JSF, Facelets

JSF Slides #2

Constructing the View

How to construct the JSF view (the server side client GUI component tree)?

Alt I: Coding Java
- Possible but tedious, boring, ...
- Must be a programmer

Alt II: Using some specific (more efficient) language
- Yes, ... **Facelets (JSF Pages)**
- More familiar for Web (page) developer

Facelets

Facelets is a view declaration (tree description) language
- Part of JSF 2.* specification
- Like XHTML(5) with some extra **JSF tag libraries**
- Must be valid XML
- Files: *.xhtml
- Using "(Unified) Expression Language" (embedded language to access beans ...)

Requesting a Facelets "page" will trigger construction of view tree (not returning the page). The (HTML) page is returned last in JSF cycle

Templating
- Possible to build modular pages

JSF tag libraries

JSF core <**f**:validateLength>, <**f**:param>,...
- General, utilities, ... non visible

JSF Standard HTML <**h**:link>, <**h**:form>, <**h**:commandButton>,...
- HTML components, forms.., if visible rendered as HTML controls

Facelets <**ui**: ... >
- Template tags, used to build modular pages, non visible

Composite component <**cc**: ....>, to build "larger" components, visible

JSTL tags (partial support, avoid), <**c:** ....>

Declaring Tag Libraries

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html ... >
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        ...
    </h:head>
    <h:body>
        ...
    </h:body>
</html>
```

JSF Standard HTML Tag Library

Main tag family to build the JSF pages (if not using higher level frameworks more to come...)

Somewhat old but pedagogical reference page
http://exadel.com/web/portal/jsftags-guide

Note : Attribute "required"

Handling Error Messages

Special tags for error messages
- <h:message..>, for specific component
- <h:messages..> (ending 's') all messages

Connect message to component

```
<h:inputText id="userName" ..../>
<h:message for="userName"... />
```

Possible to set messages in Java code, but we prefer Bean validation, more to come.. (see beansCDI)

The (Unified) Expression Language

Similar to EL in JSP's
- JSP/EL is immediate evaluation: JSP container immediately parses and resolves the expression when it processes the page and returns the response
- JSP/EL is read only
- JSP/EL syntax "${...}"

Can't do it like that in JSF
- Request are much more complicated, uses **deferred** evaluation, **the request cycle**
- JSF/UEL is read and write!
- JSF/UEL syntax "#{...}"

Referencing Objects using UEL

Managed Bean accessible in JSF-pages using the UEL (like JSP)

```
<!-- In a JSF page referencing an object -->
<...value="#{beanName}".../>
```

beanName is name of managed bean (Java object)
More to come...

UEL Value Expression

Get and set properties in beans, a **value expression**

```
// Will call bean.setData()
<h:inputText value="#{bean.data}"/>

// Will call bean.getData() (in fact a composite expr.)
<h:outputLabel value="Data on server:#{bean.data}"/>
```

A value expression can be a left or a right value

UEL Method Expressions

Call beans methods, method expression

```
<!-- Will call method click(ActionEvent e) -->
<h:commandLink … actionListener=”#{bean.click}” />
<h:commandButton … actionListener=”#{bean.click}” />
```

```
<!-- Will call method(ValueChangeEvent e) -->
<!-- All editable value holders -->
<h:inputText … valueChangeListener=”#{bean.method}”./>
```

A method expression is a right value

UEL Method Expression with Parameters

Possible to call arbitrary method

```
 // Ok, passing a String
<..."#{anyName.someMethod('abc'}".../>

// Ok passing a number
<..."#{anyName.otherMethod(123)}".../>
```

A lot of things happens in background, type coercions, ...
- Note: EL doesn't support overloading

If no parameter in call JSF will possible supply an event (for ActionEvent, ValueChangeEvent, ...)

Basic Facelets Page (in *.xhtml)

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>readWriteBean.xhtml</title>
    </h:head>
    <h:body>
        <h1>Basic page</h1>
        <h:form>
            <h:outputLabel for="txtData">Input data </h:outputLabel>
            <h:inputText id="txtData" value="#{dataBean.data}"/>
            <h:commandButton id="btnSend" value="OK"/>
        </h:form>
        <h:outputLabel id="lblOut" value="Serverdata is now:
                                #{DataBean.data}" />
    </h:body>
</html>
```

JSF Implicit Objects

Mostly same as JSP page implicit objects

Some special
- #{facesContext}, FacesContext instance for current request
- #{view}, viewroot for current component tree

Possible to get the current viewId (possible later use, "to get back")
- #{view.viewId}

The Flash Object

Another implicit object
- Represents a short term storage (alt. to session scope)
- Propagated across a single view transition (object will survive redirect). Useful in PRG (but other ways, better more to come, see CDI conversational scope)
- Will clean up before yet another view

Use in pages

```
<c:set target="#{flash}" property="returnPage"
                              value="#{view.viewId}" />
```

Access in Java code

```
FacesContext....getExternalContext.getFlash();
```

Requests

GET's: <h:link ../>, <h:button../>

```
<h:link outcome="...destination URI..." value="..link text..."  >
    <!-- Optional (will add ?reqData=...) -->
    <f:param name="reqData" value="#{myBean.data}"/>
 </h:link>
```

POST's <h:commandButton../>, <h:commandLink.. />

```
<h:form>
    <h:inputText value="#{myBean.data}" .../>
     <h:commandButton value="OK" ..../> <!-- default = submit -->
</h:form>
```

View Parameters

Special case of GET's

Possible to assign request parameters directly to bean properties

```
<!-- In target page (entry.jsf) -->
<h:body>
  <f:metadata>
   <f:viewParam name="id" value="#{blog.entryId}"/>
  </f:metadata>
...
```

Request http://domain/blog/entry.jsf?id=9, will set bean attribute entryId to 9

Have impact on request cycle, see code samples

Request URI's

The physical pages are always *.xhtml -files

Remainder: The request URI is mapped (in web.xml)

Possible mappings of requests

```
http://.../faces/shop.xhtml    (path mapping /faces/)
http://...shop.faces           (suffix mapping, *.faces)
http://...shop.xhtml           (suffix mapping, *.xhtml)
```

...

..all will pass FacesServlet and end up showing the shop.xhtml page

Facelets Templating

To reuse common content (refactoring)
- Uniform layout

Template file
- Facelets file (JSF Page)
- The overall layout and style (CSS), defines where to insert content
- Holds common parts (main menu, ...)

Template client
- Another Facelets file/page
- The content to insert

Multilevel templating possible

Template File

Define the "named" slots, ... where to insert

```
<ui:insert name="left" >
```

Also possible to just "include" some content (possible dynamic) compare JSP's

```
<ui:include src="/WEB-INF/inc-content/header.jspx"/>

<ui:include src="#{navigation.selectedPanel.
                        menuContentInclusionFile}"/>
```

Template Client

Define page specific content to insert into slots
- Specify which template to use
- Match the name of the slot in template file

```
<ui:composition template="pathToTemplate">
    <ui:define name="left">
        ... Content here ...
    </ui:define>
</ui:composition>
```

Will be inserted in slot "left" in template

NetBeans wizard will handle most of this...

Template Client Interaction

Possible to pass data between template and client

```
<!-- In client -->
<ui:composition template="pathToTemplate">
   <!-- Possible to send data to template -->
    <ui:param name=”paramToTemplate” value="..." />
</ui:composition>

<!-- In template, access data -->
<p>#{paramToTemplate}</p>
```

Templating Details

Using <ui:composition> everything outside composition trimmed

Using <ui:decorate>, same use as composition but no trimming

Facelets can't by default handle html comments (<!-- -->)???
- Use <ui:remove>... </ui:remove>
- .. or better modify web.xml, to enable HTML style comments, see code samples

Facelets and CSS

Probably best to link the CSS in the template page

Templating and URI's

User never access template file (page) directly
- Put below WEB-INF directory (private part of application)

As before, request URI is the client file (page)
- Facelets will handle the composition of template and client

JSF and Cleans URI's

No standard... as noted before

Have to rely on third party, PrettyFaces, others..

JSF Composite Components

Possible to build larger components
- A login component with two input fields and a button in a single tag

Special kind of Facelets page with
- <cc:interface>, interface to composite, how to use composite, attributes, actionSources, ...

- <cc:implementation>, implementation of composite (tags, input fields, buttons,..)

Files for composites resides in resources directory