第 5 章

ASP. NET 高级控件

第 4 章介绍的 ASP. NET 控件基本上都能在传统 HTML 控件中找到其原型，但是将大部分处理操作都转移到了服务器端进行，这样可以得到更强的控制能力和编程方便。除第 4 章介绍的控件之外，ASP. NET 还提供了大量功能更完整、更有针对性的控件，本书称为高级控件。高级控件一般都跟具体的功能有关，使用一个高级控件就能够完成一项任务的核心功能，这样就可以较大地减少编程工作量。

本章仅选择性地介绍 Calendar 控件、FileUpload 控件、Wizard 控件、PlaceHolder 控件、AdRotator 控件和验证控件等。

5. 1　Calendar 控件

使用 Calendar 控件可以显示和选择日期，并可在日历网格中显示与特定日期关联的如日程、约会等其他信息。

5. 1. 1　Calendar 控件基本概念

以前开发基于浏览器的应用程序时，有关日期型数据的操作，如输入、选择等，因为涉及格式、初值、校验等多方面的内容，程序员往往需要花费大量的精力对其进行处理。为了解决这个问题，许多天才程序员开发了封装良好的脚本控件提供给大家，但这些控件外观、接口、功能各异，继承、定制都很困难。ASP. NET 提供了 Calendar 控件，很好地解决了这个问题，使与时间有关的编程不再困难。

Calendar 控件是一个功能丰富的控件，很多与日期有关的功能都可以以该控件为基础创建。Calendar 控件本身的功能主要包括：

- 显示一个日历，包括一个月的详细日历和其他一些相关信息。
- 允许用户选择一天、一周或一个月。
- 允许用户移到下一个月或上一个月。
- 以编程方式控制选定日期的显示。

创建一个名为 Calendar 的网站及其默认主页。

从工具箱中拖一个 Calendar 控件到页面上，页面上会增加如下代码：

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

执行该页面，效果如图 5-1 所示。

图 5-1　Calendar 控件的执行效果(一)

这个界面已经很专业了，而我们除了一次拖动之外还没有修改任何代码。

Calendar 控件可以通过属性和事件来进行定制。从表 5-1 可以看出，Calendar 控件提供了丰富的属性和事件处理，使编程人员可以控制 Calendar 控件的几乎所有细节。

表 5-1　Calendar 控件常用的属性和事件

属 性 名 称	说　　明
Caption	日历标题
CaptionAlign	日历标题的对齐方式
CellPadding	单元格的内容和单元格的边框之间的间隔，以像素为单位
DayHeaderStyle	显示一周中某天的部分的样式属性
DayNameFormat	一周中各天的名称格式
DayStyle	本月日期的样式
FirstDayOfWeek	第一列显示一周中的哪天
NextMonthText	“下一月”导航元素的文本
NextPrevFormat	标题部分中，“下一月”和“上一月”导航元素的格式
NextPrevStyle	“下一月”和“上一月”导航元素的样式
OtherMonthDayStyle	非本月日期的样式
PrevMonthText	“上一月”导航元素的文本
SelectedDate	选定的日期
SelectedDates	System. DateTime 对象的集合，这些对象表示 Calendar 控件上的选定日期
SelectedDayStyle	选定日期的样式
SelectionMode	日期选择模式，该模式指定用户可以选择单日、一周还是整月
SelectMonthText	选择器列中，月份选择元素的文本
SelectorStyle	周和月选择器列的样式
SelectWeekText	选择器列中，周选择元素的文本
ShowDayHeader	是否显示一周中各天的标头
ShowGridLines	是否用网格线分隔 Calendar 控件上的日期

续表

属性名称	说　明
ShowNextPrevMonth	是否在标题部分显示“下一月”和“上一月”导航元素
ShowTitle	是否显示标题部分
TitleFormat	标题部分的格式
TitleStyle	标题部分的样式
TodayDayStyle	“今天”日期的样式
TodaysDate	今天的日期值
VisibleDate	要在 Calendar 控件上显示的月份
WeekendDayStyle	周末日期的样式
事件名称	**说　明**
DayRender	当为 Calendar 控件在控件层次结构中创建每一天时发生
SelectionChanged	当用户单击日期选择器选择了一天、一周或整月时发生
VisibleMonthChanged	当用户单击标题部分的“下一月”或“上一月”导航元素时发生

5.1.2　改变 Calendar 控件的外观

可以通过简单地改变 Calendar 控件的属性来得到丰富的外观表现形式。

在页面上再增加一个 Calendar 控件，可参照下面实例代码直接修改页面代码，也可以在设计视图中逐项修改属性值。实例代码如下。

```
<asp:Calendar ID="Calendar2" runat="server" BackColor="#C0FFFF"
    BorderColor="#00C0C0" BorderWidth="1px" Font-Names="Verdana"
    Font-Size="9pt" ForeColor="Black" Height="100px"
    NextPrevFormat="FullMonth" Width="300px" FirstDayOfWeek="Monday"
    SelectionMode="DayWeekMonth" SelectMonthText="月 &gt;"
    SelectWeekText="周 &gt;" ShowGridLines="True">
    <SelectedDayStyle BackColor="#333399" ForeColor="White" />
    <TodayDayStyle BackColor="#CCCCCC" />
    <OtherMonthDayStyle ForeColor="#999999" />
    <NextPrevStyle Font-Bold="True" Font-Size="8pt"
        ForeColor="#333333" VerticalAlign="Bottom" />
    <DayHeaderStyle Font-Bold="True" Font-Size="8pt" />
    <TitleStyle BackColor="White" BorderColor="Black" BorderWidth="4px"
        Font-Bold="True" Font-Size="12pt" ForeColor="#333399" />
</asp:Calendar>
```

在上面的实例中配置了很多属性，在此仅详细介绍一下 SelectionMode 属性的相关

内容。通过改变 SelectionMode 属性，可以控制日历的选择模式，该属性的可选值如下。

- Day(默认值)：允许选择单个日期。
- DayWeek：允许选择单个日期或整周。
- DayWeekMonth：允许选择单个日期、整周或整月。
- None：不允许选择日期，只能导航。

在本例中将 SelectionMode 属性设为了 DayWeekMonth，并设置 SelectMonthText＝"月>" SelectWeekText＝"周 >"，用户就可以通过日历左部的“月＞”超链选择整月，或通过“周＞”超链选择整周了。

页面执行效果如图 5-2 所示。

这仅仅是改变外观的一个例子，Calendar 控件上还有很多发挥空间。

七月	2012年8月						九月
月>	周一	周二	周三	周四	周五	周六	周日
周>	30	31	1	2	3	4	5
周>	6	7	8	9	10	11	12
周>	13	14	15	16	17	18	19
周>	20	21	22	23	24	25	26
周>	27	28	29	30	31	1	2
周>	3	4	5	6	7	8	9

图 5-2　Calendar 控件的执行效果(二)

5.1.3　对 Calendar 控件编程

Calendar 控件提供了 3 个特有的可编程事件，分别在不同的时机触发。通过对它们进行编程，可实现具有极强用户交互性的日期相关功能。本节仅介绍 SelectionChanged 事件的编程。当用户在 Calendar 控件中选择一天、整周或整月时，将触发 SelectionChanged 事件。

在 5.1.2 节的页面上再增加 3 个 Label 控件。在设计视图中双击日历控件，系统会自动打开源代码文件并将光标停留在函数 Calendar2_SelectionChanged()内。

此时页面代码的相关部分如下。

```
<asp:Calendar ID="Calendar2"…
    OnSelectionChanged="Calendar2_SelectionChanged">
    …
</asp:Calendar>
<asp:Label ID="Label1" runat="server"></asp:Label><br />
<asp:Label ID="Label2" runat="server"></asp:Label><br />
<asp:Label ID="Label3" runat="server"></asp:Label><br />
```

在函数 Calendar2_SelectionChanged()内输入如下代码。

```
Label1.Text="今天的日期是:"+
    Calendar2.TodaysDate.ToShortDateString();
if(Calendar2.SelectedDate !=DateTime.MinValue)
    Label2.Text="选择的开始日期是:"+
        Calendar2.SelectedDate.ToShortDateString();
Label3.Text="选择的天数是:"+
    Calendar2.SelectedDates.Count.ToString();
```

页面执行时显示当月的日历，当从左侧的选择器列中单击“周＞”超链选择一周时，页面重新加载，界面如图 5-3 所示。

图 5-3　Calendar 控件的执行效果(三)

在本书应用实例(见第 13 章)中，使用 Calendar 控件实现日程安排功能。

5.2　FileUpload 控件

应用程序中经常需要将文件上传到服务器。如本书的应用实例中，教师就需要将课件上传到服务器供学生下载学习。

以前可以使用 HTML 控件＜input id＝"File1" type＝"file" /＞来完成文件上传功能。ASP.NET 提供了 FileUpload 控件，可以达到更好的效果。FileUpload 控件常用的属性和方法如表 5-2 所示。

表 5-2　FileUpload 控件常用的属性和方法

属性名称	说　明
FileBytes	从使用 FileUpload 控件上传的文件返回一个字节数组
FileContent	Stream 对象，它指向上传的文件
FileName	上传文件的名称(不包含此文件在客户端的文件路径)
HasFile	FileUpload 控件中是否包含文件
PostedFile	上传文件的基础 HttpPostedFile 对象
方法名称	**说　明**
Focus	为控件设置输入焦点
SaveAs	将上传文件的内容保存到 Web 服务器上的指定路径

FileUpload 控件在客户端表现为一个文本输入框和一个浏览按钮，供用户选择本地文件。包含单一 FileUpload 控件的页面执行效果如图 5-4 所示。

图 5-4　FileUpload 控件的执行效果

用户选择了要上传的文件后，FileUpload 控件不会自动将该文件上传到服务器。程序员必须显式地

控制文件的提交，例如，可以提供一个“上传”按钮，用户单击该按钮时即可提交上传文件。文件上传到服务器后也不会自动保存，仍然需要程序员编程进行处理。如果提供了一个用于提交文件的按钮，在服务器端处理上传文件的代码就可以放在该按钮的单击事件处理函数中。

FileUpload 控件提供了丰富灵活的文件处理功能。

首先可以通过 HasFile 属性判断是否有上传的文件。如果有上传文件，就可以通过 FileName 属性获得上传文件的名称。

处理文件可以有多种方法。可以调用 FileUpload 控件的 SaveAs 方法将上传文件的内容保存到 Web 服务器上的指定路径；也可以通过 FileBytes 属性获得文件的二进制内容，并将内容保存到字节数组中；也可以通过 FileContent 属性将上传的文件当作流来处理。后两种方法可对文件内容进行直接操作，虽然操作复杂，但可以得到更丰富的功能，如将文件内容直接存储到数据库中。

FileUpload 控件还提供了一个 PostedFile 属性，它的类型是 HttpPostedFile 对象，通过它也可以对上传的文件进行操作，其成员和方法如表 5-3 所示。

表 5-3　HttpPostedFile 对象的成员和方法

成员名称	说　明
ContentLength	上传文件的大小（以字节为单位）
ContentType	上传文件的 MIME 内容类型
FileName	上传文件在客户端的完全限定名称（包含此文件在客户端的文件路径）
InputStream	Stream 对象，它指向上传的文件（与 FileUpload 控件的 FileContent 属性相同）
方法名称	**说　明**
SaveAs	将上传文件的内容保存到 Web 服务器上的指定路径（与 FileUpload 控件的 SaveAs 方法作用相同）

创建一个名为 FileUpload 的网站及其默认主页。为网站创建一个新的文件夹，如 Uploads。

向页面上拖放一个 FileUpload 控件、一个 Button 控件和一个 Label 控件，并为 Button 控件创建单击事件处理函数，相关页面代码如下。

```
<asp:FileUpload ID="FileUpload1" runat="server" />
<br />
<asp:Button ID="Button1" runat="server" Text="上传"
    OnClick="Button1_Click" /><br />
<asp:Label ID="Label1" runat="server"></asp:Label>
```

编写按钮的单击事件处理函数，代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string str="";
```

```
    //如果 FileUpload 控件包含文件
    if(FileUpload1.HasFile)
    {
        try
        {
            //生成完整的文件名:绝对路径+文件名
            string fn=Server.MapPath(Request.ApplicationPath)+
                "\\Uploads\\"+FileUpload1.FileName;
            //保存文件
            FileUpload1.SaveAs(fn);

            //如果保存成功,生成结果信息
            str+="客户端文件:"+FileUpload1.PostedFile.FileName;
            str+="<br/>服务器端保存:"+fn;
            str+="<br/>文件类型:"+FileUpload1.PostedFile.ContentType;
            str+="<br/>文件大小:"+FileUpload1.PostedFile.ContentLength;
        }
        catch(Exception ex)
        {
            //如果文件保存时发生异常,则显示异常信息
            str+="保存文件出错:"+ex.Message;
        }
    }
    else
    {
        //如果不包含文件,给出提示
        str="无上传文件。";
    }
    Label1.Text=str;
}
```

程序中先根据 FileUpload1 控件的 HasFile 属性判断是否有上传文件。如果有上传文件,则生成服务器端保存文件的完整文件名(有关 Server 对象的说明见 6.6 节),再调用控件的 SaveAs 方法保存文件,然后通过 PostedFile 属性来获得文件信息。

执行该页面,界面如图 5-5 所示。

图 5-5　FileUpload 控件的执行效果

页面第一次执行时没有下面的信息。单击“浏览”按钮选择一个本地文件,单击“上

传”按钮，页面重新加载。如果文件上传成功，会显示如图 5-5 的信息。还可以查看网站的 Uploads 子目录，在其中应该能够看到上传的文件。

5.3 Wizard 控件

应用程序经常需要为用户提供向导功能，如本书应用实例的学生注册功能。向导可以引导用户完成多步操作，如收集用户输入等。以前要实现类似的功能非常烦琐，主要是因为不同的页面保持相同的风格及实现在各页面间自由转换并记录状态非常麻烦。

ASP.NET 提供了 Wizard 控件，为用户提供了完成多个步骤操作的实现方法，并可方便地在各步之间前后导航。Wizard 控件提供了一种简单的机制，允许轻松地生成步骤、添加新步骤或重新安排步骤。无须编写代码即可生成线性（从一步转到下一步或上一步）和非线性（从一步转到任意其他步）的导航。该控件能够自动创建合适的按钮，如“下一步”“上一步”和“完成”等，并允许用户自定义控件的用户导航。通过配置可以使某些步骤只能被导航一次。默认情况下，Wizard 控件显示一个包含导航链接的工具栏，让用户可以从当前步骤自由转到其他步。

Wizard 控件有非常灵活的属性设置和事件处理，要详细叙述需要大量篇幅。本节仅给出一个实例，涉及 Wizard 控件使用中的一些重要话题，如果需要实现真正的导航功能，读者还需要参考 VS2010 的联机帮助。

创建一个名为 Wizard 的网站及其默认主页。

拖一个 Wizard 控件到页面上，系统会自动增加如下代码。

```
<asp:Wizard ID="Wizard1" runat="server">
    <WizardSteps>
        <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2">
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>
```

在上述代码中，使用＜asp:Wizard＞标签声明 Wizard 控件，每个 Wizard 控件又可以嵌套地包含多个导航步骤，用＜asp:WizardStep＞标签声明。执行上述页面，可以看到一个有两个步骤的导航程序。这是 Wizard 控件的默认效果，是不够的。但这也是一个很好的基础，Wizard 控件为用户提供了极其强大的扩展能力。

Wizard 控件的每个步骤均会设定一个 StepType 属性，用以指示这一步骤是开始（Start）步骤、中间（Step）步骤、结束（Finish）步骤还是最终完成（Complete）步骤。向导可以根据需要带有任意数量的中间步骤。每个步骤上都可以添加不同的控件（如 TextBox 或 ListBox 等）来支持用户操作。当到达 Complete 步骤时，前面输入的所有数据都可以访问。

虽然与其他控件一样，可以在设计视图中对 Wizard 控件进行可视化设置，但对于 Wizard 控件来说，有些操作还是使用代码的复制比较容易，用户可根据经验，两种方法结合使用。

作为实例，用户可按如下方法操作。

(1) 改变 Wizard 控件到合适的大小。

(2) 将步骤复制为 5 个，为每个步骤设置 ID 和 Title 属性。

(3) 为每个步骤设置要操作的内容，本实例简化为每步只有一个标题和一段说明文字。

(4) 增加一些特殊的导航要求，具体要求如下。

① 将第二步的 AllowReturn 属性设为 False，这样在到达第三步时就会只显示“下一步”按钮，也就不能从第三步再回到第二步了。

② 将第四步的 StepType 属性设为 Finish，这样在到达该步时就会显示“上一步”和“完成”按钮。

③ 将第五步的 StepType 属性设为 Complete，当导航到这一步时，左侧的“导航链接工具栏”和导航按钮都不再显示，但前面各步所产生的数据在此都可访问，用户可以在此步骤最终完成工作，并显示一些表示工作完成的信息。

经过上述操作，页面代码如下。

```
<asp:Wizard ID="Wizard1" runat="server" Height="144px" Width="347px">
    <WizardSteps>
        <asp:WizardStepID="WizardStep1" runat="server" Title="步骤 1">
            <h2>第一步</h2>
            将第一步所要做的工作置于此
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep2" runat="server" Title="步骤 2"
            AllowReturn="False">
            <h2>第二步</h2>
            将第二步所要做的工作置于此
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep3" runat="server" Title="步骤 3">
            <h2>第三步</h2>
            从这一步不能回退到上一步
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep4" runat="server" Title="步骤 4"
            StepType="Finish">
            <h2>第四步</h2>
            将收尾工作置于此
        </asp:WizardStep>
        <asp:WizardStep ID="WizardStep5" runat="server" Title="完成"
            StepType="Complete">
            <h2>完成</h2>
```

```
            工作完成,显示结果信息
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>
```

执行上述页面可以看出,网站已经具有很专业的导航功能了,但外观还不理想。

可以通过手工修改 StepStyle、SideBarStyle 等多个属性来控制外观,也可以简单地套用系统提供的模板方法是进入设计视图,选中 Wizard1,可以看到该控件右上角出现一个小三角形标签,称为“智能标签”。单击智能标签,可以打开一个上下文相关的菜单,在其中选择“自动套用格式”命令,再继续选择一种格式(如“简明型”)。

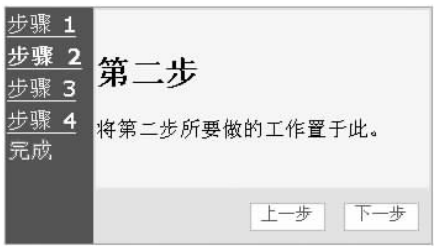

图 5-6　Wizard 控件的执行效果

再执行页面,可以看到外观已经大不相同。例如,其中第二步的执行界面如图 5-6 所示。

5.4　PlaceHolder 控件

直到今天,很多程序员仍然习惯于设计静态布局的页面,这能满足大多数的应用要求。但有些情况下程序员必须动态地控制页面上都包含哪些控件及这些控件的布局,例如,根据程序运行的上下文关系,根据登录用户的权限显示不同的按钮,与当前用户无关的按钮就不再显示,以避免引起混乱。

要达到上述动态效果有多种方法,使用 4.4 节介绍的 Panel 控件就是方法之一。在此再介绍另一个控件——PlaceHolder 控件,也能达到相似的动态效果。

PlaceHolder 控件也是一个容器控件,它可以被放置在页面上,然后在运行时动态地将子元素(子控件)添加到该容器中,已添加的子元素也可以动态地删除。所不同的是,PlaceHolder 控件是一个“空”容器,它只呈现其子元素,而没有自己的基于 HTML 的输出。

PlaceHolder 控件的使用方法是在设计时向页面增加 PlaceHolder 控件,在运行时向 PlaceHolder 控件添加子控件。

向 PlaceHolder 控件添加子控件的方法是创建要添加到 PlaceHolder 控件中的某个子控件的实例,调用 PlaceHolder 控件的 Controls 集合的 Add 方法,将子控件加入到 PlaceHolder 中。

创建一个名为 PlaceHolder 的网站及其默认主页。

从工具箱中拖一个 PlaceHolder 控件到页面上,页面上会增加如下代码:

```
<asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
```