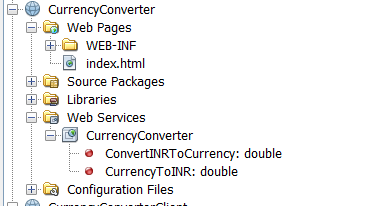
**WEBSERVICE ASSIGNMENT**

NAME: MOHD TARIK FAROOK SHAIKH  
  
ROLL NO: TYCS2425052

**Q1] Differentiate Between SOAP-Based and RESTful Web Services**

| **Feature** | **SOAP-based Web Services** | **RESTful Web Services** |
| --- | --- | --- |
| **Protocol** | Follows a strict protocol (SOAP). | Architectural style (REST) with no strict protocol. |
| **Message Format** | Uses **XML** exclusively. | Supports multiple formats like **JSON**, **XML**, **HTML**. |
| **Complexity** | More complex with rigid standards and extensive features. | Simpler, lightweight, and easier to implement. |
| **Communication** | Works with **SOAP** messages over different protocols (HTTP, SMTP, etc.). | Primarily works over **HTTP** using standard HTTP methods (GET, POST, etc.). |
| **Speed** | Slower due to the verbosity of XML. | Faster, especially when using lightweight formats like **JSON**. |
| **Security** | Built-in standards like **WS-Security** for security. | Relies on underlying protocols like **HTTPS** for security. |
| **State** | Stateless by design, but can maintain state using complex mechanisms. | Typically **stateless**, which makes it easier to scale. |
| **Use Cases** | Ideal for enterprise-level applications needing security and transactions. | Suitable for web-based applications, mobile apps, and public APIs. |
| **Error Handling** | Well-defined standard for error messages (SOAP Faults). | No specific standard; often uses HTTP status codes. |

**2****. Implement a CurrencyConverter( Indian rupees to dollar,euro or Yen and vice versa) application using Web-Services. The application should consist of both server and client parts.**



**Code :**

**CurrencyConvertor.java**

package mypack;

import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

/\*\*

\*

\* @author Mohd Tarik Shaikh

\*/

@WebService(serviceName = "CurrencyConverter")

public class CurrencyConverter {

/\*\*

\* Web service operation

\*/

@WebMethod(operationName = "ConvertINRToCurrency")

public double ConvertINRToCurrency

(@WebParam(name = "currency") String currency,

@WebParam(name = "amount") double amount) throws Exception {

// Convert the currency parameter to uppercase to handle case-insensitive inputs

currency = currency.toUpperCase();

double rate;

// Apply the exchange rate based on the currency

switch (currency) {

case "USD":

rate = 0.012; // 1 INR = 0.012 USD

break;

case "EUR":

rate = 0.011; // 1 INR = 0.011 EUR

break;

case "JPY":

rate = 1.32; // 1 INR = 1.32 JPY

break;

default:

// If an unsupported currency is provided, throw an exception

throw new Exception("Unsupported currency: " + currency);

}

// Calculate and return the converted amount

return amount \* rate;

}

/\*\*

\* Web service operation

\*/

@WebMethod(operationName = "CurrencyToINR")

public double CurrencyToINR(@WebParam(name = "currency") String currency, @WebParam(name = "amount") double amount) {

//TODO write your implementation code here:

// Convert the currency parameter to uppercase to handle case-insensitive inputs

currency = currency.toUpperCase();

double rate;

// Apply the reverse exchange rate to convert from the specified currency to INR

switch (currency) {

case "USD":

rate = 83.30; // 1 USD = 83.30 INR

break;

case "EUR":

rate = 90.50; // 1 EUR = 90.50 INR

break;

case "JPY":

rate = 0.59; // 1 JPY = 0.59 INR

break;

default:

// If an unsupported currency is provided, throw an exception

throw new IllegalArgumentException("Unsupported currency: " + currency);

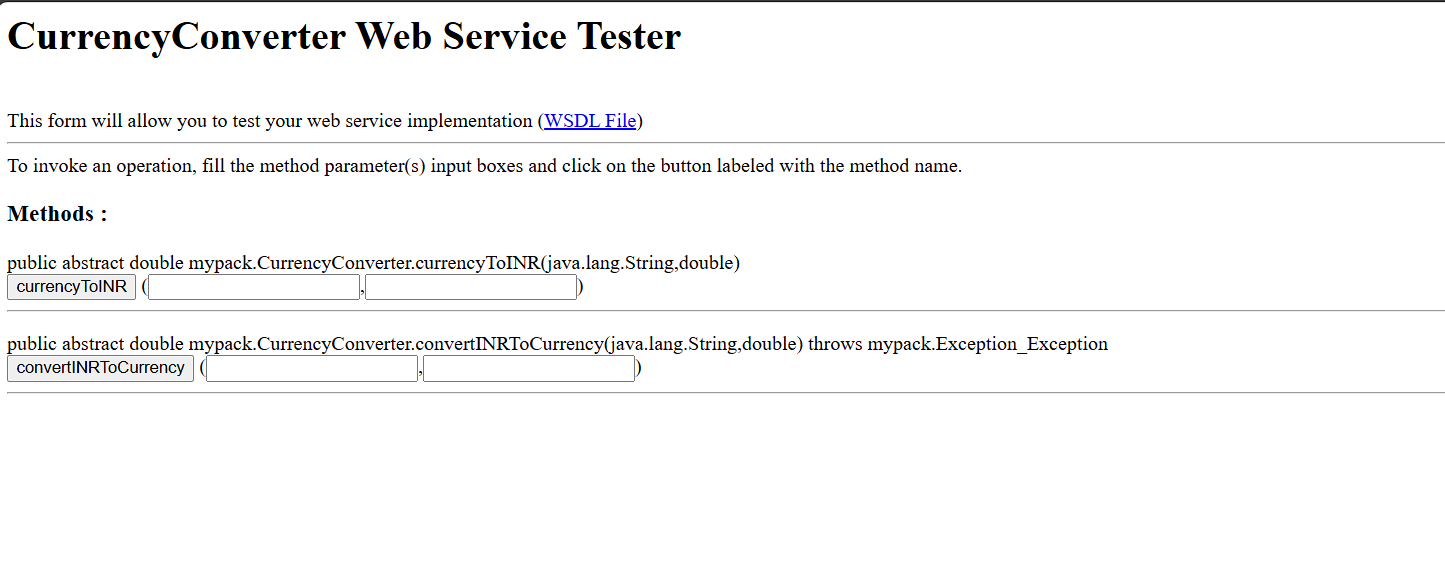
}

// Calculate and return the converted amount in INR

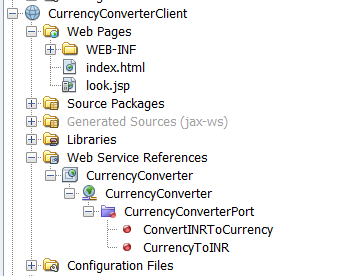
return amount \* rate;

}

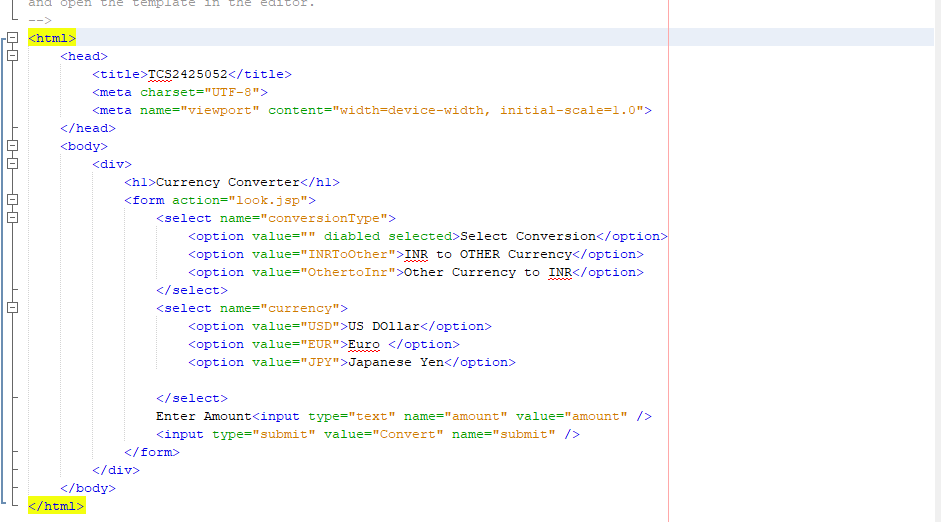
}



**Client Side :**



**Index.html:**

**Bottom of Form**

**Look.jsp:**<%--

Document : look

Created on : 22 Sep, 2024, 12:03:24 PM

Author : Mohd Tarik Shaikh

--%>

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

<!DOCTYPE html>

<html>

<head>

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

<title>JSP Page</title>

</head>

<body>

<h1>Currency Converter</h1>

<h1>Currency to INR and Vice Versa Converter</h1>

<!-- Input Form for Currency Conversion -->

<br>

<%

// Retrieve parameters from the form submission

String conversionType = request.getParameter("conversionType");

String currency = request.getParameter("currency");

String amountStr = request.getParameter("amount");

if (conversionType != null && !conversionType.isEmpty() &&

currency != null && !currency.isEmpty() &&

amountStr != null && !amountStr.isEmpty()) {

try {

// Parse the amount to a double

double amount = Double.parseDouble(amountStr);

// Initialize the web service client

mypack.CurrencyConverter\_Service service = new mypack.CurrencyConverter\_Service();

mypack.CurrencyConverter port = service.getCurrencyConverterPort();

double result = 0.0;

// Determine which conversion to use based on the user's selection

if ("INRToOther".equalsIgnoreCase(conversionType)) {

// Convert INR to the selected currency

result = port.convertINRToCurrency(currency, amount);

out.println("<h2>" + amount + " INR = " + result + " " + currency.toUpperCase() + "</h2>");

} else if ("OthertoInr".equalsIgnoreCase(conversionType)) {

// Convert the selected currency to INR

result = port.currencyToINR(currency, amount);

out.println("<h2>" + amount + " " + currency.toUpperCase() + " = " + result + " INR</h2>");

} else {

out.println("<p>Error: Invalid conversion type selected.</p>");

}

} catch (NumberFormatException e) {

out.println("<p>Error: Invalid amount format. Please enter a valid number for the amount.</p>");

} catch (Exception ex) {

out.println("<p>Error: " + ex.getMessage() + "</p>");

}

} else {

out.println("<p>Error: All fields are required. Please select the conversion type, currency, and enter an amount.</p>");

}

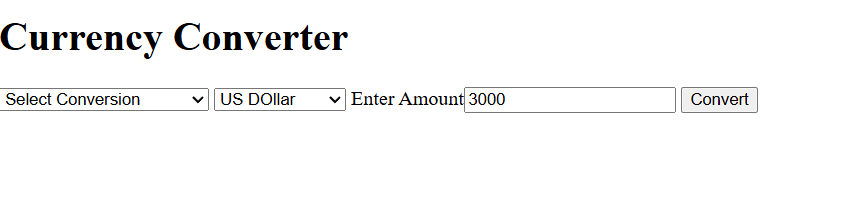
%>

</body>

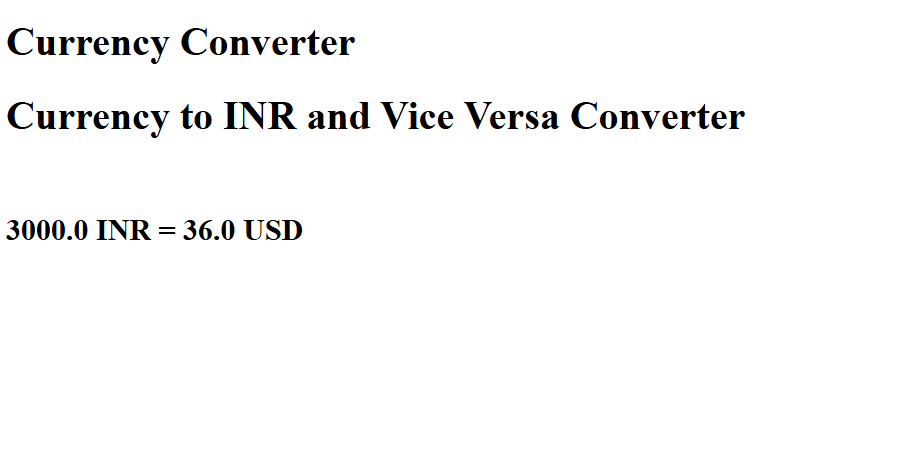
</html>

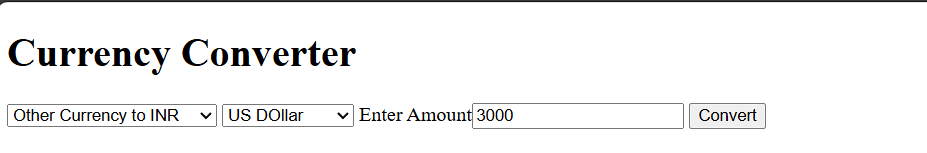
**Bottom of Form**

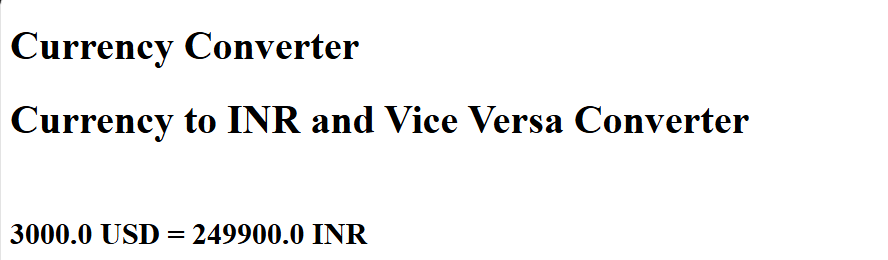
**Bottom of Form**





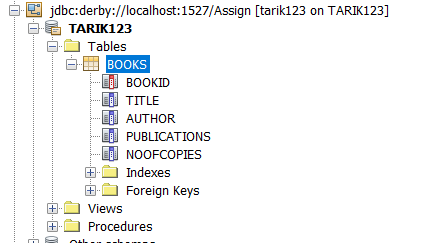




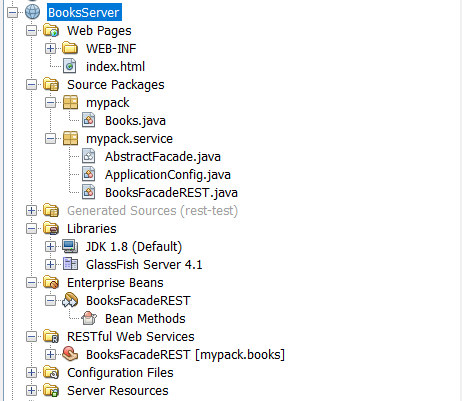


**3. Implement a RESTful Library service (with data Bookid, Title,Author,Publication, No\_of\_Copies). Create a Client that makes a request to get details of all Books. The data accepted and generated by the web service is encoded using JSON.**

**DB Created for library with books as table**



**Created Library Server by adding RestFull services using db**



/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package mypack.service;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.ws.rs.Consumes;

import javax.ws.rs.DELETE;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import mypack.Books;

/\*\*

\*

\* @author Mohd Tarik Shaikh

\*/

@Stateless

@Path("mypack.books")

public class BooksFacadeREST extends AbstractFacade<Books> {

@PersistenceContext(unitName = "BooksServerPU")

private EntityManager em;

public BooksFacadeREST() {

super(Books.class);

}

@POST

@Override

@Consumes({ "application/json"}) //"application/xml",

public void create(Books entity) {

super.create(entity);

}

@PUT

@Path("{id}")

@Consumes({ "application/json"})

public void edit(@PathParam("id") Integer id, Books entity) {

super.edit(entity);

}

@DELETE

@Path("{id}")

public void remove(@PathParam("id") Integer id) {

super.remove(super.find(id));

}

@GET

@Path("{id}")

@Produces({ "application/json"})

public Books find(@PathParam("id") Integer id) {

return super.find(id);

}

@GET

@Override

@Produces({ "application/json"})

public List<Books> findAll() {

return super.findAll();

}

@GET

@Path("{from}/{to}")

@Produces({ "application/json"})

public List<Books> findRange(@PathParam("from") Integer from, @PathParam("to") Integer to) {

return super.findRange(new int[]{from, to});

}

@GET

@Path("count")

@Produces("text/plain")

public String countREST() {

return String.valueOf(super.count());

}

@Override

protected EntityManager getEntityManager() {

return em;

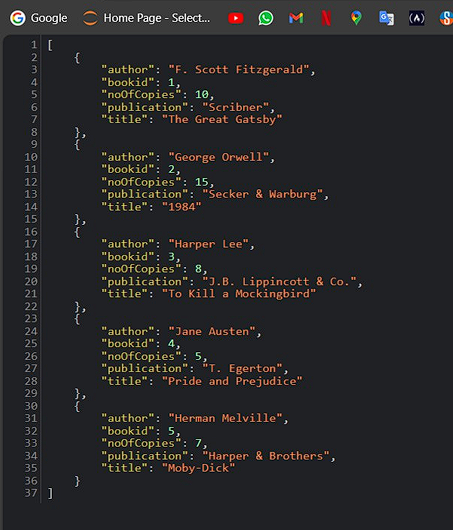
}

}

**Removed application/xml**

**Deployed the server**

**Tested the server**



**Client Side**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Library Books</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

h1 {

text-align: center;

color: seagreen;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

border: 1px solid seagreen;

padding: 5px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

h2 {

text-align: center;

color: seagreen;

}

.error {

color: red;

text-align: center;

margin-top: 20px;

}

</style>

</head>

<body>

<h1>Library Books</h1>

<table id="booksTable">

<thead>

<tr>

<th>Book ID</th>

<th>Title</th>

<th>Author</th>

<th>Publication</th>

<th>No. of Copies</th>

</tr>

</thead>

<tbody>

<!-- Book data will be inserted here -->

</tbody>

</table>

<div class="error" id="errorMessage"></div>

<script>

// Function to fetch book data

function fetchBooks() {

fetch('http://localhost:10450/LibServer/webresources/mks.books')

.then(response => {

if (!response.ok) {

throw new Error('Network response was not ok ' + response.statusText);

}

return response.json();

})

.then(data => {

console.log(data); // Log the response for debugging

const booksTableBody = document.getElementById('booksTable').getElementsByTagName('tbody')[0];

data.forEach(book => {

const row = booksTableBody.insertRow();

row.insertCell(0).textContent = book.bookid || book.id; // Adjust based on your JSON structure

row.insertCell(1).textContent = book.title;

row.insertCell(2).textContent = book.author;

row.insertCell(3).textContent = book.publication;

row.insertCell(4).textContent = book.noOfCopies;

});

})

.catch(error => {

document.getElementById('errorMessage').textContent = 'Error fetching data: ' + error.message;

});

}

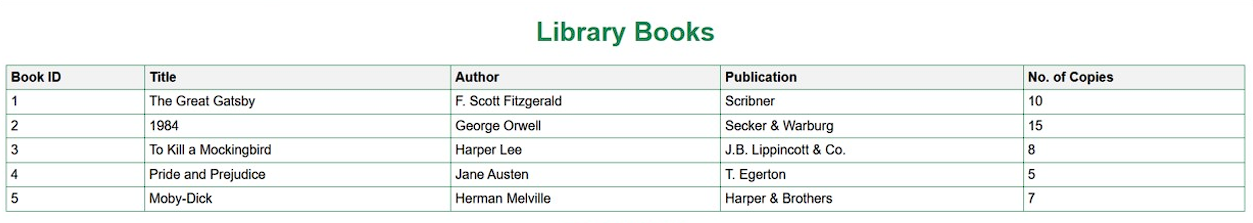
// Fetch books on page load

window.onload = fetchBooks;

</script>

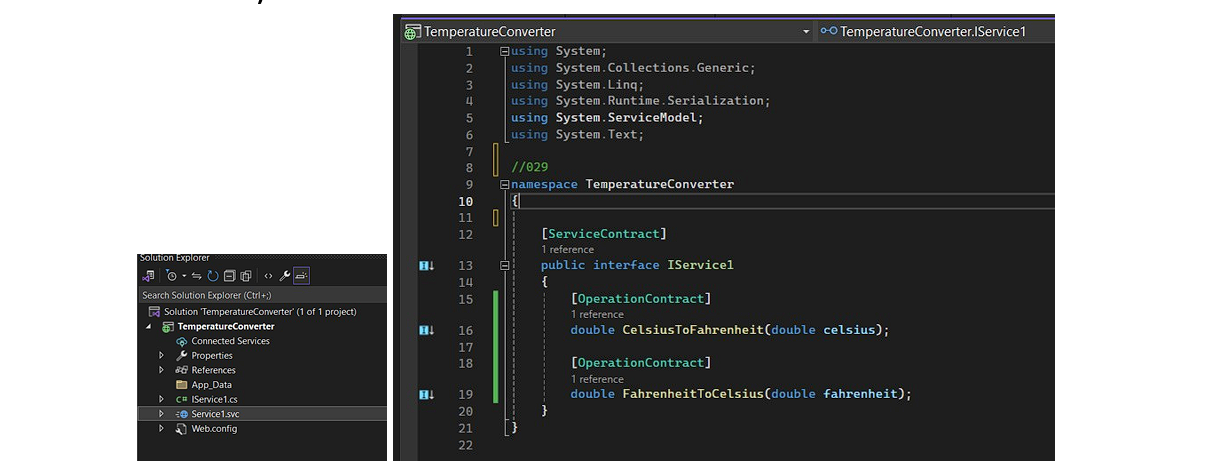
</body>

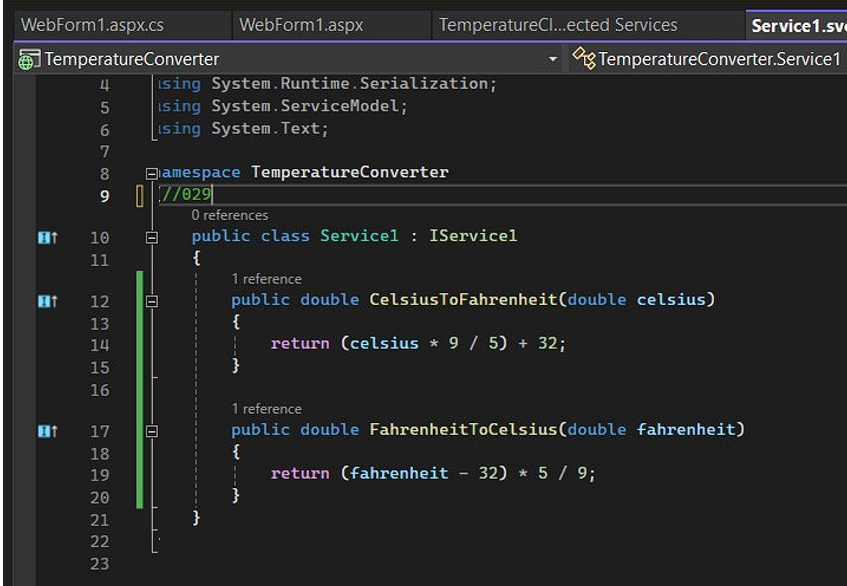
</html>

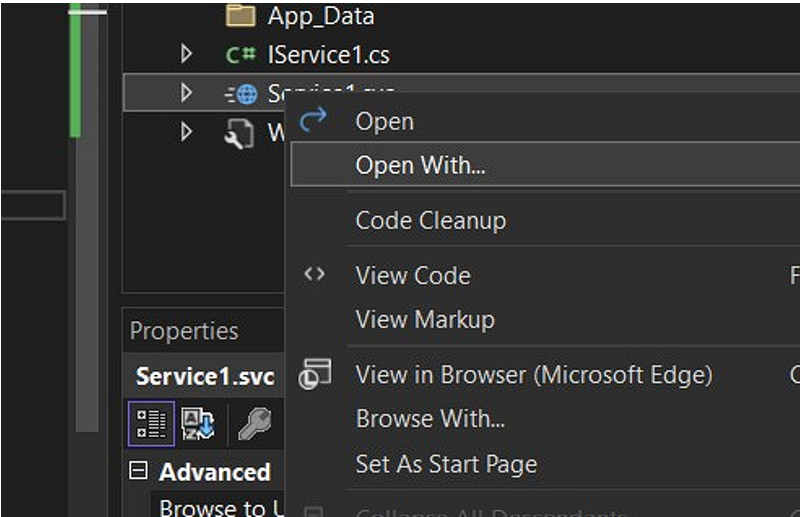


**4. Implement a WCF service TemperatureConverter (2 methods Celsius to Fahrenheit and vice versa). Create a client that uses the functionalities.**

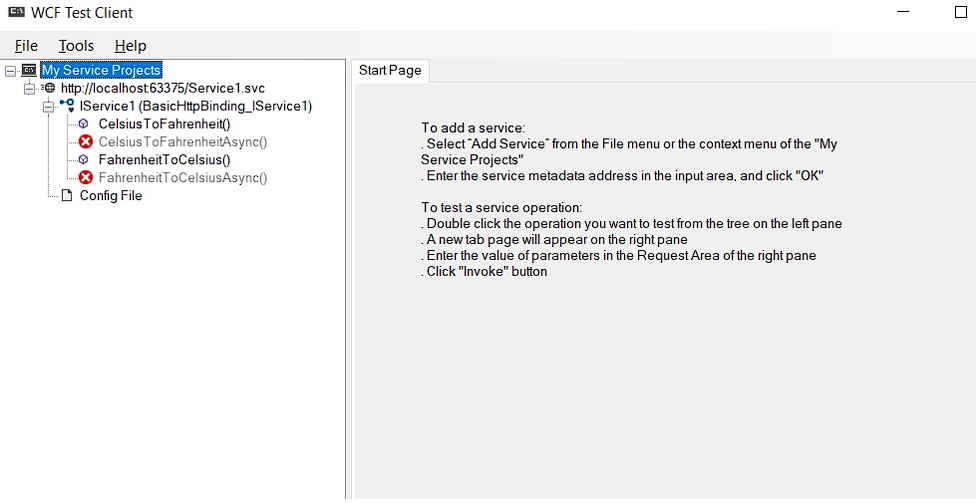
**Server:**

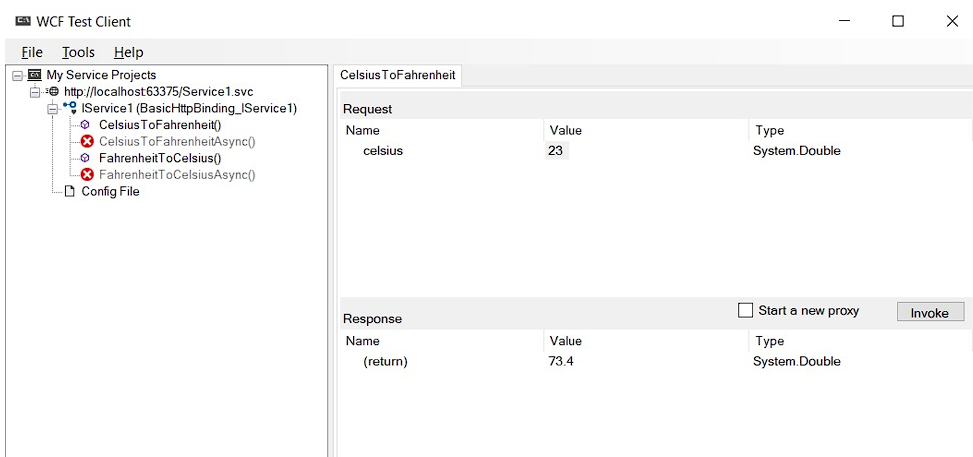
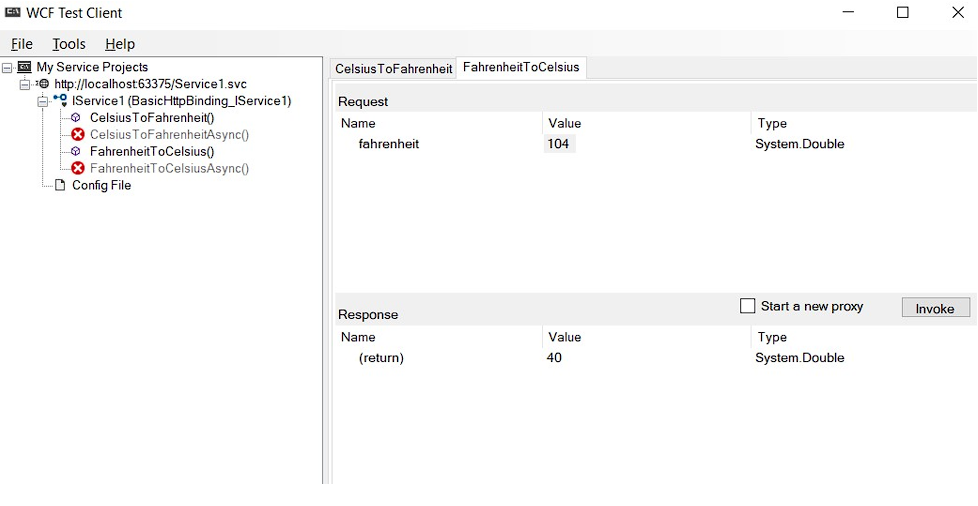






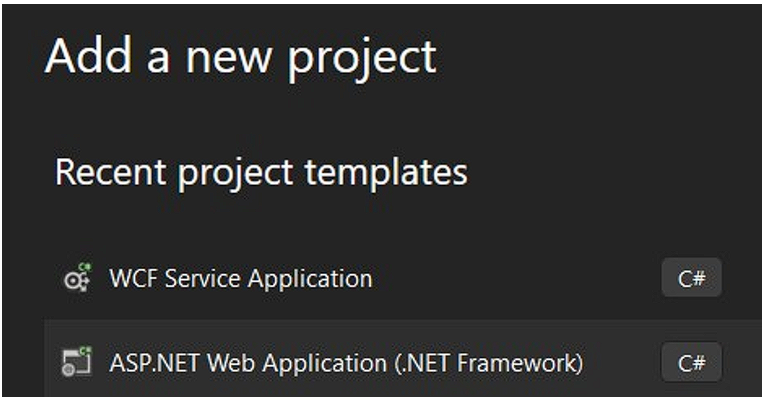
**Test it by F5 or run project**

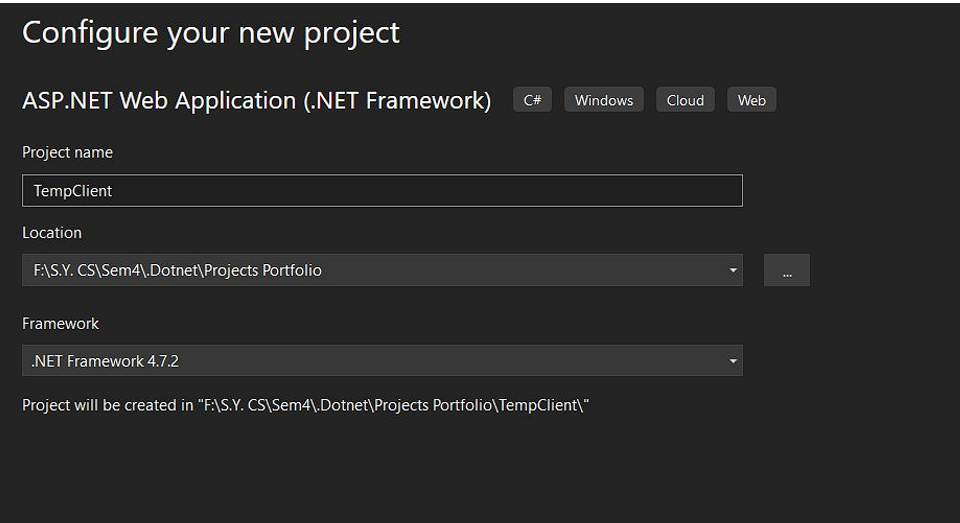


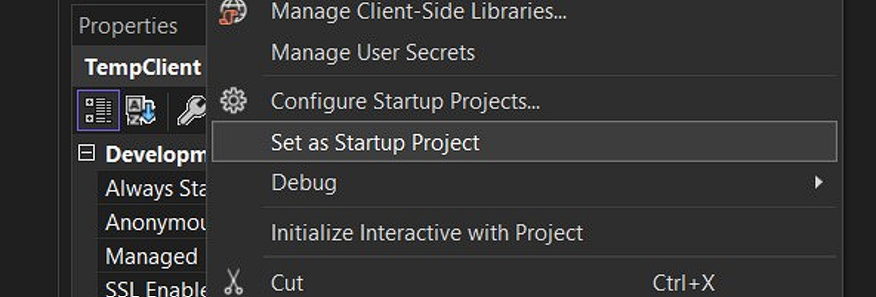
 

**Client:**

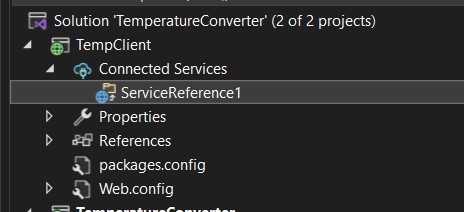
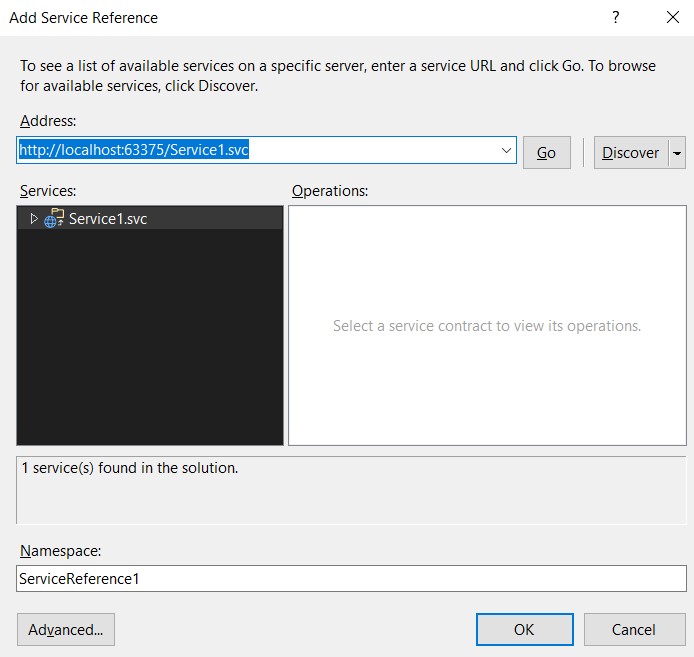
# Create a asp web application







# Add service reference and find server which we created earlier



# Reference file will added Now add a webpage in tempclient Webform1.aspx

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

<!DOCTYPE html>

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

<head runat="server">

<title>Temperature Converter</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h2>Temperature Converter</h2>

<asp:RadioButtonList ID="ConversionType" runat="server">

<asp:ListItem Text="Celsius to Fahrenheit"

Value="CelsiusToFahrenheit"></asp:ListItem>

<asp:ListItem Text="Fahrenheit to Celsius"

Value="FahrenheitToCelsius"></asp:ListItem>

</asp:RadioButtonList>

<br />

<asp:TextBox ID="InputTemperature" runat="server" placeholder="Enter temperature"></asp:TextBox>

<br />

<asp:Button ID="ConvertButton" runat="server" Text="Convert"

OnClick="ConvertButton\_Click" />

<br />

<asp:Label ID="ResultLabel" runat="server" Text=""></asp:Label>

</div>

</form>

</body>

</html>

# Webform1.aspx.cs

using System;

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

using System.Web.Services.Description; using System.Web.UI;

using System.Web.UI.WebControls;

namespace TemperatureClient

{

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

{

protected void ConvertButton\_Click(object sender, EventArgs e)

{

double inputTemperature;

if (double.TryParse(InputTemperature.Text, out inputTemperature))

{

TepmService.Service1Client client = new TepmService.Service1Client();

if (ConversionType.SelectedValue == "CelsiusToFahrenheit")

{

double result = client.CelsiusToFahrenheit(inputTemperature);

ResultLabel.Text = $"{inputTemperature} °C = {result} °F"; }

else if (ConversionType.SelectedValue == "FahrenheitToCelsius")

{

double result = client.FahrenheitToCelsius(inputTemperature);

ResultLabel.Text = $"{inputTemperature} °F = {result} °C";

}

client.Close();

} else

{

ResultLabel.Text = "Please enter a valid temperature.";

}

}

}

}

