

Lab 7

Introduction to Web Services

Introduction to WSDL

WSDL stands for Web Services Description Language.
WSDL is a language for describing web services and how to access them.
WSDL is written in XML.

What is WSDL?

- WSDL stands for Web Services Description Language
- WSDL is written in XML
- WSDL is an XML document
- WSDL is used to describe Web services
- WSDL is also used to locate Web services
- WSDL is a W3C recommendation

WSDL Describes Web Services

WSDL stands for Web Services Description Language.

WSDL is a document written in XML. The document describes a Web service. It specifies the location of the service and the operations (or methods) the service exposes.

A WSDL document is just a simple XML document.

It contains set of definitions to describe a web service.

The WSDL Document Structure

A WSDL document describes a web service using these major elements:

Element	Description
<types>	A container for data type definitions used by the web service
<message>	A typed definition of the data being communicated
<portType>	A set of operations supported by one or more endpoints
<binding>	A protocol and data format specification for a particular port type

The main structure of a WSDL document looks like this:

```
<definitions>

<types>
  data type definitions.....
</types>

<message>
  definition of the data being communicated....
</message>

<portType>
  set of operations.....
</portType>

<binding>
  protocol and data format specification....
</binding>

</definitions>
```

A WSDL document can also contain other elements, like extension elements, and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

WSDL Ports

The **<portType>** element is the most important WSDL element.

It describes a web service, the operations that can be performed, and the messages that are involved.

The <portType> element can be compared to a function library (or a module, or a class) in a traditional programming language.

WSDL Messages

The **<message>** element defines the data elements of an operation.

Each message can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

WSDL Types

The **<types>** element defines the data types that are used by the web service.

For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL Bindings

The **<binding>** element defines the data format and protocol for each port type.

WSDL Example

This is a simplified fraction of a WSDL document:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

In this example the **<portType>** element defines "glossaryTerms" as the name of a **port**, and "getTerm" as the name of an **operation**.

The "getTerm" operation has an **input message** called "getTermRequest" and an **output message** called "getTermResponse".

The **<message>** elements define the **parts** of each message and the associated data types.

Compared to traditional programming, glossaryTerms is a function library, "getTerm" is a function with "getTermRequest" as the input parameter, and getTermResponse as the return parameter.

Where to get WSDL for free web service?

- 1) <http://www.webservices.net/ws/default.aspx>
- 2) <http://www.service-repository.com/>

Exercise Requirements:

1. You'll need NetBeans IDE EE version with GlassFish
2. For Exercise 1 you need internet connection

How to create a client project to test web services? (Ex. 1)

Using NetBeans IDE, the steps are:

- 1) Create new java application
- 2) Right click on the project>new>others>Web Services>Web Service Client
- 3) Click next and choose WSDL URL and paste the URL of the WSDL file for the web service which you intend to use for this tutorial you can use the weather web service "<http://www.webservices.net/globalweather.asmx?WSDL>".
- 4) Now open the source packages and click on the class you can find.
- 5) Click on Web Service References>globalweather>GlobalWeather> GlobalWeatherSoap
- 6) Click and drag the method "Get Weather" under your class main method
- 7) Write the following lines of code in your main method to test the web service

```
String city = "Cairo";
String country = "Egypt";
String weathernow = getWeather(city, country);
System.out.println("The Weather details are:");
System.out.println(weathernow);
```
- 8) Run the web service and watch the result in the console.

You could also check out the following video describing the procedure as step by step
<http://www.youtube.com/watch?v=xzgrLLN4ILM>

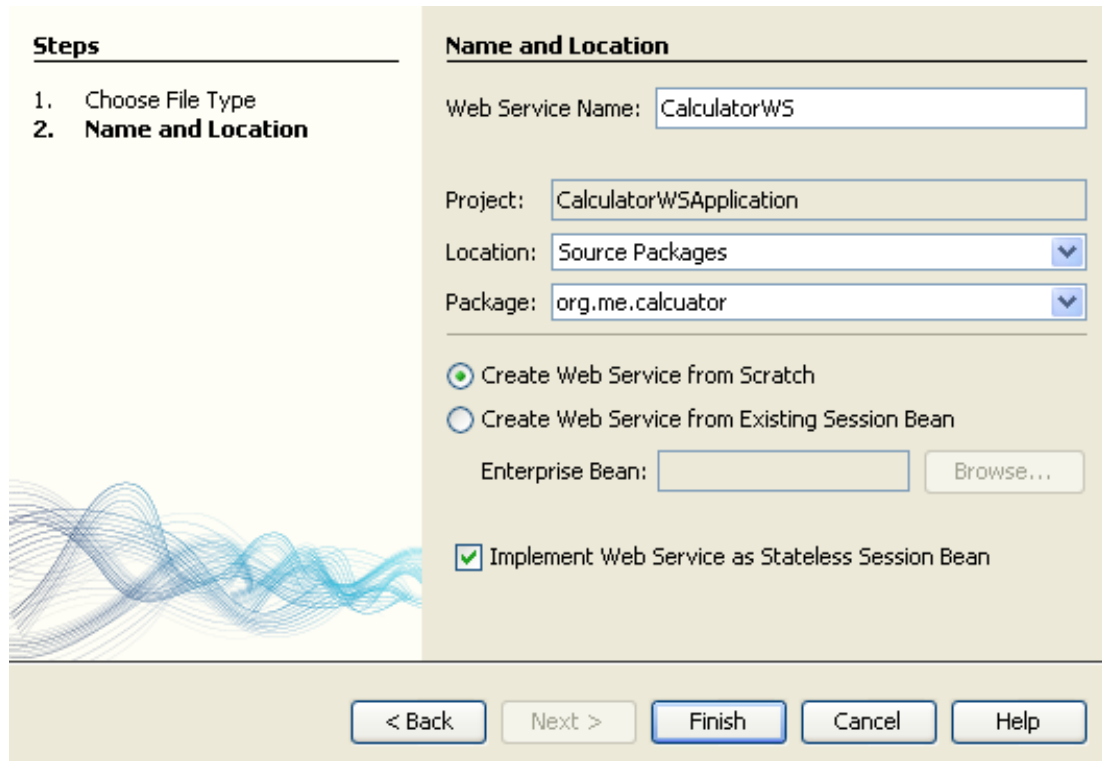
How to create the web service itself? (Ex. 2)

Creating a Web Service

1. Choose File > New Project (Ctrl-Shift-N on Linux and Windows, ⌘-Shift-N on MacOS). Select Web Application from the Java Web category.
2. Name the project CalculatorWSApplication. Select a location for the project. Click Next.
3. Select your server (Glass Fish) and Java EE version and click Finish.

Creating a Web Service from a Java Class

1. Right-click the CalculatorWSApplication node and choose New > Web Service.
2. Name the web service CalculatorWS and type org.me.calculator in Package. Leave Create Web Service from Scratch selected.
3. If you are creating a Java EE project on GlassFish or WebLogic, select Implement Web Service as a Stateless Session Bean.



The image shows a NetBeans IDE dialog box titled "Name and Location". On the left, a "Steps" panel lists: 1. Choose File Type, 2. **Name and Location**. The main area contains the following fields and options:

- Web Service Name:
- Project:
- Location: (with a dropdown arrow)
- Package: (with a dropdown arrow)
- Radio buttons for:
 - ☒ Create Web Service from Scratch
 - ☐ Create Web Service from Existing Session Bean
- Enterprise Bean: (with a "Browse..." button)
- ☒ Implement Web Service as Stateless Session Bean

At the bottom are buttons: "< Back", "Next >", "Finish", "Cancel", and "Help".

4. Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.

Adding an Operation to the Web Service

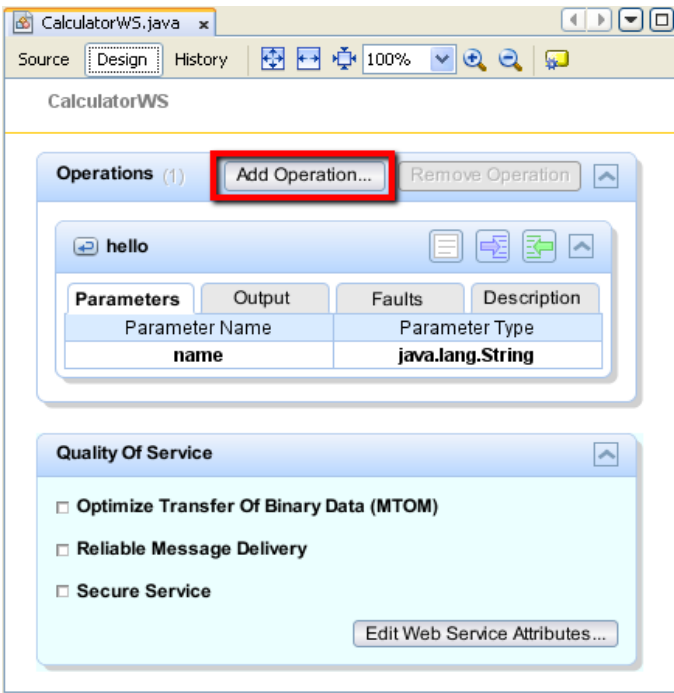
The goal of this exercise is to add to the web service an operation that adds two numbers received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service context menu.

Warning: The visual designer is not available in Maven projects.

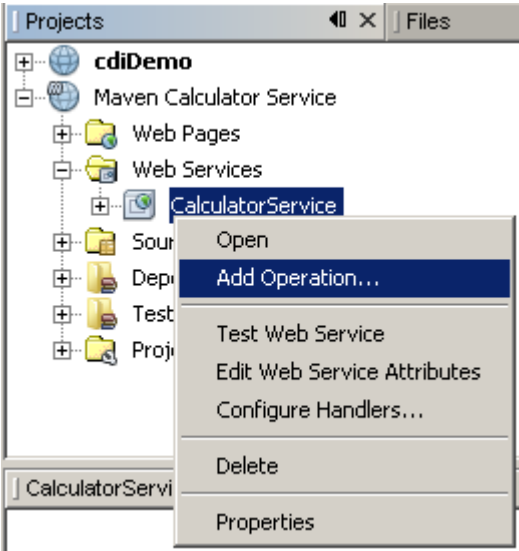
To add an operation to the web service:

1. 2 Options:

(1) Change to the Design view in the editor.

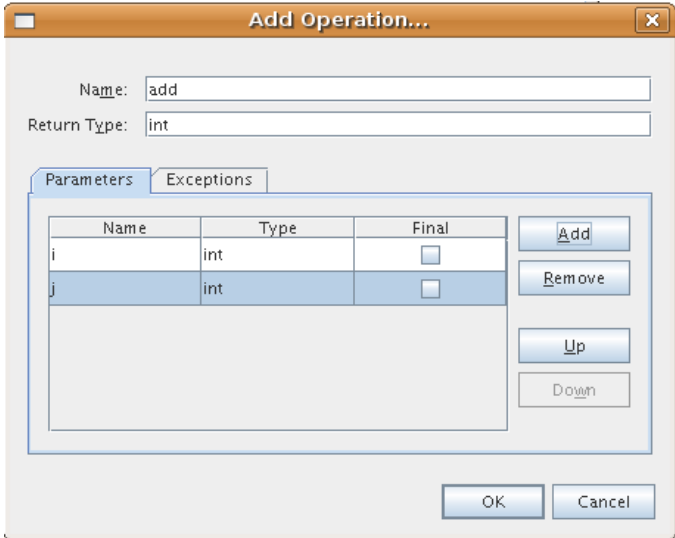


(2) Find the web service's node in the Projects window. Right-click that node. A context menu opens.



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

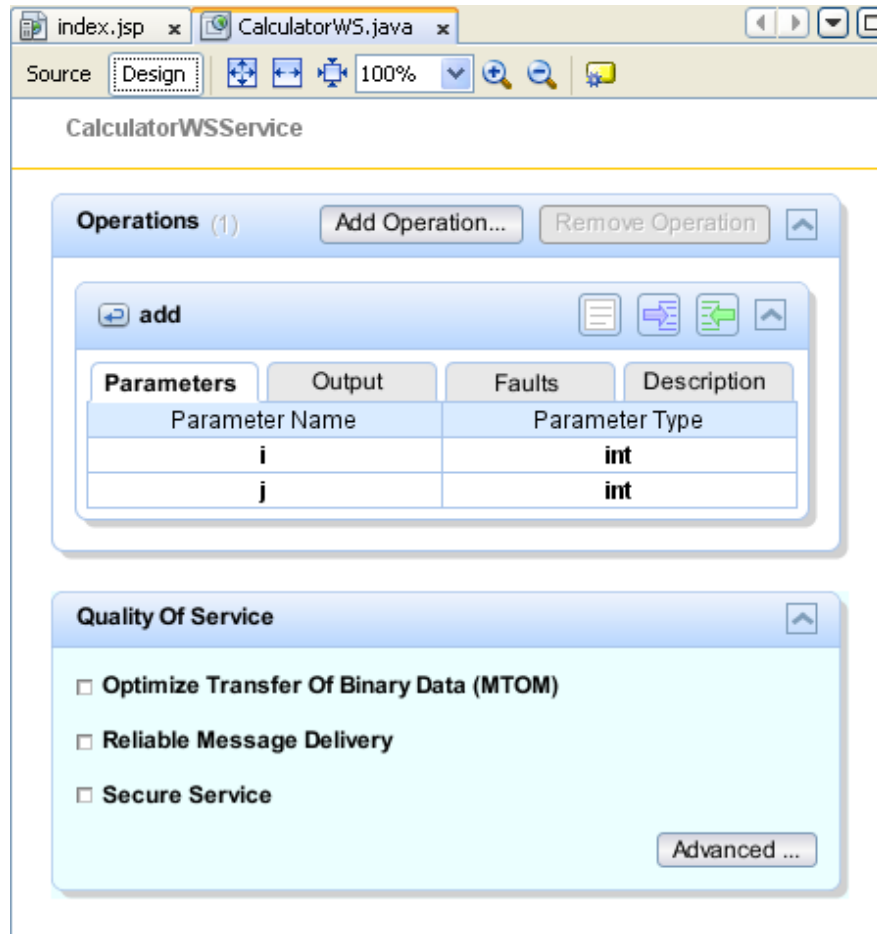
You now see the following:



Name	Type	Final
i	int	<input type="checkbox"/>
j	int	<input type="checkbox"/>

6. Click OK at the bottom of the Add Operation dialog box. You return to the editor.
7. Remove the default `hello` operation, either by deleting the `hello()` method in the source code or by selecting the `hello` operation in the visual designer and clicking Remove Operation.

The visual designer now displays the following:



8. Click Source and view the code that you generated in the previous steps. It differs whether you created the service as an Java EE stateless bean or not. Can you see the difference in the screenshots below? (A Java

EE 6 or Java EE 7 service that is not implemented as a stateless bean resembles a Java EE 5 service.)

```
package org.me.calculator;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.ejb.Stateless;

/**
 *
 * @author jeff
 */
@WebService()
@Stateless()
public class CalculatorWS {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "add")
    public int add(@WebParam(name = "i")
        int i, @WebParam(name = "j")
        int j) {
        //TODO write your implementation code
        return 0;
    }
}
```

Note. In NetBeans IDE 7.3 and 7.4 you will notice that in the generated `@WebService` annotation the service name is specified explicitly:

```
@WebService(serviceName = "CalculatorWS").
```

9. In the editor, extend the skeleton add operation to the following (changes are in bold):

```
10.     @WebMethod
11.     public int add(@WebParam(name = "i") int i, @WebParam(name = "j")
    int j) {
12.         int k = i + j;
13.         return k;
    }
```

As you can see from the preceding code, the web service simply receives two numbers and then returns their sum. In the next section, you use the IDE to test the web service.

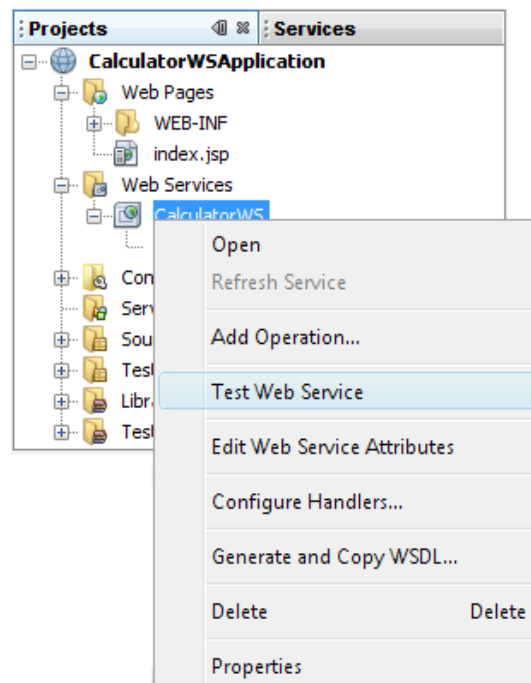
Deploying and Testing the Web Service

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 and WebLogic servers provide test clients.

If you are using the Tomcat Web Server, there is no test client. You can only run the project and see if the Tomcat Web Services page opens. In this case, before you run the project, you need to make the web service the entry point to your application. To make the web service the entry point to your application, right-click the CalculatorWSApplication project node and choose Properties. Open the Run properties and type /CalculatorWS in the Relative URL field. Click OK. To run the project, right-click the project node again and select Run.

To test successful deployment to a GlassFish or WebLogic server:

1. Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server. You can follow the progress of these operations in the CalculatorWSApplication (run-deploy) and the GlassFish server or Tomcat tabs in the Output view.
2. 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.



The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server. For the Tomcat Web Server and deployment of EJB modules, the situation is different:

- If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:

CalculatorWS Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

```
public abstract int org.netbeans.CalculatorWSProject.add(int,int)
```

(,)

- The sum of the two numbers is displayed:

add Method invocation

Method parameter(s)

Type	Value
int	2
int	3

Method returned

int : "5"

You can right click on the project and select "run"

Copy the WSDL URL from your browser and make a client like the previous exercise.