WCF – Part 1

Agenda

- Creating a simple WCF Service in VS 2013
- Testing it with "WcfTestClient.exe" tool
- Consuming it in a test console application.

Tools

- Visual Studio 2013

Version	Last updated	Comments	Modified By
V1.0	18-05-2016		Bilal Shahzad

Brief Introduction

WCF (Windows Communication Foundation) is a framework for building service-oriented architecture. One main advantage of WCF is “Interoperability”. Client (in any technology) can consume it.

[**ServiceContract**] attribute is used (on a class or interface) to make it a service.

[**OperationContract**] attribute is used on methods in service class to make them part of service. Methods without this attribute will not be exposed to client.

[**DataContract**] We can use this attribute (optionally) on our complex types to make them serializable. If we use this attribute then we’ll have to use [**DataMember**] attribute on fields which should be serialized.

End Point:

It consists of three things a) Address b) Binding c) Contract

Address: It specifies the exact location to receive the messages and it is specified as URI.

Binding: It defines the way an endpoint communicates. It comprises of some binding elements that make the infrastructure for communication. For example, a binding states the protocols used for transport like TCP, HTTP, etc.

Contract: It is the full name of our service (which has [**ServiceContract**] attribute). It basically tells the list of operations which are available by the service.

WCF vs. Web Service

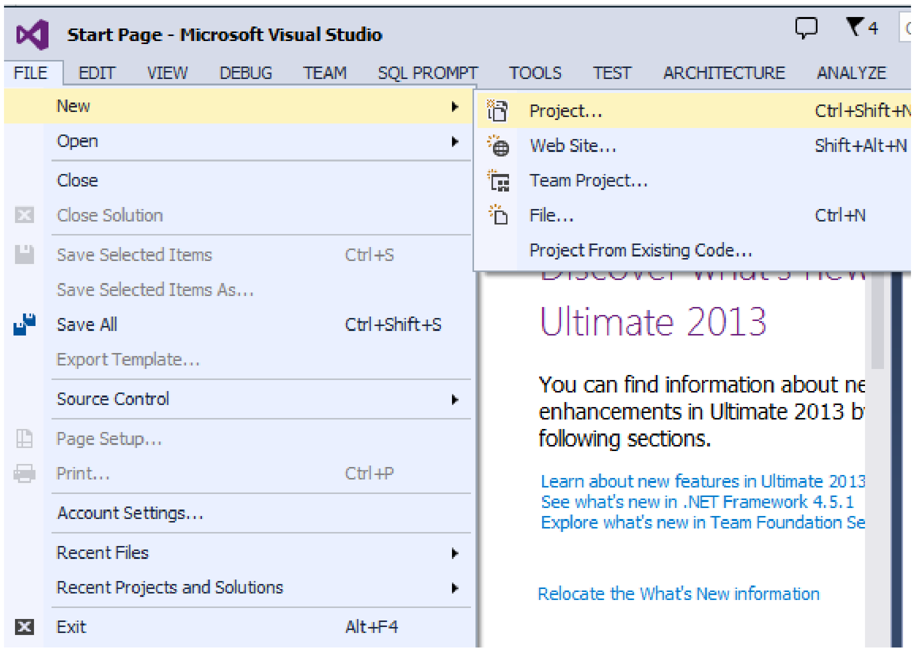
http://www.tutorialspoint.com/wcf/wcf_versus_web_service.htm

WCF Architecture

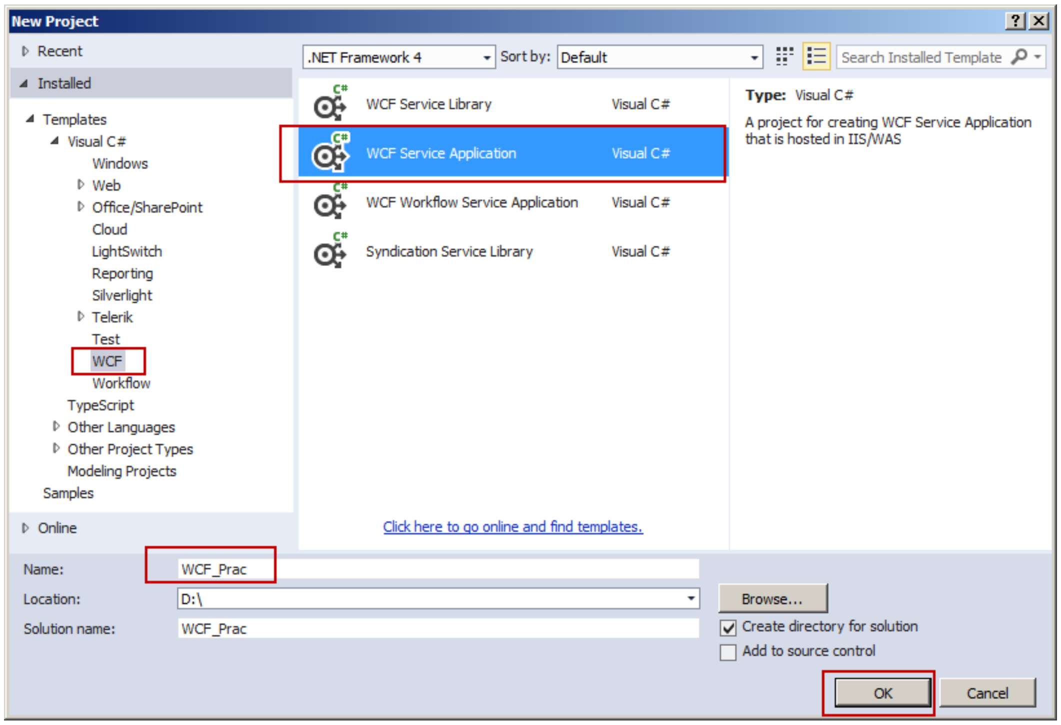
http://www.tutorialspoint.com/wcf/wcf_architecture.htm

Step by Step Walkthrough

1- Create a new project (File -> New -> Project)


Start Page - Microsoft Visual Studio
FILE EDIT VIEW DEBUG TEAM SQL PROMPT TOOLS TEST ARCHITECTURE ANALYZE
New
Open
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Source Control
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project... Ctrl+Shift+N
Web Site... Shift+Alt+N
Team Project...
File... Ctrl+N
Project From Existing Code...
Ultimate 2013
You can find information about ne enhancements in Ultimate 2013 b following sections.
Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Se
Relocate the What's New information

2- Choose “WCF” from left panel and then “WCF Service Application”. Name it and then press OK.



Type: Visual C#

A project for creating WCF Service Application that is hosted in IIS/WAS

Name: WCF_Prac

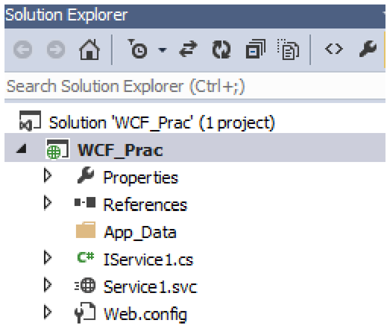
Location: D:\

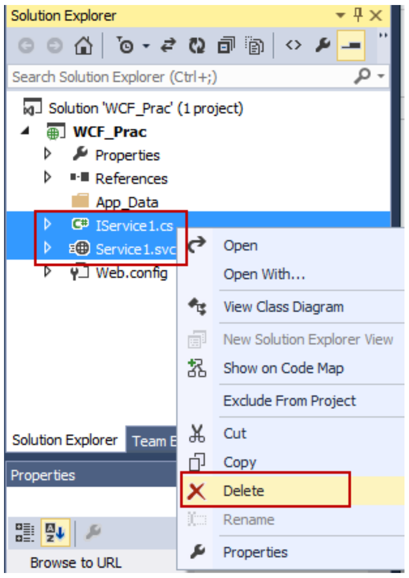
Solution name: WCF_Prac

☑ Create directory for solution

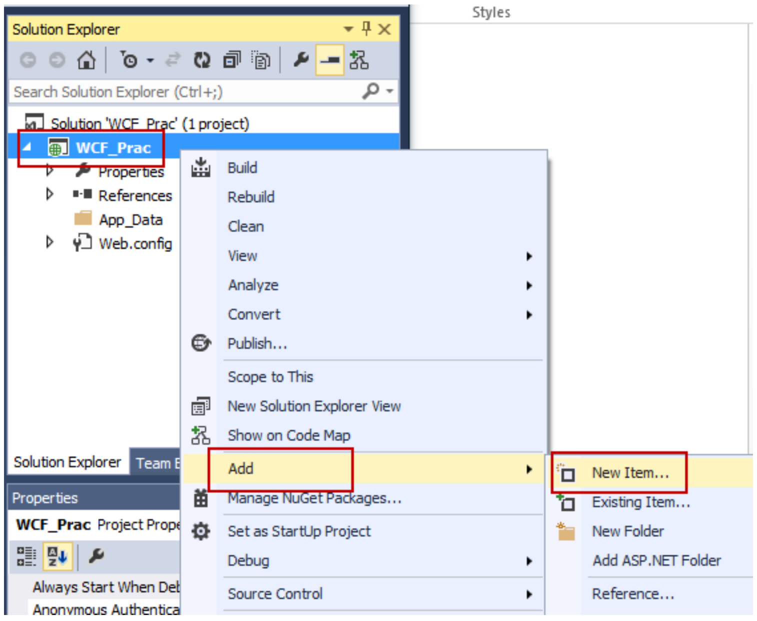
☐ Add to source control

3- You will see that “IService1.cs” & “Service1.svc” files are created by default. Select both files, right click and choose “Delete”.

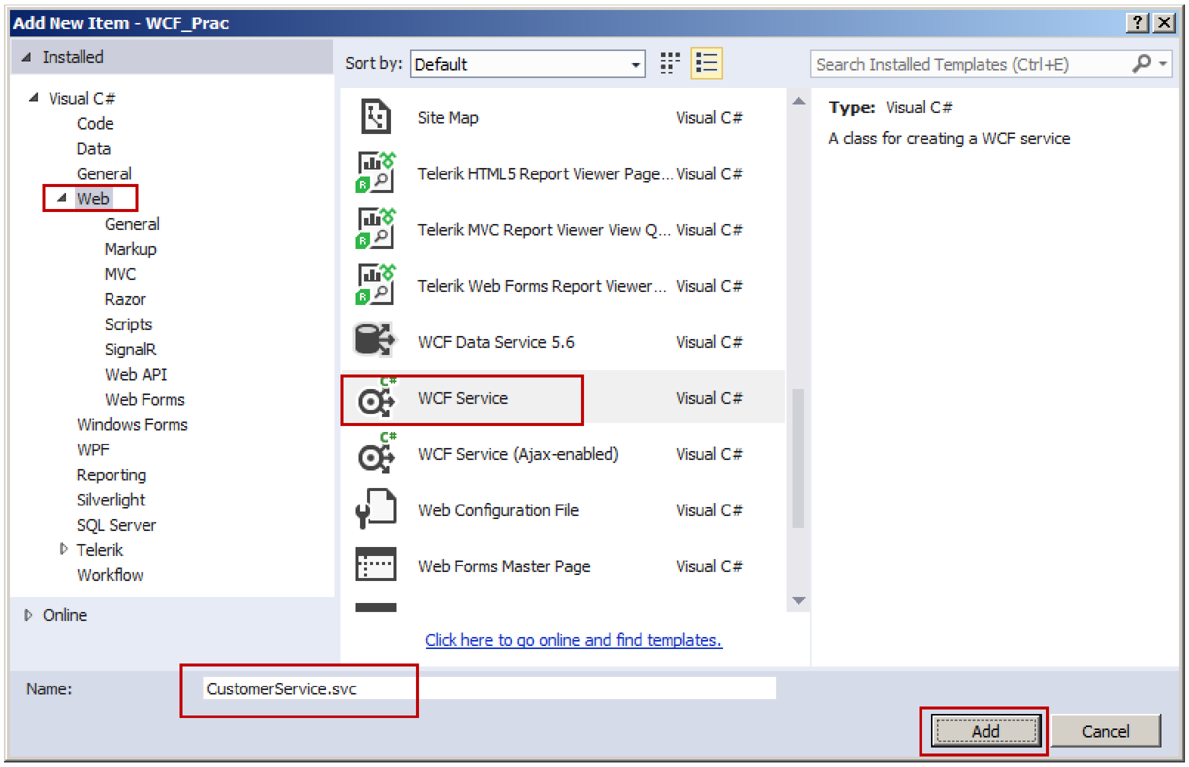




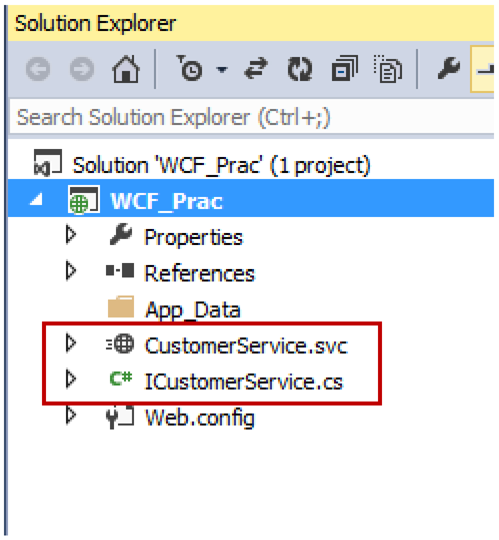
4- Add a new service. To do so, right click on your project, choose Add and then “New Item”.



5- Choose “Web” in left panel and “WCF Service” in right panel. Give “CustomerService.svc” in Name box and click “Add”.

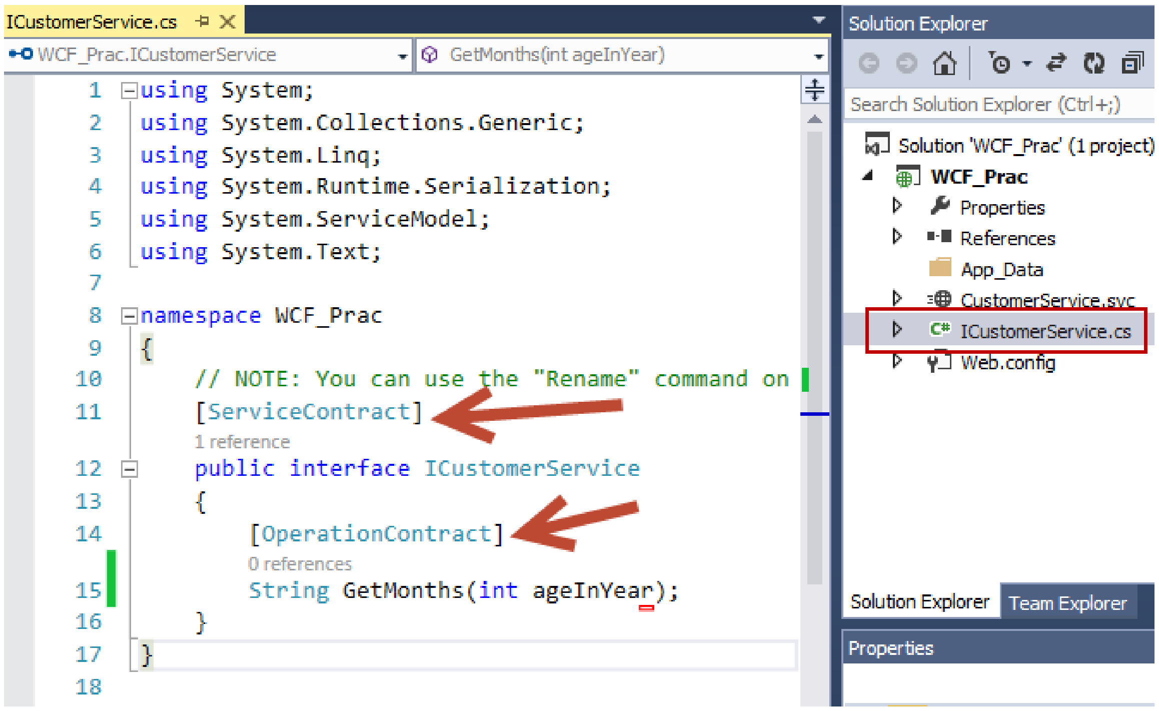

Add New Item - WCF_Prac
Installed
Visual C#
Code
Data
General
Web
General
Markup
MVC
Razor
Scripts
SignalR
Web API
Web Forms
Windows Forms
WPF
Reporting
Silverlight
SQL Server
Telerik
Workflow
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Site Map Visual C#
Telerik HTML5 Report Viewer Page... Visual C#
Telerik MVC Report Viewer View Q... Visual C#
Telerik Web Forms Report Viewer... Visual C#
WCF Data Service 5.6 Visual C#
WCF Service Visual C#
WCF Service (Ajax-enabled) Visual C#
Web Configuration File Visual C#
Web Forms Master Page Visual C#
Click here to go online and find templates.
Type: Visual C#
A class for creating a WCF service
Name: CustomerService.svc
Add
Cancel

6- You will see that two files (one interface and one svc) are created.



7- Open “ICustomerService.cs” file, change the interface code and make it like following code. Screenshot below is also showing this change. Note that we’ve decorated our interface with [**ServiceContract**] attribute. This attribute will make our interface a WCF service. We need to decorate our operations (which we want to expose to outer world) with [**OperationContract**].

```
[ServiceContract]
public interface ICustomerService
{
        [OperationContract]
         String GetMonths(int ageInYear);
}
```

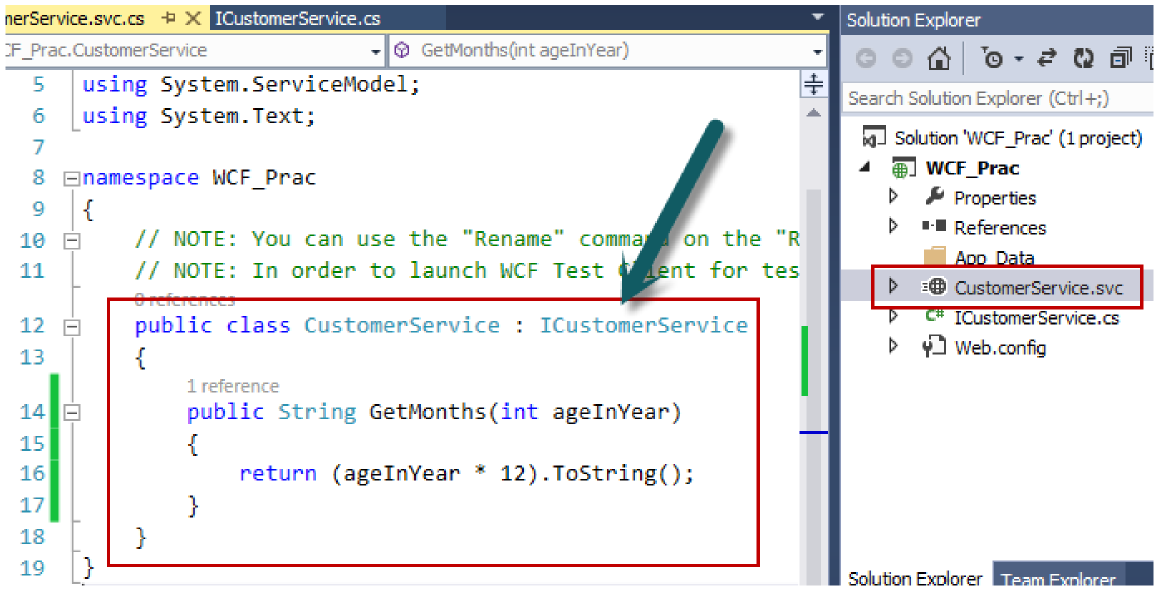


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace WCF_Prac
{
    // NOTE: You can use the "Rename" command on
    [ServiceContract]
    1 reference
    public interface ICustomerService
    {
        [OperationContract]
        0 references
        String GetMonths(int ageInYear);
    }
}
```

8- Next open “CustomerService.svc” file and make changes as shown in screenshot below. This class is implementing our service interface and also providing definition to the functions of interface. **Note:** It is not mandatory to create an interface & apply [**ServiceContract**] on that. You can create class directly and decorate with [**ServiceContract**] to make class as our service class.

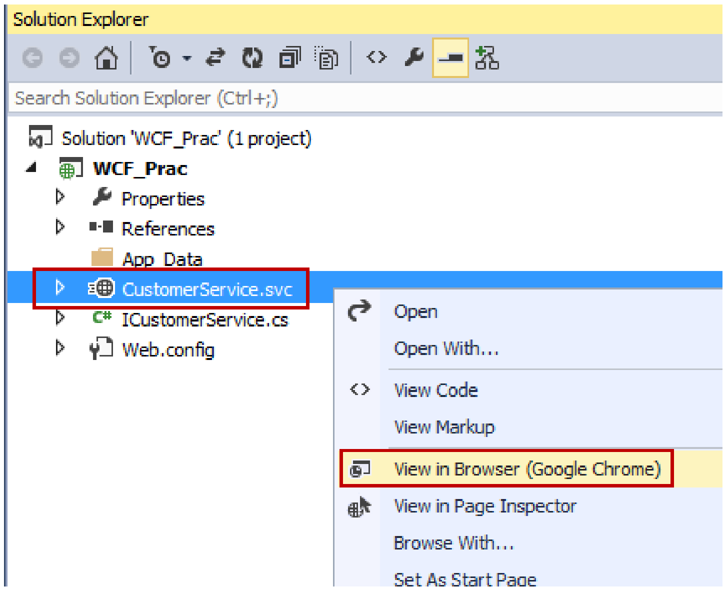
```
public class CustomerService : ICustomerService
    {
        public String GetMonths(int ageInYear)
        {
            return (ageInYear * 12).ToString();
        }
    }
```

Here is the screenshot showing the code of “CustomerService” class.

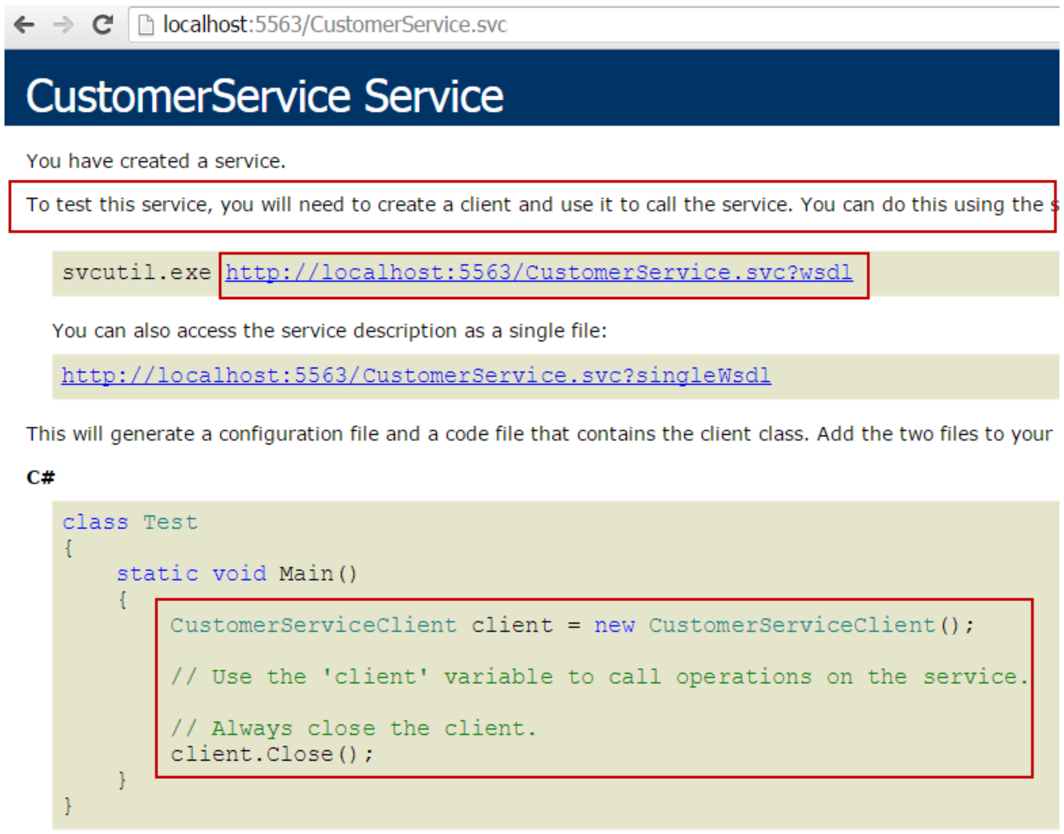


```
using System.ServiceModel;
using System.Text;

namespace WCF_Prac
{
    // NOTE: You can use the "Rename" command on the "R
    // NOTE: In order to launch WCF Test Client for tes
    public class CustomerService : ICustomerService
    {
        public String GetMonths(int ageInYear)
        {
            return (ageInYear * 12).ToString();
        }
    }
}
```

9- For understanding purposes, we can consider a service like class library. It doesn’t have any front end (or starting point). We can add its reference (service reference) and can call its functions (like we call functions of a class library). But if you right click on it and select “View in Browser”, you will see some information about the service.



10- Here we can see that our service is running on http://localhost:5563/CustomerService.svc (Root path can be different for your project). We can also see that link for WSDL is available. You can click on it to see how it looks like. Also a sample C# code is given about how to consume this service. We'll see all this later. For now, this screen is just for your information (nothing to do in this step).



localhost:5563/CustomerService.svc

CustomerService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the s

```
svcutil.exe http://localhost:5563/CustomerService.svc?wsdl
```

You can also access the service description as a single file:

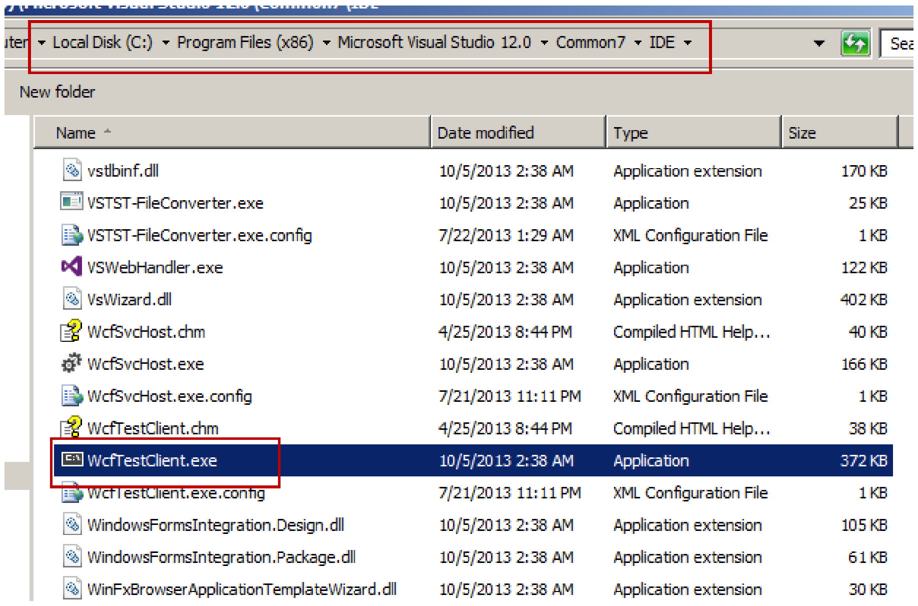
```
http://localhost:5563/CustomerService.svc?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your

C#

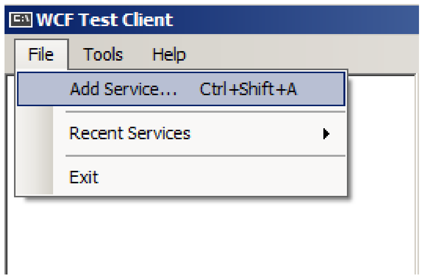
```
class Test
{
    static void Main()
    {
        CustomerServiceClient client = new CustomerServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

11- Now you can't test the functions of service in browser (by using above approach). So you can add reference of this service in other application and then call the methods to check if methods are working fine. There is a tool "WcfTestClient.exe" client which comes with Visual Studio. We can also use this to test our service. You can find this tool in "C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE" (this is path for VS 2013).

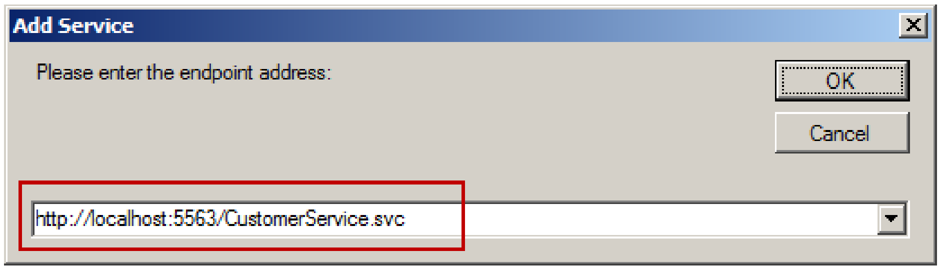

Local Disk (C:) ▸ Program Files (x86) ▸ Microsoft Visual Studio 12.0 ▸ Common7 ▸ IDE
New folder

Name	Date modified	Type	Size
vstlbinf.dll	10/5/2013 2:38 AM	Application extension	170 KB
VSTST-FileConverter.exe	10/5/2013 2:38 AM	Application	25 KB
VSTST-FileConverter.exe.config	7/22/2013 1:29 AM	XML Configuration File	1 KB
VSWebHandler.exe	10/5/2013 2:38 AM	Application	122 KB
VsWizard.dll	10/5/2013 2:38 AM	Application extension	402 KB
WcfSvcHost.chm	4/25/2013 8:44 PM	Compiled HTML Help...	40 KB
WcfSvcHost.exe	10/5/2013 2:38 AM	Application	166 KB
WcfSvcHost.exe.config	7/21/2013 11:11 PM	XML Configuration File	1 KB
WcfTestClient.chm	4/25/2013 8:44 PM	Compiled HTML Help...	38 KB
WcfTestClient.exe	10/5/2013 2:38 AM	Application	372 KB
WcfTestClient.exe.config	7/21/2013 11:11 PM	XML Configuration File	1 KB
WindowsFormsIntegration.Design.dll	10/5/2013 2:38 AM	Application extension	105 KB
WindowsFormsIntegration.Package.dll	10/5/2013 2:38 AM	Application extension	61 KB
WinFxBrowserApplicationTemplateWizard.dll	10/5/2013 2:38 AM	Application extension	30 KB

12- Open the tool. Then add a service by “Fil-> Add Service” option. We are assuming that our WCF service is still running in local server. **Note:** You can open service URL in browser to verify if it is running or not.



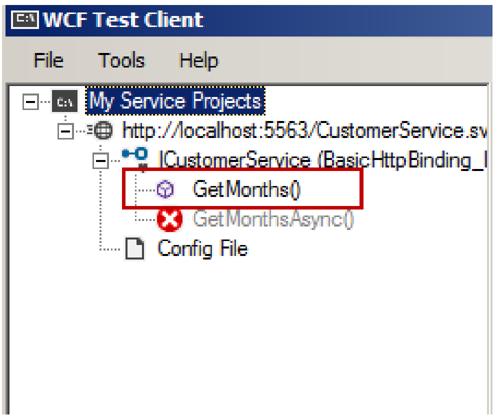
13- Provide URL of your service and click OK.



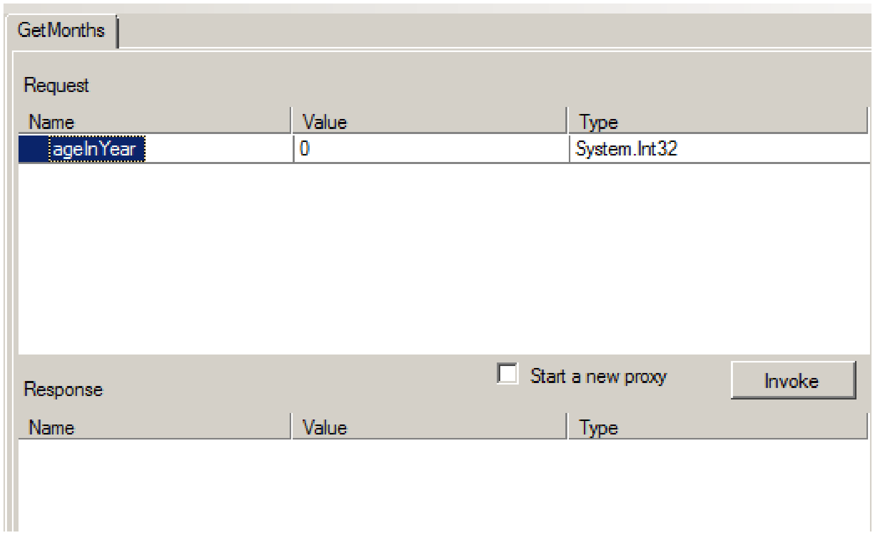
Please enter the endpoint address:

http://localhost:5563/CustomerService.svc

14- It will add service detail (& methods) in left penal. Click on “GetMonths” to check what happens in right panel.



15- In right panel, you will see fields for all the input parameters of selected method.



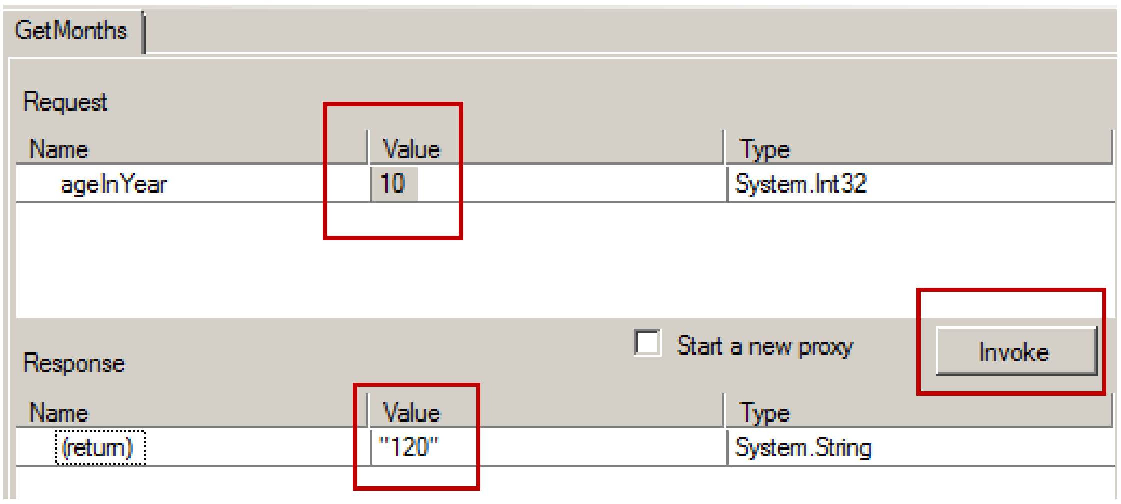
Request

Name	Value	Type
ageInYear	0	System.Int32

Response

Name	Value	Type

16- Provide some value in “Value” field and click “Invoke” button. You will see output in “Response” section.



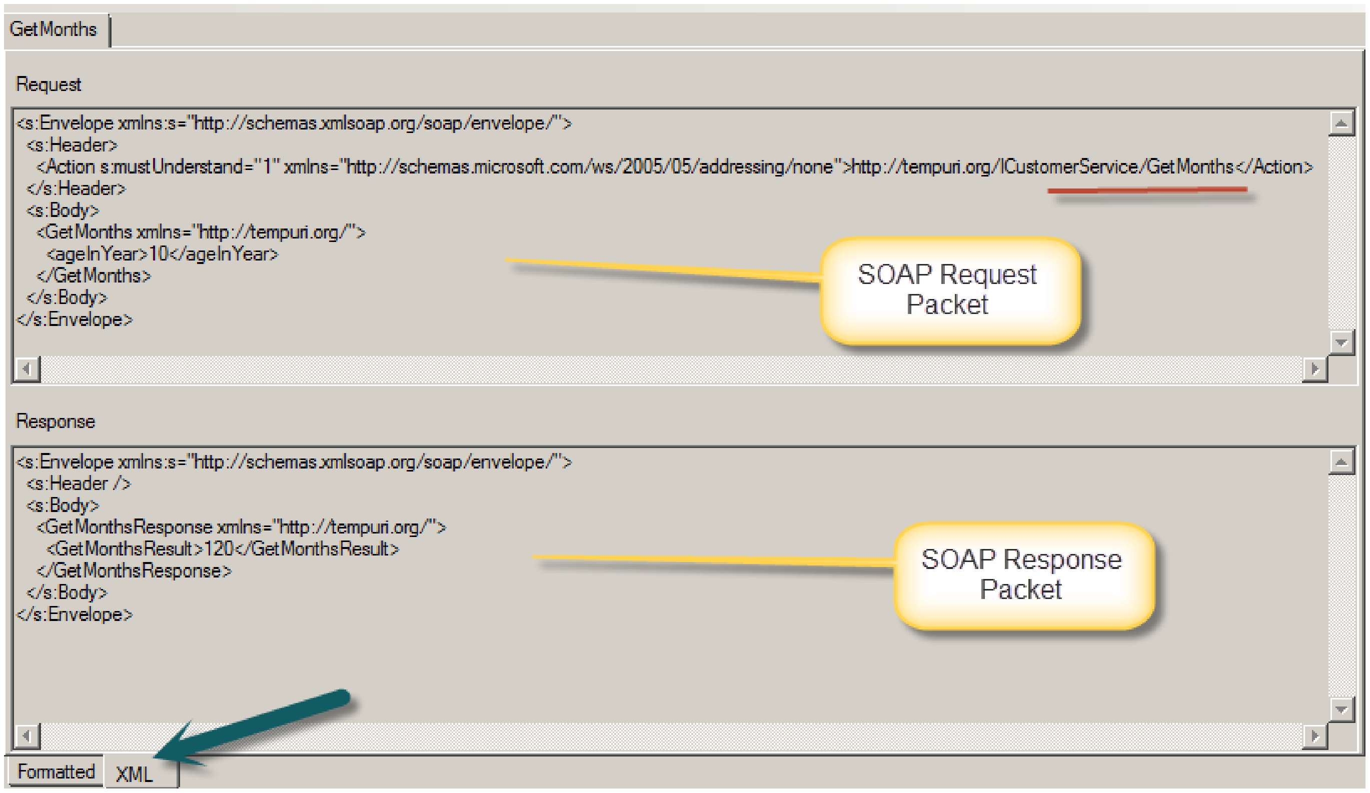
Request

Name	Value	Type
ageInYear	10	System.Int32

Response

Name	Value	Type
(return)	"120"	System.String

17- As current service (by default) is SOAP based web service so communication between client & service will happen using SOAP messages. If you click "XML" at the bottom of the window, you will see the Request SOAP message and Response SOAP message.



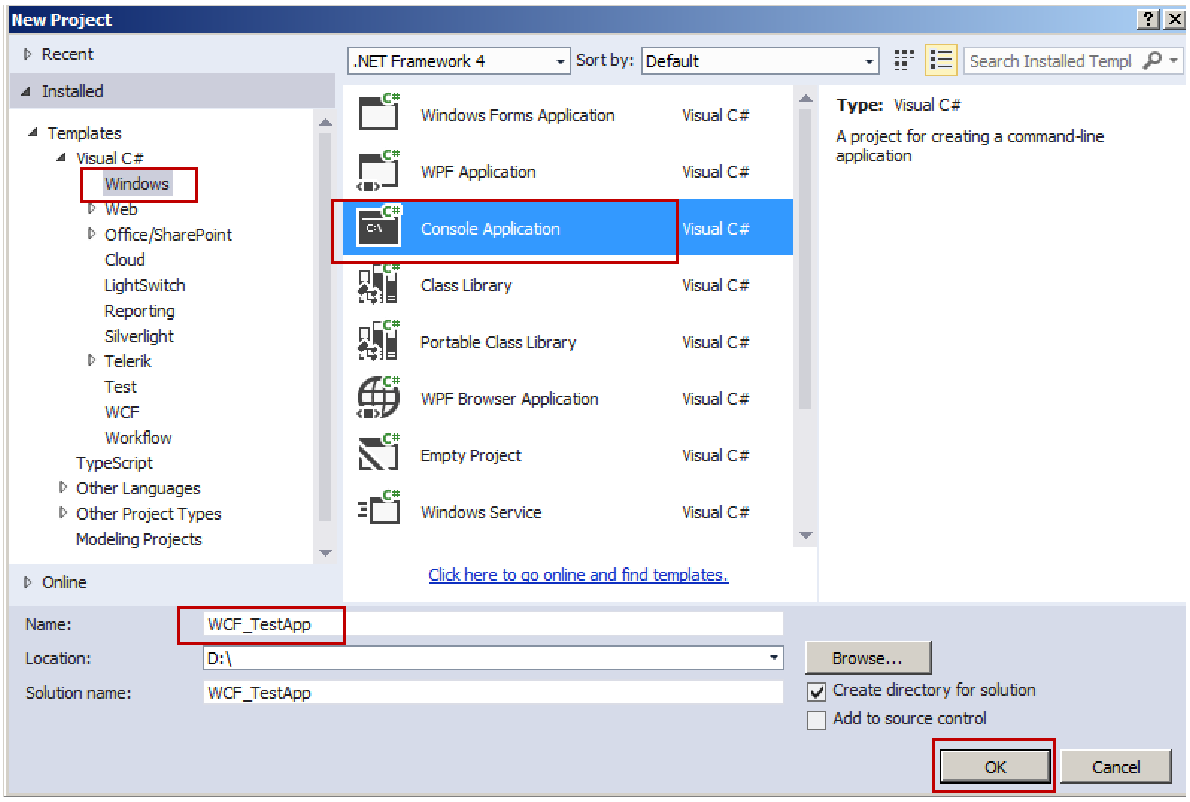
```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Action s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">http://tempuri.org/ICustomerService/GetMonths</Action>
  </s:Header>
  <s:Body>
    <GetMonths xmlns="http://tempuri.org/">
      <ageInYear>10</ageInYear>
    </GetMonths>
  </s:Body>
</s:Envelope>
```

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header />
  <s:Body>
    <GetMonthsResponse xmlns="http://tempuri.org/">
      <GetMonthsResult>120</GetMonthsResult>
    </GetMonthsResponse>
  </s:Body>
</s:Envelope>
```

Consuming a Service in an application

We can consume a service in any other application (e.g. Console, Windows, Web, WCF Service etc.). We are assuming that WCF service is still running. **Note:** You can open service URL in browser to verify if it is running or not.

18- Let's open a new copy of Visual Studio. Create a new project by File -> New -> Project. Choose "Windows" from left panel and "Console Application" from right panel. Give it some name and click "Add". Note: We are not adding a new project in existing solution of WCF but creating a separate Project.



New Project

Recent
Installed
Templates
Visual C#
Windows
Web
Office/SharePoint
Cloud
LightSwitch
Reporting
Silverlight
Telerik
Test
WCF
Workflow
TypeScript
Other Languages
Other Project Types
Modeling Projects
Online

.NET Framework 4 | Sort by: Default | Search Installed Templ

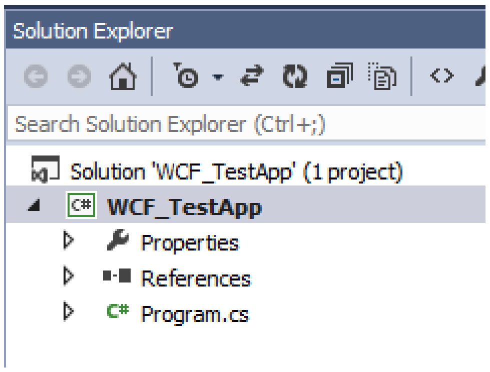
Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Class Library	Visual C#
Portable Class Library	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#

Click here to go online and find templates.

Type: Visual C#
A project for creating a command-line application

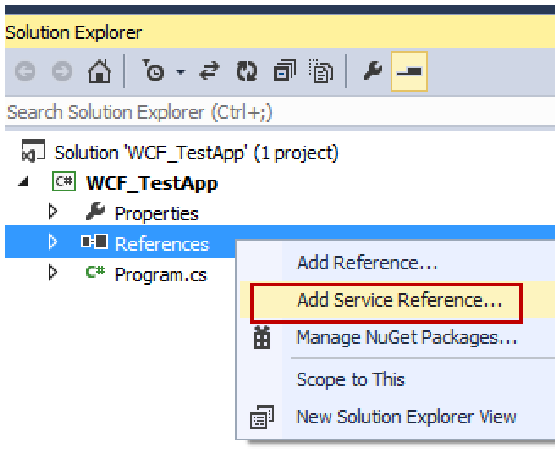
Name: WCF_TestApp
Location: D:\
Solution name: WCF_TestApp
Browse...
[✓] Create directory for solution
[ ] Add to source control
OK Cancel

19- Your application will look like following screenshot

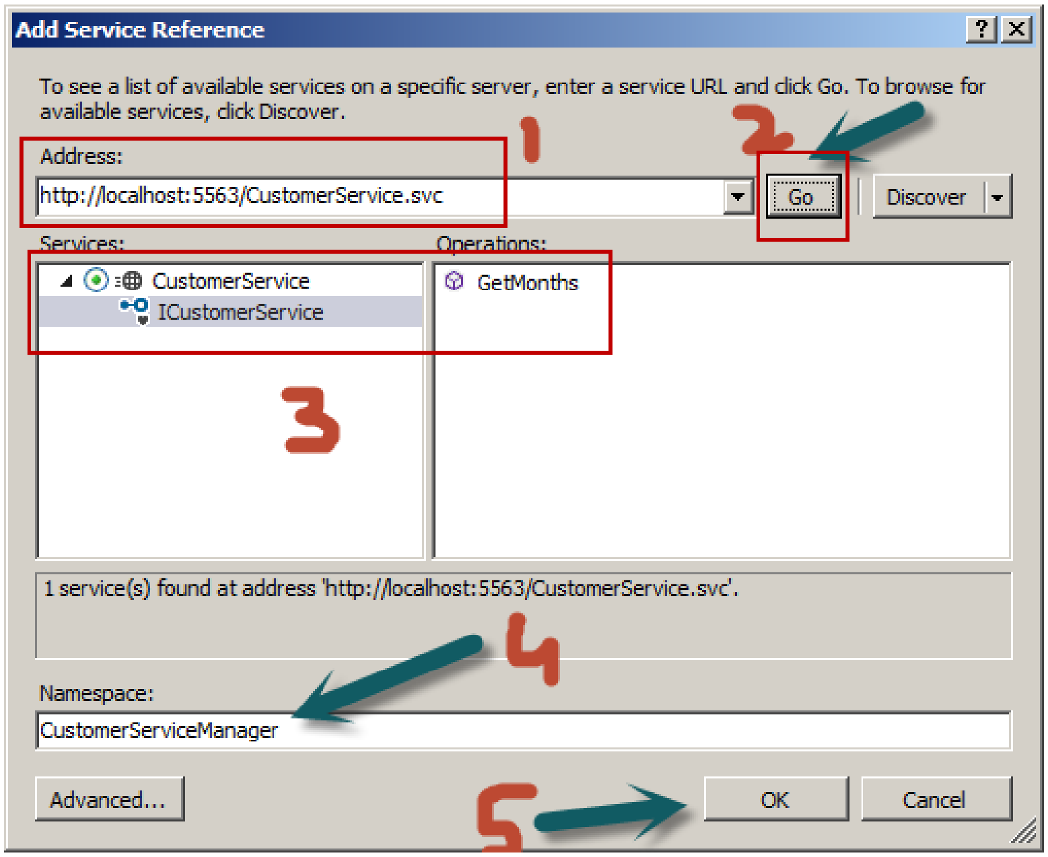


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WCF_TestApp' (1 project)
WCF_TestApp
Properties
References
Program.cs

20- Now to add service reference in this project, right click on "References" and choose "Add Service Reference".



21- In the dialog box, provide the URL of the service (which we've seen in the browser and have also used in WcfTestClient). Check Step 10 for service URL (or check screenshot below). Provide Service URL, then click on "Go" button. You will see detail of service (shown in step 3 below). Then Provide some name of namespace in which proxy classes will be created. Here we've provided "CustomerServiceManager". Then click OK.



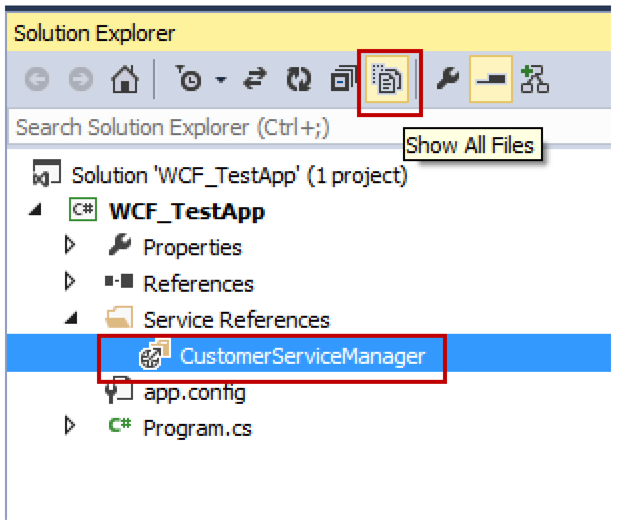
To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:5563/CustomerService.svc

1 service(s) found at address 'http://localhost:5563/CustomerService.svc'.

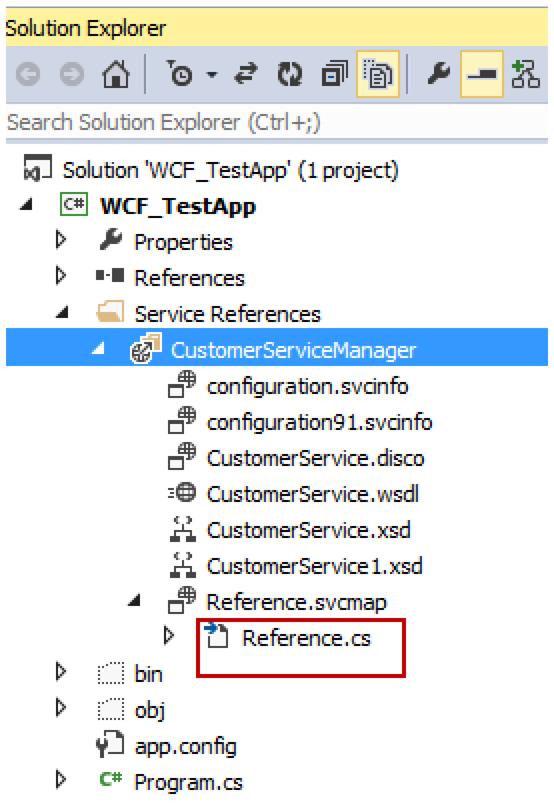
Namespace: CustomerServiceManager

22- You will see a folder “Service References” and then “CustomerServiceManager”. By default, you will see no file under “CustomerServiceManager” folder but you can view them by clicking “Show All Files” icon in top bar.



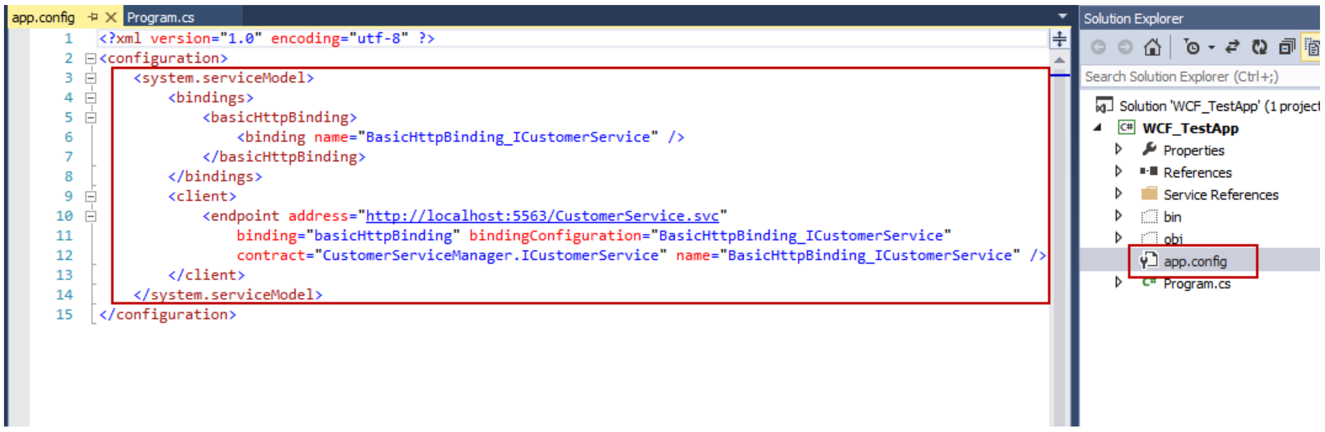
Solution 'WCF_TestApp' (1 project)
WCF_TestApp
Properties
References
Service References
CustomerServiceManager
app.config
Program.cs

23- When “Show All Files” icon will be clicked, you will see something like as shown in following screenshot. Here “Reference.cs” is the file which contains our proxy classes which are generated against our service. You can open it and give a look to it to see what is inside it.



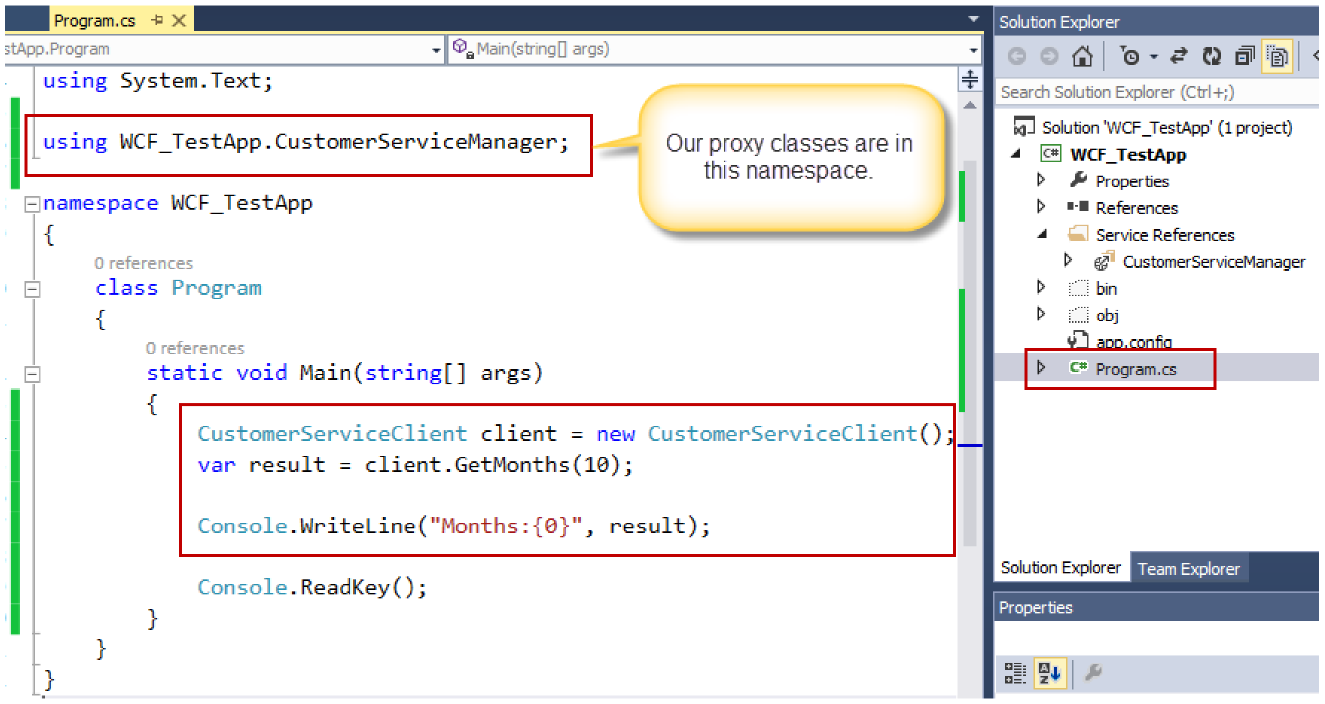
Solution 'WCF_TestApp' (1 project)
WCF_TestApp
Properties
References
Service References
CustomerServiceManager
configuration.svcinfo
configuration91.svcinfo
CustomerService.disco
CustomerService.wsdl
CustomerService.xsd
CustomerService1.xsd
Reference.svcmap
Reference.cs
bin
obj
app.config
Program.cs

24- If you open “app.config” file, you will notice that there is some configuration related to service is also added automatically during the generation of proxy classes. Here we can see ABC (Address, Binding & Contract) of our service. (Nothing to do for you in this step)



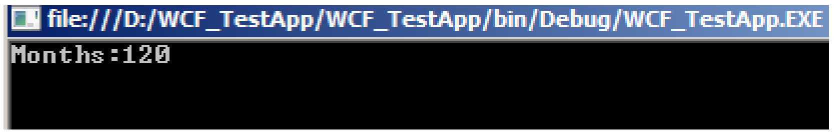
```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_ICustomerService" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:5563/CustomerService.svc"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_ICustomerService"
                contract="CustomerServiceManager.ICustomerService" name="BasicHttpBinding_ICustomerService" />
        </client>
    </system.serviceModel>
</configuration>
```

25- Now let’s use the proxy class to call service method. Open “Program.cs” file and add “using” statement to include namespace of our proxy classes. Check Step 21 for Namespace name. (You can also check “References.cs” file to see what namespace is being used in the file). Add code in “Main” method. Here we are creating instance of our client (proxy) class and then calling method using it. Result is being displayed on a console here.



```csharp
using System.Text;

using WCF_TestApp.CustomerServiceManager;

namespace WCF_TestApp
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            CustomerServiceClient client = new CustomerServiceClient();
            var result = client.GetMonths(10);

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

            Console.ReadKey();
        }
    }
}
```

Our proxy classes are in this namespace.



```
file:///D:/WCF_TestApp/WCF_TestApp/bin/Debug/WCF_TestApp.EXE
Months:120
```

Tasks for Practice

Once you are done with above tutorial. Try to work on these tasks.

1- Add a new Service file in “WCF_Prac” project (Check step 4 for how to do this). Name the service “StudentService”.
2- Add a method in interface and then in service class
    a. Public String ShowName() This method will return your name.
3- Rebuild the service.
4- Test the service using WcfTestClient and check if “ShowName” is working fine or not. You can get URL of service by right click on “StudentService.svc” file and “View in Browser”.
5- In console application (created above), add reference of this StudentService by following Step 20 above.
6- Create instance of this service client and call “ShowName” method.
7- Add a new method in “CustomerService” in WCF_Prac project. Rebuild the project. Now in console application, you will have to update reference. You can do that by right clicking on “CustomerServiceManager” reference and choosing “Update Service Reference” option.
8- Call this method with client instance we’ve created in tutorial and see if everything is working fine.

Useful Links

https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx

http://www.tutorialspoint.com/wcf/