第5章 ADO.NET技术

本章学习目标

- 理解 ADO.NET 的相关概念及其结构；
- 理解 ADO.NET 的五大对象；
- 掌握数据库的两种访问模式及其区别；
- 掌握使用 ADO.NET 技术操作数据的常用方法。

本章介绍 ADO.NET 技术，主要讲解 ADO.NET 的相关概念、ADO.NET 的结构、五大对象、两种数据库访问模式，最后用案例讲解使用 ADO.NET 技术操作数据的常用方法。

5.1 ADO.NET 简介

5.1.1 ADO.NET 的相关概念

1. 什么是 ADO

微软公司的 ADO（ActiveX Data Objects）是一个用于存取数据源的 COM 组件。COM 是微软公司为了使计算机工业的软件生产更加符合人类的行为方式开发的一种新的软件开发技术。在 COM 构架下人们可以开发出各种各样功能专一的组件，然后将它们按照需要组合起来，构成复杂的应用系统。

ADO 提供了编程语言和统一数据访问方式（OLE DB）的一个中间层。OLE 的全称是 Object Link and Embed，即对象连接与嵌入。ADO 允许开发人员编写访问数据的代码而不用关心数据库是如何实现的，只用关心数据库的连接。

ADO 最普遍的用法就是在关系数据库中查询一个表或多个表，然后在应用程序中检索并显示查询结果，还允许用户更改并保存数据。

2. 什么是 ADO.NET

ADO.NET 是微软公司提出的访问数据库的一项新技术。ADO.NET 的名称起源于 ADO，是 ADO 的升级版本，是一个类库，主要用于.NET Framework 平台对数据的操作。它提供了一致的对象模型，可以存取和编辑各种数据源的数据，为这些数据源提供一致的数据处理方式。之所以使用 ADO.NET 这个名称，是因为微软公司希望表明这是在.NET 编程环境中优先使用的数据访问接口。

5.1.2 ADO.NET 的结构

1. ADO.NET 模型

ADO.NET 采用了层次管理模型,各部分之间的逻辑关系如图 5.1 所示。

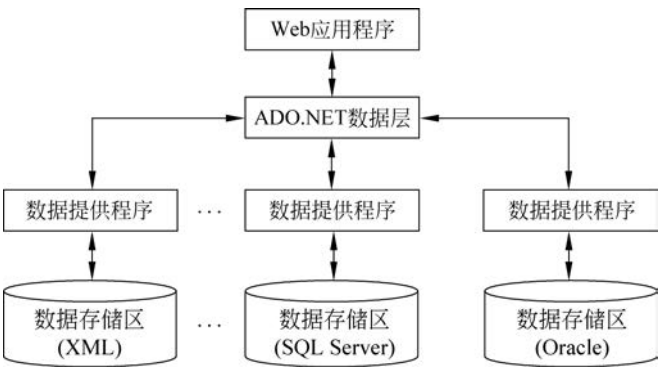

图 5.1 ADO.NET 模型

ADO.NET 模型的最顶层是 Web 应用程序,中间是 ADO.NET 数据层和数据提供程序,在这个层次中数据提供程序相当于 ADO.NET 的通用接口,各种不同的数据源要使用不同的数据提供程序。它相当于一个容器,包括一组类及相关的命令,是数据源 DataSource 与数据集 DataSet 之间的桥梁,负责将数据源中的数据读到数据集中,也可将用户处理完毕的数据集保存到数据源中。

2. ADO.NET 的组件

ADO.NET 提供了两个主要的组件来访问和操作数据,它们分别是.NET Framework 数据提供程序和数据集 DataSet。数据集 DataSet 临时存储应用程序从数据源读取的数据,可以对数据进行各种操作;数据提供程序用于建立数据集和数据源的连接。数据提供程序和数据集之间的关系如图 5.2 所示。

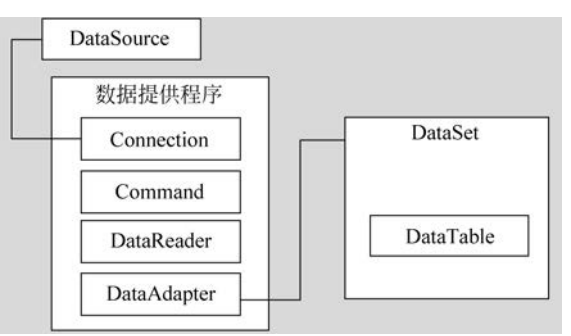

图 5.2 ADO.NET 的结构图

5.2 ADO.NET 的五大对象

ADO.NET 对象是指包含在.NET Framework 数据提供程序和数据集 DataSet 中的对象,其中,DataSet 对象是驻留在内存中的数据库,位于 System.Data 命名空间下。

ADO. NET 从数据库抽取数据后数据就存放在 DataSet 中，故可以把 DataSet 看成一个数据容器。. NET Framework 数据提供程序包括 Connection 对象、Command 对象、DataReader 对象和 DataAdapter 对象。ADO. NET 五大对象之间的关系如图 5. 3 所示。

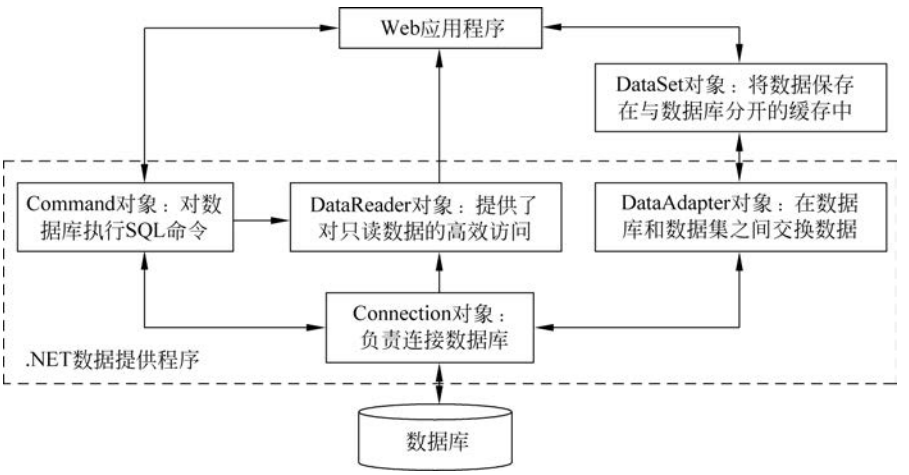

图 5. 3 ADO. NET 的五大对象

ADO. NET 的五大对象可以形象地记为连接 Connection、执行 Command、读取 DataReader、分配 DataAdapter、填充 DataSet。这正是 ADO. NET 对数据库操作的一般步骤。

. NET Framework 数据提供程序包括的 4 个对象提供了对数据库的各种不同的访问功能，对于不同的数据库，区别仅是前缀不同。例如，连接 SQL Server 数据库使用的对象名称为 SqlConnection、SqlCommand、SqlDataReader、SqlDataAdapter，连接 Access 数据库使用的对象名称为 OleDbConnection、OleDbCommand、OleDbDataReader、OleDbDataAdapter，连接 Oracle 数据库使用的对象名称是 OracleConnection、OracleCommand、OracleDataReader、OracleDataAdapter。

5.2.1 Connection 对象

1. Connection 对象概述

在开发 Web 应用程序时需要与数据库进行交互，在和数据库进行交互之前必须实现和数据库的连接。使用 Connection 对象可以实现与数据库的连接。

对于 ADO. NET，不同的数据源对应着不同的 Connection 对象。具体的 Connection 对象如表 5. 1 所示。对于 Command 对象、DataReader 对象和 DataAdapter 对象也是如此，后面不再赘述。

表 5. 1 Connection 对象

名　称	命名空间	描　述
SqlConnection	System. Data. SqlClient	表示与 SQL Server 数据库的连接对象
OleDbConnection	System. Data. OleDb	表示与 OleDb 数据源的连接对象
OdbcConnection	System. Data. Odbc	表示与 ODBC 数据源的连接对象
OracleConnection	System. Data. OracleClient	表示与 Oracle 数据库的连接对象

2. Connection 对象的常用属性

1) ConnectionString 属性

Connection 对象的数据库连接字符串保存在 ConnectionString 属性中,可以使用该属性获取或设置数据库的连接字符串。

2) State 属性

State 属性用来显示当前 Connection 对象的状态,有 Open 和 Closed 两种状态。

3. Connection 对象的常用方法

1) Open()方法

使用 Open()方法打开数据库连接。

2) Close()方法

使用 Close()方法关闭数据库连接。

4. 使用 Connection 对象连接数据库

ADO.NET 使用 SqlConnection 对象与 SQL Server 进行连接。下面介绍如何连接 SQL Server 数据库。

1) 定义数据库连接字符串

定义连接字符串的常用方式有两种。

(1) 使用 Windows 身份验证。

该方式也称为信任连接,这种连接方式有助于在连接到 SQL Server 时提供安全保护,因为它不会在连接字符串中公开用户 ID 和密码,是安全级别要求较高时推荐的数据库连接方法。其连接字符串的语法格式如下。

```
string ConnStr = "Server = 服务器名或 IP; Database = 数据库名;Integrated Security = true; ";
```

Data Source(或 Server)指定了 SQL Server 服务器的名字或 IP 地址,可以用 localhost 或圆点“.”表示本机。Integrated Security = true(或 Integrated Security = SSPI,SSPI 是 Security Support Provider Interface 的缩写)表示采用信任连接方式,即用 Windows 账号登录到 SQL Server 数据库服务器。Database(或 Initial Catalog)用于设置登录到哪个数据库中。

(2) 使用 SQL Server 身份验证。

该方式也称为非信任连接,这种连接方式把未登录的用户 ID 和密码写在连接字符串中,因此在安全级别要求较高的场合不要使用。其连接字符串的语法格式如下。

```
string ConnStr = "Server = 服务器名或 IP; Database = 数据库名; uid = 用户名;pwd = 密码";
```

uid 表示 SQL Server 登录用户名,pwd 表示 SQL Server 登录密码。

2) 创建 Connection 对象

这里以创建的数据库连接字符串为参数,调用 SqlConnection 类的构造方法创建 Connection 对象,其语法格式如下。

```
SqlConnection 连接对象名 = new SqlConnection(连接字符串);
```

用户也可以首先使用构造函数创建一个不含参数的 Connection 对象,再通过 Connection 对象的 ConnectionString 属性设置连接字符串,其语法格式如下。

```
SqlConnection 连接对象名 = new SqlConnection();
连接对象名.ConnectionString = 连接字符串;
```

以上两种方法在功能上是等效的，选择哪种方法取决于用户的个人喜好和编码风格，不过属性经过明确设置可以使代码更容易理解和调试。

3）打开数据库连接

```
连接对象名.Open();
```

5. 应用举例

【案例 5.1】 使用 Connection 对象连接数据库。

使用 Connection 对象建立与 SQL Server 数据库 test 的连接，并显示当前数据库的连接状态。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“使用 Connection 对象连接数据库”。在页面中添加一个标签控件和两个命令按钮，两个命令按钮的 Text 属性分别为“打开连接”和“关闭连接”。

3）编写程序代码

添加命名空间的引用，代码如下。

```
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象，代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

在页面载入时执行的事件代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "当前连接状态是：" + conn.State.ToString();
}
```

“打开连接”按钮被单击时执行的事件代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    conn.Open();Label1.Text = "当前连接状态是：" + conn.State.ToString();
}
```

“关闭连接”按钮被单击时执行的事件代码如下。

```
protected void Button2_Click(object sender, EventArgs e)
{
    conn.Close();Label1.Text = "当前连接状态是：" + conn.State.ToString();
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面，显示的连接状态是 Closed，打开连接时显示的连接状态是 Open，运行效果如图 5.4 所示。

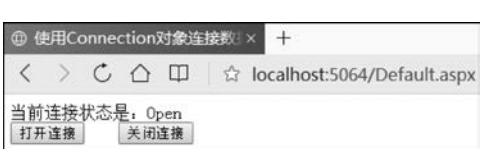

图 5.4 使用 Connection 对象连接数据库

5.2.2 Command 对象

1. Command 对象概述

ADO.NET 的 Command 对象就是 SQL 命令或者对存储过程的引用。除了检索、更新数据之外,Command 对象可用来对数据源执行一些不返回结果集的查询任务,以及用来执行改变数据源结构的数据定义命令。

在使用 Connection 对象与数据源建立连接后,可以使用 Command 对象对数据源执行查询、添加、删除和修改等各种操作,操作的实现可以使用 SQL 语句,也可以使用存储过程。根据所用的.NET Framework 数据提供程序的不同,Command 对象可以分成 4 种,分别是 SqlCommand、OleDbCommand、OdbcCommand 和 OracleCommand,在实际的编程过程中应该根据访问的数据源不同选择相应的 Command 对象。

2. Command 对象的常用属性

(1) Connection 属性:获取或设置此 Command 对象使用的 Connection 对象的名称。

(2) CommandText 属性:获取或设置对数据源执行的 SQL 语句或存储过程名。

(3) CommandType 属性:获取或设置 Command 对象要执行命令的类型。

3. Command 对象的常用方法

(1) ExecuteNonQuery()方法:执行 CommandText 属性指定的内容,并返回受影响的行数。

(2) ExecuteReader()方法:执行 CommandText 属性指定的内容,并创建 DataReader 对象。

(3) ExecuteScalar()方法:执行查询,并返回查询所返回的结果集中第一行的第一列。

4. 创建 Command 对象

Command 对象的构造函数的参数有两个,一个是需要执行的 SQL 语句,另一个是数据库连接对象。这里以它们为参数,调用 SqlCommand 类的构造方法创建 Command 对象,其语法格式如下。

```
SqlCommand 命令对象名 = new SqlCommand (SQL 语句,连接对象);
```

用户也可以首先使用构造函数创建一个不含参数的 Command 对象,再设置 Command 对象的 Connection 属性和 CommandText 属性,其语法格式如下。

```
SqlCommand 命令对象名 = new SqlCommand ();
命令对象名.Connection = 连接对象;
命令对象名.CommandText = SQL 语句;
```

5.2.3 DataReader 对象

1. DataReader 对象概述

当 Command 对象返回结果集时需要使用 DataReader 对象来检索数据。DataReader 对象返回一个来自 Command 的只读的、只能向前的数据流。DataReader 每次只能在内存中保留一行,所以开销非常小,提高了应用程序的性能。

由于 DataReader 只执行读操作，并且每次只在内存缓冲区里存储结果集中的一条数据，所以使用 DataReader 对象的效率比较高，如果要查询大量数据，同时不需要随机访问和修改数据，DataReader 是优先的选择。

2. DataReader 对象的常用属性

(1) FieldCount 属性：表示由 DataReader 得到的一行数据中的字段数。

(2) HasRows 属性：表示 DataReader 是否包含数据。

(3) IsClosed 属性：表示 DataReader 对象是否关闭。

3. DataReader 对象的常用方法

(1) Read()方法：返回 SqlDataReader 的第一条，并一条一条地向下读取。

(2) GetName()方法：通过输入列索引获得该列的名称。

(3) GetDataTypeName()方法：通过输入列索引获得该列的类型。

(4) GetValue()方法：根据传入的列的索引值返回当前记录行里指定列的值。

(5) Close ()方法：关闭 DataReader 对象。

(6) IsNull()方法：判断指定索引号的列的值是否为空，返回 true 或 false。

4. 创建 DataReader 对象

如果要创建一个 DataReader 对象，可以通过 Command 对象的 ExecuteReader()方法，其语法格式如下。

```
SqlDataReader 数据读取器对象 = 命令对象名. ExecuteReader();
```

5.2.4 DataAdapter 对象

1. DataAdapter 对象概述

DataAdapter(即数据适配器)对象是一种用来充当 DataSet 对象与实际数据源之间桥梁的对象。DataSet 对象是一个非连接的对象，它与数据源无关。DataAdapter 正好负责填充它，并把它的数据提交给一个特定的数据源，它与 DataSet 配合使用可以执行新增、查询、修改和删除等多种操作。

DataAdapter 对象是一个双向通道，用来把数据从数据源中读到一个内存表中，以及把内存中的数据写回到一个数据源中。这两种情况下使用的数据源可能相同，也可能不相同。这两种操作分别称为填充(Fill)和更新(Update)。

2. DataAdapter 对象的常用属性

(1) SelectCommand 属性：获取或设置一个语句或存储过程，在数据源中选择记录。

(2) UpdateCommand 属性：获取或设置一个语句或存储过程，更新数据源中的记录。

(3) InsertCommand 属性：获取或设置一个语句或存储过程，在数据源中插入新记录。

(4) DeleteCommand 属性：获取或设置一个语句或存储过程，从数据源中删除记录。

3. DataAdapter 对象的常用方法

(1) Fill()方法：把从数据源读取的数据行填充到 DataSet 对象中。

(2) Update()方法：在 DataSet 对象中的数据有所改动后更新数据源。

4. 创建 DataAdapter 对象

DataAdapter 对象的构造函数的参数有两个,一个是需要执行的 SQL 语句,另一个是数据库连接对象。这里以它们为参数,调用 SqlDataAdapter 类的构造方法创建 DataAdapter 对象,其语法格式如下。

```
SqlDataAdapter 数据适配器对象名 = new SqlDataAdapter (SQL 语句,连接对象);
```

用户也可以首先使用构造函数创建一个不含参数的 DataAdapter 对象,再设置 DataAdapter 对象的 Connection 属性和 CommandText 属性,其语法格式如下。

```
SqlDataAdapter 数据适配器对象名 = new SqlDataAdapter();
数据适配器对象名.Connection = 连接对象;
数据适配器对象名.CommandText = SQL 语句;
```

5.2.5 DataSet 对象

1. DataSet 对象概述

DataSet 对象是 ADO.NET 的核心组件之一。ADO.NET 从数据库抽取数据后将数据存放在 DataSet 中,故可以把 DataSet 看成一个数据容器,或称为"内存中的数据库"。

DataSet 从数据源中获取数据后就断开了与数据源之间的连接。用户可以在 DataSet 中对记录进行插入、删除、修改、查询、统计等,在完成了各项操作后还可以把 DataSet 中的数据送回数据源。

每一个 DataSet 都是一个或多个 DataTable 对象的集合,DataTable 相当于数据库中的表。DataTable 对象的常用属性主要有 Columns 属性、Rows 属性和 DefaultView 属性。

- Columns 属性:用于获取 DataTable 对象中表的列集合。
- Rows 属性:用于获取 DataTable 对象中表的行集合。
- DefaultView 属性:用于获取表的自定义视图。

2. DataSet 对象的常用属性

(1) Tables:获取包含在 DataSet 中的表的集合。用户可以通过索引来引用 Tables 集合中的一个表,例如 Tables[i]表示第 i 个表,其索引值从 0 开始编号。

(2) DataSetName:获取或设置当前 DataSet 的名称。

3. DataSet 对象的常用方法

(1) Clear()方法:删除 DataSet 对象中的所有表。

(2) Copy()方法:复制 DataSet 对象的结构和数据到另外一个 DataSet 对象中。

4. 创建 DataSet 对象

使用程序代码创建 DataSet 对象有两种方法。

第一种方法是先创建一个空的数据集对象,再把建立的数据表放到该数据集中,这种方法的语法格式如下。

```
DataSet 数据集对象名 = new DataSet();
```

第二种方法是先建立数据表,再创建包含数据表的数据集,这种方法的语法格式如下。

```
DataSet 数据集对象名 = new DataSet("表名");
```

5.3 数据库访问模式

通过 ADO. NET 执行数据库操作的过程如下。

(1) 导入相应的命名空间。

(2) 使用 Connection 对象建立与数据库的连接。

(3) 使用 Command 对象或 DataAdapter 对象对数据库执行 SQL 命令,实现对数据库的查询、插入、更新和删除操作。

(4) 通过 DataSet 对象或 DataReader 对象访问数据。

(5) 使用数据显示控件或输出语句显示数据。

ADO. NET 访问数据库的过程如图 5.5 所示。

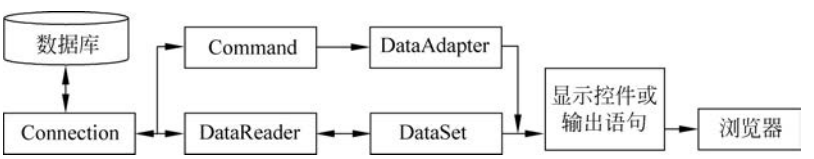

图 5.5 ADO. NET 访问数据库的过程

由图 5.5 可知,在 ADO. NET 中有两种访问数据库的模式,一种是连接模式,在保持数据库连接的方式下通过执行指定的 SQL 语句完成对数据的操作,数据的操作在断开数据库连接之前;另一种是断开模式,先将数据库中的数据读取到服务器的 DataSet 或者 DataTable 中,数据的操作在断开数据库连接之后。

5.3.1 连接模式

1. 使用连接模式访问数据库

(1) 创建 Connection 对象与数据库建立连接。

(2) 创建 Command 对象对数据库执行 SQL 命令或存储过程,包括增、删、改及查询数据库等命令。

(3) 打开与数据库的连接。

(4) 执行操作数据库的命令,如果查询数据库的数据,则创建 DataReader 对象读取 Command 命令查询到的结果集,并将查询到的结果集绑定到控件上。

(5) 关闭与数据库的连接。

2. 操作数据库的两种方法

Command 对象提供了多种完成数据库操作的方法,下面介绍常用的两种方法。

1) ExecuteReader()方法

ExecuteReader()方法提供了顺序读取数据库的方法,该方法根据提供的 select 语句返回一个 DataReader 对象,开发人员可以使用 DataReader 对象的 Read()方法循环依次读取每条记录中各字段的内容。

【案例 5.2】 使用 ExecuteReader()方法读取数据。

将 test 数据库的 staff 表中的姓名和职称字段显示在 ListBox 控件中。

(1) 新建一个空网站。

方法略。

(2) 新建网页 Default.aspx。

设置网页标题为“使用 ExecuteReader()方法读取数据”。在页面中添加一个 ListBox 控件,其 ID 属性采用默认的 ListBox1。

(3) 编写程序代码。

添加命名空间的引用,代码如下。

```
using System.Data.SqlClient;
```

在页面载入时执行的事件代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    string constr = "Server = .;Database = test;Integrated Security = true";
    SqlConnection conn = new SqlConnection(constr);
    SqlCommand comm = new SqlCommand();
    comm.Connection = conn;
    comm.CommandText = "select * from staff";
    conn.Open();
    SqlDataReader dr = comm.ExecuteReader();
    while(dr.Read())
    {
        this.ListBox1.Items.Add(string.Format("{0}\t{1}",dr[1],dr[3]));
    }
    conn.Close();
}
```

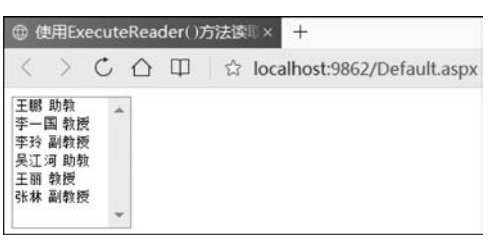

图 5.6 使用 ExecuteReader()方法读取数据

(4) 运行页面。

按 Ctrl+F5 组合键运行页面,在列表框中显示了 staff 表中的姓名和职称字段,运行效果如图 5.6 所示。

2) ExecuteNonQuery()方法

ExecuteNonQuery()方法执行 SQL 语句,并返回因操作受影响的行数。一般将其用于 update、insert、delete 或 select 语句直接操作数据库中的表数据。对于 update、insert、delete 语句,ExecuteNonQuery()方法的返回值为该语句所影响的行数;而对于 select 语句,由于执行 select 语句后数据库并无变化,所以其返回值为-1。

【案例 5.3】 使用 ExecuteNonQuery()方法更新数据。

根据输入的金额数,将 test 数据库的 staff 表中职称为“助教”的职工的工资增加。

(1) 新建一个空网站。

方法略。

(2) 新建网页 Default.aspx。

设置网页标题为“使用 ExecuteNonQuery()方法更新数据”。在页面中添加一个 GridView 控件、一个 Label 控件、一个 TextBox 控件和一个 Button 控件,Label 控件的 Text 属性为“增加工资值:”,Button 控件的 Text 属性为“加工资”,所有控件的 ID 均采用默

认名称。

(3) 编写程序代码。

添加命名空间的引用,代码如下。

```
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

在页面载入时创建 SqlCommand 对象,设置 SQL 查询语句,打开数据库连接,使用 SqlDataReader 对象来获取数据源,然后定义 GridView 控件的标题,把数据源绑定到 GridView 控件上,并显示出来。Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    SqlCommand comm = new SqlCommand();
    comm.Connection = conn;
    comm.CommandText = "select * from staff";
    conn.Open();
    SqlDataReader dr = comm.ExecuteReader();
    this.GridView1.Caption = "加工资前的职工信息表";
    this.GridView1.DataSource = dr;
    this.GridView1.DataBind();
    conn.Close();
}
```

“加工资”按钮被单击时执行的事件代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = "update staff set Salary = Salary + " + this.TextBox1.Text.Trim()
        + "where Title = '助教'";
    conn.Open();
    int iValue = cmd.ExecuteNonQuery();
    if (iValue > 0)
    {
        cmd.CommandText = "select * from staff";
        SqlDataReader dr = cmd.ExecuteReader();
        this.GridView1.Caption = "加工资后的职工信息表";
        this.GridView1.DataSource = dr;
        this.GridView1.DataBind();
        TextBox1.Text = "";
    }
    conn.Close();
}
```

(4) 运行页面。

按 Ctrl+F5 组合键运行页面,加工资前的职工信息表如图 5.7 所示。

假设要给助教增加 350 元工资,则在文本框中输入 350,单击“加工资”按钮,加工资后的职工信息表如图 5.8 所示。

图 5.7　加工资前的职工信息表

图 5.8　加工资后的职工信息表

5.3.2　断开模式

1. 断开模式概述

DataSet 对象包含多个 DataTable 对象,用于存储与数据源断开连接的数据。DataAdapter 对象可以作为数据库和内存之间的桥梁,使用 DataAdapter 对象的 Fill()方法可以提取查询结果并填充到 DataTable 中,然后关闭连接,此时处于非连接状态,然后应用程序继续处理离线的 DataSet 数据。

2. 使用断开模式访问数据库

1) 使用断开模式查询数据的步骤

(1) 创建 Connection 对象与数据库建立连接。

(2) 创建 DataAdapter 对象,并设置 select 语句。

(3) 创建 DataSet 对象或者 DataTable 对象。

(4) 使用 DataAdapter 的 Fill()方法填充 DataSet。

(5) 使用数据控件对数据进行显示。

2) 使用断开模式编辑数据的步骤

(1) 创建 Connection 对象与数据库建立连接。

(2) 创建 DataAdapter 对象,并设置 select 语句。

(3) 创建 CommandBuilder 对象。

(4) 创建 DataSet 对象或者 DataTable 对象。

(5) 使用 DataAdapter 的 Fill()方法填充 DataSet。

(6) 使用数据控件对数据进行插入、更新或删除操作。

(7) 调用 DataAdapter 对象的 Update()方法更新数据库。

3. 应用举例

【案例 5.4】 使用 DataAdapter 对象添加数据。

使用 DataAdapter 对象对 test 数据库中的 staff 表进行操作,添加新记录。

1) 新建一个空网站

方法略。

2) 新建网页 Default.aspx

设置网页标题为"使用 DataAdapter 对象添加数据"。在页面中添加一个 GridView 控件、5 个 Label 控件、5 个 TextBox 控件和一个 Button 控件,所有控件的 ID 均采用默认名称,设计视图如图 5.9 所示。

图 5.9 案例 5.4 的设计视图

3) 编写程序代码

添加命名空间的引用,代码如下。

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接字符串,代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
```

在页面载入时使用断开模式查询数据。先创建 SqlDataAdapter 对象,再创建 DataTable 对象 dt,把 SqlDataAdapter 对象中的数据填充到 dt 中,然后把 dt 绑定到 GridView 控件上显示出来。Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        using (SqlConnection conn = new SqlConnection(constr))
        {
            SqlDataAdapter da = new SqlDataAdapter("select * from staff", conn);
            DataTable dt = new DataTable();
            da.Fill(dt);
            this.GridView1.DataSource = dt;
```

```
                this.DataBind();
            }
        }
    }
```

说明：释放与数据库的连接除了使用 SqlConnection 对象的 Close()方法外，还有一种方法就是使用 using 语句。在 using 语句中不再使用 Close()方法，一旦 using 模块结束，系统立即关闭与相关对象的连接。

在“添加”按钮的单击事件中使用断开模式插入数据，代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    using (SqlConnection conn = new SqlConnection(constr))
    {
        SqlDataAdapter da = new SqlDataAdapter("select * from staff", conn);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        da.InsertCommand = builder.GetInsertCommand();
        DataTable dt = new DataTable();
        da.Fill(dt);
        this.TextBox1.Focus();
        DataRow row = dt.NewRow();
        row[0] = this.TextBox1.Text.Trim();
        row[1] = this.TextBox2.Text.Trim();
        row[2] = this.TextBox3.Text.Trim();
        row[3] = this.TextBox4.Text.Trim();
        row[4] = this.TextBox5.Text.Trim();
        dt.Rows.Add(row);
        da.Update(dt);
        this.GridView1.DataSource = dt;
        this.DataBind();
    }
}
```

说明：创建 SqlCommandBuilder 对象，根据 SqlDataAdapter 对象提供的 select 语句和连接字符串，利用 SqlCommandBuilder 对象的 GetInsertCommand()方法、GetUpdateCommand()方法、GetDeleteCommand()方法为 SqlDataAdapter 对象生成 InsertCommand、UpdateCommand 和 DeleteCommand，这样就可以调用 DataAdapter 对象的 Update 方法更新数据库。

4）运行页面

按 Ctrl+F5 组合键运行页面，运行效果如图 5.10 所示。

图 5.10　使用 DataAdapter 对象添加数据

5.3.3 两种访问模式的区别

连接模式是指客户端始终与数据源保持连接，直到程序结束，这种方式的实时性好，但独占数据库连接，在数据量小和只读的情况下优先选择这种模式。

断开模式是指客户端从数据源获取数据后断开与数据源的连接，所有的数据操作都是针对本地数据缓存里的数据，当需要从数据源获取新数据或者被处理后的数据回传时客户端再与数据源连接完成相应的操作。这种方式不独占数据库连接，但实时性差。断开模式适用于数据量大、需要修改数据同时更新数据库的场合。

5.4 使用 ADO.NET 技术操作数据

5.4.1 数据的添加

1. 数据添加概述

数据添加是程序开发过程中最基本的工作。如果采用把数据绑定到控件的方式，则需要设置每个控件的相关属性，灵活性会受到限制。如果使用 ADO.NET 对象就方便多了。在添加数据的过程中首先创建 SqlConnection 对象和 SqlCommand 对象，然后打开数据库连接，并调用 SqlCommand 对象的 ExecuteNonQuery()方法完成插入操作。

2. 应用举例

【案例 5.5】 添加单条数据。

在页面中输入邮件信息，单击“添加”按钮把该条信息添加到 test 数据库的 send 表中。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“添加单条数据”。在页面中添加一个 6 行 2 列的表格用来布局，把表格的第 1 行、第 5 行和第 6 行的单元格合并，在第 2 行和第 3 行的第 2 列中各添加一个 TextBox 控件，在第 5 行中添加两个 Button 控件，在第 6 行中添加一个 GridView 控件，所有控件的 ID 均采用默认名称。

设置 GridView1 控件的 AutoGenerateColumns 为 false，单击 GridView1 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单中选择“编辑列”选项，在打开的“字段”对话框中给 GridView1 控件绑定 send 表中的 3 个字段，并把 3 个字段的列名称都改成中文，页面设计视图如图 5.11 所示。

3）编写程序代码

添加命名空间的引用，代码如下。

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象，代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

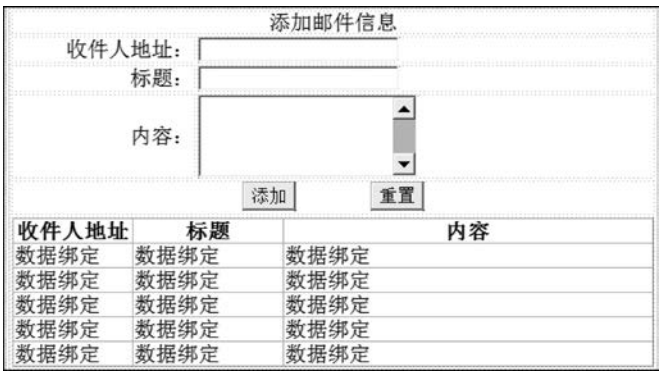

图 5.11　案例 5.5 的设计视图

在页面载入时先创建 SqlDataAdapter 对象,再创建 DataSet 对象 ds,把 SqlDataAdapter 对象中的数据填充到 ds 中,然后把 ds 绑定到 GridView 控件上显示出来。

Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    conn.Open();
    SqlDataAdapter ada = new SqlDataAdapter("select * from send", conn);
    DataSet ds = new DataSet();
    ada.Fill(ds);
    conn.Close();
    GridView1.DataSource = ds;
    GridView1.DataBind();
}
```

“添加”按钮被单击时执行的事件代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        conn.Open();
        string InsertSql = "insert into send values('" + TextBox1.Text + "','" + TextBox2.Text + "',
'" + TextBox3.Text + "')";
        SqlCommand comm = new SqlCommand(InsertSql, conn);
        comm.ExecuteNonQuery();
        SqlDataAdapter ada = new SqlDataAdapter("select * from send", conn);
        DataSet ds = new DataSet();
        ada.Fill(ds);
        conn.Close();
        GridView1.DataSource = ds;
        GridView1.DataBind();
        Response.Write("<script language = javascript>alert('数据添加成功!')</script>");
    }
    catch
    {
        Response.Write("<script language = javascript>alert('数据添加失败!')</script>");
    }
}
```

说明：try 语句块打开数据库连接，创建 SqlCommand 对象，调用 SqlCommand 对象的 ExecuteNonQuery()方法完成插入操作；然后创建 SqlDataAdapter 对象和 DataSet 对象，调用 SqlDataAdapter 的 Fill()方法填充 DataSet，使用 GridView 控件把数据显示出来。catch 语句块用来捕获异常，一旦操作失败，则抛出异常，提示“数据添加失败!”。

“重置”按钮被单击时执行的事件代码如下。

```
protected void Button2_Click(object sender, EventArgs e)
{
    TextBox1.Text = "";
    TextBox2.Text = "";
    TextBox3.Text = "";
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面，运行效果如图 5.12 所示。

图 5.12 添加单条数据

5.4.2 数据的更新

1. 更新指定的数据

在开发网站的过程中，更新指定的数据是常见的操作，可利用 update 语句改变选定行上一列或多列的值。

【案例 5.6】 更新指定的数据。

在页面的表格中选定一行数据，更新其中的一列或多列的值。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“更新指定的数据”。在页面中添加一个 6 行 2 列的表格用来布局，把表格的第 1 行、第 2 行和第 6 行的单元格合并，在第 2 行中添加一个 GridView 控件，在第 3 行至第 5 行的第 2 列中各添加一个 TextBox 控件，在第 6 行中添加一个 Button 控件，所有控件的 ID 均采用默认名称。

设置 GridView1 控件的 AutoGenerateColumns 为 false。单击 GridView1 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单中选择“编辑列”选项，在打开的“字段”对话框中给 GridView1 控件绑定 shopping 表中的 4 个字段，并把 4 个字段的列名称都改成中文；接着在“字段”对话框中添加一个 HyperLinkField 字段，其属性设置代码如下。

```
<asp:HyperLinkField DataNavigateUrlFields = "number" HeaderText = "更新" Text = "更新"
DataNavigateUrlFormatString = "Default.aspx?number = {0}"/>
```

在上面的代码中,DataNavigateUrlFields 是指绑定到超链接的 NavigateUrl 属性的字段,DataNavigateUrlFormatString 是指绑定到超链接的 NavigateUrl 属性的值应用的格式设置。页面设计视图如图 5.13 所示。

图 5.13　案例 5.6 的设计视图

3) 编写程序代码

添加命名空间的引用,代码如下。

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

编写绑定 GridView 控件的自定义方法,代码如下。

```
public void GridViewBind()
{
    conn.Open();
    SqlDataAdapter ada = new SqlDataAdapter("select * from shopping", conn);
    DataSet ds = new DataSet();
    ada.Fill(ds);
    GridView1.DataSource = ds;
    GridView1.DataBind();
    conn.Close();
}
```

在页面载入时先把 shopping 表绑定到 GridView1 控件上显示出来,然后把选定行的 3 个字段值显示在对应的文本框控件中。Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        GridViewBind();
        if (Request.QueryString["number"] != null)
        {
            conn.Open();
            SqlDataAdapter ada = new SqlDataAdapter("select * from shopping where number = " +
```

```
Request.QueryString["number"] + "", conn);
            DataSet ds = new DataSet();
            ada.Fill(ds, "shopping");
            conn.Close();
            DataRowView drv = ds.Tables["shopping"].DefaultView[0];
            this.TextBox1.Text = drv["name"].ToString();
            this.TextBox2.Text = drv["count"].ToString();
            this.TextBox3.Text = drv["price"].ToString();
        }
    }
}
```

“更新”按钮被单击时执行的事件代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        conn.Open();
        string sqltext = "update shopping set name = '" + this.TextBox1.Text + "',count = '" +
this.TextBox2.Text + "', price = '" + this.TextBox3.Text + "' where number = " + Request
["number"];
        SqlCommand comm = new SqlCommand(sqltext, conn);
        comm.ExecuteNonQuery();
        conn.Close();
        GridViewBind();
        Response.Write("<script language = javascript>alert('数据更新成功!')</script>");
    }
    catch
    {
        Response.Write("<script language = javascript>alert('数据更新失败!')</script>");
    }
}
```

4）运行页面

按 Ctrl+F5 组合键运行页面，单击商品编号 17003 一行中的“更新”超链接，在“商品数量”文本框中把牙刷的数量由 50 更新为 80，运行效果如图 5.14 所示。

图 5.14 数据更新之前的效果

单击“更新”按钮，则 GridView1 控件中牙刷的数量由 50 更新成了 80，运行效果如图 5.15 所示。

图 5.15　更新指定的数据

2. 批量更新数据

数据的批量更新是程序开发中经常用到的技术,它可以极大地提高用户的工作效率。在 update 语句中加入 where 条件可以实现批量更新数据的功能。

【案例 5.7】 批量更新数据。

根据选择的职称对具有相同职称的职工的工资做统一调整。

1) 新建一个空网站

方法略。

2) 新建网页 Default.aspx

设置网页标题为"批量更新数据"。在页面中添加一个 3 行 1 列的表格用来布局,在第 2 行中添加一个 GridView 控件,在第 3 行中添加一个 DropDownList 控件、一个 TextBox 控件、一个 Button 控件,所有控件的 ID 均采用默认名称。

设置 GridView1 控件的 AutoGenerateColumns 为 false。单击 GridView1 控件右上方的智能标记按钮,在弹出的"GridView 任务"菜单中选择"编辑列"选项,在打开的"字段"对话框中给 GridView1 控件绑定 staff 表中的 5 个字段,并把 5 个字段的列名称都改成中文。给 DropDownList 控件增加教授、副教授、讲师、助教列表项。页面设计视图如图 5.16 所示。

图 5.16　案例 5.7 的设计视图

3) 编写程序代码

添加命名空间的引用,代码如下。

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

在页面载入时把 staff 表绑定到 GridView1 控件上显示出来,Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    conn.Open();
    SqlDataAdapter ada = new SqlDataAdapter("select * from staff", conn);
    DataSet ds = new DataSet();
    ada.Fill(ds);
    GridView1.DataSource = ds;
    GridView1.DataBind();
    conn.Close();
}
```

“确定更新”按钮被单击时执行的事件代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    conn.Open();
    string sqltext = "update staff set Salary = Salary + " + TextBox1.Text + " where Title = '" +
this.DropDownList1.SelectedItem.Text + "'";
    SqlCommand comm = new SqlCommand(sqltext, conn);
    comm.ExecuteNonQuery();
    SqlDataAdapter ada = new SqlDataAdapter("select * from staff", conn);
    DataSet ds = new DataSet();
    ada.Fill(ds);
    GridView1.DataSource = ds;
    GridView1.DataBind();
}
```

4）运行页面

按 Ctrl+F5 组合键运行页面，在“将职称为”列表中选择“副教授”选项，在文本框中输入拟增加的工资值，运行效果如图 5.17 所示。

单击“确定更新”按钮，则每名职称为副教授的职工的工资都增加了 100 元，运行效果如图 5.18 所示。

图 5.17 批量更新之前的效果

图 5.18 批量更新之后的效果

5.4.3 数据的删除

1. 删除指定的数据

如果数据信息中存在错误或者重复的数据，可以选定该数据，然后将其删除。

【案例 5.8】 删除指定的数据。

在页面中选定 test 数据库的 staff 表中的一行数据删除。

1) 新建一个空网站

方法略。

2) 新建网页 Default.aspx

设置网页标题为“删除指定的数据”。在页面中添加一个 2 行 1 列的表格用来布局,在第 2 行中添加一个 GridView 控件,ID 采用默认名称。

把 GridView1 控件的 AutoGenerateColumns 设置为 false、DataKeyNames 属性设置为主键 Number。单击 GridView1 控件右上方的智能标记按钮,在弹出的“GridView 任务”菜单中选择“编辑列”,在打开的“字段”对话框中给 GridView1 控件绑定 staff 表中的 5 个字段,并把 5 个字段的列名称都改成中文;接着在“字段”对话框中添加一个 CommandField 字段,设置其 ShowDeleteButton 属性为 true,在 GridView 中显示“删除”超链接,代码如下。

```
<asp:CommandField ShowDeleteButton = "true"></asp:CommandField>
```

页面设计视图如图 5.19 所示。

删除指定的数据					
工号	姓名	性别	职称	工资	
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	删除
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	删除

图 5.19 案例 5.8 的设计视图

3) 编写程序代码

添加命名空间的引用,代码如下。

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

在页面载入时把 staff 表绑定到 GridView1 控件上显示出来,Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        conn.Open();
        SqlDataAdapter ada = new SqlDataAdapter("select * from staff", conn);
        DataSet ds = new DataSet();
        ada.Fill(ds);
        GridView1.DataSource = ds;
        GridView1.DataBind();
        conn.Close();
    }
}
```

在 GridView1 中单击要删除数据所在行的“删除”超链接,激发 GridView1 的 RowDeleting 事件实现数据的删除,代码如下。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    conn.Open();
    string sqltext = "Delete from staff where Number = '" +
            GridView1.DataKeys[e.RowIndex].Value + "'";
    SqlCommand com = new SqlCommand(sqltext, conn);
    com.ExecuteNonQuery();
    SqlDataAdapter ada = new SqlDataAdapter("select * from staff", conn);
    DataSet ds = new DataSet();
    ada.Fill(ds);
    conn.Close();
    GridView1.DataSource = ds;
    GridView1.DataBind();
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面，删除之前的效果如图 5.20 所示。

在表中单击最后一行数据所在行的“删除”超链接，执行 RowDeleting 事件后实现了这行数据的删除，效果如图 5.21 所示。

图 5.20 删除之前的效果

图 5.21 删除之后的效果

2. 批量删除数据

批量删除数据在网站及管理系统中的应用比较广泛，主要通过设置 delete 语句后的 where 条件来实现。

【案例 5.9】 批量删除数据。

在页面的下拉列表中选定一个职称，单击“删除”按钮删除该职称的所有职工数据。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“批量删除数据”。在页面中添加一个 3 行 1 列的表格用来布局，在第 2 行中添加一个 GridView 控件，在第 3 行中添加一个 DropDownList 控件和一个 Button 控件，所有控件的 ID 均采用默认名称。

设置 GridView1 控件的 AutoGenerateColumns 为 false。单击 GridView1 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单中选择“编辑列”选项，在打开的“字段”对话框中给 GridView1 控件绑定 staff 表中的 5 个字段，并把 5 个字段的列名称都改成中文。给 DropDownList 控件增加教授、副教授、讲师、助教列表项。页面设计视图如图 5.22 所示。

图 5.22　案例 5.9 的设计视图

3）编写程序代码

添加命名空间的引用,代码如下。

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,代码如下。

```
static string constr = "Server = .;Database = test;Integrated Security = true";
SqlConnection conn = new SqlConnection(constr);
```

在页面载入时把 staff 表绑定到 GridView1 控件上显示出来,Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        conn.Open();
        SqlDataAdapter ada = new SqlDataAdapter("select * from staff", conn);
        DataSet ds = new DataSet();
        ada.Fill(ds);
        conn.Close();
        GridView1.DataSource = ds;
        GridView1.DataBind();
    }
}
```

“删除”按钮被单击时执行的事件代码如下。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    conn.Open();
    string sqltext = "delete from staff where Title = '" + this.DropDownList1.SelectedItem.Text
+ "'";
    SqlCommand comm = new SqlCommand(sqltext, conn);
    comm.ExecuteNonQuery();
    SqlDataAdapter ada = new SqlDataAdapter("select * from staff", conn);
    DataSet ds = new DataSet();
    ada.Fill(ds);
    conn.Close();
    GridView1.DataSource = ds;
    GridView1.DataBind();
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面,批量删除之前的运行效果如图 5.23 所示。

选择“助教”选项，单击“删除”按钮后所有职称为“助教”的职工数据被删除了，运行效果如图 5.24 所示。

图 5.23 批量删除之前的效果

图 5.24 批量删除之后的效果

5.4.4 存取图片

1. 将上传图片存储到数据库中

在开发 Web 应用程序的过程中经常会用到图片上传功能，使用户可以将上传的图片名称存储到指定的表中，将图片存储到指定的目录下。

【案例 5.10】 将上传图片存储到数据库中。

首先获取上传控件指定的图片名称，然后获取文件扩展名并判断其格式是否符合图片类型，最后通过 SQL 语句将图片的名称插入数据库 test 的 picSend 表中，并把图片保存到指定文件夹下。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“将上传图片存储到数据库中”。在页面中添加一个 5 行 2 列的表格用来布局，将第 1 行、第 4 行、第 5 行合并，在第 2 行的第 2 列中添加一个 TextBox 控件，在第 3 行的第 2 列中添加一个 FileUpload 控件，在第 4 行中添加一个 Button 控件，在第 5 行中添加一个 GridView 控件，所有控件的 ID 均采用默认名称。

设置 GridView1 控件的 AutoGenerateColumns 为 false。单击 GridView1 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单中选择“编辑列”选项，在打开的“字段”对话框中给 GridView1 控件绑定 picSend 表中的 4 个字段，并把 4 个字段的列名称都改成中文。页面设计视图如图 5.25 所示。

图 5.25 案例 5.10 的设计视图

3）编写程序代码

在前面的案例中，数据库的连接字符串都放在 Web 窗体文件的后台代码中，这种方式不便于多个窗体页面共享数据库连接字符串。当连接字符串发生改变时需要修改所有页面。ASP.NET 提供了 System.Configuration 命名空间，其中的 Configuration 类用于管理网站配置文件，将数据库的连接字符串存放到 Web.config 中可以实现多个窗体共享数据库的连接字

符串。通常将连接字符串放在 Web.config 的<configuration>下的<connectionStrings>配置节中。

本案例中在 Web.config 文件的<configuration>节中加入以下代码。

```
<connectionStrings>
    <add name="constr" connectionString="Server=.;Database=test;Integrated Security=
true"/>
</connectionStrings>
```

添加命名空间的引用,代码如下。

```
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,其中连接 SQL Server 数据库的字符串是从<connectionStrings>节点中读取的,代码如下。

```
SqlConnection conn =
    new SqlConnection(ConfigurationManager.ConnectionStrings["constr"].ConnectionString);
```

定义 gvBind()方法使上传图片表中的内容在 GridView 控件中显示,代码如下。

```
protected void gvBind()
{
    conn.Open();
    string sqltext = "select * from picSend";
    SqlDataAdapter sda = new SqlDataAdapter(sqltext, conn);
    DataSet ds = new DataSet();
    sda.Fill(ds);
    conn.Close();
    GridView1.DataSource = ds.Tables[0].DefaultView;
        GridView1.DataBind();
}
```

页面加载时调用 gvBind()方法显示上传图片表,代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        gvBind();
    }
}
```

单击“上传”按钮后首先获取上传图片文件名,再获取上传文件的扩展名,通过 if 语句判断图片格式是否符合要求,如果符合,则通过 insert 语句将文件名存储到数据库中,并通过 SaveAs 方法将图片保存到指定目录下;如果图片格式不符合要求,则弹出“图片格式不正确!”的提示框。“上传”按钮的单击事件代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
    {
    string Name = TextBox1.Text;
    string pictureName = FileUpload1.FileName;
    string sendTime = DateTime.Now.ToString();
    string lastName = pictureName.Substring(pictureName.LastIndexOf(".") + 1);
```

```
    if (lastName.ToLower() == "jpg" || lastName.ToLower() == "bmp"||lastName.ToLower() == "gif")
    {
        string SavePath = Server.MapPath("images/") + pictureName;
        FileUpload1.PostedFile.SaveAs(SavePath);
        string sql = "insert into picSend(name,picname,sendtime) values('" + Name + "',
'" + pictureName + "','" + sendTime + "')";
        conn.Open();
        SqlCommand com = new SqlCommand(sql, conn);
        com.ExecuteNonQuery();
        conn.Close();
        gvBind();
    }
    else
        Response.Write("< script language = javascript > alert('图片格式不正确!')</script >");
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面，在“用户名：”文本框中输入用户名，通过单击“选择文件”按钮选择要上传的图片文件，单击“上传”按钮。如果文件格式符合要求，则这个上传的图片文件的相关信息就显示在页面下方的 GridView 控件中。页面的运行效果如图 5.26 所示。

图 5.26 将上传图片存储到数据库中的运行效果

2. 读取图片名称并显示图片

在将上传的图片存储到数据库之后，有时需要在数据库中读取图片的名称并显示图片。

【案例 5.11】 读取图片名称并显示图片。

把数据库 test 的 picSend 表中的图片名称绑定到下拉列表中，选择下拉列表中的图片名称，在文本框中显示该图片的名称，并将图片显示在页面中。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“读取图片名称并显示图片”。在页面中添加一个 4 行 2 列的表格用来布局，将第 1 行和第 4 行合并，在第 2 行的第 2 列中添加一个 DropDownList 控件，在第 3 行的第 2 列中添加一个 Label 控件，在第 4 行中添加一个 Image 控件，所有控件的 ID 均采用默认名称。设置 DropDownList 控件的 AutoPostBack 属性为 true。

3）编写程序代码

在 Web.config 文件的< configuration >节中加入以下代码。

```
< connectionStrings >
    < add name = "constr" connectionString = "Server = .;Database = test;Integrated Security =
true"/>
</connectionStrings >
```

添加命名空间的引用，代码如下。

```
using System.Configuration;
```

```
using System.Data;
using System.Data.SqlClient;
```

在所有事件之外定义数据库连接对象,其中连接 SQL Server 数据库的字符串是从<connectionStrings>节中读取的,代码如下。

```
SqlConnection conn =
    new SqlConnection(ConfigurationManager.ConnectionStrings["constr"].ConnectionString);
```

定义 ddlBind()方法把保存在数据表中的图片名称绑定到 DropDownList 控件中,将控件文本内容设置为数据表 picSend 中的 picname 字段,ddlBind()方法的代码如下。

```
protected void ddlBind()
{
    string sql = "select picname from picSend";
    conn.Open();
    SqlDataAdapter sda = new SqlDataAdapter(sql, conn);
    DataSet ds = new DataSet();
    sda.Fill(ds);
    DropDownList1.DataSource = ds.Tables[0].DefaultView;
    DropDownList1.DataTextField = "picname";
    DropDownList1.DataValueField = "picname";
    DropDownList1.DataBind();
}
```

定义 imgBind()方法将当前选择的图片显示在 Image 控件中,代码如下。

```
protected void imgBind()
{
    Image1.ImageUrl = "~/images/" + DropDownList1.SelectedValue;
}
```

在页面加载时,调用 ddlBind()方法将图片名称添加到下拉列表控件 DropDownList 中,调用 imgBind()方法在 Image 控件中显示当前选定的图片,把下拉列表中当前选定的图片名称读到文本框中,Page_Load 事件的代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ddlBind();
        imgBind();
        Label1.Text = this.DropDownList1.SelectedItem.Text;
    }
}
```

当改变 DropDownList 选项时,会激发 SelectedIndexChanged 事件,在该事件中调用 imgBind()方法重新显示图片,并把重新选择的图片名称读取到文本框中。DropDownList 控件的 SelectedIndexChanged 事件的代码如下。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    Label1.Text = this.DropDownList1.SelectedItem.Text;
    imgBind();
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面，在下拉列表中选择图片名称，则在 Image 控件中会显示该图片，并把其名称读取到文本框中，页面的运行效果如图 5.27 所示。

图 5.27 读取图片名称并显示图片

习题 5

1. 填空题

（1）.NET 框架中被用来访问数据库数据的组件集合称为_____。

（2）使用 DataSet 类定义数据集对象必须添加对命名空间_____的引用。

（3）Connection 对象的数据库连接字符串保存在_____属性中。

（4）ExecuteScalar()方法能够执行查询，并返回查询所返回的结果集中第_____行的第_____列。

（5）某 Command 对象 cmd 将被用来执行以下 SQL 语句以向数据源中插入新记录：

```
insert into customs(1000,"tom")
```

则语句“cmd.ExecuteNonQuery();”的返回值可能为_____或_____。

（6）DataTable 是数据集 myDataSet 中的数据表对象，假设有 9 条记录，在调用下列代码后 DataTable 中还有_____条记录。

```
dataTable.Rows[8].Delete();
```

（7）在使用 DataAdapter 对象时只需分别设置表示 SQL 命令和数据库连接的两个参数，就可以通过它的_____方法把查询结果放在一个_____对象中。

（8）如果要创建一个 DataReader 对象，可以通过 Command 对象的_____方法。

（9）用户可以首先使用构造函数创建一个不含参数的 DataAdapter 对象，再设置 DataAdapter 对象的_____属性和_____属性。

（10）每一个 DataSet 都是一个或多个_____对象的集合。

2. 单项选择题

(1) DataAdapter 对象使用与_____属性关联的 Command 对象将 DataSet 修改的数据保存到数据源。

A. SelectCommand　　B. InsertCommand

C. UpdateCommand　　D. DeleteCommand

(2) 在 ADO.NET 中,为了确保 DataAdapter 对象能够正确地将数据从数据源填充到 DataSet 中,必须事先设置好 DataAdapter 对象的_____属性。

A. DeleteCommand　　B. UpdateCommand

C. InsertCommand　　D. SelectCommand

(3) ADO.NET 可以通过设置 Connection 对象的_____属性来指定连接到数据库时的用户和密码信息。

A. ConnectionString　　B. DataSource

C. UserInformation　　D. Provider

(4) 打开与 SQL Server 数据库连接的方法是_____。

A. Close()　　B. Open()　　C. Read()　　D. Fill()

(5) SqlCommand 类可以用来实现_____。

A. 使用 SQL 语句操作数据　　B. 打开数据库的连接

C. 填充数据到内存表　　D. 关闭数据库的连接

(6) SQL Server 的 Windows 身份验证机制是指当网络用户尝试连接到 SQL Server 数据库时_____。

A. Windows 获取用户输入的用户名和密码,提交给 SQL Server 进行身份验证,并决定用户的数据库访问权限

B. SQL Server 将用户输入的用户名和密码提交给 Windows 进行身份验证,并决定用户的数据库访问权限

C. SQL Server 根据已在 Windows 网络中登录的用户的网络安全属性对用户身份进行验证,并决定用户的数据库访问权限

D. 登录到本地 Windows 的用户均可无限制地访问 SQL Server 数据库

(7) 在 ADO.NET 中,为了访问 DataTable 对象从数据源提取的数据行,可以使用 DataTable 对象的_____属性。

A. Rows　　B. Columns　　C. Constraints　　D. DataSet

(8) 下列_____不能在 DataSet 对象 ds 中添加一个名为 Customers 的 DataTable 对象。

A. DataTable dt_customers=new DataTable();

B. DataTable dt_customers=new DataTable("Customers");

C. ds.Tables.Add("Customers");

D. ds.Tables.Add(new DataTable("Customers"));

(9) Command 对象的_____方法在执行数据更新操作时使用。

A. ExecuteNonQuery()　　B. ExecuteReader()

C. ExecuteScalar()　　D. Execute()

(10) _____实现独立于任何数据源的数据访问,可以把它看成内存中的数据库,专门用来处理从数据源中读出的数据。

A. DataGroup　　B. DataSet　　C. ADO　　D. Datanet

3. 上机操作题

(1) 已知 SQL Server 数据库为 XSDB,其中有一张表为 student,表中有 3 个字段,即学号、姓名、性别。设计一个页面,在文本框控件 TextBox 中输入学号查询学生信息,如果查找到,则在标签控件 Label 中显示"查找到的信息如下:",并将该学生的各项信息分别显示在对应的 TextBox 控件中;如果查找不到,则在 Label 中显示"查无此人!"。

(2) 有一个数据库 hh,其中的 student 表中有字段信息(学号 nvarchar(12),姓名 nvarchar(10),性别 nvarchar(2))。现在页面上有 TextBox 控件 TID、TName、TSex (用来分别显示表中的 3 个字段)和一个按钮控件 btnSubmit。请编程实现单击 btnSubmit 控件后将文本框中对应的值插入到 student 数据表中。

(3) 有一个数据库 dd,其中的 ks 表中有字段信息(考号 nvarchar(10),姓名 nvarchar(10),成绩 int)。设计一个页面,在文本框控件 TextBox 中输入要删除的考号,单击"删除"按钮后从数据表 ks 中删除该条记录。

第6章 数据绑定技术

本章学习目标

- 掌握数据绑定的概念和语法；
- 掌握数据源控件的使用；
- 掌握数据显示控件的使用；
- 掌握将数据绑定到控件的方法。

本章介绍了数据绑定的概念及数据绑定语法，对常用的数据源控件和数据显示控件也做了详细讲述，最后通过几个典型案例介绍如何将数据绑定到控件上。

6.1 数据绑定概述

6.1.1 什么是数据绑定

数据绑定是一种自动将数据按照指定格式显示到界面上的技术。数据绑定技术分为简单数据绑定和复杂数据绑定两类。简单数据绑定是将控件的属性绑定到数据源中的某一个值，并且这些值将在页运行时确定。复杂数据绑定是将一组或一列值绑定到指定的控件(数据绑定控件)，如 ListBox、DropDownList、GridView 等。

6.1.2 Eval()方法和 Bind()方法

1. Eval()方法

Eval()方法是一个静态方法，用于定义单向绑定。Eval()方法是只读方法，该方法采用字段名作为参数，以字符串的形式返回该字段的值，仅用于显示。Eval 方法只能读数据，不能更新数据。

2. Bind()方法

Bind()方法也是一个静态方法，用于定义双向绑定。Bind()方法可以把数据绑定到控件，也可以把数据变更提交到数据库，因此 Bind()方法既可以显示数据又可以修改数据。

6.1.3 数据绑定语法

1. 绑定变量

通常，对网页中的各项控件属性进行数据绑定时并不是直接将属性绑定到数据源，而是通

过变量作为数据源来提供数据，然后将变量设置为控件属性。注意，这个变量必须为公有字段或受保护字段，即访问修饰符为 public 或 protected。将数据绑定到变量的语法格式如下。

```
<%#简单变量名%>
```

【案例 6.1】 绑定变量。

把存放在变量中的登录名和登录时间绑定到 Label 控件并在页面上显示出来。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“绑定变量”。在页面中添加一个 3 行 2 列的表格用来布局，将第 1 行合并，在第 2 行的第 2 列和第 3 行的第 2 列中各添加一个 Label 控件，所有控件的 ID 均采用默认名称。

3）编写程序代码

在前台界面中将存放登录名的变量 name 和登录时间的变量 loginTime 分别绑定到 Label1 控件和 Label2 控件，代码如下。

```
<asp:Label ID="Label1" runat="server" Text="<%#name%>"></asp:Label>
<asp:Label ID="Label2" runat="server" Text="<%#loginTime%>"></asp:Label>
```

在后台文件中首先定义两个变量来存放登录名和登录时间，然后在页面载入时调用 Page 对象的 DataBind 方法执行绑定，后台文件的代码如下。

```
public string name="张林";
public DateTime loginTime=DateTime.Now;
protected void Page_Load(object sender, EventArgs e)
{
    Page.DataBind();
}
```

4）运行页面

按 Ctrl＋F5 组合键运行页面，运行效果如图 6.1 所示。

图 6.1　绑定变量

2. 绑定集合

有些服务器控件是多记录控件，例如 DropDownList 控件、ListBox 控件和 GridView 控件等，这类控件可以用一个集合作为数据源。将数据绑定到集合的语法格式如下。

```
<%#简单集合>
```

【案例 6.2】 绑定集合。

将数据集合绑定到 DropDownList 控件并在页面上显示出来。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“绑定变量”。在页面中添加一个 DropDownList 控件，控件的 ID 采用默认名称。

3）编写程序代码

在前台界面中将已定义的集合绑定到 DropDownList 控件上，代码如下。

```
<asp:DropDownList ID = "DropDownList1" runat = "server" DataSource = "<% #ItemList %>">
        </asp:DropDownList>
```

在后台文件中首先定义一个 ArrayList,然后在页面载入时完成数据源的建立,最后调用 DropDownList 控件的 DataBind()方法执行绑定。

ArrayList 是命名空间 System.Collections 下的一部分,在使用该类时必须进行引用。ArrayList 对象的大小是按照其中存储的数据动态扩充与收缩的,所以在声明 ArrayList 对象时并不需要指定它的长度。后台文件的代码如下。

```
protected ArrayList ItemList = new ArrayList();
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ItemList.Add("星期一: Monday");
        ItemList.Add("星期二: Tuesday");
        ItemList.Add("星期三: Wednesday");
        ItemList.Add("星期四: Thursday");
        ItemList.Add("星期五: Friday");
        ItemList.Add("星期六: Saturday");
        ItemList.Add("星期日: Sunday");
        this.DropDownList1.DataBind();
    }
}
```

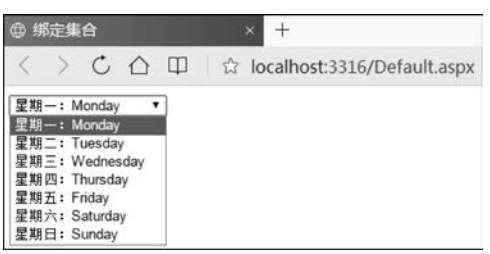

图 6.2 绑定集合

4) 运行页面

按 Ctrl+F5 组合键运行页面,运行效果如图 6.2 所示。

3. 绑定表达式

在将数据绑定到显示控件之前可以利用表达式对数据做一些简单的处理,然后将表达式的执行结果绑定到控件的属性上,通过表达式将执行的结果显示在控件之上。将数据绑定到表达式的语法格式如下。

```
<% #表达式 %>
```

4. 绑定方法

利用表达式只能在数据绑定之前对数据进行简单的处理,如果需要事先对数据进行较复杂的操作计算,可以先利用包含多个表达式的方法,然后将返回值绑定到显示控件的属性。将数据绑定到方法的语法格式如下。

```
<% #方法 %>
```

6.2 数据源控件

6.2.1 数据源控件概述

ASP.NET 内置了多种数据源控件,数据源控件是 ASP.NET 在 ADO.NET 基础上进

一步封装和抽象得到的，可以极大地减轻开发人员的工作，使他们可以不编写任何代码或者编写很少的代码就可以完成页面数据绑定和数据操作功能。

ASP.NET 包含不同类型的数据源控件，通过这些数据源控件可以访问不同类型的数据源，如数据库、XML 文件等。数据源控件没有呈现形式，即在运行时是不可见的。常见的数据源控件有 SqlDataSource、AccessDataSource、ObjectDataSource、XmlDataSource、SiteMapDataSource。

SqlDataSource 提供对使用 SQL 的数据库的访问；ObjectDataSource 允许使用自定义的类访问数据；XmlDataSource 提供对 XML 文档的访问；AccessDataSource 提供对 Access 数据库的访问；SiteMapDataSource 提供给站点导航控件用来访问基于 XML 的站点地图文件。

在实际开发中只需要设定此类控件的 DataSourceID 属性为页面上某一数据源控件的 ID 即可，不需要任何编码就可以实现数据绑定。

6.2.2　SqlDataSource 控件

1. SqlDataSource 概述

SqlDataSource 控件是所有数据源控件中最为常用的。该控件可以从绝大部分数据库中获取数据并进行相关操作，与数据绑定控件相配合可以完成许多操作任务，如分页、排序、选择、编辑等。

SqlDataSource 控件的主要属性如下。

- ConnectionString：连接字符串，其中包括数据库服务器名称、登录用户名、登录密码、数据库名称等。
- ProvideName：SqlDataSource 被设计为支持多种不同类型的数据源，因此必须为每个数据控件设置相应的数据提供程序。ASP.NET 内置了 4 个不同的数据提供程序，它们分别是 ODBC、OleDb、SqlClient 及 OracleClient，默认使用 SqlClient。
- SelectCommand：设置在执行数据记录选择操作时使用的 SQL 语句。
- InsertCommand：设置在执行数据记录添加操作时使用的 SQL 语句。
- UpdateCommand：设置在执行数据记录更新操作时使用的 SQL 语句。
- DeleteCommand：设置在执行数据记录删除操作时使用的 SQL 语句。

2. 为 SqlDataSource 配置数据源

1）选择数据源的种类

打开工具箱，在“数据”控件集中选择 SqlDataSource 控件拖放到页面上。单击控件右上方的智能标记按钮，在弹出的“SqlDataSource 任务”菜单中选择“配置数据源”选项，打开“选择您的数据连接”对话框，如图 6.3 所示。

单击“新建连接”按钮，如果是第一次配置，则会打开“选择数据源”对话框，在其中选择 Microsoft SQL Server，如图 6.4 所示。

2）添加连接

在选择好数据源之后，单击“确定”按钮打开“添加连接”对话框。

在该对话框的“服务器名”文本框中输入运行 SQL Server 实例的服务器名，若要连接运

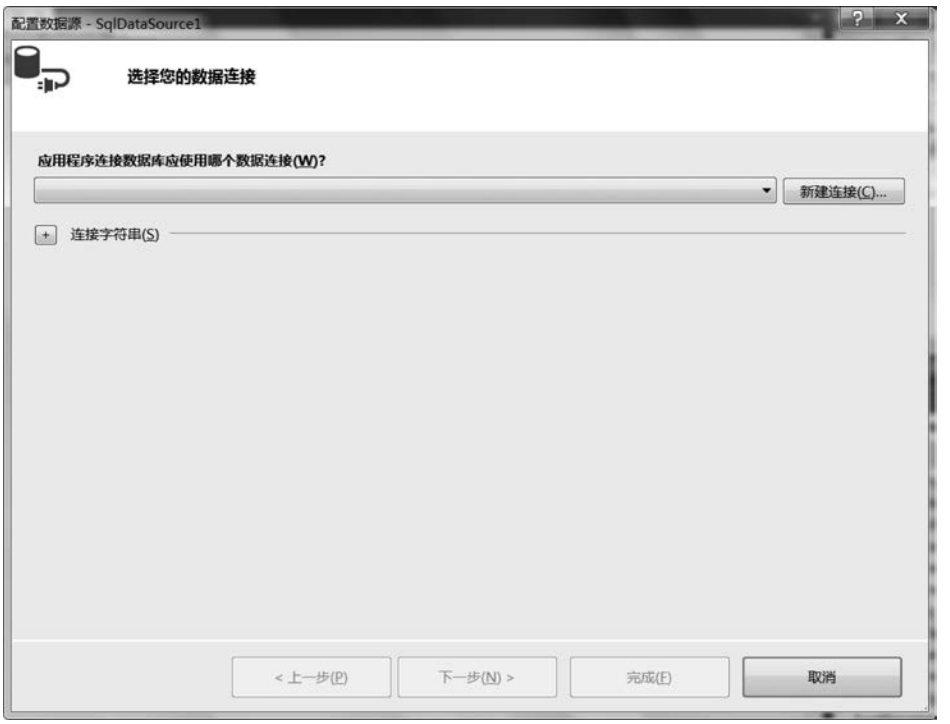

图 6.3 “选择您的数据连接”对话框

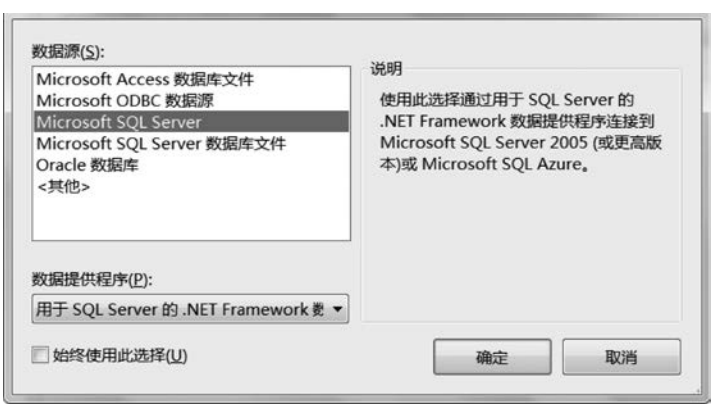

图 6.4 “选择数据源”对话框

行于本机上的 SQL Server 实例,则可以输入.或 local。在“登录到服务器”下选择身份验证方式,若要使用信任连接,可以选中“使用 Windows 身份验证”单选按钮;若要使用 SQL Server 账户进行登录,可以选择“使用 SQL Server 身份验证”单选按钮,然后输入用户名和密码,并选中“保存密码”复选框。在“选择或输入数据库名称”下面的列表中选择 test 数据库,设置好的“添加连接”对话框如图 6.5 所示。

单击“测试连接”按钮,当看到“测试连接成功”信息时单击“确定”按钮,回到“添加连接”对话框后单击“确定”按钮,返回到最初打开的“选择您的数据连接”对话框,在其中新建的连接将以“服务器名.数据库名.dbo”形式出现在“应用程序连接数据库应使用哪个数据连接”框中,如图 6.6 所示。

图 6.5　“添加连接”对话框

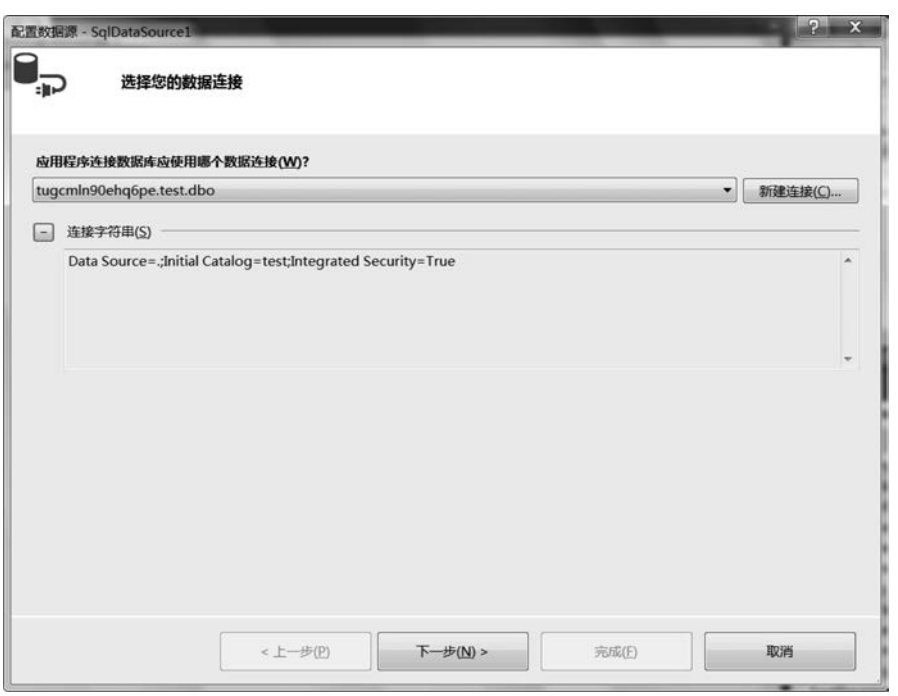

图 6.6　已选择好的数据连接

单击“下一步”按钮,当配置数据源向导提示“是否将连接保存到应用程序配置文件中”时,选中“是,将此连接另存为”复选框,在下面的文本框中会出现一个指定的连接字符串,如图 6.7 所示。

图 6.7　将连接字符串保存到应用程序配置文件中

3) 配置 Select 语句

单击“下一步”按钮,当配置数据源向导提示“希望如何从数据库中检索数据”时,选择“指定来自表或视图的列”,在“名称”下拉列表中选择 staff,在“列”列表框中选取 *(表示选择所有列),如图 6.8 所示。

图 6.8　配置 Select 语句

4）测试查询

单击“下一步”按钮，根据配置数据源向导的提示单击“测试查询”按钮，此时将出现数据源返回的数据，如图 6.9 所示。

图 6.9　测试查询

5）单击“完成”按钮

至此数据源的配置完毕，在配置数据源的过程中生成了数据源控件 SqlDataSource1 的声明标记，代码如下。

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString
 ="<% $ ConnectionStrings:testConnectionString %>" SelectCommand
 ="SELECT * FROM [staff]"></asp:SqlDataSource>
```

6.2.3　AccessDataSource 控件

1. AccessDataSource 概述

Access 数据库是提供基本关系存储的最小数据库，因为使用起来既简单又方便，所以许多小型的 Web 站点都是通过 Access 形成数据存储层。

AccessDataSource 控件使用 System. Data. OleDb 提供程序连接到 Access 数据库。AccessDataSource 控件的独有特征之一是不设置 ConnectionString 属性，用户只需要把 DataFile 属性设置为 Access 数据库文件的位置即可，AccessDataSource 将负责连接到数据库。一般情况下开发人员应将 Access 数据库放在网站的 App_Data 目录中，并通过相对路径（如～/App_Data/lx. mdb）引用这些数据库。

AccessDataSource 不能连接到受密码保护的 Access 数据库，如果要从受密码保护的 Access 数据库中检索数据，需要使用 SqlDataSource 控件。

2. 为 AccessDataSource 配置数据源

1) 选择数据库

打开工具箱,在“数据”控件集中选择 AccessDataSource 控件拖放到页面上。单击控件右上方的智能标记按钮,在弹出的“AccessDataSource 任务”菜单中选择“配置数据源”选项,打开“选择数据库”对话框,如图 6.10 所示。

图 6.10 “选择数据库”对话框

单击“浏览”按钮,打开“选择 Microsoft Access 数据库”对话框,如图 6.11 所示。

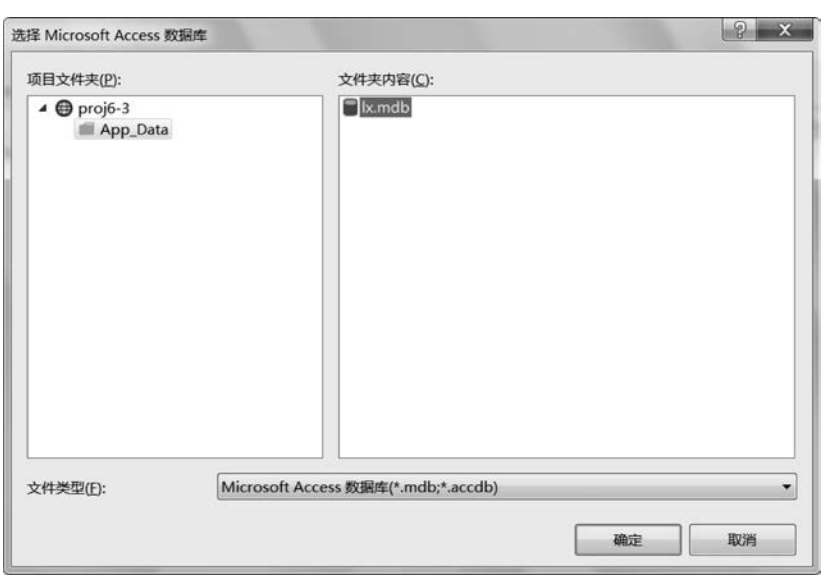

图 6.11 “选择 Microsoft Access 数据库”对话框

在“项目文件夹”的 App_Data 文件夹下选择要连接的 Access 数据文件，单击“确定”按钮，返回到“选择数据库”对话框，如图 6.12 所示。

图 6.12　已选择好的数据库

2）配置 Select 语句

单击“下一步”按钮，打开“配置 Select 语句”对话框，当配置数据源向导提示“希望如何从数据库中检索数据”时选中“指定来自表或视图的列”单选按钮，在“名称”下拉列表中选择 ks，在“列”列表框中选取 *（表示选择所有列），如图 6.13 所示。

图 6.13　“配置 Select 语句”对话框

3）测试查询

单击“下一步”按钮，根据配置数据源向导的提示单击“测试查询”按钮，出现数据源返回的数据，如图 6.14 所示。

图 6.14　测试查询

4）单击“完成”按钮

至此数据源的配置完毕，在配置数据源的过程中生成了数据源控件 AccessDataSource1 的声明标记，代码如下。

```
<asp:AccessDataSource ID="AccessDataSource1" runat="server" DataFile="~/App_Data/lx.mdb"
SelectCommand="SELECT * FROM [ks]"></asp:AccessDataSource>
```

6.2.4　ObjectDataSource 控件

1. ObjectDataSource 概述

ObjectDataSource 控件通过提供一种将数据控件绑定到中间层业务对象的方法为三层结构提供支持。

ObjectDataSource 控件可以用来绑定数据。例如，将 GridView 等绑定到该控件上。使用该控件可以轻松建立多层应用程序，不像使用 SqlDataSource 控件将数据访问逻辑混淆在用户界面中，故可以将用户界面层从业务逻辑和数据访问层中独立出来，从而得到一个三层应用程序结构。

2. 为 ObjectDataSource 配置数据源

1）新建一个类文件

该类文件中设计好了 Object 所需的增、删、改、查的对应方法。例如，类文件 operation.

cs，在其中只设计了查询的方法，代码如下。

```
public class operation
{
    public operation()
    {
    }
    public List < string > GetAllBooks()
    {
        List < string > books = new List < string >();
        books.Add("红楼梦");
        books.Add("三国演义");
        books.Add("水浒传");
        books.Add("西游记");
        return books;
    }
}
```

2）选择业务对象

打开工具箱，在“数据”控件集中选择 ObjectDataSource 控件拖放到页面上。单击控件右上方的智能标记按钮，在弹出的“ObjectDataSource 任务”菜单中选择“配置数据源”选项，打开“选择业务对象”对话框，在下拉列表中选择需要的类文件，如图 6.15 所示。

图 6.15　“选择业务对象”对话框

3）定义数据方法

单击“下一步”按钮，打开“定义数据方法”对话框，选择增、删、改、查的对应选项卡，然后在下拉列表中选择需要的方法，如图 6.16 所示。

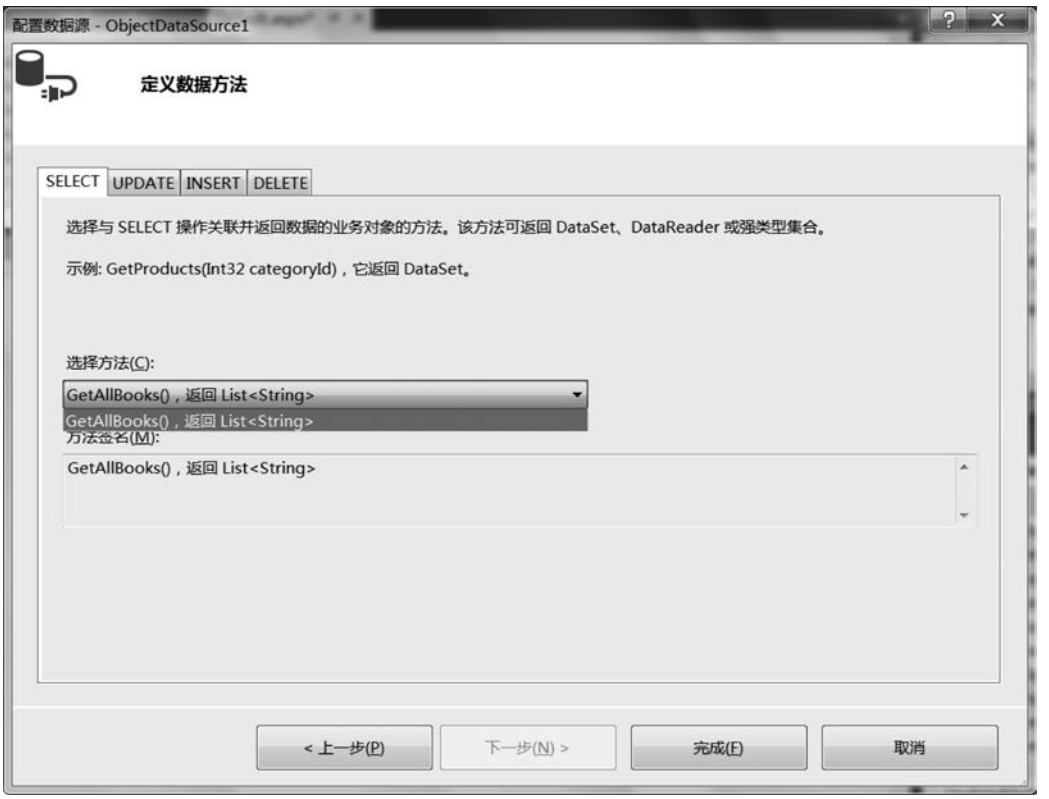

图 6.16 “定义数据方法”对话框

4）单击“完成”按钮

至此数据源的配置完毕，在配置数据源的过程中生成了数据源控件 ObjectDataSource1 的声明标记，代码如下。

```
<asp:ObjectDataSource ID = "ObjectDataSource1" runat = "server" SelectMethod = "GetAllBooks"
TypeName = "operation"></asp:ObjectDataSource>
```

6.2.5 XmlDataSource 控件

1. XmlDataSource 概述

在 Web 开发中有时数据量较小，无须在数据库中通过创建表来维护，这时可以考虑使用 XML 文件保存数据，而 XmlDataSource 可以读取 XML 文件中的数据。

2. 为 XmlDataSource 配置数据源

1）新建 XML 文件

保存到网站根文件夹下的 App_Data 文件夹中。

2）打开“配置数据源”对话框

打开工具箱，在“数据”控件集中选择 XmlDataSource 控件拖放到页面上。单击控件右上方的智能标记按钮，在弹出的“XmlDataSource 任务”菜单中选择“配置数据源”选项，打开“配置数据源”对话框，如图 6.17 所示。

图 6.17　“配置数据源”对话框

3）选择 XML 文件

单击“数据文件”右侧的“浏览”按钮，打开“选择 XML 文件”对话框，在“项目文件夹”的 App_Data 文件夹下选择要连接的 XML 数据文件，如图 6.18 所示。

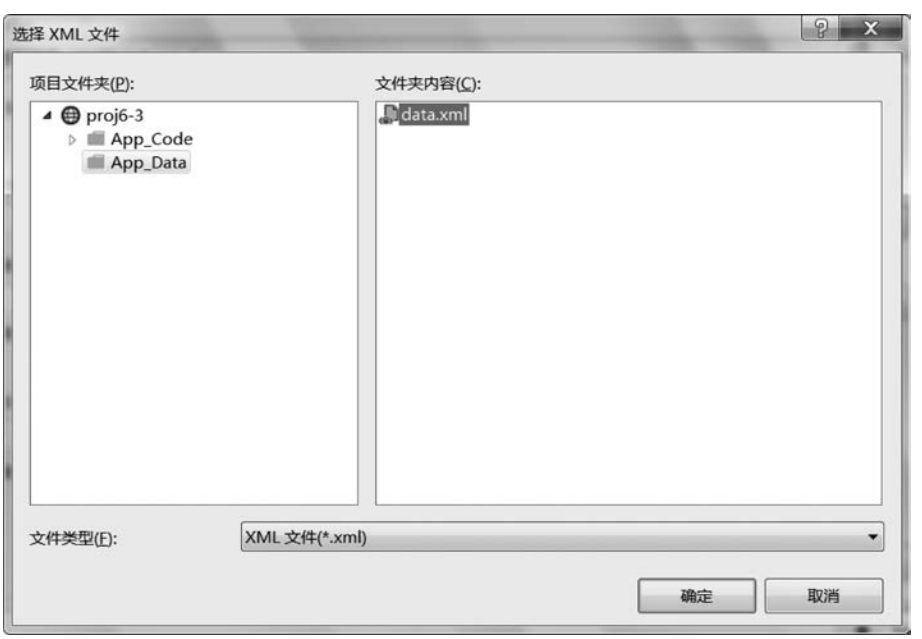

图 6.18　“选择 XML 文件”对话框

单击“确定”按钮，返回到“配置数据源”对话框，如图 6.19 所示。

4）单击“确定”按钮

至此数据源的配置完毕，在配置数据源的过程中生成了数据源控件 XmlDataSource1 的声明标记，代码如下。

图 6.19　已选择好的 XML 文件

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
    DataFile="~/App_Data/data.xml"></asp:XmlDataSource>
```

6.3　数据显示控件

数据显示控件用于在页面上显示数据。ASP.NET 有多种控件可以将数据显示在页面上，包括 GridView、ListView、DataList 和 Repeater 等。

6.3.1　GridView 控件

1. GridView 控件简介

GridView 控件可以用表格形式显示、编辑和删除多种不同数据源中的数据，但不支持数据的插入。GridView 控件的功能非常强大，开发人员可以不用编写任何代码，通过在 Visual Studio 中拖曳，并在属性面板中设置属性即可，还可以完成分页、排序、外观设置等功能。如果需要自定义格式显示各种数据，则可使用 GridView 控件提供的用于编辑格式的模板功能。

2. GridView 控件的常用属性

1）AllowPaging

其指示该控件是否支持分页。

2）AllowSorting

其指示该控件是否支持排序。

3）AutoGenerateColumns

其指示是否自动为数据源中的每个字段创建列，默认为 true。

4）DataSource

其获取或设置包含用来填充该控件的值的数据源对象。

5）DataSourceID

其指示所绑定的数据源控件。

6）Caption

在该控件的标题中显示的文本。

7）HorizontalAlign

其指示该页面上的控件水平对齐。

8）DataKeyNames

其指示要显示在控件中的项的主键字段的名称。

9）DataKeys

包含 DataKeyNames 属性中指定的键字段的值。

10）PageSize

其指示在一个页面上要显示的记录数。

3. GridView 控件的常用事件

1）PageIndexChanging

在分页按钮被单击时发生，在控件处理分页操作之前激发。

2）RowCancelingEdit

当取消修改数据时触发。

3）RowDeleting

在一行的 Delete 按钮被单击时发生，在控件删除该行之前激发。

4）RowEditing

当要编辑数据时触发。

5）RowUpdating

在一行的 Update 按钮被单击时发生，在控件更新该行之前激发。

6）SeletedIndexChanging

在选择新行时触发。

7）Sorting

在对一个列进行排序时触发。

8）RowCreated

在创建行时触发。

4. GridView 控件的常用方法

1）DataBind()

将数据源绑定到 GridView 控件。

2）DeleteRow()

从数据源中删除位于指定索引位置的记录。

3）FindControl()

在当前命名容器中搜索指定的服务器控件。

4）Sort()

根据指定的排序表达式和方向对 GridView 控件进行排序。

5) UpdateRow()

使用行的字段值更新位于指定行索引位置的记录。

5. GridView 控件的绑定列和模板列

在默认情况下,GridView 的 AutoGenerateColumns 属性为 true,可以实现自动创建列,在将 AutoGenerateColunms 属性设置为 false 时可以通过"编辑列"自定义数据绑定列。在 GridView 控件中主要有以下几种类型的绑定列和模板列。

1) BoundField

普通绑定列,用于显示普通文本,是默认的数据绑定列的类型。在显示状态的时候,一个 BoundField 总是直接把数据项作为文本显示。BoundField 的属性 DataField 设置绑定的数据列名称;属性 HeaderText 设置表头的列名称,常用于将原来为英文的字段名转换为中文显示。

2) CheckBoxField

复选框绑定列,可以使用复选框的形式显示布尔类型的数据,注意只有当该控件中有布尔类型的数据时才可以使用 CheckBoxField。

3) HyperLinkField

超链接绑定列,将所绑定的数据以超链接的形式显示出来,其属性 DataTextField 绑定的数据列将显示为超链接的文字;属性 DataNavigateUrlFields 绑定的数据列将作为超链接的 URL 地址。

4) ImageField

图片绑定列,可以在 GridView 控件所呈现的表格中显示图片列,一般情况下它绑定的内容是图片的路径。ImageField 的属性 DataImageUrlField 设置要绑定图片路径的数据列;属性 DataImageUrlFormatString 设置图片列中每个图像的 URL 的格式。

5) CommandField

命令绑定列,使用 CommandField 可以定制 GridView 控件中 Edit、Delete、Update、Cancel 和 Select 等按钮的外观。

6) ButtonField

按钮绑定列,可以通过 CommandName 设置按钮的命令,通常使用自定义的代码实现命令按钮发生的操作。与 CommandField 列不同的是,ButtonField 所定义的按钮与 GridView 没有直接关系,可以自定义相应的操作。

7) TemplateField

自定义模板绑定列,使用 TemplateField 可以在 GridView 控件的数据列中添加任何内容,TemplateField 支持以下几种类型的模板。

- AlternatingItemTemplate:定义交替行的内容和外观。
- EditItemTemplate:定义当前正在编辑的行的内容和外观。
- FooterTemplate:定义该行的页脚的内容和外观。
- HeaderTemplate:定义该行的标题的内容和外观。
- ItemTemplate:定义该行的默认内容和外观。

6. 使用 GridView 进行数据记录的编辑与删除

启用 GridView 控件的编辑和删除功能可以完成对数据表的编辑和删除操作,但是要

求数据表已经设置了主键。

【案例 6.3】 数据记录的编辑与删除。

使用数据源控件与 GridView 控件实现数据记录的编辑与删除，并在删除记录时弹出确认对话框。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“数据记录的编辑与删除”。在页面中添加一个 GridView 控件和一个 SqlDataSource 控件，所有控件的 ID 均采用默认名称。

3）为 SqlDataSource1 配置数据源

单击控件右上方的智能标记按钮，在弹出的“SqlDataSource 任务”菜单中选择“配置数据源”选项，根据向导的提示操作。当配置数据源向导提示“希望如何从数据库中检索数据”时选择“指定来自表或视图的列”，在“名称”下拉列表中选择 staff 表，在“列”列表框中选取 *（表示选择所有列），然后单击对话框右侧的“高级”按钮，打开“高级 SQL 生成选项”对话框，如图 6.20 所示。

图 6.20　“高级 SQL 生成选项”对话框

在对话框中选中“生成 INSERT、UPDATE 和 DELETE 语句”复选框，单击“确定”按钮继续按配置数据源向导的提示操作，数据源配置完成后在 Web.config 中自动增加了 <connectionStrings>节用于存放数据库连接字符串，代码如下。

```
<connectionStrings>
    <add name="testConnectionString" connectionString="Data Source=.;Initial Catalog=
test;Integrated Security=true" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

4）给 GridView 控件设定数据源

单击 GridView 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单的“选择数据源”右侧的下拉列表中选择 SqlDataSource1 作为 GridView1 控件的数据源。

5）启用 GridView 控件的相关功能

单击 GridView 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单中选择“启

用分页""启用编辑""启用删除"选项,并把 GridView 控件的 PageSize 属性设置为 3。

6)为删除设置确认提示

单击 GridView 控件右上方的智能标记按钮,在弹出的"GridView 任务"菜单中选择"编辑列"选项,打开"字段"对话框,如图 6.21 所示。

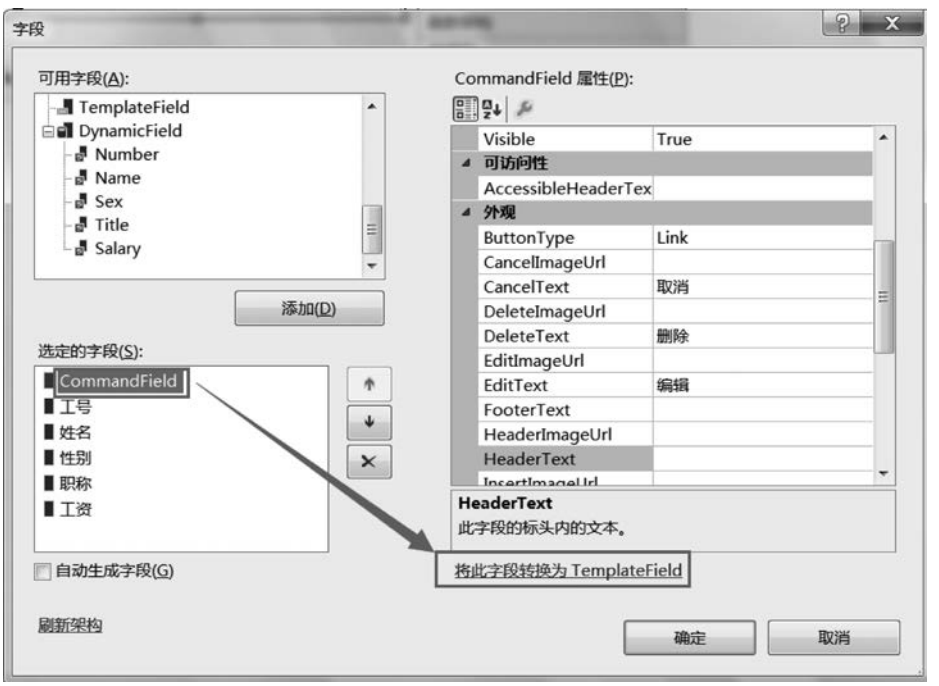

图 6.21 "字段"对话框

在"字段"对话框中把 staff 表的 5 个字段使用 HeaderText 修改成中文。单击"选定的字段"下的 CommandField,然后单击"将此字段转换为 TemplateField"超链接,最后单击"确定"按钮。CommandField 转换前的代码如下。

```
<asp:CommandField ShowDeleteButton = "true" ShowEditButton = "true"/>
```

CommandField 转换为模板列后的代码如下。

```
<asp:TemplateField ShowHeader = "false">
  <EditItemTemplate>
      <asp:LinkButton ID = "LinkButton1" runat = "server" CausesValidation = "true"
CommandName = "Update" Text = "更新"></asp:LinkButton>
      <asp:LinkButton ID = "LinkButton2" runat = "server" CausesValidation = "false"
CommandName = "Cancel" Text = "取消"></asp:LinkButton>
  </EditItemTemplate>
  <ItemTemplate>
      <asp:LinkButton ID = "LinkButton1" runat = "server" CausesValidation = "false"
CommandName = "Edit" Text = "编辑"></asp:LinkButton>
      <asp:LinkButton ID = "LinkButton2" runat = "server" CausesValidation = "false"
CommandName = "Delete" Text = "删除"></asp:LinkButton>
  </ItemTemplate>
</asp:TemplateField>
```

在上面的代码中,在 ItemTemplate 模板中的 LinkButton2 控件里增加实现删除前确认

的客户端事件及其响应的代码如下。

```
OnClientClick = "return confirm('确认删除吗?');"
```

7）运行页面

按 Ctrl＋F5 组合键运行页面，运行效果如图 6.22 所示。

单击“编辑”超链接时可以对选定行的记录修改，单击“删除”超链接会打开如图 6.23 所示的确认对话框，单击“确定”按钮则立即删除选定行的记录，单击“取消”按钮则不删除记录。

图 6.22　数据记录的编辑与删除

图 6.23　确认删除对话框

7. 使用 GridView 选择列与显示主从表

有时一个数据控件中的内容依赖于另一个控件中的内容，当选择了主表中的某条记录后，其相应的信息从另一个从表中显示。

【案例 6.4】 显示主表与从表。

在同一页面中显示主表与从表，查询学生信息及其成绩信息。

1）新建一个空网站

方法略。

2）新建网页 Default.aspx

设置网页标题为“显示主表与从表”。在页面中添加两个 GridView 控件，所有控件的 ID 均采用默认名称，其中 GridView1 用来显示主表，GridView2 用来显示从表。

3）为 GridView1 新建数据源

单击 GridView1 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单的“选择数据源”右侧的下拉列表中选择“<新建数据源>”选项，打开“选择数据源类型”对话框，在“应用程序从哪里获取数据”下单击“数据库”图标，则将在“为数据源指定 ID”下的文本框中自动出现 SqlDataSource1，如图 6.24 所示。

单击“确定”按钮，打开“配置数据源”对话框，可以根据向导提示进行后面的操作。注意，当配置数据源向导提示“希望如何从数据库中检索数据”时选中“指定来自表或视图的列”单选按钮，在“名称”下拉列表中选择“学生”表，在“列”列表框中选取 *（表示选择所有列）。

在 GridView1 控件的数据源配置好后，单击 GridView1 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单中选择“启用选定内容”选项。

4）为 GridView2 新建数据源

单击 GridView2 控件右上方的智能标记按钮，在弹出的“GridView 任务”菜单的“选择数据源”右侧的下拉列表中选择“<新建数据源>”选项，打开“选择数据源类型”对话框，在“应用程序从哪里获取数据”下单击“数据库”图标，则将在“为数据源指定 ID”下的文本框中自动出现 SqlDataSource2，单击“确定”按钮，打开“配置数据源”对话框，可以根据向导提示进行后面的操作。