PRACTICAL 1

Step 1:-

Go to File -> New Project. Select Java Web in categories and Web Application

in Projects. Click on Next to create a web based project.

Step 2:-

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

click Finish.--> tempwebservice(name)

Step 3:-

Create a web service.

Right click on Project -> New -> Web Service

Step 4:-

Enter a Web Service Name->(tempwebservice) and package name(mypack) and then click on Finish to

create a Web Service.

Step 5:-

go to design and add operation:

Step 6:-

Give Operation name F_to_C and return type as Double. After that click on Add button to give

parameters for method. Give its name as f and type as Double and then click

on the OK button. Your one operation is now successfully created.

Step 7:-

Give Operation name C_to_F and return type as Double. After that click on Add button to give

parameters for method. Give its name as f and type as Double and then click

on the OK button. Your one operation is now successfully created.

Step:- 8

Now go to source mode by clicking on Source.

Write the code in webmethod F_to_C to convert Fahrenheit to

Celsius.

Double c = (f-32)*1.8;

return c;

Write the code in webmethod C_to_F to convert Celsius to

Fahrenheit and then press Ctrl+S to save.

Double f = (c*1.8)+32;

return f;

(write a code in tempservice.java)################*MAIN*###################

---

tempservice.java:-

```
package mypack;

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



@WebService(serviceName = "tempservice")
public class tempservice {



    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
```

```
    @WebMethod(operationName = "F_to_C")
    public Double F_to_C(@WebParam(name = "val") double val) {
        return (val-32)*5/9;

    @WebMethod(operationName = "C_to_F")
    public Double C_to_F(@WebParam(name = "val") double val) {
        return (val*9/5)+32;
    }
}
```

Step 9:-

Now right click on project name and click on Deploy to deploy your project.(& Deploy it)

Step 10:-

To test your web service right click on tempwebservice (Test web service of tempwebservice)

Step 11:-

Following window will open in the 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

Step 12:-

Selected value is in celsius of 100.

Step 13:- Similarly second textbox and button will convert the numeric value into Fahrenheit.

Step 14:-

Now to consume this web service we are creating a client.(create new web application)

Step 15:-

create a new web application project with the name Temp_client.

Step 16:-

Now open the index.html page of Temp_client and write the following code into that.(in index.html)

---

index.html:

```
<html>
<head>
<title>todo Title</title>
<meta charset="UTF-8>
<meta name="viewport" content=with=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"><br>

</form>


</body>
</html>
```

Step 17:

Now create two jsp pages for both submit buttons. Right click on Web Pages -> New -> JSP

Step 18:-

Enter file name f_to_c and then click on Finish.

Step 19:-

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

Step 20:-

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

Step 21:-

Click on Browse.

Step 22:-

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

Step 23:-

Click on Finish.

Step 24:-

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

Step 25:- Now right click in between the body section and select Call Web

Service Operation as below.

Step 26:-

New window will appear, select the C_to_F by expanding it and click on OK.

So the selected code in the second pic will be automatically generated.

---

c_to_f.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>
      <%
         String d = request.getParameter("data");
         Integer dd = Integer.parseInt(d);
         try{
         mypack.Tempservice_Service  service = new mypack.Tempservice_Service();
         mypack.Tempservice Port = service.getTempservicePort();
         double c = dd;

         java.lang.Double result = port.cToF(c);
         out.println("Result = "+result);
         }catch(Exception ex){

         }
         %>
   </body>
</html>
```

Step 27:-

Now, make the selected area in step 30 as like selected area in below pic

by adding some lines of code.

Step 28:-

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.

---

f_to_c.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>
       <%
          String d = request.getParameter("data");
          Integer dd = Integer.parseInt(d);
          try{
          mypack.Tempservice_Service  service = new mypack.Tempservice_Service();
          mypack.Tempservice Port = service.getTempservicePort();
          double f = dd;
          java.lang.Double result = port.fToC(f);
          out.println("Result = "+result);
          }catch(Exception ex){
```

```
        }
        %>
    </body>
</html>
```

Step 29:-

Now run the Temp_client project. The window will be open like below.

Step 30:- Now you can to 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.

Practical 2

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

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

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

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

Step 4:-
Now we are going to create a table in the default database sample.
Right click on Table -> Create Table

Step 5:-
Give table names as FRIENDS.

Step 6:-
Now click on the Add column button to add columns in the table.
Enter details as in the pic below and select Primary key. After that click on OK
button.

Step 7:-
Now add a second column with the following detail. But don’t select
primary and click on the OK button.

Step 8:-

Now click on the OK button.

Step 9:-

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

Step 10:-

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

Step 11:-

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

Step 12:-

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

After that click on the OK button to finish.

Step 13:-

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

Step 14:-

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

Step 15:-

Choose Data Source jdbc/sample.

Step 16:-

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

Step 17:-

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

Step 18:-

Now open the selected file by double clicking on it.

Step 19:-

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 the easiest method.

Step 20:-

After that, right click on the project name and Deploy it.

Step 21 :- Now create one more Web Application as a Client. After that click

on Next and then Finish button.

Step 22:-

Create a Web Application with the name ServiceClient.

Step 23:-

Now create a RESTful Java Client.

Right click on ServiceClient -> New -> Other.

Step 24:-

Drag down and select Web Services and in side panel select RESTful Java

Client.

Step 25:-

After selecting RESTful Java Client click on Next.

Step 26:-

Enter the following data. Class Name -> Data Package -> com.ss

Step 27:- Now click on Browse button and select the option in the below pic. After select click on OK button.

Step 28:-

Click on Finish.

Step 29:- Now create a JSP page.

Right click on ServiceClient -> New -> Other

Step 30:-

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

Step 31:-

Enter File Name Del_data and click on Finish button.

Step 32:-

Now create one more JSP file by following step number 30, 31 & 32. But File

Name will be getData.

Step 33:- Now create a Java class.

Right click on ServiceClient -> New -> Other

Step 34:-

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

Step 35:-

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

Step 36:-

Your project file structure will look like below.

Step 37:-

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

---

index.html :=

```
<html>
<head>
<title> todo supply </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>
<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>
</body>
</html>
```

Step 38:-

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.

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.");
}}
```

Step 39:-

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

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 d = request.getParameter("ID");
      DeleteData.deldata(id);
      %>
  </body>
</html>
```

Step 40:-

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

Del_data.jsp

getData.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>

<style>
table{
font-family:arial,sans-serif;
border-collapse:collapse;
}
td,th{
border:1px solid #000000;
text-align:center;
padding:0px;
}

</style>

<script>
var request = new XMLHttpRequest();
request.open('GET','http://localhost:8080/Server/webresources/com.kk.friends/',true);
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>
```

Step 41:-

Now Run the ServiceClient project.

Step 42:-

On running the project the following window will open in the browser.

PRACTICAL 3

Step 1:-

First create a new project→ Web project

Step 2:-

Give the project name as CCipher

Step 3:-

Right click on project→ New→ Web Service

Step 4:-

Give the web service name as CCipherService and package as com.abc.

Step 5:-

Open CCipherService.java file and type the following code

CCipherService.java

```
package com.abc;

import java.util.Random;
import javax.jws.WebService;
impart javax.jws.WebMethod;
import java.jws.WebParam;
```

```
@WebService(serviceName = "CCipherService")
public class CCipherService{
@WebMethod(operationName="encrypt")
public String encrypt(@WebParam(name="name") String msg){
String ct="";
Random rd=new Random();

for(int i=O; i<msg.length(); i++) {
ct+=(char) (msg.charAt(i)+rd.nextInt(26));
}
return ct;
}
}
```

Step 6:-

Now clean and build the project

Step 7:-

Now deploy the project

Step 8:-

Right Click on CCipherService in Web Services →Click on Test Web Service

Step 9:-

Test the web service in the browser.

Step 10:-

Create a java application. Its purpose would be to create the java client for the above web service.

Step 11:-

Name the project as CCipherClient.

Step 12:-

Now, Right Click on the Client application→ New→ Web Service Client

Step 13:-

Paste the WSDL URL of the web service which was previously deployed and tested.

Step 14:-

Now, drag and drop the encrypt method from Web Service References into the CcipherClient.java

Step 15:-

Add the following code along with the encrypt method code.

CCipherClient.java :

```
package ccipherclient;

import java.util.Scanner;

public class CCiperClient{
 public static void main(String[] args){
Scanner sc = new Scanner(System.in);
System.out.println("Enter message");
String msg=sc.next();
System.out.println("Encrypted message: "+encrypt(msg));
}

private static String encrypt(java.lang.String name){
```

```
com.abc.CCipherService_Service = new com.abc.CCipherService_Service();
com.abc.CCipherService port = service.getCCipherServicePort();
return port.encrypt(name);
}}
```

Step 16:-

Now clean and build the CCipherClient

Step 17:-

Run the project

Step 18:-

PRACTICAL 4

Aim: Develop a client which consumes web services developed in different platforms. Create a webservice in netbean and use it in .NET 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.

Step 1:-

Create a web application in Netbeans

Step 2:-

Name the project as FactorialWebService → click on Next → click on Finish.

Step 3:-

Right Click on FactorialWebService project->New->Web Service

Step 4:- Name the web service as Factorialwebservice → give package name as abc.com

→click on Finish.

Step 5:-

Go to design page → Add operation

Give method name as factorial → Return type double → Parameter num → Type double

Step 6:-

Go to source page and Now type the code below in Factorialwebservice.java file

---

Factorialwebservice.java:

```
package abc.com;

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

@WebService(serviceName = "Factorialwebservice")
public class Factorialwebservice{
@WebMethod (operationName = "factorial")
public double factorial(@WebParam(name = "num") double num){
double f=1;
int I;
for(i=1;i<=num;i++){
f = f*I;
}
return f;
}
}
```

Step 7:-

Right click on project name FactorialWebService->clean and build

Step 8:-

Right click on FactorialWebService → test web service

Step 9:-

Test the web service in the browser.

Step 10:-

Copy the WSDL Link of the web service.Steps to create the Client app in .NET

Step 11:-

Open Visual Studio IDE → file → new project

Create a New Project in C# Empty Web Application

Step 12:-

Give the project name as WebService_Practical4_Client

Step 13:-

Right click on project → select Add → select New Item

Step 14:-

Add a new Empty web form.

Step 15:-

Design the interface and give the appropriate name and id's

Step 16:-

Right click on project name FactorialWebService->clean and build

Step 17:-

Test the WebService

Step 18:-

Copy the WSDL Link of the web service.

Step 19:-

Now RightClick on Project name and add (Service Reference in Client app)

Step 20:-

Paste the WSDL url in the address.

Step 21:-

Now double click the button and then type the following cod in the aspx.cs f

---

WebService_Practical4_Client:-

space WebService_Practical4_Client

public partial class webServicePractical4_Client : System Web.UI.Page{

protected void Page_Load(object sender,EventArgs e){

}

protected void Button1_Click(object sender, EventArgs e){

ServiceReference1.FactorialwebserviceClient client = new
ServiceReference1.FactorialwebserviceClient();

Label1.Text = "Factorial is: "+ client.factorial(Convert.ToDouble(TextBox1.Text)).ToString();

}}

Step 22:-

PRACTICAL 5

Step 1:-

Create a web application in Netbeans

Step 2:-

Name the project as FactorialWebService → click on Next → click on Finish.

Step 3:-

Right click on project name calcwebservice –new –other –web service

Step 4:-

Name the web service as calcservice → give package name as mypack

Step 5:-

Go to Design page → add operation

Give name as Addition → add two parameters a and b with type integer → click on ok.

Step 6:-

Add another operation.

Give name as Subtraction → add two parameters a and b with type integer → click

on ok.

Step 7:-

Add the following code for addition and subtraction.

---

calcservice.java:

```
package mypack;
```

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

@WebService(serviceName = "calcservice")
public class calcservice{

@WebMethod(operationName = "Addition")
public Integer Additon(@WebParam(name = "a") int a , @WebParam(name = "b") int b){
return a+b;
}

@WebMethod(operationName = "Subtraction")
public Integer Subtraction(@WebParam(name = "a") int a , @WebParam(name = "b") int b){
return a-b;
}
```

Step 8:-

Right click on project name callwebservice → deploy

Step 9:- Right click on calservice → test web service

Step 10:-

Test the web service in the browser.

Step 11:-

Copy the URL as shown below

Step 12:-

Go to file → new → project → web application.

Step 13:-

Give name as calcclient.

Step 14:-

Right click on calcclient → new → web service client

Step 15:-

Paste the URL copied above.

Step 16:-

Right click on calcclient → new → jsp

Step 17:-

Give name as index.

Step 18:-

Place your cursor below hello world or line 16 as shown in figure below

Now right click below hello world

Go to web service client resource → select call web service operation

Step 19:-

Select addition operation to invoke.

Step 20:-

Write code in index.html

---

index.html:

<html>

```
<head>
<title>todo Title</title>
<meta charset="UTF-8>
<meta name="viewport" content=with=device-width,initial-scale=1.0">
</head>
<body>

<form action="index.jsp">
Enter n1<input type="text" name="n1">
Enter n2<input type="text" name="n2">
<input type="submit" value="submit">
</form>



</body>
</html>
```

Step 21:-

Write code in index.jsp

---

index.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>
            <h1> hello world!</h1>


        <%
int x = Integer.parseInt(request.getParameter("n1"));
int y = Integer.parseInt(request.getParameter("n2"));
try{
mypack.Calcservice_Service  service = new mypack.Calcservice_Service();
mypack.Calcservice port = service.getCalcservicePort();
int a = x;
int b = y;
java.lang.Integer result = port.addition(a,b);
out.println("Result = "+result);
}catch(Exception ex){

}
 %>
</body>
</html>
```

Step 22:-

Repeat the same steps for subtract place cursor below hello world → select subtract

Step 23:-

Add code in index .jsp

---

index.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>
         <h1> hello world!</h1>


 <%
int m = Integer.parseInt(request.getParameter("n1"));
int n = Integer.parseInt(request.getParameter("n2"));
try{
mypack.Calcservice_Service  service = new mypack.Calcservice_Service();
mypack.Calcservice port = service.getCalcservicePort();
int a = m;
int b = n;
java.lang.Integer result = port.Substraction(a,b);
out.println("Result = "+result);
}catch(Exception ex){

}
 %>
</body>
```

</html>

Step 24:-

Right click on calcclient project → deploy Run index.html

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.

Step 1:-

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

Step 2:-

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

Step 3:-

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

Step 4:-

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

Right click on Table -> Create Table

Step 5:-

Give table names as FRIENDS.

Step 6:-

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

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

Step 7:-

Now add a second column with the following detail. But don’t select primary and click on the OK button.

Step 8:-

Now click on the OK button.

Step 9:-

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

Step 10:-

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

Step 11:-

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

Step 12:-

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

After that click on the OK button to finish.

Step 13:-

As you can see, I have entered 7 records.

Step 14:-

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

Step 15:-

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

Step 16:-

Choose Data Source jdbc/sample.

Step 17:-

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

Step 18:-

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

Step 19:-

Now open the selected file by double clicking on it.

Step 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 the easiest method.

Step 21:-

After that, right click on the project name and Deploy it.

Step 22:-

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

Step 23:-

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

Step 24:-

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

---

index.html:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8>
          <meta name="viewport" content=with=device-width,initial-scale=1.0">
      <title>JSP Page</title>

<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(){
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>
```

Step 25:-

Now run the Client Web Application.

Output:-

Practical 7

Aim:-Define a RESTful web service that accepts the details to be stored in a database and performs CRUD operation.

Step 1:-

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

Step 2.

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

Step 3:-

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

Step 4:-

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

Step 5:-

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

Step 6:-

A window will appear like the below pic. Enter the following data and click on Next.

Class Name -> seller

Package Name -> com.kk

Step 7:-

Click on Finish.

Step 8:-

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

Step 9:-

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

Step 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.

Step 11:-

Now click on Finish.

Step 12:-

Right click on Project Name □ New□ RESTful Web Services from Entity Classes

Step 13:-

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

Step 14:-

Now open seller.java file under com.kk package

Step 15:-

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

Step 16:-

A new list will appear. Click on Add Property.

Step 17:-

A new window will open. Enter name as firstName.Then click on OK button. Actually we are setting the getter and setter method for firstName.

Step 18:-

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

Step 19:-

Now right click on the project name and run it.

Step 20:-

A window will open in the browser like below.

Step 21:-

Now click on Show All sellers Items for CRUD operation.

Step 22:-

You can add more data by clicking on Create New seller and can view, edit and

delete by clicking on View, Edit and Destroy option.

Step 23:-

Adding one more data into the 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.

Step 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.

Practical 8

Aim:-Implement a typical service and a typical client using WCF.

Step 1:-

Open Microsoft visual studio → select create a new project

Step 2:-

Select WCF Service Application→click on Next.

Step 3:-

Give name as MathService1 → click on Create.

Step 4:-

Delete IService1.cs and Service.svc from the right side(from solution explorer).

Step 5:-

Right click on MathService1(from solution explorer)-->add→new item→select WCF service → give name as MathService → click on Add.

Step 6:-

Write code for MathService.svc.cs and MathService.cs

MathService.svx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
```

```
using System.ServiceModel;
using System.Text;

namespace MathService1
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "MathService" in code, svc and config file together.
    // NOTE: In order to launch WCF Test Client for testing this service, please select MathService.svc or MathService.svc.cs at the Solution Explorer and start debugging.
    public class MathService : IMathService
    {
        public Int32 add(Int32 n1, Int32 n2)
        {
            return add(n1, n2);
        }

        public Int32 subtract(Int32 n1, Int32 n2)
        {
            return subtract(n1, n2);
        }

        public Int32 multiply(Int32 n1, Int32 n2)
        {
            return multiply(n1, n2);
        }

        public Int32 divide(Int32 n1, Int32 n2)
        {
            return divide(n1, n2);
        }
    }
}
```

IMathService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace MathService1
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IMathService" in both code and config file together.
    [ServiceContract]
    public interface IMathService
    {
        [OperationContract]
        Int32 add(Int32 n1 , Int32 n2);

        [OperationContract]
        Int32 subtract(Int32 n1, Int32 n2);

        [OperationContract]
        Int32 multiply(Int32 n1, Int32 n2);

        [OperationContract]
        Int32 divide(Int32 n1, Int32 n2);
    }
```

}

Step 7:-

Test the service (click run).

Step 8:-

Right click on Solution ‘MathService1’.

Step 9:-

click add →new project.

Step 10:-

Select windows form application

Step 11:-

Give name as MathServiceTestApp1.

Step 12:-

create design on form1.

Step 13:-

Right click on reference as shown below →click add service reference.

Step 14:-

click on discover button →select IMathService.

Step 15:-

Give namespace as given in picture.

Step 16:-

Double click on the calculate button and write code.

---

form1.cs:

```
using MathServiceTestApp1.MathService1;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MathServiceTestApp1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
```

```
        private void Form1_Load(object sender, EventArgs e)
        {
            MathServiceClient loclient = new MathServiceClient();
            Int32 num1 = Convert.ToInt32(textBox1.Text.Trim());
            Int32 num2 = Convert.ToInt32(textBox2.Text.Trim());
            if (comboBox1.Text == "add")
            {
                textBox3.Text = loclient.add(num1, num2).ToString();
            }
            else if (comboBox1.Text == "subtract")
            {
                textBox3.Text = loclient.subtract(num1, num2).ToString();
            }
            else if (comboBox1.Text == "multiply")
            {
                textBox3.Text = loclient.multiply(num1, num2).ToString();
            }
            else
            {
                textBox3.Text = loclient.divide(num1, num2).ToString();
            }

        }
    }
}
```

Step 17:-

Right click on MathServiceTestApp1---click set as startup project.

Step 18:-

Now run the project.

Practical 9

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

Step 1:-

Open Microsoft visual studio → select create a new project

Step 2:-

Select ASP.NET Web Application(.NET Framework) to create a website → click on Next.

Step 3:-

Give the project name 'Website3' → click on create.

Step 4:-

Select Empty to create a empty website → click on create.

Step 5:-

Go to Solution Explorer → right-click → Select Add → New Item → Select WebForm

→ Default.aspx page will open.

Step 6:-

Go to Solution Explorer → right-click →Select Add → New Item → Select AJAX Enabled WCF Service → Give name as GreetingService

Step 7:-

Open GreetingService.cs and define the function Greeting as shown below.

GreetingService.cs :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Activation;
using System.ServiceModel.Web;
using System.Text;

namespace Website3
{
    [ServiceContract(Namespace = "")]
    [AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
    public class GreetingService
    {
        // To use HTTP GET, add [WebGet] attribute. (Default ResponseFormat is
WebMessageFormat.Json)
        // To create an operation that returns XML,
        //     add [WebGet(ResponseFormat=WebMessageFormat.Xml)],
        //     and include the following line in the operation body:
        //         WebOperationContext.Current.OutgoingResponse.ContentType = "text/xml";
        [OperationContract]
        public void DoWork()
        {
            // Add your operation implementation here
            return;
        }
        [OperationContract]
        public string Greeting()
        {
            return "hello";
```

```
        }

        // Add more operations here and mark them with [OperationContract]
    }
}
```

Step 8:-

Open WebForm1.aspx page and click in [Design option].

Step 9:-

From Toolbox, Drag and drop 1)Scriptmanager Control 2)Button

Step 10:-

Set value property of button as greet from properties window

Step 11:-

In the Design page right-click on Scriptmanager Control →Select Properties

option→ select Services.

Step 12:-

Set path → write GreetingService.

Step 13:-

Go to source option → write the following code.

Default.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Website3.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>


      <asp:ScriptManager ID="ScriptManager1" runat="server">
         <Services>
            <asp:ServiceReference Path="~/GreetingService.svc.cs"
         </Services>
      </asp:ScriptManager>
         <input type="button" value="greet" onclick="showgreeting()" />
       </div>
   </form>
</body>

</html>

<script language="javascript" type="text/javascript">
   function showgreeting() {
      GreetingService.Greeting(onSuccess, onError);
```

```
    }

    function onSuccess(response) {
        alert(response);
    }
    function onError(error) {
        alert("an error occurred" + error.get_message());
    }

</script>
```

Step 14:-

Run by right clicking on WebForm1.aspx in solution explorer → select view in browser.

Practical 10

Aim:-Demonstrates using the binding attribute of an endpoint element in WCF.

Step 1:-

Open Microsoft visual studio → select create a new project

Step 2:-

Select WCF Service Library → click on Next.

Step 3:-

Give the project name 'SampleCalcSvc' → click on create.

Step 4:-

Delete IService1.cs and Service.svc from solution explorer.

Step 5:-

Go to app.config (from solution explorer)and delete the selected code.

Step 6:-

Right click on SampleCalcSvc from solution explorer →add →new item →interface→give name ICalcService.cs

Step 7:-

write code in ICalcService.cs

Project_name : SampleCalcSvc

file_name : ICalcService (add item Interface)

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.ServiceModel;
using System.Text;
using System.Threading.Tasks;

namespace SampleCalcSvc
{
   [ServiceContract]
   internal interface ICalcService
   {
      [OperationContract]
      int GetSum(int a, int b);

   }
}
```

Step 8:-

Right click on SampleCalcSvc from solution explorer →add →new item →class→give name CalcService.cs

Step 9:-

write code CalcService.cs

---

file_name: CalcService(add item class)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SampleCalcSvc
{
   internal class CalcService
   {
      public int GetSum(int a , int b)
      {
         return a + b;
      }
   }
}
```

Step 10:-

Right click on SampleCalcSvc from solution explorer →build

Step 11:-

Right click on App.config from solution explorer →Edit Wcf Configuration

Step 12:-

select create new service select Browse →bin →debug →SampleCalcSvc.dll →SampleCalcSvc.CalcService

Step 13:-

Select Browse.

Step 14:-

Bin →Debug →SampleCalcSvc.dll →SampleCalcSvc. CalcService → Click next

Step 15:-

Select TCP

Step 16:-

Empty the text box--- click Next → click Yes

Step 17:-

Click on Finish

Step 18:-

Give name as SampleCalcSvcNetTcpEndpoint

Step 19:-

Right click on Endpoints→select New Service Endpoints → give name as shown below.

Step 20:-

Select host →new button → enter base address net.tcp://localhost:8080/SampleSvc → ok.

Step 21:-

Click advanced →right click service behaviour →new service behaviour configuration

Step 22:-

Click add button → Select the shown option and Click add

Step 23:-

Select service debug → make the selected option from false to true.

Step 24:-

Click on SampleCalcSvc.CalcService and select the option as shown below.

Step 25:-

Save the file→Save the project and run