16.1 ADO.NET 概述

视频讲解：光盘\TM\第 16 章\视频 16.1~16.2\ADO.NET 概述及使用 Connection 对象连接数据库.exe

ADO.NET 是一组向.NET 程序员公开数据访问服务的类，它为创建分布式数据共享应用程序提供了一组丰富的组件。ADO.NET 提供了一系列的方法，用于支持对 Microsoft SQL Server 和 XML 等数据源进行访问，还提供了通过 OLE DB 和 XML 公开的数据源提供一致访问的方法。数据客户端应用程序可以使用 ADO.NET 来连接到这些数据源，并查询、添加、删除和更新所包含的数据。

ADO.NET 支持两种访问数据的模型：无连接模型和连接模型。无连接模型将数据下载到客户机上，并在客户机上将数据封装到内存中，然后可以像访问本地关系数据库一样访问内存中的数据（如 DataSet）；连接模型则依赖于逐记录的访问，这种访问要求打开并保持与数据源的连接。

16.2 使用 Connection 对象连接数据库

16.2.1 Connection 对象概述

Connection 对象是一个连接对象，其主要功能是建立与物理数据库的连接。Connection 对象主要包括以下 4 种访问数据库的对象类（也可称为数据提供程序）：

- ☑ SQL Server 数据提供程序，位于 System.Data.SqlClient 命名空间。
- ☑ ODBC 数据提供程序，位于 System.Data.Odbc 命名空间。
- ☑ OLE DB 数据提供程序，位于 System.Data.OleDb 命名空间。
- ☑ Oracle 数据提供程序，位于 System.Data.OracleClient 命名空间。

说明 根据使用数据库的不同，引入不同的命名空间，然后通过命名空间中的 Connection 对象连接类连接数据库。例如，连接 SQL Server 数据库，首先要通过 using System.Data.SqlClient 命令引用 SQL Server 数据提供程序，然后才能调用空间下的 SqlConnection 类连接数据库。

16.2.2 连接数据库

以 SQL Server 数据库为例，如果要连接 SQL Server 数据库，必须使用 System.Data.SqlClient 命名空间下的 SqlConnection 类，所以首先要通过 using System.Data.SqlClient 命令引用命名空间。连接数据库之后，通过调用 SqlConnection 对象的 Open 方法打开数据库。另外，可以通过 SqlConnection 对象的 State 属性判断数据库的连接状态。其语法格式如下：

```
public override ConnectionState State { get; }
```

属性值：ConnectionState 枚举值之一。

ConnectionState 枚举的值及说明如表 16.1 所示。

表 16.1 ConnectionState 枚举的值及说明

枚 举 值	说 明
Broken	与数据源的连接中断。只有在连接打开之后才可能发生这种情况。可以关闭处于这种状态的连接，然后重新打开

续表

枚 举 值	说　明
Closed	连接处于关闭状态
Connecting	连接对象正在与数据源连接
Executing	连接对象正在执行命令
Fetching	连接对象正在检索数据
Open	连接处于打开状态

例 16.1　创建一个 Windows 应用程序，在 Form1 窗体中添加一个 TextBox 控件、一个 Button 控件和一个 Label 控件，分别用于输入要连接的数据库名称、执行连接数据库的操作以及显示数据库的连接状态；然后引入 System.Data.SqlClient 命名空间，使用 SqlConnection 类连接数据库。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.1**）

```
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")                                    //判断是否输入数据库名称
    {
        MessageBox.Show("请输入要连接的数据库名称");              //弹出提示信息
    }
    else
    {
        try                                                     //调用 try…catch 语句
        {
            //声明一个字符串用于存储连接数据库字符串
            string ConStr = "server=.;database=" + textBox1.Text.Trim() + ";uid=sa;pwd=";
            //创建一个 SqlConnection 对象
            SqlConnection conn = new SqlConnection(ConStr);
            conn.Open();                                        //打开连接
            if (conn.State == ConnectionState.Open)             //判断当前连接的状态
            {
                //显示状态信息
                label2.Text = "数据库【" + textBox1.Text.Trim() + "】已经连接并打开";
            }
        }
        catch
        {
            MessageBox.Show("连接数据库失败");                  //出现异常弹出提示
        }
    }
}
```

程序运行结果如图 16.1 所示。

图 16.1　连接数据库

16.2.3　关闭连接

完成对数据库的操作后，要关闭与数据库的连接，以释放占用的资源。可以通过调用 SqlConnection 对象的 Close 或 Dispose 方法关闭与数据库的连接。这两种方法的主要区别是：Close 方法用于关闭一个连接，而 Dispose 方法不仅关闭一个连接，而且还清理连接所占用的资源。当使用 Close 方法关闭连接后，可以再调用 Open 方法打开连接，不会产生任何错误；而如果使用 Dispose 方法关闭连接，则不可以再次直接用 Open 方法打开连接，必须重新初始化连接之后再打开。

例 16.2　创建一个 Windows 应用程序，首先向 Form1 窗体中添加一个 TextBox 控件和一个 RichTextBox 控件，分别用于输入连接的数据库名称和显示连接信息及错误提示；然后再添加 3 个 Button 控件，分别用于连接数据库、调用 Close 方法关闭连接再调用 Open 方法打开连接以及调用 Dispose 方法关闭并释放连接再调用 Open 方法打开连接。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.2**）

```
SqlConnection conn;                                          //声明一个 SqlConnection 对象
private void button1_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")                                 //判断是否输入数据库名称
    {
        MessageBox.Show("请输入数据库名称");                  //如果没有输入则弹出提示
    }
    else
    {
        try                                                  //调用 try…catch 语句
        {
            //建立连接数据库字符串
            string str = "server=.;database=" + textBox1.Text.Trim() + ";uid=sa;pwd=";
            conn = new SqlConnection(str);                   //创建一个 SqlConnection 对象
            conn.Open();                                     //打开连接
            if (conn.State == ConnectionState.Open)          //判断当前连接状态
            {
                MessageBox.Show("连接成功");                 //弹出提示
            }
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);                     //出现异常弹出错误信息
            textBox1.Text = "";                              //清空文本框
        }
    }
}
private void button2_Click(object sender, EventArgs e)
{
    try                                                      //调用 try…catch 语句
    {
        string str="";                                       //声明一个字符串变量
        conn.Close();                                        //使用 Close 方法关闭连接
        if (conn.State == ConnectionState.Closed)            //判断当前连接是否关闭
        {
            str="数据库已经成功关闭\n";                       //如果关闭则弹出提示
        }
        conn.Open();                                         //重新打开连接
        if (conn.State == ConnectionState.Open)              //判断连接是否打开
        {
            str += "数据库已经成功打开\n";                    //弹出提示
        }
        richTextBox1.Text = str;                             //向 richTextBox1 中添加提示信息
    }
    catch (Exception ex)
    {
        richTextBox1.Text = ex.Message;                      //出现异常，将异常添加到 richTextBox1 中
    }
}
private void button3_Click(object sender, EventArgs e)
{
    try                                                      //调用 try…catch 语句
```


猫爱中国 ngmaoi.com

```
    {
        conn.Dispose();                                    //使用 Dispose 方法关闭连接
        conn.Open();                                       //重新使用 Open 方法打开会出现异常
    }
    catch (Exception ex)
    {
        richTextBox1.Text = ex.Message;                    //将异常显示在 richTextBox1 控件中
    }
}
```

程序运行结果如图 16.2 和图 16.3 所示。

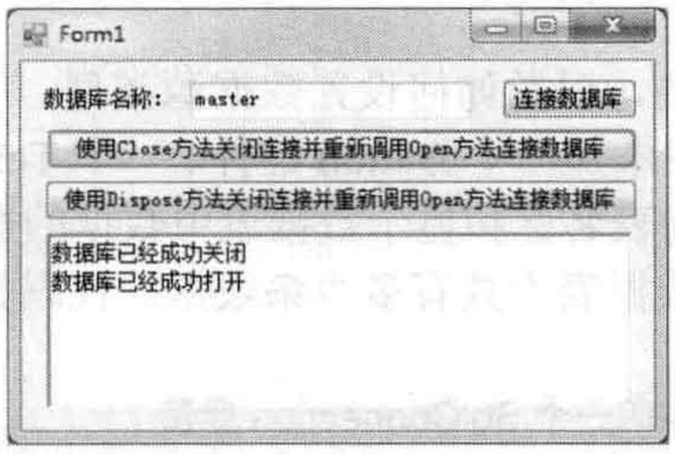

图 16.2　调用 Close 方法关闭连接

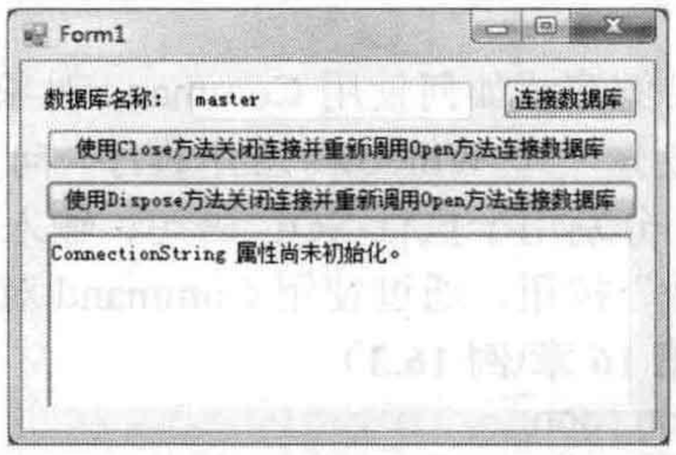

图 16.3　调用 Dispose 方法关闭并释放连接

16.3　使用 Command 对象执行 SQL 语句

视频讲解：光盘\TM\第 16 章\视频 16.3\使用 Command 对象执行 SQL 语句.exe

16.3.1　Command 对象概述

Command 对象是一个数据命令对象，其主要功能是向数据库发送查询、更新、删除、修改操作的 SQL 语句。Command 对象主要有以下几种形式。

- ☑ SqlCommand：用于向 SQL Server 数据库发送 SQL 语句，位于 System.Data.SqlClient 命名空间。
- ☑ OleDbCommand：用于向 OLE DB 公开的数据库发送 SQL 语句，位于 System.Data.OleDb 命名空间。例如，Access 数据库和 MySQL 数据库都是 OLE DB 公开的数据库。
- ☑ OdbcCommand：用于向 ODBC 公开的数据库发送 SQL 语句，位于 System.Data.Odbc 命名空间。如果数据库没有提供相应的连接程序，则应首先配置好 ODBC 连接，然后再使用 OdbcCommand。
- ☑ OracleCommand：用于向 Oracle 数据库发送 SQL 语句，位于 System.Data.OracleClient 命名空间。

注意　在使用 OracleCommand 向 Oracle 数据库发送 SQL 语句时，需要引入 System.Data.OracleClient 命名空间。但是默认情况下没有此命名空间，此时需要将程序集 System.Data.OracleClient 引入到项目中。在项目名称上单击鼠标右键，在弹出的快捷菜单中选择“添加引用”命令，在打开的“添加引用”对话框中选择 System.Data.OracleClient 程序集，单击“确定”按钮，即可将程序集添加到项目中。

16.3.2　设置数据源类型

Command 对象有 3 个重要的属性，分别是 Connection、CommandText 和 CommandType 属性。Connection

熊猫爱中国
xiongmaoi.com

属性用于设置 SqlCommand 使用的 SqlConnection 连接对象；CommandText 属性用于设置要对数据源执行的 SQL 语句或存储过程；CommandType 属性用于设置指定 CommandText 的类型。CommandType 属性的值是 CommandType 枚举值，CommandType 枚举有 3 个枚举成员，分别介绍如下。

- ☑ StoredProcedure：存储过程的名称。
- ☑ TableDirect：表的名称。
- ☑ Text：SQL 文本命令。

提示 如果要设置数据源的类型，可以通过设置 CommandType 属性来实现。

下面通过实例演示如何使用 Command 对象的这 3 个属性，以及如何设置数据源类型。

例 16.3 创建一个 Windows 应用程序，向 Form1 窗体中添加一个 Button 控件、一个 TextBox 控件和一个 Label 控件，分别用于执行 SQL 语句、输入要查询的数据表名称和显示数据表中数据的数量；单击“查询数据表中数据”按钮，通过使用 Command 对象获取指定数据表中共有多少条数据。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.3**）

```
SqlConnection conn;                                          //声明一个 SqlConnection 变量
private void Form1_Load(object sender, EventArgs e)
{
    //实例化 SqlConnection 变量 conn
    conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
    conn.Open();                                             //打开连接
}
private void button1_Click(object sender, EventArgs e)
{
try                                                          //调用 try…catch 语句
{
    //判断是否打开连接或者文本框不为空
    if (conn.State == ConnectionState.Open || textBox1.Text != "")
    {
        SqlCommand cmd = new SqlCommand();                   //创建一个 SqlCommand 对象
        cmd.Connection = conn;                               //设置 Connection 属性
        //设置 CommandText 属性，设置 SQL 语句
        cmd.CommandText = "select count(*) from " + textBox1.Text.Trim();
        //设置 CommandType 属性为 Text，使其只执行 SQL 语句文本形式
        cmd.CommandType = CommandType.Text;
        //使用 ExecuteScalar 方法获取指定数据表中的数据数量
        int i = Convert.ToInt32(cmd.ExecuteScalar());
        label2.Text = "数据表中共有：" + i.ToString() + "条数据";
    }
}
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

程序运行结果如图 16.4 所示。

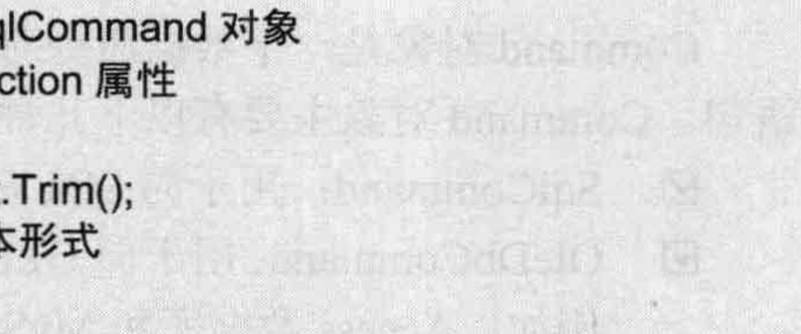

图 16.4 SqlCommand 对象执行查询语句

16.3.3 执行 SQL 语句

Command 对象提供了多种执行 SQL 语句的方法，下面以 SqlCommand 对象为例，介绍几种常用的执行 SQL 语句的方法。

熊猫爱中国 xiongmaoi.com

1．ExecuteNonQuery 方法

ExecuteNonQuery 方法用于执行 SQL 语句，并返回受影响的行数。在使用 SqlCommand 对象向数据库发送增、删、改命令时，通常使用 ExecuteNonQuery 方法执行发送的 SQL 语句。其语法格式如下：

```
public override int ExecuteNonQuery()
```

返回值：受影响的行数。

例 16.4　创建一个 Windows 应用程序。在“三八”妇女节这天，公司决定为每位女员工颁发奖金 50 元。这样，就需要向数据库发送更新命令，将数据库中所有女员工的奖金数额加上 50。这里使用 ExecuteNonQuery 方法执行发送的 SQL 语句，并获取受影响的行数。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.4**）

```
SqlConnection conn;                                          //声明一个 SqlConnection 变量
private void button1_Click(object sender, EventArgs e)
{
    //实例化 SqlConnection 变量 conn
    conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
    conn.Open();                                             //打开连接
    SqlCommand cmd = new SqlCommand();                       //创建一个 SqlCommand 对象
    //设置 Connection 属性，指定其使用 conn 连接数据库
    cmd.Connection = conn;
    //设置 CommandText 属性，设置其执行的 SQL 语句
    cmd.CommandText = "update tb_command set 奖金=50 where 性别='女'";
    //设置 CommandType 属性为 Text，使其只执行 SQL 语句文本形式
    cmd.CommandType = CommandType.Text;
    //使用 ExecuteNonQuery 方法执行 SQL 语句
    int i = Convert.ToInt32(cmd.ExecuteNonQuery());
    label2.Text = "共有" + i.ToString() + "名女员工获得奖金";
}
```

程序运行结果如图 16.5 所示。

2．ExecuteReader 方法

ExecuteReader 方法用于执行 SQL 语句，并生成一个包含数据的 SqlDataReader 对象的实例。其语法格式如下：

```
public SqlDataReader ExecuteReader()
```

返回值：一个 SqlDataReader 对象。

例 16.5　创建一个 Windows 应用程序，使用 select * from tb_command 语句对数据库进行查询，并调用 SqlCommand 对象的 ExecuteReader 方法返回一个包含 tb_command 表中所有数据的 SqlDataReader 对象，循环遍历该 SqlDataReader 对象，得到所有的员工姓名。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.5**）

```
SqlConnection conn;                                          //声明一个 SqlConnection 对象
private void button1_Click(object sender, EventArgs e)
{
//实例化 SqlConnection 变量 conn
conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
conn.Open();                                                 //打开连接
SqlCommand cmd = new SqlCommand();                           //创建一个 SqlCommand 对象
//设置 Connection 属性，指定其使用 conn 连接数据库
cmd.Connection = conn;
//设置 CommandText 属性，设置其执行的 SQL 语句
cmd.CommandText = "select * from tb_command";
//设置 CommandType 属性为 Text，使其只执行 SQL 语句文本形式
cmd.CommandType = CommandType.Text;
//使用 ExecuteReader 方法实例化一个 SqlDataReader 对象
SqlDataReader sdr = cmd.ExecuteReader();
```


熊猫爱中国
xiongmaoi.com

```
    while (sdr.Read())                                    //调用 while 语句，读取 SqlDataReader
    {
        listView1.Items.Add(sdr[1].ToString());           //将内容添加到 listView1 控件中
    }
    conn.Dispose();                                       //释放连接
    button1.Enabled = false;                              //禁用按钮
}
```

程序运行结果如图 16.6 所示。

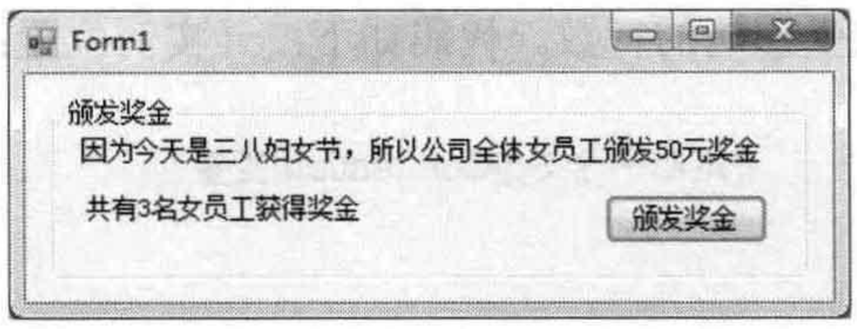

图 16.5　对数据表执行更新操作

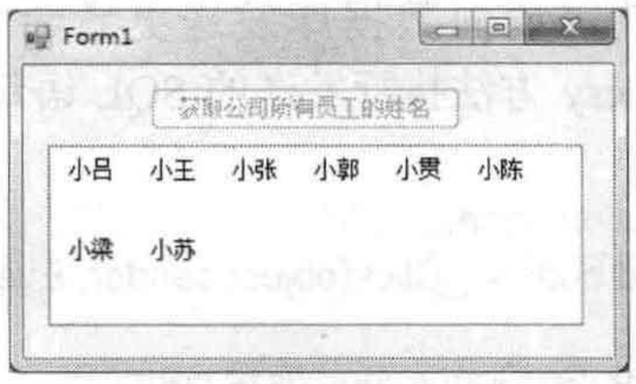

图 16.6　获取员工姓名

3．ExecuteScalar 方法

ExecuteScalar 方法用于执行 SQL 语句，并返回结果集中的第 1 行第 1 列。其语法格式如下：

```
public override Object ExecuteScalar()
```

返回值：结果集中的第 1 行第 1 列或空引用（如果结果集为空）。

说明　ExecuteScalar 方法的使用在例 16.3 中已经作过详细介绍，此处不再赘述。需要注意的是，ExecuteScalar 方法通常与聚合函数一起使用，常见的聚合函数如表 16.2 所示。

表 16.2　常见的聚合函数及说明

聚 合 函 数	说　　明
AVG(expr)	列平均值，该列只能包含数字数据
COUNT(expr)、COUNT(*)	列值的计数（如果将列名指定为 expr）、表或分组中所有行的计数（如果指定*），忽略空值，但 COUNT(*)在计数中包含空值
MAX(expr)	列中最大值（文本数据类型中按字母顺序排在最后的值），忽略空值
MIN(expr)	列中最小值（文本数据类型中按字母顺序排在最前的值），忽略空值
SUM(expr)	列值的合计，该列只能包含数字数据

16.4　使用 DataReader 对象读取数据

视频讲解：光盘\TM\第 16 章\视频 16.4\使用 DataReader 对象读取数据.exe

16.4.1　DataReader 对象概述

DataReader 对象是数据读取器对象，提供只读向前的游标，如果应用程序需要每次从数据库中取出最新的数据，或者只是需要快速读取数据，并不需要修改数据，那么就可以使用 DataReader 对象进行读取。对于不同的数据库连接，有不同的 DataReader 类型。

- ☑ 在 System.Data.SqlClient 命名空间下时，可以调用 SqlDataReader 类。
- ☑ 在 System.Data.OleDb 命名空间下时，可以调用 OleDbDataReader 类。

熊猫爱中国
xiongmaoi.com

☑ 在 System.Data.Odbc 命名空间下时，可以调用 OdbcDataReader 类。

☑ 在 System.Data.Oracle 命名空间下时，可以调用 OracleDataReader 类。

使用 DataReader 对象读取数据时，可以调用 SqlCommand 对象的 ExecuteReader 方法，根据 SQL 语句的结果创建一个 SqlDataReader 对象。

例 16.6 使用 SqlCommand 对象的 ExecuteReader 方法，创建一个读取 tb_command 表中所有数据的 SqlDataReader 对象。代码如下：

```
conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");          //连接数据库
conn.Open();                                                              //打开数据库
SqlCommand cmd = new SqlCommand();                                        //创建 SqlCommand 对象
cmd.Connection = conn;                                                    //设置对象的连接
cmd.CommandText = "select * from tb_command";                             //设置 SQL 语句
cmd.CommandType = CommandType.Text;                                       //设置以文本形式执行 SQL 语句
//使用 ExecuteReader 方法创建 SqlDataReader 对象
SqlDataReader sdr = cmd.ExecuteReader();
```

16.4.2　判断查询结果中是否有值

可以通过 SqlDataReader 对象的 HasRows 属性获取一个值，该值指示 SqlDataReader 是否包含一行或多行，即判断查询结果中是否有值。其语法格式如下：

```
public override bool HasRows { get; }
```

属性值：如果 SqlDataReader 包含一行或多行，则为 True；否则为 False。

例 16.7 创建一个 Windows 应用程序，向 Form1 窗体中添加一个 TextBox 控件和一个 Button 控件，分别用于输入要查询的表名以及执行查询操作，然后通过 SqlDataReader 对象的 HasRows 属性进行判断，如果 SqlDataReader 包含一行或多行，则为 True；否则为 False。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.7**）

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        //实例化 SqlConnection 变量 conn
        SqlConnection conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
        conn.Open();                                                      //打开连接
        //创建一个 SqlCommand 对象
        SqlCommand cmd = new SqlCommand("select * from "+textBox1.Text.Trim(), conn);
        //使用 ExecuteReader 方法创建 SqlDataReader 对象
        SqlDataReader sdr = cmd.ExecuteReader();
        sdr.Read();                                                       //调用 Read 方法读取 SqlDataReader
        if (sdr.HasRows)                                                  //使用 HasRows 属性判断结果中是否有数据
        {
            MessageBox.Show("数据表中有值");                              //弹出提示信息
        }
        else
        {
            MessageBox.Show("数据表中没有任何数据");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```


熊猫爱中国 xiongmaoi.com

程序运行结果如图 16.7 所示。

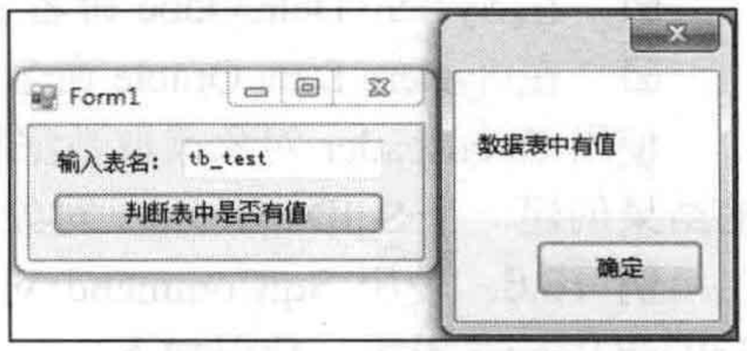

图 16.7　判断指定的数据表中是否有值

16.4.3　读取数据

如果要读取数据表中的数据，可以使用 SqlCommand 对象的 ExecuteReader 方法创建一个 SqlDataReader 对象，再调用 SqlDataReader 对象的 Read 方法读取数据。Read 方法使 SqlDataReader 前进到下一条记录，SqlDataReader 的默认位置在第 1 条记录前面，因此必须调用 Read 方法访问数据。

注意　对于每个关联的 SqlConnection，一次只能打开一个 SqlDataReader，在第 1 个关闭之前，打开另一个的任何尝试都将失败。

Read 方法的语法格式如下：

```
public override bool Read()
```

返回值：如果存在多行，则为 True；否则为 False。

在使用完 SqlDataReader 对象后，要使用 Close 方法关闭 SqlDataReader 对象。其语法格式如下：

```
public override void Close()
```

例 16.8　关闭 SqlDataReader 对象，代码如下：

```
//实例化 SqlConnection 变量 conn
SqlConnection conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
//打开连接
conn.Open();
//创建一个 SqlCommand 对象
SqlCommand cmd = new SqlCommand("select * from "+textBox1.Text.Trim(), conn);
//使用 ExecuteReader 方法创建 SqlDataReader 对象
SqlDataReader sdr = cmd.ExecuteReader();
sdr.Close();
```

注意　在使用 SqlDataReader 对象之前，必须打开数据库连接。如果针对一个 SqlConnection 创建多个 SqlDataReader 对象，在创建下一个 SqlDataReader 对象之前，要通过 Close 方法关闭上一个 SqlDataReader 对象。

16.5　数据适配器：DataAdapter 对象

视频讲解：光盘\TM\第 16 章\视频 16.5\数据适配器：DataAdapter 对象.exe

16.5.1　DataAdapter 对象概述

DataAdapter 对象是一个数据适配器对象，在 DataSet 与数据源之间起到桥梁的作用。DataAdapter 对象提供了 4 个属性，实现与数据源之间的互通。

- ☑ SelectCommand 属性：向数据库发送查询 SQL 语句。
- ☑ DeleteCommand 属性：向数据库发送删除 SQL 语句。
- ☑ InsertCommand 属性：向数据库发送插入 SQL 语句。

☑ UpdateCommand 属性：向数据库发送更新 SQL 语句。

在对数据库进行操作时，只要将这 4 个属性设置成相应的 SQL 语句即可。

另外，DataAdapter 对象中还有几个主要的方法，分别介绍如下。

（1）Fill 方法

Fill 方法主要用于填充 DataSet 数据集。其语法格式如下：

```
public int Fill(DataSet dataSet,string srcTable)
```

☑ dataSet：要用记录和架构（如果必要）填充的 DataSet。

☑ srcTable：用于表映射的源表的名称。

☑ 返回值：已在 DataSet 中成功添加或刷新的行数，这不包括受不返回行的语句影响的行。

（2）Update 方法

Update 方法用于更新数据库。在更新数据库时，DataAdapter 将调用 DeleteCommand、InsertCommand 以及 UpdateCommand 属性。其语法格式如下：

```
public int Update(DataTable dataTable)
```

☑ dataTable：用于更新数据源的 DataTable。

☑ 返回值：DataTable 中成功更新的行数。

例如，使用 DataAdapter 对象的 Fill 方法从数据源中提取数据并填充到 DataSet 时，就会用到 SelectCommand 属性中设置的命令对象。

16.5.2　填充 DataSet 数据集

利用 DataAdapter 对象的 Fill 方法可以填充 DataSet 数据集，该方法使用 Select 语句从数据源中检索数据。这里需要注意的是，与 Select 命令关联的 Connection 对象必须有效，但不需要将其打开，因为 DataAdapter 对象会自动打开和关闭数据库。

例 16.9　创建一个 Windows 应用程序，向 Form1 窗体中添加一个 Button 控件和一个 DataGridView 控件，分别用于执行数据绑定以及显示数据表中的数据（当单击 Button 控件后，程序首先连接数据库）；然后创建一个 SqlDataAdapter 对象，使用该对象的 Fill 方法填充 DataSet 数据集；最后设置 DataGridView 控件的数据源，显示查询的数据。代码如下：**（实例位置：光盘\TM\第 16 章\例 16.9）**

```
SqlConnection conn;
private void button1_Click(object sender, EventArgs e)
{
    //实例化 SqlConnection 变量 conn
    conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
    //创建一个 SqlCommand 对象
    SqlCommand cmd=new SqlCommand("select * from tb_command",conn);
    SqlDataAdapter sda = new SqlDataAdapter();          //创建一个 SqlDataAdapter 对象
    sda.SelectCommand = cmd;                            //设置 SqlDataAdapter 对象的 SelectCommand 属性为 cmd
    DataSet ds = new DataSet();                         //创建一个 DataSet 对象
    sda.Fill(ds," tb_command ");                        //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet 数据集
    dataGridView1.DataSource = ds.Tables[0];            //设置 dataGridView1 控件的数据源
}
```

程序运行结果如图 16.8 所示。

16.5.3　更新数据源

使用 DataAdapter 对象的 Update 方法，可以将 DataSet（在 16.6 节中将会详细介绍 DataSet 对象）中修

熊猫爱中国 xiongmaoi.com

改过的数据及时更新到数据库中。在调用 Update 方法之前，要实例化一个 CommandBuilder 类，该类能自动根据 DataAdapter 的 SelectCommand 属性指定的 SQL 语句判断其他的 InsertCommand、UpdateCommand 和 DeleteCommand，这样就不用设置 DataAdapter 的 InsertCommand、UpdateCommand 和 DeleteCommand 属性，而是直接使用 DataAdapter 的 Update 方法来更新 DataSet、DataTable 或 DataRow 数组。

例 16.10 创建一个 Windows 应用程序，查询 tb_command 表中的所有数据并显示在 DataGridView 控件中，单击某条数据，会显示其详细信息。当对某条数据进行修改后，单击“修改”按钮，将调用 DataAdapter 对象的 Update 方法更新数据源。代码如下：**（实例位置：光盘\TM\第 16 章\例 16.10）**

```
SqlConnection conn;                                          //声明一个 SqlConnection 变量
DataSet ds;                                                  //声明一个 DataSet 变量
SqlDataAdapter sda;                                          //声明一个 SqlDataAdapter 变量
private void Form1_Load(object sender, EventArgs e)
{
    //实例化 SqlConnection 变量 conn，连接数据库
    conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
    //创建一个 SqlCommand 对象
    SqlCommand cmd = new SqlCommand("select * from tb_command", conn);
    sda = new SqlDataAdapter();                              //实例化 SqlDataAdapter 对象
    sda.SelectCommand = cmd;                                 //设置 SqlDataAdapter 对象的 SelectCommand 属性为 cmd
    ds = new DataSet();                                      //实例化 DataSet
    sda.Fill(ds, " tb_command") ;                            //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
    dataGridView1.DataSource = ds.Tables[0];                 //设置 dataGridView1 控件的数据源
}
private void button1_Click(object sender, EventArgs e)
{
    DataTable dt = ds.Tables["tb_command"];                  //创建一个 DataTable
    sda.FillSchema(dt, SchemaType.Mapped);                   //把表结构加载到 tb_command 表中
    DataRow dr = dt.Rows.Find(txtNo.Text);                   //创建一个 DataRow
    //设置 DataRow 中的值
    dr["姓名"] = txtName.Text.Trim();
    dr["性别"] = this.txtSex.Text.Trim();
    dr["年龄"] = this.txtAge.Text.Trim();
    dr["奖金"] = this.txtJJ.Text.Trim();
    //实例化一个 SqlCommandBuilder
    SqlCommandBuilder cmdbuider = new SqlCommandBuilder(sda);
    //调用其 Update 方法将 DataTable 更新到数据库中
    sda.Update(dt);
}
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    //在 dataGridView1 控件的 CellClick 事件中实现单击某条数据显示其详细信息
    txtNo.Text = dataGridView1.SelectedCells[0].Value.ToString();
    txtName.Text = dataGridView1.SelectedCells[1].Value.ToString();
    txtSex.Text = dataGridView1.SelectedCells[2].Value.ToString();
    txtAge.Text = dataGridView1.SelectedCells[3].Value.ToString();
    txtJJ.Text = dataGridView1.SelectedCells[4].Value.ToString();
}
```

注意 在使用 DataTable 对象的 Rows.Find 方法时，所操作的表必须设置主键。

熊猫爱中国 xiongmaoi.com

程序运行结果如图 16.9 所示。

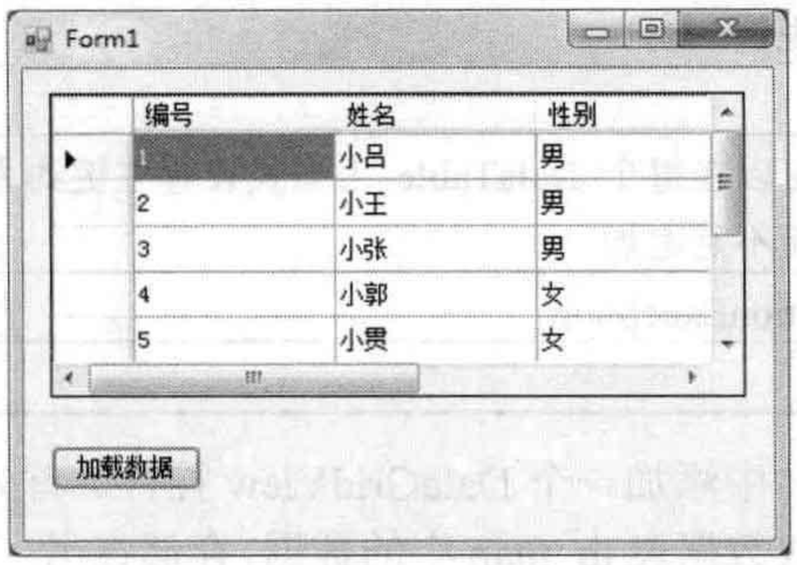

图 16.8　使用 Fill 方法填充 DataSet 数据集

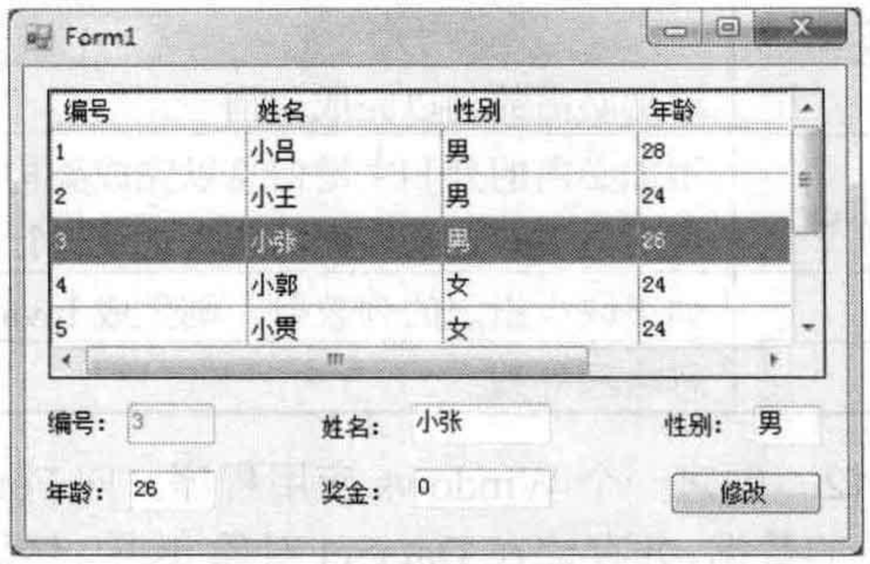

图 16.9　更新数据源

16.6　数据集：DataSet 对象

视频讲解：光盘\TM\第 16 章\视频 16.6\数据集：DataSet 对象.exe

16.6.1　DataSet 对象概述

DataSet 对象就像存放在内存中的一个小型数据库，可以包含数据表、数据列、数据行、视图、约束以及关系等。通常，DataSet 的数据来源于数据库或者 XML，为了从数据库中获取数据，需要使用数据适配器（DataAdapter）从数据库中得到数据。

说明　关于数据适配器（DataAdapter）在 16.5 节已经作过介绍，此处不再赘述。

例 16.11　使用数据适配器（DataAdapter）从数据库中查询数据，并调用其 Fill 方法填充 DataSet 对象。代码如下：

```
conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");          //连接数据库
DataSet ds = new DataSet();                                               //创建一个 DataSet
SqlDataAdapter sda = new SqlDataAdapter("select * from tb_test", conn);   //创建一个 SqlDataAdapter 对象
sda.Fill(ds);                                                             //使用 Fill 方法填充 DataSet
```

16.6.2　合并 DataSet 内容

使用 DataSet 的 Merge 方法可以将 DataSet、DataTable 或 DataRow 中的内容并入现有 DataSet（该方法主要用来将指定的 DataSet 及其架构与当前的 DataSet 进行合并），在此过程中，将根据给定的参数保留或放弃当前 DataSet 中的更改并处理不兼容的架构。其语法格式如下：

```
public void Merge(DataSet dataSet,bool preserveChanges,MissingSchemaAction  missingSchemaAction)
```

☑　dataSet：其数据和架构将被合并的 DataSet。

☑　preserveChanges：要保留当前 DataSet 中的更改则为 True，否则为 False。

☑　missingSchemaAction：MissingSchemaAction 枚举值之一。

MissingSchemaAction 枚举成员及说明如表 16.3 所示。

熊猫爱中国
xiongmaoi.com

表 16.3　MissingSchemaAction 枚举成员及说明

枚举成员	说　明
Add	添加必需的列以完成架构
AddWithKey	添加必需的列和主键信息以完成架构，用户可以在每个 DataTable 上显式设置主键约束，从而确保对与现有记录匹配的传入记录进行更新，而不是追加
Error	如果缺少指定的列映射，则生成 InvalidOperationException
Ignore	忽略额外列

例 16.12　创建一个 Windows 应用程序，向 Form1 窗体中添加一个 DataGridView 控件。首先获取数据表 tb_test 中的数据，并存储在 DataSet 对象 ds 中；然后再获取数据表 tb_man 中的数据，存储在另一个 DataSet 对象 ds1 中；最后调用 DataSet 对象的 Merge 方法将 ds 与 ds1 合并，并将合并后的 DataSet 作为 DataGridView 控件的数据源。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.12**）

```
SqlConnection conn;
private void Form1_Load(object sender, EventArgs e)
{
    //实例化 SqlConnection 对象 conn，连接数据库
    conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
    //创建两个 DataSet
    DataSet ds = new DataSet();
    DataSet ds1 = new DataSet();
    //创建一个 SqlDataAdapter 对象
    SqlDataAdapter sda = new SqlDataAdapter("select * from tb_test", conn);
    sda.Fill(ds);                                                    //使用 Fill 方法填充 DataSet
    //创建一个 SqlDataAdapter 对象
    SqlDataAdapter sda1 = new SqlDataAdapter("select * from tb_man", conn);
    SqlCommandBuilder sbl = new SqlCommandBuilder(sda1);    //创建一个 SqlCommandBuilder 对象
    sda1.Fill(ds1);                                                  //使用 Fill 方法填充 DataSet
    ds1.Merge(ds,true,MissingSchemaAction.AddWithKey);    //使用 Merge 方法将 ds 合并到 ds1 中
    dataGridView1.DataSource = ds1.Tables[0];                        //设置 dataGridView1 控件的数据源
}
```

程序运行结果如图 16.10 所示。

16.6.3　复制 DataSet 内容

为了在不影响原始数据的情况下使用数据，或者使用 DataSet 中数据的子集，可以创建 DataSet 的副本。复制 DataSet 时，可以指定以下内容。

☑　创建 DataSet 的原样副本，其中包含架构、数据、行状态信息和行版本。

☑　创建包含现有 DataSet 的架构但仅包含已修改行的 DataSet。可以返回已修改的所有行或者指定特定的 DataRowState。有关行状态的更多信息可参见行状态与行版本。

☑　仅复制 DataSet 的架构(即关系结构)，而不复制任何行。可以使用 ImportRow 将行导入现有 DataSet 数据集中的 DataTable。

使用 DataSet 对象的 Copy 方法可以创建包含其架构和数据的 DataSet 的原样副本，该方法的主要作用是复制指定 DataSet 的结构和数据。其语法格式如下：

```
public DataSet Copy()
```

返回值：新的 DataSet，具有与该 DataSet 相同的结构（表架构、关系和约束）和数据。

例 16.13　创建一个 Windows 应用程序，向 Form1 窗体中添加两个 DataGridView 控件和一个 Button 控件。其中，第 1 个 DataGridView 控件用于显示数据表 tb_test 中的数据；当单击 Button 控件后，通过 DataSet

熊猫爱中国 xiongmaoi.com

对象的 Copy 方法复制第 1 个 DataGridView 控件的 DataSet 数据源，并作为第 2 个 DataGridView 控件的数据源。代码如下：（**实例位置：光盘\TM\第 16 章\例 16.13**）

```
SqlConnection conn;                                          //声明一个 SqlConnection 变量
DataSet ds;                                                  //声明一个 DataSet 变量
private void Form1_Load(object sender, EventArgs e)
{
    //实例化 SqlConnection 变量 conn，连接数据库
    conn = new SqlConnection("server=.;database=db_14;uid=sa;pwd=");
    //创建一个 SqlCommand 对象
    SqlCommand cmd = new SqlCommand("select * from tb_test",conn);
    SqlDataAdapter sda = new SqlDataAdapter();               //创建一个 SqlDataAdapter 对象
    //设置 SqlDataAdapter 对象的 SelectCommand 属性，设置执行的 SQL 语句
    sda.SelectCommand = cmd;
    ds = new DataSet();                                      //实例化 DataSet
    sda.Fill(ds,"test");                                     //使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet
    dataGridView1.DataSource = ds.Tables[0];                 //设置 dataGridView1 的数据源
}
private void button1_Click(object sender, EventArgs e)
{
    DataSet ds1 = ds.Copy();                                 //调用 DataSet 的 Copy 方法复制 ds 中的内容
    dataGridView2.DataSource = ds1.Tables[0];                //将 ds1 作为 dataGridView2 的数据源
}
```

程序运行结果如图 16.11 所示。

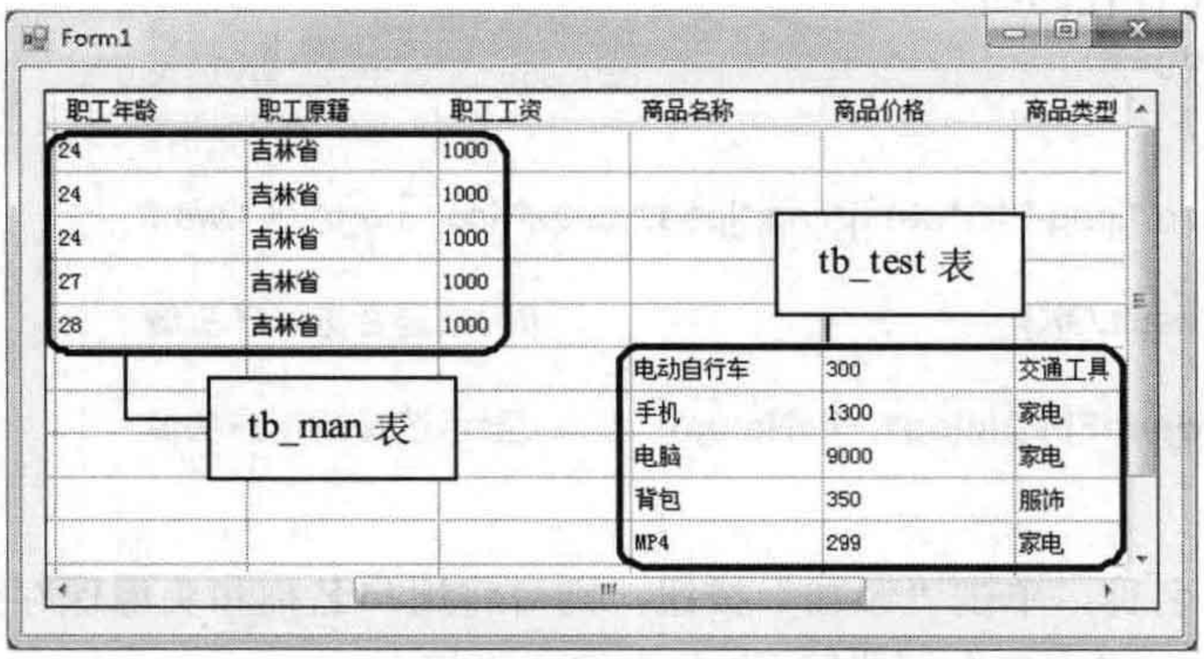

图 16.10　合并 DataSet

图 16.11　复制 DataSet

16.7　实　　战

视频讲解：光盘\TM\第 16 章\视频 16.7\实战.exe

16.7.1　使用二进制存储用户头像

在一个完善的客户管理系统中，图像数据的存取是必不可少的，如用户头像等信息。在下面的实例中，将使用 C#实现使用二进制存取用户头像的功能。

例 16.14　使用 C#实现使用二进制存储用户头像功能，程序开发步骤如下：（**实例位置：光盘\TM\第 16 章\例 16.14**）

熊猫爱中国 xiongmaoi.com