How To... Create Your First JSF Application

Applicable Releases:
SAP NetWeaver Composition Environment 7.1

Topic Area:
User Productivity
Development and Composition

Capability:
User Interface Technology
Java

Version 1.0
October 2008
These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components. Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.
**Document History**

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
### Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Example Text</em></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td>&lt;Example text&gt;</td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

### Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>📋</td>
<td>Note or Important</td>
</tr>
<tr>
<td>🎉</td>
<td>Example</td>
</tr>
<tr>
<td>🏆</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Business Scenario .................................................................................................................. 1  
2. Background Information ....................................................................................................... 1  
3. Prerequisites .......................................................................................................................... 1  
4. Step-by-Step Procedure ...................................................................................................... 2  
   4.1 Tutorial Setup ............................................................................................................... 2  
   4.2 Create a Simple Java Class ......................................................................................... 2  
   4.3 Create JSP Pages ......................................................................................................... 6  
   4.4 Build, Deploy and Run your application .......................................................................... 15
1. Business Scenario

So far, you have created a simple JSF application that can only display a text. The following guide will step you thru the creation of a simple Java EE 5 application comprising a simple java class and two JSF views. The user interface for this Web application will consist of two views and will allow you to navigate between them. We will define a static navigation, that is, clicking a particular button always selects a fixed JSF page for rendering the response.

2. Background Information

JavaServer Faces (JSF) is a relatively new user interface technology, having been added to the Java standard with Java EE 5. It is supported in the AS Java 7.1 since it supports Java EE 5. In addition the SAP NetWeaver Developer Studio (NWDS) is based on Eclipse 3.3 which in turn comes with development plug-ins for developing JSF applications. Like most other UI technologies JSF is based on the Model View Controller (MVC) architecture. To learn more about the JSF specification you can visit Sun's java website for JSF.

You will now be familiarized with additional main concepts of JSF application development by creating a navigation rule linking the JSF views in the application. You will learn about the following aspects of the JSF programming model, for example:

- Defining a java class that manages the user data
- Promoting the latter java class as a JSF Managed Bean to expose the application model to the JSF views within given scope
- Using Navigation Rule to configure a static transition from one JSF View to another

3. Prerequisites

The following is a list of all you need for developing JSF applications.

- AS Java 7.1 (CE 7.1 or NW 7.1)
- NWDS 7.1 (SP3 or higher with latest patch level).

⚠️ Note

While this tutorial is geared towards the SAP AS Java (the build/deploy steps of the guide), it wouldn’t be hard to replace the build/deploy portions with similar steps for any other Java EE 5 platform

Knowledge

- You have a basic knowledge of Java Enterprise Edition
- You have acquired some basic experience with JSF applications, for example by working through the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).
4. Step-by-Step Procedure

In the following sections, you will create a Web Module Development component containing the application business logic and an Enterprise Application needed to deploy the web module. You will get to know how to configure the navigation of the web application and how the JSF pages refer to the object properties.

This Web application will consist of two views. In the first view, the user should be able to enter his or her first and last name in input texts and navigate to the next view using a Submit button. The information will be added dynamically to an output text and displayed in the welcoming text in the second view. In the second view the user should be able to navigate to the first view using the Back button.

4.1 Tutorial Setup

1. Create a Web Module Development Component named navigationjsf/web as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).

2. Create an Enterprise Application Development Component named navigationjsf/ear as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).

4.2 Create a Simple Java Class

The java class will manage the user data. For the purpose of this exercise we are going to define two properties: firstName and lastName.

1. From the context menu of the Java Resources: source folder in the Web Module project select New → Class
2. Enter Person in the Name field and com.sap.demo.jsf.nav.beans in the Package field and click the “Finish” button. The java class will be created.

3. Declare the firstName and lastName attributes by entering the following code in its body:

```java
private String firstName;
private String lastName;
```

4. Right click on the Person.java class in the Project Explorer view and select Source → Generate Getters and Setters in the context menu.

5. Select firstName and lastName in the Generate Getters and Setters window and click the OK button. The methods will be generated.
6. The Person class source should look like the following code

```java
public class Person {
    private String firstName;
    private String lastName;

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

7. To register the Java class in the JSF application, drill into the Web Module project, in the WebContent → WEB-INF folder and open the faces-config.xml file.

   **Important**

   The Java class instance is configured in the faces-config.xml file. This file enables the JSF framework for instantiating the objects and storing them in the appropriate scope, you must configure the objects in the application configuration resource file faces-config.xml using the managed-bean XML element.

8. Go to the ManagedBean tab and click the Add button.
9. Select the *Using an existing Java class* option and enter `com.sap.demo.jsf.nav.beans.Person` in the *Qualified class name* field or click the *Browse* button to explore your project. Click the *Next* button.

10. Enter `person` in the *Name* field to reference the Person java class and select *session* in the *Scope* field.

    **Important**

    The scope allows beans and other objects to be available in different components of a web application. Managed beans registered with:

    - **None** scope are not instantiated nor stored in the request, session, or application objects
    - **Request** scope will be instantiated and stay available throughout a single HTTP request.
    - **Session** scope will be stored on the HTTP session.
    - **Application** scope retain their values throughout the lifetime of the application and are available to all users.
11. Click the Finish button to register the java class in the JSF application. The following XML code is added automatically between the `<faces-config ... >` `<faces-config>` tags

```xml
<managed-bean>
  <managed-bean-name>person</managed-bean-name>
  <managed-bean-class>
    com.sap.demo.jsf.nav.beans.Person
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

12. Save the changes

### 4.3 Create JSP Pages

1. Drill into the Web Module project and right click on the WebContent folder and in the context menu select New → JSP.
2. Enter the file name `index.jsp` and click the **Finish** button. The JSP page will be created. The `index.jsp` page should be opened in the **Web Page Editor**.

3. Click the **JSF HTML** toolset in the Palette, this will show all the UI elements available within it.

4. Drag and drop a **Form** element (found in the **JSF HTML** elements) to the **Web Page Editor**.

5. Place a **Panel Grid** element between the `<h:form>…</h:form>` tag as shown in the image below.

6. Take a look at the tags that were inserted into the JSP page. Put the cursor on the first **Output Text** UI element and then select the **Properties** view in the bottom window pane. Enter the text **First Name** in the **Value** property.
7. Click the *Edit* button next to the Style property. Change the settings as shown below or any other way you like, and then click the *OK* button.

8. Put the cursor on the second *Output Text* UI element, and replace it with a *Text Input* UI element. In the Properties view in the bottom window pane, enter the text `#{person.firstName}` in the Value property.

**Important**

The Input Text *Value* property is bound to properties of the java class with name `person`. When this page is displayed, in this case the framework calls the `getFirstName` method to obtain the current property value. When the page is submitted, the framework invokes the `setFirstName` method to set the value that the user entered.
9. Repeat steps 6-8 to enter the Last Name UI objects

10. Drag and drop a CommandButton element to the Web Page Editor. Enter submit in the Value and Action properties.

   **Note**
   
   The Action property holds a value for passing values from the page to the faces-config.xml file where the navigation from one page to another page is decided.

11. Save the changes to the index.jsp page that you made.

12. Create the second view welcome.jsp. Right click on the WebContent folder and in the context menu and select New → JSP.

13. Place an Output Text UI element between the <h:view>…</h:view> tags and select the Properties view in the bottom window pane. Enter the text Welcome in the Value property

14. Click the Edit button next to the Style property. Change the settings as shown below or any other way you like, and then click the OK button
15. Place another Output Text UI element and select the Properties view in the bottom window pane. Enter the text `#{person.firstName} #{person.lastName}` in the Value property.

16. Click the Edit button next to the Style property. Change the settings as shown below or any other way you like, and then click the OK button.
17. To create an space between the two Output Text UI element you can add 5px of left padding

18. Place another Output Text UI element and select the Properties view in the bottom window pane. Enter the text to JSF! In the Value property. Change the Style property as well.

19. Drag and drop a CommandButton element to the Web Page Editor. Enter back in the Value and Action properties.
20. Save the changes to the welcome.jsp page you just made.

21. To complete the JSF application, we need to specify the navigation rules, drill into the Web Module project, in the WebContent → WEB-INF folder and open the faces-config.xml file.

   **Note**

   A navigation rule tells the JSF implementation which page to send back to the browser after a form has been submitted. When the user clicks the Submit button, the application navigates from the index.jsp page to welcome.jsp. You specify this navigation rule in the faces-config.xml file.

22. Go to the Navigation Rule tab.

23. Click the Page element in the Palette and click in the faces-config.xml file editor.
24. A new window will pop up to select the JSP file you want to add in your navigation flow.

25. Select the index.jsp page and click the OK button. The index.jsp page will be added in your navigation flow.

26. Follow the steps 23-25 to add the welcome.jsp page to the navigation flow.

27. Click the Link element in the Palette, first select the index page and then select the welcome page. An arrow between the two pages should appear.

28. Select the arrow and then select the Properties view in the bottom window pane. Enter the text submit in the From Outcome property or click the button to open the Outcome Selection window.

Note

The From Outcome value matches the action attribute of the command button of the index.jsp page.
29. Click again the Link element in the Palette, but now first select the welcome page and then select the index page. An arrow between the two pages should appear.

30. Select the arrow and then select the Properties view in the bottom window pane. Enter the text **back** in the From Outcome property or click the button to open the Outcome Selection window.

31. Select the Source tab. The following XML code should be added automatically between the `<faces-config ...>` tags:

```xml
<navigation-rule>
    <display-name>index</display-name>
    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
        <from-outcome>submit</from-outcome>
        <to-view-id>/welcome.jsp</to-view-id>
    </navigation-case>
</navigation-rule>

<navigation-rule>
    <display-name>welcome</display-name>
    <from-view-id>/welcome.jsp</from-view-id>
    <navigation-case>
        <from-outcome>back</from-outcome>
        <to-view-id>/index.jsp</to-view-id>
    </navigation-case>
</navigation-rule>
```
32. Save the changes

4.4 Build, Deploy and Run your application

1. Right click on the Enterprise Application project node and in the context menu select Development Component → Build…

2. In the Build DCs popup window click the "OK" button to build the DCs.

3. Select the project you want to deploy by click the icon in the deploy view. In the popup window select the EAR file you want to deploy, in this case the one generated by the Enterprise Application that you created and built. Click the “OK” button.

4. Click the deploy icon to start the deployment to the server. If this is your first deploy to the server since starting the NWDS you will have to enter your user name and password which has administrator rights.

5. Open the Enterprise Application DC project and right click on the node directly under the project node. In the context menu select “Create Application.xml”. This will create and open the application.xml deployment descriptor

6. In the Editor of the application.xml switch to the source view, and sets the WAR file to “demo.sap.com~navigationjsf~web.war” and the context root to “navigation” as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).

7. Save the application.xml deployment descriptor.

8. Build and Redeploy the application.

9. Run the application using the following simplified URL:
    http://<servername>:<httpport>/navigation/faces/index.jsp

10. Results:
www.sdn.sap.com/irj/sdn/howtoguides