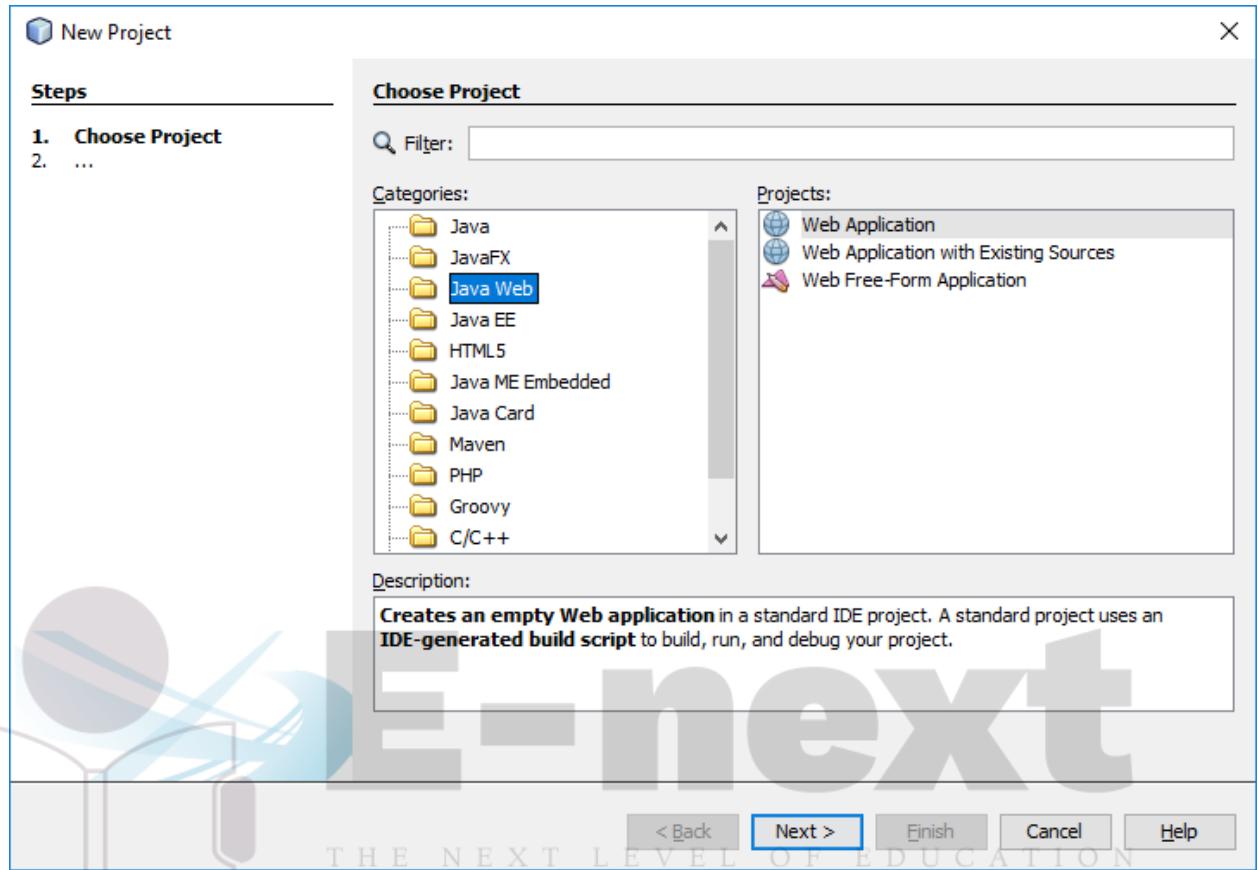
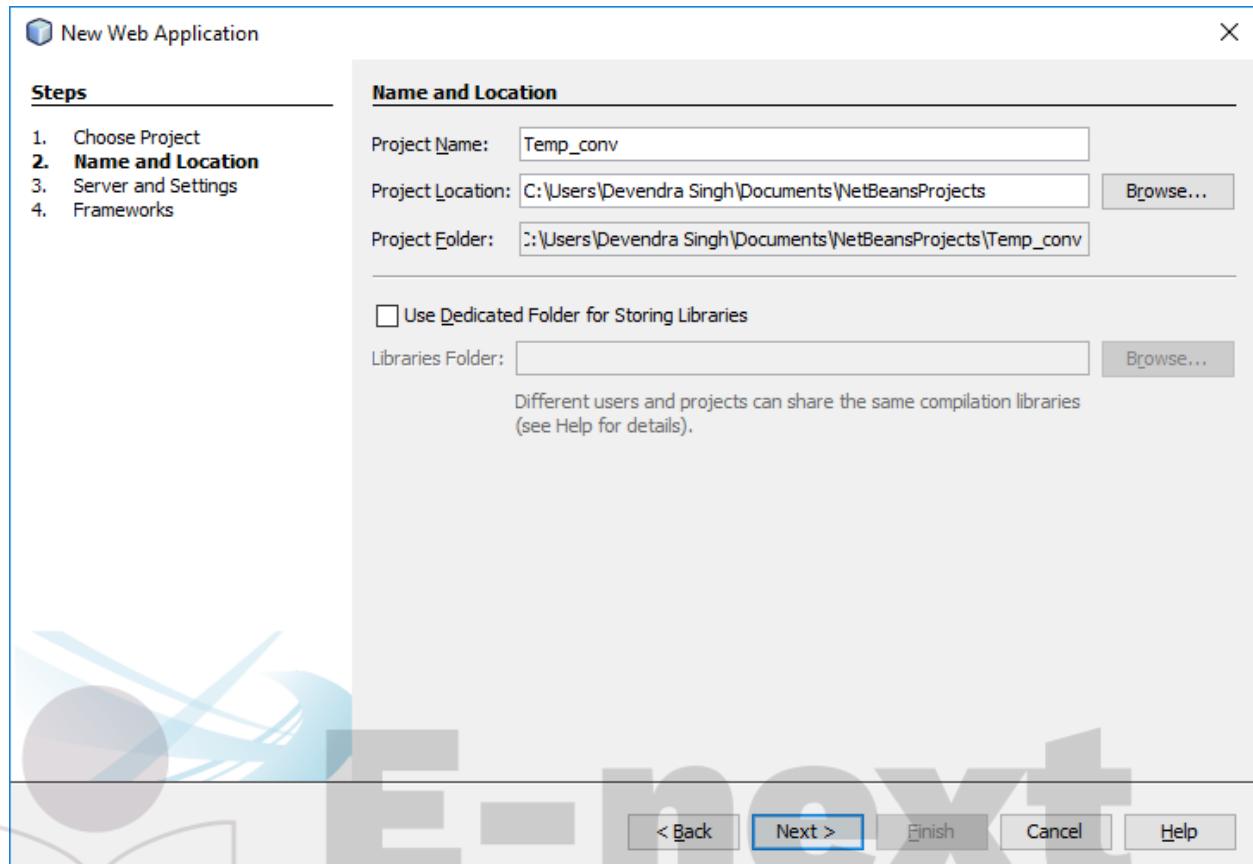


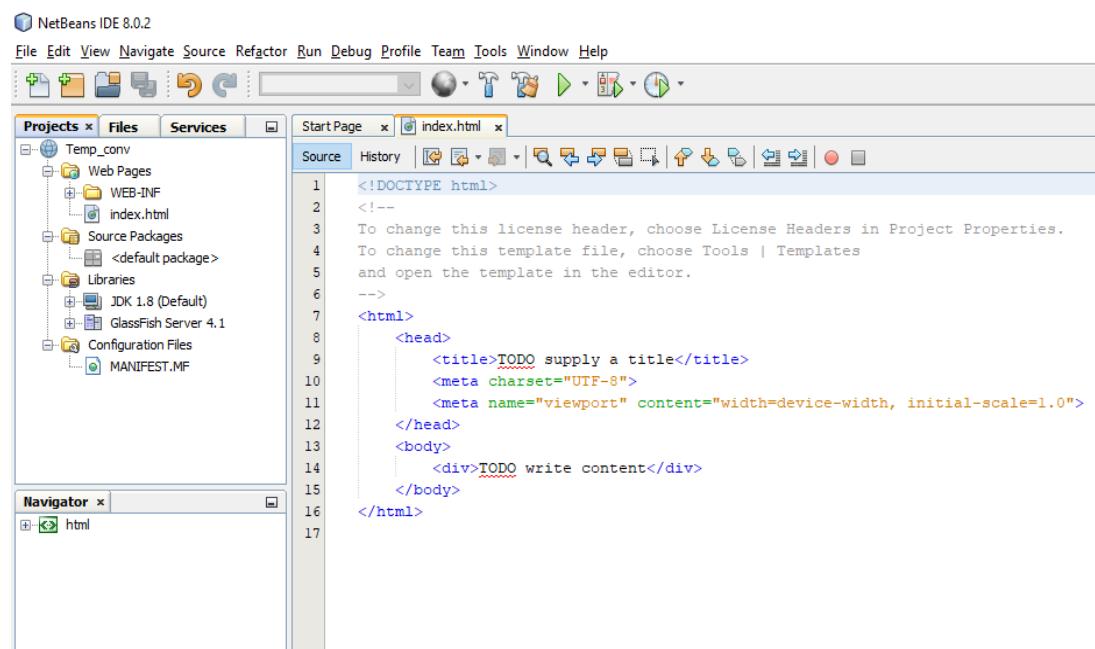
1. Go to ~~File~~ -> New Project. Select **Java Web** in categories and **Web Application** in Projects. Click on **Next** to create web based project.



2. Enter a project name whatever you want and then click on **Next**. On next page click **Finish**.

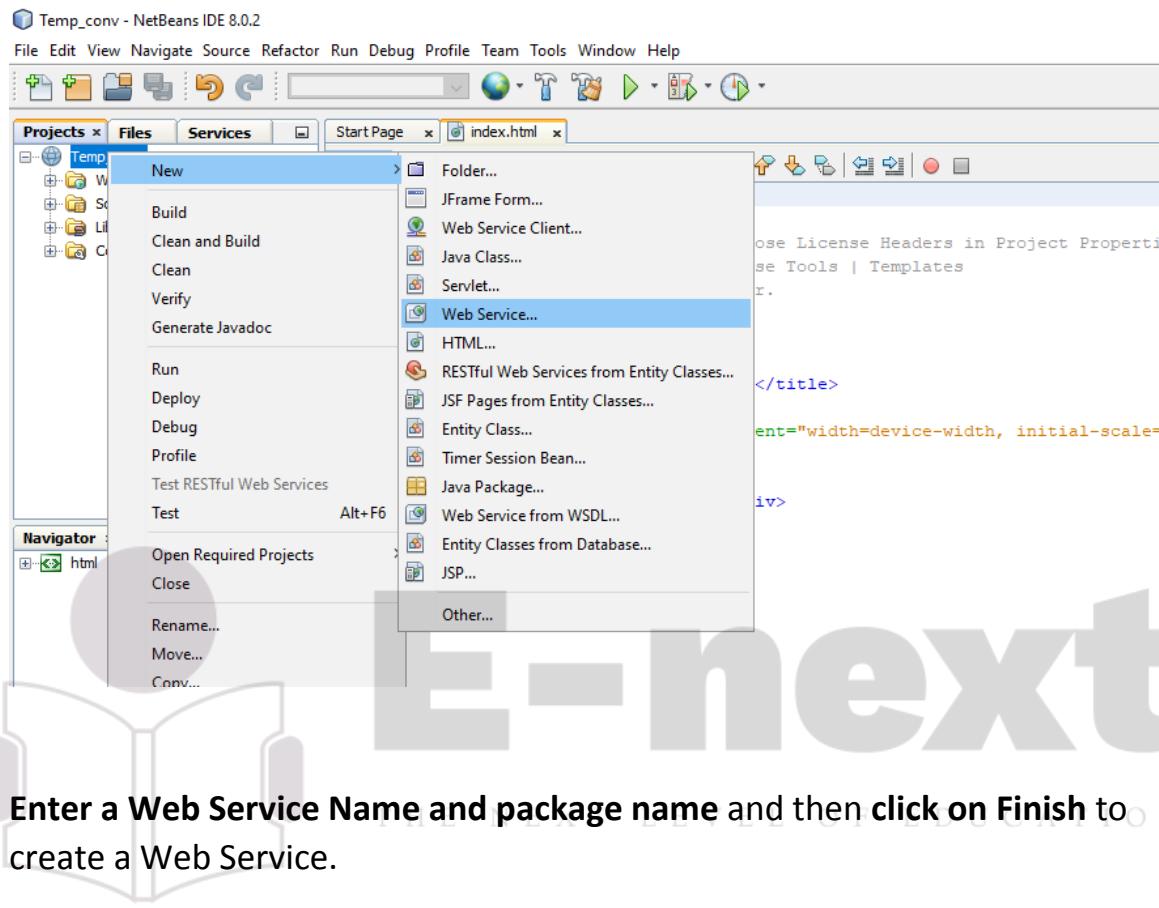


3. After completion of project creation process a window will appear like below.

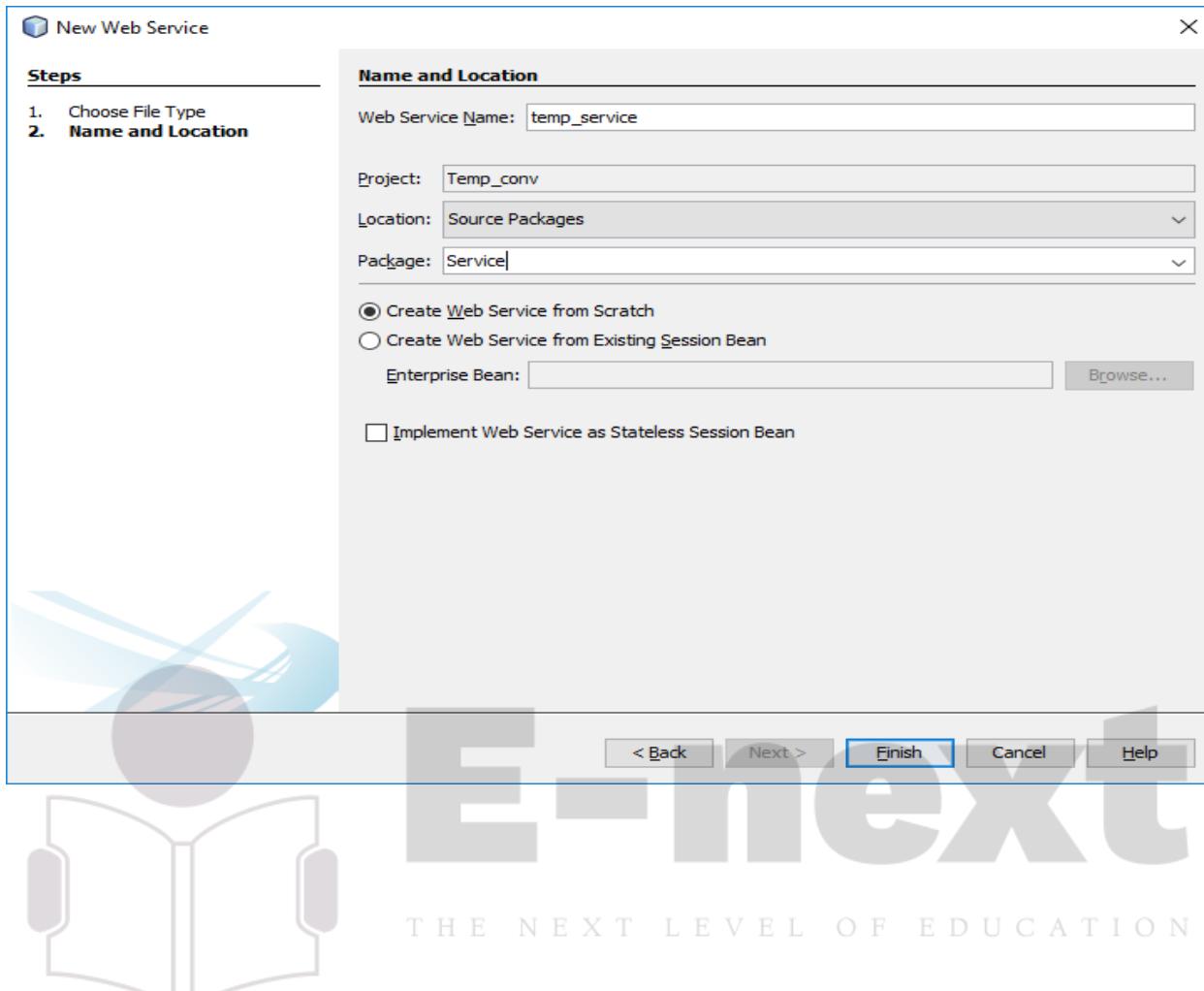


4. Create web service.

Right click on Project -> New -> Web Service

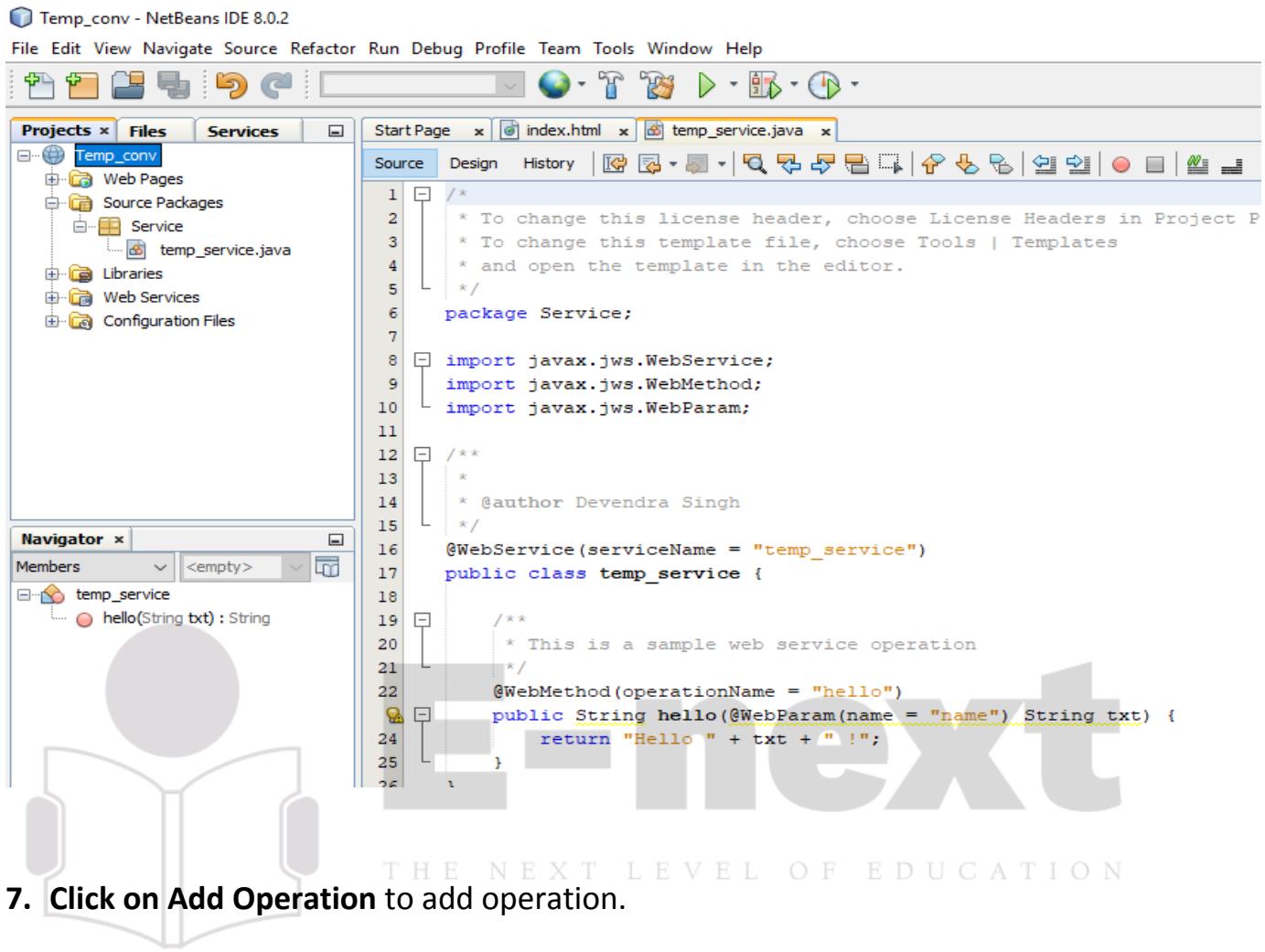


- 5. Enter a Web Service Name and package name and then click on Finish to create a Web Service.**

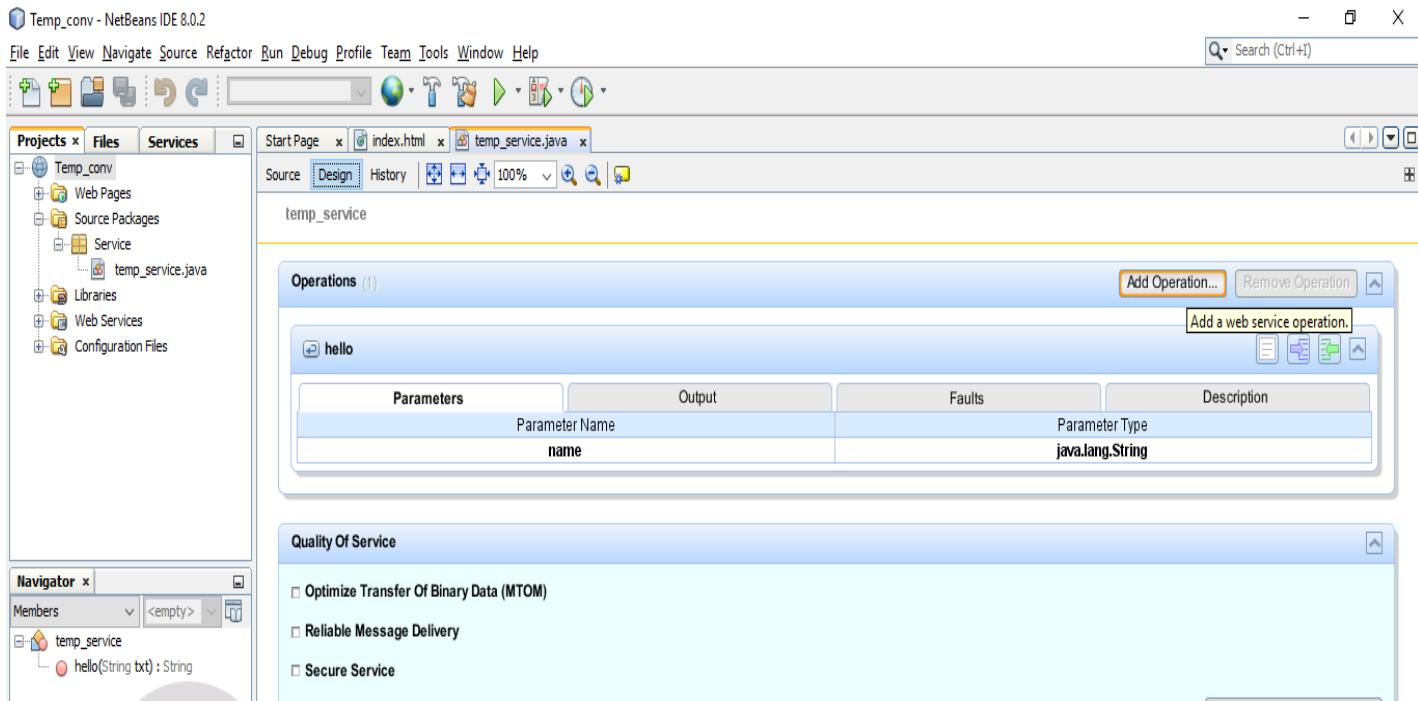


6. As you can see in following pic; In **Source Packages** there is a package **Service** which contains the service file **temp_service.java**. Open this file by double click on it, So that we can add two operation that will convert temperature from celcious to farhenheit and vice-versa.

Go to design mode by click on **Design** .

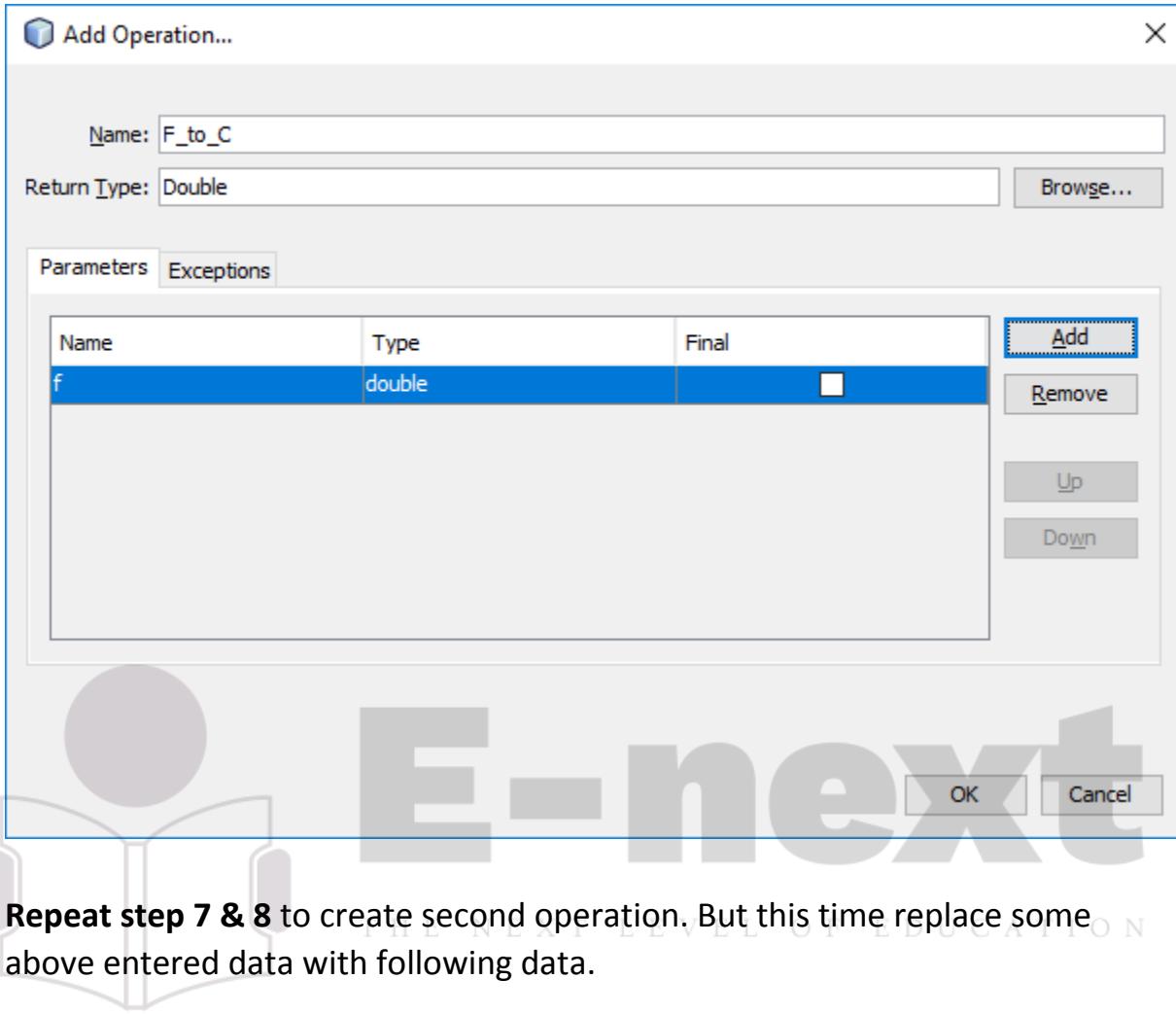


7. Click on Add Operation to add operation.



8. Give Operation name **F_to_C** and return type as **Double**. So this method will return value in Double data type. After that click on **Add** button to give parameters for method. Give its name as **f** and type as **Double** and then click on **OK** button. Your one operation is now successfully created.

THE NEXT LEVEL OF EDUCATION

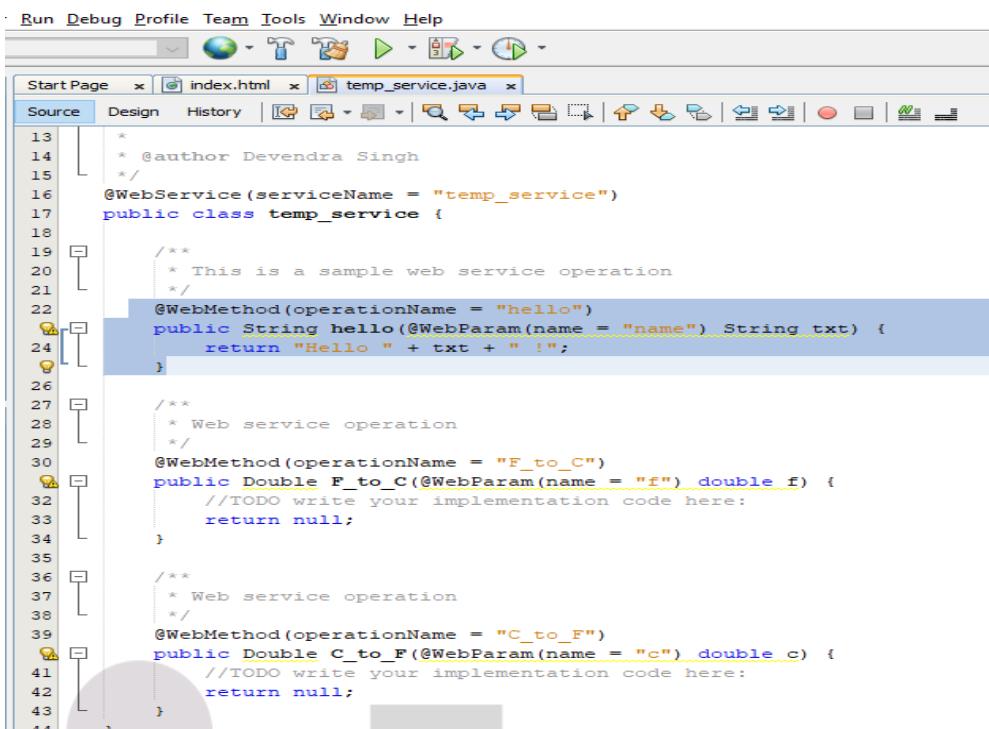


9. Repeat step 7 & 8 to create second operation. But this time replace some above entered data with following data.

F_to_C -> C_to_F

f -> c

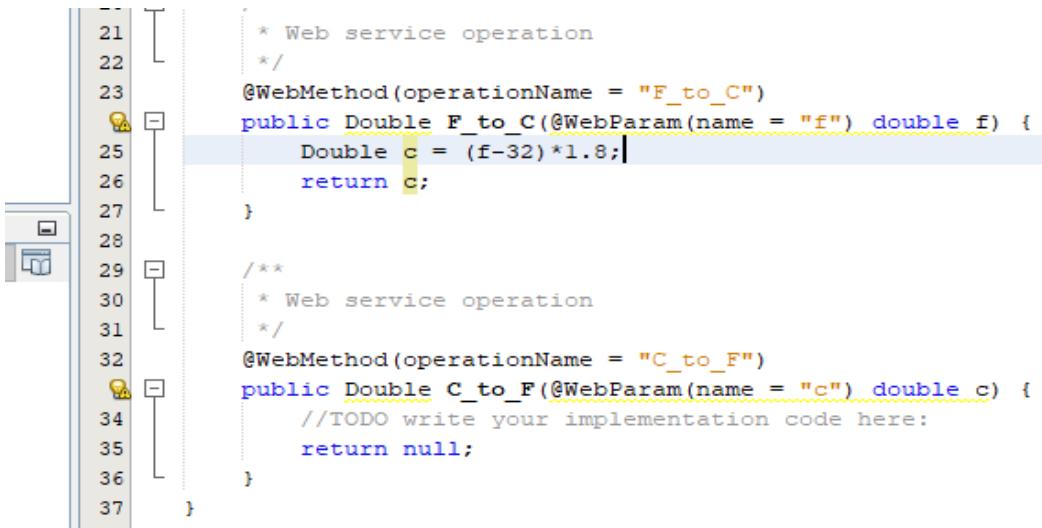
10. Now go to source mode by click on Source and **delete the selected** segment of code. Because it is default operation and we don't need this.



```
13
14     * @author Devendra Singh
15     */
16     @WebService(serviceName = "temp_service")
17     public class temp_service {
18
19         /**
20          * This is a sample web service operation
21          */
22         @WebMethod(operationName = "hello")
23         public String hello(@WebParam(name = "name") String txt) {
24             return "Hello " + txt + " !";
25         }
26
27         /**
28          * Web service operation
29          */
30         @WebMethod(operationName = "F_to_C")
31         public Double F_to_C(@WebParam(name = "f") double f) {
32             //TODO write your implementation code here:
33             return null;
34         }
35
36         /**
37          * Web service operation
38          */
39         @WebMethod(operationName = "C_to_F")
40         public Double C_to_F(@WebParam(name = "c") double c) {
41             //TODO write your implementation code here:
42             return null;
43         }
44     }
```

11. Now replace the selected area with following code to convert Fahrenheit to Celsius.

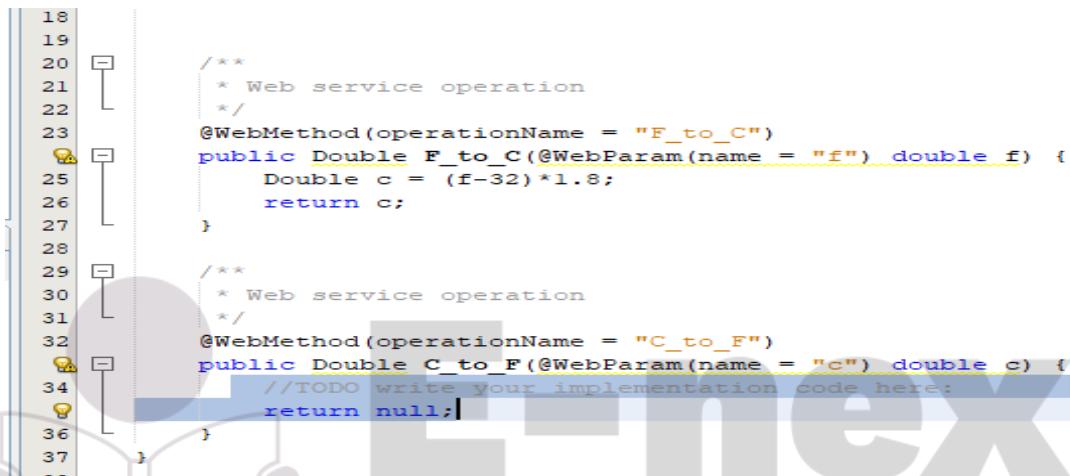
Double c = (f-32)*1.8;
return c;



```
21         /**
22          * Web service operation
23          */
24         @WebMethod(operationName = "F_to_C")
25         public Double F_to_C(@WebParam(name = "f") double f) {
26             Double c = (f-32)*1.8;
27             return c;
28         }
29
30         /**
31          * Web service operation
32          */
33         @WebMethod(operationName = "C_to_F")
34         public Double C_to_F(@WebParam(name = "c") double c) {
35             //TODO write your implementation code here:
36             return null;
37         }
38     }
```

- 12.** Now replace the selected area with following code to convert Celsius to Fahrenheit and then press Ctrl+S to save.

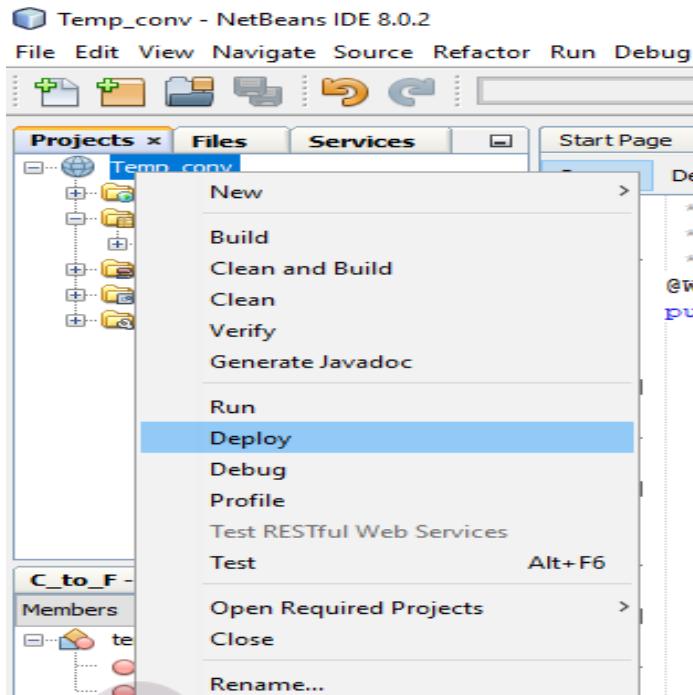
```
Double f = (c*1.8)+32;  
return f;
```



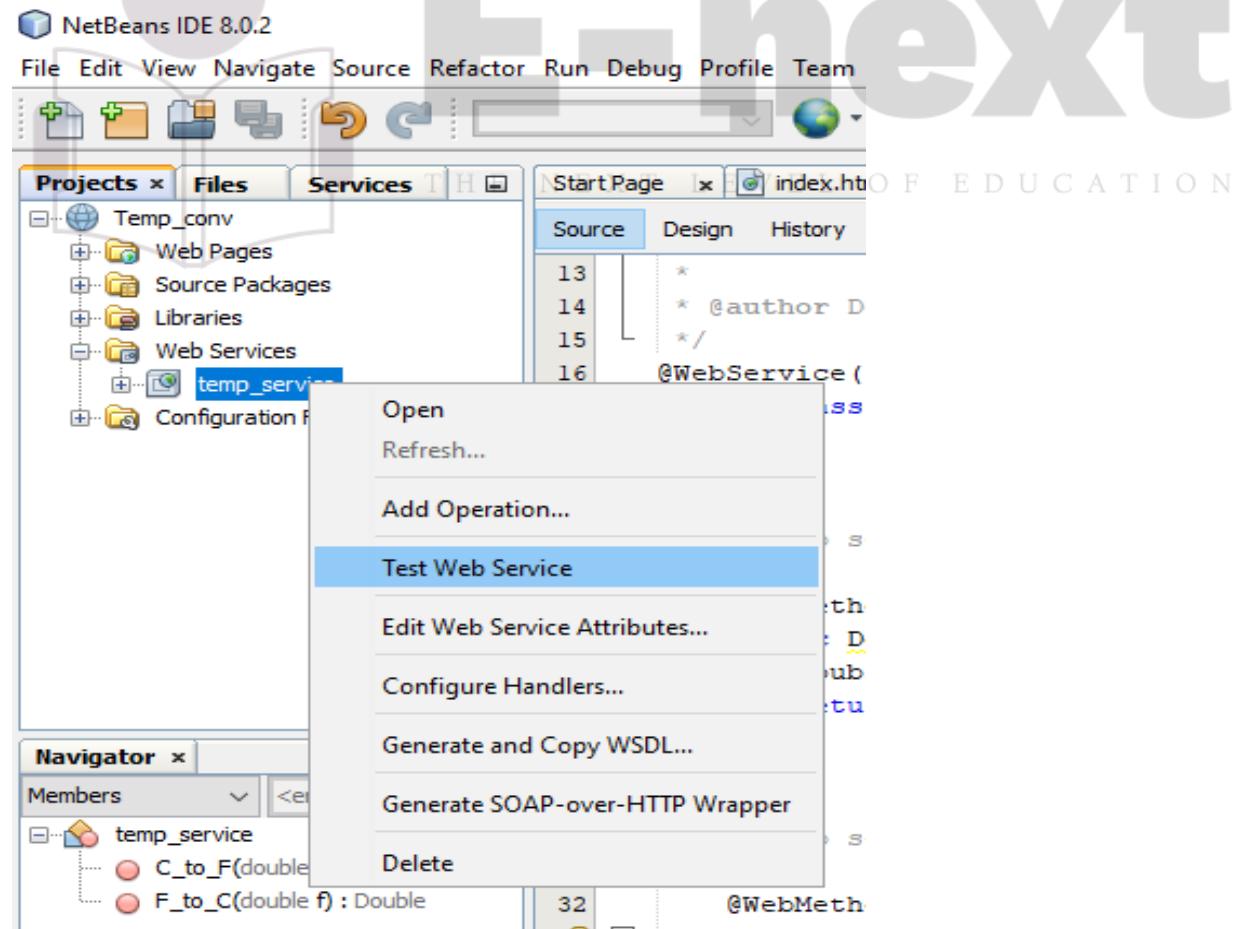
A screenshot of a Java code editor showing a portion of a service class. The code includes two methods: `F_to_C` and `C_to_F`. The `F_to_C` method converts Celsius to Fahrenheit using the formula $f = (c * 1.8) + 32$. The `C_to_F` method is annotated with `@WebMethod(operationName = "C_to_F")` and contains a comment `//TODO write your implementation code here:`. A blue selection bar highlights the line `return null;!` in the `C_to_F` method, indicating it is the part of the code that needs to be replaced.

```
18  
19  
20  
21    /**  
22    * Web service operation  
23    */  
24    @WebMethod(operationName = "F_to_C")  
25    public Double F_to_C(@WebParam(name = "f") double f) {  
26       Double c = (f-32)*1.8;  
27       return c;  
28   }  
29  
30   /**  
31   * Web service operation  
32   */  
33   @WebMethod(operationName = "C_to_F")  
34   public Double C_to_F(@WebParam(name = "c") double c) {  
35       //TODO write your implementation code here:  
36       return null;!  
37   }  
38 }
```

- 13.** Now right click on project name and click on Deploy to deploy your project.



14. To test your web service follow the following process as in picture.



15. Following window will open in in browser. Now if you will enter a numeric data into first box and you will click on first button it will convert the entered data into Celsius.



temp_service 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.Double service.TempService.fToC(double)
 ()

public abstract java.lang.Double service.TempService.cToF(double)
 ()

16. Selected value is in celsius of 56.

A screenshot of a web browser window titled "Method invocation trace" from the E-next platform. The URL is http://localhost:8080/Temp_conv/temp_s. The page displays the result of a "fToC Method invocation" with a value of 43.2. The background features the E-next logo and tagline "THE NEXT LEVEL OF EDUCATION".

Method parameter(s)

Type	Value
double	56

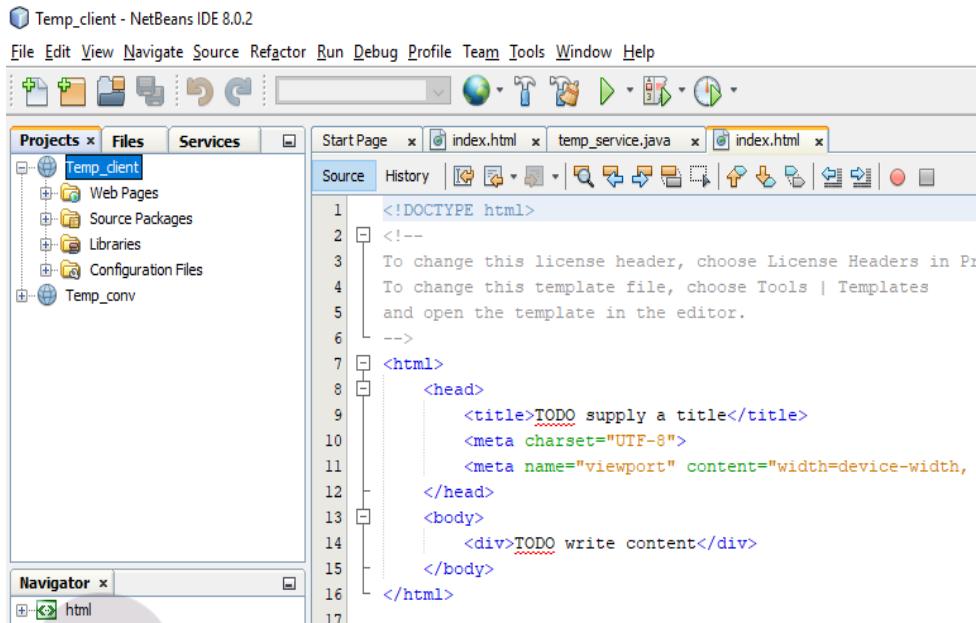
Method returned

java.lang.Double : "43.2"

SOAP Request

17. Similarly second textbox and button will convert the numeric value into Fahrenheit. Now to consume this web service we are creating a client.

18. Now create a new web application project with name as Temp_client.

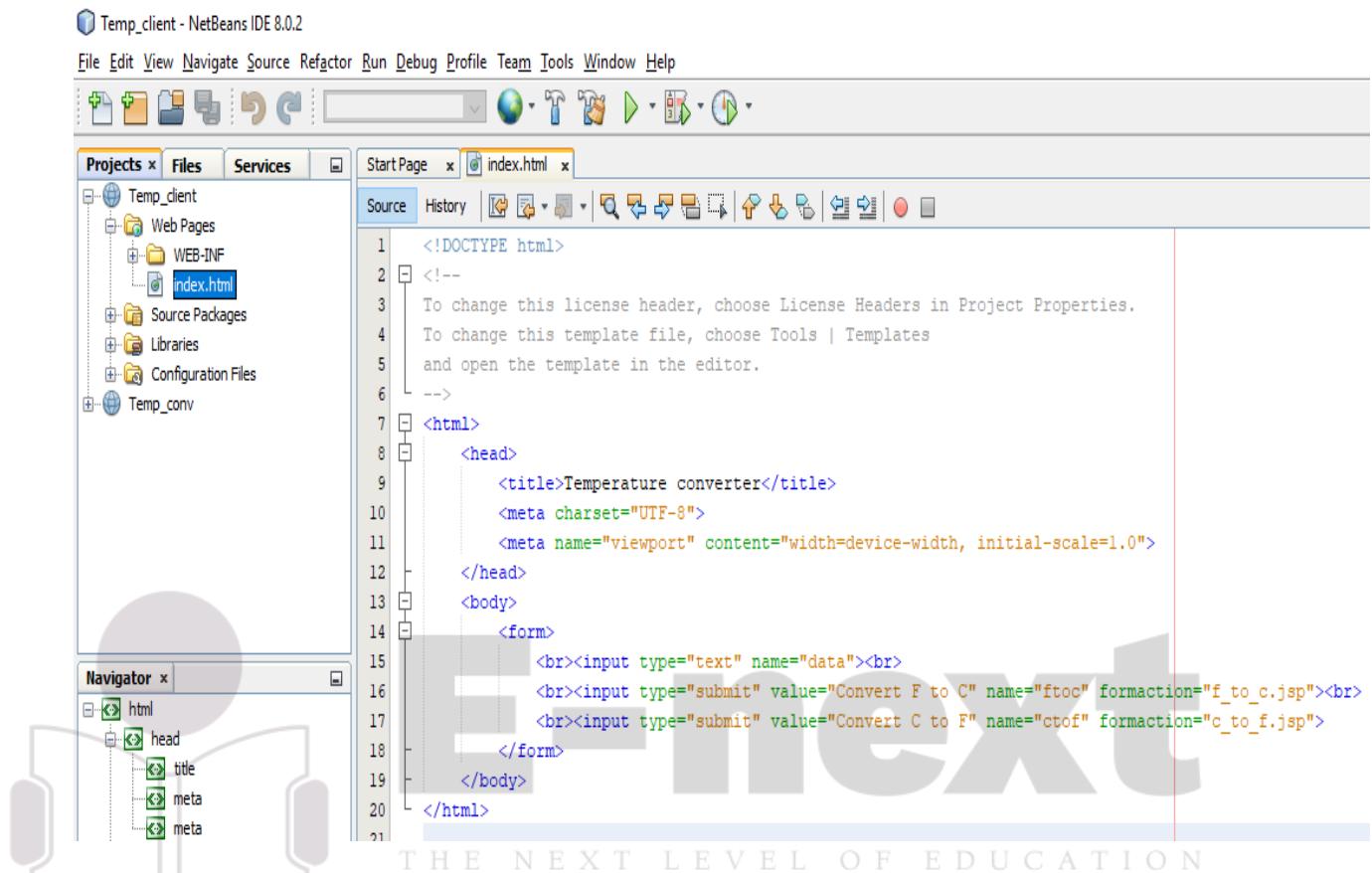


19. Now open the index.html page of Temp_client and write the following code into that.



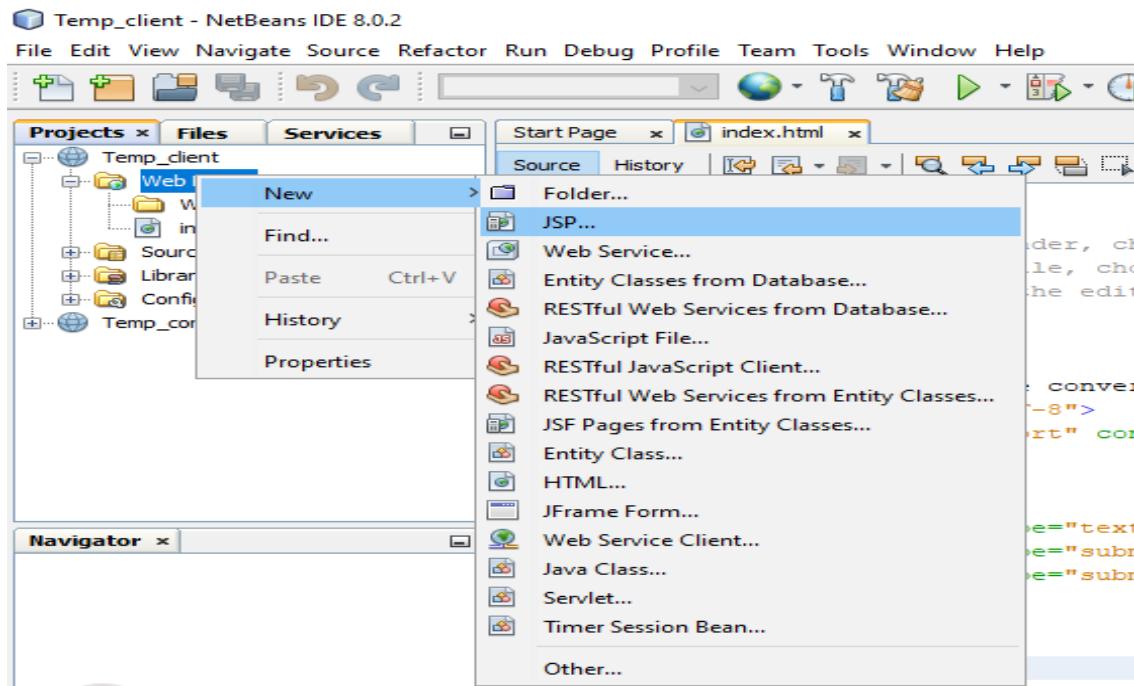
THE NEXT LEVEL OF EDUCATION

```
<html>
  <head>
    <title>Temperature converter</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form>
      <br><input type="text" name="data"><br>
      <br><input type="submit" value="Convert F to C" name="ftoc" formaction="f_to_c.jsp"><br>
      <br><input type="submit" value="Convert C to F" name="ctof" formaction="c_to_f.jsp">
    </form>
  </body>
</html>
```

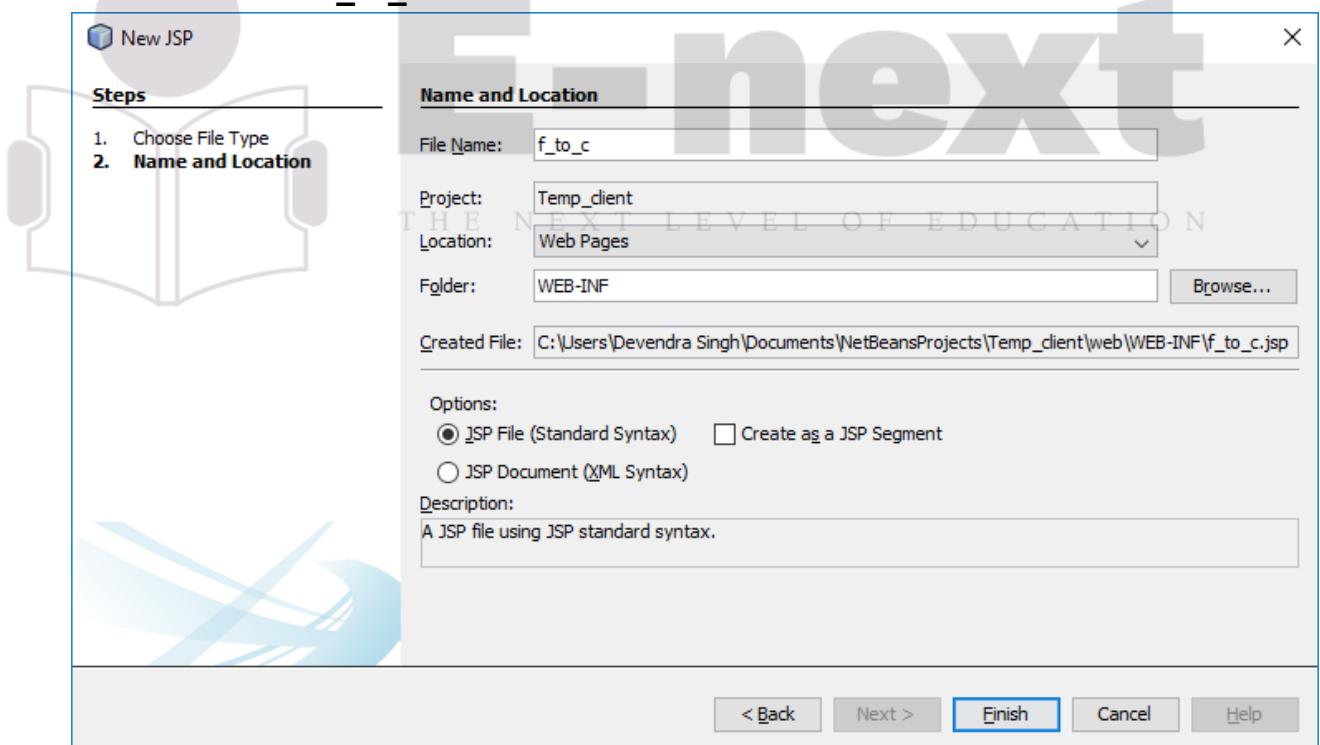


20. Now create two jsp pages for both submit button.

Right click on Web Pages -> New -> JSP

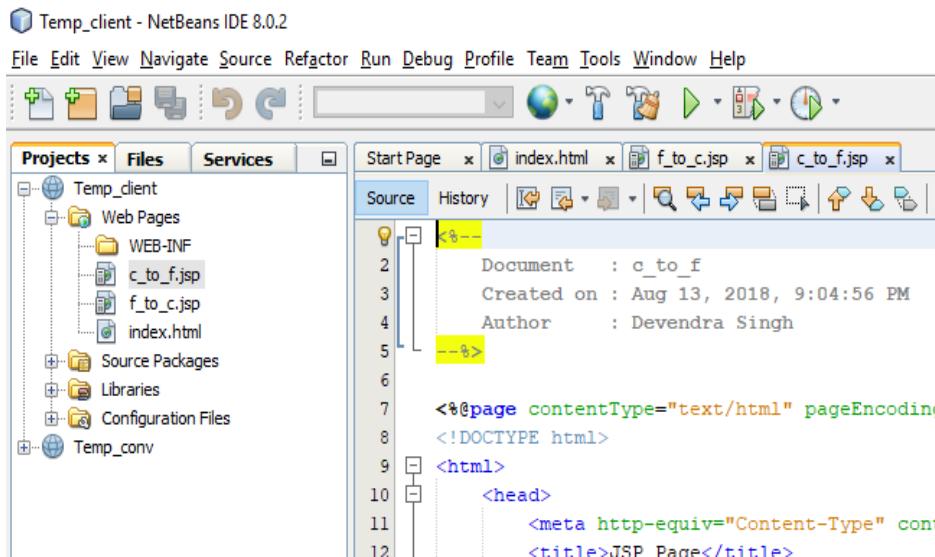


21. Enter file name f_to_c and then click on Finish.



22. Now repeat the step number 20 & 21. But give the File Name as c_to_f.

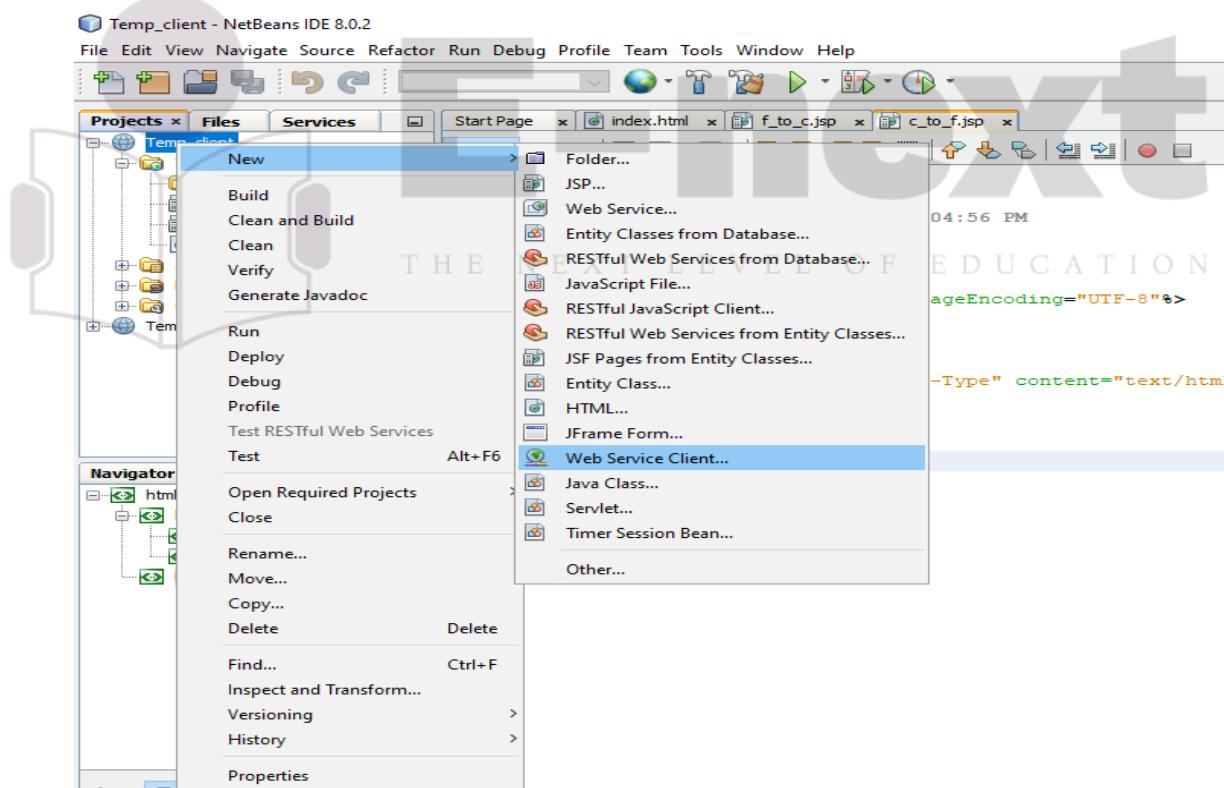
23. Now you have created two jsp files.



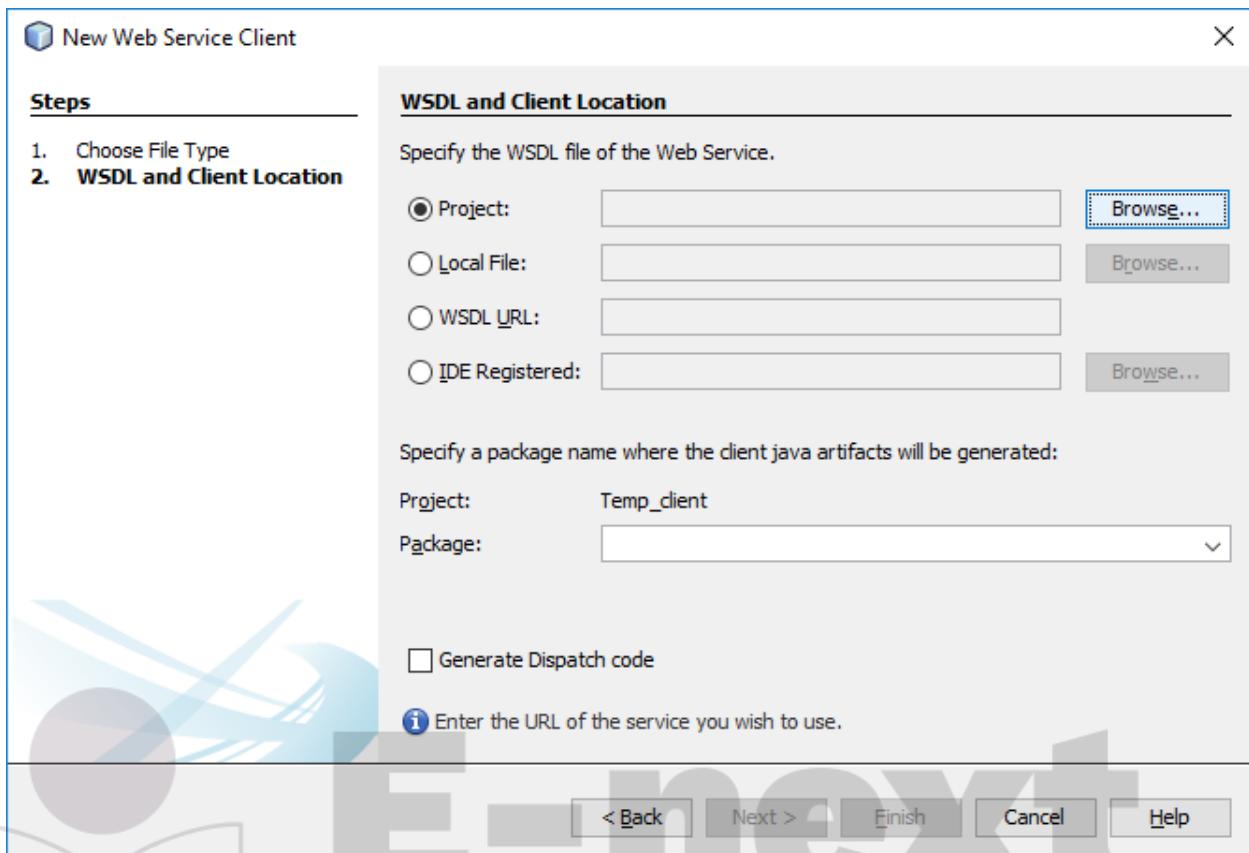
The screenshot shows the NetBeans IDE interface with the title bar "Temp_client - NetBeans IDE 8.0.2". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for file operations like New, Open, Save, and Build. The Projects tab in the left sidebar shows a project named "Temp_client" with Web Pages, WEB-INF, and Configuration Files. The Files tab shows files like index.html, f_to_c.jsp, and c_to_f.jsp. The Services tab is empty. The Source tab in the main editor shows the content of c_to_f.jsp:

```
<%--  
1 Document      : c_to_f  
2 Created on   : Aug 13, 2018, 9:04:56 PM  
3 Author        : Devendra Singh  
4 --%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding=  
8 !DOCTYPE html>  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" cont  
12     <title>JSP Page</title>
```

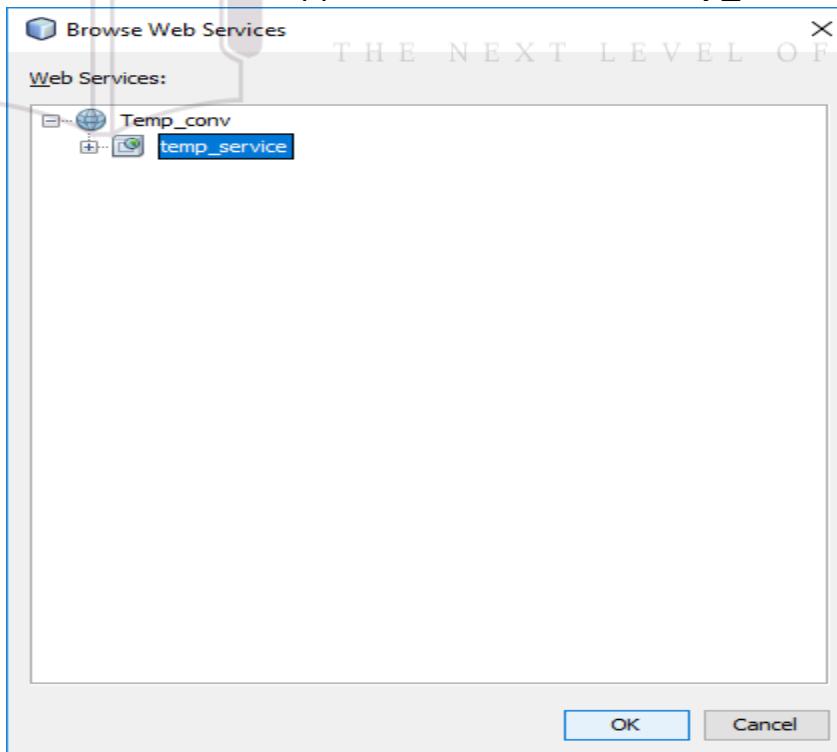
24. Right click on Temp_client and select Web Service Client as below.



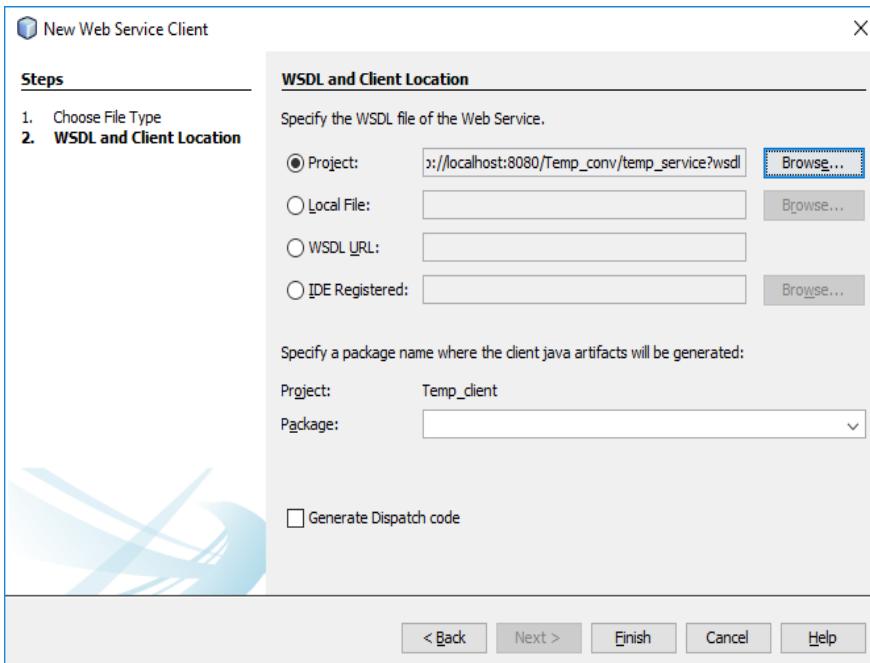
25. Click on Browse.



26. New window will appear and then select **temp_service** and click on **OK**.



27. Click on Finish.



28. Now open the c_to_f.jsp file and delete the selected line.

```
<%--  
1 Document : c_to_f  
2 Created on : Aug 13, 2018, 9:04:56 PM  
3 Author : Devendra Singh XT LEVEL OF EDUCATION  
4--%>  
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 <h1>Hello World!</h1>  
16 </body>  
17 </html>  
18
```

29. Now right click in between the body section and select Call Web Service Operation as below.

The screenshot shows an IDE interface with several tabs at the top: Start Page, index.html, f_to_c.jsp, and c_to_f.jsp. The c_to_f.jsp tab is active. Below the tabs is a toolbar with various icons. The main area displays the source code for c_to_f.jsp:

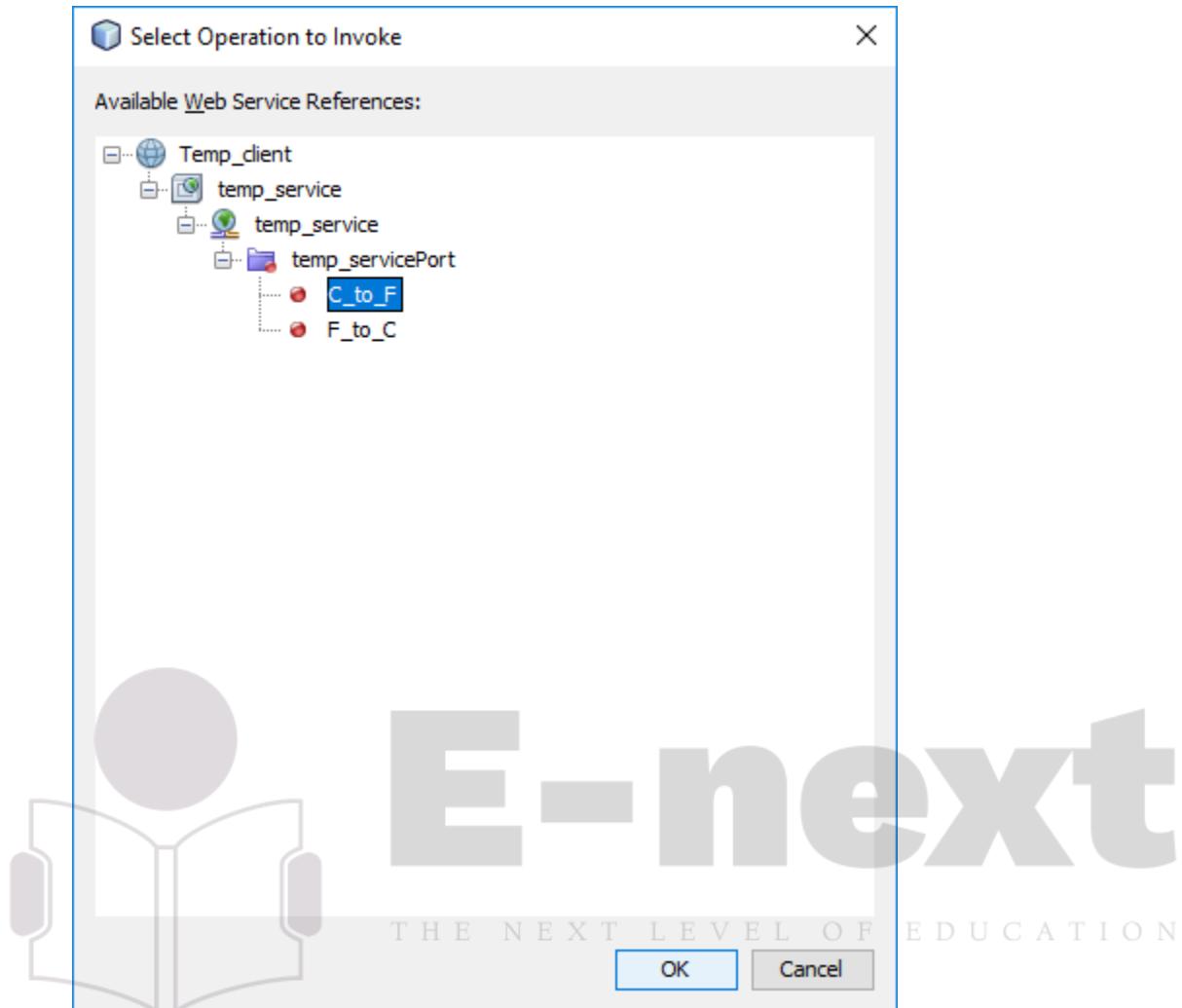
```
<%--  
1 Document      : c_to_f  
2 Created on   : Aug 13, 2018, 9:04:56 PM  
3 Author        : Devendra Singh  
4 --%>  
5  
6 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
7 <!DOCTYPE html>  
8  
9 <html>  
10 <head>  
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12     <title>JSP Page</title>  
13 </head>  
14 <body>  
15  
16     </body>  
17 </html>
```

A context menu is open over the code editor, specifically over the closing tag of the body element. The menu items include:

- Run File (Shift+F6)
- Find Usages (Alt+F7)
- Refactor (disabled)
- Format (Alt+Shift+F)
- New Watch...
- Toggle Line Breakpoint (Ctrl+F8)
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Properties
- Code Folds (disabled)
- Select in Projects
- Web Service Client Resources (disabled)
- Call Web Service Operation... (disabled)

Below the menu, there is a decorative watermark or logo for "E-NEXT THE NEXT LEVEL OF EDUCATION".

30. New window will appear select the C_to_F by expanding it and click on OK.
So that selected code in second pic will be automatically generated.



The screenshot shows the NetBeans IDE interface with the menu bar at the top. Below it is a toolbar with various icons. The main area displays a JSP file named 'c_to_f.jsp'. The code in the editor is:

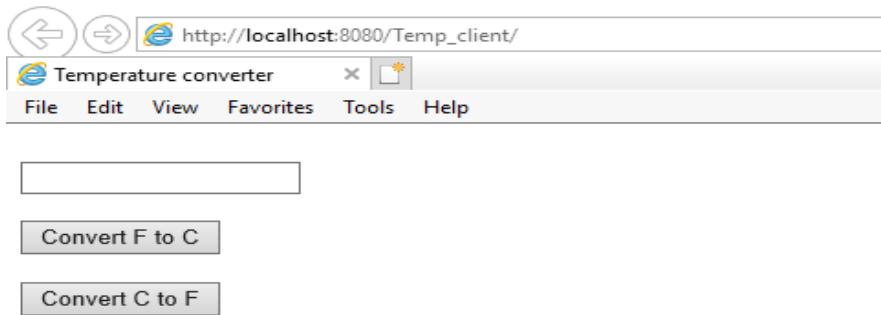
```
13 </head>
14 <body>
15
16 <%-- start web service invocation --%><hr/>
17 <%
18 try {
19     service.TempService_Service service = new service.TempService_Service();
20     service.TempService port = service.getTempServicePort();
21     // TODO initialize WS operation arguments here
22     double c = 0.0d;
23     // TODO process result here
24     java.lang.Double result = port.cToF(c);
25     out.println("Result = "+result);
26 } catch (Exception ex) {
27     // TODO handle custom exceptions here
28 }
29 <%>
30 <%-- end web service invocation --%><hr/>
31 </body>
32 </html>
```

31. Now, make the selected area in step 30 as like selected area in below pic by adding some line of code.

This screenshot shows the same NetBeans IDE environment as the previous one, but with a green rectangular selection highlighting the entire code block from line 16 to line 34. This visual cue indicates the specific area of code that needs to be modified according to the instructions in step 31.

32. Now Open the **f_to_c.jsp** file and follow the steps from 28 to 31. Only the change is in 30 number step and i.e. instead of **C_to_F**, you have to select **F_to_C**.

33. Now run the **Temp_client** project. An window will be open like below.



E-next
THE NEXT LEVEL OF EDUCATION

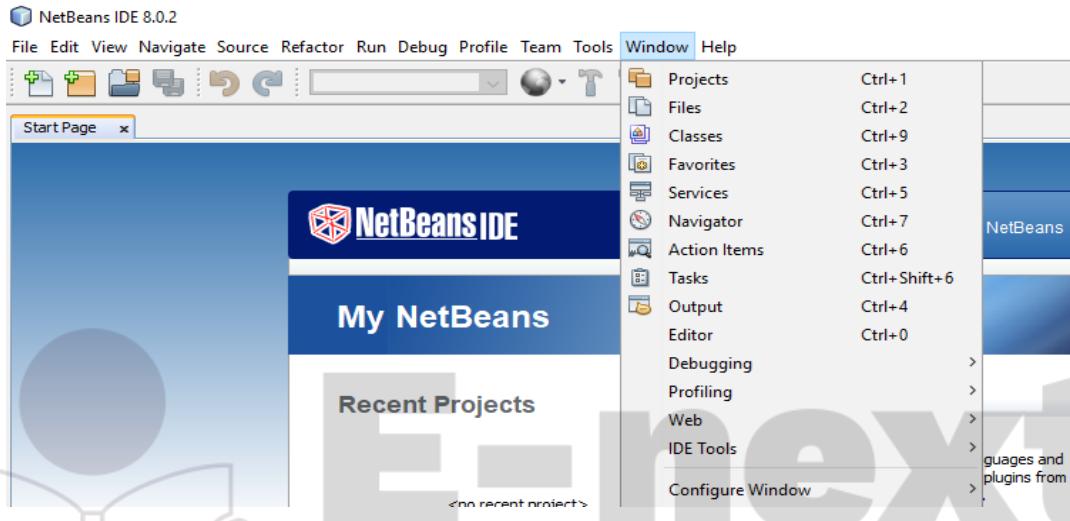
34. Now you can enter any numeric data into textbox and if you will click the first button it will convert the numeric value into Celsius and vice-versa for the second button.

35. There are so many methods to consume the web service. But I found it easy.

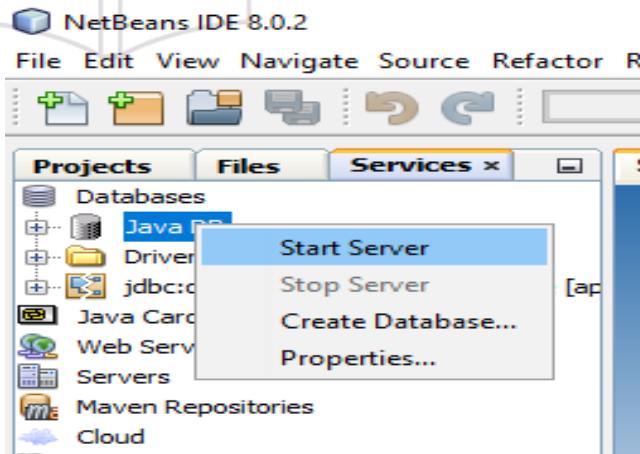
PRACTICAL-2

AIM: Write a program to implement the operation can receive request and will return a response in two ways. a) One - Way operation b) Request –Response.

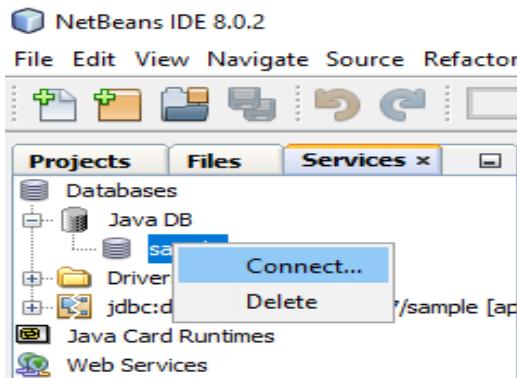
1. Click on Window menu and click on **Projects, Files & Services** to open it.



2. Right click on Java DB and then click on Start Server to start the server.

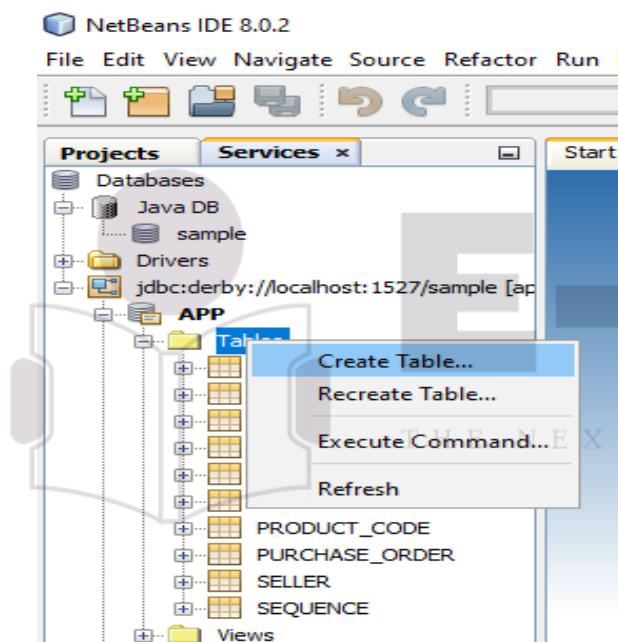


3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.

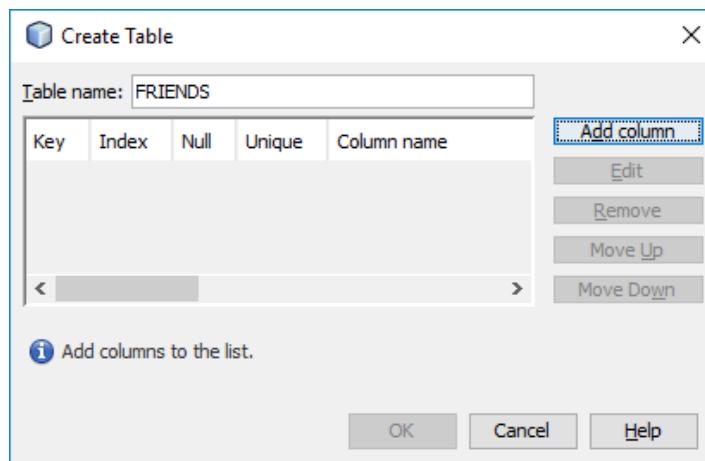


4. Now we are going to create a table in default database **sample**.

Right click on Table -> Create Table

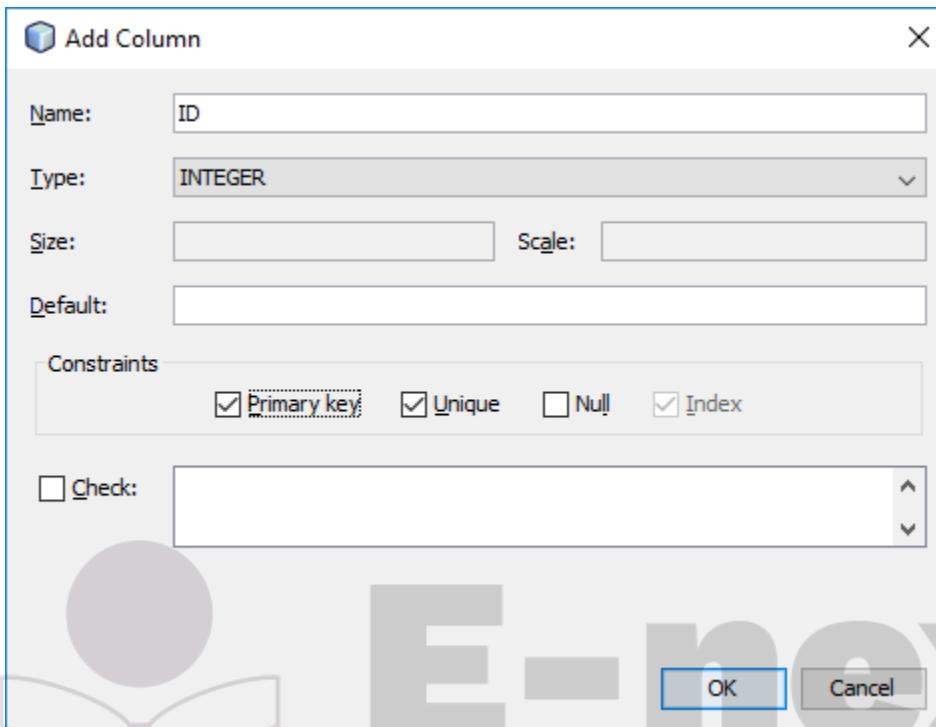


5. Give **table name as FRIENDS**.

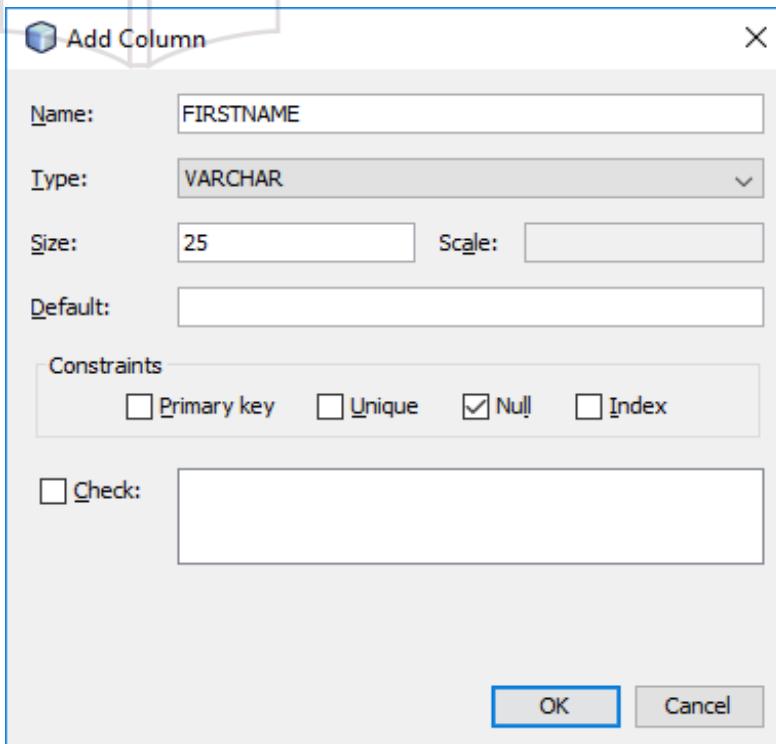


6. Now click on Add column button to add columns in table.

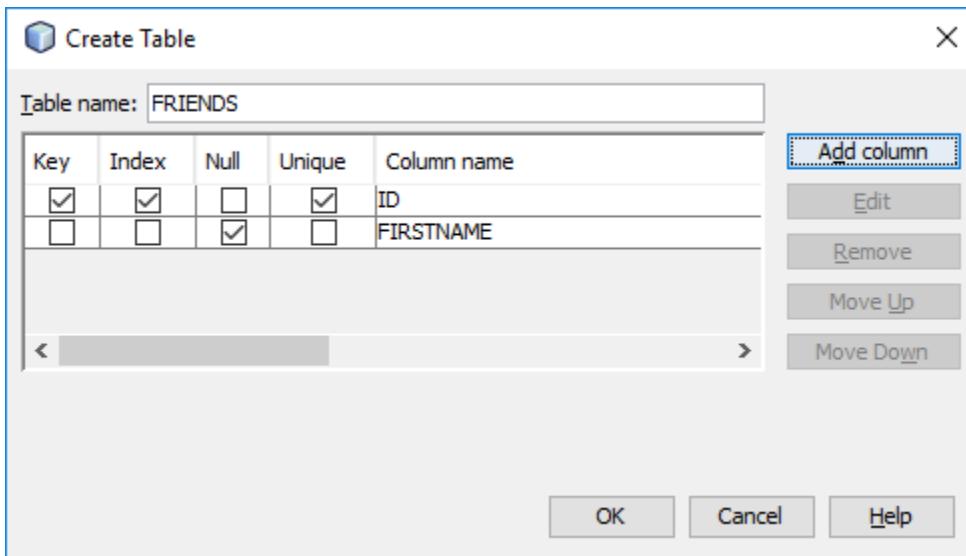
Enter details as in below pic and select Primary key. After that click on OK button.



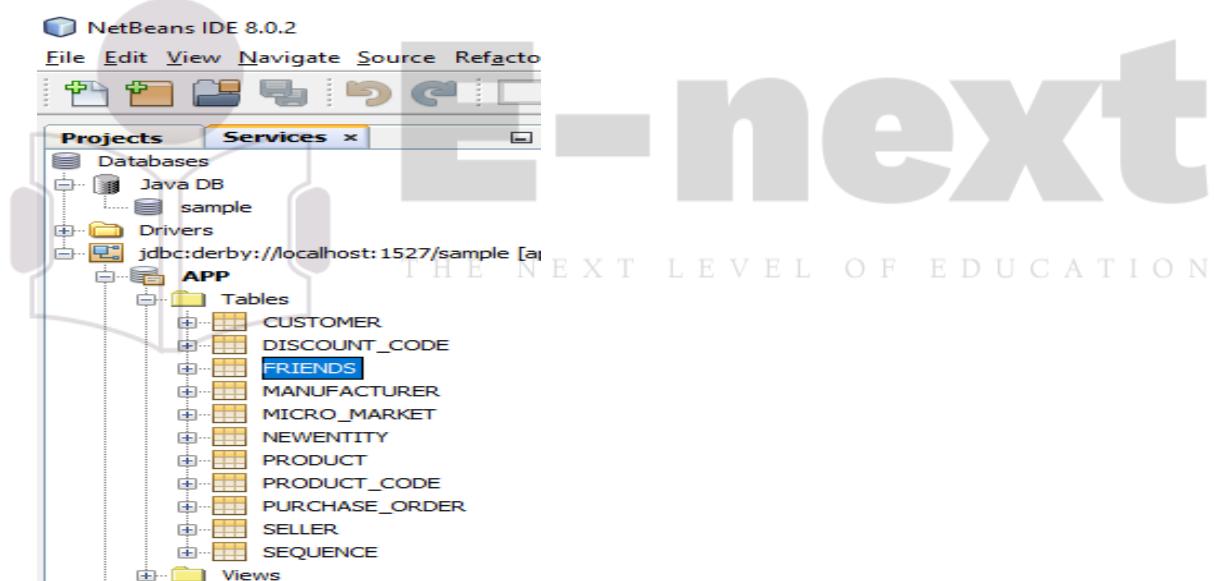
7. Now add second column with following detail. But don't select primary and click on OK button.



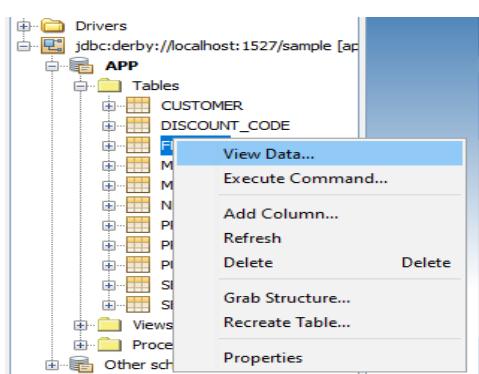
8. Now click on OK button.



9. Now you can see a table with name **FRIENDS** in the table.



10. Right click on FRIENDS to view and add records into it.

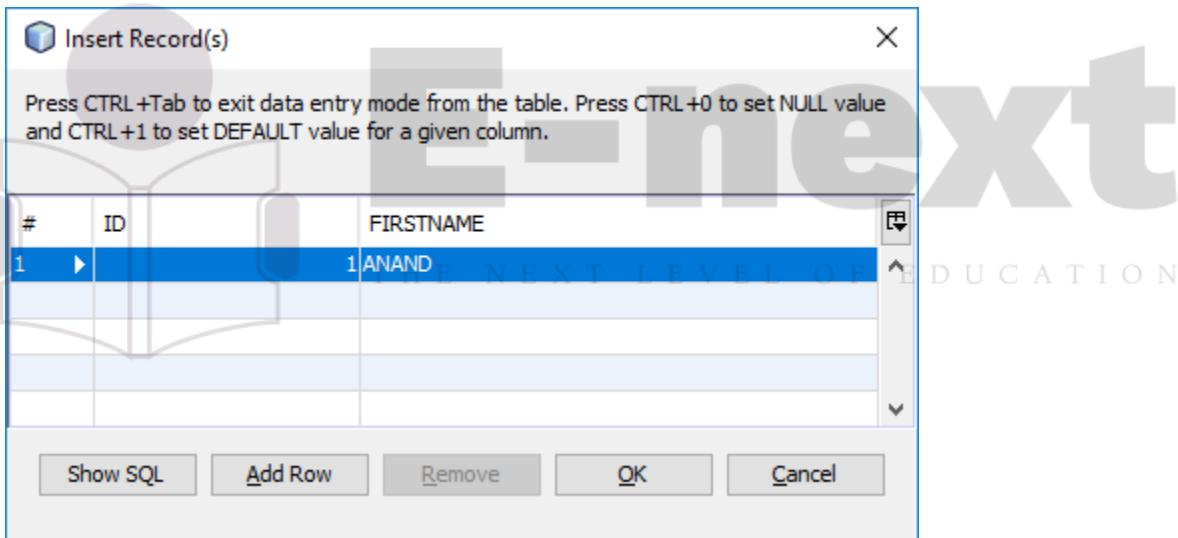


11. Now click on the leftmost icon in second panel to insert some record.

The screenshot shows the SQL Workbench/J application. On the left, the 'Projects' tab is selected, displaying a tree structure with 'Databases', 'Java DB', 'Drivers', and a connection to 'jdbc:derby://localhost:1527/sample [app]'. Under 'APP', there are several tables: CUSTOMER, DISCOUNT_CODE, FRIENDS (which is selected), MANUFACTURER, MICRO_MARKET, NEWENTITY, PRODUCT, PRODUCT_CODE, PURCHASE_ORDER, and SELLER. The right panel has a 'Start Page' tab and an 'SQL 1 [jdbc:derby://localhost:1527/sample [app on APP]]' tab. The SQL editor contains the query 'select * from APP.FRIENDS;'. Below it is a results grid with one row: # 1 and ID 1. At the bottom of the results grid is a button labeled '# Insert Record(s) (Alt+I)'.

12. Insert a record and then click on Add Row button to insert more record.

After that click on OK button to finish.

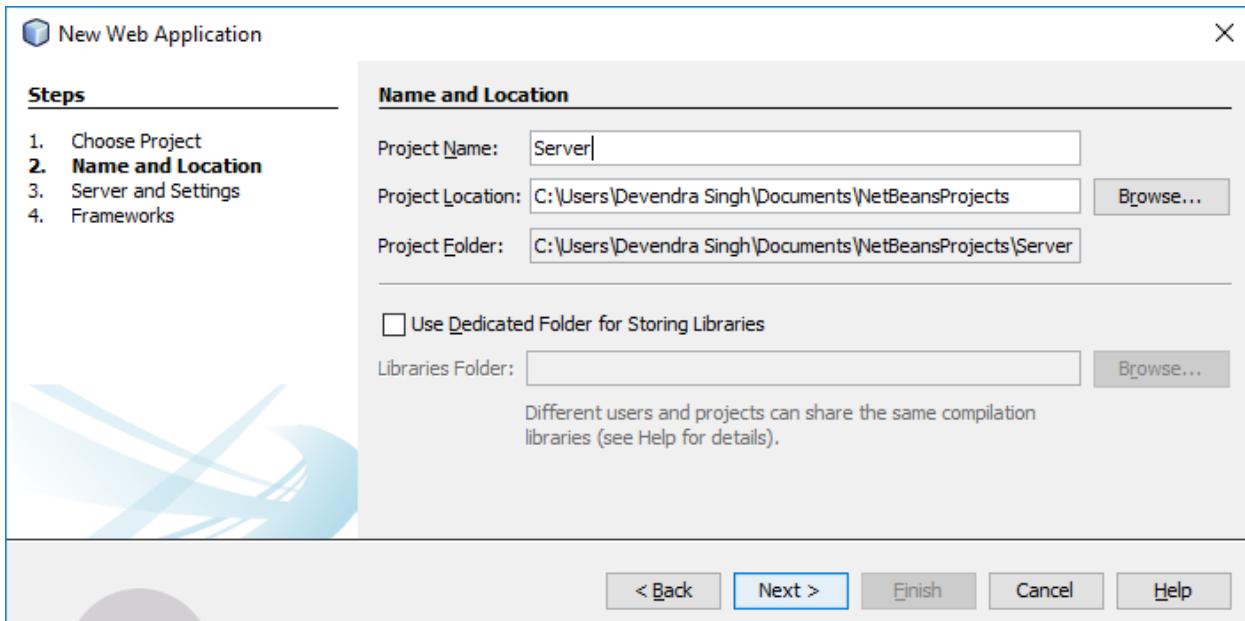


13. As you can see, I have entered 7 records.

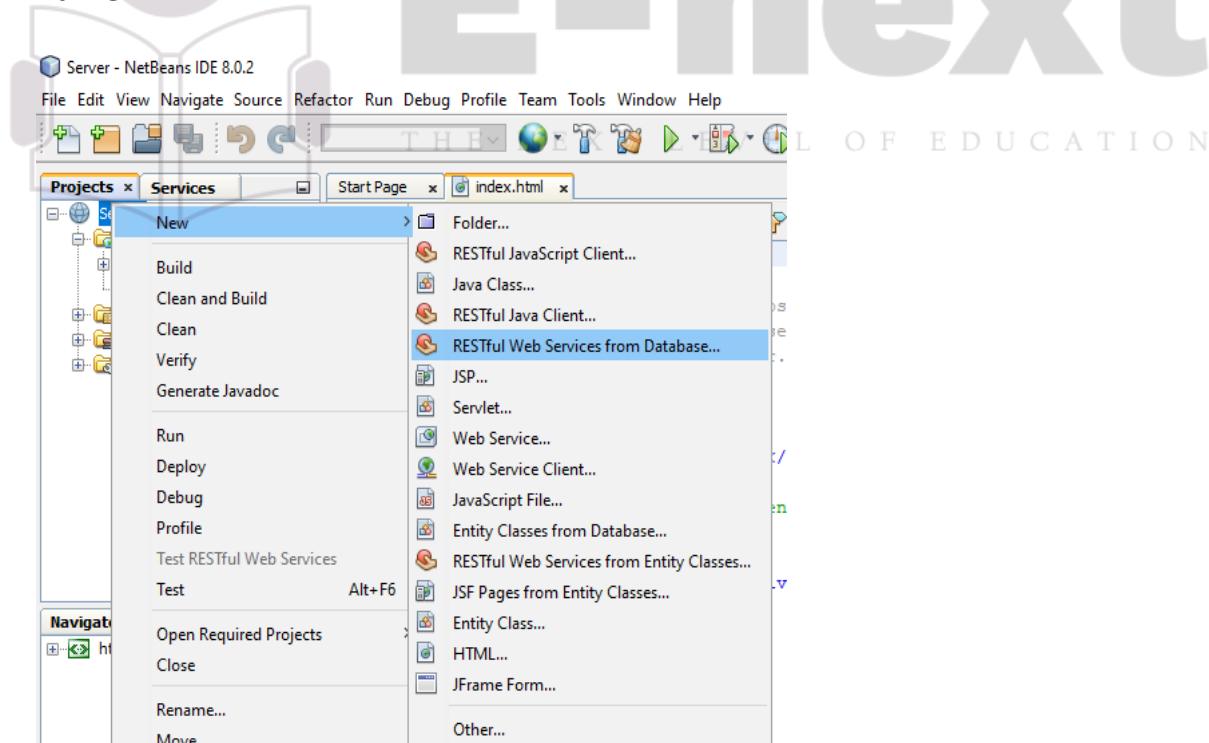
The screenshot shows the results of the 'select * from APP.FRIENDS;' query. The results grid has columns '#', 'ID', and 'FIRSTNAME'. The data is as follows:

#	ID	FIRSTNAME
1	1	ANAND
2	2	JULHAS
3	3	NIKHIL
4	4	GAGAN
5	5	RAVI
6	6	DHARMENDRA
7	7	ADARSH

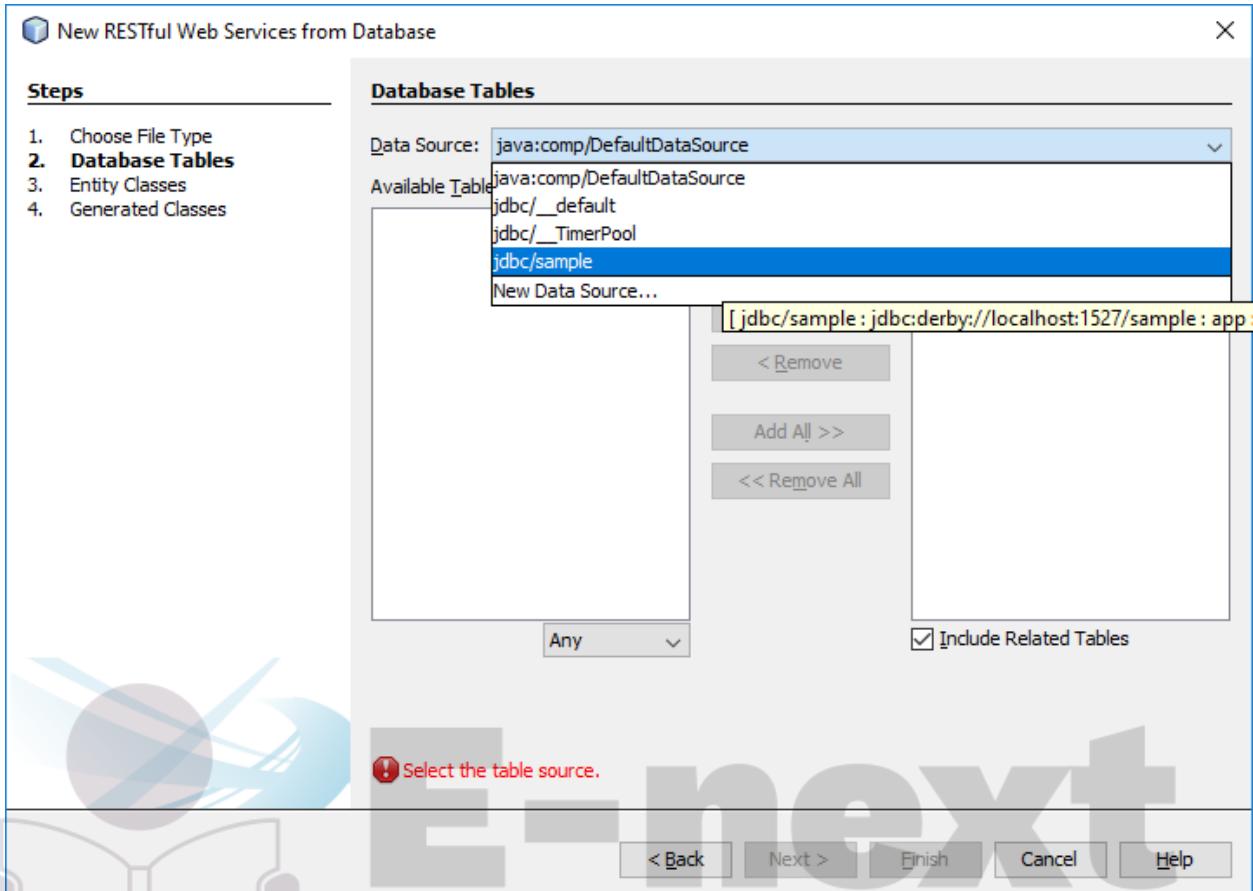
14. Now create a web application with name **Server**. After that click on **Next** and then **Finish** button.



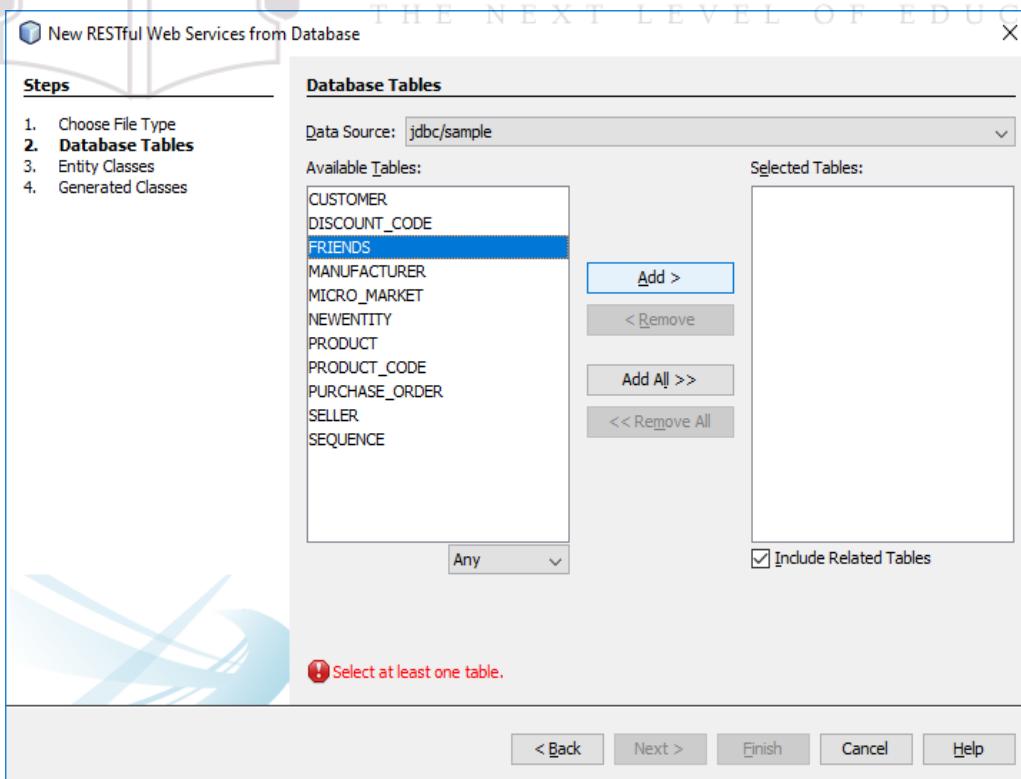
15. Now create a RESTful Web Service from Database by right click on project name.



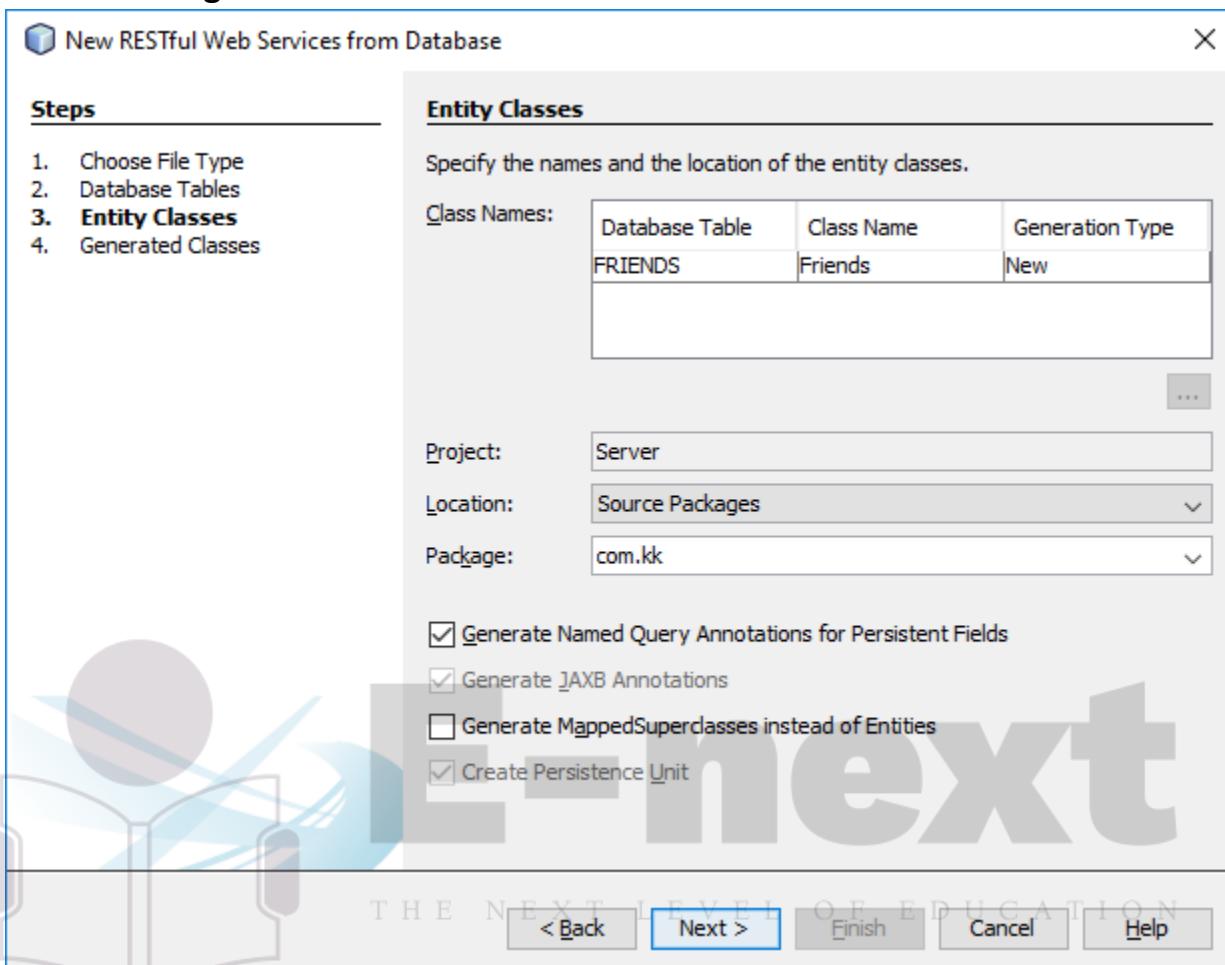
16. Choose Data Source **jdbc/sample**.



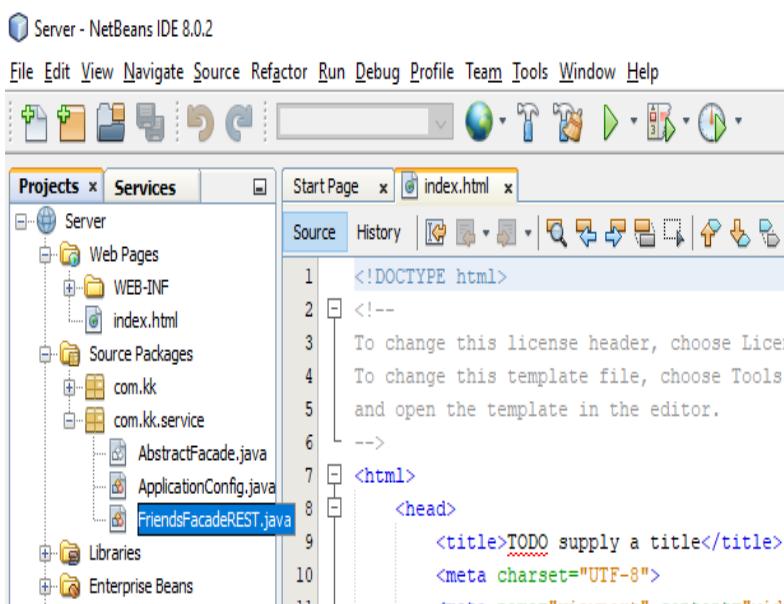
17. Now select FRIENDS and click on Add button. After that click on Next button.



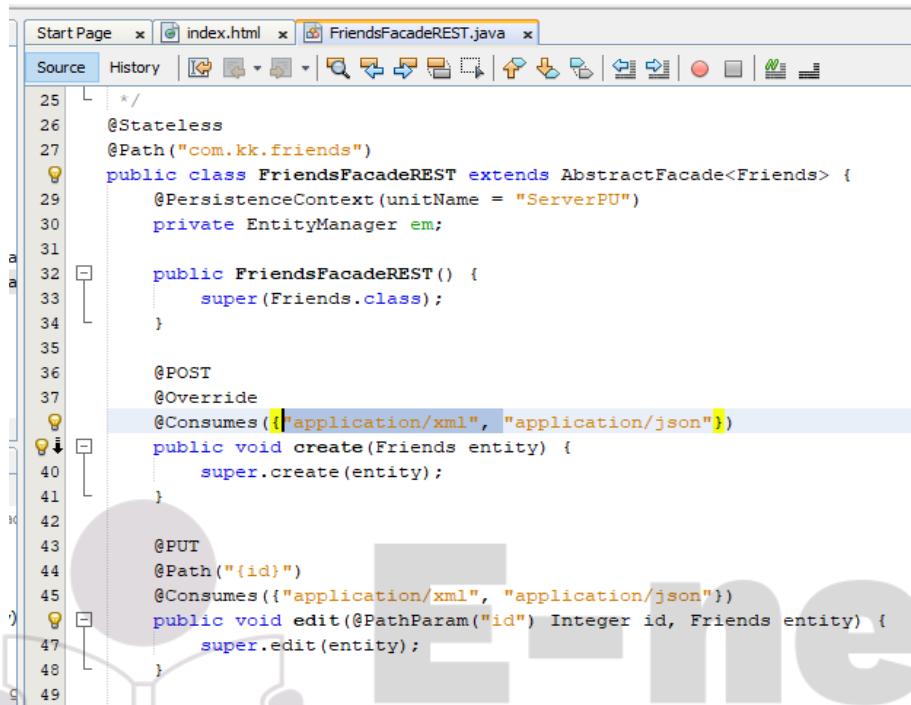
18. Enter Package name as com.kk and click on Next button and then Finish.



19. Now open selected file by double click on it.



20. Now remove the selected part from every method in this file. So that it will communicate only in JSON format. You can also use methods to convert it. But this is easiest method.



The screenshot shows the NetBeans IDE interface with the file 'FriendsFacadeREST.java' open. The code is a Java class for a RESTful web service. A specific line of code is highlighted with a yellow background:

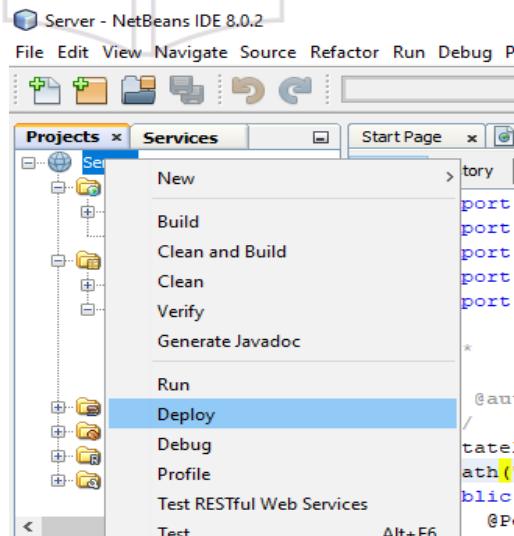
```
    @Consumes({"application/xml", "application/json"})
```

This line is part of a method annotated with @POST and @Override. The method signature is:

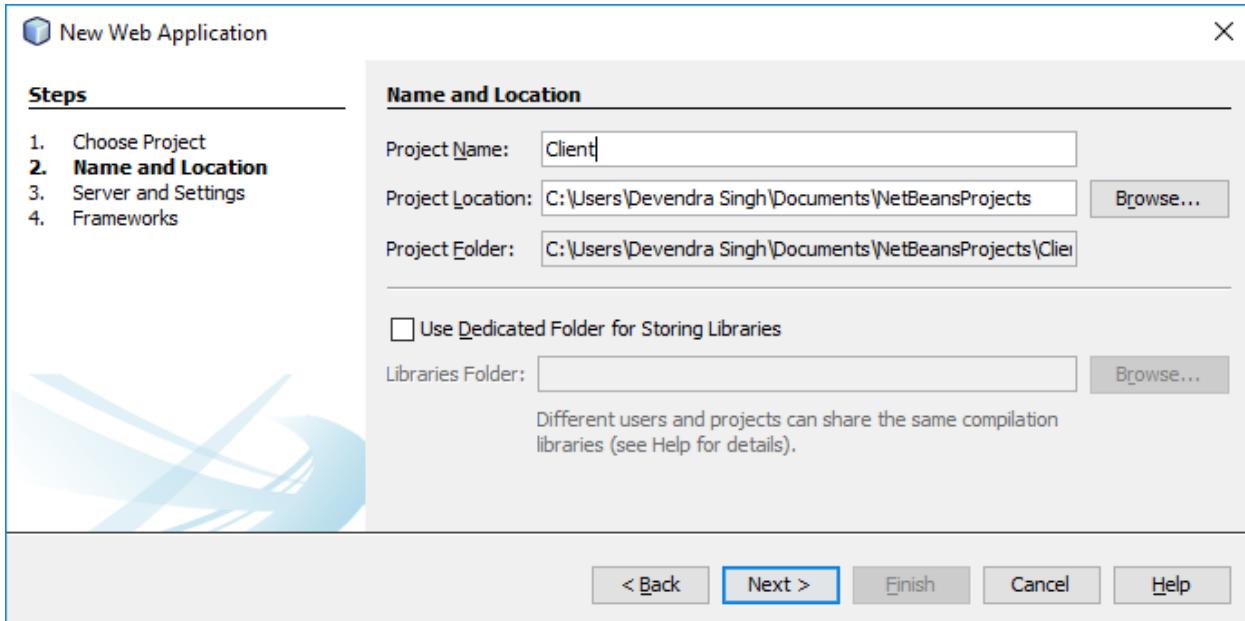
```
public void create(Friends entity) { super.create(entity); }
```

Below this, there is another method annotated with @PUT and @Path("{id}").

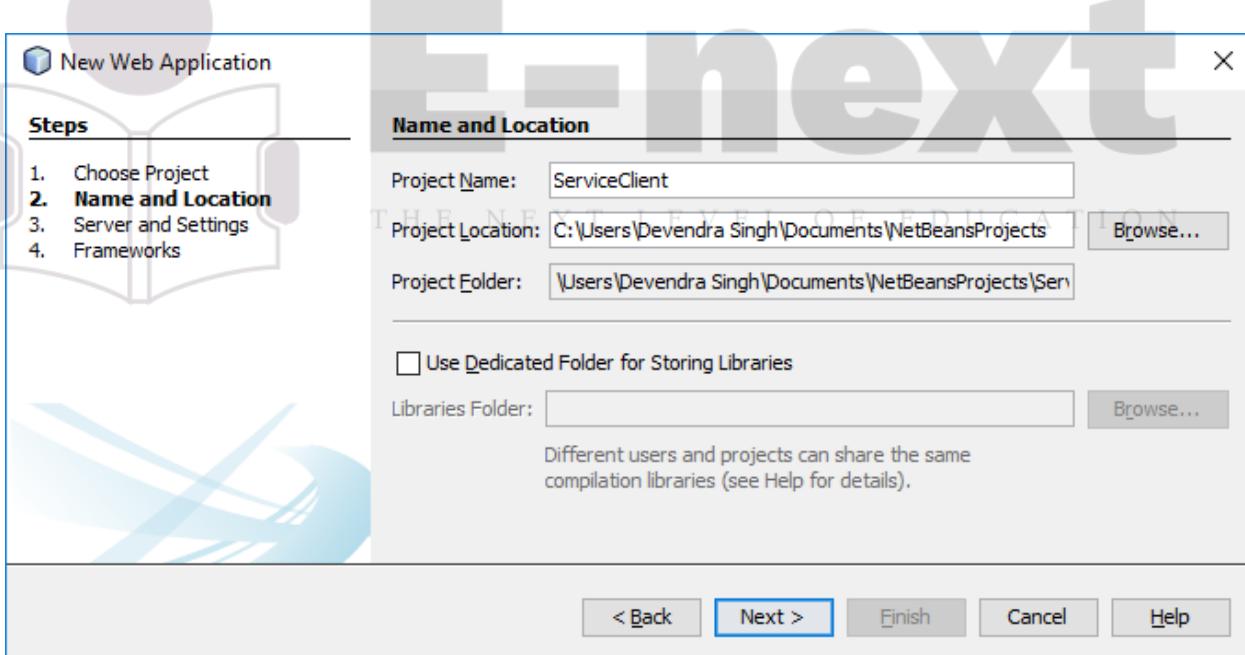
21. After that right click on project name and Deploy it.



22. Now create one more Web Application as Client. After that click on Next and then Finish button.

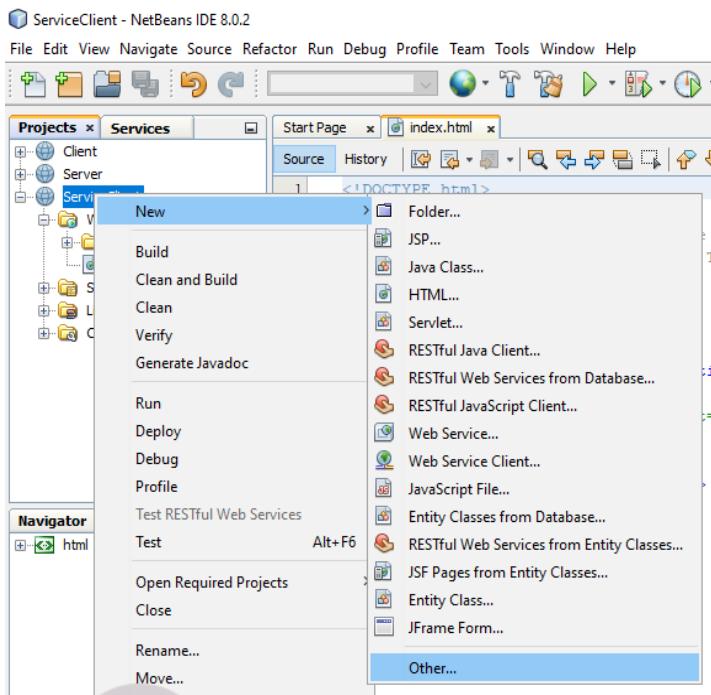


23. Create a Web Application with name ServiceClient.

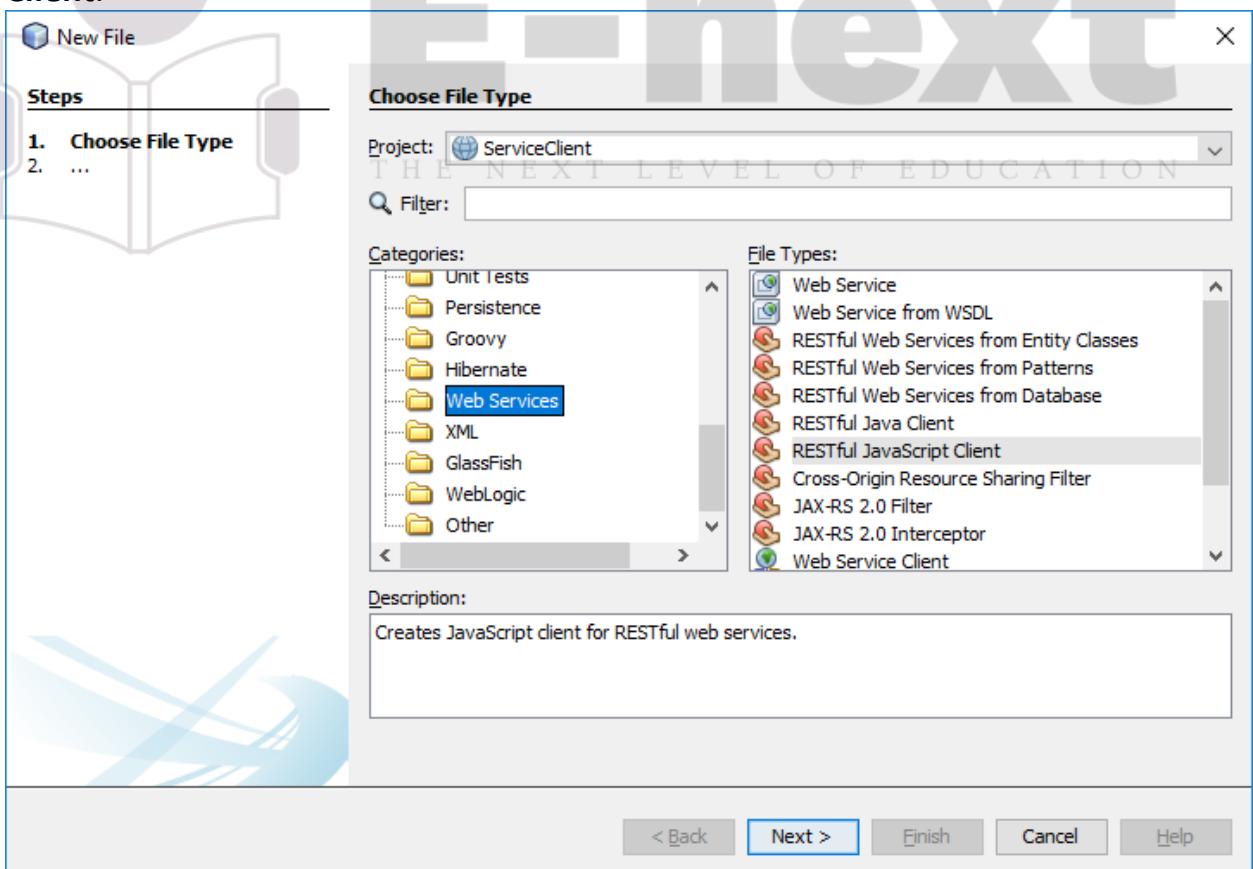


24. Now create a RESTful Java Client.

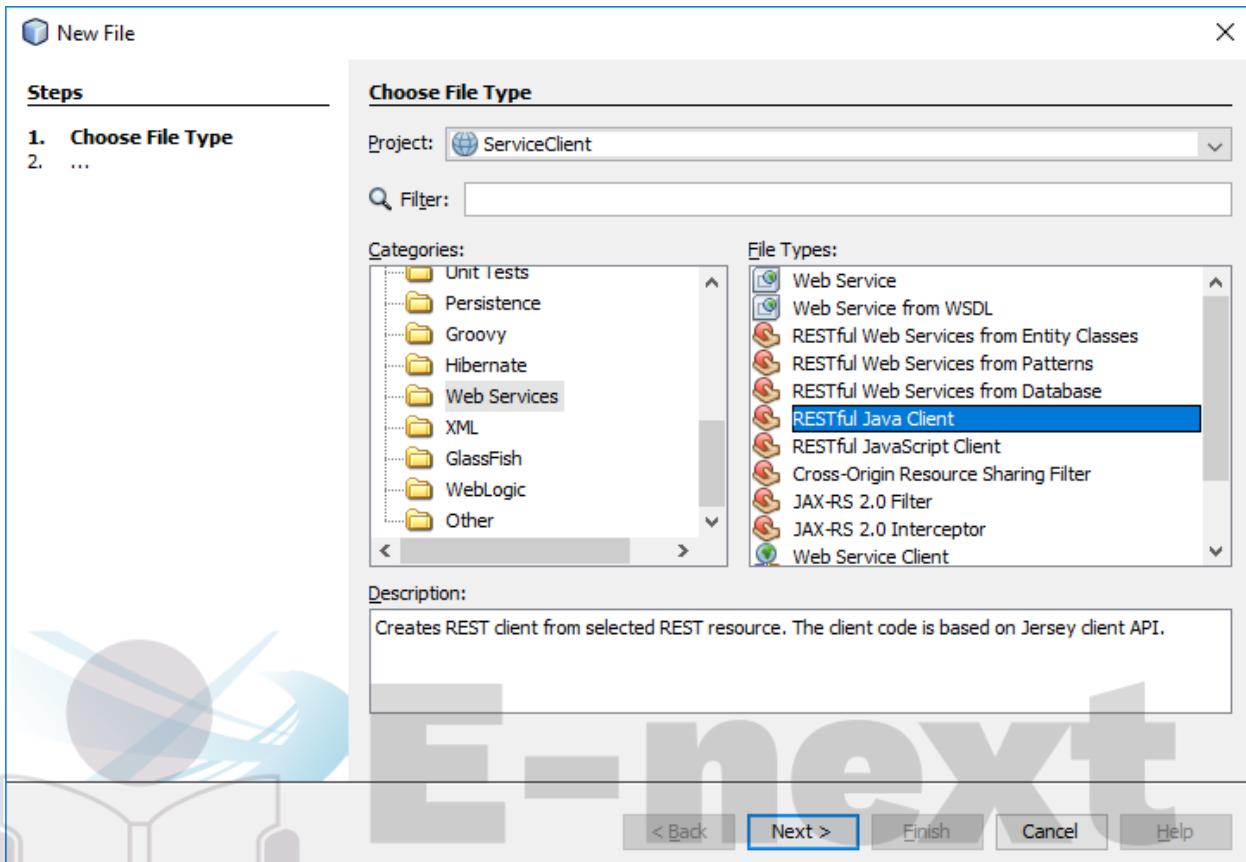
Right click on ServiceClient -> New -> Other.



25. Drag down and select Web Services and in side panel select RESTful Java Client.



26. After select RESTful Java Client click on Next.

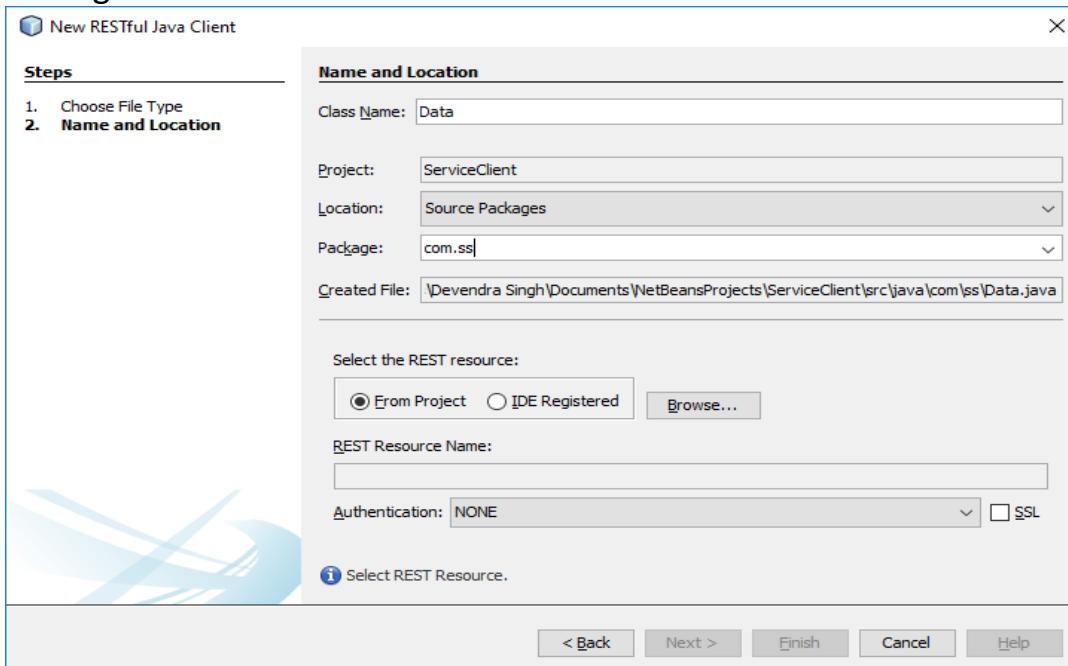


27. Enter following data.

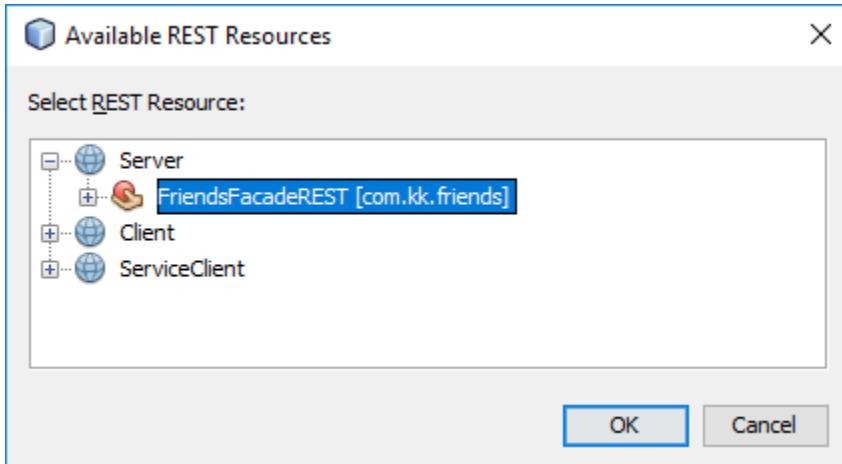
THE NEXT LEVEL OF EDUCATION

Class Name -> Data

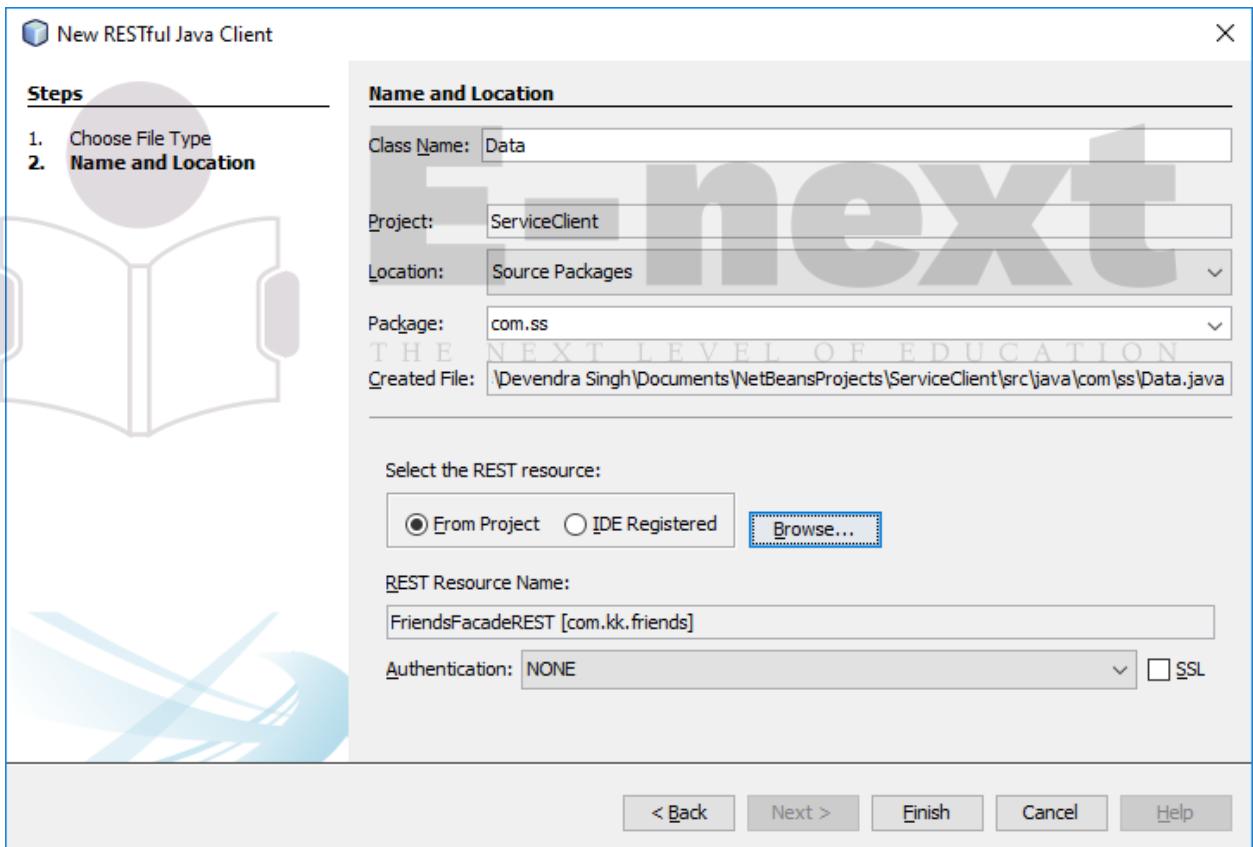
Package -> com.ss



28. Now click on **Browse** button and select the option into the below pic. After select click on **OK** button.

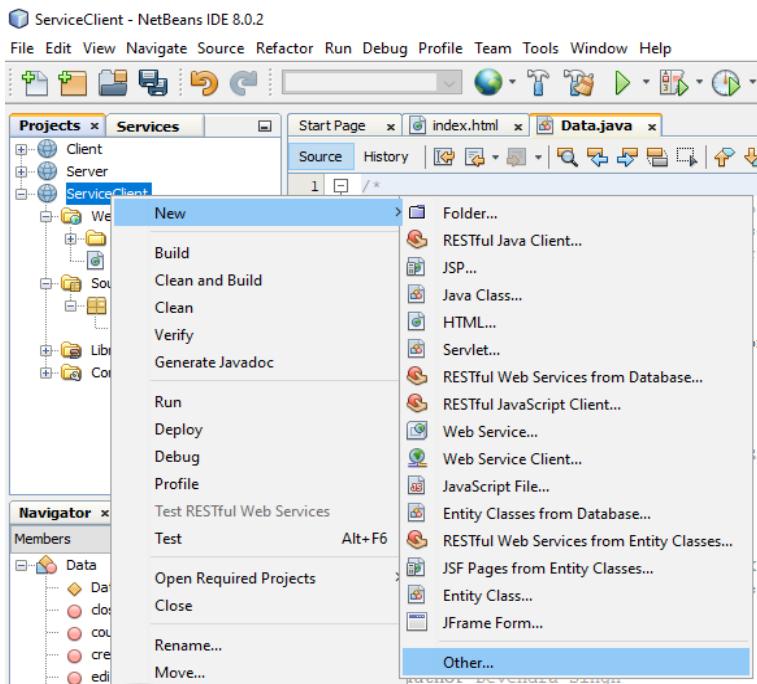


29. Click on **Finish**.

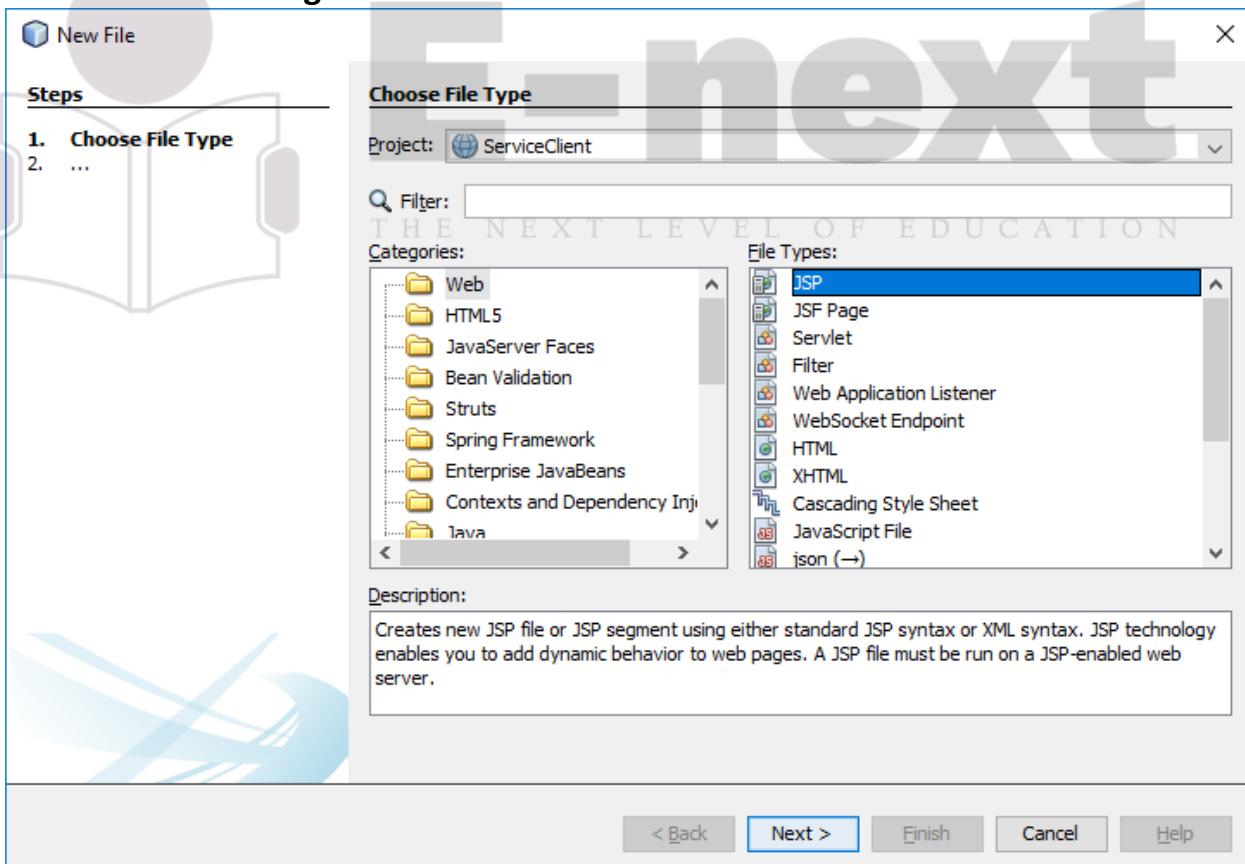


30. Now create a JSP page.

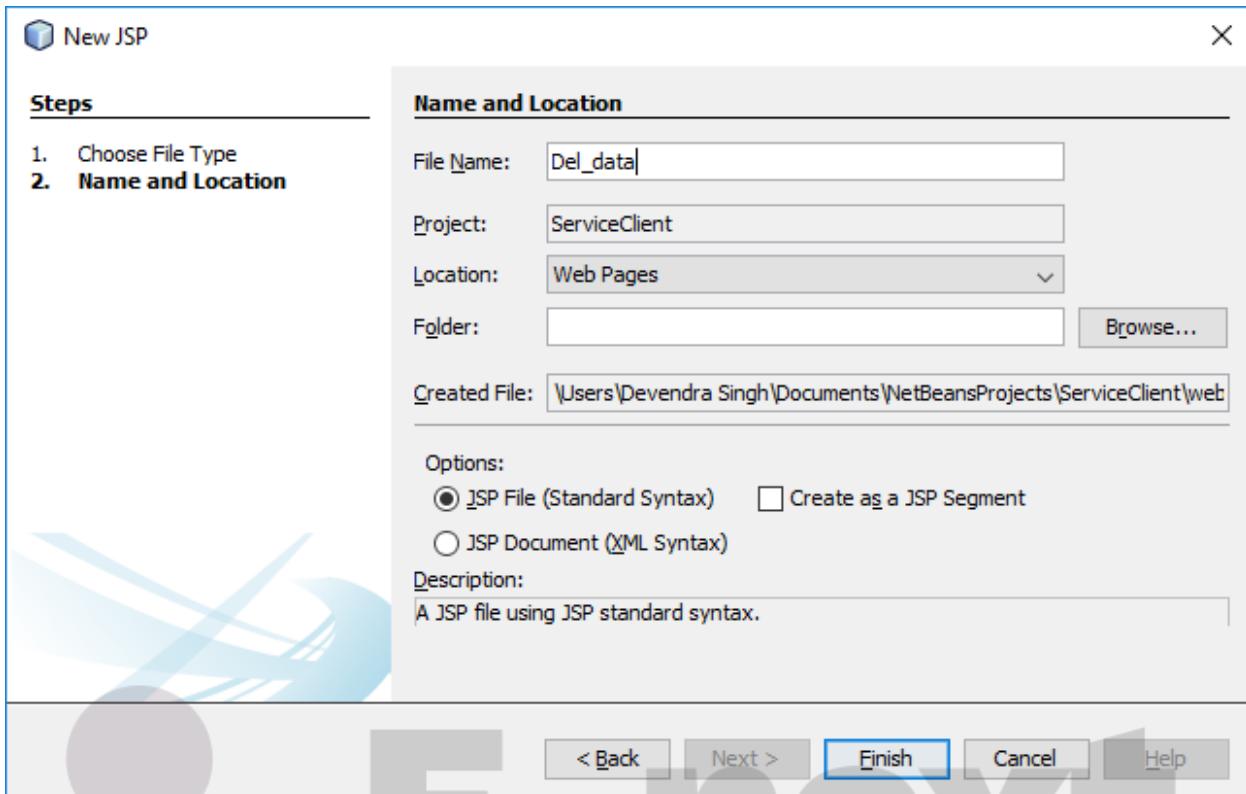
Right click on ServiceClient -> New -> Other



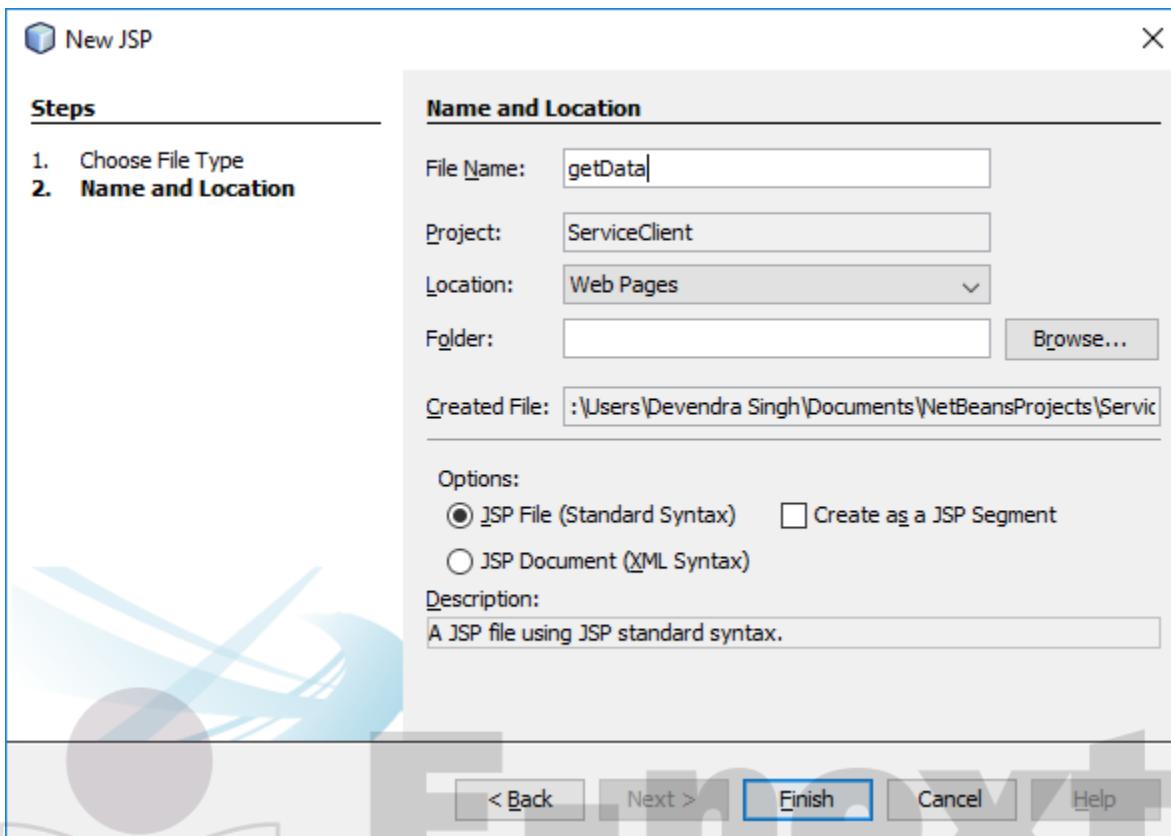
31. Select Web in Categories section -> Select JSP and click on Next button.



32. Enter File Name Del_data and click on Finish button.



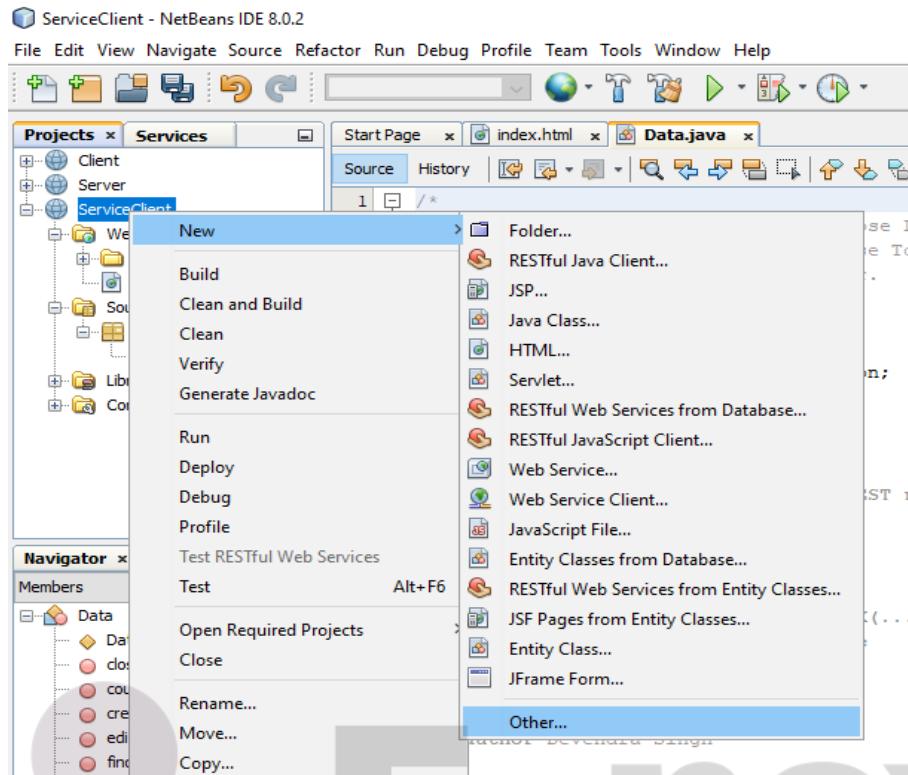
33. Now create one more JSP file by follow the step number 30, 31 & 32. But File Name will be getData.



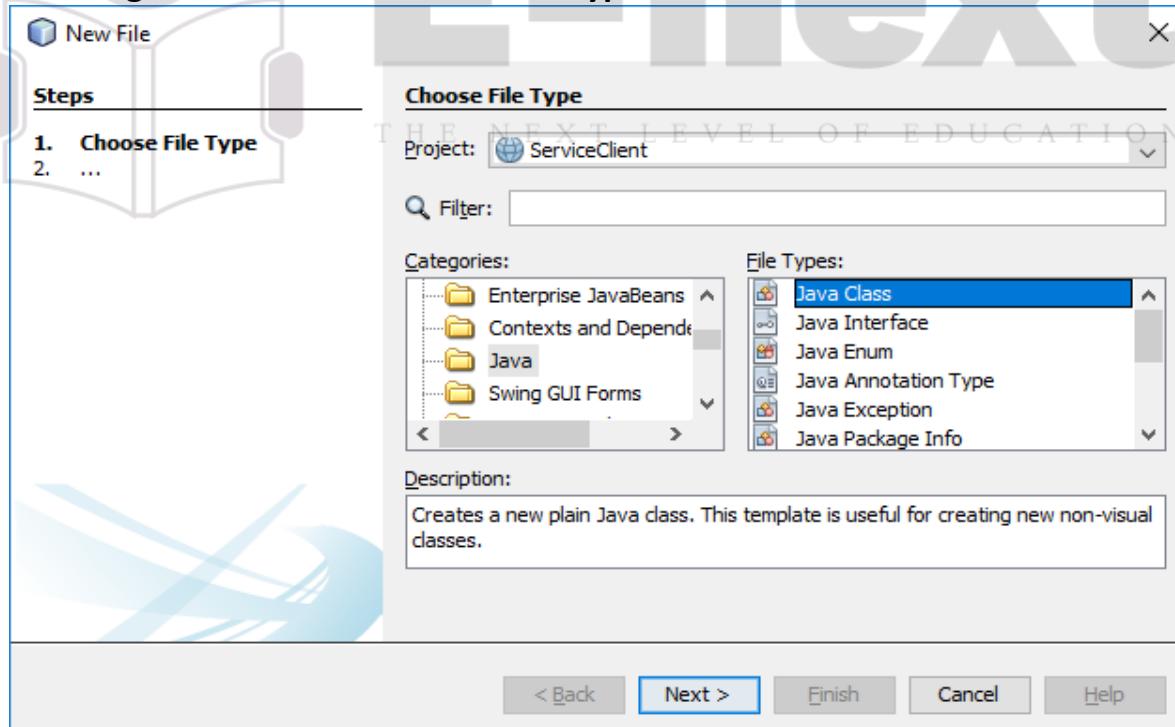
34. Now create a Java class.

Right click on ServiceClient -> New -> Other

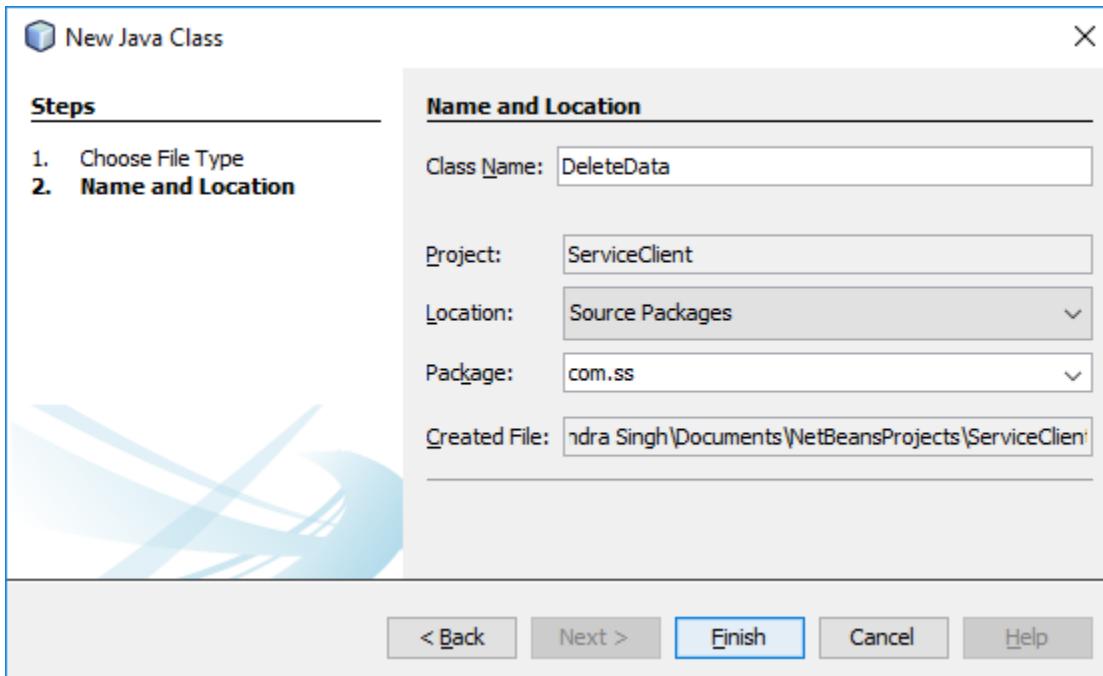
THE NEXT LEVEL OF EDUCATION



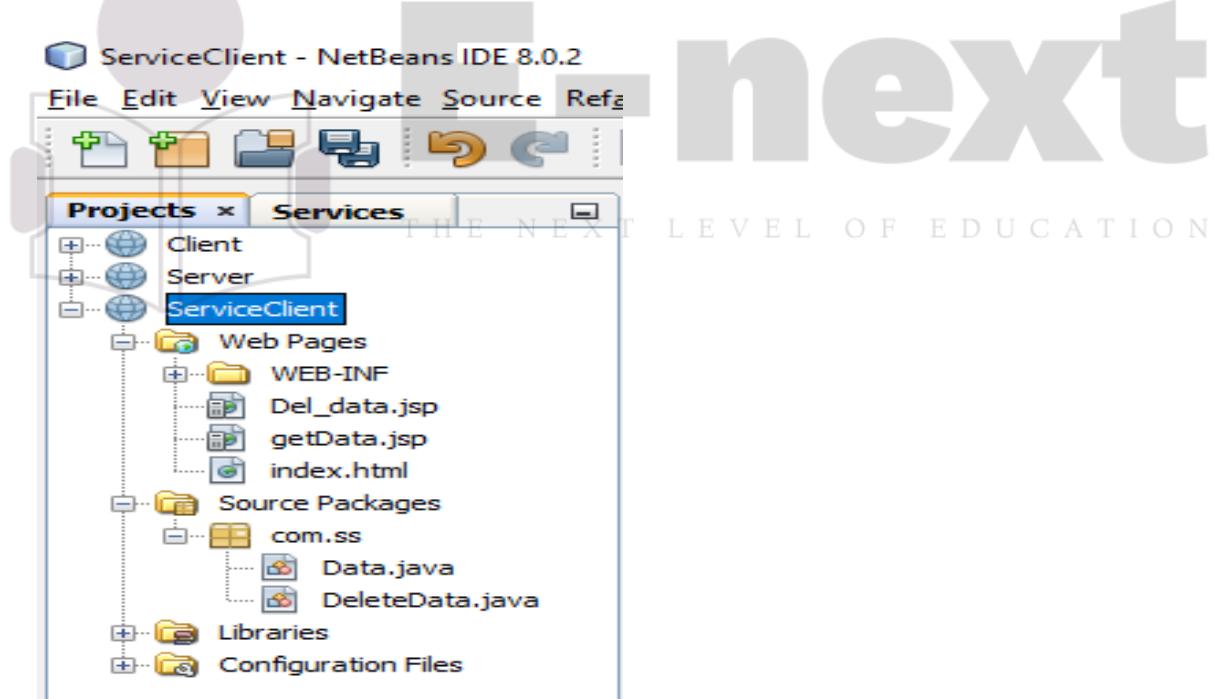
35. In Categories select Java and in File Types select Java Class. Click Next button.



36. Enter Class Name DeleteData and Package com.ss. After that click on Finish.



37. Your project file structure will look like below.



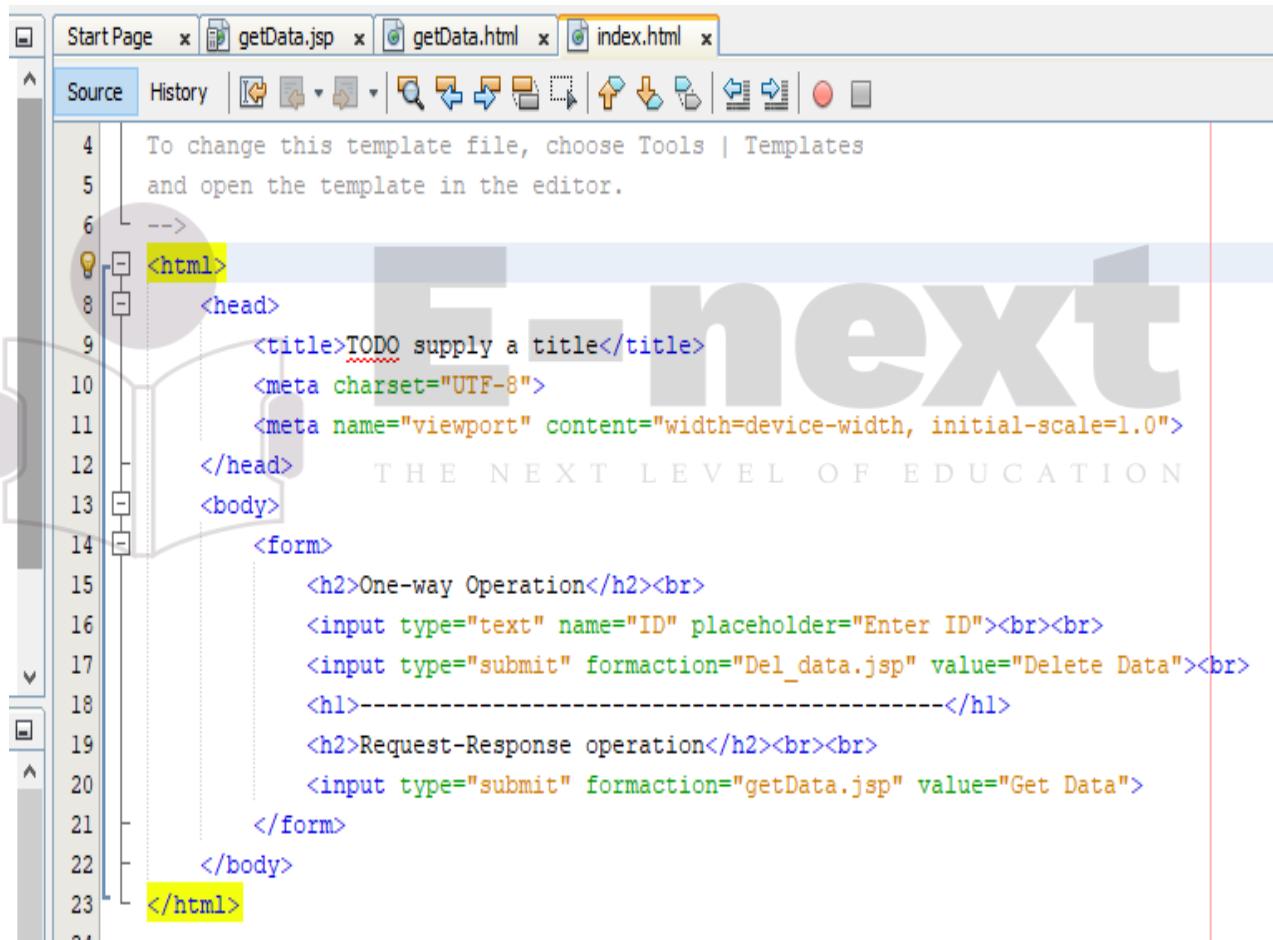
38. Now open the index.html of ServiceClient project by double click on it and add the following code in between body tag.

```
<form>
<h2>One-way Operation</h2><br>
```

```
<input type="text" name="ID" placeholder="Enter ID"><br><br>
<input type="submit" formaction="Del_data.jsp" value="Delete
Data"><br>

<h1>-----</h1>

<h2>Request-Response operation</h2><br><br>
<input type="submit" formaction="getData.jsp" value="Get Data">
</form>
```



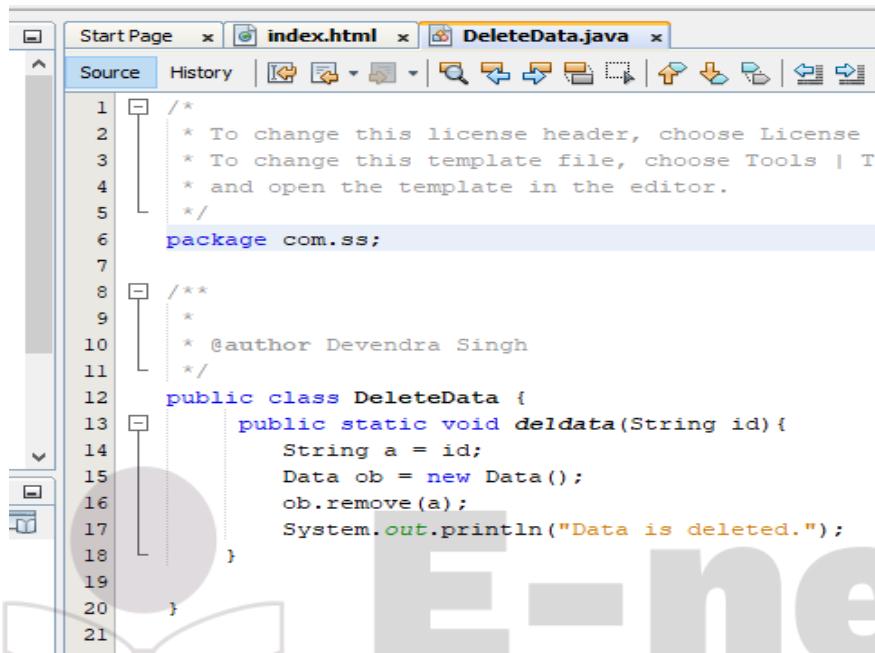
The screenshot shows a web development environment with four tabs at the top: "Start Page", "getData.jsp", "getData.html", and "index.html". The "index.html" tab is selected and displays the following HTML code:

```
4 | To change this template file, choose Tools | Templates
5 | and open the template in the editor.
6 |
7 | <html>
8 |   <head>
9 |     <title>TODO supply a title</title>
10 |    <meta charset="UTF-8">
11 |    <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 |   </head>      THE NEXT LEVEL OF EDUCATION
13 |   <body>
14 |     <form>
15 |       <h2>One-way Operation</h2><br>
16 |       <input type="text" name="ID" placeholder="Enter ID"><br><br>
17 |       <input type="submit" formaction="Del_data.jsp" value="Delete Data"><br>
18 |       <h1>-----</h1>
19 |       <h2>Request-Response operation</h2><br><br>
20 |       <input type="submit" formaction="getData.jsp" value="Get Data">
21 |     </form>
22 |   </body>
23 | </html>
```

39. Now open DeleteData.java file by double click on it and add the following code in the class and save it by pressing Ctrl+S.

```
public static void deldata(String id){
    String a = id;
```

```
Data ob = new Data();
ob.remove(a);
System.out.println("Data is deleted.");
}
```



```
/*
 * 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 com.ss;

/**
 *
 * @author Devendra Singh
 */
public class DeleteData {
    public static void deldata(String id) {
        String a = id;
        Data ob = new Data();
        ob.remove(a);
        System.out.println("Data is deleted.");
    }
}
```

40. Now open the Del_data.jsp file and replace the contents of body with the following code.

THE NEXT LEVEL OF EDUCATION

```
<%@ page import="com.ss.DeleteData"%>
<%
    String id = request.getParameter("ID");
    DeleteData.deldata(id);
%>
```

The screenshot shows a Java IDE interface. On the left, the 'Files' view displays a project structure under 'Client' and 'Server'. Under 'Client/WEB-INF', there are 'index.html', 'getData.html', and 'Del_data.jsp'. Under 'Server/Web Pages', there are 'index.html', 'getData.jsp', and 'Del_data.jsp'. The central area is the 'Source' tab of the code editor, showing the JSP code for 'Del_data.jsp'. The code includes a header with document details, a page directive, an HTML structure with a head and body section, and a scriptlet that imports a class and calls a method to delete data based on a parameter. The 'Navigator' view on the right shows the structure of the 'html' tag, including 'head', 'meta', 'title', and 'body'.

```

<%-->
Document      : Del_data
Created on   : Aug 16, 2018, 8:47:49 PM
Author        : Devendra Singh
--%>

<%@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>
        <%@ page import="com.ss.DeleteData"%>
        <%
            String id = request.getParameter("ID");
            DeleteData.deldata(id);
        %>
    </body>
</html>

```

41. Now open the **getData.jsp** file and replace the contents of **html tag** with the following code and save it.

THE NEXT LEVEL OF EDUCATION

```

<head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <style>
        table {
            font-family: arial, sans-serif;
            border-collapse: collapse;
        }

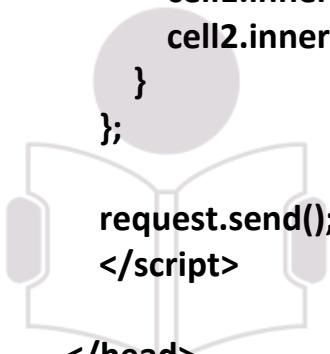
        td, th {
            border: 1px solid #000000;
            text-align: center;
            padding: 8px;
        }
    </style>

```

```
</style>
<script>
    var request = new XMLHttpRequest();
    request.open('GET',
    'http://localhost:8080/Server/webresources/com.kk.friends/', true);
    request.onload = function () {
        // begin accessing JSON data here
        var data = JSON.parse(this.response);

        for (var i = 0; i < data.length; i++) {
            var table = document.getElementById("myTable");
            var row = table.insertRow();
            var cell1 = row.insertCell(0);
            var cell2 = row.insertCell(1);
            cell1.innerHTML = data[i].id;
            cell2.innerHTML = data[i].firstname;
        }
    };
    request.send();
</script>
</head>
<body>
    <table id="myTable">
        <tr>
            <th> ID</th>
            <th>NAME</th>
        </tr>
    </table>

</body>
```



E-next

THE NEXT LEVEL OF EDUCATION

The screenshot shows the NetBeans IDE interface with the following details:

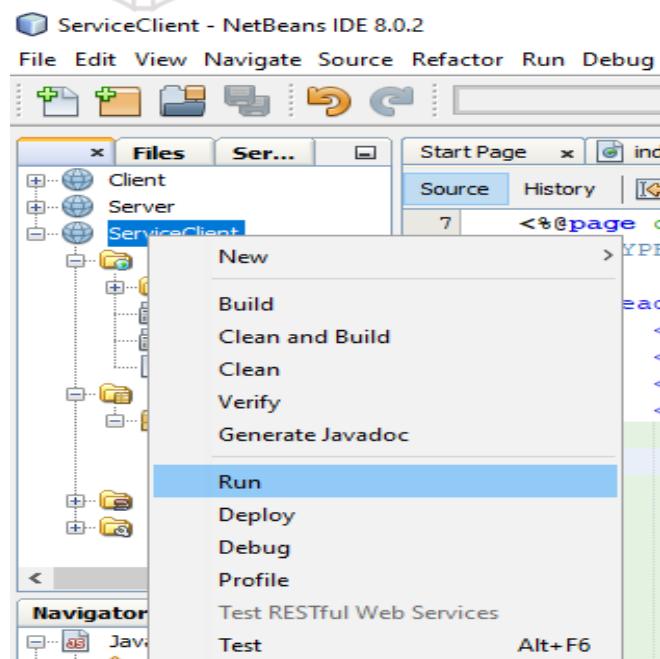
- Project Explorer:** Shows the **ServiceClient** project structure under the **Client** node. It includes **Server**, **ServiceClient**, **Web Pages** (containing **WEB-INF** with **index.html** and **getData.jsp**), **Source Packages** (containing **com.ss** with **Data.java** and **DeleteData.java**), **Libraries**, and **Configuration Files**.
- Navigator:** Shows the **JavaScript** and **CSS** resources used in the project.
- Code Editor:** Displays the **index.html** and **getData.jsp** files. The **getData.jsp** file contains JSP and JavaScript code to interact with a RESTful web service.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
            table {
                font-family: arial, sans-serif;
                border-collapse: collapse;
            }

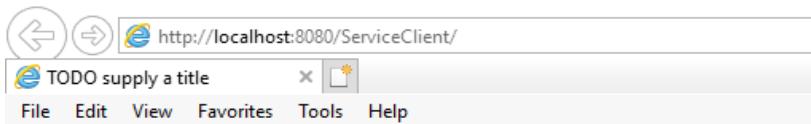
            td, th {
                border: 1px solid #000000;
                text-align: center;
                padding: 8px;
            }
        </style>
        <script>
            var request = new XMLHttpRequest();
            request.open('GET', 'http://localhost:8080/Server/webresources/com.kk.friends/', true);
            request.onload = function () {
                // begin accessing JSON data here
                var data = JSON.parse(this.response);

                for (var i = 0; i < data.length; i++) {
                    var table = document.getElementById("myTable");
                    var row = table.insertRow();
                    var cell1 = row.insertCell(0);
                    var cell2 = row.insertCell(1);
                    cell1.innerHTML = data[i].id;
                }
            }
        </script>
    </head>
    <body>
        <table border="1">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>John Doe</td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>Jane Doe</td>
                </tr>
                <tr>
                    <td>3</td>
                    <td>Mike Doe</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

42. Now Run the ServiceClient project.



43. On run the project following window will open in browser.

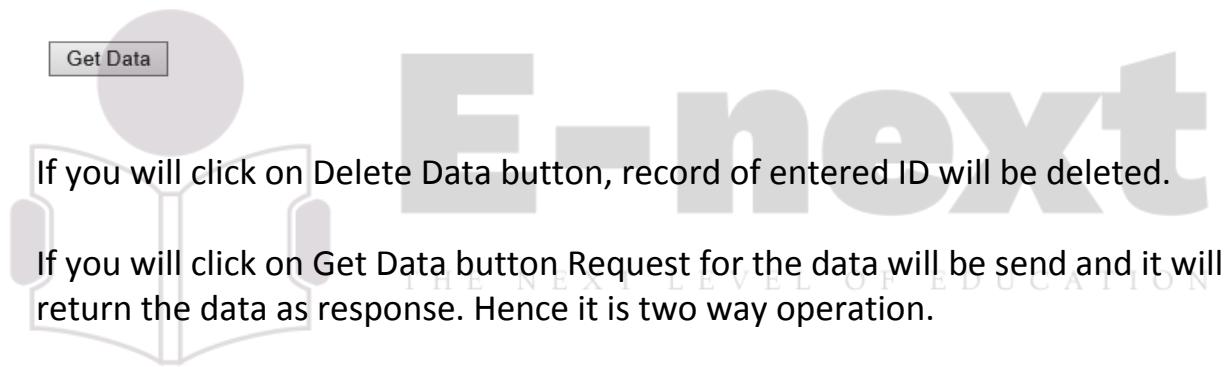


One-way Operation

Enter ID

Delete Data

Request-Response operation



44. By click on Get Data button.

A screenshot of a Microsoft Internet Explorer browser window. The address bar shows "http://localhost:8080/ServiceClient/". The title bar says "TODO supply a title". The menu bar includes File, Edit, View, Favorites, Tools, and Help. Below the menu is a toolbar with icons for back, forward, stop, and search. The main content area displays a table with columns "ID" and "NAME". The data rows are: 1 ANAND, 2 JULHAS, 3 NIKHIL, 4 RAVI, 5 GAGAN, 6 DHARMENDRA, 7 ADARSH, 8 DEVENDRA.

45. Entering ID 8 and clicking on Delete Data button.



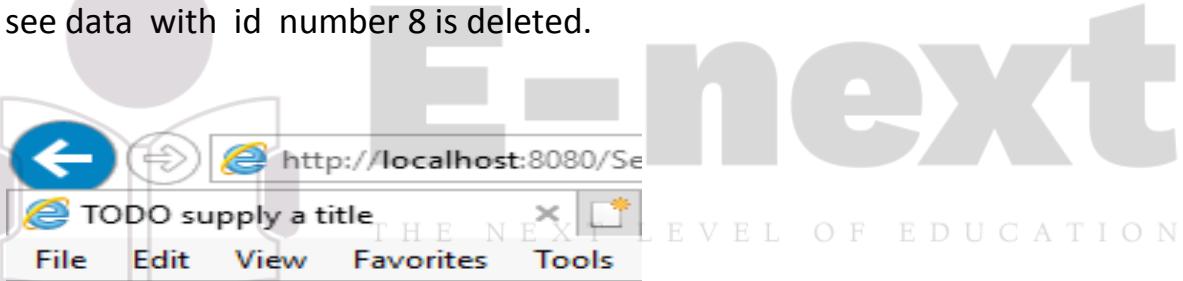
One-way Operation

Delete Data

Request-Response operation

Get Data

46. Now go back and again click on Get Data button. Refresh the page, You will see data with id number 8 is deleted.



ID	NAME
1	ANAND
2	JULHAS
3	NIKHIL
4	RAVI
5	GAGAN
6	DHARMENDRA
7	ADARSH

Mahesh Gurunani

Create new Web Project give it name pract_2

Press next

Press Finish

Right Click on Project pract_2 and select new then select web service

Give

Name to web service as myws

Give package name as ws

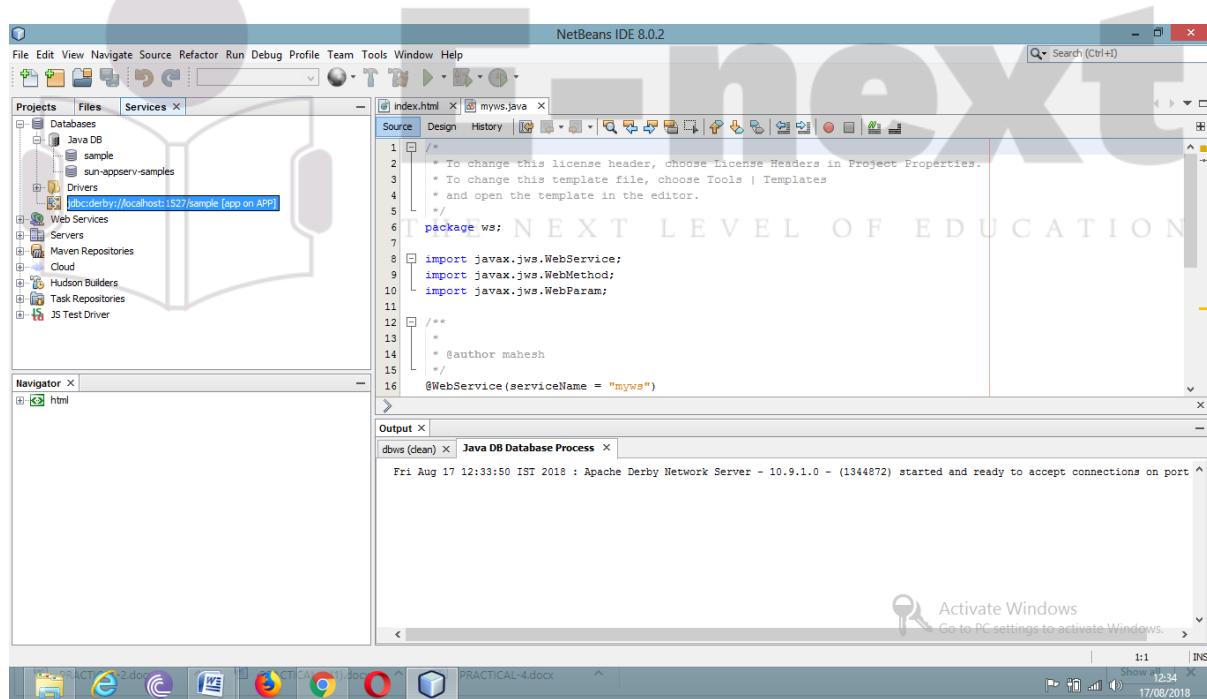
Press Finish

Go to service

Select java DB

Right click on java DB select Start server

Right click on connection string and select connect

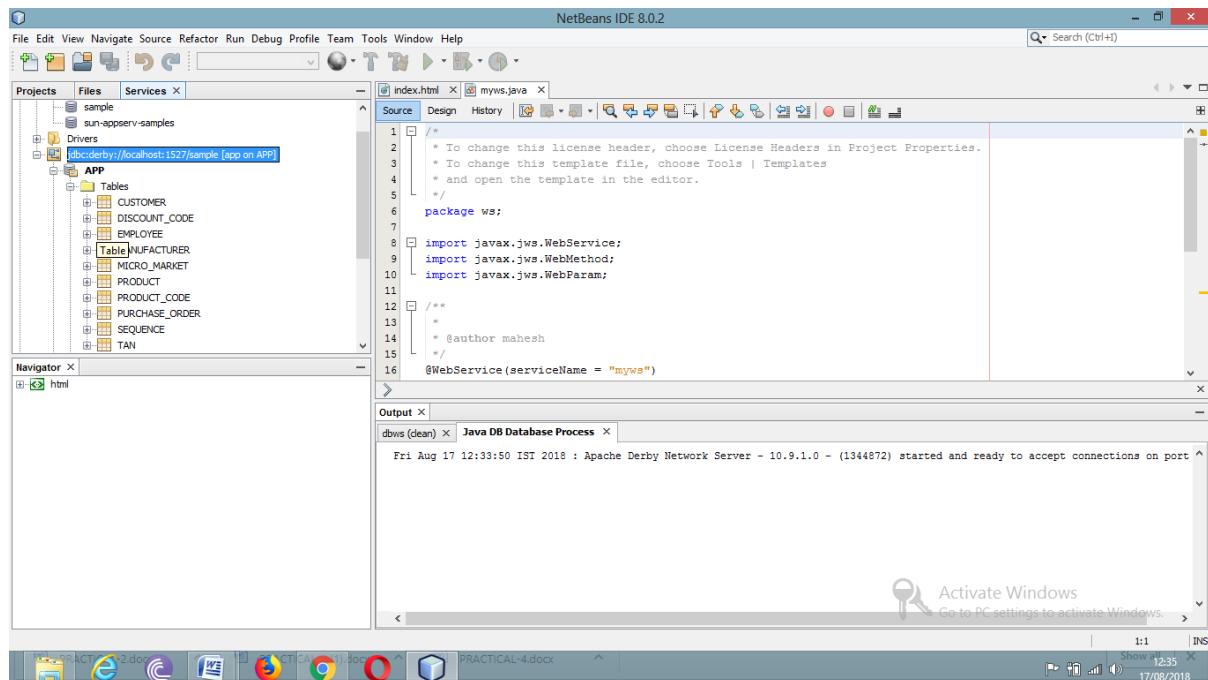


Expand connection string

It will show APP

Expand APP it will show table. Right click on table and select create table.

Mahesh Gurunani



Give name to table as tan

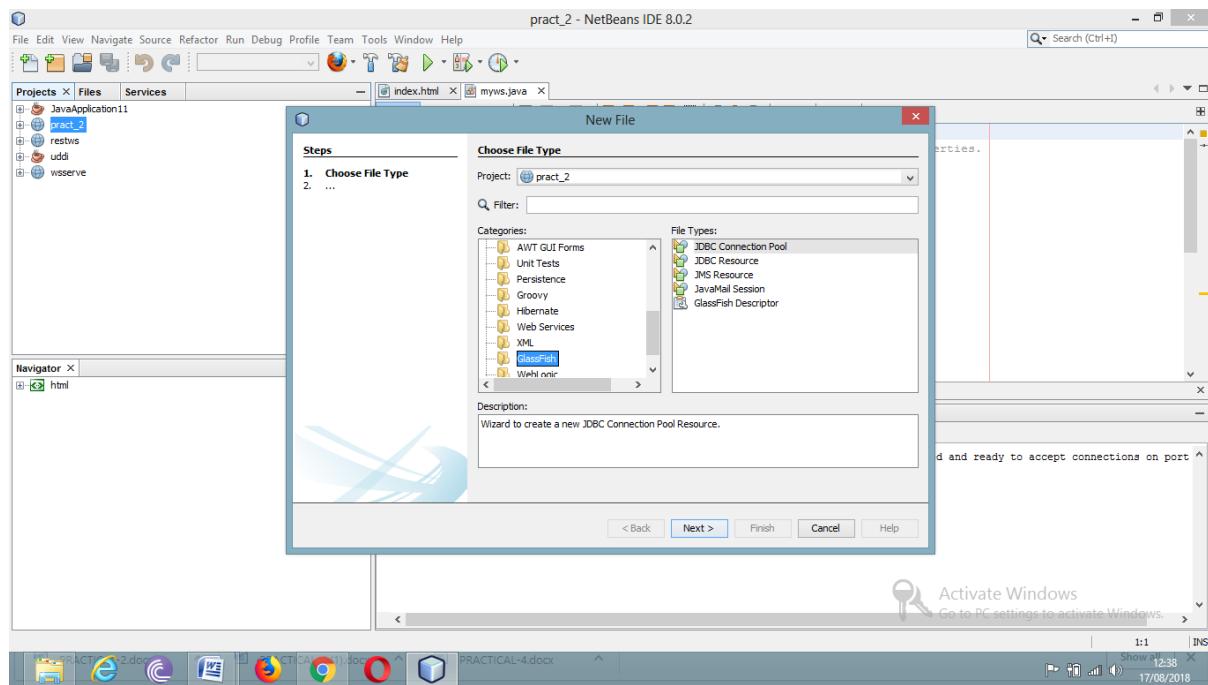
Then select add column

I have given name to table as Tan with column as rn and name

Come back to project

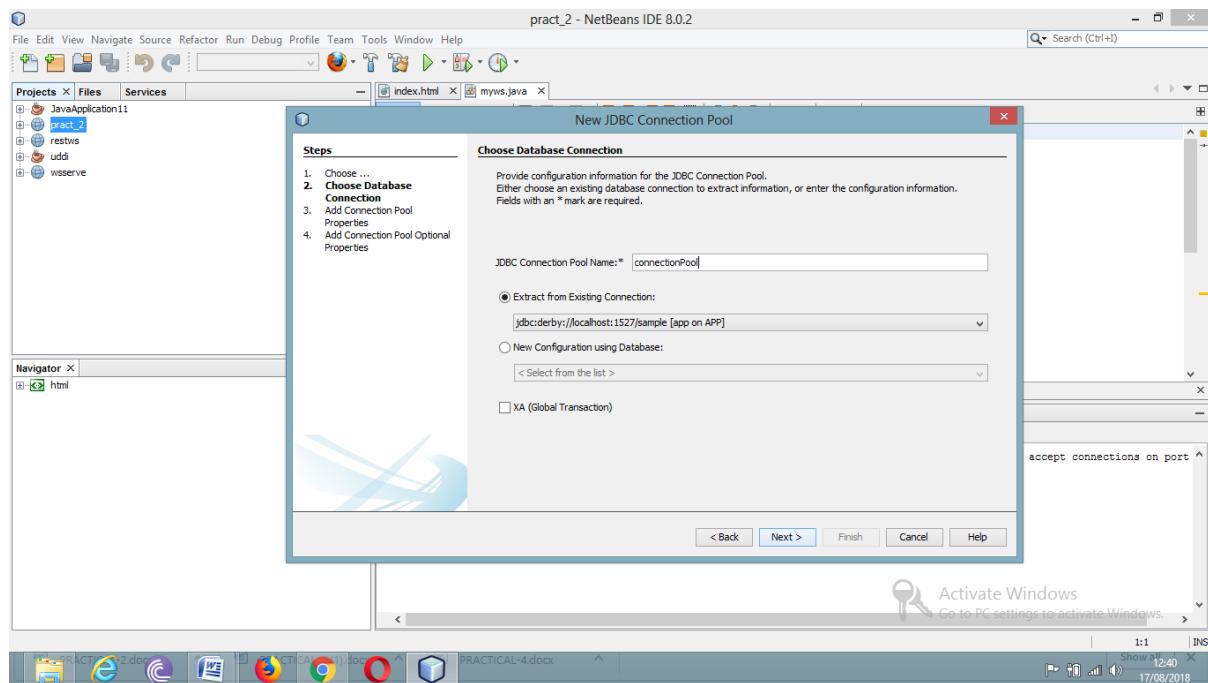
Right click Project name pract_2 select new select others in it select glassfish server

In that select jdbc connection pool

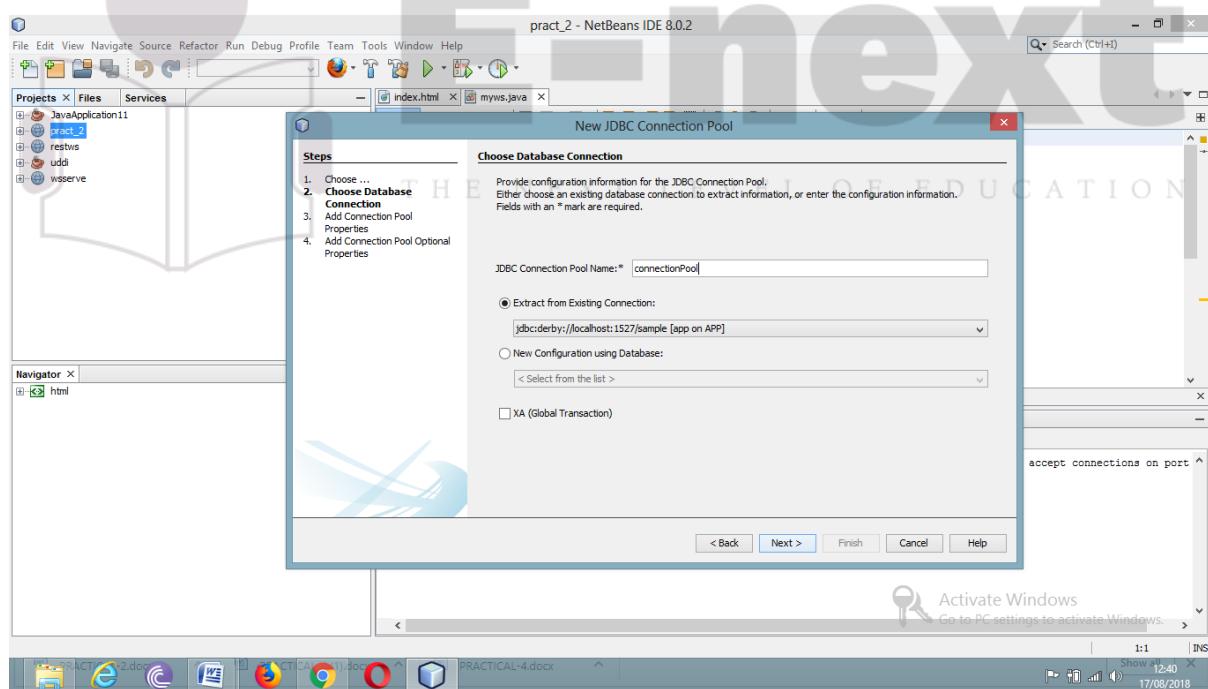


Mahesh Gurunani

Press next

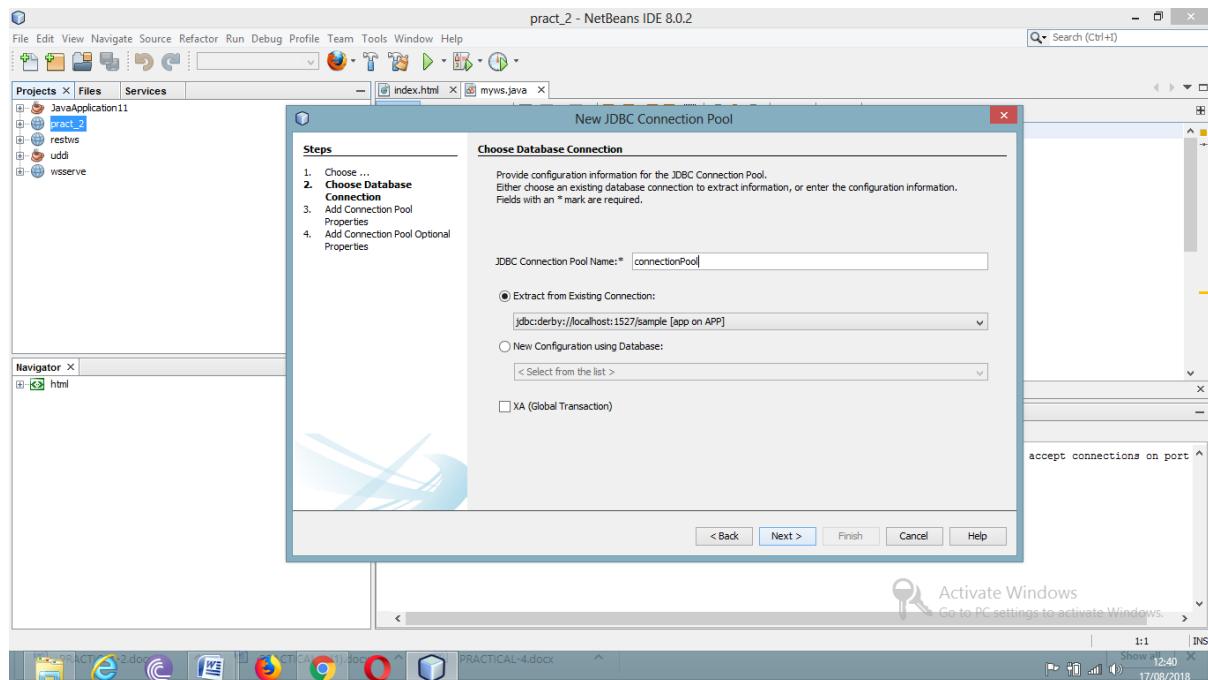


Press next

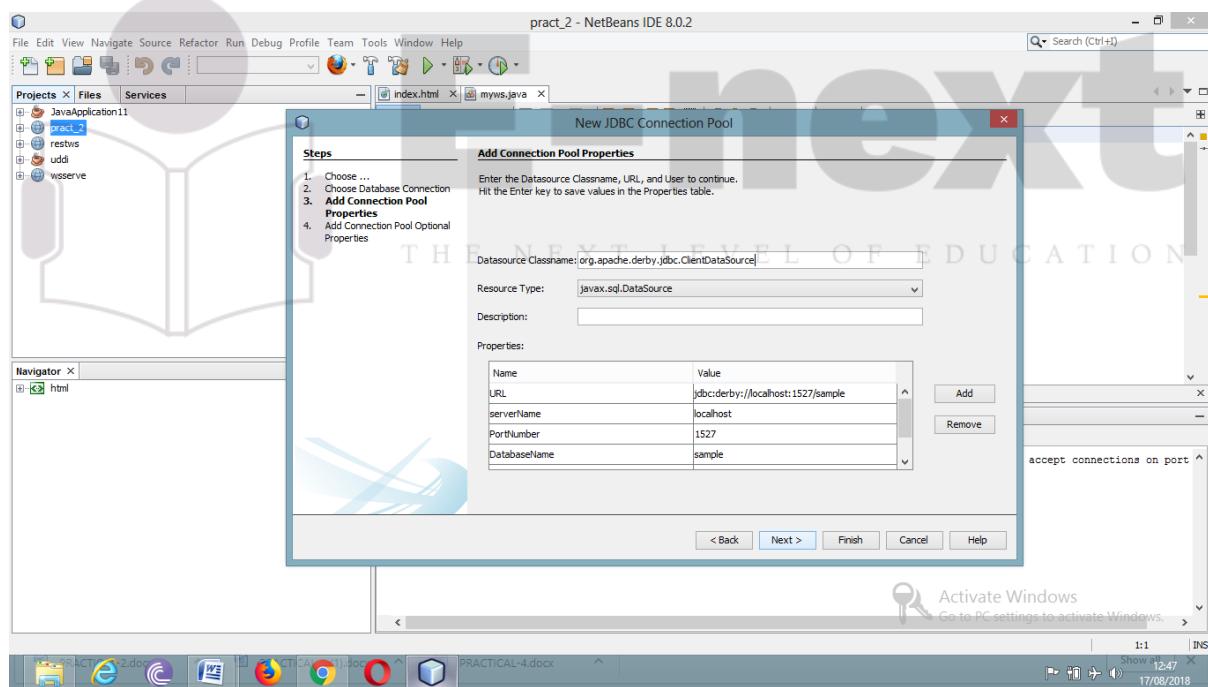


Press next

Mahesh Gurunani



Press next



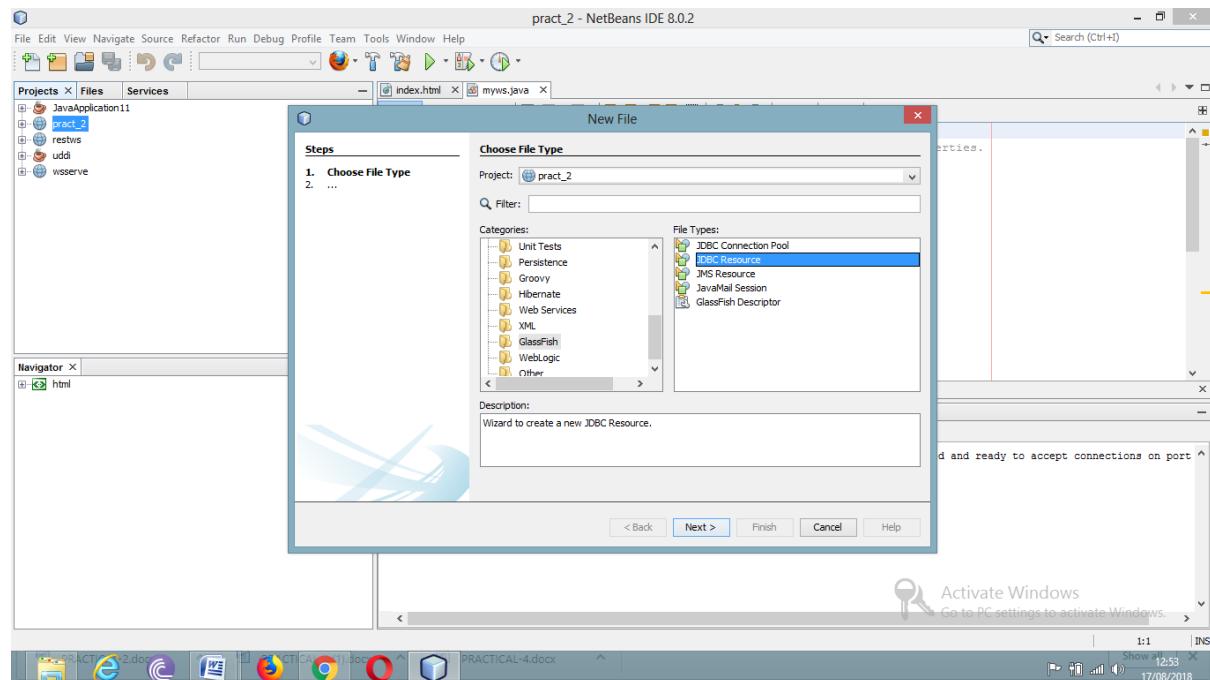
Press next

Press finish

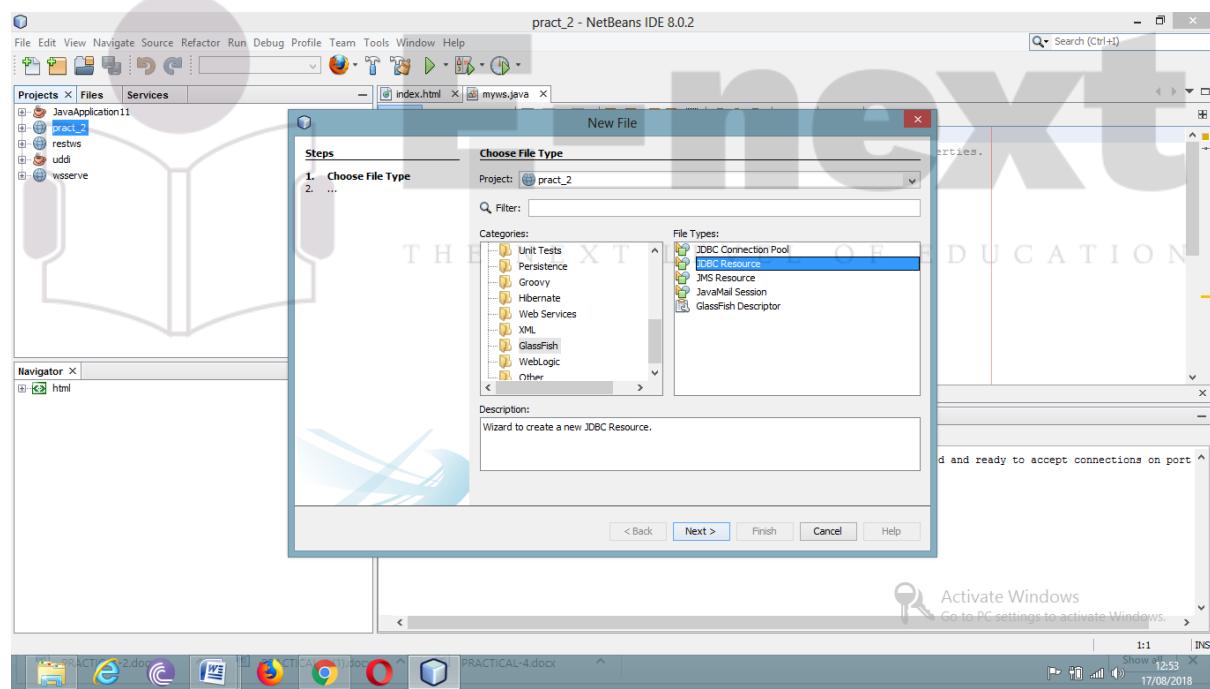
Right click project name select new select other

Select glassfish Select jdbc resource

Mahesh Gurunani

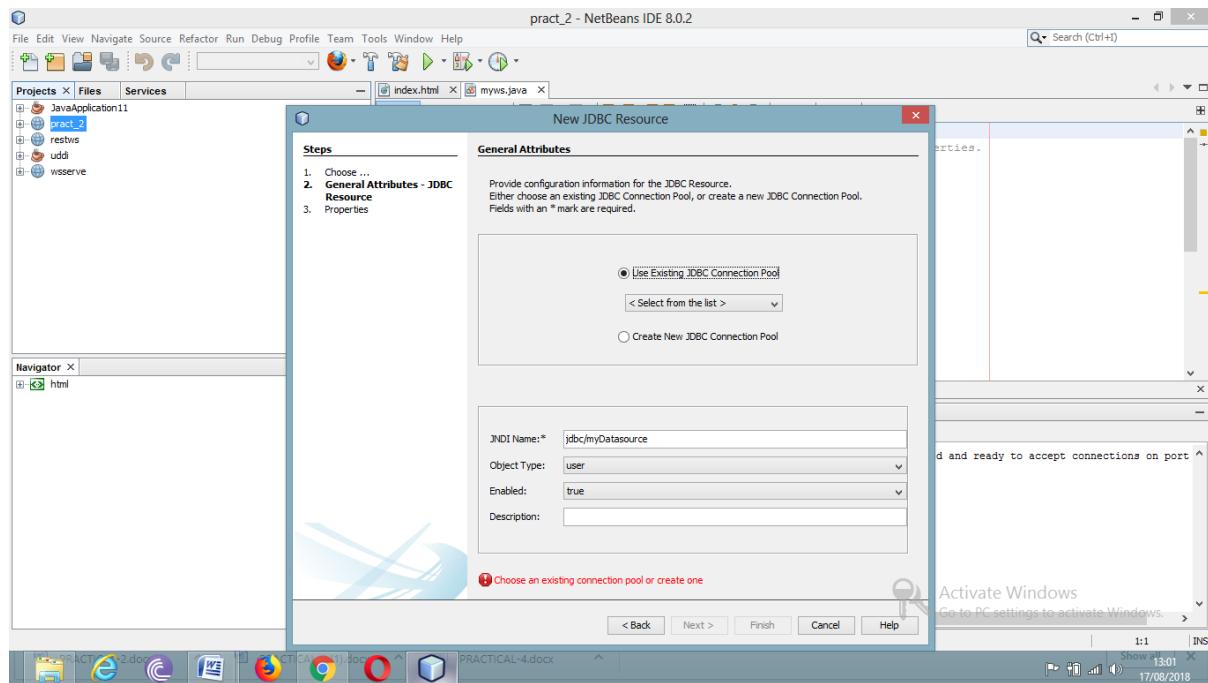


Press next



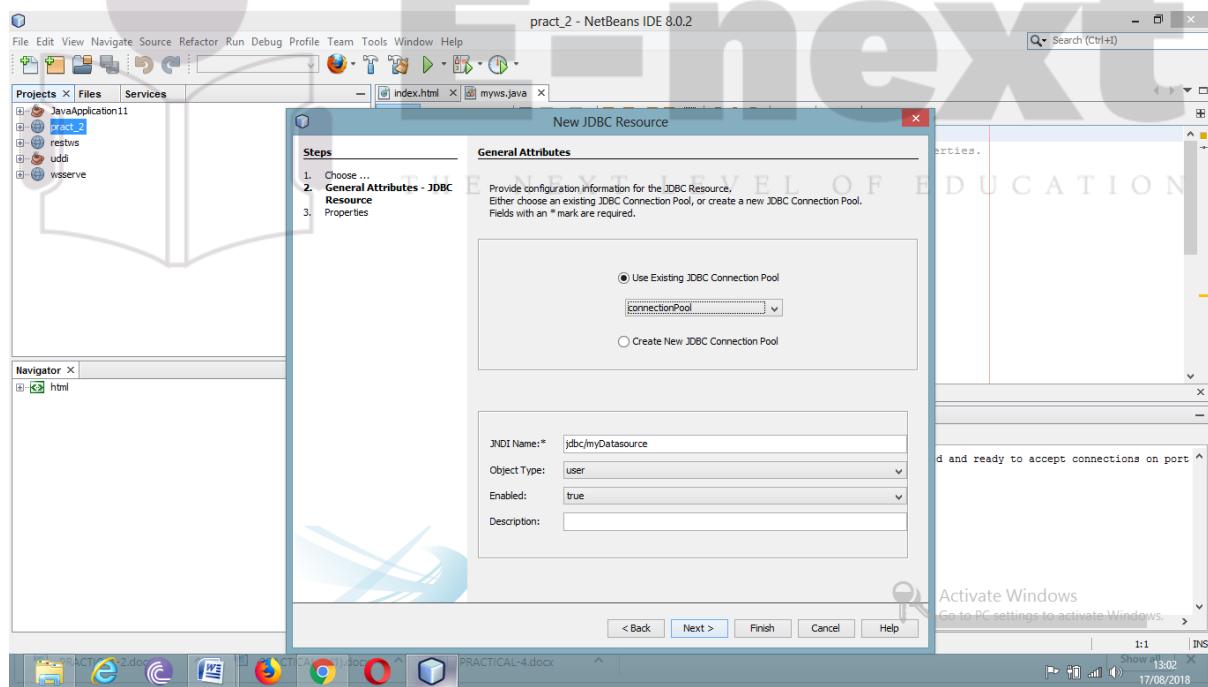
Press next

Mahesh Gurunani



Click on existing connection pool

Select from combobox connection pool



Change JNDI name to jdbc/mydb

Press next

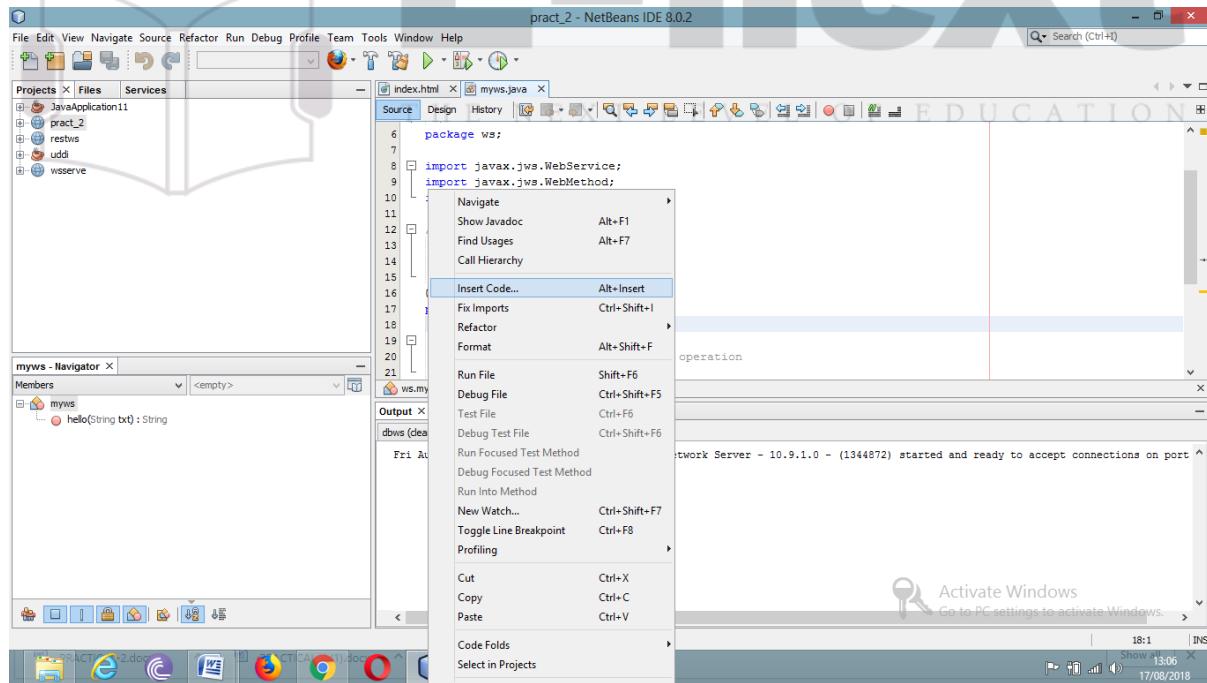
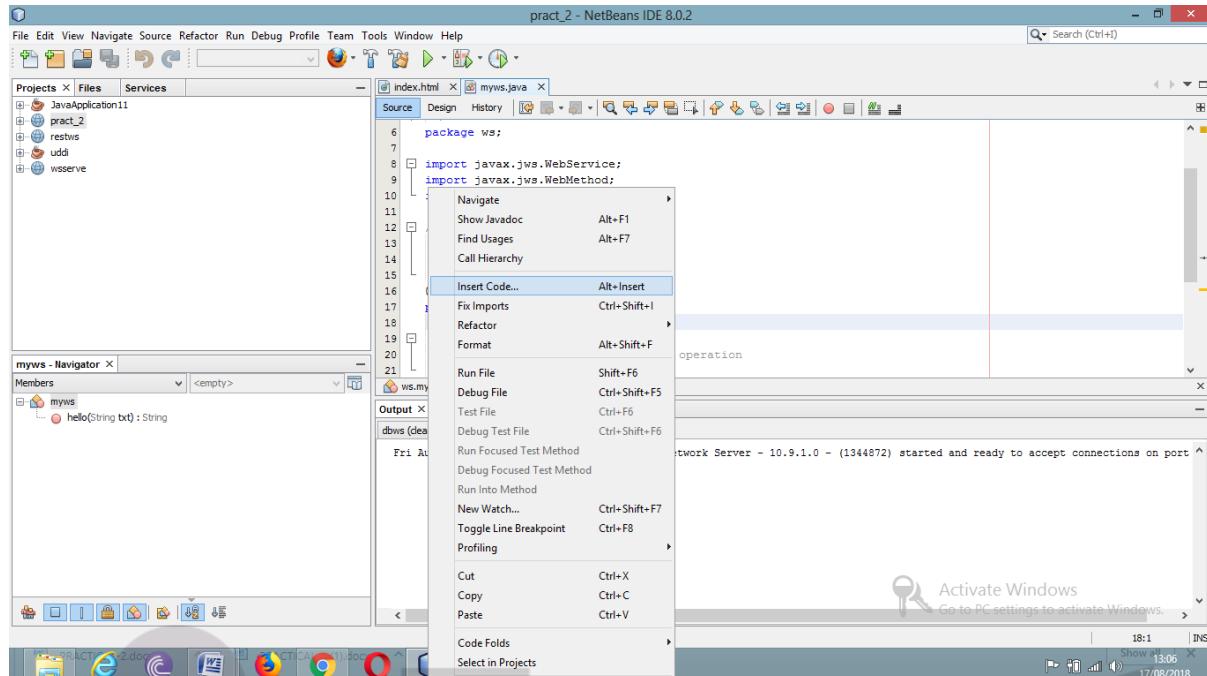
Press finish

It will open file myws.java

Mahesh Gurunani

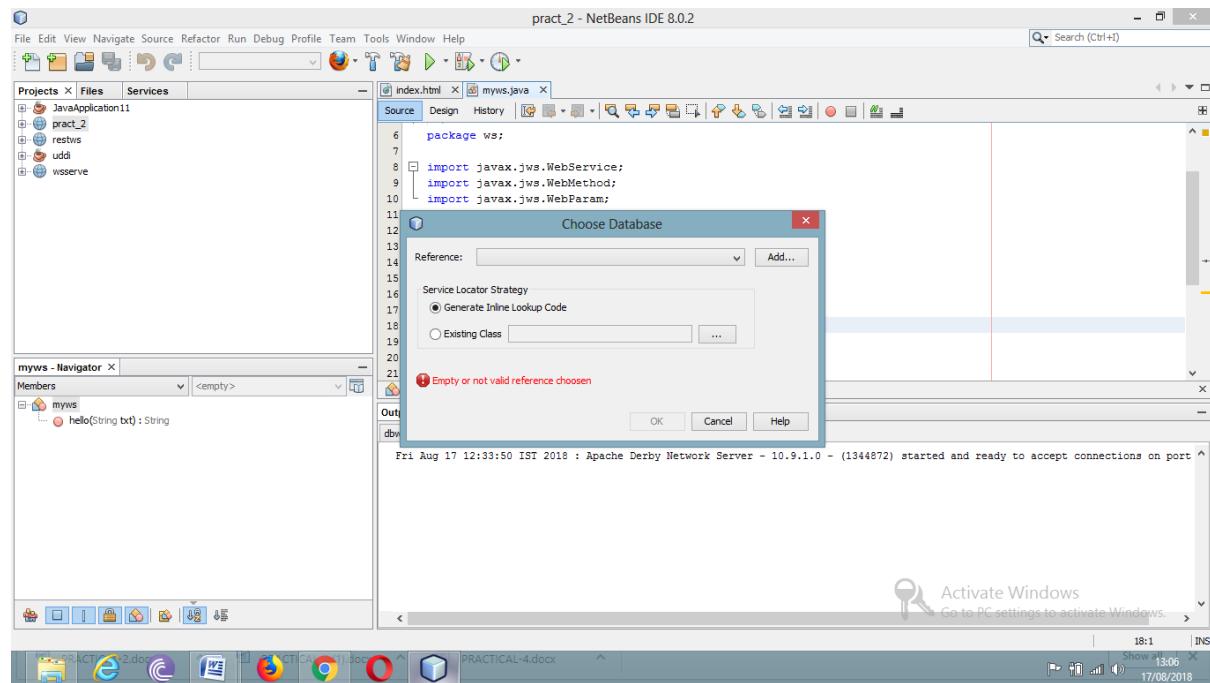
Right click on line number 18 (inside class myws)

Select insert code



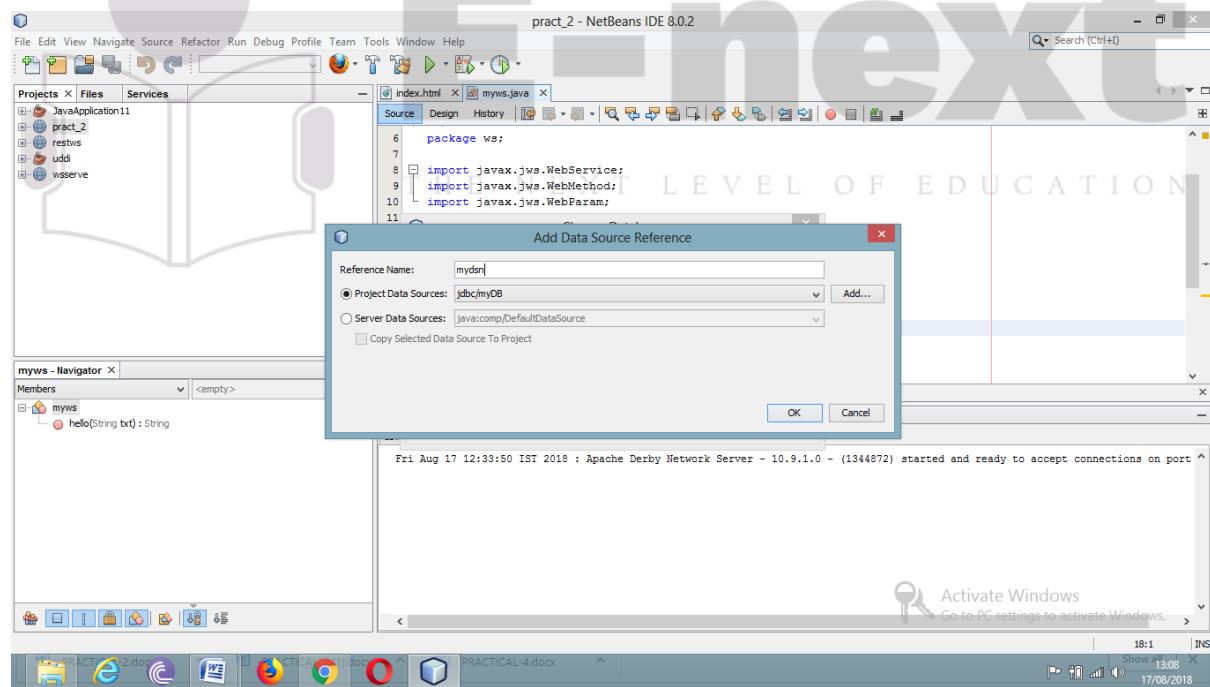
Select use database

Mahesh Gurunani



Click on add

Give any name I have given my dsn

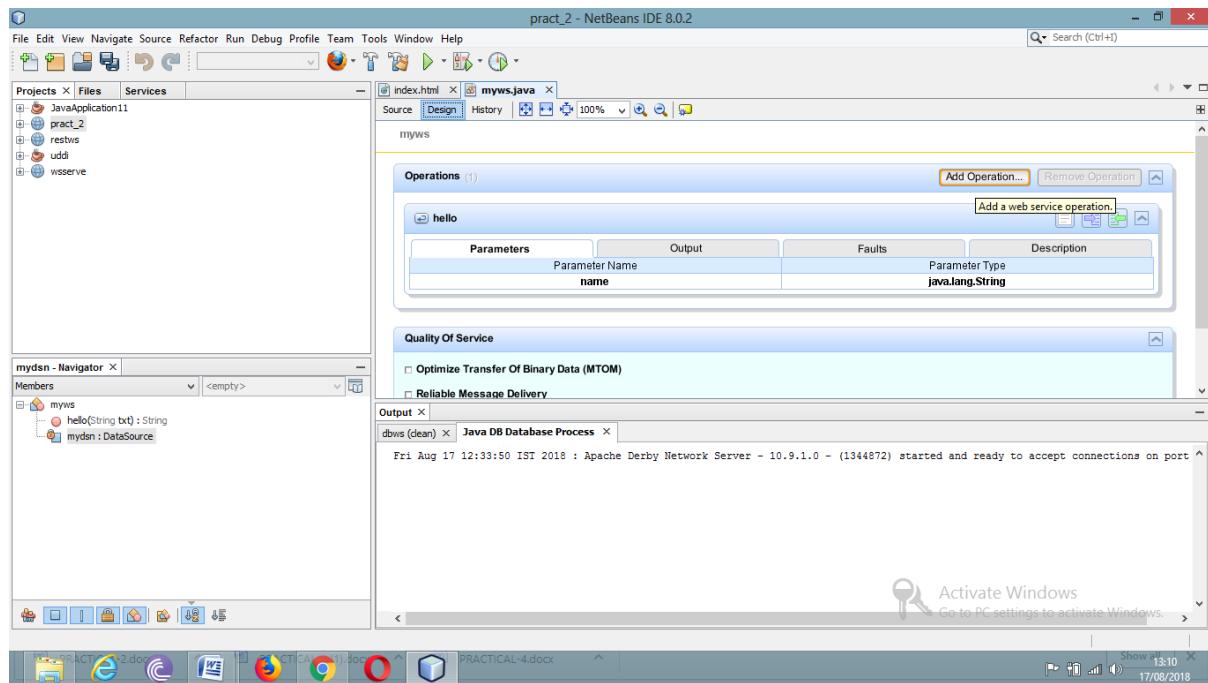


Press ok press ok

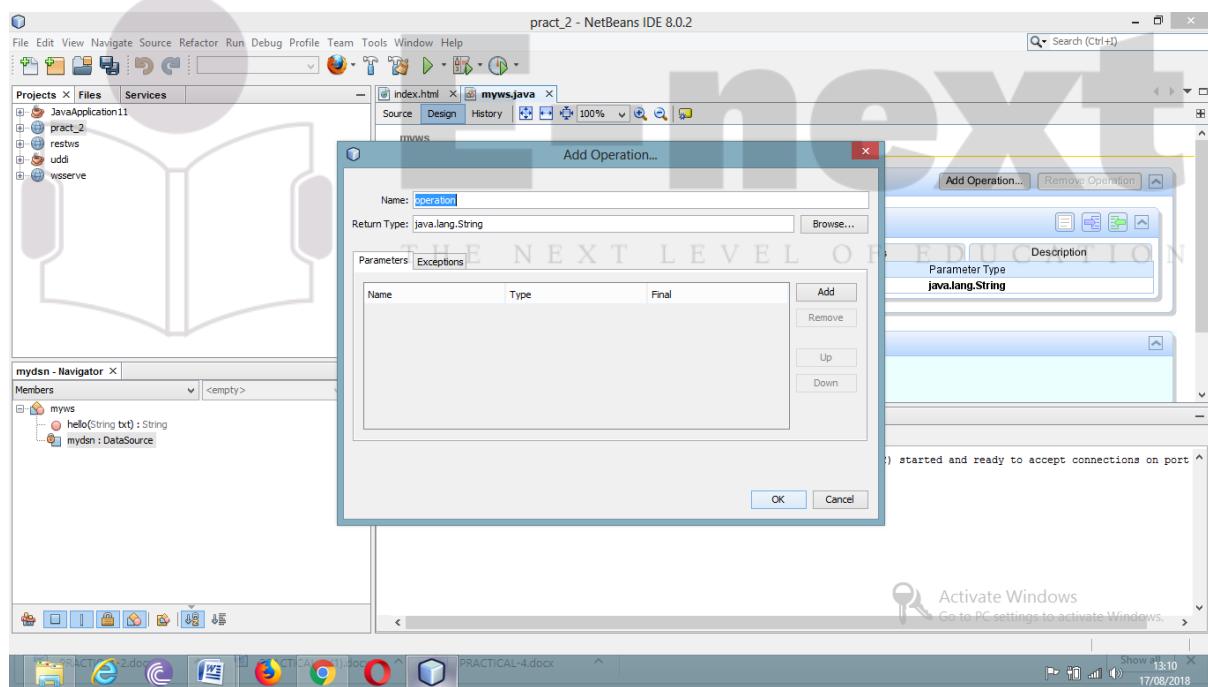
Now click on design

Select add operation

Mahesh Gurunani



It will open



We want a two way web service which takes rn from user and return name

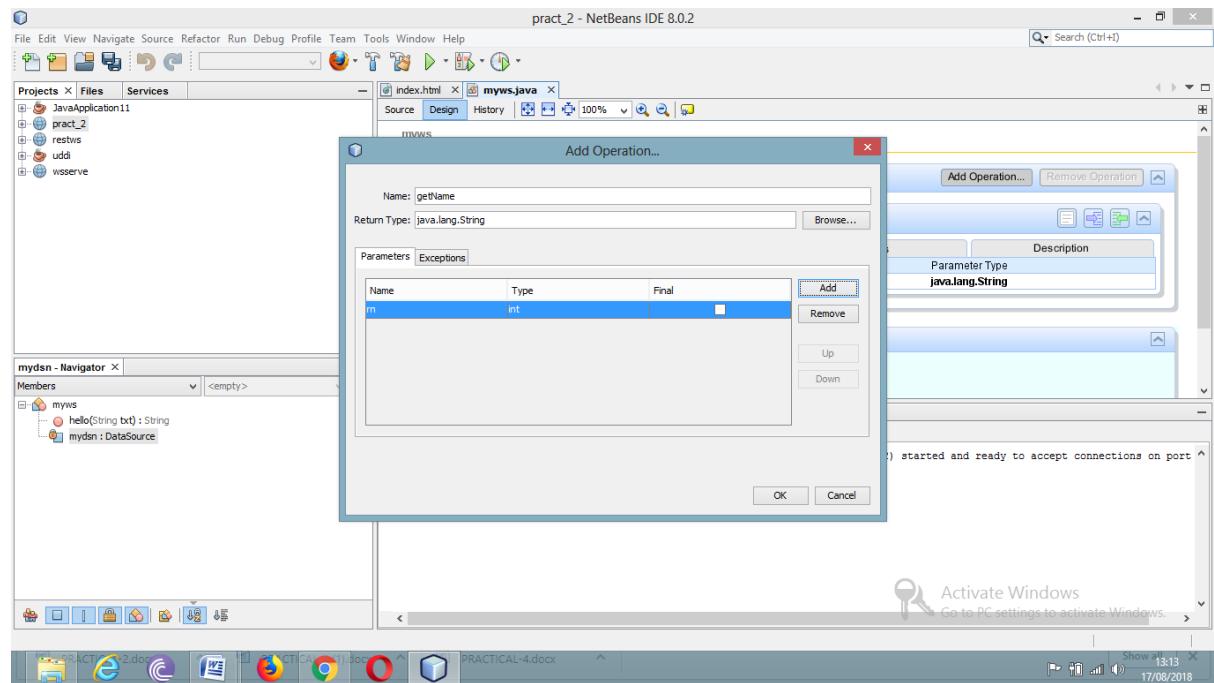
So give name as getName and return type is String

And click on add

And define parameter

My parameter name is rn and its type is int

Mahesh Gurunani



Press ok

Very similar way define one way function which insert roll no and name in database Tan

So again click on add operation give name as insert and return type as void

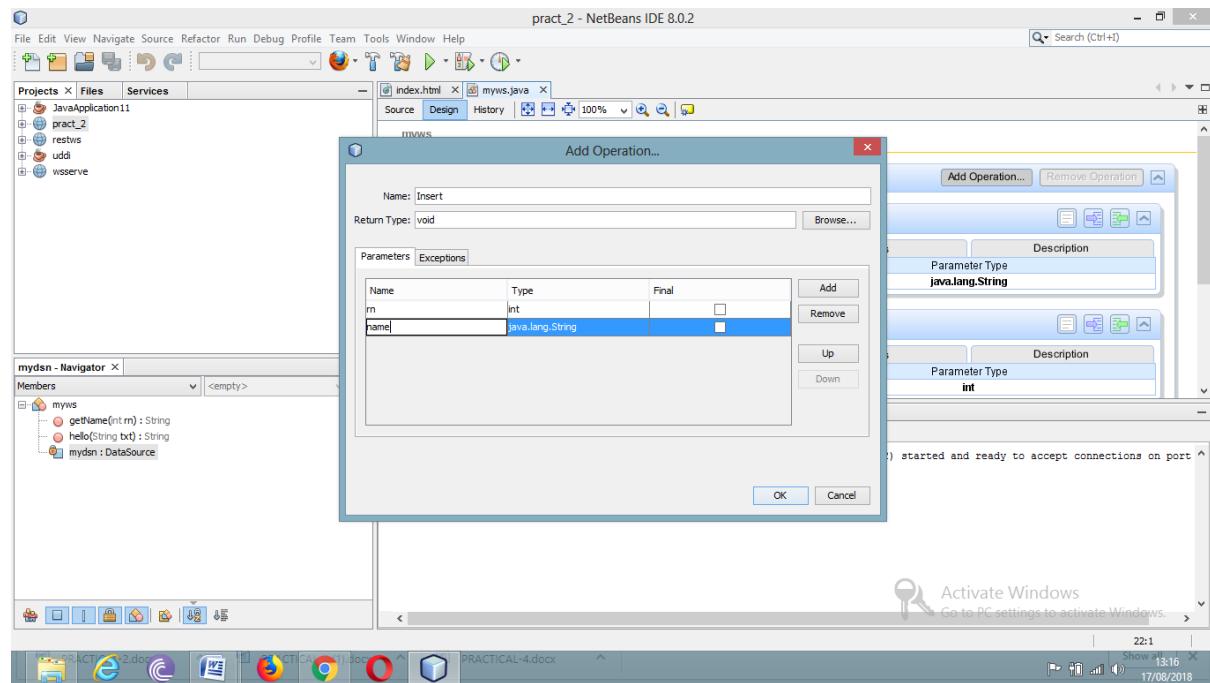
Click on add

Then give parameter as rn with type as int

Again press add

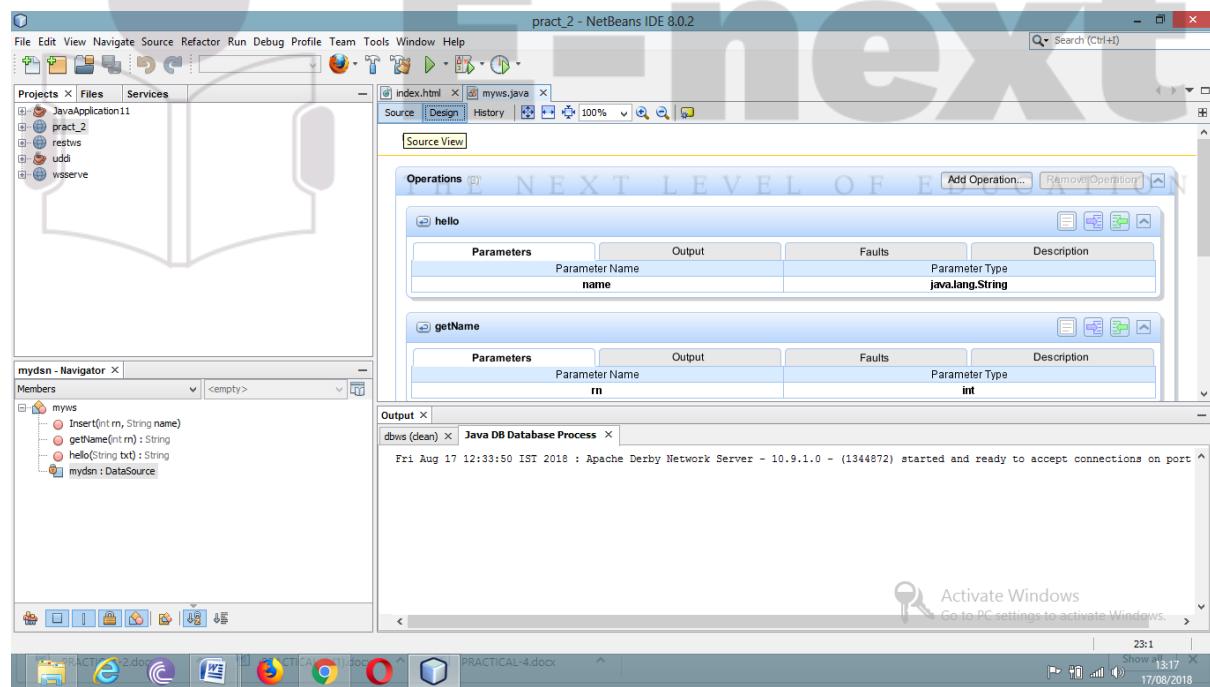
Give parameter as name and type as string

Mahesh Gurunani



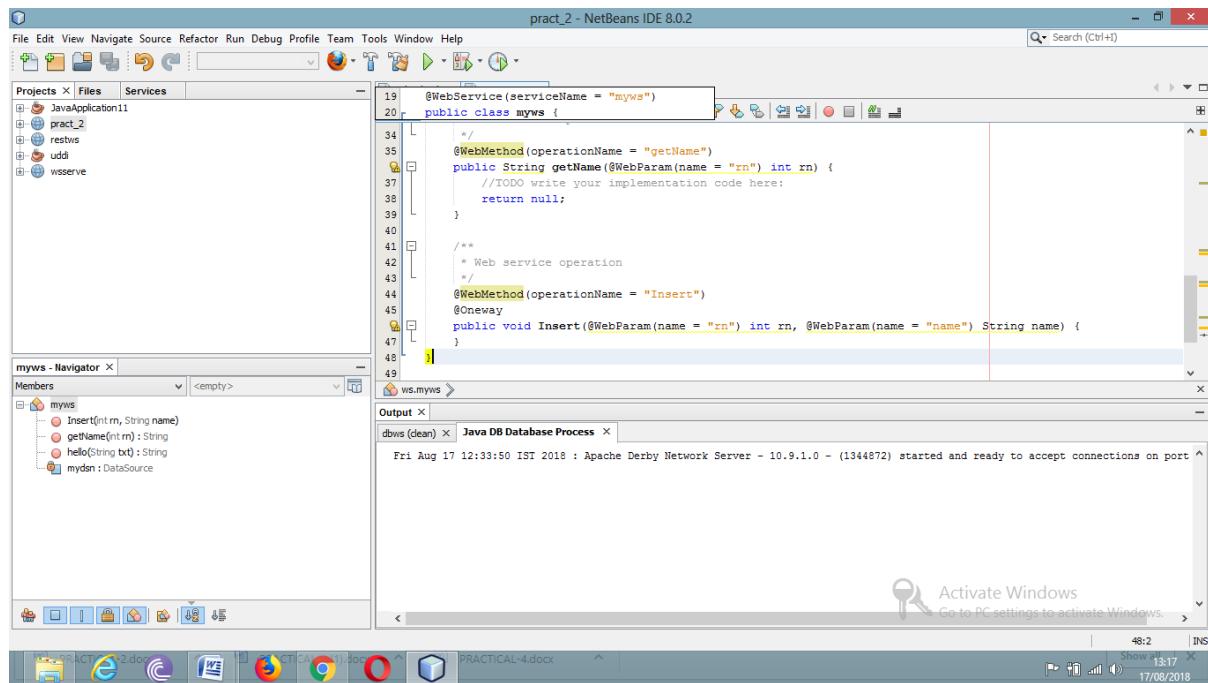
Press ok

Now click on source



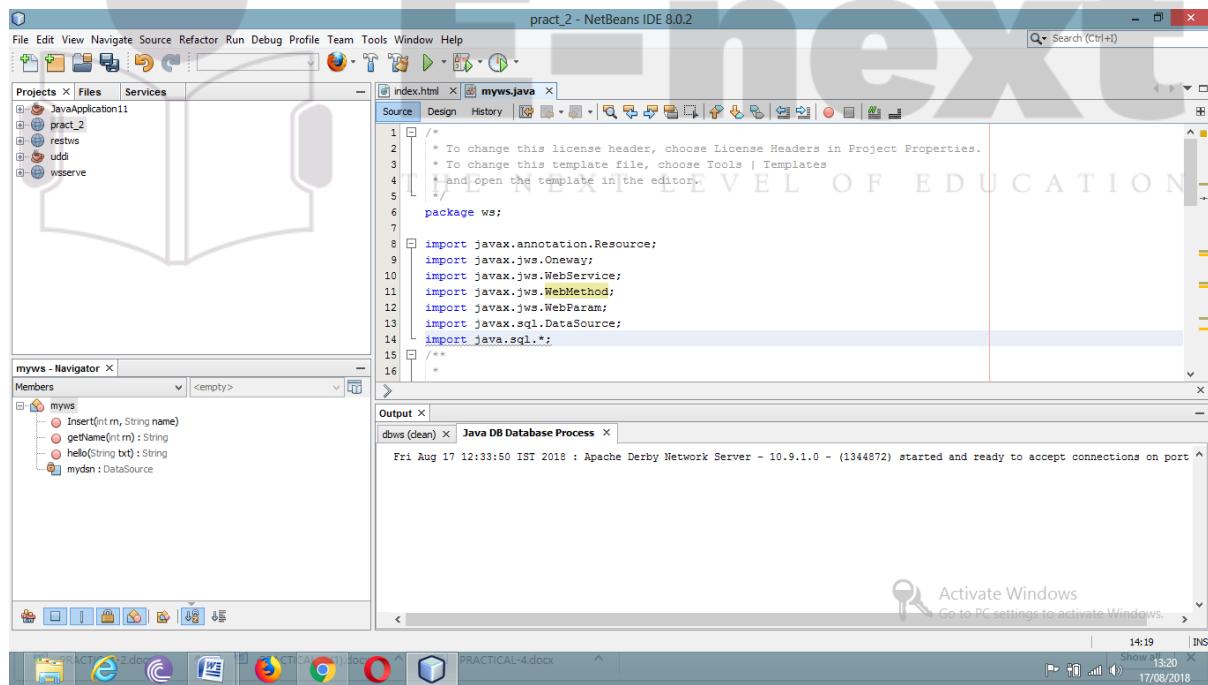
It will show you code (Functions you have created)

Mahesh Gurunani



As we are working with database so we need to import java.sql.*;

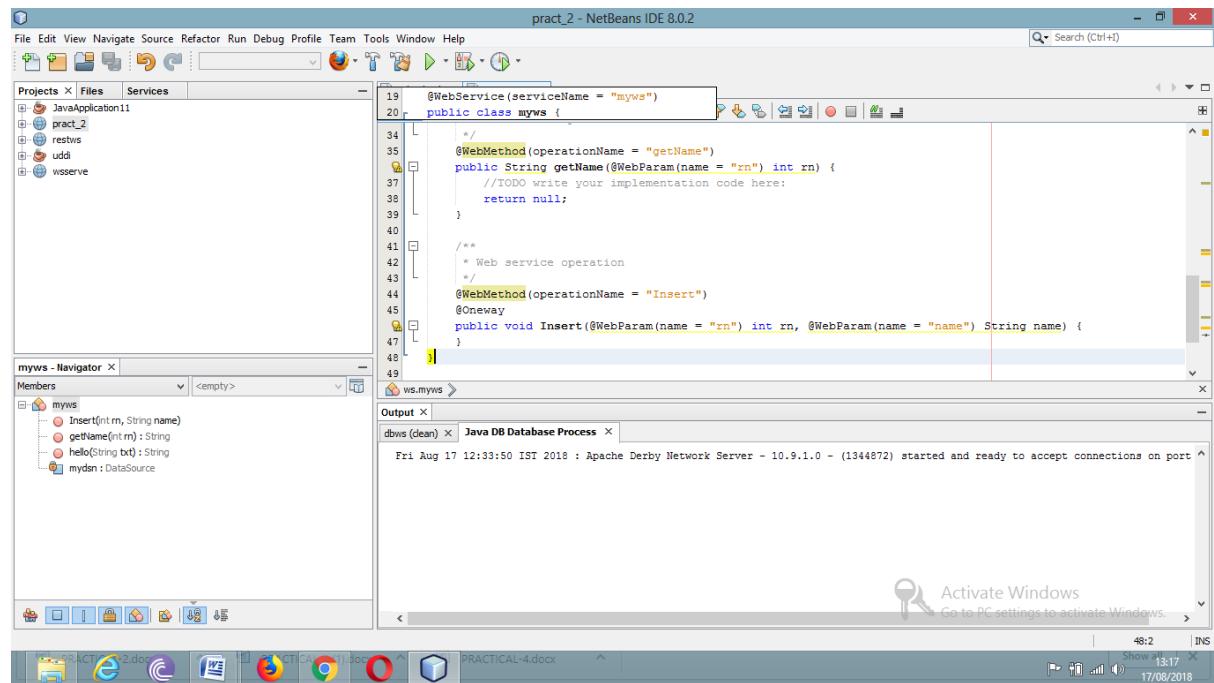
So go on the top of same file and add import java.sql.*; statement



Now come to function getName()

Go inside it and remove the line "todo code here"

Mahesh Gurunani



And type here

```
try{  
    Connection c=mydsn.getConnection();  
    PreparedStatement ps=c.prepareStatement("select * from tan where rn=?");  
    ps.setInt(1, rn);  
    ResultSet r=ps.executeQuery();  
  
    if(r.next())  
        return r.getString(2);  
  
    else  
        return "No name found";  
  
}  
catch(Exception e)  
{return "error";}
```

Now save it

Very similar way write the code for Insert Function (One way)

Mahesh Gurunani

Right Click on Project name pract_2

Select deploy

Once Deployed our web service is ready

Now We will Create Client (Servlet)

So create new Project (web java)

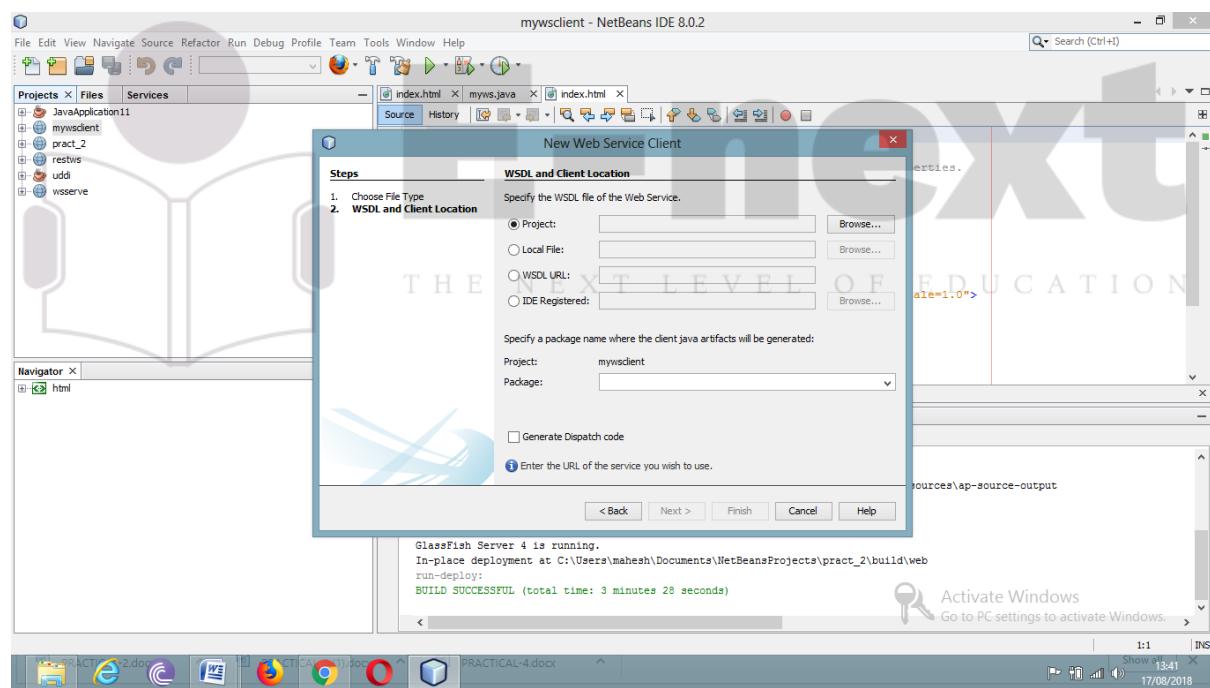
Select new project select new select web java

Give it name mywsclient Press next Press finish

Now right click on project name mywsclient

select new

select web service client



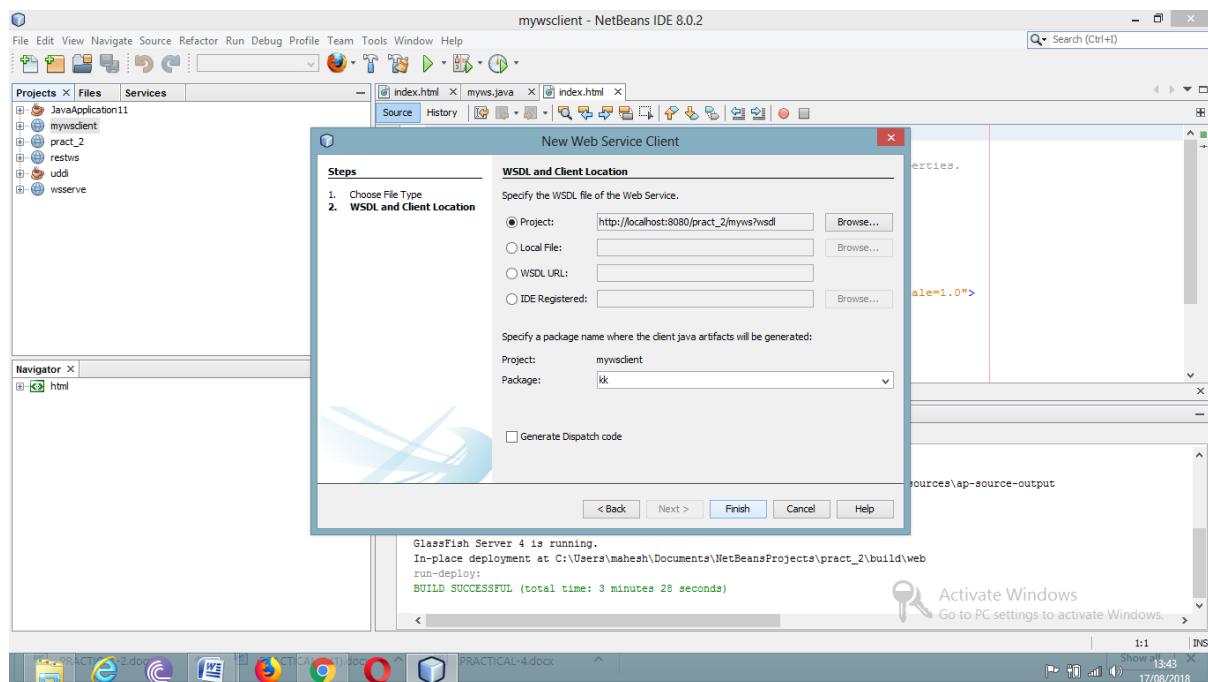
click on Project then select brows

expand pract_2

select myws and press ok

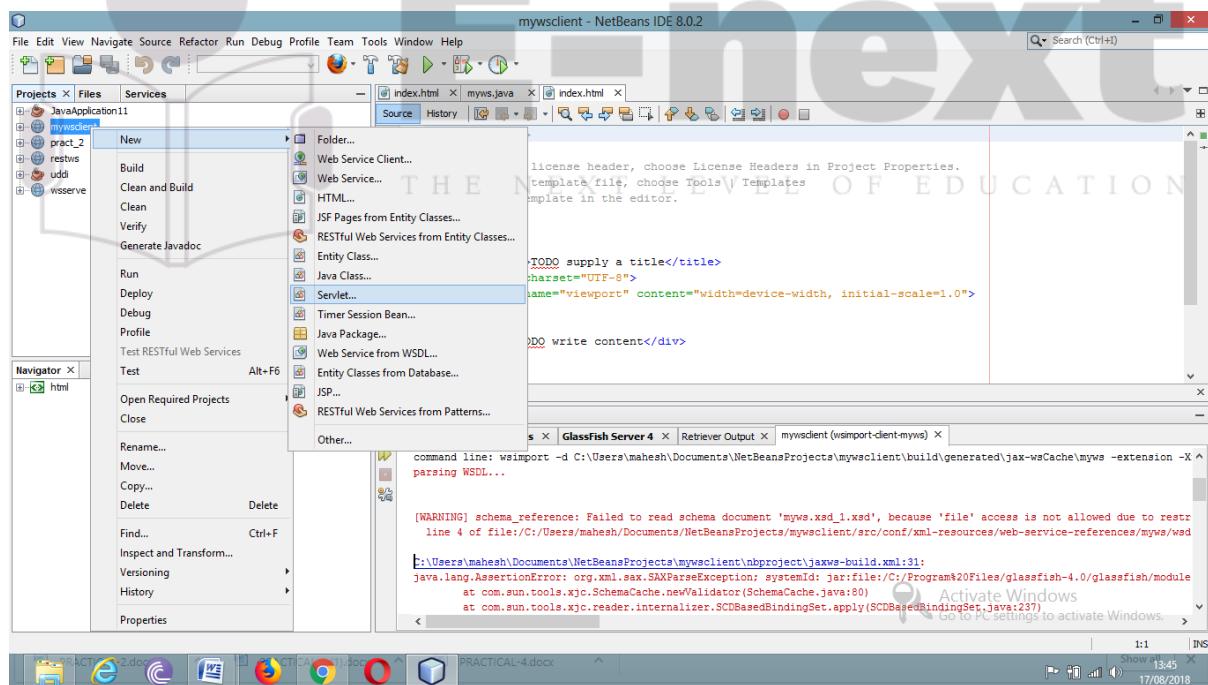
Give package name kk

Mahesh Gurunani



Press finish

Now right click on mywsclient project select new select servlet



Give servlet name as myservlet

Press finish

Now remove lines from number 34 to 42 of Process request function of myservlet

Mahesh Gurunani

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet myservlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Servlet myservlet at " + request.getContextPath() + "</body>");
        out.println("</html>");
    }
}
```

Output:

```
[WARNING] schema_reference: Failed to read schema document 'myws.xsd_1.xsd', because 'file' access is not allowed due to reattr line 4 of file:/C:/Users/mahe.../Documents/NetBeansProjects/mywsclient/src/conf/xml-resources/web-service-references/myws.wsdl  
E:\Users\mahe.../Documents\NetBeansProjects\mywsclient\nbproject\jaxws-build.xml:31:  
java.lang.AssertionError: org.xml.sax.SAXParseException: systemId: jar:file:/C:/Program$20Files/glassfish-4.0/glassfish/module  
at com.sun.tools.xjc.SchemaCache.newValidator(SchemaCache.java:80)  
at com.sun.tools.xjc.reader.internalizer.SCDBasedBindingSet.apply(SCDBasedBindingSet.java:237)
```

Now the lines which we have removed i.e. from 34 to 42 ,type there

```
out.println(getName(1));
```

Now on the project window

Expand web service reference folder

You will find there getName function

Simply drag it in side myservlet.

It will create code automatically.

Now deploy mywsclient and run mywsclient by right clicking on the code of mywsclient

PRACTICAL-4

Aim: Develop client which consumes web services developed in different platform.

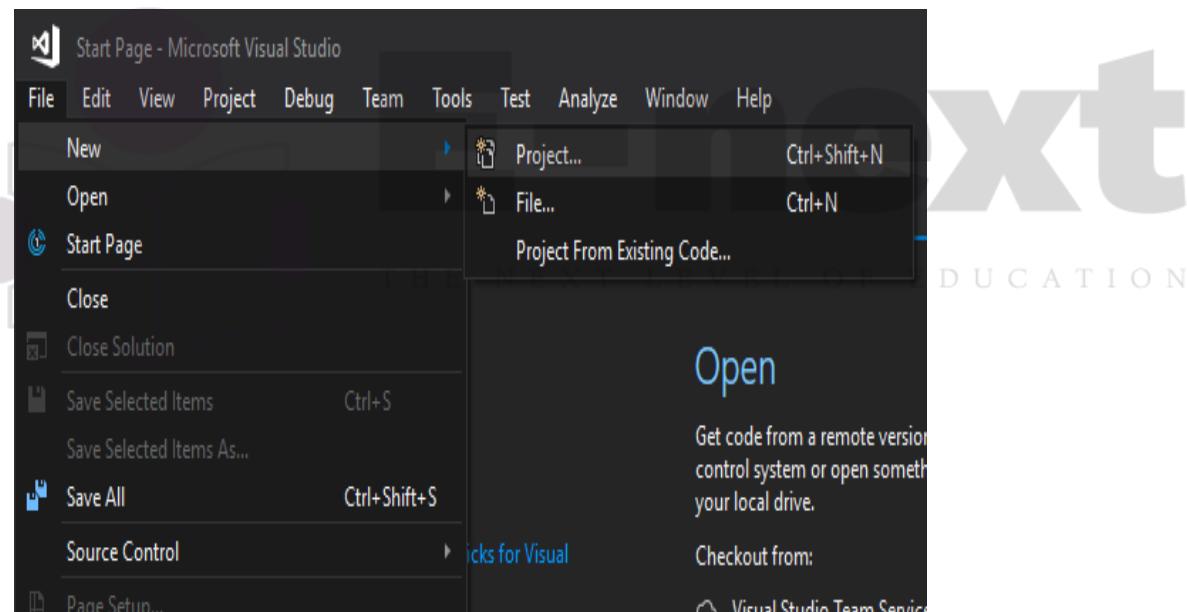
Requirement:

1. Visual Studio Community 2017
2. Version : 15.8 or latest

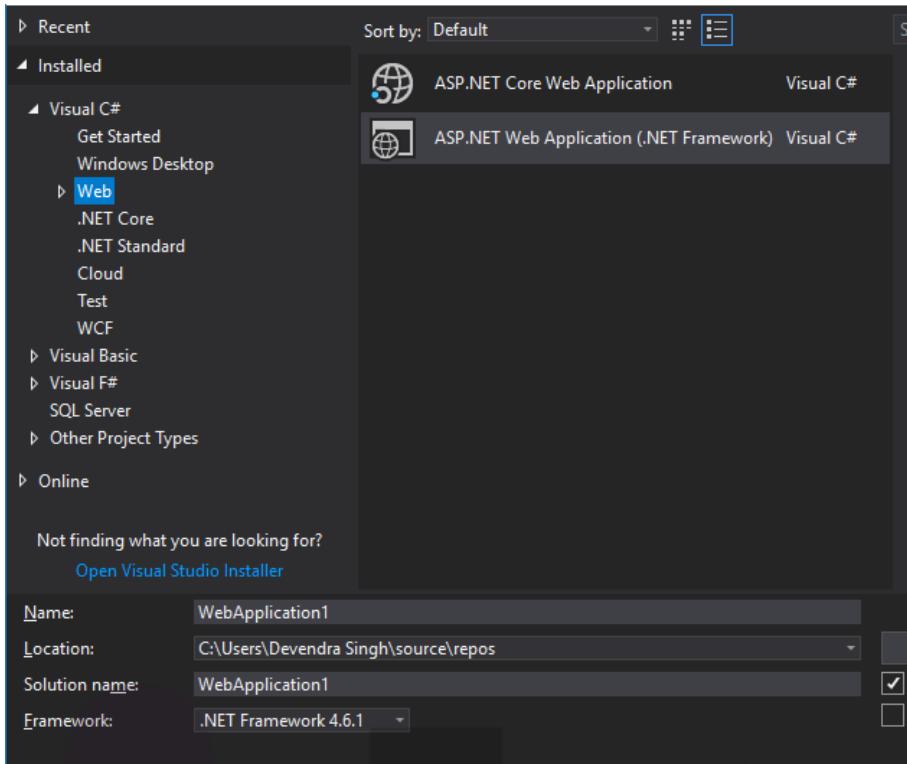
In this practical we are creating Web Service in Visual Studio ant then we will consume it in NetBeans.

1. Open Visual Studio IDE and click on File.

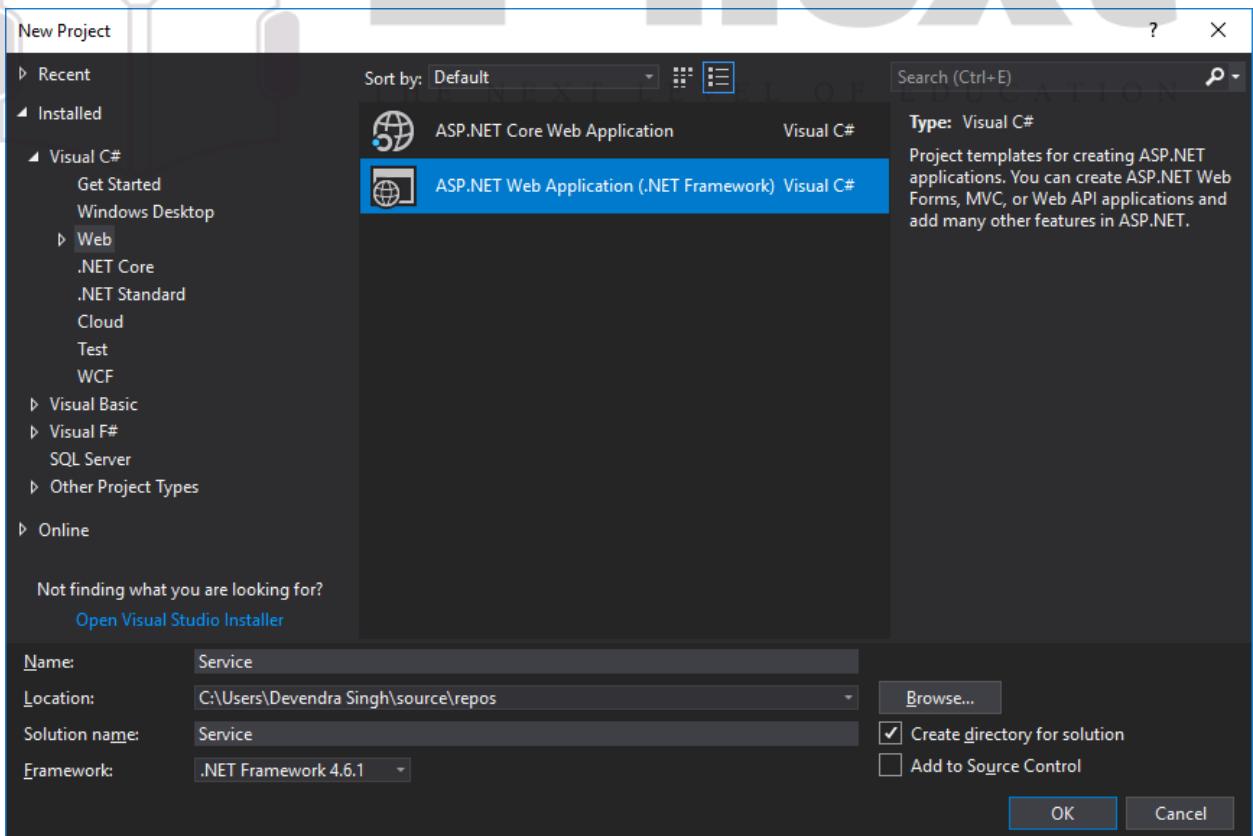
File -> New -> Project



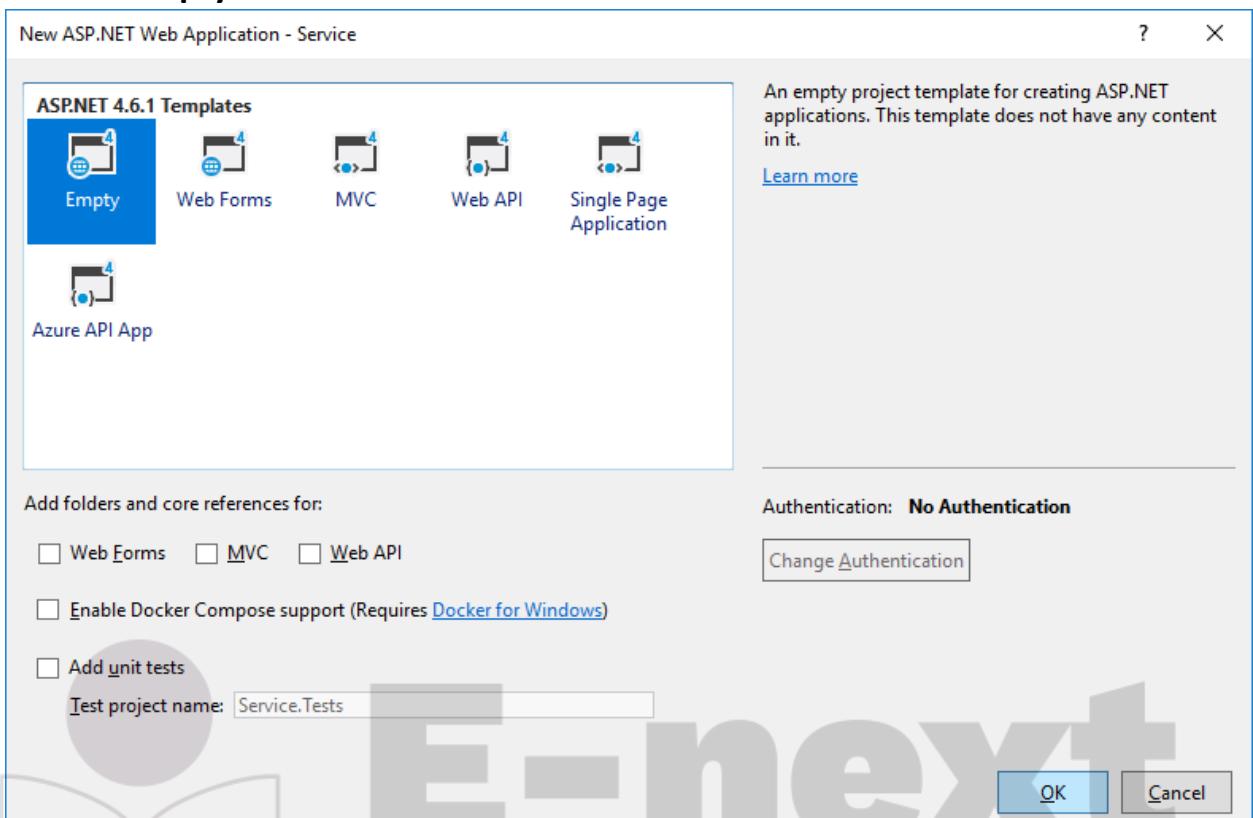
2. Click on Web.



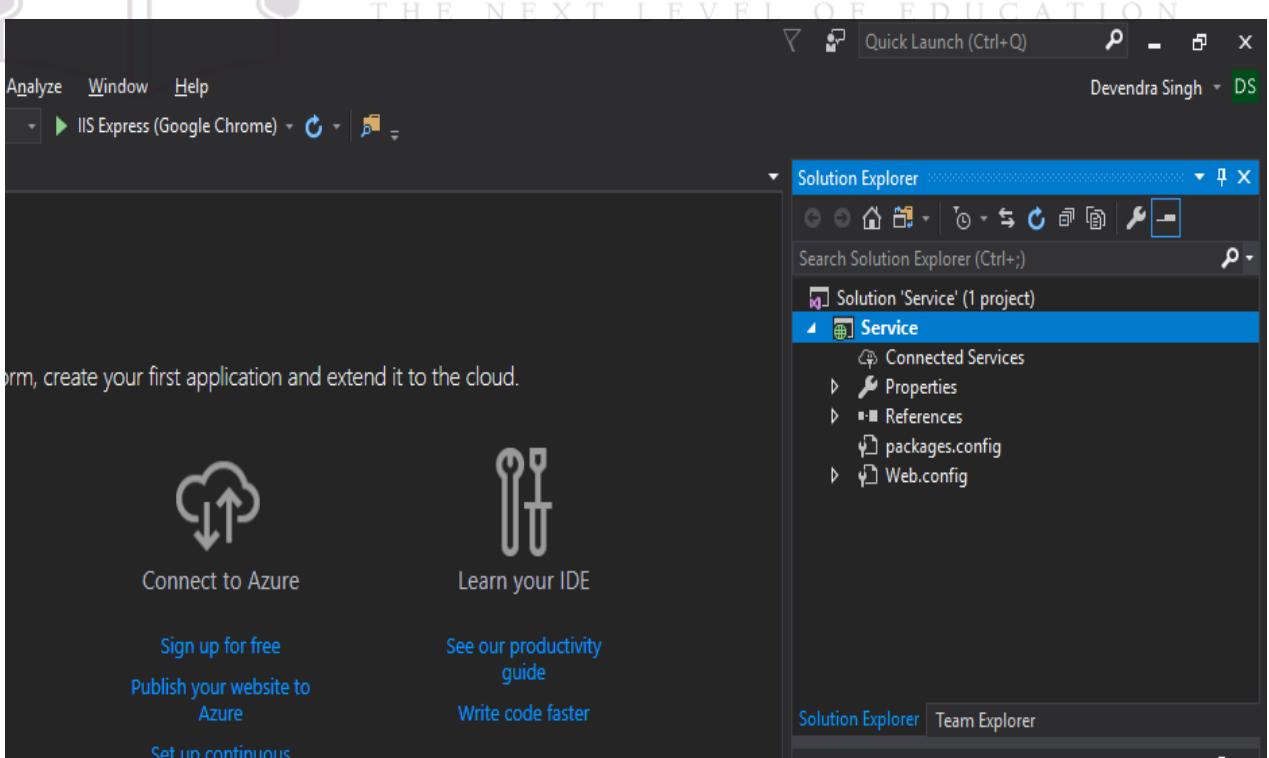
3. Select ASP.NET Web Application and give Name as Service. After that click on OK button.



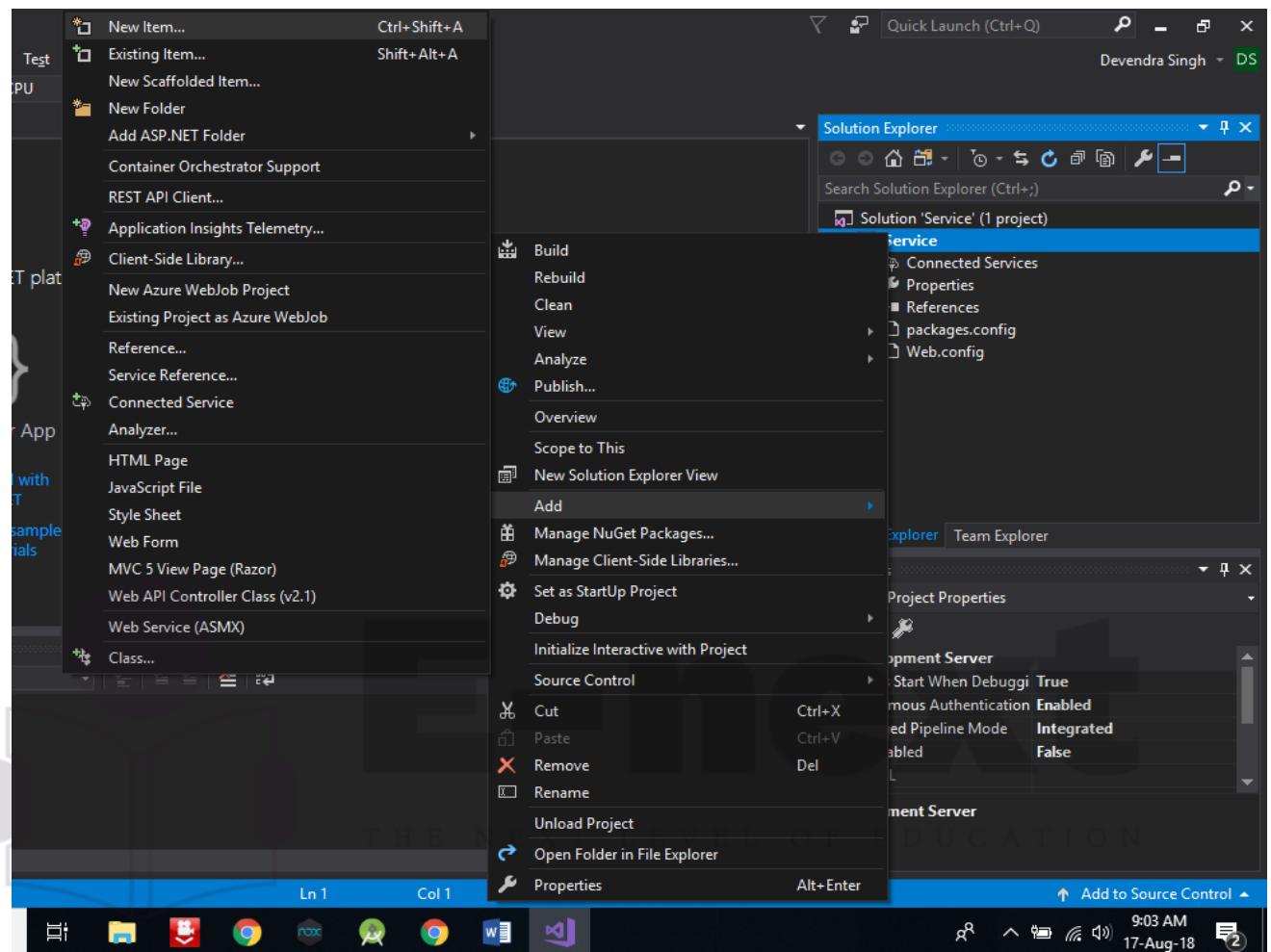
4. Select Empty and click on OK button.



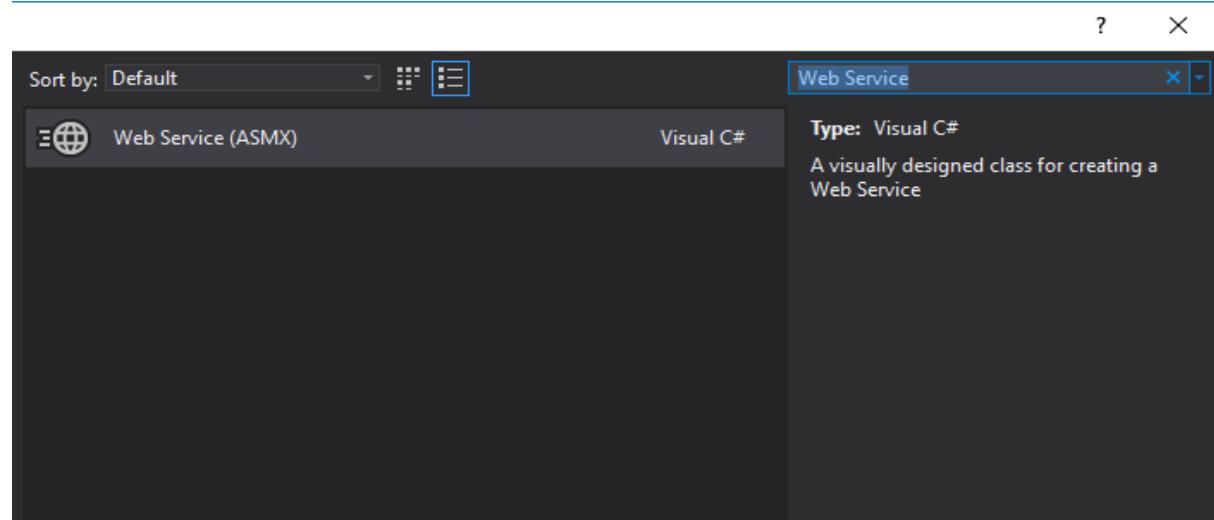
5. Now you can see, on right side in Solution Explorer Service project is created.



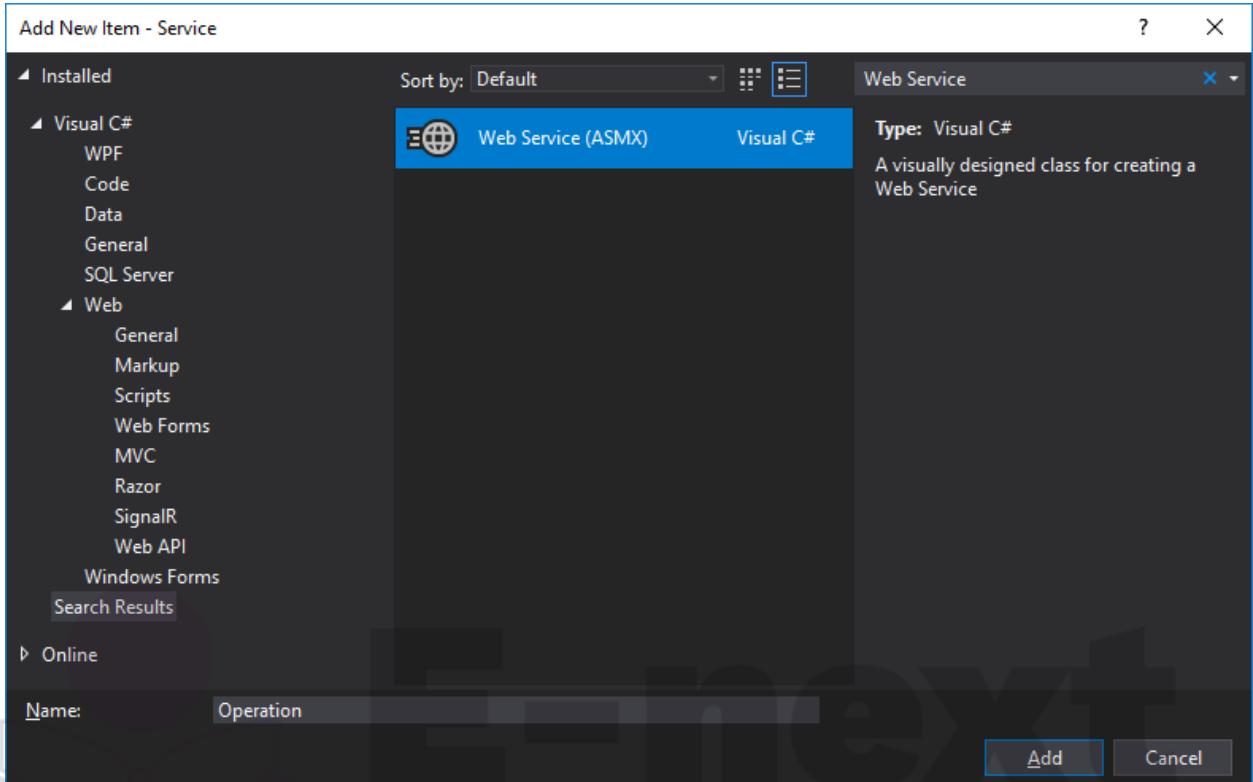
6. Right click on Service -> Add -> New Item.



7. Search for Web Service.



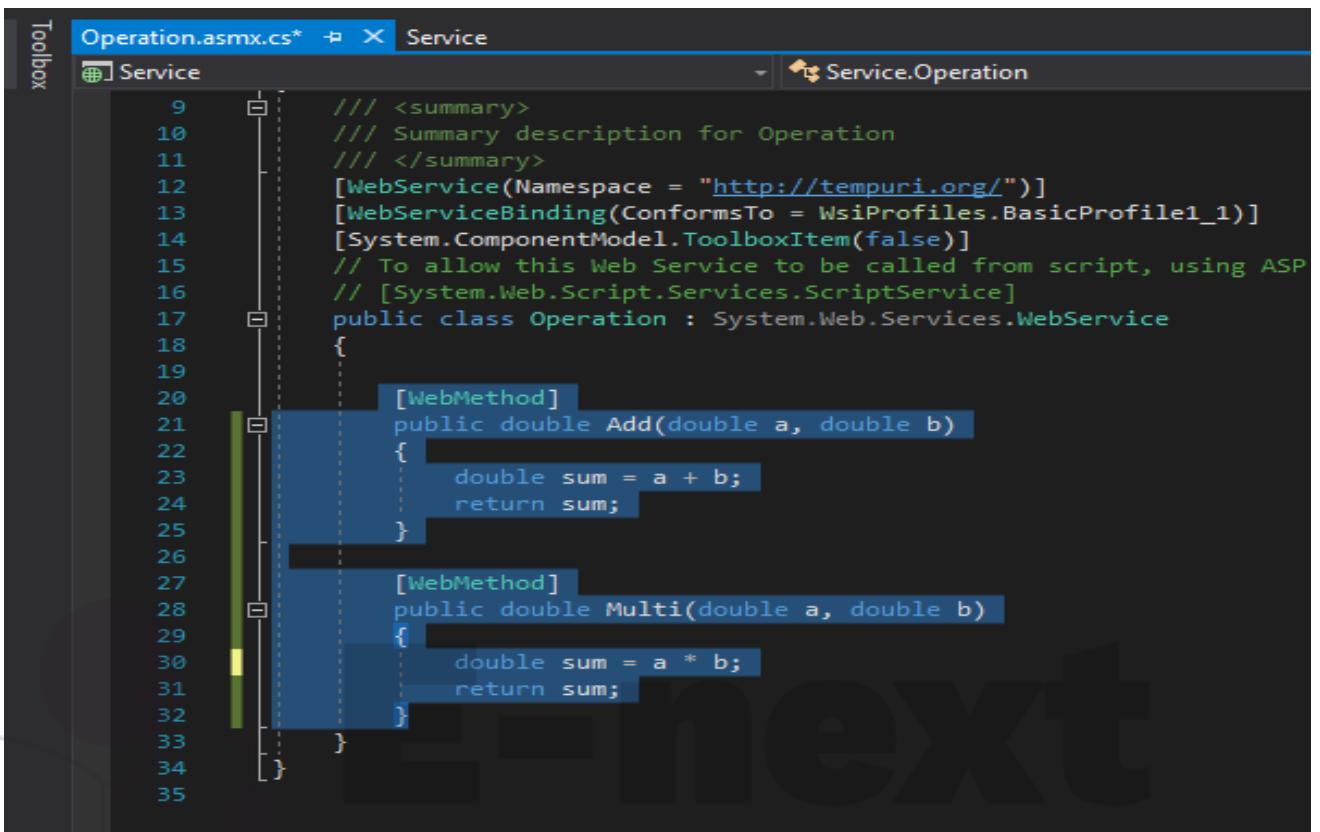
8. Select Web Service and give Name as Operation. After then click on Add button.



9. After click on Add button, Operation.asmx.cs file will be automatically open otherwise open it from Solution Explorer and Add the following code into Class Operation.

```
[WebMethod]
public double Add(double a, double b)
{
    double sum = a + b;
    return sum;
}
```

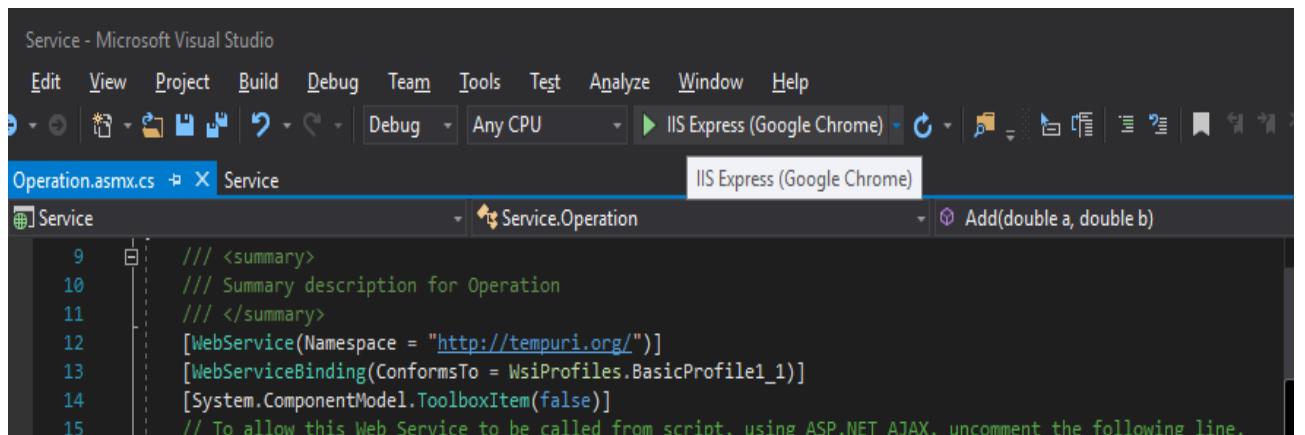
```
[WebMethod]
public double Multi(double a, double b)
{
    double sum = a * b;
    return s
}
```



```
Operation.asmx.cs*  X  Service
Service
9     /// <summary>
10    /// Summary description for Operation
11    /// </summary>
12    [WebService(Namespace = "http://tempuri.org/")]
13    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
14    [System.ComponentModel.ToolboxItem(false)]
15    // To allow this Web Service to be called from script, using ASP
16    // [System.Web.Services.ScriptService]
17    public class Operation : System.Web.Services.WebService
18    {
19
20        [WebMethod]
21        public double Add(double a, double b)
22        {
23            double sum = a + b;
24            return sum;
25        }
26
27        [WebMethod]
28        public double Multi(double a, double b)
29        {
30            double sum = a * b;
31            return sum;
32        }
33    }
34
35
```

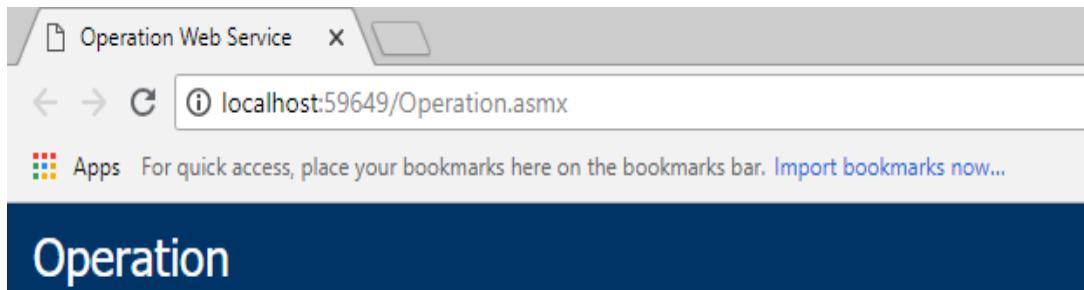
After that press **Ctrl+S** to save the methods. Actually we are creating two methods for Web Service. One is for addition of two numbers and second one is for multiplication of two numbers.

10. Now run the project by click on Green arrow button below the Window menu.



```
Service - Microsoft Visual Studio
Edit  View  Project  Build  Debug  Team  Tools  Test  Analyze  Window  Help
Operation.asmx.cs  X  Service
Service
9     /// <summary>
10    /// Summary description for Operation
11    /// </summary>
12    [WebService(Namespace = "http://tempuri.org/")]
13    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
14    [System.ComponentModel.ToolboxItem(false)]
15    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
```

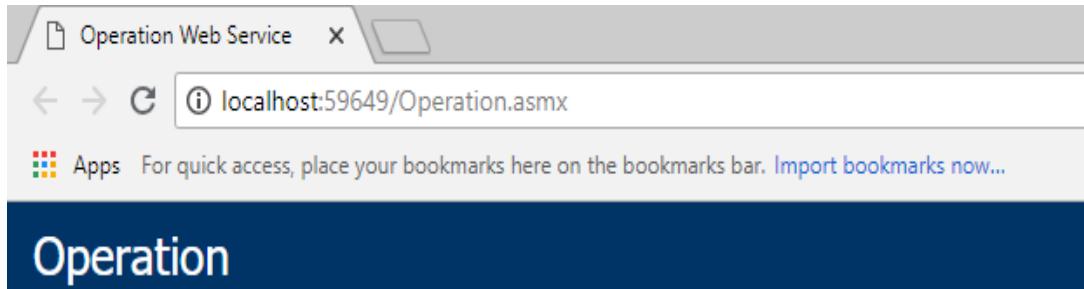
11. A window will open in browser and that is our web service.



12. You can check services by click on Add or Multi option. But we don't need this.

13. Now click on Service Description option.

THE NEXT LEVEL OF EDUCATION



The following operations are supported. For a formal definition, please review the [Service Description](#).

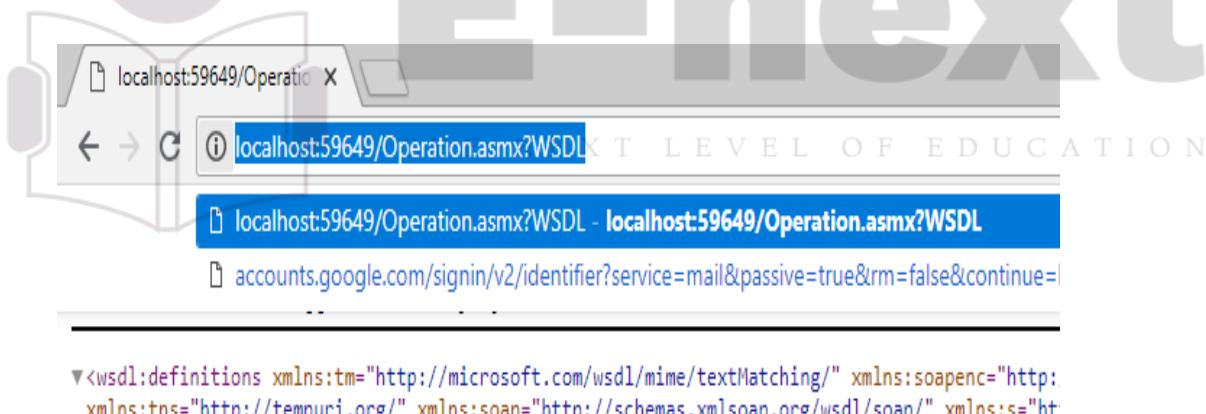
- [Add](#)
- [Multi](#)

This web service is using <http://tempuri.org/> as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other Web services. You should use a more permanent namespace.

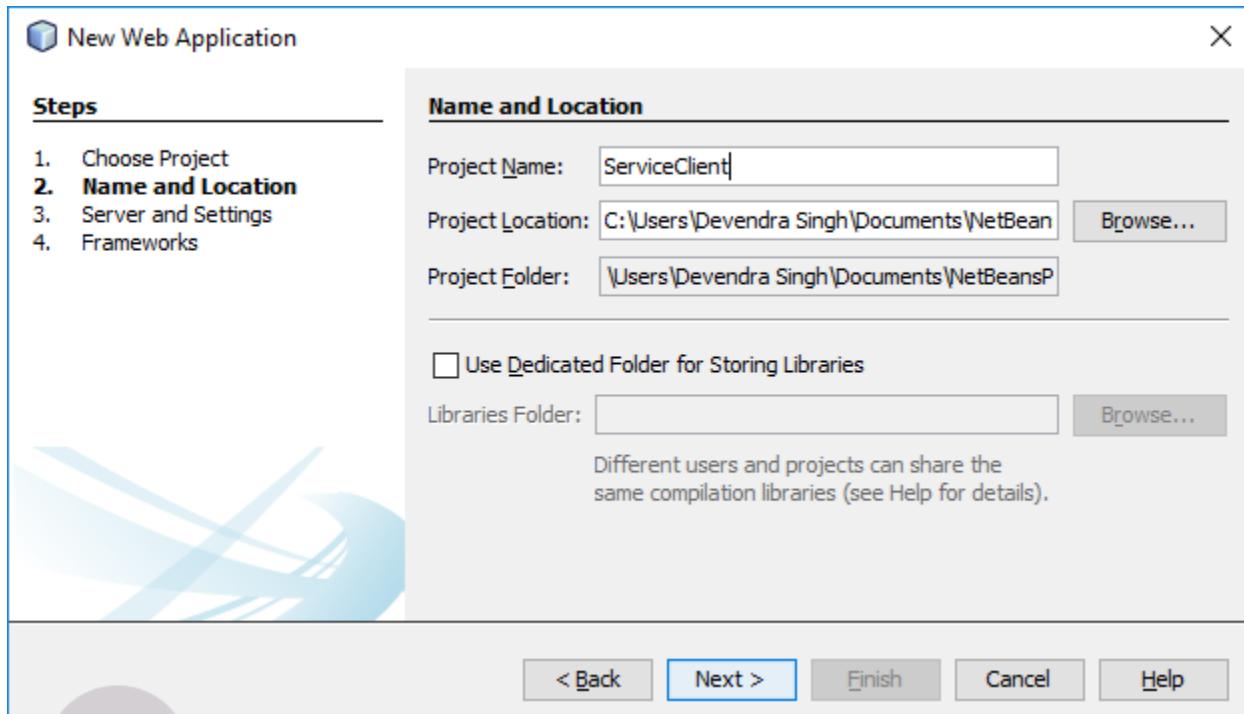
- 14.** Now a new window will open. Select the link and copy it. We will use this link in NetBeans to consume these services.



- 15.** Don't close Visual Studio and browser, just minimize it otherwise server will stop. But save the link anywhere, so that we can use it later.

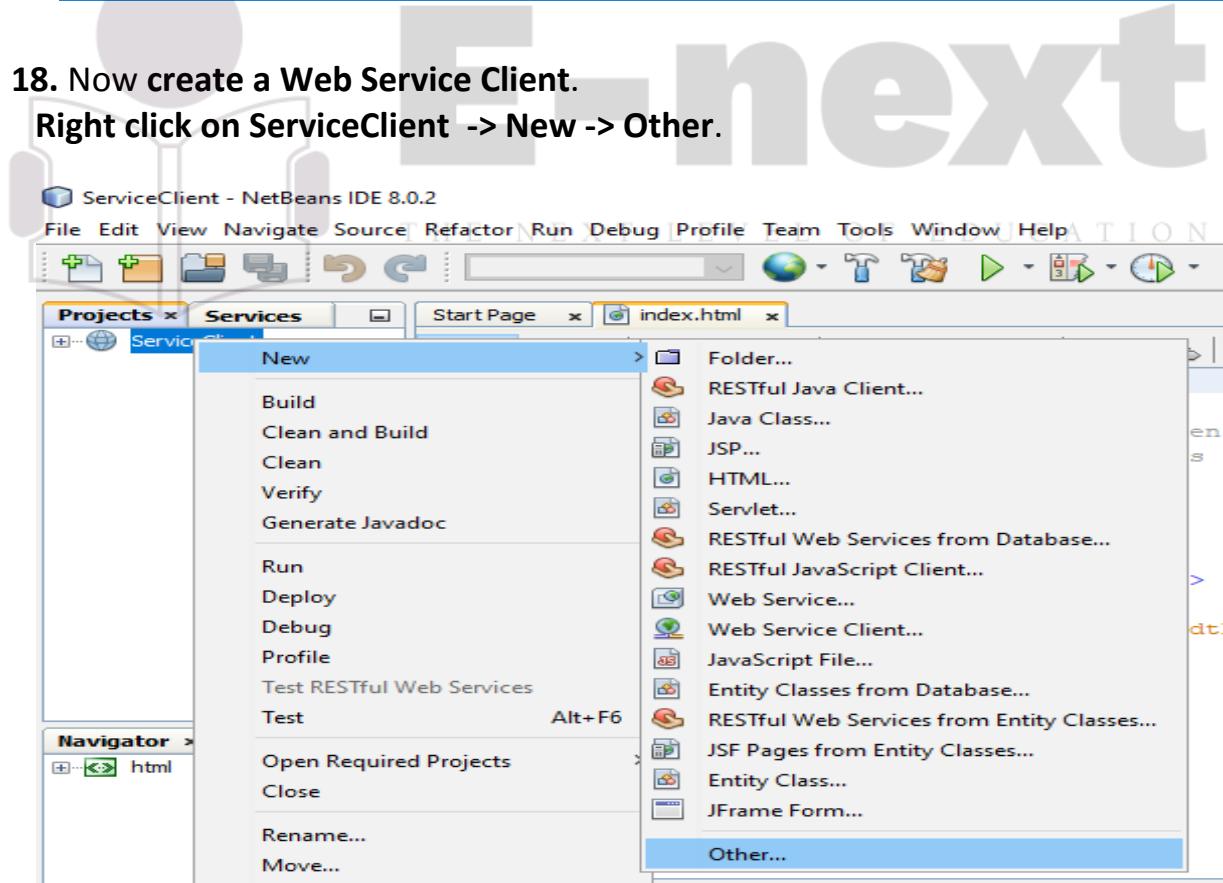
- 16.** Now open NetBeans.

- 17.** Create a Web Application with name ServiceClient. Next -> Finish.

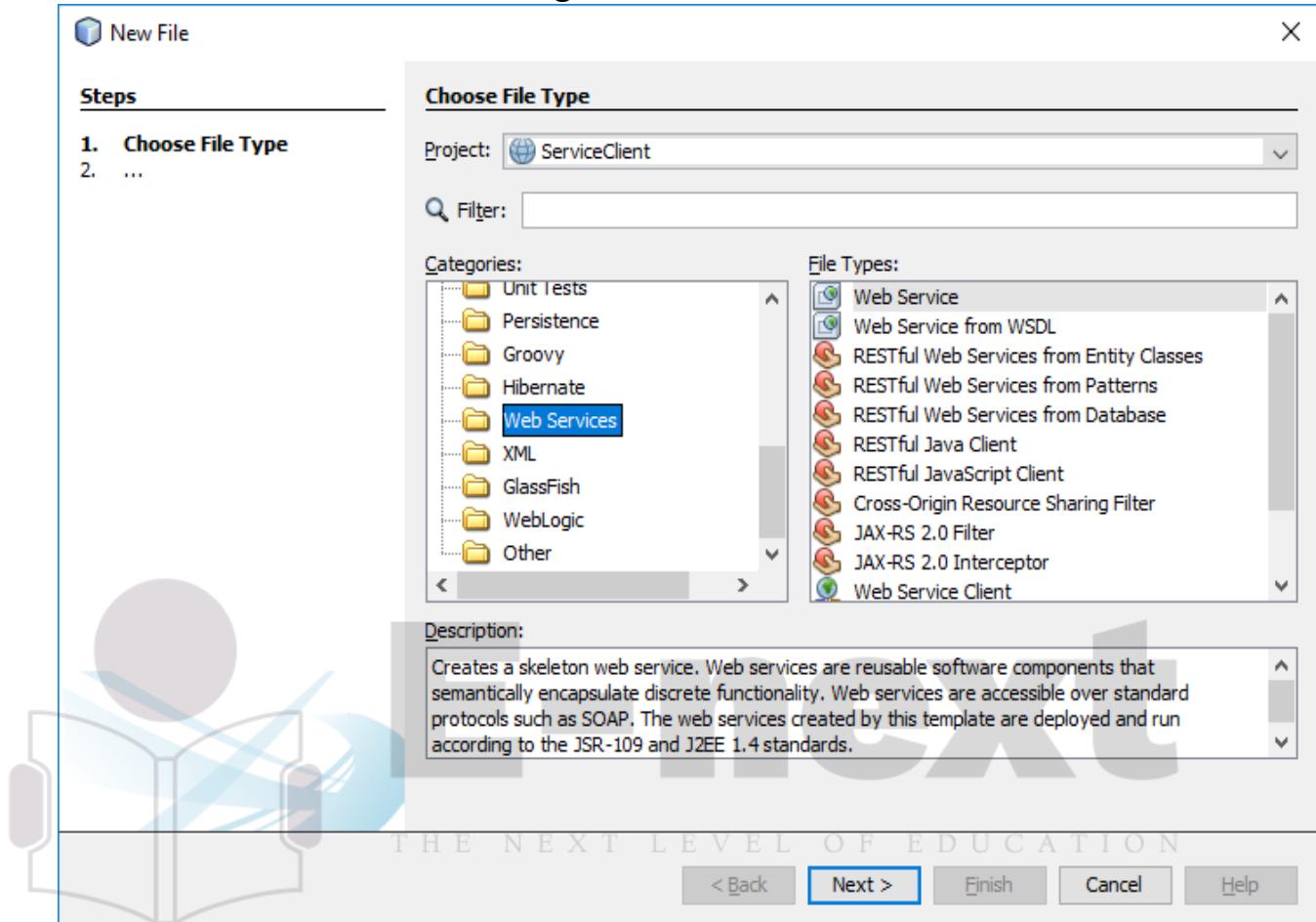


18. Now create a Web Service Client.

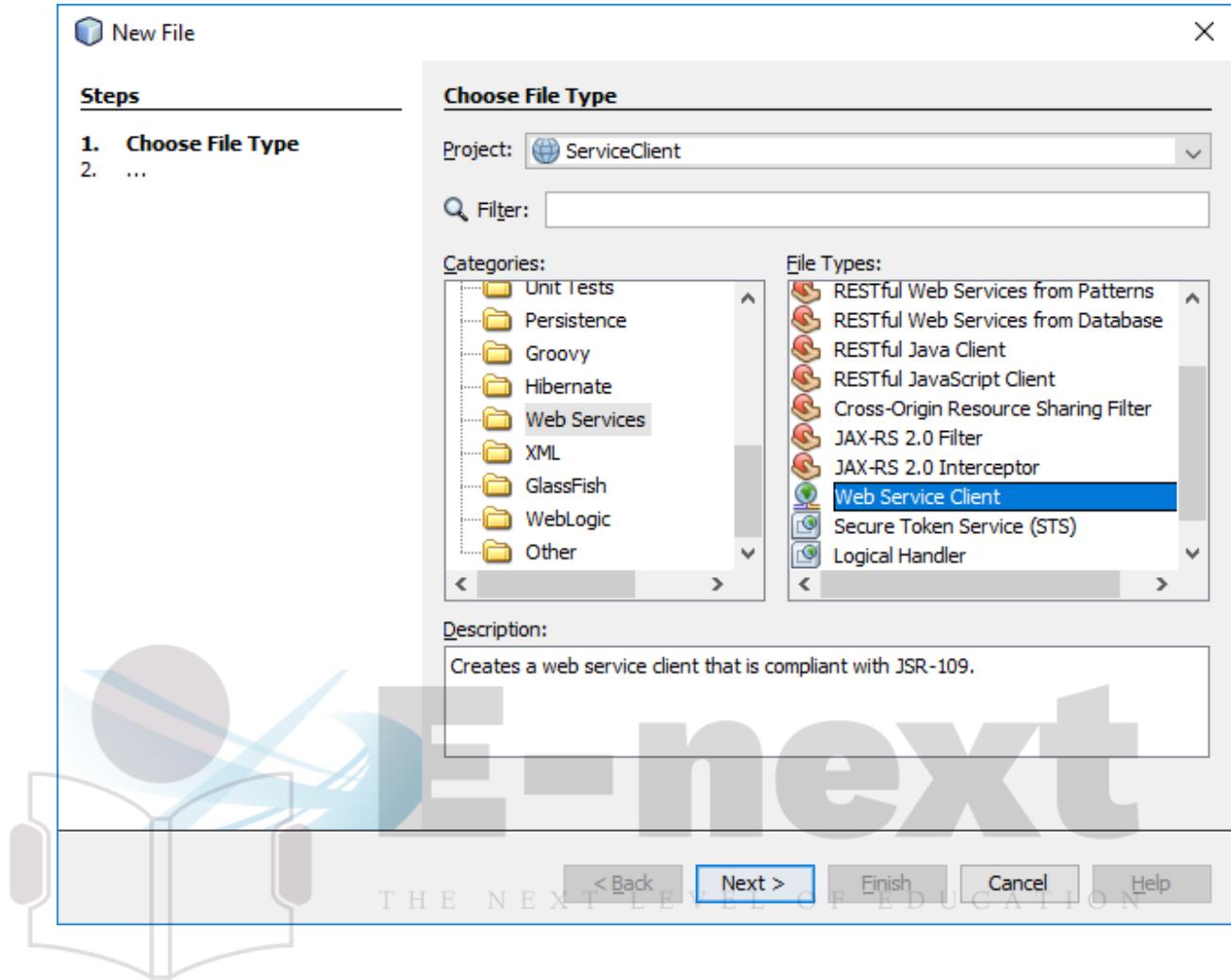
Right click on ServiceClient -> New -> Other.



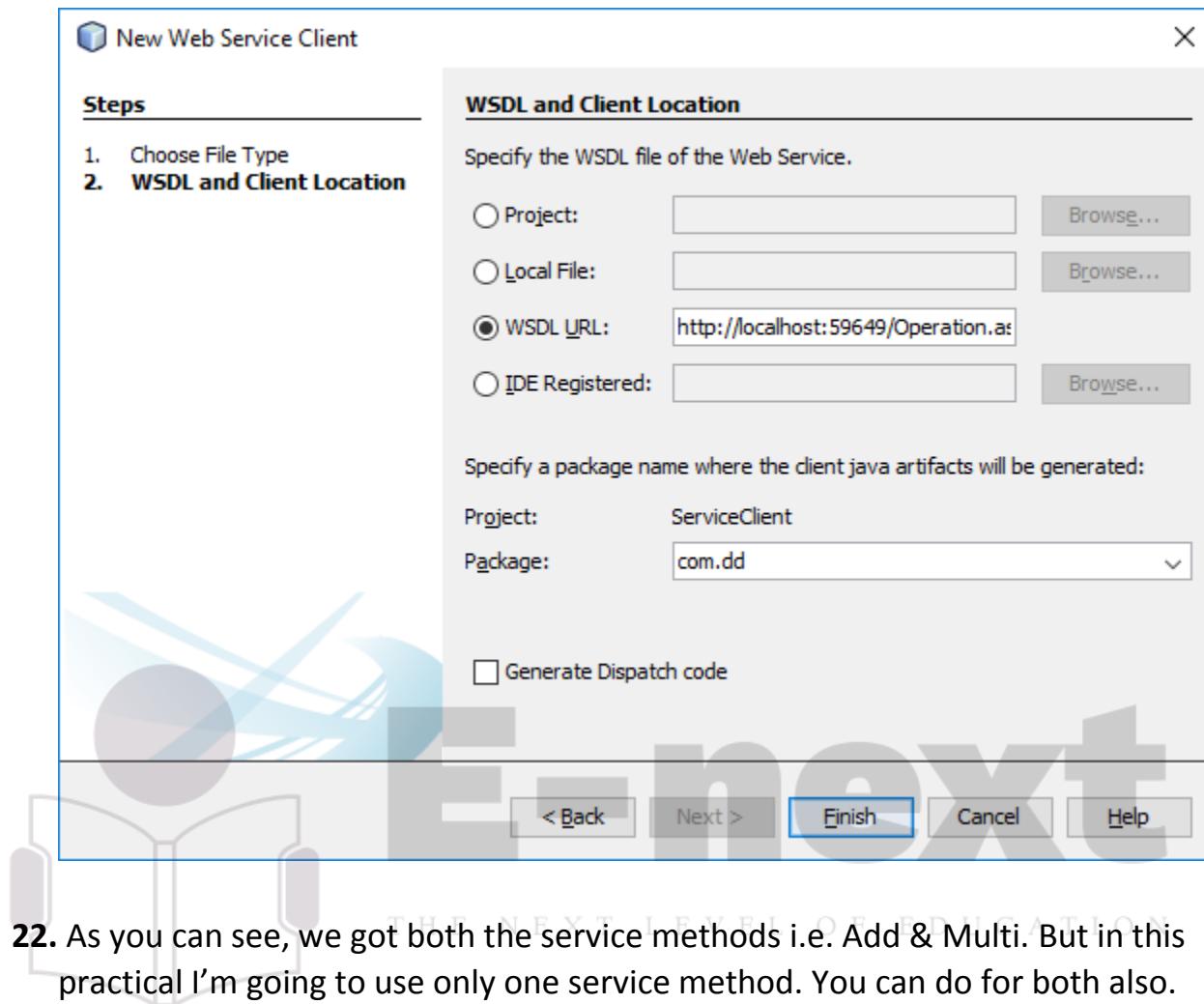
19. Now select Web Services in Categories section.



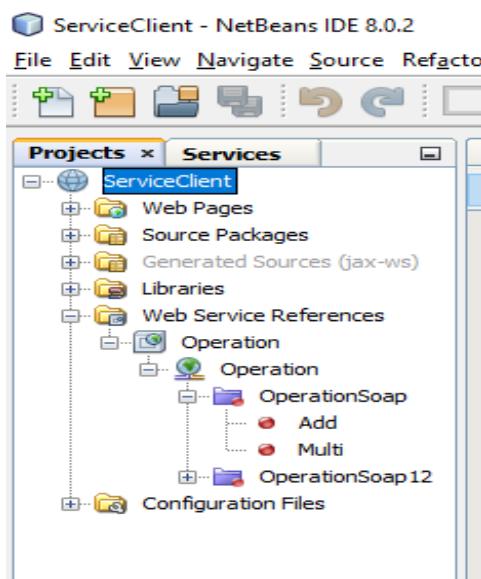
20. Select Web Service Client in File Types and Click on Next button.



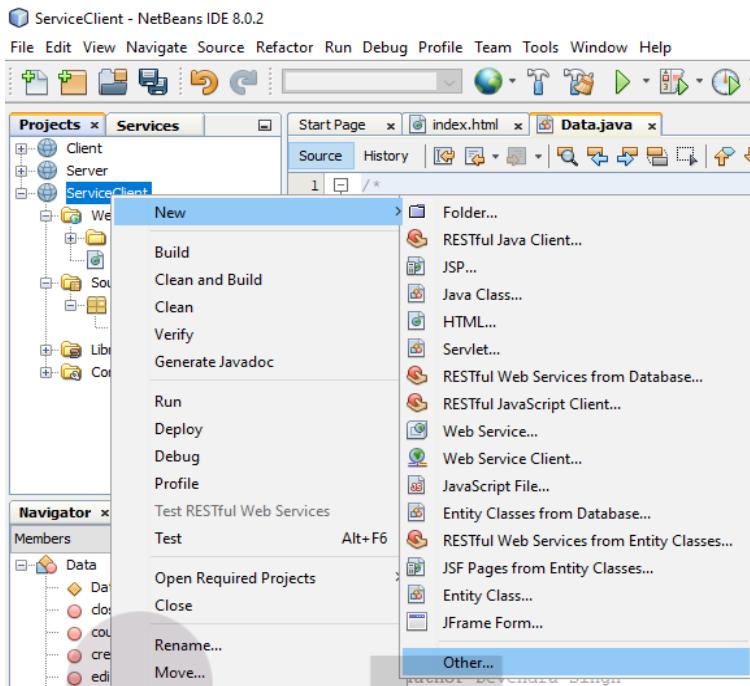
- 21. Select WSDL URL and paste the link that you have copied from browser on run of Visual Studio enter package name com.dd. After that click on Finish button.**



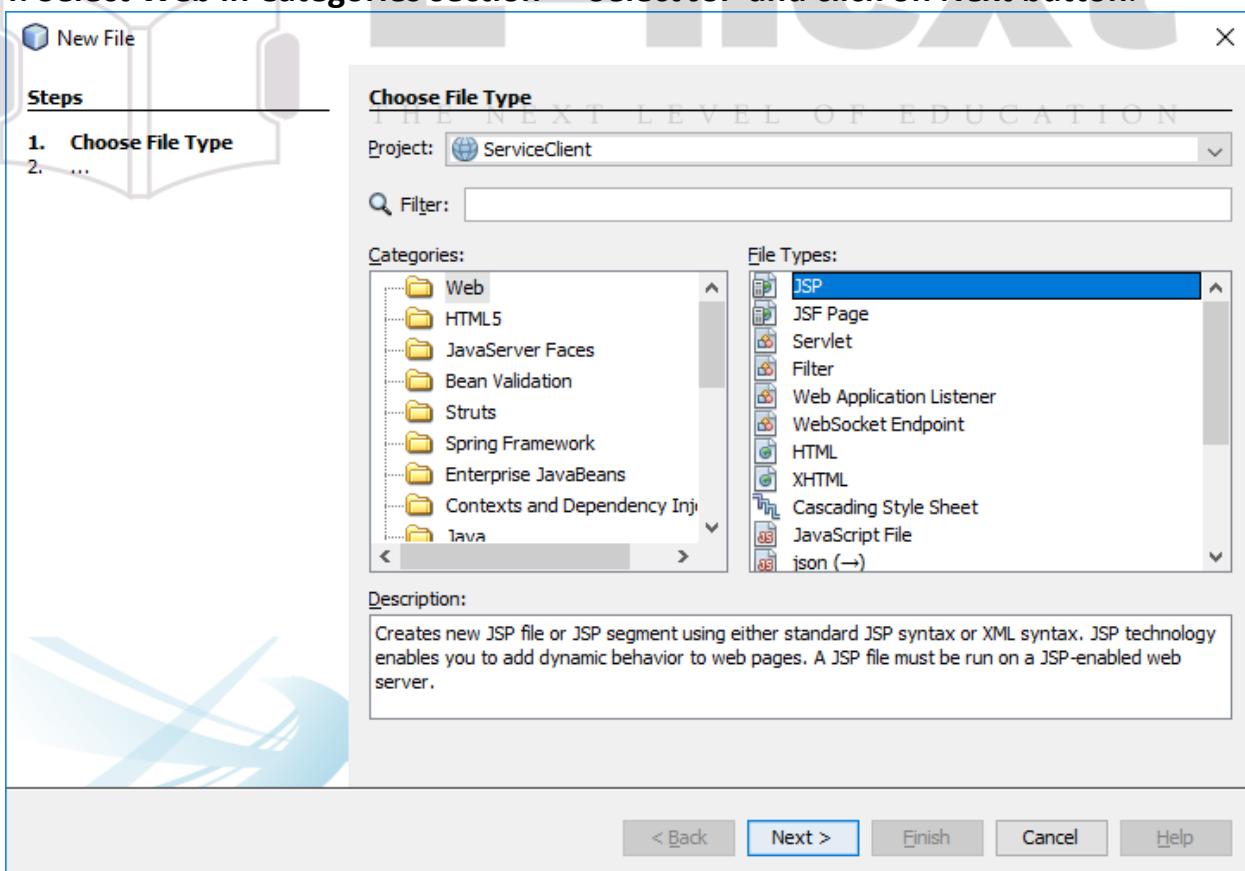
22. As you can see, we got both the service methods i.e. Add & Multi. But in this practical I'm going to use only one service method. You can do for both also.



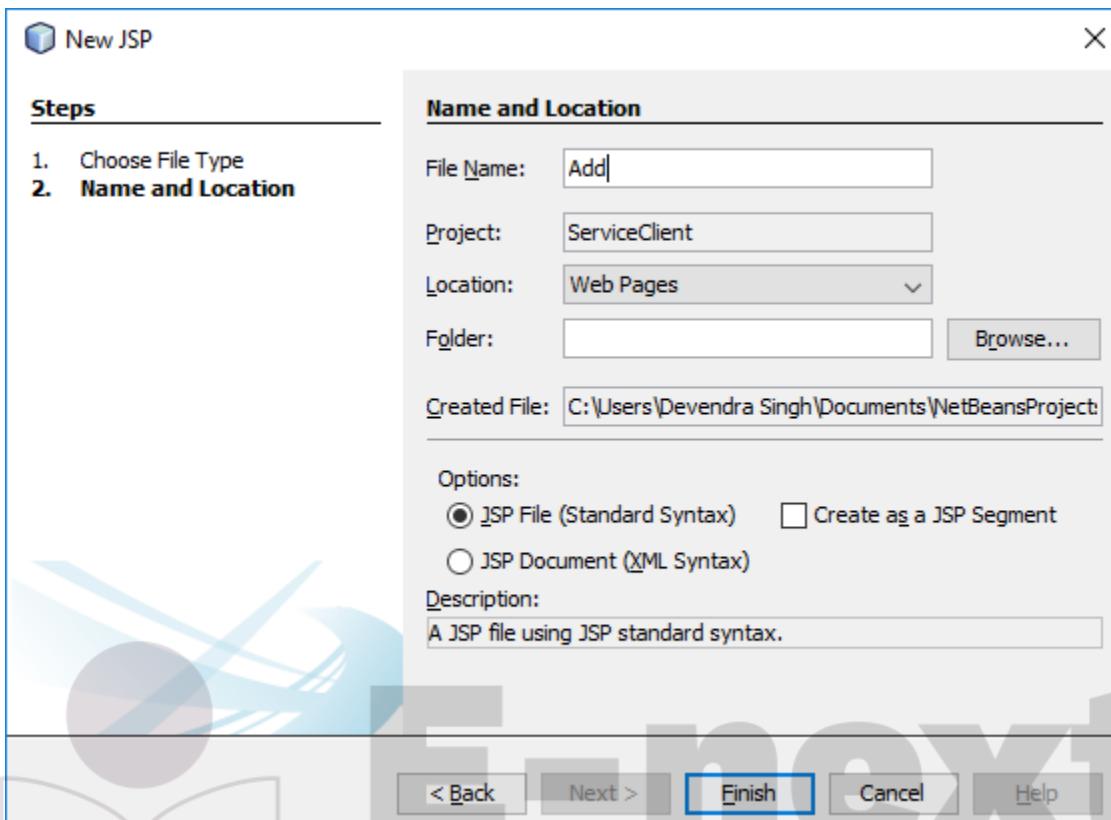
23. Now create a JSP page. Right click on ServiceClient -> New -> Other



24. Select Web in Categories section -> Select JSP and click on Next button.



25. Enter File Name Add and click on Finish button.



26. In Add.jsp file delete the selected part in body tag, because we don't need this.

THE NEXT LEVEL OF EDUCATION

<!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>
 </body>
</html>" data-bbox="154 569 867 881"/>

ServiceClient - NetBeans IDE 8.0.2

File View Navigate Source Refactor Run Debug Profile Team Tools Window Help

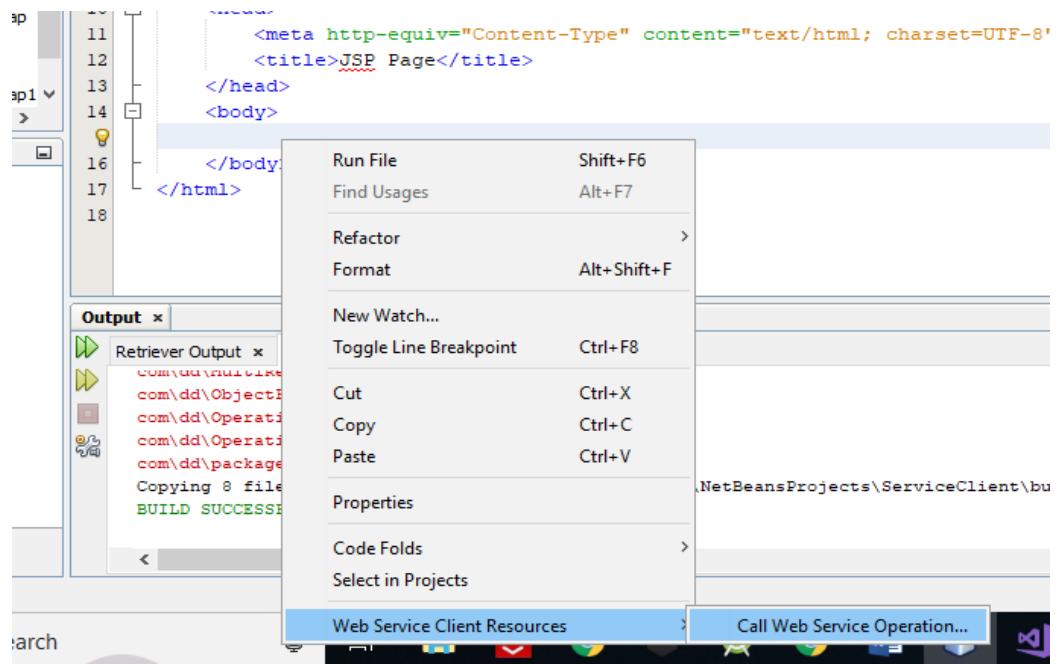
Start Page index.html Add.jsp

Source History

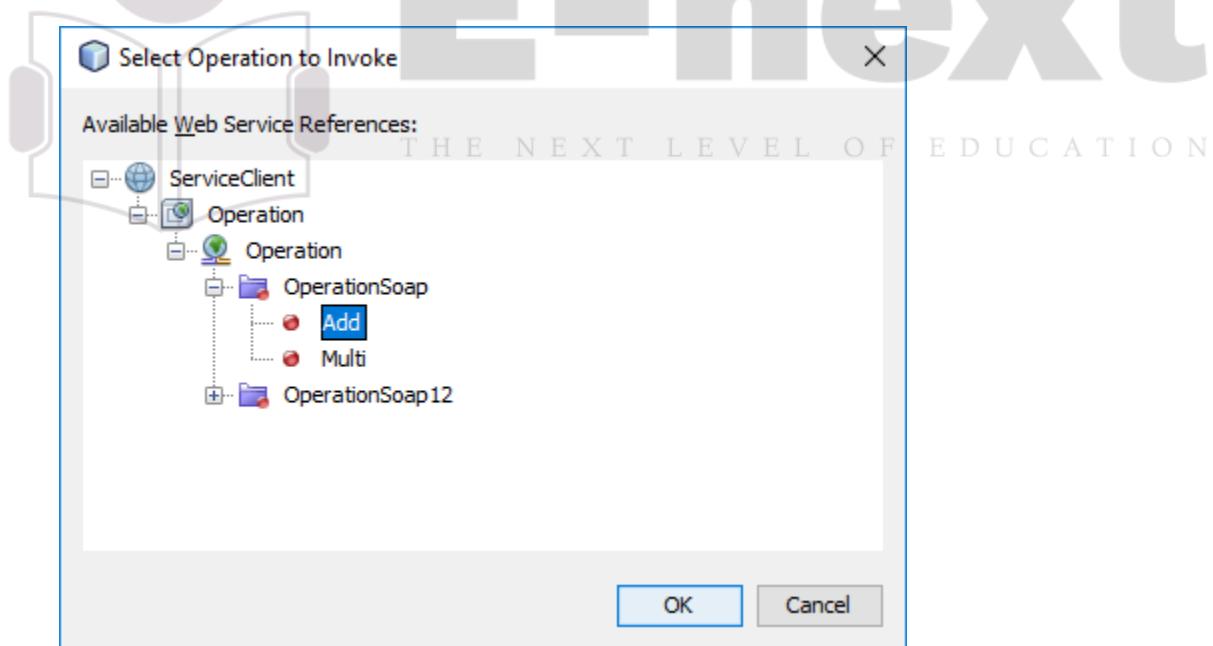
```
<%--  
Document : Add  
Created on : Aug 17, 2018, 10:58:06 AM  
Author : Devendra Singh  
--%>  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  

```

27. Right click between body tag and select Call Web Service Operation.

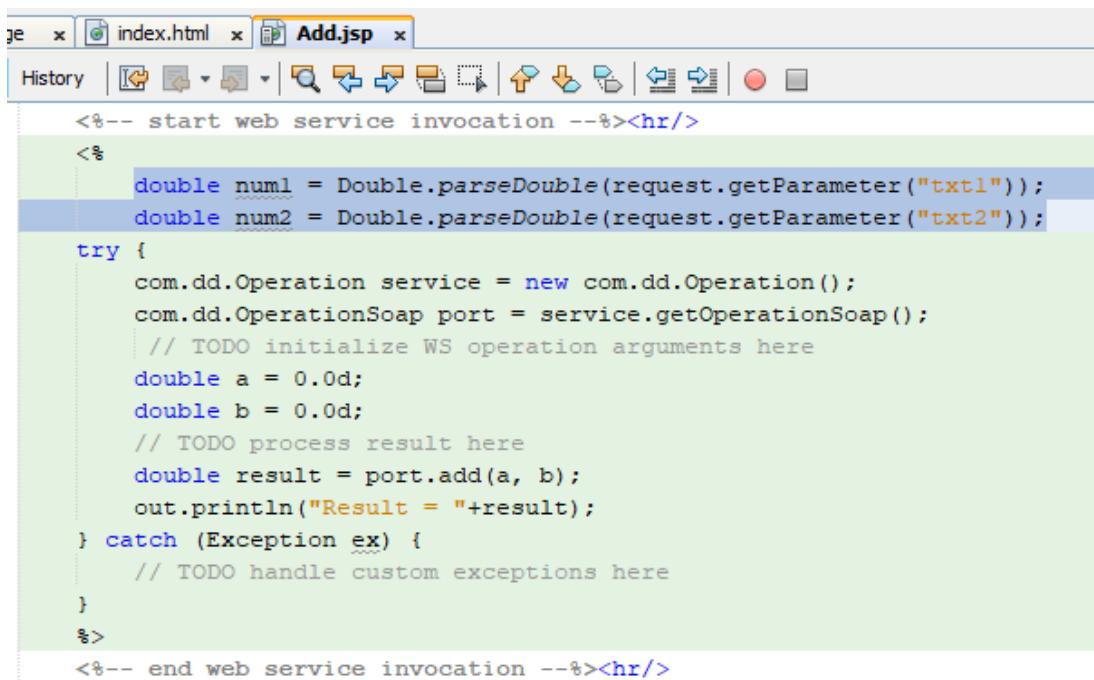


28. Expand and select Add. After select, click on OK button.



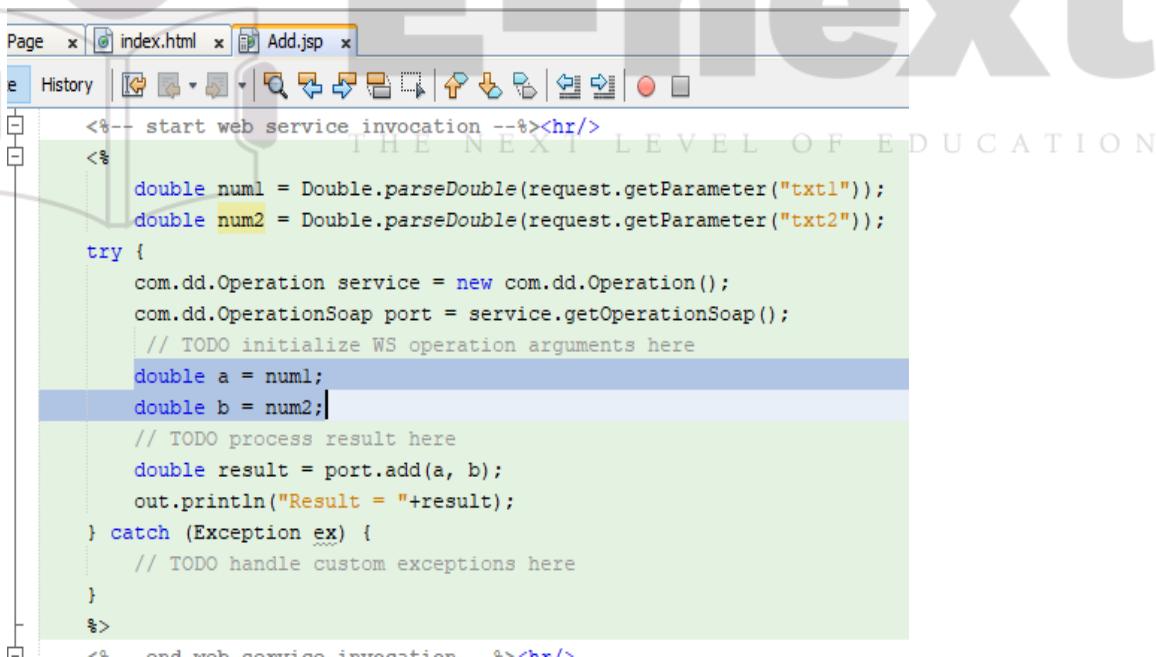
29. Now add the following code outside of try block.

```
double num1 = Double.parseDouble(request.getParameter("txt1"));
double num2 = Double.parseDouble(request.getParameter("txt2"));
```



The screenshot shows a Java IDE interface with multiple tabs at the top: 'je' (closed), 'index.html' (closed), and 'Add.jsp'. The 'Add.jsp' tab is active, displaying JSP code. The code starts with a comment: '<%-- start web service invocation --%><hr/>'. It then contains a scriptlet block: '<%'. Inside this block, two double variables are declared: 'double num1 = Double.parseDouble(request.getParameter("txt1"));' and 'double num2 = Double.parseDouble(request.getParameter("txt2"));'. A try-catch block follows: 'try {'. Inside the try block, a service object is created: 'com.dd.Operation service = new com.dd.Operation();'. A port object is then obtained from the service: 'com.dd.OperationSoap port = service.getOperationSoap();'. A TODO comment follows: '// TODO initialize WS operation arguments here'. Below this, two double variables 'a' and 'b' are initialized to 0.0d. Another TODO comment follows: '// TODO process result here'. A result variable is then assigned: 'double result = port.add(a, b);'. Finally, the result is printed: 'out.println("Result = "+result);'. The catch block handles exceptions: '} catch (Exception ex) {'. Inside the catch block, a TODO comment follows: '// TODO handle custom exceptions here'. The scriptlet block ends with '%>'. The code concludes with another comment: '<%-- end web service invocation --%><hr/>'.

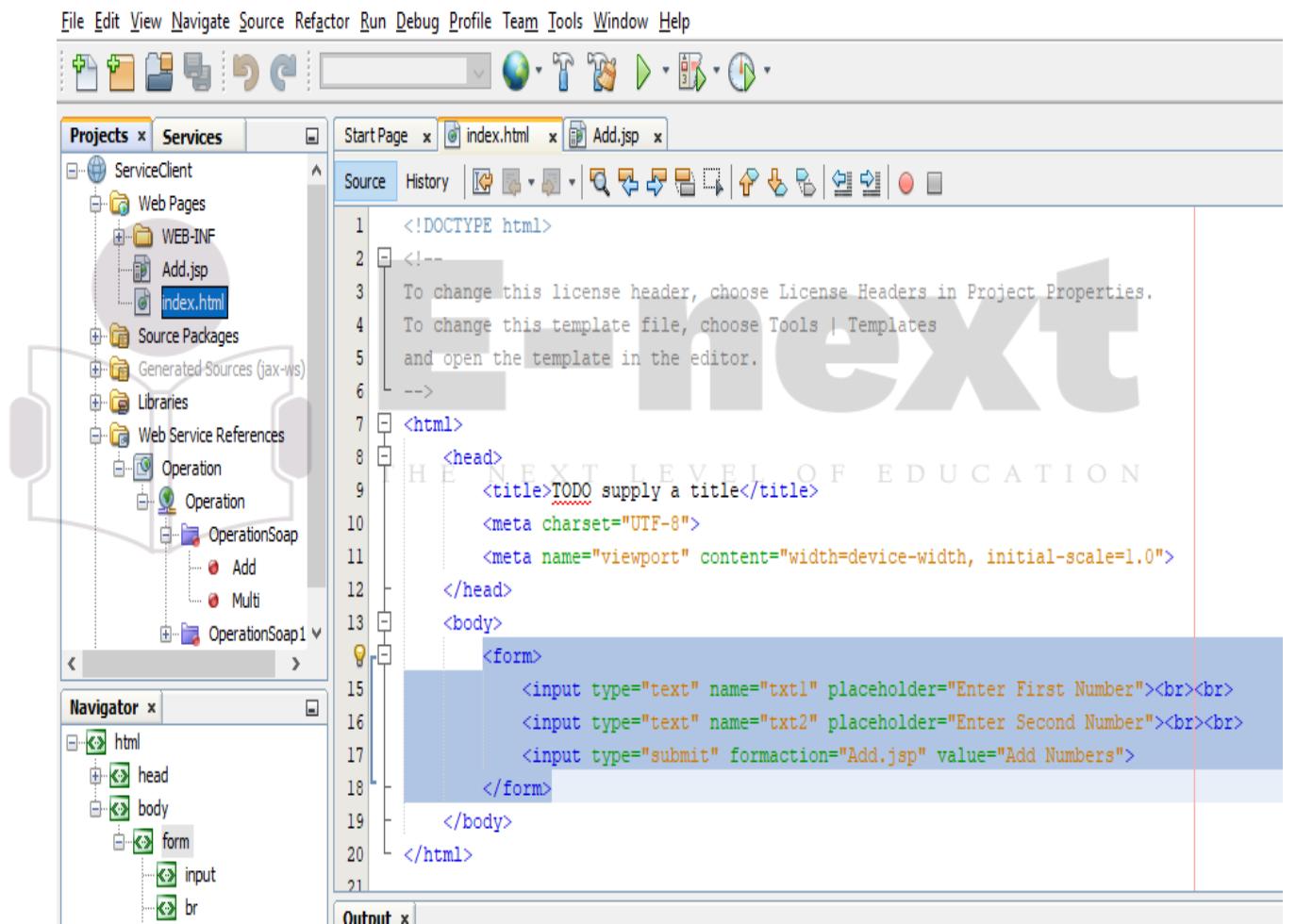
30. Now pass num1 & num2 to a & b variable respectively. After that press **Ctrl+S** to save this code.



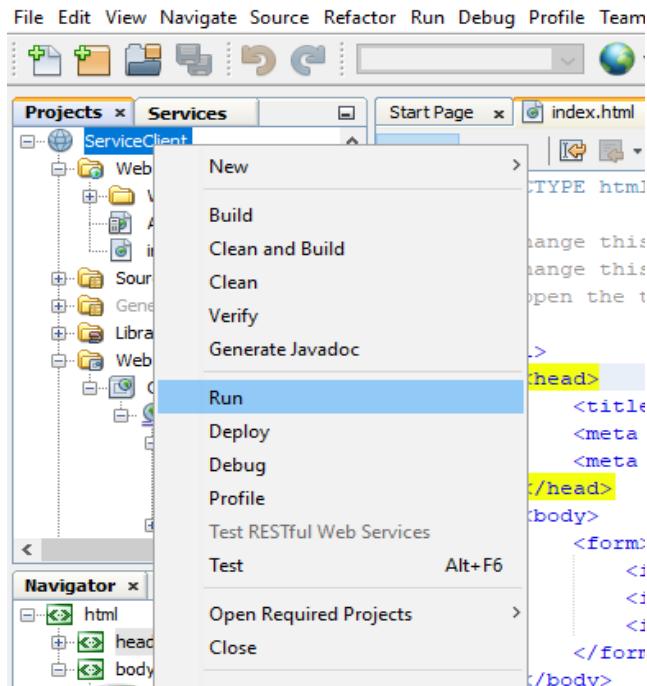
The screenshot shows the same Java IDE interface as the previous one, but the code has been modified. The variable declarations 'double num1 = Double.parseDouble(request.getParameter("txt1"));' and 'double num2 = Double.parseDouble(request.getParameter("txt2"));' have been moved inside the 'try' block. The modified code is: '<%'. Inside the try block, the variable declarations are now: 'double a = num1;' and 'double b = num2;'. The rest of the code remains the same, including the service creation, port retrieval, TODO comments, result assignment, exception handling, and final output printing. The code concludes with '<%-- end web service invocation --%><hr/>'.

31. Now open index.html file of ServiceClient project and replace the contents of body tag with following code. After that press **Ctrl+S** to save it.

```
<form>
    <input type="text" name="txt1" placeholder="Enter First
Number"><br><br>
    <input type="text" name="txt2" placeholder="Enter Second
Number"><br><br>
    <input type="submit" formaction="Add.jsp" value="Add Numbers">
</form>
```



32. Now run the ServerClient web application.

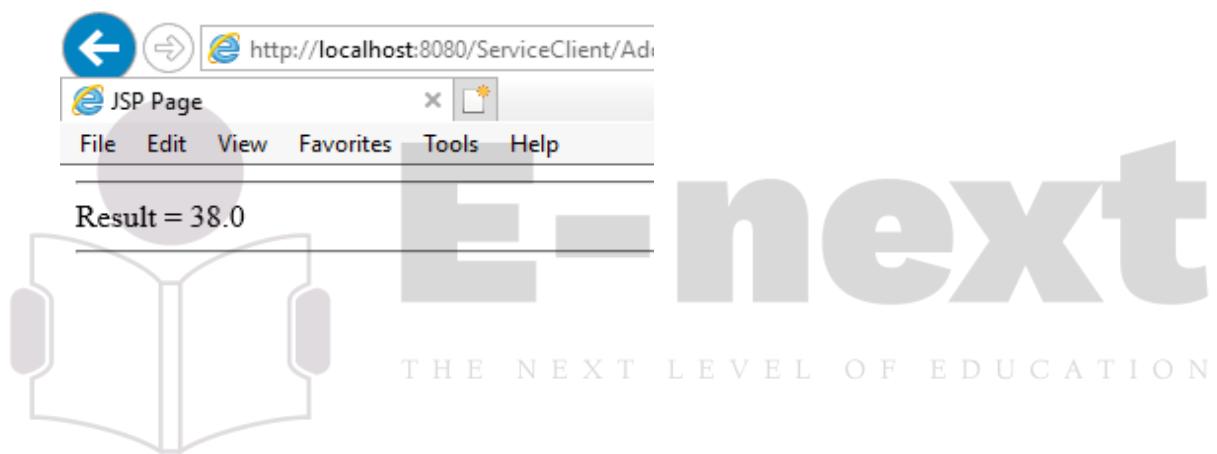
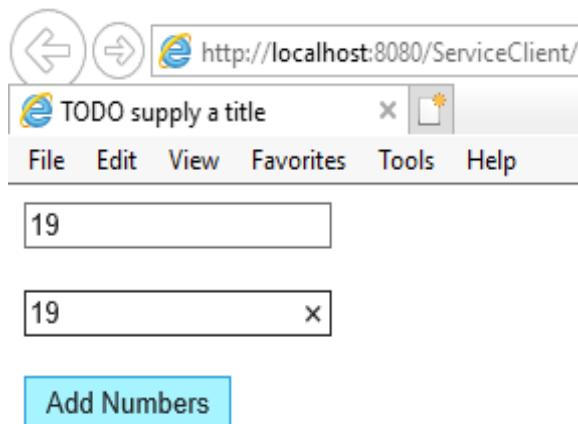


33. A window will open in browser as below.



34. Now enter two numbers and click on Add Numbers button. Wait to get result.....

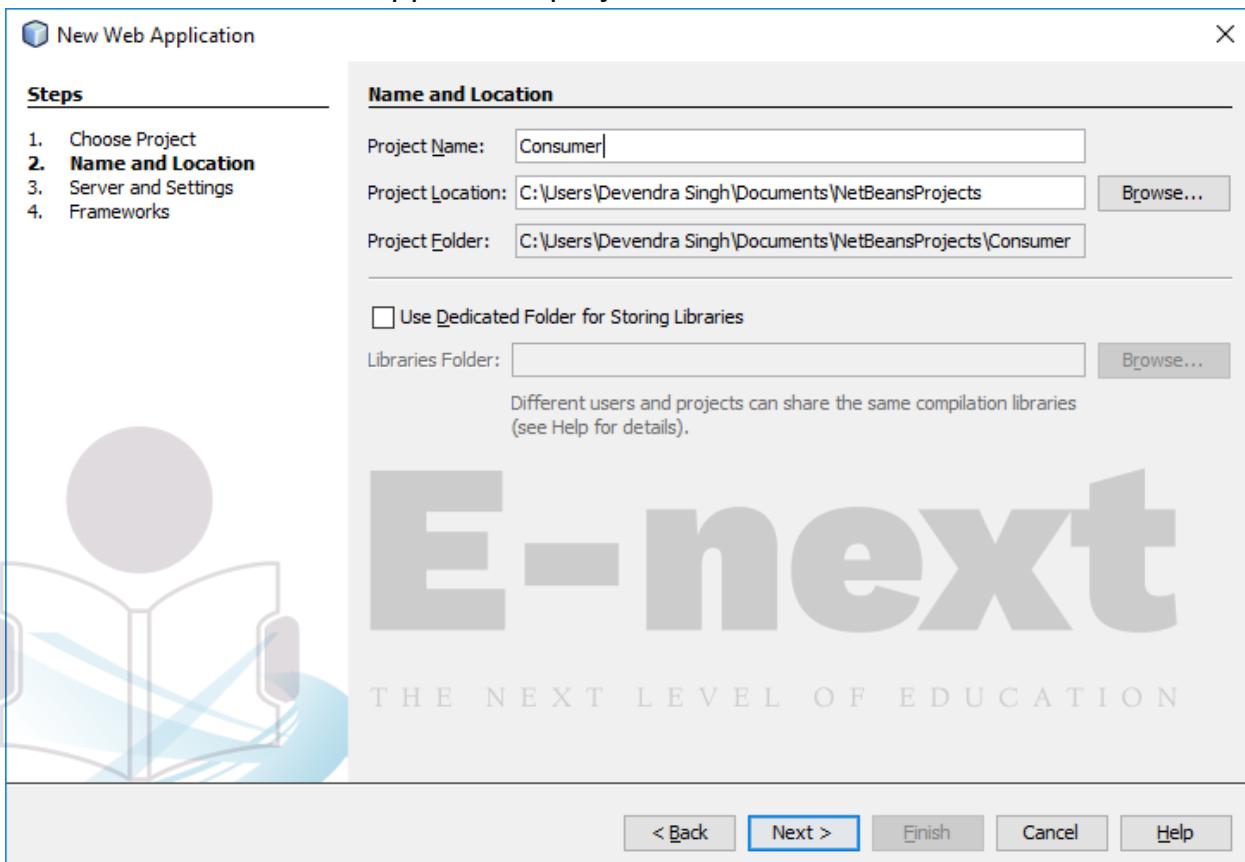
You will get result on a new page.



Note: To do this practical, follow the steps from 1 to 18 present in practical – 7.

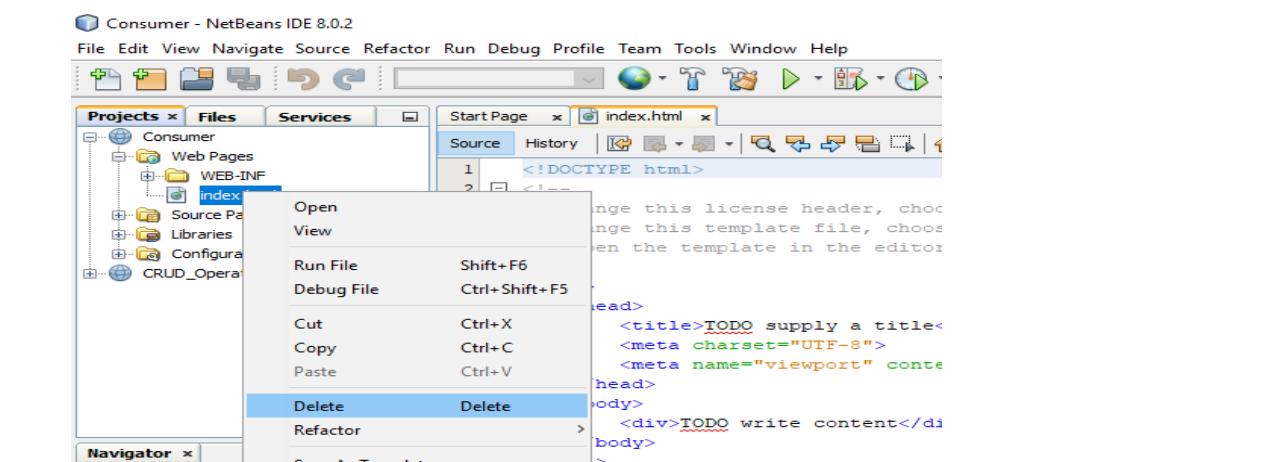
After that do not close or restart the NetBeans.

1. Create an another Web Application project and Give name as **Consumer**.

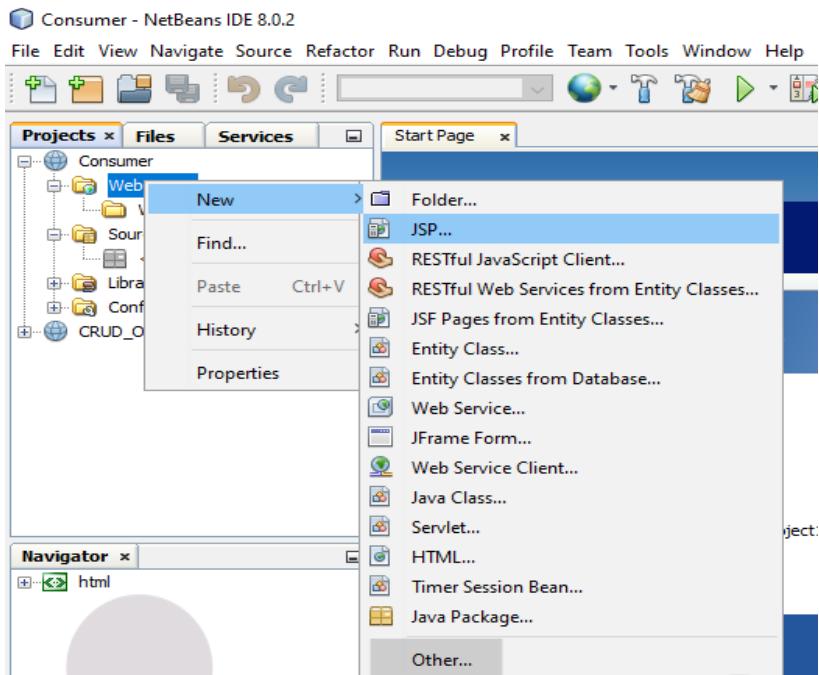


2. And create it. **Next -> Finish**.

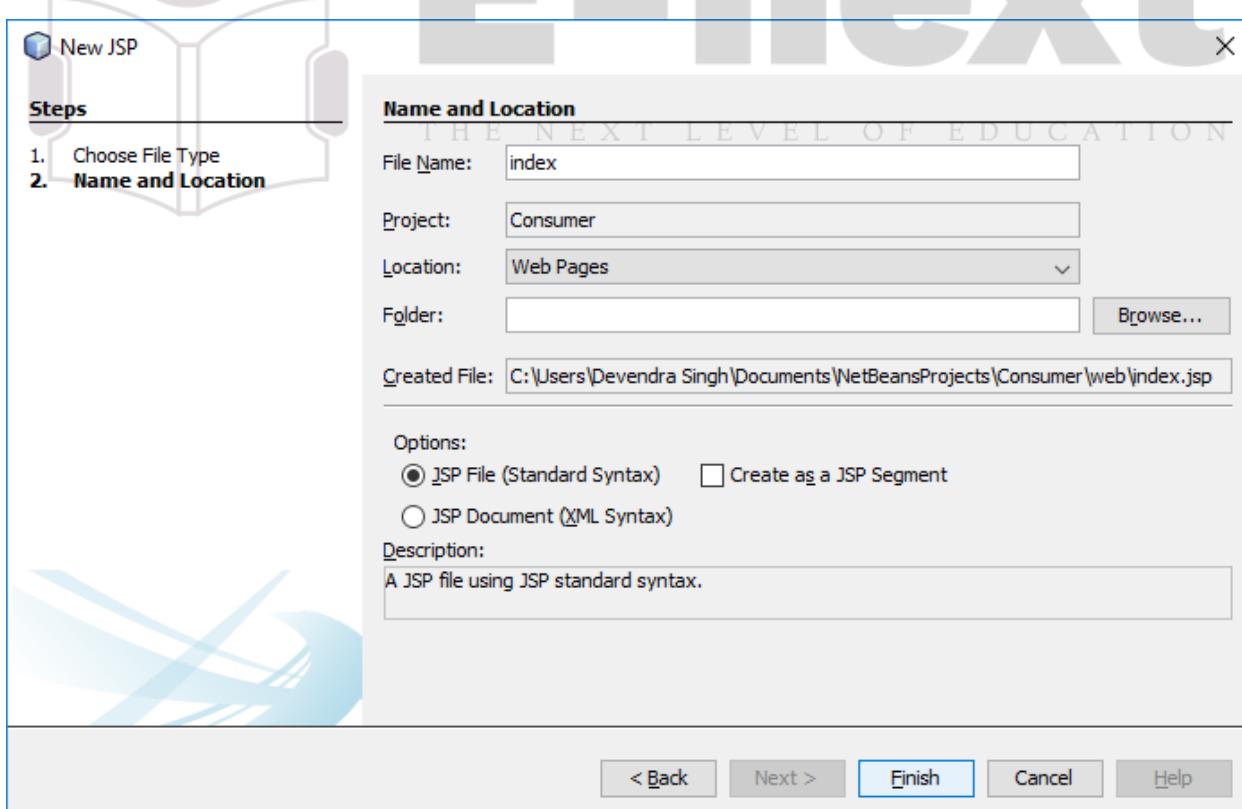
3. Now right click on index.html and delete it.



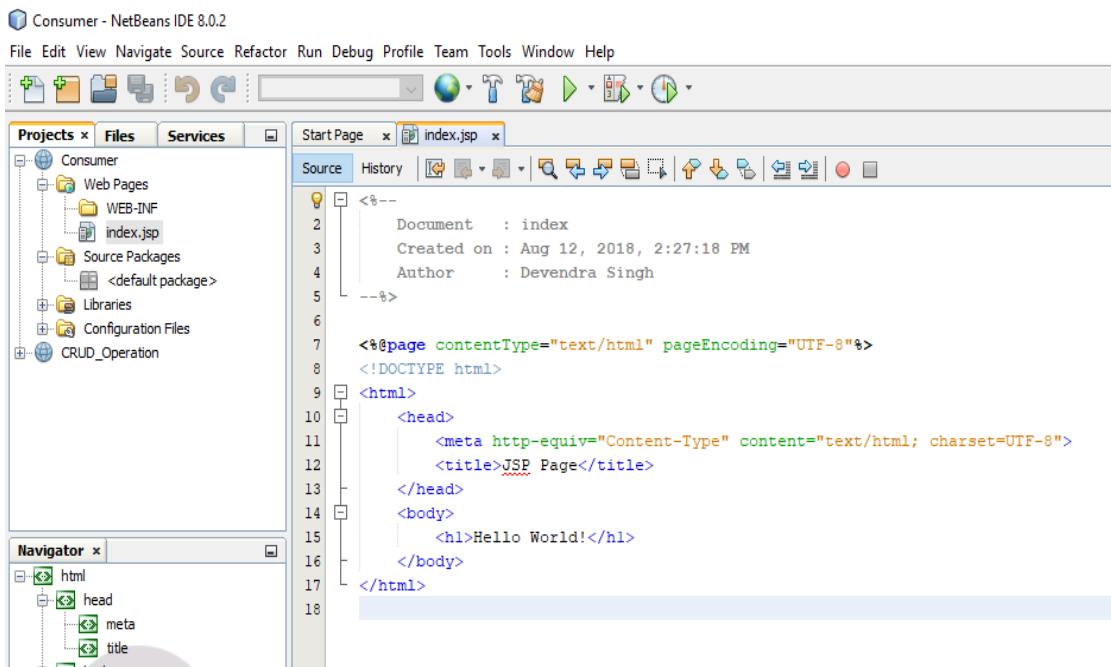
4. Now right click on Web Pages and select JSP to add a JSP page.



5. Give index name to it and then click on Finish.



6. Now index.jsp page will open like below.



7. Now select **HTML content of index.jsp** file and replace it with following bold letter codes.



```
<HTML>
<HEAD>
<script language="javascript">
var xmlhttp;
function init()
{
xmlhttp=new XMLHttpRequest();
}
function readLocal(){
if(window.localStorage){
var seller=localStorage.getItem("seller");
seller=JSON.parse(seller);
document.getElementById("firstName").value=seller.firstName;
document.getElementById("sellerid").value=seller.id;
}
}
```

```
function saveLocal()
{
var sellerid=document.getElementById("sellerid");
var
url="http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/"+sellerid.value;
xmlhttp.open('GET',url,true);
xmlhttp.send(null);
xmlhttp.onreadystatechange =function(){

if(xmlhttp.readyState==4){alert("6"+sellerid);
if(xmlhttp.status==200){alert("7"+sellerid);
var seller =eval("(" +xmlhttp.responseText+ ")");
if(window.localStorage){

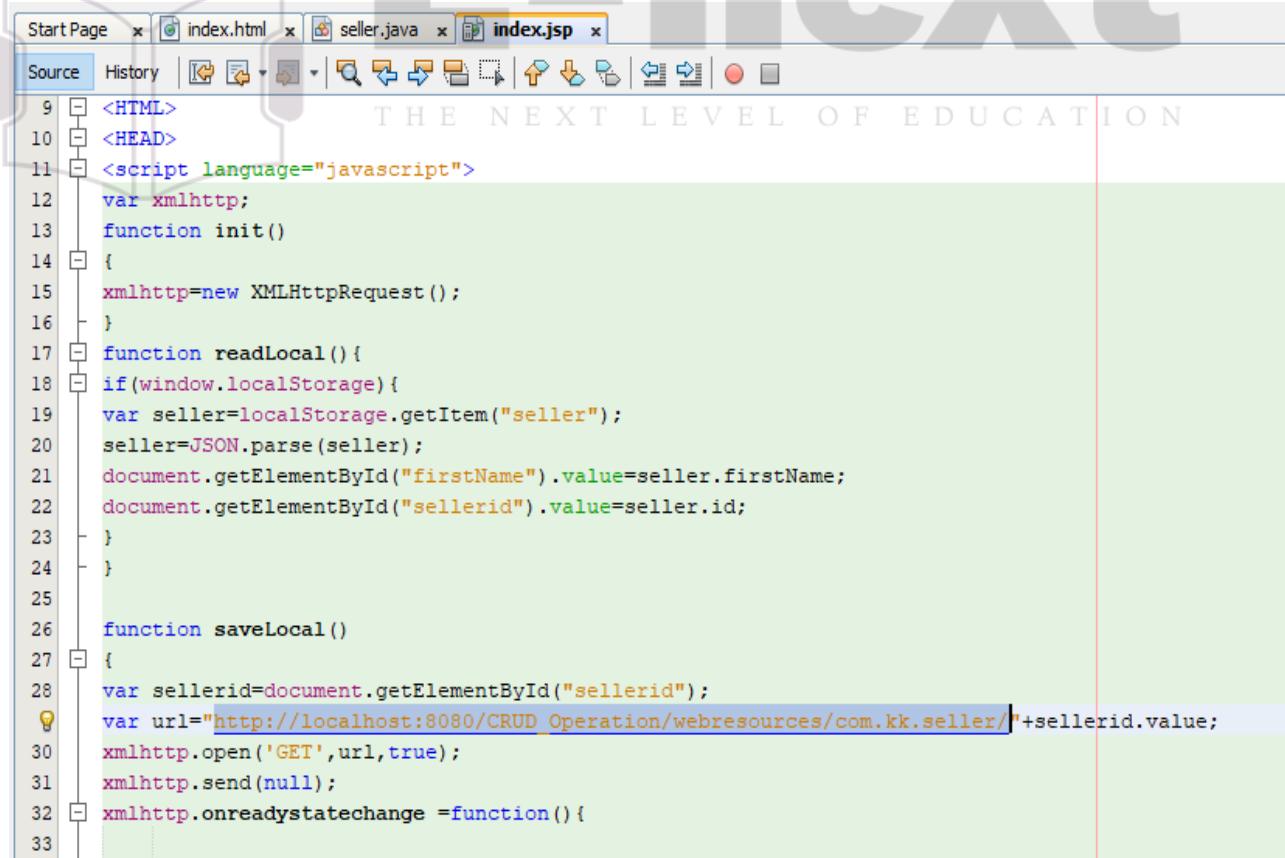
localStorage.setItem("seller",JSON.stringify(seller));
alert("information stored
successfully"+seller.firstName);
}
else{
alert("notStored");
}
}
else
alert("error");
}
};

}

</script>
</head>
<body onLoad ="init()">
<table>
<tr>
<td>Enter id:</td>
<td><input type="text" id="sellerid"/>
```

```
<input type="button" value="load employee in local browser"
onClick="saveLocal()"/>
</td>
</tr>
<tr>
<td>read from local</td>
<td><input type="button" value="Send values" onClick="readLocal()"/></td>
</tr>
<tr>
<td>first Name:</td>
<td> <input type="text" id="firstName"/></td>
</tr>
</table>
</body>
</html>
```

8. Now in the following pic, **highlighted URL** is most important. I'm explaining it next step.



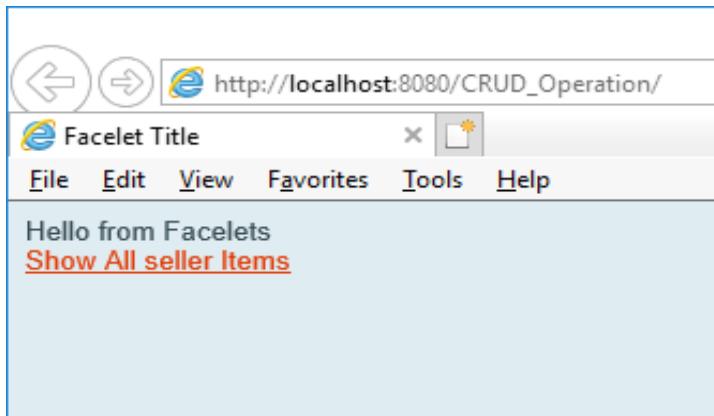
The screenshot shows a code editor window with several tabs at the top: "Start Page", "index.html", "seller.java", and "index.jsp". The "index.jsp" tab is active, showing Java code. The code includes a script block with JavaScript functions for reading and saving data from local storage. A specific line of code is highlighted with a blue background and a yellow border:

```
var url="http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/"+sellerid.value;
```

This highlighted line contains a URL that is used to make an HTTP request to a specific endpoint on a local server.

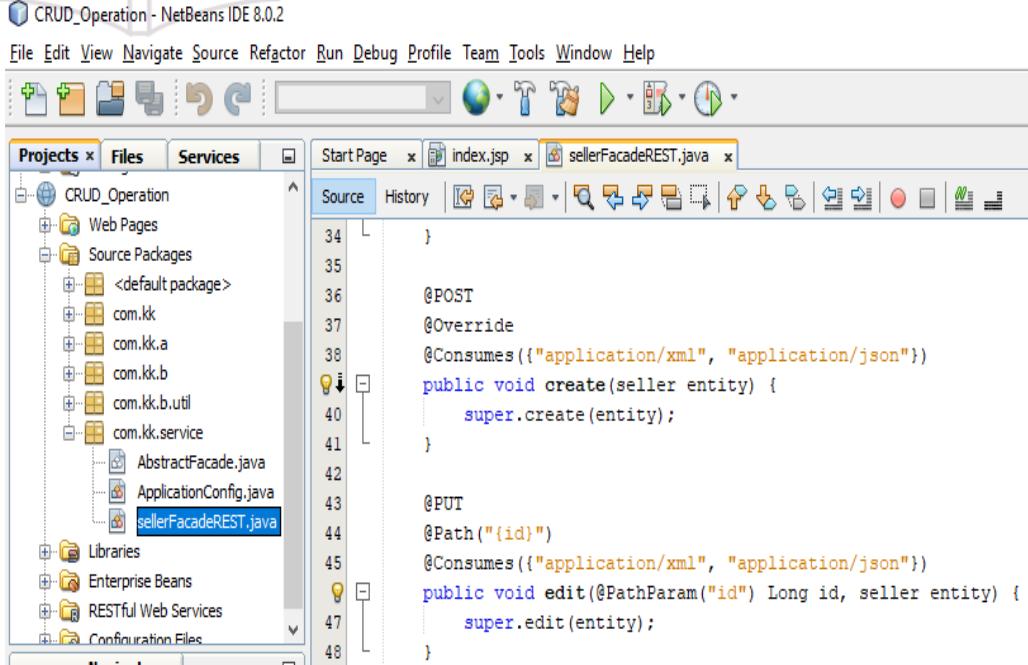
9. http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/

Red part is the URL of which is obtained in browser by running of CRUD_Operation web application.



Blue part in above link is **static**. But red part is **dynamic** which is based on practical 7. So if you have changed name anywhere then the above URL will change accordingly.

10. Now open sellerFacadeREST.java file by follow below pic available in CRUD_Operation web application.



11. Now delete the selected part. Because it will return data in XML format to the consumer. But we have written javascript in **index.jsp** for JSON data format only. After that **deploy the CRUD_Operation web application**; so that it will update the changes.

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects x Files Services
CRUD_Operation
  Web Pages
  Source Packages
    <default package>
    com.kk
    com.kk.a
    com.kk.b
    com.kk.b.util
    com.kk.service
      AbstractFacade.java
      ApplicationConfig.java
      sellerFacadeREST.java
  Libraries
  Enterprise Beans
  RESTful Web Services
  Configuration Files
find - Navigator x
Members <empty>
  sellerFacadeREST :: AbstractFacade<...
    sellerFacadeREST()
    countREST() : String
    create(seller entity)
    edit(Long id, seller entity)
    find(Long id) : seller
    findAll() : List<seller>
    findRange(Integer from, Integer to) : List<seller>
    EntityManager() : EntityManager
    remove(Long id)
    em : EntityManager
Output

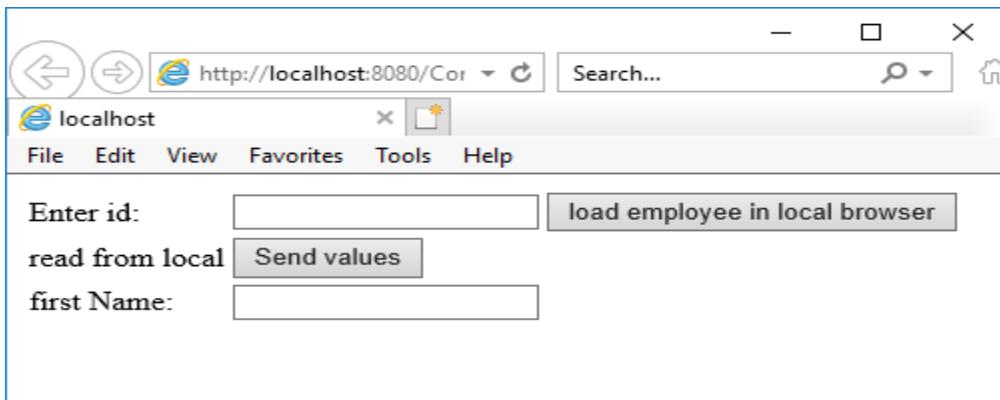
```

```

Source History | 
34 }
35
36 @POST
37 @Override
38 @Consumes({"application/xml", "application/json"})
39 public void create(seller entity) {
40     super.create(entity);
41 }
42
43 @PUT
44 @Path("{id}")
45 @Consumes({"application/xml", "application/json"})
46 public void edit(@PathParam("id") Long id, seller entity) {
47     super.edit(entity);
48 }
49
50 @DELETE
51 @Path("{id}")
52 public void remove(@PathParam("id") Long id) {
53     super.remove(super.find(id));
54 }
55
56 @GET
57 @Path("{id}")
58 @Produces({"application/xml", "application/json"})
59 public seller find(@PathParam("id") Long id){
60     return super.find(id);
61 }
62
63 @GET
64 @Override
65

```

12. Now run the Consumer application. A window will open in browser like below.



13. Now if you will enter an id into **Enter id** textbox which you have created in database (id from data in last step of practical 7) and then click on **load employee in local browser** button. It will get the detail of particular entered id and will store it into local storage of browser. Now click on **Send Values** to get the first name of entered id.

A screenshot of a web browser window. The address bar shows the URL <http://localhost:8080/Consumer/>. The menu bar includes File, Edit, View, Favorites, Tools, and Help. Below the menu is a toolbar with icons for back, forward, stop, and refresh. The main content area contains the following form fields:

Enter id:	<input type="text" value="1"/>	<input type="button" value="load employee in local browser"/>
read from local	<input type="button" value="Send values"/>	
first Name:	<input type="text" value="Devendra"/>	



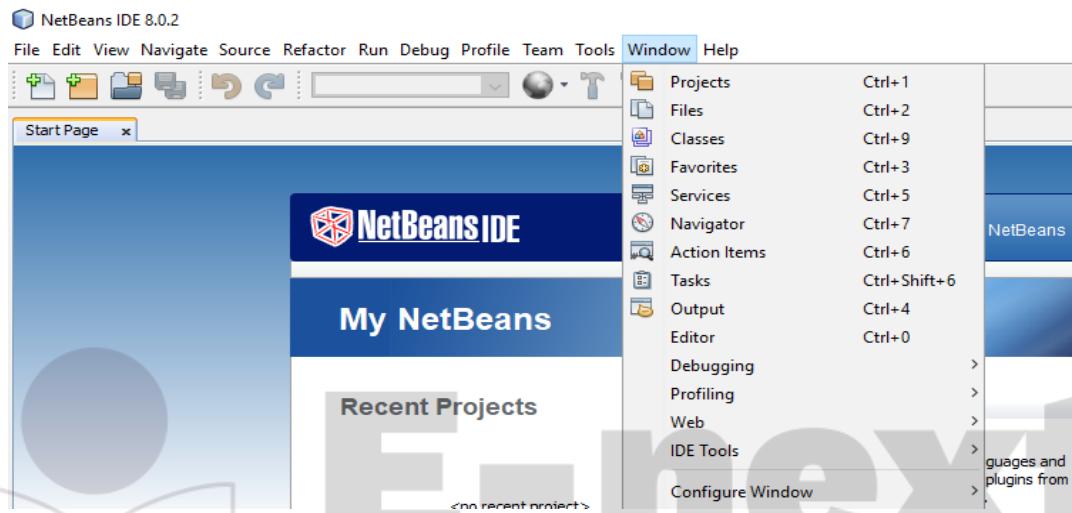
E-next

THE NEXT LEVEL OF EDUCATION

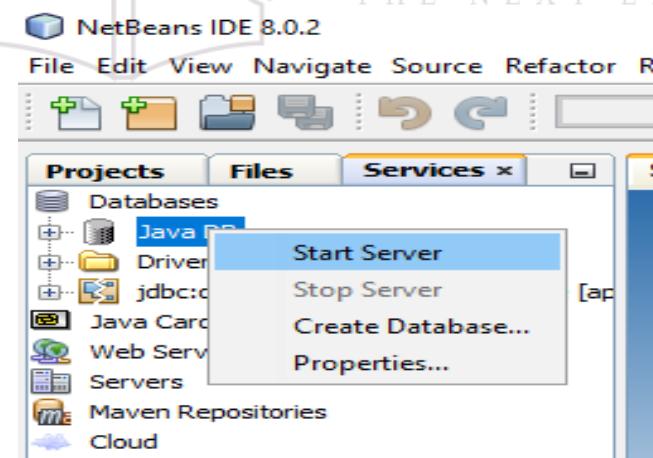
PRACTICAL-6

Aim: Define a web service method that returns the contents of a database in a JSON string. The contents should be displayed in a tabular format.

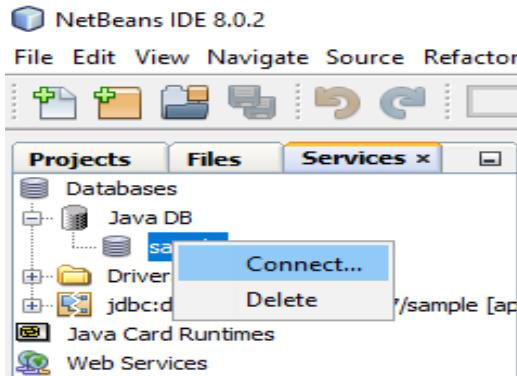
1. Click on Window menu and click on Projects, Files & Services to open it.



2. Right click on Java DB and then click on Start Server to start the server.

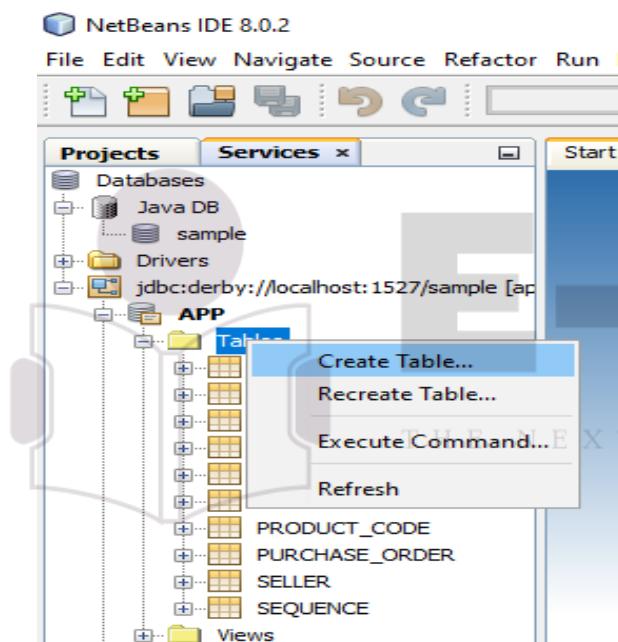


3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.

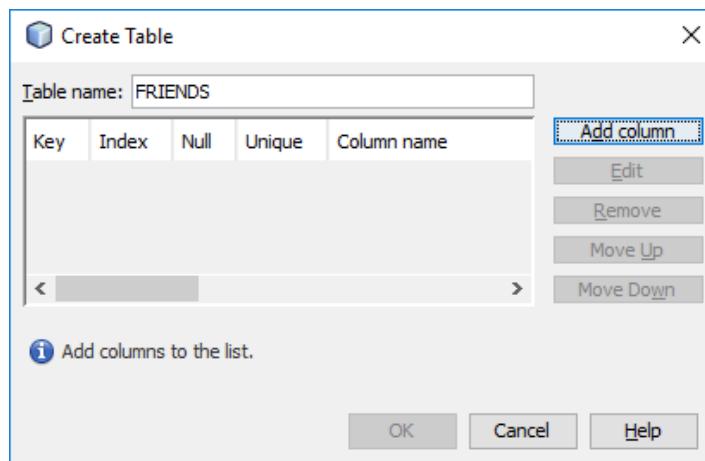


4. Now we are going to create a table in default database **sample**.

Right click on Table -> Create Table

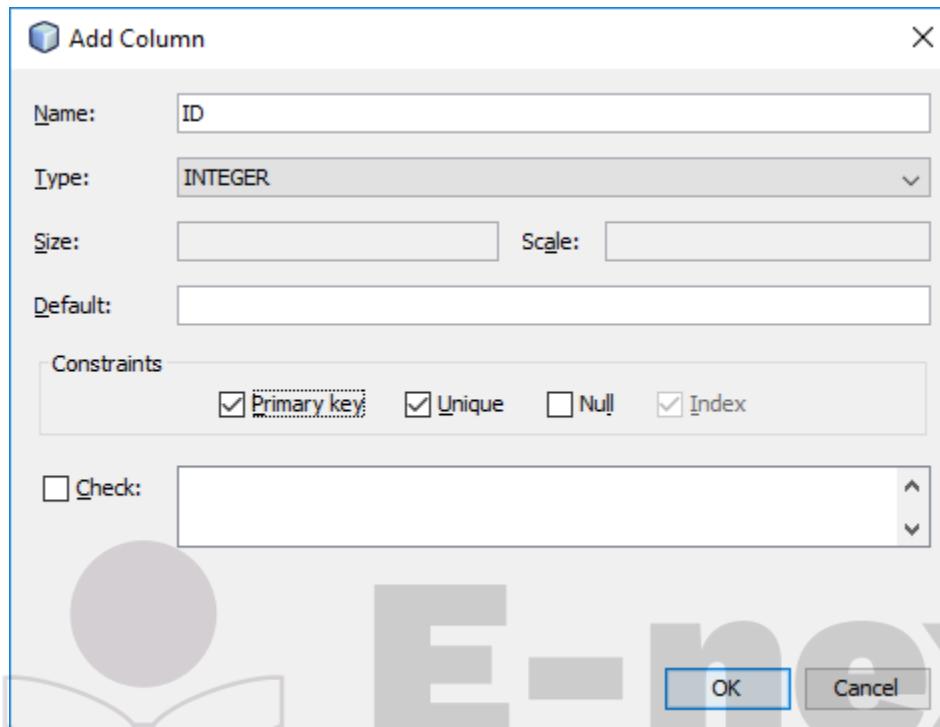


5. Give **table name as FRIENDS**.

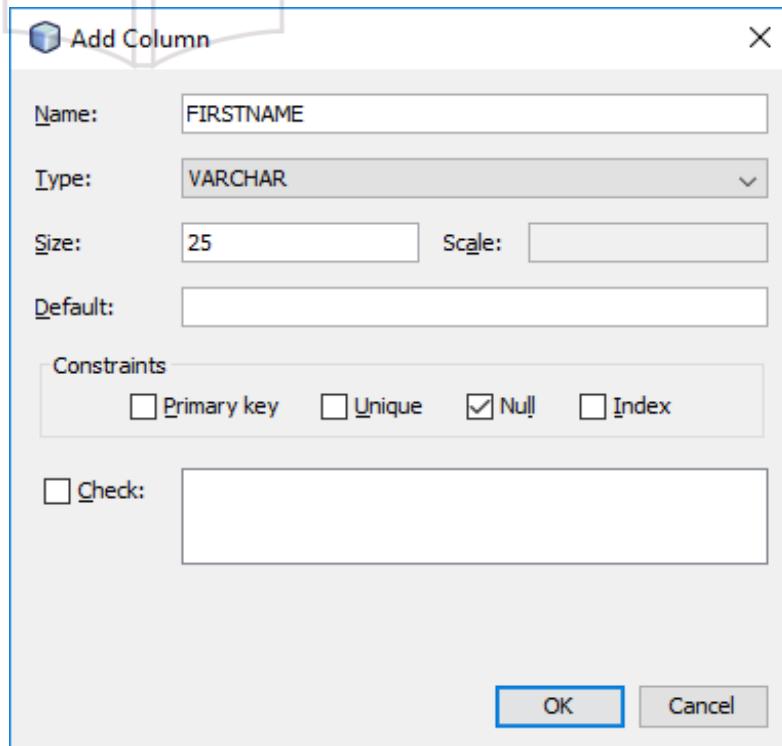


6. Now click on Add column button to add columns in table.

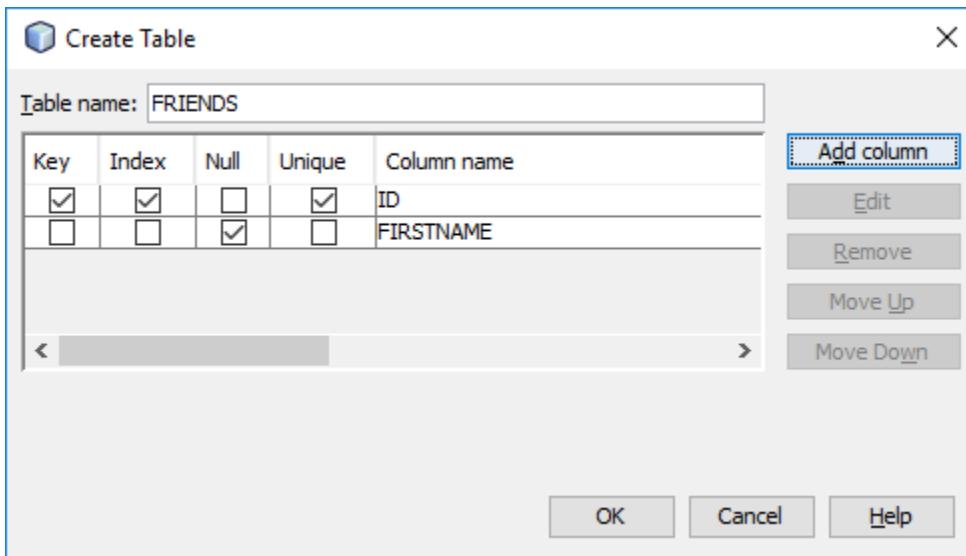
Enter details as in below pic and select Primary key. After that click on OK button.



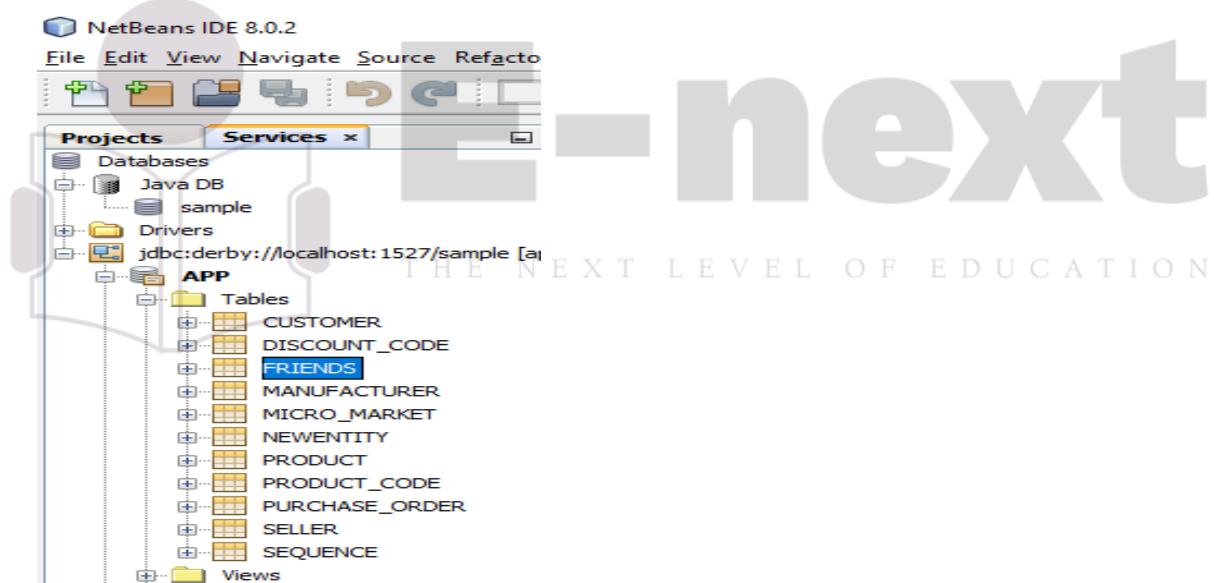
7. Now add second column with following detail. But don't select primary and click on OK button.



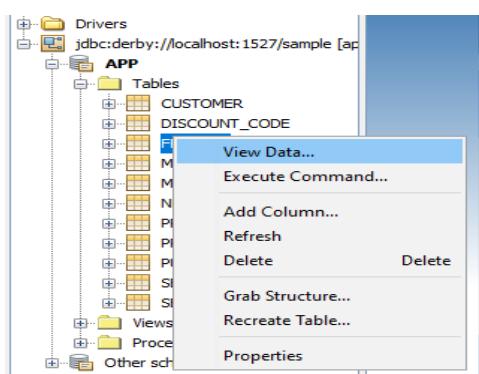
8. Now click on OK button.



9. Now you can see a table with name **FRIENDS** in the table.



10. Right click on FRIENDS to view and add records into it.

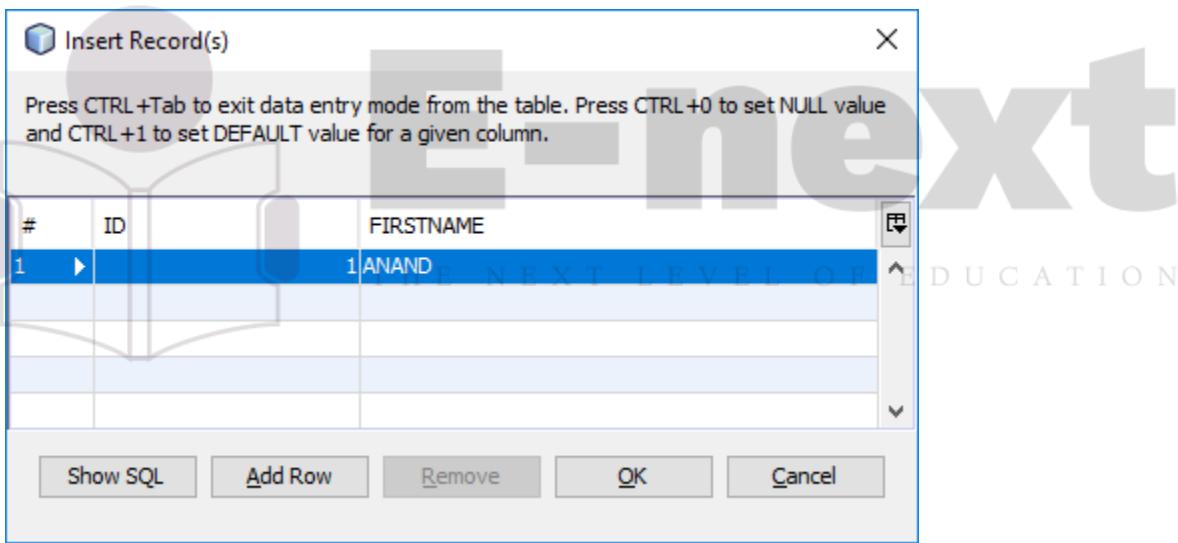


11. Now click on the leftmost icon in second panel to insert some record.

The screenshot shows the SQL Workbench/J application. On the left, the 'Projects' tab is selected, displaying a tree structure with 'Databases', 'Java DB', 'Drivers', and a connection to 'jdbc:derby://localhost:1527/sample [app]'. Under 'APP', there are several tables: CUSTOMER, DISCOUNT_CODE, FRIENDS (which is selected), MANUFACTURER, MICRO_MARKET, NEWENTITY, PRODUCT, PRODUCT_CODE, PURCHASE_ORDER, and SELLER. The right panel has a 'Start Page' tab and an 'SQL 1 [jdbc:derby://localhost:1527/sample [app on APP]]' tab. The SQL editor contains the query 'select * from APP.FRIENDS;'. Below it is a results grid with one row: # 1 and ID 1. At the bottom of the results grid is a button labeled '# Insert Record(s) (Alt+I)'.

12. Insert a record and then click on Add Row button to insert more record.

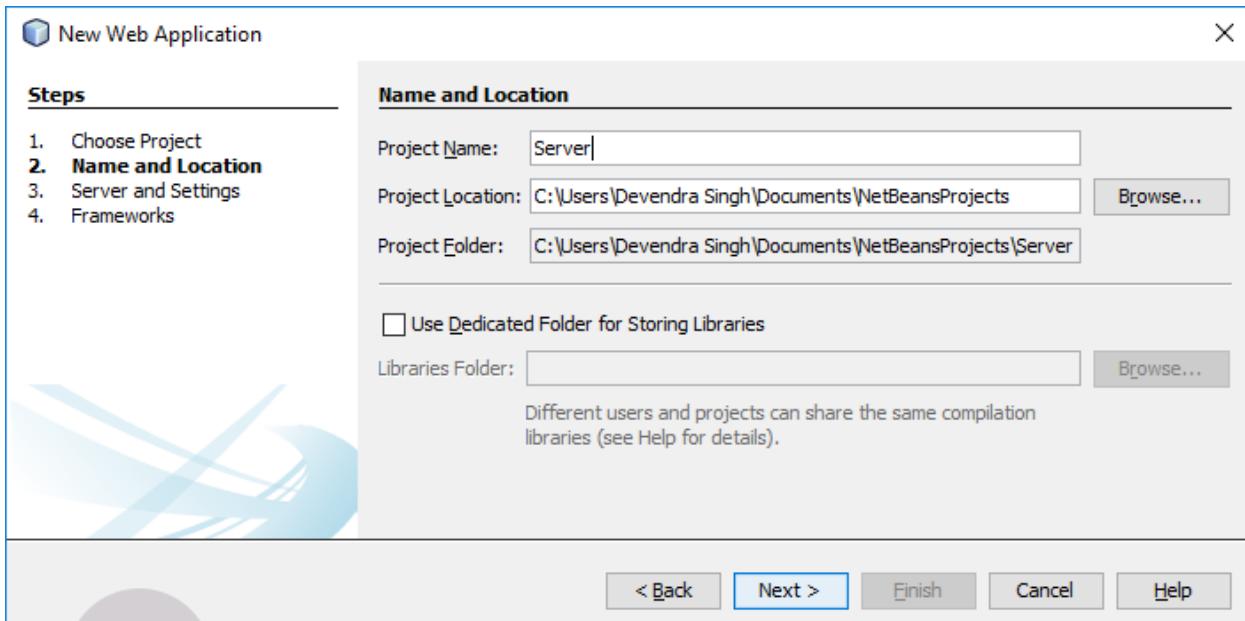
After that click on OK button to finish.



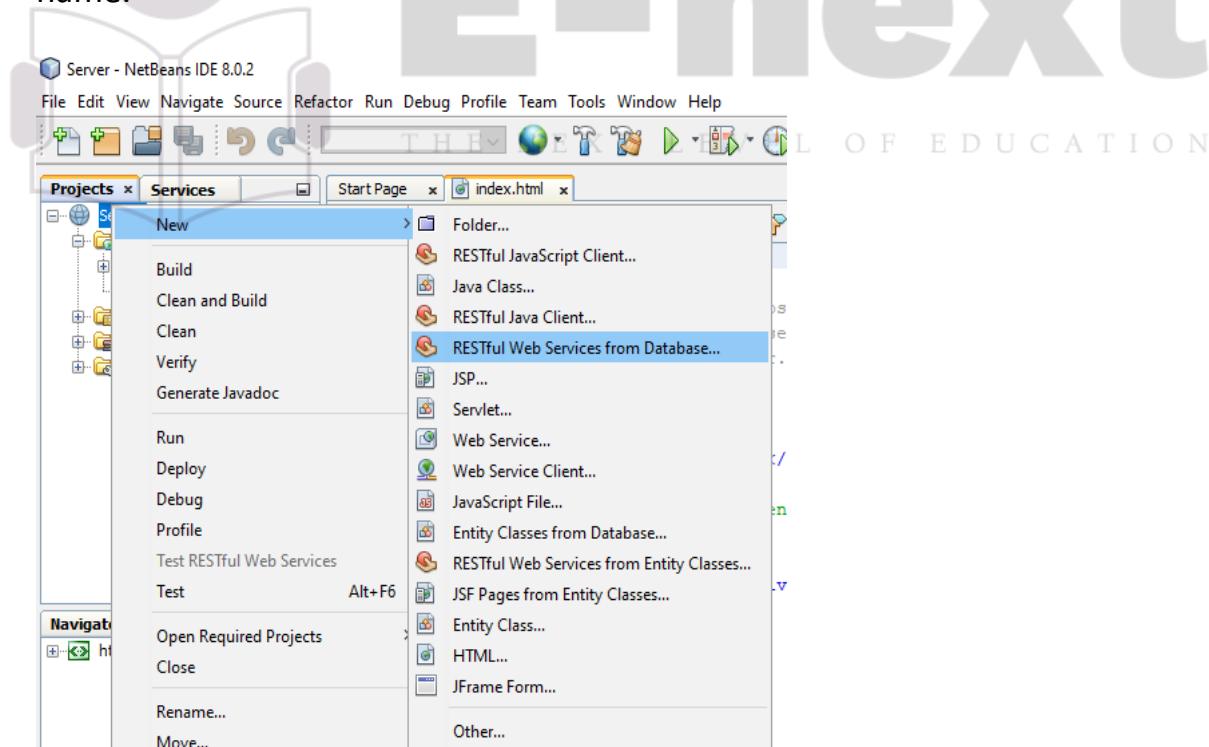
13. As you can see, I have entered 7 records.

The screenshot shows the results of the query 'select * from APP.FRIENDS;'. The results grid has columns '#', 'ID', and 'FIRSTNAME'. It displays 7 rows with the following data:
ID FIRSTNAME
1 1 ANAND
2 2 JULHAS
3 3 NIKHIL
4 4 GAGAN
5 5 RAVI
6 6 DHARMENDRA
7 7 ADARSH

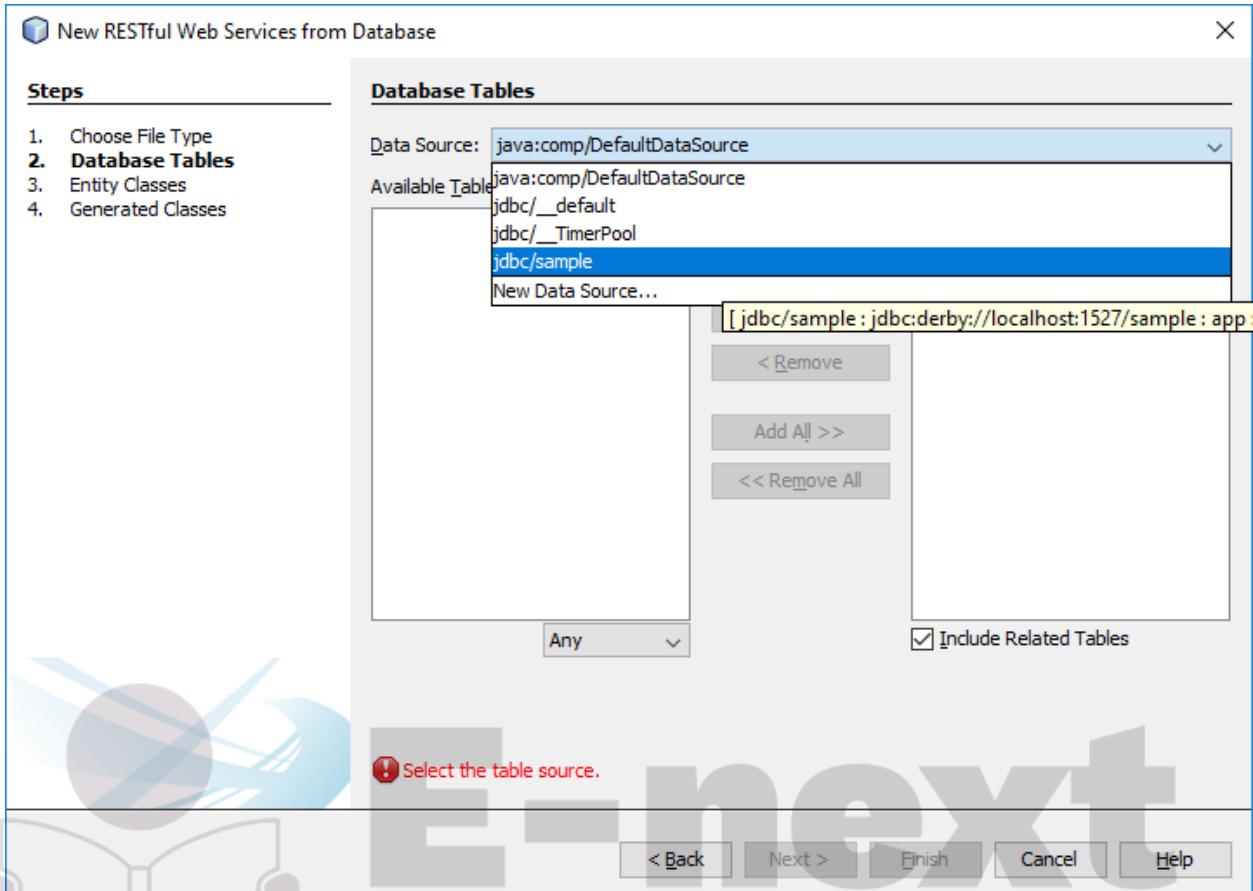
14. Now create a web application with name **Server**. After that click on **Next** and then **Finish** button.



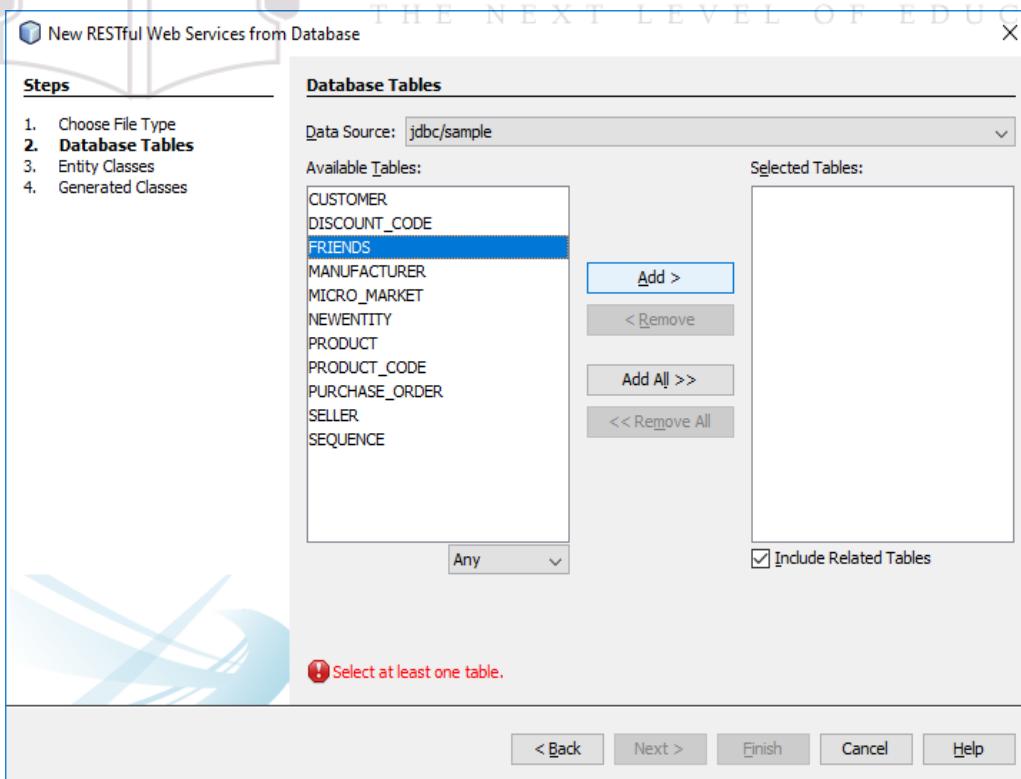
15. Now create a RESTful Web Service from Database by right click on project name.



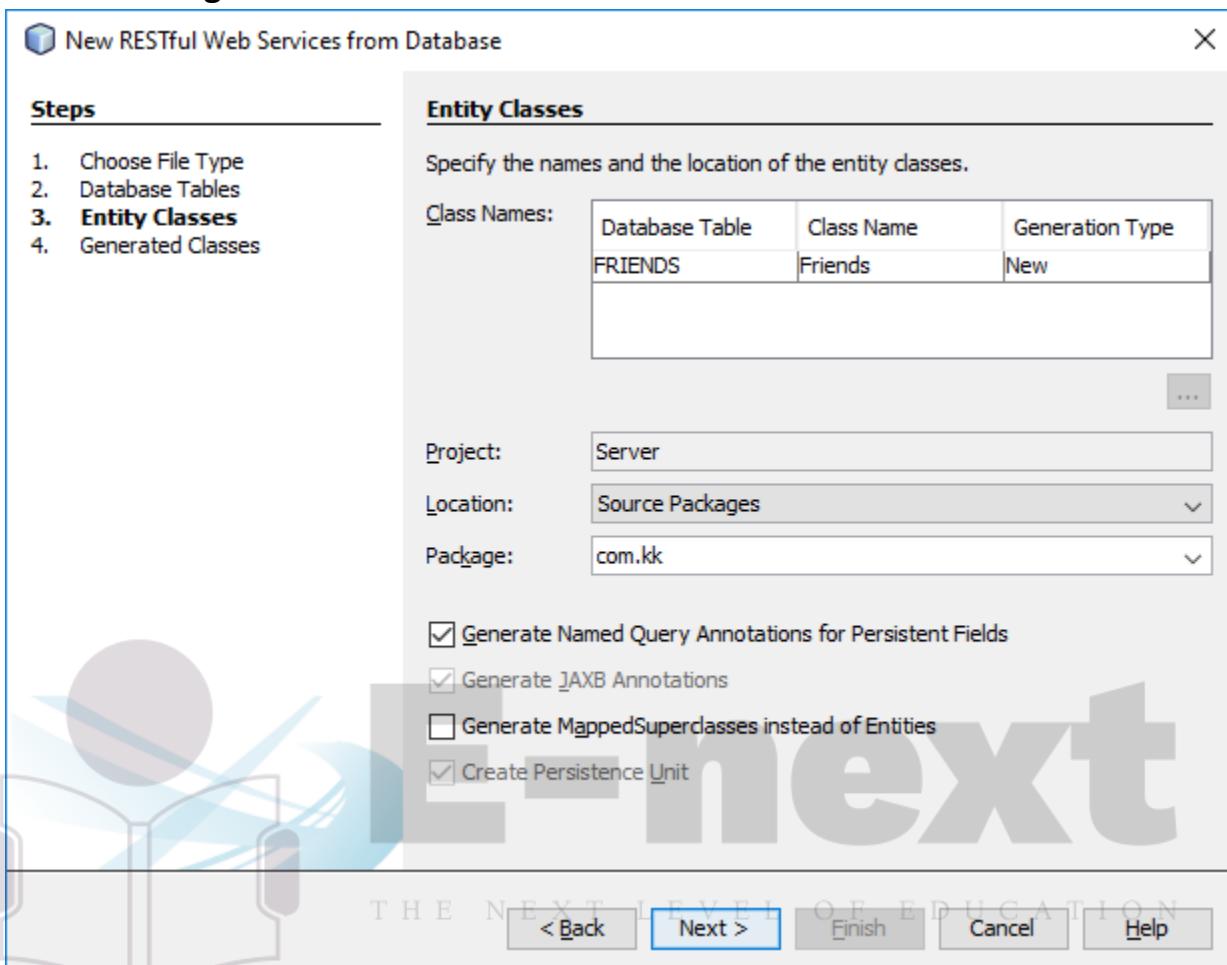
16. Choose Data Source **jdbc/sample**.



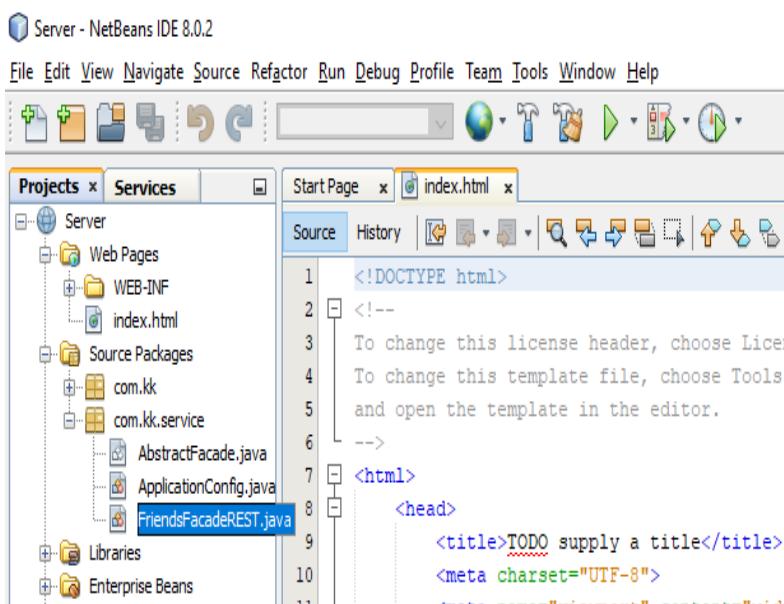
17. Now select FRIENDS and click on Add button. After that click on Next button.



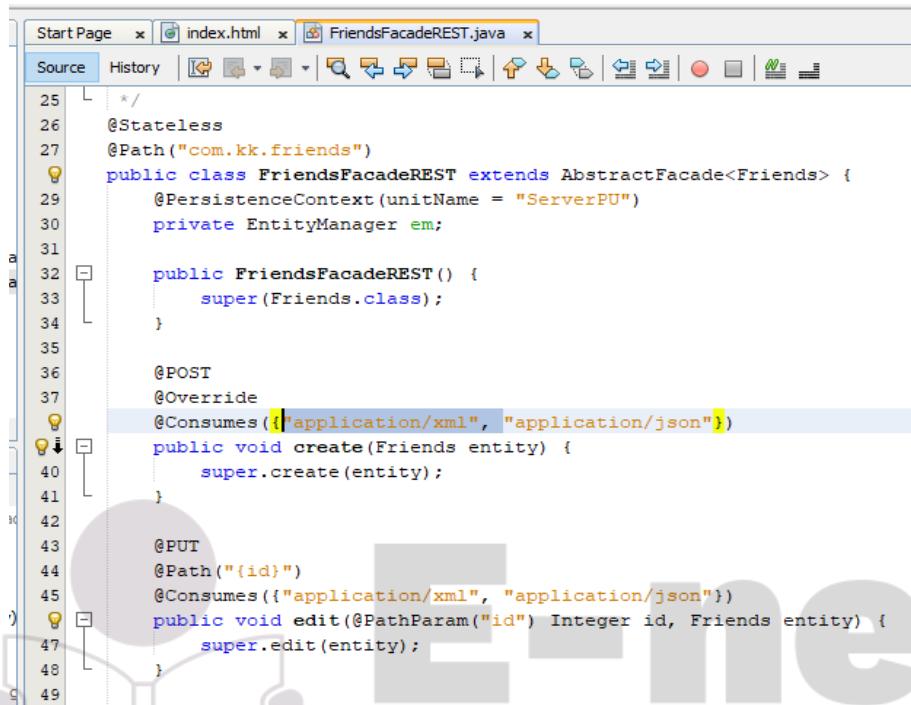
18. Enter Package name as com.kk and click on Next button and then Finish.



19. Now open selected file by double click on it.



20. Now remove the selected part from every method in this file. So that it will communicate only in JSON format. You can also use methods to convert it. But this is easiest method.



The screenshot shows the NetBeans IDE interface with the file 'FriendsFacadeREST.java' open. The code is a Java class for a RESTful service. A specific line of code is highlighted with a yellow background:

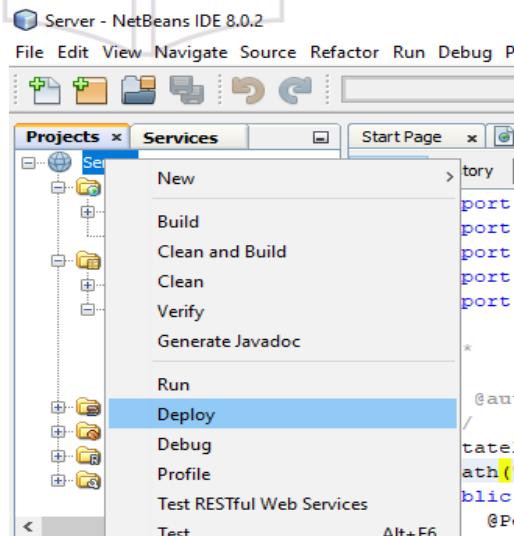
```
    @Consumes({"application/xml", "application/json"})
```

This line is part of a method annotated with @POST and @Override. The method signature is:

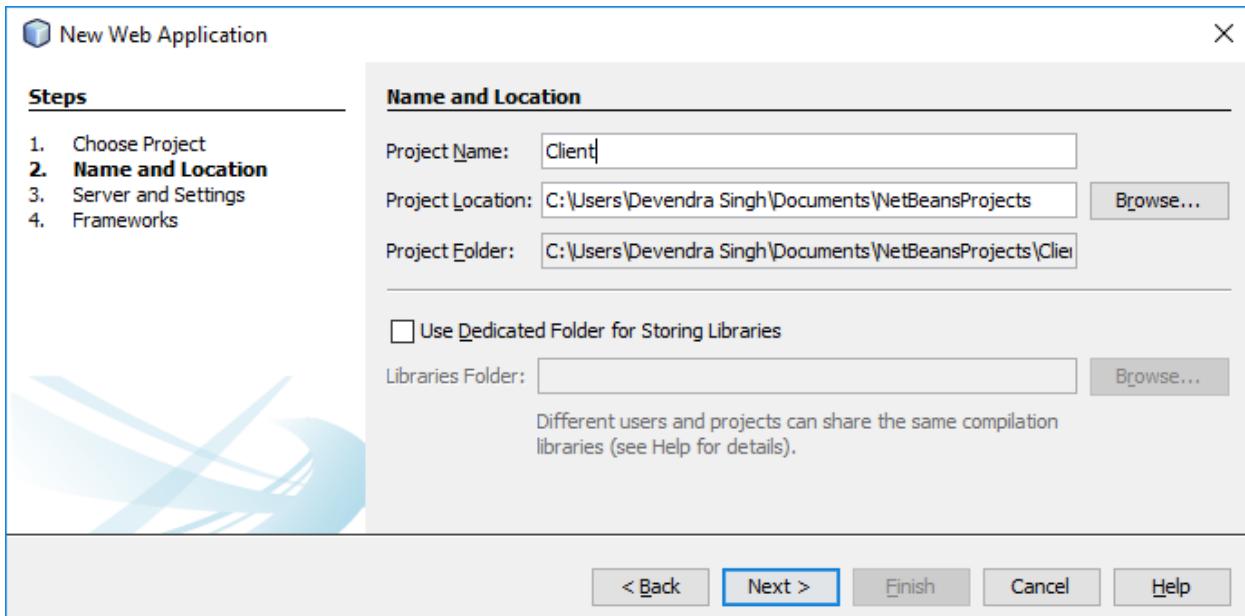
```
public void create(Friends entity) { super.create(entity); }
```

Below this, there is another method annotated with @PUT and @Path("{id}").

21. After that right click on project name and Deploy it.



22. Now create one more Web Application as Client. After that click on Next and then Finish button.



23. Now open the index.html file of Client project and add the following code in between HEAD tag.



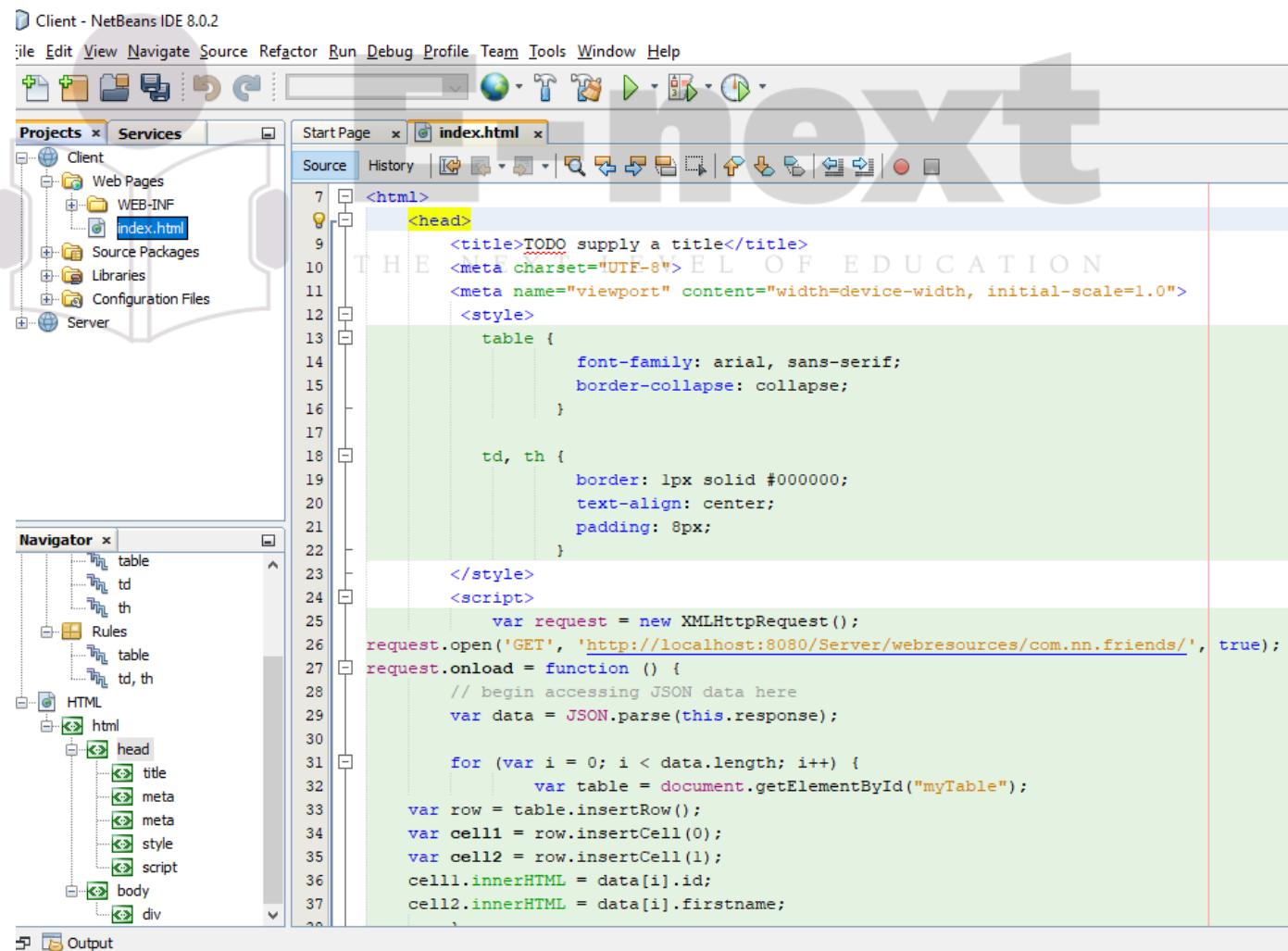
```
<style>
    table {
        font-family: arial, sans-serif;
        border-collapse: collapse;
    }
    td, th {
        border: 1px solid #000000;
        text-align: center;
        padding: 8px;
    }
</style>
<script>
    var request = new XMLHttpRequest();
    request.open('GET',
    'http://localhost:8080/Server/webresources/com.kk.friends/', true);
    request.onload = function () {
        // begin accessing JSON data here
        var data = JSON.parse(this.response);
    }
</script>
```

```

for (var i = 0; i < data.length; i++) {
    var table = document.getElementById("myTable");
    var row = table.insertRow();
    var cell1 = row.insertCell(0);
    var cell2 = row.insertCell(1);
    cell1.innerHTML = data[i].id;
    cell2.innerHTML = data[i].firstname;
}
};

request.send();
</script>

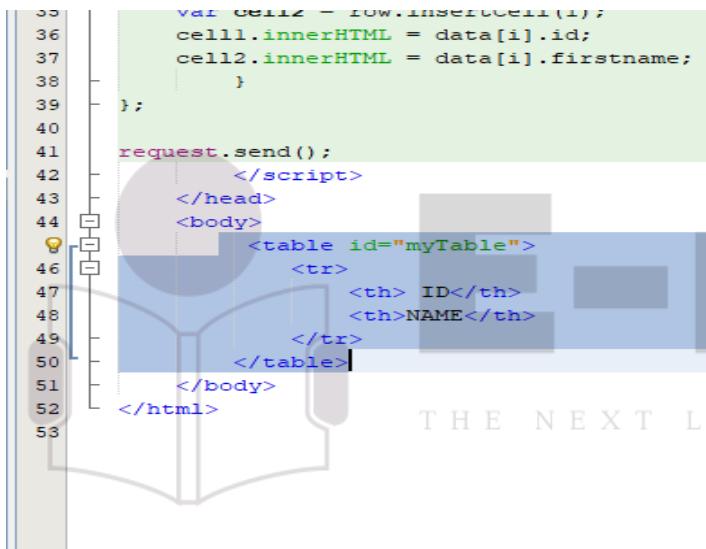
```



URL in the red font is sensitive. It will change accordingly if you have not given the project and file names as of mine.

24. Replace the content of body tag with following code.

```
<table id="myTable">
    <tr>
        <th> ID</th>
        <th>NAME</th>
    </tr>
</table>
```



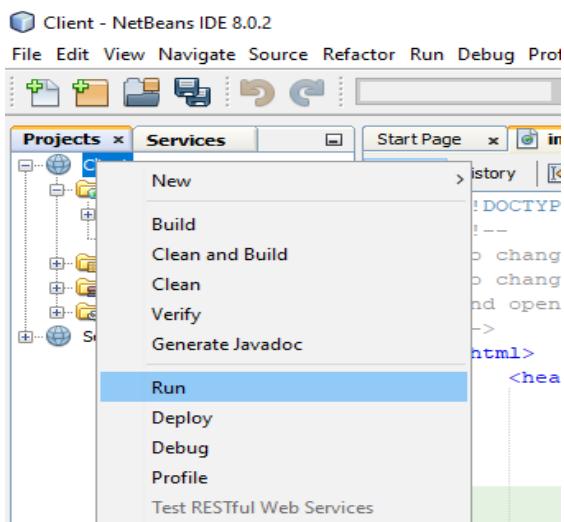
```

35     var cell2 = row.insertCell(1);
36     cell1.innerHTML = data[i].id;
37     cell2.innerHTML = data[i].firstname;
38   }
39 };
40
41 request.send();
42 </script>
43 </head>
44 <body>
45   <table id="myTable">
46     <tr>
47       <th> ID</th>
48       <th>NAME</th>
49     </tr>
50   </table>
51 </body>
52 </html>
53

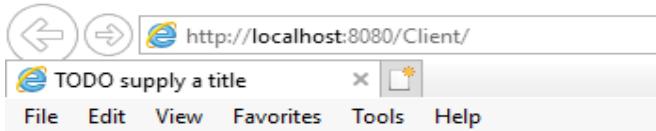
```

THE NEXT LEVEL OF EDUCATION

25. Now run the Client Web Application.



26. A window will open in browser which represents a data in tabular format.
These **data are the records entered in FRIEND table.**



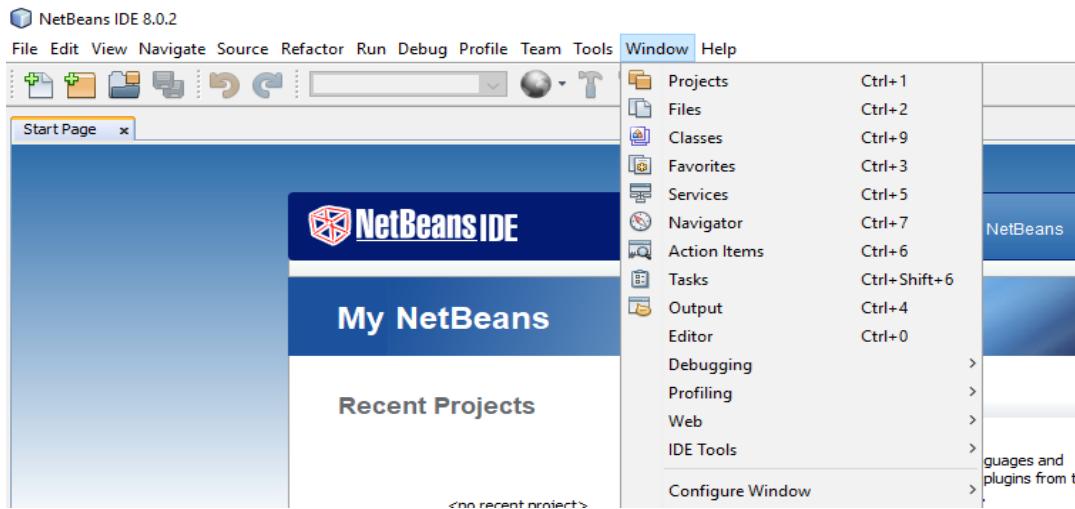
ID	NAME
1	ANAND
2	JULHAS
3	NIKHIL
4	GAGAN
5	RAVI
6	DHARMENDRA
7	ADARSH



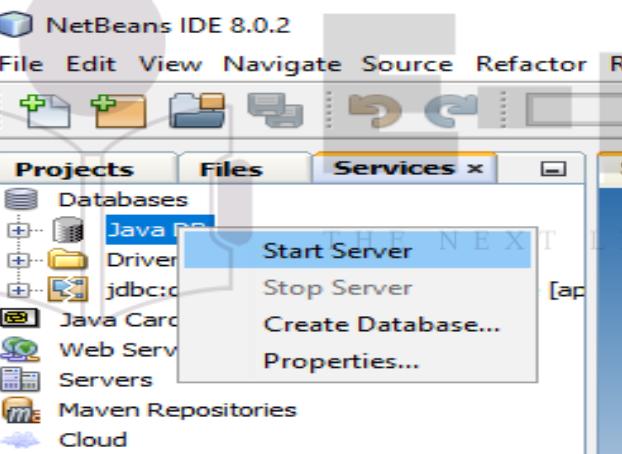
E-next

THE NEXT LEVEL OF EDUCATION

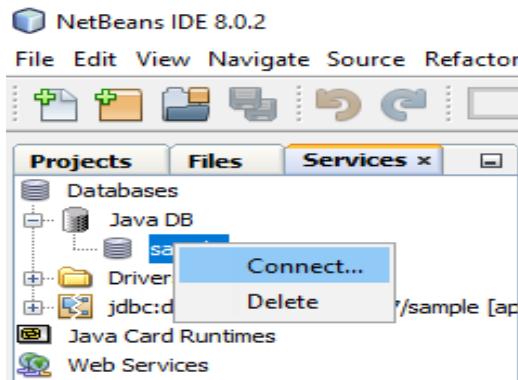
1. Click on Window menu and click on **Projects, Files & Services** to open it.



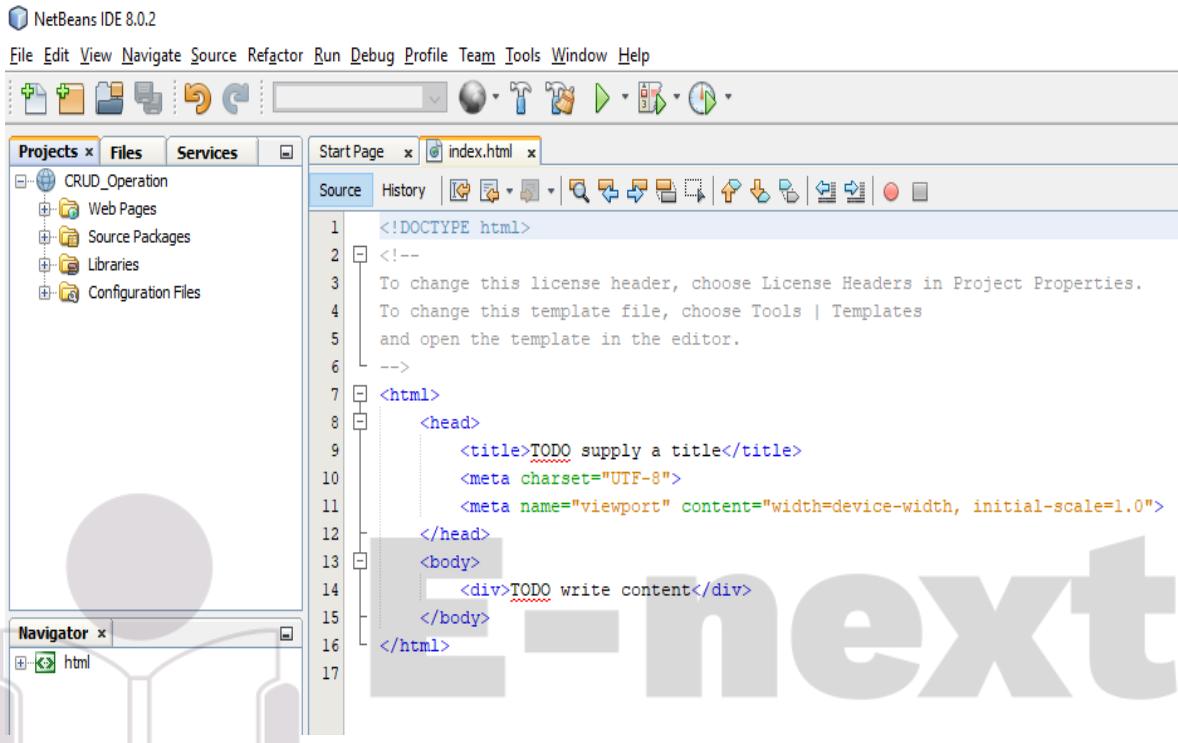
2. Right click on Java DB and then click on Start Server to start the server .



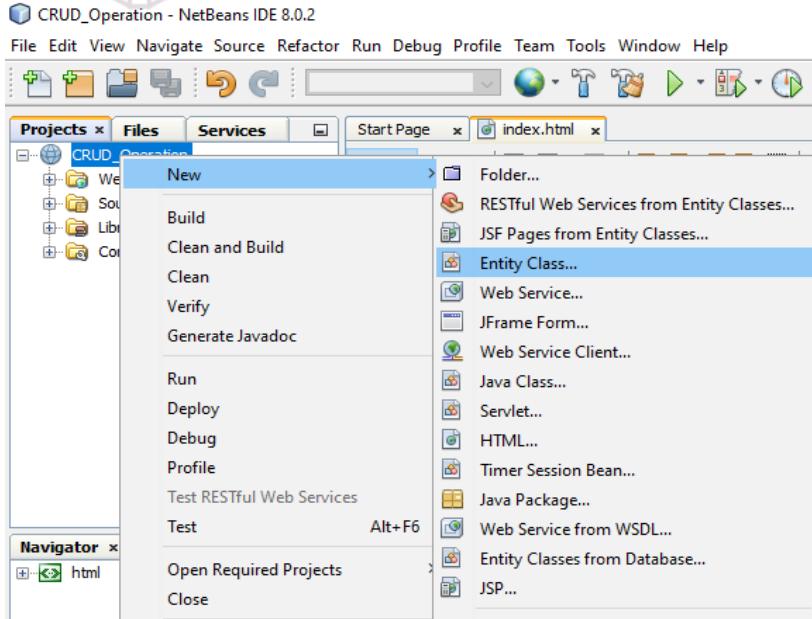
3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.



4. Now create a web application with the name **CRUD_Operation**. A window will open like following pic.



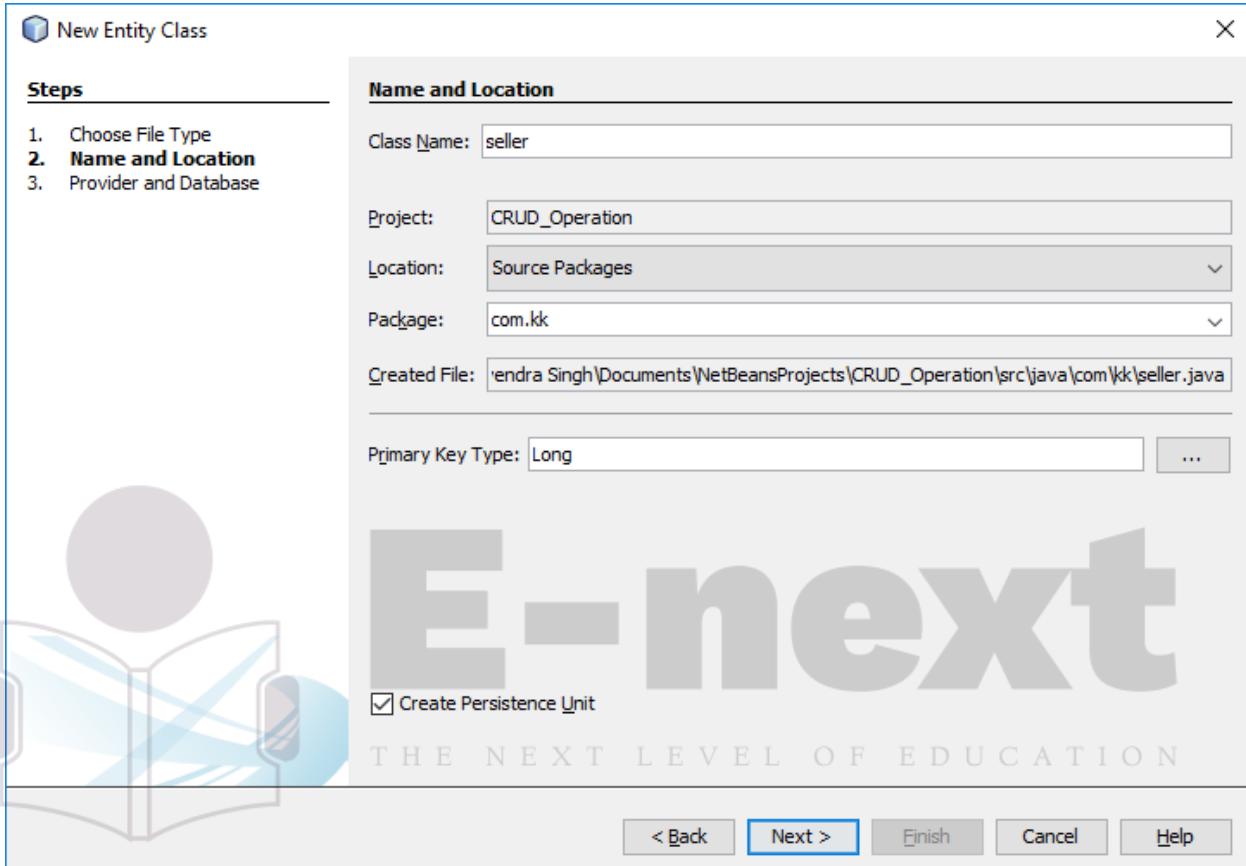
5. Create an entity class. Right click on project name -> New -> Entity Class.



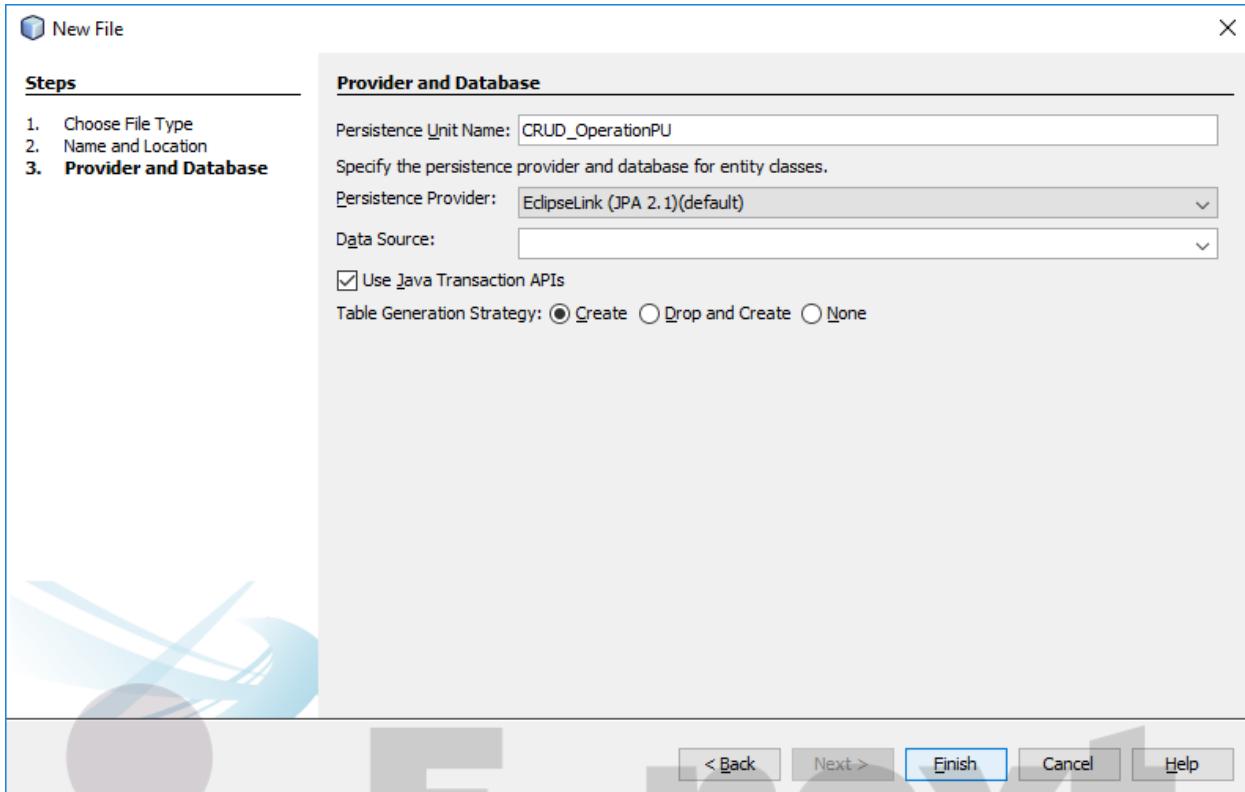
6. A window will appear like bellow pic. Enter following data and click on Next

Class Name -> seller

Package Name -> com.kk

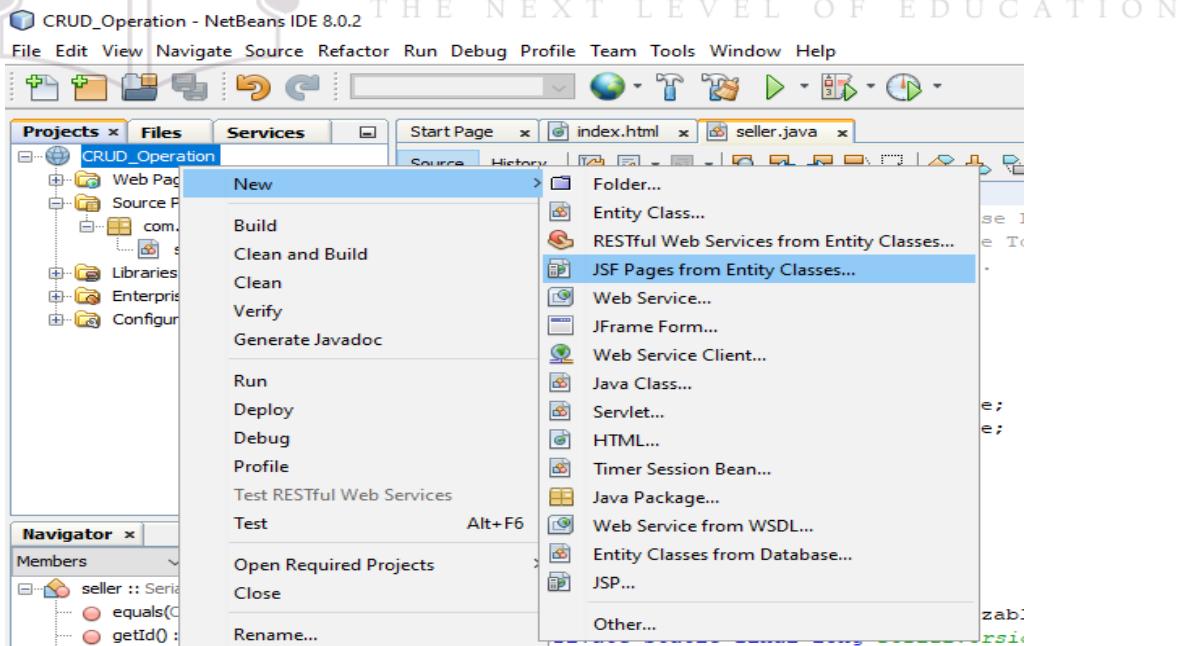


7. Click on Finish.

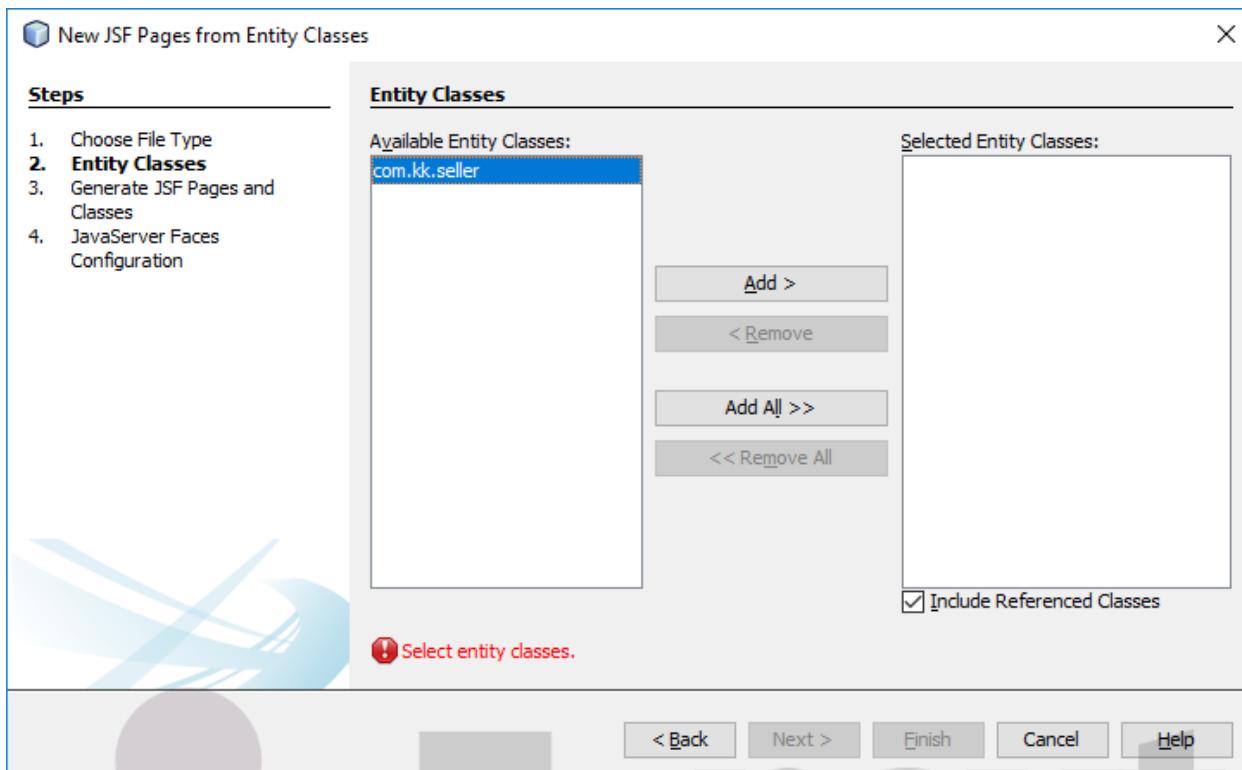


8. Right click on project name and create JSF Pages from Entity Classes.

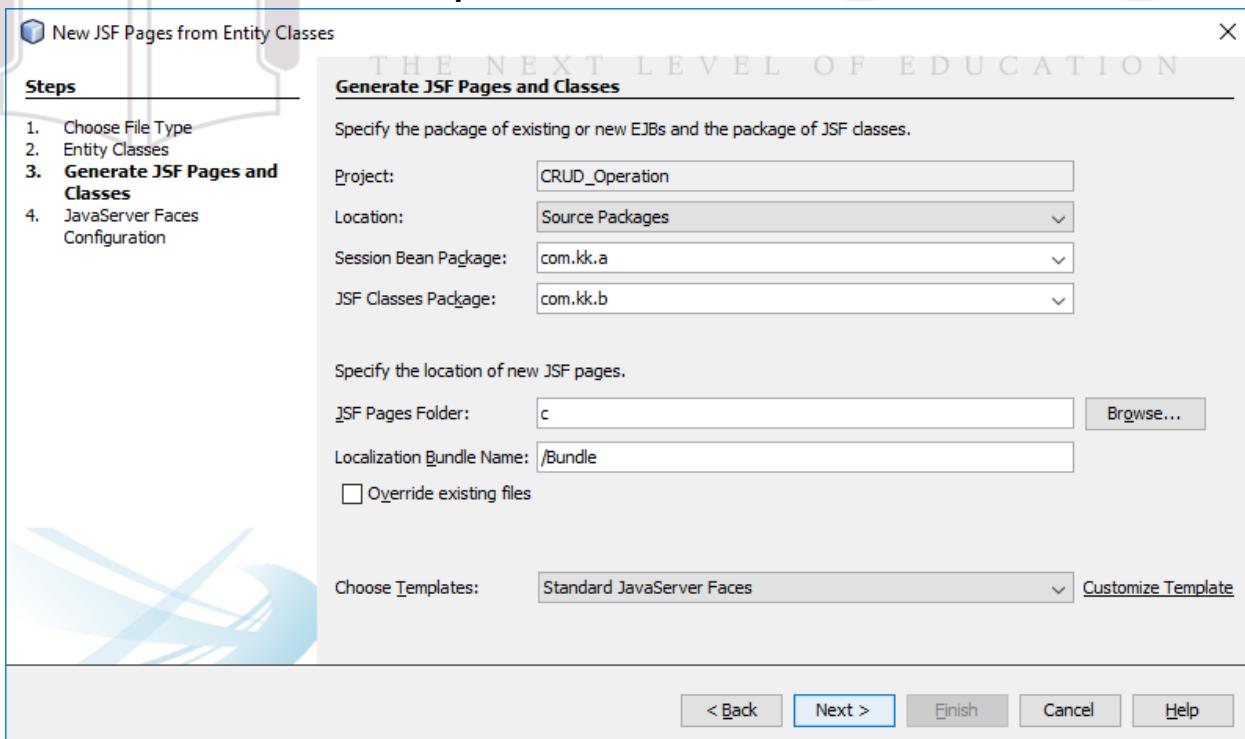
Right click on project name -> New -> JSF Pages from Entity Classes



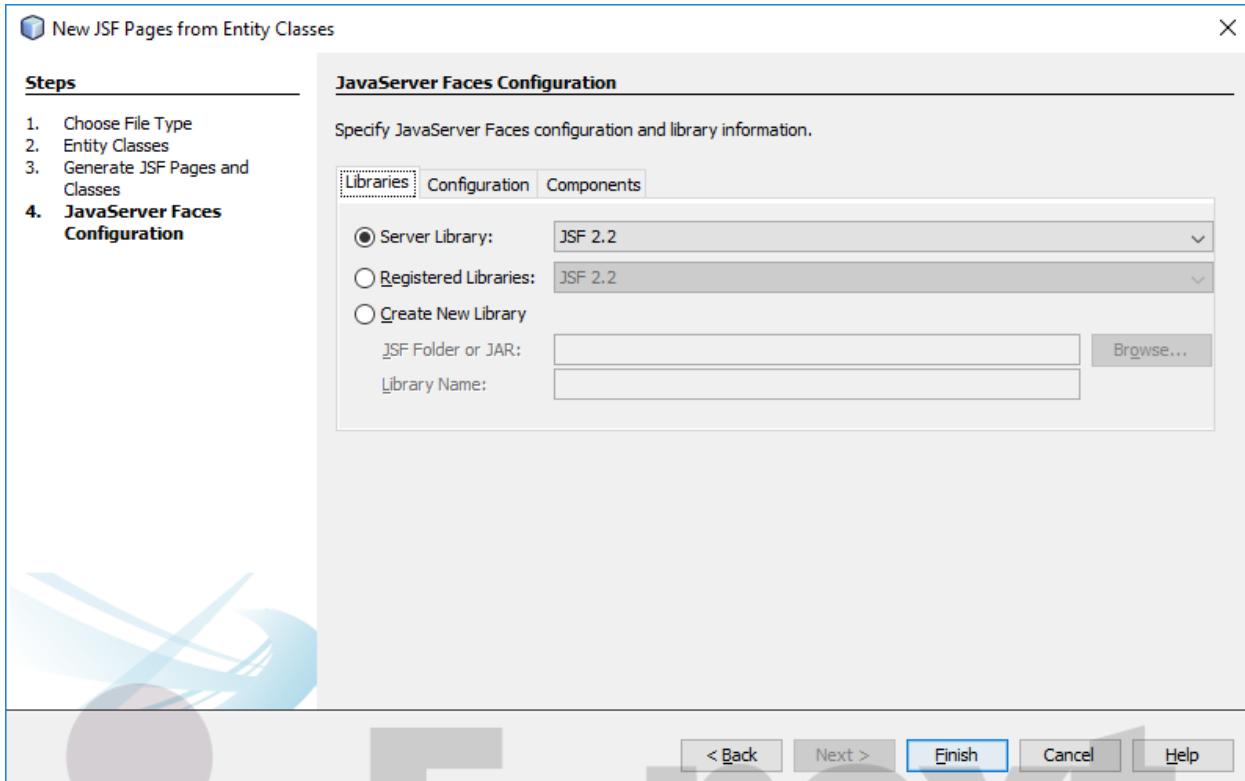
9. Select com.kk.seller and click on Add button and then Next button on below.



10. A window like below will appear on the screen. Enter the data into that window as entered in below pic and click on Next button.

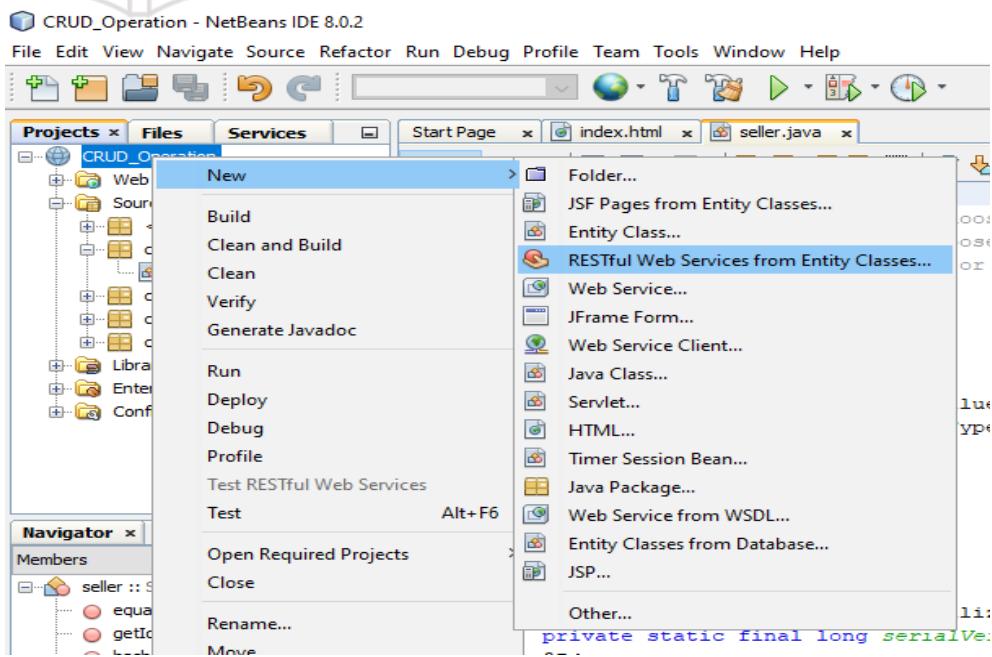


11. Now click on Finish.

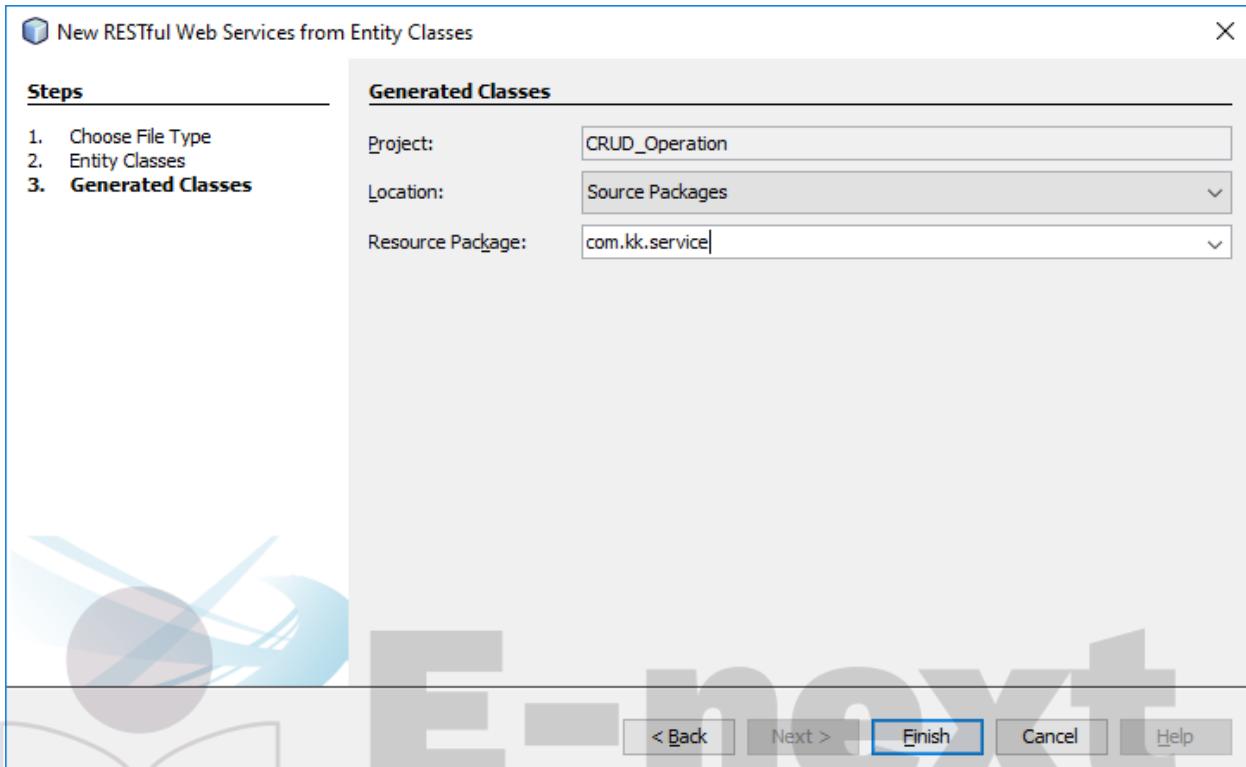


12. Right click on project name and create RESTful Web Services from Entity Classes.

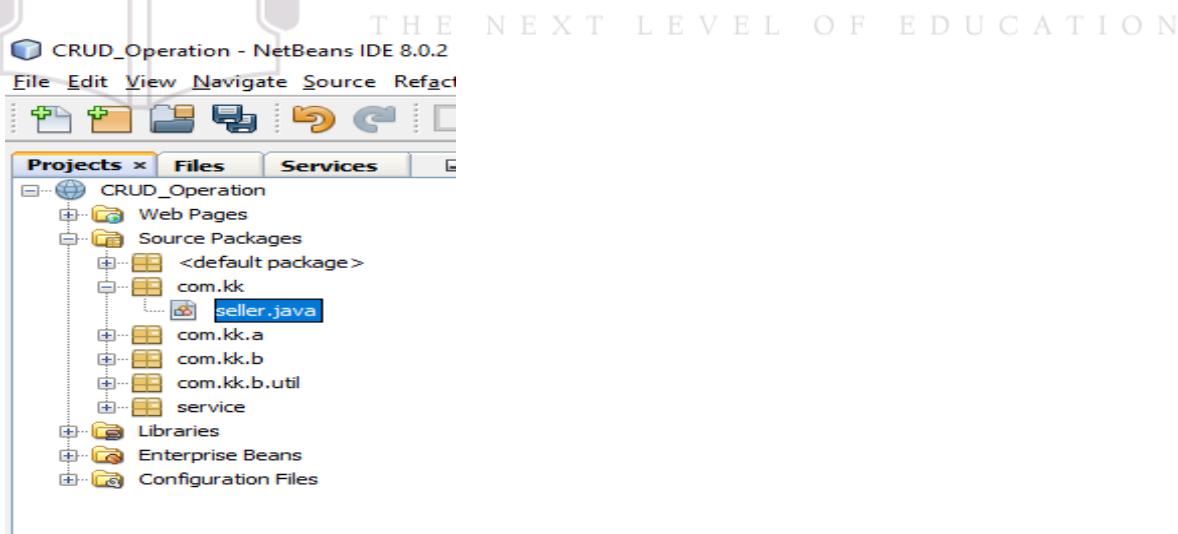
Right click on project name -> New -> RESTful Web Services from Entity Classes



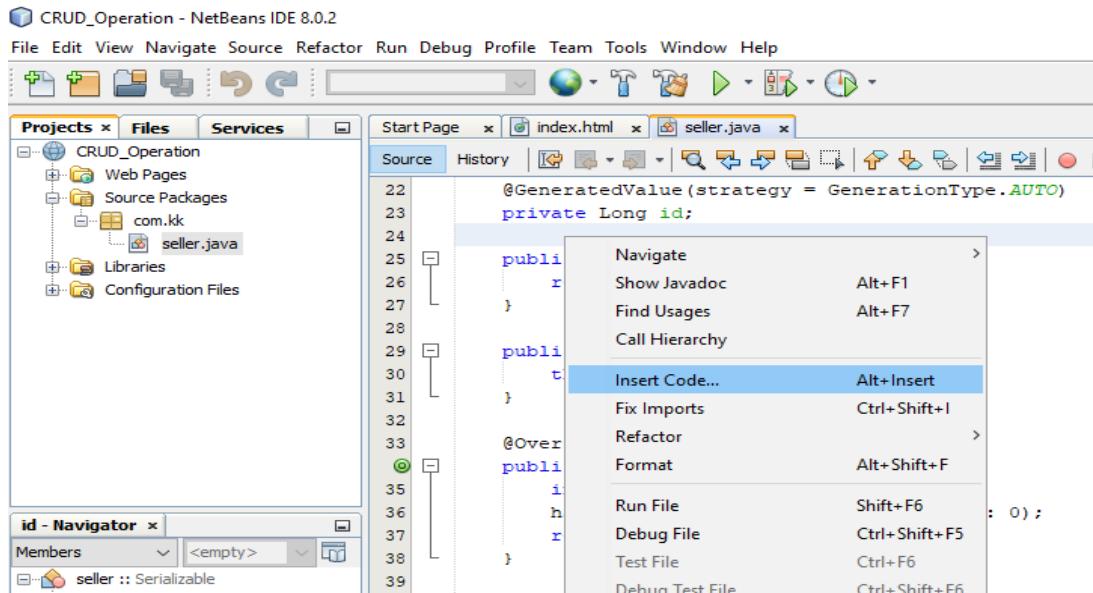
13. Repeat step 9 and then it will go on next page. Then enter the com.kk.service in Resource Package and then click on Finish button.



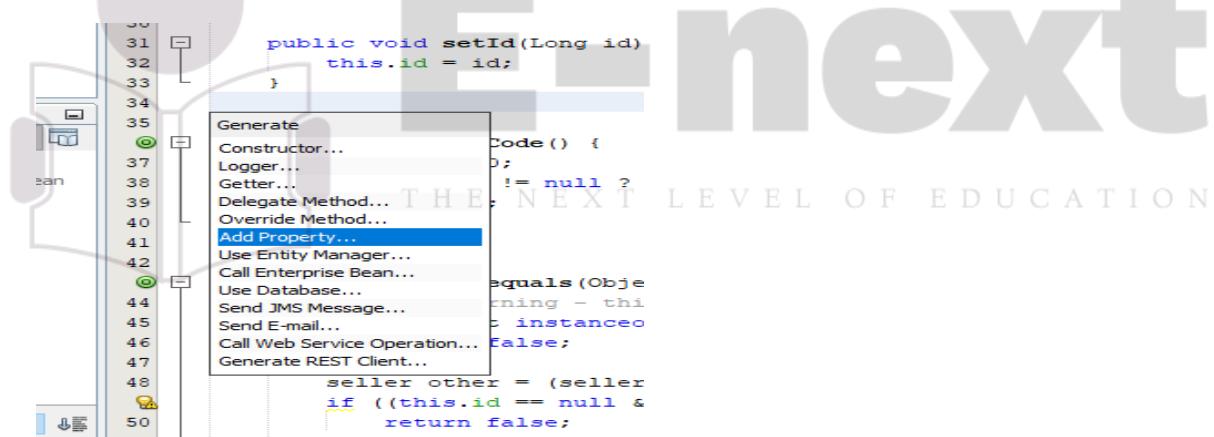
14. Now open seller.java file under com.kk package.



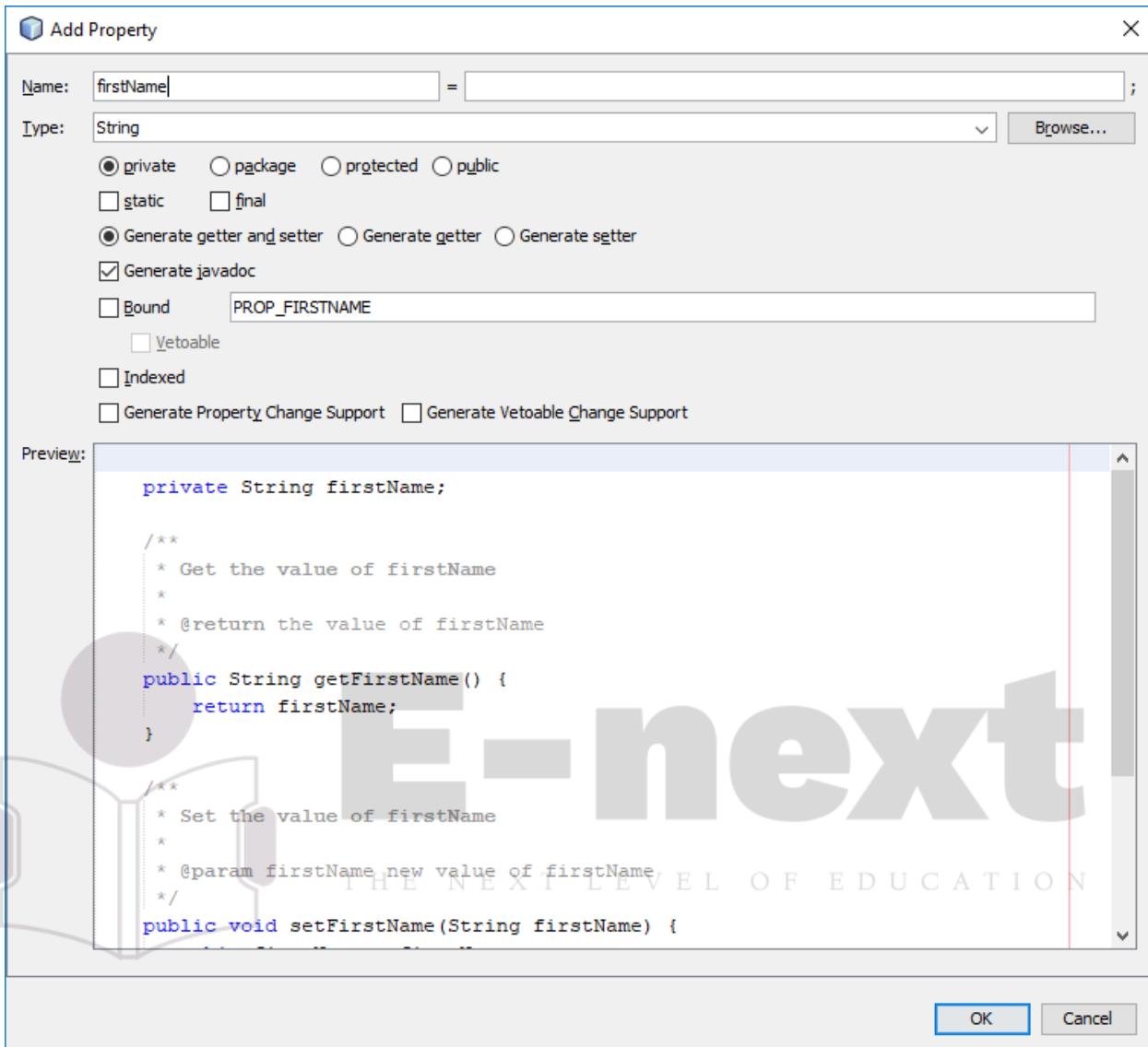
15. In this file at line number 24, do the right click and select Insert Code.



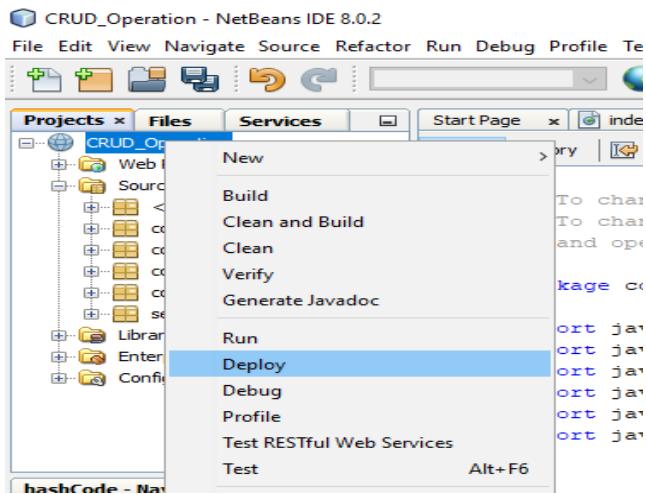
16. A new list will appear. Click on Add Property.



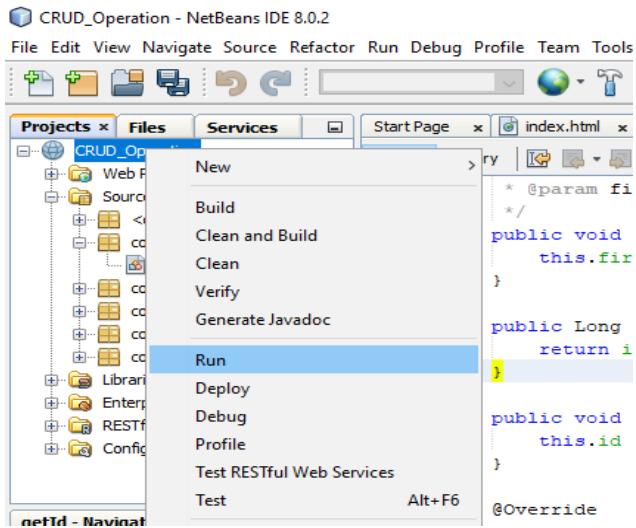
17. A new window will open. Enter name as **firstName**. Make sure name should be exact same as of mine and then click on **OK** button. Actually we are setting getter and setter method for firstName.



18. Now right click on web application name and Deploy it.



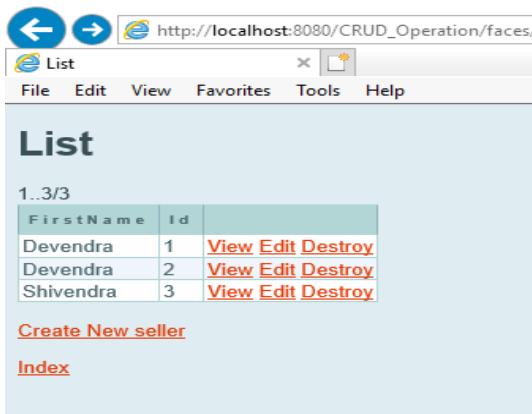
19. Now right click on project name and run it.



20. A window will open in browser like below....



21. Now click on Show All sellers Items for CRUD operation.



22. As I have added three data into database. You can add more data by click on Create New seller and can view, edit and delete by click on View, Edit and Destroy option.

FirstName	Id	
Devendra	1	View Edit Destroy
Devendra	2	View Edit Destroy
Shivendra	3	View Edit Destroy

[Create New seller](#)
[Index](#)

23. Adding one more data into database for demo. Just click on Create New seller. Enter a name into FirstName and id into Id. Now click on Save option to save the data.

Create New seller

FirstName:

Id:

[Save](#)
[Show All seller Items](#)
[Index](#)

next
THE NEXT LEVEL OF EDUCATION

24. Now click on Show All seller Items to view all records whether our data is entered or not. We can see that one more Shivendra name is appearing.

|  List × | 

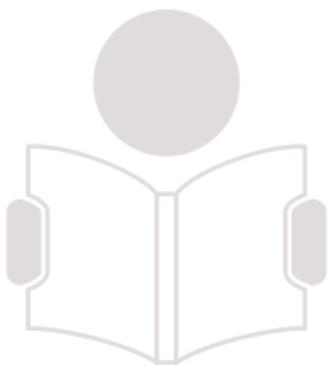
File Edit View Favorites Tools Help

List

1..4/4

FirstName	Id	
Devendra	1	View Edit Destroy
Devendra	2	View Edit Destroy
Shivendra	3	View Edit Destroy
Shivendra	4	View Edit Destroy

[Create New seller](#)



E-next
THE NEXT LEVEL OF EDUCATION