第6章

基础篇——网站前台的实现

模块1　创建IT企业网站前台

教学目标：

- 掌握创建 ASP. NET 网站的方法；
- 了解 ASP. NET 应用程序的组成；
- 了解 ASP. NET 两种编码方式；
- 了解 Page 类的方法与属性；
- 了解 Page 的生命周期及其应用；
- 学会使用 isPostBack 属性。

6.1.1　操作步骤

启动 Visual Studio 2010，选择“文件”|“新建网站”命令，打开“新建网站”对话框。在“位置”下拉列表框中选择“文件系统”选项，在“语言”下拉列表框中选择“Visual C＃”语言，单击“浏览”按钮，选择网站路径为“F:\IT企业网站\qywz”。如图 6-1 所示。单击【确定】按钮，即可生成一个新的网站。

把“课程素材”目录下的 image 文件夹拷贝到“F:\企业网站\qywz”目录下。

图 6-1　新建网站

6.1.2　相关知识

1. 解决方案的组成

下面我们来看看 Visual Studio 自动产生了哪些内容。通过“解决方案资源管理器”，我们可以看到如图 6-2 所示的效果。

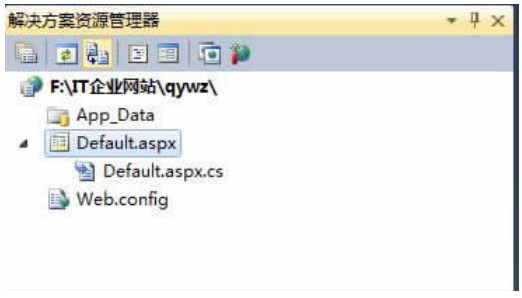

图 6-2　站点目录

站点路径下面，默认创建了一个文件夹和三个文件。

App_Data	存放数据的文件夹
Default. aspx	自动创建的页面文件
Default. aspx. cs	代码后置文件
Web. Config	站点配置文件

2. 编码方式

ASP. NET 有两种编码方式：代码内嵌和代码后置。

（1）代码内嵌的特点是 HTML 标记和服务器端运行的 C＃代码全部包含在一个. aspx 页面中。代码内嵌的缺点是页面和代码混在一起，维护起来较为麻烦。

（2）代码后置模式就是 HTML 标记和 C＃代码分别存储于不同的文件中。HTML 标记放在. aspx 页面中，而 C＃代码放在一个 aspx. cs 文件中。代码后置模式的好处是页面展示部分和逻辑控制部分分离开来，便于管理和维护。这也是微软推荐的开发方式。

（3）演示单页模型和代码页面分离模式。

3. ASP. NET 运行机制

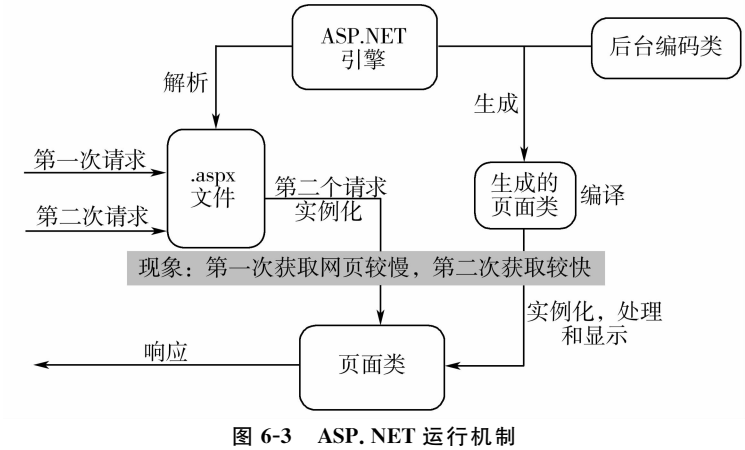

图 6-3　ASP. NET 运行机制

浏览器发出 HTTP 请求到 Web 服务器要求访问一个 Web 网页，如果是第一次请求，

ASP. NET 引擎首先编译. ASPX 文件和. CS 文件，合并生成页面类，然后执行页面中的处理，返回处理结果；当第二次请求该页面时，由于该页面类已存在，省去了编译环节，直接执行页面中的处理就行了。然后，把这个处理结果传回浏览器作为 HTTP 的响应，最后，浏览器收到这个响应后显示该网页。

4. Page 对象

前面讲到 ASP. NET 的运行机制时，我们提到了页面类，它由. ASPX 文件和. CS 文件合并生成。它继承自 System. Web. UI. Page 类。（System. Web. UI 提供用于创建 ASP. NET 网站用户界面的类和接口。）

每一个 ASP. NET 的页面对应一个页面类。Page 对象就是页面类的实例。下面我们来看一个 ASP. NET 的页面。

5. @Page 指令

打开 Default. aspx 页面源视图，在第一行可以看到这样一行代码：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

这就是常说的页面指令。

@Page 指令定义了 ASP. NET 页用于编译和解析的属性。每个 aspx 页面只能有一个@Page指令。

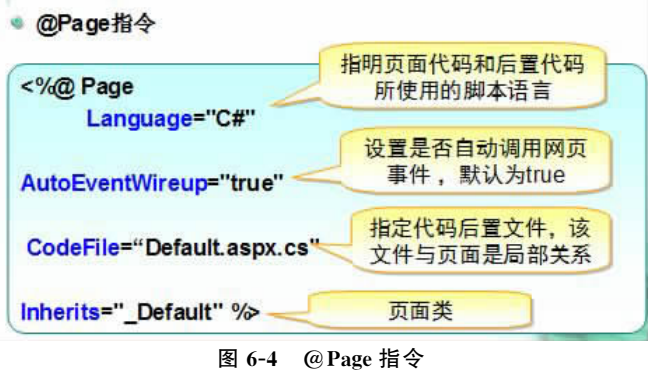

图 6-4　@Page 指令

其中，AutoEventWireup="true"指定页面事件自动触发；CodeFile="Default. aspx. cs"指定后台编码文件，使得显示界面和后台编码文件相互关联；Inherits="_Default"指定继承的类名。

6. ASP. NET 页面的生命周期

ASP. NET 页面运行的时候将经历一个生命周期，这个生命周期中会进行一系列的操作，引发一系列的事件。页面事件的引发时机见表 6-1。

表 6-1　　页面事件的引发时机

事件	引发时机
Init	页面构架初始化时
Load	页面载入内存时
控件事件	如鼠标点击时
Unload	页面内存卸载时

下面创建一个 PageLoad.aspx 页面，用于演示页面的各个事件，界面如图 6-5 所示。

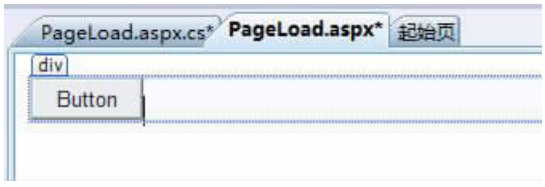

图 6-5　PageLoad.aspx 页面

代码如下：

```
public partial class PageLoad : System.Web.UI.Page
{
    protected void Page_Init(object sender, EventArgs e)
    {
        Response.Write("正在初始化(Init 事件)");
        Response.Write("<BR>");
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        if (! Page.IsPostBack)
        {
            Response.Write("这是第一次 Load (load 事件)");
        }
        else
        {
            Response.Write("这是 PostBack 后 Load (load 事件)");
            Response.Write("<BR>");
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Write("欢迎您 (Click 事件)");
        Response.Write("<BR>");
    }
}
```

事件的顺序：第一次运行 PageLoad.aspx 时，页面第一次被加载，如图 6-6 所示，可以看出，图中运行了页面的 Init 事件与 Load 事件。

图 6-6　页面首次被加载

单击“Button”按钮时，页面提交给服务器，这个过程为 PostBack(回发)。页面再次被加载，运行后产生新的页面返回，回发页面如图 6-7 所示。

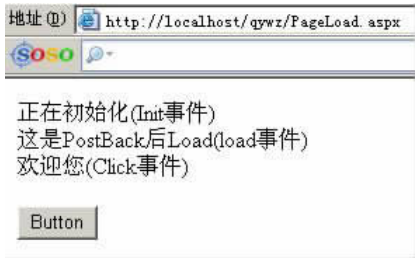

图 6-7 回发页面

从图 6-7 可看出，单击“Button1”按钮时，并不是仅仅执行 Button1_Click 事件，而是先后执行了 Init、Load、Button1_Click 事件，这就充分说明了页面的生命周期中依次引发 Init—Load—控件事件。由此可知，当控件的事件被触发时，Page_Load 事件会在控件的事件之前被触发。如果想在执行控件的事件代码时不执行 Page_Load 事件中的代码，可以通过判断属性 Page. IsPostBack 实现。

如何判断回发和首次加载？在使用过程中可以利用 Page. IsPostBack 属性的值来判断。它是一个布尔值，当该值为真(true)时，则页面回转，否则就是首次加载，如图 6-8 所示。

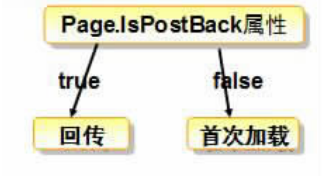

图 6-8 判断回发和首次加载

所以，上面的代码可以修改为：

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (! Page.IsPostBack)
        {
            //此次执行仅在第一次请求时运行一次的初始化代码
        }
    }
```

在本课程以后的程序中，Page_Load 通常采用以上形式。

6.1.3 实训 6

参照操作步骤，创建 IT 企业网站前台，把“课程素材”目录下的 image 文件夹拷贝到“F:\IT 企业网站\qywz”目录下。

模块 2 创建数据访问类

教学目标：

- 掌握 ADO. NET 对象的综合应用；

- 学会数据库连接类中各种方法的编写。

我们将本课程用到的方法提取出来，作为 DBHelper 类。该类包含了数据库的连接和三个方法：

（1）执行 select 语句的方法；

（2）执行 insert、delete、update 语句方法；

（3）装载数据到下拉列表框的方法。

6.2.1　操作步骤

参照 4.3.1 节的操作步骤，在模板栏中选择“类”，并将【名称 N】文本框中的名称改为 DBHelper.cs，单击【添加】按钮即可，代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public static class DBHelper
{
    private static SqlConnection conn = new SqlConnection("server=2012-20130831CV; uid=qywz;
pwd=1234;database=qywz");
    private static SqlDataAdapter da;
    private static SqlCommand cmd;

    //执行 select 语句并返回一个表
    public static DataTable getdt(string safeSql)
    {
        da = new SqlDataAdapter(safeSql, conn);
        DataTable dt = new DataTable();
        da.Fill(dt);
        conn.Close();
        return dt;
    }
    // 执行 insert delete update 语句，不返回结果
    public static void ExecuteCommand(string safeSql)
    {
        conn.Open();
        cmd = new SqlCommand(safeSql, conn);
        cmd.ExecuteNonQuery();
        conn.Close();
    }
    //装载数据到下拉列表框
     public static void loadddl(string field_name1, string field_name2, string table_name,
```

```
DropDownList DropDownList_name)
        {
            string strSQL = "select " + field_name1 + "," + field_name2 + " from " + table_name;
            da = new SqlDataAdapter(strSQL, conn);
            DataTable dt = new DataTable();
            da.Fill(dt);
            DropDownList_name.DataSource = dt;
            DropDownList_name.DataValueField = field_name1;//表示控件代表的值
            DropDownList_name.DataTextField = field_name2;//表示控件显示出的字段
            DropDownList_name.DataBind();
        }
    }
```

6.2.2 相关知识

SqlCommand 对象简介

SqlCommand 对象是用来操作数据库的对象，使用它向数据库发出各种操作命令，比如添加数据、修改数据、删除数据、查询数据。

使用 SqlCommand 对象步骤：

步骤 1　创建 SqlCommand 对象。

```
SqlCommand cmd = new SqlCommand();
```

步骤 2　通过 Connection 属性设置连接对象。

```
cmd.Connection = conn;
```

通过 CommandText 属性设置 SQL 语句：

```
cmd.CommandText = "insert into ...";
```

步骤 1 和步骤 2 可以合为一步：

```
SqlCommand cmd = new SqlCommand("insert into ...",conn);
```

步骤 3　通过方法执行命令，见表 6-2。

表 6-2　　SqlCommand 对象的方法

方法名	返回类型	说明
ExecuteNonQuery	int	执行 SQL 并返回受影响的行数
ExecuteScalar	object	执行 SQL 并返回第一行第一列数据
ExecuteReader	SqlDataReader	返回只读的数据流对象

由于对数据库的操作有许多种，因此 SqlCommand 包含操作数据库的许多不同方法。

ExecuteNonQuery 主要用来执行增加、修改、删除数据操作，调用该方法有一个返回值，即执行该操作影响数据的行数。

比如，SQL 语句删除两条数据，如果两条都删除了，就会返回 2，如果成功删除一条就返回 1。利用该返回值，可以判断对数据的操作是否成功。

ExecuteScalar 主要用来查找数据。它返回的是数据集中第 1 行第 1 列的数据。因为返回的数据类型可能是任意类型，所以调用该方法的返回结果为 object 类型，一般在使用时再把 object 类型转换成需要的其他类型。

ExecuteReader 返回用来查找多行数据，返回 SqlDataReader 对象，SqlDataReader 对象也

是 ADO. NET 的重要对象，在后面会介绍它。

这些方法这使用时可以不带任何参数。比如，cmd. ExecuteNonQuery()。

6.2.3　实训 7

参照操作步骤，创建数据访问类，命名为 DBHelper. cs。

模块 3　创建前台主页面

教学目标：

- 学会样式表的创建，会用 DIV＋CSS 布局页面；
- 学会站点地图的制作及站点地图数据源的使用；
- 学会如何在页面中插入 Flash 动画；
- 掌握使用 Menu 控件来设计水平导航条的方法；
- 掌握前台母版页的设计与实现。

6.3.1　建立样式表

打开网站，在解决方案资源管理器中，右击网站，在弹出的快捷菜单中选择【添加新项】菜单命令，弹出添加新项对话框，如图 6-9 所示。

图 6-9　添加新项

在模板栏中选择“样式表”，并将【名称 N】文本框中的名称改为 ytdh. css，然后单击【添加】

按纽,便添加了一个样式表。编写代码如下:

```
* {
    border-width: 0;
    margin:  0;
    padding: 0;
}
#title
{
    width: 777px;
    height: 100px;
}
#ytdh
{
    margin-left: auto;
    margin-right: auto;
    width: 777px;
    position: relative;
    font-weight: normal;
    background: url(image/bg-nav.png) #608fc8 no-repeat left top;
    padding: .8em 0 1em 0px;
}
#ytdh DIV
{
    background: url(image/bg-nav-side.png) #4b6cb5 no-repeat right top;
    width: 10.5 % ;
    _width: 25.5 % ;
    position: absolute;
    top: 0;
    right: 0;
    padding: .8em 0 1em 0;
}
#ytdh A:link, #ytdh A:visited
{
    color: #ffffff;
    text-decoration: none;
}
#ytdh A:hover
{
    color: #ff9900;
    text-decoration: underline;
}
#footer
```

```
{
    background: url(image/bg-nav.png) #608fc8 no-repeat left bottom;
    margin-top: 3px;
    margin-left: auto;
    margin-right: auto;
    width: 777px;
    padding: .8em 0 1em 0px;
    position: relative;
    font-family:宋体;
    font-size: 12px;
    text-align: center;
}
#footer div
{
    background: url(image/bg-nav-side.png) #4b6cb5 no-repeat right bottom;
    width: 10.5%;
    position: absolute;
    top: 0;
    right: 0;
    padding: .8em 0 1em 0;
}
```

6.3.2　CSS 技术

CSS(Cascading Style Sheets,层叠样式表)技术是一种新兴的结构描述技术,可以为 Web 结构语言中的各种元素定义尺寸、位置、背景以及文本的各种效果,是目前正在使用的 Web 前端描述语言标准。

CSS 本身是一种数据表格式,作用是为 XML、HTML 以及 XHTML 等结构化的文档添加样式描述,实现对文档中内容的排版和美化。在页面设计时采用 CSS 技术,可以有效地对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制。只要对相应的代码做一些简单修改,就可以改变同一页面的不同部分或者页数不同的网页外观和格式。

CSS 为标记语言提供了一种样式描述,定义其中的元素显示方式。CSS 在设计领域中是一个突破,仅仅通过 CSS 样式表就能够使 Web 开发者控制所有出现在 Web 中的外观及布局,并且为每种标记语言的元素和应用该元素的每个页面定义所需要的样式。简单地改变样式,所有与之相关的元素都会自动更新。

CSS 的出现还使网页文档中结构与表现的分离成为可能。将 CSS 与 XHTML 文档相结合,除了可以完全实现 HTML 语言所能实现的功能外,还增强了对各种网页元素的精确控制,从而使网页中各种元素以精确到像素的方式显示,丰富了网页元素的内涵。

6.3.3　建立网站地图

打开网站,在解决方案资源管理器中,右击网站,在弹出的快捷菜单中选择【添加新项】菜单

命令，弹出添加新项对话框，如图 6-10 所示。

图 6-10　添加新项

在模板栏中选择“站点地图”项，采用默认名称“Web. sitemap”，单击【添加】按钮，便添加了一个站点地图。源代码如下：

```
<? xml version="1.0" encoding="utf-8" ? >
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title="企业网站"  description="">

        <siteMapNode url="Default1.aspx" title="首页"  description="" />
        <siteMapNode url="gsjj.aspx" title="公司简介"  description="" />
         <siteMapNode url="" title="新闻动态"  description="">
        <siteMapNode url="newlist.aspx? id= %" title="新闻"  description="" />
        <siteMapNode url="newlist.aspx? id=1" title="公司新闻"  description="" />
        <siteMapNode url="newlist.aspx? id=2" title="行业新闻"  description="" />
      </siteMapNode>
      <siteMapNode url="prolist.aspx? id= %" title="产品信息"  description="" />
      <siteMapNode url="ordercx.aspx" title="查看订单"  description="" />
      <siteMapNode url="xgmm.aspx" title="修改密码"  description="" />
      <siteMapNode url="lxwm.aspx" title="联系我们"  description="" />
      <siteMapNode url="login.aspx" title="后台管理"  description="" />

    </siteMapNode>
</siteMap>
```

6.3.4　用户控件 head.ascx 和 footer.ascx

操作步骤如下：

1. head.ascx 用户控件

（1）打开网站，在解决方案资源管理器中，右击网站，在弹出的快捷菜单中选择【添加新项】菜单命令，弹出添加新项对话框，如图 6-11 所示。

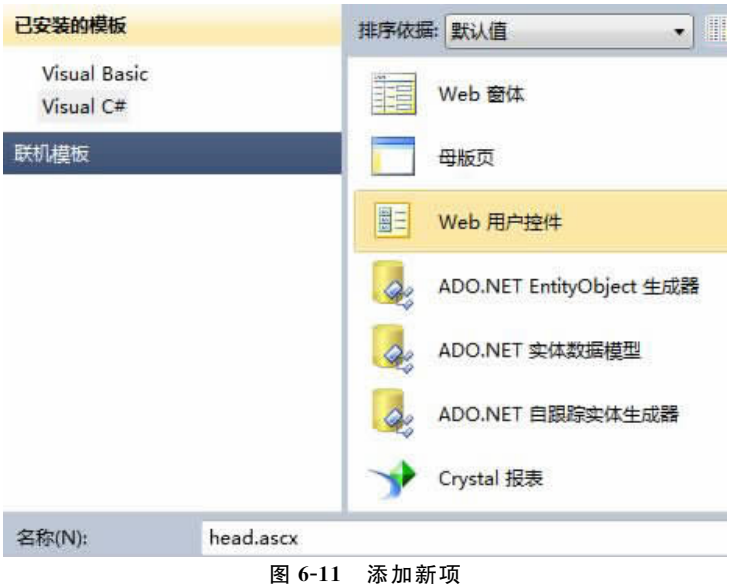

图 6-11　添加新项

在【模板 T】栏中选择“Web 用户控件”，名称改为“Head.ascx”，单击【添加】按钮，便添加一个 Web 用户控件。

（2）向 Web 用户控件加入一个层，在层中插入一幅图，源代码如下：

```
<div >
    <embed src="image/1.swf" style="width: 777px; height: 100px"></embed>
</div>
```

（3）添加一个站点地图数据源 SiteMapDataSource1，设置 ShowStartingNode 属性为 False。

（4）再加入一个层，在层中加入一个 Menu 控件，设置 Menu 的 DataSourceID 属性为 SiteMapDataSource1，Orientation 属性为 Horizontal。

（5）附加样式表。点击“样式”→“附加样式表”。源代码如下：

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="head.ascx.cs" Inherits="head" %>
<link href="ytdh.css" rel="Stylesheet" type="text/css" />
<div id="title">
    <embed src="image/1.swf" style="width: 777px; height: 100px"></embed>
</div>
<div id="ytdh">
  <asp:Menu ID="Menu1" runat="server" DataSourceID="SiteMapDataSource1" Orientation="Horizontal"
      Width="616px" Height="10px" >
    </asp:Menu>
    <div></div>
```

```
</div>
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" ShowStartingNode="False" />
```

设计界面如图 6-12 所示。

图 6-12 head. ascx 设计界面

2. footer. ascx 用户控件

向网站添加一个 footer. ascx 用户控件，向 Web 用户控件插入一个层，在层里加入一行文字，源代码如下：

```
<%@Control Language="C#"AutoEventWireup="true"CodeFile="footer.ascx.cs"Inherits="footer"%>
<link href="ytdh.css" rel="stylesheet" type="text/css" />
<div id="footer">
    Copyright&copy; 2013 版权所有：职业技术学院 计算机软件开发小组<div></div>
</div>
```

设计界面如图 6-13 所示。

图 6-13 footer. ascx 设计界面

6.3.5 前台母版页和主页面

操作步骤如下：

1. 创建前台母版页

（1）打开网站，右击网站，在弹出的快捷菜单中选择【添加新项】菜单命令，弹出添加新项对话框，如图 6-14 所示。

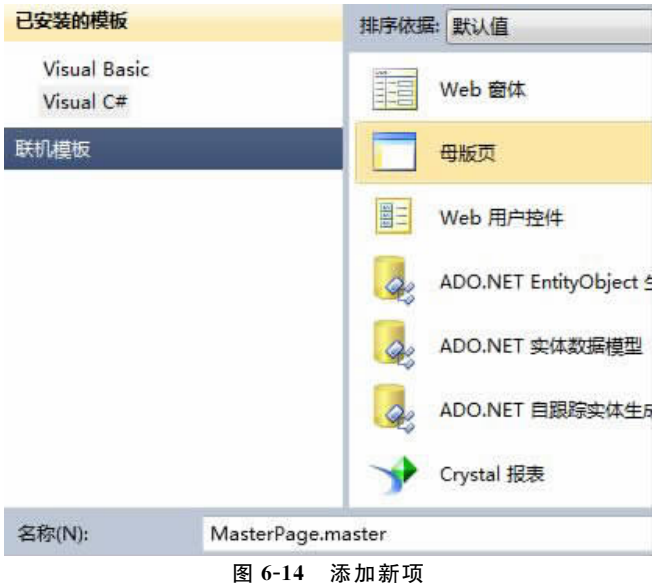

图 6-14 添加新项

在【模板 T】栏中选择“母版页”，采用默认名称，单击【添加】按钮，便添加一个模板页。

（2）剪切模板页中的 ContentPlaceHolder1 控件，向模板页插入一个 1 行 1 列的表格（表格 1），属性设置如图 6-15 所示。

图 6-15　表格属性

单元格属性设置为：align="center" valign="top"

（3）向表格 1 插入一个 2 行 1 列的表格（表格 2），向表格 2 第 1 行拖入 head.ascx；在表格 2 的下面粘贴 ContentPlaceHolder1 控件；在表格 1 的下面拖入 footer.ascx 用户控件，如图 6-16 所示。

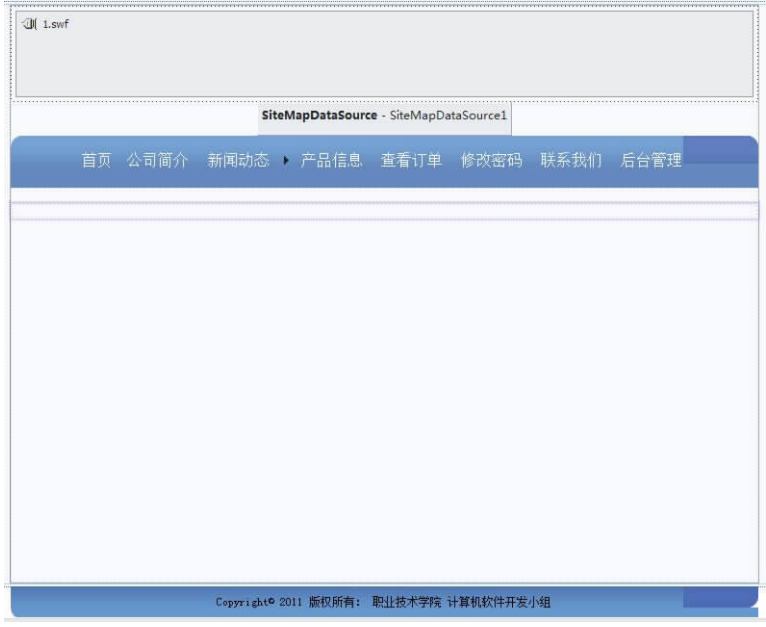

图 6-16　前台母版页界面

2. 创建前台主页面

向网站添加一个 Default. aspx 页面，选择母版页 MasterPage. master，如图 6-17 所示。

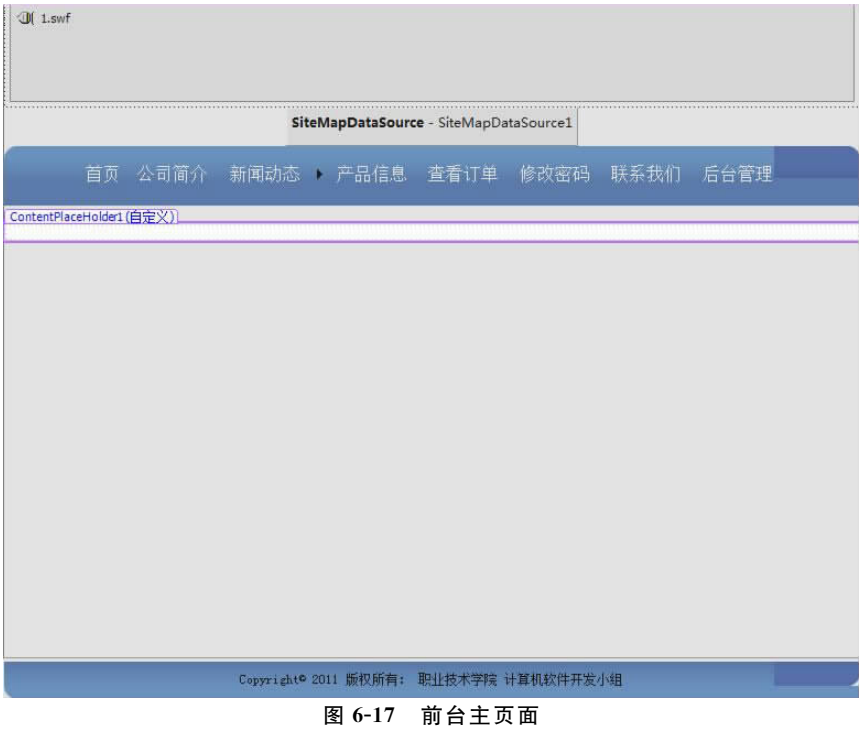

图 6-17 前台主页面

6.3.6 相关知识

用户控件简介

用户控件(User Control)是一种自定义的组合控件，通常由系统提供的可视化控件组合而成。在用户控件中不仅可以定义显示界面，还可以编写事件处理代码。当多个网页中包括有部分相同的用户界面时，可以将这些相同的部分提取出来，做成用户控件。

一个网页中可以放置多个用户控件。通过使用用户控件不仅可以减少编写代码的重复劳动，还可以使得多个网页的显示风格一致。更为重要的是，一旦需要改变这些网页的显示界面时，只需要修改用户控件本身，经过编译后，所有网页中的用户控件都会自动跟随变化。

用户控件本身就相当于一个小型的网页，同样可以为它选择单文件模式或者代码分离模式。然而用户控件与网页之间还是存在着一些区别，这些区别包括：

(1) 用户控件文件的扩展名为. ascx 而不是. aspx；代码的分离(隐藏)文件的扩展名是. ascx. cs而不是. aspx. cs。

(2) 在用户控件中不能包含<HTML>、<BODY>和<FORM>等 HTML 语言的标记。

(3) 用户控件可以单独编译，但不能单独运行。只有将用户控件嵌入到. aspx 文件中时，才能和 ASP. NET 网页一起运行。

除此以外，用户控件与网页非常相似。

页面的布局

页面可分为上、中和下三个部分，上部分显示站点的标题和导航信息，下部分显示站点的版权声明、联系方式和导航信息，中间部分为页面内容。通常，把上、下两个部分做成母版页，以后每新建 Web 窗台页时都选择母版页，以此实现网站风格的统一性和协调性。

母版页和内容页概述

ASP. NET 定义了两种新的页面类型：母版页和内容页。母版页是一个页面模板，与普通的 ASP. NET Web 页面一样，它可以包含任何 HTML、Web 控件甚至代码的组合。此外，母版页面还可以包含内容占位符——被定义的可修改的区域。每个内容页引用一个母版页并获得它的布局和内容。此外，内容页可以在任意的占位符里加入页面特定的内容。

母版页通常用于布局，即定义网站中不同网页的相同部分。母版页为具有扩展名. master 的 ASP. NET 文件，它可以包括静态文本、HTML 元素和服务器控件。

母版页代码和普通的. aspx 文件代码格式很相近，最关键的不同是母版页由特殊的@ Master 指令识别，该指令替换了用于普通. aspx 页的@Page 指令，格式如下：

```
<%@ Master Language="C#" CodeFile="MasterPage.master.cs" Inherits= "MasterPage" %>
```

母版页相当于网页的模板。在其他网页中，只要引用了母版页，母版页的页面就可以自动显示出来，设计者可以修改母版页中的内容部分，其他部分保持不变，这样就可以使所有页面的风格保持一致，给网页设计带来了很大的方便。

6.3.7　实训 8

参照操作步骤，完成下列任务：

(1) 创建包括样式表；

(2) 创建站点地图；

(3) 创建用户控件 head. ascx 和 footer. ascx；

(4) 创建母版页、前台主页面。

模块 4　创建新闻列表页面

教学目标：

- 掌握新闻列表页面的设计与实现；
- 掌握页面间传输参数的三种方式。

6.4.1　页面功能

设计一个新闻列表页面，显示新闻标题、新闻类别编号、添加时间、阅读次数等信息；该页面既能显示全部新闻，又能分类显示新闻。

根据需求，编写一条 SQL 语句，要求能够进行无条件查询和有条件查询。要达到这个目

的，只能采用模糊查询 like，如：

```
select * from 新闻表 where 新闻类别编号 like'参数'
```

这是带有参数的查询，传递参数为新闻类别编号，所显示的新闻由传递参数决定。当“参数＝%”时显示全部新闻，“参数＝1”时显示公司新闻，“参数＝2”时显示行业新闻。

设计步骤如下：

（1）设计新闻列表页面 newlist. aspx。

（2）在 Web. sitemap 文件中链接 newlist. aspx 页面，同时将参数传给 newlist. aspx 页面。

（3）在 newlist. aspx 页面接收首页传来的参数，作为查询条件值。

（4）编写 SQL 语句，查询新闻表中新闻类别编号匹配此参数的记录。

（5）调用类的方法执行 SQL 语句，并接收执行的结果。

（6）绑定数据。

（7）调试运行，输出结果。

6.4.2 操作步骤

（1）设计界面。向网站添加一个 newlist. aspx 页面，选择母版页 MasterPage. master，在空白处插入一个 2 行 1 列的表格，设置其 align 属性为 center、边框属性为 1。

（2）第 1 行样式设置为：

```
<td style="font-size:9pt; background-image: url(image/t_bg01.gif); width: 475px;
color: #ffffff; height: 16px; text-align: left">
::新闻</td>
```

（3）向第 2 行加入一个 GridView 控件，设置其属性 Font-Size＝"Small" GridLines＝"None" ShowHeader＝"False"

点击其右上角箭头，进入 GridView 任务栏，点击“编辑列”，进入“字段”对话框，在可用字段列表框中添加一个 HyperLinkField 字段，三个 BoundField 字段到选定的字段列表框。

（4）设置各字段的属性，见表 6-3。

表 6-3　　字段属性设置

字段名	属性	属性值
HyperLinkField	DataNavigateUrlFields DataTextField DataTextFormatString	新闻编号 新闻标题 :{0}
BoundField	DataField	添加时间
BoundField	DataField	新闻类别编号
BoundField	DataField	阅读次数

其中，{0}在网页浏览时会被新闻标题的值代替。设计界面界面如图 6-18 所示。

图 6-18　newlist.aspx 设计界面

（5）创建超链接。打开 Web.sitemap 文件，为“新闻”、“公司新闻”、“行业新闻”编写 url。

```
<siteMapNode url="newlist.aspx? id= %" title="新闻"  description="" />
<siteMapNode url="newlist.aspx? id=1" title="公司新闻"  description="" />
<siteMapNode url="newlist.aspx? id=2" title="行业新闻"  description="" />
```

（6）编写事件。

```
protected void Page_Load(object sender, EventArgs e)
    {
        //接收请求页面传过来的参数，作为查询的条件值。
        string xwlbbh = Request.Params["id"].ToString();
        //编写 SQL 语句查询新闻表中新闻类别编号匹配此参数的记录。
        string strsql = "select * from 新闻表 where 新闻类别编号 like '"+xwlbbh+"'order by 新闻编号 desc";
        //调用类的方法执行 SQL 语句，并接收执行的结果。
        GridView1.DataSource = DBHelper.getdt(strsql);
        GridView1.DataBind();//绑定数据。
    }
```

（7）调试运行，得出如图 6-19 所示效果图。

图 6-19 newlist. aspx 运行效果图

6.4.3 相关知识

页面间参数的传递

在 ASP. NET 中，经常需要从一个页面跳转到另一个页面，进行页面跳转时，有时需要将源页面中的某个参数传递给目标页面，此时就涉及页面间参数传递的问题。在 ASP. NET 中页面间参数的方式传递主要有以下三种：通过 URL 路径传递参数；通过 Form 表单提交传输数据；使用 Server. Transfer 变量传递参数。以上三种方式不管那一种传递过来的数据，都可以用 Request 对象来获取。Request 是一个系统对象，它和另一个系统对象 Response 负责页面输入输出控制，各自功能如下：

(1) Request 对象作用

- 获得页面输入
- URL 传参
- Form 表单提交

Request 对象的常用属性如图 6-20 所示。

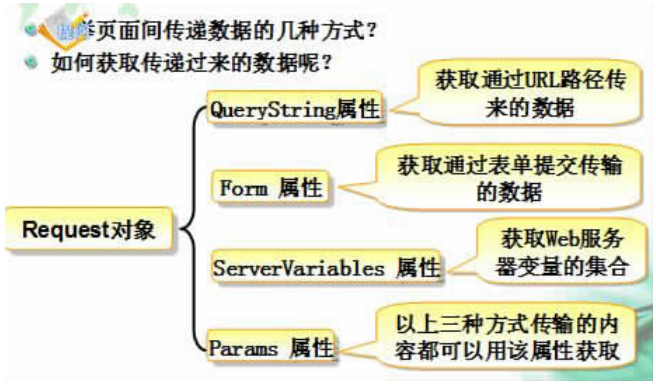

图 6-20　Request 对象属性

(2) Response 对象作用

- 在页面输出要显示的内容
- 重新定向

Response 对象的常用方法如图 6-21 所示。

图 6-21　Response 对象的方法

6.4.4　实训 9

参照操作步骤,创建新闻列表页面。

模块 5　新闻内容显示页面

教学目标:

- 掌握新闻信息内容页面的设计与实现;
- 了解 DataList 控件的基本设置;
- 掌握 DataList 控件的使用方法。

6.5.1　页面功能

新闻内容显示页面主要用来显示新闻的详细内容,包括新闻标题、添加时间、阅读次数、新闻内容等。通过这个页面,用户可以了解新闻的详细信息,用户只需要点击新闻列表页面中的新闻标题,即可连接到新闻内容页面,来获取新闻的详细信息。

设计步骤如下：

（1）设计新闻内容页面 newshow. aspx。

（2）在 newlist. aspx 页面中打开 newshow. aspx 页面，同时将新闻编号参数传给 newshow. aspx 页面。

（3）在 newshow. aspx 页面中接收 newlist. aspx 页面传来的参数，作为查询条件值。

（4）编写 SQL 语句，查询新闻表中新闻编号＝此参数的记录。

（5）调用类的方法执行 SQL 语句，并接收执行的结果。

（6）绑定数据。

6.5.2 操作步骤

（1）设计界面。向网站添加一个 newshow. aspx 页面，选择母版页 MasterPage. master，在空白处插入一个 2 行 1 列的表格，设置其 align 属性为 center、边框属性为 1。

（2）第 1 行样式设置为：

```
<td style="font-size:9pt; background-image:
url(image/t_bg01.gif); color: #ffffff;
                text-align: left">::新闻信息</td>
```

（3）向第 2 行加入一个 DataList 控件，点击其右上角箭头，进入 DataList 任务栏，点击“编辑模板”，此时进入“DataList_项模板”。

（4）在 ItemTemplate 模板中添加一个 3 行 1 列的表格，设置其边框属性为 0、align 属性为 center、Width＝100％；、height＝100％，如图 6-22 所示。

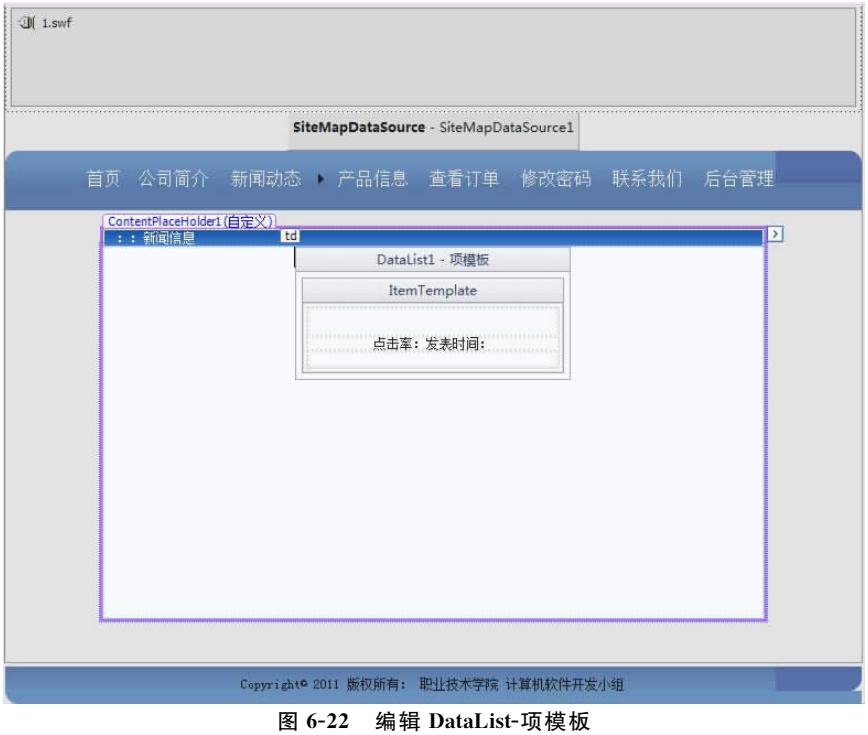

图 6-22 编辑 DataList-项模板

设置各单元格属性并绑定数据，源代码如下：

```
<table border="0" cellpadding="0" cellspacing="0" style="width: 100%; height: 100%">
<tr>
<td style="font-size:20pt; width: 100%; height: 20%; text-align: center">
<%# Eval("新闻标题")%></td>
</tr>
<tr>
<td style="width: 100%; height: 10%; text-align: center">
点击率:<%# Eval("阅读次数")%>     发表时间:<%# Eval("添加时间")%></td>
</tr>
<tr>
<td style="width: 100%; height: 70%; text-align: justify">
<%# Eval("新闻内容")%></td>
</tr>
</table>
```

结束模板编辑，得出如图 6-23 所示的效果图。

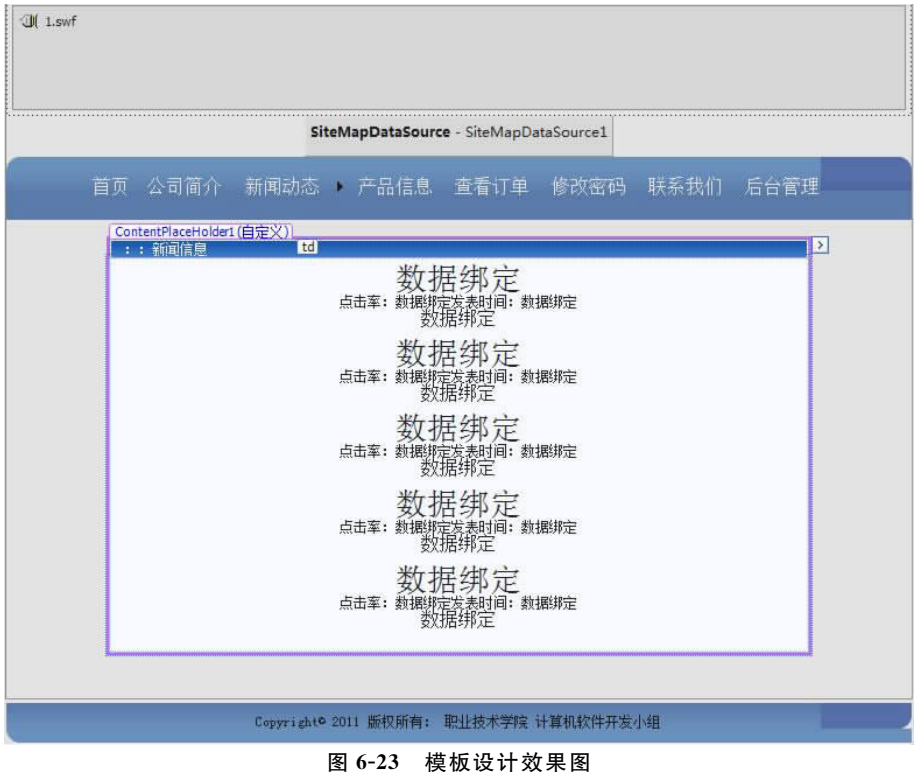

图 6-23　模板设计效果图

（5）创建超链接。打开 newlist.aspx 页面，设置 HyperLinkField 的 DataNavigateUrlFormatString 为 newshow.aspx? id={0}，如图 6-24 所示。

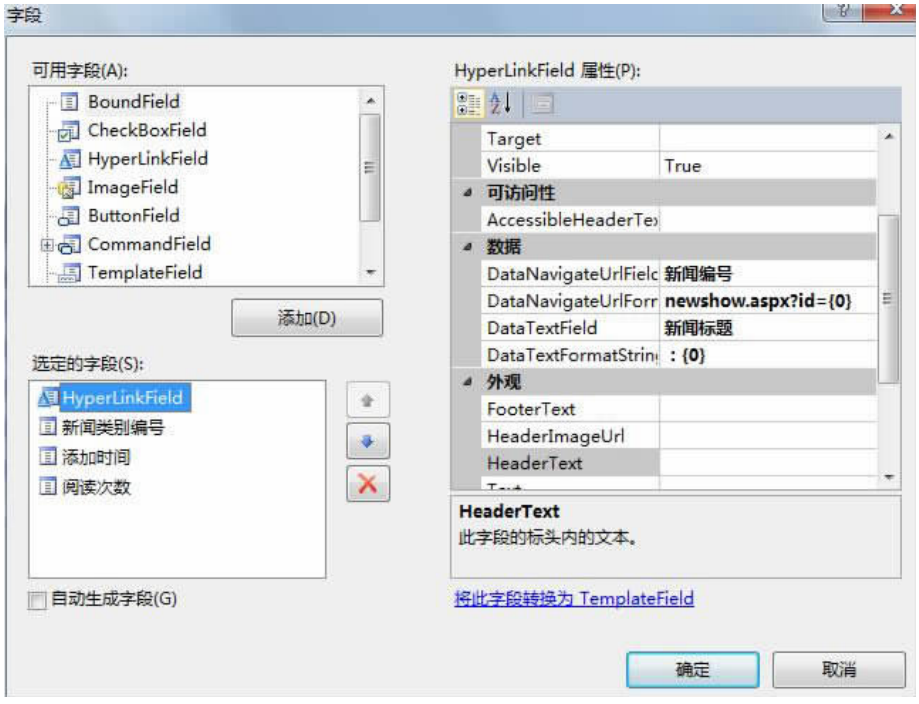

图 6-24　创建超链接

(6) 编写事件。

```
protected void Page_Load(object sender, EventArgs e)
    {
        string strsql="select  * from 新闻表 where 新闻编号=" + Request.Params["id"].ToString();
        DataTable dt = DBHelper.getdt(strsql);
        DataList1.DataSource = dt;
        DataList1.DataBind();
        //阅读次数加 1
        strsql = "update 新闻表 set 阅读次数=阅读次数+1 where 新闻编号=" + Request.Params
["id"].ToString();
        DBHelper.ExecuteCommand(strsql);
    }
```

(7) 调试运行,得出如图 6-25、图 6-26 所示的效果图。

图 6-25　newlist. aspx 运行效果图

图 6-26　newshow.aspx 运行效果图

6.5.3　相关知识

数据绑定

数据绑定要包含在<%#.....%>中。

Eval()：用于单向(只读)绑定。

<%# Eval("新闻标题")%>

Bind()：用于双向(可更新)绑定。

6.5.4　实训 10

参照操作步骤，创建新闻内容显示页面。

模块 6　创建产品列表页面

6.6.1　页面功能

设计一个产品列表页面，显示产品名称、产品图片、产品单价、产品类别等信息；该页面既能显示全部产品，又能分类显示产品。

根据需求，编写一条 SQL 语句，要求能够进行无条件查询和有条件查询。要达到这个目的，只能采用模糊查询 like，如：

```
select a.*,b.产品类别 from 产品表 a,产品类别表 b where a.产品类别编号 like '" + 参数 + "' and a.产品类别编号=b.产品类别编号;
```

这也是带有参数的查询。该参数为产品类别编号。

设计步骤如下：

(1) 设计 prolist.aspx 页面。

(2) 在 Web.sitemap 文件中链接 prolist.aspx 页面，同时将产品类别编号参数传给

prolist. aspx 页面。

(3) prolist. aspx 页面接收首页传来的参数,作为查询条件值。

(4) 编写 SQL 语句,查询产品表中产品类别编号匹配此参数的记录。

(5) 调用类的方法执行 SQL 语句,并接收执行的结果。

(6) 绑定数据。

6.6.2 操作步骤

(1) 设计界面。向网站添加一个 prolist. aspx 页面,选择母版页 MasterPage. master,在空白处插入一个 2 行 1 列的表格,设置其 align 属性为 center、边框属性为 1。

(2) 第 1 行样式设置为:

```
<td style="font-size:9pt; background-image: url(image/t_bg01.gif); color: #ffffff;
                text-align: left">::公司产品</td>
```

(3) 向表格第 2 行加入一个 DataList 控件,设置其属性:RepeatColumns = 2 RepeatDirection=Horizontal

点击其右上角箭头,进入 DataList 任务栏,点击“编辑模板”,此时进入“DataList_项模板”。

(4) 在 ItemTemplate 模板中添加一个 2 行 2 列的表格,设置其边框属性为 0、align 属性为 center。如图 6-27 所示。

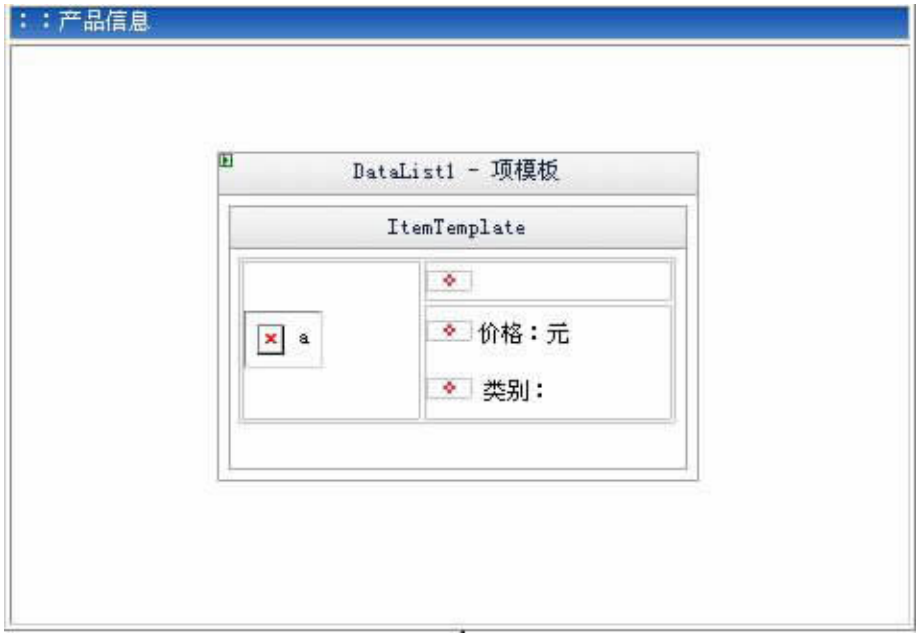

图 6-27 模板设计界面

绑定数据,源代码如下:

```
<table border="0" cellpadding="0" cellspacing="0" style="width: 206px">
<tr>
<td align="left" rowspan="2" style="width: 58px">
<img alt="" src='image/<% # Eval( "产品图片") %>' /> </td>
<td align="left" style="height: 19px">
```

```
<img alt="" src="image/dot_1.gif" />
<strong><%# Eval("产品名称")%></strong></a>
</td>
</tr>
<tr>
<td align="left" style="font-size:9pt">
<img alt="" src="image/dot_1.gif" style="width: 26px" />
单价:<%# Eval("产品单价")%> 元<br />
<br />
<img alt="c" src="image/dot_1.gif" />类别:<a href='prolist.aspx? id=<%# Eval("产品类别编号")%>'><%# Eval("产品类别")%></a>
</td>
</tr>
</table>
```

结束模板编辑,得出如图 6-28 所示的效果图。

图 6-28　模板设计效果图

(5) 创建超链接。打开 Web.sitemap 文件,为“产品信息”编写 url。

```
<siteMapNode url="prolist.aspx? id= %" title="产品信息"  description="" />
```

(6) 编写事件。

```
protected void Page_Load(object sender, EventArgs e)
    {
        string strsql = "select a.*,b.产品类别 from 产品表 a,产品类别表 b where a.产品类别编号 like '" + Request.Params["id"].ToString() + "' and  a.产品类别编号=b.产品类别编号";
        DataTable dt = DBHelper.getdt(strsql);
        DataList1.DataSource = dt;
        DataList1.DataBind();
    }
```

(7) 调试运行,得出如图 6-29 所示的效果图。

图 6-29 prolist. aspx 运行效果图

6.6.3 实训 11

参照操作步骤，创建产品列表页面。

模块 7 产品内容显示页面

6.7.1 页面功能

产品内容显示页面主要用来显示产品的详细内容，包括产品名称、产品图片、产品单价、产品介绍等。通过这个页面，用户可以了解产品的详细信息，用户只需要点击产品列表页面中的产品名称，即可连接到产品内容显示页面，来获取产品的详细信息。

6.7.2 操作步骤

(1) 设计界面。向网站添加一个 proshow. aspx 页面，选择母版页 MasterPage. master，在空白处插入一个 2 行 1 列的表格，设置其 align 属性为 center、边框属性为 1。

(2) 第 1 行样式设置为：

```
<td style="font-size:9pt; background-image: url(image/t_bg01.gif); color: #ffffff;
        text-align: left">::产品介绍</td>
```

(3) 向表格第 2 行加入一个 DataList 控件，设置其属性为：RepeatColumns = 1 RepeatDirection=Horizontal。

点击其右上角箭头，进入 DataList 任务栏，点击“编辑模板”，此时进入“DataList-项模板”。

(4) 在 ItemTemplate 模板中添加一个 5 行 1 列的表格，设置其边框属性为 0、align 属性为 center、Width=100%。设置如图 6-30 所示。

图 6-30　模板设计界面

绑定数据，源代码如下：

```
<table border="0" style="width: 100%">
<tr>
<td align="left" style="width: 100%">
产品名称:<%# Eval("产品名称")%></td>
</tr>
<tr>
<td align="left" style="width: 100%; text-align: center">
<img src='image/<%# Eval("产品图片")%>' /></td>
</tr>
<tr>
<td align="left" style="width: 100%">
产品介绍:</td>
</tr>
<tr>
<td align="left" style="width: 100%">
<%# Eval("产品介绍")%></td>
</tr>
<tr>
<td align="left" style="width: 100%">
单价:<%# Eval("产品单价")%>元              
购买 &gt;&gt;</td>
</tr>
</table>
```

结束模板编辑，得出如图 6-31 所示效果图。

图 6-31　模板设计效果图

（5）编写事件。

```
protected void Page_Load(object sender, EventArgs e)
    {
        string strsql = "select * from 产品表 where 产品编号=" + Request.QueryString["id"];
        DataTable dt = DBHelper.getdt(strsql);
        DataList1.DataSource = dt;
        DataList1.DataBind();
    }
```

（6）创建超链接。打开 prolist.aspx 页面，为“产品名称”创建超链接。

```
<a href='proshow.aspx? id=<%# Eval("产品编号")%>'><strong><%# Eval("产品名称")%></strong></a>
```

（7）调试运行。先运行 prolist.aspx 页面，如图 6-32 所示，点击“新梦 D500”，立即显示该产品的详细信息，如图 6-33 所示。

图 6-32　prolist.aspx 页面运行效果图

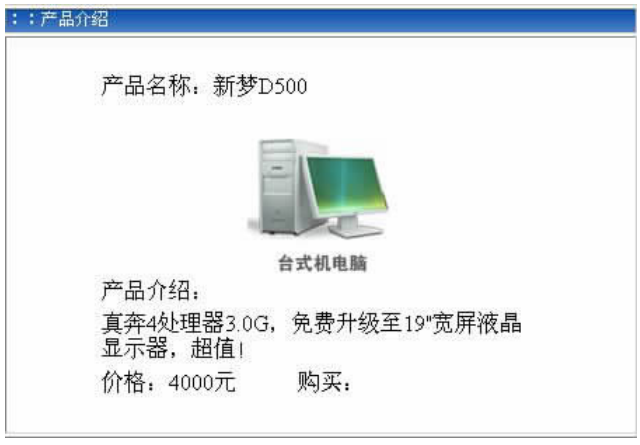

图 6-33　proshow. aspx 页面运行效果图

6.7.3　实训 12

参照操作步骤，创建产品内容显示页面。

模块 8　修改主页面

6.8.1　操作步骤

（1）把主页面设置如图 6-34 所示。

图 6-34　修改后的前台主页面

（2）编写事件。

```
string strsql;
DataTable dt = new DataTable();
protected void Page_Load(object sender, EventArgs e)
    {
        strsql = "select top 6 * from 新闻表 where 新闻类别编号='1' order by 新闻编号 desc";
         GridView1.DataSource = DBHelper.getdt(strsql);
         GridView1.DataBind();
         strsql = "select top 6 * from 新闻表 where 新闻类别编号='2' order by 新闻编号 desc";
         GridView2.DataSource = DBHelper.getdt(strsql);
         GridView2.DataBind();
         strsql = "select top4 a.*,b.产品类别 from 产品表 a,产品类别表 b where a.产品类别编号
=b.产品类别编号";
         DataList1.DataSource = DBHelper.getdt(strsql);
         DataList1.DataBind();
    }
```

（3）创建超链接。打开 Web. sitemap 文件，为“首页”编写 url。

```
<siteMapNode url="Default.aspx" title="首页"  description="" />
```

（4）运行效果如图 6-35 所示。

图 6-35　前台首页运行效果图

6.8.2　实训 13

修改前台首页，修改效果如图 6-35 所示。

模块 9　用 AJAX 设计用户登录控件

教学目标：

- 理解什么是 Ajax；
- 理解 Ajax 的使用技术；
- 掌握 ASP.NET 4.0 AJAX 控件的使用方法；
- 掌握页面间的数据传递技术；
- 会用 Session 对象保存登录用户信息。

6.9.1　控件功能

验证用户身份，检测用户名和密码是否输入合法；如果是合法用户（即用户表的用户），在 Session 中保存用户名和密码，重新定向到首页；否则，在页面输出“您输入的用户名或密码不正确！”。

根据需求，编写一条 SQL 语句，用于查找用户表中用户名和密码分别与输入的用户名和密码相等的记录。

```
string strSQL = "select * from 用户表 where 身份标志=0 and 用户名= '" + txt_name.Text.Trim()
+ "'and 密码='" + txt_pwd.Text + "'";
```

6.9.2　操作步骤

（1）启动 Visual Studio 2010，打开网站，向网站添加一个 Web 用户控件 userlogin.ascx。

（2）向 Web 用户控件加入一个 ScriptManager 控件，然后再添加一个 UpdatePanel 控件。在 UpdatePanel 控件上添加一个 4 行 2 列的表格，设置其边框属性为 0、align 属性为 center。样式设置为：style="text-align：center；width：auto；height：auto"＞；

布局如图 6-36 所示。

图 6-36　用户控件设计图

（3）放置控件到相应的单元格，并设置其属性，见表 6-4。

表 6-4　　控件属性设置

控件 ID	控件类型	属性名	属性值
txt_name	TextBox		
txt_pwd	TextBox	textmode	password
ImageButton1	ImageButton	ImageUrl	smt2_05.gif
ImageButton2	ImageButton	ImageUrl	smt2_02.gif

（4）设置 UpdatePanel 控件的 Trigger 属性，如图 6-37 所示。

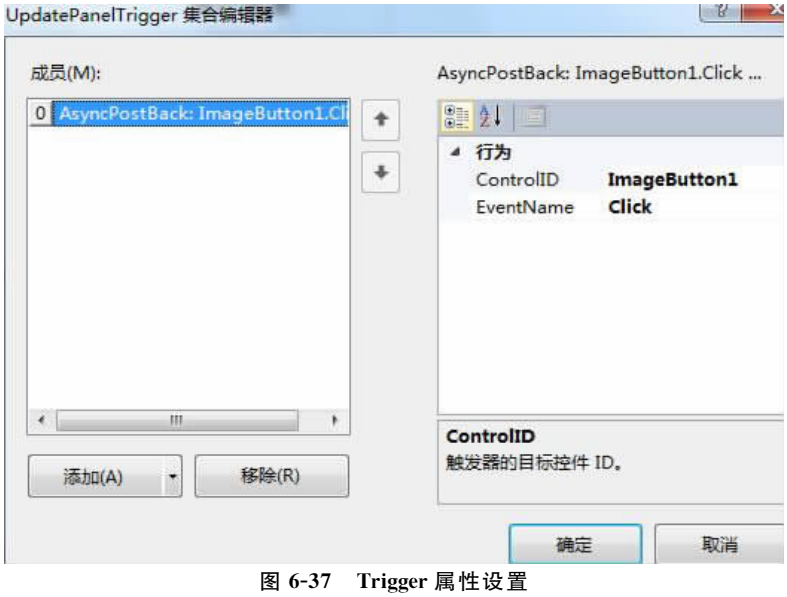

图 6-37　Trigger 属性设置

（5）编写事件。

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {
        string strSQL = "select * from 用户表 where 身份标志=0 and 用户名= '" + txt_name.
Text.Trim() + "'and 密码='" + this.txt_pwd.Text + "'";
        DataTable dt = DBHelper.getdt(strSQL);
        if (dt.Rows.Count > 0)
        {
            Session["name"] = dt.Rows[0]["用户名"].ToString().Trim();
            Session["pwd"] = dt.Rows[0]["密码"].ToString().Trim();
            ScriptManager.RegisterClientScriptBlock(ImageButton1, typeof(Button), "invalidAlert",
"alert('登录成功');", true);
        }
        else
        {
            ScriptManager.RegisterClientScriptBlock(ImageButton1, typeof(Button), "validAlert",
"alert('您输入的用户名或密码不正确');", true);
        }
    }
```

把用户控件 userlogin.ascx 拖放到主页面，如图 6-38 所示。

图 6-38　添加了用户控件的主页面

运行效果如图 6-39 所示。

图 6-39　主页面运行效果图

6.9.3 相关知识

Ajax 简介

Ajax 是 Asynchronous JavaScript＋XML(异步 JavaScript 和 XML)的简写形式,是综合异步通信、JavaScript 以及 XML 等多种网络技术的新的编程方式。可以在不重新加载整个网页的情况下,对网页的某部分进行更新。

如果从用户看到的实际效果来看,也可以形象地称之为无页面刷新。

Ajax 的优点:

- 减轻服务器的负担;
- 不对整页页面刷新;
- 把以前的一些由服务器承担的工作转移到客户端处理;
- 基于标准化的并被广泛支持的技术,不需要插件,也不需要下载小程序。

AJAX 有以下几个控件:

(1) ScriptManger(脚本管理员)控件

ScriptManager 控件是 ASP. NET 中 AJAX 功能的中心,该控件可以管理一个页面上的所有 ASP. NET AJAX 资源。ScriptManager 只能在页面中使用一次,并且必须出现在所有 ASP. NET AJAX 控件之前,ScriptManager 控件用来进行该页面的全局管理。

(2) UpdatePanel(更新区域)控件

UpdatePanel 控件可生成功能丰富的、以客户端为中心的 Web 应用程序。通过使用 UpdatePanel 控件,可以刷新页的选定部分,而不是使用回发刷新整个页面,这称为“部分页更新”。包含一个 ScriptManager 控件和一个或多个 UpdatePanel 控件的 ASP. NET 网页可自动参与部分页更新,而不需要自定义客户端脚本。

(3) UpdateProgress(更新进度)控件

当服务器端与客户端进行异步通信时,可以使用 UpdateProgress 控件告诉用户现在正在执行中。

(4) Timer(时间)控件

AJAX 提供了一个 Timer 控件,用于按定义的时间间隔执行回发。如果将 Timer 控件用于 UpdatePanel 控件,则可以按定义的时间间隔启用部分页更新。

6.9.4 实训 14

参照操作步骤,完成下列任务:

(1) 创建 Web 用户控件 userlogin. ascx,包含 Ajax 技术;

(2) 把用户控件 userlogin. ascx 拖放到主页面。

模块 10 创建修改密码页面

教学目标:

- 掌握页面间的数据传递技术;

- 会取出 Session 对象的值；
- 会使用 Response 对象和 Request 对象控制页面的输入输出；
- 了解验证控件的使用。

6.10.1　操作步骤

（1）设计界面。向网站添加一个 xgmm.aspx 页面，选择母版页 MasterPage.master，在空白处插入一个 5 行 2 列的表格，设置其边框属性为 0、align 属性为 center，设计界面如图 6-40 所示。

图 6-40　修改密码页面设计

（2）放置控件到相应的单元格，并设置其属性，见表 6-5。

表 6-5　　控件属性设置

控件 ID	控件类型	属性名	属性值
Txt1-2	TextBox	Readonly	true
Txt3	TextBox	TextMode	Password
RequiredFieldValidator1		ErrorMessage ControlToValidate	输入新密码 txt3
Button1	Button	Text	确认

（3）创建超链接。打开 Web.sitemap 文件，为“修改密码”编写 url。

```
<siteMapNode url="xgmm.aspx" title="修改密码"  description="" />
```

(4) 编写事件。

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["name"] == null)
        {
            Response.Write("<script>alert(\"请登录! \");</script>");
            Response.Redirect("Default.aspx");
        }
        if (! Page.IsPostBack)
        {
            this.txt1.Text = Session["name"].ToString();
            this.txt2.Text = Session["pwd"].ToString();
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        this.strSQL = "update 用户表 set 密码='" + this.txt3.Text.Trim() + "' where 用户名='"
+ this.txt1.Text.Trim() + "'";
        DBHelper.ExecuteCommand(strsql);
        Response.Write("修改密码成功!");
        Response.Redirect("Default.aspx")
    }
```

6.10.2 相关知识

状态管理

HTTP 是一种不保持状态的通信协议。这就是说,在网站系统中,每次浏览器与服务器之间的连接都是暂时的。当浏览器与服务器之间的一次会话结束,它们之间的连接也就自动断开了,下一次会话与本次连接无关,两次连接之间不存在任何联系。但是在网站中有时需要保持某些状态,此时,可以使用状态管理功能。ASP. NET 中状态保持通过三个对象来实现,如图 6-41所示。

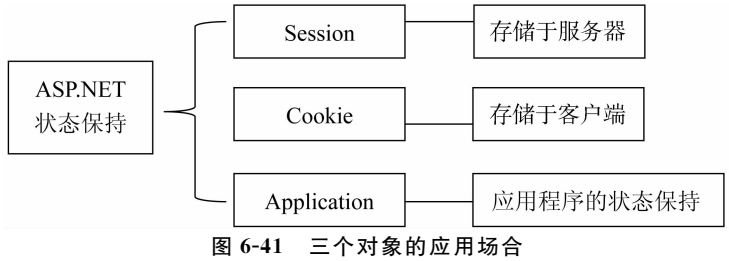

图 6-41 三个对象的应用场合

以上三个对象实际上可以看成是在一个 Web 应用程序(一个站点)的不同页面之间保持数据、传递数据的变量。

Session

Session 用于什么场合

```
Session 保持当前用户状态信息，常用于用户登录、购物车等。
保存会话状态
Session["Message"] = "MyMsg";
取出 Session 对象
string MyVar= Session["Message"].ToString();
```

6.10.3　实训 15

参照操作步骤，创建修改密码页面。

模块 11　创建用户注册页面

6.11.1　页面功能

用户注册页面功能就是向用户表添加记录，当用户进入注册页面后，可以输入用户的相关信息，然后单击“提交”按钮，当用户信息成功保存到数据库后，会弹出“注册成功的”的提示信息。此外，还要验证用户输入的信息，当信息不符合系统要求时，会自动给出提示信息。因此，注册页面的功能就是将用户信息保存到数据库和对用户信息进行验证。

6.11.2　操作步骤

（1）设计界面。向网站添加一个 useradd. aspx 页面，选择母版页 MasterPage. master，在空白处插入一个 9 行 2 列的表格，设置其边框属性为 0、align 属性为 center，合并单元格，如图 6-42 所示。

图 6-42　用户注册界面

（2）放置控件到相应的单元格，并设置其属性，见表 6-6。

表 6-6　　控件的属性设计

控件 ID	控件类型	属性名	属性值
Txt1-7	TextBox		
Button1	Button	Text	提交
RequiredFieldValidator1		ControlToValidate	txt1
		ErrorMessage	用户名不能为空
CompareValidator1		ControlToCompare ControlToValidate ErrorMessage	xt2 txt3 密码不一致
RegularExpression Validator1		ValidationExpression ControlToValidate ErrorMessage	\d{6} txt7 邮政编号无效

(3) 编写事件。

```
string strsql;
DataTable dt = new DataTable();
    protected void Page_Load(object sender, EventArgs e)
    {
        if (! this.Page.IsPostBack)
        {
            csh();
        }
    }
    private void csh()
    {
        txt1.Text = "";
        txt2.Text = "";
        txt3.Text = "";
        txt4.Text = "";
        txt5.Text = "";
        txt6.Text = "";
        txt7.Text = "";
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
            strsql = "insert into 用户表 (用户名,密码,真实姓名,电话,地址,邮编) values ('" + txt1.Text + "','" + txt2.Text + "','" + txt4.Text + "','" + txt5.Text + "','" + txt6.Text + "','" + txt7.Text + "')";
            DBHelper.ExecuteCommand(strsql);
            Response.Write("<script>alert(\"注册成功! \");</script>");
```

```
            Session["name"] = txt1.Text;
            Response.Redirect("Default.aspx");
        }
    protected void txt1_TextChanged(object sender, EventArgs e)
        {
            string strSQL = "select * from 用户表 where 用户名= '" + txt1.Text.Trim() + "'";
            DataTable dt = DBHelper.getdt(strSQL);
            if (dt.Rows.Count > 0)
            {
                Response.Write("<script>alert(\"用户名已存在,重输\");</script>");
            }
        }
```

(4) 设置 userlogin.ascx 中“免费注册”按钮的 PostBackUrl 属性为 useradd.aspx。

6.11.3　相关知识

验证控件简介

(1) 必需项验证控件 RequiredFieldValidator。该控件用来验证用户是否对某个 Web 页面中的字段进行了编辑,属性主要包括:

- ControlToValidate 属性:需要验证的控件 id;
- ErrorMessage 属性:验证不通过时显示的错误信息。

(2) 比较验证控件 CompareValidator。该控件用于将用户输入的值和其他控件的值或者常数进行比较,主要属性有:

- ControlToCompare 属性:用于比较的控件 id;
- ControlToValidate 属性:需要验证的控件 id;
- ErrorMessage 属性:验证无效时的错误信息。

(3) 正则表达式验证控件 RegularExpressionValidator。该控件用于验证相关输入控件的值是否匹配正则表达式指定的模式,一般通过 IDE 中的“正则表达式编辑器”来设置正则表达式。

6.11.4　实训 16

参照操作步骤,创建用户注册页面。

模块 12　创建产品订单页面

6.12.1　页面功能

订单页面应具备如下功能:

- 验证顾客是否登录,如果未登录,提示先登录;
- 显示顾客选购的产品信息,供顾客修改订购数量;

- 把产品插入到订单表；
- 修改该产品的销售数量和库存数量。

6.12.2 操作步骤

（1）设计界面。向网站添加一个 orderadd.aspx 页面，选择母版页 MasterPage.master，在空白处插入一个 6 行 2 列的表格，设置其边框属性为 0、align 属性为 center，如图 6-43 所示。

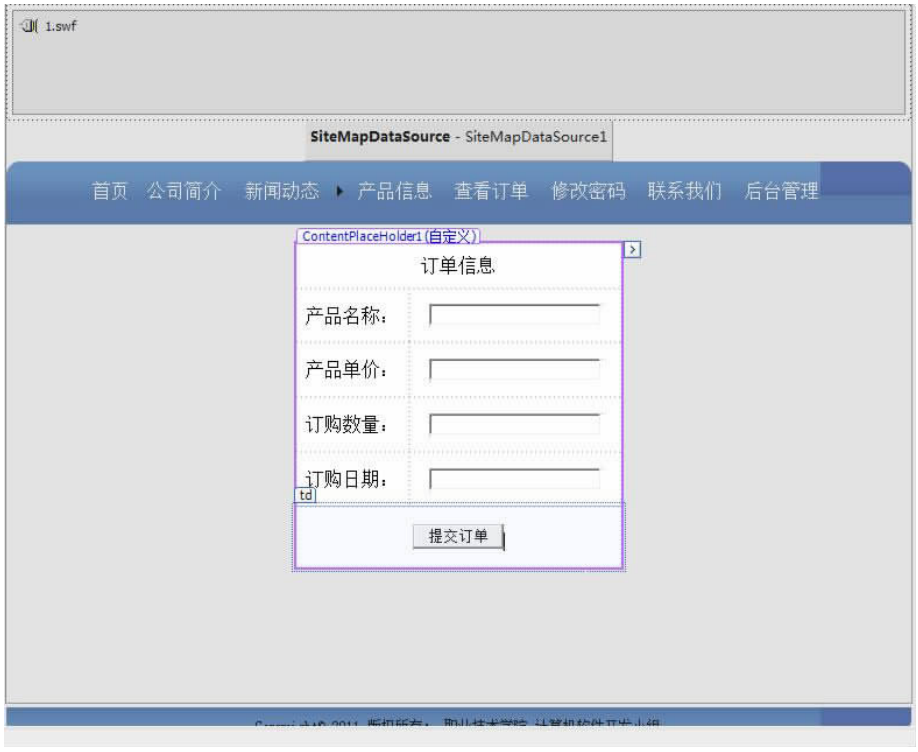

图 6-43 产品订单界面

（2）放置控件到相应的单元格，并设置其属性，见表 6-7。

表 6-7 控件的属性

控件 ID	控件类型	属性名	属性值
Txt1-4	TextBox		
Button1	Button	Text	提交订单

（3）编写事件。

```
string cpbh, strsql;
DataTable dt = new DataTable();
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["name"] == null)
        {
            Response.Write("<script>alert('尚未登录,请登录！');location='Default.aspx';</
script>");
```

```
            //当前窗口的位置转移到 Default.aspx 页面
            return;
        }
        if (! Page.IsPostBack)
        {
            cpbh = Request.Params["id"].ToString();
            strsql = "select * from 产品表 where 产品编号 =" + cpbh;
            dt = DBHelper.getdt(strsql);
            txt1.Text = dt.Rows[0]["产品名称"].ToString();
            txt2.Text = Convert.ToString(dt.Rows[0]["产品单价"]);
            txt3.Text = "1";
            txt4.Text = DateTime.Today.ToShortDateString();
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string cpmc, dj, shl, yhm, rq;
        cpmc = txt1.Text;
        dj = txt2.Text;
        shl = txt3.Text;
        yhm = Session["name"].ToString();
        rq = txt4.Text;
        strsql = "insert into 订单表(产品编号,产品名称,产品单价,订购数量,用户名,订购日期)
values ('" + cpbh + "','" + cpmc + "','" + dj + "','" + shl + "','" + yhm + "','" + rq + "')";
        this.strsql += "update 产品表 set 销售数量=销售数量+'" + shl + "',库存数量=库存数
量-'" + shl + "' where 产品编号='" + cpbh + "'";
        DBHelper.ExecuteCommand(strsql);
        Response.Write("<script>alert(\"提交成功! \");</script>");
        Response.Redirect("prolist.aspx? id= % ");
    }
```

(4) 创建超链接。打开 proshow.aspx 页面,为“购买”创建超链接。

```
<a href='orderadd.aspx? id=<% # Eval("产品编号") %> '>购买</a>
```

6.12.3　实训 17

参照操作步骤,创建产品订单页面。

模块 13　查看订单页面

6.13.1　页面功能

查看订单页面应具备如下功能:

- 显示登录用户的订单信息；
- 删除订单；
- 结账。

设计步骤：

- 设计 ordercx. aspx 页面；
- 对 ordercx. aspx 页面建立超链接；
- 编写页面 Page_Load 事件；
- 编写 GridView1_RowDeleting 事件；
- 编写 Button1_Click 事件。

6.13.2 操作步骤

(1) 设计界面。向网站添加一个 ordercx. aspx 页面，选择母版页 MasterPage. master，在空白处插入一个 3 行 1 列的表格，设置其边框属性为 0、align 属性为 center，在表格第 2 行放置一个 Button 按钮和一个 TextBox1 文本框，第 3 行放一个 GridView 控件，如图 6-44 所示。

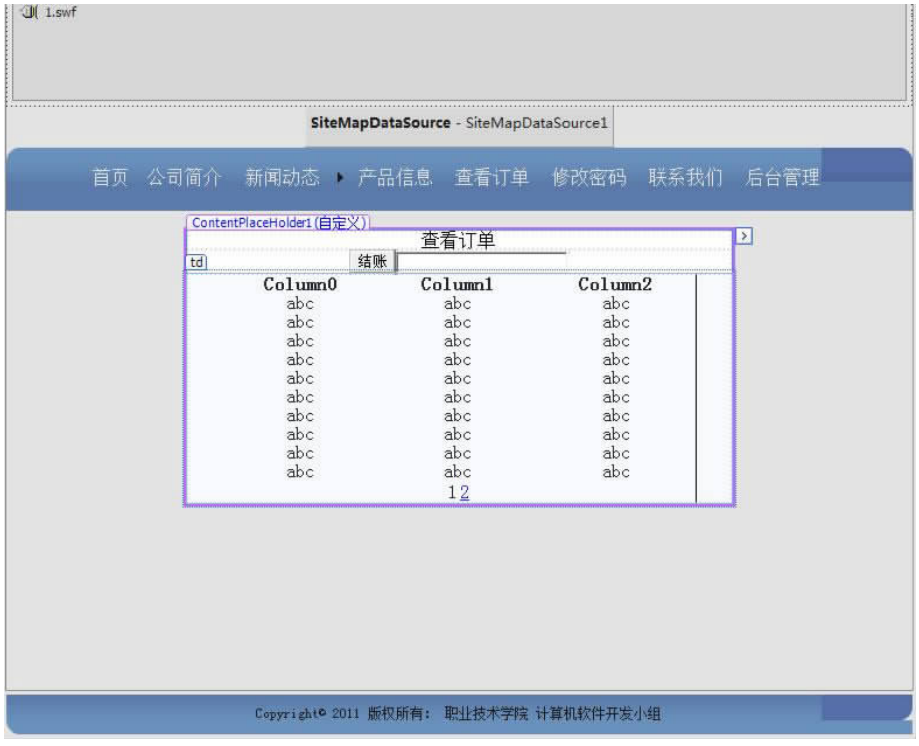

图 6-44 查看订单界面

(2) 设置控件的属性，见表 6-8。

表 6-8 GridView1 控件属性

控件 ID	控件类型	属性名	属性值
GridView1	GridView	DataKeyNames	订单编号
		Allowpaging	true

（3）点击右上角箭头，进入 GridView 任务栏，点击“编辑列”，进入“字段”对话框，在可用字段列表框中添加一个 commandfield 字段到选定的字段列表框，如图 6-45 所示。

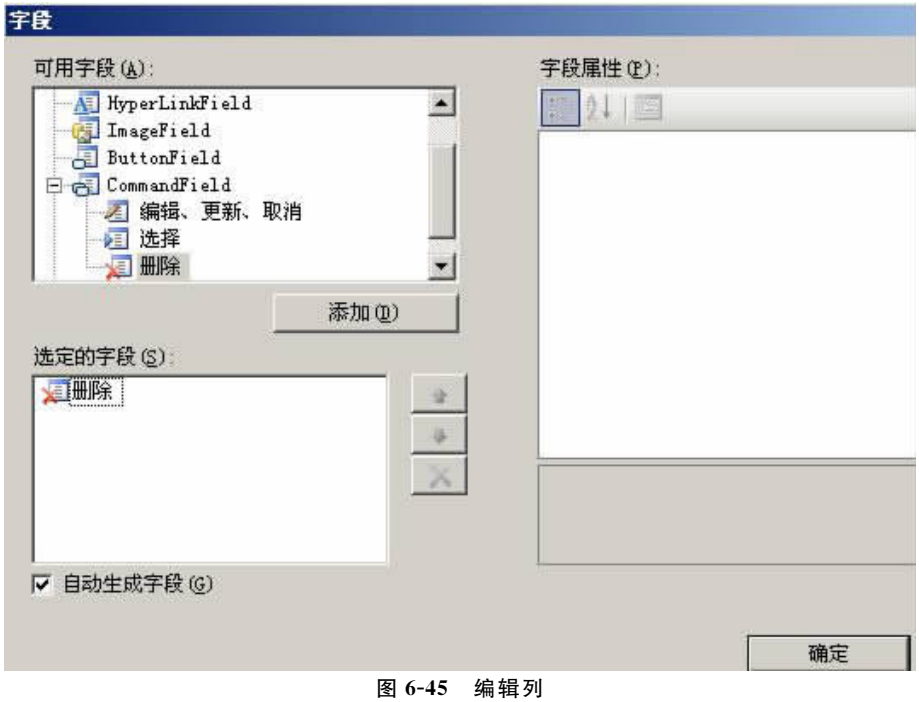

图 6-45　编辑列

按“确定”按钮，得出如图 6-46 所示的界面。

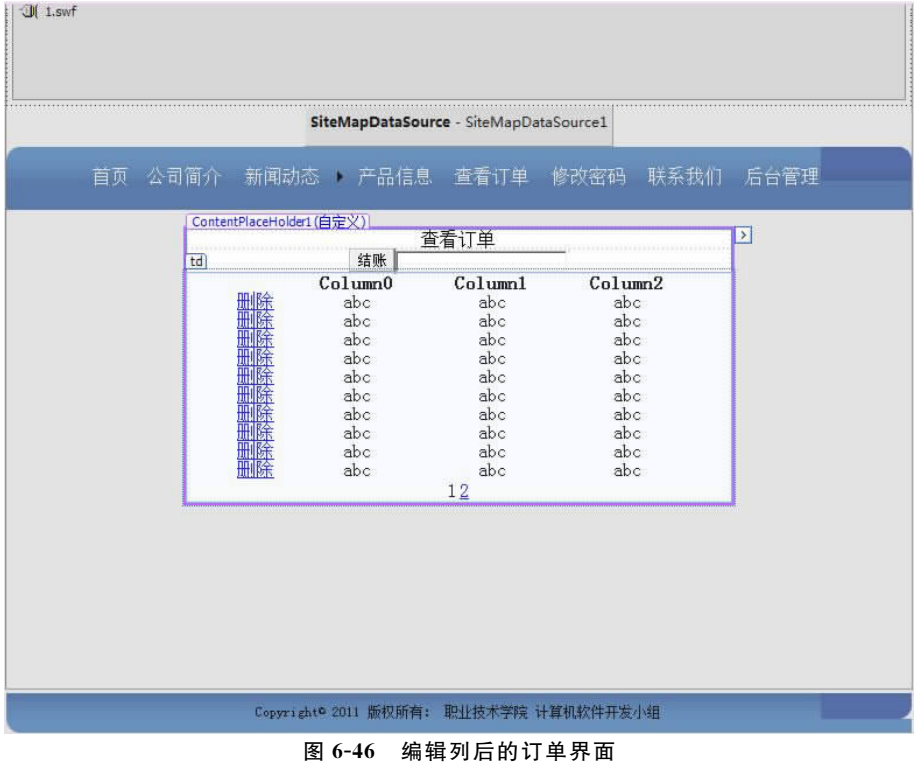

图 6-46　编辑列后的订单界面

(4) 编写事件。

```
String strsql;
   DataTable dt = new DataTable();
protected void Page_Load(object sender, EventArgs e)
    {
        if (! Page.IsPostBack)
        {
            bindgrig();
        }
    }
    void bindgrig()
    {
        strsql = "select * from 订单表 where 用户名='" + Session["name"] + "'";
        dt = DBHelper.getdt(strsql);
        GridView1.DataSource = dt;
        GridView1.DataBind();
    }
    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        strsql = "delete from 订单表 where 订单编号='" + GridView1.DataKeys[e.RowIndex].
Value.ToString() + "'";
        DBHelper.ExecuteCommand(strsql);
        bindgrig();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        double sum = 0.0;
        int hs = GridView1.Rows.Count;
        for (int ii = 0; ii < hs; ii++)
        {
            string t4 = GridView1.Rows[ii].Cells[4].Text.Trim().ToString();
            string t5 = GridView1.Rows[ii].Cells[5].Text.Trim().ToString();
            sum = sum + double.Parse(t4) * double.Parse(t5);
        }
        TextBox1.Text = sum.ToString();
        }
```

(5) 创建超链接。打开 Web.sitemap 文件,为“查看订单”编写 url。

```
<siteMapNode url="ordercx.aspx" title="查看订单"  description="" />
```

6.13.3 实训 18

参照操作步骤,创建查看订单页面。

模块 14 创建简介与联系页面

本模块属于静态页面,不用讲解,由学生自行完成。

6.14.1 公司简介页面

(1) 设计界面如图 6-47 所示。

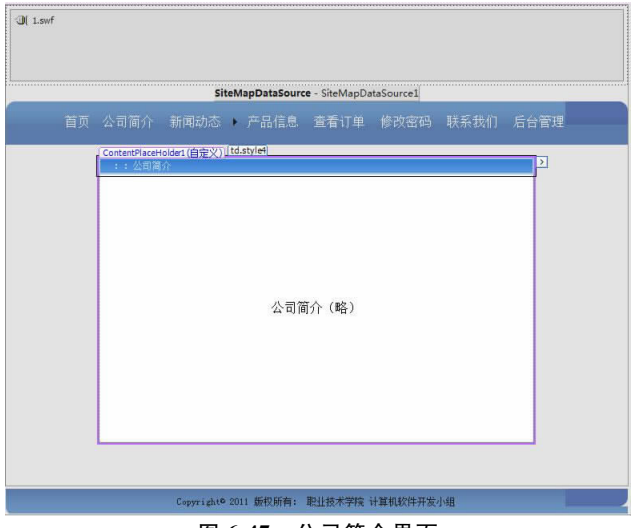

图 6-47　公司简介界面

（2）创建超链接。打开 Web.sitemap 文件，为“公司简介”编写 url。

```
<siteMapNode url="gsjj.aspx" title="公司简介"  description="" />
```

6.14.2　联系我们页面

（1）设计界面如图 6-48 所示。

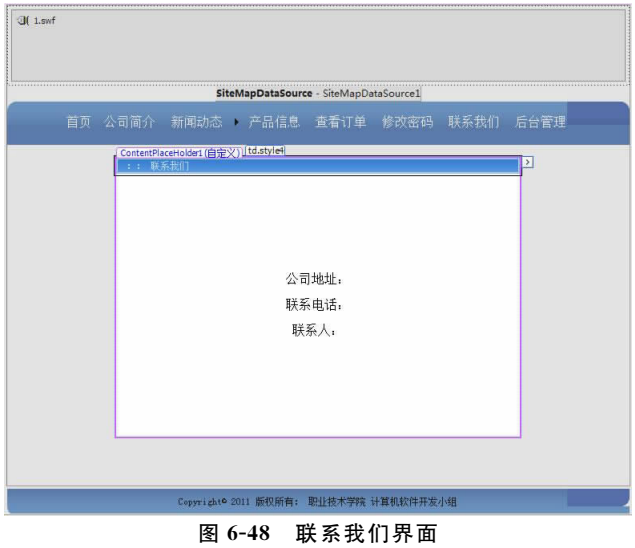

图 6-48　联系我们界面

（2）创建超链接。打开 Web.sitemap 文件，为“联系我们”编写 url。

```
<siteMapNode url="lxwm.aspx" title="联系我们"  description="" />
```

6.14.3　实训 19

参照操作步骤，完成下列任务：

（1）创建公司简介页面；

（2）创建联系我们页面。