

## 第 5 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，这些控件能够轻松的实现一个交互复杂的 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的分开。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，控件还简单易用并能够轻松上手、投入开发。

### 5.1 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，当开发人员将鼠标选择了相应的控件后，属性栏中会简单的介绍该属性的作用。如图 5-1 所示。

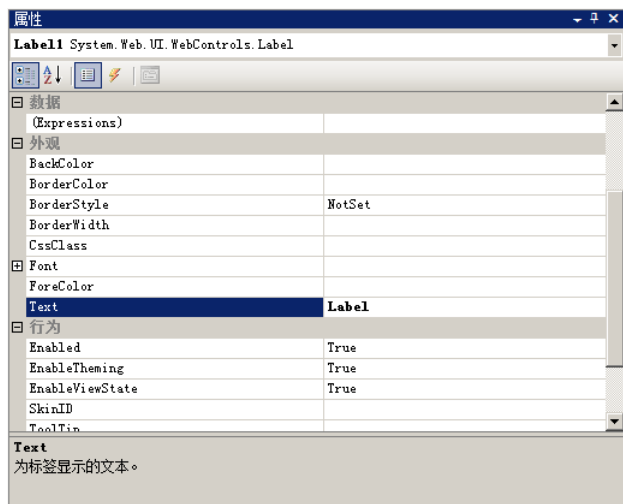


图 5-1 控件的属性

属性栏用来设置控件的属性，当控件在页面被初始化时，这些将被应用到控件。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，示例代码如下所示。

```
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 Page\_Load（页面加载事件），当页面初次被加载时，会执行 Page\_Load 中的代码。这里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。

## 5.2 简单控件

ASP.NET 提供了诸多控件，这些控件包括简单控件、数据库控件、登录控件等强大的控件。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。

### 5.2.1 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的将标签控件拖放到页面，拖放到页面后，该页面将自动生成一段标签控件的声明代码，示例代码如下所示。

```
<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，标签控件的属性能够在相应的.cs 代码中初始化，示例代码如下所示。

```
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为 Label1 的文本属性设置为“Hello World”。值得注意的是，对于 Label 标签，同样也可以显式 HTML 样式，示例代码如下所示。

```
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 效果显式，运行结果如图 5-2 所示。



图 5-2 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，不推荐使用 Label 控件，因为当服务器控件过多，会导致性能问题。使用静态的 HTML 文本能够让页面解析速度更快。

## 5.2.2 超链接控件（HyperLink）

超链接控件相当于实现了 HTML 代码中的“<a href=""></a>”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，示例代码如下所示。

```
<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。当触发了一个事件后，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所示：

- ❑ ImageUrl：要显式图像的 URL。
- ❑ NavigateUrl：要跳转的 URL。

### 1. ImageUrl 属性

设置 ImageUrl 属性可以设置这个超链接是以文本形式显式还是以图片文件显式，示例代码如下所示。

```
<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，全都实现的相同的效果。

### 2. Navigate 属性

Navigate 属性可以为无论是文本形式还是图片形式的超链接设置超链接属性，即即将跳转的页面，示例代码如下所示。

```
<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”，当点击此超链接控件后，浏览器将跳到 URL 为“http://www.shangducms.com”的页面。

### 3. 动态跳转

在前面的小结讲解了超链接控件的优点，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自己跳转的页面。以下代码实现了当用户选择 QQ 时，会跳转到腾讯网站，如果选择 SOHU，则会跳转到 SOHU 页面，示例代码如下所示。

```
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。当用户选择 qq，则为 HyperLink1 控件配置连接为 <http://www.qq.com>。

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

### 5.2.3 图像控件（Image）

图像控件用来在 Web 窗体中显示图像，图像控件常用的属性如下：

- ❑ AlternateText：在图像无法显示时显示的备用文本。
- ❑ ImageAlign：图像的对齐方式。
- ❑ ImageUrl：要显示图像的 URL。

当图片无法显示的时候，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 属性用来设置图像连接地址。同样，HTML 中也可以使用 `<img src="" alt="">` 来替代图像控件，图像控件具有可控性的优点，就是通过编程来控制图像控件，图像控件基本声明代码如下所示。

```
<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，各属性如下所示。

- ❑ ToolTip：浏览器显示在工具提示中的文本。
- ❑ 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，示例代码如下所示。

```

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

```

上述代码设置了一个图片，并当图片失效的时候提示图片连接失效。

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。

## 5.3 文本框控件（TextBox）

在 Web 开发中，Web 应用程序通常需要和用户进行交互，例如用户注册、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。

### 5.3.1 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，则可以将文本框设置为多行/或者是以密码形式显示，文本框控件常用的控件属性如下所示。

- ☐ **AutoPostBack**: 在文本修改以后，是否自动重传
- ☐ **Columns**: 文本框的宽度。
- ☐ **EnableViewState**: 控件是否自动保存其状态以用于往返过程。
- ☐ **MaxLength**: 用户输入的最大字符数。
- ☐ **ReadOnly**: 是否为只读。
- ☐ **Rows**: 作为多行文本框时所显式的行数。
- ☐ **TextMode**: 文本框的模式，设置单行，多行或者密码。
- ☐ **Wrap**: 文本框是否换行。

### 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，服务器将执行表单的操作或者执行相应方法后，再呈现给用户，例如按钮控件、下拉菜单控件等。如果将某个控件的 **AutoPostBack** 属性设置为 **true** 时，则如果该控件的属性被修改，那么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。

在使用某些特殊的控件时，如数据库控件，来显示数据库。每次打开页面执行一次数据库往返过程是非常不明智的。开发人员可以绑定数据，在加载页面时仅对页面设置一次，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。

- ☐ **MaxLength**: 在注册时可以限制用户输入的字符串长度。
- ☐ **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。
- ☐ **TextMode**: 此属性可以设置文本框的模式，例如单行、多行和密码形式。默认情况下，不设置 **TextMode** 属性，那么文本框默认为单行。

## 5.3.2 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，示例代码如下所示。

```
<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>
```

上述代码演示了三种文本框的使用方法，上述代码运行后的结果如图 5-3 所示。

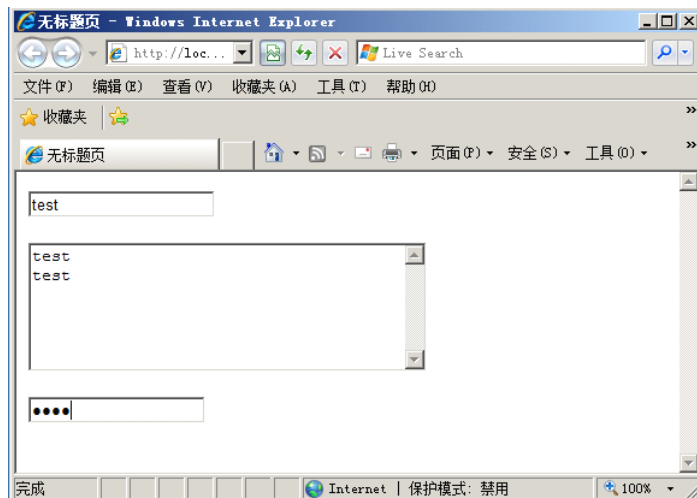


图 5-3 文本框的三种形式

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。当获取用户的值的时候，必须通过一段代码来控制。文本框控件 HTML 页面示例代码如下所示。

```
<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，示例代码如下所示：

```
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，当双击按钮时，就会触发一个按钮事件，这个事件就是将文本框内的值赋值到标签内，



运行结果如图 5-4 所示。

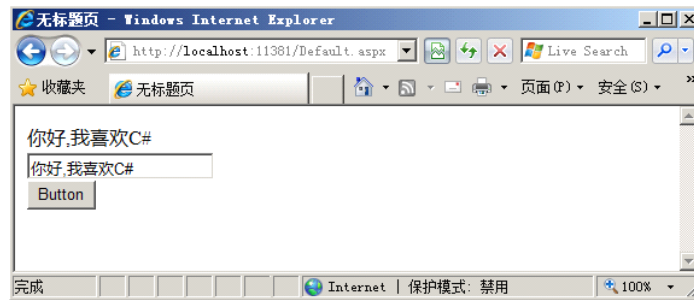


图 5-4 文本框控件的使用

同样，双击文本框控件，会触发 `TextChanged` 事件。而当运行时，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，示例代码如下所示。

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

上述代码中，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了变化之后，就会重传到服务器，这一点和 WinForm 是不同的，因为这样会大大的降低页面的效率。而当用户将文本框中的焦点移出导致 `TextBox` 就会失去焦点时，才会发生重传。

## 5.4 按钮控件（Button，LinkButton，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。在这其间，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`、`ImageButton`。

### 5.4.1 按钮控件的通用属性

按钮控件用于事件的提交，按钮控件包含一些通用属性，按钮控件的常用通用属性包括有：

- ❑ **Causes Validation**：按钮是否导致激发验证检查。
- ❑ **CommandArgument**：与此按钮管理的命令参数。
- ❑ **CommandName**：与此按钮关联的命令。
- ❑ **ValidationGroup**：使用该属性可以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。

下面的语句声明了三种按钮，示例代码如下所示。

```
<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
```

对于三种按钮，他们起到的作用基本相同，主要是表现形式不同，如图 5-5 所示。

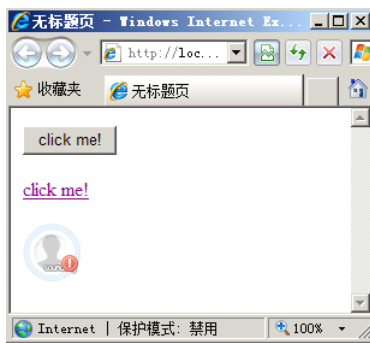


图 5-5 三种按钮类型

### 5.4.2 Click 单击事件

这三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，示例代码如下所示。

```
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 = "连接按钮被触发"; //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
}
```

上述代码分别为三种按钮生成了事件，其代码都是将 Label1 的文本设置为相应的文本，运行结果如图 5-6 所示。

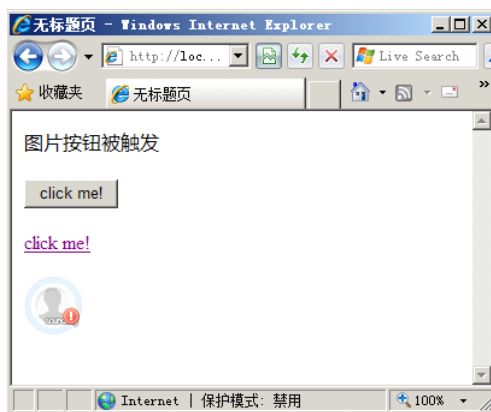


图 5-6 按钮的 Click 事件

### 5.4.3 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 5-7 所示。

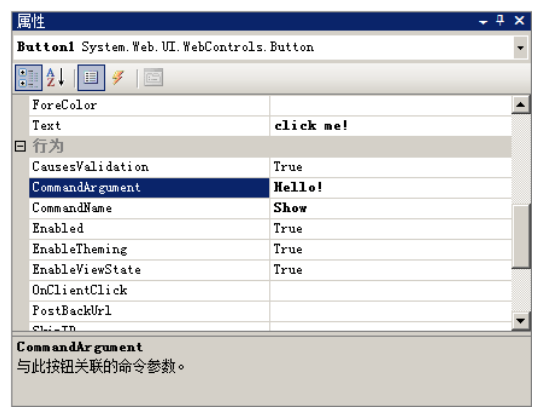



图 5-7 `CommandArgument` 和 `CommandName` 属性

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，示例代码如下所示。

```
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(); //CommandArgument 属性的值赋值给 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。可以对按钮的 `CommandArgument` 和 `CommandName` 属性分别设置，通过判断 `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的处理和响应。相比 `Click` 单击事件而言，`Command` 命令事件具有更高的可控性。

## 5.5 单选控件和单选组控件（`RadioButton` 和 `RadioButtonList`）

在投票等系统中，通常需要使用单选控件和单选组控件。顾名思义，在单选控件和单选组控件的项目中，只能在有限种选择中进行一个项目的选择。在进行投票等应用开发并且只能在选项中选择单项时，单选控件和单选组控件都是最佳的选择。

### 5.5.1 单选控件（`RadioButton`）

单选控件可以为用户选择某一个选项，单选控件常用属性如下所示。

- ☐ `Checked`：控件是否被选中。
- ☐ `GroupName`：单选控件所处的组名。
- ☐ `TextAlign`：文本标签相对于控件的对齐方式。

单选控件通常需要 `Checked` 属性来判断某个选项是否被选中，多个单选控件之间可能存在着某些联系，这些联系通过 `GroupName` 进行约束和联系，示例代码如下所示。

```
<asp:RadioButton ID="RadioButton1" runat="server" GroupName="choose"
    Text="Choose1" />
<asp:RadioButton ID="RadioButton2" runat="server" GroupName="choose"
    Text="Choose2" />
```

上述代码声明了两个单选控件，并将 GroupName 属性都设置为“choose”。单选控件中最常用的事件是 CheckedChanged，当控件的选中状态改变时，则触发该事件，示例代码如下所示。

```
protected void RadioButton1_CheckedChanged(object sender, EventArgs e)
{
    Label1.Text = "第一个被选中";
}
protected void RadioButton2_CheckedChanged(object sender, EventArgs e)
{
    Label1.Text = "第二个被选中";
}
```

上述代码中，当选中状态被改变时，则触发相应的事件。运行结果如图 5-8 所示。

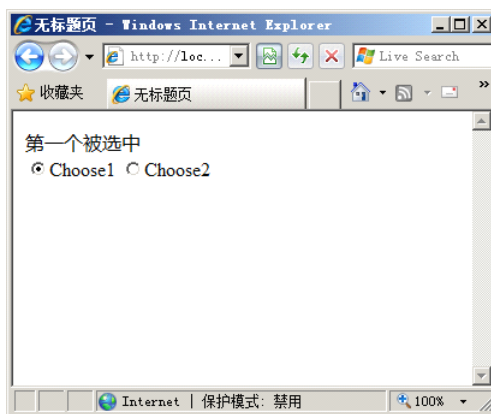


图 5-8 单选控件的使用

与 TextBox 文本框控件相同的是，单选控件不会自动进行页面回传，必须将 AutoPostBack 属性设置为 true 时才能在焦点丢失时触发相应的 CheckedChanged 事件。

### 5.5.2 单选组控件（RadioButtonList）

与单选控件相同，单选组控件也是只能选择一个项目的控件，而与单选控件不同的是，单选组控件没有 GroupName 属性，但是却能够列出多个单选项目。另外，单选组控件所生成的代码也比单选控件实现的相对较少。单选组控件添加项如图 5-9 所示。



图 5-9 单选组控件添加项

添加项目后，系统自动在.aspx 页面声明服务器控件代码，代码如下所示。

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server">
  <asp:ListItem>Choose1</asp:ListItem>
  <asp:ListItem>Choose2</asp:ListItem>
  <asp:ListItem>Choose3</asp:ListItem>
</asp:RadioButtonList>
```

上述代码使用了单选组控件进行单选功能的实现，单选组控件还包括一些属性用于样式和重复的配置。单选组控件的常用属性如下所示：

- ☐ DataMember：在数据集用做数据源时做数据绑定。
- ☐ DataSource：向列表填入项时所使用的数据源。
- ☐ DataTextField：提供项文本的数据源中的字段。
- ☐ DataTextFormat：应用于文本字段的格式。
- ☐ DataValueField：数据源中提供项值的字段。
- ☐ Items：列表中项的集合。
- ☐ RepeatColumnn：用于布局项的列数。
- ☐ RepeatDirection：项的布局方向。
- ☐ RepeatLayout：是否在某个表或者流中重复。

同单选控件一样，双击单选组控件时系统会自动生成该事件的声明，同样可以在该事件中确定代码。当选择一项内容时，提示用户所选择的内容，示例代码如下所示。

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
    Label1.Text = RadioButtonList1.Text;           //文本标签段的值等于选择的控件的值
}
```

## 5.6 复选框控件和复选组控件（CheckBox 和 CheckBoxList）

当一个投票系统需要用户能够选择多个选择项时，则单选框控件就不符合要求了。ASP.NET 还提供了复选框控件和复选组控件来满足多选的要求。复选框控件和复选组控件同单选框控件和单选组控件一样，都是通过 Checked 属性来判断是否被选择。

### 5.6.1 复选框控件（CheckBox）

同单选框控件一样，复选框也是通过 Check 属性判断是否被选择，而不同的是，复选框控件没有 GroupName 属性，示例代码如下所示。

```
<asp:CheckBox ID="CheckBox1" runat="server" Text="Check1" AutoPostBack="true" />
<asp:CheckBox ID="CheckBox2" runat="server" Text="Check2" AutoPostBack="true"/>
```

上述代码中声明了两个复选框控件。对于复选框空间，并没有支持的 GroupName 属性，当双击复选框控件时，系统会自动生成方法。当复选框控件的选中状态被改变后，会激发该事件。示例代码如下所示。

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    Label1.Text = "选框 1 被选中";           //当选框 1 被选中时
}
protected void CheckBox2_CheckedChanged(object sender, EventArgs e)
{
    Label1.Text = "选框 2 被选中,并且字体变大";           //当选框 2 被选中时
    Label1.Font.Size = FontUnit.XXLarge;
}
```

上述代码分别为两个选框设置了事件，设置了当选择选框 1 时，则文本标签输出“选框 1 被选中”，如图 5-10 所示。当选择选框 2 时，则输出“选框 2 被选中，并且字体变大”，运行结果如图 5-11 所示。

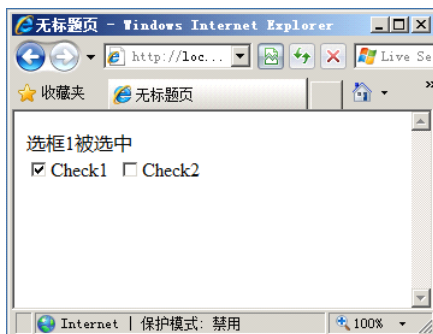


图 5-10 选框 1 被选中

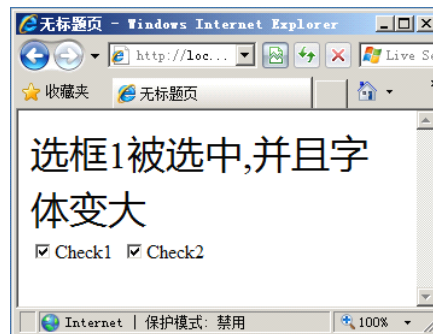


图 5-11 选框 2 被选中

对于复选框而言，用户可以在复选框控件中选择多个选项，所以就没有必要为复选框控件进行分组。在单选框控件中，相同组名的控件只能选择一项用于约束多个单选框中的选项，而复选框就没有约束的必要。

### 5.6.2 复选组控件（CheckBoxList）

同单选组控件相同，为了方便复选控件的使用，.NET 服务器控件中同样包括了复选组控件，拖动一个复选组控件到页面可以同单选组控件一样添加复选组列表。添加在页面后，系统生成代码如下所示。

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" AutoPostBack="True"
onselectedindexchanged="CheckBoxList1_SelectedIndexChanged">
    <asp:ListItem Value="Choose1">Choose1</asp:ListItem>
    <asp:ListItem Value="Choose2">Choose2</asp:ListItem>
    <asp:ListItem Value="Choose3">Choose3</asp:ListItem>
</asp:CheckBoxList>
```

上述代码中，同样增加了 3 个项目提供给用户选择，复选组控件最常用的是 `SelectedIndexChanged` 事件。当控件中某项的选中状态被改变时，则会触发该事件。示例代码如下所示。

```
protected void CheckBoxList1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (CheckBoxList1.Items[0].Selected)           //判断某项是否被选中
    {
        Label1.Font.Size = FontUnit.XXLarge;       //更改字体大小
    }
    if (CheckBoxList1.Items[1].Selected)           //判断是否被选中
    {
        Label1.Font.Size = FontUnit.XLarge;        //更改字体大小
    }
    if (CheckBoxList1.Items[2].Selected)
    {
        Label1.Font.Size = FontUnit.XSmall;
    }
}
```

上述代码中，`CheckBoxList1.Items[0].Selected` 是用来判断某项是否被选中，其中 `Item` 数组是复选组控件中项目的集合，其中 `Items[0]` 是复选组中的第一个项目。上述代码用来修改字体的大小，如图 5-12 所示，当选择不同的选项时，字体的大小也不相同，运行结果如图 5-13 所示。



图 5-12 选择大号字体

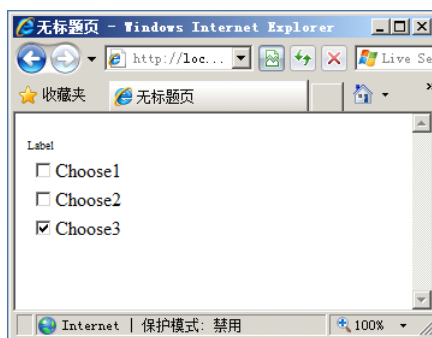


图 5-13 选择小号字体

正如图 5-12、5-13 所示，当用户选择不同的选项时，`Label` 标签的字体的大小会随之改变。

注意：复选组控件与单选组控件不同的是，不能够直接获取复选组控件某个选中项目的值，因为复选组控件返回的是第一个选择项的返回值，只能通过 `Item` 集合来获取选择某个或多个选中的项目值。

## 5.7 列表控件（`DropDownList`，`ListBox` 和 `BulletedList`）

在 Web 开发中，经常会需要使用列表控件，让用户的输入更加简单。例如在用户注册时，用户的所在地是有限的集合，而且用户不喜欢经常键入，这样就可以使用列表控件。同样列表控件还能够简化用户输入并且防止用户输入在实际中不存在的数据，如性别的选择等。

### 5.7.1 `DropDownList` 列表控件

列表控件能在一个控件中为用户提供多个选项，同时又能够避免用户输入错误的选项。例如，在用

户注册时，可以选择性别是男，或者女，就可以使用 DropDownList 列表控件，同时又避免了用户输入其他的信息。因为性别除了男就是女，输入其他的信息说明这个信息是错误的或者是无效的。下列语句声明了一个 DropDownList 列表控件，示例代码如下所示。

```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem>1</asp:ListItem>
    <asp:ListItem>2</asp:ListItem>
    <asp:ListItem>3</asp:ListItem>
    <asp:ListItem>4</asp:ListItem>
    <asp:ListItem>5</asp:ListItem>
    <asp:ListItem>6</asp:ListItem>
    <asp:ListItem>7</asp:ListItem>
</asp:DropDownList>
```

上述代码创建了一个 DropDownList 列表控件，并手动增加了列表项。同时 DropDownList 列表控件也可以绑定数据源控件。DropDownList 列表控件最常用的事件是 SelectedIndexChanged，当 DropDownList 列表控件选择项发生变化时，则会触发该事件，示例代码如下所示。

```
protected void DropDownList1_SelectedIndexChanged1(object sender, EventArgs e)
{
    Label1.Text = "你选择了第" + DropDownList1.Text + "项";
}
```

上述代码中，当选择的项目发生变化时则会触发该事件，如图 5-14 所示。当用户再次进行选择时，系统会将更改标签 1 中的文本，如图 5-15 所示。

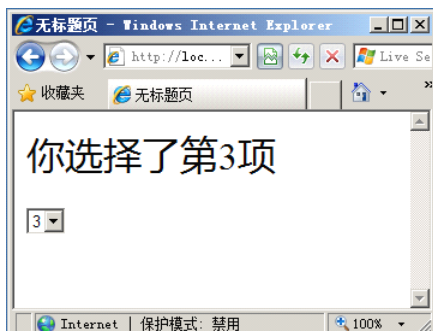
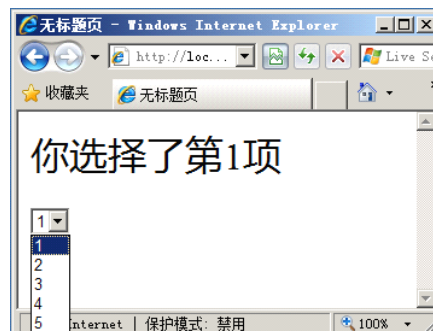


图 5-14 选择第三项



5-15 选择第一项

当用户选择相应的项目时，就会触发 SelectedIndexChanged 事件，开发人员可以通过捕捉相应的用户选中的控件进行编程处理，这里就捕捉了用户选择的数字进行字体大小的更改。

## 5.7.2 ListBox 列表控件

相对于 DropDownList 控件而言，ListBox 控件可以指定用户是否允许多项选择。设置 SelectionMode 属性为 Single 时，表明只允许用户从列表框中选择一个项目，而当 SelectionMode 属性的值为 Multiple 时，用户可以按住 Ctrl 键或者使用 Shift 组合键从列表中选择多个数据项。当创建一个 ListBox 列表控件后，开发人员能够在控件中添加所需的项目，添加完成后示例代码如下所示。

```
<asp:ListBox ID="ListBox1" runat="server" Width="137px" AutoPostBack="True">
    <asp:ListItem>1</asp:ListItem>
    <asp:ListItem>2</asp:ListItem>
    <asp:ListItem>3</asp:ListItem>
</asp:ListBox>
```

```

<asp:ListItem>4</asp:ListItem>
<asp:ListItem>5</asp:ListItem>
<asp:ListItem>6</asp:ListItem>
</asp:ListBox>

```

从结构上看，ListBox 列表控件的 HTML 样式代码和 DropDownList 控件十分相似。同样，SelectedIndexChanged 也是 ListBox 列表控件中最常用的事件，双击 ListBox 列表控件，系统会自动生成相应的代码。同样，开发人员可以为 ListBox 控件中的选项改变后的事件做编程处理，示例代码如下所示。

```

protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    Label1.Text = "你选择了第" + ListBox1.Text + "项";
}

```

上述代码中，当 ListBox 控件选择项发生改变后，该事件就会被触发并修改相应 Label 标签中文本，如图 5-16 所示。

上面的程序同样实现了 DropDownList 中程序的效果。不同的是，如果需要进行实现让用户选择多个 ListBox 项，只需要设置 SelectionMode 属性为“Multiple”即可，如图 5-17 所示。

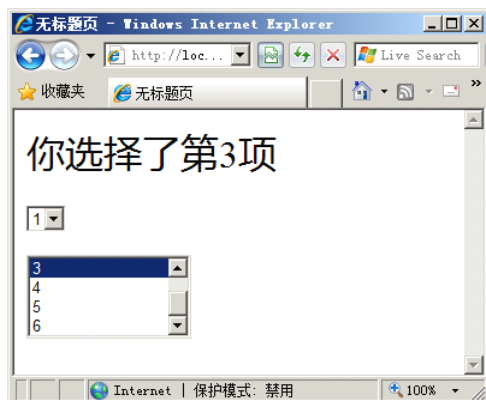


图 5-16 ListBox 单选

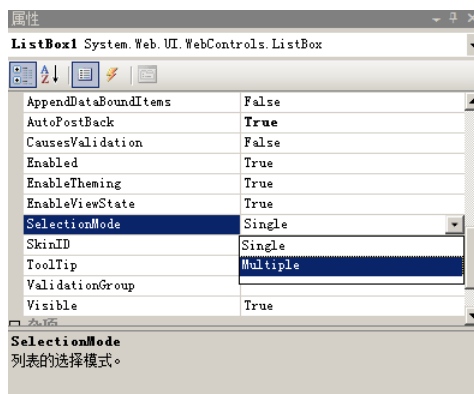


图 5-17 SelectionMode 属性

当设置了 SelectionMode 属性后，用户可以按住 Ctrl 键或者使用 Shift 组合键选择多项。同样，开发人员也可以编写处理选择多项时的事件，示例代码如下所示。

```

protected void ListBox1_SelectedIndexChanged1(object sender, EventArgs e)
{
    Label1.Text += ",你选择了第" + ListBox1.Text + "项";
}

```

上述代码使用了“+=”运算符，在触发 SelectedIndexChanged 事件后，应用程序将为 Label1 标签赋值，如图 5-18 所示。当用户每选一项的时候，就会触发该事件，如图 5-19 所示。



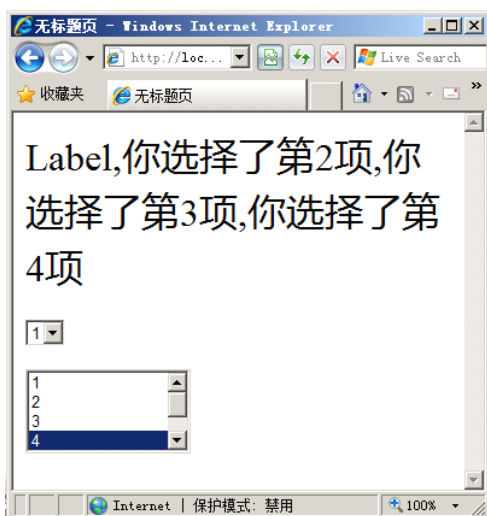


图 5-18 单选效果

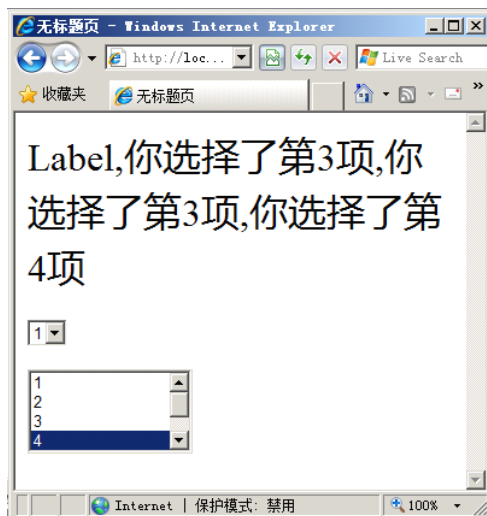


图 5-19 多选效果

从运行结果可以看出，当单选时，选择项返回值和选择的项相同，而当选择多项的时候，返回值同第一项相同。所以，在选择多项时，也需要使用 Item 集合获取和遍历多个项目。

### 5.7.3 BulletedList 列表控件

BulletedList 与上述列表控件不同的是，BulletedList 控件可呈现项目符号或编号。对 BulletedList 属性的设置为呈现项目符号，则当 BulletedList 被呈现在页面时，列表前端会则会显式项目符号或者特殊符号，效果如图 5-20 所示。

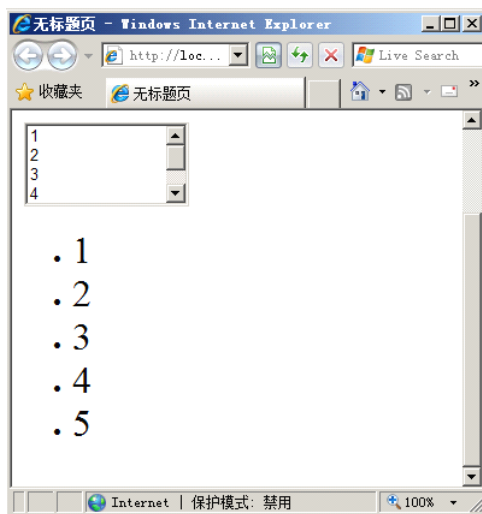


图 5-20 BulletedList 显式效果

BulletedList 可以通过设置 BulletStyle 属性来编辑列表前的符号样式，常用的 BulletStyle 项目符号编号样式如下所示。

- ☐ Circle: 项目符号设置为○。
- ☐ CustomImage: 项目符号为自定义图片。
- ☐ Disc: 项目符号设置为●。

- ❑ LowerAlpha: 项目符号为小写字母格式, 如 a、b、c 等。
- ❑ LowerRoman: 项目符号为罗马数字格式, 如 i、ii 等。
- ❑ NotSet: 表示不设置, 此时将以 Disc 样式为默认样式。
- ❑ Numbered: 项目符号为 1、2、3、4 等。
- ❑ Square: 项目符号为黑方块■。
- ❑ UpperAlpha: 项目符号为大写字母格式, 如 A、B、C 等。
- ❑ UpperRoman: 项目符号为大写罗马数字格式如 I、II、III 等。

同样, BulletedList 控件也同 DropDownList 以及 ListBox 相同, 可以添加事件。不同的是生成的事件是 Click 事件, 代码如下所示。

```
protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
{
    Label1.Text += ",你选择了第" + BulletedList1.Items[e.Index].ToString() + "项";
}
```

DropDownList 和 ListBox 生成的事件是 SelectedIndexChanged, 当其中的选择项被改变时, 则触发该事件。而 BulletedList 控件生成的事件是 Click, 用于在其中提供逻辑以执行特定的应用程序任务。

## 5.8 面板控件 (Panel)

面板控件就好像是一些控件的容器, 可以将一些控件包含在面板控件内, 然后对面板控制进行操作来设置在面板控件内的所有控件是显示还是隐藏, 从而达到设计者的特殊目的。当创建一个面板控件时, 系统会生成相应的 HTML 代码, 示例代码如下所示。

```
<asp:Panel ID="Panel1" runat="server">
</asp:Panel>
```

面板控件的常用功能就是显示或隐藏一组控件, 示例 HTML 代码如下所示。

```
<form id="form1" runat="server">
    <asp:Button ID="Button1" runat="server" Text="Show" />
    <asp:Panel ID="Panel1" runat="server" Visible="False">
        <asp:Label ID="Label1" runat="server" Text="Name:" style="font-size: xx-large"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        This is a Panel!
    </asp:Panel>
</form>
```

上述代码创建了一个 Panel 控件, Panel 控件默认属性为隐藏, 并在控件外创建了一个 Button 控件 Button1, 当用户单击外部的按钮控件后将显示 Panel 控件, cs 代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Panel1.Visible = true; //Panel 控件显示可见
}
```

当页面初次被载入时, Panel 控件以及 Panel 控件内部的服务器控件都为隐藏, 如图 5-21 所示。当用户单击 Button1 时, 则 Panel 控件可见性为可见, 则页面中的 Panel 控件以及 Panel 控件中的所有服务器控件也都为可见, 如图 5-22 所示。

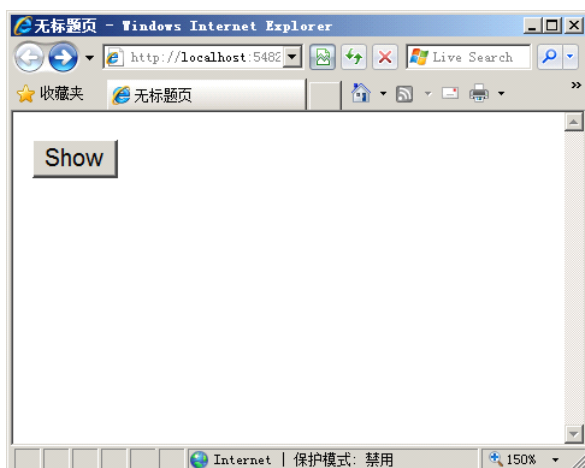


图 5-21 Panel 控件隐藏

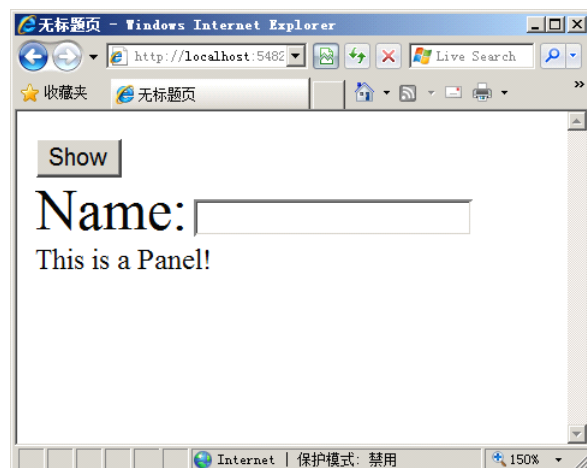


图 5-22 Panel 被显示

将 TextBox 控件和 Button 控件放到 Panel 控件中，可以为 Panel 控件的 DefaultButton 属性设置为面板中某个按钮的 ID 来定义一个默认的按钮。当用户在面板中输入完毕，可以直接按 Enter 键来传送表单。并且，当设置了 Panel 控件的高度和宽度时，当 Panel 控件中的内容高度或宽度超过时，还能够自动出现滚动条。

Panel 控件还包含一个 GroupText 属性，当 Panel 控件的 GroupText 属性被设置时，Panel 将会被创建一个带标题的分组框，效果如图 5-23 所示。

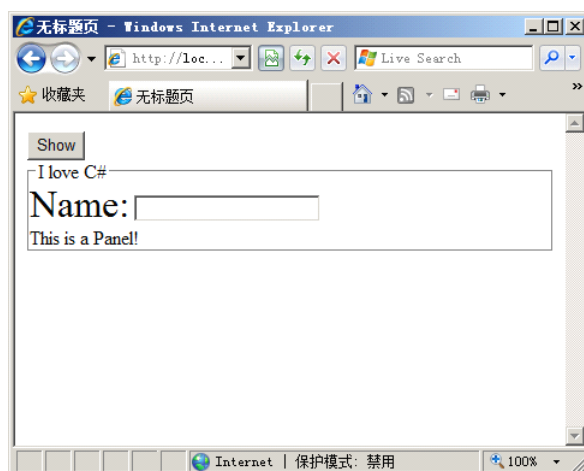


图 5-23 Panel 控件的 GroupText 属性

GroupText 属性能够进行 Panel 控件的样式呈现，通过编写 GroupText 属性能够更加清晰的让用户了解 Panel 控件中服务器控件的类别。例如当有一组服务器用于填写用户的信息时，可以将 Panel 控件的 GroupText 属性编写成为“用户信息”，让用户知道该区域是用于填写用户信息的。

## 5.9 占位控件（Placeholder）

在传统的 ASP 开发中，通常在开发页面的时候，每个页面有很多相同的元素，例如导航栏、GIF 图片等。使用 ASP 进行应用程序开发通常使用 include 语句在各个页面包含其他页面的代码，这样的方法

虽然解决了相同元素的很多问题，但是代码不够美观，而且时常会出现问题。ASP.NET 中可以使用 Placeholder 来解决这个问题，与面板控件 Panel 控件相同的是，占位控件 Placeholder 也是控件的容器，但是在 HTML 页面呈现中本身并不产生 HTML，创建一个 Placeholder 控件代码如下所示。

```
<asp:Placeholder ID="Placeholder1" runat="server"></asp:Placeholder>
```

在 CS 页面中，允许用户动态的在 Placeholder 上创建控件，CS 页面代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    TextBox text = new TextBox();           //创建一个 TextBox 对象
    text.Text = "NEW";
    this.Placeholder1.Controls.Add(text);    //为占位控件动态增加一个控件
}
```

上述代码动态的创建了一个 TextBox 控件并显示在占位控件中，运行效果如图 5-24 所示。

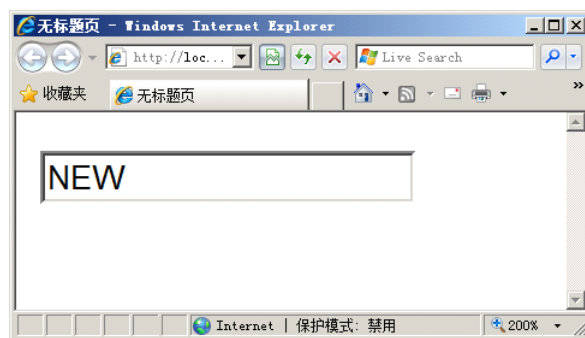


图 5-24 Placeholder 控件的使用

开发人员不仅能够通过编程在 Placeholder 控件中添加控件，开发人员同样可以在 Placeholder 控件中拖动相应的服务器控件进行控件呈现和分组。

## 5.10 日历控件（Calendar）

在传统的 Web 开发中，日历是最复杂也是最难实现的功能，好在 ASP.NET 中提供了强大的日历控件来简化日历控件的开发。日历控件能够实现日历的翻页、日历的选取以及数据的绑定，开发人员能够在博客、OA 等应用的开发中使用日历控件从而减少日历应用的开发。

### 5.10.1 日历控件的样式

日历控件通常在博客、论坛等程序中使用，日历控件不仅仅只是显示了一个日历，用户还能够通过日历控件进行时间的选取。在 ASP.NET 中，日历控件还能够和数据库进行交互操作，实现复杂的数据绑定。开发人员能够将日历控件拖动在主窗口中，在主窗口的代码视图下会自动生成日历控件的 HTML 代码，示例代码如下所示。

```
<asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
```

ASP.NET 通过上述简单的代码就创建了一个强大的日历控件，其效果如图 5-25 所示。

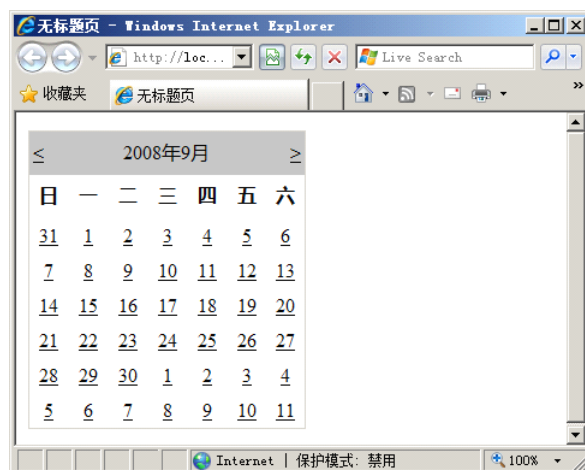


图 5-25 日历控件

日历控件通常用于显示月历，日历控件允许用户选择日期和移动到下一页或上一页。通过设置日历控件的属性，可以更改日历控件的外观。常用的日历控件的属性如下所示：

- ☐ DayHeaderStyle: 月历中显示一周中每一天的名称和部分的样式。
- ☐ DayStyle: 所显示的月份中各天的样式。
- ☐ NextPrevStyle: 标题栏左右两端的月导航所在部分的样式。
- ☐ OtherMonthDayStyle: 上个月和下个月的样式。
- ☐ SelectedDayStyle: 选定日期的样式。
- ☐ SelectorStyle: 位于月历控件左侧，包含用于选择一周或整个月的连接的列样式。
- ☐ ShowDayHeader: 显示或隐藏一周中的每一天的部分。
- ☐ ShowGridLines: 显示或隐藏一个月中的每一天之间的网格线。
- ☐ ShowNextPrevMonth: 显示或隐藏到下一个或上一个月的导航控件。
- ☐ ShowTitle: 显示或隐藏标题部分。
- ☐ TitleStyle: 位于月历顶部，包含月份名称和月导航连接的标题栏样式。
- ☐ TodayDayStyle: 当前日期的样式。
- ☐ WeekendDayStyle: 周末日期的样式。

Visual Studio 还为开发人员提供了默认的日历样式从而能够选择自动套用格式进行样式控制，如图 5-26 所示。



图 5-26 使用系统样式

除了上述样式可以设置以外，ASP.NET 还为用户设计了若干样式，若开发人员觉得设置样式非常困难，则可以使用系统默认的样式进行日历控件的样式呈现。

## 5.10.2 日历控件的事件

同所有的控件相同，日历控件也包含自身的事件，常用的日历控件的事件包括有：

- ☐ DayRender：当日期被显示时触发该事件。
- ☐ SelectionChanged：当用户选择日期时触发该事件。
- ☐ VisibleMonthChanged：当所显示的月份被更改时触发该事件。

在创建日历控件中每个日期单元格时，则会触发 DayRender 事件。当用户选择月历中的日期时，则会触发 SelectionChanged 事件，同样，当双击日历控件时，会自动生成该事件的代码块。当对当前月份进行切换，则会激发 VisibleMonthChanged 事件。开发人员可以通过一个标签来接受当前事件，当选择月历中的某一天，则此标签显示当前日期，示例代码如下所示。

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    Label1.Text =
        "现在的时间是:" + Calendar1.SelectedDate.Year.ToString() + "年"
        + Calendar1.SelectedDate.Month.ToString()+"月"
        + Calendar1.SelectedDate.Day.ToString()+"号"
        + Calendar1.SelectedDate.Hour.ToString()+"点";
}
```

在上述代码中，当用户选择了月历中的某一天时，则标签中的文本会变为当前的日期文本，如“现在的时间是 xx”之类。在进行逻辑编程的同时，也需要对日历控件的样式做稍许更改，日历控件的 HTML 代码如下所示。

```
<asp:Calendar ID="Calendar1" runat="server" BackColor="#FFFFCC"
    BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest"
    Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399" Height="200px"
    onselectionchanged="Calendar1_SelectionChanged" ShowGridLines="True"
    Width="220px">
    <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
```

```

<SelectorStyle BackColor="#FFCC66" />
<TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
<OtherMonthDayStyle ForeColor="#CC9966" />
<NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
<DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
<TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
ForeColor="#FFFFCC" />
</asp:Calendar>

```

上述代码中的日历控件选择的是 ASP.NET 的默认样式，如图 5-27 所示。当确定了日历控件样式后，并编写了相应的 SelectionChanged 事件代码后，就可以通过日历控件获取当前时间，或者对当前时间进行编程，如图 5-28 所示。

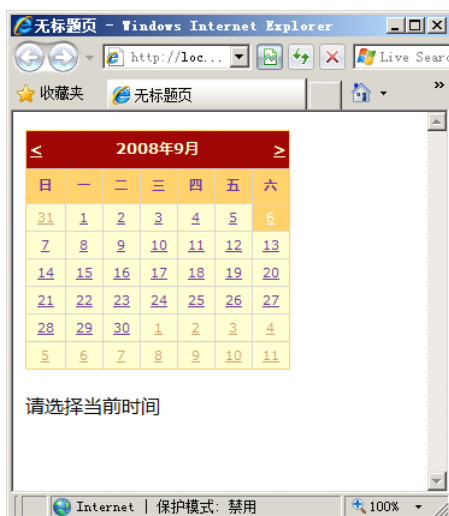


图 5-27 日历控件

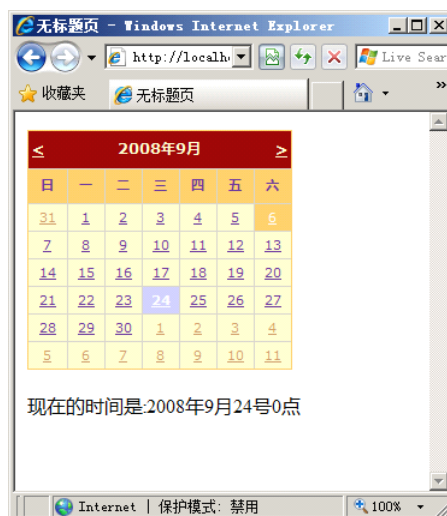


图 5-28 选择一个日期

## 5.11 广告控件（AdRotator）

在 Web 应用开发中，广告总是必不可少的。而 ASP.NET 为开发人员提供了广告控件为页面在加载时提供一个或一组广告。广告控件可以从固定的数据源中读取（如 XML 或数据源控件），并从中自动读取广告信息。当页面每刷新一次时，广告显示的内容也同样会被刷新。

广告控件必须放置在 Form 或 Panel 控件，以及模板内。广告控件需要包含图像的地址的 XML 文件。并且该文件用来指定每个广告的导航连接。广告控件最常用的属性就是 AdvertisementFile，使用它来配置相应的 XML 文件，所以必须首先按照标准格式创建一个 XML 文件，如图 5-29 所示。



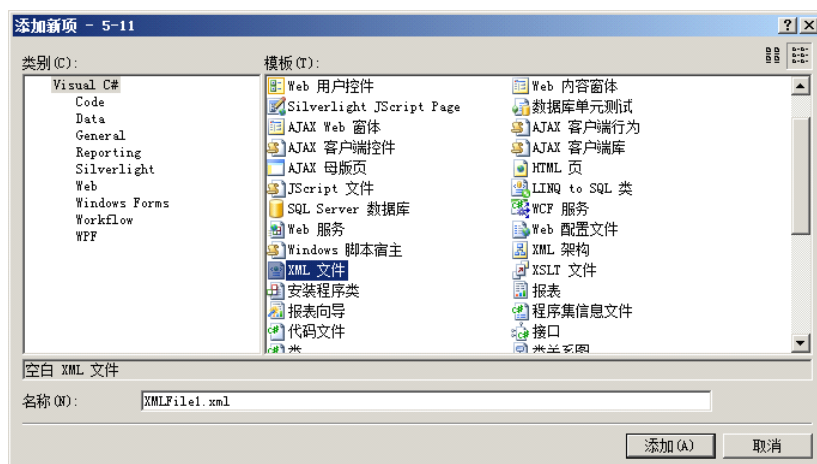


图 5-29 创建一个 XML 文件

创建了 XML 文件之后，开发人员并不能按照自己的意愿进行 XML 文档的编写，如果要正确的被广告控件解析形成广告，就需要按照广告控件要求的标准的 XML 格式来编写代码，示例代码如下所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  [<Ad>
    <ImageUrl></ImageUrl>
    <NavigateUrl></NavigateUrl>
    [<OptionalImageUrl></OptionalImageUrl>]*
    [<OptionalNavigateUrl></OptionalNavigateUrl>]*
    <AlternateText></AlternateText>
    <Keyword></Keyword>
    <Impression></Impression>
  </Ad>]*
</Advertisements>
```

上述代码实现了一个标准的广告控件的 XML 数据源格式，其中各标签意义如下所示：

- ❑ ImageUrl: 指定一个图片文件的相对路径或绝对路径，当没有 ImageKey 元素与 OptionalImageUrl 匹配时则显示该图片。
- ❑ NavigateUrl: 当用户单击广告时单没有 NaavigateUrlKey 元素与 OptionalNavigateUrl 元素匹配时，会将用户发送到该页面。
- ❑ OptionalImageUrl: 指定一个图片文件的相对路径或绝对路径，对于 ImageKey 元素与 OptionalImageUrl 匹配时则显示该图片。
- ❑ OptionalNavigateUrl: 当用户单击广告时单有 NaavigateUrlKey 元素与 OptionalNavigateUrl 元素匹配时，会将用户发送到该页面。
- ❑ AlternateText: 该元素用来替代 IMG 中的 ALT 元素。
- ❑ KeyWord: KeyWord 用来指定广告的种类。
- ❑ Impression: 该元素是一个数值，指示轮换时间表中该广告相对于文件中的其他广告的权重。

当创建了一个 XML 数据源之后，就需要对广告控件的 AdvertisementFile 进行更改，如图 5-30 所示。

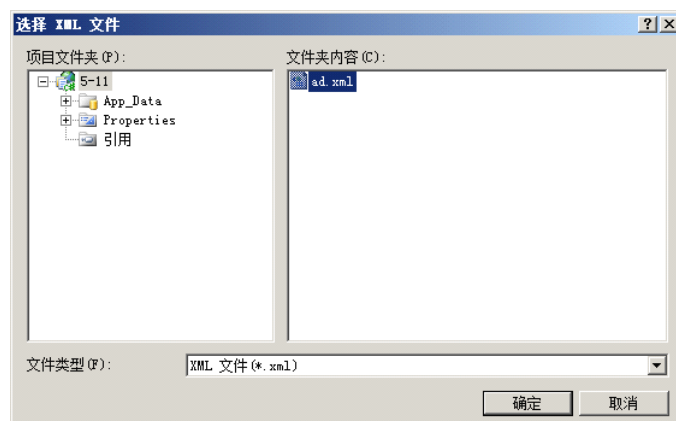


图 5-30 指定相应的数据源

配置好数据源之后，就需要在广告控件的数据源 XML 文件中加入自己的代码了，XML 广告文件示例代码如下所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>http://www.shangducms.com/images/cms.jpg</ImageUrl>
    <NavigateUrl>http://www.shangducms.com</NavigateUrl>
    <AlternateText>我的网站</AlternateText>
    <Keyword>software</Keyword>
    <Impression>100</Impression>
  </Ad>
  <Ad>
    <ImageUrl>http://www.shangducms.com/images/hello.jpg</ImageUrl>
    <NavigateUrl>http://www.shangducms.com</NavigateUrl>
    <AlternateText>我的网站</AlternateText>
    <Keyword>software</Keyword>
    <Impression>100</Impression>
  </Ad>
</Advertisements>
```

运行程序，广告对应的图像在页面每次加载的时候被呈现，如图 5-31 所示。页面每次刷新时，广告控件呈现的广告内容都会被刷新，如图 5-32 所示。

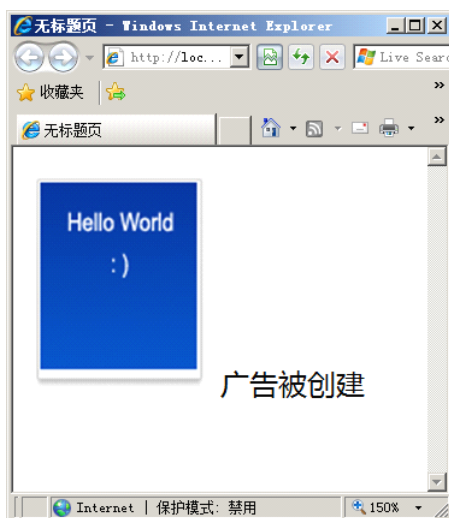


图 5-31 一个广告被呈现

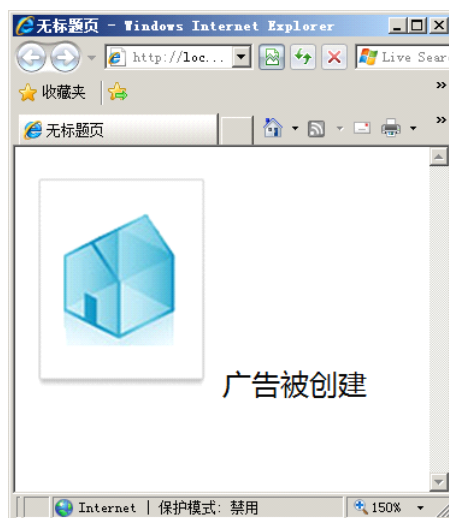


图 5-32 刷新后更换广告内容

注意：广告控件本身并不提供点击统计，所以无法计算广告是否被用户点击或者统计用户最关心的广告。

## 5.12 文件上传控件（FileUpload）

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。例如在论坛中，用户需要上传文件分享信息或在博客中上传视频分享快乐等等。上传文件在 ASP 中是一个复杂的问题，可能需要通过组件才能够实现文件的上传。在 ASP.NET 中，开发环境默认提供了文件上传控件来简化文件上传的开发。当开发人员使用文件上传控件时，将会显示一个文本框，用户可以键入或通过“浏览”按钮浏览和选择希望上传到服务器的文件。创建一个文件上传控件系统生成的 HTML 代码如下所示。

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

文件上传控件可视化设置属性较少，大部分都是通过代码控制完成的。当用户选择了一个文件并提交页面后，该文件作为请求的一部分上传，文件将被完整的缓存在服务器内存中。当文件完成上传，页面才开始运行，在代码运行的过程中，可以检查文件的特征，然后保存该文件。同时，上传控件在选择文件后，并不会立即执行操作，需要其他的控件来完成操作，例如按钮控件（Button）。实现文件上传的 HTML 核心代码如下所示。

```
<body>
  <form id="form1" runat="server">
    <div>
      <asp:FileUpload ID="FileUpload1" runat="server" />
      <asp:Button ID="Button1" runat="server" Text="选择好了，开始上传" />
    </div>
  </form>
</body>
```

上述代码通过一个 Button 控件来操作文件上传控件，当用户单击按钮控件后就能够将上传控件中选中的控件上传到服务器空间中，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    FileUpload1.PostedFile.SaveAs(Server.MapPath("upload/beta.jpg")); //上传文件另存为
```

}

上述代码将一个文件上传到了 upload 文件夹内，并保存为 jpg 格式，如图 5-33 所示。打开服务器文件，可以看到文件已经上传了，如图 5-34 所示。

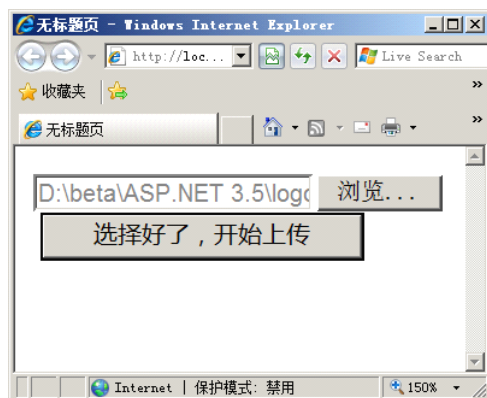


图 5-33 上传文件

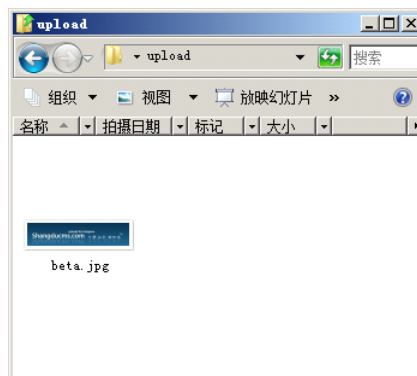


图 5-34 文件已经被上传

上述代码将文件保存在 UPLOAD 文件夹中，并保存为 JPG 格式。但是通常情况下，用户上传的并不全部都是 JPG 格式，也有可能是 DOC 等其他格式的文件，在这段代码中，并没有对其他格式进行处理而全部保存为了 JPG 格式。同时，也没有对上传的文件进行过滤，存在着极大的安全风险，开发人员可以将相应的文件上传的 cs 更改，以便限制用户上传的文件类型，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile) //如果存在文件
    {
        string fileExtension = System.IO.Path.GetExtension(FileUpload1.FileName); //获取文件扩展名
        if (fileExtension != ".jpg") //如果扩展名不等于 jpg 时
        {
            Label1.Text = "文件上传类型不正确，请上传 jpg 格式"; //提示用户重新上传
        }
        else
        {
            FileUpload1.PostedFile.SaveAs(Server.MapPath("upload/beta.jpg")); //文件保存
            Label1.Text = "文件上传成功"; //提示用户成功
        }
    }
}
```

上述代码中决定了用户只能上传 JPG 格式，如果用户上传的文件不是 JPG 格式，那么用户将被提示上传的文件类型有误并停止用户的文件上传，如果文件的类型为 JPG 格式，用户就能够上传文件到服务器的相应目录中，如图 5-35 所示。运行上传控件进行文件上传，运行结果如图 5-36 所示。

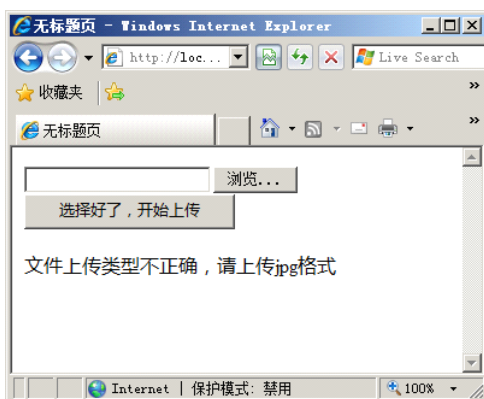


图 5-35 文件类型错误

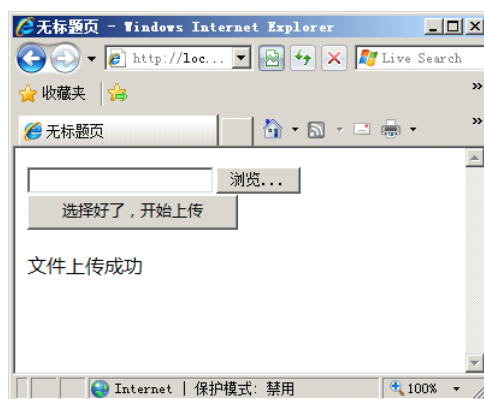


图 5-36 文件类型正确

值得注意的是, 上传的文件在.NET 中, 默认上传文件最大为 4M 左右, 不能上传超过该限制的任何内容。当然, 开发人员可以通过配置.NET 相应的配置文件来更改此限制, 但是推荐不要更改此限制, 否则可能造成潜在的安全威胁。

注意: 如果需要更改默认上传文件大小的值, 通常可以直接修改存放在 C:\WINDOWS\Microsoft.NET\Framework\V2.0.50727\CONFIG 的 ASP.NET 2.0 配置文件, 通过修改文件中的 maxRequestLength 标签的值, 或者可以通过 web.config 来覆盖配置文件。

## 5.13 视图控件 (MultiView 和 View)

视图控件很像在 WinForm 开发中的 TabControl 控件, 在网页开发中, 可以使用 MultiView 控件作为一个或多个 View 控件的容器, 让用户体验得到更大的改善。在一个 MultiView 控件中, 可以放置多个 View 控件 (选项卡), 当用户点击到关心的选项卡时, 可以显示相应的内容, 很像 Visual Studio 2008 中的设计、视图、拆分等类型的功能。

无论是 MultiView 还是 View, 都不会在 HTML 页面中呈现任何标记。而 MultiView 控件和 View 没有像其他控件那样多的属性, 惟一需要指定的就是 ActiveViewIndex 属性, 视图控件 HTML 代码如下所示。

```
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
  <asp:View ID="View1" runat="server">
    abc<br />
    <asp:Button ID="Button1" runat="server" CommandArgument="View2"
      CommandName="SwitchViewByID" Text="下一个" />
  </asp:View>
  <asp:View ID="View2" runat="server">
    123<br />
    <asp:Button ID="Button2" runat="server" CommandArgument="View1"
      CommandName="SwitchViewByID" Text="上一个" />
  </asp:View>
</asp:MultiView>
```

上述代码中, 使用了 Button 来对视图控件进行选择, 通过单击按钮, 来选择替换到【下一个】或者是【上一个】按钮, 如图 5-37 所示。在用户注册中, 这一步能够制作成 Web 向导, 让用户更加方便的使用 Web 应用。当标签显式完毕后, 会显式上一步按钮 5-38 所示。

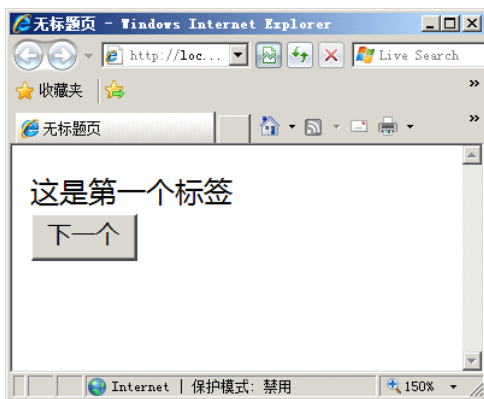


图 5-37 第一个标签

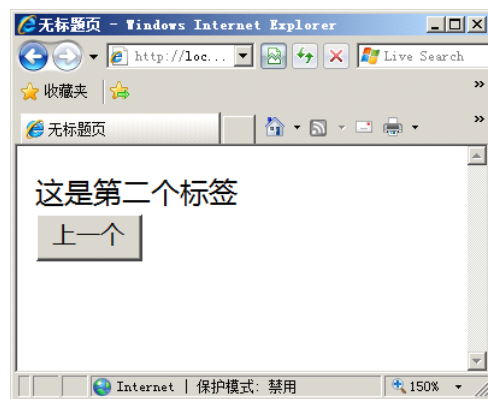


图 5-38 第二个标签

注意：在 HTML 代码中，并没有为每个按钮的事件编写代码，是因为按钮通过 `CommandArgument` 和 `CommandName` 属性操作视图控件。

`MultiView` 和 `View` 控件能够实现 `Panel` 控件的任务，但可以让用户选择其他条件。同时 `MultiView` 和 `View` 能够实现 `Wizard` 控件相似的行为，并且可以自己编写实现细节。相比之下，当不需要使用 `Wizard` 提供的方法时，可以使用 `MultiView` 和 `View` 控件来代替，并且编写过程更加“可视化”，如图 5-39 所示。

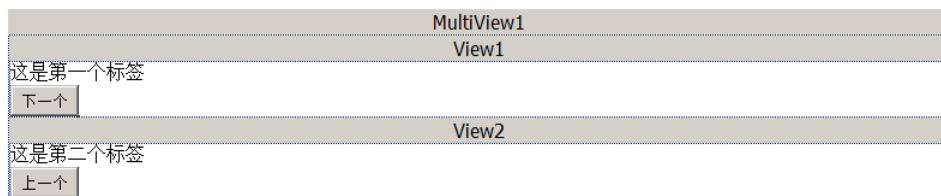


图 5-39 为每个 View 编写不同的应用

`MultiView` 和 `View` 控件也可以实现导航效果，可以通过编程指定 `MultiView` 的 `ActiveViewIndex` 属性显示相应的 `View` 控件。

注意：在 `MultiView` 控件中，第一个被放置的 `View` 控件的索引为 0 而不是 1，后面的 `View` 控件的索引依次递增。

## 5.14 表控件（Table）

在 ASP.NET 中，也提供了表控件（`Table`）来提供可编程的表格服务器控件。表中的行可以通过 `TableRow` 创建，而表中的列通过 `TableCell` 来实现，当创建一个表控件时，系统生成代码如下所示。

```
<asp:Table ID="Table1" runat="server" Height="121px" Width="177px">
</asp:Table>
```

上述代码自动生成了一个表控件代码，但是没有生成表控件中的行和列，必须通过 `TableRow` 创建行，通过 `TableCell` 来创建列，示例代码如下所示。

```
<asp:Table ID="Table1" runat="server" Height="121px" Width="177px">
<asp:TableRow>
<asp:TableCell>1.1</asp:TableCell>
<asp:TableCell>1.2</asp:TableCell>
```

```

<asp:TableCell>1.3</asp:TableCell>
<asp:TableCell>1.4</asp:TableCell>
</asp:TableRow>
<asp:TableRow>
<asp:TableCell>2.1</asp:TableCell>
<asp:TableCell>2.2</asp:TableCell>
<asp:TableCell>2.3</asp:TableCell>
<asp:TableCell>2.4</asp:TableCell>
</asp:TableRow>
</asp:Table>

```

上述代码创建了一个两行四列的表，如图 5-40 所示。

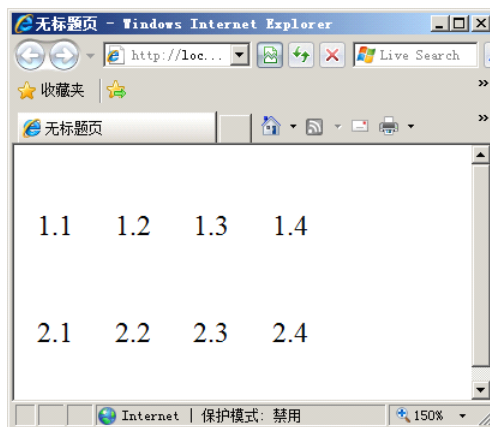


图 5-40 表控件

父 Table 控件支持一些控制整个表的外观的属性，例如字体、背景颜色等，如图 5-41 所示。TableRow 控件和 TableCell 控件也支持这些属性，同样可以用来指定个别的行或单元格的外观，运行后如图 5-42 所示。



图 5-41 Table 的属性设置



图 5-42 TableCell 控件的属性设置

表控件和静态表的区别在于，表控件能够动态的为表格创建行或列，实现一些特定的程序需求。Web 服务器控件中，Table 控件中的行是 TableRow 对象，Table 控件中的列是 TableCell 对象。可以声明这两个对象并初始化，可以为表控件增加行或列，实现动态创建表的程序，HTML 核心代码如下所示。

```

<body style="font-style: italic">
  <form id="form1" runat="server">
    <div>

```



```

<asp:Table ID="Table1" runat="server" Height="121px" Width="177px"
    BackColor="Silver">
  <asp:TableRow>
    <asp:TableCell>1.1</asp:TableCell>
    <asp:TableCell>1.2</asp:TableCell>
    <asp:TableCell>1.3</asp:TableCell>
    <asp:TableCell BackColor="White">1.4</asp:TableCell>
  </asp:TableRow>
  <asp:TableRow>
    <asp:TableCell>2.1</asp:TableCell>
    <asp:TableCell BackColor="White">2.2</asp:TableCell>
    <asp:TableCell>2.3</asp:TableCell>
    <asp:TableCell>2.4</asp:TableCell>
  </asp:TableRow>
</asp:Table>
<br />
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="增加一行" />
</div>
</form>
</body>

```

上述代码中，创建了一个二行一列的表格，同时创建了一个 Button 按钮控件来实现增加一行的效果，cs 核心代码如下所示。

```

namespace _5_14
{
    public partial class _Default : System.Web.UI.Page
    {
        public TableRow row = new TableRow(); //定义一个 TableRow 对象
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Table1.Rows.Add(row); //创建一个新行
            for (int i = 0; i < 4; i++) //遍历四次创建新列
            {
                TableCell cell = new TableCell(); //定义一个 TableCell 对象
                cell.Text = "3."+i.ToString(); //编写 TableCell 对象的文本
                row.Cells.Add(cell); //增加列
            }
        }
    }
}

```

上述代码动态的创建了一行并动态的在该行创建了四列，如图 5-43 所示。单击【增加一列】按钮，系统会在表格中创建新行，运行效果如图 5-44 所示。

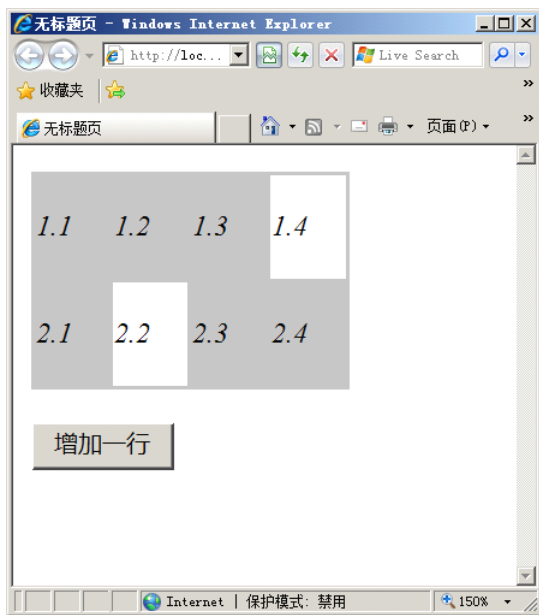


图 5-43 原表格

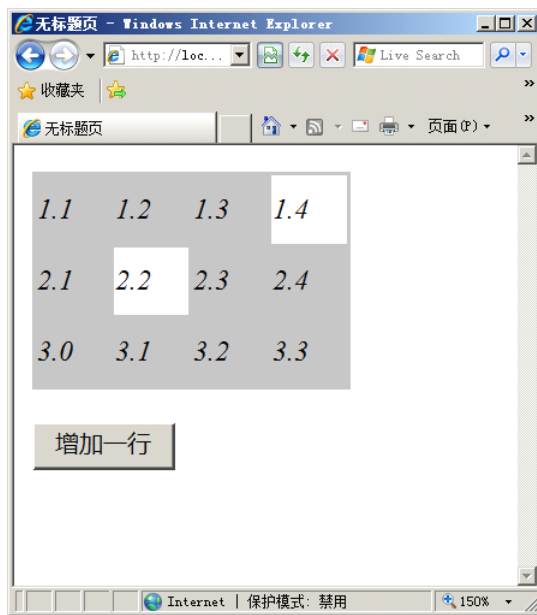


图 5-44 动态创建行和列

在动态创建行和列的时候，也能够修改行和列的样式等属性，创建自定义样式的表格。通常，表不仅用来显示表格的信息，还是一种传统的布局网页的形式，创建网页表格有如下几种形式：

- ❑ HTML 格式的表格：如<table>标记显示的静态表格。
- ❑ HtmlTable 控件：将传统的<table>控件通过添加 runat=server 属性将其转换为服务器控件。
- ❑ Table 表格控件：就是本节介绍的表格控件。

虽然创建表格有以上三种创建方法，但是推荐开发人员在使用静态表格，当不需要对表格做任何逻辑事物处理时，最好使用 HTML 格式的表格，因为这样可以极大的降低页面逻辑、增强性能。

## 5.15 向导控件（Wizard）

在 WinForm 开发中，安装程序会一步一步的提示用户安装，或者在应用程序配置中，同样也有向导提示用户，让应用程序安装和配置变得更加的简单。与之相同的是，在 ASP.NET 中，也提供了一个向导控件，便于在搜集用户信息、或提示用户填写相关的表单时使用。

### 5.15.1 向导控件的样式

当创建了一个向导控件时，系统会自动生成向导控件的 HTML 代码，示例代码如下所示。

```
<asp:Wizard ID="Wizard1" runat="server">
  <WizardSteps>
    <asp:WizardStep runat="server" title="Step 1">
    </asp:WizardStep>
    <asp:WizardStep runat="server" title="Step 2">
    </asp:WizardStep>
  </WizardSteps>
</asp:Wizard>
```

上述代码生成了 Wizard 控件，并在 Wizard 控件中自动生成了 WizardSteps 标签，这个标签规范了

向导控件中的步骤，如图 5-45 所示。在向导控件中，系统会生成 WizardSteps 控件来显示每一个步骤，如图 5-46 所示。

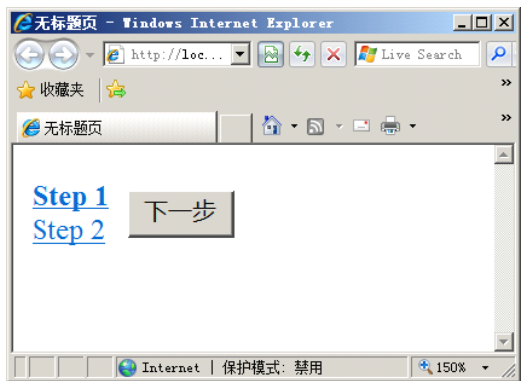


图 5-45 向导控件

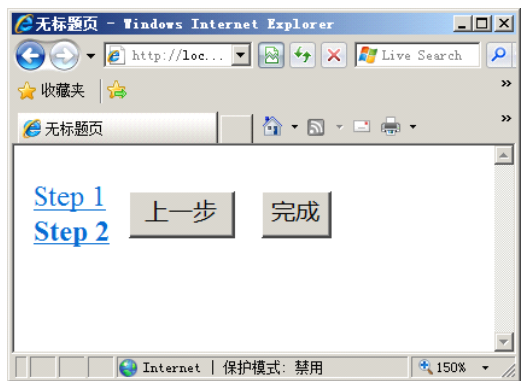


图 5-46 完成后的向导控件

在 ASP.NET 2.0 之前，并没有 Wizard 向导控件，必须创建自定义控件来实现 Wizard 向导控件的效果，如视图控件。而在 ASP.NET 2.0 之后，系统就包含了向导控件，同样该控件也保留到了 ASP.NET 3.5。向导控件能够根据步骤自动更换选项，如当还没有执行到最后一步时，会出现【上一步】或【下一步】按钮以便用户使用，当向导执行完毕时，则会显示完成按钮，极大的简化了开发人员的向导开发过程。

向导控件还支持自动显示标题和控件的当前步骤。标题使用 HeaderText 属性自定义，同时还可以配置 DisplayCancelButton 属性显示一个取消按钮，如图 5-47 所示。不仅如此，当需要让向导控件支持向导步骤的添加时，只需配置 WizardSteps 属性即可，如图 5-48 所示。

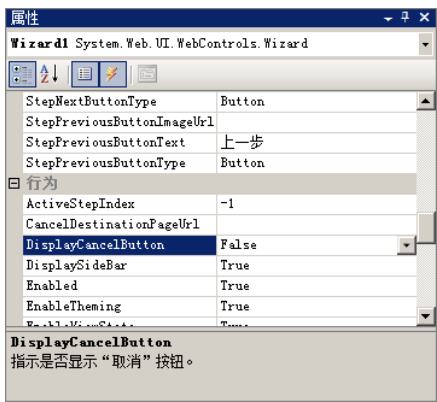


图 5-47 显式“取消”按钮

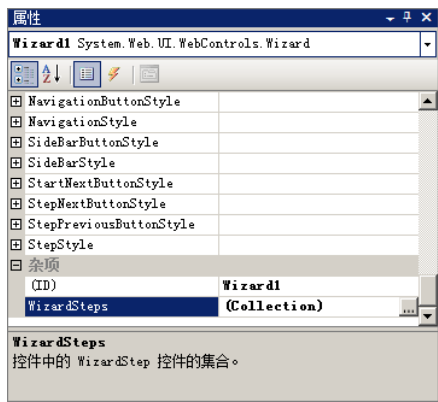


图 5-48 配置步骤

Wizard 向导控件还支持一些模板。用户可以配置相应的属性来配置向导控件的模板。用户可以通过编辑 StartNavigationTemplate 属性、FinishNavigationTemplate 属性、StepNavigationTemplate 属性以及 SideBarTemplate 属性来进行自定义控件的界面设定。这些属性的意义如下所示：

- ❑ StartNavigationTemplate: 该属性指定为 Wizard 控件的 Start 步骤中的导航区域显示自定义内容。
- ❑ FinishNavigationTemplate: 该属性为 Wizard 控件的 Finish 步骤中的导航区域指定自定义内容。
- ❑ StepNavigationTemplate: 该属性为 Wizard 控件的 Step 步骤中的导航区域指定自定义内容。
- ❑ SideBarTemplate: 该属性为 Wizard 控件的侧栏区域中指定自定义内容。

以上属性都可以通过可视化功能来编辑或修改，如图 5-49 所示。



图 5-49 导航控件的模板支持

导航控件还能够自定义模板来实现更多的特定功能，同时导航控件还能够为导航控件的其他区域定义进行样式控制，如导航列表和导航按钮等。

### 5.15.2 导航控件的事件

当双击一个导航控件时，导航控件会自动生成 FinishButtonClick 事件。该事件是当用户完成导航控件时被触发。导航控件页面 HTML 核心代码如下所示。

```
<body>
  <form id="form1" runat="server">
    <asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="2"
      DisplayCancelButton="True" onfinishbuttonclick="Wizard1_FinishButtonClick">
      <WizardSteps>
        <asp:WizardStep runat="server" title="Step 1">
          执行的是第一步</asp:WizardStep>
        <asp:WizardStep runat="server" title="Step 2">
          执行的是第二步</asp:WizardStep>
        <asp:WizardStep runat="server" Title="Step3">
          感谢您的使用</asp:WizardStep>
      </WizardSteps>
    </asp:Wizard>
    <div>
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </div>
  </form>
</body>
```

上述代码为向导控件进行了初始化，并提示用户正在执行的步骤，当用户执行完毕后，会提示感谢您的使用并在相应的文本标签控件中显示“向导控件执行完毕”。当单击向导控件时，会触发 FinishButtonClick 事件，通过编写 FinishButtonClick 事件能够为导航控件进行编码控制，示例代码如下所示。

```
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    Label1.Text = "向导控件执行完毕";
}
```

在执行的过程中，标签文本会显示执行的步骤，如图 5-50 所示。当运行完毕时，Label 标签控件会显示“向导控件执行完毕”，同时向导控件中的文本也会呈现“感谢您的使用”字样。运行结果如图 5-51 所示。



图 5-50 执行第二步

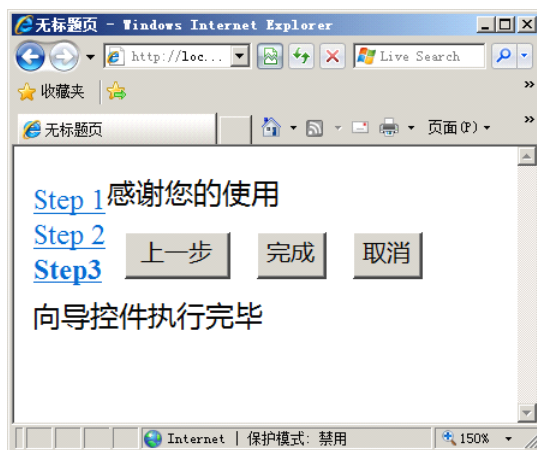


图 5-51 用户单击完成后执行事件

向导控件不仅能够使用 FinishButtonClick 事件，同样也可以使用 PreviousButtonClick 和 FinishButtonClick 事件来自定义【上一步】按钮和【下一步】按钮的行为，同样也可以编写 CancelButtonClick 事件定义单击【取消】按钮时需要执行的操作。

## 5.16 XML 控件

XML 控件可以读取 XML 并将其写入该控件所在的 ASP.NET 网页。XML 控件能够将 XSL 转换应用到 XML，还能够将最终转换的内容输出呈现在该页中。当创建一个 XML 控件时，系统会生成 XML 控件的 HTML 代码，示例代码如下所示。

```
<asp:Xml ID="Xml1" runat="server"></asp:Xml>
```

上述代码实现了简单的 XML 控件，XML 控件还包括两个常用的属性，这两个属性分别如下所示：

- ❑ DocumentSource：应用转换的 XML 文件。
- ❑ TransformSource：用于转换 XML 数据的 XSL 文件。

开发人员可以通过 XML 控件的 DocumentSource 属性提供的 XML，XSL 文件的路径来进行加载，并将相应的代码呈现到控件上，示例代码如下所示。

```
<asp:Xml ID="Xml1" runat="server" DocumentSource="~/XMLFile1.xml"></asp:Xml>
```

上述代码为 XML 控件指定了 DocumentSource 属性，通过加载 XML 文档进行相应的代码呈现，运行后如图 5-52 所示。

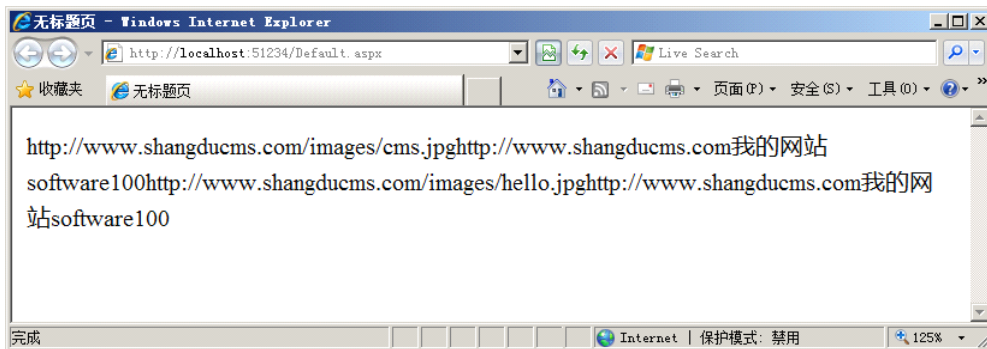


图 5-52 加载 XML 文档

XML 控件不仅能够呈现 XML 文档的内容，还能够进行相应的 XML 的文本操作。在本书的第 14 章中，会详细讲解如何使用 ASP.NET 进行操作 XML。

## 5.17 验证控件

ASP.NET 提供了强大的验证控件，它可以验证服务器控件中用户的输入，并在验证失败的情况下显示一条自定义错误消息。验证控件直接在客户端执行，用户提交后执行相应的验证无需使用服务器端进行验证操作，从而减少了服务器与客户端之间的往返过程。

### 5.17.1 表单验证控件（RequiredFieldValidator）

在实际的应用中，如在用户填写表单时，有一些项目是必填项，例如用户名和密码。在传统的 ASP 中，当用户填写表单后，页面需要被发送到服务器并判断表单中的某项 HTML 控件的值是否为空，如果为空，则返回错误信息。在 ASP.NET 中，系统提供了 RequiredFieldValidator 验证控件进行验证。使用 RequiredFieldValidator 控件能够指定某个用户在特定的控件中必须提供相应的信息，如果不填写相应的信息，RequiredFieldValidator 控件就会提示错误信息，RequiredFieldValidator 控件示例代码如下所示。

```
<body>
  <form id="form1" runat="server">
    <div>
      姓名:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
          ControlToValidate="TextBox1" ErrorMessage="必填字段不能为空"></asp:RequiredFieldValidator>
      <br />
      密码:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
      <br />
      <asp:Button ID="Button1" runat="server" Text="Button" />
    </div>
  </form>
</body>
```

在进行验证时，RequiredFieldValidator 控件必须绑定一个服务器控件，在上述代码中，验证控件 RequiredFieldValidator 控件的服务器控件绑定为 TextBox1，当 TextBox1 中的值为空时，则会提示自定义错误信息“必填字段不能为空”，如图 5-53 所示。

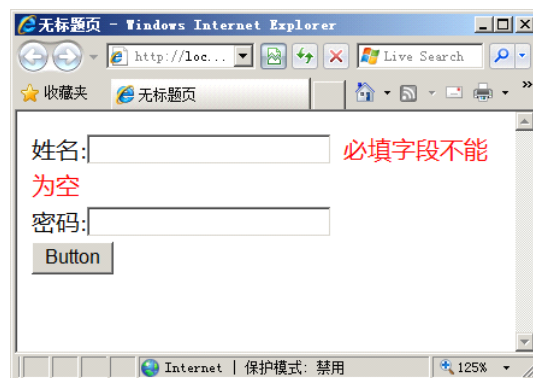


图 5-53 RequiredFieldValidator 验证控件

当姓名选项未填写时，会提示必填字段不能为空，并且该验证在客户端执行。当发生此错误时，用户会立即看到该错误提示而不会立即进行页面提交，当用户填写完成并再次单击按钮控件时，页面才会向服务器提交。

### 5.17.2 比较验证控件（CompareValidator）

比较验证控件对照特定的数据类型来验证用户的输入。因为当用户输入用户信息时，难免会输入错误信息，如当需要了解用户的生日时，用户很可能输入了其他的字符串。CompareValidator 比较验证控件能够比较控件中的值是否符合开发人员的需要。CompareValidator 控件的特有属性如下所示：

- ❑ ControlToCompare：以字符串形式输入的表达式。要与另一控件的值进行比较。
- ❑ Operator：要使用的比较。
- ❑ Type：要比较两个值的数据类型。
- ❑ ValueToCompare：以字符串形式输入的表达式。

当使用 CompareValidator 控件时，可以方便的判断用户是否正确输入，示例代码如下所示。

```
<body>
  <form id="form1" runat="server">
    <div>
      请输入生日:
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      毕业日期:
      <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
      <asp:CompareValidator ID="CompareValidator1" runat="server"
        ControlToCompare="TextBox2" ControlToValidate="TextBox1"
        CultureInvariantValues="True" ErrorMessage="输入格式错误！请改正！"
        Operator="GreaterThan"
        Type="Date">
      </asp:CompareValidator>
      <br />
      <asp:Button ID="Button1" runat="server" Text="Button" />
      <br />
    </div>
  </form>
</body>
```

上述代码判断 TextBox1 的输入的格式是否正确，当输入的格式错误时，会提示错误，如图 5-54 所示。



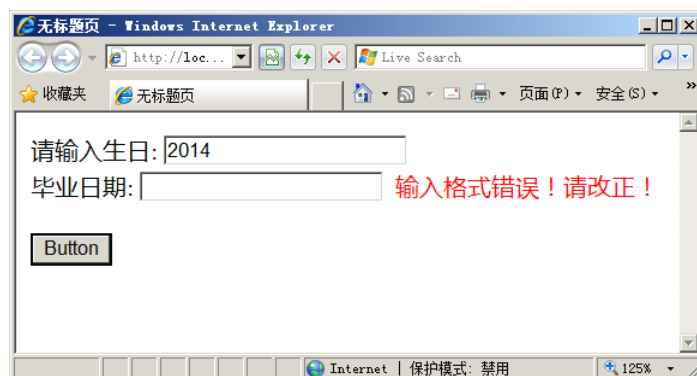


图 5-54 CompareValidator 验证控件

CompareValidator 验证控件不仅能够验证输入的格式是否正确，还可以验证两个控件之间的值是否相等。如果两个控件之间的值不相等，CompareValidator 验证控件同样会将自定义错误信息呈现在用户的客户端浏览器中。

### 5.17.3 范围验证控件（RangeValidator）

范围验证控件（RangeValidator）可以检查用户的输入是否在指定的上限与下限之间。通常情况下用于检查数字、日期、货币等。范围验证控件（RangeValidator）控件的常用属性如下所示。

- ❑ MinimumValue: 指定有效范围的最小值。
- ❑ MaximumValue: 指定有效范围的最大值。
- ❑ Type: 指定要比较的值的数据类型。

通常情况下，为了控制用户输入的范围，可以使用该控件。当输入用户的生日时，今年是 2008 年，那么用户就不应该输入 2009 年，同样基本上没有人的寿命会超过 100，所以对输入的日期的下限也需要进行规定，示例代码如下所示。

```
<div>
    请输入生日:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:RangeValidator ID="RangeValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="超出规定范围，请重新填写"
        MaximumValue="2009/1/1" MinimumValue="1990/1/1" Type="Date">
    </asp:RangeValidator>
    <br />
    <asp:Button ID="Button1" runat="server" Text="Button" />
</div>
```

上述代码将 MinimumValue 属性值设置为 1990/1/1，并能将 MaximumValue 的值设置为 2009/1/1，当用户的日期低于最小值或高于最高值时，则提示错误，如图 5-55 所示。

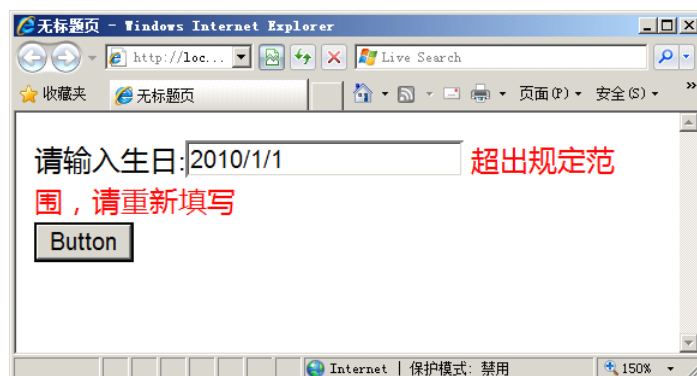


图 5-55 RangeValidator 验证控件

注意: RangeValidator 验证控件在进行控件的值的范围的设定时, 其范围不仅仅可以是一个整数值, 同样还能够是时间、日期等值。

#### 5.17.4 正则验证控件 (RegularExpressionValidator)

在上述控件中, 虽然能够实现一些验证, 但是验证的能力是有限的, 例如在验证的过程中, 只能验证是否是数字, 或者是否是日期。也可能在验证时, 只能验证一定范围内的数值, 虽然这些控件提供了一些验证功能, 但却限制了开发人员进行自定义验证和错误信息的开发。为实现一个验证, 很可能需要多个控件同时搭配使用。

正则验证控件 (RegularExpressionValidator) 就解决了这个问题, 正则验证控件的功能非常的强大, 它用于确定输入的控件的值是否与某个正则表达式所定义的模式相匹配, 如电子邮件、电话号码以及序列号等。

正则验证控件 (RegularExpressionValidator) 常用的属性是 ValidationExpression, 它用来指定用于验证的输入控件的正则表达式。客户端的正则表达式验证语法和服务端的正则表达式验证语法不同, 因为在客户端使用的是 JSript 正则表达式语法, 而在服务器端使用的是 Regex 类提供的正则表达式语法。使用正则表达式能够实现强大字符串的匹配并验证用户的输入的格式是否正确, 系统提供了一些常用的正则表达式, 开发人员能够选择相应的选项进行规则筛选, 如图 5-56 所示。

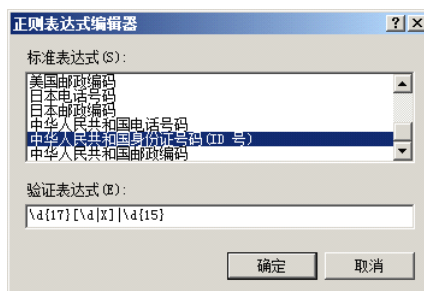


图 5-56 系统提供的正则表达式

当选择了正则表达式后, 系统自动生成的 HTML 代码如下所示。

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
    ControlToValidate="TextBox1" ErrorMessage="正则不匹配,请重新输入!"
    ValidationExpression="\d{17}[\d|X]|\d{15}">
</asp:RegularExpressionValidator>
```

运行后当用户单击按钮控件时，如果输入的信息与相应的正则表达式不匹配，则会提示错误信息，如图 5-57 所示。

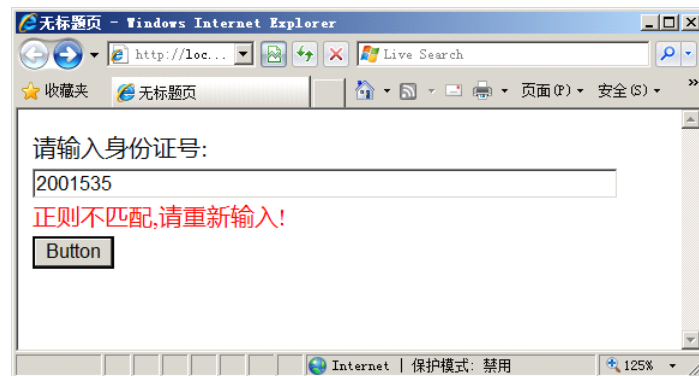


图 5-57 RegularExpressionValidator 验证控件

同样，开发人员也可以自定义正则表达式来规范用户的输入。使用正则表达式能够加快验证速度并在字符串中快速匹配，而另一方面，使用正则表达式能够减少复杂的应用程序的功能开发和实现。

注意：在用户输入为空时，其他的验证控件都会验证通过。所以，在验证控件的使用中，通常需要同表单验证控件（RequiredFieldValidator）一起使用。

### 5.17.5 自定义逻辑验证控件（CustomValidator）

自定义逻辑验证控件（CustomValidator）允许使用自定义的验证逻辑创建验证控件。例如，可以创建一个验证控件判断用户输入的是否包含“.”号，示例代码如下所示。

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    args.IsValid = args.Value.ToString().Contains(".");           //设置验证程序，并返回布尔值
}
protected void Button1_Click(object sender, EventArgs e)        //用户自定义验证
{
    if (Page.IsValid)                                           //判断是否验证通过
    {
        Label1.Text = "验证通过";                               //输出验证通过
    }
    else
    {
        Label1.Text = "输入格式错误";                           //提交失败信息
    }
}
```

上述代码不仅使用了验证控件自身的验证，也使用了用户自定义验证，运行结果如图 5-58 所示。

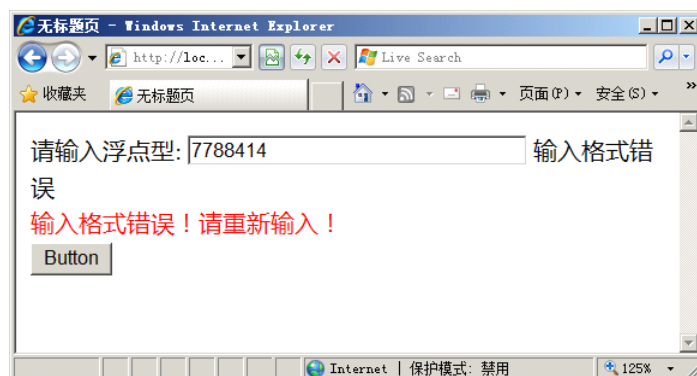


图 5-58 CustomValidator 验证控件

从 CustomValidator 验证控件的验证代码可以看出，CustomValidator 验证控件可以在服务器上执行验证检查。如果要创建服务器端的验证函数，则处理 CustomValidator 控件的 ServerValidate 事件。使用传入的 ServerValidateEventArgs 的对象的 IsValid 字段来设置是否通过验证。

而 CustomValidator 控件同样也可以在客户端实现，该验证函数可用 VBScript 或 Jscript 来实现，而在 CustomValidator 控件中需要使用 ClientValidationFunction 属性指定与 CustomValidator 控件相关的客户端验证脚本的函数名称进行控件中的值的验证。

### 5.17.6 验证组控件（ValidationSummary）

验证组控件（ValidationSummary）能够对同一页面的多个控件进行验证。同时，验证组控件（ValidationSummary）通过 ErrorMessage 属性为页面上的每个验证控件显式错误信息。验证组控件（ValidationSummary）的常用属性如下所示。

- ❑ DisplayMode: 摘要可显示为列表，项目符号列表或单个段落。
- ❑ HeaderText: 标题部分指定一个自定义标题。
- ❑ ShowMessageBox: 是否在消息框中显示摘要。
- ❑ ShowSummary: 控制是显示还是隐藏 ValidationSummary 控件。

验证控件能够显示页面的多个控件产生的错误，示例代码如下所示。

```
<body>
  <form id="form1" runat="server">
    <div>
      姓名:
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="姓名为必填项">
      </asp: RequiredFieldValidator>
      <br />
      身份证:
      <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
      <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="身份证号码错误"
        ValidationExpression="\d{17}[\d|X]\d{15}"></asp:RegularExpressionValidator>
      <br />
      <asp:Button ID="Button1" runat="server" Text="Button" />
      <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
    </div>
  </form>
</body>
```

```
</div>
</form>
</body>
```

运行结果如图 5-59 所示。

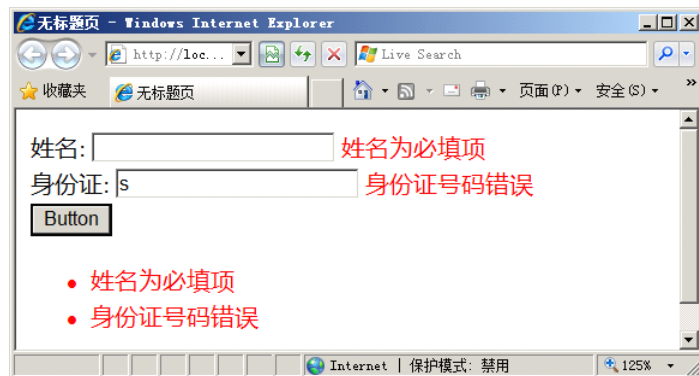


图 5-59 ValidationSummary 验证控件

当有多个错误发生时，ValidationSummary 控件能够捕获多个验证错误并呈现给用户，这样就避免了一个表单需要多个验证时需要使用多个验证控件进行绑定，使用 ValidationSummary 控件就无需为每个需要验证的控件进行绑定。

## 5.18 导航控件

在网站制作中，常常需要制作导航来让用户能够更加方便快捷的查阅到相关的信息和资讯，或能跳转到相关的版块。在 Web 应用中，导航是非常重要的。ASP.NET 提供了站点导航的一种简单的方法，即使用站点图形站点导航控件 SiteMapPath、TreeView、Menu 等控件。

导航控件包括 SiteMapPath、TreeView、Menu 三个控件，这三个控件都可以在页面中轻松建立导航。这三个导航控件的基本特征如下所示：

- ❑ SiteMapPath: 检索用户当前页面并显示层次结构的控件。这使用户可以导航回到层次结构中的其他页。SiteMap 控件专门与 SiteMapProvider 一起使用。
  - ❑ TreeView: 提供纵向用户界面以展开和折叠网页上的选定节点，以及为选定像提供复选框功能。并且 TreeView 控件支持数据绑定。
  - ❑ Menu: 提供在用户将鼠标指针悬停在某一项时弹出附加子菜单的水平或垂直用户界面。
- 这三个导航控件都能够快速的建立导航，并且能够调整相应的属性为导航控件进行自定义。

SiteMapPath 控件使用户能够从当前导航回站点层次结构中较高的页，但是该控件并不允许用户从当前页面向前导航到层次结构中较深的其他页面。相比之下，使用 TreeView 或 Menu 控件，用户可以打开节点并直接选择需要跳转的特定页。这些控件不会像 SiteMapPath 控件一样直接读取站点地图。TreeView 和 Menu 控件不仅可以自定义选项，也可以绑定一个 SiteMapDataSource。TreeView 和 Menu 控件的基本样式如图 5-60 和图 5-61 所示。

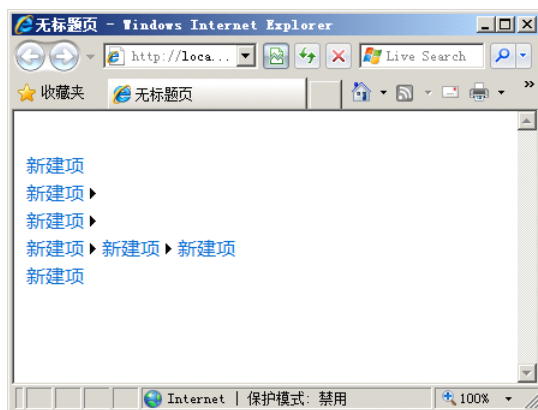


图 5-60 Menu 导航控件

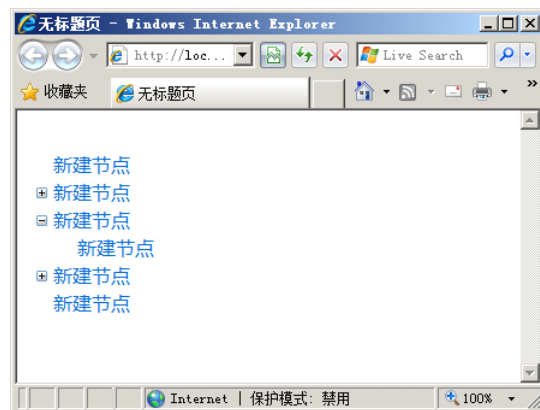


图 5-61 TreeView 导航控件

TreeView 和 Menu 控件生成的代码并不相同，因为 TreeView 和 Menu 控件所实现的功能也不尽相同。TreeView 和 Menu 控件的代码分别如下所示。

```
<asp:Menu ID="Menu1" runat="server">
  <Items>
    <asp:MenuItem Text="新建项" Value="新建项"></asp:MenuItem>
    <asp:MenuItem Text="新建项" Value="新建项">
      <asp:MenuItem Text="新建项" Value="新建项"></asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem Text="新建项" Value="新建项">
      <asp:MenuItem Text="新建项" Value="新建项"></asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem Text="新建项" Value="新建项">
      <asp:MenuItem Text="新建项" Value="新建项">
        <asp:MenuItem Text="新建项" Value="新建项"></asp:MenuItem>
      </asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem Text="新建项" Value="新建项"></asp:MenuItem>
  </Items>
</asp:Menu>
```

上述代码声明了一个 Menu 控件，并添加了若干节点。

```
<asp:TreeView ID="TreeView1" runat="server">
  <Nodes>
    <asp:TreeNode Text="新建节点" Value="新建节点"></asp:TreeNode>
    <asp:TreeNode Text="新建节点" Value="新建节点">
      <asp:TreeNode Text="新建节点" Value="新建节点"></asp:TreeNode>
    </asp:TreeNode>
    <asp:TreeNode Text="新建节点" Value="新建节点">
      <asp:TreeNode Text="新建节点" Value="新建节点"></asp:TreeNode>
    </asp:TreeNode>
    <asp:TreeNode Text="新建节点" Value="新建节点">
      <asp:TreeNode Text="新建节点" Value="新建节点"></asp:TreeNode>
    </asp:TreeNode>
    <asp:TreeNode Text="新建节点" Value="新建节点"></asp:TreeNode>
  </Nodes>
</asp:TreeView>
```

---

上述代码声明了一个 **TreeView** 控件，并添加了若干节点。

从上面的代码和运行后的实例图可以看出，**TreeView** 和 **Menu** 控件有一些区别，这些具体区别如下所示：

- ❑ **Menu** 展开时，是弹出形式的展开，而 **TreeView** 控件则是就地展开。
- ❑ **Menu** 控件并不是按需下载，而 **TreeView** 控件则是按需下载的。
- ❑ **Menu** 控件不包含复选框，而 **TreeView** 控件包含复选框。
- ❑ **Menu** 控件允许编辑模板，而 **TreeView** 控件不允许模板编辑。
- ❑ **Menu** 在布局上是水平和垂直，而 **TreeView** 只是垂直布局。
- ❑ **Menu** 可以选择样式，而 **TreeView** 不行。

开发人员在网站开发的时候，可以通过使用导航控件来快速的建立导航，为浏览者提供方便，也为网站做出信息指导。在用户的使用中，通常情况下导航控件中的导航值是不能被用户所更改的，但是开发人员可以通过编程的方式让用户也能够修改站点地图的节点。

## 5.19 其他控件

在 ASP.NET 中，除了以上常用的一些基本控件以外，还有一些其他基本控件，虽然在应用程序开发中并不经常使用，但是在特定的程序开发中，还是需要使用到这些基本的控件进行特殊的应用程序开发和逻辑处理。

### 5.19.1 隐藏输入框控件（HiddenField）

**HiddenField** 控件就是隐藏输入框控件，用来保存那些不需要显示在页面上的对安全性要求不高的数据。隐藏输入框控件作为 `<input type="hidden">` 元素呈现在 HTML 页面。由于 **HiddenField** 隐藏输入框控件的值会呈现在客户端浏览器，所以对于安全性较高的数据，并不推荐将它保存在隐藏输入框控件中。隐藏输入框控件的值通过 **Value** 属性保存，同时也可以通过代码来控制 **Value** 的值，利用隐藏输入框对页面的值进行传递，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = HiddenField1.Value;           //获取隐藏输入框控件的值
}
```

上述代码通过 **Value** 属性获取一个隐藏输入框的值，如图 5-62 所示。单击后如图 5-63 所示。



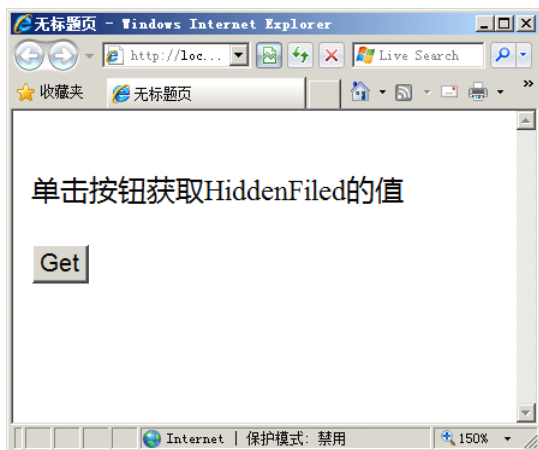


图 5-62 HiddenFiled 的值被隐藏

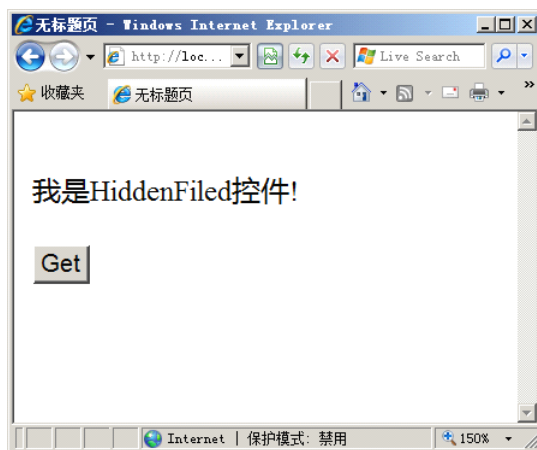


图 5-63 HiddenFiled 的值被获取

HiddenFiled 是通过 HTTP 协议进行参数传递的，所以当打开新的窗体或者使用 `method=get` 都无法使用 HiddenFiled 隐藏输入框控件。同时，隐藏输入框控件还能初始化或保存一些安全性不高的数据。当双击隐藏输入框控件时，系统会自动生成 `ValueChanged` 事件代码段，当隐藏输入框控件内的值被改变时则触发该事件，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    HiddenField1.Value = "更改了值"; //更改隐藏输入框控件的值
}
protected void HiddenField1_ValueChanged(object sender, EventArgs e) //更改将触发此事件
{
    Label1.Text = "值被更改了,并被更改成\" + HiddenField1.Value + "\";
}
```

上述代码创建了一个 `ValueChanged` 事件，并当隐藏输入框控件的值被更改时，如图 5-64 所示。单击【change】按钮后会触发按钮事件，运行结果如图 5-65 所示。

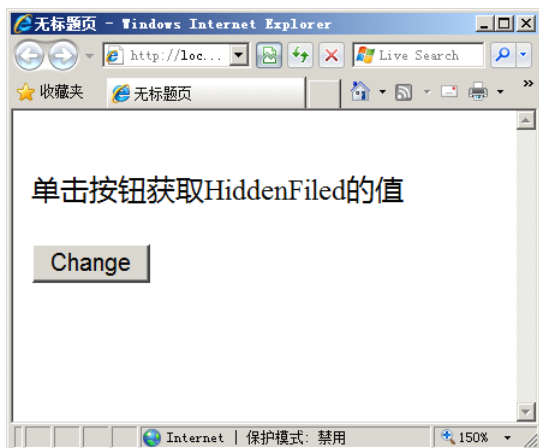


图 5-64 更新前

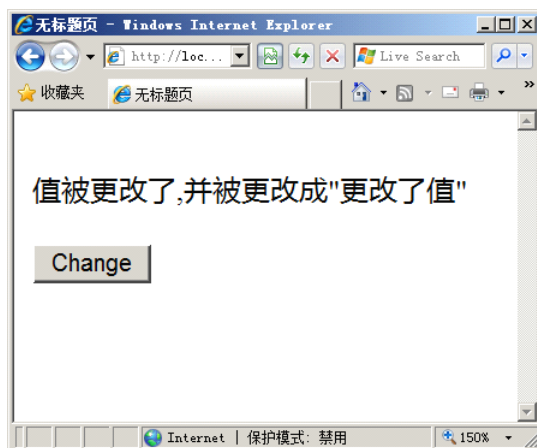


图 5-65 更新后

## 5.19.2 图片热点控件（ImageMap）

ImageMap 控件是一个让你可以在图片上定义热点（HotSpot）区域的服务器控件。用户可以通过点

击这些热点区域进行回发（PostBack）操作或者定向（Navigate）到某个 URL 位址。该控件一般用在需要对某张图片的局部范围进行互动操作。ImageMap 控件主要由两个部分组成，第一部分是图像。第二部分是作用点控件的集合。其主要属性有 HotSpotMode、HotSpots，具体如下所示。

**1. HotSpotMode（热点模式）常用选项**

- ☐ NotSet: 未设置项。虽然名为未设置，但其实默认情况下会执行定向操作，定向到你指定的 URL 位址去。如果你未指定 URL 位址，那默认将定向到自己的 Web 应用程序根目录。
- ☐ Navigate: 定向操作项。定向到指定的 URL 位址去。如果你未指定 URL 位址，那默认将定向到自己的 Web 应用程序根目录。
- ☐ PostBack: 回发操作项。点击热点区域后，将执行后部的 Click 事件。
- ☐ Inactive: 无任何操作，即此时形同一张没有热点区域的普通图片。

**2. HotSpots（图片热点）常用属性**

该属性对应着 System.Web.UI.WebControls.HotSpot 对象集合。HotSpot 类是一个抽象类，它之下有 CircleHotSpot（圆形热区）、RectangleHotSpot（方形热区）和 PolygonHotSpot（多边形热区）三个子类。实际应用中，都可以使用上面三种类型来定制图片的热点区域。如果需要使用到自定义的热点区域类型时，该类型必须继承 HotSpot 抽象类。同时，ImageMap 最常用的事件有 Click，通常在 HotSpotMode 为 PostBack 时用到。当需要设置 HotSpots 属性时，可以可视化设置，如图 5-66 所示。



图 5-66 可视化设置 HotSpots 属性

当可视化完毕后，系统会自动生成 HTML 代码，核心代码如下所示。

```
<asp:ImageMap ID="ImageMap1" runat="server" HotSpotMode="PostBack"
    ImageUrl="~/images/mobile.jpg" onclick="ImageMap1_Click">
    <asp:CircleHotSpot Radius="15" X="15" Y="15" HotSpotMode="PostBack" PostBackValue="0" />
    <asp:CircleHotSpot Radius="100" X="15" Y="15" HotSpotMode="PostBack" PostBackValue="1" />
    <asp:CircleHotSpot Radius="300" X="15" Y="15" HotSpotMode="PostBack" PostBackValue="2" />
</asp:ImageMap>
```

上述代码还添加了一个 Click 事件，事件处理的核心代码如下所示。

```
protected void ImageMap1_Click(object sender, ImageMapEventArgs e)
{
    string str="";
    switch (e.PostBackValue)
    {
        case "0":
            str = "你点击了 1 号位置，图片大小将变为 1 号"; break;
    }
}
```

```

        case "1":
            str = "你点击了 2 号位置，图片大小将变为 3 号"; break;
        case "2":
            str = "你点击了 3 号位置，图片大小将变为 3 号"; break;
    }
    Label1.Text = str;
    ImageMap1.Height = 120*(Convert.ToInt32(e.PostBackValue)+1);    //更改图片的大小
}

```

上述代码通过获取 ImageMap 中的 CricleHotSpot 控件中的 PostBackVlue 值来获取传递的参数，如图 5-67 所示。当获取到传递的参数时，可以通过参数做相应的操作，如图 5-68 所示。



图 5-67 单击图片变大



图 5-68 单击图片变小

### 5.19.3 静态标签控件（Lieral）

通常情况下 Lieral 控件无需添加任何 HTML 元素即可将静态文本呈现在网页上。与 Label 不同的是，Label 控件在生成 HTML 代码时，会呈现<span>元素。而 Lieral 控件不会向文本中添加任何 HTML 代码。如果开发人员希望文本和控件直接呈现在页面中而不使用任何附加标记时，推荐使用 Lieral 控件。

与 Label 不同的是，Lieral 控件有一个 Mode 属性，用来控制 Lieral 控件中的文本的呈现形式。当 HTML 代码被输出到页面的时候，会以解释后的 HTML 形式输出。例如图片代码，在 HTML 中是以<img src="">的形式显示的，输出到 HTML 页面后，会被显示成一个图片。

在 Label 中，Label 可以作为一段 HTML 代码的容器，当输出时，Label 控件所呈现的效果是 HTML 被解释后的样式。而 Lieral 可以通过 Mode 属性来选择输出的是 HTML 样式还是 HTML 代码，核心代码如下所示。

```

namespace _5_17
{
    public partial class Lieral : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string str = "<span style=\"color:red\">大家好</span>，您现在查看的是 HTML 样式。";//HTML 字符
            Lieral1.Text = str +

```

```

"<div style=\"border-top:1px dashed #ccc;background:gray\">单击按钮查看 HTML 代码</div>";
Label1.Text = str;                                //赋值 Label
}
protected void Button1_Click(object sender, EventArgs e)
{
    Literal1.Mode = LiteralMode.Encode;            //转换显示的模式
}
}

```

上述代码将一个 HTML 形式的字符串分别赋值给 Literal 和 Label 控件，并通过转换查看赋值的源代码，运行结果如图 5-69 和图 5-70 所示。

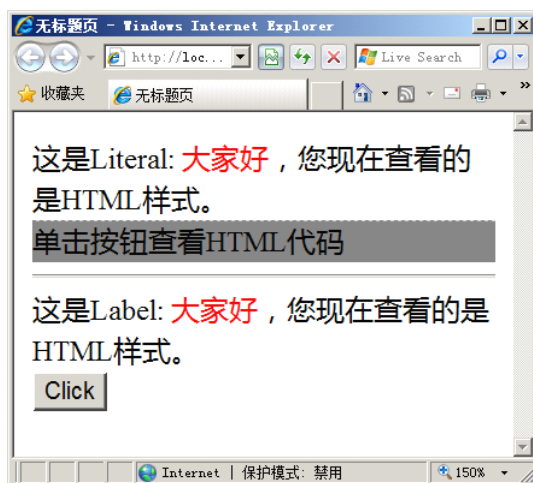


图 5-69 Literal 控件直接显式 HTML 样式

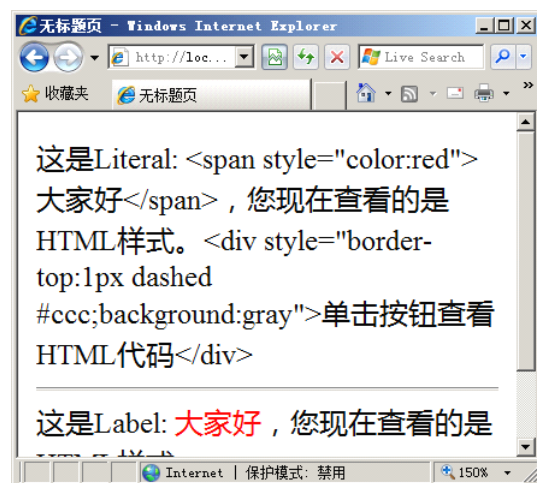


图 5-70 Literal 控件显式 HTML 代码

当点击了按钮后，更改了 Literal 的模式之后，Literal 中的 HTML 文本被直接显示，Literal 的具有三种模式，具体如下：

- ☐ Transform: 添加到控件中的任何标记都将进行转换，以适应请求浏览器的协议。
- ☐ PassThrough: 添加到控件中的任何标记都将按照原样输出在浏览器中。
- ☐ Encode: 添加到控件中的任何标记都将使用 HtmlEncode 方法进行编码，该方法将把 HTML 编码转换为其文本表示形式。

注意：PassThrough 模式和 Transform 模式在通常情况下，呈现的效果并没有区别。

#### 5.19.4 动态缓存更新控件(Substitution)

在 ASP.NET 中，缓存的使用能够极大的提高网站的性能，降低服务器的压力。而通常情况下，对 ASP.NET 整个页面的缓存是没有任何意义的，这样经常会给用户带来疑惑。Substitution 动态缓存更新控件允许用户在页上创建一些区域，这些区域可以用动态的方式进行更新，然后集成到缓存页。

Substitution 动态缓存更新控件将动态内容插入到缓存页中，并且不会呈现任何 HTML 标记。用户可以将控件绑定到页上或父用户控件上的方法，自行创建静态方法，以返回要插入到页面中的任何信息。同时，要使用 Substitution 控件，则必须符合以下标准：

- ☐ 此方法被定义为静态方法。
- ☐ 此方法接受 HttpContext 类型的参数。

❑ 此方法返回 String 类型的值。

在 ASP.NET 页面中, 为了减少用户与页面的交互中数据库的更新, 可以对 ASP.NET 页面进行缓存, 缓存代码可以使用页面参数的@OutputCache, 示例代码如下所示。

```
<%@ Page
Language="C#" AutoEventWireup="true" CodeBehind="Substitution.aspx.cs" Inherits="_5_17.Substitution" %>
<%@ OutputCache Duration="100" VaryByParam="none" %>           //增加一个页面缓存
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            当前的时间为: <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            (有缓存)<br />
            当前的时间为: <asp:Substitution ID="Substitution1" runat="server" MethodName="GetTimeNow"/>
            (动态更新)</div>
        </form>
    </body>
</html>
```

执行事件操作的 cs 页面核心代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString();           //页面初始化时, 当前时间赋值给 Label1 标签
}
protected static string GetTimeNow(HttpContext con) //注意事件的格式
{
    return DateTime.Now.ToString();                 //Substitution 控件执行的方法
}
```

上述代码对 ASP.NET 页面进行了缓存, 当用户访问页面时, 除了 Substitution 控件的区域以外区域都会被缓存, 而使用了 Substitution 控件局部在刷新后会进行更新, 运行结果如图 5-71 所示。

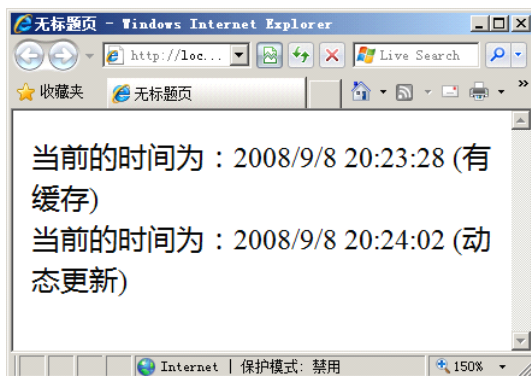


图 5-71 Substitution 动态更新

如运行结果可见, 没有使用 Substitution 控件的区域, 当页面再次被请求时, 会直接在缓存中执行。而 Substitution 控件区域内的值并不会缓存。在每次刷新时, 页面将进行 Substitution 控件的区域的局部

---

的动态更新。

## 5.20 小结

本章讲解了 ASP.NET 中常用的控件，对于这些控件，能够极大的提高开发人员的效率，对于开发人员而言，能够直接拖动控件来完成应用的目的。虽然控件是非常的强大，但是这些控件却制约了开发人员的学习，人们虽然能够经常使用 ASP.NET 中的控件来创建强大的多功能网站，却不能深入的了解控件的原理，所以对这些控件的熟练掌握，是了解控件的原理的第一步。本章还介绍了：

- 控件的属性：介绍了控件的属性。
- 简单控件：介绍了标签控件等简单控件。
- 文本框控件：介绍了文本框控件。
- 按钮控件：介绍了按钮控件的实现和按钮事件的运行过程。
- 单选控件和单选组控件：介绍了单选控件和单选组控件。
- 复选框控件和复选组控件：介绍了复选框控件和复选组控件。

这些控件为 ASP.NET 应用程序的开发提供了极大的便利，在 ASP.NET 控件中，不仅仅包括这些基本的服务器控件，还包括高级的数据源控件和数据绑定控件用于数据操作，但是在了解 ASP.NET 高级控件之前，需要熟练的掌握基本控件的使用。