Assignment 21

WCF 1

Task 1 :

Create a WCF service will implement following methods

Define OperationContract named SayHello should take name as argument and return the wishes

(Good Morning or Good Afternoon or Good Evening) based on time

Define OperationContract

named TodayProgram should take name as argument and return the

Happy weekend or Enjoy Working day

Define two end point for TCP and HTTP

Use selfhosting to host the service

Consume the service in ASP.NET MVC

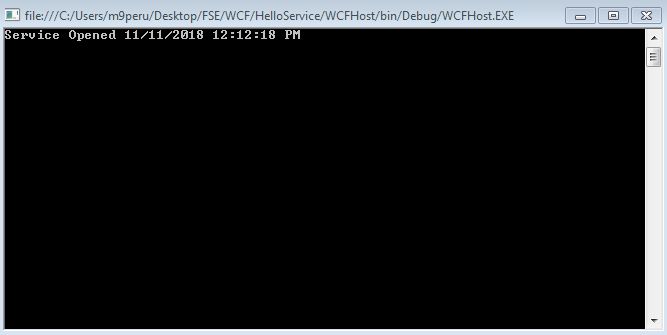
Design web form with two button

First button should invoke using http request

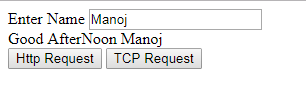
Second button should invoke using TCP request

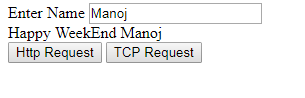
Output:

Host Screen



Client





Host Code Snippet

IServices.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 HelloService

{

// 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 SayHello(string Name);

[OperationContract]

string TodayProgram(string Name);

// TODO: Add your service operations here

}

}

Service1.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 HelloService

{

// 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

{

/// <summary>

///

/// </summary>

/// <param name="Name"></param>

/// <returns></returns>

public string SayHello(string Name)

{

string Message = null;

if (DateTime.Now.Hour < 12)

Message = "Good Morning";

else if (DateTime.Now.Hour < 16)

Message = "Good AfterNoon";

else if (DateTime.Now.Hour < 18)

Message = "Good Evening";

else

Message = "Good Night";

return Message + " " + Name;

}

/// <summary>

///

/// </summary>

/// <param name="Name"></param>

/// <returns></returns>

public string TodayProgram(string Name)

{

DayOfWeek dow=DateTime.Now.DayOfWeek;

string Message = null;

if (dow == DayOfWeek.Saturday || dow == DayOfWeek.Sunday)

{

Message = "Happy WeekEnd";

}

else

Message = "Enjoy Workingday";

return Message + " " + Name;

}

}

}

Host Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ServiceModel;

namespace WCFHost

{

class Program

{

static void Main(string[] args)

{

using (ServiceHost sv= new ServiceHost(typeof(HelloService.Service1)))

{

sv.Open();

Console.WriteLine("Service Opened "+DateTime.Now);

Console.ReadKey();

}

}

}

}

App.config

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

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />

</startup>

<system.serviceModel>

<services>

<service name="HelloService.Service1" behaviorConfiguration="mexBehaviour">

<endpoint address="HelloService" binding="basicHttpBinding" contract="HelloService.IService1">

</endpoint>

<endpoint address="GreetingService" binding="netHttpBinding" contract="HelloService.IService1">

</endpoint>

<endpoint address="mex1" binding="mexHttpBinding" contract="IMetadataExchange"></endpoint>

<endpoint address="mex2" binding="mexTcpBinding" contract="IMetadataExchange"></endpoint>

<host>

<baseAddresses>

<add baseAddress="http://localhost:8080/"/>

<add baseAddress="net.tcp://localhost:8090/"/>

</baseAddresses>

</host>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="mexBehaviour">

<!-- To avoid disclosing metadata information, set the values below to false before deployment -->

<serviceMetadata httpGetEnabled="true" />

<!-- To receive exception details in faults for debugging purposes, set the value below to true. Set to false before deployment to avoid disclosing exception information -->

<serviceDebug includeExceptionDetailInFaults="true"/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

</configuration>

WCF Service Task2, Job Openings

Create a WCF service will

implement following methods

Define OperationContract OpeningJobs return the list of openings

Define

OperationContract OpeningJobsByRole return the list of openings on specified role given in

the parameter

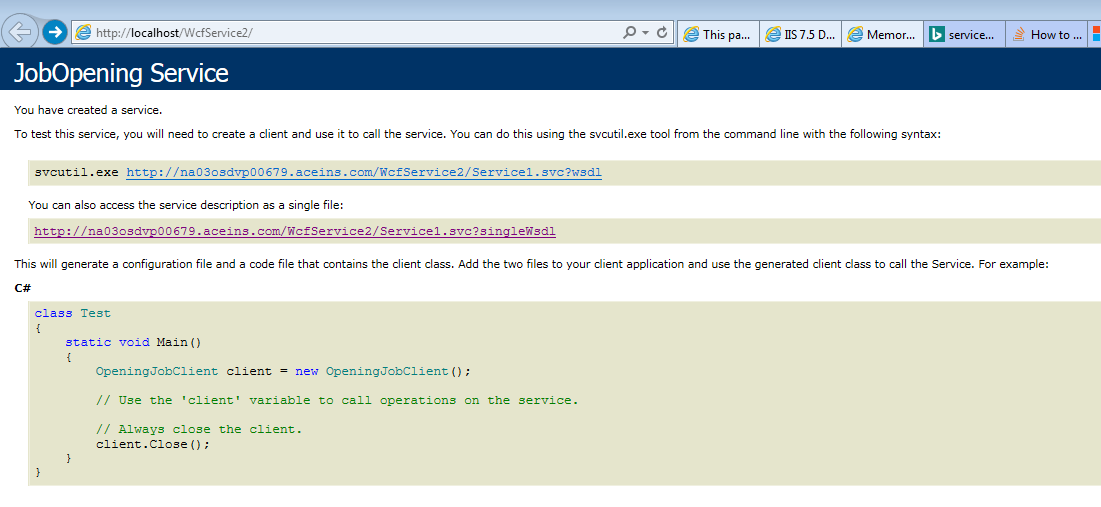
Host the Serevice in IIS

Consume the service in ASP.NET MVC

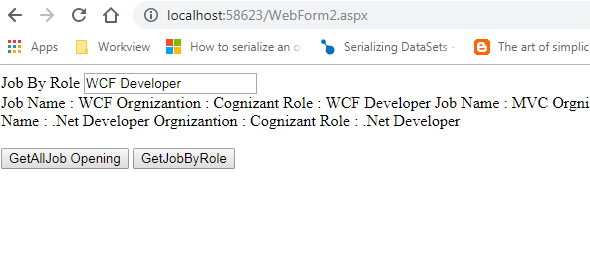
Design web form with two button

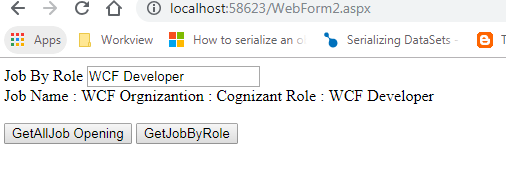
First button should invoke using http request

Output:



Client





IOpeningJob.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

namespace WcfService2

{

[ServiceContract]

public interface IOpeningJob

{

[OperationContract]

List<JobList> OpeningJobs();

[OperationContract]

List<JobList> OpeningJobsByRole(string Role);

}

}

JobList.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Web;

namespace WcfService2

{

[DataContract]

public class JobList

{

[DataMember]

public string JobName { get; set; }

[DataMember]

public string Organization { get; set; }

[DataMember]

public string Roles { get; set; }

public JobList()

{

}

public JobList(string JobName, string Org, string Roles)

{

this.JobName = JobName;

this.Organization = Org;

this.Roles = Roles;

}

}

}

Service1.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 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 JobOpening : IOpeningJob

{

/// <summary>

///

/// </summary>

/// <returns></returns>

public List<JobList> OpeningJobs()

{

List<JobList> Jobs = GetJobDetails();

return Jobs;

}

/// <summary>

///

/// </summary>

/// <param name="Role"></param>

/// <returns></returns>

public List<JobList> OpeningJobsByRole(string Role)

{

List<JobList> Jobs = GetJobDetails();

var query = from p in Jobs

where

p.Roles == Role

select p;

return query.ToList();

}

private List<JobList> GetJobDetails()

{

try

{

List<JobList> jobs = new List<JobList>();

jobs.Add(new JobList("WCF", "Cognizant", "WCF Developer"));

jobs.Add(new JobList("MVC", "Cognizant", "MVC Developer"));

jobs.Add(new JobList("ASP", "Cognizant", "ASP Developer"));

jobs.Add(new JobList(".Net Developer", "Cognizant", ".Net Developer"));

return jobs;

}

catch (Exception)

{

throw;

}

}

}

}

Web.config

<?xml version="1.0"?>

<configuration>

<appSettings/>

<connectionStrings/>

<system.web>

<compilation debug="true" targetFramework="4.0"/>

<!--

The <authentication> section enables configuration

of the security authentication mode used by

ASP.NET to identify an incoming user.

-->

<!--

The <customErrors> section enables configuration

of what to do if/when an unhandled error occurs

during the execution of a request. Specifically,

it enables developers to configure html error pages

to be displayed in place of a error stack trace.

<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">

<error statusCode="403" redirect="NoAccess.htm" />

<error statusCode="404" redirect="FileNotFound.htm" />

</customErrors>

-->

<pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID"/>

</system.web>

<!--

The system.webServer section is required for running ASP.NET AJAX under Internet

Information Services 7.0. It is not necessary for previous version of IIS.

-->

<system.webServer>

<!--

To browse web app root directory during debugging, set the value below to true.

Set to false before deployment to avoid disclosing web app folder information.

-->

<directoryBrowse enabled="true"/>

<defaultDocument>

<files>

<add value="Service1.svc"/>

</files>

</defaultDocument>

</system.webServer>

<system.serviceModel>

<services>

<service name="WcfService2.JobOpening" behaviorConfiguration="WcfService2.Service1Behavior">

<!-- Service Endpoints -->

<endpoint address="JobOpening" binding="basicHttpBinding" contract="WcfService2.IOpeningJob">

<!--

Upon deployment, the following identity element should be removed or replaced to reflect the

identity under which the deployed service runs. If removed, WCF will infer an appropriate identity

automatically.

-->

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>

<host>

<baseAddresses>

<add baseAddress="http://localhost:8095"/>

</baseAddresses>

</host>

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="WcfService2.Service1Behavior">

<!-- To avoid disclosing metadata information, set the value below to false before deployment -->

<serviceMetadata httpGetEnabled="true"/>

<!-- To receive exception details in faults for debugging purposes, set the value below to true. Set to false before deployment to avoid disclosing exception information -->

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

</serviceBehaviors>

</behaviors>

<serviceHostingEnvironment minFreeMemoryPercentageToActivateService="0"/>

</system.serviceModel>

</configuration>

Client Design page

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="WebApplication1.WebForm2" %>

<!DOCTYPE html>

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

<head runat="server">

<title></title>

</head>

<body>

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

<div>

<asp:Label ID="Label1" runat="server" Text="Job By Role"></asp:Label>

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<div id="Alldata" runat="server">

</div>

<br />

<asp:Button ID="Button1" runat="server" Text="GetAllJob Opening" OnClick="Button1\_Click" />

<asp:Button ID="Button2" runat="server" Text="GetJobByRole" OnClick="Button2\_Click" />

</div>

</form>

</body>

</html>

Client Cs page

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApplication1

{

public partial class WebForm2 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

ServiceReference2.OpeningJobClient OJ = new ServiceReference2.OpeningJobClient();

List<ServiceReference2.JobList> list = OJ.OpeningJobs().ToList();

StringBuilder sb = new StringBuilder();

foreach (ServiceReference2.JobList jl in list)

{

sb.Append("Job Name : ").Append(jl.JobName).Append(" Orgnizantion : ").Append(jl.Organization).Append(" Role : ").Append(jl.Roles).Append("\n");

}

Alldata.InnerHtml = sb.ToString();

}

protected void Button2\_Click(object sender, EventArgs e)

{

ServiceReference2.OpeningJobClient OJ = new ServiceReference2.OpeningJobClient();

List<ServiceReference2.JobList> list = OJ.OpeningJobsByRole(TextBox1.Text).ToList();

StringBuilder sb = new StringBuilder();

foreach (ServiceReference2.JobList jl in list)

{

sb.Append("Job Name : ").Append(jl.JobName).Append(" Orgnizantion : ").Append(jl.Organization).Append(" Role : ").Append(jl.Roles).Append("\n");

}

Alldata.InnerHtml = sb.ToString();

}

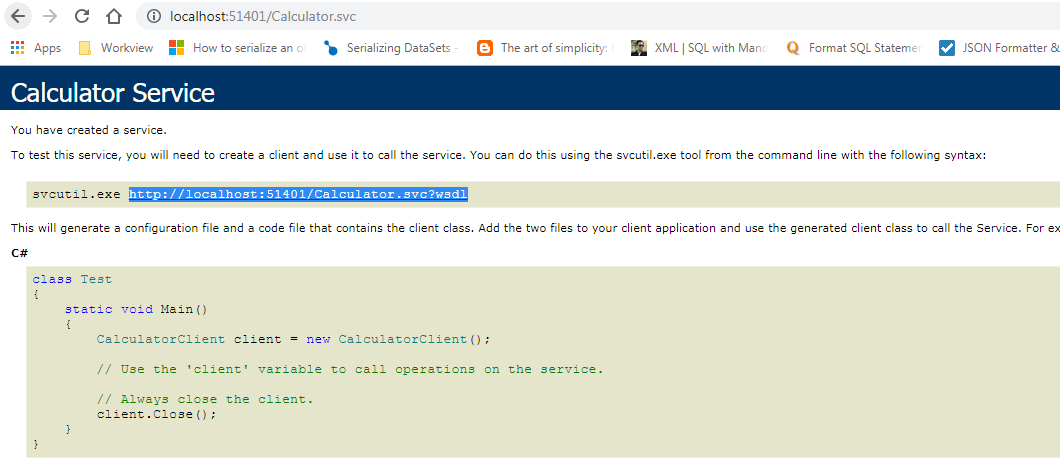
}

}

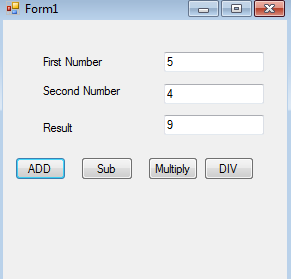
2. Create a WCF service to depict simple calculator with addition, subtraction features and deploy

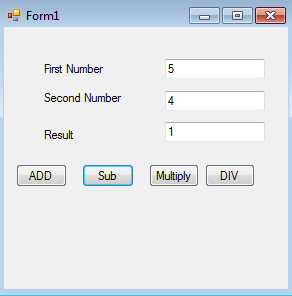
and consume the service

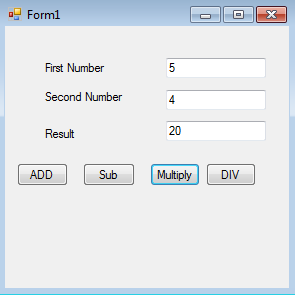
WCF Output

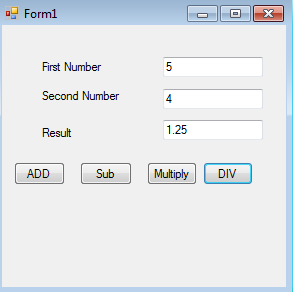


Client Output









ICalculator.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 MyCalculator

{

// 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 ICalculator

{

[OperationContract]

double Add(double n1, double n2);

[OperationContract]

double Subtract(double n1, double n2);

[OperationContract]

double Multiply(double n1, double n2);

[OperationContract]

double Divide(double n1, double n2);

// TODO: Add your service operations here

}

}

Calculator.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 MyCalculator

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Calculator" in code, svc and config file together.

// NOTE: In order to launch WCF Test Client for testing this service, please select Calculator.svc or Calculator.svc.cs at the Solution Explorer and start debugging.

public class Calculator : ICalculator

{

public double Add(double n1, double n2)

{

return n1 + n2;

}

public double Subtract(double n1, double n2)

{

return n1 - n2;

}

public double Multiply(double n1, double n2)

{

return n1 \* n2;

}

public double Divide(double n1, double n2)

{

return n1 / n2;

}

}

}

App.config

<?xml version="1.0"?>

<configuration>

<configSections>

<sectionGroup name="system.web.extensions" type="System.Web.Configuration.SystemWebExtensionsSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">

<sectionGroup name="scripting" type="System.Web.Configuration.ScriptingSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">

<section name="scriptResourceHandler" type="System.Web.Configuration.ScriptingScriptResourceHandlerSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication"/>

<sectionGroup name="webServices" type="System.Web.Configuration.ScriptingWebServicesSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">

<section name="jsonSerialization" type="System.Web.Configuration.ScriptingJsonSerializationSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="Everywhere" />

<section name="profileService" type="System.Web.Configuration.ScriptingProfileServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication" />

<section name="authenticationService" type="System.Web.Configuration.ScriptingAuthenticationServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication" />

<section name="roleService" type="System.Web.Configuration.ScriptingRoleServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" requirePermission="false" allowDefinition="MachineToApplication" />

</sectionGroup>

</sectionGroup>

</sectionGroup>

</configSections>

<appSettings/>

<connectionStrings/>

<system.web>

<compilation debug="true">

<assemblies>

<add assembly="System.Core, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>

<add assembly="System.Data.DataSetExtensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>

<add assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

<add assembly="System.Xml.Linq, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>

</assemblies>

</compilation>

<!--

The <authentication> section enables configuration

of the security authentication mode used by

ASP.NET to identify an incoming user.

-->

<authentication mode="Windows" />

<!--

The <customErrors> section enables configuration

of what to do if/when an unhandled error occurs

during the execution of a request. Specifically,

it enables developers to configure html error pages

to be displayed in place of a error stack trace.

<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">

<error statusCode="403" redirect="NoAccess.htm" />

<error statusCode="404" redirect="FileNotFound.htm" />

</customErrors>

-->

<pages>

<controls>

<add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

<add tagPrefix="asp" namespace="System.Web.UI.WebControls" assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

</controls>

</pages>

<httpHandlers>

<remove verb="\*" path="\*.asmx"/>

<add verb="\*" path="\*.asmx" validate="false" type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

<add verb="\*" path="\*\_AppService.axd" validate="false" type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

<add verb="GET,HEAD" path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" validate="false"/>

</httpHandlers>

<httpModules>

<add name="ScriptModule" type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

</httpModules>

</system.web>

<system.codedom>

<compilers>

<compiler language="c#;cs;csharp" extension=".cs" warningLevel="4"

type="Microsoft.CSharp.CSharpCodeProvider, System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">

<providerOption name="CompilerVersion" value="v3.5"/>

<providerOption name="WarnAsError" value="false"/>

</compiler>

</compilers>

</system.codedom>

<!--

The system.webServer section is required for running ASP.NET AJAX under Internet

Information Services 7.0. It is not necessary for previous version of IIS.

-->

<system.webServer>

<validation validateIntegratedModeConfiguration="false"/>

<modules>

<remove name="ScriptModule" />

<add name="ScriptModule" preCondition="managedHandler" type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

</modules>

<handlers>

<remove name="WebServiceHandlerFactory-Integrated"/>

<remove name="ScriptHandlerFactory" />

<remove name="ScriptHandlerFactoryAppServices" />

<remove name="ScriptResource" />

<add name="ScriptHandlerFactory" verb="\*" path="\*.asmx" preCondition="integratedMode"

type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

<add name="ScriptHandlerFactoryAppServices" verb="\*" path="\*\_AppService.axd" preCondition="integratedMode"

type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>

<add name="ScriptResource" preCondition="integratedMode" verb="GET,HEAD" path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />

</handlers>

<!--

To browse web app root directory during debugging, set the value below to true.

Set to false before deployment to avoid disclosing web app folder information.

-->

<directoryBrowse enabled="true"/>

</system.webServer>

<runtime>

<assemblyBinding appliesTo="v2.0.05727" xmlns="urn:schemas-microsoft-com:asm.v1">

<dependentAssembly>

<assemblyIdentity name="System.Web.Extensions" publicKeyToken="31bf3856ad364e35"/>

<bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="3.5.0.0"/>

</dependentAssembly>

<dependentAssembly>

<assemblyIdentity name="System.Web.Extensions.Design" publicKeyToken="31bf3856ad364e35"/>

<bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="3.5.0.0"/>

</dependentAssembly>

</assemblyBinding>

</runtime>

<system.serviceModel>

<services>

<service behaviorConfiguration="MyCalculator.Service1Behavior"

name="MyCalculator.Calculator">

<endpoint address="" binding="basicHttpBinding" contract="MyCalculator.ICalculator">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior name="MyCalculator.Service1Behavior">

<!-- To avoid disclosing metadata information, set the value below to false before deployment -->

<serviceMetadata httpGetEnabled="true"/>

<!-- To receive exception details in faults for debugging purposes, set the value below to true. Set to false before deployment to avoid disclosing exception information -->

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

</configuration>

Client cs file

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace MyCalculatorClient

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

CalculatorClient.CalculatorClient cc = new CalculatorClient.CalculatorClient();

textBox3.Text = cc.Add(Double.Parse(textBox1.Text), Double.Parse(textBox2.Text)).ToString();

}

private void button2\_Click(object sender, EventArgs e)

{

CalculatorClient.CalculatorClient cc = new CalculatorClient.CalculatorClient();

textBox3.Text = cc.Subtract(Double.Parse(textBox1.Text), Double.Parse(textBox2.Text)).ToString();

}

private void button3\_Click(object sender, EventArgs e)

{

CalculatorClient.CalculatorClient cc = new CalculatorClient.CalculatorClient();

textBox3.Text = cc.Multiply(Double.Parse(textBox1.Text), Double.Parse(textBox2.Text)).ToString();

}

private void button4\_Click(object sender, EventArgs e)

{

CalculatorClient.CalculatorClient cc = new CalculatorClient.CalculatorClient();

textBox3.Text = cc.Divide(Double.Parse(textBox1.Text), Double.Parse(textBox2.Text)).ToString();

}

}

}