SVKM's NMIMS University

Mukesh Patel School of Technology Management & Engineering

BTech IT SEM VIII

SUBJECT: Service Oriented Architecture **Practical Assignment: 4**

Part A (To be referred by students)

Topic: JAX- RPC Service

A. Create Java Web Application with a Web Service called interest calculation using JAX-RPC.
B. Create Client Web Service to call the interest service.

Softwares Required :
NetBeans IDE Version 7.0 with GlassFish Server 3.1
Java SDK 6 bundle

Krishna Palod (Assistant. Prof., Comp Dept.)

You can use the JAX-RPC programming model to develop SOAP-based web service clients and endpoints. JAX-RPC enables clients to invoke web services developed across heterogeneous platforms. Likewise, JAX-RPC web service endpoints can be invoked by heterogeneous clients. JAX-RPC requires SOAP and WSDL standards for this cross-platform interoperability.

JAX-RPC lets people develop a web service endpoint using either a Servlet or Enterprise JavaBeans (EJB) component model. The endpoint is then deployed on either the Web or EJB container, based on the corresponding component model. Endpoints are described using a Web Services Description Language (WSDL) document. (This WSDL document can be published in a public or private registry, though this is not required). A client then uses this WSDL document and invokes the web service endpoint.

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

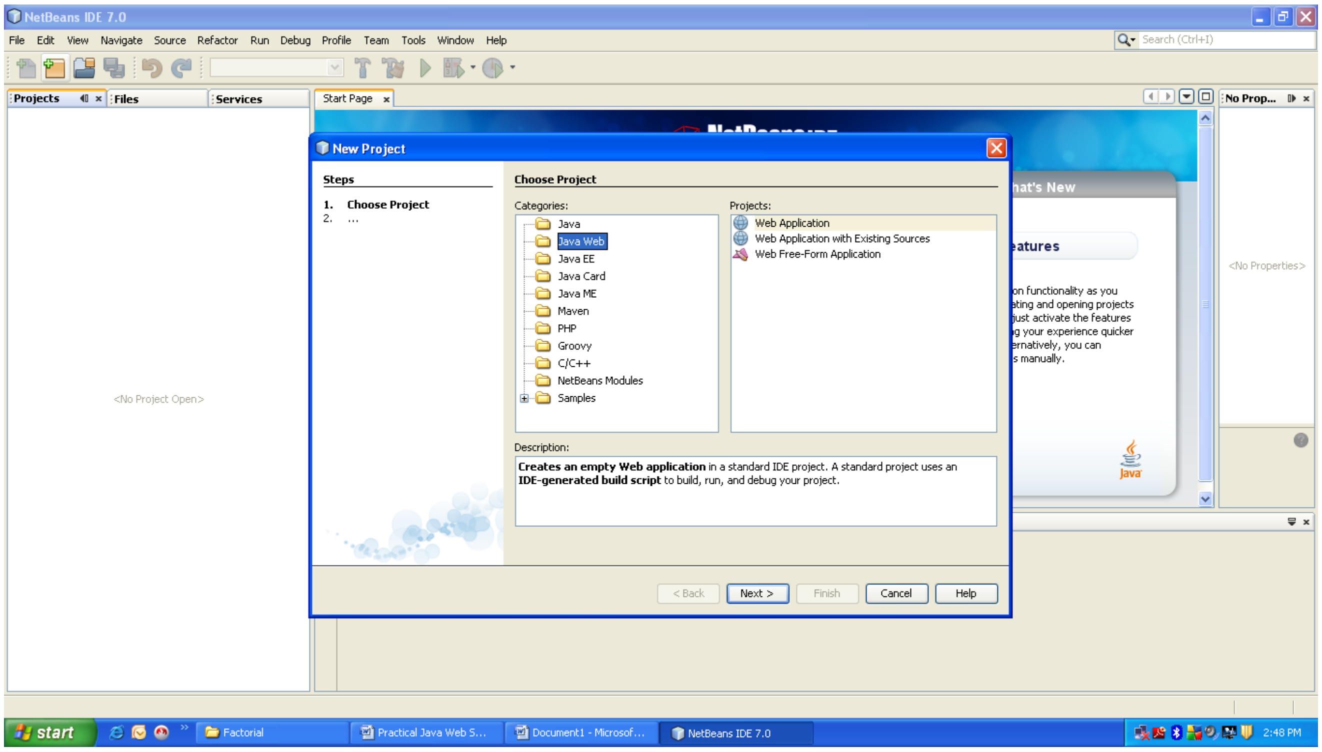
With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.

Here we develop a webservice that calculates the interest as part A. We will also write the code to call the webservice as part B.

- ✓ Part A : Steps for Creating a Web Service called interest

Start NetBeans 7.0 by clicking on it and start a new project **File->New Project**.

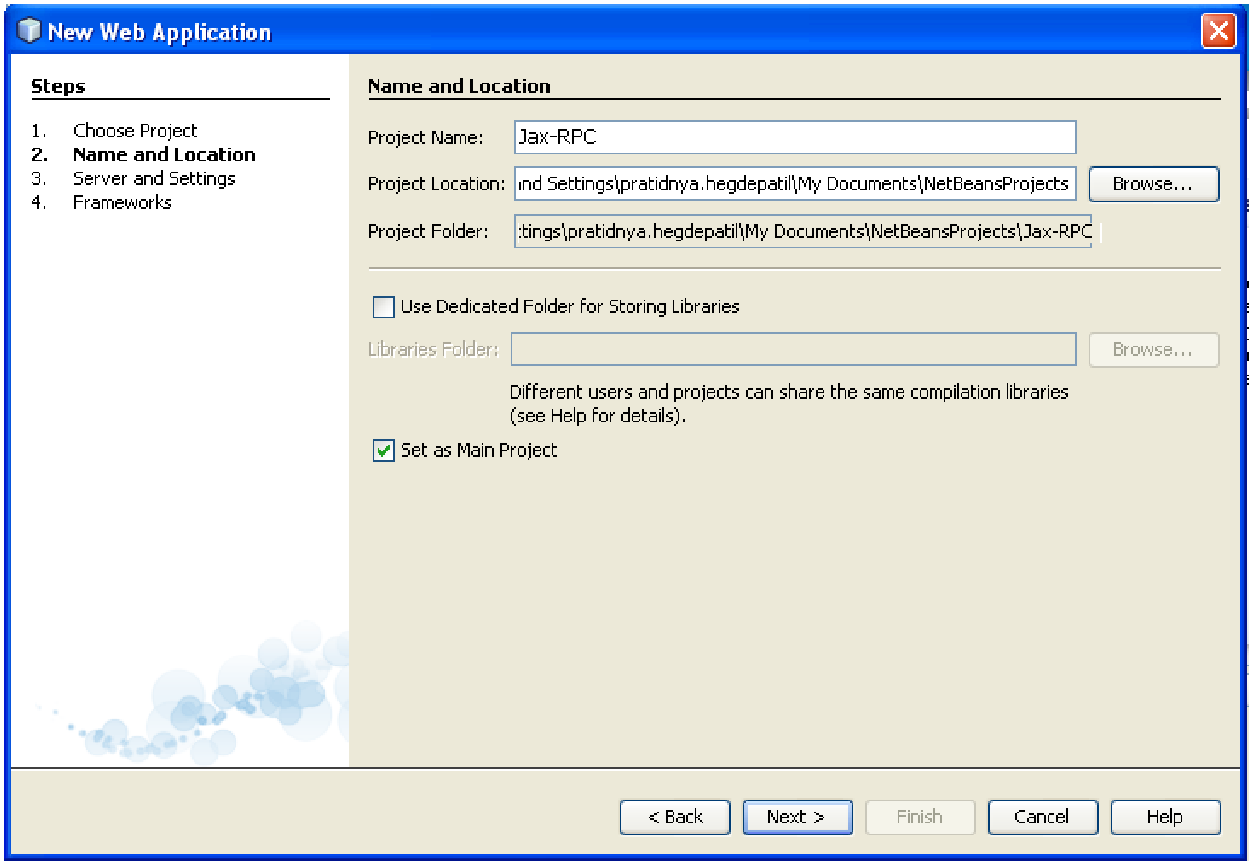
Select the New Project as **Java Web -> Web Application**.



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.

Click on Next and specify the application name as Jax-RPC.



Name and Location

Project Name: Jax-RPC

Project Location: ınd Settings\pratidnya.hegdepatil\My Documents\NetBeansProjects

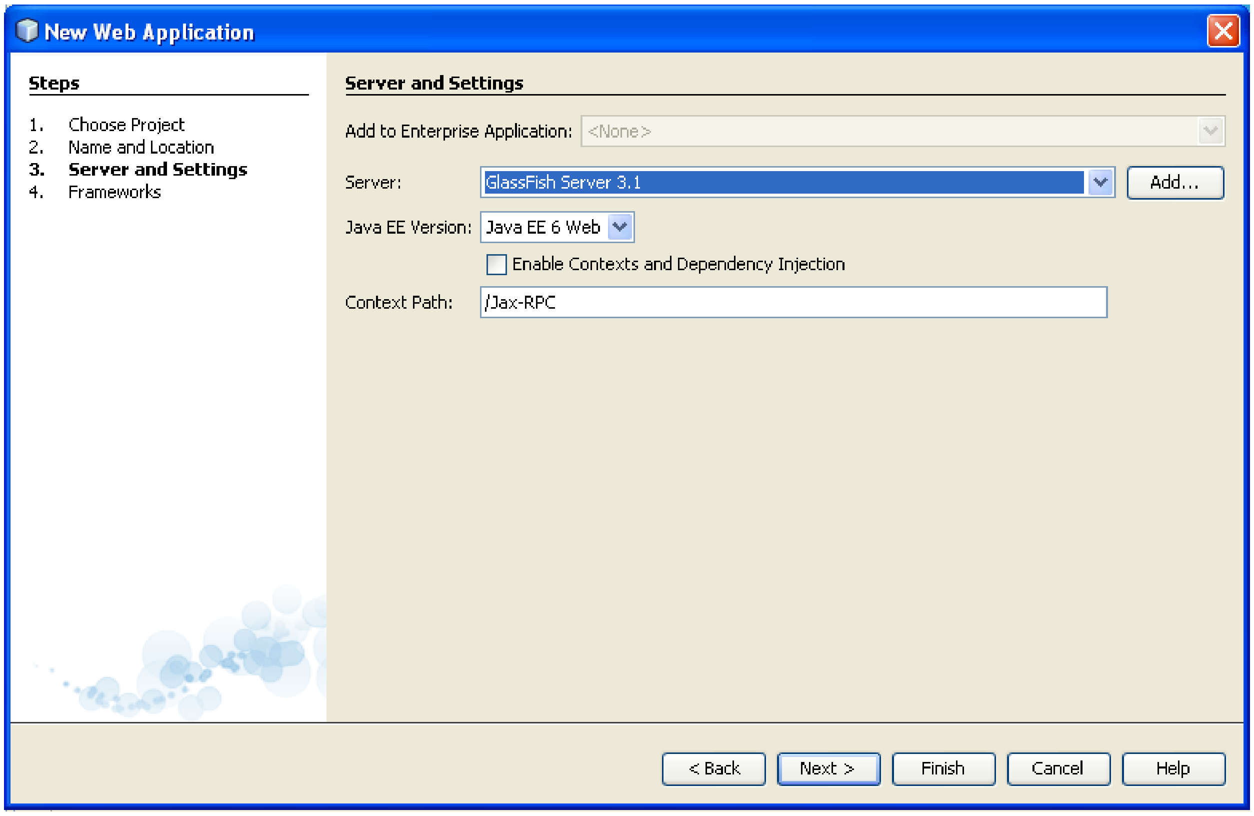
Project Folder: :tings\pratidnya.hegdepatil\My Documents\NetBeansProjects\Jax-RPC

[ ] Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

[✓] Set as Main Project



Server and Settings

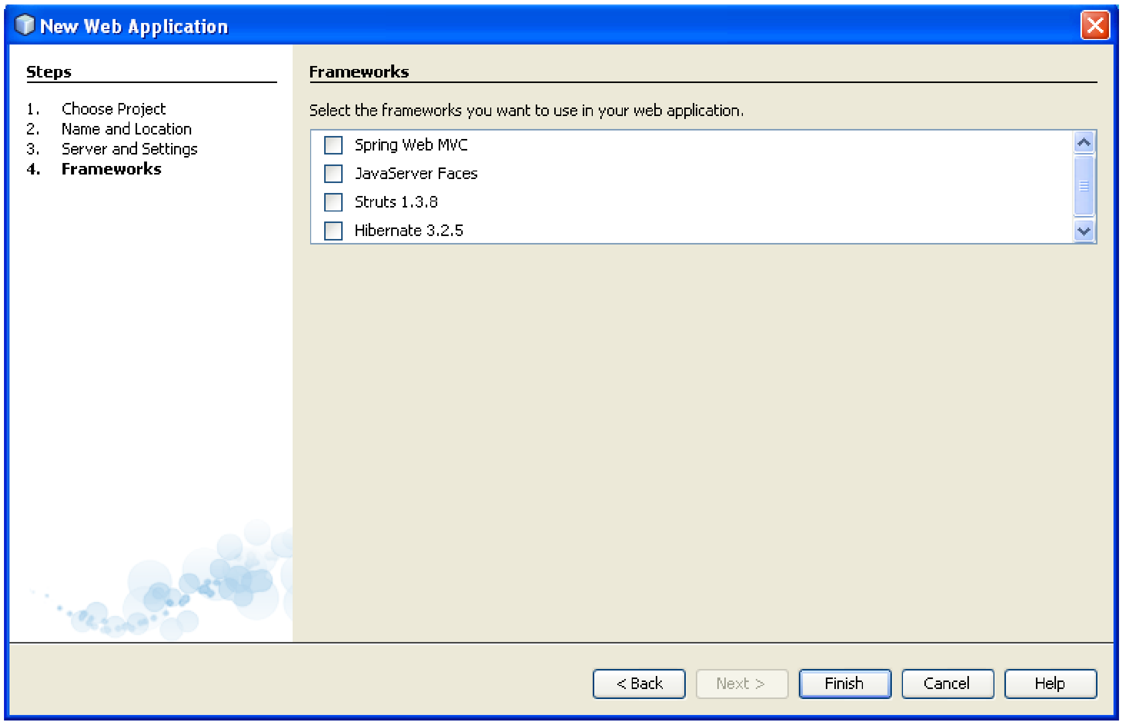
Add to Enterprise Application: <None>

Server: GlassFish Server 3.1

Java EE Version: Java EE 6 Web

[ ] Enable Contexts and Dependency Injection

Context Path: /Jax-RPC


New Web Application

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

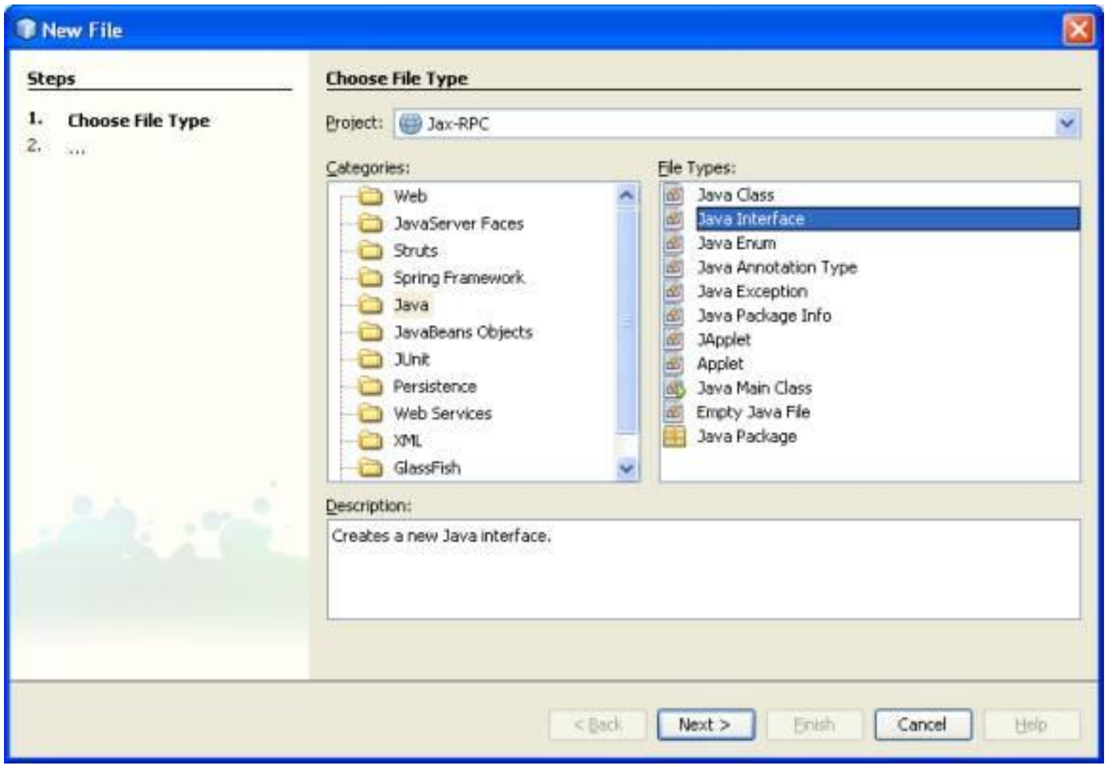
Frameworks

Select the frameworks you want to use in your web application.

- [ ] Spring Web MVC
- [ ] JavaServer Faces
- [ ] Struts 1.3.8
- [ ] Hibernate 3.2.5

Create the Java Interface :

Select the project Jax-RPC and right click on it. Select **New->Java Interface**. Name it JAX1.


New File

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

Choose File Type

Project: Jax-RPC

Categories: Web, JavaServer Faces, Struts, Spring Framework, Java, JavaBeans Objects, JUnit, Persistence, Web Services, XML, GlassFish

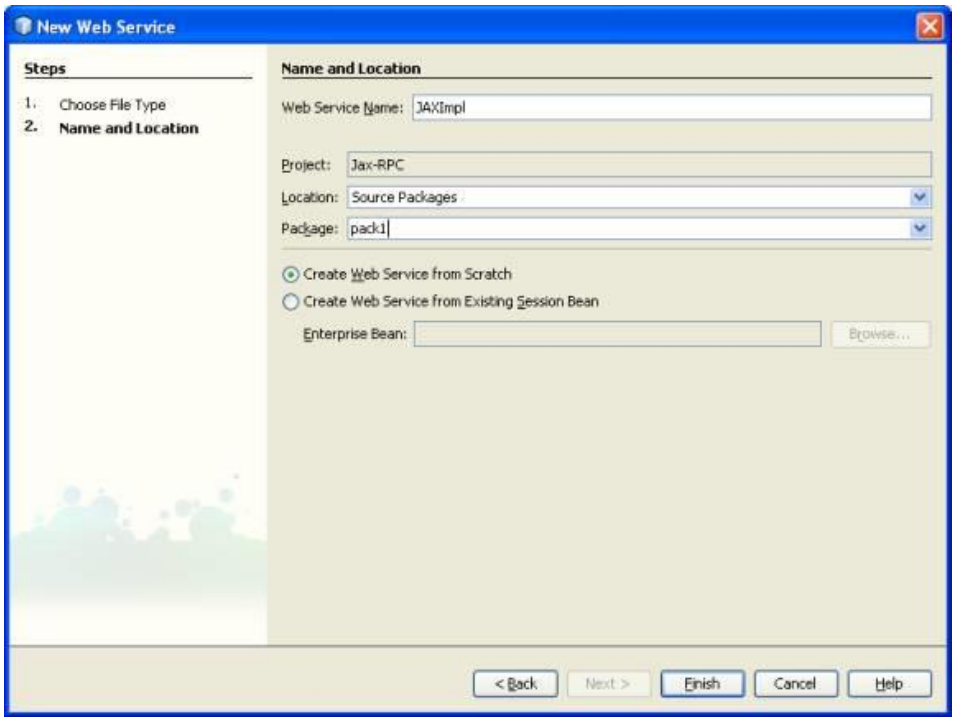
File Types: Java Class, Java Interface, Java Enum, Java Annotation Type, Java Exception, Java Package Info, JApplet, Applet, Java Main Class, Empty Java File, Java Package

Description:

Creates a new Java interface.

Rewrite the source code for JAX1.java interface to **Create a Web Service Endpoint Interface.**

Select the project JAX-RPC and right click it. Select **New->Web Service**. Call it JAXImpl in package pack1.



Web Service Name: JAXImpl

Project: Jax-RPC

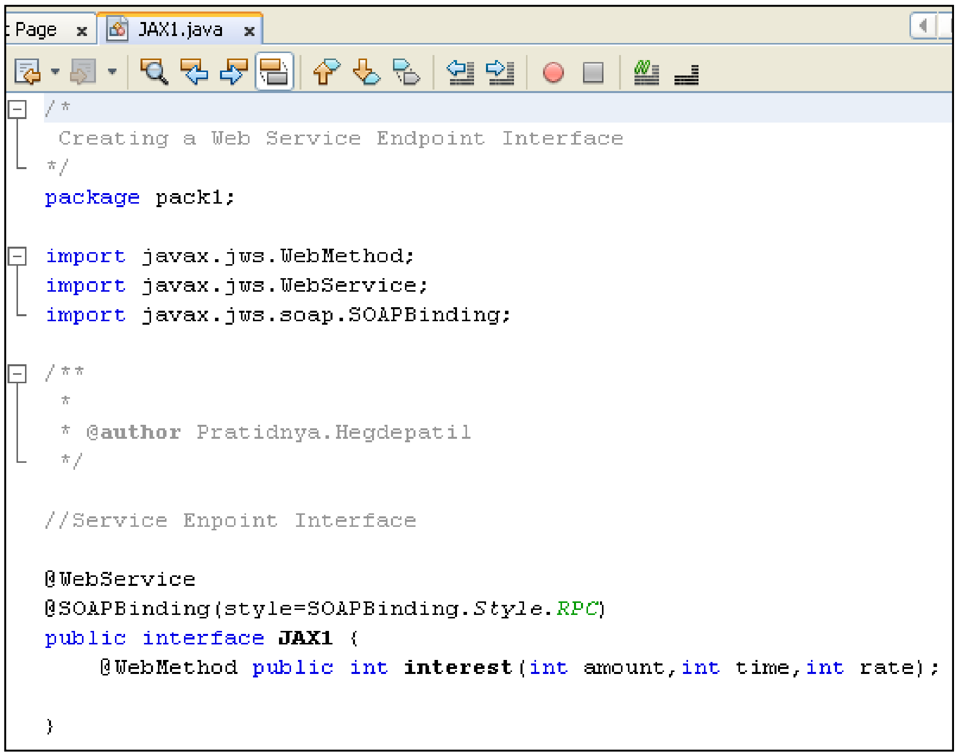
Location: Source Packages

Package: pack1

(●) Create Web Service from Scratch

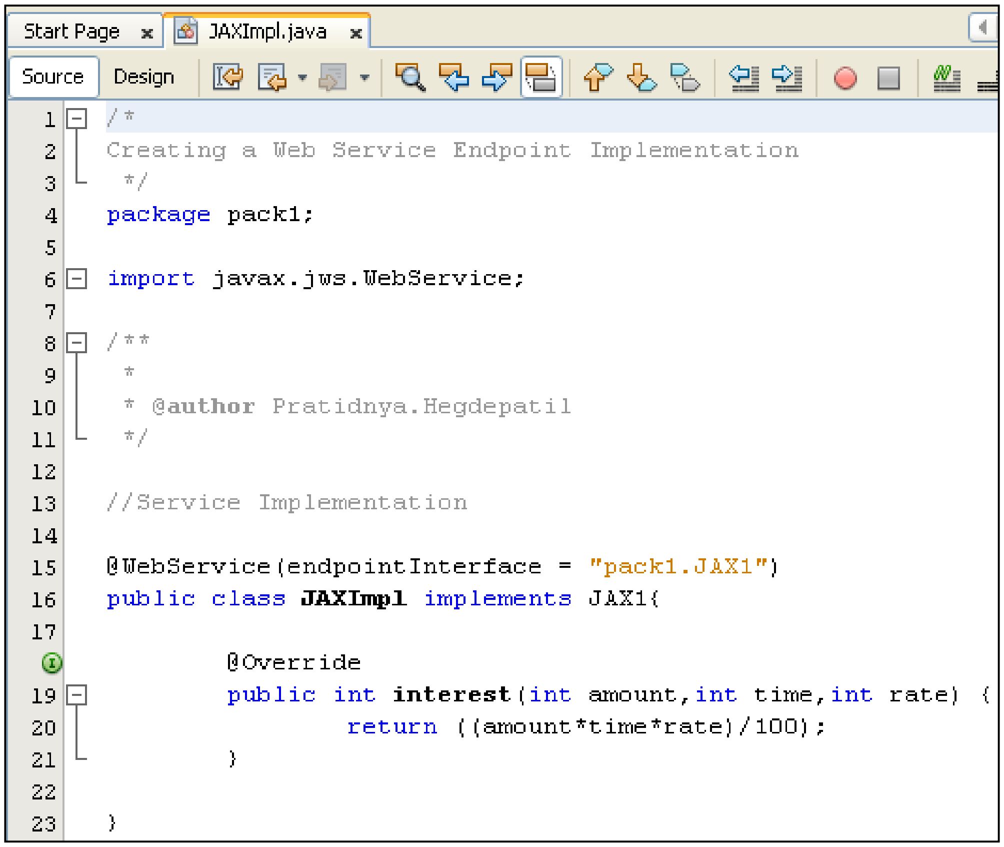
( ) Create Web Service from Existing Session Bean

Enterprise Bean:



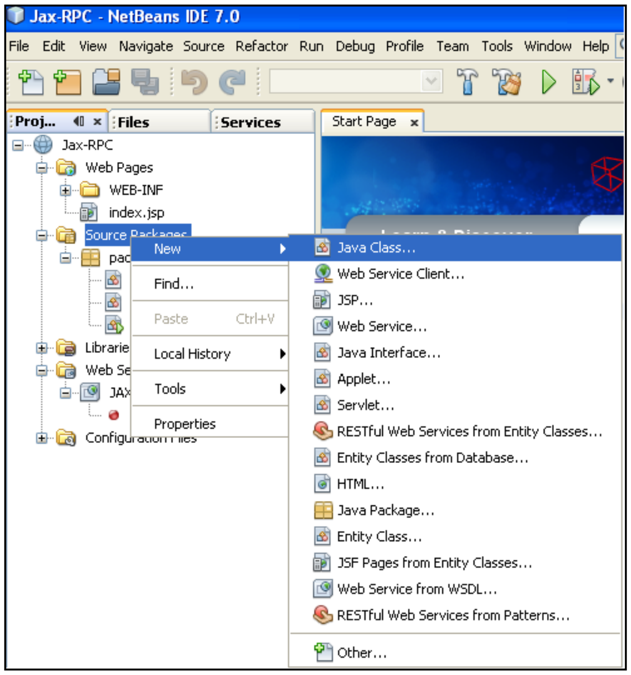
```
/*
 Creating a Web Service Endpoint Interface
*/
package pack1;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

/**
 *
 * @author Pratidnya.Hegdepatil
 */

//Service Enpoint Interface

@WebService
@SOAPBinding(style=SOAPBinding.Style.RPC)
public interface JAX1 {
    @WebMethod public int interest(int amount,int time,int rate);

}
```

Rewrite the source code for JAXImpl.java to **Create a Web Service Endpoint Implementation.**

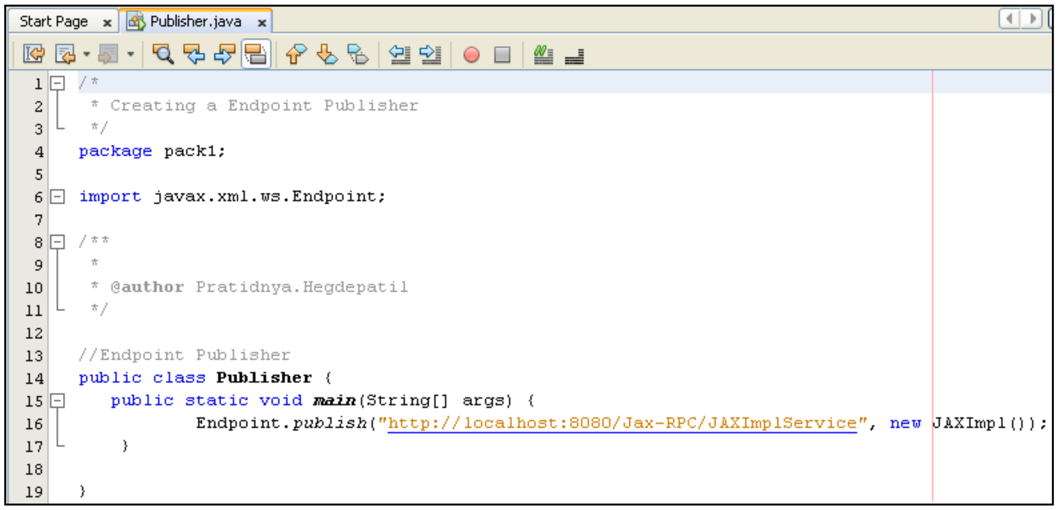


```
/*
 Creating a Web Service Endpoint Implementation
 */
package pack1;

import javax.jws.WebService;

/**
 *
 * @author Pratidnya.Hegdepatil
 */

//Service Implementation

@WebService(endpointInterface = "pack1.JAX1")
public class JAXImpl implements JAX1{

        @Override
        public int interest(int amount,int time,int rate) {
                return ((amount*time*rate)/100);
        }

}
```

Select the source packages of Jax-RPC project and right click on it. Select **New->Java Class**. Call it Publisher.

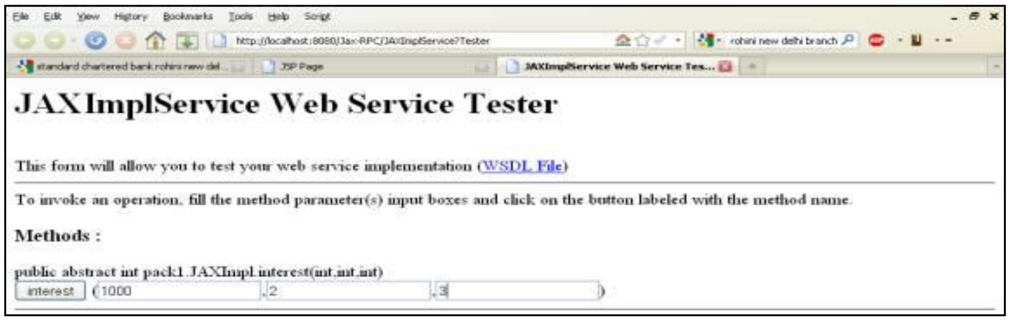

Jax-RPC - NetBeans IDE 7.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Proj... Files Services Start Page
Jax-RPC
Web Pages
WEB-INF
index.jsp
Source Packages
New
Find...
Paste Ctrl+V
Local History
Tools
Properties
Configuration Files
Java Class...
Web Service Client...
JSP...
Web Service...
Java Interface...
Applet...
Servlet...
RESTful Web Services from Entity Classes...
Entity Classes from Database...
HTML...
Java Package...
Entity Class...
JSF Pages from Entity Classes...
Web Service from WSDL...
RESTful Web Services from Patterns...
Other...

Rewrite the source code of Publisher.java to **Create a Endpoint Publisher**.



```
/*
 * Creating a Endpoint Publisher
 */
package pack1;

import javax.xml.ws.Endpoint;

/**
 *
 * @author Pratidnya.Hegdepatil
 */

//Endpoint Publisher
public class Publisher {
   public static void main(String[] args) {
           Endpoint.publish("http://localhost:8080/Jax-RPC/JAXImplService", new JAXImpl());
    }

}
```

Build & Deploy the project Jax-RPC Project. Select the Web Service JAXImpl and right click on it and select **Test Web Service**. This displays the Tester with the WSDL Document. Store the WSDL URL for later use.



http://localhost:8080/Jax-RPC/JAXImplService?Tester

JAXImplService Web Service Tester

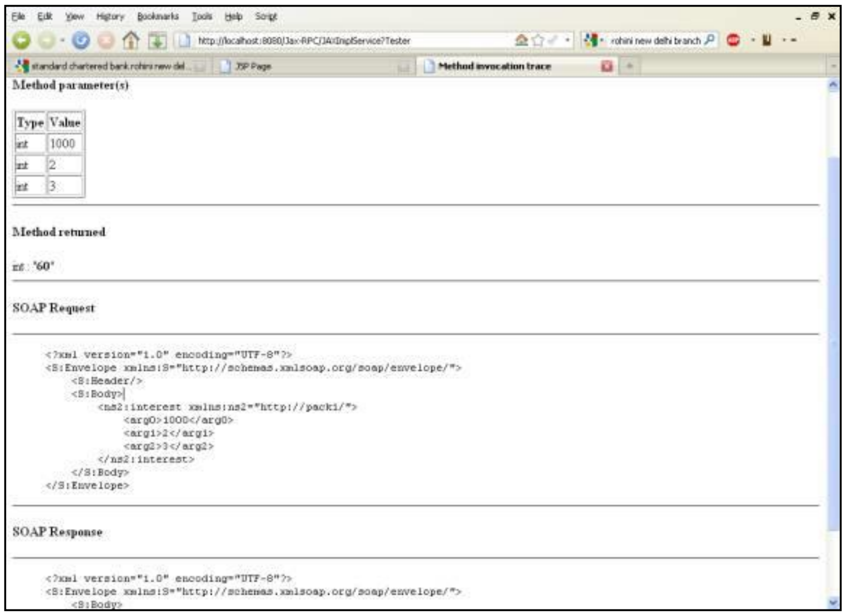
This form will allow you to test your web service implementation (WSDL File)

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 pack1.JAXImpl.interest(int,int,int)

interest (1000 ,2 ,3)



http://localhost:8080/Jax-RPC/JAXImplService?Tester

Method parameter(s)

Type	Value
int	1000
int	2
int	3

Method returned

int : "60"

SOAP Request

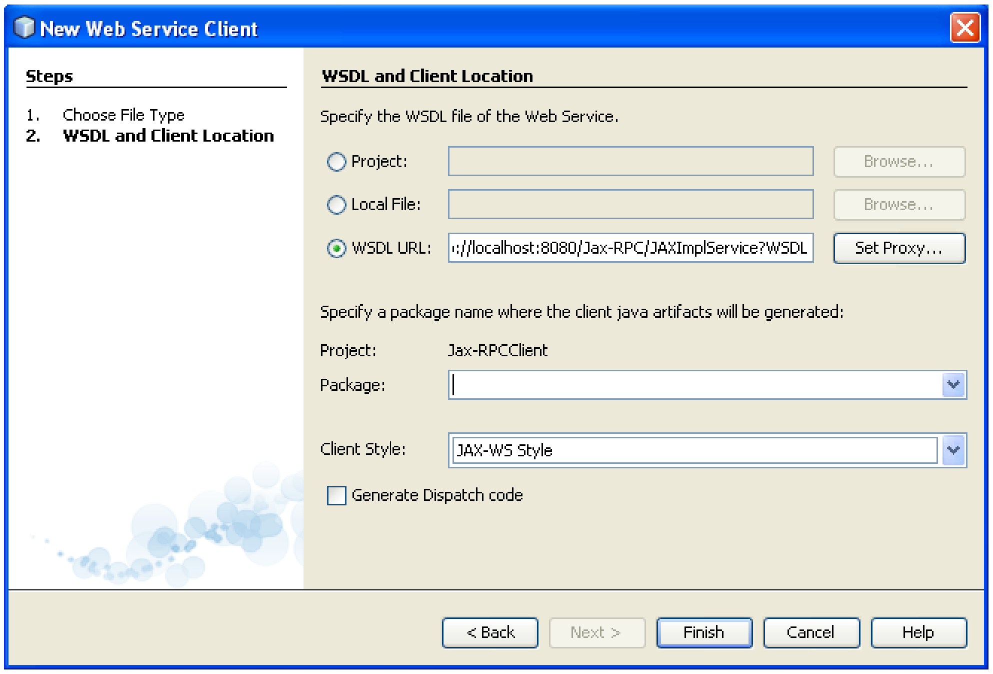
```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:interest xmlns:ns2="http://pack1/">
            <arg0>1000</arg0>
            <arg1>2</arg1>
            <arg2>3</arg2>
        </ns2:interest>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
```

- ✓ Part B : Steps for Creating a Client Web Service to call interest

Create a client web service called Jax-RPCClient by selecting **New -> Web Application**. Add the client web service by selecting the project Jax-RPCClient and right clicking on it. Select **New-> Web Service Client**, call it . Enter the WSDL URL http://localhost:8080/Jax-RPC/JAXImplService?WSDL



WSDL and Client Location

Specify the WSDL file of the Web Service.

- Project:
- Local File:
- [✓ WSDL URL:] ://localhost:8080/Jax-RPC/JAXImplService?WSDL

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

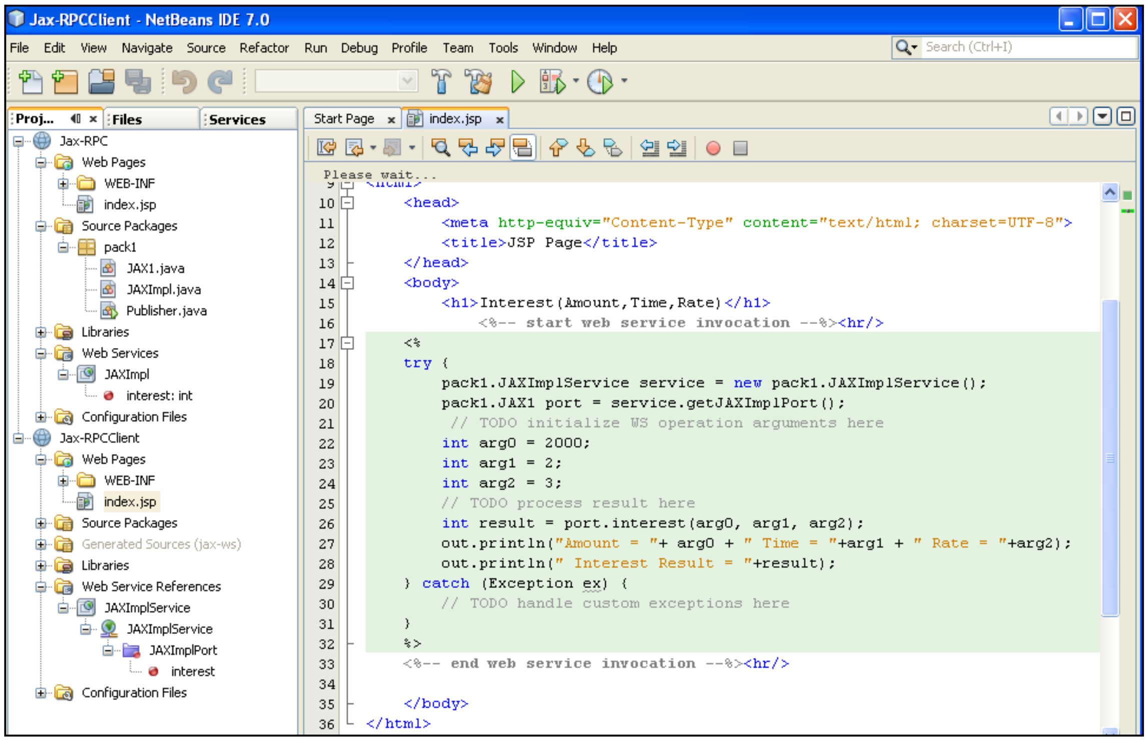
Project: Jax-RPCClient

Package:

Client Style: JAX-WS Style

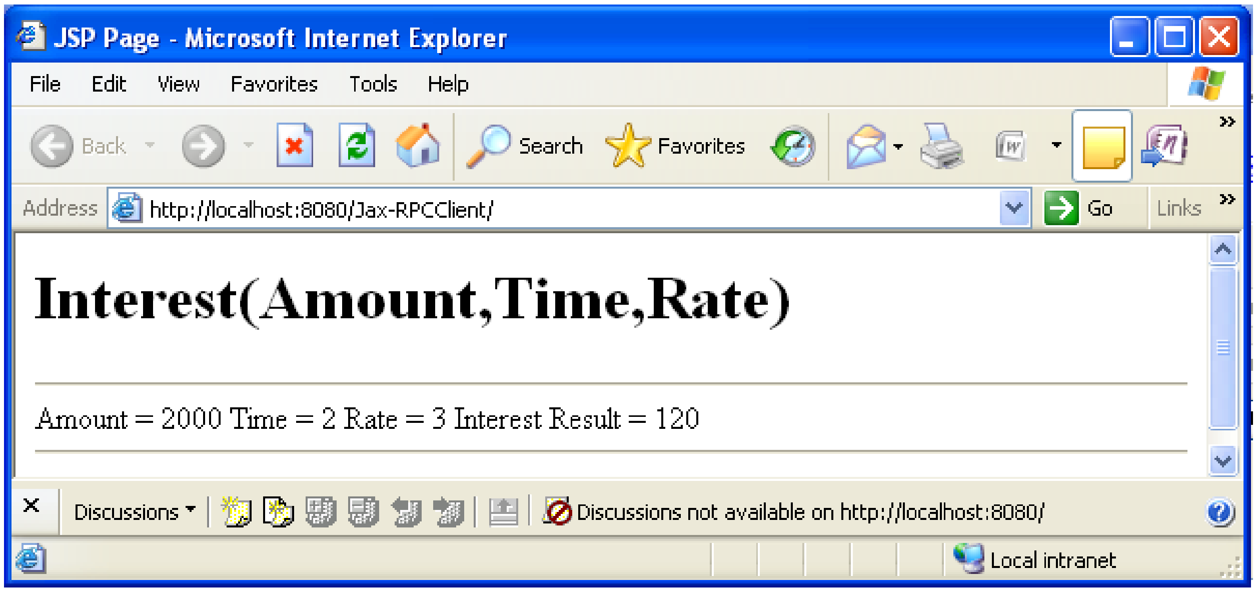
☐ Generate Dispatch code

Now drag the interest property into the index.jsp pageof the JAX-RPCClient project, in the body section.



```
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Interest(Amount,Time,Rate)</h1>
            <%-- start web service invocation --%><hr/>
    <%
    try {
        pack1.JAXImplService service = new pack1.JAXImplService();
        pack1.JAX1 port = service.getJAXImplPort();
         // TODO initialize WS operation arguments here
        int arg0 = 2000;
        int arg1 = 2;
        int arg2 = 3;
        // TODO process result here
        int result = port.interest(arg0, arg1, arg2);
        out.println("Amount = "+ arg0 + " Time = "+arg1 + " Rate = "+arg2);
        out.println(" Interest Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>

    </body>
</html>
```

Run the project. This will display the jsp page in IE with the interest output.


JSP Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/Jax-RPCClient/
Go Links
Discussions
Discussions not available on http://localhost:8080/
Local intranet

Interest(Amount,Time,Rate)

Amount = 2000 Time = 2 Rate = 3 Interest Result = 120

This completes the Practical on Web Service using JAX-RPC.

Create a GUI page for client side.

Part B (to be completed by students)

(Students must submit the soft copy as per the following segments. The soft copy must be uploaded on the Blackboard. The filename should be Batch_RollNo_Exp_No)

Roll No.:	**Name:**
Prog/Yr/Sem:	**Batch:**
Date of Experiment:	**Date of Submission:**

1. **Questions:**
   a. How this practical implementation is different from Practical 3
   b. 2. Difference between JAX-WS and JAX-RPC
2 **Output :**
   1.Output of the jsp page.
   2. Output of http://localhost:8080/Jax-RPC/JAXImplService showing the endpoint interface.
   3. Output of the WSDL document.

Conclusion (Learning Outcomes): How are the outcomes defined for the experiment in Part a fulfilled through the scenarios?