第3章

Web 窗体的基本控件

与 ASP 不同的是，ASP.NET 提供了大量的控件，这些控件能够轻松地实现交互的复杂的 Web 应用功能。在传统的 ASP 开发中，让开发人员最为烦恼的是代码的重用性太低，以及事件代码和页面代码不能很好地分开。而在 ASP.NET 中，控件不仅解决了代码重用性的问题，对于初学者而言，控件更简单易用并能够轻松上手、投入开发。

3.1 场景导入

在网站开发中，如果需要加强用户与应用程序之间的交互，就需要上传文件。上传一个图片文件，上传后在网页上可以看到，如图 3-1 所示。

图 3-1 “上传图片示例”运行效果图

3.2 控件的属性

每个控件都有一些公共属性，例如字体颜色、边框的颜色、样式等。在 Visual Studio 2008 中，当开发人员用鼠标单击相应控件的属性后，属性栏中会简单介绍该属性的作用，如图 3-2 所示。

属性栏用来设置控件的属性，当控件在页面被初始化时，这些属性将被应用到控件中。控件的属性也可以通过编程的方法在页面相应代码区域编写，示例代码如下。

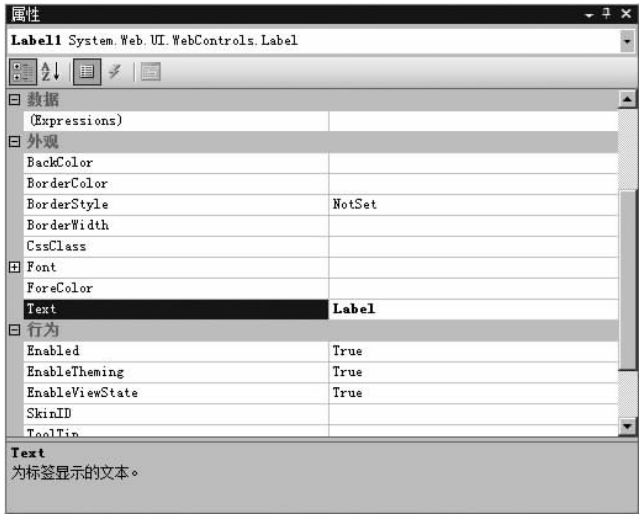

图 3-2　控件的属性

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Visible=false;                          //在 Page_Load 中设置 Label1 的可见性
}
```

上述代码编写了一个页面载入事件(Page_Load),并通过编程的方法对控件的属性进行更改,当页面加载时,控件的属性会被应用并呈现在浏览器中。

3.3　简单控件

ASP.NET 提供了诸多控件,包括简单控件、数据库控件、登录控件等。在 ASP.NET 中,简单控件是最基础也是经常被使用的控件,简单控件包括标签控件(Label)、超链接控件(HyperLink)以及图像控件(Image)等。

3.3.1　标签控件(Label)

在 Web 应用中,希望显式的文本不能被用户更改,或者当触发事件时,某一段文本能够在运行时更改,则可以使用标签控件(Label)。开发人员可以非常方便地将标签控件拖放到页面,完成拖放后,该页面将自动生成一段标签控件的声明代码,示例代码如下。

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

上述代码中,声明了一个标签控件,并将这个标签控件的 ID 属性设置为默认值 Label1。由于该控件是服务器端控件,所以在控件属性中包含 runat="server"属性。该代码还将标签控件的文本初始化为 Label,开发人员能够配置该属性进行不同文本内容的呈现。

注意:通常情况下,控件的 ID 也应该遵循良好的命名规范,以便维护。

同样，标签控件的属性也能够在相应的.cs 代码中初始化，示例代码如下。

```
protected void Page_PreInit(object sender, EventArgs e)
{
    Label1.Text="Hello World";                          //标签赋值
}
```

上述代码在页面初始化时将 Label1 的文本属性设置为"Hello World"。值得注意的是，对于 Label 标签，同样也可以显示 HTML 样式，示例代码如下。

```
protected void Page_PreInit(object sender, EventArgs e)
{
    Label1.Text="Hello World<hr/><span style=\"color:red\">A Html Code</span>";
                                                        //输出 HTML
    Label1.Font.Size=FontUnit.XXLarge;                  //设置字体大小
}
```

上述代码中，Label1 的文本属性被设置为一串 HTML 代码，当 Label 文本被呈现时，会以 HTML 效果显示，运行结果如图 3-3 所示。

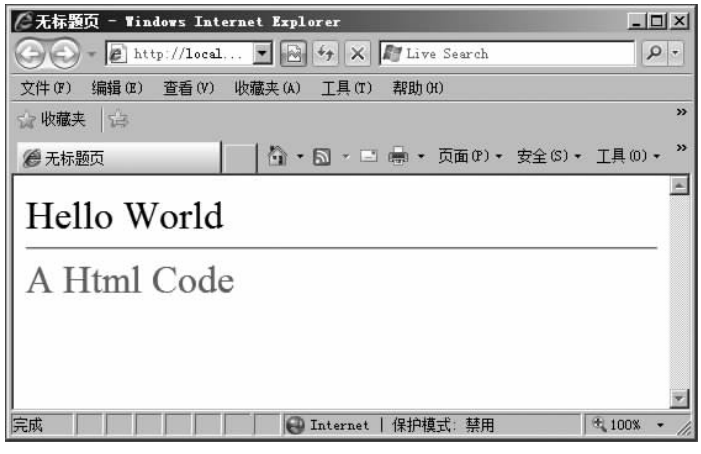

图 3-3　Label 的 Text 属性的使用

如果开发人员只是为了显示一般的文本或者 HTML 效果，则不推荐使用 Label 控件，因为服务器控件过多会导致性能问题，而使用静态的 HTML 文本能够让页面解析速度更快。

3.3.2　超链接控件(HyperLink)

超链接控件相当于实现了 HTML 代码中的效果，当然，超链接控件有自己的特点，当拖动一个超链接控件到页面时，系统会自动生成控件声明代码，示例代码如下。

```
<asp:HyperLink ID="HyperLink1" runat="server">HyperLink</asp:HyperLink>
```

上述代码声明了一个超链接控件，相对于 HTML 代码形式。超链接控件可以通过传递指定的参数来访问不同的页面。当触发一个事件后，超链接的属性可以被改变。超链接控件通常使用的两个属性如下。

(1) ImageUrl：要显示图像的 URL。

(2) NavigateUrl：要跳转到的 URL。

1. ImageUrl 属性

ImageUrl 属性可以设置这个超链接是以文本形式显示还是以图片形式显示，示例代码如下。

```
<asp:HyperLink ID="HyperLink1" runat="server"
    ImageUrl="http://www.shangducms.com/images/cms.jpg">
    HyperLink
</asp:HyperLink>
```

上述代码将文本形式显示的超链接变为了图片形式的超链接，虽然表现形式不同，但是不管是图片形式还是文本形式，都可以实现相同的效果。

2. NavigateUrl 属性

无论是文本形式还是图片形式的超链接，都可以通过 Navigate 属性来设置超链接属性，即将跳转的页面，示例代码如下。

```
<asp:HyperLink ID="HyperLink1" runat="server"
    ImageUrl="http://www.shangducms.com/images/cms.jpg"
    NavigateUrl="http://www.shangducms.com">
    HyperLink
</asp:HyperLink>
```

上述代码使用了图片超链接的形式，其中图片来自 http://www.shangducms.com/images/cms.jpg，当单击此超链接控件后，浏览器将跳到 URL 为 http://www.shangducms.com 的页面。

3. 动态跳转

超链接控件的优点在于能够对控件进行编程，来按照用户的意愿跳转到需要跳转的页面。以下代码实现了当用户选择 QQ 时跳转到腾讯网站，而选择 sohu 时跳转到 sohu.com 页面的功能，示例代码如下。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    if(DropDownList1.Text=="qq")                    //如果选择 qq
    {
        HyperLink1.Text="qq";                       //文本为 qq
        HyperLink1.NavigateUrl="http://www.qq.com"; //URL 为 qq.com
    }
    else                                            //选择 sohu
```

```
    {
        HyperLink1.Text="sohu";                                    //文本为 sohu
        HyperLink1.NavigateUrl="http://www.sohu.com";              //URL 为 sohu.com
    }
}
```

上述代码使用了 DropDownList 控件，当用户选择不同的值时，对 HyperLink1 控件进行操作，即当用户选择 qq 时，则为 HyperLink1 控件配置链接为 http://www.qq.com。

注意：与标签控件相同的是，如果只是为了单纯地实现超链接，同样不推荐使用 HyperLink 控件，因为过多地使用服务器控件同样有可能造成性能问题。

3.3.3　图像控件(Image)

图像控件用来在 Web 窗体中显示图像，其常用的属性如下。

(1) AlternateText：在图像无法显示时显示的备用文本。

(2) ImageAlign：图像的对齐方式。

(3) ImageUrl：要显示图像的 URL。

当图片无法显示时，图片将被替换成 AlternateText 属性中的文字，ImageAlign 属性用来控制图片的对齐方式，而 ImageUrl 属性用来设置图像链接地址。同样，HTML 中也可以使用<img src=""alt="">来替代图像控件，图像控件的可控性优点就是基于可通过编程来控制图像控件，图像控件基本声明代码如下。

```
<asp:Image ID="Image1" runat="server" />
```

除了显示图形以外，Image 控件的其他属性还允许为图像指定各种文本，具体如下。

(1) ToolTip：浏览器显示在工具提示中的文本。

(2) GenerateEmptyAlternateText：如果将此属性设置为 true，则呈现的图片的 alt 属性将设置为空。

开发人员能够为 Image 控件配置相应的属性以便在浏览时呈现不同的样式，创建一个 Image 控件也可以直接通过编写 HTML 代码进行呈现，示例代码如下。

```
<asp:Image ID="Image1" runat="server"
AlternateText="图片链接失效" ImageUrl="http://www.shangducms.com/images/cms
.jpg" />
```

上述代码设置了一个图片，并当图片失效时提示图片链接失效。

注意：当双击图像控件时，系统并没有生成事件所需要的代码段，这说明 Image 控件不支持任何事件。

3.4　文本框控件(TextBox)

在 Web 开发中，通常需要和用户进行交互，如用户注册、登录、发帖等，那么就需要文本框控件(TextBox)来接受用户输入的信息。开发人员还可以使用文本框控件制作高级的文

本编辑器用于 HTML 以及文本的输入、输出。

3.4.1 文本框控件的属性

通常情况下,默认的文本控件(TextBox)是一个单行的文本框,用户只能在文本框中输入一行内容。通过修改该属性,可以将文本框设置为多行或者是以密码形式显示,文本框控件常用的属性如下所示。

1. AutoPostBack(自动回传)属性

在网页的交互中,如果用户提交了表单,或者执行了相应的方法,那么该页面将会发送到服务器上,服务器执行完表单的操作或者相应方法后,再呈现给用户,如按钮控件、下拉菜单控件等。如果将某个控件的 AutoPostBack 属性设置为 true,则当该控件的属性被修改时,同样会使页面自动发回到服务器。

2. EnableViewState(控件状态)属性

ViewState 是 ASP.NET 中用来保存 Web 控件回传状态的一种机制,它是由 ASP.NET 页面框架管理的一个隐藏字段。在回传发生时,ViewState 数据同样将回传到服务器,ASP.NET 框架解析 ViewState 字符串并为页面中的各个控件填充该属性。而填充后,控件通过使用 ViewState 将数据重新恢复到以前的状态。

在使用某些特殊控件时,如用数据库控件来显示数据库。如果每次打开页面,就执行一次数据库往返过程,将是非常不明智的。开发人员可以绑定数据,在加载页面时仅对页面设置一次,然后在后续的回传中,控件将自动从 ViewState 中重新填充,减少了数据库的往返次数,从而不使用过多的服务器资源。在默认情况下,EnableViewState 的属性值通常为 true。

3. 其他属性

上面的两个属性是比较重要的属性,其他的属性也经常使用。

(1) MaxLength:在注册时可以限制用户输入的字符串长度。

(2) ReadOnly:如果将此属性设置为 true,那么文本框内的值是无法被修改的。

(3) TextMode:此属性可以设置文本框的模式,例如单行、多行或密码形式。如果不设置 TextMode 属性,那么文本框默认为单行。

(4) Columns:用于设置文本框的宽度。

(5) Rows:设置多行文本框可显示的行数。

(6) Wrap:可设置文本框是否换行。

3.4.2 文本框控件的使用

在默认情况下,文本框为单行类型,同时文本框模式也包括多行和密码,示例代码如下。

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<br />
<br />
<asp:TextBox ID="TextBox2" runat="server" Height="101px" TextMode="MultiLine"
    Width="325px"></asp:TextBox>
<br />
<br />
<asp:TextBox ID="TextBox3" runat="server" TextMode="Password"></asp:TextBox>
```

上述代码演示了 3 种文本框的使用方法，运行后的结果如图 3-4 所示。

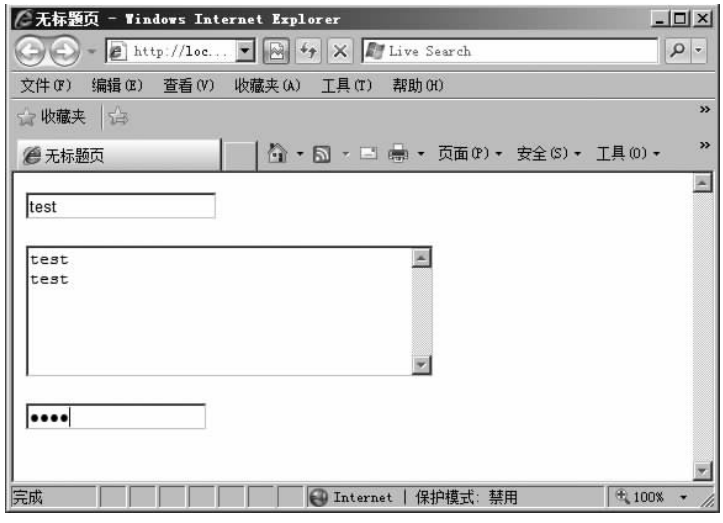

图 3-4　文本框的 3 种形式

文本框无论是在 Web 应用程序开发还是 Windows 应用程序开发中都是非常重要的。文本框在用户交互中能够起到非常重要的作用。在文本框的使用中，通常需要获取用户在文本框中输入的值或者检查文本框属性是否被改写。当获取用户的值时，必须通过一段代码来控制。文本框控件 HTML 页面示例代码如下。

```
<form id="form1" runat="server">
<div>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    <br />
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <br />
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text=
   "Button"/>
    <br />
</div>
</form>
```

上述代码声明了一个文本框控件和一个按钮控件，当用户单击按钮控件时，就需要实现标签控件的文本改变。为了实现相应的效果，可以通过编写.cs 文件代码进行逻辑处理，示例代码如下。

```
namespace _5_3                                          //页面命名空间
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
                                                        //页面加载时触发
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
                                                        //双击按钮时触发的事件
        {
            Label1.Text=TextBox1.Text;                  //标签控件的值等于文本框中控件的值
        }
    }
}
```

上述代码中，双击按钮就会触发一个按钮事件，这个事件就是将文本框内的值赋到标签内，运行结果如图 3-5 所示。

图 3-5　文本框控件的使用

同样，双击文本框控件，会触发 TextChange 事件。而运行时，当文本框控件中的字符变化后，并没有自动回传，是因为默认情况下，文本框的 AutoPostBack 属性被设置为 false。当 AutoPostBack 属性被设置为 true 时，文本框的属性变化，则会发生回传，示例代码如下。

```
protected void TextBox1_TextChanged(object sender, EventArgs e)    //文本框事件
{
    Label1.Text=TextBox1.Text;                                      //控件相互赋值
}
```

上述代码为 TextBox1 添加了 TextChanged 事件。在 TextChanged 事件中，并不是每一次文本框的内容发生变化都会重传到服务器，这一点和 WinForm 是不同的，因为这样会大大地降低页面的效率。只有当用户将文本框中的焦点移出导致 TextBox 失去焦点时，才会发生重传。

3.5　按钮控件(Button、LinkButton 和 ImageButton)

在 Web 应用程序和用户交互时，常常需要提交表单、获取表单信息等操作。在这期间，按钮控件是非常必要的。按钮控件能够触发事件，或者将网页中的信息回传给服务器。在

ASP. NET 中，包含三类按钮控件，分别为 Button、LinkButton 和 ImageButton。

3.5.1　按钮控件的通用属性

按钮控件用于事件的提交，按钮控件常用的一些通用属性如下。

(1) Causes Validation：按钮是否导致激发验证检查。

(2) CommandArgument：与此按钮关联的命令参数。

(3) CommandName：与此按钮关联的命令。

(4) ValidationGroup：指定单击按钮时调用页面上的哪些验证程序。如果未建立任何验证组，则会调用页面上的所有验证程序。

下面的语句声明了 3 种按钮，示例代码如下。

```
<asp:Button ID="Button1" runat="server" Text="Button" />  //普通的按钮
<br />
<asp:LinkButton ID="LinkButton1" runat="server">LinkButton</asp:LinkButton>
                                                          //Link 类型的按钮
<br />
<asp:ImageButton ID="ImageButton1" runat="server" />      //图像类型的按钮
```

对于 3 种按钮，它们起到的作用基本相同，主要是表现形式不同，如图 3-6 所示。

3.5.2　Click 单击事件

这 3 种按钮控件对应的事件通常是 Click 单击和 Command 命令事件。在 Click 单击事件中，通常用于编写用户单击按钮时所需要执行的事件，示例代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text="普通按钮被触发";                          //输出信息
}
protected void LinkButton1_Click(object sender, EventArgs e)
{
    Label1.Text="链接按钮被触发";                          //输出信息
}
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    Label1.Text="图片按钮被触发";                          //输出信息
}
```

上述代码分别为 3 种按钮生成了事件，其代码都是将 Label1 的文本设置为相应的文本，运行结果如图 3-7 所示。

3.5.3　Command 命令事件

按钮控件中，Click 事件并不能传递参数，所以处理的事件相对简单。而 Command 事件

图 3-6　3 种按钮类型

图 3-7　按钮的 Click 事件

可以传递参数，负责传递参数的是按钮控件的 CommandArgument 和 CommandName 属性，如图 3-8 所示。

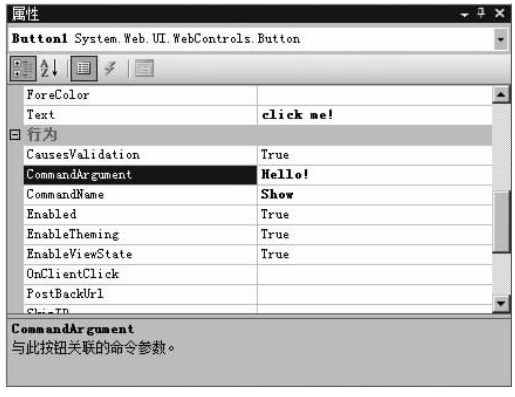

图 3-8　CommandArgument 和 CommandName 属性

将 CommandArgument 和 CommandName 属性分别设置为"Hello!"和"Show"，单击 创建一个 Command 事件并在事件中编写相应代码，示例代码如下。

```
protected void Button1_Command(object sender, CommandEventArgs e)
{
    if(e.CommandName=="Show")     //如果 CommandName 属性的值为 Show,则运行下面代码
    {
        Label1.Text=e.CommandArgument.ToString();
                                  //将 CommandArgument 属性的值赋给 Label1
    }
}
```

注意：当按钮同时包含 Click 和 Command 事件时，通常情况下会执行 Command 事件。

Command 有一些 Click 不具备的好处，就是传递参数。可以分别设置按钮的 CommandArgument 和 CommandName 属性来执行相应的方法，这样一个按钮控件就能够实现不同的方法，使得多个按钮与一个处理代码关联或者一个按钮根据不同的值进行不同