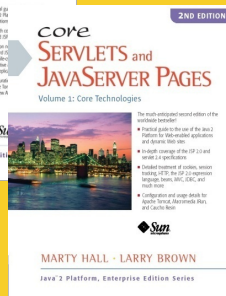
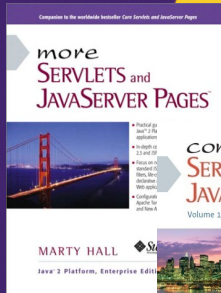




# JSF 2.0: Page Templating with Facelets

Originals of Slides and Source Code for Examples:  
<http://www.coreservlets.com/JSF-Tutorial/jsf2/>

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live training on JSF 2.x, please see courses at <http://courses.coreservlets.com/>.**



**Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.**

- Courses developed and taught by Marty Hall
    - JSF 2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 6 or 7 programming, custom mix of topics
    - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, etc.) or survey several
  - Courses developed and taught by coreservlets.com experts (edited by Marty)
    - Spring, Hibernate/JPA, EJB3, GWT, Hadoop, SOAP-based and RESTful Web Services
- Contact [hall@coreservlets.com](mailto:hall@coreservlets.com) for details**

# Topics in This Section

- **Motivation**
- **Basic mechanism**
  - Template file
  - Client file
- **Templating with includes**
  - Reusable chunks of content that are not part of template, but are used in multiple pages
- **Handling relative URLs**
  - And relocatable resources

5

© 2012 Marty Hall



## Basics

**Customized Java EE Training: <http://courses.coreservlets.com/>**

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

# Need for Page Templating

- **Avoiding repetition in facelets pages**
  - OOP provides good reuse for Java code.
  - But what about facelets code?
    - Also want to avoid repeating nearly-identical code there
    - Very common to have multiple pages that share the same basic layout and same general look



- **Inadequacies of jsp:include**
  - JSF pages normally eschew JSP tags
  - Even if you use jsp:include, it has problems
    - No real templates or named sections (ala Struts Tiles)
    - Can't easily pass data to included pages

7

# Basic Steps

- **Define a template file**
  - Content that will appear in all clients is entered directly
  - Content that can be replaced in client files is marked with ui:insert (with default values in case client does not supply content)
- **Define a client file that uses the template**
  - Use ui:composition to specify the template file used
  - Use ui:define to override content for each replaceable section in template (marked in template with ui:insert)
- **Access the client file in a browser**
  - <http://host/app/clientfile.jsf>
    - Users never access template files directly

8

# Simplified Example

- **/templates/template-1.xhtml**

```
...
<h:body>Content shared by all client files
<h2><ui:insert name="title">Default Title</ui:insert></h2>
More content shared by all clients
<ui:insert name="body">Default Body</ui:insert>
</h:body> ...
```

- **client-file-1.xhtml**

```
<ui:composition template="/templates/template-1.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>
```

Replaces the default content of the "title" section from the template. If this is omitted in the client, then default content is used instead.

9

# Template File: Details

- **Namespaces**

- Minimum: schemas for xhtml, facelets h: and ui:

```
<!DOCTYPE ... (standard xhtml) >
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
```

```
...
</html>
```

- If you use other tag libraries (f, c, etc.), list those also

```
<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">
```

```
...
</html>
```

10

# Template File: Details

- **Insert shared content literally**
  - If you have HTML or JSF data that will be used by all clients, place it directly in template file
  - You can use any normal JSF elements
    - Regular HTML, expression language, h:dataTable, ui:repeat, composite componets, etc.
- **Mark replaceable sections with ui:insert**
  - Give a name to the section (client will refer to name)
  - Put default content between `<ui:insert...>` and `</ui:insert>`
    - If clients don't supply anything for name, they get default
- **Put template files in separate folder**
  - So they won't be confused with user-accessible files.
    - Some developers put them in folder under WEB-INF

11

# Template File: Example (/templates/sample-template.xhtml)

```
<!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.

12

# Bean Code

```
package coreservlets;

import javax.faces.bean.*;

@ManagedBean
public class NumGenerator {
    public double getRandomNum() {
        return (Math.random());
    }
}
```

13

# Client File: Details

- **Use ui:composition with schema references**
  - `<ui:composition xmlns="..." xmlns:ui="..." ...>`
    - Text outside of ui:composition ignored for output
- **Use “template” attribute to refer to template file**
  - `<ui:composition xmlns="..." xmlns:ui="..."  
template="/templates/some-template.xhtml">`
- **Use ui:define to supply content for sections**
  - `<ui:define name="section-name-from-template-file">`  
Content to replace default for that section in template  
XHTML tags, JSF tags, EL expressions, components, etc.  
`</ui:define>`
- **File goes in normal location for JSF file**
  - Needs to be accessible to user
    - If file is blah.xhtml, it will be accessed as blah.jsf (or whatever url-pattern was in web.xml – see intro sections of JSF tutorial)

14

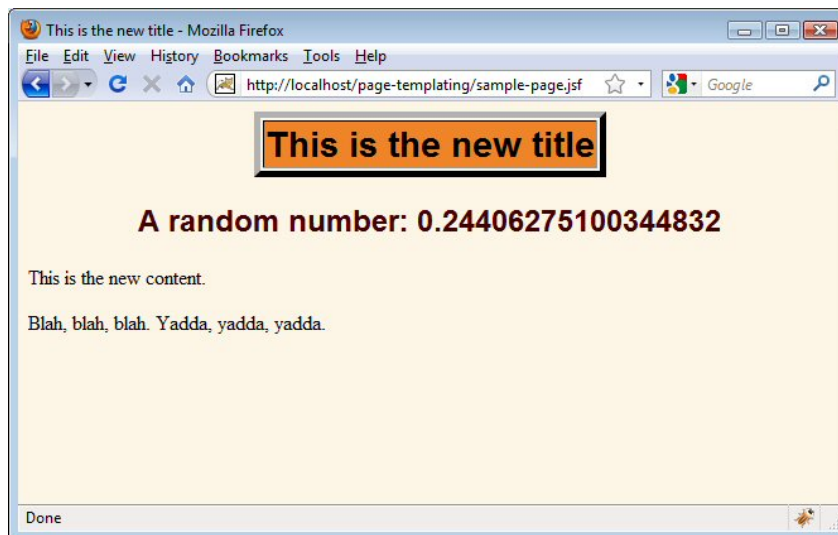
# Client File: Example (/sample-page.xhtml)

```
<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.

15

# Client File: Result



16



# Including Files in Templates

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Motivation

- **Content that is shared by all clients**
  - Put directly in template file
- **Content that is specific to individual client**
  - Put in client file as body of ui:define (replacing a section from the template file)
- **Content that is shared by some clients?**
  - And perhaps inserted in different places in each client
  - Problems
    - Not used by all clients, so can't go directly in template file
    - Used by more than one client, so would be repetitive to specify the content in the client file with ui:define
  - Solution
    - Put reusable content in separate file
    - In client file, use ui:include in the body of ui:define

## Using ui:include

- **In last example, content that was always shared went in template file**
  - What if content is only *sometimes* shared?
- **Options**
  - Content that is common to all clients
    - Goes in template file
  - Content that is specific to a particular client
    - Goes in client file in the body of ui:define
  - Content that is shared by multiple clients, but might not be shared by all clients
    - Put in separate file
    - Load it in client file by using ui:include in the body of ui:define

19

## Template File

- **Same rules as before**
  - Insert shared content literally
  - Mark replaceable sections with ui:insert
  - Output dynamic values with #{blah}
  - Put template files in separate folder

20

## File to be Included (Snippet File)

- **Enclose content in ui:composition**
  - With appropriate schema references, as before
- **Place snippets in separate folder**
  - Avoid confusion with user-accessible files or templates
- **Follow XML syntax**
  - This is different from files that are included with `jsp:include`, where snippet file is just a bunch of orphaned tags. Here, the entire file is legal XML.
    - Content outside `ui:composition` (if any) is ignored
- **Snippets can use templates**
  - But then top-level tag in template file will be `<ui:composition>`, not `<html>`
    - Since those templates build snippets, not full HTML files

21

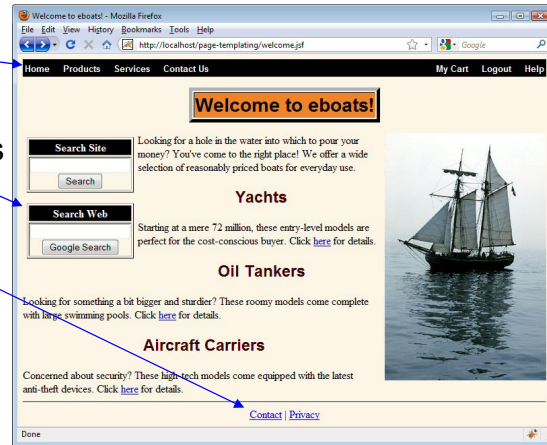
## Client File

- **Use ui:composition as before**
  - With schema references and “template” attribute
  - `<ui:composition xmlns="..." xmlns:ui="..."  
template="/templates/some-template.xhtml">`
- **Use ui:define with ui:include to refer to snippets**
  - `<ui:define name="section-name1-from-template-file">  
    <ui:include src="/snippets/some-snippet.xhtml"/>  
</ui:define>`
  - `<ui:define name="section-name2-from-template-file">  
    Client-specific content  
</ui:define>`
- **Client file goes in normal location for JSF file**
  - Needs to be accessible to user
    - If file is `blah.xhtml`, it will be accessed as `blah.jsf` (or whatever url-pattern was in `web.xml` – see intro sections of JSF tutorial)

22

# Example: An Online Boat Store

- **Common to all pages**
  - DOCTYPE; head, title, and body tags; style sheet; border around heading
    - Content goes directly in template
- **Common to some pages (potentially changeable)**
  - Header
  - Search box
  - Footer
    - Content goes in separate files that are loaded via `ui:include` within `ui:define`
- **Unique to pages**
  - Title text
  - Body
    - Content goes directly inside `ui:define` in client files



23

## Template File (Top) (/templates/eboats-template.xhtml)

```
<!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">Title</ui:insert></title>
    <link rel="stylesheet" type="text/css"
          href="./css/styles.css"/>
  </h:head>
  <h:body>
    <ui:insert name="header">Header</ui:insert>
    <p/>
```

24

## Template File (Continued) (/templates/eboats-template.xhtml)

```
<table border="5" align="center">
  <tr><th class="title">
    <ui:insert name="title">Title</ui:insert>
  </th></tr>
</table>
<p/>
<table width="75" align="left" cellspacing="5">
<tr><td><ui:insert name="menu">Menu</ui:insert></td></tr>
</table>
<p/>
<ui:insert name="body">Body</ui:insert>
<br clear="all"/>
<hr/>
<ui:insert name="footer">Footer</ui:insert>
</h:body></html>
```

25

## Header File (/snippets/header.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets">
<table width="100%" class="dark">
<tr><th align="left">
  <a href="welcome.jsf" class="white">Home</a>
  &#160;&#160;&#160;
  <a href="products.jsf" class="white">Products</a>
  &#160;&#160;&#160;
  <a href="services.jsf" class="white">Services</a>
  &#160;&#160;&#160;
  <a href="contact.jsf" class="white">Contact Us</a>
</th><th align="right">
  <a href="cart.jsf" class="white">My Cart</a>
  &#160;&#160;&#160;
  <a href="logout.jsf" class="white">Logout</a>
  &#160;&#160;&#160;
  <a href="help.jsf" class="white">Help</a>
</th></tr></table>
</ui:composition>
```

As of Mojarra 2.0.2, you cannot use named HTML character entities (e.g., &nbsp; for a non-breaking space) in included pages. Instead, you have to use numeric character entities (e.g., &#160; for a non-breaking space). This is scheduled to be fixed in later release.

26

# Footer File 1

## (/snippets/footer-full.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets">
<div align="center">
  <a href="welcome.jsf">Home</a> |
  <a href="contact.jsf">Contact</a> |
  <a href="privacy.jsf">Privacy</a>
</div>
</ui:composition>
```

This footer is included in most, but not all, pages that use eboats-template.xhtml. The welcome page uses an abbreviated footer that does not include welcome.jsf (since the user is *already* on welcome.jsf).

A later part of this tutorial will discuss the problems that would happen if this snippet were to be included in pages that were in subdirectories instead of at the top level of the Web application: the relative URLs would no longer work. The style sheet and image in the top-level eboats-template.xhtml file have similar problems. In a few slides from now, we'll improve this example to include fixes for those problems.

27

# Footer File 2

## (/snippets/footer-no-home.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets">
<div align="center">
  <a href="contact.jsf">Contact</a> |
  <a href="privacy.jsf">Privacy</a>
</div>
</ui:composition>
```

This is the same as the previous slide except for the omission of the link to the home page.

28

# The Search Box Snippets

- **Goal**
  - Have various versions of the Web search box for different search engines (Google, Bing, Yahoo, etc.)
  - Avoid repeating the local-search box and most of the formatting
- **Solution**
  - Make a general search menu template
  - Use that template for each of the search-engine specific entries
- **Difference from previous template examples**
  - The search menu is not a full xhtml file, so it should not contain DOCTYPE, <html>, <head>, <body>, etc.
    - So, just use a snippet, and enclose it in ui:composition

29

## Search Box Template (Top) (/templates/search-box-template.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets">
<div align="center" >
<table border="1"><tr bgcolor="black"><th>
  <font color="white">Search Site</font>
</th></tr><tr><th>
  <form action="siteSearch">
    <input type="text" name="query"/><br/>
    <input type="submit" value="Search"/>
  </form>
</th></tr></table>
```

Although this is a template file, the client file that uses it will be a snippet file (which in turn will be included in another page), not a top-level page. So, the main tag is <ui:composition...>, not <html...>

30

## Search Box Template (Bottom) (/templates/search-box-template.xhtml)

```
<p/>
<table border="1"><tr bgcolor="black"><th>
  <font color="white">Search Web</font>
</th></tr><tr><th>
  <ui:insert name="search-form">
    <h1>Missing search form</h1>
  </ui:insert>
</th></tr></table></div>
</ui:composition>
```

31

## Google Search Box (/snippets/google-search-box.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="/templates/search-box-template.xhtml">
  <ui:define name="search-form">
    <form action="http://google.com/search">
      <input type="text" name="q"/><br/>
      <input type="submit" value="Google Search"/>
    </form>
  </ui:define>
</ui:composition>
```

This file is both a snippet (which will be included in another page) and a client file (which uses a template). This idea of composing templates out of other templates is very powerful. An upcoming section discusses composite components, which are even more powerful.

32

# Bing Search Box

(/snippets/bing-search-box.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="/templates/search-box-template.xhtml">
  <ui:define name="search-form">
    <form action="http://bing.com/search">
      <input type="text" name="q"/><br/>
      <input type="submit" value="Bing Search"/>
    </form>
  </ui:define>
</ui:composition>
```

This is the same as the Google search box except for the URL of the action and the title of the button.

33

# Yahoo Search Box

(/snippets/bing-search-box.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="/templates/search-box-template.xhtml">
  <ui:define name="search-form">
    <form action="http://search.yahoo.com/search">
      <input type="text" name="p"/><br/>
      <input type="submit" value="Yahoo Search"/>
    </form>
  </ui:define>
</ui:composition>
```

This is the same as the previous search box except for the URL of the action, the name of the input element, and the title of the button.

34

## Putting it Together: Client File 1 (/welcome.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="/templates/eboats-template.xhtml">

  <ui:define name="title">
    Welcome to eboats!
  </ui:define>

  <ui:define name="header">
    <ui:include src="/snippets/header.xhtml"/>
  </ui:define>

  <ui:define name="menu">
    <ui:include src="/snippets/google-search-box.xhtml"/>
  </ui:define>
```

35

## Client File 1 Continued (/welcome.xhtml)

```
<ui:define name="body">
  <p/>
  
  Looking for a hole in the water into which to pour your money?
  You've come to the right place! We offer a wide selection of
  reasonably priced boats for everyday use.
  <h2>Yachts</h2>
  ... (more body content)
</ui:define>

<ui:define name="footer">
  <ui:include src="/snippets/footer-no-home.xhtml"/>
</ui:define>
</ui:composition>
```

36

# Client File 1: Result



37

# Client File 2 (/tankers.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    template="/templates/eboats-template.xhtml">

    <ui:define name="title">
        Eboats Oil Tankers!
    </ui:define>

    <ui:define name="header">
        <ui:include src="/snippets/header.xhtml"/>
    </ui:define>

    <ui:define name="menu">
        <ui:include src="/snippets/yahoo-search-box.xhtml"/>
    </ui:define>
```

38

# Client File 2 Continued (/tankers.xhtml)

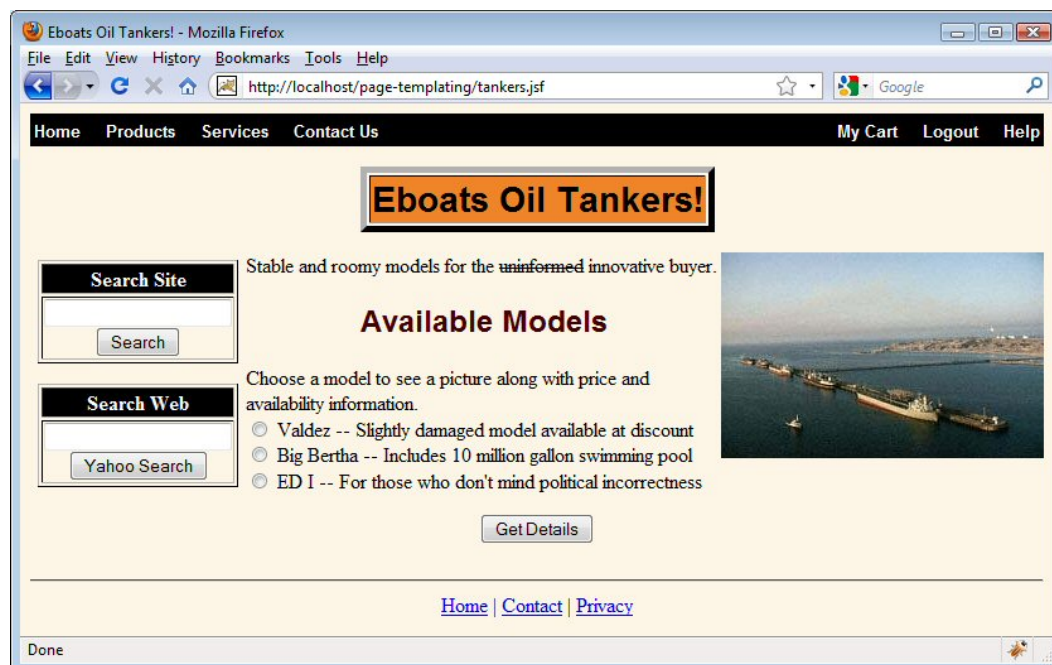
```
<ui:define name="body">

<p/>
Luxurious models for the <s>wasteful</s> wealthy buyer.
<h2>Available Models</h2>
Choose a model to see a picture along with price and
availability information.
... (more body content)
</ui:define>

<ui:define name="footer">
  <ui:include src="/snippets/footer-full.xhtml"/>
</ui:define>
</ui:composition>
```

39

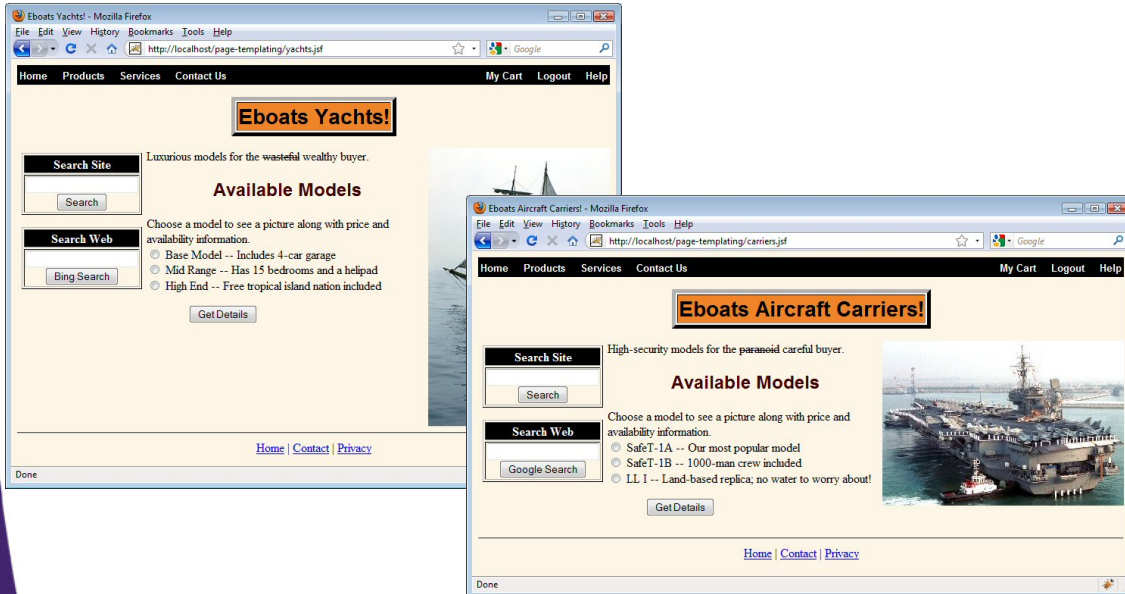
# Client File 2: Result



40

# Client Files 3 and 4

- **Almost-identical approach**
  - Full code online



41

© 2012 Marty Hall



# Handling Relative URLs in Templates

**Customized Java EE Training:** <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

# Problems with Previous Example

- **Issue**
  - Templates and snippets used simple relative URLs
- **Example**
  - Suppose client file was in `subdir/client.xhtml`, accessed with `http://host/context-root/subdir/client.jsf`
    - All relative URLs would refer to `subdir`
  - Hypertext links
    - `<a href="welcome.jsf">` would now refer to `http://host/context-root/subdir/welcome.jsf`
  - Images
    - `` would now refer to `http://host/context-root/subdir/images/yacht.jpg`
  - Style sheets
    - `<link ...href="./css/styles.css"/>` would now refer to `http://host/context-root/subdir/css/styles.css`
  - JavaScript files
    - Not used in this example, but would have similar problems

43

# Solutions

- **Hypertext links**
  - Use `<a href="#{request.contextPath}/blah.jsf">`
    - Or `<h:outputLink value="#{request.contextPath}/blah.jsf"/>`
- **Images**
  - Put in images folder under “resources” and use `<h:graphicImage name="blah.jpg" library="images"/>`
    - Can also use `<h:graphicImage url="/images/blah.jpg"/>`
- **Style sheets**
  - Put in css folder under “resources” and use `<h:outputStylesheet name="blah.css" library="css"/>`
- **JavaScript files**
  - Put in scripts folder under “resources” and use `<h:outputScript name="blah.js" library="scripts"/>`

44

## Reminder – h:head and h:body

- **Plan ahead: always use h:head and h:body**
  - Never just <head> and <body> (so JSF can find regions)
- **Reasons**
  - f:ajax
    - This tag automatically inserts scripts, and can't find the place to insert unless you use h:head and h:body
  - h:outputStylesheet
    - <h:outputStylesheet> does not have to be in the head, so JSF needs to be able to find the head to insert the final output (<link>) that loads the style sheet.
  - h:outputScript
    - Again, <h:outputScript> does not have to be in the head, so JSF needs to be able to find the head (default) or body (if you use target="body") to insert the final output (<script>) that loads the JavaScript file

45

## Hypertext Links: Relative URLs

- **Prepend #{request.contextPath}**
  - Unlike images, style sheets, and scripts, JSF has no builtin way to build context-relative links.
  - However, it is easy to do so yourself
- **Examples**
  - <a href="#"#{request.contextPath}/blah.jsf">
  - <h:outputLink value="#"#{request.contextPath}/blah.jsf"/>
    - Assume that your context root is /my-context. Then, both of the above build <a href="/my-context/blah.jsf">

46

## Images: Relative URLs

- **Use h:graphicImage with name and library**
  - JSF 2.0 added the concept of resource folders. You make a folder literally called “resources”, then make subfolders inside (you can choose the name of the folders inside).
    - For “library”, give the name of the images subfolder
    - For “name”, give the image filename
  - This has two advantages beyond just context-relative URLs: versioning, localization, and resources in the classpath
  - Or, use `<h:graphicImage url="/foo/bar.jpg"/>`
    - Outputs ``. This is a holdover from JSF 1.x. Resources are better in most cases.
- **Example**
  - `<h:graphicImage name="blah.gif" library="images"/>`
    - Outputs ``, where src refers to JSF page that outputs `/my-context/resources/images/blah.gif`

47

## Style Sheets: Relative URLs

- **Use h:outputStylesheet with name & library**
  - Again, make “resources” folder with subfolder inside
    - For “library”, give the name of the CSS subfolder
    - For “name”, give the filename of the stylesheet
  - It is not necessary to put `<h:outputStylesheet>` in the head
    - This makes it easy for components or included pages to refer to the style sheets they need. The real reference will come out in the head, and if there are multiple uses of `h:outputStylesheet` in the same page (e.g., from two included pages), it will build only one ref in main page
- **Example**
  - `<h:outputStylesheet name="blah.css" library="css"/>`
    - Outputs `<link type="text/css" rel="stylesheet" href="..." />` where href refers to JSF page that outputs `/my-context/resources/css/blah.css`

48

# Scripts: Relative URLs

- **Use h:outputScript with name & library**
  - Again, make “resources” folder with subfolder inside
    - For “library”, give the name of the JavaScript subfolder
    - For “name”, give the filename of the JavaScript file
  - It is not necessary to put `<h:outputScript>` in the head
    - As with style sheets, this makes it easy for components or included pages to refer to the scripts they need. Again, if the tag is used multiple times, only one ref to the script appears in the real page.
    - You can also say `target="body"` if you want script reference to appear in body instead of in head.
- **Example**
  - `<h:outputScript name="blah.js" library="scripts"/>`
    - Outputs `<script type="text/javascript" src="..." />` where `src` refers to JSF page that outputs `/my-context/resources/scripts/blah.js`

49

# Example: Referring to Files from Templates or Includes



50

# Example: yacht-template.xhtml (Top)

```
<!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>
    <h:outputStylesheet name="styles.css" library="css"/>
    <h:outputScript name="jquery.js" library="scripts"/>
    <h:outputScript name="jquery-setup.js" library="scripts"/>
  </h:head>
  <h:body>
    <div align="center">
      <table border="5" align="center">
        <tr><th class="title">
          <ui:insert name="title">Title</ui:insert>
        </th></tr>
      </table>
      <h2>Some of our Yachts</h2>
```

The pushbutton on the next slide uses jQuery to highlight bullets. jQuery is explained in a separate tutorial at [coreservlets.com](http://coreservlets.com), but the point is that this page needs a context-path-relative way to load the JavaScript files and the stylesheet.

51

# Example: yacht-template.xhtml (Continued)

```
<table>
<tr><th>Small Yacht</th><th>Medium Yacht</th></tr>
<tr><td>
  <h:graphicImage name="small-yacht.jpg" library="images"/>
</td><td>
  <h:graphicImage url="/images/medium-yacht.jpg"/>
</td></tr></table>

<h2>Benefits</h2>
<div id="benefits">
  <ui:insert name="benefits">
    List of benefits of our yachts goes here.
    <ul><li>Benefit 1</li><li>...</li></ul>
  </ui:insert>
</div>
<input type="button" id="highlight-button"
       value="Highlight Benefits"/>
```

The top form is better for new apps, but the bottom form is supported for backward compatibility with facelets in JSF 1.x.

52

## Example: yacht-template.xhtml (Continued)

```
<div align="center">
  <a href="#{request.contextPath}/welcome.jsf">Home</a> |
  <a href="#{request.contextPath}/contact.jsf">Contact</a> |
  <a href="#{request.contextPath}/privacy.jsf">Privacy</a>
</div>
</div></h:body></html>
```

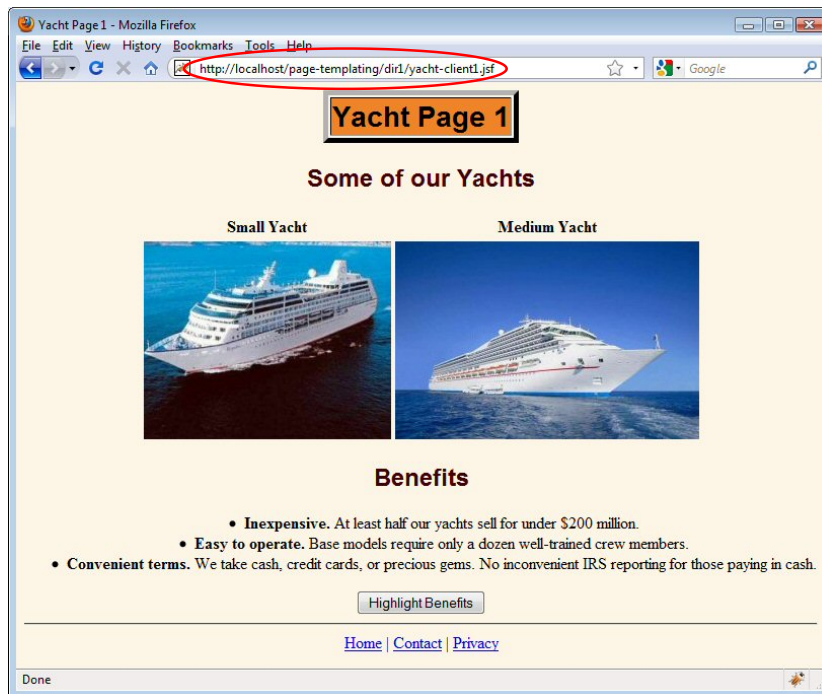
53

## Example: yacht-client1.xhtml (in "dir1" subfolder)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="/templates/yacht-template.xhtml">
  <ui:define name="title">Yacht Page 1</ui:define>
  <ui:define name="benefits">
    <ul>
      <li><b>Inexpensive.</b> At least half our yachts
        sell for under $200 million.</li>
      <li><b>Easy to operate.</b> Base models
        require only a dozen well-trained
        crew members.</li>
      <li><b>Convenient terms.</b> We take cash,
        credit cards, or precious gems. No
        inconvenient IRS reporting for those
        paying in cash.</li>
    </ul>
  </ui:define>
</ui:composition>
```

54

# Result: yacht-client1.jsf



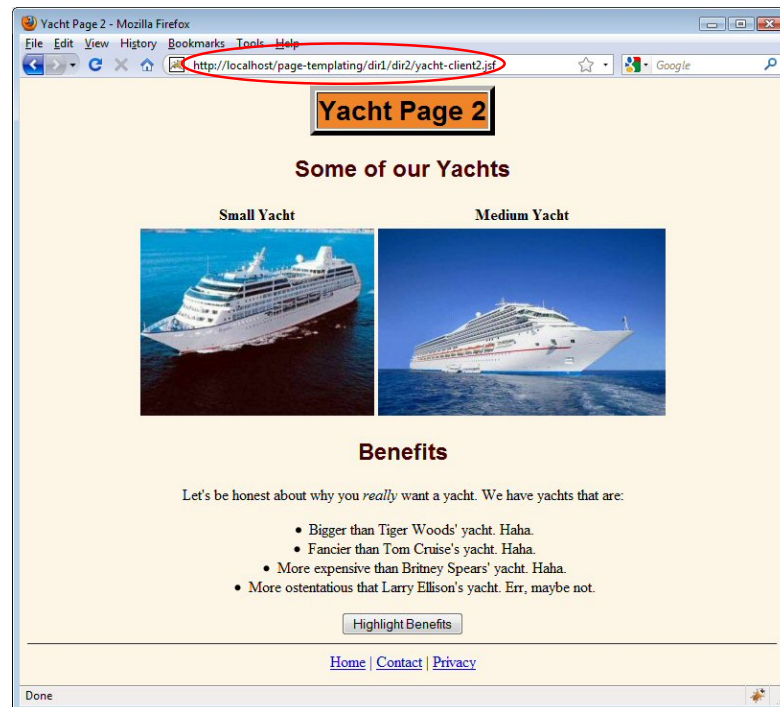
55

# Example: yacht-client2.xhtml (in "dir1/dir2" subfolder)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="/templates/yacht-template.xhtml">
  <ui:define name="title">Yacht Page 2</ui:define>
  <ui:define name="benefits">
    <p>Let's be honest about why you <i>really</i>
      want a yacht. We have yachts that are:</p>
    <ul>
      <li>Bigger than Tiger Woods' yacht. Haha.</li>
      <li>Fancier than Tom Cruise's yacht. Haha.</li>
      <li>More expensive than Britney Spears' yacht. Haha.</li>
      <li>More ostentatious that Larry Ellison's yacht.
        Err, maybe not.</li>
    </ul>
  </ui:define>
</ui:composition>
```

56

# Result: yacht-client2.jsf



57

© 2012 Marty Hall



## Wrap-Up

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

# Summary

- **Template file**
  - Directly insert content that will be used by all clients
  - Mark replaceable regions with `ui:insert`
- **Client file**
  - Use `ui:composition` to designate template that is used
  - Override replaceable regions with `ui:define`
  - Insert snippets with `ui:include`
- **Relative URLs in templates**
  - Hypertext links: `<a href="#{request.contextPath}/..." />`
  - Images: `<h:graphicImage name="..." library="..." />`
  - CSS: `<h:outputStylesheet name="..." library="..." />`
  - Scripts: `<h:outputScript name="..." library="..." />`

59

© 2012 Marty Hall



## Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.