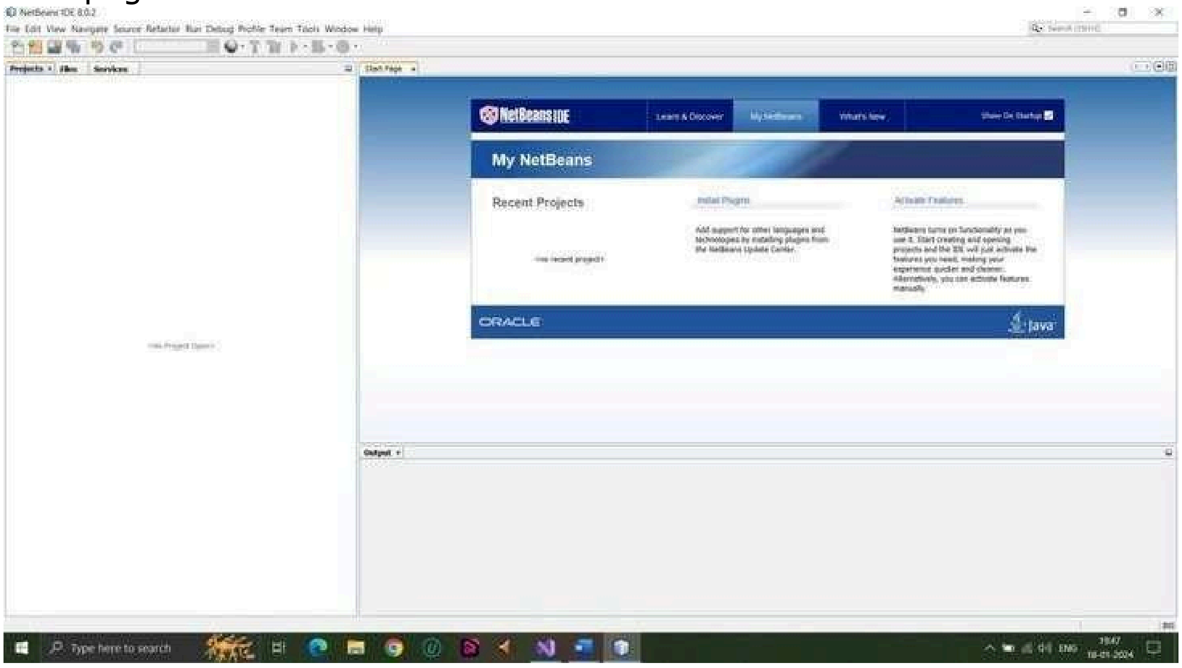
Practical 1

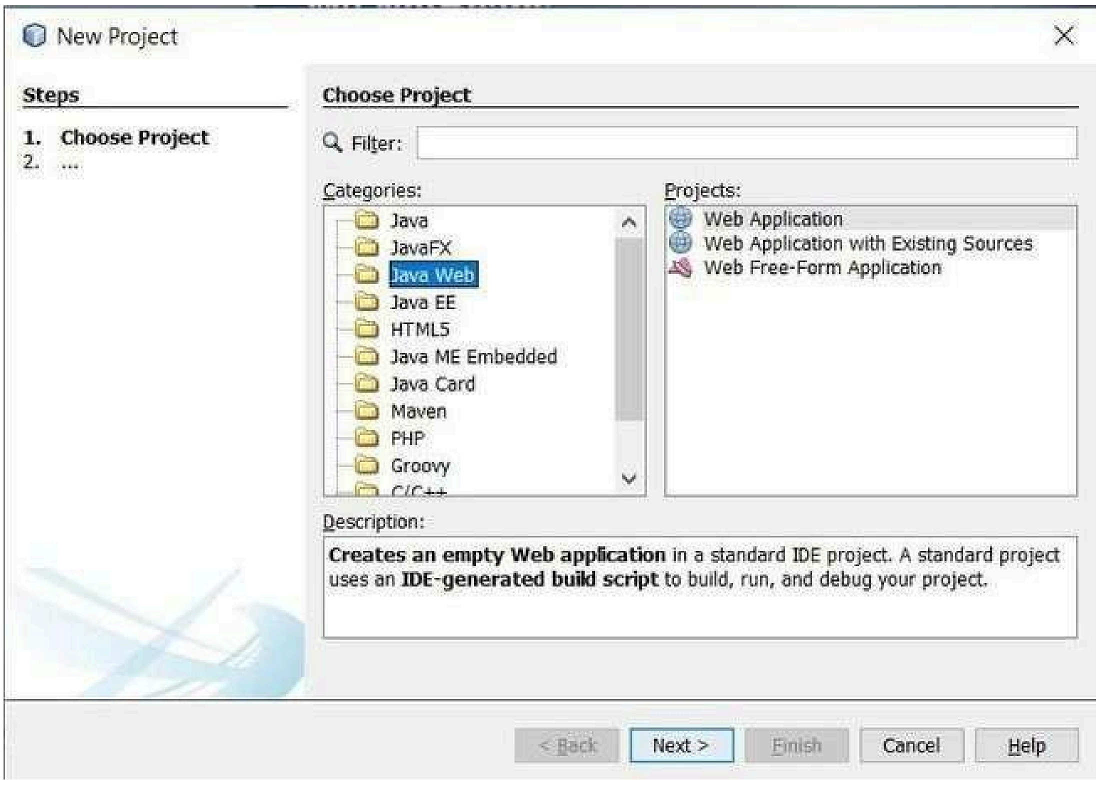
Aim: Create a Simple SOAP service for consuming java web services in java.

Steps:

1. Open the NetBeans , and you will get the following screen. Close the start page.



2. Now click on the file tab and click on new project you will get the following screen :



New Project

Steps

1. Choose Project
2. ...

Choose Project

Filter:

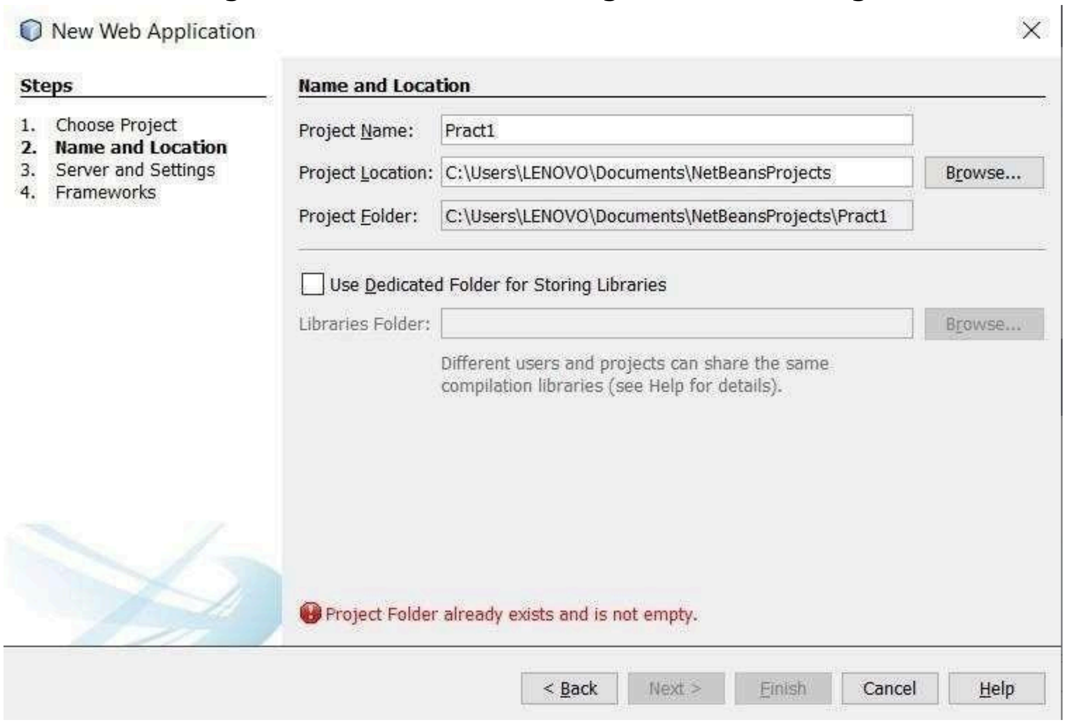
Categories: Java, JavaFX, Java Web, Java EE, HTML5, Java ME Embedded, Java Card, Maven, PHP, Groovy, C/C++

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.

< Back | Next > | Finish | Cancel | Help

3. In Categories select Java Web and in Projects , Select Web Application .After selecting click on next You will get the following window:



New Web Application

Steps

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

Name and Location

Project Name: Pract1

Project Location: C:\Users\LENOVO\Documents\NetBeansProjects

Project Folder: C:\Users\LENOVO\Documents\NetBeansProjects\Pract1

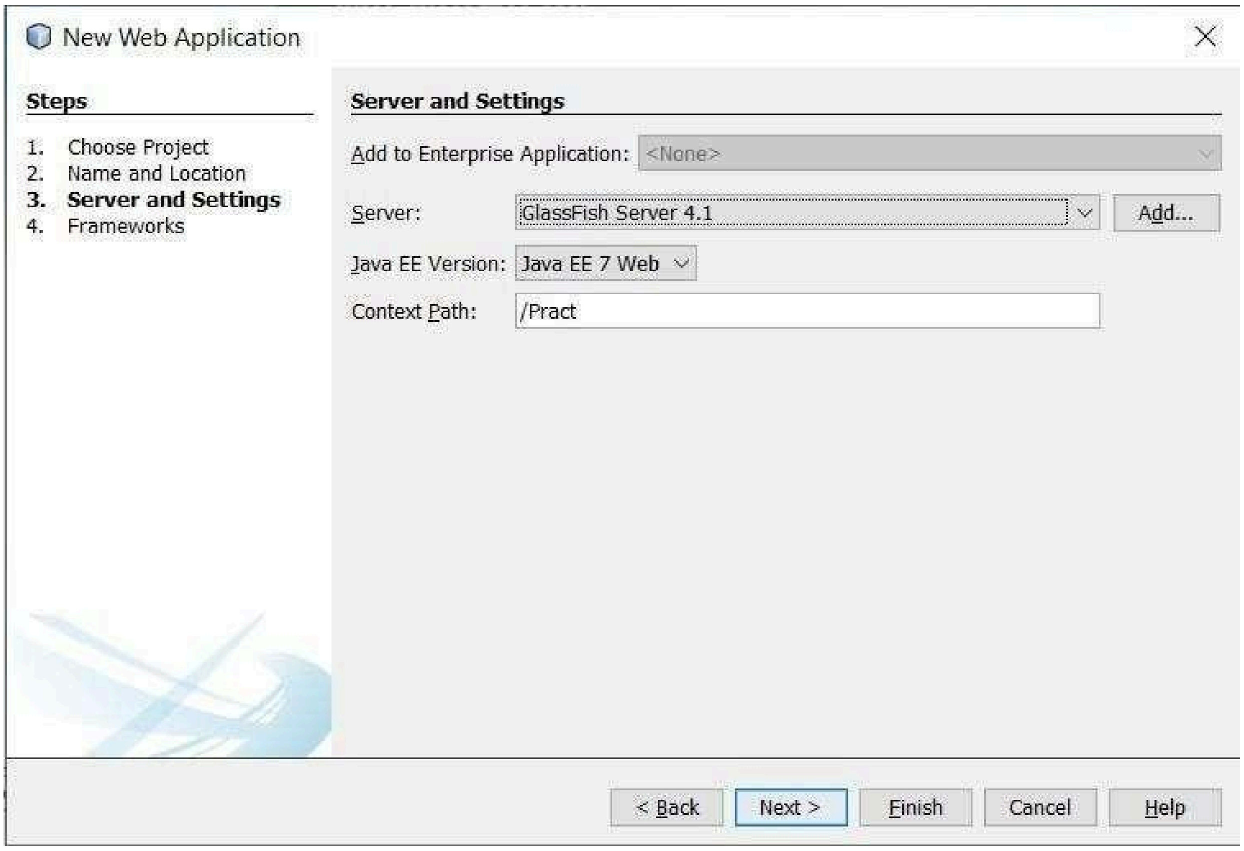
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

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

Project Folder already exists and is not empty.

4. Now Give name to the Project Name: , and click on next. You will get the following window then again click on finish



New Web Application

Steps

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

Server and Settings

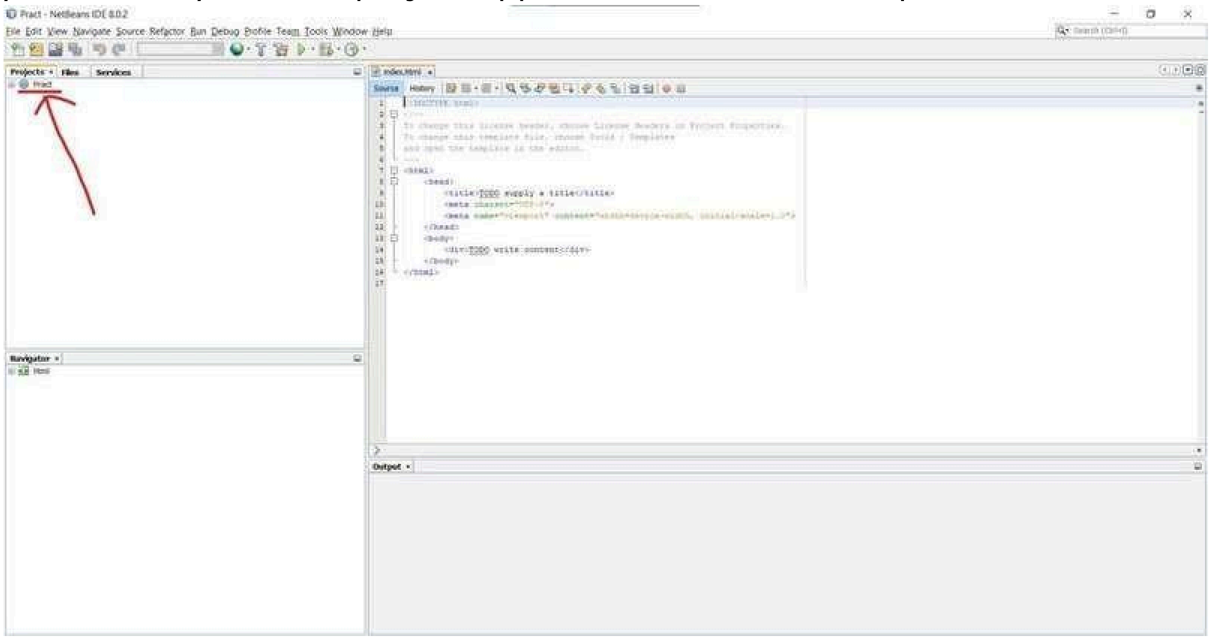
Add to Enterprise Application: <None>

Server: GlassFish Server 4.1

Java EE Version: Java EE 7 Web

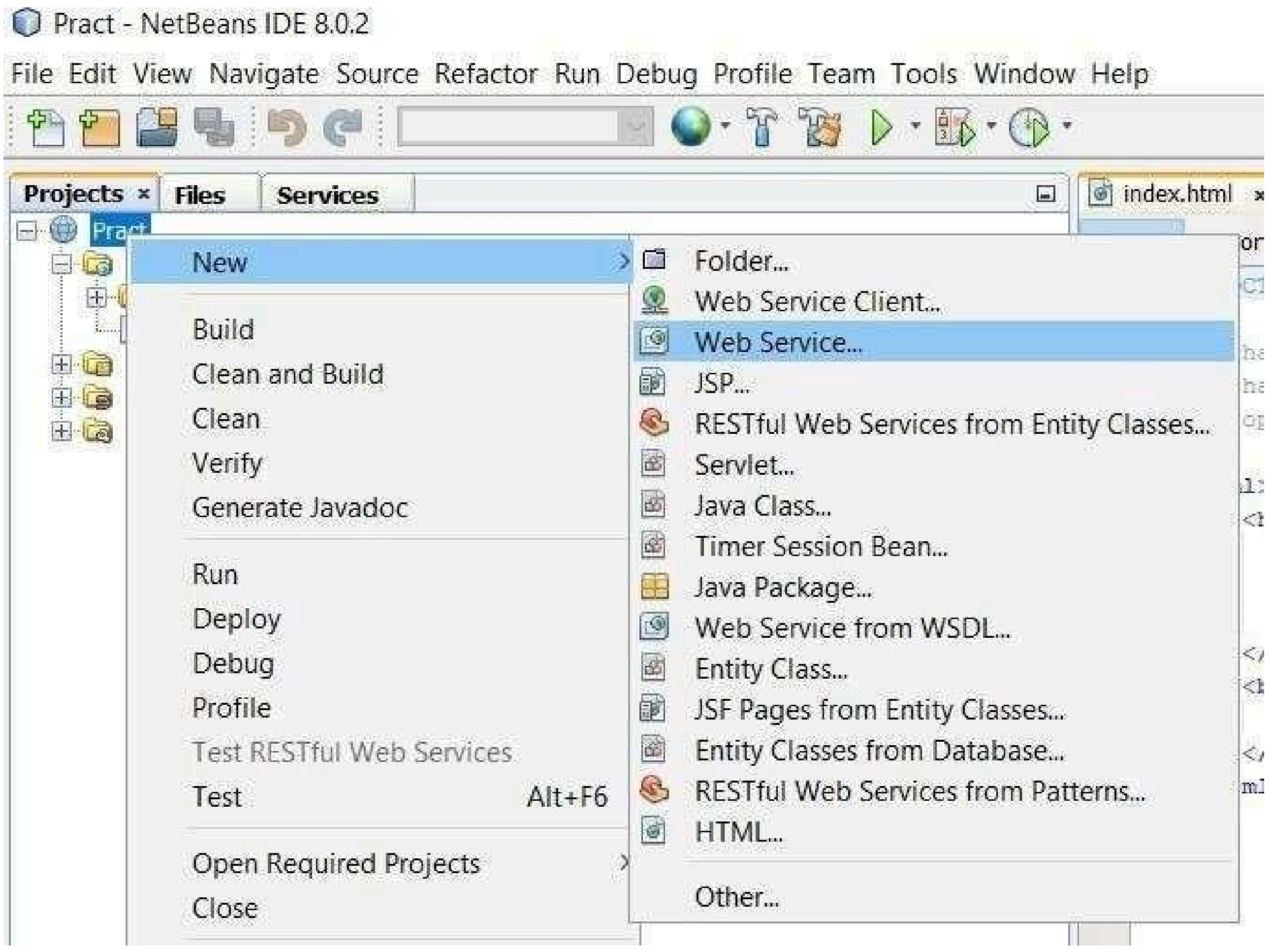
Context Path: /Pract

5. You will get the following screen now carefully see in projects section your recently created project appears double click to expand it:

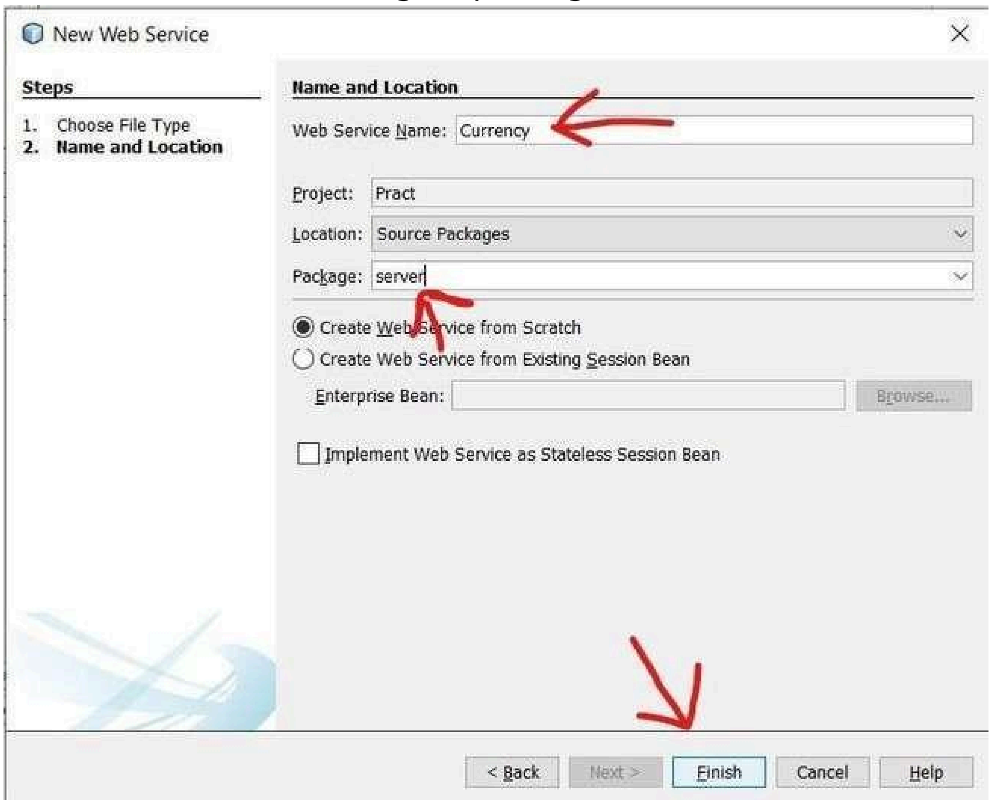

Pract - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
Services
Pract
Navigator
html
index.html
Source
History
Output

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

6. Now Right click on the project and select new and then select Web Service ,As shown Below



7. After clicking Web Service the following window should appear , now give name to web service and give package as “server”. As shown in the image



New Web Service

Steps

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

Name and Location

Web Service Name: Currency

Project: Pract

Location: Source Packages

Package: server

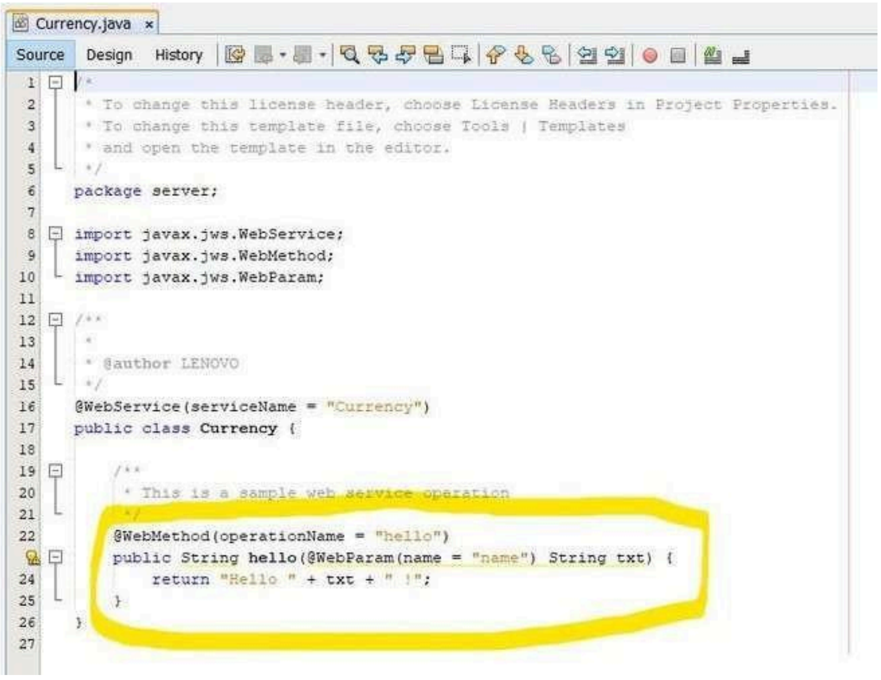
● Create Web Service from Scratch

○ Create Web Service from Existing Session Bean

Enterprise Bean:

☐ Implement Web Service as Stateless Session Bean

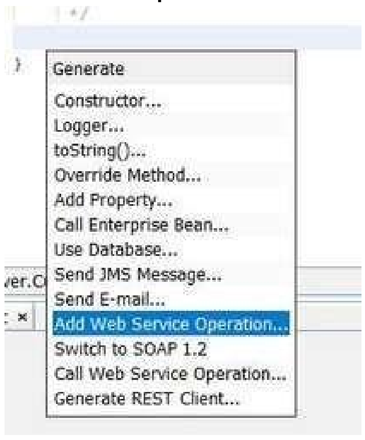
After clicking finish you should get the following window erase the mentioned code



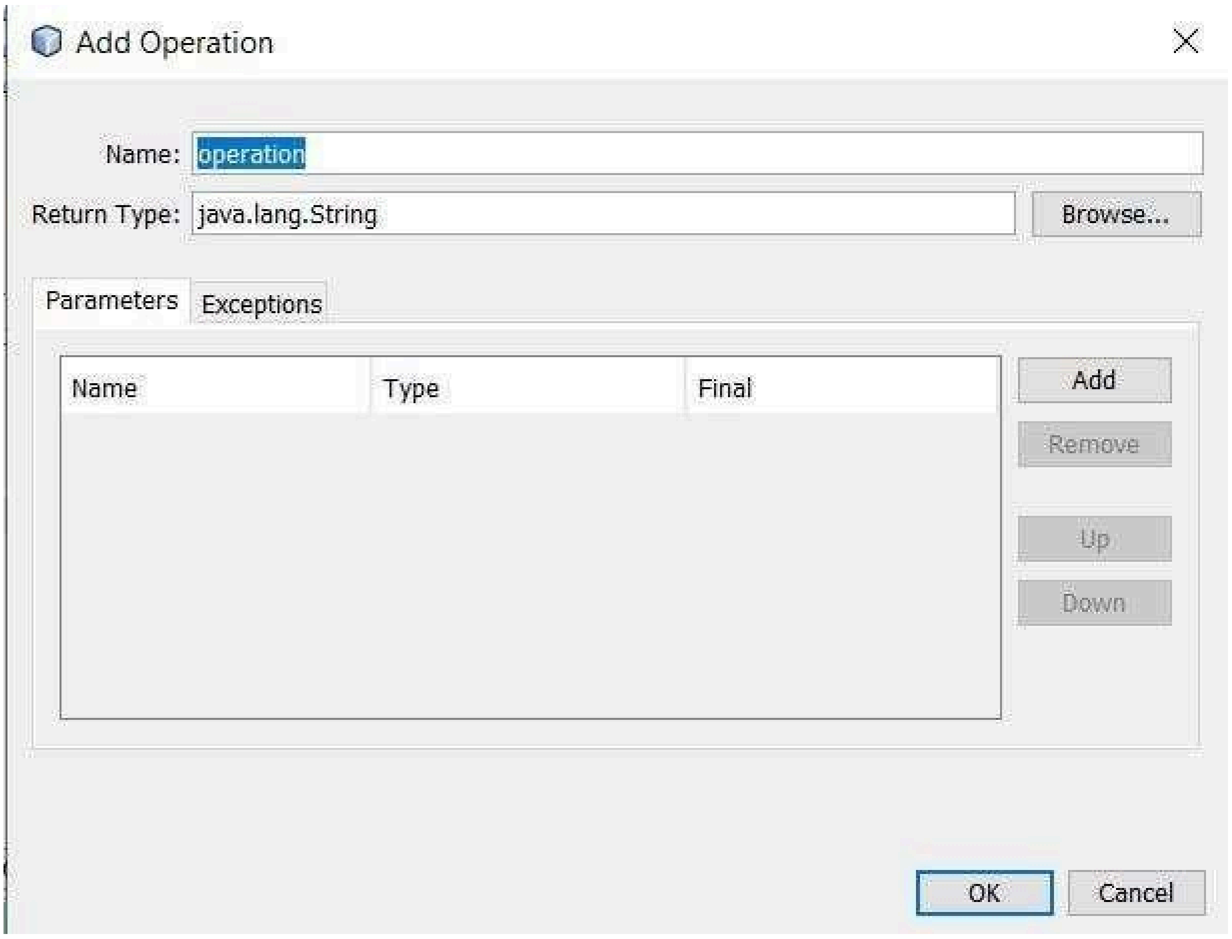
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package server;

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

/**
 *
 * @author LENOVO
 */
@WebService(serviceName = "Currency")
public class Currency {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

8. Now right click anywhere and click on insert code and select Add Web

Service Operation



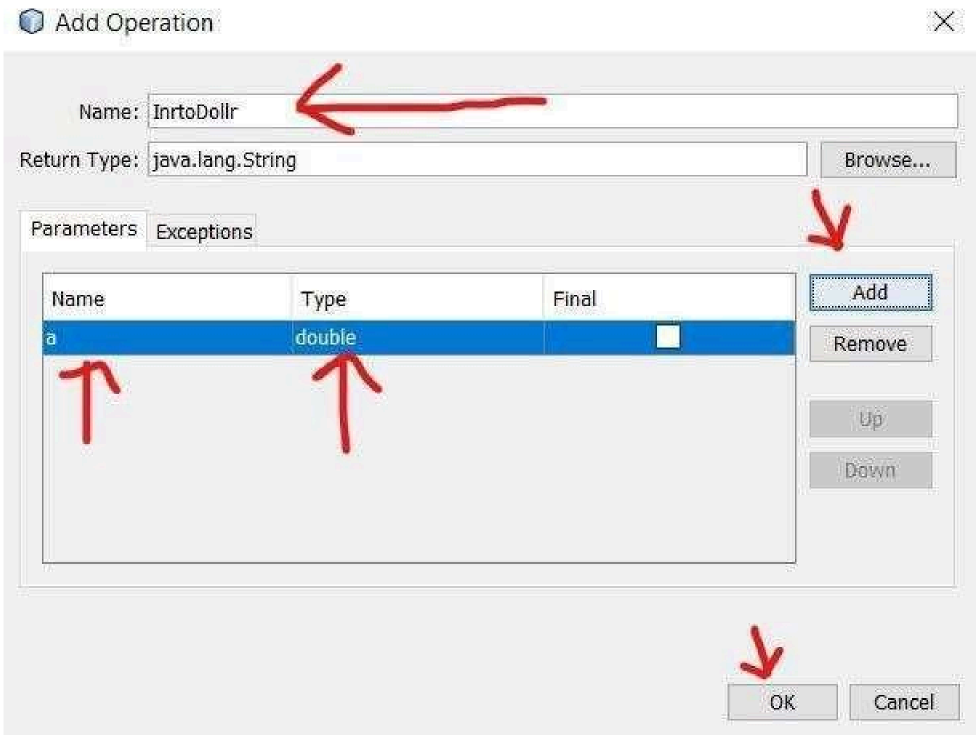
9. The following window would appear :
   Just give name to the method or operation and click on add button to add parameters to the method as here we are converting dollar to rupees we should need only one parameter .


Add Operation

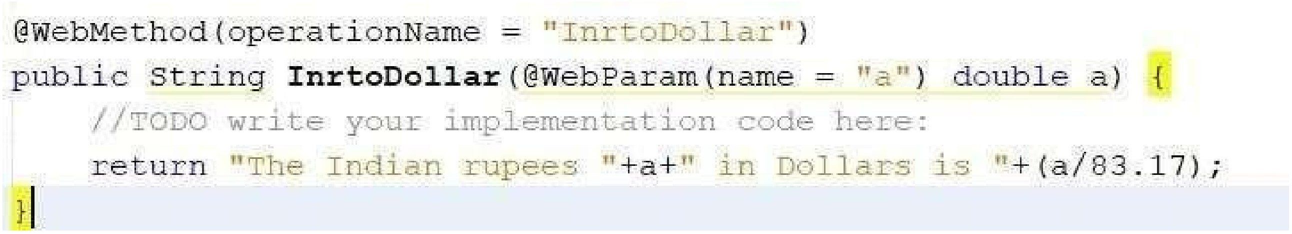
Name: operation

Return Type: java.lang.String

10. Give the name to the variable select data type as double and click on ok as shown below:

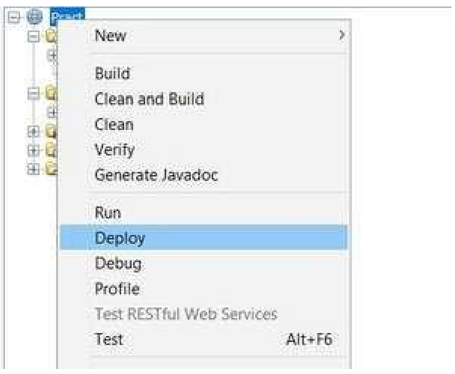


11. After clicking ok code will auto generated , make changes In that code as mentioned below:

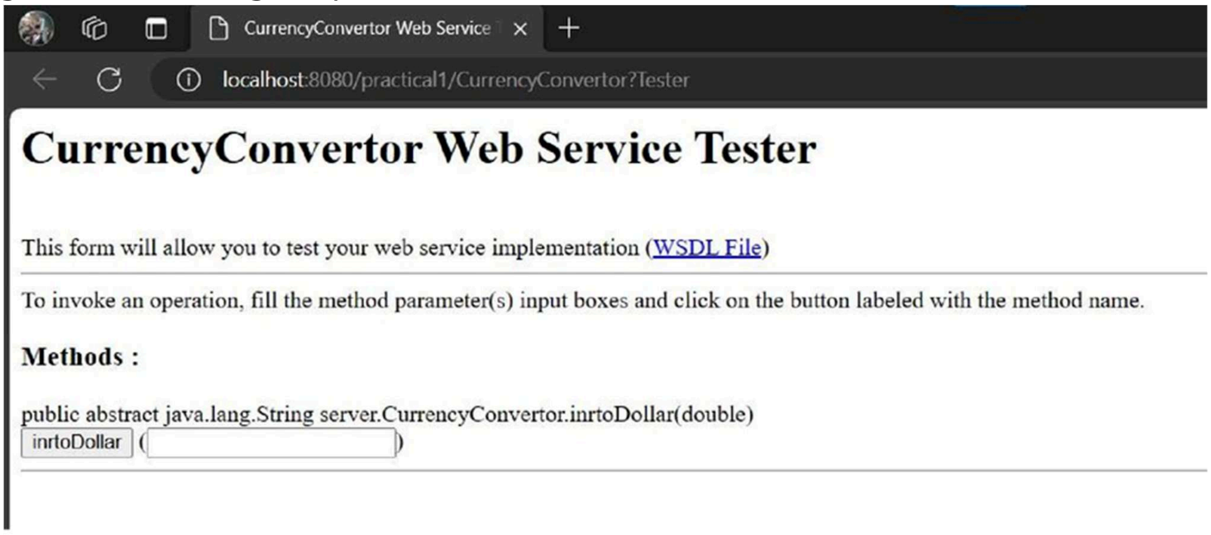


```
@WebMethod(operationName = "InrtoDollar")
public String InrtoDollar(@WebParam(name = "a") double a) {
    //TODO write your implementation code here:
    return "The Indian rupees "+a+" in Dollars is "+(a/83.17);
}
```

12. Now our web service is ready now right click on project and click on deploy



13. Now right click on web service and click on “test web service” you will get the following output:



localhost:8080/practical1/CurrencyConvertor?Tester

CurrencyConvertor Web Service Tester

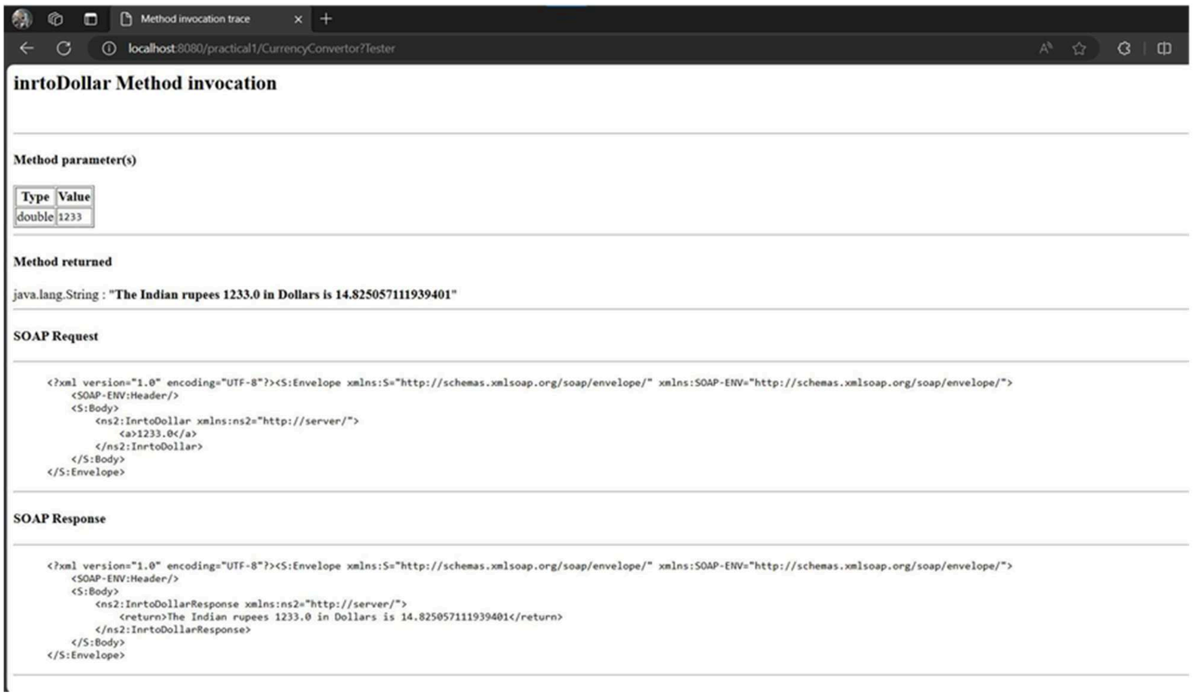
This form will allow you to test your web service implementation (WSDL File)

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 server.CurrencyConvertor.inrtoDollar(double)

inrtoDollar ( )



localhost:8080/practical1/CurrencyConvertor?Tester

inrtoDollar Method invocation

Method parameter(s)

Type	Value
double	1233

Method returned

java.lang.String : **"The Indian rupees 1233.0 in Dollars is 14.825057111939401"**

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:InrtoDollar xmlns:ns2="http://server/">
            <a>1233.0</a>
        </ns2:InrtoDollar>
    </S:Body>
</S:Envelope>
```

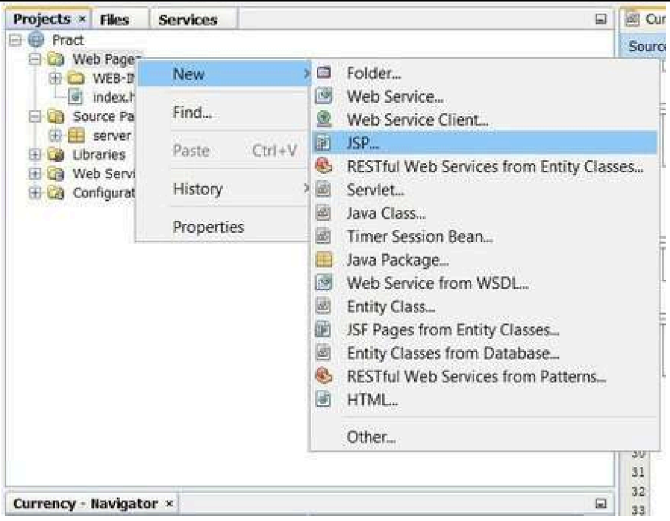
SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:InrtoDollarResponse xmlns:ns2="http://server/">
            <return>The Indian rupees 1233.0 in Dollars is 14.825057111939401</return>
        </ns2:InrtoDollarResponse>
    </S:Body>
</S:Envelope>
```

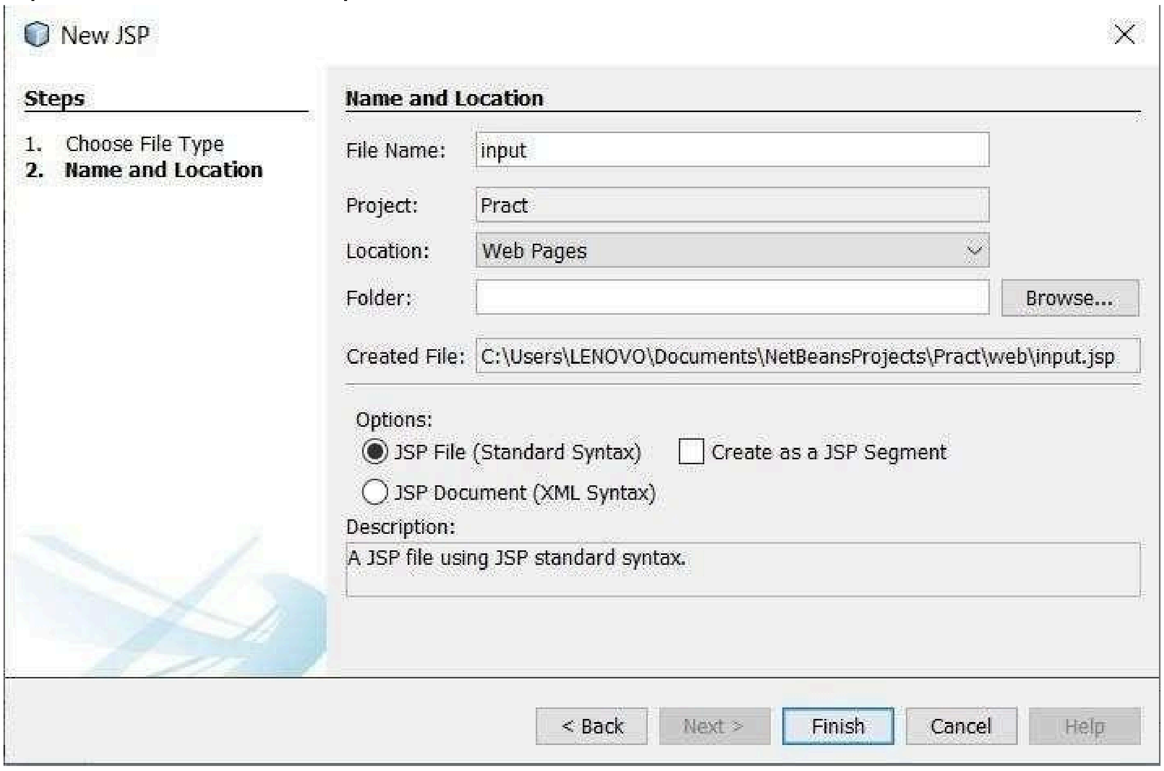
So this is how we created our web service and deployed it.

Creating a java Client using jsp

14. Now our web service is successfully deployed. Right click on web pages and select new and select jsp as shown below:



15. Give name and click on finish as shown below do it 2 times on for input and one for output


New JSP

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

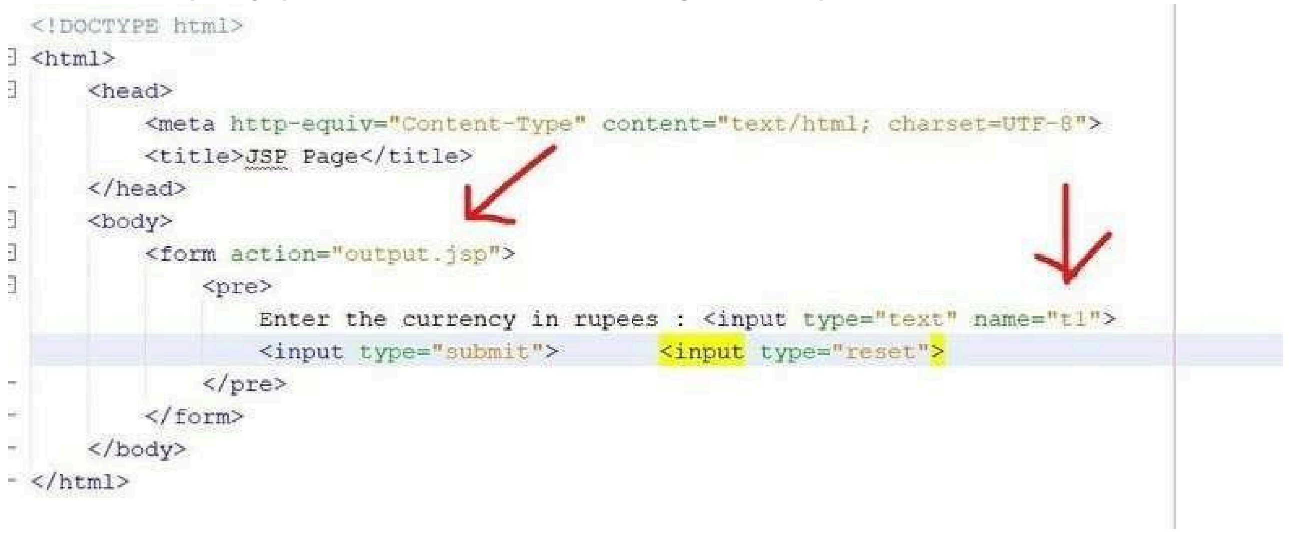
Name and Location

File Name: input
Project: Pract
Location: Web Pages
Folder:
Created File: C:\Users\LENOVO\Documents\NetBeansProjects\Pract\web\input.jsp

Options:
[✓ JSP File (Standard Syntax)] ☐ Create as a JSP Segment
JSP Document (XML Syntax)

Description:
A JSP file using JSP standard syntax.

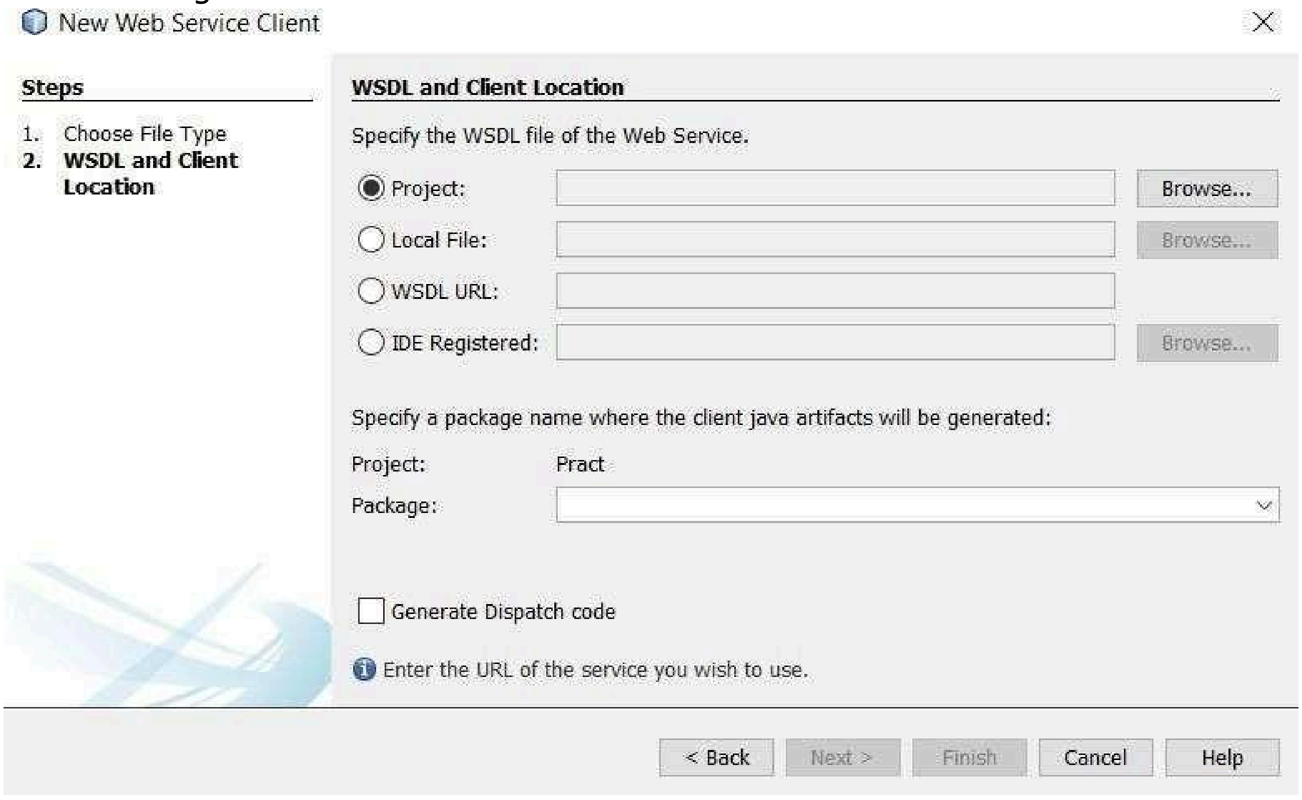
16. In input.jsp create a form for taking user input as shown below:



```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="output.jsp">
            <pre>
                Enter the currency in rupees : <input type="text" name="t1">
                <input type="submit">       <input type="reset">
            </pre>
        </form>
    </body>
</html>
```

In this code set **action = the jsp file where u want output and give name to textbox input.**

17. Now we have to create web service client, for same right click on project and select new thenselect web service client you will get the following screen:



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: [ ]
- ( ) IDE Registered: [ ] Browse...

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

Project: Pract

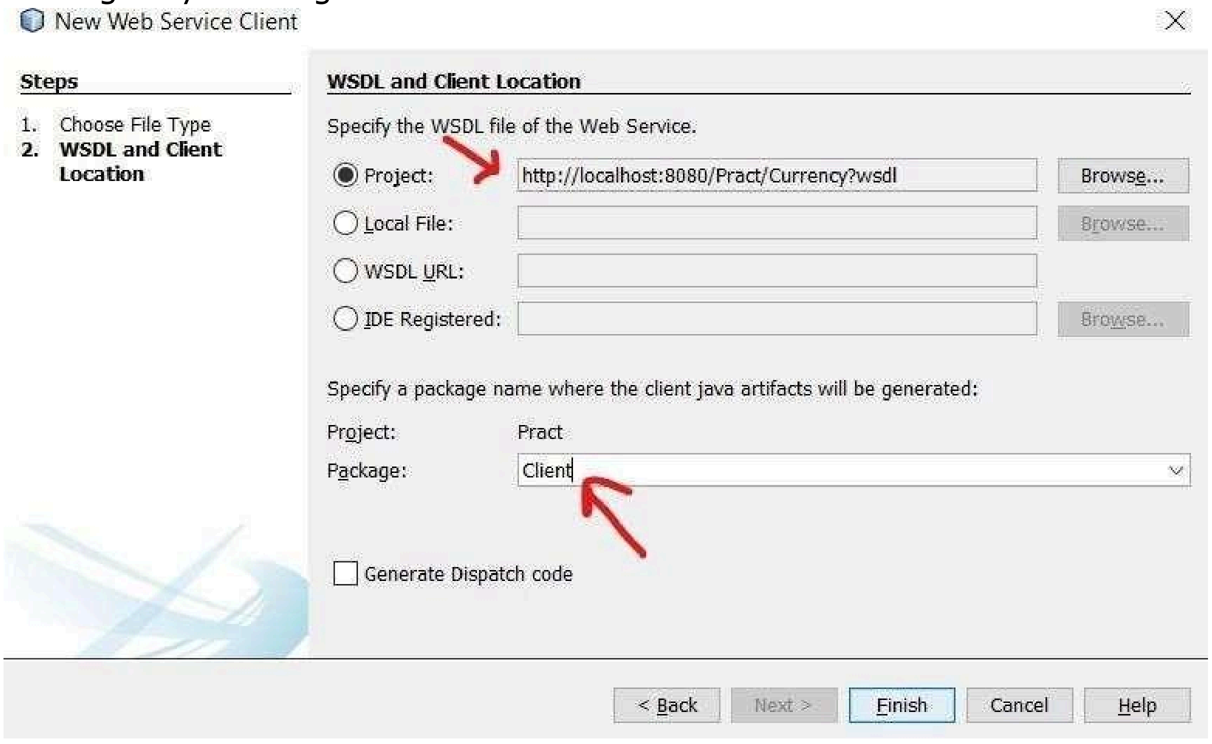
Package: [ ]

☐ Generate Dispatch code

Enter the URL of the service you wish to use.

18. At this step click on browse and select your project and click ok after

clicking ok you will get wsdl url as shown below:


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:] http://localhost:8080/Pract/Currency?wsdl
Local File:
WSDL URL:
IDE Registered:

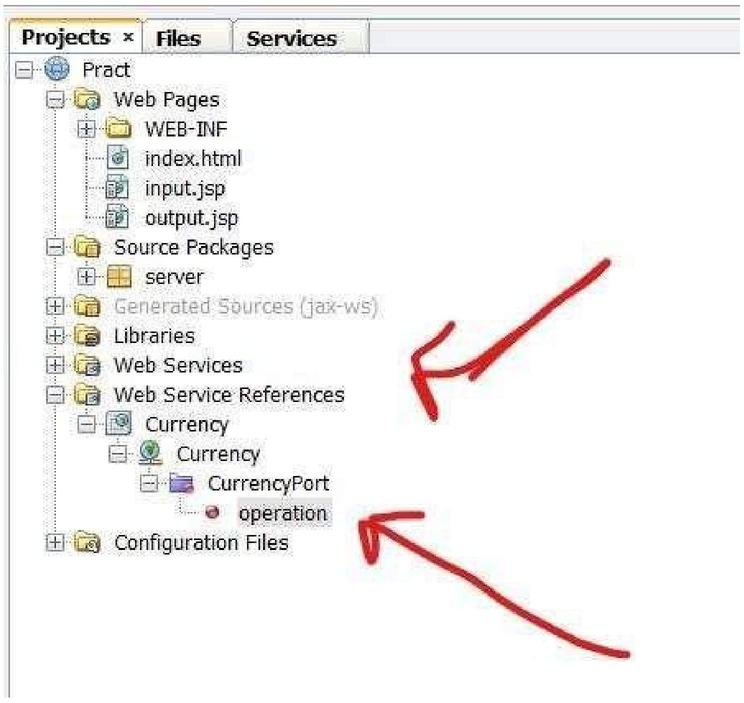
Specify a package name where the client java artifacts will be generated:

Project: Pract
Package: Client

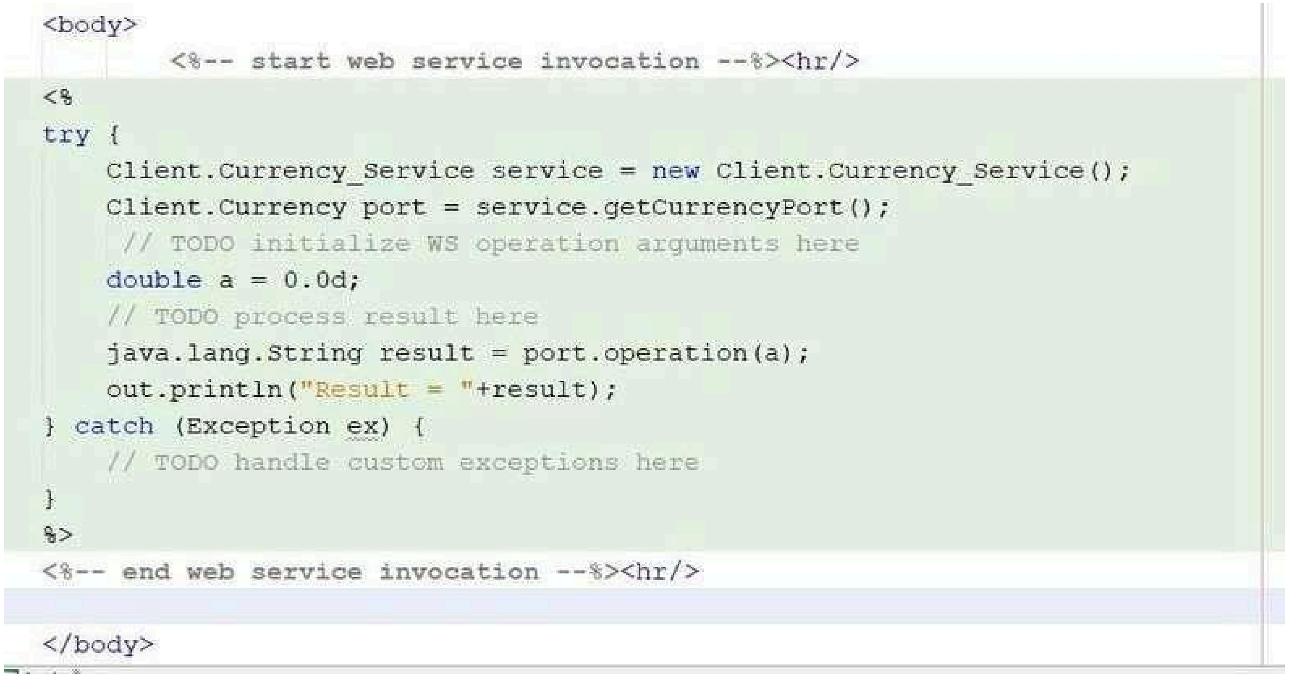
Generate Dispatch code

At this stage copy the url and paste it. Give the package name as "Client". Now click on finish.

19. Now in the project section you can see a new folder has been created named "web service reference". Double click to expand it you should get the following :

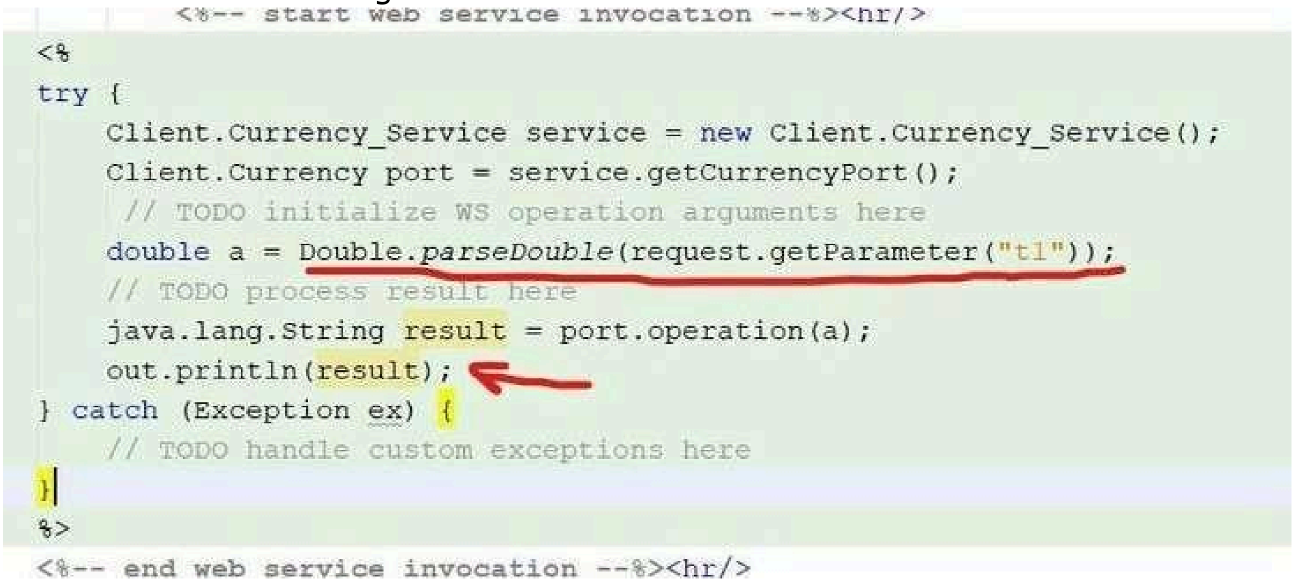


Hold the operation or your operation name and drag it into the output.jsp in body tag as shown : You will get the following auto generated code:



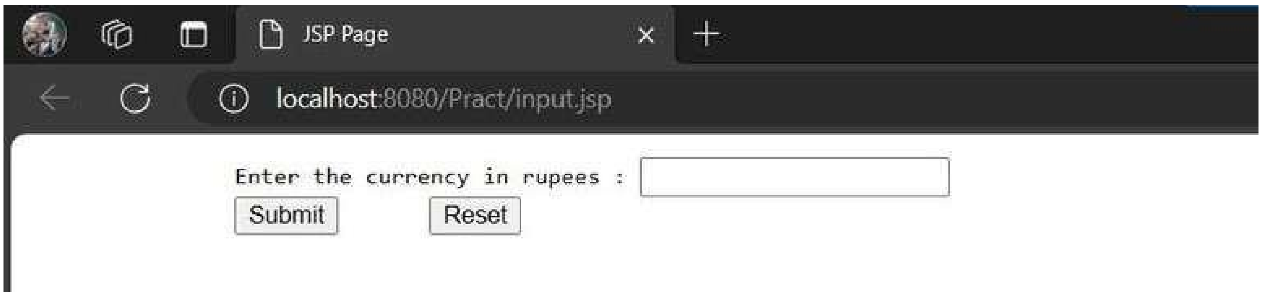
```
<body>
        <%-- start web service invocation --%><hr/>
<%
try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
     // TODO initialize WS operation arguments here
    double a = 0.0d;
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println("Result = "+result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>

</body>
```

20. Now make changes as shown below in the code:



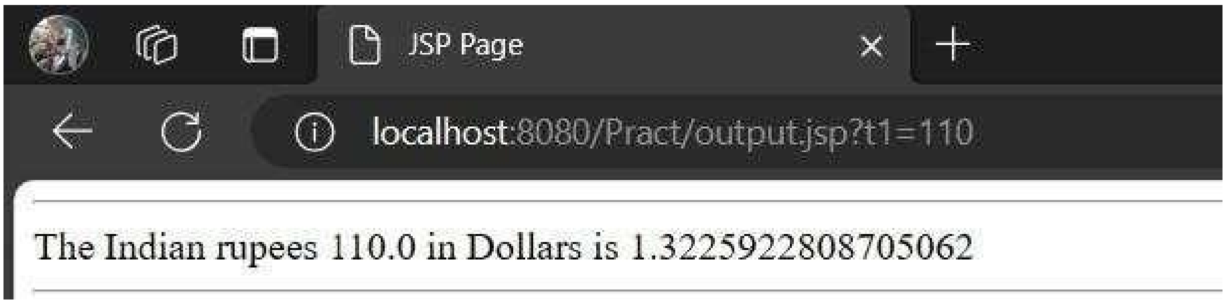
```
        <%-- start web service invocation --%><hr/>
<%
try {
    Client.Currency_Service service = new Client.Currency_Service();
    Client.Currency port = service.getCurrencyPort();
     // TODO initialize WS operation arguments here
    double a = Double.parseDouble(request.getParameter("t1"));
    // TODO process result here
    java.lang.String result = port.operation(a);
    out.println(result);
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>
```

21. Now Deploy our project again and right click on input.jsp and click run file you will redirect to a browser page as shown :



Enter the currency in rupees :

22. Enter any numerical value in text box and click on submit you will get the following output:



```
The Indian rupees 110.0 in Dollars is 1.3225922808705062
```