请求/响应模式

单工模式

双工模式

* **请求/响应**

请求/响应通信是指客户端向服务端发送消息后，服务端会向客户端发送响应。这也意味着在接收到服务的响应以前，客户端不会继续执行操作。

[OperationContract]

string getOrders(DateTime dt)；

1. **请求/响应模式是默认的消息模式。**
2. **返回值是void，也属于请求/响应消息交换。**
3. **除非客户端异步调用操作，否则客户端将停止处理，直到收到返回消息，即使该消息正常情况下为空时也是如此。**

* 缺点

– 如果执行操作需要很长的时间，则会降低客户端性能和响应能力。

* 优点

– 响应消息中可返回SOAP 错误，这表明可能在通信或处理中发生了一些与服务有关的错误状况

**单工模式**

**单向通信应是指通信只在一个方向进行，即从客户端流向服务。服务不会发送响应，而客户端也不会期望会有响应。这种情况下，客户端发送消息，然后继续执行。**

单向操作是客户端调用操作并在WCF 将消息写入网络后继续进行处理的操作。通常这意味着，除非在出站消息中发送的数据极其庞大，否则客户端几乎立即继续运行（除非发送数据时出错）。此种类型的消息交换模式支持从客户端到服务应用程序的类似于事件的行为。

**若要为返回void 的操作指定单向消息交换，请将IsOneWay 属性设置为true，默认为false.**

[OperationContract(IsOneWay=true)]

void insertEmp(Employee emp);

**双工模式**

**双工模式的特点是，无论使用单向消息发送还是请求/答复消息发送方式，服务和客户端均能够独立地向对方发送消息。对于必须直接与客户端通信或向消息交换的任意一方提供异步体验（包括类似于事件的行为）的服务来说，这种双向通信形式非常有用。**

– 若要设计双工协定，还必须设计回调协定，并将该回调协定的类型分配给标记服务协定的ServiceContractAttribute 属性(attribute)的CallbackContract 属性(property)。

– 若要实现双工模式，您必须创建第二个接口，该接口包含在客户端调用的方法声明

**[ServiceContract(SessionMode=SessionMode.Required,**

**CallbackContract=typeof(IServiceDuplexCallback))]**

**public interface IServiceClass**

**{**

**[OperationContract(IsOneWay = true)]**

**void AddNumber();**

**}**

**public interface IServiceDuplexCallback**

**{**

**[OperationContract(IsOneWay = true)]**

**void Calculate(double result);**

**}**

示例代码：

1请求响应模式：

1. 创建WCF服务库

接口Iservice

namespace mywcfService

{

// 注意: 如果更改此处的接口名称“IService1”，也必须更新 App.config 中对“IService1”的引用。

[ServiceContract]

public interface IService1

{

[OperationContract]

string getServerTime();

}

}

Iservice的实现

namespace mywcfService

{

// 注意: 如果更改此处的类名“IService1”，也必须更新 App.config 中对“IService1”的引用。

public class Service1 : IService1

{

#region IService1 成员

public string getServerTime()

{

System.Threading.Thread.Sleep(5000);

return DateTime.Now.ToString();

}

#endregion

}

}

1. 新建客户端窗体应用程序，并从解决方案中添加服务，后添加按钮

//请求响应模式

private void button1\_Click(object sender, EventArgs e)

{

myService.Service1Client client = new myClient.myService.Service1Client();

string t = client.getServerTime();

}

单工通信：

接口：

[OperationContract(IsOneWay = true)]

void setServerTime();

实现：

public void setServerTime()

{

System.Threading.Thread.Sleep(5000);

}

客户端

//单工

private void button2\_Click(object sender, EventArgs e)

{

myService.Service1Client client = new myClient.myService.Service1Client();

client.setServerTime();

}

双工通信示例：

1. 创建WCF服务库，Iservice1接口处代码：

[ServiceContract(SessionMode=SessionMode.Required,CallbackContract=typeof(IServiceDuplexCallback))]

public interface IService1

{

[OperationContract(IsOneWay = true)]

void AddNumber(int x,int y);

[OperationContract(IsOneWay = true)]

void SubtractNumber(int x, int y);

}

public interface IServiceDuplexCallback

{

[OperationContract(IsOneWay = true)]

void Calculate(int result);

}

1. Iservice1对应实现的代码：

[ServiceBehavior(InstanceContextMode=InstanceContextMode.PerSession)]

public class Service1 : IService1

{

//声明回调对象。

IServiceDuplexCallback callback

{

//得到回调的通道

get { return OperationContext.Current.GetCallbackChannel<IServiceDuplexCallback>(); }

}

#region IService1 成员

public void AddNumber(int x, int y)

{

int z = x + y;

//服务端执行完这个addnumber方法后，通过callback.Calculate()将结果返回

callback.Calculate(z);

}

public void SubtractNumber(int x, int y)

{

int z = x - y;

//服务端执行完这个addnumber方法后，通过callback.Calculate()将结果返回

callback.Calculate(z);

}

#endregion

}

1. WCF服务库对应的app.config配置文件需要修改一下对应的绑定为wsDualHttpBinding
2. 客户端，添加服务引用。后添加CallbackHander.cs类，实现回调接口方法

public class CallbackHander : myService.IService1Callback

{

#region IService1Callback 成员

public void Calculate(int result)

{

MessageBox.Show("结果为： " + result);

}

#endregion

}

1. 添加窗体按钮事件

//双工

private void button3\_Click(object sender, EventArgs e)

{

InstanceContext ic = new InstanceContext(new CallbackHander());

myService.Service1Client client = new myClient.myService.Service1Client(ic);

client.AddNumber(3, 4);

}