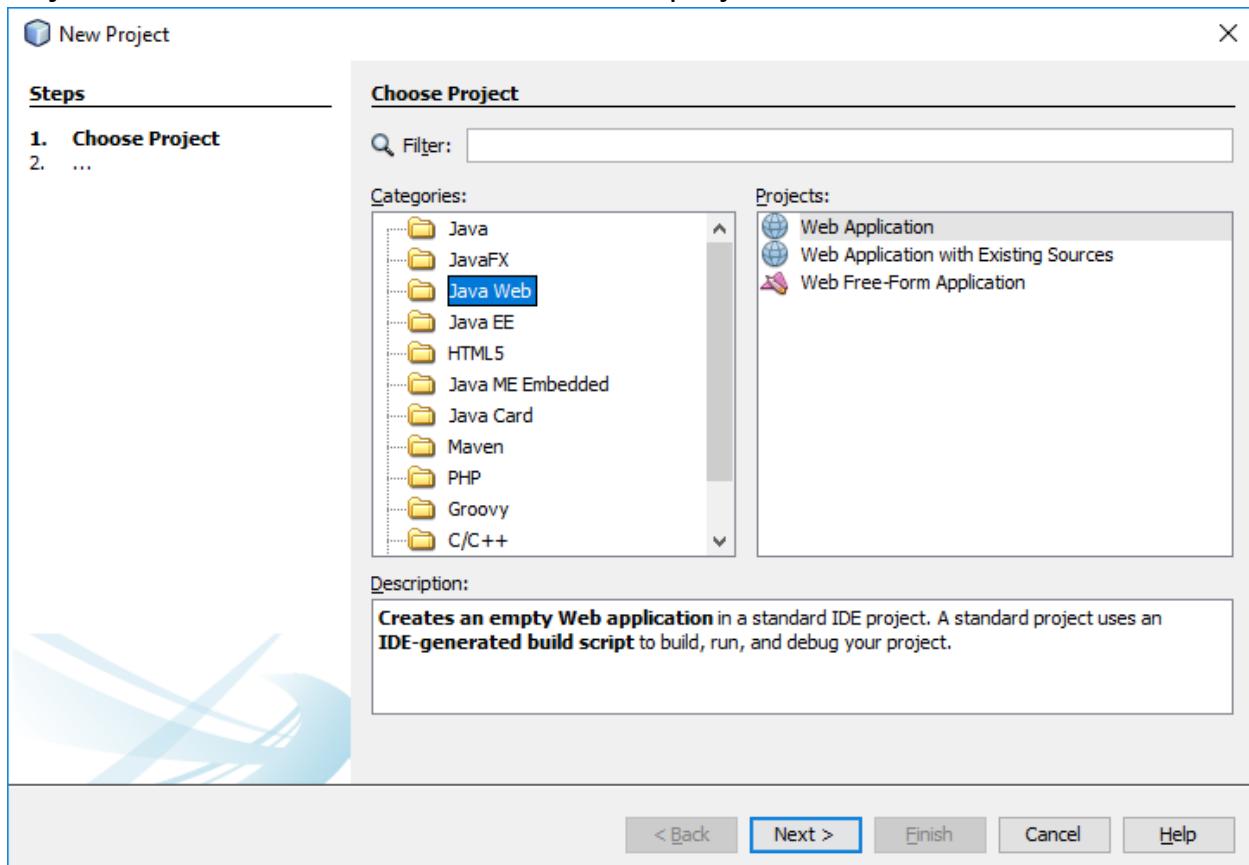
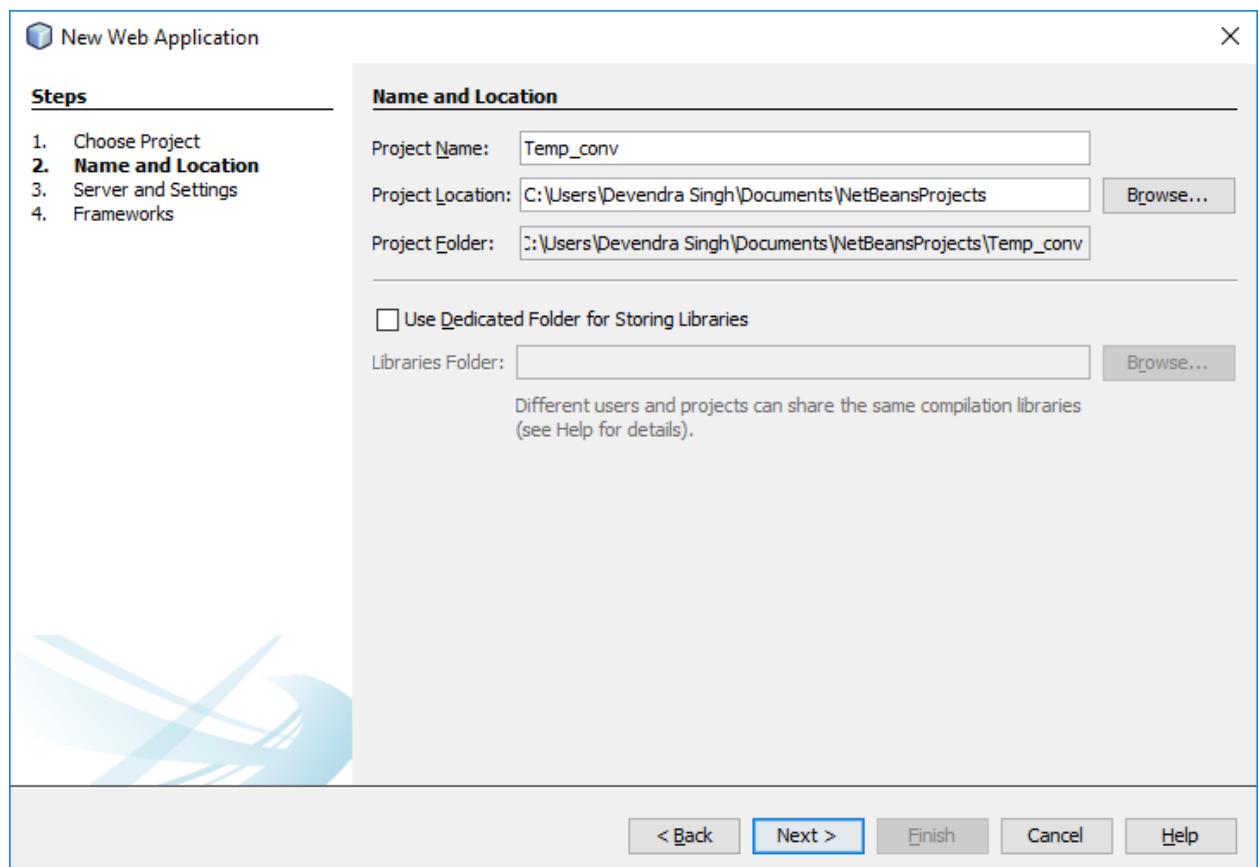


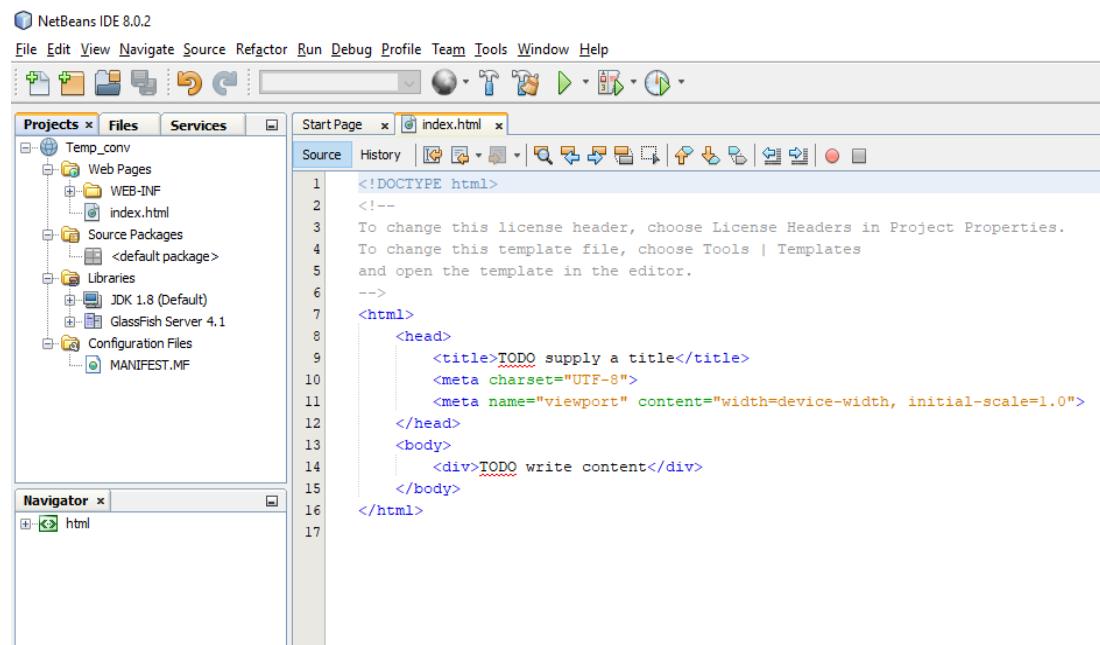
1. Go to **File -> New Project**. Select **Java Web** in categories and **Web Application** in Projects. Click on **Next** to create web based project.



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

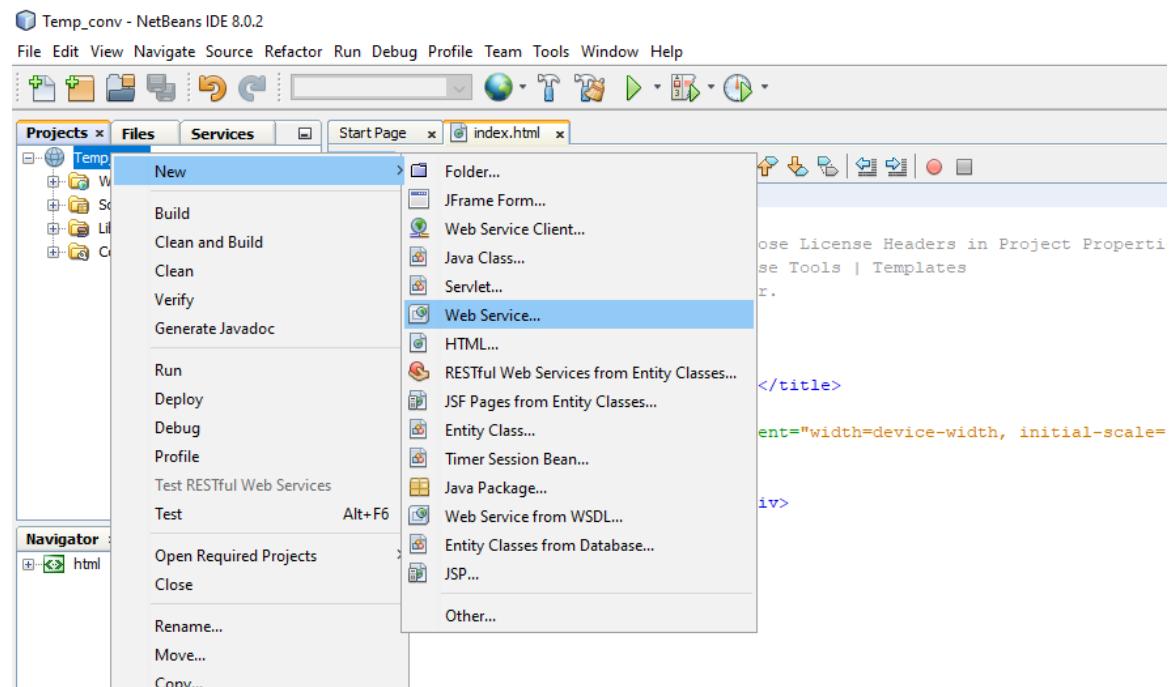


3. After completion of project creation process a window will appear like below.

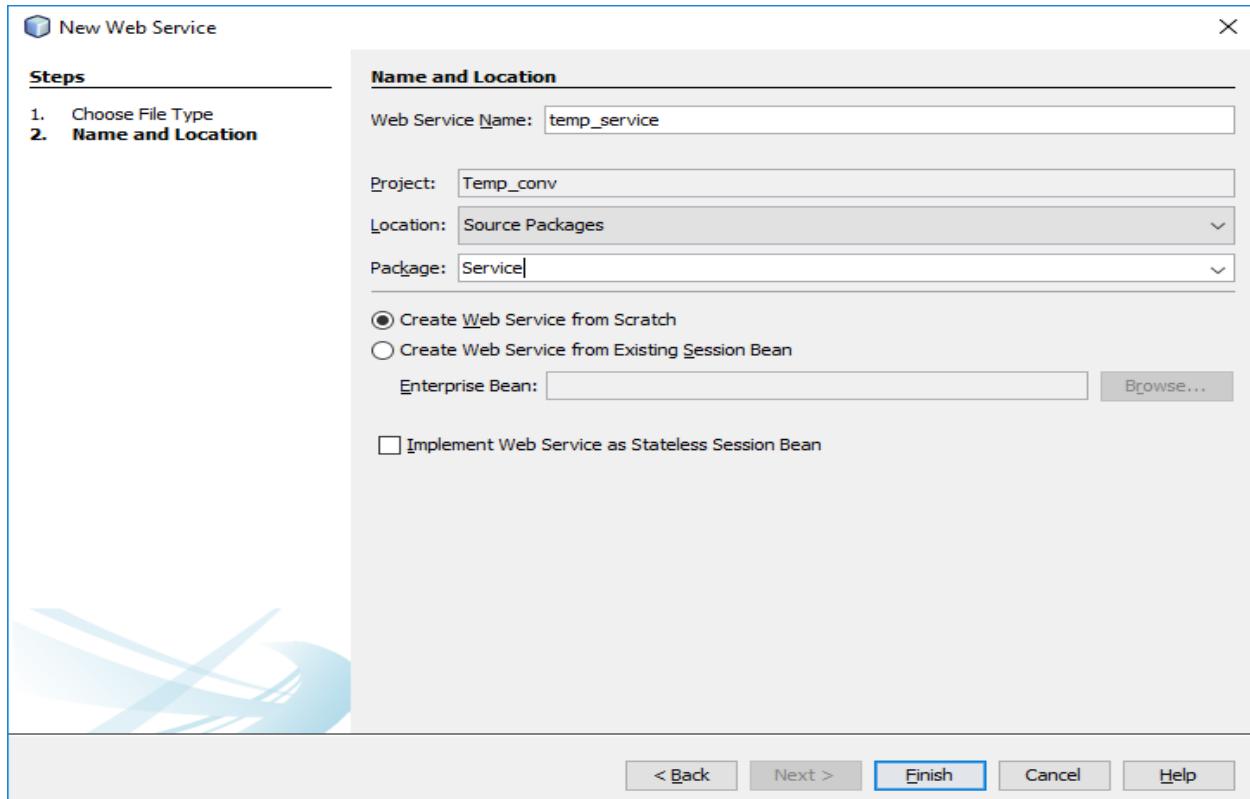


4. Create web service.

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.

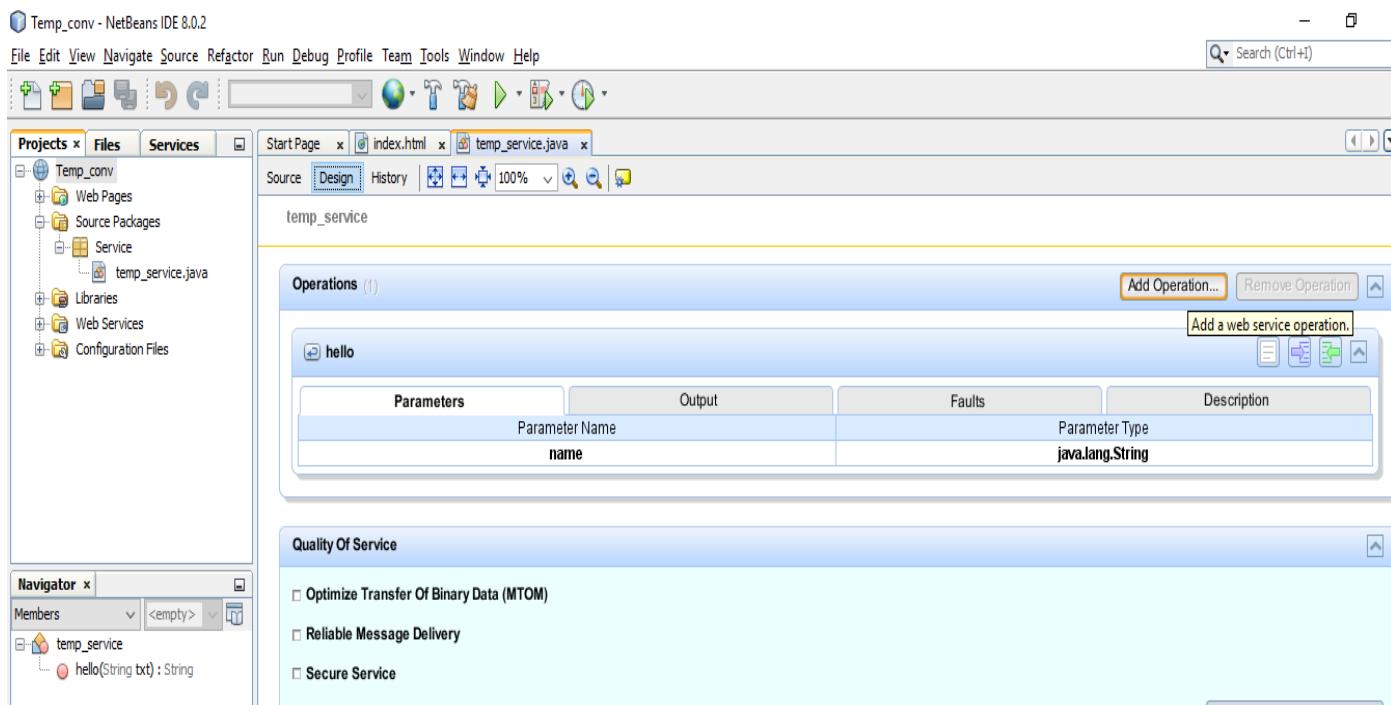
Go to design mode by click on **Design**.

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

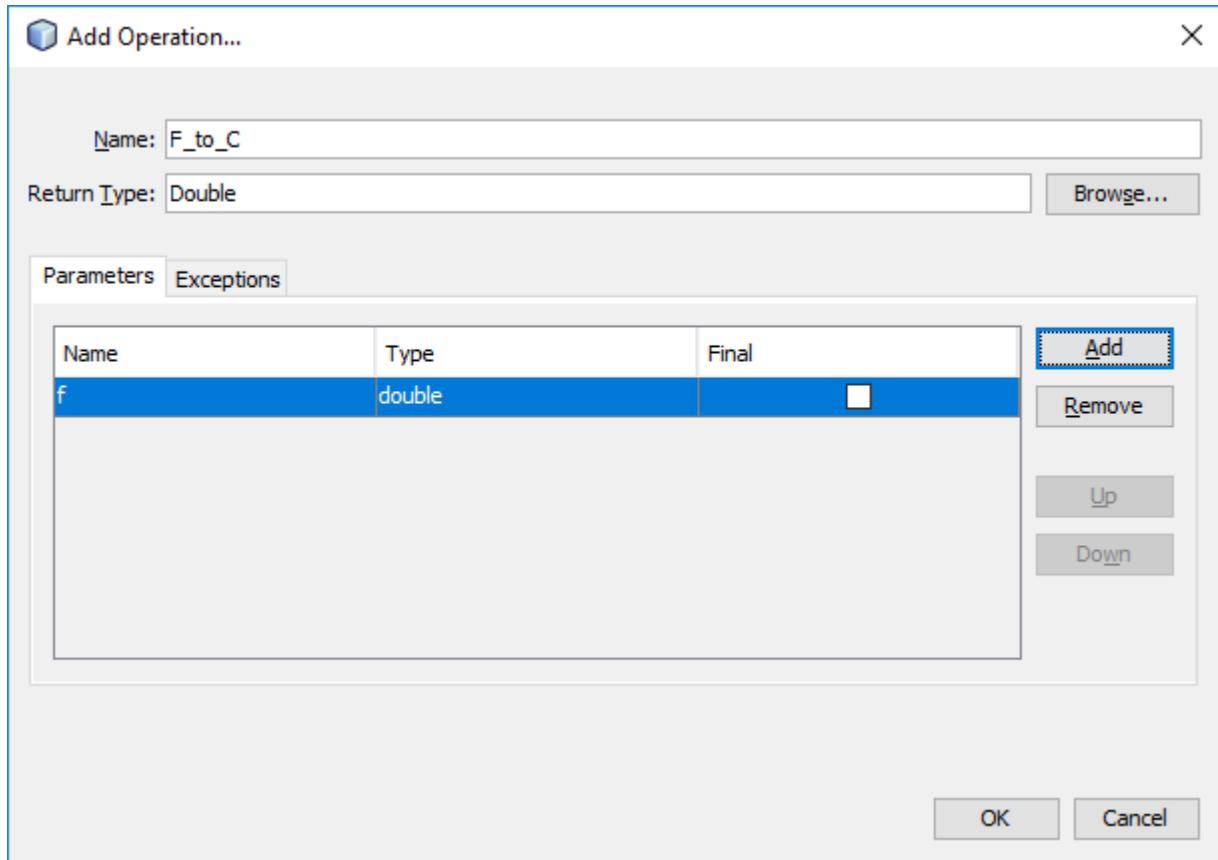
- Title Bar:** Temp_conv - NetBeans IDE 8.0.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations, search, and run.
- Projects Tab:** Shows the project structure under 'Temp_conv': Web Pages, Source Packages (with a Service folder containing temp_service.java), Libraries, Web Services, and Configuration Files.
- Navigator Tab:** Shows the members of the temp_service class, specifically the hello(String txt) method.
- Editor Tab:** The 'temp_service.java' file is selected. The code is as follows:

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package Service;
7
8  import javax.jws.WebService;
9  import javax.jws.WebMethod;
10 import javax.jws.WebParam;
11
12 /**
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 }
```

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.



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 * @author Devendra Singh
14 */
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 }
```

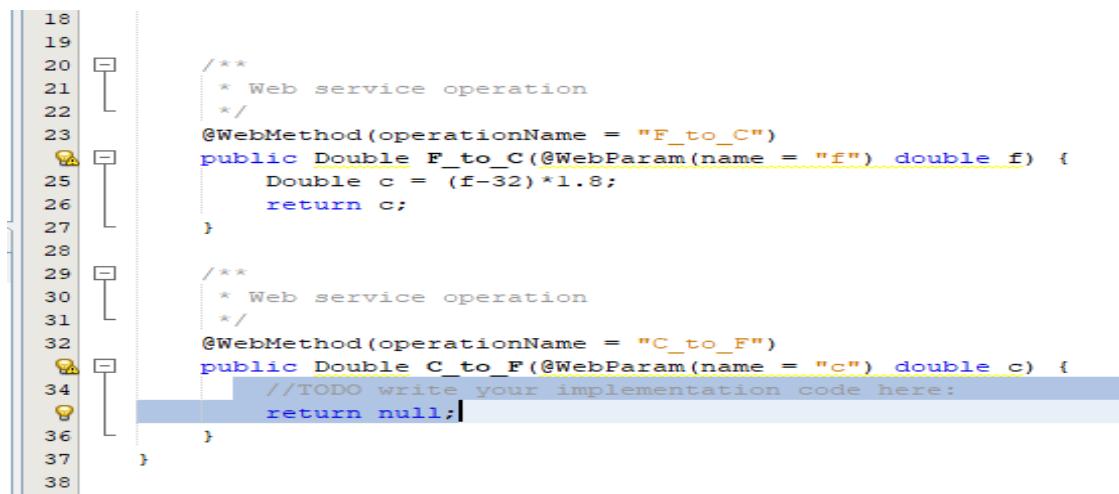
11. Now replace the selected area with following code to convert Fahrenheit to Celsius.

```
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 }
```

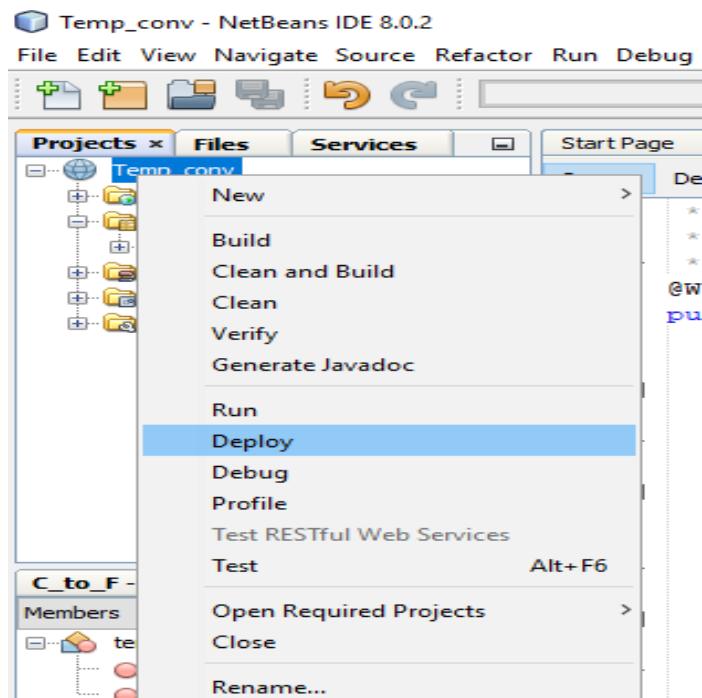
12. Now replace the selected area with following code to convert Celsius to Fahrenheit and then press Ctrl+S to save.

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

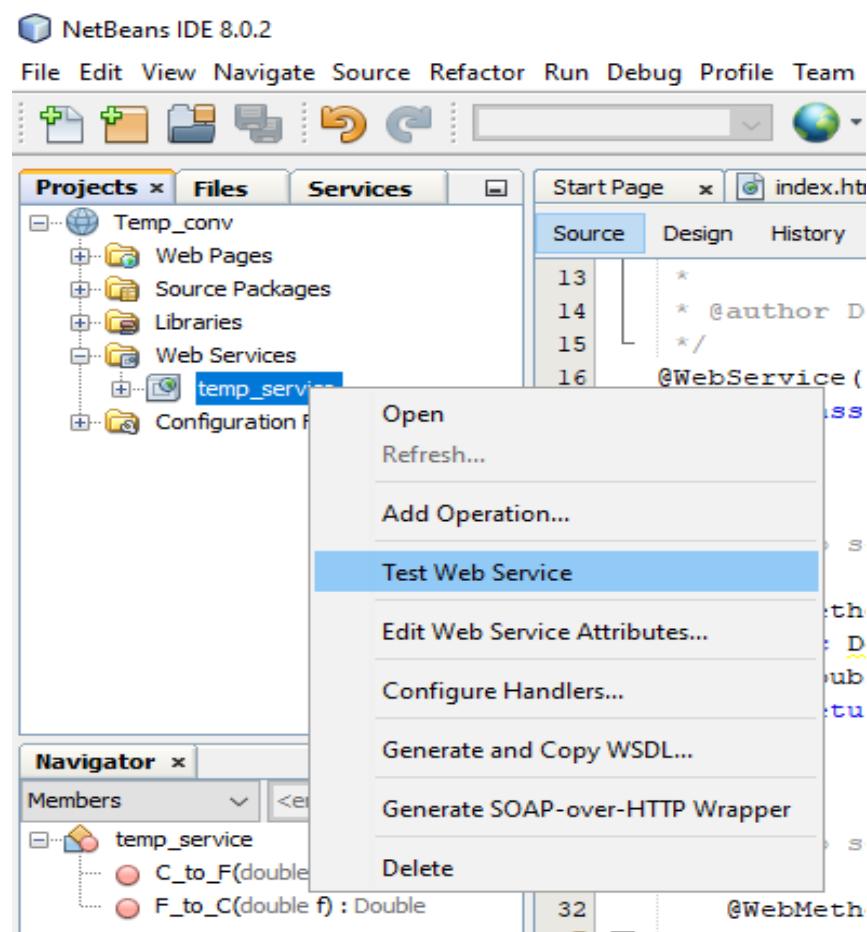


```
18  
19  
20     /**  
21      * Web service operation  
22     */  
23     @WebMethod(operationName = "F_to_C")  
24     public Double F_to_C(@WebParam(name = "f") double f) {  
25         Double c = (f-32)*1.8;  
26         return c;  
27     }  
28  
29     /**  
30      * Web service operation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

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)
fToC

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

- 16.** Selected value is in celsius of 56.

fToC Method invocation

Method parameter(s)

| Type | Value |
|--------|-------|
| double | 56 |

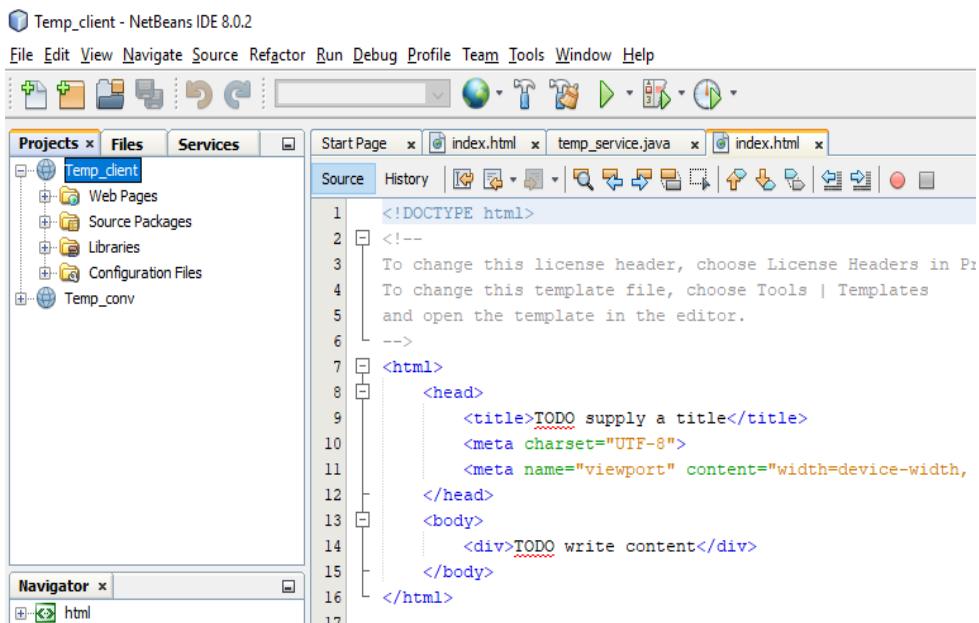
Method returned

[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.

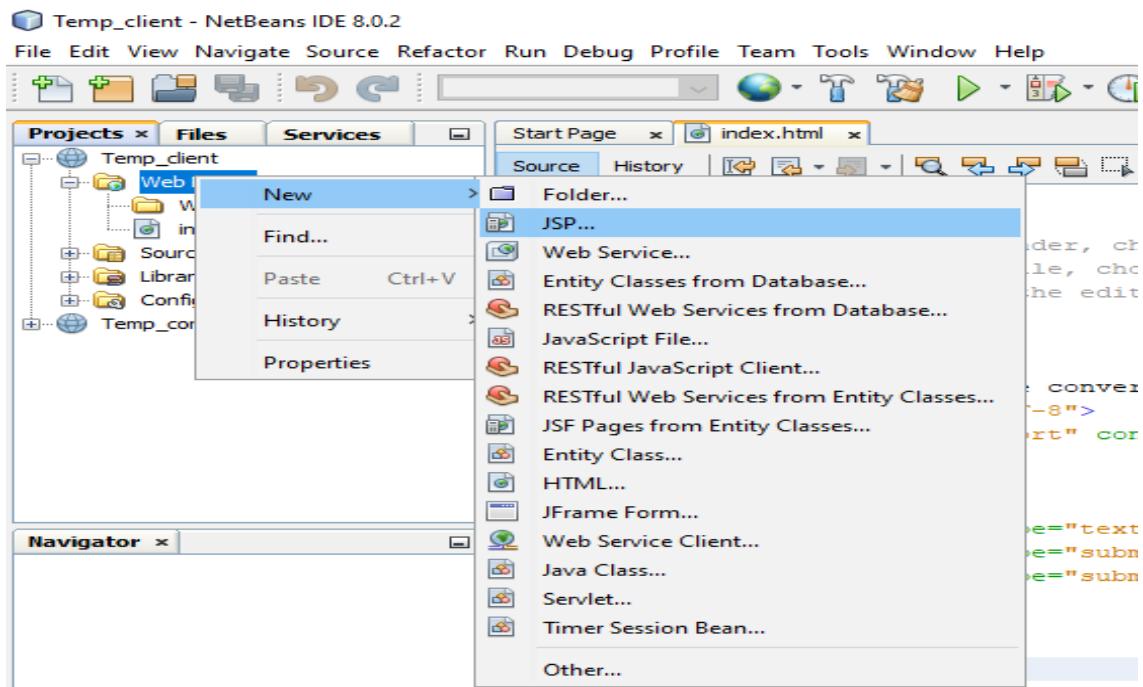
```
<html>
  <head>
    <title>Temperature converter</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  </head>
  <body>
    <form>
      <br><input type="text" name="data"><br>
      <br><input type="submit" value="Convert F to C" name="ftoc"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

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

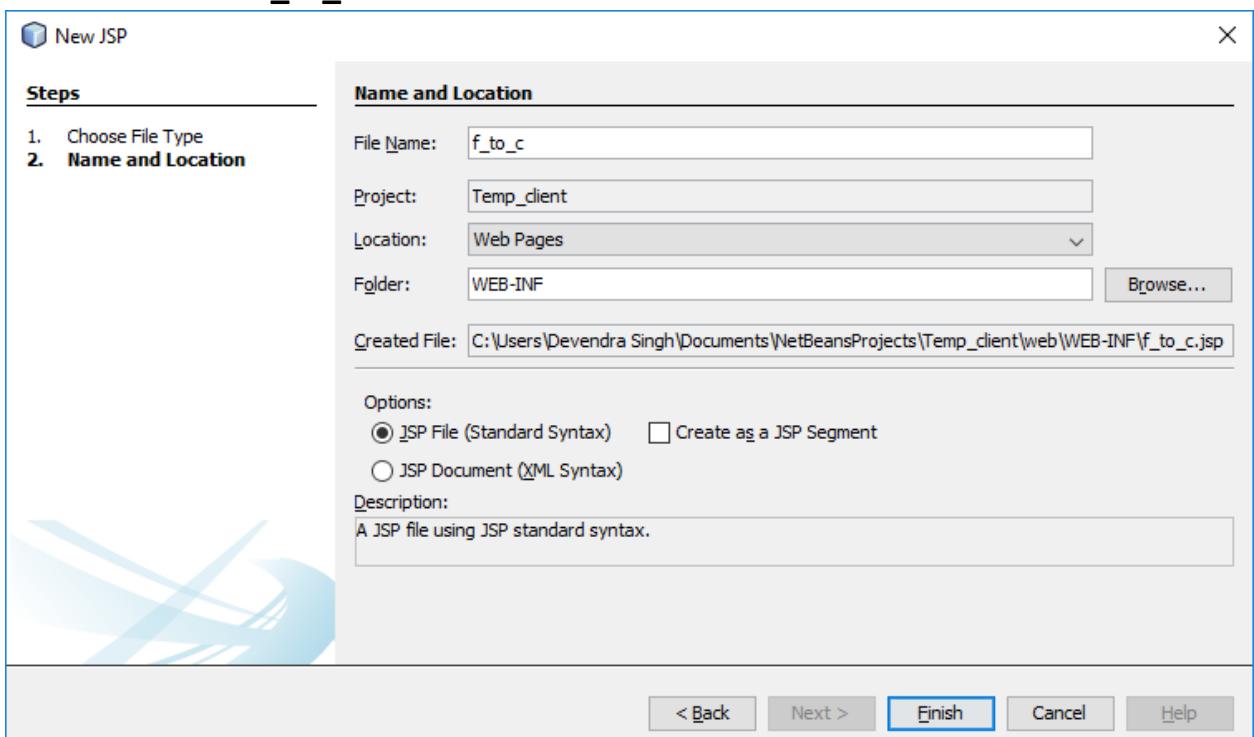
- Projects pane:** Shows the project structure with 'Temp_client' selected. Under 'Web Pages', 'WEB-INF' is expanded, showing 'index.html' which is currently selected.
- Navigator pane:** Shows the structure of the 'index.html' file:
 - html
 - head
 - title
 - meta
 - meta
 - form
 - input type="text" name="data"
 - input type="submit" value="Convert F to C" name="ftoc" formaction="f_to_c.jsp"
 - input type="submit" value="Convert C to F" name="ctof" formaction="c_to_f.jsp"
 - body
 - html
- Source tab:** Displays the source code of 'index.html'. The code includes DOCTYPE, head (with title and meta tags), a form containing two submit buttons (one for F to C conversion and one for C to F conversion), and a body section.

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.

Temp_client - NetBeans IDE 8.0.2

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

Projects x Files Services

Start Page x index.html x f_to_c.jsp x c_to_f.jsp x

Source History

Document : c_to_f
Created on : Aug 13, 2018, 9:04:56 PM
Author : Devendra Singh

1 <%@page contentType="text/html" pageEncoding="UTF-8"%>

2 <!DOCTYPE html>

3 <html>

4 <head>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <title>JSP Page</title>

7 </head>

8 <body>

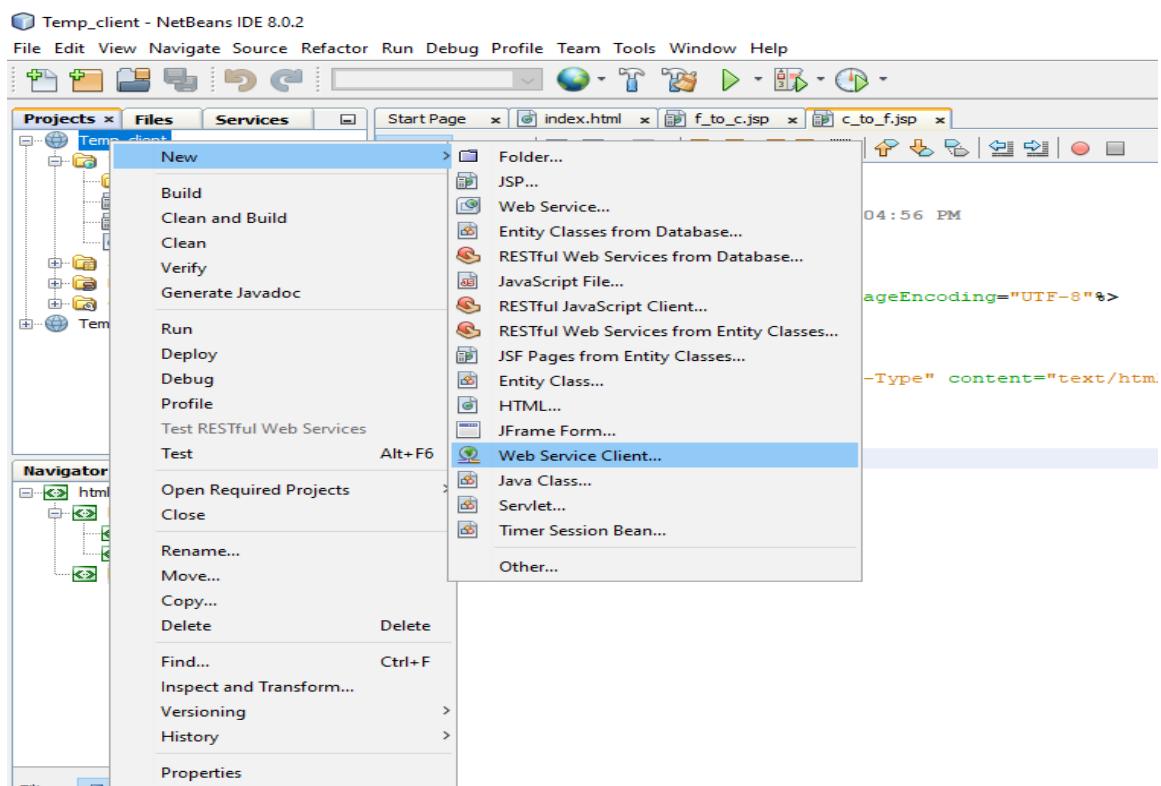
9 <h1>Hello World</h1>

10 </body>

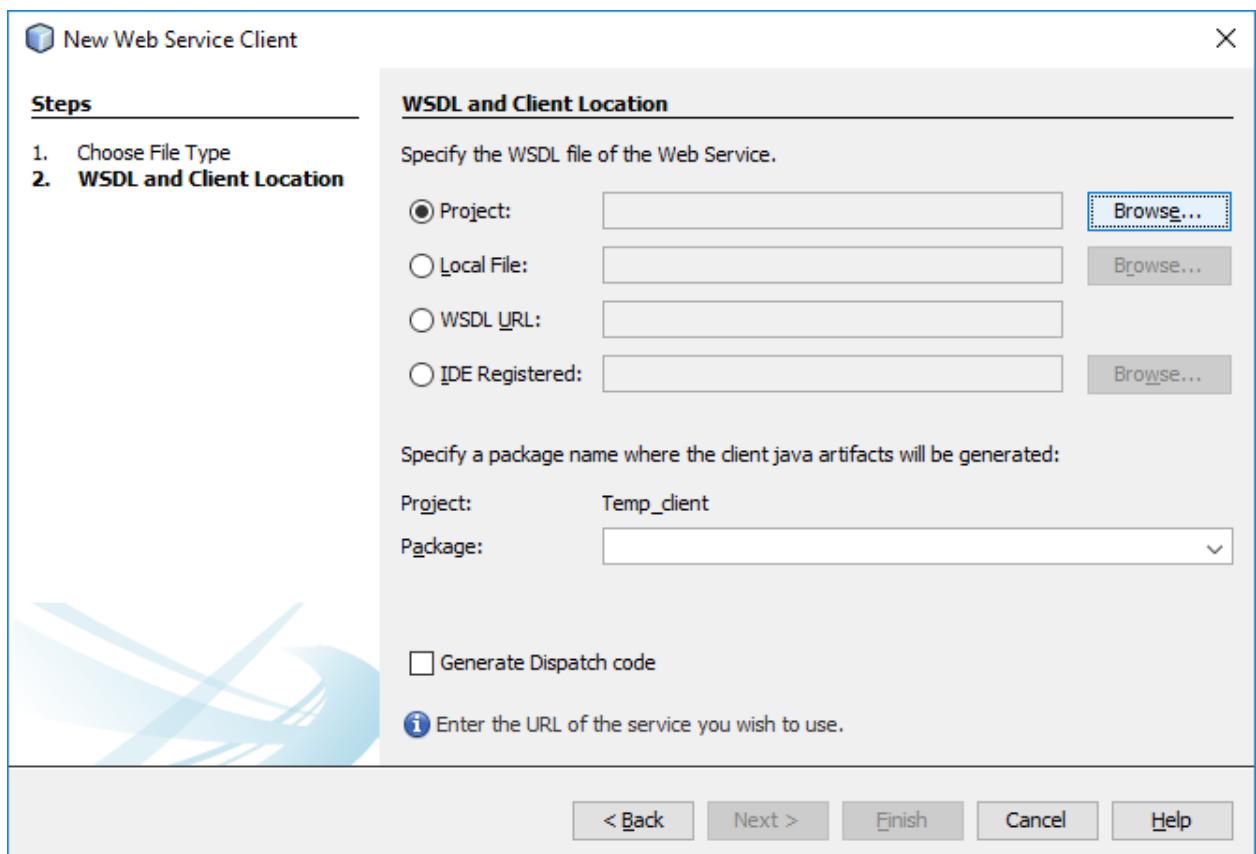
11 </html>

12 <%@page contentType="text/html" pageEncoding="UTF-8"%>

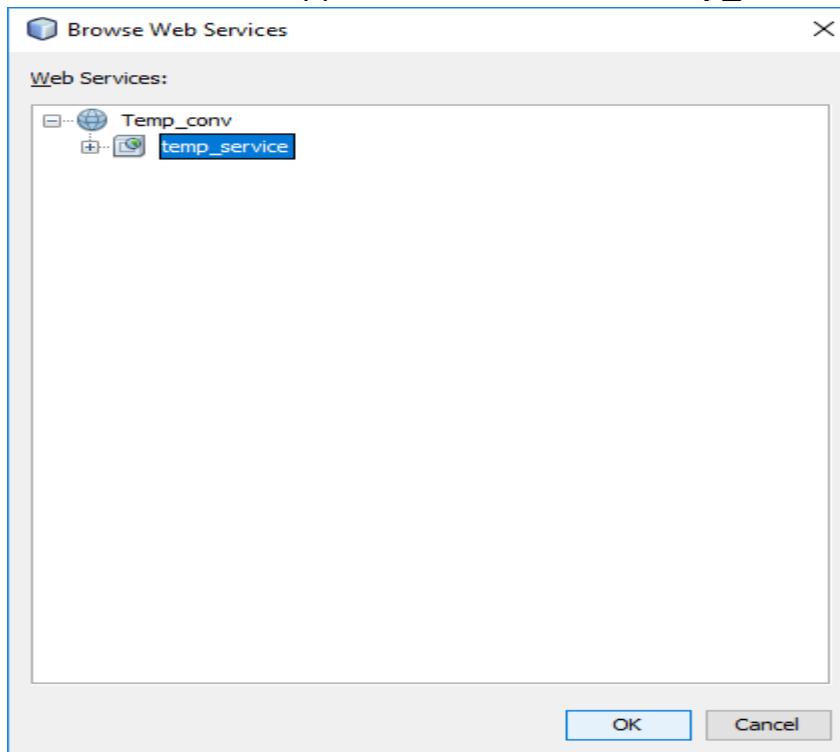
24. Right click on Temp_client and select Web Service Client as below.



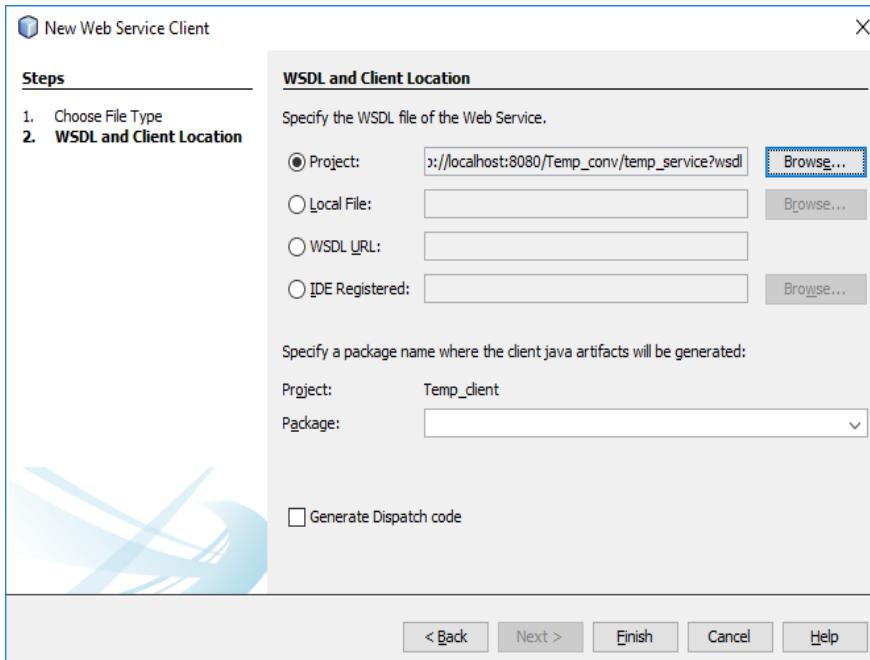
25. Click on Browse.



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.

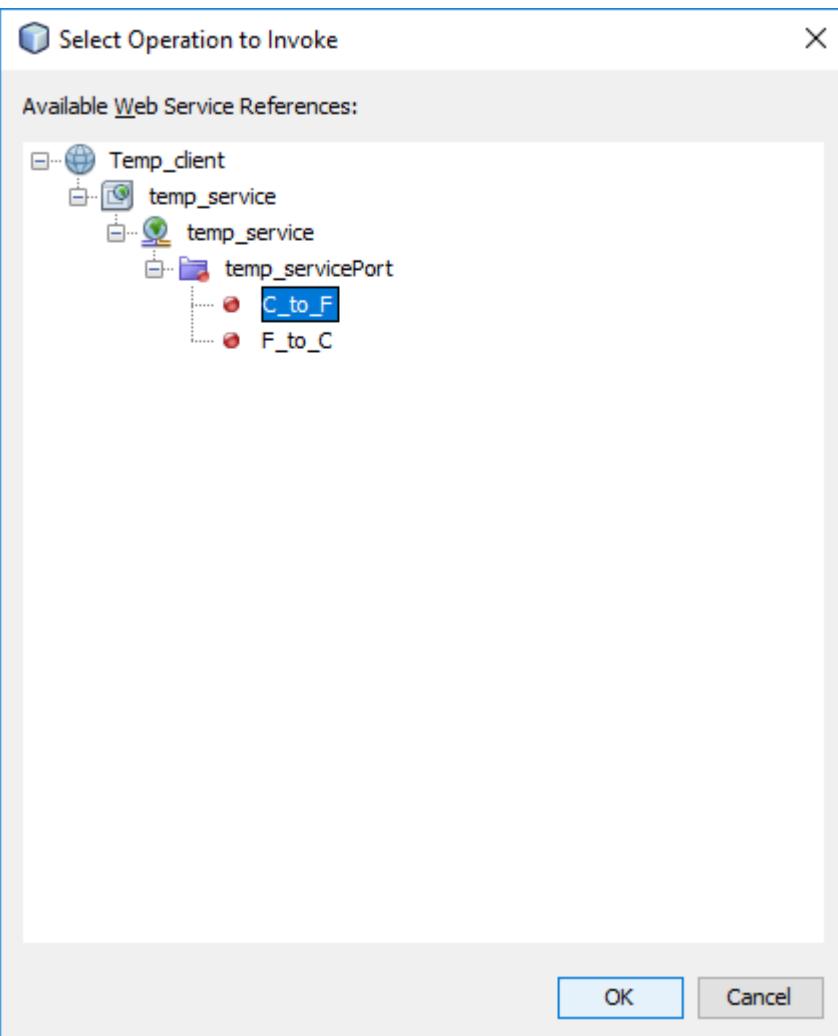
```
<%--  
2 Document : c_to_f  
3 Created on : Aug 13, 2018, 9:04:56 PM  
4 Author : Devendra Singh  
5 --%>  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <!DOCTYPE html>  
9 <html>  
10 <head>  
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
12 <title>JSP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

The screenshot shows an IDE interface with a code editor displaying a JSP file named 'c_to_f.jsp'. The code includes standard JSP declarations and an HTML structure. A context menu is open over the code, listing various options like 'Run File', 'Find Usages', 'Refactor', etc. At the bottom of the menu, there are two items: 'Web Service Client Resources' and 'Call Web Service Operation...'. The 'Call Web Service Operation...' option is highlighted with a blue background.

```
<%--  
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 <html>  
9 <head>  
10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
11 <title>JSP Page</title>  
12 </head>  
13 <body>  
14 </body>  
15 </html>
```

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.



OK

Cancel

```

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

```

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

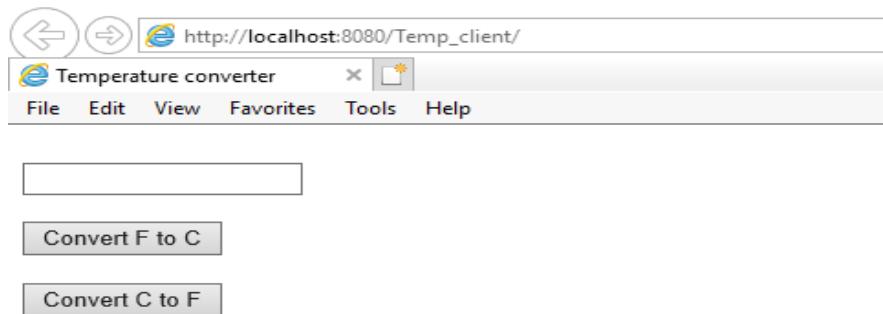
```

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

```

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.



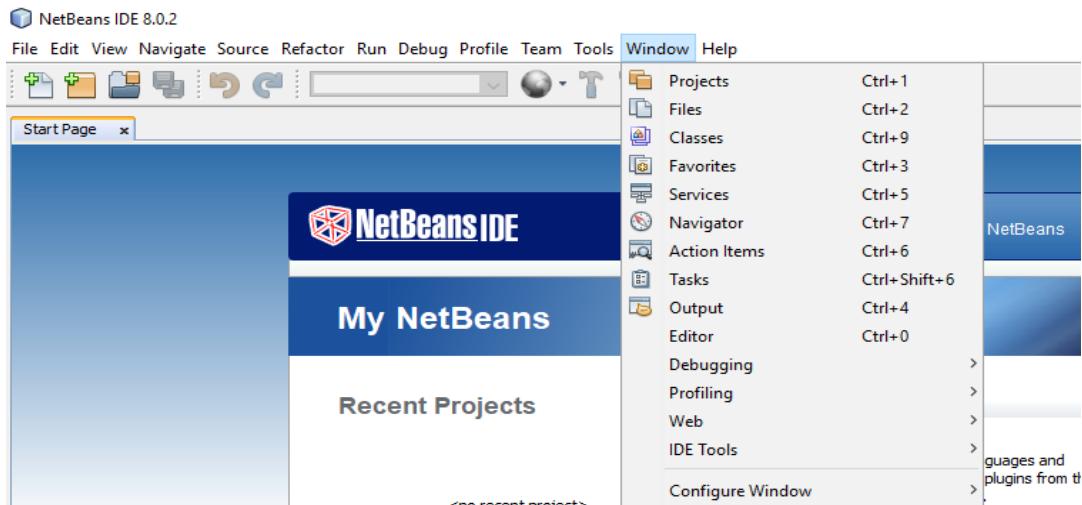
34. Now you can enter any numeric data into textbox and if you will click the first button it will convert the numeric value into Celsius and vice-versa for the second button.

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

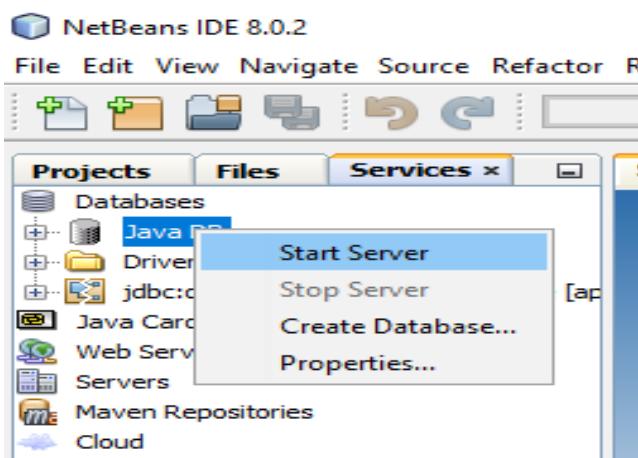
PRACTICAL-2

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.

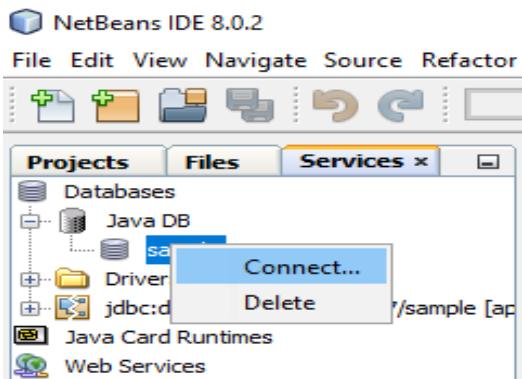
1. Click on Window menu and click on **Projects, Files & Services** to open it.



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

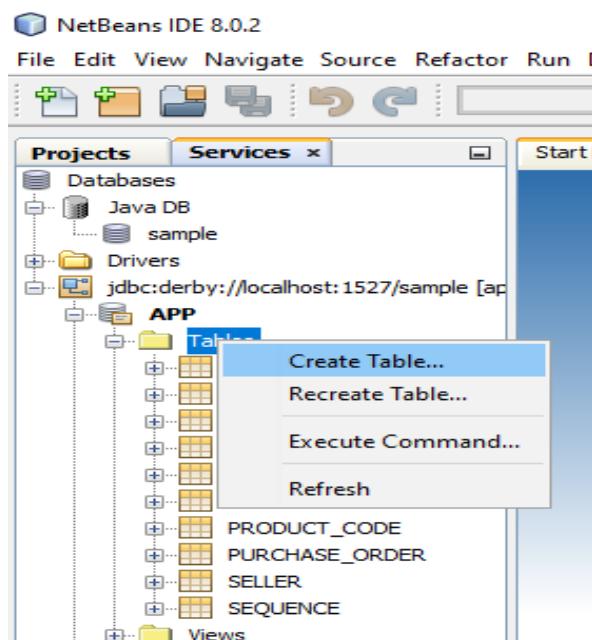


3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.

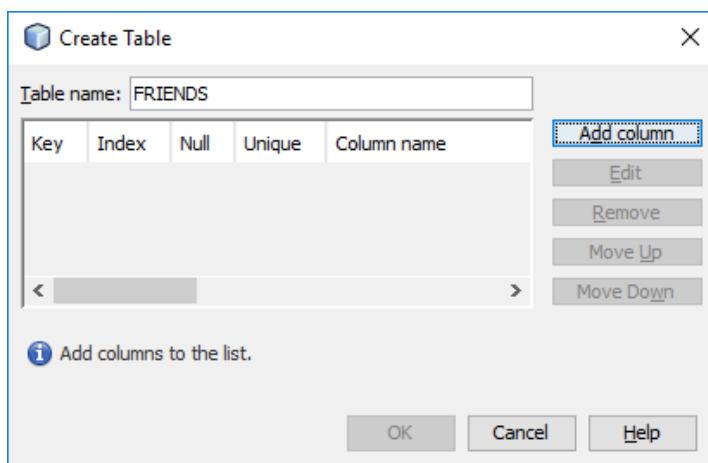


4. Now we are going to create a table in default database **sample**.

Right click on Table -> Create Table

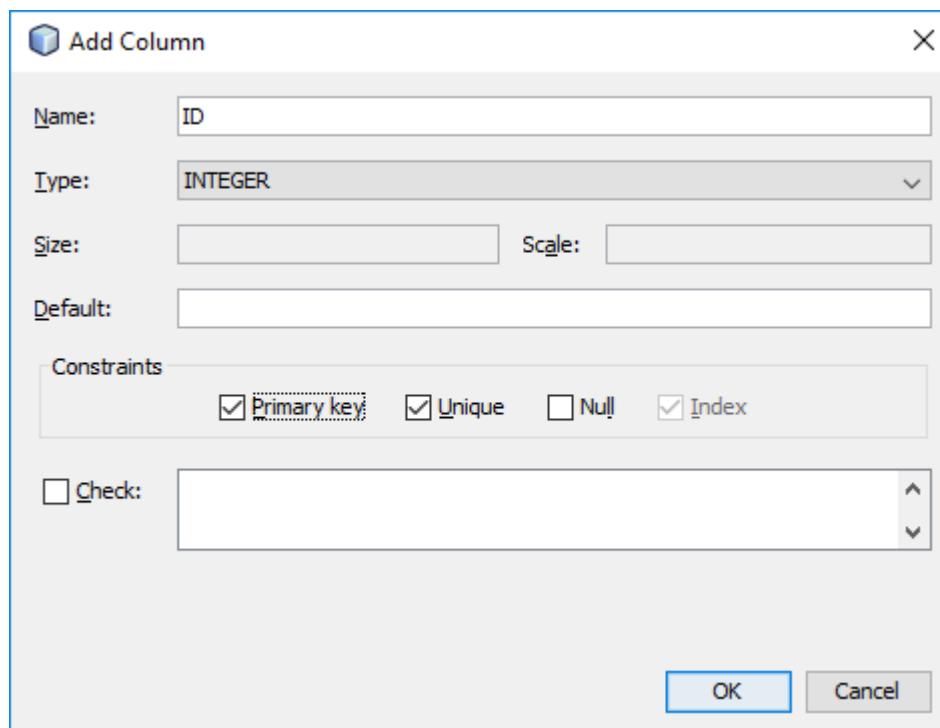


5. Give **table name as FRIENDS**.

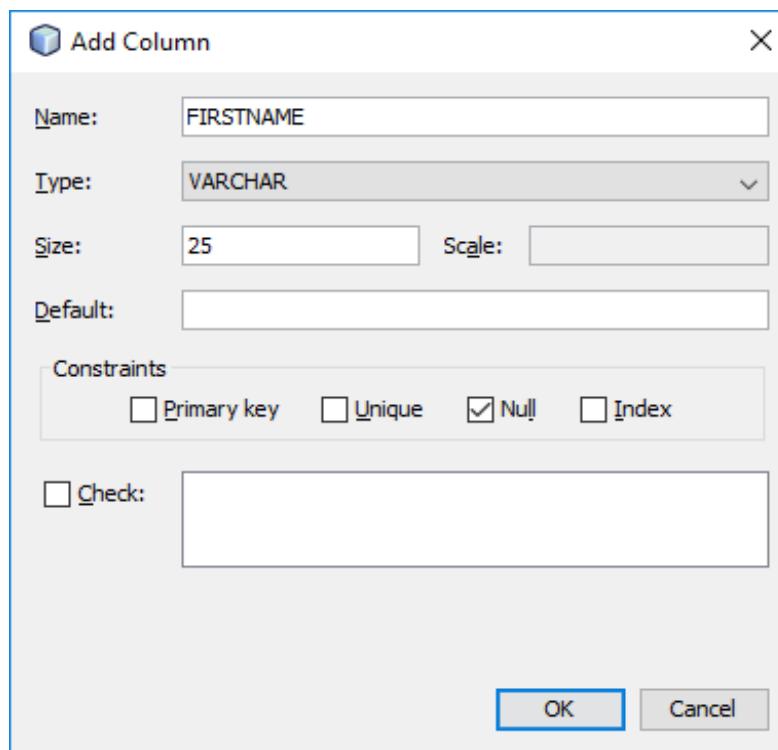


6. Now click on Add column button to add columns in table.

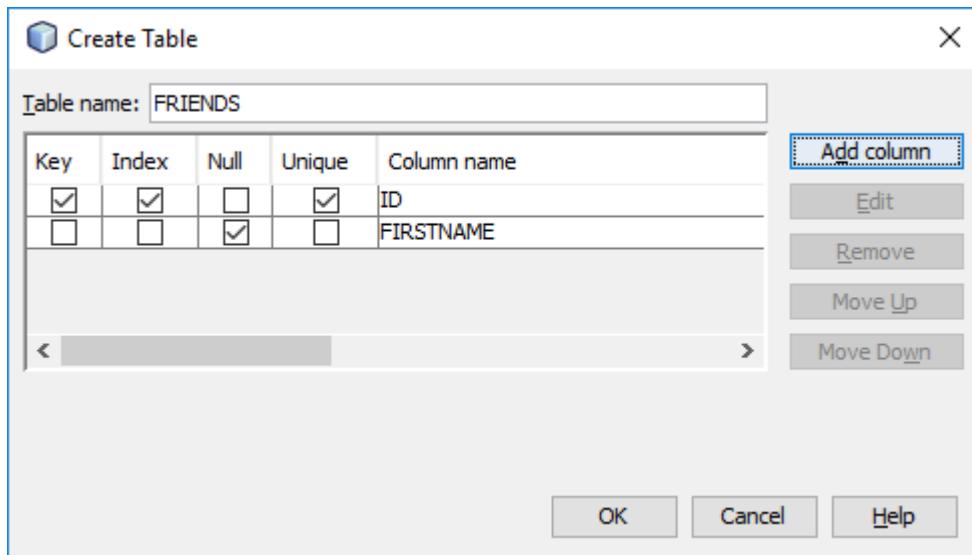
Enter details as in below pic and select Primary key. After that click on OK button.



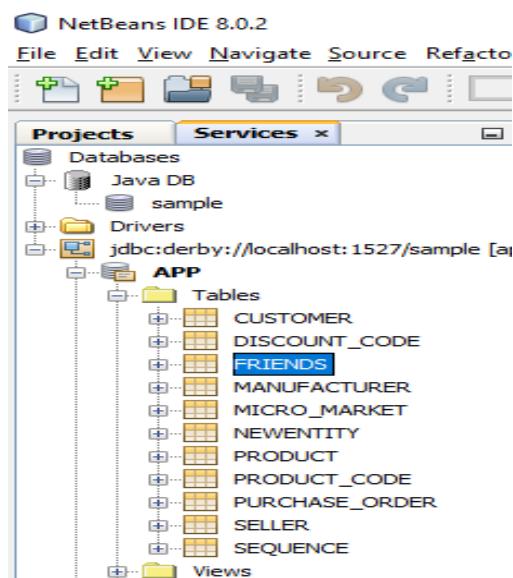
7. Now add second column with following detail. But don't select primary and click on OK button.



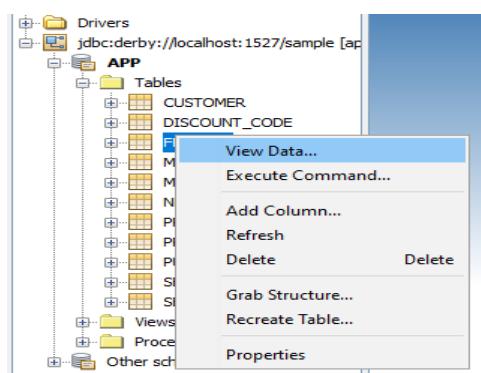
8. Now click on OK button.



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.

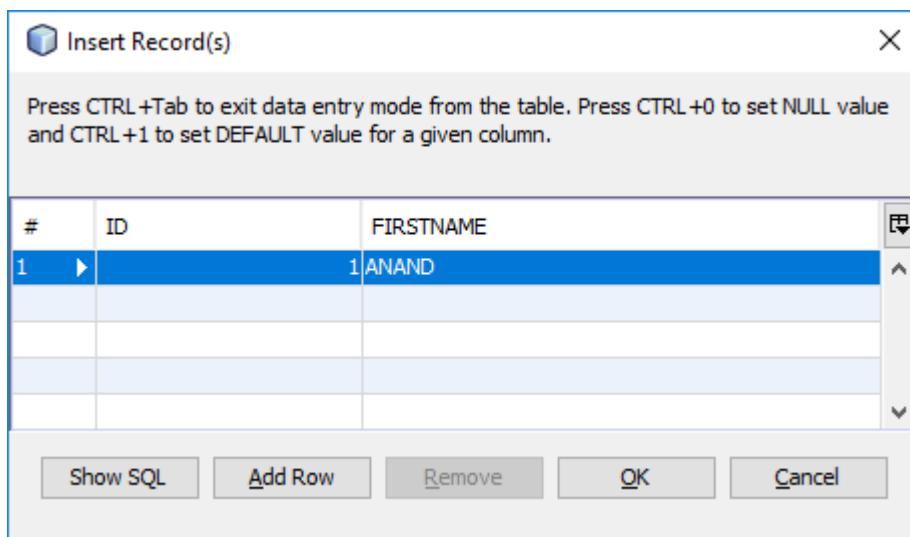


11. Now click on the leftmost icon in second panel to insert some record.

The screenshot shows the SQL Workbench/J application. On the left, the 'Projects' and 'Services' tabs are selected. Under 'Databases', there is a 'Java DB' section with a 'sample' database. Below it is a 'Drivers' section with a connection to 'jdbc:derby://localhost:1527/sample [app]'. The 'APP' schema is expanded, showing 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 with a connection to 'jdbc:derby://localhost:1527/sample [app on APP]'. It contains a SQL editor with the query 'select * from APP.FRIENDS;' and a results grid below it. The results grid shows 7 rows of data with columns '#', 'ID', and 'FIRSTNAME'. The first row is highlighted with a blue background.

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

After that click on OK button to finish.

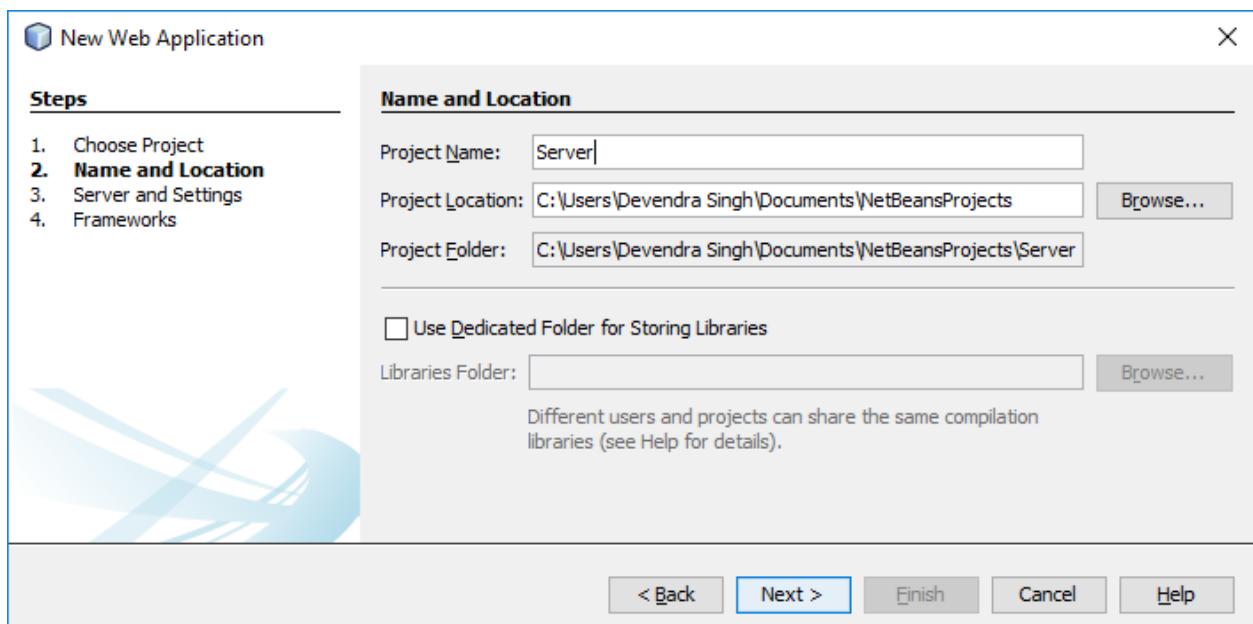


13. As you can see, I have entered 7 records.

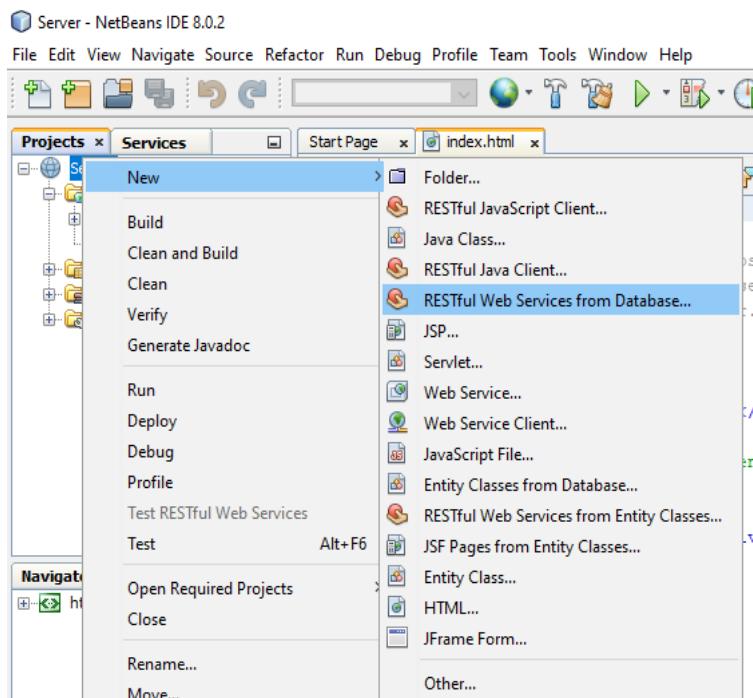
The screenshot shows the results of the 'select * from APP.FRIENDS;' query in the SQL Workbench/J interface. The results grid has columns '#', 'ID', and 'FIRSTNAME'. The data consists of 7 rows, each with a unique ID and a corresponding FIRSTNAME: 1 ANAND, 2 JULHAS, 3 NIKHIL, 4 GAGAN, 5 RAVI, 6 DHARMENDRA, and 7 ADARSH.

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

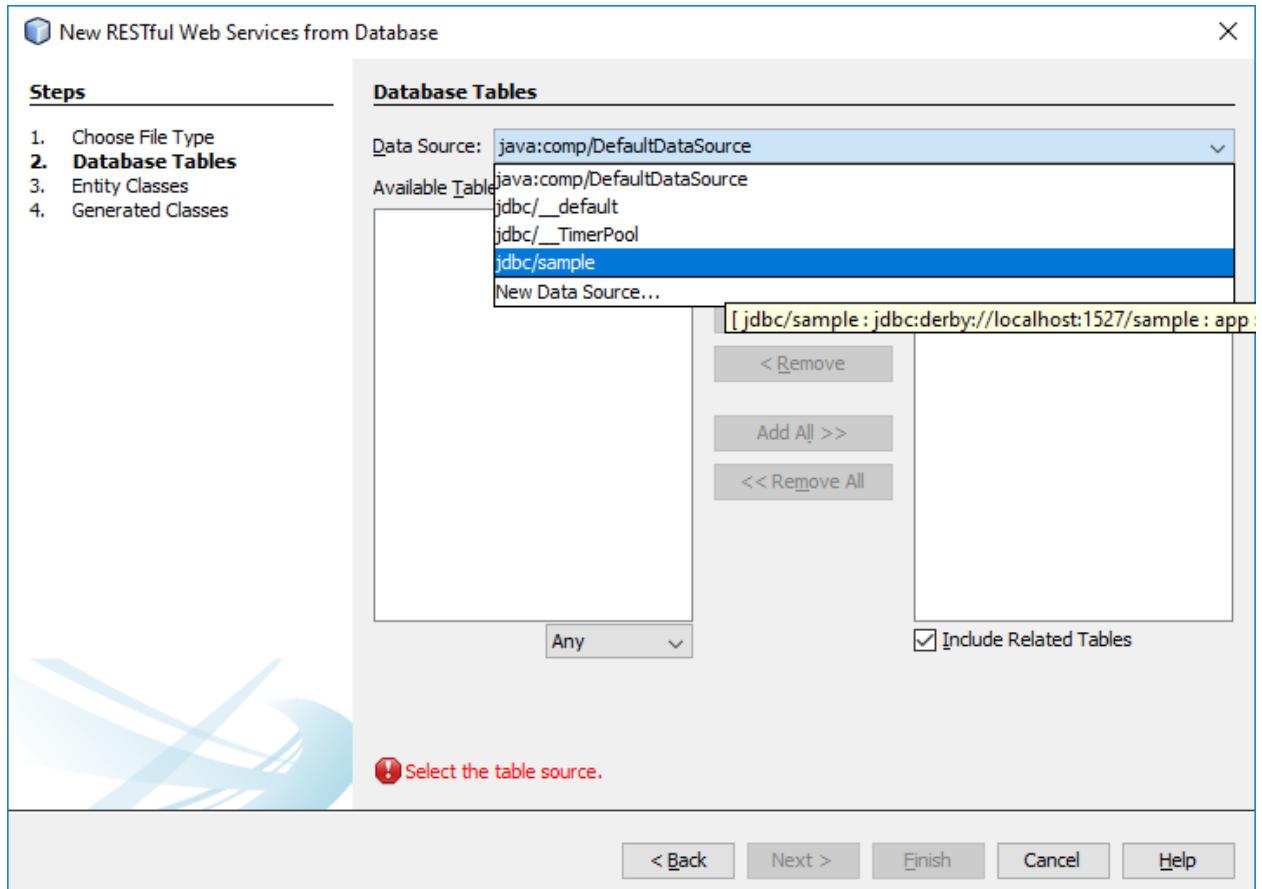
14. Now create a web application with name Server. After that click on Next and then Finish button.



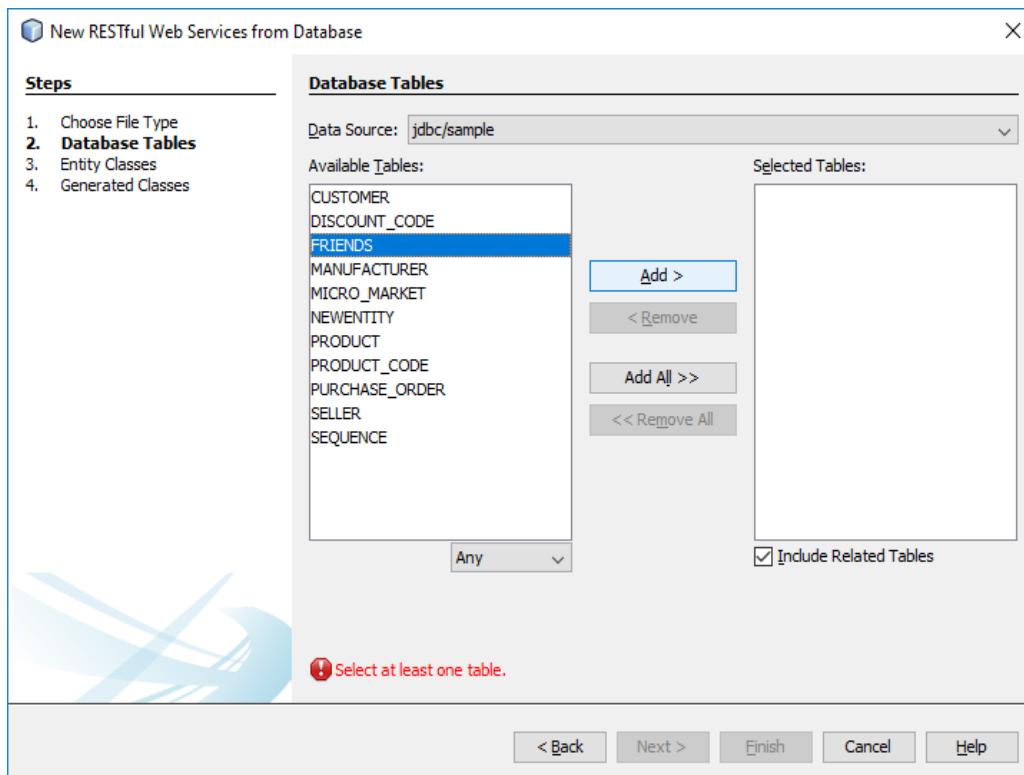
15. Now create a RESTful Web Service from Database by right click on project name.



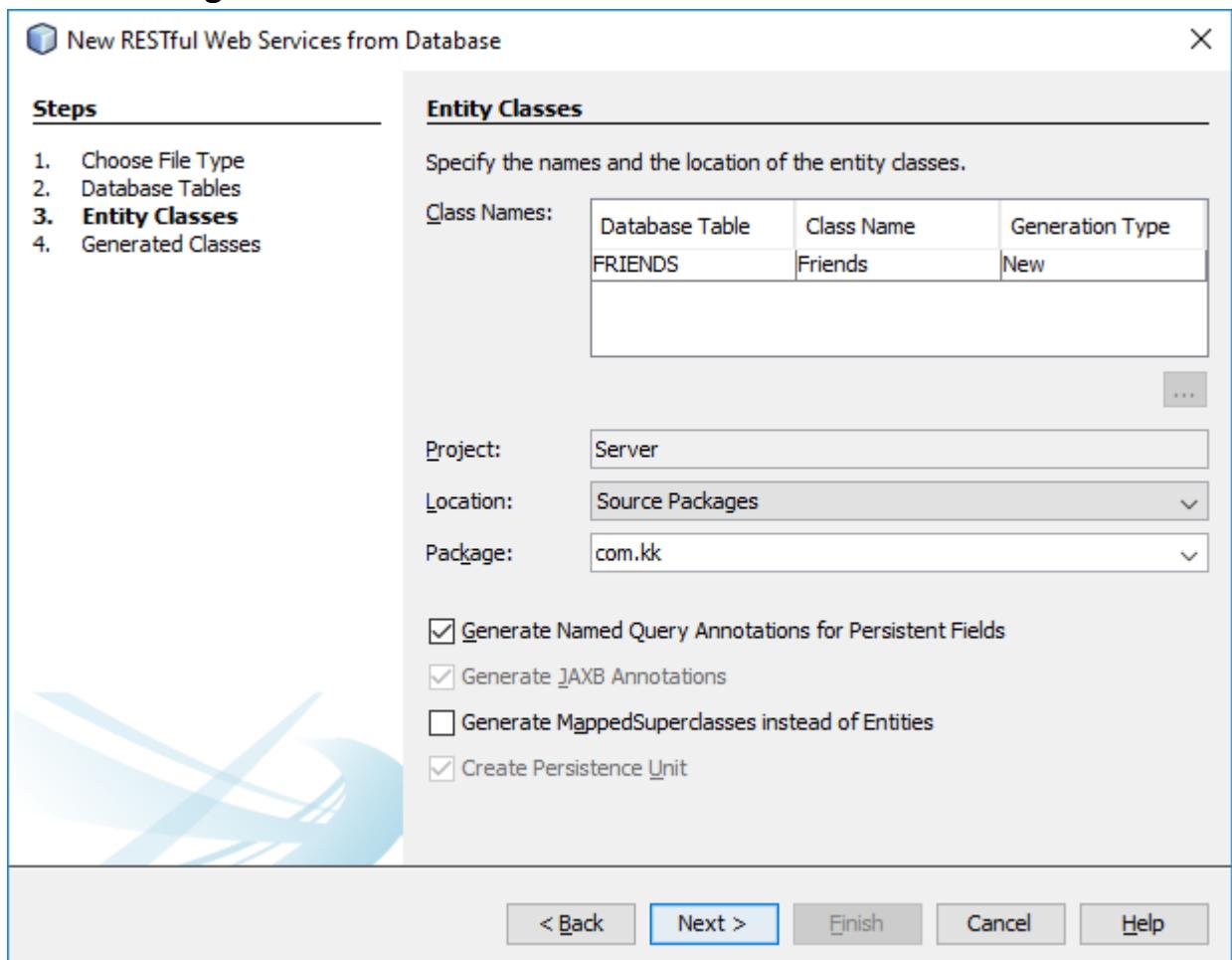
16. Choose Data Source jdbc/sample.



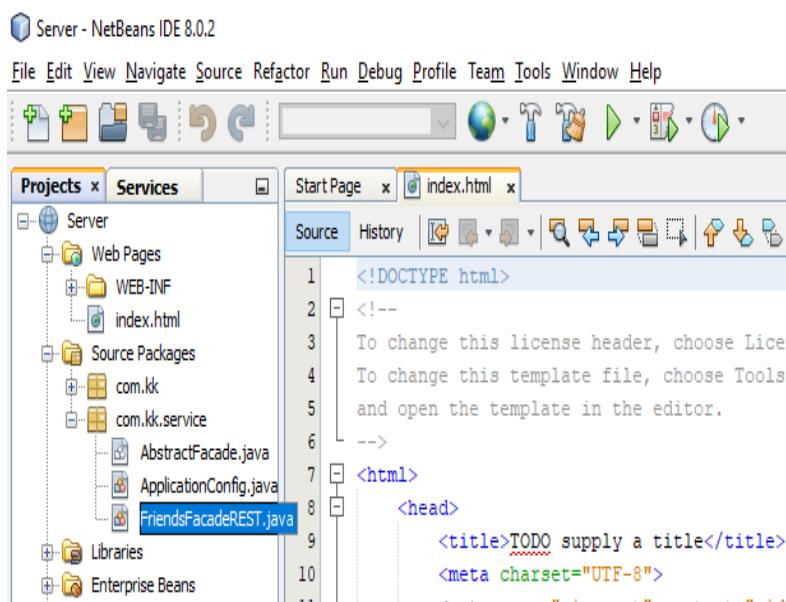
17. Now select FRIENDS and click on Add button. After that click on Next button.



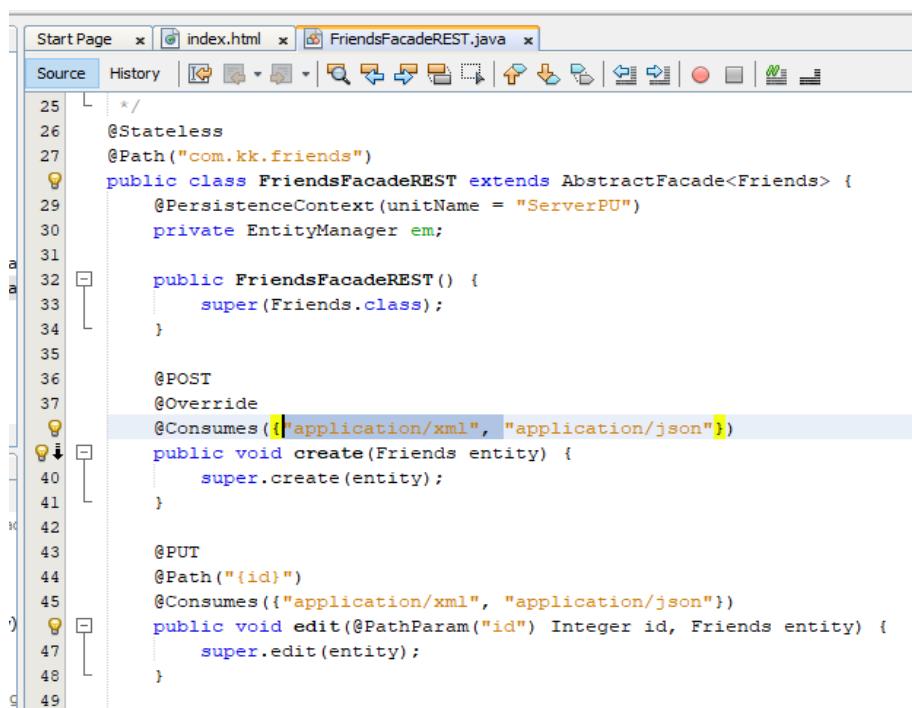
18. Enter Package name as com.kk and click on Next button and then Finish.



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



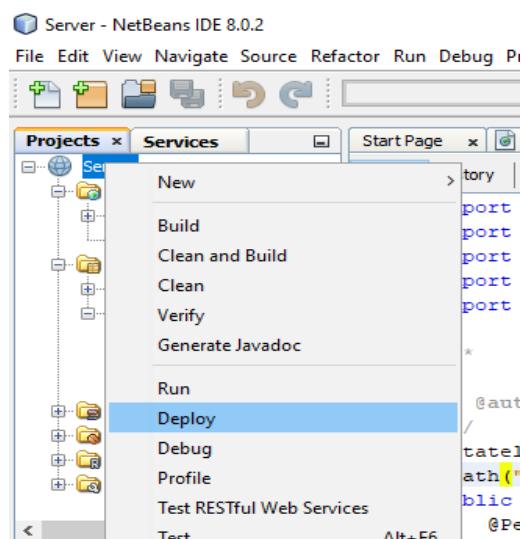
20. Now remove the selected part from every method in this file. So that it will communicate only in JSON format. You can also use methods to convert it. But this is easiest method.



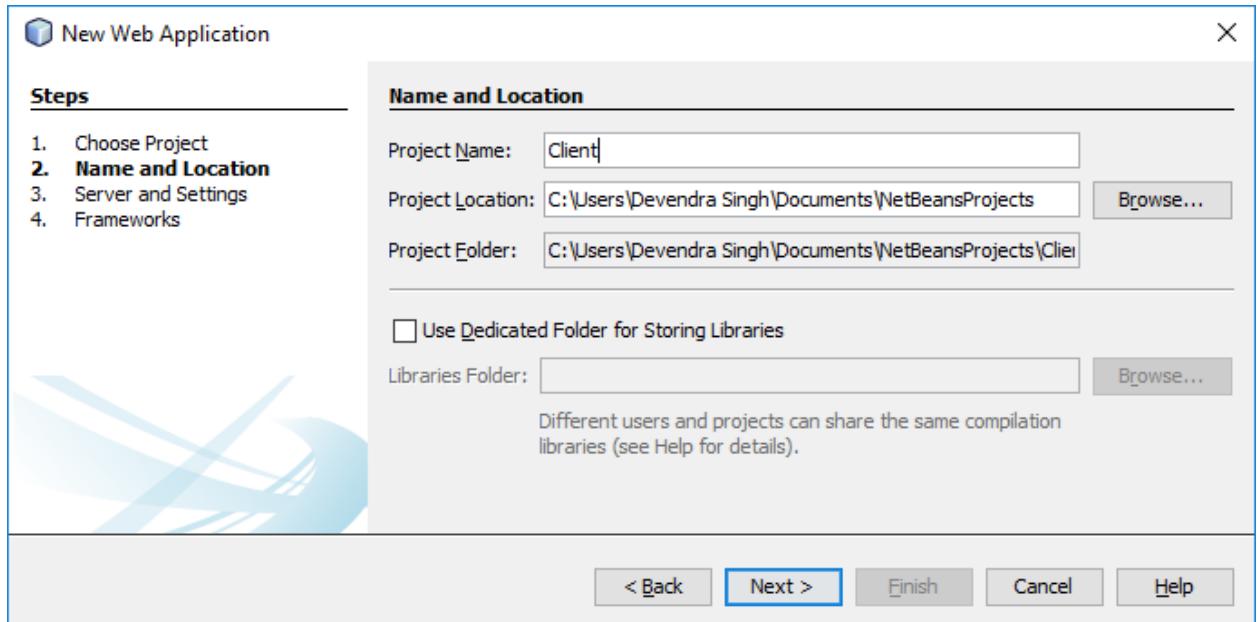
The screenshot shows the NetBeans IDE interface with the code editor open. The file is FriendsFacadeREST.java. The code defines a RESTful service for managing Friends entities. A specific section of the code is highlighted in light blue, indicating it is selected for modification:

```
25  */
26  @Stateless
27  @Path("com.kk.friends")
28  public class FriendsFacadeREST extends AbstractFacade<Friends> {
29      @PersistenceContext(unitName = "ServerPU")
30      private EntityManager em;
31
32      public FriendsFacadeREST() {
33          super(Friends.class);
34      }
35
36      @POST
37      @Override
38      @Consumes({"application/xml", "application/json"})
39      public void create(Friends entity) {
40          super.create(entity);
41      }
42
43      @PUT
44      @Path("{id}")
45      @Consumes({"application/xml", "application/json"})
46      public void edit(@PathParam("id") Integer id, Friends entity) {
47          super.edit(entity);
48      }
49 }
```

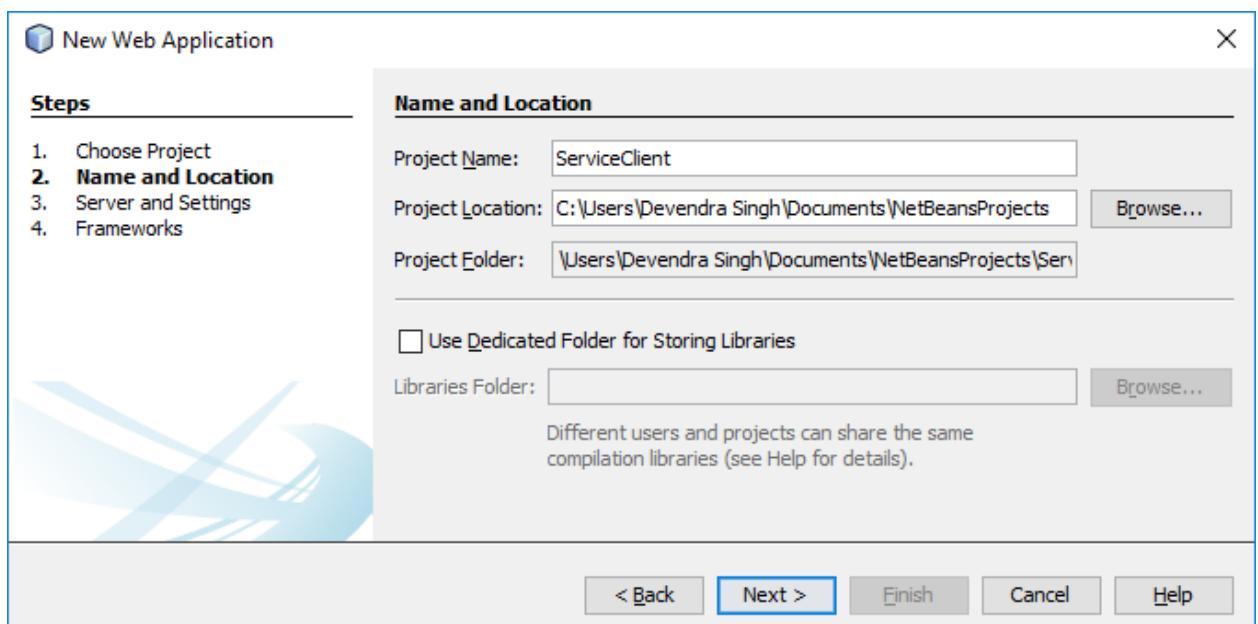
21. After that right click on project name and Deploy it.



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

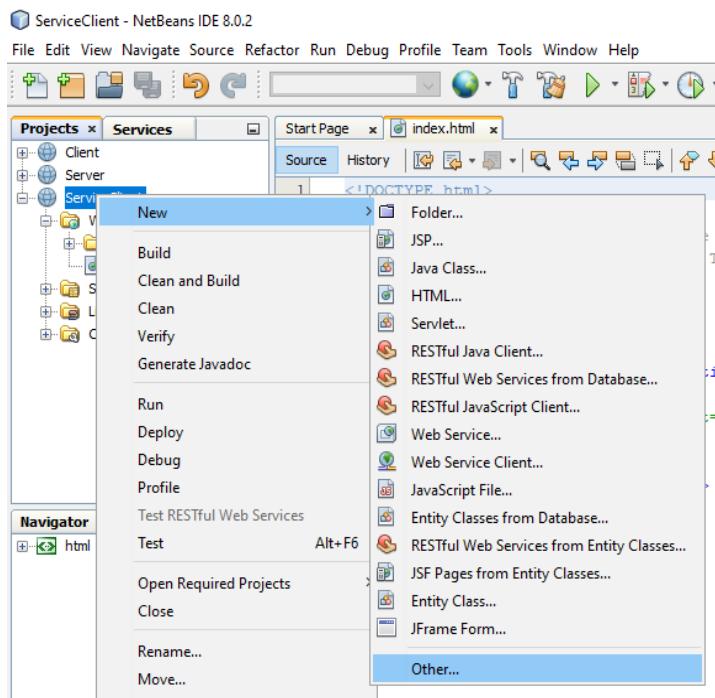


23. Create a **Web Application** with name **ServiceClient**.

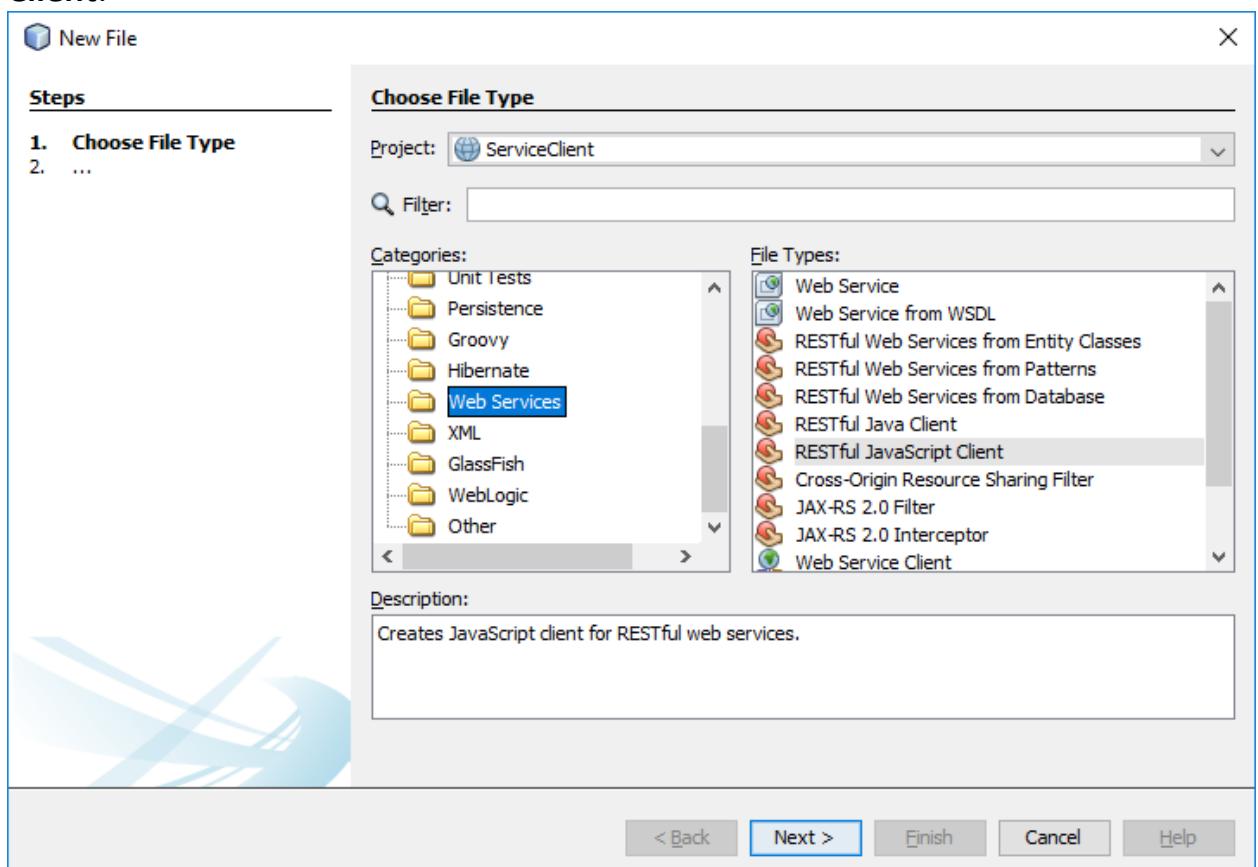


24. Now create a RESTful Java Client.

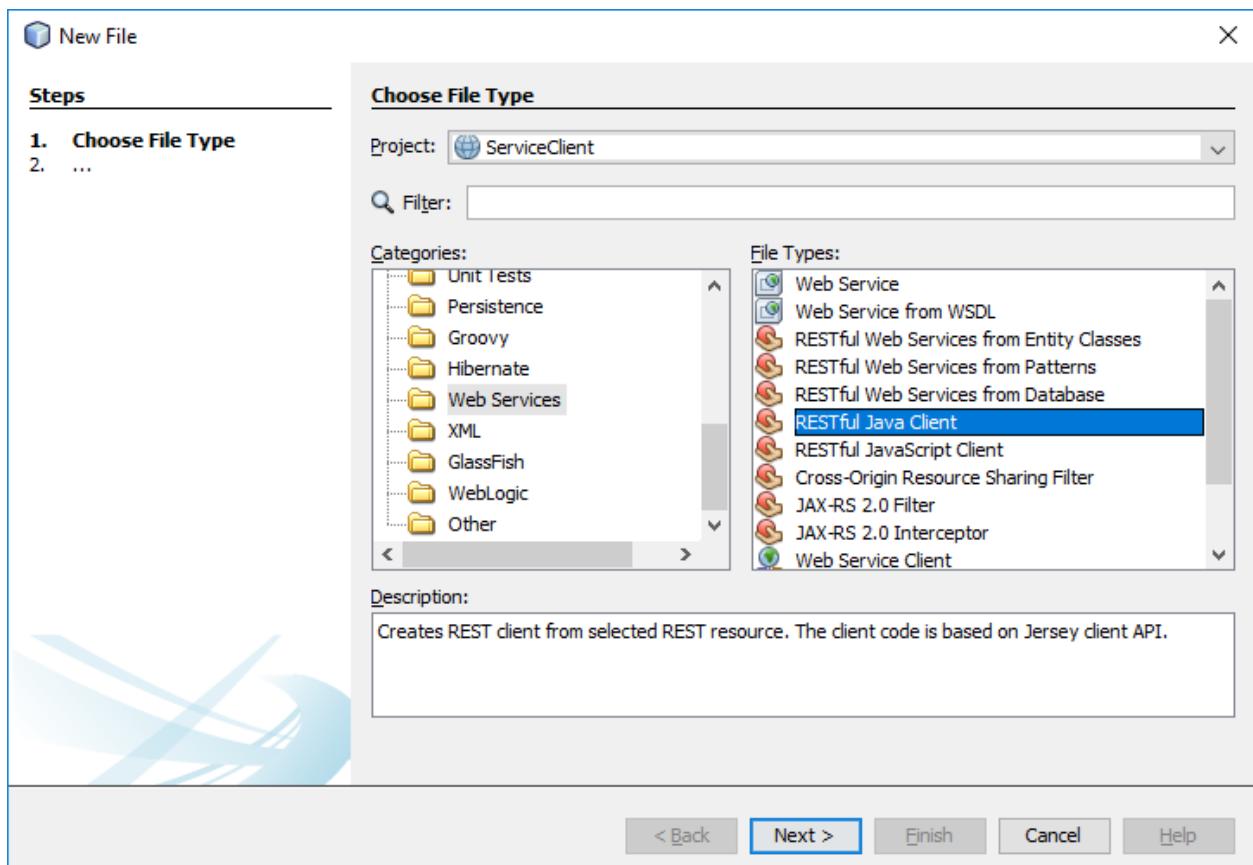
Right click on **ServiceClient** -> New -> Other.



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



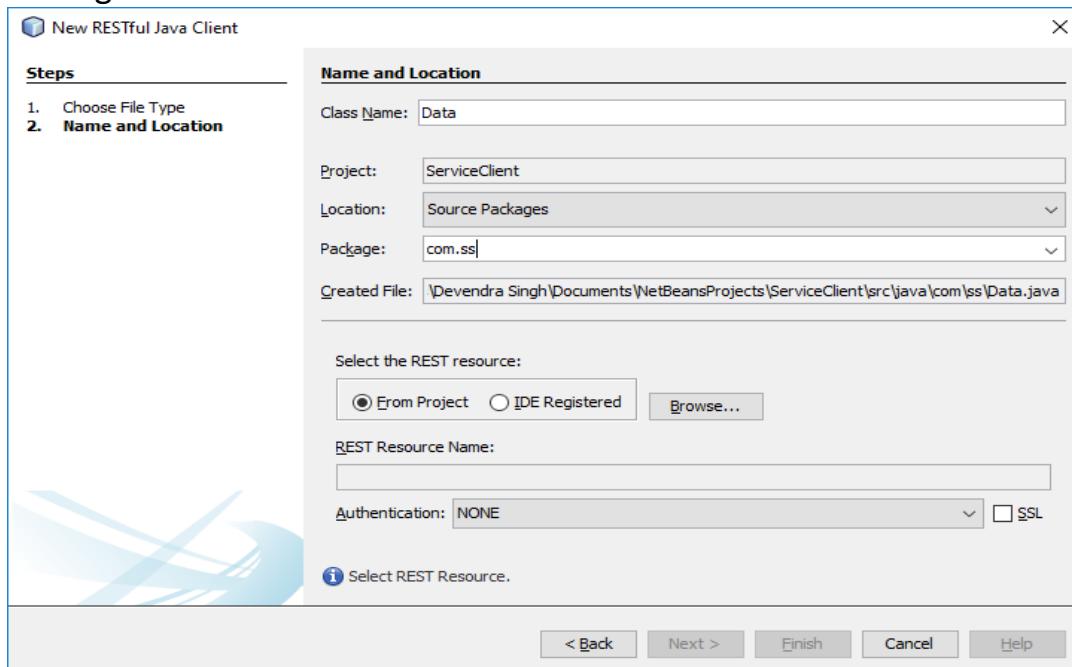
26. After select RESTful Java Client click on Next.



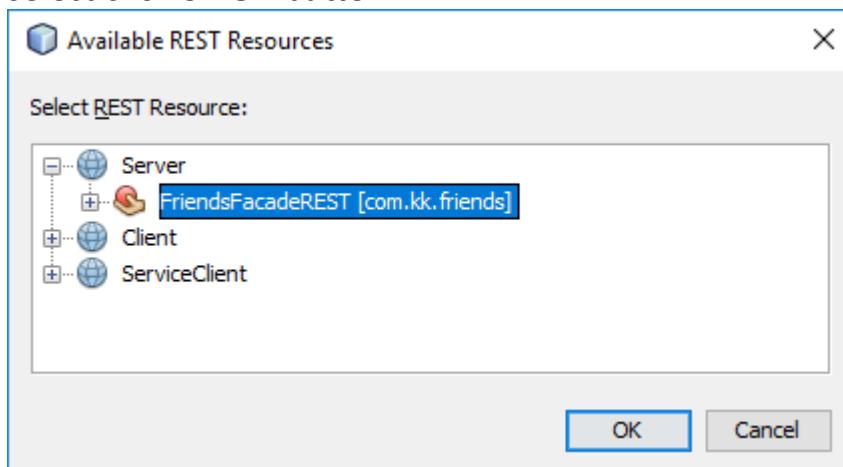
27. Enter following data.

Class Name -> Data

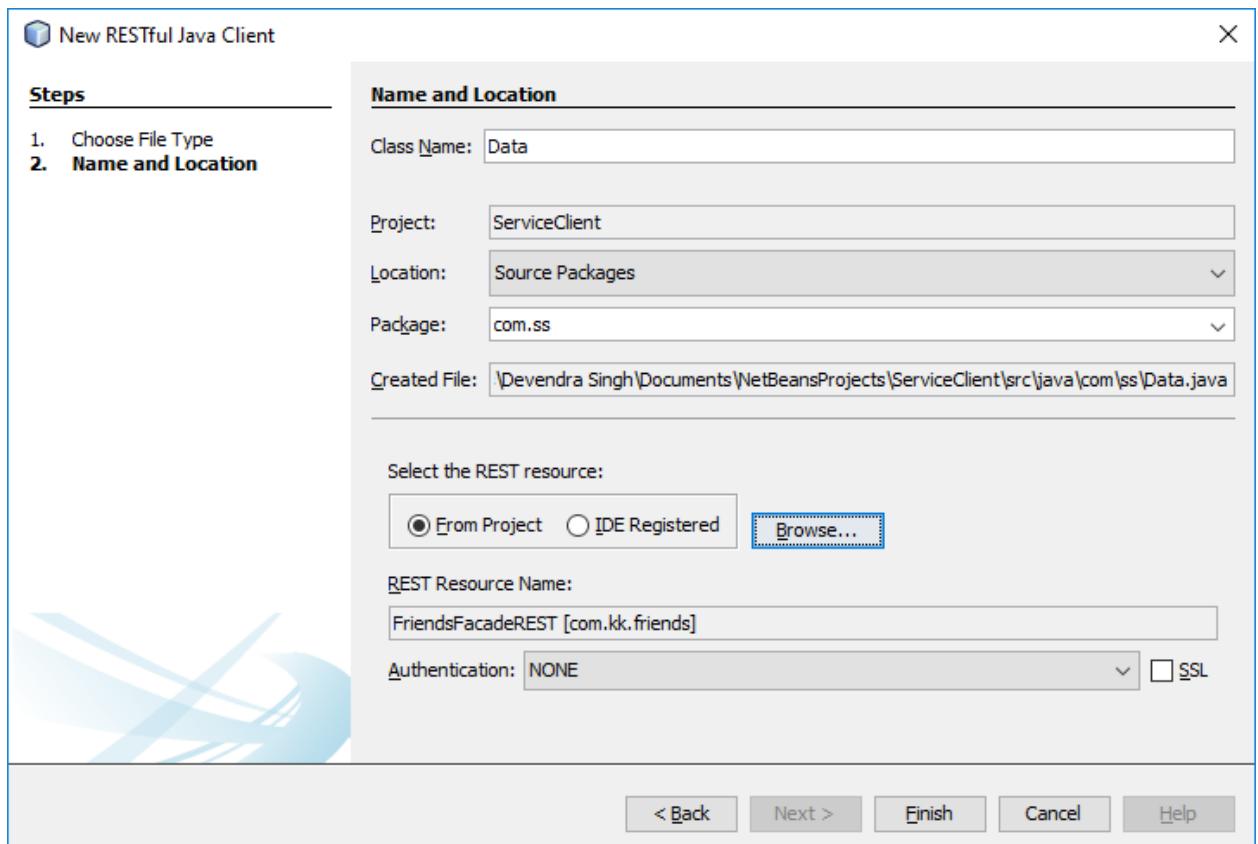
Package -> com.ss



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

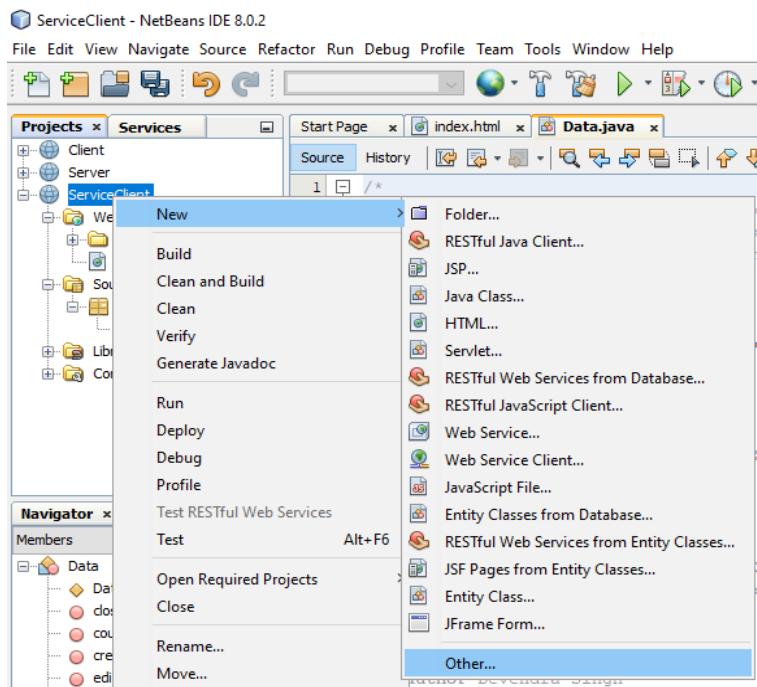


29. Click on **Finish**.

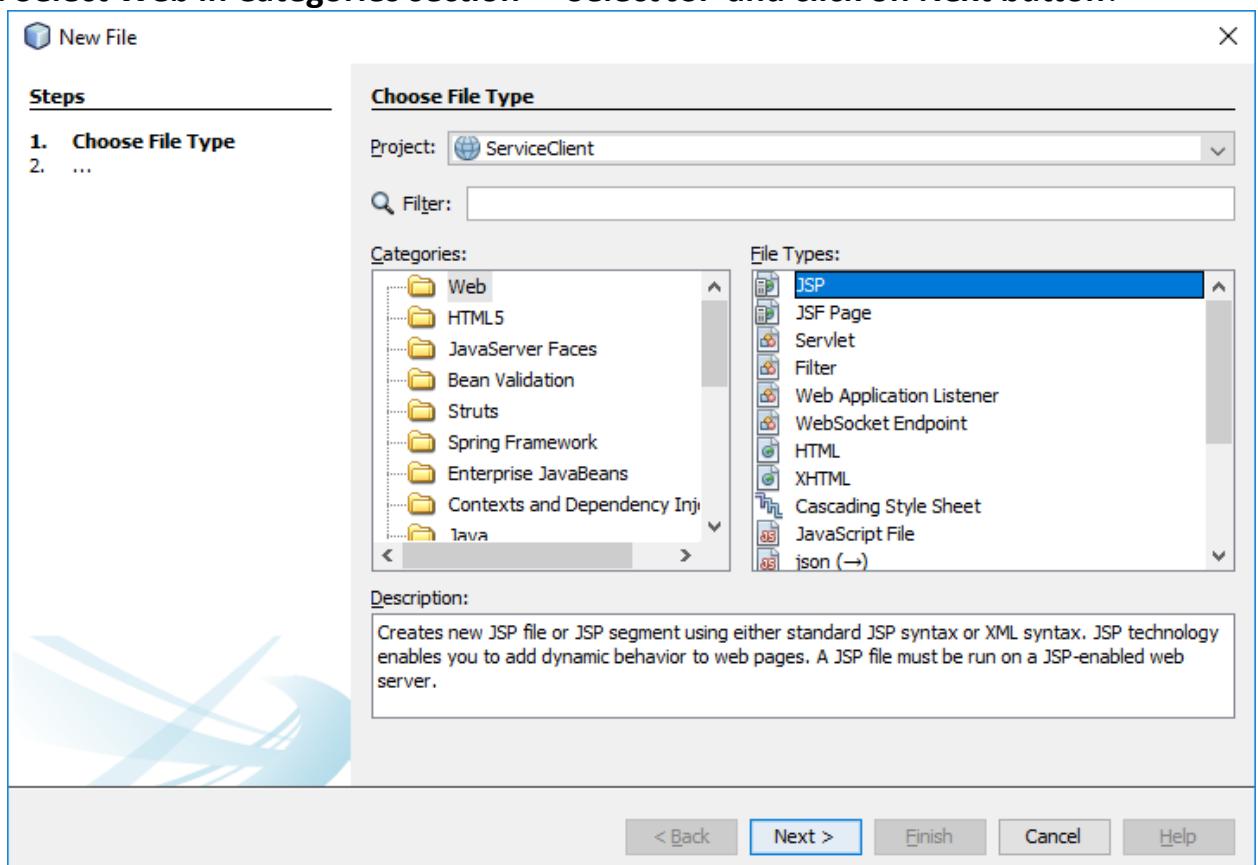


30. Now create a JSP page.

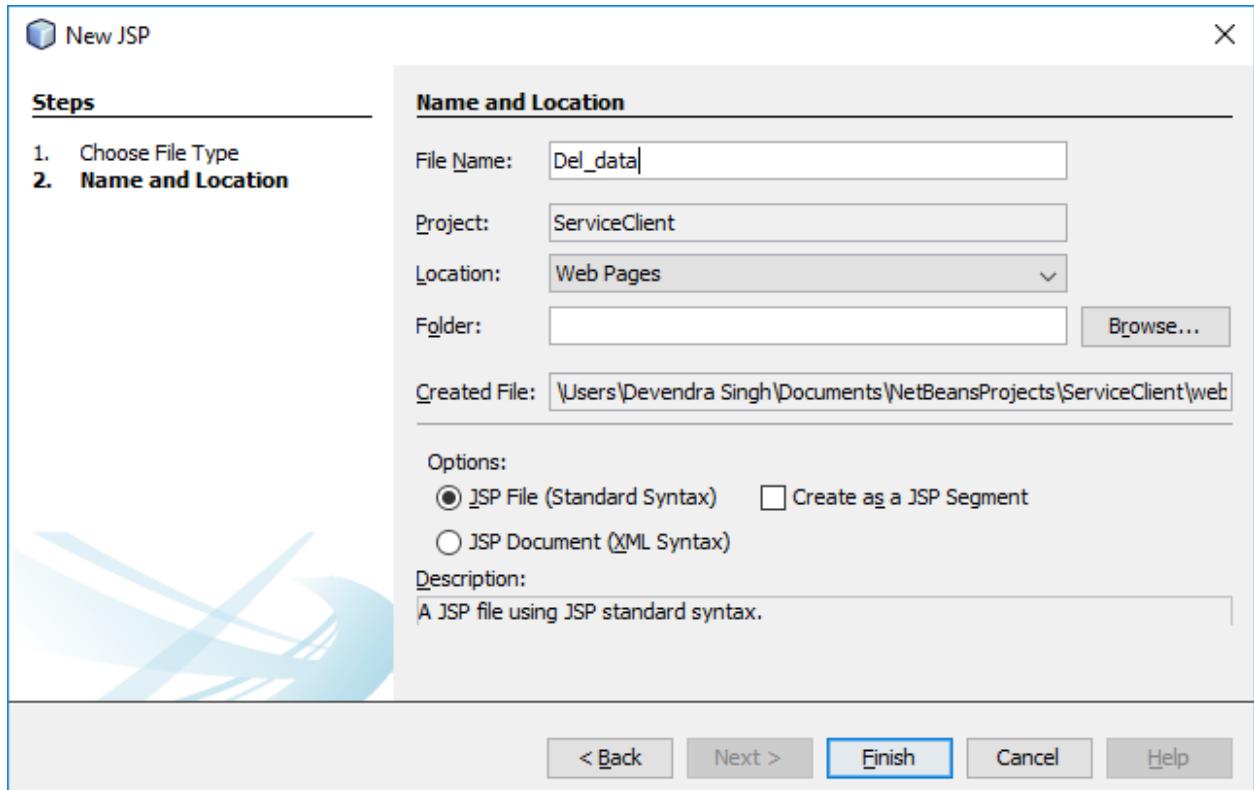
Right click on ServiceClient -> New -> Other



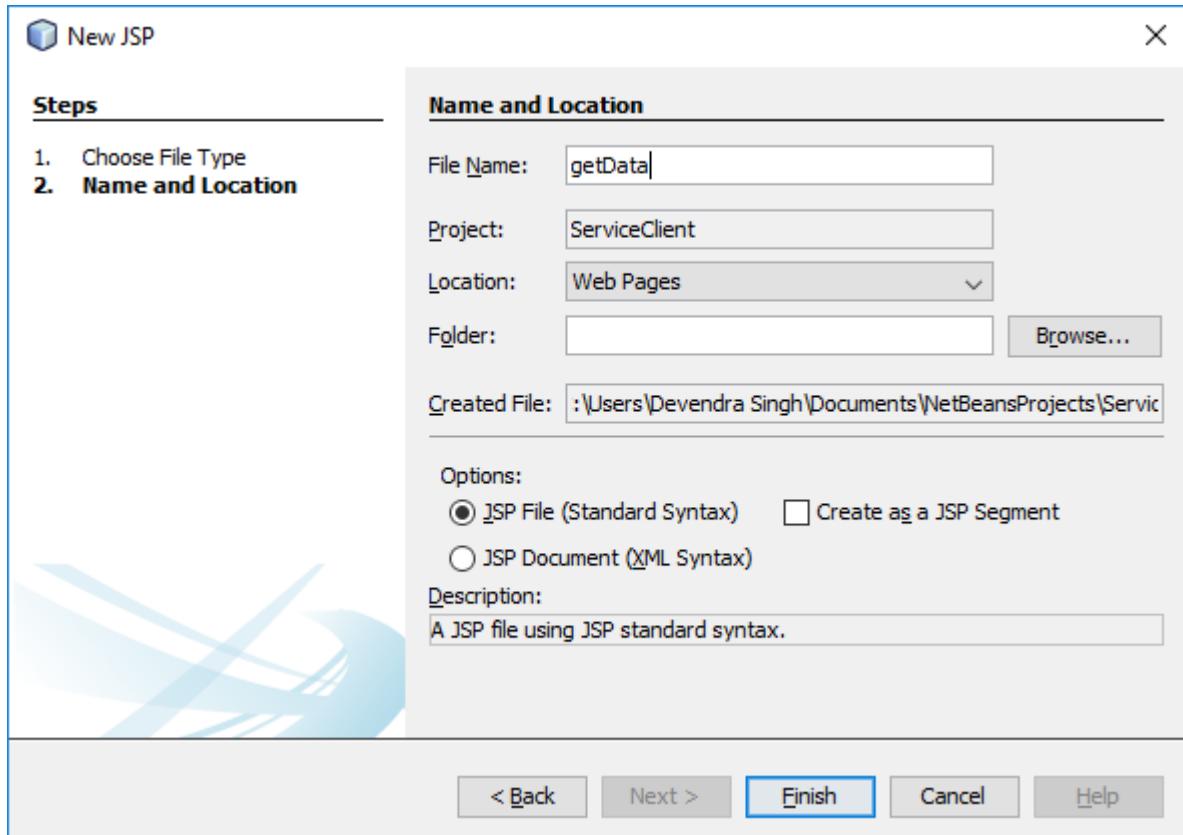
31. Select Web in Categories section -> Select JSP and click on Next button.



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

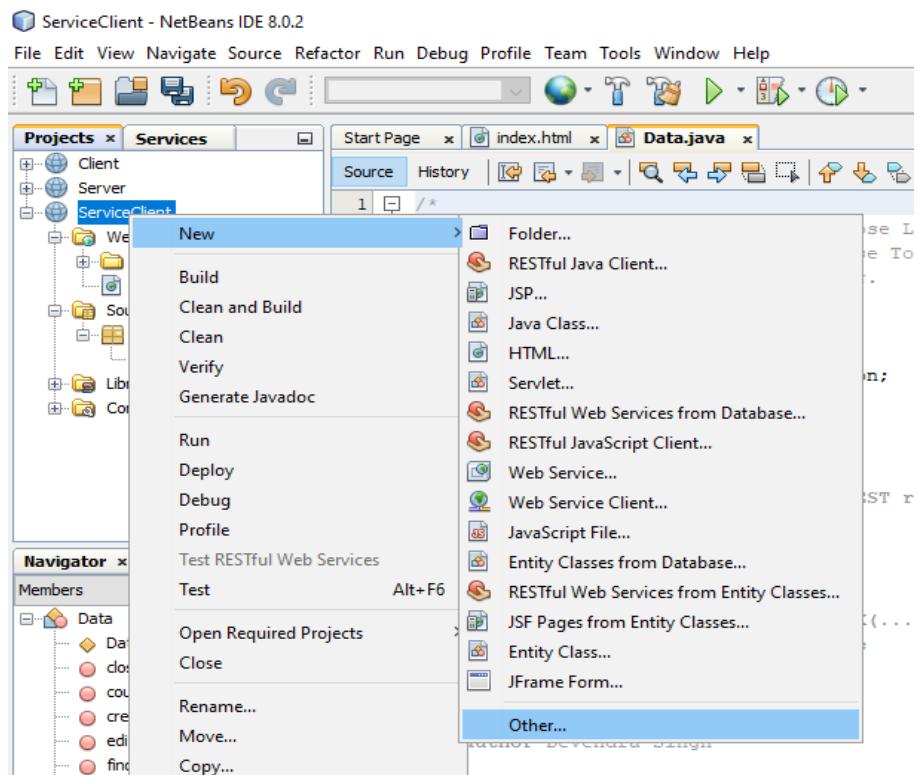


33. Now create one more JSP file by follow the step number 30, 31 & 32. But File Name will be getData.

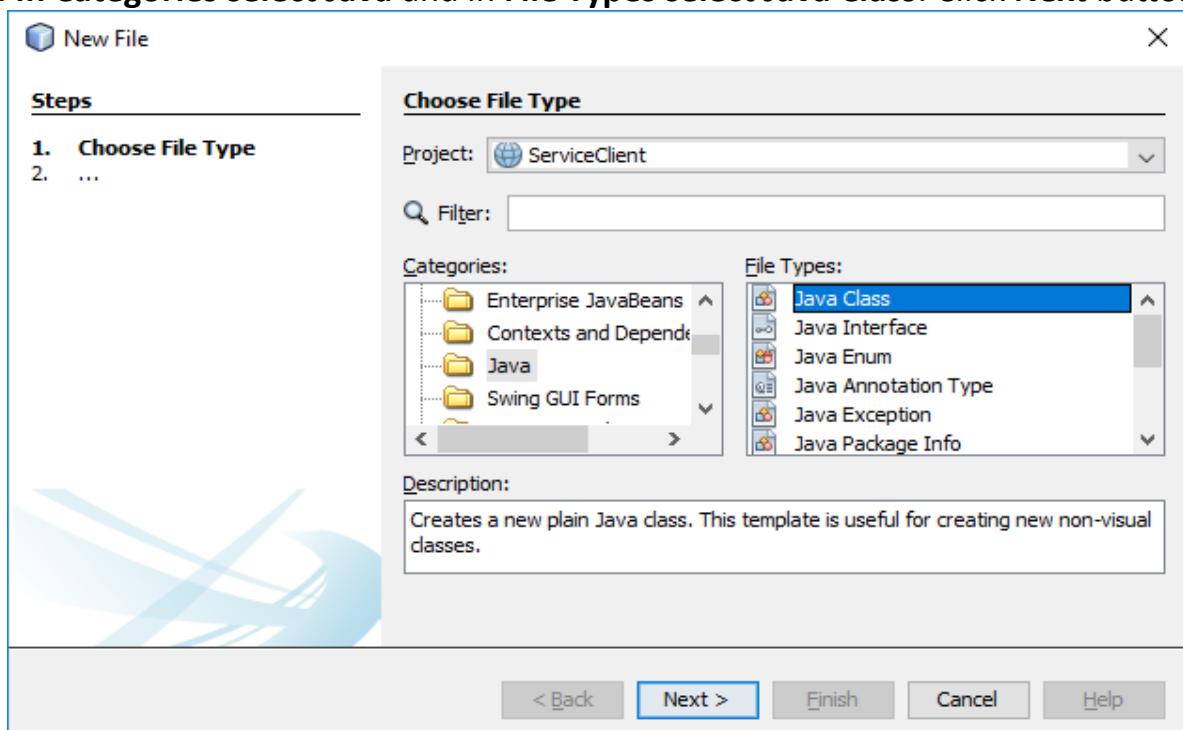


34. Now create a Java class.

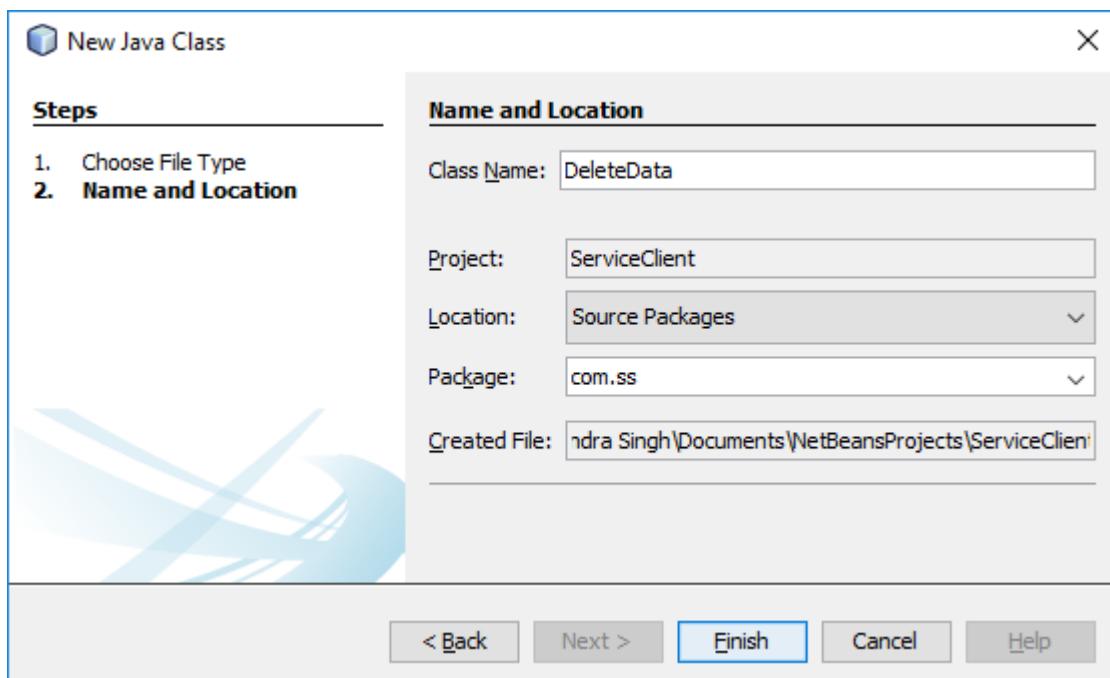
Right click on ServiceClient -> New -> Other



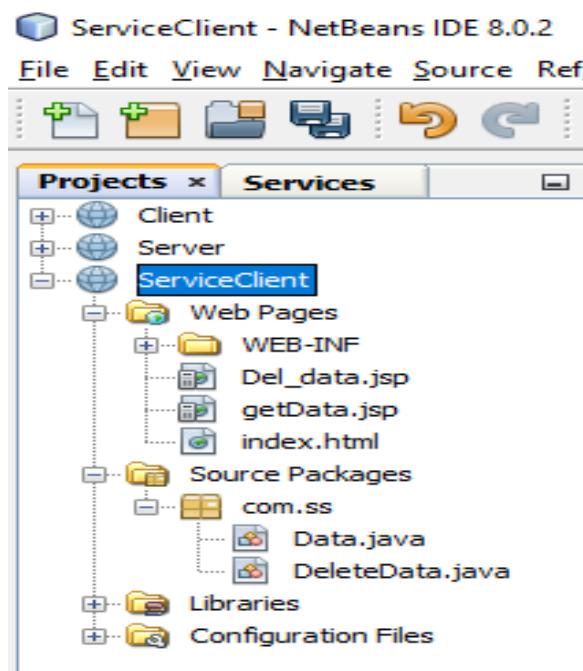
35. In Categories select Java and in File Types select Java Class. Click Next button.



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 multiple tabs at the top: Start Page, getData.jsp, getData.html, and index.html. The index.html tab is active. The code editor displays the following HTML structure:

```

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>
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.");
}

```

```

1 /*
2 * To change this license header, choose License !
3 * To change this template file, choose Tools | T
4 * and open the template in the editor.
5 */
6 package com.ss;
7
8 /**
9 *
10 * @author Devendra Singh
11 */
12 public class DeleteData {
13 public static void deldata(String id) {
14 String a = id;
15 Data ob = new Data();
16 ob.remove(a);
17 System.out.println("Data is deleted.");
18 }
19 }
20
21

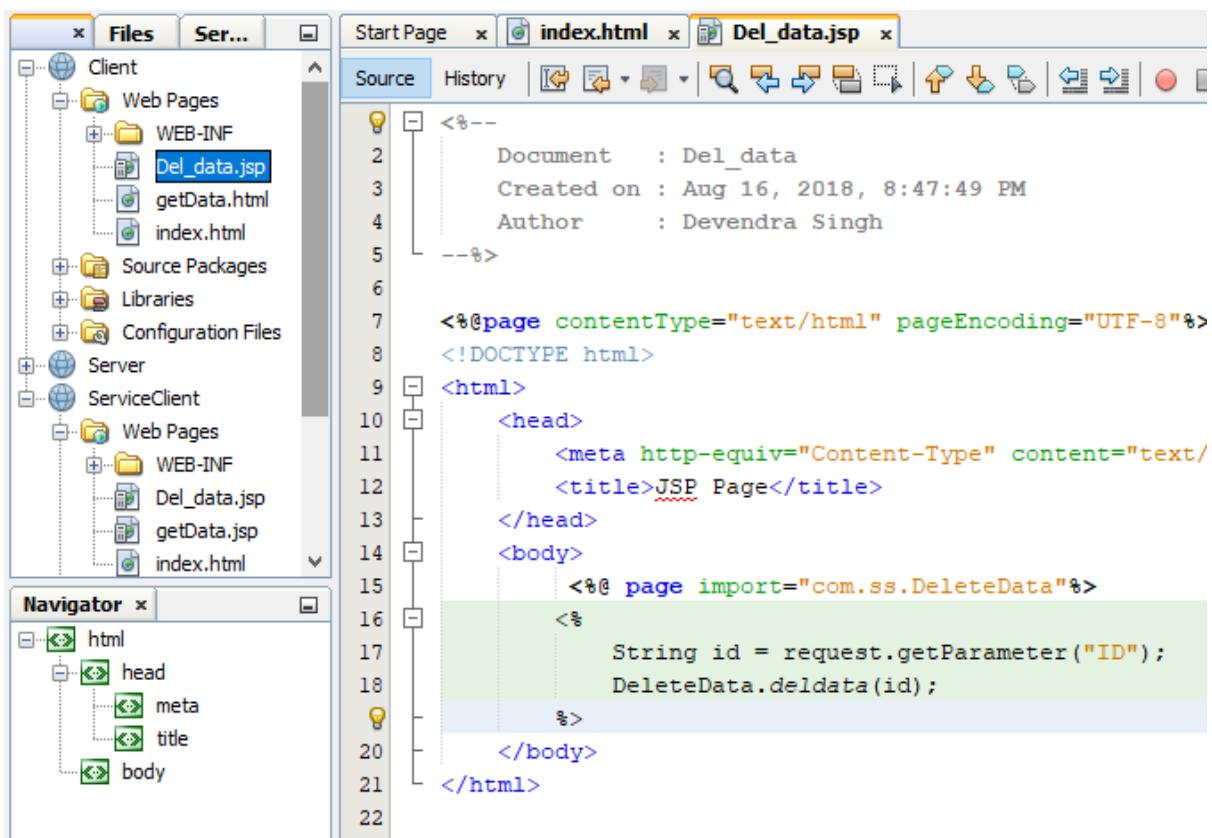
```

40. Now open the Del\_data.jsp file and replace the contents of body with the following code.

```

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

```



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

```

<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>

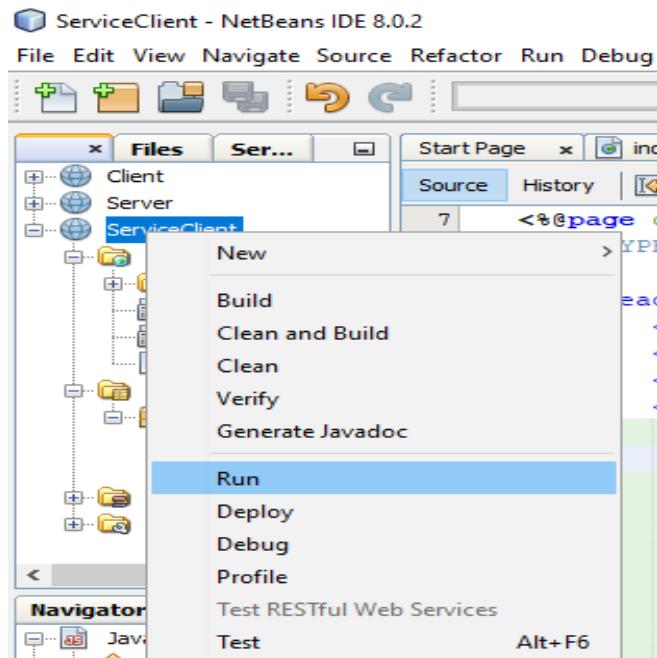
</head>
<body>
 <table id="myTable">
 <tr>
 <th> ID</th>
 <th>NAME</th>
 </tr>
 </table>

</body>
```

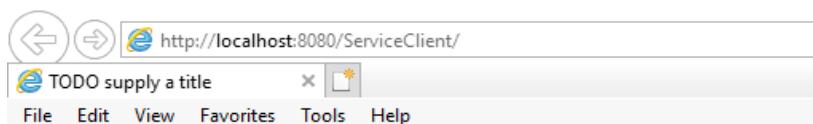
The screenshot shows the NetBeans IDE interface with the following details:

- 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** sections. Under **JavaScript**, there is a **request** node with an **onload()** function. Under  **CSS**, there are **Elements** (with **table**, **td**, and **th** nodes), **Ids** (with **#myTable**), and **Rules** (with **table**, **td**, and **th** nodes).
- Code Editor:** Displays the **getData.jsp** file. The code includes JSP tags, HTML, and CSS styles for a table. It also contains a JavaScript block that uses XMLHttpRequest to make a GET request to <http://localhost:8080/Server/webresources/com.kk.friends/>. The script begins by parsing the JSON response and then iterates through the data to insert rows and cells into a table with ID **#myTable**.

## 42. Now Run the ServiceClient project.



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



## One-way Operation

---

## Request-Response operation

If you will click on Delete Data button, record of entered ID will be deleted.

If you will click on Get Data button Request for the data will be send and it will return the data as response. Hence it is two way 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 table with columns 'ID' and 'NAME'. The table contains the following data:

| ID | NAME       |
|----|------------|
| 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     |

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

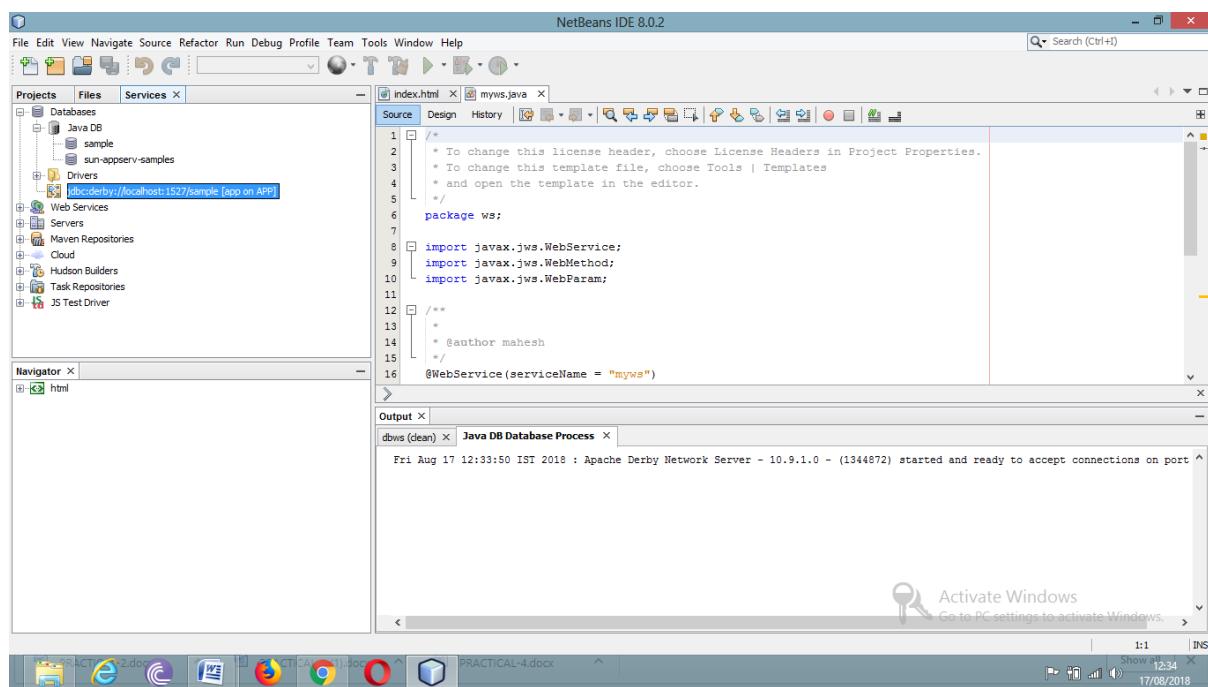
Press Finish

Go to service

Select java DB

Right click on java DB select Start server

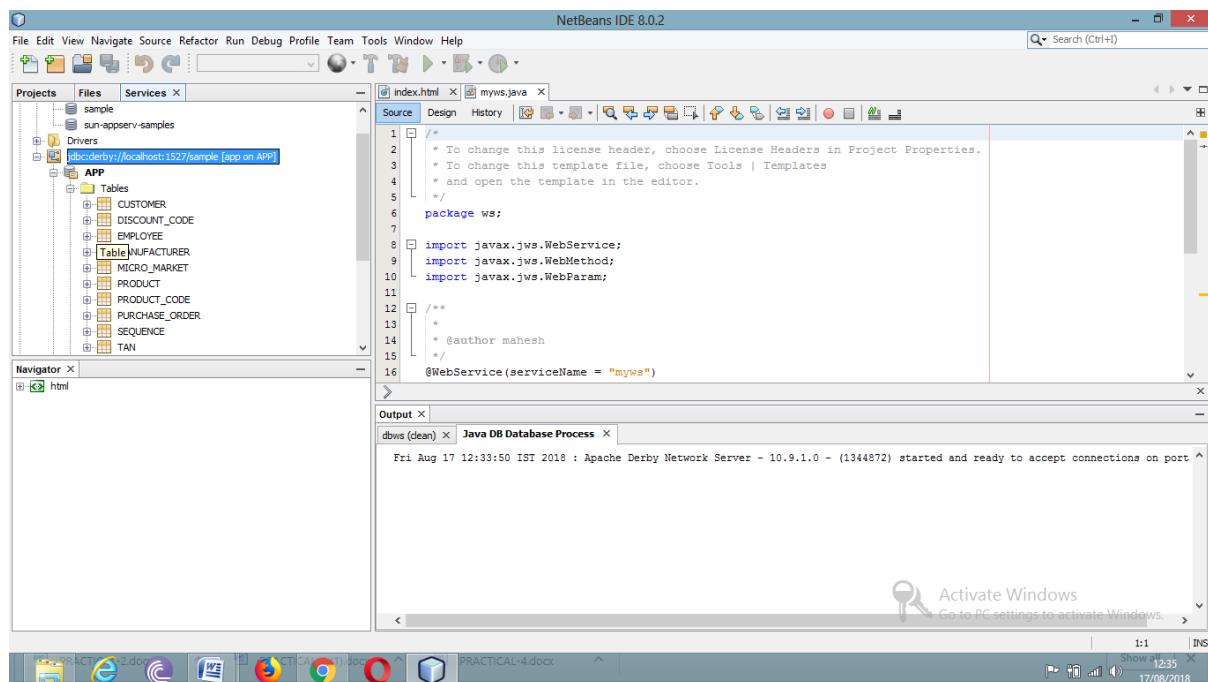
Right click on connection string and select connect



Expand connection string

It will show APP

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



Give name to table as tan

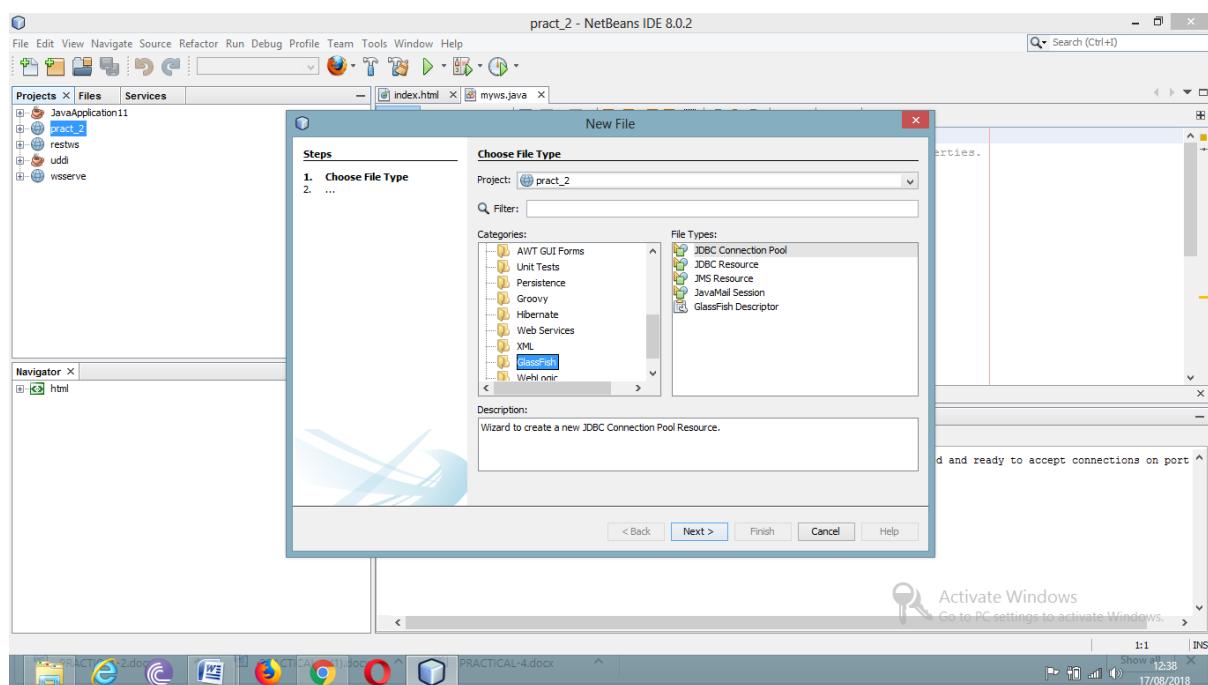
Then select add column

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

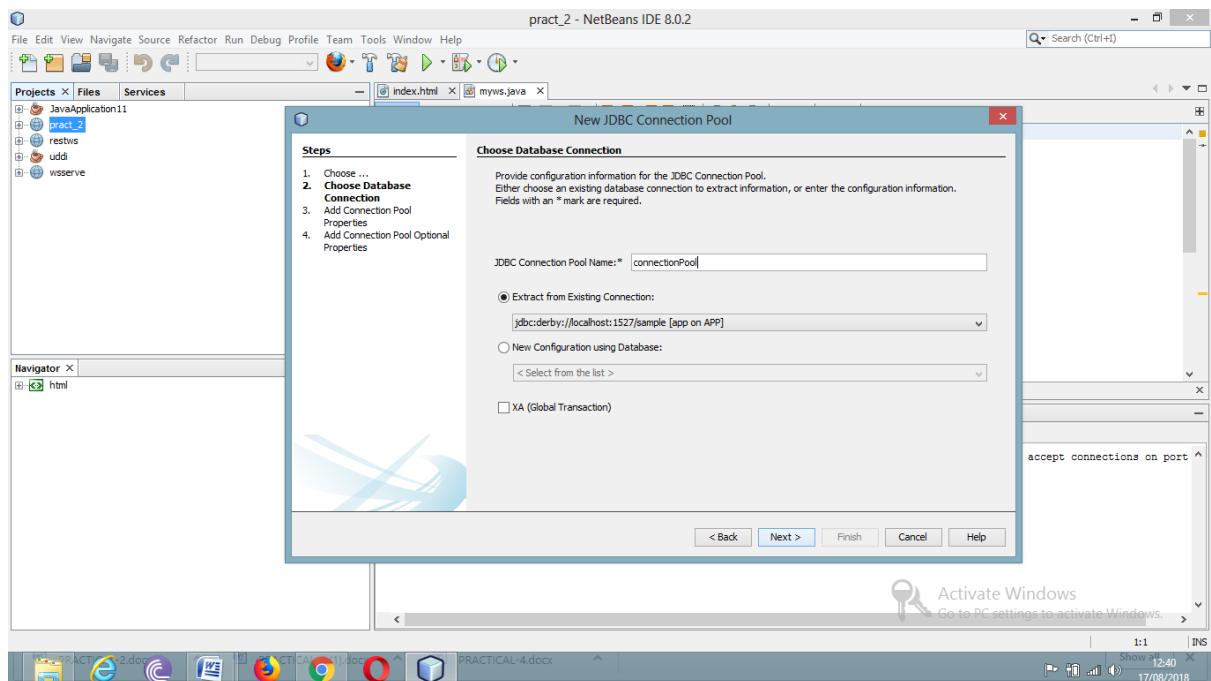
Come back to project

Right click Project name pract\_2 select new select others in it select glassfish server

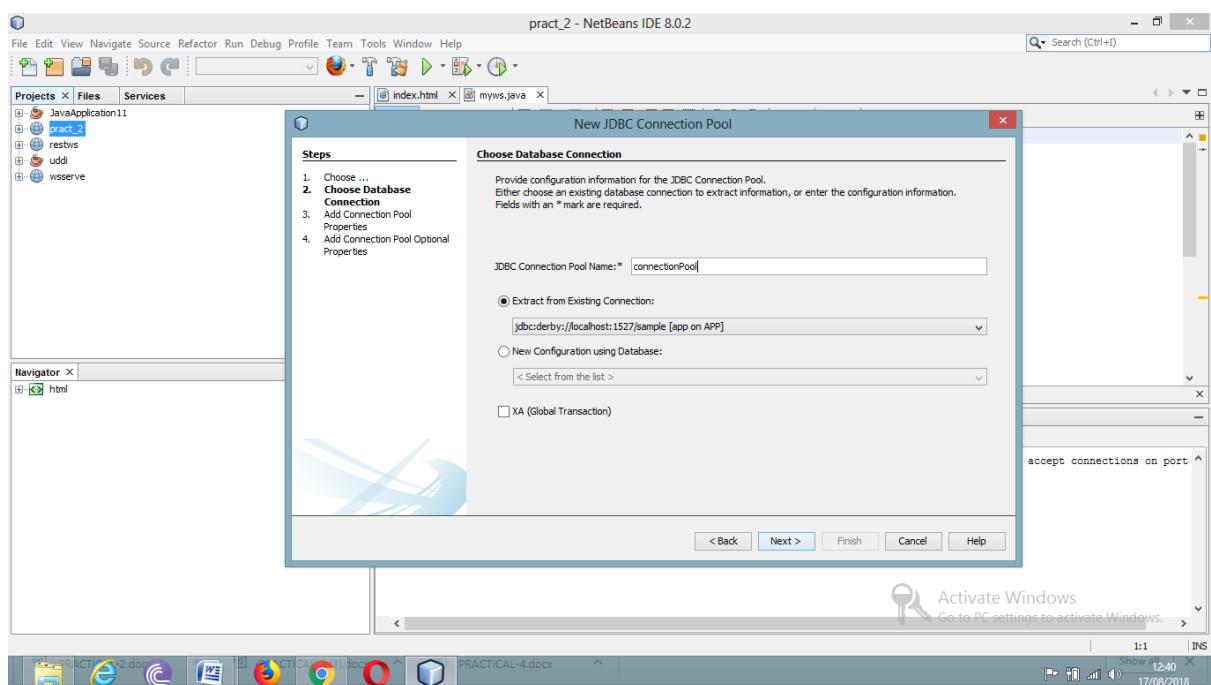
In that select jdbc connection pool



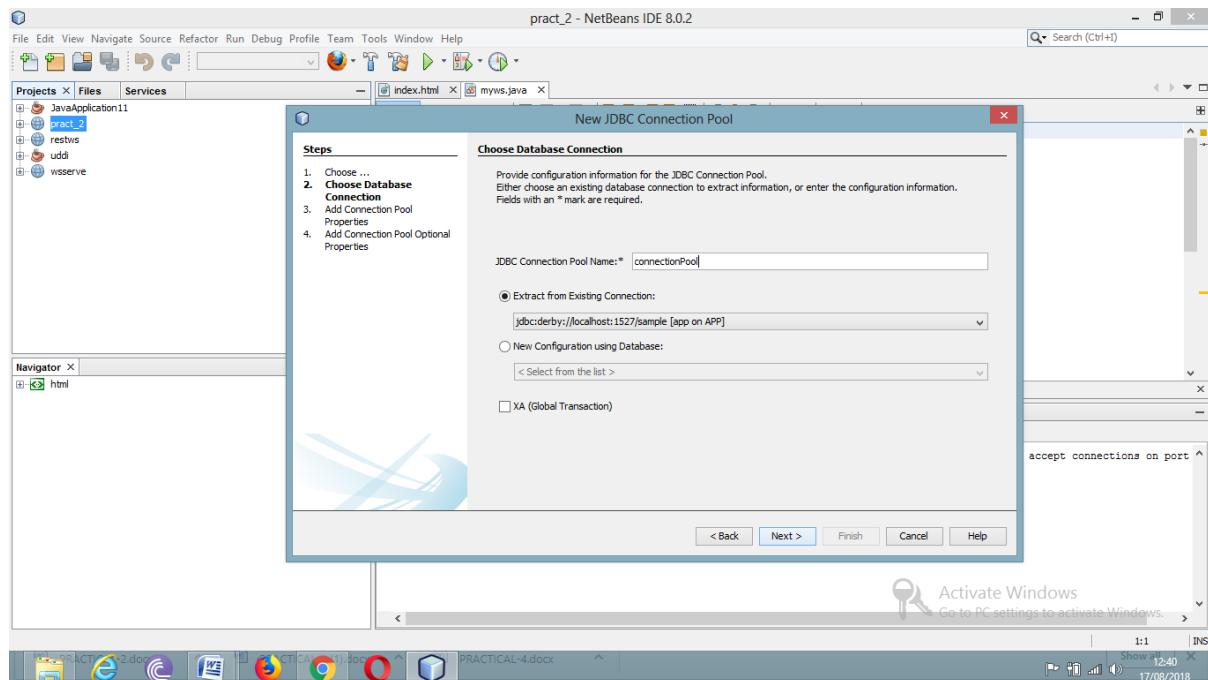
Press next



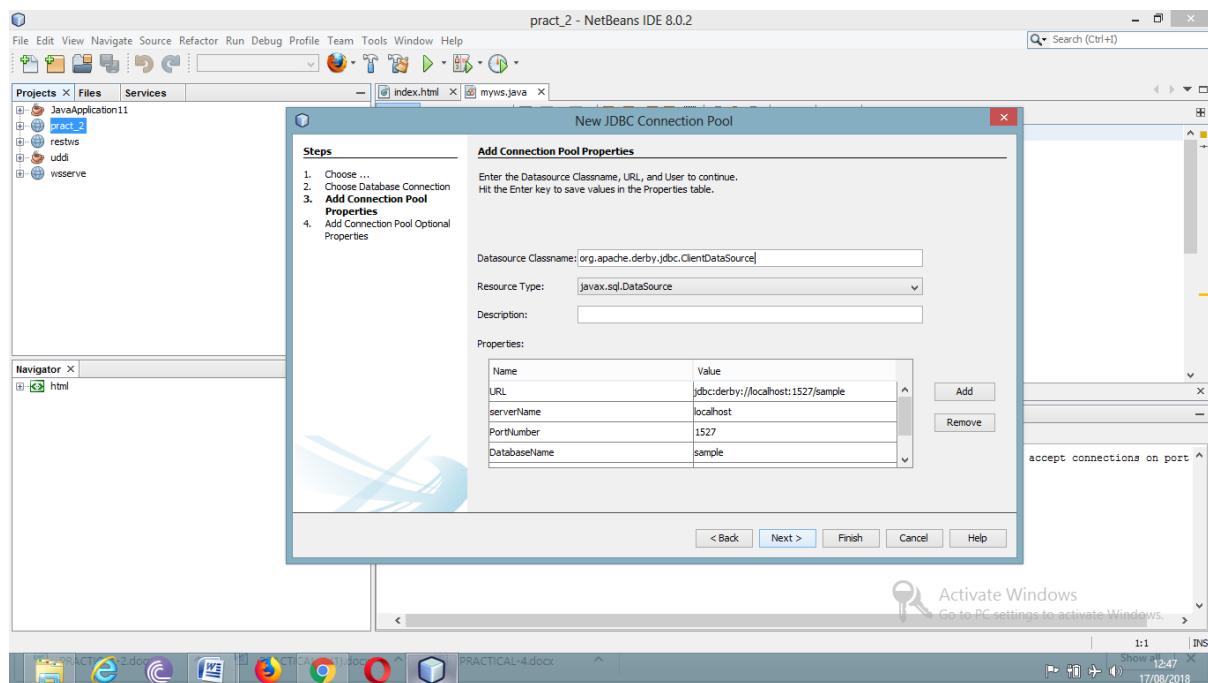
Press next



Press next



Press next

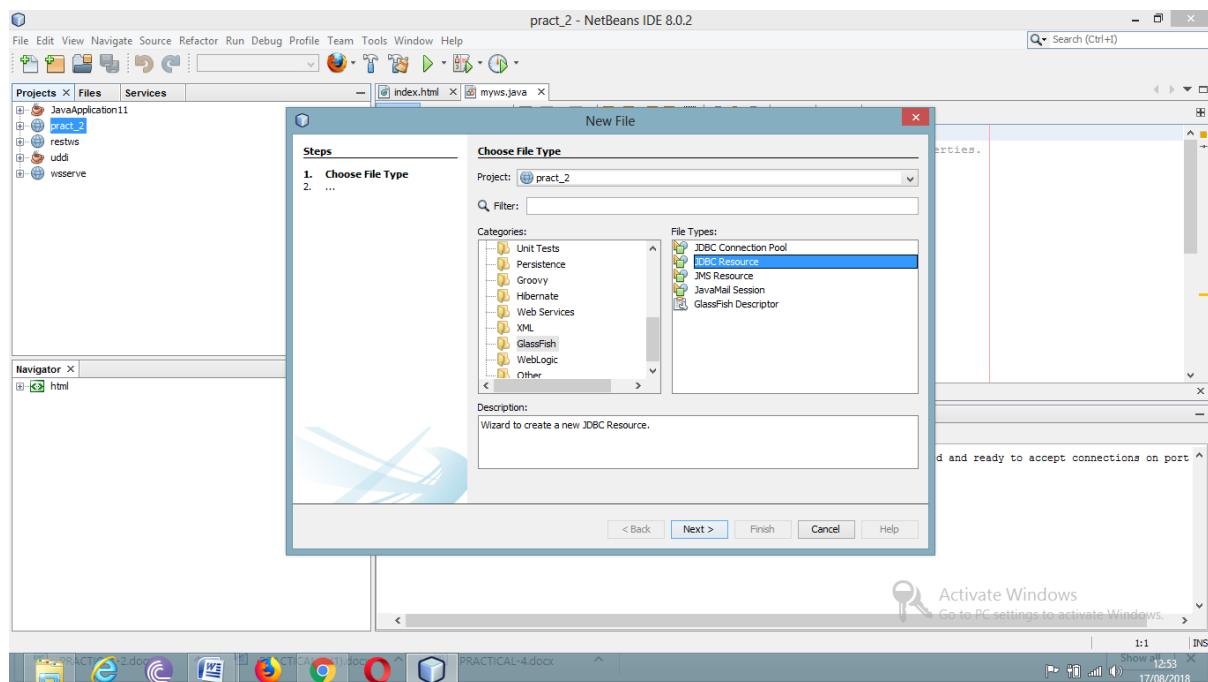


Press next

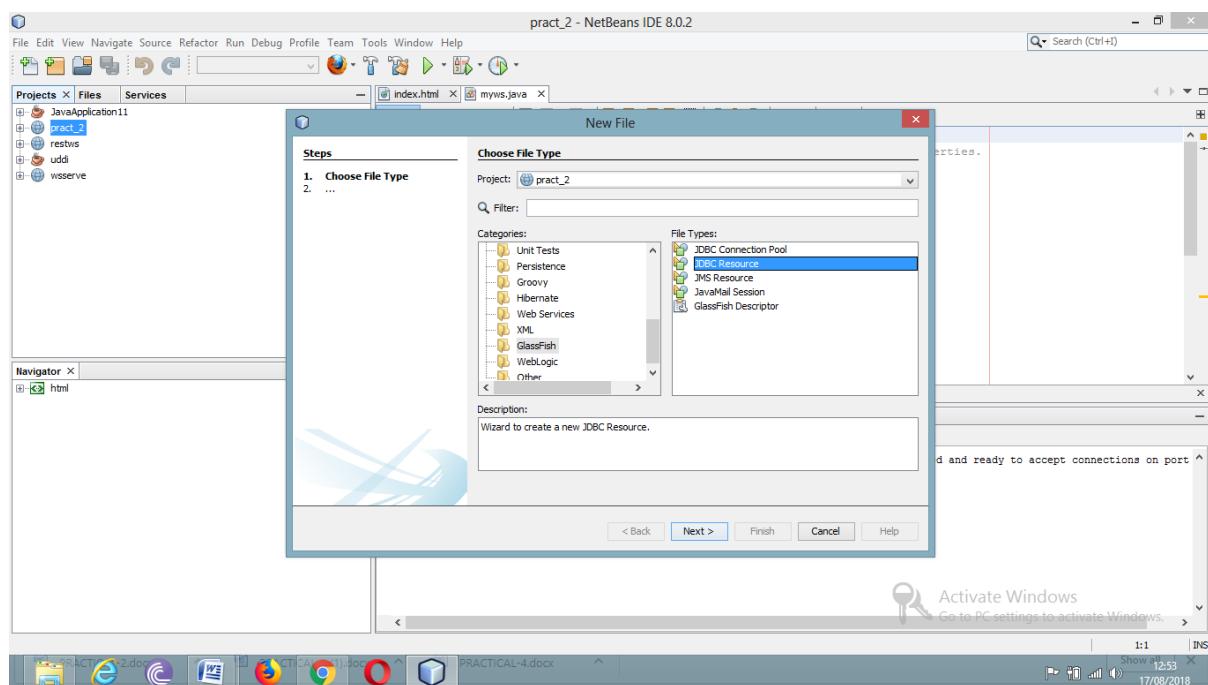
Press finish

Right click project name      select new      select other

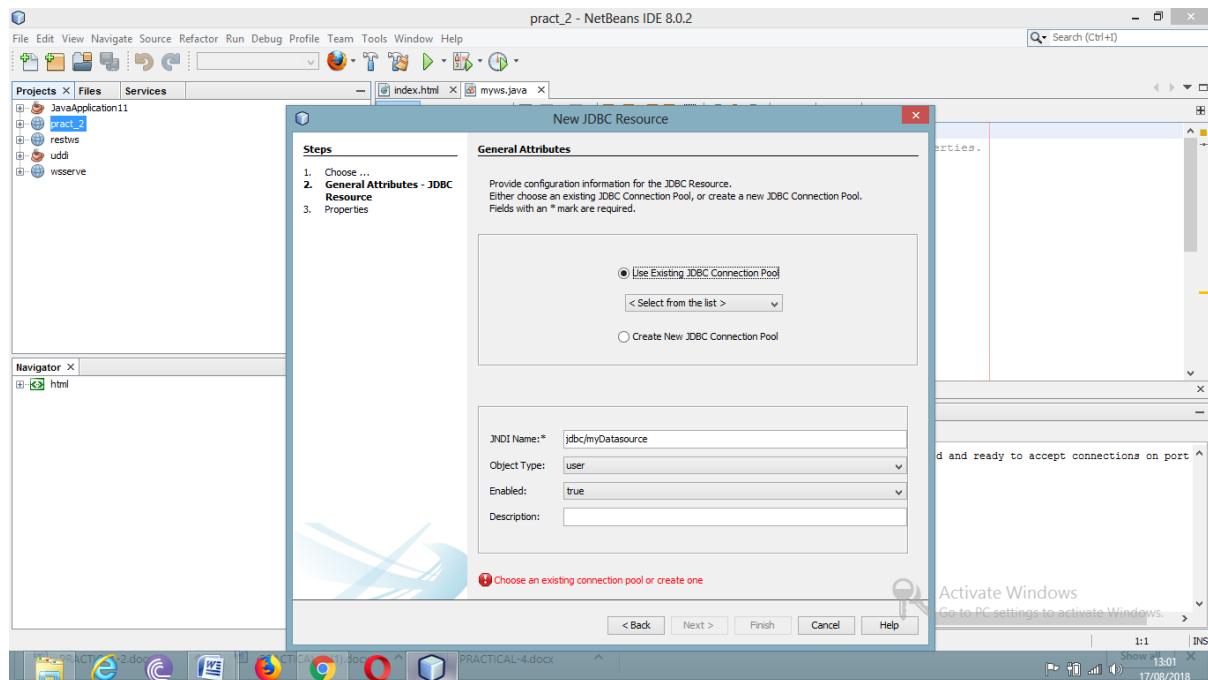
Select glassfish      Select jdbc resource



Press next

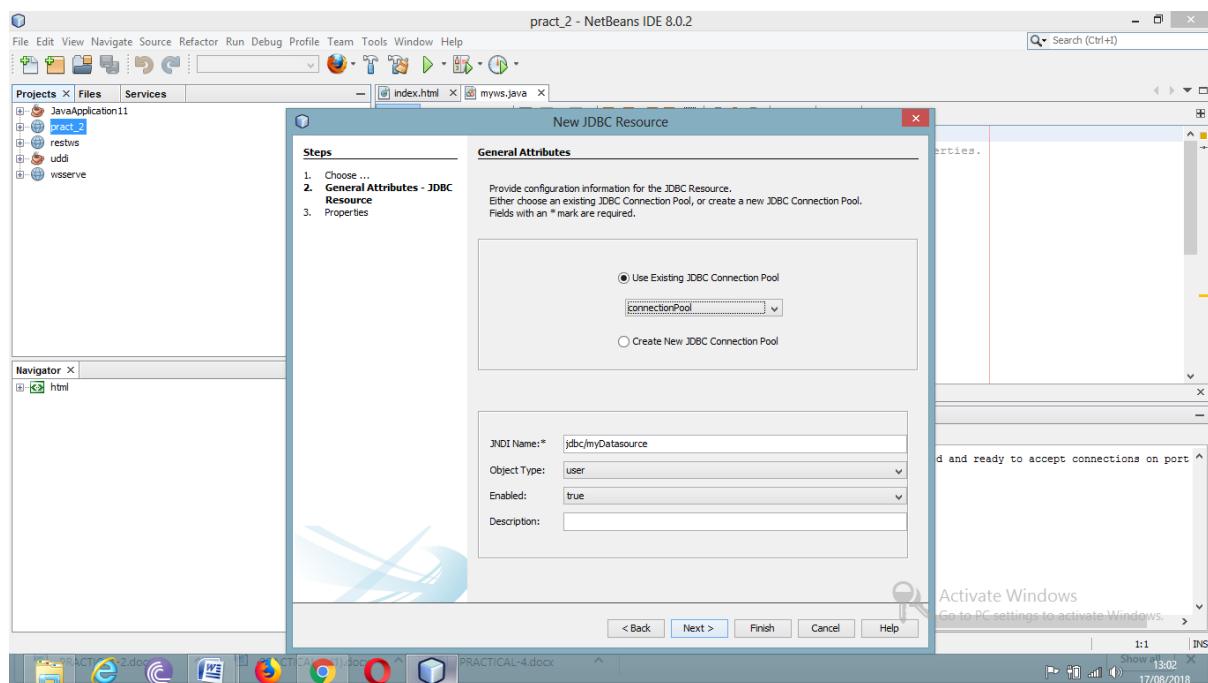


Press next



Click on existing connection pool

Select from combobox connection pool



Change JNDI name to jdbc/mydb

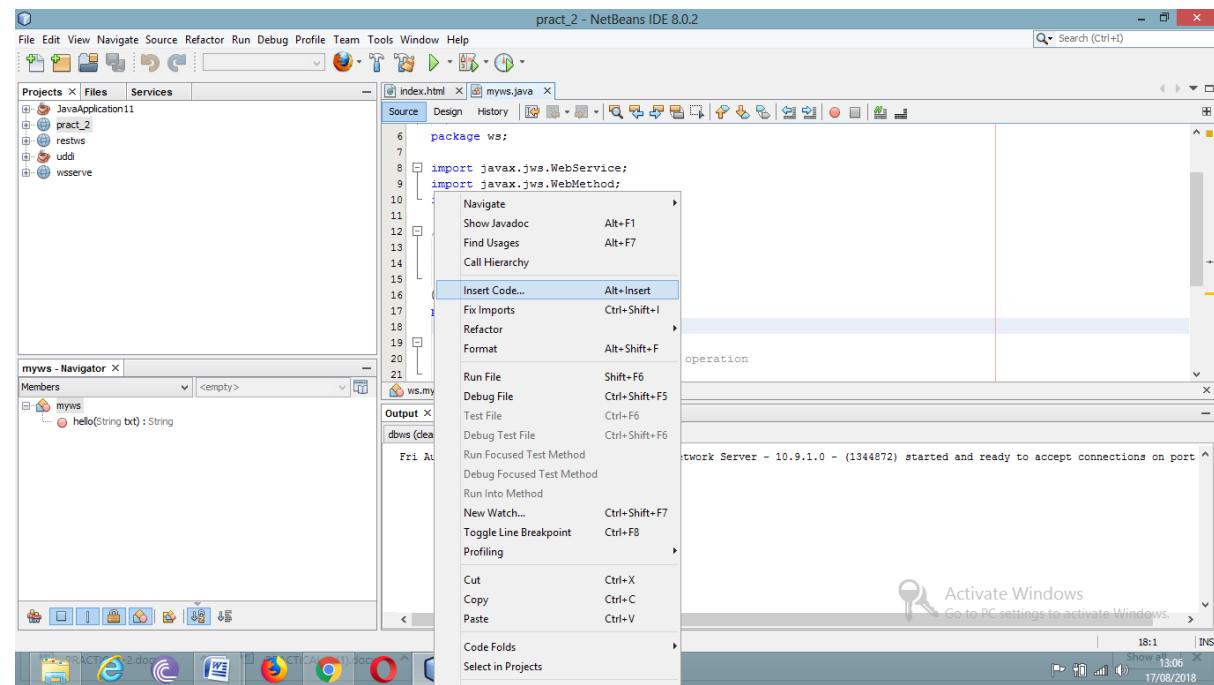
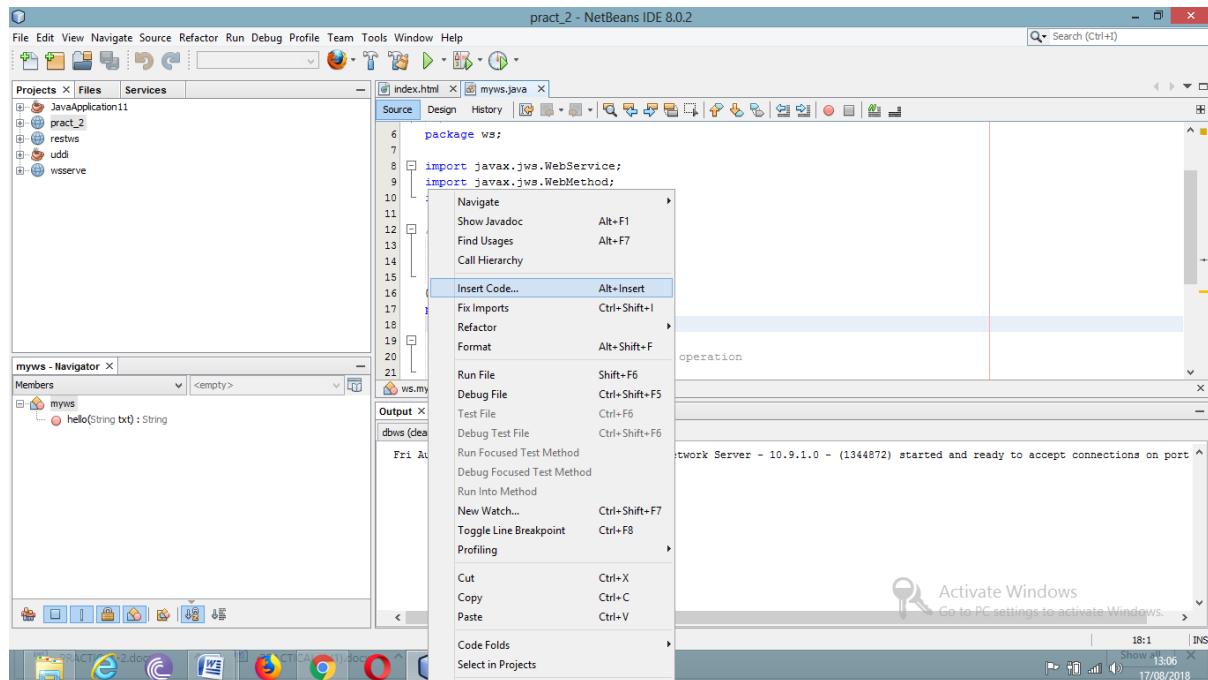
Press next

Press finish

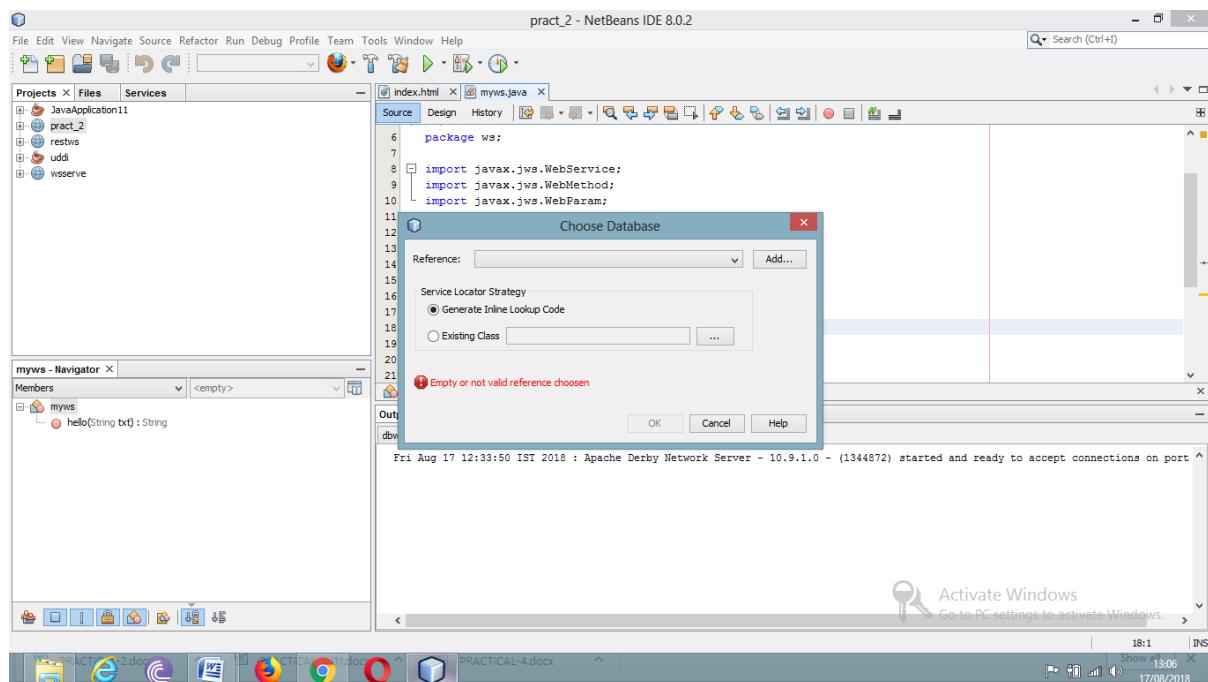
It will open file myws.java

Right click on line number 18 (inside class myws)

Select insert code

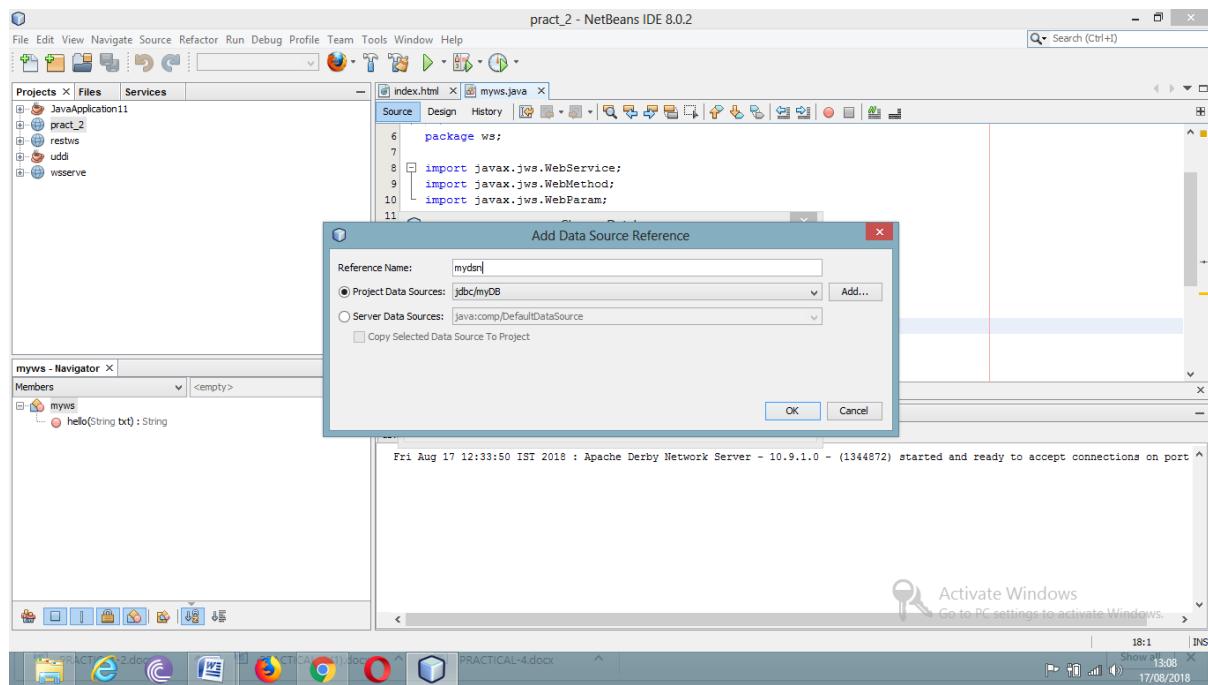


Select use database



Click on add

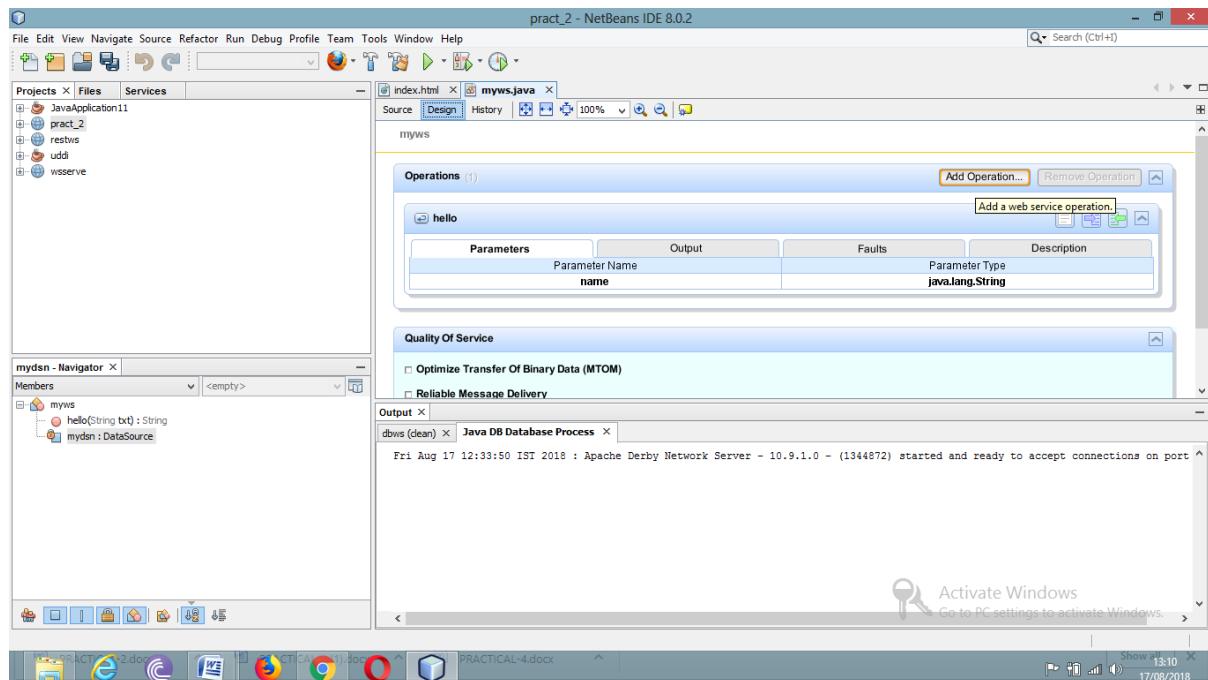
Give any name I have given my dsn



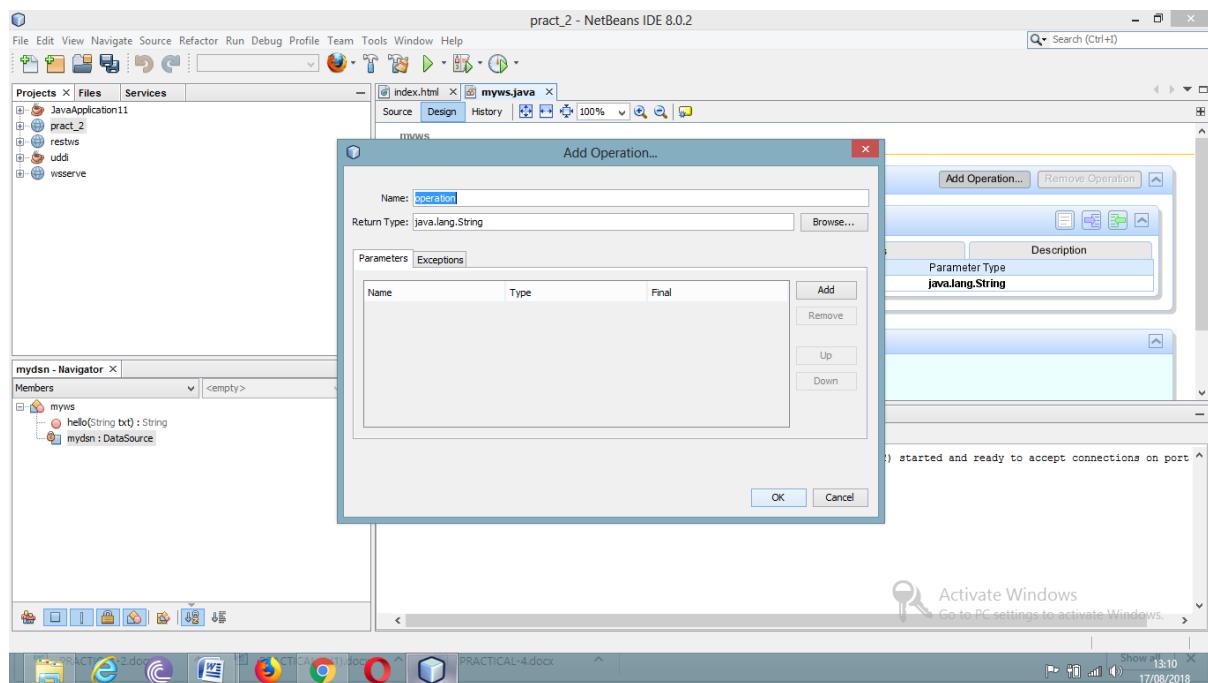
Press ok press ok

Now click on design

Select add operation



It will open



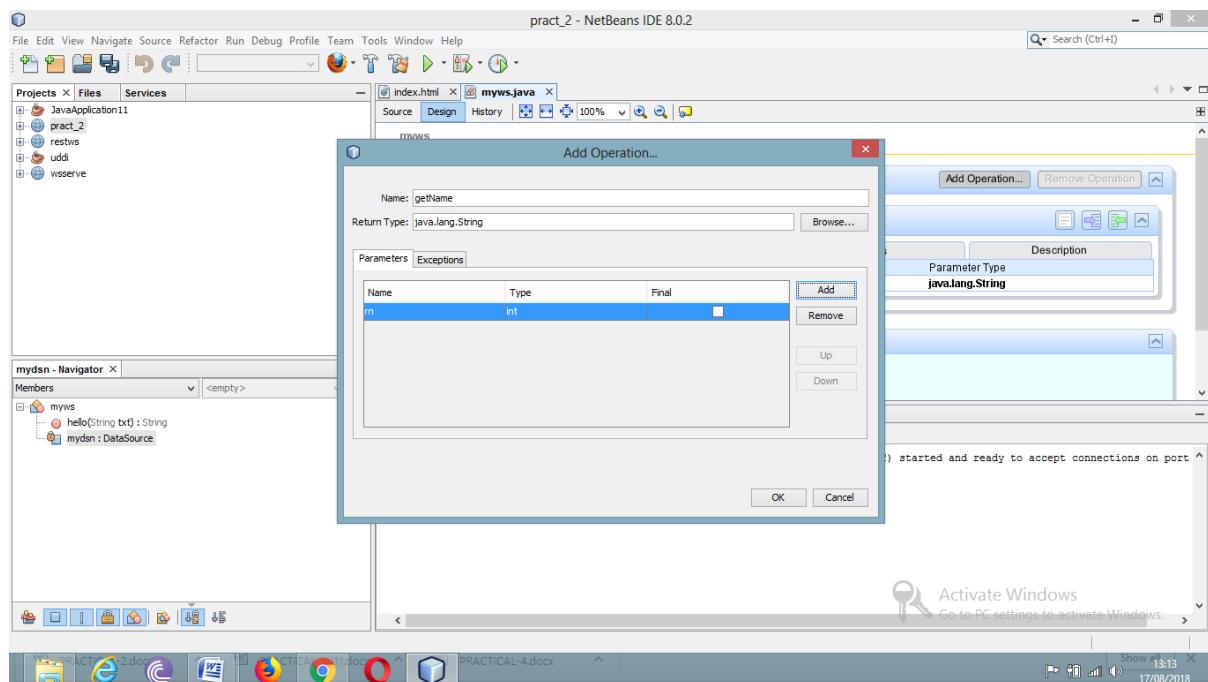
We want a two way web service which takes rn from user and return name

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



Press ok

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

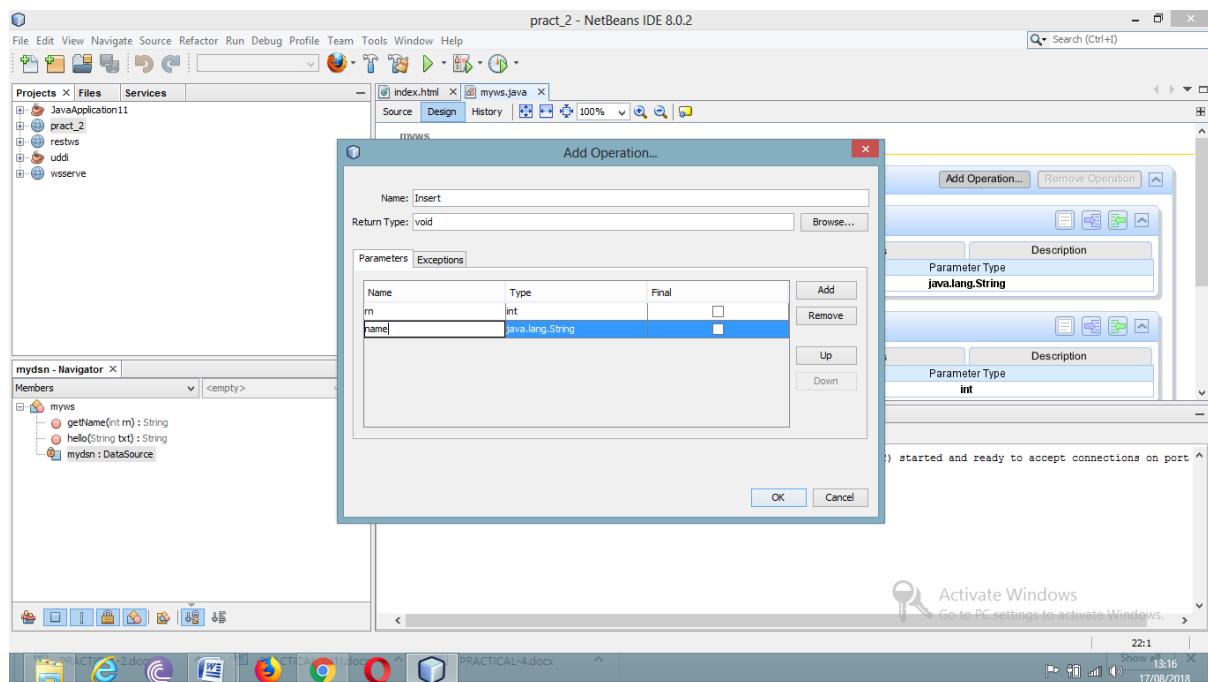
So again click on add operation give name as insert and return type as void

Click on add

Then give parameter as rn with type as int

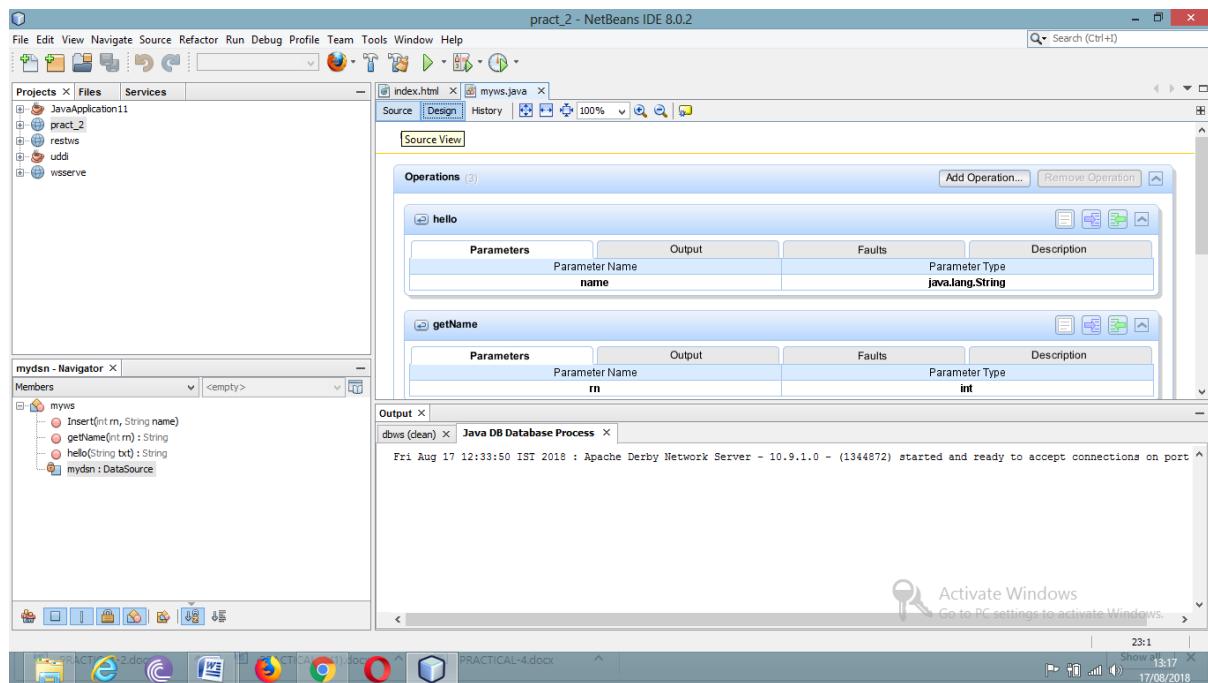
Again press add

Give parameter as name and type as string

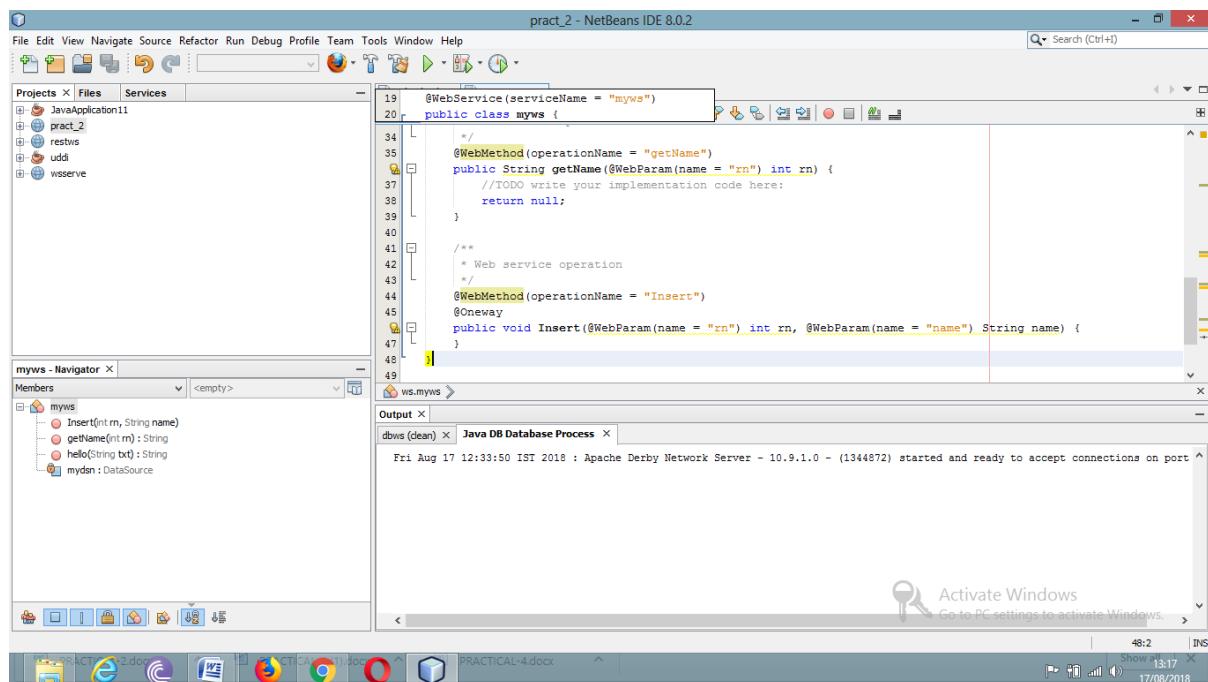


Press ok

Now click on source

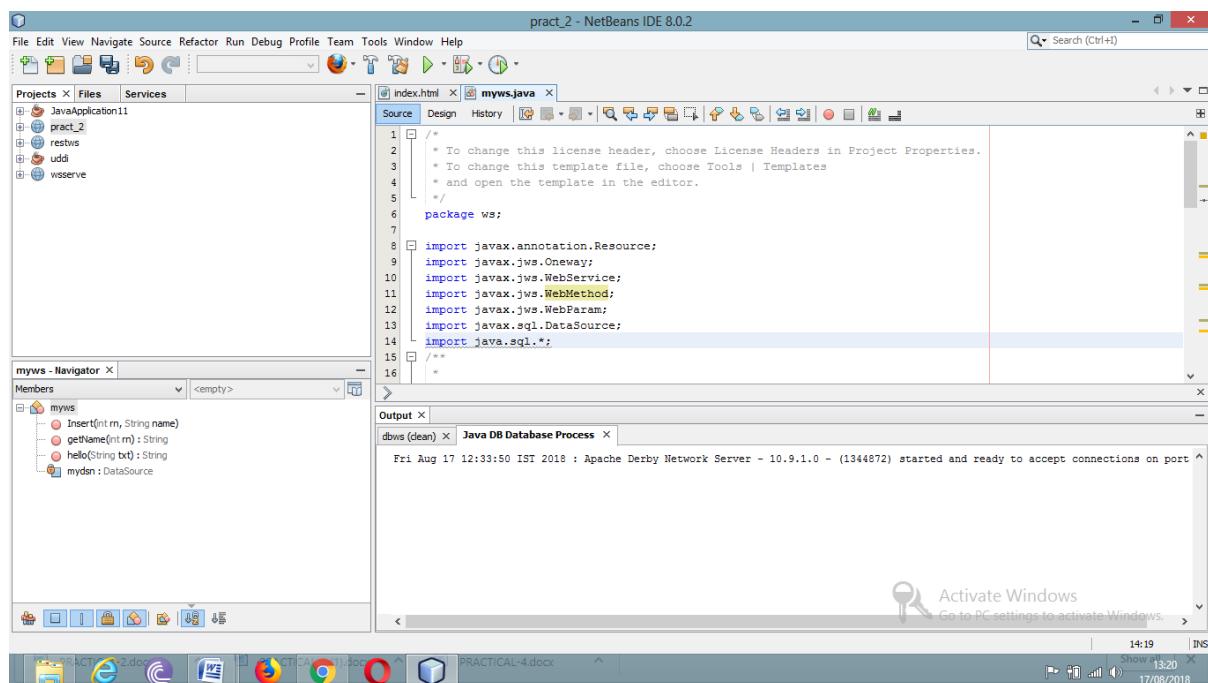


It will show you code (Functions you have created)



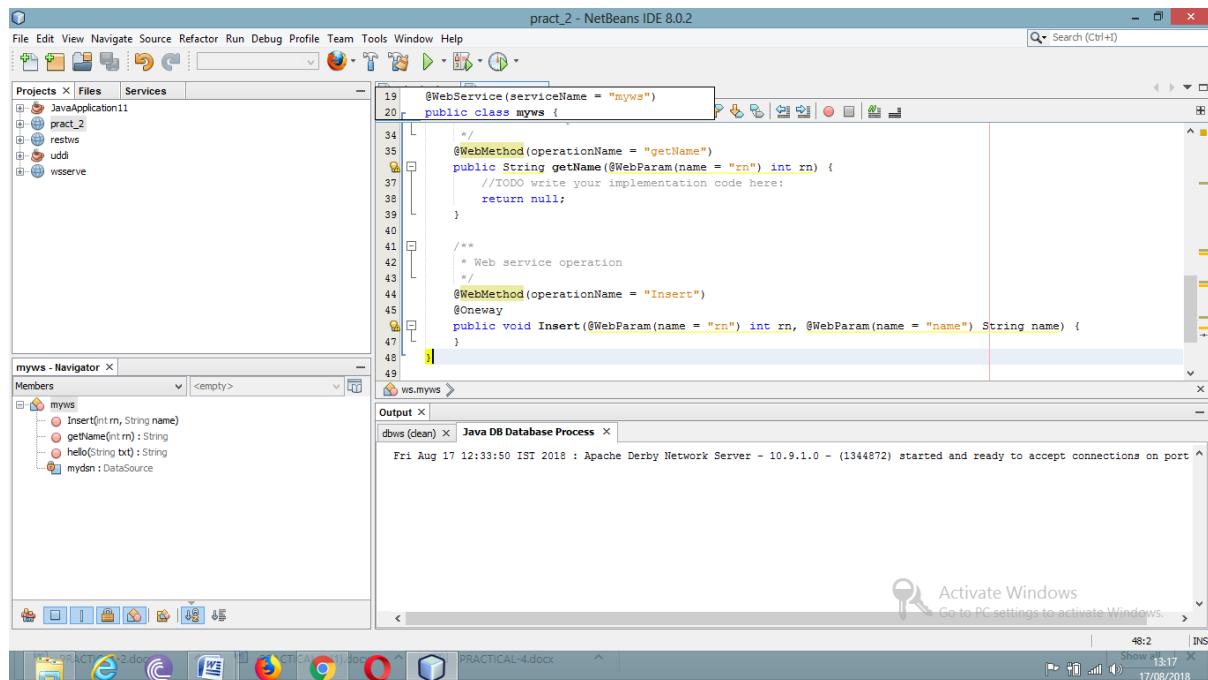
As we are working with database so we need to import java.sql.\*;

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”



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)

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)

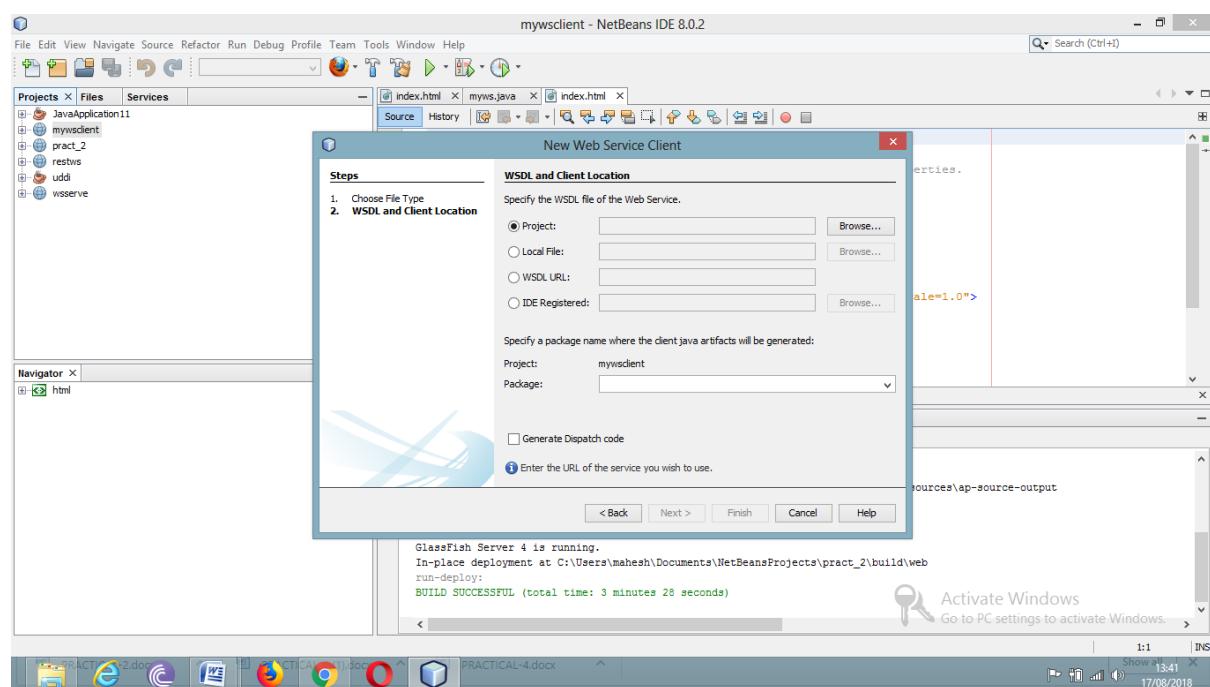
Select new project select new select web java

Give it name mywsclient Press next Press finish

Now right click on project name mywsclient

select new

select web service client

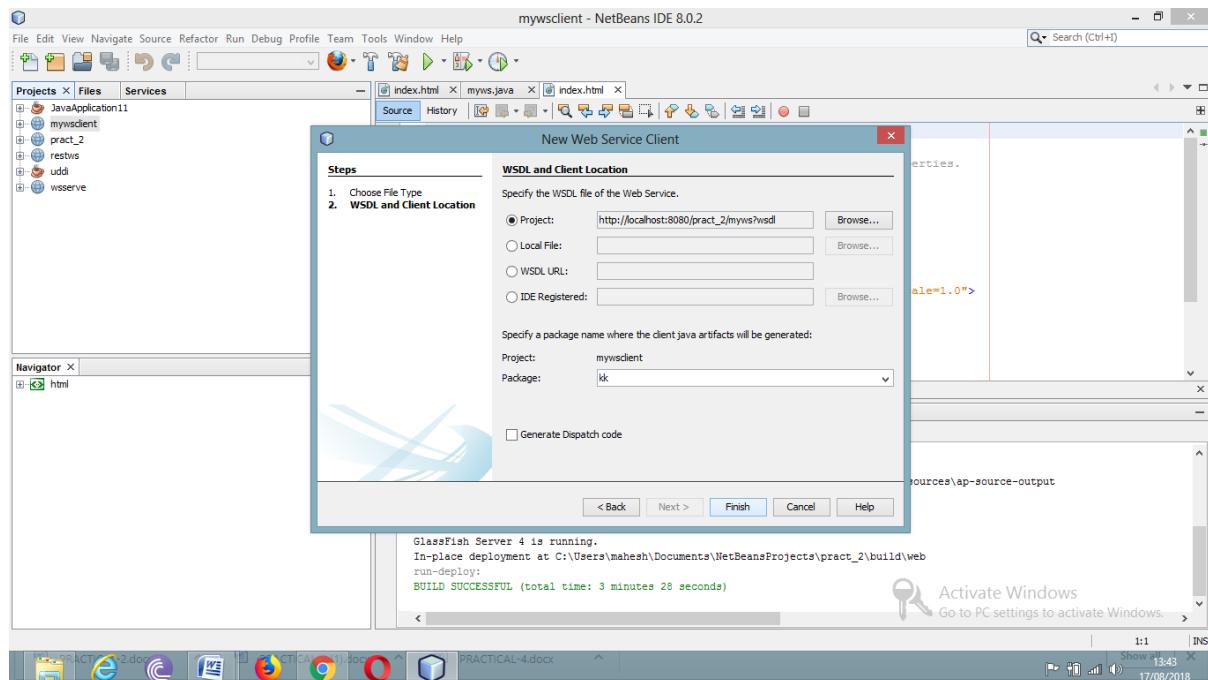


click on Project then select brows

expand pract\_2

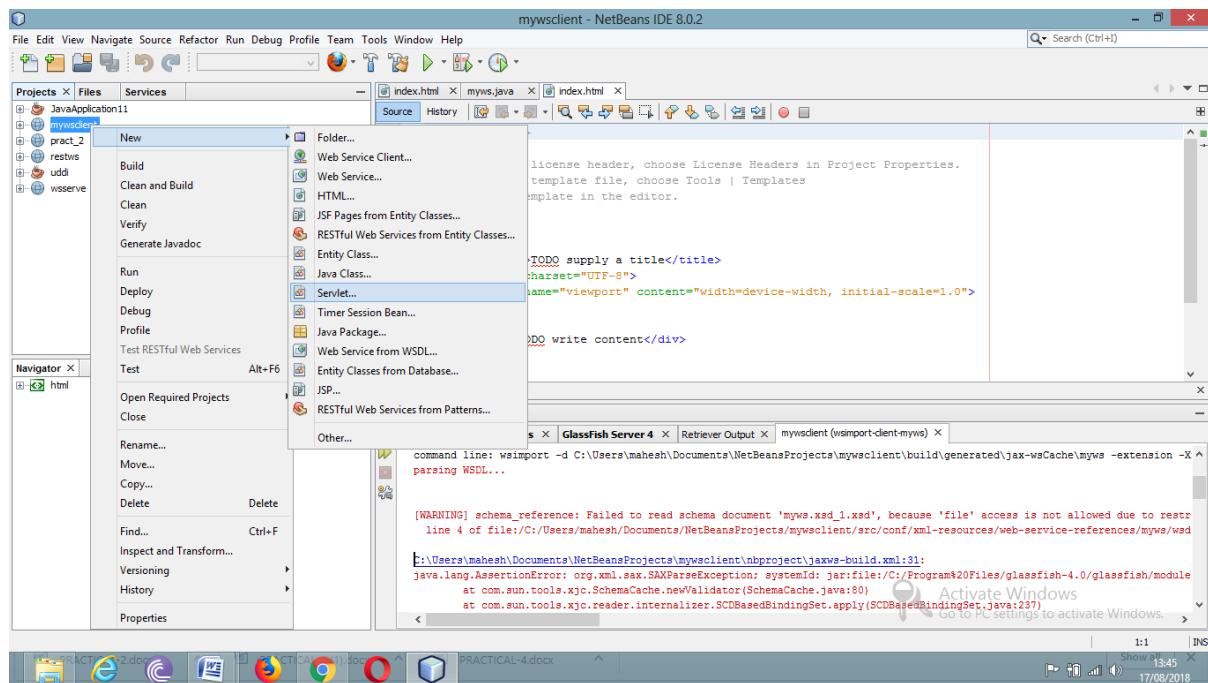
select myws and press ok

Give package name kk



Press finish

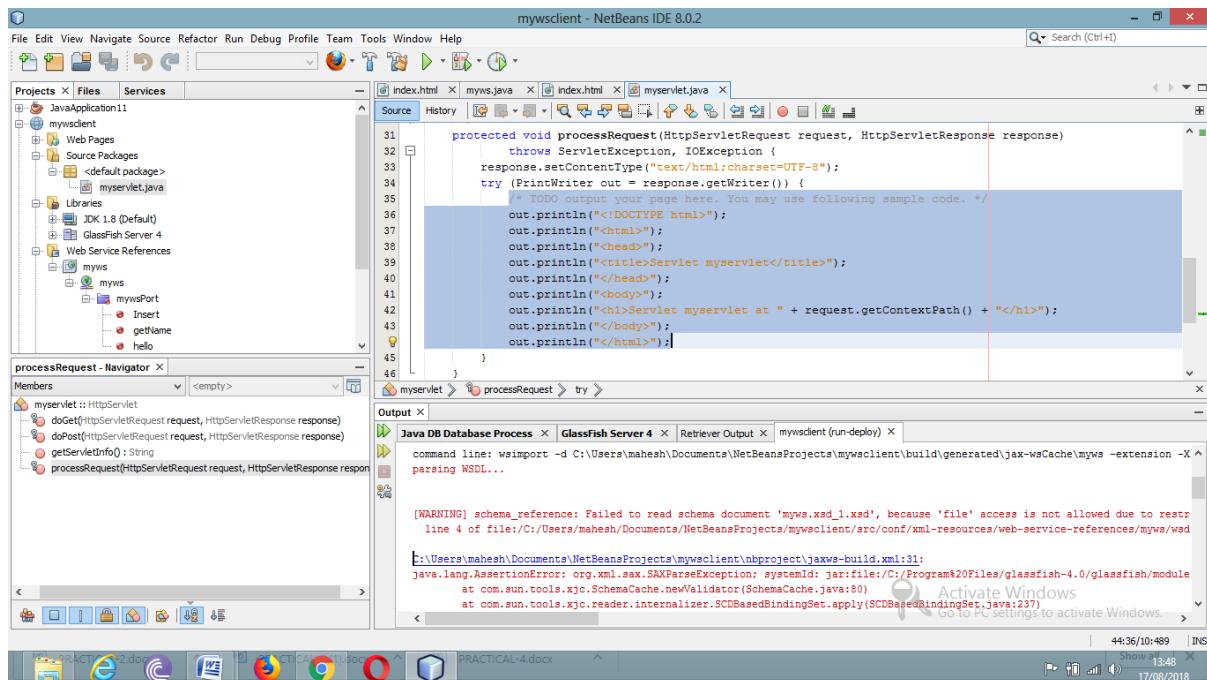
Now right click on mywsclient project select new select servlet



Give servlet name as myservlet

Press finish

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



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.

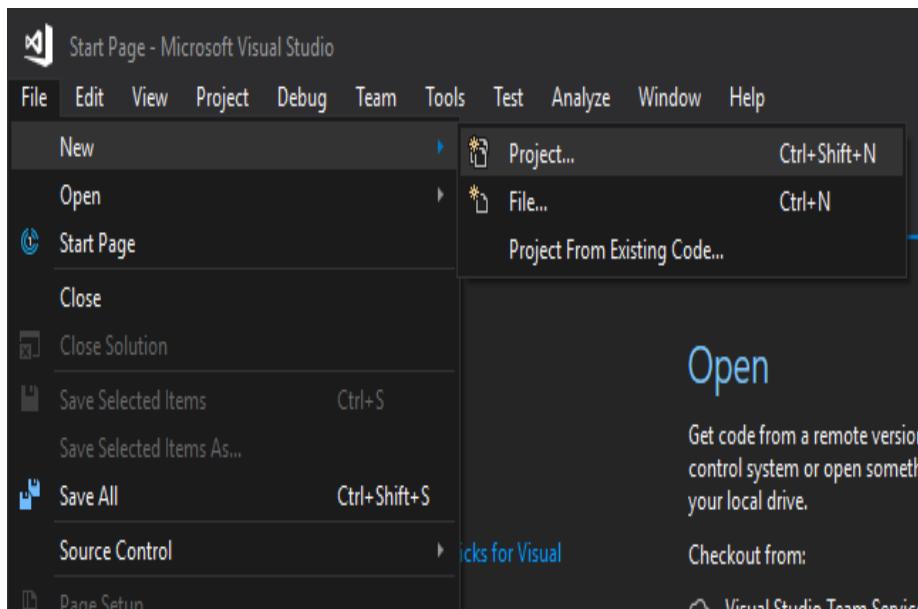
**Requirement:**

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

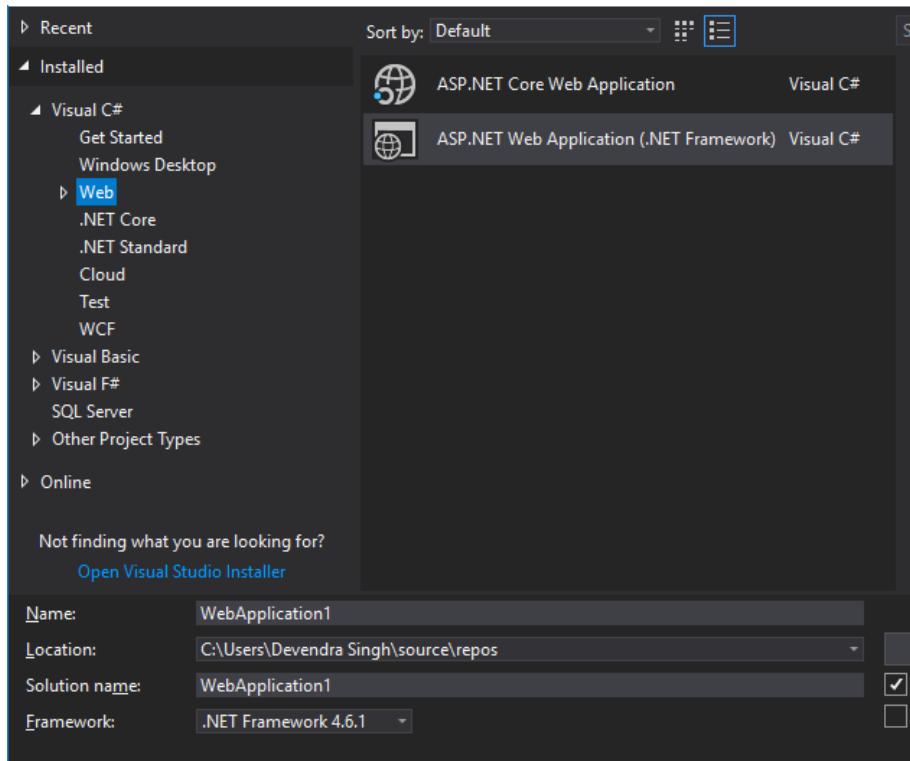
In this practical we are creating Web Service in Visual Studio ant then we will consume it in NetBeans.

**1. Open Visual Studio IDE and click on File.**

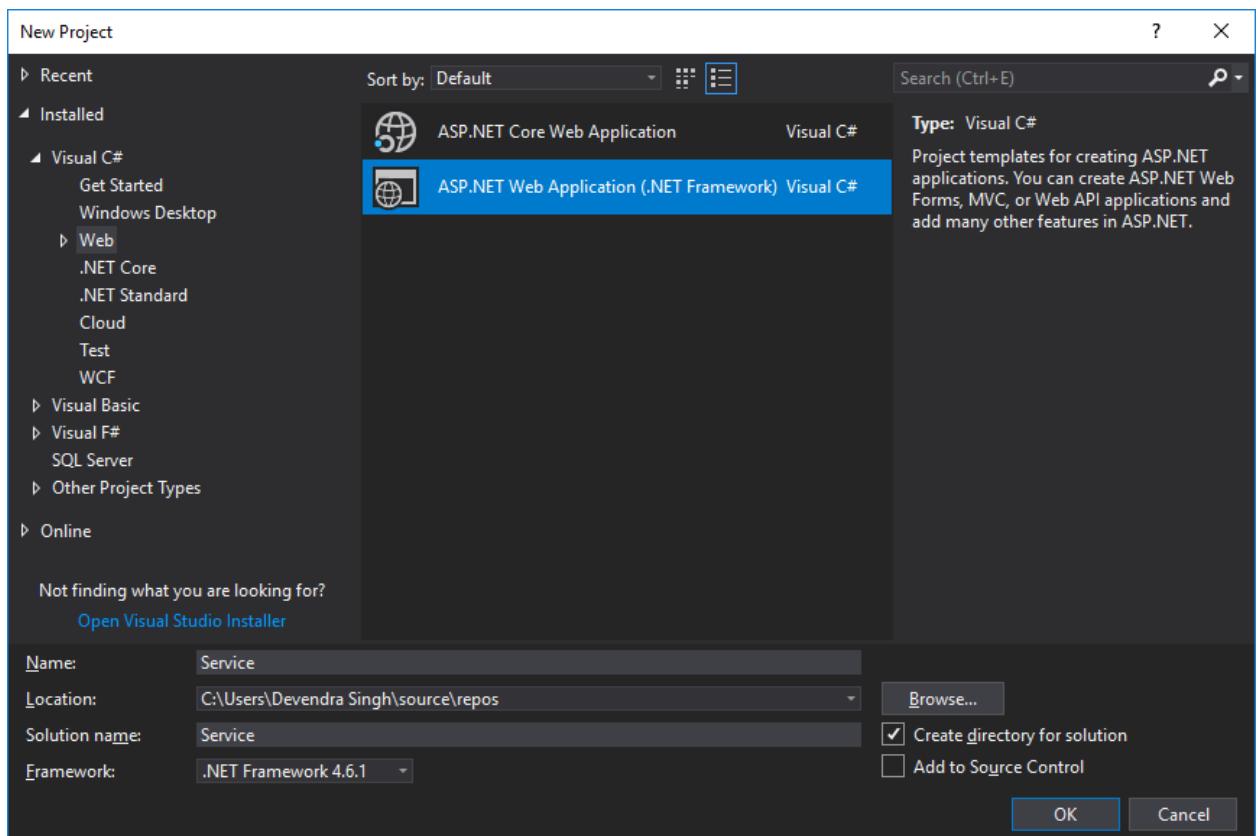
**File -> New -> Project**



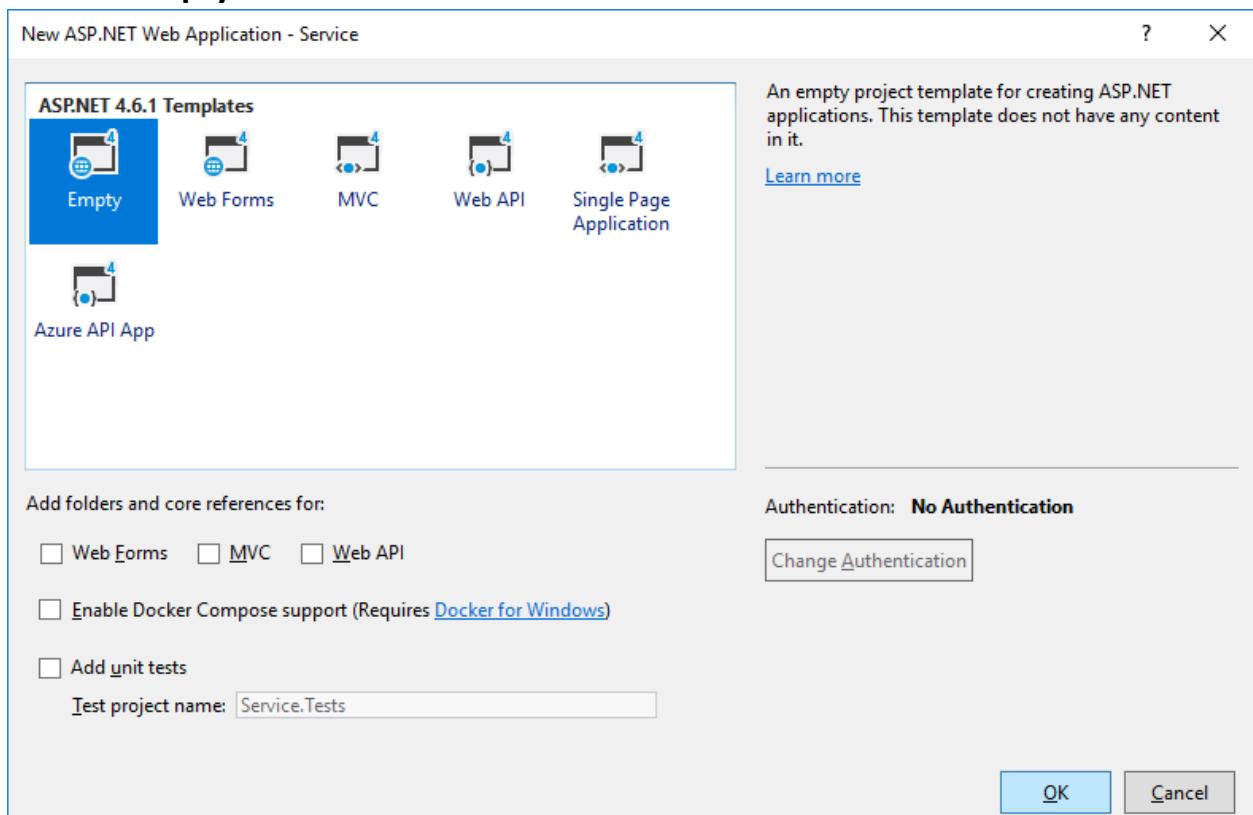
**2. Click on Web.**



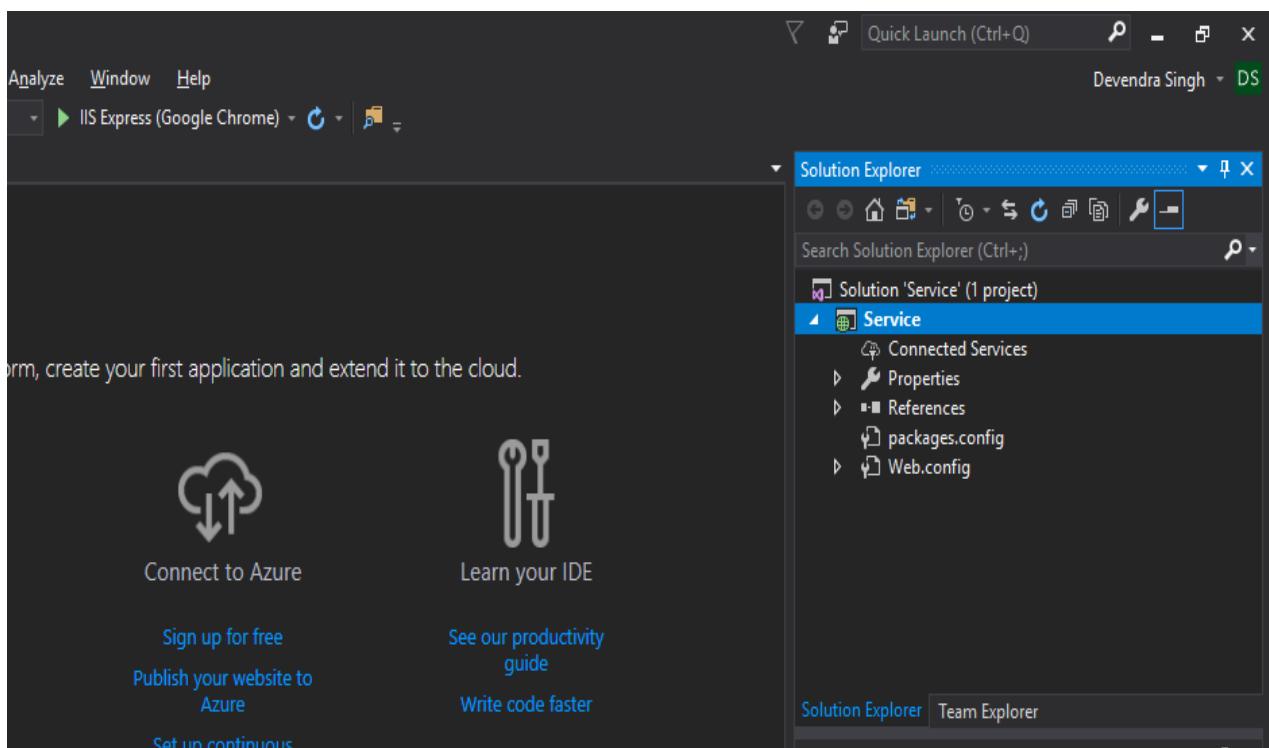
### 3. Select ASP.NET Web Application and give Name as Service. After that click on OK button.



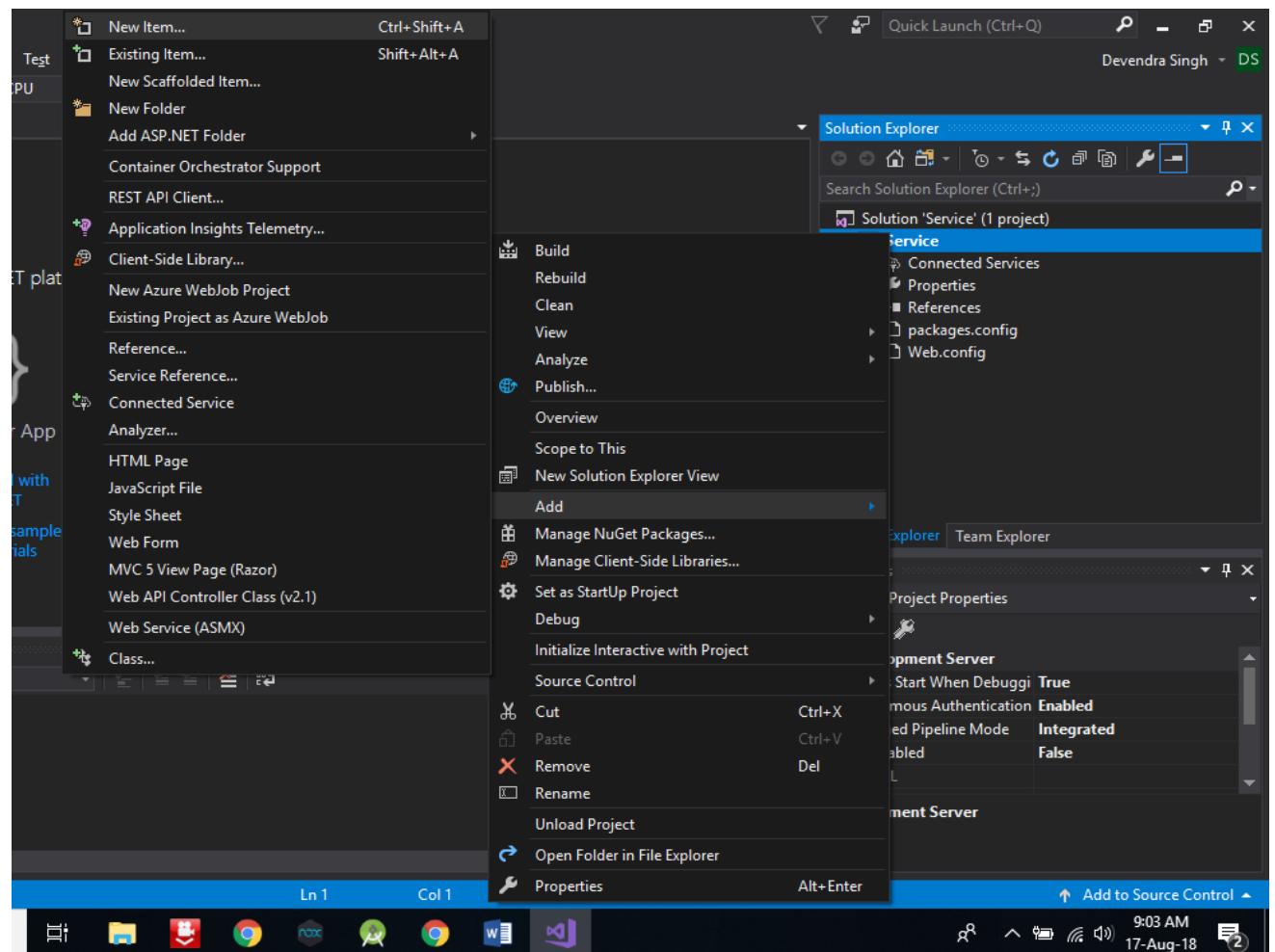
**4. Select Empty and click on OK button.**



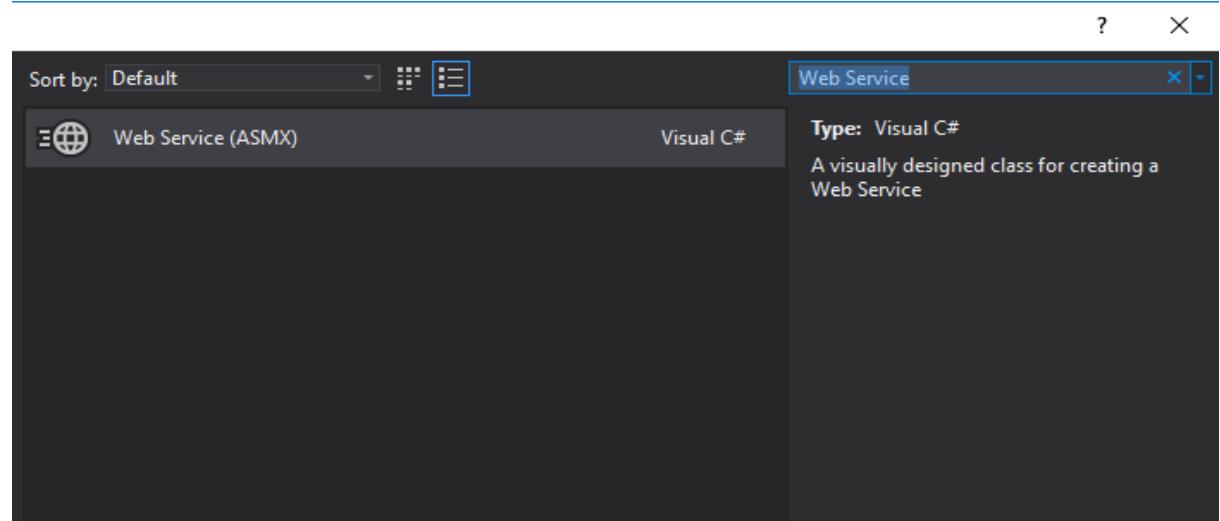
**5. Now you can see, on right side in Solution Explorer Service project is created.**



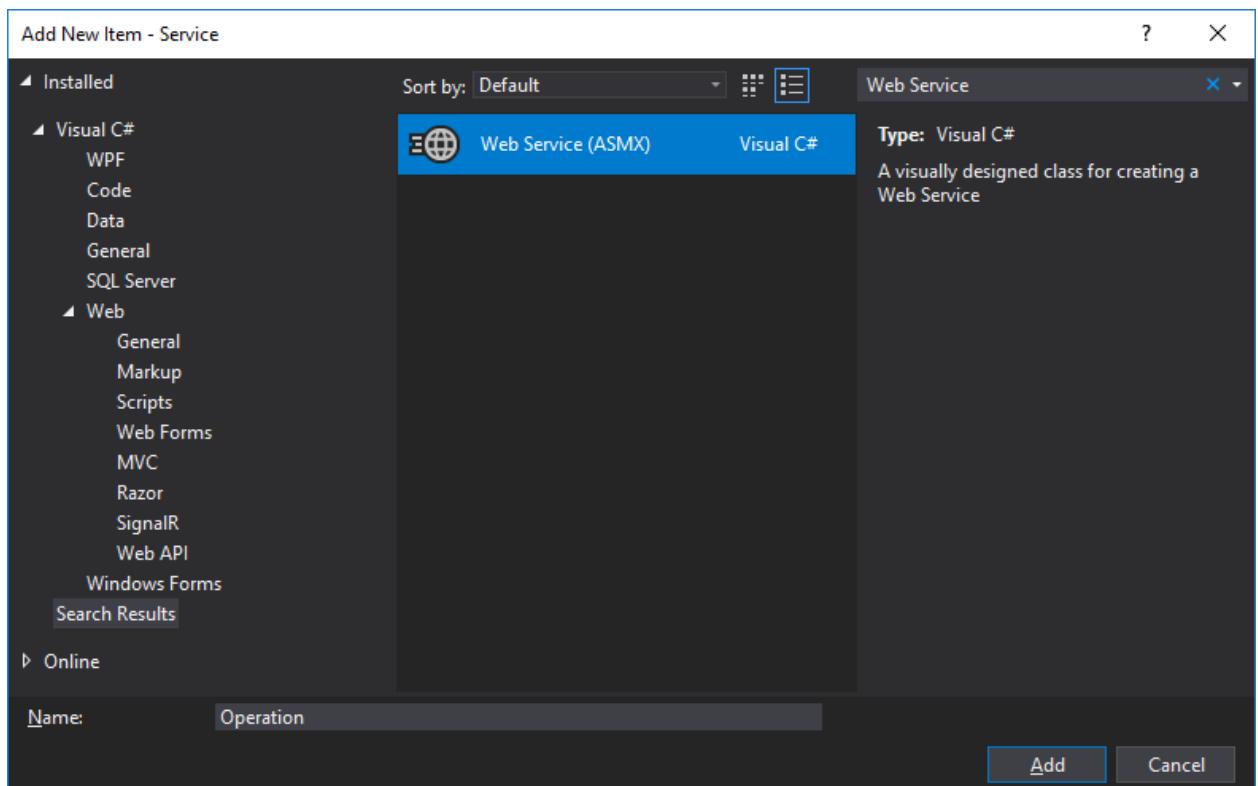
## 6. Right click on Service -> Add -> New Item.



## 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.****

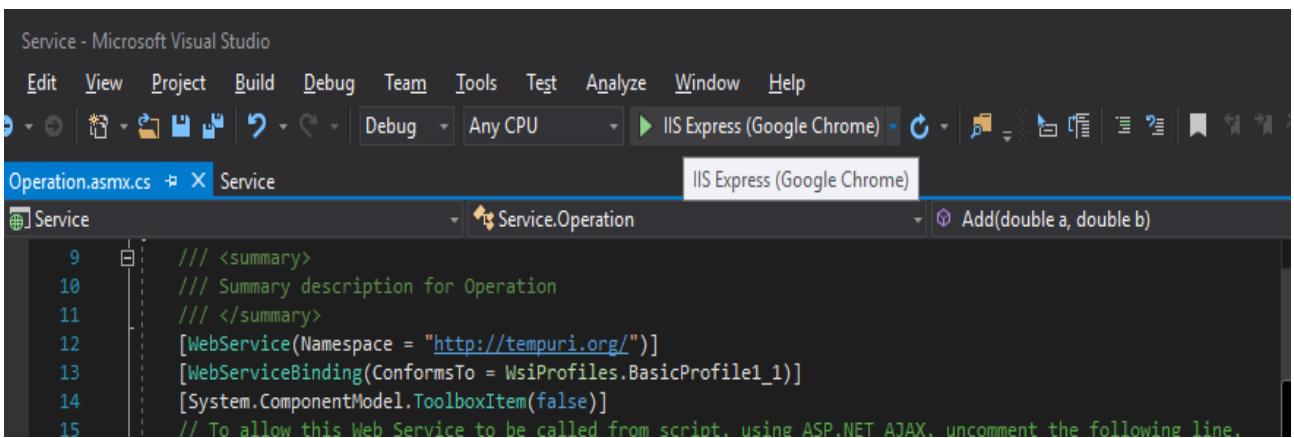
```
[WebMethod]
public double Add(double a, double b)
{
 double sum = a + b;
 return sum;
}
```

```
[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.**



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

The screenshot shows a browser window titled "Operation Web Service". The address bar displays "localhost:59649/Operation.asmx". Below the address bar, there is a message: "For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...". The main content area has a dark blue header with the word "Operation". Below the header, a message states: "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.**

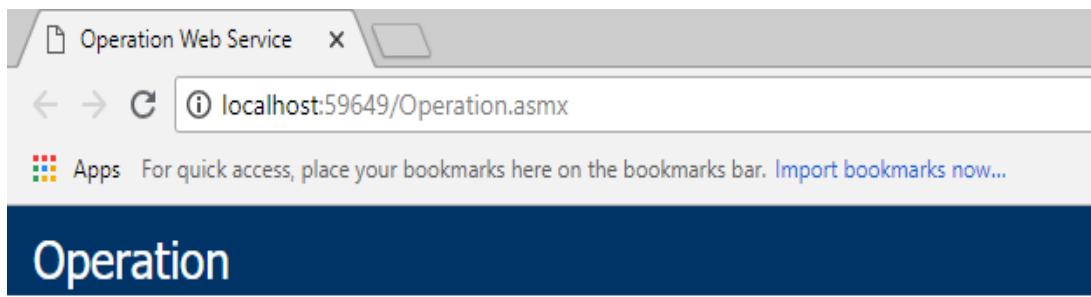
Each XML Web service needs a unique namespace in order for client applications to distinguish it from other Web services. You should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's URL as the namespace. (Namespaces need not point to actual resources on the Web. (XML Web service namespaces are URIs.))

For XML Web services created using ASP.NET, the default namespace can be changed using the `WebService` class's `Namespace` property. Below is a code example that sets the namespace to "http://microsoft.com/webservices/"

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

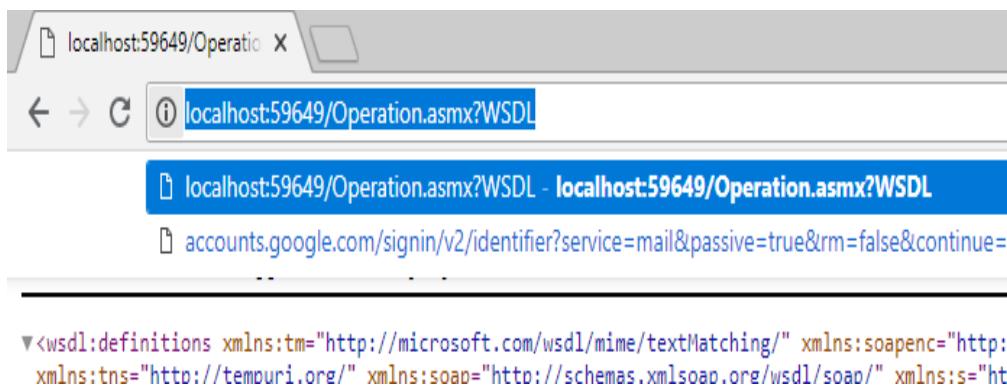
- [Add](#)
- [Multi](#)

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. It is recommended that you 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.**

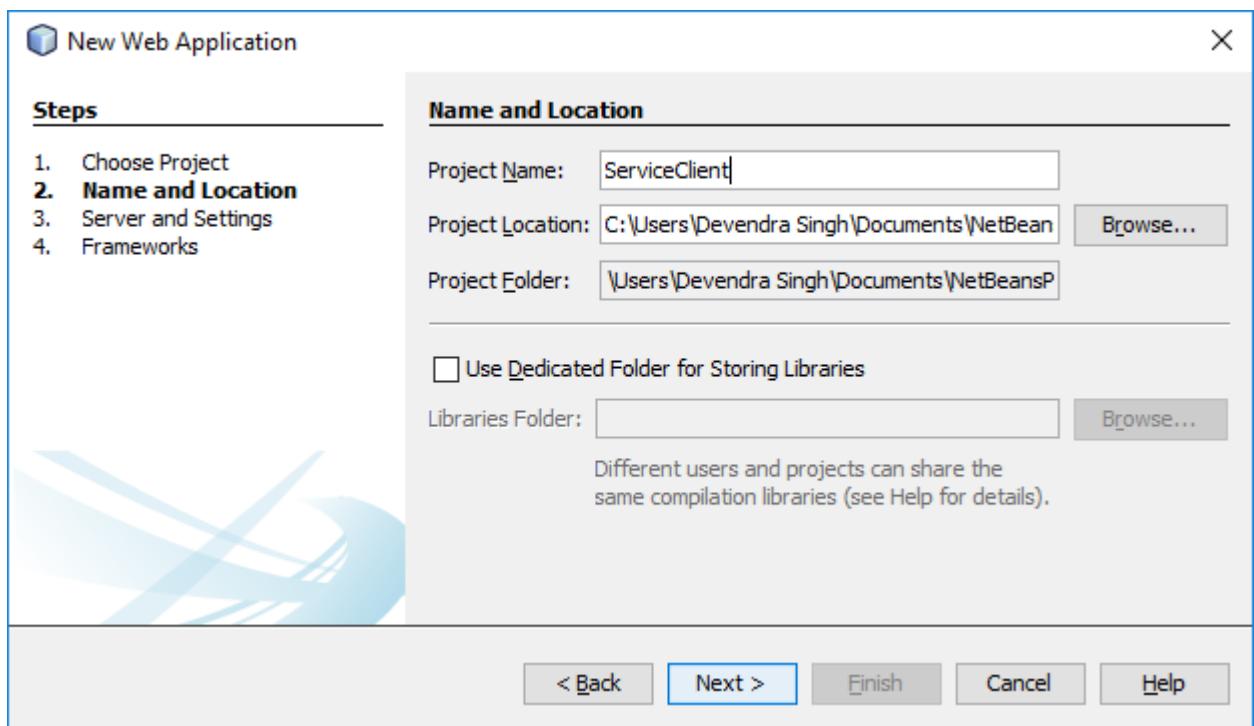


```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="http://tempuri.org/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://tempuri.org/Operation" xmlns="http://schemas.xmlsoap.org/wsdl/"></wsdl:definitions>
```

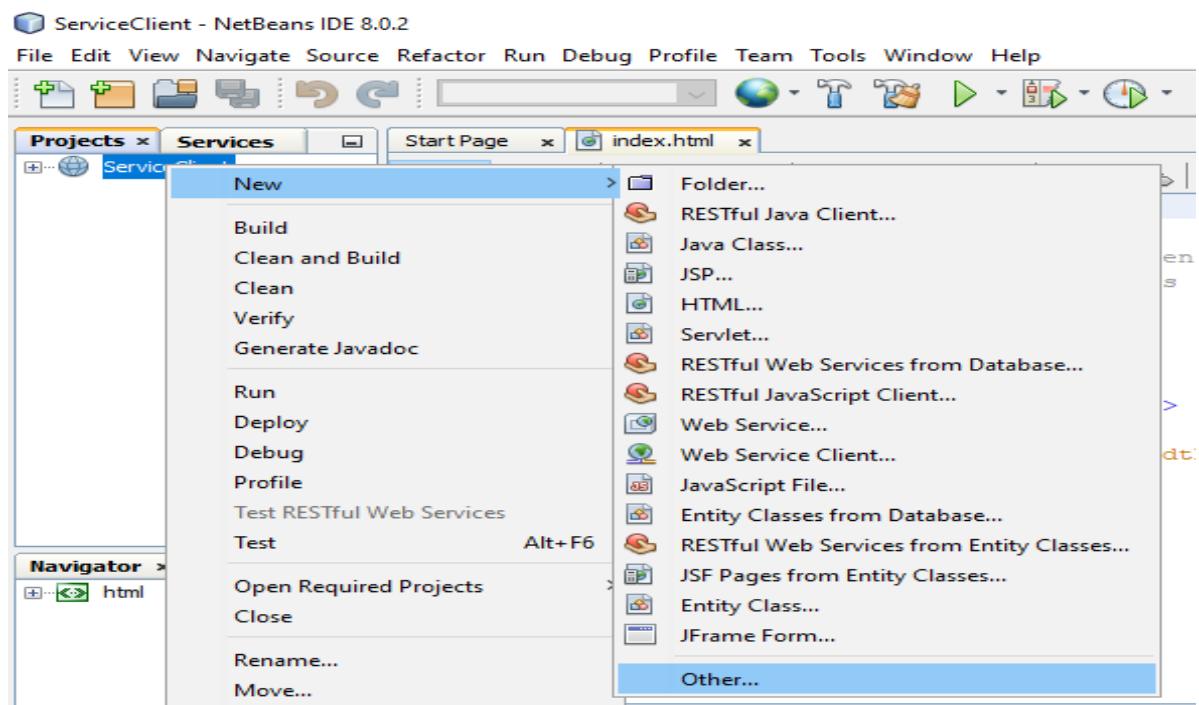
**15. Don't close Visual Studio and browser, just minimize it otherwise server will stop. But save the link anywhere, so that we can use it later.**

**16. Now open NetBeans.**

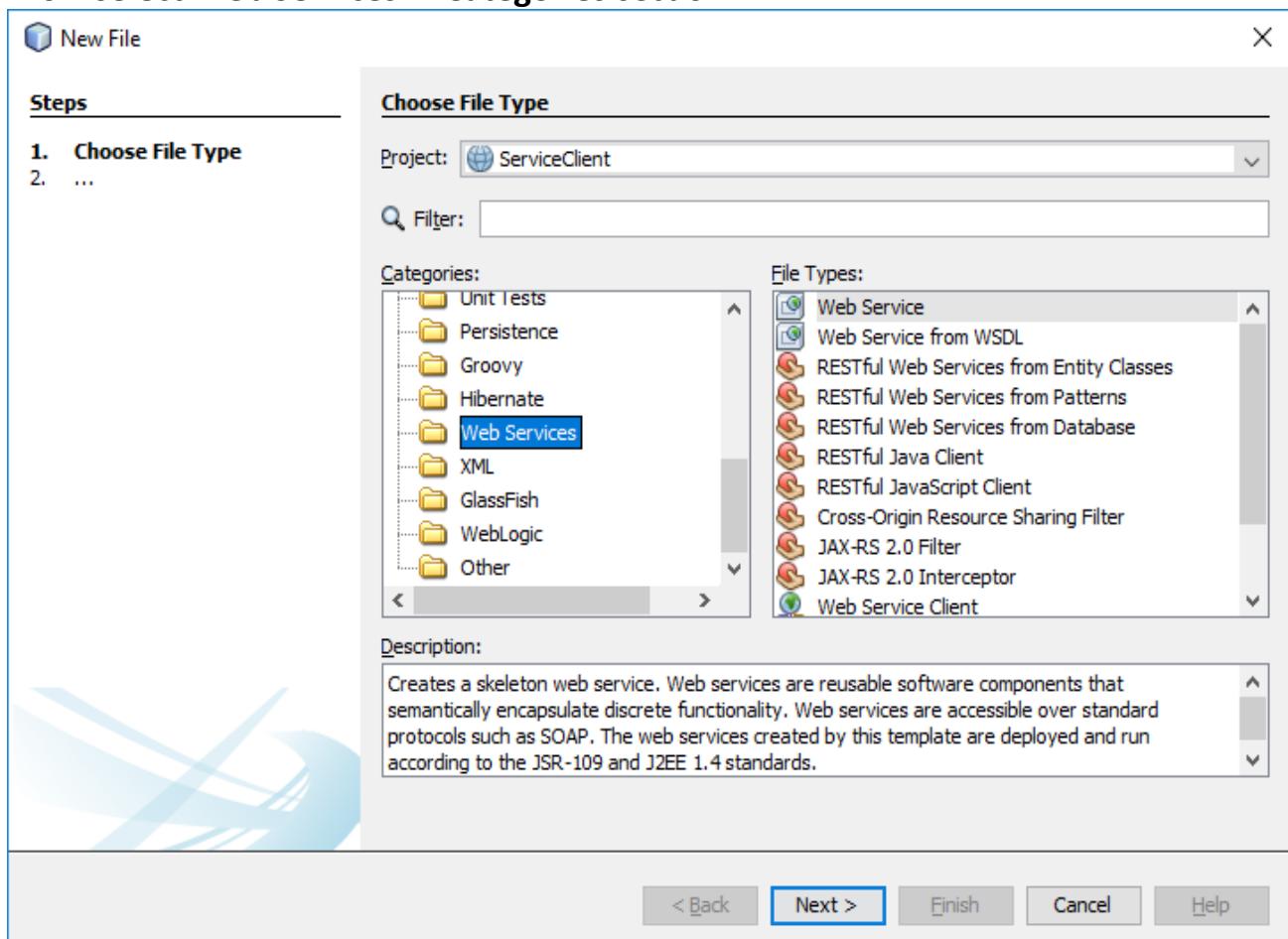
**17. Create a Web Application with name ServiceClient. Next -> Finish.**



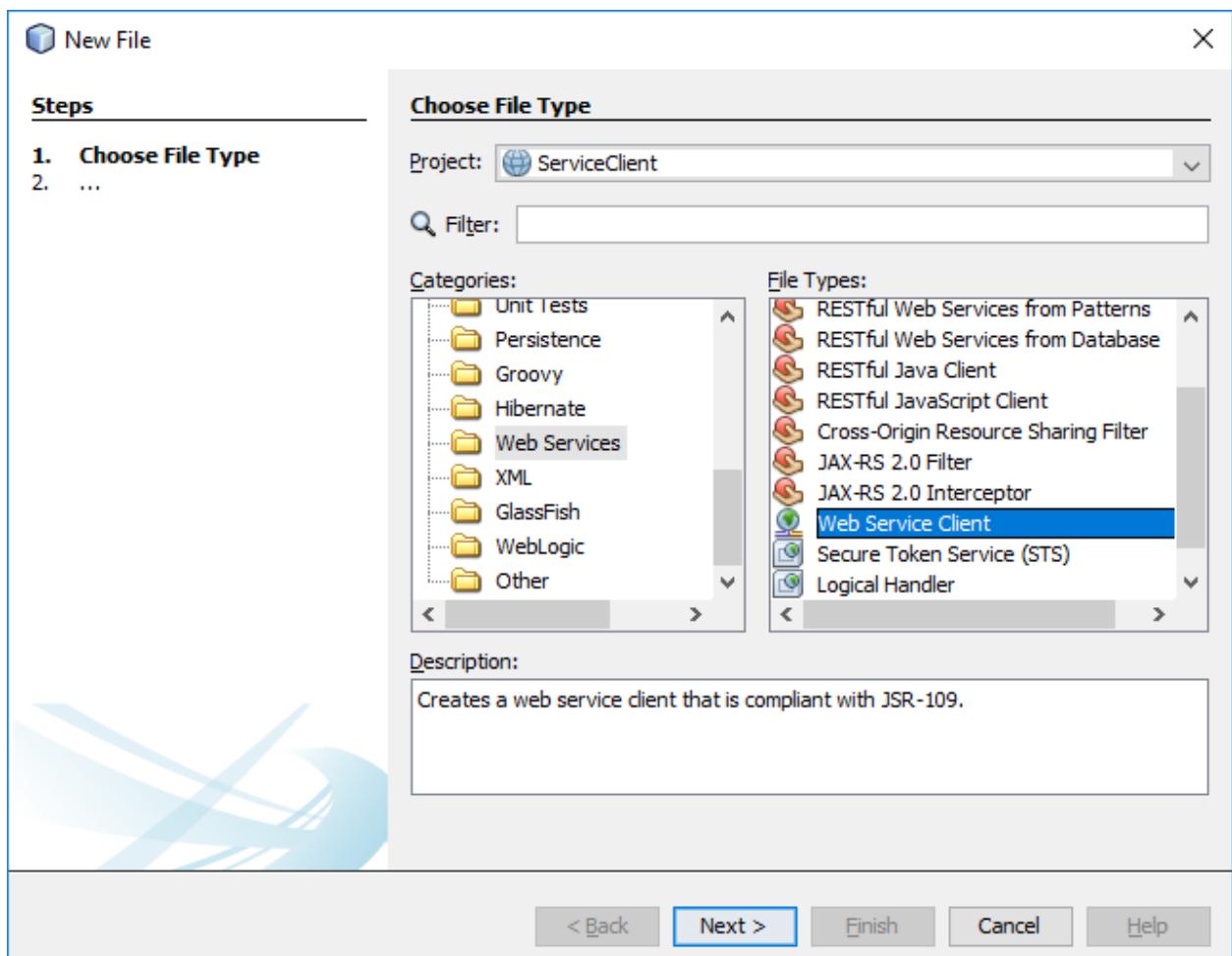
**18. Now create a Web Service Client.**  
Right click on ServiceClient -> New -> Other.



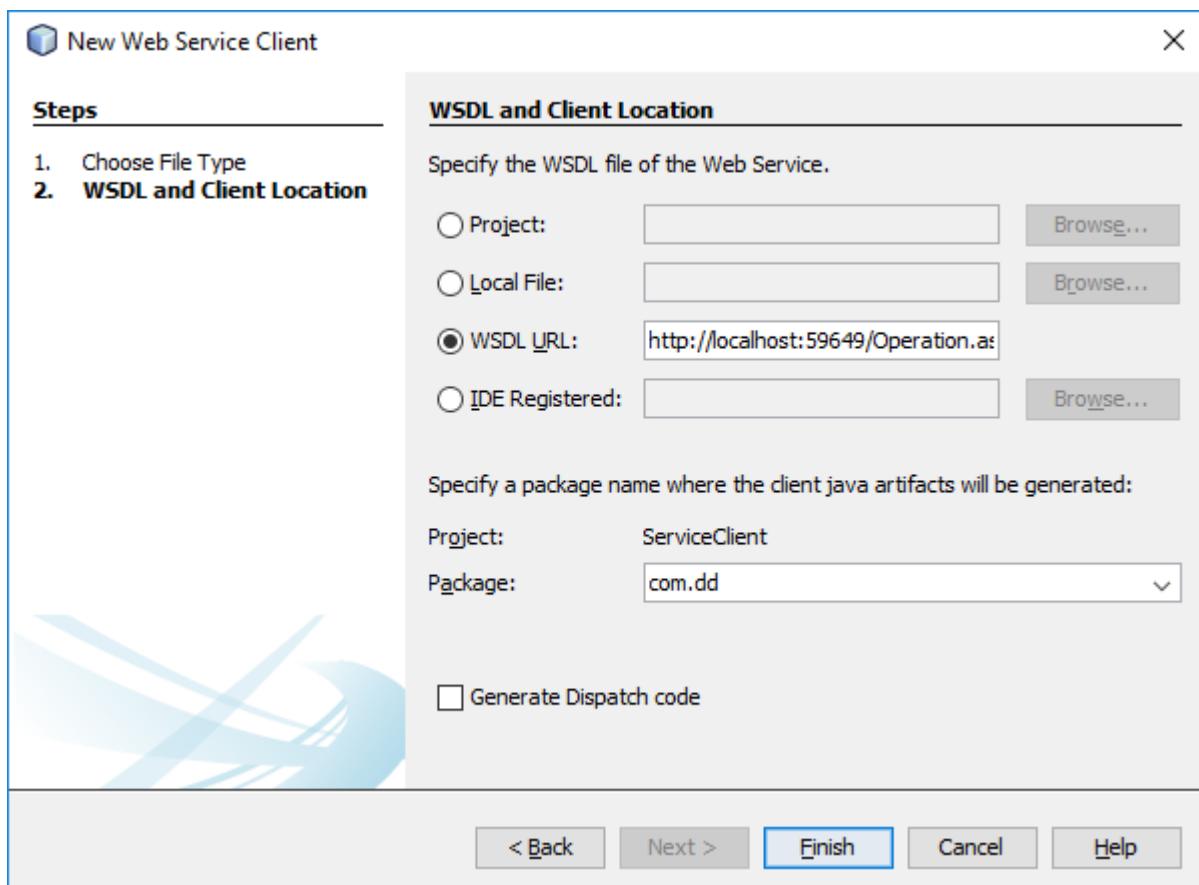
**19. Now select Web Services in Categories section.**



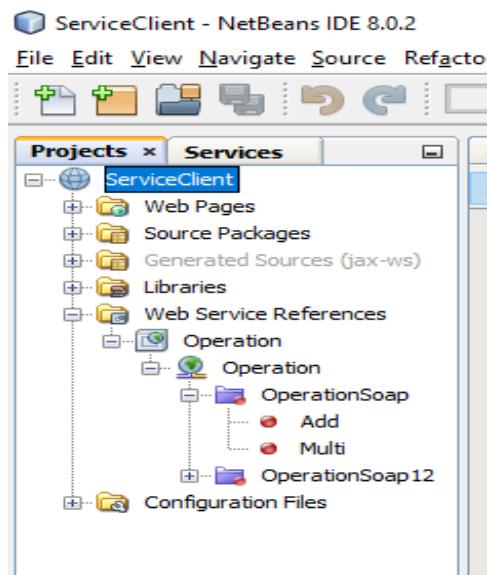
**20. Select Web Service Client in File Types and Click on Next button.**



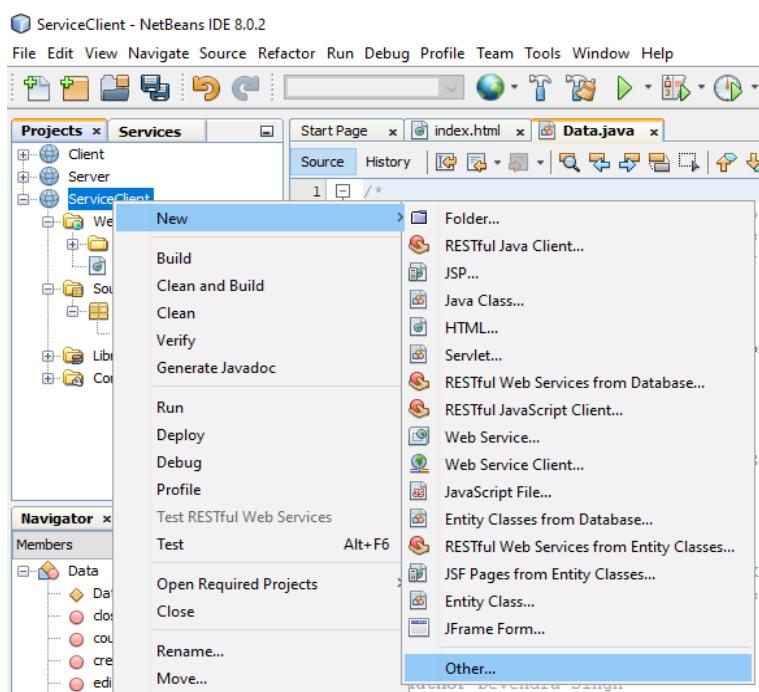
**21. Select WSDL URL and paste the link that you have copied from browser on run of Visual Studio enter package name com.dd. After that click on Finish button.**



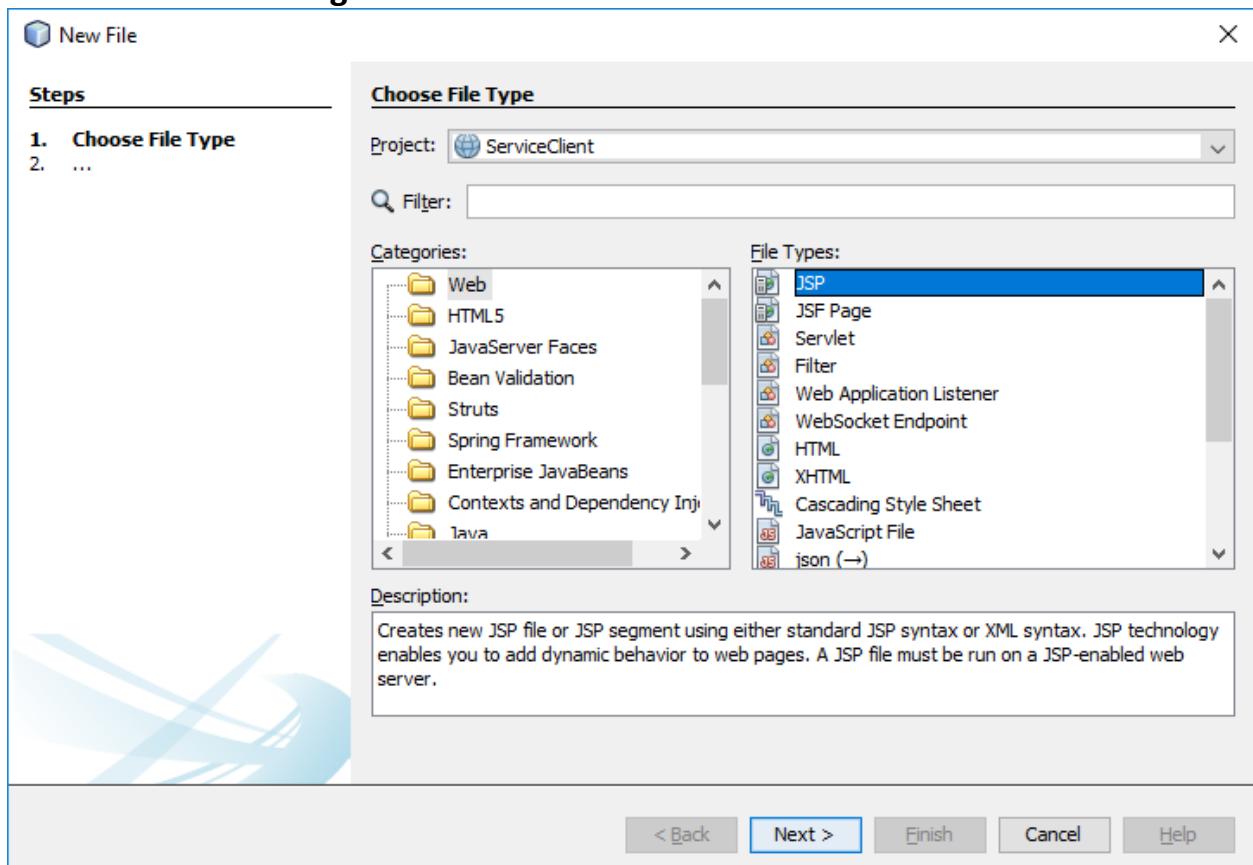
**22.** As you can see, we got both the service methods i.e. Add & Multi. But in this practical I'm going to use only one service method. You can do for both also.



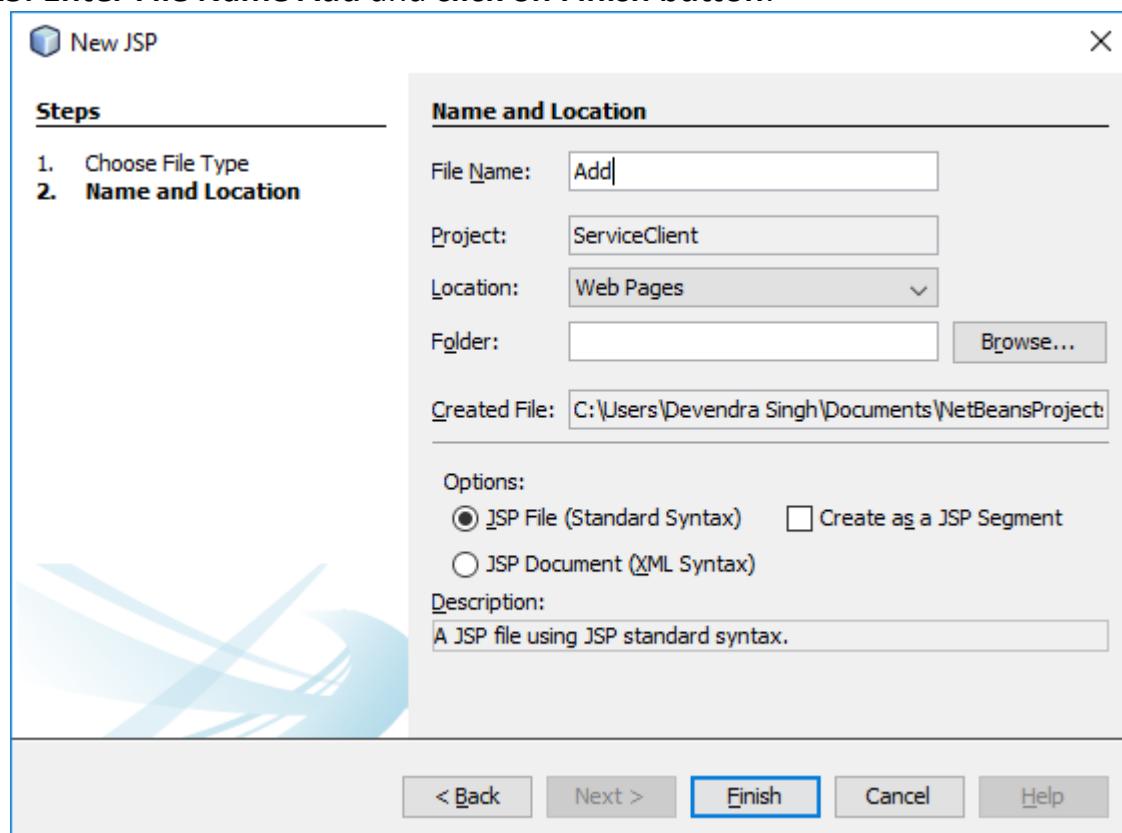
**23.** Now create a JSP page. Right click on ServiceClient -> New -> Other



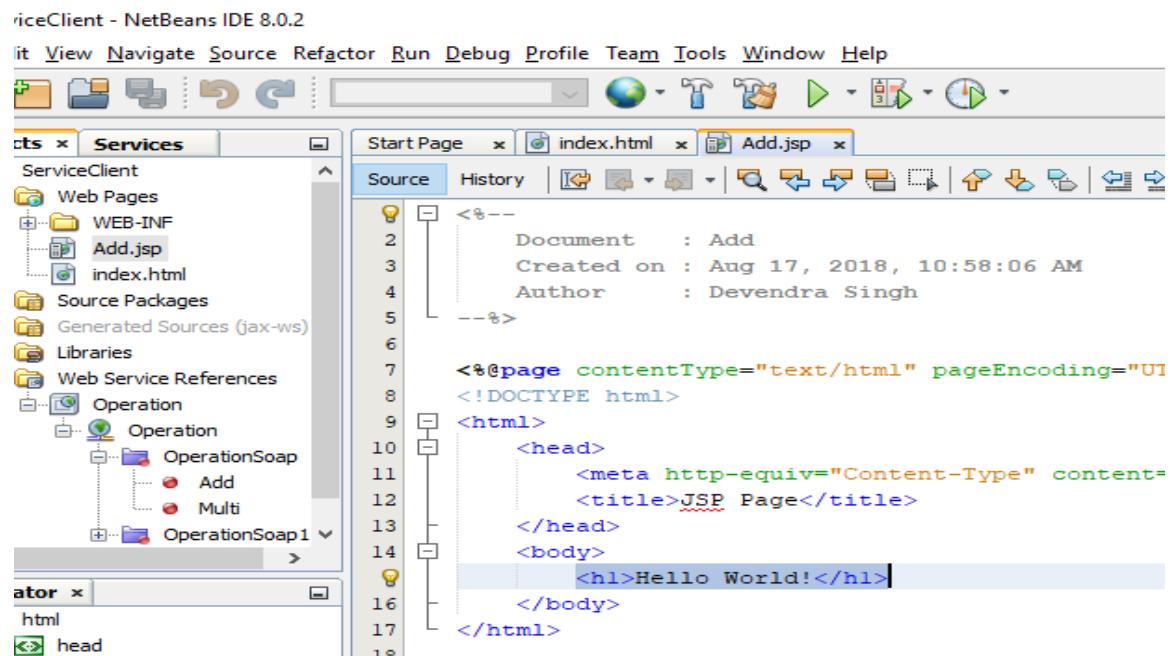
#### 24. Select Web in Categories section -> Select JSP and click on Next button.



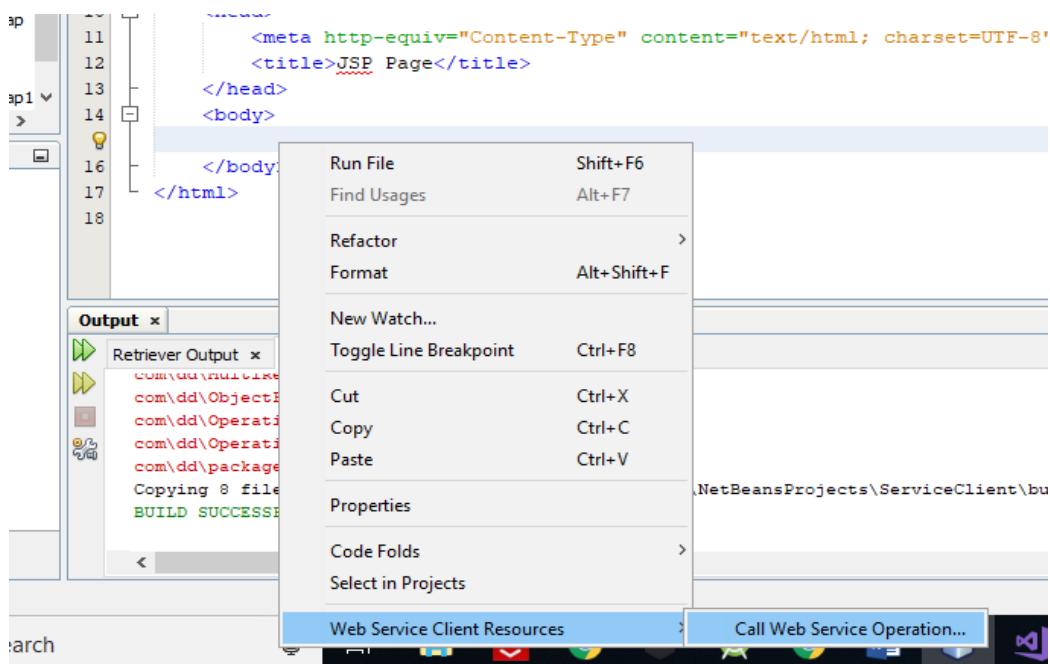
**25. Enter File Name Add and click on Finish button.**



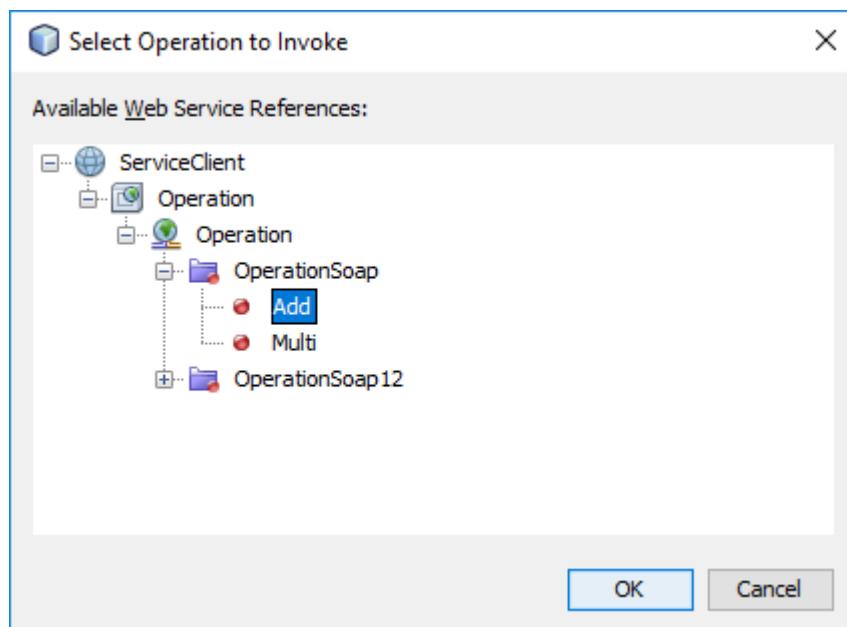
**26. In Add.jsp file delete the selected part in body tag, because we don't need this.**



**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 the file 'Add.jsp' open. The code is a JSP page that starts with a comment for web service invocation. It then declares two double variables, num1 and num2, by parsing parameters from the request. A try block is used to create an Operation service and get its OperationSoap port. Inside the try block, variables a and b are initialized to 0.0d. A comment indicates where to initialize WS operation arguments. The code then adds a and b using the port's add method and prints the result. A catch block handles exceptions. Finally, it ends the web service invocation with a comment.

```
<%-- start web service invocation --%><hr/>
<%
 double num1 = Double.parseDouble(request.getParameter("txt1"));
 double num2 = Double.parseDouble(request.getParameter("txt2"));
try {
 com.dd.Operation service = new com.dd.Operation();
 com.dd.OperationSoap port = service.getOperationSoap();
 // TODO initialize WS operation arguments here
 double a = 0.0d;
 double b = 0.0d;
 // TODO process result here
 double result = port.add(a, b);
 out.println("Result = "+result);
} catch (Exception ex) {
 // TODO handle custom exceptions here
}
%>
<%-- 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 with the 'Add.jsp' file open. The code has been modified. The variable declarations for num1 and num2 have been moved inside the try block, and they are now assigned the values of a and b respectively. The rest of the code remains the same, including the comments and the catch block.

```
<%-- start web service invocation --%><hr/>
<%
try {
 double num1 = Double.parseDouble(request.getParameter("txt1"));
 double num2 = Double.parseDouble(request.getParameter("txt2"));
 com.dd.Operation service = new com.dd.Operation();
 com.dd.OperationSoap port = service.getOperationSoap();
 // TODO initialize WS operation arguments here
 double a = num1;
 double b = num2;
 // TODO process result here
 double result = port.add(a, b);
 out.println("Result = "+result);
} catch (Exception ex) {
 // TODO handle custom exceptions here
}
%>
<%-- 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">

 <input type="text" name="txt2" placeholder="Enter Second
Number">

 <input type="submit" formaction="Add.jsp" value="Add Numbers">
</form>

```

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

- File Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard icons for file operations like Open, Save, Find, etc.
- Project Explorer (Projects x Services):**
  - ServiceClient project selected.
  - Web Pages folder contains WEB-INF and index.html.
  - WEB-INF contains Add.jsp.
  - index.html is currently selected.
  - Source Packages, Generated Sources (jax-ws), Libraries, and Web Service References are also listed.
- Navigator (Navigator x):**
  - html node expanded.
  - head node under html.
  - body node under html.
  - form node under body.
  - input node under form.
  - br node under form.
- Source Editor (Start Page x | index.html x | Add.jsp x):**
  - Source tab selected.
  - Code content:

```

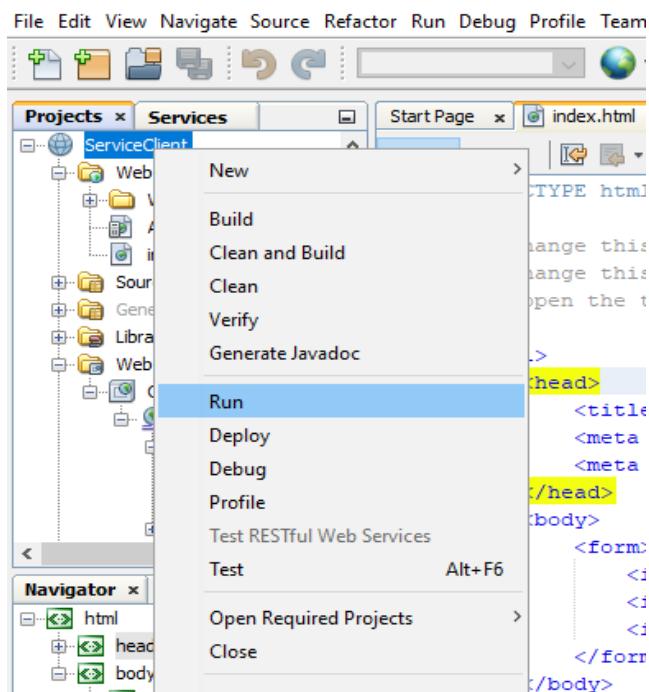
1 <!DOCTYPE html>
2 <!--
3 To change this license header, choose License Headers in Project Properties.
4 To change this template file, choose Tools | Templates
5 and open the template in the editor.
6 -->
7 <html>
8 <head>
9 <title>TODO supply a title</title>
10 <meta charset="UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 </head>
13 <body>
14 <form>
15 <input type="text" name="txt1" placeholder="Enter First Number">

16 <input type="text" name="txt2" placeholder="Enter Second Number">

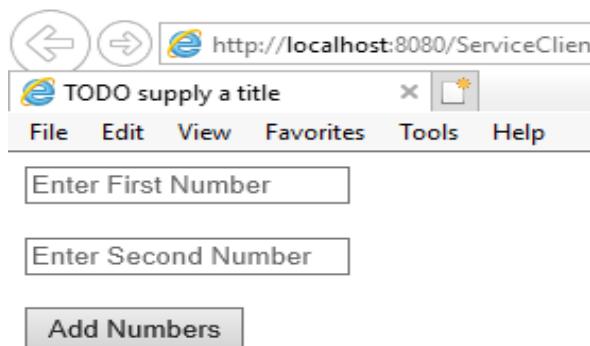
17 <input type="submit" formaction="Add.jsp" value="Add Numbers">
18 </form>
19 </body>
20 </html>

```
- Output (Output x):** Not visible in the screenshot.

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.

http://localhost:8080/ServiceClient/

TODO supply a title

File Edit View Favorites Tools Help

19

19

Add Numbers

http://localhost:8080/ServiceClient/Add

JSP Page

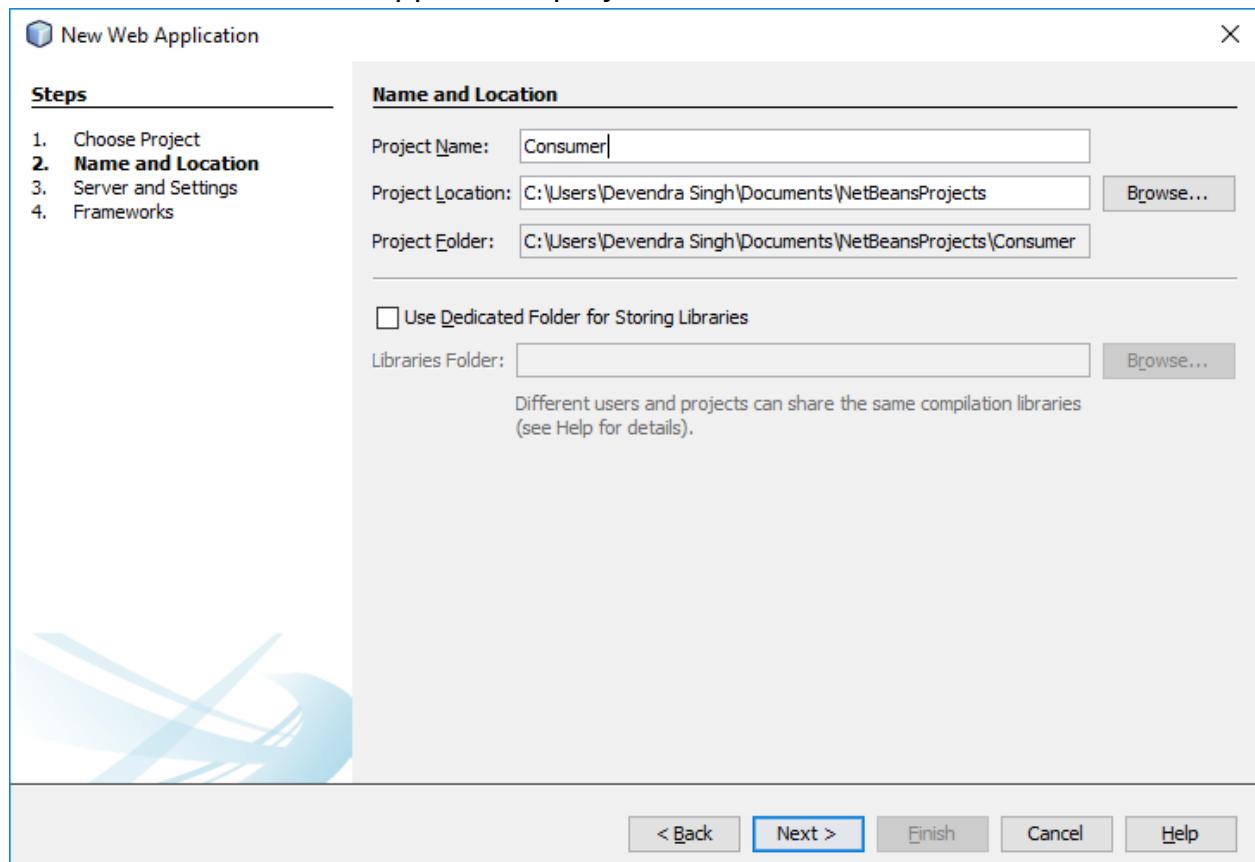
File Edit View Favorites Tools Help

Result = 38.0

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

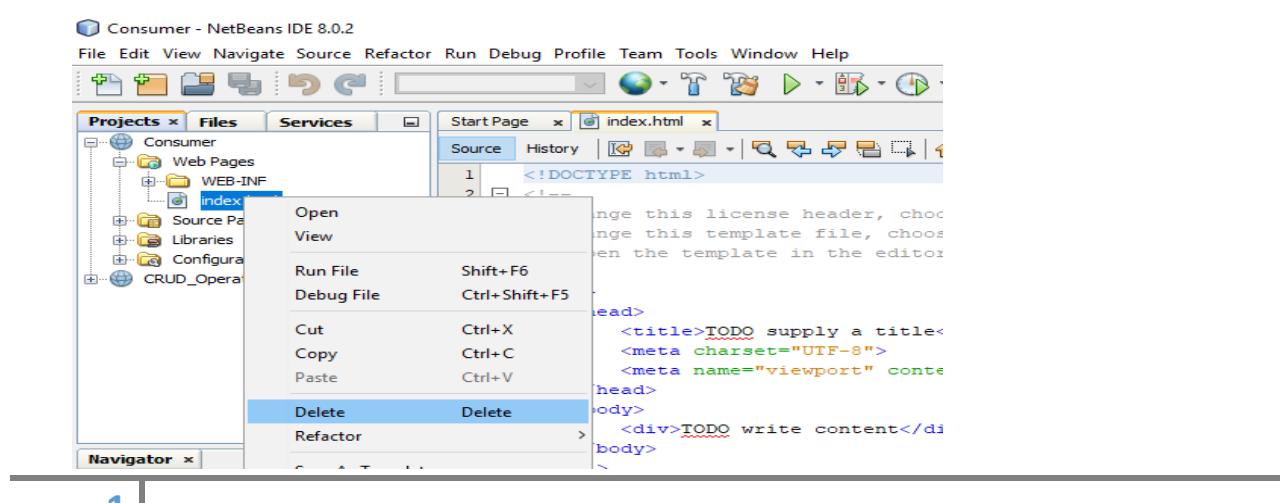
After that do not close or restart the NetBeans.

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

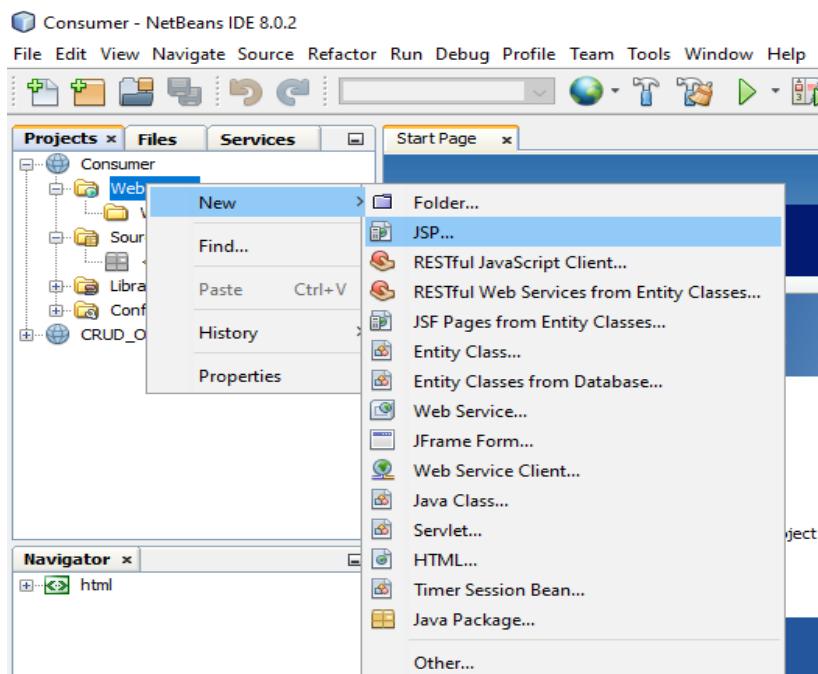


**2.** And create it. **Next -> Finish**.

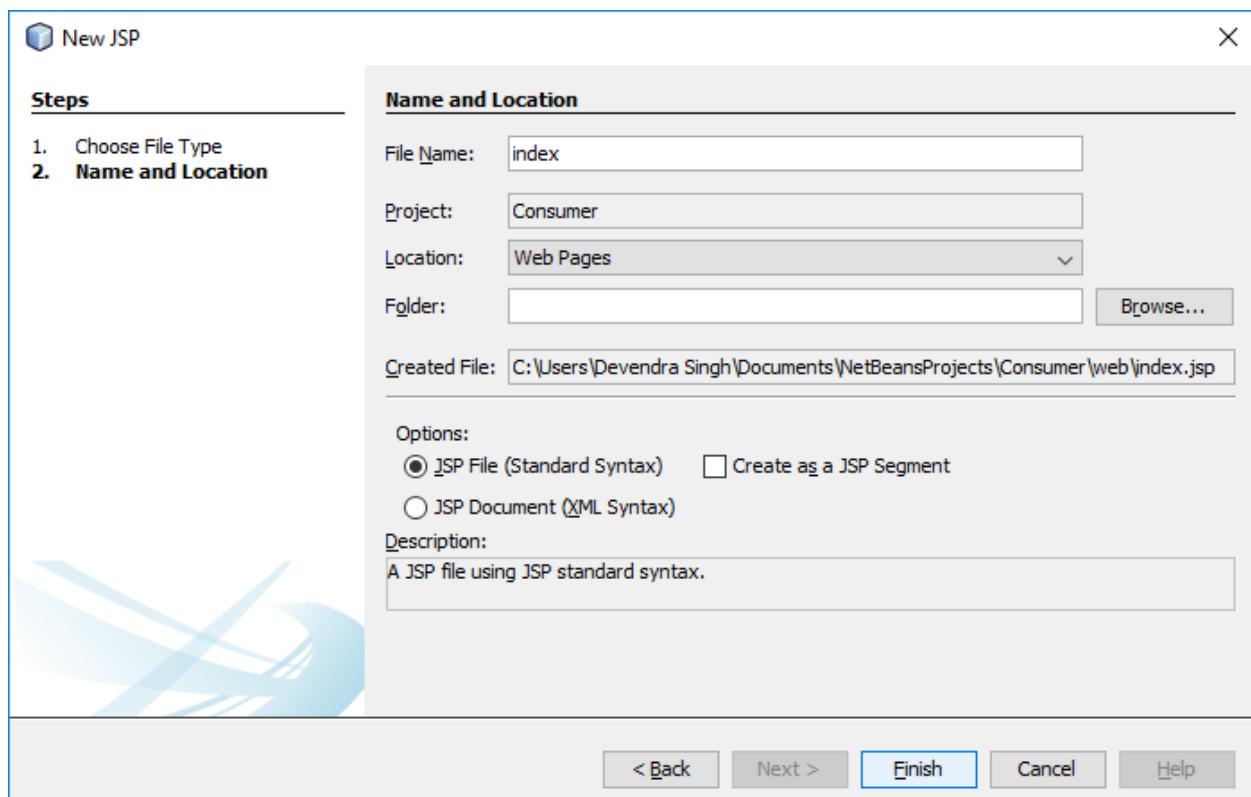
**3.** Now right click on index.html and delete it.



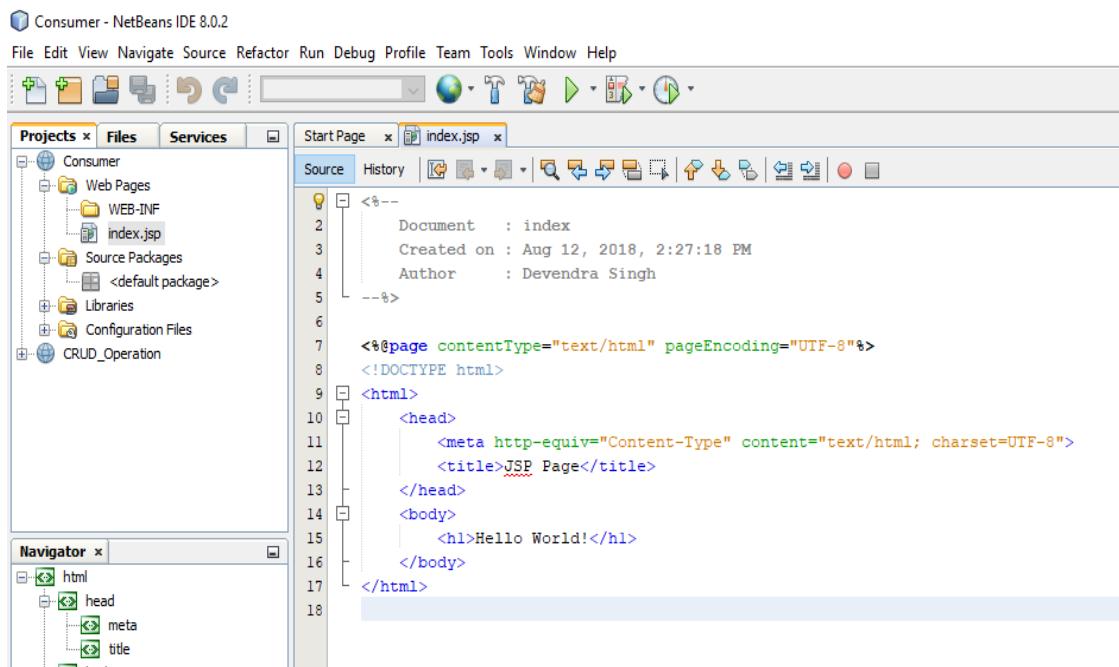
4. Now right click on Web Pages and select JSP to add a JSP page.



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>

```

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

The screenshot shows a code editor interface with multiple tabs at the top: Start Page, index.html, seller.java, and index.jsp. The index.jsp tab is active. Below the tabs is a toolbar with various icons. The main area contains Java script code. A tooltip with a question mark icon is positioned over the URL in line 29, which is highlighted in blue.

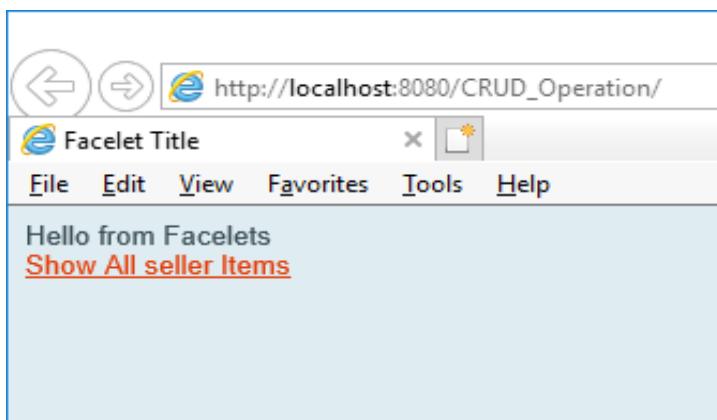
```

9 <HTML>
10 <HEAD>
11 <script language="javascript">
12 var xmlhttp;
13 function init()
14 {
15 xmlhttp=new XMLHttpRequest();
16 }
17 function readLocal(){
18 if(window.localStorage){
19 var seller=localStorage.getItem("seller");
20 seller=JSON.parse(seller);
21 document.getElementById("firstName").value=seller.firstName;
22 document.getElementById("sellerid").value=seller.id;
23 }
24 }
25
26 function saveLocal()
27 {
28 var sellerid=document.getElementById("sellerid");
29 var url="http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/"+sellerid.value;
30 xmlhttp.open('GET',url,true);
31 xmlhttp.send(null);
32 xmlhttp.onreadystatechange =function() {
33

```

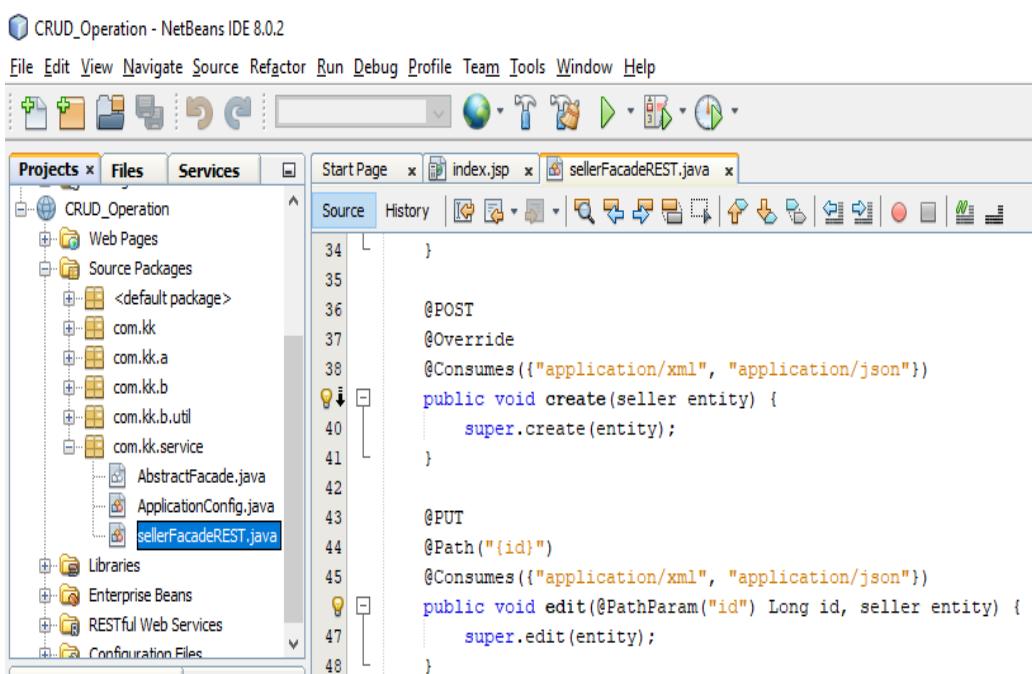
## 9. [http://localhost:8080/CRUD\\_Operation/webresources/com.kk.seller/](http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/)

Red part is the URL of which is obtained in browser by running of CRUD\_Operation web application.



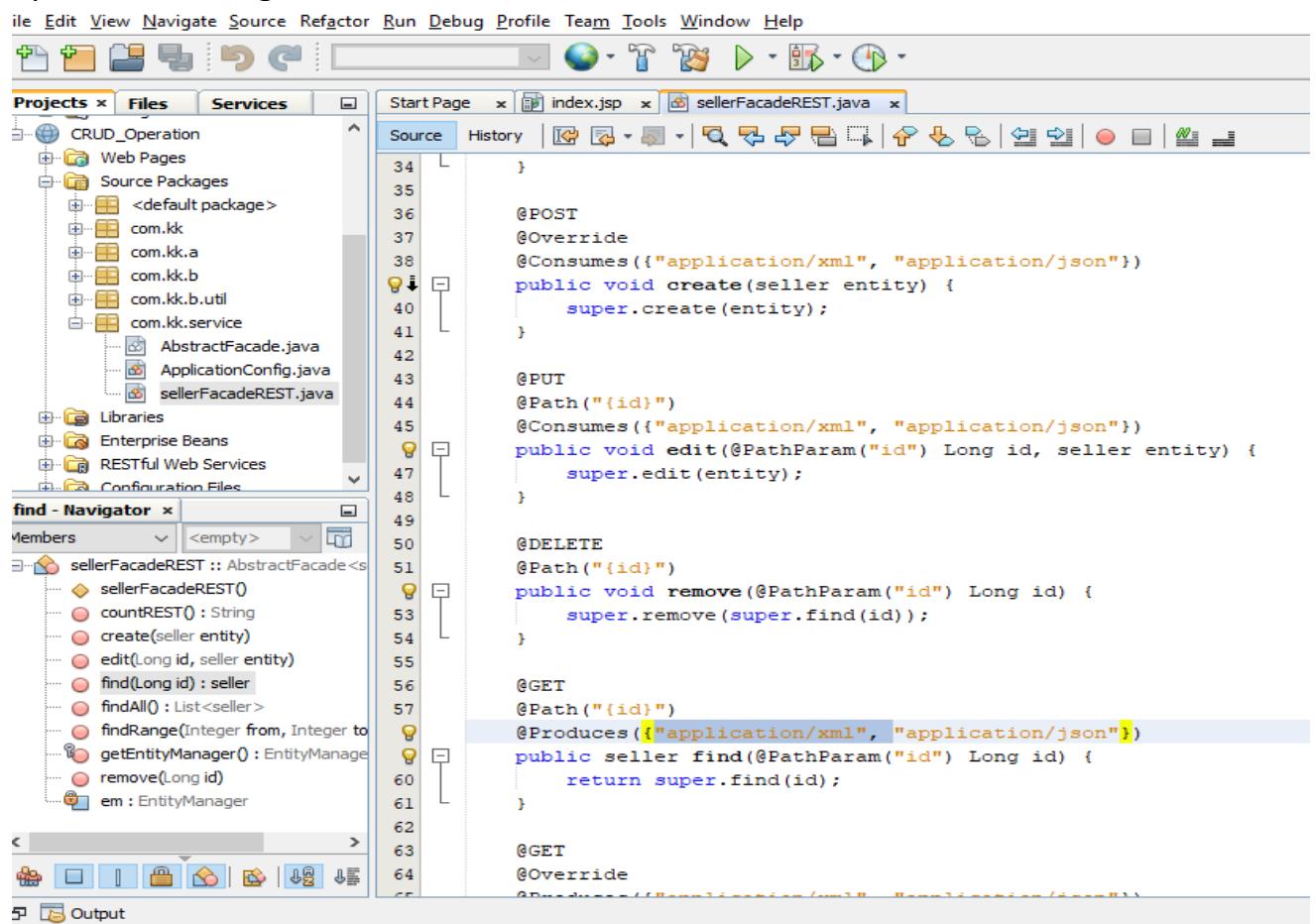
Blue part in above link is **static**. But red part is **dynamic** which is based on practical 7. So if you have changed name anywhere then the above URL will change accordingly.

## 10. Now open sellerFacadeREST.java file by follow below pic available in CRUD\_Operation web application.



```
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 }
```

**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.



The screenshot shows the JBoss Seam IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The left sidebar displays the project structure under 'CRUD\_Operation' with 'Web Pages', 'Source Packages', 'Libraries', 'Enterprise Beans', 'RESTful Web Services', and 'Configuration Files'. The main editor window shows the Java code for **sellerFacadeREST.java**. A portion of the code is highlighted in yellow, indicating it is selected:

```
 }
 @POST
 @Override
 @Consumes({"application/xml", "application/json"})
 public void create(seller entity) {
 super.create(entity);
 }

 @PUT
 @Path("{id}")
 @Consumes({"application/xml", "application/json"})
 public void edit(@PathParam("id") Long id, seller entity) {
 super.edit(entity);
 }

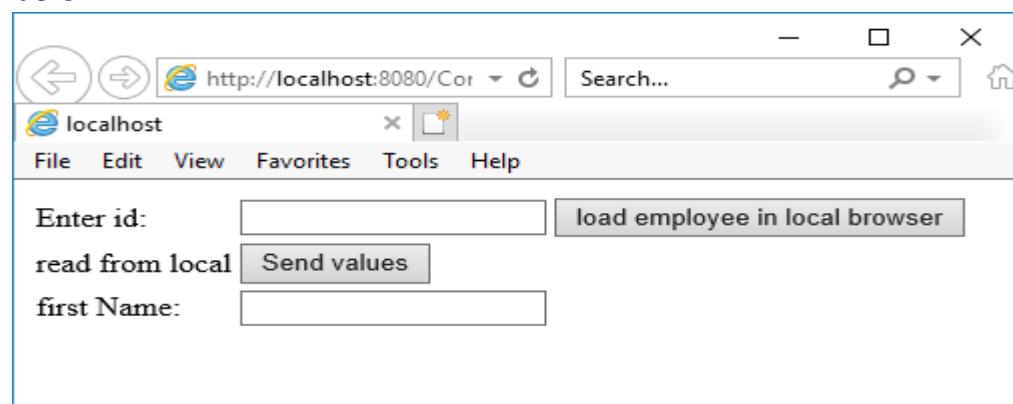
 @DELETE
 @Path("{id}")
 public void remove(@PathParam("id") Long id) {
 super.remove(super.find(id));
 }

 @GET
 @Path("{id}")
 @Produces({"application/xml", "application/json"})
 public seller find(@PathParam("id") Long id) {
 return super.find(id);
 }

 @GET
 @Override
 @Consumes("text/html")
 @Produces("text/html")
 public String index() {
 return "/index";
 }
}
```

The 'find - Navigator' panel on the left lists the methods and fields of the **sellerFacadeREST** class. The 'Output' panel at the bottom shows a single entry: 'INFO: JBoss Seam 3.0.0.Final - 20120621\_1422'.

**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.

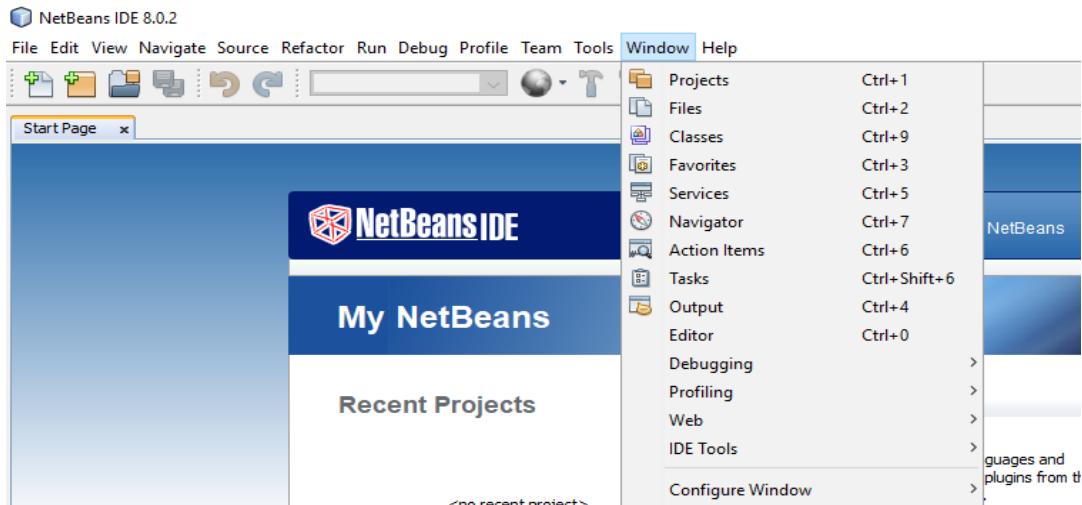
The screenshot shows a web browser window with the following details:

- Address bar: http://localhost:8080/Consumer/
- Toolbar: Back, Forward, Stop, Refresh, Home (localhost), File, Edit, View, Favorites, Tools, Help
- Form fields:
  - Enter id:
  - load employee in local browser
  - read from local
  - first Name:

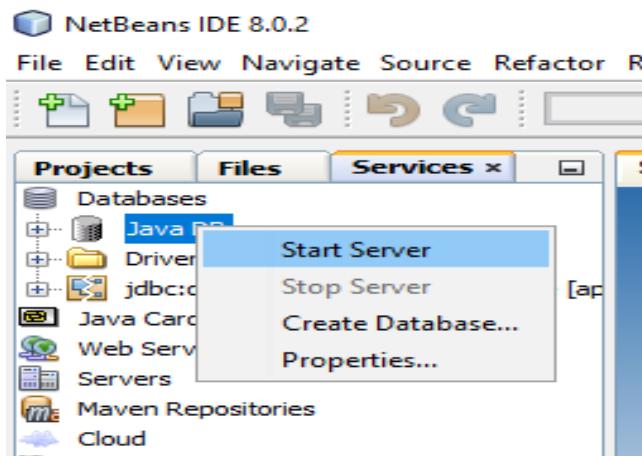
## PRACTICAL-6

**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.

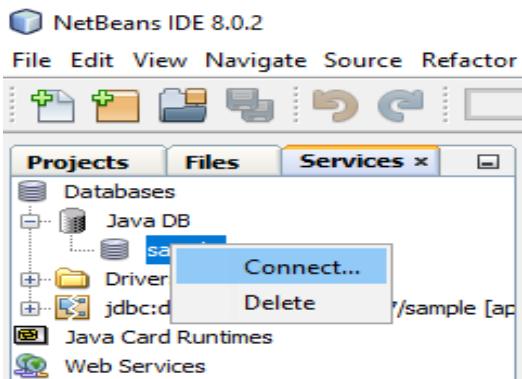
1. Click on Window menu and click on Projects, Files & Services to open it.



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

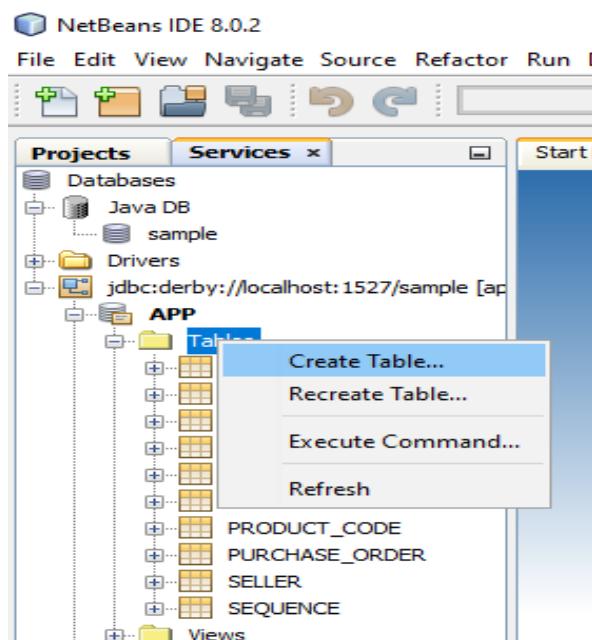


3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.

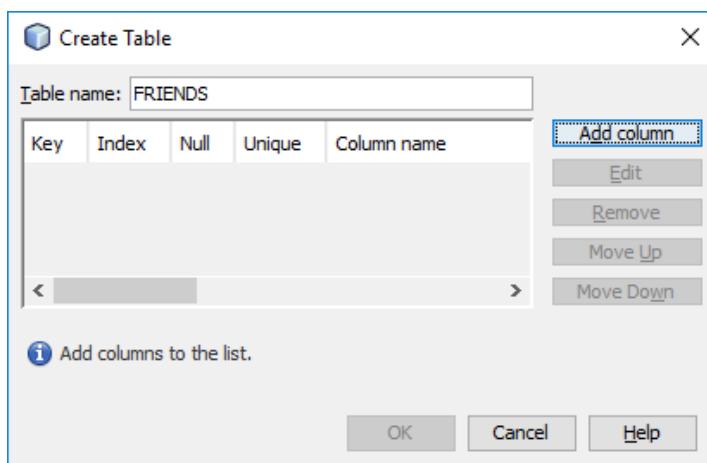


4. Now we are going to create a table in default database **sample**.

**Right click on Table -> Create Table**

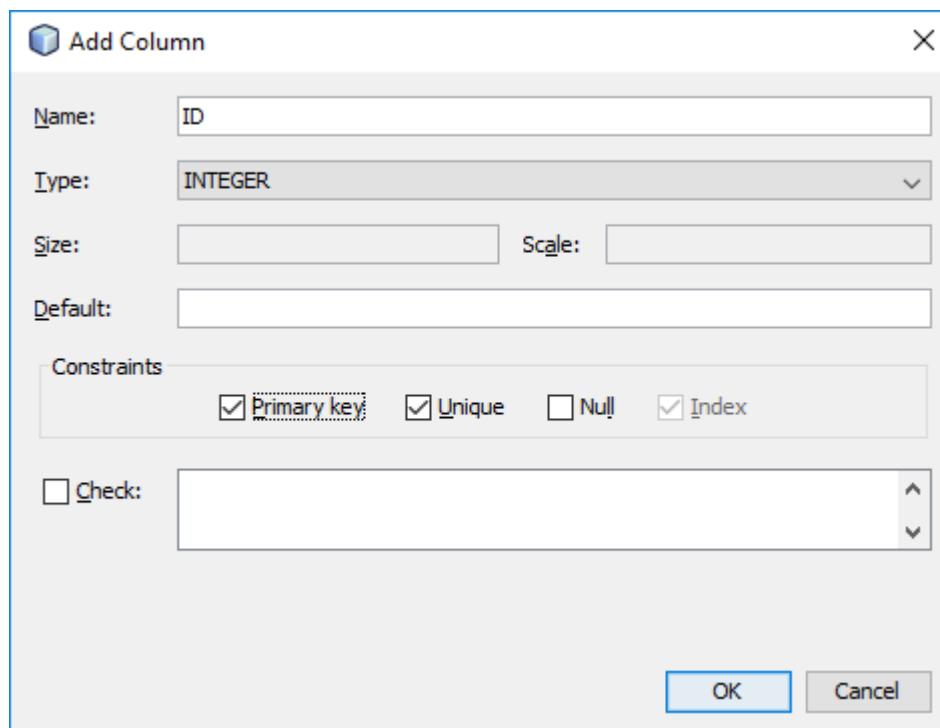


5. Give **table name as FRIENDS**.

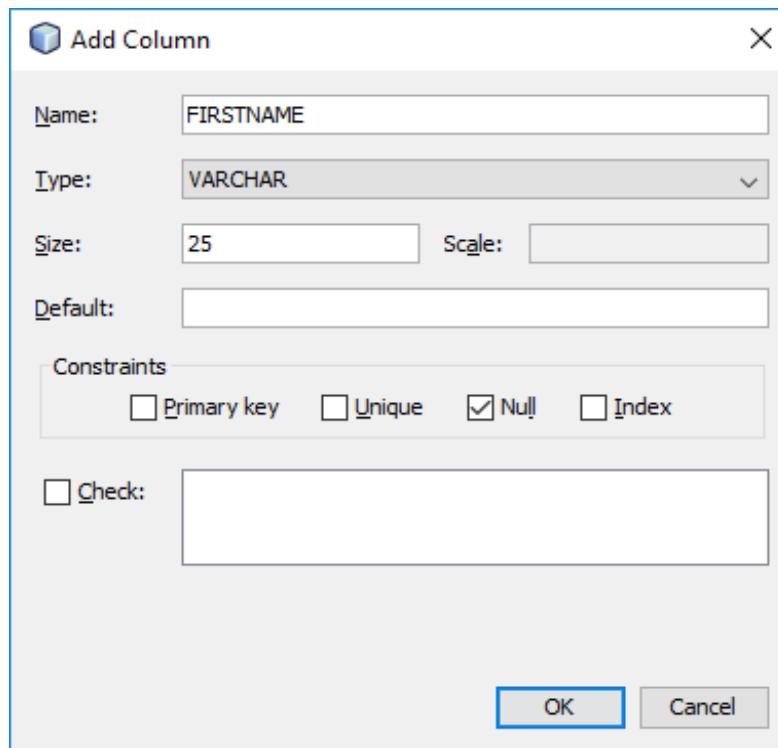


6. Now click on Add column button to add columns in table.

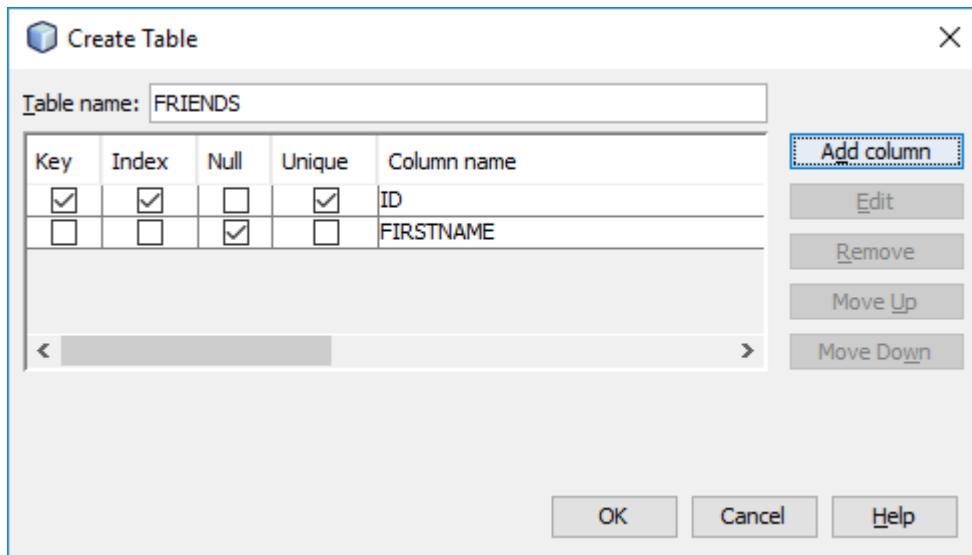
Enter details as in below pic and select Primary key. After that click on OK button.



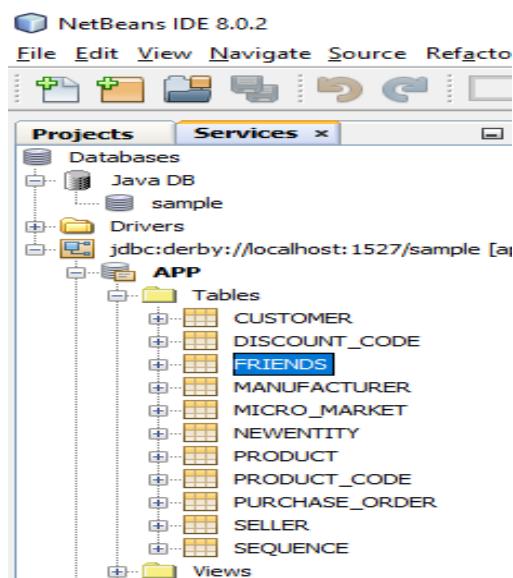
7. Now add second column with following detail. But don't select primary and click on OK button.



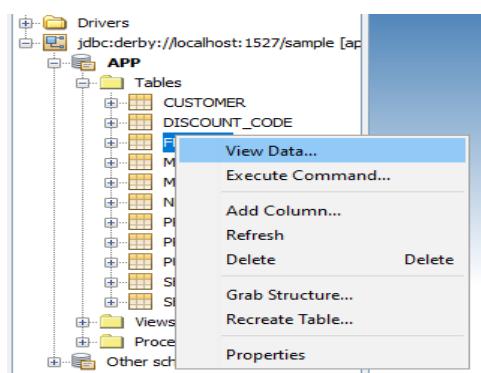
8. Now click on OK button.



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.

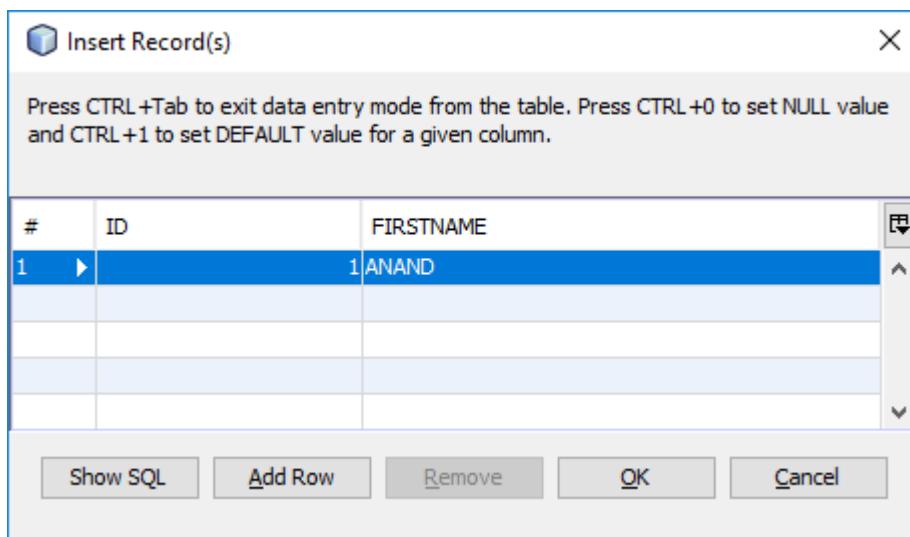


**11. Now click on the leftmost icon in second panel to insert some record.**

The screenshot shows the SQL Workbench/J application. On the left, the 'Projects' and 'Services' tabs are selected. Under 'Databases', there is a 'Java DB' section with a 'sample' database. Below it is a 'Drivers' section and a connection to 'jdbc:derby://localhost:1527/sample [app]'. The 'APP' schema is expanded, showing 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 with a connection to 'jdbc:derby://localhost:1527/sample [app on APP]'. It contains a SQL editor with the query 'select \* from APP.FRIENDS;' and a results grid below it. The results grid shows 7 rows of data with columns '#', 'ID', and 'FIRSTNAME'. The first row is highlighted with a blue background.

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

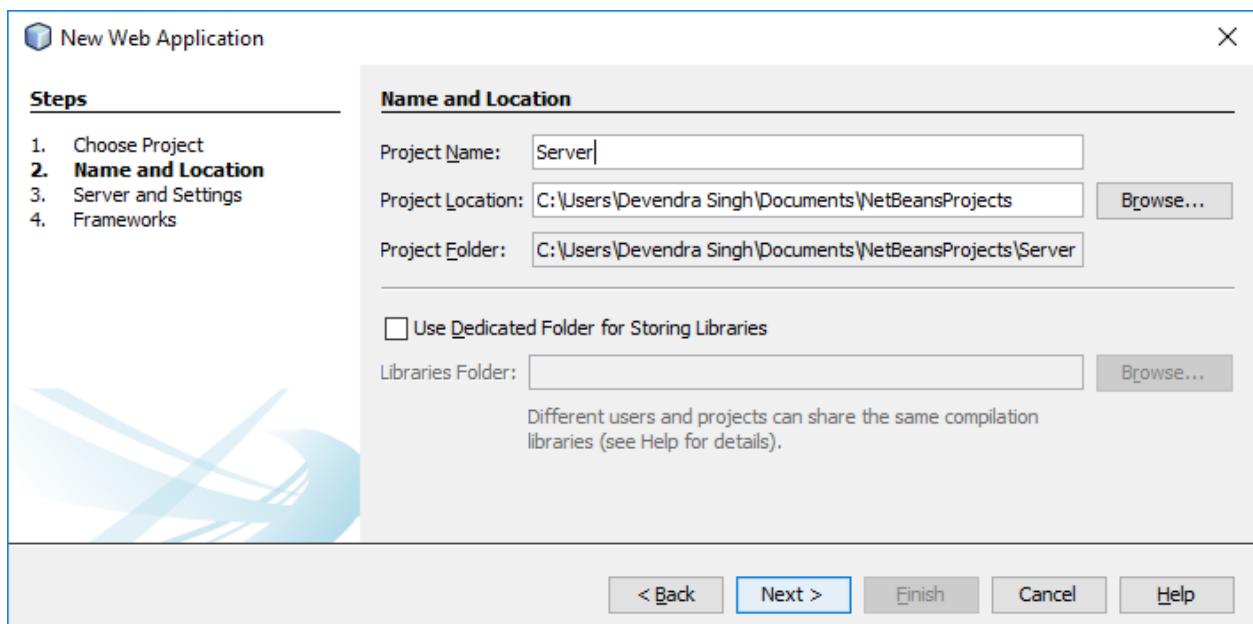
After that click on OK button to finish.



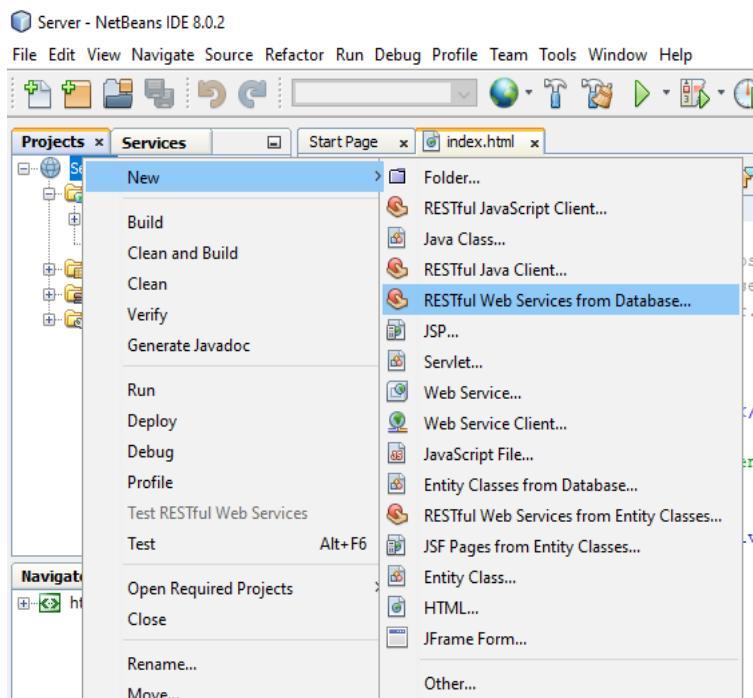
**13. As you can see, I have entered 7 records.**

The screenshot shows the results of the 'select \* from APP.FRIENDS' query in the SQL Workbench/J interface. The results grid has columns '#', 'ID', and 'FIRSTNAME'. The data consists of 7 rows: (1, ANAND), (2, JULHAS), (3, NIKHIL), (4, GAGAN), (5, RAVI), (6, DHARMENDRA), and (7, ADARSH). The 'Page Size' is set to 20, and the 'Total Rows' is 7. The 'Add Row' button is visible at the bottom of the results grid.

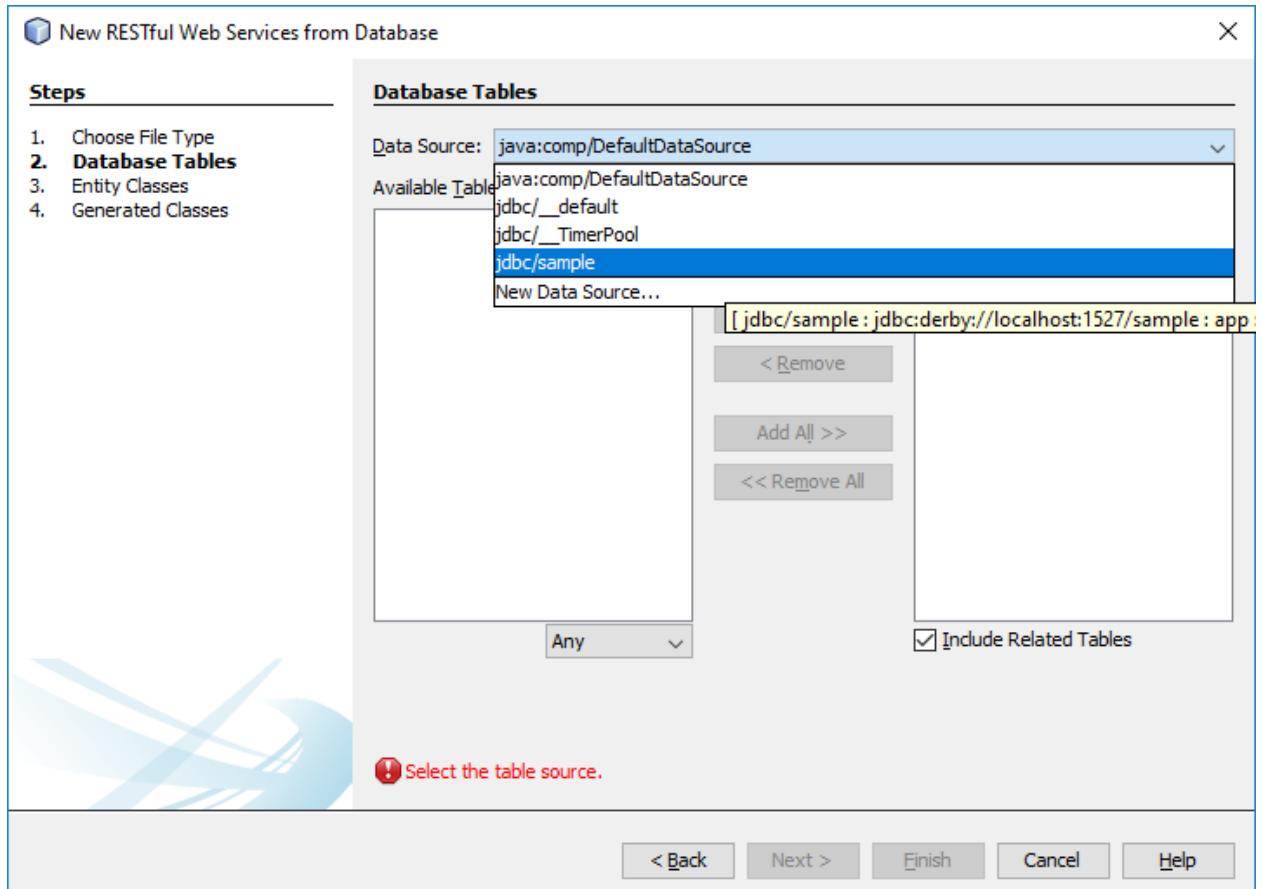
**14. Now create a web application with name Server. After that click on Next and then Finish button.**



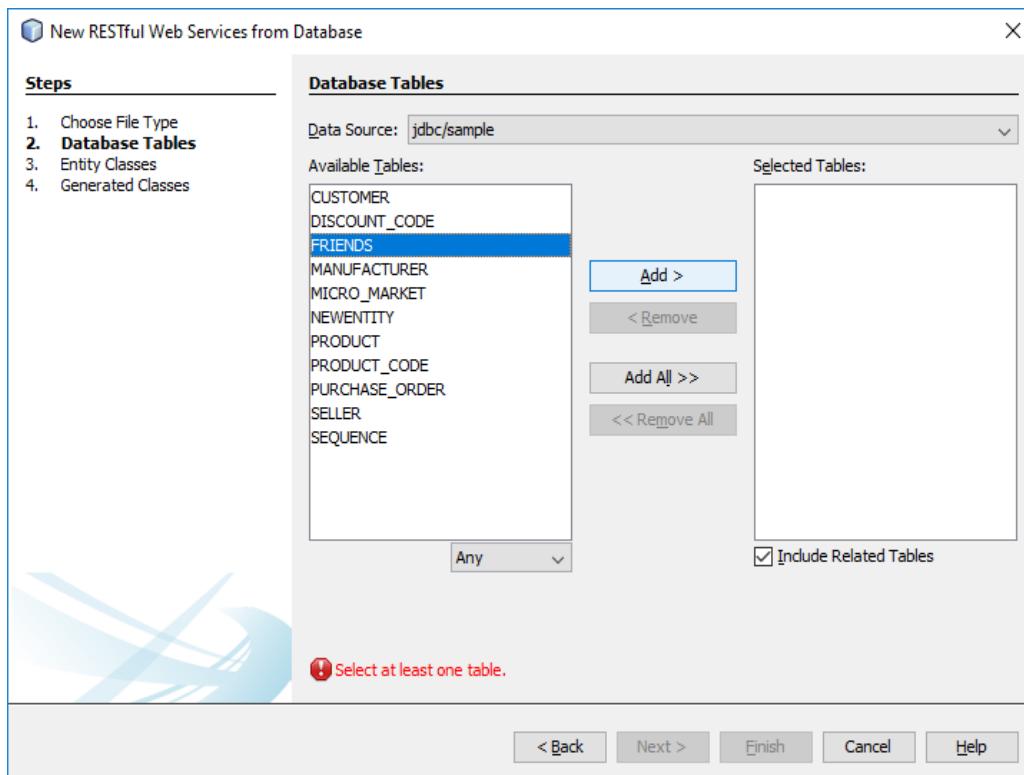
**15. Now create a RESTful Web Service from Database by right click on project name.**



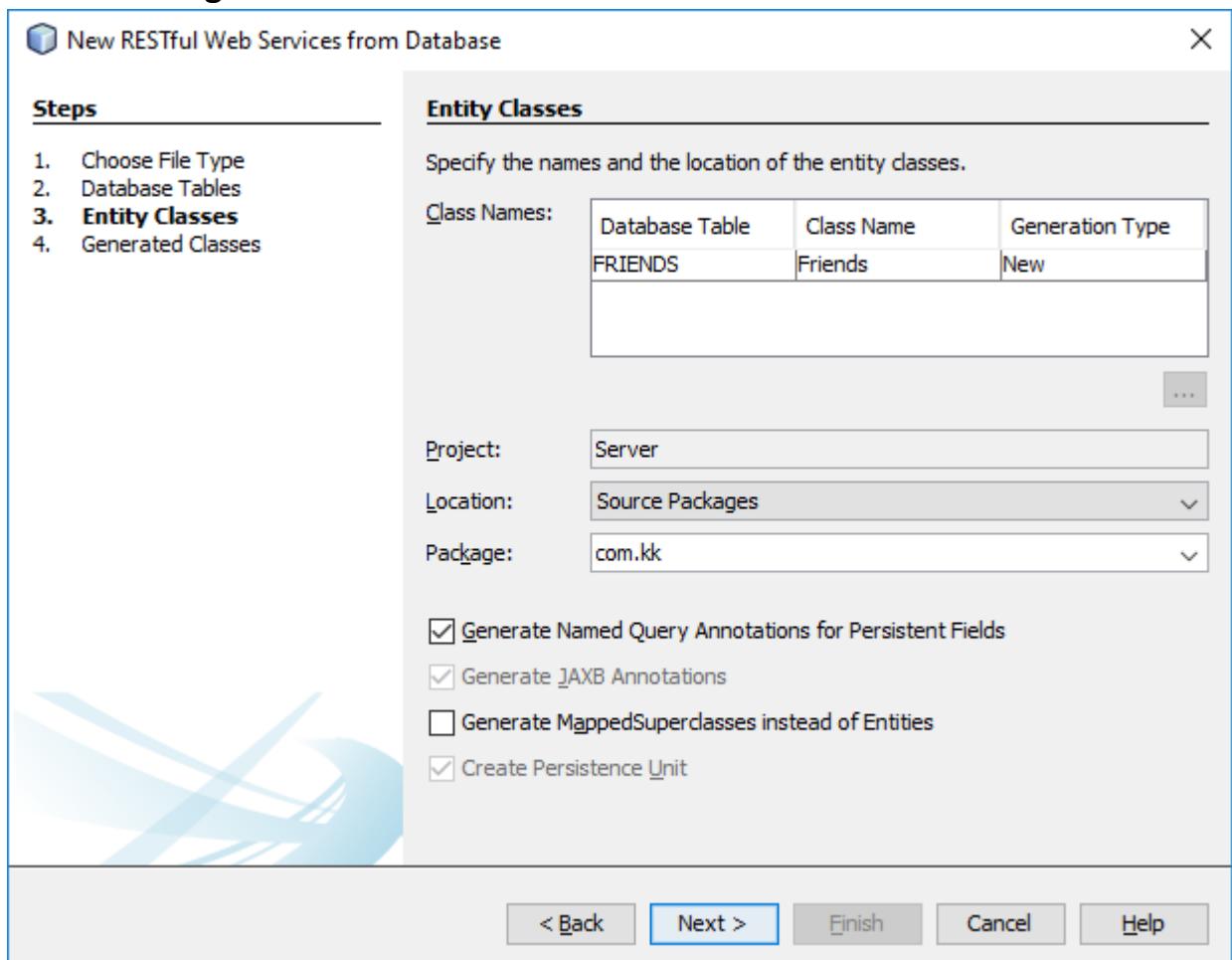
**16. Choose Data Source jdbc/sample.**



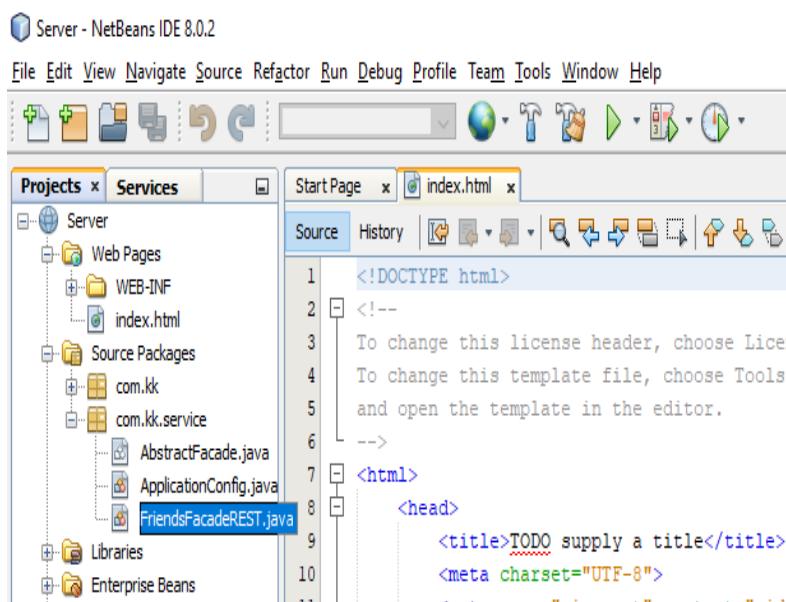
17. Now select FRIENDS and click on Add button. After that click on Next button.



**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.**

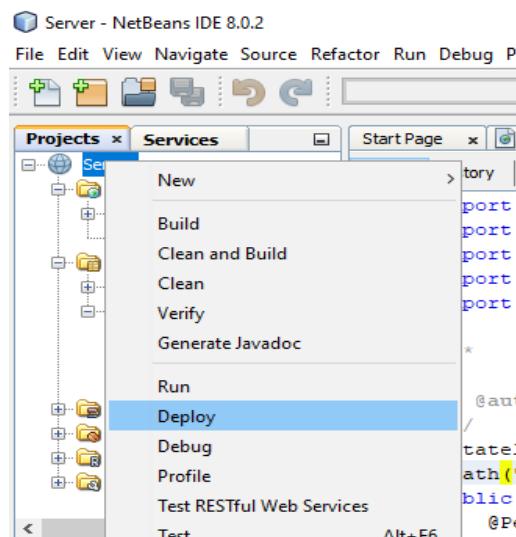


**20. Now remove the selected part from every method in this file. So that it will communicate only in JSON format. You can also use methods to convert it. But this is easiest method.**

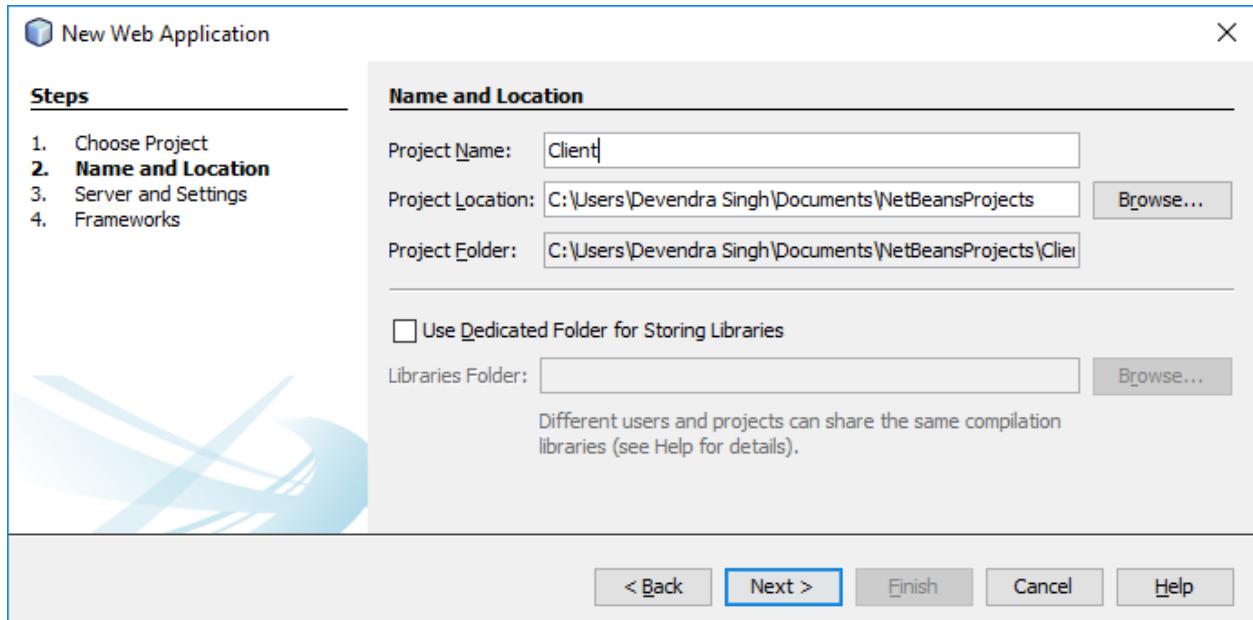
The screenshot shows the NetBeans IDE interface with the code editor open to FriendsFacadeREST.java. The code defines a RESTful service for managing Friends entities. The `@Consumes` annotation on the `create` and `edit` methods is highlighted in yellow, indicating it is selected for deletion. The code is as follows:

```
25 */
26 @Stateless
27 @Path("com.kk.friends")
28 public class FriendsFacadeREST extends AbstractFacade<Friends> {
29 @PersistenceContext(unitName = "ServerPU")
30 private EntityManager em;
31
32 public FriendsFacadeREST() {
33 super(Friends.class);
34 }
35
36 @POST
37 @Override
38 @Consumes({"application/xml", "application/json"})
39 public void create(Friends entity) {
40 super.create(entity);
41 }
42
43 @PUT
44 @Path("{id}")
45 @Consumes({"application/xml", "application/json"})
46 public void edit(@PathParam("id") Integer id, Friends entity) {
47 super.edit(entity);
48 }
49 }
```

**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>

```

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

- Title Bar:** Client - NetBeans IDE 8.0.2
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations, search, and navigation.
- Projects Tab:** Shows a project named "Client" containing "Web Pages", "Source Packages", "Libraries", and "Configuration Files". A file named "index.html" is selected.
- Navigator Tab:** Shows the structure of the "index.html" file, including sections like "table", "td", "th", "Rules", and "HTML".
- Source Tab:** Displays the code of "index.html". The code is as follows:

```

<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.nn.friends/', true);
 request.onload = function () {
 // begin accessing JSON data here
 var data = JSON.parse(this.response);

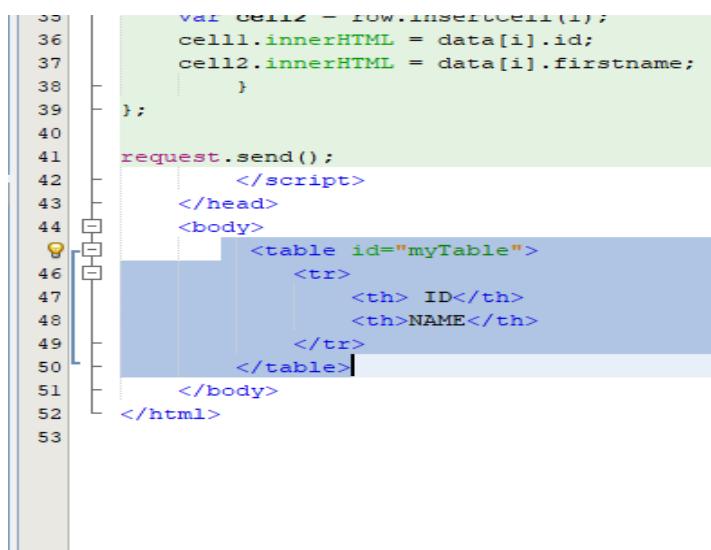
 for (var i = 0; i < data.length; i++) {
 var table = document.getElementById("myTable");
 var row = table.insertRow();
 var cell1 = row.insertCell(0);
 var cell2 = row.insertCell(1);
 cell1.innerHTML = data[i].id;
 cell2.innerHTML = data[i].firstname;
 }
 };
 </script>
 </head>
 <body>
 <table border="1"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

```

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

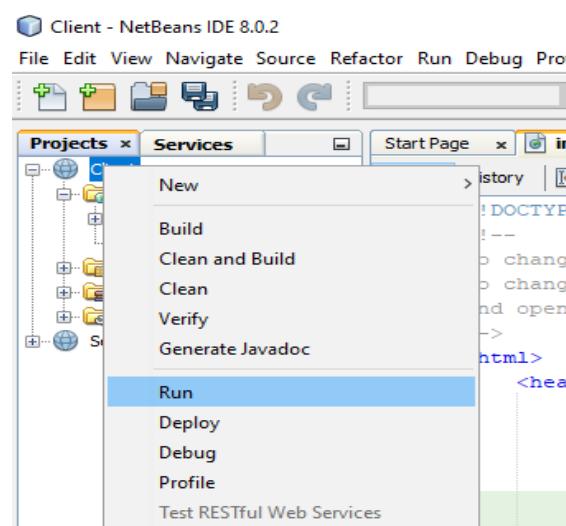
#### 24. Replace the content of body tag with following code.

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



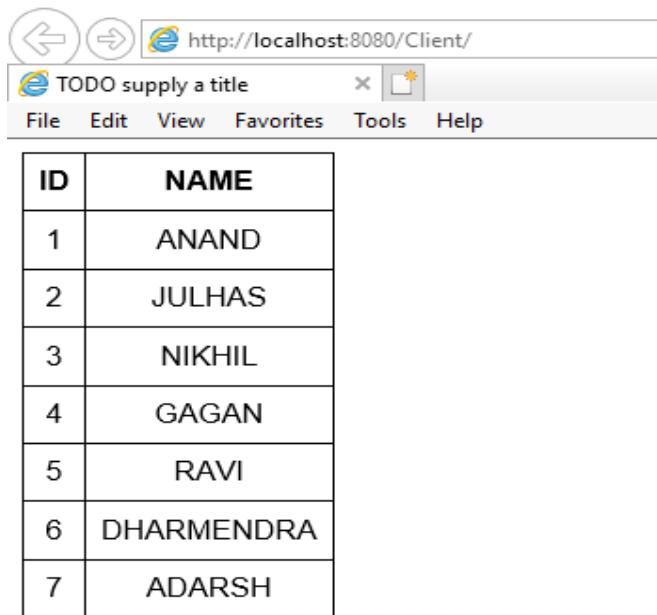
```
35 var cell1 = row.insertCell(1);
36 cell1.innerHTML = data[i].id;
37 cell2.innerHTML = data[i].firstname;
38 }
39 };
40
41 request.send();
42 </script>
43 </head>
44 <body>
45 <table id="myTable">
46 <tr>
47 <th> ID</th>
48 <th>NAME</th>
49 </tr>
50 </table>
51 </body>
52 </html>
53
```

#### 25. Now run the Client Web Application.



**26.** A window will open in browser which represents a data in tabular format.

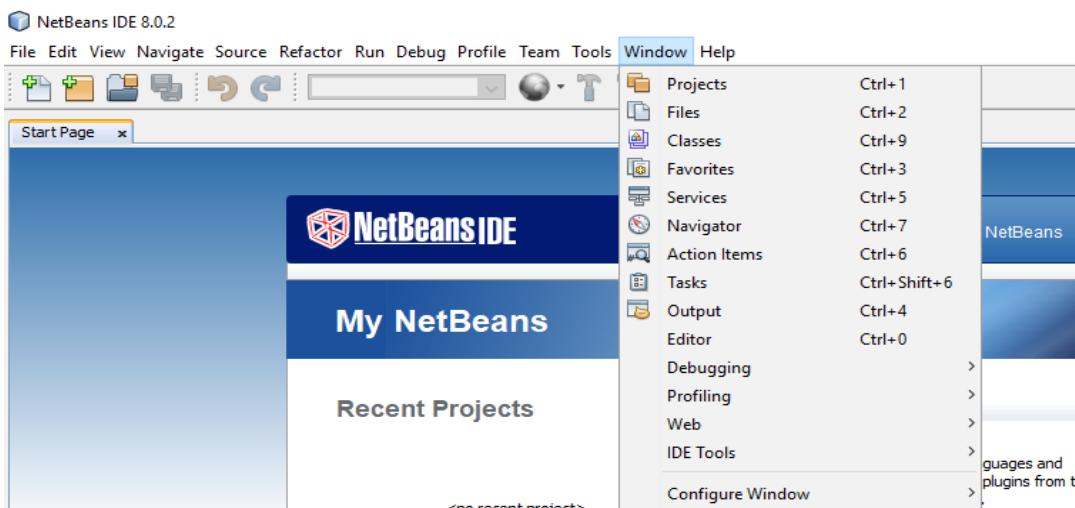
These **data are the records entered in FRIEND table.**



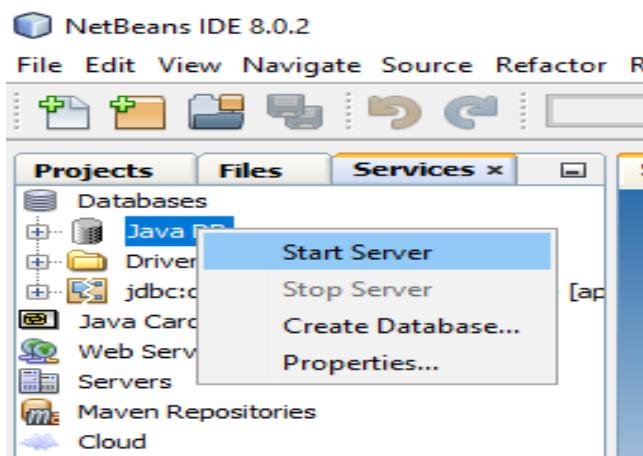
The screenshot shows a web browser window with the URL <http://localhost:8080/Client/>. The title bar says "TODO supply a title". The menu bar includes File, Edit, View, Favorites, Tools, and Help. Below the menu is a table with two columns: ID and NAME. The data rows are:

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

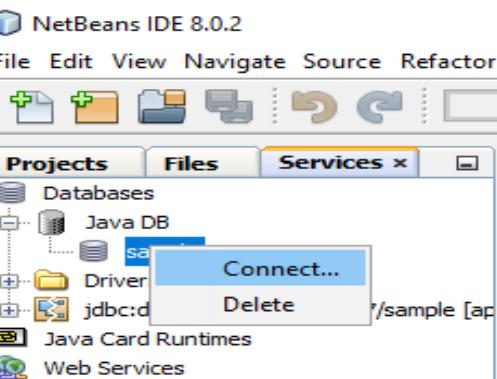
1. Click on Window menu and click on Projects, Files & Services to open it.



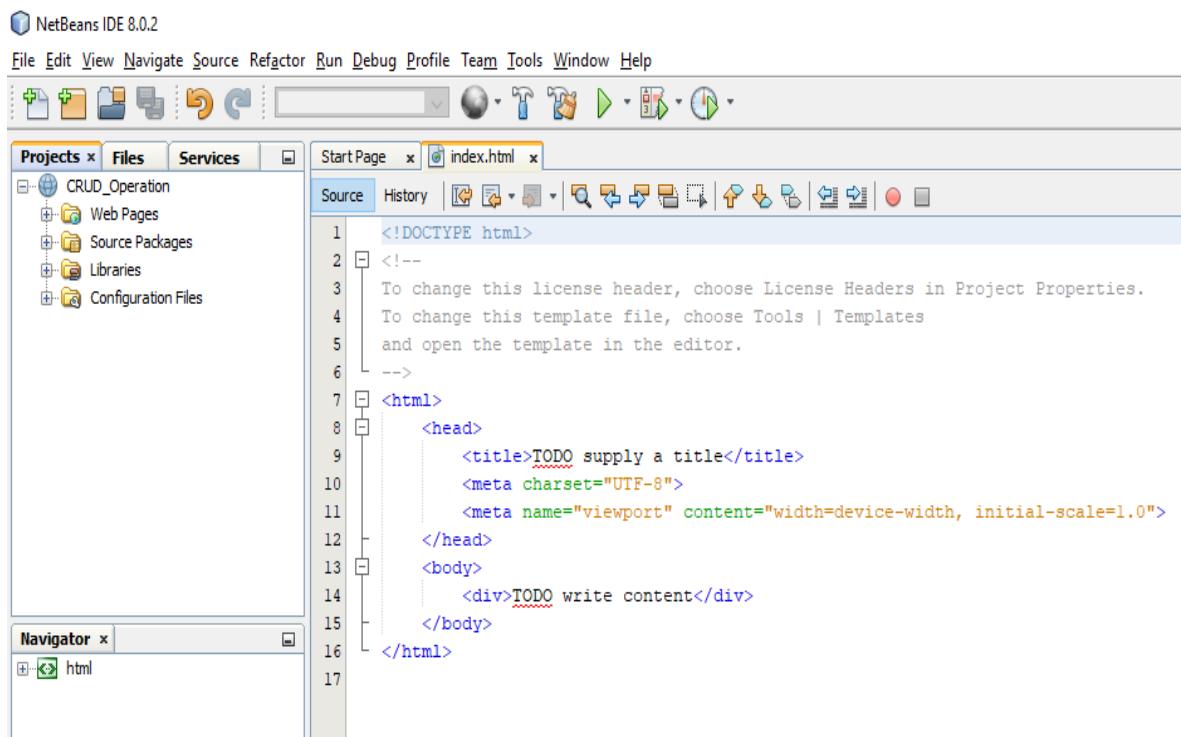
2. Right click on Java DB and then click on Start Server to start the server .



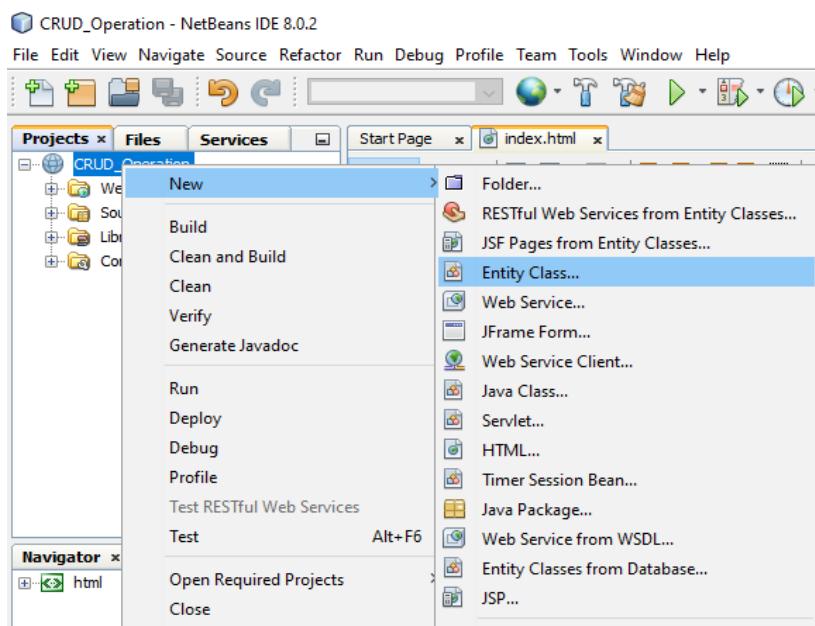
3. Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.



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



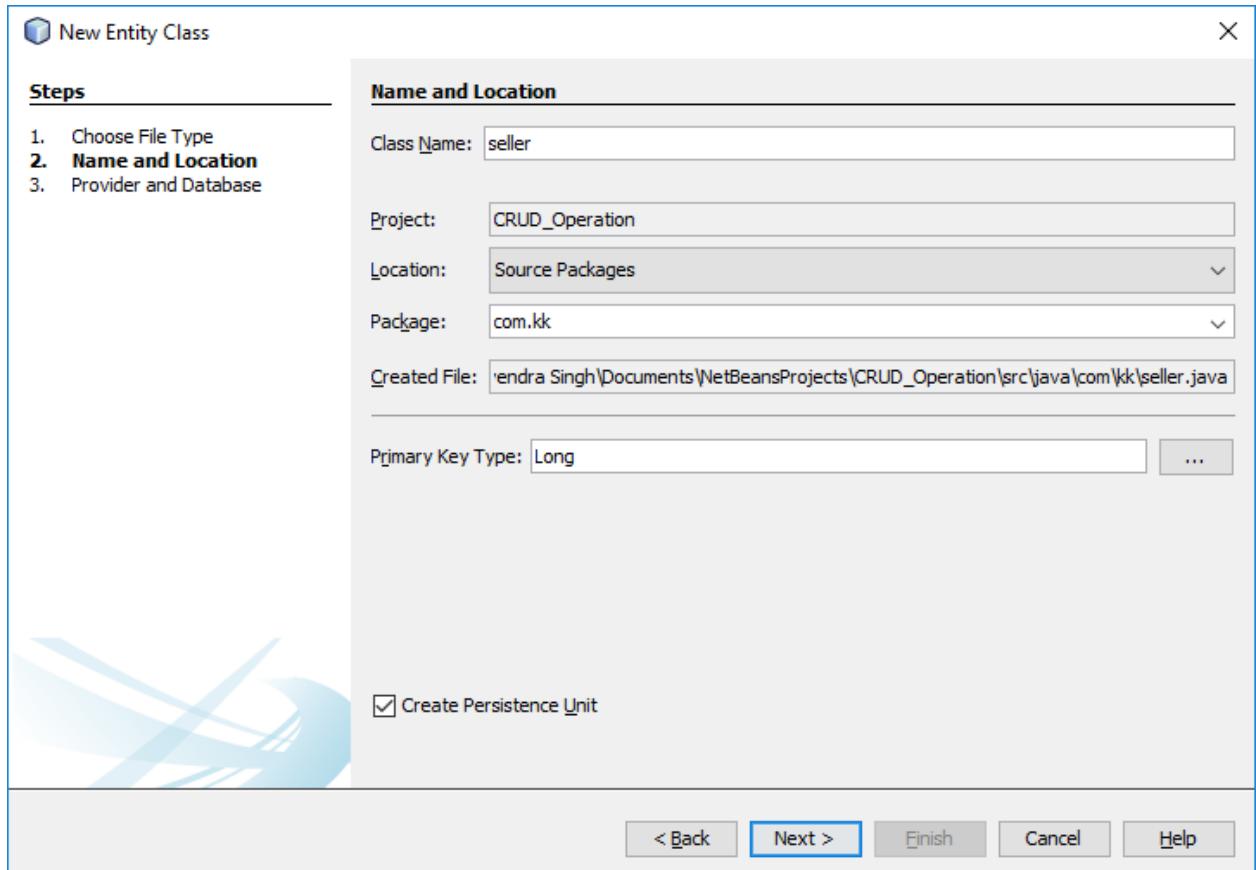
5. Create an entity class. Right click on project name -> New -> Entity Class.



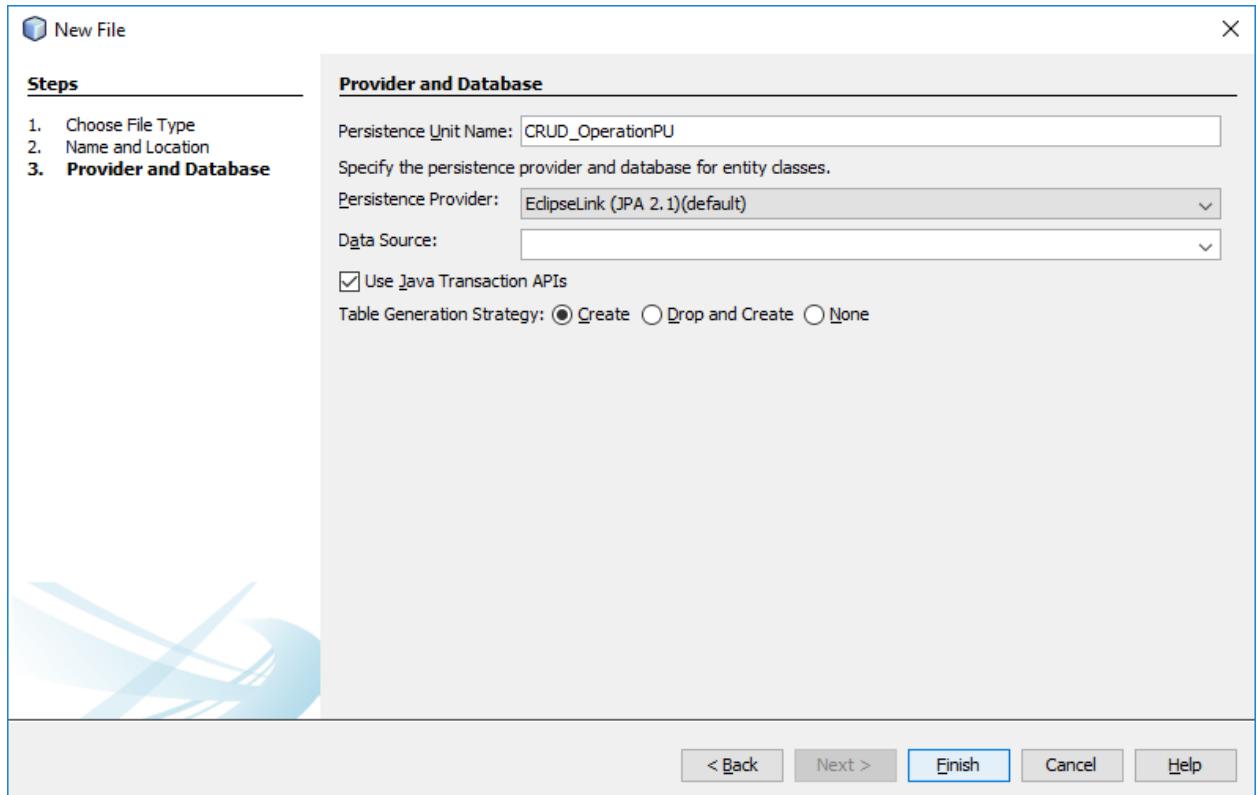
6. A window will appear like bellow pic. Enter following data and click on Next ....

**Class Name -> seller**

**Package Name -> com.kk**

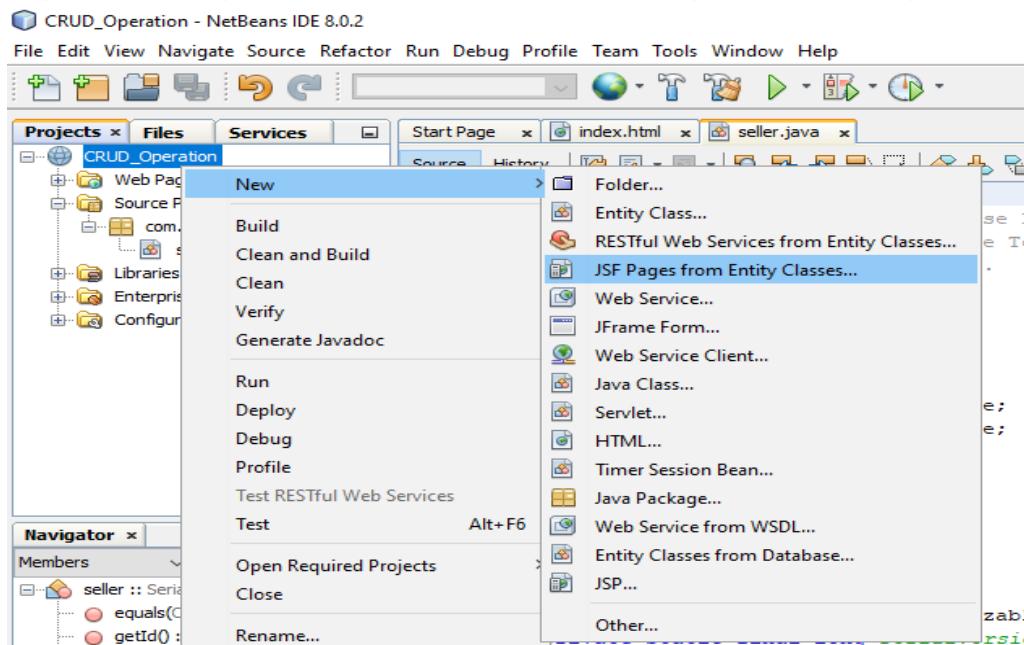


7. Click on Finish.

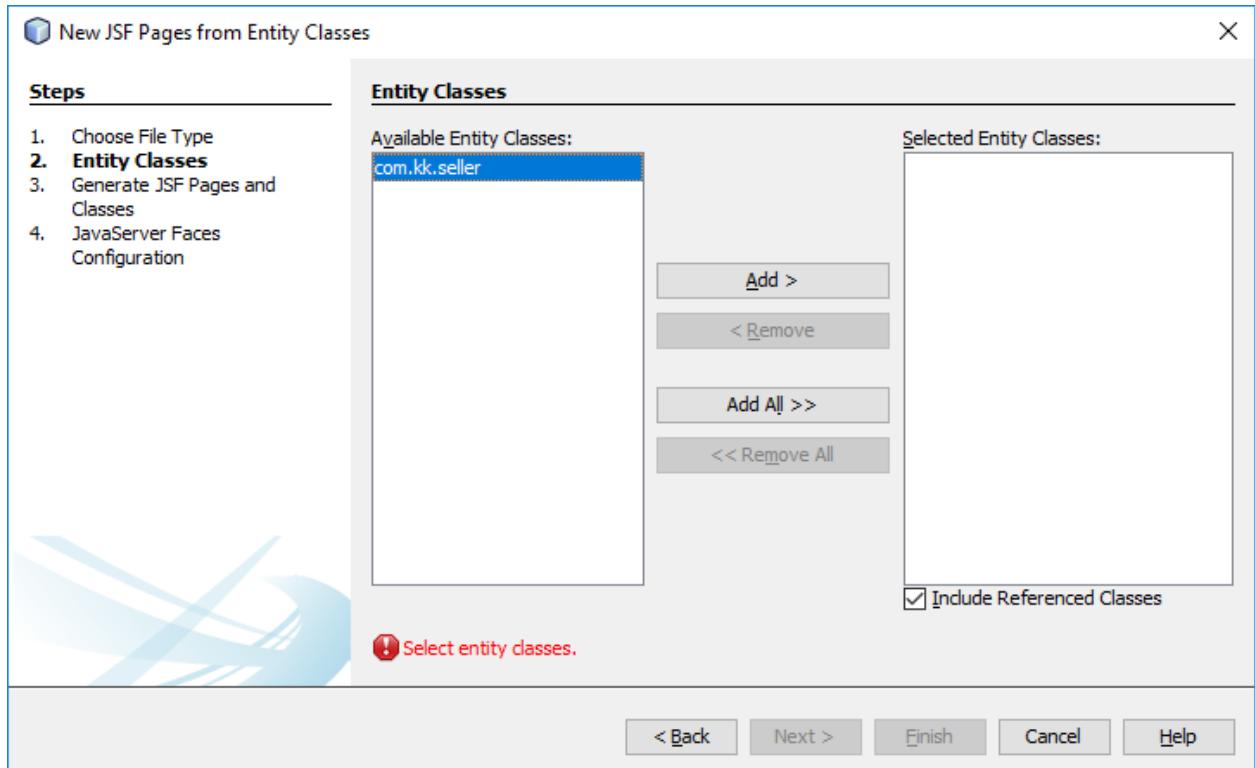


- Right click on project name and create JSF Pages from Entity Classes.

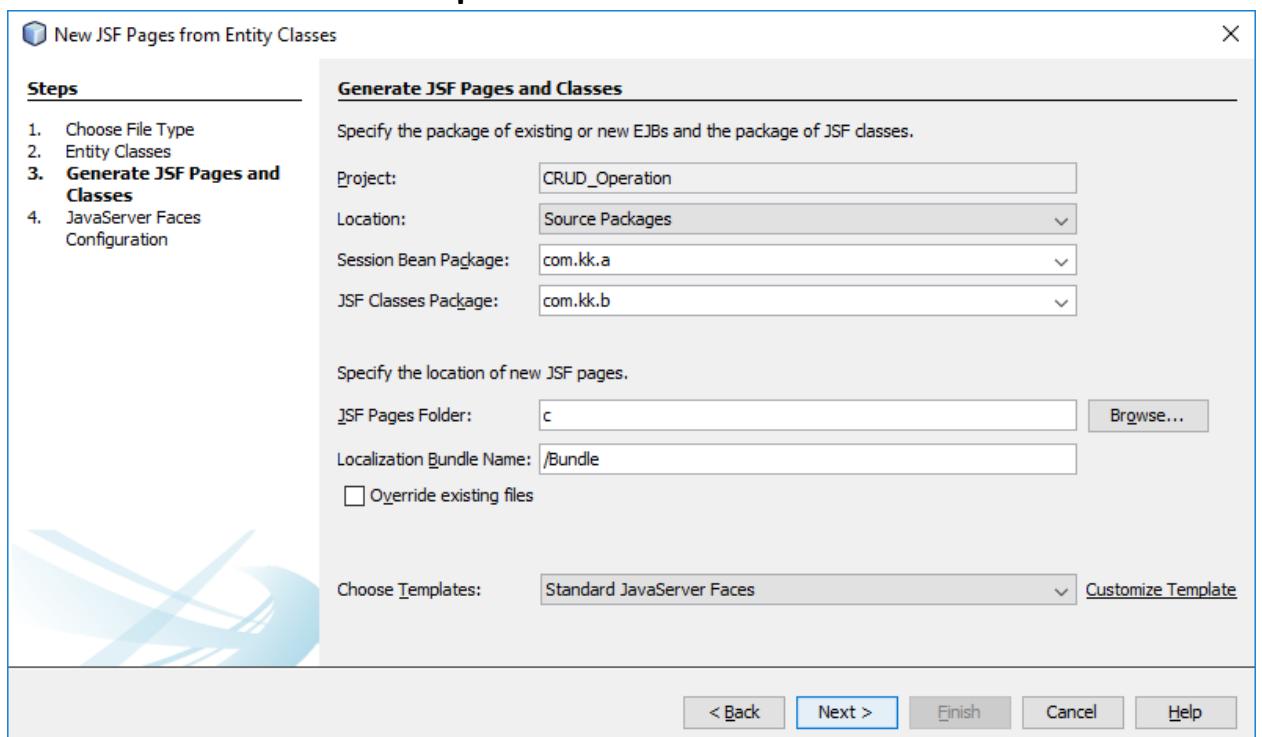
**Right click on project name -> New -> JSF Pages from Entity Classes**



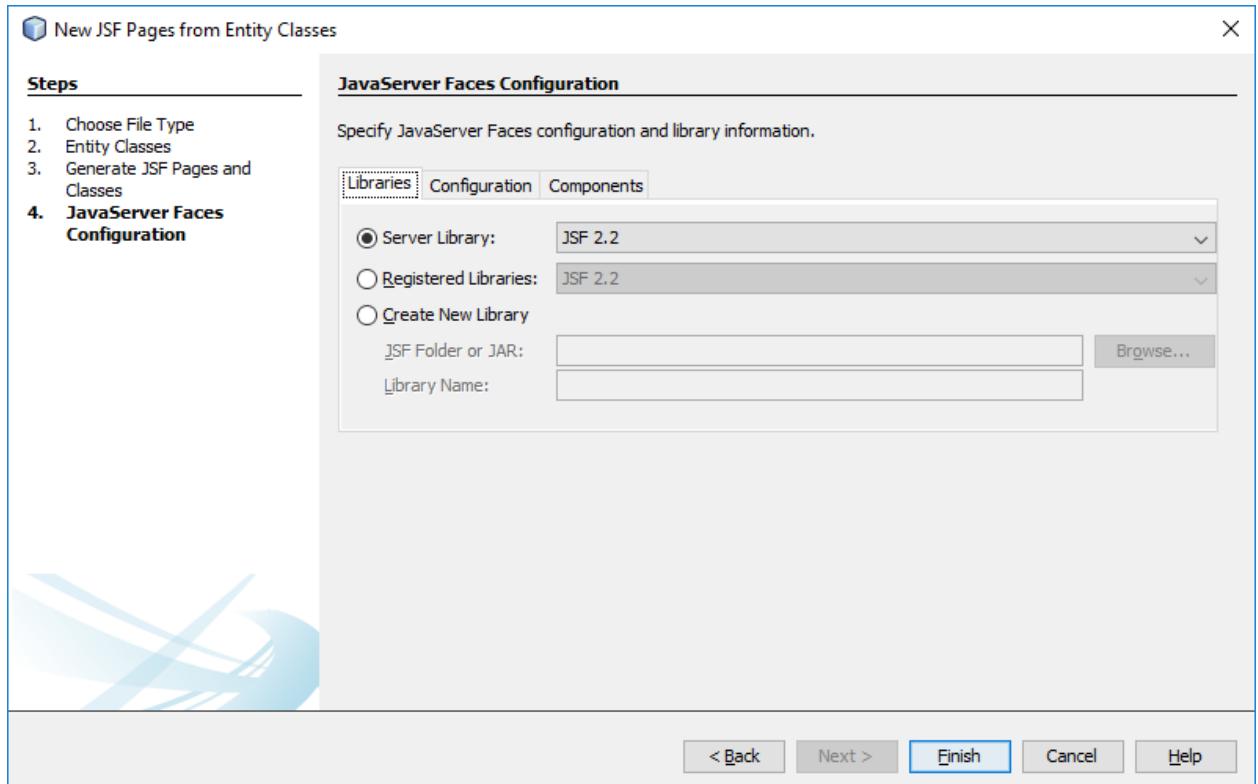
- Select `com.kk.seller` and click on Add button and then Next button on below.



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.

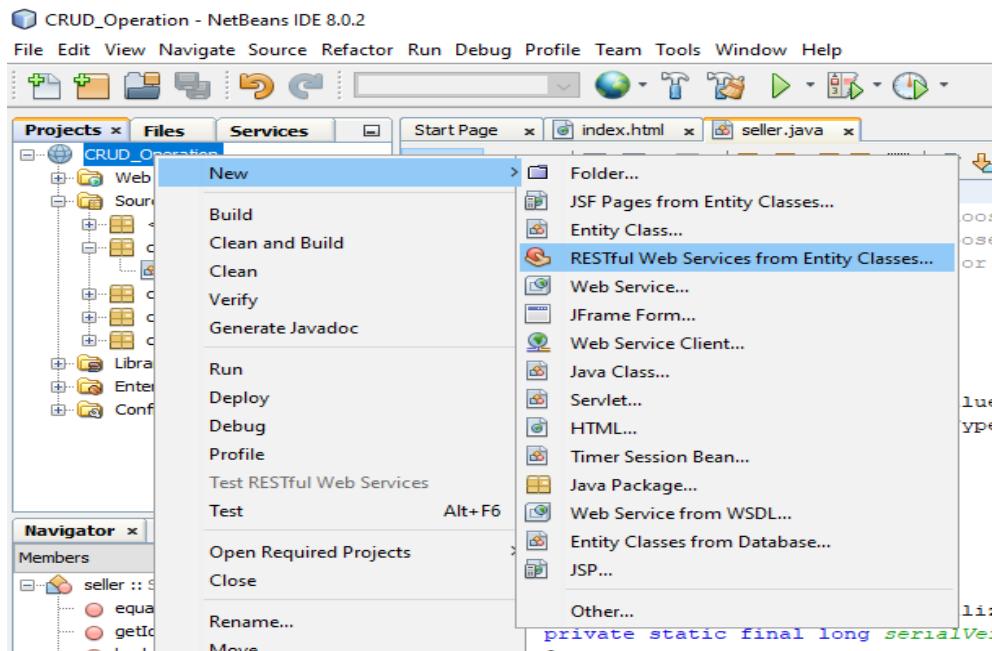


11. Now click on Finish.

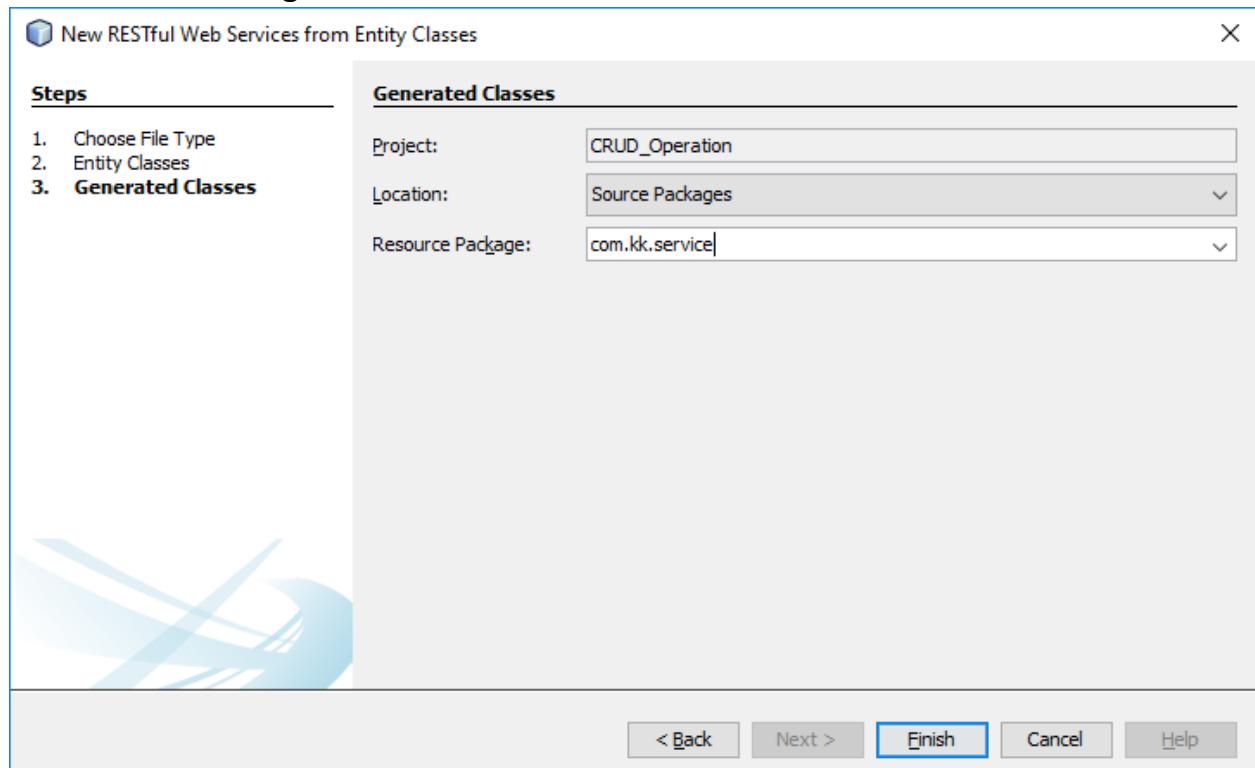


12. Right click on project name and create RESTful Web Services from Entity Classes.

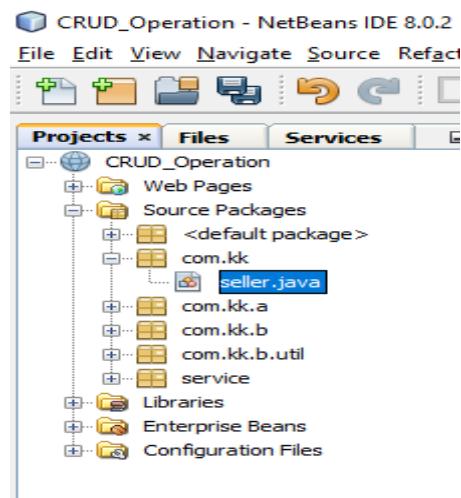
**Right click on project name -> New -> RESTful Web Services from Entity Classes**



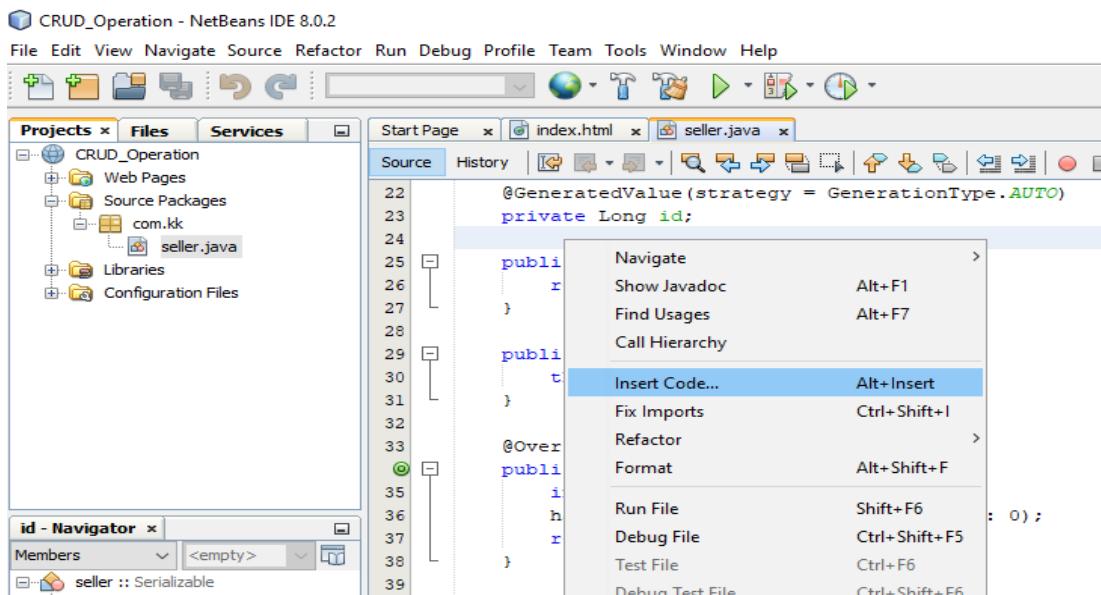
**13. Repeat step 9 and then it will go on next page. Then enter the com.kk.service in Resource Package and then click on Finish button.**



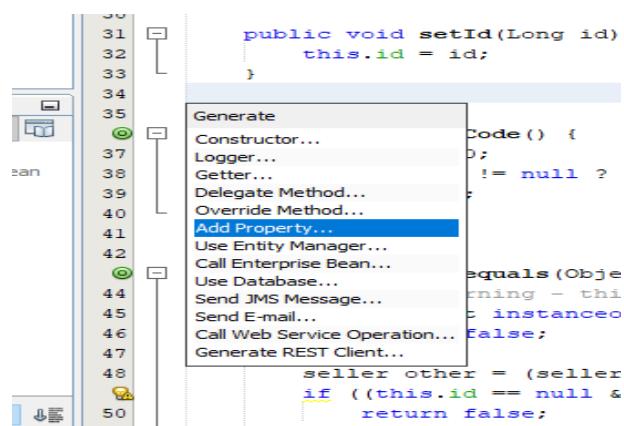
**14. Now open seller.java file under com.kk package.**



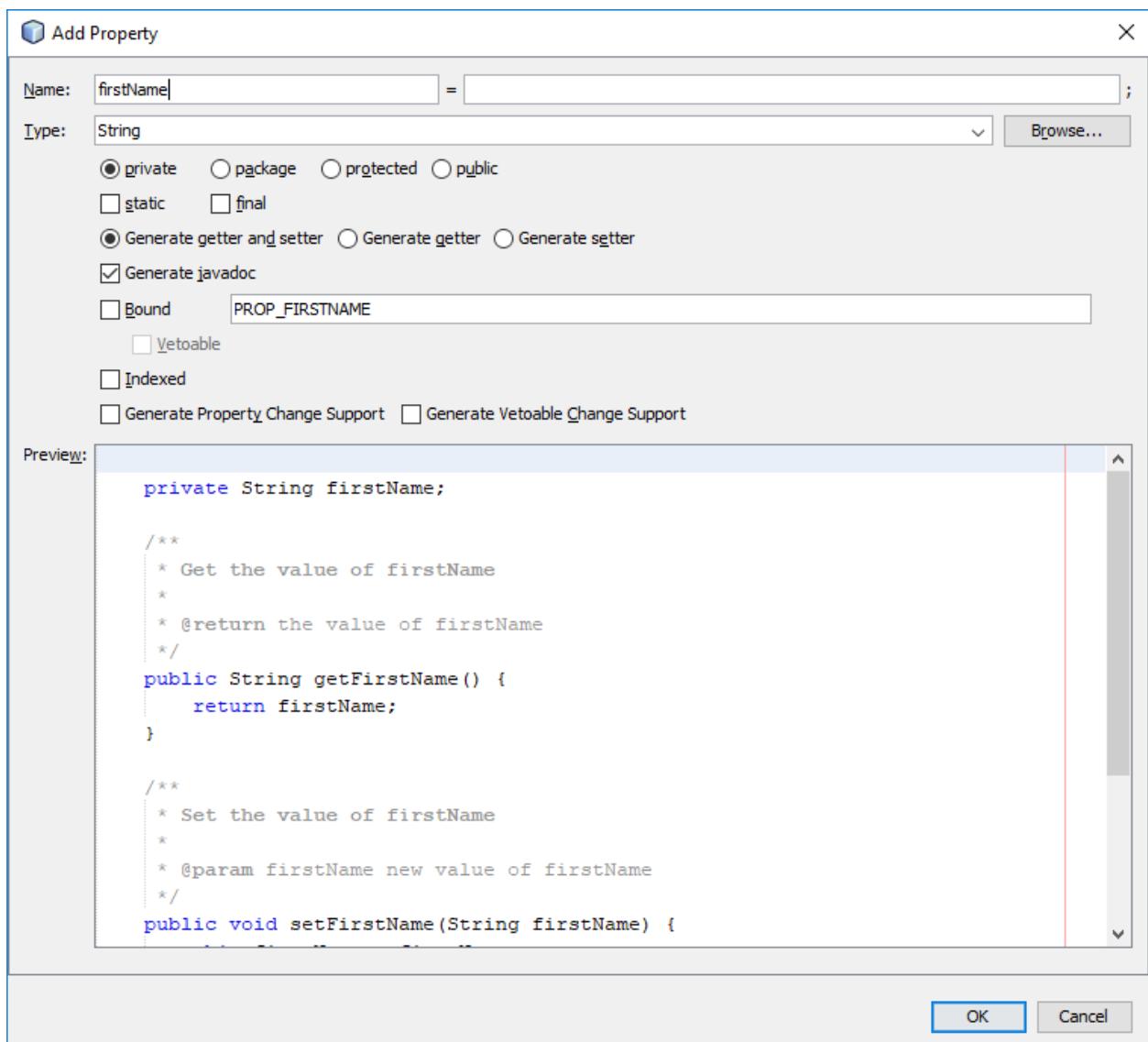
**15. In this file at line number 24, do the right click and select Insert Code.**



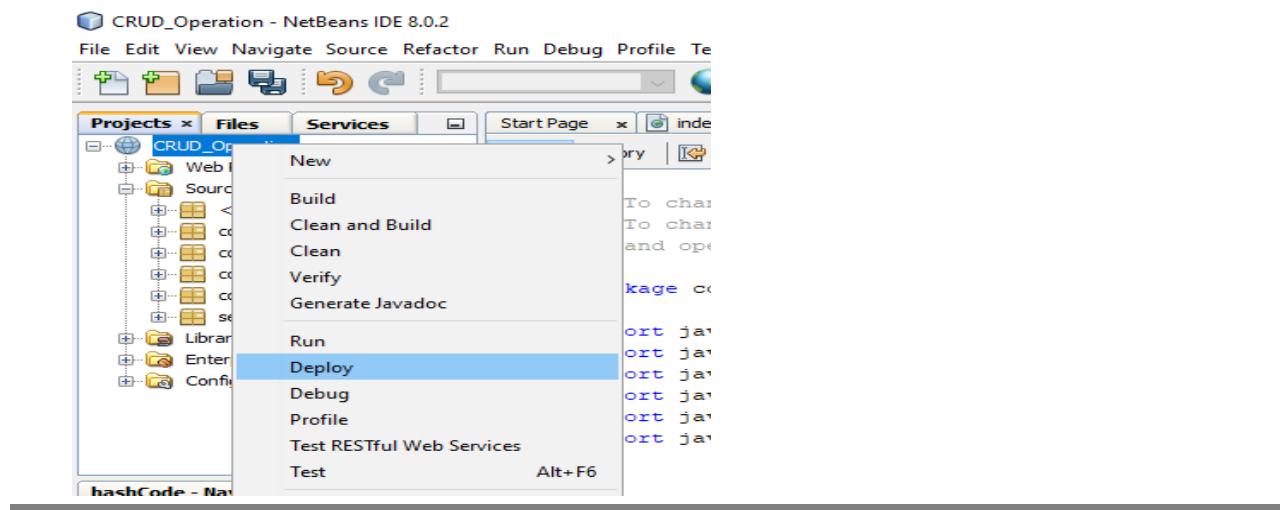
**16.** A new list will appear. Click on Add Property.



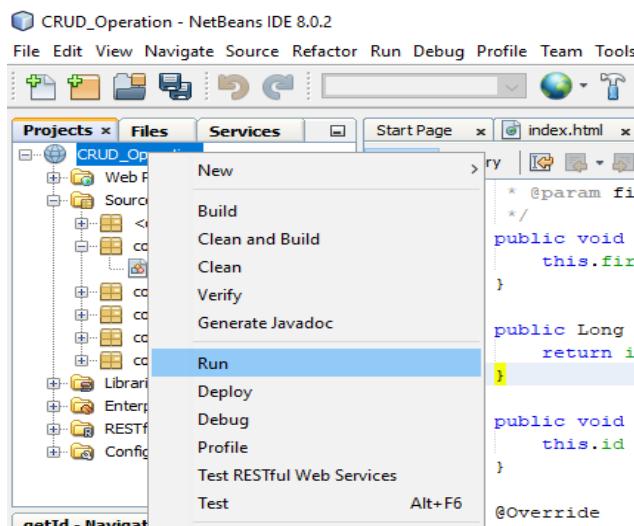
**17.** A new window will open. Enter name as **firstName**. Make sure name should be exact same as of mine and then **click on OK button**. Actually we are setting getter and setter method for **firstName**.



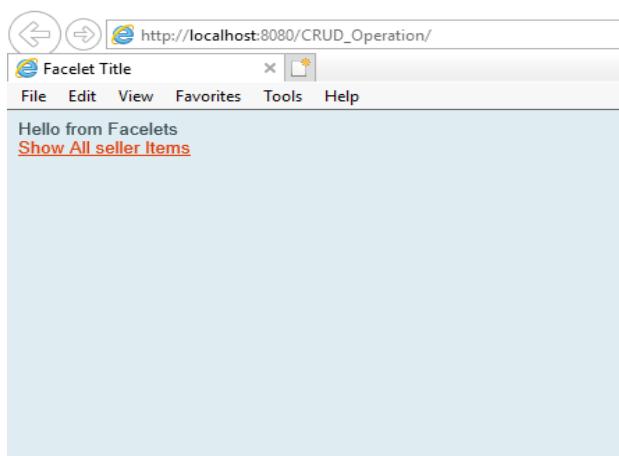
## 18. Now right click on web application name and Deploy it.



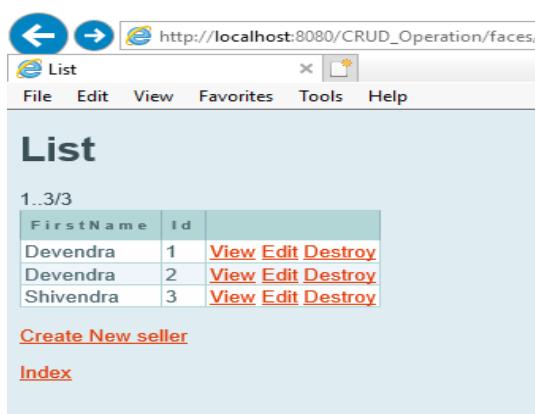
**19. Now right click on project name and run it.**



**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**.

The screenshot shows a web browser window with the URL [http://localhost:8080/CRUD\\_Operation/faces/](http://localhost:8080/CRUD_Operation/faces/). The title bar says "List". The page content displays a table with 3 rows, each representing a seller. The columns are "FirstName" and "Id". The rows are:

| FirstName | Id |
|-----------|----|
| Devendra  | 1  |
| Devendra  | 2  |
| Shivendra | 3  |

Below the table are three hyperlinks for each row: "View", "Edit", and "Destroy". At the bottom of the page are links for "Create New seller", "Index", and "Logout".

**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.

The screenshot shows a web browser window with the URL [http://localhost:8080/CRUD\\_Operation/faces/CreateNewSeller.jsf](http://localhost:8080/CRUD_Operation/faces/CreateNewSeller.jsf). The title bar says "Create New seller". The page content has two input fields: "FirstName" with value "Shivendra" and "Id" with value "4". Below the inputs are three hyperlinks: "Save", "Show All seller Items", and "Index".

**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.

The screenshot shows a web application window titled "List". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The main content area displays a table with 4 rows, labeled "1..4/4". The columns are "FirstName" and "Id". Each row contains a "FirstName" value ("Devendra", "Devendra", "Shivendra", "Shivendra") and an "Id" value (1, 2, 3, 4). To the right of each "Id" is a link group in red: "View", "Edit", and "Destroy". Below the table is a blue button labeled "Create New seller".

| First Name | ID |                                                                   |
|------------|----|-------------------------------------------------------------------|
| Devendra   | 1  | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Destroy</a> |
| Devendra   | 2  | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Destroy</a> |
| Shivendra  | 3  | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Destroy</a> |
| Shivendra  | 4  | <a href="#">View</a> <a href="#">Edit</a> <a href="#">Destroy</a> |

[Create New seller](#)