第 3 章　常用控件应用

学习目标：

(1) 掌握常用控件属性设置。

(2) 掌握常用控件的常见事件实现。

3.1　知识梳理

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。

3.1.1　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相应的控件后，属性栏中会简单地介绍该属性的作用，如图 3-1 所示。

“属性”对话框用来设置控件的属性，当页面初始化时，控件的这些属性将被应用到控件。控件的属性也可以通过编程的方式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。

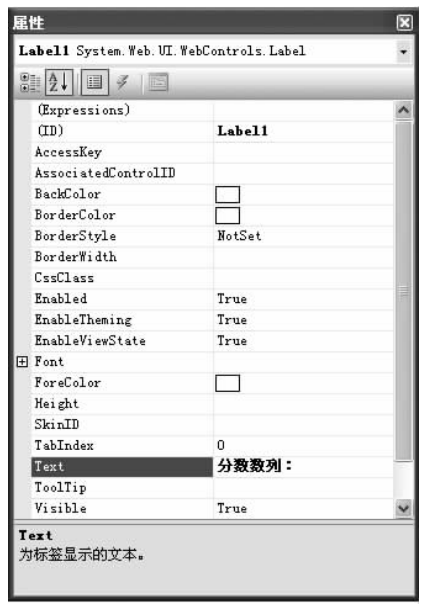

图 3-1　控件的属性

3.1.2 标签控件

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，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。

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”。

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。

3.1.3 文本框控件

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。

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示的行数。
- 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，如果用户提交表单，或者执行相应的方法，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。

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 3 种文本框的使用方法。

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：

```
//单击按钮时触发的事件
protected void Button1_Click(object sender,EventArgs e)
{
    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，当单击按钮时，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。

同样，文本框内容发生变化，会触发 TextChange 事件。而在运行时，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：

```
//文本框事件
protected void TextBox1_TextChanged(object sender,EventArgs e)
{
    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。

3.1.4 按钮控件

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。注意，按

钮控件是非常重要的。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3类按钮控件，分别为Button、LinkButton、ImageButton。

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如下所示。

- 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3种按钮，示例代码如下所示：

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

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。

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。

3. Command 命令事件

在按钮控件中，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 3-2 所示。

图 3-2　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)
{
    //如果 CommandName 属性的值为 Show,则运行下面代码
    if(e.CommandName=="Show")
    {
        //CommandArgument 属性的值赋值给 Label1
        Label1.Text=e.CommandArgument.ToString();
    }
}
```

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。

3.1.5 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，只能在有限的选项中进行选择。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。

1. 单选控件

单选控件(RadioButton)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。

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。

2. 单选组控件

与单选控件相同，单选组控件(RadioButtonList)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 3-3 所示。

图 3-3　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。
- DataTextFiled：提供项文本的数据源中的字段。
- DataTextFormat：应用于文本字段的格式。
- DataValueFiled：数据源中提供项值的字段。
- Items：列表中项的集合。
- RepeatColumn：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
}
```

3.1.6 复选框控件

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(CheckBox)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。当双击复选框控件时，系统会自动生成事件。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”。当选择选框 2 时，则输出“选框 2 被选中，并且字体变大”。

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。

3.1.7 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>
```

在上述代码中，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)            //判断是否被选中
    {
        Label1.Font.Size=FontUnit.XSmall;          //更改字体大小
    }
}
```

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，当选择不同的选项时，字体的大小也不相同。

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。

3.1.8 列表控件

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输入，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。

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信息。因为性别除了男就是女，输入其他信息说明这个信息是错误或者是无效的。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 SelectedIndexChanged 事件。系统会将更改标签 1 中的文本。

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 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: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。

上面的程序同样实现了 DropDownList 中程序的效果。不同的是，如果需要实现让用户选择多个 ListBox 项，只需要设置 SelectionMode 属性为 Multiple 即可。

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。当用户每选一项的时候，就会触发该事件。

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。

3.1.9 图像控件

图像控件(Image)用来在 Web 窗体中显示图像，图像控件常用的属性如下。

- AlternateText：在图像无法显式时显示的备用文本。
- 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：在工具提示中显示的文本。
- GenerateEmptyAlternateText：如果将此属性设置为 true，则呈现图片的 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。

3.1.10 超链接控件

超链接控件(HyperLink)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示还是以图片文件显示，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

在前面小节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。

3.1.11 面板控件

面板控件(Panel)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。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。

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。

GroupText 属性能够进行 Panel 控件的样式呈现，通过编写 GroupText 属性能够更加清晰地让用户了解 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。

3.1.12 表单验证控件

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，系统提供了表单验证控件(RequiredFieldValidator)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 3-4 所示。

图 3-4 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。

3.1.13 比较验证控件

比较验证控件(CompareValidator)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,用户很可能输入其他字符串。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 的输入格式是否正确，当输入的格式错误时，会提示错误，如图 3-5 所示。

图 3-5 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。

3.1.14 范围验证控件

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3-6所示。

图3-6 RangeValidator验证控件

注意：RangeValidator验证控件在进行控件的值的范围设定时，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。

3.1.15 正则验证控件

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强大，它用于确定输入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 3-7 所示。

图 3-7　系统提供的正则表达式

当选择正则表达式后，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 3-8 所示。

图 3-8　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。

3.2 任务实现

3.2.1 任务1：带有头像的留言板

【任务描述】

设计如图3-9所示的页面，当选择下拉列表序号，显示相应的图像；当单击“发表留言”按钮，则获取页面的相应信息。

图3-9 页面视图

【任务实施】

(1) 给工程添加一个文件夹，命名为images，并添加若干图片文件，命名为01.jpg、02.jpg…。

(2) 页面设计源代码如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title>带有头像的留言板</title></head>
<body>
    <form id="form1" runat="server">
    <div>
        <table style="margin-top: 0px;margin-left: 0px;clip: rect(auto auto
        auto auto);">
            <tr>
```