第二篇 WEB 页面

第一章　WEB 页面简介

2.1.1 WEB FORM

表单，英文单词是 Form，学习过 VB 的朋友一定不会陌生。在 MS.NET 架构里，Form 是一个经常使用到的词汇。比如：编写 Windows 应用时会提到 Windows Form，编写 Web 应用时会提到 Web Form。Windows Form 可以看作一个 Windows 窗体，这和在 VB 里面一样。而 Web Form 则代表了一个一个的 Web 页面。总的看来，Form 就像是一个容纳各种控件的容器，各种控件都必须直接或者间接的和它有依存关系。Form 在这里译作“ WEB 表单 ”似乎有些不妥。“ 表单 ”这个词，在 WEB 程序员看来，总是和 HTML 里面的“ Form ”相混淆。“ WEB 表单 ”似乎翻译成“ WEB 页面 ”更加妥当一些。

大家还记得 VB 里面的 Form 实际上就是一个对象吧，它可以有自己的属性、方法、事件等等。WEB 表单，或者说 WEB 页面，实际上是一个“ 对象 ” (Object)。MS.NET 架构里面一个比较重要的概念就是“ 对象 ”：所有的控件都是对象，甚至数据类型都成了对象；每种数据类型都有自己特有的属性和方法。我们在后面的编程中将可以体会到。

WEB FORM 的后缀名是 ASPX。当一个浏览器第一次请求一个 ASPX 文件时，WEB FORM 页面将被 CLR（common language runtime）编译器编译。此后，当再有用户访问此页面的时候，由于 ASPX 页面已经被编译过，所以，CLR 会直接执行编译过的代码。这和 ASP 的情况完全不同。ASP 只支持 VBScript 和 JavaScript 这样的解释性的脚本语言。所以 ASP 页面是解释执行的。当用户发出请求后，无论是第一次，还是第一千次，ASP 的页面都将被动态解释执行。而 asp.net 支持可编译的语言，包括 VB.NET、C#、Jscript.NET 等。所以，asp.net 是一次编译多次执行。

为了简化程序员的工作，ASPX 页面不需要手工编译，而是在页面被调用的时候，由 CLR 自行决定是否编译。一般来说，下面两种情况下，ASPX 会被重新编译：

1．ASPX 页面第一次被浏览器请求；

２．ASPX 被改写

由于 ASPX 页面可以被编译，所以 ASPX 页面具有组件一样的性能。这就使得 ASPX 页面至少比同样功能的 ASP 页面快 250%！

下面我们来看一下简单的 WEB 页面。

2.1.2 我的第一个 Page

把下面的代码拷贝到 myfirstpage.aspx 文件中，然后从浏览器访问这个文件：

```
<!--源文件：form\web 页面简介\myfirstpage.aspx-->
<form action="myfirstpage.aspx" method="post">
```

```
<h3>  姓名: <input id="name" type=text>

所在城市:   <select id="city" size=1>
                <option>北京</option>
                <option>上海</option>
                <option>重庆</option>
            </select>

<input type=submit value="查询">

</form>
```

你可能觉得这个页面太简单了，用 HTML 就可以完成。是的！微软建议你将所有的文件哪怕是纯 HTML 文件都保存为 ASPX 文件后缀，这样可以加快页面的访问效率！不仅仅是在 asp.net 环境中，在 IIS5.0 以后的 ASP3.0 就已经支持这个特性了。

由于我们没有对表单提交做任何响应，所以，当你按下“ 查询 ”按钮，页面的内容没有什么改变。

下面我们将逐步使用 asp.net 的思考方式，来完成我们的页面。

2.1.3 WEB 页面处理过程

这一节我们将深入到 asp.net 内部，看看页面是怎样被处理的。

和所有的服务器端进程一样，当 ASPX 页面被客户端请求时，页面的服务器端代码被执行，执行结果被送回到浏览器端。这一点和 ASP 并没有太大的不同。

但是，asp.net 的架构为我们做了许多别的事情。比如，它会自动处理浏览器的表单提交，把各个表单域的输入值变成对象的属性，使得我们可以像访问对象属性那样来访问客户的输入。它还把客户的点击映射到不同的服务器端事件。

了解 WEB 页面的处理过程很重要。这样你可以仔细地优化你的代码，提高代码的效率。

2.1.3.1 页面的一次往返处理

用户对 Server Control 的一次操作，就可能引起页面的一次往返处理：页面被提交到服务器端，执行响应的事件处理代码，重建页面，然后返回到客户端。

正因为每个 Control 都可能引发一次页面的服务器端事件，所以，asp.net 尽量减少了控件的事件类型。很多组件都只有 OnClick 事件。特别的，asp.net 不支持服务器端的 OnMouseOver 事件。因为 OnMouseOver 事件发生得非常频繁。所以，支持服务器端的 OnMouseOver 事件是非常不现实的。

2.1.3.2 页面重建

每一次页面被请求，或者页面事件被提交到服务器，asp.net 运行环境将执行必要的代码，重建整个页面，把结果页面送到浏览器，然后抛弃页面的变量、控件的状态和属性等等页面信息。当下一次页面被处理时，asp.net 运行环境是不知道它的上一次执行情况的。在

这个意义上，ASPX 页面是没有状态的。这也是 HTTP 协议的特点（为了加速页面的访问，在 asp.net 页面里面可以使用缓存机制，也就是保存页面的执行结果，下一次页面被请求时，直接送回上一次的执行结果。）。

在 ASP 中，当页面被提交到服务器端时，只有那些用户输入的值被传递到服务器。其他的比如组件的属性、变量的值，是不会传递的。所以服务器无法了解组件的进一步的信息。

在 asp.net 中，页面对象的属性、页面控件的属性被称为“ view state ”（页面状态）。页面状态在 asp.net 中被受到特别关照。请看服务器端(**page1.aspx**)的代码：

```
<!--源文件：form\web 页面简介\page1.aspx-->
 <HTML>
 <BODY>
 <SCRIPT language="VB" runat="server">
   Sub ShowValues(Sender As Object, Args As EventArgs)
     divResult.innerText = "You selected '" _
 & selOpSys.value & "' for machine '" _
     & txtName.value & "'."
     End Sub
 </SCRIPT>
 <DIV id="divResult" runat="server">
</DIV>
<FORM runat="server">
机器名：
<INPUT type="text" id="txtName" runat="server">
   <P />
 操作系统：
       <select id="selOpSys" size="1" runat="server">
           <OPTION>Windows 95</OPTION>
          <OPTION>Windows 98</OPTION>
          <OPTION>Windows NT4</OPTION>
           <OPTION>Windows 2000</OPTION>
        </SELECT>
      <P />
 <INPUT type="submit" value="Submit" runat="server" onserverclick="ShowValues">
 </FORM>
   </BODY>
</HTML>
```

运行后将自动被解释成客户端代码，如下：

```
<HTML>
<BODY>
You selected 'Windows 98' for machine 'iceberg'.
<FORM name="ctrl0" method="post" action="pageone.aspx" id="ctrl0">
<INPUT type="hidden" name="__VIEWSTATE" value="a0z1741688109__x">
机器名:
<INPUT type="text" id="txtName" name="txtName" value="tizzy">
```

```
        <P  />
操作系统:
<SELECT  id="selOpSys"  size="1"  name="selOpSys">
   <OPTION  value="Windows  95">Windows  95</OPTION>
     <OPTION  selected  value="Windows  98">Windows  98</OPTION>
          <OPTION  value="Windows  NT4">Windows  NT4</OPTION>
   <OPTION  value="Windows  2000">Windows  2000</OPTION>
     </SELECT>
 <P  />
   <INPUT  type="submit"  value="Submit">
   </FORM>
 </BODY>
</HTML>
```

对于上面的代码，服务器端控件能在服务器端脚本中被自由运用。如果我们用传统的 ASP 代码实现上述的功能的话：

```
If  Len(Request.Form("selOpSys"))  >  0  Then
     StrOpSys  =  Request.Form("selOpSys")
     StrName  =  Request.Form("txtName")
     Response.Write("You  selected  '"  &  strOpSys  _
     &  "'  for  machine  '"  &  strName  &  "'.")
End  If
```

如果我们用 asp.net 的话，程序代码如下：

```
 If  Len(selOpSys.value)  >  0  Then
     Response.Write("You  selected  '"  &  selOpSys.value  _
     &  "'  for  machine  '"  &  txtName.value  &  "'.")
 End  If
```

通过上面例子不难看出：asp.net 页面具有组件方式的方便性和灵活性。

请注意：asp.net 通过把页面的状态封装到一个隐藏的输入域，从而可以在不同的页面之间实现传递页面的状态。

另外，asp.net 也支持应用程序一级的状态管理。这个特性在 ASP 中就已经实现。

2.1.3.3 页面处理内部过程

我们来看看页面处理的内部过程。下面的过程是依次进行的：

2.1.3.3.1 Page_load

首先，页面的状态被恢复，然后触发 Page_OnLoad 事件。在这个过程中，你可以读取或者重置页面的属性和控件的属性，根据 IsPostBack 属性判定页面是否为第一次被请求，执行数据绑定，等等。

现在我们通过一个具体的例子，来详细讲述 Page_load 事件：

我们所做的这个例子关于用户登录的。

我们先来看 page.aspx 的代码：

```
<!--源文件：form\web 页面简介\page.aspx-->
<%@ Register TagPrefix="Acme" TagName="Login" Src="page.ascx" %>
    <html>
    <title>登录演示</title>
    <script language="VB" runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)
      If (Page.IsPostBack)
      MyLabel.Text &= "用户名：" & MyLogin.UserId & "<br>"
      MyLabel.Text &= "密码： " & MyLogin.Password & "<br>"
      End If
    End Sub
</script>
<body style="font: 10pt verdana">
 <center> <h3>登录</h3></center>
  <form runat="server">
    <Acme:Login id="MyLogin" UserId="" Password="" BackColor="beige" runat="server"/>
  </form>
  <asp:Label id="MyLabel" runat="server"/>
 </body>
</html>
```

在这个文件中，我们使用了 Page_OnLoad 事件的 IsPostBack 属性，用来显示用户登录时的用户名和密码。

在来看一下 page.ascx 文件：

```
    <!--源文件：form\web 页面简介\page.ascx-->
    <script language="VB" runat="server">
    Public BackColor As String = "white"
    Public Property UserId As String
      Get
        Return UserName.Text
      End Get
      Set
        UserName.Text = Value
      End Set
    End Property
    Public Property Password As String
      Get
        Return Pass.Text
      End Get
      Set
        Pass.Text = Value
      End Set
    End Property
 </script>
```

```
<center>
<table style="background-color:<%=BackColor%>;font: 10pt verdana;border-width:1;
      border-style:solid;border-color:black;" cellspacing=15>
  <tr>
     <td><b>用户名: </b></td>
     <td><ASP:TextBox id="UserName" runat="server"/></td>
  </tr>
  <tr>
     <td><b>密码: </b></td>
 <td><ASP:TextBox id="Pass" TextMode="Password" runat="server"/></td>
  </tr>
  <tr>
     <td></td>
     <td><ASP:Button Text="提交" runat="server"/></td>
  </tr>
</table>
</center>
```

在这个文件中，我们设置了控件的属性。使之能在 page.aspx 中调用

程序的运行如图：

在下一个例子中，我们将使用 Page_OnLoad 事件，来执行数据绑定：

文件 **databind.aspx** 代码如下：

```
<!--源文件：form\web 页面简介\databind.aspx-->
<html>
<head>
<title>数据绑定演示</title>
  <script language="VB" runat="server">
      Sub Page_Load(sender As Object, e As EventArgs)
          If Not IsPostBack Then
              Dim values as ArrayList= new ArrayList()
              values.Add ("北京")
              values.Add ("上海")
              values.Add ("杭州")
              values.Add ("成都")
              values.Add ("重庆")
              values.Add ("西安")
              DropDown1.DataSource = values
              DropDown1.DataBind
          End If
      End Sub
  ‘定义按钮的单击事件
    Sub SubmitBtn_Click(sender As Object, e As EventArgs)
    ’结果显示
        Label1.Text = "你选择的城市是： " + DropDown1.SelectedItem.Text
    End Sub
```

```
    </script>
</head>
<body>
<center><h3><font face="Verdana">数据绑定演示</font></h3></center>
    <form runat=server>
<center><asp:DropDownList id="DropDown1" runat="server" /></center>
<center><asp:button Text="提交" OnClick="SubmitBtn_Click" runat=server/></center>
        <p>
<center><asp:Label id=Label1 font-name="Verdana" font-size="10pt" runat="server" /></center>
   </form>
  <body>
</html>
```

程序运行效果如图：

当我们点击“ 提交 ”按钮时：

在下面的例子中，我们将用 page_load 事件来对数据库进行连接：

我们还要说明的是如果使用 SQL 语句对数据库进行操作的时候，就需要在页面中导入 System.Data 和 System.Data.SQL 名字控件，文件 **pagedata.aspx** 的代码如下：

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SQL" %>
```

程序代码如下（pagedata.aspx）：

```
<!--源文件：form\web 页面简介\pagedata.aspx-->
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SQL" %>
<html>
<script language="VB" runat="server">
Sub Page_Load(Src As Object, E As EventArgs)
    Dim DS As DataSet
    Dim MyConnection As SQLConnection
    Dim MyCommand As SQLDataSetCommand
    ‘ 同数据库进行连接，采用 sql server 数据库
    MyConnection = New SQLConnection("server='iceberg';uid=sa;pwd=;database=info")
    ‘ 执行 SQL 操作
    MyCommand = New SQLDataSetCommand("select * from infor",MyConnection)
    DS = New DataSet()
    MyCommand.FillDataSet(ds, "infor")
    MyDataGrid.DataSource=ds.Tables("infor").DefaultView
    MyDataGrid.DataBind()
End Sub
</script>
<center>
<body>
  <h3><font face="Verdana">Page_load 事件演示</font></h3>
  <ASP:DataGrid id="MyDataGrid" runat="server"
    Width="600"
    BackColor="white"
    BorderColor="black"
    ShowFooter="false"
    CellPadding=3
    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="8pt"
    HeaderStyle-BackColor="#aaaadd"
    MaintainState="false"
  />
/body>
</center>
</html>
```

在这个程序中，我们在 page_load 事件中，我们做了哪些事呢？
① 与数据库连接。在这个例子中，我们使用 SQL Server 作为后台数据库。在这个库中，我们建立了 info 数据库，在数据库中有一张 infor 表。
② 执行 SQL 操作
③ 将筛选后的数据显示出来
我们再来看看程序运行的效果：

上面就是对 Page_load 事件的介绍，相信大家通过例子能对该事件有个理解。

2.1.3.3.2 事件处理

这一阶段处理表单的事件。你可以处理特定的事件，也可以在表单需要校验的情况下，根据 IsValid 属性判定页面的输入是否有效。

Web Form 提供了一些具有验证功能的服务器控件。这些控件提供了一套简单易用并且很强大的功能能检查输入时是否有错误。而且，还能显示提示信息给用户。

对于每个控件来说，都有一特定的属性，来验证输入的值是否有效。我们来看一下对输入控件需要验证的属性：

控件	需要验证的属性
HtmlInputText	Value
HtmlTextAreaHtm	Value
HtmlSelect	Value
HtmlInputFile	Value
TextBox	Text
ListBox	SelectedItem
DropDownList	SelectedItem
RadioButtonList	SelectedItem

好了，有了上面的介绍，我们就以例子来讲解表单的有效性验证。

在下面一个简单的例子中，我们将对用户的输入验证。

如 Validate.aspx 的内容如下：

```
<!--源文件：form\web 页面简介\validate.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub ValidateBtn_Click(sender As Object, e As EventArgs)
            If (Page.IsValid) Then
                lblOutput.Text = "页面有效!"
             Else
                lblOutput.Text = "在页面中不能出现空项!"
             End If
        ‘判断是否输入为数字
            if not isnumeric(TextBox1.text) then
             lbloutput.text="请输入数值!"
            End if
        End Sub
    </script>
 </head>
<body>
<center><h3><font face="Verdana">验证表单的例子</font></h3></center>
<p>
<form runat="server">
<title>表单验证</title>
<center>
 <table bgcolor="white" cellpadding=10>
    <tr valign="top">
      <td colspan=3>
        <asp:Label  ID="lblOutput"  Text="请填写下面的内容"  ForeColor="red"
Font-Name="Verdana" Font-Size="10" runat=server /><br>
      </td>
    </tr>
    <tr>
      <td align=right>
        <font face=Verdana size=2>储蓄卡类型:</font>
      </td>
      <td>
        <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow" runat=server>
            <asp:ListItem>绿卡</asp:ListItem>
            <asp:ListItem>牡丹卡</asp:ListItem>
        </ASP:RadioButtonList>
      </td>
```

```
        <td align=middle rowspan=1>
          <asp:RequiredFieldValidator id="RequiredFieldValidator1"
                ControlToValidate="RadioButtonList1"
                Display="Static"
                InitialValue="" Width="100%" runat=server>
                *
          </asp:RequiredFieldValidator>
        </td>
      </tr>
      <tr>
        <td align=right>
          <font face=Verdana size=2>卡号:</font>
        </td>
        <td>
          <ASP:TextBox id=TextBox1 runat=server />
        </td>
        <td>
          <asp:RequiredFieldValidator id="RequiredFieldValidator2"
                ControlToValidate="TextBox1"
                Display="Static"
                Width="100%" runat=server>
                *
          </asp:RequiredFieldValidator>
        </td>
      </tr>
        <td>
      </tr>
      <tr>
        <td></td>
        <td>
          <ASP:Button id=Button1 text="验证" OnClick="ValidateBtn_Click" runat=server />
        </td>
        <td></td>
      </tr>
      </table>
</center>
</form>
</body>
</html>
```

我们对验证按钮的 OnClick 事件进行编程，其中用到了 IsNumeric()函数，用来判断变量是否为数值型的。我们还可以用 IsData()函数对输入的日期进行判断。IsData()接受的合法日期为 100 年 1 月 1 日到 9999 年 12 月 31 日。

运行如图：

当我们在卡号一栏中输入一些字母，而不是数值时，页面上将会提示你输入数值。

让我们再举一个很有用的验证应用：

当用户在填写个人信息的时候，往往需要输入身份证号，那么我们是如何进行身份证号的验证呢？

要解决这个问题，首先，让我们先看看我国的身份证号是如何编码的。

1　　2　　3　　4　5

XX XXXX XXXXXX XX　X　（这个是没有升位以前的一个身份证号码的组成方式）

1 省　2 地市 3 生日 4 顺序码 5 性别

在这个例子中，我们只对省份进行判断。

身份编码一览表：

北京	11	吉林	22	福建	35	广东	44	云南	53
天津	12	黑龙江	23	江西	36	广西	45	西藏	54
河北	13	上海	31	山东	37	海南	46	陕西	61
山西	14	江苏	32	河南	41	重庆	50	甘肃	62
内蒙古	15	浙江	33	湖北	42	四川	51	青海	63
辽宁	21	安徽	34	湖南	43	贵州	52	宁夏	64
新疆	65	台湾	71	香港	81	澳门	82	国外	91

在这个程序中，仅仅作了一个简单的判断

Validate1.aspx 的文件内容如下：

```
<!--源文件：form\web 页面简介\validate1.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub ValidateBtn_Click(sender As Object, e As EventArgs)
```

```
            If (Page.IsValid) Then
                lblOutput.Text = "页面有效!"
            Else
                lblOutput.Text = "在页面中不能出现空项!"
            End If
            If    not isnumeric(TextBox1.text) then
                bloutput.text="请输入数值!"
            End if
            ‘ 在这里我们只作了一个简单的判断。使用了 left$（）函数
            if left$(textbox1.text,2)<>"11" then
                lbloutput.text="请验证你的身份证输入"
            End if
 End Sub
    </script>
</head>
<body>
<center><h3><font face="Verdana">验证表单的例子</font></h3></center>
<p>
<form runat="server">
<title>表单验证</title>
<center>
  <table bgcolor="white" cellpadding=10>
      <tr valign="top">
        <td colspan=3>
          <asp:Label  ID="lblOutput"  Text="请填写下面的内容"  ForeColor="red"
Font-Name="Verdana" Font-Size="10" runat=server /><br>
        </td>
      </tr>
      <tr>
        <td align=right>
          <font face=Verdana size=2>身份证号:</font>
        </td>
        <td>
          <ASP:TextBox id=TextBox1 runat=server />
        </td>
        <td>
          <asp:RequiredFieldValidator id="RequiredFieldValidator2"
              ControlToValidate="TextBox1"
              Display="Static"
              Width="100%" runat=server>
              *
          </asp:RequiredFieldValidator>
        </td>
      </tr>
```

```
        <td>
    </tr>
    <tr>
        <td></td>
        <td>
            <ASP:Button id=Button1 text="验证" OnClick="ValidateBtn_Click" runat=server />
        </td>
        <td></td>
    </tr>
    </table>
</center>
</form>
</body>
</html>
```

在这个程序中，我们仅对北京地区的身份证号进行了验证，我们使用 Left$()函数把字符串的前两个字符取出进行比较。如果大家感兴趣的话，可以把这个程序补充完整。
程序的运行如图：

这是输入正确的情况，如输入不正确，则显示（如图）：

我们在验证的时候，有时需要进行特殊的验证。在下面的表中，列出了需要进行特殊验证时要使用的特殊控件。

控件	描述
RequiredFieldValidator	使用户在输入时，不是使这一项为空
CompareValidator	对两个控件的值进行比较
RangeValidator	对输入的值进行控制，使其值界定在一定范围内
RegularExpressionValidator	把用户输入的字符和自定义的表达式进行比较
CustomValidator	自定义验证方式
ValidationSummary	在一个页面中显示总的验证错误

现在对各个验证控件介绍：

1．RequiredFieldValidator

下面的这个例子，演示了 **RequiredFieldValidator 控件**的使用方法。

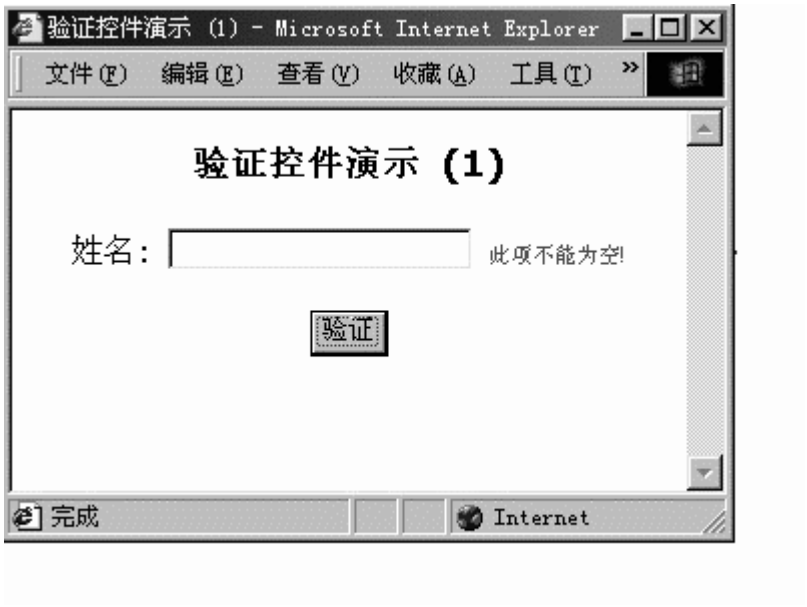
validate3.aspx 文件：

```
<!--源文件：form\web 页面简介\validate3.aspx-->
<html>
<body>
<center>
<title>验证控件演示 (1)</title>
<h3><font face="Verdana">验证控件演示 (1)</font></h3>
<form runat=server>
    姓名: <asp:TextBox id=Text1 runat="server"/>
    <asp:RequiredFieldValidator id="RequiredFieldValidator1" ControlToValidate="Text1"
Font-Name="Arial" Font-Size="11" runat="server">
```

```
            此项不能为空!
        </asp:RequiredFieldValidator>
        <p>
        <asp:Button id="Button1" runat="server" Text="验证" />
    </form>
</center>
</body>
</html>
```

当我们不在文本框中输入内容的时候，页面上将会出现不能为空的提示。

程序运行如下：



2．CompareValidator 控件

为了比较两个控件的值，此时我们需要使用 **CompareValidator** 控件。

在下面的这个例子中，我们将讲解 **CompareValidator** 控件的用法。

先看文件 **validata4.aspx**：

```
<!--源文件：form\web 页面简介\validate4.aspx-->
<%@ Page clienttarget=downlevel %>
<html>
<title>CompareValidator 控件示例</title>
<head>
    <script language="VB" runat="server">
        Sub Button1_OnSubmit(sender As Object, e As EventArgs)
            If Page.IsValid Then
                lblOutput.Text = "比较正确!"
            Else
                lblOutput.Text = "比较不正确!"
            End If
        End Sub
        Sub lstOperator_SelectedIndexChanged(sender As Object, e As EventArgs)
```

```
            comp1.Operator = lstOperator.SelectedIndex
            comp1.Validate
        End Sub
    </script>
</head>
<body>
<center>
    <h3><font face="Verdana">CompareValidator 控件示例</font></h3>
    <form runat=server>
      <table bgcolor="#eeeeee" cellpadding=10>
      <tr valign="top">
        <td>
            <h5><font face="Verdana">字符串 1:</font></h5>
            <asp:TextBox Selected id="txtComp" runat="server"></asp:TextBox>
        </td>
        <td>
            <h5><font face="Verdana">比较运算符:</font></h5>
            <asp:ListBox                                                    id="lstOperator"
OnSelectedIndexChanged="lstOperator_SelectedIndexChanged" runat="server">
                    <asp:ListItem Selected Value="Equal" >=</asp:ListItem>
                    <asp:ListItem Value="NotEqual" ><></asp:ListItem>
                    <asp:ListItem Value="GreaterThan" >></asp:ListItem>
                    <asp:ListItem Value="GreaterThanEqual" >>=</asp:ListItem>
                    <asp:ListItem Value="LessThan" ><</asp:ListItem>
                    <asp:ListItem Value="LessThanEqual" >=<</asp:ListItem>
            </asp:ListBox>
        </td>
        <td>
            <h5><font face="Verdana">字符串 2:</font></h5>
            <asp:TextBox id="txtCompTo" runat="server"></asp:TextBox><p>
            <asp:Button    runat=server    Text="  验  证  "    ID="Button1"
onclick="Button1_OnSubmit" />
        </td>
      </tr>
      </table>
      <asp:CompareValidator id="comp1" ControlToValidate="txtComp" ControlToCompare =
"txtCompTo" Type="String" runat="server"/>
      <br>
     <asp:Label ID="lblOutput" Font-Name="verdana" Font-Size="10pt" runat="server"/>
    </form>
</center>
</body>
</html>
```

在上面的代码中，我们实现了对两个控件的值进行比较。

程序运行如下：

当我们在两个文本框中输入值，然后选定运算符后，点验证按钮后，在页面上将显示

比较结果:：

3．RangeValidator 控件

RangeValidator 控件主要界定输入的值的范围。因为有时我们要求输入的值是要有一定范围的，所以我们要使用 RangeValidator 来判断。

在下面的这个例子中，我们将来介绍 RangeValidator 控件。

请看 **validata5.aspx** 的程序内容：

```
<!--源文件：form\web 页面简介\validate5.aspx-->
<%@ Page clienttarget=downlevel %>
<html>
<center>
<title>RangeValidator 控件演示</title>
<head>
    <script language="VB" runat="server">
        Sub Button1_Click(sender As Object, e As EventArgs)
            If (Page.IsValid) Then
                lblOutput.Text = "结果正确!"
            Else
                lblOutput.Text = "结果不正确!"
            End If
        End Sub
```

```
        Sub lstOperator_SelectedIndexChanged(sender As Object, e As EventArgs)
            rangeVal.Type = lstType.SelectedIndex
            rangeVal.Validate
        End Sub
    </script>
</head>
<body>

    <h3><font face="Verdana">RangeValidator 控件演示</font></h3>
    <p>
    <form runat="server">
      <table bgcolor="#eeeeee" cellpadding=10>
      <tr valign="top">
        <td>
            <h5><font face="Verdana">输入要验证的值:</font></h5>
            <asp:TextBox Selected id="txtComp" runat="server"/>
        </td>
        <td>
            <h5><font face="Verdana">数据类型:</font></h5>
            <asp:DropDownList                                                    id="lstType"
OnSelectedIndexChanged="lstOperator_SelectedIndexChanged"    runat=server>
                <asp:ListItem Selected Value="String" >String</asp:ListItem>
                <asp:ListItem Value="Integer" >Integer</asp:ListItem>
            </asp:DropDownList>
        </td>
        <td>
            <h5><font face="Verdana">最小值:</font></h5>
            <asp:TextBox id="txtMin" runat="server" />
        </td>
        <td>
            <h5><font face="Verdana">最大值:</font></h5>
            <asp:TextBox id="txtMax" runat="server" /><p>
            <asp:Button  Text="验证"  ID="Button1"  onclick="Button1_Click"  runat="server"
/>
        </td>
      </tr>
    </table>
    <asp:RangeValidator     id="rangeVal"     Type="String"     ControlToValidate="txtComp"
MaximumControl="txtMax" MinimumControl="txtMin" runat="server"/>
    <br>
    <asp:Label id="lblOutput" Font-Name="verdana" Font-Size="10pt" runat="server" />
    </form>
</body>
</center>
```

</html>

当我们在三个文本框中分别输入要验证的值，最大值，和最小值，然后按下验证按钮，页面上将显示判断的结果。

在本例中我们只能比较 integer 和 string 的值，当然，我们也可以增加数据类型，如 double 型，float 型，date 型，currency 型等。

结果运行如下：

4. RegularExpressionValidator 控件

我们在制作网站的时候,尤其是各种电子商务网站,首先都会让用户填写一些表格来获取注册用户的各种信息,因为用户有可能输入各式各样的信息,而有些不符合要求的数据会给我们的后端 ASP 处理程序带来不必要的麻烦,甚至导致网站出现一些安全问题。因此我们在将这些信息保存到网站的数据库之前,要对这些用户所输入的信息进行数据的合法性校验,以便后面的程序可以安全顺利的执行。

使用 RegularExpressionValidator 服务器控件，可以用来检查我们输入的信息是否和我们的自定义的表达式一致。比方说用它可以检查 e-mail 地址，电话号码等合法性。

在讲述 RegularExpressionValidator 服务器控件使用之前，我们先来了解一下正则表达式（RegularExpression）的来源：

正则表达式的“祖先”可以一直上溯至对人类神经系统如何工作的早期研究。Warren McCulloch 和 Walter Pitts 这两位神经生理学家研究出一种数学方式来描述这些神经网络。1956 年, 一位叫 Stephen Kleene 的美国数学家在 McCulloch 和 Pitts 早期工作的基础上，发表了一篇标题为“神经网事件的表示法”的论文，引入了正则表达式的概念。正则表达式就是用来描述他称为“正则集的代数”的表达式，因此采用“正则表达式”这个术语。随后，发现可以将这一工作应用于使用 Ken Thompson 的计算搜索算法的一些早期研究，Ken Thompson 是 Unix 的主要发明人。正则表达式的第一个实用应用程序就是 Unix 中的 qed 编辑器。如他们所说，剩下的就是众所周知的历史了。从那时起直至现在正则表达式都是基于文本的编辑器和搜索工具中的一个重要部分。

其实，正则表达式（RegularExpression）是一个正则表达式就是由普通字符（例如字符 a 到 z）以及特殊字符（称为元字符）组成的文字模式。该模式描述在查找文字主体时待匹

配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

使用正则表达式，就可以：

1. 测试字符串的某个模式。例如，可以对一个输入字符串进行测试，看在该字符串是否存在一个电话号码模式或一个信用卡号码模式。这称为数据有效性验证。

2. 替换文本。可以在文档中使用一个正则表达式来标识特定文字，然后可以全部将其删除，或者替换为别的文字。

3. 根据模式匹配从字符串中提取一个子字符串。可以用来在文本或输入字段中查找特定文字。

例如，如果需要搜索整个 web 站点来删除某些过时的材料并替换某些 HTML 格式化标记，则可以使用正则表达式对每个文件进行测试，看在该文件中是否存在所要查找的材料或 HTML 格式化标记。用这个方法，就可以将受影响的文件范围缩小到包含要删除或更改的材料的那些文件。然后可以使用正则表达式来删除过时的材料，最后，可以再次使用正则表达式来查找并替换那些需要替换的标记。

另一个说明正则表达式非常有用的示例是一种其字符串处理能力还不为人所知的语言。VBScript 是 Visual Basic 的一个子集，具有丰富的字符串处理功能。与 C 类似的 Visual Basic Scripting Edition 则没有这一能力。正则表达式给 Visual Basic Scripting Edition 的字符串处理能力带来了明显改善。不过，可能还是在 VBScript 中使用正则表达式的效率更高，它允许在单个表达式中执行多个字符串操作。

正是由于“ 正则表达式 ”的强大功能，才使得微软慢慢将正则表达式对象移植到了视窗系统上面。在书写正则表达式的模式时使用了特殊的字符和序列。下表描述了可以使用的字符和序列，并给出了实例。

字符描述： \：将下一个字符标记为特殊字符或字面值。例如"n"与字符"n"匹配。"\n"与换行符匹配。序列"\\"与"\"匹配，"\("与"("匹配。

^ ：匹配输入的开始位置。

$ ：匹配输入的结尾。

* ：匹配前一个字符零次或几次。例如，"zo*"可以匹配"z"、"zoo"。

+ ：匹配前一个字符一次或多次。例如，"zo+"可以匹配"zoo",但不匹配"z"。

? ：匹配前一个字符零次或一次。例如，"a?ve?"可以匹配"never"中的"ve"。

.：匹配换行符以外的任何字符。

(pattern) 与模式匹配并记住匹配。匹配的子字符串可以从作为结果的 Matches 集合中使用 Item [0]...[n]取得。如果要匹配括号字符(和)，可使用"\(" 或 "\)"。

x|y：匹配 x 或 y。例如 "z|food" 可匹配 "z" 或 "food"。"(z|f)ood" 匹配 "zoo" 或 "food"。

{n}：n 为非负的整数。匹配恰好 n 次。例如，"o{2}" 不能与 "Bob 中的 "o" 匹配，但是可以与"foooood"中的前两个 o 匹配。

{n,} ：n 为非负的整数。匹配至少 n 次。例如，"o{2,}"不匹配"Bob"中的"o"，但是匹配"foooood"中所有的 o。"o{1,}"等价于"o+"。"o{0,}"等价于"o*"。

{n,m} ：m 和 n 为非负的整数。匹配至少 n 次，至多 m 次。例如，"o{1,3}" 匹配"fooooood"中前三个 o。"o{0,1}"等价于"o?"。

[xyz] ：一个字符集。与括号中字符的其中之一匹配。例如，"[abc]" 匹配"plain"中的"a"。

[^xyz] ：一个否定的字符集。匹配不在此括号中的任何字符。例如，"[^abc]" 可以匹配"plain"中的"p".

[a-z] ：表示某个范围内的字符。与指定区间内的任何字符匹配。例如，"[a-z]"匹配"a"

与"z"之间的任何一个小写字母字符。

[^m-z] ：否定的字符区间。与不在指定区间内的字符匹配。例如，"[m-z]"与不在"m"到"z"之间的任何字符匹配。

\b ：与单词的边界匹配，即单词与空格之间的位置。例如，"er\b" 与"never"中的"er"匹配，但是不匹配"verb"中的"er"。

\B ：与非单词边界匹配。"ea*r\B"与"never early"中的"ear"匹配。

\d ：与一个数字字符匹配。等价于[0-9]。

\D ：与非数字的字符匹配。等价于[^0-9]。

\f ：与分页符匹配。

\n ：与换行符字符匹配。

\r ：与回车字符匹配。

\s ：与任何白字符匹配，包括空格、制表符、分页符等。等价于"[\f\n\r\t\v]"。

\S ：与任何非空白的字符匹配。等价于"[^ \f\n\r\t\v]"。

\t ：与制表符匹配。

\v ：与垂直制表符匹配。

\w ：与任何单词字符匹配，包括下划线。等价于"[A-Za-z0-9_]"。

\W ：与任何非单词字符匹配。等价于"[^A-Za-z0-9_]"。

\num ：匹配 num 个，其中 num 为一个正整数。引用回到记住的匹配。例如，"(.)\1"匹配两个连续的相同的字符。

\n：匹配 n，其中 n 是一个八进制换码值。八进制换码值必须是 1, 2 或 3 个数字长。

例如，"\11" 和 "\011" 都与一个制表符匹配。"\0011"等价于"\001" 与 "1"。八进制换码值不得超过 256。否则，只有前两个字符被视为表达式的一部分。允许在正则表达式中使用 ASCII 码。

\xn：匹配 n，其中 n 是一个十六进制的换码值。十六进制换码值必须恰好为两个数字长。例如，"\x41"匹配"A"。"\x041"等价于"\x04" 和 "1"。允许在正则表达式中使用 ASCII 码。

RegularExpressionValidator 有两种主要的属性来进行有效性验证。ControlToValidate 包含了一个值进行验证。如取出文本框中的值。如 ControlToValidate="TextBox1" ValidationExpression 包含了一个正则表达式进行验证。

好了，有了上面的叙述，我们就举个例子来说明正则表达式。比如，我们想要对用户输入的电子邮件进行校验，那么，什么样的数据才算是一个合法的电子邮件呢？我可以这样输入：test@yesky.com，当然我也会这样输入：xxx@yyy.com.cn,但是这样的输入就是非法的：xxx@@com.cn 或者@xxx.com.cn，等等，所以我们得出一个合法的电子邮件地址至少应当满足以下几个条件：

1. 必须包含一个并且只有一个符号“ @”
2. 第一个字符不得是“ @”或者“ .”
3. 不允许出现“ @.”或者.@
4. 结尾不得是字符“ @”或者“ .”

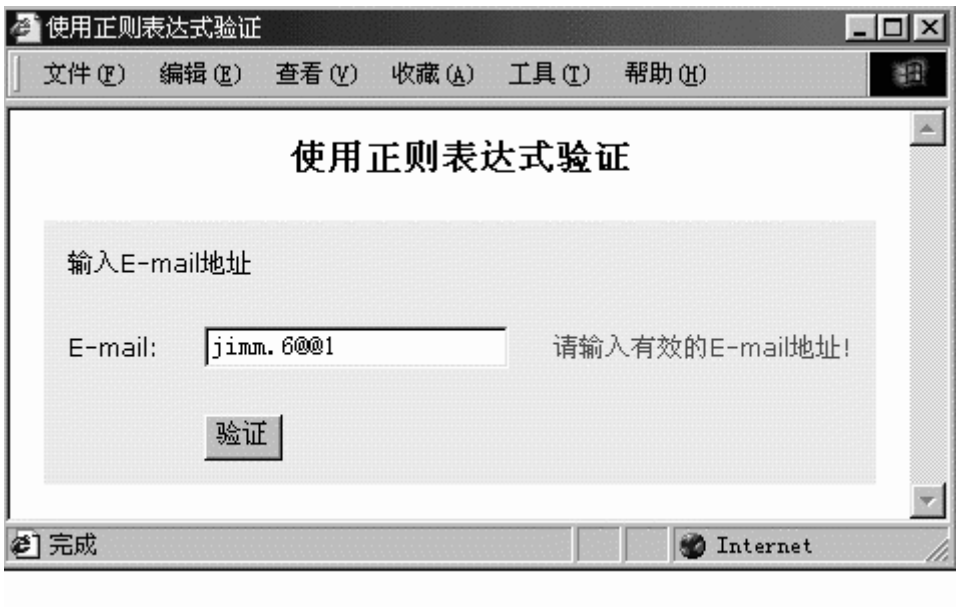
所以根据以上的原则和上面表中的语法，我们很容易的就可以得到需要的模板如下："="^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$"

请看 **validata6.aspx** 的内容：

```
<!--源文件：form\web 页面简介\validate6.aspx-->
</head>
<body>
```

```
<center><h3><font face="Verdana">使用正则表达式验证</font></h3></center>
<p>
<form runat="server">
<center>
<title>使用正则表达式验证</title>
    <table bgcolor="#eeeeee" cellpadding=10>
    <tr valign="top">
      <td colspan=3>
        <asp:Label ID="lblOutput" Text="输入 E-mail 地址" Font-Name="Verdana"
Font-Size="10pt" runat="server"/>
      </td>
    </tr>
    <tr>
      <td align=right>
        <font face=Verdana size=2>E-mail:</font>
      </td>
      <td>
        <ASP:TextBox id=TextBox1 runat=server />
      </td>
      <td>
        <asp:RegularExpressionValidator id="RegularExpressionValidator1" runat="server"
            ControlToValidate="TextBox1"

ValidationExpression="^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+
$"
            Display="Static"
            Font-Name="verdana"
            Font-Size="10pt">
              请输入有效的 E-mail 地址!
        </asp:RegularExpressionValidator>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <ASP:Button text="验证" OnClick="ValidateBtn_Click" runat=server />
      </td>
      <td></td>
    </tr>
    </table>
</center>
</form>
</body>
</html>
```

这样，我们只要定制不同的模板，就可以实现对不同数据的合法性校验了。所以，正则表达式对象中最重要的属性就是：“ Pattern”属性,只要真正掌握了这个属性，才可以自由的运用正则表达式对象来为我们的数据校验进行服务。
程序的运行效果如图：



通过上面的介绍，我们对数据验证的方法有了一定的认识。在下面的内容中，我们还将通过更具体的实例，来对数据的有效性验证进行讲解。

2.1.3.3.3 Page_Unload

这个阶段页面已经处理完毕，需要做些清理工作。一般地，你可以在这个阶段关闭打开的文件和数据库链路，或者释放对象。
1、断开数据库连接
请看如下脚本：

```
<script language="VB" runat="server">
‘定义一个共有变量
    public Dim MyConnection As SQLConnection
‘ 定义 Page_Load 事件
    Sub Page_Load(Src As Object, E As EventArgs)
        Dim DS As DataSet

        Dim MyCommand As SQLDataSetCommand
          MyConnection = New SQLConnection("server='iceberg';uid=sa;pwd=;database=info")
MyCommand = New SQLDataSetCommand("select * from infor",MyConnection)
            Myconnection.open()
            DS = New DataSet()
            MyCommand.FillDataSet(ds, "infor")
            MyDataGrid.DataSource=ds.Tables("infor").DefaultView
```

```
                MyDataGrid.DataBind()
 End Sub
’定义 Page_UnLoad 事件
 Sub Page_UnLoad(Src As Object, E As EventArgs)
’与数据库断开连接
    MyConnection.Close()
 End Sub
```

现在我们在来看一个对文件操作的例子。

在这个例子中，我们使用的了两个事件，Page_Load 事件和 Page_Unload 事件。在 Page_Load 事件先创建一个文件，然后向这个文件中写入内容。在 Page_Unload 事件中我们将此文件关闭。

代码如下：

```
<%@ import namespace="system.io" %>
<html>
<head>
<title>ASP.NET 测试 写 文本文件</title>
</head>
<body>
<script language="vb" runat="server">
public   Dim writeFile As StreamWriter
Sub Page_Load(Sender As Object,E as EventArgs)
writeFile = File.CreateText( "c:\test.txt" )
writeFile.WriteLine( "这是一个测试文件!" )
writeFile.WriteLine( "使用了 Page_Load 事件和 Page_Unload 事件!" )
Response.Write( "test.txt 创建 并 写入 成功!" )
End Sub
Sub Page_UnLoad(Sender AS Object, E as EventArgs)
writeFile.Close
End Sub
</script>
</body>
</html>
```

这样，我们就使用了 Page_Load 事件和 Page_Unload 事件。很明显，我们定义 Page_Load 事件，是因为这个阶段页面已经处理完毕，需要做些清理工作。

上面我们分析了页面处理最重要的几个阶段。应该说明的是：页面的处理过程远比上面的复杂，因为每个控件都需要初始化。在后面的章节中，我们还将了解到更加详细的页面处理过程。

2.1.4 Web Form 事件模型

在 asp.net 中，事件是一个非常重要的概念。我们举两个例子来说明在 Web Form 中的应用。

2.1.4.1 例子一：多按钮事件

我们在一个<form></form>里面有几个按钮，多个事件的响应我们该怎么处理呢？在 asp.net 中有很好的处理机制，我们可以在一个页面中写几个方法来分别响应不同的事件。

在下面的例子中，将根据五个按钮的功能，我们定义了五个方法：AddBtn_Click(Sender As Object, E As EventArgs)、AddAllBtn_Click(Sender As Object, E As EventArgs)、RemoveBtn_Click(Sender As Object, E As EventArgs)、RemoveAllBtn_Click(Sender As Object, E As EventArgs)、result(Sender As Object,E As EventArgs)，分别用来处理全部加进、单个加进、单个取消、全部取消和提交事件。我们的 form 提交的时候，还是提交给本页面，由本页面进行处理，代码如下：

<form action="menent.aspx" runat=server>

其中，menent.aspx 就是本页面。

Menent.aspx 文件代码如下：

```
<!--源文件：form\web 页面简介\menent.aspx-->
<html>

    <script language="VB" runat="server">

        Sub AddBtn_Click(Sender As Object, E As EventArgs)

            If Not (AvailableFonts.SelectedIndex = -1)
                InstalledFonts.Items.Add(New ListItem(AvailableFonts.SelectedItem.Value))
                AvailableFonts.Items.Remove(AvailableFonts.SelectedItem.Value)
            End If
        End Sub

        Sub AddAllBtn_Click(Sender As Object, E As EventArgs)

            Do While Not (AvailableFonts.Items.Count = 0)
                InstalledFonts.Items.Add(New ListItem(AvailableFonts.Items(0).Value))
                AvailableFonts.Items.Remove(AvailableFonts.Items(0).Value)
            Loop
        End Sub

        Sub RemoveBtn_Click(Sender As Object, E As EventArgs)

            If Not (InstalledFonts.SelectedIndex = -1)
                AvailableFonts.Items.Add(New ListItem(InstalledFonts.SelectedItem.Value))
                InstalledFonts.Items.Remove(InstalledFonts.SelectedItem.Value)
            End If
        End Sub
```

```
    Sub RemoveAllBtn_Click(Sender As Object, E As EventArgs)

        Do While Not (InstalledFonts.Items.Count = 0)
            AvailableFonts.Items.Add(New ListItem(InstalledFonts.Items(0).Value))
            InstalledFonts.Items.Remove(InstalledFonts.Items(0).Value)
        Loop
    End Sub

Sub result(Sender As Object,E As EventArgs)

    dim tmpStr as String

        tmpStr="<br>"
    Do While Not (InstalledFonts.Items.Count = 0)
        tmpStr=tmpStr & InstalledFonts.items(0).value & "<br>"
        InstalledFonts.items.remove(InstalledFonts.items(0).value)
        Loop

    tmpStr=System.Web.HttpUtility.UrlEncodeToString(tmpStr,System.Text.Encoding.UTF
            8)
    Page.Navigate("result.aspx?InstalledFonts=" & tmpStr)

End Sub

</script>

<body bgcolor="#ccccff">
<center>
    <h3><font face="Verdana">.NET->不同事件的处理方法！</font></h3>
</center>
<center>
    <form action="menent.aspx" runat=server>

        <table>
            <tr>
                <td>
                    现有字体
                </td>
                <td>
                    <!-- Filler -->
                </td>
                <td>
                    选择的字体
                </td>
```

```
            </tr>
            <tr>
              <td>
                <asp:listbox id="AvailableFonts" width="100px" runat=server>
                    <asp:listitem>Roman</asp:listitem>
                    <asp:listitem>Arial Black</asp:listitem>
                    <asp:listitem>Garamond</asp:listitem>
                    <asp:listitem>Somona</asp:listitem>
                    <asp:listitem>Symbol</asp:listitem>
                </asp:listbox>
              </td>
              <td>
                <!-- Filler -->
              </td>
              <td>
                <asp:listbox id="InstalledFonts" width="100px" runat=server>
                    <asp:listitem>Times</asp:listitem>
                    <asp:listitem>Helvetica</asp:listitem>
                    <asp:listitem>Arial</asp:listitem>
                </asp:listbox>
              </td>
            </tr>
            <tr>
              <td>
                <!-- Filler -->
              </td>
              <td>
                <asp:button text="<<==" OnClick="RemoveAllBtn_Click" runat=server/>
                <asp:button text="<--" OnClick="RemoveBtn_Click" runat=server/>
                <asp:button text="-->" OnClick="AddBtn_Click" runat=server/>
                <asp:button text="==>>" OnClick="AddAllBtn_Click" runat=server/>
    <asp:label id="Message" forecolor="red" font-bold="true" runat=server/>
              </td>
              </tr>
    <tr align=center>
    <td    align=center>
                <asp:button text="提交" Onclick="result" runat=server/>
    <!-- Filler -->
              </td>
            </tr>
          </table>

      </form>
  </center>
```

```
    </body>

</html>
```

写一个页面，在提交时候接收相关信息。我们在页面进入的时候取得传送过来的数值，用：

```
    Request.Params("InstalledFonts")
```

来获得，具体来看我们的文件 **result.aspx** 的代码：

```
<!--源文件：form\web 页面简介\result.aspx-->
    <html>
    <script language="VB" runat="server">
        Sub Page_Load(Sender As Object, E As EventArgs)
            If Not (Page.IsPostBack)
                NameLabel.Text = Request.Params("InstalledFonts")
            End If
        End Sub
    </script>

    <BODY >
        <h3><font face="Verdana">.NET->多事件处理！</font></h3>
        <p>
        <p>
        <hr>
        <form action="controls_NavigationTarget.aspx" runat=server>
            <font face="Verdana">
                Hi,你的选择是: <asp:label id="NameLabel" runat=server/>!
            </font>
        </form>
    </body>
</html>
```

程序运行如下：

当我们点击提交按钮的时候，将显示：

2.1.4.2 例子二：AutoPostBack

PostBack 属性在 Page_Load 事件中出现的，在一个用户请求结束后，如果页面重新 Load，则返回一个 true。这对初始化一个页面来讲是一件非常容易的事情，下面看我们的代码：

```
Sub Page_Load(Sender as Object，e as EventArgs)
   If    IsPostBack and ( TextBox2.Text = “”)
         TextBox2.Text=”Hello” & TextBox1.Text & “!! 你好啊！”
   End If
End Sub
```

如果 IsPostBack 返回一个真值并且 TextBox2.Text 为空，程序执行它下面的语句。在另外一个方面，我们设置一个标识：

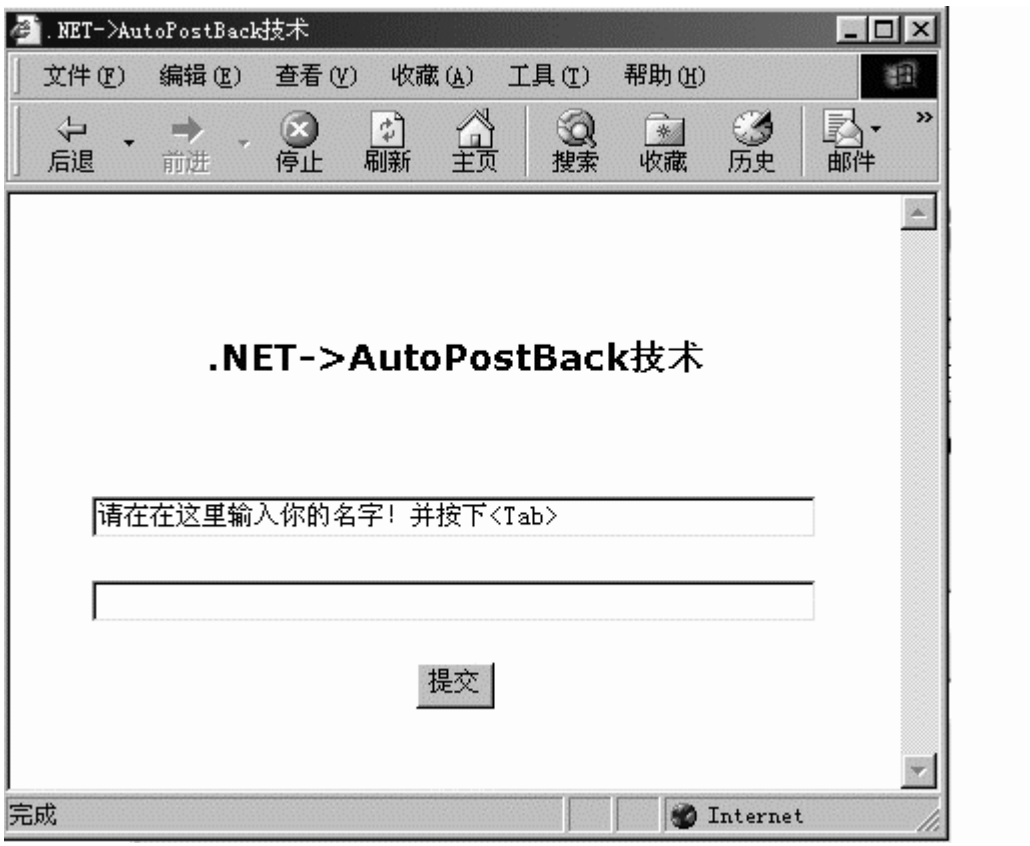
```
<asp:TextBox id="TextBox1" Text="请在在这里输入你的名字！并按下<Tab>"
      AutoPostBack="True" Columns=50 runat="server"/>
```

我们设定 AutoPostBack="True"，自动 PostBack，下面是我们的完整的代码 (autopostback..aspx)：

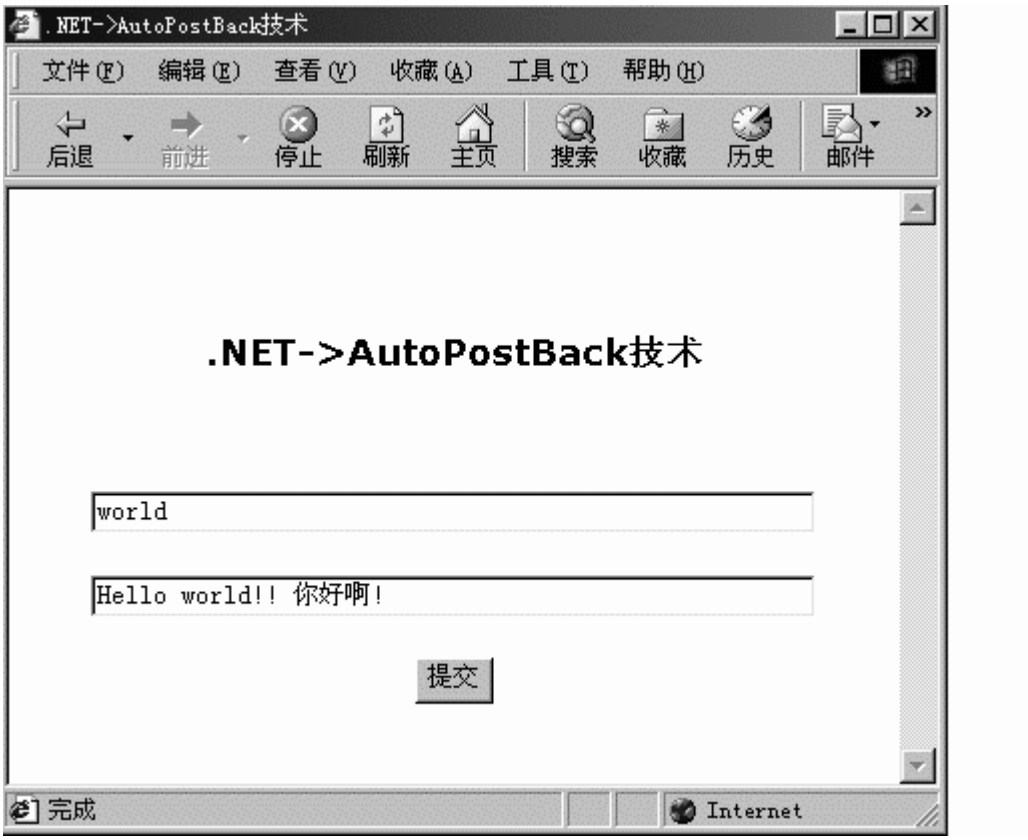
```
<!--源文件：form\web 页面简介\autopostback.aspx-->
<html>
<head>
 <script language="VB" runat="server">

      Sub Page_Load(Sender as Object，e as EventArgs)
      If    IsPostBack and ( TextBox2.Text = “”)
            TextBox2.Text=”Hello” & TextBox1.Text & “!! 你好啊！”
      End If
   End Sub

 </script>
</head>
<body>
 <center>
    <br><br><br>
      <h3><font face="Verdana">.NET->AutoPostBack 技术</font></h3>
  <br><br>
</center>
<center>
 <form runat="server">
  <p>
```

```
      <asp:TextBox id="TextBox1" Text="请在在这里输入你的名字！并按下<Tab>"
            AutoPostBack="True" Columns=50 runat="server"/>
   <p>
      <asp:TextBox id="TextBox2" Columns=50 runat="server"/>
  <p>
       <asp:Button Text="提交" Runat="server"/>
  <p>
  </form>
</center>
</body>
</html>
```

访问如下：



输入完成后，按下<Tab>键，得到如下结果：



2.1.5 小结

在这一章中，我们对 Web Form 页面进行了介绍，通过几个实例，我们分别介绍了 Server 控件，HTML Server 控件，以及 Web Form 的事件模型。在下面的章节中，我们将对本章所涉及的概念进行更深入的讲解。

第二章 服务器端控件

2.2.1 服务器端控件示例

在讲述服务器端控件的时候，我们先来讲述一个具体的例子。

我们说过，在 asp.net 里面，一切都是对象。我们也谈到：WEB 页面本身就是一个对象。

或者说，WEB 页面就是一个对象的容器。那么，这个容器可以装些什么东西呢？这一节我们学习服务器端控件，英文是 Server Control。这是 WEB 页面能够容纳的对象之一。

什么是 Control？熟悉 VB 的读者肯定再清楚不过了。简单地说，Control 就是一个可重用的组件或者对象，这个组件不但有自己的外观，还有自己的数据和方法，大部分组件还可以响应事件。通过微软的集成开发环境(Visual Studio.NET 7.0)，你可以简单地把一个 Control 拖放到一个 Form 中。

那为什么叫“ Server Control ”？这是因为这些 Control 是在服务器端存在的。服务器端控件也有自己的外观，在客户端浏览器中，Server Control 的外观由 HTML 代码来表现。Server Control 会在初始化时，根据客户的浏览器版本，自动生成适合浏览器的 HTML 代码。以前我们做网页或者做 ASP 程序时候，必须考虑到浏览器的不同版本对 HTML 的支持有所不同，比如 Netscape 和 IE 对 DHTML 的支持就有所不同。当时，解决浏览器版本兼容问题的最有效方法，就是在不同版本的浏览器中作测试。现在，由于 Server Control 自动适应不同的浏览器版本，也就是自动兼容不同版本的浏览器，程序员的工作量减轻了许多。下面，我们来看看如何在 WEB FORM 中嵌入 Server Control。我们的例子是从上一节继承来的：

如图：

下面是实现如图效果的代码：(**sample.aspx**)

```
<!--源文件：form\ServerControl\sample.aspx-->
<html>
    <script language="VB" runat=server>
        Sub SubmitBtn_Click(Sender As Object, E As EventArgs)
            Message.Text = "Hi " & Name.Text & ", 你选择的城市是: " & city.SelectedItem.Text
        End Sub
    </script>
    <body>
        <center>
        <form action="form2.aspx" method="post" runat="server">
            <h3>姓名: <asp:textbox id="Name" runat="server"/>
```

```
所在城市:  <asp:dropdownlist id="city" runat=server>
                <asp:listitem>北京</asp:listitem>
                <asp:listitem>上海</asp:listitem>
                <asp:listitem>重庆</asp:listitem>
            </asp:dropdownlist>
<asp:button type=submit text="确 定 " OnClick="SubmitBtn_Click" runat="server"/>
<p>
<asp:label id="Message" runat="server"/>
</form>
</center>
</body>
</html>
```

请注意：上面的代码中我们使用了三种 Server Control，分别是：

1、 asp:textbox

2、asp:dropdownlist

3、asp:label

我们注意到三个控件都有相同的 RunAt 属性：**RunAt=”Sevrer”**。所有的服务器端控件都有这样的属性。这个属性标志了一个控件是在 Server 端进行处理的。

我们看下面的代码：

```
<script language="VB" runat=server>
    Sub SubmitBtn_Click(Sender As Object, E As EventArgs)
        Message.Text = "Hi " & Name.Text & ", 你选择的城市是: " & city.SelectedItem.Text
    End Sub
</script>
```

用过 VB 的朋友是不是觉得很熟悉？没错，这是用 VB 写的一个事件处理函数，void SubmitBtn_Click(Object sender , EventArgs e) ，你可能一看就明白了，void 代表该函数没有返回值，该函数带有两个参数，可是这里的 Sender 的意义是什么意思呢？它的用处又到底是什么呢？其实很简单，这个 Sender 就是这个事件的触发者。这里，Sender 就是被 Click 的 button。其中代码只有一行， 你可能注意到这行代码中的 Message、Name、city 你并没有定义，那么它们从哪里来的呢？

看下面的代码：:

```
<form action="form2.aspx" method="post" runat="server">
<h3> Name: <asp:textbox id="Name" runat="server"/>
        Category:  <asp:dropdownlist id="city" runat=server>
                <asp:listitem>北京</asp:listitem>
                <asp:listitem>上海</asp:listitem>
                <asp:listitem>重庆</asp:listitem>
            </asp:dropdownlist>
<asp:button type=submit text="确定" OnClick="SubmitBtn_Click" runat="server"/>
```

```
<p>
        <asp:label id="Message" runat="server"/>
    </form>
```

我们发现每个服务端的控件都带有一个 ID 号。而我们在 VB.NET 代码中所引用的就是这些 ID。我们可以认为 ID 就是控件的名称。在 ASP 中我们也使用过 ID 吧。那时候，ID 属性和 Name 属性并没有什么不同：

```
<input id=email name=email >
```

在客户端，我们通过 VBSCript 代码或者 Jscript 代码，可以这样访问 Form 表单的 Input 域：

```
<SCRIPT LANGUAGE=javascript>
<!--
    document.all(“email”)=”darkman@yesky.com”;
//-->
</SCRIPT>
```

从上面的代码可以看出，在 DHTML 中，我们也是通过 ID 来访问 Form 表单的输入域的。在 ASPX 中，情况有些类似之处。差别在于：一个在客户端，一个在服务器端。

如果你和第一节的例子代码对比一下，你会发现：这个表单的写法和 html 表单完全不同了吧？首先，所有的表单项包括表单本身后面都加上了 runat=server，这句话的意思就是说这个是服务器端控制项，另外象传统表单的什么<input type=text>等的写法都变了，你仔细观察一下可以看出，原来的文本框变为<asp:textbox>，选择框变为<asp:dropdownlist>，选择框选项变为<asp:listitem>，而 submit 按钮变为<asp:button>，这个按钮对应的控制函数就刚才我提到的那个 SubmitBtn_Click 函数，它是运行在服务器端的。另外还有一个服务器端控制<asp:label id= ”Message ” runat= ”server ”/>，这个 asp:label 是传统表单所没有的，它是一个服务器端文本控制，那么就存在一个问题，如果传统的 HTML 里没有这个元素，那么 ASP.NET 是怎么接收的呢？你运行一下这个程序，然后看一下 HTML 源码，你会发现这么一行：

```
<input type="hidden" name="__VIEWSTATE value="…" />
```

对，ASP.NET 就是通过这个隐藏表单的形式传递过去的。

所以，一个客户端控件，加上 runat=Server 就变成服务器端控件，服务器端控件能在服务器端脚本中被自由运用。在以后的章节中，我们还要对常用的服务器端控件进行详细介绍。

2.2.2 文本输入控件

文本输入控件目的是让用户输入文本，文本模式是一个单行的输入框，但是用户可以根据自己的需要把它改成密码输入模式或者多行输入模式。

在此我们用单行文本输入模式和密码模式来说明，在 asp.net 中，输入一个文本，可用下面的语句来表示：

```
<!--输入邮件地址-->
 <asp:TextBox id=email width=200px maxlength=60 runat=server />
```

第一句为注释，我们可以设定输入框的宽度和文本的长度，runat=server 为表示运行于服务器端，下面我们来看看输入控件的校验，一个简单的语句就可以实现我们在普通的 html 页

面上的复杂的 JavaScript、VBScript 或者其他代码的验证。首先我们用户必须输入邮件地址：

```
<!--验证邮件的有效性！->不能为空-->
 <asp:RequiredFieldValidator id="emailReqVal"
      ControlToValidate="email"
      ErrorMessage="Email. "
      Display="Dynamic"
      Font-Name="Verdana" Font-Size="12"
      runat=server>
      *
 </asp:RequiredFieldValidator>
```

ControlToValidate="email"属性为针对 TextBox id=email 的文本框，Display 属性我们定义为“ Dynamic ”，即为当鼠标光标所在位置发生变化时属性根据条件变化。如果为空，则打印出“ * “ 字符出来。

在通常情况下，邮件地址总会包含一些特定的字符，我们在验证的时候，就可以要求用户输入的内容中包含我们规定的字符。

```
<!--验证邮件的有效性！->必须包含有效字符-->
<asp:RegularExpressionValidator id="emailRegexVal"
      ControlToValidate="email"
      Display="Static"
      ValidationExpression="^[\w-]+@[\w-]+\.(com|net|org|edu|mil)$"
      Font-Name="Arial" Font-Size="11"
      runat=server>
      不是有效邮件地址
</asp:RegularExpressionValidator>
```

ControlToValidate="email"语句跟上面一样，
ValidationExpression="^[\w-]+@[\w-]+\.(com|net|org|edu|mil)$"表示我们在邮件里要包含的内容，如果没有包含，则打印出“ 不是有效邮件地址“ 这个提示。

对密码也是同样的道理的，主要的差别是，对于密码，通常我们要确认一次，校验两次输入的密码是否相等。下面是我们的代码：

```
<!--输入确认密码->两个密码是否相等-->
<asp:CompareValidator id="CompareValidator1"
      ControlToValidate="passwd2" ControlToCompare="passwd"
      Display="Static"
      Font-Name="Arial" Font-Size="11"
      runat=server>
      密码不相等
</asp:CompareValidator>
```

ControlToValidate="passwd2" ControlToCompare="passwd"此语句即为两个密码之间的比较，不相等，打印出“ 密码不相等“ 的提示。

下面是我们的完整的代码(**textbox.aspx**)：

```
<!--源文件：form\ServerControl\textbox.aspx-->
<html>
<body>
 <br><br><br>
  <center>
    <h3><font face="Verdana">.NET->文本控件</font></h3>
  </center>
    <form method=post runat=server>
    <hr width=600 size=1 noshade>
 <br><br>
    <center>
    <!--标题-->
    <asp:ValidationSummary ID="valSum" runat="server"
        HeaderText="按照下面的要求填写:"
        DisplayMode="SingleParagraph"
        Font-Name="verdana"
        Font-Size="12"
        />
    <p>
    <table border=0 width=600>
    <tr>
      <td align=right>
        <font face=Arial size=2>电子邮件:</font>
      </td>
      <td>
    <!--输入邮件地址-->
     <asp:TextBox id=email width=200px maxlength=60 runat=server />
      </td>
      <td>
    <!--验证邮件的有效性！->不能为空-->
        <asp:RequiredFieldValidator id="emailReqVal"
            ControlToValidate="email"
            ErrorMessage="Email. "
            Display="Dynamic"
            Font-Name="Verdana" Font-Size="12"
            runat=server>
            *
        </asp:RequiredFieldValidator>
    <!--验证邮件的有效性！->必须包含有效字符-->
        <asp:RegularExpressionValidator id="emailRegexVal"
            ControlToValidate="email"
            Display="Static"
```

```
            ValidationExpression="^[\w-]+@[\w-]+\.(com|net|org|edu|mil)$"
            Font-Name="Arial" Font-Size="11"
            runat=server>
            不是有效邮件地址
        </asp:RegularExpressionValidator>
    </td>
</tr>

<tr>
    <td align=right>
        <font face=Arial size=2>密码:</font>
    </td>
    <td>
<!--输入密码-->
        <asp:TextBox id=passwd TextMode="Password" maxlength=20 runat=server/>
    </td>
    <td>
<!--输入密码->密码不能为空-->
        <asp:RequiredFieldValidator id="passwdReqVal"
            ControlToValidate="passwd"
            ErrorMessage="Password. "
            Display="Dynamic"
            Font-Name="Verdana" Font-Size="12"
            runat=server>
            *
        </asp:RequiredFieldValidator>
<!--输入密码->包含其中有效字符-->
        <asp:RegularExpressionValidator id="passwdRegexBal"
            ControlToValidate="passwd"
            ValidationExpression=".*[!@#$%^&*+;:].*"
            Display="Static"
            Font-Name="Arial" Font-Size="11"
            Width="100%" runat=server>
            密码必须包含 (!@#$%^&amp;*+;:)
        </asp:RegularExpressionValidator>
    </td>
</tr>
<tr>
    <td align=right>
        <font face=Arial size=2>再次输入密码:</font>
    </td>
    <td>
<!--输入确认密码->-->
        <asp:TextBox id=passwd2 TextMode="Password" maxlength=20 runat=server/>
```

```
        </td>
        <td>
    <!--输入确认密码->不能为空-->
          <asp:RequiredFieldValidator id="passwd2ReqVal"
               ControlToValidate="passwd2"
               ErrorMessage="Re-enter Password. "
               Display="Dynamic"
               Font-Name="Verdana" Font-Size="12"
               runat=server>
               *
          </asp:RequiredFieldValidator>
    <!--输入确认密码->两个密码是否相等-->
          <asp:CompareValidator id="CompareValidator1"
               ControlToValidate="passwd2" ControlToCompare="passwd"
               Display="Static"
               Font-Name="Arial" Font-Size="11"
               runat=server>
               密码不相等
          </asp:CompareValidator>
        </td>
      </tr>
   </table>
     <p>
          <input runat="server" type=submit value="提交">
      <p>
      <hr width=600 size=1 noshade>

   </form>
  </center>
</body>
</html>
```

运行结果如下：

我们不按照要求的输入，开到了下面的提示：

多行文本输入控件一般用来输入相关的内容，比如用户简短介绍、相关信息的补充等，一般情况下可以不用限制用户的输入。当然有些时候像留言板，我们不希望用户的输入内容中包含 HTML 的相关标记，这个时候我们就可以用上面的同样的方法来限制用户的输入，用法都是一样的，在此我们就不打算举个例子来说明了。

2.2.3 按钮控件

按钮控件的目的是使用户对页面的内容作出判断，当按下按钮后，页面会对用户的选择作出一定的反应，达到与用户交互的目的。

按钮控件的使用虽然很简单，但是按钮控件却是最常用的服务器控件之一，值得我们学习。对按钮控件的使用要注意它的 3 个事件和一个属性，即:

OnClick 事件，即用户按下按钮以后，即将触发的事件。我们通常在编程中，利用此事件，完成对用户选择的确认、对用户表单的提交、对用户输入数据的修改等等。

OnMouseOver 事件，当用户的光标进入按钮范围触发的事件。为了使页面有更生动的显示，我们可以利用此事件完成，诸如，当光标移入按钮范围时，使按钮发生某种显示上的改变，用以提示用户可以进行选择了。

OnMouseOut 事件，当用户光标脱离按钮范围触发的事件。同样，为使页面生动，当光标脱离按钮范围时，也可以发生某种改变，如恢复原状，用以提示用户脱离了按钮选择范围，若此时按下鼠标，将不是对按钮的操作。

Text 属性，按钮上显示的文字，用以提示用户进行何种选择。

例子：下例将显示 3 个按钮，分别演示 3 种事件的处理。

当按下第一个按钮时，根据 <asp:button id="btn1" text="OnClick 事件演示" Width=150px Onclick="btn1_Onclick" runat=server />的定义将调用 btn1_OnClick 过程，该过程的作用，即在按钮后显示 lbl1 控件的内容”OnClick 事件触发”

当移动光标到第二个按钮时，根据按钮定义 <asp:button id="btn2" text="OnMouseOver 事件演示" Width=150px OnMouseOver="this.style.backgroundColor='lightgreen'" OnMouseOut="this.style.backgroundColor='buttonface'" runat=server />，光标移动到按钮上时，按钮的背景色应该变为淡绿色。

当移动光标到第三个按钮时，根据其定义<asp:button id="btn3" text="OnMouseOut 事件演示" Width=150pxOnMouseOver="this.style.fontWeight='bold'" OnMouseOut = "this.style.fontWeight = 'normal'" runat=server />，按钮的字体变为黑体，但是我们要观察的是当又把光标移开后，第三个按钮是否恢复正常的字体。

1． 源程序(**FormButton.aspx**)

```
<!--源文件：form\ServerControl\formbutton.aspx-->
<!-- 文件名：FormButton.aspx -->
<html>
<script language="vb" runat=server>
sub btn1_OnClick(s as object,e as EventArgs)
lbl1.text="Onclick 事件触发"
end sub
```

```
</script>

 <head>
 <title>
  Button 控件的使用
 </title>
 </head>

<body bgcolor=#ccccff>
 <center>
  <h2>Button 控件三种事件的响应实验</h2>
  <hr>
 </center>
     <form runat=server>
          <asp:button id="btn1" text="OnClick 事件演示" Width=150px
     Onclick="btn1_Onclick" runat=server />
              
          <asp:label id="lbl1" runat=server/>
          <br>

          <asp:button id="btn2" text="OnMouseOver 事件演示" Width=150px
     OnMouseOver="this.style.backgroundColor='lightgreen'"
          OnMouseOut="this.style.backgroundColor='buttonface'"
          runat=server />

          <br>

          <asp:button id="btn3" text="OnMouseOut 事件演示" Width=150px
     OnMouseOver="this.style.fontWeight='bold'"
     OnMouseOut="this.style.fontWeight='normal'"
       runat=server />

     </form>

<body>
```

2．开始时的输出画面如下：

3．当按下“ OnClick 事件演示 ”按钮后，lbl1 显示“ OnClick 事件触发”：

4．当移动光标到“ OnMouseOver 事件演示“ 按钮时，该按钮背景色变为淡绿色：

5．当光标移动到“ OnMouseOut 事件演示“ 按钮时，该按钮的字体变为黑体，但是我们需观察的却是再移开光标后，字体是否恢复正常，结论是会的，这里只给出了移动到该按钮时的画面，移开后的画面由于和开始画面一样，就不演示了。

2.2.4 复选控件

在日常信息输入中，我们会遇到这样的情况，输入的信息只有两种可能性（例如：性别、婚否之类）,如果采用文本输入的话，一者输入繁琐，二者无法对输入信息的有效性进行控制，这时如果采用复选控件（CheckBox），就会大大减轻数据输入人员的负担，同时输入数据的规范性得到了保证。

CheckBox 的使用比较简单，主要使用 id 属性和 text 属性。Id 属性指定对复选控件实例的命名，Text 属性主要用于描述选择的条件。另外当复选控件被选择以后，通常根据其 Checked 属性是否为真来判断用户选择与否。

例如：使用复选控件

```
…
<asp:CheckBox id=”chkbox1” text=”中国人” runat=server/>
 …
 判断条件满足否：
…
If chkbox1.Checked=True
    LblTxt.text=”是中国人”
Else
     LblTxt.text=”不是中国人”
End If
…
```

2.2.5 单选控件

使用单选控件的情况跟使用复选控件的条件差不多，区别的地方在于，单选控件的选择可能性不一定是两种，只要是有限种可能性，并且只能从中选择一种结果，在原则上都可以用单选控件（RadioButton）来实现。

单选控件主要的属性跟复选控件也类似，也有 id 属性、text 属性，同样也依靠 Checked 属性来判断是否选中，但是与多个复选控件之间互不相关不同，多个单选控件之间存在着联系，要么是同一选择中的条件，要么不是。所以单选控件多了一个 GroupName 属性，它用来指明多个单选控件是否是同一条件下的选择项，GroupName 相同的多个单选控件之间只能有一个被选中。

例如：对单选控件的使用

```
…
    年龄选择：
    <asp:RadioButton id=”rbtn11” text=”20 岁以下” GroupName=”group1” runat=server /><br>
    <asp:RadioButton id=”rbtn12” text=”20-30 岁” GroupName=”group1” runat=server /><br>
    <asp:RadioButton id=”rbtn13” text=”30-40 岁” GroupName=”group1” runat=server /><br>
    <asp:RadioButton id=”rbtn14” text=”40 岁以上” GroupName=”group1” runat=server /><br>
    工作收入：
    <asp:RadioButton id=”rbtn21” text=”1000 元以下” GroupName=”group2” runat=server
/><br>
```

<asp:RadioButton id=”rbtn22” text=”1000-2000 元 ” GroupName=”group2” runat=server/>

<asp:RadioButton id=”rbtn23” text=”2000 元以上” GroupName=”group2” runat=server />

…

对选择条件的判断：

```
…
If    rbtn11.Checked = True
      LblResult1.text=”20 岁以下”
Else if rbtn12.Checked = True
      LblResult1.text=”20-30 岁”
Else if rbtn13.Checked = True
      LblResult1.text=”30-40 岁”
Else
      LblResult1.text=”40 岁以上”
End If

If rbtn21.Checked = True
      LblResult2.text=”1000 元以下”
Else if rbtn22.Checked = True
      LblResult2.text=”1000-2000 元”
Else
      LblResult.text=”2000 元以上”
End If
…
```

2.2.6 列表框

列表框（ListBox）是在一个文本框内提供多个选项供用户选择的控件，它比较类似于下拉列表，但是没有显示结果的文本框。到以后，我们会知道列表框实际上很少使用，大部分时候，我们都使用列表控件 DropDownList 来代替 ListBox 加文本框的情况，在这里对列表框的讨论，主要是为以后的应用学习一些简单的控件属性。

列表框的属性 SelectionMode，选择方式主要是决定控件是否允许多项选择。当其值为 ListSelectionMode.Single 时，表明只允许用户从列表框中选择一个选项；当值为 List.SelectionMode.Multi 时，用户可以用 Ctrl 键或者是 Shift 键结合鼠标，从列表框中选择多个选项。

属性 DataSource，说明数据的来源，可以为数组、列表、数据表。

方法 DataBind，把来自数据源的数据载入列表框的 items 集合。

例子：在这里我们将结合前面学习的按钮控件（Button）、复选控件（CheckBox）、单选控件（RadioButton）以及列表框（ListBox）给出一个实例。

首先，页面加载时，我们产生一个数组 Values，并添加 4 个关于水果的测试数据到 Valuse 数组。然后列表框从数组取得数据加载进自己的 items 集合。然后根据复选控件 chkBold 的状态决定是否用黑体字输出列表框的选择结果。根据单选控件 rbtnMulti 和 rbtnSingle 的状态

决定下一次列表框是否允许多选，最后按下按钮控件提交表单，显示选择的结果。

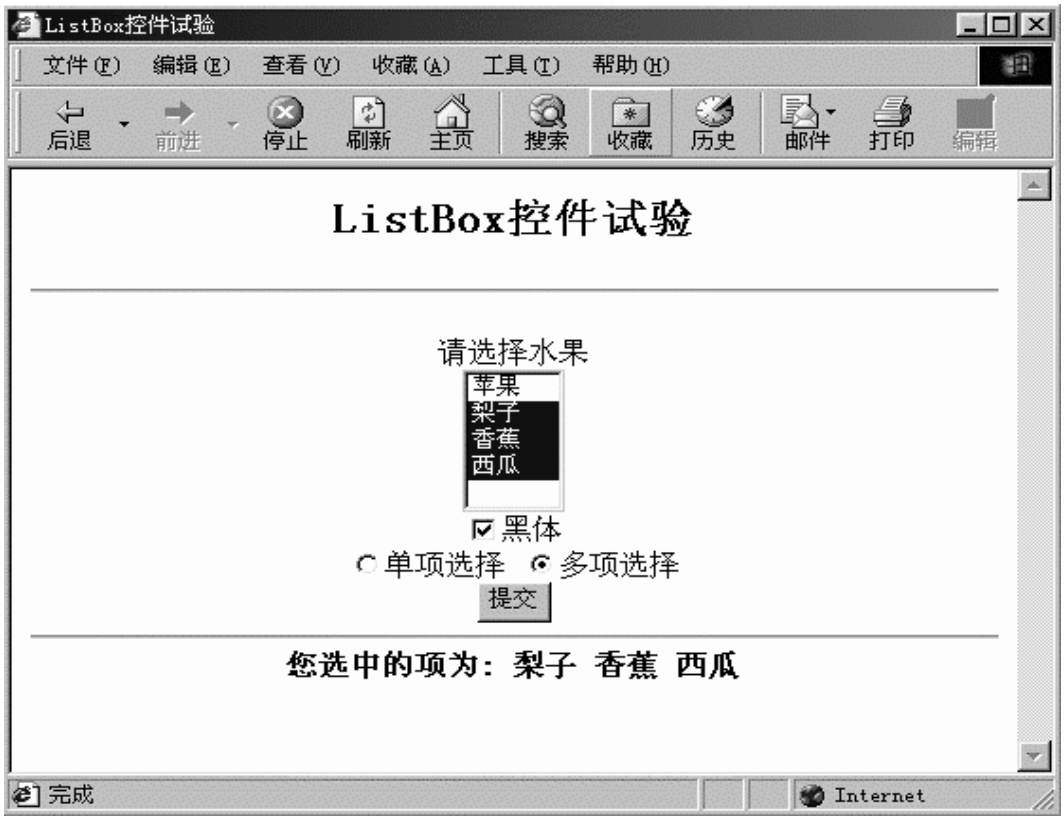
1．源程序(FormListBox.aspx)

```
<!--源文件：form\ServerControl\formlistbox.aspx-->
<html>
 <head>
 <title>
 ListBox 控件试验
 </title>
 </head>

  <script language="VB" runat=server>
  Sub Page_Load()

  '如果选中黑体复选控件，把文本标签的字体设为黑体
  If chkBold.Checked
        lblTxt.font.bold=True
  Else
        lblTxt.font.bold=False
  End If

  '如果选中多选的单选控件，那么则把列表框设为允许多选
  If   rbtnMulti.Checked
    list1.SelectionMode=ListSelectionMode.Multiple
  Else
    list1.SelectionMode=ListSelectionMode.Single
  End If

  If Not IsPostBack
  '第一次请求时，为列表框设置数据
  Dim values as ArrayList=new ArrayList()

        values.add("苹果")
        values.add("梨子")
        values.add("香蕉")
        values.add("西瓜")
        list1.datasource=values
        list1.databind
  Else
  '把从列表框选中的内容赋予文本标识，如果未选择，显示"未选择"
  Dim i as int32
  Dim tmpStr as String

  ‘对列表框 list1 的 items 集合轮询，根据其 Selected 属性，判断是否被选中
```

```
    For i=0 to list1.items.count-1
    If list1.items(i).selected
    tmpStr=tmpStr & " " & list1.items(i).text
    End If
    Next

    If tmpStr is Nothing
    tmpStr="未选择"
    End If

    lblTxt.text="您选中的项为: " & tmpStr

    End If

End Sub
    </script>

     <body >
        <center>
         <h2>ListBox 控件试验</h2>
         <hr>
         <form method="POST" runat=server>
            请选择水果
            <br>
          <asp:ListBox id="list1" runat=server/>
            <br>
            <asp:CheckBox id="chkBold" text="黑体" runat=server />
            <br>
            ‘第一次设置为单项选择
            <asp:RadioButton id="rbtnSingle" Checked=True text="单项选择"
                                              groupname="group1" runat=server />
            <asp:RadioButton id="rbtnMulti" text="多项选择"
                                          groupname="group1" runat=server />
            <br>
      <asp:button text="提交" runat=server />
            <hr>
      <asp:label id="lblTxt" runat=server />
         </form>
      </center>
     </body>

</html>
```

2．开始时的画面输出，第一次缺省设置为单项选择

3．选择以黑体字输出，并且允许多项选择后的画面：



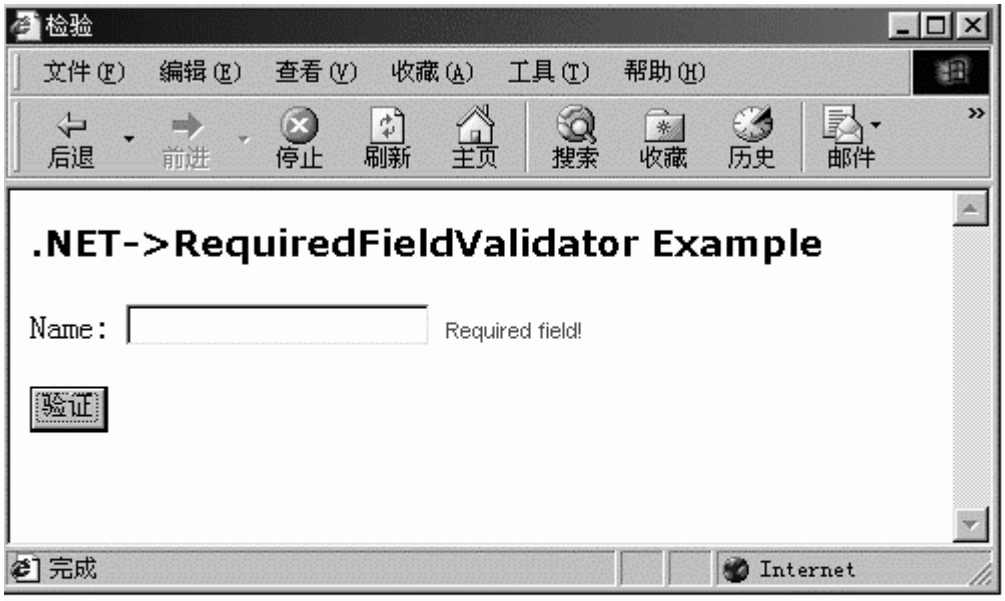
2.2.7 RequiredFieldValidator

这个 RequiredFieldValidator 服务器控件保证用户不会跳过一个入口，如果用户输入的值跟符合 RequiredFieldValidator 的要求，这个值就是有效的；否则，不会跳过这一输入步骤而往下走。

下面的例子(RequiredFieldValidator.aspx)验证了这个要求：

```
<!--源文件：form\ServerControl\requiredfieldvalidator.aspx-->
<html>
<title>检验</title>
    <h3><font face="Verdana">.NET->RequiredFieldValidator Example</font></h3>
    <form runat=server>
        Name: <asp:TextBox id=Text1 runat="server"/>
        <asp:RequiredFieldValidator id="RequiredFieldValidator1" ControlToValidate="Text1"

Font-Name="Arial" Font-Size="11" runat="server">
            Required field!
        </asp:RequiredFieldValidator>
        <p>
        <asp:Button id="Button1" runat="server" Text="验证" />
    </form>
</body>
</html>
```

运行效果如下：



2.2.8 ValidationSummary

用户的输入有时候是按照一定的顺序来的，有效性控件验证用户的输入并设置一个属性来线使用户的输入是否通过了验证。当所用得验证项都被处理之后，页面的 IsValid 属性就被设置，当有其中的一个验证没有通过时，整个页面将会不被通过验证。

当页面的 IsValid 属性为 false 时，ValidationSummary 属性将会表现出来。他获得页面上的每个确认控件并对每个错误提出修改信息。

文件 **Summary.aspx** 的内容：

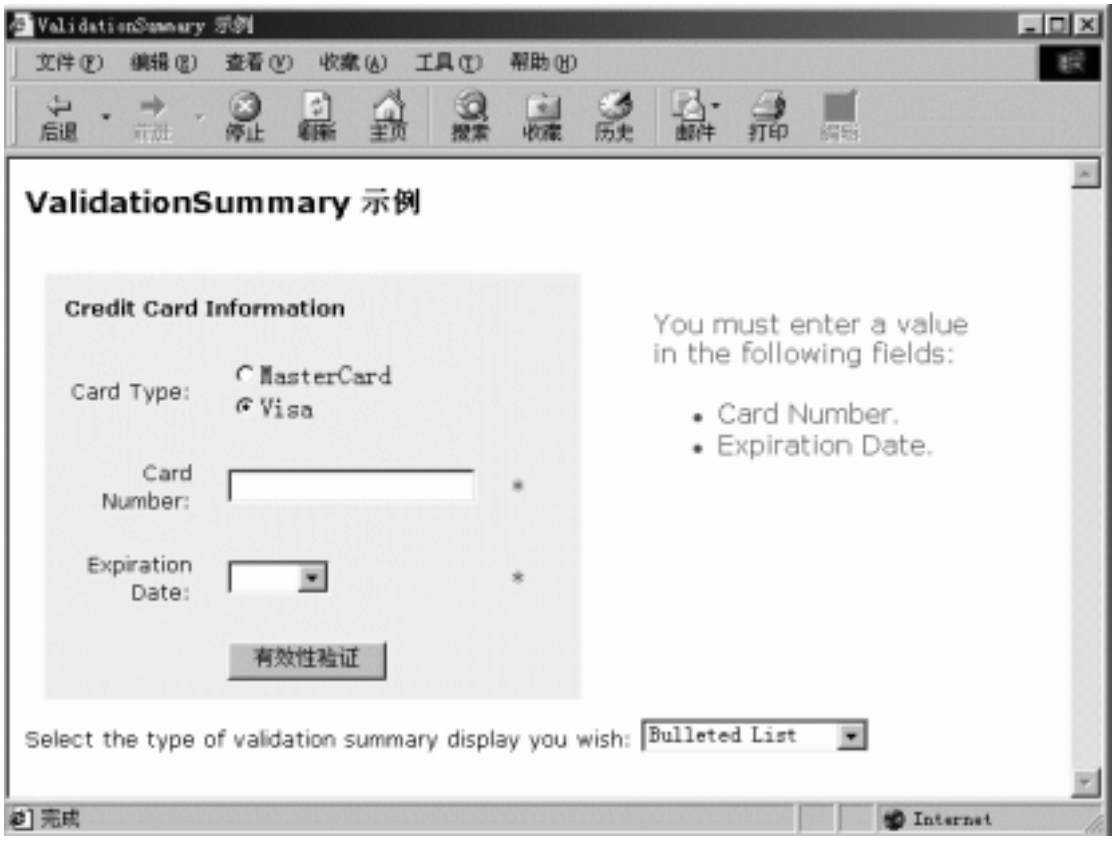
```
<!--源文件：form\ServerControl\summary.aspx-->
<%@ Page clienttarget=downlevel %>
<html>
<head>
    <script language="VB" runat="server">

        Sub ListFormat_SelectedIndexChanged(sender As Object, e As EventArgs)

            ' Change display mode of the validator summary when a new option
            ' is selected from the "ListFormat" dropdownlist
            valSum.DisplayMode = ListFormat.SelectedIndex

        End Sub

    </script>

</head>
<BODY>
<h3><font face="Verdana">ValidationSummary Sample</font></h3>
<p>

<form runat="server">
<table cellpadding=10>
    <tr>
        <td>
            <table bgcolor="#eeeeee" cellpadding=10>
            <tr>
              <td colspan=3>
              <font face=Verdana size=2><b>Credit Card Information</b></font>
              </td>
            </tr>
            <tr>
              <td align=right>
                <font face=Verdana size=2>Card Type:</font>
              </td>
```

```
    <td>
      <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow"
            runat=server>
        <asp:ListItem>MasterCard</asp:ListItem>
        <asp:ListItem>Visa</asp:ListItem>
      </ASP:RadioButtonList>
    </td>
    <td align=middle rowspan=1>
      <asp:RequiredFieldValidator id="RequiredFieldValidator1"
        ControlToValidate="RadioButtonList1"
        ErrorMessage="Card Type. "
        Display="Static"
        InitialValue="" Width="100%" runat=server>
        *
      </asp:RequiredFieldValidator>
    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Verdana size=2>Card Number:</font>
    </td>
    <td>
      <ASP:TextBox id=TextBox1 runat=server />
    </td>
    <td>
      <asp:RequiredFieldValidator id="RequiredFieldValidator2"
        ControlToValidate="TextBox1"
        ErrorMessage="Card Number. "
        Display="Static"
        Width="100%" runat=server>
        *
      </asp:RequiredFieldValidator>

    </td>
  </tr>
  <tr>
    <td align=right>
      <font face=Verdana size=2>Expiration Date:</font>
    </td>
    <td>
      <ASP:DropDownList id=DropDownList1 runat=server>
        <asp:ListItem></asp:ListItem>
        <asp:ListItem >06/00</asp:ListItem>
        <asp:ListItem >07/00</asp:ListItem>
```

```
                    <asp:ListItem >08/00</asp:ListItem>
                    <asp:ListItem >09/00</asp:ListItem>
                    <asp:ListItem >10/00</asp:ListItem>
                    <asp:ListItem >11/00</asp:ListItem>
                    <asp:ListItem >01/01</asp:ListItem>
                    <asp:ListItem >02/01</asp:ListItem>
                    <asp:ListItem >03/01</asp:ListItem>
                    <asp:ListItem >04/01</asp:ListItem>
                    <asp:ListItem >05/01</asp:ListItem>
                    <asp:ListItem >06/01</asp:ListItem>
                    <asp:ListItem >07/01</asp:ListItem>
                    <asp:ListItem >08/01</asp:ListItem>
                    <asp:ListItem >09/01</asp:ListItem>
                    <asp:ListItem >10/01</asp:ListItem>
                    <asp:ListItem >11/01</asp:ListItem>
                    <asp:ListItem >12/01</asp:ListItem>
                </ASP:DropDownList>
            </td>
            <td>
                <asp:RequiredFieldValidator id="RequiredFieldValidator3"
                    ControlToValidate="DropDownList1"
                    ErrorMessage="Expiration Date. "
                    Display="Static"
                    InitialValue=""
                    Width="100%"
                    runat=server>
                    *
                </asp:RequiredFieldValidator>
            </td>
            <td>
        </tr>
        <tr>
            <td></td>
            <td>
                <ASP:Button id=Button1 text="有效性验证" runat=server />
            </td>
            <td></td>
        </tr>
        </table>
    </td>
    <td valign=top>
        <table cellpadding=20><tr><td>
        <asp:ValidationSummary ID="valSum" runat="server"
            HeaderText="You must enter a value in the following fields:"
```

```
                    Font-Name="verdana"
                    Font-Size="12"
                    />
                </td></tr></table>
            </td>
        </tr>
</table>

<font face="verdana" size="-1">Select the type of validation summary display you wish: </font>

<asp:DropDownList id="ListFormat" AutoPostBack=true

OnSelectedIndexChanged="ListFormat_SelectedIndexChanged" runat=server >
    <asp:ListItem>List</asp:ListItem>
    <asp:ListItem selected>Bulleted List</asp:ListItem>
    <asp:ListItem>Single Paragraph</asp:ListItem>
</asp:DropDownList>

</form>

</body>
</html>
```

结果如下：



2.2.9 使用 panel 控件

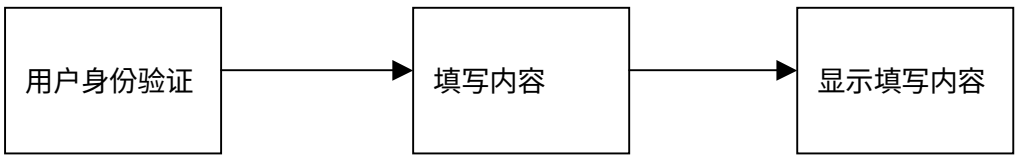
我们在进行用户信息录入的时候，如果使用 ASP 程序写的话，通常需要三个网页：

① 进行用户身份检查

② 填写相关的内容

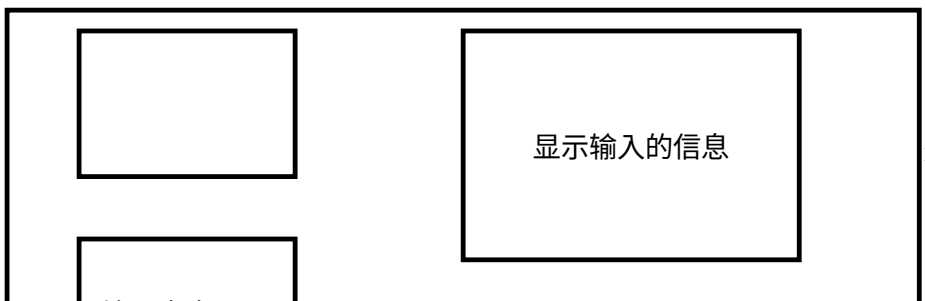
③ 显示你填写的内容

填写的流程如下：



这样的话，我们将分别设计 3 个网页。这样会显得很麻烦。可喜的是，我们可以使用 asp.net 中的 panel 控件，在一个页面中即可实现上述的功能。

流程如下：



好了，有了上面的叙述，请看 **panel.aspx** 文件内容：

```
<!--源文件：form\ServerControl\panel.aspx-->
 <Html>
 <Body bgcolor="White">
 <center><H3>使用 Panel 控件示例<Hr></H3></center>
<title>使用 Panel 控件示例</title>
<script Language="VB" runat="server">
 Sub Page_Load(sender As Object, e As EventArgs)
      If Not Page.IsPostBack Then
          panel2.Visible = False
          panel3.Visible = False
      End If
 End Sub
Sub Button1_Click(sender As Object, e As EventArgs)
      panel1.Visible = False
      panel2.Visible = True
 End Sub

 Sub Button2_Click(sender As Object, e As EventArgs)
      panel2.Visible = False
      panel3.Visible = True
      Span1.InnerHtml    = "用户名: " & UserID.Text & "<BR>"
      Span1.InnerHtml &= "密码: " & Password.Text & "<BR>"
      Span1.InnerHtml &= "姓名: " & Name.Text & "<BR>"
      Span1.InnerHtml &= "电话: " & Tel.Text & "<BR>"
      Span1.InnerHtml &= "E-mail: " & mail.Text & "<BR>"
      Span1.InnerHtml &= "地址: " & Addr.Text & "<P>"

 End Sub
 Sub Button3_Click(sender As Object, e As EventArgs)
      Span1.InnerHtml &= "输入完成!"
      Button3.Visible = False
 End Sub
 </script>
```

```
<Form runat="server">
<center>
<asp:Panel id="panel1" runat="server">
 <Font Color="#800000"><B>第一步：请输入用户名和密码</B></Font><Blockquote>
        用户名：<asp:TextBox id="UserID" runat="server" Text="kjwang"/><p>
          密码：<asp:TextBox id="Password" TextMode="Password"
                Text="kj6688" runat="server"/><p>
         <Input Type="Button" id="Button1" value="  登录  "
                 OnServerClick="Button1_Click" runat="server">
 </Blockquote>
 </asp:Panel>
 <asp:Panel id="panel2" runat="server">
 <Font Color="#800000"><B>第二步：请输入用户信息</B></Font><Blockquote>
         姓名：<asp:TextBox id="Name" runat="server" Text="小李"/><p>
         电话：<asp:TextBox id="Tel" runat="server" Text="(023)65355678" /><p>
         E-mail：<asp:TextBox id="mail" runat="server" Text="jimmy.zh@263.net" /><p>
         地址：<asp:TextBox  id="Addr"  runat="server"  Text="重庆市人民路 115#"  Size="40"
/><p>
         <Input Type="Button" id="Button2" value="申请"
                 OnServerClick="Button2_Click" runat="server">
 </Blockquote>
 </asp:Panel>
  <asp:Panel id="panel3" runat="server">
 <Font Color="#800000"><B>第三步：请确认你的输入</B></Font><Blockquote>
     <Span id="Span1" runat="server"/>
     <Input Type="Button" id="Button3" value="  确认  "
             OnServerClick="Button3_Click" runat="server">
 </Blockquote>
 </asp:Panel>
</center>
 </form>
 <Hr></body>
 </html>
```

请看程序的运行效果：

第一步：


使用Panel控件示例
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 前进 停止 刷新 主页 搜索 收藏 历史 邮件 打印
地址(D) http://213.213.213.126/quickstart/aspplus/samples/webforms/book/liu/panel.aspx
使用Panel控件示例
第一步：请输入用户名和密码
用户名：zy
密码：******
登录
完成
Internet

第二步：


使用Panel控件示例
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 前进 停止 刷新 主页 搜索 收藏 历史 邮件 打印
地址(D) http://213.213.213.126/quickstart/aspplus/samples/webforms/book/liu/panel.aspx
使用Panel控件示例
第二步：请输入用户信息
姓名：小李
电话：(023)65355678
E-mail：jimmy.zh@263.net
地址：重庆市人民路115#
申请
完成
Internet

第三步：

大家可以留意浏览器的地址栏，我们注意到地址都是相同的。我就是我们使用 pann 控件所得到的效果。

2.2.10 选择控件

选择的方式有两种 :单选、多选。我们下面用具体的例子来说明这两种选择控件在 asp.net 上的实现。

对单选控件，asp.net 里面有一个专用的表示：RadioButtonList，我们看下面的代码：

```
<!--列出选择内容-->
<ASP:RadioButtonList id=ccType Font-Name="Arial" RepeatLayout="Flow"
                     runat=server>
<asp:ListItem>招行一卡通</asp:ListItem>
<asp:ListItem>牡丹卡</asp:ListItem>
</ASP:RadioButtonList>
```

我们在取值的时候，就可以用一个语句：

```
Request.QueryString(“ccType”)
```

来取得这个值。下面是我们这个说明的完整代码（**RadioButtonList.aspx**）：

```
<!--源文件：form\ServerControl\RadioButtonList.aspx-->
<html>
<body >
<center>
<br><br>
    <h3><font face="Verdana">.NET->选择控件!</font></h3>
<br><br><br>
</center>
    <form method=post runat=server>
    <hr width=600 size=1 noshade>

    <center>
    <asp:ValidationSummary ID="valSum" runat="server"
        HeaderText="你必须填写下面的内容:"
        DisplayMode="SingleParagraph"
        Font-Name="verdana"
        Font-Size="12"
        />
    <p>
    <!-- 信用卡信息 -->
    <table border=0 width=600>
    <tr>
      <td colspan=3>
      <center>
      <font face=Arial size=2><b>信用卡信息</b></font>
      </center>
      </td>
    </tr>
    <tr>
      <td align=right>
        <font face=Arial size=2>信用卡类型:</font>
      </td>
      <td>
    <!--列出选择内容-->
        <ASP:RadioButtonList id=ccType Font-Name="Arial" RepeatLayout="Flow"
runat=server>
            <asp:ListItem>招行一卡通</asp:ListItem>
            <asp:ListItem>牡丹卡</asp:ListItem>
        </ASP:RadioButtonList>
      </td>
      <td>
        <asp:RequiredFieldValidator id="ccTypeReqVal"
            ControlToValidate="ccType"
```

```
                ErrorMessage="信用卡类型. "
                Display="Static"
                InitialValue=""
                Font-Name="Verdana" Font-Size="12"
                runat=server>
                *
            </asp:RequiredFieldValidator>
        </td>
    </tr>
    </table>
    <p>
    <input runat="server" type=submit value="提交">
    <p>
    <hr width=600 size=1 noshade>
</form>
</center>
</body>
</html>
```

我们的运行效果如下：

我们选择一个并提交，则会提交成功；反之，如果我们没有选择就提交，会出现如下的信息：

我们再来看看多选的情况：

```
'选择项列表
<asp:CheckBoxList id=Check1 runat="server">
    <asp:ListItem>北京</asp:ListItem>
    <asp:ListItem>深圳</asp:ListItem>
    <asp:ListItem>上海</asp:ListItem>
    <asp:ListItem>广州</asp:ListItem>
    <asp:ListItem>南宁</asp:ListItem>
    <asp:ListItem>重庆</asp:ListItem>
</asp:CheckBoxList>
```

跟我们上面的单选控件就相差在定义上，我们用 CheckBoxList 来描述我们的选择框，我们写一个方法来响应我们的“ 提交“ 事件：

```
'响应按钮事件
Sub Button1_Click(sender As Object, e As EventArgs)
    Dim s As String = "被选择的选项是:<br>"
    Dim i As Int32
    For i = 0 to Check1.Items.Count-1
```

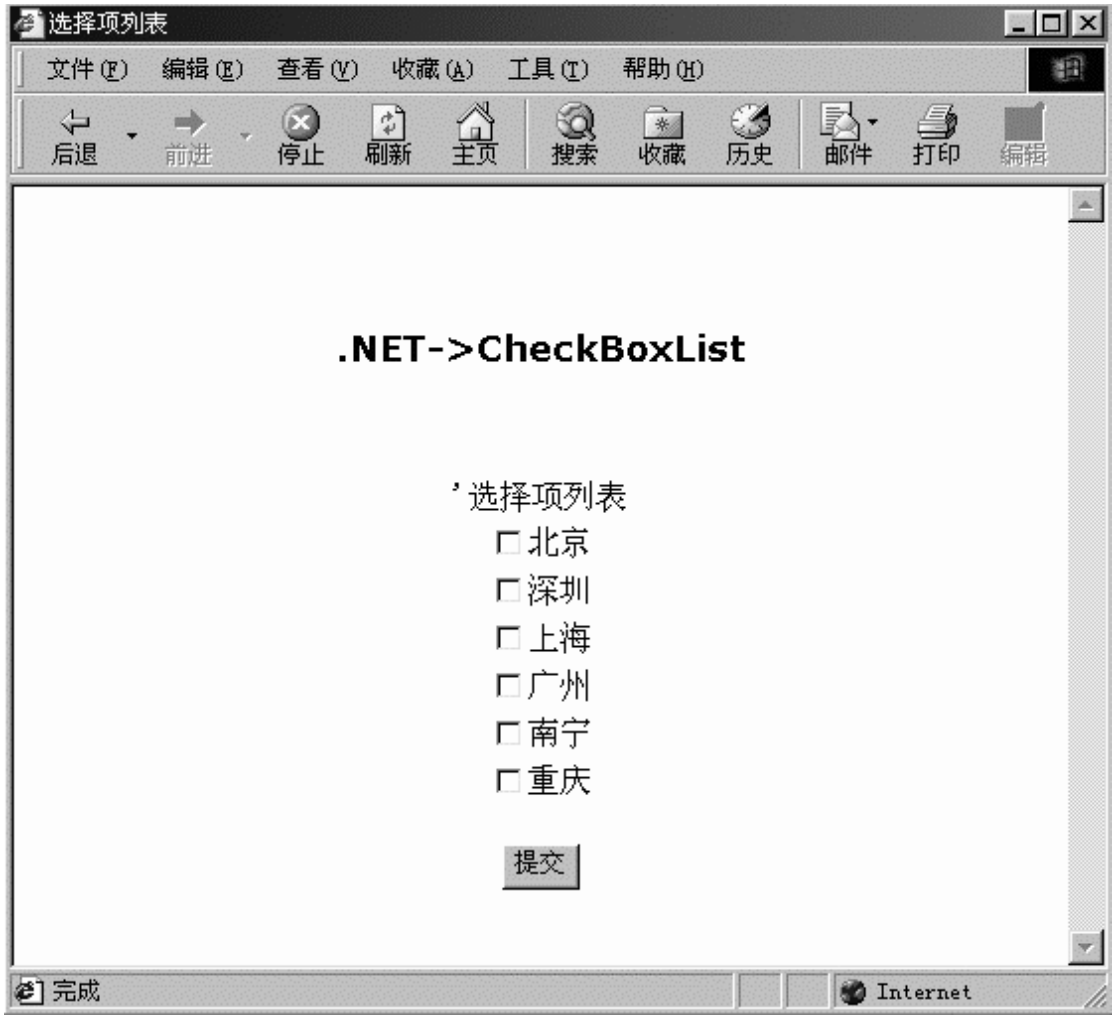
```
                If Check1.Items(i).Selected Then
                    '列出选择项
                    s = s & Check1.Items(i).Text
                    s = s & "<br>"
                End If
            Next
            '打印出选择项
            Label1.Text = s
        End Sub
```

这个方法为定义打印被选择的选项。具体的内容如下(**list.aspx**)：

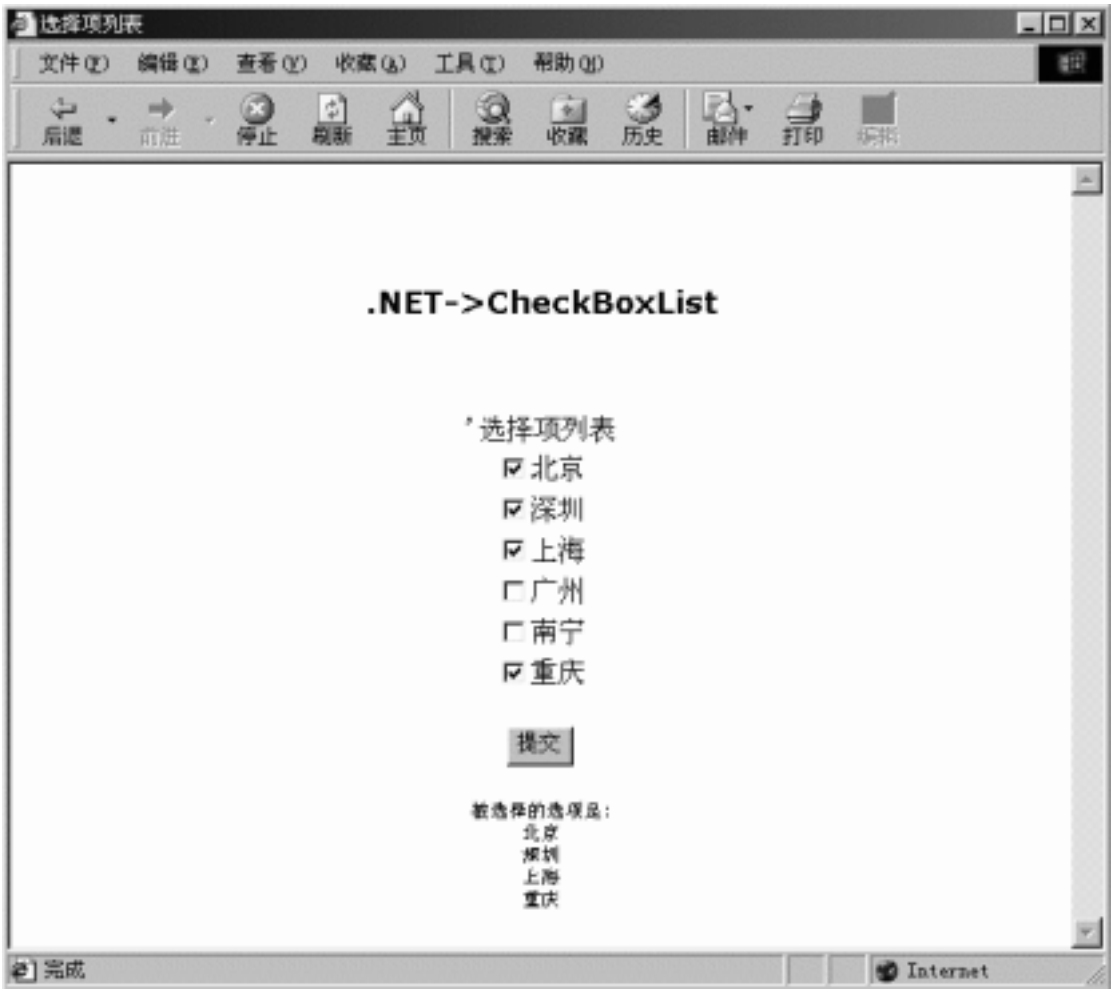
```
<!--源文件：form\ServerControl\list.aspx-->
    <html>
    <head>
    <script language="VB" runat="server">
      '响应按钮事件
      Sub Button1_Click(sender As Object, e As EventArgs)
          Dim s As String = "被选择的选项是:<br>"
          Dim i As Int32
          For i = 0 to Check1.Items.Count-1
              If Check1.Items(i).Selected Then
                  '列出选择项
                  s = s & Check1.Items(i).Text
                  s = s & "<br>"
              End If
          Next
          '打印出选择项
          Label1.Text = s
      End Sub
    </script>
</head>
<body bgcolor="#ccccff">
<br><br><br>
<center>
    <h3><font face="Verdana">.NET->CheckBoxList</font></h3>
</center>
<br><br>
<center>
    <form runat=server>
    '选择象列表
        <asp:CheckBoxList id=Check1 runat="server">
            <asp:ListItem>北京</asp:ListItem>
            <asp:ListItem>深圳</asp:ListItem>
            <asp:ListItem>上海</asp:ListItem>
            <asp:ListItem>广州</asp:ListItem>
```

```
            <asp:ListItem>南宁</asp:ListItem>
            <asp:ListItem>重庆</asp:ListItem>
        </asp:CheckBoxList>
        <p>
            <asp:Button id=Button1 Text="提交" onclick="Button1_Click" runat="server"/>
        <p>
            <asp:Label id=Label1 font-name="Verdana" font-size="8pt" runat="server"/>
    </form>
</center>
</body>
</html>
```

我们看到显示如下：



选择几个选项，并点击“ 提交“ 按钮“ ，看到如下的情况：



2.2.11 ImageButton 控件

ImageButton 控件

当我们在浏览网页的时候，可能会发现这样一种情况：当鼠标移到图象按钮上和当鼠标移走的时候，将会发现同一按钮上将会显示不同的两个图片。我们可以用 Image Button 控件的 onmouseout 和 onmouseover 事件来实现。

请看 **Image.aspx** 中的代码：

```
<!--源文件：form\ServerControl\image.aspx-->
<html>
<Body BgColor="White">
<center><H3>ImageButton 控件演示</H3></center>
<title>ImageButton 控件演示</title>
<script Language="VB" runat="server">
Sub Button1_Click(sender As Object, e As ImageClickEventArgs)
```

```
'定义当我们点击按钮的时候，将访问的网页
Page.Navigate( "http://www.yesky.com" )
End Sub
</script>
<Form runat="server">
<center>
<asp:ImageButton OnClick="Button1_Click"
ImageUrl="18.gif" id="Button1" runat="server"
OnMouseOut="this.src='18.gif';"
OnMouseOver="this.src='19.gif';" />
</center>
</Form>
<asp:Label id="Label1" runat="server"/>
</Body>
</Html>
```

在这段程序中，我们使用了 onmouseout 和 onmouseover 事件。
请看程序的演示效果：

当鼠标移动到按钮上的时候，将显示：

2.2.12 列表控件

在 asp.net 中，有几种方法可以应用于列表控件。我们可以在 aspx 代码中直接嵌入相关的代码，也可以在页面装入的时候加载这些列表信息。

下面是具体的应用，我们先看看在 aspx 中的列表方法：

```
<!--列表->列出内容-->
    <asp:DropDownList id=DropDown1 runat="server">
        <asp:ListItem>北京</asp:ListItem>
        <asp:ListItem>深圳</asp:ListItem>
        <asp:ListItem>上海</asp:ListItem>
        <asp:ListItem>广州</asp:ListItem>
        <asp:ListItem>南宁</asp:ListItem>
        <asp:ListItem>重庆</asp:ListItem>
    </asp:DropDownList>
```

我们在需要取出所选的数据时，直接去取 id 值，即 DropDown1；我们再定一一个方法，响应“ 提交“ 按钮的事件，就可以了。下面是完整的代码(**DropDown.aspx**)：

```
<!--源文件：form\ServerControl\dropdown.aspx-->
<html>
<head>

    <script language="VB" runat="server">
```

```
    '在点击按钮时候响应
        Sub list_Click(sender As Object, e As EventArgs)
            Label1.Text="你的选择是: " + DropDown1.SelectedItem.Text
        End Sub

    </script>

</head>
<body bgcolor="#ccccff">
<br><br><br>
<center>
    <h3><font face="Verdana">.NET->列表控件</font></h3>
</center>
<br><br>
<center>
    <form runat=server>
    <!--列表->列出内容-->
        <asp:DropDownList id=DropDown1 runat="server">
            <asp:ListItem>北京</asp:ListItem>
            <asp:ListItem>深圳</asp:ListItem>
            <asp:ListItem>上海</asp:ListItem>
            <asp:ListItem>广州</asp:ListItem>
            <asp:ListItem>南宁</asp:ListItem>
            <asp:ListItem>重庆</asp:ListItem>
        </asp:DropDownList>

        <asp:button text="提交" OnClick="list_Click" runat=server/>

        <p>

        <asp:Label id=Label1 font-name="Verdana" font-size="10pt" runat="server">

        </asp:Label>

    </form>
</center>
</body>
</html>
```

我们看看运行效果：

点击提交按钮时候看到下面的效果：

下面我们再来看看另外一个列表控件的使用，我们定义一个在页面装载的时候调用的方法：

```
'再页面装载的时候调用的方法：
    Sub Page_Load(sender As Object, e As EventArgs)
        If Not IsPostBack Then
            Dim values as ArrayList= new ArrayList()
            values.Add ("北京")
            values.Add ("深圳")
            values.Add ("上海")
            values.Add ("广州")
            values.Add ("南宁")
            values.Add ("重庆")
            '设定 DropDown1 的数据源为 values，即上面定义的信息
            DropDown1.DataSource = values
            '数据的绑定
            DropDown1.DataBind
        End If
    End Sub
```

我们在 aspx 代码中调用它：

```
<!--列出列表信息-->
<asp:DropDownList id="DropDown1" runat="server" />
```

就这样的一个简单的语句就可以了，下面是这个文件的完整的代码：

```
<html>
<head>

<script language="VB" runat="server">
'再页面装载的时候调用的方法：
    Sub Page_Load(sender As Object, e As EventArgs)

        If Not IsPostBack Then
            Dim values as ArrayList= new ArrayList()
            values.Add ("北京")
            values.Add ("深圳")
            values.Add ("上海")
            values.Add ("广州")
            values.Add ("南宁")
            values.Add ("重庆")
        '设定 DropDown1 的数据源为 values，即上面定义的信息
```

```
                DropDown1.DataSource = values
            '数据的绑定
                DropDown1.DataBind
             End If
        End Sub

    '提交按钮响应的方法
        Sub select02_Click(sender As Object, e As EventArgs)
            Label1.Text = "你的选择是: " + DropDown1.SelectedItem.Text
        End Sub

    </script>

</head>
<body BGCOLOR="#CCCCFF">

<br><br><br>
<center>
    <h3><font face="Verdana">.NET->列表控件</font></h3>
</center>
<br><br>
<center>
    <form runat=server>

    <!--列出列表信息-->
        <asp:DropDownList id="DropDown1" runat="server" />

        <asp:button Text="提交" OnClick="select02_Click" runat=server/>
        <p>
        <asp:Label id=Label1 font-name="Verdana" font-size="10pt" runat="server" />

    </form>
</center>
</body>
</html>
```

运行的结果跟上面的一样。

2.2.13 重复列表 Repeator

这种服务器控件会以给定的形式重复显示数据项目，故称之为重复列表。使用重复列表有两个要素，即数据的来源和数据的表现形式。数据来源的指定由控件的 DataSource 属性决定，并调用方法 DataBind 绑定到控件上。这里需要说明的是数据取出以后如何表现的问

题，即如何布局。重复列表的数据布局是由给定的模板来决定的，由于重复列表没有缺省的模板，所以使用重复列表时至少要定义一个最基本的模板“ ItemTemplate”。

重复列表支持以下模板标识，所谓模板就是预先定义的一种表现形式，以后我们还会就这个问题专门讨论，这里就不在多说。

1） ItemTemplate 模板，数据项模板，必需的，它定义了数据项极其表现形式。
2） AlternatingItemTemplate 模板，数据项交替模板，为了使相邻的数据项能够有所区别，可以定义交替模板，它使得相邻的数据项看起来明显不同，缺省情况下，它和 ItemTemplate 模板定义一致，即缺省下相邻数据项无表示区分。
3） SeparatorTemplate 模板，分割符模板，定义数据项之间的分割符。
4） HeaderTemplate 模板，报头定义模板，定义重复列表的表头表现形式。
5） FooterTemplate 模板，表尾定义模板，定义重复列表的列表尾部的表现形式。

切记，由于缺乏内置的预定义模板和风格，在使用重复列表时，请一定记住要使用 HTML 格式定义自己的模板。

下面给出一个例子，看它是如何使用重复列表控件的。下面的例子首先在页面加载过程时把数据装载，并绑定到两个重复列表上；然后以一个 2 列的表格显示；最后把所有数据显示到一行上面，并且国家和领导人之间以 3 个中横线分隔，每一国家之间以竖划线分隔。

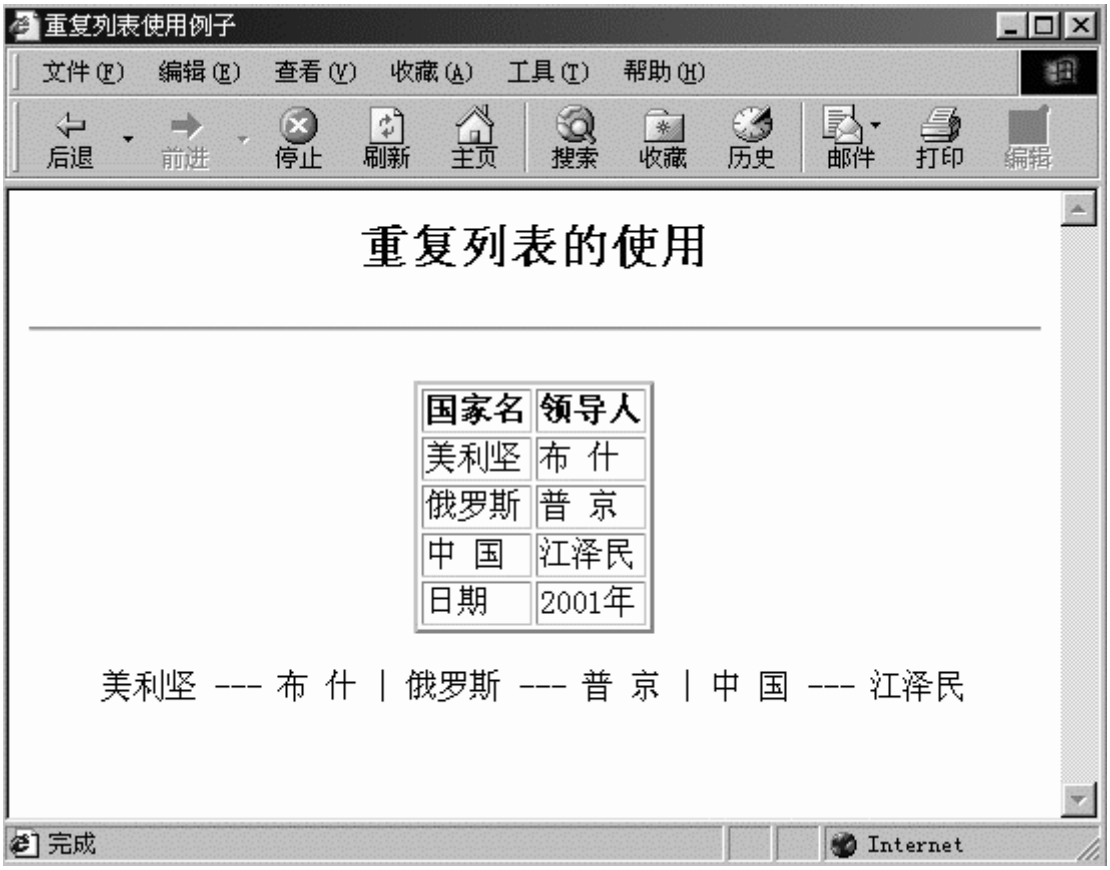
1． 源代码(**FormRepeater.aspx**)

```
<!--源文件：form\ServerControl\FormRepeater.aspx-->
<html>
  <head>
    <script language="vb" runat=server>

        Class Leader
          '定义一个类 Leader
        dim strCountry as String
        dim strName as String

        Public Sub New(country As String, name As String)
                    MyBase.New
                    strName = name
                    strCountry= country
                End Sub

                ReadOnly Property Name As String
                  Get
                    Return strName
                  End Get
                End Property

                ReadOnly Property Country As String
                  Get
                    Return strCountry
                  End Get
```

```
            End Property

        End Class

    sub Page_Load(s as object,e as eventargs)
      dim leaders as ArrayList = New ArrayList()
      if Not Page.IsPostBack
      '加载数据
          leaders.add(new leader("美利坚","布　什"))
          leaders.add(new leader("俄罗斯","普　京"))
          leaders.add(new leader("中　国","江泽民"))

          Repeater1.DataSource=leaders
          Repeater2.DataSource=leaders
          Repeater1.DataBind
          Repeater2.DataBind
      end if
    end sub
  </script>
  <title>
    重复列表使用例子
  </title>
</head>

  <center>
   <h2>重复列表的使用</h2>
   <hr>
   <br>
    '以表格形式显示国家，领导人信息
   <asp:Repeater id="Repeater1" runat=server>
        '定义表头
   <template name=HeaderTemplate>
        <table border=2>
          <tr>
            <th>
            国家名
            </th>
            <th>
            领导人
            </th>
          </tr>
      </template>

       '定义数据显示格式
```

```
            <template name=ItemTemplate>
               <tr>
                  <td>
                  <%# DataBinder.Eval(Container.DataItem,"Country") %>
                  </td>
                  <td>
                  <%# DataBinder.Eval(Container.DataItem,"Name") %>
                  </td>
               </tr>
            </template>

             '定义表尾
            <template name=FooterTemplate>
               <tr>
                 <td>日期</td>
                 <td>2001 年</td>
               </tr>
               </table>
            </template>
    </asp:Repeater>

    <br>
    <asp:Repeater id=Repeater2 runat=server>
        '国家和领导人以|分割显示
       <template name=ItemTemplate>
            <%# DataBinder.Eval(Container.DataItem,"Country") %>
            ---
            <%# DataBinder.Eval(Container.DataItem,"Name") %>
       </template>

       <template name=SeparatorTemplate>
            |
       </template>
    </asp:Repeater>
    </center>
  </body>
<html>
```

2．输出结果



2.2.14 数据列表 DataList

数据列表显示跟重复列表（Repeator）比较类似，但是它可以选择和修改数据项的内容。数据列表的数据显示和布局也如同重复列表都是通过“ 模板 ”来控制的。同样的，模板至少要定义一个“ 数据项模板 ”（ItemTemplate）来指定显示布局。数据列表支持的模板类型更多，它们如下：

1） ItemTemplate 模板，数据项模板，必需的，它定义了数据项极其表现形式。
2） AlternatingItemTemplate 模板，，数据项交替模板，为了使相邻的数据项能够有所区别，可以定义交替模板，它使得相邻的数据项看起来明显不同，缺省情况下，它和 ItemTemplate 模板定义一致，即缺省下相邻数据项无表示区分。
3） SeparatorTemplate 模板，分割符模板，定义数据项之间的分割符。
4） SelectedItemTemplate 模板，选中项模板，定义被选择的数据项的表现内容与布局形式，当未定义”SelectedItemTemplate”模板时，选中项的表现内容与形式无特殊化，由 ItemTemplate 模板定义所决定。
5） EditItemTemplate 模板，修改选项模板，定义即将被修改的数据项的显示内容与布局形式，缺省情况下，修改选项模板就是数据项模板（ItemTemplate）的定义。
6） HeaderTemplate 模板，报头定义模板，定义重复列表的表头表现形式。
7） FooterTemplate 模板，表尾定义模板，定义重复列表的列表尾部的表现形式。

数据列表还可以通过风格形式来定义模板的字体、颜色、边框。每一种模板都有它自己的风格属性。例如，可以通过设置修改选项模板的风格属性来指定它的风格。

此外，还有一些其他属性可以导致数据列表的显示有较大的改变，下面择重说明。

RepeatLayout：显示布局格式，指定是否以表格形式显示内容。

RepeatLayout.Table 指定布局以表格形式显示。

RepeatLayout.Flow 指定布局以流格式显示，即不加边框。

RepeatDirection：显示方向，指定显示是横向显示还是纵向显示

RepeatDirection.Horizontal 指定是横向显示

RepeatDirection.Vertical 指定是纵向显示

RepeatColumns：一行显示列数，指定一行可以显示的列数，缺省情况下，系统设置为一行显示一列。这里需要注意的是，当显示方向不同时，虽然一行显示的列数不变，但显示的布局和显示内容的排列次序却有可能大不相同。

例如：有 10 个数据需要显示，RepeatColumns 设定为 4，即一行显示 4 列时

当 RepeatDirection=RepeatDirection.Horizontal 横向显示时，显示布局如下：

Item1　Item2　Item3　Item4

Item5　Item6　Item7　Item8

Item9　Item10

当 RepeatDirection=RepeatDirection.Vertical 纵向显示时，显示布局如下：

Item1　Item4　Item7　Item10

Item2　Item5　Item8

Item3　Item6　Item9

BorderWidth：当 RepeatLayout=RepeatLayout.Table 即以表格形式显示时，边框的线宽度 Unit.Pixel(x) x>=0,当 x 为 0 时无边框

GridLines: 当 RepeatLayout=RepeatLayout.Table 以表格形式显示时，在表格当中是否有网隔线分离表格各单元。

GridLines=GridLines.Both，有横向和纵向两个方向的分割线。

GirdLines=GridLines.None，无论横向还是纵向均无分割线。

例子：演示以上介绍的各属性的设置对数据列表输出的影响，并且当数据项被选中时，数据项以粉红色来反显。

1．源程序(**DataList.aspx**)

```
<!--源文件：form\ServerControl\dataList.aspx-->
<%@ Import Namespace="System.Data" %>
<html>
<script language="VB" runat="server">
'创建初始化表和载入实验数据
    Function LoadData() As ICollection
        Dim dt As DataTable
        Dim dr As DataRow
        Dim i As Integer
        '创建数据表
        dt = New DataTable
        '建立数据项结构
        dt.Columns.Add(New DataColumn("Content", GetType(String)))
        '载入 10 个实验数据
```

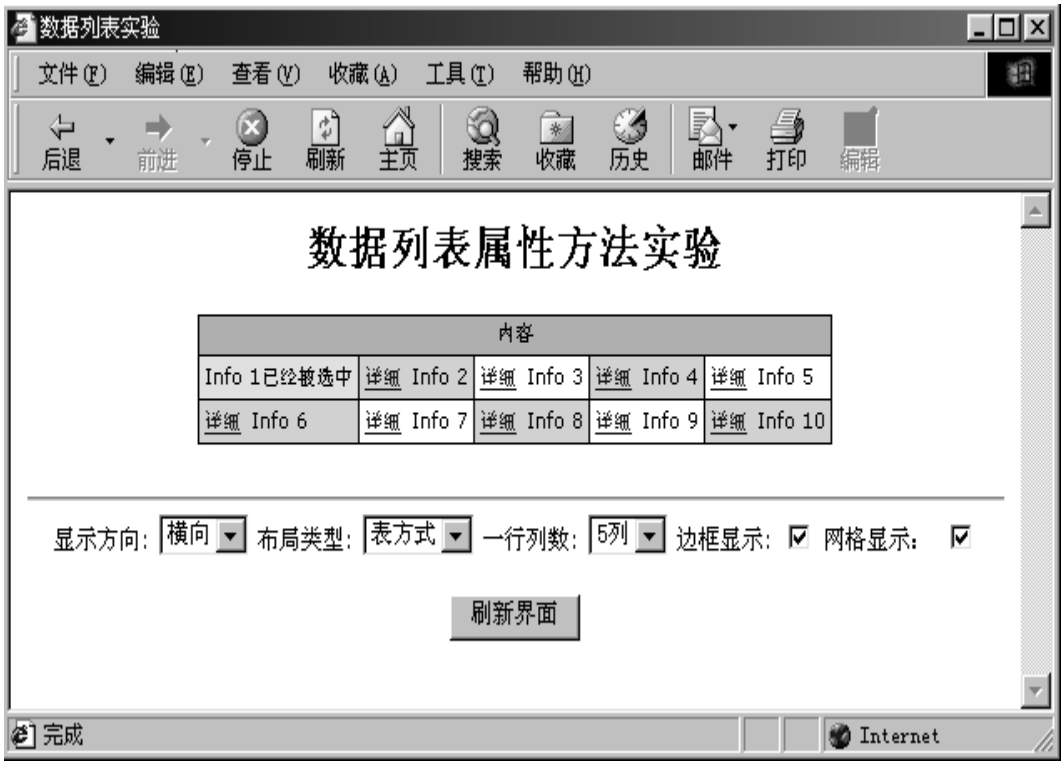
```
        For i = 1 To 10
            dr = dt.NewRow()
            dr(0) = "Info " & i.ToString()
            dt.Rows.Add(dr)
        Next
        '为数据表建立一个数据视图，并将其返回
        LoadData = New DataView(dt)
    End Function
    Sub Page_Load(s As Object, e As EventArgs)
        If Not IsPostBack Then
            DataList1.DataSource = LoadData()
            DataList1.DataBind
        End If
    End Sub
    Sub DataList1_ItemCommand(s As Object, e As DataListCommandEventArgs)
        Dim cmd As String = e.CommandSource.CommandName

        If cmd = "select" Then
            DataList1.SelectedIndex = e.Item.ItemIndex
        End If

        DataList1.DataSource = LoadData()
        DataList1.DataBind
    End Sub
'当刷新按钮按下后，对数据列表属性重新设置
    Sub RefreshBtn_Click(s As Object, e As EventArgs)
        If lstDirection.SelectedIndex = 0
            DataList1.RepeatDirection = RepeatDirection.Horizontal
        Else
            DataList1.RepeatDirection = RepeatDirection.Vertical
        End If

        If lstLayout.SelectedIndex = 0
            DataList1.RepeatLayout = RepeatLayout.Table
        Else
            DataList1.RepeatLayout = RepeatLayout.Flow
        End If
        If chkBorder.Checked And DataList1.RepeatLayout = RepeatLayout.Table Then
            DataList1.BorderWidth = Unit.Pixel(1)
        Else
            DataList1.BorderWidth = Unit.Pixel(0)
        End If

        If chkGridLines.Checked And DataList1.RepeatLayout = RepeatLayout.Table then
```

```
                DataList1.GridLines = GridLines.Both
            Else
              DataList1.GridLines = GridLines.None
            End If
            DataList1.RepeatColumns=lstColsPerLine.SelectedIndex + 1
        End Sub
</script>
<head>
<title>
数据列表实验
</title>
</head>
<body>
  <center>
  <h2>
  数据列表属性方法实验
  </h2>
      <form runat=server>
      <font face="Verdana" size="-1">
          <asp:DataList id="DataList1" runat="server"
              BorderColor="black"
              CellPadding="3"
              Font-Name="Verdana"
              Font-Size="8pt"
              HeaderStyle-BackColor="#aaaadd"
              AlternatingItemStyle-BackColor="#ccccff"
          SelectedItemStyle-BackColor="#ffccff"
              OnItemCommand="DataList1_ItemCommand"
              >
                <template name="HeaderTemplate">
                <h><center>内容</center></h>
                </template>
                <template name="ItemTemplate">
                  <asp:LinkButton id="DetailBtn" runat="server" Text=" 详 细 "
CommandName="select" />
                  <%# DataBinder.Eval(Container.DataItem, "Content") %>
                </template>
                <template name="SelectedItemTemplate">
                  <%# DataBinder.Eval(Container.DataItem, "Content") %>已经被选中
                </template>
          </asp:DataList>
          <p>
          <hr>
          显示方向:
```

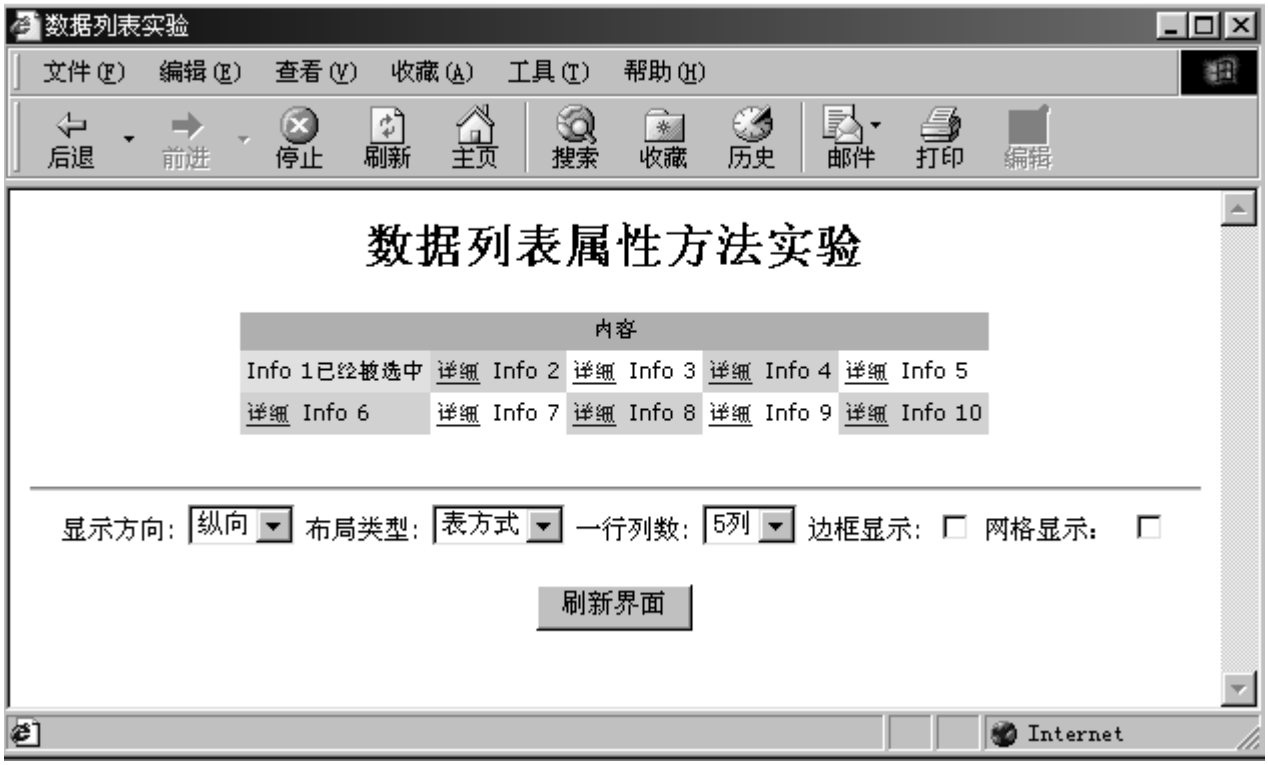
```
            <asp:DropDownList id=lstDirection runat="server">
                <asp:ListItem>横向</asp:ListItem>
                <asp:ListItem>纵向</asp:ListItem>
            </asp:DropDownList>
            布局类型:
            <asp:DropDownList id=lstLayout runat="server">
                <asp:ListItem>表方式</asp:ListItem>
                <asp:ListItem>流方式</asp:ListItem>
            </asp:DropDownList>
            一行列数:
            <asp:DropDownList id=lstColsPerLine runat="server">
                <asp:ListItem>1 列</asp:ListItem>
                <asp:ListItem>2 列</asp:ListItem>
                <asp:ListItem>3 列</asp:ListItem>
                <asp:ListItem>4 列</asp:ListItem>
                <asp:ListItem>5 列</asp:ListItem>
            </asp:DropDownList>
            边框显示:
            <asp:CheckBox id=chkBorder runat="server" />
            网格显示：
        <asp:CheckBox id=chkGridLines runat="server" />
            <p>

            <asp:Button id=RefreshBtn Text="刷新界面" OnClick="RefreshBtn_Click"
runat="server"/>

        </font>
        </form>
    </center>
</body>
</html>
```

2．开始时的界面显示如下，（方向为横向，表方式，一行一列，无边框及网格）

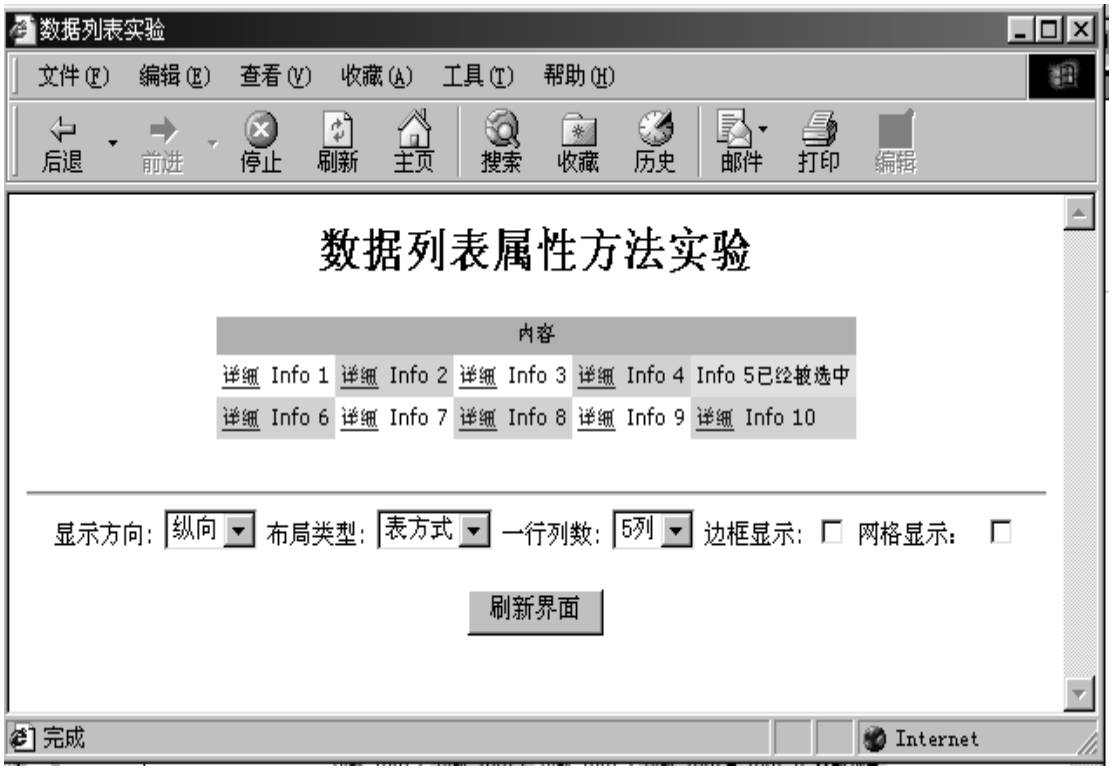
3．当选择显示方向为横向，表方式，一行含 5 列，显示边框和网格时，界面显示如下：



4．选择纵向显示，表方式，一行含 5 列，无边框，无网格时，界面显示如下：



5．当在步骤 4 的基础上选择了第 5 项数据项时，界面显示如下：



接下来，我们讨论一下一种比较有实际意义的应用，即对选中数据项的修改的实现。

首先是对模板 EditItemTemplate 的定义，通常做法是排列可以进行修改的内容，然后定义一个修改确认键和一个修改取消键。

然后应定义数据列表支持的三种消息处理函数即 OnEditCommand、OnUpdateCommand、OnCancelCommand(编辑事件处理、修改事件处理、撤消修改事件处理)

编辑事件处理：通常设置数据列表的 EditItemIndex 属性为选中的数据项索引，然后重载数据列表。

```
Protected Sub DataList_EditCommand(Source As Object, e As DataListCommandEventArgs)
    DataList1.EditItemIndex = CType(e.Item.ItemIndex, Integer)
    ‘重新加载并绑定数据
    BindList()
End Sub
```

取消修改事件处理：通常设置数据列表的 EditItemIndex 为-1，表示没有数据项需要修改，然后重载数据列表

```
Protected Sub DataList_CancelCommand(Source As Object, e As DataListCommandEventArgs)
    DataList1.EditItemIndex = -1
    BindList()
End Sub
```

修改事件处理：通常先修改数据源的数据，然后设置数据列表的 EditItemIndex 为-1，最后重载数据列表。

```
Sub DataList_UpdateCommand(Source As Object, e As DataListCommandEventArgs)
‘修改数据源数据，应根据具体情况而变
ModifySource()
DataList.EditItemIndex=-1
BindList
End Sub
```

例子：显示一个关于书籍修改的实例。一条书籍记录包含序号、书名、价格信息。初始化数据时，我们设置序号为 1-6，书名为“ 书名 ”+序号，价格为 1.11*序号。

1． 源程序(**FormDataList01.aspx**)

```
<<!--源文件：form\ServerControl\FormDataList01.aspx-->
<%@ Import Namespace="System.Data" %>
<html>
    <script language="VB" runat="server">
    dim Book As DataTable
    dim BookView As DataView
    '设置数据源，并绑定
    Sub BindList()
        DataList1.DataSource= BookView
        DataList1.DataBind
    End Sub

    Sub Page_Load(s As Object, e As EventArgs)

        Dim dr As DataRow
    '如果没有连接变量 session_book，定义数据表 Book,并载入实验数据
    if session("session_Book") = Nothing then
            Book = New DataTable()
            Book.Columns.Add(new DataColumn("num", GetType(string)))
            Book.Columns.Add(new DataColumn("name", GetType(String)))
            Book.Columns.Add(new DataColumn("price", GetType(String)))
            session("session_Book") = Book
            '载入部分测试数据
            For i = 1 To 6
                dr = Book.NewRow()
                dr(0)=i.ToString
                dr(1) = "书名 " & i.ToString
                dr(2) = ( 1.11* i).ToString
                Book.Rows.Add(dr)
            Next
       '有 session_book 变量，直接引用
```

```
            Else
                Book = session("session_Book")
            end if
        '产生数据视图，并按 num 字段排序
        BookView = New DataView(Book)
        BookView.Sort="num"
        '初次需绑定数据源
        if Not IsPostBack then
            BindList
        End If

End Sub

'编辑处理函数
Sub DataList_EditCommand(sender As Object, e As DataListCommandEventArgs)
        DataList1.EditItemIndex = e.Item.ItemIndex
        BindList
End Sub
'取消处理函数
Sub DataList_CancelCommand(sender As Object, e As DataListCommandEventArgs)
        DataList1.EditItemIndex = -1
        BindList
End Sub
'更新处理函数
Sub DataList_UpdateCommand(sender As Object, e As DataListCommandEventArgs)
        Dim lbl1 As Label = e.Item.FindControl("lblNum")
        Dim txt2 As TextBox = e.Item.FindControl("txtBook")
        Dim txt3 As TextBox = e.Item.FindControl("txtPrice")

        dim strNum as String
        dim strBook as String
        dim strPrice as String

        strNum=lbl1.text
        strBook=txt2.text
        strPrice=txt3.text
        '用先删除再插入的方式，实现数据的更新操作
        BookView.RowFilter = "num='" & strNum & "'"
        If BookView.Count > 0 Then
            BookView.Delete(0)
        End If

        BookView.RowFilter = ""
dim dr as DataRow=Book.NewRow()
```

```
            dr(0) = strNum
            dr(1) = strBook
            dr(2) = strPrice
            Book.Rows.Add(dr)

            DataList1.EditItemIndex = -1
            BindList
        End Sub

        </script>
<head>
<title>
数据列表修改实验
</title>
</head>
<body>
<center>
        <h2>数据列表修改实验</h2>
        <hr>
        <p></p>

        <form runat=server>
        <font face="Verdana" size="-1">
            <!--编辑时显示绿色，并定义编辑、修改、取消时的处理函数-->
            <asp:DataList id="DataList1" runat="server"
                BorderColor="black"
                BorderWidth="1"
                GridLines="Both"
                CellPadding="3"
                CellSpacing="0"
                Font-Name="Verdana"
                Font-Size="8pt"
                Width="150px"
                HeaderStyle-BackColor="#aaaadd"
                AlternatingItemStyle-BackColor="Gainsboro"
                EditItemStyle-BackColor="green"
                OnEditCommand="DataList_EditCommand"
                OnUpdateCommand="DataList_UpdateCommand"
                OnCancelCommand="DataList_CancelCommand"
                >
                  <template name="HeaderTemplate">
                  <center><h>书籍序号</h></center>
                  </template>
                  <template name="ItemTemplate">
```

```
                    <asp:LinkButton id="button1" runat="server" Text="详 细 "
CommandName="edit" />
                    <%# Container.DataItem("name")  %>
                </template>
                <template name="EditItemTemplate">
                    书籍: 序号
                    <asp:Label id="lblNum" runat="server" Text='<%#
Container.DataItem("num")  %>' /><br>
                    书名:
                    <asp:TextBox id="txtBook" runat="server" Text='<%#
Container.DataItem("name")  %>' /><br>
                    价格:
                    <asp:TextBox id="txtPrice" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "price") %>' />
                    <br>
            <center>
                    <asp:Button id="button2" runat="server" Text="修 改 "
CommandName="update" />
                    <asp:Button id="button3" runat="server" Text="撤 消 "
CommandName="cancel" />
                    </center>
                </template>
            </asp:DataList>
        </font>
        </form>
</center>
</body>
</html>
```

2．准备对第 2 项进行修改，此时的画面如下：

3.把序号为 2 的书籍的价格改为 9.99 以后，重新进入其编辑状态后，它的输出画面如下：

2.2.15 数据表格 DataGrid

数据表格服务器端控件以表格形式显示数据内容，同时还支持数据项的选择、排序、分页和修改。缺省情况下，数据表格为数据源中每一个域绑定一个列，并且根据数据源中每一个域中数据的出现次序把数据填入数据表格中的每一个列中。数据源的域名将成为数据表格的列名，数据源的域值以文本标识形式填入数据表格中。

通过直接操作表格的 Columns 集合，可以控制数据表格各个列的次序、表现方式以及显示内容。缺省的列为 Bound 型列，它以文本标识的形式显示数据内容。此外，还有许多类型的列类型可供用户选择。

列类型的定义有两种方式：显视的用户定义列类型和自动产生的列类型（AutoGenerateColumns）。当两种列类型定义方式一起使用时，先用用户定义列类型产生列的类型定义，接着剩下的再使用自动列定义规则产生出其他的列类型定义。请注意自动定义产生的列定义不会加入 Columns 集合。

列类型介绍：

1） bound column ，列可以进行排序和填入内容。这是大多数列缺省用法。

两个重要的属性为：HeaderText 指定列的表头显示

DataField 指定对应数据源的域

2） hyperlink column，列内容以 hyperlink 控件方式表现出来。它主要用于从数据表格的一个数据项跳转到另外的一个页面，做出更详尽的解释或显示。

重要的属性有：

HeaderText 指定列表头的显示

DataNavigateUrlField 指定对应数据源的域作为跳转时的参数

DataNavigateUrlFormatString 指定跳转时的 url 格式
DataTextField 指定数据源的域作为显示列内容来源

3） button column，把一行数据的用户处理交给数据表格所定义的事件处理函数。通常用于对某一行数据进行某种操作，例如，加入一行或者是删去一行数据等等。
重要的属性有：

HeaderText 指定列表头的显示
Text 指定按钮上显示的文字
CommandName 指定产生的激活命令名

4） Template column，列内容以自定义控件组成的模板方式显示出来。通常用作用户需要自定义显示格式的时候。
5） Edit Command column，当数据表格的数据项发生编辑、修改、取消修改时，相应处理函数的入口显示。它通常结合数据表格的 EditItemIndex 属性来使用，当某行数据需要编辑、修改、取消操作时，通过它进入相应的处理函数。例如，当需要对某行数据进行修改（update）时，通过它进入修改的处理步骤中。

其他重要列属性介绍：

1） Visible 属性，控制定义的列是否出现在显示的数据列表中。
2） AllowSorting 属性，是否可以进行列排序。当 AollowSorting=true 时，可以以点击列的列表头的方式，把数据以该列次序进行排序。缺省的（即载入数据后）的排序方式，实际上是以数据在数据源中的排列次序进行排序的。
3） AllowPage 属性，是否以分页方式显示数据。当对有大量数据的数据源进行显示时，可以以例如 10 行一页的方式来显示数据，同时显示一个下页/前页的按钮，按下按钮可以以向前或向后的方式浏览整个数据源的数据。当 AllowPage=true 时，即以分页方式进行显示。可以通过设定 CurrentPageIndex 属性来直接跳转到相应的数据页。

例子：演示以上各种类型的列定义的用法

1．源程序(**FormDataGrid.aspx**)

```
<!--源文件：form\ServerControl\FormDataGrid.aspx-->
<%@ Import Namespace="System.Data" %>

<html>
<script language="VB" runat="server">
    dim Order as DataTable
    dim OrderView as DataView

    '对数据表格 1 创建数据表，并返回数据视图
    Function LoadData() As ICollection

        Dim dt As DataTable
        Dim dr As DataRow
        Dim i As Integer

        '创建数据表
```

```
        dt = New DataTable
        dt.Columns.Add(New DataColumn("Num", GetType(Integer)))
        dt.Columns.Add(New DataColumn("Name", GetType(String)))
        dt.Columns.Add(New DataColumn("DtTm", GetType(DateTime)))
        dt.Columns.Add(New DataColumn("Assembly", GetType(Boolean)))
        dt.Columns.Add(new DataColumn("Price", GetType(Double)))

        '载入数据
        For i = 1 To 6
            dr = dt.NewRow()
            dr(0) = i
            dr(1) = "书名  " + i.ToString()
            dr(2) = DateTime.Now.ToShortTimeString
            If (i Mod 2 <> 0) Then
                dr(3) = True
            Else
                dr(3) = False
            End If
            dr(4) = 1.11 * i
            '把产生的数据加入数据表中
            dt.Rows.Add(dr)
        Next


        LoadData = New DataView(dt)

    End Function

    '页面初始化，分别对 DataGrid1 和 DataGrid2 绑定数据源
    Sub Page_Load(sender As Object, e As EventArgs)

        If Session("session_order") = Nothing Then
            Order = New DataTable()
            Order.Columns.Add(new DataColumn("Name", GetType(string)))
            Order.Columns.Add(new DataColumn("Price", GetType(string)))
            Session("session_order") = Order
        Else
            Order = Session("session_order")
        End If
        OrderView = New DataView(Order)
        DataGrid2.DataSource = OrderView
        DataGrid2.DataBind

        If Not IsPostBack Then
```

```
    DataGrid1.DataSource = LoadData()
        DataGrid1.DataBind
        End If

    End Sub

  '对 ButtonColumns 的处理函数集合
    Sub Grid_Command(sender As Object, e As DataGridCommandEventArgs)

        Dim dr As DataRow = order.NewRow()

        Dim Cell1 As TableCell = e.Item.Cells(3)
        Dim Cell2 As TableCell = e.Item.Cells(6)
        Dim name As String = Cell1.Text
        Dim price As String = Cell2.Text

        If e.CommandSource.CommandName = "Add" Then
            dr(0) = name
            dr(1) = price
            order.Rows.Add(dr)
        Else
            OrderView.RowFilter = "name='" & name & "'"
            If OrderView.Count > 0 Then
                OrderView.Delete(0)
            End If
            OrderView.RowFilter = ""
        End If
        DataGrid2.DataBind()
    End Sub

</script>
<head>
<title>
数据表格实验
</title>
</head>

<body>
<center>
  <h2>数据表格列类型实验</h2>
  <hr>
  <p></p>

    <form runat=server>
```

```
        <h3><b>图书清单</b></h3>
        <ASP:DataGrid id="DataGrid1" runat="server"
          BorderColor="black"
          BorderWidth="1"
          GridLines="Both"
          CellPadding="3"
          CellSpacing="0"
          Font-Name="Verdana"
          Font-Size="8pt"
          HeaderStyle-BackColor="#aaaadd"
          AutoGenerateColumns="false"
          OnItemCommand="Grid_Command">
            <property name="Columns">
              <!-- 2 个 ButtonColumn 示例-->
              <asp:ButtonColumn HeaderText="操作" Text="订购" CommandName="Add" />
              <asp:ButtonColumn HeaderText="操作" Text="退订" CommandName="Remove"
/>
              <!-- HyperLinkColumn 示例  -->
          <asp:HyperLinkColumn
                  HeaderText="链接"
                  DataNavigateUrlField="Num"
                  DataNavigateUrlFormatString="FormDataGrid01.aspx?id={0}"
                  DataTextField="Num"
                  Target="_new"
              />
              <!-- 2 个标准 BoundColumn 示例  -->
              <asp:BoundColumn HeaderText="书  名" DataField="Name" />
              <asp:BoundColumn HeaderText="入库时间" DataField="DtTm"/>
              <! -- 1 个 TemplateColumn 示例  ,以 CheckBox 来表示布尔型数据  -->
              <asp:TemplateColumn HeaderText="合  集">
                  <template name="ItemTemplate">
                      <asp:CheckBox                ID=Chk1                Checked='<%#
DataBinder.Eval(Container.DataItem, "Assembly") %>' Enabled="false" runat="server" />
                  </template>
              </asp:TemplateColumn>

              <asp:BoundColumn    HeaderText="  价      格  "    DataField="Price"
DataFormatString="{0:c}" ItemStyle-HorizontalAlign="right" />
            </property>

        </asp:DataGrid>
       <hr>
      <h3><b>订购清单</b></h3>
      <ASP:DataGrid id="DataGrid2" runat="server"
```

```
            BorderColor="black"
            BorderWidth="1"
            CellPadding="3"
            Font-Name="Verdana"
            Font-Size="8pt"
            HeaderStyle-BackColor="#aaaadd"
            />

    </form>
</center>
</body>
</html>
```

文件 **FormDataGrid01.aspx** 的内容：

```
<!--源文件：form\ServerControl\FormDataGrid01.aspx-->
<html>
<head>
<title>
数据表格链接测试实验
</title>
<script language="VB" runat="server">

    Dim num As String

    Sub Page_Load(sender As Object, e As EventArgs)
        num=Request.QueryString("id")
    End Sub

</script>

</head>
<body bgcolor=#ccccff>
<center>
    <h2>数据表格链接测试结果画面</h2>
    <hr>
    <p></p>

    <h4>您选择的是 第<u> <%= num %></u>本藏书</h4>

</body>
</html>
```

2．开始时画面：

3．当选择订购了第一本和第三本后的画面如下：

4．当选择退订第三本书后的画面如下：

5．当点击连接第六项时的画面如下：

2.2.16 小结

本章主要讲述了几个服务器端的控件、它们的校验、取值方法等，从中我们可以看到 asp.net 中各种控件功能是非常强大的，如上面的例子所示，我们甚至可以用一个简单的语句就可以验证输入的合法性。对取值，我们也有简单的方法，对比于用 html 所写的代码，我们觉得用 asp.net 所写的是简单了很多。

第三章 自定义控件

asp.net 中提供的增加内嵌服务器控件的功能，使你能够多次的轻松增加你所定义的各种控件。事实上，对于表单等各种控件，可以不用更改或者稍微更改一下就可以多次使用的。在通常情况下，我们把一个用作服务器控件的 web 表单统称为用户控件，我们用一个.ascx 为后缀的文件保存起来，这样的保存使得它不被当作一个 web 表单来运行，当我们在一个.aspx 文件中使用它时，我们用 Register 方法来进行调用，假设我们有一个文件名为 saidy.ascx 的文件，我们用下面的语句来调用它：

```
<%@ Register TagPrefix="Acme" TagName="Message" Src="saidy.ascx" %>
```

上面的 TagPrefix 标记为用户控件确定个唯一的名字空间，TagName 为用户控件确定一个唯一的名称，你也可以用其它的名字代替“ Message “ ，Src 为确定所包含的文件名称和路径。这样，我们就可以用下面的语句来调用它了：

```
<Acme:Message runat="server"/>
```

下面我们来看看具体的应用

2.3.1 小页面控件

我们建立两个简单文件来说明这个控件的使用方法：con01.aspx、con01.ascx，在 **con01.ascx** 文件里我们只有一句话：

```
<a href="http://www.yesky.com">欢迎访问天极网站</a>
```

然后我们在文件 **con01.aspx** 里面进行注册：

```
<%@ Register TagPrefix="saidy" TagName="info" Src="con01.ascx" %>
```

页面上的应用我们用这句话来表达：

```
<saidy:info runat="server"/>
```

con01.aspx 文件的完整代码如下：

```
<!--源文件：form\CustomControl\con01.aspx-->
<!--注册小页面控件-->
<%@ Register TagPrefix="saidy" TagName="info" Src="con01.ascx" %>
<html>
<body>
<BR><BR><BR>
<CENTER>
   调用结果
<BR><BR>
   <saidy:info runat="server"/>
</CENTER>
<BR><BR>
</body>
</html>
```

下面我们访问 con01.aspx，显示如下：

2.3.2 代码和模板的分离

在编制 asp.net 程序时，我们会使用模板(Template)。那么什么是模板呢？相信大家都使用过 WORD，当我们在新建一个 WORD 文件的时候，我们可以建立模板。通过使用模板，我们就固定了文档的风格，这样就可以在模板上完善我们的内容。所以我们使用模板一个好处是：文字录入和编排界面是分开的。而且模板可以重复使用。好了，通过上面的介绍，我们对模板就有了一定的认识。 我们在编制.NET 程序时，使用模板将对主程序代码大大简化。模板的定义是使用<template>和</template>标示符的。文件保存为.ascx 文件。下面的代

码是一个典型的模板的定义。

```
 <template name="itemtemplate">
   <table cellpadding=10 style="font: 10pt verdana">
        <tr>
         <td valign="top">
       <b>所在系: </b><%# DataBinder.Eval(Container.DataItem, "dept") %><br>
       <b>姓名: </b><%# DataBinder.Eval(Container.DataItem, "name") %><br>
       <b>性别: </b><%# DataBinder.Eval(Container.DataItem, "sex") %><br>
       <b>年级: </b><%# DataBinder.Eval(Container.DataItem, "grade") %>
           </td>
         </tr>
</table>
 </template>
```

在这一模板中，我们使用了数据绑定控件，关于数据绑定控件，请参阅其它章节。同时我们还定义了数据的显示方式。那么在主程序中如何调用呢？请看下面的代码：

```
1.  <%@ Register TagPrefix="Acme" TagName="StuList" Src="form32.ascx" %>
2.  <html>
3.  <body style="font: 10pt verdana">
4.  <b><center><h3>模板示例</h3></center></b>
5.  <form runat="server">
6.  <Acme: StuList runat="server"/>
7.  </form>
8.  </body>
9.  </html>
```

其实，模板也属于自定义控件(User Control)，所以我们在使用时，要先注册(Register)。对主程序的第一行代码，TagPrefix 定义了一个不重复的名字空间(Name Space)。TagName 为自定义控件定义了一个名称。然后，我们就要指明使用的模板的文件名。注册完自定义控件后，我们就可以把此控件认为是服务器端控件。要使用服务器端控件，我们要做什么工作呢？对了，要使用 runat=”server”属性了。请参考第 7 行代码。

好了，现在我们就看一个完整的例子！这个例子包含了两个文件，一个主程序文件（template.aspx），另一个是用户自定义控件文件（template.ascx）。先看 template.aspx 文件。

```
<!--源文件：form\CustomControl\template.aspx-->
   <%@ Register TagPrefix="Acme" TagName="stuList" Src="zy.ascx" %>
  <html>
  <body style="font: 10pt verdana">
  <b><center><h3>模板示例</h3></center></b>
  <form runat="server">
  <Acme:stuList runat="server"/>
  </form>
```

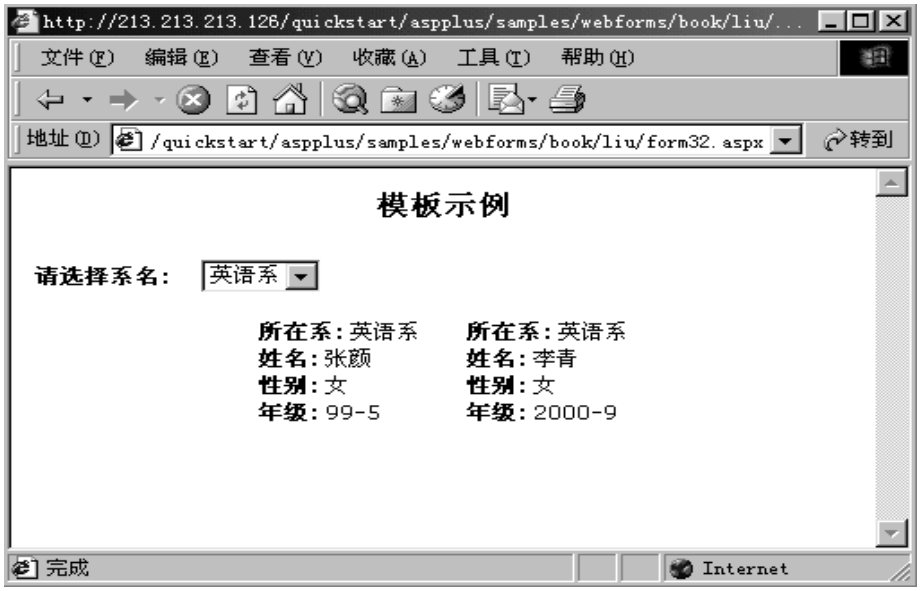
```
</body>
</html>
```

现在我们再来看 **template.ascx**：

```
<!--源文件：form\CustomControl\template.ascx-->
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SQL" %>
<script language="VB" runat="server">
      Sub Page_Load(Src As Object, E As EventArgs)
            If Not (Page.IsPostBack)
              Dim DS As DataSet
              Dim MyConnection As SQLConnection
              Dim MyCommand As SQLDataSetCommand
    MyConnection = New SQLConnection("server='iceberg';uid=sa;pwd=;database=info")
    MyCommand = New SQLDataSetCommand("select * from infor where dept='" &
Category.SelectedItem.Value & "'", MyConnection)
              DS = New DataSet()
              MyCommand.FillDataSet(DS, "infor")
              MyDataList.DataSource = DS.Tables("infor").DefaultView
              MyDataList.DataBind()
          End If
      End Sub

      Sub Category_Select(Sender As Object, E As EventArgs)
              Dim DS As DataSet
              Dim MyConnection As SQLConnection
              Dim MyCommand As SQLDataSetCommand
        MyConnection = New SQLConnection("server='iceberg';uid=sa;pwd=;database=info")
        MyCommand = New SQLDataSetCommand("select * from infor where dept='" &
Category.SelectedItem.Value & "'", MyConnection)
              DS = New DataSet()
              MyCommand.FillDataSet(DS, "infor")
              MyDataList.DataSource = DS.Tables("infor").DefaultView
              MyDataList.DataBind()
      End Sub
</script>
<table style="font: 10pt verdana">
<center>
<tr>
  <center><td><b>请选择系名:</b></td></center>
<td style="padding-left:15">
<center>          <ASP:DropDownList AutoPostBack="true" id="Category"
OnSelectedIndexChanged="Category_Select" runat="server">
          <ASP:ListItem value="信息系">信息系</ASP:ListItem>
          <ASP:ListItem value="工程系">工程系</ASP:ListItem>
```

```
            <ASP:ListItem value="英语系">英语系</ASP:ListItem>
          </ASP:DropDownList></center>

</td>
  </tr>
</table>
<ASP:DataList id="MyDataList" BorderWidth="0" RepeatColumns="2" runat="server">
<template name="itemtemplate">
   <table cellpadding=10 style="font: 10pt verdana">
            <tr>
              <td valign="top">
                <b>所在系: </b><%# DataBinder.Eval(Container.DataItem, "dept") %><br>
                <b>姓名: </b><%# DataBinder.Eval(Container.DataItem, "name") %><br>
                <b>性别: </b><%# DataBinder.Eval(Container.DataItem, "sex") %><br>
                <b>年级: </b><%# DataBinder.Eval(Container.DataItem, "grade") %>
              </td>
            </tr>
</center>
</table>
      </template>
</ASP:DataList>
```

运行的效果图如下：



这样，一个完整的例子就做好了！实现了代码和模板的分离。试一下吧！

2.3.3 自定义控件

在 asp.net 中，除了我们应用的服务端控件之外，我们还可以创建自己的服务端控件，

这样的控件叫 Pagelet。我们来介绍如何创建一个 Pagelet，这个 Pagelet 的功能是在被访问时返回一个消息。

我们创建一个 Pagelet，用来返回一个消息在客户端的浏览器上：
Welcome.ascx：
<!--源文件：form\CustomControl\welcome.ascx-->
欢迎来到我这里啊！！！

就这么简单，当然你也可以让它复杂一点。当一个 Pagelet 被创建后，我们就可以通过下面的记录指示来调用它：

<% @ Register TagPrefix="wmessage" TagName="wname" Src="Welcome.ascx" %>

TagPrefix 为 Pagelet 指定一个唯一的名字空间，TagName 是 Pagelet 的唯一名字，当然你也可以换成其他的不是”wname“的名称如：TagName="saidy"。Src 属性是指指向 Pagelet 的虚拟路径。

一旦我们注册了 Pagelet，我们就可以向用普通的空件一样来应用它：

<wmessage:wname runat="server"/>

下面的例子示范了自定义的控件的应用（**welcome.aspx**）：

```
<!--源文件：form\CustomControl\welcome.aspx-->
<%@ Register TagPrefix="wmessage" TagName="wname" Src="Welcome.ascx" %>
<html>
<title>自定义的控件</title>
<h3>.NET->Pagelet</h3>
<wmessage:wname runat="server"/>
</body>
</html>
```

客户端的访问如下：

2.3.4 组合控件

1.定义
以类组合形式把已有的控件编译后形成自己定制的控件。实际上组合控件在效果上与利用内置控件形成的用户自定义控件一样，不同处在于，用户自定义控件含有一个.ascx 的纯文本控制文件，而组合控件则利用编译后的代码。
2．步骤
1.)重新定义从 Control 继承来的 CreateChildControls 方法。
2.)如果组合控件要保持于页面上，须完成 System.Web.UI.INamingContainer 接口。
3．例子:
演示一个自定义控件，当选择不同按钮时显示不同内容。
1)控件定义

```
'文件名:form\CustomControl\FormCustom.vb
Option Strict Off

Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls

Namespace test
'定义类 tryVB
Public Class tryVB : Inherits Control : Implements INamingContainer
'定义属性 value,实为 TextBox 控件的 Text 属性
Public Property value As String
    Get
    Dim Ctrl As TextBox = Controls(1)
    Return Ctrl.text
    End Get

    Set
    Dim Ctrl As TextBox = Controls(1)
    Ctrl.Text = value
    End Set
End Property

Protected Overrides Sub CreateChildControls()
'重载 CreateChildControls 方法
    Me.Controls.Add(New LiteralControl("选择结果为: "))

    Dim Box As New TextBox
```

```
        Box.Text = "    "
        Me.Controls.Add(box)

End Sub

End Class

End Namespace
```

2)定义控件的编译

:批处理文件 form\CustomControl**FormCustom.bat** 的内容：

```
vbc /t:library /out:..\bin\testVB.dll /r:System.dll /r:System.Web.dll FormCustom.vb
```

请注意把生成的 testVB.dll 放到正确的目录中，以便 asp.net 解释时能够找到相应的类。

3)自定义组合控件的使用

```
<!--源文件：form\CustomControl\formcustom.aspx-->
<%@ Register TagPrefix="test" Namespace="test" %>
<!--首先注册 test 命名空间-->
<html>
    <script language="VB" runat=server>
        Private Sub LeftBtn_Click(Sender As Object, E As EventArgs)
        '当选择左边的按钮时的显示
            CustControl.Value = "您选择的是 Yes 按钮"
        End Sub
        Private Sub RightBtn_Click(Sender As Object, E As EventArgs)
        '当选择右边的按钮时的显示
            CustControl.value = "您选择的是 No 按钮"
        End Sub

    </script>

    <body>
    <center>
        <form method="POST" action="formcustom.aspx" runat=server>
            <!--引用自定义的组合控件 tryVB-->
            <test:tryVB id="CustControl" runat=server/>
            <br>
            <!--画两个按钮供选择-->
            <asp:button text="是[Yes]" OnClick="LeftBtn_Click" runat=server/>
            <asp:button text="否[No]" OnClick="RightBtn_Click" runat=server/>
        </form>
```

```
    </center>
    </body>
</html>
```

输出结果：

2.3.5 继承控件

在学习了微软公司的.NET 平台为我们提供的大量功能强大的服务器端控件的使用方法以后，随着应用的深入，一些新的问题又出现了。首先是虽然有着大量的控制灵活的控件，但是否就真的满足了我们所有的需求？有时候，我们需要某种控件部分功能，又希望不要费太大的力气去实现，是否可以利用现有的控件来实现。再则，我们希望对某种控件进行改造，使它具有自己所希望的外形或者结果，而不是它缺省的方式运行。最后我们是否可以把自己经常用到的逻辑规则或者是应用界面作成用户控件，然后使用它就如同使用服务器控件那样方便。

其实以上三个问题，在现代面向对象设计方法中，是可以找到答案的。为最大可能的利用现有的开发成果，我们使用“ 继承 ”这一手段来节省开发的费用。光有继承不足以形成自己的应用，我们还可以利用“ 重载 ”和“ 多态 ”来形成自己的应用特点，使之区别于被继承的对象。为了使应用更加简洁和对外隐藏内部的实现、进一步实现代码重用，我们又使用了“ 封装 ”。

微软的.NET 平台是支持面向对象的设计方式的新型平台，所以支持并且鼓励用户在应用中设计和使用自己定义的控件。设计用户自己的控件就如同上面所述，有如下步骤：

1． 从 System.Web.UI.Control 类继承，并形成自己的类

为继承 Control 类，我们需引用 System、System.Web、System.Web.UI 类库，在 vb 环境下使用标识 Imports 来引入。为方便使用，我们还需定义一个命名空间以容纳多个类。在 vb 环境中使用 Namespace 空间名和 End Namespace 标识对来定义一个命名空间。定义一个类使用 Class ClassName 和 End Class 标识对。为表明类之间的继承关系，可以使用 Inherits 标识。

继承控件的类定义框架定义如下：

```
Imports System
Imports System.Web
Imports System.Web.UI

Namespace MyNamespace
    Public Class MyClass:Inherits Control
    …
    End Class
End Namespace
```

一个最简单的例子是从 Control 继承一个类，然后重载其 Render 方法。调用其 Render 方法即在页面以 h2 字体写出一行字。

```
Imports System
Imports System.Web
Imports System.Web.UI

Namespace MyNamespace
    Public Class MyClass:Inherits Control

    Protect overrides Sub Render(OutPut  as  HtmlTextWriter)
        OutPut.Write("<h2>这是一个最简单的控件继承例子！</h2>")
    End Sub
    End Class
End Namespace
```

2．定义自己的属性和方法，包括重载一些初始化的方法。

在 vb 中，以标识 overrides 指明该方法是一个重载函数。例如上面所举的 Render 方法：Protect overrides Sub Render(Output as HtmlTextWriter)

属性定义就较为复杂一点，首先是定义内部变量，可以为 Public 或者是 Private,当为 Public 时可以被外部直接存取，这种方式面向对象方法并不提倡，为 Private 时，不能直接被外部存取，只有通过内部提供的属性定义方式来存取；然后对需要提供给外部使用的内部变量进行属性存取方式定义。在 vb 中使用 Property 属性名 As 类型和 End Property 标识对来定义，Get/End Get 标识对间定义如何通过属性取得内部变量的值，Set/End Set 标识对间定义如何设置内部变量值。

例如：描述一个人的帐号信息，大致需要设定帐号（AcctNo）、身份证号（IdNo）、余额（Balance）、有效状态(Stat)

```
Imports System
Imports System.Web
Imports System.Web.UI

Namespace MyNamespace
‘定义一个枚举变量，0—正常  1—销户  2—其他状态(挂失、冻结等等)
    Public Enum Status
        Active = 0
        Deactive = 1
        Other = 2
     End Enum

    Public Class Account : Inherits Control

        Private _AcctNo As String
        Private _IdNo As String
        Private _Balance As Currency
        Private _Stat As Status

        Public Property AcctNo As String
            Get
                Return _AcctNo
            End Get
            Set
                _AcctNo = Value
            End Set
        End Property

        Public Property IdNo As String
            Get
                Return _IdNo
            End Get
            Set
                _IdNo = Value
            End Set
        End Property

        Public Property Balance As Currency
            Get
                Return _Balance
            End Get
            Set
                _Balance = Value
            End Set
```

```
        End Property

    Public Property    Stat    As    Status
                Get
                    Return _Stat
                End Get
                Set
                    _Stat = Value
                End Set
            End Property
    …
        End Class

    End Namespace
```

而方法的定义就比较灵活，可以根据设计要求，提供相应的功能，例如大多数的类一般都会提供创建或者是初始化类的方法。我们仍以上面的帐号类为例，定义一个 New 方法：

```
。。。
Public Sub New(AcctNo1 As String,IdNo1 As String,Balance1 As Currency,Stat1 As Status)
MyBase.New
Me.AcctNo = AcctNo1
Me.IdNo = IdNo1
Me.Balance = Balance1
Me.Stat = Stat1
End Sub
。。。
```

3.定义自己应用界面

一个用户自定义的控件一般来说较为复杂，由至少一个以上的内置控件构成，这时就需要重载从 Control 类继承来的 CreateChildControls 方法，并在其中生成界面控件。如果用户定义的控件会在一个页面中反复使用，最好 implements System.Web.UI.INamingContainer，它会为该控件创建一个唯一的命名空间。

例如：下面的例子将创建一个控件，它由一段说明文字和一个文本输入框构成。

```
Imports   System
Imports    System.Web
Imports    System.Web.UI
Imports    System.Web.UI.WebControls

Namespace MyNamespace
Public Class Myclass :Inherits Control : Implements INamingContainer
…
Protected Overrides Sub CreateChildControls()
```

```
            Me.Controls.Add(New LiteralControl("<h3>请输入: "))

            Dim txtBox As New TextBox
            txtBox.Text = ""
            Me.Controls.Add(txtBox)

            Me.Controls.Add(New LiteralControl("</h3>"))
        End Sub
…
End Class
End Namespace
```

4．定义自己控件的消息处理函数。

自己定义的控件含有两种类型的消息，一是包含的子控件所产生的消息，二是自定义的控件消息。

子控件产生的消息处理函数可由 AddHandler 函数来指定，其用法如下：

AddHandler 子控件.消息，AddressOf 消息处理函数

例如：自定义控件中含有一个 Button 控件，并定义其处理函数 MyBtn_Click()

```
 …
 Private Sub MyBtn_Click(Sender as Objects, E as    EventArgs)
 …
 End Sub

Protected override Sub CreateChildControls()
…
Dim MyBtn    As    New    Button
MyBtn.text=””
AddHandler    MyBtn.Click , AddressOf    MyBtn_Click
Me.Controls.Add(MyBtn)
…
End Sub
```

自定义的控件消息则需要先定义事件说明,格式如下

Public Event 消息名(Sender as Object,E as EventArgs)

例如：Public Event Change(Sender as Object,E as EventArgs)

然后定义事件发出函数,例如：

```
Protected Sub OnChange(E as EventArgs)
  RaiseEvent Change(Me,E)
End Sub
```

再然后定义引起事件发生的过程（可不写）

例如：

```
        Private Sub TextBox_Change(Sender As Object, E As EventArgs)
            OnChange(EventArgs.Empty)
```

```
        End Sub
```

最后定义何时触发事件函数，同样使用 AddHandler 函数

例如：

```
    …
    Protected override Sub CreateChildControls()
…
    Dim MyBox as New TextBox
    MyBox.Text=””
    AddHandler MyBox.TextChanged , AddressOf TextBox_Change
    Me.Controls.Add(MyBox)
…
    End Sub
    …
```

5．最后，谈一谈继承控件的使用，首先应把预先写好的继承控件编译成.DLL 文件

编译格式为：

vbc　/t:library /out:MyDll.dll /r:System.Web.dll MyVb.vb

vbc 为 vb.net 的编译器

/t:表示编译类型，library 为链接库，exe 为独立可执行文件

/out:指定输出文件名

/r:表示需要引用的 DLL 文件

MyVb.vb:指自己编写的继承控件 vb 源程序

然后，为在自己的页面中引用自己定义的控件，需在 aspx 文件头进行注册，

```
<%@ Register TagPrefix="标记前缀" Namespace="命名控件" %>
```

最后，就如同使用内置控件一样，在页面中使用自己定义的控件：

```
<命名空间名：类名 ……　runat=server />
```

下面举一个具体的例子来说明：

我们仍然以开始定义的用户帐号为例来定义一个继承控件，该类有 4 个属性分别为客户帐号、身份证号、帐户余额、帐户状态，其用户界面设定为 4 个文本框供输入属性值以供修改，另外加 2 个按钮以供确认，同时为该控件设定一个事件 Click,当按下确认键后，修改控件属性值，并且在页面中显示自定义控件的属性值，以确认事件确实生效了。

1．控件定义文件

```
'文件名：form\CustomControl\Inherit.vb
Option Strict Off

Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls
```

```
Namespace MyNamespace

   Public Enum Status
            Active      = 0
            Deactive = 1
            Other       = 2
   End Enum

Public Class MyAccount:Inherits Control:Implements INamingContainer
'从 Control 类继承，并且有自己的命名空间
   Private _AcctNo As String
   Private _IdNo As String
   Private _Balance As Decimal
   Private _Stat As Status

   Public Event Click(Sender as Object,E as EventArgs)
'定义控件自身的 Click 事件

'对属性存取的定义
        Public Property AcctNo As String
            Get
                Return _AcctNo
            End Get
            Set
                _AcctNo = Value
            End Set
        End Property

        Public Property IdNo As String
            Get
                Return _IdNo
            End Get
            Set
                _IdNo = Value
            End Set
        End Property

        Public Property Balance As Decimal
            Get
                Return _Balance
            End Get
            Set
                _Balance = Value
            End Set
```

```
    End Property

    Public Property   Stat   As   Status
            Get
                Return _Stat
            End Get
            Set
                _Stat = Value
            End Set
        End Property

Public Sub New()
    MyBase.New
    Me.AcctNo = ""
    Me.IdNo = ""
    Me.Balance = "0.0"
    Me.Stat = "0"
End Sub

Protected Sub OnClick(E as EventArgs)
    RaiseEvent Click(Me,E)
End Sub



'界面定义为 4 个属性文本输入框，加一个确定和取消键
Protected Overrides Sub CreateChildControls()
    Me.Controls.Add(New LiteralControl("<h3>客户帐号："))
    dim txtAcctNo as New TextBox
    txtAcctNo.text=_AcctNo
    Me.Controls.Add(txtAcctNo)

    Me.Controls.Add(New LiteralControl("<br>身份证号："))
    dim txtIdNo   as New TextBox
    txtIdNo.text=_IdNo
    Me.Controls.Add(txtIdNo)

    Me.Controls.Add(New LiteralControl("<br>帐户余额："))
    dim txtBalance as New TextBox
    txtBalance.text=_Balance
    Me.Controls.Add(txtBalance)

    Me.Controls.Add(New LiteralControl("<br>帐户状态："))
    dim txtStat     as New TextBox
```

```
    txtStat.text=_Stat
    Me.Controls.Add(txtStat)

    Me.Controls.Add(New LiteralControl("<br><br><br>"))
    dim Btn1 as New Button
    Btn1.text="确 认"
    AddHandler Btn1.Click,AddressOf Btn1_Click
    Me.Controls.Add(Btn1)

    dim Btn2 as New Button
    Btn2.text="取 消"
    Me.Controls.Add(Btn2)

    Me.Controls.Add(New LiteralControl("</h3>"))
End Sub

Private Sub Btn1_Click(Sender as Object,E as EventArgs)
dim ctrl1 as TextBox=controls(1)
Me.AcctNo=ctrl1.text
dim ctrl2 as TextBox=controls(3)
Me.IdNo=ctrl2.text
dim ctrl3 as TextBox=controls(5)
Me.Balance=CDbl(ctrl3.text)
dim ctrl4 as TextBox=controls(7)
Me.Stat=Cint(ctrl4.text)
OnClick(E)
End Sub

End Class

End Namespace
```

2． 控件编译文件

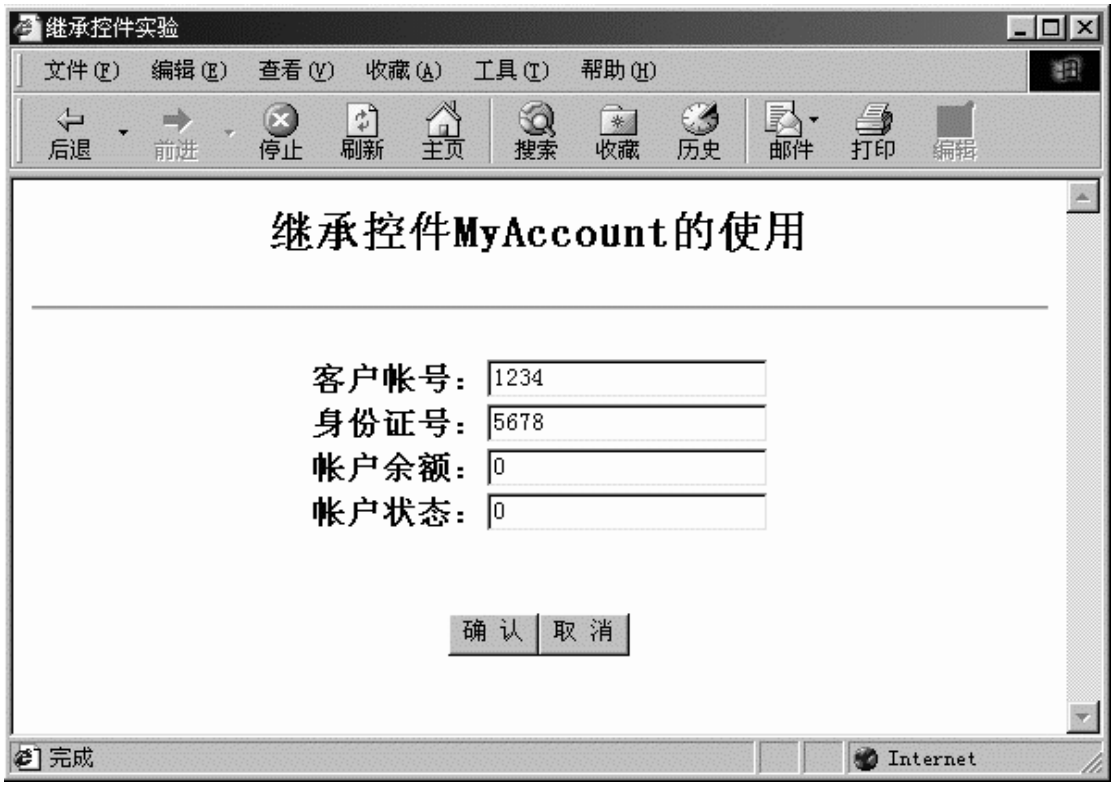
```
rem inherit.vb 的编译文件 文件名：form\CustomControl\i.bat
vbc    /t:library /out:.\bin\MyNamespaceVB.dll /r:System.Web.dll inherit.vb
```

3. 页面使用文件

```
<!--源文件：form\CustomControl\FormInherit.aspx-->
<%@ Register TagPrefix="MyNamespace" Namespace="MyNamespace" %>
<html>
<script language="vb" runat=server>
```

```
sub acct_click(s as object, e as eventargs)
dim strTxt as string
strTxt="<hr>AcctNo="& acct1.AcctNo & "<br>"
strTxt=strTxt & "IdNo=" & acct1.IdNo & "<br>"
strTxt=strTxt & "Balance=" & acct1.Balance & "<br>"
strTxt=strTxt & "Stat=" & acct1.Stat
response.write(strTxt)
end sub
</script>
<head>
<title>
继承控件实验
</title>
</head>

<body bgcolor=#ccccff>
<center>
<h2>继承控件 MyAccount 的使用</h2>
<hr>
<br>
<form action="forminherit.aspx" method="post" runat=server>
<MyNamespace:MyAccount id="acct1" AcctNo="1234" IdNo="5678" OnClick="acct_click"
runat=server />
</form>
</center>
</body>
</html>
```

4.开始的输出画面：



5.修改后，按确认键后的效果：

2.3.6 小结

本章在前一章学习了服务器端控件的基础上，探讨了如何在利用已有控件的基础上，开发具有自己特色的自定义控件。使用自定义控件的好处在于：

1. 简化了自己程序开发的周期
2. 有助于形成具有自己特色的风格
3. 隔离了错误发生的根源，修改自己定义的控件时，不必考虑其他因素

第四章 HTML 控件

HTML 控件在服务器端是可见的，所以我们可以根据它来按照我们的意愿来编写。HTML 控件表现为一些可见的控件。

2.4.1 HtmlButton

HtmlButton server control 就象 HTML4.0 中的<button>一样，但是这与<Input type=”button”>不一样的，我们看下面的例子 button.aspx：

响应按钮事件：

```
<script language="VB" runat="server">
    Sub Button1_OnClick(sender As Object, e As EventArgs)
        Span1.InnerHtml="你点击了 Button1"
    End Sub
    Sub Button2_OnClick(sender As Object, e As EventArgs)
        Span1.InnerHtml="你点击了 Button2"
    End Sub
</script>
```

对两个 button 的描述：

button1：

```
<button id="Button1" onServerClick="Button1_OnClick" style="font: 8pt
        verdana;background-color:lightgreen;border-color:black;height=30;width:1
        00" runat="server">
    <img src="/quickstart/aspplus/images/right4.gif"> Click me!
</button>
```

button2，我们增加了鼠标事件：

```
<button id=Button2 onServerClick="Button2_OnClick"   style="font: 8pt
    verdana;background-color:lightgreen;border-color:black;height=30;width:100"
    onmouseover="this.style.backgroundColor='yellow'"
    onmouseout="this.style.backgroundColor='lightgreen'"
    runat="server">
```

Click me too!
</button>

点击 button2，并把鼠标移到它的上面：

第二节 HtmlForm

一个 HtmlForm Control 必须要处理 PostBack 请求，一个 Web Form 只有一个<form>标记。**Form.aspx** 中 form 的表示：

```
<form id=ServerForm runat=server>
  <button id=Button1 runat="server"
          onServerClick="Button1_OnClick">Button1</button>

    
  <span id=Span1 runat=server />
  <p>
  <button id=Button2 runat="server"
          onServerClick="Button2_OnClick">Button2</button>
    
  <span id=Span2 runat=server />
</form>
```

响应鼠标按钮事件：

```
Sub Button1_OnClick(sender As Object, e As EventArgs)
    Span1.InnerHtml = "It is Button1"
End Sub
Sub Button2_OnClick(sender As Object, e As EventArgs)
    Span2.InnerHtml = "It is Button2"
End Sub
```

看如下结果：

点击两个按钮，同时显示信息：

2.4.2 HtmlImages

我们通过一个<img src="">这个标记来显示图片：

```
<img ID="Image1" src="images/4.gif" runat="server"/>
```

我们根据 ID 号为<img >提供图片来源：

```
Sub SubmitBtn_Click(sender As Object, e As EventArgs)
    Image1.Src="images/" & Select1.Value
End Sub
```

建立一个选择控件来与用户的交互：

选择面部表情文件:

```
<select id="Select1" runat="server">
    <option Value="4.gif">4</option>
```

```
        <option Value="5.gif">5</option>
        <option Value="6.gif">6</option>
        <option Value="7.gif">7</option>
        <option Value="8.gif">8</option>
        <option Value="9.gif">9</option>
    </select>
    <input type="submit" runat="server" Value="提交"
        OnServerClick="SubmitBtn_Click">
```

我们运行如下：

选择相应的文件号，点击按钮，图片就显示出来。

2.4.3 TextArea

象在 HTML 中的一样，在 asp.net 中的 TextArea 也是一个多行输入框。TextArea 的宽度由他的 Cols 属性决定，长度由 Rows 属性决定。

Textarea.aspx 中定义输入：

```
<textarea id="TextArea1" cols=40 rows=4 runat=server />
```

我们用 TextArea1.Value 取得输入的值。具体如下(**textarea.aspx**)：

```
<!--源文件： form\HtmlControl\textarea.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub SubmitBtn_Click(sender As Object, e As EventArgs)
            Span1.InnerHtml = "下面是你所写的 :<br>" & TextArea1.Value
        End Sub
    </script>
</head>
<body bgcolor="#ccccff">
    <h3><font face="Verdana">.NET->HtmlTextArea</font></h3>
    <form runat=server>
      <font face="Verdana" size="-1">
        写吧: <br>
        <textarea id="TextArea1" cols=40 rows=4 runat=server />
        <input type=submit value="Submit" OnServerClick="SubmitBtn_Click" runat=server>

        <p>
        <span id="Span1" runat="server" />
      </font>
    </form>
</body>
</html>
```

2.4.4 InputHidden

我们可以用隐藏输入控件来处理一些我们要传送而又不想在页面上显示出来的信息，例如在电子商务网站中，我们向银行网关接口传送我们的订单信息，我们就可以用隐藏输入控件来处理。

我们下面的例子用不可见的值来取得输入值，再把不可见值显示出来。

Inputhidden.aspx 隐藏输入控件：

```
<input id="HiddenValue" type=hidden value="隐藏的字符" runat=server>
```

初始值为“隐藏的字符“，在我们第一次点击按钮时候显示出来，我们的方法：

```
    Sub SubmitBtn_Click(sender As Object, e As EventArgs)

        HiddenValue.Value = StringContents.Value
    End Sub
```

这个方法把输入值赋给不可见的控件。我们完整的代码如下(**hidden.aspx**)：

```
<!--源文件： form\HtmlControl\hidden.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub Page_Load(sender As Object, e As EventArgs)
            If IsPostBack Then
                Span1.InnerHtml="隐藏值: <b>" & HiddenValue.Value & "</b>"
            End If
        End Sub
        Sub SubmitBtn_Click(sender As Object, e As EventArgs)
            HiddenValue.Value = StringContents.Value
        End Sub
    </script>
</head>
<body>
    <h3><font face="Verdana">.NET->HtmlInputHidden</font></h3>
    <form runat=server>
        <input id="HiddenValue" type=hidden value="隐藏的字符" runat=server>
        请输入: <input id="StringContents" type=text size=40 runat=server>
        <p>
        <input type=submit value="确定" OnServerClick="SubmitBtn_Click" runat=server>
        <p>
        <span id=Span1 runat=server>显示隐藏的字符</span>
```

```
    </form>
</body>
</html>
```

我们输入 InputHidden，便点击按钮，则显示出默认的隐藏值。

2.4.5 HtmlTable

HtmlTable 服务控件能让你轻松的创建你的表格的行和列，也可以按照程序的模式自动生成表格。

我们的例子展示了这个特性：

```
<table id="Table1" CellPadding=4 CellSpacing=0 Border="1" runat="server" />
```

这就是在 asp.net 中，表格的表示。做两个 Select 控件来让用户选择表格的属性：

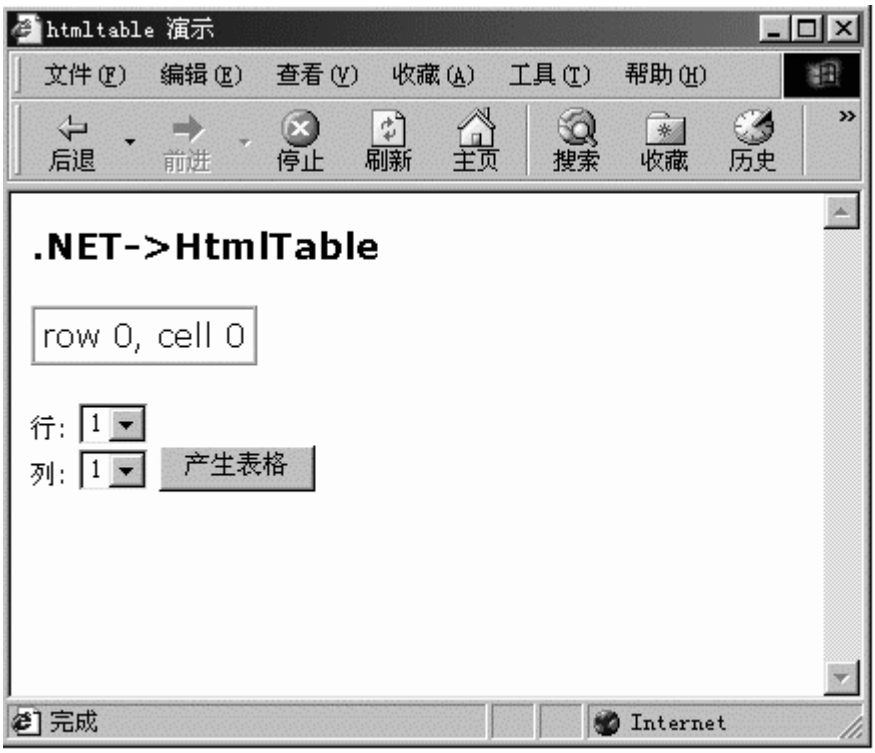
```
    <p>
      行:
      <select id="Select1" runat="server">
```

```
            <option Value="1">1</option>
            <option Value="2">2</option>
            <option Value="3">3</option>
            <option Value="4">4</option>
        </select>
        <br>
        列:
        <select id="Select2" runat="server">
            <option Value="1">1</option>
            <option Value="2">2</option>
            <option Value="3">3</option>
            <option Value="4">4</option>
        </select>
```

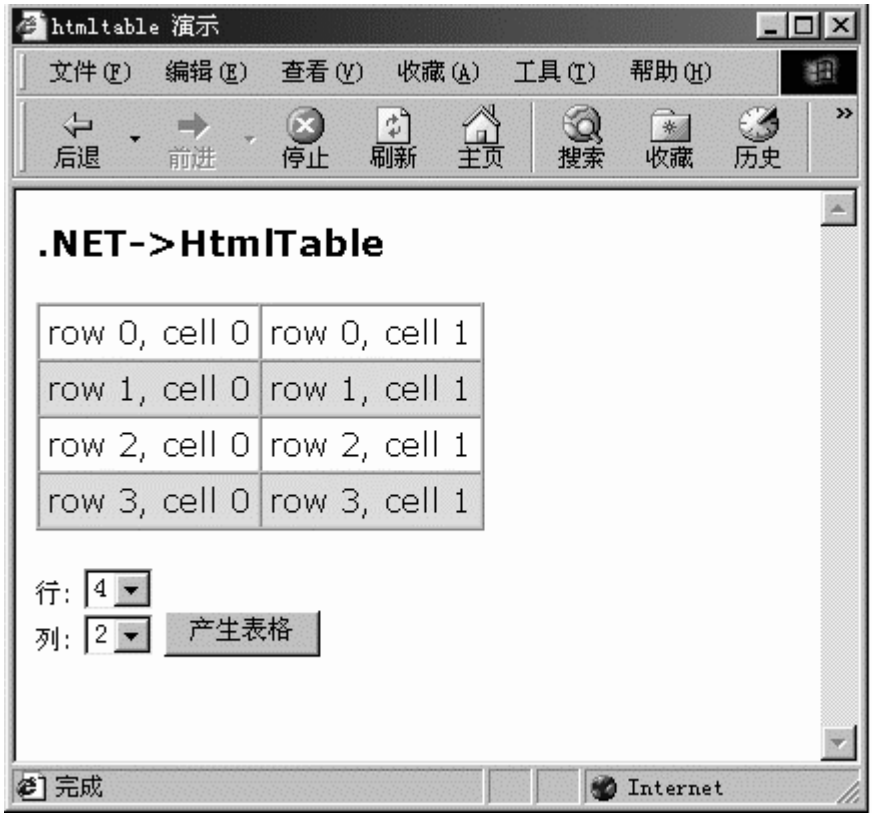
在用户提交的时候，实际上我们示对页面进行了刷新，即在 Page_Load 方法里面处理，具体如下(**htmltable.aspx**)：

```
<!--源文件： form\HtmlControl\htmltable.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub Page_Load(sender As Object, e As EventArgs)
            Dim numrows As Integer
            Dim numcells As Integer
            Dim i As Integer = 0
            Dim j As Integer = 0
            Dim Row As Integer = 0
            Dim r As HtmlTableRow
            Dim c As HtmlTableCell

            ' 产生表格
            numrows = CInt(Select1.Value)
            numcells = CInt(Select2.Value)
            For j = 0 To numrows-1
                r = new HtmlTableRow()
                If (row Mod 2 <> 0) Then
                    r.BgColor = "Gainsboro"
                End If
                row += 1
                  For i = 0 To numcells-1
                    c = new HtmlTableCell()
                    c.Controls.Add(new LiteralControl("row " & j & ", cell " & i))
                    r.Cells.Add(c)
                Next i
                   Table1.Rows.Add(r)
            Next j
```

```
        End Sub
    </script>
</head>
<body>
    <h3><font face="Verdana">.NET->HtmlTable</font></h3>
    <form runat=server>
    <font face="Verdana" size="-1">
        <p>
        <table id="Table1" CellPadding=4 CellSpacing=0 Border="1" runat="server" />
        <p>
        行:
        <select id="Select1" runat="server">
            <option Value="1">1</option>
            <option Value="2">2</option>
            <option Value="3">3</option>
            <option Value="4">4</option>
        </select>
        <br>
        列:
        <select id="Select2" runat="server">
            <option Value="1">1</option>
            <option Value="2">2</option>
            <option Value="3">3</option>
            <option Value="4">4</option>
        </select>
          <input type="submit" value="产生表格" runat="server">
    </font>
    </form>
</body>
</html>
```



选择并提交，我们的表格就出来了：



2.4.6 HtmlGenericControl

HtmlGenericControl 提供一个服务器控件，用来执行那些不直接的表现出来的未知的 Html Control 标识。

例子文件 **Gerecolor.aspx**：

```
<!--源文件： form\HtmlControl\Gerecolor.aspx-->
<html>
<head>
    <script language="VB" runat="server">
    Sub SubmitBtn_Click(sender As Object, e As EventArgs)
        Body.Attributes("bgcolor") = ColorSelect.Value
    End Sub
    </script>
</head>
<body id=Body runat=server>
    <h3><font face="Verdana">.NET->HtmlGenericControl</font></h3>
    <form runat=server>
      <p>
      Select a background color for the page: <p>
      <select id="ColorSelect" runat="server">
          <option>White</option>
          <option>Wheat</option>
          <option>Gainsboro</option>
          <option>LemonChiffon</option>
      </select>
      <input type="submit" runat="server" Value="Apply" OnServerClick="SubmitBtn_Click">
    </form>
</body>
</html>
```

我们的运行结果如下：

选择你所要的颜色，则面背景颜色就会改变。

2.4.7 HtmlInputButton

其实我们在上面的应用的时候就大概的应用过这个控件的，现在我们专门来讲讲它。这个控件有几个功能，可以是普通的按钮来响应一般的事件；可以是 Submit 按钮；也可以是 Reset 按钮。

2.4.7.1 一般性的按钮

这个控件不是响应表单中通常的 Submit 或者 Reset 事件的，而是响应我们为他定制的的事件(**button.aspx**)。

```
<!--源文件： form\HtmlControl\button.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub Button1_Click(sender As Object, e As EventArgs)
            Span1.InnerHtml = "你点击了这个按钮！"
        End Sub
    </script>
</head>
    <h3><font face="Verdana">.NET->HtmlInputButton->button</font></h3>
    <form runat=server>
        <p>
```

```
        <input type=button value="Button1" onServerClick="Button1_Click" runat="server">
          
        <span id=Span1 runat=server />
    </form>
</body>
</html>
```

点击按钮，如下：

2.4.7.2 Submit 与 Reset

对这两个属性大家肯定很熟悉的了，在这里我们简单的介绍一下。在 asp.net 中，对这两个按钮，我们也用到了<input type= 这个标示，只是我们在后面加上 runat=”server“，表示我们这个按钮在 asp.net 的框架之内，我们必须写方法来响应这个事件。如果没有 runat=”server“这个修饰，我们可以把这个控件当作普通的 html 按钮，不用写法响应事件，如：

```
<input type=reset value="重写" >
```

这是一个标准的 html 表示，我们在 asp.net 中的表示：

```
<input type=submit value="提交" OnServerClick="SubmitBtn_Click" runat=server>
```

我们在按钮按下时候响应事件 SubmitBtn_Click，具体如下**(button2.aspx)**：

```
<!--源文件： form\HtmlControl\button2.aspx-->
<html>
<head>
    <script language="VB" runat="server">
        Sub SubmitBtn_Click(sender As Object, e As EventArgs)
          If Password.Value = "saidy2001" Then
              Span1.InnerHtml = "密码正确！"
          Else
              Span1.InnerHtml="密码错误！"
          End If
        End Sub
        ' Sub ResetBtn_Click(sender As Object, e As EventArgs)
        '     Name.Value = ""
        '     Password.Value = ""
        'End Sub

    </script>
</head>
<BODY BGCOLOR="#CCCCFF">
    <h3><font face="Verdana">.NET->Submit and Reset</font></h3>
    <form runat=server>
        输入名字: <input id="Name" type=text size=40 runat=server>
        <p>
        输入密码: <input id="Password" type=password size=40 runat=server> (密码是：
                  saidy2001)
        <p>
        <input type=submit value="提交" OnServerClick="SubmitBtn_Click" runat=server>
        <input type=reset value="重写" OnServerClick="ResetBtn_Click" runat=server>
        <p>
        <span id="Span1" style="color:red" runat=server></span>
    </form>
</body>
</html>
```

看到如下的效果：

输入名字和提示的密码，如下：

2.4.8 小结

本章介绍了服务器端的 html 控件，虽然它们的功能都可以以简单的 html 语言来实现，但是在 asp.net 中依然提供了对它们的实现。以 html 语言书写和以服务器端控件的实现在思

维方式上已经有了很大的不同，对于 html 语言而言，只是一种标识；而对服务器端 html 控件而言，却已演变成为一段程序，一个对象。两者的区别不仅仅是，一个后缀名为.html，另一个为.aspx。html 文件依赖于服务器端对标识的解释执行，html 控件却可以被编译执行，两者在效率上的差异不言而喻。