

## [老老实实学WCF] 第七篇 会话

### 老老实实学WCF

#### 第七篇 会话

通过前几篇的学习，我们已经掌握了WCF的最基本的编程模型，我们已经可以写出完整的通信了。从这篇开始我们要深入地了解这个模型的高级特性，这些特性用来保证我们的程序运行的高效、稳定和安全。

首先我们来学习会话。

#### 1. 什么是会话

会话是通信双方进行通信的一个时间片、一个语境或者说一个上下文，在这个特定的环境中，通信的双方是彼此认识的，就像两个人在聊天，他们都很清楚谁在聆听自己讲话，也很清楚对方讲的话是给自己听的，简单的说就是通信双方是可以记住彼此的。

一旦会话结束了，通信双方就忘记了彼此，即使他们再次建立会话，他们也不会记得他们上次会话的内容，也就是他们不记得他们曾经见过面。

这在我们现实世界中或许很难想象，但是在通信的世界里就是这样的。服务端不可能记住每个跟他通信的人，他只能在一段时间内(会话)记住一个人。

这个特性是很有用的，有些逻辑需要客户端和服务端通信多次才能完成，在这个期间双方需要记住彼此，而且会话也是很多其他特性实现的基础，例如双工通信。

#### 2. 如何建立会话

那么我们要想建立一个会话通信，应该具备怎样的条件呢？

- (1) 需要支持会话的绑定。绑定描述了双方的通信方式，不同的绑定对会话的支持是不同的，比如basicHttpBinding是不支持会话的，而wsHttpBinding就是支持的。要建立会话通信，这个通信必须首先使用支持会话的绑定。
- (2) 让服务协定支持会话，服务协定实际上就是通信的通道(见第四、五篇)，让服务协定支持会话，那么就可以在这个通信通道上支持会话了。

选择支持会话的绑定我们知道怎么做，可如何让服务协定支持会话呢？要用到在修饰服务协定的ServiceContract属性，我们知道被这个属性修饰的接口是一个服务协定，其实这个属性也拥有属性，其中一个属性叫做SessionMode。这是一个枚举，我们通过设置这个枚举的值来配置服务协定是否支持会话。例如：

```
[csharp]
1. [ServiceContract(SessionMode = SessionMode.Required)]
2.     public interface IHelloWCF
3.     {
4.         [OperationContract]
5.         string HelloWCF();
6.     }
```

这段代码中，我把SessionMode设置为了Required，这表示调用这个服务协定的客户端必须使用会话。

SessionMode有三个可能的值：

- 1) Allowed：这是默认值，表示这个服务协定是允许会话的，客户端可以选择用会话连接，也可以选择不用会话连接。
- 2) Required：表示服务协定要求客户端连接必须使用会话。
- 3) NotAllowed：表示服务协定不允许使用会话连接。

这些配置需要搭配其他的配置才能起到实际意义，比如服务实例模式，服务端和客户端调用模式等等，等我们了解到这些特性的时候再展开，现在我们只需要知道，前两种配置是支持会话的，第三种是不支持的。

### 3. 一个简单的例子

我们通过一个简单的例子来看看允许会话与不允许的区别，我修改了前几篇中寄存在IIS中的服务，代码如下：

```
[csharp]
1. using System;
2. using System.ServiceModel;
3.
4. namespace LearnWCF
5. {
6.     [ServiceContract(SessionMode = SessionMode.Allowed)]
7.     public interface IHelloWCF
8.     {
9.         [OperationContract]
10.         string HelloWCF();
11.     }
12.
13.     public class HelloWCFService : IHelloWCF
14.     {
15.         private int _Counter;
16.         public string HelloWCF()
17.         {
18.             _Counter++;
19.             return "Hello, you called " + _Counter.ToString() + " time(s)";
20.         }
21.     }
22. }
```

首先我们把服务协定的会话模式设置为允许会话(Allowed)，在服务实现中，我为服务实现类定义了一个计数器成员，每次调用都会将这个计数器加一，然后返回一句话告诉客户端调用了多少次。

服务端的配置文件如下：

```
[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>
```

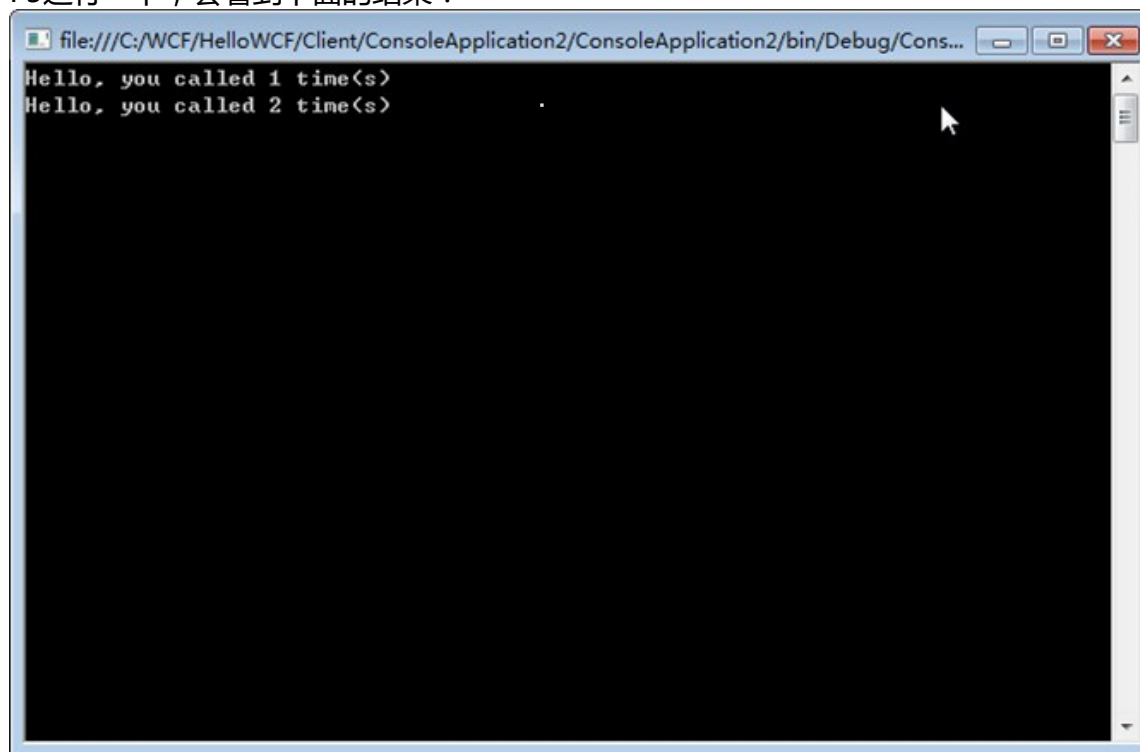
在这里我配置了支持会话的wsHttpBinding。

客户端的调用代码如下：

```
[csharp]
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5.
6. namespace ConsoleApplication2
7. {
8.     class Program
9.     {
10.         static void Main(string[] args)
11.         {
12.             ConsoleApplication2.ServiceReference1.HelloWCFClient client = new ServiceReference1.HelloWCFClient();
13.             Console.WriteLine(client.HelloWCF());
14.             Console.WriteLine(client.HelloWCF());
15.             client.Close();
16.             Console.Read();
17.         }
18.     }
19. }
```

就是连续调用两次服务端的方法并输出结果。

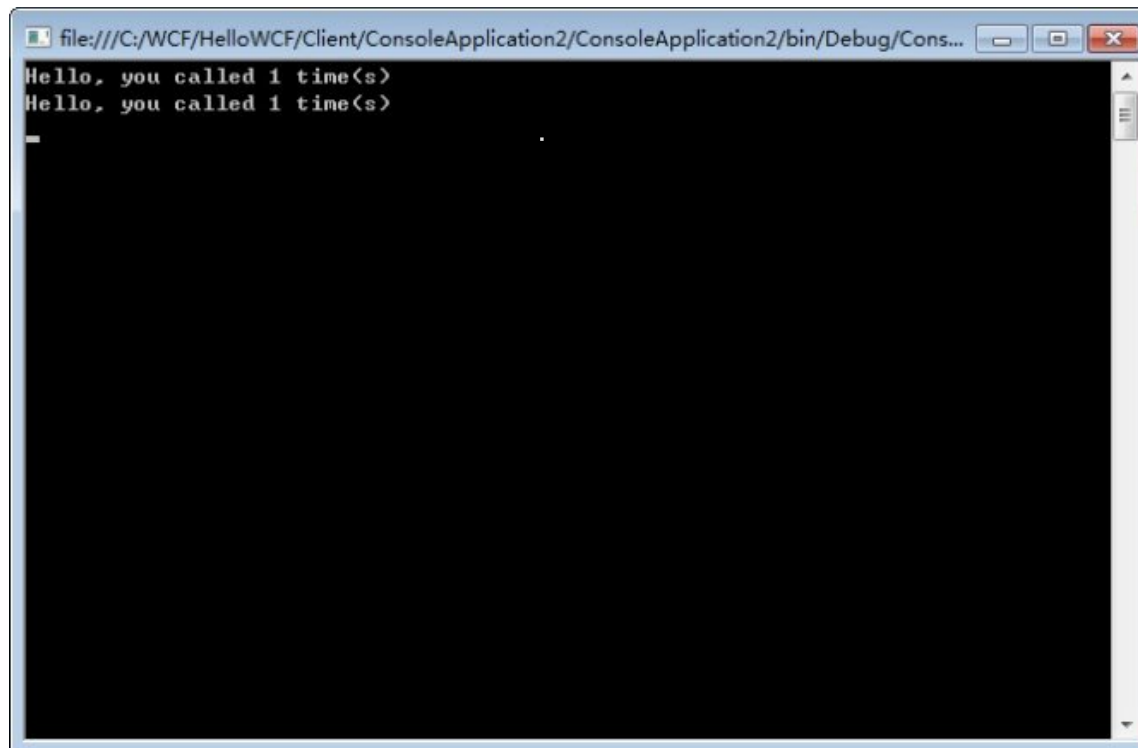
F5运行一下，会看到下面的结果：



```
file:///C:/WCF/HelloWCF/Client/ConsoleApplication2/ConsoleApplication2/bin/Debug/Cons...  
Hello, you called 1 time(s)  
Hello, you called 2 time(s)
```

我们看到提示调用了两次，也就是说服务端记住了客户端，当他第二次调用的时候将计数器加一，就返回了调用两次。当然这个局面的形成还受到实例上下文模式为PerSession的影响，我们后面会展开，总之服务协议支持会话，才出现了这个局面。

如果我们把SessionMode改成NotAllowed，其他不改动，结果就会是下面的样子：



结果两次都是1，说明服务器在第二次受到调用的时候已经忘记了之前那个客户端，他又分配了一个新的计数器给这个客户端，所以计数就总是1了。

其实这个例子是很粗糙的，这里面还有些其他的影响因素，我们就是通过这个例子来看看 SessionMode 的一方面影响。

#### 4. 总结

这一篇的内容比较少，我们应该记住一些要点，在以后接触更多特性的时候才不会混淆。

- (1) 是否支持会话首先取决于选择的绑定。
- (2) 是否支持会话通过配置服务协定的ServiceContract属性的SessionMode属性实现的。