

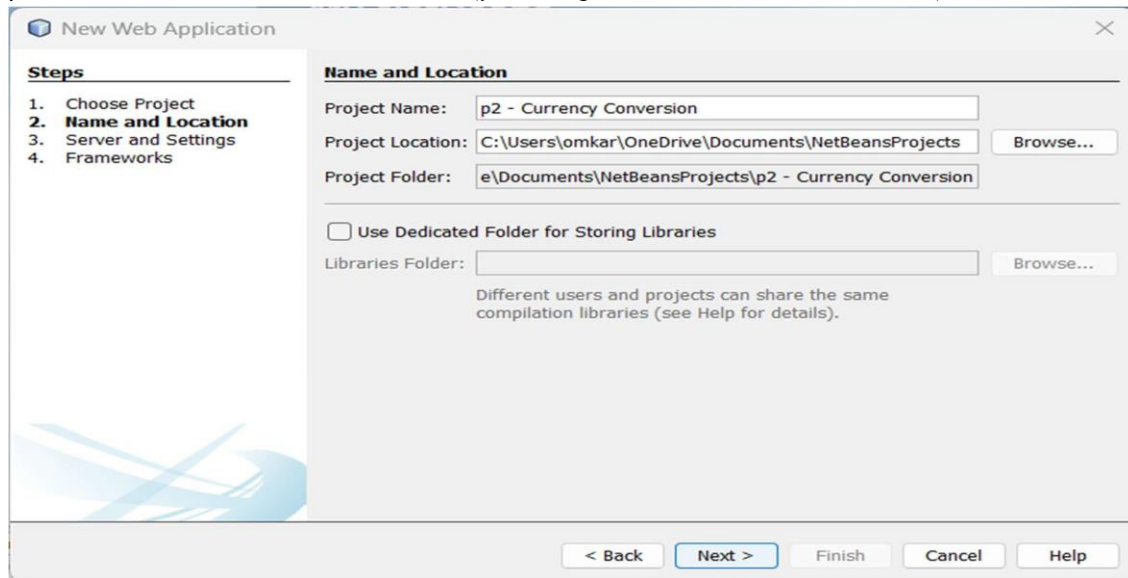
Roll no: 18

Practical 2 - Create a Simple SOAP service:

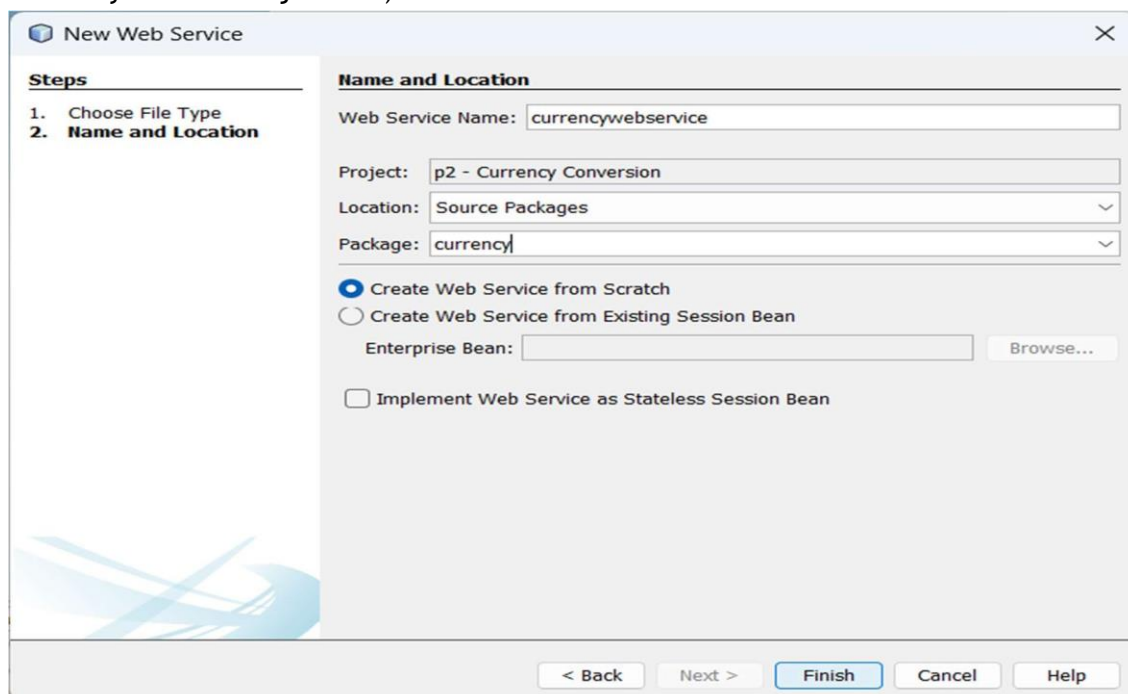
Aim: Define a simple web service like converting Rs into Dollar and call it from different platform like JAVA and .NET.

Steps:

Step 1: Create a Web Application: File > New Project > Java Web > Web Application > Name: p2 > Add GlassFish Server 4.1 > Finish > (you will get index.html, delete this file).



Step 2: Create a Web Service in that project: Right Click on Project Name > New > Web Service... > Name: currencywebservice and Package Name: currency > OK > (you will get currencywebservice.java file).



Step 3: As we have created our WEB SERVICE (i.e. currencywebservice), WE'LL NOW ADD SOME FUNCTIONALITIES (i.e. temperature conversion methods). So, in the class currencywebservice, add 2 methods by doing:

a) Right Click > Insert Code > Add Web Service Operation...>

Name: INRtoDOLL >

Return Type: string >

Add a Parameter 'INR' with Type: double > OK >

Now add formula in return statement: (INR + " rupees in dollar are " + (INR/83.11)) .

b) Right Click > Insert Code > Add Web Service Operation...>

Name: DOLLtoINR >

Return Type: string >

Add a Parameter 'DOLL' with Type: double > OK >

Now add formula in return statement: (DOLL + " dollars in rupees are " + (DOLL*83.11)).

The screenshot shows the 'Add Operation' dialog box with the following details:

- Name:** INRtoDOLL
- Return Type:** java.lang.String
- Parameters:** A table with one parameter named 'INR' of type 'double'.

Name	Type	Final
INR	double	<input type="checkbox"/>

Buttons on the right: Add, Remove, Up, Down. Buttons at the bottom: OK, Cancel.

The screenshot shows the 'Add Operation' dialog box with the following details:

- Name:** DOLLtoINR
- Return Type:** java.lang.String
- Parameters:** A table with one parameter named 'DOLL' of type 'double'.

Name	Type	Final
DOLL	double	<input type="checkbox"/>

Buttons on the right: Add, Remove, Up, Down. Buttons at the bottom: OK, Cancel.

Step 4: Compile(F9) and Deploy (right click on Project Name > Deploy).

Step 5: Now we will test our web service. So, navigate to: Projects > Project Name > Web Services > right click on currencywebservice > click Test Web Service > (the browser will open). (currencywebservice here is a WSDL file).

← ↻ 🏠 ⓘ localhost:8080/p2_-_Currency_Conversion/currencywebservice?Tester

currencywebservice 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 currency.Currencywebservice.dollToINR(double)
dollToINR (10)

public abstract java.lang.String currency.Currencywebservice.inRtoDOLL(double)
inRtoDOLL (200)

dollToINR Method invocation

Method parameter(s)

Type	Value
double	10

Method returned

java.lang.String : **"10.0 dollars in rupees is 831.1"**

Step 6: Now we will be using the created Web Service (i.e. currencywebservice) by creating our WEB SERVICE CLIENT. So, Right Click on Project Name > New > Other > Web Services > Web Service Client... > Browse your Project Name and select WSDL file > Add Package Name: money > Finish.

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: 080/p2_-_Currency_Conversion/currencywebservice?wsdl

☐ Local File:

☐ WSDL URL:

☐ IDE Registered:

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

Project: p2 - Currency Conversion

Package: money

☐ Generate Dispatch code

< Back Next > **Finish** Cancel Help

Step 7: Now to make front page, we need to create a JSP file. So, Right Click on Project Name > New > JSP > File Name: input > Finish. □ Now from the palette, add HTML Form (Action: output.jsp, Method: Post).

Project: p2 - Currency Conversion

New

- Folder...
- Web Service...
- JSP...**
- Web Service Client...
- RESTful Web Services from Entity Classes...
- Servlet...

Start Page x currencywebservice.java x input.jsp x

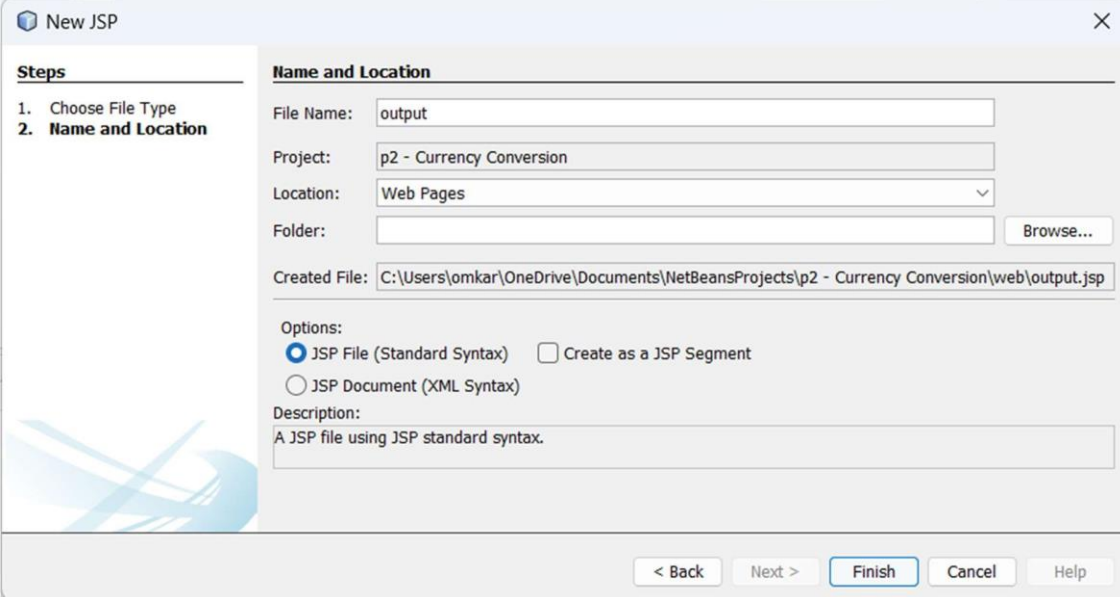
Source History

```

1  <!--
2      Document    : input
3      Created on  : 25 Mar, 2024, 11:53:27 PM
4      Author     : omkar
5  -->
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE html>
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>JSP Page</title>
13     </head>
14     <body>
15         <form action="output.jsp" method="post">
16             <label for="inr">Enter currency in rupees: </label>
17             <input type="text" id="inr" name="i1"><br>
18
19             <label for="doll">Enter currency in dollars: </label>
20             <input type="text" id="doll" name="d1">
21         </form>
22     </body>
23 </html>
24

```

Step 8: Now to create the action behind that, we will create another JSP file, with a name that is declared in the previous JSP file. So, Right Click on Project Name > New > JSP > File Name: output.jsp > Finish. Then add some `<%JSP code%>` under body.



The 'New JSP' dialog box is shown with the 'Name and Location' tab selected. The 'File Name' field contains 'output'. The 'Project' field shows 'p2 - Currency Conversion'. The 'Location' dropdown is set to 'Web Pages'. The 'Folder' field is empty, with a 'Browse...' button next to it. The 'Created File' path is 'C:\Users\omkar\OneDrive\Documents\NetBeansProjects\p2 - Currency Conversion\web\output.jsp'. Under 'Options', 'JSP File (Standard Syntax)' is selected with a radio button, and 'Create as a JSP Segment' is unchecked. The 'Description' field contains 'A JSP file using JSP standard syntax.' At the bottom, there are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

New JSP

Steps

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

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

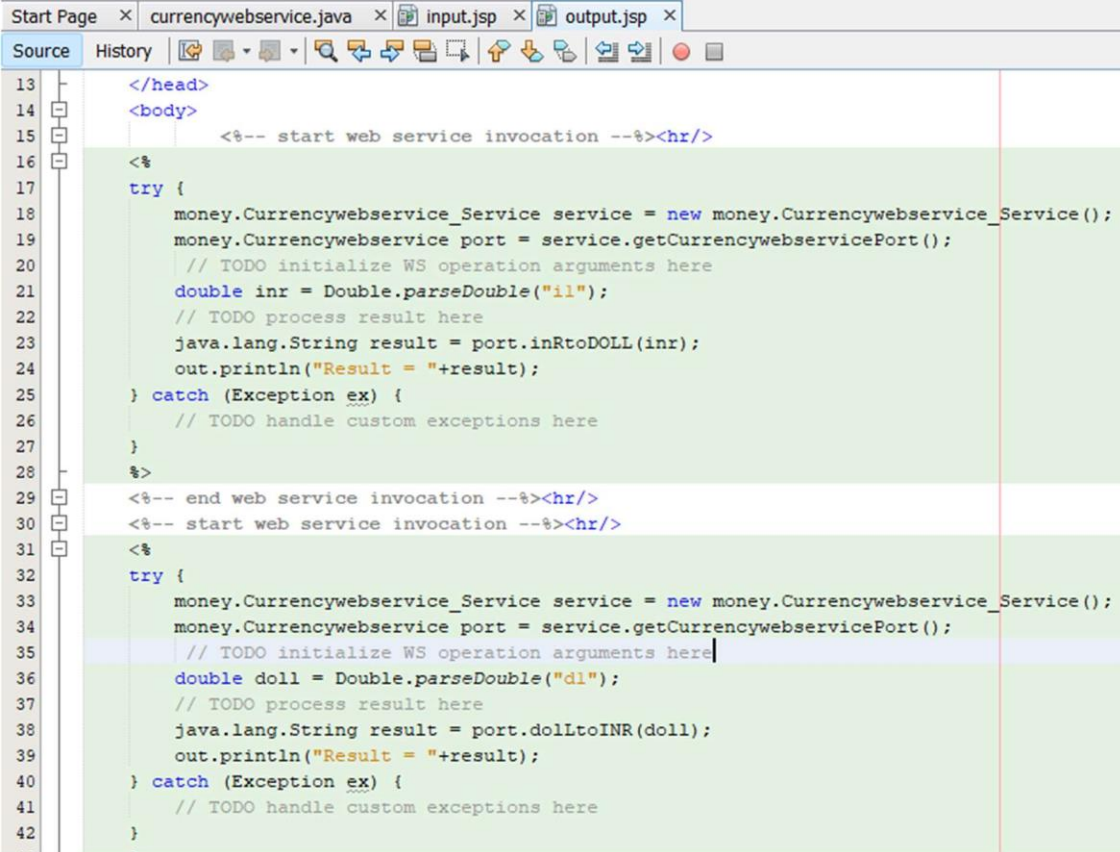
Options:

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

Description:

< Back Next > **Finish** Cancel Help



The NetBeans IDE interface shows three open files: 'currencywebservice.java', 'input.jsp', and 'output.jsp'. The 'output.jsp' file is active, displaying JSP code. The code includes a header, a body, and two JSP snippets. The first snippet (lines 17-28) calls 'inRtoDOLL' and the second (lines 32-42) calls 'dollarToINR'. Both snippets use 'money.Currencywebservice_Service' and 'money.Currencywebservice' objects. The code is highlighted with a green background.

```
13 </head>
14 <body>
15 <!-- start web service invocation --><hr/>
16 <%
17 try {
18     money.Currencywebservice_Service service = new money.Currencywebservice_Service();
19     money.Currencywebservice port = service.getCurrencywebservicePort();
20     // TODO initialize WS operation arguments here
21     double inr = Double.parseDouble("11");
22     // TODO process result here
23     java.lang.String result = port.inRtoDOLL(inr);
24     out.println("Result = "+result);
25 } catch (Exception ex) {
26     // TODO handle custom exceptions here
27 }
28 %>
29 <!-- end web service invocation --><hr/>
30 <!-- start web service invocation --><hr/>
31 <%
32 try {
33     money.Currencywebservice_Service service = new money.Currencywebservice_Service();
34     money.Currencywebservice port = service.getCurrencywebservicePort();
35     // TODO initialize WS operation arguments here
36     double doll = Double.parseDouble("d1");
37     // TODO process result here
38     java.lang.String result = port.dollarToINR(doll);
39     out.println("Result = "+result);
40 } catch (Exception ex) {
41     // TODO handle custom exceptions here
42 }
43 %>
```

Step 9: Now RUN “index.jsp” file.

Step 10: Finish!

Outputs:

The image shows two screenshots of a web browser. The top screenshot shows the 'input.jsp' page where the user has entered '110' for rupees and '76' for dollars. The bottom screenshot shows the 'output.jsp' page displaying the conversion results.

localhost:8080/p2_-_Currency_Conversion/input.jsp

Enter currency in rupees:

Enter currency in dollars:

localhost:8080/p2_-_Currency_Conversion/output.jsp

Result = 110.0 rupees in dollar is 1.323547106244736

Result = 76.0 dollars in rupees is 6316.36
