtime"**, Wiley Publishing, 2008

[2] Adobe devNet http://www.adobe.com/devnet/air/articles/introduction_to_air_security.html/

[3]. Wikipedia http://en.wikipedia.org/wiki/Adobe_Air/