

第 8 章 数据绑定

目 录

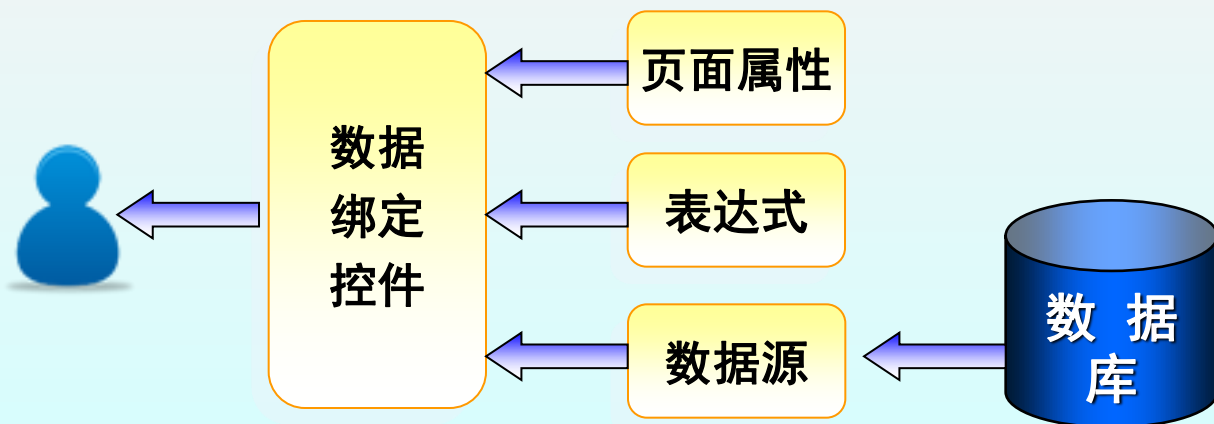
-  1 数据绑定基础.....
-  2 数据源控件.....
-  3 数据绑定控件.....
-  4 实例.....

8.1 数据绑定基础

➤ 什么是数据绑定

就是把数据和控件相关联，由控件负责自动显示数据的一种方式。通常都是通过声明的方式将数据和控件关联起来，实现自动展示。

ASP.NET中的大部分控件都支持**单值数据绑定**，还有一些控件支持**重复值绑定**，也就是说以列表或表格的方式呈现出一组项目，可以绑定到一个数据集自动、重复的获取集合中的每一项并呈现在页面上。



8.1.1 数据绑定表达式

➤ 数据绑定表达式格式

```
<%# data_bind_expression %>
```

说明：

- 可以输出页面的属性值，成员变量值或函数的返回值
- 这些属性、变量及函数具有protected或public可见性



示例

使用数据绑定表达式:

```
<%# EmployeeName %> // 绑定页面属性  
<%# getUsername( ) %> // 绑定页面中的方法  
<%# “Tom” + “Cat” %> // 绑定字符串表达式  
<%# DateTime.Now %> // 绑定当前时间  
<%# Request.Url %> // 绑定当前页面的URL
```

8.1.2 单值绑定

➤ 步骤

将数据绑定表达式
赋值给控件的某个属性



在页面或控件上调用
DataBind()方法

```
<asp:Label ID="lblUser" runat="server" Text="<%=# currentUser %>" />
```

```
lblUser.DataBind();
```



示例

使用数据绑定表达式实现单值绑定：页面代码

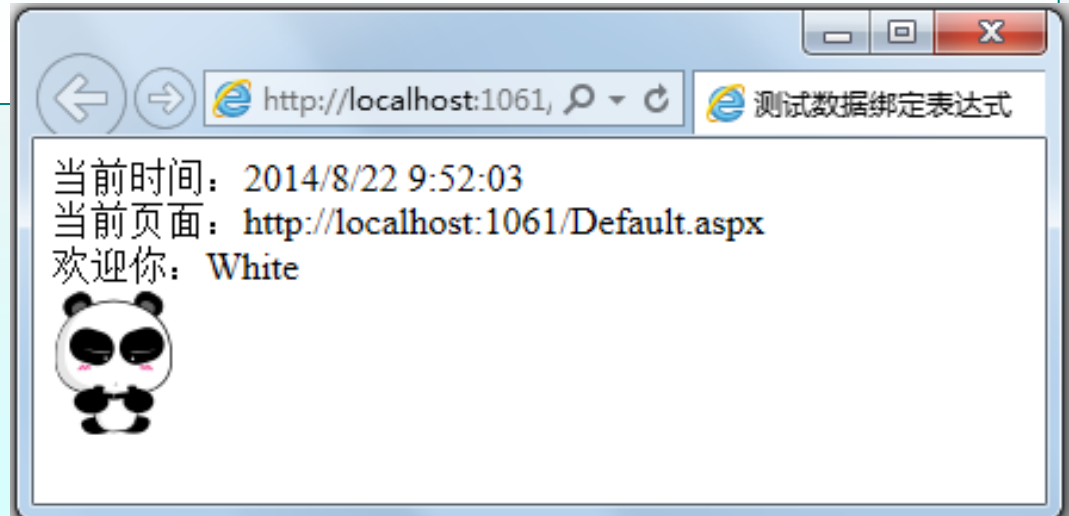
```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>测试数据绑定表达式</title></head>
<body>
<form id="form1" runat="server">
    当前时间： <%# DateTime.Now %><br />
    当前页面： <%# Request.Url %> <br />
    欢迎你：
    <asp:Label ID="lblUser" runat="server" Text="<%# CurrentUser %>" />
    <asp:Image ID="imgUser" runat="server" ImageUrl="<%# getImg() %>" />
</form>
</body>
</html>
```




示例

使用数据绑定表达式实现单值绑定： 后台代码

```
protected string CurrentUser {  
    get { return "White"; }  
}  
  
protected string getImg() {  
    return "./img/user.png";  
}  
  
protected void Page_Load(object sender, EventArgs e) {  
    this.DataBind();  
}
```



8.1.3 重复值绑定

➤ 概述

- 重复值绑定允许将一个列表的信息绑定到一个控件上
- 列表可以是自定义对象的集合，也可以是行的集合

➤ 常用列表控件

- DropDownList
- ListBox
- CheckBoxList
- RadioButtonList
- BulletedList

➤ 列表控件的基本属性

属性名	属性描述
DataSource	数据源对象，包含要显示的数据，该对象通常实现ICollection接口
DataSourceID	数据源对象的ID，通过该属性可以链接列表控件和数据源控件。该属性与DataSource只能设置一个，不能同时使用。
DataTextField	数据源中可以包含多个数据项（列），但列表控件中只能显示单个列的值，DataTextField属性指定包含要显示在页面上字段的名称（绑定到行集时）或属性名称（绑定到对象集时）
DataValueField	该属性和DataTextField属性类似，但从数据项中获得的数据不会显示在页面上，而是保存在底层HTML标签的value属性上，允许以后在代码中读取该属性值。该属性通常用于保存唯一值或主键。
DataTextFormatString	定义一个可选的字符串，格式化DataTextField的值

➤ 数据绑定方法

在代码中绑定

- 1) 设置列表控件的**DataSource**属性为集合对象
- 2) 显式的调用列表控件的**DataBind()** 方法

声明方式绑定

- 1) 设置列表控件的**DataSourceID**属性为集合对象
- 2) 不需调用**DataBind()** 方法，系统自动执行绑定

示例 建立NorthWind数据库的按类别查询产品信息的页面

```
protected void Page_Load(object sender, EventArgs e){  
    if (!Page.IsPostBack) {  
        using (SqlConnection conn = new SqlConnection(connstr)) {  
            SqlCommand cmd = conn.CreateCommand();  
            cmd.CommandText =  
                "Select CategoryID, CategoryName from Categories";  
            conn.Open();  
            SqlDataReader reader = cmd.ExecuteReader();  
            ddlcategory.DataSource = reader;  
            ddlcategory.DataTextField = "CategoryName";  
            ddlcategory.DataValueField = "CategoryID";  
            ddlcategory.DataBind();  
        }  
        bindproducts();  
    }  
}
```



```
private void bindproducts() {  
    string catid = ddlcategory.SelectedValue;  
    using (SqlConnection conn = new SqlConnection(connstr)) {  
        SqlCommand cmd = conn.CreateCommand();  
        cmd.CommandText =  
            "Select ProductName from Products where CategoryID= @catid";  
        cmd.Parameters.AddWithValue ("@catid", catid);  
        conn.Open();  
        SqlDataReader reader = cmd.ExecuteReader();  
        Sqlproduct.DataSource = reader;  
        DataReader reader = cmd.ExecuteReader();  
        bll bllproduct.DataTextField = "productname";  
        bllproduct.DataBind();  
    }  
}
```

8.2 数据源控件

8.2.1 数据源控件概述

➤ 为什么要使用数据源控件

- 使用数据源控件，只需配置即可完成数据访问操作
- 配合功能强大的数据绑定控件，不用编写一行代码，就可以开发出功能强大的数据库应用

示例 使用数据源控件和数据绑定控件，开发员工管理程序

- 创建员工管理程序的Web页面
- 添加数据绑定控件GridView到Web页面
- 为GridView控件配置数据源控件
 - ⊕ 选择数据源类型
 - ⊕ 配置数据库连接
 - ⊕ 配置数据检索命令
 - ⊕ 配置数据增删改命令（可选）
- 配置数据绑定控件GridView
- 运行并测试Web应用程序

➤ .NET框架中的常用数据源控件

控件名	作用
SqlDataSource	代表使用ADO.NET提供程序连接的关系数据库中得数据，支持使用SqlClient、OleDb、Odbc 或 OracleClient连接到的任何关系数据库。
ObjectDataSource	代表多层体系结构中得中间层对象。较复杂的应用程序通常将表示层同业务层分开，并在业务对象中封装处理逻辑，ObjectDataSource使开发人员能够在n层体系结构的应用程序中使用数据源控件。
AccessDataSource	代表使用Microsoft Access数据库的数据源控件，这是一种文件数据源
XmlDataSource	是向数据绑定控件提供XML数据的数据源控件，通过它可以获取分层数据或表格数据，通常用于显示只读方案中的分层XML数据
SiteMapDataSource	是站点地图数据的数据源，利用它可以使TreeView、Menu等控件绑定到分层的站点地图数据。

8.2.2 使用SqlDataSource控件

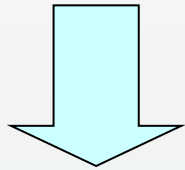
➤ SqlDataSource控件概述

- 可以连接到任何拥有ADO.NET数据提供程序的数据源，包括SQL Server、Oracle以及基于OLE DB或ODBC的数据源
- 会根据配置自动创建Connection对象、Command对象及DataReader对象等，以完成各项数据访问操作
- 需要配置一系列参数，主要包括：
 - ⊕ 数据库连接字符串
 - ⊕ 数据增删改查命令
 - ⊕ 各种命令参数

➤ 配置连接字符串

在web.config文件中配置连接串

```
<connectionStrings>  
  <add name="connstr" connectionString="Data Source=.\sqlexpress;  
    Initial Catalog=northwind; Integrated Security=True"  
    providerName="System.Data.SqlClient"/>  
</connectionStrings>
```



在SqlDataSource标记中引用连接串

```
<asp:SqlDataSource ConnectionString =  
  "<%"$ ConnectionStrings:connstr %">" ..... />
```

➤ 配置查询命令

在数据源控件中配置查询命令

```
<asp:SqlDataSource ID="dsEmployees" runat="server"  
    ConnectionString="<%%$ ConnectionStrings:connstr %>"  
    SelectCommand =  
        "SELECT EmployeeID, LastName, FirstName FROM Employees"  
</asp:SqlDataSource>
```



建立根据居住地查询员工基本信息的程序

步骤1：定义主表数据源

```
<asp:SqlDataSource ID="dsCity" runat="server"  
    ConnectionString="<%%$ ConnectionStrings:connstr %>"  
    SelectCommand="select distinct city from employees">  
</asp:SqlDataSource>
```

步骤2：定义下拉列表框以选择城市

```
<asp:DropDownList ID="ddlCity" runat="server" AutoPostBack="True"  
    DataSourceID="dsCity" DataTextField="city"  
    DataValueField="city">  
</asp:DropDownList>
```

步骤3: 定义子表数据源

<asp
Co
Se

<S

</S

</asp

步骤

<asp

</asp

根据居住地查询员工信息 - Windows Internet Explorer

http://localhost

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

★ 收藏夹 根据居住地查询员工信息

London

EmployeeID	LastName	FirstName	Title	City
5	Buchanan	Steven	Sales Manager	London
6	Suyama	Michael	Sales Representative	London
7	King	Robert	Sales Representative	London
9	Jerry	Mouse	Sales Representative	London

完成 本地 Intranet 105%



示例 建立可更新员工信息的应用程序

使用数据源更新数据 - Windows Internet Explorer

http://localhost:1245/dsUpdate.aspx

文件(E) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

★ 收藏夹 使用数据源更新数据

	EmployeeID	LastName	FirstName	Title	City
编辑	1	Davolio	Nancy	Sales Representative	Seattle
更新 取消	2	Fuller	Andrew	Vice President, Sales	Tacoma
编辑	3	Leverling	Janet	Sales Representative	Kirkland
编辑	4	Peacock	Margaret	Sales Representative	Redmond
编辑	5	Buchanan	Steven	Sales Manager	London
编辑	6	Suyama	Michael	Sales Representative	London
编辑	7	King	Robert	Sales Representative	London
编辑	8	Callahan	Laura	Inside Sales Coordinator	Seattle
编辑	9	Jerry	Mouse	Sales Representative	London
编辑	13	Cat	Tom		
编辑	14	Mouse	Jerry		

本地 Intranet 105%

8.2.3 使用ObjectDataSource控件

➤ 为什么要使用ObjectDataSource控件

- 大型Web应用通常采用N层体系结构
- 表示层（页面）将调用业务层或数据访问层的方法
- 各层之间通常使用业务对象（值对象）来传递数据
- 为实现这种灵活性，可以使用ObjectDataSource控件



示例

建立分层结构的员工信息查询程序

步骤1：定义代表员工信息的业务对象类

```
public class Employee {  
    private int empid;  
    public int Empid {  
        get { return empid; }        set { empid = value; }  
    }  
    private string firstname;  
    public string Firstname {  
        get { return firstname; }    set { firstname = value; }  
    }  
    private string lastname;  
    public string Lastname {  
        get { return lastname; }    set { lastname = value; }  
    }  
    private string city;  
    public string City {  
        get { return city; }        set { city = value; }  
    }  
}
```

步骤2：定义数据访问层对象类（代码框架）

```
public class EmployeeDB {  
    // 从配置文件中读取数据库连接字符串  
    string connstr =  
        ConfigurationManager.ConnectionStrings ["connstr"].ConnectionString;  
    // 获取所有员工居住的城市  
    public List<String> getcitys() { ..... }  
    // 获取居住在某城市的所有员工的信息  
    public List<Employee> getemployeesbycity(string city) { ..... }  
    // 根据员工号获取某员工的基本信息  
    public Employee getemployee(int empid) { ..... }  
    // 向数据库中插入一个员工的信息  
    public int insertemployee(Employee emp) { ..... }  
    // 从数据库中删除一条员工的信息  
    public int deleteemployee(int empid) { ..... }  
    // 将员工信息的更改保存到数据库中  
    public int updateemployee  
        (int employeeID, string firstName, string lastName, string city) { ..... }  
}
```

步骤3: 建立Web页面, 添加DropDownList及GridView控件

步骤4: 配置dsCity数据源

```
<asp:ObjectDataSource ID="dsCity" runat="server" SelectMethod="getcitys"  
    TypeName="EmployeeDB" >  
</asp:ObjectDataSource>
```

步骤5: 配置dsEmployee数据源

```
<asp:ObjectDataSource ID="dsEmployee" runat="server"  
    SelectMethod="getemployeesbycity" TypeName="EmployeeDB" >  
    <SelectParameters>  
        <asp:ControlParameter ControlID="ddlCity" Name="city"  
            PropertyName="SelectedValue" Type="String" />  
    </SelectParameters>  
</asp:ObjectDataSource>
```

步驟6：配置DropDownList及GridView控件

```
<asp:DropDownList ID="ddlCity" runat="server" DataSourceID="dsCity"
    AutoPostBack="True">
</asp:DropDownList>
<asp:GridView ID="gvEmployees" runat="server" AutoGenerateColumns="False"
    DataSourceID="dsEmployee">
    <Columns>
        <asp:CommandField ShowEditButton="True" />
        <asp:BoundField DataField="Employeeid" HeaderText="Employeeid"
            SortExpression="Employeeid" />
        <asp:BoundField DataField="Firstname" HeaderText="Firstname"
            SortExpression="Firstname" />
        <asp:BoundField DataField="Lastname" HeaderText="Lastname"
            SortExpression="Lastname" />
        <asp:BoundField DataField="City" HeaderText="City"
            SortExpression="City" />
    </Columns>
</asp:GridView>
```



为员工管理应用增加数据更新的功能

步骤1: 在业务类中定义更新员工信息的方法

```
public void updateemployee
    (int employeeid, string firstname, string lastname, string city)
{
    using (SqlConnection conn = new SqlConnection(connstr)) {
        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText="UPDATE Employees SET LastName=@last,
            FirstName= @first, City = @city WHERE EmployeeID = @id";
        cmd.Parameters.AddWithValue (" @id", employeeid);
        cmd.Parameters.AddWithValue (" @last", lastname);
        cmd.Parameters.AddWithValue (" @first", firstname);
        cmd.Parameters.AddWithValue (" @city", city);
        conn.Open();
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }
}
```

步骤2：为数据源配置Update方法

```
<asp:ObjectDataSource ID="dsEmployee" runat="server"
    SelectMethod="getemployeesbycity" TypeName="EmployeeDB"
    UpdateMethod="updateemployee" >
    <UpdateParameters>
        <asp:Parameter Name="employeeid" Type="Int32" />
        <asp:Parameter Name="firstname" Type="String" />
        <asp:Parameter Name="lastname" Type="String" />
        <asp:Parameter Name="city" Type="String" />
    </UpdateParameters>
    <SelectParameters>
        <asp:ControlParameter ControlID="ddlCity" Name="city"
            PropertyName="SelectedValue" Type="String" />
    </SelectParameters>
</asp:ObjectDataSource>
```

步骤3：为GridView控件配置编辑功能

8.3 数据绑定控件

➤ 常用数据绑定控件

- **GridView**
- **ListView**
- **DetailsView**
- **FormView**

8.3.1 GridView控件

➤ 为GridView定义列

- **BoundField**: 显示数据源中指定字段的文本
- **CheckBoxField**: 对布尔字段创建一个复选框显示其状态
- **ImageField**: 显示二进制字段中的图像数据
- **ButtonField**: 为列表中的每个项目创建一个按钮，用于捕获事件并编写处理代码
- **CommandField**: 为列表中的每个项目提供选择、编辑等常用功能的按钮
- **HyperLinkField**: 为列表中的每个项目创建一个超链接
- **TemplateField**: 以自定义模板来显示数据或创建控件，为开发提供最大的灵活性

➤ **BoundField**列的属性及用法

属性名	描述
DataField	本列中要显示的数据项的字段名或属性名
DataFormatString	格式化字符串，用于控制本列中数据的显示格式
HeaderText	设置标题行要显示的文本
HeaderImageUrl	设置标题行要显示的图像
FooterText	设置脚注行要显示的文本
SortExpress	排序表达式，用于执行基于该列的排序
ReadOnly	当记录处于编辑模式时，该列是否允许修改
Vissible	该列是否显示在页面上，为假时不显示

示例：DataFormatString属性的用法

```
<asp:BoundField DataField="UnitPrice" HeaderText="Price"
DataFormatString="{0: C}" />
```

占位符 格式指示器

常用格式化字符串

数据类型	格式化串	作用	示例
数值型	{0: C}	货币格式表示	\$1,234.50，其中货币符号与地区相关
	{0: E}	科学计数法表示	1.23450E+004
	{0: P}	百分比表示	45.6%
	{0: F?}	固定小数位数	对123.4，采用{0: F3}格式化为123.400，而采用{0: F0}则格式化为123
日期型	{0: d}	使用短日期格式	具体格式取决于区域设置中的短日期格式
	{0: D}	使用长日期格式	具体格式取决于区域设置中的长日期格式
	{0: s}	ISO标准格式	yyyy-MM-ddTHH:mm:ss， 例如2011-07-20T10:00:23
	{0: M}	月日格式	MMMM dd，例如January 20
	{0: G}	一般格式	依赖于区域设置，例如10/30/2011 10:00:23 AM

➤ HyperLinkField列的用法

应用场景：概要页面到详情页面的导航

HyperLinkField示例 - Mozilla Firefox

文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

HyperLinkField示例 +

EmployeeID		City
1	Davolio Nancy	Seattle
2	Fuller And	Tacoma
3	Leverling Janet	Kirkland
4	Peacock Margaret	Redmond
5	Buchanan Steven	London
6	Suyama Michael	London
7	King Robert	London
8	Callahan Laura	Seattle
9	Jerry Mouse	London

http://localhost...Detail.aspx?id=1

员工详细信息 - Mozilla Firefox

文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

员工详细信息 +

EmployeeID: 1
LastName: Davolio
FirstName: Nancy
Title: Sales Representative
BirthDate: 1948-12-8 0:00:00
Address: 507 - 20th Ave. E. Apt. 2A
City: Seattle
Country: USA
HomePhone: (206) 555-9857



员工信息列表到详情页面导航

员工信息浏览页面：在GridView上添加超链接列，并配置如下属性

- ✓ DataTextField：设置该列要显示的数据源中的字段
- ✓ DataNavigateUrlFormatString：设置超链接的URL格式
- ✓ DataNavigateUrlFields：设置导航中要传递的参数列表

员工信息详情页面：配置数据源如下

```
<asp:SqlDataSource ID="dsEmployee" runat="server"
    ConnectionString="<%%$ ConnectionStrings:nwconnstr %>"
    SelectCommand="SELECT EmployeeID, LastName, FirstName, Title, BirthDate,
        Address,City, Country, HomePhone FROM Employees
        Where EmployeeID=@ID">
    <SelectParameters>
        <asp:QueryStringParameter Name="ID" QueryStringField="id" />
    </SelectParameters>
</asp:SqlDataSource>
```

➤ 对GridView排序

- GridView控件提供了**内置排序功能**，无需任何编码
- 为启用排序，需设置GridView的**AllowSorting**属性为真，并且为每个可排序的列定义**排序表达式**。
- 排序表达式包含一个或一系列用逗号分隔的字段名，每个字段名后还可以加上ASC或DESC以限定升/降序。
- 一般情况下，真正实现排序逻辑的是数据源控件而不是GridView控件，GridView控件只是展示数据，并利用数据源控件的排序功能。



示例 为员工查询页面增加排序功能

```
<asp:GridView ID="gvEmployee" runat="server" AllowSorting="True"
    AutoGenerateColumns="False" DataKeyNames="EmployeeID"
    DataSourceID="dsEmployee" >
    <Columns>
        <asp:BoundField DataField="EmployeeID" HeaderText="EmployeeID"
            InsertVisible="False" ReadOnly="True" SortExpression="EmployeeID" />
        <asp:BoundField DataField="FirstName" HeaderText="FirstName"
            SortExpression="FirstName" />
        <asp:BoundField DataField="LastName" HeaderText="LastName"
            SortExpression="LastName" />
        <asp:BoundField DataField="Title" HeaderText="Title" />
    </Columns>
</asp:GridView>

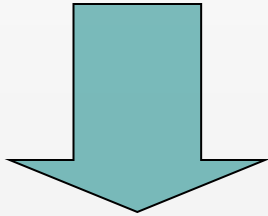
<asp:SqlDataSource ID="dsEmployee" runat="server"
    ConnectionString="<%= $ConnectionStrings.nwconnstr %>"
    SelectCommand="Select EmployeeID, FirstName, LastName, Title
        From Employees">
</asp:SqlDataSource>
```


➤ 对GridView显示的数据分页

- 当GridView中要呈现的记录数量较多时，就要启用分页。
 -
- GridView对分页提供内建支持，可以和数据源控件配合实现简单的分页控制，也可以实现自定义分页控制。
- 分页相关的属性和方法：
 - ✓ **AllowPaging**：是否启用分页，默认为False
 - ✓ **PageSize**：获取或设置每页中显示的记录数，默认为10
 - ✓ **PageIndex**：获取或设置当前显示页的页面号（从0开始编号）
 - ✓ **PagerSettings**：一组分页控件的设置项，决定了分页控件出现的位置及它们包含的文本、图片等
 - ✓ **PageIndexChanging**事件：当点击了分页按钮时将触发该事件，可以捕获它以定制分页代码

自动分页的缺陷：无法减少数据库查询的数据量

- ✓ 每次当改变页码时，都需要获取所有满足条件的记录，然后绑定指定页的数据，并将其余数据丢弃。这样会造成数据库的沉重负荷以及大量的冗余数据传输。
- ✓ 使用数据源控件时，缓存机制会大幅度提高自动分页的性能，减少连接数据库的次数。但若返回成千上万条记录，也会消耗大量的内存。



解决方案：自定义分页

- ✓ 需要编程获取指定页的数据，并绑定到GridView。
- ✓ 既可以使用ADO.NET访问数据库，也可以定制ObjectDataSource访问数据库。



列表显示员工的基本信息，每页显示5条记录

步骤1：开发数据访问类

// 计算数据表中的记录总条数

```
public int countemployees() {  
    int cnt = 0;  
    using (SqlConnection conn = new SqlConnection(connstr)) {  
        SqlCommand cmd = conn.CreateCommand();  
        cmd.CommandText = "Select Count(EmployeeID) From Employees ";  
        conn.Open();  
        cnt = (int) cmd.ExecuteScalar();  
        cmd.Dispose();  
    }  
    return cnt;  
}
```

续：开发数据访问类

// 从数据库中查询指定页的记录，start为起始记录号，count为每页记录数

```
public List<Employee> getemployees( int start, int count ){  
    List<Employee> list = new List<Employee>();  
    using (SqlConnection conn = new SqlConnection(connstr)) {  
        SqlCommand cmd = conn.CreateCommand();  
        cmd.CommandText = "SELECT EmployeeID, LastName, FirstName, Title "  
            + "FROM (SELECT EmployeeID, LastName, FirstName, Title, "  
            + "ROW_NUMBER() OVER(ORDER BY EmployeeID) as RowNum "  
            + " FROM Employees ) as emps Where RowNum Between @start and @end";  
        cmd.Parameters.AddWithValue ( "@start", start);  
        cmd.Parameters.AddWithValue ( "@end", start + count);  
        conn.Open();  
        SqlDataReader dr = cmd.ExecuteReader();  
  
        .....  
    }  
    return list;  
}
```

步骤2：声明ObjectDataSource数据源

```
<asp:ObjectDataSource ID="dsEmployee" runat="server"  
    EnablePaging="True"  
    TypeName="EmployeeDB"  
    SelectCountMethod="countemployees"  
    SelectMethod="getemployees"  
    MaximumRowsParameterName="count"  
    StartRowIndexParameterName="start">  
</asp:ObjectDataSource>
```

步骤3：在GridView中启用自定义分页

```
<asp:GridView ID="gvEmployee" runat="server"  
    DataSourceID="dsEmployee" AllowPaging="True" PageSize="5"  
    ..... >  
</asp:GridView>
```

➤ 在GridView中处理行数据

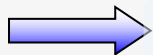
创建**命令列**或**按钮列**
以触发行事件



捕获特定事件
并编码进行处理

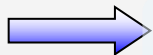
➤ 常用命令及触发事件

Delete命令



RowDeleting、RowDeleted事件

Edit命令



RowEditing事件

Update命令



RowUpdating、RowUpdated事件

Cancel命令



RowCancelingEdit事件

Select命令



SelectedIndexChanging、
SelectedIndexChanged事件



创建根据员工查询其订单的程序

设计思路：

- 创建两个**GridView**，一个显示员工列表，另一个显示订单列表。
- 在第一个网格上捕获行选择事件，然后根据所选行的**EmployeeID**，在**Order**表中查询订单信息并绑定到第二个网格上显示

步骤1：创建页面，并添加检索员工信息的数据源控件

```
<asp:SqlDataSource ID="dsEmployee" runat="server"
    ConnectionString="<%%$ ConnectionStrings:nwconnstr %>"
    SelectCommand="SELECT EmployeeID, LastName, FirstName, Title
                  FROM Employees" >
</asp:SqlDataSource>
```

步骤2：添加显示员工信息的网格控件**gvEmployee**

步骤3：添加显示员工订单列表的网格控件**gvOrder**

步骤4. 向`gvEmployee`中添加一个`CommandField`列 并设置其`CommandName`属性

步骤

pr

}

在GridView中选择行并显示子表数据 - Windows Internet Explorer

http://localhost:2717/databindctls/Sel

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

★ 收藏夹 在GridView中选择行并显示子表数据

查询订单	EmployeeID	LastName	FirstName	Title
选择	1	Davolio	Nancy	Sales Representative
选择	2	Fuller	And	Vice President, Sales
选择	3	Leverling	Janet	Sales Representative
选择	4	Peacock	Margaret	Sales Representative
选择	5	Buchanan	Steven	Sales Manager
选择	6	Suyama	Michael	Sales Representative
选择	7	King	Robert	Sales Representative
选择	8	Callahan	Laura	Inside Sales Coordinator
选择	9	Jerry	Mouse	Sales Representative

OrderID	CustomerID	OrderDate
10258	ERNSH	1996-7-17 0:00:00
10270	WARTH	1996-8-1 0:00:00
10275	MAGAA	1996-8-7 0:00:00
10285	QUICK	1996-8-20 0:00:00
10292	TRADH	1996-8-28 0:00:00

本地 Intranet 105%

说明：向**SelectedIndexChanged**事件中如何获取该行的某列数据。

- 方式1：若该列设置为主键，则通过**SelectedDataKey** 属性获取

```
int id = (int)gvEmployee.SelectedDataKey.Value;
```

```
int id = (int)gvEmployee.SelectedDataKey.Values[0];
```

- 方式2：若该列不是主键列，则从当前行的指定单元格中获取

```
string eid = gvEmployee.SelectedRow.Cells[1].Text;
```



示例

手工编码，实现员工信息的各项编辑操作

设计思路：

- 捕获RowEditing、RowUpdating、RowCancelingEdit事件。
- 通过设置EditIndex属性并重新绑定数据源可使某行进入编辑模式

The screenshot shows a Windows Internet Explorer browser window titled "更改员工信息". The address bar shows the URL "http://localhost:2717/da". The browser displays a table with employee information. The table has columns: 操作 (Action), ID, First Name, Last Name, and Title. The first row is in edit mode, with the "操作" column containing "更新" (Update) and "取消" (Cancel) buttons. The other rows have a single "编辑" (Edit) button in the "操作" column.

操作	ID	First Name	Last Name	Title
更新 取消	1	Nancy	Davolio	Sales Representative
编辑	2	And	Fuller	Vice President, Sales
编辑	3	Janet	Leverling	Sales Representative
编辑	4	Margaret	Peacock	Sales Representative
编辑	5	Steven	Buchanan	Sales Manager
编辑	6	Michael	Suyama	Sales Representative
编辑	7	Robert	King	Sales Representative
编辑	8	Laura	Callahan	Inside Sales Coordinator
编辑	9	Moussa	Jones	Sales Representative

步骤1: 创建页面, 并添加GridView控件

```
<asp:GridView ID="gvEmployee" runat="server"
    AutoGenerateColumns="False"
    onrowediting="gvEmployee_RowEditing"
    onrowupdating="gvEmployee_RowUpdating">
    onrowcancelingedit="gvEmployee_RowCancelingEdit"
    <Columns>
        <asp:CommandField HeaderText="Edit" ShowEditButton="True" />
        <asp:BoundField DataField="EmployeeID" HeaderText="EmployeeID"
            ReadOnly="True" />
        <asp:BoundField DataField="FirstName" HeaderText="FirstName" />
        <asp:BoundField DataField="LastName" HeaderText="LastName" />
        <asp:BoundField DataField="Title" HeaderText="Title" />
    </Columns>
</asp:GridView>
```

步骤2: 在Page_Load事件中编码填充GridView

```
protected void Page_Load (object sender, EventArgs e) {  
    if (!Page.IsPostBack) {    bindgv(); }  
}  
  
private void bindgv() {  
    using (SqlConnection conn = new SqlConnection(connstr)) {  
        conn.Open();  
        SqlCommand cmd = conn.CreateCommand();  
        cmd.CommandText =  
            "SELECT EmployeeID, FirstName, LastName, Title FROM Employees";  
        SqlDataReader dr = cmd.ExecuteReader();  
        gvEmployee.DataSource = dr;  
        gvEmployee.DataBind();  
        dr.Close();  
        cmd.Dispose();  
    }  
}
```

步骤3: 处理Edit按钮事件

```
protected void gvEmployee_RowEditing
    (object sender, GridViewEditEventArgs e) {
    gvEmployee.EditIndex = e.NewEditIndex;
    bindgv();
}
```

步骤4: 处理Cancel按钮事件

```
protected void gvEmployee_RowCancelingEdit
    ( object sender, GridViewCancelEventArgs e) {
    gvEmployee.EditIndex = -1;
    bindgv();
}
```

步骤5: 处理Update按钮事件

```
protected void gvEmployee_RowUpdating
    (object sender, GridViewUpdateEventArgs e) {
    GridViewRow row = gvEmployee.Rows[e.RowIndex];
    string id = row.Cells[1].Text;
    string fname = ((TextBox)row.Cells[2].Controls[0]).Text;
    string lname = ((TextBox)row.Cells[3].Controls[0]).Text;
    string title = ((TextBox)row.Cells[4].Controls[0]).Text;
    updategv(id, fname, lname, title);

    gvEmployee.EditIndex = -1;
    bindgv();
}
```

续：处理Update按钮事件

```
private void updategv(string id, string fname, string lname, string title) {  
    using (SqlConnection conn = new SqlConnection(connstr)) {  
        conn.Open();  
        SqlCommand cmd = conn.CreateCommand();  
        cmd.CommandText = @"Update Employees set FirstName=@fname,  
                             LastName=@lname, Title=@title  
                             Where EmployeeID=@id";  
        cmd.Parameters.AddWithValue("@id", id);  
        cmd.Parameters.AddWithValue("@fname", fname);  
        cmd.Parameters.AddWithValue("@lname", lname);  
        cmd.Parameters.AddWithValue("@title", title);  
        cmd.ExecuteNonQuery();  
        cmd.Dispose();  
    }  
}
```

➤ 在GridView中使用模板列

- 使用模板（Template）定制单条记录的显示格式
- 数据绑定控件自动将模板应用于所有要显示的记录
- 使用模板列，可以完全按自己的想法来布置页面，摆脱固有模板的限制，提供更大的灵活性



示例 列表显示所有员工的详细信息

由于员工信息数据项较多，而且有的数据项很长，难以在传统的表行中显示出来，所以要定制模板

自定义员工信息显示模板

```
<asp:GridView ID="gvEmployee" runat="server"
    AutoGenerateColumns="False" GridLines="None">
    <HeaderStyle BackColor="Silver" />
    <Columns>
        <asp:TemplateField HeaderText="Employees">
            <ItemTemplate> <b>
                <%# Eval ("EmployeeID") %>-<%# Eval ("FirstName") %>
                <%# Eval ( "LastName") %> <hr /></b>
                <small> <%# Eval ("Address") %><br />
                <%# Eval ("City") %>, <%# Eval ("Country") %><br />
                <%# Eval ("HomePhone") %><br />
                <%# Eval ("Notes") %><br /><br /></small>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

常用的模板类型及用途：

类型	用途
ItemTemplate	普通项目模板，用于浏览状态
AlternatingItemTemplate	交替项模板，浏览状态下可使用该模板使相邻的两行数据采用不同的样式显示，增强对比度。
EditItemTemplate	编辑项模板， 只对当前EditIndex指向的项目起作用。
HeaderTemplate	表头的模板
FooterTemplate	页脚的模板



使用模板列编辑员工的称谓及备注信息

设计思路：

- 称谓只能从 “Mr.”、 “Dr.”、 “Ms.”及 “Mrs.”四者中选择一个。
- 备注信息较长，要使用多行文本框编辑
- 针对浏览和编辑，分别定制ItemTemplate和EditItemTemplate模板

普通项模板：用于浏览数据

```
<ItemTemplate><b>  
    <%=# Eval("EmployeeID") %> - <%=# Eval ("TitleOfCourtesy")%>  
    <%=# Eval ("FirstName") %> <%=# Eval ("LastName") %> <hr /></b>  
    <%=# Eval ("Address") %><br /> <%=# Eval ("HomePhone") %><br />  
    <%=# Eval ("Notes") %><br /><br />  
</ItemTemplate>
```

编辑项模板：用于更改数据

```
<EditItemTemplate><b>
    <asp:Label ID="lblid" runat="server"
        Text='<%# Bind ("EmployeeID") %>' /> -
    <asp:DropDownList ID="ddltitle" runat="server"
        DataSource="<%# CourtesyTitle %>"
        SelectedValue='<%# Bind ("TitleOfCourtesy") %>'>
</asp:DropDownList>
    <%# Eval ("FirstName") %>
    <%# Eval ("LastName") %><hr /></b>
    <%# Eval ("Address") %><br />
    <%# Eval ("HomePhone") %><br />
    <asp:TextBox ID="tbxNotes" runat="server"
        Text='<%# Bind ("Notes") %>' TextMode="MultiLine" </asp:TextBox>
    <br />
</EditItemTemplate>
```

关于绑定方式:

- **Eval**方式: 单向绑定, 只需将数据源的数据绑定到控件上显示, 不需回传控件的值给数据源
- **Bind**方式: 双向绑定, 要将更改后的值回传给**DataSource**控件, 以便它能够以这些数据为参数执行更新语句

使用模板列显示和编辑员工信息 - Windows Internet Explorer

http://localhost:2717/databindctls/TempEdit.aspx

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

★ 收藏夹 使用模板列显示和编辑员工信息

Edit Employees Information

1 - Ms. Nancy Davolio

Mr.

507 Dr. Ave. E. Apt. 2A

(20) Ms. 9857

Mrs.

Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call."

2 - Dr. And Fuller

908 W. Capital Way

(206) 555-9482

Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.

操作

更新 取消

编辑

完成

本地 Intranet 105%

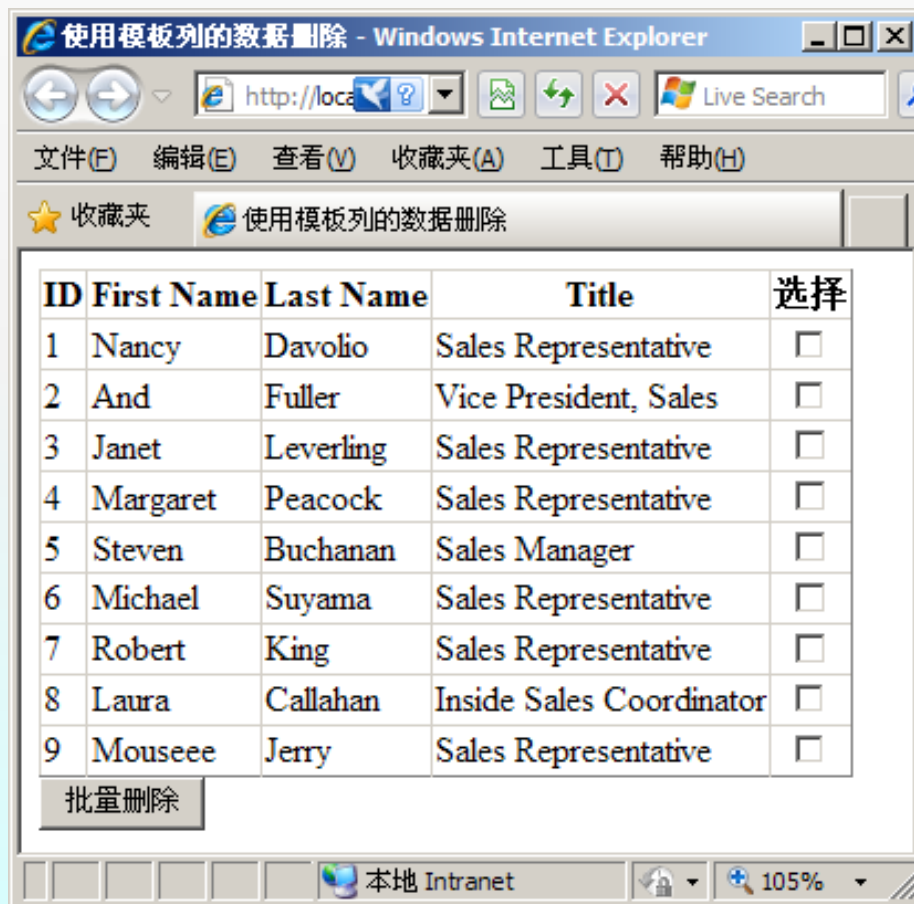


示例

使用模板列实现多条记录的批量删除

设计思路：

- 使用模板列在每行显示一个复选框
- 用户点选多个复选框后，再获取所有选择项的主键，进行批量操作。



页面控件声明

```
<asp:GridView ID="gvEmployee" runat="server"
    AutoGenerateColumns="False" DataKeyNames="employeeid">
    <Columns>
        <asp:BoundField DataField="EmployeeID" HeaderText="ID"
            ReadOnly="True" />
        <asp:BoundField DataField="FirstName" HeaderText="First Name" />
        <asp:BoundField DataField="LastName" HeaderText="Last Name" />
        <asp:BoundField DataField="Title" HeaderText="Title" />
        <asp:TemplateField HeaderText="选择">
            <ItemTemplate>
                <asp:CheckBox ID="cbxSel" runat="server" />
            </ItemTemplate>
            <ItemStyle HorizontalAlign="Center" />
        </asp:TemplateField>
    </Columns>
</asp:GridView>

<asp:Button ID="btndel" runat="server" Text="批量删除"
    onclick="btndel_Click" OnClientClick="return confirm('确认删除吗 ? ');" />
```

批量删除按钮的事件处理代码

```
protected void btnDel_Click(object sender, EventArgs e) {  
    List<int> ids = getselectedids ( ) ;  
    if (ids.Count > 0) {  
        StringBuilder sb = new StringBuilder( );  
        sb.Append( "Delete from Employees where employeeid in (" );  
        foreach (int eid in ids) {    sb.Append(eid);  sb.Append(",");    }  
        sb[sb.Length - 1] = ')';  
        delselected(sb.ToString( ) ) ;  
    }  
}
```


获取所有被选员工代码的函数

```
private List<int> getselectedids( ) {  
    List<int> list = new List<int>( ) ;  
    foreach (GridViewRow row in gvEmployee.Rows) {  
        if (row.RowType == DataControlRowType.DataRow) {  
            CheckBox cbx = row.FindControl ("cbxsel") as CheckBox ;  
            if ( cbx != null && cbx.Checked ) {  
                list.Add ((int)gvEmployee.DataKeys [row.RowIndex]. Value);  
            }  
        }  
    }  
    return list;  
}
```

8.3.2 ListView控件

➤ 概述

- .NET 3.5中新增的控件，取代以往版本中的Repeater
- 是一个非常灵活的轻量级的控件，完全根据自定义的模板来呈现内容
- 提供了对选择、编辑等高级特性的支持
- 使用ListView最常见的原因是为了创建特殊的布局

➤ ListView可使用的模板

列	描述
ItemTemplate	设置所有数据项（没有使用 AlternatingItemTemplate 时）或奇数行数据项（使用 AlternatingItemTemplate 时）的内容和格式
AlternatingItemTemplate	和 ItemTemplate 配合，设置偶数行的内容和格式
ItemSeparatorTemplate	设置在项目中间绘制的分隔符的格式
SelectedItemTemplate	设置当前选定项目的内容和格式
EditItemTemplate	设置数据项在编辑模式下使用的控件
InsertItemTemplate	设置插入新项目时使用的控件
LayoutTemplate	设置包装项目列表的标记
GroupTemplate	若使用了分组功能，则设置包装项目组的标记
GroupSeparatorTemplate	设置项目组的分隔符格式
EmptyDataTemplate	当绑定的数据对象为空时（没有记录或对象），使用该模板设置显示的提示信息



使用ListView控件显示员工的信息

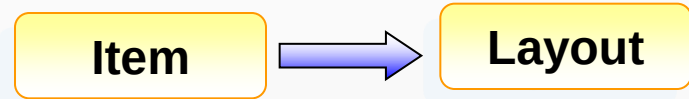
ListView控件的声明

```
<asp:ListView ID="lvEmployee" runat="server" DataSourceID="dsEmp">
  <LayoutTemplate>
    <span id="itemPlaceholder" runat="server"></span>
  </LayoutTemplate>
  <ItemTemplate><b>
    <%# Eval("EmployeeID") %> -
    <%# Eval("TitleOfCourtesy")%>
    <%# Eval("FirstName") %>
    <%# Eval("LastName") %>
    <hr /></b>
    <%# Eval("Address") %><br />
    <%# Eval ("HomePhone") %><br />
    <%# Eval ("Notes") %><br /> <br />
  </ItemTemplate>
</asp:ListView>
```

说明:

- **ListView中至少要定义两个模板:**

- 项目模板: **ItemTemplate**
- 布局模板: **LayoutTemplate**



- **通过一个占位符将项目添加到布局中**
- **这个占位符的ID属性一定要用itemPlaceholder**

➤ 使用GroupTemplate分组项

```
<asp:ListView ID="lvEmployee" runat="server" GroupItemCount="3"
  DataSourceID="dsEmployee" >
  <LayoutTemplate>
    <table border="0" cellpadding="10" width="100%">
      <tr id="groupPlaceholder" runat="server"></tr> </table>
    </LayoutTemplate>
    <GroupTemplate>
      <tr><td runat="server" id="itemPlaceholder" /></tr>
    </GroupTemplate>
    <ItemTemplate>
      <td valign="top"> <b>
        <%=# Eval("EmployeeID") %> - <%=# Eval("TitleOfCourtesy")%>
        <%=# Eval("FirstName") %> <%=# Eval("LastName") %> </b><hr />
        <%=# Eval("Address") %><br /> <%=# Eval("HomePhone") %><br />
        <%=# Eval("Notes") %><br /> </td>
      </ItemTemplate>
    </asp:ListView>
```

说明:

Item



Group



Layout

在ListView中使用GroupTemplate - Windows Internet Explorer

http://localhost:2717/databindctls/ListViewGroup.aspx

文件(F) 编辑(E) 查看(V) 收藏夹(A) 工具(T) 帮助(H)

★ 收藏夹 在ListView中使用GroupTemplate

1 - Ms. Nancy Davolio 507 - 20th Ave. E. Apt. 2A (206) 555-9857 Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call."	2 - Dr. And Fuller 908 W. Capital Way (206) 555-9482 Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.	3 - Ms. Janet Leverling 722 Moss Bay Blvd. (206) 555-3412 Janet has a BS degree in chemistry from Boston College (1984). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.
4 - Mrs. Margaret Peacock 4110 Old Redmond Rd. (206) 555-8122 Margaret holds a BA in English literature from Concordia College	5 - Mr. Steven Buchanan 14 Garrett Hill (71) 555-4848 Steven Buchanan graduated from St. Andrews University, Scotland, with a BSC degree in 1976	6 - Mr. Michael Suyama Coventry House Miner Rd. (71) 555-7773 Michael is a graduate of Sussex University (MA, economics, 1983) and

完成

本地 Intranet 105%

8.3.3 DetailsView控件

➤ 概述

- 每次显示一条记录
- 包含一个可选的分页按钮，用于在一组记录间导航
- 使用表格布局的方式，每次显示一条记录，将每个数据项显示在表格的一行中



示例

使用DetailsView控件浏览员工信息

```
<asp:DetailsView ID="dvEmployee" runat="server"
    AutoGenerateRows="False" DataKeyNames="EmployeeID"
    DataSourceID="dsEmployee" AllowPaging="True" >
    <Fields>
        <asp:BoundField DataField="EmployeeID" HeaderText="EmployeeID" />
        <asp:BoundField DataField="FirstName" HeaderText="FirstName" />
        <asp:BoundField DataField="LastName" HeaderText="LastName" />
        <asp:BoundField DataField="Address" HeaderText="Address" />
        <asp:BoundField DataField="HomePhone" HeaderText="Phone" />
        <asp:BoundField DataField="Notes" HeaderText="Notes" />
    </Fields>
</asp:DetailsView>

<asp:SqlDataSource ID="dsEmployee" runat="server"
    ConnectionString="<%%$ ConnectionStrings:nwconnstr %>"
    SelectCommand="SELECT EmployeeID, FirstName, LastName, Address,
        HomePhone,Notes FROM Employees" >
</asp:SqlDataSource>
```

➤ 在DetailsView中编辑数据

- 设置相应属性，即可自动生成相应按钮
 - AutoGenerateInsertButton
 - AutoGenerateEditButton
 - AutoGenerateDeleteButton
- 提供了3种操作模式
 - 只读模式
 - 插入模式
 - 编辑模式
- 使用CurrentMode属性获取当前的模式
- 使用ChangeMode()方法改变当前模式

8.3.4 FormView控件

➤ 概述

- 可以完全控制单条记录显示及编辑的样式
- 完全依赖于模板，提供最大的灵活性
- 可以使用多种模板
 - ItemTemplate
 - EditItemTemplate
 - InsertItemTemplate
 - EmptyDataTemplate
 - HeaderTemplate
 - FooterTemplate
 - PagerTemplate
 -



使用FormView开发一个员工信息的增删改查程序

声明数据源

```
<asp:SqlDataSource ID="dsEmployee" runat="server"
    ConnectionString="<%%$ ConnectionStrings:nwconnstr %>"
    SelectCommand="SELECT EmployeeID, FirstName, LastName,
                    TitleOfCourtesy, Notes FROM Employees"
    UpdateCommand="Update Employees set FirstName=@firstname,
                    LastName=@lastname, TitleOfCourtesy=@titleofcourtesy, Notes=@notes
                    Where EmployeeID=@employeeid"
    InsertCommand="INSERT INTO Employees( LastName, FirstName,
                    TitleOfCourtesy, Notes) VALUES (@lastname, @firstname,
                    @titleofcourtesy, @notes)"
    DeleteCommand="Delete From Employees Where EmployeeID=@employeeid">
</asp:SqlDataSource>
<asp:SqlDataSource ID="dsTitle" runat="server"
    ConnectionString="<%%$ ConnectionStrings:nwconnstr %>"
    SelectCommand="SELECT distinct TitleOfCourtesy FROM Employees">
</asp:SqlDataSource>
```

定制FormView模板：普通项目模板

```
<ItemTemplate>
    EmployeeID: <%# Eval("EmployeeID") %><br />
    FirstName: <%# Eval("FirstName") %><br />
    LastName: <%# Eval("LastName") %><br />
    TitleOfCourtesy: <%# Eval("TitleOfCourtesy") %><br />
    <asp:LinkButton ID="EditButton" runat="server" CausesValidation="False"
        CommandName="Edit" Text="编辑" />&nbsp;
    <asp:LinkButton ID="DeleteButton" runat="server" Text="删除"
        CausesValidation="False" CommandName="Delete" />&nbsp;
    <asp:LinkButton ID="NewButton" runat="server" CausesValidation="False"
        CommandName="New" Text="新建" />
</ItemTemplate>
```

定制FormView模板： 编辑项目模板

```
<EditItemTemplate>
```

```
EmployeeID: <asp:Label ID="EmployeeIDLabel1" runat="server"
```

```
Text='<%# Eval("EmployeeID") %>' /><br />
```

```
FirstName: <asp:TextBox ID="FirstNameTextBox" runat="server"
```

```
Text='<%# Bind("FirstName") %>' /> <br />
```

```
LastName: <asp:TextBox ID="LastNameTextBox" runat="server"
```

```
Text='<%# Bind("TitleOfCourtesy") %>' /> <br />
```

```
TitleOfCourtesy: <asp:DropDownList ID="ddltitle" runat="server"
```

```
DataSourceID="dsTitle" DataValueField="TitleOfCourtesy"
```

```
SelectedValue='<%# Bind("TitleOfCourtesy") %>' /> <br />
```

```
<asp:LinkButton ID="UpdateButton" runat="server"
```

```
CausesValidation="True" CommandName="Update" Text="更新" />
```

```
<asp:LinkButton ID="UpdateCancelButton" runat="server"
```

```
CausesValidation="False" CommandName="Cancel" Text="取消" />
```

```
</EditItemTemplate>
```

定制FormView模板：新建项目模板

<InsertItemTemplate>

FirstName: <asp:TextBox ID="FirstNameTextBox" runat="server"

Text='<%# Bind("FirstName") %>' />

LastName: <asp:TextBox ID="LastNameTextBox" runat="server"

Text='<%# Bind("LastName") %>' />

TitleOfCourtesy: <asp:DropDownList ID="ddltitle" runat="server"

DataSourceID="dsTitle" DataValueField="TitleOfCourtesy"

SelectedValue='<%# Bind("TitleOfCourtesy") %>' />

<asp:LinkButton ID="InsertButton" runat="server" CausesValidation="True"

CommandName="Insert" Text="插入" />

<asp:LinkButton ID="InsertCancelButton" runat="server"

CausesValidation="False" CommandName="Cancel" Text="取消" />

</InsertItemTemplate>

➤ 总结

- **FormView也支持只读、插入和编辑三种模式，也可在各种模式之间切换**
- **FormView不支持自动创建按钮列，你必须手工创建各种按钮对象**
- **所有按钮的CommandName属性必须设置为合适的值，这样才能触发相应的事件**

FormView中可以使用的CommandName值

命令	作用
Edit	适用于ItemTemplate，从只读模式切换到编辑模式以编辑当前项
Cancel	适用于EditItemTemplate和InsertItemTemplate，在编辑或插入模式下放弃数据，返回到只读模式
Update	适用于EditItemTemplate，将编辑后的数据保存下来，并返回只读模式
New	适用于ItemTemplate，插入一条新数据并转入编辑模式
Insert	适用于InsertItemTemplate，将插入的数据保存下来，并返回只读模式
Delete	适用于ItemTemplate，直接删除当前项

8.4 实例：使用数据绑定控件的分类浏览商品信息页面

8.4.1 设计说明

该实例中，要显示图书类别信息作为导航菜单，并在点击某类别名时显示该类别下的所有图书信息。图书类别信息和图书列表信息各用一个ListView控件来显示，合理的设计项目显示模板会达到较好的页面效果。

8.4.2 程序实现

