5.2 Consuming Web Services Using WCF



This section will guide you to:

* Create a WCF Library project to define a Service Contract
* Create a Windows Console project to implement the Service Contract
* Create a Windows Console project to act as Client for the WCF Service

**Development Environment:**

* Visual Studio 2019 Community Version
* Windows 10

This guide has twelve subsections, namely:

5.2.1 Creating a WCF Library Project to define a Service Contract

5.2.2 Editing IService1.cs

5.2.3 Editing Service1.cs

5.2.4 Editing App.Config

5.2.5 Creating a Windows Console project to implement the Service Contract

5.2.6 Adding code to Program.cs

5.2.7 Creating a Windows Console project to act as client for the WCF Service

5.2.8 Updating App.config

5.2.9 Adding code to Program.cs

5.2.10 Building the solution

5.2.11 Publishing and running the project

5.2.12 Pushing the code to your GitHub repositories

**Step 5.2.1:** Creating a WCF Library project to define and implement a Service Contract

* Open Visual Studio.
* From the top menu, select **File->New->Project.**
* In **Create A New Project** screen, select **WCF Service Library** from the list of available project types and click on **Next.**
* Enter **Project Name** as Phase3Section5.5 and click on **Create.**
* This will create the files for a WCF project.

**Step 5.2.2:** Editing IService1.cs

* In the **Solution Explorer,** double click IService1.cs and add the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Runtime.Serialization;

**using** System.ServiceModel;

**using** System.Text;

**namespace** Phase3Section5.\_5

{

[ServiceContract(Namespace = "http://Microsoft.ServiceModel.Samples")]

**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);

}

}

**Step 5.2.3:** Editing Service1.cs

* In the **Solution Explore,** double click **Service1.cs.**
* Enter the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Runtime.Serialization;

**using** System.ServiceModel;

**using** System.Text;

**namespace** Phase3Section5.\_5

{

**public** **class** CalculatorService : ICalculator

{

**public** **double** Add(**double** n1, **double** n2)

{

**double** result = n1 + n2;

Console.WriteLine("Received Add({0},{1})", n1, n2);

// Code added to write output to the console window.

Console.WriteLine("Return: {0}", result);

**return** result;

}

**public** **double** Subtract(**double** n1, **double** n2)

{

**double** result = n1 - n2;

Console.WriteLine("Received Subtract({0},{1})", n1, n2);

Console.WriteLine("Return: {0}", result);

**return** result;

}

**public** **double** Multiply(**double** n1, **double** n2)

{

**double** result = n1 \* n2;

Console.WriteLine("Received Multiply({0},{1})", n1, n2);

Console.WriteLine("Return: {0}", result);

**return** result;

}

**public** **double** Divide(**double** n1, **double** n2)

{

**double** result = n1 / n2;

Console.WriteLine("Received Divide({0},{1})", n1, n2);

Console.WriteLine("Return: {0}", result);

**return** result;

}

}

}

**Step 5.2.4:** Editing App.config

* In the **Solution Explore,** double click **App.config.**
* Enter the following code:

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

<configuration>

<appSettings>

<add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />

</appSettings>

<system.web>

<compilation debug="true" />

</system.web>

<!-- When deploying the service library project, the content of the config file must be added to the host's

  app.config file. System.Configuration does not support config files for libraries. -->

<system.serviceModel>

<services>

<service name="Phase3Section5.\_5.CalculatorService">

<host>

<baseAddresses>

<add baseAddress = "http://localhost:8733/Phase3Section5.\_5/CalculatorService/" />

</baseAddresses>

</host>

<!-- Service Endpoints -->

<!-- Unless fully qualified, address is relative to base address supplied above -->

<endpoint address="" binding="wsHttpBinding" contract="Phase3Section5.\_5.ICalculator">

<!--

              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.

          -->

<identity>

<dns value="localhost"/>

</identity>

</endpoint>

<!-- Metadata Endpoints -->

<!-- The Metadata Exchange endpoint is used by the service to describe itself to clients. -->

<!-- This endpoint does not use a secure binding and should be secured or removed before deployment -->

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

</service>

</services>

<behaviors>

<serviceBehaviors>

<behavior>

<!-- To avoid disclosing metadata information,

          set the values below to false before deployment -->

<serviceMetadata httpGetEnabled="True" httpsGetEnabled="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>

**Step 5.2.5:** Creating a Windows Console project to implement the Service Contract

* In **Solution Explorer,** right click the Solution and choose **Add->New Project.**
* In **Create A New Project** screen, select **Console App (.NET Framework)** from the list of available project types and click on **Next.**
* Enter **Project Name** as Phase3Section5.5Host and click on **Create.**
* This will create the files for a Windows Console Project.
* In **Solution Explorer,** expand **Phase3Section5.5Host->References,** right click **References,** and choose **Add Reference.**
* Click **Assembly** tab and from the list of assemblies, check **System.ServiceModel,** and click **Ok.**
* In **Solution Explorer,** expand **Phase3Section5.5Host->References,** right click **References,** and choose **Add Reference.**
* Click the Projectstab. From the list of projects, check **Phase3Section5.5** andclick **Ok.**

**Step 5.2.6:** Adding code to Program.cs

* In **Solution Explorer,** double click **Program.cs** and add the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**using** System.ServiceModel;

**using** System.ServiceModel.Description;

**using** Phase3Section5.\_5;

**namespace** Phase3Section5.\_5Host

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

// Step 1: Create a URI to serve as the base address.

Uri baseAddress = **new** Uri("http://localhost:8733/Phase3Section5.\_5/");

// Step 2: Create a ServiceHost instance.

ServiceHost selfHost = **new** ServiceHost(**typeof**(CalculatorService), baseAddress);

**try**

{

// Step 3: Add a service endpoint.

selfHost.AddServiceEndpoint(**typeof**(ICalculator), **new** WSHttpBinding(), "CalculatorService");

// Step 4: Enable metadata exchange.

ServiceMetadataBehavior smb = **new** ServiceMetadataBehavior();

smb.HttpGetEnabled = **true**;

selfHost.Description.Behaviors.Add(smb);

// Step 5: Start the service.

selfHost.Open();

Console.WriteLine("The service is ready.");

// Close the ServiceHost to stop the service.

Console.WriteLine("Press <Enter> to terminate the service.");

Console.WriteLine();

Console.ReadLine();

selfHost.Close();

}

**catch** (CommunicationException ce)

{

Console.WriteLine("An exception occurred: {0}", ce.Message);

selfHost.Abort();

}

}

}

}

**Step 5.2.7:** Creating a Windows Console project to act as Client for the WCF Service

* In **Solution Explorer,** right click the Solution and choose **Add->New Project.**
* In **Create A New Project** screen, select **Console App (.NET Framework)** from the list of available project types and click on **Next.**
* Enter **Project Name** as Phase3Section5.5Client and click on **Create.**
* This will create the files for a Windows Console project.
* In **Solution Explorer,** expand **Phase3Section5.5Client->References,** right click **References,** and choose **Add Reference.**
* Click **Assembly** tab and from the list of assemblies, check **System.ServiceModel,** and click **Ok.**
* In **Solution Explorer,** expand **Phase3Section5.5Client->References,** right click **References** and choose **Add Service Reference.**
* Click **Discover**. It will find the Calculator Service. Click **Ok.**

**Step 5.2.8:** Updating App.config

* In **Solution Explorer,** double App.config and make sure the script is as below:

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

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

<system.serviceModel>

<bindings>

<wsHttpBinding>

<binding name="WSHttpBinding\_ICalculator" />

</wsHttpBinding>

</bindings>

<client>

<endpoint address="http://localhost:8733/Phase3Section5.\_5/CalculatorService/"

binding="wsHttpBinding" bindingConfiguration="WSHttpBinding\_ICalculator"

contract="ServiceReference1.ICalculator" name="WSHttpBinding\_ICalculator">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

</client>

</system.serviceModel>

</configuration>

**Step 5.2.9:** Adding code to Program.cs

* In **Solution Explorer,**  double click **Program.cs** and add the following code:

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**using** Phase3Section5.\_5Client.ServiceReference1;

**namespace** Phase3Section5.\_5Client

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

//Step 1: Create an instance of the WCF proxy.

CalculatorClient client = **new** CalculatorClient();

// Step 2: Call the service operations.

// Call the Add service operation.

**double** value1 = 100.00D;

**double** value2 = 15.99D;

**double** result = client.Add(value1, value2);

Console.WriteLine("Add({0},{1}) = {2}", value1, value2, result);

// Call the Subtract service operation.

value1 = 145.00D;

value2 = 76.54D;

result = client.Subtract(value1, value2);

Console.WriteLine("Subtract({0},{1}) = {2}", value1, value2, result);

// Call the Multiply service operation.

value1 = 9.00D;

value2 = 81.25D;

result = client.Multiply(value1, value2);

Console.WriteLine("Multiply({0},{1}) = {2}", value1, value2, result);

// Call the Divide service operation.

value1 = 22.00D;

value2 = 7.00D;

result = client.Divide(value1, value2);

Console.WriteLine("Divide({0},{1}) = {2}", value1, value2, result);

// Step 3: Close the client to gracefully close the connection and clean up resources.

Console.WriteLine("\nPress <Enter> to terminate the client.");

Console.ReadLine();

client.Close();

}

}

}

**Step 5.2.10:** Building the solution

* From the top menu, choose **Build->Build Solution.**
* If any compile errors are shown, fix them as required.

**Step 5.2.11:** Publishing and running the project

* Make sure Visual Studio is running in Administrator Privileges. If not, start it from the Windows Start menu by right clicking on **Visual Studio** and choosing **Run As Administrator.**
* In **Solution Explorer,** right click **Phase3Section5.5Client** and choose **Set as Startup Project.**
* From the topmenu, choose **Debug->Start Without Debugging.**
* This will execute the WCF Service .

**Step 5.2.12:**Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master