

Background:

The JavaServer Faces framework is the standard Java API for building user interface components in web applications. You can think of the JSF framework as a toolbox full of ready-to-use components that you can quickly and easily reuse in your web application. These components can be simple, like text input fields that get and store user data, to more complex components, like a formatted date field with a pop-up calendar. You embed the components into JSP pages and use the framework to handle navigation between different JSP pages.

Creating the Project:

1. Choose File > New Project to open the New Project wizard. Under Categories select Web; under Projects select Web Application. Click Next.
2. Name the project jBirthInfo, specify a location for the project on your computer, then click Next.
3. In the third step of the wizard, Server and Settings, specify the server and Java version to be used with the project (or accept default settings). Click Next.
4. In the Frameworks step, select JavaServer Faces and click Finish

The IDE creates a project template for the entire application, and opens an empty JSP page (welcomeJSF.jsp) in the Source Editor. This file is the default page in the web.xml deployment descriptor. Notice that the JSF libraries, such as jsf-impl.jar, are included in GlassFish and are added to the project's classpath. Expand Configuration Files and notice that the IDE has created a faces-config.xml file, which controls the behavior of JSF components in the web application. The IDE has also registered the Faces servlet in the web.xml deployment descriptor. The Faces servlet handles navigation between JSP pages that are controlled by the JSF framework.

Creating the JSP Pages:

Create a new JSP page called `greeting.jsp` that welcomes the user and collects his or her information.

Then create a `success.jsp` page that congratulates the user in response to receiving data from the form.

Creating the Greeting Page:

1. In the Projects window, right-click the project node and choose New > JSP. Name the file greeting. Make sure the JSP File (Standard Syntax) option is selected and click Finish. The IDE creates the new JSP file and opens it in the Source Editor. Also, note that the file is added to the Web Pages node in the Projects window.
2. In the Source Editor, declare the JSF tag libraries in `greeting.jsp`. Do this by adding the following code to the top of the file:

```
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core" %>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>
```

Note that you can make use of the IDE's built-in support for code completion. As you type, press Ctrl-Space to list suggestions based on the context. In this manner, code completion can help you add tag names and attributes, such as the URIs of the tag libraries.

3. Change the contents of both the title and h2 tags to Welcome to jBirthInfo.
4. Now add a JSF form to the file. In the Palette (Shift-Ctrl-8; Shift-␣-8 on Mac), expand the JSF category. You can drag-and-drop items from the Palette directly into the Source Editor. Click the JSF Form button, drag the item to a point below the h2 tags, and release the mouse button. In the dialog box that displays, leave Empty Form selected and click OK. The IDE fills in the following code (shown in **bold**):

```
<h2>Welcome to jBirthInfo</h2>
  <f:view>
    <h:form>
    </h:form>
  </f:view>
```

5. You can use `inputText` components to get user input and a `commandButton` component to submit the form. In the Source Editor, change the contents of the `<h:form>` tags to the following (changes in **bold**):

```
<f:view>
  <h:form>
    <p>Enter your name: <h:inputText value="name" /></p>
    <p>Enter your birthday: <h:inputText value="birthday" /></p>
    <h:commandButton value="Submit" action="submit" />
  </h:form>
</f:view>
```

To format your code, right-click in the Source Editor and choose Format (Alt-Shift-F; Ctrl-Shift-F on Mac).

Creating the Success Page:

Now create a JSP page that says 'Congratulations'.

1. Create a new JSP file as described above. Name the file `success`.
2. Change the contents of the file to the following:

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Congratulations</title>
</head>
<body>
<h2>Congratulations</h2>
<p>You've successfully registered with jBirthInfo.</p>
</body>
```

Note that the file so far only contains plain HTML, so there is no need to declare JSF tag libraries yet.

Setting the Page Navigation:

Page navigation in the JSF framework is controlled by the `faces-config.xml`. For each JSP page in the project, you set up a navigation rule in `faces-config.xml` which contains one or more navigation cases. Here, you can simply map the submit action from the `commandButton` to `success.jsp`, so that the user sees a success message no matter what is entered in the fields.

1. In the Projects window, double-click `faces-config.xml` to open the file in the Source Editor. In the toolbar above the file, click XML to display the file in plain XML.
2. Right-click anywhere in the file and choose Java ServerFaces > Add Navigation Rule. Type `/greeting.jsp` in the Rule from View field and optionally enter a description of the rule.



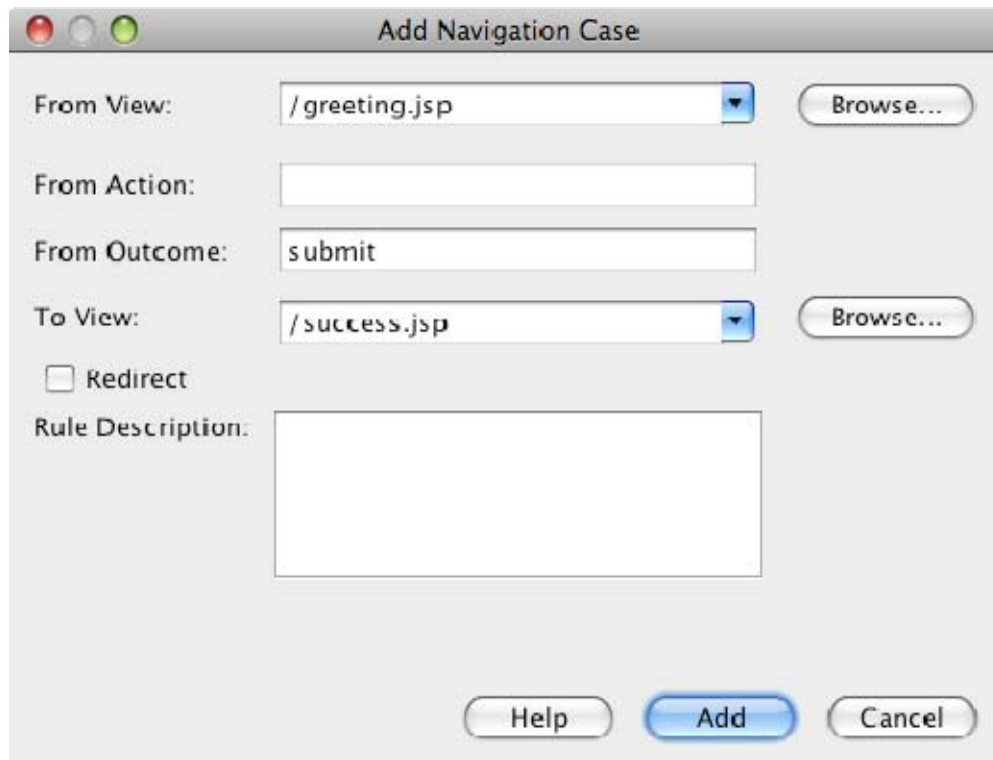
Click Add. The following code is entered into `faces-config.xml`:

```
<navigation-rule>
```

```
<description>
    handle user input
</description>
<from-view-id>/greeting.jsp</from-view-id>
</navigation-rule>
```

3. Again right-click inside faces-config.xml and choose Java ServerFaces > Add Navigation Case. In the dialog that displays, set the following:

- From View: /greeting.jsp
- From Outcome: submit
- To View: /success.jsp



Click Add. The IDE enters the following code into faces-config.xml (changes are in **bold**):

```
<navigation-rule>
    <description>
        handle user input
    </description>
    <from-view-id>/greeting.jsp</from-view-id>
```

```
<navigation-case>
    <from-outcome>submit</from-outcome>
    <to-view-id>/success.jsp</to-view-id>
</navigation-case>
</navigation-rule>
```

Configuring and Running the Application:

Set the IDE to display `greeting.jsp` when it runs the application and, finally, test the application.

1. In the Projects window, right-click the project node and choose Properties.
2. Click the Run node and type `/faces/greeting.jsp` in the Relative URL field. This allows you to specify the entry point for the application in the IDE. Click OK.
3. Add a simple stylesheet to the project. One easy way to do this is by saving this sample stylesheet to your computer. Copy the file, then in the IDE, select the Web Pages node in the Projects window and press Ctrl-V. The file is added to your project.
4. Link the stylesheet to your JSP pages by adding the following reference between the `<head>` tags of both `greeting.jsp` and `success.jsp`:

```
<link rel="stylesheet" type="text/css" href="stylesheet.css">
```

5. Right-click the project node and choose Run. The IDE builds the project, starts the application server, deploys the application, and shows the `greeting.jsp` page in the default external browser:

Note: Because you have changed the entry point for the application to `greeting.jsp`, you can now delete `welcomeJSF.jsp`, which was generated by default when the project was created. The page is not required for this tutorial, nor follow-up tutorials.

Adding a JSF Managed Bean:

In the previous section you created a simple web application with JSF components. However, the web application does not really do anything - yet. In order to add rich functionality to JSF web applications, you can

associate UI components with backing beans. A backing bean, also called a *JSF managed bean*, is a regular JavaBeans component whose bean properties and methods are available to the JSF components.

In this section, you create a `UserBean` managed bean that will expose two bean properties: `name` and `birthday`.

1. In the Projects window, right-click the project node and choose `New > Other` (Ctrl-N; ⌘-N on Mac). Under the JavaServer Faces category, select the JSF Managed Bean template and click Next.
2. Name the bean `UserBean` and create a new package named `birthInfo.user` to put it in. Leave the rest of the settings at their default values and click Finish.

The IDE opens `UserBean.java` in the Source Editor and adds the following bean declaration to `faces-config.xml`:

```
<managed-bean>
    <managed-bean-name>UserBean</managed-bean-name>
    <managed-bean-class>birthInfo.user.UserBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

3. Add the following field declarations (shown in **bold**) to `UserBean.java`:

```
public class UserBean {
    String name;
    String birthday;
}
```

4. Generate getters and setters for the fields: right-click anywhere in the file and choose `Refactor > Encapsulate Fields`. In the dialog box that displays, select getter and setter options for both `name` and `birthday`, then click Refactor.



The IDE switches the access level for the fields to **private** and creates getter and setter methods directly in the file.

5. In `greeting.jsp`, make the following changes (shown in **bold**).

```
<f:view>
  <h:form>
    <p>Enter your name: <h:inputText value="#{UserBean.name}" /></p>
    <p>Enter your birthday: <h:inputText value="#{UserBean.birthday}"
  /></p>
    <h:commandButton value="Submit" action="submit" />
  </h:form>
</f:view>
```

As you enter changes, make use of the IDE's code completion support for `UserBeans.java` and its properties by pressing `Ctrl-Space` and choosing available options.

6. Add the JSF `taglib` declarations to `success.jsp`. You can copy and paste them from `greeting.jsp`.
7. Add an empty JSF form to `success.jsp` by clicking the JSF Form button in the Palette (`Shift-Ctrl-8`;) and dragging and it to a point below the `<h2>` tags in the Source Editor.
8. Make the following changes to `success.jsp` (changes in **bold**):

```
<h2>Congratulations</h2>
<f:view>
  <h:form>
    <p>You've successfully registered with jBirthInfo.</p>
    <p>Your name is <h:outputText value="#{UserBean.name}" /></p>
```

```
        <p>Your birthday is <h:outputText value="#{UserBean.birthday}"
/></p>
    </h:form>
</f:view>
```

9. In the Projects window, right-click the project node and choose Run. The same `greeting.jsp` page displays in a browser when the application is redeployed and run. When you enter values and click Submit, `success.jsp` now displays the values you entered in `greeting.jsp`