How To... Call An Enterprise Service From JSF

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>Example Text</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>Example text</td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td>Example text</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>Example text</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>EXAMPLE TEXT</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 Locate Enterprise Service ............................................................................................ 2
   4.3 Test the Enterprise Service in Web Services Navigator (Optional) ......................... 8
   4.4 Import Enterprise Service Definition ........................................................................... 10
   4.5 Generate Enterprise Service Proxy ............................................................................. 12
   4.6 Create a Java class .................................................................................................... 15
   4.7 Create JSF Pages ...................................................................................................... 18
   4.8 Build, Deploy and Run your application ..................................................................... 24
1. Business Scenario

The following guide will explain how to create a JSF application that invokes an Enterprise Service using Java EE annotation. It will also explain how to locate and test Enterprise Services by browsing through the “Enterprise Services Index” in the Enterprise Service-Oriented Architecture website.

For the purpose of this tutorial the Enterprise Service to be consumed is the Read Employee Basic Data operation, located in the Manage Employee In service interface from the Business Partner Data Management Process component. This Enterprise Service will allow you to search employee’s personal information by providing the employee ID number.

2. Background Information

Enterprise SOA has been defined by SAP as “an open architecture for adaptive business solutions”. It allows you to encapsulate the business logic and expose it as Enterprise Services. Enterprise Services are highly-integrated Web Services combined with business semantics that can be located and invoked by other applications over a network. To learn more about the Enterprise SOA you can visit Enterprise Service-Oriented Architecture website

In addition, this example demonstrates the use of Java EE Annotations. Annotations are Java modifiers, similar to public and private, that simplify the application development process by allowing developers to specify within the Java class itself how the application component behaves in the container and requests for dependency injection. To learn more about Java EE Annotations you can visit Introduction to Java EE 5 Technology

3. Prerequisites

The following is a list of all you need for developing JSF applications and invoking Enterprise Services.

- 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

- A registered user on the ES Workplace who can consume the Enterprise Services. To get your user for free, you can visit Enterprise SOA Registration Form

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 JSF tutorials (Create a Hello World Application using JavaServer Faces [Extern] and Create Your First JSF Application [Extern])
4. Step-by-Step Procedure

In the following sections, you will create a Web Module Development component and an Enterprise Application needed to deploy the web module. You will get to know how to locate an Enterprise Service in the Enterprise SOA website and how to invoke it from your JSF page.

This Web application will consist of two views. In the first view, the user should be able to enter the employee ID in the input text to find the employee information. If no information is found, an error view will be displayed. From the error 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 `esjsf/web`.
2. Create an Enterprise Application Development Component named `esjsf/ear`.

4.2 Locate Enterprise Service

The fastest way to locate your Enterprise Service is to browse through the “Enterprise Services Index”. The navigation hierarchy is `Process Component → Service Interface(s) → Service Operation(s)`. For
more details, check the [ES Workplace Handbook](www.sdn.sap.com) to efficiently browse and testdrive Enterprise Services.

1. Go to the [www.sdn.sap.com](www.sdn.sap.com)

2. In the left navigation menu select Service-Oriented Architecture → Explore Enterprise Services → ES Workplace

3. In the main page click in the ERP Process Components link to get the list of the process components found in the ERP system

4. Select the Business Partner Data Management process component.
5. The selection of *Business Partner Data Management* results in a list of enterprise service interfaces supporting this process component. Select the *Manage Employee In* service interface.

<table>
<thead>
<tr>
<th>Service Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Replication Requesting Out</td>
</tr>
<tr>
<td>Employee Email Data Management In</td>
</tr>
<tr>
<td>Employee Payment Card Management In</td>
</tr>
<tr>
<td>Entries for Employee</td>
</tr>
<tr>
<td>Manage Business Partner Sales Account In</td>
</tr>
<tr>
<td>Manage Customer In</td>
</tr>
<tr>
<td>Manage Employee In</td>
</tr>
</tbody>
</table>

6. This modeled interface comprises several Service Operations (Enterprise Services): Cancel Personal Address, Change Personal Address, Create Personal Address, Read Employee Basic Data and so on.

7. Select the *Read Employee Basic Data* Enterprise Service

<table>
<thead>
<tr>
<th>Service Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel Personal Address</td>
</tr>
<tr>
<td>Change Personal Address</td>
</tr>
<tr>
<td>Create Personal Address</td>
</tr>
<tr>
<td>Read Employee Basic Data</td>
</tr>
<tr>
<td>Read Employee Competency</td>
</tr>
<tr>
<td>Read Employee Dependant</td>
</tr>
<tr>
<td>Read Employee Name</td>
</tr>
<tr>
<td>Read Employee Photo</td>
</tr>
</tbody>
</table>

8. The Enterprise Service is by default documented with a definition, Technical Data, Business Context, etc. In the *Technical Data* section, select the *WSDL (backend)* link, which points to the related backend system and shows the WSDL for this enterprise service.
9. You will be prompted for authentication, enter your username and password which you obtained at the Enterprise SOA Registration Form. The next screen would bring up the WSDL as an XML file.

**Note**

The WSDL descriptor describes the services in a language-independent manner. The WSDL file does not specify what the service does. It specifies only the parameter and return types.
10. The WSDL describes an EmployeeBasicDataByEmployeeQueryResponse_In operation name as follows:

```xml
<wSDL:operation name="EmployeeBasicDataByEmployeeQueryResponse_In">
  <wSDL:input message="tns:EmployeeBasicDataByEmployeeQuery" />
  <wSDL:output message="tns:EmployeeBasicDataByEmployeeResponse" />
  <wSDL:fault name="exception00" message="tns:exception00" />
</wSDL:operation>
```

...
11. It also defines the `EmployeeBasicDataByEmployeeQuery` and `EmployeeBasicDataByEmployeeResponse` types as follows:

```xml
<xsd:element name="EmployeeBasicDataByEmployeeQuery"
    type="xi1:EmployeeBasicDataByEmployeeQueryMessage" />
<xsd:element name="EmployeeBasicDataByEmployeeResponse"
    type="xi1:EmployeeBasicDataByEmployeeResponseMessage" />
```

12. With this information you will be able to use the enterprise service from your Application.

13. Save the WSDL by using the `Save as` option of your browser. Note that the file extension has to be changed from XML to WSDL.
How To... Call An Enterprise Service From JSF

4.3 Test the Enterprise Service in Web Services Navigator (Optional)

The Web Services Navigator (WS Navigator) allows you to test an enterprise service without having a client for it.

1. To start the WS Navigator, in a Web browser, enter the following address:
   

2. The WS Navigator opens. In the WSDL URL field, enter the following address and push the Go button:
   
   sap-client=800&wsdl=1.1&mode=sap_wsdl](http://erp.esworkplace.sap.com/sap/bc/srt/xip/sap/ecc_employeebasicbyemployeedqr/version2?
   sap-client=800&wsdl=1.1&mode=sap_wsdl)

3. To execute the Web service operations, enter your user name and password, which you obtained at the Enterprise SOA Registration Form.

4. In the Select Operation step, choose EmployeeBasicDataByEmployeeQueryResponse_In...
5. In the *Enter Input Parameters* step, in the *Parameters* area enter the *EmployeeID* and push the *Execute* button.

6. In the *Result* step, the system displays the input parameters and the result of the test.
4.4 Import Enterprise Service Definition

You need to import a WSDL document to be able to generate enterprise service proxies.

**Important**

You can import WSDL documents from the following sources:

- Enterprise Services Repository (ESR): You import the WSDL document for a Service Interface (SI) modeled in the ESR
- File system or a remote location such as a URL
- Services Registry (SR): You import the WSDL document of a Web service published in the SR

1. Choose *File → Import → Web services → Import WSDL.*
2. The WSDL Import wizard starts. In Output folder field, browse to the Web module project where you want to import the WSDL document. Under Available wsdl sources choose Remote Location / File System, and then choose the Next button.

3. In the URL field, provide the path to the WSDL document, and then choose Finish.
4. The WSDL document is imported in the SAP NetWeaver Developer Studio. The original filename is preserved. If the WSDL document imports other WSDL documents or schemas, their filenames are preserved as well.

4.5 Generate Enterprise Service Proxy

In this part you will generate a deployable proxy out of the downloaded WSDL.

Note
A deployable proxy is a Web service client, which can and has to be deployed on the application server to be operational. A deployable proxy is consumed from an Enterprise Java Bean, a servlet or a Java class and is created and configured by the responsible...
How To... Call An Enterprise Service From JSF

managing container of the application server. Deployable proxies are used through the Java EE 5 injection mechanism.

1. In the Project Explorer of the Web Module project, choose the WSDL document, and then from the context menu, choose Web Services → Generate Client.

2. The Web Services wizard opens. On the Web Services screen, move the slider to the Develop Client position. In this case, the Web service framework only generates the relevant artifacts. Push the Next button.

Note

Under Configuration, choose Service EAR Project, and make sure that the correct service EAR project is selected in the Service project field.
3. On the **Client Generation Configuration** screen, choose the src folder of the Web Module project and choose the way you want to resolve the possible collisions in the WSDL to Java mapping during the conversion of the WSDL document to Java artifacts. Push the **Finish** button.
4. The framework generates the service endpoint interface, reference classes and the service class in the output folder and the class path is updated with additional JAR files.

4.6 Create a Java class

Use this procedure to invoke the Enterprise Service proxy in your application.

1. From the context menu of the Java Resources: source folder in the Web Module project, create a Java class. Enter EmployeeClient in the Name field, com.sap.tutorial.jsf.es.beans in the Package field.
2. Add the following annotated field to the Java class to inject the service reference:

```java
@WebServiceRef (name = "EmployeeBasicData")
EmployeeBasicDataByEmployeeQueryResponseInService service;
```

**Note**

The proxy is consumed from an EJB or Web module and is created and configured by the responsible managing container. Therefore, to allow the container to create and configure the proxy, a Java EE injection mechanism has to be used. This is achieved with the `@WebServiceRef(name="MyName")`. The string "MyName" has to be unique for the Enterprise JavaBean that uses the client or for the whole Web module.

```java
public class EmployeeClient {
    @WebServiceRef (name = "EmployeeBasicData")
    EmployeeBasicDataByEmployeeQueryResponseInService service;
}
```

3. Declare the following attributes and generate the corresponding **Getters** and **Setters** methods

```java
String employeeID;
String employeeName;
String employeeLastname;
XMLGregorianCalendar employeeBirthday;
String employeeMaritalStatus;
```

4. Create a **search** method and implement it with the following code

```java
public String search(){
    try {
        // Retrieve an instance of the configuration port
        EmployeeBasicDataByEmployeeQueryResponseInService port = service.
        .getEmployeeBasicDataByEmployeeQueryResponse_InSoapBinding();
        // specify username and password issued for you via SDN
        // to authenticate to the backend as follows
        javax.xml.ws.BindingProvider bp =
            (javax.xml.ws.BindingProvider) port;
        Map<String, Object> context = bp.getRequestContext();
        context.put("javax.xml.ws.security.auth.username", "username");
        context.put("javax.xml.ws.security.auth.password", "password");
        // Construct the input to be applied to the service call
        EmployeeID employeeID = new EmployeeID();
```
employeeID.setValue(this.employeeID);
EmployeeBasicDataSelectionByEmployee employeeByEmployee =
    new EmployeeBasicDataSelectionByEmployee();
employeeByEmployee.setEmployeeID(employeeID);
EmployeeBasicDataByEmployeeQueryMessage input =
    new EmployeeBasicDataByEmployeeQueryMessage();
input.setEmployeeBasicDataSelectionByEmployee(employeeByEmployee);
// Invoke the business methods
EmployeeBasicDataByEmployeeResponseMessage output = port
    .employeeBasicDataByEmployeeQueryResponseIn(input);
Employee employee = output.getEmployee();
// Get the response
this.employeeName = employee.getCommon().getName().getGivenName();
this.employeeLastname =
    employee.getCommon().getName().getFamilyName();
this.employeeBirthday = employee.getCommon().getBirthDate();
this.employeeMaritalStatus =
    employee.getCommon().getMaritalStatusName().getValue();
return "ok";
} catch (Exception e) {
    // Clean the object and return an error
    this.employeeID = "";
    this.employeeName = "";
    this.employeeLastname = "";
    this.employeeBirthday = null;
    this.employeeMaritalStatus = "";
    return "error";
}

5. Configure the EmployeeClient Java class in the application configuration resource file faces-config.xml using the managed-bean XML element. Enter employeeClient in the Name field to reference the EmployeeClient java class and select session in the Scope field. The following XML code will be added in the Source tab

<managed-bean>
    <managed-bean-name>employeeClient</managed-bean-name>
    <managed-bean-class>
        com.sap.tutorial.jsf.es.beans.EmployeeClient
    </managed-bean-class>
</managed-bean>
4.7 Create JSF Pages

1. From the context menu of the Java Resources: source folder in the Web Module project, create a new package com.sap.tutorial.jsf.es.util

2. Create the ResourceBundle. Choose the com.sap.tutorial.jsf.es.util package and enter messages.properties in the File Name.

3. Enter the following keys and values for the English version of the localized messages:
   
title=Search Employee
emp_id=Employee ID
search_button=Search
emp_name=Name
emp_lastname=Last Name
emp_birthday=Birthday
emp_marital=Marital Status
errorOccurred=Sorry, there is no employee with the ID specified.
errorBack=Please go back and try again
back_button=Back

4. For simplicity, only the English version of the localized message is created. Optionally you can create other versions of the localized messages and specify which languages are supported for this application as indicated in the Product Offer tutorial Part 3 (International JSF application [extern]).

5. Expose the ResourceBundles by adding the following XML code in the Source tab of the faces-config.xml file:

   <application>
     <resource-bundle>
       <base-name>com.sap.tutorial.jsf.es.util.messages</base-name>
       <var>msgs</var>
     </resource-bundle>
   </application>

6. Create a Style file as indicated in the Product Offer tutorial (Convert and Validate Data [extern]). You are going to use the same CSS classes, so you can copy the styles.css file from that tutorial.

7. Drill into the Web Module project and right click on the WebContent folder and in the context menu select New → JSP.
8. 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.

9. Include the style sheet by adding a `link` element inside the `head` element as shown in the following code:

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

10. The following table contains the hierarchy of the UI elements contained in the `index` view:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ViewRoot UI element</code></td>
<td></td>
</tr>
<tr>
<td><code>Form UI element in the UI-element ViewRoot</code></td>
<td></td>
</tr>
<tr>
<td><code>OutputText UI element in the UI-element Form</code></td>
<td>Value: #{msgs.title}</td>
</tr>
<tr>
<td></td>
<td>styleClass: title</td>
</tr>
<tr>
<td><code>PanelGrid UI element in the UI-element Form</code></td>
<td>Border: 0</td>
</tr>
<tr>
<td></td>
<td>Columns: 2</td>
</tr>
<tr>
<td><code>OutputText UI element in the UI-element PanelGrid</code></td>
<td>Value: #{msgs.emp_id}</td>
</tr>
<tr>
<td></td>
<td>styleClass: label</td>
</tr>
<tr>
<td><code>InputText UI element in the UI-element PanelGrid</code></td>
<td>Value: #{employeeClient.employeeID}</td>
</tr>
<tr>
<td><code>CommandButton UI element in the UI-element Form</code></td>
<td>Value: #{msgs.search_button}</td>
</tr>
<tr>
<td></td>
<td>action: #{employeeClient.search}</td>
</tr>
<tr>
<td><code>PanelGrid UI element in the UI-element Form</code></td>
<td>Border: 0</td>
</tr>
<tr>
<td></td>
<td>Columns: 2</td>
</tr>
<tr>
<td><code>OutputText UI element in the UI-element PanelGrid</code></td>
<td>Value: #{msgs.emp_name}</td>
</tr>
<tr>
<td></td>
<td>styleClass: label</td>
</tr>
<tr>
<td><code>OutputText UI element in the UI-element PanelGrid</code></td>
<td>Value: #{employeeClient.employeeName}</td>
</tr>
<tr>
<td></td>
<td>styleClass: text</td>
</tr>
</tbody>
</table>
OutputText UI element in the UI-element *PanelGrid*
value #{msgs.emp_lastname}
styleType label

OutputText UI element in the UI-element *PanelGrid*
value #{employeeClient.employeeLastname}
styleType text

OutputText UI element in the UI-element *PanelGrid*
value #{msgs.emp_marital}
styleType label

OutputText UI element in the UI-element *PanelGrid*
value #{employeeClient.employeeMaritalStatus}
styleType text

OutputText UI element in the UI-element *PanelGrid*
value #{msgs.emp_birthday}
styleType label

OutputText UI element in the UI-element *PanelGrid*
value #{employeeClient.employeeBirthday}
styleType text

11. Result of index.jsp
12. Save the changes you made.

13. Create the another view selecting New → JSP in the context menu of the WebContent folder

14. Enter the file name errorDisplay.jsp and click the Finish button. The JSP page will be created. The errorDisplay.jsp page should be opened in the Web Page Editor

15. Include the style sheet by adding a link element inside the head element as shown in the following code

```html
<head>

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

...</head>
```

16. The following table contains the hierarchy of the UI elements contained in the index view:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewRoot UI element</td>
<td></td>
</tr>
<tr>
<td>Form UI element in the UI-element ViewRoot</td>
<td></td>
</tr>
<tr>
<td>OutputText UI element in the UI-element Form</td>
<td></td>
</tr>
</tbody>
</table>
value #{msgs.errorOccurred}

styleClass errorMessage

OutputText UI element in the UI-element Form
value #{msgs.errorBack}
styleClass errorMessage

CommandButton UI element in the UI-element Form
value #{msgs.back_button}
action back

17. Result of errorDisplay.jsp

18. 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.

19. Go to the Navigation Rule tab to define the navigation. The navigation flow should look like the following image:

Note

The CommandButton UI element in the index.jsp view has the search method in the action property. If it returns the “error” string, the application will navigate to the errorDisplay.jsp view.
20. Select the Source tab. The following XML code should be added automatically between the `<faces-config ...>` `<faces-config>` tags

```xml
<navigation-rule>
    <display-name>index</display-name>
    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
        <from-outcome>error</from-outcome>
        <to-view-id>/errorDisplay.jsp</to-view-id>
        <redirect />
    </navigation-case>
</navigation-rule>

<navigation-rule>
    <display-name>errorDisplay</display-name>
    <from-view-id>/errorDisplay.jsp</from-view-id>
    <navigation-case>
        <from-outcome>back</from-outcome>
        <to-view-id>/index.jsp</to-view-id>
    </navigation-case>
</navigation-rule>
```

21. Save the changes you made
4.8 Build, Deploy and Run your application

1. Create the application.xml deployment descriptor, sets the WAR file to “demo.sap.com~esjsf~web.war” and the context root to “esjsf” as indicated in the Hello World JSF tutorial (Create a Hello World Application using JavaServer Faces [Extern]).
2. Save changes.
3. Build and deploy the application.
4. Run the application using the following simplified URL:
   http://<servername>:<httpport>/esjsf/faces/index.jsp
5. Results:
How To... Call An Enterprise Service From JSF

[Image of search employee interface with employee ID 1015, showing name Alexander, last name Rickes, marital status Married, and birthday 1952-11-13]

[Image of search employee interface with message saying there is no employee with the ID specified, asking to try again]

October 2008