Web Services

Journal

Index

Practical 1: Write a program to create a calculator with Remote Method Invocation (RMI).

CalculatorInterface.java

```
import java.rmi.*;

public interface CalculatorInterface extends Remote {
	public double add(double x, double y) throws RemoteException;
	public double sub(double x, double y) throws RemoteException;
	public double multiply(double x, double y) throws
RemoteException;
	public double divide(double x, double y) throws
RemoteException;
}
```

CalculatorRemote.java

```
import java.rmi.*;
import java.rmi.server.*;

public class CalculatorRemote extends UnicastRemoteObject implements
CalculatorInterface{
	CalculatorRemote() throws RemoteException{
		super();
	}
	public double add(double x, double y) {
		return x + y;
	}
	public double sub(double x, double y) {
		return x - y;
	}
	public double multiply(double x, double y) {
		return x * y;
	}
	public double divide(double x, double y) {
		return x/y;
	}
}
```

CalculatorServer.java

```
import java.rmi.*;
import java.rmi.server.*;

public class CalculatorServer {
	public static void main(String[] args) {
		try {
			CalculatorInterface obj = new CalculatorRemote();
			Naming.rebind("rmi://localhost:8888/send", obj);
		} catch(Exception e) {
			System.out.println(e);
		}
	}
}
```

CalculatorClient.java

```
import java.rmi.*;
import java.util.Scanner;

public class CalculatorClient {
	public static void main(String[] args) {
		try {
			String op;
			double res;
			CalculatorInterface obj =
(CalculatorInterface)Naming.lookup("rmi://localhost:8888/send");
			Scanner sc = new Scanner(System.in);
			System.out.println("Enter First Number: ");
			double num1 = sc.nextDouble();
			System.out.println("Enter Second Number: ");
			double num2 = sc.nextDouble();
			sc.nextLine();
			System.out.println("Enter Operator: \n+\n-\n*\n/");
			op = sc.nextLine();
			switch(op) {
				case "+":
					res = obj.add(num1, num2);
					System.out.println("Result: " + res);
```

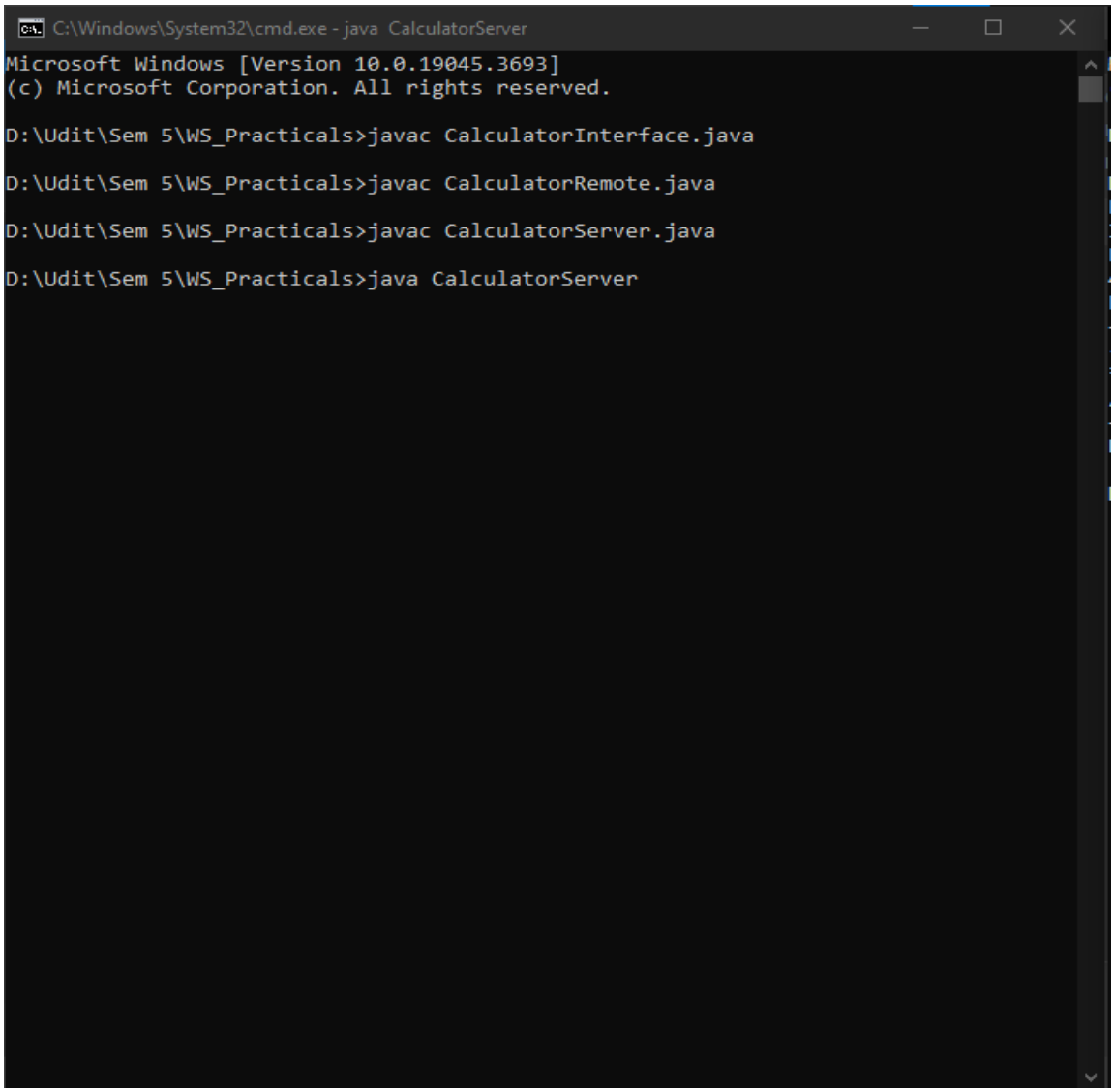
```
                        break;
                    case "-":
                        res = obj.sub(num1, num2);
                        System.out.println("Result: " + res);
                        break;
                    case "*":
                        res = obj.multiply(num1, num2);
                        System.out.println("Result: " + res);
                        break;
                    case "/":
                        res = obj.divide(num1, num2);
                        System.out.println("Result: " + res);
                        break;
                }
            } catch(Exception e) {
                System.out.println(e);
            }
        }
}
```

Steps:

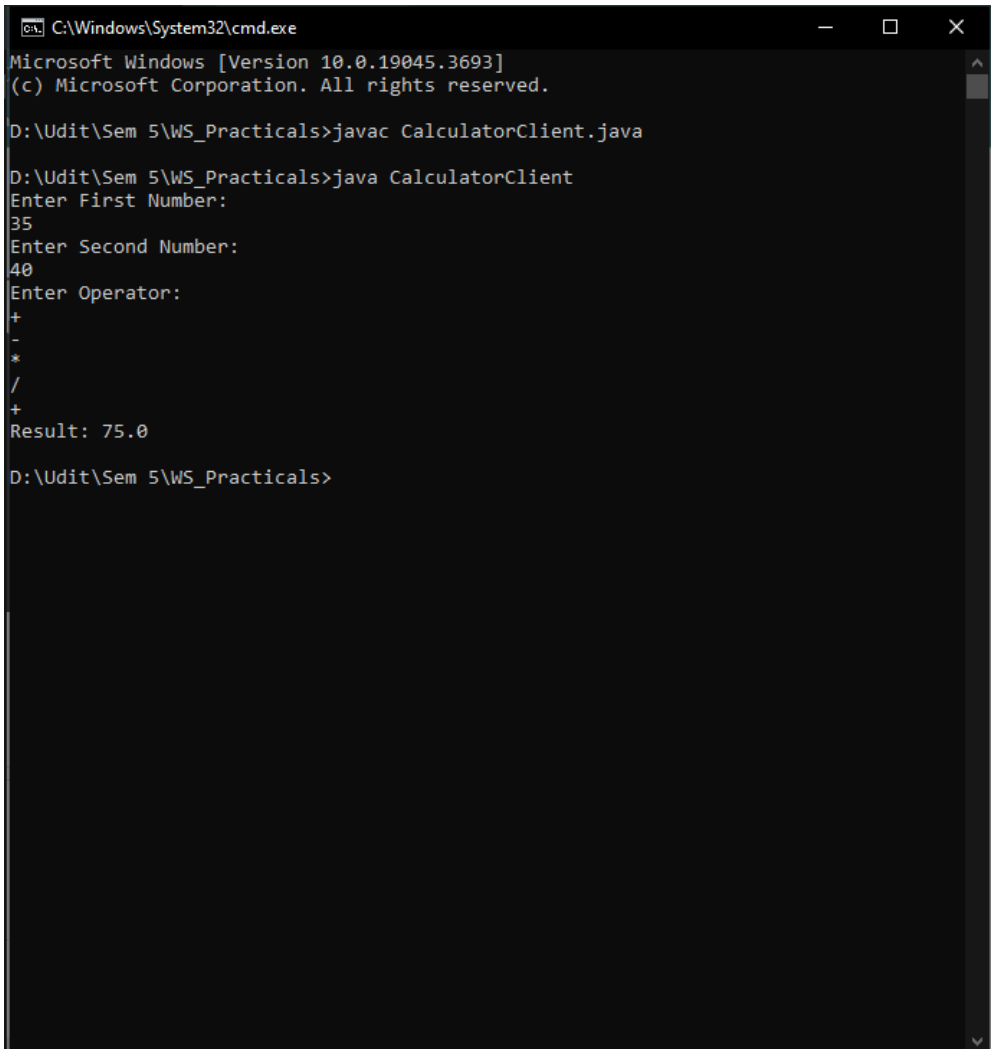
1. Open 3 different command prompts.
2. Type the following on one of them:

   rmiregistry 8888

3. Type the following on another command prompt:



4. Type the following on another command prompt:



Practical 2: Write a program to implement Hello username web service.

Creating a HelloService Web Service:

1. Open Netbeans 8.2
2. Open New Project -> Java Web -> Web Application
3. Type Project Name HelloWorld
4. Right-click on the project name -> New -> Web Service
5. Enter name and package and Finish

WSPract2.java

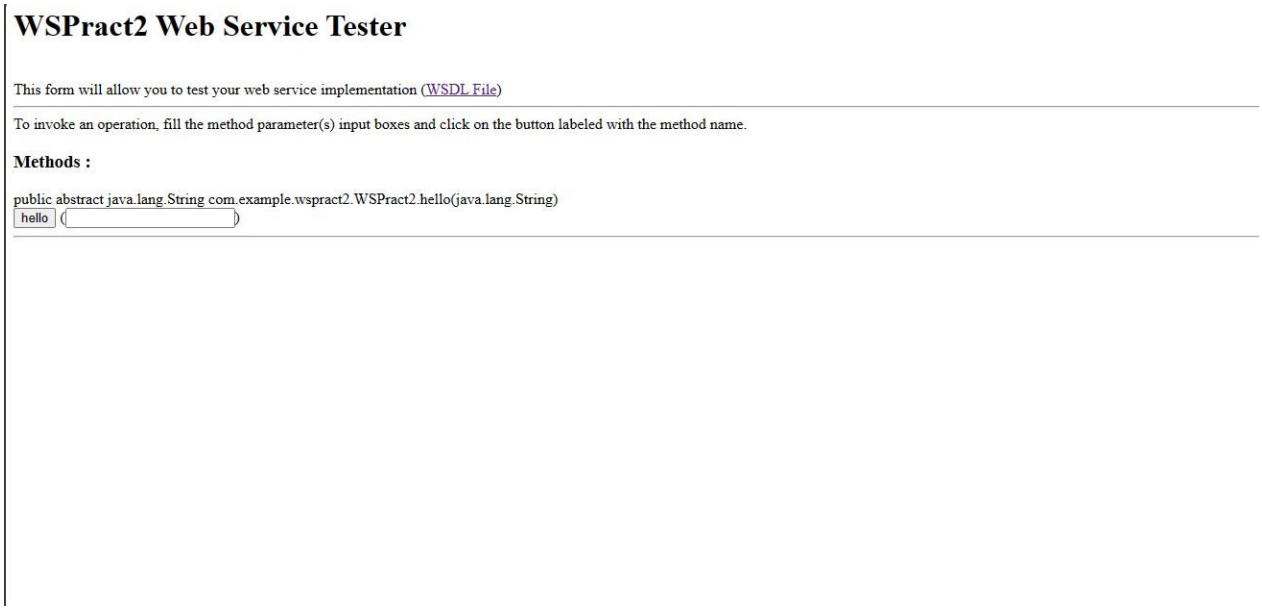
```java
package com.example.WSPract2;

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

/**
 *
 * @author damle
 */
@WebService(serviceName = "WSPract2")
public class WSPract2 {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "username") String user) {
        return "Hello " + user + "!";
    }
}
```

6. Right-click on the project name, select clean and build.
7. Right-click on the project name and deploy.
8. Right-click on the web service file, and select Test Web Service.

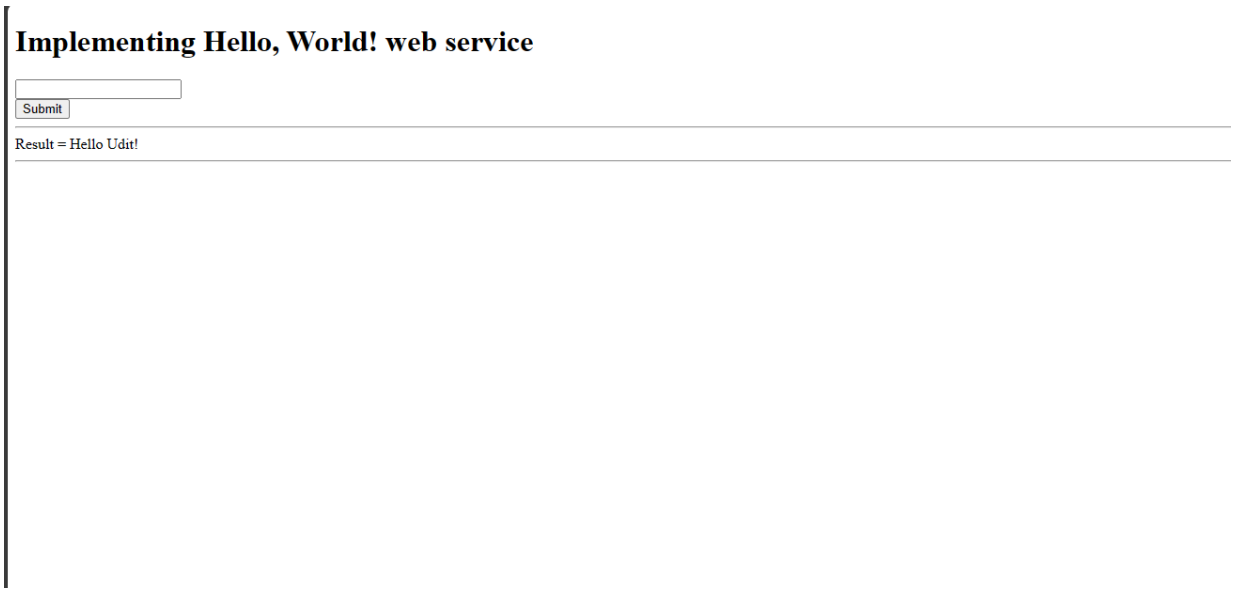


9. Click on WSDL File -> copy URL
10. Create another project (Client).
11. Click on Project name -> New -> Web Service Client
12. Select the WSDL URL option and paste the WSDL URL copied before
13. Create index.jsp file
14. Right-click and select Web Service Client Resources -> Call Web Service Operation
15. Expand until you see hello operation
16. Select the hello operation and the code will automatically be added to the JSP page

```
<%@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>Implementing Hello, World! web service</h1>
        <form action="index.jsp">
            <input type="text" name="user">
            <br>
            <input type="submit" name="hello">
            <br>
        </form>
```

```
    <%-- start web service invocation --%><hr/>
    <%
    try {
      com.example.wspract2.WSPract2_Service service = new
com.example.wspract2.WSPract2_Service();
      com.example.wspract2.WSPract2 port = service.getWSPract2Port();
       // TODO initialise WS operation arguments here
      java.lang.String username = "";
        username = request.getParameter("user");
      // TODO process result here
      java.lang.String result = port.hello(username);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

17. Right-click on JSP file and select Run file



Practical 3: Write a program to implement a simple Web Service that converts the temperature from Fahrenheit to Celsius and vice versa.

Creating a Temperature Conversion Web Service:

1. Open Netbeans 8.2
2. Open New Project -> Java Web -> Web Application
3. Type Project Name HelloWorld
4. Right-click on the project name -> New -> Web Service
5. Enter name and package and Finish
6. Go to Design View & remove the default template operation.
7. Click on Add Operation.
8. Enter the name of the operation(F_to_C), return type and parameters -> Click OK.
9. Do the same for C_to_F.

temp_service.java

```java
package com.example;

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

/**
 *
 * @author damle
 */
@WebService(serviceName = "temp_service")
public class temp_service {

    /**
     * This is a sample web service operation
     */


    /**
     * Web service operation
     */
    @WebMethod(operationName = "F_to_C")
    public Double F_to_C(@WebParam(name = "f") double f) {
        Double c = (f-32)*1.8;
        return c;
```

```
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "C_to_F")
    public Double C_to_F(@WebParam(name = "c") double c) {
        Double f = (c*1.8)+32;
        return f;
    }
}
```

10. Right-click on the project name, select clean and build.
11. Right-click on the project name and deploy.
12. Right-click on the web service file, and select Test Web Service.
13. Click on WSDL File -> copy URL
14. Create another project (Client).
15. Click on Project name -> New -> Web Service Client
16. Select the WSDL URL option and paste the WSDL URL copied before
17. Create index.html file:

```
<html>
<head>
<title>Temperature converter</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
</head
<body>
<form>
<input type="text" name="data"><br>
<input type="submit" value="Convert F to C" name="ftoc"
formaction="f_to_c.jsp"><br>
<input type="submit" value="Convert C to F" name="ctof"
formaction="c_to_f.jsp">
</form>
</body>
</html>
```

18. Create two JSP files named f_to_c.jsp & c_to_f.jsp

For both files:

19. Right-click and select Web Service Client Resources -> Call Web Service Operation
20. Expand until you see the appropriate operation.
21. Select the operation and the code will automatically be added to the JSP page.

c_to_f.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title></title>
    </head>
    <body>
        <h1>C to F</h1>

    <%-- start web service invocation --%><hr/>
    <%
        String d = request.getParameter("data");
        int dd = Integer.parseInt(d);
    try {
      com.example.TempService_Service service = new
com.example.TempService_Service();
      com.example.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>
```

f_to_c.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title></title>
    </head>
    <body>
        <h1>F To C</h1>

    <%-- start web service invocation --%><hr/>
    <%
        String d = request.getParameter("data");
        int dd = Integer.parseInt(d);
    try {
      com.example.TempService_Service service = new
com.example.TempService_Service();
      com.example.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>
```

Practical 4: Write a program to implement a simple Web Service to make a calculator with the simple 4 operations (Addition, Subtraction, Multiplication, Division).

Creating a Temperature Conversion Web Service:
1. Open Netbeans 8.2
2. Open New Project -> Java Web -> Web Application
3. Type Project Name HelloWorld
4. Right-click on the project name -> New -> Web Service
5. Enter name and package and Finish
6. Go to Design View & remove the default template operation.
7. Click on Add Operation.
8. Enter the name of the operation(add), return type and parameters(double) -> Click OK.
9. Do the same for sub, multiply & divide.

Calculator.java

```
package com.example;

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

/**
 *
 * @author damle
 */
@WebService(serviceName = "Calculator")
public class Calculator {


    /**
     * Web service operation
     */
    @WebMethod(operationName = "add")
    public Double add(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 + num2;
    }
```

```
    /**
     * Web service operation
     */
    @WebMethod(operationName = "sub")
    public Double sub(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 - num2;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "mul")
    public Double mul(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 * num2;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "div")
    public Double div(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 / num2;
    }
}
```

10. Right-click on the project name, select clean and build.
11. Right-click on the project name and deploy.
12. Right-click on the web service file, and select Test Web Service.
13. Click on WSDL File -> copy URL.
14. Create another project (Client).
15. Click on Project name -> New -> Web Service Client
16. Select the WSDL URL option and paste the WSDL URL copied before

17. Create index.html file:

```
<!DOCTYPE html>
<!--
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.
-->
<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>
        <h1>Calculator</h1>
        <h2>Enter Two Numbers:</h2>
        <form>
            <input type="text" name="num1"><br>
            <input type="text" name="num2"><br><br><br>
            <input type="submit" value="Add" formaction="add.jsp">
            <input type="submit" value="Subtract"
formaction="sub.jsp"><br>
            <input type="submit" value="Mutiply"
formaction="mul.jsp">
            <input type="submit" value="Divide" formaction="div.jsp">
        </form>
    </body>
</html>
```

18. Create four JSP files named: add.jsp, sub.jsp, mul.jsp, and div.jsp.

For allfiles:

19. Right-click and select Web Service Client Resources -> Call Web Service Operation
20. Expand until you see the appropriate operation.
21. Select the operation and the code will automatically be added to the JSP page.

add.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Addition</title>
    </head>
    <body>
        <h1>Sum Result</h1>

    <%-- start web service invocation --%><hr/>
    <%
        String snum1 = request.getParameter("num1");
        String snum2 = request.getParameter("num2");
    try {
      com.example.Calculator_Service service = new
com.example.Calculator_Service();
      com.example.Calculator port = service.getCalculatorPort();
       // TODO initialize WS operation arguments here
      double num1 = Integer.parseInt(snum1);
      double num2 = Integer.parseInt(snum2);
      // TODO process result here
      java.lang.Double result = port.add(num1, num2);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

sub.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Subtraction</title>
    </head>
    <body>
        <h1>Subtraction Result: </h1>

    <%-- start web service invocation --%><hr/>
    <%
        String snum1 = request.getParameter("num1");
        String snum2 = request.getParameter("num2");
    try {
      com.example.Calculator_Service service = new
com.example.Calculator_Service();
      com.example.Calculator port = service.getCalculatorPort();
       // TODO initialize WS operation arguments here
      double num1 = Integer.parseInt(snum1);
      double num2 = Integer.parseInt(snum2);
      // TODO process result here
      java.lang.Double result = port.sub(num1, num2);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

mul.jsp

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

```
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Multiplication</title>
    </head>
    <body>
        <h1>Multiplication Result: </h1>

    <%-- start web service invocation --%><hr/>
    <%
        String snum1 = request.getParameter("num1");
        String snum2 = request.getParameter("num2");

        try {
      com.example.Calculator_Service service = new
com.example.Calculator_Service();
      com.example.Calculator port = service.getCalculatorPort();
       // TODO initialize WS operation arguments here
      double num1 = Integer.parseInt(snum1);
      double num2 = Integer.parseInt(snum2);
      // TODO process result here
      java.lang.Double result = port.mul(num1, num2);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

div.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
```

```
        <title>Division</title>
    </head>
    <body>
        <h1>Division Result: </h1>

    <%-- start web service invocation --%><hr/>
    <%
        String snum1 = request.getParameter("num1");
        String snum2 = request.getParameter("num2");
    try {
      com.example.Calculator_Service service = new
com.example.Calculator_Service();
      com.example.Calculator port = service.getCalculatorPort();
       // TODO initialize WS operation arguments here
      double num1 = Integer.parseInt(snum1);
      double num2 = Integer.parseInt(snum2);
      // TODO process result here
      java.lang.Double result = port.div(num1, num2);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

22. Lastly, run index.html file.

Practical 5: Write a program to implement a simple Web service to make a calculator with simple 2 operations (Addition and subtraction) and create a Web application as a Web Service Client that consumes the service.

Creating a Temperature Conversion Web Service:

1. Open Netbeans 8.2
2. Open New Project -> Java Web -> Web Application
3. Type Project Name HelloWorld
4. Right-click on the project name -> New -> Web Service
5. Enter name and package and Finish
6. Go to Design View & remove the default template operation.
7. Click on Add Operation.
8. Enter the name of the operation(add), return type and parameters(double) -> Click OK.
9. Do the same for sub.

Calculator.java

```
package com.example;

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

/**
 *
 * @author damle
 */
@WebService(serviceName = "Calculator")
public class Calculator {


    /**
     * Web service operation
     */
    @WebMethod(operationName = "add")
    public Double add(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 + num2;
```

```
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "sub")
    public Double sub(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 - num2;
    }
}
```

10. Right-click on the project name, select clean and build.
11. Right-click on the project name and deploy.
12. Right-click on the web service file, and select Test Web Service.
13. Click on WSDL File -> copy URL.
14. Create another project (Client).
15. Click on Project name -> New -> Web Service Client
16. Select the WSDL URL option and paste the WSDL URL copied before
17. Create index.html file:

```
<!DOCTYPE html>
<!--
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.
-->
<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>
        <h1>Calculator</h1>
        <h2>Enter Two Numbers:</h2>
        <form>
```

```
            <input type="text" name="num1"><br>
            <input type="text" name="num2"><br><br><br>
            <input type="submit" value="Add" formaction="add.jsp">
            <input type="submit" value="Subtract"
formaction="sub.jsp"><br>
            <input type="submit" value="Mutiply"
formaction="mul.jsp">
            <input type="submit" value="Divide" formaction="div.jsp">
        </form>
    </body>
</html>
```

18. Create four JSP files named: add.jsp, sub.jsp, mul.jsp, and div.jsp.

For all files:

19. Right-click and select Web Service Client Resources -> Call Web Service Operation
20. Expand until you see the appropriate operation.
21. Select the operation and the code will automatically be added to the JSP page.

add.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Addition</title>
    </head>
    <body>
        <h1>Sum Result</h1>

    <%-- start web service invocation --%><hr/>
    <%
        String snum1 = request.getParameter("num1");
        String snum2 = request.getParameter("num2");
    try {
      com.example.Calculator_Service service = new
com.example.Calculator_Service();
```

```
      com.example.Calculator port = service.getCalculatorPort();
       // TODO initialize WS operation arguments here
      double num1 = Integer.parseInt(snum1);
      double num2 = Integer.parseInt(snum2);
      // TODO process result here
      java.lang.Double result = port.add(num1, num2);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

sub.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Subtraction</title>
    </head>
    <body>
        <h1>Subtraction Result: </h1>

    <%-- start web service invocation --%><hr/>
    <%
        String snum1 = request.getParameter("num1");
        String snum2 = request.getParameter("num2");
    try {
      com.example.Calculator_Service service = new
com.example.Calculator_Service();
      com.example.Calculator port = service.getCalculatorPort();
       // TODO initialize WS operation arguments here
      double num1 = Integer.parseInt(snum1);
      double num2 = Integer.parseInt(snum2);
```

```
      // TODO process result here
      java.lang.Double result = port.sub(num1, num2);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

22. Lastly, run index.html file.

Practical 6: Write a program to implement the WCF to create a hello username service.

1. Open Visual Studio -> New Project -> WCF Service Application
2. Go to 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 WcfService1
{
    [ServiceContract]
    public interface IService1
    {
```

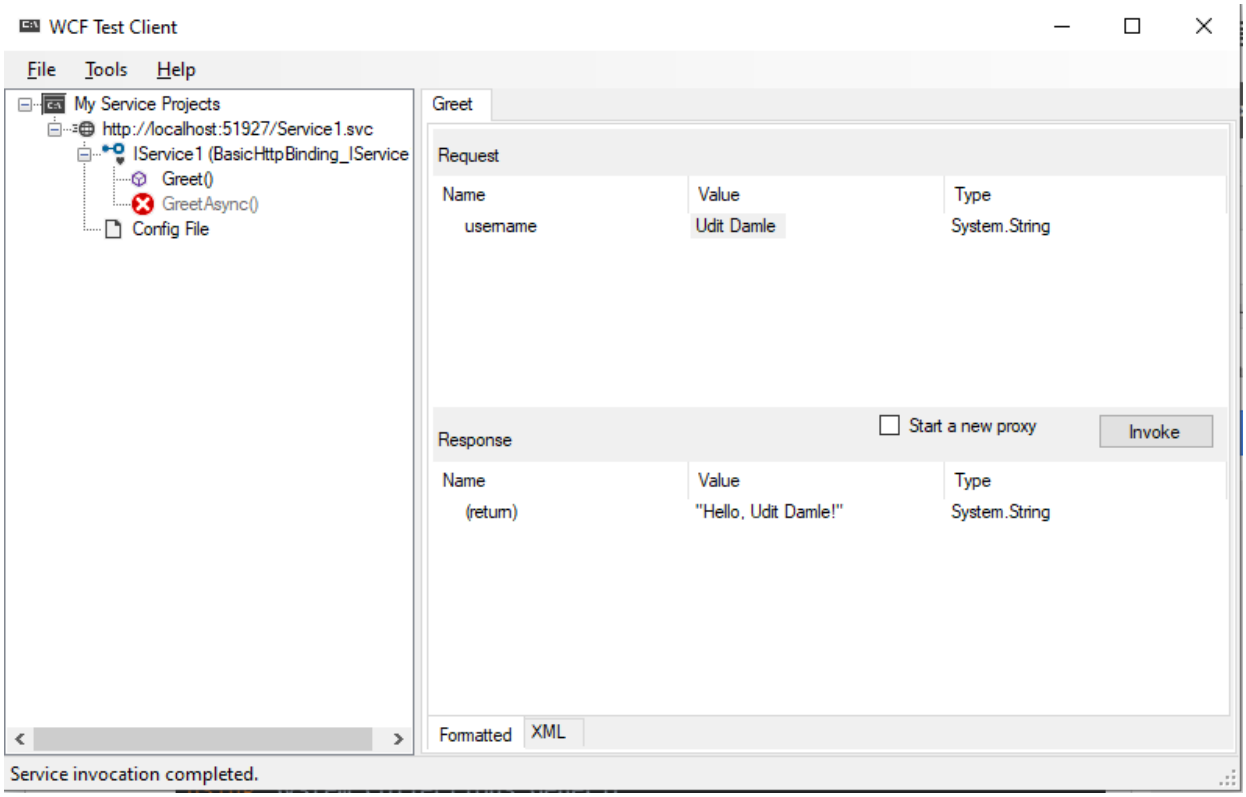
```
        [OperationContract]
        string Greet(string username);
    }
}
```

3. Go to Service1.svc.cs & write the implementation logic:

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

namespace WcfService1
{
    public class Service1 : IService1
    {
        string IService1.Greet(string username)
        {
            return $"Hello, {username}!";
        }
    }
}
```

4. Open Service1.svc.cs and click on the run icon to open WCF Test Client.


WCF Test Client
File
Tools
Help
My Service Projects
http://localhost:51927/Service1.svc
IService1 (BasicHttpBinding_IService
Greet()
GreetAsync()
Config File
Greet
Request
Name
Value
Type
username
Udit Damle
System.String
Response
Start a new proxy
Invoke
Name
Value
Type
(return)
"Hello, Udit Damle!"
System.String
Formatted
XML
Service invocation completed.

Practical 7: Write a program to implement the WCF to convert temperature from Fahrenheit to Celsius service and create a client using WCF.

1. Open Visual Studio -> New Project -> WCF Service Application
2. Go to 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 WCFTempConv
{
    [ServiceContract]
    public interface IService1
    {

        [OperationContract]
        double CtoF(double cel);

        [OperationContract]
        double FtoC(double fah);

    }
}
```

3. Go to Service1.svc.cs & write the implementation logic:

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

namespace WCFTempConv
```

```
{
    public class Service1 : IService1
    {
        double IService1.CtoF(double cel)
        {
            return (cel * 9)/5 + 32;
        }

        double IService1.FtoC(double fah)
        {
            return (fah - 32) * 5 / 9;
        }
    }
}
```

4. Open Service1.svc.cs and click on the run icon to open WCF Test Client.
5. Right-click on Solution name -> new Project -> Web Application
6. Create web form (WCFClient.aspx)
7. Add two Textfields and 2 buttons.
8. Right-click on project name -> Add -> service reference & enter address of Web Service.
9. Select the service & click on OK.

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Enter Temperature: <asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox> <br />
            <asp:Button ID="Button1" runat="server" Text="Convert to
Fahrenheit" OnClick="Button1_Click" /> <br />
            <asp:Button ID="Button2" runat="server" Text="Convert to
```

```
Celcius" OnClick="Button2_Click" /><br /><br />
            Result: <asp:TextBox ID="TextBox2"
runat="server"></asp:TextBox>
        </div>
    </form>
</body>
</html>
```

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

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

        }

        ServiceReference1.Service1Client client = new
ServiceReference1.Service1Client();
        protected void Button1_Click(object sender, EventArgs e)
        {
            double val = double.Parse(TextBox1.Text);
            TextBox2.Text = client.CtoF(val).ToString();
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            double val = double.Parse(TextBox1.Text);
            TextBox2.Text = client.FtoC(val).ToString();
        }
    }
```

```
}
```

1. Open Visual Studio -> New Project -> WCF Service Application
2. Go to 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
{
    // 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]
        double add(double num1, double num2);

        [OperationContract]
        double sub(double num1, double num2);

        [OperationContract]
        double mul(double num1, double num2);

        [OperationContract]
        double div(double num1, double num2);
    }
}
```

3. Go to Service1.svc.cs & write the implementation logic:

```
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
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu
to change the class name "Service1" in code, svc and config file
together.
    // NOTE: In order to launch WCF Test Client for testing this
service, please select Service1.svc or Service1.svc.cs at the
Solution Explorer and start debugging.
    public class Service1 : IService1
    {
        double IService1.add(double num1, double num2)
        {
            return num1 + num2;
        }
```

```
        double IService1.sub(double num1, double num2)
        {
            return num1 - num2;
        }

        double IService1.mul(double num1, double num2)
        {
            return num1 * num2;
        }

        double IService1.div(double num1, double num2)
        {
            return num1/num2;
        }
    }
}
```

Practical 8: Write a program to implement the Java Web Service to create a calculator (Add and Subtract). Also, make a service consumer that consumes the service in the WCF.

Creating a Temperature Conversion Web Service:
1. Open Netbeans 8.2
2. Open New Project -> Java Web -> Web Application
3. Type Project Name HelloWorld
4. Right-click on the project name -> New -> Web Service
5. Enter name and package and Finish
6. Go to Design View & remove the default template operation.
7. Click on Add Operation.
8. Enter the name of the operation(add), return type and parameters(double) -> Click OK.
9. Do the same for sub.

```
package com.example;

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

/**
 *
 * @author damle
 */
@WebService(serviceName = "Calculator")
public class Calculator {


    /**
     * Web service operation
     */
    @WebMethod(operationName = "add")
    public Double add(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 + num2;
    }
```

```
    /**
     * Web service operation
     */
    @WebMethod(operationName = "sub")
    public Double sub(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 - num2;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "mul")
    public Double mul(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 * num2;
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "div")
    public Double div(@WebParam(name = "num1") double num1,
@WebParam(name = "num2") double num2) {
        //TODO write your implementation code here:
        return num1 / num2;
    }
}
```

10. Right-click on the project name, select clean and build.
11. Right-click on the project name and deploy.
12. Right-click on the web service file, and select Test Web Service.
13. Click on WSDL File -> copy URL.
14. Create another project in Visual Studio.
15. Click on Project name -> New -> Web Application.
16. Click on Project name-> add -> service reference
17. Paste the WSDL URL in the address textfield and click Go.

18. Give the appropriate name in the namespace and click OK.

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

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

        }

        ServiceReference2.SimpleCalculatorClient client = new
ServiceReference2.SimpleCalculatorClient();
        protected void Button1_Click(object sender, EventArgs e)
        {
            double num1 = double.Parse(TextBox1.Text);
            double num2 = double.Parse(TextBox2.Text);
            TextBox3.Text = client.add(num1, num2).ToString();
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            double num1 = double.Parse(TextBox1.Text);
            double num2 = double.Parse(TextBox2.Text);
            TextBox3.Text = client.sub(num1, num2).ToString();
        }
}
```

Practical 9: Design a web service for the college containing two functions:

- **1st function:** getCutoff() accepts the course name as a parameter and returns last year's cut-off for that course as a floating-point number.
- **2nd function:** getFees() accepts course name as a parameter and returns this year's course fees as floating-point numbers.
- Design a client to test the above web service.

collegeinfo.java

```
package com.example;

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

/**
 *
 * @author damle
 */
@WebService(serviceName = "Pract9")
public class Pract9 {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "getCutOff")
    public double getCutOff(@WebParam(name = "courseName") String
courseName) {
        //TODO write your implementation code here:
        double cutOff;
        if(courseName.equals("BCA")) {
            cutOff = 70;
        } else if(courseName.equals("BScIT")) {
            cutOff = 75;
        } else if(courseName.equals("BScITHons")) {
            cutOff = 80;
        } else {
            cutOff = 0;
        }
        return cutOff;
```

```
    }

    /**
     * Web service operation
     */
    @WebMethod(operationName = "getFees")
    public double getFees(@WebParam(name = "courseName") String
courseName) {
        //TODO write your implementation code here:
        double fees;
        if(courseName.equals("BCA")) {
            fees = 100000;
        } else if(courseName.equals("BScIT")) {
            fees = 110000;
        } else if(courseName.equals("BScITHons")) {
            fees = 120000;
        } else {
            fees = 0;
        }
        return fees;
    }
}
```

index.html

```
<!DOCTYPE html>
<!--
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.
-->
<html>
    <head>
        <title>College Web Service</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    </head>
```

```
    <body>
        <form>
            <input type="Text" name="course"><br>
            <input type="Submit" value="Get Cut-off"
formaction="cutoff.jsp"><br>
            <input type="Submit" value="Get fees"
formaction="fees.jsp"><br>
        </form>
    </body>
</html>
```

cutoff.jsp

```
<%--
    Document   : cutoff
    Created on : 27 Nov, 2023, 9:42:56 PM
    Author     : damle
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>cutoff</title>
    </head>
    <body>
        <h1>Cut-off</h1>

    <%-- start web service invocation --%><hr/>
    <%
    try {
      com.example.Pract9_Service service = new
com.example.Pract9_Service();
      com.example.Pract9 port = service.getPract9Port();
       // TODO initialize WS operation arguments here
      java.lang.String courseName = request.getParameter("course");
      // TODO process result here
```

```
      double result = port.getCutOff(courseName);
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

fees.jsp

```
<%--
    Document   : fees
    Created on : 27 Nov, 2023, 9:50:31 PM
    Author     : damle
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Fees</title>
    </head>
    <body>
        <h1>Fees</h1>

    <%-- start web service invocation --%><hr/>
    <%
    try {
      com.example.Pract9_Service service = new
com.example.Pract9_Service();
      com.example.Pract9 port = service.getPract9Port();
       // TODO initialize WS operation arguments here
      java.lang.String courseName = request.getParameter("course");
      // TODO process result here
      double result = port.getFees(courseName);
```

```
      out.println("Result = "+result);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```