

Creating Your First J2EE Application



SAP NetWeaver 04



Copyright

© Copyright 2004 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

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.

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
<i>Example text</i>	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.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
Example text	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Creating Your First J2EE Application	5
Creating an EJB Module Project.....	6
Developing the Calculator Session Bean	7
Creating the Enterprise Bean Archive	10
Creating a Web Module Project.....	11
Developing the JavaBean CalcProxy	12
Developing the Calculator.jsp	15
Building the Web Archive.....	18
Assembling the Calculator Application	20
Deploying and Running the Calculator Application.....	22



Creating Your First J2EE Application

Purpose

The SAP Web Application Server provides an integrated framework for developing and using applications written according to the Java™ 2 Enterprise Edition (J2EE) standards. You can create your application using the SAP NetWeaver Developer Studio, and then deploy the application on the SAP J2EE Engine.

This guide contains the steps that you must follow to successfully develop, deploy, and run a J2EE application that consists of a session enterprise bean, a JavaBean, and a JSP component.

The objective of this simple application is to show you how to:

- Use the J2EE toolset in the SAP NetWeaver Developer Studio when creating J2EE applications
- Implement business logic using an enterprise bean
- Create the front end of a J2EE application
- Connect the front end to the underlying business component

The design implemented here is one of the possible scenarios for the Calculator application. It demonstrates the use of the model-view-controller (MVC) pattern. The model layer is represented by the Calculator session bean, which does the actual work – adds, subtracts, divides, or multiplies two arbitrary numbers. The Calculator JSP is the front end component providing the view of the application. Finally, the application uses a JavaBean in the controller layer for the communication between the front-end and the enterprise bean.

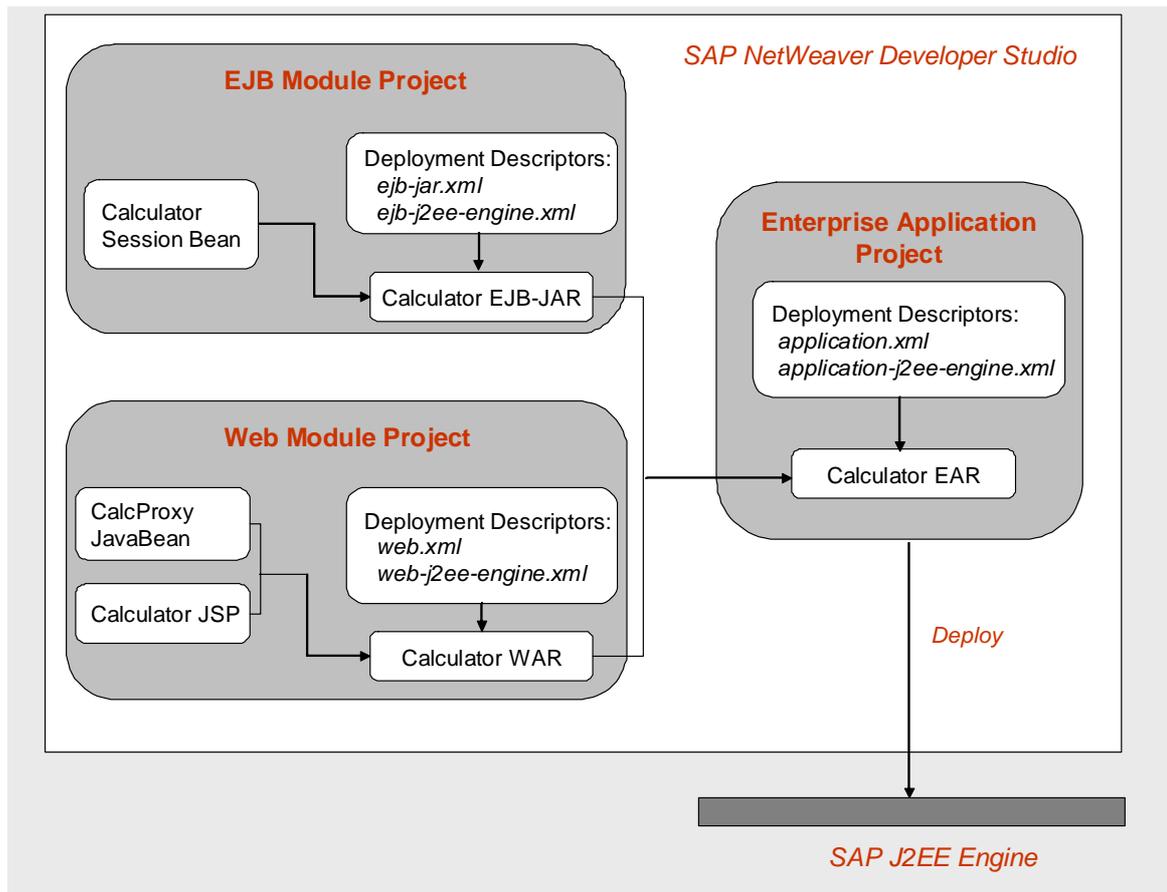
Process of Developing the Calculator Application

The SAP NetWeaver Developer Studio enables you to work either using the development configuration, or in a component-free environment. This tutorial is based on a component-free scenario.

The process of developing and deploying the Calculator application consists of the following steps:

- [Creating an EJB Project \[page 6\]](#)
- [Developing the enterprise bean \[page 7\]](#)
- [Assembling the enterprise bean archive \[page 10\]](#)
- [Creating a Web Project \[page 11\]](#)
- [Developing the JavaBean \[page 12\]](#)
- [Developing the JSP \[page 15\]](#)
- [Assembling the web archive \[page 18\]](#)
- [Assembling the application \[page 20\]](#)
- [Deploying and running the application on the SAP J2EE Engine \[page 22\]](#).

This process is illustrated in the graphic below:



Requirements

In order to create and run the Calculator application as described in the tutorial, you have to meet the following requirements:

- The SAP NetWeaver Developer Studio is installed on your computer.
- You have access to the SAP J2EE Engine.
- The procedures in this tutorial assume that you use the workbench default settings (Auto build, etc.).

Where to Go from Here

Start developing your first J2EE application by [creating an EJB Project \[page 6\]](#).



Creating an EJB Module Project

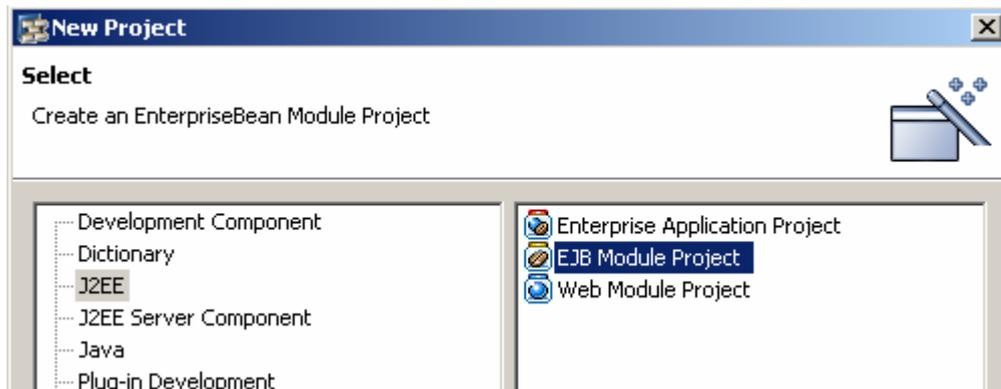
Use this procedure to create the EJB module project, which will provide the framework for the development of the Calculator enterprise bean.

Prerequisites

- You have launched the SAP NetWeaver Developer Studio.

Procedure

1. Choose *File* → *New...* → *Project* to start the *New Project* wizard.
2. Select *J2EE* (in the left-hand pane) and *EJB Module Project* (in the right-hand pane). Choose *Next*.

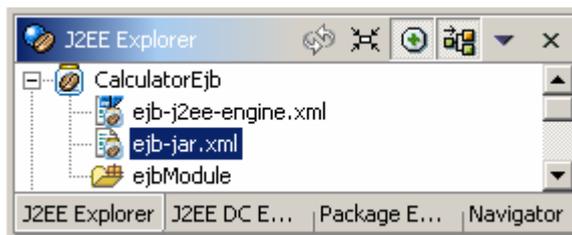


3. Give your EJB module project the name **CalculatorEjb** and leave the default settings for *Project contents* unchanged.
4. Choose *Finish*.

Result

The wizard generates a project structure for your new EJB module project. It also contains the deployment descriptor files **ejb-jar.xml** and **ejb-j2ee-engine.xml**.

It automatically opens the *J2EE* perspective and displays the new project structure in the *J2EE Explorer*.



Next step:

You can now [create an enterprise bean \[page 7\]](#) using the EJB module project structure.

Developing the Calculator Session Bean

Use this procedure to create a session enterprise bean that calculates the result from two float numbers. The enterprise bean has four business methods for adding, subtracting, dividing, and multiplying the numbers.

Prerequisites

- You have created the EJB module project *CalculatorEjb*. The structure of your project is currently displayed in the *J2EE Explorer*.

Procedure

Specifying the general properties of the Session Bean

1. In the J2EE Explorer, select the project node *CalculatorEjb* and open the context menu. Choose *New* → *EJB...*
2. In the New EJB wizard screen that appears, assign the following values to the properties of the new EJB.

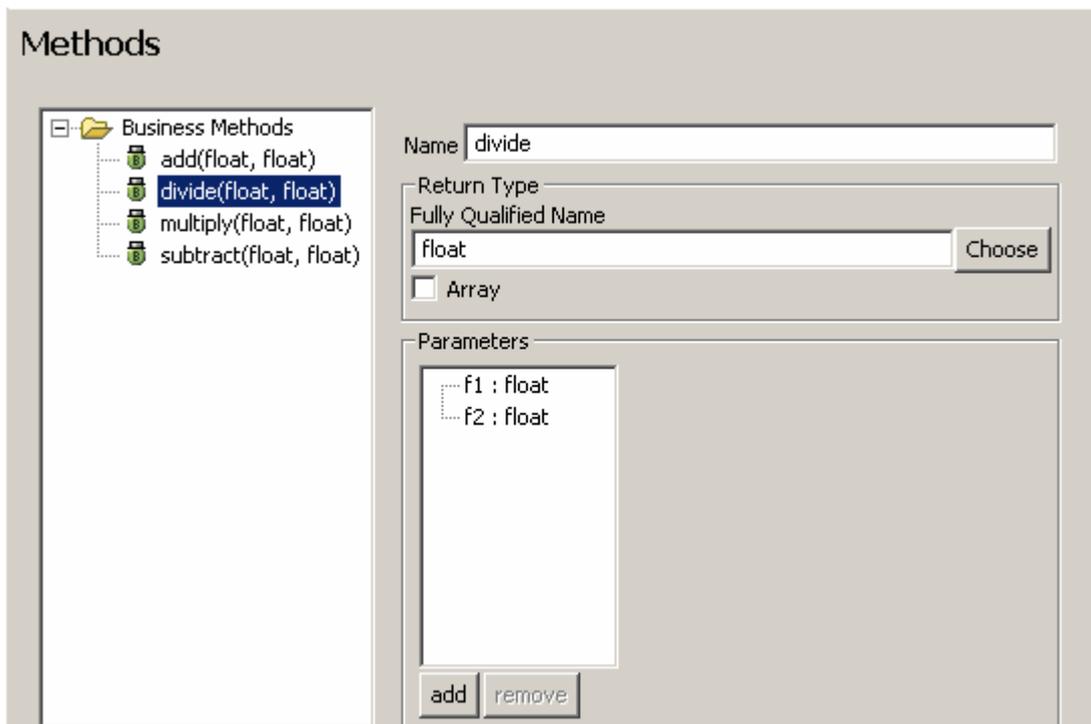
Property	Value
EJB Name	Calculator
EJB Project	CalculatorEjb
Bean Type	Stateless Session Bean
Default Package	com.sap.examples.calculator
Generate default interfaces	disabled

3. Choose *Next*.
The New EJB wizard appears and displays the screen containing names of the remote and the local interfaces.
4. Accept the suggested names of the local and remote interfaces and choose *Next*.
In the screen that appears, you can add the required business methods.

Creating business methods

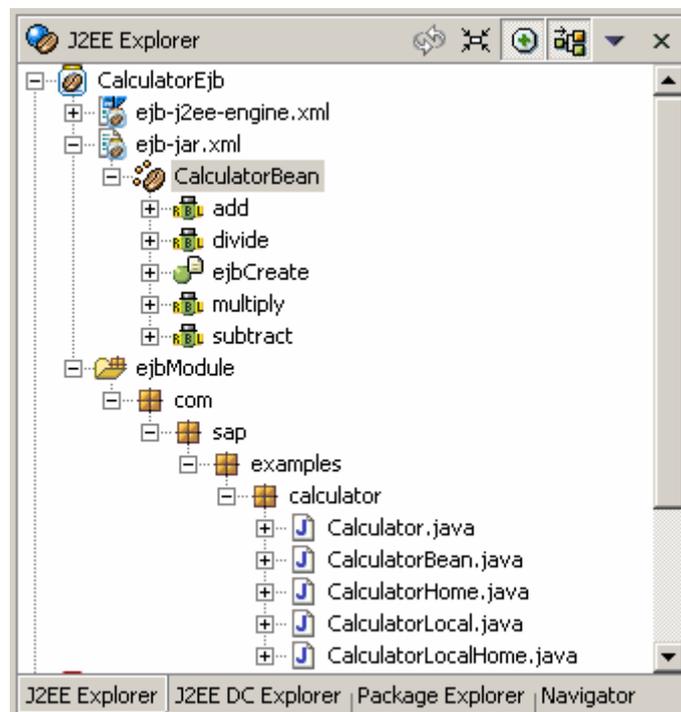
In the Calculator bean, you need to add and implement the four business methods: **add**, **subtract**, **multiply**, and **divide**. For all of these methods, you will specify *float* as the return value and two parameters **f1** and **f2**, each of them also of type *float*.

5. To add the first method **add**, select the *Business Methods* folder in the left pane of the wizard screen and choose *Add*.
6. Enter the name **add** and assign the return type **float**.
7. Add the parameters **f1** and **f2** of type **float** to this method.
8. In the left pane of the wizard screen, select the *Business Methods* folder to update the display.
9. To add the other methods (**subtract**, **multiply**, and **divide**), repeat the last steps as appropriate.



10. Choose *Finish* to create the Session Bean.

In the J2EE Explorer, a new substructure named *CalculatorBean* is inserted automatically under the project node *CalculatorEjb* → *ejb-jar.xml*.



Implementing the business methods

11. In the project structure, expand the *CalculatorEjb* → *ejb-jar.xml* node and double-click the name *CalculatorBean*.
12. In the overview that appears, choose *Navigate to Bean Class*.

The Java Editor opens and displays the source code of the generated bean class.

13. Navigate to the implementation of the business methods.
14. Modify the return values as follows:

```
/**
 * Business Method.
 */
public float add(float f1, float f2) {
    return f1 + f2;
}

/**
 * Business Method.
 */
public float subtract(float f1, float f2) {
    return f1 - f2;
}

/**
 * Business Method.
 */
public float multiply(float f1, float f2) {
    return f1 * f2;
}

/**
 * Business Method.
 */
public float divide(float f1, float f2) {
    return f1 / f2;
}
```

15. Save the contents of the editor by choosing the appropriate icon from the toolbar.
The Developer Studio updates and compiles the project sources.

Result

The Calculator bean has been created in `com.sap.examples.calculator` package. The package now contains the bean's class and the bean's home and remote interfaces. You have also completely implemented the business methods of the Session Bean.

Next step:

You continue with [creating a Enterprise Bean archive \[page 10\]](#).



Creating the Enterprise Bean Archive

Use this procedure to create a JAR file that contains the Calculator session bean and deployment descriptors – `ejb-jar.xml` and `ejb-j2ee-engine.xml`.

Prerequisites

- You have created and compiled the Calculator enterprise bean in the project `CalculatorEjb`.

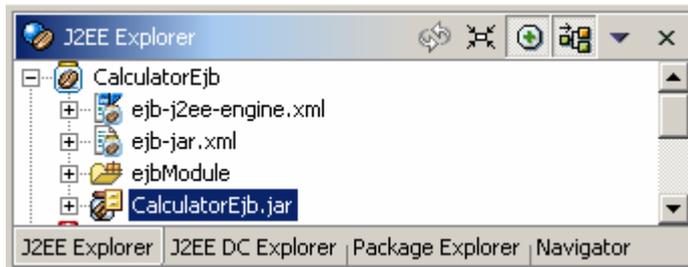
Procedure

1. In the *J2EE Explorer*, select the project node `CalculatorEjb` and open the context menu.

2. Choose *Build Ejb.jar File*.
3. Confirm the message informing you that the build has been successful by choosing *Ok*.

Result

The archive file *CalculatorEjb.jar* has been generated. It contains all the classes of the EJB module project.



Next step:

Now continue with [creating the Web project \[page 11\]](#).



Creating a Web Module Project

Use this procedure to create the Web module project, which provides the framework for developing the Calculator JSP and the JavaBean helper class.

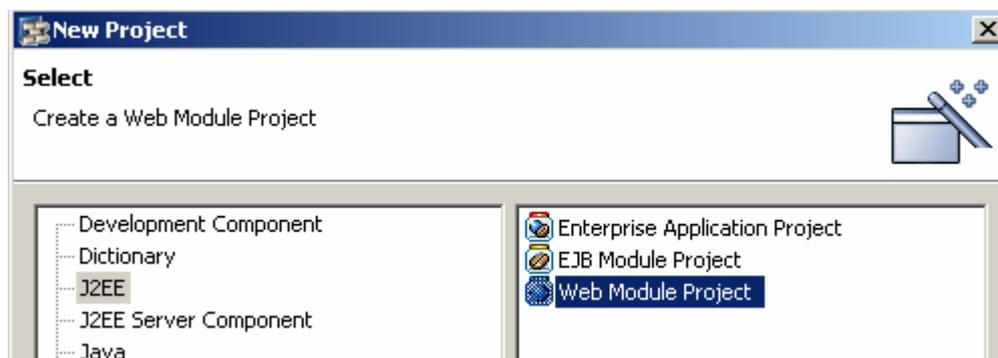
Prerequisites

- You have already created the EJB module project **CalculatorEjb**.

Procedure

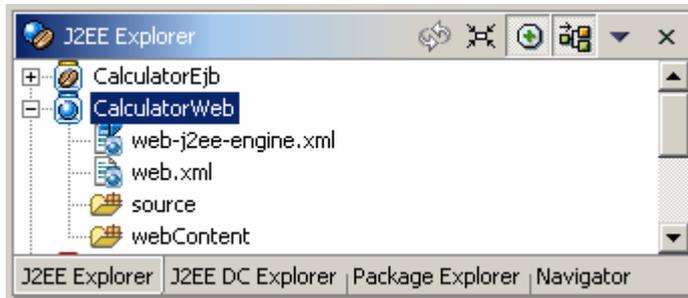
Creating the project structure

1. Choose *File* → *New* → *Project* to start the *New Project* wizard.
2. Select *J2EE* in the left-hand pane, and *Web Module Project* in the right-hand pane. Choose *Next*.



3. Enter the project name `CalculatorWeb` and leave the default settings unchanged. Choose *Finish*.

The project structure for the new Web module project is generated and displayed in the J2EE Explorer. It also contains the deployment descriptors `web.xml` and `web-j2ee-engine.xml`.



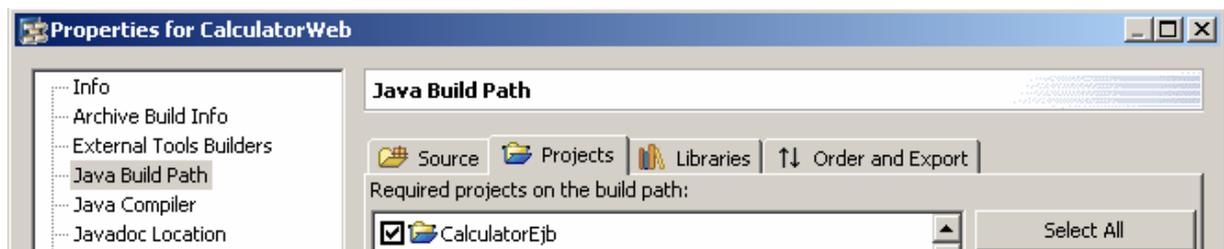
Specifying the project dependencies

Since the EJB business methods from the Session Bean are to be referenced in the Web resources, you will need to specify the EJB module project `CalculatorEjb` in the project build path.

1. In the *J2EE Explorer*, select the project node `CalculatorWeb` and open the context menu.
2. Choose *Properties*.
3. Choose *Java Build Path* in the left frame. In the right frame, select the *Projects* tab.
4. Select `CalculatorEjb` and confirm with *OK*.

Result

The Web project `CalculatorWeb` is created. For the build path, the `CalculatorEjb` is referenced in the project dependencies.



Next step:

You can now go on with [creating a JavaBean \[page 12\]](#) for the Calculator application.



Developing the JavaBean CalcProxy

Use this procedure to create a JavaBean that retrieves the home interface of the Calculator session bean by performing a lookup operation in the naming system, and which invokes the bean's business methods. The JavaBean also passes the numbers entered in the JSP to the enterprise bean and then returns the result to the JSP.

Prerequisites

- You have created the Web module project `CalculatorWeb`, and you have added the EJB module project `CalculatorEjb` to its build path.

Procedure

Creating the JavaBean class

1. In the *J2EE Explorer*, select *CalculatorWeb* and open the context menu.
2. Choose *New* → *Package...*
3. Enter `com.sap.examples.calculator.beans` as the package name. Choose *Finish*.
The package is created in *CalculatorWeb/source* directory.
4. In the *J2EE Explorer*, select *CalculatorWeb* again and choose *New* → *Java Class...* from the context menu.
5. To specify the JavaBean, choose *Browse* next to the *Package* field and then choose `com.sap.examples.calculator.beans`.
6. Enter `CalcProxy` in the *Name* field.

New Java Class

Java Class
Create a new Java class.

Source Folder:

Package:

Enclosing type:

Name:

Modifiers: public default private protected
 abstract final static

Superclass:

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)

Constructors from superclass

Inherited abstract methods

< Back Next > Finish Cancel

7. Leave the other default settings unchanged and choose *Finish*.

The wizard generates a new java file *CalcProxy.java* and opens the Java editor automatically, displaying the generated content of the new class.

Adding Source Code

1. Add the following source code to implement the logic of the *CalcProxy*.
 - a. First, define a method that retrieves the *Calculator* bean's home interface by performing a lookup operation, and initializes the enterprise bean by invoking the *create()* method of the home interface. You will call this method *init()*.



```
public class CalcProxy {  
  
    private Calculator calc;  
  
    public void init() throws Exception {  
  
        //Lookup the enterprise bean  
        try {  
            InitialContext ctx = new InitialContext();  
            Object ob = ctx.lookup("java:comp/env/ejb/CalculatorBean");  
            CalculatorHome home = ( CalculatorHome )  
PortableRemoteObject.narrow( ob, CalculatorHome.class );  
  
            //Initialize the enterprise bean  
            calc = home.create();  
        } catch ( Exception e ) {  
            throw new Exception("Error instantiating Calculator EJB" +  
e.toString());  
        }  
    }  
}
```

- b. Next, write the constructor of the class and invoke the *init()* method in the constructor.



```
public CalcProxy() throws Exception {  
    init();  
}
```

- c. Finally, write a method *getResult()* that parses the input parameters, invokes the enterprise bean's business methods, and returns the result of the relevant calculation.



```
public float getResult( String firstNumber, String secondNumber, String  
expression ) throws Exception {  
    float result = 0;  
    try {  
        if ( firstNumber != null && secondNumber != null ) {
```

```

//Parse the input parameters
float first = Float.parseFloat( firstNumber );
float second = Float.parseFloat( secondNumber );
int expr = Integer.parseInt( expression );

//Invoke the relevant method of the enterprise bean
switch ( expr ) {
    case 1:
        result = calc.multiply( first, second );
        break;
    case 2:
        result = calc.divide( first, second );
        break;
    case 3:
        result = calc.add( first, second );
        break;
    case 4:
        result = calc.subtract( first, second );
        break;
}
}
} catch (Exception re){
    throw new Exception("Fill in all required fields with appropriate
values!");
}

//Return the result of the calculation
return result;
}

```

2. If necessary, correct the formatting of the code lines by choosing *Source* → *Format* from the context menu.
3. To add the required import statements, position the cursor anywhere in the Java editor and choose *Source* → *Organize Imports*.
4. Choose  `javax.rmi.PortableRemoteObject` and confirm by choosing *Finish*.
The appropriate import statements are added to the source code.
5. Save the contents of the editor using the appropriate icon from the toolbar.
The JavaBean `CalcProxy` is also automatically compiled in `CalculatorWebt/bin` directory. To navigate to this directory, choose the *Navigator* view.

Result

You have created and completely implemented the JavaBean `CalcProxy`, which will serve as the controller part of your Calculator application.

Next step:

Next you [develop the Calculator JSP \[page 15\]](#).



Developing the Calculator.jsp

Use this procedure to create a JSP that invokes the `CalcProxy` JavaBean and acts as a front end to your application.

The JSP contains an input form where you can specify which type of operation you want the enterprise bean to perform, and the numbers that are calculated. The JSP sends the input to the enterprise bean, and then displays the result.

Prerequisites

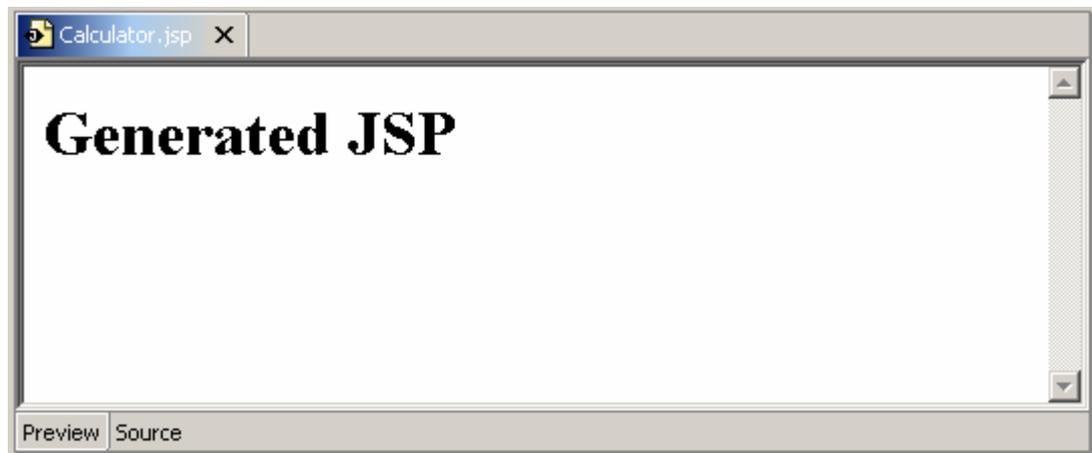
- You have created the Web module project `CalculatorWeb`.

Procedure

Creating the JSP

1. In the *J2EE Explorer*, select *CalculatorWeb* and open the context menu.
2. Choose *New* → *JSP...*
3. Give your JSP the name `Calculator`.
4. Choose *Finish*.

The wizard creates the JSP in the project folder *CalculatorWeb/webContent* and opens the JSP Editor automatically. This editor displays the default content in the *Preview* pane.



Adding Source Code

5. To edit the JSP code, choose the *Source* tab.
 - a. After the page language declaration at the beginning of the page, write a `<jsp:useBean>` statement that enables the JSP to use the `CalcProxy` JavaBean:



```
<jsp:useBean id="calc" scope="session"
class="com.sap.examples.calculator.beans.CalcProxy" />
```

- b. In the HTML part of the JSP, create a form for the input parameters. Insert the code for the form between the `<body></body>` tags, after the `<h1></h1>` tags. The form must include a drop-down menu where the user can choose an operation, input fields for the numbers to be calculated, and a button for submitting the input parameters. The method of the form is `post`.



```
<FORM METHOD="post" ACTION="Calculator.jsp">
<P>Select Operation:</P>
```

```

<P><SELECT NAME="expression">
<OPTION VALUE=1>Multiply
<OPTION VALUE=2>Divide
<option VALUE=3>Add
<option VALUE=4>Subtract
</SELECT></P>
<P>First number:</P>
<P><INPUT NAME="firstnumber" size=10></P>
<P>Second number:</P>
<P><INPUT name="secondnumber" size=10></P>
<P><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Calculate"></P>
</FORM>

<P>
<HR HEIGHT="1px" WIDTH="80%" COLOR="#000000">
</P>

```

- c. In the HTML part of the JSP, enter the code that enables the JSP to retrieve the result of the calculations using the `JavaBean`, or return a message in case of error. Append this code between the `<body></body>` tags.



```

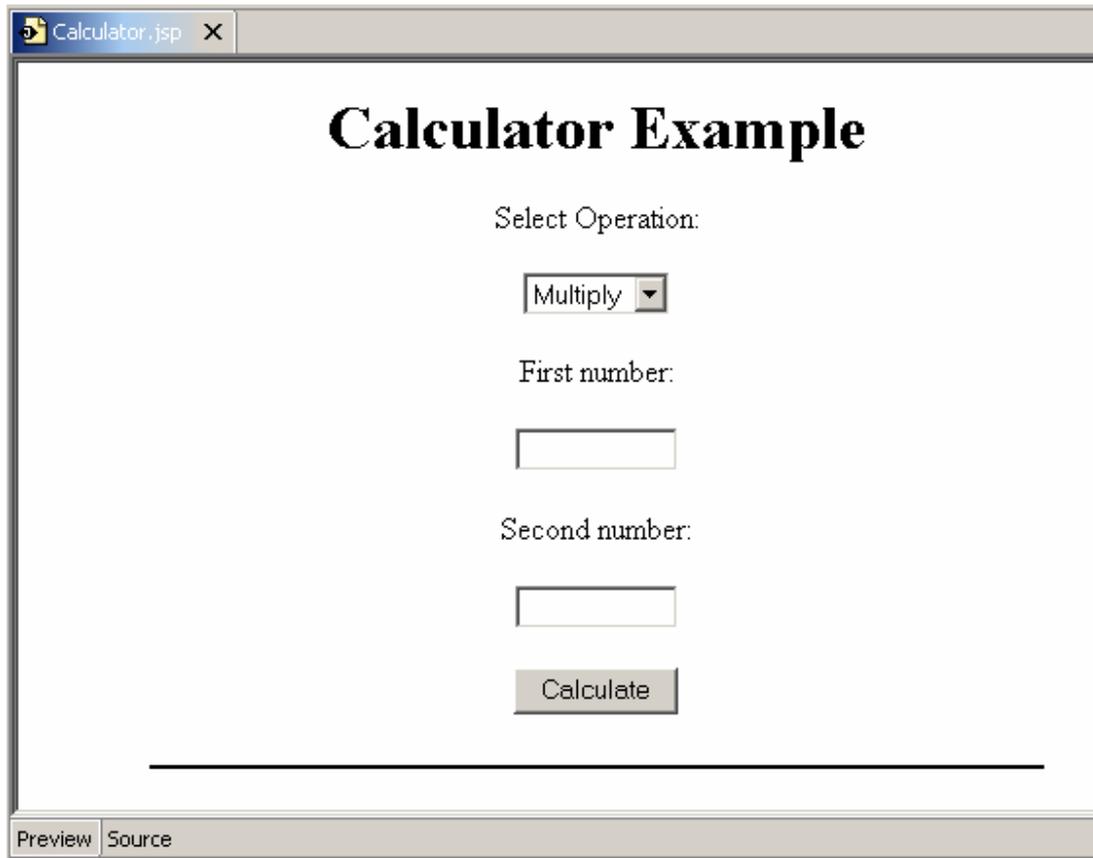
<% try { %>
<P>
<B> <%= "Result is " +
calc.getResult(request.getParameter( "firstnumber" ),
request.getParameter( "secondnumber" ),
request.getParameter( "expression" )) %></B>
</P>
<% } catch (Exception ex) { %>
    <%=ex.getMessage() %>
<% } %>

```

- d. You can edit HTML contents to design the view of the JSP:
- Enter **Calculator** between the `<title></title>` tags
 - Enter **Calculator Example** between the `<h1></h1>` tags
 - Center the text using the `<center></center>` tags in the body part of the page
6. If necessary, correct the formatting of these code lines by choosing *Format* from the context menu.
7. Save the contents of the editor.

Result

The front end of the Calculator application is ready. If you now choose the *Preview* tab in the JSP Editor, the following preview layout is displayed:



Next step:

In the next step you will [create a Web archive \[page 18\]](#).



Building the Web Archive

Use this procedure to create a WAR file that contains the *Calculator.jsp*, the *CalcProxy* JavaBean, and deployment descriptors – *web.xml* and *web-j2ee-engine.xml*.

Prerequisites

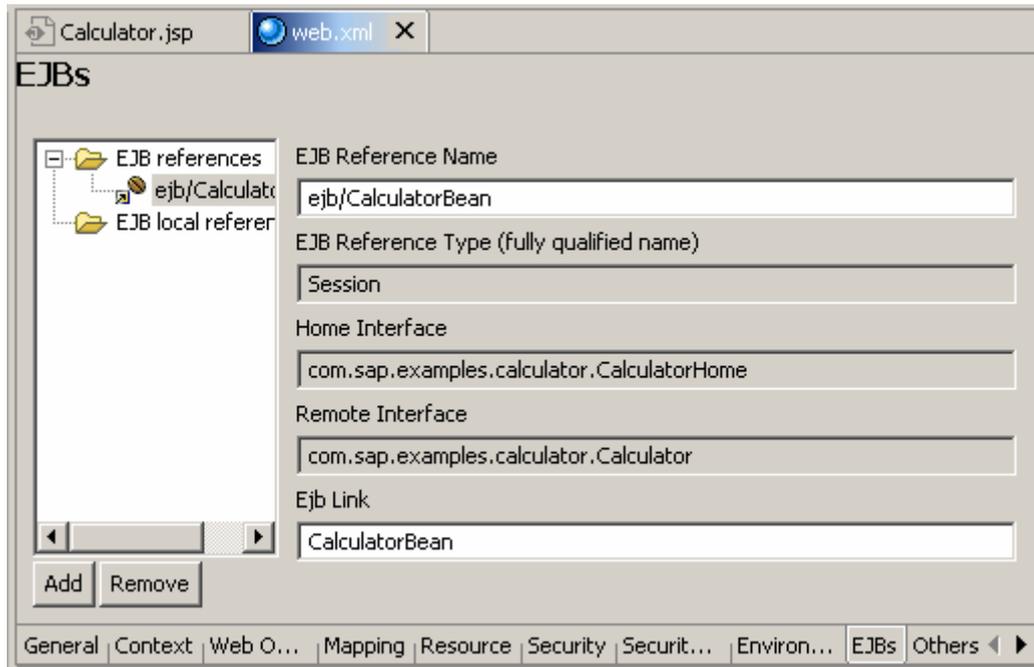
- You have created the Web module project `CalculatorWeb` that contains the *CalcProxy* JavaBean and *Calculator.jsp*.

Procedure

Entering a reference to an EJB using `web.xml`

1. In *J2EE Explorer*, expand the project structure of *CalculatorWeb* and double-click the *web.xml* node.
The Developer Studio opens a multipage editor, which allows you to enter descriptions in the standard deployment descriptor **web.xml**.
2. Choose the *EJBs* tab.
3. Select the node *EJB References* in the left frame and choose *Add*.

4. From the selection list that appears, select *CalculatorEjb* and confirm using *OK*.
5. Enter the name *CalculatorBean* for the *Ejb Link*.



6. Save your changes to the deployment descriptor and close the editor.



The EJB Reference enables you to look up the referenced enterprise bean.

Creating a War File

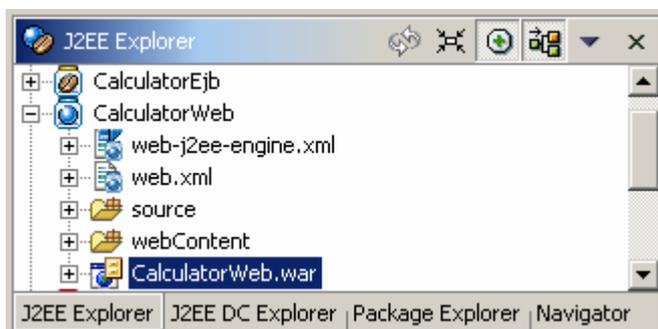
7. In *J2EE Explorer*, select the project node *CalculatorWeb* and open the context menu.
8. Choose *Build Web Archive*.

The system displays a message to inform you that the build was successful.

9. Close the screen with *OK*.

Result

The Web archive file *CalculatorWeb.war* has been generated. It contains all the classes, Web resources of the Web module project, and the entries in the deployment descriptors.



Next step:

Now you can [assemble the application \[page 20\]](#).



Assembling the Calculator Application

Use this procedure to package the Calculator session bean, the CalcProxy JavaBean, and the Calculator JSP in an application archive (EAR file). The EAR file contains the JAR and the WAR files with the application components, as well as deployment descriptors – *application.xml* and *application-j2ee-engine.xml*.

Prerequisites

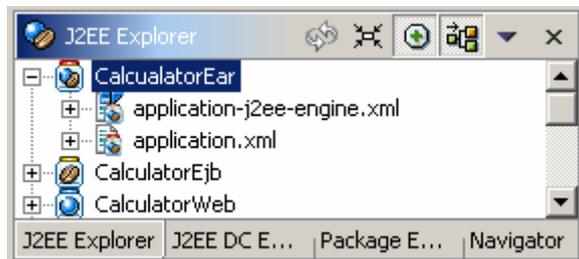
- You have already created the archive files `CalculatorEjb.jar` and `CalculatorWeb.war`.

Procedure

Creating the Enterprise Application Project

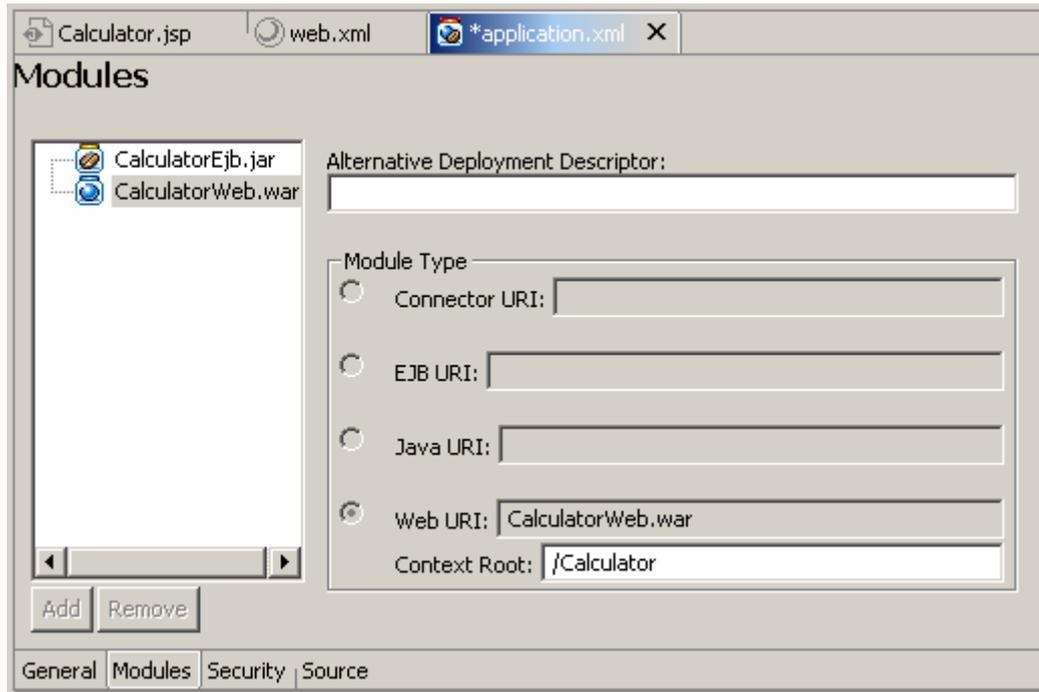
1. Choose *File* → *New* → *Project*.
2. Select *J2EE* (the left-hand pane) and *Enterprise Application Project* (the right-hand pane). Choose *Next*.
3. Enter `CalculatorEar` as the project name and leave the default settings unchanged. Choose *Next*.
4. Select *CalculatorEjb* and *CalculatorWeb* as referenced projects. Choose *Finish*.

The wizard generates the project structure of the new Enterprise Application project.



Defining the Context Root

5. Expand the project node and double-click the *application.xml*.
The deployment descriptor editor appears.
6. Choose the *Modules* tab.
7. Select *CalculatorWeb.war* as a module.
8. Change the name of the predefined value in the *Context Root* field to `/Calculator`.
The new context root `/Calculator` will be used for starting the application.



9. Save the editor contents.

Building the EAR file

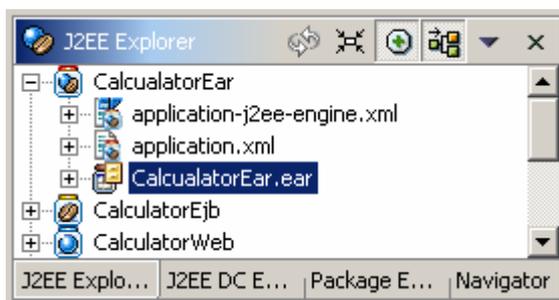
10. In the *J2EE Explorer* view, select *CalculatorEar* and open the context menu.
11. Choose *Project* → *Rebuild All*.
12. Choose *Build Application Archive*.

The system displays a message to inform you that the build has been successful.

13. Close the screen by choosing *Ok*.

Result

The Enterprise archive file *CalculatorEar.ear* has been generated. It contains all the EJB classes and all Web resources, along with the deployment descriptors from the Enterprise Application project.



Next step:

You can [deploy the application \[page 22\]](#) on SAP J2EE Engine.



Deploying and Running the Calculator Application

Prerequisites

- You have created the Calculator EAR file *CalculatorEar.ear*.
- You have made sure that the SAP J2EE Engine has been launched.
To do this, refer to: [Starting and Stopping the SAP System \[extern\]](#).

Procedure

Deploying the EAR

1. In the *J2EE Explorer*, expand the *CalculatorEar* tree structure and select the EAR file *CalculatorEar.ear*.
2. Check if the J2EE Engine is running.
3. Open the context menu and choose *Deploy to J2EE engine*.
4. Enter the SDM password, if required.

The system displays a message to inform you that the deployment process has finished successfully.

Running the Calculator Application

5. Call the calculator application by entering its URL in the Web browser:

`http://<host>:50000/<contextRoot>/<Name_of_JSP.jsp>`



Example: `http://localhost:50000/Calculator/Calculator.jsp`.

