第 8 章　Web 窗体的数据控件

在了解了 ADO.NET 基础后，就可以使用 ADO.NET 提供的对象进行数据库开发和操作。ASP.NET 还提供了一些 Web 窗体的数据控件，开发人员能够智能的配置与数据库的连接，而不需要手动的编写数据库连接。ASP.NET 不仅提供了数据源控件，还提供了能够显示数据的控件，简化了数据显示的开发，开发人员只需要简单的修改模板就能够实现数据显示和分页。

8.1　数据源控件

数据源控件很像 ADO.NET 中的 Connection 对象，数据源控件用来配置数据源，当数据控件绑定数据源控件时，就能够通过数据库源控件来获取数据源中的数据并显示。而无需通过程序实现数据源代码的编写。

8.1.1　SQL 数据源控件（SqlDataSource）

SqlDataSource 控件代表一个通过 ADO.NET 连接到 SQL 数据库提供者的数据源控件。并且 SqlDataSource 能够与任何一种 ADO.NET 支持的数据库进行交互，这些数据库包括 SQL Server、ACCESS、Oledb、Odbc 以及 Oracle。

SqlDataSource 控件能够支持数据的检索、插入、更新、删除、排序等，以至于数据绑定控件可以在这些能力被允许的条件下自动的完成该功能，而不需要手动的代码实现。并且 SqlDataSource 控件所属的页面被打开时，SqlDataSource 控件能够自动的打开数据库，执行 SQL 语句或存储过程，返回选定的数据，然后关闭连接。SqlDataSource 控件强大的功能极大的简化了开发人员的开发，缩减了开发中的代码。但是 SqlDataSource 控件也有一些缺点，就是在性能上不太适应大型的开发，而对于中小型的开发，SqlDataSource 控件已经足够了。

1．建立 SqlDataSource 控件

ASP.NET 提供的 SqlDataSource 控件能够方便的添加到页面，当 SqlDataSource 控件被添加到 ASP.NET 页面中时，会生成 ASP.NET 标签，示例代码如下所示。

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"></asp:SqlDataSource>
```

切换到视图模式下，点击 SqlDataSource 控件会显式【配置数据源……】，单击【配置数据源……】连接时，系统能够智能的提供 SqlDataSource 控件配置向导，如图 8-1 所示。

在新建数据源后，开发人员可以选择是否保存在 web.config 数据源中以便应用程序进行全局配置，通常情况下选择保存。由于现在没有连接，单击【新建连接】按钮选择或创建一个数据源。单击后，系统会弹出对话框用于选择数据库文件类型，如图 8-2 所示。

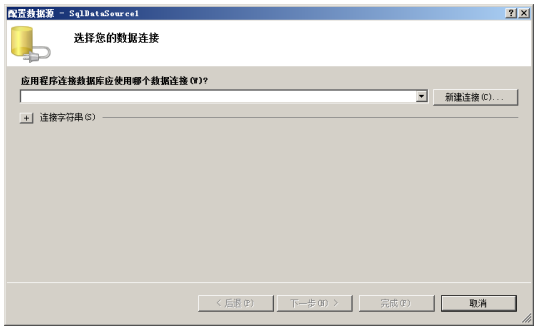


图 8-1　配置 SqlDataSource 控件

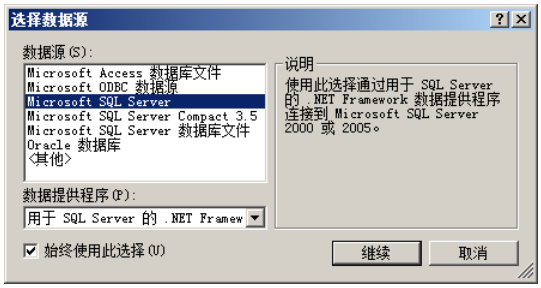


图 8-2　选择数据源

当选择完后，配置信息就会显式在 web.config 中。当需要对用户控件进行维护时，可以直接修改 web.config，而不需要修改每个页面的数据源控件，这样就方便了开发和维护。当选择了数据源后，需要对数据源的连接进行配置，这一步与 ADO.NET 中的 Connection 对象一样，就是要与数据库建立连接，当配置好连接后，可以单击【测试连接】按钮来测试是否连接成功，如图 8-3 和图 8-4 所示。

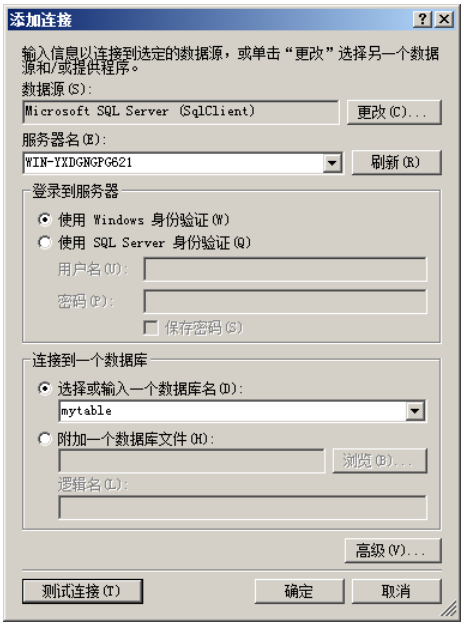


图 8-3　添加连接

图 8-4　测试连接

连接成功后，单击【确定】按钮，系统会自动添加连接，如图 8-5 所示。连接添加成功后，在 web.config 配置文件中，就有该连接的连接字串，代码如下所示。

```
<connectionStrings>
    <add name="mytableConnectionString" connectionString="Data
    Source=WIN-YXDGNGPG621;Initial Catalog=mytable; Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

数据源控件可以指定开发人员所需要使用的 Select 语句或存储过程，开发人员能够在配置 Select 语句窗口中进行 Select 语句的配置和生成，如果开发人员希望手动编写 Select 语句或其他语句，可以单击【指定自定义 SQL 语句或存储过程】按钮进行自定义配置，Select 语句的配置和生成如图 8-6 所示。

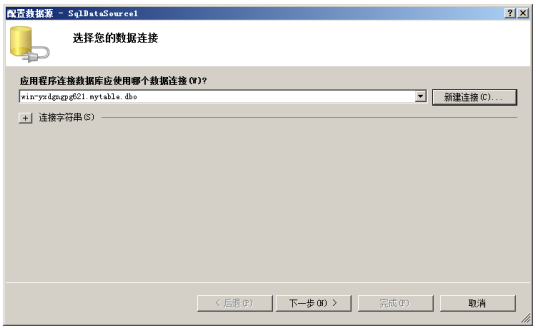


图 8-5　成功添加连接

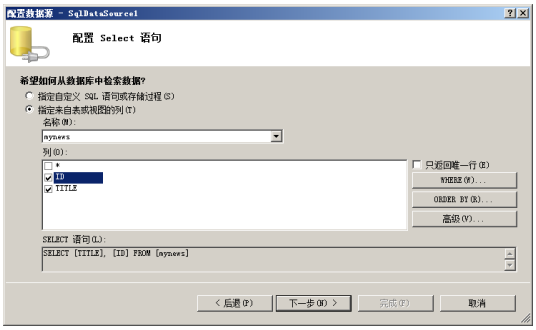


图 8-6　配置使用 Select 语句

对于开发人员，只需要勾选相应的字段，选择 Where 条件和 Order By 语句就可以配置一个 Select 语句。但是，通过选择只能够查询一个表，并实现简单的查询语。如果要实现复杂的 SQL 查询语句，可以单击【指定自定义 SQL 语句或存储过程】进行自定义 SQL 语句或存储过程的配置，如图 8-7 所示，开发人员选择了一个 getdetail 的存储过程作为数据源。

单击【下一步】按钮，就需要对相应的字段进行配置，这些字段就像 ADO.NET 中的参数化查询一样。在数据源控件中，也是通过@来表示参数化变量，当需要配置相应的字段，例如配置 WHERE 语句等就需要对参数进行配置，如图 8-8 所示。

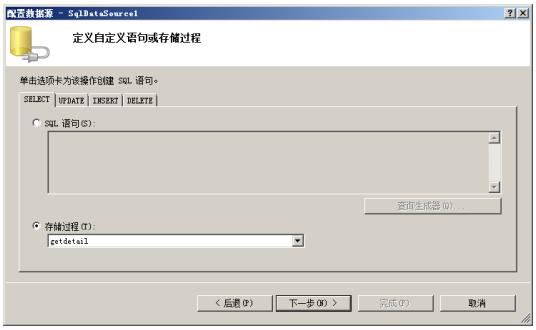


图 8-7　定义自定义语句或存储过程

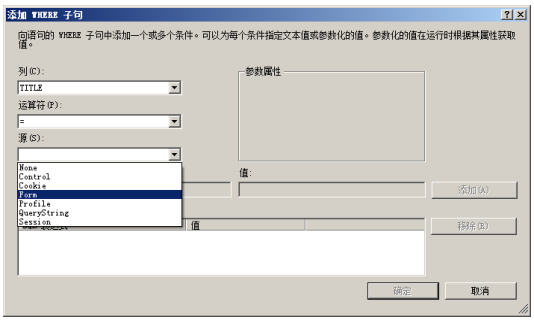


图 8-8　添加 WHERE 子句

添加 WHERE 子句时，SQL 语句中的值可以选择默认值、控件、Cookie 或者是 Session 等。当配置完成后，就可以测试查询，如果测试后显示的结果如预期一样，则可以单击完成，如图 8-9 所示。

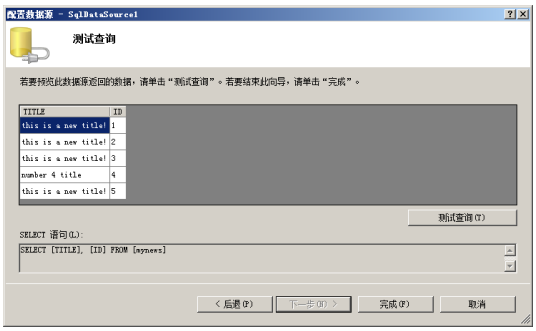


图 8-9　测试查询并完成

完成后，SqlDataSource 控件标签代码如下所示。

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:mytableConnectionString %>"
    SelectCommand="SELECT [TITLE], [ID] FROM [mynews]">
```

```
</asp:SqlDataSource>
```

2．配置 SqlDataSource 控件属性

SqlDataSource 控件还包括一些可视化属性，这些属性包括删除查询（DeleteQuery）、插入查询（InsertQuery）、检索查询（SelectQuery）以及更新查询（UpdateQuery）。当需要使用可视化属性时，需选择【使用自定 SQL 语句或存储过程】复选框，在导航中可以使用查询生成器生成查询语句，如图 8-10 所示。

图 8-10　自定义语句或存储过程

选择【查询生成器】按钮，系统会提示选择相应的表并通过相应的表来生成查询语句，如图 8-11 和图 8-12 所示。

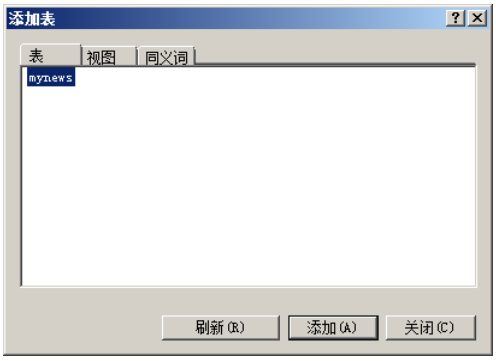


图 8-11　选择相应的表

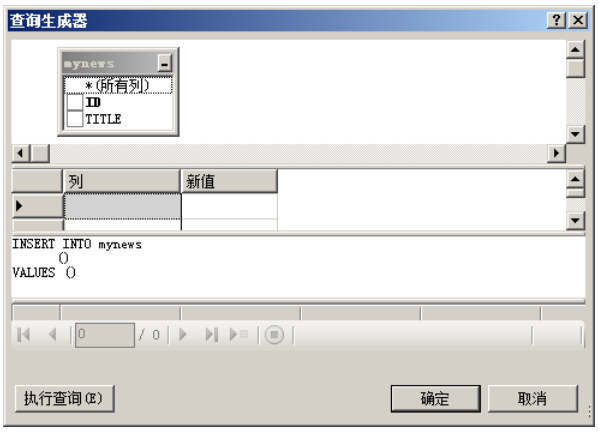


图 8-12　使用查询生成器

配置相应的查询语句后，SqlDataSource 控件的 HTML 代码如下所示。

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:mytableConnectionString %>"
    InsertCommand="INSERT INTO mynews(ID) VALUES ('control title')"
    SelectCommand="SELECT [TITLE], [ID] FROM [mynews]">
</asp:SqlDataSource>
```

上述代码自动增加了一个 InsertCommand 并指定了 Insert 语句。开发人员可以为 SqlDataSource 控件指定四个命令参数：SelectCommand、UpdateCommand、DelectCommad 和 InsertCommand。每个都是数据源控件的单一属性，开发人员可以配置相应的语句指定 Select、Update、Delete 以及 Insert 方法。

SqlDataSource 控件同时能够使用缓存来降低页面与数据库之间连接频率，这样可以避免开销很大的查询操作，以及建立连接和关闭连接操作。只要数据库是相对稳定不变的，则可以使用 SqlDataSource 控件的缓存属性（EnableCaching）来进行缓存。在默认情况下，缓存属性（EnableCaching）是关闭的，

需要开发人员自行设置缓存属性。

8.1.2 Access 数据源控件（AccessDataSource）

在上一章中介绍了如何使用 ADO.NET 中 OleDb 来连接和读取 Access 数据库。Access 数据库是一种桌面级的数据库，当对应用程序性能，以及数据库性能要求不是很高，并且数据量不需很大时，可以考虑选择 Access 数据库。

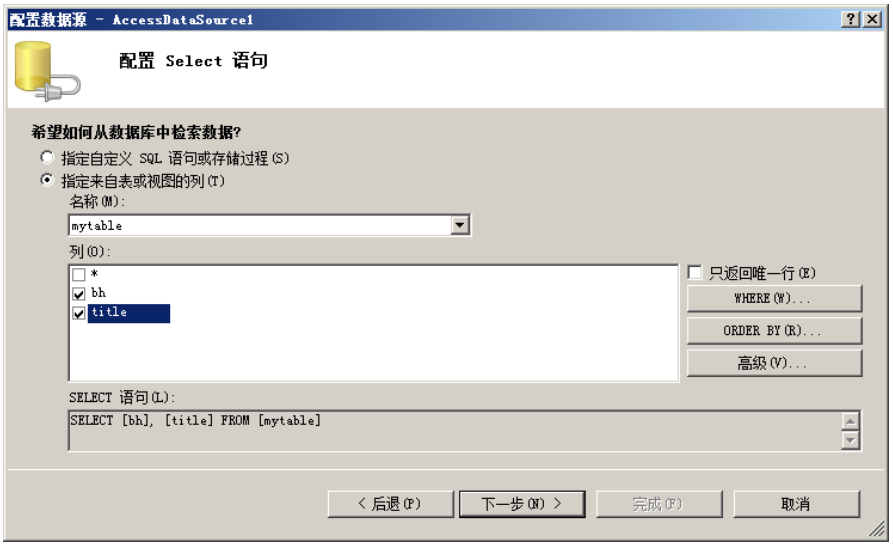
SqlDataSource 能够与任何一种 ADO.NET 支持的数据源进行交互，这些数据源包括 SQL Server、Access、Oledb、Odbc 以及 Oracle。但是 Access 数据库有专门的数据源控件，就是 AccessDataSource。AccessDataSource 控件同配置 SqlDataSource 控件基本相同，如图 8-13 所示。

与 SqlDataSource 不同的是，SqlDataSource 主要采用的是 ConnectionString 属性连接数据库，而 Access 则采用的是 AccessDataSource 方式连接数据库。因为 Access 数据库是以文件的形式存在于系统中的，所以主要采用 DataFile 属性直接以文件地址的方式进行连接。要连接 Access 数据库，则必须选择 Access 数据库文件，如图 8-14 所示。

图 8-13　选择数据库

图 8-14　选择 Access 文件

在选择了 Access 数据库文件后，单击【确定】按钮，系统就会为开发人员配置连接字串，在核对无误后，单击【下一步】按钮进入 Select 语句的配置。同 SqlDataSource 控件一样，同样能够配置 Select 语句或自定义存储过程，如图 8-15 所示。



同样 8-15　配置 Access 数据库的 Select 语句

其他步骤与 SqlDataSource 相同，当创建完成后，AccessDataSource 控件的 HTML 代码如下所示。

```
<asp:AccessDataSource ID="AccessDataSource1" runat="server"
    DataFile="~/acc.mdb"
    SelectCommand="SELECT [bh], [title] FROM [mytable]">
</asp:AccessDataSource>
```

当需要使用 Access 数据库，推荐将 Access 数据库文件保存在 App_Data 文件夹中。以保证数据库文件是私有的，因为 ASP.NET 不允许直接请求 App_Data 文件夹。

注意：AccessDataSource **控件不支持访问受密码保护的** Access **数据库文件，如果需要访问受密码保护的** Access **数据库文件，则需要使用** SqlDataSource **控件。**

8.1.3 目标数据源控件（ObjectDataSource）

大多数 ASP.NET 数据源控件，如 SqlDataSource 都是在两层应用程序层次结构中使用。在该层次结构中，表示层（ASP.NET 网页）可以与数据层（数据库和 XML 文件等）直接进行通信。但是，常用的应用程序设计原则是将表示层与业务逻辑相分离，而将业务逻辑封装在业务对象中。这些业务对象在表示层和数据层之间形成一层，从而生成一种三层应用程序结构。ObjectDataSource 控件通过提供一种将相关页上的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。在不使用扩展代码的情况下，ObjectDataSource 使用中间层业务对象以声明方式对数据执行选择、插入、更新、删除、分页、排序、缓存和筛选操作。

也就是说，SqlDataSource 是两层模型中使用的.页面通过直接访问数据库.。ObjectDataSource 用于三层模型中，也就是将中间业务对象通过其访问数据库的。然后中间层业务对象再用在表示层中，例如在开发中使用的自定义控件。ObjectDataSource 的业务对象是可以用检索或更新数据的业务对象，例如 Bin 或 App_Code 目录中定义的对象，选择业务对象如图 8-16 所示。

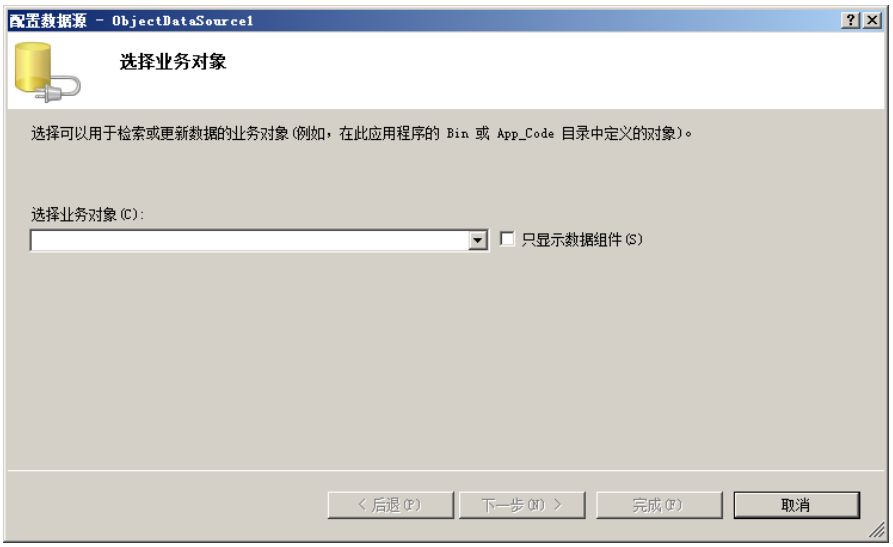


图 8-16　选择业务对象

可以创建一个类库，并在 ASP.NET 网站中添加引用，这样就可以通过 ObjectDataSource 对象选择该类库中的方法，如图 8-17 和图 8-18 所示。

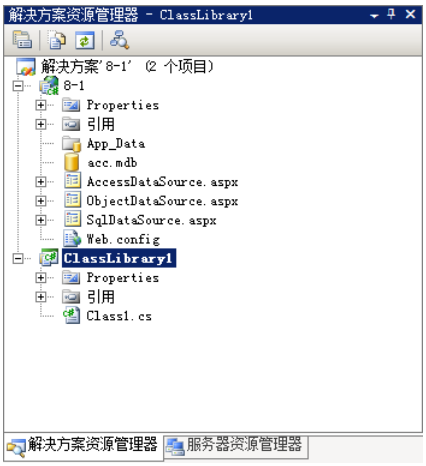


图 8-17　添加类库

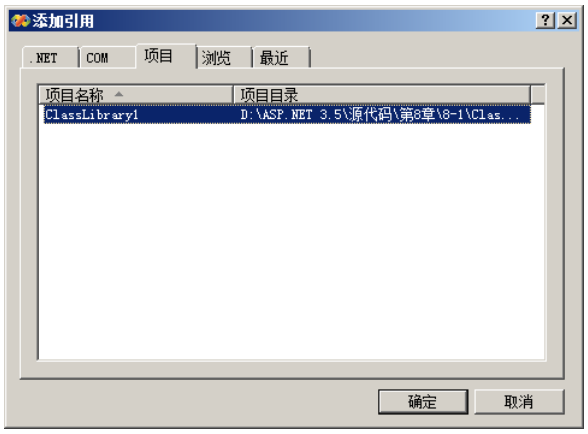


图 8-18　添加引用

ObjectDataSource 控件对象模型类似于 SqlDataSource 控件。ObjectDataSource 公开一个 TypeName 属性，该属性指定要实例化来执行数据操作的对象类型，也就是类的名称。与 SqlDataSource 的命令属性类似，同样 ObjectDataSource 包括四个重要属性，这四个属性分别为 SelectMethod、UpdateMethod、InsertMethod 和 DeleteMethod，分别用于指定要执行这些数据操作关联类型的方法。选择对象后，就可以配置 SelectMethod、UpdateMethod、InsertMethod 和 DeleteMethod 属性的方法。示例代码如下所示。

```
public class Class1                                    //创建类库
{
    public string GetTitle()                           //创建方法
    {
        name = "title";                                //变量赋值
        return name;                                   //返回 name
    }
    public void InsertTitle()                          //创建方法
    {
        name = "insert";                               //变量赋值
    }
    public string name;                                //创建共有变量 name
}
```

ObjectDataSource 控件可以使用 Class1 中的对象，如图 8-19 所示。

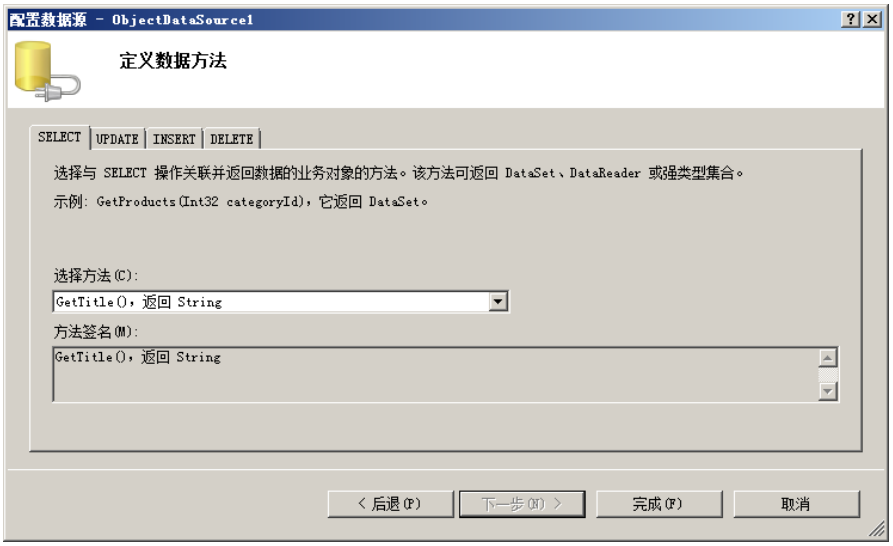


图 8-19　定义数据方法

ObjectDataSource 控件可以使开发人员将诸如 GridView 和 DropDownList 这样的用户界面控件绑定到一个中间层组件。能够无需编写任何代码即可绑定到一个组件，从而极大的简化用户界面。与其他的数据源控件相同，ObjectDataSource 控件在运行时可以接受参数，并在参数集合中对参数进行管理。每一项数据操作都有一个相关的参数集合。对于选择操作，可以使用 SelectParameters 集合，对于更新操作，可以使用 UpdateParameters 集合，而给予 InsertParameters、UpdateParameters、DeleteParameters 集合，需要分别确定相应操作所需调用的方法。

8.1.4　LINQ 数据源控件（LinqDataSource）

语言集成查询（LINQ）是一种查询语法，它可定义一组查询运算符，以便在任何基于.NET 的编程语言中以一种声明性的方式来表示遍历、筛选和投影操作。数据对象可以是内存中的数据集合，或者是表示数据库中数据的对象。无需为每个操作编写 SQL 命令，即可检索或修改数据。

使用 LinqDataSource 控件，开发人员可以通过在标记文本中设置属性从而在 ASP.NET 网页中使用 LINQ。LinqDataSource 控件使用 LINQ to SQL 来自动生成数据命令。LINQ 数据源可以是 LINQ 数据库或数组等以集合形式表现的数据库，有关 LINQ 的知识会有专门的章节讲解，在这里使用数组作为数据源，示例代码如下所示。

```
public string[] arr={"1","2","3","4"};                                    //创建数组
```

在 ASP.NET 页面中使用 LINQ 数据源控件可以对 LINQ 数据源进行查询，LINQ 数据源控件代码如下所示。

```
<asp:LinqDataSource ID="LinqDataSource1" runat="server">
</asp:LinqDataSource>
```

创建了 LINQ 数据源控件，同样单击【配置数据源……】按钮可以进行 LINQ 数据源控件的数据源配置，如图 8-20 所示。

当选择上下文对象后，需要配置数据选择，LINQ 数据源控件同样支持 Group 和 Where 关键字，如图 8-21 所示。

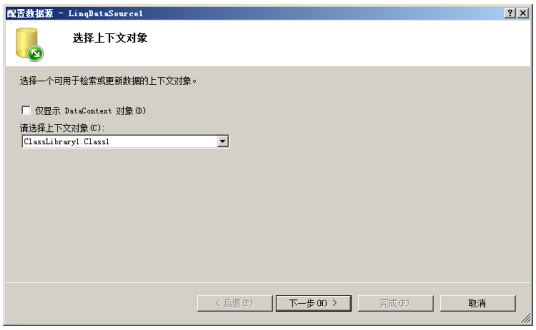


图 8-20　选择上下文对象

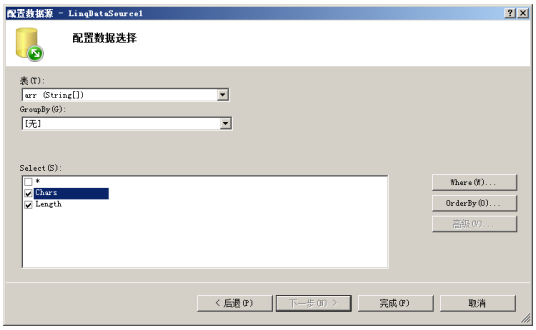


图 8-21　配置数据选择

配置完成后，LINQ 数据源控件 HTML 代码如下所示。

```
<asp:LinqDataSource ID="LinqDataSource1" runat="server"
    ContextTypeName="ClassLibrary1.Class1" Select="new (Length, Chars)"
    TableName="arr">
</asp:LinqDataSource>
```

当完成 LINQ 数据源控件（LinqDataSource）的配置后，就可以通过控件绑定 LINQ 数据源控件来获取 LINQ 数据库中的信息。LinqDataSource 控件按以下顺序应用数据操作：

- Where：指定要返回的数据记录。
- Order By：排序。
- Group By：聚合共享值的数据记录。
- Order Groups By：对分组数据进行排序。
- Select：指定要返回的字段或属性。
- Auto-sort：按用户选定的属性对数据记录进行排序。
- Auto-page：检索用户选定的数据记录的子集。

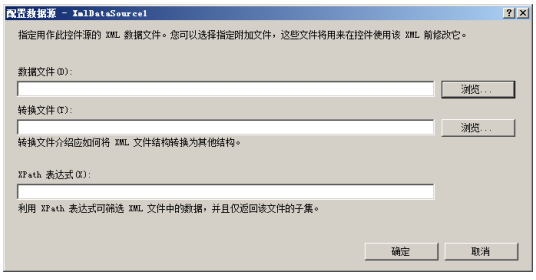
LINQ 是 ASP.NET 3.5 中增加的一种语言集成查询，该控件的高级属性和方法在 ASP.NET 3.5 与 LINQ 中会详细讲解。

8.1.5　Xml 数据源控件（XmlDataSource）

Xml 数据源控件可以让数据绑定控件轻易的连接到 XML 数据源。在只读方案下通常使用 XmlDataSource 控件显示分层 XML 数据，但同样可以使用该控件显示分层数据和表格数据。

1．建立 XmlDataSource 控件

与 AccessDataScource 相同的是，XmlDataSource 控件同样使用 DataFile 属性指定 XML 文件并加载 XML 数据，如图 8-22 所示。数据源是 XML 文件，单击【浏览】按钮选择数据文件，如图 8-23 所示。



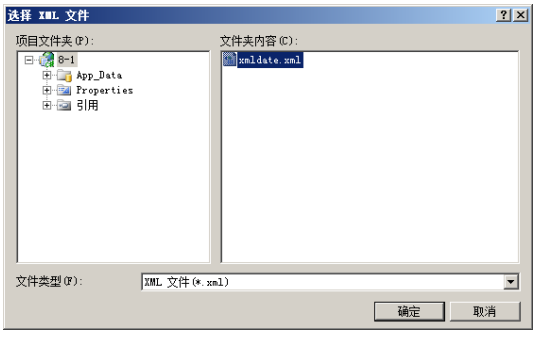


图 8-22　配置数据源　　图 8-23　选择 XML 数据源

选择数据源后，单击确定并完成数据源的配置即可，配置完成数据源后，XmlDataSource 控件的 HTML 代码如下所示。

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

上述代码指定了 DataFile 属性的所属的文件，当配置完成后，XmlDataSource 控件就可以和数据绑定控件结合使用了。

2．XmlDataSource 控件的使用

当配置完成 XmlDataSource 后，就可以和数据绑定控件结合使用。在使用数据绑定控件前，先配置 XML 数据文件，示例代码如下所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<news>
    <title>新闻标题 1</title>
    <time>2008</time>
    <author>guojing</author>
    <content>这是新闻正文</content>
    <title>新闻标题 2</title>
    <time>2008</time>
    <author>guojing</author>
    <content>这是新闻正文</content>
</news>
```

上述代码配置了 XML 数据文件，配置完成后，可以通过数据绑定控件来访问，可以使用 TreeView 控件，示例代码如下所示。

```
<asp:TreeView ID="TreeView1" runat="server" DataSourceID="XmlDataSource1">
</asp:TreeView>
```

上述代码只能够显示 XML 数据文件中各个节点的名称，并不能显示各个节点的值，必须为显示的节点做配置。在控件侧边单击【TreeNode 数据绑定】选项，并选择相应的列进行节点配置，如图 8-24 所示。

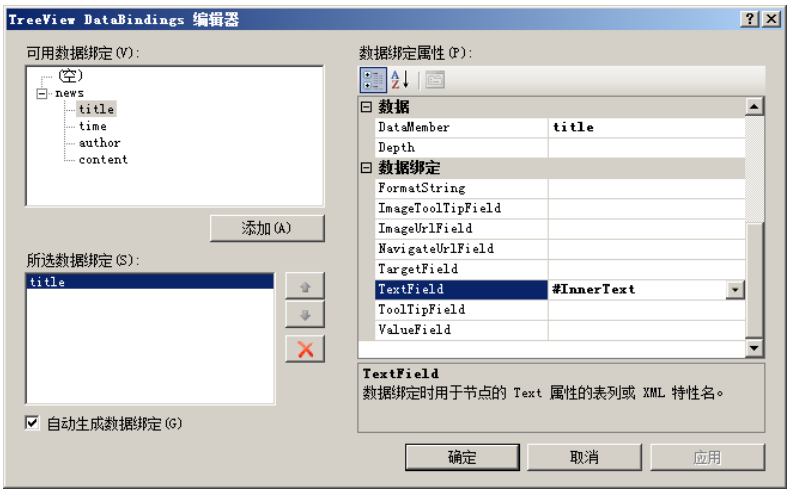


图 8-24　选择列配置 TextFiled

配置 TextFiled 后，各个节点的值会显示为 XML 数据中标签内的值，而 XmlDataSource 控件的 HTML

代码则会被系统自动替换，示例代码如下所示。

```
<asp:TreeView ID="TreeView1" runat="server" DataSourceID="XmlDataSource1"
    ImageSet="Contacts" NodeIndent="10">
        <ParentNodeStyle Font-Bold="True" ForeColor="#5555DD" />
        <HoverNodeStyle Font-Underline="False" />
        <SelectedNodeStyle Font-Underline="True" HorizontalPadding="0px"
        VerticalPadding="0px" />
    <DataBindings>
        <asp:TreeNodeBinding DataMember="title" Text="title" TextField="#InnerText" Value="title" />
    </DataBindings>
    <NodeStyle Font-Names="Verdana" Font-Size="8pt" ForeColor="Black"
    HorizontalPadding="5px" NodeSpacing="0px" VerticalPadding="0px" />
</asp:TreeView>
```

运行后，相应的节点则会显示为标签的相应的值，如图 8-25 所示。

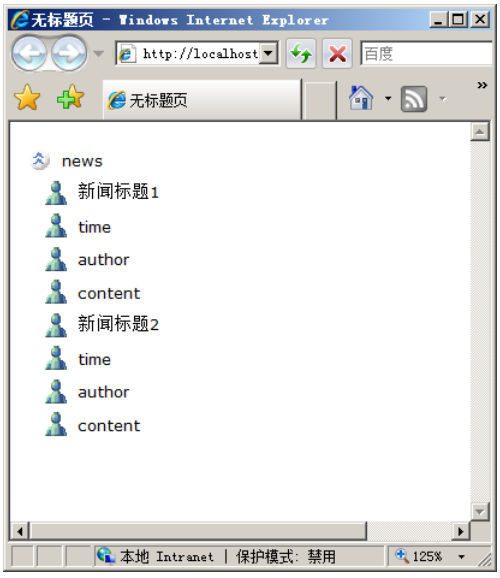


图 8-25　XmlDataSource 数据绑定

XmlDataSource 控件一般用于只读的数据方案。数据绑定控件显示 XML 数据，还可以通过 XmlDataSource 来编辑 XML 数据。但是当 XmlDataSource 控件加载时，必须使用 DataFile 属性加载，而不能从 Data 属性中指定的 XML 的字符串进行加载。

8.1.6　站点导航控件（SiteMapDataSource）

为了引导用户在站点的各个页面能够流畅跳转，需要在每个页面加入页面导航。在 ASP 的开发过程中，必须手动的为每个页面加入导航，这样不仅加大了开发的复杂度，也让代码的复用性变低。相对于手动加入导航更好的解决方法则是使用 js 在各个页面引用导航，但是一旦页面变得很多，可能会导致让 js 页面效率变低。而在 ASP.ENT 2.0 以后的版本，微软提供了导航控件让导航菜单的创建、自定义和维护变得更加的简单。

SiteMapDataSource 控件包含来自站点地图的导航数据，这些数据包括有关网站中的页的信息，例如网站页面的标题、说明信息以及 URL 等。如果将导航数据存储在一个地方，则可以方便的在网站的导航菜单添加和删除项。站点地图提供程序中检索导航数据，然后将数据传递给可显示该数据的数据绑定

控件，显示导航菜单。

如果需要使用 SiteMapDataSource 控件，用户必须在 Web.sitemap 文件中描述站点的结构，示例代码如下所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title="根目录"  description="根目录">
        <siteMapNode url="SqlDataSource.aspx" title="SqlDataSource.aspx"  description="SQL 数据库" />
        <siteMapNode url="AccessDataSource" title="AccessDataSource"  description="Access 数据库" />
        <siteMapNode url="LinqDataSource" title="LinqDataSource"  description="Linq" />
        <siteMapNode url="ObjectDataSource" title="ObjectDataSource"  description="Object" />
        <siteMapNode url="XmlDataSource" title="XmlDataSource"  description="Xml" />
    </siteMapNode>
</siteMap>
```

上述代码描述了网站的目录结构，在文件中，必须有一个根为 siteMapNode 的元素作为 siteMap 元素的自己，并定义以下常用属性：

- title：为站点地图节点指定一个标题，该标题将显示为网页的连接文本。
- Url：为网页指定 URL。支持相对或绝对路径。
- Description：为站点地图的节点添加描述，当用户鼠标移动到该栏目时，则会显示描述信息。
- StartFormCurrentNode：当设置为 true 时，则可以从该节点开始检索站点地图结构。
- StartingNodeOffset：当属性设置为 2 时可以检索当前地图结构。

SiteMapDataSource 控件无需配置，拖放一个 TreeView 控件和一个 SiteMapDataSource 控件在页面，指定 TreeView 数据源即可，如图 8-26 所示。

图 8-26　配置数据源

配置完成后，数据绑定控件会自动读取 Web.sitmap 文件并生成导航。当使用了 SiteMapDataSource 控件后，数据绑定控件就能够绑定 SiteMapDataSource 控件并自动读取相应的值并生成导航，当需要对导航进行修改时，只需要修改 Web.sitemap 即可，方便了站点导航功能的使用和维护。运行后如图 8-27 所示。

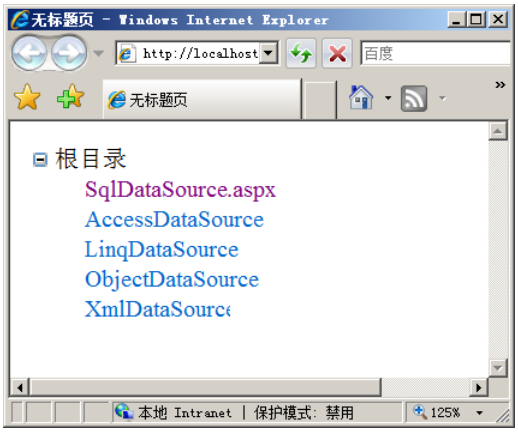


图 8-27　SiteMapDataSource 控件数据显示

8.2　重复列表控件（Repeater）

重复列表控件（Repeater）是一个可重复操作的控件。它能够通过使用模板显示一个数据源的内容，而且开发人员可以轻松的配置这些模板，Repeater 控件包括如标题和页脚这样的数据，它可以遍历所有的数据选项并将其应用到模板中。

重复列表控件并不是从 WebControl 派生出来，重复列表控件可以直接操控 HTML 文件或者样式表来编写模板和控制属性。重复列表控件支持 5 种模板，用来显示相应的界面信息，这 5 种模板的功能如下所示：

- AlternatingItemTemplate：指定如何显示其他选项。
- ItemTemplate：指定如何显示选项。
- HeaderTemplate：建立如何显示标题。
- FooterTemplate：建立如何显示页脚。
- SeparatorTemplate：指定如何显示不同选项之间的分隔符。

在上面 5 种模板中，惟一需要使用的是 ItemTemplate 模板，其他的模板可以选用。示例代码如下所示。

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
    <ItemTemplate>
        <%# Eval("title")%>
    </ItemTemplate>
</asp:Repeater>
```

“<%#%>”符号之间的语句表示数据绑定表达式，可以直接使用数据源控件中查询出来字段。在 Repeaater 中间，使用 ItemTemplate 制作模板，在 ItemTemplate 模板中可以直接使用 HMTL 制作样式。在数据显示中，可以直接使用“<%#%>”绑定数据库中的列，例如当数据源控件中查询了一个 title 列时，则在 Repeater 控件中直接使用“<%#Eval(“title”)%>”方式显式 title 字段的值。

显示字段有几种方法，其中“<%#Eval(“字段名称”)%>”是最方便的显示字段的方法，能够方便的在模板中嵌入，其他方法还有使用“<%#DataBlinder.Eval(Container.DataItem,“字段名称”)%>”方式来绑定相关的列。示例代码如下所示。

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
    <ItemTemplate>
```

```
            <div style="border-bottom:1px dashed #ccc; padding:5px 5px 5px 5px;">
                <%# Eval("title")%>
             </div>
        </ItemTemplate>
    </asp:Repeater>
```

上述代码自定义了一个HTML代码，增加了一个DIV标签，该标签设置了CSS属性border-bottom:1px dashed #ccc; padding:5px 5px 5px 5px;。Repeater 控件能够自动的重复该模板。当数据库中的数据完毕后，则不再重复，运行结果如图 8-28 所示。

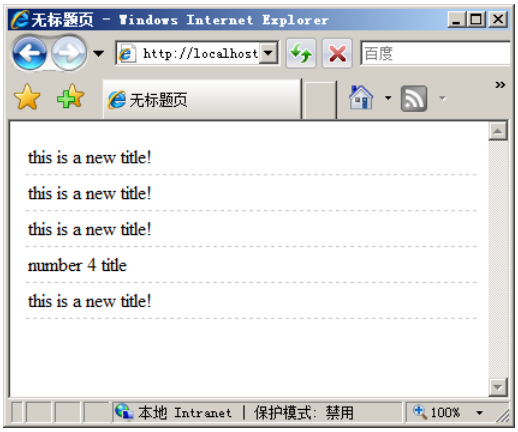


图 8-28　Repeater 控件

重复列表控件最常用的事件有 ItemCommand、ItemCreated、ItemDataBound。当创建一个项或者一个项被绑定到数据源时，将触发 ItemCreated 和 ItemDataBound 事件。当重复列表控件中有按钮被激发时，会触发 ItemCommand 事件。

在 ItemCommand 中，为了自定义按钮控件相应事件，开发人员必须指定 RepeaterCommandEventArgs 参数获取 CommandArgument、CommandName 和 CommandSource 三个属性对应的值，示例代码如下所示。

```
    <asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1"
    onitemcommand="Repeater1_ItemCommand">
        <ItemTemplate>
            <div style="border-bottom:1px dashed #ccc; padding:5px 5px 5px 5px;">
                <%# Eval("title")%>
                <asp:Button ID="Button1" runat="server" Text="按钮"
                CommandArgument='<%# Eval("title")%>'/>
             </div>
        </ItemTemplate>
    </asp:Repeater>
```

上述代码增加了一个按钮控件，并配置按钮控件的命令参数为数据库中的 title 的值。当单击按钮控件时，则会触发 ItemCommand，示例代码如下所示。

```
        protected void Repeater1_ItemCommand(object source, RepeaterCommandEventArgs e)
        {
            Label1.Text = "用户选择了" + e.CommandArgument.ToString();                    //显式选择项
        }
```

上述代码当指定了执行按钮控件触发的事件，运行结果如图 8-29 和图 8-30 所示。

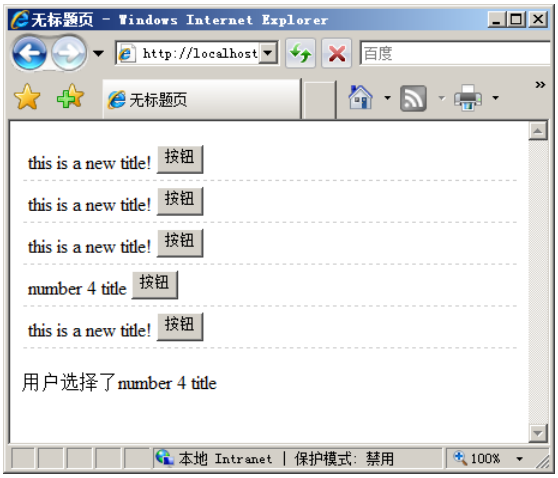


图 8-29　ItemCommand 事件

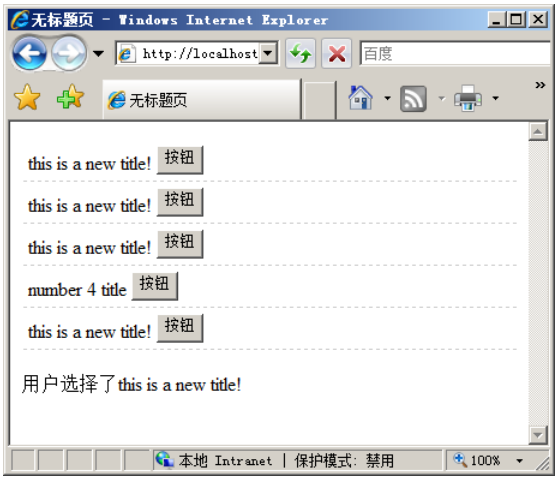


图 8-30　用户选择单击后

Repeat 控件需要一定的 HTML 知识才能显示数据库的相应信息，虽然增加了一定的复杂度，但是却增加了灵活性。Repeat 控件能够按照用户的想法显示不同的样式，让数据显示更加丰富。

8.3　数据列表控件（DataList）

DataList 控件支持各种不同的模板的样式，通过为 DataList 指定不同的样式，可以自定义 DataList 控件的外观。与 Repeater 控件相同的是，DataList 控件同样也支持自定义 HTML，但是 DataList 控件具备 Repeater 控件不具有的特性，DataList 控件常用属性如下所示。

- AltermatingItemStyle：编写交替行的样式。
- EditItemStyle：正在编辑的项的样式。
- FooterStyle：列表结尾处的脚注的样式。
- HeaderStyle：列表头部的标头的样式。
- ItemStyle：单个项的样式。
- SelectedItemStyle：选定项的样式。
- SeparatorStyle：各项之间分隔符的样式。

通过修改 DataList 控件的相应的属性，能够实现复杂的 HTML 样式而不需要通过变成实现。而 DataList 控件能够套用自定义格式实现更多的效果，如图 8-31 所示。

通过属性生成器，同样可以通过勾选相应的项目来生成属性，这些属性能够极大的方便开发人员制作 DataList 控件的界面样式，如图 8-32 所示。

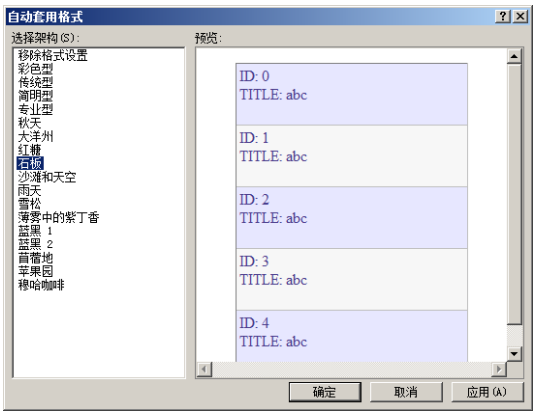


图 8-31　自动套用格式

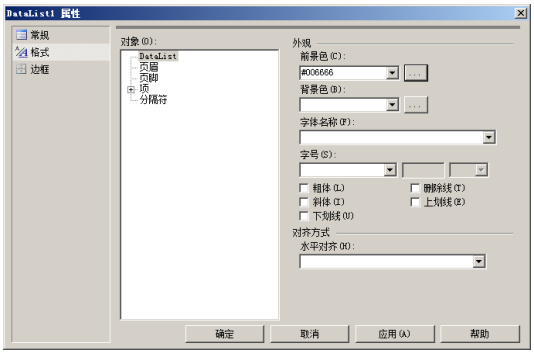


图 8-32　属性生成器

DataList 控件经常在开发中使用，DataList 控件不仅能够支持 Repeater 控件中的 ItemCommand、ItemCreated、ItemDataBound 事件，还支持更多的服务器事件。对项中的按钮进行操作，如果按钮的 CommandName 属性为“edit”，则该按钮则可以引发 EditorCommand 事件，同样也可以配置不同的 CommandName 属性来实现不同的操作。编辑 DataList 控件，并编辑相应的 HTML 代码，让 DataList 控件包括按钮，并为按钮配置相应的 CommandName 属性，示例代码如下所示。

```
    <asp:DataList ID="DataList1" runat="server" BackColor="White"
    BorderColor="#E7E7FF" BorderStyle="None" BorderWidth="1px" CellPadding="3"
    DataKeyField="ID" DataSourceID="SqlDataSource1" Font-Bold="False"
    Font-Italic="False" Font-Overline="False" Font-Strikeout="False"
    Font-Underline="False" GridLines="Horizontal" Width="100%"
    ondeletecommand="DataList1_DeleteCommand">
        <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
        <AlternatingItemStyle BackColor="#F7F7F7" />
        <ItemStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
        <SelectedItemStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
    <ItemTemplate>
        新闻 ID:
        <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' />
        <br />
        新闻编号:
        <asp:Label ID="TITLELabel" runat="server" Text='<%# Eval("TITLE") %>' />
        <br />
        <asp:Button ID="Button1" runat="server" Text="删除"
        CommandName="delete" CommandArgument='<%# Eval("ID") %>'/>
    </ItemTemplate>
</asp:DataList>
```

上述代码创建了一个 DataList 控件并配置了按钮控件，并将按钮控件的 CommandName 属性配置为“delete”，则触发该按钮则会引发 DeleteCommand 事件。在属性窗口中找到 DeleteCommand 事件，双击【DeleteCommand】连接系统会自动生成 DeleteCommand 事件相应的方法。当生成了 DeleteCommand 事件后，可以在代码段中编写相应的方法，示例代码如下所示。

```
        protected void DataList1_DeleteCommand(object source, DataListCommandEventArgs e)
        {
            Label1.Text = e. CommandArgument.ToString()+"被执行";
```

```
        }
```

当用户单击了相应的按钮时会触发 DeleteCommand 事件。开发人员能够通过传递过来的参数，可以编写相应的方法，运行结果如图 8-33 所示。

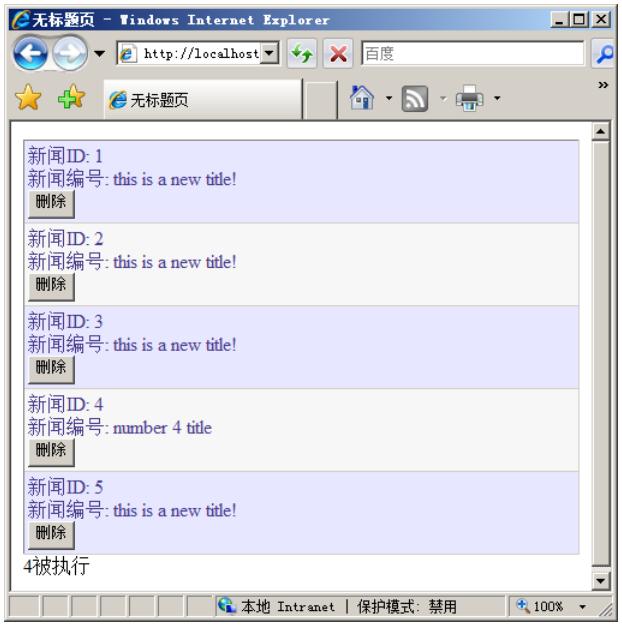


图 8-33　触发 DeleteCommand 事件

程序运行后，当用户单击了相应的按钮时，开发人员可以通过获取传递的 CommandArgument 参数的值来编写相应的方法从而执行实现不同的应用。

8.4　数据列表控件（GridView）

GridView 是 ASP.NET 中功能非常丰富的控件之一，它可以以表格的形式显示数据库的内容并通过数据源控件自动绑定和显示数据。开发人员能够通过配置数据源控件对 GridView 中的数据进行选择、排序、分页、编辑和删除功能进行配置。GridView 控件还能够指定自定义样式，在没有任何数据时可以自定义无数据时的 UI 样式。

1．建立 GridView 控件

GridView 控件为开发人员提供了强大的管理方案，同样 GridView 也支持内置格式，单击【自动套用格式】连接可以选择 GridView 中的默认格式，如图 8-34 所示。

GridView 是以表格为表现形式，GridView 包括行和列，通过配置相应的属性能够编辑相应的行的样式，同样也可以选择【编辑列】选项来编写相应的列的样式，如图 8-35 所示。

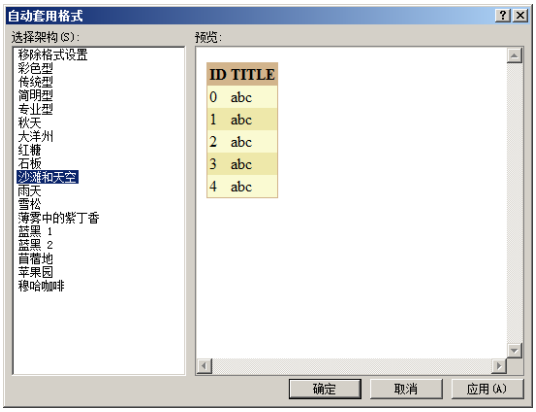


图 8-34　自动套用格式

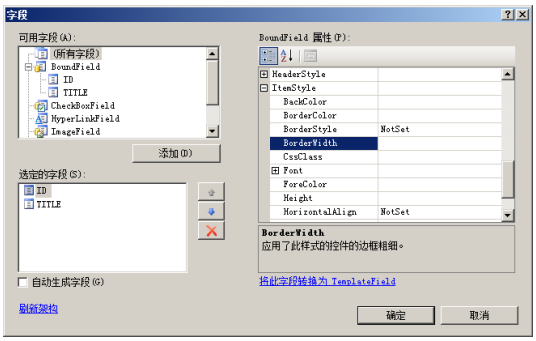


图 8-35　编辑列

GridView 控件提供两个用户绑定到数据的选项，其一是使用 DataSourceID 进行数据绑定，这种方法通常情况下是绑定数据源控件；而另一种则是使用 DataSource 属性进行数据绑定，这种方法能够将 GridView 控件绑定到包括 ADO.NET 数据和数据读取器内的各种对象。

使用 DataSourceID 进行数据绑定，可以让 GridView 控件能够自动的处理分页、选择等操作，如图 8-36 所示。而使用 DataSource 属性进行数据绑定，则需要开发人员通过编程实现分页等操作。GridView 控件能够自定义字段，单击【添加列】按钮，可以选择相应类型的列。在添加列选项中，GridView 控件支持多种列类型的列，包括复选框、图片、单选框、超链接等，如图 8-37 所示。

图 8-36　可选相应操作

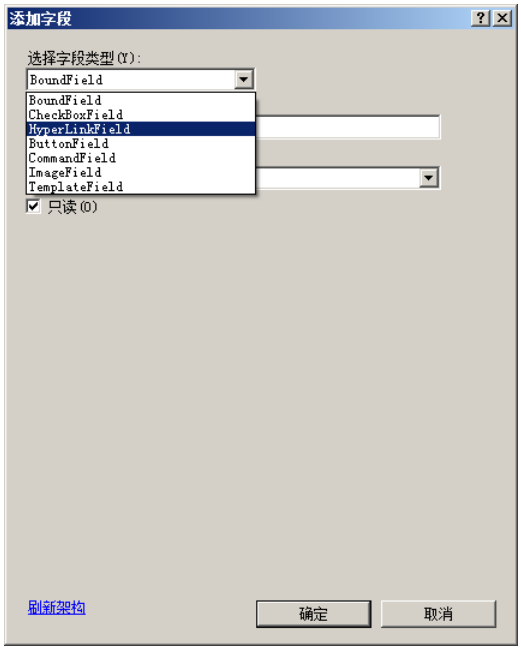


图 8-37　添加字段

添加自定义字段，GridView 控件支持从数据源中读取相应的数据源来配置相应的字段，来让开发人员自定义的读取数据源中的相应字段来自定义开发，如图 8-38 所示。当选择从数据源中获取文本，可以通过 Format 的形式编写相应的文本。例如，从数据源中获取 title 列，而显示文本为“这是一个标题：title 值”，则可以编写为“这是一个标题：{0}”，如图 8-39 所示。

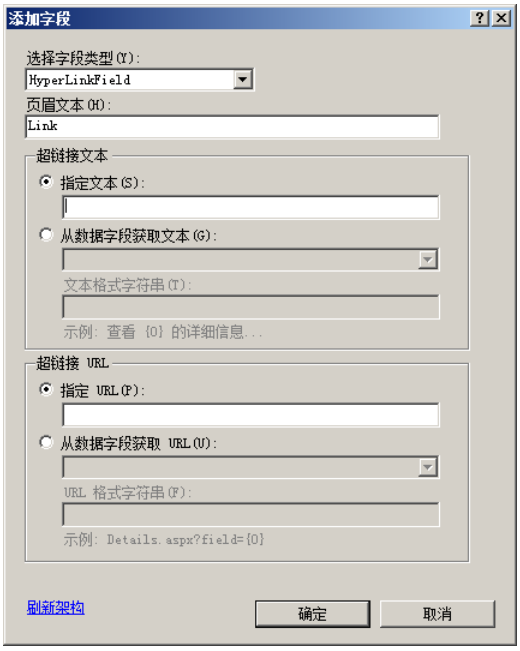


图 8-38　添加字段

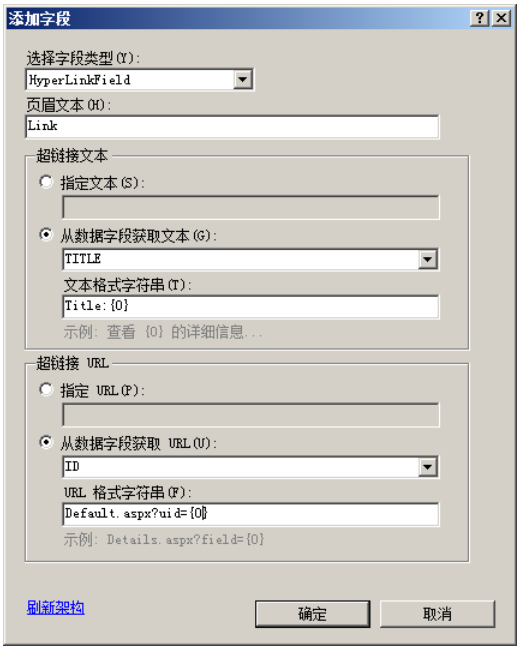


图 8-39　格式化字符串输出

配置完成后，GridView 控件的 HTML 标签生成代码如下所示：

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True"
    AllowSorting="True" AutoGenerateColumns="False"
    BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
    CellPadding="2" DataKeyNames="ID" DataSourceID="SqlDataSource1"
    ForeColor="Black" GridLines="None" Width="100%">
        <FooterStyle BackColor="Tan" />
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False"
                    ReadOnly="True" SortExpression="ID" />
                <asp:BoundField DataField="TITLE" HeaderText="TITLE" SortExpression="TITLE" />
                <asp:HyperLinkField DataNavigateUrlFields="ID"
                    DataNavigateUrlFormatString="Default.aspx?uid={0}" DataTextField="TITLE"
                    DataTextFormatString="Title:{0}" HeaderText="Link" />
            </Columns>
        <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
          HorizontalAlign="Center" />
        <SelectedRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
        <HeaderStyle BackColor="Tan" Font-Bold="True" />
    <AlternatingRowStyle BackColor="PaleGoldenrod" />
</asp:GridView>
```

上述代码使用了一个默认格式，并新建了一个超链接文本类型的列，当单击超文本链接，则会跳转到另一个页面。

2. GridView 控件的常用事件

GridView 支持多个事件，通常对 GridView 控件进行排序、选择等操作时，同样会引发事件，当创建当前行或将当前行绑定至数据时发生的事件，同样，单击一个命令控件时也会引发事件。GridView 控件常用的事件如下所示。

- RowCommand：在 GridView 控件中单击某个按钮时发生。此事件通常用于在该控件中单击某个按钮时执行某项任务。
- PageIndexChanging：在单击页导航按钮时发生，但在 GridView 控件执行分页操作之前。此事件通常用于取消分页操作。
- PageIndexChanged：在单击页导航按钮时发生，但在 GridView 控件执行分页操作之后。此事件通常用于在用户定位到该控件中不同的页之后需要执行某项任务时。
- SelectedIndexChanging：在单击 GridView 控件内某一行的 Select 按钮（其 CommandName 属性设置为“Select”的按钮）时发生，但在 GridView 控件执行选择操作之前。此事件通常用于取消选择操作。
- SelectedIndexChanged：在单击 GridView 控件内某一行的 Select 按钮时发生，但在 GridView 控件执行选择操作之后。此事件通常用于在选择了该控件中的某行后执行某项任务。
- Sorting：在单击某个用于对列进行排序的超链接时发生，但在 GridView 控件执行排序操作之前。此事件通常用于取消排序操作或执行自定义的排序例程。
- Sorted：在单击某个用于对列进行排序的超链接时发生，但在 GridView 控件执行排序操作之后。此事件通常用于在用户单击对列进行排序的超链接之后执行某项任务。
- RowDataBound：在 GridView 控件中的某个行被绑定到一个数据记录时发生。此事件通常用于在某个行被绑定到数据时修改该行的内容。
- RowCreated：在 GridView 控件中创建新行时发生。此事件通常用于在创建某个行时修改该行的布局或外观。
- RowDeleting：在单击 GridView 控件内某一行的 Delete 按钮（其 CommandName 属性设置为“Delete”的按钮）时发生，但在 GridView 控件从数据源删除记录之前。此事件通常用于取消删除操作。
- RowDeleted：在单击 GridView 控件内某一行的 Delete 按钮时发生，但在 GridView 控件从数据源删除记录之后。此事件通常用于检查删除操作的结果。
- RowEditing：在单击 GridView 控件内某一行的 Edit 按钮（其 CommandName 属性设置为“Edit”的按钮）时发生，但在 GridView 控件进入编辑模式之前。此事件通常用于取消编辑操作。
- RowCancelingEdit：在单击 GridView 控件内某一行的 Cancel 按钮（其 CommandName 属性设置为“Cancel”的按钮）时发生，但在 GridView 控件退出编辑模式之前。此事件通常用于停止取消操作。
- RowUpdating：在单击 GridView 控件内某一行的 Update 按钮（其 CommandName 属性设置为“Update”的按钮）时发生，但在 GridView 控件更新记录之前。此事件通常用于取消更新操作。
- RowUpdated：在单击 GridView 控件内某一行的 Update 按钮时发生，但在 GridView 控件更新记录之后。此事件通常用来检查更新操作的结果。
- DataBound：此事件继承自 BaseDataBoundControl 控件，在 GridView 控件完成到数据源的绑定后发生。

需要指定相应的事件，则必须添加一个 RowCommand 事件，GridView 控件 HTML 代码如下所示.

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True"
 AllowSorting="True" AutoGenerateColumns="False"
 BackColor="LightGoldenrodYellow" BorderColor="Tan" BorderWidth="1px"
 CellPadding="2" DataKeyNames="ID" DataSourceID="SqlDataSource1"
```

```
    ForeColor="Black" GridLines="None" onrowcommand="GridView1_RowCommand"
    Width="100%">
        <FooterStyle BackColor="Tan" />
            <Columns>
                <asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False"
                ReadOnly="True" SortExpression="ID" />
                <asp:BoundField DataField="TITLE" HeaderText="TITLE" SortExpression="TITLE" />
                <asp:HyperLinkField DataNavigateUrlFields="ID"
                DataNavigateUrlFormatString="Default.aspx?uid={0}" DataTextField="TITLE"
                DataTextFormatString="Title:{0}" HeaderText="Link" />
                    <asp:ButtonField ButtonType="Button" CommandName="
                    Select" HeaderText="选择按钮" ShowHeader="True" Text="按钮" />
            </Columns>
        <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
         HorizontalAlign="Center" />
        <SelectedRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
        <HeaderStyle BackColor="Tan" Font-Bold="True" />
    <AlternatingRowStyle BackColor="PaleGoldenrod" />
    </asp:GridView>
```

上述代码创建了一个 GridView 控件，并增加了一个按钮控件，并且为按钮控件的 CommandName 属性赋值为 Select，当单击按钮控件时，则会触发 RowCommand 事件，CS 页面代码如下所示。

```
        protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            Label1.Text = e.CommandName + "事件被触发";
        }
```

当单击按钮时，GridView 控件会选择相应的行。在 GridView 控件的 RowCommand 事件中，同样可以通过 GridView 控件的中按钮的 CommandArgument 属性获取相应的操作并执行相应代码。GridView 控件运行结果如图 8-40 和图 8-41 所示。

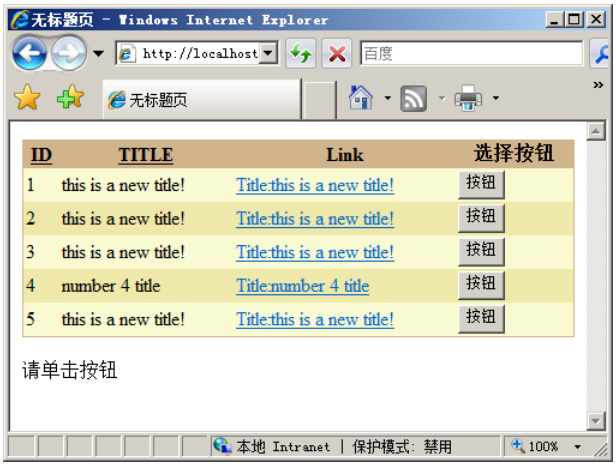


图 8-40　GridView 控件的事件

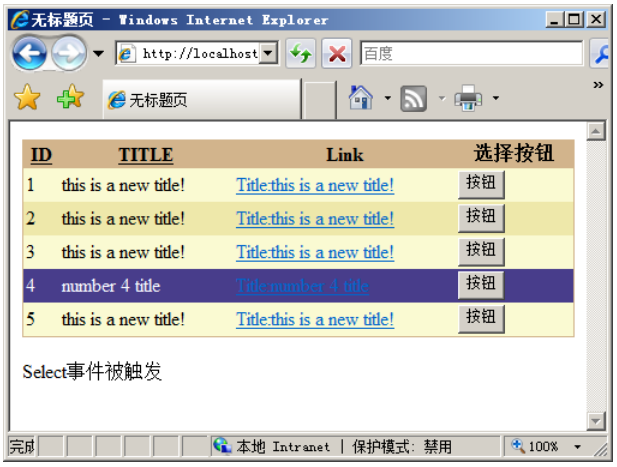


图 8-41　触发 Select 选择事件

注意：在执行其他事件时，如 RowDeleted、GridView 控件首先执行 RowDataBound 代码，然后执行 RowCommnad、RowDeleting 以及 RowDeleted 等事件。

8.5 数据绑定控件（FormView）

FormView 控件只能显示数据库中一行的数据，并且提供对数据的分页操作，FormView 控件可以以一种不规则的外观来将数据呈现给用户。FormView 控件同样支持模板，以方便开发人员自定义 FormView 控件的 UI，FormView 控件支持的模板如下所示：

- ItemTemplate：用于在 FormView 种呈现一个特殊的记录。
- HeaderTemplate：用于指定一个可选的页眉行。
- FooterTemplate：用于指定一个可选的页脚行。
- EmptyDataTemplate：当 FormView 的 DataSource 缺少记录的时候，EmptyDataTemplate 将会代替 ItemTemplate 来生成控件的标记语言。
- PagerTemplate：如果 FormView 启用了分页的话，这个模板可以用于自定义分页的界面。
- EditItemTemplate / InsertItemTemplate：如果 FormView 支持编辑或插入功能，那么这两种模板可以用于自定义相关的界面。

通过编辑 ItmTemplate，能够自定义 HTML 以呈现数据，这种情况很像 Repeater 控件。FormView 控件同样支持自动套用格式，选择【自动套用格式】选项就能够为 FormView 控件选择默认格式，选择后如图 8-42 所示。

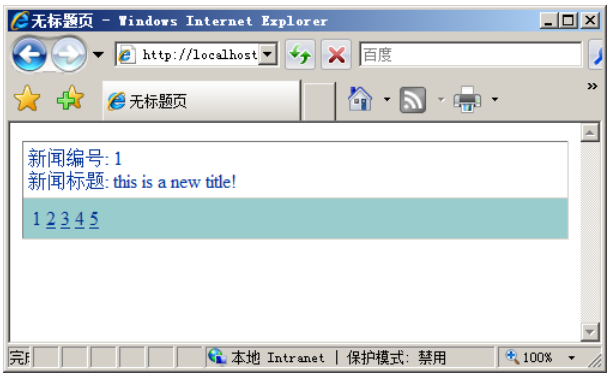


图 8-42　自定义 FormView 控件

当 FormView 控件界面编写完成后，HTML 代码如下所示。

```
<asp:FormView ID="FormView1" runat="server" AllowPaging="True"
 BackColor="White" BorderColor="#3366CC" BorderStyle="None" BorderWidth="1px"
 CellPadding="4" DataKeyNames="ID" DataSourceID="SqlDataSource1"
 GridLines="Both" Width="100%">
    <FooterStyle BackColor="#99CCCC" ForeColor="#003399" />
        <RowStyle BackColor="White" ForeColor="#003399" />
            <EditItemTemplate>
                ID:
                  <asp:Label ID="IDLabel1" runat="server" Text='<%# Eval("ID") %>' /><br />
                TITLE:
                  <asp:TextBox ID="TITLETextBox" runat="server" Text='<%# Bind("TITLE") %>' /><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>
```

```
            <InsertItemTemplate>
                TITLE:
                  <asp:TextBox ID="TITLETextBox" runat="server" Text='<%# Bind("TITLE") %>' /> <br />
                  <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>
            <ItemTemplate>
                新闻编号:
                  <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' /><br />
                新闻标题:
                <asp:Label ID="TITLELabel" runat="server" Text='<%# Bind("TITLE") %>' /><br />
            </ItemTemplate>
        <PagerStyle BackColor="#99CCCC" ForeColor="#003399" HorizontalAlign="Left" />
        <HeaderStyle BackColor="#003399" Font-Bold="True" ForeColor="#CCCCFF" />
        <EditRowStyle BackColor="#009999" Font-Bold="True" ForeColor="#CCFF99" />
    </asp:FormView>
```

上述代码创建了 FormView 控件，并为 FormView 控件自定义了若干模板。刚才只是编写了 ItemTemplate 模板，但是 EdititemTemplate 也已经在 HTML 标签中生成。

注意：FormView **控件模板中的相应数据字段也是通过数据绑定语法实现的**，如<%# Eval("**字段名称**") %>。

FormView 控件同样支持对当前数据的更新、删除、选择等操作。当拖放一个按钮控件时，可以选择 DataBindings 来为按钮控件的属性做相应的配置，如图 8-43 所示。

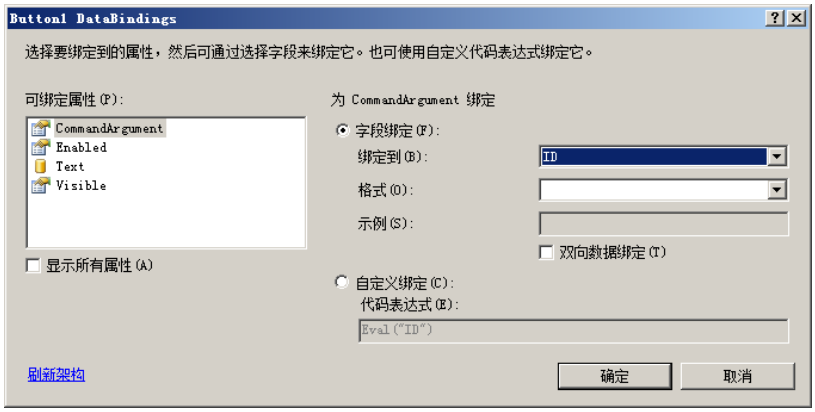


图 8-43　DataBindings

当单击 FormView 中的控件时，会触发 Command 事件，要使用 FormView 控件进行更新等操作，必须在相应的模式下更新才行，例如当需要更新操作时，则必须在编辑模式下才能进行更新操作。当执行相应的操作时，例如更新操作，则必须在编辑模式下进行操作，并需要使用 ItemUpdated 事件来编写相应的更新事件。编写 FormView 控件中的 ItemTemplate 和 EditItemTemplate，生成的 HTML 代码如下所示。

```
    <asp:FormView ID="FormView1" runat="server" AllowPaging="True"
        BackColor="White" BorderColor="#3366CC" BorderStyle="None" BorderWidth="1px"
        CellPadding="4" DataKeyNames="ID" DataSourceID="SqlDataSource1"
        GridLines="Both" Width="100%" onitemcommand="FormView1_ItemCommand"
```

```
        onitemupdated="FormView1_ItemUpdated">
            <FooterStyle BackColor="#99CCCC" ForeColor="#003399" />
            <RowStyle BackColor="White" ForeColor="#003399" />
            <EditItemTemplate>
                新闻编号:
                <asp:Label ID="IDLabel1" runat="server" Text='<%# Eval("ID") %>' />
                <br />
                新闻标题:
                <asp:TextBox ID="TITLETextBox" runat="server" Text='<%# Bind("TITLE") %>' />
                <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>
            <InsertItemTemplate>
                TITLE:
                <asp:TextBox ID="TITLETextBox" runat="server" Text='<%# Bind("TITLE") %>' />
                <br />
                <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>
            <ItemTemplate>
                新闻编号:
                    <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' /><br />
                新闻标题:
                    <asp:Label ID="TITLELabel" runat="server" Text='<%# Bind("TITLE") %>' /><br />
            </ItemTemplate>
        <PagerStyle BackColor="#99CCCC" ForeColor="#003399" HorizontalAlign="Left" />
        <HeaderStyle BackColor="#003399" Font-Bold="True" ForeColor="#CCCCFF" />
        <EditRowStyle BackColor="#009999" Font-Bold="True" ForeColor="#CCFF99" />
    </asp:FormView>
```

上述代码编写了 FormView 控件中的 ItemTemplate 和 EditItemTemplate。在页面中，增加了按钮来切换 FormView 控件的编辑模式，按钮控件代码如下所示。

```
        <asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="Edit" />
```

当单击按钮时，FormView 控件会更改其编辑模式，示例代码如下所示。

```
        protected void Button2_Click(object sender, EventArgs e)
        {
            FormView1.ChangeMode(FormViewMode.Edit);                    //更改编辑模式
        }
```

当更改了编辑模式后，FormView 控件允许在当前页面直接更改数据的值，并通过 ItemUpdated 进行更新，示例代码如下所示。

```
        protected void FormView1_ItemUpdated(object sender, FormViewUpdatedEventArgs e)
        {
            Label1.Text = "相应值被更新";                                //提示已被更改
            FormView1.ChangeMode(FormViewMode.ReadOnly);                //更改编辑模式
        }
```

上述代码允许开发人员能够自定义数据操作，通过对象 e 的值来获取相应的数据字段的值并进行更新，运行结果如图 8-44 和 8-45 所示。

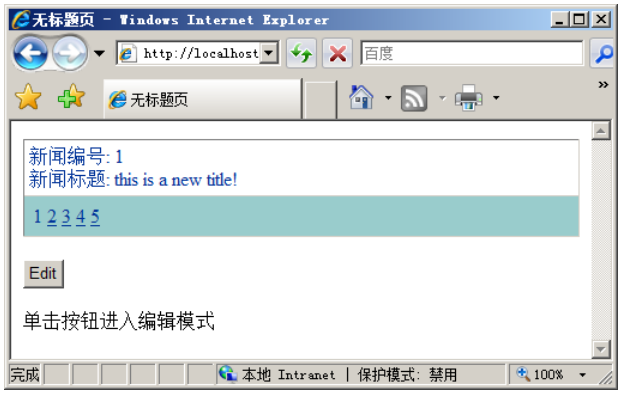


图 8-44　视图模式

图 8-45　编辑模式

当单击了其中的更新，则会触发 ItemUpdated 事件，开发人员能够通过编写 ItemUpdated 事件来进行相应的更新操作。值得注意的是，通常情况下数据源控件必须支持更新操作才能够执行更新，在配置数据源时，需要为更新语句进行配置。在配置和生成 SQL 语句中必须选择【高级】选项、勾选【生成 update、insert、delete 语句】复选框才能够让数据源控件支持更新等操作，如图 8-46 所示。

图 8-46　高级数据源配置

如果数据绑定控件需要使用 Insert 等语句时，则数据源控件需配置高级 SQL 生成选项，开发人员还能够在数据源控件的 HTML 代码中进行相应的 SQL 语句的更改已达到自定义数据源控件的目的。

8.6　数据绑定控件（DetailsView）

DetailsView 控件与 FormView 在很多情况下非常类似，DetailsView 控件通常情况下也只能够显示一行的数据，同 FormView，DetailsView 控件支持对数据源控件中的数据进行插入、删除和更新。但是 DetailsView 控件与 FormView 控件不同的是，DetailsView 控件不支持 ItemTemplate 模板，这也就是说，DetailsView 控件是以一种表格的形式所呈现的。

相比之下，DetailsView 控件能够支持 Ajax，因为 FormView 控件完全由模板驱动，但是 FormView 控件对验证控件的支持较好。而 DetailsView 控件可以通过选择是否包括更新，删除等操作，而无需手动的添加相应的事件，比 FormView 控件更加方便，如图 8-47 和图 8-48 所示。

图 8-47　配置 DetailsView 任务

图 8-48　减少任务配置

当选择了【启用分页】选项后 DetailsView 控件就能够自动进行分页。开发人员还可以配置 PagerSettings 属性允许自定义 DetailsView 控件生成分页用户界面的外观，它将呈现向前和向后导航的方向控件，PagerSettings 属性的常用模式有：

- NextPrevious：以前一个，下一个形式显示。
- NextPreviousFirstLast：以前一个，下一个，最前一个，最后一个形式显示。
- Numeric：以数字形式显示。
- NumericFirstLast::：以数字，最前一个，最后一个形式显示。

当完成配置 DetailsView 控件后，DetailsView 控件无需通过外部控件来转换 DetailsView 控件的编辑模式，DetailsView 控件自动会显示更新、插入、删除等按钮来更改编辑模式，如图 8-49 所示。

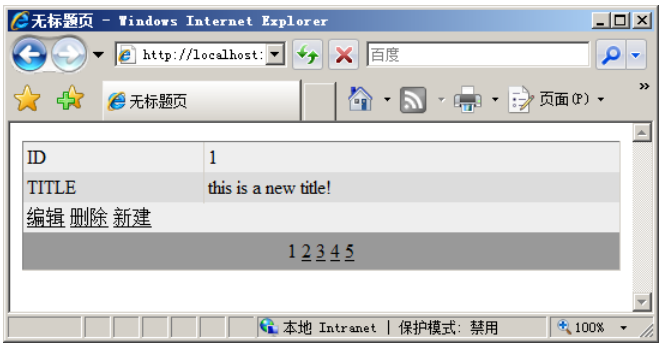


图 8-49　DetailsView 控件

编辑完成后，DetailsView 控件生成的 HTML 代码如下所示。

```
<asp:DetailsView ID="DetailsView1" runat="server" AllowPaging="True"
        AutoGenerateRows="False" BackColor="White" BorderColor="#999999"
        BorderStyle="None" BorderWidth="1px" CellPadding="3" DataKeyNames="ID"
        DataSourceID="SqlDataSource1" GridLines="Vertical" Height="50px" Width="100%">
    <FooterStyle BackColor="#CCCCCC" ForeColor="Black" />
    <RowStyle BackColor="#EEEEEE" ForeColor="Black" />
        <PagerStyle BackColor="#999999" ForeColor="Black" HorizontalAlign="Center" />
            <Fields>
                <asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False"
                ReadOnly="True" SortExpression="ID" />
                <asp:BoundField DataField="TITLE" HeaderText="TITLE" SortExpression="TITLE" />
                <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
                ShowInsertButton="True" />
            </Fields>
        <HeaderStyle BackColor="#000084" Font-Bold="True" ForeColor="White" />
```

```
            <EditRowStyle BackColor="#008A8C" Font-Bold="True" ForeColor="White" />
        <AlternatingRowStyle BackColor="#DCDCDC" />
    </asp:DetailsView>
```

如上一节内容所讲，在数据源控件的配置中配置 SQL 语句，需要选择高级，勾选【生成 update、insert、delete 语句】复选框以支持自动生成更新、删除等语句的生成。当勾选了【生成 update、insert、delete 语句】复选框后，数据源控件代码如下所示。

```
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
            ConnectionString="<%$ ConnectionStrings:mytableConnectionString %>"
            DeleteCommand="DELETE FROM [mynews] WHERE [ID] = @ID"
            InsertCommand="INSERT INTO [mynews] ([TITLE]) VALUES (@TITLE)"
            SelectCommand="SELECT * FROM [mynews]"
            UpdateCommand="UPDATE [mynews] SET [TITLE] = @TITLE WHERE [ID] = @ID">
        <DeleteParameters>
            <asp:Parameter Name="ID" Type="Int32" />
        </DeleteParameters>
        <UpdateParameters>
            <asp:Parameter Name="TITLE" Type="String" />
            <asp:Parameter Name="ID" Type="Int32" />
        </UpdateParameters>
            <InsertParameters>
            <asp:Parameter Name="TITLE" Type="String" />
        </InsertParameters>
    </asp:SqlDataSource>
```

从上述代码可以看出，数据源控件自动生成了相应的 SQL 语句，如图 8-50 所示。当执行更新、删除等操作时，则会默认执行该语句。运行结果如图 8-51 所示。

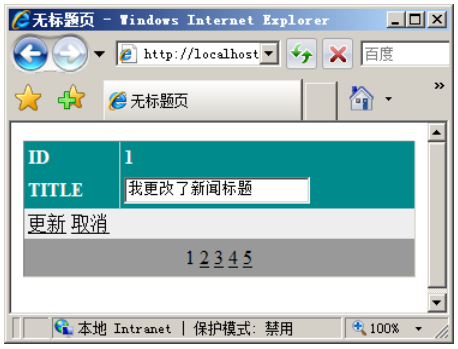


图 8-50　更改相应字段的值

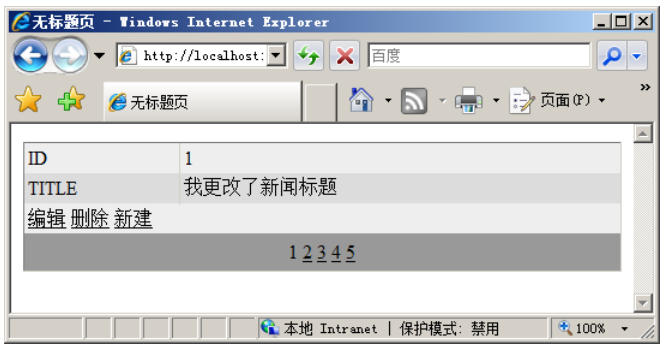


图 8-51　更改后的控件呈现

8.7　数据绑定控件（ListView）

ListView 控件是 ASP.NET 3.5 中新增的数据绑定控件，ListView 控件是介于 GridView 控件和 Repeater 之间的另一种数据绑定控件，相对于 GridView 来说，它有着更为丰富的布局手段，开发人员可以在 ListView 控件的模板内写任何 HTML 标记或者控件。相比于 GridView 和 Repeater 控件而言，ListView 支持的模板如下所示：

- AlternatingItemTemplate：交替项目模板，用不同的标记显示交替的项目，便于查看者区别连续不断的项目。

- EditItemTemplate：编辑项目模板，控制编辑时的项目显示。
- EmptyDataTemplate：空数据模板，控制 ListView 数据源返回空数据时的显示。
- EmptyItemTemplate：空项目模板，控制空项目的显示。
- GroupSeparatorTemplate：组分隔模板，控制项目组内容的显示。
- GroupTemplate：组模板，为内容指定一个容器对象，如一个表行、div 或 span 组件。
- InsertItemTemplate：插入项目模板，用户插入项目时为其指定内容。
- ItemSeparatorTemplate：项目分隔模板，控制项目之间内容的显示。
- ItemTemplate 项目模板：控制项目内容的显示。
- LayoutTemplate：布局模板，指定定义容器对象的根组件，如一个 table、div 或 span 组件，它们包装 ItemTemplate 或 GroupTemplate 定义的内容。
- SelectedItemTemplate：已选择项目模板，指定当前选中的项目内容的显示。

其中最为常用的控件包括 LayoutTemplate 和 ItemTemplate，LayoutTemplate 为 ListView 控件指定了总的标记，而 ItemTemplate 指定的标记用于显示每个绑定的记录，用来编写 HTML 样式。ListView 控件能够自动套用 HTML 格式，如其他控件一样，可以选择默认模板，单击【配置 ListView】连接进行格式套用，如图 8-52 所示。

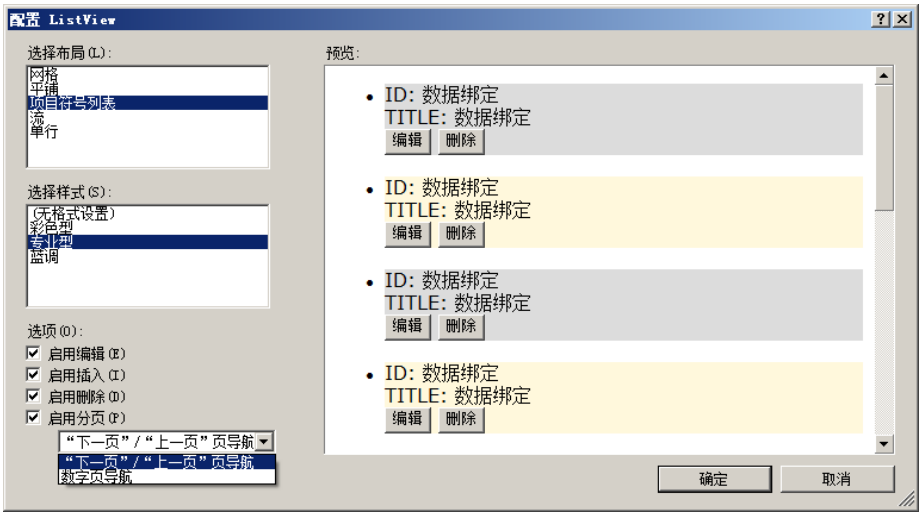


图 8-52　配置 ListView

开发人员能够选择相应的布局并选择相应的样式来确定 ListView 控件的界面，开发人员还可以通过选择【启用编辑】、【启用插入】等选项简化开发。

注意：当需要执行相应的数据操作时，数据源控件的高级选项都应该勾选。

当选择相应的布局方案和样式后，系统生成的 ListView 控件的 HTML 代码如下所示。

```
<asp:ListView ID="ListView1" runat="server" DataKeyNames="ID"
    DataSourceID="SqlDataSource1" InsertItemPosition="LastItem">
    <AlternatingItemTemplate>
        <li style="background-color: #FFF8DC;">ID:
            <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' />
            <br />
            TITLE:
            <asp:Label ID="TITLELabel" runat="server" Text='<%# Eval("TITLE") %>' />
            <br />
            <asp:Button ID="EditButton" runat="server" CommandName="Edit" Text="编辑" />
```

```
                <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="删除" />
            </li>
        </AlternatingItemTemplate>
        <LayoutTemplate>
            <ul ID="itemPlaceholderContainer" runat="server"
            style="font-family: Verdana, Arial, Helvetica, sans-serif;">
                <li ID="itemPlaceholder" runat="server" />
            </ul>
                <div style="text-align: center;background-color: #CCCCCC;font-family: Verdana, Arial,
                  Helvetica, sans-serif;color: #000000;">
                <asp:DataPager ID="DataPager1" runat="server">
            <Fields>
                <asp:NextPreviousPagerField ButtonType="Button" ShowFirstPageButton="True"
                ShowLastPageButton="True" />
            </Fields>
        </asp:DataPager>
            </div>
        </LayoutTemplate>
        <InsertItemTemplate>
            <li style="">TITLE:
            <asp:TextBox ID="TITLETextBox" runat="server" Text='<%# Bind("TITLE") %>' />
            <br />
            <asp:Button ID="InsertButton" runat="server" CommandName="Insert" Text="插入" />
            <asp:Button ID="CancelButton" runat="server" CommandName="Cancel" Text="清除" />
            </li>
        </InsertItemTemplate>
        <SelectedItemTemplate>
            <li style="background-color: #008A8C;font-weight: bold;color: #FFFFFF;">ID:
            <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' />
            <br />
            TITLE:
            <asp:Label ID="TITLELabel" runat="server" Text='<%# Eval("TITLE") %>' />
            <br />
            <asp:Button ID="EditButton" runat="server" CommandName="Edit" Text="编辑" />
            <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="删除" />
        </li>
        </SelectedItemTemplate>
        <EmptyDataTemplate>
            未返回数据。
        </EmptyDataTemplate>
         <EditItemTemplate>
            <li style="background-color: #008A8C;color: #FFFFFF;">ID:
            <asp:Label ID="IDLabel1" runat="server" Text='<%# Eval("ID") %>' />
            <br />
            TITLE:
            <asp:TextBox ID="TITLETextBox" runat="server" Text='<%# Bind("TITLE") %>' />
            <br />
            <asp:Button ID="UpdateButton" runat="server" CommandName="Update" Text="更新" />
            <asp:Button ID="CancelButton" runat="server" CommandName="Cancel" Text="取消" />
            </li>
```

```
            </EditItemTemplate>
            <ItemTemplate>
            <li style="background-color: #DCDCDC;color: #000000;">ID:
                <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' />
                <br />
                TITLE:
                <asp:Label ID="TITLELabel" runat="server" Text='<%# Eval("TITLE") %>' />
                    <br />
                <asp:Button ID="EditButton" runat="server" CommandName="Edit" Text="编辑" />
                <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="删除" />
            </li>
        </ItemTemplate>
        <ItemSeparatorTemplate>
            <br />
        </ItemSeparatorTemplate>
    </asp:ListView>
```

上述代码定义了 ListView 控件，系统默认创建了相应的模板，开发人员能够编辑相应的模板样式来为不同的编辑模式显示不同的用户界面。同时，用户可以无需代码实现就能够实现删除，更新以及添加等操作，运行结果如图 8-53 所示。

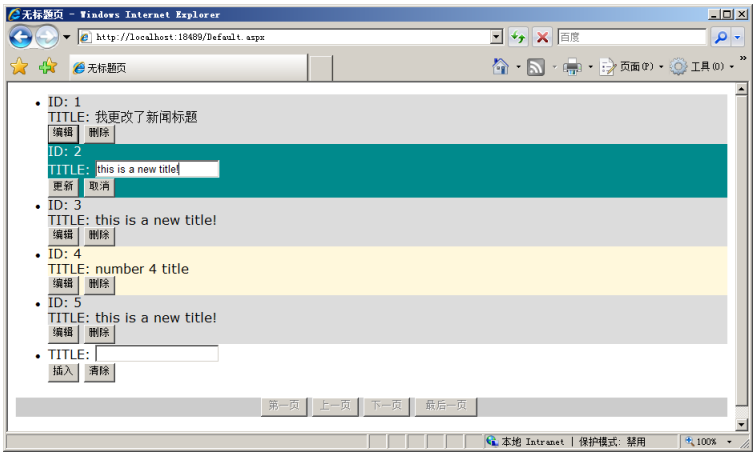


图 8-53　ListView 控件

LayoutTemplate 和 ItemTemplate 是标识定义控件的主要布局的根模板。通常情况下，它包含一个占位符对象，例如表行 tr 或 div 元素。此元素将由 ItemTemplate 模板或 GroupTemplate 模板中定义的内容替换。

如果需要定义自定义用户界面，则必须使用 LayoutTemplate 模板可以作为 ListView 控件的父容器。LayoutTemplate 模板是 ListView 控件所必需的。相同的是，LayoutTemplate 内容也需要包含一个占位符控件。占位符控件必须将包含 runat="server"属性，并且将 ID 属性设置为 ItemPlaceholderID 或 GroupPlaceholderID 属性的值，示例代码如下所示。

```
    <ItemTemplate>
        <td runat="server" style="background-color:#DCDCDC;color: #000000;">
            ID:
            <asp:Label ID="IDLabel" runat="server" Text='<%# Eval("ID") %>' /><br />
            TITLE:
            <asp:Label ID="TITLELabel" runat="server" Text='<%# Eval("TITLE") %>' /><br />
            <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="删除" /><br />
```

```
            <asp:Button ID="EditButton" runat="server" CommandName="Edit" Text="编辑" /><br />
        </td>
    </ItemTemplate>
```

ListView 控件的事件和 FormView 控件的事件基本相同，同样可以为 ListView 控件执行更新、删除或添加等事件编写相应的代码。当执行更新前、更新时都可以触发相应的事件，示例代码如下所示。

```
    protected void ListView1_ItemUpdated(object sender, ListViewUpdatedEventArgs e)
    {
        Label1.Text = "更新已经发生";                                    //触发更新事件
    }
```

当运行后，则会触发 ItemUpdated 事件，运行结果如图 8-54 所示。

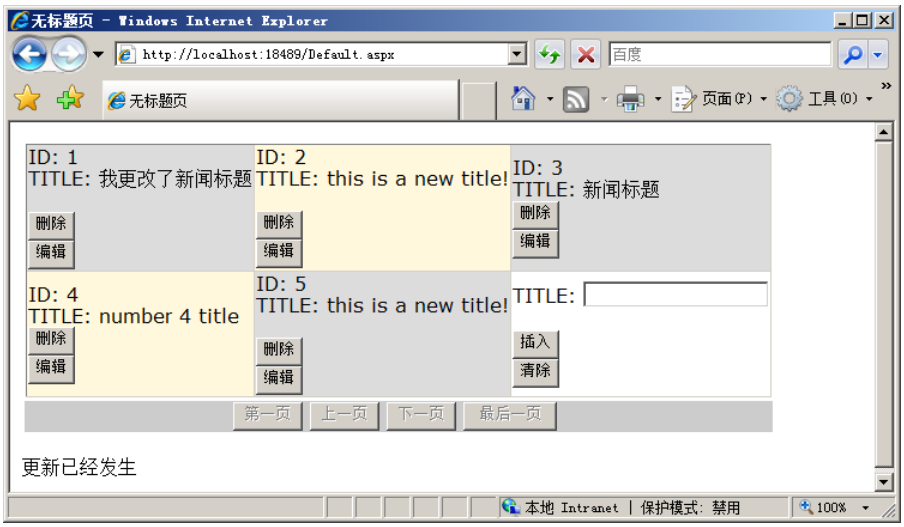


图 8-54　ItemUpdated 事件

ListView 控件不仅能够支持 FormView 控件的事件，而 ListView 控件具有更多的布局手段。ListView 控件能为开发人员在开发中提供极大的遍历，当如果需要进行相应的数据操作，又需要快捷的显式数据和添加数据时，ListView 控件是极佳的选择。

8.8　数据绑定控件（DataPager）

DataPager 控件通过实现 IPageableItemContainer 接口实现了控件的分页。在 ASP.NET 3.5 中，ListView 控件适合可以使用 DataPager 控件进行分页操作。要在 ListView 中使用 DataPager 控件只需要在 LayoutTemplate 模板中加入 DataPager 控件。DataPager 控件包括两种样式，一种是“上一页/下一页”样式，第二种是“数字”样式，如图 8-55 和图 8-56 所示。

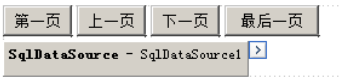


图 8-55　文本样式

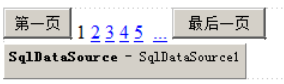


图 8-56　数字样式

当使用“上一页/下一页”样式时，DataPager 控件的 HTML 实现代码如下所示。

```
    <asp:DataPager ID="DataPager1" runat="server">
        <Fields>
            <asp:NextPreviousPagerField ButtonType="Button" ShowFirstPageButton="True"
```

```
            ShowLastPageButton="True" />
        </Fields>
    </asp:DataPager>
```

当使用“数字”样式时，DataPager 控件的 HTML 实现代码如下所示。

```
    <asp:DataPager ID="DataPager1" runat="server">
        <Fields>
            <asp:NextPreviousPagerField ButtonType="Button" ShowFirstPageButton="True"
            ShowNextPageButton="False" ShowPreviousPageButton="False" />
            <asp:NumericPagerField />
            <asp:NextPreviousPagerField ButtonType="Button" ShowLastPageButton="True"
            ShowNextPageButton="False" ShowPreviousPageButton="False" />
        </Fields>
    </asp:DataPager>
```

除了默认的方法来显示分页样式，还可以通过向 DataPager 中的 Fields 中添加 TemplatePagerField 的方法来自定义分页样式。在 TemplatePagerField 中添加 PagerTemplate，在 PagerTemplate 中添加任何服务器控件，这些服务器控件可以通过实现 TemplatePagerField 的 OnPagerCommand 事件来实现自定义分页。

8.9 小结

本章介绍了有关 ASP.NET 中绑定数据和数据源相关的控件，在 ASP.NET 中，这些控件强大的功能让开发变得更加的简单。在 ASP.NET 中，正是因为这些数据源控件和数据绑定控件，让开发人员在页面开发时，无需更多的操作即可实现强大的功能，解决了在传统的 ASP 中难以解决的问题。本章还包括：

- ADO.NET：讲解了 ADO.NET，并介绍了使用 ADO.NET 连接数据库。
- 数据源控件：包括 SqlDataSource 等常用的数据源控件，并一步步的介绍了数据源控件的配置。
- 重复列表控件：讲解了如 Repeater 之类的重复列表控件。
- 数据绑定控件：讲解了常用的数据绑定控件并使用数据绑定控件对数据进行更新，删除等操作。

数据操作无论是在 Web 开发还是在 Win Form 开发中，都是要经常使用的，数据控件能够极大的简化开发人员对数据的操作，让开发更加迅速。