Java Server Faces (JSF)

Table of Contents

Chapter 5

Facelets

Chapter 5 Outline

- Facelets and Facelets Tags
- Templating with Facelets
- Custom Tags
- Handling Relative URLs

What is Facelets ?

- Facelets was originally developed as an alternative to the JSP-based view handler in JSF 1.x
- In JSF 2.0, Facelets replaces JSP as JSF's default view technology
- Facelets supports tags for templating and other purposes
- Namespace to use Facelets tags

```
xmlns:ui="http://java.sun.com/jsf/facelets"
```

Facelets Tags

Facelets tags can be grouped in these categories:

- Including content from other XHTML pages (ui:include)
- Building pages from templates (ui:composition, ui:decorate, ui:insert, ui:define, ui:param)
- Creating custom components without writing Java code (ui:component, ui:fragment)
- Other utilities (ui:debug, ui:remove, ui:repeat)

Facelets Tags (cont'd)

- ui:include
  - Includes content from another XML file.
- ui:insert
  - Inserts content into a template. That content is defined inside the tag that loads the template.
- ui:define
  - Defines content that is inserted into a template with a matching ui:insert.

Facelets Tags (cont'd)

Greenshop

```
<div id="heading">
    <ui:insert name="heading">
        <ui:include src="/sections/jets/header.xtml"/>
    </ui:insert>
</div>
<ui:define name="content">
    If you are concerned about the environment ….
</ui:define>
</div>
<div id="content“>  <ui:insert name="content"/>  </div>
```

Need for Page Templating

- Avoiding repetition in facelets pages
  - facelets code want to avoid repeating nearly code there
  - Very common to have multiple pages that share the same essential layout and same general look
- Inadequacies of jsp:include
  - JSF pages normally avoid JSP tags
  - if you use jsp:include
    - No real templates or named sections
    - Can't easily pass data to included pages

Templating with Facelets (cont'd)

- Templating steps
  - Define a template file
  - Define a client file that uses the template

/templates/master.xhtml

shared content by all clients

```
<ui:insert name="title">Def
<ui:insert name="body">De
```

Insert shared content literally

Mark replaceable sections with **ui:insert**

Use **ui:composition** with template attribute

Use **ui:define** to supply content for sections

Home.xhtml

```
<ui:composition template="/templates/master.xhtml">
        <ui:define name="title">Title text</ui:define>
        <ui:define name="body">
                Content to go in "body" section of template
        </ui:define>
</ui:composition>
```

Templating with Facelets (cont'd)

- Template File

```
<!DOCTYPE ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>
<title>
  <ui:insert name="title">Default Title</ui:insert>
</title>
<link rel="stylesheet" type="text/css"
      href="./css/styles.css"/>
</h:head>
<h:body>
<table border="5" align="center"><tr><th class="title">
  <ui:insert name="title">Default Title</ui:insert>
</th></tr></table>
<h2>A random number: #{numGenerator.randomNum}</h2>
<ui:insert name="content">Default Content</ui:insert>
</h:body></html>
```

The parts not marked with ui:insert will appear in all client files. This can include dynamic content as with the random number below.

These sections can be replaced in client files via a ui:define that uses the same name

Templating with Facelets (cont'd)

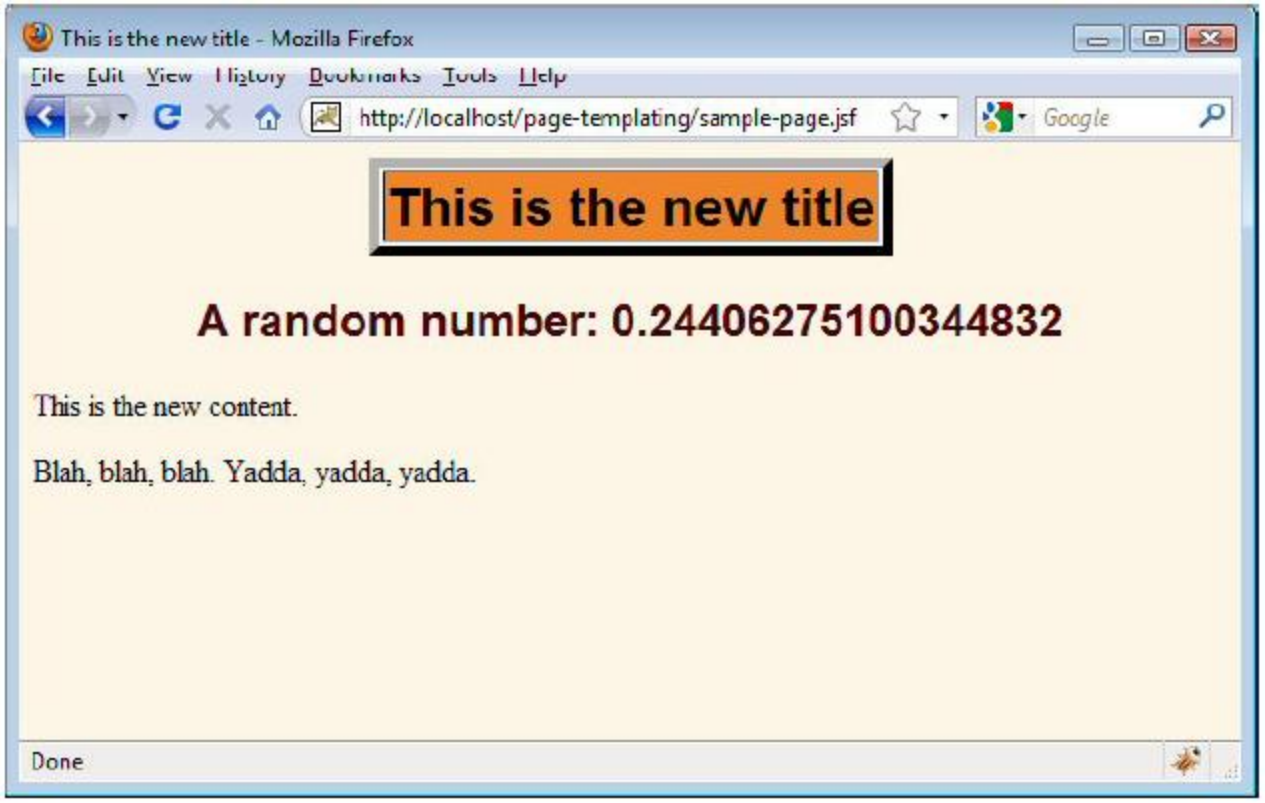
- Client File

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    template="/templates/sample-template.xhtml">
  <ui:define name="title">
    This is the new title
  </ui:define>
  <ui:define name="content">
    This is the new content.
    <p/>
    Blah, blah, blah. Yadda, yadda, yadda.
  </ui:define>
</ui:composition>
```

This client file does not directly use any of the h: tags, so xmlns:h=... is omitted from the start tag of ui:composition. But if h or f or c tag libraries are used directly in the client file, the schemas must be declared there.

Templating with Facelets (cont'd)

- Result



Organizing Views With Templates

- Content that is shared by all clients
  - Put it directly in template file
- Content that is specific to individual client
  - Put it in client file
- Content that is shared by some clients?
  - Problems
    - Not used by all clients, so can t' go directly in template file
    - Used by more than one client, so would be repetitive
  - Solution
    - Put reusable content in separate file
    - In client file, use **ui:include** in the body of **ui:define**

Templating with Facelets: Example

Pieces/DefaultMenu.xhtml

```
<ui:composition xmlns:h=http://java.sun.com/jsf/html
                xmlns:f=http://java.sun.com/jsf/core
                xmlns:ui=http://java.sun.com/jsf/facelets
                xmlns=http://www.w3.org/1999/xhtml >
   <div id="menu">
      <ul>
         <li id="h"><h:link outcome="Home" value="Home"></h:link></li>
         <li id="l"><h:link outcome="Login" value="Login"></h:link></li>
         <li id="r"><h:link  value="Register"></h:link></li>
         <li id="v"><h:link  outcome="Vote" value="Vote"></h:link></li>
         <li id="s"><h:link  value="Search"></h:link></li>
      </ul>
   </div>
</ui:composition>
```

Templating with Facelets: Example

```
<ui:composition     xmlns=http://www.w3.org/1999/xhtml
                    xmlns:ui=http://java.sun.com/jsf/facelets  >
   <div id="sidebar">
      <ul>
         <li>
            <h2>Inside Home+Garden</h2>
            <img alt="Home"   src="images/AtHome.jpg" />
         </li>
         <li>
            <h2>Categories</h2>
            <ul>
               <li><span>12004</span><a href="#">Clothing</a></li>
                        :
            </ul>
         </li>
      </ul>
   </div>
   <!-- end #sidebar -->
</ui:composition>
```

Pieces/SideBar.xhtml

Templating with Facelets: Example

Pieces/Header.xhtml

```
<ui:composition xmlns=http://www.w3.org/1999/xhtml
                xmlns:h=http://java.sun.com/jsf/html
                xmlns:f=http://java.sun.com/jsf/core
                xmlns:ui=http://java.sun.com/jsf/facelets >

    <div id="header">
        <div id="logo">
            <h1><a href="#">Greenshop</a></h1>
            <p>by JETS Team </p>
        </div>
    </div>
    <!-- end #header -->

</ui:composition>
```

Templating with Facelets: Example

Pieces/Footer.xhtml

```
<ui:composition xmlns:h=http://java.sun.com/jsf/html
                xmlns=http://www.w3.org/1999/xhtml
                xmlns:f=http://java.sun.com/jsf/core
                xmlns:ui=http://java.sun.com/jsf/facelets >

   <div id="footer-content">
      <center><p> (c) 2009 Sitename.com. Design by <a
           href="http://www.nodethirtythree.com">nodethirtythree</a> and <a
           href="http://www.freecsstemplates.org">Free CSS Templates</a>.</p>
      </center>
   </div>
   <!-- end #footer -->

</ui:composition>
```

Templating with Facelets: Example

Templates/masterLayout.xhtml

```
<div id="wrapper">
   <ui:include src="/pieces/Header.xhtml"/>
   <ui:include src="/pieces/DefaultMenu.xhtml"/>
   <div id="page">
      <div id="content">
         <div id="banner"><img src="images/img07.jpg" alt="" /></div>
         <div id="myPost" class="post">
            <h2 class="title"><a href="#">Greenshop</a></h2>
               <p class="meta"><ui:insert name="heading">Home and Garden</ui:insert></p>
            <div id="entry" class="entry">
               <ui:insert name="entry"></ui:insert>
            </div>
         </div>
      </div>
      <ui:include src="/pieces/SideBar.xhtml"/>
      <div style="clear: both;"/>
   </div>
</div>
<ui:include src="/pieces/Footer.xhtml"/>
<ui:insert name="activeTab"></ui:insert>
```

Templating with Facelets: Example

Home.xhtml

```
<ui:composition template="templates/masterLayout.xhtml">

    <ui:define name="activeTab">
        <script type="text/javascript">
            document.getElementById("h").className="current_page_item";
        </script>
    </ui:define>
</ui:composition>
```

Templating with Facelets: Example

Login.xhtml

```
<ui:composition template="templates/masterLayout.xhtml">
    <ui:define name="heading">Login Page</ui:define>
    <ui:define name="entry">
      <p>Please enter your Login Data</p>
      <table>
          :
      </table>
    </ui:define>
    <ui:define name="activeTab">
      <script type="text/javascript">
        document.getElementById("l").className="current_page_item";
      </script>
    </ui:define>
</ui:composition>
```

Templating with Facelets: Example

Vote.xhtml

```
<ui:composition template="templates/masterLayout.xhtml">
    <ui:define name="heading">Vote Page</ui:define>
    <ui:define name="entry">
      <p>Please Choose yOur favOrite CategOry </p>
      <table>
                :
      </table>
    </ui:define>
    <ui:define name="activeTab">
      <script type="text/javascript">
        document.getElementById("v").className="current_page_item";
      </script>
    </ui:define>
  </ui:composition>
```

Templating with Facelets: Decorators

- Tiles approach gives you a lot of flexibility when you have a complex set of pages
- But for a simple application !!

- Decorators are a more content-centric approach
- "template-in-template".
- You write your pages as usual, but you surround the contents with a <ui:decorate>
- With decorators, as with compositions, you can override defaults with <ui:define> tags
- tags outside the <ui:decorate> are not trimmed off

Templating with Facelets: Example

Templates/masterLayout.xhtml

```
<ui:composition>
    <f:view>
     <f:form>
        <div id="wrapper">
            <ui:include src="/pieces/Header.xhtml"/>
            <ui:include src="/pieces/DefaultMenu.xhtml"/>
            <div id="page">
                <div id="content">
                        :
                </div>
                <ui:include src="/pieces/SideBar.xhtml"/>
                <div style="clear: both;"/>
            </div>
          </div>
          <ui:include src="/pieces/Footer.xhtml"/>
          <ui:insert name="activeTab"></ui:insert>
              :
 <ui:composition>
```

Templating with Facelets: Decorators

WelcomeTemp.xhtml

```
<ui:decorate template="templates/masterLayout.xhtml">

        <ui:define name="">
                                        </ui:define>
</ui:decorate>

<ui:decorate template="templates/masterLayout.xhtml">

        <ui:define name="">
        </ui:define>
</ui:decorate>
```

Templating with Facelets: Parameters

- you can supply arguments in two ways
  - ui:define
    - Provide markup that is inserted into the template
  - ui:param
    - sets an EL variable for use in a template
    - The ui:param tag can also be used as a child of a ui:include tag

Home.xhtml

```
<ui:composition template="templates/masterTemplate.xhtml">
        <ui:param name="currentDate" value="#{someBean.currentDate}"/>
</ui:composition>
```

masterLayout.xhtml

```
<body>
    Today's date: #{currentDate}"/>
</body>
```

Custom Tags with Facelets

- In JSF 2.0, you are allow to create your custom tag to render a pre-defined content.
- A custom tag is look like a normal JSF tag, and uses "**ui:composition**" to insert content into the page.

Steps to create a custom Facelet tag:

- Uses :ui:compisit
- Declares the cust
- Tell Facelets to l
- Uses your Custo

```
<ui:composition>
        <h:commandl
        <h:commandl
</ui:composition>

        </h:body>
</facelet-taglib>
```

Custom Tags with Facelets

- Components and Fragments
  - If you change **ui:composition** in the previous example to the **ui:component** tag, then the child elements are placed inside a JSF component.
  - The component is then added to the view
  - You can supply **id**, **binding**, and **rendered** attributes with the **ui:component** tag
  - Similarly, the **ui:fragment** tag is an analog to **ui:decorate** that generates a component

```
<ui:fragment rendered="#{name == planetarium.selectedPlanet}">
        Conditionally included children
</ui:fragment>
```

Loose Ends

<ui:debug>

- When the ui:debug tag is placed in an XHTML page
- It creates a component and adds it to the component tree.
- That debug component captures debugging information
  - the current state of the component tree
  - the scoped variables in the application
- If the user presses CTRL + SHIFT + d, JSF opens a window that shows the debugging information captured by the debug component.
- Typically, the best place to put the ui:debug tag is in an application's main template

```
<ui:debug hotkey="i"/>
```

Loose Ends

<ui:remove>

- Strategy of commenting out parts to isolate the offending component.
- Why not XML Comments ?

```
<!-- <h:commandButton id="loginButton"
 value="#{msgs.loginButtonText}"
 action="planetarium"/> -->
```

JSF will process the value expression #{msgs.loginButtonText}, and place the result, as a comment !!

- Instead, use ui:remove, like this:

```
<ui:remove>
    <h:commandButton id="loginButton" value="#{msgs.loginButtonText}" action="planetarium"/>
</ui:remove>
```

Facelets Tags (cont'd)

- ui:composition
  - composition is a sequence of elements that can be inserted somewhere else.
  - The composition can have variable parts (specified with ui:insert children).
  - When used with a template attribute,
    - the template is loaded.
    - The children of this tag determine the variable parts of the template.
    - The template contents replaces this tag.

Facelets Tags (cont'd)

- ui:decorate
  - specifies a page into which parts can be inserted.
  - The variable parts are specified with ui:insert children.
  - When used with a template attribute,
    - template is loaded.
    - The children of this tag determine the variable parts of the template.

Facelets Tags (cont'd)

- ui:param
  - Specifies a parameter that is passed to an included file or a template.
- ui:component
  - This tag is the same as ui:composition, except that it creates a component that is added to the component tree.
- ui:fragment
  - This tag is like ui:decorate, except that it creates a component that is added to the component tree.

Handling White Space

- By default, whitespace is trimmed around components

```
<h:outputText value="#{msgs.name}"/>

<h:inputText value="#{user.name}"/>
```

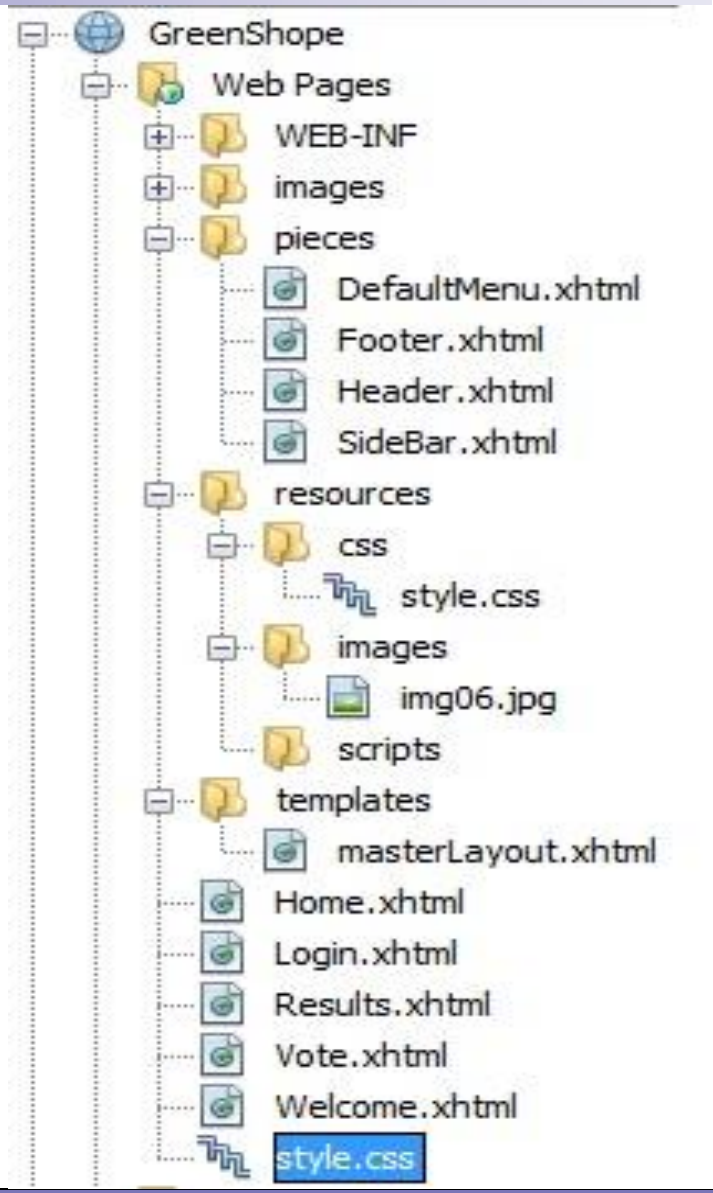
- Facelets won't turn that whitespace into a text component
- tag sequence would not work correctly inside an h:panelGrid
- If you have two links in a row

```
<h:commandLink value="Previous" .../> <h:commandLink value="Next" .../>
```

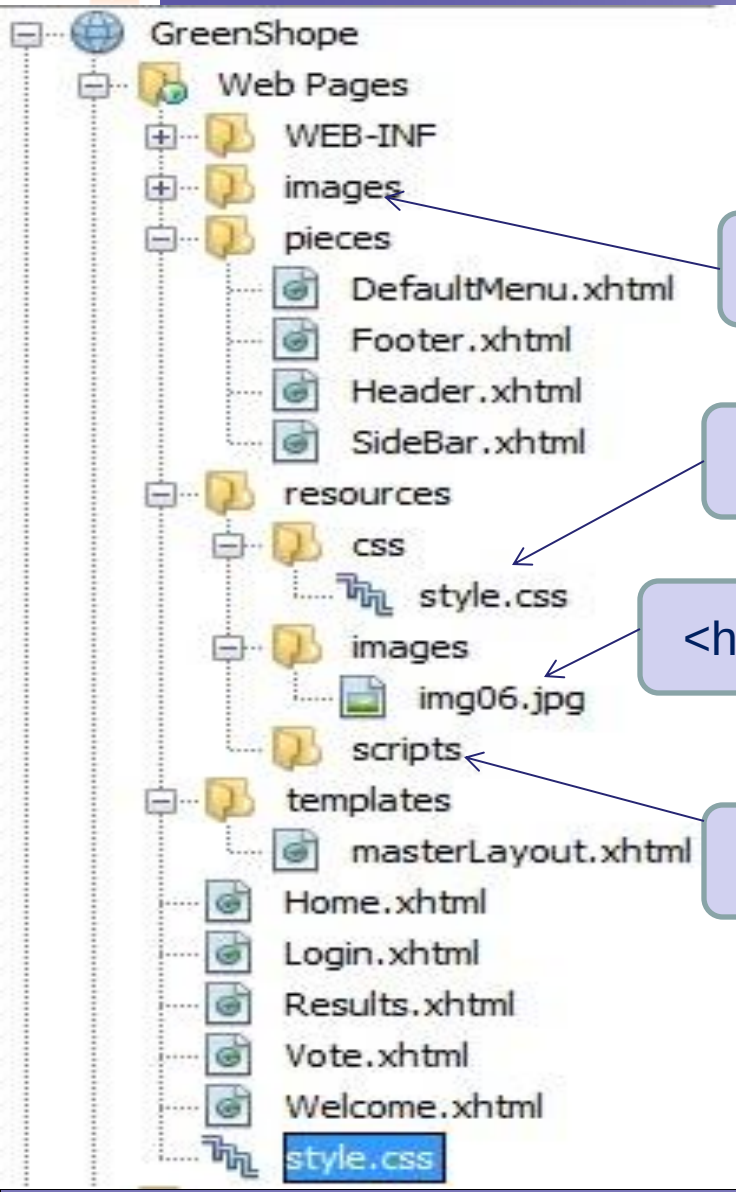
  - yield links PreviousNext
  - Add a space with a value expression #{' '}

Handling Relative URLs

- Issue
  - Templates and pieces used simple relative URLs
- Example
  - Suppose client file was in pieces/DefaultMenu.xhtml, Accessed with http://host/context-root/pieces/DefaultMenu.jsf
    - All relative URLs would refer to Pieces
  - Hypertext links <a href="Footer.jsf"> would now refer to http://host/context-root/pieces/Footer.jsf
  - Images <img src=" /images/img01.jpg"> would now refer http://host/context-root/pieces/images/img01.jpg
  - Style sheets <link href="./css/styles.css"/> would now refer to http://host/context-root/pieces/css/styles.css



Handling Relative URLs



Solution

<h:graphicImage url="/images/img01.jpg"/>

<h:outputStylesheet name="style.css" library="css"/>

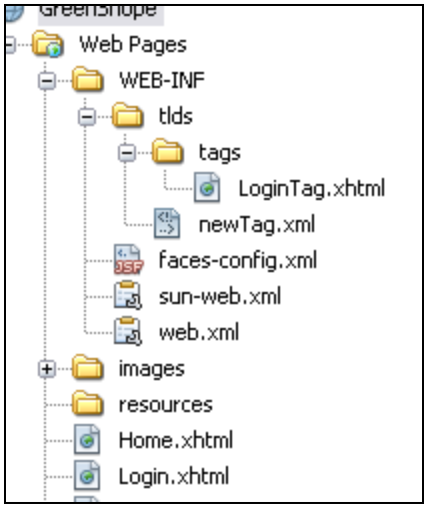
<h:graphicImage name="img06.jpg" library="images"/>

<h:outputScript name="script01.js" library="scripts"/>

Lab Exercise

Assignments

- Create custom component for the login form using facelets.




```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
    <ui:composition>
        <h:form >
            <p>Please enter your Login Data</p>
            <table>
                <tr>
                    <td>Name:</td>
                    <td><h:inputText value="#{userName}" id="name" required="true"/></td>
                    <h:message for="name"/>
                </tr>
                <tr>
                    <td>Password:</td>
                    <td><h:inputSecret value="#{password}"/></td>
                </tr>
                <tr>
                    <td><h:commandButton value="Login" action="#{loginAction.back}" /></td>

                </tr>

            </table>
        </h:form>
    </ui:composition>
</html>
```

```
<facelet-taglib>
   <namespace>http://corejsf.com/facelets</name
   <tag>
      <tag-name>Login</tag-name>
      <source>tags/LoginTag.xhtml</source>
   </tag>
```

```
<context-param>
<param-name>facelets.LIBRARIES</param-name>
<param-value>/WEB-INF/tlds/newTag.xml</param-value>
</context-param>
```

```
<custom:Login userName="#{myBean.userName}" password="#{myBean.password}"
loginAction="#{myBean}"/>
```

Assignments

- Re-organize (Template) your JSF Application Views using JSF Facelet Tags
  - Extract your reusable page snippets like Header, Footer and Menu in stand alone facelets
  - Create your template with shared contents and use it in your views
  - You can insert page snippets any where in template file or directly in your view