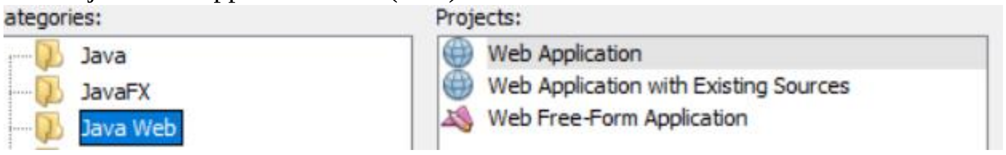
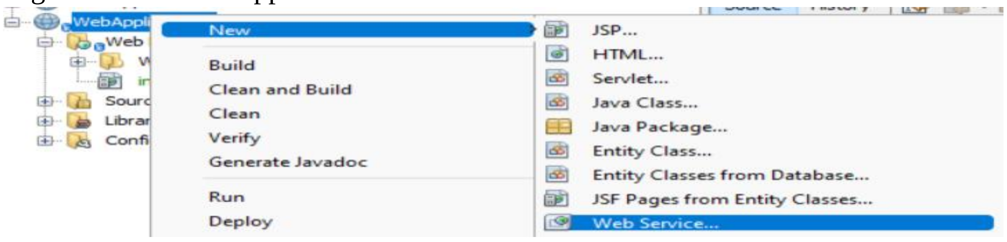
Practical 1

Aim- Simple web service that converts the temperature from Fahrenheit to Celsius and vice a versa

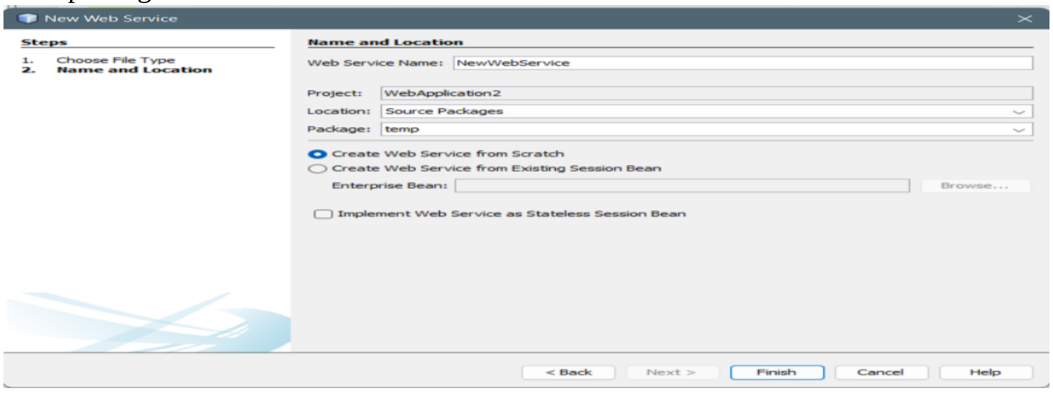
Create java web application next(next) finish


ategories:
Java
JavaFX
Java Web
Projects:
Web Application
Web Application with Existing Sources
Web Free-Form Application

Right click on web application which u created add web service in it



Give package name



Web Service Name: NewWebService

Project: WebApplication2

Location: Source Packages

Package: temp

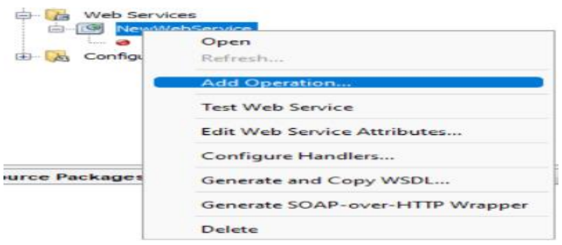
[✓ Create Web Service from Scratch]

Create Web Service from Existing Session Bean

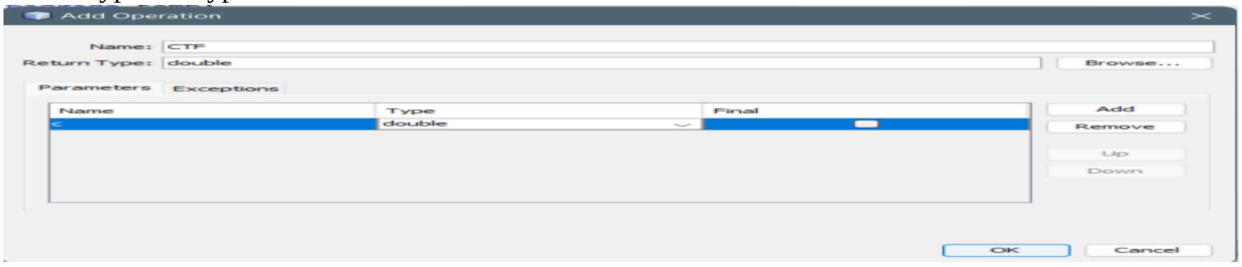
Enterprise Bean:

Implement Web Service as Stateless Session Bean

Click on web service u created right click and add operation



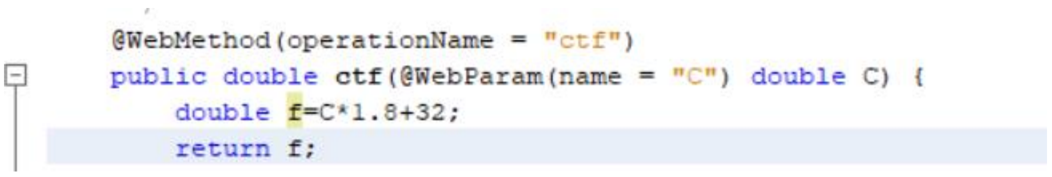
Return type and type same as above Double



Name: CTF

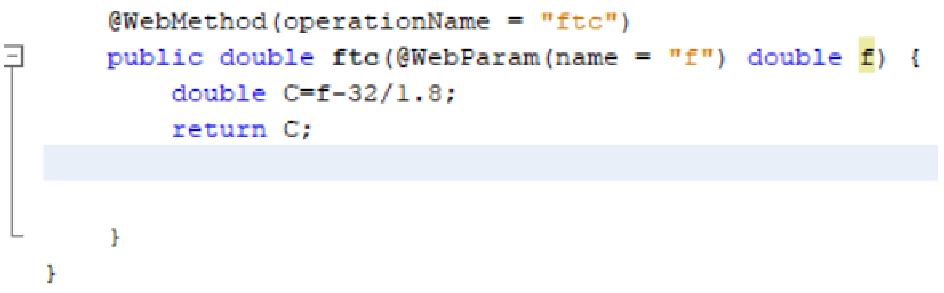
Return Type: double

Name	Type	Final
c	double	



```
@WebMethod(operationName = "ctf")
public double ctf(@WebParam(name = "C") double C) {
    double f=C*1.8+32;
    return f;
```

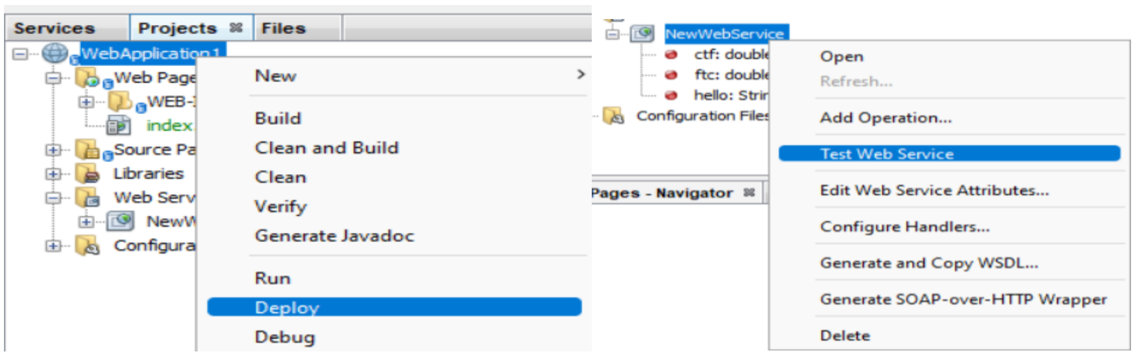
Write the following code in webservice.java
double f=C*1.8+32;
return f;
follow the same process as above to create for Fahrenheit to Celsius



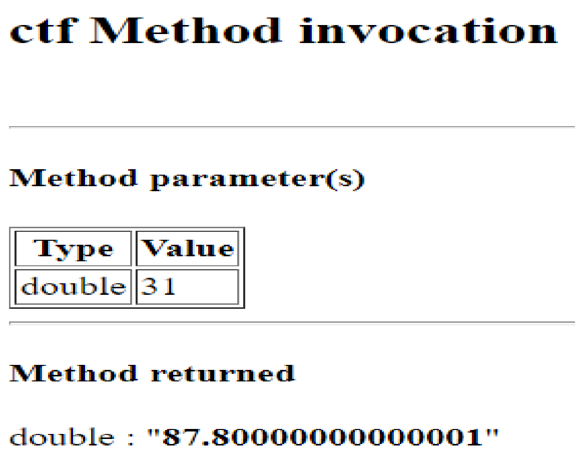
```
    @WebMethod(operationName = "ftc")
    public double ftc(@WebParam(name = "f") double f) {
        double C=f-32/1.8;
        return C;


    }
}
```

Right click on webservice which u created deploy and then test web service



output



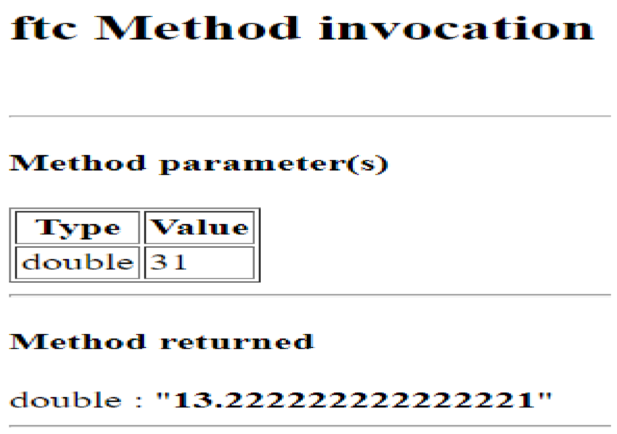
ctf Method invocation

Method parameter(s)

Type	Value
double	31

Method returned

double : "87.80000000000001"



ftc Method invocation

Method parameter(s)

Type	Value
double	31

Method returned

double : "13.222222222222221"

Practical No 2

Aim:Write a program to implement the operation can receive Request and will return a response in two ways.

1. One-Way Operation
2. Request-Response

File (Server)

StudentsFacadeREST.java

```
package com.kk.service;
import com.kk.Students;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Stateless
@Path("com.kk.students")
public class StudentsFacadeREST extends AbstractFacade<Students> {
 @PersistenceContext(unitName = "ServerPU")
 private EntityManager em;
 public StudentsFacadeREST() {
 super(Students.class); }
 @POST
 @Override
 public void create(Students entity) {
 super.create(entity); }
 @PUT
 @Path("{id}")
 public void edit(@PathParam("id") Integer id, Students entity) {
 super.edit(entity); }
 @DELETE
 @Path("{id}")
 public void remove(@PathParam("id") Integer id) {
 super.remove(super.find(id)); }
 @GET
 @Path("{id}")
 @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
 public Students find(@PathParam("id") Integer id) {
 return super.find(id); }
 @GET
 @Override
 @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
 public List<Students> findAll() {
 return super.findAll(); }
 @GET
 @Path("{from}/{to}")
 @Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
 public List<Students> findRange(@PathParam("from") Integer from, @PathParam("to") Integer
to) {
 return super.findRange(new int[]{from, to}); }
 @GET
 @Path("count")
 @Produces(MediaType.TEXT_PLAIN)
 public String countREST() {
 return String.valueOf(super.count()); }
 @Override
 protected EntityManager getEntityManager() {
 return em; }}
```

Students.java

```
package com.kk;
import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
@Entity
@Table(name = "STUDENTS")
@XmlRootElement
@NamedQueries({
 @NamedQuery(name = "Students.findAll", query = "SELECT s FROM Students s"),
 @NamedQuery(name = "Students.findById", query = "SELECT s FROM Students s WHERE s.id
= :id"),
 @NamedQuery(name = "Students.findByName", query = "SELECT s FROM Students s WHERE
s.name = :name")})
```

```
public class Students implements Serializable {
 private static final long serialVersionUID = 1L;
 @Id
 @Basic(optional = false)
 @NotNull
 @Column(name = "ID")
 private Integer id;
 @Size(max = 30)
 @Column(name = "NAME")
 private String name;
 public Students() { }
 public Students(Integer id) {
 this.id = id; }
 public Integer getId() {
 return id; }
 public void setId(Integer id) {
 this.id = id;}
 public String getName() {
 return name; }
 public void setName(String name) {
 this.name = name; }
 @Override
 public int hashCode() {
 int hash = 0;
 hash += (id != null ? id.hashCode() : 0);
 return hash; }
 @Override
 public boolean equals(Object object) {
 // TODO: Warning - this method won't work in the case the id fields are not set
 if (!(object instanceof Students)) {
 return false; }
 Students other = (Students) object;
 if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
 return false; }
 return true; }
 @Override
 public String toString() {
 return "com.kk.Students[ id=" + id + " ]"; }}
```
ServerClient(File)

del_data.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
 <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
 <title>JSP Page</title> </head> <body>
 <%@ page import="com.ss.deleteData" %>
 <% String id=request.getParameter("Id");
 deleteData.deldata(id); %>
 </body></html>
```

getData.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html> <head>
 <title> TODO Supply a title </title>
 <meta charset="UTF-8" >
 <meta name="viewport" content="width=device-width,initial-scale=1.0">
 <style>
 table{
 font-family: arial,sans-serif;
 border-collapse: collapse; }
 td,th{
 border: 1px solid #000000;
 text-align: center;
 padding: 8px; }
 </style> <script>
 var request = new XMLHttpRequest();
 request.open('GET','http:://localhost:8080/Server/webresources/com.kk.students/',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("STUDENTS");
 var row=table.insertRow();
 var cell1=row.insertCell(0);
 var cell2=row.insertCell(1);
 cell1.innerHTML=data[i].id;
 cell2.innerHTML=data[i].name; } };
 request.send();
 </script> </head> <body>
 <table id="STUDENTS"><tr> <th> ID  </th>
 <th>NAME</th>
 </tr> </table> </body></html>
```

Index.html

```
<html><head>
<title> TODO supply a title</title>
<meta charset=”UTF-8”>
<meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
</head><body><form>
```

```
<h2> One-way Operation</h2><br><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>Reuest-Response Operation</h2>
<input type=”submit” formaction=”getData.jsp” value=”get data”>
</form></body></html>
```

Data.java

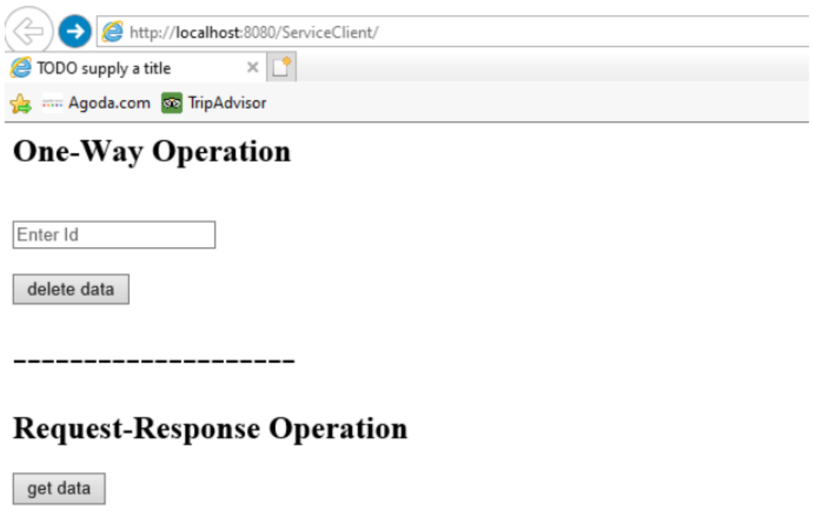
```
package com.ss;
import javax.ws.rs.ClientErrorException;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.WebTarget;
public class Data {
 private WebTarget webTarget;
 private Client client;
 private static final String BASE_URI = "http://localhost:8080/Server/webresources";
 public Data() {
 client = javax.ws.rs.client.ClientBuilder.newClient();
 webTarget = client.target(BASE_URI).path("com.kk.students"); }
 public String countREST() throws ClientErrorException {
 WebTarget resource = webTarget;
 resource = resource.path("count");
 return resource.request(javax.ws.rs.core.MediaType.TEXT_PLAIN).get(String.class); }
 public void edit(String id) throws ClientErrorException {
 webTarget.path(java.text.MessageFormat.format("{0}", new Object[]{id})).request().put(null); }
 public <T> T find_XML(Class<T> responseType, String id) throws ClientErrorException {
 WebTarget resource = webTarget;
 resource = resource.path(java.text.MessageFormat.format("{0}", new Object[]{id}));
 return resource.request(javax.ws.rs.core.MediaType.APPLICATION_XML).get(responseType); }
 public <T> T find_JSON(Class<T> responseType, String id) throws ClientErrorException {
 WebTarget resource = webTarget;
 resource = resource.path(java.text.MessageFormat.format("{0}", new Object[]{id}));
 return resource.request(javax.ws.rs.core.MediaType.APPLICATION_JSON).get(responseType); }
 public <T> T findRange_XML(Class<T> responseType, String from, String to) throws
ClientErrorException {
 WebTarget resource = webTarget;
 resource = resource.path(java.text.MessageFormat.format("{0}/{1}", new Object[]{from, to}));
 return resource.request(javax.ws.rs.core.MediaType.APPLICATION_XML).get(responseType); }
 public <T> T findRange_JSON(Class<T> responseType, String from, String to) throws
ClientErrorException {
 WebTarget resource = webTarget;
 resource = resource.path(java.text.MessageFormat.format("{0}/{1}", new Object[]{from, to}));
 return resource.request(javax.ws.rs.core.MediaType.APPLICATION_JSON).get(responseType); }
 public void create() throws ClientErrorException {
```

```
 webTarget.request().post(null);}
 public <T> T findAll_XML(Class<T> responseType) throws ClientErrorException {
 WebTarget resource = webTarget;
 return resource.request(javax.ws.rs.core.MediaType.APPLICATION_XML).get(responseType); }
 public <T> T findAll_JSON(Class<T> responseType) throws ClientErrorException {
 WebTarget resource = webTarget;
 return resource.request(javax.ws.rs.core.MediaType.APPLICATION_JSON).get(responseType); }
 public void remove(String id) throws ClientErrorException {
 webTarget.path(java.text.MessageFormat.format("{0}", new Object[]{id})).request().delete();}
 public void close() {
 client.close(); }}
```

DeleteData.java

```
package com.ss;
public class deleteData {
 public static void deldata(String id){
 String a=id;
 Data ob=new Data();
 ob.remove(a);
 System.out.println("Data is Deleted"); }}
```

output



One-Way Operation

Request-Response Operation

Practical 3

AIM: A PROGRAM TO IMPLEMENT BUSINESS UDDI REGISTRY ENTRY.

Follow below steps to execute UDDI registry program and find below attached code with output.

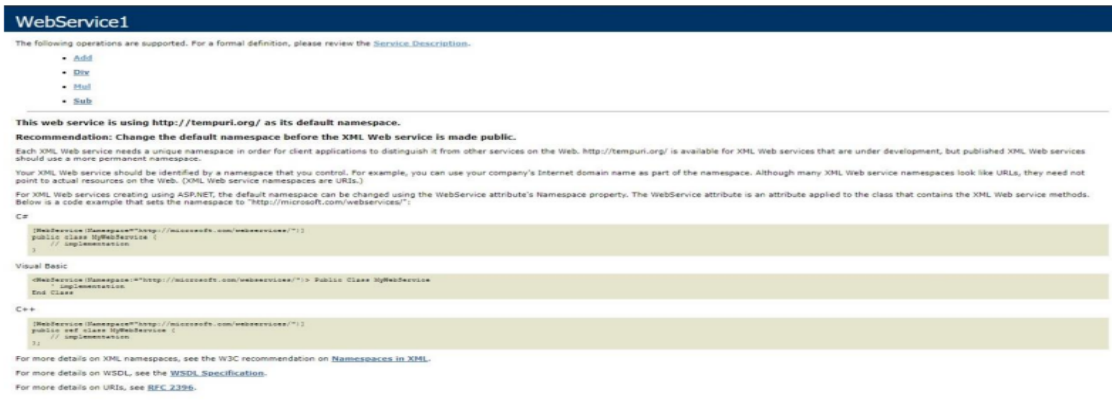
1. Download Apache netbeans IDE 12.2 software and install it.
2. Click on File and new project, create maven project. Next you have to choose a web application, click next and check your workspace location and at last click finish.
3. Your project structure will get created. Inside source packages create new package and give proper package name according to your project name and institute.
4. Inside the project again click on 'New' button and create a new Java class.
5. Start coding in java class by creating constructor and main method.
6. Create static method which will have one url defined and JSON instance created.
7. With the help of HttpURLConnection open new connection by using URL mentioned above.
8. Next you will have to set some required parameters in that connection i.e. connect timeout, request property, DoOutput, DoInput and type of request metod.
9. Then create object of Output Stream and by calling write method get bytes data from json which you mentioned in the initial stage.
10. Create object of Input stream by getting inputstream from connection object.
11. Initialize result variable.
result = IOUtils.toString(in, "UTF-8"); Here 'in' will be inputStream object.
12. Create JSON object like below :
JSONObject myResponse = new JSONObject(result);
13. Finally print result by getting string data from 'myResponse' object.
14. Please handle all exception wherever required. And close connections properly to avoid any loss of data.

CODE:

```
import java.io.BufferedInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import static java.lang.System.in;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;
import static javax.ws.rs.client.Entity.json;
import org.apache.commons.io.IOUtils;
import org.json.JSONObject;
public class Post_JSON {
```

```
public static void main(String[] args) throws IOException {
Post_JSON.Post_JSON();}
public static void Post_JSON() throws IOException {
String query_url = "https://gurujsonrpc.com/guru";
String json = "{ \"method\" : \"guru.test\", \"params\" : [ \"jinu awad\" ],\"id\" : 123 }";{
URL url;
url = new URL(query_url);
HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
conn.setConnectTimeout(5000);
conn.setRequestProperty("Content-Type", "application/json;
charset=UTF-8");
conn.setDoOutput(true);
conn.setDoInput(true);
conn.setRequestMethod("POST");
OutputStream os = conn.getOutputStream();
os.write(json.getBytes("UTF-8"));
os.close();
Logger.getLogger(Post_JSON.class.getName()).log(Level.SEVERE, null,ex);
// read the response
try {
InputStream in = new
BufferedInputStream(conn.getInputStream());
String result;
result = IOUtils.toString(in, "UTF-8");
System.out.println(result);
System.out.println("result after Reading JSON Response");
JSONObject myResponse = new JSONObject(result);
System.out.println("jsonrpc- " + myResponse.getString("jsonrpc"));
System.out.println("id- " + myResponse.getInt("id"));
System.out.println("result- " + myResponse.getString("result"));
} catch (IOException ex) {
Logger.getLogger(Post_JSON.class.getName()).log(Level.SEVERE, null,ex);}
try {
in.close();
} catch (IOException ex) {
Logger.getLogger(Post_JSON.class.getName()).log(Level.SEVERE, null,ex);}
conn.disconnect();}}}
```

OUTPUT:



WebService1

The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Div
- Mul
- Sub

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 services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services 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 Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

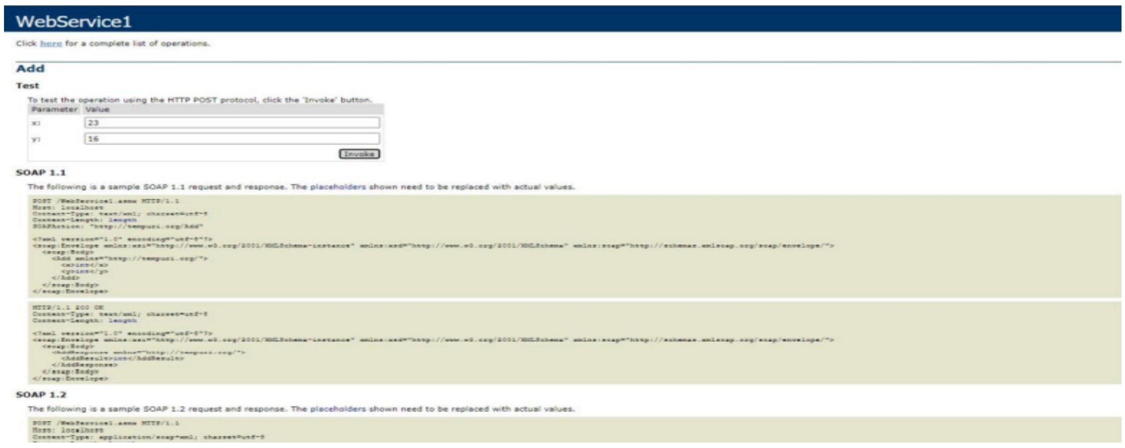
C++

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.



WebService1

Click here for a complete list of operations.

Add

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
x:	23
y:	16

Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /WebService1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Add xmlns="http://tempuri.org/">
      <x>int</x>
      <y>int</y>
    </Add>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>int</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.

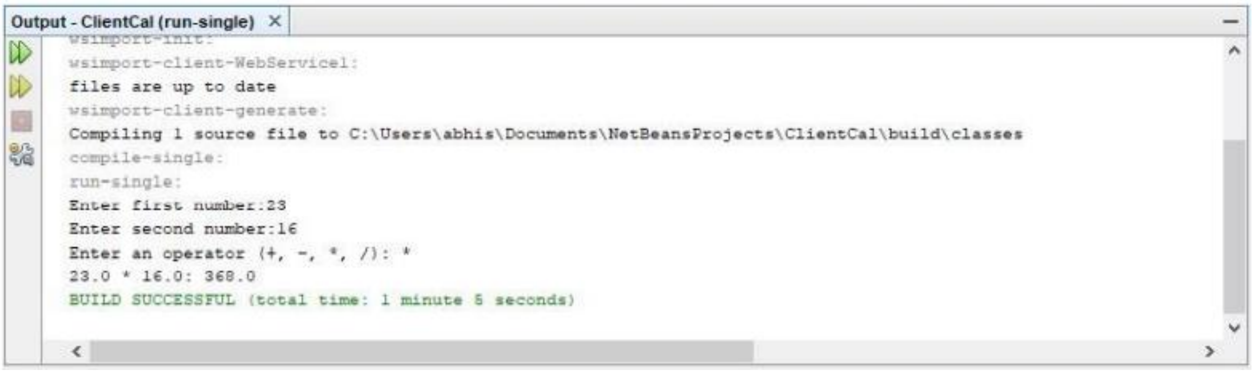
```
POST /WebService1.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
```

Practical 4.

Aim Develop a client which consumes web services developed in different platforms.

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Web; using
System.Web.Services;
namespace Calculator {
 /// <summary>
 /// Summary description for WebService1
 /// </summary>
 [WebService(Namespace = "http://tempuri.org/")]
 [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
 [System.ComponentModel.ToolboxItem(false)]
 // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment
the following line.
 // [System.Web.Script.Services.ScriptService]
 public class WebService1 : System.Web.Services.WebService {
 public WebService1() {
 //Uncomment the following line if using designed components
 //InitializeComponent();  }
 [WebMethod]
 public int Add(int x, int y)  {
 return x + y;  }
 [WebMethod]
 public int Sub(int x, int y)  {
 return x - y;  }
 [WebMethod]
 public int Mul(int x, int y)  {
 return x * y;  }
 [WebMethod]
 public int Div(int x, int y)  {
 return x / y;  }  } }
```

OUTPUT



```
wsimport-init:
wsimport-client-WebService1:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\Users\abhis\Documents\NetBeansProjects\ClientCal\build\classes
compile-single:
run-single:
Enter first number:23
Enter second number:16
Enter an operator (+, -, *, /): *
23.0 * 16.0: 368.0
BUILD SUCCESSFUL (total time: 1 minute 5 seconds)
```

Practical No:- 5

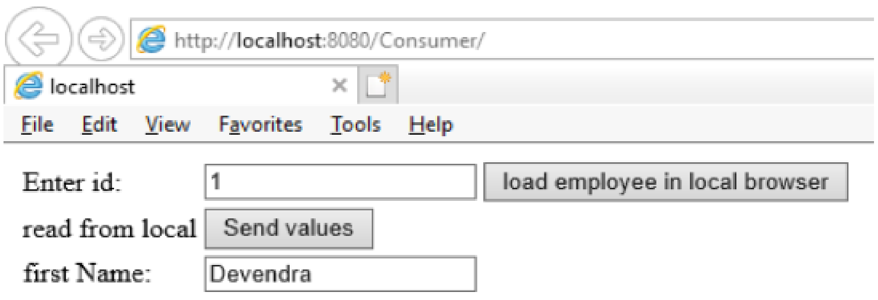
Aim:- Write a JAX-WS web service to perform the following operations. Define a Servlet / JSP that consumes the web service.

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
8. Right click on project name and create JSF Pages from Entity
Classes.
Right click on project name -> New -> JSF Pages from Entity
Classes
9. Select com.kk.seller and click on Add button and then Next button on below.
10.A window like below will appear on the screen. Enter the data
into that window as entered in below pic and click on Next
button.
11.Now click on Finish.
12.Right click on project name and create RESTful Web Services
from Entity Classes.
Right click on project name -> New -> RESTful Web Services
from Entity Classes
13.Repeat step 9 and then it will go on next page. Then enter the com.kk.service
in Resource Package and then click on Finish button.
14.Now open seller.java file under com.kk package.
15.In this file at line number 24, do the right click and select Insert Code.
10
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. Create an another Web Application project and Give name as Consumer.
20. And create it. Next -> Finish.
21. Now right click on index.html and delete it.
22. Now right click on Web Pages and select JSP to add a JSP page.
23. Give index name to it and then click on Finish.
24. Now index.jsp page will open like below.
25. 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.first
Name;
document.getElementById("sellerid").value=seller.id;}}5
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");
</script></head>
alert("error");}};}
<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>
```

26. Now in the following pic, highlighted URL is most important. I'm explaining it next step.
27. 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 . So if you have changed name anywhere then the above URL will change accordingly.
28.Now open sellerFacadeREST.java file by follow below pic available in CRUD_Operation web application.
29. 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.
30.Now run the Consumer application. A window will open in browser like below.
31.Now if you will enter an id into Enter id textbox which you have created in database 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.


http://localhost:8080/Consumer/
localhost
File
Edit
View
Favorites
Tools
Help
load employee in local browser
Send values
Enter id: 1
read from local
first Name: Devendra

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

Code:

1. Click on window menu and click on projects, files and services to open it.
2. Right click on java DB and then click on Start server to the server.
3.Now expand Java DB and right click on sample and then click on correct 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. (As per your wish)
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.
(As per your wish tabe name can be given)
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.
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 entered 5 or 7 records.
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 methods to converts it. But this is easiest method.
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.
24. Replace the content of body tag with following code.

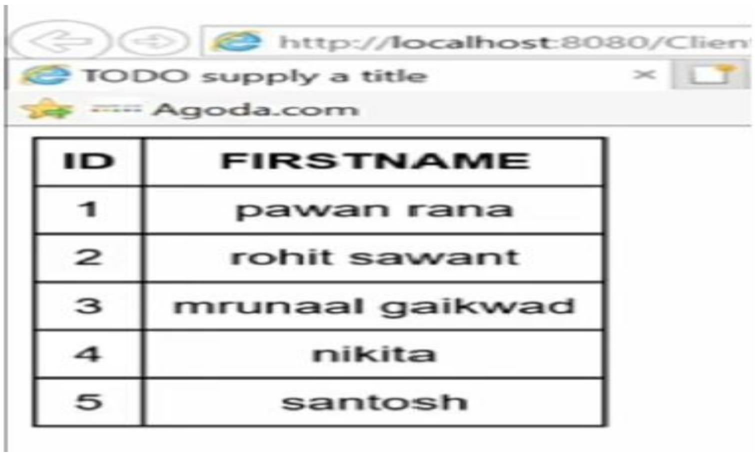
```
<table id=”myTable”>
<tr> <th> ID </th>
<th> NAME </th> </tr>
</table>
```

25. Now run the Client web application.
A window will open in browser which represents a data in tabular format.
These data are the records entered in FRIEND table.

Index.html

```
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
table {
font-family: arial, sans-serif;
border-collapse: collapse; }
td, th {
border: 1px solid #000000;
text-align: center;
padding: 8px; }
</style> <script>
var request = new XMLHttpRequest();
request.open('GET', 'http://localhost:8080/Server/webresources/com.kk.friends/', true);
request.onload = function () {
// begin accessing JSON data here
var data = JSON.parse(this.response);
for (var i = 0; i < data.length; i++) {
var table = document.getElementById("myTable");
var row = table.insertRow();
var cell1 = row.insertCell(0);
var cell2 = row.insertCell(1);
cell1.innerHTML = data[i].id;
cell2.innerHTML = data[i].firstname; } };
Request.send();
</script> </head>
<body>
<table id=”myTable”>
<tr> <th> ID </th>
<th> NAME </th> </tr>
</table> </body> </html>
```

Output:

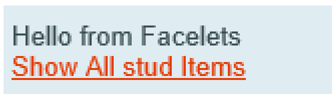


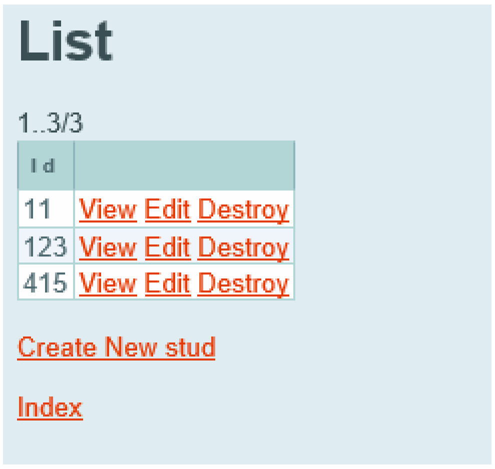
ID	FIRSTNAME
1	pawan rana
2	rohit sawant
3	mrunaal gaikwad
4	nikita
5	santosh

Practical 7

Aim: Define a Restful web service that accepts the details to be stored in the database and perform CRUD operations

Step1: File -> New Project -> Java Web -> Web Application -> CRUD Operation -> Next
Right click -> CRUD Operation -> New -> Entity Class(Not found then other -> Persistance)-> New Entitiy class -> Student package Name com.kk -> Next -> Finish
Step2: Right click on project -> New -> JSF pages from entitiy class (not found -> other -> persisitance -> select JSP pages)
Step3: Select com.kk student -> add -> Next -> bean package = com.kk.a JSF class package = com.kk.b
JSF pages Folder C -> Next -> Finish
Step4: Right click on project -> new -> Restful web service from entity class -> com.kk.student -> add -> resource package com.kk.service -> finish
Step:5 open student.java -> After private long ID press enter -> R/C -> Insert Code -> Add property name -> first ID & type string OR
Step6: get & set first add
R/C on CRUD operation -> clean and build -> deploy -> Run
Output:


Hello from Facelets
Show All stud Items


List

1..3/3

Id	
11	View Edit Destroy
123	View Edit Destroy
415	View Edit Destroy

Create New stud

Index

Practical 8

Aim: We will be making a web based arithmetic calculator using Visual Studio 2019 & WCF (Windows Communication Foundation).

Requirement: -

Visual Studio 2019 (WCF module should be installed)
Browser (chrome, edge, brave)

Steps:

1 - Open Visual Studio > *Create a new project*
2 - Select WCF Service Application and press Next
3 - Name your project and select .Net Framework 4 (as show on picture below) and click Create.
4- Once in the project delete the 2 files (shown in picture)
Ps: They are example files so we don't need them.
5-Now right click on MyService2021 > Add > New Item
6- Select WCF Service in Templates. Give it name as arithmeticOperation and click on Add button.
7- Open both files arithmeticOperation.svc and IarithmaticOperation.cs
8- In arithmeticOpration.svc replace Public void DoWork() by code below

```
        public double Sum(double a, double b)      {
      double result = a + b;
      return result;}
    public double Mul(double a, double b)    {
      double result = a * b;
      return result;     }
    public double Sub(double a, double b)   {
      double result = a - b;
      return result; }
    public double Div(double a, double b)       {
      double result = a / b;
      return result;     }
```

9- in IarithmaticOperation.cs replace [OperationContract] void DoWork() with this code

```
           [OperationContract]
    double Sum(double a, double b);
    [OperationContract]
    double Mul(double a, double b);
    [OperationContract]
    double Sub(double a, double b);
    [OperationContract]
    double Div(double a, double b);
```

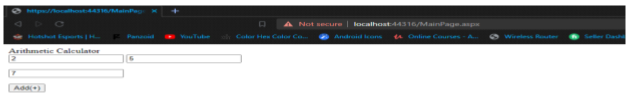
10- Set arithmeticOperation.svc as Start Page
11- Run the Program and copy your localhost url
12- Now right click on solution explorer and add a new project to create client.
13- **I**n Project Type select Web and in Templates select ASP.Net Web Application. Give it name as ClientSide and then click on Next button
14- Right click on Service ClientSide in Solution Explorer and select Add Service Reference.
15- Paste the Copied localhost URL (step 8) in Address and click Discover then press OK to finish setup
16- Now right click on ClientSide > Add > New Item
17- Select Web Form name it MainPage

18- Now switch to Design mode
19- Make a form using tool bar with 3 inputs and 1 button(we will add more buttons)
20- Change the name of Button by clicking on it and changing text in font properties
21- Double click on button to code its functions
22- Write the code on the button_click()

```
ServiceReference1.arithmaticOperationClient ob = new ServiceReference1.arithmaticOperationClient();
double num1 = double.Parse(TextBox1.Text);
double num2 = double.Parse(TextBox2.Text);
TextBox3.Text = Convert.ToString(ob.Sum(num1, num2));
```

23- Now right click on MainPage.aspx in ClientSide set it as start page.
24- Now click on your ClientSide project and set it as setup as startup project.
25- Now Run it.

Output:



Arithmetic Calculator

2 5

7

Add(+)

Practical 9

Aim: Use WCF to create a basic ASP.NET Asynchronous JavaScript and XML (AJAX) service.

Steps :

1) Open Visual Studio □ File □ New □ Project □ ASP.NET Web Application
2) Add a default WebForm.aspx and rename it.
3) Project Name □ Add □ New Item □ AJAX Enabled Service
4) Open AJAX File add the following code by replacing DoWork() Method

```
public double Sum(double a, double b) {
double result = a + b;
return result; }
```

5) Download jquery then in..
 Project Name □ Add □ Existing Item □ Add the Library for JQuery in Project
6) Open MyPage.aspx Add the Following Code

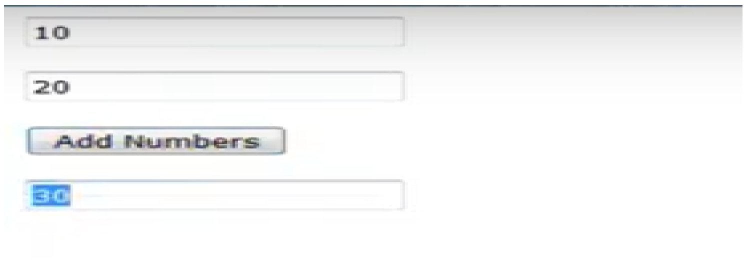
```
＜input id="txt1" type="text" /＞＜br /＞＜br /＞
＜input id="txt2" type="text" /＞＜br /＞＜br /＞
＜input id="btn" type="button" value="Add Numbers" /＞＜br /＞＜br /＞
＜input id="txt3" type="text" /＞
```

6) Add the following code in Default.aspx in ＜head＞ tag

```
＜script type="text/javascript" src="jquery-1.2.js"＞＜/script＞
＜script type="text/javascript" ＞
$(document).ready(function (){
$("#btn").click(function (){
var num1 = $("#txt1").val();
var num2 = $("#txt2").val();
$.ajax({
url: "MyAJAXService.svc/Sum",
type: "POST",
contentType: "application/json; charset=utf-8",
data:JSON.stringify({a: num1, b: num2}),
dataType: "json",
success : function(data){ $("#txt3").val(data.d); },
error : function(err){
alert(err); } }); }); }); ＜/script＞
```

Rename the service file with ur service file name ...
7) Build & run the Project

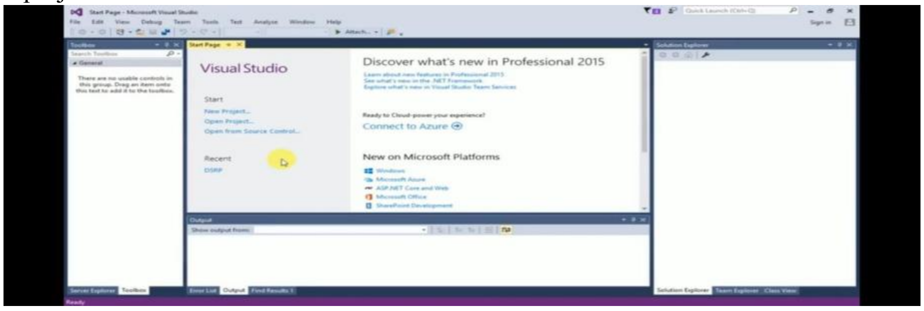
Output:


10
20
Add Numbers
30

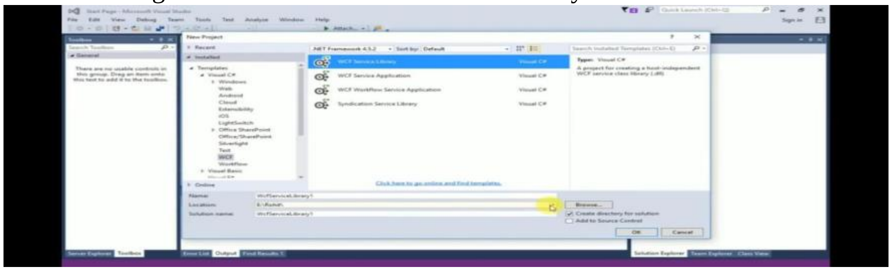
Practical 10

Aim: DEMONSTRATES USING THE BINDING ATTRIBUTES OF AN ENDPOINT ELEMENT IN WCF.

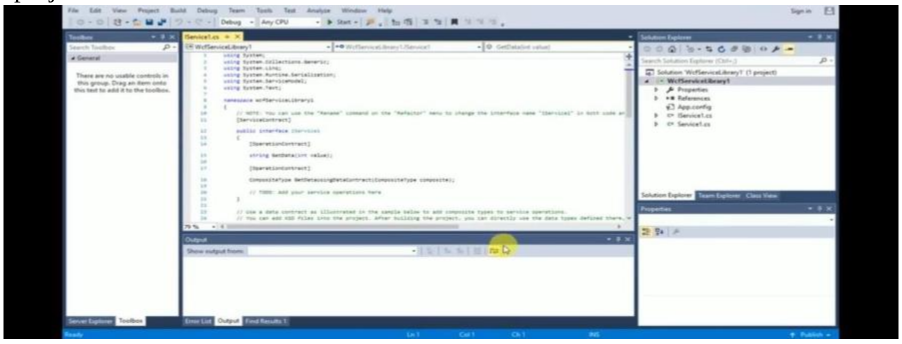
STEP 1: New project click on "Ok"



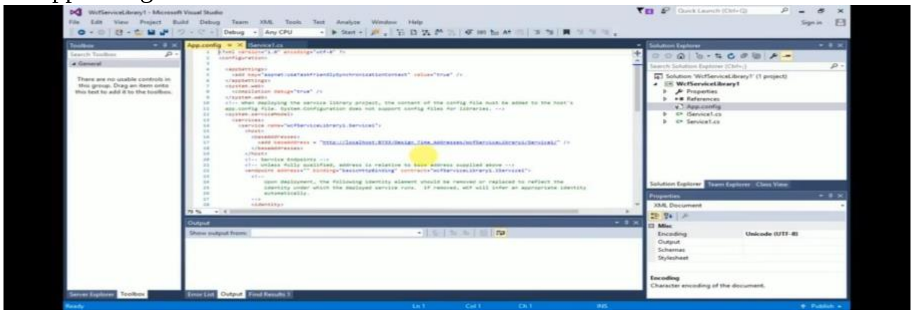
STEP 2: Select WCF in categories and click on WCF service library.



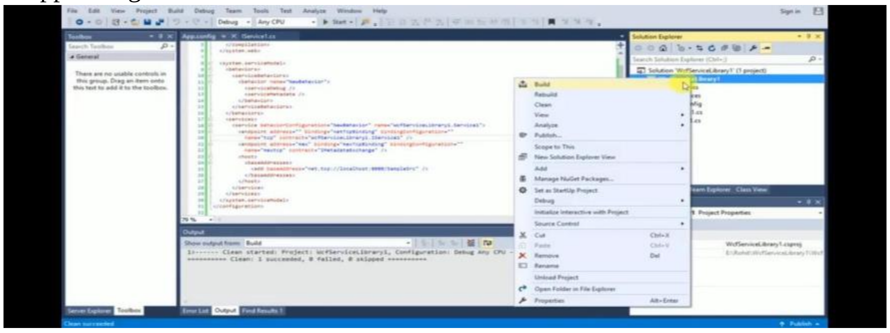
STEP 3: New project created.



STEP 4: Select App.config and double click on it. Click on clean.

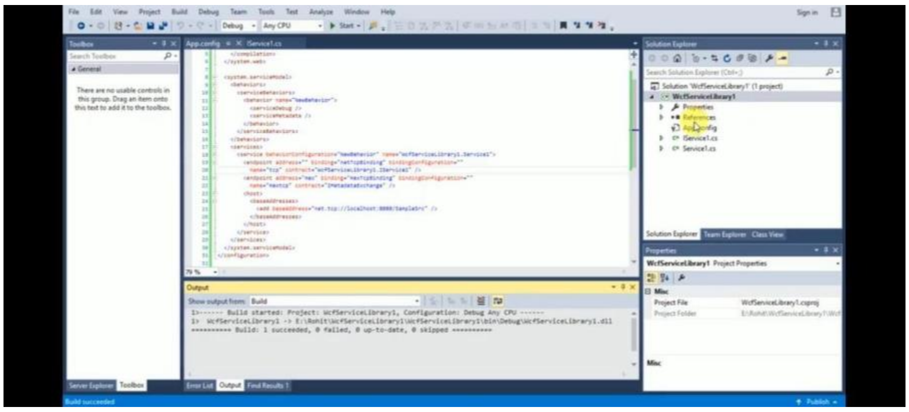


STEP 5: Select App.config and click on build.



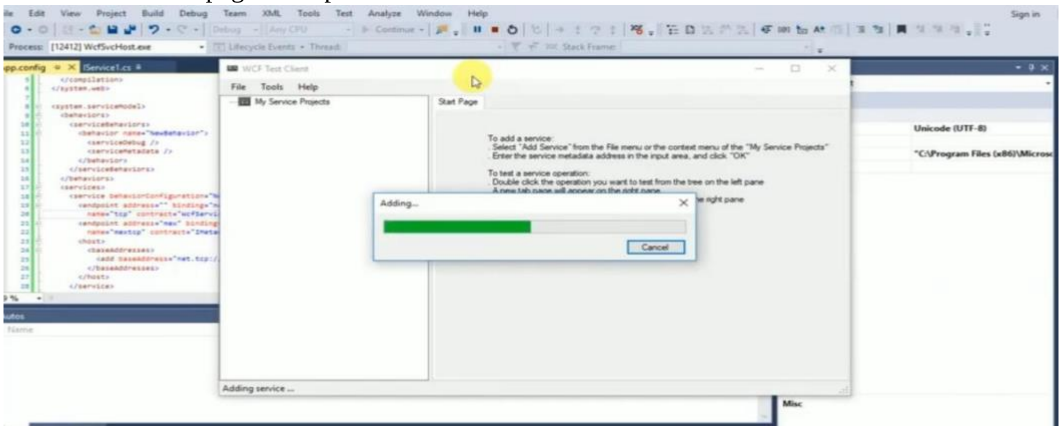
```
1>------ Clean started: Project: WcfServiceLibrary1, Configuration: Debug Any CPU ------
========== Clean: 1 succeeded, 0 failed, 0 skipped ==========
```

STEP 6: After successfully building. Click on start. And run the code.



```
1>------ Build started: Project: WcfServiceLibrary1, Configuration: Debug Any CPU ------
1>  WcfServiceLibrary1 -> E:\Rohit\WcfServiceLibrary1\WcfServiceLibrary1\bin\Debug\WcfServiceLibrary1.dll
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

STEP 7: WCF test client page will open.



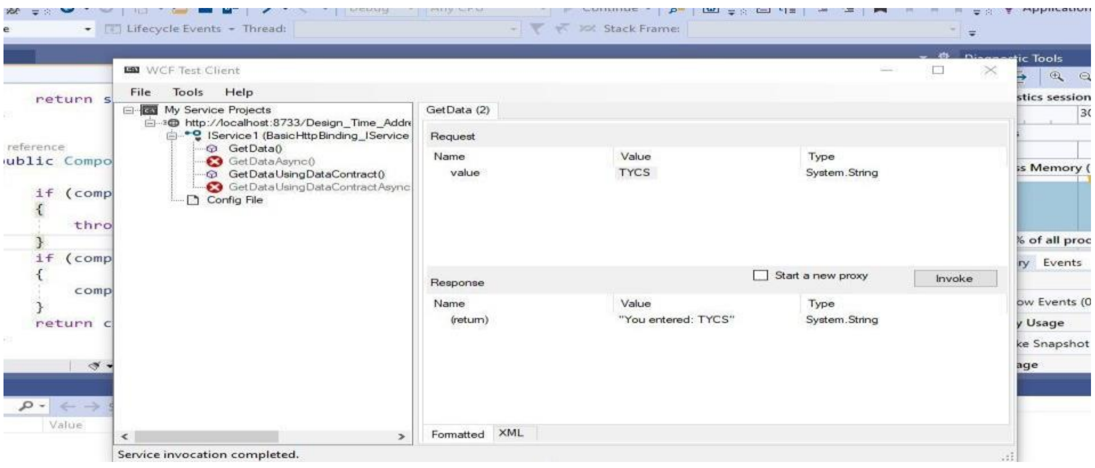
To add a service:
 Select "Add Service" from the File menu or the context menu of the "My Service Projects"
 Enter the service metadata address in the input area, and click "OK"

To test a service operation:
 Double click the operation you want to test from the tree on the left pane
 A new tab page will appear on the right pane

Adding...

Adding service ...

STEP 8: Click on get data and add the value (TYCS).



WCF Test Client

My Service Projects
- http://localhost:8733/Design_Time_Addr
  - IService1 (BasicHttpBinding_IService
    - GetData()
    - GetDataAsync()
    - GetDataUsingDataContract()
    - GetDataUsingDataContractAsync
  - Config File

GetData (2)

Request

Name	Value	Type
value	TYCS	System.String

Response

Name	Value	Type
(return)	"You entered: TYCS"	System.String

Service invocation completed.

Response will be “You entered = TYCS”