C#下数据库编程

（一）

前言：

曾几何时，OO 语言大家族中又多了一位新成员，他有个坚强、锐利而又不失好听的名字，C#（读做 C Sharp）。C#由 Microsoft 一手打造，更重要的是其总设计师就是 Turbo Pascal、Delphi 的缔造者、天才的 Anders。记得在一次 C#的演示会上，Anders 和 Microsoft 的主管人员曾立下壮志，要让 C#取代 VC++，成为今后.NET 开发的首选语言。但后来的许多负面的议论也不免让人对这个 C#表示怀疑，有人说 C#就是 JAVA 的翻版，几乎没有什么自己的特点，还有人说 C#效率低下，而且和 VB.NET 有很大类同，根本就没有意义推出，不久就会下台并最终死亡，等等。我们也不禁怀疑起来，C#真的这么命短？真的会与 JAVA 没有什么大的交锋就英年早逝？

两年多过去了，这种议论大部分已经随着事实不攻自破。C#不但没有死亡，而且有着超乎寻常的生命力，正在用他锋利的语言利剑，让世人真正领教了什么叫做系出名门、天生立志。在.NET 平台开发上，C#已经成为无可争议的首选语言，更令人吃惊的是，这个人称与 JAVA 雷同的“仿制品”语言却不知道为什么，一步步在蚕食 JAVA 的地盘，令 JAVA 的缔造者已经开始感到不安，更让 JAVA 的使用者或多或少有些动摇。

大约 1 年前，我接触到了 C#，并且试着以我一贯审视 IT 界发展的眼光去审视他。通过一些日子的学习与体会，我确实感觉这个语言的不平凡性，特别是他充分利用了.NET 的优势和特点，并有 VS.NET 的完美集成于.NET 开发平台中。在我长年开发的数据库领域，我试着用 C#结合 ADO.NET 开发了一些项目。现借这个机会，和读者们一起分享 C#开发数据库的快乐。

正文：

如果你以前用过 Visual Foxpro 开发数据库项目，你就会有这种体会，VFP 对数据库的操作，还是一种对数据库文件的操作，比如：

```
OPEN DATABASE MyDatabase        &&打开数据库 MyDatabase
USE MyTable       &&打开数据库中的 MyTable 表
GO 5              &&将 Cursor 跳到第 5 条记录
REPLACE MyName WITH “杨扬”  &&用“杨扬”替换第 5 条记录上的 MyName 字段
```

```
GO BOTTOM              &&将 Cursor 跳到最后一条记录
LOCATE FOR MyName==”杨扬”      &&查找并定位
IF FOUND()
    ? “FOUND!”
ELSE
    ? “NOT FOUND!”
ENDIF
USE       &&关闭 MyTable
CLOSE DATABASE MyDatabase       &&关闭数据库 MyDatabase
```

这段小程序可以说是 VFP 中比较简单的一段小程序了，但非常有代表性。从这段程序我们可以感觉到，在 VFP 想操纵一个数据库中的一个表需要许多类似文件的操作，比如：打开数据库、打开表、跳转 Cursor、读取字段内容、查找字段内容、关闭表、关闭数据库等等。这些操作虽然直观易懂，但十分不方便使用，而且如果同时有多个表打开，经常会出现表的轮换访问的问题，就需要不断的切换表的工作区，十分的麻烦和容易出错。最关键的是，这种操作方法不符合 OO 思想的精华——封装。

如果您熟悉 OO 编程思想，或者曾经有 OO 设计经验，您可能会和我一样这样想，如果一个数据库就是一个对象，所有的操作、信息都通过方法（Method）、属性（Attribure）、事件（Event）提供出来，供开发者使用，那该多好啊。C#正是借助基于这种思想设计的数据库访问技术 ADO.NET，并提供了一系列方便实用的类。应用这些数据库访问的类，您就可以轻松、准确而且是面向对象的操纵数据库中的各种数据了。

如图，这就是 C#中提供的数据库访问 ADO.NET 的结构图。

从这张图中，我们可以清楚的了解到 ADO.NET 的数据访问技术的架构。ADO.NET 支持 SQL Server 数据访问和 OLE DB 数据访问。两者相比，前者是针对 SQL Server 的数据库访问引擎，所以访问 SQL Server 数据库效率会高许多，但只支持 SQL Server。后者是比较通用的数据库访问引擎，可以支持广泛的数据库，但效率不如前者。对开发者来说，如果不用到某种数据库的特性，其大体使用方法是一致的。

上述内容指数据库的连接部分，也就是上图中的 Connetion 对象。Connection 对象提供

了与具体数据库的连接方式，具体你是用 SqlConnection 对象还是 OleDbConnection 对象，这个根据你的数据库类型由你选择而定，下面的叙述中，为了不占用过多的篇幅，在无特殊内容的地方，不再分开叙述。

下面给出两段典型的数据库连接的例子。在此之前，请在程序头部 using 处添加 using System.Data.SqlClient 或 System.Data.OleDb，以保证数据库访问时用到的命名空间能引用

SQL Server 数据访问

```
string strConn="Integrated Security=SSPI;Initial Catalog=MyDatabase;Data Source=YY-POWERPC ";
SqlConnection myConnection = new SqlConnection (strConn);
myConnection.Open();
```

OleDb 数据访问

```
String strConn="Provider=SQLOLEDB;Data Source=localhost;Initial Catalog=MyDatabase;Integrated Security=SSPI";
OleDbConnection myConnection=new OleDbConnection (strConn);
myConnection.Open();
```

上述步骤执行后，如果没有异常抛出，便可成功完成与数据库的连接。在完成了与数据库的连接后，接下来就是建立一个 DataAdapter 对象，来完成可访问数据库的工作。DataAdapter 的工作是后面 DataSet 的基础,其内容就是建立一个 DataSet 与数据库的中间层，来协调访问。由于 DataAdapter 与 DataSet 的关系十分紧密，我就结合在一起介绍了。DataAdapter 也分为 SqlDataAdapter 和 OleDbDataAdatper 两种。下面给出一段典型代码（OleDbDataAdapter 的情况可类推）：

```
SqlDataAdapter myDataAdapter = new SqlDataAdapter ();
DataSet myDataSet = new DataSet ();
string strCom = "SELECT * FROM 会员信息表";
myDataAdapter.SelectCommand = new SqlCommand (strCom,myConnection);
SqlCommandBuilder myCB = new SqlCommandBuilder (myDataAdapter);
myDataAdapter.Fill (myDataSet,"会员信息表");
```

这段代码用到了 SqlDataAdapter、DataSet、SqlCommand、SqlCommandBuilder 四个对象。从上图中可以看出，SqlDataAdapter 的作用就是负责与数据库的通讯访问，同时与 DataSet 相连，它的内部有四个很重要的 Command 对象（同样分为 SqlCommand 和 OleDbCommand），都是访问数据库必用的，分别为 SelectCommand、InsertCommand、UpdateCommand、DeleteCommand 对象。这些 Command 对象便是专门用来完成对数据库的查询、插入、更新、删除操作，它们就像四个大臣，在 DataAdapter 的控制下分别主管各自的事情。其中 SelectCommand 是它们四个中的老大，由它可以自动的构造生成另外的三个。构造生成的过程就是应用 CommandBuilder。在这之前，我们只需要指定 DataAdapter 中的 SelectCommand 对象，就可以了。

在设置好了 SQL Select 语句后，就可以开始填充相应的数据集了。方法是应用 DataAdapter 的 Fill 方法，参数为 DataSet 及其中的某个 DataTable。这里要着重讲一讲 DataSet 对象。如果你用过 ADO 中的 Recordset 对象，你可能会感觉到 DataSet 和 Recordset 的差别。Recordset 一般只能应用于单表，即一个 Recordset 对应于一张表。而 DataSet 中有一个 DataTableCollection，即一个 DataTable 集合，可以包含多个 DataTable 对象。DataTable 对象看上去就更加像一张表了，其中有 DataRowCollection、DataColumnCollection、ConstraintCollection。它们分别代表 DataRow（数据行）、DataColumn（数据列）、Constraint（约束关系）的对象集合。可能说这么多，你已经有点儿迷糊了，先看一句代码吧。

```
myDataSet.Tables["MyTable"].Rows[3]["MyName"] = "杨扬";
```

怎么样？是不是一下子就明白了许多。这是一句多么完美的 OO 思想表达出的语句啊！这句就是将 DataSet 下的一个名为 MyTable 的“虚表”中的第 4 行的 MyName 字段的内容改为“杨扬”。为什么叫“虚表”呢？这是因为 ADO.NET 的一个特点就是脱机连接数据库。这样可以减少网络通讯的压力，提高效率。你可能会问 DataColumn 在哪呢？Rows[3]就表示了第 4 行记录，换句话说 Rows[3]就是一个 DataRow 对象，一加上[“MyName”]就自动定位到了 MyName 字段的内容。ADO.NET 规定，访问表中内容必须是先行后列的原则，Column[“MyName”][3]是不允许的。当然，DataColumn 也是什么重要的，比如想查看某列的列头（Field），可以用 Column[1].ColumnName 更改。

好了，现在我觉得您应该大体上明白 C#+ADO.NET 是如何操纵数据库的了吧，其实这里只是讲了一小部分，因为 ADO.NET 数据库访问技术包含的内容太多了，不是一两遍文章

就可以说清的。还有许多许多有用的操作，比如添加、修改、删除、更新、查询等等都还没有介绍。当然，我会陆续推出数据库编程的后继文章，教初学者快速出门。如果您有什么好的意见和建议，也可以发 Email 给我:compking@21cn.com。谢谢各位，下回见：）

C#下数据库编程

（二）

正文：

上次我们留下了些具体问题没有解决，比如：如何向数据库添加一条记录，如何修改或删除一条记录，等等。这些问题相信所有初学 C#+ADO.NET 编程的人都会遇到，而且开发 MIS 系统时这些也都是必须要解决的问题。下面我来用几个实例，来说明具体的实现思路和方法。

首先，我们先来添加一条记录。首先要明确一个思想，那就是我在第一篇中说过的，在 C#中没有类似 Visual Foxpro 中的文件式数据库访问技术，如：

USE MyTable

APPEND BLANK

REPLACE MyName WITH “杨扬”

USE

在 C#中，一切操作都要 OO，即以操纵对象的方式来完成。这个思想一定要贯穿整个 C#编程过程中。那么如何来添加记录呢？如果你还记得上篇那幅插图的话，就会记得图中有一个 DataSet 对象其中包括的 DataRow。我在上篇也提到过，DataRow 就是 DataSet 中 DataTable 对象的一个子对象，代表一个数据行对象。而 DataTable 对象就是由一些 DataRow 对象组成的 DataRowCollection 对象而构成的。我们添加一条记录，其实就是添加一个 DataRow 对象到已有的 DataTable 对象的 DataRowCollection 中。明白了这个思路，我们就来看看具体的代码该如何编制了。

先复习一下以前这段代码，这样我们不会感觉不知所措：

```
SqlDataAdapter myDataAdapter = new SqlDataAdapter ();

DataSet myDataSet = new DataSet ();

string strCom = "SELECT * FROM myTable";

myDataAdapter.SelectCommand = new SqlCommand (strCom,myConnection);

SqlCommandBuilder myCB = new SqlCommandBuilder (myDataAdapter);
```

```
myDataAdapter.Fill (myDataSet,"myTable");
```

这时我们已经得到了一个填充好的 myDataSet，其中有一个 DataTable 对象叫 myTable。然后我们通过 DataTable 对象的 NewRow 方法构造一个新的 DataRow 对象，在完成设定并赋值后，由 DataTable.Rows（即 DataRowCollection）的 Add 方法来完成添加。

```
DataRow myDataRow;     //定义一个 DataRow

DataTable myDataTable;//定义一个 DataTable

myDataTable = myDataSet.Tables["myTable"];       //引用 DataSet 中的一个
DataTable

myDataRow = myDataTable.NewRow();        //调用 NewRow 方法得到一个 DataRow

myDataRow["myName"] = "杨扬";    //将此 DataRow 中的 myName 字段置为“杨扬”

myDataTable.Rows.Add(myDataRow);     //将此 DataRow 添加到 myDataTable 中
```

怎么样？是不是感觉非常直观，是不是有种很清爽的感觉。的确，OO 的设计方法在开始时会有些不适应，但只要你领悟了其中的思想，你就会感觉一切操作都很简单，不用过多的考虑细节。不过要说明的是，你用的上述方法添加完成后，记录并没有进入到真正的数据库中，而只是在 DataSet 中完成了添加。如果想要在数据库中真正添加，还要应用 DataAdapter 的 Update 方法将 DataSet 回写到数据库中。这种方法虽然没有直接操作文件显得快捷，但却是现在技术发展的趋势和必然要求，因为现在的数据库系统几乎都是面向网络环境的，特别是分布式的系统更是对数据库访问技术提出了更高的要求。以往的那种独享操纵数据库的方式是绝对不能适合的。ADO.NET 访问数据库的方法就是数据库连接成功后，将数据库中表信息的子集创建一个 DataSet，然后建议断开数据库连接。在完成对数据库的一系列操作后，再将 DataSet 回写到数据库的相应表中。这种方式有许多优点，比如减少网络通讯压力，便于事务处理，提高访问效率等等，但缺点就是必须多增加一步回写操作。不过我相信与众多优点相比，这点也算不上什么。

接下来，我们来看看修改操作的实现。有了上面添加操作的方法，修改操作我想也就容易多了，即直接对 DataTable 对象中的某个 DataRow 对象进行修改。请看下面的代码。

```
DataRow myDataRow;     //定义一个 DataRow

DataTable myDataTable;//定义一个 DataTable

myDataTable = myDataSet.Tables["myTable"];       //引用 DataSet 中的一个 DataTable

myDataRow = myDataTable.Rows[1];         //得到一个要修改的 DataRow

myDataRow["myName"] = "杨扬";   //将此 DataRow 中的 myName 字段置为“杨扬”
```

有了上面的基础，我觉得删除操作读者都可以自己无师自通了。先自己想想该如何做，我马上就告诉大家。说对了，就是应用 DataTable 对象的 DataRow 子对象的 Delete 方法。比如删除第 1 条记录，代码如下：

```
DataRow myDataRow;     //定义一个 DataRow

DataTable myDataTable;//定义一个 DataTable

myDataTable = myDataSet.Tables["myTable"];       //引用 DataSet 中的一个 DataTable

myDataRow = myDataTable.Rows[1];     //得到一个要删除的 DataRow

myDataRow.Delete();
```

DataRow 的 Delete 方法不像 Visual Foxpro 中的 DELETE 命令，只做一个删除标记，还要配合 PACK 才能真正删除记录。DataRow 的 Delete 方法是一次性删除，当然只是删除了 DataTable 中的某条记录，还要 Update 的。

上面总说要应用 Update 将 DataTable 回写到数据库的表中，可具体做法是什么呢？下面代码给出具体的方法。

```
myDataAdapter.Update(myDataSet,"myTable");
```

好了，现在可以说您已经学会了 C#+ADO.NET 数据库编程技术的添加、修改、删除、更新操作了，加上您以前学过的 OO 设计方法和可视化开发方法，用 Visual C#.NET 可以编写简单的数据库小程序了，是不是觉得特兴奋。当然仅仅这些操作只能编写一个数据库范例，真正的数据库开发要比这复杂的多。但是什么复杂的事物都是由简单的构成的。下一篇我还要教大家更兴奋的内容——查询操作，以后还会教一些更高级数据库编程。