

[老老实实学WCF] 第六篇 元数据交换

老老实实学WCF

第六篇 元数据交换

通过前两篇的学习，我们了解了WCF通信的一些基本原理，我们知道，WCF服务端和客户端通过共享元数据(包括服务协定、服务器终结点信息)在两个终结点上建立通道从而进行通信。我们通过手写代码(或配置)的方式为服务端编写了元数据信息，没有借助元数据交换就实现了通信。然而在实际应用中，元数据往往是很多的，而且重复编写元数据的工作也是不值得的，因此必然会用到元数据交换的方式让客户端获取元数据，本篇我们就来进一步了解一下元数据和元数据交换。

1. 元数据是怎样提供的

我们知道，元数据包括了要和服务端进行通信的所有信息，包括服务协定接口、服务端终结点地址、绑定等信息，它指出了客户端应该到何处去寻找服务以及怎样调用服务的一切线索。但是服务端是怎样公布其元数据的呢？

答案是使用WSDL文件，WSDL即Web Service Description Language，Web服务描述语言，它是一个XML文件，在这个文件中按照一定的标准来对Web Service进行描述，他是符合W3C标准的，因为WCF是被设计为供不同平台调用的服务框架，所以客户端可能是非微软平台的，比如Java什么的。因此WCF必须使用WSDL这种国际标准的描述方法来描述服务才能被众多的平台所访问。

2. 元数据交换的过程是怎样的

在WCF服务端的运行时，有一组类库随时待命把服务的元数据输出为WSDL描述提供给请求者，只要有客户端按照服务端约定的方法来请求元数据，服务端立即将服务运行时状态写成WSDL文件提供。客户端得到的实际上就是WSDL文件(还有一些框架描述文件XSD)，客户端拿到文件后再使用自己的方法来解读WSDL，把他翻译成客户端可用的源代码或配置文件，这时客户端就得到了服务的编程模型，通过一些代理类，客户端甚至可以像调用本地对象一样使用WCF服务。

因此整个过程是这样：客户端向服务端请求元数据交换-->服务端运行时将元数据编写成WSDL文件提供-->客户端获得文件-->客户端翻译文件-->客户端根据翻译结果生成本地类代码和配置-->客户端获得服务的本地编程模型。

这就是元数据交换的过程。

3. 获得WSDL

在微软平台中，有两种方法来进行元数据交换，第一是使用服务引用，第二是使用元数据实用工具(svcutil.exe)来进行，我们先学习这个工具。
这个工具可以在Windows SDK中找到，具体位置为 C:\Program Files\Microsoft SDKs\Windows\v6.0\Bin，如果你有VS2010，可以启动VS2010的命令行工具，这样就可以在任何目录下使用这个程序。

我们先看一个例子，就是我们在前几篇中建立的IIS服务HelloWCFService，它被我寄宿在IIS中。

源代码如下(HelloWCF.cs)：

```
[csharp]
1.  using System;
2.  using System.ServiceModel;
3.
4.  namespace LearnWCF
5.  {
6.      [ServiceContract]
7.      public interface IHellloWCF
8.      {
9.          [OperationContract]
10.         string HelloWCF();
11.     }
12.
13.     public class HelloWCFService : IHellloWCF
14.     {
15.         public string HelloWCF()
16.         {
17.             return "Hello WCF!";
18.         }
19.     }
20. }
```

配置文件(web.config)如下：

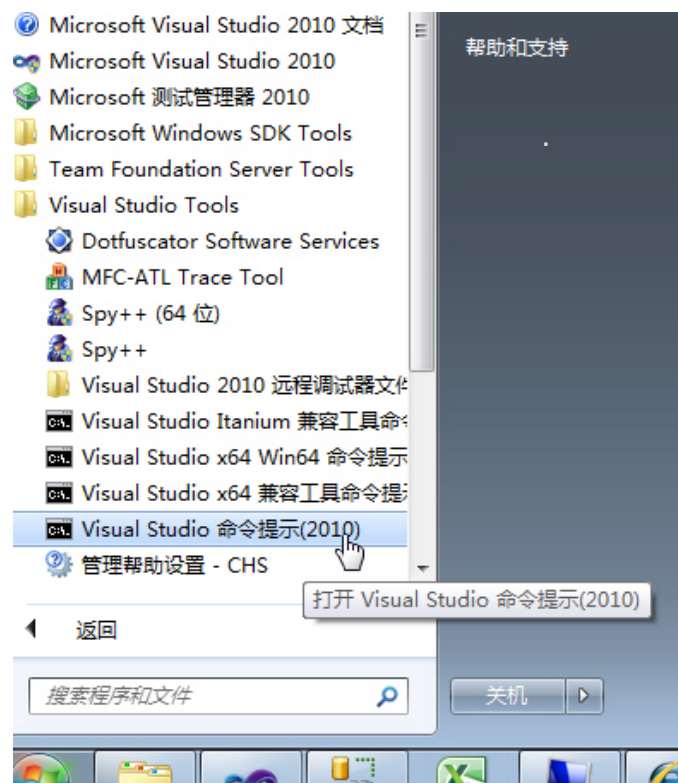
```
[html]
1.  <configuration>
```

```
2.     <system.serviceModel>
3.       <services>
4.         <service name="LearnWCF.HelloWCFService" behaviorConfiguration="metadataExchange">
5.           <endpoint address="" binding="wsHttpBinding" contract="LearnWCF.IHelloWCF"/>
6.           <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
7.         </service>
8.       </services>
9.       <behaviors>
10.        <serviceBehaviors>
11.          <behavior name="metadataExchange">
12.            <serviceMetadata httpGetEnabled="true"/>
13.          </behavior>
14.        </serviceBehaviors>
15.      </behaviors>
16.    </system.serviceModel>
17.  </configuration>
```

在浏览器中输入服务地址会如下图所示：



看到系统提示的那行命令了么？系统在告诉我们如何使用svcutil.exe来获得元数据。我们现在试一下，首先打开VS2010命令行：
开始-->所有程序-->Visual Studio 2010-->Visual Studio Tools-->Visual Studio命令行提示



命令行工具的窗口是这样的：



```
Visual Studio 命令提示(2010)
Setting environment for using Microsoft Visual Studio 2010 x86 tools.

c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC>cd\

c:\>cd wcf

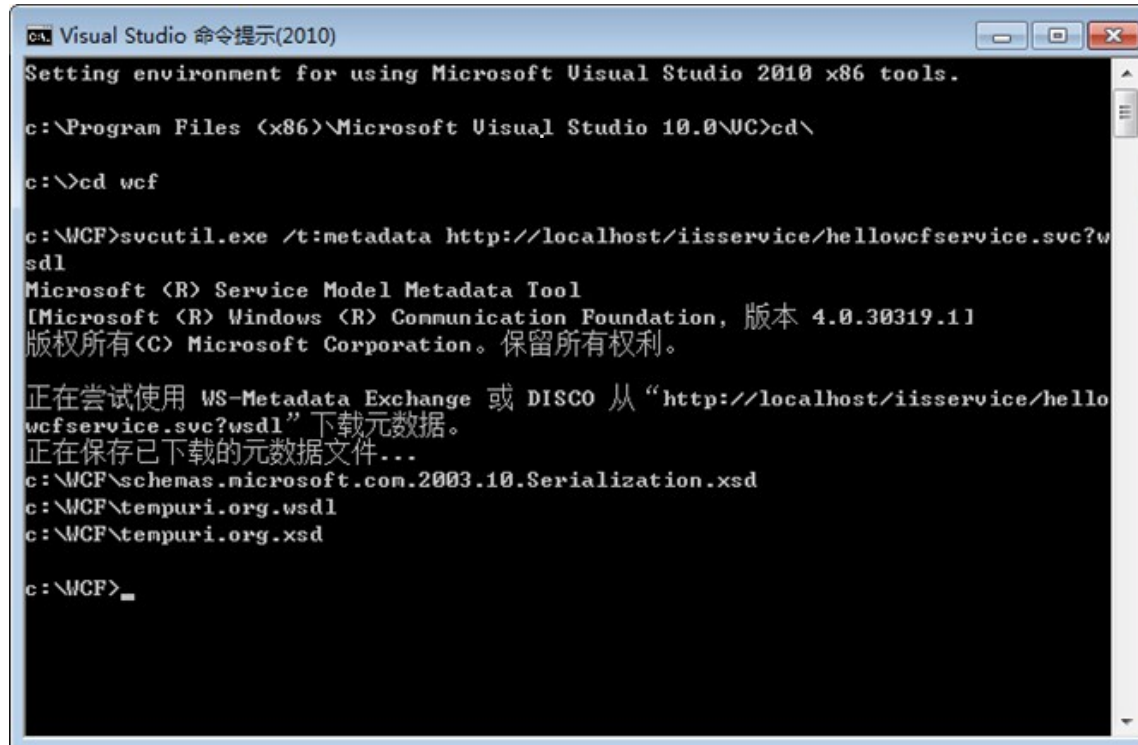
c:\WCF>
```

我们导航到一个目录下准备获得元数据文件。

我们暂时不按照浏览器提供给我们方法做，因为按照那个方法做就把获得WSDL和翻译WSDL为客户端代码合在一起了，我们先获得WSDL元数据文件，看看它是什么样子的。我们按如下的指令做：

- [plain]
1. `svcutil.exe /t:metadata http://localhost/iisservice/hellowcfservice.svc?wsdl`

我们加入了一个参数/t:metadata 表示只输出元数据，不生成代码。命令的执行过程如下：



```
Visual Studio 命令提示(2010)
Setting environment for using Microsoft Visual Studio 2010 x86 tools.

c:\Program Files (x86)\Microsoft Visual Studio 10.0\VC>cd\

c:\>cd wcf

c:\WCF>svcutil.exe /t:metadata http://localhost/iisservice/hellowcfservice.svc?wsdl
Microsoft (R) Service Model Metadata Tool
[Microsoft (R) Windows (R) Communication Foundation, 版本 4.0.30319.11]
版权所有(C) Microsoft Corporation。保留所有权利。

正在尝试使用 WS-Metadata Exchange 或 DISCO 从“http://localhost/iisservice/hello
wcfservice.svc?wsdl”下载元数据。
正在保存已下载的元数据文件...
c:\WCF\schemas.microsoft.com.2003.10.Serialization.xsd
c:\WCF\tempuri.org.wsdl
c:\WCF\tempuri.org.xsd

c:\WCF>
```

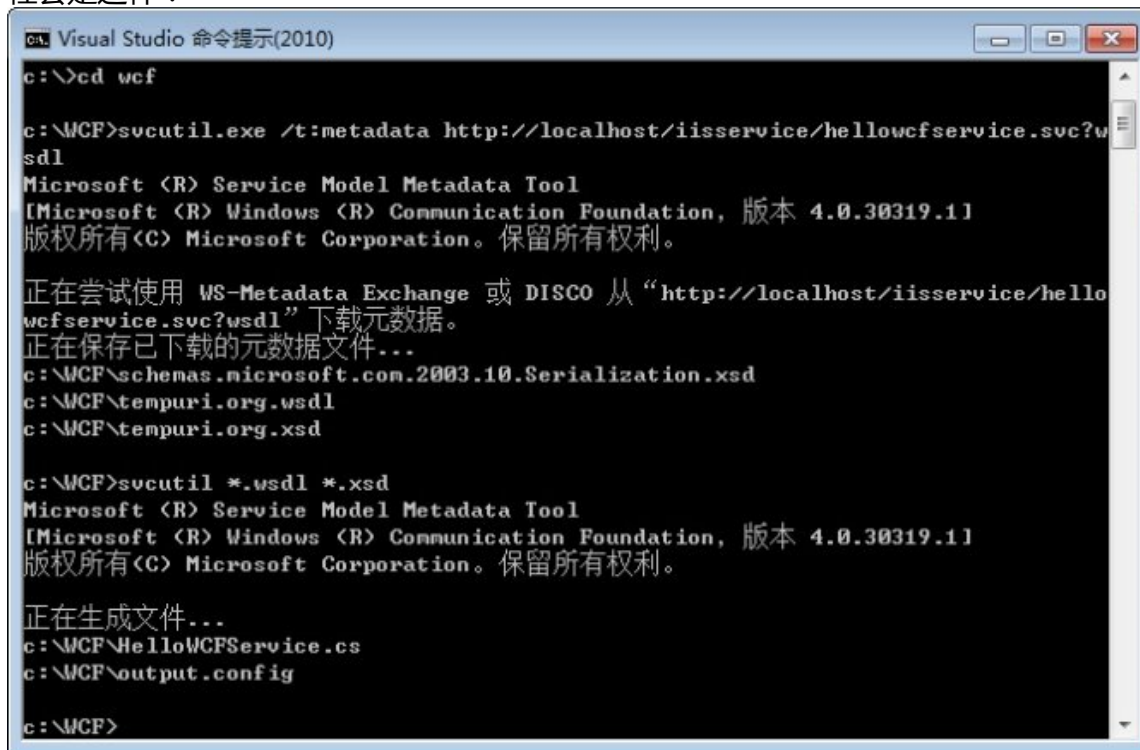
可以看到生成了3个文件，包括两个架构文件和一个WSDL文件，这些就是服务端元数据的描述了，所有的客户端请求到的实际上都是这个文件。WSDL的规范比较多，关于它的内容，我们今后再展开来看，不过简单的打开看一下就能看出一些与服务协定、绑定、操作这些东西相关的地方。

4. 翻译WSDL文件

WSDL是一个XML文件，其实就是个文本文件，客户端必须将其按照自己的平台特点把他翻译成本地代码文件来使用。svcutil当然会提供这个功能。在wsdl文件所在目录下使用如下的命令就可以把WSDL文件翻译成本地代码文件：

```
[plain]
1. svcutil *.wsdl *.xsd
```

顾名思义，就是根据当前目录下的所有的WSDL文件和XSD文件来生成客户端代码文件。过程会是这样：



```
Visual Studio 命令提示(2010)
c:\>cd wcf

c:\WCF>svcutil.exe /t:metadata http://localhost/iiservice/hellowcfservice.svc?wsdl
Microsoft (R) Service Model Metadata Tool
[Microsoft (R) Windows (R) Communication Foundation, 版本 4.0.30319.11]
版权所有(C) Microsoft Corporation。保留所有权利。

正在尝试使用 WS-Metadata Exchange 或 DISCO 从“http://localhost/iiservice/hellowcfservice.svc?wsdl”下载元数据。
正在保存已下载的元数据文件...
c:\WCF\schemas.microsoft.com.2003.10.Serialization.xsd
c:\WCF\tempuri.org.wsdl
c:\WCF\tempuri.org.xsd

c:\WCF>svcutil *.wsdl *.xsd
Microsoft (R) Service Model Metadata Tool
[Microsoft (R) Windows (R) Communication Foundation, 版本 4.0.30319.11]
版权所有(C) Microsoft Corporation。保留所有权利。

正在生成文件...
c:\WCF\HelloWCFService.cs
c:\WCF\output.config
c:\WCF>
```

可以看到，生成了一个cs文件和一个配置文件，这些就是根据WSDL文件翻译成的客户端代码文件了。打开来看看，一定不陌生，就是使用ClientBase<>来生成一个客户端代理类并把终结点的信息配置在了.config文件里。把这两个文件包含在客户端的项目中并把output.config改成app.config就可以了。

4. 更好地使用元数据交换工具

之前我们了解了使用svcutl.exe来获取WSDL并翻译成客户端代码的过程。实际上这两步可以合二为一。直接执行下面的命令可以直接获得客户端文件：

```
[plain]
1. svcutl.exe http://localhost/iisservice/hellowcfservice.svc?wsdl
```

这样它就不会生成WSDL而直接生成客户端文件了。
不过按照这样的方式生成的文件可能不太符合我们的要求，我们可以加上一些参数来指定我们输出的文件名：

```
[plain]
1. svctuil.exe /out:ClientProxy.cs /config:app.config http://localhost/iisservice/hellowcfservice.svc?wsdl
```

这样输出的文件我们就可以直接包含在客户端项目中使用了。

5. 使用服务引用

其实使用服务引用跟使用svcutl.exe生成的客户端模型是一样的，不过服务引用保留了WSDL文件(以及一些相关的七七八八的文件)，没有svcutl.exe来得那么清爽，但是它跟VS2010集成，使用起来很简单，而且当服务发生变化时，只需要右击服务引用选择更新服务就可以重新下载WSDL了。

6. 展开一点点

作为服务端，公开元数据是需要配置的，不同的配置会导致元数据公开的方式不同。

我们要记住，WCF服务端公开元数据必须具备两个条件：

(1) 为服务添加ServiceMetadata行为。

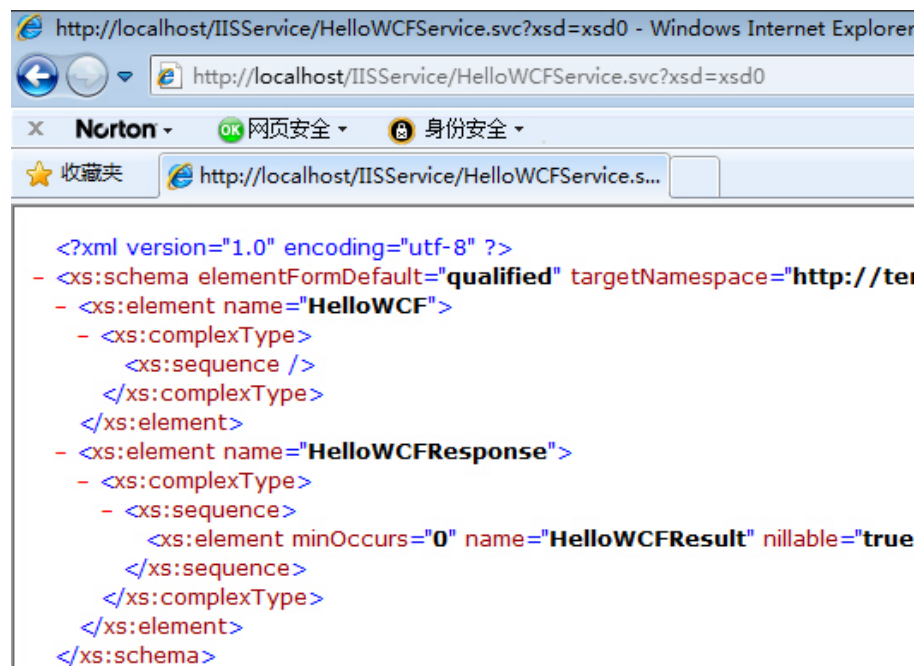
(2) 打开元数据交换终结点。

二者缺一不可。

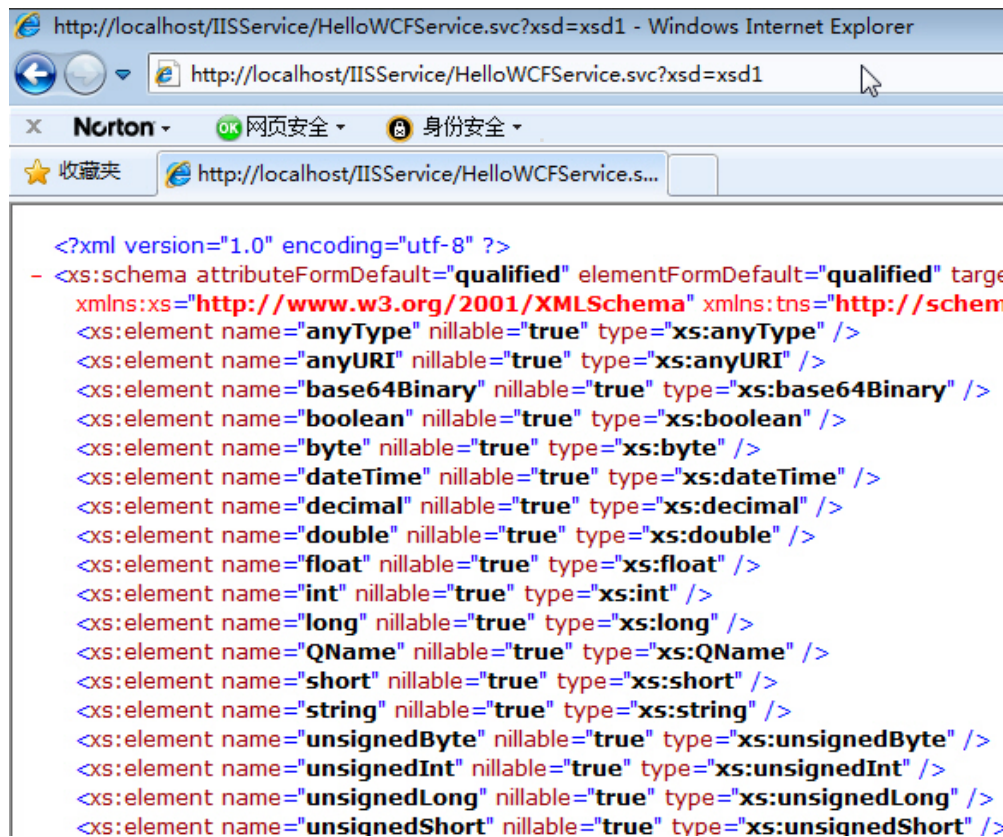
WCF的公开元数据的手段主要有两种：

第一种：通过HTTP GET方法。

这就是在前文中我们看到的方法，我们可以使用HTTP Get的方法来获得WSDL文件即在服务地址.svc后面跟上?wsdl的方法直接请求到WSDL文件。我们可以直接在浏览器中输入服务端地址.svc?wsdl，浏览器就直接获得了WSDL文件并为我们显示出来了。



```
<?xml version="1.0" encoding="utf-8" ?>
- <xs:schema elementFormDefault="qualified" targetNamespace="http://tei
- <xs:element name="HelloWCF">
  - <xs:complexType>
    <xs:sequence />
  </xs:complexType>
</xs:element>
- <xs:element name="HelloWCFResponse">
  - <xs:complexType>
    - <xs:sequence>
      <xs:element minOccurs="0" name="HelloWCFResult" nillable="true
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```



如果想采用这种元数据公开方式，必须配置服务的ServiceMetadata行为，并指定 `httpGetEnabled = "true"`，而元数据公开终结点不必配置，系统会自动配置一个，配置文件的写法如下：

```
[html]
1. <configuration>
2.   <system.serviceModel>
3.     <services>
4.       <service name="LearnWCF.HelloWCFService" behaviorConfiguration="metadataExchange">
5.         <endpoint address="" binding="wsHttpBinding" contract="LearnWCF.IHelloWCF"/>
6.       </service>
7.     </services>
```

```
8.     <behaviors>
9.         <serviceBehaviors>
10.             <behavior name="metadataExchange">
11.                 <serviceMetadata httpGetEnabled="true"/>
12.             </behavior>
13.         </serviceBehaviors>
14.     </behaviors>
15. </system.serviceModel>
16. </configuration>
```

在这种配置下，访问元数据的方法是访问下面的地址：

[plain]

```
1. http://localhost/iisservice/hellowcfService.svc?wsdl
```

第二种：通过MEX元数据交换终结点。

在这种方式下，我们首先要保证服务拥有ServiceMetadata行为，但是httpGetEnabled可以不必为true。此外我们还需要为服务显式地添加一个终结点，这个终结点的地址、绑定和协定都是指定的我们不能更改

[html]

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

配置文件的写法如下：

```
[html]
1. <configuration>
2.     <system.serviceModel>
3.         <services>
4.             <service name="LearnWCF.HelloWCFService" behaviorConfiguration="metadataExchange">
5.                 <endpoint address="" binding="wsHttpBinding" contract="LearnWCF.IHelloWCF"/>
6.                 <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
7.             </service>
8.         </services>
9.         <behaviors>
10.             <serviceBehaviors>
11.                 <behavior name="metadataExchange">
```

```
12.         <serviceMetadata />
13.     </behavior>
14. </serviceBehaviors>
15. </behaviors>
16. </system.serviceModel>
17. </configuration>
```

如果按这种配置，我们必须按照如下地址来访问公开的元数据

```
[plain]
1. http://localhost/iisservice/hellowcfservice.svc/mex
```

注意，由于没有开启HTTP GET，我们不能在浏览器中直接输入这个地址来获取WSDL了(会提示400错误)，我们必须通过svcutil.exe或添加服务引用的方式来访问。

使用svcutil.exe或服务引用的时候可以不关心元数据公开方式是HTTP GET还是Mex，他们会自动寻找到合适的方式，只需要把服务的svc文件地址输入就可以了，但是我们应该知道，这两种元数据公开的方式是有区别的。

6. 总结

通过今天的学习，我们进一步了解了WCF元数据的和元数据交换的原理。虽然我们在实际工程中都会并且应该使用元数据交换工具来帮助提高效率，但是这背后发生的所有环节也是我们应该掌握的。

相关资源

MSDN关于Svcutil.exe用法的文档

<http://msdn.microsoft.com/zh-cn/library/aa347733.aspx>

(<http://msdn.microsoft.com/zh-cn/library/aa347733.aspx>)