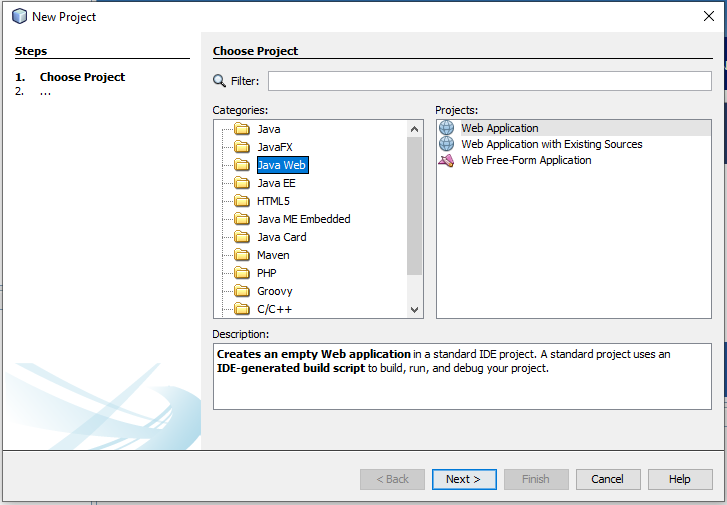
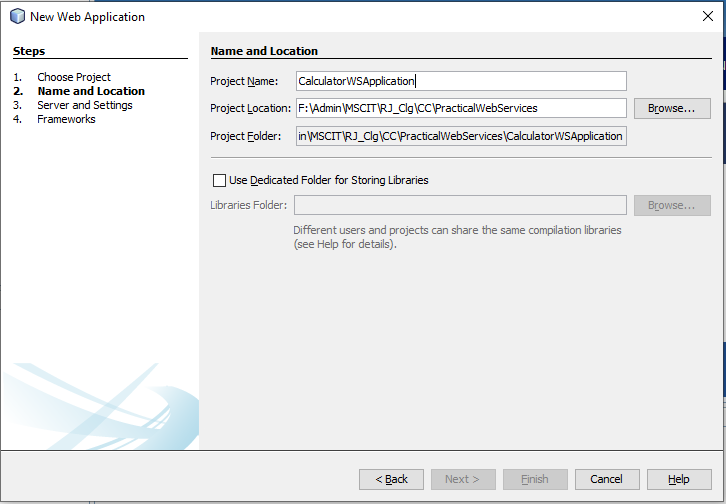
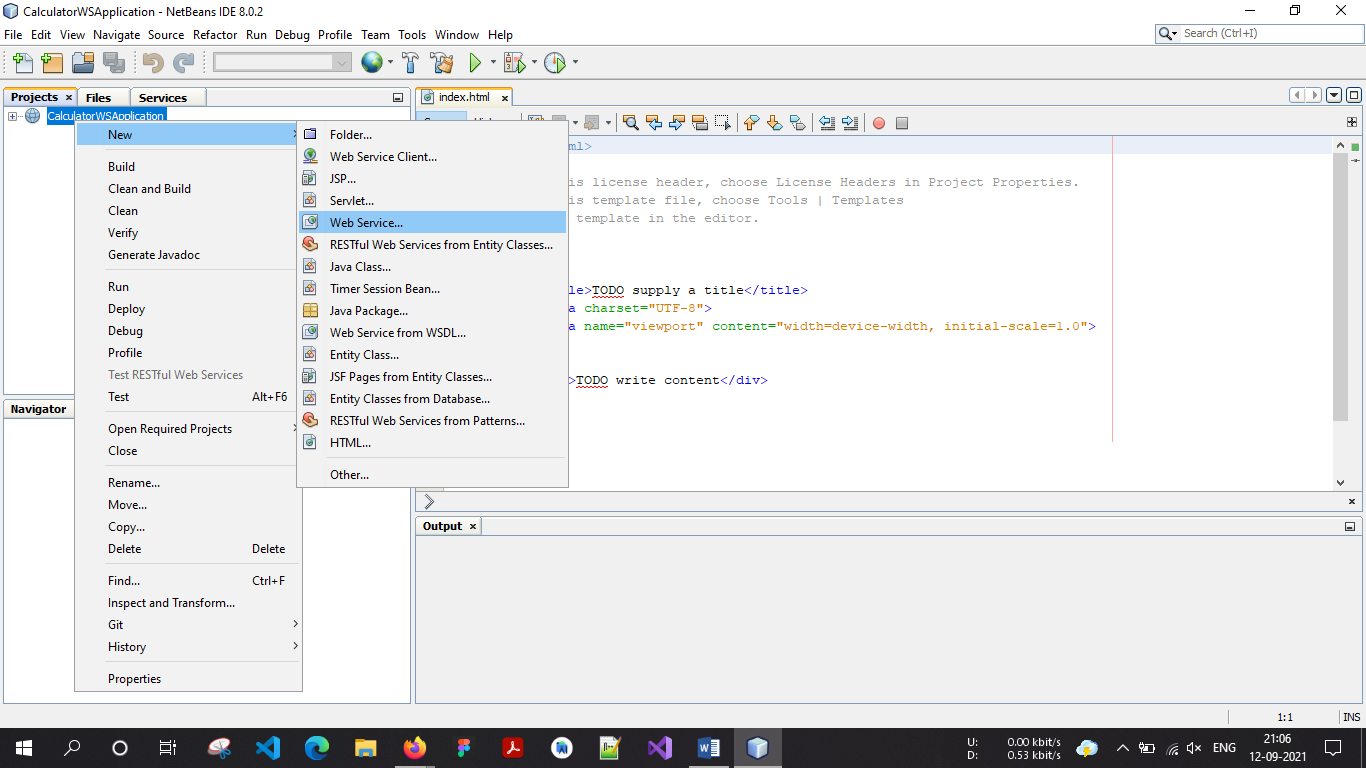
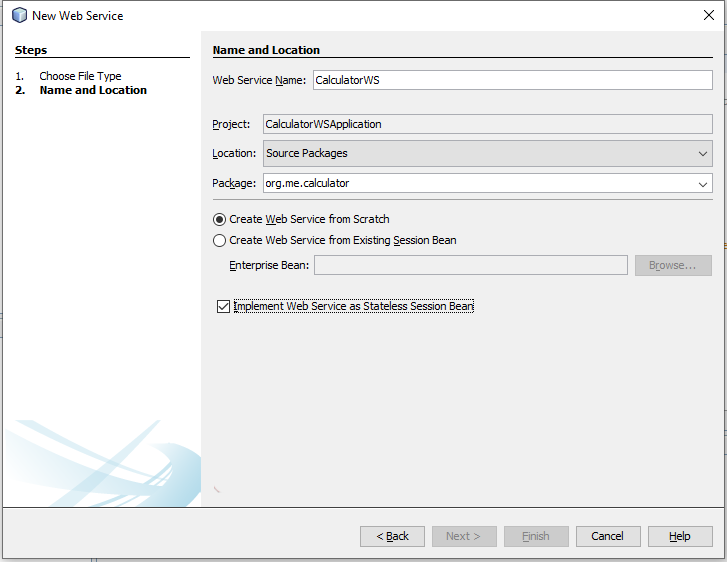
1. **Create  A Webservice to do add and subtract operation.and Consume webservice in**
2. **Choosing a Container**
3. **Choose File > New Project (Ctrl + Shift + N). Select Web Application from the Java Web.**
4. **Name the project CalculatorWSApplication. Select a location for the project. Click Next.**

****

1. **Select your server and Java EE version and click Finish.**
2. **Creating a Web Service from a Java Class**
3. **Right-click the CalculatorWSApplication node and choose New > Web Service**



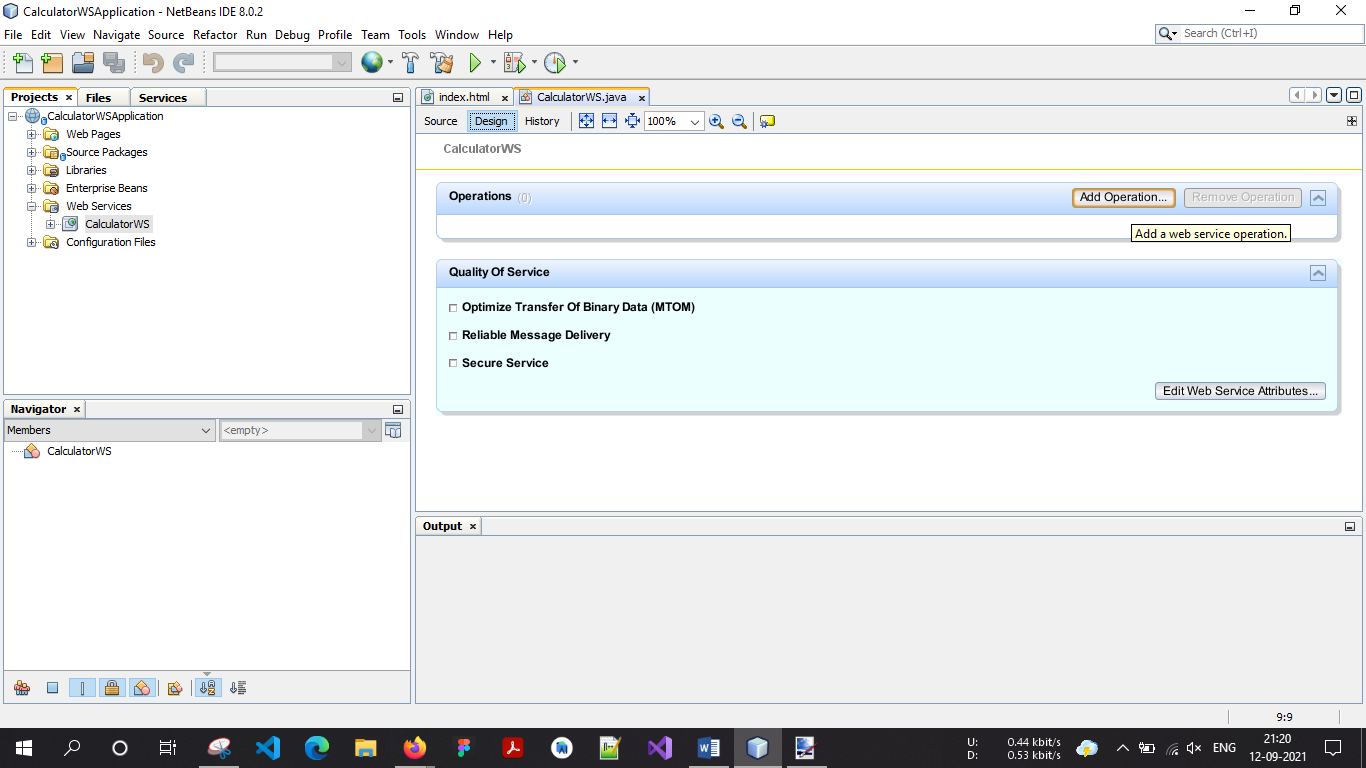
1. **Name the web service CalculatorWS and type org.me.calculator in Package.**
2. **Leave Create Web Service from Scratch selected. If you are creating a Java EE 6 project on GlassFish,**
3. **Select Implement Web Service as a Stateless Session Bean.**

****

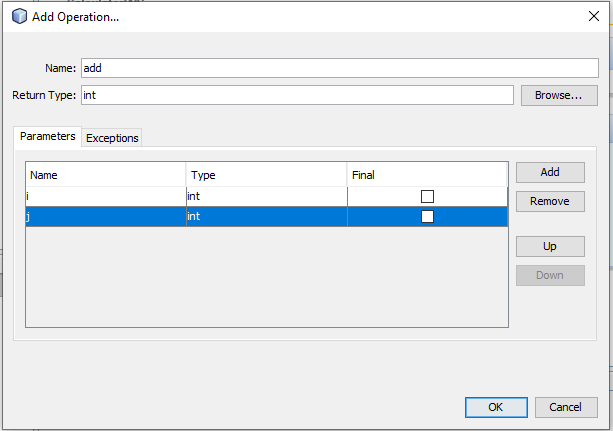
1. **Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.**
2. **Adding an Operation to the Web Service**

The goal of this exercise is to add to the web service an operation that adds two numbers received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service context menu.

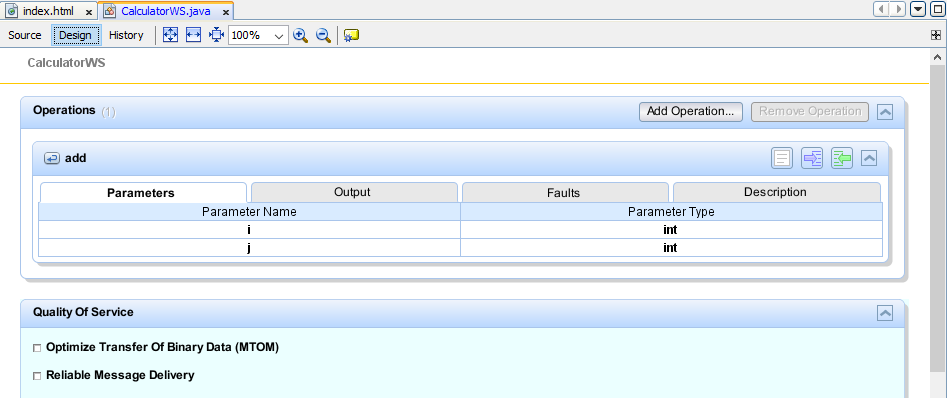
1. **To add an operation to the web service:**
2. Change to the Design view in the editor.



1. Click **Add Operation** in either the visual designer or the context menu. The Add Operation dialog opens.
2. In the upper part of the Add Operation dialog box, type ***add*** in Name and type ***int*** in the Return Type drop-down list.
3. In the lower part of the Add Operation dialog box, click **Add** and *create a parameter* of type **int** named ***i***.
4. Click Add again and create a parameter of type **int** called ***j***. You now see the following:



1. Click OK at the bottom of the Add Operation dialog box. You return to the editor.
2. The visual designer now displays the following:



1. Click Source. And code the following.

/\*\*

     \* Web service operation

     \*/

    @WebMethod(operationName = "add")

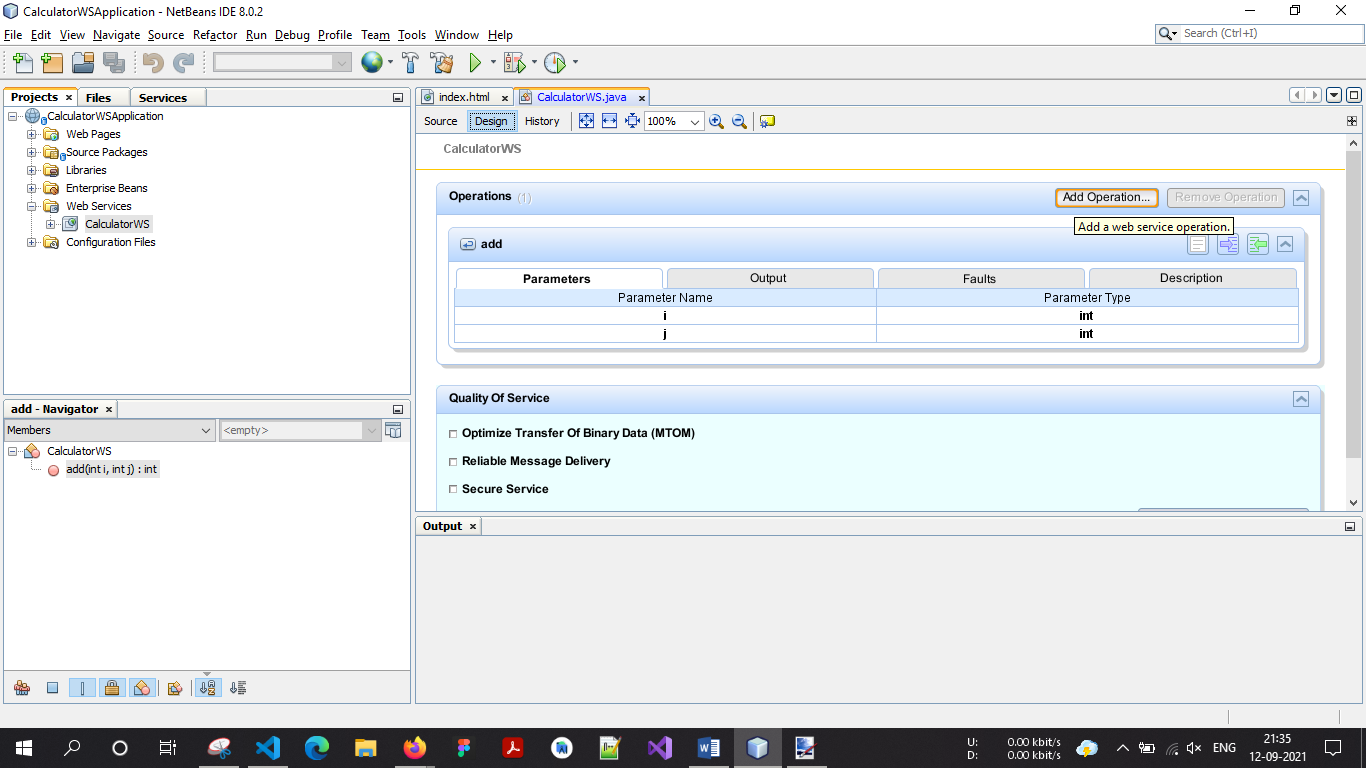
    public int add(@WebParam(name = "i") int i, @WebParam(name = "j") int j) {

        int k = i + j;

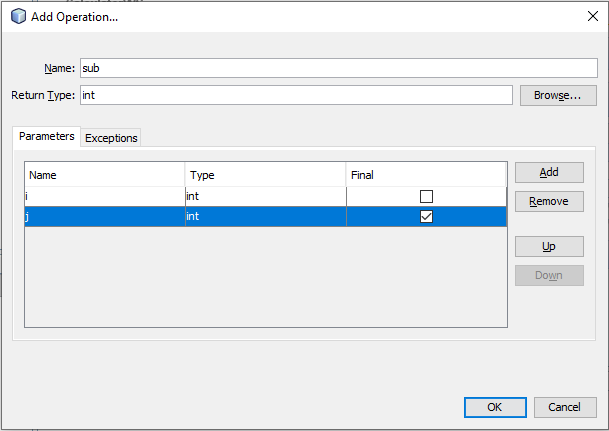
        return k;

    }

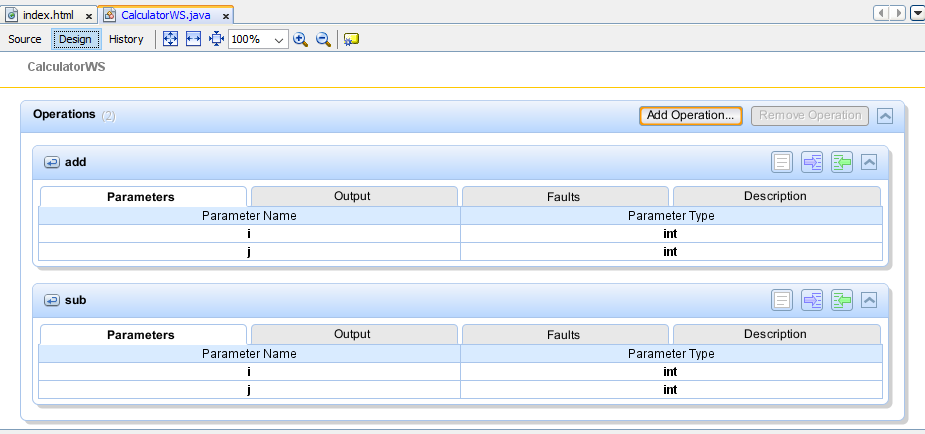
1. **To sub an operation to the web service:**
2. Change to the Design view in the editor.



1. Click **Add Operation** in either the visual designer or the context menu. The Add Operation dialog opens.
2. In the upper part of the Add Operation dialog box, type ***sub*** in Name and type ***int*** in the Return Type drop-down list.
3. In the lower part of the Add Operation dialog box, click **Add** and *create a parameter* of type **int** named ***i***.
4. Click Add again and create a parameter of type **int** called ***j***. You now see the following:



1. Click OK at the bottom of the Add Operation dialog box. You return to the editor.
2. The visual designer now displays the following:



1. Click Source. And code the following.

/\*\*

     \* Web service operation

     \*/

    @WebMethod(operationName = "sub")

    public int sub(@WebParam(name = "i") int i, @WebParam(name = "j") final int j) {

        int k = i - j;

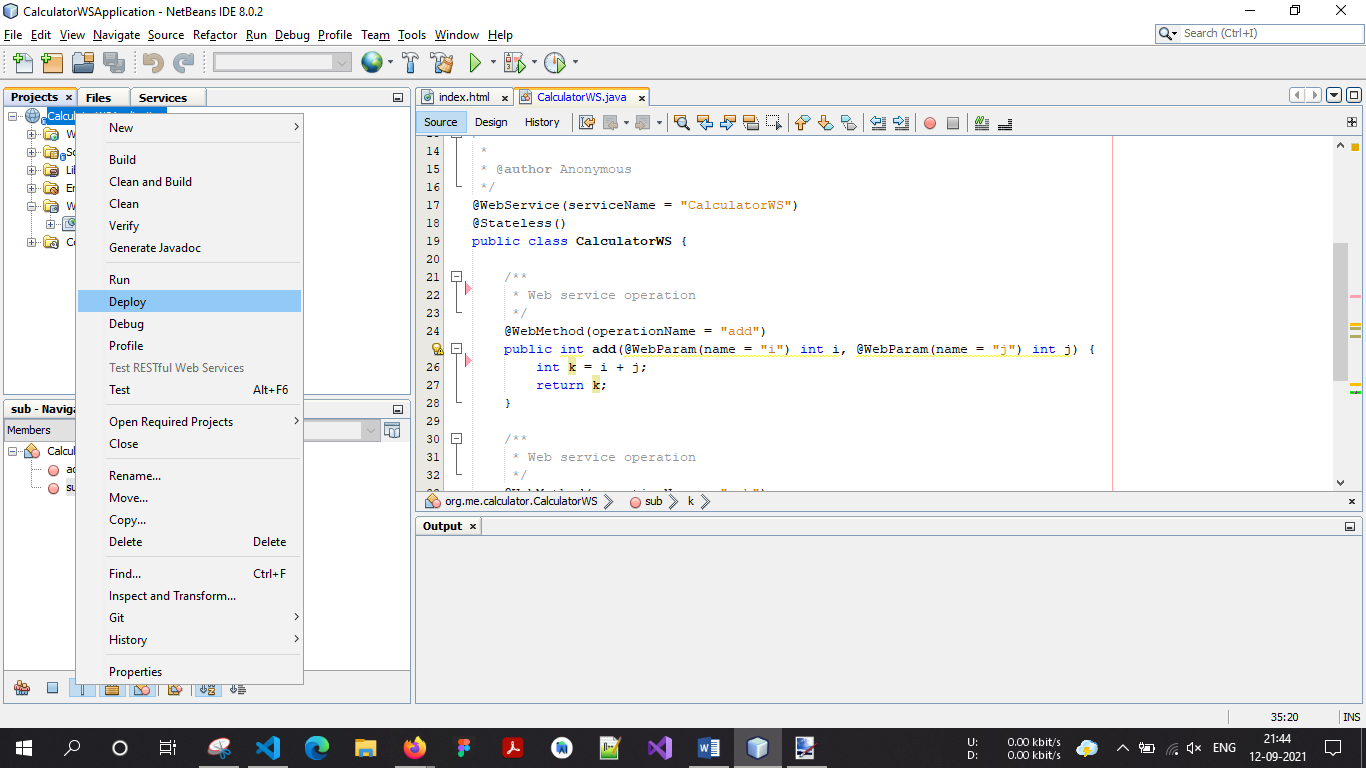
        return k;

    }

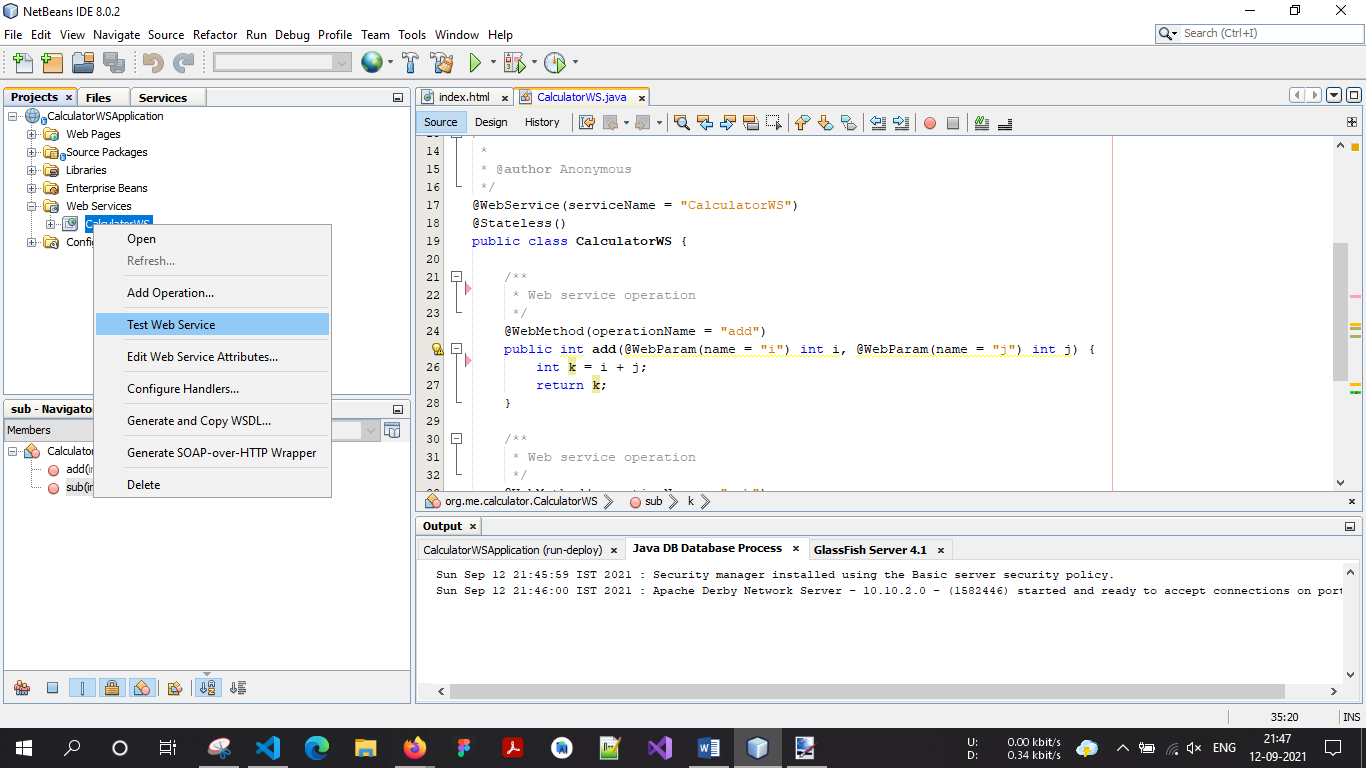
1. **Deploying and Testing the Web Service**

After you deploy a web service to a server, you can use the IDE to open the server's test client, if the server has a test client. The GlassFish servers provide test clients.

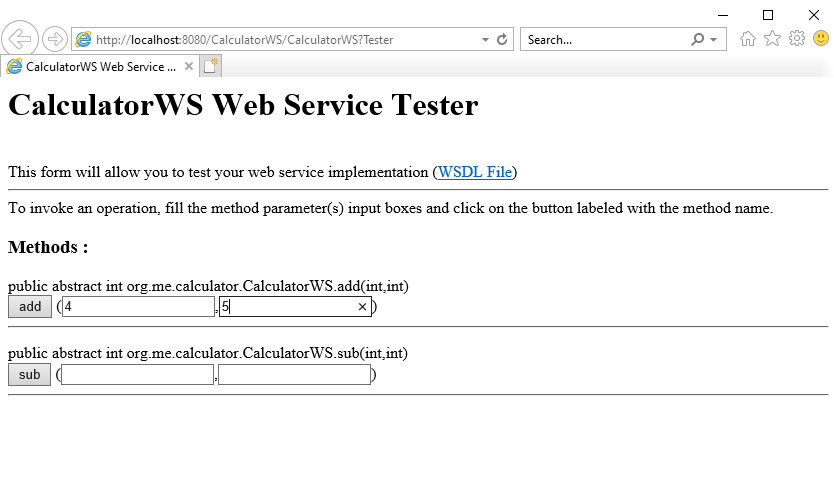
1. **To test successful deployment to a GlassFish server:**
2. Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server



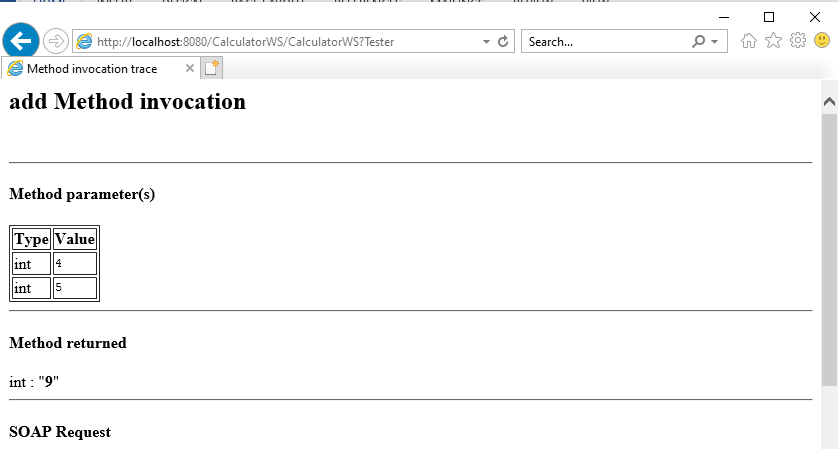
1. In the IDE's Projects tab, expand the Web Services node of the CalculatorWSApplication project. Right-click the CalculatorWS node, and choose Test Web Service



1. The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server.
2. If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:



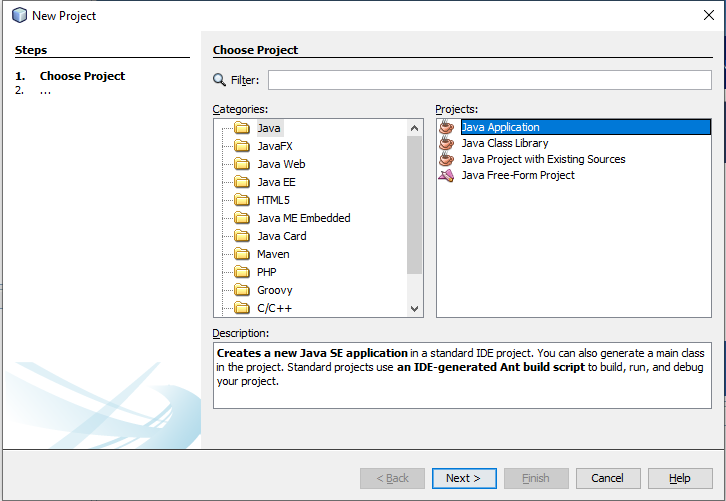
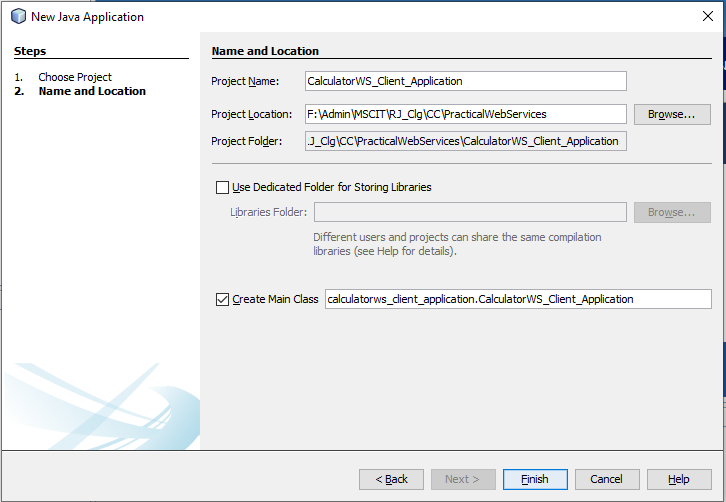
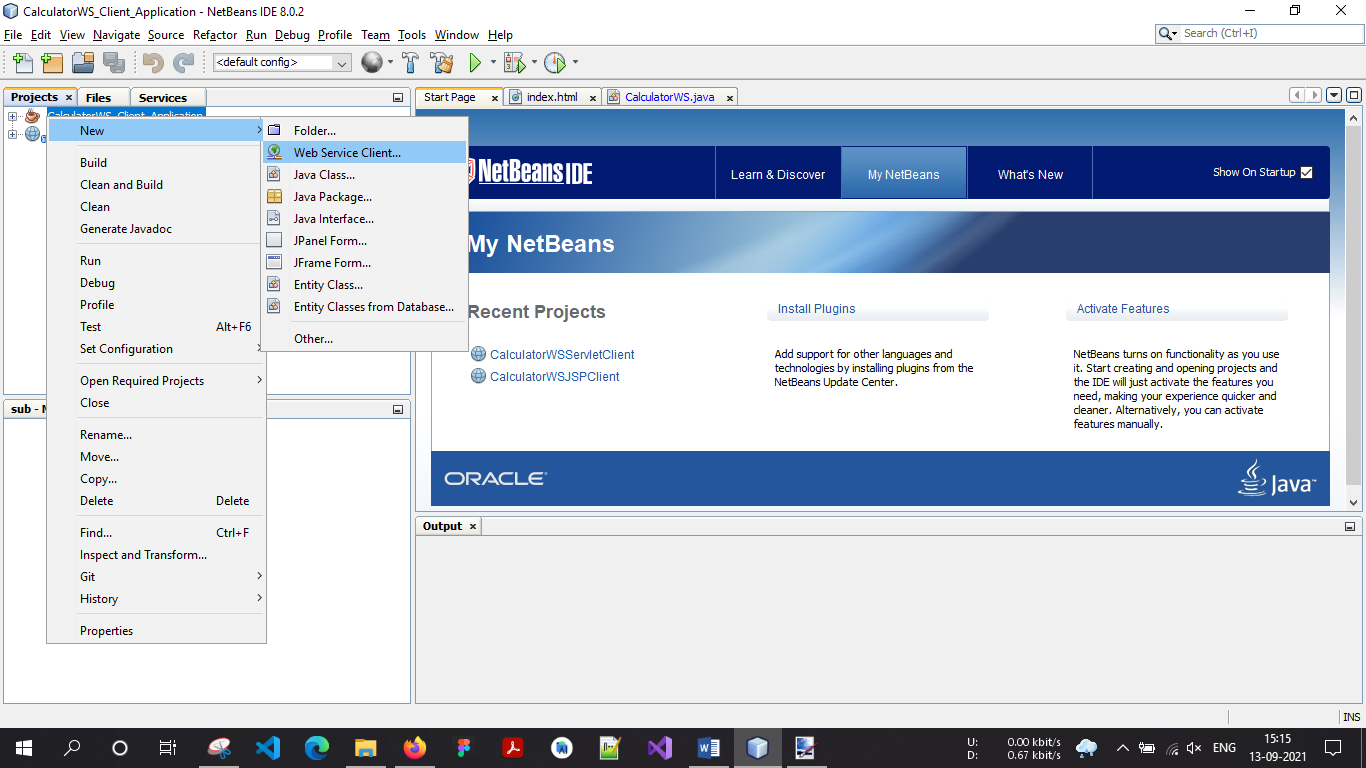
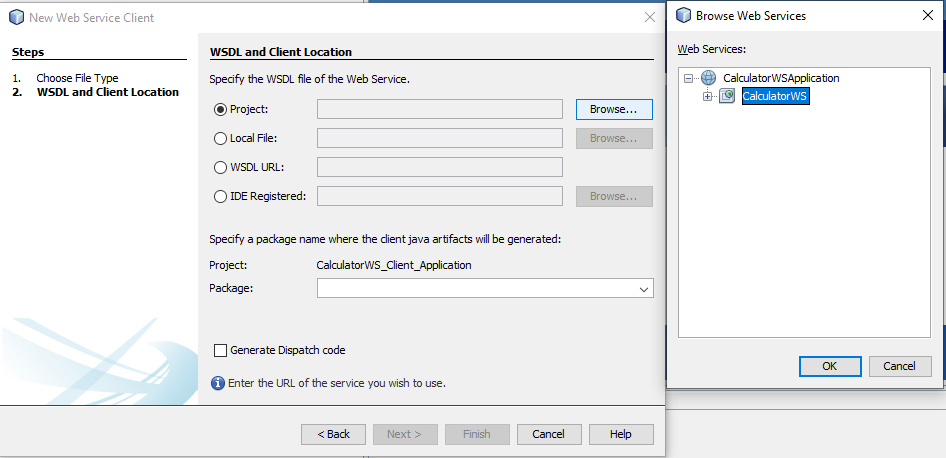
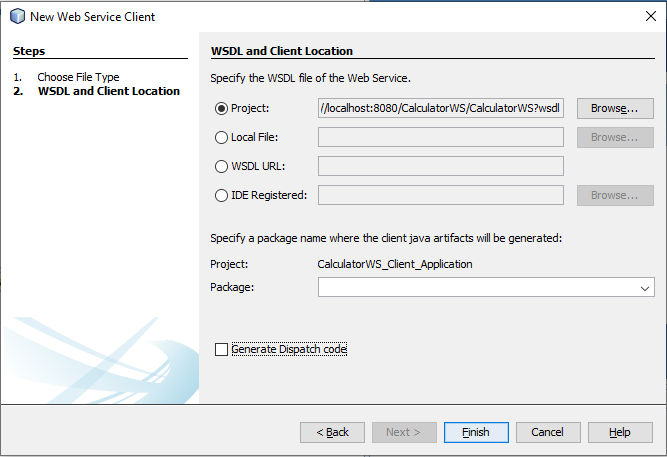
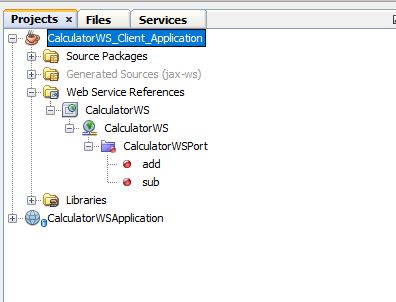
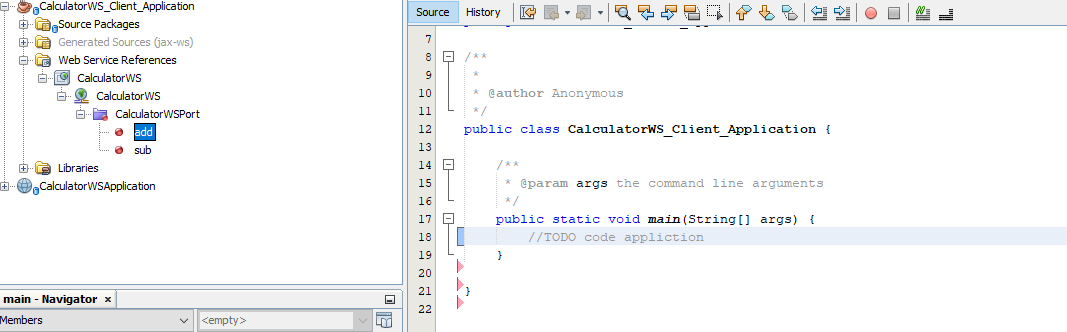
1. The sum of the two numbers is displayed:



1. Same as for sub operation insert two integer value on input box then show the result.

**Consuming the Web Service**

Now that you have deployed the web service, you need to create a client to make use of the web service's add method and sub method.

1. **Client-1: Java Class in Java SE Application**
2. Choose File > New Project. Select Java Application from the Java category.
3. Name the project CalculatorWS\_Client\_Application.
4. Leave Create Main Class selected and accept all other default settings. Click Finish.
5. **Right-click the CalculatorWS\_Client\_Application node and choose New > Web Service Client. The New Web Service Client wizard opens**
6. **Select Project as the WSDL source. Click Browse. Browse to the CalculatorWS web service in the CalculatorWSApplication project. When you have selected the web service, click OK.**
7. ** Do not select a package name. Leave this field empty.**
8. **Leave the other settings at default and click Finish. The Projects window displays the new web service client, with a node for the add method that you created:**
9. **Double-click your main class so that it opens in the Source Editor. Drag the add node below the main() method (or ) you can right-click in the editor and then choose Insert Code > Call Web Service Operation.**

**You now see the following code getting added on dragging:**

/\*\*

     \* @param args the command line arguments

     \*/

    public static void main(String[] args) {

        //TODO code appliction

    }

    private static int add(int i, int j) {

        org.me.calculator.CalculatorWS\_Service service = new org.me.calculator.CalculatorWS\_Service();

        org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

        return port.add(i, j);

    }

    private static int sub(int i, int j) {

        org.me.calculator.CalculatorWS\_Service service = new org.me.calculator.CalculatorWS\_Service();

        org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

        return port.sub(i, j);

    }

1. **In the main() method body, replace the TODO comment with code that initializes values for i and j, calls add() and sub(), and prints the result.**

public static void main(String[] args) {

        try{

            // Addition

            int a1 = 3;

            int a2 = 4;

            int addResult = add(a1, a2);

            System.out.println("Addition of " + a1 + " and " + a2 + " is " + addResult);

            // Substraction

            int s1 = 5;

            int s2 = 4;

            int subResult = sub(s1, s2);

            System.out.println("Substraction of "+ s1 + " and "+ s2 + " is " + subResult);

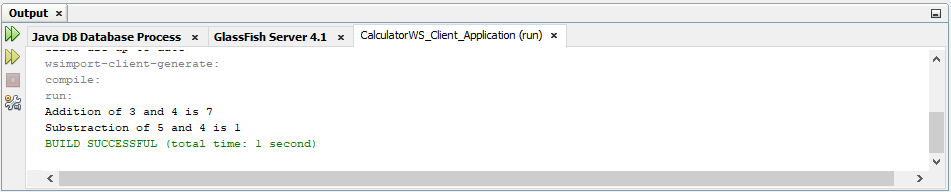
        }catch(Exception ex) {

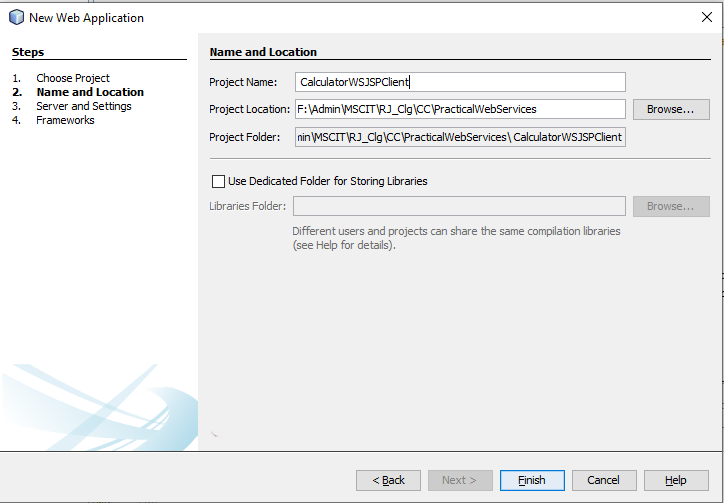
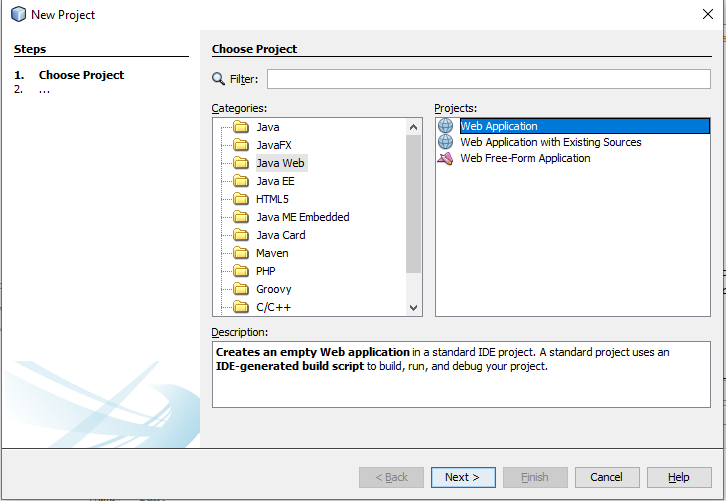
            System.out.println("Exception: " + ex);

        }

    }

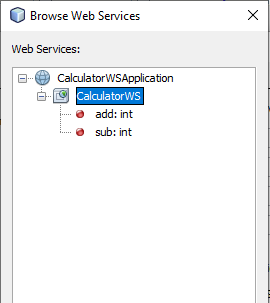
1. **Right-click the project node and choose Run.**

**The Output window now shows the sum:**

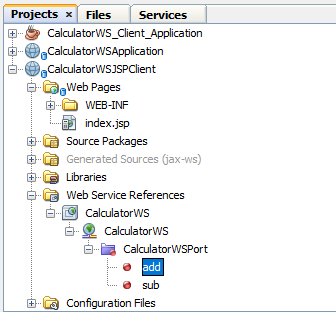
1. **Client-2: JSP Page in Web Application**
2. Choose File > New Project .
3. Select Web Application from the Java Web category. Name the project CalculatorWSJSPClient . Click Next and then click Finish.
4. Expand the Web Pages node under the project node and delete index.html .
5. Right-click the Web Pages node and choose New > JSP in the popup menu.

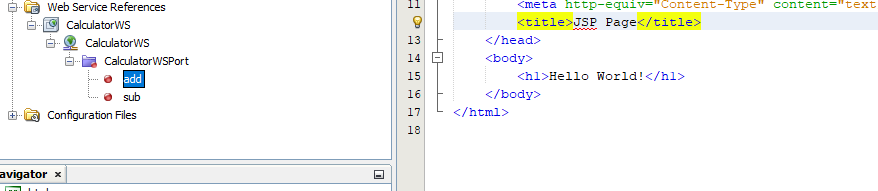
**Note:** If JSP is not available in the popup menu, choose New > Other and select JSP in the Web category of the New File wizard.

1. Type index for the name of the JSP file in the New File wizard. Click Finish.
2. Right-click the CalculatorWSJSPClient node and choose New > Web Service Client.
3. Select Project as the WSDL source.
4. Click Browse. Browse to the CalculatorWS web service in the CalculatorWSApplication project. When you have selected the web service, click OK.



1. Do not select a package name. Leave this field empty.
2. Leave the other settings at default and click Finish.

The Projects window displays the new web service client, as shown below:

1. In the Web Service References node, expand the node that represents the web service. The add operation, which you will invoke from the client, is now exposed.
2. Drag the add operation to the client’s index.jsp page, and drop it below the H1 tags.
3. The code for invoking the service’s operation is now generated in the index.jsp page, as you can see here:

<%-- start web service invocation --%><hr/>

    <%

    try {

    org.me.calculator.CalculatorWS\_Service service = new org.me.calculator.CalculatorWS\_Service();

    org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

     // TODO initialize WS operation arguments here

    int i = 0;

    int j = 0;

    // TODO process result here

    int result = port.add(i, j);

    out.println("Result = "+result);

    } catch (Exception ex) {

    // TODO handle custom exceptions here

    }

    %>

    <%-- end web service invocation --%><hr/>

    <%-- start web service invocation --%><hr/>

    <%

    try {

    org.me.calculator.CalculatorWS\_Service service = new org.me.calculator.CalculatorWS\_Service();

    org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

     // TODO initialize WS operation arguments here

    int i = 0;

    int j = 0;

    // TODO process result here

    int result = port.sub(i, j);

    out.println("Result = "+result);

    } catch (Exception ex) {

    // TODO handle custom exceptions here

    }

    %>

    <%-- end web service invocation --%><hr/>

Make the Change the value for i and j from 0 to other integers, such as 3 and 4.

Replace the commented out TODO line in the catch block with

out.println("exception" + ex);

<%-- start web service invocation --%><hr/>

    <%

    try {

    org.me.calculator.CalculatorWS\_Service service = new org.me.calculator.CalculatorWS\_Service();

    org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

     // TODO initialize WS operation arguments here

    int i = 3;

    int j = 4;

    // TODO process result here

    int result = port.add(i, j);

    out.println("Addition Result = "+result);

    } catch (Exception ex) {

    // TODO handle custom exceptions here

    }

    %>

    <%-- end web service invocation --%><hr/>

    <%-- start web service invocation --%><hr/>

    <%

    try {

    org.me.calculator.CalculatorWS\_Service service = new org.me.calculator.CalculatorWS\_Service();

    org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

     // TODO initialize WS operation arguments here

    int i = 5;

    int j = 4;

    // TODO process result here

    int result = port.sub(i, j);

    out.println("Substraction Result = "+result);

    } catch (Exception ex) {

    // TODO handle custom exceptions here

    }

    %>

    <%-- end web service invocation --%><hr/>

1. Right-click the project node and choose Run.

The server starts, if it wasn’t running already. The application is built and deployed, and the browser opens, displaying the calculation result: