**[ADO.NET入门教程（六） 谈谈Command对象与数据检索](http://www.cnblogs.com/liuhaorain/archive/2012/02/27/2361825.html)**

**摘要**

到目前为止，我相信大家对于ADO.NET如何与外部数据源建立连接以及如何提高连接性能等相关知识已经牢固于心了。连接对象作为ADO.NET的主力先锋，为用户与数据库交互搭建了扎实的桥梁。它的一生是平凡而又伟大的，总是尽自己最大的努力为用户搭建一条通往数据库的平坦大道。相比连接对象来说，Command对象似乎耀眼的多。Command对象在ADO.NET世界里总是忙忙碌碌，它就像一个外交官，为用户传达了所有操作数据库的信息。

**1. 准备**

      学习知识最快也最好的方法，那就是将理论与实践相结合。为了帮助大家更好的理解和掌握Command对象，我也准备了很多实践的例子。我希望大家能做好充分的准备，这样的话不至于在实践的时候手忙脚乱。您需要准备以下几件事情：

（1）确保你的电脑装有SQL Server 2005/2008数据库服务器。如果未装有SQL Server服务器，**[点此下载](http://www.microsoft.com/betaexperience/pd/SQLEXP08V2/enus/" \t "_blank)** SQL Server 2008 EXPRESS R2。

（2）创建一个名为db\_MyDemo的数据库。

[复制代码](javascript:void(0);)

USE master;  
GO  
CREATE DATABASE db\_MyDemo  
ON  
(  
 NAME = MyDemo\_data, /\*指定文件的逻辑名称\*/  
 FILENAME = 'D:\mydemo\_dat.mdf',/\*物理文件名称\*/  
 SIZE = **10**,/\*指定文件大小，单位MB\*/  
 MAXSIZE = **50**,/\*文件最大值，单位MB\*/  
 FILEGROWTH = **5** /\*自动增量\*/  
)  
LOG ON  
(  
 Name = MyDemo\_log,  
 FILENAME = 'D:\mydemo\_log.ldf',  
 SIZE = 5MB,  
 MAXSIZE = 25MB,  
 FILEGROWTH = 5MB  
)  
GO

[复制代码](javascript:void(0);)

（3）创建顾客表tb\_SelCustomer。

[复制代码](javascript:void(0);)

use db\_MyDemo;  
GO  
CREATE TABLE tb\_SelCustomer  
(  
 ID INT IDENTITY(**1**,**1**) PRIMARY KEY, /\*ID,主键\*/  
 Name varchar(**20**) NOT NULL, /\*姓名\*/  
 Sex char(**1**) default('0'), /\*性别：0为男，1为女，默认为0\*/  
 CustomerType char(**1**) default('0'), /\*客户类型：0为普通用户，1为VIP用户，默认为0\*/  
 Phone varchar(**12**), /\*联系电话\*/  
 Email varchar(**50**), /\*电子邮件\*/  
 ContactAddress varchar(**200**), /\*联系地址\*/  
 Lat float, /\*所在位置维度，用于在地图显示\*/  
 Lng float, /\*所在位置经度，用于在地图显示\*/  
 Postalcode varchar(**10**), /\*邮政编码\*/  
 Remark varchar(**50**) /\*备注\*/  
)

[复制代码](javascript:void(0);)

 好了，恭喜你！终于把准备工作做好了。下面，让我们一起来揭开Command对象的面纱！

**2. 什么是Command对象？**

      我们知道ADO.NET最主要的目的对外部数据源提供一致的访问。而访问数据源数据，就少不了增删查改等操作。尽管Connection对象已经我们连接好了外部数据源，但它却忠于职守，并不提供对外部数据源的任何操作。就在纠结万分的时刻，Command对象诞生了。**它封装了所有对外部数据源的操作（包括增、删、查、改等SQL语句与存储过程），并在执行完成后返回合适的结果。**与Connection对象一样，对于不同的数据源，ADO.NET提供了不同的Command对象。具体来说，可分为以下Command对象。

|  |  |
| --- | --- |
| **.NET数据提供程序** | **对应Command对象** |
| 用于 OLE DB 的 .NET Framework 数据提供程序 | OleDbCommand对象 |
| 用于 SQL Server 的 .NET Framework 数据提供程序 | SqlCommand对象 |
| 用于 ODBC 的 .NET Framework 数据提供程序 | OdbcCommand 对象 |
| 用于 Oracle 的 .NET Framework 数据提供程序 | OracleCommand 对象 |

不管是哪种Command对象，它都继承于DBCommand类。与DBConnection类一样，DBCommand类也是抽象基类，不能被实例化，并期待派生类（对应于特定.NET数据提供程序的Command类）来实现方法。 DBCommand类定义了完善的健全的数据库操作的基本方法和基本属性，它的结构如下：

public abstract class DbCommand : Component,   
 IDbCommand, IDisposable

从上面我们可以知道，它继承了Component类以及IDbCommand接口和IDisposable接口。

**3. 必须掌握的几个属性**

**CommandText:**获取或设置对数据源执行的文本命令。默认值为空字符串。

**CommandType:** 命令类型，指示或指定如何解释CommandText属性。CommandType属性的值是一个枚举类型，定义结构如下：

public enum CommandType   
{ Text = 1, //SQL 文本命令。（默认。）   
 StoredProcedure = 4, // 存储过程的名称。   
 TableDirect = 512 //表的名称。   
}

需要特别注意的是，**将CommandType 设置为 StoredProcedure 时，应将 CommandText 属性设置为存储过程的名称。** 当调用 Execute 方法之一时，该命令将执行此存储过程。

**Connection:** 设置或获取与数据源的连接。

**Parameters:** 绑定SQL语句或存储过程的参数。参数化查询中不可或缺的对象，非常重要。

**Tranction:** 获取或设置在其中执行 .NET Framework 数据提供程序的 Command 对象的事务。

**4. 必须掌握的几个方法**

**ExecuteNonQuery:** 执行不返回数据行的操作，并返回一个int类型的数据。

**注意：**对于 UPDATE、INSERT 和 DELETE 语句，返回值为该命令所影响的行数。 对于其他所有类型的语句，返回值 为 -1。

**ExecuteReader:** 执行查询，并返回一个 DataReader 对象。  
 **ExecuteScalar:** 执行查询，并返回查询结果集中第一行的第一列（object类型）。如果找不到结果集中第一行的第一列，则返回 null 引用。

**5. 如何创建Command对象？**

      在创建Command对象之前，你需要明确两件事情：**（1）你要执行什么样的操作？（2）你要对哪个数据源进行操作？**明白这两件事情，一切都好办了。我们可用通过string字符串来构造一条SQL语句，也可以通过Connection对象指定连接的数据源。那么我们如何将这些信息交给Command对象呢？一般来说，有两种方法：

**（1）通过构造函数。**代码如下：

string strSQL = "Select \* from tb\_SelCustomer";  
SqlCommand cmd = new SqlCommand(strSQL, conn);

**（2）通过Command对象的属性。**代码如下：

SqlCommand cmd = new SqlCommand();  
cmd.Connection = conn;  
cmd.CommandText = strSQL;

**提示：**上面两个实例是相对于SQL Server来说的，如果你访问其他数据源，应当选择其他的Command对象。具体参照 #2 什么是Command对象 中的表格。

**6. 选择合适的执行命令**

      Command对象提供了丰富的执行命令操作，具体方法可参考 [**#4 必须掌握的几个方法**](http://www.cnblogs.com/liuhaorain/archive/2012/02/27/2361825.html#title_4) 。凡是有利有弊，Comandante对象既然提供多种执行命令，我们在实际开发中就要有所取舍，选择合适的执行命令。其实，用户对数据源的操作不外乎**CRUD-S**（Create、Update、Delete、Select）操作。下面我将探讨如何在不同的场景选择合适的执行命令。

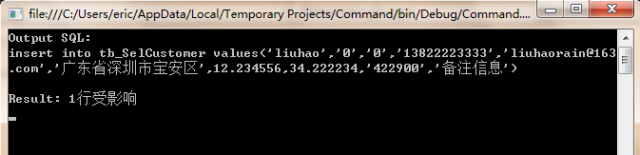
**（1）场景一：执行CRUD操作，不返回数据行，返回影响的行数（可选）**

      当我们对数据表的行（记录）进行增加，删除，更新操作或者处理数据定义语句（比如用Create Table来创建表结构）时，实际上数据库是不返回数据行的，仅仅返回一个包含影响行数信息的整数。一般地，在执行非查询操作时，我们需要调用ExcuteNonQuery方法。还是，先看一个实例吧！我们在tb\_SelCustomer表中插入一行记录，代码如下：

[复制代码](javascript:void(0);)

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Data;//必须引入  
using System.Data.SqlClient;//必须引入  
  
namespace Command  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 string connSQL = @"Data Source=.\SQLEXPRESS; Initial Catalog=db\_MyDemo; Integrated Security=SSPI";//构造连接字符串  
 SqlConnectionStringBuilder connStr = new SqlConnectionStringBuilder(connSQL);  
  
 using(SqlConnection conn = new SqlConnection(connStr.ConnectionString))  
 {  
 //拼接SQL语句  
 StringBuilder strSQL = new StringBuilder();  
 strSQL.Append("insert into tb\_SelCustomer ");  
 strSQL.Append("values(");  
 strSQL.Append("'liuhao','0','0','13822223333','liuhaorain@163.com','广东省深圳市宝安区',12.234556,34.222234,'422900','备注信息')");  
  
 Console.WriteLine("Output SQL:\n{0}",strSQL.ToString());  
  
 //创建Command对象  
 SqlCommand cmd = new SqlCommand();  
 cmd.Connection = conn;  
 cmd.CommandType = CommandType.Text;  
 cmd.CommandText = strSQL.ToString();  
  
 try  
 {  
 conn.Open();//一定要注意打开连接  
  
 int rows = cmd.ExecuteNonQuery();//执行命令  
 Console.WriteLine("\nResult: {0}行受影响",rows);  
 }  
 catch(Exception ex)  
 {  
 Console.WriteLine("\nError：\n{0}", ex.Message);  
 }  
 }  
  
 Console.Read();  
 }  
 }  
}

 运行后，输出结果如下：



从上面输出信息我们可以看到，已经成功的添加一行数据。

**（2）场景二：执行Select操作，返回多个数据**

      当我们通过执行Select操作返回一行或多行数据时，这时候ExcuteNonQuery就需要休息了。不过也别慌，话说Command对象手下猛将如云，处理这种事情就需要请ExcuteReader方法上场了。ExcuteReader方法返回一个DataReader对象。**DataReader是一个快速的，轻量级，只读的遍历访问每一行数据的数据流。**使用DataReader时，需要注意以下几点：

* DataReader一次遍历一行数据，并返回一个包含列名字集合。
* 第一次调用Read()方法获取第一行数据，并将游标指向下一行数据。当再次调用该方法时候，将读取下一行数据。
* 当检测到不再有数据行时，Read()方法将返回false。
* 通过HasRows属性，我们知道查询结果中是否有数据行。
* 当我们使用完DataReader时，一定要注意关闭。SQL Server默认只允许打开一个DataReader。

好吧，还是先看一个简单的例子吧。查询出tb\_SelCustomer表中所有的数据。代码如下：

[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 Command2  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 //构造连接字符串  
 SqlConnectionStringBuilder strConn = new SqlConnectionStringBuilder();  
 strConn.DataSource = @"(local)\SQLEXPRESS";  
 strConn.InitialCatalog = "db\_MyDemo";  
 strConn.IntegratedSecurity = true;  
  
 using (SqlConnection conn = new SqlConnection(strConn.ConnectionString))  
 {  
 string strSQL = "select \* from tb\_SelCustomer";  
 SqlCommand cmd = new SqlCommand(strSQL, conn);//使用构造函数方式创建Command对象  
  
 conn.Open();//记得打开连接  
  
 try  
 {  
 SqlDataReader reader = cmd.ExecuteReader();//执行ExecuteReader  
  
 if (reader != null && reader.HasRows)  
 {  
 int rows = 0;//记录行数  
 Console.WriteLine("\*\*\*\*\*\*\*\*\*\*Records of tb\_SelCustomer\*\*\*\*\*\*\*\*\*\*\n");  
 while (reader.Read())  
 {  
 for (int i = 0; i < reader.FieldCount; ++i)  
 {  
 Console.WriteLine("{0}:{1}", reader.GetName(i), reader.GetValue(i));  
 }  
 ++rows;  
 }  
 Console.WriteLine("\n共{0}行记录", rows);  
 }  
 reader.Close();//关闭DataReader  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine("\nError:\n{0}", ex.Message);  
 }  
 }  
  
 Console.Read();  
 }  
 }  
}

执行上面代码后，我们可以看到在**场景（一）** 中添加的数据。结果下图所示。



**（3）场景三：执行Select操作，返回单个值**

      上面两个场景相信大家都十分熟悉了。但是，当我们在操作数据库时仅仅只需要返回一个值（比如返回行数），那该怎么办呢？在此，我不得不承认Command对象确实人才济济，ExcuteScalar方法就是处理单个数据最优秀的人才。**ExcuteScalar返回一个System.Object类型的数据**，因此我们在获取数据时需要进行强制类型转换。当没有数据时，ExcuteScalar方法返回System.DBNull。理论在于实践，我们还是先看看一个实例吧！获取tb\_SelCustomer中的行数。代码如下：

[复制代码](javascript:void(0);)

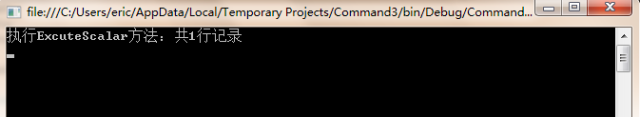
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Data;  
using System.Data.SqlClient;  
  
namespace Command3  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 string connSQL = @"Data Source=.\SQLEXPRESS; Initial Catalog=db\_MyDemo; Integrated Security=SSPI";

using (SqlConnection conn = new SqlConnection(connSQL))  
{  
string strSQL = "select count(\*) from tb\_SelCustomer";  
SqlCommand cmd = new SqlCommand(strSQL, conn);//创建Command对象   
  
try  
{  
conn.Open();//一定要注意打开连接   
int rows = (int)cmd.ExecuteScalar();//执行命令   
Console.WriteLine("执行ExcuteScalar方法：共{0}行记录", rows);  
}  
catch (Exception ex)  
{  
Console.WriteLine("\nError：\n{0}", ex.Message);  
}  
}

Console.Read();  
 }  
 }  
}

[复制代码](javascript:void(0);)

 输出结果如下：



**7. 总结**

Command对象就是一个载体。它向数据库传达了用户的操作信息，而数据库则通过Command对象向用户返回处理结果。