MALAD KANDIVALI EDUCATION SOCIETY'S
NAGINDAS KHANDWALA COLLEGE OF COMMERCE,
ARTS & MANAGEMENT STUDIES & SHANTABEN NAGINDAS
KHANDWALA COLLEGE OF SCIENCE
MALAD [W], MUMBAI – 64
(AUTONOMOUS)

(Reaccredited 'A' Grade by NAAC)
(AFFILIATED TO UNIVERSITY OF MUMBAI)
(ISO 9001:2015)

CERTIFICATE

Name: **Mr. Kuldeep Shushil Patel**

Roll No: **574** **Programme**: BSc CS **Semester**: V

This is certified to be a bonafide record of practical works done by the above student in the college laboratory for the course **Web Services** (Course Code: **1856UCSPR**) for the partial fulfillment of Fifth Semester of BSc CS during the academic year 2020-2021.

The journal work is the original study work that has been duly approved in the year 2020-2021 by the undersigned.

External Examiner **Ms. ANISHA ASIRVATHAM**
(Subject-In-Charge)

Date of Examination: **(College Stamp)**

Name: Kuldeep Shushil Patel **Roll No: 574**

Subject: Web Services Practical (1856UCSPR)

Class: T.Y.B.SC (CS) **SEMESTER-V**

NO	DATE	TITLE	SIGN
1.	10/08/2020	Write a program to implement to create a simple web service that converts the temperature from Fahrenheit to Celsius and vice a versa.	
2.	17/08/2020	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	
3.	24/08/2020	Develop client which consumes web services developed in different platform.	
4.	07/09/2020	Write a JAX-WS web service to perform the following operations. Define a Servlet / JSP that consumes the web service.	
5.	14/09/2020	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.	
6.	21/09/2020	Define a RESTful web service that accepts the details to be stored in a database and performs CRUD operation.	
7.	28/09/2020	Implement a typical service and a typical client using WCF.	
8.	05/10/2020	. Use WCF to create a basic ASP.NET Asynchronous JavaScript and XML (AJAX) service.	
9.	12/10/2020	Demonstrates using the binding attribute of an endpoint element in WCF.	

Practical : 1

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

Theory:

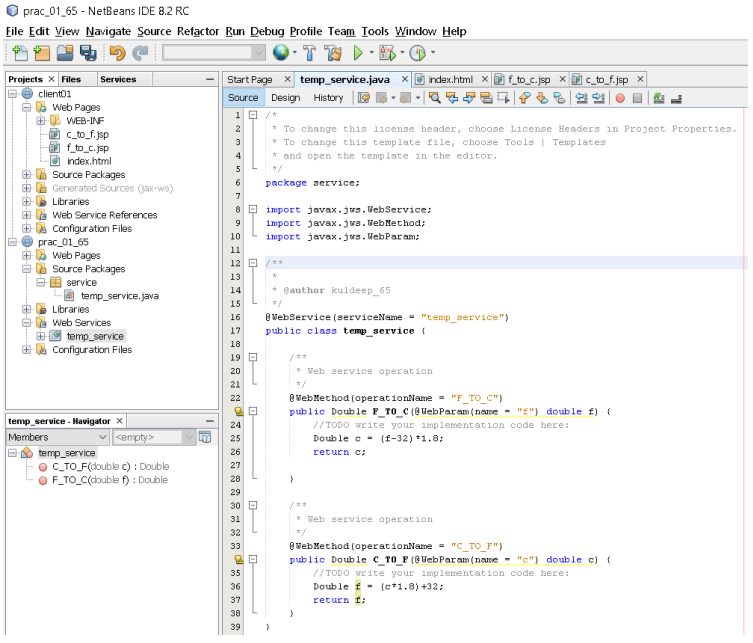
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains.
- These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service.
- For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. ➢
- Celsius to Fahrenheit Conversion Formula

  $$°F = (°C \times 9/5) + 32$$
- Fahrenheit to Celsius Conversion Formula

  $$°C = (°F - 32) \times 5/9$$

Code:

Step1 : created the temp_service.java file and edited the server side.



Step 2 : created the jsp file (c_to_f.jsp) and edited the client side.

client01 - NetBeans IDE 8.2 RC

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

Projects | Files | Services

- client01
  - Web Pages
    - WEB-INF
    - c_to_f.jsp
    - f_to_c.jsp
    - index.html
  - Source Packages
  - Generated Sources (jax-ws)
  - Libraries
  - Web Service References
  - Configuration Files
- prac_01_65
  - Web Pages
  - Source Packages
    - service
      - temp_service.java
  - Libraries
  - Web Services
    - temp_service
  - Configuration Files

Navigator

- html
  - head
    - meta
    - title
  - body
    - hr
    - hr

Start Page | temp_service.java | index.html | f_to_c.jsp | c_to_f.jsp

Source | History

```
<%--
    Document   : c_to_f
    Created on : 26 Sep, 2020, 5:03:58 PM
    Author     : kuldeep
--%>

<%@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/>
    <%
        String d=request.getParameter("data");
        Integer dd=Integer.parseInt(d);

    try {
        com.dd.TempService_Service service = new com.dd.TempService_Service();
        com.dd.TempService port = service.getTempServicePort();
         // TODO initialize WS operation arguments here
        double c = dd;
        // TODO process result here
        java.lang.Double result = port.cTOF(c);
        out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

Step 3: created the jsp file (f_to_c.jsp) and again edited the client side.

client01 - NetBeans IDE 8.2 RC

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

Projects | Files | Services

- client01
  - Web Pages
    - WEB-INF
    - c_to_f.jsp
    - f_to_c.jsp
    - index.html
  - Source Packages
  - Generated Sources (jax-ws)
  - Libraries
  - Web Service References
  - Configuration Files
- prac_01_65
  - Web Pages
  - Source Packages
    - service
      - temp_service.java
  - Libraries
  - Web Services
    - temp_service
  - Configuration Files

Navigator

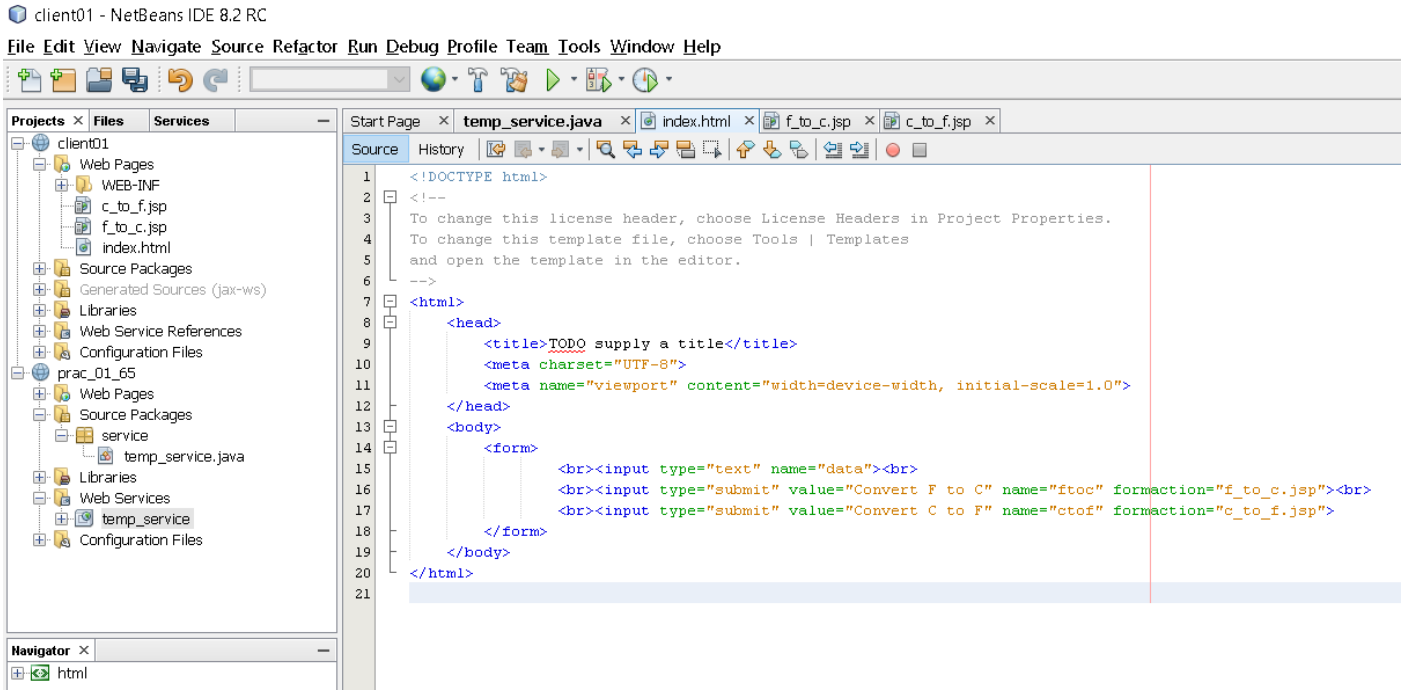
- html
  - head
    - meta
    - title
  - body
    - hr
    - hr

Start Page | temp_service.java | index.html | f_to_c.jsp | c_to_f.jsp

Source | History

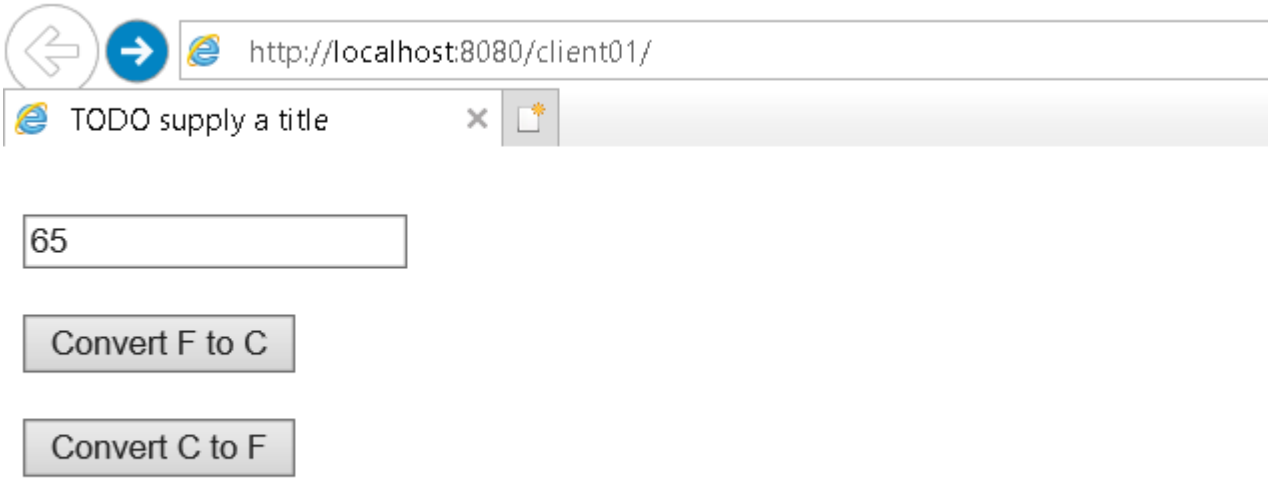
```
<%--
    Document   : f_to_c
    Created on : 26 Sep, 2020, 5:03:35 PM
    Author     : kuldeep
--%>

<%@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/>
    <%
        String d=request.getParameter("data");
        Integer dd=Integer.parseInt(d);
    try {
        com.dd.TempService_Service service = new com.dd.TempService_Service();
        com.dd.TempService port = service.getTempServicePort();
         // TODO initialize WS operation arguments here
        double f = dd;
        // TODO process result here
        java.lang.Double result = port.fTOC(f);
        out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

Step 4 : created html file for client side.

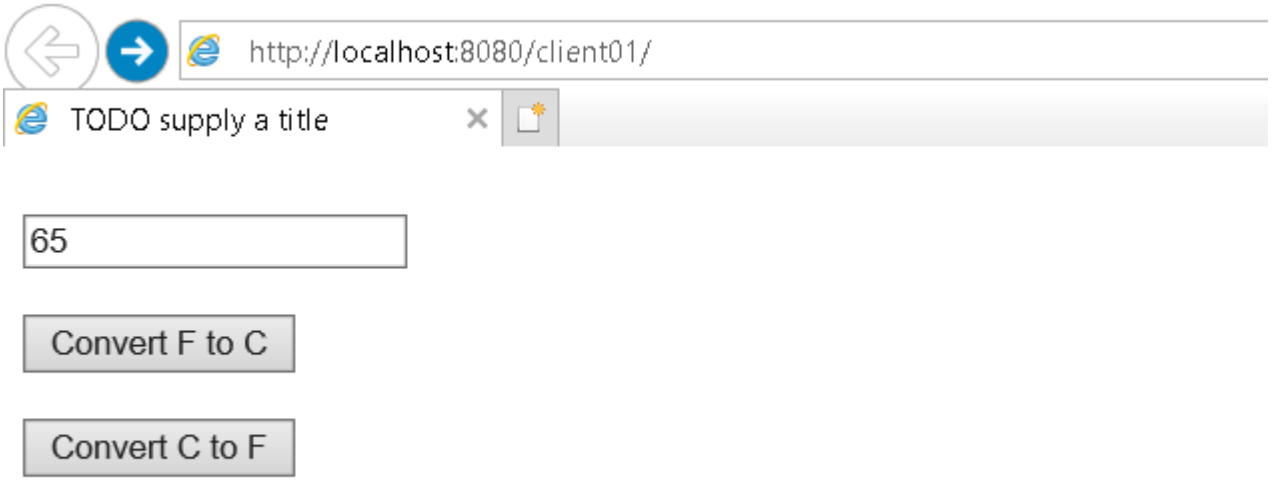


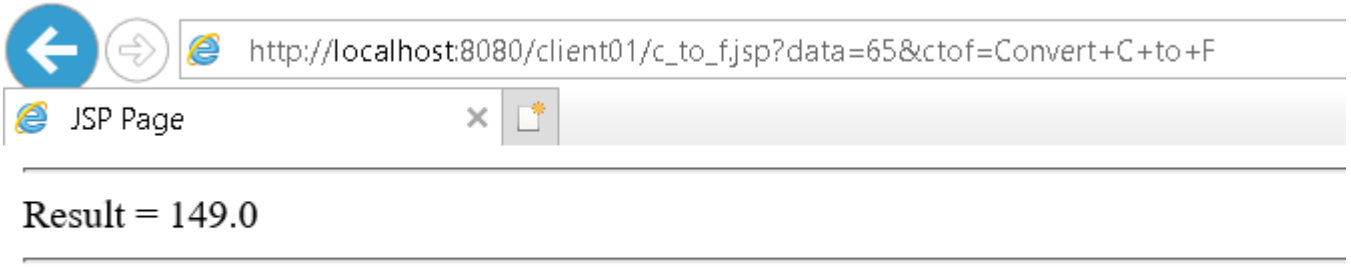
Output :

For converting f_to_c



For converting c_to_f





Practical : 2 / 5

Aim : write a program to implement the operation can receive request and will return a response in two ways

Theory :

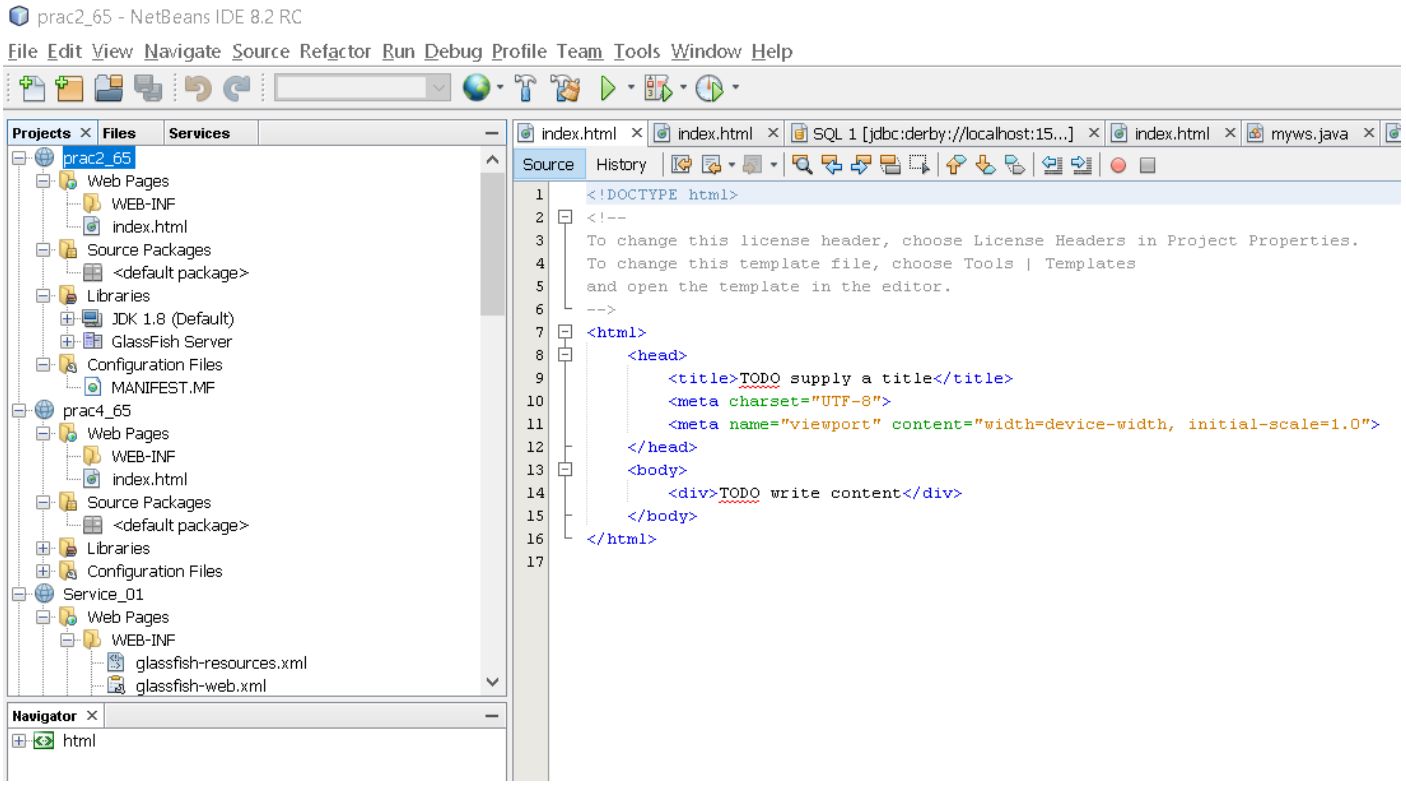
- WSDL stands for Web Services Description Language. ➢ WSDL is used to describe web services.
- WSDL is written in XML
- WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services)..
- One-way Operation The grammar for a one-way operation is: * The input element specifies the abstract message format for the one-way operation.
- Request-response Operation The grammar for a request-response operation is:
- The input and output elements specify the abstract message format for the request and response, respectively. The optional fault elements specify the abstract message format for any error messages that may be output as the result of the operation.

A] One – Way operation

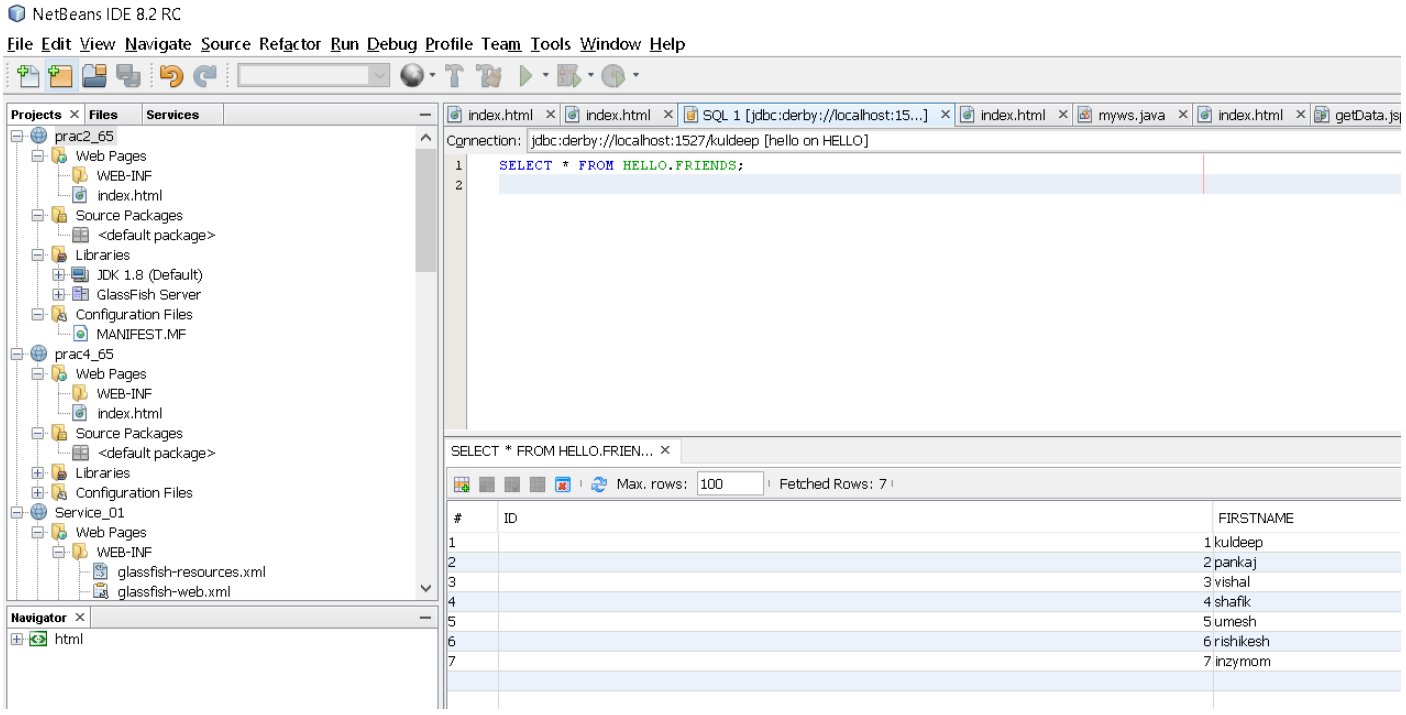
B] Request – Response

Code :

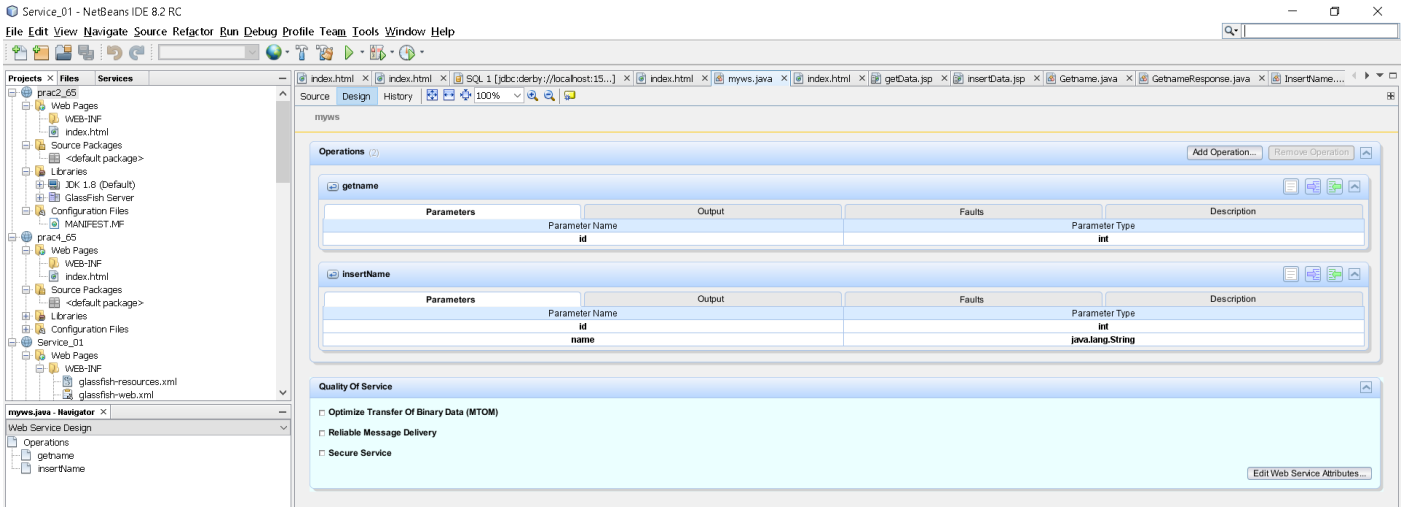
Step 1 : created the web application name prac2_65



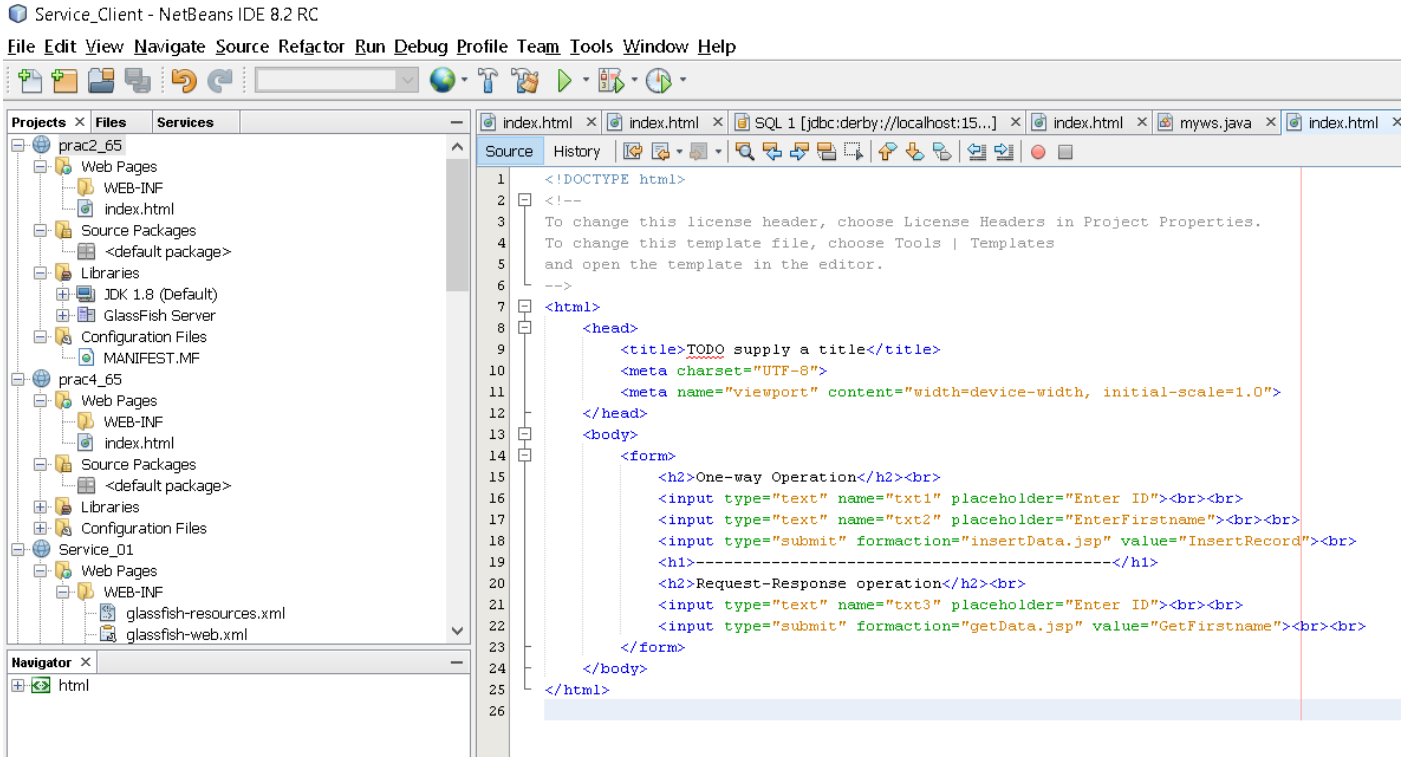
Step 2 : created the table under kuldeep database as shown below



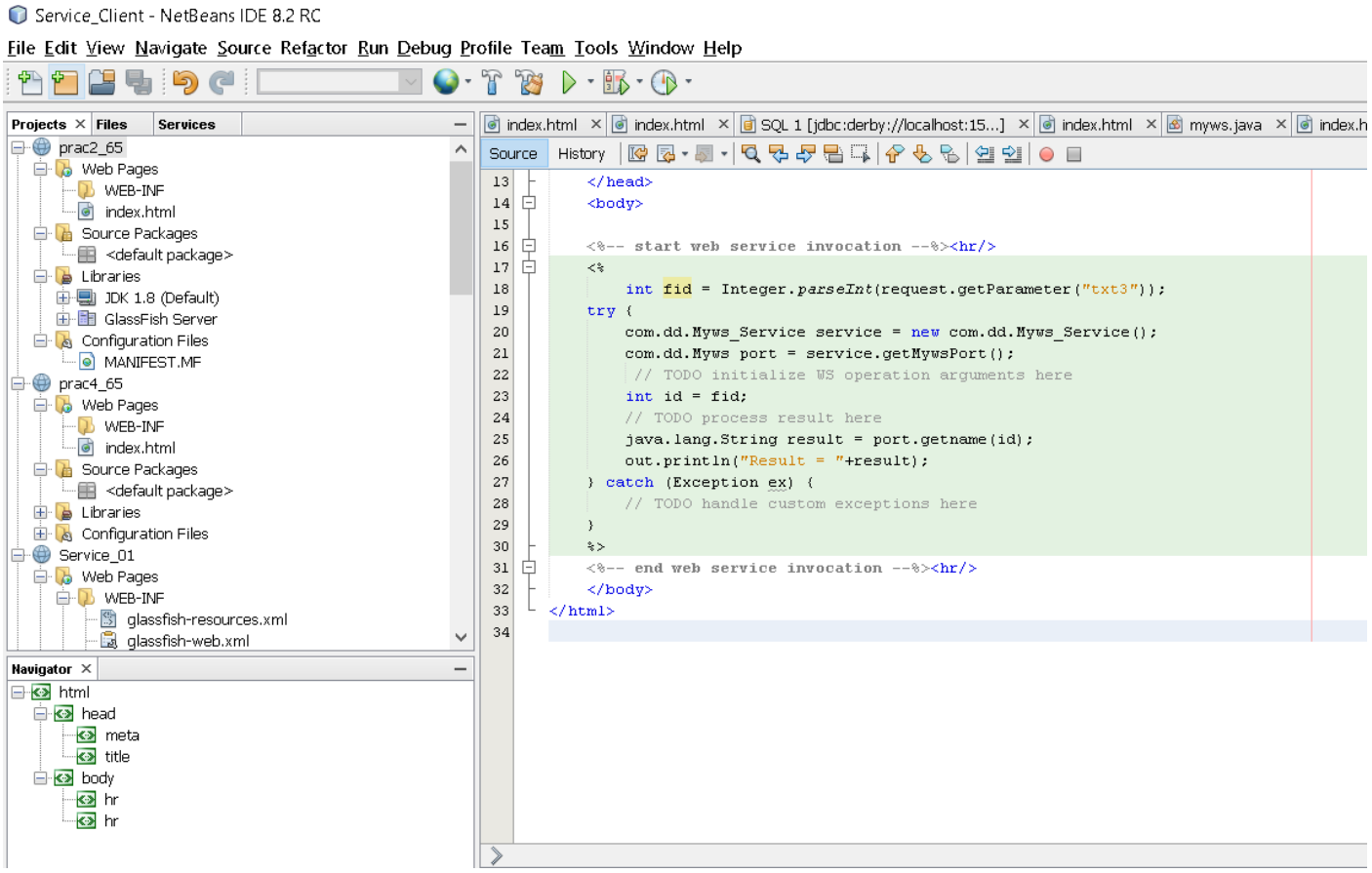
Step 3 : given the parameter Id in INTEGER and NAME in VARHCHAR



Step 4: calling the function in file index.html



Step 5 : created the jsp file insertdata.jsp



Step 6 : the author name is kuldeep

prac2_65 - NetBeans IDE 8.2 RC

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

Projects | Files | Services

- prac2_65
  - Web Pages
    - WEB-INF
    - index.html
  - Source Packages
    - <default package>
  - Libraries
    - JDK 1.8 (Default)
    - GlassFish Server
  - Configuration Files
    - MANIFEST.MF
- prac4_65
  - Web Pages
    - WEB-INF
    - index.html
  - Source Packages
    - <default package>
  - Libraries
  - Configuration Files
- Service_01
  - Web Pages
    - WEB-INF
      - glassfish-resources.xml
      - glassfish-web.xml

Navigator

Members | <empty>

- Getname
  - getId() : int
  - setId(int value)
  - id : int

insertData.jsp

Source | History

```
<%--
    Document   : insertDate
    Created on : 27 Sep, 2020, 5:33:37 PM
    Author     : kuldeep
--%>

<%@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/>
    <%
        int fid = Integer.parseInt(request.getParameter("txt1"));
        String fname = request.getParameter("txt2");


    try {
        com.dd.Myws_Service service = new com.dd.Myws_Service();
        com.dd.Myws port = service.getMywsPort();
         // TODO initialize WS operation arguments here
        int id = fid;
        java.lang.String name = "fnam";
        // TODO process result here
        java.lang.String result = port.insertName(id, name);
        out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

Service_Client - NetBeans IDE 8.2 RC

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

Projects | Files | Services

- sunmscapi.jar
- sunpkcs11.jar
- zipfs.jar
- GlassFish Server
- Test Libraries
- Web Services
  - myws
- Configuration Files
- Server Resources
- Service

New
Build
Clean and Build
Clean
Verify
Generate Javadoc
Run
Deploy
Debug
Profile
Test RESTful Web Services
Test Alt+F6
Run Selenium Tests
Open Required Projects
Close
Rename...
Move...
Copy...
Delete Delete
Find... Ctrl+F
Inspect and Transform...
Versioning
History
Properties

Navigator

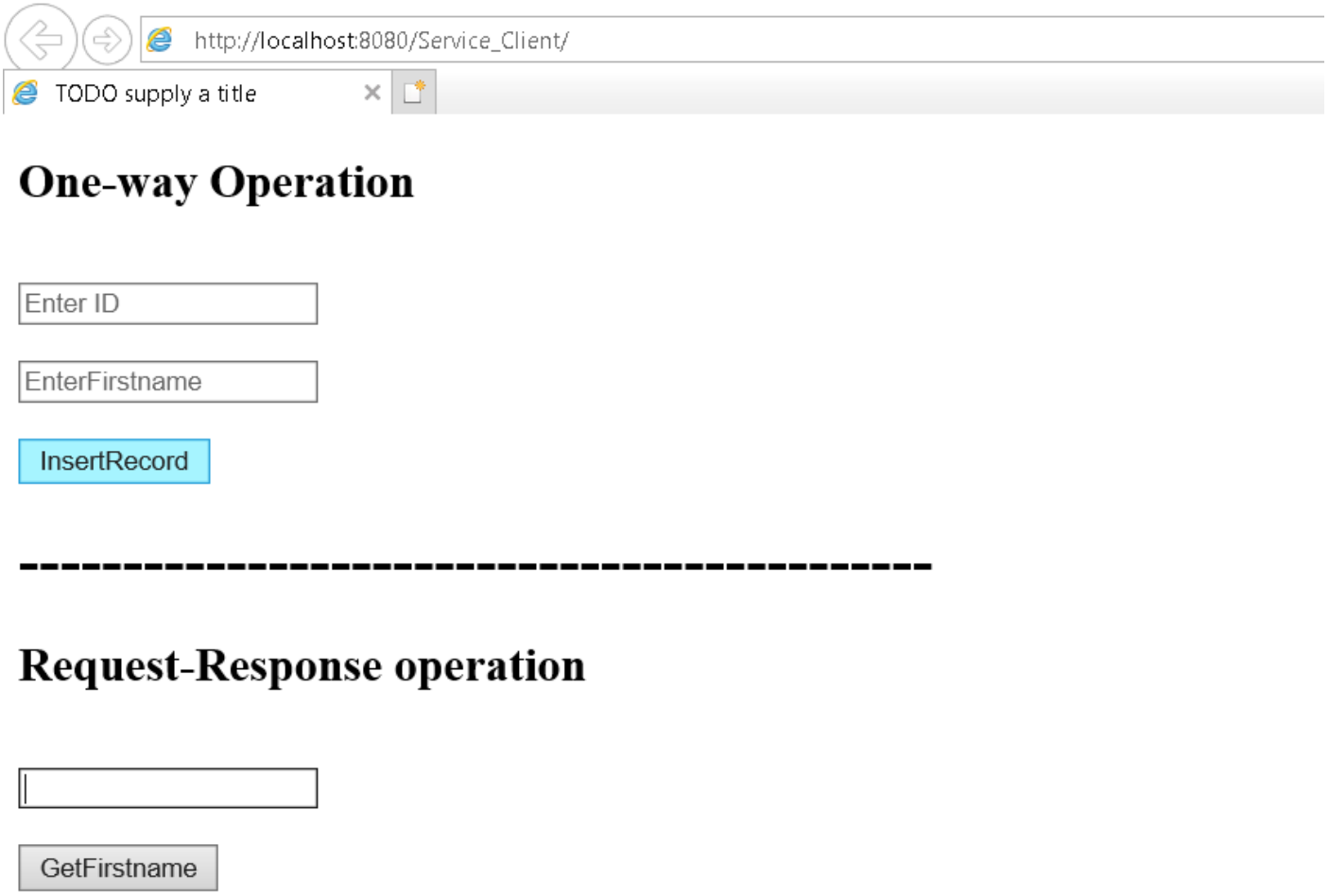
Members

...java | index.html | getData.jsp | Getname.java | GetnameResponse.java | InsertName.java | InsertNameRespo

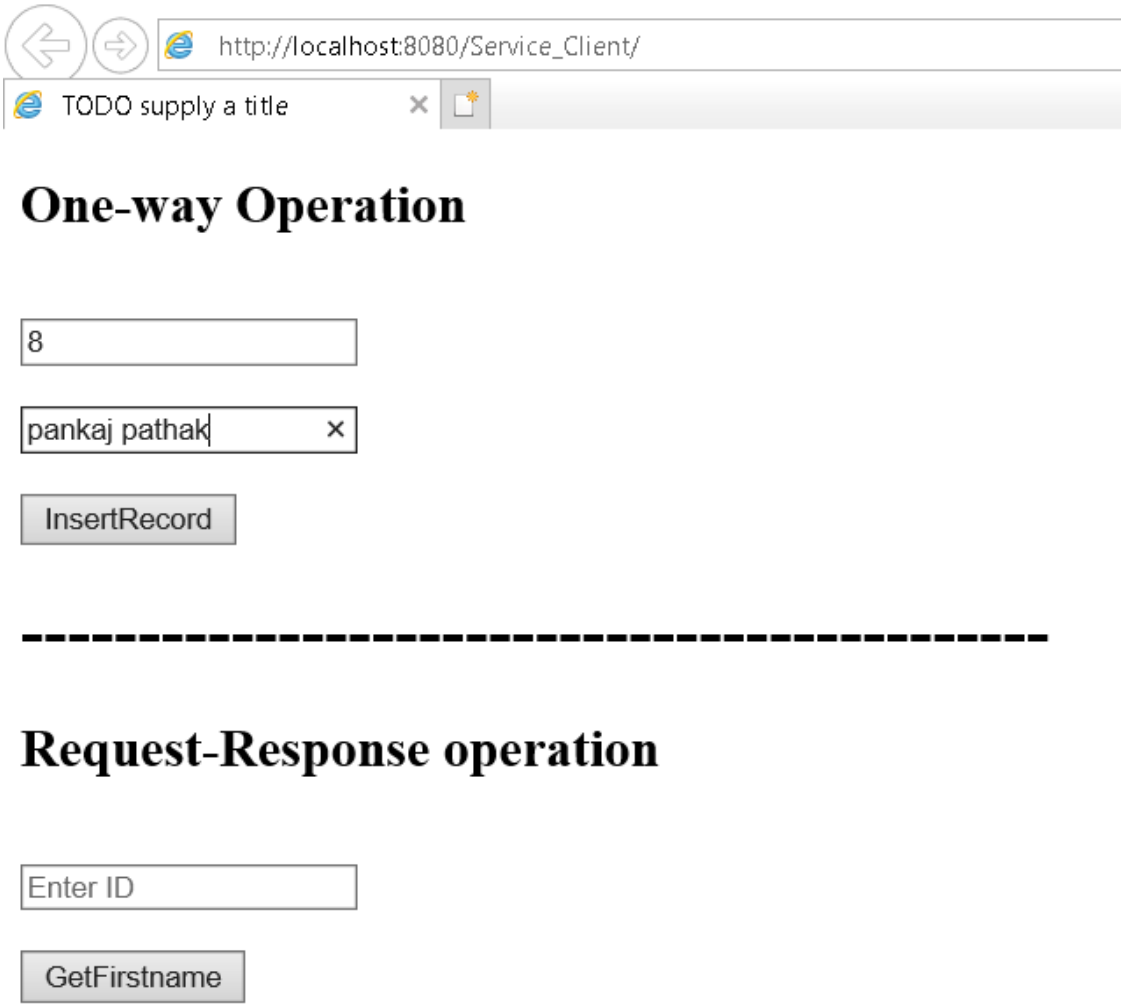
Source | History

```
        URL url = null;
        WebServiceException e = null;
        try {
            url = new URL("http://localhost:8080/Service_01/myws?wsdl");
        } catch (MalformedURLException ex) {
            e = new WebServiceException(ex);
        }
        MYWS_WSDL_LOCATION = url;
        MYWS_EXCEPTION = e;
    }

    public Myws_Service() {
        super(__getWsdlLocation(), MYWS_QNAME);
    }

    public Myws_Service(WebServiceFeature... features) {
        super(__getWsdlLocation(), MYWS_QNAME, features);
    }

    public Myws_Service(URL wsdlLocation) {
        super(wsdlLocation, MYWS_QNAME);
    }

    public Myws_Service(URL wsdlLocation, WebServiceFeature... features) {
        super(wsdlLocation, MYWS_QNAME, features);
    }

    public Myws_Service(URL wsdlLocation, QName serviceName) {
        super(wsdlLocation, serviceName);
    }

    public Myws_Service(URL wsdlLocation, QName serviceName, WebServiceFeature... features) {
        super(wsdlLocation, serviceName, features);
    }

    /**
     *
     * @return
     *     returns Myws
     */
    @WebEndpoint(name = "mywsPort")
    public Myws getMywsPort() {
        return super.getPort(new QName("http://kk.com/", "mywsPort"), Myws.class);
    }
```

Output


http://localhost:8080/Service_Client/
TODO supply a title
One-way Operation
Enter ID
EnterFirstname
InsertRecord

Request-Response operation
GetFirstname


http://localhost:8080/Service_Client/
TODO supply a title
One-way Operation
8
pankaj pathak
InsertRecord

Request-Response operation
Enter ID
GetFirstname

Practical 3 :

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

Requirement :

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

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

Theory :

- **Advantages of Soap Web Services** :

WS Security : SOAP defines its own security known as WS Security.

Language and Platform independent : SOAP web services can be written in any programming language
and executed in any platform.

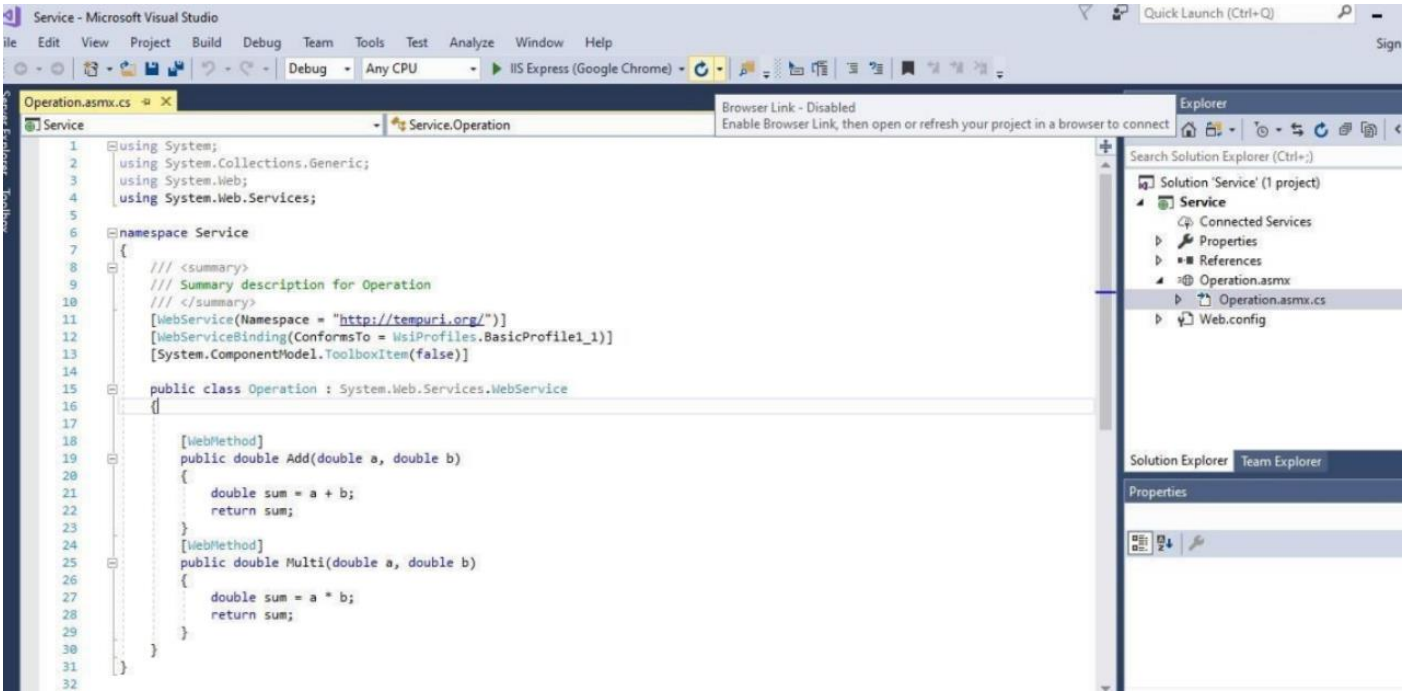
- **Disadvantages of Soap Web Services** :

Slow : SOAP uses XML format that must be parsed to be read. It defines many standards that must be
followed while developing the SOAP applications. So it is slow and consumes more bandwidth and
resource.

WSDL dependent : SOAP uses WSDL and doesn't have any other mechanism to discover the service.

Code :

Step 1 : created the project the visual studio and written code for addition and multiplication of the numbers



Step 2 : now started the execution and get this operation page in browser

Operation Web Service

localhost:40221/Operation.asmx

Apps easy The Architecture of...

Operation

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

- Add
- Multi

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

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

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under developmen 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
```

Step 3: in netbeans in created to pages Add.jsp and Multi.jsp

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

Projects Files Services

- ServiceClient
- ServiceClient1
  - Web Pages
    - WEB-INF
    - Add.jsp
    - index.html
  - Source Packages
  - Generated Sources (jax-ws)
  - Libraries
  - Web Service References
    - Operation
      - Operation
        - OperationSoap
          - Add
          - Multi
        - OperationSoap12

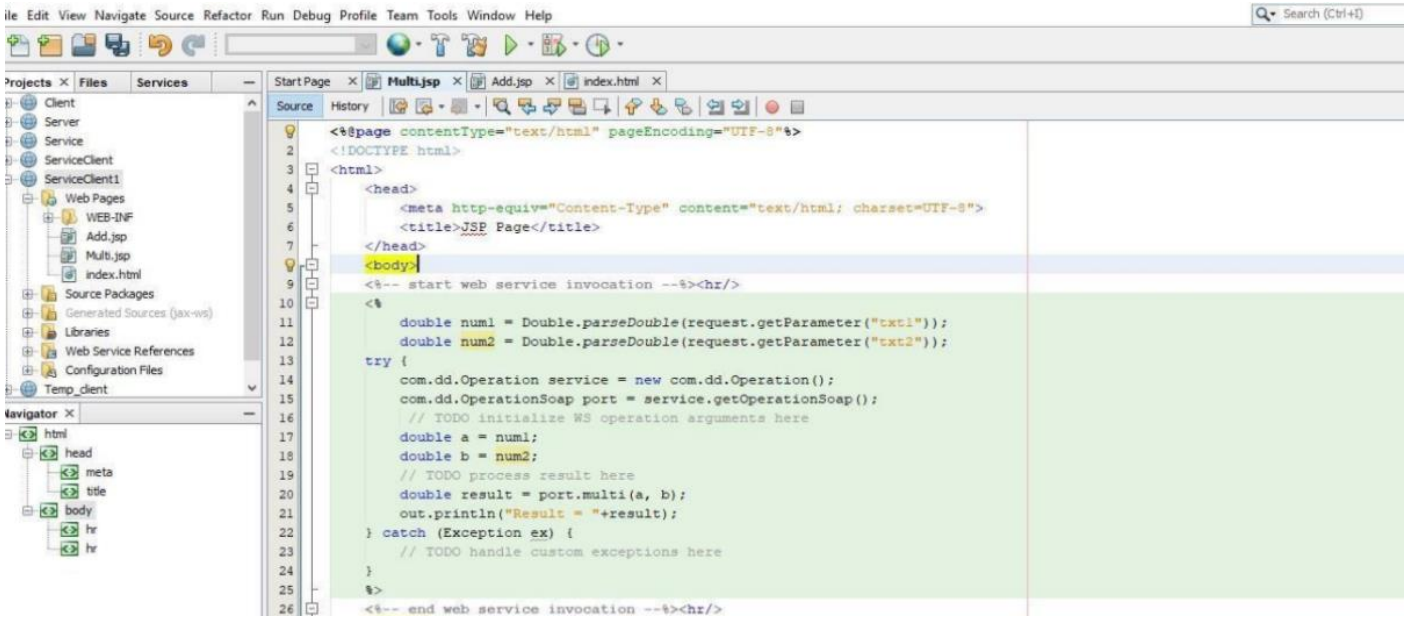
Navigator

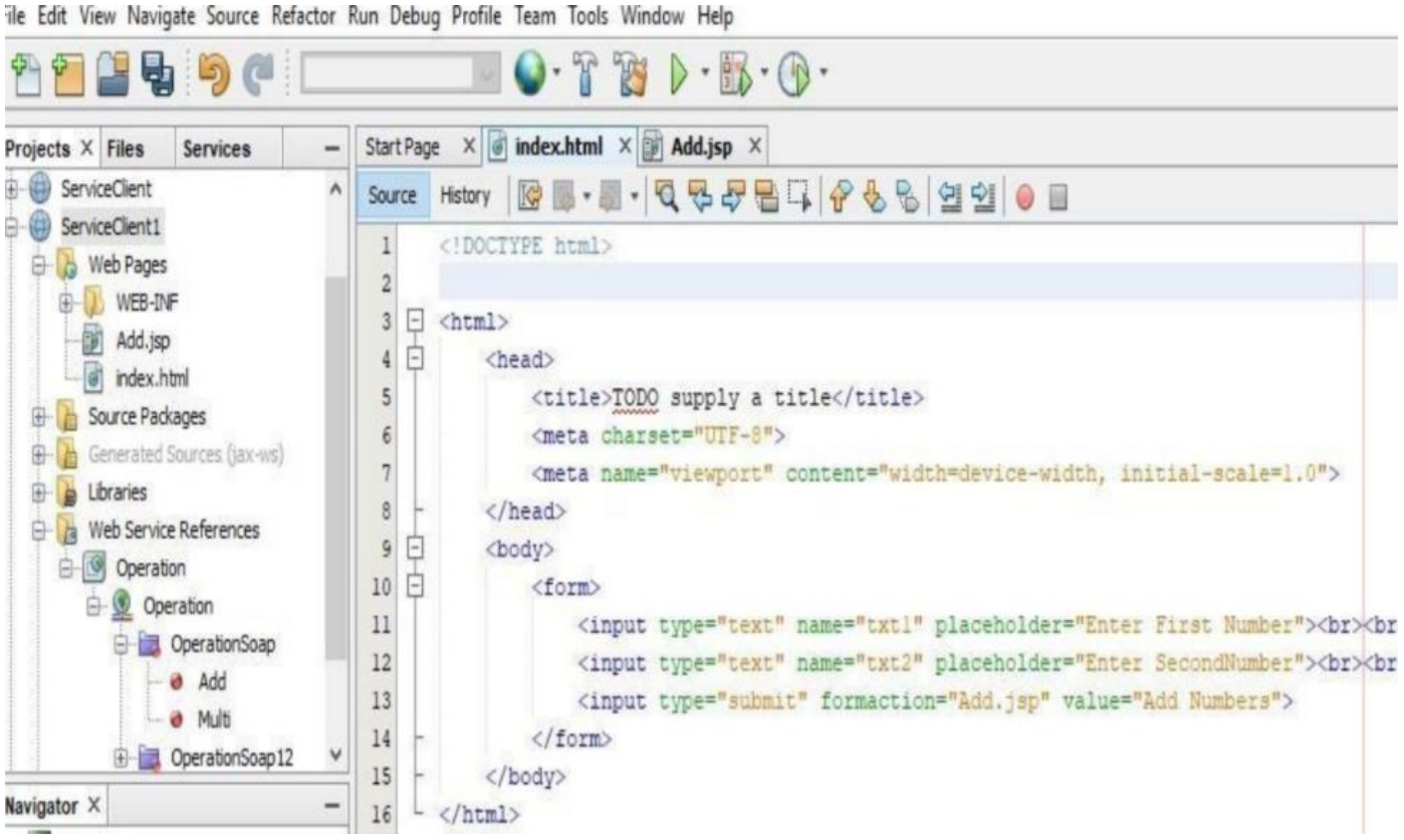
- html
  - head
    - meta
    - title
  - body
    - hr
    - hr

Start Page | index.html | Add.jsp

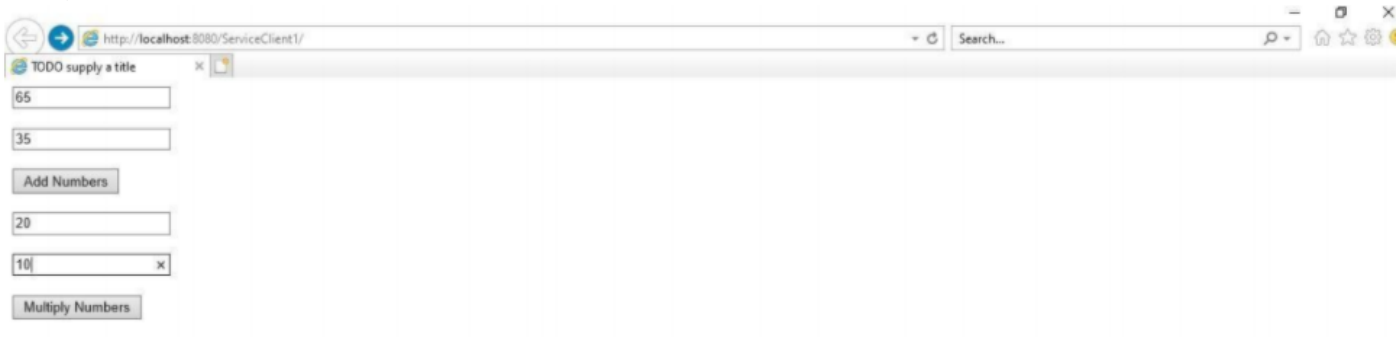
Source History

```
<%@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.Operation service = new com.dd.Operation();
        com.dd.OperationSoap port = service.getOperationSoap();
         // TODO initialize WS operation arguments here
        double a = num1;
        double b = num2;
        double result = port.add(a, b);
        out.println("Result = "+result);
    } catch (Exception ex) {
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
```


Start Page
Multi.jsp
Add.jsp
index.html
Source
History
<%@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.Operation service = new com.dd.Operation();
com.dd.OperationSoap port = service.getOperationSoap();
// TODO initialize WS operation arguments here
double a = num1;
double b = num2;
// TODO process result here
double result = port.multi(a, b);
out.println("Result = "+result);
} catch (Exception ex) {
// TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>


Start Page
index.html
Add.jsp
Source
History
<!DOCTYPE html>
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<form>
<input type="text" name="txt1" placeholder="Enter First Number">
<input type="text" name="txt2" placeholder="Enter SecondNumber">
<input type="submit" formaction="Add.jsp" value="Add Numbers">
</form>
</body>
</html>

Output :


http://localhost:8080/ServiceClient1/
Search...
TODO supply a title
65
35
Add Numbers
20
10
Multiply Numbers


http://localhost:8080/ServiceClient1/Add.jsp?txt1=65&txt2=35
Search...
JSP Page
Result = 100.0


http://localhost:8080/ServiceClient1/Multi.jsp?txt1=20&txt2=10
Search...
JSP Page
Result = 200.0

Practical 4

Aim : write a jax-ws to perform the following operation define a servlet/jsp that consume the web service

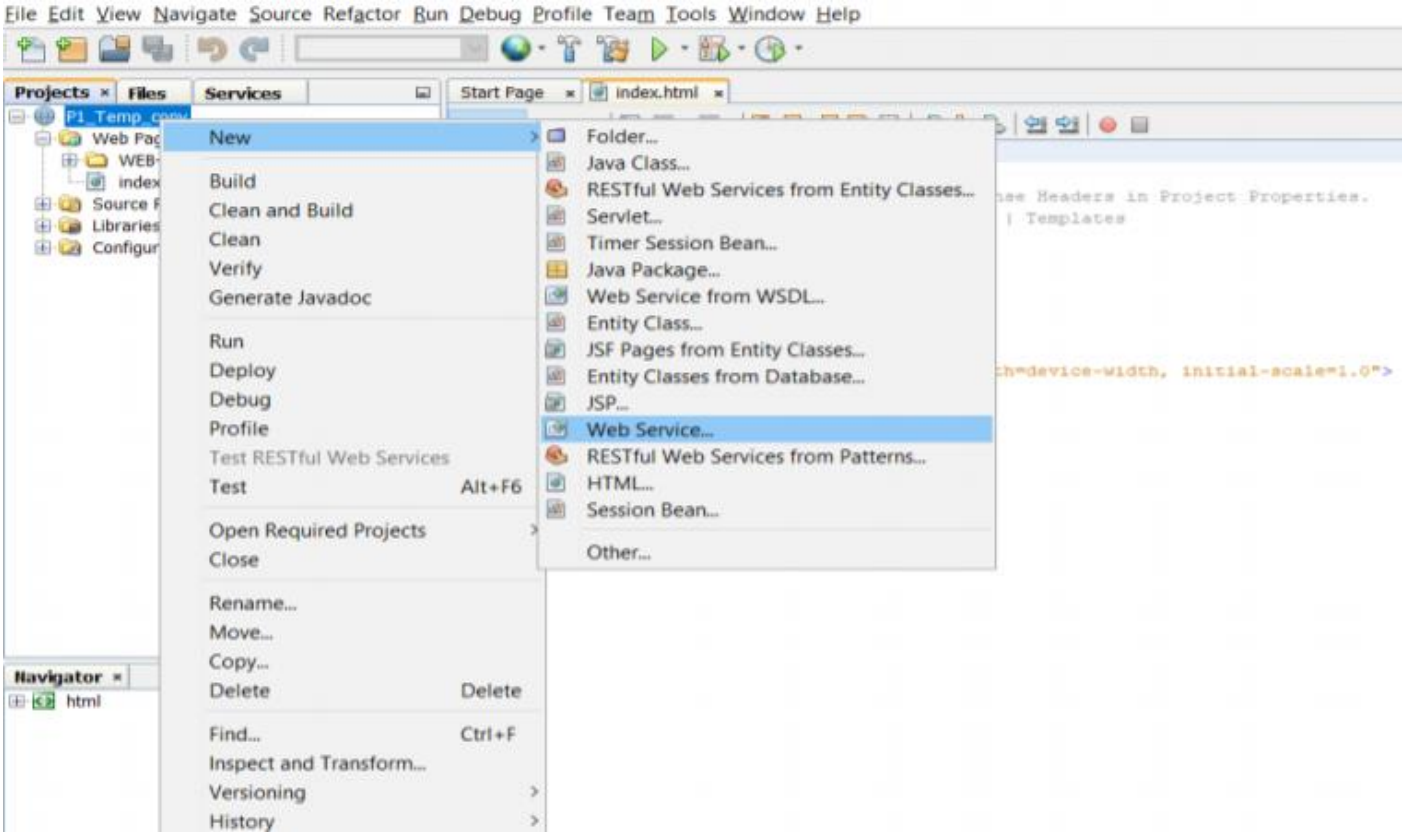
Theory :

- Java API for XML Web Services (JAX-WS) is one of a set of Java technologies used to develop Web services.
- JAX-WS belongs to what Sun Microsystems calls the "core Web services" group. Like most of the core group, JAX-WS is typically used in conjunction with other technologies. Those other technologies may also come from the core Web services group (JAXB, for example), as well as from enhanced Web services (WSIT), secure Web services (WSIT, WS-Security), legacy Web services (JAX-RPC), and systems management services (WS-Management) groups.
- JAX-WS is a fundamental technology for developing SOAP (Simple Object Access Protocol) and RESTful (Web services that use representational state transfer, or REST, tools) Java Web services, where JAX-WS is designed to take the place of the JAVA-RPC (Remote Procedure Call) interface in Web services and Web-based applications.
- JAX-WS is also used to build Web services and corresponding clients that communicate using XML to send messages or use remote procedure calls to exchange data between client and service provider.
- JAX-WS represents remote procedure calls or messages using XML-based protocols such as SOAP, but hides SOAP's innate complexity behind a Java-based API.
- Developers use this API to define methods, then code one or more classes to implement those methods and leave the communication details to the underlying JAX-WS API. Clients create a local proxy to represent a service, then invoke methods on the proxy.
- The JAX-WS runtime system converts API calls and matching replies to and from SOAP messages.

Code :

Step 1 :

Created the web services



Step 2 :

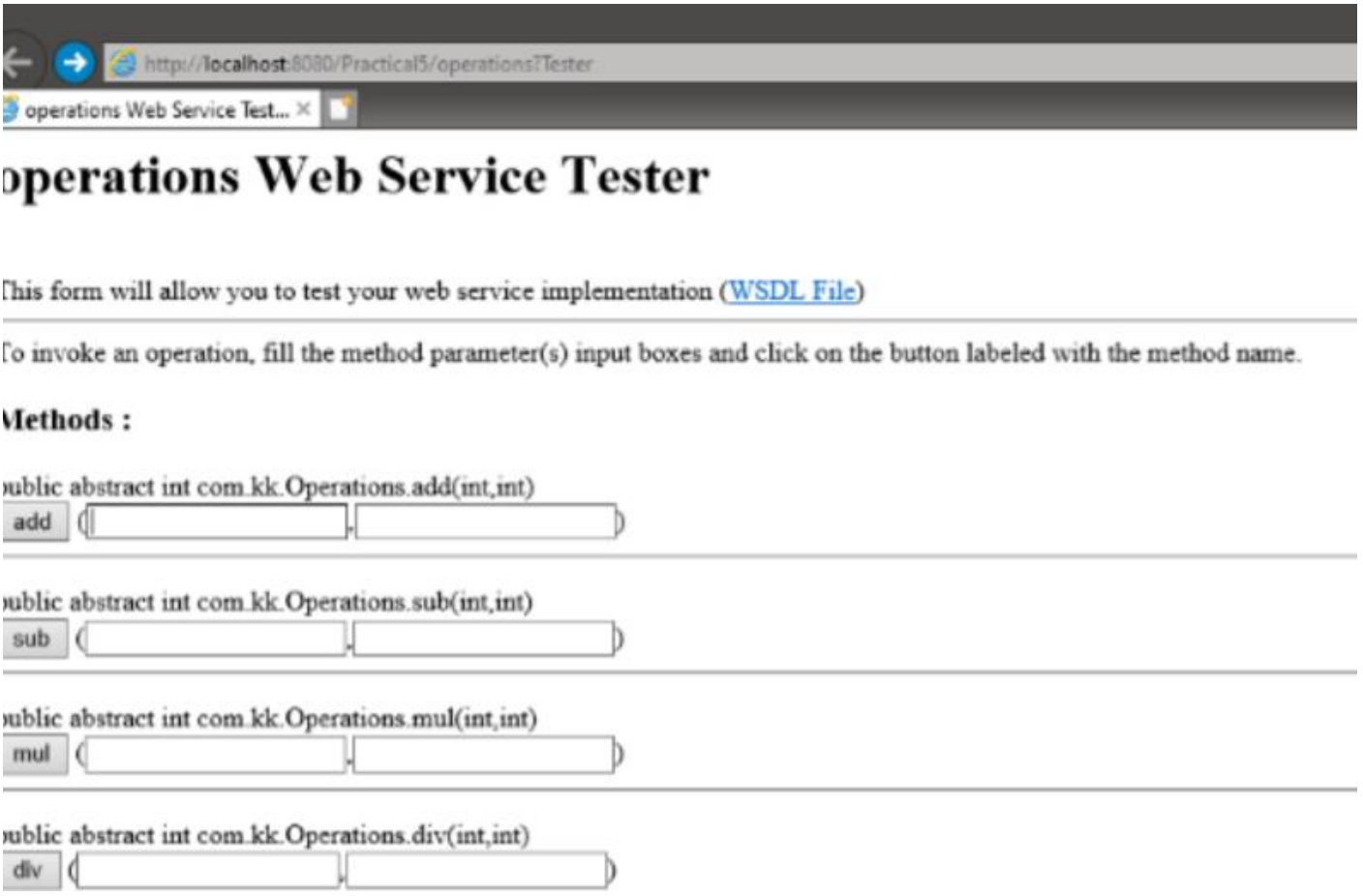
```
Source  Design  History
1  /*
2   * To change this license header, choose License Headers in Project
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package com.kk;
7
8  import javax.jws.WebService;
9  import javax.jws.WebMethod;
10 import javax.jws.WebParam;
11
12 /**
13  *
14  * @author DELL
15  */
16 @WebService(serviceName = "operations")
17 public class operations {
18
19     /**
20      * This is a sample web service operation
21      */
22     @WebMethod(operationName = "hello")
23     public String hello(@WebParam(name = "name") String txt) {
```

Step 3

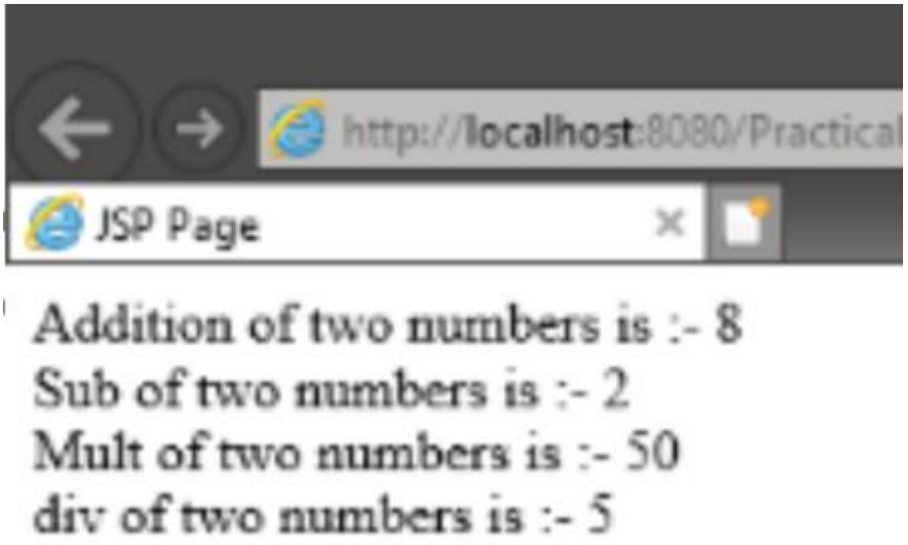
```
Source  Design  History
1  package com.kk;
2
3  import javax.jws.WebService;
4  import javax.jws.WebMethod;
5  import javax.jws.WebParam;
6
7  @WebService(serviceName = "operations")
8  public class operations {
9
10     @WebMethod(operationName = "Sub")
11     public int Sub(@WebParam(name = "a") int a, @WebParam(name = "b") int
12         int c= a-b ;
13         return c;
14
15     }
16     @WebMethod(operationName = "Div")
17     public int Div(@WebParam(name = "a") int a, @WebParam(name = "b") int
18         int c= a/b ;
19         return c;
20
21     }
22     @WebMethod(operationName = "Add")
23     public int Add(@WebParam(name = "a") int a, @WebParam(name = "b") int
```

Step 4:

Output :



Output of addition ,sub,multi,div



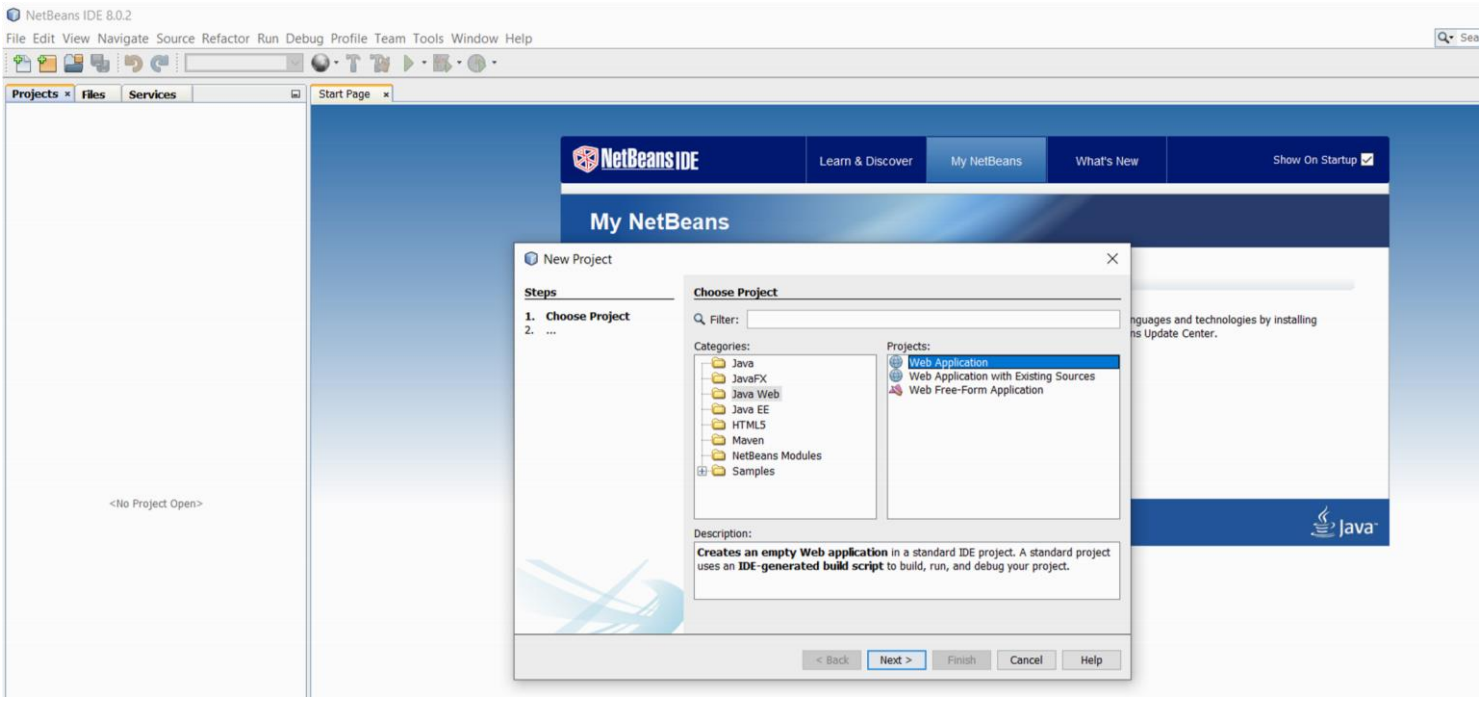
Practical 6

Aim : Define a RESTful web service that accept the detail to be stored in database and performs CRUD operation.

Theory :

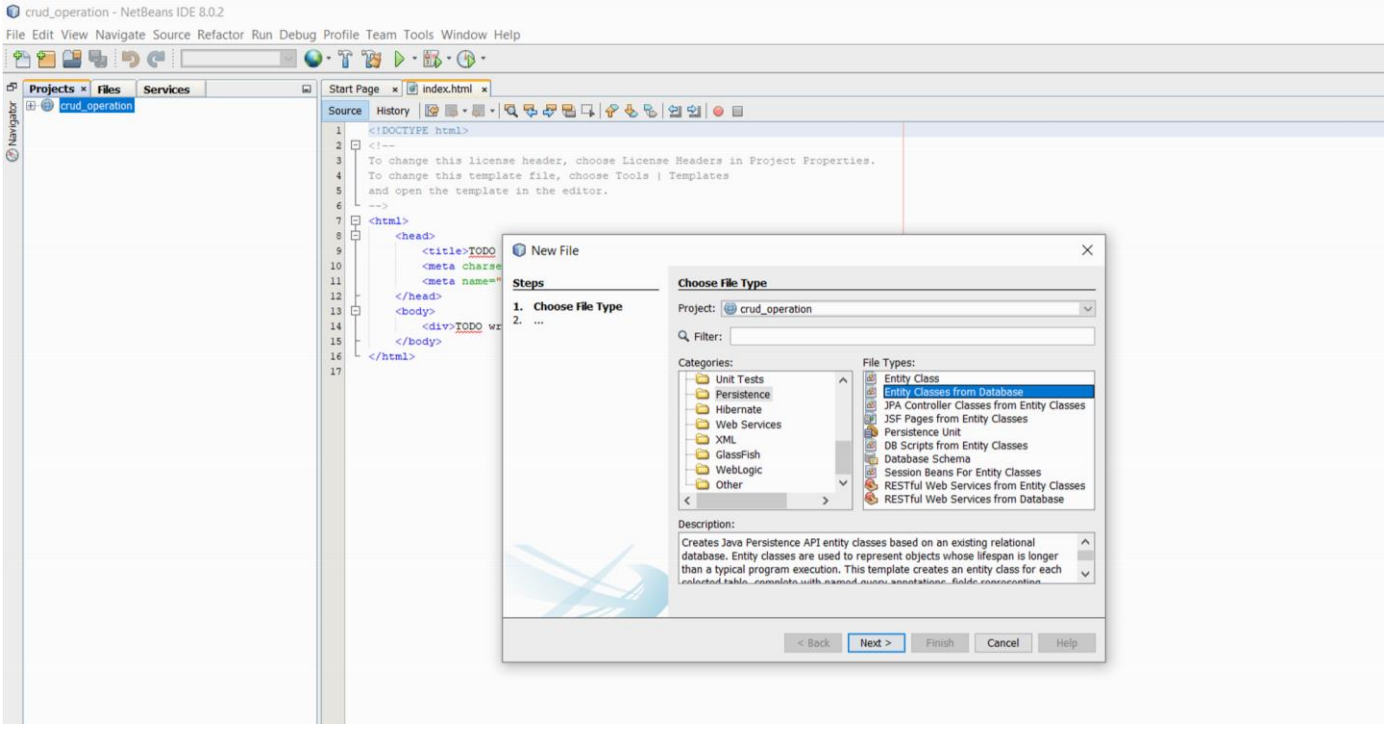
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems.
- Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.
- This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.
- RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource.
- RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.
- Web services based on REST Architecture are known as RESTful web services. These webservices uses HTTP methods to implement the concept of REST architecture.
- A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

code :



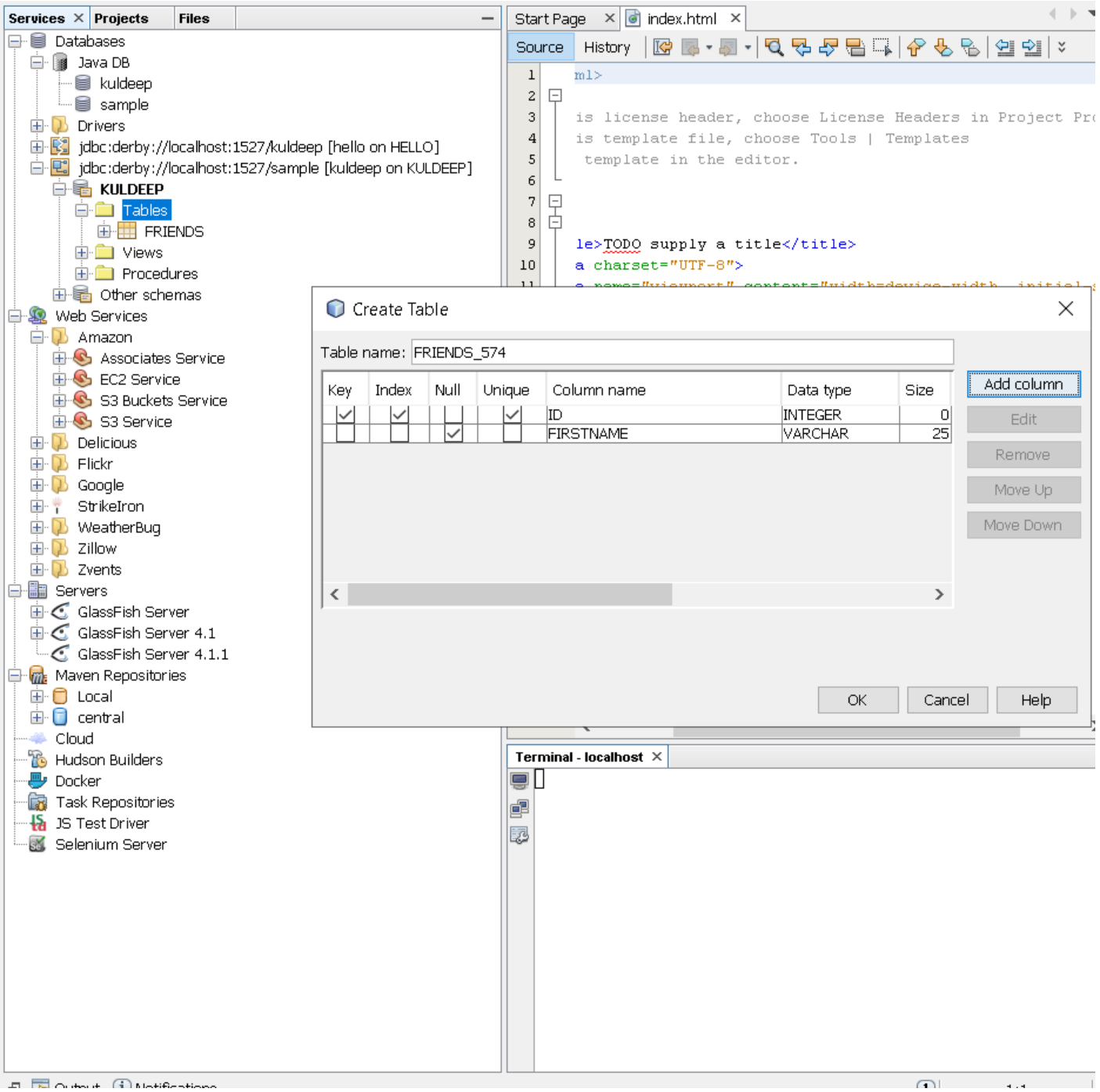
Step 1 :

Created the entity class for data bases



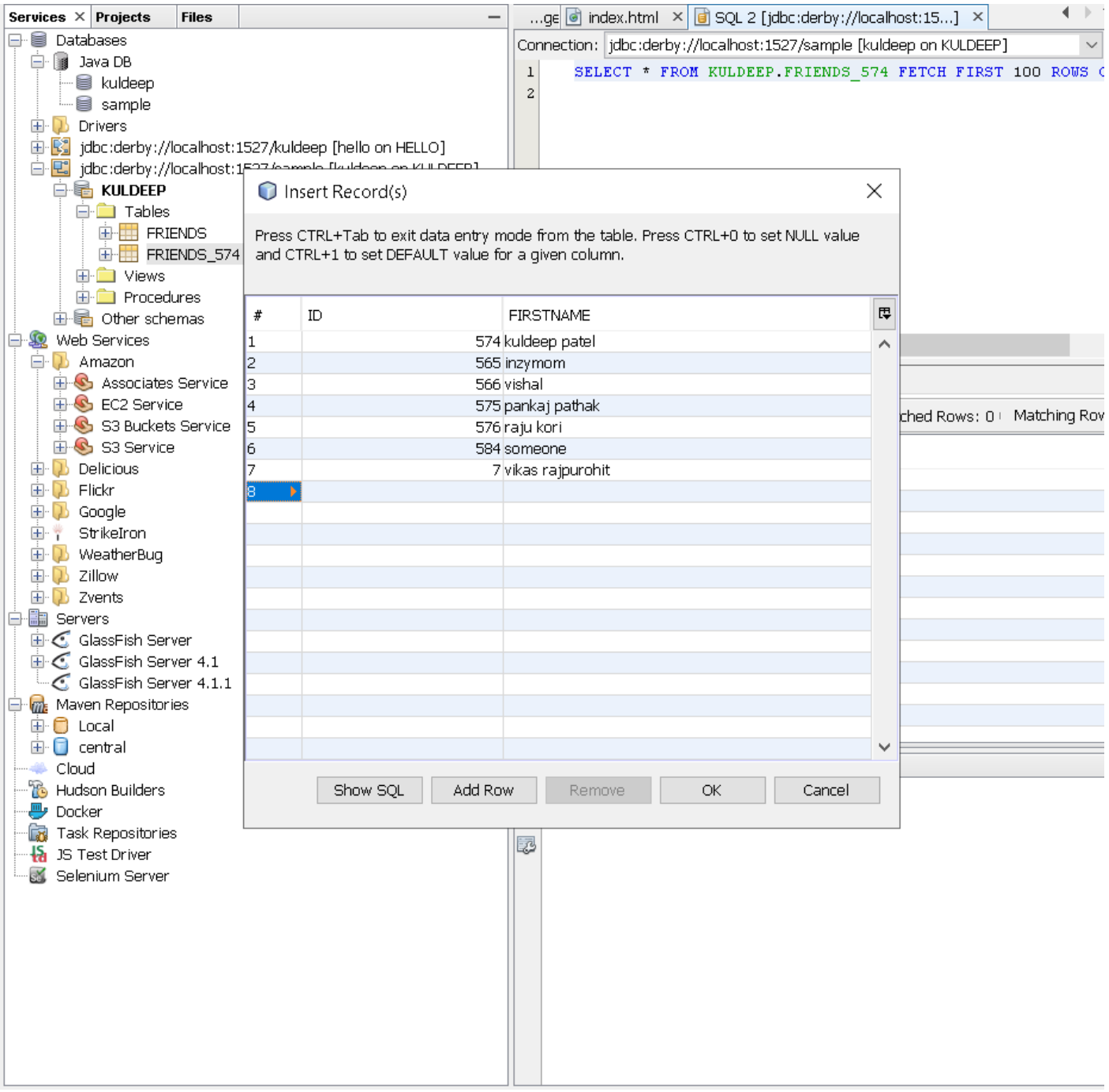
Step 2 :

Created the table name as FRIENDS_574

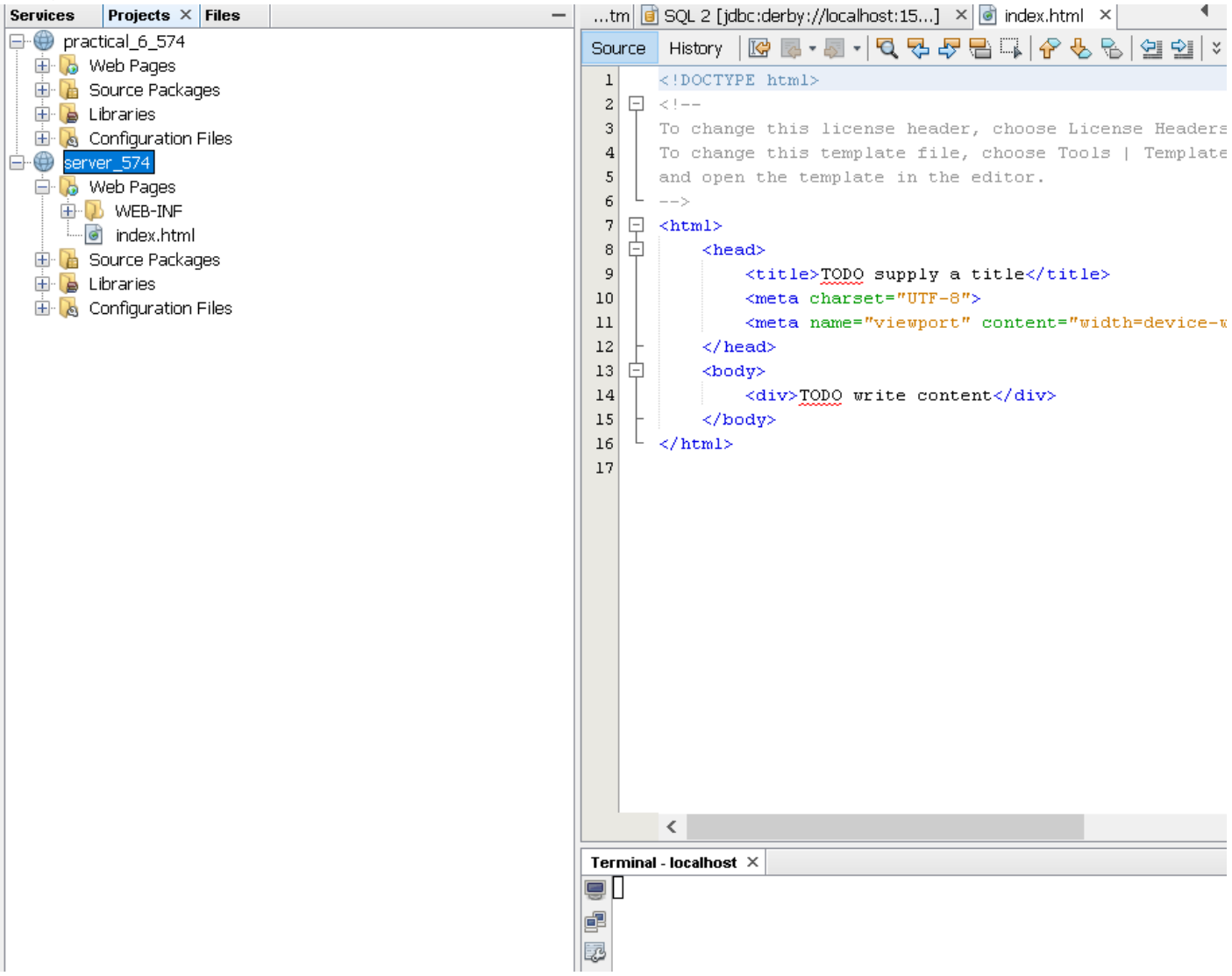


STEP 3 :

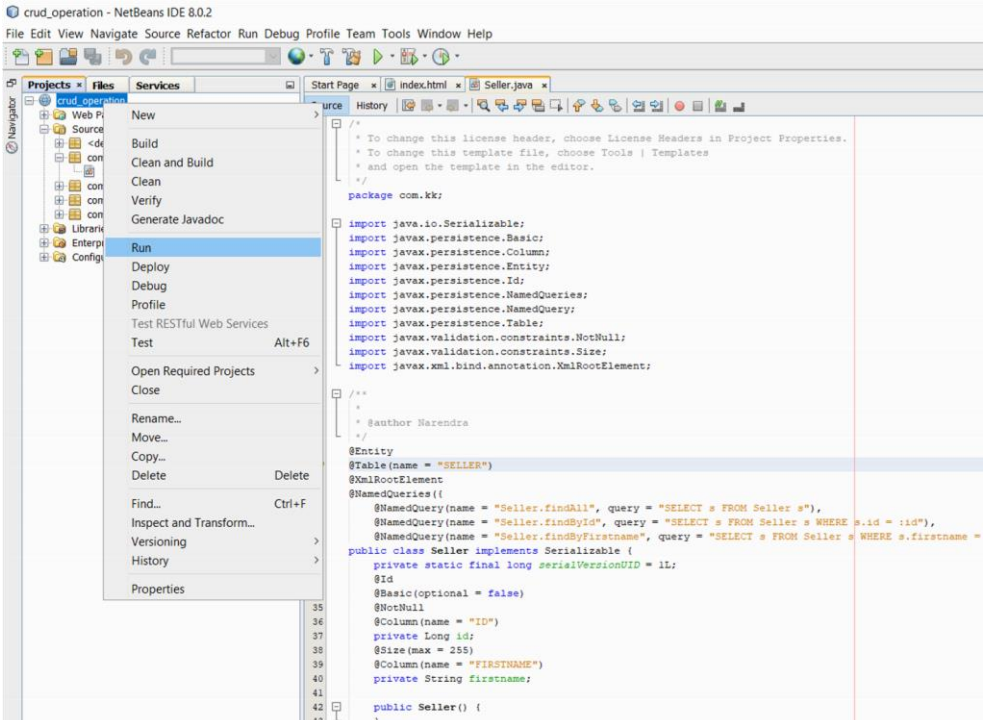
Inserted the record as given below



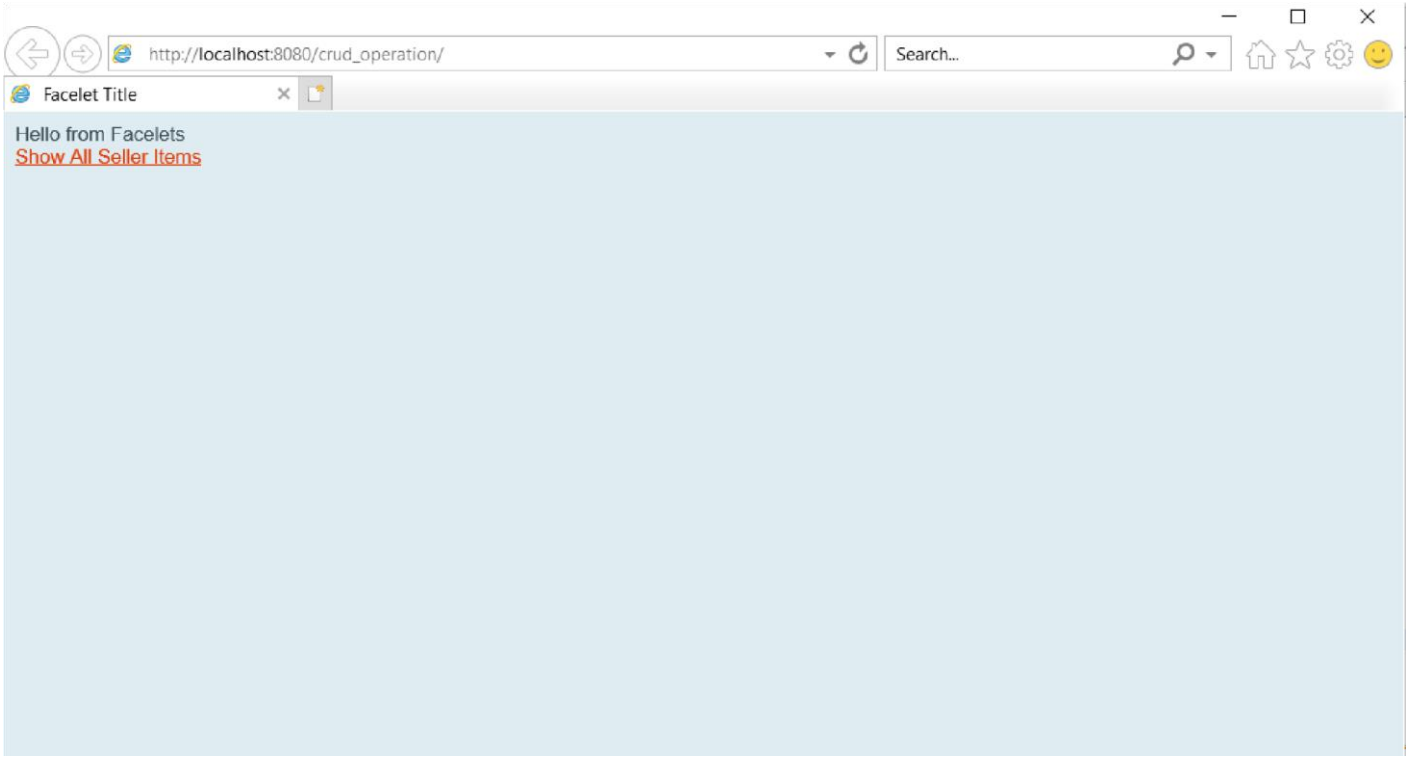
Step 4 :

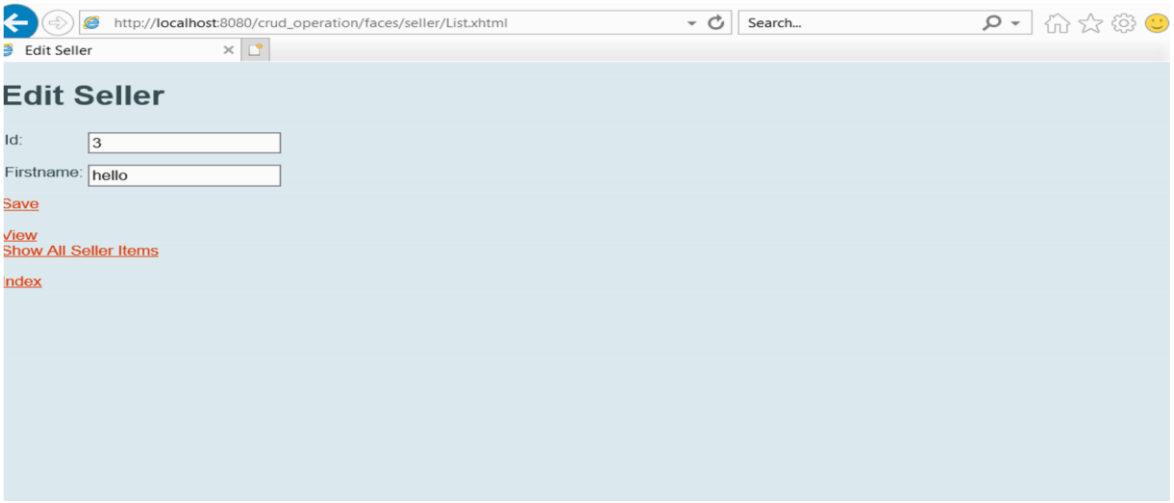


Step 5 : running the project

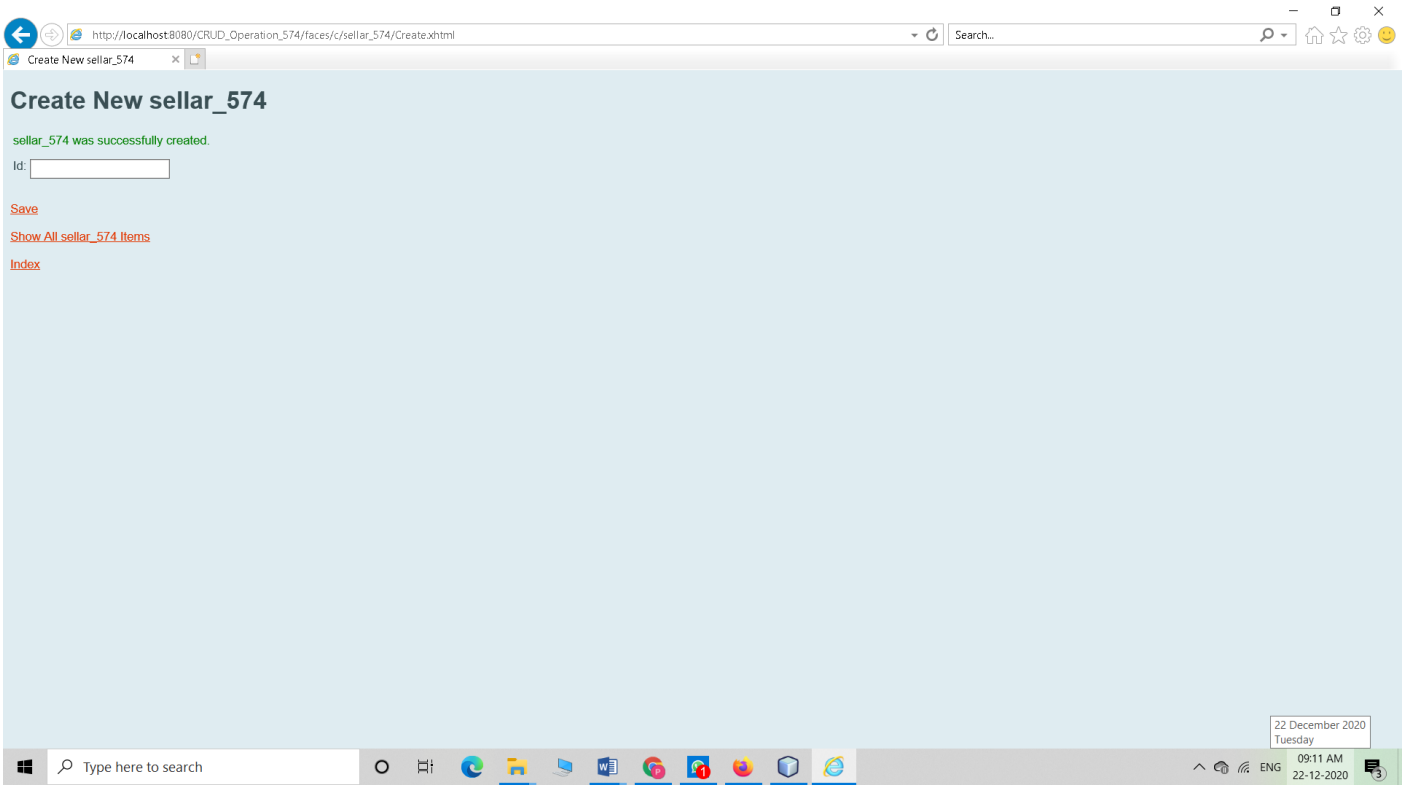


Output :


http://localhost:8080/crud_operation/
Search...
Facelet Title
Hello from Facelets
Show All Seller Items


http://localhost:8080/crud_operation/faces/seller/List.xhtml
Search...
Edit Seller
Edit Seller
Id:
3
Firstname:
hello
Save
View
Show All Seller Items
Index

TYCS - 574 KULDEEP PATEL



Practical 7

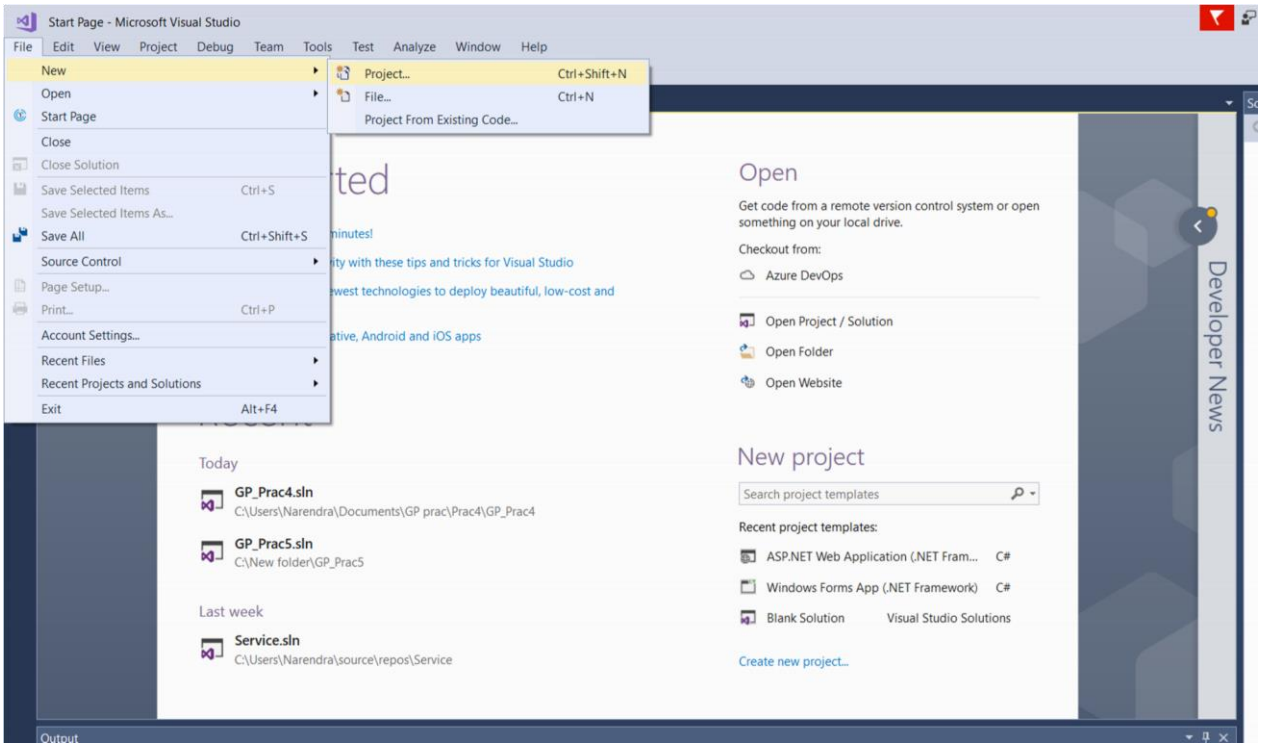
Aim: Implement a typical service and a typical client using WCF

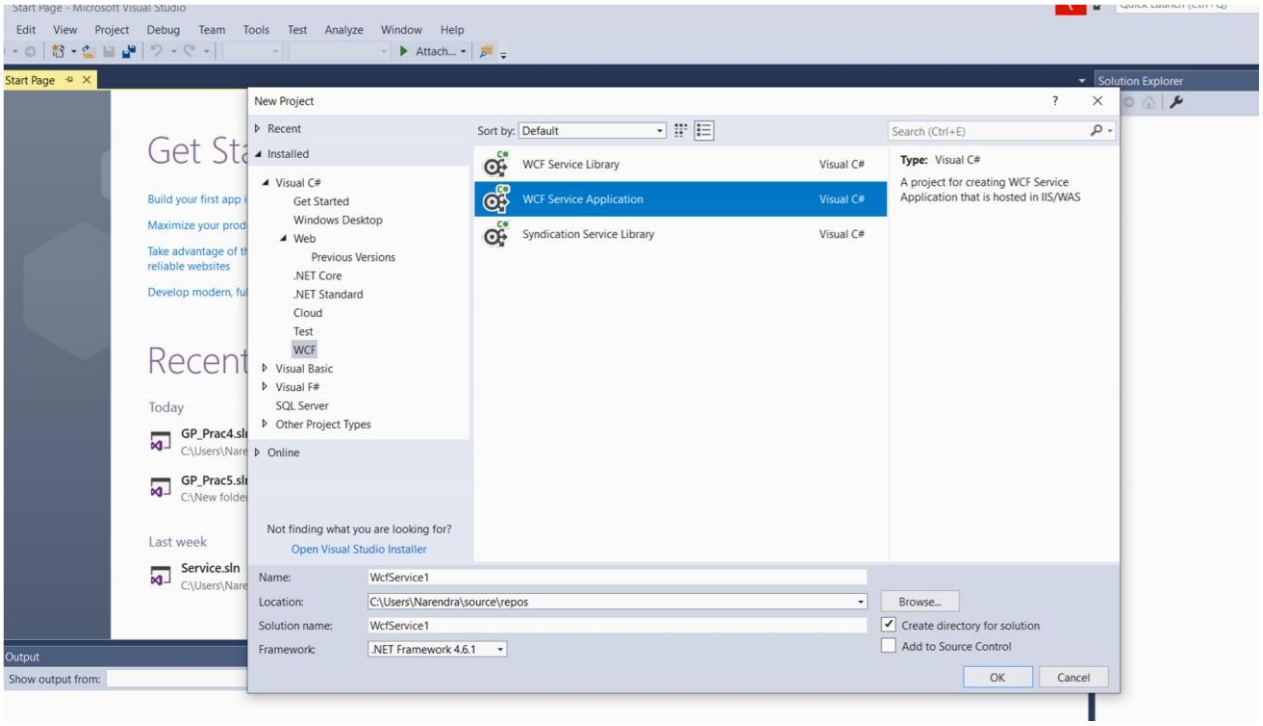
Theory :

- A client application uses the WCF client proxy to communicate with the service.
- Client applications usually import a service's metadata to generate WCF client code that can be used to invoke the service.
- The basic steps for creating a WCF client include the following:Compile the service code
- Generate the WCF client proxy.
- Instantiate the WCF client proxy.
- The WCF client proxy can be generated manually by using the Service Model Metadata Utility Tool (SvcUtil.exe) for more information see, Service Model Metadata Utility Tool (Svcutil.exe).
- The WCF client proxy can also be generated within Visual Studio using the Add Service Reference feature.
- To generate the WCF client proxy using either method the service must be running.
- If the service is self-hosted you must run the host. If the service is hosted in IIS/WAS you do not need to do anything else.
- The Svcutil.exe is a command-line tool for generating code from metadata.

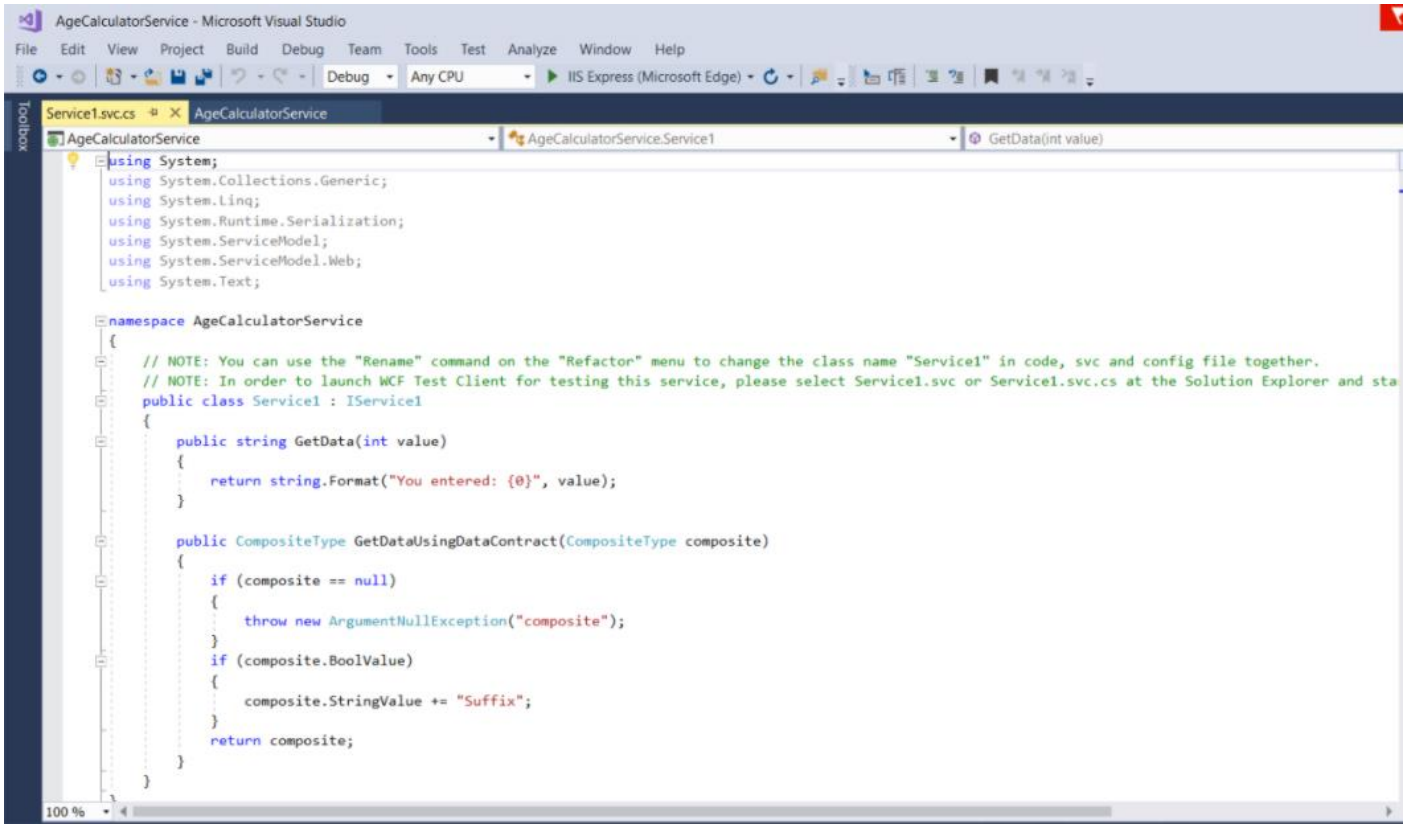
Steps :

Step 1 : created the project





Step 2 : inserted the code



AgeCalculatorService - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug | Any CPU | IIS Express (Microsoft Edge)

IService1.cs | Service1.svc.cs

AgeCalculatorService | AgeCalculatorService.CompositeType | BoolValue

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

namespace AgeCalculatorService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.
    [ServiceContract]
    public interface IService1
    {

        [OperationContract]
        string GetData(int value);

        [OperationContract]
        CompositeType GetDataUsingDataContract(CompositeType composite);

        // TODO: Add your service operations here
    }


    // Use a data contract as illustrated in the sample below to add composite types to service operations.
    [DataContract]
    public class CompositeType
    {
        bool boolValue = true;
        string stringValue = "Hello ";
```

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug | Any CPU | IIS Express (Microsoft Edge)

Service1.svc | IService1.cs* | Service1.svc.cs

AgeCalculatorService | AgeCalculatorService.IService1 | calculateDays(int day, int Month, int year)

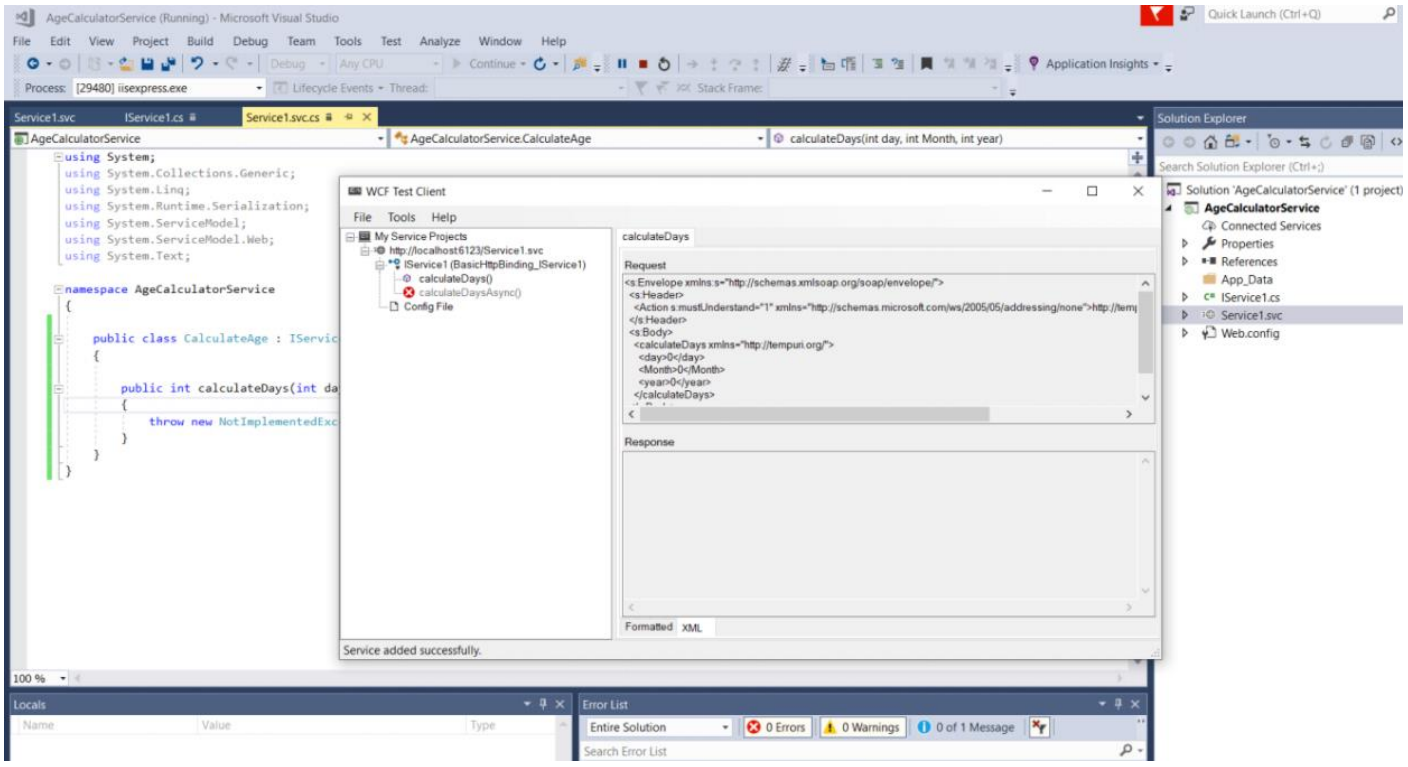
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace AgeCalculatorService
{

    [ServiceContract]
    public interface IService1
    {

        [OperationContract]
        int calculateDays(int day, int Month, int year);

        // TODO: Add your service operations here
    }


}
```

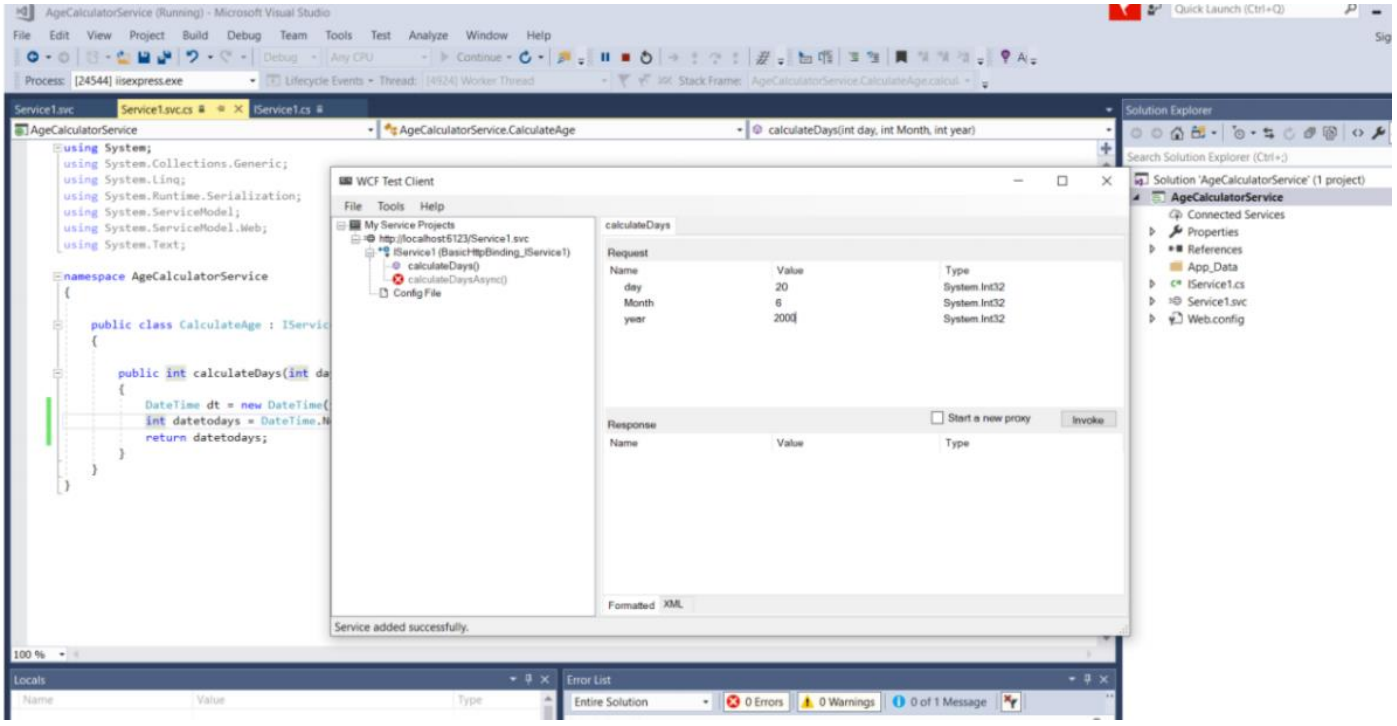
Output

Show output from: Debug

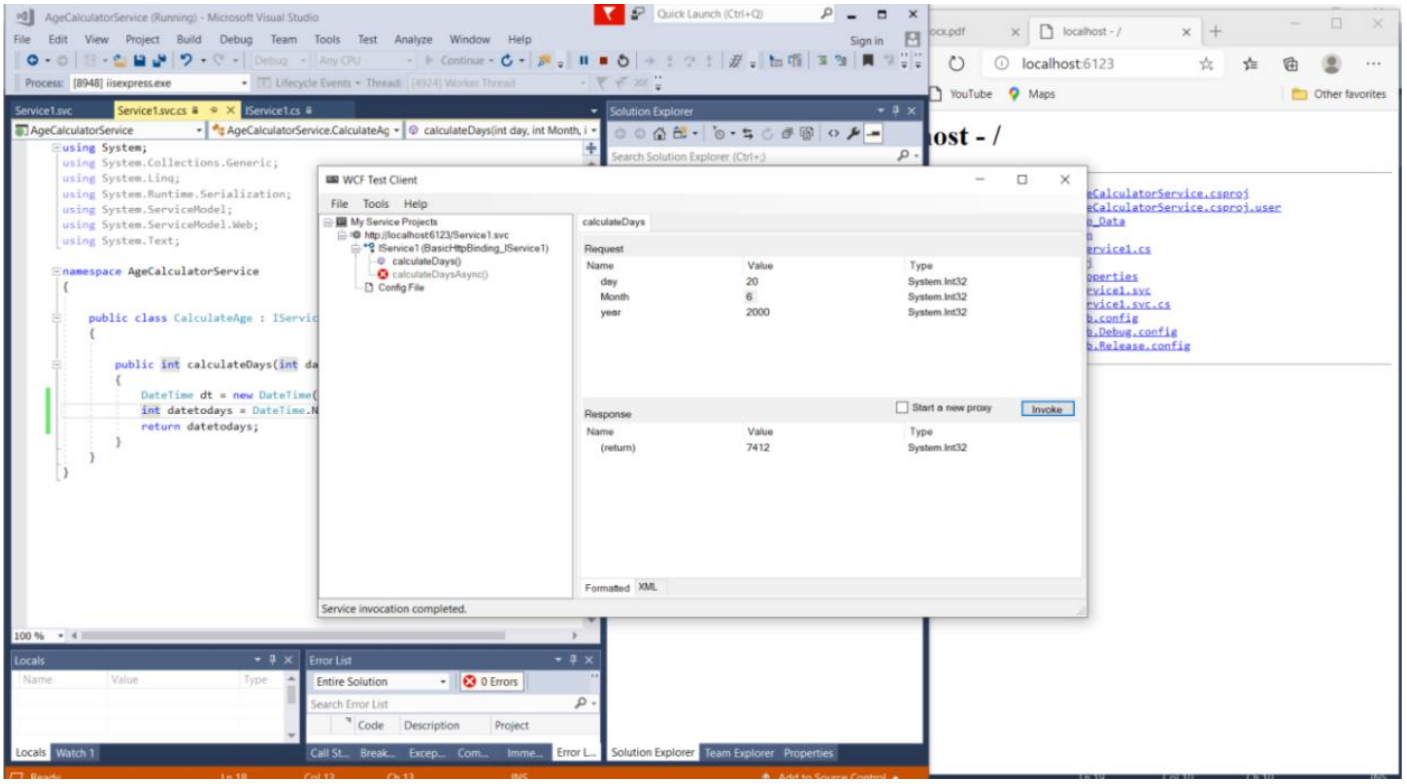
```
The program '[26248] WCFTestClient.exe' has exited with code -1 (0xffffffff).
The program '[12752] iisexpress.exe' has exited with code -1 (0xffffffff).
```

Step 3 : created the wcf test client





Output :


AgeCalculatorService (Running) - Microsoft Visual Studio
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;
namespace AgeCalculatorService
public class CalculateAge : IServic
public int calculateDays(int da
DateTime dt = new DateTime(
int datetodays = DateTime.N
return datetodays;
WCF Test Client
File Tools Help
My Service Projects
http://localhost:6123/Service1.svc
IService1 (BasicHttpBinding_IService1)
calculateDays()
calculateDaysAsync()
Config File
calculateDays
Request
Name Value Type
day 20 System.Int32
Month 6 System.Int32
year 2000 System.Int32
Start a new proxy
Invoke
Response
Name Value Type
(return) 7412 System.Int32
Formatted XML
Service invocation completed.
Solution Explorer
Locals
Error List
Entire Solution
0 Errors
Code Description Project
Solution Explorer Team Explorer Properties
localhost:6123
YouTube Maps
Other favorites

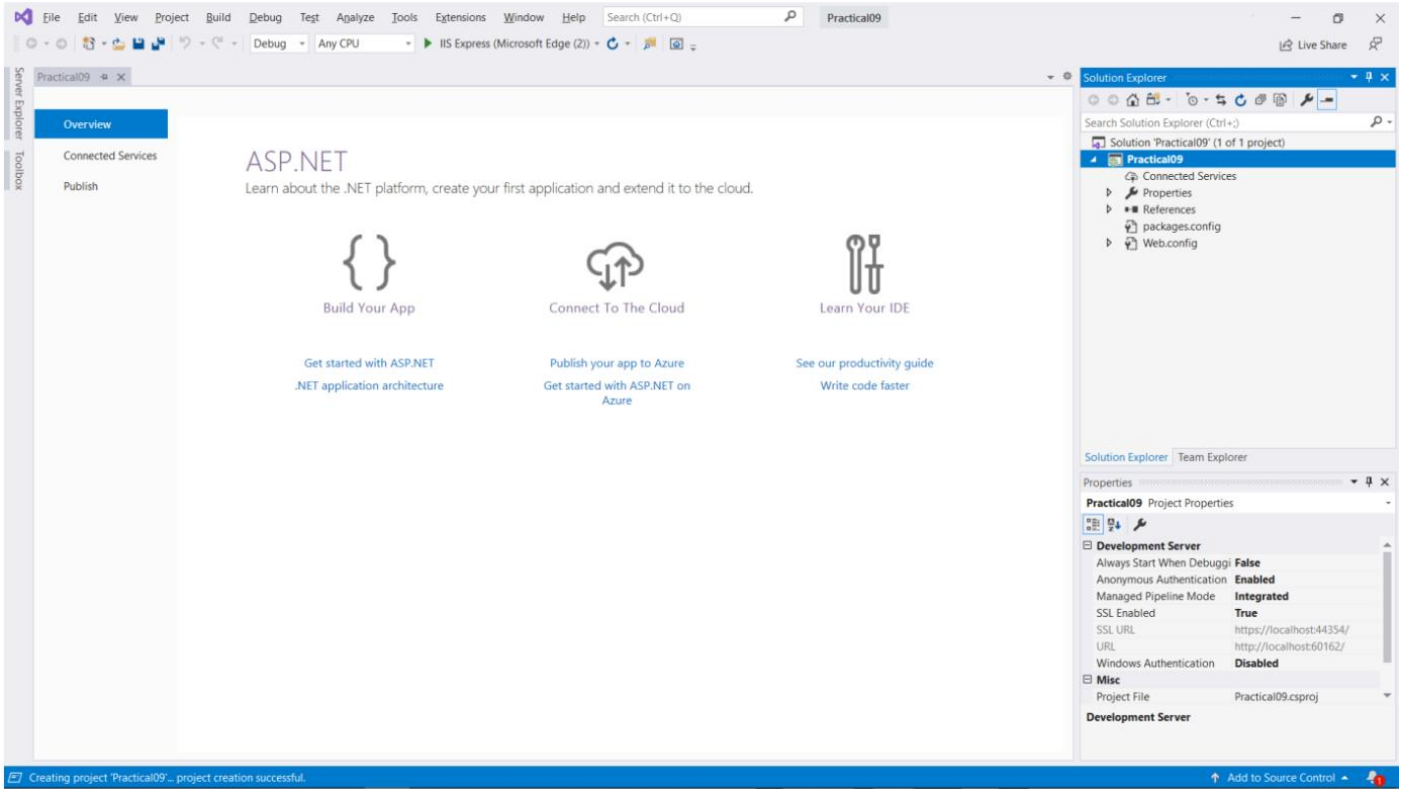
Practical no 8

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

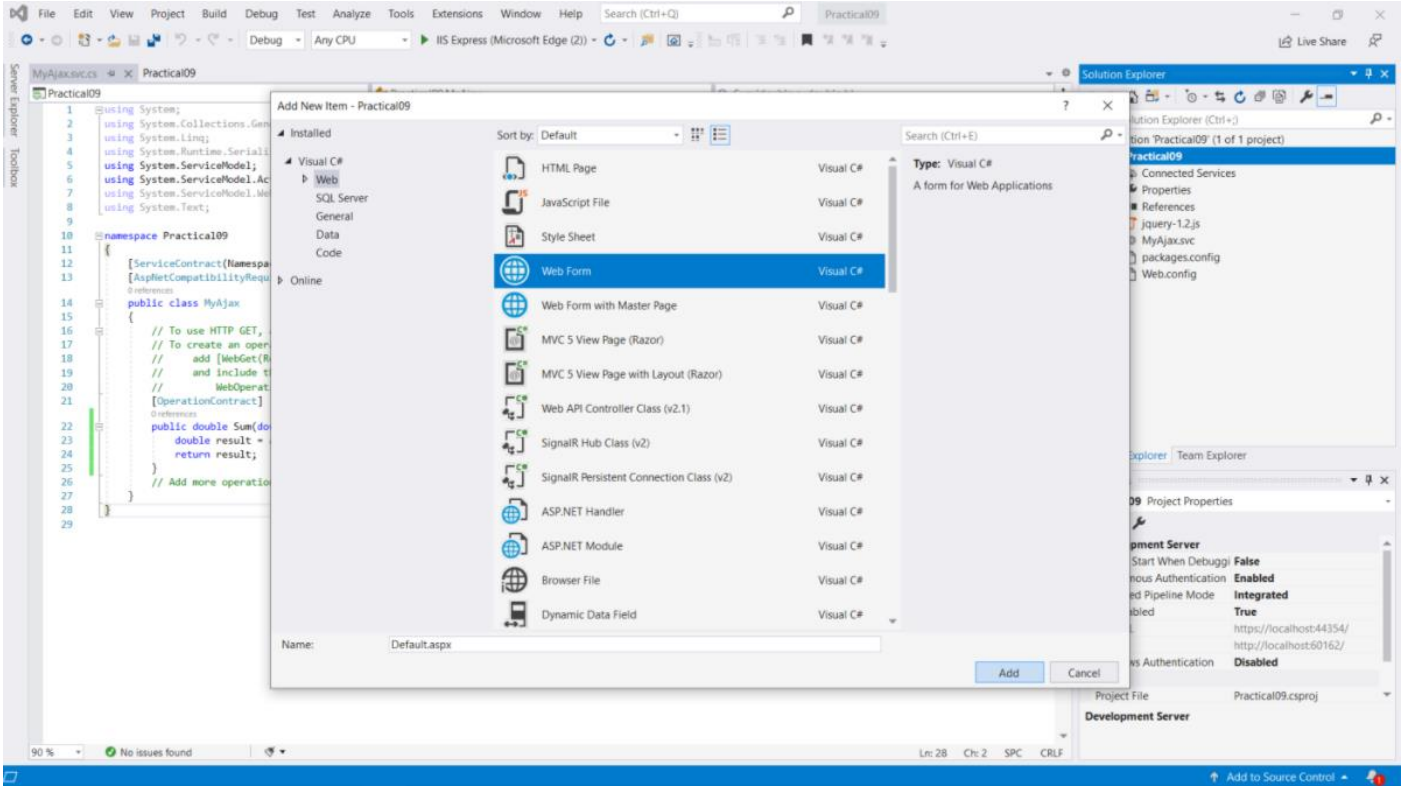
Theory :

- WCF is Microsoft's unified programming model for building service-oriented applications.
- It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments.
- ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices.
- ASP.NET Web API is an ideal platform for building RESTful applications on the.
- NET Framework. This topic presents some guidance to help you decide which technology will best meet your needs.
- Use WCF to create reliable, secure web services that are accessible over a variety of transports.
- Use ASP.NET Web API to create HTTP-based services that are accessible from a wide variety of clients.
- Use ASP.NET Web API if you are creating and designing new REST-style services.
- Although WCF provides some support for writing REST-style services, the support for REST in ASP.NET Web API is more complete and all future REST feature improvements will be made in ASP.NET Web API.
- If you have an existing WCF service and you want to expose additional REST endpoints, use WCF and the WebHttpBinding.

Steps 1: created the project



Step 2: creating the web form



```
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 Practical09
{
    [ServiceContract(Namespace = "")]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
    public class MyAjax
    {
        // 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;
        }

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

```
        double result = a + b;
        return result;
    }
    // Add more operations here and mark them with [OperationContract]
}
}
```

Step 3: inserted the code which is given

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Practical09.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
     <script src="Scripts/jquery-1.7.1.min.js" type="text/javascript"></script>
     <script type="text/javascript">
       $(document).ready(function(){
         $("#btn").click( function () {
             var num1 = $("#Txt1").val();
             var num2 = $("#Txt2").val();
         $.ajax({
         type: "POST",
         contentType: "application/json; charset=utf-8",
             url: "MyAjax.svc/Sum",
             data: JSON.stringify({ a: num1 , b:num2 }),
         dataType: "json",
             success: function (data) {
                 $("#Txt3").val(data.d);
         },
         error: function (errormsg) {
         alert(errormsg.responseText);
         }
         });
     });
     });
     </script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <input id="Txt1" type="text" /><br/><br/>
            <input id="Txt2" type="text" /><br/><br/>
            <input id="btn" type="button" value="Add Numbers" /><br/><br/>
            <input id="Txt3" type="text"/>
        </div>
    </form>
</body>
</html>
```

Ouput :

Practical 9

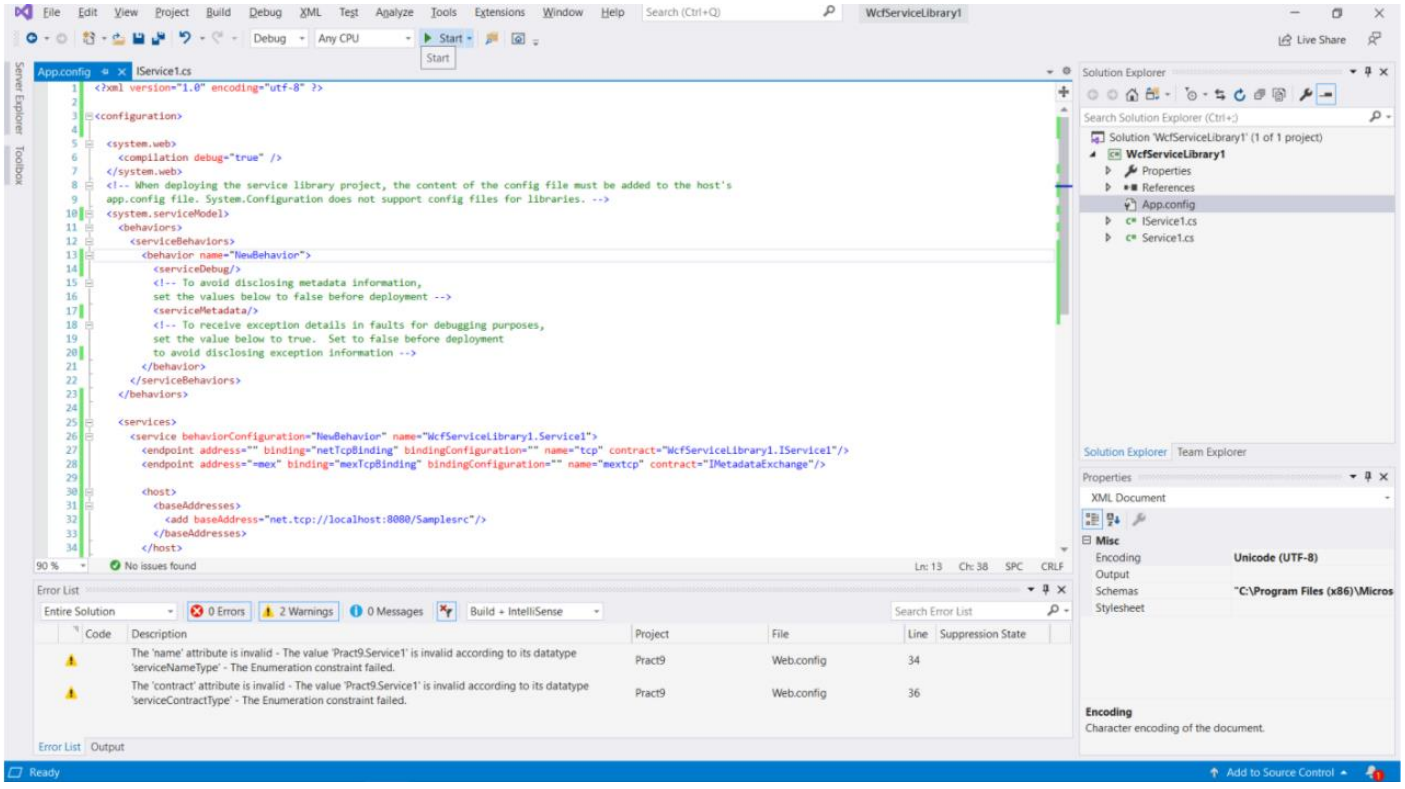
Aim : Demonstrates using the binding attribute of an endpoint element in WCF.

Theory :

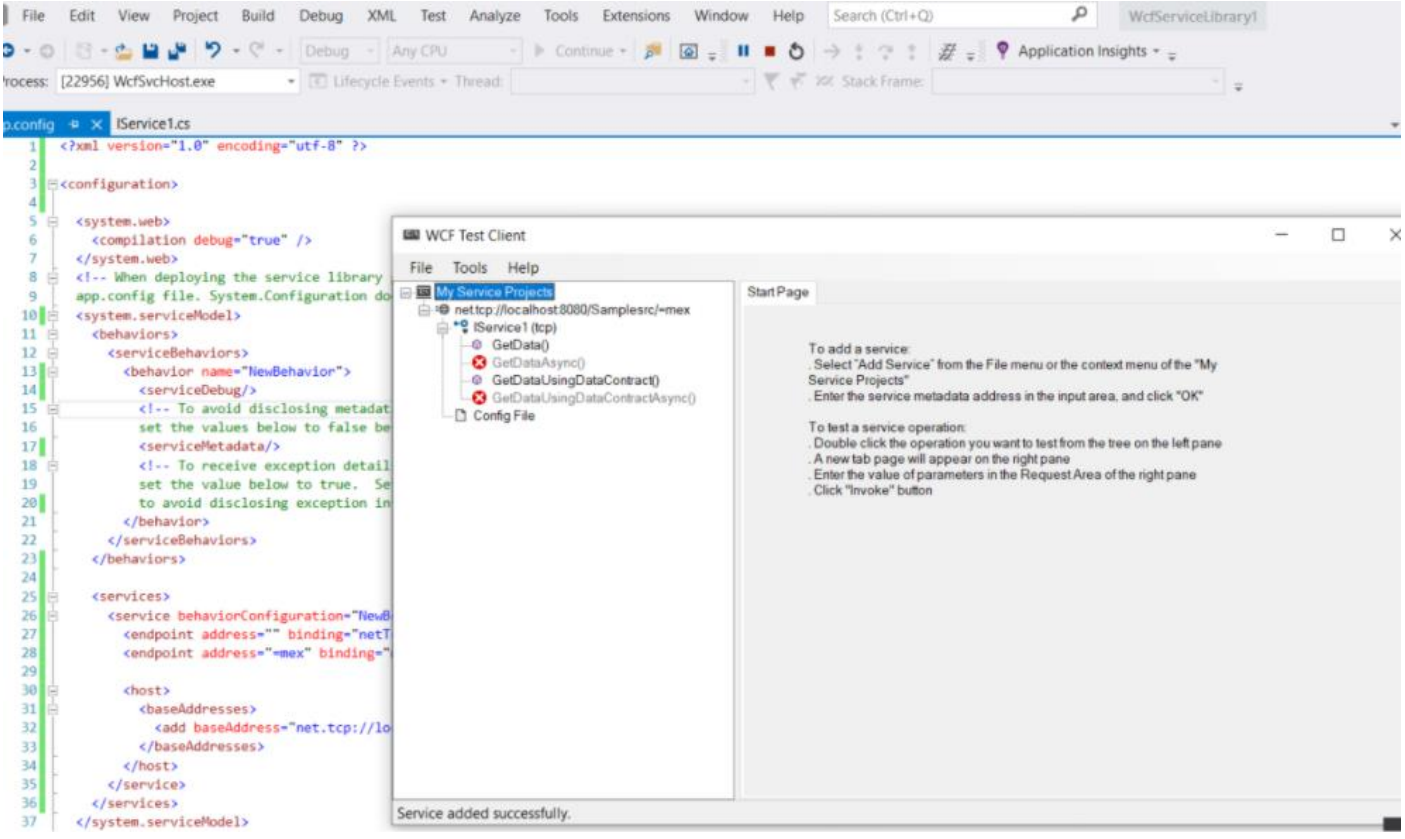
- A binding specifies transports (HTTP, TCP, pipes, Message Queuing) and protocols (Security,Reliability, Transaction flows) and consists of binding elements, each of which specifies an aspect of how an endpoint communicates with the world.
- For example, specifying the <basicHttpBinding> element indicates to use HTTP as the transport for an endpoint.
- This is used to wire up the endpoint at run time when the service using this endpoint is opened. There are two kinds of bindings: predefined and custom.
- Predefined bindings contain useful combinations of elements that are used in common scenarios. For a list of predefined binding types that WCF provides, see System-Provided Bindings.
- If no predefined binding collection has the correct combination of features that a service application needs, you can construct custom bindings to meet the application's requirements. For more information about custom bindings, see <customBinding>.Using bindings entails two basic
- Select or define a binding.
- The easiest method is to choose one of the system-provided bindings and use its default settings. You can also choose a system-provided binding and reset its property values to suit your requirements.
- Alternatively, you can create a custom binding and set every property as required.
- Create an endpoint that uses this binding

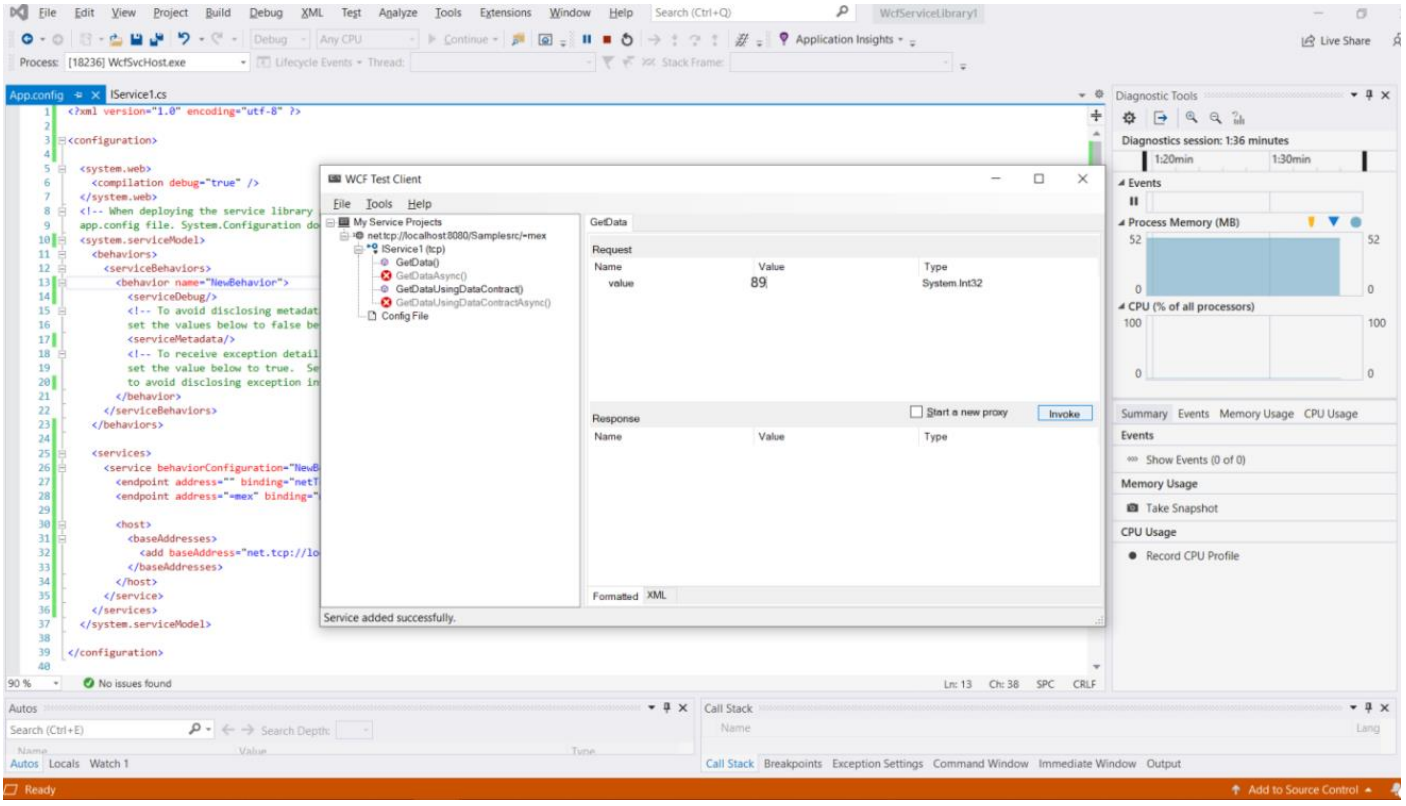
step 1 : created the project

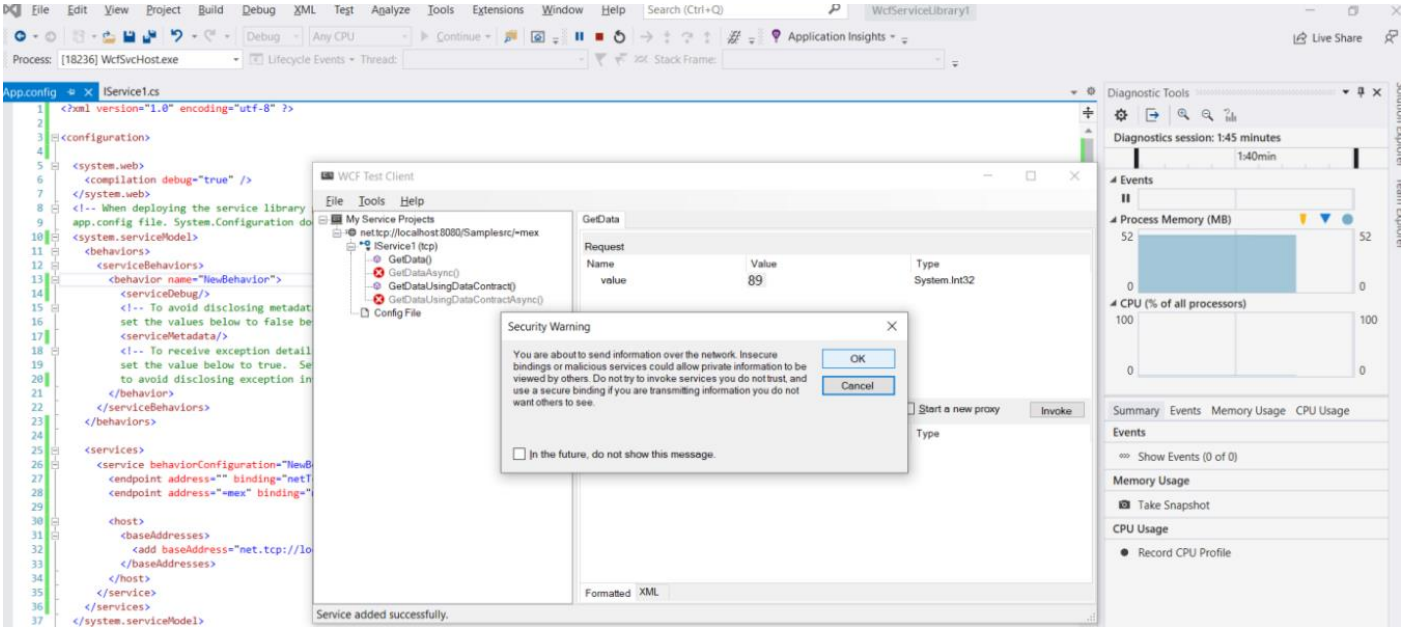
step 2 : inserted the code which is given



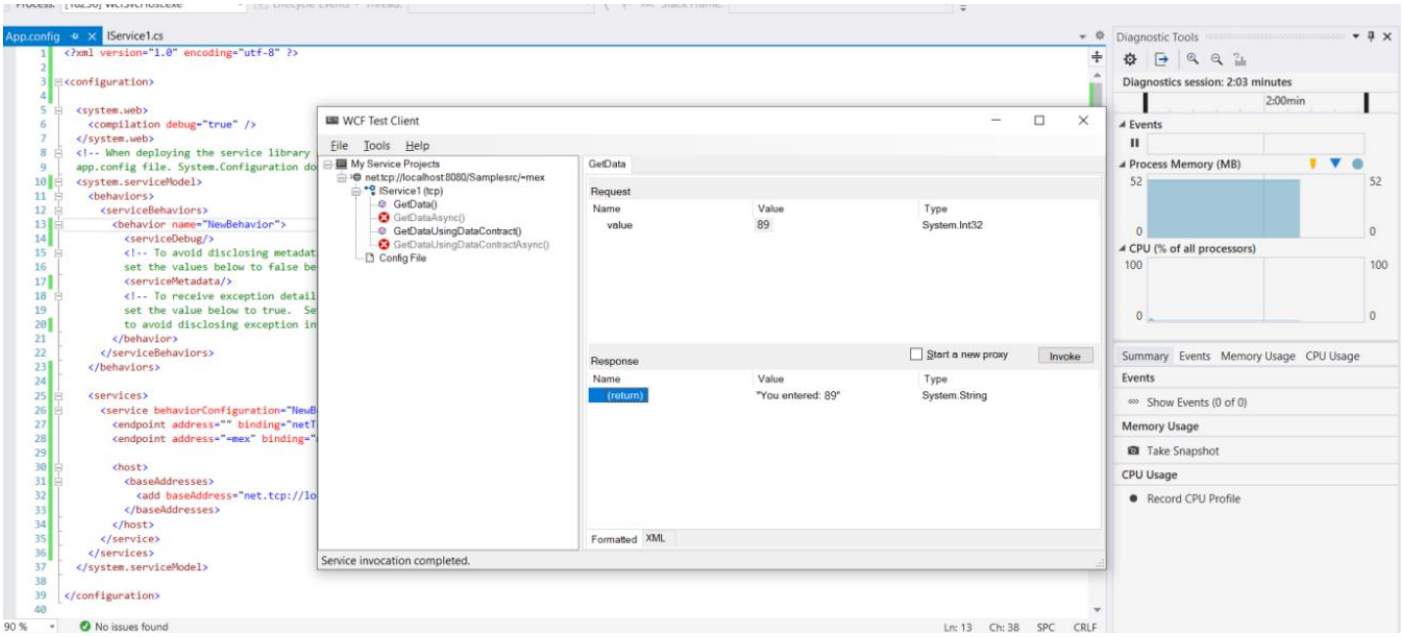
Step 3: running the project




File Edit View Project Build Debug XML Test Analyze Tools Extensions Window Help
Process: [18236] WcfSvcHost.exe
App.config
IService1.cs
WCF Test Client
File Tools Help
My Service Projects
net.tcp://localhost:8080/Samplesrc/~mex
IService1 (tcp)
GetData()
GetDataAsync()
GetDataUsingDataContract()
GetDataUsingDataContractAsync()
Config File
GetData
Request
Name
Value
Type
value
89
System.Int32
Response
Start a new proxy
Invoke
Formatted
XML
Service added successfully.
Diagnostic Tools
Diagnostics session: 1:36 minutes
Events
Process Memory (MB)
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 0)
Take Snapshot
Record CPU Profile
No issues found
Autos
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Ready
Add to Source Control


File Edit View Project Build Debug XML Test Analyze Tools Extensions Window Help
Process: [18236] WcfSvcHost.exe
App.config
IService1.cs
WCF Test Client
File Tools Help
My Service Projects
net.tcp://localhost:8080/Samplesrc/~mex
IService1 (tcp)
GetData()
GetDataAsync()
GetDataUsingDataContract()
GetDataUsingDataContractAsync()
Config File
GetData
Request
Name
Value
Type
value
89
System.Int32
Security Warning
You are about to send information over the network. Insecure bindings or malicious services could allow private information to be viewed by others. Do not try to invoke services you do not trust, and use a secure binding if you are transmitting information you do not want others to see.
OK
Cancel
In the future, do not show this message.
Start a new proxy
Invoke
Formatted
XML
Service added successfully.
Diagnostic Tools
Diagnostics session: 1:45 minutes
Events
Process Memory (MB)
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 0)
Take Snapshot
Record CPU Profile

Output :


App.config
IService1.cs
WCF Test Client
File Tools Help
My Service Projects
net.tcp://localhost:8080/Samplesrc/~mex
IService1 (tcp)
GetData()
GetDataAsync()
GetDataUsingDataContract()
GetDataUsingDataContractAsync()
Config File
GetData
Request
Name Value Type
value 89 System.Int32
Response
Start a new proxy
Invoke
Name Value Type
(return) "You entered: 89" System.String
Formatted XML
Service invocation completed.
Diagnostic Tools
Diagnostics session: 2:03 minutes
Events
Process Memory (MB)
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Show Events (0 of 0)
Memory Usage
Take Snapshot
CPU Usage
Record CPU Profile
No issues found