AIR Phone Book application - Part 2 (Functions and WebService)

Posted At : 3 April 2009 15:13 | Posted By : Shaun McCran
Related Categories: Flex Remoting, AIR, RIA, Flex

As our application starts I want to fire the request for data, so we call an init() method on initialization. Also I have turned the flex chrome off here with 'showFlexChrome="false"'.

The init() method calls an event listener that controls the window movement (Drag and drop) and makes a call to getData().

```
import mx.controls.Alert;
import mx.collections.*;
import mx.rpc.events.FaultEvent;
import mx.rpc.events.ResultEvent;
import mx.collections.ArrayCollection;

[Bindable]
private var loadedData:ArrayCollection;
public function init():void
{
// start the move listener
moveListener()
// get the remote data
getData()
}
```

The moveListener() method adds a listener to the outerCanvas element which forms the application 'border'. When this event is fired it calls moveWindow.

The getData() function calls the webservice, and specifies which method to call.

```
public function moveListener():void
{
// mover event
outerCanvas.addEventListener( MouseEvent.MOUSE_DOWN, moveWindow );
}

public function getData():void
{
popData.getData();
}
public function moveWindow( event:MouseEvent ):void
{
var str:String = event.target.valueOf();

// if its a datagrid then don't do the move
if ( str.search("displayPeople") >= 1)
{
// Do nothing
}
else
{
stage.nativeWindow.startMove();
}
}
```

The moveWindow function also contains a check to see if the datagrid was the event target, as this was interfering with the functionality of the Datagrid. It would be interesting to see if anyone else has a more elegant solution to this, rather than a specific element check.

To populate our datagrid we need to use a data provider. In FLEX applications I usually use the RemoteObject function, but for AIR I've been using the WebService tag.

```
WebService id="popData" wsdl="http://url/wld/services/phoneBook.cfc?wsdl" showBusy
<mx:operation name="getData" fault="faultHandler(event)" result="resultsHandler(ev
</mx:WebService>
```

The final two 'chrome' functions we need are the minimize and close functions. I will detail handling custom chrome in another article.

```
public function onMinimize():void
{
stage.nativeWindow.minimize();
}

public function onClose():void
{
stage.nativeWindow.close();
}
```

Our WebService is referencing two functions. A results handler and a fault handler.

```
public function resultsHandler(event:ResultEvent):void
{
// trace(event.result)
displayPeople.dataProvider = popData.getData.lastResult
}

public function faultHandler(event:FaultEvent):void
{
Alert.show("Error: " + event.fault.faultString, "Application Error"
}
```

The resultsHandler() simply assigns the datagrids dataprovider as the result of the WebService call. By adding DataGridColumn's to the datagrid with the right naming convention our results from the returned query object will map directly to our datagrid.

The faultHandler() function simply Alerts a user to a fault event.

Lastly I have a function that assigns the image source dynamically based on the click event in the datagrid.

```
public function changeImage(img:String):void
{
userImage.visible = true
userImage.source = "http://url/wld/phonebook/" + img
}
```

So that completes the Phone Book AIR application. There are one or two tweaks I'd like to make in the image handling, but otherwise its exactly the spec I had in mind.

You can view the full code here .