**异步调用**

操作系统发展到今天已经十分精巧，线程就是其中一个杰作。操作系统把 CPU 处理时间划分成许多短暂时间片，在时间 T1 执行一个线程的指令，到时间 T2又执行下一线程的指令，各线程轮流执行，结果好象是所有线程在并肩前进。这样，编程时可以创建多个线程，在同一期间执行，各线程可以“并行”完成不同的任务。

在单线程方式下，计算机是一台严格意义上的冯·诺依曼式机器，一段代码调用另一段代码时，只能采用同步调用，必须等待这段代码执行完返回结果后，调用方才能继续往下执行。

有了多线程的支持，可以采用异步调用，调用方和被调方可以属于两个不同的线程，调用方启动被调方线程后，不等对方返回结果就继续执行后续代码。

比如：比如FileStream对象的Read方法

* 同步方式
  + int Read(byte[] buffer,int offset,int count)
* 异步方式
  + IAsyState BeginRead(byte[] buffer,int offset,int count, IAsyncResult callback,Object asyState)
  + int EndRead(IAsyncResult ar)

WCF异步的二种方式

* 异步与否由客户端来决定，在设计之初，不应该去考虑服务的调用者调用的方式。

WCF可以由二种方式生成客户端异步调用的代理类:

1、用svcutil生成异步功能的代理类

2、修改接口定义异步方法

第一种方法例子：

1，WCF服务端写法接口实现方法：

public string hello(string name)

{

System.Threading.Thread.Sleep(5000);

return "hello " + name;

}

2，客户端添加服务引用，同时 勾选 生成异步操作；

3，点击异步按钮，代码如下（注意，注释部分为原先调用方式代码）

//异步

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

{

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

client.helloCompleted += new EventHandler<WCFclient.myService.helloCompletedEventArgs>(client\_helloCompleted);

client.helloAsync("北风");

//client.Beginhello("shiduo", doCallBack, client);

}

void client\_helloCompleted(object sender, WCFclient.myService.helloCompletedEventArgs e)

{

MessageBox.Show(e.Result);

}

//private void doCallBack(IAsyncResult ar)

//{

// string s= ((myService.Service1Client)ar.AsyncState).Endhello(ar);

// MessageBox.Show(s);

//}

**异步调用方法1**

我们用svcutil来生成客户端代理代码的时候,只需添加 /async便可以生成有异步功能的代理类了.

而在IDE中,操作就更加简单,就是在添加ServiceReference的时候,选择高级选项,钩选“创建异步方法”

例子：svcutil /async <http://localhost:8080/Iservice>

**异步调用方法2**

* 修改在客户端的服务定义，增加操作的异步方法。
* 在客户端决定采用异步方式调用服务的操作时，虽然需要修改客户端的服务契约接口，但并不会影响服务端的契约定义。
* [OperationContract(AsyncPattern = true, Action = "",ReplyAction="")]
* IAsyncResult BeginTransferFile(AsyncCallback callback, object asyncState);
* byte[] EndTransferFile(IAsyncResult result);

此处写第二种方法的例子

**异步的优缺点**

* 优点
  + 在I/O受限等情况下，异步能提高性能，并且能更加充分利用多核CPU的优点。
  + 异步能增强系统健壮性(如果一个程序死掉了，其他程序还能继续使用)
  + 异步能改善用户体验
* 缺点
  + 滥用异步，会影响性能
  + 增加编程难度

目前计算机都是双核以上，一个线程总会占用一个cpu（逻辑核）；

**异步与线程**

**实现异步，主要包含以下几种方法**

**1)　使用专用线程，方法为：**

　　System.Threading.ThreadStart ts = new System.Threading.ThreadStart(void(object state) target);

　　System.Threading.Thread th = new System.Threading.Thread(ts);

　　ts.Start();

**2)　使用线程池中的线程**

　　线程是一种比较宝贵的资源，所以使用的时候就要加倍珍惜，线程池中线程在使用完成之后并不是马上销毁，而是回到池中等待下一次的使用，这样就可以较少线程创建的消耗。使用方法如下：

　　ThreadPool.QueueUserWorkItem(WaitCallback callback)

　　需要注意的是此种方法使用的均为后台线程

**3)　使用异步编程模型**

　　这种方法是MS推荐的使用方法，该模型普遍格式为：

　　BeginXXX(…IAsyCallBack callback,object asyState);

　　EndXXX(IAsyState ar);

**4) 使用BackgroundWorker**