第 6 章 ASP.NET 控件与内置对象

ASP.NET 框架中的控件分为 HTML 服务器控件和 Web 服务器控件，两者都提供了一些能够简化开发工作的特性。ASP.NET 有 7 个内置对象，这些内置对象使用户和程序之间能够更好地交互。本章主要介绍 HTML 服务器控件和 Web 服务器控件以及 ASP.NET 内置对象的使用方法。

6.1 HTML 服务器控件

服务器控件就是在服务器端解析的控件。在 ASP.NET 中，服务器控件就是有“runat＝"server"”标记的控件，这些控件经过处理后会生成客户端呈现代码发送到客户端。服务器控件所在位置示意图如图 6-1 所示。

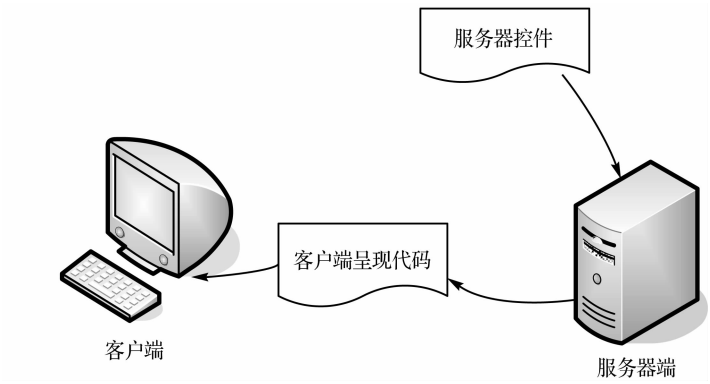

图 6-1 服务器控件所在位置示意图

传统的 HTML 元素是不能被 ASP.NET 服务器端直接使用的，但是将这些 HTML 元

素的功能在服务器端封装之后，开发人员就能很轻松地在服务器端使用这些 HTML 元素。

6.1.1　HTML 服务器控件简介

HTML 服务器控件存在于 System.Web.UI.HtmlControls（简写为 HtmlControl）命名空间中。在该命名空间中，包含 20 多个 HTML 服务器控件类，根据类型可以分为 HTML 输入控件（HtmlInputControl）和 HTML 容器控件（HtmlContainerControl）两个基类。

页面上的任何元素都可以通过添加属性“runat＝"server"”来将其转换为 HTML 服务器控件，如果这些 HTML 元素没有相应的服务器端的类，ASP.NET 将使用 HtmlGenericControl 类（派生自 HtmlContainerControl 类）来表示 HTML 服务器控件。HtmlControl 控件的基本属性如表 6-1 所示。

表 6-1　HtmlControl 控件的基本属性

属　性	说　明
Attributes	允许向 HTML 控件标签添加属性，如可以向一个文本框 TextBox 添加 OnFocus 属性和一些指定的代码，使之能够在获取焦点时执行一些客户端行为
Disabled	获取和设置控件的启用或禁用状态
Style	返回应用到控件的样式的集合
TagName	返回控件的标签名，如 HtmlImage 的 TagName 属性将返回 img 标签值

【例 6-1】 新建一个名为 HtmlAttributesSettingsDemo 的网站，演示 HTML 服务器控件的用法。

Default.aspx 文件的后台代码如下：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        //使用 HtmlControl 类的 Attributes 属性为 Button1 控件添加 onclick 事件
        Button1.Attributes.Add("onclick","alert('HTML 控件属性设置演示')");
        //设置 Button1 的背景色
        Button1.Style.Add("background-color","green");
        //在浏览器中输出 HTML 控件的属性
        Response.Write(Button1.Attributes[("Style")]+"<br/>");
        foreach(string key in Button1.Attributes.Keys)
        {
            Response.Write(key+"="+Button1.Attributes[key]+"<br/>");
        }
    }
}
```

程序运行结果如图 6-2 所示。

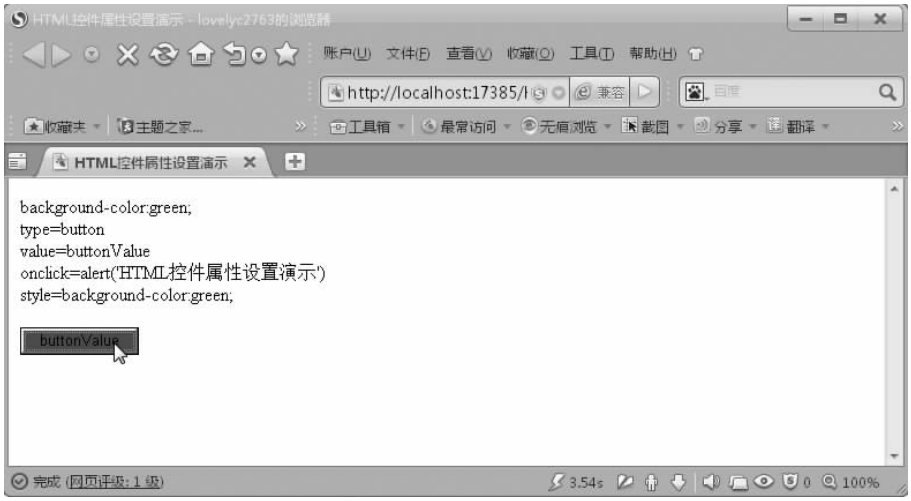

图 6-2　HTML 控件属性设置

6.1.2　HTML 容器控件类和输入类

HtmlContainerControl 类是所有容器控件的基类。HTML 容器控件是指在其控件中可以包含其他 HTML 内容或控件，如 HtmlTable 控件和 Div 控件。HtmlContainerControl 控件常用的属性如表 6-2 所示。

表 6-2　HtmlContainerControl 控件常用的属性

属 性 名 称	说　　明
InnerHtml	获取或设置位于指定的 HTML 服务器控件的开始标记和结束标记之间的内容
InnerText	在 HTML 控件的开闭标签内设置或者返回纯文本，所有的 HTML 语法将被自动地解析为文本

HtmlInputControl 类是所有允许用户交互的控件的基类，如 HtmlInputText 类和 HtmlInputButton 类。它们都生成<input>标签。

例如，HtmlInputText 类生成<input type ="text"/>。

HtmlInputControl 控件常用的属性如表 6-3 所示。

表 6-3　HtmlInputControl 控件常用的属性

属 性 名 称	说　　明
Value	获取或设置与 HtmlInputControl 控件关联的值
Style	获取应用于 ASP.NET 文件中指定的 HTML 服务器控件的所有级联样式表(CSS)属性的集合

6.1.3　HTML 服务器控件类

可供使用的 HTML 服务器控件类及其说明如表 6-4 所示。

表 6-4　HTML 服务器控件类及其说明

类	说　明
HtmlAnchor	允许编程访问服务器上的 HTML＜a＞元素
HtmlButton	允许以编程方式访问服务器上的 HTML＜button＞元素
HtmlContainerControl	用做映射到需要具有开始标记和结束标记的 HTML 元素的 HTML 服务器控件的抽象基类
HtmlControl	定义 ASP.NET 页框架中的所有 HTML 服务器控件所通用的方法、属性(property)和事件
HtmlForm	提供对服务器上的 HTML＜form＞元素的编程访问
HtmlGenericControl	定义不由特定的.NET Framework 类表示的所有 HTML 服务器控件元素的方法、属性和事件
HtmlHead	提供对服务器代码中 HTML＜head＞元素的编程访问
HtmlImage	提供对服务器代码中 HTML＜img＞元素的编程访问
HtmlInputButton	允许以编程方式访问服务器上的 HTML＜input type＝"button"/＞、＜input type＝"submit"/＞和＜input type＝"reset"/＞元素
HtmlInputCheckBox	允许以编程方式访问服务器上的 HTML＜input type＝"checkbox"/＞元素
HtmlInputControl	用做定义所有 HTML 输入控件所共有的方法、属性和事件的抽象基类，如＜input type＝"text"/＞、＜input type＝"submit"/＞和＜input type＝"file"/＞元素
HtmlInputFile	允许编程访问服务器上的 HTML＜input type＝"file"/＞元素
HtmlInputHidden	允许编程访问服务器上的 HTML＜input type＝"hidden"/＞元素
HtmlInputImage	允许编程访问服务器上的 HTML＜input type＝"image"/＞元素
HtmlInputPassword	允许编程访问服务器上的 HTML＜input type＝"password"/＞元素
HtmlInputRadioButton	允许编程访问服务器上的 HTML＜input type＝"radio"/＞元素
HtmlInputReset	允许编程访问服务器上的 HTML＜input type＝"reset/＞"元素
HtmlInputSubmit	允许编程访问服务器上的 HTML＜input type＝"submit"/＞元素

6.1.4　编程创建 HTML 服务器控件

使用 HTML 控件类时，可以使用面向对象的方式来动态创建 HTML 服务器控件，用对象的方式设置控件的属性，最后将控件添加到页面容器的控件集中。

【例 6-2】 创建一个名为 DynamicCreateHtmlControls 的网站，演示如何编程创建HTML服务器控件。

Default.aspx 文件的后台代码为：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        //首先创建一个 HtmlTable 对象
        HtmlTable table1=new HtmlTable();
        //设置 HtmlTable 的格式属性
        table1.Border=2;
        table1.CellPadding=10;
        table1.CellSpacing=10;
        table1.BorderColor="blue";
        Response.Write("<font Color=red  Size=6>表格标题</font>");
        //下面的代码向 HtmlTable 中添加内容
        HtmlTableRow row;
        HtmlTableCell cell;
        for(int i=1;i<= 6;i++)
        {
            //创建一个新的行,并设置其背景色属性
            row=new HtmlTableRow();
            row.BgColor=(i%2==0?"lightyellow":"lightcyan");
            for(int j=1;j<= 8;j++)
            {
                //创建一个新的列,为其设置文本
                cell=new HtmlTableCell();
                cell.InnerHtml="";
                //添加列对象到当前的行
                row.Cells.Add(cell);
            }
            //添加行对象到当前的 Htmltable
            table1.Rows.Add(row);
        }
        //添加 HtmlTable 到当前页的控件集合中
        this.Controls.Add(table1);
    }
}
```

程序运行结果如图 6-3 所示。

图 6-3　动态创建 HTML 控件

6.1.5　处理服务器端事件

尽管已经为 HTML 服务器控件添加了“runat＝"server"”标记，但是在 Visual Studio 2010 的“属性”面板中是看不到有任何事件列表的，而且如果像常规 Web 控件那样双击服务器端控件以生成代码框架，Visual Studio 2010 生成的仍然是客户端的代码框架，如何产生服务器端事件呢？那么先来了解 HTML 服务器控件事件，如表 6-5 所示。

表 6-5　HTML 服务器控件事件

事 件 名 称	提供事件的 HTML 服务器控件
ServerClick	HtmlAnchor、HtmlButton、HtmlInputButton、HtmlInputImage、HtmlInputReset
ServerChange	HtmlInputText、HtmlInputCheckBox、HtmlInputRadioButton、HtmlInputHidden、HtmlSelect、HtmlTextArea

【例 6-3】　创建一个名为 HtmlControlEvent 的网站，演示服务器端事件的处理方法。Default.aspx 文件的后台代码为：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Button1Click(object sender,EventArgs e)
    {
        Response.Write(DateTime.Now.ToString());
    }
    protected void ImageServerClick(Object sender,ImageClickEventArgs e)
    {
```

```
            Label1.Text="鼠标单击位置点在:("+e.X.ToString()+","+
e.Y.ToString()+"). ";
        }
        protected void Button2Click(object sender,EventArgs e)
        {
            Response.Write("第二个按钮");
        }
    }
```

程序运行结果如图 6-4 所示。

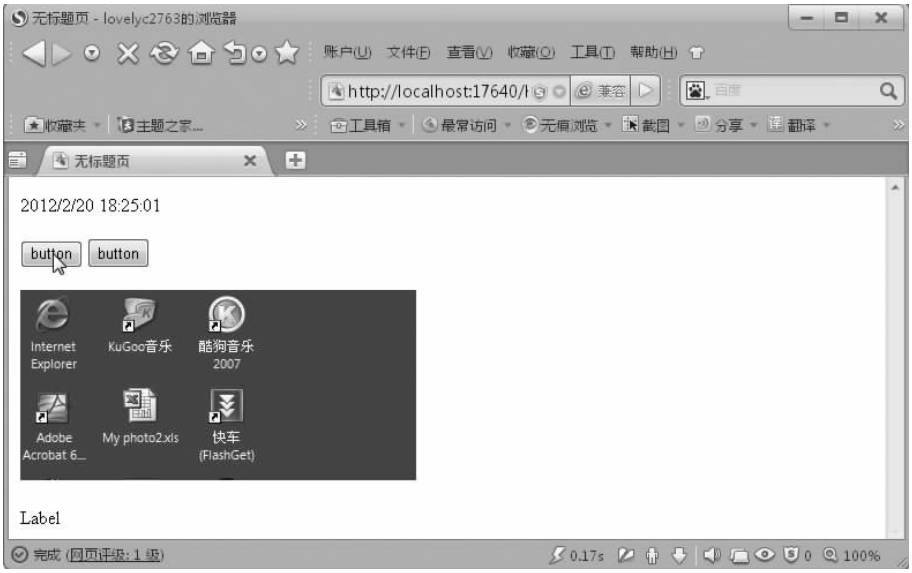

图 6-4　服务器端事件程序运行结果

6.2　常用的 Web 服务器控件

Web 服务器控件是 ASP.NET 应用程序中最常使用的控件,它位于 System.Web.UI.WebControls 命名空间中。所有的 Web 服务器控件都从 WebControl 基类派生。

与 HTML 服务器控件相比,Web 服务器控件提供一个相对抽象的、一致的编程模型。相对抽象是指 Web 服务器控件不必像 HTML 服务器控件那样必须一一对应一个 HTML 标签,事实上很多复杂的 Web 服务器控件所输出的客户端代码也非常复杂。Web 服务器控件也具有一些独有的特性,比如自动回发特性等。

6.2.1　基本 Web 服务器控件介绍

Windows 应用程序的开发中提供了基础的控件集合，如 Labels、Buttons 和 TextBox 等，ASP.NET 同样也提供了这些标准的 Web 服务器控件。基本的 Web 服务器控件复制了每个 HTML 服务器控件的功能，它们都继承自 WebControl 控件并且添加了属于控件自身的属性和事件。基本的 Web 服务器控件如表 6-6 所示。

表 6-6　基本 Web 服务器控件列表

Web 服务器控件	相应的 HTML 标签	说　明
<asp:Button>	<input type="submit"/> 或	按钮控件，在页面上显示一个按钮
<asp:CheckBox>	☐	在页面上显示一个复选框
<asp:FileUpload>	<input type="file"/>	文件上传控件
<asp:HiddenField>	<input type="hidden"/>	隐藏域表单控件
<asp:HyberLink>	<a>…	超级链接
<asp:Image>	<img/>	图片控件，用于显示图片
<asp:ImageButton>	<input type="image"/>	图片按钮控件
<asp:ImageMap>	<map>…</map>	图像映射控件
<asp:Label>	…	显示文本
<asp:LinkButton>	<a>…	显示一个超链接
<asp:Panel>	…	显示一段文本块
<asp:RadioButton>	☐	显示一个 RadioButton 控件
<asp:Table>	<table>…</table>	表格控件
<asp:TableCell>	<td>…</td>	表格列控件
<asp:TextBox>	____	文本框控件

6.2.2　Panel 控件介绍

下面以 Panel 控件为例说明 Web 服务器控件的属性和方法的用法。

【例 6-4】 新建一个名为 ScrollPanel 的网站，演示 Panel 控件的用法。

Default.aspx 文件页面前台代码为：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!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>
        </div>
        <asp:Panel ID="Panel1" runat="server" Height="588px" Width="512px"
BorderStyle="Solid" BorderWidth="1px" ScrollBars="Auto">
        <div>
            <asp:DropDownList ID="DropDownList1" runat="server">
                <asp:ListItem>C#程序设计入门</asp:ListItem>
                <asp:ListItem>C# 2008 编程指南</asp:ListItem>
                <asp:ListItem>LINQ in Action</asp:ListItem>
                <asp:ListItem>WPF 程序设计</asp:ListItem>
            </asp:DropDownList>
            <br/>
            DropDownList 控件<br/>
            <asp:ListBox ID="ListBox1" runat="server">
                <asp:ListItem>C#程序设计入门</asp:ListItem>
                <asp:ListItem>C# 2008 编程指南</asp:ListItem>
                <asp:ListItem>LINQ in Action</asp:ListItem>
                <asp:ListItem>WPF 程序设计</asp:ListItem>
            </asp:ListBox>
            <br/>
            ListBox 控件<br/>
            <asp:CheckBoxList ID="CheckBoxList1" runat="server"
RepeatColumns="2">
                <asp:ListItem>C#程序设计入门</asp:ListItem>
                <asp:ListItem>C# 2008 编程指南</asp:ListItem>
                <asp:ListItem>LINQ in Action</asp:ListItem>
                <asp:ListItem>WPF 程序设计</asp:ListItem>
            </asp:CheckBoxList>
            <br/>
            CheckBoxList 控件<br/>
            <asp:RadioButtonList ID="RadioButtonList1" runat="server">
                <asp:ListItem>C#程序设计入门</asp:ListItem>
                <asp:ListItem>C# 2008 编程指南</asp:ListItem>
                <asp:ListItem>LINQ in Action</asp:ListItem>
```

```
            <asp:ListItem>WPF 程序设计</asp:ListItem>
        </asp:RadioButtonList>
        <br/>
        RadioButtonList 控件<br/>
        <asp:BulletedList ID="BulletedList1" runat="server"
BulletImageUrl="~/1.bmp">
            <asp:ListItem>C# 程序设计入门</asp:ListItem>
            <asp:ListItem>C# 2008 编程指南</asp:ListItem>
            <asp:ListItem>LINQ in Action</asp:ListItem>
            <asp:ListItem>WPF 程序设计</asp:ListItem>
        </asp:BulletedList>
        BulletedList 控件</div>
    </asp:Panel>
    </form>
</body>
</html>
```

程序运行结果如图 6-5 所示。

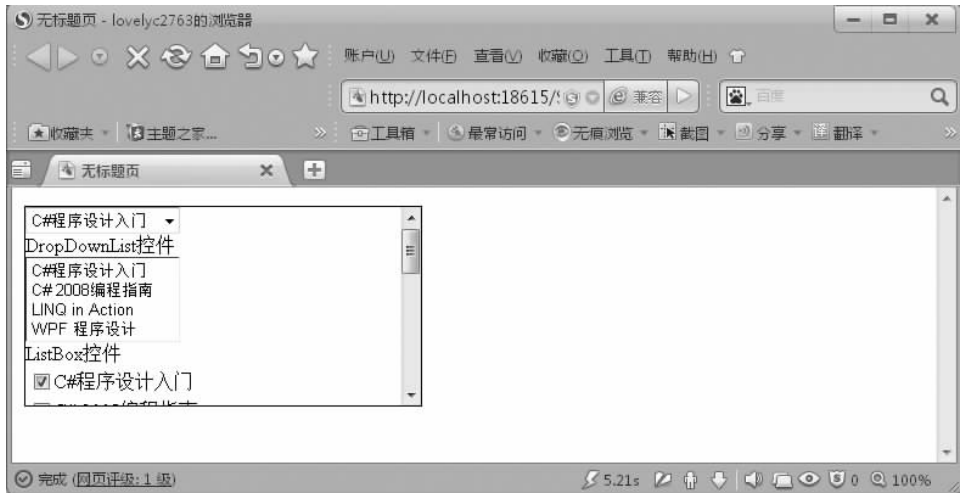

图 6-5　滚动面板示例

6.2.3　列表控件介绍

列表控件是一类特殊的控件，这类控件能够同时显示多行数据，如一个列表框、下拉列表框等。这类控件可以绑定到数据源以显示多行数据，或者通过编程的方式填入数据。多数列表控件允许用户选择一个或多个列表项，但 BulletedList 控件除外，该控件显示一个静态的编号列表。列表控件的名称及说明如表 6-7 所示。

表 6-7　列表控件的名称及说明

控件名称	控件说明
＜asp:DropDownList＞	下拉列表框控件，每个列表项都是一个＜asp:ListItem＞项，在 HTML 中被输出为＜select＞标签
＜asp:ListBox＞	列表框控件，保存一系列＜asp:ListItem＞集合，在 HTML 中被输出为＜select＞标签
＜asp:CheckBoxList＞	复选框列表控件，每一项都是一个复选框，也是一个＜asp:ListItem＞项
＜asp:RadioButtonList＞	单选按钮列表控件，每一项都是一个单选按钮，也是一个＜asp:ListItem＞项
＜asp:BulletedList＞	编号列表控件，每一项都具有一个项目符号或编号，也是一个＜asp:ListItem＞项

1)可选择的列表控件

那些能够让用户选择的列表控件称为可选择列表控件，这些控件具有额外的属性和方法以响应用户的选择事件。

例如，可以将可选择列表控件的 SectionMode 属性设置为 Multiple，这样将允许用户进行多项选择。在可选择的列表控件中，RadioButtonList 和 CheckBoxList 提供了额外的布局选项，使它们能够呈现多列的布局方式。表 6-8 列出了这两个列表控件的布局属性。

表 6-8　RadioButtonList 和 CheckBoxList 的布局属性

属性名称	属性说明
RepeatLayout	用于指定 RadioButtonList 和 CheckBoxList 是输出表结构形式还是非表结构形式，可选值有 Table 和 Flow
RepeatDirection	指定输出时是垂直方向还是水平方向
RepeatColumns	设置列表控件的多列布局，条件是 RepeatLayout 属性要设置为 Table
CellPadding、CellSpacing、TextAlign	如果 RepeatLayout 设置为 Table，这些属性用于配置列的空间和对齐方式

2)BulletedList 列表控件

BulletedList 控件是列表控件中不能被用户选择的控件，它可以使用常用的列表控件的方法和属性。除此之外，BulletedList 控件还提供了如表 6-9 所示的布局属性。

表 6-9　BulletedList 控件的布局属性

属性名称	属性说明
BulletStyle	指定列表的显示类型
BulletImageUrl	如果 BulletStyle 设置为 CustImage，该属性用于指定在列表左侧显示的自定义图像
FirstBulletNumber	指定列表的显示顺序。例如，如果设置 BulletStyle 为 Numbered，FirstBulletNumber 为 5，则显示将使用 5，6，7……
DisplayMode	指定是否将每个列表项显示为文本或者超链接

6.2.4　表格式控件

Web 服务器控件提供了一个类似的 Table 控件，用于呈现一个表格。每个 Table 对象由一个或多个 TableRow 对象组成，每个 TableRow 对象又由一个或多个 TableCell 对象组成。在每个 TableCell 对象中，可以包含其他的 ASP.NET 控件或者 HTML 显示内容。图 6-6 所示为一个典型的 Table 对象的示意图。

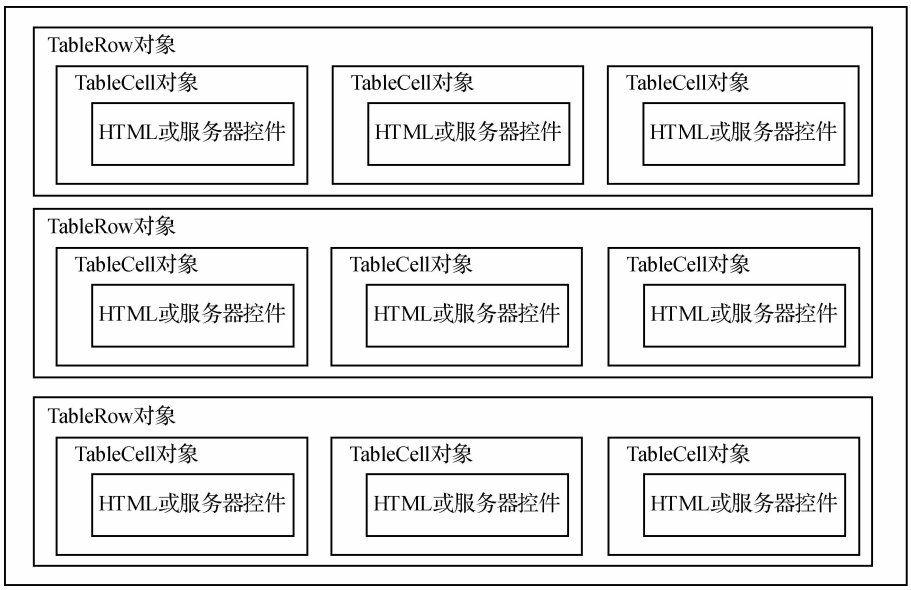

图 6-6　Table 对象示意图

【例 6-5】 新建一个名为 ListControlDemo 的网站展示列表控件。

Default.aspx 文件后台代码为：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        //在这里必须要检查 IsPostBack 标记，如果是回送页面，
        //就不要再多次添加列表项，只在页面第 1 次加载时添加列表项
        if(! Page.IsPostBack)
        {
            for(int i=3;i<=5;i++)
            {
                //每个列表项控件具有一个 Items 属性，这是一个 ListItemCollection
                //类型的集合项，可以调用其 Add 方法来添加列表项
                ListBox1.Items.Add("列表项"+i.ToString());
                DropDownList1.Items.Add("列表项"+i.ToString());
```

```
                    CheckBoxList1.Items.Add("列表项"+i.ToString());
                    RadioButtonList1.Items.Add("列表项"+i.ToString());
                    BulletedList1.Items.Add("列表项"+i.ToString());
                }
            }
        }
        protected void Button1_Click(object sender,EventArgs e)
        {
            Label1.Text="<b>ListBox 控件</b><br/>";
            foreach(ListItem li in ListBox1.Items)
            {
                Label1.Text+="-"+li.Text+"<br/>";
            }
            Label1.Text+="<b>DropDownList 控件</b><br/>";
            foreach(ListItem li in DropDownList1.Items)
            {
                Label1.Text+="-"+li.Text+"<br/>";
            }
            Label1.Text+="<b>RadioButtonList 控件</b><br/>";
            foreach(ListItem li in RadioButtonList1.Items)
            {
                Label1.Text+="-"+li.Text+"<br/>";
            }
            Label1.Text+="<b>CheckBoxList 控件</b><br/>";
            foreach(ListItem li in CheckBoxList1.Items)
            {
                Label1.Text+="-"+li.Text+"<br/>";
            }
            Label1.Text+="<b>BulletedList 控件</b><br/>";
            foreach(ListItem li in BulletedList1.Items)
            {
                Label1.Text+="-"+li.Text+"<br/>";
            }
        }
        protected void Button2_Click(object sender,EventArgs e)
        {
            Label1.Text+="列表中选定项的第一个索引号是:"+
ListBox1.SelectedIndex+"<br/>";
            Label1.Text+="列表中第一个选定项的 Text 是:"+
```

```
ListBox1.SelectedItem.Text＋"<br/>";
            Label1.Text＋＝"列表中第一个选定项的值是:"＋ListBox1.SelectedValue＋
"<br/>";
            foreach(ListItem li in ListBox1.Items)
            {
                //ListItem 的 Selected 属性用于判断是否被选中
                if(li.Selected)
                {
                    Label1.Text＋＝li.Text ＋"<br/>";
                }
            }
        }
        protected void RadioButtonList1_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }
        protected void DropDownList1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            Label1.Text＝"当前选择的是"＋DropDownList1.SelectedItem.Text;
        }
    }
```

程序运行结果如图 6-7 所示。

图 6-7　列表控件示例

6.3 验证控件

为了使开发的应用程序具有较强的健壮性，通常需要对各种输入控件进行验证。如一个只能输入数字的文本框，就不能让用户随意地输入一些文本，或者用户在不知不觉的状态下输入了错误的数据，这些错误的数据被提交到服务器后，会导致 ASP. NET 引发一个异常。

ASP. NET 提供了一套验证控件，可以以声明的方式来完成验证的过程。这些控件大多在客户端完成验证过程，也可以定义服务器的验证方式。

6.3.1 验证控件介绍

ASP. NET 3.5 提供了 5 种类型的验证控件，其中有 4 个控件用于对指定类型的错误进行验证，另外一个控件用于自定义验证。这 5 种验证控件的名称及说明如表 6-10 所示。

表 6-10 5 种验证控件名称及说明

控件名称	控件说明
RequiredFieldValidator	判断输入控件中是否包含一个字符串
RangeValidator	验证输入控件中的值是否在一个指定的范围之内
CompareValidator	验证输入控件中的值是否匹配在其他输入控件中的值，或者由用户指定的一个固定的值
RegularExpressionValidator	验证输入控件中的值是否匹配一个正则表达式
CustomValidator	自定义的验证控件

6.3.2 基类 BaseValidator

验证控件位于 System. Web. UI. Webcontrols 命名空间中，所有的验证控件都派生自 BaseValidator 基类。BaseValidator 基类定义了验证控件的基本功能，如表 6-11 所示。

表 6-11 验证控件基类的成员

成员名称	成员说明
ControlToValidate	指定要验证的输入控件
Display	指定错误消息如何显示，可以设置为 Static 和 Dynamic
EnableClientScript	指定是否允许客户端验证的布尔属性，默认值为 True
Enable	指定允许或者禁用验证控件。如果控件被禁用，将不进行任何验证
ErrorMessage	指定将要显示在 ValidationSummary 控件中的错误消息

续表

成 员 名 称	成 员 说 明
Text	在验证控件旁边显示的文本
IsValid	判断相应的输入控件值是否有效
SetFocusOnError	这是个布尔属性，用于指定当验证失败时是否将输入焦点置于验证失败的错误控件上
ValidationGroup	这是一个非常有用的属性，当页面上有多个验证控件时，可以将验证控件进行逻辑分组，这样将不会发生验证冲突
Validate	这是一个方法，它将重新验证控件并且更新 IsValid 属性。当页面回发时，它将自动被调用。可以在程序代码中调用它来强制进行验证

6.3.3　RequiredFieldValidator 控件

RequiredFieldValidator 控件用于强制用户输入。

在用户注册窗口中，用户名和密码是用户必须输入的，因此可以在这两个 TextBox 控件旁边分别添加一个 RequiredFieldValidator 控件。可以通过 Visual Studio 2010 工具栏的验证栏来拖动 RequiredFieldValidator 控件到页面的设计视图，然后为该控件的 ControlToValidate 属性指定文本框即可。

声明代码如下：

```
用户注册窗口<br/><hr/>
        用户名:   
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1" ErrorMessage="请输入姓名">
        </asp:RequiredFieldValidator>
        <br/>
        密码:       
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server" ControlToValidate="TextBox2" ErrorMessage="请输入密码">
        </asp:RequiredFieldValidator>
        <br/>
        重复密码:<asp:TextBox ID="TextBox3" runat="server">
        </asp:TextBox>
        <br/>
        中文姓名:<asp:TextBox ID="TextBox4" runat="server">
        </asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat=
```

```
"server" ControlToValidate="TextBox4" ErrorMessage="请输入中文姓名">
            </asp:RequiredFieldValidator>
            <br/>
            电子邮箱:<asp:TextBox ID="TextBox5" runat="server">
            </asp:TextBox>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat=
"server" ControlToValidate="TextBox5" ErrorMessage="请输入电子邮箱">
            </asp:RequiredFieldValidator>
            <br/>
```

6.3.4 RangeValidator 控件

RangeValidator 控件用于指定输入范围,即定义某控件中输入的数据的范围段。

假定希望用户的真实年龄为 20～30 岁,那么可以使用一个 RangeValidator 控件将输入的数值限制在指定的范围内。从工具箱的验证栏中拖一个 RangeValidator 控件到真实年龄 TextBox 控件的旁边,将 ControlToValidate 属性设置为该 TextBox 控件,设置 Type 属性为 Integer,MinimumValue 属性为 20,MaximumValue 属性为 30。修改后页面的相关源码如下:

```
<br>
    真实年龄:<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
    <asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="TextBox6" ErrorMessage="请输入 20-30 之间的年龄:)"
MaximumValue="30" MinimumValue="20" Type="Integer">
    </asp:RangeValidator>
<br/>
```

6.3.5 CompareValidator 控件

用户注册窗口要求“重复密码”文本框具有与“密码”文本框相同的密码,这时可以使用 CompareValidator 控件进行比较验证。方法是:在“重复密码”文本框旁边添加一个 CompareValidator 控件,设置 ControlToValidate 属性为“重复密码”文本框,ControlToCompare 属性为“密码”文本框,Type 属性保留默认值 String。如果需要比较某个固定值,可以在 ValueToCompare 属性中输入要比较的值。修改后页面的相关源码如下:

```
<br/>
        密码:       
        <asp:TextBox ID="TextBox2" runat="server" TextMode="Password">
        </asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat=
```

```
"server" ControlToValidate="TextBox2" ErrorMessage="请输入密码">
        </asp:RequiredFieldValidator>
    <br/>
        重复密码:<asp:TextBox ID="TextBox3" runat="server" TextMode=
"Password"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox2" ControlToValidate="TextBox3" ErrorMessage=
"请重复输入密码">
        </asp:CompareValidator>
    <br/>
```

6.3.6　RegularExpressionValidator 控件

正则表达式是查找和替换文本模式的一种简洁而灵活的表示法。在“查找和替换”对话框中执行“快速查找”、“在文件中查找”、“快速替换”或“在文件中替换”操作时，可以在该对话框的“查找内容”和“替换为”字段中使用一组专用的正则表达式。

如图 6-8 所示，要启用正则表达式，应在“查找和替换”对话框中展开“查找选项”选项，勾选“使用”复选框，然后在下面的下拉列表框中选择“正则表达式”选项，此时“查找内容”和“替换为”字段旁的右三角按钮（即“表达式生成器”按钮）变为可用。单击此按钮可以列表显示最常用的正则表达式。当选择列表上的某个正则表达式时，该正则表达式将插入“查找内容”或“替换为”字段中光标所在的位置。如果选中列表中的“完整字符列表”选项，会显示帮助主题。主题的内容涵盖 Visual Studio“查找和替换”功能可以识别的所有正则表达式。此时可以复制主题中的正则表达式，然后将其粘贴到“查找内容”或“替换为”字段中。

图 6-8　正则表达式的使用

6.3.7 CustomValidator 控件

CustomValidator 控件可以让开发人员自定义客户端验证或者服务器端验证，它有一个重要的属性和一个重要的事件。

ClientValidationFunction 属性：指定一个用于完成客户端验证的函数名称。

ServerValidate 事件：在该事件中添加要执行服务器端验证的代码。

下面举例说明 CustomValidator 控件的用法。

新建一个名为 ValidatorDemo 的网站，再在 Default.aspx 文件上布局控件，相应的 CustomValidator 控件的 ServerValidate 事件为：

```
protected void CustomValidator1_ServerValidate(object source,
ServerValidateEventArgs args)
    {
        try
        {
            //必须向 args.IsValid 属性赋验证的结果值
            args.IsValid=args.Value=="许三多";     //算式的优先级妙用
        }
        catch
        {
            args.IsValid=false;
        }
}
```

程序运行结果如图 6-9 所示。

图 6-9 CustomValidator 自定义验证

假如需要让中文姓名使用一个固定值，如等于“许三多”，则可以添加一个 CustomValidator 控件到“中文姓名”文本框旁边，然后指定 ControlToValidate 和ErrorMessage 属性的值。

6.3.8　ValidationSummary 控件

ValidationSummary 控件提供了统一的显示验证错误信息的方式，但它不进行任何的验证工作。ValidationSummary 控件显示每个验证控件的 ErrorMessage 属性的信息。

ValidationSummary 控件可以在页面的某个位置统一显示错误消息，也可以使用一个弹出式的窗口显示错误信息，只需要将 ShowMessageBox 属性设置为 True 即可，也可以同时将 ShowSummary 与 ShowMessage 属性设置为 True，同时显示错误信息。DisplayMode 属性用于设置显示的模式，可选值有 SingleParagraph、List 和 BulletList。HeaderText 属性用于设置错误概要的标题。

6.3.9　ValidationGroup 属性

对于很多复杂的页面来说，在同一页面可能有多个控件组。例如，在某页面上放置了多个 Panel 控件，每个 Panel 控件中又放置了一个 TextBox 控件和一个 Button 控件，如果希望单击某个 Panel 控件中的 Button 控件时，只对属于该 Panel 控件中的 TextBox 控件执行验证，则可以使用验证组(ValidationGroup)属性。

【例 6-6】 新建一个名为 ValidationGroupDemo 的 ASP.NET Web 站点，在 Default.aspx 中放置 3 个 Panel 控件，在每个 Panel 控件中放置一个 TextBox 控件、一个 Button 控件和一个 RequiredFieldValidator 控件，并在底部放两个 CheckBox 控件和一个 ValidationSummary 控件。

Default.aspx 文件后台代码为：

```
///<summary>
///ValidationGroup 的使用要求按钮和验证控件在同一个组里
///</summary>
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {

    }
}
```

程序运行的结果如图 6-10 所示。

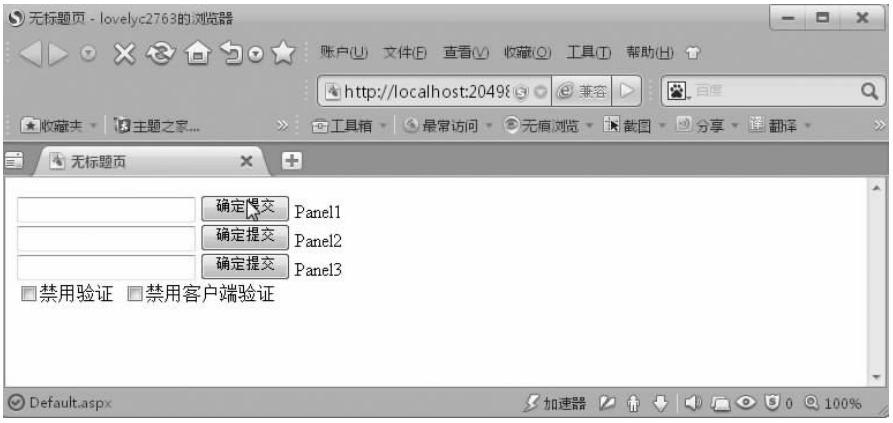

图 6-10　ValidationGroup 控件示例

6.3.10　读取和修改验证控件的属性

与其他 ASP.NET 服务器控件一样，开发人员可以通过编程的方式读取和修改验证控件的属性。为了访问页面上所有的验证控件，可以使用 Page.Validators 集合属性来进行遍历。

下面继续对【例 6-6】进行介绍。假定页面上有一个用于控制验证控件是否有效的开关控件 CheckBox，当勾选该复选框后将禁用所有的验证控件，则可以编码如下：

```
protected void CheckBox1_CheckedChanged(object sender,EventArgs e)
{
    //遍历页面上所有的验证控件
    foreach(BaseValidator validator in Page.Validators)
    {
        //启用或禁用所有的验证控件
        validator.Enabled=CheckBox1.Checked;
        //启用或禁用所有的客户端验证
        validator.EnableClientScript=CheckBox2.Checked;
    }
}
```

6.3.11　Calendar 日期控件

Calendar 控件用于在页面上显示一个日历，它在 ASP.NET 应用程序开发中十分常用。Calendar 控件主要用于完成以下两个方面的功能：

(1)显示和选择日期；

(2)在日历网格中显示约会或其他信息。

Calendar 控件的某种外观格式如图 6-11 所示。

图 6-11　Calendar 控件的某种外观格式

Calendar 控件提供了大量的格式化属性，如标题栏的显示方式和日期的显示方式等。表 6-12 列出了 Calendar 控件提供的样式属性。

表 6-12　Calendar 控件样式属性及说明

属 性 名 称	属 性 说 明
DayHeaderStyle	用于设置 Calendar 的标题栏，也就是星期栏
DayStyle	指定当前月的日期显示样式
NextPreStyle	设置标题栏的导航控件的样式，用于移到上一月或者下一月
OtherMonthDayStyle	在 Calendar 控件中可以同时显示多个月的日期，使用这个属性设置其他月的样式
SelectedDayStyle	设置在 Calendar 控件中选中的日期的样式
SelectorStyle	周或月日期选择控件的样式
TitleStyle	标题选择项的样式
TodayDayStyle	当前日期选择项样式
WeekendDayStyle	周末项的样式

使用 Calendar 控件限制用户所能选择的日期是很容易的，此时需要处理 Calendar.DayRender 事件。该事件在 Calendar 显示到页面上时触发。该事件提供了一个 DayRenderEventArgs 类型的参数，通过该参数的 Day 属性可以获取当前的日期。例如，假定用户不能选择星期天这个日期，示例代码如下：

```
protected void Calendar1_DayRender(object sender,DayRenderEventArgs e)
{
    if(e.Day.IsWeekend)
    {
        //IsSelectable 布尔属性控制日期是否可被选择
        e.Day.IsSelectable=false;
    }
}
```

在 Calendar 控件中，可以利用 SelectedDate 属性以编程的方式来选择日期。例如，可以在 DayRender 事件中添加以下代码来选择当前日期加上 2 之后的日期：

```
Calendar1.SelectedDate=DateTime.Now.AddDays(2).Date;
```

注意:以编程方式设置日期不会引发 SelectionChanged 事件。

【例 6-7】 新建一个名为 CalendarDemo 的网站,演示日期控件的使用方法。

为 Default.aspx 文件添加一个日期控件,代码如下:

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {

    }
    protected void Calendar1_SelectionChanged(object sender,EventArgs e)
    {
        //Label1.Text="当前选择的日期为:"+Calendar1.SelectedDate.
        //ToShortDateString();
        Label1.Text="当前选择的日期为:<br/>";
        foreach(DateTime dt in Calendar1.SelectedDates)
        {
            Label1.Text+=dt.ToShortDateString()+"<br/>";
        }
    }
    protected void Calendar1_PreRender(object sender,EventArgs e)
    {

    }
    protected void Calendar1_DayRender(object sender,DayRenderEventArgs e)
    {
        if(e.Day.IsWeekend)
        {
            //IsSelectable 布尔属性控制日期是否可被选择
            e.Day.IsSelectable=false;
        }
        if(e.Day.IsOtherMonth)
        {
            //e.Cell 属性代表一个日期框
            e.Cell.Text="-";
        }
        //检查日期是否为 10 月 1 日
        if(e.Day.Date.Day==1&&e.Day.Date.Month==10)
        {
```

```
            e.Cell.BackColor＝System.Drawing.Color.Yellow;
            //向该列添加静态文本
            Label lbl＝new Label();
            lbl.Text＝"＜br/＞今天是国庆节!";
            e.Cell.Controls.Add(lbl);        //是可以这样用的
        }
        if(e.Day.Date.Day＝＝4&&e.Day.Date.Month＝＝5)
        {
          e.Cell.BackColor＝System.Drawing.Color.Yellow;
          //向该列添加静态文本
          Label lbl＝new Label();
          lbl.Text＝"＜br/＞今天是五四青年节!";
          e.Cell.Controls.Add(lbl);
        }
        //Calendar1.SelectedDate＝DateTime.Now.AddDays(4).Date;
        //下面的代码将使用 SelectedDates 选中 2010 年 12 月的每个星期天
        SelectedDatesCollection theDates＝Calendar1.SelectedDates;
        theDates.Clear();
        theDates.Add(new DateTime(2010,12,5));
        theDates.Add(new DateTime(2010,12,12));
        theDates.Add(new DateTime(2010,12,19));
        theDates.Add(new DateTime(2010,12,26));

    }
}
```

程序运行结果如图 6-12 所示。

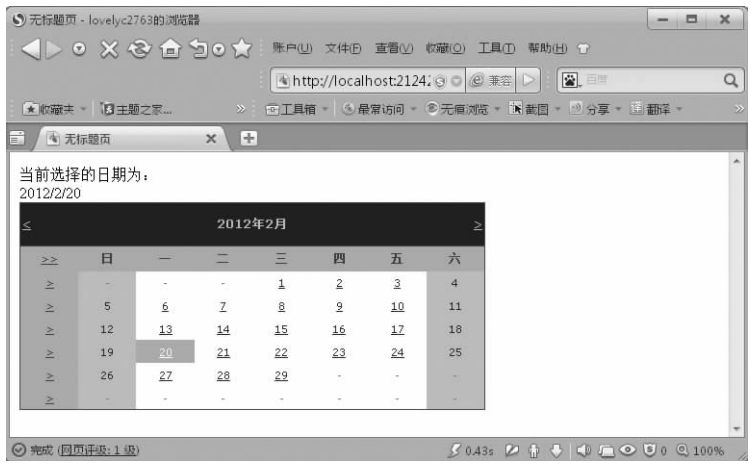

图 6-12　日期控件示例

6.3.12 MultiView 多视图控件

在 Windows 应用程序开发中，一定会用到 PageControl 或者 TabControl 这类控件，该类控件在一个 Form 中可以布局多个视图，用户单击切换按钮就可以切换到不同的视图。可以使用 MultiView 控件来实现类似的效果。

MultiView 控件可以让开发人员定义多个视图，在同一时刻只显示其中一个。可从工具箱的标准栏拖曳一个 MultiView 控件到设计视图。

【例 6-8】 新建一个名为 MultiViewDemo 的网站，演示多视图控件的使用。拖动一个 MultiView 控件到 Default.aspx 页面上，再在该控件上放置 3 个 View 视图控件。

Default.aspx 前台主要代码如下：

```
<body>
      <form id="form1" runat="server">
      <div>
          <asp:MultiView ID="MultiView1" runat="server"
ActiveViewIndex="0">
          <asp:View ID="View1" runat="server">
                <asp:Label ID="Label1" runat="server" Text="这是视图一">
                </asp:Label>
                <asp:Button ID="btnNext1" runat="server" Text="下一页"
CommandName="NextView" onclick="btnNext1_Click"/>
                </asp:View>
          <asp:View ID="View2" runat="server">
                <asp:Label ID="Label2" runat="server" Text="这是视图二">
                </asp:Label>
                <asp:Button ID="btnPrevious2" runat="server" Text="上一
页" CommandName="PrevView"/>
                <asp:Button ID="btnNext2" runat="server" Text="下一页"
CommandName="NextView" onclick="btnNext2_Click"/>
          </asp:View>
          <asp:View ID="View3" runat="server">
              <asp:Label ID="Label3" runat="server" Text="这是视图三">
              </asp:Label>
              <asp:Button ID="btnPrevious3" runat="server" Text="上一页"
CommandName="PrevView"/>
           </asp:View>
          </asp:MultiView>
      </div>
    </form>
</body>
```

程序运行结果如图 6-13 所示。

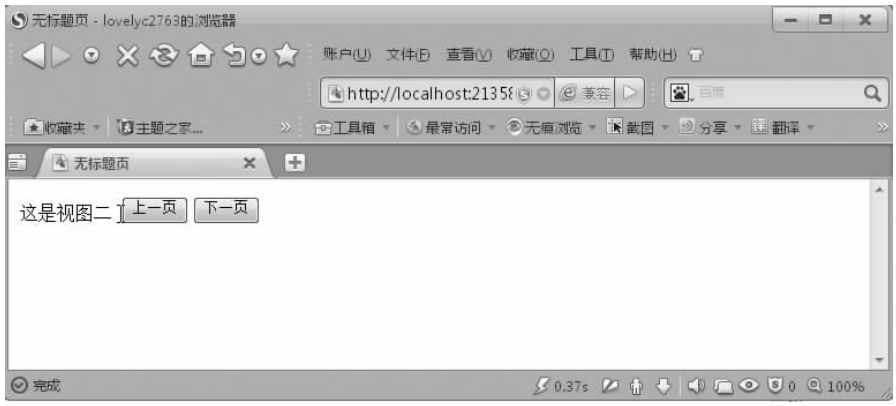

图 6-13　多视图控件示例

6.4　ASP.NET 的内置对象

ASP.NET 的内置对象主要包括：Page 对象、Response 对象、Request 对象、Application 对象、Server 对象、Cookie 对象、Session 对象。

6.4.1　Page 对象

Page 对象其实就是 C# 中 Web 应用程序的 aspx 文件，它又称为窗体面。

Page 对象有许多常用的属性（如 IsPostBack、IsValid 等），以及几个很重要的事件（如 Init、Load、Unload、Error 等）。这几个属性具体说明如下：

（1）Page_Init：用于初始化所有值或程序。

（2）Page_Load：页面加载事件。

（3）Page_Unload：完成页面呈现或者加载之后，将激发 Page_Unload 事件。

（4）Page_Error：如果在页面处理过程中出现一些错误，就会激发 Error 事件。并且这些错误事件提供了处理的方法。

其中，Page_Init 事件和 Page_Load 事件的区别是：Page_Init 事件是完成初始化工作，而 Page_Load 事件是在初始化的基础上加载内容。例如，当用户在浏览页面时触发了某个事件，客户端就会将窗口数据传送到服务器，服务器需要重新加载一次页面，然后再将数据返回到客户端，于是客户端也再次加载，但是这一次加载不再进行初始化，而是直接运行 Page_Load 事件。

6.4.2 Response 对象

Response 对象是 Web 开发中常用的对象之一，用于动态响应客户端的请求，并将响应信息返回到客户端浏览器中。

Response 对象有两个重要的方法，分别是 Write()和 Redirect()。

(1)Write()方法用于将数据输出到客户端浏览器，其语法格式如下：

```
Response.Write("字符串/变量")
```

该方法通常在程序中可简写，例如：

```
Response.Write("<script>alert('输入不能为空值！')</script>");
```

(2)Redirect()方法用于重新将网页转向另一个网址，其语法格式如下：

```
Response.Redirect("URL")
```

其中，URL 就是要重定向到的页面网址，例如：

```
protected void Button1_Click(object sender,EventArgs e)
{
    Response.Redirect("http://www.qq.com");        //转到新页
}
```

【例 6-9】 新建一个名为 EX_6_4 的网站，演示 Response 对象在 Web 开发中的重定向功能。在 Default.aspx 页面添加一个按钮，并添加单击事件代码。

```
protected void Button1_Click(object sender,EventArgs e)
{
    Response.Redirect("http://www.qq.com");
   //Response.Redirect("Default2.aspx");
}
```

程序运行结果如图 6-14 所示。

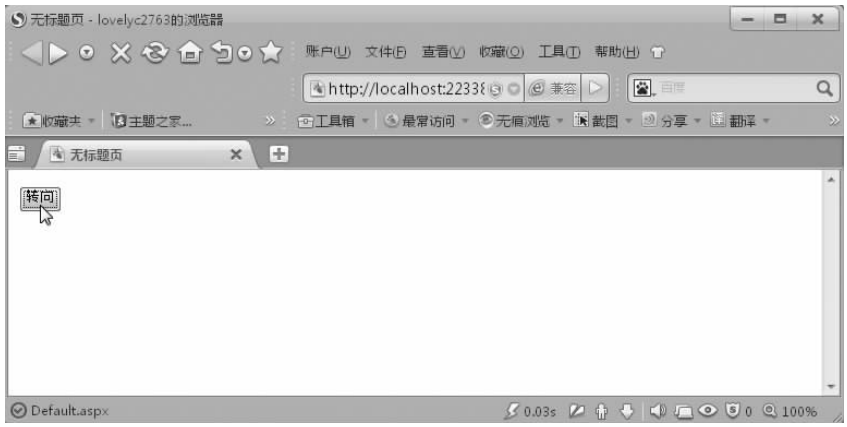

图 6-14 Response 对象重定向功能示例

6.4.3　Request 对象

Request 对象称为请求对象，其功能是从浏览器中得到数据。它有两种取得数据的方法，分别是 Request.Form()和 Request.QueryString()。

【例 6-10】　新建一个网站，演示 Request 对象的作用。该网站的功能是：将在 Default.aspx页面中发送的留言在 Default2.aspx 页面中显示出来。

Default.aspx 文件的后台代码如下：

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As System.Object,ByVal e As System.
EventArgs)Handles MyBase.Load
        Button1.Text="发送留言"
    End Sub
    //单击"发送留言"按钮后执行下面的程序
     Protected Sub Button1_Click(ByVal sender As System.Object,ByVal e As
System.EventArgs)Handles Button1.Click
        Dim tmp,tmp1 As String
        tmp="~/Default2.aspx? t1="
        tmp+=Request.Form("Textbox1") & "&t2=" & Request.Form("Text-
box2")
        'tmp1=Request.Form("Textbox1")
        'Response.Redirect(tmp1)
        Response.Redirect(tmp)
    End Sub
End Class
```

Default2.aspx 文件的代码如下：

```
Partial Class Default2
    Inherits System.Web.UI.Page
     Protected Sub Page_Load(ByVal sender As Object,ByVal e As System.
EventArgs) Handles Me.Load
        Label1.Text=Request.QueryString("t1") & "  留言人:" & Request.
QueryString("t2")
    End Sub
End Class
```

程序运行结果如图 6-15 和图 6-16 所示。

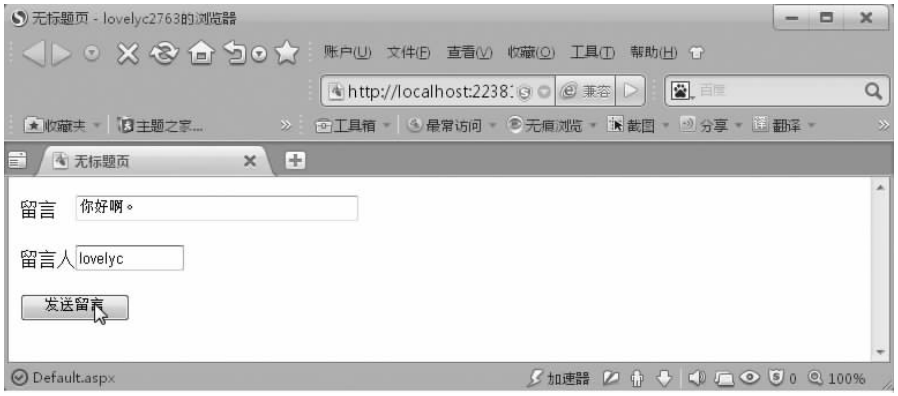

图 6-15　留言的页面

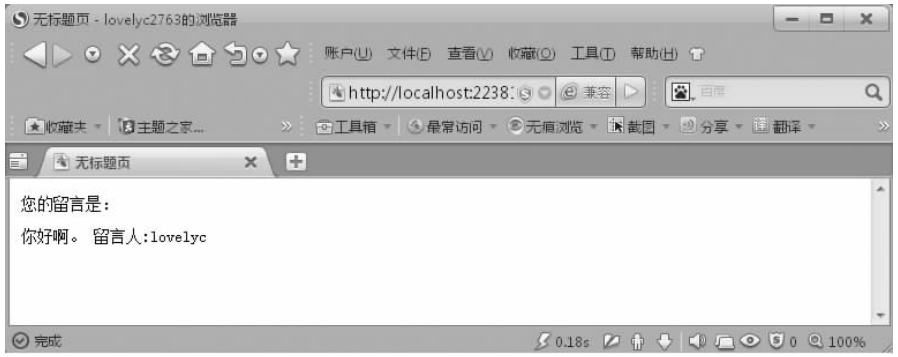

图 6-16　接收留言显示的页面

客户端的基本信息，如浏览器类型、浏览器版本号、用户所用的语言以及编码方式等都封装在 Request 对象中，可以使用 Request 对象来读取这些信息。例如：

客户端浏览器版本信息：<%=Request. UserAgent %>。

客户端 IP 地址：<%=Request. UserHostAddress %>。

客户端机器的 DNS 名称：<%=Request. UserHostName %>。

当前文件服务器端物理路径：<%=Request. PhysicalApplicationPath %>。

在 Request 对象中，QueryString()用得最为广泛，常用其返回 URL 参数中的结果。例如，对于 URL 参数“demo. aspx? ID=11”，可以用 Request. QueryString[ID]来获得它后面的数字 11。这在数据库编程中经常用到。

如果要从浏览器的地址栏中获取数据，也可以利用 Request 对象读取。它可以读取其他页面提交过来的数据。这些数据提交的方式有两种：一种是通过 Form 表单提交的(POST 方法)，另一种是通过超级链接后面的参数提交的，即 URL 方式(GET 方法)。

【例 6-11】 新建一个网站，用于获取浏览器信息并显示。

Default. aspx 文件的代码如下：

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object,ByVal e As
```

```
System.EventArgs) Handles Me.Load
        Response.Write("您的浏览器相关信息如下<br>")
        Response.Write("浏览器名称:" & Request.Browser.Browser & "<br>")
        Response.Write("浏览器类型:" & Request.Browser.Type & "<br>")
        Response.Write("浏览器版本:" & Request.Browser.Version & "<br>")
        Response.Write("操作系统平台:" & Request.Browser.Platform & "<br>")
        Response.Write("浏览器是否为测试版:" & Request.Browser.Beta & "<br>")
        Response.Write("是否为 16 位环境:" & Request.Browser.Win16 & "<br>")
        Response.Write("是否为 32 位环境:" & Request.Browser.Win32 & "<br>")
        Response.Write("浏览器是否支持框架:" & Request.Browser.Frames & "<br>")
        Response.Write("浏览器是否支持表格:" & Request.Browser.Tables & "<br>")
        Response.Write("浏览器是否支持 Cookie:" & Request.Browser.Cookies & "<br>")
        Response.Write("浏览器是否支持 VBScript:" & Request.Browser.
VBScript & "<br>")
        Response.Write("浏览器是否支持 JaveScript:" &Request.Browser.
JavaScript & "<br>")
        Response.Write("浏览器是否支持 ActiveXControl:" & Request.Browser.
ActiveXControls & "<br>")
    End Sub
End Class
```

程序运行结果如图 6-17 所示。

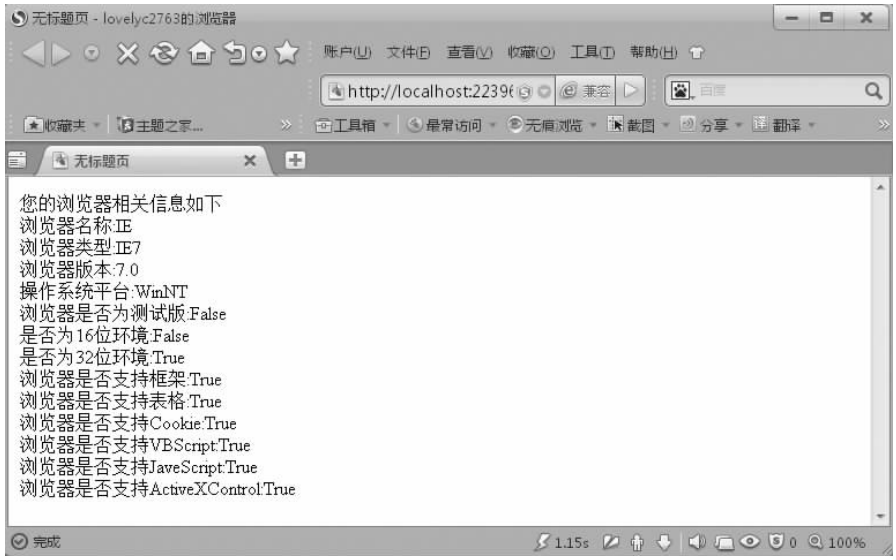

图 6-17　用 Request 获取浏览器信息

6.4.4 Application 对象

Application 对象是 HttpApplicationState 类的一个实例，它可以生成一个所有 Web 应用程序都可以存取的变量。这个变量的使用范围涵盖全部使用者，只要是正在使用这个网页的程序都可以存取这个变量。

应用程序状态的使用方式与 Session 基本一致，都支持相同的对象类型，信息都保存在服务器上，并且使用一致的语法。使用应用程序变量的一个常见的例子是网页计数器。

【例 6-12】 新建一个网站，添加 Global.asax 文件，并利用 Application 对象存储网站来访者数量。

文件代码如下：

```
<%@ Application Language="C#" %>
<script runat="server">
    void Application_Start(object sender,EventArgs e)
    {
        //在应用程序启动时运行的代码
        Application["counter"]=0;
    }
    void Application_End(object sender,EventArgs e)
    {
        // 在应用程序关闭时运行的代码
    }
    void Application_Error(object sender,EventArgs e)
    {
        //在出现未处理的错误时运行的代码
    }
    void Session_Start(object sender,EventArgs e)
    {
        //在新会话启动时运行的代码
        Application.Lock();
        Application["counter"]=(int)Application["counter"]+1;
        Application.UnLock();
    }
    void Session_End(object sender,EventArgs e)
    {
       //在会话结束时运行的代码
       //注意:只有在 Web.config 文件中的 sessionstate 模式设置为 InProc 时,才
       //会引发 Session_End 事件,如果会话模式设置为 StateServer 或 SQLServer,
       //则不会引发该事件
```

```
            Application.Lock();
            Application["counter"]=(int)Application["counter"]-1;
            Application.UnLock();
        }
</script>
```

Default.aspx 文件的代码如下：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        Label1.Text=Application["counter"].ToString().Trim();
        Response.Write("浏览器名称:"+Request.Browser.Browser+"<br>");
        Response.Write("当前在线人数:"+Label1.Text);
    }
}
```

程序运行结果如图 6-18 所示。

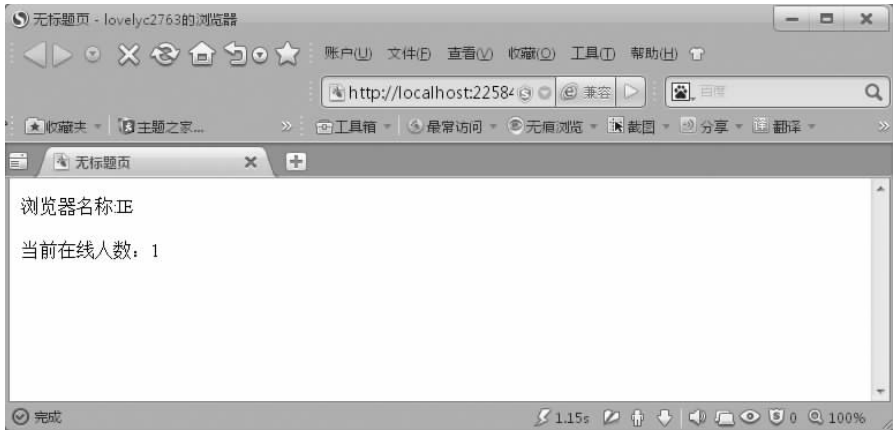

图 6-18　使用 Application 对象统计在线人数

【例 6-13】 新建一个网站，演示 Application 对象的读取。

Default.aspx 文件代码如下：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        Application["app1"]="app1";  //添加 Application 对象 app1
        Application["app2"]="app2";  //添加 Application 对象 app2
        Application["app3"]="app3";  //添加 Application 对象 app3
        Application["app4"]="app4";  //添加 Application 对象 app4
```

```
            string[] App=new string[Application.Count];   //计算对象的数量
            App=Application.AllKeys;                      //添加到数组中
            for(int i=0;i<App.Length;i++)                 //for 循环
            {
                Response.Write(App[i].ToString());        //输出对象
                Response.Write("<br>");                   //回车换行
            }
        }
}
```

程序运行结果如图 6-19 所示。

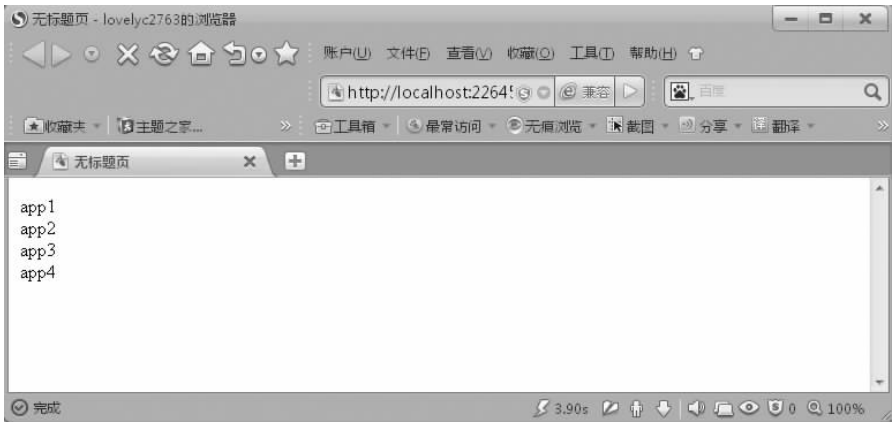

图 6-19　Application 对象的读取

6.4.5　Server 对象

Server 对象提供了访问远程服务器信息的方法和属性，其中大多数方法和属性是为实用程序的功能服务的，如获取计算机名称、服务器的语言、服务器的 IP 地址、文件在服务器上的物理路径等。

```
Server.MachineName                  //获取服务器名称
Server.MapPath("Default.aspx")      //获取文件物理路径
```

【例 6-14】 新建一个网站，使用 Server 对象提供的方法和属性访问远程服务器信息。Default.aspx 文件的代码如下：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e) //页面加载事件
    {
        //查看文件 Default.aspx 的物理路径
        Response.Write(Server.MapPath("Default.aspx"));
        Response.Write("</p>");              //输出换行符
```

```
            //不被 HTML 解析
            Response.Write(Server.HtmlEncode("<B>08 软件工程.NET 课程</B>"));
            Response.Write("</p>");
            //被 HTML 解析了
            Response.Write(Server.HtmlDecode("<B>08 软件工程.NET 课程</B>"));
            Response.Write("</p>");
            Response.Write("计算机名称:");              //输出字符串
            Response.Write(Server.MachineName);         //输出计算机名称
        }
}
```

程序运行结果如图 6-20 所示。

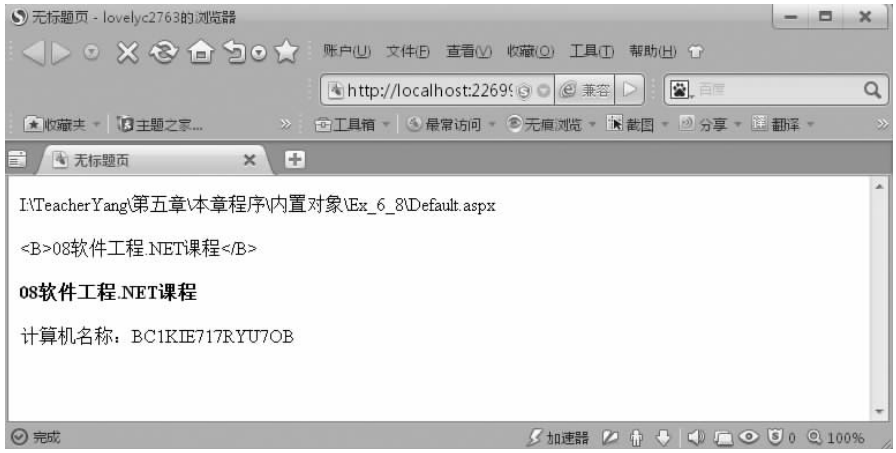

图 6-20　使用 Server 对象访问远程服务器信息

6.4.6　Cookie 对象

Cookie 提供了一种在 Web 应用程序中存储用户特定信息(如历史记录或用户首选项)的方法。Cookie 是一小段文本信息,随着请求和响应在 Web 服务器和客户端之间传递。Cookie 包含用户每次访问站点时 Web 应用程序都可以读取的信息,并将用户信息存储在客户端硬盘上的 Internet 临时文件夹中。

只要用户没有清除浏览器端的 Cookie 文件,当再次请求站点中的页面时,浏览器便会在本地硬盘上查找与该 URL 关联的 Cookie。

Cookie 文件中保存的文本信息(不超过 4 096 字节)很容易被截取而造成资源的外泄,因此在开发应用程序时,对于一些机密性强的信息,如银行卡账号等,不要使用 Cookie 技术来保存。

ASP.NET 中可以使用 Response 对象向浏览器发送 Cookie,使用 Request 对象获取 Cookie,Cookie 信息以名/值对的形式保存。每个 Cookie 必须具有一个唯一的名称,以便于浏览器进行识别,如果浏览器检测到具有相同名称的 Cookie,则会覆盖其中的

一个。

【例 6-15】 新建一个名为 UserCookies 的网站，演示 Cookie 对象的读取。

Default.aspx 文件的代码如下：

```
public partial class _Default:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
    }
    protected void Button2_Click(object sender,EventArgs e)
    {
        //创建一个 Cookie 对象
        HttpCookie cookie=new HttpCookie("BooksInfo");
        //为 Cookie 赋值
        cookie["BookName"]=TextBox1.Text;
        cookie["ISBN"]=TextBox2.Text;
        cookie["Price"]=TextBox3.Text;
        //将 Cookie 对象添加到 Cookies 集合中并发送到浏览器
        Response.Cookies.Add(cookie);
        //设置 Cookie 对象的过期时间
        cookie.Expires=DateTime.Now.AddDays(1);
    }
    protected void Button1_Click(object sender,EventArgs e)
    {
        //从 Request.Cookies 集合中获取指定名称的 Cookie 对象
        HttpCookie cookie=Request.Cookies["BooksInfo"];
        //获取 Cookie 中指定键的值
        Label1.Text="Cookie 中的书籍信息为:<br/>";
        Label1.Text+=cookie["BookName"]+"<br/>";
        Label1.Text+=cookie["ISBN"]+"<br/>";
        Label1.Text+=cookie["Price"]+"<br/>";
    }
}
```

程序运行结果如图 6-21 和图 6-22 所示。

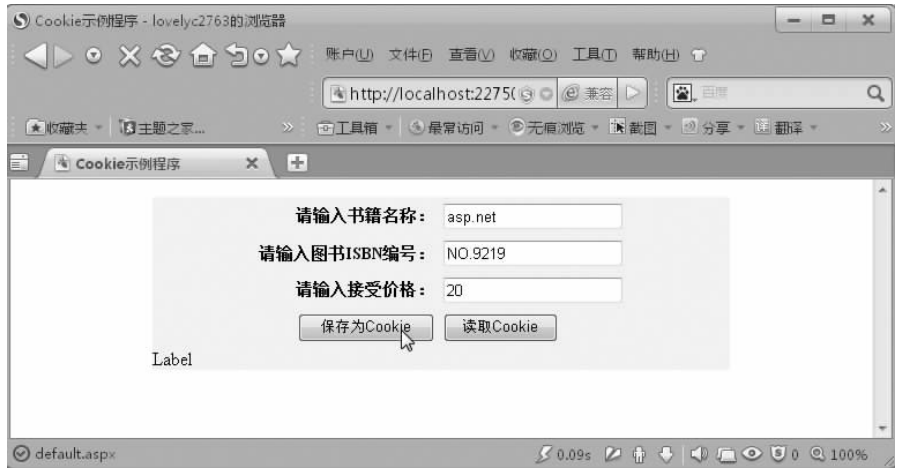

图 6-21　保存 Cookie

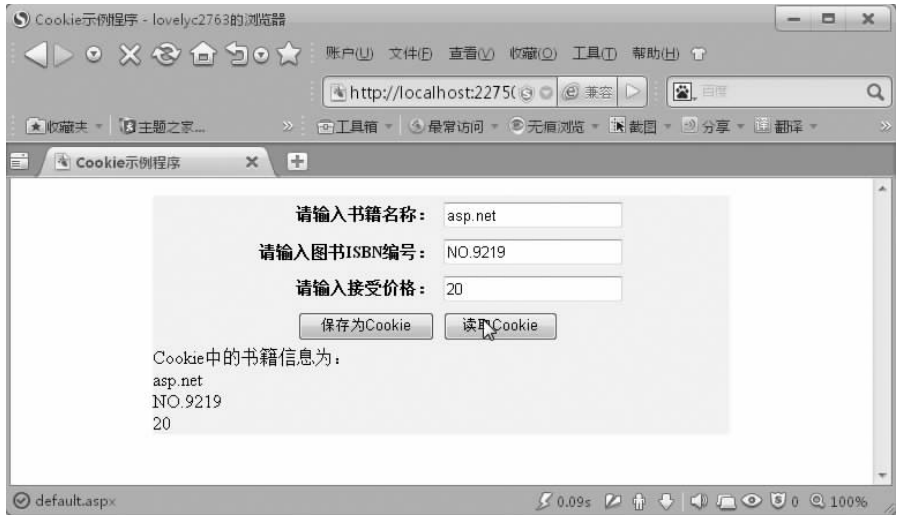

图 6-22　读取 Cookie 结果

6.4.7　Session 对象

当一个用户开始访问 Web 应用程序时，就会产生一个会话状态。不同的用户具有不同的会话状态，即如果有 1 万个用户，将会有 1 万个会话状态。

会话状态在存储与用户相关的信息方面非常有用，如购物网站的购物车就使用会话状态来存储。但使用会话状态并不是无偿的，会话状态需要维护每个用户的信息，这将消耗大量的服务器端资源。

由于必须为每个用户维护一个会话状态，所以 ASP.NET 将会为每个新用户创建一个唯一的会话 ID。这个会话 ID 是一个 120 位的标识符，ASP.NET 使用一种保密算法生成这个值，以保证该值的唯一性。当客户端持有一个会话 ID 时，ASP.NET 将搜寻相应的会话，

提取用户在会话中存储的对象，放入一个指定的集合中让用户访问。

会话状态保存在服务器端，开发人员可以指定要存储会话的类型，如可以指定存储到服务器端内存、SQL Server、专门的状态服务器等，还可以在 Web.config 配置文件中配置会话状态。

访问或者存取会话状态可以使用 Page.Session 属性。例如，下面的代码使用 Session 属性存储一个 DataSet 对象，并从 Session 中获取 DataSet 对象：

```
Session["myDataSet"]=DataSetObject;
DataSetObject=(DataSet)Session["myDataSet"];
```

会话状态对当前用户来说是全局性的，不论用户访问哪个页面，都可以获取存储在会话状态中的信息，但下面的做法将会导致会话状态丢失：

(1)用户关闭并重新启动了浏览器；

(2)用户使用不同的浏览器访问了相同的页面。由于不同的浏览器处理会话的方式不同，因此也会导致会话状态的丢失。

【例 6-16】 新建一个网站，展示 ASP.NET 内置对象的综合应用。该网站的功能是获取服务器系统信息并显示。

Default.aspx 文件的代码如下：

```
//System.Runtime.InteropServices 提供了相应的类或者方法来支持
//托管/非托管模块间的互相调用
using System.Runtime.InteropServices;
public partial class _Default:System.Web.UI.Page
{
    public SqlConnection sqlcon=new
    SqlConnection(ConfigurationManager.AppSettings["ConnString"]);
    protected void Page_Load(object sender,EventArgs e)
    {
        if(!IsPostBack)
        {
            Response.Cookies.Add(new HttpCookie("CheckCode",""));
            if(Request.Cookies["CheckCode"]==null)
            {
                chcoo.Text="您的浏览器设置已被禁用 Cookies,您必须设置浏览器允许使用 Cookies 选项后才能使用本程序。";
                chcoo.Visible=true;
                return;
            }
        }
        chcoo.Text="可写";
        String ServerStart=((Double)System.Environment.TickCount / 3600000).
```

```
ToString("N2");
                Label1.Text=ServerStart;
                String serverOS=Environment.OSVersion.ToString();
                OS.Text=serverOS;
                String CpuSum=Environment.GetEnvironmentVariable("NUMBER_OF_
PROCESSORS");
                //CPU 个数
                cpunum.Text=CpuSum;
                 String CpuType=Environment.GetEnvironmentVariable("PROCESSOR_
IDENTIFIER");
                //CPU 类型
                cputype.Text=CpuType;
                String ServerCache=Cache.Count.ToString();
                 //应用程序缓存总数,应用程序占用内存
                Needcatch.Text=ServerCache;
                //服务器名
                String MachineName=Server.MachineName;
                sname.Text=MachineName;
                //服务器域名
                String ServerName=Request.ServerVariables["SERVER_NAME"];
                Surl.Text=ServerName;
                //DotNET 版本
                String ServerNet=".NET CLR "+Environment.Version.ToString();
                NetV.Text=ServerNet;
                //获得主机名
                string hostName=Dns.GetHostName();
                //获得 IP 列表
                IPAddress[] addressList=Dns.GetHostByName(hostName).AddressList;
                sip.Text=addressList[0].ToString();
                //脚本超时时间
                tout.Text=Server.ScriptTimeout.ToString();
                HttpBrowserCapabilities bc=new HttpBrowserCapabilities();
                bc=Request.Browser;
                cos.Text=bc.Platform.ToString();
                iev.Text=bc.Type.ToString();
                cip.Text=getIp();
                spath.Text=Request.ServerVariables["APPL_PHYSICAL_PATH"];
                scount.Text=Session.Contents.Count.ToString();
```

```
        }
        private string getIp()
        {/* 穿过代理服务器获取远程用户真实的 IP 地址:REMOTE_ADDR 获取的是 ipHead
中的数据,HTTP_X_FORWARDED_FOR 获取的是 httpHead 中的数据 */
            if(Request.ServerVariables["HTTP_VIA"]! =null)
                return Request.ServerVariables["HTTP_X_FORWARDED_FOR"].
ToString();
            else
                return Request.ServerVariables["REMOTE_ADDR"].ToString();
        }
    }
```

程序运行结果如图 6-23 所示。

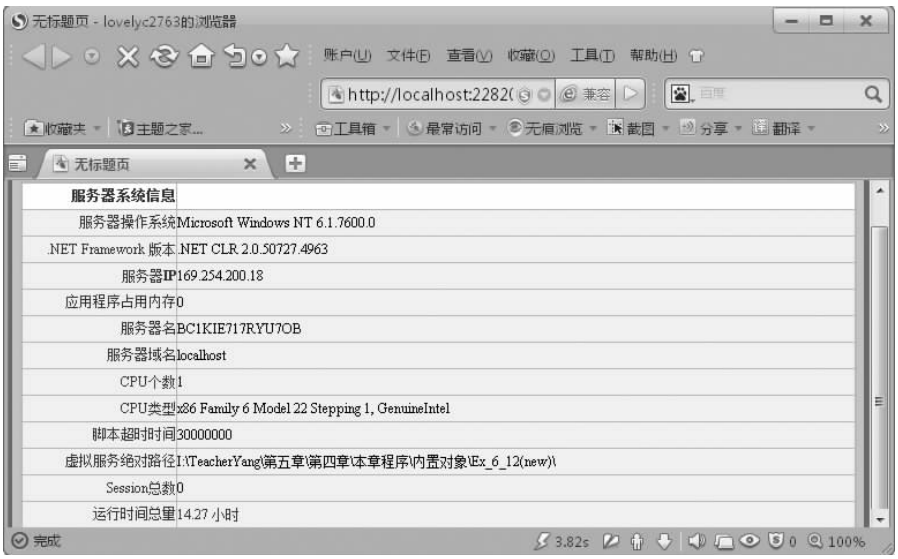

图 6-23　ASP. NET 内置对象的综合应用

习　题　6

(1)HTML 服务器控件在哪个命名空间中？它分为哪两个基类？

(2)如何在 HTML 控件中添加服务器端运行事件？

(3)如何动态创建一个 HtmlTable 对象实例？

(4)基本的 Web 服务器控件有哪些？

(5)常用的列表控件有哪些？它们又可以分为哪两类？

(6)ASP. NET 3.5 的 5 种验证控件分别是什么？如何使用？

(7)如何使用 ValidationGroup 属性？

(8)Calendar 日期控件的作用是什么？它提供了哪些样式属性？

(9)MultiView 控件的作用是什么？

(10)ASP. NET 内置对象主要包括哪些？

(11)什么是 Page 对象？它有哪些主要的事件？

(12)Request. Form()和 Request. QueryString()分别如何使用？

(13)Cookie 对象有何作用？

(14)如何通过内置对象获取服务器名、主机名和主机 IP 地址？