

First of all create an XML file. I did one by copying a couple of books from the Amazon web site. Here is the simple structure that I created:

Here is how the XML file looks like:

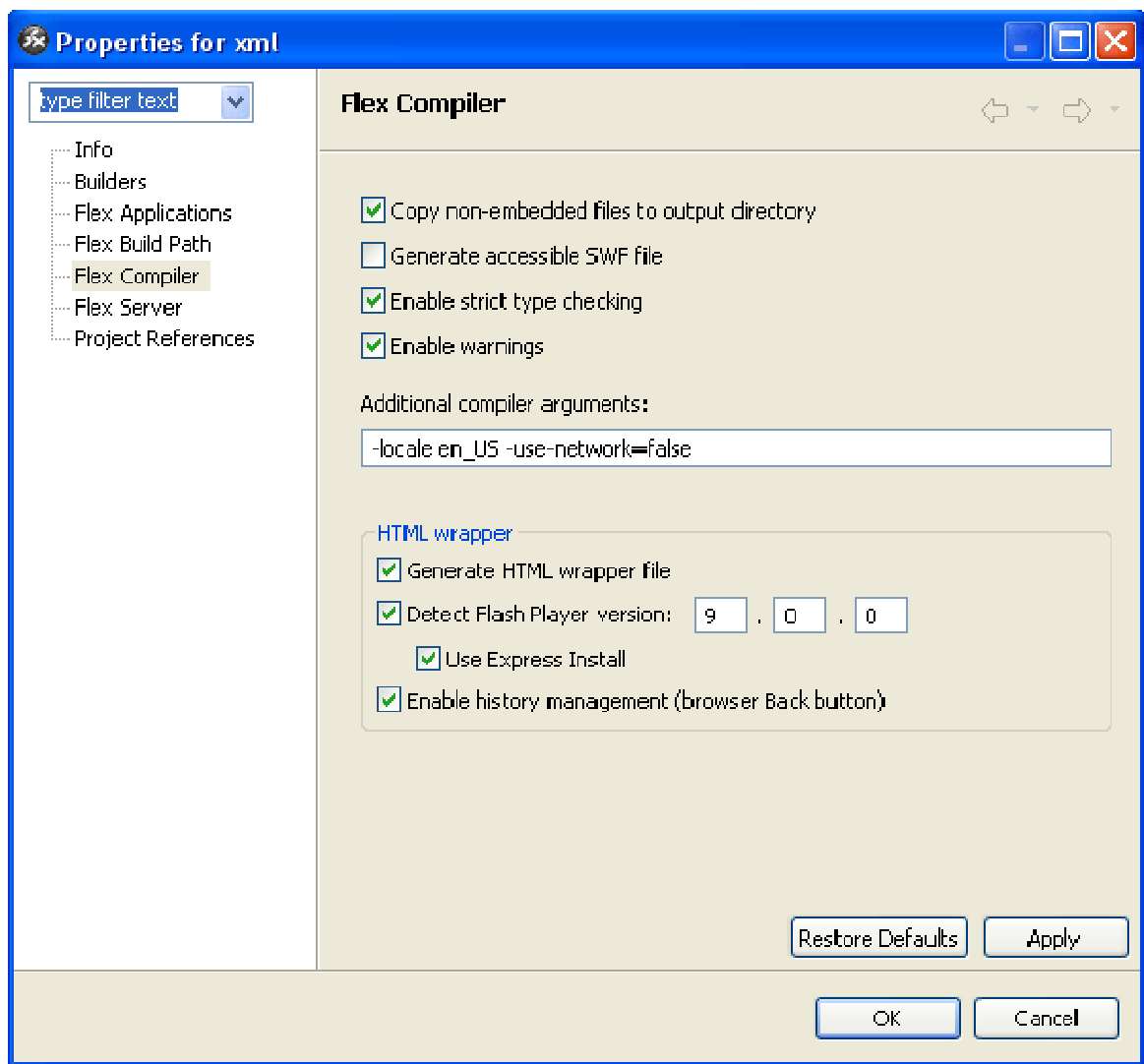
```
<?xml version="1.0" ?>
<books>
<item>
<title>The Complete Guide to Flex 2 and ActionScript 3.0</title>
</item>
<item>
<title>Programming Flex 2 (Programming)</title>
</item>
<item>
<title>Adobe Flex 2: Training from the Source</title>
</item>
</books>
```

This file as a simple structure. Main node is books, then child nodes are called “item”. What we will learn on this tutorial is how to recall the title of the list of books from the XML file. This may be quite simple to lots, of people, but it was quite difficult to me indeed.

Ok, let’s open Flex now. Create a new project, and save the empty file somewhere on your computer. Once you have done so, we need to pick up the XML file just created, and save it on the “bin” folder inside the Flex project folder.

Now, because Flex will look on the Internet to try to find the XML file, we need to specify, on Flex builder, that the file is local and we do this by going inside the Project Menu > Properties > Flex Compiler.

On the little box that says “Additional compiler arguments” we need to add the string “-use -network=false”. In this way Flex will know that the file is local and that does not have to go on the Internet to find it. Here is a screenshot on how to complete this step:



Now that we have done this, we can start to write the Flex code. Here is the complete code, than I will go step by step, or is better to say line after line, and try to make you understand... I really hope so 😊 Is not a lot of code by the way, so don't be scared!

Here we go!

```
<?xml version="1.0" encoding="utf-8" ?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
initialize="myService.send()">
<mx:Script>
<![CDATA[
import mx.collections.ArrayCollection;
import mx.rpc.events.ResultEvent;
```

```

[Bindable]
private var myData:ArrayCollection;

private function resultHandler(event:ResultEvent):void {
myData = event.result.books.item;

}

]]>
</mx:Script>

<mx:HTTPService id="myService"
url="books.xml"
result="resultHandler(event)"/>

<mx:Text id="something" text="{myData.getItemAt(0).title}"/>

</mx:Application>

```

Block by block:

First of all, we declare that the document is xml and all those other stuff. What is important about this block is to set the initialize function. This is the function that recalls the XML files. A call to the file must be linked to an event. The event is the initialization on the program.

```

<?xml version="1.0" encoding="utf-8" ?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
initialize="myService.send()">

```

Then we import classes necessary to make the file work. We need two classes. The first one: ArrayCollection is the one needed to handle arrays. The second one, ResultEvent is the class needed to read the results coming from reading an XML file.

```

<mx:Script>
<![CDATA[
import mx.collections.ArrayCollection;
import mx.rpc.events.ResultEvent;

```

Next we declare that datas from now on will be bindable. If you don't know this word, here is a definition that comes from Google. I really don't know how to explain it: [Bindable](#). Then we create the array that will store all the results from the call to the XML file.

```
[Bindable]
private var myData:ArrayCollection;
```

Next we define a function that will go down to the nodes of the XML files needed when we show the results from the call. As you can see we go down from books, that is the main node of the XML file and then down one more step to item. That is the information that we are trying to access. Finally, we close the script tag.

```
private function resultHandler(event:ResultEvent):void {
myData = event.result.books.item;
}

]]>
</mx:Script>
```

Now we do the http request to the XML file. Note that the request itself doesn't do nothing because it needs to be recalled by an event. As I said before, the http request is called when we start the Flex Application, by using the parameter "initialize" with the suffix send(). The suffix send() will make the http request make a call to the XML file. Also we instruct the call to go down on the nodes of the file needed by the program.

```
<mx:HTTPService id="myService"
url="books.xml"
result="resultHandler(event)"/>
```

Finally we test that everything works by writing the first result from the HTTP request. Note that rather than writing myData[0], Flex advise to call the element of the array by using the function getItemAt and then the key of the array needed. Also if the item node would contain more informations rather than just the title you could recall it by appending the name of the node at the end of the variable that will give the value to the text field. Finally we close the application

```
<mx:Text id="something" text="{myData.getItemAt(0).title}"/>
```

```
</mx:Application>
```

I hope that this is clear enough and I will be happy to receive any modification or explanation of the processes used by Flex to achieve such result. In the while, no sparkles neither special effects