Component Suite in JSF

Overview

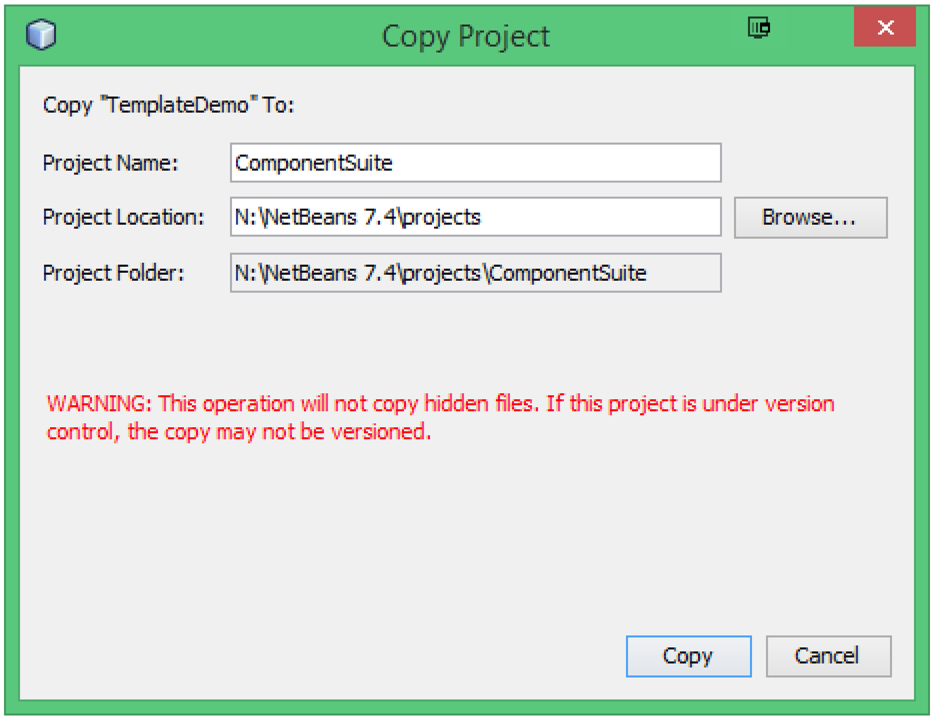
@author R.L. Martinez, Ph.D.

In the previous tutorial we used the JSF framework along with the Facelets view definition language to create a template and a client page that utilized the template structure and settings. By using NetBeans, the CSS files that define the style settings for the pages were automatically created. In this tutorial, we will copy the TemplateDemo project and name it ComponentSuite.

Using the PrimeFaces Component Suite with JSF

In this tutorial, we add the PrimeFaces component suite to the TemplateDemo project. PrimeFaces includes a number of user interface controls that can be used in conjunction with JSF to produce rich user interface functionality for web applications. PrimeFaces relies upon the popular jQuery library to accomplish much of its user interface enhancement. We will add lightbox and password controls to the project to demonstrate use of the suite.

Begin by R-clicking TemplateDemo in the Project group and selecting copy. Name the new project ComponentSuite and save to your projects folder. Disregard the warning since we are not using a version control system.



Copy "TemplateDemo" To:

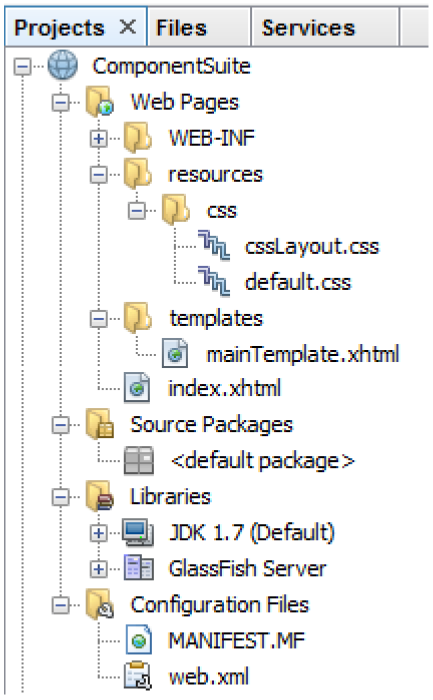
Project Name: ComponentSuite

Project Location: N:\NetBeans 7.4\projects

Project Folder: N:\NetBeans 7.4\projects\ComponentSuite

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

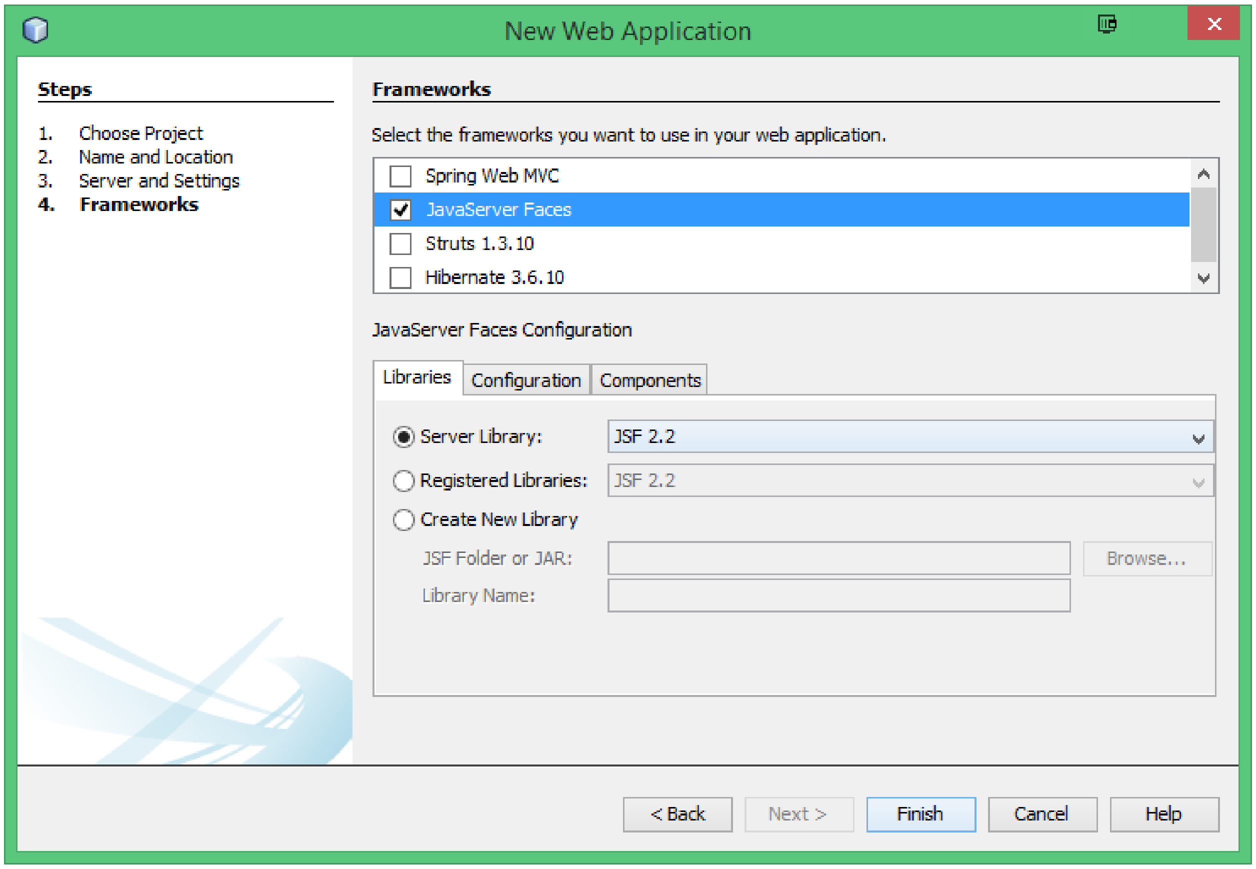
Note that the CSS files and mainTemplate.xhtml exist. firstTemplateClient.xhtml should be deleted since we will modify the index.xhtml file to the use the mainTemplate.xhtml and PrimeFaces. Your project structure should appear like that shown.



```
ComponentSuite
  Web Pages
    WEB-INF
    resources
      css
        cssLayout.css
        default.css
    templates
      mainTemplate.xhtml
    index.xhtml
  Source Packages
    <default package>
  Libraries
    JDK 1.7 (Default)
    GlassFish Server
  Configuration Files
    MANIFEST.MF
    web.xml
```

We are now ready to add the PrimeFaces component suite.

When creating a new project in NetBeans, a dialog like that shown is presented in which the addition of frameworks is the fourth step. However, since we are starting with an existing project (ComponentSuite), we will move to the next dialog which enables us to add a component suite to an existing project.


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.

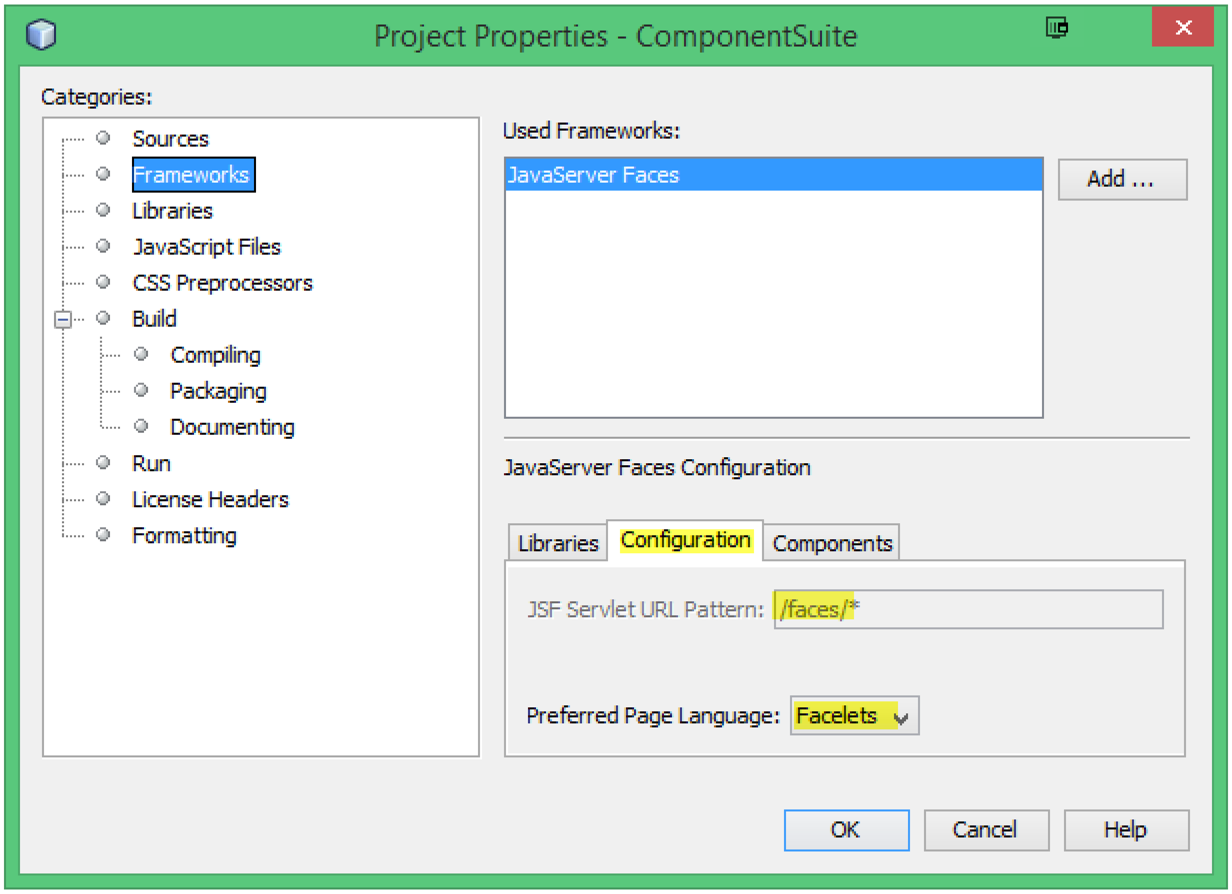
- [ ] Spring Web MVC
- [x] JavaServer Faces
- [ ] Struts 1.3.10
- [ ] Hibernate 3.6.10

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:

To add a component suite to an existing web application (a.k.a project), R-click the project name, ComponentSuite | Properties | Frameworks and the dialog is shown. Note that the JavaServer Faces framework had been previously added to TemplateDemo which was copied to create the ComponentSuite project.



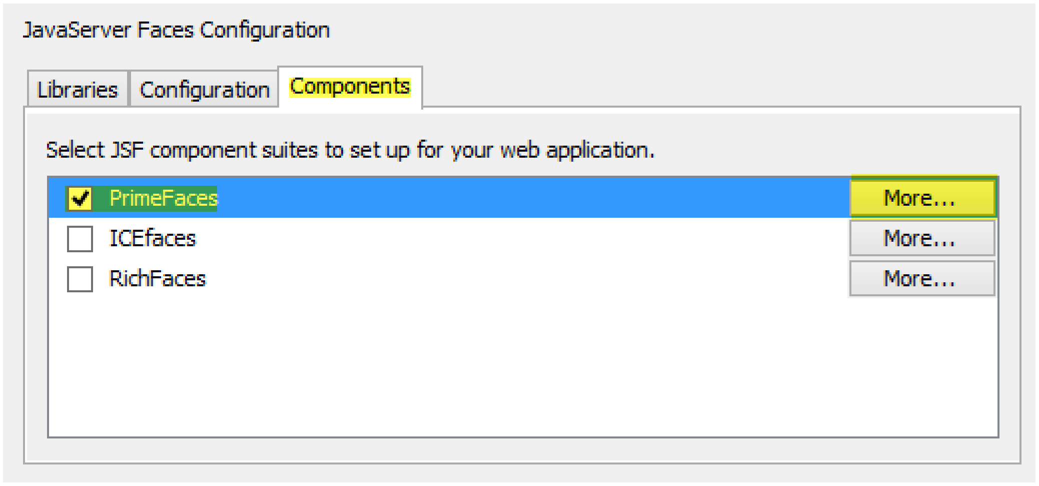
Used Frameworks: JavaServer Faces

JavaServer Faces Configuration

JSF Servlet URL Pattern: /faces/*

Preferred Page Language: Facelets

Select the Components tab to see the list of components currently available.

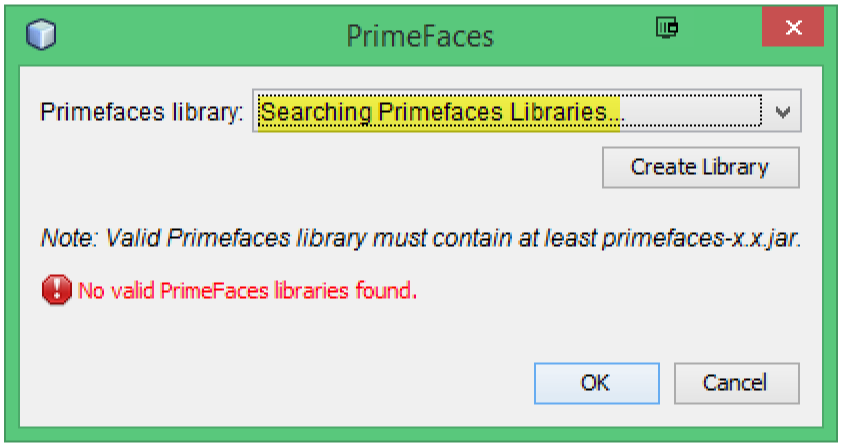


JavaServer Faces Configuration

Select JSF component suites to set up for your web application.

- [✓] PrimeFaces
- [ ] ICEfaces
- [ ] RichFaces

This dialog appears even if PrimeFaces libraries are installed. It may require a minute or so to locate any PrimeFaces libraries.



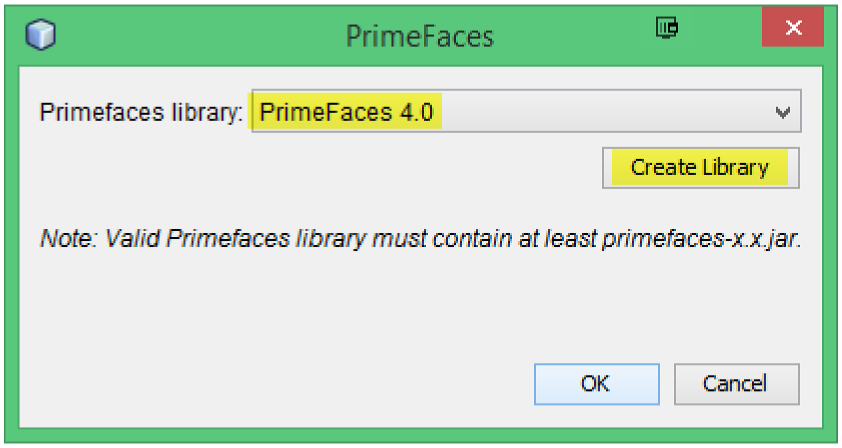
PrimeFaces

Primefaces library: Searching Primefaces Libraries...

Note: Valid Primefaces library must contain at least primefaces-x.x.jar.

No valid PrimeFaces libraries found.

If you have the PrimaeFaces 4.0 library already configured in NetBeans you should see the dialog below and can select OK. If the PrimeFaces 4.0 library is **NOT** configured in NetBeans, click Create Library.

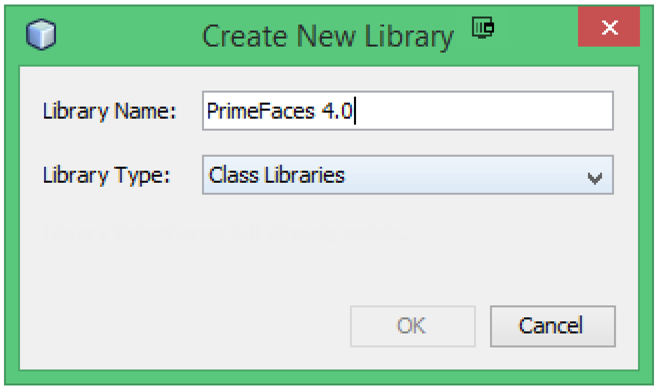


PrimeFaces

Primefaces library: PrimeFaces 4.0

Note: Valid Primefaces library must contain at least primefaces-x.x.jar.

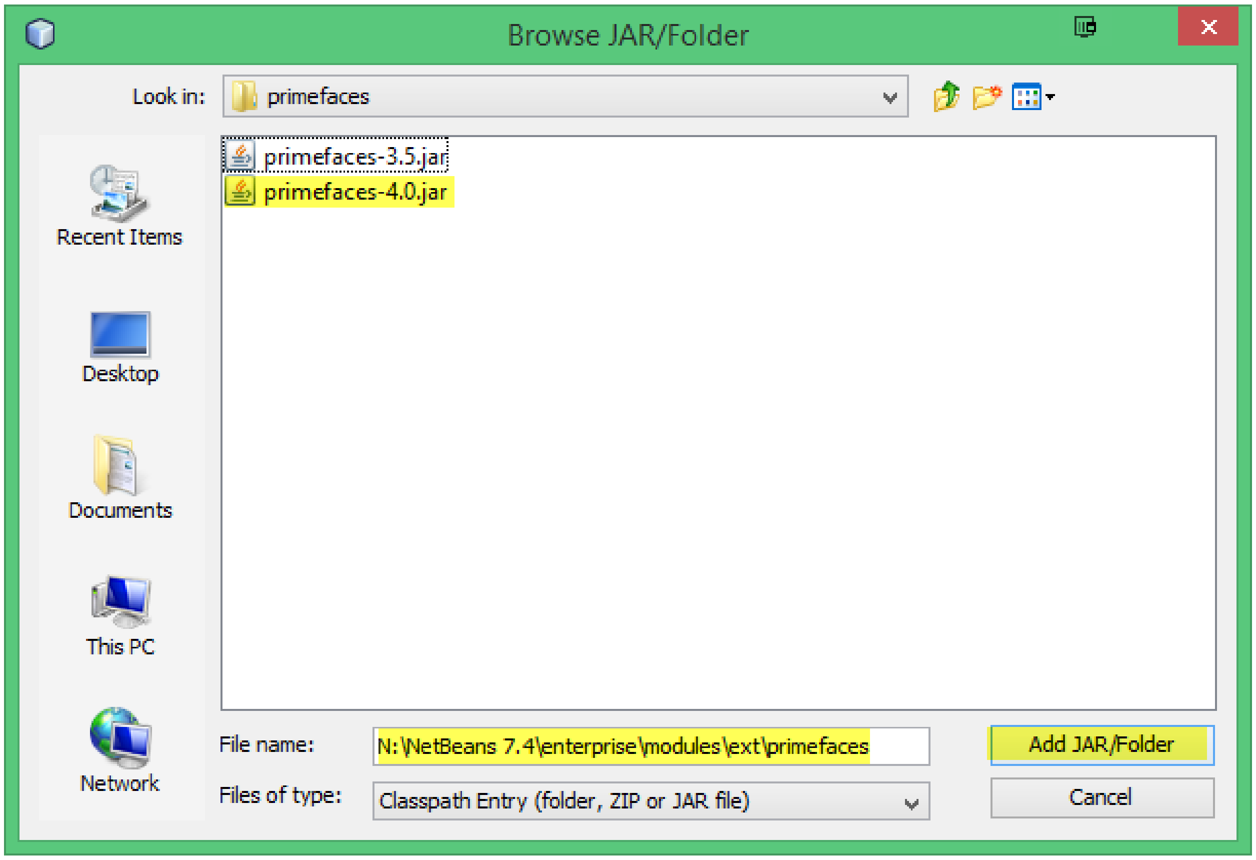
Name the new library PrimeFaces 4.0 in the Create New Library box.



Library Name: PrimeFaces 4.0

Library Type: Class Libraries

Navigate to the location where the primefaces-4.0.jar is located. See the location highlighted for an example. If you have **NOT** downloaded and installed the primefaces-4.0.jar file in a directory, see the next step and come back to the Browse JAR/Folder dialog.



Look in: primefaces

primefaces-3.5.jar
primefaces-4.0.jar

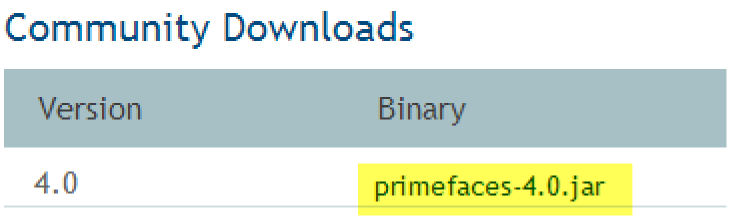
File name: N:\NetBeans 7.4\enterprise\modules\ext\primefaces

Files of type: Classpath Entry (folder, ZIP or JAR file)

If you need to download primefaces-4.0.jar, download it from:

http://primefaces.org/downloads

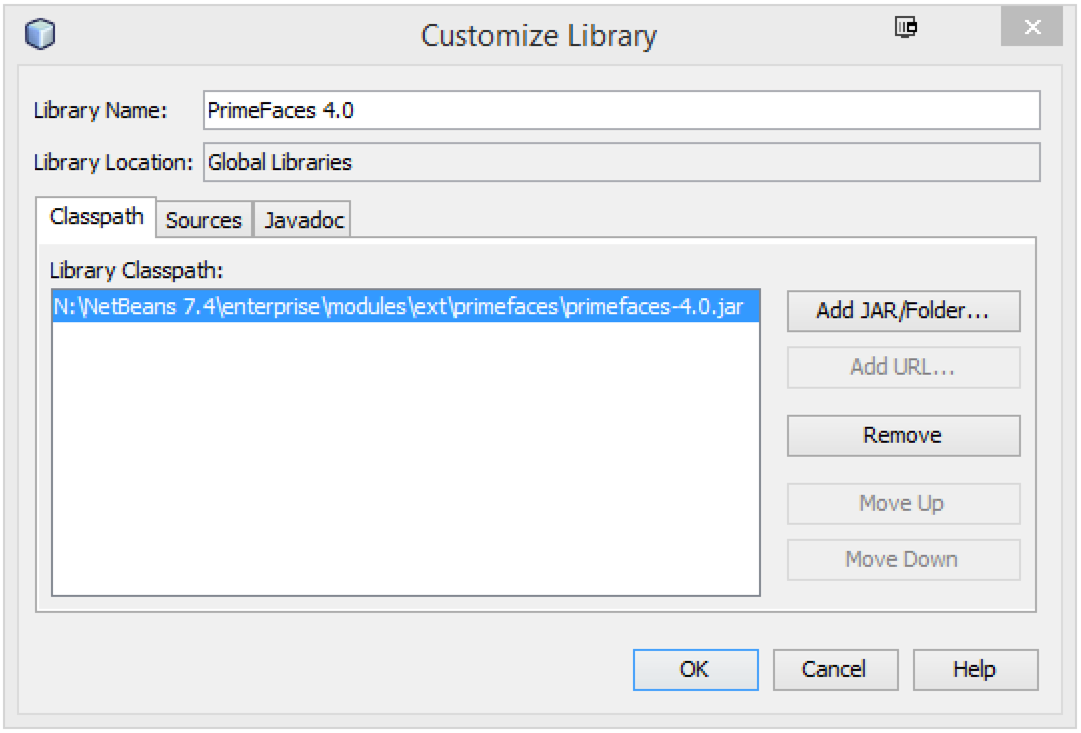
Make this selection shown to download. After the download, move the file to the "File Name" location like that shown in the Browse JAR/Folder dialog above.



Community Downloads

Version	Binary
4.0	primefaces-4.0.jar

After clicking Add JAR/Folder in the Browse JAR/Folder dialog, the Customize Library box is shown. Click OK to add the new Library with the associated primefaces-4.0.jar file.



Customize Library

Library Name: PrimeFaces 4.0

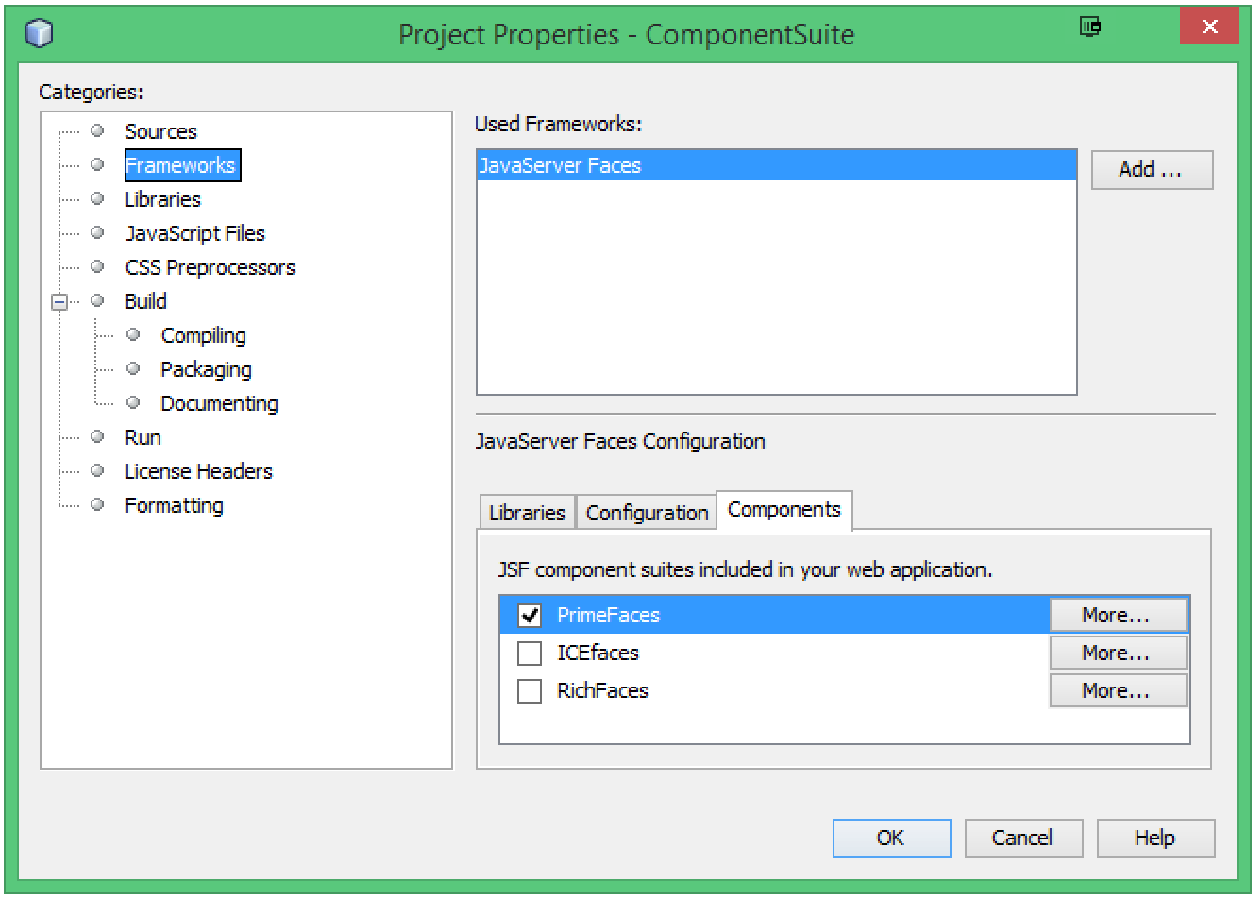
Library Location: Global Libraries

Classpath | Sources | Javadoc

Library Classpath:

N:\NetBeans 7.4\enterprise\modules\ext\primefaces\primefaces-4.0.jar

Now click OK to complete the addition of the component suite.


Project Properties - ComponentSuite

Categories:

- Sources
- Frameworks
- Libraries
- JavaScript Files
- CSS Preprocessors
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- License Headers
- Formatting

Used Frameworks:

JavaServer Faces

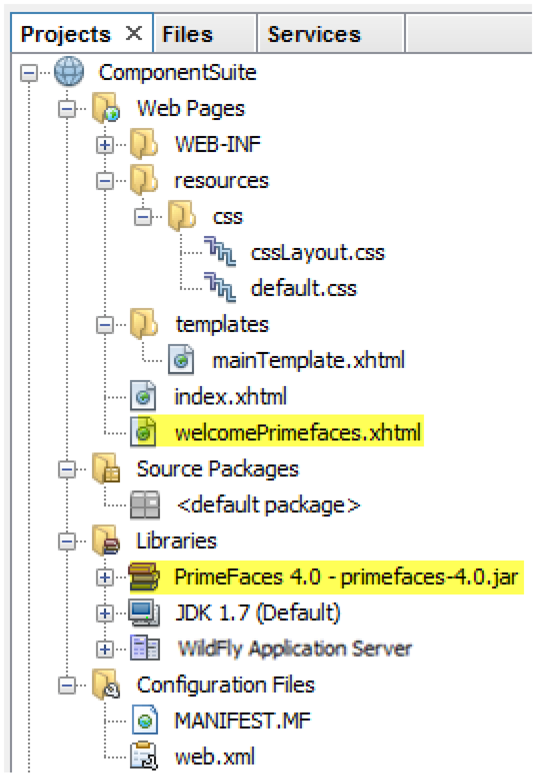
JavaServer Faces Configuration

Libraries | Configuration | Components

JSF component suites included in your web application.

- [✓] PrimeFaces
- [ ] ICEfaces
- [ ] RichFaces

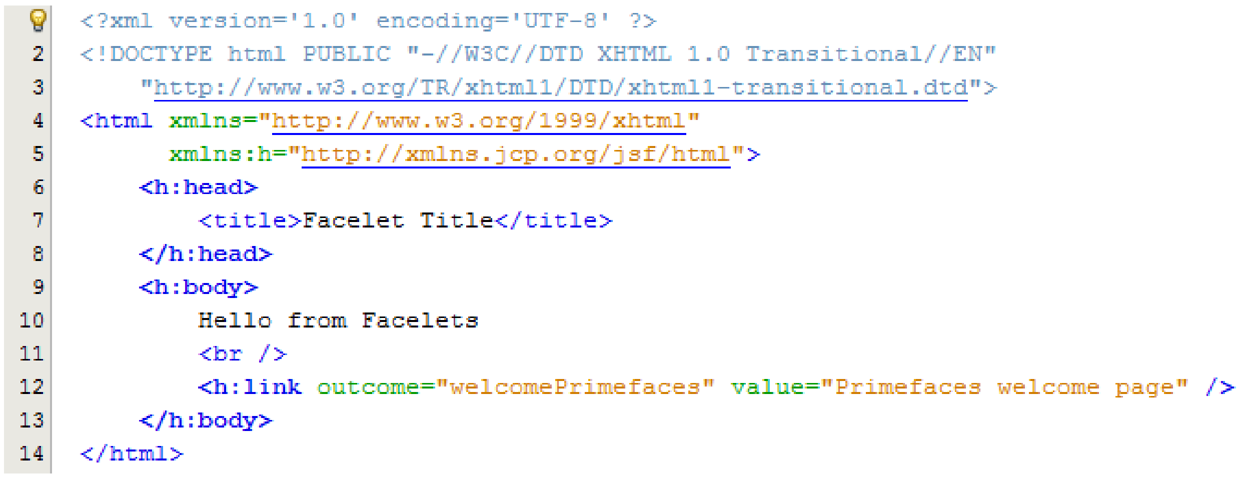
Additions to the ComponentSuite project are highlighted. The PrimeFaces 4.0 library contains packages and classes that will be used in this tutorial and subsequent sections. The welcomePrimefaces.xhtml was automatically added to the project and is accessed from a link added to index.xhtml.



```
ComponentSuite
    Web Pages
        WEB-INF
        resources
            css
                cssLayout.css
                default.css
        templates
            mainTemplate.xhtml
        index.xhtml
        welcomePrimefaces.xhtml
    Source Packages
        <default package>
    Libraries
        PrimeFaces 4.0 - primefaces-4.0.jar
        JDK 1.7 (Default)
        WildFly Application Server
    Configuration Files
        MANIFEST.MF
        web.xml
```

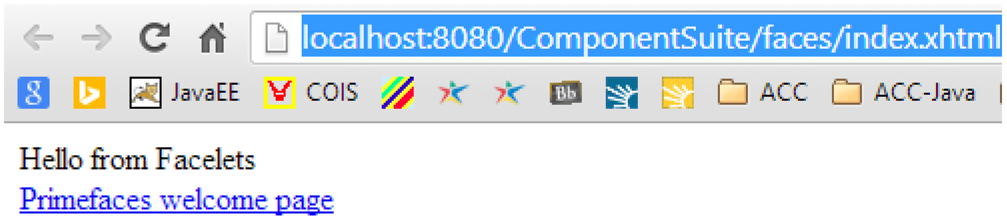
Open the index.xhtml and notice that line 12 was added when the component suite was included.

It is a link to the welcomePrimefaces.xhtml file.



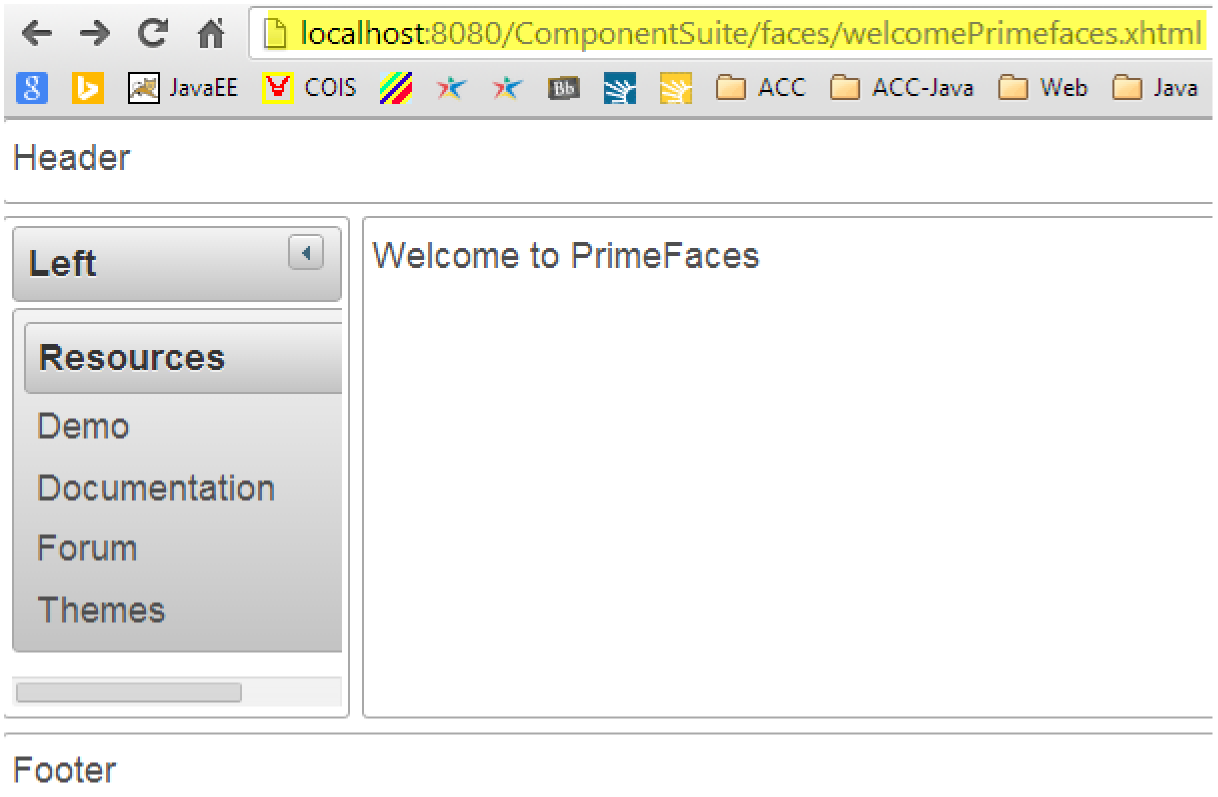
```
1   <?xml version='1.0' encoding='UTF-8' ?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4   <html xmlns="http://www.w3.org/1999/xhtml"
5         xmlns:h="http://xmlns.jcp.org/jsf/html">
6       <h:head>
7  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

Run the index.xhtml file by R-clicking the file and selecting Run. The output is shown.



Hello from Facelets
Primefaces welcome page

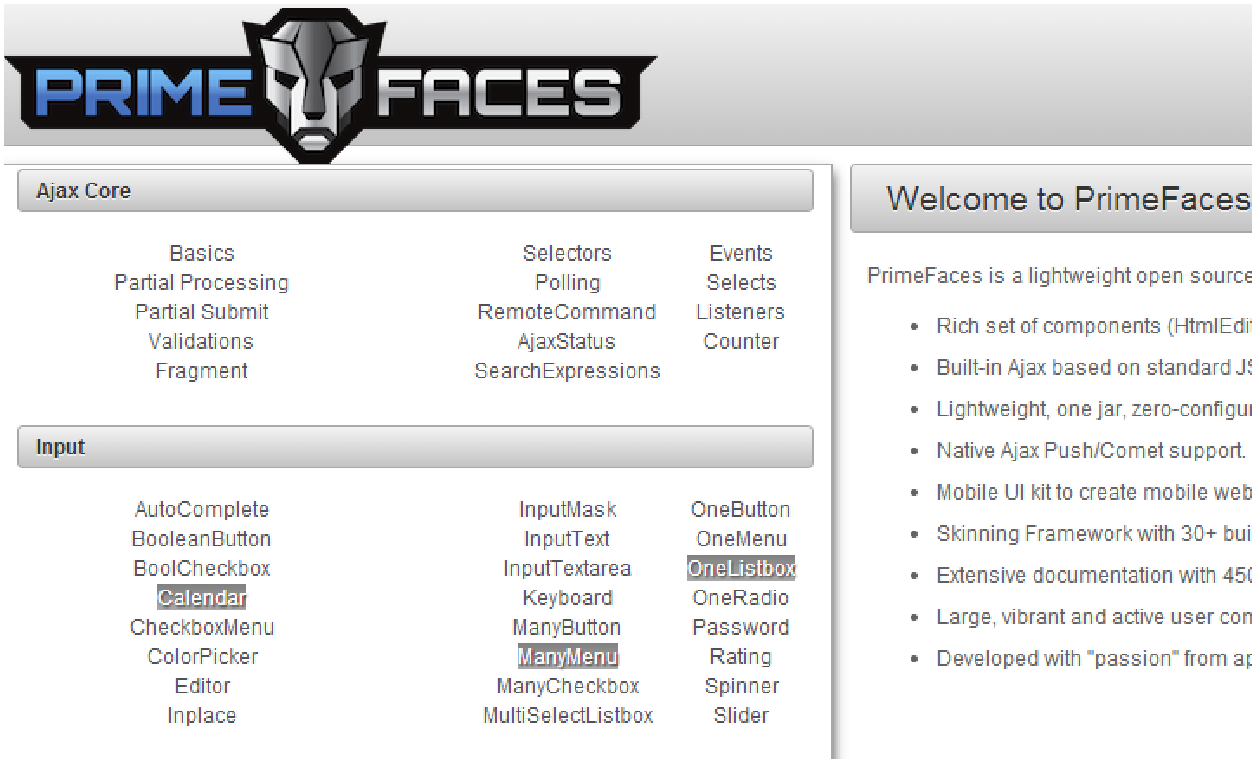
Click the link PrimeFaces welcome page on the index.xhtml page to access the page shown.



Welcome to PrimeFaces

Component Suite in JSF

Click the Demo menu item to access the PrimeFaces demo page, some of which is shown.

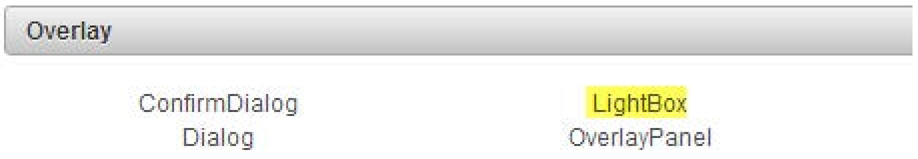


Welcome to PrimeFaces

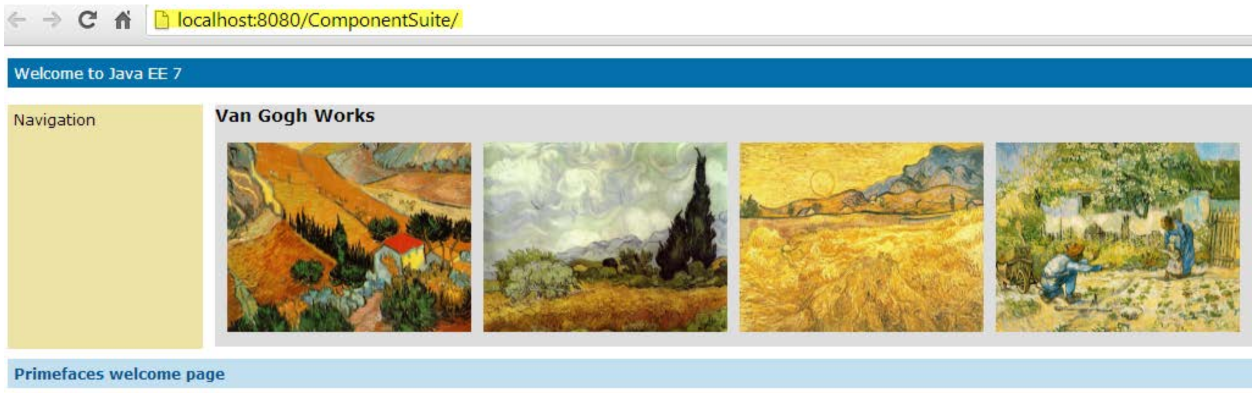
PrimeFaces is a lightweight open source

- Rich set of components (HtmlEdi
- Built-in Ajax based on standard JS
- Lightweight, one jar, zero-configu
- Native Ajax Push/Comet support.
- Mobile UI kit to create mobile web
- Skinning Framework with 30+ bui
- Extensive documentation with 45(
- Large, vibrant and active user con
- Developed with "passion" from ap

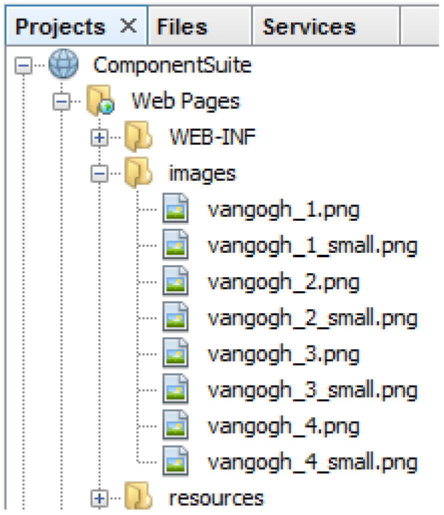
Notice the component Lightbox in the Overlay section as shown. Click on Lightbox to see an example implementation and the sample code provided. We will implement our own version of a lightbox by modifying index.xhtml.



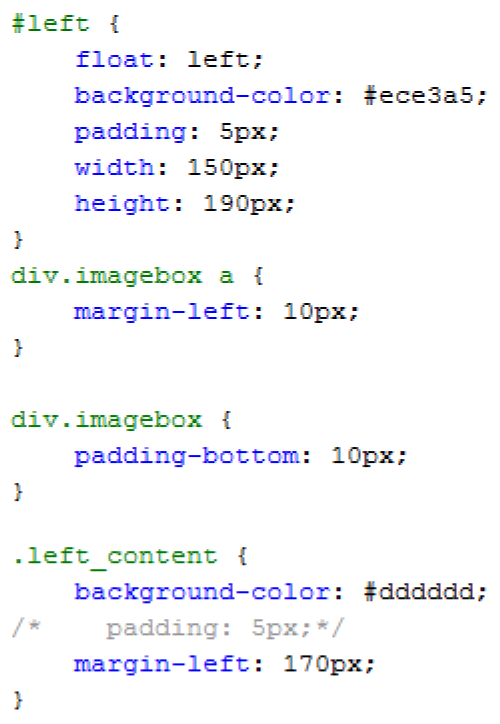
This is the resulting output of the index.xhtml page containing the lightbox which we will employ below.



To implement the lightbox, we will need to have four standard size images (600 x 450-500), four smaller versions of the images (200 x 150-160), apply minor modifications to a CSS file, and modify index.xhtml. A web/images folder was created for this exercise and 8 images were added to the folder.

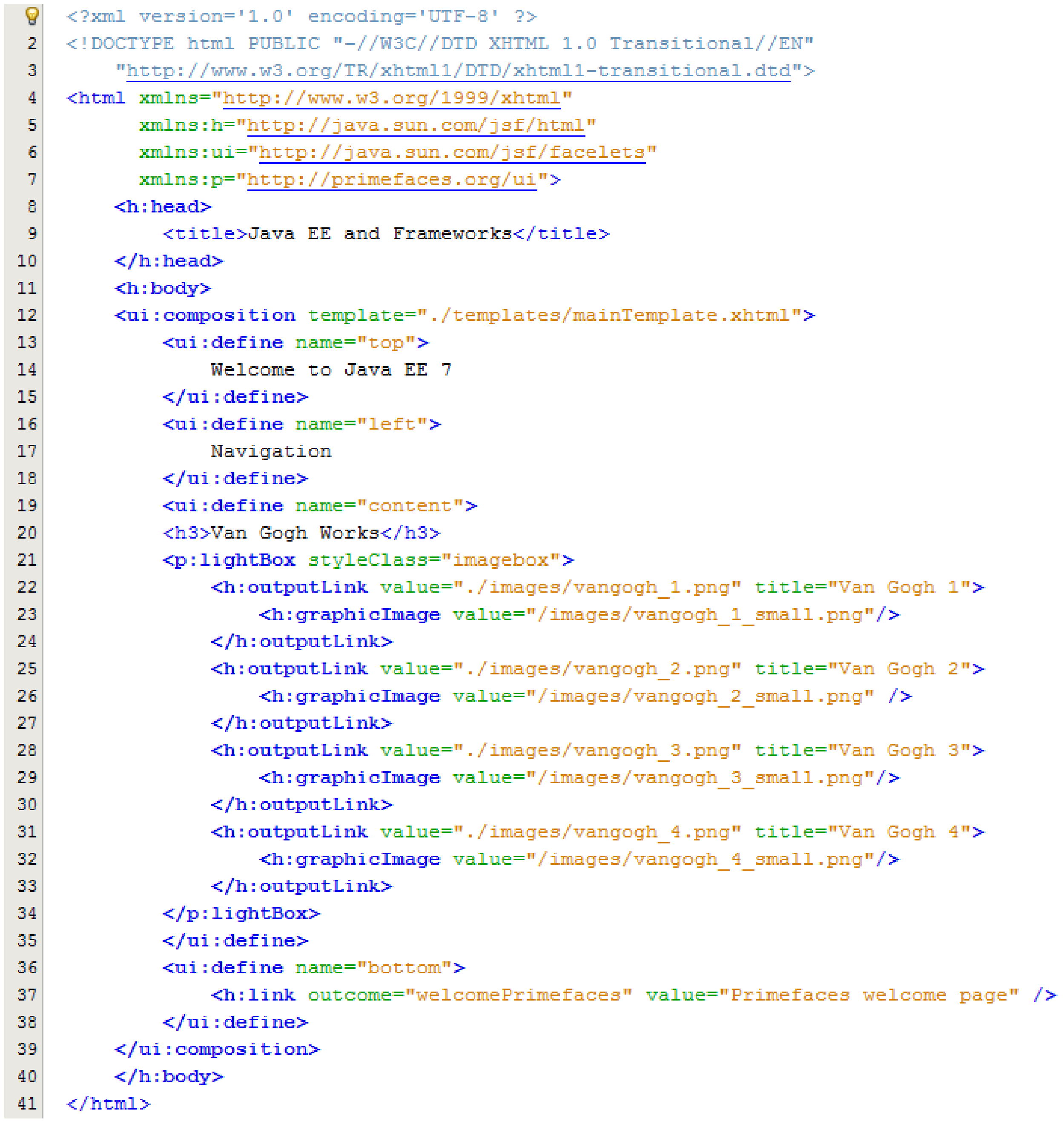


Change or add the selectors in cssLayout.css as shown.



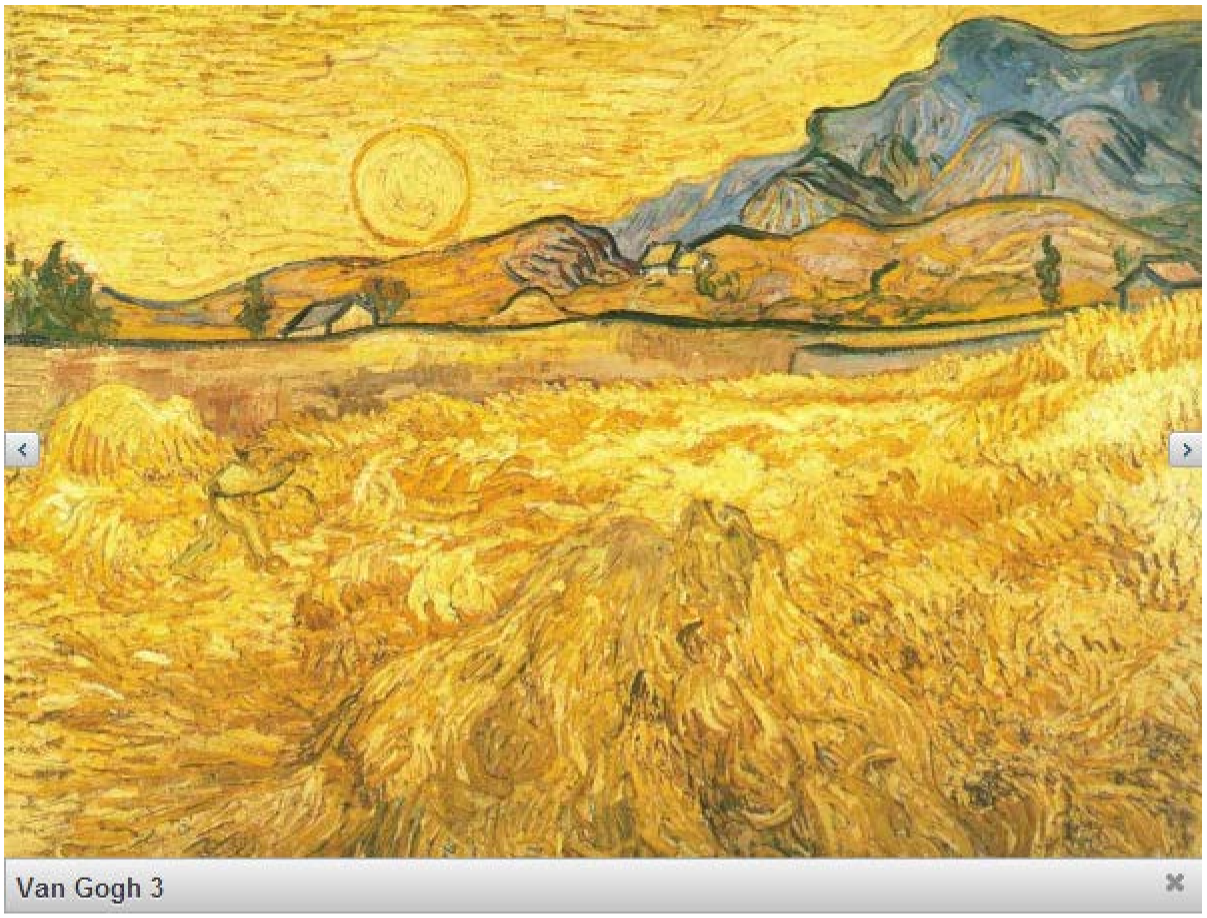
```
#left {
    float: left;
    background-color: #ece3a5;
    padding: 5px;
    width: 150px;
    height: 190px;
}
div.imagebox a {
    margin-left: 10px;
}

div.imagebox {
    padding-bottom: 10px;
}

.left_content {
    background-color: #dddddd;
/*    padding: 5px;*/
    margin-left: 170px;
}
```

Make the modifications to index.xhmtl as shown to implement the lightbox. As a web development exercise, produce your own sets of 4 standard size images and 4 smaller versions of the same images and place them in the web/images folder. Use an image editing program of your choice. A variety of free image manipulation programs are available for download if you do not currently have one. GIMP (GNU Image Manipulation Program) is popular and has many image editing capabilities.

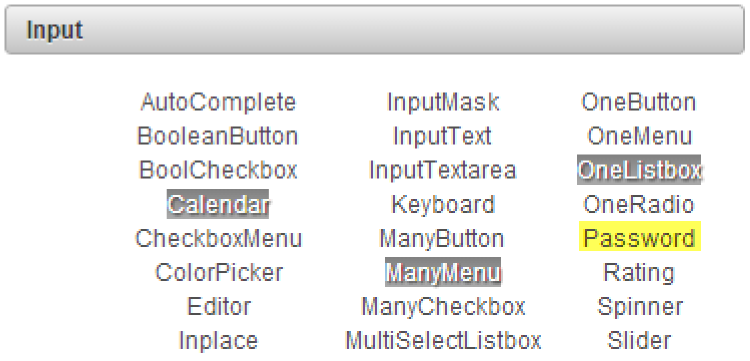


```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Java EE and Frameworks</title>
    </h:head>
    <h:body>
    <ui:composition template="./templates/mainTemplate.xhtml">
        <ui:define name="top">
            Welcome to Java EE 7
        </ui:define>
        <ui:define name="left">
            Navigation
        </ui:define>
        <ui:define name="content">
        <h3>Van Gogh Works</h3>
        <p:lightBox styleClass="imagebox">
            <h:outputLink value="./images/vangogh_1.png" title="Van Gogh 1">
                <h:graphicImage value="/images/vangogh_1_small.png"/>
            </h:outputLink>
            <h:outputLink value="./images/vangogh_2.png" title="Van Gogh 2">
                <h:graphicImage value="/images/vangogh_2_small.png" />
            </h:outputLink>
            <h:outputLink value="./images/vangogh_3.png" title="Van Gogh 3">
                <h:graphicImage value="/images/vangogh_3_small.png"/>
            </h:outputLink>
            <h:outputLink value="./images/vangogh_4.png" title="Van Gogh 4">
                <h:graphicImage value="/images/vangogh_4_small.png"/>
            </h:outputLink>
        </p:lightBox>
        </ui:define>
        <ui:define name="bottom">
            <h:link outcome="welcomePrimefaces" value="Primefaces welcome page" />
        </ui:define>
    </ui:composition>
    </h:body>
</html>
```

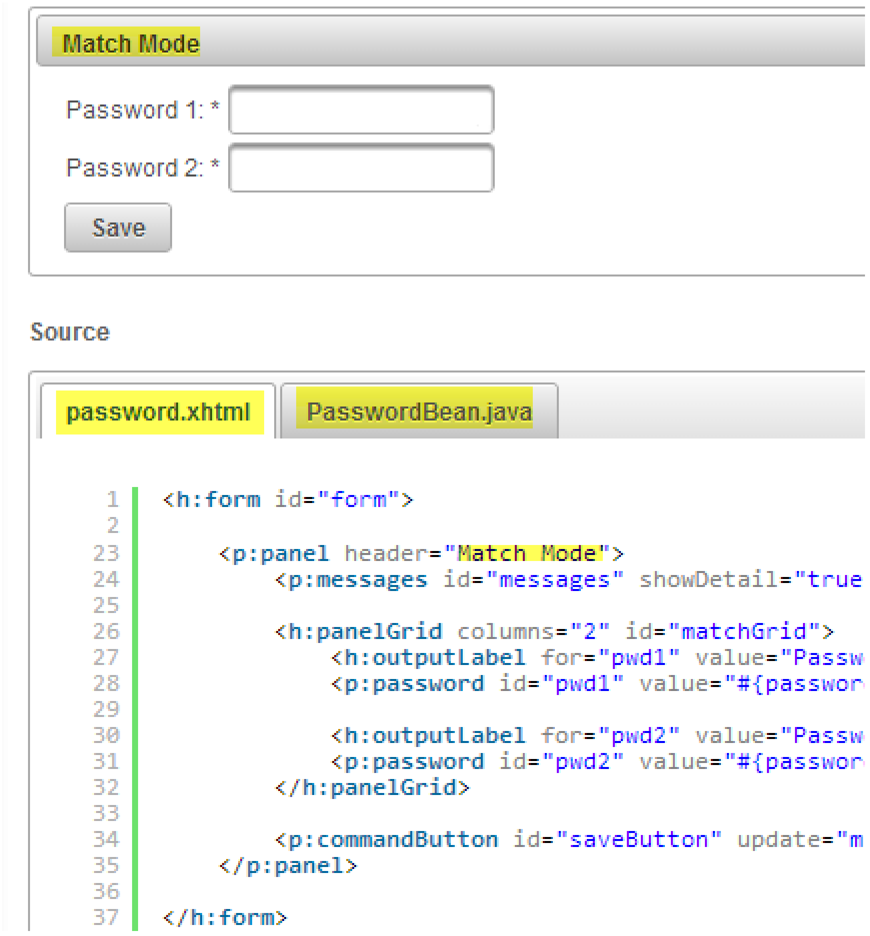
To see the lightbox in action, run the file by R-clicking the file and selection Run. After the page opens click on any of the images to see the larger images displayed in the lightbox. The output below is the result of clicking the third image. The arrows can be used to navigate to different images in the series. The lightbox can be closed by either clicking the x in the lower right corner or clicking off of the image.



Let's add one more PrimeFaces component to the page. Select the password control located in the input section of the PrimeFaces Demo page (accessed above).

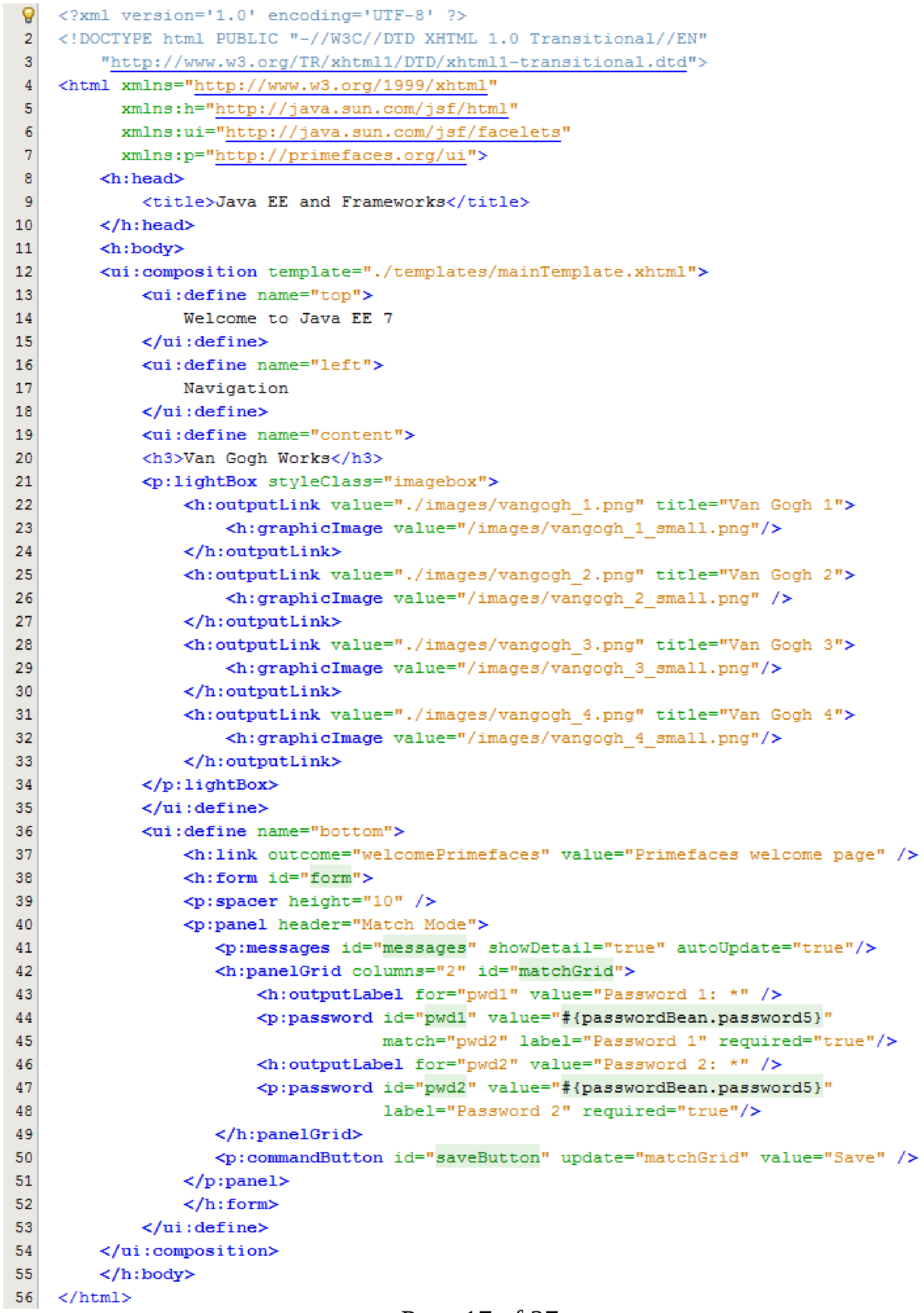


Our goal for this portion of the tutorial will be to produce password match mode controls like that shown in the example. Note that some of the code in password.xhtml and PasswordBean.java will be required.



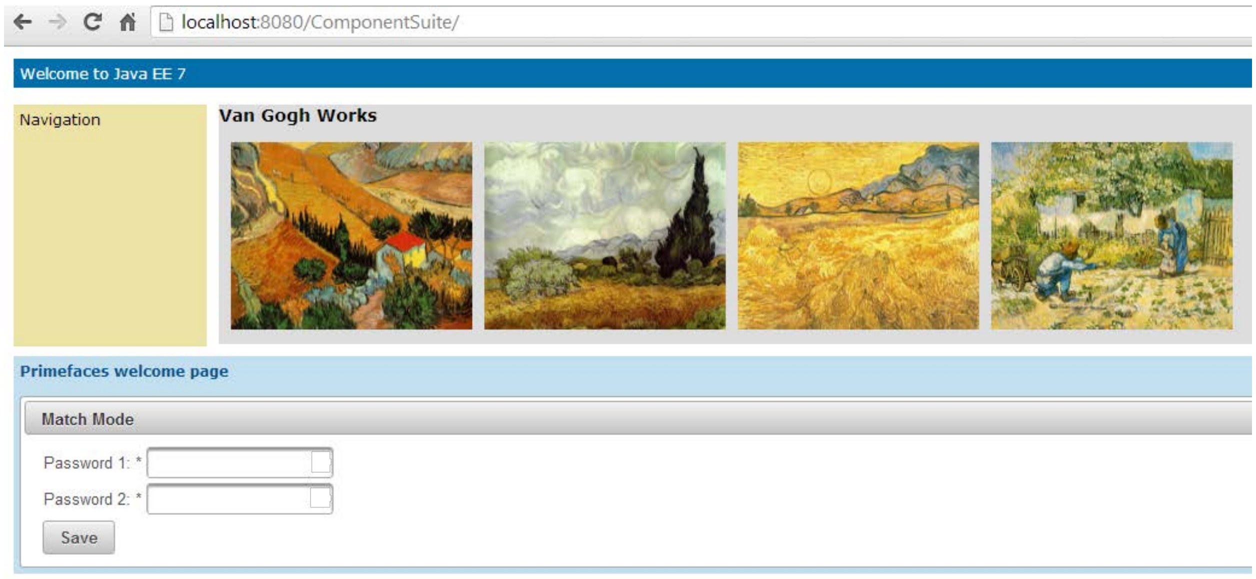
```
<h:form id="form">

    <p:panel header="Match Mode">
        <p:messages id="messages" showDetail="true

        <h:panelGrid columns="2" id="matchGrid">
            <h:outputLabel for="pwd1" value="Passw
            <p:password id="pwd1" value="#{passwor

            <h:outputLabel for="pwd2" value="Passw
            <p:password id="pwd2" value="#{passwor
        </h:panelGrid>

        <p:commandButton id="saveButton" update="m
    </p:panel>

</h:form>
```

Modify the index.xhtml file as shown.



```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<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:p="http://primefaces.org/ui">
    <h:head>
        <title>Java EE and Frameworks</title>
    </h:head>
    <h:body>
    <ui:composition template="./templates/mainTemplate.xhtml">
        <ui:define name="top">
            Welcome to Java EE 7
        </ui:define>
        <ui:define name="left">
            Navigation
        </ui:define>
        <ui:define name="content">
        <h3>Van Gogh Works</h3>
        <p:lightBox styleClass="imagebox">
            <h:outputLink value="./images/vangogh_1.png" title="Van Gogh 1">
                <h:graphicImage value="/images/vangogh_1_small.png"/>
            </h:outputLink>
            <h:outputLink value="./images/vangogh_2.png" title="Van Gogh 2">
                <h:graphicImage value="/images/vangogh_2_small.png" />
            </h:outputLink>
            <h:outputLink value="./images/vangogh_3.png" title="Van Gogh 3">
                <h:graphicImage value="/images/vangogh_3_small.png"/>
            </h:outputLink>
            <h:outputLink value="./images/vangogh_4.png" title="Van Gogh 4">
                <h:graphicImage value="/images/vangogh_4_small.png"/>
            </h:outputLink>
        </p:lightBox>
        </ui:define>
        <ui:define name="bottom">
            <h:link outcome="welcomePrimefaces" value="Primefaces welcome page" />
            <h:form id="form">
            <p:spacer height="10" />
            <p:panel header="Match Mode">
               <p:messages id="messages" showDetail="true" autoUpdate="true"/>
               <h:panelGrid columns="2" id="matchGrid">
                   <h:outputLabel for="pwd1" value="Password 1: *" />
                   <p:password id="pwd1" value="#{passwordBean.password5}"
                               match="pwd2" label="Password 1" required="true"/>
                   <h:outputLabel for="pwd2" value="Password 2: *" />
                   <p:password id="pwd2" value="#{passwordBean.password5}"
                               label="Password 2" required="true"/>
               </h:panelGrid>
               <p:commandButton id="saveButton" update="matchGrid" value="Save" />
            </p:panel>
            </h:form>
        </ui:define>
    </ui:composition>
    </h:body>
</html>
```

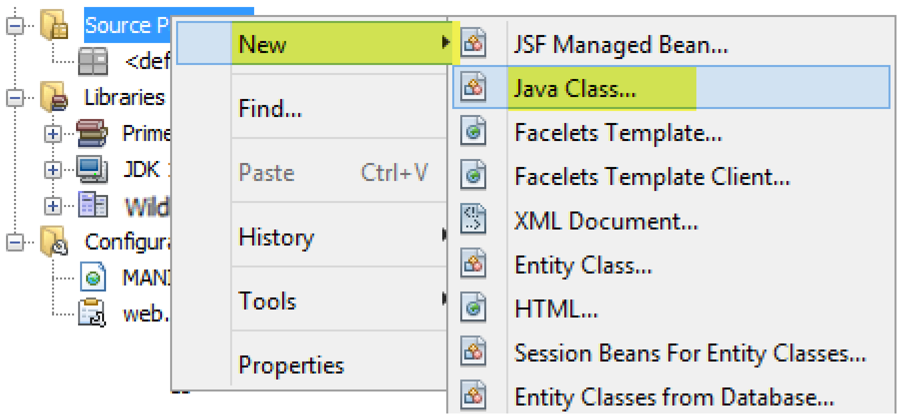
This is the output of index.xhtml which now includes both the lightbox and the password controls.



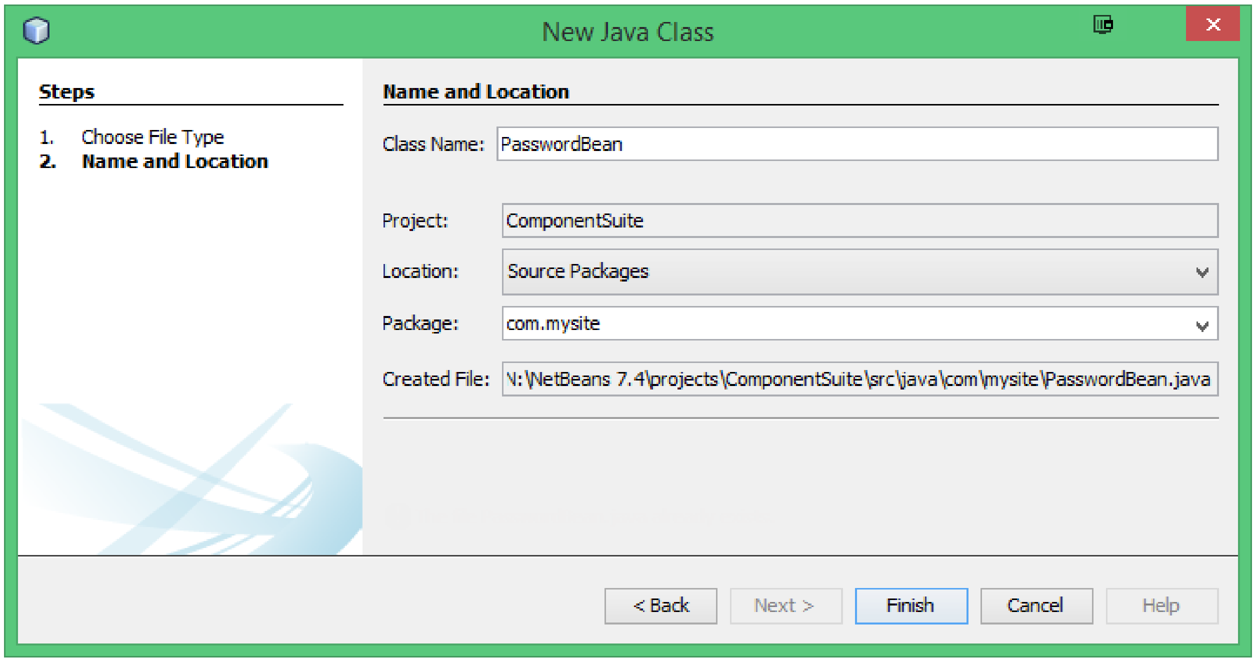
Using a Managed Bean

Now we will add the password bean to the project to enable the password matching functionality. The password bean will be managed by JSF (via the servlet container). Beans can be registered by xml files or by annotations. Annotations are used in this example due to the benefit of collocating registration with the bean.

R-click Source Packages | New | Java Class.



Enter the settings as shown and select Finish.



Name and Location

Class Name: PasswordBean

Project: ComponentSuite

Location: Source Packages

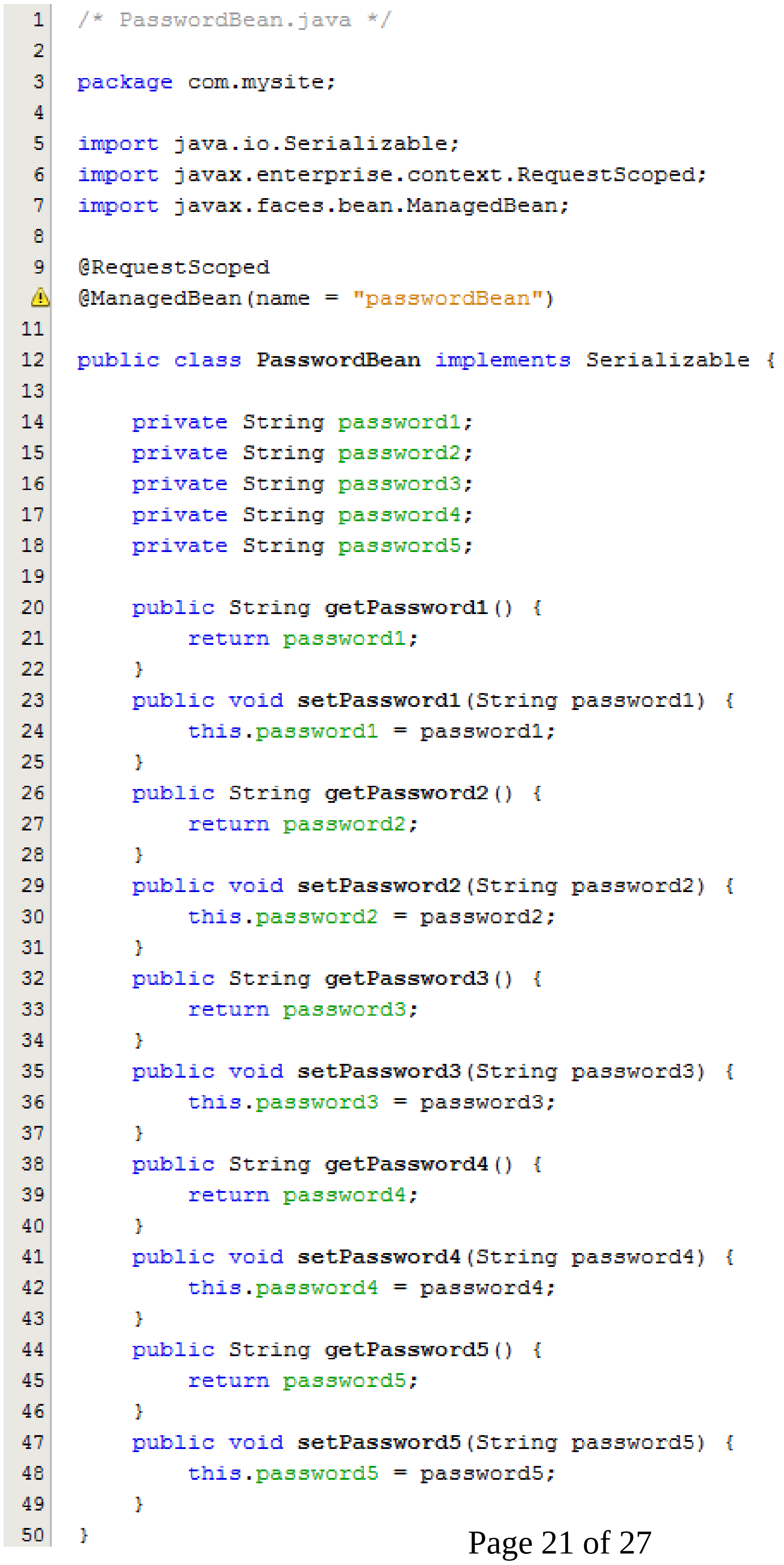
Package: com.mysite

Created File: \:\NetBeans 7.4\projects\ComponentSuite\src\java\com\mysite\PasswordBean.java

Replace the initial content of PasswordBean.java with that shown and save the changes. A Java bean is a class that meets the following 3 criteria: serializeable (state of the object can be saved), includes a public no-argument constructor, includes public getter and setter methods for private variables. The bean is considered managed if it is registered with and managed by the JSF framework which is accomplished by lines 6-10. Recall that registration is also possible with an xml file. The @ManagedBean annotation is shown on line 10 in PasswordBean.java.

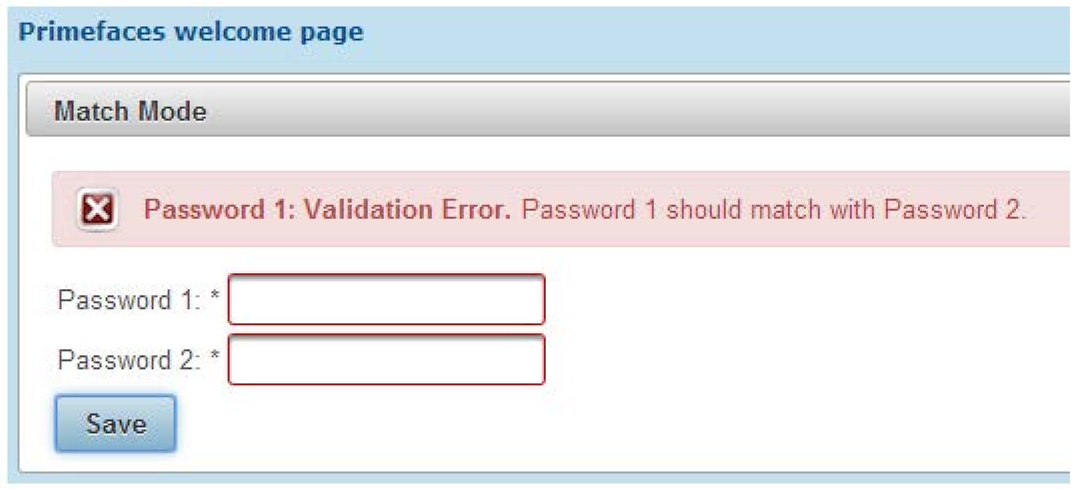
There are six scope options available for managed beans and are listed in the following table. Beans can be created early by setting the eager attribute to true. The default setting is lazy with which beans are created per the scope settings below. Notice on line 9 that the PasswordBean is set using the @RequestScoped annotation. To prevent state confusion, it is usually preferred to specify @RequestScoped for beans based on UI interactions.

Scope	Lifetime
@RequestScoped	Created with an HTTP request and destroyed when HTTP response is complete.
@NoneScoped	Created with an EL evaluation and destroyed when evaluation is complete.
@ViewScoped	Created with an HTTP request and destroyed when user posts-back to a different JSF view.
@SessionScoped	Created with an HTTP request and destroyed when the session is invalidated.
@ApplicationScoped	Created with an HTTP request and destroyed when the web application terminates.
@CustomScoped	Lifetime determined by customized settings in an Map.



```
/* PasswordBean.java */

package com.mysite;

import java.io.Serializable;
import javax.enterprise.context.RequestScoped;
import javax.faces.bean.ManagedBean;

@RequestScoped
@ManagedBean(name = "passwordBean")

public class PasswordBean implements Serializable {

    private String password1;
    private String password2;
    private String password3;
    private String password4;
    private String password5;

    public String getPassword1() {
        return password1;
    }
    public void setPassword1(String password1) {
        this.password1 = password1;
    }
    public String getPassword2() {
        return password2;
    }
    public void setPassword2(String password2) {
        this.password2 = password2;
    }
    public String getPassword3() {
        return password3;
    }
    public void setPassword3(String password3) {
        this.password3 = password3;
    }
    public String getPassword4() {
        return password4;
    }
    public void setPassword4(String password4) {
        this.password4 = password4;
    }
    public String getPassword5() {
        return password5;
    }
    public void setPassword5(String password5) {
        this.password5 = password5;
    }
}
```

Try the password control functionality (calling methods in PasswordBean.java) by inserting different values in the boxes and pressing Save which produces the warning as shown.


Primefaces welcome page

Match Mode

Password 1: Validation Error. Password 1 should match with Password 2.

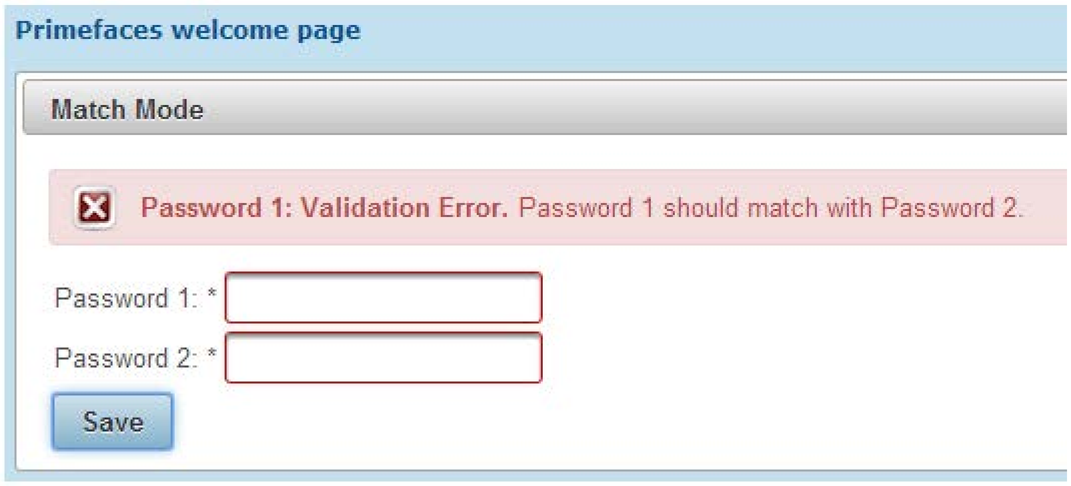
Using Context Dependency Injection

The advisory on line 10 in PasswordBean.java states that annotations from javax.faces.bean package will be deprecated in the next version of JSF and that CDI (Context Dependency Injection) annotations should be used instead. CDI allows the developer to determine when and where code will be inserted based on dependency of that code. CDI is a more modern replacement for managed beans and should be used for new development. There are more examples of CDI in upcoming tutorials. The code is updated in the next listing.

To address the advisory on line 10, make the following changes. Only lines 6-10 require updates. The advisory no longer appears and the file now uses the modern CDI approach.

```java
/* PasswordBean.java */

package com.mysite;

import java.io.Serializable;
import javax.enterprise.context.Dependent;
import javax.inject.Named;

@Dependent
@Named

public class PasswordBean implements Serializable {
```

Verify that the password control functionality continues to operate when using CDI by inserting different values in the boxes and pressing Save. The same validation error is shown which confirms operation of the control.


Primefaces welcome page

Match Mode

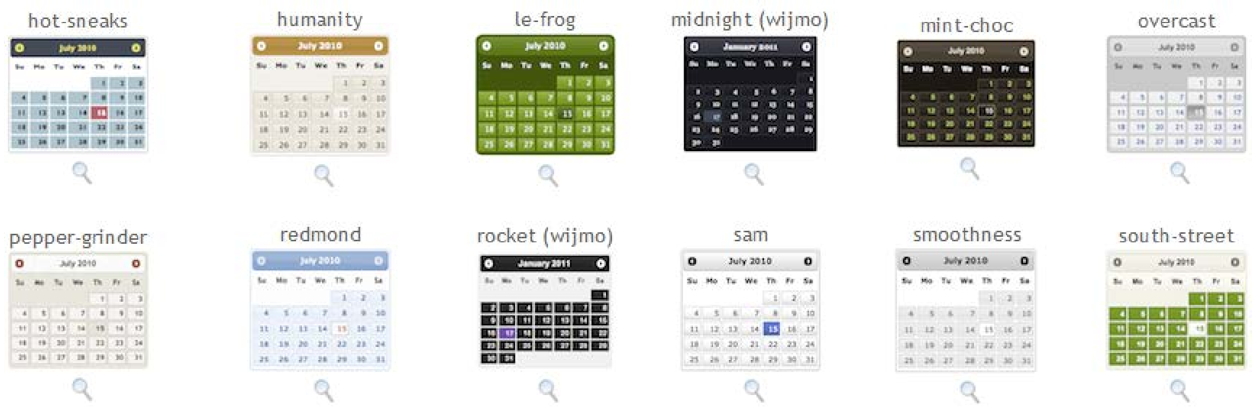
Password 1: Validation Error. Password 1 should match with Password 2.

Password 1: *

Password 2: *

Applying a PrimeFaces Theme

In addition to the many advanced user interface controls provided by PrimeFaces, there are also pre-built themes that can be applied for a variety of colors and styles. Navigate to http://primefaces.org/themes to review the community (free) themes. A few are shown below.

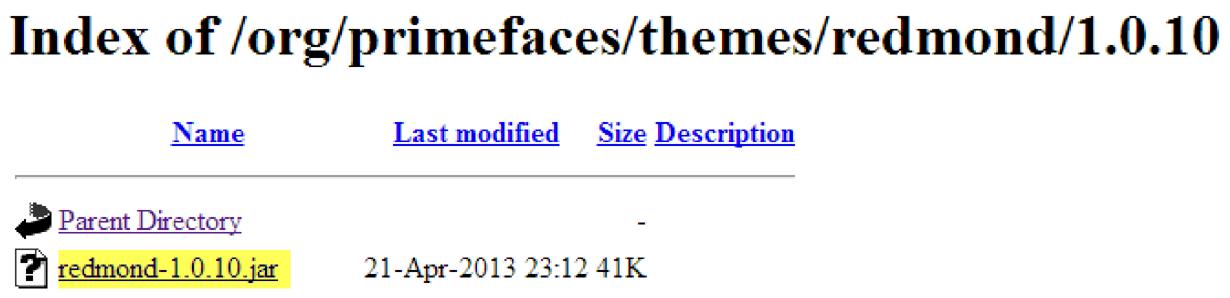


Adding a theme to a project involves just three steps: modify web.xml, download the theme .jar file, add the .jar file to the project libraries. The addition to web.xml is shown below for the redmond theme. Set the param-value to correspond to the theme which you want to apply.

```
<context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>redmond</param-value>
</context-param>
```

Download the theme of your choice from the PrimeFaces repository (redmond-1.0.10.jar file was used for this tutorial but any theme is fine):

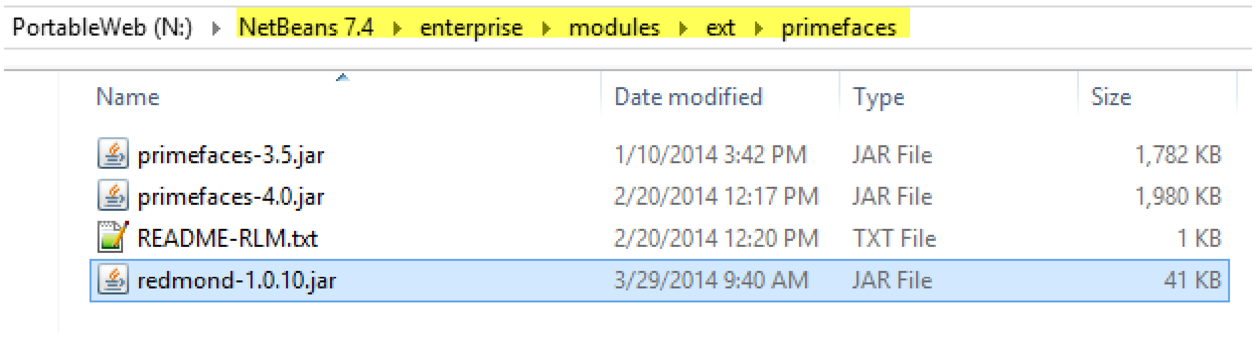
http://repository.primefaces.org/org/primefaces/themes/



Index of /org/primefaces/themes/redmond/1.0.10

Name	Last modified	Size	Description
Parent Directory		-	
redmond-1.0.10.jar	21-Apr-2013 23:12	41K	

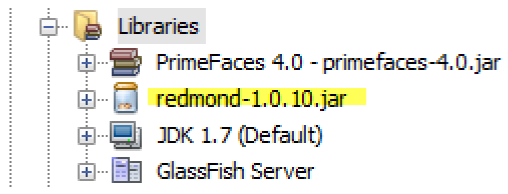
Add that .jar file to the same location where you have the primefaces-4.0.jar installed. On my machine it is NetBeans 7.4/enterprise/modules/ext/primefaces.



PortableWeb (N:) ▸ NetBeans 7.4 ▸ enterprise ▸ modules ▸ ext ▸ primefaces

Name	Date modified	Type	Size
primefaces-3.5.jar	1/10/2014 3:42 PM	JAR File	1,782 KB
primefaces-4.0.jar	2/20/2014 12:17 PM	JAR File	1,980 KB
README-RLM.txt	2/20/2014 12:20 PM	TXT File	1 KB
redmond-1.0.10.jar	3/29/2014 9:40 AM	JAR File	41 KB

In NetBeans, right-click Libraries in the ComponentSuite project and select Add JAR/Folder… Navigate to and select redmond-1.0.10.jar.

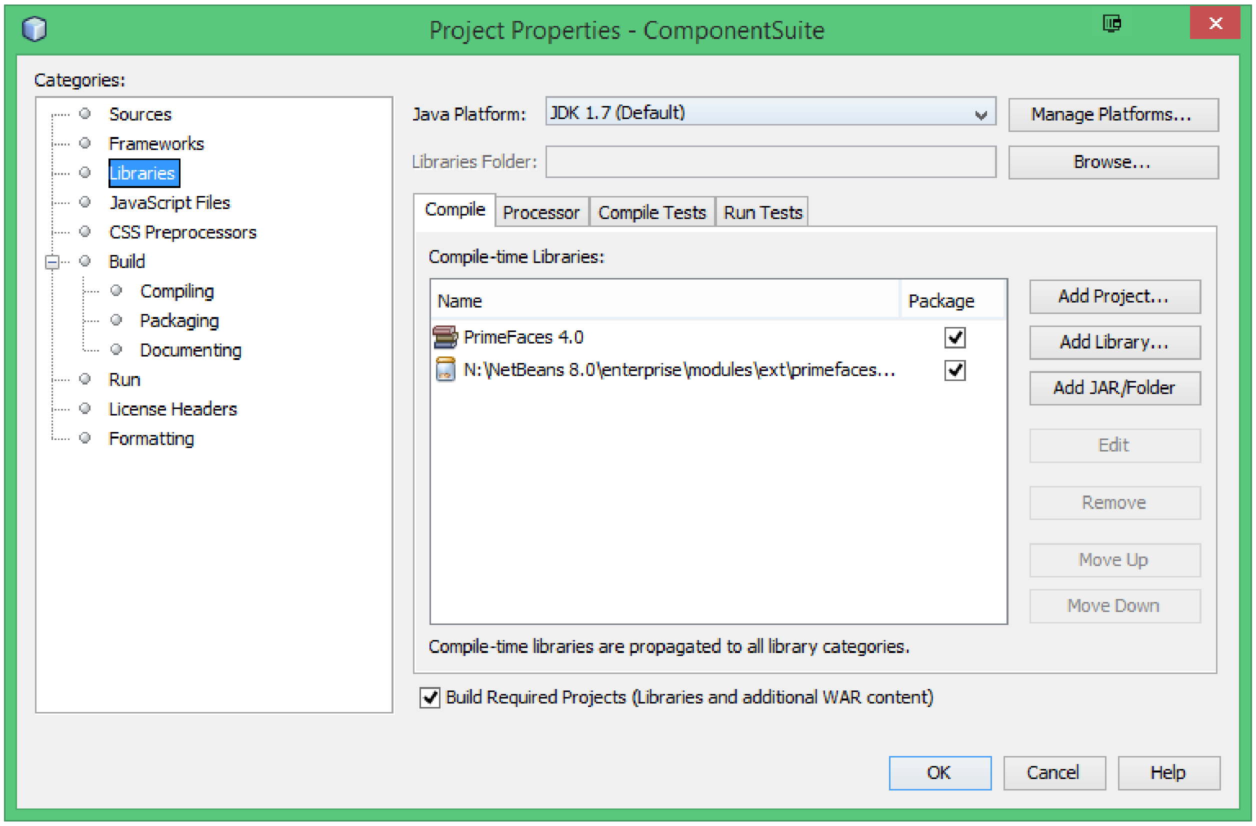


```
Libraries
  PrimeFaces 4.0 - primefaces-4.0.jar
  redmond-1.0.10.jar
  JDK 1.7 (Default)
  GlassFish Server
```

You can also add or view libraries and .jar files associated with the project by R-clicking the project name | Properties to display the Project Properties dialog as shown. Select the Libraries category to see the libraries and .jar files. Notice the library icon for PrimeFaces and the .jar file icon for the redmond file. See a few other NetBeans icons at this link:

http://docs.oracle.com/cd/E50453_01/doc.80/e50452/work_java_code.htm#r1c1-t5

Also note that the dialog below is shown after the project was upgraded to NetBeans v8.0.


Project Properties - ComponentSuite

Categories:

- Sources
- Frameworks
- Libraries
- JavaScript Files
- CSS Preprocessors
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- License Headers
- Formatting

Java Platform: JDK 1.7 (Default)

Libraries Folder:

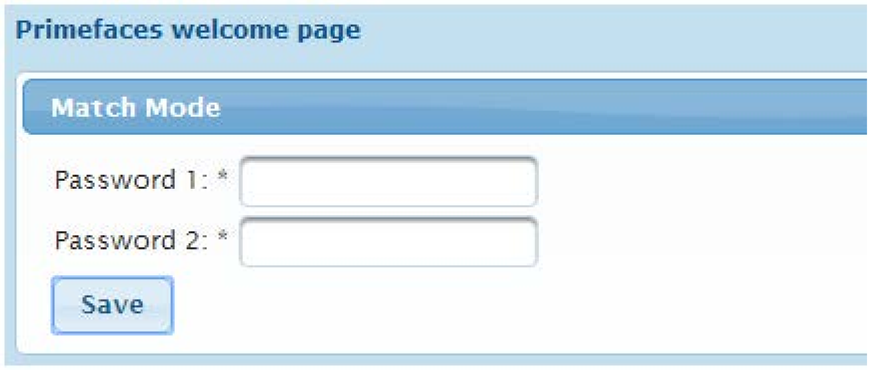
Compile-time Libraries:

Name	Package
PrimeFaces 4.0	✓
N:\NetBeans 8.0\enterprise\modules\ext\primefaces...	✓

Compile-time libraries are propagated to all library categories.

✓ Build Required Projects (Libraries and additional WAR content)

Run the project and notice that the redmond theme is applied to the password control (Match Mode is blue).



In this tutorial we started by creating a copy of the TemplateDemo project and naming the new project ComponentSuite. Then the PrimeFaces component suite was added to the project. The lightbox control from PrimeFaces was implemented in index.xhtml. Then the password matching control was included which supplies a warning if passwords do not agree. Finally, we applied a PrimeFaces theme to present a new user perspective.