SVKM's NMIMS University
Mukesh Patel School of Technology Management & Engineering

BTech IT SEM VIII

SUBJECT: Service Oriented Architecture **Practical Assignment: 3**

Part A (To be referred by students)

Topic: To Create a factorial Service and create client for the same

Aim: To create a Factorial service and create client for the same

Prerequisite: Basic knowledge of Java and net beans

Outcome: After successful completion of this practical students will be able to

1. Understand how to create Service using Jax-ws API
2. Create JSP client for the same

Theory

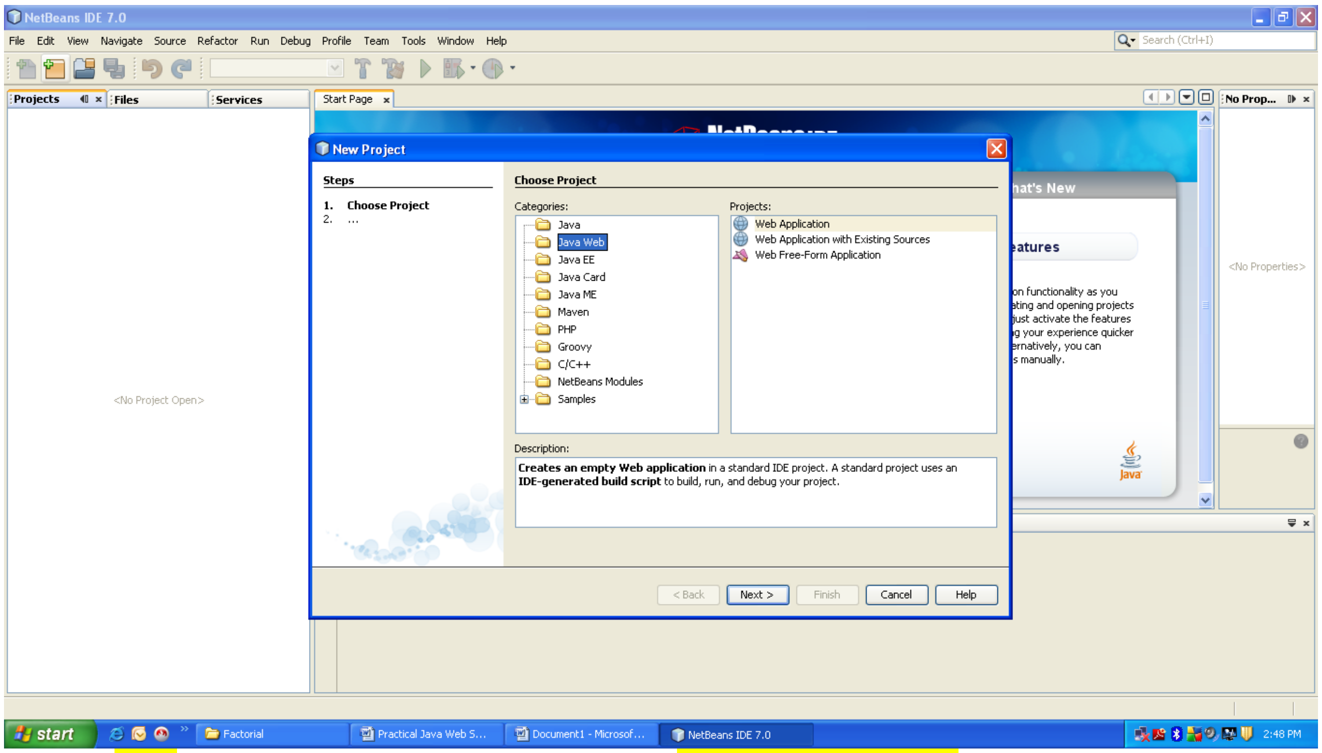
The **Java API for XML Web Services (JAX-WS)** is a Java programming language API for creating web services. JAX-WS is also used to build Web services and corresponding clients that communicate using XML to send messages or use remote procedure calls to exchange data between client and service provider.

JAX-WS represents remote procedure calls or messages using XML-based protocols such as SOAP, but hides SOAP's innate complexity behind a Java-based API. Developers use this API to define methods, then code one or more classes to implement those methods and leave the communication details to the underlying JAX-WS API. Clients create a local proxy to represent a service, then invoke methods on the proxy. The JAX-WS runtime system converts API calls and matching replies to and from SOAP messages.

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

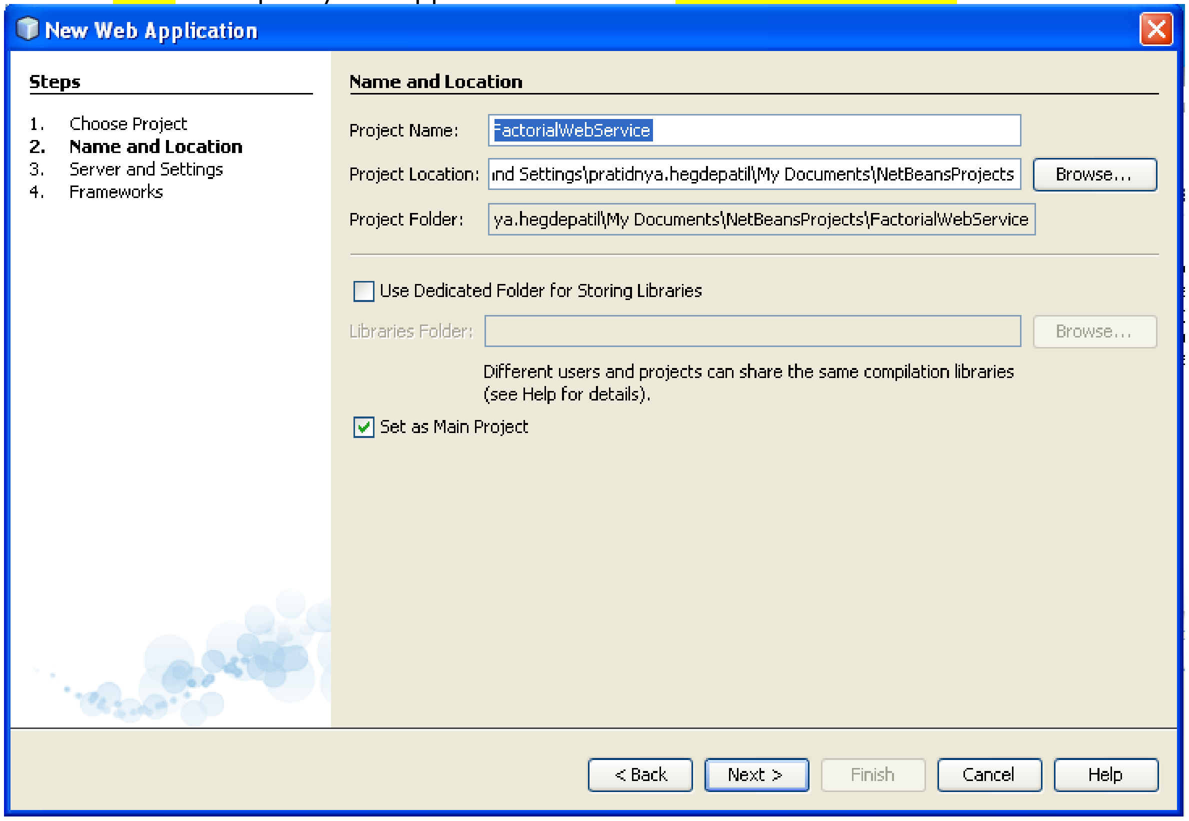
- ✓ Part A : Steps for Creating a Web Service called factorial

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 FactorialWebService.



Project Name: FactorialWebService

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

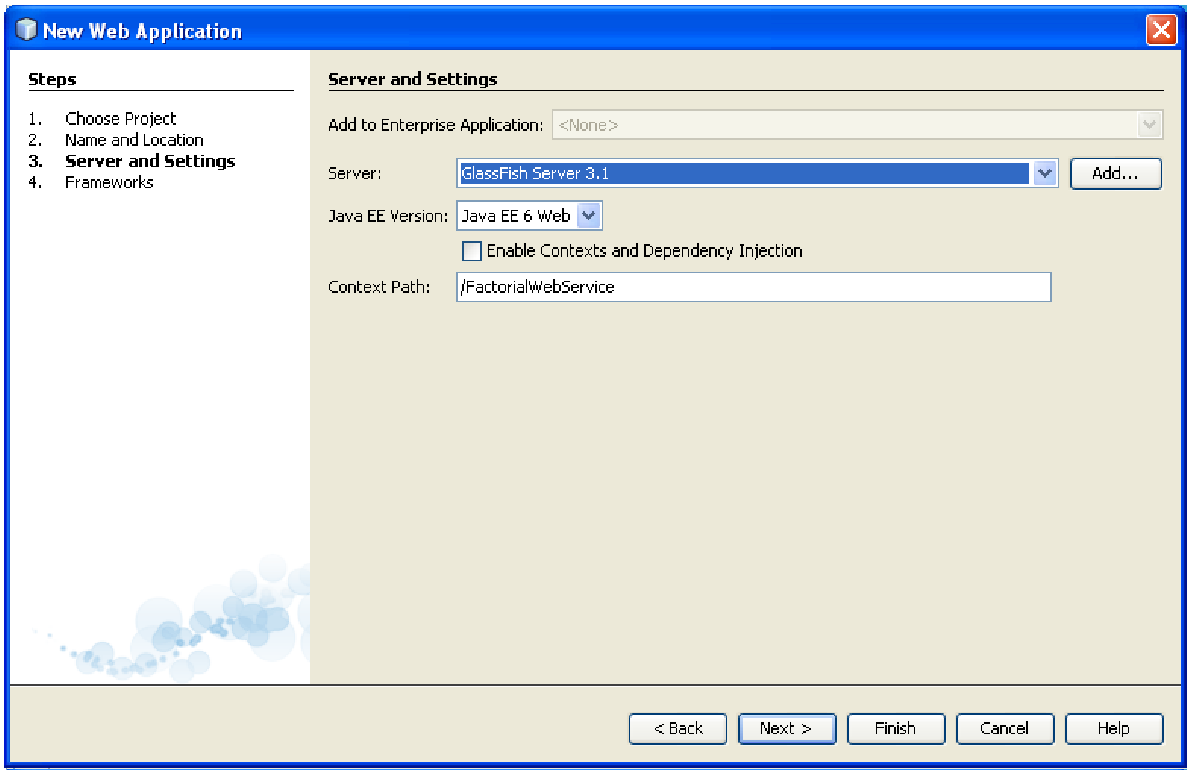
Project Folder: ya.hegdepatil\My Documents\NetBeansProjects\FactorialWebService

☐ 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


New Web Application

Steps

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

Server and Settings

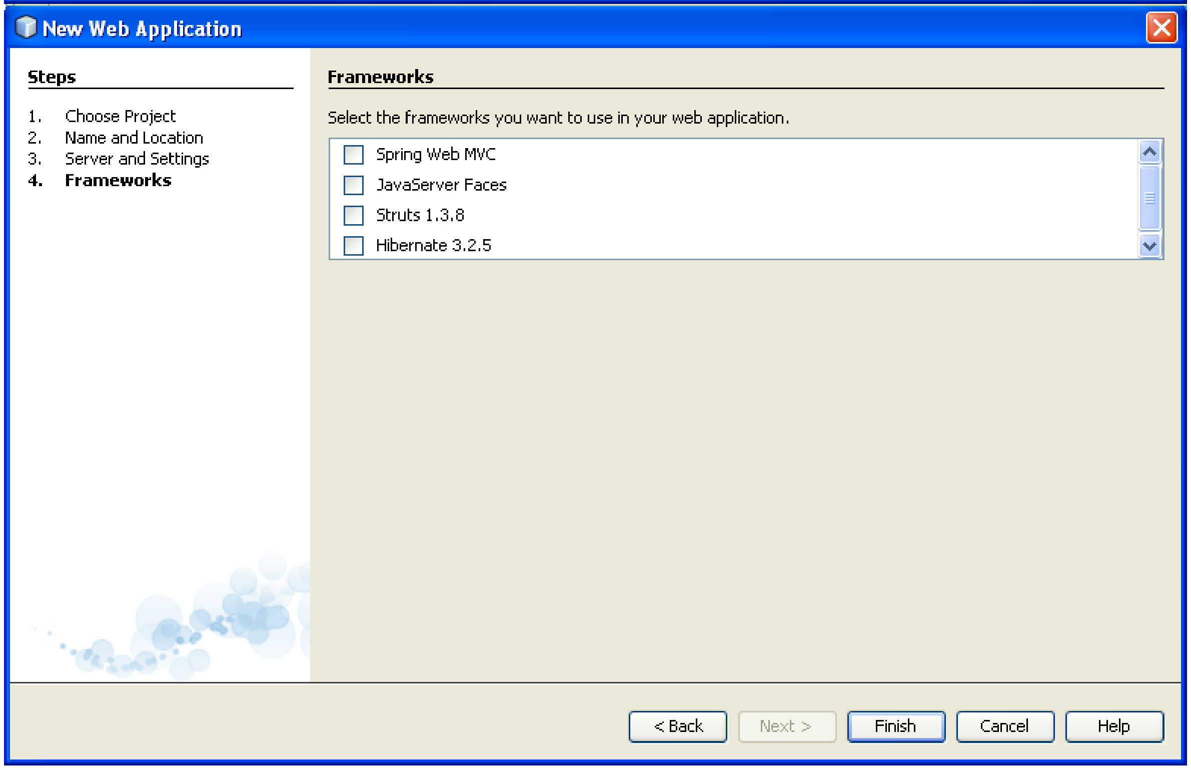
Add to Enterprise Application: <None>

Server: GlassFish Server 3.1

Java EE Version: Java EE 6 Web

[ ] Enable Contexts and Dependency Injection

Context Path: /FactorialWebService


New Web Application

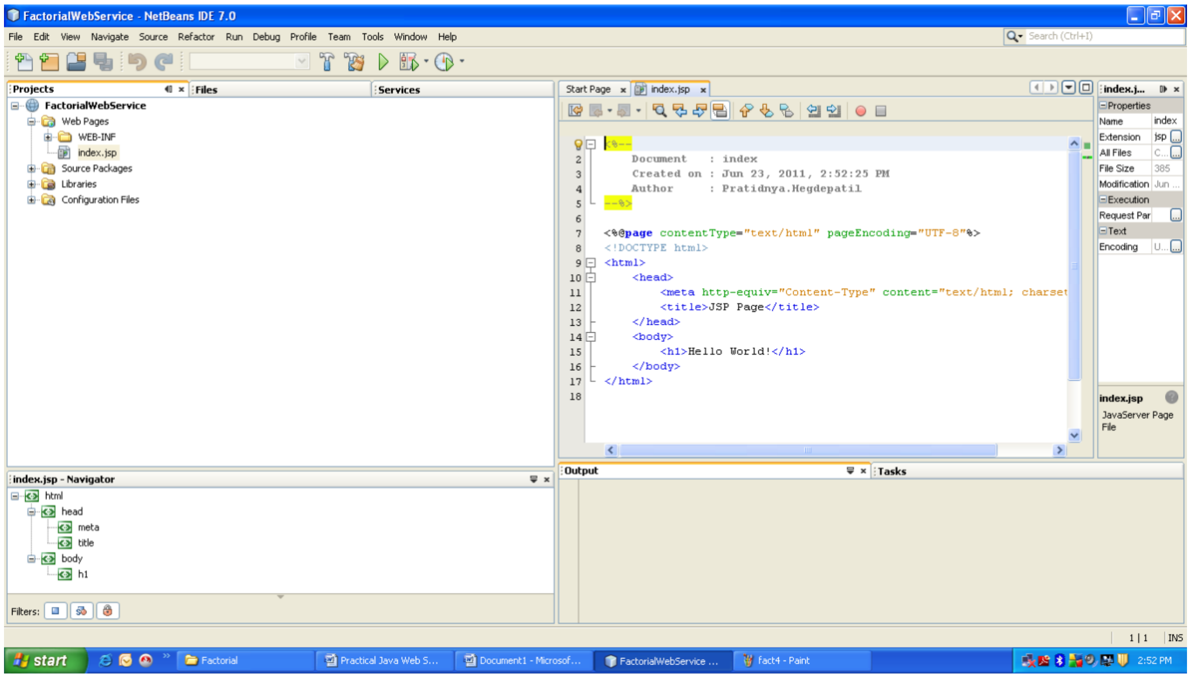
Steps

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.

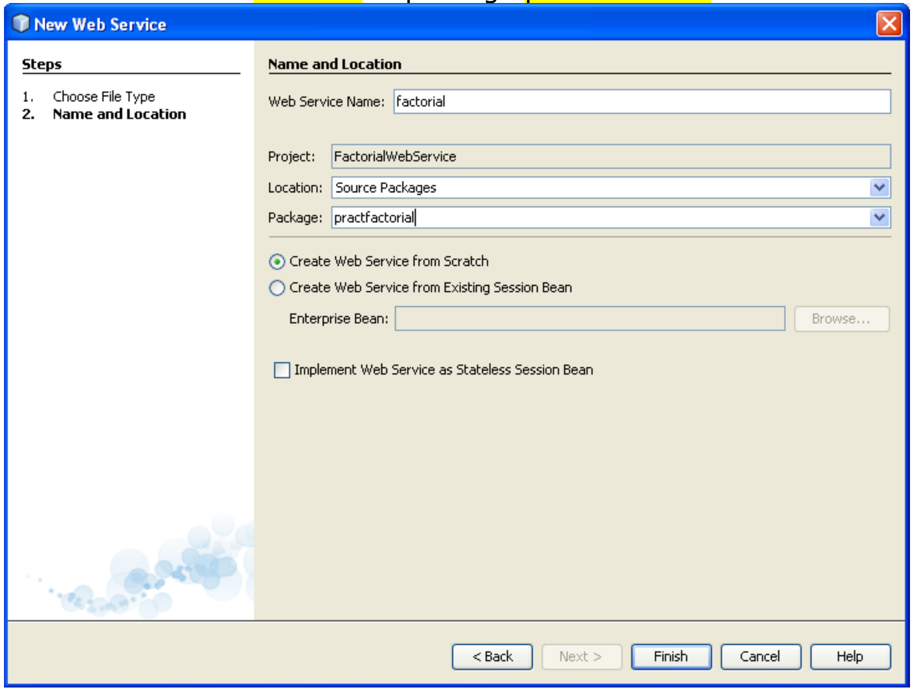
- [ ] Spring Web MVC
- [ ] JavaServer Faces
- [ ] Struts 1.3.8
- [ ] Hibernate 3.2.5



```
<%--
    Document   : index
    Created on : Jun 23, 2011, 2:52:25 PM
    Author     : Pratidnya.Hegdepatil
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

This displays the created application FactorialWebService on the Projects list with the default index.jsp page as Hello World.

Right Click on the Application FactorialWebService and select New->Web Service. Create the web service called factorial in package practfactorial.



New Web Service

Steps

1. Choose File Type
2. **Name and Location**

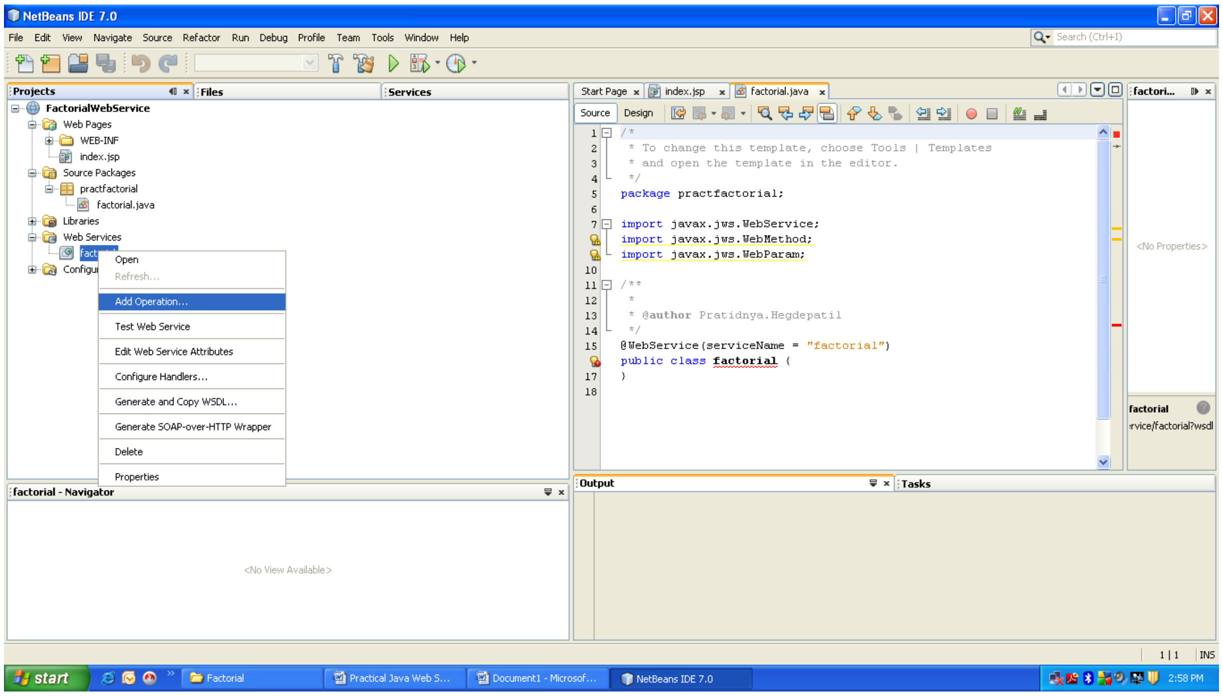
Name and Location

Web Service Name:	factorial
Project:	FactorialWebService
Location:	Source Packages
Package:	practfactorial

[✓ Create Web Service from Scratch]
Create Web Service from Existing Session Bean
Enterprise Bean:

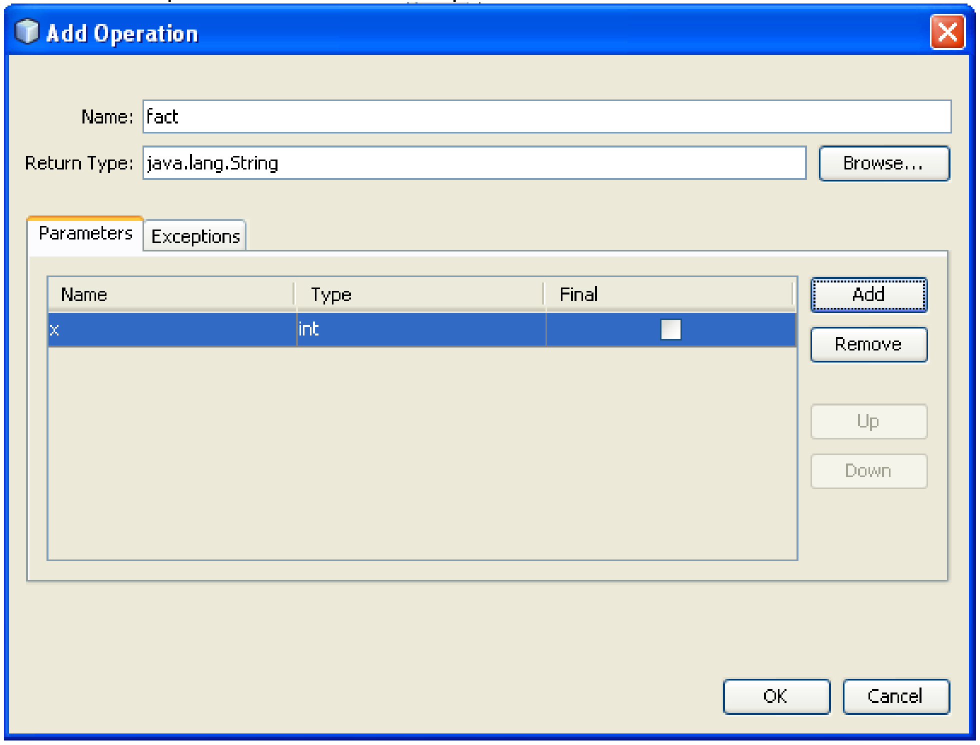
☐ Implement Web Service as Stateless Session Bean

Now Select the created web service factorial and add an operation to it. Right Click->Add Operation.



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package practfactorial;

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

/**
 *
 * @author Pratidnya.Hegdepatil
 */
@WebService(serviceName = "factorial")
public class factorial {
}
```

Create the operation as fact with parameter x as int.

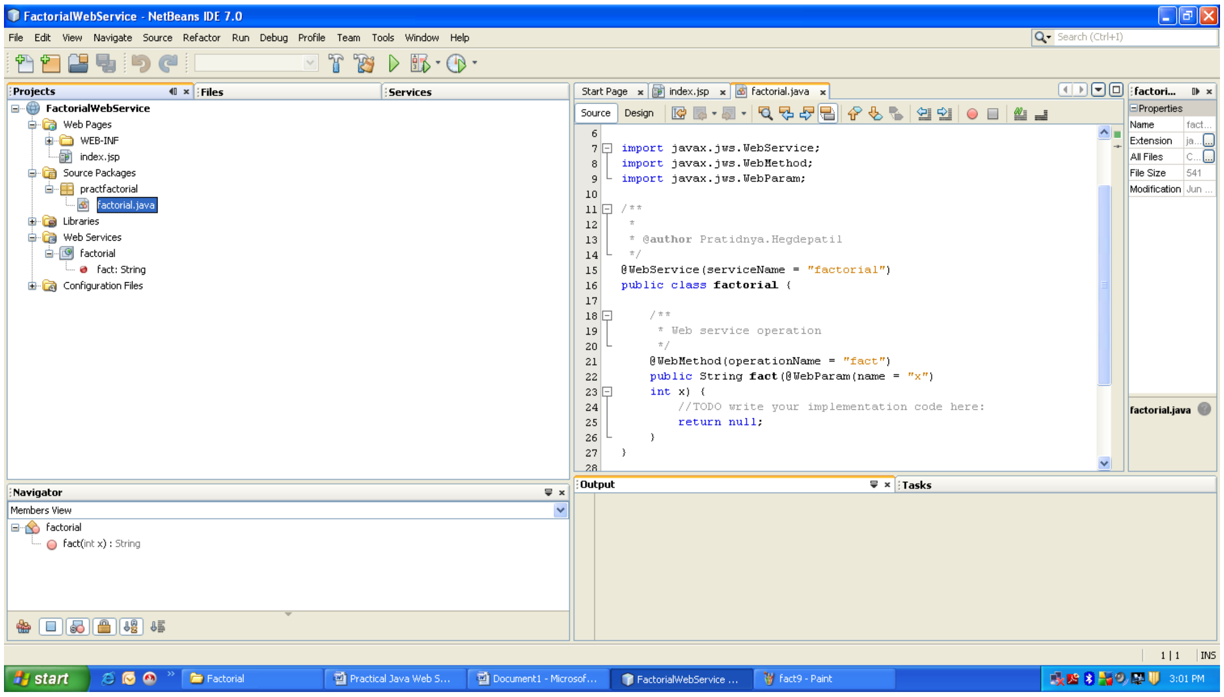


Name: fact

Return Type: java.lang.String

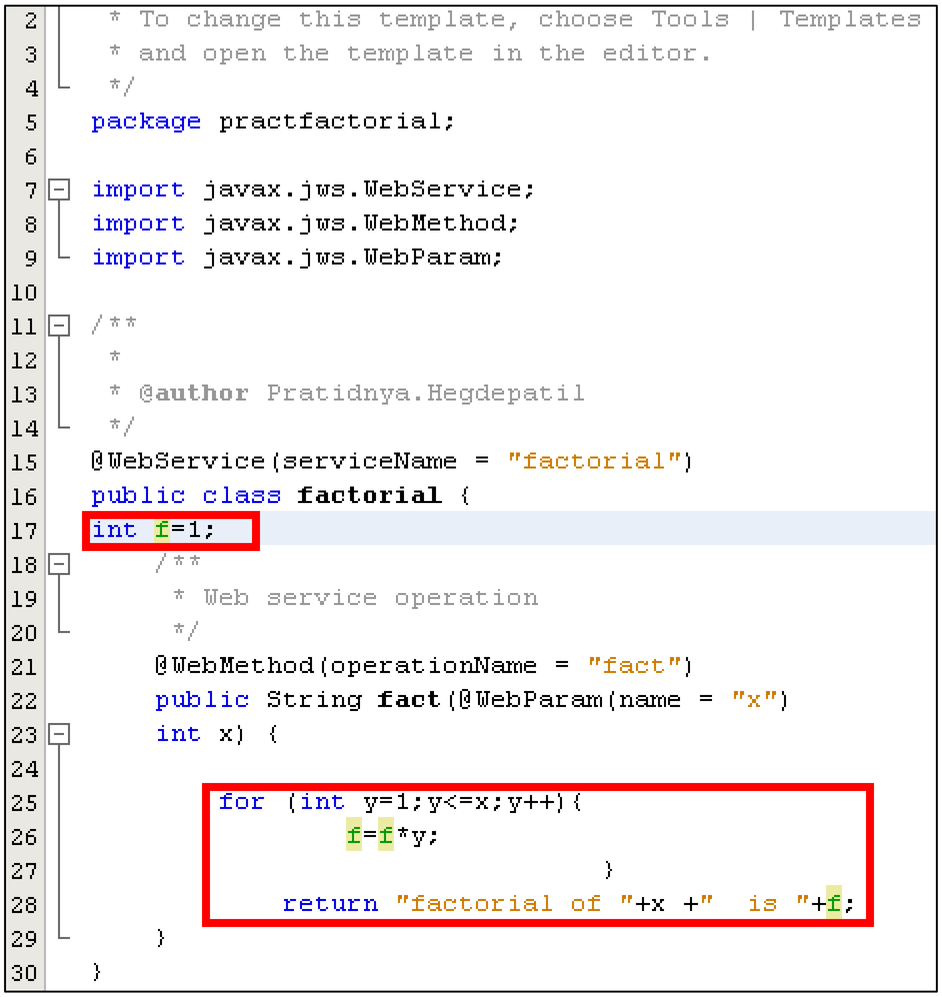
Name	Type	Final
x	int	☐

Now click on the source view Here make some modification in the business logic for factorial calculation we can see the code of factorial.java which is our web service. It has @WebServices as factorial, @WebMethod as fact and @WebParam as x.



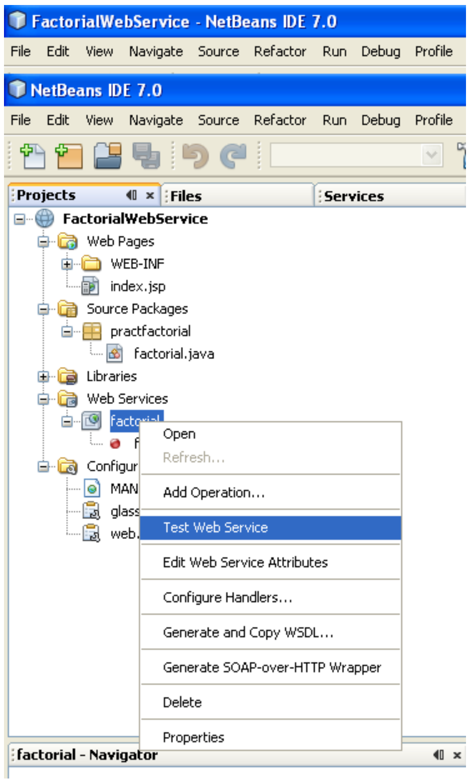
```java
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Pratidnya.Hegdepatil
 */
@WebService(serviceName = "factorial")
public class factorial {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "fact")
    public String fact(@WebParam(name = "x")
    int x) {
        //TODO write your implementation code here:
        return null;
    }
}
```

Add the code in the appropriate place which is highlighted in a red box. This is the logic to calculate factorial given the parameter x by the client service.

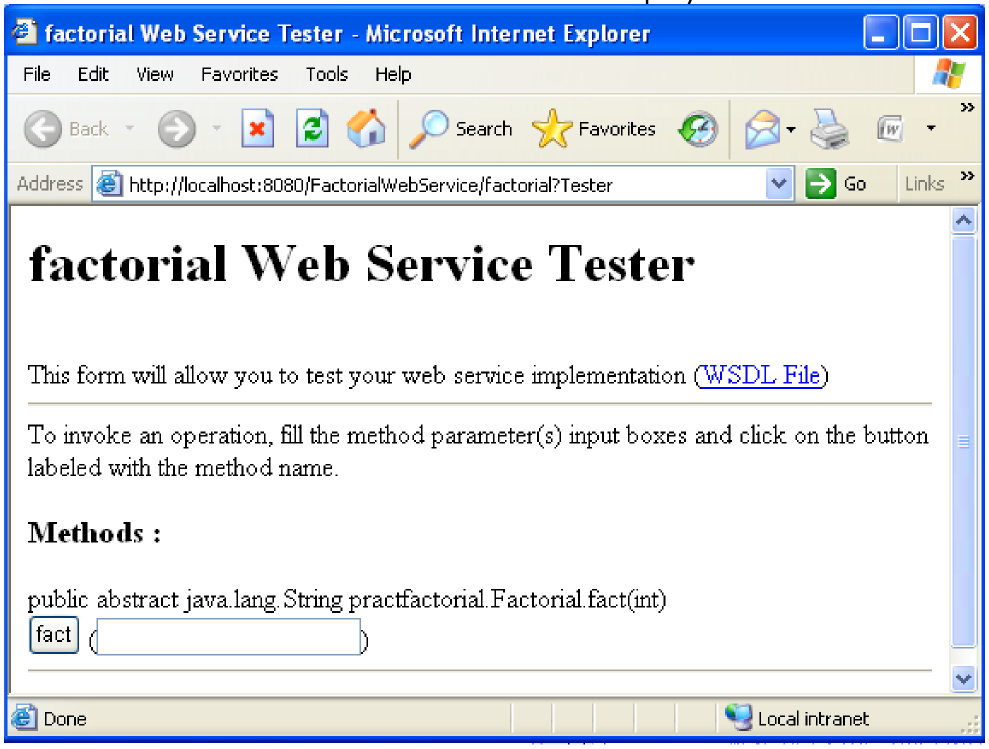


```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package practfactorial;

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

/**
 *
 * @author Pratidnya.Hegdepatil
 */
@WebService(serviceName = "factorial")
public class factorial {
int f=1;
    /**
     * Web service operation
     */
    @WebMethod(operationName = "fact")
    public String fact(@WebParam(name = "x")
    int x) {

        for (int y=1;y<=x;y++){
                f=f*y;
                                }
            return "factorial of "+x +"  is "+f;
    }
}
```

In the project window right click on the application FactorialWebService and Build it. After successfully build then deploy it. This will start the GlassFishServer 3.1.
After the application is up working now select the service factorial and right click on it and select Test Web Service.



It will run the web service on the server and displayed in the browser.


factorial Web Service Tester - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links
Done
Local intranet

http://localhost:8080/FactorialWebService/factorial?Tester

factorial Web Service Tester

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

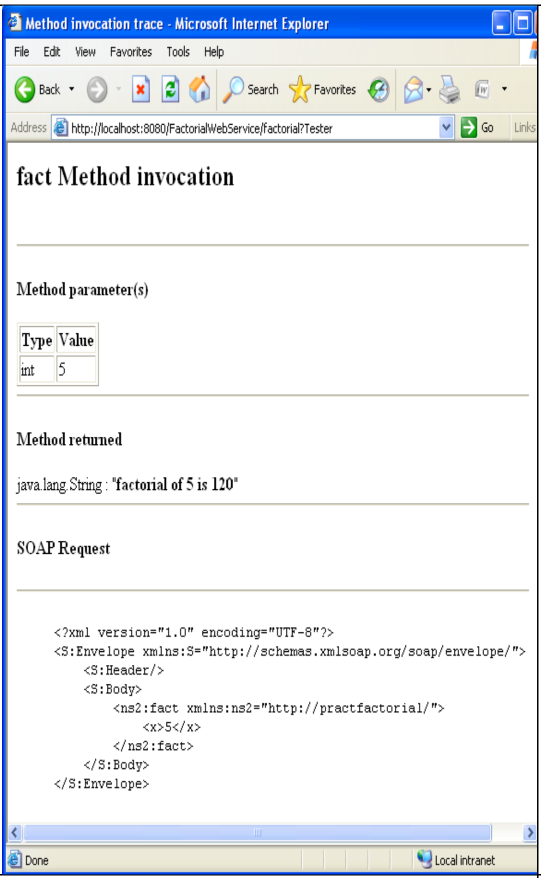
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String practfactorial.Factorial.fact(int)

fact ( )

We can enter a fact number 5 to obtain the answer click on fact button, This invokes the fact operation and displays the SOAP request & response code. The WSDL file can be displayed by clicking on WSDL File in the factorial Web Service Tester page.



fact Method invocation

Method parameter(s)

Type	Value
int	5

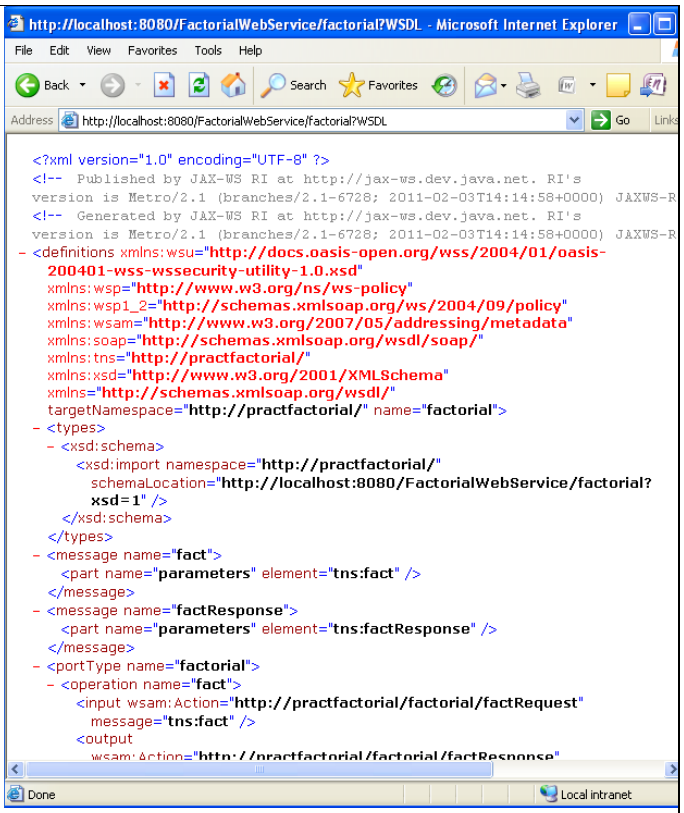
Method returned

java.lang.String : "factorial of 5 is 120"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:fact xmlns:ns2="http://practfactorial/">
            <x>5</x>
        </ns2:fact>
    </S:Body>
</S:Envelope>
```

Fact method invocation



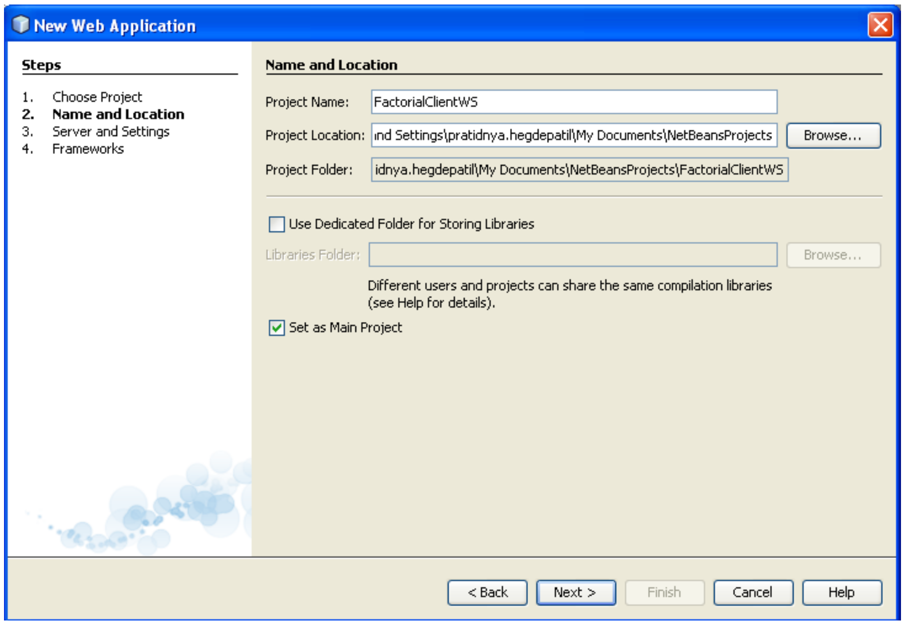
```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's
version is Metro/2.1 (branches/2.1-6728; 2011-02-03T14:14:58+0000) JAXWS-R
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's
version is Metro/2.1 (branches/2.1-6728; 2011-02-03T14:14:58+0000) JAXWS-R
- <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
   200401-wss-wssecurity-utility-1.0.xsd"
   xmlns:wsp="http://www.w3.org/ns/ws-policy"
   xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://practfactorial/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   targetNamespace="http://practfactorial/" name="factorial">
 - <types>
   - <xsd:schema>
       <xsd:import namespace="http://practfactorial/"
         schemaLocation="http://localhost:8080/FactorialWebService/factorial?
         xsd=1" />
     </xsd:schema>
   </types>
 - <message name="fact">
     <part name="parameters" element="tns:fact" />
   </message>
 - <message name="factResponse">
     <part name="parameters" element="tns:factResponse" />
   </message>
 - <portType name="factorial">
   - <operation name="fact">
       <input wsam:Action="http://practfactorial/factorial/factRequest"
         message="tns:fact" />
       <output
         wsam:Action="http://practfactorial/factorial/factResponse"
```

Contents of WSDL file.

Copy the WSDL URL in a safe place as this is required when the client web service is created.
http://localhost:8080/FactorialWebService/factorial?WSDL

- ✓ Part B : To create the client web service which will use the factorial service already created.

Start a new project File->New Project.
Select the New Project as Java Web -> Web Application. Specify the Project Name as FactorialClientWS.



New Web Application

Steps

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

Name and Location

Project Name: FactorialClientWS

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

Project Folder: idnya.hegdepatil\My Documents\NetBeansProjects\FactorialClientWS

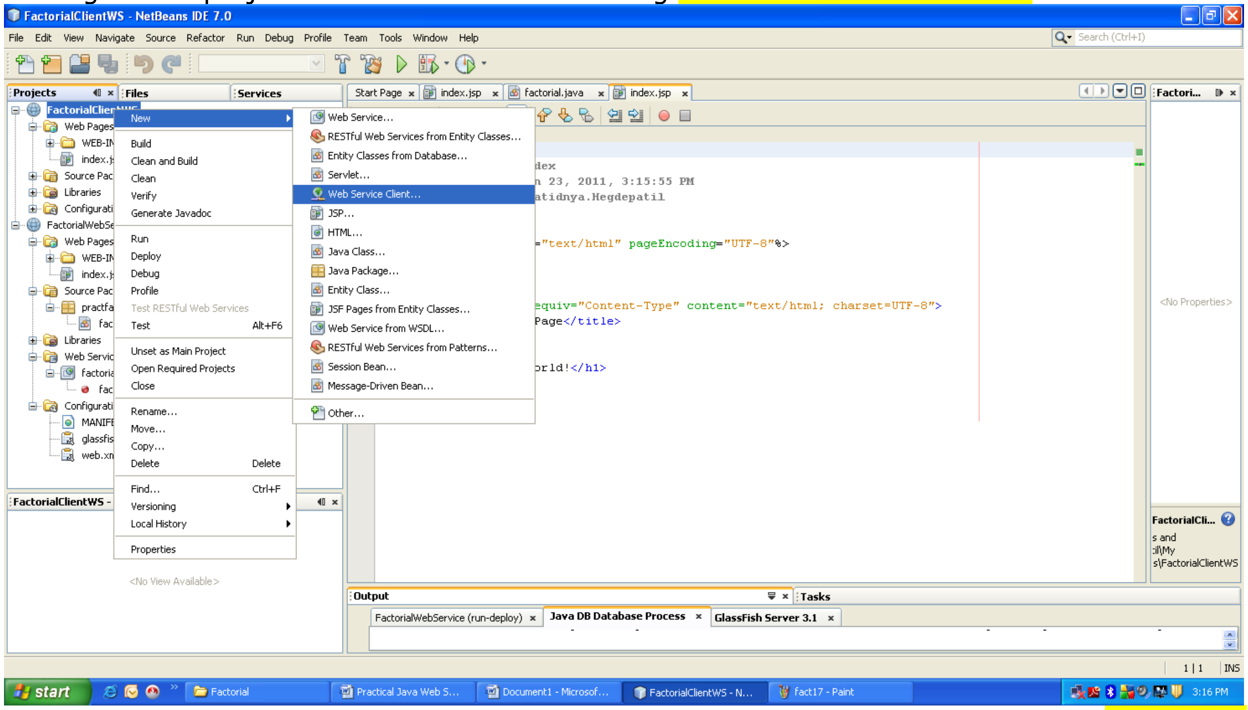
[ ] Use Dedicated Folder for Storing Libraries

Libraries Folder:

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

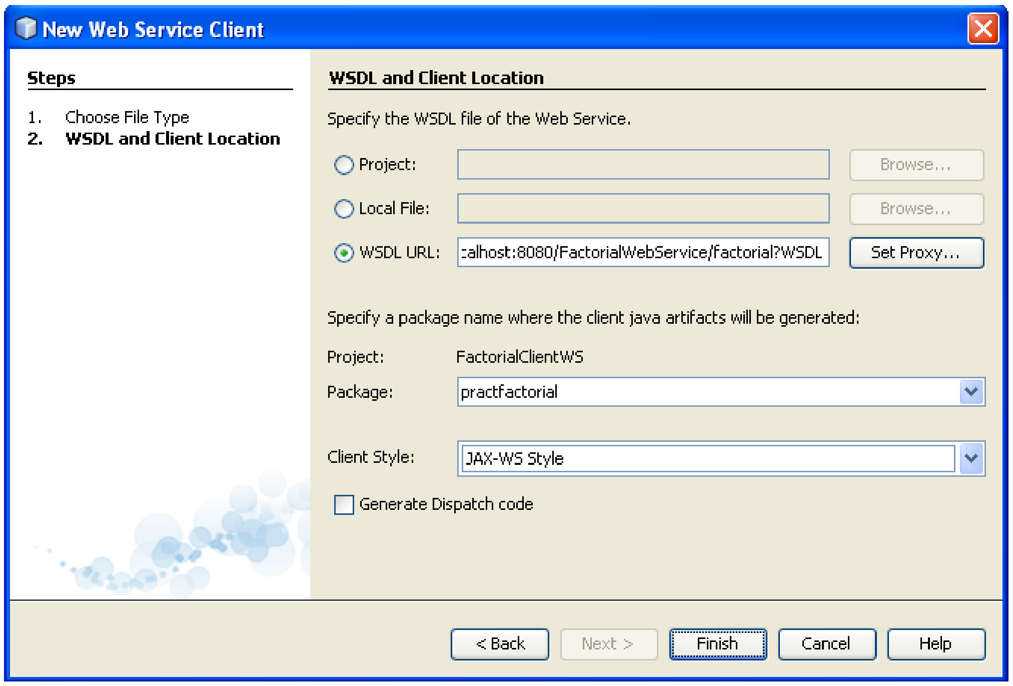
[✓] Set as Main Project

After the client project application is created we have to add a web service to it by right clicking on the project FactorialClientWS selecting New->Web Service Client.



```
dex
n 23, 2011, 3:15:55 PM
atidnya.Hegdepatil

="text/html" pageEncoding="UTF-8"%>

equiv="Content-Type" content="text/html; charset=UTF-8">
Page</title>

orld!</h1>
```

Add the WSDL URL which was displayed when creating the service and paste it in WSDL URL http://localhost:8080/FactorialWebService/factorial?WSDL. Specify the package as practfactorial.



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:
- Local File:
- WSDL URL: :alhost:8080/FactorialWebService/factorial?WSDL

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

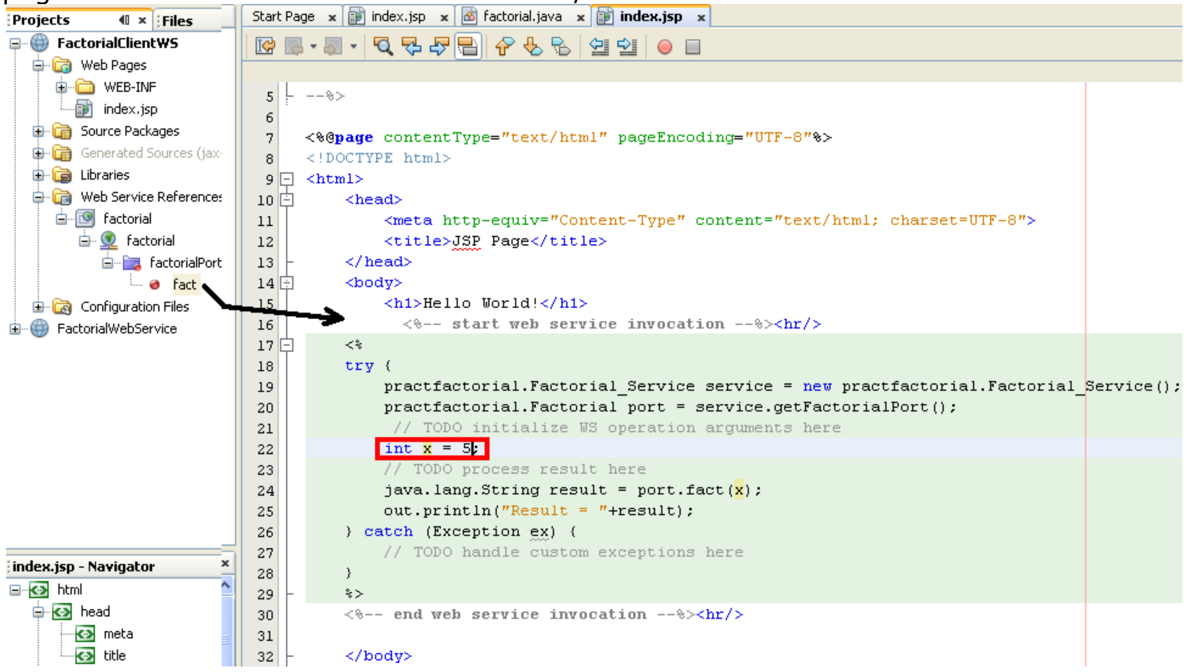
Project: FactorialClientWS

Package: practfactorial

Client Style: JAX-WS Style

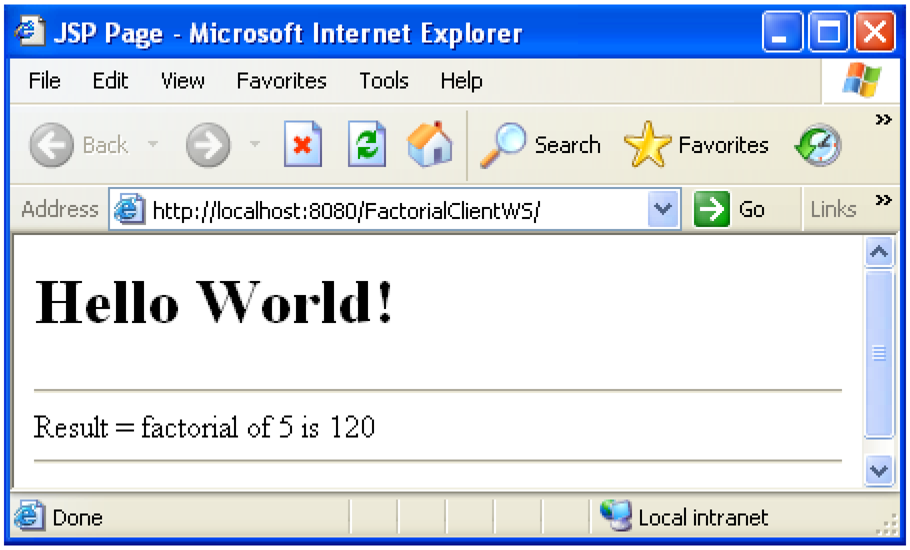
Generate Dispatch code

After this the client service fact has to be added to the index.jsp page as output for the client application. For this select the fact operation and simply drag it into the index.jsp page content after the <h1>Hello World </h1> html code and rewrite the value of x as 5.



```
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
          <%-- start web service invocation --%><hr/>
    <%
    try {
        practfactorial.Factorial_Service service = new practfactorial.Factorial_Service();
        practfactorial.Factorial port = service.getFactorialPort();
         // TODO initialize WS operation arguments here
        int x = 5;
        // TODO process result here
        java.lang.String result = port.fact(x);
        out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>

    </body>
```

Now run the project by right clicking on FactorialClientWS and selecting Run. This will display a web page with the contents Hello World And the result of the factorial 5 which is 120. We can re run the project with different factorial values.


JSP Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/FactorialClientWS/ Go Links
Done Local intranet

Hello World!

Result = factorial of 5 is 120

This ends the factorial web service practical.

Also create a GUI on client side using jsp page and call the service.

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 within 2 hours of practical session.)

Roll no.:	**Name:**
Class:	**Batch:**
Date of Experiment:	**Date of Submission:**
Grade:	

1. **Program scenario and Program code: (**Paste you program code)
   a. Service code (.java) and client code (.jsp)
2. **Output: (**Paste your program input and output screen shots)
   a. JSP page output http://localhost:8080/FactorialClientWS/
   b. The Soap Request & Response http://localhost:8080/FactorialWebService/factorial?Tester
   c. WSDL Document http://localhost:8080/FactorialWebService/factorial?WSDL

3. **Observation, learning and conclusion:** Your learning from this practical.
4. **Questions:**

1. What is JAX-WS?
2. What Is the Difference Between a Web Service and an SOA Service?
3. What are the core principles that form the baseline foundation for SOA?

Note: Save your project FactorialWebService_your Roll no
Doc file should be saved with Practical3_Rollno.doc/docx