SOAP Web Services JAX-WS

Laboratory of Service Design and Engineering

2011/2012

JAX-WS 2.0

JAX-WS

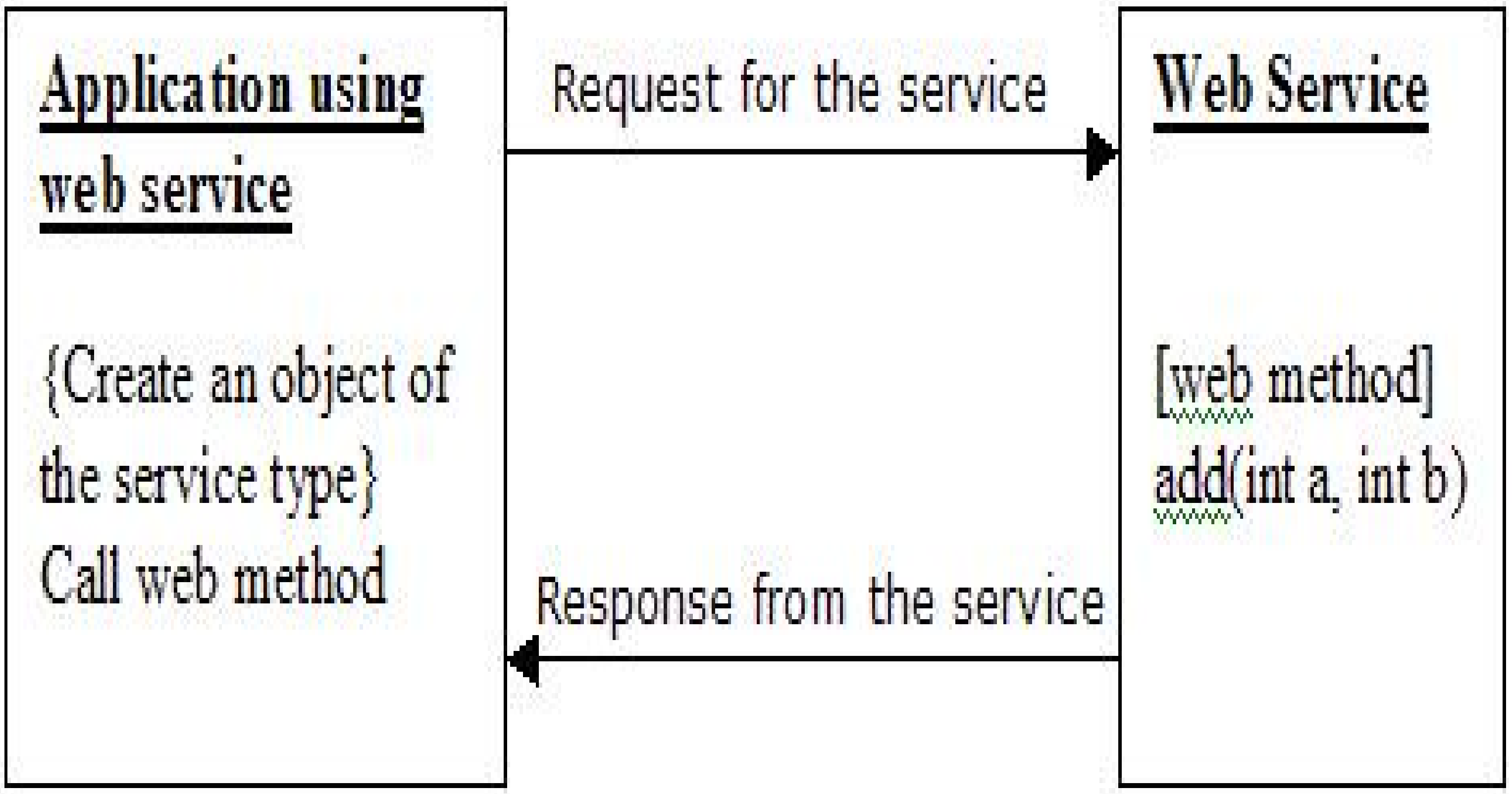
- Part of Java EE.
- New in Java SE 6.
- API stack for web services.
- Replaces JAX-RPC.
- New API's:

    JAX-WS, SAAJ, Web Service metadata

- New packages:

    javax.xml.ws, javax.xml.soap,javax.jws

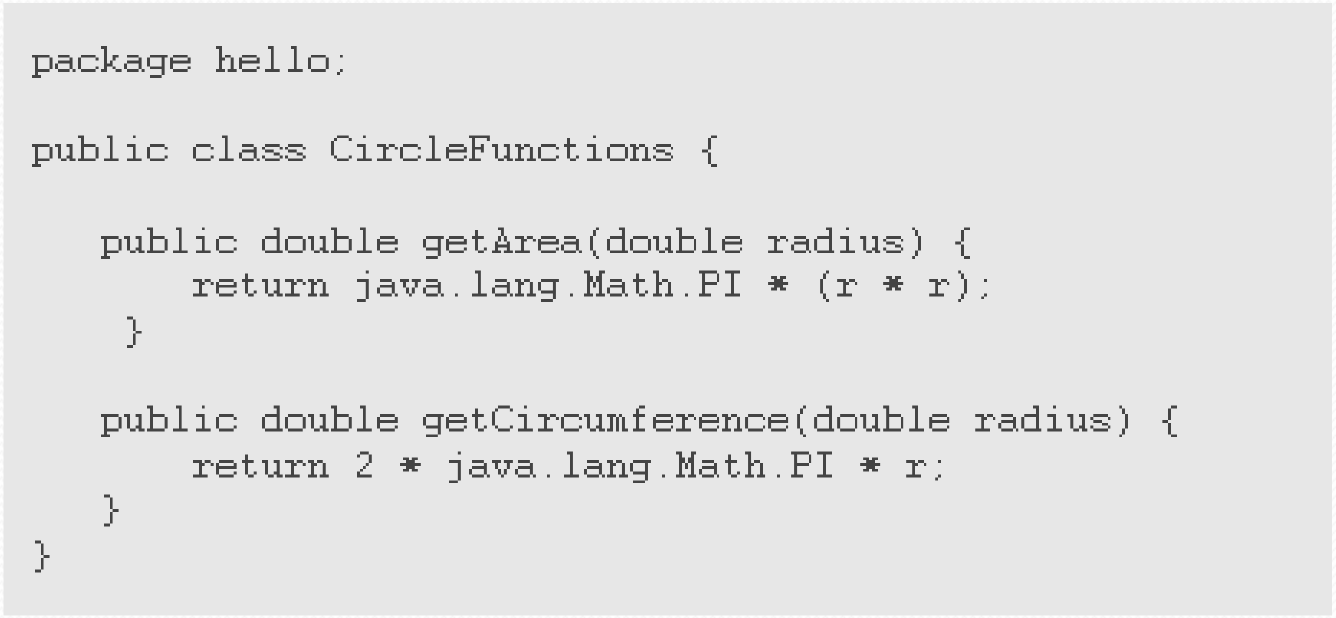
Web Service


Application using web service
{Create an object of the service type}
Call web method
Request for the service
Response from the service
Web Service
[web method]
add(int a, int b)

Using JAX-WS 2.0

- JAX-WS 2.0 is extremely easy to use
- We show you how to create a simple web service using JAX-WS 2.0 with Java SE 6 technology.
- The first thing you need is a class with one or more methods that you wish to expose as a web service.

Creating Web Service



```
package hello;

public class CircleFunctions {

   public double getArea(double radius) {
       return java.lang.Math.PI * (r * r);
    }

   public double getCircumference(double radius) {
       return 2 * java.lang.Math.PI * r;
   }
}
```

- To expose these methods, you must add two things:
  - an import statement javax.jws.WebService package
  - @WebService annotation at the beginning that tells the Java interpreter that you intend to publish the methods of this class as a web service.

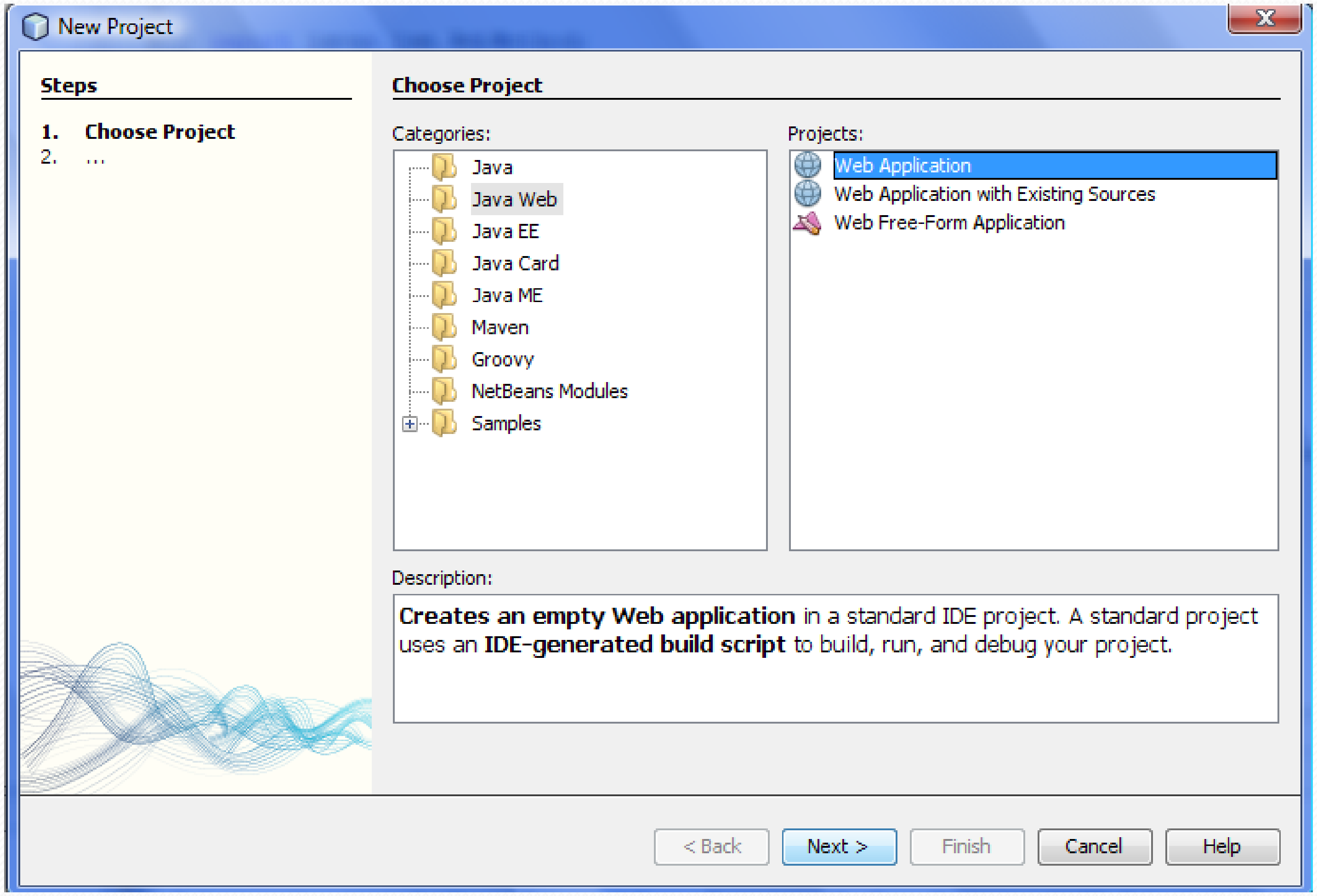
Creating Web Service

```
package hello;

import javax.jws.WebService;

@WebService

public class CircleFunctions {
   public double getArea(double r) {
       return java.lang.Math.PI * (r * r);
    }

   public double getCircumference(double r) {
        return 2 * java.lang.Math.PI * r;
    }

}
```

Creating Web Service

- With NetBeans 1.9.1
- JAX-WS 2.0
- Java SE 6
- GlassFish Server

Create New Web App



New Project

Steps

1. **Choose Project**
2. ...

Choose Project

Categories:

- Java
- Java Web
- Java EE
- Java Card
- Java ME
- Maven
- Groovy
- NetBeans Modules
- Samples

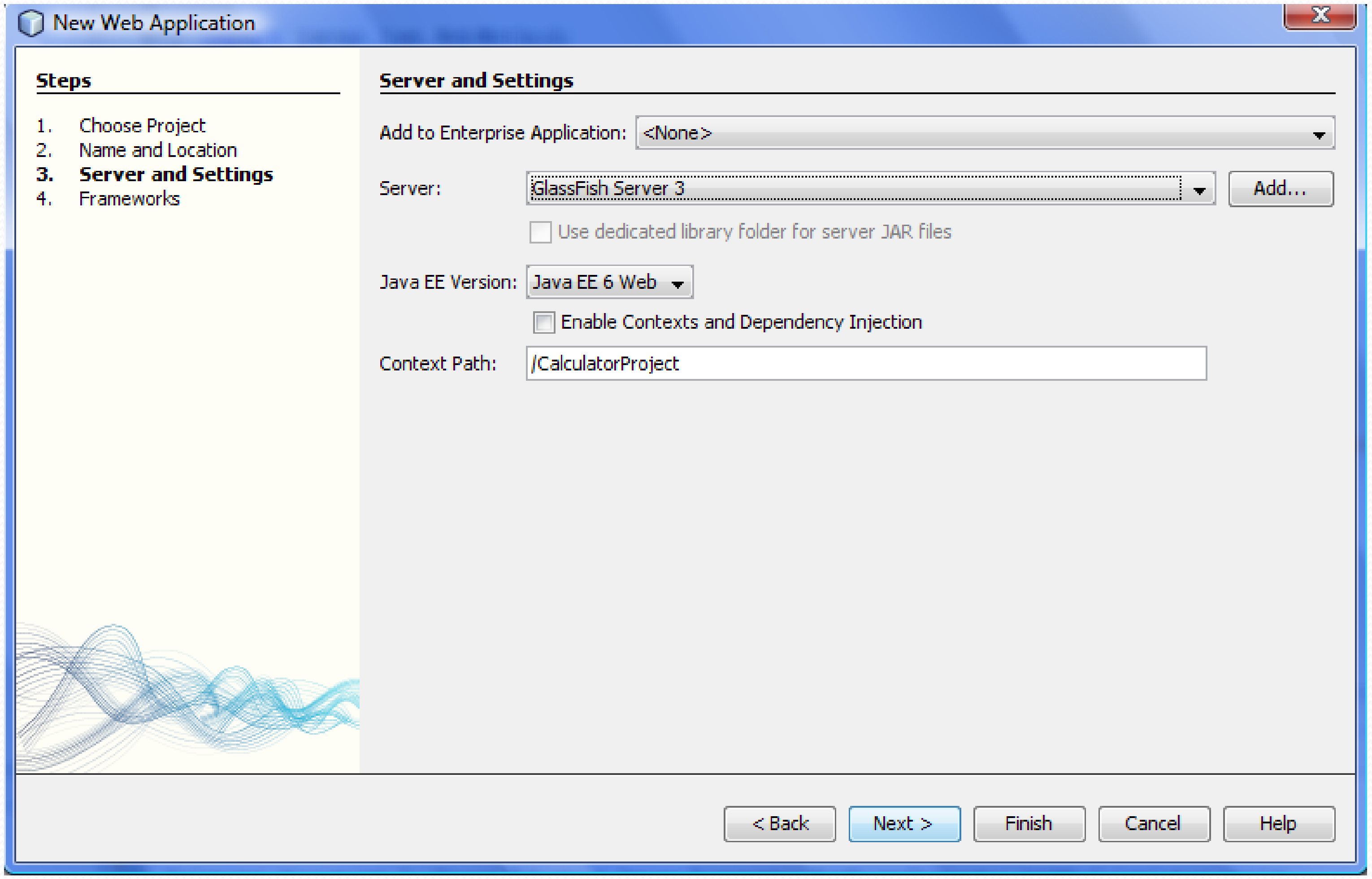
Projects:

- Web Application
- Web Application with Existing Sources
- Web Free-Form Application

Description:

Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

Set Context Path & Server



Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server: GlassFish Server 3 [Add...]

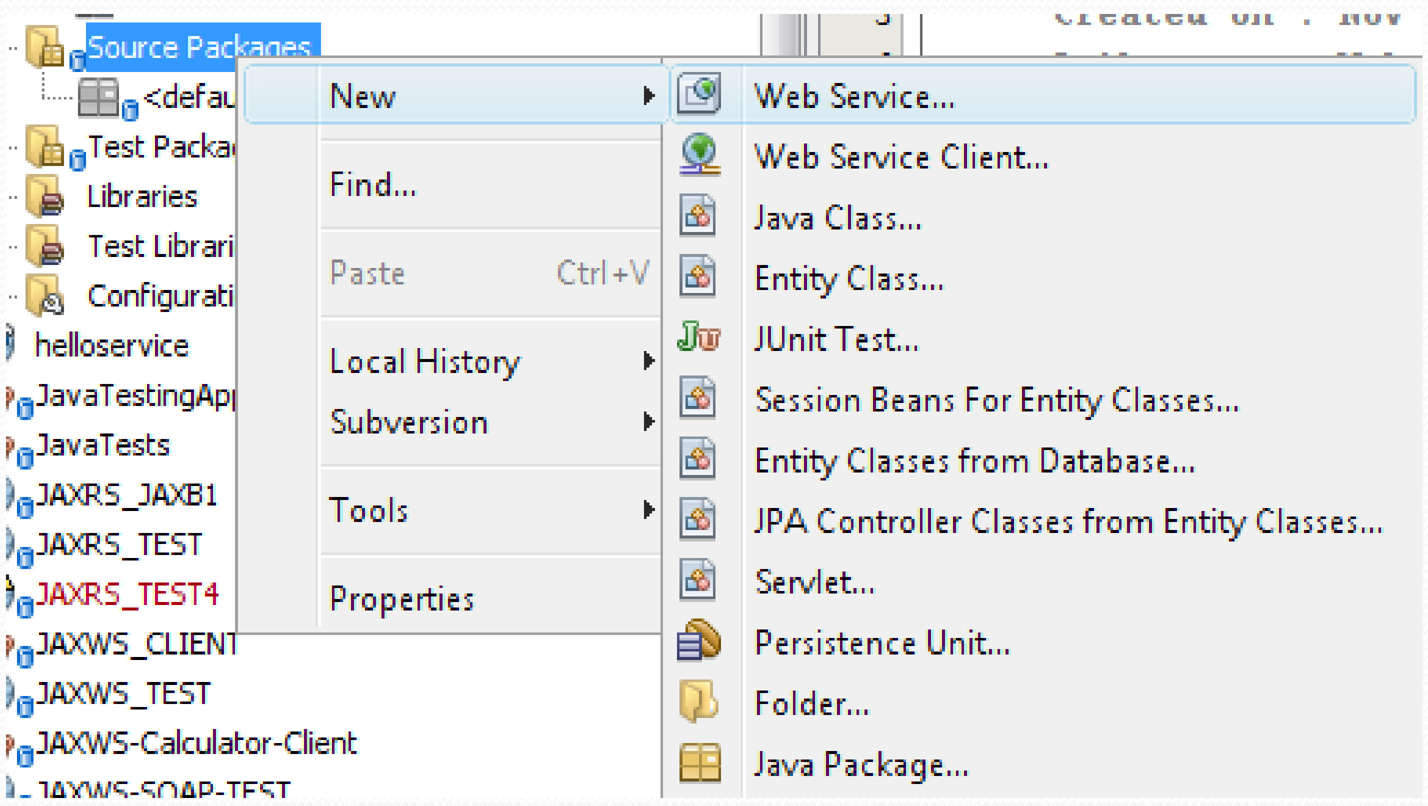
☐ Use dedicated library folder for server JAR files

Java EE Version: Java EE 6 Web

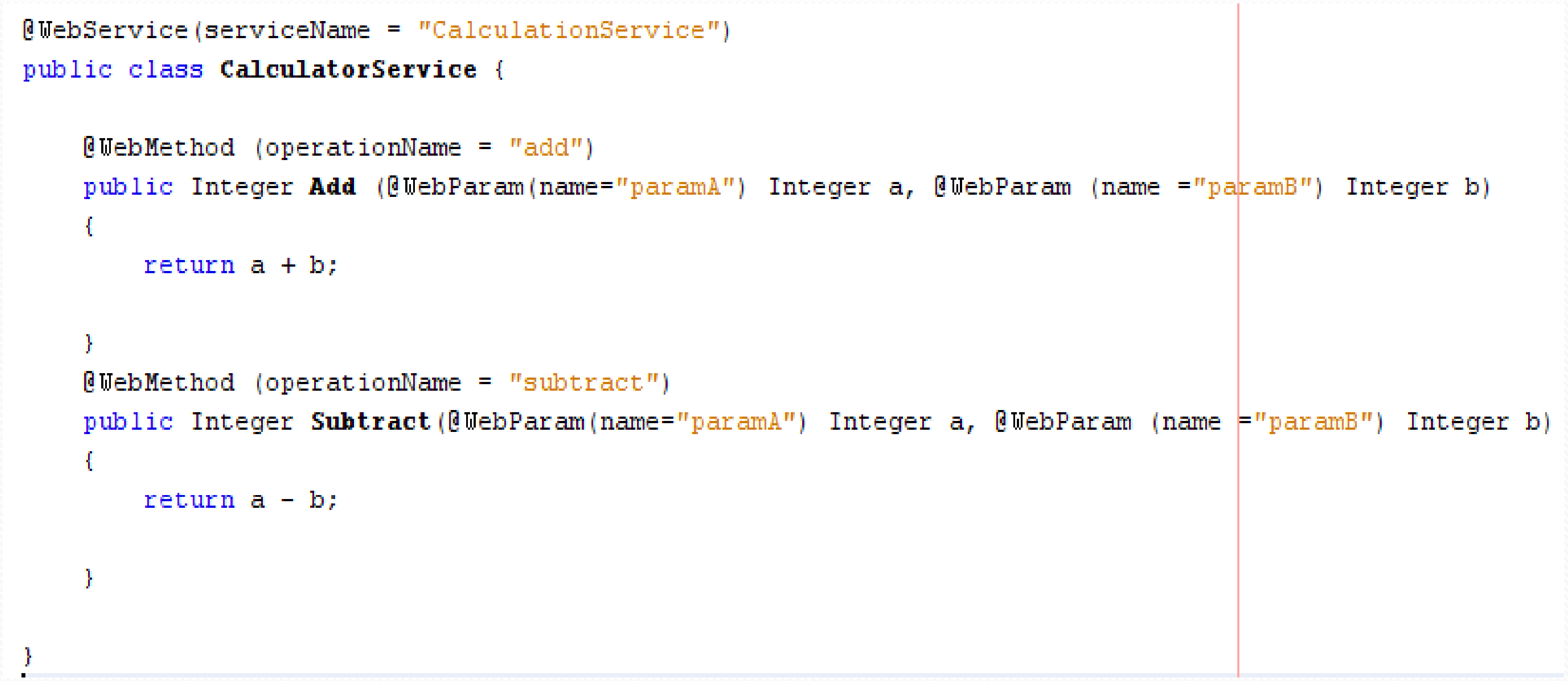
☐ Enable Contexts and Dependency Injection

Context Path: /CalculatorProject

Add new Web Service



Your Class

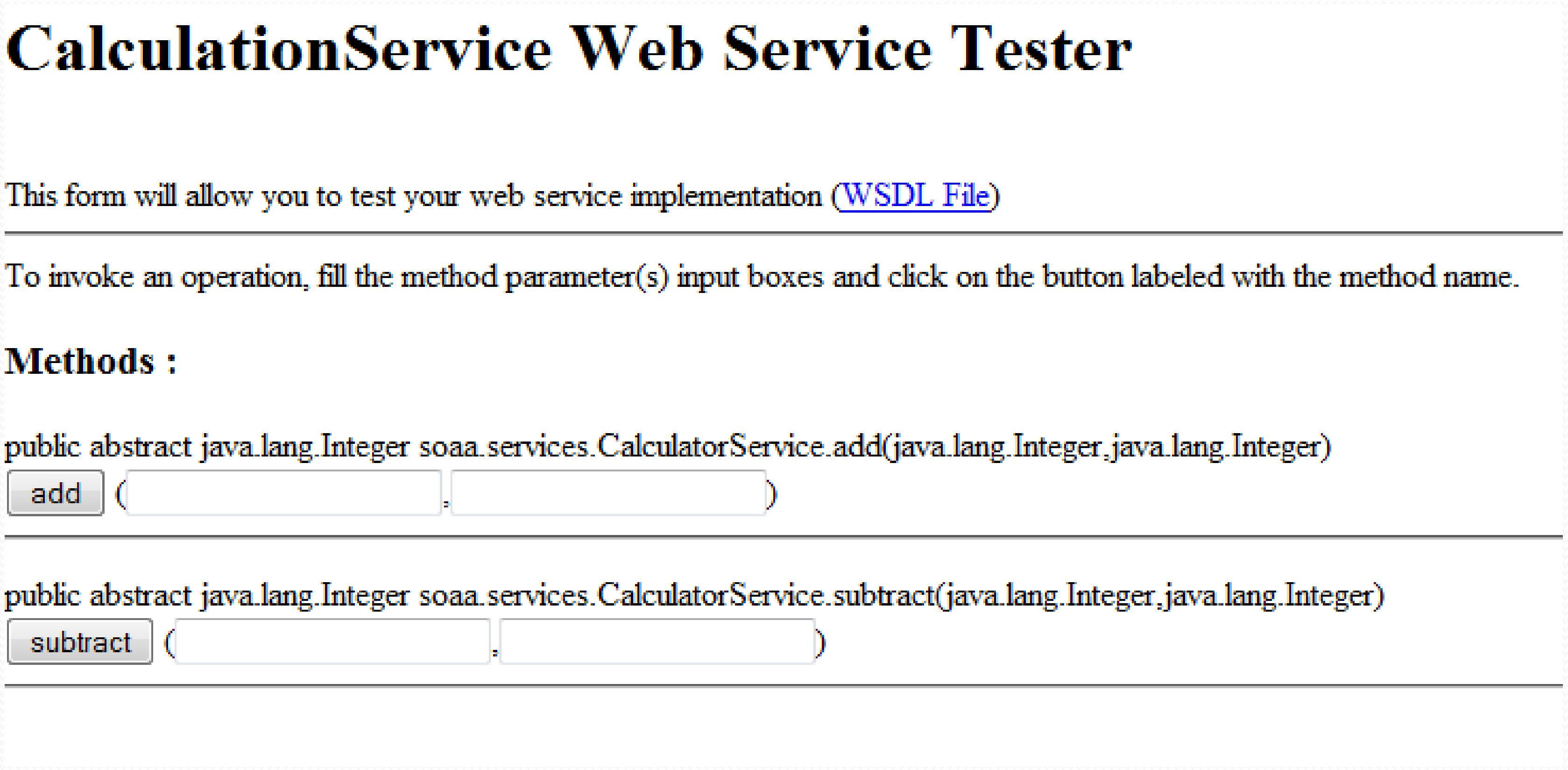


```
@WebService(serviceName = "CalculationService")
public class CalculatorService {

    @WebMethod (operationName = "add")
    public Integer Add (@WebParam(name="paramA") Integer a, @WebParam (name ="paramB") Integer b)
    {
        return a + b;

    }
    @WebMethod (operationName = "subtract")
    public Integer Subtract(@WebParam(name="paramA") Integer a, @WebParam (name ="paramB") Integer b)
    {
        return a - b;

    }

}
```

Deploying & Testing

- Just deploy from Netbeans from deploy option.
- Test your service through tester
  - http://localhost:8080/app-name/service-name?Tester
- Check WSDL
  - http://localhost:8080/app-name/service-name?WSDL

Tester



CalculationService Web Service Tester

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

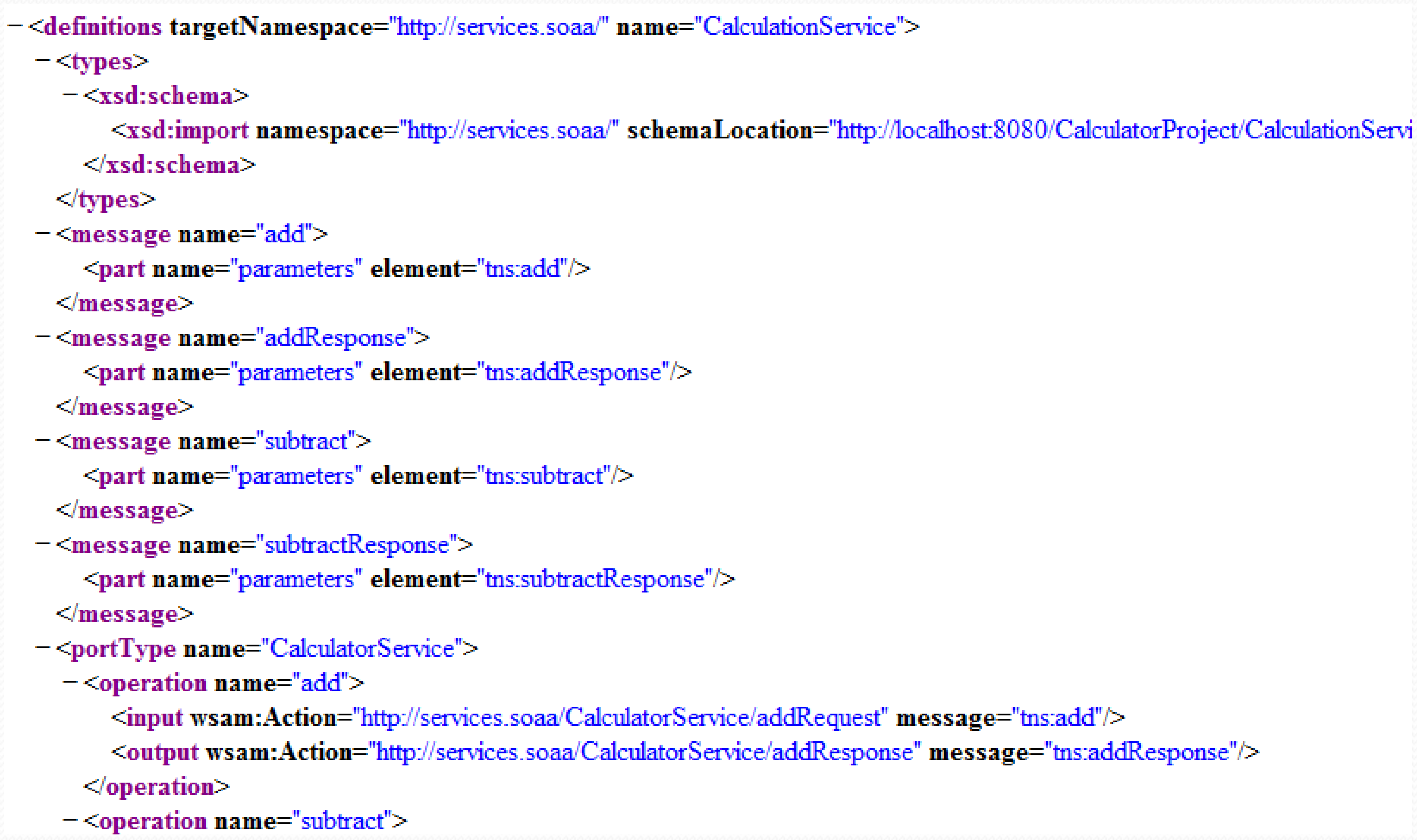
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.Integer soaa.services.CalculatorService.add(java.lang.Integer,java.lang.Integer)

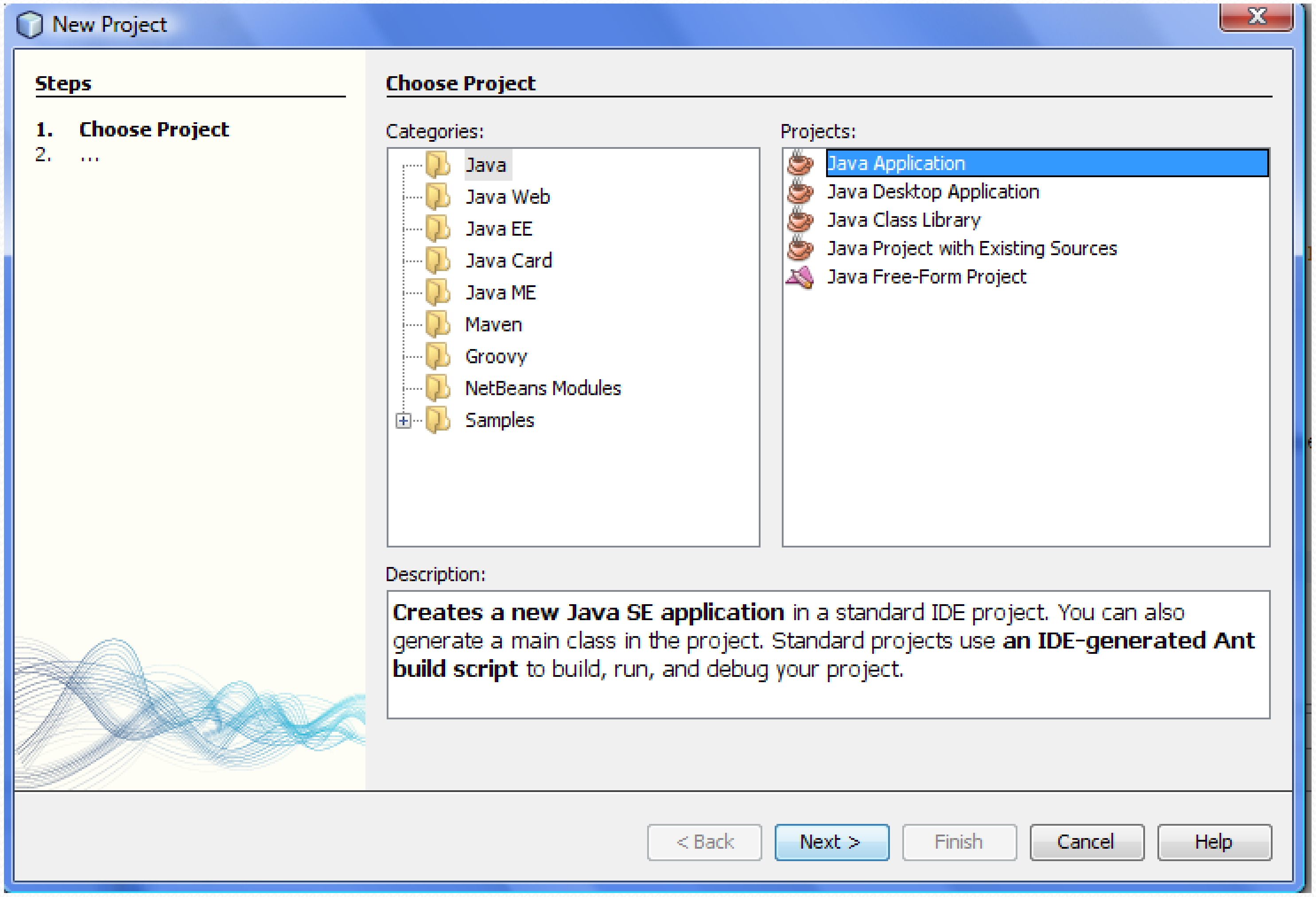
public abstract java.lang.Integer soaa.services.CalculatorService.subtract(java.lang.Integer,java.lang.Integer)

WSDL



```
− <definitions targetNamespace="http://services.soaa/" name="CalculationService">
  − <types>
    − <xsd:schema>
        <xsd:import namespace="http://services.soaa/" schemaLocation="http://localhost:8080/CalculatorProject/CalculationServi
      </xsd:schema>
    </types>
  − <message name="add">
      <part name="parameters" element="tns:add"/>
    </message>
  − <message name="addResponse">
      <part name="parameters" element="tns:addResponse"/>
    </message>
  − <message name="subtract">
      <part name="parameters" element="tns:subtract"/>
    </message>
  − <message name="subtractResponse">
      <part name="parameters" element="tns:subtractResponse"/>
    </message>
  − <portType name="CalculatorService">
    − <operation name="add">
        <input wsam:Action="http://services.soaa/CalculatorService/addRequest" message="tns:add"/>
        <output wsam:Action="http://services.soaa/CalculatorService/addResponse" message="tns:addResponse"/>
      </operation>
    − <operation name="subtract">
```

Client Project



New Project

Steps

1. Choose Project
2. ...

Choose Project

Categories:

- Java
- Java Web
- Java EE
- Java Card
- Java ME
- Maven
- Groovy
- NetBeans Modules
- Samples

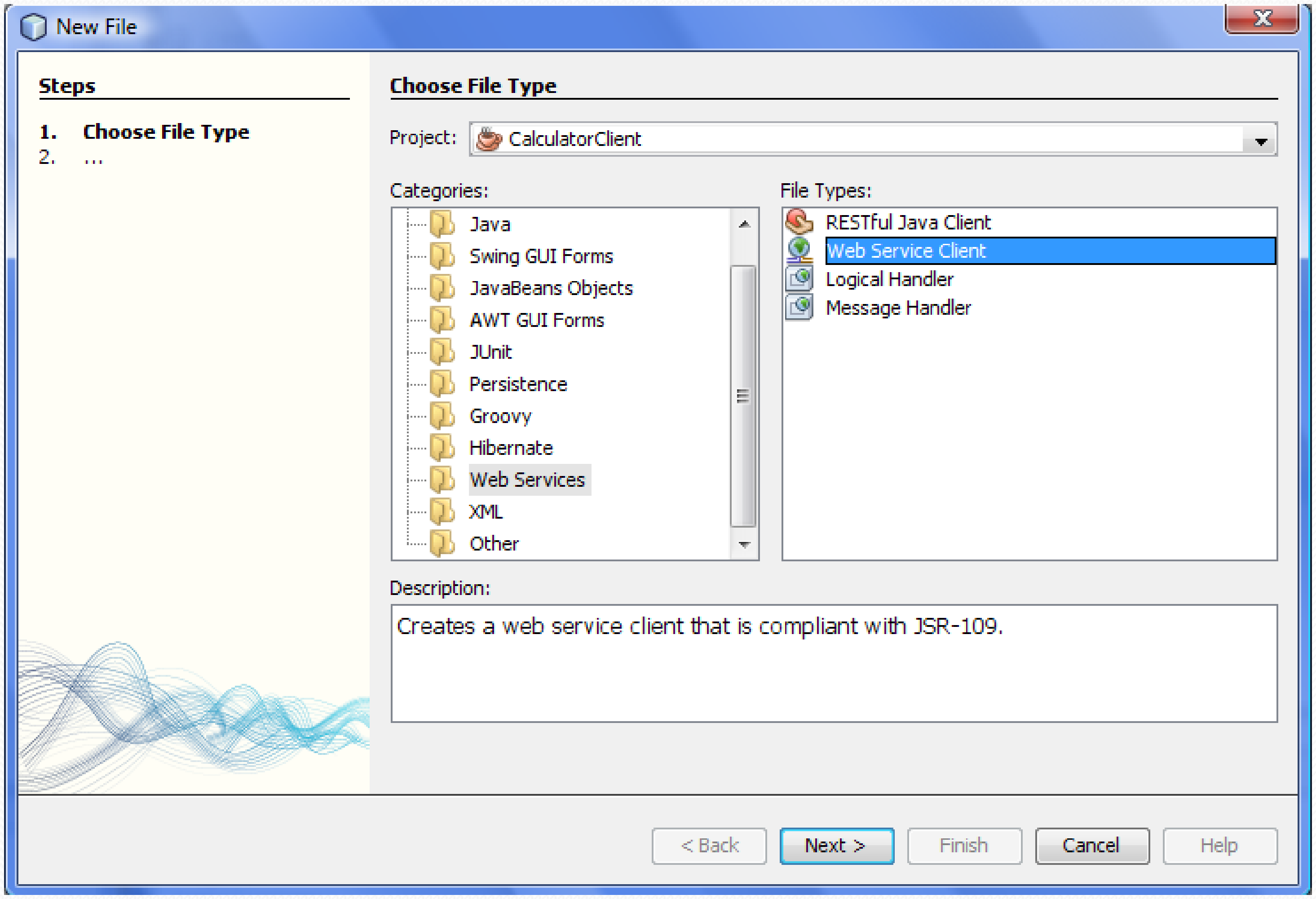
Projects:

- Java Application
- Java Desktop Application
- Java Class Library
- Java Project with Existing Sources
- Java Free-Form Project

Description:

Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Create Service client


New File

Steps

1. **Choose File Type**
2. ...

Choose File Type

Project: CalculatorClient

Categories:
- Java
- Swing GUI Forms
- JavaBeans Objects
- AWT GUI Forms
- JUnit
- Persistence
- Groovy
- Hibernate
- Web Services
- XML
- Other

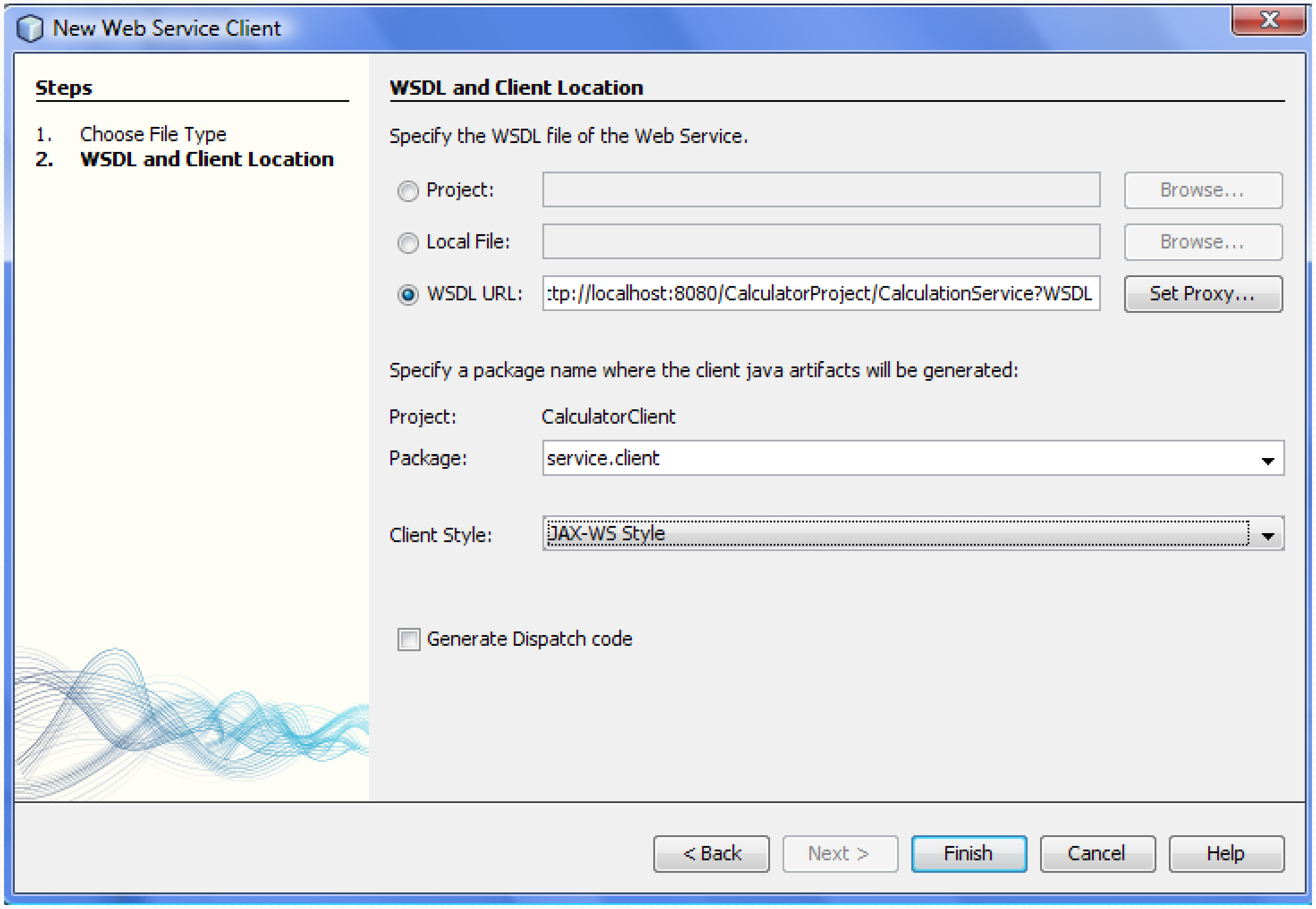
File Types:
- RESTful Java Client
- Web Service Client
- Logical Handler
- Message Handler

Description:

Creates a web service client that is compliant with JSR-109.

< Back | Next > | Finish | Cancel | Help

Give WSDL path


New Web Service Client

Steps

1. Choose File Type
2. **WSDL and Client Location**

WSDL and Client Location

Specify the WSDL file of the Web Service.

( ) Project: Browse...

( ) Local File: Browse...

(●) WSDL URL: :tp://localhost:8080/CalculatorProject/CalculationService?WSDL Set Proxy...

Specify a package name where the client java artifacts will be generated:

Project: CalculatorClient

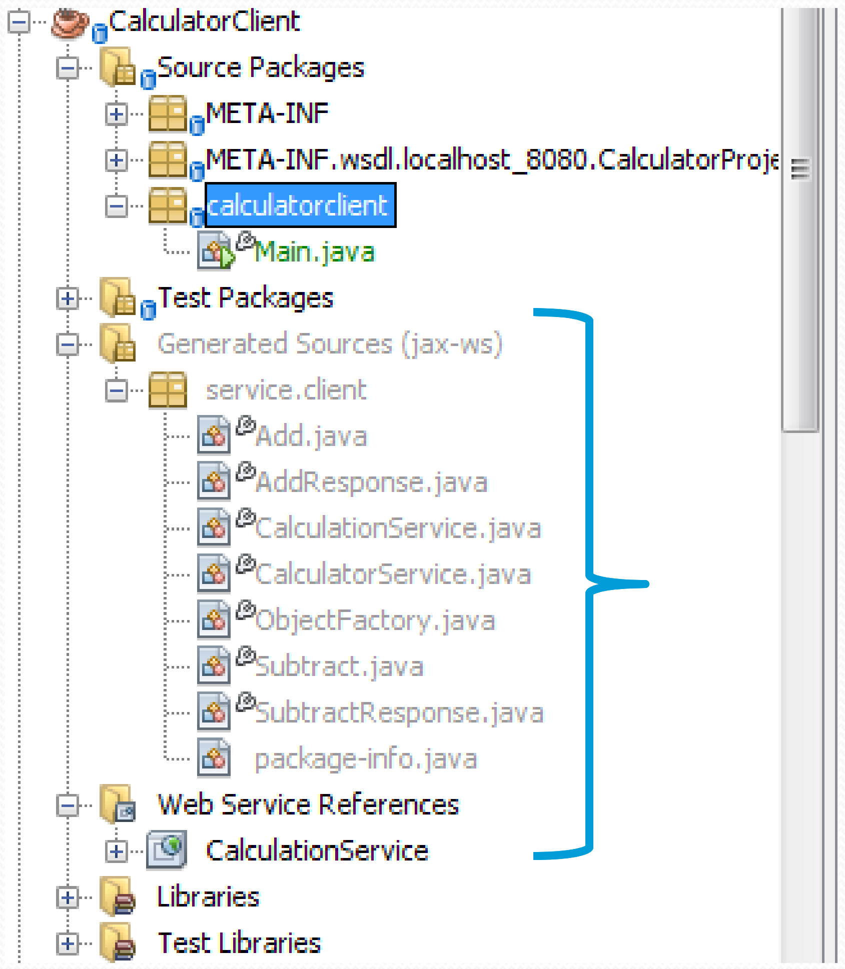
Package: service.client

Client Style: JAX-WS Style

[ ] Generate Dispatch code

< Back | Next > | Finish | Cancel | Help

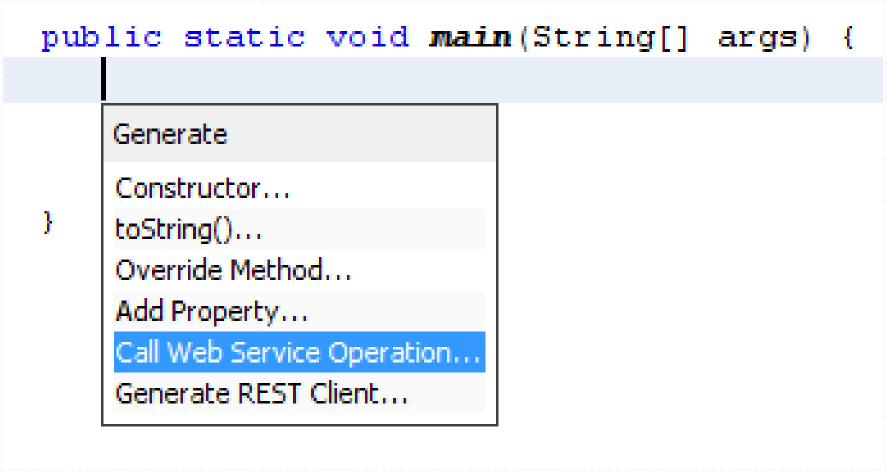
Stub generation



```
CalculatorClient
    Source Packages
        META-INF
        META-INF.wsdl.localhost_8080.CalculatorProje
        calculatorclient
            Main.java
    Test Packages
    Generated Sources (jax-ws)
        service.client
            Add.java
            AddResponse.java
            CalculationService.java
            CalculatorService.java
            ObjectFactory.java
            Subtract.java
            SubtractResponse.java
            package-info.java
    Web Service References
        CalculationService
    Libraries
    Test Libraries
```

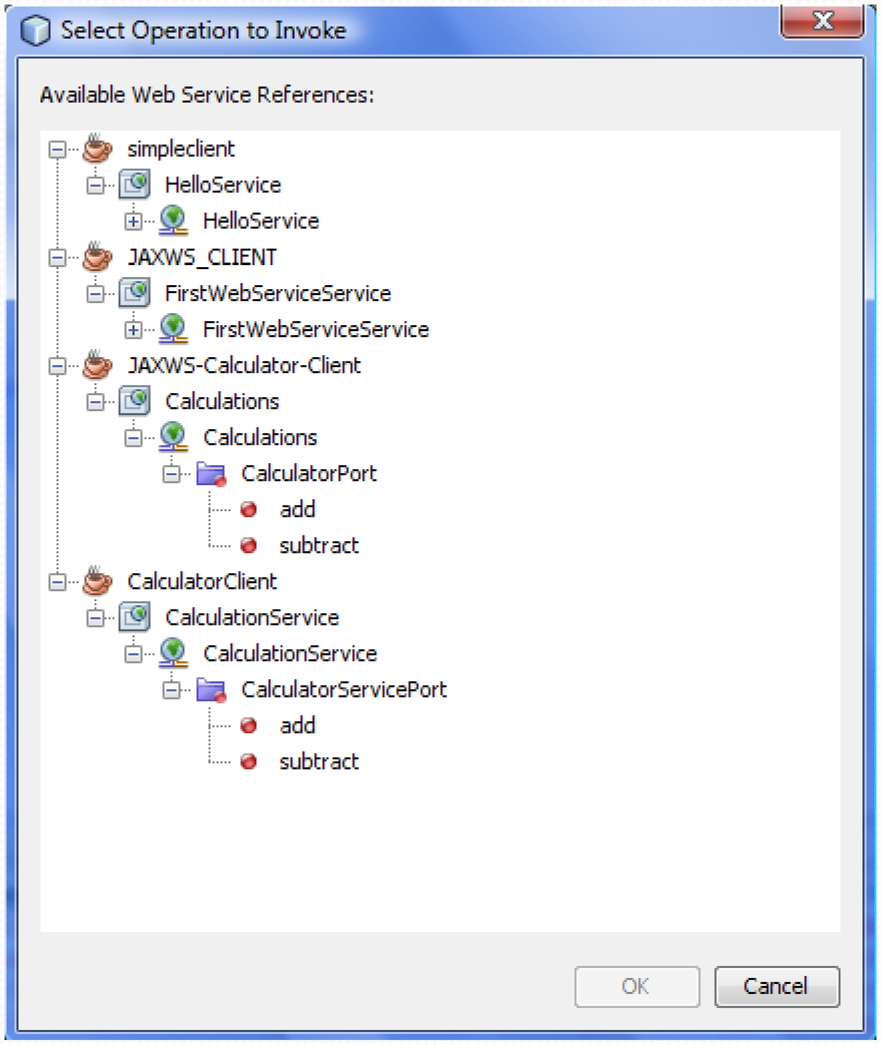
Main Class:

- Adding web service operation
  - Use ALT + INSERT or right click inside main class to insert code
  - Choose call web service operation



```
public static void main(String[] args) {

}
```

Select web service method

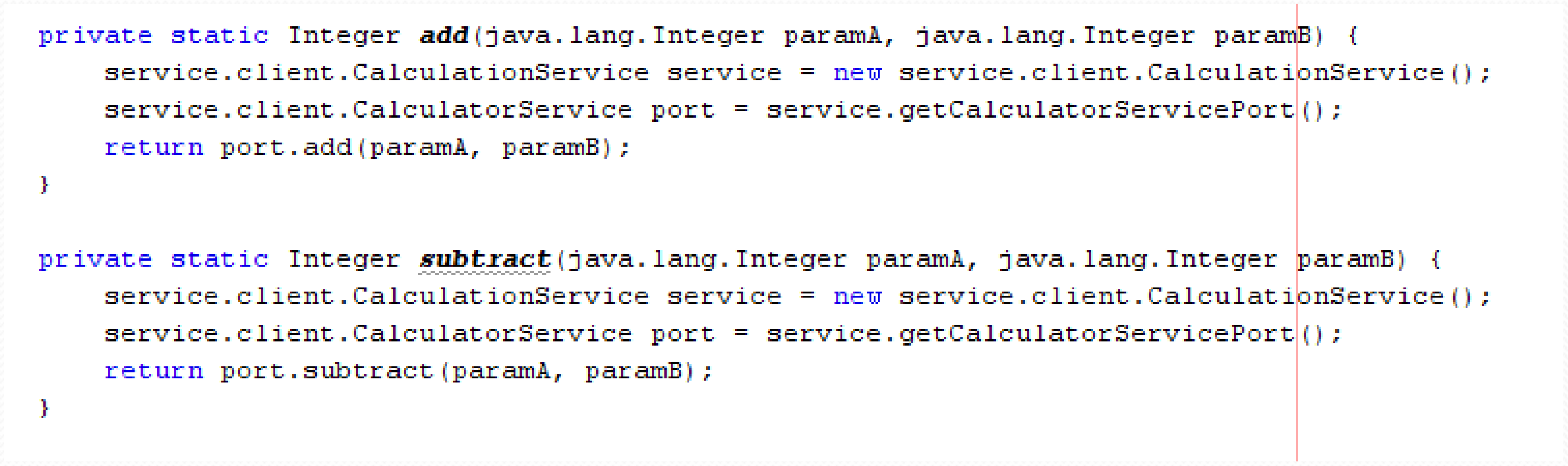


Select Operation to Invoke

Available Web Service References:

```
simpleclient
    HelloService
        HelloService
JAXWS_CLIENT
    FirstWebServiceService
        FirstWebServiceService
JAXWS-Calculator-Client
    Calculations
        Calculations
            CalculatorPort
                add
                subtract
CalculatorClient
    CalculationService
        CalculationService
            CalculatorServicePort
                add
                subtract
```

Web Service method



```
private static Integer add(java.lang.Integer paramA, java.lang.Integer paramB) {
    service.client.CalculationService service = new service.client.CalculationService();
    service.client.CalculatorService port = service.getCalculatorServicePort();
    return port.add(paramA, paramB);
}

private static Integer subtract(java.lang.Integer paramA, java.lang.Integer paramB) {
    service.client.CalculationService service = new service.client.CalculationService();
    service.client.CalculatorService port = service.getCalculatorServicePort();
    return port.subtract(paramA, paramB);
}
```

Let do this Exercise together

Exercise

- Use http://www.webservicex.net/
- Use there weather service
- Call the service using JAX-WS methodology
- Ask for Milan’s weather.

Assignment-3

- All requirements are the same as assignment-2.
- Just provide same services using SOAP.
- You can use any method to implement SOAP which we learned in class.
- The use of JAX-WS is recommended.

- Deadline: 7th December 2011 (midnight)