

Date: 19/08/2020

Practical no 1

AIM: Write a program to implement to create a simple web service that converts the temperature from Fahrenheit to Celsius and vice versa.

Steps:

1. Create a Web Application of "ASP.NET" using(.Net Framework 4.7).
2. Give a suitable title to the project and solution.
3. Initialize it as "Empty" solution.
4. On Solution Explorer, right click and add a "Web Service(asmx)" to the solution.
5. Moving ahead we already have the files initialized for returning "Hello World".
6. Remove/overwrite the "Hello World" "WebMethod" and add your own "WebMethods" to the source file.
7. Save it and try out the web-service using the play button to host the web service on a local "IIS Express" server.
8. Once the web service successfully runs and gives the desired output in form of XML documents; We will Proceed towards making client side pages.
9. In solution explorer, right click on the connected services tab and click on "add service reference" option.
10. Next discover the web service we just created and click on the service you created and click ok.
11. Now you have successfully connected the web service to the solution.
12. Add a new WebForm in the same solution and start designing the client side UI.
13. After completing the UI design open the backend C# code.
14. Define methods for Button Clicks either explicitly or by double clicking the respective buttons in the design section of the form .
15. In the button click methods first create a SOAP object for the "webservice.WebService1SoapClient()" class.
16. Using the SOAP object invoke the web-service methods and pass the appropriate data from the input field casted to the data type used in web-service method.

17. Next try running the ASPX page using the local server, add exception handling for handling exceptions if required.

Code:**WebService1.asmx.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace Temp
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]

    public class WebService1 : System.Web.Services.WebService
    {
        [WebMethod]
        public double celsius_to_farhenheit(double celsius)
        {
            return ((celsius * 9 / 5) + 32);
        }
        [WebMethod]
        public double farhenheit_to_celsius(double farhenheit)
        {
            return ((farhenheit - 32) * 5 / 9);
        }
    }
}
```

WebForm1.aspx:

[illegible]

WebForm1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Temp
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {

            double result;
            try
            {
                ServiceReference1.WebService1SoapClient client = new
ServiceReference1.WebService1SoapClient();

                result = client.celsius_to_farhenheit(Convert.ToDouble(TextBox1.Text));

                if (DropDownList1.SelectedValue.Equals("Celsius"))
                {
                    Label2.Text = result.ToString();
                    Label4.Text = "&deg;Farhenheit";
                }
                if (DropDownList1.SelectedValue.Equals("Farhenheit"))
                {
                    Label2.Text = "Already in Farhenheit";
                    Label4.Text = "";
                }
            }
            catch (System.FormatException)
            {
                Label2.Text = "Invalid Inputs";
                Label4.Text = "";
            }
        }
    }
}
```

```
protected void Button2_Click(object sender, EventArgs e)
{
    double result;
    try
    {
        ServiceReference1.WebService1SoapClient client = new
ServiceReference1.WebService1SoapClient();
        result = client.fahrenheit_to_celsius(Convert.ToDouble(TextBox1.Text));
        if (DropDownList1.SelectedValue.Equals("Celsius"))
        {
            Label2.Text = "Already in Celsius";
            Label4.Text = "";
        }
        if (DropDownList1.SelectedValue.Equals("Farhenheit"))
        {
            Label2.Text = result.ToString();
            Label4.Text = "&deg;celsius";
        }
    }
    catch (System.FormatException)
    {
        Label2.Text = "Invalid Inputs";
        Label4.Text = "";
    }
}
}
```

Outputs:

WebService1

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

- [celsius to farhenheit](#)
- [farhenheit to celsius](#)

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](#).

WebService1

Click [here](#) for a complete list of operations.

celsius_to_farhenheit

Test

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

Parameter	Value
celsius:	<input type="text"/>

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/celsius_to_farhenheit"

<?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>
    <celsius_to_farhenheit xmlns="http://tempuri.org/">
      <celsius>double</celsius>
    </celsius_to_farhenheit>
  </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/">
```

Input Temperature ° Celsius ▼

Result :

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<double xmlns="http://tempuri.org/">37.4</double>
```

Input Temperature ° Celsius ▼

Result :

Input Temperature ° Celsius ▼

Result : 37.4 °Fahrenheit

Input Temperature ° Fahrenheit ▼

Result : -17.777777777778 °celsius

Date: 26/08/2020

Practical no 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

A)One-Way**1. WebForm.aspx file**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="prac2a.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
      <br />
      <asp:Label ID="lblPagedate" runat="server"> </asp:Label>
      <br />
      <br />
      <asp:Label ID="lblServicedate" runat="server"> </asp:Label>
    </div>
  </form>
</body>
</html>
```

2. WebForm.aspx.cs file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```



```
namespace prac2a
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            //Time show before the service is calling
            lblPagedate.Text = "on the load time the time is " + DateTime.Now.ToString();
            ServiceReference1.Service1Client sc = new ServiceReference1.Service1Client();
            sc.OneWayMessage();
            //after service is calling that time show
            lblServicedate.Text = "After Calling the service the time is " + DateTime.Now.ToString();
        }
    }
}
```

3. IService.cs:-

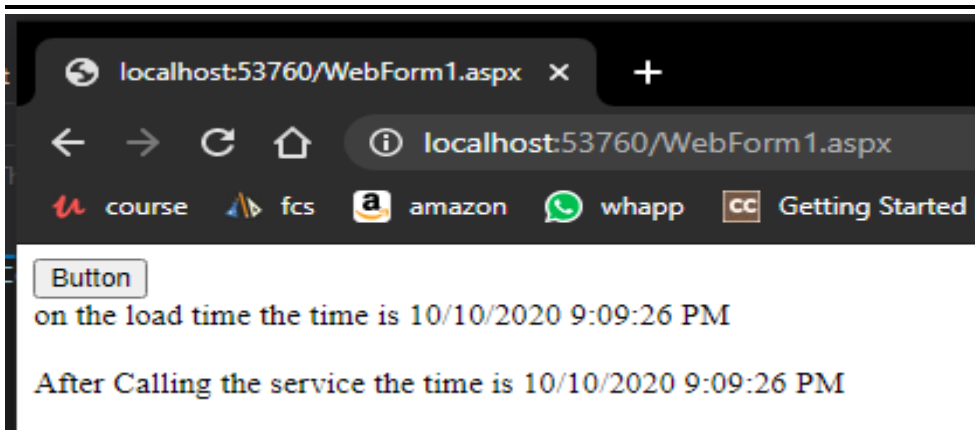
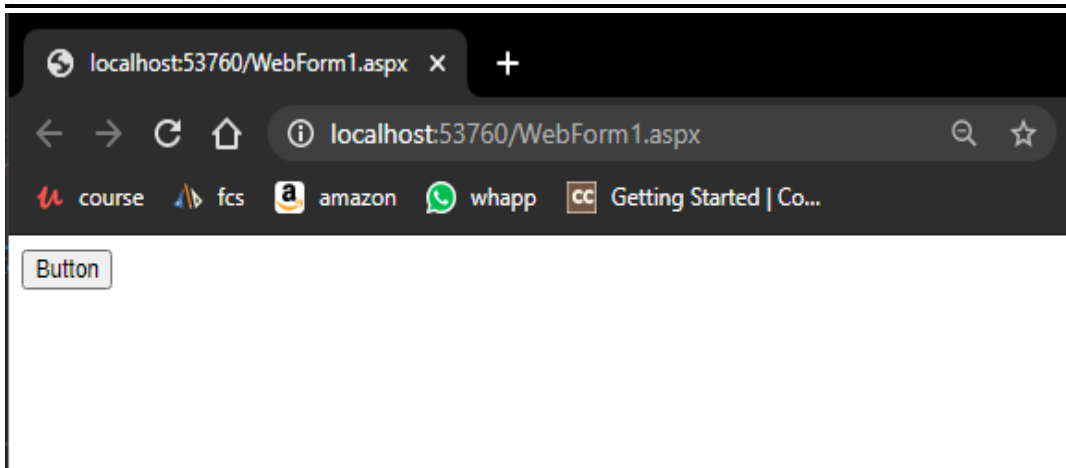
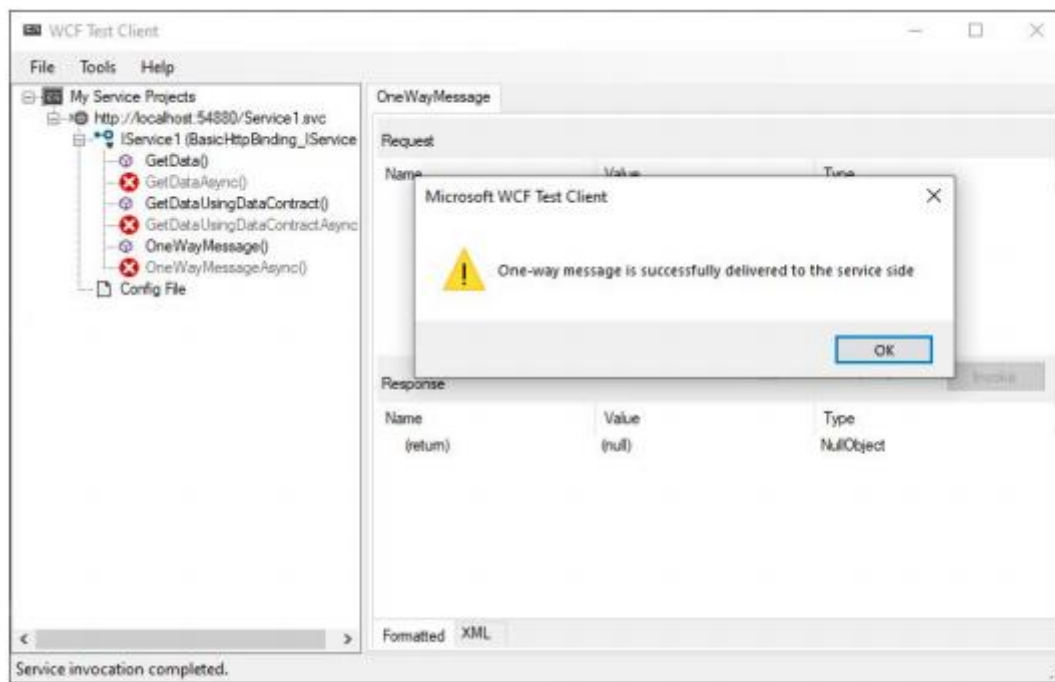
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace prac2a
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract(IsOneWay = true)]
        void OneWayMessage();
    }
}
```

4. Service1.svc.cs

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

namespace prac2a
{
    public class Service1 : IService1
    {
        public void OneWayMessage()
        {
            Thread.Sleep(2000);
        }
    }
}
```

Output:-

B) Request –Response**1. WebForm.aspx file**

```
<% @ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WcfService2.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
            <br />
            <asp:Label ID="lblPagedate" runat="server"></asp:Label>
            <br />
            <br />
            <asp:Label ID="lblServicedate" runat="server"></asp:Label>

        </div>
    </form>
</body>
</html>
```

2. WebForm.aspx.cs file

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WcfService2
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void Button1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

```
{
    lblPagedate.Text = "Page date is " + DateTime.Now.ToString();
    ServiceReference1.Service1Client sc = new ServiceReference1.Service1Client();
    lblServicedate.Text = sc.RequestReplyPattern();
}
}
```

3. IService.cs:-

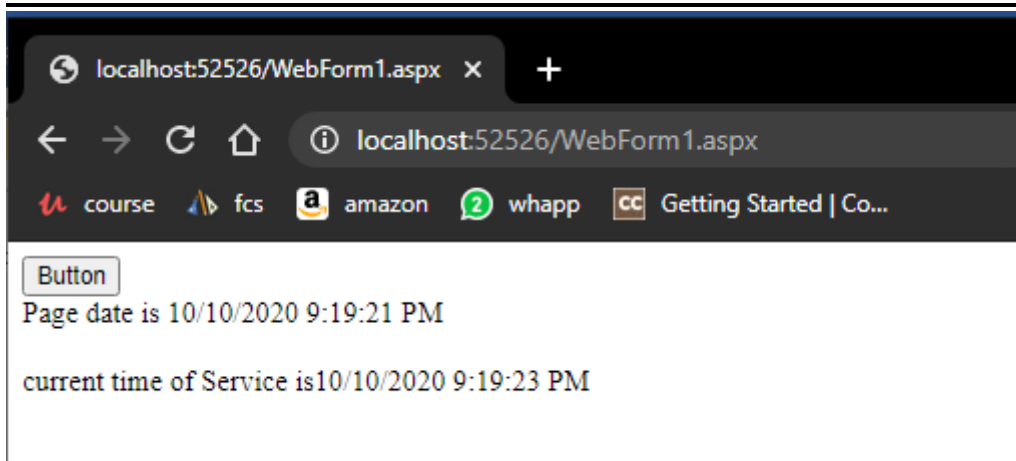
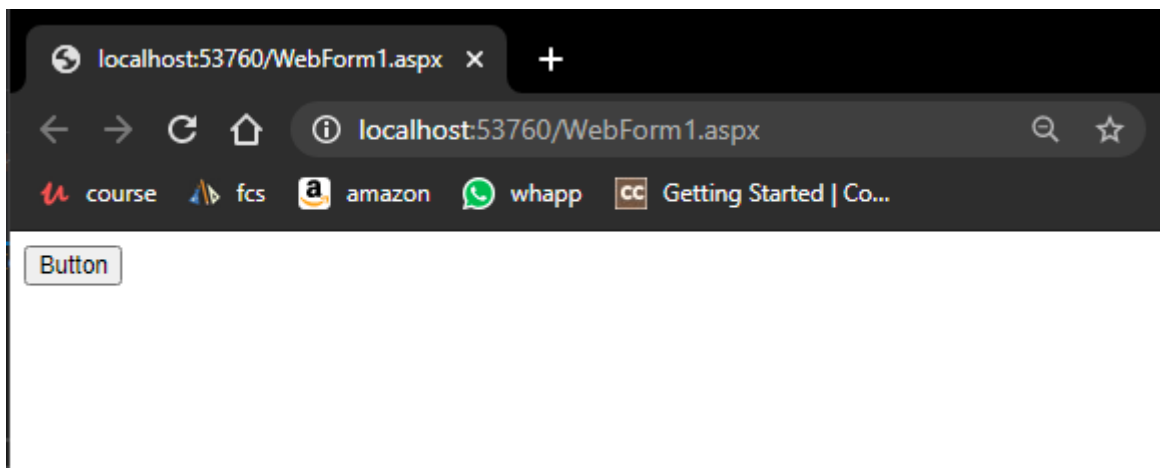
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
namespace WcfService2
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract(IsOneWay = false)]
        //declare the method which return type is string
        string RequestReplyPattern();
    }
}
```

4. Service1.svc.cs

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

namespace WcfService2
{
```

```
public class Service1 : IService1
{
    public string RequestReplyPattern()
    {
        Thread.Sleep(2000);
        return "current time of Service is" + DateTime.Now.ToString();
    }
}
```

Output:-

Date: 30/09/2020

Practical no 3

AIM: Demonstrates using the binding attribute of an endpoint element in WCF with webform.

Program Code:-**IService1.cs**

```
using System;
using System.Collections.Generic;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
namespace WcfService7
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
        string Recharge(string Name, string company, string number, int amount);
    }
}
```

Service1.svc.cs

```
using System;
using System.Collections.Generic;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
namespace WcfService7
{
    public class Service1 : IService1
    {
        public string Recharge(string Name, string company, string number, int amount)
        {
            string message = string.Empty;

            if (string.IsNullOrEmpty(Name) || string.IsNullOrEmpty(company))
            {
                message = "Please Enter your name or company name";
            }
        }
    }
}
```

```
    }  
    else  
    {  
        if (number.Length == 10 && amount > 0)  
        {  
            message = "Recharge of " + amount + " Rs has been done successfully.";  
        }  
        else  
        {  
            message = "Recharge unsuccessfull. Please try again";  
        }  
    }  
    return message;  
}  
}
```

WebForm.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WcfService7.WebForm1" %>
```

```
<!DOCTYPE html>
```

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

```
<head runat="server">
```

```
    <title>>Mobile Recharge<</title>
```

```
    <style>
```

```
    * {
```

```
    box-sizing: border-box
```

```
    }
```

```
    /* Add padding to containers */
```

```
    .container {
```

```
    padding: 16px;
```

```
    }
```

```
    /* Full-width input fields */
```

```
    input[type=text], input[type=password] {
```

```
    width: 100%;
```

```
    padding: 15px;
```

```
    margin: 5px 0 22px 0;
```

```
    display: inline-block;
```

```
    border: none;
```

```
    background: #f1f1f1;
```

```
    }
```

```
    input[type=text]:focus, input[type=password]:focus {
```



```
background-color: #ddd;
outline: none;
}
/* Overwrite default styles of hr */
hr {
border: 1px solid #f1f1f1;
margin-bottom: 25px;
}
/* Set a style for the submit/register button */
.registerbtn {
background-color: #4CAF50;
color: white;
padding: 16px 20px;
margin: 8px 0;
border: none;
cursor: pointer;
width: 100%;
opacity: 0.9;
}
.registerbtn:hover {
opacity: 1;
}

/* Add a blue text color to links */
a {
color: dodgerblue;
}
/* Set a grey background color and center the text of the "sign in" section */
.signin {
background-color: #f1f1f1;
text-align: center;
}
</style>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <div class="container">
        <caption class="auto-style1">
          <h1>Recharge</h1>
          <hr>
          <label for="email"><b>Name</b></label>
          <asp:TextBox ID="name" placeholder="Enter full name"
runat="server"></asp:TextBox>
          <label for="company"><b>Company</b></label>
          <asp:TextBox ID="company" placeholder="Enter full name"
runat="server"></asp:TextBox>
```

```
<label for="no"><b>Mobile number</b></label>
<asp:TextBox ID="no" placeholder="Enter full name" runat="server"></asp:TextBox>
<label for="amount"><b>Amount</b></label>
<asp:TextBox ID="amount" placeholder="Enter amount"
runat="server"></asp:TextBox>
<hr>
<asp:Button ID="rechargebtn" runat="server" Text="Recharge" BackColor="#4CAF50"
OnClick="rechargebtn_Click" Height="48px" Width="159px" />
<hr>
<asp:Label ID="lbl" runat="server" Text=""></asp:Label>
</div>
</div>
</form>

</body>
</html>
```

WebForm.aspx.cs

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

using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WcfService7
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        protected void rechargebtn_Click(object sender, EventArgs e)
        {
            try
            {
                ServiceReference1.Service1Client client = new ServiceReference1.Service1Client();
                lbl.Text = client.Recharge(name.Text, company.Text, no.Text,
                Convert.ToInt32(amount.Text));
            }
            catch (Exception)
            {

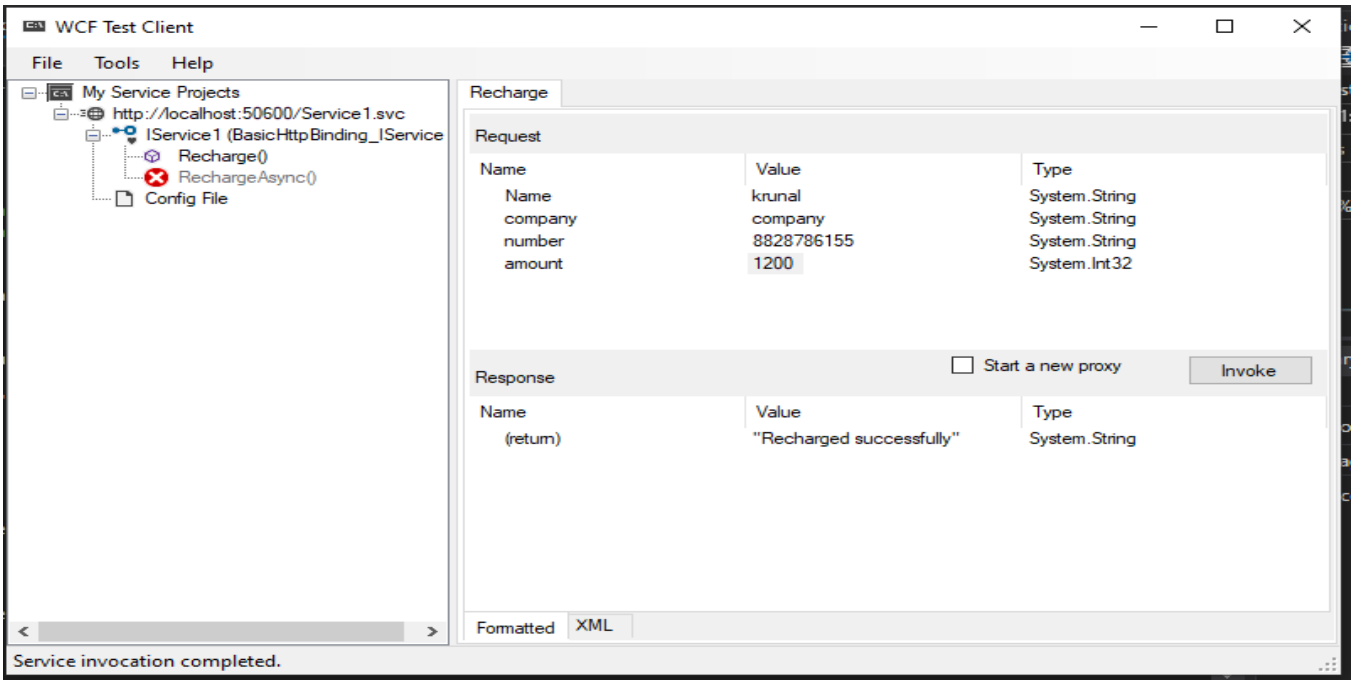
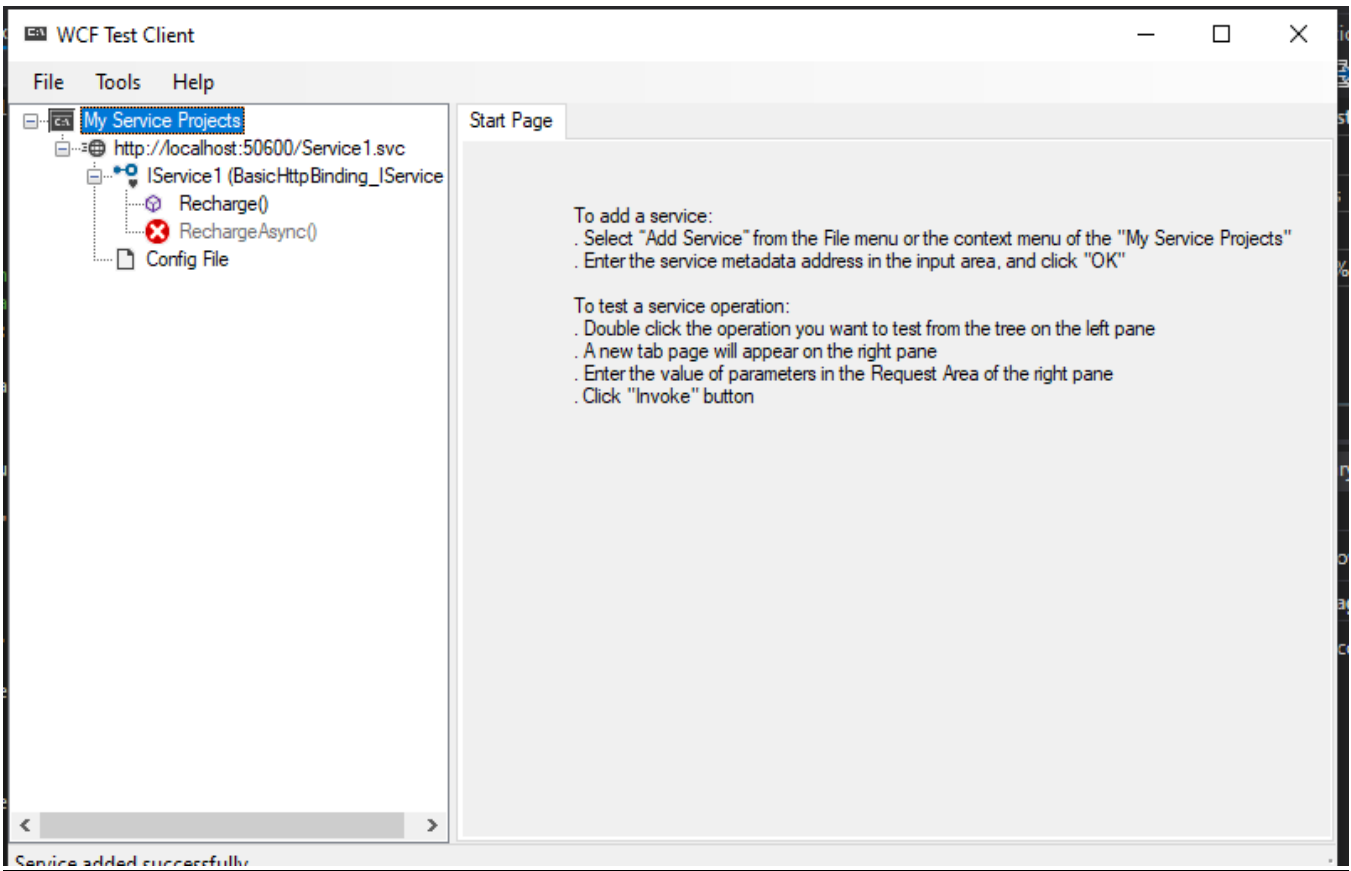
            }
        }
    }
}
```

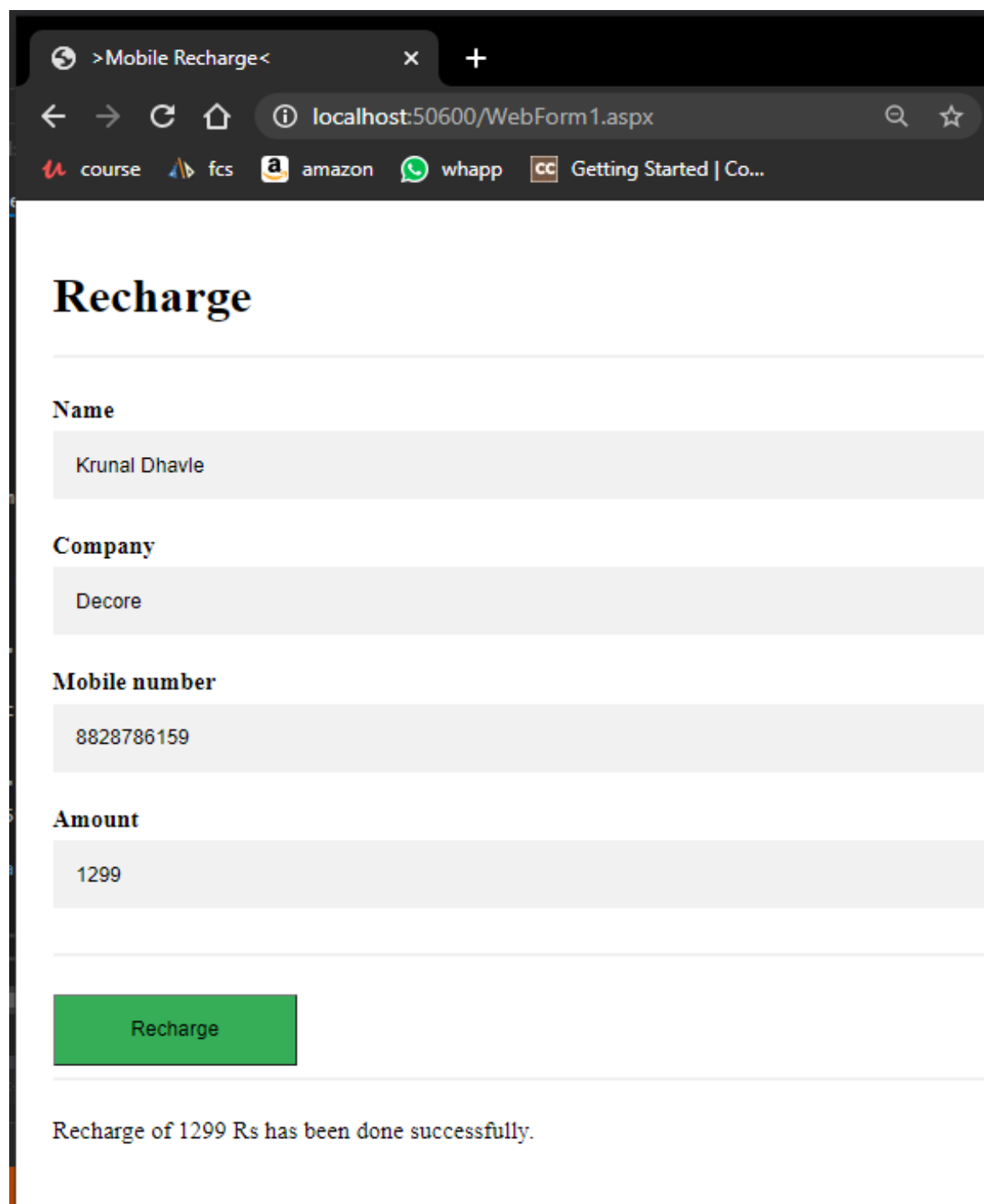
```
}  
  
}
```

WebConfig file

```
<bindings>  
  <basicHttpBinding>  
    <binding name="BasicHttpBinding IService1" />  
  </basicHttpBinding>  
</bindings>  
<client>  
  <endpoint address="http://localhost:50600/Service1.svc" binding="basicHttpBinding"  
    bindingConfiguration="BasicHttpBinding IService1" contract="ServiceReference1.IService1"  
    name="BasicHttpBinding IService1" />  
</client>
```

Output:





The screenshot shows a web browser window with a dark theme. The address bar displays 'localhost:50600/WebForm1.aspx'. The browser's tab bar shows a single tab titled '>Mobile Recharge<'. The browser's bookmark bar contains several icons: 'course', 'fcs', 'amazon', 'whapp', and 'Getting Started | Co...'. The main content area of the browser displays a web form titled 'Recharge'. The form has four input fields: 'Name' (containing 'Krunal Dhavle'), 'Company' (containing 'Decore'), 'Mobile number' (containing '8828786159'), and 'Amount' (containing '1299'). Below these fields is a green button labeled 'Recharge'. At the bottom of the form, a message states: 'Recharge of 1299 Rs has been done successfully.'

Recharge

Name

Krunal Dhavle

Company

Decore

Mobile number

8828786159

Amount

1299

Recharge

Recharge of 1299 Rs has been done successfully.

> Mobile Recharge <

localhost:50600/WebForm1.aspx

course fcs amazon whapp Getting Started | Co...

Recharge

Name

Krunal Dhavle

Company

Enter full name

Mobile number

08828786159

Amount

12200

Recharge

Please Enter your name or company name

Date: 07/10/2020

Practical no 4

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

Program Code:-**WebForm1.aspx**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication2.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script src="jquery.js"></script>
  <script type="text/javascript">
    $(document).ready(function () {
      $("#btn").click(function () {
        var num1 = $("#txt1").val();
        var num2 = $("#txt2").val();
        $.ajax({
          url: "Service1.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>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <input id="txt1" type="text" />
      <br />
      <br />
      <input id="txt2" type="text" />
      <br />
    </div>
  </form>
</body>
</html>
```

```
<br />
<input id="btn" type="button" value="Add Number" />
<br />
<br />
<input id="txt3" type="text" />
<p> Performed by krunal 713</p>
</div>
</form>
</body>
</html>
```

Service1.svc.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 WebApplication2
{
    [ServiceContract(Namespace = "Multiplication")]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
    public class Service1
    {
        [OperationContract]
        public double Sum(double a, double b)
        {
            double result = a + b;
            return result;
        }
    }
}
```

Web.Config

```
<?xml version="1.0" encoding="utf-8"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
```



```

<system.web>
  <compilation debug="true" targetFramework="4.7.1"/>
  <httpRuntime targetFramework="4.7.1"/>
</system.web>
<system.codedom>
  <compilers>
    <compiler language="c#;cs;csharp" extension=".cs"
      type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Mi-
crosoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.0.0, Culture=neutral, PublicKey-
Token=31bf3856ad364e35"
      warningLevel="4" compilerOptions="/langversion:default /nowarn:1659;1699;1701"/>
    <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
      type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Mi-
crosoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.0.0, Culture=neutral, PublicKey-
Token=31bf3856ad364e35"
      warningLevel="4" compilerOptions="/langversion:default /nowarn:41008 /de-
fine:_MYTYPE=\&quot;Web\&quot; /optionInfer+"/>
  </compilers>
</system.codedom>

<system.serviceModel>
  <behaviors>
    <endpointBehaviors>
      <behavior name="WebApplication2.Service1AspNetAjaxBehavior">
        <enableWebScript />
      </behavior>
    </endpointBehaviors>
  </behaviors>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
    multipleSiteBindingsEnabled="true" />
  <services>
    <service name="WebApplication2.Service1">
      <endpoint address="" behaviorConfiguration="WebApplication2.Service1AspNetAjaxBehav-
ior"
        binding="webHttpBinding" contract="WebApplication2.Service1" />
    </service>
  </services>
</system.serviceModel>
</configuration>

```

Output:-

Add Number

Performed by krunal 713

Add Number

Performed by krunal 713

Date: 13/10/2020

Practical no 5**AIM:** Implement a typical service and a typical client using WCF.**Program Code:-****IService1.cs**

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

namespace WcfService8
{
    [ServiceContract]
    public interface IService1
    {
        [OperationContract]
        int add(int a, int b);
        [OperationContract]
        int Sub(int a, int b);
        [OperationContract]
        int Mul(int a, int b);
        [OperationContract]
        int Div(int a, int b);
    }
}
```

```
}  
  
}
```

Service1.svc.cs:-

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Runtime.Serialization;  
using System.ServiceModel;  
using System.ServiceModel.Web;  
using System.Text;  
namespace WcfService8  
{  
    public class Service1 : IService1  
    {  
        int IService1.add(int a, int b)  
        {  
            return (a + b);  
        }  
        int IService1.Sub(int a, int b)  
        {  
            if (a >= b)  
                return (a - b);  
            else  
                return (b - a);  
        }  
        int IService1.Mul(int a, int b)
```

```
{
    return (a * b);
}
int IService1.Div(int a, int b)
{
    if (a >= b)
        return (a / b);
    else
        return (b / a);
}
}
```

Webform1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WcfService8.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server" style="background-color:lightgray">
        <div>
            Enter first number
            <asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox> &nbsp; &nbsp; &nbsp; &nbsp;
            Enter second number
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox> <br /><br />
```

```
<asp:Button ID="Button1" runat="server" Text="Add" OnClick="btnadd_Click" />&nbsp;
&nbsp;&nbsp; 
<asp:Button ID="Button2" runat="server" Text="Sub" OnClick="btnsub_Click"/>&nbsp;
&nbsp;&nbsp; 
<asp:Button ID="Button3" runat="server" Text="Mul" OnClick="btnmul_Click"/>&nbsp;
&nbsp;&nbsp; 
<asp:Button ID="Button4" runat="server" Text="Div" OnClick="btndiv_Click"/> <br /><br />
<asp:Label ID="Label2" runat="server" Text="Result:-"></asp:Label>&nbsp; &nbsp;&nbsp;&nbsp;
<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox> <br /><br /><br /><br />
performed by krunal 713
</div>
</form>
</body>
</html>
```

WebForm.aspx.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WcfService8

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        ServiceReference1.Service1Client Client = new ServiceReference1.Service1Client();

        protected void Page_Load(object sender, EventArgs e)
```

```
{ }  
int a, b;  
protected void btnadd_Click(object sender, EventArgs e)  
{  
    a = Convert.ToInt32(TextBox1.Text);  
    b = Convert.ToInt32(TextBox2.Text);  
    int Addition = Client.add(a, b);  
    TextBox3.Text = Addition.ToString();  
}  
protected void btnsub_Click(object sender, EventArgs e)  
{  
    a = Convert.ToInt32(TextBox1.Text);  
    b = Convert.ToInt32(TextBox2.Text);  
    int sub = Client.Sub(a, b);  
    TextBox3.Text = sub.ToString();  
}  
protected void btnmul_Click(object sender, EventArgs e)  
{  
    a = Convert.ToInt32(TextBox1.Text);  
    b = Convert.ToInt32(TextBox2.Text);  
    int mul = Client.Mul(a, b);  
    TextBox3.Text = mul.ToString();  
}  
protected void btndiv_Click(object sender, EventArgs e)  
{  
    a = Convert.ToInt32(TextBox1.Text);  
    b = Convert.ToInt32(TextBox2.Text);  
    int div = Client.Div(a, b);
```

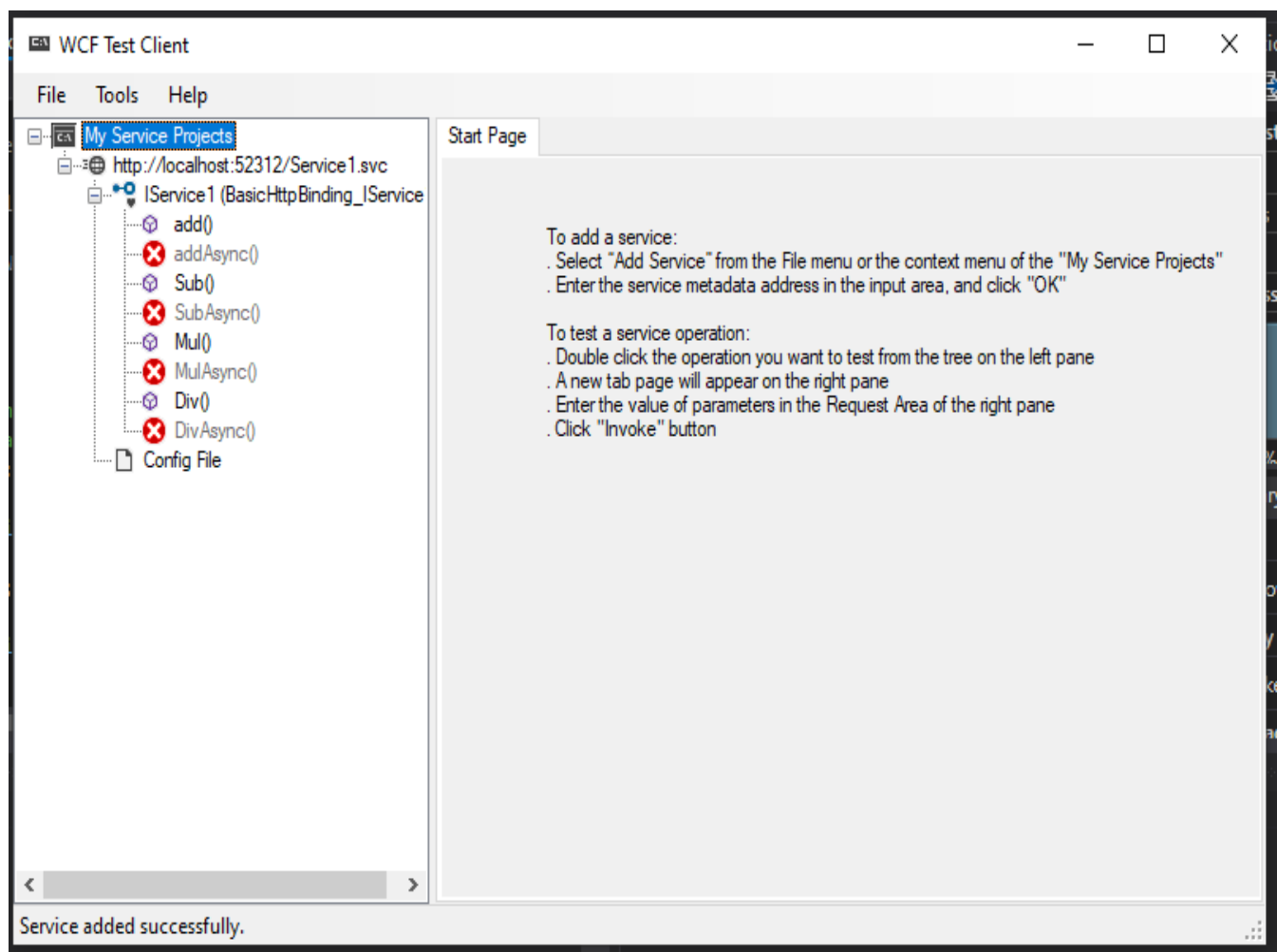
```
TextBox3.Text = div.ToString();
```

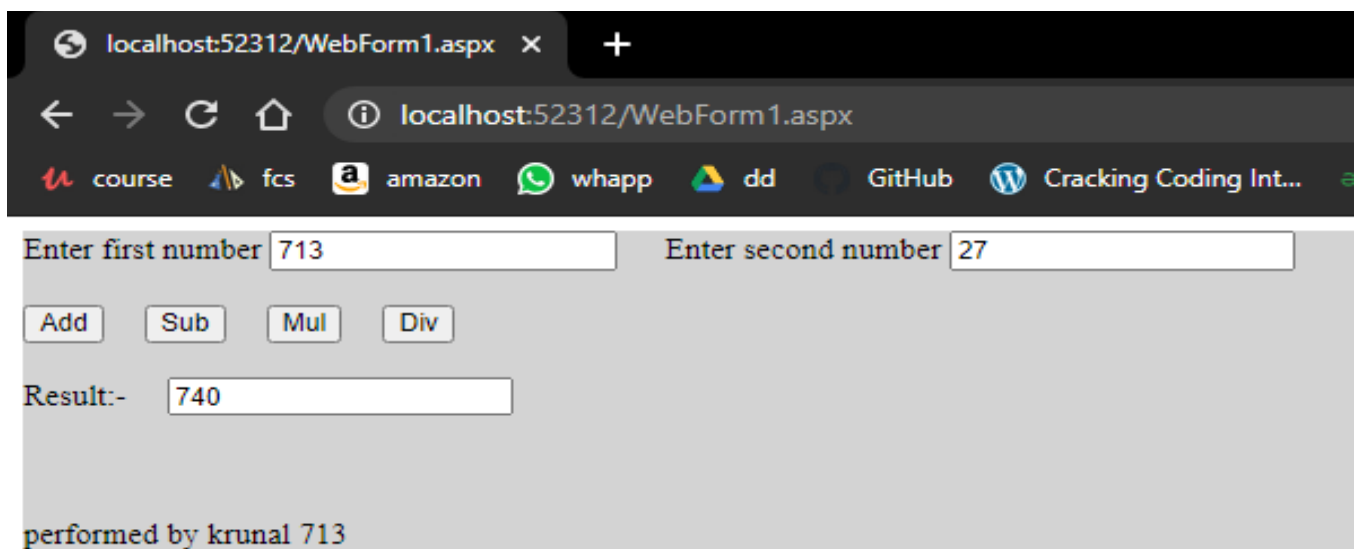
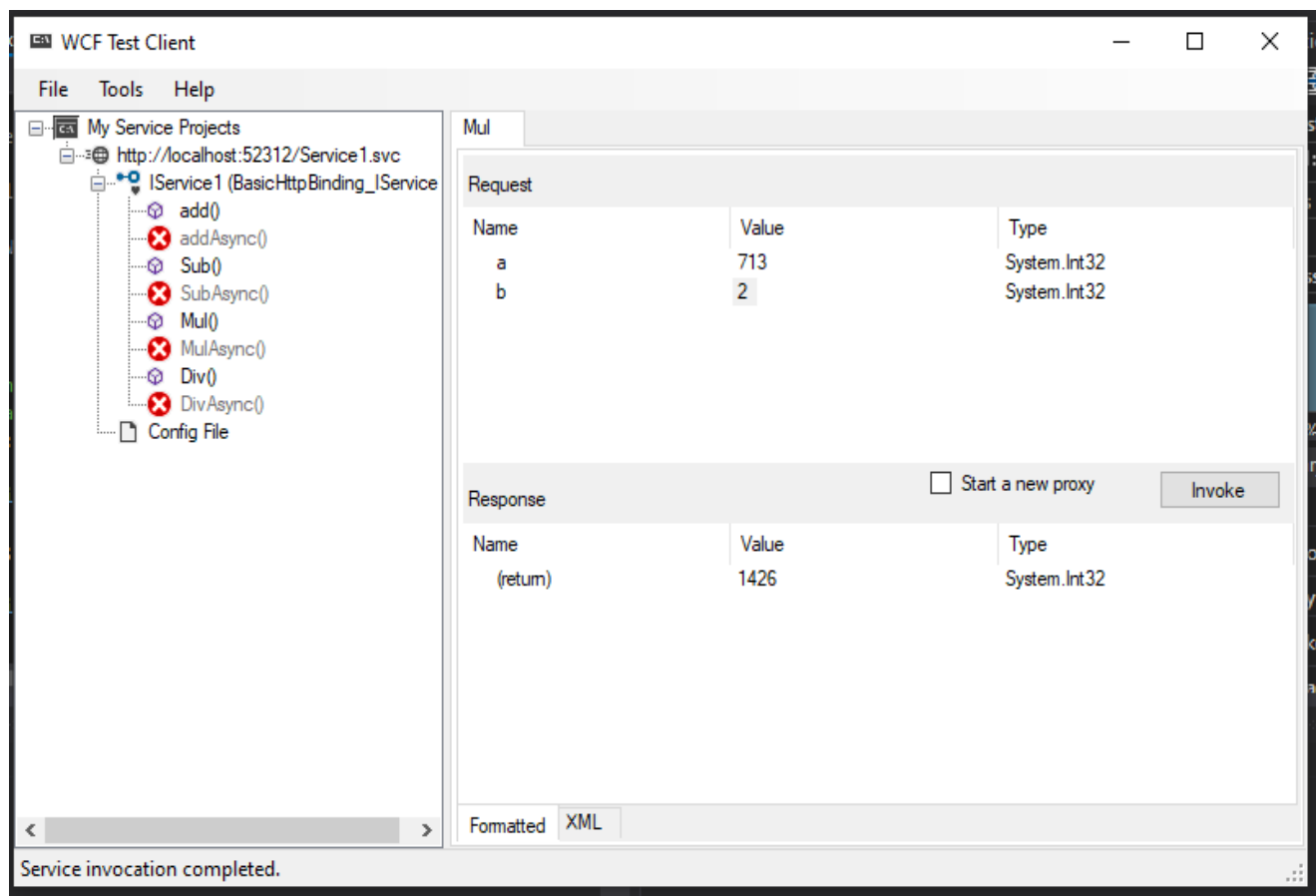
```
}
```

```
}
```

```
}
```

Output:-





Date: 14/10/2020

Practical no 6**AIM:** Develop client which consumes web services developed in different platform.**Program Code:-****WebService1.asmx.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
namespace WebApplication4
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    public class WebService1 : System.Web.Services.WebService
    {
        [WebMethod]
        public double Mod(double number1 , double number2)
        {
            return (number1 % number2);
        }
    }
}
```

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mod </title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form>
      <input type="text" name="txt1" placeholder="Enter First Number"><br><br>
      <input type="text" name="txt2" placeholder="Enter Second Number"><br><br>
      <input type="submit" formaction="Mod.jsp" value="Mod Number"><br><br>
      performed by krunal dhavle 713;
    </form>
  </body>
</html>
```

Mod.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>
```

```
<%-- start web service invocation --%><hr/>

<%

    double num1 = Double.parseDouble(request.getParameter("txt1"));
    double num2 = Double.parseDouble(request.getParameter("txt2"));
    try {
        com.dd.WebService1 service = new com.dd.WebService1();
        com.dd.WebService1Soap port = service.getWebService1Soap();
        // TODO initialize WS operation arguments here
        double number1 = num1;
        double number2 = num2;
        // TODO process result here
        double result = port.mod(number1, number2);
        out.println("Result = "+result);
    } catch (Exception ex) {
    }
%>

</body>

</html>
```

Output:-

WebService1

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

- [Mod](#)

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 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. 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 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.

Mod

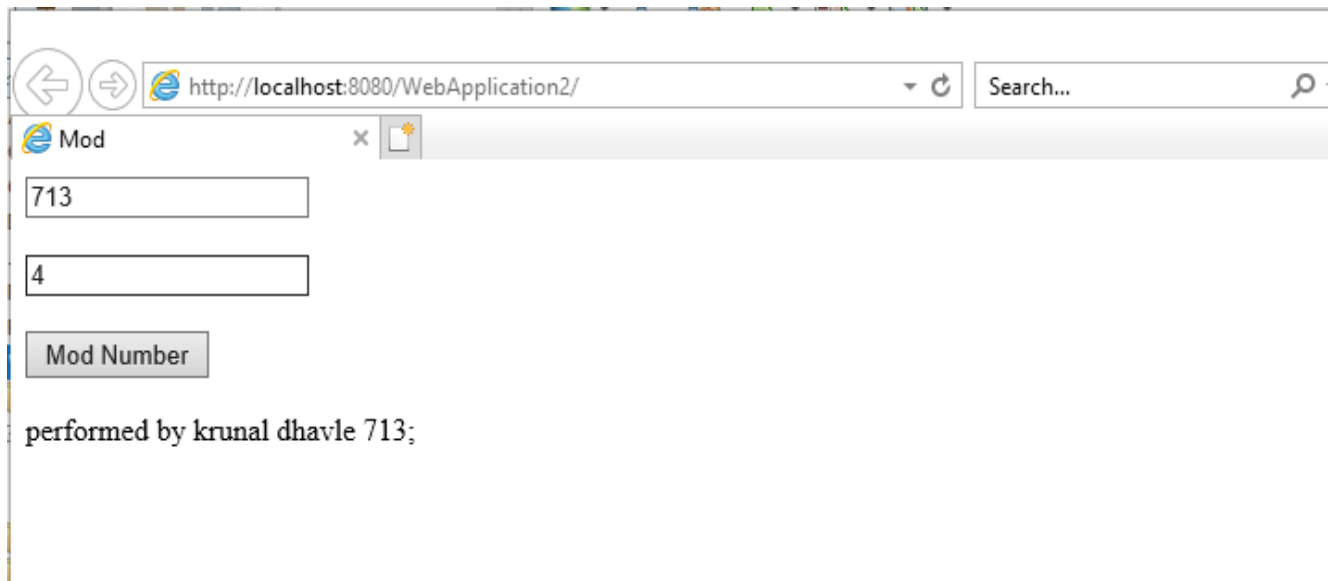
Test

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

Parameter	Value
number1:	<input type="text" value="12"/>
number2:	<input type="text" value="8"/>

Invoke

SOAP 1.1



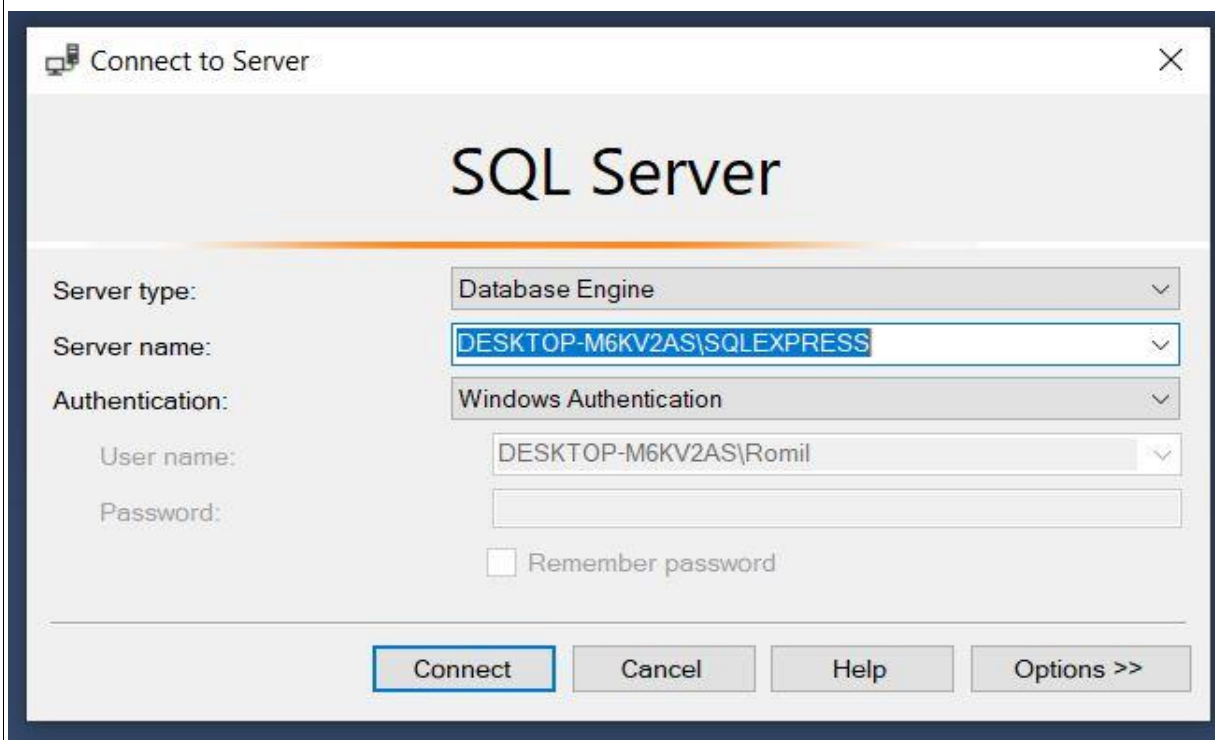
Date: 21/10/2020

Practical no 7

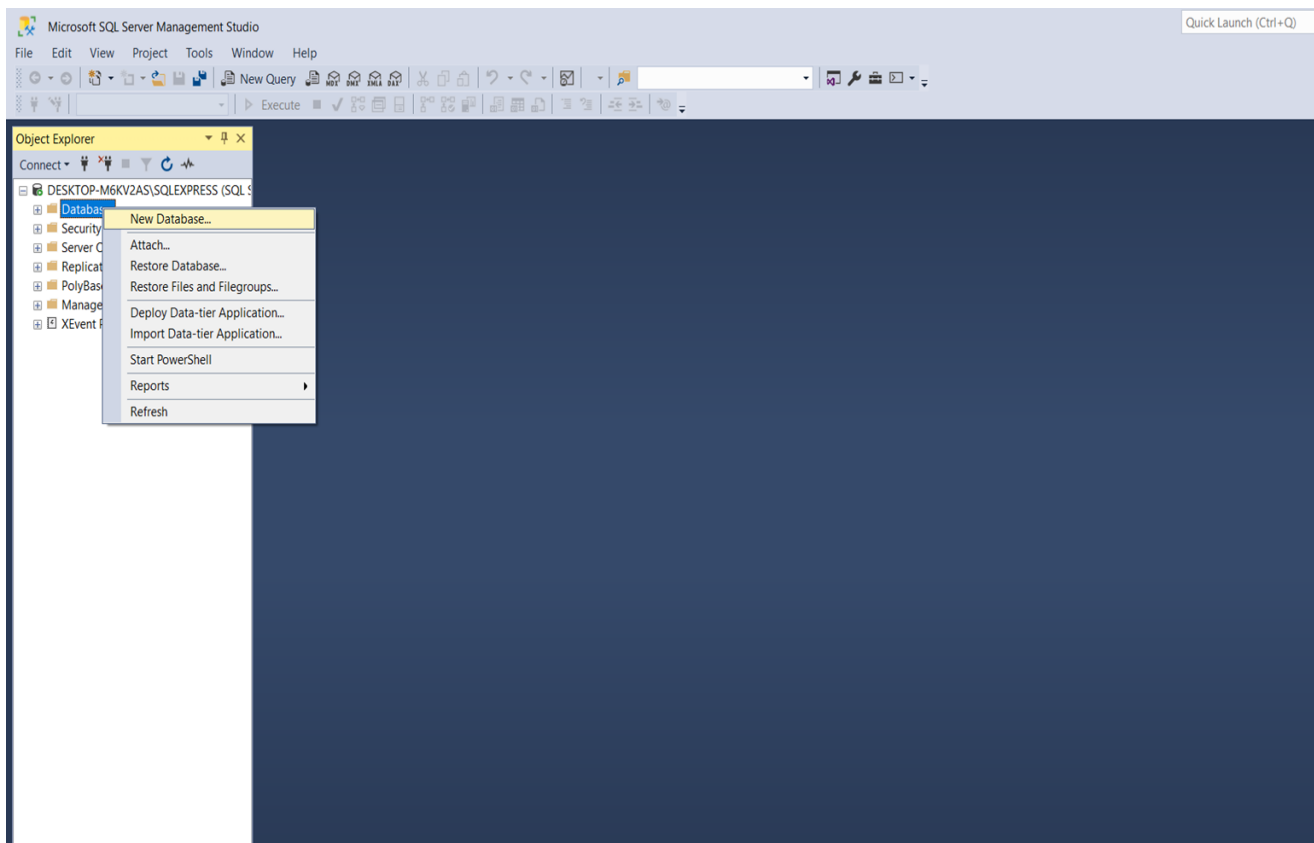
AIM: ; Define a web service method that returns the contents of a database in a JSON string. The contents should be displayed in a tabular format.

Steps for creating a Database:

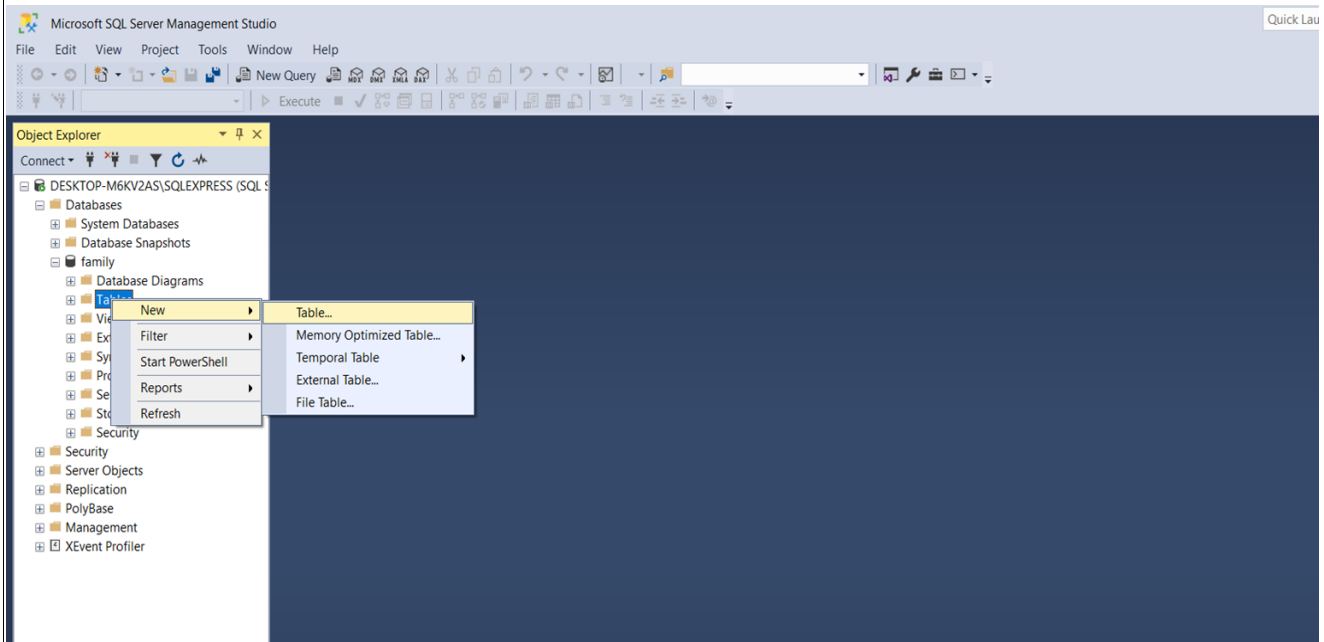
- 1] Open Microsoft SQL Server Management Studio. After opening the Management Studio it will ask to connect to a server. Select a proper server and click connect.



- 2] From object explorer right click on **Databases** and click **New Database**. Give name to Database.



3] Right Click on **Tables** and select **New ->Table**.



- 4] Edit column name and datatypes.
- 5] Save the table and give a name.
- 6] From **Tables** select the table you created. Right click on it and select **Edit top 200 rows**.
- 7] Enter the data into the table and save.

SQLQuery2.sql - DE...QJO\BlackBot (69)) DESKTOP-8F3RQJO\S...mily - dbo.family X DESKTOP-8F3RQJO\S...mily - dbo.family*					
	name	gender	age	relation	dob
	krunal	male	21	son	09/09/1999
	lara	female	20	daughter	10/11/2000
	varsha	female	42	mom	10/10/1982
	harsha	female	45	mom	4/12/1974
	snuffy	male	3	pet	9/8/2017
▶*	NULL	NULL	NULL	NULL	NULL

CODE:-**Family.cs:**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace prac7
{
    public class Family
    {
        public string name { get; set; }
        public string gender { get; set; }
        public string age { get; set; }
        public string relation { get; set; }
        public string dob { get; set; }
    }
}

```

Webservice.asmx.cs:

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

```

```
using System.Web.Script.Serialization;
using System.Web.Services;

namespace prac7
{
    /// <summary>
    /// Summary description for WebService1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    public class WebService1 : System.Web.Services.WebService
    {

        [WebMethod]
        public string JsonData()
        {
            List<Family> objFamilylist = new List<Family>();
            string str = ConfigurationManager.ConnectionStrings["DBContext"].ConnectionString;
            using (SqlConnection con = new SqlConnection(str))
            {
                SqlCommand cmd = new SqlCommand("select * from family", con);
                cmd.Connection = con;
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    Family family = new Family();
                    family.name = rdr["Name"].ToString();
                    family.gender = rdr["Gender"].ToString();
                    family.age = rdr["Age"].ToString();
                    family.relation = rdr["Relation"].ToString();
                    family.dob = rdr["Dob"].ToString();
                    objFamilylist.Add(family);
                }
                JavaScriptSerializer js = new JavaScriptSerializer(); return js.Serialize(objFamilylist);
            }
        }
    }
}
```

Web.config:-

```

<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.7.1"/>
    <httpRuntime targetFramework="4.7.1"/>
  </system.web>
  <connectionStrings>
    <add name="DBContext" providerName="System.Data.SqlClient" connectionString="Data
Source=DESKTOP-8F3RQJO\SQLEXPRESS01;Initial Catalog=family;Integrated Security=True "/>
  </connectionStrings>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="WebService1Soap" />
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost:58555/WebService1.asmx"
        binding="basicHttpBinding"
        bindingConfiguration="WebService1Soap"
        contract="ServiceReference1.WebService1Soap"
        name="WebService1Soap" />
    </client>
  </system.serviceModel>
</configuration>

```

WebService1 Web Service

localhost:50624/WebService1.asmx

amazon whapp dd GitHub cs AC cshell git js30 q cp sp Other bookmarks

WebService1

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

- [JsonData](#)

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.)

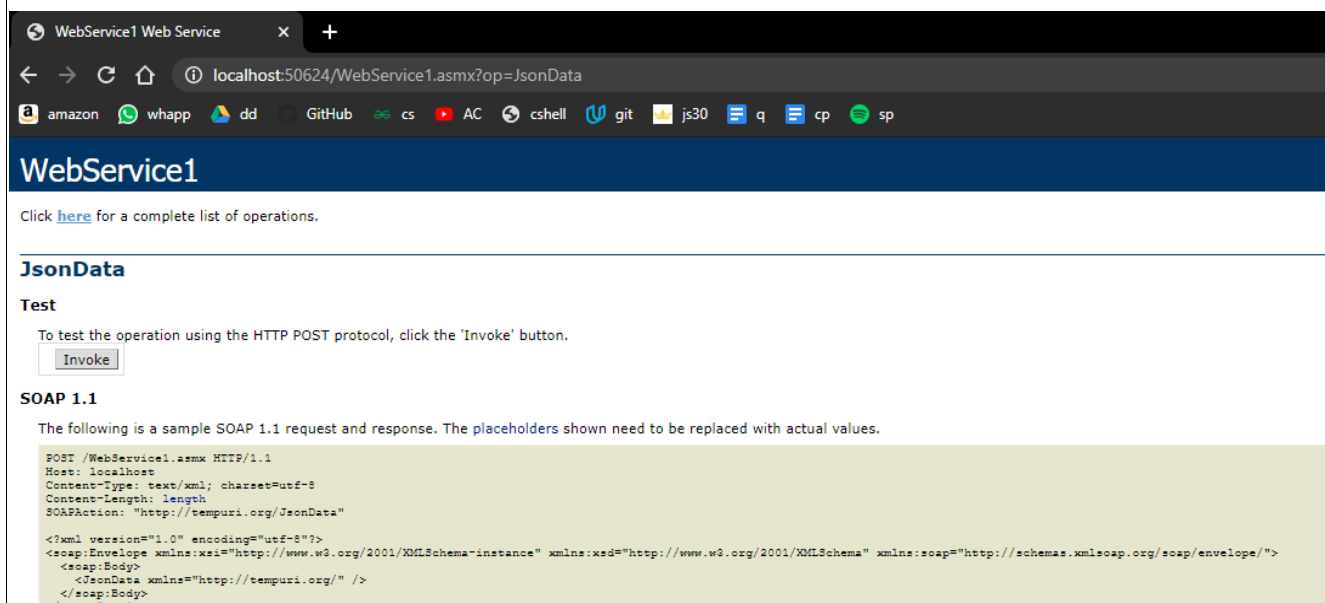
For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

```

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

```

Visual Basic



Now adding Webform in the same solution explorer .

(adding newtonsoft.json)

Webform1.aspx.cs:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="prac7.WebForm1" %>

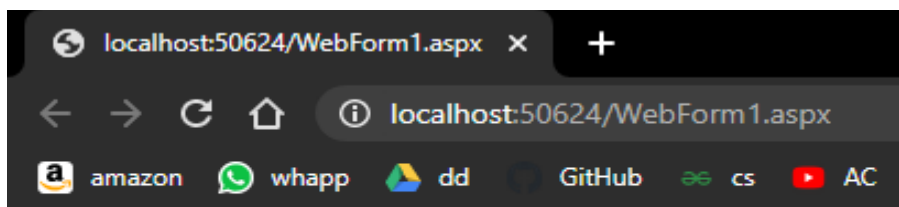
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
<div>
<asp:GridView runat="server" ID="GridView"
OnSelectedIndexChanged="GridView_SelectedIndexChanged"></asp:GridView>
</div>
<h3>Performed By krupal, 713</h3>
</form>
</body>
</html>
```

Webform.aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="prac7.WebForm1" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
<div>
<asp:GridView runat="server" ID="GridView"
OnSelectedIndexChanged="GridView_SelectedIndexChanged"></asp:GridView>
</div>
<h3>Performed By krunal, 713</h3>
</form>
</body>
</html>
```

Output:-



name	gender	age	relation	dob
krunal	male	21	son	09/09/1999
lara	female	20	daughter	10/11/2000
varsha	female	42	mom	10/10/1982
harsha	female	45	mom	4/12/1974
snuffy	male	3	pet	9/8/2017

Performed By krunal, 713

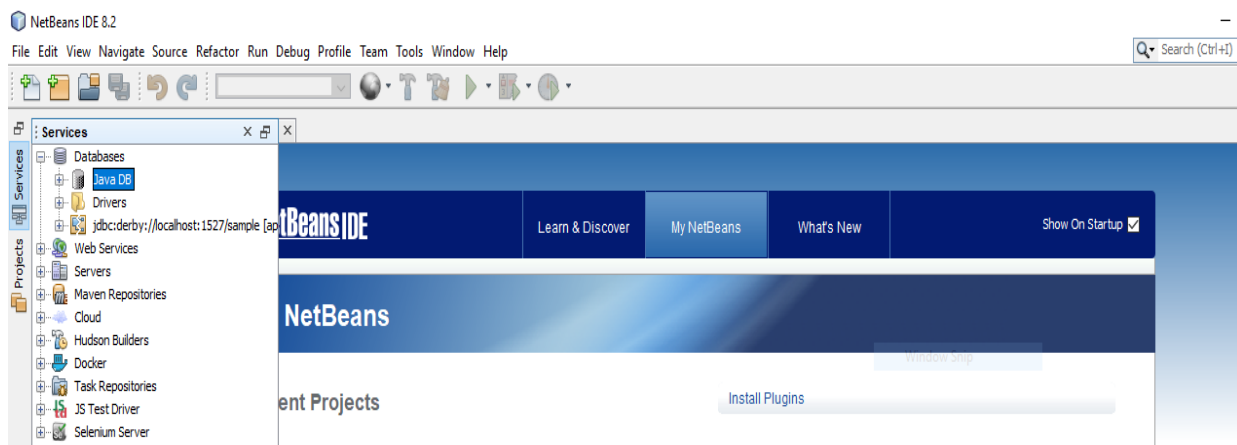
Date: 28/10/2020

Practical no 8

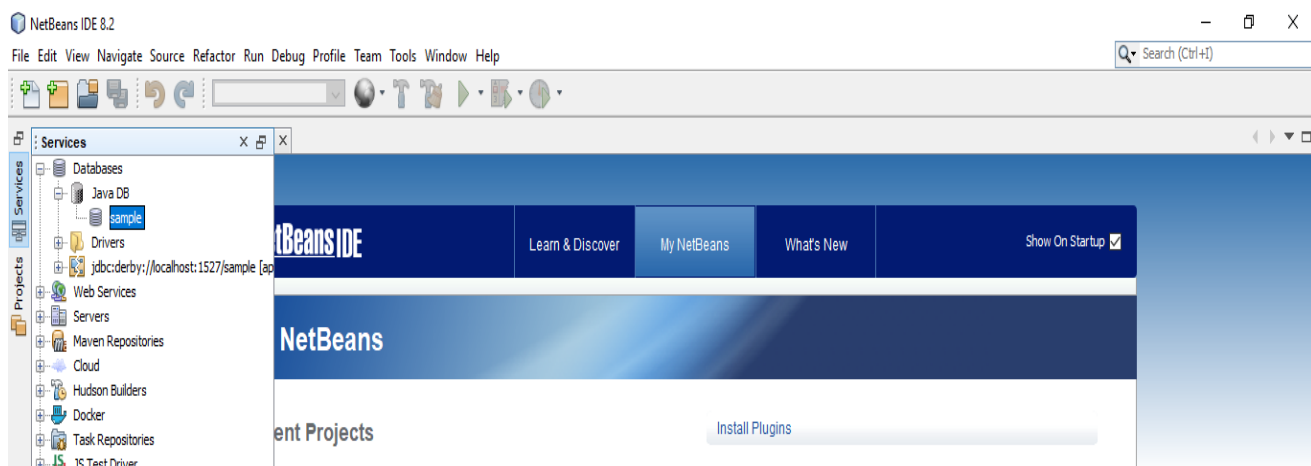
AIM: Define a RESTful web service that accepts the details to be stored in a database and performs CRUD operation.

Steps :-

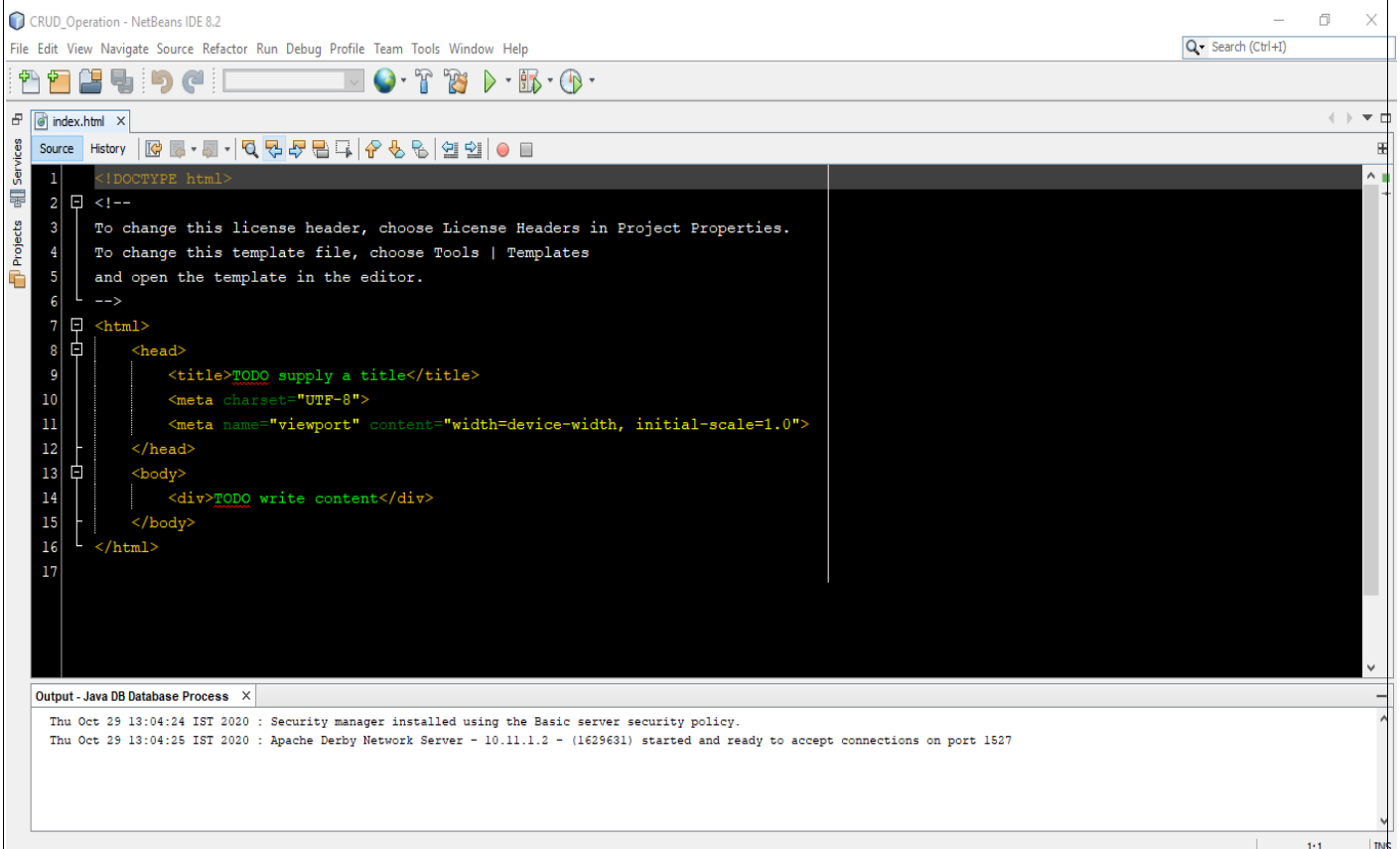
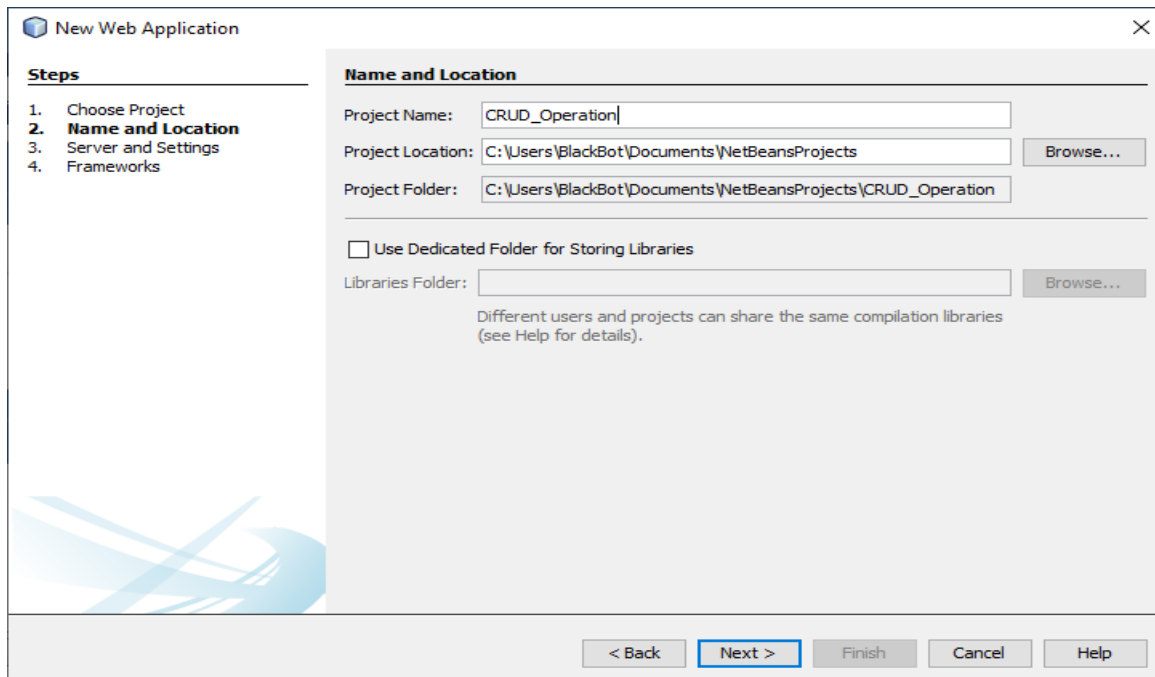
1) Right click on Java DB and then click on Start Server to start the server .



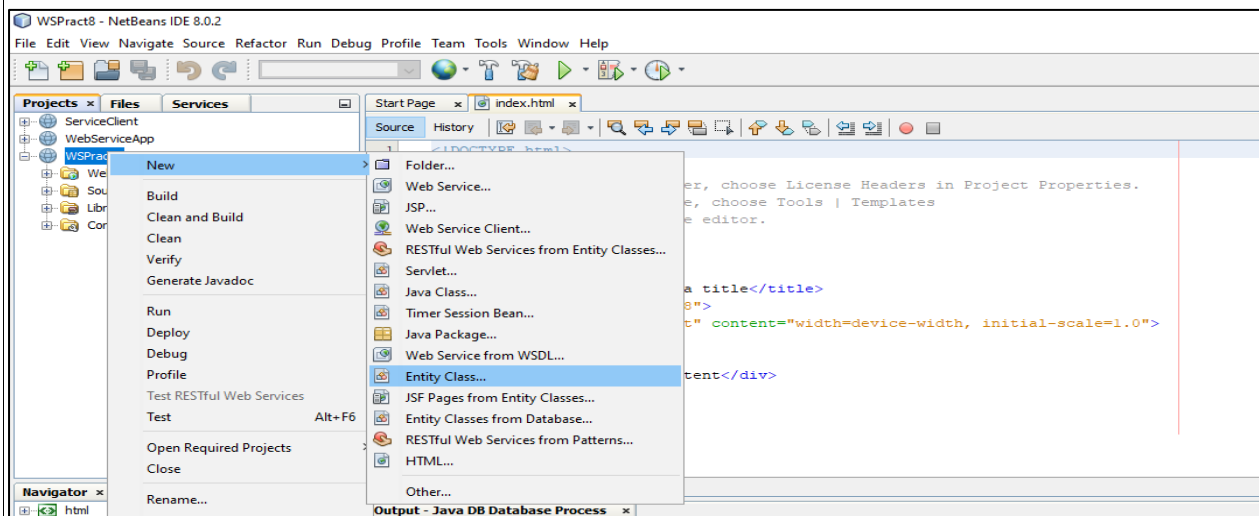
2) Now expand Java DB and right click on sample and then click on connect to connect the sample database with server.



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



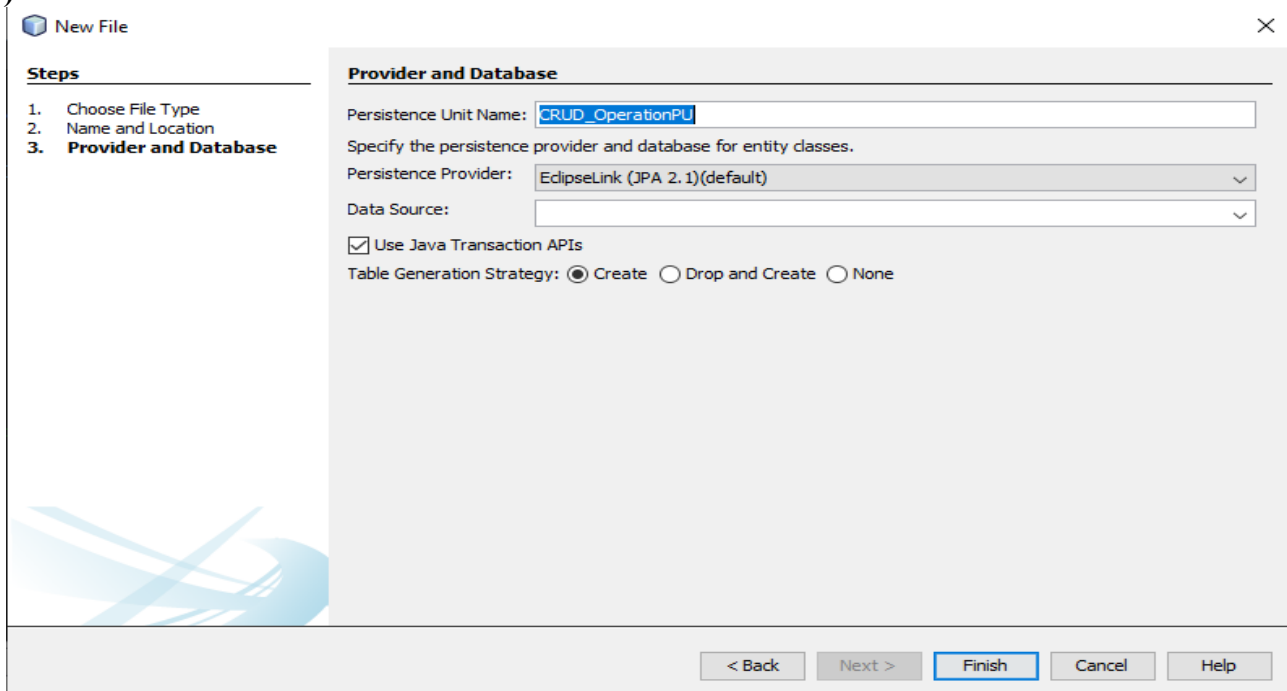
4) Create an entity class. **Right click on project name -> New -> Entity Class.**



5) A window will appear like bellow pic. Enter following data and click on Next
Class Name : students
Package name : CRUD Operation

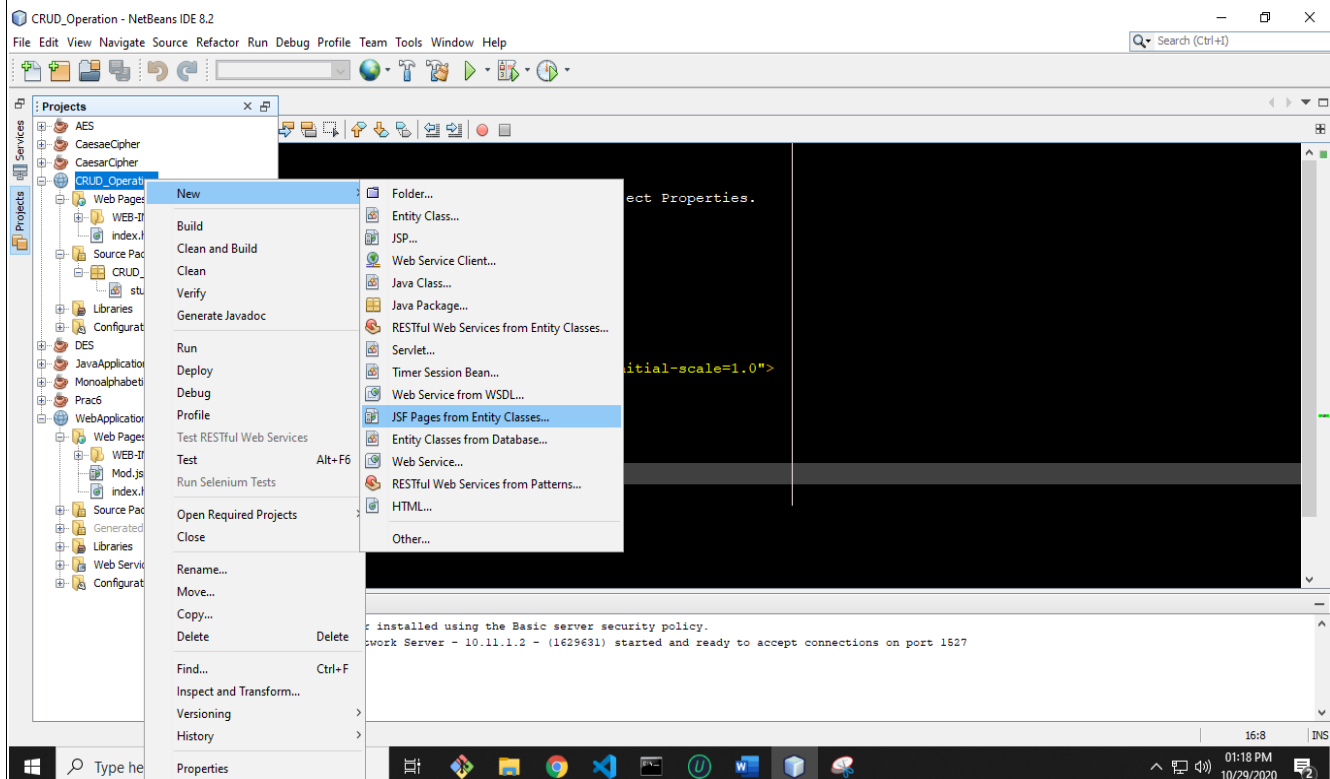
A screenshot of the 'New Entity Class' dialog box in NetBeans. The dialog has a 'Steps' section on the left with three steps: '1. Choose File Type', '2. Name and Location', and '3. Provider and Database'. The 'Name and Location' section is active, showing fields for 'Class Name' (students), 'Project' (CRUD_Operation), 'Location' (Source Packages), 'Package' (CRUD_Operation), and 'Created File' (Documents\NetBeansProjects\CRUD_Operation\src\java\CRUD_Operation\students.java). There is a 'Primary Key Type' field set to 'Long' and a 'Create Persistence Unit' checkbox which is checked. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

6) Click on Finish.



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

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



8) Select `wspract8.students` and click on Add button and then Next button on below.

New JSF Pages from Entity Classes

Steps

1. Choose File Type
2. **Entity Classes**
3. Generate JSF Pages and Classes
4. JavaServer Faces Configuration

Entity Classes

Available Entity Classes:

CRUD_Operation.students

Selected Entity Classes:

☒ Include Referenced Classes

Select entity classes.

< Back Next > Finish Cancel Help

New JSF Pages from Entity Classes

Steps

1. Choose File Type
2. **Entity Classes**
3. Generate JSF Pages and Classes
4. JavaServer Faces Configuration

Entity Classes

Available Entity Classes:

Selected Entity Classes:

CRUD_Operation.students

☒ Include Referenced Classes

< Back Next > Finish Cancel Help

9) 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

The screenshot shows the 'New JSF Pages from Entity Classes' dialog box. On the left, a 'Steps' list shows: 1. Choose File Type, 2. Entity Classes, 3. **Generate JSF Pages and Classes**, and 4. JavaServer Faces Configuration. The main area is titled 'Generate JSF Pages and Classes' and contains the following fields and options:

- Project: CRUD_Operation
- Location: Source Packages (dropdown)
- Session Bean Package: CRUD_Operation (dropdown)
- JSF Classes Package: CRUD_Operation (dropdown)
- Specify the location of new JSF pages:
 - JSF Pages Folder: (empty text field) with a 'Browse...' button.
 - Localization Bundle Name: /Bundle
 - ☐ Override existing files
- Choose Templates: Standard JavaServer Faces (dropdown) with a 'Customize Template' link.

At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

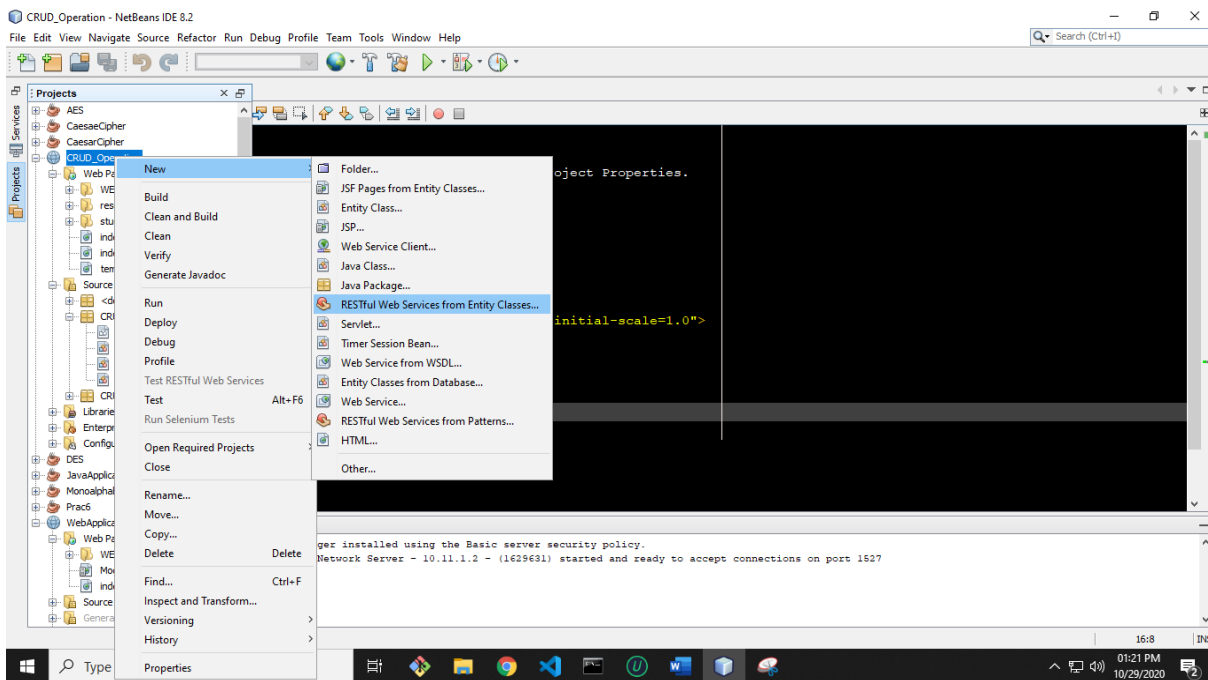
10) Now click on Finish

The screenshot shows the 'New JSF Pages from Entity Classes' dialog box, Step 4: JavaServer Faces Configuration. The 'Steps' list on the left shows: 1. Choose File Type, 2. Entity Classes, 3. Generate JSF Pages and Classes, and 4. **JavaServer Faces Configuration**. The main area is titled 'JavaServer Faces Configuration' and contains the following fields and options:

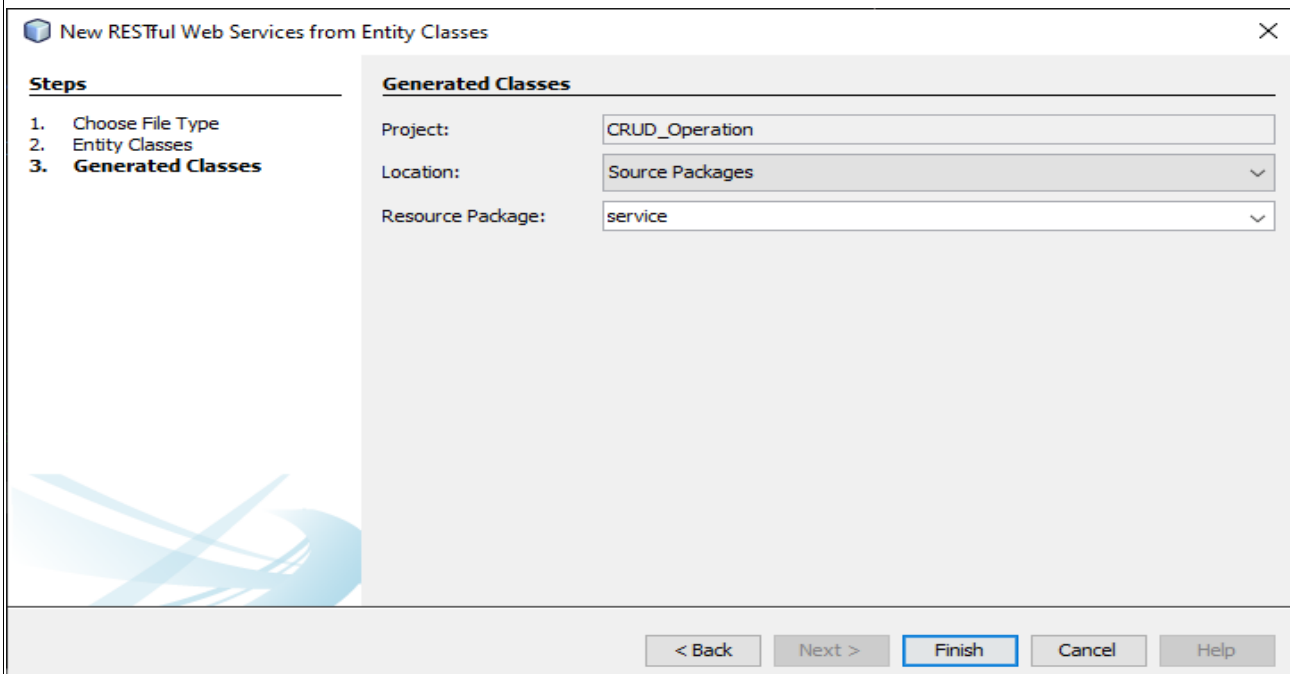
- Specify JavaServer Faces configuration and library information.
- Libraries (selected tab), Configuration, Components
- ☒ Server Library: JSF 2.2 (dropdown)
- ☐ Registered Libraries: JSF 2.2 (dropdown)
- ☐ Create New Library
 - JSF Folder or JAR: (empty text field) with a 'Browse...' button.
 - Library Name: (empty text field)

At the bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

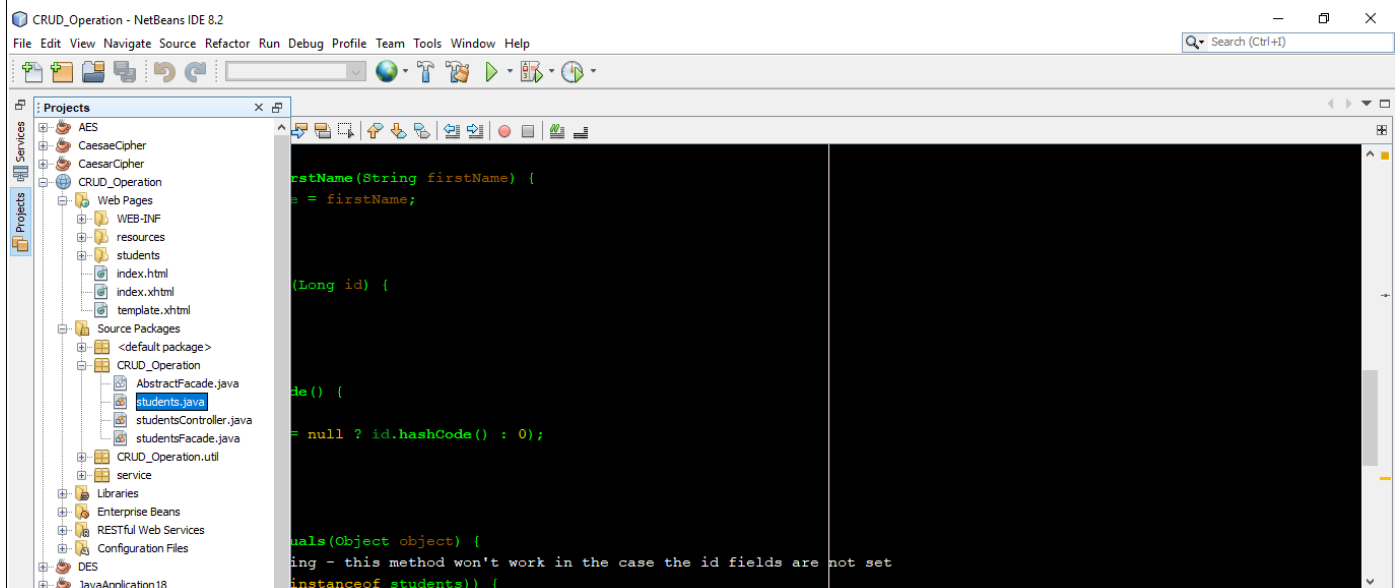
- 11) Right click on project name and create RESTful Web Services from Entity Classes.
Right click on project name -> New -> RESTful Web Services from Entity Classes



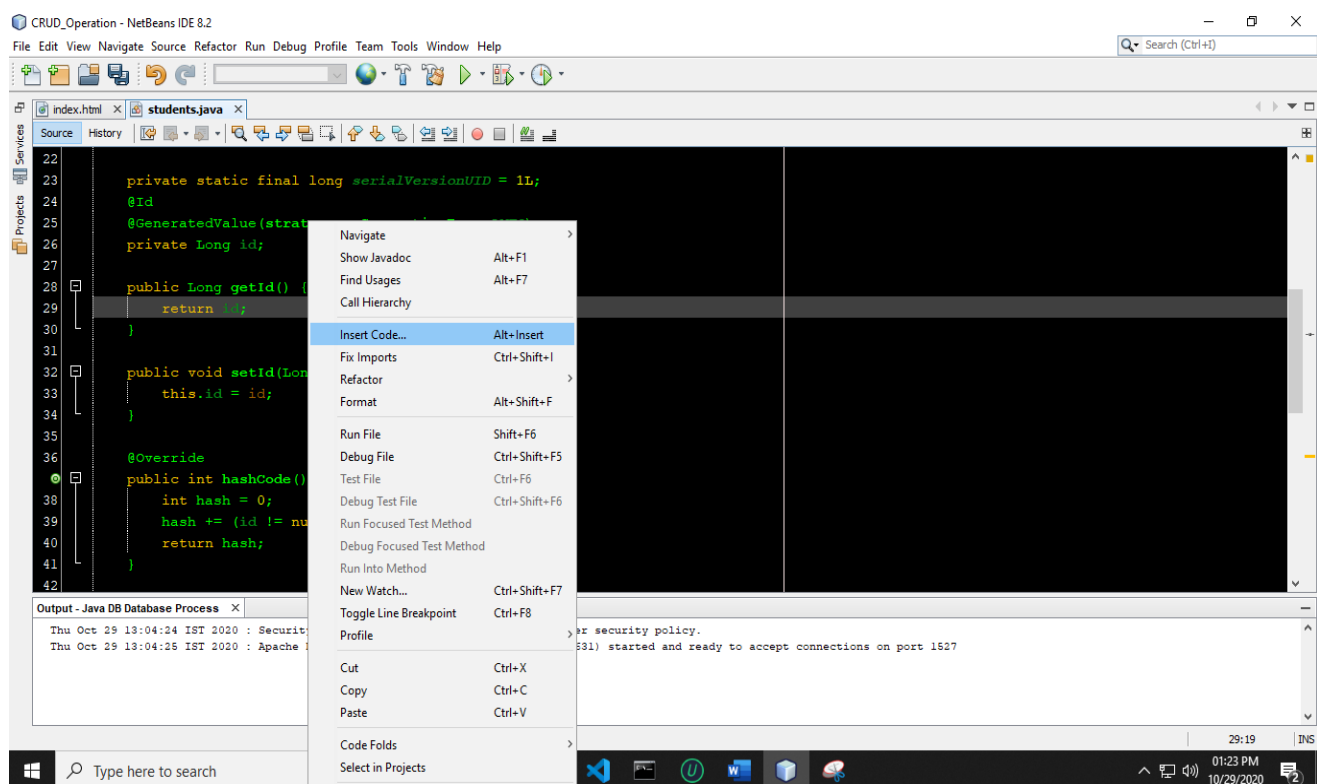
- 12) Repeat step 9 and then it will go on next page. Then **enter the service in Resource Package** and then **click on Finish button**.



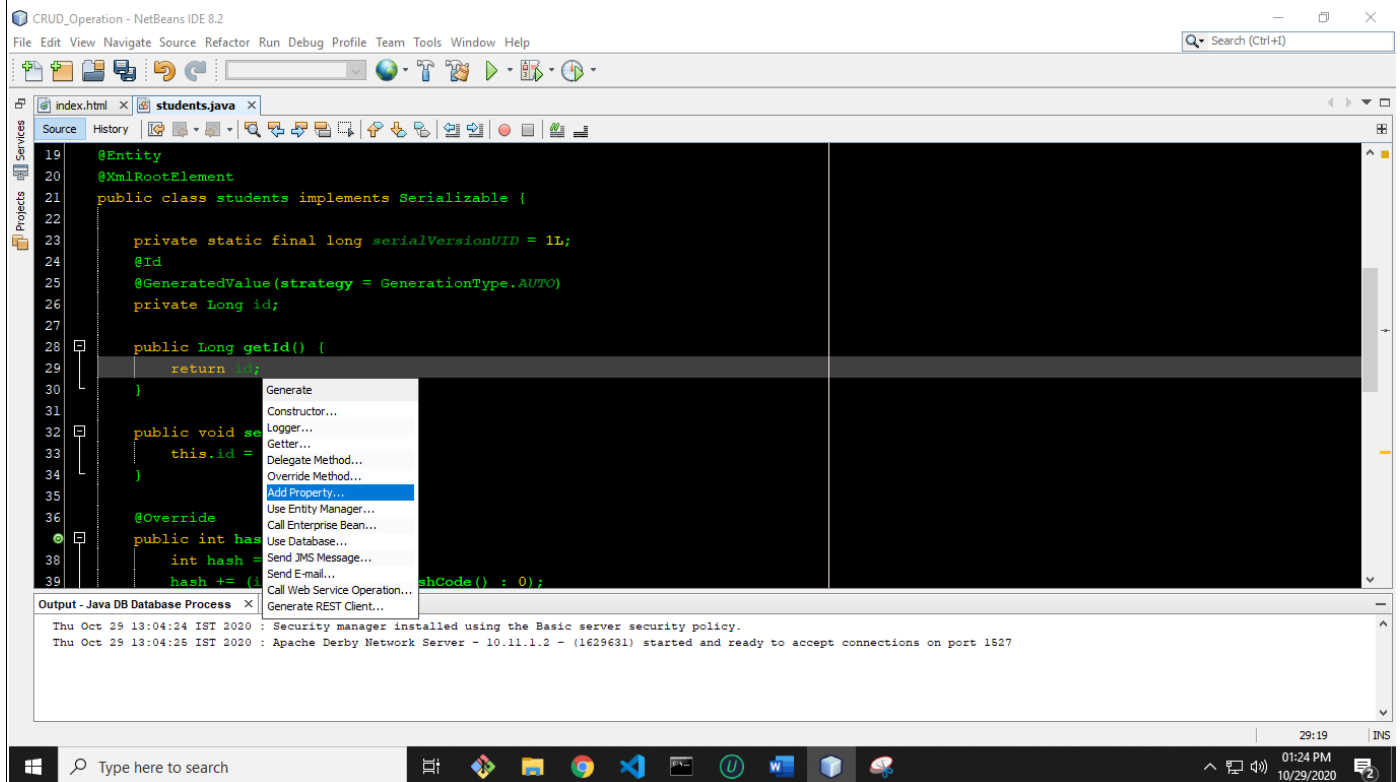
13) Now open students.java file under wspract8 package.



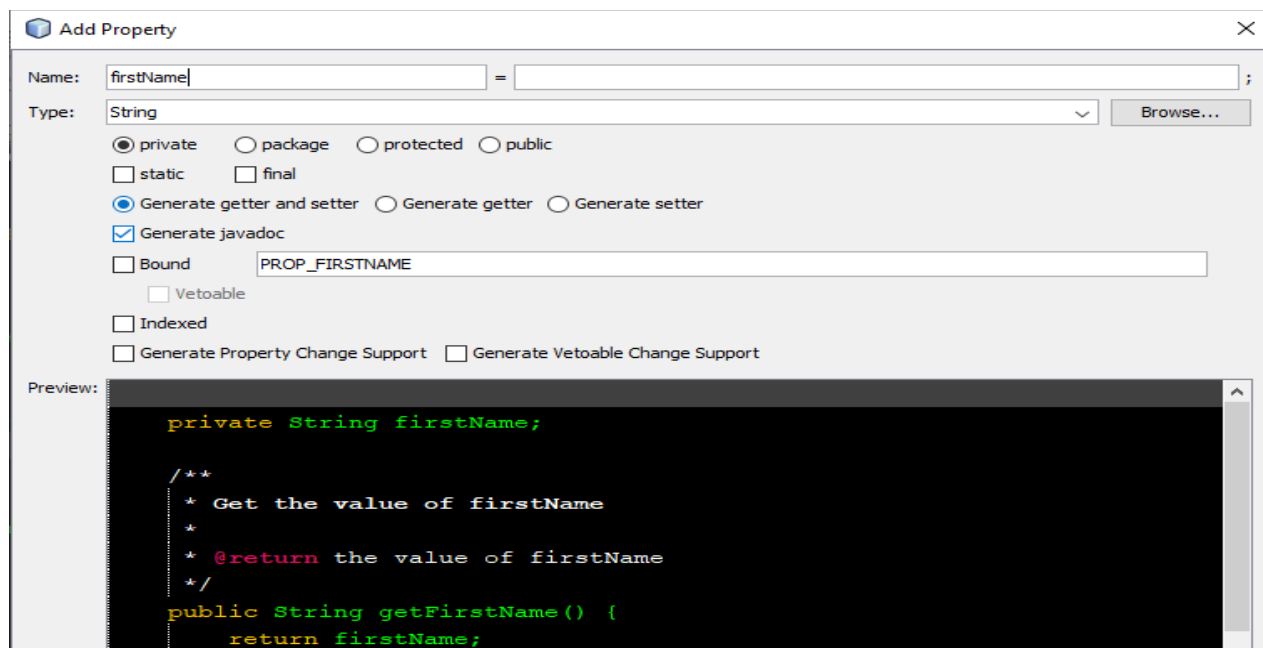
14) In this file below line private Long id; do the right click and select Insert Code.



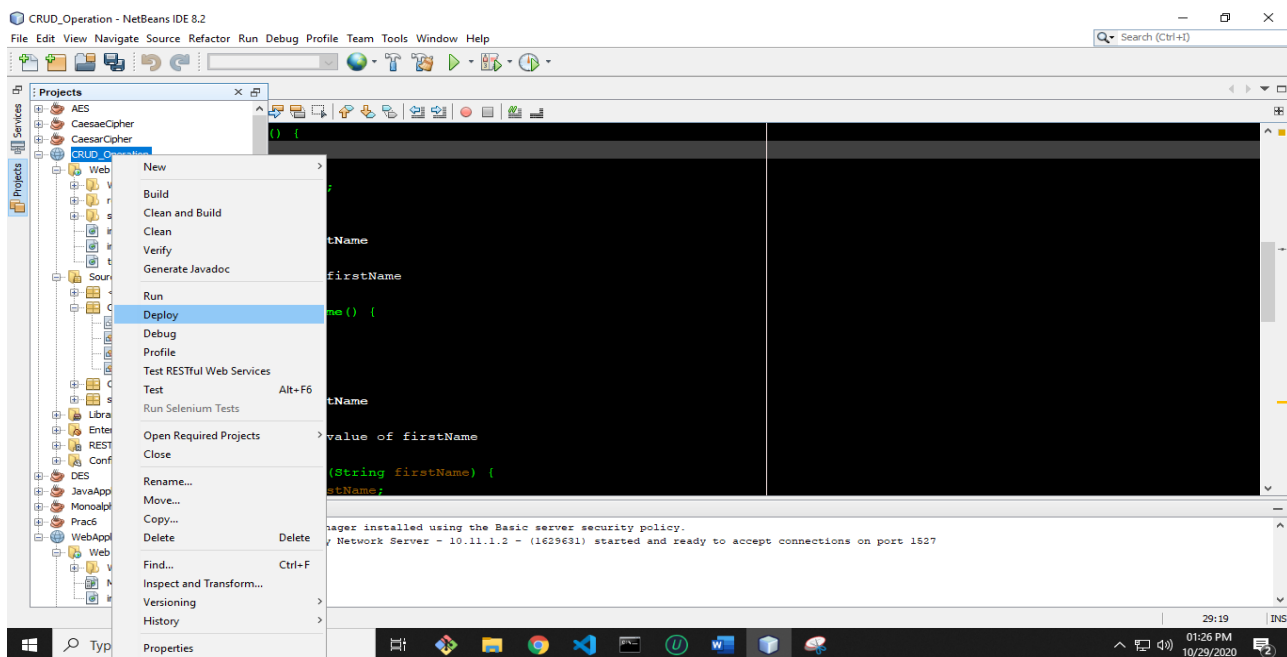
15) A new list will appear. Click on Add Property.



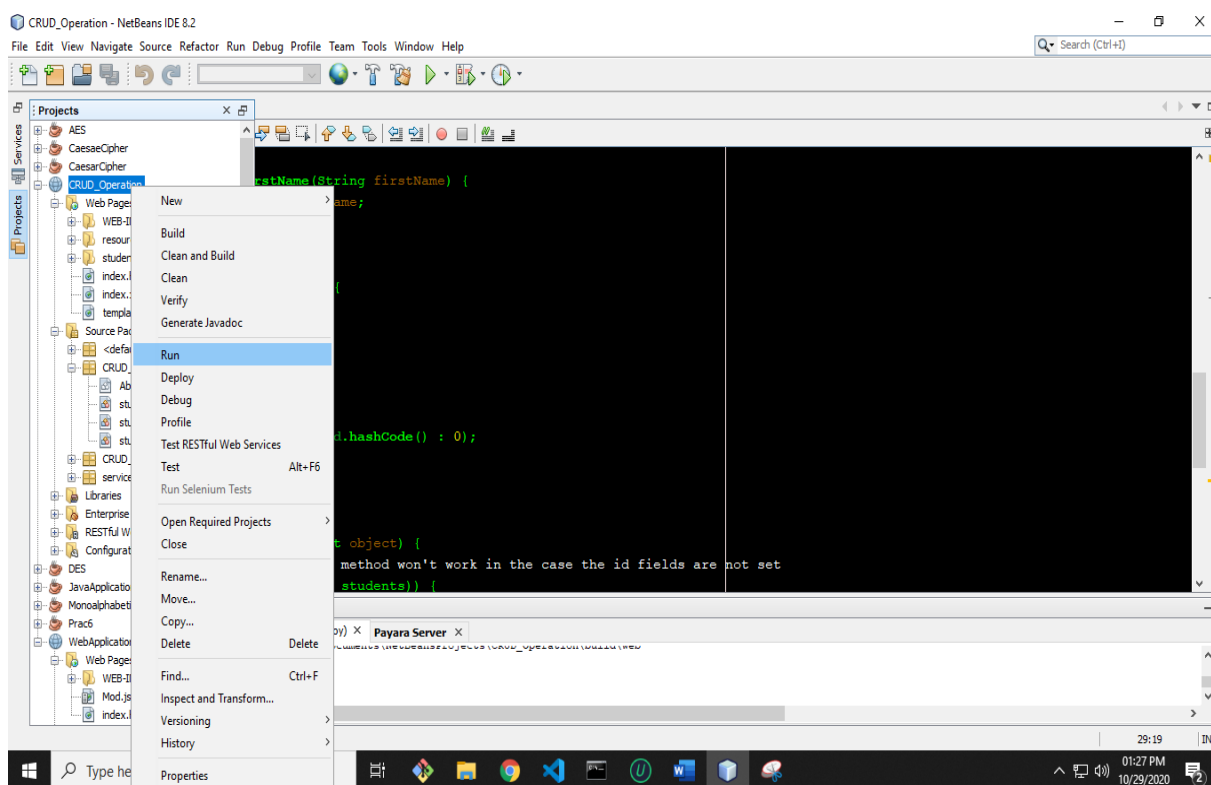
16) A new window will open. Enter name as **firstName**. Make sure name should be exact same as of mine and then click on OK button. Actually we are setting getter and setter method for firstName.



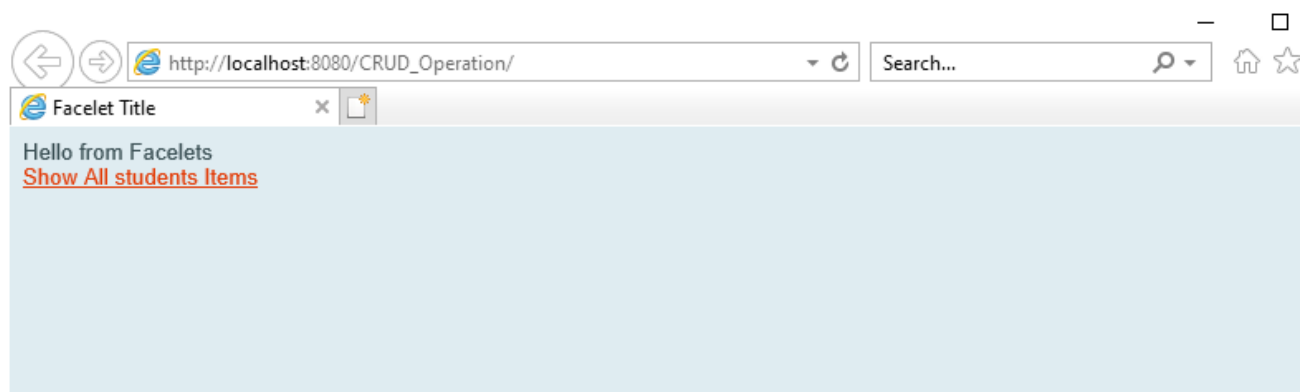
17) Now right click on web application name and Deploy it.



18) Now right click on project name and run it.

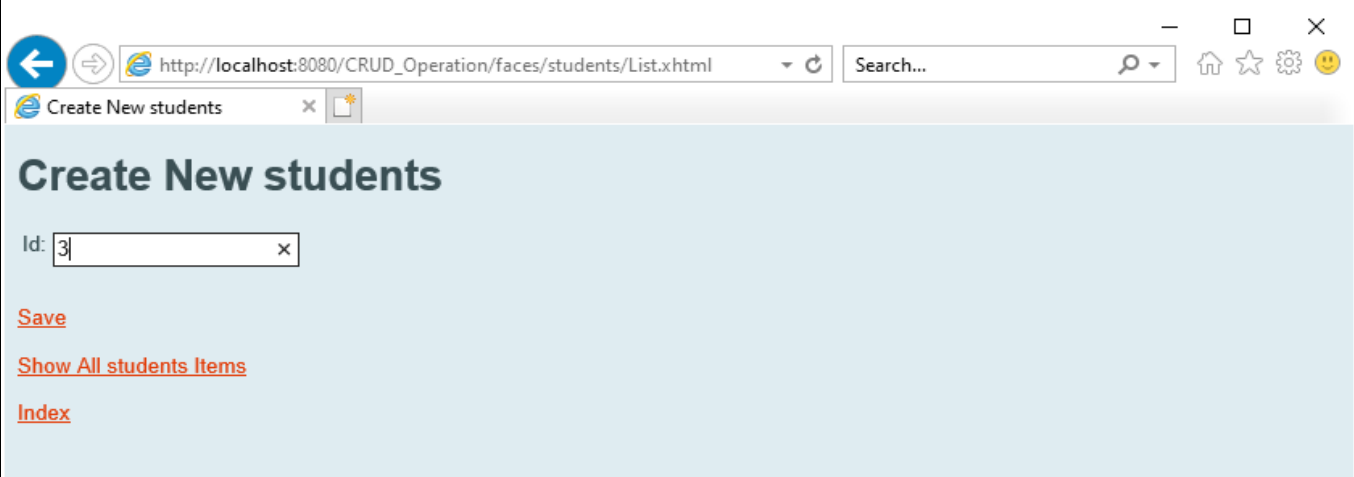
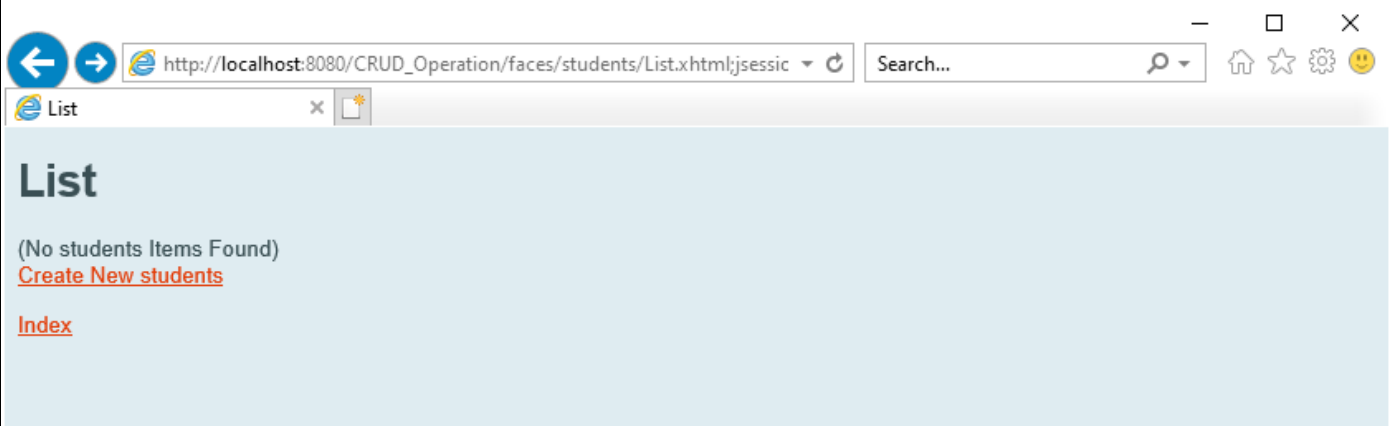


19) A window will open in browser like below.

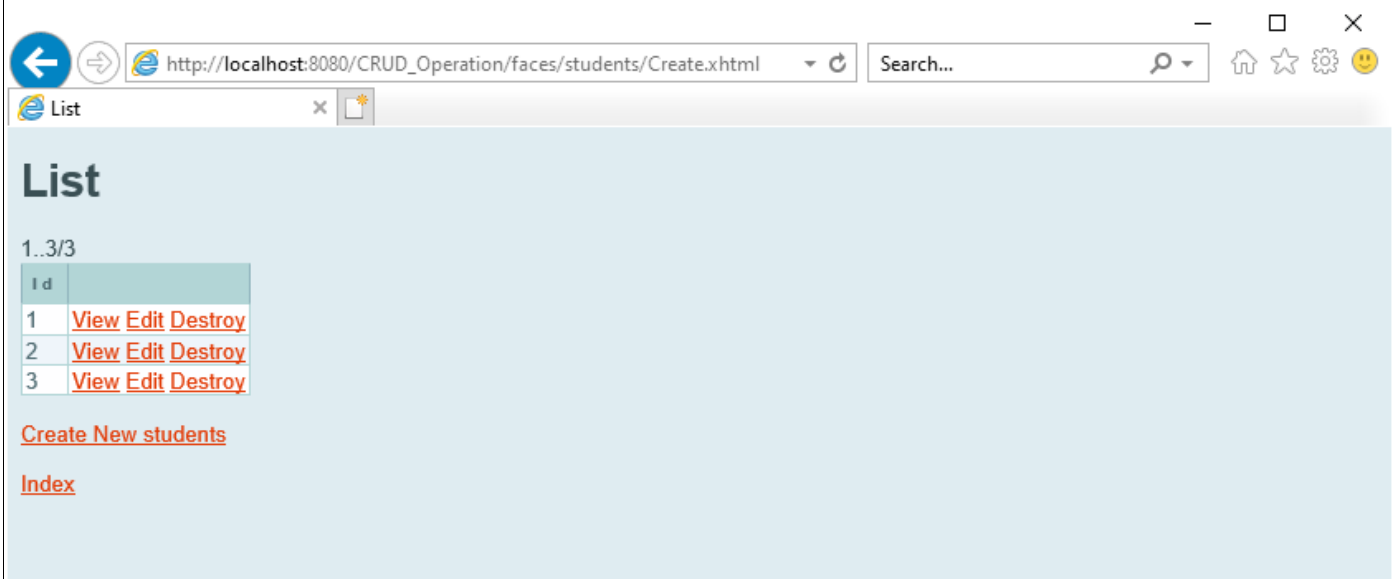


20) Now **click on Show All students** Items for CRUD operation.

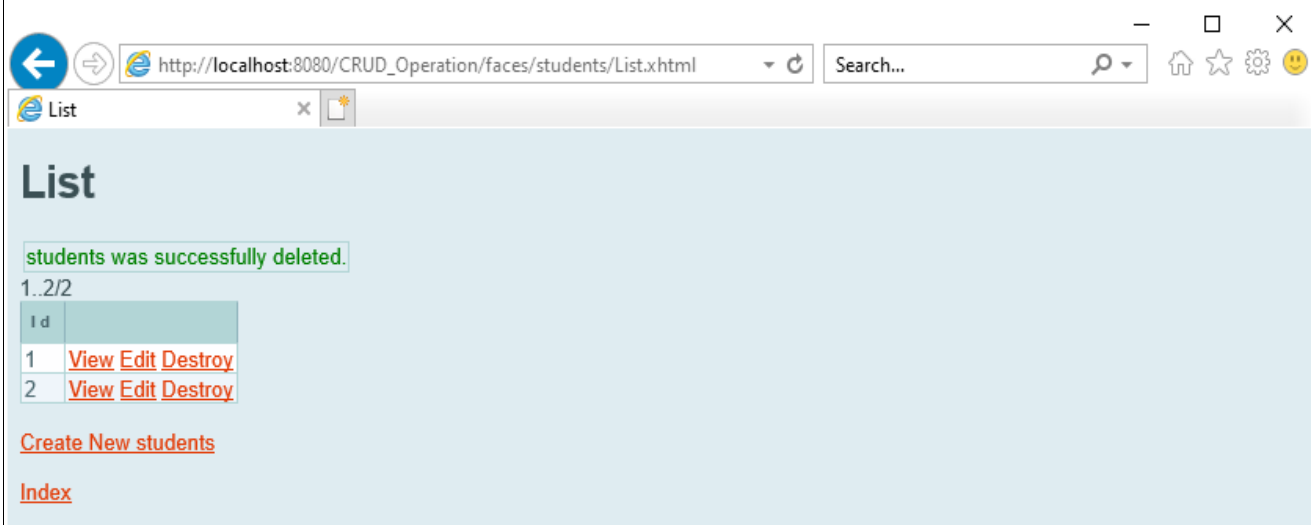
Just click on Create New students. **Enter a name into FirstName** and **id into Id**. Now **click on Save** option to save the data.



21) Now click on Show All students Items to view all records whether our data is entered or not.



22) Delete the Record



Date: 18/11/2020

Practical no 9

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

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.

Steps :-

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

New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

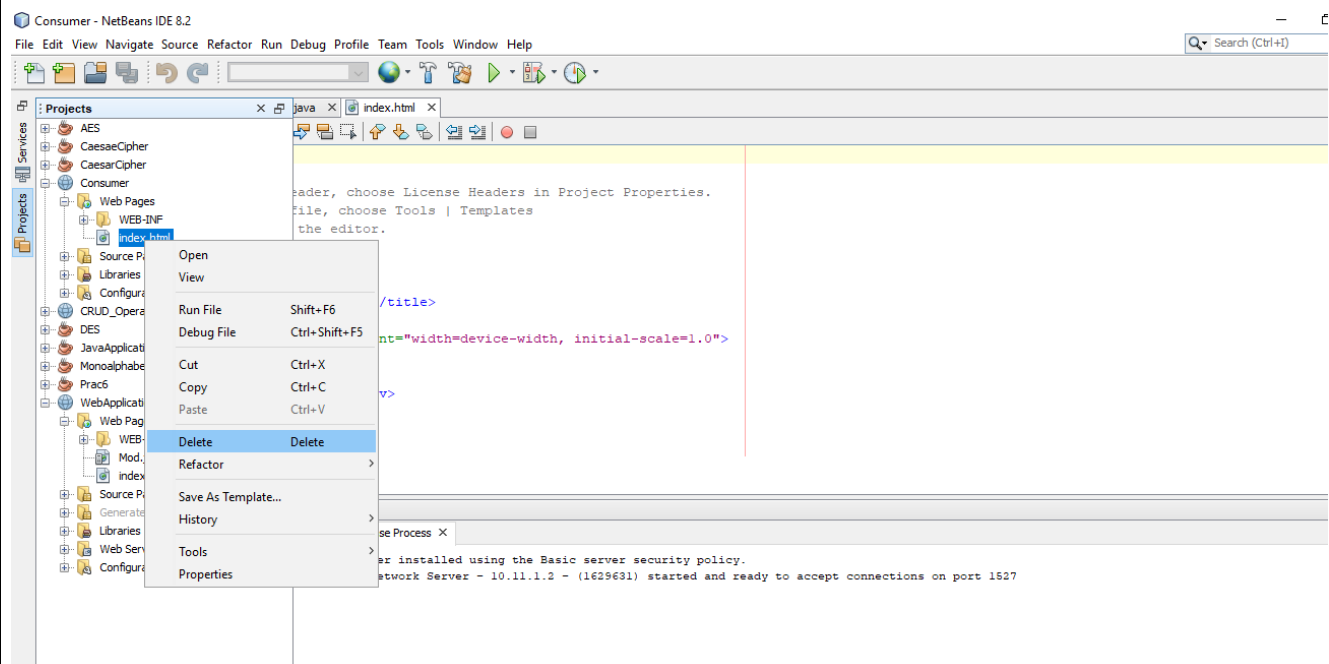
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

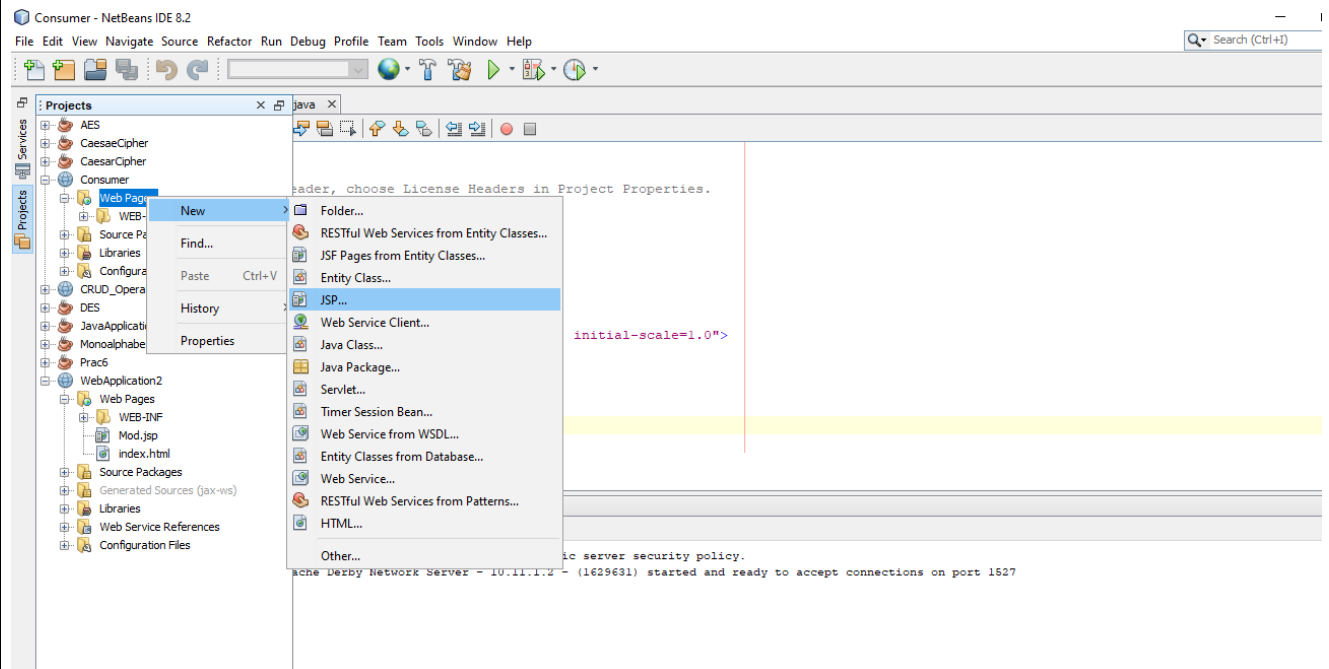
< Back Next > Finish Cancel Help

2) And create it. Next Finish.

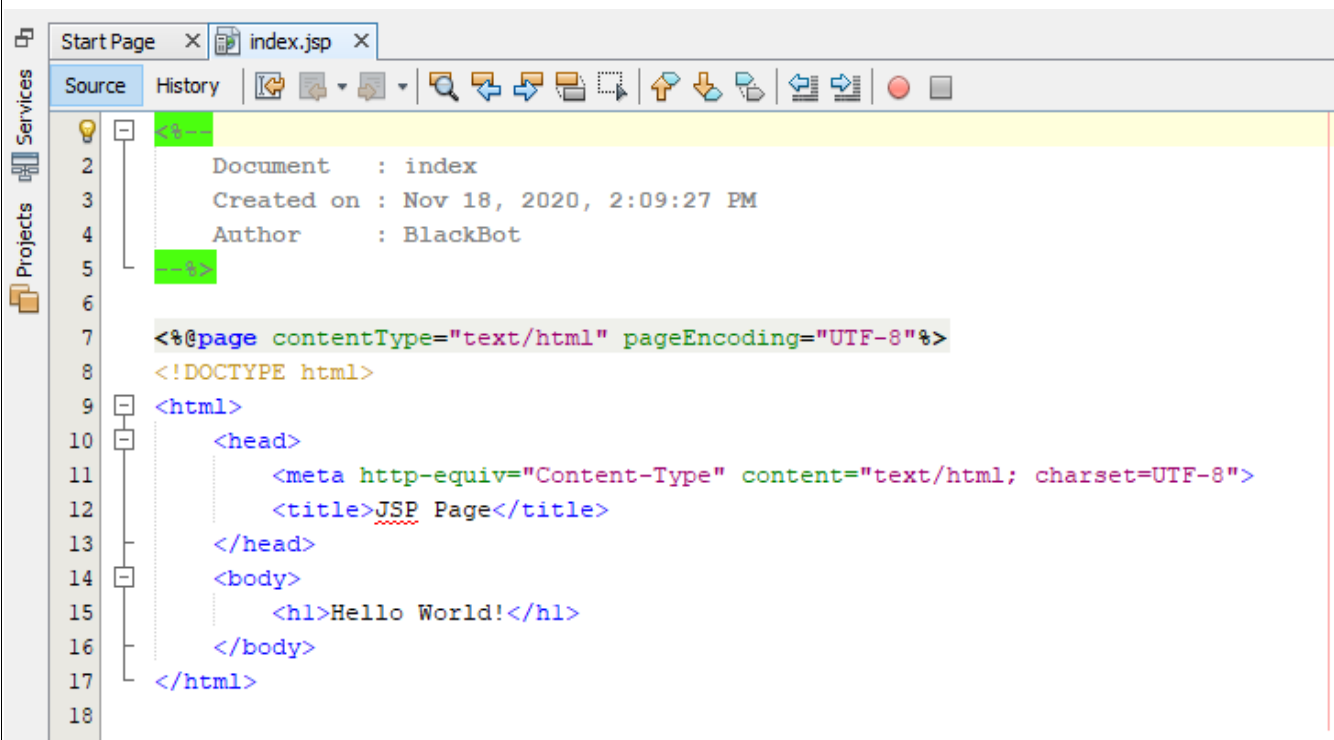
3) Now right click on index.html and delete it.



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.

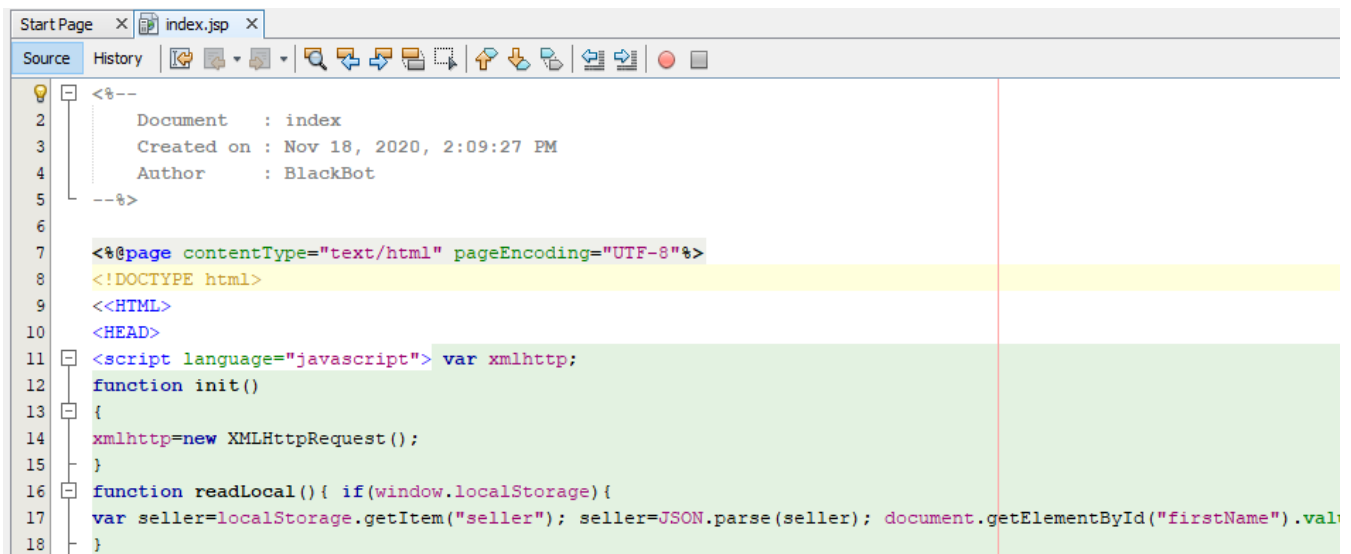


- 7) Now select HTML content of index.jsp file and replace it with following bold letter codes.

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<<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");
  
```

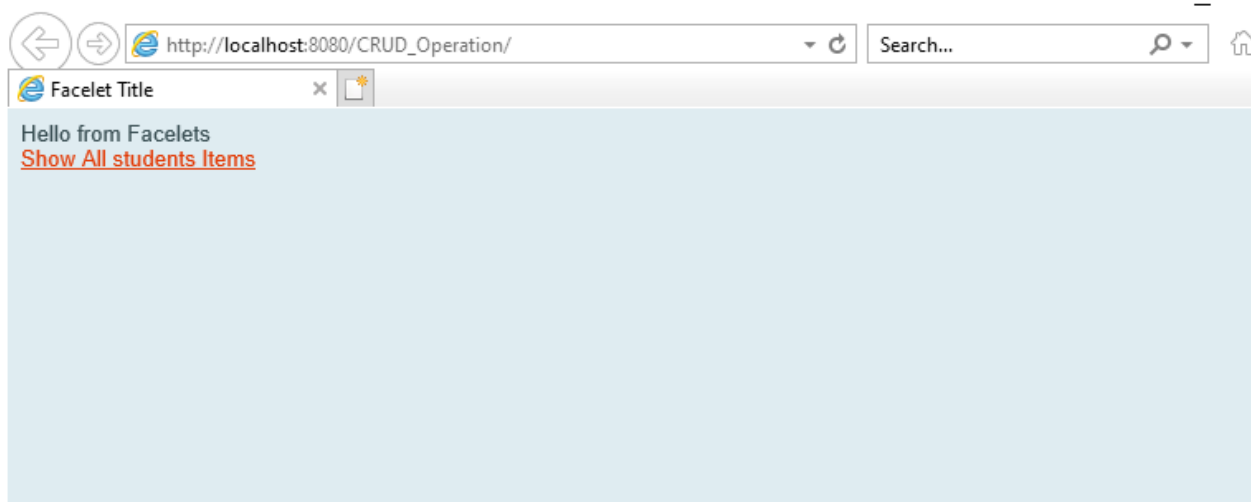
```
var url="http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/"+ sellerid.value;
xmlhttp.open('GET',url,true); xmlhttp.send(null);
xmlhttp.onreadystatechange =function() {
if(xmlhttp.readyState===4){alert("6"+sellerid);
if(xmlhttp.status===200){alert("7"+sellerid);
var seller =eval("(" +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>
<tr>
<td colspan="2">performed by krunal 713
</td>
</tr>
</table>
</body>
</html>
```



```
1 <!--
2     Document   : index
3     Created on  : Nov 18, 2020, 2:09:27 PM
4     Author      : BlackBot
5 -->
6
7 <?xml contentType="text/html" pageEncoding="UTF-8"?>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <script language="javascript"> var xmlhttp;
12 function init()
13 {
14     xmlhttp=new XMLHttpRequest();
15 }
16 function readLocal(){ if(window.localStorage){
17     var seller=localStorage.getItem("seller"); seller=JSON.parse(seller); document.getElementById("firstName").value=
18 }
```

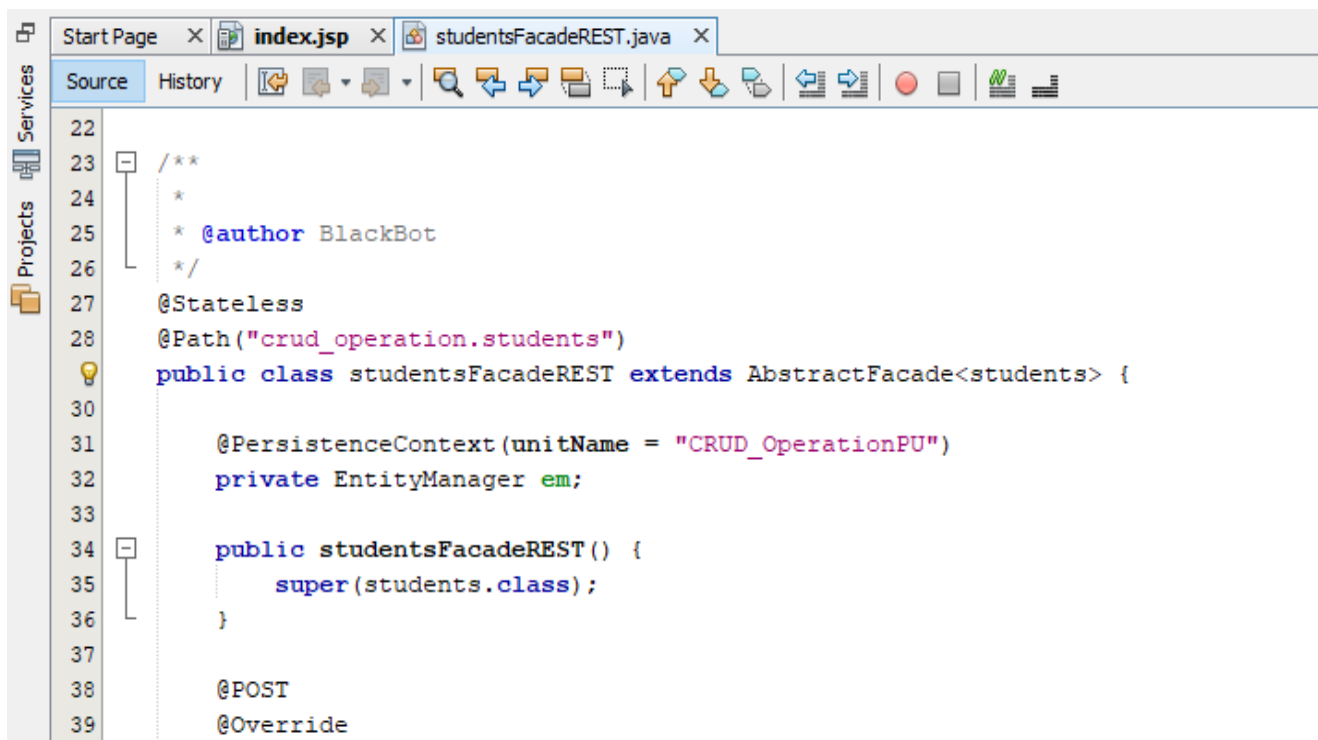
9) http://localhost:8080/CRUD_Operation/webresources/com.kk.seller/

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



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

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

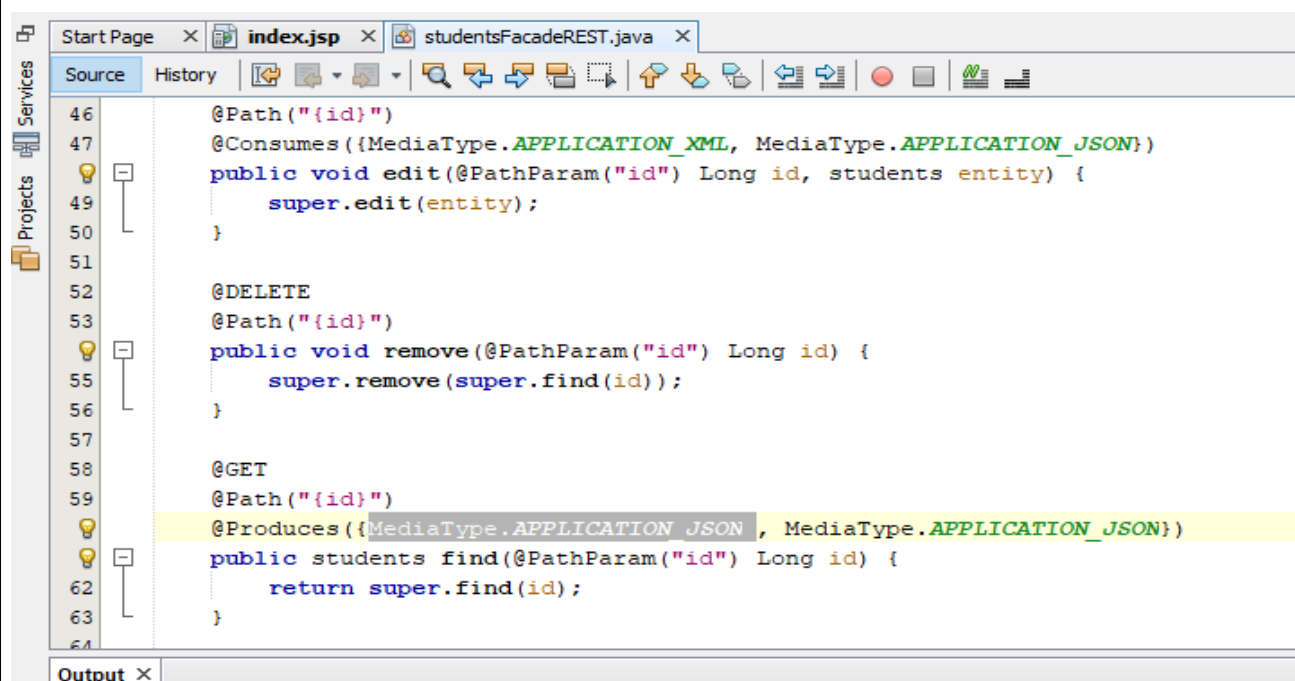


```
22
23 /**
24  *
25  * @author BlackBot
26  */
27 @Stateless
28 @Path("crud_operation.students")
29 public class studentsFacadeREST extends AbstractFacade<students> {
30
31     @PersistenceContext(unitName = "CRUD_OperationPU")
32     private EntityManager em;
33
34     public studentsFacadeREST() {
35         super(students.class);
36     }
37
38     @POST
39     @Override
```

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.



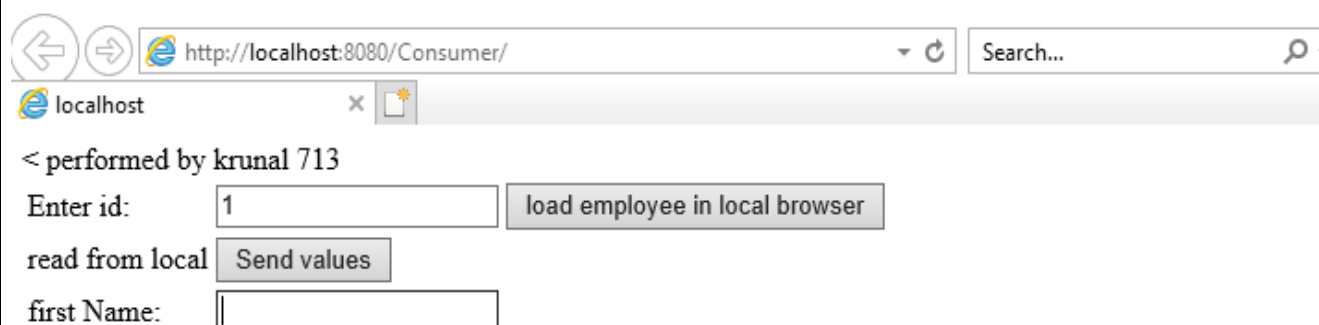
```

46  @Path("{id}")
47  @Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
48  public void edit(@PathParam("id") Long id, students entity) {
49      super.edit(entity);
50  }
51
52  @DELETE
53  @Path("{id}")
54  public void remove(@PathParam("id") Long id) {
55      super.remove(super.find(id));
56  }
57
58  @GET
59  @Path("{id}")
60  @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
61  public students find(@PathParam("id") Long id) {
62      return super.find(id);
63  }
64

```

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.



< performed by krunal 713

Enter id:

read from local

first Name: