**Objective**

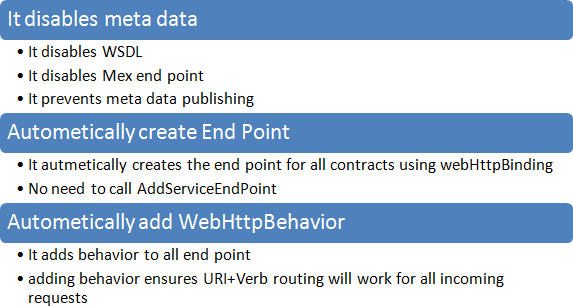
In this article, I will explain

1. What is WebServiceHost Factory class?
2. What is its function?
3. Where to use it?
4. One sample on hosting WCF REST Service using WebServiceHost.

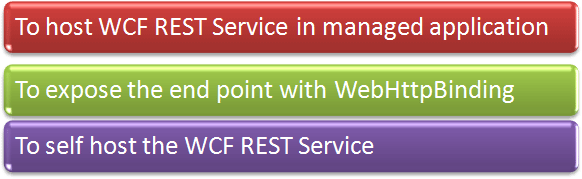
**What is WebServiceHost factory?**

1. This is useful to host WCF REST End points.
2. This is derived from ServiceHost.
3. WebServiceHost has some extra functionality.
4. WebServiceHost works exactly the same way ServiceHost works.

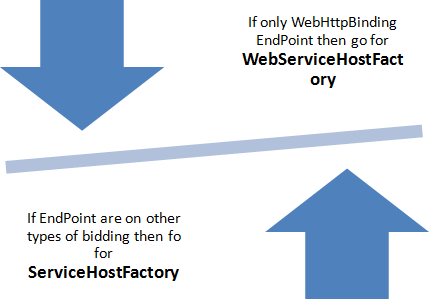
**WebServiceHost factory perform following special task**



**Where to use it?**



**How to choose between ServiceHost and WebServiceHost?**



**Example**

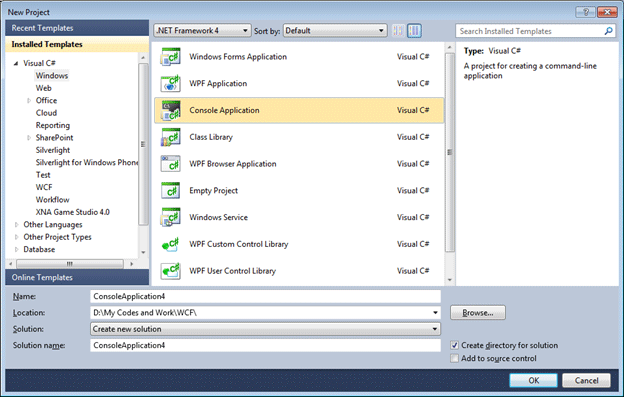
In this example, I will show you step by step, how we can create a REST Service and host it in a console application using WebServiceHost factory.

Below section is taken from my previous posted article on Hosting WCF REST Service in Console Application.

Follow the steps as below,

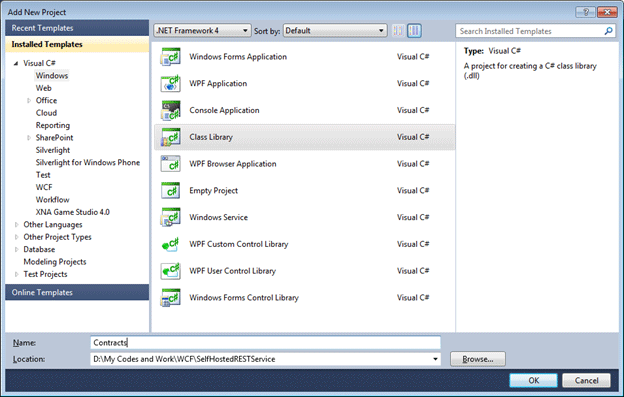
**Step1**

Create a New project. Select Console application as project type.



**Step 2**

Add a new project to same solution.  Choose the project type as class library.



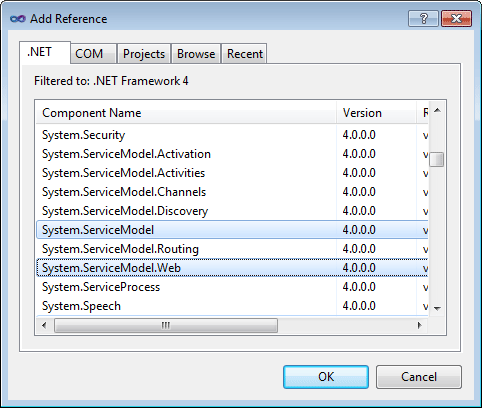
**Step 3**

Add below references in both projects console and service library

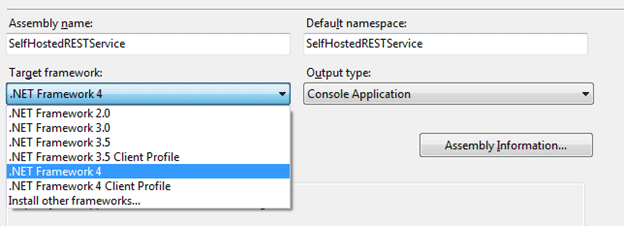
System.ServiceModel;

System.ServiceModel.Description;

System.ServiceModel.Web;



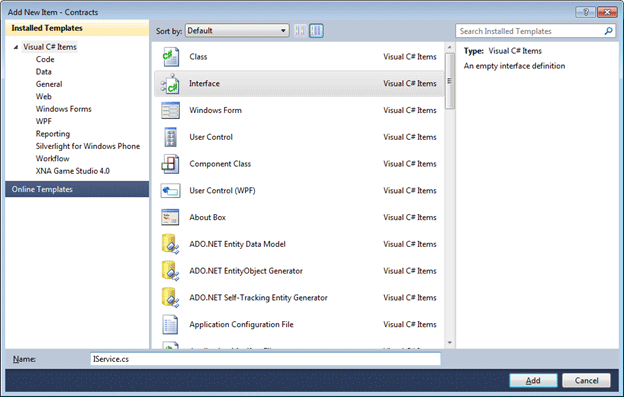
If you are not able to get System.ServiceModel.Web  dll  by default , when you are adding as Add Reference in your console application project and class library project  , then follow the below steps

1. Right click on your console application project or class library project
2. Select properties from context menu
3. From Application tab, if you see by default .NET Framework 4 Client profile is selected. Change that to .NET Framework 4.0. So your target framework should be .NET Framework 4.0.   
     
   

Now you should able to see all the dll when you add service reference in project.

**Step 4**

In this step, I will create contract of service

1. Open the Contract (Class Library project).
2. Delete Class1.
3. Right click and add a new item then select Interface from Code tab.   
     
   
4. Make Interface as public and put ServiceContract attribute.
5. Declare two operation contracts. Make one attributed with WebGet and another attributed with Web Invoke.

**IService.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ServiceModel;

using System.ServiceModel.Web;

namespace Contracts

{

[ServiceContract]

public interface IService

{

[OperationContract]

[WebGet]

string GetMessage(string inputMessage);

[OperationContract]

[WebInvoke]

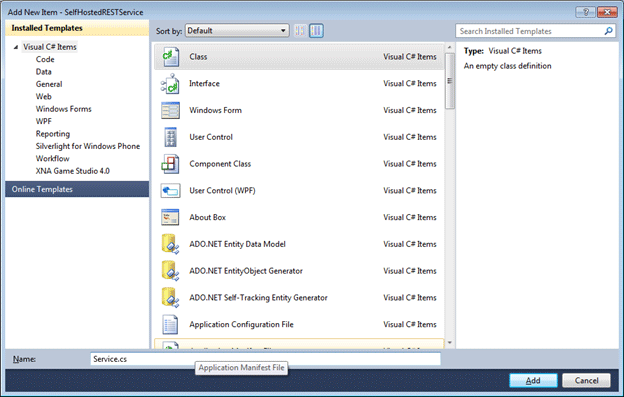
string PostMessage(string inputMessage);

}

}

**Step 5**

In this step, I will implement the contract in Service file. To do so,

1. Right click and add a class in Console application.   
     
   
2. Give any name; I am giving name here Service of the class.
3. Implement the interface (Contract IService) in this class.

**Service.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Contracts;

namespace SelfHostedRESTService

{

public class Service :IService

{

public string GetMessage(string inputMessage)

{

return "Calling Get for you " + inputMessage;

}

public string PostMessage(string inputMessage)

{

return "Calling Post for you " + inputMessage;

}

}

}

**Step 6**

In this step, I will host the service in a console application. So to do so

1. Open Program.cs
2. Create instanced of WebServieceHostFactory   
     
   10.gif
3. Add a service end point   
     
   11.gif
4. Add service debug behavior   
     
   12.gif

As we know, REST service used webHttpBindding.  And we need to make sure that HttpHelpPageEanbled is false.

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ServiceModel;

using System.ServiceModel.Description;

using System.ServiceModel.Web;

using Contracts;

namespace SelfHostedRESTService

{

class Program

{

static void Main(string[] args)

{

WebServiceHost host = new WebServiceHost(typeof(Service), new Uri("http://localhost:8000"));

ServiceEndpoint ep = host.AddServiceEndpoint(typeof(IService),new WebHttpBinding(), "");

ServiceDebugBehavior stp=host.Description.Behaviors.Find<ServiceDebugBehavior>();

stp.HttpHelpPageEnabled = false;

host.Open();

Console.WriteLine("Service is up and running");

Console.WriteLine("Press enter to quit ");

Console.ReadLine();

host.Close();

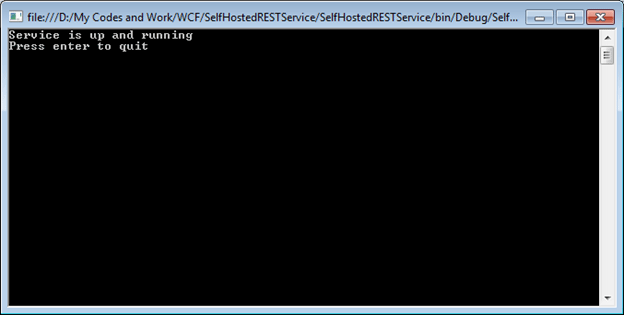
}

}

}

**Step 7**

Just press F5 to run the service.



Now once service is up and running, I can test,

[OperationContract]

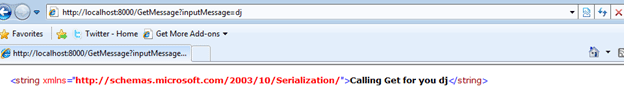
[WebGet]

string GetMessage(string inputMessage);

In browser.  Because it is attributed with WebGet. To test this, I need to browse to URL,

http://localhost:8000/GetMessage?inputMessage=dj

If you see WebServiceHost , I  am hosting my REST service  in http://localhost:8000/ and then I need to append the method name and then parameter I need to pass.



**Step 8**

In this step, I will create a Console client which will consume the REST client.  So to do so , follow the below steps

1. Create a console project.
2. Add reference of System.ServiceModel and System.ServiceModel.[Web](http://www.watchesn.com) (See Step 3).
3. Add reference of Contract class library (we created in Step 2)
4. Write below code in Program.cs

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.ServiceModel.Description;

using Contracts;

namespace ClientForRESTSelfHosted

{

class Program

{

static void Main(string[] args)

{

ChannelFactory<IService> cf = new ChannelFactory<IService>(new WebHttpBinding(), "http://localhost:8000");

cf.Endpoint.Behaviors.Add(new WebHttpBehavior());

IService channel = cf.CreateChannel();

Console.WriteLine(channel.GetMessage("Dhananjay Get"));

Console.WriteLine(channel.PostMessage("Dhananjay Post"));

Console.Read();

}

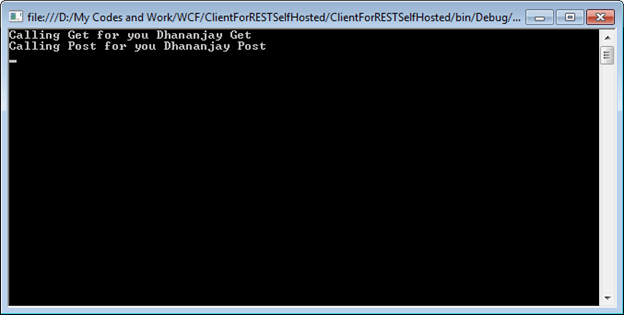
}

}

**Explanation**

1. I am creating an instance of Channel Factory.
2. I am passing IService (from dll of Contract class library) as type to channel factory.
3. I am passing as parameter type of binding that is WebHttpBinding and address.
4. Address is exactly as same as of the address was in host instance.
5. I am creating a channel of type IService1.
6. Calling the methods on the channel.

Press F5 to run the application



I hope this post was useful. Thanks for reading. Happy coding.