# **[C#操作SQL Server数据库](https://www.cnblogs.com/rainman/archive/2012/03/13/2393975.html)**

### **1、概述**

ado.net提供了丰富的数据库操作，这些操作可以分为三个步骤：

* 第一，使用SqlConnection对象连接数据库；
* 第二，建立SqlCommand对象，负责SQL语句的执行和存储过程的调用；
* 第三，对SQL或存储过程执行后返回的“结果”进行操作。

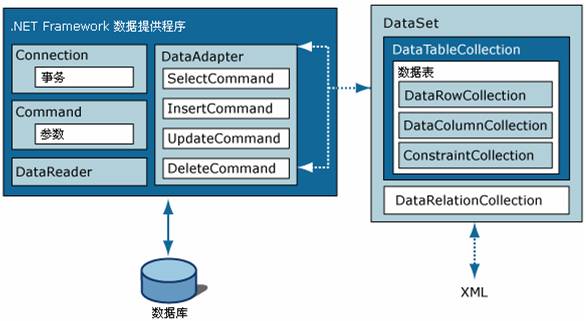
对返回“结果”的操作可以分为两类：

* 一是用SqlDataReader直接一行一行的读取数据集；
* 二是DataSet联合SqlDataAdapter来操作数据库。

两者比较：

* SqlDataReader时刻与远程数据库服务器保持连接，将远程的数据通过“流”的形式单向传输给客户端，它是“只读”的。由于是直接访问数据库，所以效率较高，但使用起来不方便。
* DataSet一次性从数据源获取数据到本地，并在本地建立一个微型数据库（包含表、行、列、规则、表之间的关系等），期间可以断开与服务器的连接，使用SqlDataAdapter对象操作“本地微型数据库”，结束后通过SqlDataAdapter一次性更新到远程数据库服务器。这种方式使用起来更方，便简单。但性能较第一种稍微差一点。（在一般的情况下两者的性能可以忽略不计。）

一张十分出名的ADO.NET结构图：



### **2、连接字符串的写法**

string connectString = "Data Source=.;Initial Catalog=Student;Integrated Security=True";

### **3、SqlConnection对象**

命名空间：System.Data.SqlClient.SqlConnection;

返回数据库连接对象，参数字符串。实例化“连接对象”，并打开连接

SqlConnection sqlCnt = new SqlConnection(connectString);

sqlCnt.Open();

使用完成后，需要关闭“连接对象”

sqlCnt.Close();

### **4、SqlCommand对象**

命名空间：System.Data.SqlClient.SqlCommand;

SqlCommand对象用于执行数据库操作，操作方式有三种：

* SQL语句：command.CommandType = CommandType.Text;
* 存储过程：command.CommandType = CommandType.StoredProcedure;
* 整张表：command.CommandType = CommandType.TableDirect;

实例化一个SqlCommand对象

SqlCommand command = new SqlCommand();

command.Connection = sqlCnt; // 绑定SqlConnection对象

或直接从SqlConnection创建

SqlCommand command = sqlCnt.CreateCommand();

### **常用方法：**

* command.ExecuteNonQuery(): 返回受影响函数，如增、删、改操作；
* command.ExecuteScalar()：执行查询，返回首行首列的结果；
* command.ExecuteReader()：返回一个数据流（SqlDataReader对象）。

### **常用操作**

##### **① 执行SQL**

SqlCommand cmd = conn.CreateCommand(); //创建SqlCommand对象

cmd.CommandType = CommandType.Text;

cmd.CommandText = "select \* from products = @ID"; //sql语句

cmd.Parameters.Add("@ID", SqlDbType.Int);

cmd.Parameters["@ID"].Value = 1; //给参数sql语句的参数赋值

##### **② 调用存储过程**

SqlCommand cmd = conn.CreateCommand();

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.CommandText = "存储过程名";

##### **③ 整张表**

SqlCommand cmd = conn.CreateCommand();

cmd.CommandType = System.Data.CommandType.TableDirect;

cmd.CommandText = "表名"

### **5、SqlDataReader对象**

命名空间：System.Data.SqlClient.SqlDataReader;

SqlDataReader对象提供只读单向数据的功能，单向：只能依次读取下一条数据；只读：DataReader中的数据是只读的，不能修改；相对地DataSet中的数据可以任意读取和修改.

它有一个很重要的方法，是Read()，返回值是个布尔值，作用是前进到下一条数据，一条条的返回数据，当布尔值为真时执行，为假时跳出。如

SqlCommand command = new SqlCommand();

command.Connection = sqlCnt;

command.CommandType = CommandType.Text;

command.CommandText = "Select \* from Users";

SqlDataReader reader = command.ExecuteReader(); //执行SQL，返回一个“流”

while (reader.Read())

{

Console.Write(reader["username"]); // 打印出每个用户的用户名

}

### **6、DataSet对象**

### **6.1 SqlDataAdapter;**

命名空间：System.Data.SqlClient.SqlDataAdapter;

SqlDataAdapter是SqlCommand和DataSet之间的桥梁，实例化SqlDataAdapter对象：

SqlConnection sqlCnt = new SqlConnection(connectString);

sqlCnt.Open();

// 创建SqlCommand

SqlCommand mySqlCommand = new SqlCommand();

mySqlCommand.CommandType = CommandType.Text;

mySqlCommand.CommandText = "select \* from product";

mySqlCommand.Connection = sqlCnt;

// 创建SqlDataAdapter

SqlDataAdapter myDataAdapter = new SqlDataAdapter();

myDataAdapter.SelectCommand = mySqlCommand; // 为SqlDataAdapter对象绑定所要执行的SqlCommand对象

上述SQL可以简化为

SqlConnection sqlCnt = new SqlConnection(connectString);

sqlCnt.Open();

// 隐藏了SqlCommand对象的定义，同时隐藏了SqlCommand对象与SqlDataAdapter对象的绑定

SqlDataAdapter myDataAdapter = new SqlDataAdapter("select \* from product", sqlCnt);

##### **属性和方法**

* myDataAdapter.SelectCommand属性：SqlCommand变量，封装Select语句；
* myDataAdapter.InsertCommand属性：SqlCommand变量，封装Insert语句；
* myDataAdapter.UpdateCommand属性：SqlCommand变量，封装Update语句；
* myDataAdapter.DeleteCommand属性：SqlCommand变量，封装Delete语句。
* myDataAdapter.fill()：将执行结果填充到Dataset中，会隐藏打开SqlConnection并执行SQL等操作。

### **6.2 SqlCommandBuilder;**

命名空间：System.Data.SqlClient.SqlCommandBuilder。

对DataSet的操作（更改、增加、删除）仅是在本地修改，若要提交到“数据库”中则需要SqlCommandBuilder对象。用于在客户端编辑完数据后，整体一次更新数据。具体用法如下：

SqlCommandBuilder mySqlCommandBuilder = new SqlCommandBuilder(myDataAdapter); // 为myDataAdapter赋予SqlCommandBuilder功能

myDataAdapter.Update(myDataSet, "表名"); // 向数据库提交更改后的DataSet，第二个参数为DataSet中的存储表名，并非数据库中真实的表名（二者在多数情况下一致）。

### **6.3 DataSet**

命名空间：System.Data.DataSet。

数据集，本地微型数据库，可以存储多张表。

使用DataSet第一步就是将SqlDataAdapter返回的数据集（表）填充到Dataset对象中：

SqlDataAdapter myDataAdapter = new SqlDataAdapter("select \* from product", sqlCnt);

DataSet myDataSet = new DataSet(); // 创建DataSet

myDataAdapter.Fill(myDataSet, "product"); // 将返回的数据集作为“表”填入DataSet中，表名可以与数据库真实的表名不同，并不影响后续的增、删、改等操作

##### **① 访问DataSet中的数据**

SqlDataAdapter myDataAdapter = new SqlDataAdapter("select \* from product", sqlCnt);

DataSet myDataSet = new DataSet();

myDataAdapter.Fill(myDataSet, "product");

DataTable myTable = myDataSet.Tables["product"];

foreach (DataRow myRow in myTable.Rows) {

foreach (DataColumn myColumn in myTable.Columns) {

Console.WriteLine(myRow[myColumn]); //遍历表中的每个单元格

}

}

##### **② 修改DataSet中的数据**

SqlDataAdapter myDataAdapter = new SqlDataAdapter("select \* from product", sqlCnt);

DataSet myDataSet = new DataSet();

myDataAdapter.Fill(myDataSet, "product");

// 修改DataSet

DataTable myTable = myDataSet.Tables["product"];

foreach (DataRow myRow in myTable.Rows) {

myRow["name"] = myRow["name"] + "商品";

}

// 将DataSet的修改提交至“数据库”

SqlCommandBuilder mySqlCommandBuilder = new SqlCommandBuilder(myDataAdapter);

myDataAdapter.Update(myDataSet, "product");

注意：在修改、删除等操作中表product必须定义主键，select的字段中也必须包含主键，否则会提示“对于不返回任何键列信息的 SelectCommand,不支持 UpdateCommand 的动态 SQL 生成。”错误

##### **③ 增加一行**

SqlDataAdapter myDataAdapter = new SqlDataAdapter("select \* from product", sqlCnt);

DataSet myDataSet = new DataSet();

myDataAdapter.Fill(myDataSet, "product");

DataTable myTable = myDataSet.Tables["product"];

// 添加一行

DataRow myRow = myTable.NewRow();

myRow["name"] = "捷安特";

myRow["price"] = 13.2;

//myRow["id"] = 100; id若为“自动增长”，此处可以不设置，即便设置也无效

myTable.Rows.Add(myRow);

// 将DataSet的修改提交至“数据库”

SqlCommandBuilder mySqlCommandBuilder = new SqlCommandBuilder(myDataAdapter);

myDataAdapter.Update(myDataSet, "product");

##### **④ 删除一行**

SqlDataAdapter myDataAdapter = new SqlDataAdapter("select \* from product", sqlCnt);

DataSet myDataSet = new DataSet();

myDataAdapter.Fill(myDataSet, "product");

// 删除第一行

DataTable myTable = myDataSet.Tables["product"];

myTable.Rows[0].Delete();

SqlCommandBuilder mySqlCommandBuilder = new SqlCommandBuilder(myDataAdapter);

myDataAdapter.Update(myDataSet, "product");

##### **属性**

* Tables：获取包含在DataSet中的表的集合。
* Relations：获取用于将表链接起来并允许从父表浏览到子表的关系的集合。
* HasEroors：表明是否已经初始化DataSet对象的值。

##### **方法**

* Clear清除DataSet对象中所有表的所有数据。
* Clone复制DataSet对象的结构到另外一个DataSet对象中，复制内容包括所有的结构、关系和约束，但不包含任何数据。
* Copy复制DataSet对象的数据和结构到另外一个DataSet对象中。两个DataSet对象完全一样。
* CreateDataReader为每个DataTable对象返回带有一个结果集的DataTableReader，顺序与Tables集合中表的显示顺序相同。
* Dispose释放DataSet对象占用的资源。
* Reset将DataSet对象初始化。

### **7、释放资源**

资源使用完毕后应及时关闭连接和释放，具体方法如下：

myDataSet.Dispose(); // 释放DataSet对象

myDataAdapter.Dispose(); // 释放SqlDataAdapter对象

myDataReader.Dispose(); // 释放SqlDataReader对象

sqlCnt.Close(); // 关闭数据库连接

sqlCnt.Dispose(); // 释放数据库连接对象