JSF, Facelets, Composite Components and Suites

JSF Slides #4

Facelets

Facelets refers to the view declaration language for JavaServer Faces technology (confusing)
- I say templating system (and JSF pages)

Basically
- A way to compose pages
- Tags for composing
- Possible with data transfer between composed parts
- Composite components, build own custom components (a complete login screen, etc.)

Facelets Templating

To reuse common content, uniform layout

Template file

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

Template client

- The content to insert

Multilevel templating possible

[Facelets tag reference]

Template File

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

```
<!-- Theres some content named leftSlot, insert here -->
 <ui:insert name=”leftSlot” >
```

- 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 specific content to insert into slots
- Specify which template to use
- Match the name of the slot in template file
- Below will be inserted in slot "leftSlot" in template

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

- NetBeans wizard will handle most of this...

Client Data Transfer

Possible to pass data from client to template

```
<!-- 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 and URLs

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

Request URL is the client file (page)
- Facelets will handle the composition of template and client

Facelets Details

Using <ui:composition> everything outside composition trimmed

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

Also a tag for iteration (creating tables), <ui:repeat>

Facelets and Comments

Using <!-- … -->

- Will comment out but JSF still processes value expression and output the result to the generated HTML page

To really comment out use

```
// In web.xml
<context-param>
    <param-name>facelets.SKIP_COMMENTS</param-name>
    <param-value>true</param-value>
</context-param>
```

Or

```
<!-- In page -->
<ui:remove> … </ui:remove>
```

Facelets and CSS

No problems just link to

- Probably best to link CSS in template page

JSF Composite Components

Possible to build larger components (possible reuse)
- 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

Higher Level JSF Component Suites

Many higher level component suites built on top of JSF (see showcase/components for all)

- ICEFaces
- PrimeFaces
- OpenFaces

Higher level, very advanced, components

- Calendars, Datatables, Terminal, OS, many more…
- Tags for components
- NetBeans have support for PrimeFaces, ICEFaces, RichFaces, Properties > Frameworks > Components
- Probably shouldn't mix frameworks

PrimeFaces Datatable

```
<p:dataTable var="car" value="#{dtBasicView.cars}">
    <p:column headerText="Id">
        <h:outputText value="#{car.id}" />
    </p:column>

    <p:column headerText="Year">
        <h:outputText value="#{car.year}" />
    </p:column>

    <p:column headerText="Brand">
        <h:outputText value="#{car.brand}" />
    </p:column>

    <p:column headerText="Color">
        <h:outputText value="#{car.color}" />
    </p:column>
</p:dataTable>
```

Angular Faces

Why not combine Angular and JSF?

- Angular behaviour on client
- JSF on server …
- … AngularFaces!