

Working with XML and Flex

How do I display my server-based XML data in Flex?

Rollo has the following XML on the SMU server:

```
<?xml version='1.0' ?>
<friends>

  <friend>
    <name>Paris Hilton</name>
    <email>paris@gmail.com</email>
    <zip>90210</zip>
    <pin>p11</pin>
  </friend>

  <friend>
    <name>GO GO</name>
    <email>gogo@gmail.com</email>
    <zip>10020</zip>
    <pin>go</pin>
  </friend>

</friends>
```

One of the simplest ways is to set up an `<mx:XML>` element that will automatically load the XML from the source URL.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
3
4   <mx:XML id="rollopals" source="http://lyle.smu.edu/~coyle/rollo/friends.xml"/>
5
6   <mx:Panel title="Rollo XML">
7
8     <mx:DataGrid dataProvider="{rollopals..friend}">
9       <mx:columns>
10        <mx:DataGridColumn dataField="name" headerText="Name"/>
11        <mx:DataGridColumn dataField="email" headerText="Email"/>
12        <mx:DataGridColumn dataField="zip" headerText="Zip"/>
13        <mx:DataGridColumn dataField="pin" headerText="Pin"/>
14      </mx:columns>
15    </mx:DataGrid>
16
17  </mx:Panel>
18 </mx:Application>
19
```

Output :

Rollo XML			
Name	Email	Zip	Pin
Paris Hilton	paris@gmail.com	90210	p11
GO GO	gogo@gmail.com	10020	go

Line 4: Automatically loads XML from the web. If the file cannot be located an error is reported. The XML object that is created has the id=rollopals.

Line 6: Sets up a panel to hold the DataGrid. This is not necessary but will prove useful when we want to add other UI components.

Line 8: Sets up the DataGrid with a dataProvider that maps to the XML coming in from the server. Here we use {rollopals..friend}. This will collect all <friend> elements anywhere in the XML. Each friend element is mapped to a row of the DataGrid and the content of the friend element feeds the columns of the display. However, we MUST tell Flex what to display in each column using <mx:columns>.

Lines 9-14: Define the columns of the DataGrid. We use dataField="zip" to specify which subelements of friend to map to a column. The column header is given by headerText="Zip".

That's it. You are able to display XML from a server in Flex.

What if my XML is more complex with nested elements?

Then we need more complex DataGrid techniques (using label functions to map specific parts of a complex XML document to the DataGrid). When getting started, I recommend But keeping your XML simple, and not using nested elements. By keeping things simple as in this example, the Flex is easier to work with.

How can I modify my incoming XML?

Use ActionScript code to modify the XML using Flex E4X dot notation.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
3   applicationComplete="modifyXML();" >
4
5   <mx:XML id="rollopals" source="http://lyle.smu.edu/~coyle/rollo/friends.xml"/>
6
7   <mx:Script>
8     <![CDATA[
9       public function modifyXML(): void {
10        rollopals.friend[1].name = "goon-go";
11      }
12    ]]>
13 </mx:Script>
14
15 <mx:Panel title="Rollo XML">
16
17   <mx:DataGrid dataProvider="{rollopals.friend}">
18     <mx:columns>
19       <mx:DataGridColumn dataField="name" headerText="Name"/>
20       <mx:DataGridColumn dataField="email" headerText="Email"/>
21       <mx:DataGridColumn dataField="zip" headerText="Zip"/>
22       <mx:DataGridColumn dataField="pin" headerText="Pin"/>
23     </mx:columns>
24   </mx:DataGrid>
25
26 </mx:Panel>
27 </mx:Application>
28
```

Line 3: tells Flex to execute the method modifyXML after the application loads. The code uses E4x dot notation to modify the second xml entry ([0] is the first).

Line 10: modifies the XML, changing the name, which is automatically reflected in the DataGrid.

Rollo XML			
Name	Email	Zip	Pin
Paris Hilton	paris@gmail.com	90210	p11
goon-go	gogo@gmail.com	10020	go

How can I write these changes back to the server?

To GET data from a server all you need is to place the file on the server, give it read permission, test it with a browser, and you're ready to write a Flex client to retrieve it. However, to add data to a server, you will need a program (Java servlet, JSP, PHP, etc.) to take the data from an HTTP POST request and write it to a file on the server.

To write data to a server from Flex, you can use the `URLLoader` and `URLRequest` classes, to set up the request data and then send it to some URL. However, using these classes are bit more complex than reading XML from a server, so I like to make sure I have my server code working before I try and tackle anything too complex.

To help in the process, I have written a small PHP program that echoes back any incoming POST data. The php program located at <http://lyle.smu.edu/~coyle/php/>

```
1 <?php
2
3 #ini_set("error_reporting","E_ALL");
4 error_reporting(E_ALL);
5 ini_set('display_errors','1');
6
7 print "ECHO SERVER SAYS:";
8
9 if ( $_SERVER['REQUEST_METHOD'] === 'POST' ) {
10 // Read the input from stdin
11 $postText = trim(file_get_contents('php://input'));
12 }
13 else {
14 $postText = "The client did NOT send a POST request";
15 }
16
17 print $postText;
18
19 print ":ECHO SERVER DONE";
20
21
22 ?>
```

Here is the modified Flex program that writes to the PHP program on the server.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
3   applicationComplete="sendXML();" >
4   <mx:Script>
5     <![CDATA[
6
7       import mx.controls.Alert;
8       import mx.rpc.events.ResultEvent;
9
10      private function sendXML(): void {
11        var myUrlLoader:URLLoader = new URLLoader();
12        var urlRequest:URLRequest = new URLRequest(
13          "http://lyle.smu.edu/~coyle/php/flextest/echoPost.php");
14
15        var msg:String = "<msg>What is up?</msg>";
16
17        myUrlLoader.addEventListener(Event.COMPLETE, handlerComplete);
18        urlRequest.data = msg;
19        urlRequest.method = URLRequestMethod.POST;
20        myUrlLoader.load(urlRequest);
21      }
22
23      private function handlerComplete(event:Event) {
24        var response:URLLoader = URLLoader(event.target);
25        var responseData:String = response.data;
26        Alert.show(responseData);
27      }
28    ]]>
29   </mx:Script>
30
31
32 </mx:Application>
33

```

This method is called when the server sends its response to Flex

Line 3: Executes the **sendXML** method as soon as the application loads.
 Line 13: Identifies the server program to call
 Line 17: Sets up a listener function for an **Event.COMPLETE** event when the server returns with its data.
 Line 18: Sets up the text string to be used in the payload of the POST
 Line 20: This is where the action happens.

Line 24: the handler code when **Event.COMPLETE**
 Line 26: creates some output so we can see that things are working OK

