Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Advanced Web Technologi
6) Ajax in JSF2.x (and some other rarities)

Emmanuel Benoist

Fall Term 2016-17

Computer Science Division

Ajax and JSF2.x

- Ajax
  - AJAX Principles
  - AJAX example in PHP
- Ajax in JSF
  - JSF1.2 vs JSF 2.0
  - The f:ajax tag

Ajax

AJAX Principles

Ajax Principles

AJAX application life cycle.

- Use Javascript for collecting information
- Create a HTTP Request (containing a random number in order to avoid caching)
- Send this request and organize a handler for being executed after the reception of the response.
- Display the results inside the DOM.
- ...

AJAX example in PHP

AJAX Example

- **We have a Form containing a selection box**
- **On Change of the selection, the function `showCustomer()` is executed**
- **The function creates an Object (XMLHttpRequest or its MS-cousins)**
- **A request is sent to a PHP file,**
- **The PHP program generates a Table**
- **The table is included in the html DOM.**

The form containing a selection

```
<form>
Select a Customer:
<select name="customer" onchange="showCustomer(this.value↘
→)">
<option value="ALFKI">Alfreds Futterkiste
<option value="NORTS">North/South
<option value="WOLZA">Wolski Zajazd
</select>
</form>
<p>
<div id="txtHint"><b>Customer info will be listed here.</b↘
→></div>
</p>
```

Show Customer

We create a Request and send it using the XML/HTTP object

```
function showCustomer(str) {
  xmlHttp=GetXmlHttpObject();
  if (xmlHttp==null) {
    alert ("Your browser does not support AJAX!");
    return;
  }
  var url="getcustomer.php";
  url=url+"?q="+str;
  url=url+"&sid="+Math.random();
  xmlHttp.onreadystatechange=stateChanged;
  xmlHttp.open("GET",url,true);
  xmlHttp.send(null);
}
```

Function for creating the XML/HTTP object

```
function GetXmlHttpObject()
{
 var xmlHttp=null;
 try { // Firefox, Opera 8.0+, Safari
   xmlHttp=new XMLHttpRequest();
 }
 catch (e) { // Internet Explorer
     try {
         xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
     }
     catch (e) {
         xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
     }
 }
 return xmlHttp;
}
```

Handler for the Response

- **Each Time the state change, the handler function is executed:**
- It should only react for state 4 since it means: response received.

```
function stateChanged() {
  if (xmlHttp.readyState==4){
    document.getElementById("txtHint").innerHTML=
              xmlHttp.responseText;
  }
}
```

The PHP Program used to generate the response

- **The program reads the informtion and generate a response:**

```
$q=$_GET["q"];
$db_table = array(...);
echo "<table border='1'>
<tr> <th>Firstname</th> <th>Lastname</th></tr>";
$row = $db_table[$q];
 echo "<tr> echo " <td>" .. $row['FirstName'] .. "</td>";
 echo "<td>" .. $row['LastName'] .. "</td></tr>";
 echo "</table>";
```

Ajax in JSF

JSF1.2 vs JSF 2.x

Ajax in JSF

- **JSF 1.2**
  - Libraries of components
  - Designed for AJAX support (javascript and so on)
  - Ajax4jsf or RichFaces
- **JSF2.x**
  - Inherits from RichFaces functionalities
  - Part of the default framework

Principles

- **JSF 2.x provides a set of predifined tags**
  - Offer a native AJAX support.
  - `f:ajax`
- **Partial Rendering of the page**
  - For some events (specified in tag)
  - Form is sent to the server
  - Server executes life cycle
  - Rerenders only some elements

The f:ajax tag

Sending an Ajax Request

- **Use the tag `f:ajax`**
- **Is a client side behaviour**
  - it is always added as a child tag (behavior) to another UI component

```
<h:form>
    <h:panelGrid>
        <h:inputText value="#{bean.text}" >
            <f:ajax event="keyup" />
        </h:inputText>
        <h:outputText id="text" value="#{bean.text}" />
    </h:panelGrid>
</h:form>
```

Description of The Event Attribute

- **Event:**
  - String on which event Ajax request will be fired.
  - If not specified, a default behavior based on parent component will be applied.
  - The default event is action for ActionSource (ie: button) components and valueChange for EditableValueHolder components (ie: input).
  - action and valueChange are actual String values that can be applied applied event attribute.
- **For the event attribute you do not specify the "onevent" (onkeyup) like in JavaScript, but only the event (`keyup`)**
  - Each component has a default event (if no event is specified)
  - For `h:inputText` it is `onchange`

Partial View Rendering

- **Only re-render the components that need it**
  - Specify to the ajax component which component(s) need to be re-rendered
- **The partial view is re-rendered on server**
  - The part of the form corresponding to the element is sent
  - The corresponding component tree is updated, and the corresponding backing beans,
  - The partial view is re-rendered
  - Only the required part of the DOM is sent in XML

Partial View Rendering (Cont.)

```
<h:form>
    <h:panelGrid>
        <h:inputText value="#{bean.text}" >
            <f:ajax event="keyup" render="text" />
        </h:inputText>
        <h:outputText id="text" value="#{bean.text}" />
    </h:panelGrid>
</h:form>
```

`http://localhost:8080/ajax2.0/faces/softSec.xhtml`

Execute a specific method (listener)

- **Need the following bean**

```
@ManagedBean(name = "bean")
public class Bean {
    private String text; // getter and setter
    ...
    private Integer count; // getter and setter
    public void countListener(AjaxBehaviorEvent event) {
       count = text.length();
   }
}
```

Listener

- **The listener Attribute**
  - Contains the listener method to invoke during Ajax request
- **One can re-render many components, use space as sperator**

```
<h:form>
    <h:panelGrid>
        <h:inputText value="#{bean.text}" >
            <f:ajax event="keyup" render="text count"
                        listener="#{bean.countListener}" />
        </h:inputText>
        <h:outputText id="text" value="#{bean.text}" />
        <h:outputText id="count" value="#{bean.count}" />
    </h:panelGrid>
</h:form>
```

The render attribute

- **In our example, render pointed to actual component id we want to render back.**
- **render attribute could be set to the following values:**
  - `@all` Render all components in view
  - `@none` Render no component in view (= default)
  - `@this` Render only the component that triggered the Ajax request
  - `@form` Render all components within this form
  - id's One or more id's of components to be rendered
  - EL EL expression which resolves to Collection of Strings

Render Attribute with an EL expression

- **It is possible to determine which components to render in run time**
  - You bind renderer to `#{bean.renderComponents}`
  - Initial page is rendered, getRenderComponents() returns the list "compId1" and "compId2"
  - On next Ajax request, compId1 and compId2 will be rendered.
  - During this request, `#{bean.renderComponents}` could be changed in runtime. (new value `compId3` and `compId4`)
  - You also have to remember to rerender this control (in order to update the id's)
  - The new values are now present in the page
  - By the next ajax request, those components will be re-rendered.

Partial View Processing

- **Normally in JSF (without Ajax)**
  - The entire form is processed on the server
  - All components go through all phases (Apply Request Values, Process Validation, Update Model)
- **With Ajax only the needed information is processed**
  - You can choose which components will get processed

```
<h:commandButton value="Click" >
  <f:ajax execute="@form" render="time" />
</h:commandButton>
```

Partial View Processing (Cont.)

- **Different values possible for** `execute`
  - `@all` : Process all components in view
  - `@none` : Process no components
  - `@this` : Process only this component
  - `@form` : Process all components within this form
  - `id's` : Id's of the components to be processed, space sparated
  - `EL` : An expression returning a Collection of Strings

More Examples

- **f:ajax can be around the controls:**
  - Uses the default behavioural events:
  - onChange for boolean check box
  - onChange for inputText
  - onClick for button
  - no behaviour with ajax for the panelGrid

```
<f:ajax>
  <h:panelGrid>
    <h:selectBooleanCheckbox>
    <h:inputText>
    <h:commandButton>
  </h:panelGrid>
</f:ajax>
```

More Examples (Cont.)

- **In the example:**
  - on click is added to panelGrid, inputText and commandButton.
  - commandButton would also have onfocus event applied to it.

```
<f:ajax event="click">
  <h:panelGrid>
    <h:selectBooleanCheckbox>
    <h:inputText>
    <h:commandButton>
     <f:ajax event="focus" />
    </h:commanButton>
  </h:panelGrid>
</f:ajax>
```

Conclusion - Ajax

- **AJAX makes web sites much more dynamic**
  - Contents can be updated in real time
- **Ajax in JSF**
  - Very easy
  - No change in the basic concepts (components, events, . . . are ajax-compatible)
- **BUT Ajax is not just a new technique, it is a change in the paradigme**
  - From Page oriented programming toward Event Oriented Programming
  - Form Web Site to Integrated Application
  - Navigation has to be new conceptualized.

References

- **JSF 2 fu, Part 3: Event handling, JavaScript, and Ajax**, David Geary, July 2009
  `http://www.ibm.com/developerworks/java/library/j-jsf2fu3/`
- **Learning JSF2: Ajax in JSF - using f:ajax tag** Max Katz, April 2010
  `http://mkblog.exadel.com/2010/04/learning-jsf-2-ajax-in-jsf-using-fajax-tag/`

Conclusion - JSF

- **Java Server Faces 2.x**
  - Is a Web Front-End for Java
  - Easy to integrate in Java environment
- **Web Tier : Easy to be written**
  - Web Pages using Templates (XHTML with facelets)
  - New useful components
  - Write your own library of components
- **Easy to integarate in Java**
  - Just need some Managed Beans for the glue
  - No web any more after it: no reference to Request or Response in Java program