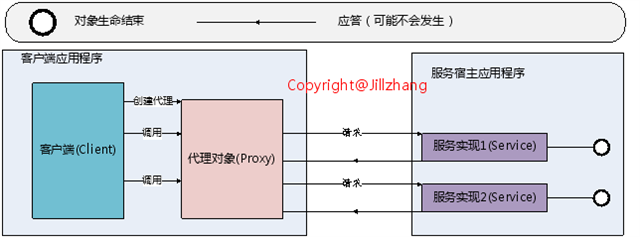
学习WCF的目的在于使用WCF建立服务，使Sivlerlight程序能够查询和修改Oracle数据的内容，数据库的操作不可避免要涉及数据库事务（Transaction），而基于数据库事务的操作则要求对于某特定客户端程序，开始事务（Transaction Begin）后，事务对象保持不变，才能保证Commit或Rollback操作的成功，为此，查询了相关资料，好像可以通过WCF的服务实例（Service Instance）来控制。

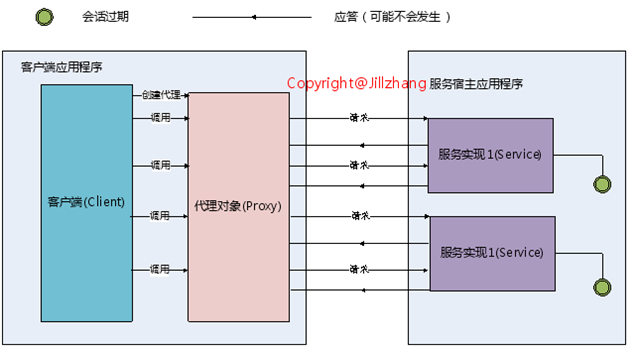
**一、基本介绍**

尝试了新建一个基本的WCF实例后，开始学习WCF的服务实例，服务实例主要控制WCF客户端与服务端交互时服务端的行为，有三类实例模式：

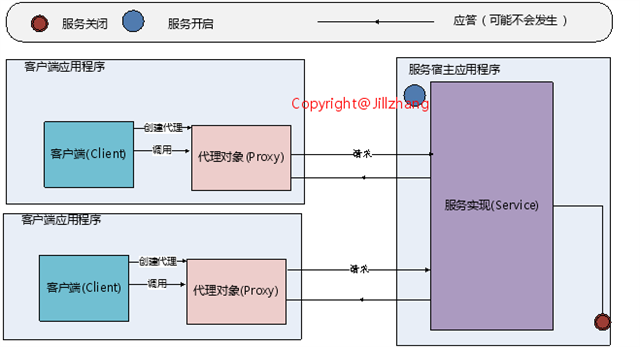
1：单调服务（Per-Call Service）：每次的客户端请求分配一个新的服务实例。



2：会话服务（Sessionful Service）：则为每次客户端连接分配一个服务实例



3：单例服务（Singleton Service）：所有的客户端会为所有的连接和激活对象共享一个相同的服务实例。类似于Net Remoting的SingleTon模式



**二、示例代码**

通过一个例子来说明几种服务的区别：

1.契约（CONTRACT）

包括三个函数：SetValue、GetValue、GetSid

[ServiceContract]

1. public interface IService1
2. {
4. [OperationContract]
5. string SetValue(int value);
7. [OperationContract]
8. string GetSid();
10. [OperationContract]
11. string GetValue();
12. // TODO: Add your service operations here
13. }

服务功能实现（ServiceBehavior）

（1）：构造函数：打印当前连接的SessionId

（2）：SetValue函数：为类中的intvalue变量赋值

（3）：GetValue函数：获取类中的intvalue变量的值

（4）：GetSid函数：获取当前连接的SessionId

（5）：Dispose函数：显示断开连接的信息，以及该连接的SessionId

public class Service1 : IService1,IDisposable

1. {
2. private int intvalue;
3. private string currentsid;
5. public Service1()
6. {
7. try
8. {
9. currentsid = OperationContext.Current.SessionId;
10. }
11. catch (Exception ex)
12. {
13. Console.WriteLine("get current sid failed,operatincontext.currect is null?" +(OperationContext.Current == null) +"  ");
14. }
16. Console.WriteLine("Service1 connected:" + currentsid);
17. }

20. public string SetValue(int value)
21. {
22. intvalue = value;
23. return string.Format("You entered: {0}", value);
24. }
26. public string GetSid()
27. {
28. return currentsid;
29. }
31. public string GetValue()
32. {
33. return intvalue.ToString();
34. }
35. #region IDisposable Members
37. public void Dispose()
38. {
40. Console.WriteLine("Service1 disconnected:" + currentsid);
42. }
43. #endregion
45. }

2.寄宿（Host）：

ServiceReference1.Service1Client sc1 = new ServiceReference1.Service1Client();

1. string a;
2. int tmpInt;
3. while ((a = Console.ReadLine()) != "exit")
4. {
5. if (int.TryParse(a,out tmpInt))
6. Console.WriteLine(sc1.SetValue(tmpInt));
7. else if (a == "get")
8. Console.WriteLine(sc1.GetValue());
9. else if (a == "sid")
10. Console.WriteLine(sc1.GetSid());
11. }
13. ((IDisposable)sc1).Dispose();

Host对应的app.config:

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

1. - <configuration>
2. - <system.serviceModel>
3. <diagnostics performanceCounters="All" />
4. - <behaviors>
5. - <serviceBehaviors>
6. - <behavior name="NewBehavior0">
7. <serviceMetadata httpGetEnabled="true" httpGetUrl="http://127.0.0.1:8585/is/metadata" />
8. </behavior>
9. </serviceBehaviors>
10. </behaviors>
11. - <services>
12. - <service behaviorConfiguration="NewBehavior0" name="WcfService1.Service1">
13. <endpoint address="" binding="wsDualHttpBinding" bindingConfiguration="" name="ep1" contract="WcfService1.IService1" />
14. - <host>
15. - <baseAddresses>
16. <add baseAddress="http://127.0.0.1:8585/is" />
17. </baseAddresses>
18. </host>
19. </service>
20. </services>
21. </system.serviceModel>
22. </configuration>

这里使用了**wsDualHttpBinding**的 binging，是因为并不是所有的Binding都支持 Session ，对Per-Call和Singleton而言，binding影响不大。对PerSession，就必须使用特定的Binding, 才可以,否则最终指向的是PerCall，具体的可再进行相关查询

3.客户端（Client）

功能：

当用户在命令行输入一个数字时，则调用SetValue函数，将该数字赋给WCF的intvalue值并打印提示信息，

当用户在命令行中输入get时，调用GetValue函数获取intvalue值并打印出来

当用户在命令行中输入sid时，调用GetSid函数获取当前的SessionId并打印出来

当用户在命令行中输入exit时，断开连接并退出函数

ServiceReference1.Service1Client sc1 = new ServiceReference1.Service1Client();

1. string a;
2. int tmpInt;
3. while ((a = Console.ReadLine()) != "exit")
4. {
5. if (int.TryParse(a,out tmpInt))
6. Console.WriteLine(sc1.SetValue(tmpInt));
7. else if (a == "get")
8. Console.WriteLine(sc1.GetValue());
9. else if (a == "sid")
10. Console.WriteLine(sc1.GetSid());
11. }
13. ((IDisposable)sc1).Dispose();

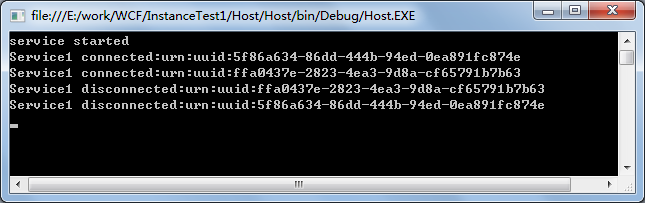
**三、三种实例模式的测试**

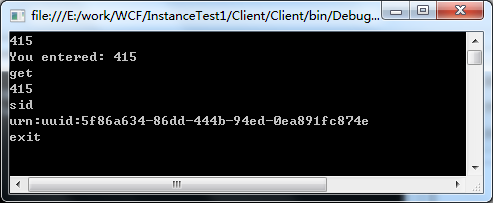
1：persession模式

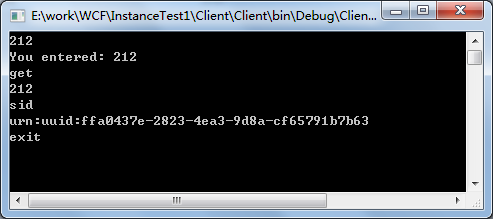
WCF默认即为该模式，不用对代码进行修改，直接运行即可。

打开多个Client命令行，发现每打开一个命令行，Host的命令行即会打印出一个SessionID

在各Client命令行中，执行setvalue,getvalue,getsid的操作，在Session内部都是一致的，且不与其它Session相关，相关截图如下：







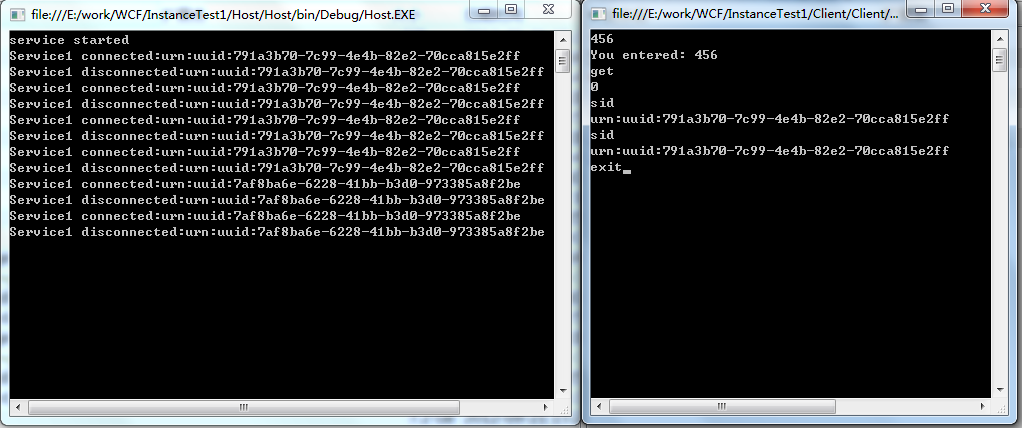
2：percall模式

需要在Service1的类声明前加上一句声明，加完后如下：

**[csharp]** [view plaincopy](http://blog.csdn.net/rrrrssss00/article/details/7992338)

1. [ServiceBehavior(InstanceContextMode = InstanceContextMode.PerCall)]
2. public class Service1 : IService1,IDisposable

该模式下，每一次任何操作均会打开一个新的Session，但SessionID相同，操作完成后马上关闭该Session，且intvalue的值也不会保存，如果打开多个Client，其SessoinID也相同，截图如下：



3.Single模式

需要在Service1的类声明前加上一句声明，加完后如下：

**[csharp]** [view plaincopy](http://blog.csdn.net/rrrrssss00/article/details/7992338)

1. <span style="font-size:18px;"> [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
2. public class Service1 : IService1,IDisposable</span>

该模式下，无论打开多少个Client，服务端只有一个Session，多个Client的intvalue值是共享的（也就是说ClientA将该值设为123，那ClientB获取的值就是123）

此外，在这个模式下，没有SessionID的概念，在尝试获取这个ID时，会返回空值，截图如下：

