Introduction to Flex and Air

CS174

Chris Pollett

Nov. 24, 2008.

Outline

- Flash, Flex, and Adobe Air
- Installation
- Simple Program
- Components

Flash

- Flash was originally a product to make it easy to produce vector based animations on the web created around 1996 based on technology purchased by Macromedia (now Adobe) from Futurewave.
- Flash programs are typically executed by the Flash player plug-in which is (Sep 2008) available in some browser on 99% of desktops (Flash >7, for Flash 9 is 98%).
- It is used to create Rich Internet Applications (RIA) and competes against AJAX, Silverlight (Windows), Applets (Java), etc.
- It can also be used to do tracking (PIE -- Persistent Identification Elements -- not turned off as often as cookies)
- It has the advantage over AJAX that as the plug-in is produced by one source, it runs the same in any browser. It also supports richer kinds of media like Video. (YouTube).

Flex

- The main authoring tool for Flash is Flash Professional, which is GUI IDE designed to make it easy to create Flash.
- Flex is an Adobe product which gives a more programmatic way to create Flash files.
- There is a Flex IDE, Flex Builder, which can also be used, however, the basic SDK is open sourced under the Mozilla License (can mix proprietary and open source code).
- We will only look at the basic SDK which we will access mainly from the command line.
- There are other open-source tool chains which can be used to create .swf files, notably, OpenLaszlo.

Adobe Air

- Air is a product released by Adobe in Feb, 2008 to make it easy to convert Flash programs into stand-alone applications.
- Since Flash is cross-platform, these apps are as well, and it makes it relatively easy to write code which will work on PCs, Mac's, and Linux.
- Air apps can do more things then Web Flash apps. For instance, you can access the file system, drag and drop from the OS to your app, etc.

Installation

- Currently (Nov 2008), the release version of Flex is Flex 3.2.
- It can be found at:

  http://www.adobe.com/products/flex/flexdownloads/
- Download this file and unzip it somewhere you know, say C:\FlexSDK (Windows) or /usr/local/FlexSDK (*nix including Mac)
- Then add C:\FlexSDK\bin or /usr/local/FlexSDK /bin to your PATH variable (use control panel Windows, or edit .profile, .cshrc, etc on *nix).
- Restart your shell.
- The programs to compile AIR apps come with Flex 3.2; however, to test it you still need the AIR runtime:

  http://get.adobe.com/air/

Creating a simple program

- Creating a Flex App involves writing a mxml file which is a XML language which may be scripted (as Javascript can be used to script HTML) using Adobe's Actionscript language (a variant of Javascript).
- You then compile this code using the mxmlc compiler.
- To include the compiled code into HTML file you need to use an object/embed tag

A simple MXML file (test.mxml)

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%" >
   <mx:Button label="Hello World
    Button"></mx:Button>
</mx:Application>
```

This could then be compiled by typing:

```
mxmlc test.mxml
```

This should produce a test.swf file.

Including your .swf file in a web page

- To include your program in a web page you need to an object tag.
- For instance, you could use:

```
<html>
<body>
 <object type="application/x-shockwave-flash" data="test.swf" width="320"
    height="240">
   <param name="src" value="test.swf" />
   <!-- name="movie" works as well -->
  </object>
</body>
</html>
```

Making an Air App

- To make an Air App out of the last example first we change the Application tag to WindowedApplication:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication
   xmlns:mx="http://www.adobe.com/2006/mxml"
   width="100%" height="100%" >
  <mx:Button label="Hello World
   Button"></mx:Button>
</mx:WindowedApplication>
```

- Then we compile it using amxmlc rather than mxmlc:

  amxmlc test.mxml

Application Descriptor File

- In order to package our swf file as an app we need to create a basic application descriptor file.
- If our swf file is name.swf, typically, this will be called name-app.xml.

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<application xmlns="http://ns.adobe.com/air/application/1.5">
  <id>org.pollett.CS174Fall2008.HelloWorld</id>
<filename>test2</filename>
  <version>1.0</version>
  <initialWindow>
     <title>Hello World Example</title>
     <content>test2.swf</content>
     <visible>true</visible>
  </initialWindow>
</application>
```

- id -just needs to be unique to tell if our app is already installed.
- To run the app using the debug launcher one could then type:

```
adl test2-app.xml
```

Creating and Signing Your App

- As part of the app creation process you need to get your app signed.
- Usually, this would be by a trusted third-party (read cost you $$$) such as Thawte or Verisign.
- This signature is checked when the app is actually installed.
- For the purposes of this class I will show you how to create a self-signed app.
- First you need to create a self-signed cert that meets RSA's pkcs12 standards:

adt -certificate -cn ChrisPollett 1024-RSA mytest.p12 secret

- secret here is the password being used.
- Next we compile our app using the .p12 file:

adt -package -storetype pkcs12 -keystore mytest.p12 \ test2.air test2-app.xml test2.swf

- Clicking on the .air file produced will install the app (at least on a MAC).

Components

- Flex applications are built out of components.
- There are visual components such as Containers and user interface controls
- There are non-visual components such as data components and utility components.
- Let’s take a look at these.

Containers and UI controls

- A container is a component that can be used to hold other components and might be used for layout purposes:

```
<mx:VBox>
    <mx:Button label="Button 1" />
    <mx:Button label="Button 2" />
</mx:VBox>
```

  There is also an HBox tag, tags for gridlayout, tiles, etc.

  Layouts can be nested.
- UI Controls. We already seen Button. There are many others such as ComboBox, Text, TextInput, etc.
- Properties of a Control can be set using either: tag attributes, nested tags, or ActionScript.
- You often give tags id attributes. The UI control object can then be used elsewhere as a variable (data binding) provided the id name is is put in {}.
- See next slide for an examples.

Example 2

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"  width="100%"
    height="100%" >
   <mx:Vbox>
   <mx:HBox>
      <mx:Button> <mx:label>Button 1</mx:label></mx:Button>
      <mx:Button label="Button 2" />
   </mx:HBox>
   <mx:ComboBox>
      <mx:dataProvider>
         <mx:ArrayCollection>
            <mx:String>Item 1</mx:String>
            <mx:String>Item 2</mx:String>
            <mx:String>Item 3</mx:String>
         </mx:ArrayCollection>
      </mx:dataProvider>
   </mx:ComboBox>
   <mx:Text text="hello" width="100" height="15" />
    <mx:Text text="{input.text}" width="150" height="20" />
   <mx:TextInput id="input" />
</mx:VBox>
</mx:Application>
```