



JSF 2.0: Building Composite Components

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 for details**

Topics in This Section

- **Idea**
- **Basic components**
- **Passing values to components**
- **Using ui:repeat inside components**
- **Handling relative URLs in components**
- **Nested components**

5

© 2012 Marty Hall



Overview

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.

Idea

- **JSF 2.0 added “composite components”**
 - Reusable, named chunks of facelets code
 - Can pass arguments
- **Similar to JSP tag files**
 - But have use of JSF tags and JSF expression language
 - Real components: can have listeners, events, facets, etc.
- **Abbreviated example**
 - resources/utils/myComponent.xhtml (snippet)

```
<composite:implementation>
  This is a test
</composite:implementation>
```
 - some-page.xhtml (main page)

```
<utils:myComponent/>
```

7

Making and Using Custom Components: Overview

- **Define a component**
 - Put foo.xhtml in “resources/bar” folder
 - Define available attributes inside composite:interface
 - Build output inside composite:implementation
 - Use composite: namespace in component file
 - <html ...
xmlns:composite="http://java.sun.com/jsf/composite">
- **Use component namespace in facelets page**
 - <html ...
xmlns:bar="http://java.sun.com/jsf/composite/bar">
- **Use the component in facelets page**
 - <bar:foo ... />

8



Basic Components

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.

Super-Quick Example

- **Component File**

resources/**utils**/**myComponent**.xhtml

```
<!DOCTYPE ...>
<html ...
  xmlns:composite=
    "http://java.sun.com/jsf/composite">
<head><title>...</title></head>
<body>

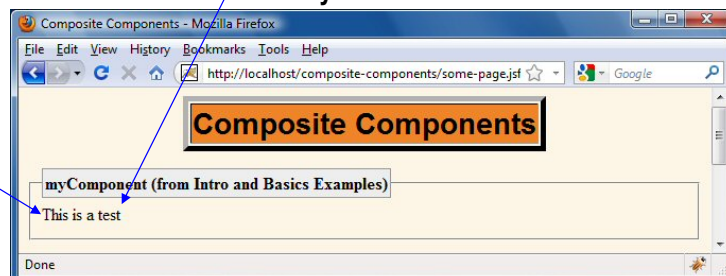
<composite:interface/>

<composite:implementation>
  This is a test
</composite:implementation>
</body></html>
```

- **Main File**

some-page.xhtml

```
<!DOCTYPE ...>
<html ...
  xmlns:utils=
    "http://java.sun.com/jsf/composite/utils">
<h:head><title>Composite Components</title>
...
<h:body>...
<utils:myComponent/>...
</h:body></html>
```



Component File: Summary

- **In subfolder of “resources” folder**
 - E.g., resources/someDir/someFile.xhtml
 - We already used “resources” in section on relocatable resources in tutorial on page templating
- **Basic structure**
 - Looks like xhtml file
 - head and body tags are simply to satisfy editor
 - Declares composite: namespace in <html...> start tag
 - <html ... xmlns:composite="http://java.sun.com/jsf/composite">
 - Contains composite:interface
 - Defines attributes used (if any)
 - Contains composite:implementation
 - Creates final output
 - Only content of composite:implementation becomes output

11

Component File: Outline

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite">
<head><title>(For validation only)</title></head>
<body>

<composite:interface>
  <!-- Declare attributes here -->
</composite:interface>

<composite:implementation>
  <!-- Create output here -->
</composite:implementation>

</body></html>
```

12

Main File: Summary

- **Regular facelets file (someFile.xhtml)**
 - No special folder name.
 - Accessed with someFile.jsf (or someFile.faces, etc.) in normal manner.
- **Basic structure**
 - Normal facelets structure
 - h:head, h:body, and other normal JSF constructs
 - Declares component namespace in <html...> start tag
 - xmlns:someDir="http://java.sun.com/jsf/composite/someDir"
 - Technically, there is no requirement that the namespace (the part after xmlns:) match the folder name. But, it is traditional to do so.
 - Use component
 - <someDir:someFile/> (if no attributes)
 - <someDir:someFile att1="..." ... /> (if it uses attributes)

13

Main File: Outline

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:someDir="http://java.sun.com/jsf/composite/someDir">
<h:head>
...
</h:head>
<h:body>
...
<someDir:someFile/>
...
</h:body></html>
```

14

Example

- **Make a bulleted list of request info**
 - Requesting host
 - User agent
 - Current Locale
 - Session ID
- **May want to reuse list, so make component**
 - resources/**utils**/info1.xhtml
 - Generates list inside composite:implementation
 - composite-components.xhtml
 - Declares **utils**: namespace
 - Uses **<utils:info1/>**

15

Example: Summary

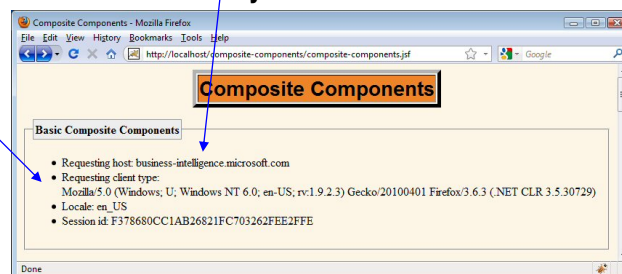
• Component File

```
resources/utils/info1.xhtml
<!DOCTYPE ...>
<html ... xmlns:composite=
    "http://java.sun.com/jsf/composite">
<head><title>...</title></head>
<body>
<composite:interface/>
<composite:implementation>
<ul>
<li>Requesting host:
    #{request.remoteHost}</li>
<li>Requesting client type:<br>
    #{header["User-Agent"]}</li>
<li>Locale: #{view.locale}</li>
<li>Session id: #{session.id}</li>
</ul>
</composite:implementation>
</body></html>
```

• Main File

some-page.xhtml

```
<!DOCTYPE ...>
<html ...
    xmlns:utils=
        "http://java.sun.com/jsf/composite/utils">
<h:head><title>Composite Components</title>
...
<h:body>...
<utils:info1/>...
</h:body></html>
```



16

Component File (resources/utills/info1.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite">
<head><title>(For validation only)</title></head>
<body>

<composite:interface/> <!-- No attributes -->

<composite:implementation>
  <ul>
    <li>Requesting host: #{request.remoteHost}</li>
    <li>Requesting client type:<br/>
      #{header["User-Agent"]}</li>
    <li>Locale: #{view.locale}</li>
    <li>Session id: #{session.id}</li>
  </ul>
</composite:implementation>

</body></html>
```

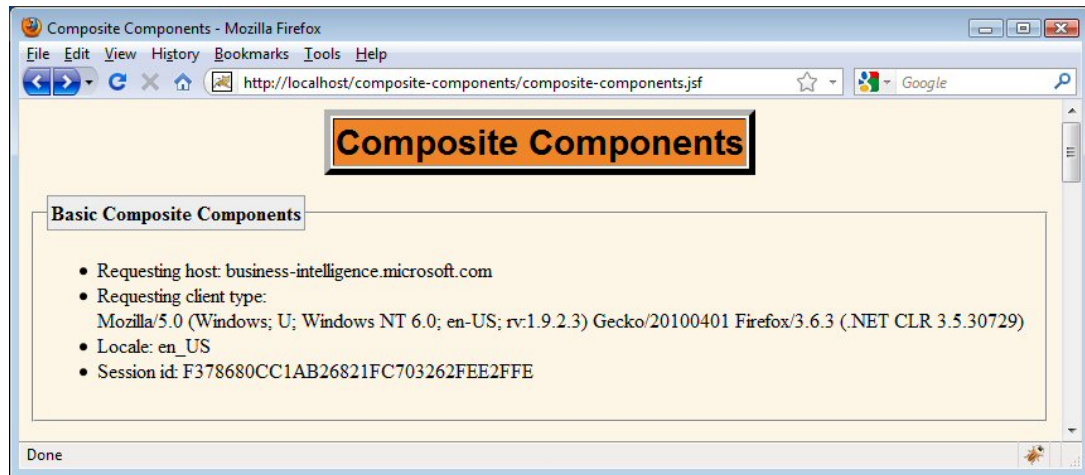
17

Main File (composite-components.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:utills="http://java.sun.com/jsf/composite/utills">
<h:head>
...
</h:head>
<h:body>
...
<fieldset>
<legend>Basic Composite Components</legend>
<utills:info1/>
</fieldset>
...
</h:body></html>
```

18

Results



19

© 2012 Marty Hall



Components with Attributes

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.

Component File: Summary

- **In subfolder of “resources” folder**
 - E.g., resources/someDir/someFile.xhtml
- **Basic structure**
 - Looks like xhtml file as before
 - Declares composite: namespace as before
 - Contains composite:interface
 - Defines attributes used

```
<composite:interface>
  <composite:attribute name="attributeName"/>
</composite:interface>
```
 - Contains composite:implementation
 - Creates final output. **Refers to attributes with cc.attrs**

```
<composite:implementation>
...#{cc.attrs.attributeName}...
</composite:implementation>
```

21

Main File: Summary

- **Declares namespace as in previous example**
 - In <html...> start tag as before

```
xmlns:someDir="http://java.sun.com/jsf/composite/someDir"
```
- **Uses component with the attribute**
 - <someDir:someFile attributeName="..."/>

22

More on cc.attrs

- **“cc” is predefined variable**
 - Stands for “composite component”
 - Refers to top-level component in component file, of type `UINamingContainer`
- **Main property is “attrs”**
 - This is a specially defined attribute that contains a Map of the attributes used to call the component
 - Main page: `<someDir:someFile message="test"/>`
 - Component: `<h2>Message: #{cc.attrs.message}</h2>`
- **Other useful attributes**
 - parent, children, clientId

23

Component File: Outline

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite">
<head><title>(For validation only)</title></head>
<body>

<composite:interface>
  <composite:attribute name="attributeName"/>
</composite:interface>

<composite:implementation>
  Blah #{cc.attrs.attributeName} Blah
</composite:implementation>

</body></html>
```

24

More on composite:attribute

- **Basic usage**

```
<composite:interface>  
  <composite:attribute name="attributeName"/>  
</composite:interface>
```

- **Basic attributes**

- name
 - Attribute name as used in component in main page
- required
 - Is attribute required? (default: false)
 - **Warning! Mojarra 2.0.2 does not enforce this! Fixed in 2.0.3.**
- default
 - For non-required attributes, value to use if none supplied

- **Advanced attributes**

- type, method-signature, targets
 - For attributes that expect methods or complex values

25

Mini Example

- **composite:interface**

```
<composite:attribute name="msg1" required="true"/>  
<composite:attribute name="msg2" default="Hi"/>
```

Reminder: required is not properly enforced in Mojarra 2.0.2 and earlier.

- **composite:implementation**

```
<h2>Message 1: "#{cc.attrs.msg1}"</h2>  
<h2>Message 2: "#{cc.attrs.msg2}"</h2>
```

- **Main page**

```
<utils:showMessages msg1="Hello, world"/>
```

- **Result**

```
Message 1: "Hello, world"  
Message 2: "Hi"
```

26

Example

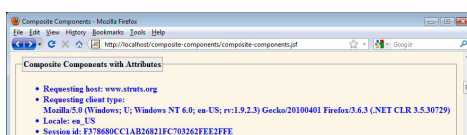
- **Make a bulleted list of request info as before**
 - Requesting host, user agent, current Locale, session ID
- **Support styleClass attribute**
 - To specify CSS style of resultant list
- **Component**
 - resources/Utils/info2.xhtml
 - Declares styleClass inside composite:interface
 - Generates list inside composite:implementation
 - composite-components.xhtml
 - Declares utils: namespace
 - Uses <utils:info2 styleClass="some-css-name"/>

27

Example: Summary

- **composite:interface**
<composite:attribute name="styleClass"/>
- **composite:implementation**
<ul class="#{cc.attrs.styleClass}">
 Requesting host: #{request.remoteHost}
 Requesting client type
 #{header["User-Agent"]}
 Locale: #{view.locale}
 Session id: #{session.id}

- **Main page**
<utils:info2 styleClass="blue"/>
- **Result**



28

Component File (resources/utills/info2.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite">
...
<composite:interface>
  <composite:attribute name="styleClass"/>
</composite:interface>

<composite:implementation>
  <ul class="#{cc.attrs.styleClass}">
    <li>Requesting host: #{request.remoteHost}</li>
    <li>Requesting client type:
      <br/> #{header["User-Agent"]}</li>
    <li>Locale: #{view.locale}</li>
    <li>Session id: #{session.id}</li>
  </ul>
</composite:implementation>
</body></html>
```

29

Main File (composite-components.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:utills="http://java.sun.com/jsf/composite/utills">
<h:head>
...
</h:head>
<h:body>
...
<fieldset>
<legend>Composite Components with Attributes</legend>
<utills:info2 styleClass="blue"/>
</fieldset>
...
</h:body></html>
```

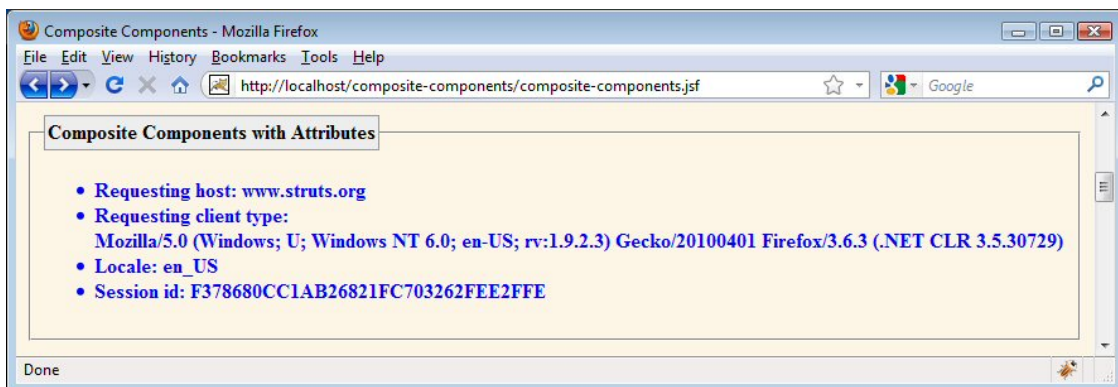
30

CSS File

```
.italic { font-style: italic; }  
.blue { font-weight: bold; color: blue; }  
...
```

31

Results



32



Using ui:repeat with Composite Components

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.

Component File: Summary

- **In subfolder of “resources” folder**
 - E.g., resources/someDir/someFile.xhtml
- **Basic structure**
 - Looks like xhtml file as before
 - Declare composite: *and* ui: namespaces
 - `<html ... xmlns:composite="http://java.sun.com/jsf/composite" xmlns:ui="http://java.sun.com/jsf/facelets">`
 - Contains composite:interface
 - Defines attributes as before
 - Contains composite:implementation
 - Creates final output
 - Uses `{cc.attrs.attributeName}` as before
 - Uses `ui:repeat` to build output

More on ui:repeat

- **Sample usage**

```
<ul>
  <ui:repeat var="color" value="#{item.availableColors}">
    <li>#{color}</li>
  </ui:repeat>
</ul>
```

- **More attributes**

```
<ui:repeat var="someVar"
  value="#{someBean.someCollection}"
  varStatus="statusVariable"
  offset="..."
  size="..."
  step="...">
  ...
</ui:repeat>
```

- **More details on ui:repeat**

- See tutorial section on looping

35

Example

- **Pass in array or List**

- Turn it into `` list showing individual entries

- **Attributes**

- **value** (EL expression to designate collection)
- **styleClass** (to specify CSS style of resultant `` list)

- **Component**

- resources/**utils**/list.xhtml
 - Declares **value** and **styleClass** inside `composite:interface`
 - Generates list using `ui:repeat` (`composite:implementation`)
- composite-components.xhtml
 - Declares **utils**: namespace
 - Uses `<utils:list value="#{someBean.someCollection}" styleClass="some-css-name"/>`

36

Component File (resources/utills/list.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite"
      xmlns:ui="http://java.sun.com/jsf/facelets">
...
<composite:interface>
  <composite:attribute name="value" required="true"/>
  <composite:attribute name="styleClass" default="italic"/>
</composite:interface>

<composite:implementation>
  <ul class="#{cc.attrs.styleClass}">
    <ui:repeat var="listItem" value="#{cc.attrs.value}">
      <li>#{listItem}</li>
    </ui:repeat>
  </ul>
</composite:implementation>
</body></html>
```

37

Main File (composite-components.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:utills="http://java.sun.com/jsf/composite/utills">
...
<fieldset>
<legend>Composite Components that Use ui:repeat</legend>
<h2>First coreservlets.com Developer</h2>
<ul>
  <li>Level: #{person1.level}</li>
  <li>First name: #{person1.firstName}</li>
  <li>Last name: #{person1.lastName}</li>
  <li>Languages: <utills:list value="#{person1.languages}"
                          styleClass="blue"/></li>
</ul>
```

38

Main File: Continued (composite-components.xhtml)

```
<hr/>
<h2>Second coreservlets.com Developer</h2>
<p>
Our second developer (#{person2.level}-level) is
#{person2.firstName} #{person2.lastName}.
He is proficient in:</p>
<utils:list value="#{person2.languages}"/>
</fieldset>
<p/>
</fieldset>
```

39

Base Class for Beans

```
public class Programmer {
    private String firstName, lastName, level;
    private String[] languages;

    public Programmer(String firstName,
                      String lastName,
                      String level,
                      String... languages) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.level = level;
        this.languages = languages;
    }

    // Getter methods getFirstName, getLastName, etc.
    // Note that getLanguages returns an array
```

40

Bean: Person1

```
@ManagedBean
public class Person1 extends Programmer {
    public Person1() {
        super("Larry", "Ellison", "Junior",
            "SQL", "Prolog", "OCL", "Datalog");
    }
}
```

Bean: Person2

```
@ManagedBean
public class Person2 extends Programmer {
    public Person2() {
        super("Larry", "Page", "Junior",
            "Java", "C++", "Python", "Go");
    }
}
```

Results



43

© 2012 Marty Hall



Handling Relative URLs in Components

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 Simple Relative URLs

- **Issue**
 - Suppose component used simple relative URLs
 - Suppose main page was moved to sub directory
- **Example**
 - Hypertext links
 - `` would now refer to `http://host/context-root/subdir/welcome.jsf`
 - Images
 - `` would now refer to `http://host/context-root/subdir/images/pic.jpg`
 - Style sheets
 - `<link ...href="./css/styles.css"/>` would now refer to `http://host/context-root/subdir/css/styles.css`
 - JavaScript files
 - `<script src="./scripts/my-script.js"...>` would now refer to `http://host/context-root/subdir/scripts/my-script.js`

45

Solutions (Relocatable Resources)

- **Hypertext links**
 - Use ``
 - 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"/>`

46

Component File (resources/utills/info3.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html ... xmlns:composite="http://java.sun.com/jsf/composite"
      xmlns:utills="http://java.sun.com/jsf/composite/utills">
...
<composite:interface>
  <composite:attribute name="styleClass"/>
</composite:interface>
<composite:implementation>
  <table align="right">
    <tr><th>
      Actual file is resources/images/duke-guitar.png
      <h:graphicImage name="duke-guitar.png" library="images"/>
    </th></tr>
    <tr><th>
      <a href="#{request.contextPath}/composite-components.jsf">
        Return to main page</a>
      </th></tr>
  </table>
  <utills:info2 styleClass="#{cc.attrs.styleClass}"/>
</composite:implementation>
</body></html>
```

47

Main File 1 (relative-urls.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:utills="http://java.sun.com/jsf/composite/utills">
<h:head><title>Composite Components</title>
<h:outputStylesheet name="styles.css" library="css"/>
</h:head>
<h:body>
...
<fieldset>
<legend>Component that Uses Relative URLs</legend>
<utills:info3/>
</fieldset>
...
</h:body></html>
```

48

Main File 2 (someDirectory/relative-urls.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:utils="http://java.sun.com/jsf/composite/utils">
  <h:head><title>Composite Components</title>
  <h:outputStylesheet name="styles.css" library="css"/>
</h:head>
<h:body>
  ...
  <fieldset>
  <legend>Component that Uses Relative URLs</legend>
  <utils:info3/>
  </fieldset>
  ...
</h:body></html>
```

This is an exact copy of the file from the top-level directory.

49

Results

The image displays two screenshots of a Mozilla Firefox browser window. The top screenshot shows the page with the title "Composite Components" and a legend "Component that Uses Relative URLs". The bottom screenshot shows the same page but with a 3D character playing a guitar and a "Return to main page" link. Both screenshots show the browser's address bar with the URL "http://localhost/composite-components/relative-urls.jsf" and "http://localhost/composite-components/someDirectory/relative-urls.jsf" respectively.

50



Nested Components

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.

Idea

- **Two component files**
 - First one refers to second one
- **Design options**
 - Both components could be used independently
 - So, provide explicit attributes for both components
 - First component passes attributes to second component
 - Second (nested) component specific to first (outer)
 - Provide explicit attributes for first component
 - Second component directly reads attributes of first component by using `#{cc.parent.attrs.attributeName}`
- **Compatibility note**
 - Due to bug in very early JSF 2 versions, `#{cc.parent.attrs...}` works only in Mojarra 2.0.3 or later.

First Component File: Summary

- **In subfolder of “resources” folder**
 - E.g., resources/someDir/someFile.xhtml
- **Basic structure**
 - Looks like xhtml file as before
 - But also declares namespace for second component
 - Contains composite:interface
 - Defines attributes used as before
 - `<composite:attribute name="attributeName"/>`
 - Contains composite:implementation
 - Creates final output. Refers to attributes with cc.attrs.
 - Also refers to second component.
 - `...#{cc.attrs.attributeName}...`
 - `<someNamespace:someComponent/>`

53

Second Component File: Summary

- **In subfolder of “resources” folder**
 - E.g., resources/someDir/someFile.xhtml
- **Basic structure**
 - Looks like xhtml file as before
 - Contains composite:interface
 - Does not repeat attributes defined by first component
 - But could still define some new attributes
 - Contains composite:implementation
 - Creates final output.
 - Refers to locally-defined attributes with cc.attrs.
 - Refers to attributes of surrounding component with cc.parent.attrs
 - `...#{cc.parent.attrs.attributeName}...`

54

Example

- **Make a bulleted list describing programmer**
 - First name, last name, etc.
- **Use nested component for languages**
 - To build another `` list
- **Components**
 - resources/utills/programmerInfo.xhtml
 - Declares utills: namespace
 - Declares programmer inside composite:interface
 - Refers to `#{cc.attrs.programmer.property}`
 - Calls `<utills:langList/>`
 - resources/utills/langList.xhtml
 - Does not declare any attributes
 - Refers to `#{cc.parent.attrs.programmer.property}`

55

Component File 1 (resources/utills/programmerInfo.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite"
      xmlns:utills="http://java.sun.com/jsf/composite/utills">
...
<composite:interface>
  <composite:attribute name="programmer" required="true"/>
  <composite:attribute name="styleClass" default="italic"/>
</composite:interface>
<composite:implementation>
  <ul class="#{cc.attrs.styleClass}">
    <li>Level: #{cc.attrs.programmer.level}</li>
    <li>First name: #{cc.attrs.programmer.firstName}</li>
    <li>Last name: #{cc.attrs.programmer.lastName}</li>
    <li><utills:langList/></li>
  </ul>
</composite:implementation>
</body></html>
```

56

Component File 2 (resources/utils/langList.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite"
      xmlns:ui="http://java.sun.com/jsf/facelets">
...
<composite:interface/>

<composite:implementation>
  #{cc.parent.attrs.programmer.firstName}'s languages:
  <ul>
    <ui:repeat var="language"
              value="#{cc.parent.attrs.programmer.languages}">
      <li>#{language}</li>
    </ui:repeat>
  </ul>
</composite:implementation>
</body></html>
```

Reminder: cc.parent fails in Mojarra 2.0.2 and earlier.

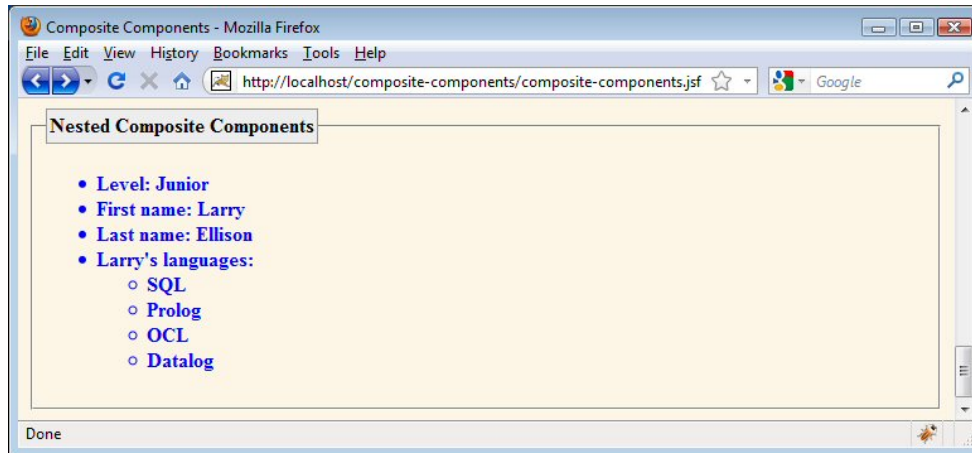
57

Main File (composite-components.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:utils="http://java.sun.com/jsf/composite/utils">
<h:head>
...
</h:head>
<h:body>
...
<fieldset>
<legend>Nested Composite Components</legend>
<utils:programmerInfo programmer="#{person1}"
                      styleClass="blue"/>
</fieldset>
...
</h:body></html>
```

58

Results



59

© 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

- **Define a component**
 - Put `foo.xhtml` in “resources/`bar`” folder
 - Define available attributes inside **composite:interface**
 - Use `<composite:attribute name="attributeName" ... />`
 - Build output inside **composite:implementation**
 - Use `#{cc.attrs.attributeName}`
 - Possibly use `ui:repeat`. Be careful with relative URLs.
- **Use component in main page**
 - Define namespace
 - `<html ...`
`xmlns:bar="http://java.sun.com/jsf/composite/bar">`
 - Use the component in facelets page
 - `<bar:foo attributeName="..." />`

61

© 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.