Adobe Flash Islands for Web Dynpro Java: Creating an Interactive Slider

Applies to:

Web Dynpro for Java applications in SAP NetWeaver CE 7.11. For more information, visit the User Interface Technology homepage.

Summary

With this small Slider example the interaction between Flash Island and the embedding Web Dynpro application is explained.

Flex project download: Flex Slider project

Web Dynpro Java DC: Web Dynpro Java Slider SCA

Author: Stefanie Bacher

Company: SAP AG

Created on: 19 January 2009

Author Bio

Stefanie Bacher works as a product specialist and information developer within the SAP NetWeaver Solution Management Rollout Team for User Intercation Technology. She focusses on Web Dynpro Java.

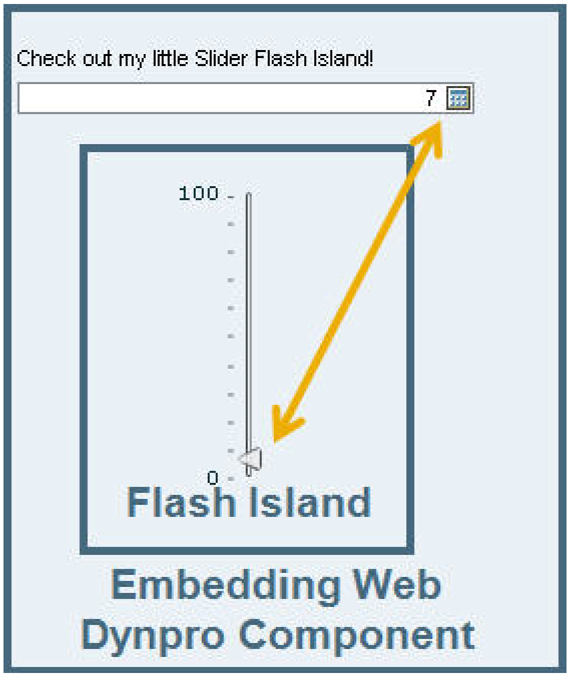
Table of Contents

Introduction

This small sample is designed to explain the interaction between a Flash component and the embedding Web Dynpro Java application.

As a Flash component we will use a slider that allows choosing a value between 0 and 100. On Web Dynpro side an *InputField* displays this value. If you now change the value of the *InputField*, the slider's thumb moves to this value, and the other way round: If you change the slider's value, this value is also displayed in the *InputField*.

With this sample you can see how the data has to be bound to enable this connection and how the event on Flash side has to be handled to let Web Dynpro know that the data has changed.


Check out my little Slider Flash Island!
7
100
0
Flash Island
Embedding Web Dynpro Component

There are different steps to establish this connection between Web Dynpro and the embedded Flash Island:

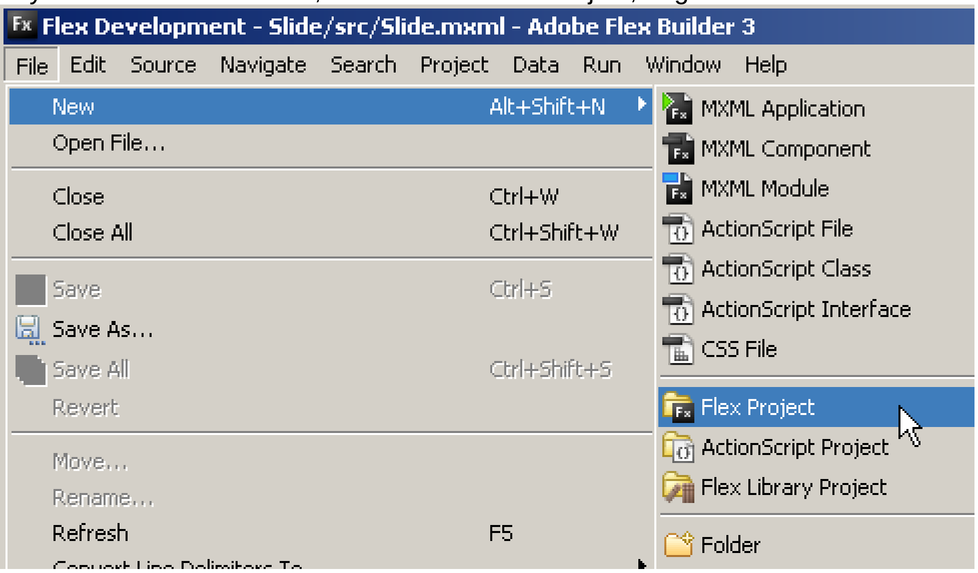
- The Flex component itself has to be wrapped into a Flash Island.
- This Flash Island has to be embedded into Web Dynpro Java.

Prepare the Flex Project

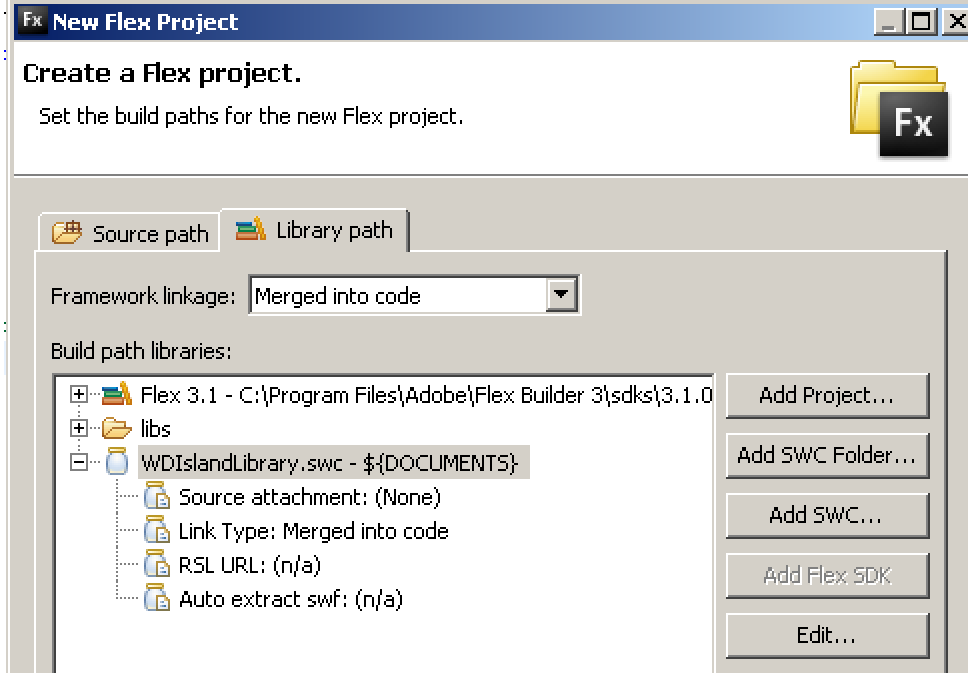
To implement Flash Island for Web Dynpro Java applications integrating Flash Islands, you require the following:

- Adobe Flex Builder 2 or 3
- Adobe Flash Player 9 Update 3
- Flash Island Flex library **`WDIslandLibrary.swc`**.
  To enable the use of a Flash/Flex component in Web Dynpro, the component must exist in the form of a *FlashIsland*. This wrapping enables the communication between the Flex component and the Web Dynpro Framework. This library is available on the SAP NetWeaver Application Server CE under the following path:
  ...\usr\sap\ce1\j00\j2ee\cluster\apps\sap.com\tc~wd~dispwda\servlet_jsp\webdynpro\resources\sap.com\tc~wd~dispwda\root\global\activeComp\

1. In your Adobe Flex Builder, create a new Flex Project, e. g. called *Slider*.



2. To add **`WDIslandLibrary.swc`** to the Library path, choose *Add SWC...*, browse to the directory where you have stored the library and add it.



Create a Flex project.

Set the build paths for the new Flex project.

Framework linkage: Merged into code

Build path libraries:

- Flex 3.1 - C:\Program Files\Adobe\Flex Builder 3\sdks\3.1.0
- libs
- WDIslandLibrary.swc - ${DOCUMENTS}
  - Source attachment: (None)
  - Link Type: Merged into code
  - RSL URL: (n/a)
  - Auto extract swf: (n/a)

3. To specify the Flex Compiler, select the project, open the context menu and choose Flex Compiler. Use *Flex 2.0.1. Hotfix 3* as *SDK*.

4. Confirm with Finish. The Flex project is created.

Create the Slider

In this sample we will use a vertical slider which is part of Adobe Flex control library.

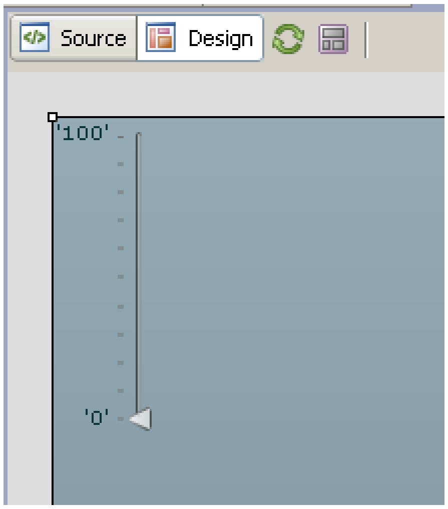
> You can copy the complete code from the appendix of this document.

1. You can now switch to the *Source* tab for the *Slider.mxml* editor.
2. To insert the vertical slider, add the following lines of code into the *mx:Application* tag:

```
<mx:Script>
      <![CDATA[
      import mx.events.SliderEvent;
      import mx.controls.sliderClasses.Slider;
      [Bindable]
      public var xvalue:int;
      ]]>
</mx:Script>
      <mx:VSlider
      id="slideIn"
      value="{xvalue}"
      maximum="100"
      minimum="0"
      tickInterval="10"
      labels="['0', '100']"
      width="50" >
</mx:VSlider>
```

You declared a variable *xvalue* that is used as value for the slider's thumb. This variable will be bound in a later step to the Web Dynpro context.

Switching to the *Design* tab, your slider should now look like this:



Wrapping the Flex component into a Flash Island

To establish and manage the connection to Web Dynpro, you have to implement some steps:

1. To register your Flex component as a Flash Island, create a method called *initApp().* In this method call the *FlashIsland's* method *register()* with the parameter *this*. This method is called during initialization of the Flex application:

```
<mx:Application
      xmlns:mx="http://www.adobe.com/2006/mxml"
      layout="vertical"
      backgroundColor="#EAF1F6"
      width="100%"
      height="100%"
      initialize="initApp()"
      backgroundGradientColors="#EAF1F6">
<mx:Script>
```

```
        <![CDATA[
        import sap.FlashIsland;
        import mx.events.SliderEvent;
        import mx.controls.sliderClasses.Slider;
        [Bindable]
        public var xvalue:int;
        public function initApp():void
        {
              FlashIsland.register(this);
        }
………
```

Additionally the background color is set to the SAP's standard background color to get a seamless display on the screen.

Handle the Flash Event

1. To be able to react on the event that is triggered if the user changes the thumb's position, create the following function:

```
<mx:Script>
      <![CDATA[
      ……
      private function sliderChange(event:SliderEvent):void
      {
            var currentSlider:VSlider=event.currentTarget as VSlider;
            xvalue = currentSlider.value;
            FlashIsland.storeProperty(this,"xvalue",xvalue);
            FlashIsland.fireEvent(this, 'change', null );
      }
      ]]>
</mx:Script>
```

   First the current value is stored in the *xvalue* variable and then the method *storeProperty* is called that ensures that the value is saved before the event is fired. At least the event itself is fired.

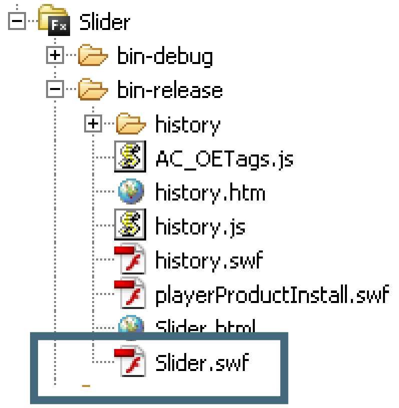
2. To call this function with the slider's change event, add the following properties to the slider:

```
<mx:VSlider
      id="slideIn"
      value="{xvalue}"
      maximum="100"
      minimum="0"
      width="50"
      liveDragging="false"
      change="sliderChange(event)">
</mx:VSlider>
```

   liveDragging has to be set to false to ensure that only one event is triggered when the user releases the slider's thumb.

The Flash component is now finished and you can build the *swf* file that is needed in Web Dynpro. To do this:

1. Select *Project -> Export Release Build...* and follow the wizard.
   You now can find the *Slider.swf* file in the *bin-release* folder.

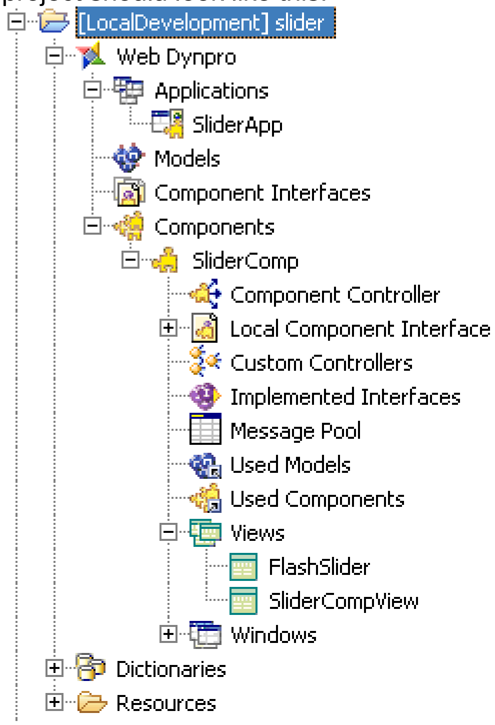


Create the FlashIsland UI element in Web Dynpro Java

You now have finished all steps needed on Flex site for creating a Flash Island that can be used in Web Dynpro Java.

The Flash Island UI element spans an entire view, that's why you have to create a separate view for it.

1. Create a Web Dynpro Development component (DC) and add a new view to the window. You DC project should look like this:

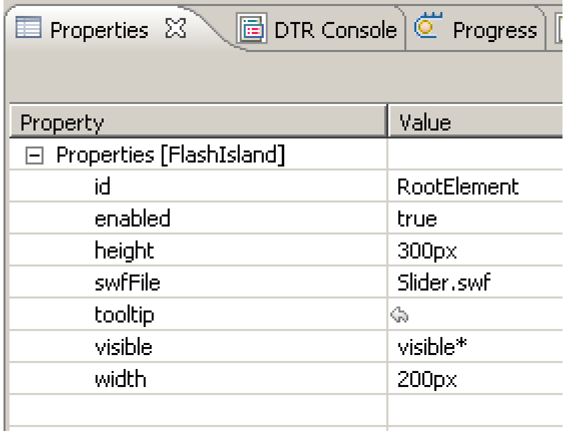


2. Add a *ViewContainerUIElement* to the *SliderCompView* and add the *FlashSlider* view into.
3. Open the *FlashSlider* view and replace the *RootElement's* default container with a *FlashIsland*. To do so, select the *RootElement*, open the context menu and choose *Replace With... -> FlashIsland*.
4. Specify appropriate *width* and *height* values to avoid scroll bars to appear.
5. Insert the following view elements into this *RootElement*:

- *Property* (`GACProperty`)
- *Event* (`GACEvent`)

The *Property* will help you bind the slider's value and the *Event* has to be connected to the event you've created in the Flash component.

6. Copy the *Slider.swf* file into the mime repository (The relevant path is: <*Web Dynpro DC name*> → *Resources* → *src* → *mimes* → *Components* → <*component name*>) and specify *Slider.swf* as value for the *FlashIsland's swfFile* property.



Property	Value
⊟ Properties [FlashIsland]	
id	RootElement
enabled	true
height	300px
swfFile	Slider.swf
tooltip	
visible	visible*
width	200px

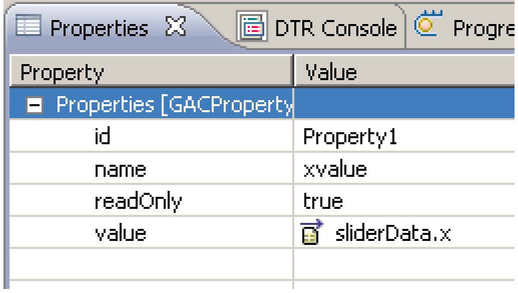
Define the data flow

We now have to create a context attribute to be able to store and transfer the slider's value.

1. In your component controller, create a context node called *sliderData* and an attribute called *x* of type *integer* inside this node.
2. Set the node's cardinality property to *1:n*
3. Map this node to both view's contexts.
4. Create an *InputField* in the *SliderCompView* and bind the value to the *x*-attribute.

Enable data flow between FlashIsland and Web Dynpro component

1. Switch back to the *FlashSlider* view and select the *Property1* element and set the following values:
   - Enter *xvalue* as *name*.
     The *name* property's value has to be identical to the variable's name of the Flash component!



Property	Value
⊟ Properties [GACProperty	
id	Property1
name	xvalue
readOnly	true
value	sliderData.x

   - Bind the *value* property to the *x* attribute.
2. For the *Event* view element set *change* as *name*. Ensure that this value is identical to the slider's event name!
3. Create an action as event handler to be able to trigger a roundtrip when the user triggers the event.
4. Assign this action also to the *InputField*'s *onEnter* event.

Running the application

1. Save your DC, choose *Deploy New Archive and Run* from the context menu.
2. Play around and have fun!

Related Content

Implementing Adobe Flash Islands for Web Dynpro Java

SDN Blog: SAP Libraries Required for Adobe Flash Islands in Web Dynpro

For more information, visit the User Interface Technology homepage.

Appendix:

Complete Flash code:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
      xmlns:mx="http://www.adobe.com/2006/mxml"
      layout="vertical"
      backgroundColor="#EAF1F6"
      width="100%"
      height="100%"
      initialize="initApp()"
      backgroundGradientColors="#EAF1F6">
<mx:Script>
      <![CDATA[
      import sap.FlashIsland;
      import mx.events.SliderEvent;
      import mx.controls.sliderClasses.Slider;
      [Bindable]
      public var xvalue:int;
      public function initApp():void
      {
            FlashIsland.register(this);
      }
      private function sliderChange(event:SliderEvent):void
      {
            var currentSlider:VSlider=event.currentTarget as VSlider;
            xvalue = currentSlider.value;
            FlashIsland.storeProperty(this,"xvalue",xvalue);
            FlashIsland.fireEvent(this, 'change', null );
      }
      ]]>
</mx:Script>
<mx:VSlider
      id="slideIn"
      value="{xvalue}"
      maximum="100"
      minimum="0"
      width="50"
      liveDragging="false"
      change="sliderChange(event)">
</mx:VSlider>
</mx:Application>
```

Copyright

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.