第 1 章 网页指令

1.1 @ Assembly指令——程序指令

@Assembly 指令用于在编译时将程序集链接到页面，这使得开发人员可以使用程序集公开的所有类和方法等。@Assembly 指令可以在.aspx 页、.ascx 文件、.master 页和.asax 文件中使用。

 语法

```
//第一种
<%@ Assembly Name="assemblyname"%>
//第二种
<%@ Assembly Src ="pathname"%>
```

关于上述@ Assembly 指令语法中各属性的说明如表 1.1 所示。

表 1.1 @ Assembly 指令的属性说明

属　性	描　述
Name	指定编译页面时链接的程序集
Src	指定要动态编译并链接到当前页面

必须在@Assembly 指令中包含 Name 或 Src 属性，但不能在同一个指令中包含两者。如果需要同时使用这两个属性，则必须在文件中包含多个@Assembly 指令。

在链接 Web 应用程序的 Bin 目录中的程序集时，将自动链接到该应用程序中的 ASP.NET 文件。这样的程序集不需要@Assembly 指令。用户若要禁用此功能，可将应用程序中 Web.config 文件内的<assemblies>节移除下面一行：

```
<add assembly="*"/>
```

 示例 1　使用@Assembly 指令链接到用户定义的程序集 MyAssembly。

```
<%@Assembly Name=MyAssembly%>
```

 示例 2　使用@Assembly 指令链接 Visual Basic 源文件 MySource.vb。

```
<%@Assembly Src =MySource.vb%>
```

1.2 @ Control指令——控制指令

@ Control 指令与@Page 指令基本相似，在.aspx 文件中包含@Page 指令，而在.ascx 文件中则不包含@Page 指令，但包含@Control 指令。该指令只能用于用户控件中。用户控件在带有.ascx 扩展名的文件中进行定义。每个.ascx 文件只能包含一条@Control 指令。此外，对于每个@Control 指令，只允许定义一个 Language 属性，因为每个控件只能使用一种语言。

> **注意** @Control 指令与适用于整个源文件的其他指令有许多共同的属性，例如稍后将做讲解的@Page 指令（在网页的.aspx 文件中使用）和@Master 指令（在母版页的.master 文件中使用）。

语法

```
<%@ Control attribute="value" [attribute="value" ... ] %>
```

其中，attribute 包含@ Control 指令所有的属性，其常用属性如表 1.2 所示。

表 1.2 @ Control 指令属性说明

属 性	描 述
AutoEventWireup	设置控件的事件是否自动匹配。如果启用事件自动匹配，则为 True；否则为 False。默认值为 True
ClassName	一个字符串，用于指定需在请求时进行动态编译的控件的类名，此值可以是任何有效的类名，并且可以包括类的完整命名空间（即一个完全限定的类名）；如果没有为此属性指定值，则已编译控件的类名将基于该控件的文件名
CodeFile	指定所引用的控件代码隐藏文件的路径。此属性与 Inherits 属性一起使用，将代码隐藏源文件与用户控件相关联。该属性只对已编译控件有效
Debug	指示是否应使用调试符号编译控件。如果应使用调试符号编译控件，则为 True；否则为 False。由于此设置将影响性能，因此应该只在开发期间将该属性设置为 True
EnableTheming	指示控件是否使用主题。如果使用主题，则为 True；否则为 False。默认值为 True
EnableViewState	指示是否跨控件请求维护视图状态。如果维护视图状态，则为 True；否则为 False。默认值为 True
Inherits	定义供控件继承的代码隐藏类。它可以是从 UserControl 类派生的任何类。与包含代码隐藏类源文件的路径的 CodeFile 属性一起使用
Language	指定在编译控件中所有内联呈现（<% %>和<%= %>）和代码声明块时使用的语言。值可以表示任何.NET Framework 支持的语言，包括 Visual Basic、C#或 JavaScript。对于每个控件，只能使用和指定一种语言
Src	指定包含链接到控件的代码的源文件路径。在所链接的源文件中，可选择在类中或在代码声明块中包括控件的编程逻辑。可以使用 Src 属性将生成提供程序链接到控件。在 ASP.NET 中，将代码隐藏源文件链接到控件的首选方法是，使用 Inherits 属性指定一个类，并使用 CodeFile 属性指定该类的源文件的路径

示例 当新添加一个.ascx 页时，在页面中@Control 指令的默认代码如下，并显示设置 EnableViewState 属性为 false 来禁止跨 HTTP 请求保存视图状态。代码如下：

```
<% @ Control Language="C#" AutoEvent="true" CodeFile="rmgwh.ascx.cs" Inherits=
"Controls_mrgwh"  EnableViewState="false" %>
```

典型应用 本示例主要利用 Web 用户控件制作一个留言本导航条，其运行效果如图 1.1 所示。

图 1.1 利用 Web 用户控件制作留言本导航条

说明 创建好用户控件后，必须添加到其他 Web 页中才能显示出来，不能直接作为一个网页来显示，因此也就不能设置用户控件为“起始页”。

实现本示例的源代码如下：

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="header.ascx.cs" Inherits=
"header" %>
<table style="width: 675; height: 189px;">
    <tr>
        <td colspan="7" style="height: 105px; text-align: center">
            <asp:Image ID="Image1" runat="server" ImageUrl="~/image/banner.jpg" /></td>
    </tr>
    <tr>
        <td colspan="7" style="height: 37px; text-align: center">
            <table style="width: 81%; height: 25px; border-top-width: thin; border-
           left-width: thin; border-left-color:
#ffcc99; border-bottom-width: thin; border-bottom-color: #ffcc99; border-top-color:
#ffcc99; border-right-width: thin;
border-right-color: #ffcc99;">
                <tr>
                    <td style="width: 100px; padding-right: 0px; padding-left: 0px;
                    padding-bottom: 0px; margin: 0px;
padding-top: 0px; height: 36px;">
                        <asp:Image ID="Image2" runat="server" ImageUrl="~/image/s1.jpg"
NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 100px; padding-right: 0px; padding-left: 0px;
                    padding-bottom: 0px; margin: 0px;
 padding-top: 0px; height: 36px;">
                        <asp:ImageButton ID="ImgBtnShouYe" runat="server" ImageUrl=
                       "~/image/s2.jpg"
NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 93px; padding-right: 0px; padding-left: 0px;
                    padding-bottom: 0px; margin: 0px;
padding-top: 0px; height: 36px;">
                        <asp:ImageButton ID="ImgBtnSelect" runat="server" ImageUrl=
                       "~/image/s3.jpg"
NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 86px; padding-right: 0px; padding-left: 0px;
                    padding-bottom: 0px; margin: 0px;
padding-top: 0px; height: 36px;">
                        <asp:ImageButton ID="ImgBtnAdd" runat="server" ImageUrl=
                       "~/image/s4.jpg"
NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 87px; padding-right: 0px; padding-left: 0px; padding-
                    bottom: 0px; margin: 0px;
```

```
    padding-top: 0px; height: 36px;">
                        <asp:ImageButton ID="ImgBtnDelete" runat="server" ImageUrl=
                        "~/image/s5.jpg"
    NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 100px; padding-right: 0px; padding-left: 0px;
                    padding-bottom: 0px; margin: 0px;
    padding-top: 0px; height: 36px;">
                        <asp:ImageButton ID="ImgBtnLogin" runat="server" ImageUrl=
                        "~/image/s6.jpg"
    NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 100px; height: 36px; padding-right: 0px; padding-
                    left: 0px;
    padding-bottom: 0px; margin: 0px; padding-top: 0px;">
                        <asp:ImageButton ID="ImgBtnZhuce" runat="server" ImageUrl=
                        "~/image/s7.jpg"
    NavigateUrl="~/Default.aspx" CausesValidation="False" /></td>
                    <td style="width: 100px; height: 36px; padding-right: 0px; padding-
                    left: 0px;
    padding-bottom: 0px; margin: 0px; padding-top: 0px;">
                        <asp:Image ID="Image3" runat="server" ImageUrl="~/image/s8.
                        jpg" /></td>
                </tr>
            </table>
        </td>
    </tr>
</table>
```

1.3　@ Implements指令——接口指令

@ Implements 指令指示当前的 ASP.NET 应用程序文件（网页、用户控件或母版页）实现指定的.NET Framework 接口。

语法

```
<%@ Implements interface="ValidInterfaceName" %>
```

其中 interface 属性用来指定要在页或用户控件中实现的接口。

在 Web 窗体页中实现接口时，开发人员可以在代码声明块中的<script>元素的开始标记和结束标记之间创建其事件、方法和属性，但不能使用该指令在代码隐藏文件中实现接口。

示例　本示例演示了一个用户控件，它包含一个用于访问 IWebPart 接口属性的 @ Implements 指令，源代码如下：

```
<%@Control language="C#" ClassName="CalendarUserControl" %>
<%@ implements
    interface="System.Web.UI.WebControls.WebParts.IWebPart" %>
```

1.4　@ Import指令——导入指令

@Import 指令用于将命名空间显式导入到 ASP.NET 应用程序文件中，并且导入该命名空间的所有类和接口。导入的命名空间可以是.NET Framework 类库的一部分，也可以是用户定义的命名空间的一部分。

语法
```
<%@ Import namespace="value" %>
```

其中，namespace 属性用来指定要导入的命名空间的完全限定名，它可以包括任何包含在.NET Framework 中的命名空间或自定义命名空间。

@Import 指令不能有多个 namespace 属性。若要导入多个命名空间，请使用多条@Import 指令。

在 ASP.NET 中，已命名空间是默认导入的，默认导入的空间如下：

- System
- System.Collections
- System.Collections.Specialized
- System.Configuration
- System.Text
- System.Text.RegularExpressions
- System.Web
- System.Web.Caching
- System.Web.Profile
- System.Web.Security
- System.Web.SessionState
- System.Web.UI
- System.Web.UI.HtmlControls
- System.Web.UI.WebControls
- System.Web.UI.WebControls.WebParts

示例 下面的代码示例为导入.NET Framework 基类命名空间 System.Net 和用户定义的命名空间 mrbccd。源代码如下：

```
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace=" mrbccd " %>
```

1.5 @ Master指令——母版页指令

@Master 指令只能在母版页的.master 文件中使用，用于标识 ASP.NET 母版页。每个.master 文件只能包含一条@Master 指令。

语法
```
<%@ Master attribute="value" [attribute="value"...] %>
```

其中，attribute 表示@ Master 指令中的各属性，@ Master 指令属性说明如表 1.3 所示。

表 1.3 @ Master 指令属性说明

属 性	描 述
AutoEventWireup	指示是否可以使用语法 Page 且不使用任何显式挂钩或事件签名，为特定的生命周期阶段定义简单的事件处理程序。如果启用了事件自动连接，则为 True；否则为 False。默认值为 True

（续）

属　性	描　述
ClassName	指定自动从标记生成并在处理母版页时自动进行编译的类的类名。此值可以是任何有效的类名，并且可以包括命名空间
CodeFile	指定包含分部类的单独文件的名称，该分部类具有事件处理程序和特定于母版页的其他代码
CompilationMode	指定是否在运行时编译 ASP.NET 母版页。选项包括：Always 表示始终编译页；Auto 在 ASP.NET 要避免编译页（如有可能）的情况下使用；Never 表示永远不编译页或控件。默认值为 Always
EnableTheming	指示在应用主题时是否可以修改母版页的外观和母版页上控件的外观。如果可以应用主题，则为 True；否则为 False。默认值为 True。设置 EnableTheming 属性主要用于以下情况：默认情况下在 Web.config 文件中定义了页主题，并且将该页主题应用于所有页
EnableViewState	指示是否在页请求之间保持视图状态。如果要保持视图状态，则为 True；否则为 False。默认值为 True
Inherits	指定供页继承的代码隐藏类。它可以是从 MasterPage 类派生的任何类
Language	指定在对页中所有内联呈现（<% %> 和 <%= %>）和代码声明块进行编译时使用的语言。值可以表示.NET Framework 支持的任何语言，包括 VB（Visual Basic）、C#和 JavaScript
MasterPageFile	指定用做某个母版页的.master 文件。定义嵌套母版页方案中的子母版页时，在母版页中使用 MasterPageFile 属性
Src	指定在请求页时动态编译的代码隐藏类的源文件名称。可以选择将页的编程逻辑包含在代码隐藏类中或.aspx 文件的代码声明块中

示例 本示例以 C#作为内联代码语言，事件处理代码在名为 MasterPage.master.cs 的分部类中定义。可以在 MasterPage.master.cs.文件中找到 MasterPage 类的代码，示例的源代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>
```

除了上述@Master 指令的典型应用示例外，在@Master 指令中 MasterPageFile 属性也非常重要，该属性用来指定用做某个母版页的.master 文件。在定义嵌套母版页方案中的子母版页时，在母版页中使用 MasterPageFile 属性。

以下这个示例先给出了父母版页 MasterPage.master，然后在子母版页 SubMasterPage.master.中应用 MasterPageFile 属性指定应用这个给定的父母版页 MasterPage.master。

父母版页 MasterPage.master 声明的主要源代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <table background="image/博客首页面.JPG" style="width: 994px; height: 527px">
```

```
        <tr>
          …//省去系统部分自动生成的代码
            <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
           </asp:contentplaceholder>
          </td>
         <td style="width: 117px; height: 127px">
        …//部分代码省去
      </table>
      </div>
    </form>
    </body>
    </html>
```

在子母版页 SubMasterPage.master 中应用 MasterPageFile 来指向所应用的父母版页 MasterPage.master，源代码如下：

```
    <%@ Master Language="C#" AutoEventWireup="true" CodeFile="SubMasterPage.master.cs"
       Inherits="SubMasterPage" MasterPageFile="~/MasterPage.master" %>
    <asp:Content ID=content1 runat=server ContentPlaceHolderID=ContentPlaceHolder1>
    <table bgcolor=green>
      <tr>
        <td width=170 >
          <h1>
           …//省略子母版页添加的内容
          </h1>
        </td>
        <td>
          <asp:contentplaceholder id="ContentPlaceHolder2" runat="server">
          </asp:contentplaceholder>
        </td>
      </tr>
    </table>
    </asp:Content>
```

典型应用 下面根据@ Master 指令来了解母版页中嵌套母版页的应用。

所谓嵌套母版页就是在大的母版页中包含一个小的母版页，以此来完成页面构建工作。但需要明确的是无论母版页如何嵌套，都必须包含一个内容页（其原因是扩展名为.master 的文件不能被浏览器访问）。典型的嵌套母版页如图 1.2 所示。

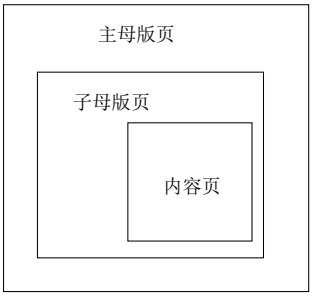

图 1.2 嵌套母版页的示意图

> **说明** 利用嵌套的母版页可以创建组件化的母版页。例如，大型网站可能包含一个用于定义站点外观的总体母版页。然而，不同的网站内容合作伙伴又可以定义各自的子母版页，这些子母版页引用网站母版页，并相应地定义合作伙伴内容的外观。

使用嵌套母版页时，任何子母版页都被看做普通的内容页，并以内容页的形式加以实现。子母版页是一种包含<asp:Content>和<asp:ContentPlaceHolder>元素组合的内容页。与其他内容页一样，一个子母版

页指向一个母版页，并且为其父母版页的占位符提供内容块。

> 说明　内容页是只包含<asp:Content>服务标签的 ASP.NET 页，它的关键部分就是 Content 控件，可称为内容控件，母版页中的占位符所对应的具体内容就包含在这个控件中。

本示例在 Visual Studio 2008 中所设计的子母版页如图 1.3 所示。

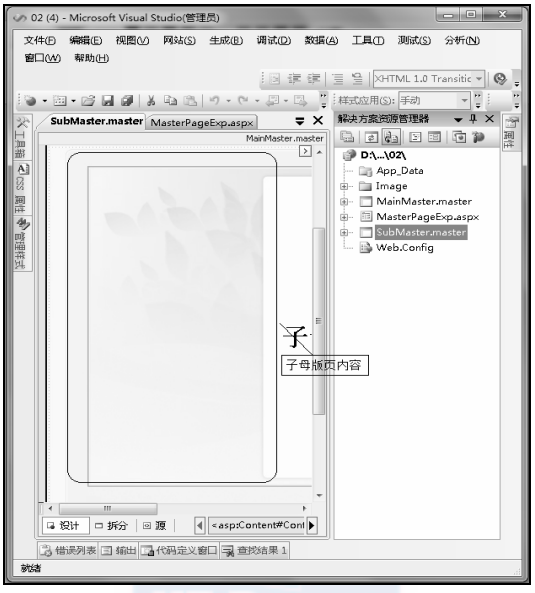

图 1.3　Visual Studio 2008 子母版页的设计页面

程序实现的主要步骤如下：

（1）新建一个网站，将其默认的主页 Default.aspx 重新命名为 MasterPageExp.aspx。

（2）在“解决方案管理器”面板中，右击网站名称，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项…”对话框，首先添加两个母版页，分别命名为 MainMaster（主母版页）和 SubMaster（子母版页），然后再添加一个 Web 窗体，命名为 MasterPageExp.aspx，并将其做为 SubMasterPage（子母版页）的内容页。

示例运行结果如图 1.4 所示。

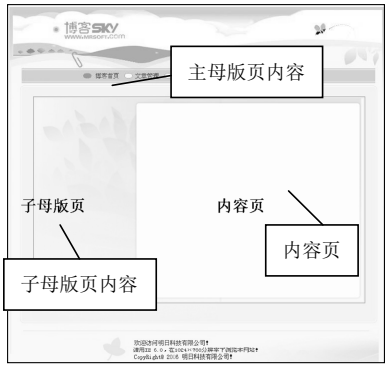

图 1.4　嵌套母版页示例图

（3）实现步骤如下：

主母板页 MainMaster.master 源代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MainMaster.master.cs"
Inherits="MainMaster" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>主母版页</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <table style="width: 759px; height: 758px" cellpadding="0" cellspacing="0">
            <tr>
                <td style="background-image: url(Image/baner.jpg); width: 759px; height:
                153px">
                </td>
            </tr>
            <tr>
                <td style="width: 759px; height: 498px" align="center" valign="middle">
        <asp:contentplaceholder id="MainContent" runat="server">
        </asp:contentplaceholder>
                </td>
            </tr>
            <tr>
                <td style="background-image: url(Image/3.jpg); width: 759px; height:
                107px">
                </td>
            </tr>
        </table>
    </div>
    </form>
</body>
</html>
```

子母版页 SubMaster.master 的源代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="SubMaster.master.cs"
Inherits="SubMaster"
MasterPageFile ="~/MainMaster.master" %>
<asp:Content id="Content1" ContentPlaceholderID="MainContent" runat="server">
    <table style="background-image: url(Image/2.jpg); width:759px; height: 498px">
    <tr>
    <td align ="center" valign ="middle">
        <h1>子母版页</h1>
    </td>
    <td align ="center" valign ="middle">
        <asp:contentplaceholder id="SubContent" runat="server">
        </asp:contentplaceholder>
    </td>
    </tr>
```

```
    </table>
</asp:Content>
```

（4）子母版页中的 Content 控件中声明的 ContentPlaceholder 控件用于为内容页实现占位。

内容页 MasterPageExp.aspx 的源代码如下：

```
<%@ Page Language="C#" MasterPageFile="~/SubMaster.master" AutoEventWireup="true"
CodeFile="Default.aspx.cs"
Inherits="_Default" Title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="SubContent" Runat="Server">
<table style="width :451px; height :391px">
<tr>
<td>
<h1>内容页</h1>
</td>
</tr>
</table>
</asp:Content>
```

1.6　@ MasterType指令——母版页类型指令

@MasterType 指令为 ASP.NET 页的 Master 属性分配类名，使得该页可以获取对母版页成员的强类型引用。

语法

```
<%@ MasterType attribute="value" [attribute="value"...] %>
```

其中，attribute 为@MasterType 指令的属性，具体说明如表 1.4 所示。

表 1.4　@MasterType 指令的属性说明

属　性	描　述
TypeName	指定母版页的类型名称
VirtualPath	指定生成强类型的文件的名称

注意 如果未定义 VirtualPath 属性，则此类型必须存在于当前链接的某个程序集（如 Bin 或 App_Code）中，而且 TypeName 属性和 VirtualPath 属性不能同时存在于@MasterType 指令中，如果同时存在，则@MasterType 指令失效。

示例　下面的代码示例演示如何设置 ASP.NET 母版页的虚拟路径。

```
<%@ MasterType VirtualPath="~/masters/SourcePage.master"" %>
```

典型应用　本示例实现的是使用@MasterType 指令引用母版页的公共属性，通过在内容页中添加@MasterType 指令，并将 Welcome 字样赋予母版页的公共属性。可使内容页中的 Master 属性被强类型化。Master 属性可被声明为动态创建的页面类的真实类型，并且允许用户编写强类型代码。另外，在设置@MasterType 指令时，必须设置 VirtualPath 属性，以便指定与内容页相关的母版页存储地址。示例运行结果如图 1.5 所示。

图 1.5 使用@MasterType 指令获取母版中的属性

下面介绍示例的实现步骤：

（1）新建一个网站，将其命名为 MasterType_Text。

（2）在“解决方案资源管理器”面板中，右击网站名称 MasterType_Text，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项…”对话框，首先添加一个母版页，命名为 MasterPage.master，再添加一个 Web 窗体，命名为 Default.aspx，作为母版页的内容页。

（3）分别在母版页和内容页上添加一个 Label 控件。母版页 Label 控件的 ID 属性为 labMaster，用于显示系统日期；内容页 Label 控件的 ID 属性为 labContent，用于显示母版页中的 Label 控件值。

（4）主要程序代码如下。

在母版页中定义了一个 String 类型的公共属性 MValue。示例的源代码如下：

```
public partial class MasterPage : System.Web.UI.MasterPage
{
    string mValue = "";
    public string MValue
    {
        get
        {
            return mValue;
        }
        set
        {
            mValue = value;
        }
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        this.labMaster.Text = "今天是"+DateTime.Today.Year+"年"+DateTime.Today.Month+"月
"+DateTime .Today.Day+"日";
    }
}
```

并且通过<%=this.MValue%>显示在母版页中，示例的源代码如下：

```
<td style="background-image: url(Image/baner.jpg); height: 153px" align="center">
<asp:Label ID="labMaster" runat="server"></asp:Label>
<%=this.MValue%>
```

在内容页代码头的设置中，增加了<%@ MasterType%>，并在其中设置了 VirtualPath 属性，用于设置被强类型转化的母版页的 URL 地址。示例的源代码如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="Default.aspx.cs"
Inherits="_Default" Title="Untitled Page" %>
<%@ MasterType VirtualPath ="~/MasterPage.master" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <table align="center">
        <tr>
            <td style="width: 86px; height: 21px;">
                <asp:Label ID="labContent" runat="server" Width="351px"></asp:Label></td>
        </tr>
    </table>
</asp:Content>
```

在内容页的 Page_Load 事件下，通过 Master 对象引用母版页中的公共属性，并将 Welcome 字段赋给母版页中的公共属性。示例代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Master.MValue = "Welcome";
}
```

1.7 @ OutputCache指令——输出指令

@OutputCache 指令用于以声明的方式控制 ASP.NET 页，或页中包含的用户控件的输出缓存策略。页输出缓存，就是在内存中存储处理后的 ASP.NET 页的内容。这一机制允许 ASP.NET 向客户端发送页响应，而不必再次经过页处理生命周期。

页输出缓存对于那些不经常更改，但需要大量处理才能创建的页特别有用。例如，如果创建大通信的网页来显示不需要频繁更改新的数据，页输出缓存则可以极大地提高该页的性能。用户可以分别为每个页配置页缓存，用户也可以在 Web.config 文件中创建缓存配置文件。利用缓存配置文件，只定义一次缓存设置就可以在多个页中使用这些设置。

语法

```
<%@ OutputCache Duration="#ofseconds"
   Location="Any | Client | Downstream | Server | None |
    ServerAndClient "
    Shared="True | False"
    VaryByControl="controlname"
    VaryByCustom="browser | customstring"
    VaryByHeader="headers"
    VaryByParam="parametername"
    VaryByContentEncoding="encodings"
```

```
    CacheProfile="cache profile name | ""
    NoStore="true | false"
    SqlDependency="database/table name pair | CommandNotification"
%>
```

关于上述@OutputCache 指令的常用属性如表 1.5 所示。

表 1.5 @ OutputCache 指令常用属性说明

属 性	描 述
Duration	必选属性。页或用户控件进行缓存的时间（以秒计），在页或用户控件上设置该属性为来自对象的 HTTP 响应建立了一个过期策略，并将自动缓存页或用户控件输出
Location	OutputCacheLocation 枚举值之一。默认值为 Any。包含在用户控件（.ascx 文件）中的@OutputCache 指令不支持此属性
CacheProfile	与该页关联的缓存设置的名称。这是可选属性，默认值为空字符串（""）
Shared	一个布尔值，确定用户控件输出是否可以由多个页共享，默认值为 False。包含在 ASP.NET 页（.aspx 文件）中的@OutputCache 指令不支持此属性
SqlDependency	标识一组数据库/表名称对的字符串值，页或控件的输出缓存依赖于这些名称对
VaryByCustom	表示自定义输出缓存要求的任意文本。如果赋予该属性的值为 browser，缓存将随浏览器名称和主要版本信息的不同而异。如果输入自定义字符串，则必须在应用程序的 Global.asax 文件中重写 GetVaryByCustomString 方法
VaryByParam	分号分隔的字符串列表，用于使输出缓存发生变化。默认情况下，这些字符串与随 GET 方法属性发送的查询字符串值对应，或与使用 POST 方法发送的参数对应。将该属性设置为多个参数时，对于每个指定参数组合，输出缓存都包含一个不同版本的请求文档。可能的值包括 none、星号（*）及任何有效的查询字符串或 POST 参数名称
VaryByControl	一个分号分隔的字符串列表，用于更改用户控件的输出缓存。这些字符串代表用户控件中声明的 ASP.NET 服务器控件的 ID 属性值

上述列表中的@OutputCache 指令属性有一些需要注意的地方，说明如下。

- CacheProfile 属性：包含在用户控件（.ascx 文件）中的@ OutputCache 指令不支持此属性。在页中指定此属性时，属性值必须与 outputCacheSettings 节下面的 outputCacheProfiles 元素中一个可用项的名称匹配。如果此名称与配置文件项不匹配，将引发异常。
- Location 属性：OutputCacheLocation 枚举值之一，其中 OutputCacheLocation 枚举用于指定有效值，用于控制资源的输出缓存 HTTP 响应的位置。
- Shared 属性：如果将 Shared 属性设置为 True，则多个 Web 窗体页可以访问缓存的用户控件输出。如果不将该属性设置为 True，则默认行为是为包含用户控件的每一页缓存用户控件输出的一个版本。通过启用 Shared 属性，可能可以节省大量内存。
- VaryByParam 属性：在 ASP.NET 页和用户控件上使用@OutputCache 指令时，需要该属性或 VaryByControl 属性。如果没有包含它，则发生分析器错误。如果不希望通过指定参数来改变缓存内容，请将值设置为 none。如果希望通过所有的参数值改变输出缓存，请将属性设置为星号（*）。

示例 下面的代码示例演示如何设置页或用户控件进行输出缓存的持续时间。示例的源代码如下：

```
<%@ OutputCache Duration="100" VaryByParam="none" %>
```

下一个代码示例演示如何指示输出缓存按页或用户控件的位置对它们进行缓存，并根据窗体的 POST

方法或查询字符串对窗体参数进行计数。每个收到的具有不同位置或计数参数（或两者）的 HTTP 请求都进行 10 秒的缓存处理。带有相同参数值的任何后继请求都将从缓存中得到满足，直至超过输入的缓存期。示例的源代码如下：

```
<%@ OutputCache Duration="10" VaryByParam="location;count" %>
```

1.8 @ Page 指令——页指令

@Page 指令允许开发人员为页面指定多个配置选项，并且该指令只能在 Web 窗体页中使用。每个.aspx 文件只能包含一条@Page 指令。@Page 指令可以指定：页面中代码的服务器编程语言；页面是将服务器代码直接包含在其中（即单文件页面），还是将代码包含在其中（即代码隐藏页面）；调试和跟踪选项，以及页面是否为某母版页的内容页。

语法

```
<%@ Page attribute="value" [attribute="value"...] %>
```

关于上述@ Page 指令常用属性说明如表 1.6 所示。

表 1.6 @ Page 指令属性说明

属 性	描 述
AsyncTimeOut	定义在处理异步任务时使用的超时时间间隔（以秒为单位）。默认值为 45 秒，该值必须是整数
AutoEventWireup	指示页的事件是否自动绑定。如果启用了事件自动绑定，则为 True；否则为 False。默认值为 True
Buffer	确定是否启用了 HTTP 响应缓冲。如果启用了页缓冲，则为 True；否则为 False。默认值为 True
ClassName	一个字符串，指定在请求页时将进行动态编译的页类的名称。此值可以是任何有效的类名，并且可以包括完全限定的类名。如果未指定该属性的值，则已编译页的类名将基于页的文件名
CodeFileBaseClass	指定页的基类及其关联的代码隐藏类的类型名称。此属性是可选的，但如果使用此属性，则必须同时使用 CodeFile 属性
CodeFile	指定指向页引用的代码隐藏文件的路径。此属性与 Inherits 属性一起使用，可以将代码隐藏源文件与网页相关联。此属性仅对编译的页有效
Debug	指示是否应使用调试符号编译该页。如果使用调试符号编译该页，则为 True；否则为 False。由于此设置影响性能，因此只应在开发期间将此属性设置为 True
EnableEventValidation	在回发和回调方案中启用事件验证。如果验证事件，则为 True；否则为 False。默认值为 True
EnableSessionState	定义页的会话状态要求。如果启用了会话状态，则为 True；如果可以读取会话状态但不能进行更改，则为 ReadOnly；否则为 False。默认值为 True。这些值是不区分大小写的
EnableTheming	指定是否在页上使用主题。如果使用主题，则为 True；否则为 False。默认值为 True
EnableViewState	指定是否在页请求之间保持视图状态。如果要保持视图状态，则为 True；否则为 False。默认值为 True

（续）

属　性	描　述
Inherits	定义供页继承的代码隐藏类。它可以是从 Page 类派生的任何类。此属性与 CodeFile 属性一起使用，后者包含指向代码隐藏类的源文件的路径
Language	指定在编译控件中所有内联呈现（<% %>和<%= %>）和代码声明块时使用的语言。值可以表示任何.NET Framework 支持的语言，包括 Visual Basic、C#或 JavaScript。对于每个控件，只能使用和指定一种语言
MasterPageFile	设置内容页的母版页或嵌套母版页的路径。支持相对路径和绝对路径
Src	指定包含链接到页的代码的源文件路径，在链接的源文件中，可以选择将页的编程逻辑包含在类中或代码声明块中
StyleSheetTheme	指定在页上使用的有效主题标识符。如果设置了 StyleSheetTheme 属性，则单独的控件可以重写主题中包含的样式设置。这样，主题可以提供站点的整体外观，同时，利用 StyleSheetTheme 属性中包含的设置可以自定义页及其各个控件的特定设置
Theme	指定在页上使用的有效主题标识符。如果设置 Theme 属性时没有使用 StyleSheetTheme 属性，则将重写控件上单独的样式设置，允许用户创建统一而一致的页外观
Trace	指定是否启用跟踪。如果启用了跟踪，则为 True；否则为 False。默认值为 False
TraceMode	指定当启用跟踪时如何为页显示跟踪消息。可能的值为 SortByTime 和 SortByCategory。当启用跟踪时，默认值为 SortByTime
Transaction	指定在页上是否支持事务。可能的值有 Disabled、NotSupported、Supported、Required 和 RequiresNew。默认值为 Disabled
ValidateRequest	指定是否应发生请求验证。如果为 True，请求验证将根据具有潜在危险的值的硬编码列表检查所有输入数据。如果出现匹配情况，将引发 HttpRequestValidation Exception 异常。默认值为 True

若要定义@Page 指令的多个属性，请使用一个空格分隔每个属性/值对。对于特定属性，不要在将该属性与其值相连的等号（=）两侧加空格。

> **注意** @Page 指令具有大量与应用于整个源文件的其他指令（例如，在 Web 用户控件的.ascx 文件中使用的@Control 指令，以及在母版页的.master 文件中使用的@Master 指令）共用的属性。

示例 关于@Page 指令下面分别给予几个示例讲解其一些重要属性的应用。

1．AutoEventWireup 属性

指示页的事件是否自动绑定。

ASP.NET 默认值为 True。ASP.NET 页触发的事件，如 Inti、Load 等，在默认情况下，可以使用“ Page_事件名 ”（例如 Page_Load、Page_Init 等）的命名约定，将页事件绑定到相应的方法，编辑页面时 ASP.NET 将查找基于此命名约定的方法，并自动执行。

本示例代码指示如果要显示声明事件的处理程序，可以将 AutoEventWireup 属性设置为 false。代码如下：

```
<%@Page Language="C#" AutoEventWireup = false" %>
```

2．CodeFile 属性

指定指向页引用的代码隐藏文件的路径。此属性与 Inherits 属性一起使用，可以将代码隐藏源文件与网页相关联。此属性仅对编译的页有效。

本示例代码指示 ASP.NET 页编译器使用 C#作为页的服务器端代码语言，在添加一个新页时，设置该页代码隐藏文件的路径为“ Default.aspx.cs ”，代码如下：

```
<%@Page Language="C#" AutoEventWireup = true" CodeFile=" Default.aspx.cs" Inherits=
"Default1"%>
```

3．EnableTheming 属性

EnableTheming 属性指定在应用主题时是否可以修改母版页的外观和母版页上控件的外观。如果可以应用主题，则为 True；否则为 False。默认值为 True。

本示例以 C#作为内联代码语言，并显式地声明了 EnableTheming 属性为 True，以便应用主题，即通过应用主题来修改母版页的外观和母版页上控件的外观，代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage"
EnableTheming="true" %>
```

4．ValidateRequest 属性

ValidateRequest 属性指示是否应发生请求验证。如果为 True，则请求验证将根据具有潜在危险的值的硬编码列表检查所有输入数据。如果出现匹配情况，则将引发 HttpRequestValidationException 异常。默认值为 True。

> **注意** 该功能有助于减少对简单页或 ASP.NET 应用程序进行跨站点脚本攻击的风险。如果应用程序不能正确验证用户输入，则可能会受到多种类型格式错误的输入的攻击，包括跨站点脚本攻击和 Microsoft SQL Server 注入式攻击。

本示例演示的是在博客应用程序中通过添加引用方式添加第三方控件 FreeTextBox 来填写留言信息，当其他信息添加完成后单击“ 提交 ”按钮时，会出现如图 1.6 所示的错误提示“ 从客户端 (DJieshao=" 1314521 ") 中检测到有潜在危险的 Request.From 值 ”。

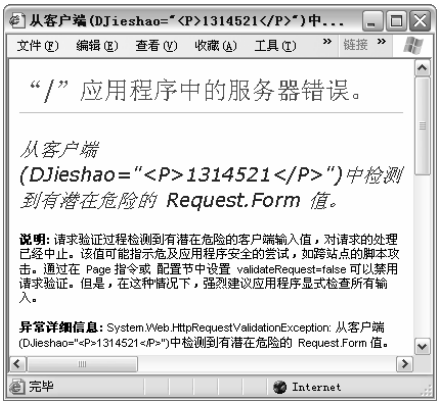

图 1.6 ValidateRequest 属性设置出现的错误

要解决上图中出现的“ 从客户端（DJieshao=" 1314521 "）中检测到的有潜在危险的 Request.From

值”的错误，只需在 Web.config 配置文件的 pages 元素标记中将 ValidateRequest 属性设置为 false 即可，设置的源代码如下：

```
<pages ValidateRequest="false" />
```

另一种处理这个 ValidateRequest 属性设置出现错误的方法，可在出错的这个页的页码头设置 ValidateRequest 属性为 false，设置的源代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" ValidateRequest="false" CodeFile=
"CreatPersonShop.aspx.cs"
Inherits="CreatPersonShop" %>
```

1.9 @ PreviousPageType指令——获取页指令

@PreviousPageType 指令为 ASP.NET 页提供用于获取上一页强类型的方法，可通过 PreviousPage 属性访问上一页，该指令只能用于 Web 窗体页（.aspx 文件）。

语法

```
<%@ PreviousPageType attribute="value" [attribute="value"...] %>
```

其中，attribute 为@PreviousPageType 指令的属性，具体说明如表 1.7 所示。

表 1.7 @ PreviousPageType 指令的属性说明

属 性	描 述
TypeName	指定母版页的类型名称
VirtualPath	指定生成强类型的文件的路径

注意 使用@PreviousPageType 指令获取 PreviousPage 属性的强类型。只能在 Web 窗体页（.aspx 文件）上使用@PreviousPageType 指令。如果同时定义了属性 TypeName 和 VirtualPath，则@PreviousPageType 指令将失败。

示例 本示例设置母版页虚拟路径的源代码如下：

```
<%@ PreviousPageType VirtualPath="~/SourcePage.aspx"%>
```

1.10 @ Reference指令——连接指令

@Reference 指令以声明的方式将网页、用户控件或 COM 组件连接至当前的网页或用户控件。使用此指令可以动态编译与生成提供程序关联的页面、用户控件或另一个类型的文件，并将其链接到包含@Reference 指令的当前网页、用户控件或母版页文件，这样就可以从当前文件内部引用外部编译的对象及其公共成员。

语法

```
<%@ Reference Page="path to .aspx page"
   Control="path to .ascx file"
   virtualPath="path to file" %>
```

关于上述@Reference 指令语法中的属性说明如表 1.8 所示。

表 1.8　@Reference 指令属性

属　性	描　述
Page	外部页，ASP.NET 应动态编译该页并将它链接到包含@Reference 指令的当前文件
Control	外部用户控件，ASP.NET 应动态编译该控件并将它链接到包含@Reference 指令的当前文件
VirtualPath	引用的虚拟路径。只要生成提供程序存在，可以是任何文件类型。例如，它可能会指向母版页

说明 使用此指令可以动态编译与生成提供程序关联的页面、用户控件或另一个类型的文件，并将其链接到包含@Reference 指令的当前网页、用户控件或母版页文件，这样用户就可以从当前文件内部引用外部编译的对象及其公共成员。

示例　本示例使用@Reference 指令链接到用户控件，示例的源代码如下：

```
%@Reference Control="MyControl.ascx"%
```

1.11　@Register指令——关联指令

@Register 指令创建标记前缀和自定义控件之间的关联，这为开发人员提供了一种在 ASP.NET 应用程序文件（包括网页、用户控件和母版页）中引用自定义控件的简单方法。

在以下情况中，使用@Register 指令：

- 以声明方式将自定义服务器控件添加到网页、用户控件、母版页或外观文件。
- 以声明方式将用户控件添加到网页、用户控件、母版页或外观文件。

语法

```
//第一种
<%@ Register tagprefix="tagprefix"  namespace="namespace" assembly="assembly" %>
//第二种
<%@ Register tagprefix="tagprefix"  namespace="namespace" %>
//第三种
<%@ Register tagprefix="tagprefix" tagname="tagname" src="pathname" %>
```

关于上述 Register 指令语法中的属性及说明如表 1.9 所示。

表 1.9　@ Register 指令属性

属　性	描　述
assembly	与 tagprefix 属性关联的命名空间所驻留的程序集
namespace	正在注册的自定义控件的命名空间
src	与 tagprefix:tagname 对关联的声明性 ASP.NET 用户控件文件的位置（相对的或绝对的）
tagname	与类关联的任意别名，此属性只用于用户控件
tagprefix	一个任意别名，它提供对包含指令的文件中所使用的标记的命名空间的短引用

关于@Register 指令属性的应用说明如下：

- 对于声明性用户控件，请使用 tagname、tagprefix 和 src 属性。在页中声明控件时，前两个属性总

是以冒号分隔对（tagprefix:tagname）的形式一起使用，同时还可以将多个命名空间映射到同一tagname，如以下示例所示：

```
<% @Register tagprefix="tag1" namespace="MyNamespace1"/>
<% @Register tagprefix="tag1" namespace="MyNamespace2"/>
```

- src 属性值既可以是相对路径，也可以是从应用程序的根目录到用户控件源文件的绝对路径。

为方便使用，建议使用相对路径。例如，假设将应用程序的所有用户控件文件存储在应用程序根目录的子目录“ \UserControl ”中。若要包括 myUserControl.ascx 文件中的用户控件，请在@Register 指令中包含以下内容：

```
Src="~\ UserControl \ myUserControl.ascx"
```

说明 代字号(~)字符表示应用程序的根目录。如果用户控件和包含该控件的页位于同一目录中，则 src 属性值应当为.ascx 文件的文件名和扩展名。

- 当包含已经编译为.dll 文件供应用程序使用的自定义服务器控件时，请将 tagprefix 属性与 assembly 和 namespace 属性一起使用。如果没有包含 namespace 属性，或者给该属性分配了一个空字符串（""），则会出现分析器错误。

注意 在开发自定义服务器控件时，必须将其包含在命名空间中，如果没有包含在命名空间中，则将无法从 ASP.NET 页中访问该控件。

示例 以下示例使用@Register 指令声明 tagprefix 和 tagname 别名，同时分配 src 属性，以在网页内引用用户控件。示例的源代码如下：

```
<asp:calendar id="Calendar1" runat="server" />
<%@ Page %>
<%@ register tagprefix="uc1"  tagname="CalendarUserControl"
   src="~/CalendarUserControl.ascx" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Calendar Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <uc1:calendarusercontrol runat="server" />
  </form>
</body>
</html>
```

上述示例代码中，tagprefix 属性分配一个用于标记的任意前缀值“ uc1 ”。tagname 属性使用分配给用户控件的类名称的值“ CalendarUserControl ”（尽管此属性的值是任意的，并可以使用任何字符串值，但是不必使用所引用的控件的类名称）。src 属性指向用户控件的源文件“ ~/CalendarUserControl.ascx ”（相对于应用程序根文件夹）。

所以，可以按照如下形式引用用户控件（即使用前缀、冒号及标记名称）。代码如下：

```
<uc1:calendarusercontrol runat="server" />
```

典型应用　在讲解@Control 指令的典型应用时，编写了一个用户控件，并将其命名为 WebUserControl.ascx，那么在将这个制作好的 Web 用户控件添加至普通的窗体页（即.aspx 页）中时，就要应用@Register 指令对创建的用户控件进行注册操作。示例运行结果如图 1.7 所示。

图 1.7　使用 Web 用户控件制作导航条

实现本示例的源代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits=
"_Default" %>
<%@ Register Src="Control/header.ascx" TagName="header" TagPrefix="uc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11
/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>使用 Web 用户控件制作导航条</title>
    </style>
</head>
<body style="text-align: center;">
    <form id="form1" runat="server">
        <table>
            <tr>
                <td colspan="5" style="text-align: center" class="style1">
                    <uc1:header ID="Header1" runat="server" />
                    </td>
            </tr>
            </table>
    </form>
</body>
</html>
```

当将用户控件拖曳到 Default.aspx 页后，本示例代码中 HTML 视图下代码顶端将会自动生成如下所示的代码：

```
<%@ Register Src="Control/header.ascx" TagName="header" TagPrefix="uc1" %>
```

1.12　@ WebHandler指令——处理指令

定义 HTTP 处理程序（.ashx）文件的属性和编译选项。

语法

```
<%@ WebHandler attribute="value" [attribute="value"...] %>
```

其中，attribute 属性如表 1.10 所示。

表 1.10 attribute 属性

属 性	描 述
Class	指定将在请求处理程序时动态编译的类的名称。此值可以引用从 IHttpHandler 继承且可以包括完全限定类名的任何类
CodeBehind	指定包含与处理程序关联的类的已编译文件名称，该属性不能在运行时使用。此属性是为了与以前版本的 ASP.NET 兼容
CompilerOptions	指定包含用于编译处理程序的选项的字符串，在 C#和 Visual Basic 中，这是编译器命令行开关的序列
Debug	如果使用调试符号编译处理程序，则为 True；否则为 False。由于此设置会影响性能，因此只应在开发期间将此属性设置为 True
Description	提供该处理程序的文本说明。ASP.NET 分析器忽略该值
Language	指定编译处理程序中的所有代码时使用的语言。值可以表示任何.NET Framework 语言，包括 Visual Basic、C#或 JScript。每个处理程序只能使用一种语言
WarningLevel	指定用户希望编译器将警告视为错误（从而停止对处理程序进行编译）的编译器警告级别。警告级别可以是 0～4

对于@WebHandler 指令，读者需要理解以下 3 方面内容：

- 此指令仅在用做 HTTP 处理程序的文件中有效。在默认情况下，ASP.NET 将扩展名为.ashx 的文件视为处理程序。

注意 如果用户不希望对处理程序使用.ashx 文件扩展名，则可以创建作为标准类（.cs 或.vb 文件）的处理程序，此标准类实现 IHttpHandler 接口。处理程序类文件不要求@WebHandler 指令。

- 每个文件只能包含一条@WebHandler 指令。由于每个处理程序只能使用一种语言，因此每条@WebHandler 指令只能定义一个 Language 属性。

注意 @WebHandler 指令与适用于整个源文件的其他指令有许多共同的属性，例如在网页的.aspx 文件中使用的@Page 指令。

- 若要定义@WebHandler 指令的多个属性，请用一个空格分隔每个属性/值对，不要在将属性与其值相连的等号（=）的任意一侧包含空格。

示例 本示例使用@WebHandler 指令指示 ASP.NET 页编译器使用 C#作为内联代码语言并指定了类名。代码如下：

```
<%@ WebHandler Language=" C#" Class="MyTestHandler" %>
```