　转眼wcf技术已经出现很多年了，也在.net界混的风生水起，同时.net也是一个高度封装的框架，作为在wcf食物链最顶端的我们所能做的任务已经简单的不能再简单了，

再简单的话马路上的大妈也能写wcf了，好了，wcf最基本的概念我们放在后面慢慢分析，下面我们来看看神奇的3个binding如何KO我们实际场景中的80%的业务场景。

一：basicHttpBinding

　　作为入门第一篇，也就不深入谈谈basic中的信道栈中那些啥东西了，你只需要知道有ABC三个要素，注意不是姨妈巾哦，如果需要详细了解，可以观赏我[以前的系列](http://www.cnblogs.com/huangxincheng/archive/2011/10/23/2221845.html" \t "_blank)。在

这里我就不多说了，太简单的东西没意思，先看个例子简单感受了，你只需知道的是basic走的是http协议就好了，传输消息为soap。

1. 契约

[复制代码](javascript:void(0);)

1 using System.Runtime.Serialization;

2 using System.ServiceModel;

3

4 namespace MyService

5 {

6 [ServiceContract]

7 public interface IHomeService

8 {

9 [OperationContract]

10 int GetLength(string name);

11 }

12 }

[复制代码](javascript:void(0);)

2. 实现类

[复制代码](javascript:void(0);)

1 using System;

2 using System.Messaging;

3 using System.Threading;

4

5 namespace MyService

6 {

7 public class HomeService : IHomeService

8 {

9 public int GetLength(string name)

10 {

11 return name.Length;

12 }

13 }

14 }

[复制代码](javascript:void(0);)

3. 服务启动

[复制代码](javascript:void(0);)

1 using System;

2 using System.ServiceModel;

3

4 namespace MyService

5 {

6 class Program

7 {

8 static void Main(string[] args)

9 {

10 using (ServiceHost host = new ServiceHost(typeof(HomeService)))

11 {

12 try

13 {

14 host.Open();

15

16 Console.WriteLine("服务开启！");

17

18 Console.Read();

19 }

20 catch (Exception e)

21 {

22 Console.WriteLine(e.Message);

23 }

24 }

25 }

26 }

27 }

[复制代码](javascript:void(0);)

4. 配置config文件

[复制代码](javascript:void(0);)

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

<configuration>

<system.serviceModel>

<bindings>

<netTcpBinding>

<binding name="IHomeServiceBinding" />

</netTcpBinding>

</bindings>

<behaviors>

<serviceBehaviors>

<behavior name="">

<serviceMetadata httpGetEnabled="true" />

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

<services>

<service name="MyService.HomeService">

<endpoint address="http://127.0.0.1:1920/HomeService" binding="basicHttpBinding" contract="MyService.IHomeService">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

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

<host>

<baseAddresses>

<add baseAddress="http://127.0.0.1:1920"/>

</baseAddresses>

</host>

</service>

</services>

</system.serviceModel>

</configuration>

[复制代码](javascript:void(0);)

5. 然后通过 servicehost 启动服务端

[复制代码](javascript:void(0);)

using System;

using System.ServiceModel;

namespace MyService

{

class Program

{

static void Main(string[] args)

{

using (ServiceHost host = new ServiceHost(typeof(HomeService)))

{

try

{

host.Open();

Console.WriteLine("服务开启！");

Console.Read();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

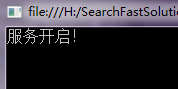
}

}

}

}

[复制代码](javascript:void(0);)



好了，到现在为止，服务端全部开启完毕，接下来我们通过“添加服务引用”，来添加对客户端的引用

[复制代码](javascript:void(0);)

1 using System;

2

3 namespace ConsoleApplication1

4 {

5 class Program

6 {

7 static void Main(string[] args)

8 {

9 HomeServiceReference.HomeServiceClient client = new HomeServiceReference.HomeServiceClient();

10

11 var s = client.GetLength("12345");

12

13 Console.WriteLine("长度为:{0}", s);

14

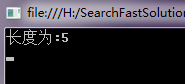
15 Console.Read();

16 }

17 }

18 }

[复制代码](javascript:void(0);)



麻蛋，就这么简单，是的，就这样简单的五步，基于http的通信就这样被不小心的完成了，真不好意思。

二：netTcpBinding

　　有了basic的代码，现在我们要改成tcp通信，这会通信走的是字节流，很简单，改一下服务端的config文件就好了，大家也知道这种性能要比basic好。

[复制代码](javascript:void(0);)

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

<configuration>

<system.serviceModel>

<behaviors>

<serviceBehaviors>

<behavior name="mxbehavior">

<serviceMetadata httpGetEnabled="true" />

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

<services>

<service name="MyService.HomeService" behaviorConfiguration="mxbehavior">

<endpoint address="net.tcp://localhost:19200/HomeService" binding="netTcpBinding" contract="MyService.IHomeService">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

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

<host>

<baseAddresses>

<add baseAddress="http://localhost:1920/HomeService"/>

</baseAddresses>

</host>

</service>

</services>

</system.serviceModel>

</configuration>

[复制代码](javascript:void(0);)

三：netMsmqBinding

　　msmq这个玩意，我想大家都清楚，一个物理上的文件，好处呢，你也明白，就是client和service的所有通信都要经过它的手，这样任何一方出了问题，只要

它在就没问题了。同样我们把tcp改成msmq也是非常简单的，不过要注意，msmqbinding中是不可以让契约方法有返回值的。所以我们加上isoneway就好了。

[复制代码](javascript:void(0);)

using System.Runtime.Serialization;

using System.ServiceModel;

namespace MyService

{

[ServiceContract]

public interface IHomeService

{

[OperationContract(IsOneWay = true)]

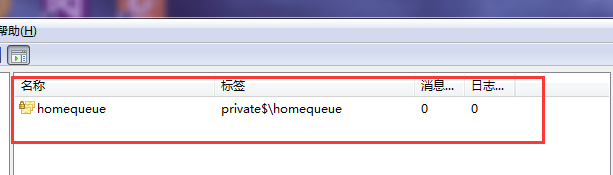
void GetLength(string name);

}

}

[复制代码](javascript:void(0);)

然后我在mmc上新建一个消息队列，如下：



然后我们再改动以下配置文件

[复制代码](javascript:void(0);)

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

<configuration>

<system.serviceModel>

<behaviors>

<serviceBehaviors>

<behavior name="mxbehavior">

<serviceMetadata httpGetEnabled="true" />

<serviceDebug includeExceptionDetailInFaults="true" />

</behavior>

</serviceBehaviors>

</behaviors>

<bindings>

<netMsmqBinding>

<binding name="msmqbinding">

<security mode="None"/>

</binding>

</netMsmqBinding>

</bindings>

<services>

<service name="MyService.HomeService" behaviorConfiguration="mxbehavior">

<endpoint address="net.msmq://localhost/private/homequeue" binding="netMsmqBinding"

contract="MyService.IHomeService" bindingConfiguration="msmqbinding">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

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

<host>

<baseAddresses>

<add baseAddress="http://localhost:19200/HomeService"/>

</baseAddresses>

</host>

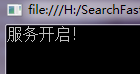
</service>

</services>

</system.serviceModel>

</configuration>

[复制代码](javascript:void(0);)



纵观上面的三种binding，配置起来何其简单，底层的各种通讯协议貌似对我来说都是透明的，其实呢？？？wcf在底层做了何其多的事情，而我却没有挖掘。。。

这对码农里说也是一种悲哀啊。。。出了问题就只能祷告上天。。。下一篇我会开始深入剖析。