

## Templates in Java EE 7

### Overview

@author R.L. Martinez, Ph.D.

The following tutorial consists of steps to create a new Java EE 7 project named TemplateDemo. Templates are useful constructs which can greatly enhance developer productivity. In Java EE 7, templates are enabled using the JavaServer Faces framework (JSF).

JSF is a Java EE framework for web user interface development with supports the Model-View-Controller (MVC) pattern. MVC is an approach that separates application data (model) from the user interface (view) and the application logic (controller). Beginning with JSF version 2.0, the page definition language can use Facelets which is based on well-formed xml pages instead of JSP with previous versions of JSF.

### Creating and Using Java EE 7 Templates

Java EE 7 templates offer a considerable advantage over other template systems such as .dwt (Dreamweaver Web Templates or Dynamic Web Templates). In Java EE 7, the templates are dynamically assembled at runtime compared to the template static assembly at design time in other models. When using the .dwt model, templates are used to construct and then store each resulting page. So, if a template is used with 10 pages, each of those 10 pages has copies of the template HTML markup. With Java EE 7 on the other hand, the servlet container (Undertow in WildFly; Catalina in many Java web servers) combines the templates and clients at runtime. Therefore, the clients do not contain template code until runtime. Maintenance and storage are much more efficient with the Java EE 7 approach.

You may notice that NetBeans and the projects are installed on drive N:\, a USB flash drive. However, this is only done for portability and testing. It is strongly recommended that you install NetBeans and all other required software on the local C: drive. The configurations are much easier to manage and troubleshoot when all components are installed on C:\.

Also, you should be using the latest version of all development software listed in the tutorials. For instance, in the images below replace N:\NetBeans 7.4\projects with C:\NetBeans X.X\projects where X.X is the latest version. The same applies to the JDK, MySQL, JSF, phpMyAdmin, Spring MVC, etc. If significant

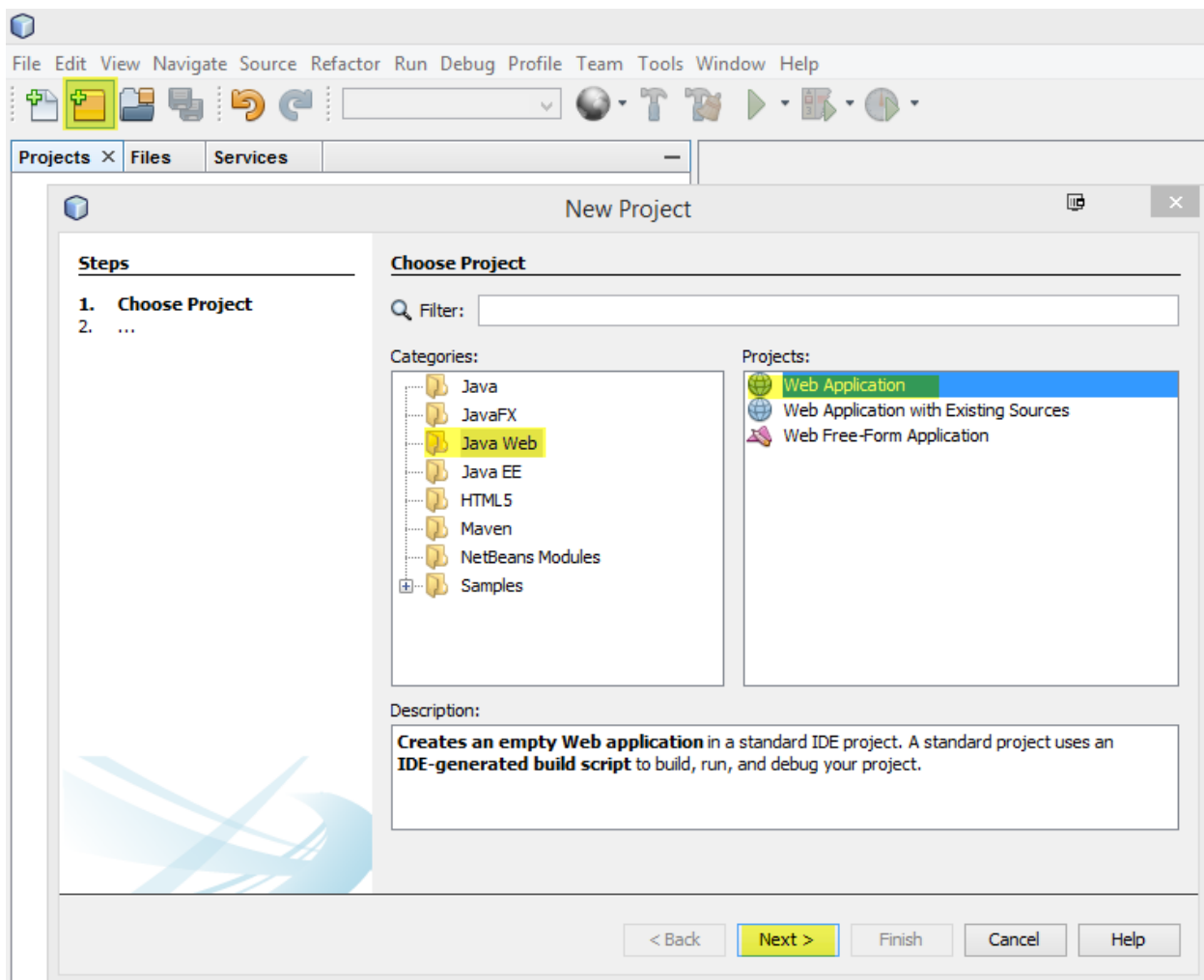
## Templates in Java EE 7

discrepancies between the versions shown and the latest then the divergences will be noted.

It is important to follow the steps very precisely to help reduce the chances for incorrect configurations and errors. Be patient and take your time. Also, repeating the tutorials in the course for practice will reinforce the concepts, enhance understanding, and improve skills.

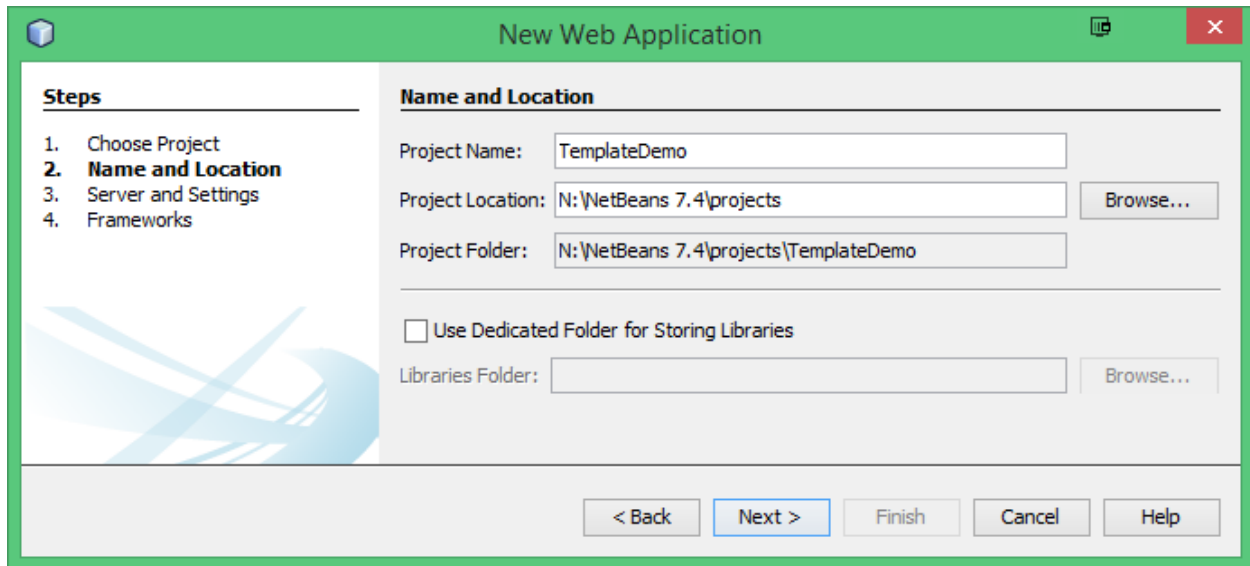
Steps:

Create a new project by clicking the New Project button, making the selections, and then click Next.

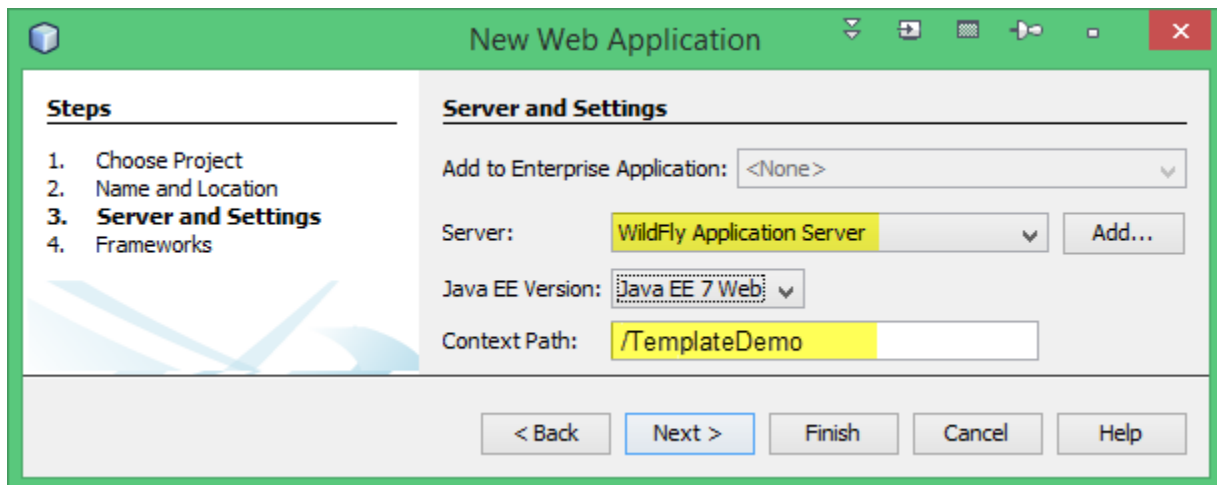


## Templates in Java EE 7

Make selections like those shown (your location/folder will likely be different) and then click Next.

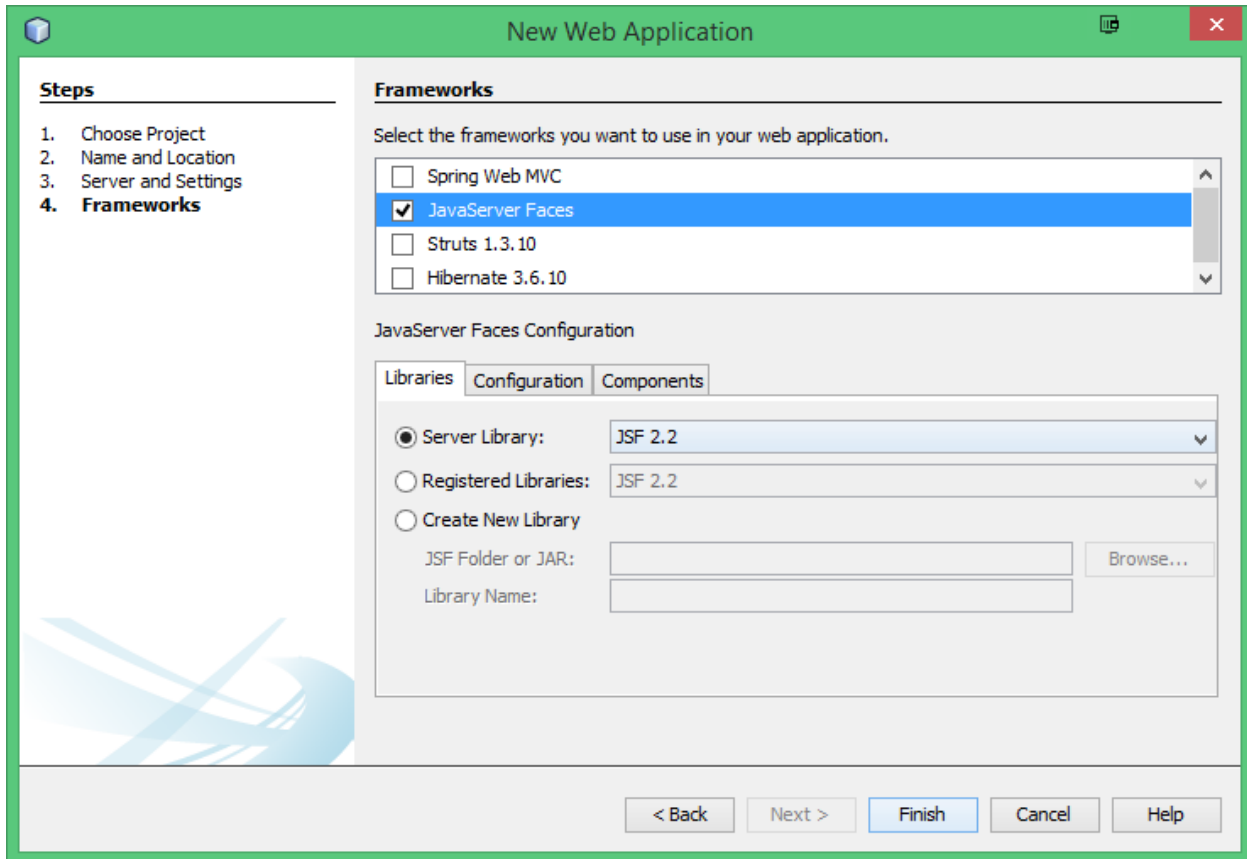


Make selections like those shown and then click Next. Notice the WildFly Server and Java EE 7 Web items are selected.



The next dialog box presents an opportunity to add framework(s) to the application. Select JavaServer Faces.

## Templates in Java EE 7



**New Web Application**

**Steps**

1. Choose Project
2. Name and Location
3. Server and Settings
4. **Frameworks**

**Frameworks**

Select the frameworks you want to use in your web application.

- ☐ Spring Web MVC
- ☒ **JavaServer Faces**
- ☐ Struts 1.3.10
- ☐ Hibernate 3.6.10

**JavaServer Faces Configuration**

Libraries Configuration Components

☒ **Server Library:** JSF 2.2

☐ Registered Libraries: JSF 2.2

☐ Create New Library

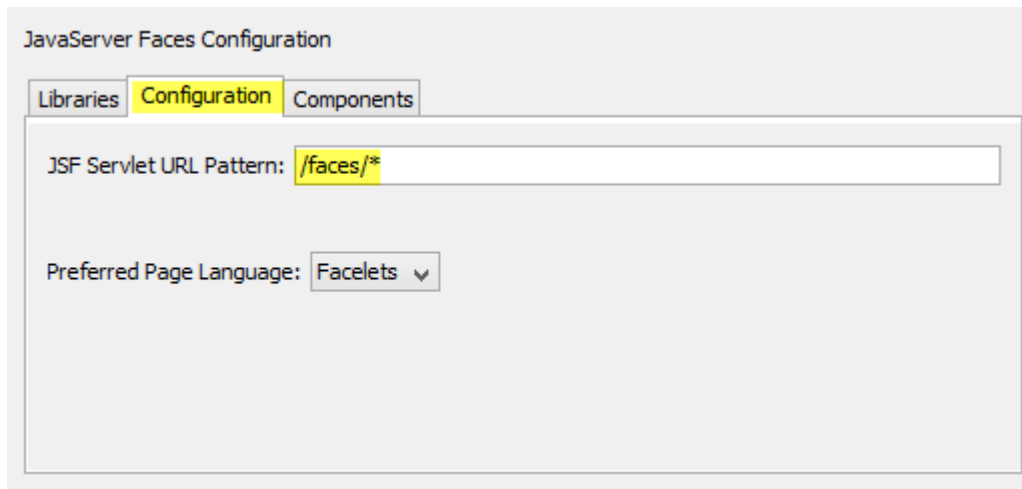
JSF Folder or JAR:

Library Name:

< Back Next > **Finish** Cancel Help

The configuration tab should reflect the options shown. Starting with JSF version 2.0, JSF can (and should) be constructed using Facelets as the preferred page view definition language. The older and less preferred JSP model is also possible. However, Facelets is the superior method since it is based on well-formed xhtml pages, template coding and processing is much improved, and Facelets do not employ the embedded Java code within the view page model used by JSP.

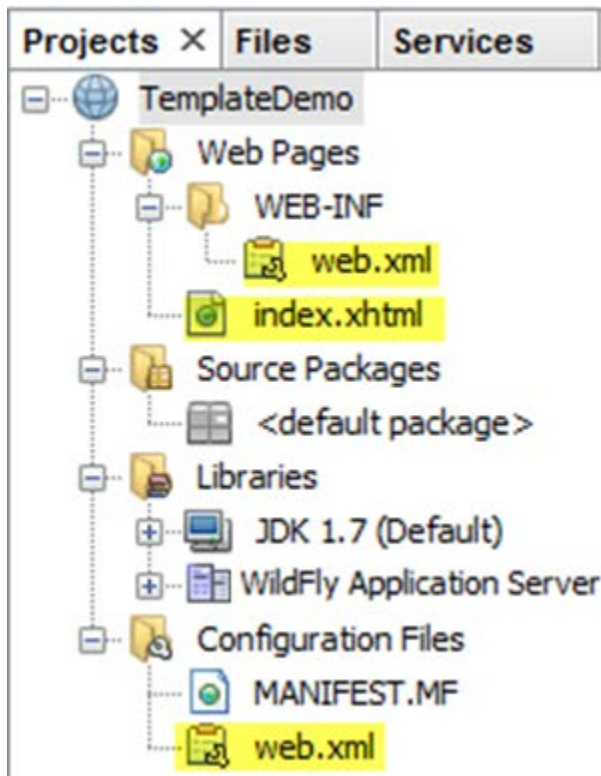
## Templates in Java EE 7



For this project, a component suite will not be added under the Components tab. We will add and use the PrimeFaces component suite in another tutorial. The component suites offer user interface functionality (components) beyond those offered by default with JSF.

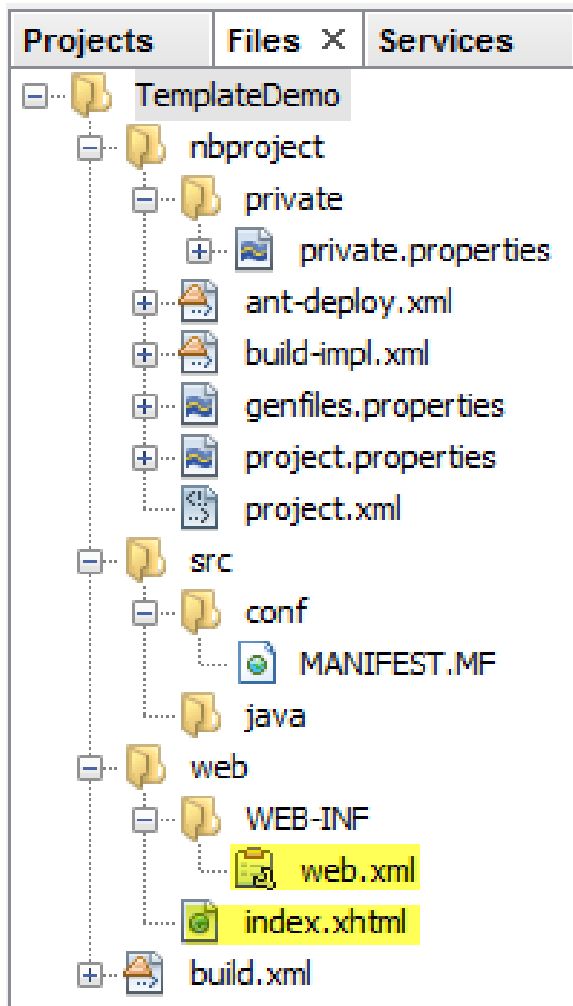
Select Finish in the New Web Application dialog. Review the project structure created by NetBeans for this Java web application based on Java EE 7, deployed on the WildFly server, and using the JSF framework. An index.xhtml and web.xml files were generated by NetBeans. The Project window offers a logical view of the project structure. Actual physical directory locations (as shown in the Files window) will be different from the virtual directory structure in the Projects window. Note that web.xml appears in two logical locations (Configuration Files and Web Pages/WEB-INF).

## Templates in Java EE 7



View the file and directory structure by selecting the Files window. Notice the locations of two files with which we will work, `index.xhtml` and `web.xml`. In particular, `web.xml` is depicted in its actual physical directory location of `web/WEB-INF/web.xml`. Also, `index.xhtml` is shown in `web/index.xhtml`. Keep in mind that the `/faces/*` URL pattern was specified when creating the project. This can be observed in `web.xml` on lines 17 and 25. More on this when `web.xml` is discussed after reviewing `index.html`.

## Templates in Java EE 7



The starting index.xhtml is shown which should be automatically opened when the project was created. Notice the `<h:...>` tags. The `h:` is referred to as a name prefix and is used in xhtml pages to prevent naming conflicts. When name prefixes are used, an xml namespace (xmlns) must be defined which is accomplished on line 5. More on this when additional name prefixes are utilized.

More here:

[http://www.w3schools.com/xml/xml\\_namespaces.asp](http://www.w3schools.com/xml/xml_namespaces.asp)

## Templates in Java EE 7

```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml"
5       xmlns:h="http://xmlns.jcp.org/jsf/html">
6   <h:head>
7       <title>Facelet Title</title>
8   </h:head>
9   <h:body>
10       Hello from Facelets
11   </h:body>
12 </html>
```

Double-click web.xml to open the file. The file was automatically created by NetBeans to establish configuration information for using JSF with Facelets. Notice on line 17 that the URL pattern used by the application is /faces/\* which was the default specified during project creation. Also notice on line 25 that the welcome-file is faces/index.xhtml. The welcome file is the file that the server responds with when the site (web application) is accessed by the site top-level URL (e.g. java.net, jcp.org, java.com). From the Project window review above, recall that the actual physical directory location of the file is web/index.xhtml. The faces/index.xhtml logical association is accomplished by JSF.



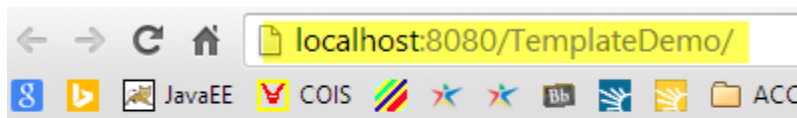
## Templates in Java EE 7

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
6      <context-param>
7          <param-name>javax.faces.PROJECT_STAGE</param-name>
8          <param-value>Development</param-value>
9      </context-param>
10     <servlet>
11         <servlet-name>Faces Servlet</servlet-name>
12         <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
13         <load-on-startup>1</load-on-startup>
14     </servlet>
15     <servlet-mapping>
16         <servlet-name>Faces Servlet</servlet-name>
17         <url-pattern>/faces/*</url-pattern>
18     </servlet-mapping>
19     <session-config>
20         <session-timeout>
21             30
22         </session-timeout>
23     </session-config>
24     <welcome-file-list>
25         <welcome-file>faces/index.xhtml</welcome-file>
26     </welcome-file-list>
27 </web-app>

```

R-click TemplateDemo in the Project window and select Run to see the output below. Notice that the top-level URL is used to access the application when Run is selected.



Hello from Facelets

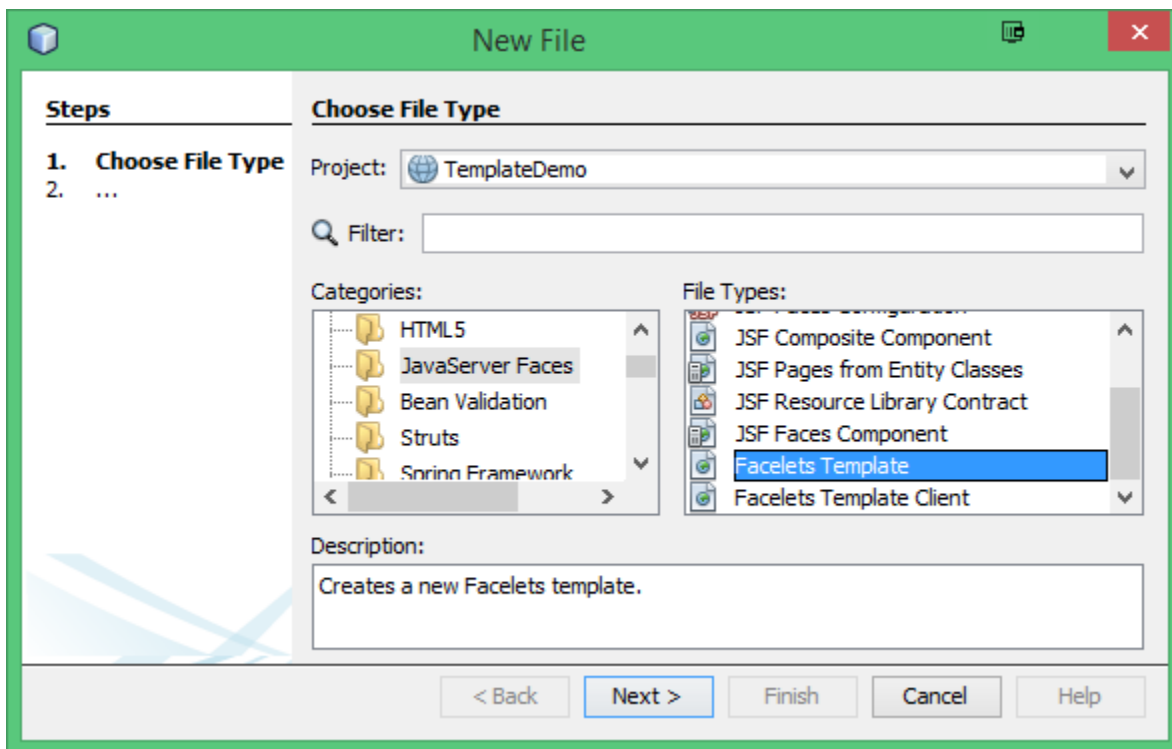
To confirm that the faces/index.xhtml welcome-file is being served, the full logical URL to the file can be specified to see the following output which is the same page as served above.

## Templates in Java EE 7



Now that the fundamental project structure is in place, we are ready to add a Facelets Template. After adding a template to the project, we will add Facelets Template Clients that use the template.

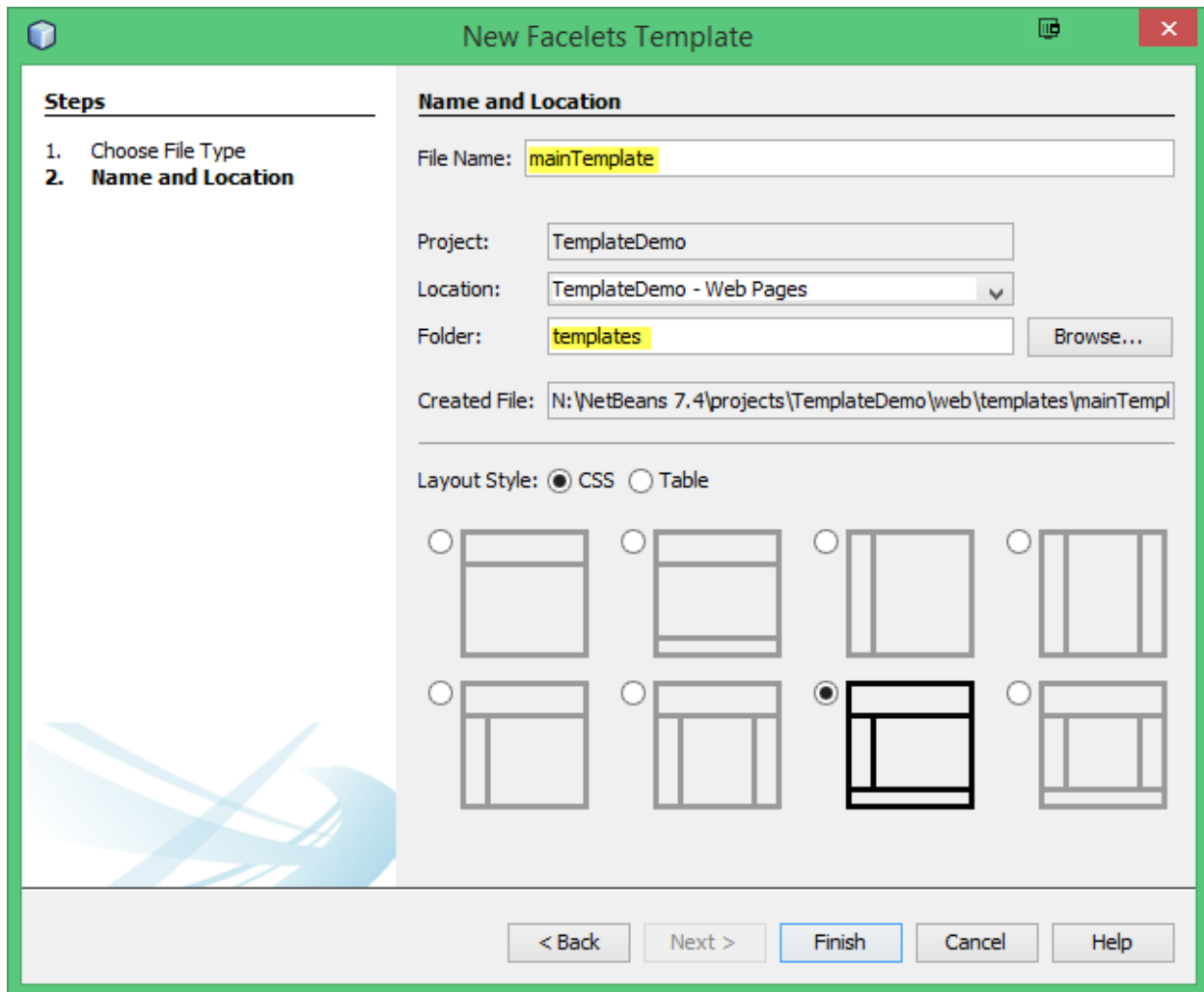
R-click TemplateDemo in the Projects window and select New | Facelets Template. If Facelets Template does not appear as an option you may need to select Other at the bottom of the fly-out menu and then JavaServer Faces | Facelets Template like shown. Select Next.



Make the selections as shown in the New Facelets Template dialog. The templates folder will be created under the Web Pages virtual folder (and the web physical folder). There are eight beginning/common layouts from which to choose. Select

## Templates in Java EE 7

the one depicted which includes a header, left navigation, content, and footer areas. JSF refers to these areas as top, left, content, and bottom. A developer can start with any of the samples shown and modify the template later. Select Finish.

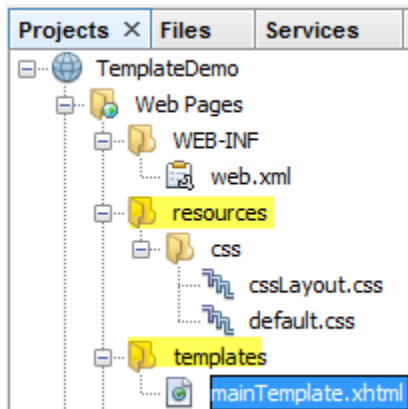


Notice in the Projects window that two new folders are added, resources and templates. The templates folder contains the mainTemplate.xhtml we just created and will inspect shortly.

The resources folder contains the css folder which contains two CSS files automatically generated by NetBeans which we will review soon. The CSS files provide starting style rules for the mainTemplate. For instance, we should expect

## Templates in Java EE 7

to find CSS rules for style properties associated with the top, left, content, and bottom areas of the template.



The mainTemplate.xhtml file should be open and appear like that shown. On line 5, a namespace for a name prefix (ui:) is specified. The ui: name prefix is part of the Facelets namespace. The ui: name prefix is used in the file to identify insert areas into which template clients can insert custom content.

The ui: prefixes are used on lines 18, 22, 25, and 29. The areas coincide with top, left, content, and bottom portions of the page. By specifying `<ui: insert...>`, template clients can potentially provide their own content for those areas. ui; prefixes are also used in template clients but the clients use `<ui: define...>` instead of `<ui: insert...>`.

Notice the two JSF outputStylesheet tags on lines 10 and 11. These tags render HTML link elements of type text/css which refer to the CSS style sheets automatically created by NetBeans and used by the page.

## Templates in Java EE 7

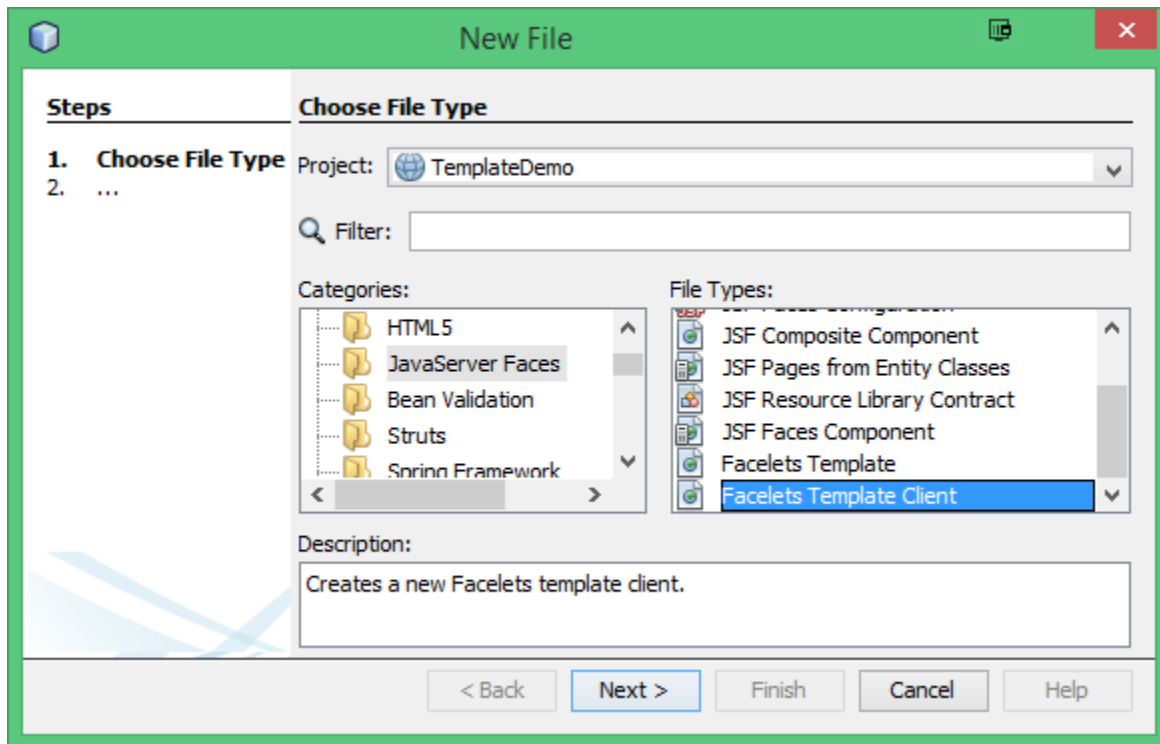
```

1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml"
5      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
6      xmlns:h="http://xmlns.jcp.org/jsf/html">
7
8      <h:head>
9          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
10         <h:outputStylesheet name="./css/default.css"/>
11         <h:outputStylesheet name="./css/cssLayout.css"/>
12         <title>Facelets Template</title>
13     </h:head>
14
15     <h:body>
16
17         <div id="top">
18             <ui:insert name="top">Top</ui:insert>
19         </div>
20         <div>
21             <div id="left">
22                 <ui:insert name="left">Left</ui:insert>
23             </div>
24             <div id="content" class="left_content">
25                 <ui:insert name="content">Content</ui:insert>
26             </div>
27         </div>
28         <div id="bottom">
29             <ui:insert name="bottom">Bottom</ui:insert>
30         </div>
31
32     </h:body>
33
34 </html>

```

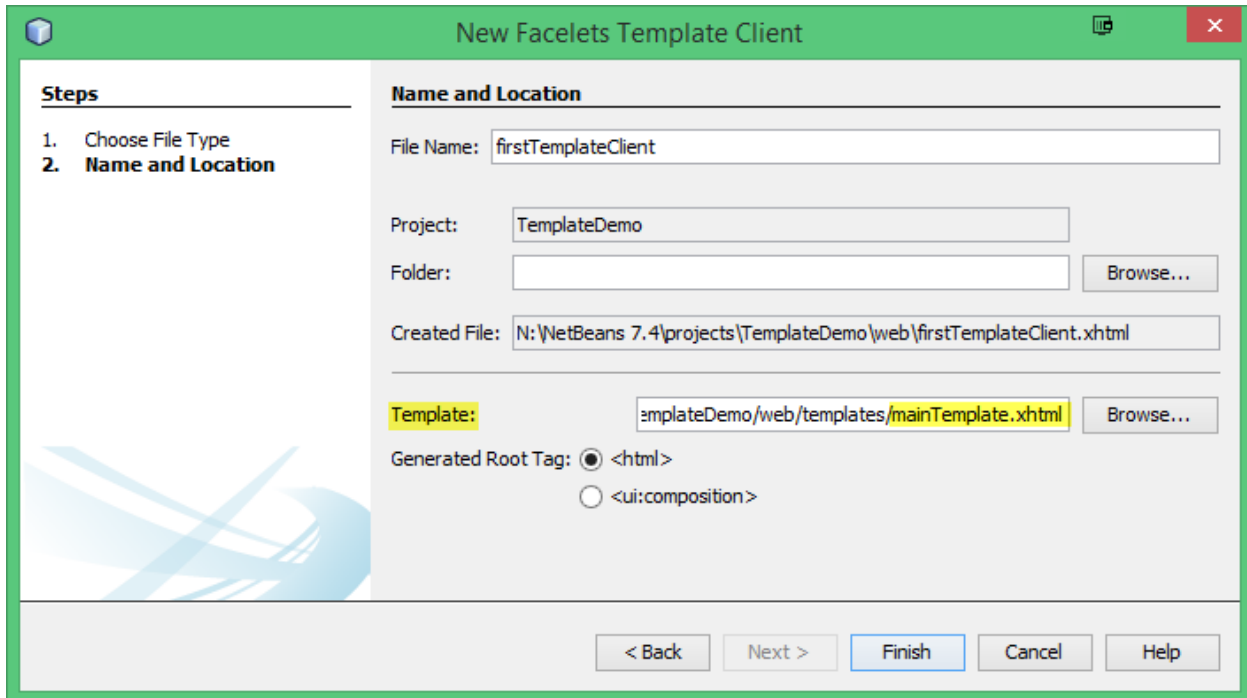
Now that we have a template, we are now ready to create a Facelets Template Client. R-click TemplateDemo in the Project Window and select Facelets Template Client. If it does not appear in the fly-out menu select Other | JavaServer Faces | Facelets Template Client and then Next.

## Templates in Java EE 7



Make the selections shown. Note that this file, `firstTemplateClient.xhtml`, will be based on the `mainTemplate.xhtml` template.

## Templates in Java EE 7



The image shows a 'New Facelets Template Client' dialog box with a green title bar. On the left, a 'Steps' pane shows '1. Choose File Type' and '2. Name and Location' (the current step). The main area is titled 'Name and Location' and contains the following fields and controls:

- File Name:** A text field containing 'firstTemplateClient'.
- Project:** A text field containing 'TemplateDemo'.
- Folder:** An empty text field with a 'Browse...' button to its right.
- Created File:** A text field showing the path 'N:\NetBeans 7.4\projects\TemplateDemo\web\firstTemplateClient.xhtml'.
- Template:** A text field containing ':emplateDemo/web/templates/mainTemplate.xhtml' (note the typo in the image) with a 'Browse...' button to its right.
- Generated Root Tag:** Two radio buttons. The first is selected and labeled '<html>'. The second is labeled '<ui:composition>'.

At the bottom of the dialog are five buttons: '< Back', 'Next >', 'Finish' (highlighted in blue), 'Cancel', and 'Help'.

The firstTemplateClient.xhtml is created, opened, and is as shown. Notice the top, left, content, and bottom ui: define sections that were automatically created by NetBeans. These are the same sections contained within mainTemplate.xhtml upon which the firstTemplateClient.xhtml is based. In the current example, each of the ui: define tags on the client has a corresponding ui: insert tag in the template. Notice the template relationship is established on line 9 by referencing mainTemplate.xhtml via a ui: composition tag and the template property.

## Templates in Java EE 7

```

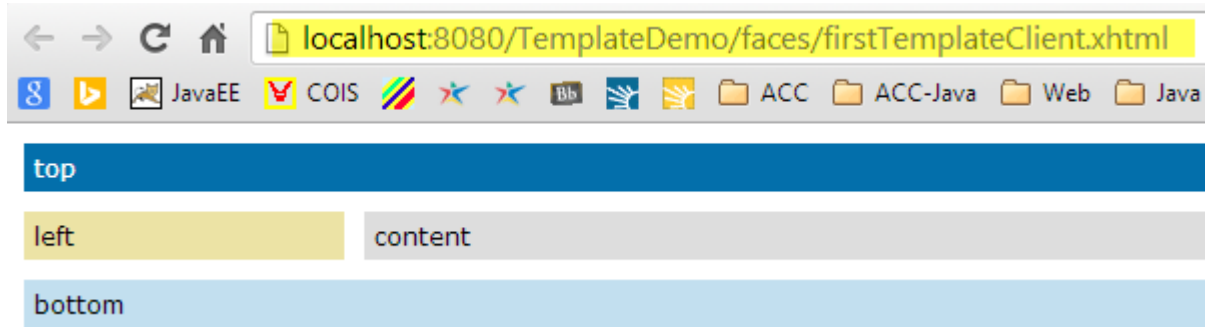
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml"
5      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
6
7      <body>
8
9          <ui:composition template="./templates/mainTemplate.xhtml">
10
11              <ui:define name="top">
12                  top
13              </ui:define>
14
15              <ui:define name="left">
16                  left
17              </ui:define>
18
19              <ui:define name="content">
20                  content
21              </ui:define>
22
23              <ui:define name="bottom">
24                  bottom
25              </ui:define>
26
27          </ui:composition>
28
29      </body>
30  </html>

```

Let's test the relationship between the template and the client by R-clicking firstTemplateClient.xhtml and selecting Run. The resulting page is shown below. Notice that the sections top, left, content, and bottom are displayed. The colors and positions of the sections are determined by the CSS files referenced on lines 10 and 11 of mainTemplate.xhtml



## Templates in Java EE 7



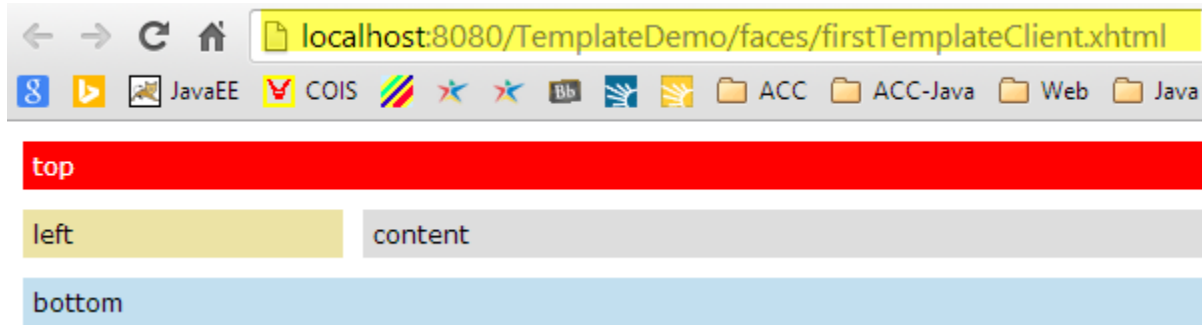
The CSS files are in the web/resources/css folder. A partial listing of cssLayout.css is shown. The id selectors top and bottom are shown. The file also contains id selectors for left and right even though right is not used in the mainTemplate.xhtml or in firstTemplateClient.xhtml. The top id selector is used to apply styles to the div in mainTemplate.xhtml with the id="top". To see the effect of a simple change to the top div change the background-color to red as shown on line 4 in the CSS file.

```

1  #top {
2      position: relative;
3      /*    background-color: #036fab;*/
4      background-color: red;
5      color: white;
6      padding: 5px;
7      margin: 0px 0px 10px 0px;
8  }
9
10 #bottom {
11     position: relative;
12     background-color: #c2dfef;
13     padding: 5px;
14     margin: 10px 0px 0px 0px;
15 }
```

Notice that the top div now has a red background. Restore the default setting by uncommenting line 3 and removing line 4.

## Templates in Java EE 7



Let's make another change to better understand templates. Comment the bottom section in firstTemplateClient.xhtml as shown (lines 23-25).

```

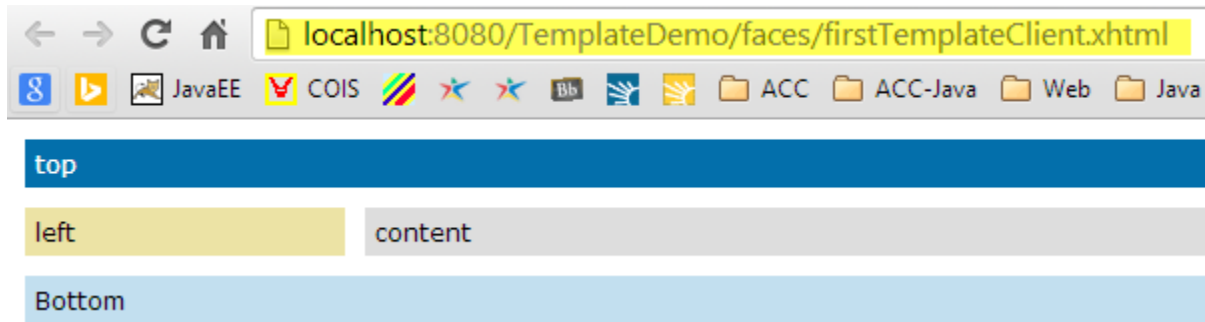
19         <ui:define name="content">
20             content
21         </ui:define>
22
23     <!--         <ui:define name="bottom">
24                 bottom
25         </ui:define>-->

```

Refresh the page and notice that bottom is still there. However, something is different. The content in the bottom section (the innerhtml) now has an uppercase B. Why? Look at the bottom div in mainTemplate.xhtml (lines 28-30). It is spelled with an uppercase B. This confirms that HTML markup in the template file will be rendered unless overridden by that of the client.

What happens in the converse situation? That is, what will occur if the client contains ui: define sections that do not have corresponding ui: insert tags in the template? Items that appear within the ui: composition tags in the client but do not have corresponding ui: insert tags in the template are ignored. However, elements outside of the ui: composition tag are rendered. Remove the comments from lines 23-25 and save.

## Templates in Java EE 7



Now that we have used JSF and Facelets to create a template and a template client that uses the template, we are ready to expand coverage of JSF by working with a component suite of user interface controls. In the next tutorial, the Primefaces component suite is added to the project to extend the user interface capabilities of JSF.