

Hands on Flex!
Connecticut ColdFusion User Group.
September 12, 2006.

Library application:

Step 1: Introduction.

- Overview
- DB Schema
- Web Service Address
 - Setting up the Web Service

Step 2: Starting the application

- Create a project titled "library"
- Add content are, for components:
 - Panels & Data grid

```
<mx:HBox width="100%">
  <mx:Panel title="Item List" layout="vertical" width="30%">
    <mx:DataGrid id="dg" width="100%">

      </mx:DataGrid>
    </mx:Panel>

    <mx:Panel width="70%" title="Item Details">

      </mx:Panel>
    </mx:HBox>
```

- Create a script block:

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

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

Step 3: Call data to populate the data grid

- Create the Web Service call:

```
<mx:WebService id="libraryItemList"
  wsdl="http://www.cfugitives.com/library.cfc?wsdl"
  useProxy="false" showBusyCursor="true">
  <mx:operation name="getLibraryItems"/>
</mx:WebService>
```

- Call the Web Service


```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute"
  creationComplete="libraryItemList.getLibraryItems()">
```
- Populate the data grid with the results


```
<mx:DataGrid
  dataProvider="{libraryItemList.getLibraryItems.lastResult}"
  id="dg" width="100%">
  <mx:columns>
    <mx:DataGridColumn dataField="ITEMID"
      visible="false"/>
    <mx:DataGridColumn dataField="ITEMNAME"
      visible="true"/>
    <mx:DataGridColumn dataField="ITEMTYPE"
      visible="true"/>
  </mx:columns>
</mx:DataGrid>
```

Step 4: Create a form to display detail information

- Create a Form


```
<mx:Form>
  <mx:FormItem label="Item Name:">
    <mx:Text id="itemName" />
  </mx:FormItem>
  <mx:FormItem label="Item Type:">
    <mx:Text id="itemType"/>
  </mx:FormItem>
</mx:Form>
```
- Bind the form items to the data grid


```
<mx:Form>
  <mx:FormItem label="Item Name:">
    <mx:Text id="itemName"
      text="{dg.selectedItem.ITEMNAME}"/>
  </mx:FormItem>
  <mx:FormItem label="Item Type:">
    <mx:Text id="itemType"
      text="{dg.selectedItem.ITEMTYPE}"/>
  </mx:FormItem>
</mx:Form>
```

Step 5: Create an “Edit” state to edit the item details

- Switch to design mode
- Create a button on the form – “Edit”
 - Label = “Edit”
 - “On Click” = “currentState = ‘edit’ ” (Note: Case sensitive)
 - Remove the form item label (form item is created automatically)

- Create a new state, called 'edit'



- Remove the “text” items in the form, replace them with “textinput” controls
 - Note: Make sure you are placing the new controls inside the form items.
 - Set the “text” property of the first textinput control to be:
 - { dg.selectedItem.ITEMNAME }
 - Set the “text” property of the second textinput control to be:
 - { dg.selectedItem.ITEMTYPE }
- Edit the “edit” button:
 - Set the “text” property to “Update”
 - Set the “On click” property to “currentState = “

Step 6: Update the database with edit entries

- Add a new method to the web service


```
<mx:operation name="updateItem"
  fault="Alert.show(event.fault.message)"
  result="handleUpdateResult(event)">
  <mx:request>
    <itemID>
      { dg.selectedItem.ITEMID }
    </itemID>
    <itemName>
      { item_name_edit.text }
    </itemName>
    <itemType>
      { item_type_edit.text }
    </itemType>
  </mx:request>
</mx:operation>
```
- Create a function to handle the method result


```
public function handleUpdateResult(event:ResultEvent):void{
  currentState = "";
  libraryItemList.getLibraryItems();
}
```
- Change the “update” button to call the new method
 - This can be done in two manners:
 - Update the code


```
<mx:SetEventHandler target="{button1}" name="click"
```

```
handler="libraryItemList.updateItem()"/>
```

- Update the button on the design view
 - Change the “On Click” property of the button to:
libraryItemList.updateItem()