

# 第15章 ADO.NET数据库访问技术

15.1 数据库概述

15.2 ADO.NET模型

15.3 ADO.NET的数据访问对象

15.4 DataSet对象

15.5 数据绑定

15.6 DataView对象

15.7 DataGridView控件

# 15.1 数据库概述

## 15.1.1 关系数据库的基本结构

1. 表
2. 记录
3. 字段
4. 关系
5. 索引
6. 视图
7. 存储过程

## 样本Access数据库：School.accdb

student表

学号	姓名	性别	民族	班号
1	王华	女	汉族	07001
3	李兵	男	汉族	07001
8	马棋	男	回族	07002
2	孙丽	女	满族	07002
6	张军	男	汉族	07001

## score表

学号	课程名	分数
1	C语言	80
3	C语言	76
8	C语言	88
2	C语言	70
6	C语言	90
1	数据结构	83
3	数据结构	70
8	数据结构	79
2	数据结构	52
6	数据结构	92

## 15.1.2 结构化查询语言 (SQL)

数据定义语句：CREATE

数据操纵语句：INSERT、UPDATE、DELETE

数据查询语句：SELECT

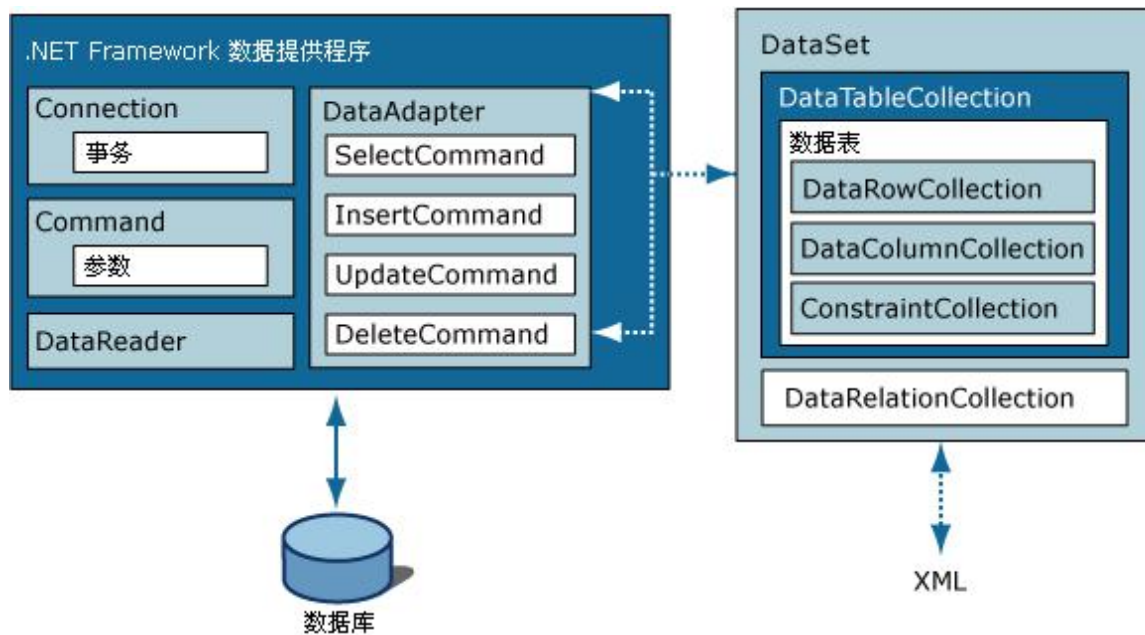
## 15.2 ADO.NET模型

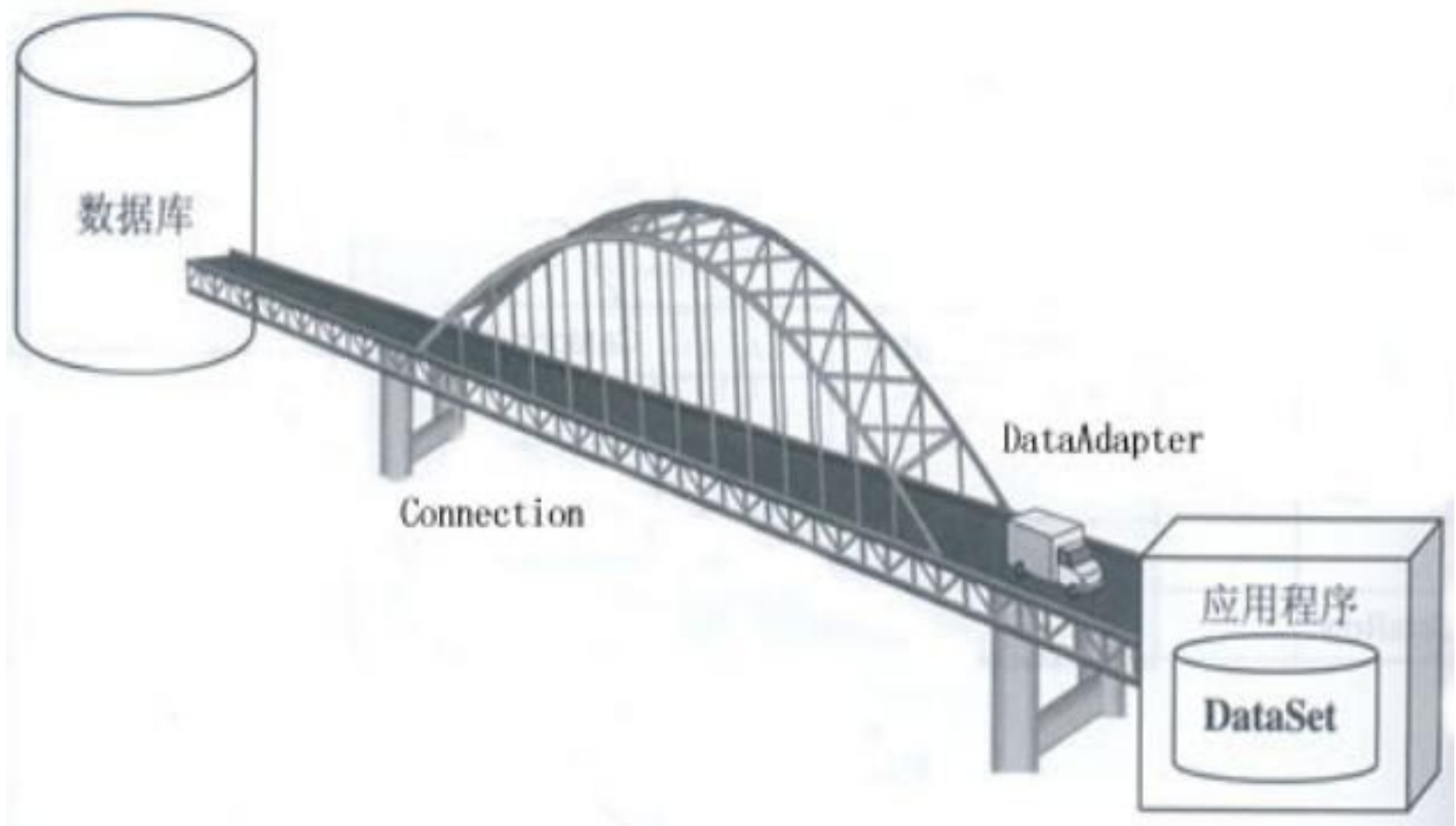
### 15.2.1 ADO.NET简介

ADO.NET是在.NET Framework上访问数据库的一组类库，它利用.NET Data Provider（数据提供程序）以进行数据库的连接与访问。

通过ADO.NET，数据库程序设计人员能够很轻易地使用各种对象来访问符合自己需求的数据库内容。

## 15.2.2 ADO.NET体系结构







## 1. .NET Data Provider

**.NET Data Provider**是指访问数据源的一组类库，主要是为了统一对于各类型数据源的访问方式而设计的一套高效能的类数据库。

下表给出了.NET Data Provider中包含的4个对象。

对象名称	功能说明
Connection	提供和数据源的连接功能。
Command	提供运行访问数据库命令，传送数据或修改数据的功能，例如运行SQL命令和存储过程等。
DataAdapter	是DataSet对象和数据源间的桥梁。DataAdapter使用4个Command对象来运行查询、新建、修改、删除的SQL命令，把数据加载到DataSet，或者把DataSet内的数据送回数据源。
DataReader	通过Command对象运行SQL查询命令取得数据流，以便进行高速、只读的数据浏览。

---

在.NET Framework中常用的有如下4组数据提供程序：

- (1) SQL.NET Data Provider
  - (2) OLEDB.NET Data Provider
  - (3) ODBC.NET Data Provider
  - (4) ORACLE.NET Data Provider
-

## 2. DataSet

DataSet（数据集）是ADO.NET离线数据访问模型中的核心对象，主要使用时机是在内存中暂存并处理各种从数据源中所取回的数据。

DataSet其实就是一个存放在内存中的数据暂存区，这些数据必须通过DataAdapter对象与数据库进行数据交换。在DataSet内部允许同时存放一个或多个不同的数据表（DataTable）对象。

这些数据表是由数据列和数据域所组成的，并包含有主索引键、外部索引键、数据表间的关系（Relation）信息以及数据格式的条件限制（Constraint）。

## 15.2.3 ADO.NET数据库的访问流程

ADO.NET数据库访问的一般流程如下：

- (1) 建立Connection对象，创建一个数据库连接。
- (2) 在建立连接的基础上可以使用Command对象对数据库发送查询、新增、修改和删除等命令。
- (3) 创建DataAdapter对象，从数据库中取得数据。
- (4) 创建DataSet对象，将DataAdapter对象填充到DataSet对象（数据集）中。
- (5) 如果需要，可以重复操作，一个DataSet对象可以容纳多个数据集。
- (6) 关闭数据库。
- (7) 在DataSet上进行所需要的操作。数据集的数据要输出到窗体中或者网页上面，需要设定数据显示控件的数据源为数据集。

## 15.3 ADO.NET的数据访问对象

### 15.3.1 OleDbConnection对象

在数据访问中首先必须是建立数据库的物理连接。

.NET Data Provider使用OleDbConnection类的对象标识与一个数据库的物理连接。

## 1. OleDbConnection类

OleDbConnection类的属性	说 明
ConnectionString	获取或设置用于打开数据库的字符串。
ConnectionTimeout	获取在尝试建立连接时终止尝试并生成错误之前所等待的时间。
Database	获取当前数据库或连接打开后要使用的数据库的名称。
DataSource	获取数据源的服务器名或文件名。
Provider	获取在连接字符串的“Provider=”子句中指定的OLEDB提供程序的名称。
State	获取连接的当前状态。其取值及其说明如表15.7所示。

OleDbConnection类的方法	说 明
Open	使用 ConnectionString 所指定的属性设置打开数据库连接。
Close	关闭与数据库的连接。这是关闭任何打开连接的首选方法。
CreateCommand	创建并返回一个与 OleDbConnection 关联的 OleDbCommand 对象。
ChangeDatabase	为打开的OleDbConnection 更改当前数据库。

---

## 2. 建立连接字符串ConnectionString

建立连接的核心是建立连接字符串ConnectionString。建立连接主要有两种方法。

### (1) 直接建立连接字符串

直接建立连接字符串的方式是：先创建一个OleDbConnection对象，将其ConnectionString属性设置为如下值：

**Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#程序  
\ch15\school.accdb**

其中Provider和Data Source是必选项，如果Access数据库没有密码，后两者都可以省略。由Access数据库是基于文件的数据库，因此在实际项目中应该将Data Source属性值转化为服务器的绝对路径。

最后用Open方法打开连接。

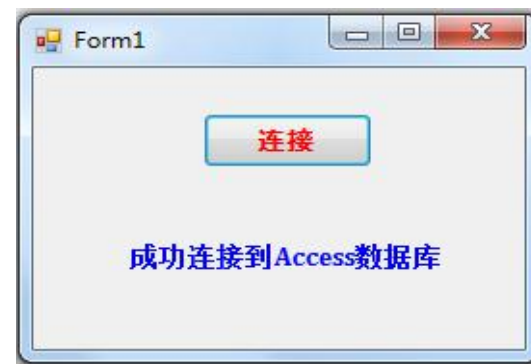
---



**【例15.10】** 设计一个窗体，说明直接建立连接字符串的连接过程。

**Form1, 事件过程:**

```
private void button1_Click(object sender, EventArgs e)
{
    string mystr;
    OleDbConnection myconn = new OleDbConnection();
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#
程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
    if (myconn.State == ConnectionState.Open)
        label1.Text = "成功连接到Access数据库";
    else
        label1.Text = "不能连接到Access数据库";
    myconn.Close();
}
```



## (2) 通过属性窗口建立连接字符串

① 先要在窗体上放置一个OleDbConnection控件。

② 在“属性”窗口中单击OleDbConnection控件的ConnectionString属性右侧的  按钮，从弹出的下拉列表中选择“新建连接”选项，打开“添加连接”对话框，操作如下。

添加连接

输入信息以连接到选定的数据源，或单击“更改”选择另一个数据源和/或提供程序。

数据源(S):  
Microsoft SQL Server (OLE DB) 更改(C)...

服务器名(E):  
刷新(R)

登录到服务器

☒ 使用 Windows 身份验证(W)  
☐ 使用 SQL Server 身份验证(Q)

用户名(U):  
密码(P):  
☐ 保存密码(S)

连接到数据库

☒ 选择或输入数据库名称(D):  
附加数据库文件(H):  
逻辑名(L):  
高级(V)...

测试连接(T) 确定 取消

更改数据源

数据源(S):  
Microsoft Access 数据库文件  
Microsoft SQL Server  
Oracle 数据库  
<其他>

说明  
使用此选择通过用于 OLE DB 的 .NET Framework 数据提供程序连接到 Microsoft Access 数据库文件。

数据提供程序(P):  
用于 OLE DB 的 .NET Framework 数据提供程序

☐ 始终使用此选择(U)

确定 取消

添加连接

输入信息以连接到选定的数据源，或单击“更改”选择另一个数据源和/或提供程序。

数据源(S):

Microsoft Access 数据库文件 (OLE DB) 更改(C)...

数据库文件名(D):

D:\C#程序\ch15\school.accdb 浏览(B)...

登录到数据库

用户名(U): Admin

密码(P):

☐ 保存密码(S)

高级(V)...

测试连接(T) 确定 取消

③ 单击“测试连接”按钮确定连接是否成功。在测试成功后单击“确定”按钮返回。此时，ConnectionString属性值改为：

**Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#程序  
\ch15\school.accdb**

**两种方法建立的连接字符串是相同的。**

**【例15.11】** 设计一个窗体，说明通过属性窗口建立连接字符串的连接过程。

Form2：有一个命令按钮button1、一个标签label1和一个OleDbConnection控件oleDbConnection1（已建连接）。

事件过程：

```
private void button1_Click(object sender, EventArgs e)
{
    oleDbConnection1.Open();
    if (oleDbConnection1.State == ConnectionState.Open)
        label1.Text = "成功连接到Access数据库";
    else
        label1.Text = "不能连接到Access数据库";
    oleDbConnection1.Close();
}
```

## 15.3.2 OleDbCommand对象

建立数据连接之后，就可以执行数据访问操作和数据操纵操作了。一般对数据库的操作被概括为**CRUD**—**Create**、**Read**、**Update**和**Delete**。在**ADO.NET**中定义**OleDbCommand**类去执行这些操作。

OleDbCommand类的属性	说 明
CommandText	获取或设置要对数据源执行的 T-SQL 语句或存储过程。
CommandTimeout	获取或设置在终止执行命令的尝试并生成错误之前的等待时间。
CommandType	获取或设置一个值，该值指示如何解释 CommandText 属性。其取值如表15.10所示。
Connection	数据命令对象所使用的连接对象
Parameters	参数集合（OleDbParameterCollection）



OleDbCommand类的方法	说 明
CreateParameter	创建OleDbParameter对象的新实例。
ExecuteNonQuery	针对Connection 执行SQL语句并返回受影响的行数。
ExecuteReader	将CommandText发送到Connection并生成一个OleDbDataReader。
ExecuteScalar	执行查询，并返回查询所返回的结果集中第一行的第一列。忽略其他列或行。

## 2. 创建OleDbCommand对象

OleDbCommand类的主要构造函数如下：

OleDbCommand();

OleDbCommand(cmdText) ;

OleDbCommand(cmdText, connection);

其中，cmdText参数指定查询的文本。connection参数是一个OleDbConnection，它表示到Access数据库的连接。

例如，以下语句创建一个OleDbCommand对象mycmd:

```
OleDbConnection myconn = new OleDbConnection();
```

```
mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#  
程序\ch15\school.accdb";
```

```
myconn.ConnectionString = mystr;
```

```
myconn.Open();
```

```
OleDbCommand mycmd = new OleDbCommand("SELECT * FROM  
student",myconn);
```

### 3. 通过OleDbCommand对象返回单个值

在OleDbCommand的方法中，ExecuteScalar方法执行返回单个值的SQL命令。

例如，如果想获取Student数据库中学生的总人数，则可以使用这个方法执行SQL查询**SELECT Count(\*) FROM student。**

示例：

```
string mystr,mysql;  
OleDbConnection myconn = new OleDbConnection();  
OleDbCommand mycmd = new OleDbCommand();  
mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#程序\ch15\school.accdb";  
myconn.ConnectionString = mystr;  
myconn.Open();  
mysql = "SELECT AVG(分数) FROM score";  
mycmd.CommandText = mysql;  
mycmd.Connection = myconn;  
textBox1.Text = mycmd.ExecuteScalar().ToString();  
myconn.Close();
```



## 4. 通过OleDbCommand对象执行修改操作

在OleDbCommand的方法中，ExecuteNonQuery方法执行不返回结果的SQL命令。

该方法主要用来更新数据，通常使用它来执行UPDATE、INSERT和DELETE语句。

该方法不返回行，对于UPDATE、INSERT和DELETE语句，返回值为该命令所影响的行数，对于所有其他类型的语句，返回值为-1。

**【例15.13】** 设计一个窗体，通过OleDbCommand对象将score表中所有分数增5分和减5分。

Form4, 设计界面

事件过程:

```
private void Form4_Load(object sender, EventArgs e)
{
    string mystr;
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\C#程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
}
```



```
private void button1_Click(object sender, EventArgs e)
{
    string mysql;
    mysql = "UPDATE score SET 分数=分数+5";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    mycmd.ExecuteNonQuery();
}

private void button2_Click(object sender, EventArgs e)
{
    string mysql;
    mysql = "UPDATE score SET 分数=分数-5";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    mycmd.ExecuteNonQuery();
}
```





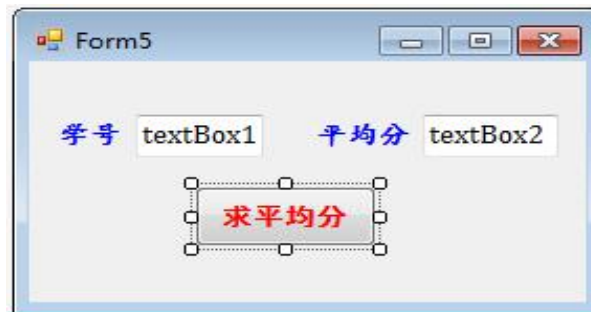
## 5. 在OleDbCommand对象的命令中指定参数

OleDb.NET Data Provider支持执行命令中包含参数的情况，也就是说，可以使用包含参数的数据命令或存储过程执行数据筛选操作和数据更新等操作，其主要流程如下：

- (1) 创建Connection对象，并设置相应的属性值。
- (2) 打开Connection对象。
- (3) 创建Command对象并设置相应的属性值，其中SQL语句含有占位符。
- (4) 创建参数对象，将建立好的参数对象添加到Command对象的Parameters集合中。
- (5) 为参数对象赋值。
- (6) 执行数据命令。
- (7) 关闭相关对象。

**【例15.14】** 设计一个窗体，通过OleDbCommand对象  
求出指定学号学生的平均分。

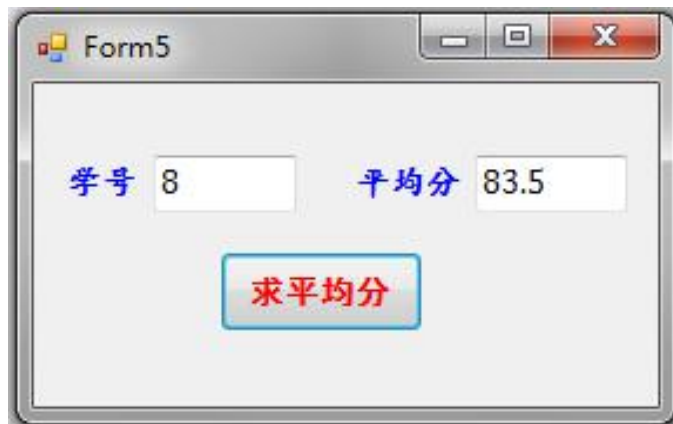
Form5，设计界面



事件过程：

```
private void button1_Click(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    OleDbCommand mycmd = new OleDbCommand();
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\C#程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
}
```

```
mysql = "SELECT AVG(分数) FROM score WHERE 学号=@no";  
mycmd.CommandText = mysql;  
mycmd.Connection = myconn;  
mycmd.Parameters.Add("@no", OleDbType.VarChar, 5).Value =  
    textBox1.Text; //设置参数值  
textBox2.Text = mycmd.ExecuteScalar().ToString();  
myconn.Close();  
}
```



### 15.3.3 DataReader对象

当执行返回结果集的命令时，需要一个方法从结果集中提取数据。处理结果集的方法有两个：

- (1) 使用DataReader对象（数据阅读器）；
- (2) 同时使用DataAdapter对象（数据适配器）和ADO.NET DataSet。

使用DataReader对象可以从数据库中得到只读的、只能向前的数据流。使用DataReader对象还可以提高应用程序的性能，减少系统开销，因为同一时间只有一条行记录在内存中。

## 1. DataReader类的属性和方法

DataReader类的属性	说 明
FieldCount	获取当前行中的列数
IsClosed	获取一个布尔值，指出DataReader对象是否关闭
RecordsAffected	获取执行SQL语句时修改的行数

<b>DataReader类的方法</b>	<b>说明</b>
<b>Read</b>	将DataReader对象前进到下一行并读取，返回布尔值指示是否有多行。
<b>Close</b>	关闭DataReader对象。
<b>IsDBNull</b>	返回布尔值，表示列是否包含NULL值。
<b>NextResult</b>	将DataReader对象移到下一个结果集，返回布尔值指示该结果集是否有多行。
<b>GetBoolean</b>	返回指定列的值，类型为布尔值。
<b>GetString</b>	返回指定列的值，类型为字符串。
<b>GetByte</b>	返回指定列的值，类型为字节。
<b>GetInt32</b>	返回指定列的值，类型为整型值。
<b>GetDouble</b>	返回指定列的值，类型为双精度值。
<b>GetDateTime</b>	返回指定列的值，类型为日期时间值。
<b>GetOrdinal</b>	返回指定列的序号或数字位置（首为0）。
<b>GetBoolean</b>	返回指定列的值，类型为对象。

## 2. 创建DataReader对象

DataReader类没有提供公有的构造函数。通常调用Command类的ExecuteReader方法，这个方法将返回一个DataReader对象。

例如，以下代码创建一个myreader对象：

```
OleDbCommand cmd = new OleDbCommand(CommandText,  
    ConnectionObject);
```

```
OleDbDataReader myreader = cmd.ExecuteReader();
```

**注意：** OleDbDataReader对象不能使用new来创建。

### 3. 遍历 OleDbDataReader 对象的记录

使用 While 循环来遍历记录：

```
while (myreader.Read())  
{  
    //读取数据  
}
```



## 4. 访问字段中的值

使用以下语句获取一个OleDbDataReader对象：

```
OleDbDataReader myreader = mycmd.ExecuteReader();
```

### (1) Item属性

每一个DataReader对象都定义了一个Item属性，此属性返回一个代码字段属性的对象。Item属性是DataReader对象的索引。需要注意的是Item属性总是基于0开始编号的：

```
myreader[FieldName]
```

```
myreader[FieldIndex]
```

### (2) Get方法

每一个DataReader对象都定义了一组Get方法，那些方法将返回适当类型的值。例如：

```
myreader.GetInt32[0]    //第1个字段值
```

```
myreader.GetString[1]   //第2个字段值
```

**【例15.15】** 设计一个窗体，通过OleDbDataReader对象输出所有学生记录。

Form6, 设计界面

事件过程:



```
private void button1_Click(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    OleDbCommand mycmd = new OleDbCommand();
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\C#程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM student";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
```

```

OleDbDataReader myreader = mycmd.ExecuteReader();
listBox1.Items.Add("学号\t姓名\t性别\t民族\t班号");
listBox1.Items.Add("=====");
//循环读取信息
while (myreader.Read())
    listBox1.Items.Add(String.Format("{0}\t{1}\t{2}\t{3}\t{4}",
        myreader[0].ToString(), myreader[1].ToString(),
        myreader[2].ToString(), myreader[3].ToString(),
        myreader[4].ToString()));
myconn.Close();
myreader.Close();
}

```

学号	姓名	性别	民族	班号
1	王华	女	汉族	07001
3	李兵	男	汉族	07001
8	马棋	男	回族	07002
2	孙丽	女	满族	07002
6	张军	男	汉族	07001

输出所有学生记录

### 15.3.4 OleDbDataAdapter对象

OleDbDataAdapter对象（数据适配器）可以执行SQL命令以及调用存储过程、传递参数，最重要的是取得数据结果集，在数据库和DataSet对象之间来回传输数据。

OleDbDataAdapter类的属性	说明
SelectCommand	获取或设置SQL语句或存储过程，用于选择数据源中的记录。
InsertCommand	获取或设置SQL语句或存储过程，用于将新记录插入到数据源中。
UpdateCommand	获取或设置SQL语句或存储过程，用于更新数据源中的记录。
DeleteCommand	获取或设置SQL语句或存储过程，用于从数据集中删除记录。
AcceptChangesDuringFill	获取或设置一个值，该值指示在任何Fill操作过程中，是否接受对行所做的修改。
AcceptChangesDuringUpdate	获取或设置在Update期间是否调用AcceptChanges。
FillLoadOption	获取或设置LoadOption，后者确定适配器如何从DbDataReader中填充DataTable。
MissingMappingAction	确定传入数据没有匹配的表或列时需要执行的操作。
MissingSchemaAction	确定现有DataSet架构与传入数据不匹配时需要执行的操作。
TableMappings	获取一个集合，它提供源表和DataTable之间的映射。

OleDbDataAdapter类的方法	说明
Fill	用来自动执行OleDbDataAdapter对象的SelectCommand属性中相对应的SQL语句，以检索数据库中的数据，然后更新数据集中的DataTable对象，如果DataTable对象不存在，则创建它。
FillSchema	将DataTable添加到DataSet中，并配置架构以匹配数据源中的架构。
GetFillParameters	获取当执行SQL SELECT语句时由用户设置的参数。
Update	用来自动执行UpdateCommand、InsertCommand或删除Command属性相对应的SQL语句，以使数据集中的数据来更新数据库。

## 2. 创建OleDbDataAdapter对象

创建OleDbDataAdapter对象有两种方式。

### (1) 用程序代码创建OleDbDataAdapter对象

OleDbDataAdapter类有以下构造函数：

`OleDbDataAdapter();`

`OleDbDataAdapter(selectCommandText);`

`OleDbDataAdapter(selectCommandText,selectConnection);`

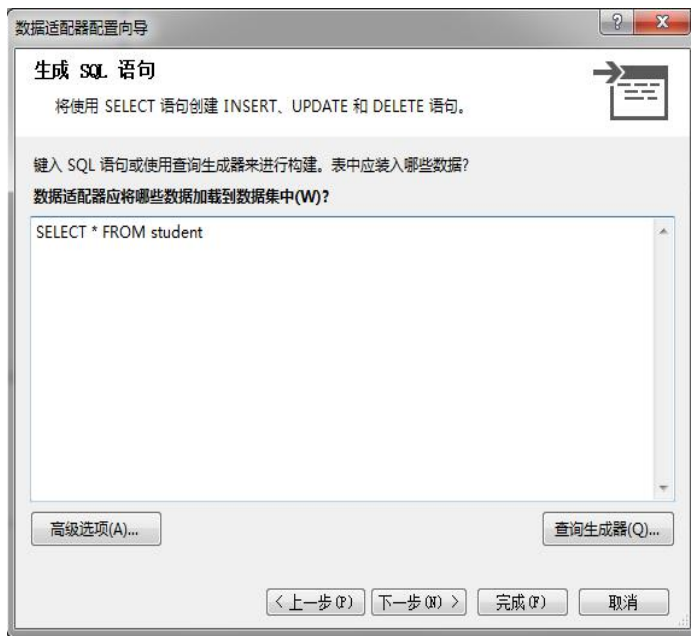
`OleDbDataAdapter((selectCommandText,selectConnectionString);`

## (2) 通过设计工具创建OleDbDataAdapter对象

从工具箱中的“数据”选项卡中选取OleDbDataAdapter并拖放到窗体中，这时会出现数据适配器配置向导。操作如下：







### 3. 使用Fill方法

Fill方法用于向DataSet对象填充从数据源中读取的数据。调用Fill方法的语法格式有多种，常见的格式如下：

OleDbDataAdapter对象名.Fill(DataSet对象名, "数据表名");

其中第一个参数是数据集对象名，表示要填充的数据集对象；第二个参数是一个字符串，表示在本地缓冲区中建立的临时表的名称。

例如，以下语句用course表数据填充数据集mydataset1：

OleDbDataAdapter1.Fill(mydataset1,"course");



DataSet对象

## 4. 使用Update方法

Update方法用于将数据集DataSet对象中的数据按InsertCommand属性、DeleteCommand属性和UpdateCommand属性所指定的要求更新数据源，即调用3个属性中所定义的SQL语句来更新数据源。

OleDbDataAdapter对象名.Update(DataSet对象名,[数据表名]);

例如，以下语句创建一个OleDbCommandBuilder对象mycmdbuilder，用于产生myadp对象的InsertCommand、DeleteCommand和UpdateCommand属性值，然后调用Update方法执行这些修改命令以更新数据源：

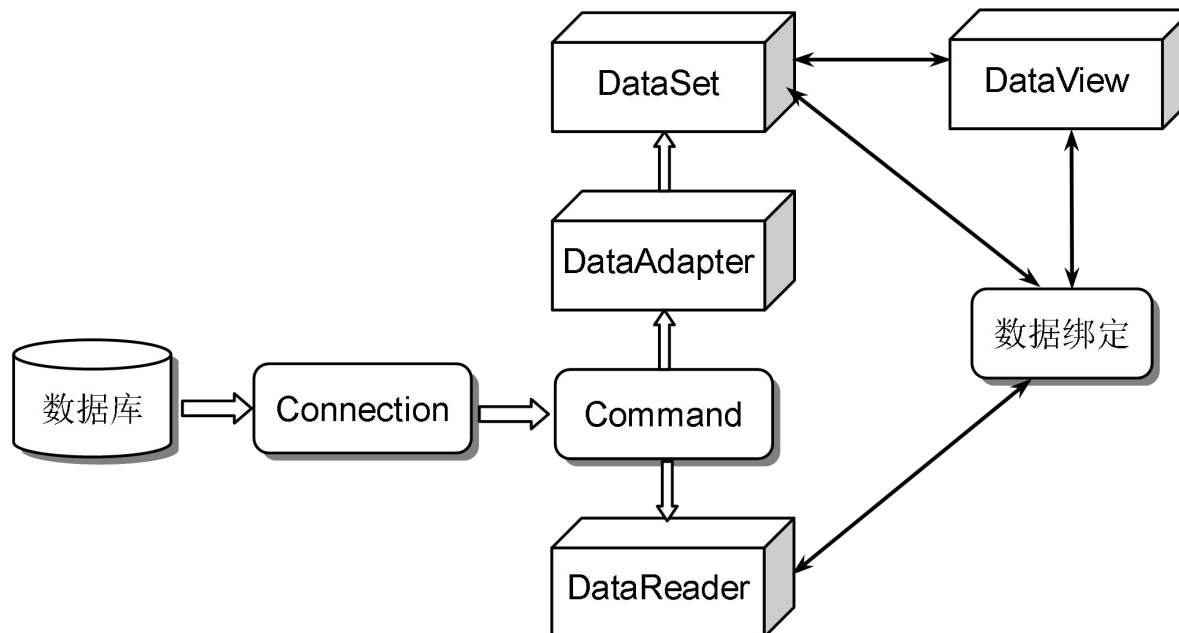
```
OleDbCommandBuilder mycmdbuilder = new  
    OleDbCommandBuilder(myadp);  
myadp.Update(myds, "student");
```



较少使用

## 15.4 DataSet对象

### 15.4.1 DataSet对象概述



## 15. 4. 2 Dataset对象的属性和方法

Dataset对象的属性	说 明
CaseSensitive	获取或设置一个值，该值指示DataTable对象中的字符串比较是否区分大小写。
DataSetName	获取或设置当前DataSet的名称。
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。
Tables	获取包含在DataSet中的表的集合。

<b>DataSet对象的方法</b>	<b>说 明</b>
<b>AcceptChanges</b>	提交自加载此DataSet或上次调用AcceptChanges以来对其进行的所有更改。
<b>Clear</b>	通过移除所有表中的所有行来清除任何数据的DataSet。
<b>CreateDataReader</b>	为每个DataTable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。
<b>GetChanges</b>	获取DataSet的副本，该副本包含自上次加载以来或自调用AcceptChanges以来对该数据集进行的所有更改。
<b>HasChanges</b>	获取一个值，该值指示DataSet是否有更改，包括新增行、已删除的行或已修改的行。
<b>Merge</b>	将指定的DataSet、DataTable或DataRow对象的数组合并到当前的DataSet或DataTable中。
<b>Reset</b>	将DataSet重置为其初始状态。

### 15.4.3 Tables集合和DataTable对象

DataSet对象的Tables属性由表组成，每个表是一个DataTable对象。

Tables集合的属性	说 明
Count	Tables集合中表个数。
Item	检索Tables集合中指定索引处的表。



Tables集合的方法	说 明
Add	向Tables集合中添加一个表。
AddRange	向Tables集合中添加一个表的数组。
Clear	移除Tables集合中的所有表。
Contains	确定指定表是否在Tables集合中。
Equals	判断是否等于当前对象。
GetType	获取当前实例的Type。
Insert	将一个表插入到Tables集合中指定的索引处。
IndexOf	检索指定的表在Tables集合中的索引。
Remove	从Tables集合中移除指定的表。
RemoveAt	移除Tables集合中指定索引处的表

## 2. DataTable对象

DataSet对象由若干个DataTable对象组成，可以使用

`DataSet.Tables[“表名”]`

或

`DataSet.Tables[“表索引”]`

来引用其中的DataTable对象。

<b>DataTable对象的属性</b>	<b>说 明</b>
<b>CaseSensitive</b>	指示表中的字符串比较是否区分大小写。
<b>ChildRelations</b>	获取此DataTable的子关系的集合。
<b>Columns</b>	获取属于该表的列的集合。
<b>Constraints</b>	获取由该表维护的约束的集合。
<b>DataSet</b>	获取此表所属的DataSet。
<b>DefaultView</b>	返回可用于排序、筛选和搜索DataTable的DataView。
<b>ExtendedProperties</b>	获取自定义用户信息的集合。
<b>ParentRelations</b>	获取该DataTable的父关系的集合。
<b>PrimaryKey</b>	获取或设置充当数据表主键的列的数组。
<b>Rows</b>	获取属于该表的行的集合。
<b>TableName</b>	获取或设置DataTable的名称。

<b>DataTable对象的方法</b>	<b>说 明</b>
<b>AcceptChanges</b>	提交自上次调用AcceptChanges以来对该表进行的所有更改。
<b>Clear</b>	清除所有数据的DataTable。
<b>Compute</b>	计算用来传递筛选条件的当前行上的给定表达式。
<b>CreateDataReader</b>	返回与此DataTable中的数据相对应的DataTableReader。
<b>ImportRow</b>	将DataRow复制到DataTable中，保留任何属性设置以及初始值和当前值。
<b>Merge</b>	将指定的DataTable与当前的DataTable合并。
<b>NewRow</b>	创建与该表具有相同架构的新DataRow。
<b>Select</b>	获取DataRow对象的数组。

### 3. 建立包含在数据集中的表

建立包含在数据集中的表的方法主要有以下两种。

#### (1) 利用Fill方法自动建立DataSet中的DataTable对象

先通过OleDbDataAdapter对象从数据源中提取记录数据，然后调用其Fill方法，将所提取的记录存入DataSet中对应的表内，如果DataSet中不存在对应的表，Fill方法会先建立表再将记录填入其中。

例如，以下语句向DataSet对象myds中添加一个表course及其包含的数据记录：

```
DataSet myds = new DataSet();
```

```
OleDbDataAdapter myda =
```

```
new OleDbDataAdapter("SELECT * From course",myconn);
```

```
myda.Fill(myds, "course");
```

## (2) 将建立的DataTable对象添加到DataSet中

先建立DataTable对象，然后调用DataSet的表集合属性Tables的Add方法，将DataTable对象添加到DataSet对象中。

例如，以下语句向DataSet对象myds中添加一个表，并返回表的名称course：

```
DataSet myds = new DataSet();  
DataTable mydt = new DataTable("course");  
myds.Tables.Add(mydt);  
textBox1.Text = myds.Tables["course"].TableName;  
//文本框中显示 “course”
```

---

以下内容自学：

- ✓ Columns集合和DataColumn对象
  - ✓ Rows集合和DataRow对象
  - ✓ Relations集合和DataRelation对象
-



## 15.5 数据绑定

### 15.5.1 数据绑定概述

C#的大部分控件都有数据绑定功能，例如Label、TextBox、dataGridView等控件。当控件进行数据绑定操作后，该控件即会显示所查询的数据记录。窗体控件的数据绑定一般可以分为两种方式：单一绑定和复合绑定。

## 1. 单一绑定

所谓单一绑定是指将单一的数据元素绑定到控件的某个属性。例如，将TextBox控件的Text属性与student数据表中的姓名列进行绑定。

单一绑定是利用控件的DataBindings集合属性来实现的，其一般形式如下：

控件名称.DataBindings.Add("控件的属性名称",数据源,"数据成员");

这3个参数构成了一个Binding对象。也可以先创建Binding对象，再使用Add方法将其添加到DataBindings集合属性中。

## Binding对象的构造函数如下：

`Binding("控件的属性名称",数据源,"数据成员");`

例如，以下语句建立myds数据集的“Student.学号”列到一个控件Text属性的绑定。

```
DataSet myds = new DataSet();
```

...

```
Binding mybinding = new Binding("Text",myds,"Student.学号");  
textBox1.DataBindings.Add(mybinding)
```



将myds中Student表的学号与textBox1绑定起来

这种方式是将每个文本框与一个数据成员进行绑定，  
不便于数据源的整体操作。

---

C#提供了BindingSource类（在工具箱中，对应该控件的图标为），它用于封装窗体的数据源，实现对数据源的整体导航操作。

BindingSource类的常用构造函数如下：

**BindingSource();**

**BindingSource(dataSource, dataMember);**

其中，dataSource指出BindingSource对象的数据源。  
dataMember指出要绑定的数据源中的特定列或列表名称。  
即用指定的数据源和数据成员初始化BindingSource类的新实例。

---

**【15.18】** 设计一个窗体，用于实现对student表中所有记录进行浏览操作。

## Form9窗体

设计界面：

The screenshot shows a Windows form titled 'Form9'. Inside the form, there is a group box labeled '学生记录' (Student Record). Within this group box, there are five text boxes: 'textBox1' for '学号' (Student ID), 'textBox2' for '姓名' (Name), 'textBox3' for '性别' (Gender), 'textBox4' for '民族' (Nationality), and 'textBox5' for '班号' (Class Number). Below the group box, there are four buttons: a first button with a red '<' and a black '<-' symbol, a second button with a red '<-' symbol, a third button with a red '>-' symbol, and a fourth button with a red '>' and a black '>-' symbol. The fourth button is currently selected, indicated by a dashed border and small handles.

代码：

```
namespace proj15_1
{
    public partial class Form9 : Form
    {
        BindingSource mybs = new BindingSource(); //类变量
        public Form9()
        {
            InitializeComponent();
        }
    }
}
```

```
private void Form9_Load(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    DataSet myds = new DataSet();
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\C#程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM student";
    OleDbDataAdapter myda = new OleDbDataAdapter(mysql, myconn);
    myda.Fill(myds, "student");
    mybs = new BindingSource(myds, "student");
    //用数据源myds和表student创建新实例mybs
    Binding mybinding1 = new Binding("Text", mybs, "学号");
    textBox1.DataBindings.Add(mybinding1);
    Binding mybinding2 = new Binding("Text", mybs, "姓名");
    textBox2.DataBindings.Add(mybinding2);
    Binding mybinding3 = new Binding("Text", mybs, "性别");
    textBox3.DataBindings.Add(mybinding3);
    Binding mybinding4 = new Binding("Text", mybs, "民族");
    textBox4.DataBindings.Add(mybinding4);
    Binding mybinding5 = new Binding("Text", mybs, "班号");
    textBox5.DataBindings.Add(mybinding5);
    myconn.Close();
}
```

```
private void button1_Click(object sender, EventArgs e)
{   if (mybs.Position != 0)
        mybs.MoveFirst(); //移到第一个记录
}
private void button2_Click(object sender, EventArgs e)
{   if (mybs.Position !=0)
        mybs.MovePrevious(); //移到上一个记录
}
private void button3_Click(object sender, EventArgs e)
{   if (mybs.Position != mybs.Count - 1)
        mybs.MoveNext(); //移到下一个记录
}
private void button4_Click(object sender, EventArgs e)
{   if (mybs.Position != mybs.Count - 1)
        mybs.MoveLast(); //移到最后一个记录
}
}
}
```

## 运行界面



The screenshot shows a Windows application window titled "Form9". Inside the window, there is a form titled "学生记录" (Student Record). The form contains several text input fields with labels in blue text:

- 学号 (Student ID): 3
- 姓名 (Name): 李兵
- 性别 (Gender): 男
- 民族 (Ethnicity): 汉族
- 班号 (Class Number): 07001

At the bottom of the form, there are four buttons with navigation icons:

- |<-
- <-
- >
- >|



## 2. 复合绑定

所谓复合绑定是指一个控件和一个以上的数据元素进行绑定，通常是指将控件和数据集中的多个数据记录或者多个字段值、数组中的多个数组元素进行绑定。

ComboBox、ListBox和CheckedListBox等控件都支持复合数据绑定。在实现复合绑定时，关键的属性是DataSource和DataMember（或DisplayMember）等。

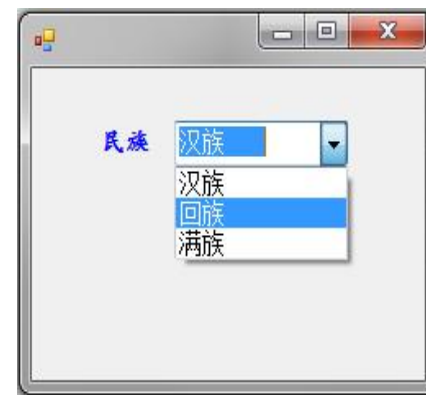
复合绑定的语法格式如下：

控件对象名称.DataSource = 数据源

控件对象名称.DisplayMember = 数据成员

例如，一个窗体myForm中有一个组合框comboBox1，在该窗体中设计以下Load事件过程：

```
private void myForm_Load(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    DataSet myds = new DataSet();
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#
程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT distinct 民族 FROM student";
    OleDbDataAdapter myda = new OleDbDataAdapter(mysql, myconn);
    myda.Fill(myds, "student");
    comboBox1.DataSource = myds;
    comboBox1.DisplayMember = "student.民族";
    myconn.Close();
}
```



## 15.5.2 BindingNavigator控件

在大多数情况下，BindingNavigator控件（绑定到导航工具栏）与BindingSource控件（绑定到数据源）成对出现，用于浏览窗体上的数据记录，并与它们交互。

在这些情况下，BindingSource属性被设置为作为数据源的关联BindingSource控件（或对象）。

## BindingNavigator成员和BindingSource成员的对应关系

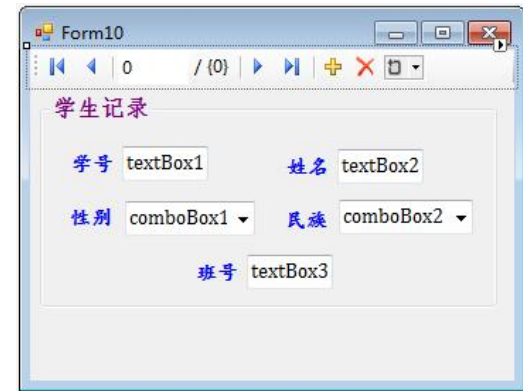
UI 控件	BindingNavigator成员	BindingSource成员
移到最前	MoveFirstItem	MoveFirst
前移一步	MovePreviousItem	MovePrevious
当前位置	PositionItem	Current
统计	CountItem	Count
移到下一条记录	MoveNextItem	MoveNext
移到最后	MoveLastItem	MoveLast
新添	AddNewItem	AddNew
删除	DeleteItem	RemoveCurrent

**【15.19】**设计一个窗体，通过BindingNavigator控件实现对student表中所有记录进行浏览操作。

## Form10窗体

设计界面：

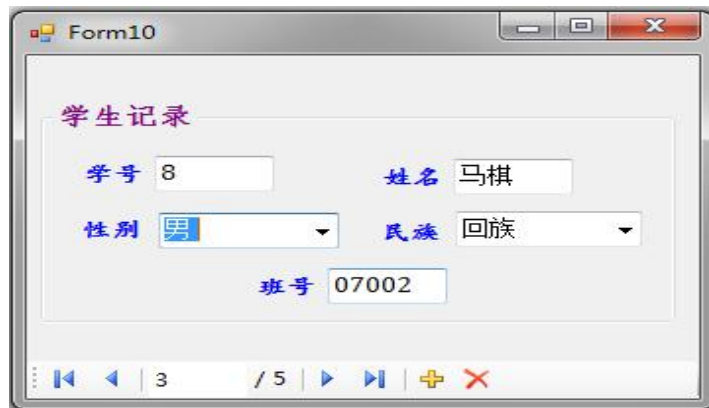
事件过程：



```
private void Form10_Load(object sender, EventArgs e)
{
    OleDbConnection myconn = new
    OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
    Source=D:\C#程序\ch15\school.accdb");
    OleDbDataAdapter myda = new OleDbDataAdapter("SELECT *
    FROM student", myconn);
    DataSet myds = new DataSet();
    BindingSource mybs = new BindingSource();
    myconn.Open();
    myda.Fill(myds, "student");
}
```

```
mybs = new BindingSource(myds, "student");  
    //用数据源myds和表student创建新实例mybs  
Binding mybinding1 = new Binding("Text", mybs, "学号");  
textBox1.DataBindings.Add(mybinding1);  
    //将student.学号与textBox1文本框绑定起来  
Binding mybinding2 = new Binding("Text", mybs, "姓名");  
textBox2.DataBindings.Add(mybinding2);  
Binding mybinding3 = new Binding("Text", mybs, "性别");  
comboBox1.DataBindings.Add(mybinding3);  
Binding mybinding4 = new Binding("Text", mybs, "民族");  
comboBox2.DataBindings.Add(mybinding4);  
Binding mybinding5 = new Binding("Text", mybs, "班号");  
textBox3.DataBindings.Add(mybinding5);  
bindingNavigator1.Dock = DockStyle.Bottom;  
bindingNavigator1.BindingSource = mybs;  
myconn.Close();  
comboBox1.Items.Add("男"); comboBox1.Items.Add("女");  
comboBox2.Items.Add("汉族");comboBox2.Items.Add("回族");  
comboBox2.Items.Add("满族");comboBox2.Items.Add("土家族");  
}
```

## 运行界面



The screenshot shows a Windows application window titled "Form10". Inside the window, there is a form titled "学生记录" (Student Record). The form contains the following fields:

- 学号 (Student ID): 8
- 姓名 (Name): 马棋
- 性别 (Gender): 男 (Male)
- 民族 (Nationality): 回族 (Hui Nationality)
- 班号 (Class Number): 07002

At the bottom of the window, there is a navigation bar with the following elements:

- Navigation icons: back, forward, search, and other controls.
- Page indicator: 3 / 5
- Buttons: a yellow plus sign and a red minus sign.

## 15.6 DataView对象

### 15.6.1 DataView对象概述

**DataView对象类似于数据库中的View功能，提供DataTable列（Column）排序、过滤记录（Row）及记录的搜索，它的一个常见用法是为控件提供数据绑定。**

**DataView对象的构造函数如下：**

**DataView()**

**DataView(table)**

**DataView(table, RowFilter, Sort, RowState)**



---

为给定的DataTable创建一个新的DataView，可以声明该DataView，把DataTable的一个引用mydt传给DataView构造函数，例如：

```
DataView mydv = new DataView(mydt);
```

在第一次创建DataView时，DataView默认为mydt中的所有行。用过滤条件属性可以得到DataView中数据行的一个子集合，也可以为这些数据排序。

---

**DataTable对象提供DefaultView属性返回默认的数据视图对象。例如：**

```
DataGridView mydv = new DataGridView();  
mydv = myds.Tables["student"].DefaultView;
```

**上述代码从myds数据集中取得student表的默认内容，再利用相关控件（如DataGridView）显示内容，指定数据来源为mydv。**

<b>DataView对象的属性</b>	<b>说 明</b>
<b>AllowDelete</b>	设置或获取一个值，该值指示是否允许删除。
<b>AllowEdit</b>	获取或设置一个值，该值指示是否允许编辑。
<b>Allownew</b>	获取或设置一个值，该值指示是否可以使用Addnew方法添加新行。
<b>ApplyDefaultSort</b>	获取或设置一个值，该值指示是否使用默认排序。
<b>Count</b>	在应用RowFilter和RowStateFilter之后，获取DataView中记录的数量。
<b>Item</b>	从指定的表获取一行数据。
<b>RowFilter</b>	获取或设置用于筛选在DataView中查看哪些行的表达式。
<b>RowStateFilter</b>	获取或设置用于DataView中的行状态筛选器。
<b>Sort</b>	获取或设置 DataView 的一个或多个排序列以及排序顺序。
<b>Table</b>	获取或设置源DataTable。

<b>DataGridView对象的方法</b>	<b>说 明</b>
<b>Addnew</b>	将新行添加到DataGridView中。
<b>Delete</b>	删除指定索引位置的行。
<b>Find</b>	按指定的排序关键字值在 DataGridView 中查找行。
<b>FindRows</b>	返回DataRowView对象的数组，这些对象的列与指定的排序关键字值匹配。
<b>ToTable</b>	根据现有DataGridView中的行，创建并返回一个新的 DataTable。

## 15.6.2 DataView对象的列排序设置

DataView取得一个表之后，利用Sort属性指定依据某些列（Column）排序，Sort属性允许复合键的排序，列之间使用逗号隔开即可。

排序的方式又分为升序（Asc）和降序（Desc），在列之后接Asc或Desc关键字即可。

**【15.20】** 设计一个窗体，使用DataView对象在列表框中按学号升序、分数降序排序显示所有成绩记录。

Form10窗体

设计界面：只有一个列表框listBox1。

事件过程：

```
private void Form11_Load(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    DataSet myds = new DataSet();

    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\C#程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
```

```
mysql = "SELECT * FROM score";  
OleDbDataAdapter myda = new OleDbDataAdapter(mysql, myconn);  
myda.Fill(myds, "score");  
myconn.Close();  
DataView mydv = new DataView(myds.Tables["score"]);  
mydv.Sort = "学号 ASC,分数 DESC";  
listBox1.Items.Add("学号\t课程名\t\t分数");  
for (int i = 0; i < mydv.Count; i++)  
{  
    listBox1.Items.Add(String.Format("{0}\t{1,-6}\t{2}",  
        mydv[i]["学号"], mydv[i]["课程名"],mydv[i]["分数"]));  
}  
}
```



学号	课程名	分数
1	数据结构	83
1	C语言	80
2	C语言	70
2	数据结构	52
3	C语言	76
3	数据结构	70
6	数据结构	92
6	C语言	90
8	C语言	88
8	数据结构	79

**思考题：**可以修改SELECT语句实现同样的功能吗？

### 15.6.3 DataView对象的过滤条件设置

获取数据的子集合可以用DataView类的RowFilter属性或RowStateFilter属性来实现。

RowFilter属性用于提供过滤表达式。RowFilter表达式可以非常复杂，也可以包含涉及多个列中的数据和常数的算术计算与比较。

RowFilter属性的值是一个条件表达式。同查询语句的模糊查询一样，该属性也有Like子句及%字符。

RowStateFilter属性指定从DataTable中提取特定数据子集合的值。



例如：

...

```
DataGridView mydv = new DataGridView(myds.Tables["score"]);
```

```
mydv.Sort = "学号 ASC,分数 DESC";
```

```
mydv.RowFilter = "分数>80";
```

...

## 15.7 DataGridView控件

DataGridView控件用于在窗体中显示表格数据。

### 15.7.1 创建DataGridView对象

从工具箱中将DataGridView控件拖放到窗体上，此时在DataGridView控件右侧出现“DataGridView任务”菜单，其操作步骤如下。

**DataGridView 任务**

选择数据源: (无) ▼

[编辑列...](#)

[添加列...](#)

☒ 启用添加

☒ 启用编辑

☒ 启用删除

☐ 启用列重新排序

[在父容器中停靠](#)

数据源配置向导

选择数据源类型


应用程序将从哪里获取数据(W)?

☒ 数据库 ☐ 服务 ☐ 对象 ☐ SharePoint

允许您连接到数据库并为您的应用程序选择数据库对象。

< 上一步(P) **下一步(N) >** 完成(F) 取消

数据源配置向导



选择您的数据连接

应用程序连接数据库应使用哪个数据连接(W)?

school.acddb

新建连接(C)...

此连接字符串中似乎包含连接到数据库所需的敏感数据(例如密码)，而在连接字符串中存储敏感数据会带来安全风险。是否在连接字符串中包含敏感数据?

☐ 否，从连接字符串中排除敏感数据。我将在应用程序代码中设置此信息(E)。

☒ 是，在连接字符串中包含敏感数据(I)。

☒ 将保存到应用程序中的连接字符串(展开可查看详细信息)(S)


< 上一步(P)

下一步(N) >

完成(F)

取消

数据源配置向导



将连接字符串保存到应用程序配置文件中

在应用程序配置文件中存储连接字符串可简化维护和部署工作。要在应用程序配置文件中保存连接字符串，请在框中输入一个名称再单击“下一步”。

是否将连接字符串保存到应用程序配置文件中?

☒ 是，将连接保存为(Y):

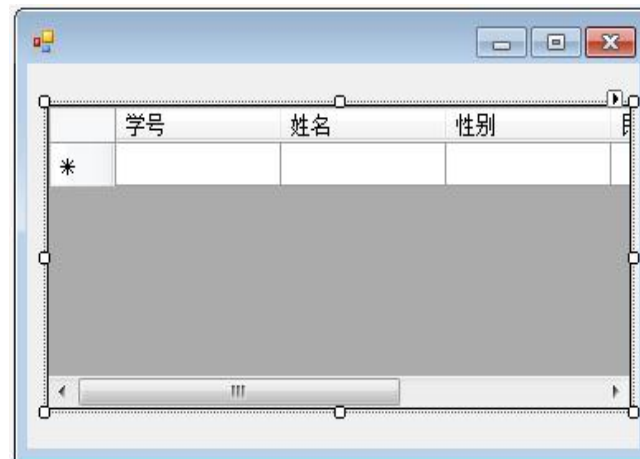
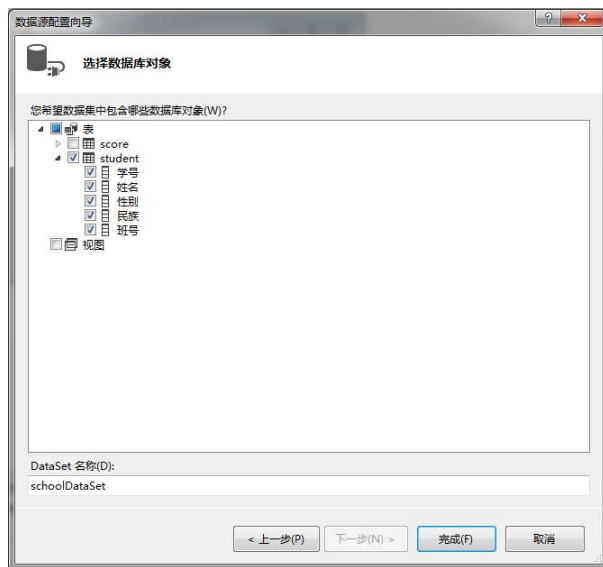
schoolConnectionString

< 上一步(P)

下一步(N) >

完成(F)

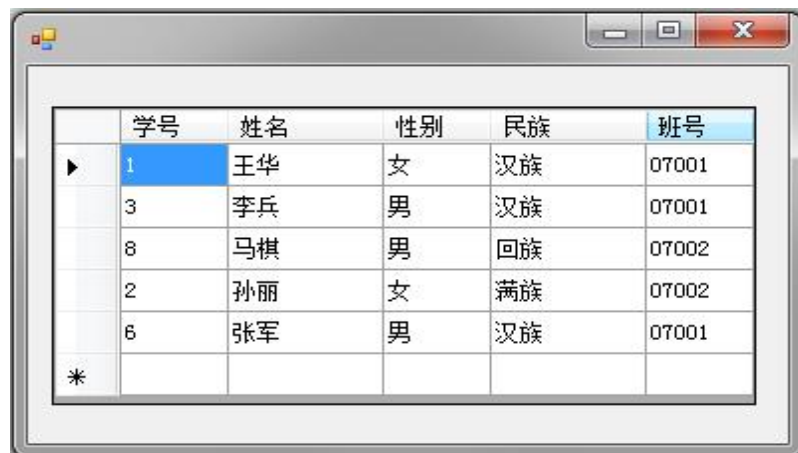
取消



选中DataGrid1View1控件，单击鼠标右键，在弹出的快捷菜单中选择“编辑列”命令，打开“编辑列”对话框，将每个列的AutoSizeMode属性设置为AllCells，还可以改变每个列的样式等，单击“确定”按钮返回。



## 运行结果



A screenshot of a Windows application window with a standard title bar (minimize, maximize, close buttons). The window contains a table with 6 columns: an index column, '学号' (Student ID), '姓名' (Name), '性别' (Gender), '民族' (Ethnicity), and '班号' (Class ID). The table has 6 data rows. The first row is selected, highlighting the index '1' and the student '王华'. A small black triangle icon is in the first column of the first row. A small asterisk icon is in the first column of the last row.

	学号	姓名	性别	民族	班号
▶	1	王华	女	汉族	07001
	3	李兵	男	汉族	07001
	8	马棋	男	回族	07002
	2	孙丽	女	满族	07002
	6	张军	男	汉族	07001
*					

## 15.7.2 DataGridView的属性、方法和事件

**DataGridView对象的常用属性如表15.41所示。其中Columns属性是一个列集合，由Column列对象组成，每个Column列对象的常用属性如表15.42所示。**

**DataGridView对象的常用方法如表15.43所示。其常用事件如表15.44所示。**



## 1. 基本数据绑定

例如，在一个窗体myForm1上拖放一个dataGridView1对象后，不设计其任何属性，可以使用以下程序代码来实现基本数据的绑定：

```
private void myForm1_Load(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    DataSet myds = new DataSet();
    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#
程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM student";
    OleDbDataAdapter myda = new OleDbDataAdapter(mysql, myconn);
    myda.Fill(myds, "student");
    dataGridView1.DataSource = myds.Tables["student"];
}
```

## 2. 设计显示样式

可以通过GridColor属性设置其网格线的颜色，例如，设置GridColor颜色为蓝色：

```
DataGridView1.GridColor = Color.Blue;
```

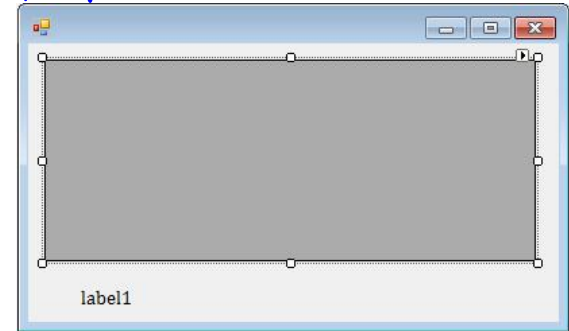
通过BorderStyle属性设置其网格的边框样式，其枚举值为：FixedSingle、Fixed3D和none。通过CellBorderStyle属性设置其网格单元的边框样式等。

**【15.22】** 设计一个窗体，采用DataGridView控件来实现对student表中所有记录进行浏览操作。

Form10窗体

设计界面。

事件过程：



```
private void Form13_Load(object sender, EventArgs e)
{
    string mystr,mysql;
    OleDbConnection myconn = new OleDbConnection();
    DataSet myds = new DataSet();

    mystr=@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=D:\C#
程序\ch15\school.accdb";
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM student";
    OleDbDataAdapter myda = new OleDbDataAdapter(mysql, myconn);
    myda.Fill(myds, "student");
```

```
dataGridView1.DataSource = myds.Tables["student"];
dataGridView1.GridColor = Color.RoyalBlue;
dataGridView1.ScrollBars = ScrollBars.Vertical;
dataGridView1.CellBorderStyle =
    DataGridViewCellBorderStyle.Single;
dataGridView1.Columns[0].AutoSizeMode =
    DataGridViewAutoSizeColumnMode.AllCells;
dataGridView1.Columns[1].AutoSizeMode =
    DataGridViewAutoSizeColumnMode.AllCells;
dataGridView1.Columns[2].AutoSizeMode =
    DataGridViewAutoSizeColumnMode.AllCells;
dataGridView1.Columns[3].AutoSizeMode =
    DataGridViewAutoSizeColumnMode.AllCells;
dataGridView1.Columns[4].AutoSizeMode =
    DataGridViewAutoSizeColumnMode.AllCells;
myconn.Close();
label1.Text = "";
}
```

```
private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
{   label1.Text = "";
    try
    {   if (e.RowIndex < dataGridView1.RowCount - 1)
        label1.Text = "选择的学生学号为:" +
            dataGridView1.Rows[e.RowIndex].Cells[0].Value;
    }
    catch(Exception ex)
    {
        MessageBox.Show("需选中一个学生记录", "信息提示");
    }
}
```



### 15.7.3 DataGridView与DataView对象结合

**DataGridView**对象用于在窗体上显示记录数据，而**DataView**对象可以方便地对源数据记录进行排序等操作，两者结合可以设计复杂的应用程序。

【15.23】设计一个窗体，用于实现对student表中记录的通用查找和排序操作。

设计界面：

The design interface shows a window with a large empty rectangular area at the top. Below this area, there are two main sections: '查询条件设置' (Query Condition Setting) and '排序' (Sort). The '查询条件设置' section contains five input fields: '学号' (Student ID) with a text box (textBox1), '姓名' (Name) with a text box (textBox2), '性别' (Gender) with a dropdown menu (comboBox1), '民族' (Nationality) with a dropdown menu (comboBox2), and '班号' (Class ID) with a dropdown menu (comboBox3). There are '确定' (OK) and '重置' (Reset) buttons below these fields. The '排序' section contains a dropdown menu (comboBox4) and two radio buttons for '升序' (Ascending) and '降序' (Descending). There are also '确定' (OK) and '重置' (Reset) buttons below these options.

运行界面：

The run interface shows the window with a table of search results. The table has columns: '学号' (Student ID), '姓名' (Name), '性别' (Gender), '民族' (Nationality), and '班号' (Class ID). The first two rows are highlighted in blue. The first row has '学号' 8, '姓名' 马棋, '性别' 男, '民族' 回族, and '班号' 07002. The second row has '学号' 2, '姓名' 孙丽, '性别' 女, '民族' 满族, and '班号' 07002. Below the table, the '查询条件设置' section shows the search criteria: '学号' is empty, '姓名' is empty, '性别' is empty, '民族' is empty, and '班号' is 07002. The '排序' section shows '升序' (Ascending) selected. There are '确定' (OK) and '重置' (Reset) buttons.

学号	姓名	性别	民族	班号
8	马棋	男	回族	07002
2	孙丽	女	满族	07002
*				

The run interface shows the window with a table of search results. The table has columns: '学号' (Student ID), '姓名' (Name), '性别' (Gender), '民族' (Nationality), and '班号' (Class ID). The first two rows are highlighted in blue. The first row has '学号' 2, '姓名' 孙丽, '性别' 女, '民族' 满族, and '班号' 07002. The second row has '学号' 8, '姓名' 马棋, '性别' 男, '民族' 回族, and '班号' 07002. Below the table, the '查询条件设置' section shows the search criteria: '学号' is empty, '姓名' is empty, '性别' is empty, '民族' is empty, and '班号' is 07002. The '排序' section shows '学号' (Student ID) selected in the dropdown menu. There are '确定' (OK) and '重置' (Reset) buttons.

学号	姓名	性别	民族	班号
2	孙丽	女	满族	07002
8	马棋	男	回族	07002
*				

## 15.7.4 通过DataGridView对象更新数据源

当运行时DataGridView对象中的数据可以修改，只是内存中的数据发生了更改，对应数据源数据并没有改动。

为了更新数据源，需对相应的OleDbDataAdapter对象执行Update方法。

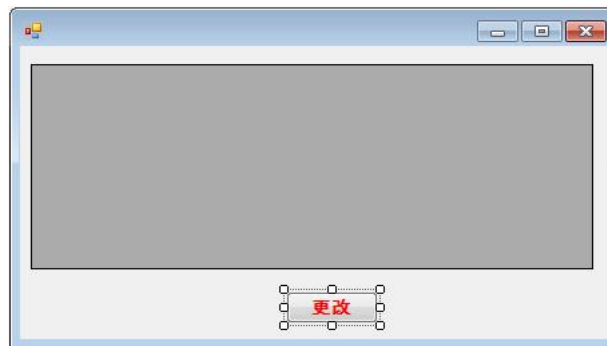


**【15.24】** 设计一个窗体，用于实现对student表中记录的修改操作。

Form15窗体

设计界面。

主要事件过程：



```
private void button1_Click(object sender, EventArgs e)
{
    OleDbCommandBuilder mycmdbuilder = new OleDbCommandBuilder(myda);
    //获取对应的修改命令
    if (myds.HasChanges()) //如果有数据改动
    {
        try
        {
            myda.Update(myds, "student"); //更新数据源
        }
        catch(Exception ex)
        {
            MessageBox.Show("数据修改不正确，如学号重复等","信息提示");
        }
    }
}
```

运行界面：



修改后单击“更改”，数据库发生更新

——本章完——