客户端调用时需要生成一个代理类

WCF的宿主程序：

IIS、控制台程序、。。。

高层不应该依赖于底层，而是应该依赖于抽象；在高层与底层之间形成一个抽象层。

也就是说高层与底层之间的依赖是松散的，松散耦合。

契约式编程带来的意义：接口的意义。

如果我们采用 类库的形式 它会给我们提供测试客户端。

**服务是一组向客户端提供可用功能的端点(Endpoints)。而端点（终结点）则是网络上的一个能将消息送达的资源。客户端按照与服务之间的契约(Contract)来格式化消息，并将消息发送给端点来访问端点的功能。服务在端点指定的地址（Address）上监听具有特定格式的送达消息。**

**多层分布式架构分析**

传统C/S架构的缺点：

* 当客户端数目激增时，服务器的性能将会因为无法进行负载平衡而大大下降。而一旦应用的需求发生变化，客户端和服务器端的应用程序则都需要修改，这样给应用的维护和升级带来了极大的不便，而且大量数据的传输也增加了网络的负载。

分布式架构的概念

在多层分布式应用中，客户端和服务器之间可以加入一层或多层应用服务程序，这种程序称为“应用服务器”(Application Server)。开发人员可以将企业应用的商业逻辑放在中间层服务器上，而不是客户端，从而将应用的业务逻辑与用户界面隔离开，在保证客户端功能的前提下，**为用户提供一个瘦的(thin)界面**。

这意味着如果需要修改应用程序代码，则可以只在一处(中间层服务器上)修改，而不用修改成千上万的客户端应用程序。 从而使开发人员可以专注于应用系统核心业务逻辑的分析、设计和开发，简化了企业系统的开发、更新和升级工作，极大增强了企业应用的伸缩性和灵活性。

企业要构建多层分布式系统，必须遵循分布式的工业标准，基于什么样的标准直接影响到企业应用系统的开放性和可扩展性。

* 目前分布式对象的标准主要有三种：Microsoft的DCOM、Sun的RMI以及OMG组织的CORBA。
* DCOM是基于Windows环境的分布式对象标准，因此支持的平台种类有限。
* RMI是以Java语言为主体的分布式对象架构，适合大型企业的跨平台需求，但现实的应用系统环境一般是由多种不同的程序语言建立起来的，只依赖一种程序语言构建的企业应用是很少见的。
* CORBA是由800多个大型软、硬件公司参与的OMG组织所制定的分布式对象标准，获得IBM、Sun Microsystems、Oracle、Sybase、Novell、Netscape等大型公司的支持，CORBA标准实现了不同平台之间对象的通信及互操作。

Web service 的组成

1. HTTP(Transport):基于HTTP传输协议，可以穿越防火墙，为互操作性带来好处
2. XML:数据格式
3. SOAP：封装协议，把xml格式外面在套上soap的这种头（相当于信封）
4. WSDL(API Description):方法的一些描述
5. UDDI:相当于一个电话本，作为webservice的目录

并不是所有的对象都可以序列化为所有对象，比如说二维数组。

**服务端**；

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

[WebMethod]

public byte[] getData()

{

DataTable dt = new DataTable("emp");

dt.Columns.Add("NO");

dt.Columns.Add("name");

dt.Columns.Add("age");

DataRow dr = dt.NewRow();

dr["NO"] = "1001";

dr["name"] = "lishiduo";

dr["age"] = "23";

BinaryFormatter bf = new BinaryFormatter();

MemoryStream ms = new MemoryStream();

bf.Serialize(ms, dt); //序列化

byte[] bt = ms.ToArray();

ms.Close();

return bt;//返回字节数组

}

**客户端**

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

{

host.Service1 hs = new webServiceClient.host.Service1();

byte[] bt = hs.getData();

//反序列化

BinaryFormatter bf = new BinaryFormatter();

MemoryStream ms=new MemoryStream(bt);

DataTable dt=(DataTable)bf.Deserialize(ms); //反序列化

ms.Close();

dataGridView1.DataSource = dt;

}

实例二：显示图片

服务端：

[WebMethod]

public byte[] getImage()

{

FileStream fs = new FileStream(Server.MapPath("cc.jpg"),FileMode.OpenOrCreate);

byte[] bt = new byte[fs.Length];

fs.Read(bt, 0, (int)fs.Length);

fs.Close();

return bt;

}

客户端：

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

{

host.Service1 hs = new webServiceClient.host.Service1();

byte[] bt = hs.getImage();

MemoryStream ms = new MemoryStream(bt);

pictureBox1.Image = Image.FromStream(ms);

ms.Close();

}

异步调用webService

客户端：

[WebMethod]

public string hello()

{

Thread.Sleep(5000);

return "hello world";

}

服务端：

//同步：在五秒之内我们不能干其他事情，影响客户体验

private void button4\_Click(object sender, EventArgs e)

{

host.Service1 hs = new webServiceClient.host.Service1();

string result = hs.hello();

MessageBox.Show(result);

}

//异步

private void button5\_Click(object sender, EventArgs e)

{

host.Service1 hs = new webServiceClient.host.Service1();

hs.helloCompleted += new webServiceClient.host.helloCompletedEventHandler(hs\_helloCompleted);

hs.helloAsync();//:该方法是由代理类来生成的

}

void hs\_helloCompleted(object sender, webServiceClient.host.helloCompletedEventArgs e)

{

MessageBox.Show(e.Result);

}

Web service的缺点

1. 速度不高
2. 安全性低
3. 事务操作需要使用EnterpriseService

.NET Remoting从某种意义上讲是DCOM的替代品。是一种基于.net平台的分布式远程对象访问技术。

* **Remoting是一个分布式处理服务。服务器端首先创建通道(Channel),并自动开启监听通道。**

**客户端发出的请求，传递远程对象。**

* **因此，编写Remoting程序，主要分为三部分：**

**1、编写被传递的远程对象；**

**2、编写服务器端监听程序；**

**3、编写客户端请求和处理对象程序；**

接口部分（类库）：

namespace ClassLibrary1

{

public interface Ihello

{

string hello(string name);

}

}

服务端（窗体程序）：

1. 实现接口对应的方法：

namespace Server

{

//远程对象,继承MarshalByRefObject后，对象可以跨应用程序域进行通讯

class myHello : MarshalByRefObject,Ihello

{

#region hello 成员

public string hello(string name)

{

throw new NotImplementedException();

}

#endregion

}

}

1. 启动

//启动

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

{

//创建tcp通道

TcpChannel channel = new TcpChannel(3333);

//注册通道

ChannelServices.RegisterChannel(channel, false);

//注册远程对象

RemotingConfiguration.RegisterWellKnownServiceType(typeof(myHello), "hello", WellKnownObjectMode.SingleCall);

label1.Text = "服务已启动....";

}

客户端（窗体程序）：

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

{

Ihello obj= (Ihello)Activator.GetObject(typeof(Ihello), "tcp://127.0.0.1:3333/hello");

string s= obj.hello("beifeng");

MessageBox.Show(s);

}