**[ADO.NET入门教程（五） 细说数据库连接池](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html)**

**题外话**

通过前几章的学习，不知道大家对ADO.NET有一定的了解了没有。撇开文章质量不讲，必须肯定的是，我是用心去写每一篇文章的。无论是是在排版上，还是在内容选取上我都花了不少心思。我希望通过本系列文章，无论是新手还是老手，在ADO.NET上都能有所收获。**如果大家觉得有帮助，我希望能得到您的推荐和关注，让我知道您对我的肯定。如果大家觉得我写的不好，我也很乐意听取批评的意见，让我们一起进步。**

**摘要**

今天我要讲的是数据库连接池。说实话，我表示鸭梨很大。因为相比其他章节来说，连接池相对来说难理解一点。我要用最通俗的语句给大家讲明白，讲透彻却也不是一件很容易的事。但是，连接池又是非常重要的知识点，特别是在部署多用户程序时，显得尤为重要。所以，我不但要讲，而且要讲的透彻。通过本文，你将理解连接池的基本原理已经如何利用连接池来提高应用程序的性能。

**目录**

* [什么是连接池?](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_1)
* [连接池的工作原理](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_2)
* [说说连接池的几个非常重要属性](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_3)
* [连接池异常与处理方法](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_4)
* [监视连接状态的方法](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_5)
* [高效使用连接池的基本原则](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_6)

**1. 什么是连接池?**

      在上篇文章[《你必须知道的ADO.NET（四） 品味Connection对象》](http://www.cnblogs.com/liuhaorain/archive/2012/02/15/2349886.html)中，我已经强调过，**建立一个数据库连接是一件非常耗时（消耗时间）耗力（消耗资源）的事情。**之所以会这样，是因为连接到数据库服务器需要经历几个漫长的过程:建立物理通道（例如套接字或命名管道），与服务器进行初次握手，分析连接字符串信息，由服务器对连接进行身份验证，运行检查以便在当前事务中登记等等。我们先不管为什么会有这样的机制，存在总是有它的道理。既然新建一条连接如此痛苦，那么为什么不重复利用已有的连接呢？

      实际上，ADO.NET已经为我们提供了名为连接池的优化方法。**连接池就是这样一个容器：它存放了一定数量的与数据库服务器的物理连接。**因此，当我们需要连接数据库服务器的时候，只需去池（容器）中取出一条空闲的连接，而不是新建一条连接。这样的话，我们就可以大大减少连接数据库的开销，从而提高了应用程序的性能。

**PS:**本来做了2张图片来描述连接池的，无奈公司装有监控软件，不能上传，所以只能等下次有时间上传了。

**2. 连接池的工作原理**

**2.1 创建连接池**

      需要说明的是，**连接池是具有类别区分的。**也就是说，同一个时刻同一应用程序域可以有多个不同类型的连接池。那么，连接池是如何标识区分的？细致的讲，是由进程、应用程序域、连接字符串以及windows标识（在使用集成的安全性时）共同组成签名来标识区分的。但对于同一应用程序域来说，一般只由连接字符串来标识区分。当打开一条连接时，如果该条连接的类型签名与现有的连接池类型不匹配，则创建一个新的连接池。反之，则不创建新的连接池。

      一个典型的创建连接的实例：

[复制代码](javascript:void(0);)

//创建连接对象1  
using (SqlConnection conn1 =  
  
 new SqlConnection( "DataSource=(local);Integrated Security=SSPI;Initial Catalog=Northwind"))   
{   
 conn1.Open();   
}  
  
  
//创建连接对象2  
using (SqlConnection conn2 =  
  
 new SqlConnection( "DataSource=(local);Integrated Security=SSPI;Initial Catalog=pubs"))   
{   
 conn2.Open();   
}  
  
  
//创建连接对象3  
using (SqlConnection conn3 =  
  
 new SqlConnection( "DataSource=(local);Integrated Security=SSPI;Initial Catalog=Northwind"))   
{   
 conn3.Open();   
}

[复制代码](javascript:void(0);)

上面实例中，我创建了三个SqlConnection对象，但是管理时只需要两个连接池。细心的朋友，可能早已发现**conn1与conn3的连接字符串相同，所以可以共享一个连接池，而conn2与conn1与conn3不同，所以需要创建新的连接池。**

**2.2 分配空闲连接**

      当用户创建连接请求或者说调用Connection对象的Open时，**连接池管理器首先需要根据连接请求的类型签名找到匹配类型的连接池，然后尽力分配一条空闲连接。**具体情况如下：

* 如果池中有空闲连接可用，返回该连接。
* 如果池中连接都已用完，创建一个新连接添加到池中。
* 如果池中连接已达到最大连接数，请求进入等待队列直到有空闲连接可用。

**2.3 移除无效连接**

      无效连接，即不能正确连接到数据库服务器的连接。对于连接池来说，存储的与数据库服务器的连接的数量是有限的。因此，对于无效连接，如果如不及时移除，将会浪费连接池的空间。其实你不用担心，连接池管理器已经很好的为我们处理了这些问题。如果连接长时间空闲，或检测到与服务器的连接已断开，连接池管理器会将该连接从池中移除。

**2.4 回收使用完的连接**

**当我们使用完一条连接时，应当及时关闭或释放连接，以便连接可以返回池中重复利用。**我们可以通过Connection对象的Close或Dispose方法，也可以通过C#的using语句来关闭连接。

**3. 说说几个非常重要属性**

**连接池的行为可以通过连接字符串来控制**，主要包括四个重要的属性：

* **Connection Timeout：**连接请求等待超时时间。默认为15秒，单位为秒。
* **Max Pool Size:**连接池中最大连接数。默认为100。
* **Min Pool Size:**连接池中最小连接数。默认为0。
* **Pooling:**是否启用连接池。**ADO.NET默认是启用连接池的，**因此，你需要手动设置Pooling=false来禁用连接池。

还是看一个实例来理解连接池的属性吧。代码如下：

[复制代码](javascript:void(0);)

SqlConnectionStringBuilder connStr = new SqlConnectionStringBuilder();  
 connStr.DataSource = @".\SQLEXPRESS";  
 connStr.InitialCatalog = "master";  
 connStr.IntegratedSecurity = true;  
  
 connStr.Pooling = true; //开启连接池  
 connStr.MinPoolSize = 0; //设置最小连接数为0  
 connStr.MaxPoolSize = 50; //设置最大连接数为50   
 connStr.ConnectTimeout = 10; //设置超时时间为10秒  
  
 using( SqlConnection conn = new SqlConnection(connStr.ConnectionString))  
 {  
 ;//todo  
 }

[复制代码](javascript:void(0);)

**4. 连接池异常与处理方法**

      当用户打开一个连接而没有正确或者及时的关闭时，经常会引发“连接泄露”问题。泄露的连接，会一直保持打开状态，直到调用Dispose方法，垃圾回收器（GC）才关闭和释放连接。与ADO不同，ADO.NET需要手动的关闭使用完的连接。**一个重要的误区是：当连接对象超出局部作用域范围时，就会关闭连接。**实际上，当超出作用域时，**释放的只是连接对象而非连接资源。**好吧，还是先看看一个实例吧。

[复制代码](javascript:void(0);)

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Data;  
using System.Data.SqlClient;  
  
namespace ConnectionPool  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 SqlConnectionStringBuilder connStr = new SqlConnectionStringBuilder();  
 connStr.DataSource = @".\SQLEXPRESS";  
 connStr.InitialCatalog = "master";  
 connStr.IntegratedSecurity = true;  
  
 connStr.MaxPoolSize = 5;//设置最大连接池为5  
 connStr.ConnectTimeout = 1;//设置超时时间为1秒  
  
 SqlConnection conn = null;  
 for (int i = 1; i <= 100; ++i)  
 {  
 conn = new SqlConnection(connStr.ConnectionString);  
 try  
 {  
 conn.Open();  
 Console.WriteLine("Connection{0} is linked",i);  
 }  
 catch(Exception ex)  
 {  
 Console.WriteLine("\n异常信息:\n{0}",ex.Message);  
 break;  
 }  
 }  
  
 Console.Read();  
  
 }  
 }  
}

[复制代码](javascript:void(0);)

为了使结果更明显，我特地将最大连接数设置为5，超时时间为1秒。运行后，很快得到以下结果。

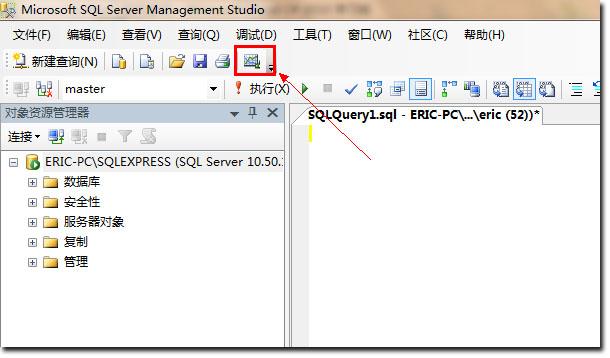


从上面的结果我们很明显的知道，连接出现了异常。我们已经知道连接池的最大连接数为5，当创建第6条连接时，由于连接池中连接数量已经达到了最大数并且没有空闲的连接，因此需要等待连接直到超时。当超过超时时间时，就出现了上述的连接异常。因此，**我必须再次强调，使用完的连接应当尽快的正确的关闭和释放。**

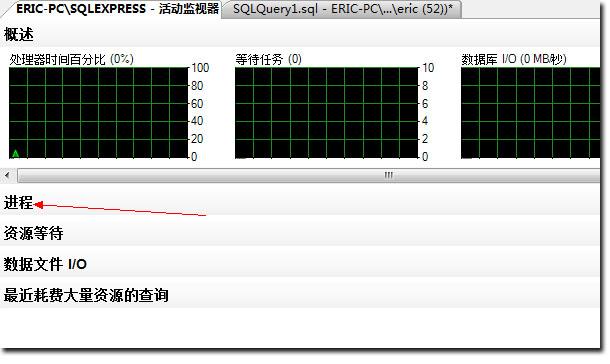
**5. 监视SQL Server连接状态的方法**

**（1）通过活动监视器**

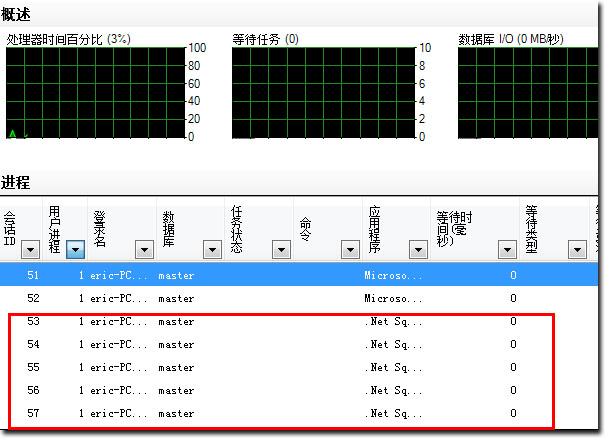
**第一步：**打开MSSMS管理器，单击“**活动监视器**”图标。



**第二步：**在打开活动监视器视图中，单击“**进程**”选项卡。



**第三步：**运行 [**#4 连接池异常与处理方法**](http://www.cnblogs.com/liuhaorain/archive/2012/02/19/2353110.html#title_4) 中的例子，则可以看到打开的5条连接，如下图所示。

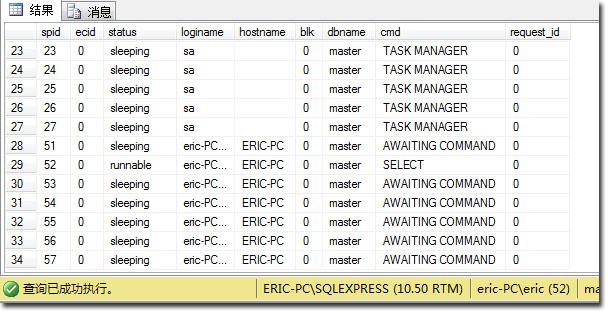


**（2）使用T-SQL语句**

同样，通过执行系统存储过程**sp\_who**，我们也可以监视连接状态。

exec sp\_who

可得到以下结果：



**6. 高效使用连接池的基本原则**

      用好连接池将会大大提高应用程序的性能。相反，如果使用不当的话，则百害而无一益。一般来说，应当遵循以下原则：

* 在最晚的时刻申请连接，在最早的时候释放连接。
* 关闭连接时先关闭相关用户定义的事务。
* 确保并维持连接池中至少有一个打开的连接。
* 尽力避免池碎片的产生。主要包括集成安全性产生的池碎片以及使用许多数据库产生的池碎片。

**提示：**池碎片是许多 Web 应用程序中的一个常见问题，应用程序可能会创建大量在进程退出后才会释放的池。 这样，将打开大量的连接，占用许多内存，从而导致性能降低。