FlexJS Developer Setup

History

- 1/11/2013 - MXML-based project can now be compiled into a working HTML/JS project
- 2/10/2013 - Added HTTPService, LazyCollection, and a List class.
- 3/01/2013 - Added Adobe Flash Builder integration.
- 4/23/2013 - Updated to use FalconJX instead of FalconJS.
- 6/27/2013 - Added the beginnings of MXML states and data binding.
- 7/22/2015 - Updated to latest process.
- 8/24/2015 - Updated with pre-requisites and automated script info
- 4/02/2016 - No need for linters and hinters

Set Up

This is the setup for developers working with the source code. If you only want to use Flash Builder to build applications with the FlexJS framework, see: Using FlexJS with Adobe Flash Builder, for IntelliJ, see Using FlexJS with IntelliJ IDEA.

First you will need the following prerequisites:

A. Java SE JDK 1.6 or later

Some computers come with Java installed, but not only do you need a Java runtime (JRE) you need a Java Development Kit (JDK). To find out if you have a JDK installed run

```
javac -version
```

The Java Compiler (javac) should print out its version if its already installed. If not, download and install a JDK from http://www.oracle.com/technetwork/java/javase/downloads/index.html. Then:

- Add the "bin" folder from wherever you installed it to your environment's PATH variable.
- Create a new JAVA_HOME environment variable that points to the folder containing unzipped files.

B. Apache Ant

- Get Apache Ant from https://ant.apache.org/bindownload.cgi. Unzip it somewhere.
- Add the "bin" folder from wherever you install it to your environment's PATH variable.
- Create a new ANT_HOME environment variable that points to the folder containing the unzipped files.

C. Adobe AIR SDK

- Get a recent Adobe AIR SDK for Windows or Mac. Unzip it somewhere. Note that there are two types of AIR SDKs for each platform and the default type does not work well with Flex so use the links provided or make sure you select the Flex-compatible versions on the Adobe site.
- Create a new AIR_HOME environment variable that points to the folder containing the unzipped files.

D. Adobe Flash Player playerglobal.swc

- Get a recent playerglobal.swc file from the Adobe Flash Player Downloads page: https://www.adobe.com/support/flashplayer/downloads.html. You will have to scroll down a bit to find the link. Here, for example, is a link to the playerglobal.swc for Adobe Flash Player 18.0. The filename will include the version number (e.g. playerglobal18_0.swc). Rename the file to playerglobal.swc and place it in a folder containing the version number (e.g. 18.0/playerglobal.swc on Mac and 18.0\playerglobal.swc on Windows). For example, many people create a /Users/myusername/adobe/flashplayer folder on Mac and c:\adobe\flashplayer folder on Windows, and store a playerglobal.swc in /Users/myusername/adobe/flashplayer/18.0/playerglobal.swc on Mac and/or c:\adobe\flashplayer\18.0\playerglobal.swc on Windows.
- Create a PLAYERGLOBAL_HOME environment variable that points to the folder containing the folder with the version number. In the example above, it would be /Users/myusername/adobe/flashplayer on Mac and/or c:\adobe\flashplayer on Windows.

E. Adobe Flash Player Projector Content Debugger

- Get a recent Projector Content Debugger (aka Standalone Debug Player) for your platform from the Adobe Flash Player Downloads page: https://www.adobe.com/support/flashplayer/downloads.html.
- Create a FLASHPLAYER_DEBUGGER environment variable that points to the Debugger. For example, on Mac, if you store the Debugger in "/Users/myusername/adobe/flashplayer/18.0/Flash Player Debugger.app", you would set the environment variable to: "/Users/myusername/adobe/flashplayer/18.0/Flash Player Debugger.app/Contents/MacOS/Flash Player Debugger" (without quotes) and on Windows, if you store the Debugger in c:\adobe\flashplayer\18.0\flashplayer_18_sa_debug.exe, you would set the environment variable to c:\adobe\flashplayer\18.0\flashplayer_18_sa_debug.exe.

F. Git

Git is a source code management application. Some computers come with Git installed. To find out if you have Git installed run

```
git --version
```

Git should print out its version if it is already installed. If not, install Git for your platform from here: http://git-scm.com/downloads. Many folks prefer GUI versions of Git, which is fine as long as there is a command-line version as well. Add Git to your PATH environment variable.

Downloading and Building the Source

There are two ways to build the source on a computer that has never had the source before.

If you have installed FlexJS per the instructions in Using FlexJS with Adobe Flash Builder, you can just go to the folder where you installed FlexJS and run "ant". The script should ask you for a folder to store all of the source code and then download (clone) all of the repos, ask you to approve up to 4 licenses, then build all of the source in the repos (except the source in the FlexJS folder). Once the script completes, it will tell you to change to the FlexJS source folder and run "ant" again there. Below is the steps it is performing.

Another way is to create an empty folder, change the current directory to that folder, then get the ASJS code from Git as follows:

```
git clone https://git-wip-us.apache.org/repos/asf/flex-asjs flex-asjs
cd flex-asjs
git checkout develop
```

Then run "ant all". Either method is performing all the steps below. If you are using one of the methods above skip past this section to "After The Build Completes".

1. Get the ASJS code from Apache Git via

   ```
   git clone https://git-wip-us.apache.org/repos/asf/flex-asjs flex-asjs
   cd flex-asjs
   git checkout develop
   ```

2. Get the Falcon code from Apache Git at

   ```
   git clone https://git-wip-us.apache.org/repos/asf/flex-falcon.git flex-falcon
   cd flex-falcon
   git checkout develop
   ```

3. Get the FlexUnit code from Apache Git at

   ```
   git clone https://git-wip-us.apache.org/repos/asf/flex-flexunit.git flex-flexunit
   cd flex-flexunit
   git checkout develop
   ```

4. Get the Flex SDK code from Apache Git at

   ```
   git clone https://git-wip-us.apache.org/repos/asf/flex-sdk.git flex-sdk
   cd flex-sdk
   git checkout develop
   ```

5. Get the TLF code from Apache Git at

   ```
   git clone https://git-wip-us.apache.org/repos/asf/flex-tlf.git tlf
   cd tlf
   git checkout develop
   ```

6. Get the BlazeDS code from Apache Git at

```
git clone https://git-wip-us.apache.org/repos/asf/flex-blazeds.git flex-blazeds
cd flex-blazeds

git checkout develop
```

7. In the flex-sdk folder, run 'ant' to build the sdk.
8. In the flex-falcon folder:
    a. In the compiler folder run 'ant all' to build the compiler and extern swcs.
    b. In the compiler.jx folder, run 'ant' to build the cross-compiler
9. In the flex-flexunit folder, run ant to build flexunit.
10. In the flex-asjs folder, run 'ant' to build the FlexJS libraries.

After The Build Completes

1. In the flex-asjs folder, change to examples/flexjs/DataBindingExample and run 'ant' there. It should create bin-debug/DataBindingExample and bin/js-debug and bin/js-release folders containing a debug SWF, a debug HTML/JS/CSS version and a minified HTML/JS/CSS version respectively. Open the .html file in those folders in your browser.

Using Eclipse

We are using Eclipse Helios for Java 1.6 compatibility because at least one of the jars (flex-oem-compiler) needs to be 1.6 compatible to work with Flash Builder. There are eclipse project files in the flex-falcon repo. The run/debug configuration for building a FlexJS SWF in Eclipse needs the following settings (Assuming the path to the flex-asjs repo is /Users/myusername/git/apache/flex/flex-asjs):

Program Arguments:

```
+flexlib="/Users/myusername/git/apache/flex/flex-asjs/frameworks" -debug /Users/myusername/git/apache/flex/flex-asjs/examples/flexjs/DataBindingExample/src/DataBindingExample.mxml
```

VM Arguments:

```
-Xmx384m -Dsun.io.useCanonCaches=false -Dflexcompiler="/Users/myusername/git/apache/flex/flex-falcon/compiler" -Dflexlib="/Users/myusername/git/apache/flex/flex-asjs/frameworks"
```

And the environment variables:
PLAYERGLOBAL_HOME are points to the folder containing versions of playerglobal.swc as specified above. Thus, if you followed the instructions above on a Mac, you would set the variable to: /Users/myusername/adobe/flashplayer

Using Adobe Flash Builder

Each of the SWCs has a set of Flash Builder project files. Actually two sets. In the frameworks/projects/Core folder are the Flash Builder project files for compiling with the compiler settings required for the version of the framework that runs in Flash, and in frameworks/js/FlexJS/projects/CoreJS are another set of Flash Builder project files for compiling with the compiler settings required for cross-compiling the same source in frameworks/projects/Core, and for building the "JS" version of the SWC. For more information on what a "JS" SWC is, click here.

It is recommended to use a new empty Workspace. You will have to specify two "Linked Resources" in Flash Builder's Preferences (under Workspace). Create a FLEX_ASJS linked resource that points to the working copy for flex-asjs. Create a FLEX_FALCON linked resource that points to an installed FlexJS SDK. Setup that same FlexJS SDK as a Flex SDK in Flash Builder's Preferences. Then, import the projects. You may find that Flash Builder hangs building the Core SWC. If that happens, quit Flash Builder and re-open it. The other projects should then build. Flash Builder may also hang if you clean all projects or the Core project. You may find it useful to define a Working Set (of "Resources") and put all projects besides Core in that working set so you can more easily get them to build separately.

The Demo

MXML->SWF
The latest demo is http://home.apache.org/~aharui/FlexJS/FlexJSStore. Run the bin-release/FlexJSStore.html and right-click and choose "View Source" to see the source. You should be able to compile FlexJSStore.mxml into a working SWF by using the Apache Flex SDK Installer to install a FlexJS release and using the js/bin/mxmlc on Mac or js/bin/mxmlc.bat on Windows to compile the source. This is a port of the old FlexStore demo to FlexJS. You can click on the Products button, hover over the people and see toolTips, drag them to the compare window and see drag-and-drop work and effects play.

MXML->JS
The compiler also generated a JS version of the example. You should find an index.html file and a single minified .js file in bin/js-release and a debug version with many .js files in bin/js-debug.

If you then open the HTML file in a browser (I've used FireFox, Chrome and Safari on Mac and FireFox, Chrome and IE8 and IE9 on Windows) you will again see the people and tooltips and drag-and-drop and effects but hey! no Flash is involved. It is pure HTML/JS. Compare the SWF version and the HTML/JS version. They should look pretty similar. I don't know if we can make it exactly the same as it looks in Flash, but we should be able to get better visual match.

MXML->Cordova/PhoneGap

Once you have the HTML and JS file(s), you can copy these into the www folder in a Cordova/PhoneGap application and publish it and see it work on a phone or desktop emulator.

Discussion

As you can see, the FlexJS framework and FalconJX compiler can create applications using familiar tools, components, and workflows. You can use Adobe Flash Builder or other Flex-capable IDE to quickly build or prototype an application on Flash using structured-programming techniques afforded by ActionScript, then convert it to a HTML/JS application that can run on browsers or mobile devices without Flash.