[老老实实学WCF] 第一篇 Hello WCF

老老实实学WCF

第一篇 Hello WCF

WCF(Windows Communication Foundation)是微软公司推出的面向服务技术的集大成者，涵盖继承了其之前发布的所有的分布式应用程序的编程模型，涉及面之广，技术之复杂，结构之零散，让我们初学这门技术的菜鸟时常有无处下手的感觉，此系列博文系笔者艰难探索WCF技术过程的学习笔记，笔者抱着老老实实的态度，力图扎扎实实，循序渐进地学好这门技术，文中难免有疏漏和错误之处，还请诸位大牛不吝赐教。仅以此为与诸位共同研讨，与求索WCF技术的同学们共勉。

1.必备的基础知识

WCF技术涉及的知识点和概念庞杂零散，时常会被各种教材资料开头便抛出的诸多名词概念弄晕，因此我们先了解必备的知识，尽快进入动手编码。

(1) 体系中的角色。

> 在WCF的结构中，至少应该有两个角色：服务端和客户端。服务端公开一个或多个服务，客户端访问这些服务以获得相应的逻辑。

(2) 服务(Service)和操作(Operation)

> 服务端公开了一个或多个服务，每个服务都拥有一个或多个操作，客户端通过调用服务的操作来获得服务提供的逻辑。

(3) 终结点(EndPoint)

> 服务端公开了服务和操作，客户端如何访问到他们呢？就是通过终结点，终结点相当于服务的地址，类似于门牌号，客户端就是循着这个地址来找到服务和操作的。终结点是和服务相对应的，找到了终结点就找到了服务，找到了服务就获得了操作。

(4) 绑定(Binding)

> 绑定描述了客户端与服务端沟通的方式，双方沟通的语言，或者说协议，双方必须保持一致才能相互沟通，比如一个服务端把服务公开，要求必须用http协议来访问，那么此时绑定就是http绑定，客户端必须用http的方式来与这个服务端通信才能访问到服务。

(5) 元数据(Metadata)

> 现在服务端公开了服务，客户端循着正确的终结点，持着正确的绑定找到了服务端，此时服务端笑而不语，客户端不知所措，因为客户端根本不知道服务端公布了那些服务、操作、数据。客户端一个调用也写不出来。虽然客户端不需知道服务和操作的具体实现细节，但是它必须知道服务的接口描述(或类型描述)、操作的方法签名，数据的类描述，这些描述就叫做元数据，服务端必须通过某种方法把元数据交给客户端，这种方法叫做元数据交换(Metadata Exchange)。

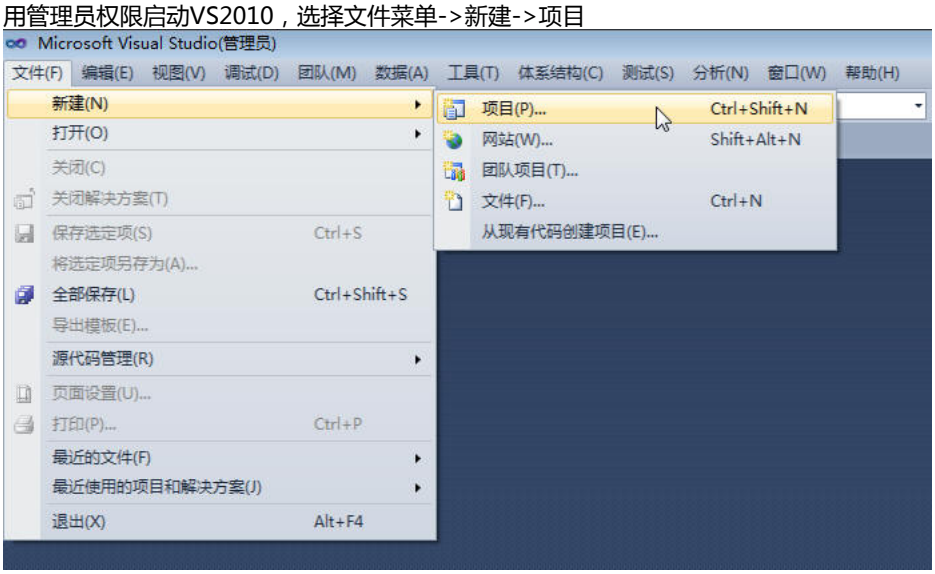
了解了这些概念的大概意思就可以了，我们在后面慢慢研究。

2. 开始动手: Hello WCF

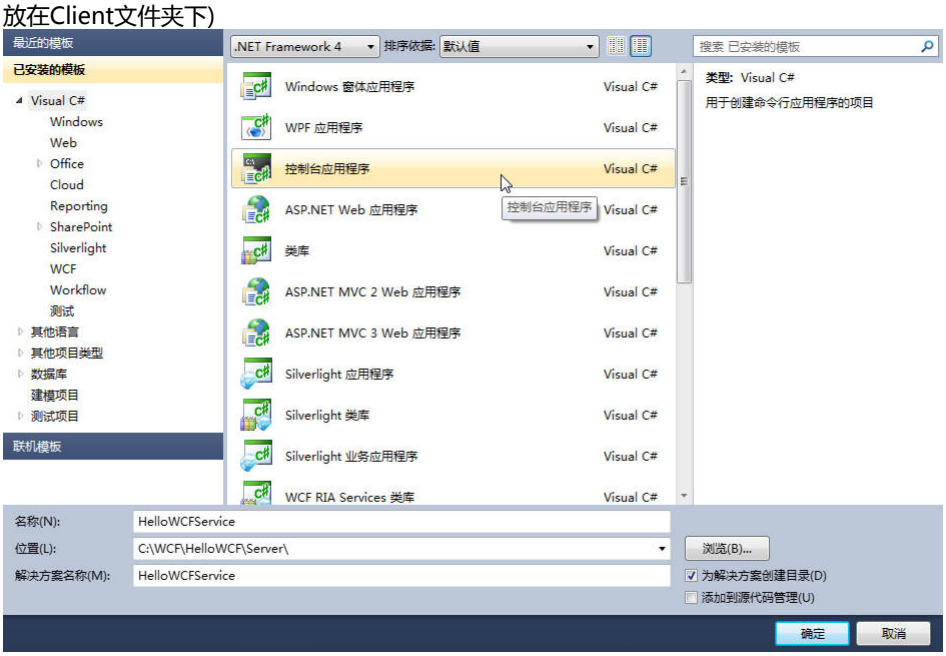
现在就开始动手编码，我先说明一下我的实验环境：
Windows 7 家庭高级版
.Net Framework 4.0
Microsoft Visual Studio 2010 旗舰版 with Service Pack 1

(1) 创建服务端程序

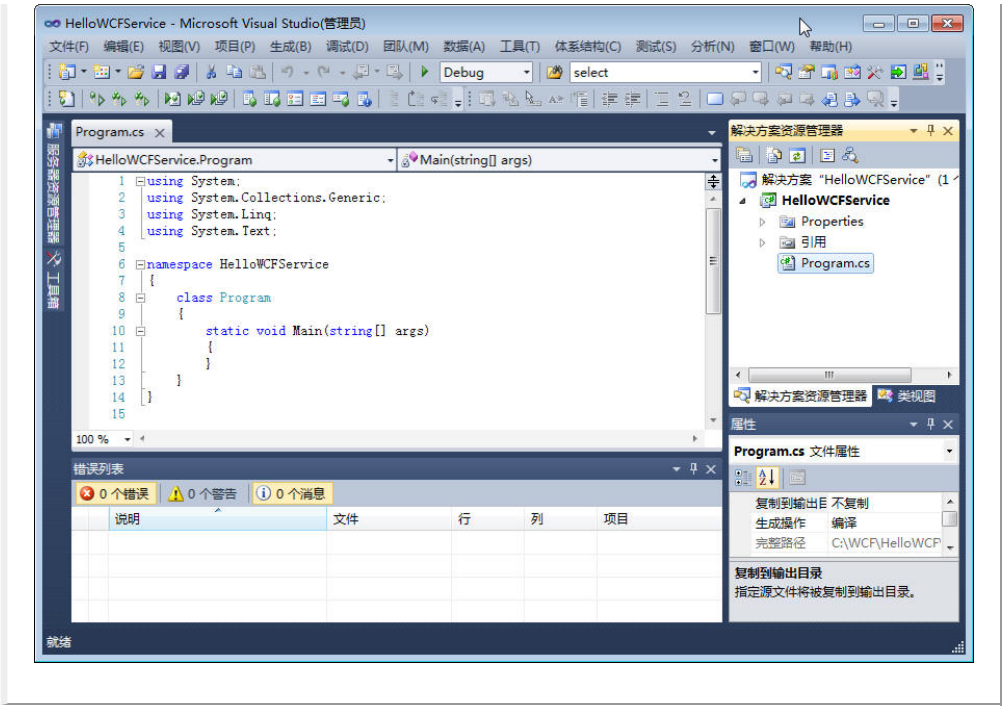
WCF的服务必须有一个宿主来承载，这个宿主的选择很多，控制台程序、窗体程序、IIS、Windows服务等等。我就选一个最简单的控制台程序来承载。
用管理员权限启动VS2010，选择文件菜单->新建->项目



选择C#控制台应用程序，取名为HelloWCFService，放在Server文件夹下(待会儿的客户端放在Client文件夹下)



这样，我们得到了一个干净的控制台应用程序，只有一个Program.cs。



(2) 创建服务协定与实现服务协定

我们在前文的概念中了解到了服务，这里又碰到了服务协定，在这里有必要展开一下，我们已经理解了什么是服务，然而服务毕竟是个很笼统的概念，在编程模型中，服务是如何实现的呢？他的编程语言表示法又是什么呢，就是服务协定(Service Contract)与服务类(Service Class)，一般情况下，我们把服务协定表示为一个接口，然后用一个类去实现这个接口，实现这个接口的类就叫做服务类了。这个接口和这个类共同描述了一个服务。编写接口的过程叫做创建服务协定，编写实现接口的类叫做实现服务协定。举例说明：
下面的代码描述了怎样创建一个服务协定

```
[csharp]

1. [ServiceContract]
2. interface IHelloWCFService
3. {
4.     [OperationContract]
5.     string HelloWCF();
6. }
```

所有标记为 [ServiceContract] 属性的接口都表示该接口是一个服务协定，所有标记为 [OperationContract] 属性的方法都表示该方法是一个操作协定，服务协定和操作协定是可以被客户端调用到的，没有这些标记的接口和方法，服务端本地当然还可以用，但是客户端就访问不到了。

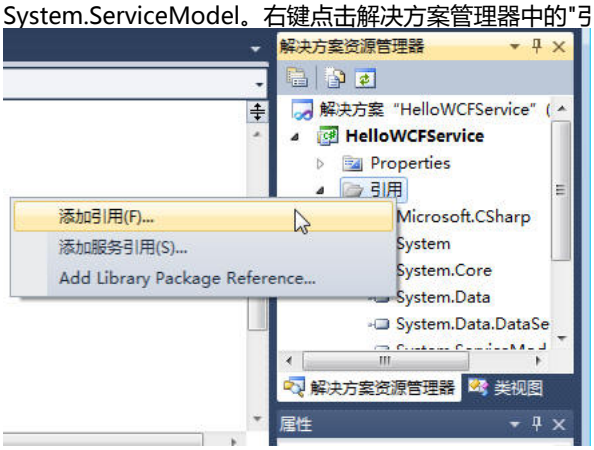
创建了服务协定后，我们再写一个类去实现它，那么这个服务就有了实体，这个实体是它逻辑真正存在的地方。举例说明：
下面的代码描述了怎样实现一个服务协定

```
[csharp]

1. public class HelloWCFService:IHelloWCFService
2. {
3.     public string HelloWCF()
4.     {
5.         return "Hello WCF!";
6.     }
7. }
```

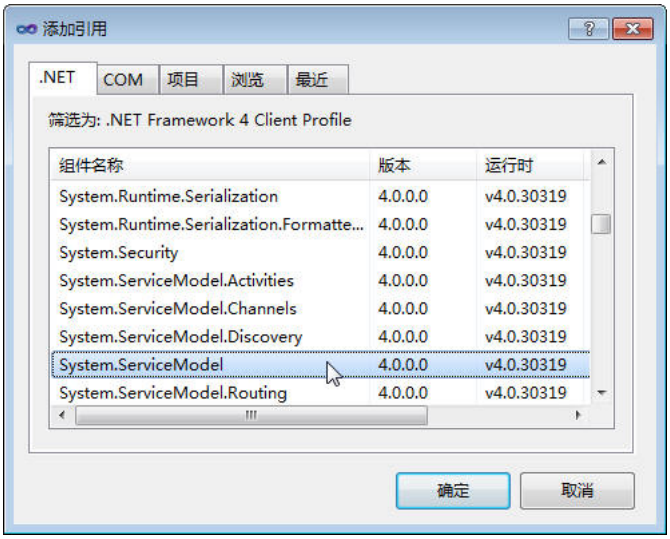
这个类实现了服务协定接口而成为了服务类，他实现了操作协定HelloWCF()，逻辑很简单，返回一个字符串"Hello WCF!"。

[ServiceContract]和[OperationContract]属性也可作用于普通的类和方法，我们也可以不编写接口而直接把属性作用于HelloWCFService类，这样创建服务协定和实现服务协定就合二而一了。但是推荐用接口和实现分开的方法，可以带来灵活性的优点，这个我体会的也不多，等以后再展开。

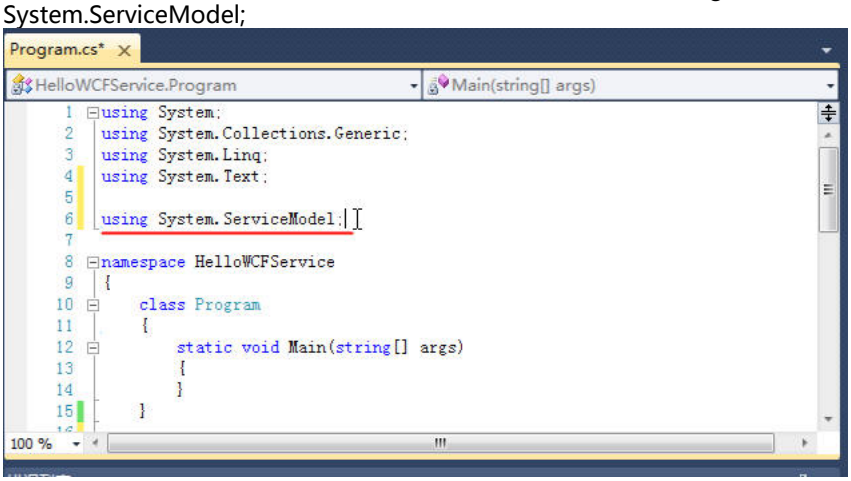
现在我们来动手。
在编写和实现服务协定之前，我们要先添加一个引用，WCF的核心部件都在其中，他就是System.ServiceModel。右键点击解决方案管理器中的"引用"，选择"添加引用"



选择.Net标签卡中System.ServiceModel程序集。



添加完程序集引用后，我们要在源代码中导入命名空间，加一条 using System.ServiceModel;



然后我们把服务协定的定义和实现代码输入，就在Program.cs中，写在Program类的后面。

```
namespace HelloWCFService
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }

    [ServiceContract]
    interface IHelloWCFService
    {
        [OperationContract]
        string HelloWCF();
    }

    public class HelloWCFService : IHelloWCFService
    {
        public string HelloWCF()
        {
            return "Hello WCF!";
        }
    }
}
```

编译一下，没有问题，到这里，服务协定的建立和实现就完成了。

(3) 配置和承载服务。

我们写好了服务，接下来要在宿主中把它承载起来，才能被客户端访问和调用。WCF使用ServiceHost 的实例来承载服务，像我们这样的控制台程序，我们需要自己手动创建ServiceHost 实例，如果把服务寄存在IIS 中，那么IIS会替我们生成和管理ServiceHost 实例。

我们知道，我们的服务端很可能承载着很多的服务，每一个服务都有一个地址与之对应，而一个服务可以公开多个终结点，每一个终结点都有一个地址，比方说，服务好象一栋房子，房子的地址是唯一的，如和平大街1号，而终结点相当于房门，每个房门都有个房门号，这个号码也是地址，他们是不同的，但是穿过这道门，你进入的都是这个房子。
更进一步说，我们所说的服务，实际上服务实现类本体(即HelloWCFService)，在复杂的情况下，服务实现类本体可能实现多个服务协定接口，通过不同的服务协定接口访问服务实现类本体所获得的操作是不一样的。服务协定接口的信息是存储在终结点中的，因此客户端循着不同的终结点得到不同的服务协定，进而访问服务实现类而获得不同的操作，还是刚才的例子，这栋服务的房子可能有很多房间，你从车库那个门(车库协定接口的终结点)进入房子就只能得到车库的操作，而无法看到客厅里的情况。这就是多终结点(或者说多地址)可以对应同一服务的原因。
为什么插播这么多呢，这个要帮助我们理解地址(Address)。如果理解不了上面说的也没关系，我们可以想象一下，有一家银行(服务)坐落于和平大街1号。这个银行有两个门，1号门和2号门。1号门是对公业务，2号门是对私业务。如果某公司要办理一笔贷款，那么这个办事的人(客户端) 就会循着地址找到1号门进去办事了。
这很好理解。看下面两个概念：
服务地址：和平大街1号(即房子的地址，即HelloWCFService服务的地址，只有一个)
终结点地址：和平大街1号1号门(即门的地址，即终结点的地址，存在多个，每个终结点都有一个)
可以看出，终结点地址是服务地址的子地址，或相对地址；服务地址是终结点地址的父地址，也称基地址(BaseAddress)。

好，我们回到ServiceHost，每一个ServiceHost 实例对应和管理一个服务，因此，在建立ServiceHost 之前，要先为其指定一个基地址(即服务地址)，这样ServiceHost 就能根据基地址来为自己管理的服务建立的终结点的相对地址了。

下面的代码建立一个Uri对象，这个对象可以作为基地址传送给ServiceHost。

```
[csharp]
1. Uri baseAddress = new Uri("http://localhost:8000/MyService");
```

这里使用的localhost 表示本机，8000端口是随便选的，只要不和系统保留端口冲突就可以了，后面的MyService是服务地址，这个名字可以随便起，和服务实现类的名字相同也可以，他们没有联系。

接下来，我们正式建立ServiceHost，由于我们只有一个服务(HelloWCFService类)，所以只建立一个ServiceHost就可以了。
下面的代码建立一个ServiceHost对象

```
[csharp]
1. ServiceHost host = new ServiceHost(typeof(HelloWCFService), baseAddress);
```

ServiceHost 的构造函数接受两个参数，第二个参数基地址就不用说了；第一个参数是服务类型，注意看这里是HelloWCFService类，而不是IHelloWCFService接口，这里我们看出，服务的本体是服务实现类，而非协定接口，毕竟实实在在的逻辑在实现类里面，我们要寄宿的是它。

服务已经寄宿好了(还没有运行，只是定义好了寄宿主)，接下来要公开终结点了，就是为房子造门了，门是属于房子的，当然应该由房子来负责建立。
下面的代码为服务添加一个终结点

```
[csharp]
1. host.AddServiceEndpoint(typeof(IHelloWCFService), new WSHttpBinding(), "HelloWCFService");
```

第一个参数指定了服务协定，请注意，这里是协定接口类型了，一个服务实现类可能实现多个服务协定接口，这样每个服务协定接口都有一个终结点与之对应了。第二个参数定义了绑定，即客户端与服务端沟通的方式，这里指定了一个wsHttpBinding的实例，第三个参数是终结点的地址。这里给了一个相对地址，终结点的最终地址将会把这个相对地址与基地址组合，即

```
[html]
1. http://localhost:8000/MyService/HelloWCFService
```

到这里，我们把宿主、服务、终结点都建好了，先把代码敲上去

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.ServiceModel;
using System.ServiceModel.Description;

namespace HelloWCFService
{
    class Program
    {
        static void Main(string[] args)
        {
            Uri baseAddress = new Uri("http://localhost:8000/MyService");

            ServiceHost host = new ServiceHost(typeof(HelloWCFService), baseAddress);

            host.AddServiceEndpoint(typeof(IHelloWCFService), new WSHttpBinding(), "HelloWCFService");
        }
    }
```

不要着急，不要着急，休息，休息一会儿。

这样是不是就OK了呢？恩，还差一点儿。

我们还没有启用元数据交换，也就是说客户端还没有得到关于服务协定的任何信息，它无法做出任何调用。所以，我们得想个办法让服务端把自己的服务协定元数据交给客户端。
启用元数据交换，是通过服务元数据行为的相关设置来实现的，要进行这方面的设置，首先要引入命名空间: System.ServiceModel.Description

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.ServiceModel;
using System.ServiceModel.Description;

namespace HelloWCFService
{
    class Program
    {
        static void Main(string[] args)
        {
            Uri baseAddress = new Uri("http://l

            ServiceHost host = new ServiceHost(

            host.AddServiceEndpoint(typeof(IHel
        }
```

接下来，我们创建一个服务元数据行为对象。然后把它加入到宿主ServiceHost 对象的行为集合中去，这样元数据交换就被打开了
以下代码建立一个服务元数据行为对象

```
[csharp]
1. ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
```

构造函数没有参数，这真是我们喜闻乐见的。

服务元数据行为有一个属性HttpGetEnabled，我们要把它设为true ，这样，客户端就可用HTTP GET的方法从服务端下载元数据了。
以下代码设置HttpGetEnabled属性

```
[csharp]
1. smb.HttpGetEnabled = true;
```

最后，把这个服务元数据行为对象添加到宿主ServiceHost 对象的行为集合中去。

```
[csharp]
1. host.Description.Behaviors.Add(smb);
```

这样，服务的元数据交换就配置好了。(其实这样不算配置好，必须添加元数据交换终结点才行，只是.Net Framework 4.0替我们做了，以后再展开)

我们把代码敲上去

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.ServiceModel;
using System.ServiceModel.Description;

namespace HelloWCFService
{
    class Program
    {
        static void Main(string[] args)
        {
            Uri baseAddress = new Uri("http://localhost:8000/MyService");

            ServiceHost host = new ServiceHost(typeof(HelloWCFService), baseAddress);

            host.AddServiceEndpoint(typeof(IHelloWCFService), new WSHttpBinding(), "HelloWCFService");

            ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
            smb.HttpGetEnabled = true;
            host.Description.Behaviors.Add(smb);
        }
    }
```

最后一步，启动宿主，开始服务。

```
[csharp]

1.  host.Open();
```

因为是控制台程序，我们需要添加一些输出和一些等待输入来保持程序的持续运行。等待用户输入任意键结束程序

```
[csharp]

1.  Console.WriteLine("Service is Ready");
2.  Console.WriteLine("Press Any Key to Terminated...");
3.  Console.ReadLine();
```

最后关闭宿主

```
[csharp]

1.  host.Close();
```

到此，服务的配置和承载就完成了。代码敲上去

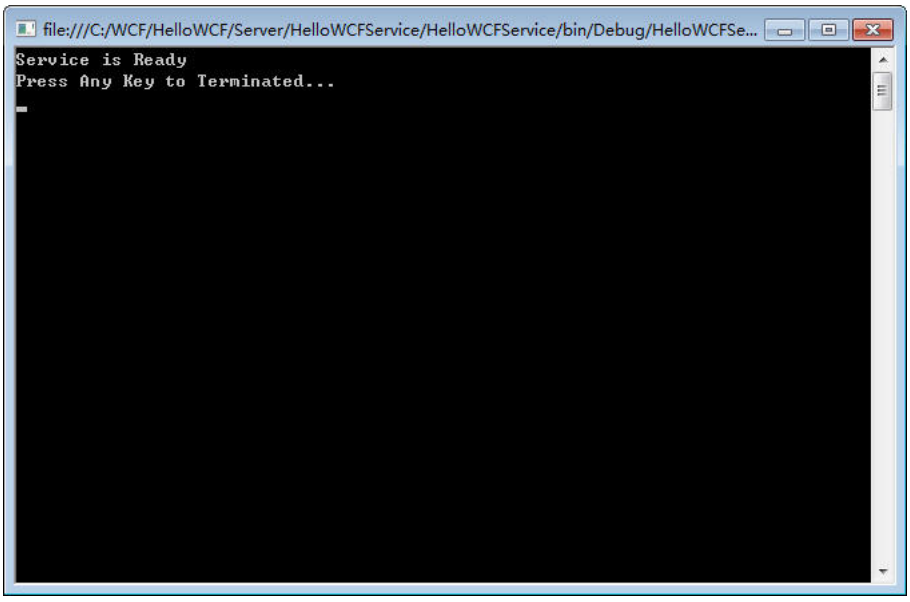
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.ServiceModel;
using System.ServiceModel.Description;

namespace HelloWCFService
{
    class Program
    {
        static void Main(string[] args)
        {
            Uri baseAddress = new Uri("http://localhost:8000/MyService");

            ServiceHost host = new ServiceHost(typeof(HelloWCFService), baseAddress);

            host.AddServiceEndpoint(typeof(IHelloWCFService), new WSHttpBinding(), "HelloWCFService");

            ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
            smb.HttpGetEnabled = true;
            host.Description.Behaviors.Add(smb);

            host.Open();

            Console.WriteLine("Service is Ready");
            Console.WriteLine("Press Any Key to Terminated...");
            Console.ReadLine();

            host.Close();
        }
    }
```

以下是服务端程序的所有代码：

```
[csharp]
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using System.ServiceModel;
using System.ServiceModel.Description;

namespace HelloWCFService
{
    class Program
    {
        static void Main(string[] args)
        {
            Uri baseAddress = new Uri("http://localhost:8000/MyService");

            ServiceHost host = new ServiceHost(typeof(HelloWCFService), baseAddress);

            host.AddServiceEndpoint(typeof(IHelloWCFService), new WSHttpBinding(), "HelloWCFServ

            ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
            smb.HttpGetEnabled = true;
            host.Description.Behaviors.Add(smb);

            host.Open();

            Console.WriteLine("Service is Ready");
            Console.WriteLine("Press Any Key to Terminated...");
            Console.ReadLine();

            host.Close();
        }
    }

    [ServiceContract]
    interface IHelloWCFService
    {
        [OperationContract]
        string HelloWCF();
    }

    public class HelloWCFService : IHelloWCFService
    {
        public string HelloWCF()
        {
            return "Hello WCF!";
        }
    }
}
```

按F5运行一下，你看到程序是下面的样子。



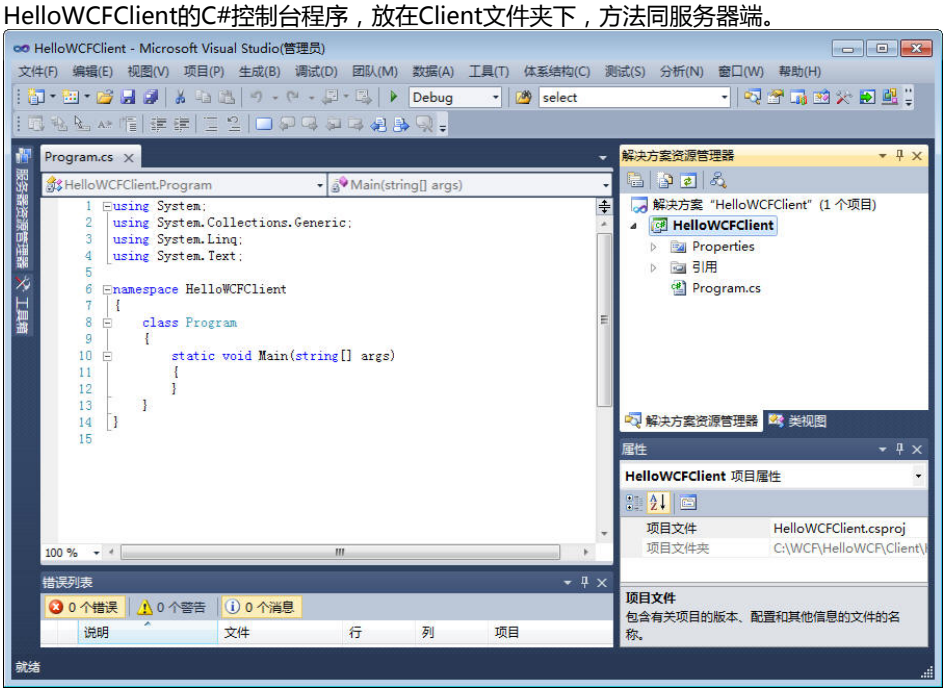
为了验证我们的宿主和服务运行正常，在上面的程序运行的时候，不要关掉它，然后打开浏览器，输入服务地址：

```
[html]
1. http://localhost:8000/MyService
```

如果你看到如下的画面，就表示宿主和服务运行正常。

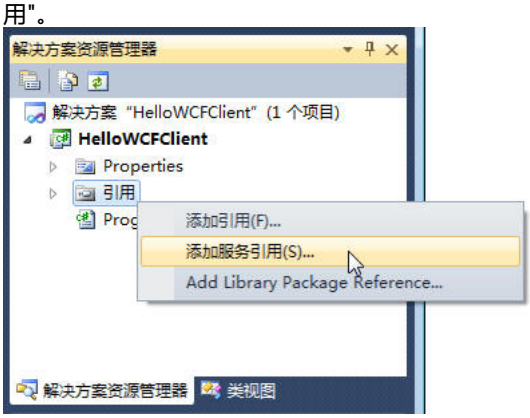
(4) 创建和使用客户端

接下来我们来做客户端的部分，仍然是使用控制台应用程序。建立一个名为HelloWCFClient的C#控制台程序，放在Client文件夹下，方法同服务器端。

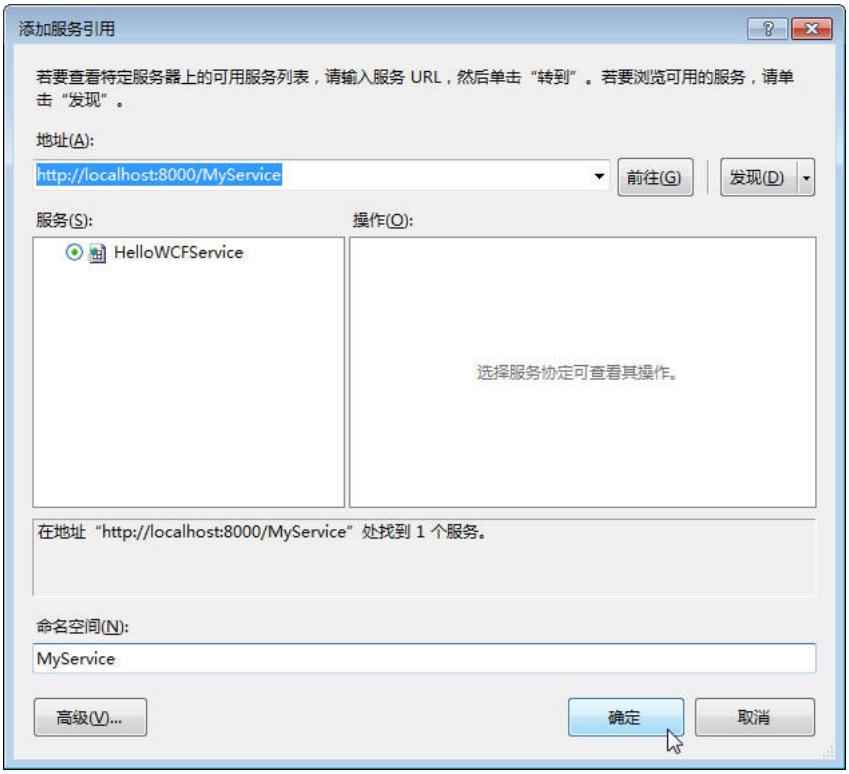


要调用服务端的服务，必须先得到服务协定和操作协定的描述元数据，刚才我们已经打开了服务端的元数据交换功能，现在可以直接去下载了，VS2010 提供了添加服务引用的功能，让我们很方便的得到服务的元数据(还可以使用Svcutil实用工具来生成元数据再包含在客户端中，以后展开)。

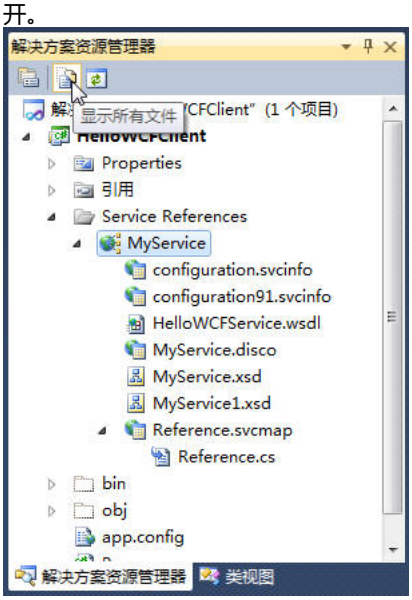
首先要保证服务端的宿主是运行着的，然后右击客户端项目中的"引用"，选择"添加服务引用"。



在弹出的对话框中的地址一栏输入服务端的服务地址，点击"前往"，VS2010会前往服务地址处下载元数据，然后报告发现的服务，在下面的命名空间中为我们的元数据类型建立一个新的命名空间，这里可以随意命名，我就把她命名为MyService



点击确定，我们可以看到引用中多了许多东西，这些都是元数据，具体内容以后再研究展开。



添加完服务引用后，VS2010 会为我们建立一个客户端代理类，名字以服务名+Client来命名，所以我们这里的代理类叫做HelloWCFServiceClient，就在我们刚刚指定的MyService命名空间下，我们创建一个这个类的对象，然后通过这个对象来调用服务操作。

```
[csharp]
1. MyService.HelloWCFServiceClient client = new MyService.HelloWCFServiceClient();
```

使用client 对象，我们就可以调用服务中公开的操作了。

```
[csharp]
1. string strRet = client.HelloWCF();
```

操作协定HelloWCF 是我们在服务协定中定义，在服务类中实现的，它返回一个字符串"Hello WCF!"。我们建立一个string 变量来接收它。

然后我们把结果输出，并关闭代理。

```
[csharp]

Console.WriteLine(strRet);
Console.ReadLine();

client.Close();
```

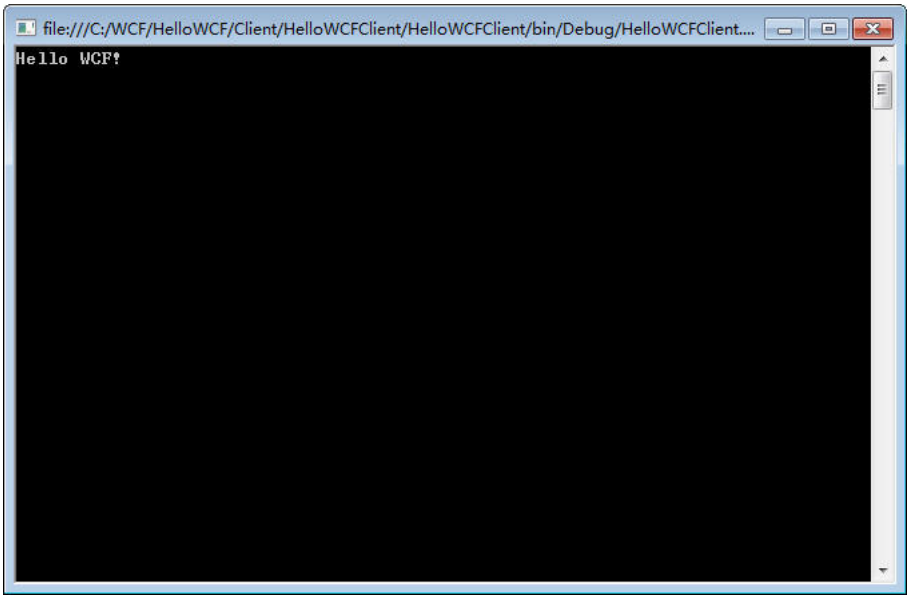
至此，客户端的部分也完成，把代码敲上去。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWCFClient
{
    class Program
    {
        static void Main(string[] args)
        {
            MyService.HelloWCFServiceClient client = new MyService.HelloWCFServiceClient();

            string strRet = client.HelloWCF();

            Console.WriteLine(strRet);
            Console.ReadLine();

            client.Close();
        }
    }
}
```

客户端部分的完整代码：

```
[csharp]

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWCFClient
{
    class Program
    {
        static void Main(string[] args)
        {
            MyService.HelloWCFServiceClient client = new MyService.HelloWCFServiceClient();

            string strRet = client.HelloWCF();

            Console.WriteLine(strRet);
            Console.ReadLine();

            client.Close();
        }
    }
}
```

按F5运行客户端程序，记得此时必须保持服务端程序是运行着的，你会看到下面的结果



大功告成，第一个WCF 程序完成了

Hello WCF!

3. 总结

通过这个例子，我们以最简单的方式实现了一个WCF 程序，我们应该了解一个WCF 程序最基本的编程步骤。
(1) 定义服务协定接口和接口中的操作协定。
(2) 用一个服务类实现服务协定接口和接口中的操作协定。
(3) 建立一个宿主程序用来寄存服务。
[针对控制台程序宿主]
(4) 为服务建立ServiceHost 对象并指定要寄存的服务类和服务基地址。
(5) 为ServiceHost 添加终结点，指定其服务协定、绑定类型和终结点地址(或相对地址)。
(6) 为ServiceHost 添加服务元数据行为对象，要将该对象的HttpGetEnabled属性设为true以启动元数据交换。
(7) 启动ServiceHost。
(8) 建立客户端。
(9) 为客户端添加指向服务基地址的服务引用以下载元数据。
(10) 客户端使用代理类调用服务操作。

简而言之。
(1) 定义服务协定。
(2) 实现服务协定。
(3) 配置和建立宿主。
(4) 配置和建立客户端。
(5) 调用服务。

4. 相关资源

徐长龙老师播讲的MSDN Webcast 教程<<跟我一起从零开始学WCF>>
http://msdn.microsoft.com/zh-cn/hh148206 (http://msdn.microsoft.com/zh-cn/hh148206)

MSDN 技术资源库中 WCF的入门教程，本例主要参考该教程
http://msdn.microsoft.com/zh-cn/library/ms734712.aspx
(http://msdn.microsoft.com/zh-cn/library/ms734712.aspx)