Introduction to FLEX

Bernard JORION

LOGICA – JAVA SIG – December 10th, 2009

(last version: 23-12-09)

I. What is FLEX?

I.1. In a few words...

- **Flex** is a set of technologies (developed by Adobe) that allows you to rapidly build **Flash** Applications.
- It can be seen a new language using both **XML tags** and a variant of JavaScript (this variant is called **ActionScript**).
- Once compiled, you get a Flash file that is useable by any browser using the standard Flash Player. In other words, you do not need any new plug-in or player to execute such a file.

I.2. How do you get it?

- The Flex SDK (Software Development Kit) is available on the site of Adobe (http://www.adobe.com/cfusion/entitlement/index.cfm?e=flex3sdk)
- It's free since the version 2.0, and open-source since the version 3.0. The current version is 3.5. Flex 4 will be released in 2010.
- Like the Java SDK, the Flex SDK is a command-line set of utilities. The most important one is the Flex compiler (**mxmlc.exe**).

I.3. Quick Start

- Flex files are basically **XML files**. That means you need a good knowledge of XML to start programming in Flex. Any good text editor is enough to start writing an XML file.
- By convention, Flex files have as extension **.mxml**.

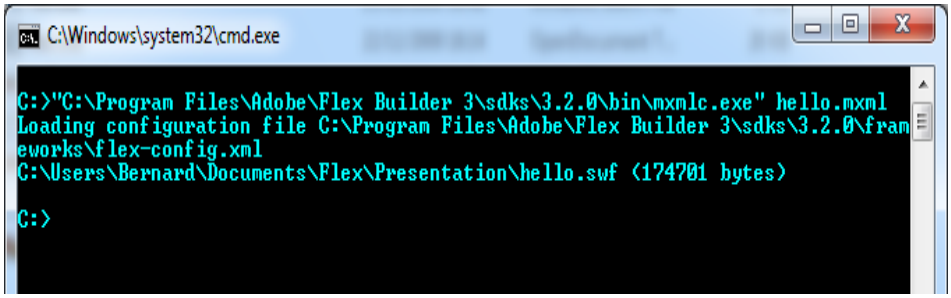
The file below (**hello.mxml**) is the most simple flex program you can write.

```
01  <?xml version="1.0" encoding="utf-8"?>
02  <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*" name="hello.mxml">
03      <mx:Label text="Hello Logica" color="white" fontWeight="bold" fontSize="20" />
04  </mx:Application>
```

Line **01** is the standard opening line for any well-written XML file. Lines **02** and **04** declare the start and the end of the Flex code. They are more or less always the same.

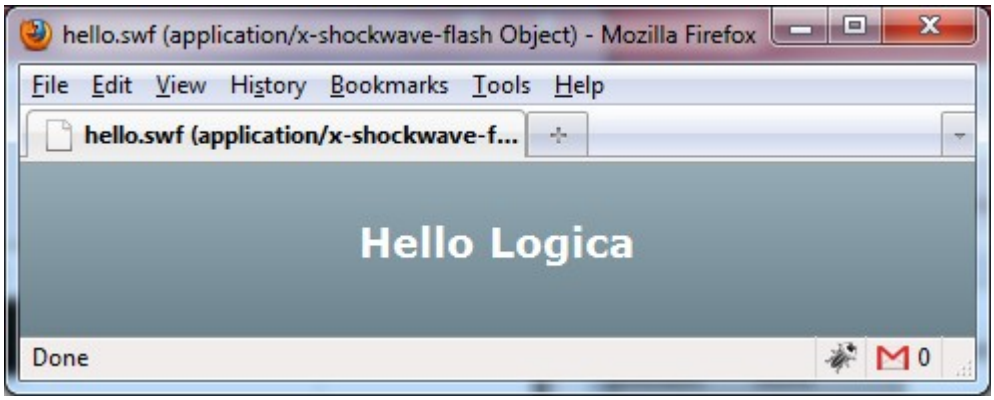
Line **03** is our Flex code. As you can guess, it will display a label “Hello Logica” in white, bold and a size of 20. The **Label** tag has many more attributes whose meaning is pretty straightforward. Except for the text, all those attributes are optional.

Now let us compile that file:



We called **mxmlc.exe** to compile **hello.mxml**. The compiler produced the file **hello.swf**. The **swf** extension means this is a Flash file. *A Flex file is always compiled to a Flash file*. This makes it possible for any browser to display it without knowing anything about Flex.

Now we have that file, we can open it directly with any browser



or you can insert it in an HTML page, like any other Flash component.

Create the file **hello.html** as given below

```
<html>
<head>
        <title>Hello</title>
</head>

<body>
    <center>
        <embed src="hello.swf" />
    </center>
</body>
</html>
```

The **embed** tag is not the best way to insert a Flash component, but it is the simplest one. You will agree it is difficult to make something simpler that the code above.

Open now the hello.html file with any browser and you will get something like

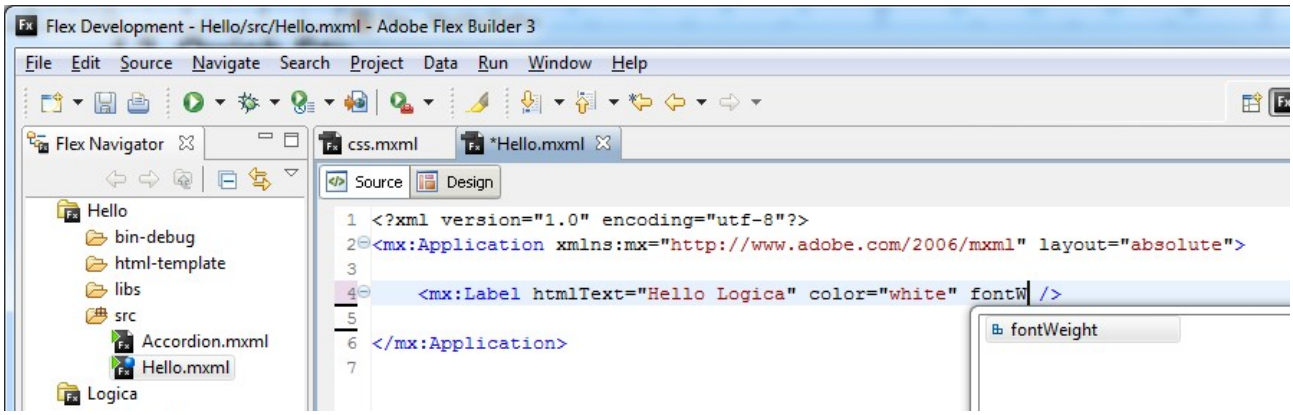
Of course, it is possible to control everything from the background color to the exact position of the component on the page.

I.4. Is there any professional tool to develop in Flex?

Sure (nobody wants to develop using only Notepad). Adobe sells an editor called **Flex Builder**. It is based on Eclipse, so it will be familiar to anybody who already knows Eclipse.

You can download a trial version (free for 60 days) on
http://www.adobe.com/cfusion/entitlement/index.cfm?e=flexbuilder3

This editor will facilitate your life by providing wizards and context-sensitive help.



Flex Builder also offers a WYSIWYG editor (the **Design** tab above, next to the **Source** tab), the possibility to debug to the code step-by-step (something you cannot do with the **mxmlc** compiler) and to execute the code outside any browser, and many other features I am not even aware of... :-)

II. A little bit of theory about Flex

II.1. Where is it used?

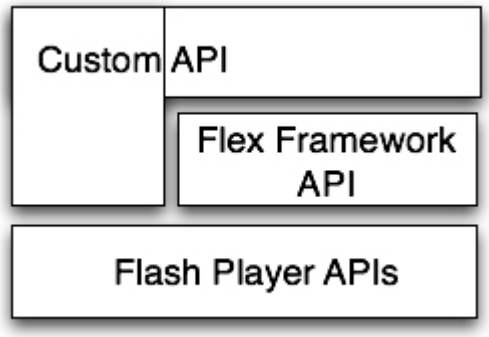
Flex is used to develop components for the browser (the client side of a web application). Therefore it is **not** used in any way on the server side. Most of the time, the server side of a Flex application is written in Java (although you could do it in any language, like PHP or .Net).

The application on the server side does not need to be aware it communicates with a Flex component as the data are normally exchanged as a text file (most of the time an XML file).

II.2. Flash or Flex?

Actually, there is no difference between the two. Flex is just an "easiest" way to write Flash components. The compiled file is a Flash file. So when I talk about a Flex component, I actually mean a Flash component.

The **Flex framework** is just a layer of abstraction above the Flash API. It consists of thousand of lines of code, all of which run instructions that the Flash Player can understand.



The **Custom API** is your own code. You see that from your code you can call the Flex framework or directly work with the Flash API if you want to do it the hard way :-)

The main advantage of Flex is that it is easier to learn and to write, and you have at your disposition hundreds of powerful components (visual and non-visual) you would have to write yourself with Flash.

II.3. Any documentation?

Yes, plenty of it. Adobe offers an excellent support for its products.

- Flash API: http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/
- Flex API: http://livedocs.adobe.com/flex/3/langref/
- Flex example components: http://www.adobe.com/devnet/flex/samples/photo_explorer/

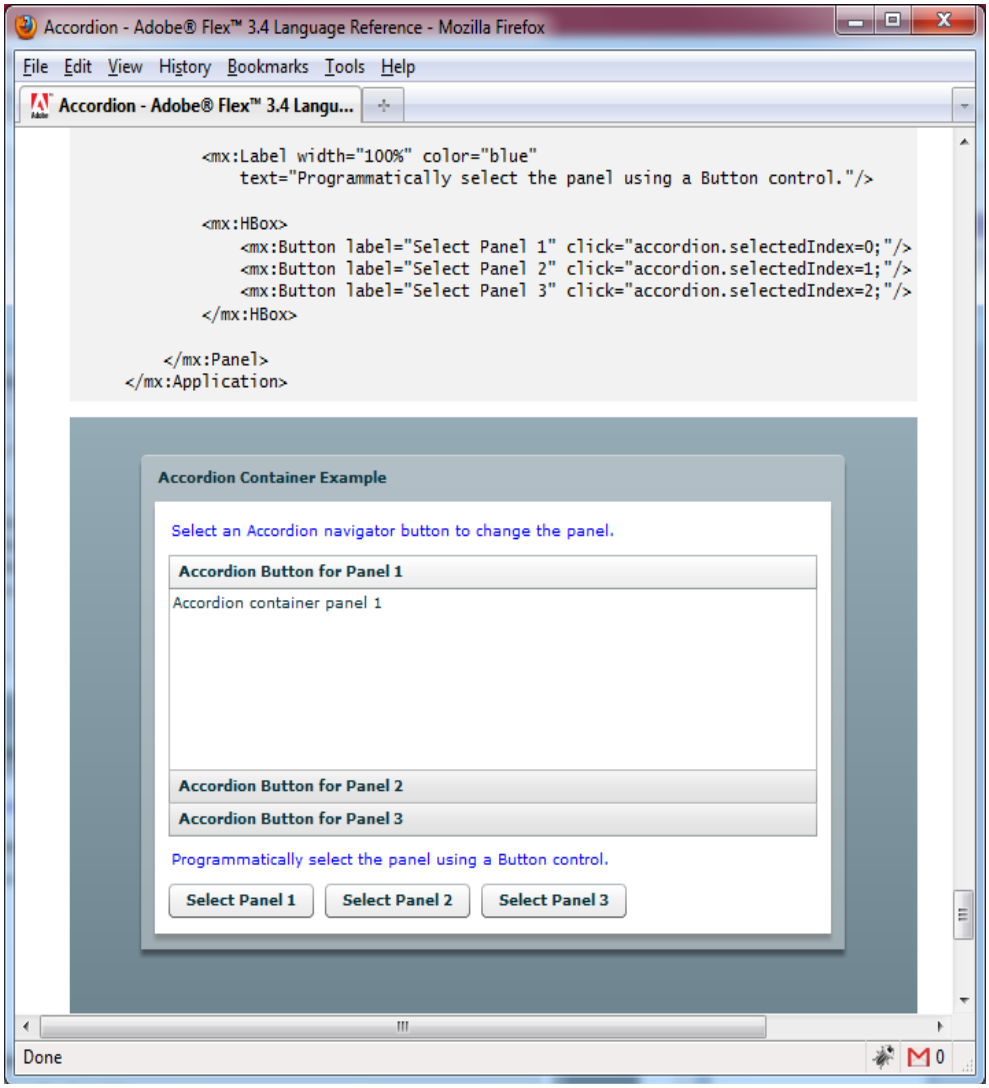
I recommend the photo explorer component given above. It is really beautiful, and the source code is available.

You'll find on internet an unlimited amount of articles, tutorials, examples, and blogs about Flash and Flex, as well as many books.

To start learning Flex, its API site (see link above) contains a detailed explanation and each time one or several components you can directly test. For example the **Accordion** entry

(http://livedocs.adobe.com/flex/3/langref/mx/containers/Accordion.html)

You will find at the bottom of the page both the code source of an example and the resulting component. You will be able to interact directly with that component:



II.4. ActionScript

You do not write a whole component using only XML tags. You also need a language to handle the events generated by the user's action (like clicking on a button). That language is called **ActionScript** and is based on JavaScript (you can see it as an customized or advanced version of JavaScript).

Actually when you develop in Flash, you write your code exclusively in ActionScript. When you use Flex XML tags (like `<mx:Label ...>`), the compiler will convert those tags into ActionScript code. That is why the file generated by the compiler can be read by a simple Flash player: it contains only ActionScript (converted to a binary format).

For the non-technical readers, remember also that JavaScript has nothing to do with Java, except the name (and the name was chosen for marketing purposes).

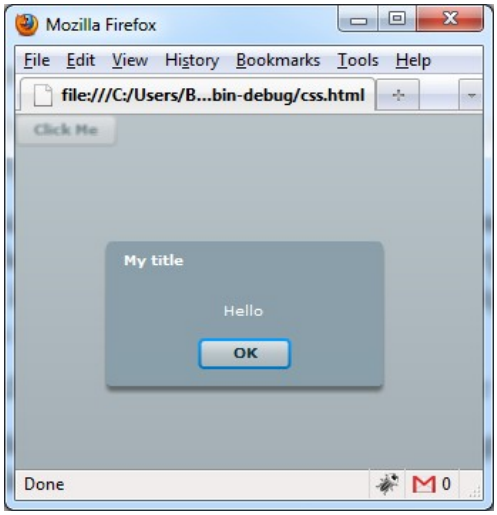
Let us see an example of ActionScript

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
3
4     <mx:Script>
5       <![CDATA[
6         import mx.controls.Alert;
7
8         function action(): void
9         {
10          var text:String = "Hello";
11          var title:String = "My title";
12          Alert.show(text, title);
13        }
14      ]]>
15    </mx:Script>
16
17    <mx:Button label="Click Me" styleName="example" click="action()"/>
18
19  </mx:Application>
```

- ActionScript must be enclosed inside **<mx:Script>** tags.
- Line 6, we import a class from the Flex API. This class (**Alert**) contains methods and attributes, like any Java class.
- Line 8, we define a function named **action.** This function does not return anything, whence its return type: **void.** *The main difference between ActionScript and JavaScript is that ActionScript is a typed language*. Each variable must be assigned a type when it is first declared, and functions must declare a return type.
- Lines 10 and 11 we declare two variables. They are typed as String (using **:String**)
- Line 12 we call the static method **show** on the Alert object, giving as parameters the two variables declared the lines above.
- Line 17 we create a button (using a Flex XML tag). The **click** event of this button (fired when the user clicks on it) will call the function **action.**

You can mix, to some point, ActionScript inside the XML tags like you do with JavaScript inside HTML tags.

If you compile the code above and execute it, you will first see a button called "**Click Me**". And by clicking on it, a pop-up will appear:



II.5. Is that all about ActionScript?

No, of course. Using both Flex XML tags and ActionScript you can create whole applications. ActionScript allows you to handle events, make HTTP requests (and read the responses), read and write XML files, dynamically create objects, and so on...

You can also write whole classes in ActionScript. Those files have the ".**as**" extension and can be re-used through all your applications.

II.6. A few disadvantages of Flex

- Sometimes heavy: with only one small XML tag, you can unknowingly incorporate many lines of ActionScript code, or some libraries not even used. That's why the shortest file we wrote using Flex (the **hello.swf** file) is 171 KB large. Not much by today standards, but you should always be careful it does not grow too much
- If you write for instance a wizard using Flex (or Flash for that matter), you will be able (as you can expect) to navigate through this wizard using the **back** and **next** buttons. But do not forget that by doing so you stay inside the same HTML page (in other words, the URL in your browser does not change). That means you will not be able to bookmark a given step in your wizard.
  For a wizard that does not matter much, but if you write a movie catalog for instance, that becomes a problem because the user will not be able to bookmark the page of his favorite movie. Note that there are ways to solve that issue.

II.7. Example

- www.belgacom.tv: a site completely built with Flex + Java.

III. Other Resources

Several frameworks have been developed by Adobe or some third-parties companies to extend or facilitate the use of Flex.

III.1. Frameworks and tools

- **Adobe BlazeDS** is a library that provides a complete publish/subscribe infrastructure. It allows Flex clients and the server to exchange messages in real time.

Using BlazeDS, Flex application can directly invoke methods of Java objects deployed in an application server (or PHP, …). In that way, it is similar to SOAP, except it is using a binary format (AMF: Action Message Format) while SOAP uses standard XML.

Spring developed its own BlazeDS module, an additional layer between Adobe BlazeDS and Spring's beans. If you need to write a Flex component and you are already using Spring, it is worth a look.

- **Adobe Integrated Runtime** (AIR): a cross-platform runtime environment (meaning it is working on Windows, Linux, Mac). It allows you to develop applications that can be used both remotely (on internet) or locally (on your desktop).

III.2. Demo Application

During the presentation, I showed a small web application I developed for testing purposes (using Flex for the client side and Java / Struts for the server side). It would not be worth it to represent the whole application here, but I can always forward it to you if you want to have at look at the source code.

The main purpose of this application was to give more details about the way Flex and Java can communicate with each other (using an XML file).

III.3. Learning Flex

Becoming fluent in Flex (beside some design skills) would require a few weeks of work. And that would only be the beginning. Like Java, Flex knows many extensions and libraries (for Java, think of Hibernate, Spring, JavaEE, …). But as in Java, you can write your own decent applications within just a few days.

A book I recommend about Flex is **Programming Flex 2** (O'Reilly):
http://oreilly.com/catalog/9780596526894/index.html

That's all Folks!