第 3 章　HTML 控件和 Web 服务器控件

工作任务

- 网上商城界面的实现；
- 网上商城新产品上架功能的实现。

3.1　HTML 控件

3.1.1　几个对比实验

将如下程序写入记事本后保存为 test1.html 格式的文件。

```
<!--传统的 HTML 静态网页写作方式-->
<html>
<body>
<a href="http://www.microsoft.com">请单击这里</a>
</body>
</html>
```

然后双击保存的页面，页面中将显示一个简单的超链接。

将如下程序代码写入记事本后保存为 test2.asp 格式的文件。

```
<!--为了动态的设定标注的属性,必须在标注中插入许多程序-->
<html>
<%
strAddress="http://www.microsoft.com"
%>
<a href=<%=strAddress%>>请单击这里</a>
</html>
```

然后在 IIS 下浏览页面，也可以看到同样的超链接。

在 Visual Studio 中建立一个页面 test3.aspx，并且将如下代码输入页面 test3.aspx 的 form 中。

```
<!--ASP.NET 的 HTML 控件可以利用程序直接控制-->
<html>
<a id="Anchor1" runat="Server">请单击这里</a>
</html>
```

然后打开后置代码页 test3.aspx.cs，在 Page_Load 方法中输入如下代码：

```
Anchor1.Href="http://www.microsoft.com"
```

执行页面可以看到相同的超链接。

实验结果分析

传统的 HTML 标注无法利用程序直接控制，这是因为 HTML 标注当初设计时并没有彻底对象化；所以如果要动态地利用程序设定标注的属性，必须插入 ASP 程序。以前的 ASP 程序代码非常杂乱，常常会看到标注中插入许多叙述性的程序，这样会导致程序代码在维护以及阅读上的困难（如文件 test2.asp）。

ASP.NET 为了要解决这种杂乱无章的程序写作风格，便将 HTML 标注对象化而产生出 HTML 控件，HTML 控件可以让程序直接控制并设定其属性（如文件 test3.aspx）。

3.1.2 基本概念

HTML 控件是 ASP.NET 所提供的控件（也称为 Server 控件），是在服务器端执行的组件，可以产生标准的 HTML 文件。一般来说，标准的 HTML 标签无法动态控制其属性、使用方法和接收事件，必须使用其他的程序语言来控制标签，这对于使用 ASP 程序设计来说很不方便，而且会使 ASP 程序比较杂乱。ASP.NET 在这方面开发了新的技术，即将 HTML 标签对象化，使程序（如 Visual Basic.NET、C＃等）可以直接控制 HTML 标签，对象化后的 HTML 标签称为 HTML 控件。

3.1.3 HTML 控件的应用

3.1.3.1 网上商城产品上架功能的实现

功能：商城产品上架的实现。

使用的数据库：EShop。

使用的数据表：ProdInfo。

实现步骤如下：

本模块用到的页面控件及属性如表 3-1 所示。

表 3-1

控件类型	ID	TextMode	Text
DropDownList	ddlType		
TextBox	tbProdname	Singleline	
TextBox	tbPrice	Singleline	
TextBox	tbVipPrice	Singleline	

续表

控件类型	ID	TextMode	Text
Input(File)	PhotoPic		
TextBox	tbfactory	Singleline	
Button	btnUpload		上传
TextBox	tbBand	Singleline	
TextBox	tbDescription	MultiLine	
Button	btnAdd		产品上架
Label	lbmessage		
Image	imgPhoto		

(1) 查看给出的数据库中的数据表 ProdInfo，界面如图 3-1 所示。

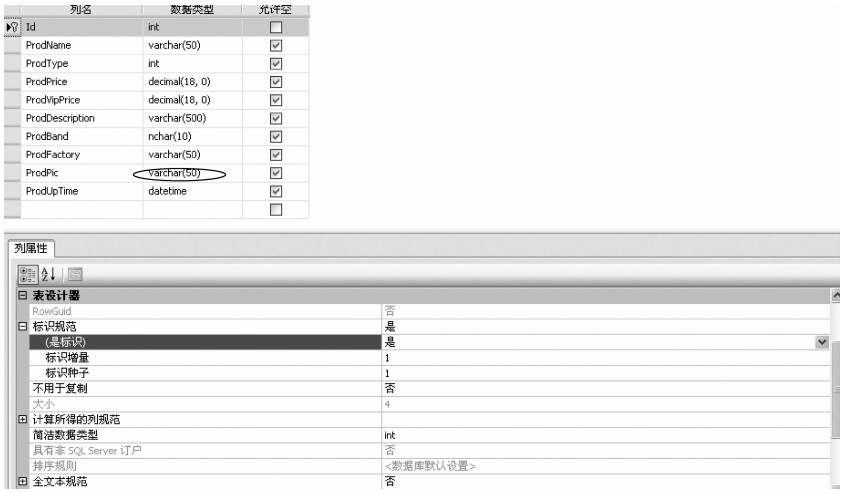

图　3-1

(2) 新建一个页面 AddProduct.aspx，并在其中拖入控件，界面如图 3-2 所示。

(3) 为每个控件重新命名其 ID，商品照片上传使用的是 HTML 的 File 控件，并且打开源代码。

在该控件中增加 runat＝“server”之后，在后置代码中才能找到该控件，如图 3-3 所示。

图 3-4 是在 AddProduct.aspx.cs 页面中看到的内容。

(4) 双击“上传”按钮后添加如下代码：

图　3-2

```
protected void btnUpload_Click(object
sender, EventArgs e)
    {
        string strFileName=FilePic.
```

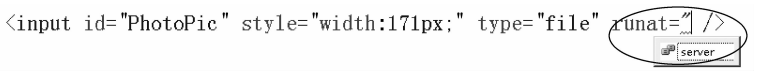

图 3-3

图 3-4

```
        PostedFile.FileName;
        string strindex=strFileName.Substring(strFileName.LastIndexOf(".")+1, 3);
        strFileName=DateTime.Now.Year.ToString()+DateTime.Now.Month.ToString()+
        DateTime.Now.Day.ToString()+DateTime.Now.Hour.ToString()+DateTime.
        Now.Minute.ToString()+DateTime.Now.Second.ToString()+"."+strindex;
    //使用年月日时分秒构建图片文件的名称,防止文件名重复并覆盖的现象发生
        ViewState["filename"]=strFileName.ToString();
        string strPath=Server.MapPath("ImgProduct\\");
        FilePic.PostedFile.SaveAs(strPath+strFileName);
        PhotoPic.ImageUrl=strPath+strFileName;
    }
```

注意：产品图片的名字采用当前系统时间的时分秒,目的是为了防止由于用户上传的文件重名而导致图片文件被覆盖,影响图片的正常显示。

另外,在数据库中只保存图片所在的路径,便于访问图片并显示。

在项目中建立文件夹 ImgProduct 用来存放商品的图片,如图 3-5 所示。

(5) 双击“产品上架”按钮后添加如下代码,同时在页面顶端添加对 Cuse 对象的实例化代码。

图 3-5

```
EShop.CommonUse Cuse=new EShop.CommonUse();
protected void btnAdd_Click(object sender, EventArgs e)
    {
        string ProdName=txtProdName.Text;
        string ProdType=ddlProdType.SelectedValue;
        string ProdNum=txtProdNum.Text;
        string ProdPrice=txtProdPrice.Text;
        string ProdIntro=txtProdIntro.Text;
        string ProdVipPrice=txtVipPrice.Text;//获取页面中用户输入控件的数据
        if (ViewState["filename"] !=null)
        {
```

```
            string ProdPic=ViewState["filename"].ToString();
Cuse. Insert ( " ProdName, ProdType, ProdPrice, ProdMemberPrice, ProdIntro,
ProdStoreNum,ProdPic, ProdAddTime", " '" + ProdName +" ', '" + ProdType +" ',"+
ProdPrice+","+ProdVipPrice+",'"+ProdIntro+"','"+ProdNum+"','"+ProdPic+"','"
+DateTime.Now.ToString()+"'", "ProdInfo");
            lbmessage.Text="产品上架成功";
            Response.Redirect("ProductManage.aspx");
        }
        else
        {
            lbmessage.Text="产品图片还没修改上传!";
            PhotoPic.Focus();
        }
    }
```

(6) 打开后置代码添加如下代码,完成对产品分类的数据绑定。

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            bindtype();
                //绑定产品类别表,注意要放到 IsPostBack 中防止回传后取出默认值
        }
    }
    protected void bindtype()
        {
            ddlProdType.DataTextField="CategoriesName";
            ddlProdType.DataValueField="CategoriesName";
            ddlProdType.DataSource=Cuse.GetList("*", "", "Categries");
                //查询出产品分类
            ddlProdType.DataBind();
        }
```

注意:每一个商品都有相应的分类,而分类是已经添加了,存在于数据库里面,现在只需选择该商品对应的分类就可以了。所以可以选择“列表菜单”这个控件,直接显示所有商品分类。

3.1.3.2　知识点详细描述

1. HTML 服务器控件及其功能

HTML 服务器控件就是 HTML 元素,它所公开的对象模型十分紧密地映射到相应控件所呈现的 HTML 元素上,这些元素包含使其自身在服务器上可见并可编程的属性。

HTML 服务器控件提供了以下功能：

（1）可以在服务器上使用面向对象技术对其进行编程的对象模型；

（2）提供一组事件，可以为其编写事件处理程序，事件处理在服务器代码中完成；

（3）在客户端脚本中处理事件的能力；

（4）自动维护控件状态；

（5）与验证控件进行交互，便于验证用户是否在控件中输入了适当的信息；

（6）数据绑定到一个或多个控件属性上；

（7）如果 Web 窗体页显示在支持层叠样式表的浏览器中，则支持 HTML 4.0 样式；

（8）直接可用的自定义属性。

2. HTML 服务器控件层次结构

HTML 服务器控件层次结构如图 3-6 所示。

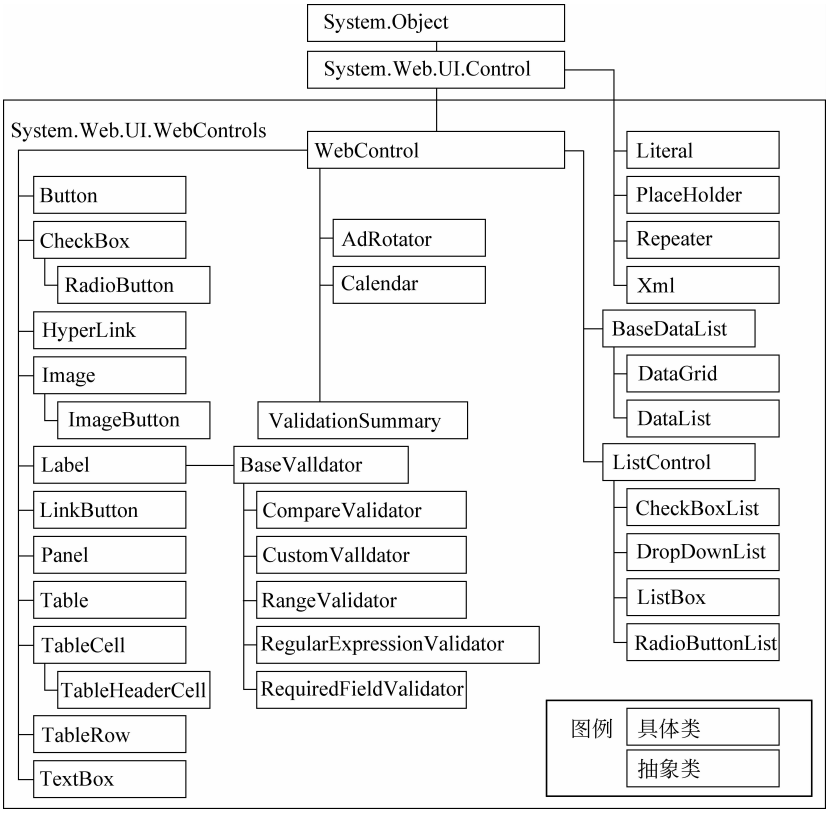

图 3-6

3. 添加 HTML 服务器控件

1）使用 Web 窗体设计器添加 HTML 服务器控件

（1）在“设计”视图下打开要添加控件的 Web 窗体页。

（2）从工具箱的“HTML 选项卡”中双击 HTML 元素，或者将该元素直接拖到窗体页中。

(3) 右击该元素,然后从弹出式菜单中选择“作为服务器控件运行”,将 HTML 元素转换为 HTML 服务器控件。

2) 使用 ASP.NET 语法添加 HTML 服务器控件

(1) 切换到 HTML 视图。

(2) 输入要用做控件的元素的常规 HTML 语法。

(3) 设置该控件的 ID 属性,该值对于该页面应为唯一。

(4) 设置 runat="server"属性,以将 HTML 元素转换为 HTML 服务器控件。

4. HTML 服务器控件的通用属性

1) 所有 HTML 服务器控件共享的属性

- Attributes
- Disabled Style
- TagName
- Visible

2) 所有 HTML 输入控件共享的属性

- Name
- Value
- Type

3) 所有 HTML 容器控件共享的属性

- InnerHtml
- InnerText

5. HTML 服务器控件详解

1) HtmlGenericControl 控件

HtmlGenericControl 控件映射到不由特定的 .NET 框架类表示的 HTML 元素(如<body>和 等)。从工具箱的“HTML 选项卡”中将相应元素拖到页面上,然后用鼠标右击该元素并选择“在服务器端运行”,将生成映射为 元素的 HtmlGenericControl 控件。

声明 HtmlGenericControl 控件的语法如下:

```
<span | body | div | p | span | font | others id="编程标识符" runat="server">
  Contentbetweentags
</span | body | div | p | span | font | others>
```

2) HtmlForm 控件

HtmlForm 控件是 Web 窗体页上的服务器控件容器,该控件映射到 HTML 元素<form>。所有回发给服务器的服务器控件都必须放在 HtmlForm 控件的开始和结束标记之间。在工具箱的“HTML 选项卡”中,没有与 HtmlForm 控件相应的图标。每当在 Visual Studio .NET 中创建一个 Web 窗体时,总会自动创建一个 HtmlForm 容器,其他控件将被包含在该容器内。

声明 HtmlForm 控件的语法如下:

```
<form id="编程标识符" method="post | get" action="srcpageURL" runat="server">
    其他控件、输入表单域等
</form>
```

3）HtmlInputButton 控件

HtmlInputButton 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝button＞、＜input type＝submit＞和＜input type＝reset＞，并允许创建命令按钮、提交按钮或重置按钮。若要创建 HtmlInputButton 控件，首先将工具箱的“HTML 选项卡”中的相应元素拖到页面上，然后用鼠标右击所添加的按钮，并从弹出式菜单中选择“作为服务器控件运行”。

声明 HtmlInputButton 控件的语法如下：

```
<input type="button | submit | reset" id="编程标识符"
    OnServerClick="事件过程名称" runat="server">
```

4）HtmlInputText 控件

HtmlInputText 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝text＞和＜input type＝password＞，并允许创建单行文本框以接收用户输入。

声明 HtmlInputText 控件的语法如下：

```
<input type=text | password id="编程标识符"
maxlength="最大字符数" size="宽度"
value="文本框内容" runat="server">
```

HtmlInputText 控件的主要成员如下：

- Type 属性
- MaxLength 属性
- Size 属性
- Value 属性
- ServerChange 事件

5）HtmlInputRadioButton 控件

HtmlInputRadioButton 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝radio＞并允许在 Web 页上创建单选按钮，使用该控件对 HTML 元素＜input type＝radio＞进行编程。

声明 HtmlInputRadioButton 控件的语法如下：

```
<input type=radio id="编程标识符" checked name="单选按钮组名称" runat="server">
```

HtmlInputRadioButton 控件的主要成员如下：

- Name 属性
- Value 属性
- Checked 属性
- ServerChange 事件

6）HtmlInputCheckBox 控件

HtmlInputCheckBox 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝checkbox＞，并允许创建使用户可以选择 True 或 False 状态的复选框控件。

声明 HtmlInputCheckBox 控件的语法如下：

```
<input type=checkbox id="编程标识符" checked runat="server">
```

HtmlInputCheckBox 控件的主要成员如下：

- Checked 属性
- Value 属性
- ServerChange 事件

7）HtmlInputImage 控件

HtmlInputImage 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝image＞并允许创建显示图像的按钮。

声明 HtmlInputImage 控件的语法如下：

```
<input type="image" id="编程标识符" src="图像路径" value="值" align="对齐方式"
alt="替代文本" width="边框宽度" runat="server">
```

HtmlInputImage 控件的主要成员如下：

- Scr 属性
- Value 属性
- Align 属性
- Alt 属性
- Width 属性
- ServerClick 事件

8）HtmlInputFile 控件

HtmlInputFile 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝file＞并允许将文件上传到服务器。

声明 HtmlInputFile 控件的语法如下：

```
<input type=file id="编程标识符" accept="MIME 编码类型" maxlength="文件路径最大
长度">
```

HtmlInputFile 控件的主要成员如下：

- Accept 属性
- MaxLength 属性
- Size 属性
- PostedFile 属性

9）HtmlInputHidden 控件

HtmlInputHidden 控件创建一个服务器端控件，该控件映射到 HTML 元素＜input type＝hidden＞并允许将信息存储在窗体上不可查看的控件中。

声明 HtmlInputHidden 控件的语法如下：

```
<input type="hidden" id="编程标识符"value="值" runat="server">
```

10）HtmlAnchor 控件

HtmlAnchor 控件创建一个服务器端控件，用于创建移动到该页上的其他位置或其他 Web 页的超链接。

声明 HtmlAnchor 控件的语法如下：

```
<a id="编程标识符" href="链接的 URL" name="书签名称" OnServerClick="事件处理程序名
称" target="框架或窗口名称" title="提示文本" runat="server">链接文本或图像</a>
```

HtmlAnchor 控件的主要成员如下：

- Href 属性
- Name 属性

- Title 属性
- Target 属性
- ServerClick 事件

11) HtmlButton 控件

HtmlButton 创建一个服务器端控件,该控件映射到 HTML 元素<button>,可以用于创建由嵌入的 HTML 元素(甚至其他 Web 窗体控件)构成的按钮。

声明 HtmlButton 控件的语法如下:

```
<button id="编程标识符"
OnServerClick="事件处理过程名称" runat="server">
    按钮文本、图像或控件等
</button>
```

12) HtmlSelect 控件

HtmlSelect 控件创建一个服务器端控件,该控件映射到 HTML 元素<select>并允许创建列表框控件。使用 HtmlSelect 控件对 HTML 元素<select>进行编程。

声明 HtmlSelect 控件的语法如下:

```
<select id="编程标识符" DataSource="数据绑定来源" DataMember="数据成员值"
DataTextField="绑定到选项文本的字段" DataValueField="绑定到选项值的字段"
Multiple Items="选项集合" SelectedIndex="当前选取项的索引" Size="可见的项数"
Value="当前项的值" OnServerChange="事件处理程序" runat="server">
        <option value="值">文本</option>
</select>
```

HtmlSelect 控件的主要成员如下:

- DataSource 属性
- DataMember 属性
- DataTextField 属性
- DataValueField 属性
- Multiple 属性
- Items 属性
- SelectedIndex 属性
- Size 属性
- Value 属性
- DataBind 方法
- ServerChange 事件

13) HtmlTextArea 控件

HtmlTextArea 控件创建一个服务器端控件,该控件映射到 HTML 元素<textarea>并允许创建多行文本框。使用 HtmlTextArea 控件可以对 HTML 元素<textarea>进行编程。

声明 HtmlTextArea 控件的语法如下:

```
<textarea id="编程标识符" Name="名称" Cols="高度" Rows="宽度" runat="server">
    文本框内容
</textarea>
```

HtmlTextArea 控件主要成员如下: