12.2异步执行命令 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/140/2021_2022_122_E5_BC_82_E6_AD_A5_c100_140281.htm 在 ADO.NET 2.0 以前，通过 Command 类（如 SqlCommand、OleDbCommand等）执行 SQL 命令的线程一定要停下来等待执行结果。ADO.NET 2.0 新增了异步程序访问接口（asynchronous API），让线程发出命令后可以继续执行接下去的程序代码。而在 ADO.NET 2.0 当前的版本只有 SqlClient 支持异步程序访问接口。以往编写程序时，我们可以直接通过 .NET Framework 所提供的多线程机制，或是以 Delegate 类包装多线程的方式，在 .NET Framework 所提供的异步架构下，设计调用执行 Command 对象实例。这些方法都是让一条工作线程（Worker Thread）停止在后台中等待执行结果，一旦有结果后，工作线程再通过标准的机制告知结果。而何谓多线程与分时多工呢？我们简单地解释一下。你可以想象自己是一个中央处理器（CPU），老板同时交付你很多的工作。在此假设有 A、B、C 三个工作，而你限制自己每小时换做一个不同的工作，所以第一个小时先做 A，时间到后换做 B，一个小时后再换做 C，如此周而复始地切换，而每一条线程就代表了使用中央处理器的单位时间与数据结构。而你在工作时，需要从背后的公文柜中把相关的文件搬到办公桌上，时间一到就需要把正在做的数据放回到相关的公文柜中，空出桌面后再从其他的公文柜搬出下一个工作所需的数据。范例中你是 CPU，公文柜就可以类比成存储器，办公桌则是中央处理器的临时存储器。此种线程交换的过程称为 Context Switch。当然，线程的配置还有

操作系统与中央处理器的用户模式和核心模式切换、优先权顺序、堆栈配置等等诸多复杂的行为，但上述简略的类比应已经让你有了利用中央处理器时间区段所完成多线程程序运行的概念。另外，ADO.NET 2.0 的 SqlClient 利用到 Windows 2000/XP/2003 等操作系统提供的非同步 I/O 机制，也就是每个进程（Process）只需要利用一条线程来处理所有的 I/O 需求，而不是程序设计师自己创建多条工作线程分别等待不同的执行结果。如此可以避免整个应用程序耗在等待的工作线程，有效使用操作系统与中央处理器资源，进而提升整体系统的执行效率。12.2.1 异步执行的方法原本 ADO.NET 的 Command 对象执行 SQL 语法的方法有 ExecuteReader、ExecuteNonQuery、ExecuteXmlReader 以及 ExecuteScalar 等，搭配 .NET Framework 原来就提供的异步模型惯例，除了 ExecuteScalar 方法外，其余的方法都新增了以 Begin 和 End 关键字开始的一对方法。也就是说 ExecuteReader 方法是同步执行，若要以异步的方式执行相同的功能，则调用 BeginExecuteReader 和 EndExecuteReader 这一组方法。在 .NET Framework 中，以 Begin 为字首的函数负责传入同名函数所需的参数，而以 End 为字首的函数用来取回执行结果，例如某个函数的定义如下：Function MyFun（ByVal intIn As Integer） As myObj则以异步调用的起始函数定义如下：Function BeginMyFun（ByVal intIn As Integer, ByVal callback As System.AsyncCallback, ByVal asyncState As Object） As IAsyncResult除了原来函数所传入的 intIn 参数放在第一个位置外，Begin~ 系列的函数会多加存放回调函数（Delegation）的指针参数，也就是上述语法中的 callback 参数。并提供语法中

的 asyncState 参数，让你设置想要带到 End~ 对应函数的信息。而 Begin~ 系列函数最后返回的是代表异步执行状态的 IAsyncResult 对象实例，而不是原本同步执行函数的返回结果，你可以藉此查询异步执行的状况。而获得执行结果的函数定义如下：Function EndMyFun（ByVal asyncResult As System.IAsyncResult） As myObj在调用与 Begin~ 对应的 End~ 函数时，需要带入 Begin~ 函数所返回的 IAsyncResult 对象实例。另外，你可以看到原先我们同步执行时返回的 myObj 对象，在此通过 End~ 函数返回，也就是异步执行完毕后，取回与原先同步执行函数相同的执行结果。由于我们在执行完 Command 对象访问数据库的方法后，都会返回对象，如 ExecuteReader 取回 DataReader 实例；ExecuteNonQuery 取回受影响的记录条数；ExecuteXmlReader 取回 XmlReader 实例。因此大概都需要通过 End 系列函数来获得执行结果，否则这些结果就遗失在系统中。若要异步执行 Command 命令，另一个必需设置的是：数据库连接字符串内要加上 async=true 属性。若连接字符串没有加上该属性，而通过 Command 对象实例调用异步执行的方法，则会产生异常（exception）。若 Command 通过连接执行时，重头到尾都是以同步的方式执行，则依照默认 async=false 的方式设置比较节省资源。若某些命令需要同步执行，另一些需要异步执行，则可以考虑使用不同的连接。在介绍范例应用程序前，我们先稍微谈一下 .NET Framework 所提供的公共的异步运行应用程序设计模式，不只是 ADO.NET 2.0，在其他访问耗时的程序编写上，也都可以套用这个模式。12.2.2 异步运行应用程序设计模式.NET Framework 内置了让应用程序异步运行的功能，让你

在编写应用程序时，不会因为某些耗时等待的操作让程序停止响应，操作界面停滞让用户感觉起来好像死机一样。一般会以多线程的方式处理这种需求，但若你不熟悉线程的运行，或是想利用线程池（Thread Pool）的好处，都可以在较为耗时的操作上，采用 .NET Framework 所提供的异步功能。