



Java is a trademark of Sun Microsystems, Inc.

JavaOneSM

BOF-4146: Building a JSF Component with AJAX – It's easy!

Jim Driscoll & Ryan Lubke
Sun Microsystems
Project Mojarra – the JSF RI

Agenda

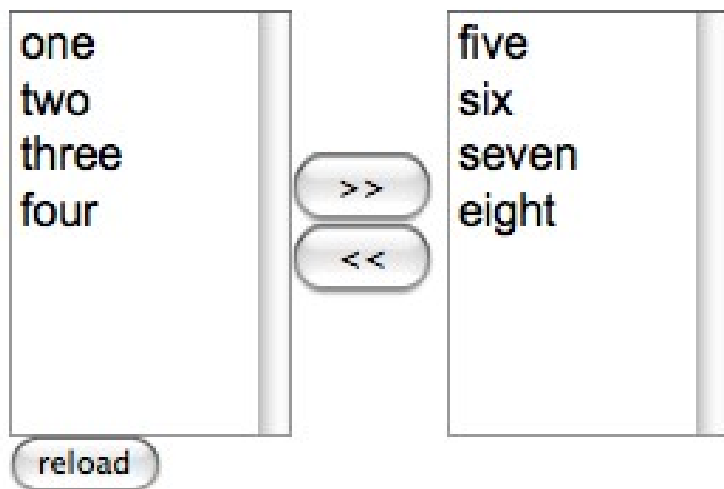
- > Intro
- > Switchlist aka Shuttle
 - In page implementation
 - Ajax implementation
 - Component
 - Component & Ajax
- > Live Code Demo – Zipcode lookup component
- > Spinner – a different problem (time permitting)

Features of JSF 2 we'll cover

- > A couple annotations
- > A little Facelets
- > `<f:ajax>` tag
- > Resources
- > Composite components, and `#{cc}`, the implicit EL object

Switchlist (aka “Shuttle”)

Switchlist Example



- > Two lists, with controls to move values
- > Need two lists, for selected values
- > Need two maps, for list contents
- > Need two Action methods to move values

Switchlist Component Backing Bean

`@ManagedBean(name="switchlist")`

`@SessionScoped`

```
public class SwitchlistBean implements Serializable {
```

```
    private Map<String, String> items1 = new LinkedHashMap<String, String>();
```

```
    private Map<String, String> items2 = new LinkedHashMap<String, String>();
```

```
    private String[] list1 = null;
```

```
    private String[] list2 = null;
```

```
    { items1.put("one", "one");
```

```
      items1.put("two", "two");
```

```
      items1.put("three", "three");
```

```
      items1.put("four", "four");    } // and the same for items2
```

Plus associated Getters and Setters...

Switchlist Component Backing Bean

```
public void move1to2(ActionEvent ae) {
    if (list1 != null && list1.length > 0) {
        for (String item : list1 ) {
            items2.put(item, items1.remove(item));
        }
    }
}
```

```
public void move2to1(ActionEvent ae) {
    if (list2 != null && list2.length > 0) {
        for (String item : list2 ) {
            items1.put(item, items2.remove(item));
        }
    }
}
```

Switchlist in Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
<h:head>
    <title>Switchlist Example</title>
</h:head>
<h:body>
    <h1>Switchlist Example</h1>
    <h:form id="form1">
        <h:outputStylesheet name="switchlist.css"/>
```

Switchlist in Page

```
<h:selectManyListbox value="#{switchlist.list1}" styleClass="switchlist">
  <f:selectItems value="#{switchlist.items1}"/>
</h:selectManyListbox>
<h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
  <h:commandButton value="&gt;&gt;"
actionListener="#{switchlist.move1to2}" styleClass="switchlistButton" />
  <h:commandButton value="&lt;&lt;"
actionListener="#{switchlist.move2to1}" styleClass="switchlistButton" />
</h:panelGroup>
<h:selectManyListbox value="#{switchlist.list2}" styleClass="switchlist">
  <f:selectItems value="#{switchlist.items2}"/>
</h:selectManyListbox>
</h:form>
</h:body>
</html>
```

Switchlist in Page

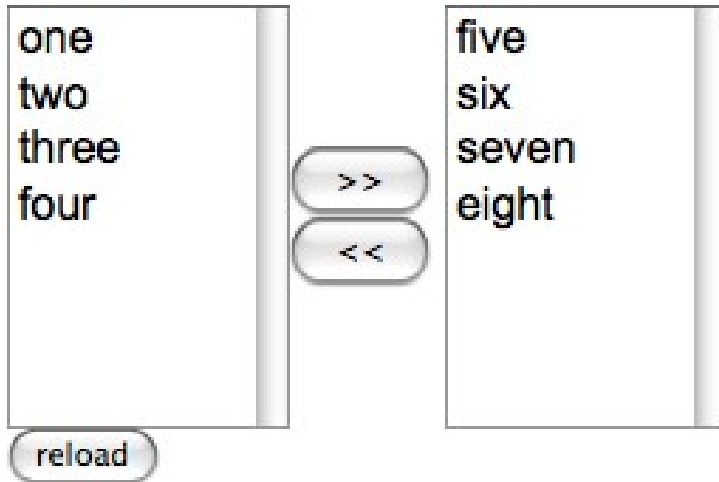
```
<h:selectManyListbox value="#{switchlist.list1}" styleClass="switchlist">
  <f:selectItems value="#{switchlist.items1}"/>
</h:selectManyListbox>
<h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
  <h:commandButton value="&gt;&gt;"
actionListener="#{switchlist.move1to2}" styleClass="switchlistButton" />
  <h:commandButton value="&lt;&lt;"
actionListener="#{switchlist.move2to1}" styleClass="switchlistButton" />
</h:panelGroup>
<h:selectManyListbox value="#{switchlist.list2}" styleClass="switchlist">
  <f:selectItems value="#{switchlist.items2}"/>
</h:selectManyListbox>
</h:form>
</h:body>
</html>
```

Backtracking a bit - Switchlist in Page

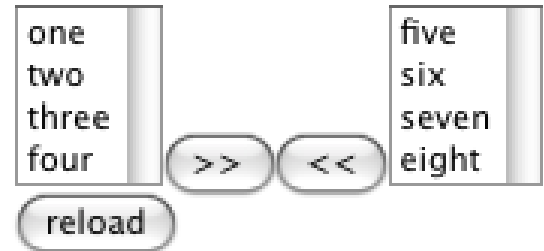
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core">
<h:head>
  <title>Switchlist Example</title>
</h:head>
<h:body>
  <h1>Switchlist Example</h1>
  <h:form id="form1">
    <h:outputStylesheet name="switchlist.css"/>
```

With and without Styling

Switchlist Example



Switchlist Example



Resources

- > `h:outputStylesheet` calls a resource
- > Resources are placed in `{context}/resources`
- > You can use resources for libraries, images, css, javascript and composite components
 - Lots of options, this is a simple case
- > So when you call
 - `<h:outputStylesheet name="switchlist.css">`
 - It serves the file `{context}/resources/switchlist.css`

resources/switchlist.css

```
.switchlist {  
    font-size: medium;  
    font-family: Arial, sans-serif;  
    height: 150px;  
    width: 100px;  
}  
.switchlistButtons {  
    width: 55px;  
    display: inline-block;  
    margin-top: 50px;  
    vertical-align: top;  
}  
.switchlistButton {  
    width: 50px;  
    height: 25px;  
}
```

Just one problem

- > As implemented, you're getting a full page refresh every time you click a button
- > Let's fix that, by adding the `<f:ajax>` tag

Adding the Ajax tag

```
<h:panelGroup id="buttonGroup">
  <h:commandButton id="button1" value="&gt;&gt;"
    actionListener="#{listholder.move1to2}">
    <f:ajax execute="@this list1" render="list1 list2"/>
  </h:commandButton>
  <h:commandButton id="button2" value="&lt;&lt;"
    actionListener="#{listholder.move2to1}" >
    <f:ajax execute="@this list2" render="list1 list2"/>
  </h:commandButton>
</h:panelGroup>
```

About Execute and Render

- > Execute is where you do:
 - Get the new values from the page
 - Validation
 - Pushing the values into the backing beans
 - Execute any listener/controller logic
- > Render is where:
 - The results are “drawn”, or rendered, to the browser

What the f:ajax tag does

- > f:ajax modifies the tag that wraps it
 - Or the tag(s) it wraps
- > Requests are sent via Ajax
- > Execute attribute says to evaluate component(s) (both the button and the list)
- > Render says to redraw component(s) (both lists)
- > We could have just “cheated” and instead said:
 - `<f:ajax execute="@form" render="@form"/>`

Making a component

- > First we'll do this without Ajax, then add it later
- > Composing a component with other components is similar to making a template.
- > Create a file in the resources directory, with the name of the component library and of the component
 - So, for us,
`{context}/resources/switchlist/switchlist.xhtml`

In the using page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ez="http://java.sun.com/jsf/composite/switchlist">
<h:head>
    <title>Switchlist Example</title>
</h:head>
<h:body>
    <h1>Switchlist Example</h1>
    <h:form id="form1">
        <ez:switchlist id="switchlist"
            selected1="#{listholder.list1}" selected2="#{listholder.list2}"
            items1="#{listholder.items1}" items2="#{listholder.items2}"
            move1to2="#{listholder.move1to2}" move2to1="#{listholder.move2to1}"/>
    </h:form>
</h:body>
</html>
```

In the component, part 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:cc="http://java.sun.com/jsf/composite">
<head><title>not output, here for validation only</title></head>
<body>
<cc:interface>
    <cc:attribute name="selected1" />
    <cc:attribute name="selected2" />
    <cc:attribute name="items1" />
    <cc:attribute name="items2" />
    <cc:attribute name="move1to2" targets="move1to2" method-signature="void
f1(javax.faces.event.ActionEvent)" />
    <cc:attribute name="move2to1" targets="move2to1" method-signature="void
f2(javax.faces.event.ActionEvent)" />
</cc:interface>
```

In the component, part 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:cc="http://java.sun.com/jsf/composite">
<head><title>not output, here for validation only</title></head>
<body>
<cc:interface>
    <cc:attribute name="selected1" />
    <cc:attribute name="selected2" />
    <cc:attribute name="items1" />
    <cc:attribute name="items2" />
    <cc:attribute name="move1to2" targets="move1to2" method-signature="void
f1(javax.faces.event.ActionEvent)" />
    <cc:attribute name="move2to1" targets="move2to1" method-signature="void
f2(javax.faces.event.ActionEvent)" />
</cc:interface>
```

In the component, part 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:cc="http://java.sun.com/jsf/composite">
<head><title>not output, here for validation only</title></head>
<body>
<cc:interface>
    <cc:attribute name="selected1" />
    <cc:attribute name="selected2" />
    <cc:attribute name="items1" />
    <cc:attribute name="items2" />
    <cc:attribute name="move1to2" targets="move1to2" method-signature="void
f1(javax.faces.event.ActionEvent)" />
    <cc:attribute name="move2to1" targets="move2to1" method-signature="void
f2(javax.faces.event.ActionEvent)" />
</cc:interface>
```

In the component, part 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:cc="http://java.sun.com/jsf/composite">
<head><title>not output, here for validation only</title></head>
<body>
<cc:interface>
    <cc:attribute name="selected1" />
    <cc:attribute name="selected2" />
    <cc:attribute name="items1" />
    <cc:attribute name="items2" />
    <cc:attribute name="move1to2" targets="move1to2" method-signature="void
f1(javax.faces.event.ActionEvent)" />
    <cc:attribute name="move2to1" targets="move2to1" method-signature="void
f2(javax.faces.event.ActionEvent)" />
</cc:interface>
```

In the component, part 2

```
<cc:implementation>
```

```
  <div id="{cc.clientId}">
    <h:outputStylesheet name="switchlist/switchlist.css"/>
    <h:selectManyListbox value="{cc.attrs.selected1}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items1}"/>
    </h:selectManyListbox>
    <h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
      <h:commandButton id="move1to2" value="&gt;&gt;"
actionListener="{cc.attrs.move1to2}" styleClass="switchlistButton"/>
      <h:commandButton id="move2to1" value="&lt;&lt;"
actionListener="{cc.attrs.move2to1}" styleClass="switchlistButton"/>
    </h:panelGroup>
    <h:selectManyListbox value="{cc.attrs.selected2}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items2}"/>
    </h:selectManyListbox>
  </div>
```

```
</cc:implementation>
```

```
</body>
```

```
</html>
```

In the component, part 2

```
<cc:implementation>
  <div id="{cc.clientId}">
    <h:outputStylesheet name="switchlist/switchlist.css"/>
    <h:selectManyListbox value="{cc.attrs.selected1}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items1}"/>
    </h:selectManyListbox>
    <h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
      <h:commandButton id="move1to2" value="&gt;&gt;"
actionListener="{cc.attrs.move1to2}" styleClass="switchlistButton"/>
      <h:commandButton id="move2to1" value="&lt;&lt;"
actionListener="{cc.attrs.move2to1}" styleClass="switchlistButton"/>
    </h:panelGroup>
    <h:selectManyListbox value="{cc.attrs.selected2}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items2}"/>
    </h:selectManyListbox>
  </div>
</cc:implementation>
</body>
</html>
```

In the component, part 2

```
<cc:implementation>
  <div id="{cc.clientId}">
    <h:outputStylesheet name="switchlist/switchlist.css"/>
    <h:selectManyListbox value="{cc.attrs.selected1}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items1}"/>
    </h:selectManyListbox>
    <h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
      <h:commandButton id="move1to2" value="&gt;&gt;"
actionListener="{cc.attrs.move1to2}" styleClass="switchlistButton"/>
      <h:commandButton id="move2to1" value="&lt;&lt;"
actionListener="{cc.attrs.move2to1}" styleClass="switchlistButton"/>
    </h:panelGroup>
    <h:selectManyListbox value="{cc.attrs.selected2}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items2}"/>
    </h:selectManyListbox>
  </div>
</cc:implementation>
</body>
</html>
```

In the component, part 2

```
<cc:implementation>
  <div id="{cc.clientId}">
    <h:outputStylesheet name="switchlist/switchlist.css"/>
    <h:selectManyListbox value="{cc.attrs.selected1}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items1}"/>
    </h:selectManyListbox>
    <h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
      <h:commandButton id="move1to2" value="&gt;&gt;"
actionListener="{cc.attrs.move1to2}" styleClass="switchlistButton"/>
      <h:commandButton id="move2to1" value="&lt;&lt;"
actionListener="{cc.attrs.move2to1}" styleClass="switchlistButton"/>
    </h:panelGroup>
    <h:selectManyListbox value="{cc.attrs.selected2}" styleClass="switchlist">
      <f:selectItems value="{cc.attrs.items2}"/>
    </h:selectManyListbox>
  </div>
</cc:implementation>
</body>
</html>
```

In the component, part 2

```
<cc:implementation>
  <div id="#{cc.clientId}">
    <h:outputStylesheet name="switchlist/switchlist.css"/>
    <h:selectManyListbox value="#{cc.attrs.selected1}" styleClass="switchlist">
      <f:selectItems value="#{cc.attrs.items1}"/>
    </h:selectManyListbox>
    <h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
      <h:commandButton id="move1to2" value="&gt;&gt;"
actionListener="#{cc.attrs.move1to2}" styleClass="switchlistButton"/>
      <h:commandButton id="move2to1" value="&lt;&lt;"
actionListener="#{cc.attrs.move2to1}" styleClass="switchlistButton"/>
    </h:panelGroup>
    <h:selectManyListbox value="#{cc.attrs.selected2}" styleClass="switchlist">
      <f:selectItems value="#{cc.attrs.items2}"/>
    </h:selectManyListbox>
  </div>
</cc:implementation>
</body>
</html>
```

Next revision

- > We'll add an ajax tag to have it not do a full page refresh
- > We'll change the component to use only two attributes
- > We'll change the backing bean to separate out functionality in a better way (MVC)
- > We'll put the switchlist component in a new library, named switchlistajax, so now:
 - `{context}/resources/switchlistajax/switchlist.xhtml`

Rework the backing classes

- > A Switchlist controller class
 - Hide the move methods from the enduser
- > A ListHolder interface
 - Needed for the controller
- > Two data holding classes, implementing ListHolder

ListHolder interface

```
package switchlist;

import java.util.Map;

public interface ListHolder {

    public String[] getList();

    public void setList(String[] list);

    public Map<String, String> getItems();

}
```

ListHolder1 (of 2) model bean

```
@ManagedBean(name="listholder1")
```

```
@SessionScoped
```

```
public class ListHolder1 implements ListHolder, Serializable {
```

```
    String[] list = null;
```

```
    Map<String, String> items = new LinkedHashMap<String, String>();
```

```
    // ... initializing items here ... //
```

```
    public String[] getList() {
```

```
        return list;
```

```
    }
```

```
    public void setList(String[] list) {
```

```
        this.list = list;
```

```
    }
```

```
    public Map<String, String> getItems() {
```

```
        return items;
```

```
    }
```

```
}
```

SwitchlistController Controller

```
@ManagedBean
@RequestScoped
public class SwitchlistController implements Serializable {
    ListHolder listholder1, listholder2;
    public String m1_2() {
        String[] list1 = listholder1.getList();
        Map<String, String> items2 = listholder2.getItems();
        Map<String, String> items1 = listholder1.getItems();
        if (list1 != null && list1.length > 0) {
            for (String item : list1) {
                items2.put(item, items1.remove(item));
            }
        }
        return null;
    }
    public void setListHolder1(ListHolder listholder1) {
        this.listholder1 = listholder1;
    }
}
```

Using Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ez="http://java.sun.com/jsf/composite/switchlistajax">
<h:head>
    <title>Switchlist Example</title>
</h:head>
<h:body>
    <h1>Switchlist Example</h1>
    <h:form id="form1">
        <ez:switchlist id="switchlist"
            listholder1="#{listholder1}" listholder2="#{listholder2}"/>
    </h:form>
</h:body>
</html>
```

In the component, part 1

```
<cc:interface>  
  <cc:attribute name="listholder1">  
    <cc:attribute name="list"/>  
    <cc:attribute name="items"/>  
  </cc:attribute>  
  <cc:attribute name="listholder2" >  
    <cc:attribute name="list"/>  
    <cc:attribute name="items"/>  
  </cc:attribute>  
</cc:interface>
```

In the component, part 1

```
<cc:interface>  
  <cc:attribute name="listholder1">  
    <cc:attribute name="list"/>  
    <cc:attribute name="items"/>  
  </cc:attribute>  
  <cc:attribute name="listholder2" >  
    <cc:attribute name="list"/>  
    <cc:attribute name="items"/>  
  </cc:attribute>  
</cc:interface>
```

In the component, part 2

```
<h:selectManyListbox id="list1" value="#{cc.attrs.listholder1.list}"
styleClass="switchlist">
  <f:selectItems value="#{cc.attrs.listholder1.items}"/>
</h:selectManyListbox>
<h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
  <h:commandButton id="move1to2" value="&gt;&gt;"
action="#{switchlistController.m1_2}" styleClass="switchlistButton">
  <f:setPropertyActionListener value="#{listholder1}"
target="#{switchlistController.listHolder1}"/>
  <f:setPropertyActionListener value="#{listholder2}"
target="#{switchlistController.listHolder2}"/>
  <f:ajax execute="@this list1" render="list1 list2"/>
</h:commandButton>
```

.... more code here, in a similar fashion

```
</cc:implementation>
```

In the component, part 2

```
<h:selectManyListbox id="list1" value="#{cc.attrs.listholder1.list}"
styleClass="switchlist">
  <f:selectItems value="#{cc.attrs.listholder1.items}"/>
</h:selectManyListbox>
<h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
  <h:commandButton id="move1to2" value="&gt;&gt;"
action="#{switchlistController.m1_2}" styleClass="switchlistButton">
  <f:setPropertyActionListener value="#{listholder1}"
target="#{switchlistController.listHolder1}"/>
  <f:setPropertyActionListener value="#{listholder2}"
target="#{switchlistController.listHolder2}"/>
  <f:ajax execute="@this list1" render="list1 list2"/>
</h:commandButton>
```

.... more code here, in a similar fashion

```
</cc:implementation>
```

In the component, part 2

```
<h:selectManyListbox id="list1" value="#{cc.attrs.listholder1.list}"
styleClass="switchlist">
  <f:selectItems value="#{cc.attrs.listholder1.items}"/>
</h:selectManyListbox>
<h:panelGroup id="buttonGroup" styleClass="switchlistButtons">
  <h:commandButton id="move1to2" value="&gt;&gt;"
action="#{switchlistController.m1_2}" styleClass="switchlistButton">
  <f:setPropertyActionListener value="#{listholder1}"
target="#{switchlistController.listHolder1}"/>
  <f:setPropertyActionListener value="#{listholder2}"
target="#{switchlistController.listHolder2}"/>
  <f:ajax execute="@this list1" render="list1 list2"/>
</h:commandButton>
```

.... more code here, in a similar fashion

```
</cc:implementation>
```

You can now make Ajax components

- > That wasn't so bad, was it? No JavaScript!
- > We covered:
 - A small amount of Facelets
 - A little bit of Annotations
 - The basics of the f:ajax tag
 - Basic use of the resources directory
 - Creating a composite component
 - The `#{cc}` implicit EL object

Live Coding Demo

- > Address component with zip lookup

Another trick to know

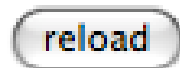
- > Eventually, you'll use JavaScript in a component
 - The ajax tag can only take you so far
- > When using an external JavaScript file, it's only served **once**. That means that including it in your a component gets harder.
 - How will you keep track of the clientid?
- > There's an easy solution, fortunately – keep track of the ID of your component inside the component itself, with context or state.
- > Let's review another component to see how...

The Spinner Component

Spinner Example



Last number submitted: 0



- > An input tag that takes a value, and an increment
- > When you press the up or down arrow, the number is incremented or decremented.
- > No Ajax, just JavaScript

Using Page (edited for brevity)

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ez="http://java.sun.com/jsf/composite/spinnerStyled">
```

..... *code omitted*

```
<h:form id="form1" prependId="false">
  <ez:spinner value="#{number.number}" increment="10" id="spinner" />
  <br/>
  <h:outputText value="Last number submitted: #{number.number}" />
  <br/>
  <h:commandButton value="reload" type="submit"/>
  <h:messages/>
</h:form>
```

.... *code omitted*

Spinner Component - interface

```
<composite:interface>  
  <composite:attribute name="value" />  
  <composite:attribute name="increment" />  
</composite:interface>
```

Spinner Component – Implementation

Part 1

```
<composite:implementation>
  <span id="{cc.clientId}"
  <h:inputText id="number" value="{cc.attrs.value}"/>
  <h:panelGroup id="spinnerButtonPanel" >
    <h:commandButton id="forward" value="#652;"
      onclick="return changeNumber(1);" />
    <h:commandButton id="back" value="v"
      onclick="return changeNumber(-1);" />
  </h:panelGroup>
```

Spinner Component – Implementation

Part 2

```
<script type="text/javascript">
  function changeNumber(direction) {
    var increment = Number("#{cc.attrs.increment}");
    if (isNaN(increment) || increment == 0 ) { increment = 1; }
    var entry =
      document.getElementById("#{cc.clientId}"+":"+ "number");
    var val = Number(entry.value);
    if (isNaN(val)) { val = 0; }
    entry.value = val + (direction * increment);
    return false;
  }
</script>
</span>
</composite:implementation>
```

Just one problem with this example

- > It only works one per page, since
 - JavaScript would be included each time
 - The `cc.clientId` would be set to different values
- > Easy to fix – separate out JavaScript into a file
 - Resource loading handles multiple requests automatically
- > But you need to pass the `cc.clientId` at some point, and remember it in state.

Spinner Component – Implementation With External JavaScript

```
<cc:implementation>
  <span id="#{cc.clientId}">
    <h:outputScript name="spinnerFinal/spinner.js" target="head" />
    <script type="text/javascript">
      init("#{cc.clientId}", "#{cc.attrs.increment}");
    </script>
    <h:inputText id="number" value="#{cc.attrs.value}" />
    <h:panelGroup id="spinnerButtonPanel" >
      <h:commandButton id="forward" value="#652;"
        onclick="return changeNumber("#{cc.clientId}',1);" />
      <h:commandButton id="back" value="v"
        onclick="return changeNumber("#{cc.clientId}',-1);" />
    </h:panelGroup>
  </span>
</cc:implementation>
```

Spinner Component – External JS File

```
if (!window["spinnerInc"]) {  
    var spinnerInc = {};  
}
```

```
function init(cid, increment) {  
    spinnerInc[cid] = Number(increment);  
    if (isNaN(spinnerInc[cid]) || spinnerInc[cid] == 0 ) {  
        spinnerInc[cid]= 1;  
    }  
}
```

```
function changeNumber(cid, direction) {  
    var entry = document.getElementById(cid+":"+ "number");  
    var val = Number(entry.value);  
    if (isNaN(val)) { val = 0; }  
    entry.value = val + (direction * spinnerInc[cid]);  
    return false;  
}
```

For more information on JSF 2

- > Jim Driscoll's blog:
<http://weblogs.java.net/blog/driscoll/>
- > Ryan Lubke's blog: <http://blogs.sun.com/rlubke/>
- > webtier@glassfish.dev.java.net
- > Project Mojarra - <http://mojarra.dev.java.net>
- > Project GlassFish - <http://glassfish.dev.java.net>



JavaOneSM

Thank You

Jim Driscoll
Ryan Lubke
webtier@glassfish.dev.java.net

