# 第1章

# 认识 ASP. NET

ASP. NET 是创建动态网页的一种强大的服务器端技术，是一种基于 B/S(浏览器/服务器)架构的应用程序，可创建动态、可交互的 Web 页面。与其他的服务器端技术相比，ASP. NET 以其不可替代的优势在 Web 开发上占有了一片广阔的天地。

**学习目标**

- 了解 C# 的历史。
- 掌握. NET 的框架结构。
- 熟练掌握 ASP. NET 的特点。
- 了解 B/S 架构。

## 1.1 C# 的历史

C#(读做 C sharp)是微软公司发布的一种面向对象的运行于. NET Framework 之上的高级程序设计语言。C# 与 Java 语言的语法非常相似，学习过 Java 语言的程序员再学习 C# 会节省很多时间。不过 C# 也有 Java 语言所没有的特点。

### 1. 中间代码

计算机并不能直接执行高级语言代码，只能执行机器语言代码，因此，用高级语言编写的程序都需要进行编译或解释成机器语言。在 ASP. NET 中，源代码并未直接编译成机器语言代码，而是先编译为微软中间语言(Microsoft intermediate language，MSIL 或简写为 IL)，然后由即时(just-in-time，JIT)编译器进一步编译成机器语言。其中，JIT 编译器并非一次完全编译，而是调用哪部分代码就编译哪部分，这样启动时间更短。同时，编译好的代码再次运行时不需要重新编译，这极大地提高了 Web 应用程序的性能。ASP. NET 页面编译过程如图 1-1 所示。

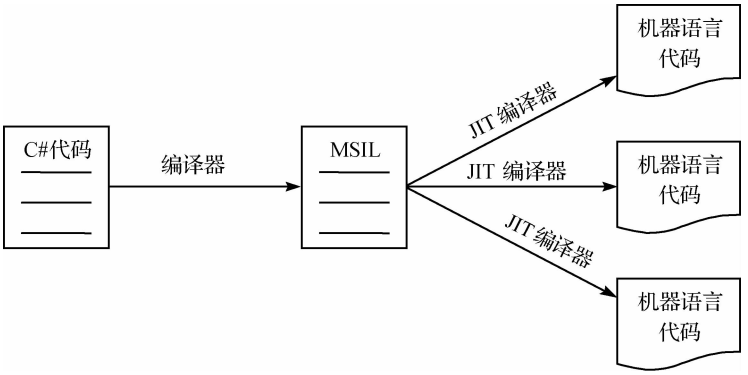


图 1-1　ASP. NET 页面编译过程

### 2. 基本的数据类型

C＃拥有比 C、C++ 或者 Java 更广泛的数据类型。这些类型包括 bool、byte、ubyte、short、ushort、int、uint、long、ulong、float、double 和 decimal。C＃中的数值类型都分为有符号型和无符号型，如 int 为有符号整型，uint 为无符号整型。与 C 不同的是，C＃中的一个字符变量包含的是 16 位 Unicode 字符，而不是 8 位。在 C＃中出现的新数据类型是 decimal 类型，类似于 double 类型，用来存放货币数据。

### 3. 与 COM 的集成

C＃对 Windows 程序最大的卖点可能就是它与 COM 的无缝集成了（COM 是微软的 Win32 组件技术）。实际上，最终有可能在任何. NET 语言中编写 COM 客户端和服务器端。

C＃是兼顾系统开发和应用开发的最佳实用语言，并且很有可能成为编程语言历史上的第一个“全能”型语言。C＃从出现至今，经历了几个阶段，这几个阶段的新内容及运行环境如表 1-1 所示。

表 1-1　C＃历史版本

| C＃版本 | 内　　容 | 运行环境 |
|---|---|---|
| 1. 0<br>1. 1 | 完全模仿 Java，并保留了 C/C++ 的一些特性，如 struct | CLR 1. 1 |
| 2. 0 | 加入了泛型 | CLR 2. 0 |
| 3. 0<br>3. 5 | 引入了 Linq（语言集成查询） | CLR 2. 0 |
| 4. 0 | 加入了 dynamic 关键字 | CLR 4. 0 |
| 4. 5 | 开发者预览版、异步文件操作 | CLR 4. 5 |

## 1.2 .NET 框架简介

. NET 框架（. NET Framework）是由微软开发的，一个致力于敏捷软件开发（agile software development）、快速应用开发（rapid application development）、平台无关性和网络透明

化的软件开发平台。. NET 框架是一个多语言组件开发和执行环境，它提供了一个跨语言的统一编程环境。. NET 框架的目的是便于开发人员更容易地建立 Web 应用程序和 Web 服务，使得 Internet 上的各应用程序之间可以使用 Web 服务进行沟通。从层次结构来看，. NET框架包括 3 个主要组成部分：公共语言运行时(common language runtime，CLR)、服务框架(services framework)以及上层的两类应用模板——传统的 Windows 应用程序模板(Windows Forms，简写为 WinForms)和基于 ASP. NET 的面向 Web 的网络应用程序模板(Web Forms 和 Web Services)。

公共语言运行时(CLR)是一个运行时环境，是所有. NET 应用程序都使用的编程基础。管理代码是 CLR 的基本原则，能够被管理的代码称为托管代码，反之则称为非托管代码。CLR 包含两个组成部分：CTS(通用类型系统)和 CLS(公共语言规范)。

### 1. CTS

在 Visual Studio 2010 中可以使用多种语言开发 Web 系统，这些语言都由 CLR 进行管理。这些语言的数据类型是不同的，CLR 是如何对这些语言进行管理的呢？CLR 通过 CTS 来解决不同语言的数据类型不同的问题，如 C # 中的整型是 int，而 VB. NET 中的是 Integer，通过 CTS 就可以把它们编译成通用的类型 Int32。所有的. NET 语言共享这个类型系统，在它们之间实现无缝相互操作。

### 2. CLS

在实际应用中可以发现，两种不同的语言不仅仅是数据类型有差别，不同语言的基本语法也不尽相同，那么如何解决这个问题呢?. NET 通过 CLS 限制了由这些不同点引发的互操作性问题。CLS 用来定义. NET 语言都应遵循的规则，通常包含标准数据类型和准则集。

## 1.3 ASP. NET 的特点

提到 ASP. NET，就不得不先了解活动服务器页面(active server pages，ASP)。为什么称为活动服务器页面？这是因为以前的互联网全部是由静态的 HTML 页面组成的，如果需要更新网站的内容，就不得不制作大量的 HTML 页面，有了 ASP 以后，我们就能够根据不同的用户，在不同的时间向用户显示不同的内容。但是由于 ASP 程序和网页的 HTML 混合在一个文件中的，使得程序看上去相当混乱，于是，微软推出了 ASP. NET。ASP. NET 将 WinForms 中的事件模型带入了 Web 应用程序的开发，程序员只需要拖动控件，处理控件的属性，不需要面对庞杂的 HTML 编码，可以说这是一项具有革命性意义的技术。此外，微软还提出了后置代码技术，将 HTML 页面和服务器代码分离开来，使程序更加清晰易懂。下面来看一下 ASP. NET 的优点。

### 1. 与浏览器无关

ASP. NET 生成的代码遵循万维网联盟(world wide Web consortium，W3C)推荐的可扩展超文本置标语言(extensible hypertext markup language，XHTML)标准，该标准的承诺是：只需设计页面一次，即可让该页以完全相同的方式在任何浏览器中显示和工作。例如，

程序在 IE 中显示的效果与在火狐等其他浏览器中显示的效果是一致的。因此使用 ASP. NET 开发的网站与浏览器无关。

### 2. 方便设置断点,易于调试

在站点的开发中如何调试代码一直是程序员非常关心的问题。由于 Web 服务器不受 IDE 的约束,所以人们一直希望能出现一个 IDE 可以调试网站代码。又由于 Internet 信息服务器(Internet information server,IIS)是由微软开发的,所以微软首先解决了这个问题,于是,ASP. NET 可以设置断点,调试程序。

### 3. 编译后执行,提高运行效率

计算机编程语言的执行有两种方法:一种是编译执行,如 C++;另一种是解释执行,如 Java。编译执行是指程序代码首先由编译器将其编译为机器代码,然后执行。这种方法的好处是,除了第一次编译运行较慢之外,再次执行的速度较快,因为第二次无须重新编译,就可以直接执行机器代码。解释执行是指程序代码由解释器一行一行地解释执行,这样不会产生最终的机器代码,但是每次执行程序时都需要重新解释。它的运行速度不如编译执行方法的快,但是这种方法跨平台性较好。

ASP. NET 的程序执行方法属于编译执行,但其并未直接编译成机器语言,而是先编译为微软中间语言,然后由 JIT 编译器进一步编译成机器语言。而且 JIT 编译器也并非一次完全编译,而是调用哪部分代码就编译哪部分,这样启动时间更短。同时,编译好的代码再次运行时不需要重新编译,极大地提高了 Web 应用程序的性能。ASP. NET 的程序编译方法见图 1-1。

### 4. 丰富的控件库

以前使用 Dreamweaver 开发网站时,常用的只有一些简单的 HTML 控件。而在浏览网页时,经常会看到一些树型菜单、横向菜单、导航条、验证功能和各种绑定数据等功能,这些功能用其他的开发方法来实现都需要大量的代码,不仅浪费了大量的时间,代码也容易出现错误。在 ASP. NET 中实现就不用这么麻烦了。ASP. NET 提供了大量的控件,通过这些控件,可以很容易地实现这些功能,只需要进行简单的设置即可。

### 5. 代码后置,使代码更清晰

ASP. NET 采用代码后置技术,将页面设计和服务代码相分离,这样可以使代码更清晰,有利于团队分工合作开发。

## 1.4 B/S 框架的特点

B/S(浏览器/服务器)结构是由 C/S(客户/服务器)结构改进而来的。C/S 结构开发的软件需要客户在客户端的计算机上安装相应的软件才能够实现协同处理数据,而这些软件会随着时间的推移而升级或淘汰,这样每个客户在经过一段时间的使用之后都需要对软件进行升级,或者淘汰。由于客户量较大,这样做起来很麻烦。那么能不能不在客户端安装软件而实现类似的功能呢?基于这个原因,出现了 B/S 结构,客户端只需要通过浏览器访问服务器端的网站即可,无须安装其他软件。而浏览器是在安装操作系统时自带的,一旦软件需

要更新，只需在服务器端升级软件即可，无须每一个客户都升级一次，这样就解决了上面的问题。B/S 结构实质上是一种三层结构，其工作原理如图 1-2 所示。

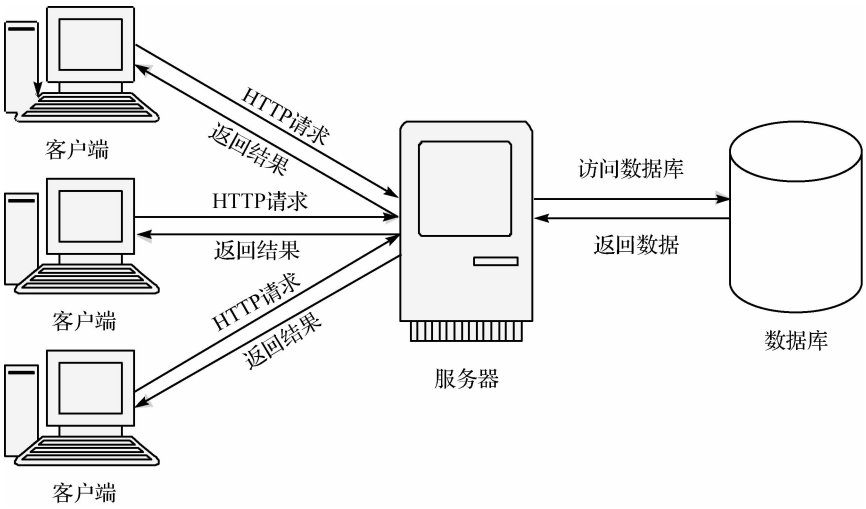


图 1-2　B/S 工作原理

B/S 结构具有以下优点。

(1)具有分布性特点，可以随时随地进行查询、浏览等业务处理。

(2)业务扩展简单、方便，通过增加网页即可增加服务器功能。

(3)维护简单、方便，只需要改变网页，即可实现所有用户的同步更新。

(4)开发简单，共享性强。

## 1.5 项目实战：使用 ASP. NET 创建个人站点

了解了 ASP. NET 的相关理论知识之后，下面来学习如何使用 Visual Studio 2010 创建 ASP. NET 应用程序。

打开 Visual Studio 2010，选择“文件”→“新建”→“网站”命令，在打开的“新建网站”对话框中选择使用 Visual C# 模板，创建“ASP. NET 网站”，在“Web 位置”下拉列表框中选择“文件系统”选项，并修改文件保存的位置后单击“确定”按钮，如图 1-3 所示，这时站点就建好了。

在“新建网站”对话框的“Web 位置”下拉列表框中有 3 个选项，分别表示创建站点的 3 种方式，这 3 种方式的名称和含义如下。

(1)文件系统方式。这种方式适用于在本地计算机上没有安装 IIS 或不希望通过 IIS 创建站点时的一种解决方案，将站点文件存储在本地硬盘的指定位置或存放于局域网的一个共享位置。那么有人可能就会有疑问了，没有安装 IIS 如何对 Web 站点进行测试呢？在 Visual Studio 2010 中有一个内置的简易 IIS，可以满足测试工作。这种方式的好处是，在开发的过程中外界不能访问站点，只能进行本地访问；缺陷是无法使用 IIS 中的某些特定功能。因此，可以在开发的过程中使用文件系统方式创建和调试站点，开发结束后再将创建的站点移植到 IIS 上。

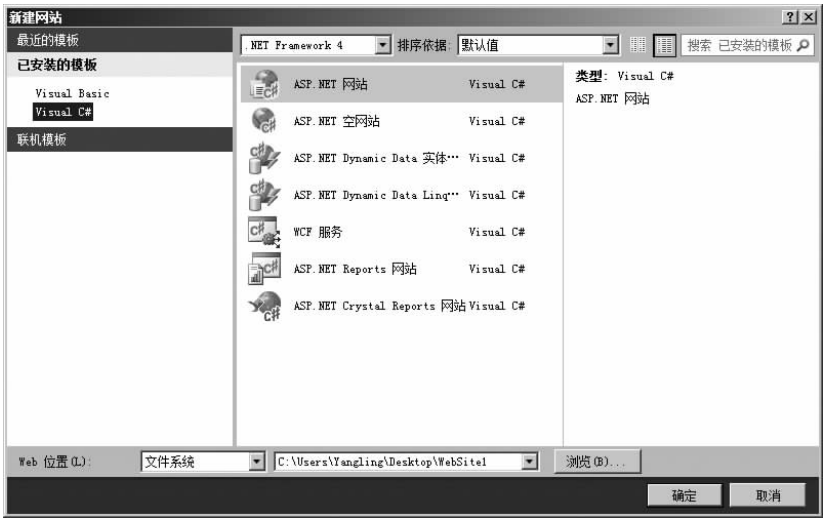


图 1-3 “新建网站”对话框

(2)HTTP 方式。HTTP 方式又称远程站点方式,这种方式可以在服务器上保存文件,通过 HTTP 方式访问它们。但是使用这种方式时,配置站点相当复杂,而且在开发过程中允许外界访问,缺点较多,所以一般不建议使用这种方式创建站点。

(3)FTP 方式。这种方式也可以在服务器上保存文件,但是通过 FTP 的方式来访问它们,是共享环境的一种配置。在共享环境下,许多用户可以同时使用项目,还可以使用 FTP 设置远程编辑文件。这种方式的最大缺陷是,不能使用源代码管理器来管理代码,团队中的多个人可能对项目进行相互矛盾的修改。因此,也不建议使用这种方式创建站点。

站点创建好后,可以看到 Visual Studio 2010 自动在站点中添加了一个 Default. aspx 网页,该网页使用 Site. master 母版。可以看到,在 IDE 中对网页的编辑方式有 3 种,分别是设计视图、拆分视图和源视图,默认为源视图。这 3 种视图的用法与 Dreamweaver 中网页设计的方法一致。此时,如果单击工具栏上的“启动”按钮 或按 F5 键,即可运行应用程序,但是因为是第一次运行该站点,所以会弹出图 1-4 所示的“未启用调试”对话框。该对话框的含义是:该站点无法在调试模式下运行,如果希望在调试模式下运行,则需修改 Web. config 配置文件,好在 Visual Studio 2010 可以自动帮用户修改配置文件,用户只需要单击“确定”按钮即可;如果不希望在调试模式下运行,则选中“不进行调试直接运行”单选按钮。在这里我们希望使用调试模式,所以直接单击“确定”按钮即可。

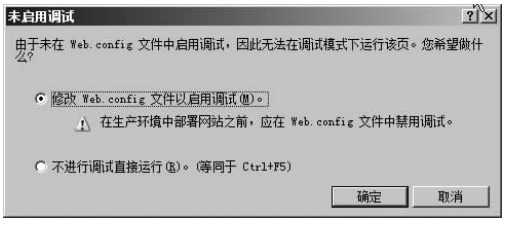


图 1-4 “未启用调试”对话框

这时，应用程序就可以运行了。网站启动后运行 Default. aspx 网页，运行结果如图 1-5所示。


图 1-5　默认的 Default. aspx 网页

下面来看 Visual Studio 2010 在新建的站点中自动添加了哪些内容。通过“解决方案资源管理器”面板可以看到这些文件，如图 1-6 所示。

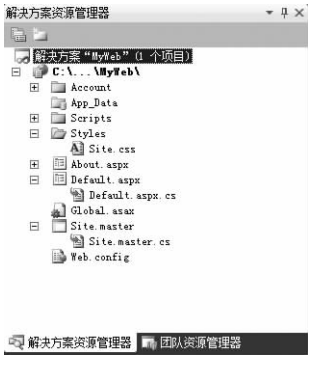


图 1-6　解决方案资源管理器

解决方案的名称为前面新建网站的文件夹名称，该项目下面自动创建了以下内容。

(1)Account 文件夹。该文件夹里面有 4 个网页，分别是登录页面、注册页面、修改密码页面和修改密码成功跳转页面，在网站需要这些页面时可以对这些页面进行修改，以达到用户的要求。

(2)App_Data 文件夹。该文件夹用来存放用户数据文件，如 Access 数据库文件、用户创建的 XML 文件或文本文件等。

(3)Scripts 文件夹。用来存放客户端脚本文件，可以将网站中用到的 JavaScript 文件放在这个文件夹中。

(4)Styles 文件夹。用来存放 CSS 文件，如网站的整体样式就存放在该文件夹中的 Site. css文件中。

(5)About. aspx 文件和 Default. aspx 文件。About. aspx 文件是 IDE 自动创建的“关于”页面；Default. aspx 文件是前面运行所看到的页面。

(6)Default. aspx. cs 文件。该文件是 Default. aspx 的后置文件，用来编写服务器端的

C#代码。当然这些代码也可以嵌入到 Default. aspx 网页中。

(7)Global. asax 文件。即 ASP. NET 应用程序文件,提供了一种在一个中心位置响应应用程序级或模块级事件的方法。

(8)Site. master 文件。母版文件,在后面的章节中会详细介绍母版文件的用法。

(9)Web. config 文件。配置文件,如可以在这个文件中设置该应用程序在调试模式下运行,方法是修改配置文件中"compilation"节点的"debug"属性值为"true"。

注意:在硬盘上找到新建网站的位置,会发现根本就没有解决方案文件(扩展名为. sln),那么这个文件在哪里呢?查看图 1-6 中所示的解决方案属性,会发现这个文件在"C:\Users\用户名\Documents\Visual Studio 2010\Projects\项目名称"目录中,解决方案的名称就是项目名称. sln,其中的用户名用计算机名称替换,项目名称用新建网站中命名的文件夹名替换。

在其他的 Web 网站开发中所看到的服务器代码都是内嵌在网页中的,而通过前面解决方案的介绍,可以看到了微软的一项新技术——代码后置。那么代码内嵌与代码后置各有什么优点?如何使用 ASP. NET 实现这两种技术呢?

(1)代码后置。代码后置是 ASP. NET 常用的编码方式。使用这种方式编写的 ASP. NET 网页由页面文件(扩展名为. aspx)和代码文件(扩展名为. cs)组成。其中页面文件一般负责设计页面控件和样式等 HTML 代码,而代码文件负责编写事件等服务器代码。这种方式的好处是,页面内容和代码相分离,代码更清晰,程序可读性更好。

下面以实例来演示代码后置的用法。在前面的 Default. aspx 页面文件中加入 HTML 代码,具体代码如下。

```
<asp:TextBox ID="txtAdd1" runat="server" Width="40px"></asp:TextBox>+
<asp:TextBox ID="txtAdd2" runat="server" Width="42px"></asp:TextBox>=
<asp:TextBox ID="txtSum" runat="server" ReadOnly="True" Width="45px">
</asp:TextBox>
<asp:Button ID="btnOK" runat="server" Text="计算"/>
```

页面设计好了之后,双击"计算"按钮,为按钮添加单击事件,则会在 Default. aspx. cs 文件中添加相应的方法,在该方法中添加如下的后置代码:

```
protected void btnOK_Click(object sender,EventArgs e)
{
    int add1=int.Parse(txtAdd1.Text.Trim());
    int add2=int.Parse(txtAdd2.Text.Trim());
    txtSum.Text=(add1+add2).ToString();
}
```

运行 MyWeb 应用程序,浏览 Default. aspx 页面,在第一个文本框中输入"13",在第二个文本框中输入"22",单击"计算"按钮,运行结果如图 1-7 所示。


图 1-7　代码后置示例网页

(2)代码内嵌。顾名思义,代码内嵌就是将服务器代码嵌入到页面文件中,这时需要将代码嵌入到“<%”和“%>”之间。这样做的好处是,不需要额外的文件来编写服务器代码,而只需要一个页面文件即可。下面用一个例子来说明代码内嵌的用法。

首先选择“网站”→“添加新项”命令(或者右击资源管理器中的网站,在弹出的快捷菜单中选择“添加新项”命令),在弹出的“添加新项”对话框中选择“Web 窗体”选项,修改文件名称为“ShowDate. aspx”,并取消选中“将代码放在单独的文件中”复选框,然后单击“添加”按钮,如图 1-8 所示,即可添加新的代码内嵌页面。


图 1-8　“添加新项”对话框

在新建的 ShowDate. aspx 页面中添加如下的 HTML 代码。

```
<%@ Page Language="C#" %>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
```

```
        <title>显示当前日期</title>
    </head>
    <body>
        <form id="form1"runat="server">
        <div>
        <script language="C#"runat="server">
            string dateNomw;
        </script>
        <%
            dateNomw=DateTime.Now.Year.ToString()+"年";
            dateNomw=dateNomw+DateTime.Now.Month.ToString()+"月";
            dateNomw=dateNomw+DateTime.Now.Day.ToString()+"日";
            dateNomw="丽丽你好,今天是"+dateNomw;
        %>
        <%=dateNomw %>
        </div>
        </form>
    </body>
</html>
```

浏览 ShowDate.aspx 页面,运行结果如图 1-9 所示。

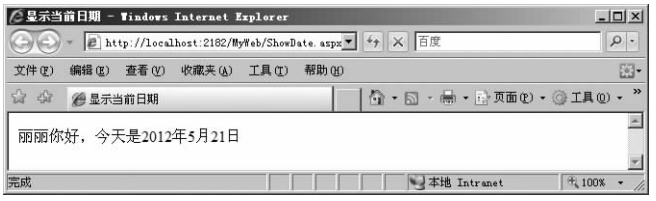


图 1-9　代码内嵌示例页面

学会了如何创建网站和如何编写代码,那么一旦应用程序较大,不能一次完成,下次还需要打开该应用程序该怎么办？有两种方法可以打开网站:一是按前面所讲的目录找到要打开网站的解决方案,双击即可打开;二是打开 Visual Studio 2010,选择“文件”→“打开”→“网站”命令,在弹出的“打开”对话框中选中网站所在的文件夹即可。

## 习　　题

### 一、填空题

1. C#是微软公司发布的一种_____的高级程序设计语言。

2. 在 ASP. NET 中源代码并未直接编译成_____。
3. C# 中的一个字符变量包含的是_____字符。
4. CLR 包括两个组成部分：_____和_____。
5. ASP. NET 将 WinForms 中的_____带入了 Web 应用程序的开发。

## 二、选择题

1. C# 3.0 的运行环境为(　　)。
   A. CLR 1.0　　B. CLR 2.0　　C. CLR 3.0　　D. CLR 4.0
2. 假设 txtUser 是文本框控件的 ID 号，那么用户输入的内容是(　　)。
   A. txtUser. Name　　B. txtUser. Value
   C. txtUser. Caption　　D. txtUser. Text
3. App_Data 文件夹用来存放(　　)。
   A. 图片文件　　B. 样式文件　　C. 数据文件　　D. 配置文件
4. 假设 lblMessage 是标签控件的 ID 号，那么可以设置标签上显示“请输入：”的是(　　)。
   A. lblMessage. Name＝"请输入："　　B. lblMessage. Value＝"请输入："
   C. lblMessage. Caption＝"请输入："　　D. lblMessage. Text＝"请输入："
5. 以下不是 B/S 结构的优点的是(　　)。
   A. 具有分布性特点，可以随时随地进行查询、浏览等业务处理
   B. 业务扩展简单、方便，通过增加网页即可增加服务器功能
   C. 维护简单、方便，只需要改变网页，即可实现所有用户的同步更新
   D. 事务处理能力大

## 三、综合题

1. C# 有几个版本？每个版本有什么新特性？运行环境是什么？
2. 简述代码后置和代码内嵌各有什么优点。
3. 使用代码后置的方法编写一个乘法计算器。
4. 使用代码内嵌的方法显示系统当前的时间。

# 第2章

# C#基础及面向对象技术

ASP.NET 支持的开发语言有 C#、Visual Basic.NET、J#等。C#是微软公司为.NET平台设计的一种全新的、彻底的、面向对象的编程语言，它具有清晰的面向对象语法结构、优秀的编程开发环境和高效的编译、测试和开发工具。C#已经成为.NET 平台下最主要的开发语言，广泛应用于 ASP.NET 项目开发。掌握 C#编程知识是实现 Web 系统的基础，灵活运用 C#将使 Web 系统的开发事半功倍。

**学习目标**

- 掌握 C#的基本数据类型，能根据需要定义适合的变量和常量。
- 掌握运算符和表达式的规范，正确使用运算符和表达式。
- 掌握 C#中基本的程序语句。
- 掌握数组的使用方法。
- 了解面向对象的基本概念。
- 掌握 C#中类、对象、继承和多态的使用方法。

## 2.1 C#语法基础

语法是程序设计的基础，是程序的基本构成部分。和其他高级程序设计语言一样，C#也有其数据类型、变量和常量、运算符和表达式、数组、分支语句和循环语句等，下面分别介绍。

### 2.1.1 数据类型

计算机是按一定的规范来存储和处理数据的，不同的数据有不同的组织形式。数据类型定义了数据的基本存储格式和运算方法。C#是强类型语言，因此每个变量和对象必须具

有声明类型。C# 中的数据类型分为值类型和引用类型。值类型包括数值类型、枚举类型和结构；引用类型包括类、接口、数组和委托。常用的基本数值类型主要有整型、浮点类型、逻辑类型。C# 常见的数据类型如表 2-1 所示。

表 2-1　C# 常见的数据类型

| 类　别 | 数据类型 | 所占字节数 | 说　明 | 取值范围 |
|---|---|---|---|---|
| 整型 | byte | 1 | 8 位的无符号整数 | 0～255 |
| | sbyte | 1 | 8 位的有符号整数，不符合 CLS | －128～127 |
| | short | 2 | 16 位的有符号整数 | －32 768～32 767 |
| | ushort | 2 | 16 位无符号整数 | 0～65 535 |
| | int | 4 | 32 位的有符号整数 | －2 147 483 648～2 147 483 647 |
| | uint | 4 | 32 位无符号整数，不符合 CLS | 0～4 294 967 295 |
| | long | 8 | 64 位的有符号整数 | $-2^{63}$～$2^{63}-1$ |
| | ulong | 8 | 64 位无符号整数，不符合 CLS | 0～$2^{64}-1$ |
| 浮点类型 | float | 4 | 单精度(32 位)浮点数字，7～8 位有效数字 | ±1.5e－45～±3.4e38 |
| | double | 8 | 双精度(64 位)浮点数字，15～16 位有效数字 | ±5.0e－324～±1.7e308 |
| 逻辑类型 | bool | 1 | 布尔值(真或假) | true、false |
| 其他 | char | 2 | Unicode(16 位)字符 | 'A'等 |
| | decimal | 16 | 十进制(128 位)值 | ±1.0e－28～±7.9e28 |
| 类对象类型 | object | — | 对象层次结构的根 | — |
| | string | — | Unicode 字符的不变的定长串 | — |

## 2.1.2　变量和常量

### 1. 变量

变量和常量是程序设计中用来存储数据的单元。变量是程序运行过程中值可以改变的量，常量是程序运行过程中值不变的量。变量的值可以发生变化，但名称保持不变，存储位置不变。变量必须先定义后使用。变量定义的语法格式如下。

```
[访问修饰符] 数据类型 变量名[=变量初始值];
```

例如：

```
int x;              //定义一个整型变量
float y=1.0F;       //定义一个浮点型变量，并赋初值 1.0
```

### 2. 常量

常量是值不变的量，用 const 关键字声明，它可以使代码更容易阅读。常量定义的语法格式如下。

```
[访问修饰符] const 数据类型 常量名=常量;
```

例如：

```
const double PI=3.14159265;          //定义一个常量 PI,其值为 3.14159265
```

## 2.1.3 运算符和表达式

运算符是指定在表达式中执行哪些操作的符号。C# 提供了大量的运算符。

### 1. 运算符

C# 中的运算符包括算术运算符、关系运算符、逻辑运算符、位运算符等，是实现数据处理的基本操作。只有熟练掌握运算符的基本功能，才能更好地解决实际问题。表 2-2 按照优先级的顺序列出了 C# 的运算符及其结合方向。

表 2-2 运算符及其结合方向

| 类 别 | 运 算 符 | 结合方向 |
|---|---|---|
| 初级运算符 | () . [] | 从左至右 |
| 一元运算符 | + - ! ~ ++ -- | 从右至左 |
| 乘除运算符 | * / % | 从左至右 |
| 加减运算符 | + - | 从左至右 |
| 移位运算符 | << >> | 从左至右 |
| 关系运算符 | < > <= >= is as | 从左至右 |
| 等式运算符 | == != | 从左至右 |
| 位运算符 | & ^ \| | 从左至右 |
| 条件运算符 | && \|\| ?: | 从左至右 |
| 赋值运算符 | = *= /= %= += -= <<= >>= &= ^= \|= | 从左至右 |

### 2. 表达式

表达式是符合语法规定的数据和运算符的代码段。

表达式中可以包含文本值、方法调用、运算符及操作数等。例如，有整型变量 a，则可以有赋值表达式 a=5。表达式还可以处理控件的属性值，用来控制程序结果或者显示方法。

表达式的运算顺序按运算符的优先级从高到低进行的。

**【例 2-1】** 表达式的使用。

添加一个新的页面，右击页面，在弹出的快捷菜单中选择“查看代码”命令，然后编写下列代码。

```
protected void Page_Load(object sender,EventArgs e)
{
    int ivar1=1,ivar2=2;
    bool bvar1=true,bvar2=false;
    Response.Write("算数运算<br>");                    //网页输出
    Response.Write("ivar1="+ivar1+",ivar2="+ivar2+"<br>");
    Response.Write("ivar1 乘以 ivar2 的结果是:"+(ivar1*ivar2)+"<br>");
    Response.Write("ivar1 除以 ivar2 的结果是:"+(ivar1/ivar2)+"<br>");
    Response.Write("ivar1 除以 ivar2 的余数是:"+(ivar1%ivar2)+"<br>");
    Response.Write("逻辑运算<br>");
    Response.Write("bvar1="+bvar1+",bvar2="+bvar2+"<br>");
    Response.Write("bvar1 和 bvar2 与运算结果是:"+(bvar1&&bvar2)+"<br>");
    Response.Write("bvar1 和 bvar2 或运算结果是:"+(bvar1||bvar2)+"<br>");
    Response.Write("<br>");
}
```

程序运行结果如图 2-1 所示。

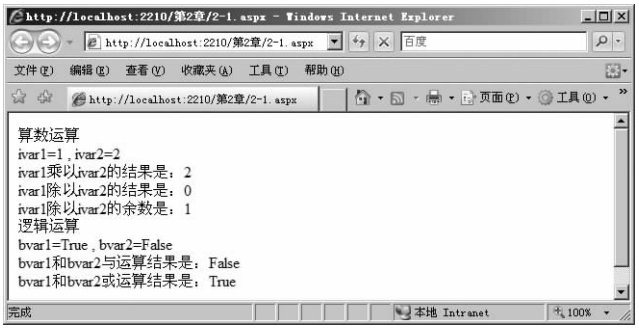


图 2-1　表达式的使用示例程序运行结果

## 2.1.4　数组

C# 中的数组是一种引用数据类型，它由一组相同数据类型的元素构成。数组类型是由抽象类 System.Array 派生而来的。

数组类型是通过指定数组元素的数据类型、数组的维数（秩）和数组的每个维度的上限和下限来定义的。数组元素通过数组名和下标访问。下标是数组指定维数中的一个序号，表示数据元素在该维中的位置。数组下标从 0 开始索引，最后一个下标为数组长度减 1。

### 1. 一维数组

1）一维数组的定义

一维数组的定义格式有以下几种。

格式 1：

```
<数据类型>[] 数组名;
```

格式 2：

```
<数据类型>[] 数组名=new <数据类型>[长度];
```

格式 3：

```
<数据类型>[] 数组名=new <数据类型>[]{数组元素};
```

格式 4：

```
<数据类型>[] 数组名={数组元素};
```

格式 1 仅创建了数组的引用，并没有实例化数组，也就是说，没有创建存储数据的空间，以后要存储数据时必须先开辟存储单元。格式 2 和格式 3 在定义数组的同时，使用 new 语句开辟了存储空间。格式 4 在定义数组的同时，开辟了和数组元素个数相同的存储单元。

例如，定义以下 4 个存储整型数据的数组。

```
int[] iarray1;
int[] iarray2=new int[10];
int[] iarray3=new int[5]{1,2,3,4,5};
int[] iarray4={1,2,3,4,5,6};
```

其中，数组 iarray1 没有任何存储单元，不能存储数据。使用数组 iarray1 之前应先用 new 语句创建存储单元，代码如下。

```
iarray1= new int[5];
```

2)一维数组元素的使用

一维数组通过数组名和下标确定数组元素，格式如下。

```
数组名[下标];
```

例如：

```
iarray2[3]=5;        //给数组 iarray2 下标为 3 的元素赋值为 5
```

3)一维数组的常用属性和方法

数组是一种引用类型，有自己的属性和方法。通过属性和方法可以方便数组的操作。一维数组的常用属性和方法如表 2-3 所示。

表 2-3　一维数组的常用属性和方法

| 类　别 | 名　称 | 说　明 |
|---|---|---|
| 属性 | Length | 获得数组元素的个数 |
| | Rank | 获得数组的维数(秩)，对于一维数组来说，Rank 总是为 1 |
| | GetLength(int) | 获得指定维度的元素个数 |

续表

| 类　别 | 名　称 | 说　明 |
| --- | --- | --- |
| 方法 | Sort | 对一维数组对象中的元素进行排序 |
| | Reverse | 反转一维数组或数组中部分元素的顺序 |

【例 2-2】　一维数组降序输出。

```
protected void Page_Load(object sender,EventArgs e)
{
    int[] iarray=new int[5]{11,2,23,4,15};
    Response.Write("排序前数组元素内容是<br>");
    for(int i=0;i<5;i++)
    {
        Response.Write("第"+i+"下标元素的值是:"+iarray[i]+"<br>");
    }
    Array.Sort (iarray);
    Array.Reverse (iarray);
    Response.Write("排序后数组元素内容是<br>");
    for (int i=0;i<5;i++)
    {
        Response.Write("第"+i+"下标元素的值是:"+iarray[i]+"<br>");
    }
}
```

程序运行结果如图 2-2 所示。


图 2-2　一维数组降序输出示例程序运行结果

### 2. 二维数组

二维数组具有两个维，通常将第一维称为行，第二维称为列。数组中元素的个数为行数乘以列数。

1)二维数组的定义

二维数组的定义格式也有以下几种。

格式 1：

```
<数据类型>[,] 数组名;
```

格式 2：

```
<数据类型>[,] 数组名=new <数据类型>[长度 1,长度 2];
```

格式 3：

```
<数据类型>[,] 数组名=new <数据类型>[长度 1,长度 2]{{元素 1,元素 2,…},{元素 i,元素 j,…},…};
```

格式 4：

```
<数据类型>[,] 数组名= {{元素 1,元素 2,…},{元素 i,元素 j,…},…};
```

二维数组与一维数组一样，创建时用 new 语句分配内存空间。二维数组的两个维度用逗号分开。

2)二维数组元素的使用

二维数组通过数组名和一对下标确定数组元素，格式如下。

```
数组名[下标 1,下标 2];
```

**【例 2-3】** 利用二维数组存储学生成绩，第 1 列存储语文成绩，第 2 列存储数学成绩，第 3 列存储总成绩，输出结果。

```
protected void Page_Load(object sender,EventArgs e)
{
    float[,] score=new float[2,3]{{88,75,0},{87,90,0}};
    for(int i=0;i<2;i++)
      score[i,2]=score[i,0]+score[i,1];
    for (int i=0;i<2;i++)
    {
        Response.Write("第"+(i+1).ToString()+"名同学的成绩是:<br>");
        Response.Write("语文:"+score[i,0]+",数学:"+score[i,1]+",总分:"+score[i,2]+"。");
        Response.Write("<br>");
    }
}
```

程序运行结果如图 2-3 所示。


图 2-3　二维数组的使用示例程序运行结果

## 2.1.5　分支语句

程序中需要根据条件执行不同代码段时，就应使用分支语句。C# 中主要有两种分支语句：if 语句和 switch 语句。

### 1. if 语句

if 语句是最常用的分支语句，程序根据条件的“真”和“假”决定执行的路径。if 语句的格式有以下几种。

格式 1：

```
if(布尔表达式)
    {语句块}
```

格式 2：

```
if(布尔表达式)
    {语句块 1}
else
    {语句块 2}
```

格式 3：

```
if(布尔表达式 1)
    {语句块 1}
else if(布尔表达式 2)
    {语句块 2}
…
else if(布尔表达式 n)
    {语句块 n}
else
    {语句块 n+1}
```

**【例 2-4】**　判断一个整数是不是奇数。

```
protected void Page_Load(object sender,EventArgs e)
```

```
{
    int x=15;
    if (x%2==1)
    {
        Response.Write(x.ToString()+"是奇数");
    }
    else
    {
        Response.Write(x.ToString()+"不是奇数");
    }
}
```

程序运行结果如图 2-4 所示。


图 2-4　if 语句的用法示例程序运行结果

### 2. switch 语句

如果使用 if 语句解决多个并列分支的问题，会出现程序冗长且不直观的情况。C＃使用 switch 语句来处理多分支的条件问题。switch 语句的格式如下。

```
switch(表达式)
{
    case 常量表达式 1:
        {语句块 1}
        break;
    case 常量表达式 2:
        {语句块 2}
        break;
    …
    case 常量表达式 n:
        {语句块 n}
        break;
    default:
        {语句块 n+1}
        break;
}
```

【例 2-5】 将成绩转换成对应的等级。90 分以上为优秀，80～89 分为良好，70～79 分为中等，60～69 分为及格，60 分以下为不及格。

```
protected void Page_Load(object sender,EventArgs e)
{
    int score=85;
    switch(score/10)
    {
        case 10:
        case 9:
            Response.Write("优秀");break;
        case 8:
            Response.Write("良好");break;
        case 7:
            Response.Write("中等");break;
        case 6:
            Response.Write("及格");break;
        default:
            Response.Write("不及格");break;
    }
}
```

程序运行结果如图 2-5 所示。


图 2-5　switch 语句的用法示例程序运行结果

## 2.1.6　循环语句

程序中有些操作是重复进行的，使用循环语句可以简化这些程序。C# 提供了 4 种循环语句结构解决问题，它们分别是 while 语句、do-while 语句、for 语句和 foreach 语句。

### 1. while 语句

while 语句的格式如下。

```
while(布尔表达式)
    {语句块}
```

while 语句的执行顺序如下。

(1)判断布尔表达式的值。
(2)若布尔表达式的值为 true,执行语句块,然后程序跳转到(1)。
(3)若布尔表达式的值为 false,while 语句结束。
以下代码使用 while 语句实现求 1～100 的和。

```
int i=1,sum=0;
while(i<=100)
{
    sum=sum+i;
    i=i+1;
}
```

### 2. do-while 语句

do-while 语句的格式如下。

```
do
    {语句块}
while(布尔表达式);
```

do-while 语句的执行顺序如下。
(1)执行语句块。
(2)判断布尔表达式的值,如果值为 true,就返回(1)继续执行;如果值为 false,就终止循环。
以下代码使用 do-while 语句实现求 1～100 的和。

```
int i=1,sum=0;
do
{
    sum=sum+i;
    i=i+1;
}while(i<=100);
```

注意: 当第 1 次循环的条件为 true 时,while 循环和 do-while 循环的执行过程一样;但当第 1 次循环的条件为 false 时,while 循环不执行循环体,do-while 循环执行一次循环体。

### 3. for 语句

for 语句一般用来执行确定次数的循环。for 语句可以代替 while 语句,使用频率比较高。for 语句的基本格式如下。

```
for([表达式 1];[循环条件];[表达式 2])
    {语句块}
```

for 语句的执行顺序如下。
(1)执行表达式 1,只执行 1 次。

(2)判断循环条件的值,如果值为 true,就转至(3);如果值为 false,就转至(5)。
(3)执行循环体语句块,继续执行(4)。
(4)执行表达式 2,转至(2)。
(5)循环结束。

**【例 2-6】** 求 1～100 的和并输出结果。

```
protected void Page_Load(object sender,EventArgs e)
{
    int i,sum=0;
    for(i=1;i<=100;i++)
    {
        sum=sum+i;
    }
    Response.Write("1 至 100 的和为:"+sum.ToString());
}
```

程序运行结果如图 2-6 所示。


图 2-6　for 语句的用法示例程序运行结果

### 4. foreach 语句

C# 中的 foreach 语句用于对一个数组、字符串等集合类型实例中的每个元素访问一次。语句的格式如下。

```
foreach(元素类型 循环变量 in 元素集合)
    {语句块}
```

下面的代码利用 foreach 语句访问数组中的每一个元素。

```
int[] array={1,2,3,4,5,6};
foreach(int i in array)
    Response.Write(i);
```

### 5. 跳转语句

跳转语句可以控制程序跳转到某个位置,改变原来程序的进程。

1)break 语句

在 switch 语句、while 语句、do-while 语句、for 语句和 foreach 语句中,可以使用 break

语句直接退出当前所在的循环或 switch 语句。

2)continue 语句

在 while 语句、do-while 语句、for 语句和 foreach 语句中,可以使用 continue 语句结束本轮循环,进行下一次循环。

## 2.2 C# 面向对象编程

面向对象编程是流行的软件编程技术之一,应用范围十分广泛。C# 是微软推出的一种完全面向对象的程序设计语言,符合现代软件开发设计,提高了软件开发的效率。C# 实现了封装、继承和多态等多种面向对象技术。

### 2.2.1 面向对象概述

面向对象编程就是按着人们认识客观世界的方式,建立事物及事物与事物之间联系的模型,强调直接以事物为中心来思考问题、认识问题,并根据这些事物的本质特点,把它们抽象地表示为系统中的类,作为系统的基本构成单元,构建完整的现实世界模型。

面向对象编程是使用对象、类、封装、继承、消息等基本概念进行程序设计。面向对象编程是当前软件开发的主流技术,它具有 3 个基本特征:封装、继承和多态。

在面向对象编程中,属性和方法是构成对象的重要因素。属性用来描述对象的静态特征。所谓静态特征是指对象本身的形态,例如,以人为研究对象,对人的描述就有姓名、年龄、性别等特征。方法用来描述对象的动态特征和行为,例如,人有打招呼、跑步等行为。

面向对象编程有以下优点。

(1)面向对象编程以对象为基础,反映了现实世界,符合人的思维习惯,使得程序更易于设计。

(2)面向对象编程实现了封装,实现了数据隐藏,将现实事物属性和行为抽象归纳在一起,把对象作为整体处理。

(3)面向对象编程实现了继承和多态,增加了代码的重用性,易于大型软件的开发。

### 2.2.2 类

类是 C# 语言的基础,C# 中所有代码都在类中,体现了封装性。类是一种引用数据类型,可以包含数据成员和方法成员。

C# 中类的定义格式如下。

```
[修饰符] class <类名> [:基类或接口]
{
    [<修饰符>] <数据成员>
    [<修饰符>] <方法成员>
}
```

C# 只支持单继承，即一个类只能有一个父类。一个类可以实现多个接口。如果一个类声明时继承了一个基类和一个或多个接口，那么基类名称必须写在前面。

类的修饰符可以是 abstract、public、private、internal、partial、sealed 和 static。这些修饰符的含义如表 2-4 所示。

表 2-4　类修饰符的含义

| 修 饰 符 | 说　　明 |
| --- | --- |
| public | 公有类，所有程序都能访问 |
| private | 只有自身成员才能访问 |
| internal | 本程序集（同一命名空间）内部的成员可以访问 |
| partial | 修饰类时表示该类是部分类。C# 允许将一个类分成几部分写在不同的文件中，最终编译时合并成一个文件，但各个部分不能分散在不同的程序集中 |
| abstract | 修饰类时表示该类为抽象类，不能创建该类的实例 |
| sealed | 修饰类时表示该类不能被继承；修饰方法时表示该方法不能被重写 |
| static | 修饰类时表示该类是静态类，不能实例化该类的对象。该类所有成员为静态，只能通过“类. 成员名”的方式访问 |

类的数据成员主要有常量、字段和属性。常量用来存储始终不变的量，在编译时就给定值，通常用 const 修饰。字段用来存储对象的状态数据。虽然 C# 可以定义字段，但与 Java 不同，C# 中字段类的数据成员一般作为类封装的数据，在类的内部使用。属性是对象属性的外在体现，用来进行数据通信。C# 则使用属性存储外部访问的数据，并通过定义 get 和 set 属性访问器控制属性的读和写。

类的方法成员主要有方法、事件、运算器、构造函数等。方法是类中声明的函数，用来描述对象的动态过程。构造函数是和类名相同的函数，创建对象时自动执行，完成对象初始化操作。事件是建立在委托基础上的方法，通过委托实现消息处理。

**【例 2-7】**　下面定义一个 Person 类，其中包含姓名和年龄两个属性以及 SayHello() 方法。

在“解决方案资源管理器”面板中，右击网站，在弹出的快捷菜单中选择“添加新项”命令，然后在弹出的对话框的“文件类型”列表中选择“类”选项，在“名称”文本框中输入文件的名称“Person”，单击“添加”按钮。这时会弹出一个提示对话框，提示是否将文件存储在 App_Code文件夹中，如图 2-7 所示。单击“是”按钮，如果 App_Code 文件夹不存在，系统将自动添加该文件夹，并将类文件 Person. cs 存储在该文件中。


图 2-7　添加类文件提示

类文件代码如下。

```
public class Person
{
    private string name;                    //定义私有字段用于存储姓名
    private int age;                        //定义私有字段用于存储年龄
    public string Name                      //定义 Name 属性用于信息传递
    {
        get{ return name;}
        set{ name=value;}
    }
    public int Age                          //定义 Age 属性用于信息传递
    {
        get{ return age;}
        set{ age=value;}
    }
    public Person(string name,int age)      //构造方法
    {
        this.name=name;
        this.age=age;
    }
    public string SayHello()                //普通公有方法,定义对象后可以调用
    {
        return"你好,我是"+name+",我今年"+age+"岁。";
    }
}
```

### 2.2.3 对象

在面向对象编程中,对象是类的实例。对象就是一个研究对象、现实世界中的一个事物,有具体的属性和行为。用类定义对象的语法如下。

```
类名 对象名=new 类名([参数列表]);
```

**【例 2-8】** 利用【例 2-7】中的类,定义一个对象并调用 SayHello()方法。

创建 2-8.aspx 文件,编写 Load 事件代码如下。

```
protected void Page_Load(object sender,EventArgs e)
{
    Person p1=new Person("徐明",25);
    Response.Write(p1.SayHello());
}
```

程序运行结果如图 2-8 所示。


图 2-8　对象的定义和用法示例程序运行结果

## 2.2.4　继承

继承是面向对象编程中的一个重要的技术，为软件重用提供了基础。如果 B 类继承了 A 类，就说 A 类是基类，B 类是派生类，B 类可以使用 A 类定义过的非私有数据和方法，这就是两个类的继承关系。

C# 中的继承遵循以下规则。

(1)继承是可以传递的。如果 B 类继承自 A 类，C 类继承自 B 类，那么 C 类可以继承 B 类和 A 类的成员。

(2)如果派生类中存在和基类同名的成员，则覆盖基类中的成员。使用 new 关键字修饰派生类中和基类中的同名方法，可以隐藏基类中的方法。

**【例 2-9】** 编写一个 Teacher 类，继承自 Person 类，添加一个职称字段和属性，编写一个 SayHello()方法说明职称。

添加类 Teacher 的代码如下。

```
public class Teacher:Person
{
    private string title;        //职称字段
    public string Title          //职称属性
    {
        get{ return title;}
        set{ title=value;}
    }
    public Teacher(string name,int age,string title ):base(name,age)
    {
        this.title=title;
    }
    public new string SayHello()
    {
        return"你好,我是"+Name+",我今年"+Age+"岁,现职:"+Title;
    }
}
```

创建 2-9. aspx 文件，编写 Load 事件代码如下。

```
protected void Page_Load(object sender,EventArgs e)
{
    Teacher t1=new Teacher("王强",39,"副教授");
    Response.Write(t1.SayHello());
}
```

程序运行结果如图 2-9 所示。


图 2-9　继承的用法示例程序运行结果

## 2.2.5　多态

多态是指同一个操作作用于不同的对象时，产生不同的执行结果的现象。程序中多态的情况有多种，可以通过类中方法的重载实现，也可以通过子类方法对父类方法的覆盖实现。

### 1. 利用方法重载实现多态

方法重载是指在同一个类中有多个同名的方法，这些方法的参数各不相同。参数不同指的是：参数的类型不同、个数不同或传递方式不同。

**【例 2-10】** 在 Teacher 类中编写重载方法 SayHello()。

Teacher 类代码如下。

```
public class Teacher:Person
{
    private string title;        //职称字段
    public string Title          //职称属性
    {
        get{ return title;}
        set{ title=value;}
    }
    public Teacher(string name,int age,string title ):base(name,age)
    {
        this.title=title;
    }
    public new string SayHello()
```

```
    {
        return"你好,我是"+Name+",我今年"+Age+"岁,现职:"+Title;
    }
    public string SayHello(string department)       //方法重载
    {
        return"你好,我是"+Name+",在"+department+"系工作。";
    }
}
```

创建 2-10.aspx 文件,编写 Load 事件代码如下。

```
protected void Page_Load(object sender,EventArgs e)
{
    Teacher t1=new Teacher("王强",39,"副教授");
    Response.Write(t1.SayHello());
    Response.Write("<br>");
    Response.Write(t1.SayHello("信息系"));
}
```

程序运行结果如图 2-10 所示。

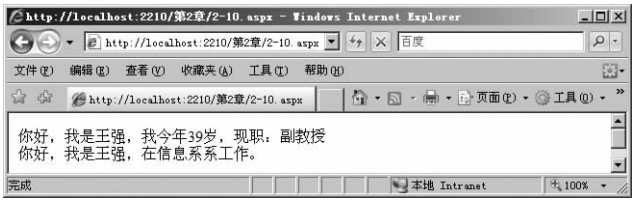


图 2-10　方法重载的使用示例程序运行结果

### 2. 通过方法隐藏实现多态

除了可以用方法重载实现多态外,还可以使用虚方法和方法隐藏来实现多态。这里只介绍通过方法隐藏实现多态的方法。在继承中,派生类中和子类的同名方法可以使用 new 修饰符隐藏基类中的方法,而且 C# 允许基类的引用指向派生类的对象,同样基类引用调用的方法可能产生不同结果。这就是通过方法隐藏实现多态。

**【例 2-11】**　在【例 2-10】的基础上,编写 Student 类继承自 Person 类,隐藏 SayHello() 方法。

Student 类代码如下。

```
public class Student:Person
{
    private int grade;          //年级字段
    public int Grade            //年级属性
```

```
    {
        get{ return grade;}
        set{ grade=value;}
    }
    public Student(string name,int age,int grade):base(name,age)
    {
        this.grade=grade;
    }
    public new string SayHello()
    {
        return"你好,我是"+Name+",我今年"+Age+"岁,在"+grade+"年级。";
    }
}
```

创建 2-11.aspx 文件,编写 Load 事件代码如下。

```
protected void Page_Load(object sender,EventArgs e)
{
    Person p1=new Person("赵伟",30);
    Person p2=new Teacher("王强",39,"副教授");
    Person p3=new Student("张良",19,2);
    Response.Write(p1.SayHello());
    Response.Write(((Teacher)p2).SayHello());
    Response.Write(((Student)p3).SayHello());
}
```

程序运行结果如图 2-11 所示。


图 2-11　方法隐藏的使用示例程序运行结果

## 2.3 异 常 处 理

程序执行过程中,由于软件或硬件的原因导致了不期望或者不需要的事件,称为异常。软件设计者应该对程序运行过程中产生的异常进行处理,进行相关提示,使程序能继续工作。

C#提供的异常处理语句如下。

```
try
{
    //可能产生异常的程序块
}
catch(异常类型 1 异常对象)
{
    //处理异常类型 1 的程序块
}
catch(异常类型 2 异常对象)
{
    //处理异常类型 2 的程序块
}
…
finally
{
    //最终要执行的语句块
}
```

【例 2-12】　利用异常处理监视除法。

```
protected void Page_Load(object sender,EventArgs e)
{
    int dividend=10,divisor=0,result;
    try
    {
        result=dividend/divisor;
        Response.Write(result);
    }
    catch
    {
        Response.Write("<Script Language=JavaScript>alert('产生异常');
</Script>");
    }
    finally
    {
        Response.Write("<Script Language=JavaScript>alert('执行结束');
</Script>");
    }
}
```

程序运行，弹出异常对话框，如图 2-12 所示。


图 2-12　异常对话框

## 2.4 项目实战：存款计算器

本项目主要利用所学知识编写一个复利存款计算器，输入存款本金和存款时间，自动根据设定利率进行计算。

### 2.4.1 实施计划

#### 1. 计算方法分析

利息采用复利计算法，假定存款本金为 $P$，年利率为 $i$，存款时间为 $N$ 年，则本金与利息之和公式为：

$$P(1+i)^N$$

#### 2. 实施步骤

本项目可以分为界面设计和代码编写两个步骤进行。

### 2.4.2 项目实施

(1)建立 ASP. NET 网站，添加 DepositComputor. aspx 页面。

(2)添加控件。添加 2 个文本框，ID 分别是 txtPrincipal 和 txtYear；添加 3 个标签控件，ID 分别是 lblInterestRate、lblInterest 和 lblAmount；添加 1 个按钮控件，ID 是 btnComputor。

(3)编写代码。双击按钮控件，编写单击事件代码如下。

```
protected void btnComputor_Click(object sender,EventArgs e)
{
    double principal,interestrate=0,interest=0,amout=0;
    int year;
    try
    {
        principal=double.Parse(txtPrincipal.Text);
        year=Int32.Parse(txtYear.Text);
        if(year>=1&&year<=5&&year! =4)
        {
            switch(year)
            {
```

```
                    case 1：interestrate＝3.5;break;
                    case 2：interestrate＝4.4;break;
                    case 3：interestrate＝5.0;break;
                    case 5：interestrate＝5.5;break;
                }
                amout＝principal * Math.Pow((1＋interestrate/100),year);
                interest＝amout－principal;
            }
            else
            {
                throw new Exception("年限不对");
            }
        }
        catch
        {
            Response.Write("＜Script Language＝JavaScript＞alert('输入数据不对');
＜/Script＞");
        }
        finally
        {
            lblInterestRate.Text＝interestrate.ToString();
            lblInterest.Text＝string.Format("{0,10:c}",interest);
            lblAmount.Text＝string.Format("{0,10:c}",amout);
        }
    }
```

### 2.4.3　项目测试

运行测试，修改错误。程序运行结果如图 2-13 所示。


图 2-13　存款计算器运行结果

# 习　　题

## 一、填空题

1. C# 中所有的数据类型可以归纳为两种：一种是_____类型；另一种是_____类型。

2. C# 中分支语句有_____语句和_____语句。

3. 执行循环时，利用_____语句可以跳出循环。

4. 面向对象的三大特征是_____、_____和_____。

5. C# 中捕获异常的语句是_____。

## 二、选择题

1. 表达式 5%3 和－5%3 的值为(　　)。

A. 2　－1　　B. 2　－2　　C. 2　2　　D. 2

2. 面向对象技术特征不包含(　　)。

A. 消息　　B. 继承　　C. 多态　　D. 封装

3. C# 用(　　)进行对象状态的描述。

A. 封装　　B. 消息　　C. 类　　D. 方法

4. 下面程序代码的运行结果是(　　)。

```
int i=0;
while(i<5)
{
    if(i<5) break;
}
Response.Write(i);
```

A. 0　　B. 1　　C. 5　　D. 死循环，无结果

5. 下面关于数组的描述错误的是(　　)。

A. 数组中的元素可以是任何类型

B. 数组可以是一维的，也可以是二维的

C. 数组长度指的是数组元素的个数

D. 语句“int []array;”定义了一个数组，可以存储整型数据

6. 下面关于封装的说法中，错误的是(　　)。

A. 封装是保护内部数据的一种方法

B. 封装避免内部数据被恶意访问

C. 类的所用成员都定义成私有的，有助于起到封装的作用

D. 封装是尽可能隐藏类内部的细节

## 三、综合题

1. 程序控制语句有哪几种？
2. 简单描述面向对象编程的优点。
3. 编写程序，求 1！＋2！＋3！＋…＋$n$！的值，$n$ 的值由用户输入。
4. 定义一个矩形类，其中包含长和宽两个属性，求周长和面积两个公有方法。

# 第3章

# 页面设计技术及导航

Web 开发过程中除了功能实现之外，还有一个非常重要的环节，即站点界面的设计。本章主要介绍站点界面设计的相关知识。ASP. NET 中提供了丰富的解决方案。母版页常用于网站各页面中一致内容的显示，如页眉、页脚等。每个网站都需要导航，本章介绍的导航控件就可以实现“面包屑”导航和树型目录导航。

## 学习目标

- 掌握使用 CSS 布局的方法。
- 能够使用样式生成器生成常用样式。
- 掌握主题的使用方法。
- 会使用母版页控制页面风格。
- 能使用站点地图设置“面包屑”站点导航。
- 能使用站点地图和 XML 文件控制树型导航。
- 能使用代码动态生成树型导航。

## 3.1 页面设计技术

随着网站开发技术的快速发展，CSS 技术也有长足的进步。特别是 HTML 5 技术推出之后，CSS 技术受到各大计算机软件业巨头的追捧，如微软、谷歌、Adobe 等公司都宣布重点投入该领域，这也进一步奠定了 CSS 技术的重要地位。通过定义 CSS，能让网页具有美观一致的界面，可以将网页设计得更加绚丽多彩。一个样式文件可以作用于多个页面，具有更好的易用性和扩展性。通过修改样式文件，能设计出内容相同而外观不同的多种风格页面。

## 3.1.1　CSS 的应用

### 1. CSS 的概念

级联样式表(cascading style sheet,CSS)又称“风格样式表”,用来进行网页风格设计。例如,让链接文字未被单击时是蓝色的,当用鼠标指针指向它时文字变成红色的且有下划线,这就是一种风格。通过设立 CSS,可以统一地控制 HTML 中各标志的显示属性。CSS 可以使人更有效地控制网页外观。

按照样式表定义的位置,可将样式表分为以下 3 种类型。

(1)全局样式。即将样式放在一个独立的 CSS 文件中,所有的 aspx 网页都可以使用,使用时需要在<head>下引用。例如,有以下代码:

```
<link rel="stylesheet" type="text/css" href="css/css.css">
```

(2)页面样式。即将样式放在头部<head>的位置,在本页中的元素都可以使用。例如,有以下代码:

```
<Style type="text/css">
    P{
        color:red;
        font-size:30px;
        font-family:隶书;
     }
</Style>
```

(3)行内样式。即只对当前的控件有效。例如,有以下代码:

```
<body>
    <p style="color:red;font-size:30px;font-family:隶书;">我的 ASP.NET
应用程序</p>
</body>
```

当用户只需要定义当前网页的样式时,可使用页面样式表。页面样式表是一种级联样式表,“嵌”在网页的<head>标记符内。如果要单独设置某个控件或<html>标记的样式,则应该使用行内样式。

如果一个页面同时使用了这 3 种类型的样式,那么它们之间的优先级是:行内样式的优先级最高,页面样式其次,全局样式最低。

### 2. CSS 的定义

CSS 的定义是由三部分构成:选择器(selector)、属性(properties)和属性的取值(value)。

语法格式如下。

```
selector
{
```

```
    property:value;
}
```

需要说明的是，选择器可以有多种形式，一般是要定义样式的 HTML 标记，如 body、p、table 等。

可以通过以上语法定义某个 HTML 标记的属性和值，属性和值要用冒号隔开，例如：

```
body{color:black}          /* 此例的效果是使页面中的文字为黑色 */
```

如果属性的值是由多个单词组成的，那么必须在值上加引号。例如，字体的名称经常是几个单词的组合：

```
p{font-family:"sans serif"}          /* 此例定义段落字体为 sans serif */
```

如果需要对一个选择器指定多个属性，可以使用分号将所有的属性和值分开，例如：

```
p{text-align:center;color:red}     /* 此例定义段落居中排列，且段落中的文字
                                      为红色 */
```

### 3. CSS 选择器

1）HTML 标记选择器

把 HTML 的标记名作为选择器，表示要将样式规则应用于这类 HTML 标记上。

下面举例说明 HTML 标记选择器的用法。在下面的代码中，样式选择器用的是 p，则该样式将对该网页中所有的 标签之间的内容起作用。

**【例 3-1】** HTML 标记选择器的用法示例。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>HTML 标记选择器</title>
    <style type="text/css">
        p{ color:red;
        font-family:"隶书";
        font-size:24px;}          /* 字体为红色，隶书，字体大小为 24px */
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <H2>静夜思</H2>
        <P>床前明月光，</P>
        <P>疑是地上霜。</P>
        <P>举头望明月，</P>
        <P>低头思故乡。</P>
```

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

2)class 选择器

要将一种 HTML 标记的各个网页元素分成几类,可以将这一种 HTML 标记的每个标记的 class 属性设置为不同的值,然后在样式表中分别为这一类的 HTML 标记的各个 class 属性定义不同的样式。

**【例 3-2】** class 选择器的用法示例。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1"runat="server">
    <title>class 选择器</title>
    <style type="text/css">
        p.r { color:red;font-size:24px;}
        p.b { color:Blue;font-size:14px;}
        p.g { color:Green;font-size:30px;}
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <H2>静夜思</H2>
        <p class="g">床前明月光,</p>
        <p class="r">疑是地上霜。</p>
        <p class="b">举头望明月,</p>
        <p>低头思故乡。</p>
    </div>
    </form>
</body>
</html>
```

3)ID 选择器

ID 属性是 HTML 元素的唯一标识。CSS 中的 ID 选择器就是为了给网页中某一特定且唯一 ID 属性值的 HTML 元素应用该样式。定义 ID 选择器时,要在 ID 名称前加一个“#”。

**【例 3-3】** ID 选择器的用法示例。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
```

```
    <title>ID 选择器</title>
    <style type="text/css">
        .myStyle { color:red;font-size:24px;}
        #divStyle{ background-color:#e4e0e0;padding:10px;border:1px solid;}
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div id="divStyle">
            <H2>静夜思</H2>
            <p class="mystyle">床前明月光,</p>
            <p class="mystyle">疑是地上霜。</p>
            <p class="mystyle">举头望明月,</p>
            <p>低头思故乡。</p>
        </div>
    </form>
</body>
</html>
```

4)混合选择器

在实际的 Web 项目样式设计中,往往会有将不同类型的选择器混合使用的情况。既有 class 选择器和 HTML 标记选择器的混合使用、HTML 标记多层的混合使用,也有 ID 选择器与 HTML 标记选择器的混合使用。混合使用可以使样式看起来更有层次感、更清晰、更易维护。它们之间是包含与被包含的关系。

**【例 3-4】** class 选择器与 HTML 标记选择器的混合用法示例。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>class 选择器与 HTML 标记选择器混合</title>
    <style type="text/css">
        div.myStyle { font-size:24px;}
        div.myStyle p{ color:Red;}
        div.myStyle H2{ color:Blue;}
    </style>
</head>
<body>
    <form id="form1" runat="server">
        <div class="myStyle">
            <H2>静夜思</H2>
```

```
            <P>床前明月光,</P>
            <P>疑是地上霜。</P>
            <P>举头望明月,</P>
            <P>低头思故乡。</P>
        </div>
    </form>
</body>
</html>
```

5)伪元素选择器

伪元素选择器是一种定义同一个 HTML 元素的各种状态和其所包括的部分内容的方式。最常见的链接伪元素有以下几个。

(1)a:link:指示未访问过的链接的样式。

(2)a:active:指示鼠标按下时链接的样式。

(3)a:hover:指示鼠标移动到链接上的样式(IE 6.0 只有 a 标记支持此伪类)。

(4)a:visited:指示已访问过的链接的样式。

### 4. 在 ASP.NET 下创建 CSS

对于开发者来说,在 ASP.NET 下创建 CSS 样式是一件非常简单的事。对于精通 CSS 的开发人员,可以直接在页面的 HTML 标记中添加相应的 CSS 样式。如果对 CSS 不太了解或太不熟练也没有关系,Visual Studio 2010 给用户提供了非常方便的 CSS 样式生成器。针对前面介绍的 CSS 样式的 3 种不同的类型,Visual Studio 2010 提供了不同的 CSS 样式生成方式。

1)全局样式

如果用户欲将样式单独存在样式文件中,可以使用下面的方法来生成样式。首先,建立一个网站,并打开这个网站。然后,选择“文件”→“新建”→“文件”命令,在弹出的“添加新项”对话框中选择“样式表”选项,输入文件名后单击“添加”按钮即可。

打开新建的样式文件,文件中会有一个自动生成的 body 的空样式,如果不需要可以删除。在样式文件的空白处右击,在弹出的快捷菜单中选择“添加样式规则”命令,弹出“添加样式规则”对话框,如图 3-1 所示。

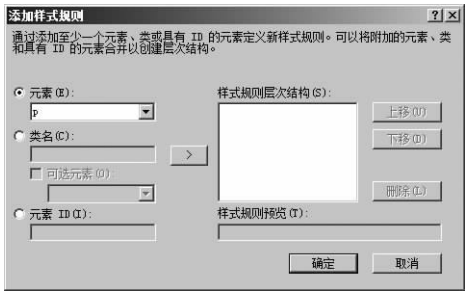


图 3-1　“添加样式规则”对话框

在“添加样式规则”对话框中可以添加前面介绍过的不同类型的样式选择器，其中，“元素”选项对应 HTML 标记选择器和伪元素选择器，“类名”选项对应 class 选择器和混合选择器，“元素 ID”选项对应 ID 选择器。选择要添加的选择器类型，输入名称，单击“确定”按钮即可添加成功。

这时只是添加了样式名称，下面要对样式进行具体设置。将鼠标移动到刚才添加的“p”样式的两个大括号之间并右击，在弹出的快捷菜单中选择“生成样式”命令，在弹出的“修改样式”对话框中可以进行样式的具体设置，如图 3-2 所示。设置完成后，系统会自动生成下面的样式代码。

```
p
{
    font-size:20px;
    font-style:italic;
    color:#FF0000;
}
```

这给对样式不熟悉的开发者提供了非常方便的生成样式的方法。


图 3-2 “修改样式”对话框

2)页面样式和行内样式

打开已有的网站，在“解决方案资源管理器”面板中打开要添加样式的文件。选择“视图”→“CSS 属性”命令，打开“CSS 属性”面板，然后就可以进行页面的样式设计了，如图 3-3 所示。

在设计区域中，将光标移动到想添加样式的 HTML 标记上或选中该标记，如选中＜div＞标记；然后，在左边的“CSS 属性”列表框中右击，在弹出的快捷菜单中选择“新建样式”命令，打开样式生成器，如图 3-4 所示。读者会发现，此样式生成器与图 3-2 中所示类似，只是增加了“选择器”下拉列表框，同时在“定义位置”下拉列表框中包含不同的选项，可以对当前页面进行设置，也可以新建样式表，还可以对已有的样式表进行修改。

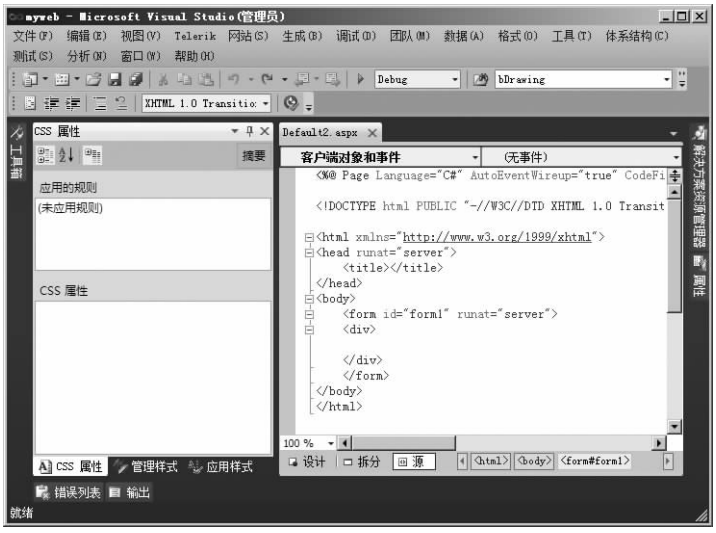


图 3-3　页面样式设计


图 3-4　样式生成器

按照图 3-4 所示进行样式设置，设置完成后单击“确定”按钮，生成图 3-5 所示的样式代码。如果还选中了图 3-4 中的“将新样式应用于文档选择内容”复选框，样式则会自动引用到刚才选中的 标记中。

如果需要修改或删除样式，可以在左边的“CSS 属性”列表框中进行。

选择图 3-5 中左边的“管理样式”标签，在其中可以方便地管理当前页面的样式，包括样式的新建、修改、删除等，还可以附加样式表，如图 3-6 所示。

选择图 3-5 中左边的“应用样式”标签，可以对当前页面的任何可以使用样式的元素进行样式应用。方法是，先将光标移动到需要使用样式的标记上，然后在“应用样式”选项中选择要使用的样式，则系统会自动引用选中的样式。当然，如果对已经使用的样式不满意，可以使用“清除样式”功能来取消已经使用的样式，如图 3-7 所示。

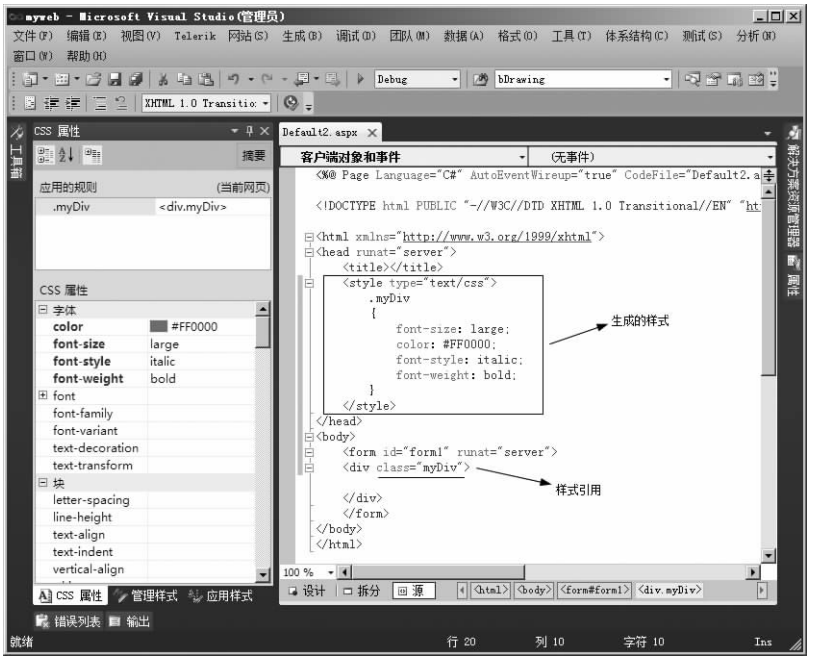


图 3-5　页面生成样式


图 3-6　“管理样式”标签


图 3-7　“应用样式”标签

### 5. 盒子模型

盒子模型是 CSS 中的经典模型，如图 3-8 所示。内容(content)、填充(padding)、边框(border)、边界(margin)，都是 CSS 盒子模式具备的属性。

盒子模型最里面的部分是实际的 content。直接包围 content 的是 padding，padding 呈现了 content 的背景。padding 的边缘是 border。border 以外是 margin，margin 默认是透明的，因此不会遮挡其后的任何元素。

在 CSS 盒子模型中，width 和 height 指的是 content 区域的宽度和高度。增加 padding、border 和 margin 的 width 与 height 不会影响 content 区域的尺寸，但是会增加 box 的总尺寸。假设框的每个边上有 10 px(像素)的 margin 和 5 px 的 padding，如果希望这个元素框

达到 100 px，则需要将内容的宽度设置为 70 px，CSS 代码如下。

```
#box{
    width:70px;
    margin:10px;
    padding:5px;
}
```

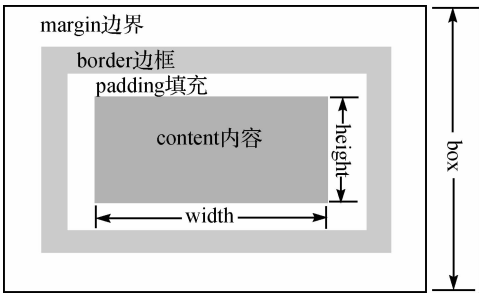


图 3-8　盒子模型

部分元素尺寸示例如图 3-9 所示。

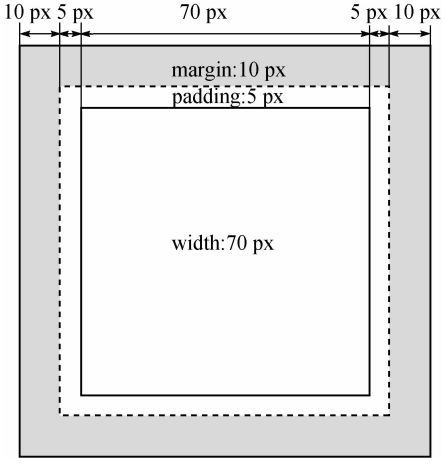


图 3-9　部分元素尺寸示例

(1)padding 属性，用来定义元素的内边距。padding 属性接受长度值或百分比值，但不允许使用负值。

例如，如果希望所有 h1 元素的各边都有 10 px 的内边距，则只需定义下面的代码：

```
h1{padding:10px;}
```

还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值。例如：

```
h1{padding: 10px 0.25em 2ex 20%;}
```

padding 属性包括 4 个子属性：padding-top、padding-right、padding-bottom、padding-left。下面的规则实现的效果与上面的简写规则实现的效果是完全相同的。

```
h1{
    padding-top: 10px;
    padding-right: 0.25em;
    padding-bottom: 2ex;
    padding-left: 20%;
}
```

(2)margin 属性。它是设置外边距的最简单的方法。该属性接受多种长度单位，这些长度单位可以是 px(像素)、in(英寸)、mm(毫米)或 em(字体尺寸单位)。还可以设置margin 为 auto。但常见的做法是为外边距设置长度值。

下面的代码为 h1 元素的 4 个边分别定义了不同的外边距：

```
h1{margin:10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距(top)开始围着元素顺时针旋转的：

```
margin:top right bottom left
```

margin 的默认值是 0，所以如果没有为 margin 声明值，就不会出现外边距。但是在实际中，浏览器对许多元素都提供了预定的样式，外边距也不例外。例如，在支持 CSS 的浏览器中，系统会在每个段落元素的上面和下面生成“空行”。因此，如果没有为 p 元素声明外边距，浏览器可能会自己应用一个外边距。当然，只要特别作了声明，就会覆盖默认样式。

有时在设计时会输入一些重复的值，例如：

```
p{margin:0.5em 1em 0.5em 1em;}
```

通过值复制，可以不必重复地输入这对数字。上面的规则与下面的规则是等价的：

```
p {margin:0.5em 1em;}
```

这两个值可以取代前面的 4 个值。这是如何做到的呢？CSS 定义了一些规则，允许为外边距指定少于 4 个的值，规则如下。

①如果缺少左外边距的值，则使用右外边距的值。

②如果缺少下外边距的值，则使用上外边距的值。

③如果缺少右外边距的值，则使用上外边距的值。

图 3-10 提供了更直观的方法来理解这一点。

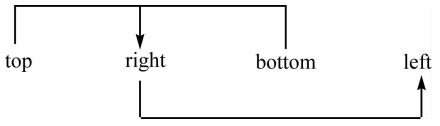


图 3-10　margin 的值复制规则

## 3.1.2　主题和皮肤

一个网站的界面是否美观，直接影响着站点的受欢迎程度。随着网站的快速发展，人们平时上网时更倾向于访问界面美观的站点。这要求开发人员在设计 Web 应用时，重视功能实现的同时，也要重视用户界面的设计。现在程序员在进行 Web 界面设计时都使用 CSS，而 ASP. NET 中也为设计人员提供了一种更简单、更易操作的实现网站美观的控制技术——主题(theme)。它是 Microsoft ASP. NET 2. 0 中的一项新增功能，使用此功能可以一次定义一组控件的外观，并可以将该外观应用于整个 Web 应用程序。例如，通过利用主题功能，可以在一个中心位置为应用程序中的所有 TextBox 控件定义共同的外观，如背景颜色和前景颜色。使用主题功能可以轻松地建立并维护整个网站外观的一致性。

### 1. 主题概述

1)主题的概念

主题由一组元素组成：外观文件、级联样式表、图像和其他资源，主题中至少包含外观。主题是在网站或 Web 服务器上的特殊目录 App_Themes 中定义的。主题是一组服务端控件的属性设置的集合，提供一种简单的方法来设置控件的样式属性。主题具有以下特性。

(1)主题只在 Web Control 中有效。

(2)主题可以设置在内容页面上，但不能设置在母版页(master page)上。

(3)主题上设置的服务端控件的样式覆盖页面上设置的样式。

(4)如果在页面上设置 Enable Theming＝"false"，主题无效。

(5)要在页面中动态地设置主题，就必须在页面生命周期 Page_Preinit 事件之前。

(6)主题包括 Skin 文件和 CSS 文件。

2)主题与 CSS

CSS 是 Web 界面设计中的主要技术，可以定义丰富的外观样式，在 ASP. NET 中与主题配合使用，能够实现更加丰富的界面效果，效率也非常快，特别是对服务端控件方面的设置。CSS 规则是有缺点的，虽然 CSS 可以设定相应元素的显示细节(如字体、边框、前景和背景图片等)及页面布局，但它只限于一组固定的样式特性。CSS 对 ASP. NET 服务端样式控制的不足，使用主题可以得到很好地弥补。例如，如果希望在定义服务端控件样式的同时，也定义控件的部分行为事件，这时主题就是一个不二的选择。和 CSS 样式类似，主题也可以定义一组控件的样式特性，并且可以应用到多个页面。

主题与 CSS 的主要区别如下。

(1)CSS 由客户端浏览器解释，主题是在服务端实现的。当主题作用到页面时，格式化后的最终页面被发送到用户浏览器。

(2)主题基于控件而不是 HTML。

(3)可以使用配置文件来应用主题。这样就能在不修改任何页面的情况下，对整个网站的主题进行调整和修改。

(4)主题不能像 CSS 那样级联，如果在一个主题和控件中同时设定了一个属性，那么主题中定义的值将覆盖控件的属性。如果要改变这样的规定，应提高页面属性的优先级。

3)Skin 文件

Skin 文件是外观文件或皮肤文件,它包含各个控件(如 Button、Label、TextBox、Calendar 控件等)的属性设置。在 Web 项目下的主题文件夹中可以包含一个或多个 Skin 文件,不同控件的样式都包含在 Skin 文件中。当然,也可以为每个控件单独定义一个 Skin 文件,这要视情况而定。页面使用 Skin 文件中的控件皮肤样式时,可以使用 SkinId 来确定。它能为同一个控件在不同页面上定义不同的皮肤样式。

例如,下面是 Button 控件的控件外观设置:

```
<asp:Button SkinId="buttonSkin" runat="server" BackColor="#669999"
BorderColor="White" ForeColor="White" Text="Button"/>
```

使用 Skin 文件时需要注意以下几点。

(1)Skin 文件中的控件样式必须包含 runat="server"属性。

(2)扩展名必须是. skin。

(3)在定义控件皮肤样式时,只能包含属性,不能包含 ID 属性。

4)主题文件组织结构

如前面所讲,每个主题下包含许多类型的文件,如皮肤文件、CSS 文件、脚本文件、资源文件、图片文件等。在 Web 项目下,主题默认保存在 App_Themes 文件夹下。对于有多个主题的 Web 项目可以建立多个主题文件夹,并将不同主题的皮肤等资源分别管理。

如图 3-11 所示,在 App_Themes 目录下包括 3 个子目录 Blue、Default 和 Red,说明这个 Web 项目中定义了 3 个主题。每个主题目录下都包括一个皮肤文件和一个 CSS 文件。多个主题可以让用户随时体验不同的界面风格,会产生很好的交互效果。


图 3-11　主题目录结构 1

常见的主题目录中文件的命名方式有以下 3 种。

(1)根据 Web 项目或网站风格,每个主题文件夹中都包含相同的 SkinId 的多个控件外观定义,见图 3-11。这种方式比较常见,适合页面较多、比较复杂的情况。

(2)根据控件类型来组织主题下的皮肤文件,如图 3-12 所示。每个皮肤文件中只定义一种控件的一组样式。这种方式适合比较简单的 Web 项目或网站。

(3)根据网页名称确定皮肤文件的名称。如图 3-13 所示,每个皮肤文件中定义对应网页中控件的样式信息。当然,有多少个页面,就需要定义多少个皮肤文件。这种方式可以在页面比较少的时候使用。


图 3-12　主题目录结构 2


图 3-13　主题目录结构 3

## 2. 创建主题

使用 Visual Studio 2010 创建主题非常简单，当然，如果想设计出既好看又实用的主题需要一定的艺术功底。下面将对主题的建立与应用的步骤作详细介绍。

1)添加 App_Themes 文件夹

打开“解决方案资源管理器”面板，右击项目名称，在弹出的快捷菜单中选择“添加 ASP.NET文件夹”→“主题”命令。如果项目中不存在 App_Themes 文件夹，系统会自动添加，并在 App_Themes 下添加一个主题文件夹；如果已经有 App_Themes 文件夹，就直接将主题文件夹添加到 App_Themes 下，并将其改名为“Default”，如图 3-14 所示。


图 3-14　添加主题

2)添加 Skin 文件

右击图 3-14 中的“Default”文件夹，在弹出的快捷菜单中选择“添加新项”命令，在弹出的对话框中选择“外观文件”选项，并输入名称“SkinFile.skin”，然后单击“添加”按钮。

打开 SkinFile.skin 文件，会看到对外观模板的介绍信息以及外观的示例代码，信息如下。

```
    默认的外观模板。以下外观仅作为示例提供。
    命名的控件外观。SkinId 的定义应唯一，因为在同一主题中不允许一个控件类型有重复的 SkinId，如：
    <asp:GridView runat="server" SkinId="gridviewSkin" BackColor="White">
        <AlternatingRowStyle BackColor="Blue"/>
    </asp:GridView>
    默认外观。未定义 SkinId。在同一主题中每个控件类型只允许有一个默认的控件外观。
    <asp:Image runat="server" ImageUrl="~/images/image1.jpg"/>
```

下面就可以添加控件的外观代码了。如果对相关的外观代码非常熟悉，可以直接编写外观代码，但是这种方式下系统没有提供控件属性的智能化提示功能。编写控件的外观代码文件有一种既方便又不容易出错的方法，即先在普通的 Web 页面中添加相应的控件，然后设置属性，最后将生成好的属性代码复制到 Skin 文件中，再进行相应修改就可以了，具体步骤如下。

(1)在 Web 项目或网站中随意打一个页面文件，如果没有，则添加一个页面文件，如

demo. aspx。打开该文件并切换到设计视图。

(2)在 demo. aspx 文件中添加控件,并设置各控件的相应属性,如字体、颜色、边框、字体大小、背景色、前景色、宽度和高度等。为了方便,可以将每个控件放在单独的行中,如图 3-15 所示。

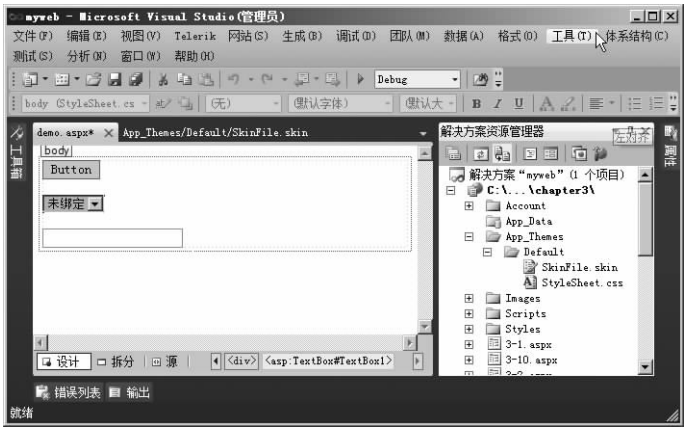


图 3-15　控件外观设置

(3)将上面设置控件属性操作生成的外观代码复制到 SkinFile. skin 文件中。在复制时应注意,一定要复制完整代码。可以先在设计视图中选中一个控件,然后切换到源视图中,系统会自动选中该控件的所有外观代码,然后复制即可。

(4)打开 SkinFile. skin 文件,修改每个控件的 ID 属性为 SkinID,然后修改其值。当然,如果控件少,则可以直接删除 ID 属性。最终定义好的 SkinFile. skin 文件内容如下。

```
<asp:TextBox SkinID="txtMainSkin" runat="server" BorderColor="#999999"
        BorderStyle="Solid" BorderWidth="1px" Height="20px"></asp:TextBox>
<asp:Button SkinID="btnMainSkin" runat="server" BorderColor="#3366FF"
        BorderStyle="Solid" BorderWidth="1px" Height="20px" Text="Button">
</asp: Button>
<asp:DropDownList SkinID="ddlMainSkin" runat="server" BackColor="Silver"
        Height="22px"></asp:DropDownList>
```

3)添加 CSS 文件

在正常的 Web 项目或网站开发时,样式与主题往往是同时使用的,各取所长,优势互补。这里使用 CSS 来控件页面的背景颜色、字体大小、DIV 居中对齐等。添加 CSS 文件的具体步骤如下。

(1)在 Web 项目中添加 StyleSheet. css 文件,添加方式与添加主题是一样的。打开样式文件,默认有一个空的 Body 样式规则,代码如下:

```
body{ }
```

(2)在 body 后的一对大括号中间右击,在弹出的快捷菜单中选择"生成样式"命令,在弹出的"修改样式"对话框中设置"字体"选项中的 font-size 为"4px","背景"选项中的 back-

ground-color 为“＃F1F1F1”。生成如下样式：

```
body
{
    font-size:14px;
    background-color:＃F1F1F1;
}
```

(3)在空白处右击，在弹出的快捷菜单中选择“添加样式规则”命令，在弹出的对话框中选中“类名”单选按钮，并输入“topDiv”，然后单击“确定”按钮。在新建样式规则后的大括号中右击，在弹出的快捷菜单中选择“生成样式”命令，在弹出的“修改样式”对话框中设置“方框”选项中的 margin 为“auto”，“定位”选项中的 width 为“1002px”。生成如下样式代码：

```
.topDiv
{
    margin:auto;
    width:1002px;
}
```

(4)重复上面的步骤，添加新的样式规则 contentDiv，生成样式代码如下：

```
.contentDiv
{
    margin:auto;
    width:602px;
}
```

### 3. 应用主题及样式

1)给页面添加主题

在“解决方案资源管理器”面板中打开要使用主题的页面文件，也可以新添加一个页面文件，如“ThemeDemo. aspx”。切换到页面的设计视图下，在空白处右击，在弹出的快捷菜单中选择“属性”命令，在弹出的“属性”面板顶部的下拉列表框中选择 DOCUMENT 选项，在下面的列表框中选择 StyleSheetTheme 选项，设置其值为 Default，如图 3-16 所示。然后该页面就可以使用该主题了。


图 3-16　给页面引用主题

2)给页面添加样式

打开 ThemeDemo. aspx 文件，切换到源视图，在＜title＞标记下加入样式表的引用，代码如下：

```
＜link type＝"text/css" href＝"App_Themes/Default/StyleSheet. css"/＞
```

3)页面中使用主题与样式

打开 ThemeDemo. aspx 文件,打开工具箱,在页面上放置两个<DIV>标记。右击其中一个 DIV 标记,在弹出的快捷菜单中选择“属性”命令,然后在“属性”面板中修改 Class 的属性的值为 topDiv,如图 3-17 所示。使用同样的方法,修改另一个 DIV 标记的 Class 属性的值为“contentDiv”。

在第一个 DIV 标记中添加一个图片,代码如下:

```
<asp:Image ID="Image1" runat="server" ImageUrl="~/Images/topImage.jpg"/>
```

在第二个 DIV 标记中添加 4 个 TextBox 控件、2 个 Button 控件和 1 个 DropDownList 控件。选中 TextBox 控件,在“属性”面板中将 SkinID 属性设置成 txtMainSkin,如图 3-18 所示。将 Button 控件的 SkinID 属性设置成 btnMainSkin;将 DropDownList 控件的 SkinID 属性设置成 ddlMainSkin。

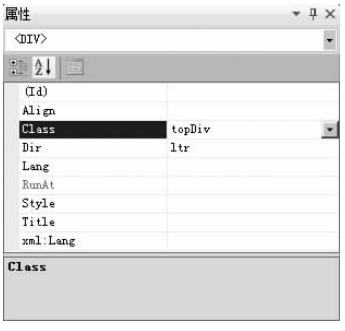


图 3-17 DIV 使用样式

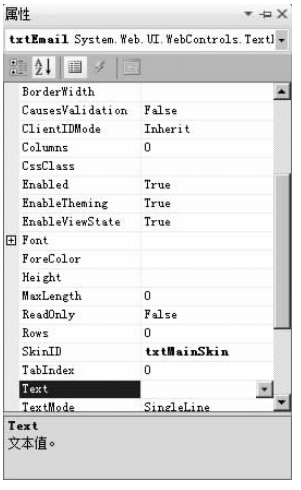


图 3-18 设置控件的 SkinID 属性

设计效果如图 3-19 所示。

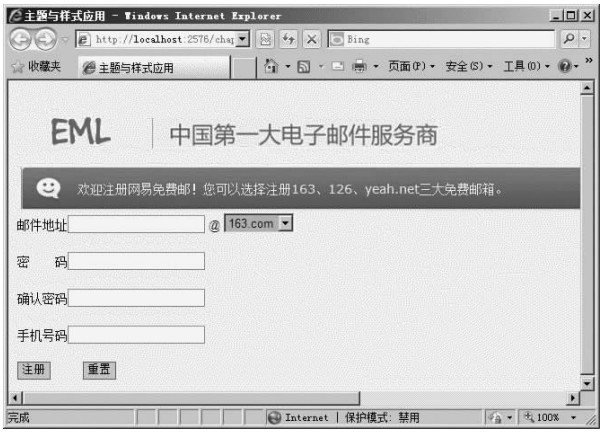


图 3-19 设计效果

ThemeDemo.aspx 对应的完整界面代码如下。

```
    <!--StyleSheetTheme 指定要使用的主题-->
    <%@ Page Language="C#" AutoEventWireup="true" CodeFile="ThemeDemo.
aspx.cs" Inherits="ThemeDemo" StyleSheetTheme="Default" %>
    <! DOCTYPE html PUBLIC"-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.
w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title>主题与样式应用</title>
        <link type="text/css" href="App_Themes/Default/StyleSheet.css"/>
    </head>
    <body>
        <form id="form1" runat="server">
            <div class="topDiv">              <!--指定要使用的样式-->
             <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/top-
Image.jpg"/>
            </div>
            <div class="contentDiv">          <!--指定要使用的样式-->
            <p></p>
            <p>邮件地址<asp:TextBox ID="txtEmail" runat="server" SkinID=
"txtMainSkin"></asp:TextBox>
            <!--以上指定要使用的主题-->
            @<asp:DropDownList ID="ddlEmailType" runat="server" SkinID=
"ddlMainSkin">
                <asp:ListItem>163.com</asp:ListItem>
                <asp:ListItem>QQ.com</asp:ListItem>
                <asp:ListItem>yahoo.cn</asp:ListItem>
            </asp:DropDownList>
            </p><p>密        码<asp:Text-
Box ID="txtPassword" runat="server" SkinID="txtMainSkin"></asp:TextBox></p>
            <!--以上指定要使用的主题-->
            <p>确认密码<asp:TextBox ID="txtPassword1" runat="server"
SkinID="txtMainSkin"></asp:TextBox></p>
            <!--以上指定要使用的主题-->
            <p>手机号码<asp:TextBox ID="txtPhone" runat="server"
SkinID="txtMainSkin"></asp:TextBox></p>
            <!--指定要使用的主题-->
```

```
            <p><asp:Button ID="btnReg" runat="server" SkinID="btnMain-
Skin" Text="注册"/>
            <! ——以上指定要使用的主题——>
                   
            <asp:Button ID="btnClear" runat="server" SkinID="btnMainSkin"
Text="重置"/>
            <! ——以上指定要使用的主题——>
                </p>
            </div>
            </form>
    </body>
    </html>
```

## 3.1.3 母版页的设计及使用

### 1. 母版页概述

母版页是 ASP. NET 中新引入的概念，它很好地实现了界面设计的模块化，并且实现了代码的重用。它就像婚纱影楼中的婚纱照模板，同一个婚纱照模板可以给不同的新人用。只要把新人的照片贴在已有的婚纱照模板上，就可以形成一张漂亮的婚纱照，这大大简化了婚纱艺术照的设计复杂度。这里的母版页就像婚纱照模板，而内容页面就像两位新人的照片。

母版页是扩展名为. master 的 ASP. NET 文件，具有包括静态文本、HTML 元素和服务器控件的预定义布局。母版页由特殊的@Master 指令识别，替换了用于普通. aspx 页的@Page指令。该指令举例如下。

```
    <%@ Master Language="C#" %>
```

除@Master 指令外，母版页还包含页面的所有顶级 HTML 元素，如 html、head 和 form。例如，在母版页上可以将一个 HTML 表用于布局，将一个 img 元素用于公司徽标，将静态文本用于版权声明，并使用服务器控件创建站点的标准导航。也就是说，可以在母版页中使用任何 HTML 元素和 ASP. NET 元素。

1)可替换内容占位符

除在所有页面上显示的静态文本和控件外，母版页还包括一个或多个 ContentPlaceHolder 控件。在母版页中会定义一些占位符控件，表示这部分有内容，只是内容不确定，然后在内容页中定义可替换占位符的内容。

**【例 3-5】** ContentPlaceHolder 控件的用法。

```
    <%@ Master Language="C#" %>
    <! 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>Master page title</title>
</head>
<body>
    <form id="form1" runat="server">
        <table>
          <tr>
              <td><asp:contentplaceholder id="Main" runat="server"/></td>
              <td><asp:contentplaceholder id="Footer" runat="server"/></td>
          </tr>
        </table>
    </form>
</body>
</html>
```

2)内容页

通过创建各个内容页来定义母版页中各个占位符控件的内容,这些内容页绑定到特定母版页的 ASP.NET 文件(.aspx 文件以及可选的代码隐藏文件)。通过包含指向要使用的母版页的 MasterPageFile 属性,在内容页的@Page 指令中建立绑定。

在内容页中,通过添加 Content 控件并将这些控件映射到母版页上的 ContentPlaceHolder 控件来创建内容。例如,母版页可能包含名为 Main 和 Footer 的内容占位符。在内容页中,可以创建两个 Content 控件,一个映射到 ContentPlaceHolder 控件 Main,另一个映射到 ContentPlaceHolder 控件 Footer,如图 3-20 所示。

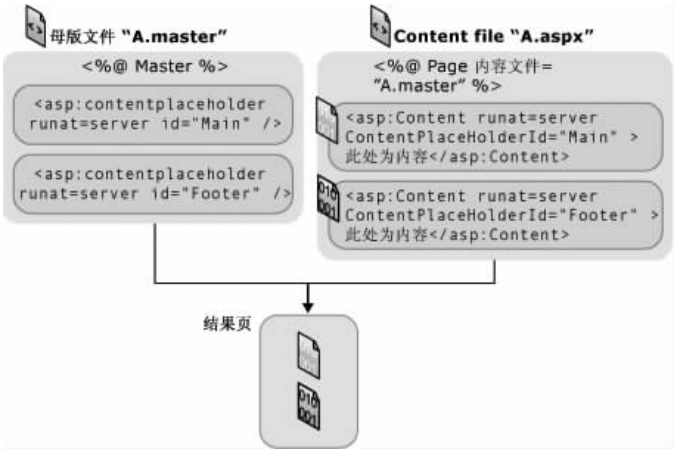


图 3-20　母版页

在运行时,母版页是按照下面的步骤处理的。

(1)用户通过输入内容页的 URL 来请求某页。

(2)获取该页后,读取@Page 指令。如果该指令引用一个母版页,则也读取该母版页。如果这是第一次请求这两个页,则两个页都要进行编译。

(3)包含更新内容的母版页合并到内容页的控件树中。

(4)各个 Content 控件的内容合并到母版页中相应的 ContentPlaceHolder 控件中。

(5)浏览器中呈现得到的合并页。

3)母版页的优点

母版页提供了开发人员已通过传统方式创建的功能,包括重复复制现有代码、文本和控件元素;使用框架集;对通用元素使用包含文件;使用 ASP. NET 用户控件等。

母版页具有以下优点。

(1)使用母版页可以集中处理页面的通用功能。

(2)使用母版页可以方便地创建一组控件和代码,并将结果应用于一组页。例如,可以在母版页上使用控件来创建一个应用于所有页的菜单。

(3)通过允许控制占位符控件的呈现方式,使用户可以在细节上控制最终页的布局。

(4)母版页提供了一个对象模型,使用该对象模型可以从各个内容页自定义母版页。

## 2. 建立母版页

在制作母版页前需要准备一些图片素材,如图标、LOGO 等。这里主要介绍功能的实现,不介绍图片的设计与处理。建立母版页的具体步骤如下。

1)创建母版页

打开“解决方案资源管理器”面板,右击,在弹出的菜单中选择“添加新项”命令,在弹出的对话框中添加母版页,如图 3-21 所示。


图 3-21　添加母版页

母版页的扩展名为. master,默认名称是 MasterPage. master。每个母版页都可以添加一个或多个 ContentPlaceHolder 标记。打开新建的母版页,默认生成以下代码。

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="Master-
Page.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">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

从上面的代码中可以看出，默认的母版页中有两个可以编辑的区域，名称分别为 head 和 ContentPlaceHolder1。下面修改默认母版页的布局，效果如图 3-22 所示。

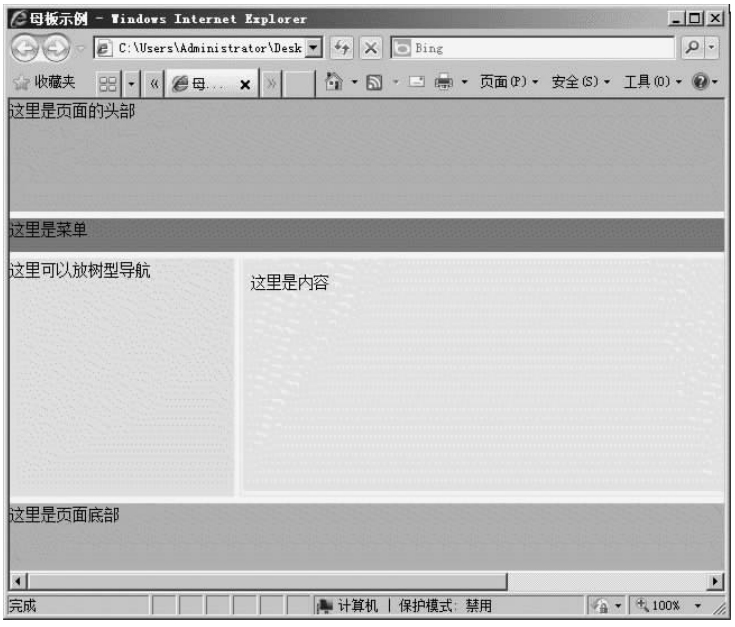


图 3-22　母版页布局

其中，在“内容”和“树型导航”中添加两个 ContentPlaceHolder 控件，因为这两个地方可能需要变动。当然，这是根据实现 Web 项目的需要而定。

现在的页面布局都是使用 DIV 和 CSS 相结合的方式，将图 3-22 转化成 Div 布局，如图 3-23 所示，括号中是 Div 的名称。

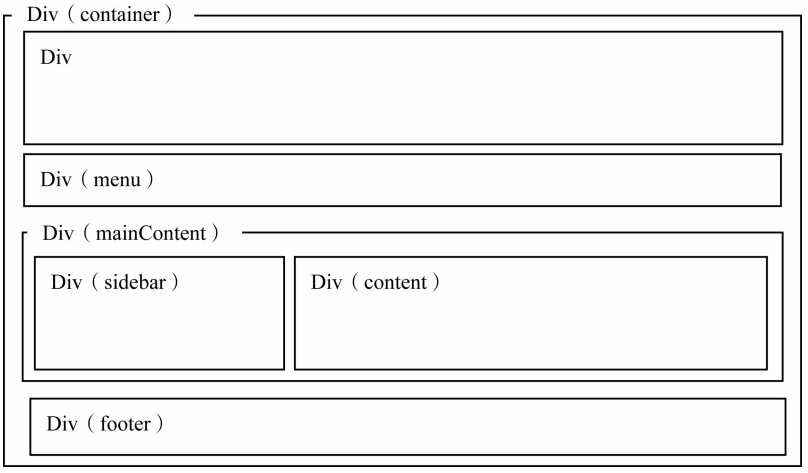


图 3-23　Div 布局页面

图 3-23 对应的 MasterPage. master 母版文件的代码如下。

```
    <%@ 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">
        <title></title>
        <link href="MasterPage.css" rel="stylesheet" type="text/css"/>
        <asp:ContentPlaceHolder id="head" runat="server">
        </asp:ContentPlaceHolder>
    </head>
    <body>
        <form id="form1" runat="server">
        <div id="container">
        <div id="header">这里是页面的头部</div>
        <br class="clearfloat"/>
        <div id="menu">这里是菜单</div>
            <br class="clearfloat"/>
                <div id="mainContent">
                <div id="sidebar">这里可以放树型导航
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
runat="server"></asp:ContentPlaceHolder>
            <br/>
        </div>
```

```
        <div id="content">这里是内容
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
            </asp:ContentPlaceHolder>
        </div>
        </div>
        <br class="clearfloat"/>
        <div id="footer">这里是页面底部</div>
        </div>
        </form>
    </body>
    </html>
```

上面的代码中包括了 3 个 ContentPlaceHolder 控件；同时在<title>下面引用了一个 MasterPage. css 样式文件，这个文件需要先建立，然后再引用。

创建 CSS 文件的方法前面已作介绍。这里打开 MasterPage. css 文件，然后使用样式生成器生成如下样式。

```
/* 总体 body 样式 */
body{font-size:14px;margin:0;line-height:150%;}
/* 主体样式 */
#container{margin:0 auto;width:900px;}
/* 头部样式 */
#header{height:100px;background:#6cf;margin-bottom:5px;}
/* 菜单样式 */
#menu{height:30px;background:#09c;margin-bottom:5px;}
/* 内容样式 */
#mainContent{overflow:auto;zoom:1;margin-bottom:5px;}
/* 左边导航样式 */
#sidebar{float:left;width:200px;background:#9ff;}
/* 右边导航样式 */
#content{float:right;width:675px;padding:10px;background:#cff;}
/* 页面底部样式 */
#footer{height:60px;background:#6cf;text-align:center;}
/* 清除样式 */
.clearfloat{clear:both;height:0;font-size:1px;line-height:0px;}
```

将以上样式与前面的母版文件相结合。

下面需要在母版页中添加具体的内容，包括 header、menu、footer 等 DIV 内容。

这里在 header 中插入一个图片，代码如下。

```
<img alt="" class="style1" src="Images/logo.gif"/>
```

在 menu 中输入几个菜单项，在 footer 中输入一些版权信息，效果如图 3-24 所示。


图 3-24　母版页效果

2)使用母版页

母版页建立后，就可以使用了。使用母版页的内容页就是普通的页面，但是在页面的代码构成上还有以下区别。

(1)文件头部没有<！DOCTYPE html>、<html>、<head>、<body>等页面标记，因为这些元素都是继承母版得到的。

(2)在文件的头部增加了引用母版页的声明，MasterPageFile＝"～/MasterPage.master"。

(3)母版中添加的 ContentPlaceHolder 控件自动变成以<asp:Content>开头的标记，在这个标记中可以添加任何页面上允许的 HTML 标记或后台代码。

打开“解决方案资源管理器”面板，在网站节点上右击，在弹出的快捷菜单中选择“添加新项”命令。在弹出的对话框中选择“Web 窗体”模板；输入新建页面名称，如“3-11. aspx”；然后选中“选择母版页”复选框，这个最重要，否则建立的是普通页面；最后单击“添加”按钮，如图 3-25 所示。


图 3-25　创建母版内容页

然后弹出母版页选择窗口，选择要使用的母版，单击“确定”按钮即可。

打开刚才建立的“3-11. aspx”文件，切换到设计视图，此时会发现，只有包含 ContentPlaceHolder 控件的部分可以修改，其他部分都是无法选择的状态，这就是要在母版中添加 ContentPlaceHolder 控件的原因，即在母版中确定可编辑区域。这里要说明的是，在设计母版页时，添加了 3 个 ContentPlaceHolder 控件，但在 3-11. aspx 页的设计视图中只看到了两个；如果切换到源视图，就可以看到 3 个 ContentPlaceHolder 控件了，因为有一个 ContentPlaceHolder 控件是在母版的<head>标记中的，在内容页的设计视图中无法看到。

从工具箱中拖动一个 Calendar 控件到左边的 ContentPlaceHolder 控件中，在右边的 ContentPlaceHolder 控件中输入一些文字，最终效果如图 3-26 所示。


图 3-26　内容页

3)访问母版中的控件

当页面是由母版页创建而来时，母版页会提供一个 Master 对象，内容页使用该对象得到或赋值母版页上控件的属性。

在 Master 对象下提供了一个 FindControl()方法，利用该方法可以在母版中查找到指定名称的控件，然后进行读取或赋值操作，具体操作步骤如下。

(1)打开 MasterPage. master 母版文件，切换到设计视图，在页面上“当前用户：”后面添加一个 Label 控件，并修改 ID 属性为 lbluser。

(2)打开内容页 3-11. aspx，在 Page_Load 事件中设计代码如下。

```
protected void Page_Load(object sender,EventArgs e)
{
    (Master.FindControl("lbluser") as Label).Text="管理员";
}
```

上面的语句是在内容页加载显示时，把母版页中名称为 lbluser 的标签的内容修改为“管理员”。

## 3.2 站 点 导 航

站点导航是一个 Web 项目中的重要组成部分。虽然使用超链接或者服务器代码可以实现网页间的切换效果，但是随着网页不断增多，页面间的关系就会变得非常复杂，而不宜维护，无法集中管理。在 ASP. NET 中使用站点导航功能可以解决这一问题。

### 3.2.1 站点地图

为了使用 ASP. NET 中的导航功能，就必须有一种标准的方式来组织站点中的各个页面。其中，需要包含各个页面的名称，还需要包含它们之间的层次关系。例如，开发一个 ASP. NET 学习网站，就有可能包含一个这样的导航路径：ASP. NET→页面布局→DIV 布局。

承载以上导航信息的文件，在 ASP. NET 中称为站点地图，其文件名必须是 web. sitemap。打开“解决方案资源管理器”面板，在站点根节点上右击，在弹出的快捷菜单上选择“添加新项”命令，在弹出的对话框中选择“站点地图”选项，文件名称默认为“web. sitemap”，因此不用修改文件名称，然后单击“添加”按钮即可。

打开文件，会发现内容是以 XML 格式组织的，此文件需要放置在应用程序的根目录下。下面是一个简单的站点地图的代码。

```
<? xml version="1.0"encoding="utf-8"? >
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="~/index.aspx" title="首页" description=
"ASP.NET学习网站">
      <siteMapNode url="~/SiteMap/news.aspx" title="技术前沿"
description=""/>
        <siteMapNode url="~/SiteMap/page.aspx" title="页面布局"
description="">
          <siteMapNode url="~/SiteMap/Div.aspx" title="MVC 框架"
description=""/>
            <siteMapNode url="~/SiteMap/css.aspx" title="工厂模式"
description=""/>
          </siteMapNode>
      <siteMapNode url="~/SiteMap/admin.aspx" title="管理员后台"
description=""/>
    </siteMapNode>
</siteMap>
```

表 3-1 描述了 web. sitemap 的常用属性及说明。

表 3-1　web. sitemap 的常用属性及说明

| 属　性 | 说　明 |
| --- | --- |
| siteMap | 根节点，一个站点地图只能有一个 siteMap 元素 |
| siteMapNode | 对应于页面的节点，一个节点描述一个页面 |
| title | 描述页面(这与页面头部的＜title＞标记没有任何联系，虽然它们的值可以相同) |
| url | 文件在解决方案中的位置 |
| description | 说明性文本 |

编写站点地图需要注意以下几点。

(1)站点地图的根节点为＜siteMap＞元素，每个文件有且仅有一个根节点。

(2)＜siteMap＞下一级有且仅有一个＜siteMapNode＞节点。

(3)＜siteMapNode＞下面可以包含多个新的＜siteMapNode＞节点。

(4)在站点地图中，同一个 url 仅可以出现一次。

### 3.2.2　SiteMapPath 控件

前面介绍了站点地图，如何使用呢？ASP. NET 提供了一组控件，SiteMapPath 控件就是其中之一。它可以为站点提供“面包屑”导航(该名称来源于格林童话《汉泽尔和格雷特尔》)，即在站点的设计中，给用户提供一条方便的“路”。SiteMapPath 导航使用起来非常方便，它使用站点地图作为控件的数据源。所以要使用该控件，首先要有站点地图。

需要注意的是，只有在站点地图中声明的文件中才能使用 SiteMapPath 控件，如果将 SiteMapPath 控件放在站点地图没有列出的页面中，它是不会显示任何信息的。

SiteMapPath 控件的常用属性如表 3-2 所示。

表 3-2　SiteMapPath 控件的常用属性

| 属　性 | 说　明 |
| --- | --- |
| CurrentNodeStyle | 定义当前节点的外观样式 |
| NodeStyle | 定义 SiteMapPath 控件中所有导航节点的外观样式 |
| PathSeparator | 设置导航路径中节点之间的分隔符 |
| PathSeparatorStyle | 定义分隔符的样式 |
| RootNodeStyle | 定义根节点样式 |

下面使用 3. 2. 1 节中创建的站点地图，在 3. 1. 3 节的母版页中使用 SiteMapPath 导航，具体步骤如下。

#### 1. 修改 MasterPage. master

在“解决方案资源管理器”面板中找到 MasterPage. master 文件并打开，切换到设计视图；打开“工具箱”，在导航中找到 SiteMapPath 控件，并将其拖拽到页面中。这里具体添加到母版页“当前位置：”的后面，如图 3-27 所示。


图 3-27　在母版页中添加 SiteMapPath 控件

默认情况下，导航的分隔符是“>”，如果需要使用其他分隔符，修改 PathSeparator 属性即可。如果希望将导航分隔符设置为图片，则需要使用分隔符模板。方法是，选中 SiteMapPath 控件，单击⊡按钮，在弹出的菜单中选择“编辑模板”选项，然后按照图 3-28 所示编辑分隔符模板。

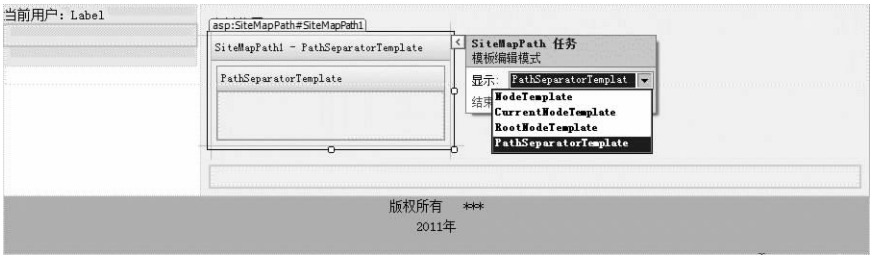


图 3-28　编辑模板

编辑完后，选择“结束模板编辑”命令即可。需要说明的是，如果设置了分隔符属性，又添加了分隔符模板，则显示时以模板为准。

### 2. 添加内容页

打开“解决方案资源管理器”面板，在 Web 项目根节点上右击，在弹出的快捷菜单中选择“新建文件夹”命令，并将其命名为 SiteMap。为什么命名为 SiteMap？因为 web. sitemap文件中的 url 所指定的文件除了 index. aspx 文件外，其他文件都是保存在 SiteMap 中的。然后，在该文件夹中建立一个使用母版的内容页 news. aspx，打开浏览效果如图 3-29 所示。


图 3-29　SiteMapPath 控件使用效果

news. aspx 文件在 web. sitemap 文件中的 title 就定义为“技术前沿”。SiteMapPath 控件完美地显示了“面包屑导航”效果。

## 3.2.3　TreeView 控件

提到树型导航，大家都不陌生，Windows 系统使用“Windows 资源管理器”处理保存在

文件系统中的文件，可以展开或折叠带有子层次结构的节点，就是标准的树型结构。

实现树型导航的方法有很多，如 JavaScript、ASP 等技术。但使用这些传统的方式编写树型导航，需要复杂而庞大的编码。新版本 ASP. NET 中提供的 TreeView 导航控件，可以像 SiteMapPath 控件那样，简单设置即可显示强大的导航。

根据数据源类型的不同，可以将 TreeView 的使用方式分为两种：使用站点地图作为数据源和使用 XML 文件作为数据源。

### 1. 使用站点地图作为 TreeView 控件的数据源

使用 3. 2. 1 中建立的站点地图作为数据源。在“解决方案资源管理器”面板中打开前面建立的 news. aspx 文件，并在页面中添加 TreeView 控件，然后设置数据源。如图 3-30 所示，在“选择数据源”下拉列表框中选择“新建数据源”选项，弹出“数据源配置向导”对话框，选择从站点地图获取数据，然后单击“确定”按钮。随即会发现在 TreeView 控件下面自动添加了一个 SiteMapDataSource 控件。此控件是隐形控件，运行时不会显示，只是为 TreeView 控件提供数据。


图 3-30　设置 TreeView 控件数据源

树型导航的最终效果如图 3-31 所示。


图 3-31　树型导航最终效果

### 2. 使用 XML 文件作为 TreeView 控件的数据源

前面使用了站点地图文件作为 TreeView 控件的数据源，但这种方法不够灵活。例如，一般使用站点地图的树型导航会把所有页面列出来，如果想要屏蔽一些页面，虽然可以在站点地图中使用用户权限来实现，但还是不够灵活。对于图 3-31 所示的“管理员后台”节点，不同的管理员可能会列出不同的功能导航。在这种情况下，可以使用 XML 文件作为 TreeView 控件的数据源。

1）添加 XML 文件

打开“解决方案资源管理器”面板，在 SiteMap 文件夹中添加一个 XML 文件。方法与添加普通页面一样，只不过在选择模板时选择“XML 文件”，命名为“adminMenu. xml”。该 XML 文件的内容可以参考站点地图文件中的内容。与站点地图相比，这里需要的 XML 文

件没有那么多限制条件，甚至从站点地图中复制部分代码过来就可以，只要符合 XML 标准即可。设置的 XML 文件内容如下。

```
<? xml version="1.0"encoding="utf-8"? >
<siteMapNode Id="root" url="" title="管理员控制面板" description="">
  <siteMapNode url="" title="用户管理" description="">
    <siteMapNode url="~\SiteMap\ListAllUsers.aspx" title="管理用户"
description=""/>
    <siteMapNode url="~\SiteMap\UserStatue.aspx" title="状态管理"
description=""/>
    <siteMapNode url="~\SiteMap\ListAllUsers.aspx" title="用户列表"
description=""/>
  </siteMapNode>
  <siteMapNode url="" title="新闻管理" description="">
    <siteMapNode url="~\SiteMap\AddNews.aspx" title="发布新闻"
description=""/>
  </siteMapNode>
  <siteMapNode url="~\SiteMap\LoginOut.aspx" title="退出" description="管
理员退出">
  </siteMapNode>
</siteMapNode>
```

该文件中还没有所指示的页面，不过没有关系，这并不影响该控件的正常显示。

2)添加内容页

打开“解决方案资源管理器”面板，在 SiteMap 文件夹中添加一个使用 MasterPage. master 母版的内容页，并命名为 admin. aspx。打开此文件，切换到设计视图，添加一个 TreeView 控件到左边的 ContentPlaceHolder 中。选中 TreeView 控件，单击▷按钮，在“选择数据源”下拉列表框中选择“新建数据源”选项，如图 3-30 所示。在弹出的“数据源配置向导”对话框中选择 XML 文件后，单击“确定”按钮，弹出“配置数据源”对话框，如图 3-32 所示。可单击“数据文件”文本框右边的“浏览”按钮，选择前面创建的 adminMenu. xml 文件，单击“确定”按钮。

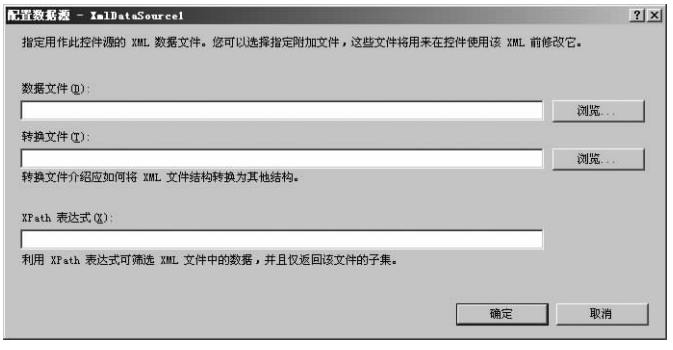


图 3-32 “配置数据源”对话框

此时会发现 TreeView 控件所有的节点都自动填充成 siteMapNode。但这并不是真正想要的内容，需要进一步设置。选中 TreeView 控件，单击⊡按钮，在“TreeView 任务”下拉菜单中选择“编辑 TreeNode 数据绑定”命令，如图 3-33 所示。


图 3-33　“TreeView 任务”下拉菜单

在弹出的“TreeView DataBindings 编辑器”对话框中，选择“可用数据绑定”列表框中的 siteMapNode 选项，并单击“添加”按钮，将其添加到下面的“所选数据绑定”列表框中，同时设置 NavigateUrlField 和 TextField 属性为 XML 文件中的相应节点，如图 3-34 所示。


图 3-34　编辑数据绑定

如果觉得 TreeView 不够美观，ASP. NET 还提供了一组定义好的样式供选择。方法是，在 TreeView 控件上右击，在弹出的快捷菜单中选择“自动套用格式”命令，在弹出的“自动套用格式”对话框中选择喜欢的样式，如“XP 资源管理器”，如图 3-35 所示。

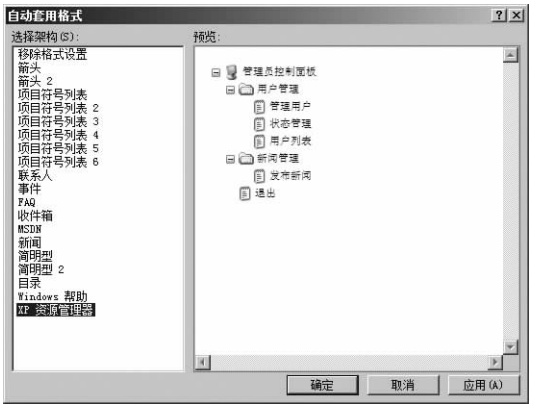


图 3-35　“自动套用格式”对话框

TreeView 控件的使用效果如图 3-36 所示。


图 3-36　TreeView 控件的使用效果

### 3. 常用属性、方法和事件

TreeView 控件的功能非常强大，不仅可以使用站点地图和 XML 文件，还可以使用后台代码动态地生成树中的节点，这样它就可以很好地与数据库结合，实现更强大的功能。

1）常用属性

TreeView 控件的属性非常多，表 3-3 只列出了部分常用的属性。如果在开发过程中对某个属性不理解，可以在其“属性”面板中选择该属性，下面的提示栏中会显示中文提示。ASP. NET 这一点设计得非常人性化。

表 3-3　TreeView 常用属性

| 属　性 | 说　明 | 属　性 | 说　明 |
|---|---|---|---|
| Height | 控件的高度 | ShowLines | 是否显示层级连接线 |
| Width | 控件的宽度 | ShowExpandCollapse | 是否在节点旁边显示展开/折叠图标 |
| BackColor | 背景颜色 | ExpandDepth | 默认情况下展开树的多少级别 |
| BorderColor | 边框颜色 | Nodes | 树的初始化节点集合 |
| BorderStyle | 边框样式 | NodeIndent | 设置子节点的缩进量（以像素为单位） |
| BorderWidth | 边框宽度 | ExpandedImageUrl | 展开时显示的节点图标 |
| CssClass | 应用于该控件的 CSS 类名 | NodeWrap | 节点文本是否自动换行 |

2）常用方法

TreeView 控件的每个节点都是一个 TreeNode 对象，并具有 Text 属性和 Value 属性。

其中，Text 属性用于设置在节点上显示的文字，Value 属性用于设置节点对应的值。TreeView 控件的绝大部分操作都是针对节点的，所以表 3-4 列出了常用的针对 Nodes 的方法。

表 3-4　TreeView 控件的常用方法

| 属　性 | 说　明 |
| --- | --- |
| Clear() | 无参数，清空 TreeView 下所有的节点信息 |
| Add() | 需要一个 TreeNode 参数，添加一个新的节点 |
| Remove() | 需要一个 TreeNode 参数，删除一个指定的节点 |
| RemoveAt() | 需要一个 int 参数，删除一个指定索引的节点 |
| ExpandAll() | 展开树中的每个节点 |

3）常用事件

TreeView 控件有以下两个常用事件。

（1）SelectedNodeChanged：当选择了不同的节点时激发。

（2）TreeNodeExpanded：当节点展开时激发。该事件中带有参数“TreeNodeEventArgs e”，该参数用于指明哪个节点被展开了。

**【例 3-6】** 使用代码动态创建图 3-37 所示的树型导航。

（1）打开“解决方案资源管理器”面板，添加新的页面文件，命名为 3-6.aspx。

（2）打开 3-6.aspx 文件，切换到设计视图，从工具箱中拖拽一个 TreeView 控件到页面中。

（3）打开 3-6.aspx.cs 文件，在 Page_Load 中设计代码如下。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class _3_6:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        //添加根节点
        TreeView1.Nodes.Add(new TreeNode("管理员控制面板","admin"));
        //添加子节点
        TreeView1.Nodes[0].ChildNodes.Add(new TreeNode("用户管理","user-
Manage"));
        //添加叶子节点
        TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(new TreeNode("管
理用户","users"));
```

```
            TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(new TreeNode("状
态管理","status"));
            TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(new TreeNode("用
户列表","usersList"));
            TreeView1.Nodes[0].ChildNodes.Add(new TreeNode("新闻管理","news-
Manage"));
            TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(new TreeNode("发
布新闻","addNews"));
            TreeView1.Nodes[0].ChildNodes.Add(new TreeNode("退出","exit"));
        }
    }
```

运行效果如图 3-37 所示。


图 3-37　动态生成树

实际开发 Web 项目时，树的节点信息可能要从数据库中读取，这里没有介绍与数据库相关的操作方法，读者可自行查阅相关资料。

### 3.2.4　Menu 控件

菜单是大家非常熟悉的一种导航方式，也是大部分 Web 项目都使用的一种导航方式。ASP. NET 中也提供了一个菜单导航控件，那就是 Menu 控件。

Menu 控件有两种显示模式：静态模式和动态模式。静态显示意味着 Menu 控件始终是完全展开的，整个结构都是可视的，用户可以单击任何部位。而在动态显示的菜单中，只有指定的部分是静态的，且只有将鼠标指针放置在父节点上时才会显示其子菜单项。

可以在 Menu 控件中直接配置其内容，也可以通过将该控件绑定到数据源的方式来指定其内容，这一点与 TreeView 控件是一致的。无须编写任何代码，便可控制 ASP. NET Menu 控件的外观、方向和内容。除公开的可视属性外，Menu 控件还支持 ASP. NET 控件的外观和主题。

在使用方式上，Menu 控件和 TreeView 控件类似，也可以使用站点地图文件和 XML 文件作为数据源。唯一不同的是，Menu 控件会智能提示是以静态，还是以动态来显示视图。如图 3-38 所示，单击“视图”下拉列表框的下三角按钮，可设置为“动态”或“静态”。如

果动手做试验，可能会发现这两种设置的效果是一样的。事实上，Menu 控件的动态显示或静态显示是通过属性来设置的。

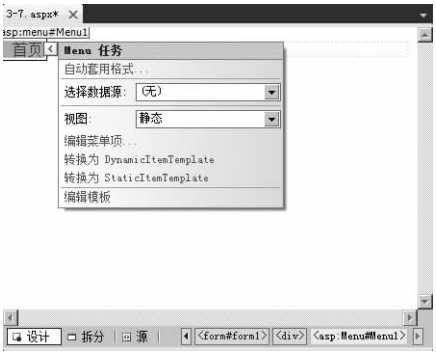


图 3-38　Menu 的智能提示

### 1. 静态与动态设置

1）静态显示

使用 Menu 控件的 StaticDisplayLevels 属性可以控制静态显示行为。StaticDisplayLevels 属性指示从根菜单算起，静态显示菜单的层数。例如，将 StaticDisplayLevels 设置为 3，则菜单将以静态显示的方式展开其前 3 层。静态显示的最小层数为 1，如果将该值设置为 0 或负数，该控件将会引发异常。

2）动态显示

MaximumDynamicDisplayLevels 属性指定在静态显示层后应动态显示的菜单的节点层数。例如，某个菜单有 3 个静态层和 2 个动态层，则菜单的前 3 层静态显示，后 2 层动态显示。

如果将 MaximumDynamicDisplayLevels 设置为 0，则不会动态显示任何菜单节点；如果将其设置为负数，则会引发异常。

### 2. 常用属性

Menu 控件的属性较多，表 3-5 只列出了一些常用属性，其中部分属性是集合属性。

表 3-5　Menu 常用属性

| 属　性 | 说　明 | 属　性 | 说　明 |
|---|---|---|---|
| Height | 控件的高度 | DisappearAfter | 鼠标指针不再置于菜单上后显示动态菜单的持续时间 |
| Width | 控件的宽度 | Items | 菜单项的集合 |
| BackColor | 背景颜色 | ItemWrap | 菜单项的文本是否应换行 |
| BorderColor | 边框颜色 | Orientation | 菜单的静态部分是以水平方式，还是以垂直方式呈现 |
| BorderStyle | 边框样式 | Forecolor | 控件中文本的颜色 |
| BorderWidth | 边框宽度 | CssClass | 应用于该控件的 CSS 类名 |

### 3. 菜单样式

设置 Menu 控件的样式取决于对菜单项的外观控制的程度和控制的位置。例如，如果希望每个菜单项层都有一个一致的外观，可使用 LevelMenuItemStyles 属性来设置每个菜单层的样式。如果希望所有静态菜单项的外观都相同，所有动态菜单项的外观也都相同，则可以使用 StaticMenuStyle 属性来控制所有静态菜单项的外观，使用 DynamicMenuStyle 属性来控制所有动态菜单项的外观。StaticMenuStyle 属性和 DynamicMenuStyle 属性分别影响整组静态菜单项和动态菜单项。例如，使用 DynamicMenuStyle 属性指定一个边框，则整个动态区域将会有一个边框。

StaticMenuItemStyle 属性和 DynamicMenuItemStyle 属性只影响单个菜单项。例如，使用 DynamicMenuItemStyle 属性指定一个边框，则每个动态菜单项都有它自己的边框。

当鼠标指针置于菜单项上时，StaticSelectedStyle 属性和 DynamicSelectedStyle 属性仅影响所选的菜单项；而 StaticHoverStyle 属性和 DynamicHoverStyle 属性影响鼠标指针置于菜单项上菜单项的样式。

**【例 3-7】** Menu 实例。

(1)打开“解决方案资源管理器”面板，添加一个新的页面文件，并命名为 3-7. aspx。

(2)打开 3-7. aspx 文件，切换到设计视图，拖拽一个 Menu 控件到页面中。

(3)选择 Menu 控件，单击▷按钮，在弹出的“Menu 任务”下拉菜单中选择“编辑菜单项”命令。在弹出的对话框中添加菜单项，单击 按钮可以添加根节点，单击 按钮可以添加子节点。每个菜单项都可以设置下面的属性。

①Text：菜单项的显示文本。

②Value：菜单项的值。

③NavigateUrl：菜单项被选中时定位到的 URL。

④ImageUrl：用于菜单项的图标设定。

以上的属性可以有选择地填写，最终添加的菜单项如图 3-39 所示。


图 3-39 添加菜单项

(4)设置 Menu 控件属性，包括控件的背景色、字体大小、静态及动态显示背景色与前景色、边距等。最终生成的 Menu 控件的属性代码如下。

```
    <asp:Menu ID="NavigationMenu1" runat="server" orientation="Horizontal"
StaticEnableDefaultPopOutImage="False" BackColor="#009CCE" Font-Bold=
"True" Font-Size="14px" ForeColor="Black"/>
        <DynamicHoverStyle BackColor="#53DCDC"/>
        <DynamicMenuItemStyle HorizontalPadding="4px" VerticalPadding="4px"
Font-Size="12px"/>
        <DynamicMenuStyle HorizontalPadding="2px" VerticalPadding="2px"
Width="80px" BackColor="#9cf7f7"/>
          <Items>
              <asp:MenuItem Text="主页" Value="主页"></asp:MenuItem>
              <asp:MenuItem Text="技术前沿" Value="技术前沿">
              <asp:MenuItem Text="Web 开发" Value="Web 开发"></asp:MenuItem>
              <asp:MenuItem Text="移动 Web" Value="移动 Web">
              <asp:MenuItem Text="HTML 5" Value="HTML 5"></asp:MenuItem>
              <asp:MenuItem Text="JQuery" Value="JQuery"></asp:MenuItem>
              </asp:MenuItem>
              </asp:MenuItem>
              <asp:MenuItem Text="页面布局" Value="页面布局">
              <asp:MenuItem Text="DIV 布局" Value="DIV 布局"></asp:MenuItem>
              <asp:MenuItem Text="CSS" Value="CSS"></asp:MenuItem>
              </asp:MenuItem>
              <asp:MenuItem Text="MVC 框架" Value="MVC 框架">
              <asp:MenuItem Text="MVC 信息" Value="MVC 信息"></asp:MenuItem>
              <asp:MenuItem Text="MVC 学习资料" Value="MVC 学习资料">
              </asp:MenuItem>
              </asp:MenuItem>
              <asp:MenuItem Text="工厂模式" Value="工厂模式">
              <asp:MenuItem Text="多层框架" Value="多层框架">
              </asp:MenuItem>
              <asp:MenuItem Text="三层框架" Value="三层框架">
              </asp:MenuItem>
              </asp:MenuItem>
          </Items>
        <StaticHoverStyle  BackColor="#53DCDC"/>
      <StaticMenuItemStyle HorizontalPadding="10px" ItemSpacing="2px"
VerticalPadding="3px"/>
    </asp:Menu>
```

最终设计效果如图 3-40 所示。


图 3-40　Menu 控件使用样式

## 3.3 项目实战:有才网络公司网站

本项目以一个简单的网络公司企业网站为实例,来介绍母版、站点地图、导航菜单等技术在实际项目中的运用。

### 3.3.1　实施计划

有才网络公司的网站是一个典型的企业展示网站,它以介绍企业业务为主要内容来组织网站页面,其主要功能模块如下。

(1)网站建设:展示公司的产品,以及好网站的素材。

(2)网站推广:介绍公司的网站推广产品。

(3)域名注册:介绍公司域名注册流程及收费情况。

(4)虚拟主机:介绍虚拟主机种类及收费情况。

(5)企业邮局:介绍企业邮箱业务相关说明及资费情况。

(6)客户服务:提供与客户联系的方式和相关服务说明。

(7)代理合作:介绍代理合作说明及利益分成。

(8)论坛交流:提供一个与客户及其他人员交流的平台。

总体来说,这是一个注重企业展示的网站。在该网站的开发中,导航是必不可少的元素,可以使用上面介绍的不同导航技术来开发企业的网站样式框架。这个企业网站的模块很多,限于篇幅,这里只重点介绍网站母版的设计,以及其中“网站建设”模块的制作。

### 3.3.2　项目实施

(1)打开 Visual Studio 2010,选择“文件”→“新建”→“网站”命令,新建一个网站,并将其命名为 WebCompany。

(2)打开“解决方案资源管理器”面板,新建一个 Images 文件夹,保存网站需要用到的图片素材,并导入需要的素材图片。

(3)新建一个样式文件,命名为 main. css,具体内容见网站中的样式文件。

(4)编写站点地图 web. sitemap,代码如下。

```
<? xml version="1.0"encoding="utf-8"?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="default.aspx" title="有才首页" description="">
      <siteMapNode url="web.aspx" title="网站建设" description="">
        <siteMapNode url="static.aspx" title="静态网站" description=""/>
        <siteMapNode url="asp.aspx" title="Asp 网站" description=""/>
        <siteMapNode url="aspnet.aspx" title="ASP.NET 网站" description=""/>
      </siteMapNode>
      <siteMapNode url="popularize.aspx" title="网站推广" description="">
        <siteMapNode url="baidu.aspx" title="Baidu 推广" description=""/>
        <siteMapNode url="google.aspx" title="Google 推广" description=""/>
      </siteMapNode>
        <siteMapNode url="reg.aspx" title="域名注册" description=""/>
        <siteMapNode url="host.aspx" title="虚拟主机" description=""/>
        <siteMapNode url="email.aspx" title="企业邮局" description=""/>
        <siteMapNode url="client.aspx" title="客户服务" description=""/>
        <siteMapNode url="acting.aspx" title="代理合作" description=""/>
        <siteMapNode url="forum.aspx" title="论坛交流" description=""/>
    </siteMapNode>
</siteMap>
```

(5)编写网站建设功能树型目录所需要的 XML 文件,并保存为 menuTree.xml,内容如下。

```
<? xml version="1.0"encoding="utf-8"?>
<siteMapNode Id="root" url="" title="网站展示" description="">
  <siteMapNode url="" title="案例展示" description="">
    <siteMapNode url="classic.aspx" title="典型案例" description=""/>
    <siteMapNode url="news.aspx" title="最新案例" description=""/>
  </siteMapNode>
  <siteMapNode url="" title="网站分类" description="">
    <siteMapNode url="fashion.aspx" title="时尚品牌" description=""/>
    <siteMapNode url="house.aspx" title="房地产网" description=""/>
    <siteMapNode url="company.aspx" title="优秀企业" description=""/>
    <siteMapNode url="woman.aspx" title="女性时尚" description=""/>
    <siteMapNode url="flash.aspx" title="动感 FLASH" description=""/>
    <siteMapNode url="cartoon.aspx" title="卡通漫画" description=""/>
    <siteMapNode url="cool.aspx" title="立体酷站" description=""/>
    <siteMapNode url="composite.aspx" title="综合门户" description=""/>
    <siteMapNode url="Korea.aspx" title="韩国风格" description=""/>
```

```
        <siteMapNode url="Japan.aspx" title="日本风格" description=""/>
        <siteMapNode url="other.aspx" title="其他类别" description=""/>
    </siteMapNode>
</siteMapNode>
```

(6)创建母版页,并添加布局。如果觉得使用 Visual Studio 2010 设计界面不舒服,可以使用 Dreamweaver 设计页面布局,然后将 HTML 代码复制到母版页,将共用可替换部分加入 ContentPlaceHolder。

在母版页中加入 Menu 控件和 SiteMapPath 控件。引入前面设计的 main.css 文件,代码如下。

```
<LINK href="main.css" type=text/css  rel=stylesheet>
```

母版页设计布局在 IDE 中的效果如图 3-41 所示。


图 3-41　母版页布局效果

(7)创建“网站建设”页面,并保存为 web.aspx。在页面中添加 TreeView 控件,同时设计显示内容。最终该页面在 IDE 中的效果如图 3-42 所示。


图 3-42　“网站建设”页面设计效果

### 3.3.3　项目测试

在 Visual Studio 2010 中调试非常简单，按 F5 键运行后就可以看到设计的效果。如果出现错误，在编译时会在下面的错误列表中提示，双击列表中的错误可以直接定位到错误所在的位置，然后修改即可。

关于导航有几点需要说明。

(1)母版页只提供一个页面框架，内容页才是具体显示的页面。

(2)母版页可以放置导航控件。如果使用 SiteMapPath 控件，它是根据内容页的路径显示当前的路径，而不是依据母版页。

(3)只有在站点地图中写明某页面的路径，该页面才能显示 SiteMapPath 导航。

(4)SiteMapPath 控件只能绑定站点地图，而 TreeView 控件和 Menu 控件还可以绑定 XML 文件。

本实例只是简单展示企业网站母版页的页面布局设计，包含的具体功能比较少，离一个完整的企业网站还有距离。在此基础上，读者至少可以在以下几个方面做进一步的改进。

(1)增加更详细的菜单导航内容，更具体的提示信息。

(2)进一步细化母版页中的布局。

(3)建立内容页，命名为站点地图或 MenuTree. xml 文件中的文件名称，并设计其页面内容。

(4)增加后台管理功能(这需要数据库操作的相关知识，读者可以在后面章节中学习相关内容后把功能实现)。

(5)将母版页中的布局改成 DIV 布局。

以上的改进都是进一步完善企业网站必须要实现的，有兴趣的读者可以自行尝试。

## 习　　题

### 一、填空题

1. 样式表定义 # title {color: red} 表示网页中 id 为_____的元素中的内容是_____的。

2. List-style-type: square 表示列表项符号是_____。

3. 在样式表中可以利用_____属性实现两个 DIV 重叠效果。

4. _____属性是 CSS 用来更改背景颜色的。

5. 母版中可编辑区域至少有_____个。

6. 母版文件的扩展名为_____。

7. Menu 控件有两种显示模式：_____和_____。

## 二、选择题

1. CSS 是(　　)的缩写。

A. colorful style sheet　　B. computer style sheet

C. cascading style sheet　　D. creative style sheets

2. 引用外部样式表示的元素应该放在(　　)。

A. HTML 文档的开始的位置　　B. HTML 文档的结束的位置

C. 在 head 元素中　　D. 在 body 元素中

3. 元素中定义样式表的属性名是(　　)。

A. style　　B. class　　C. styles　　D. font

4. 下面的选项中,定义样式表的正确格式是(　　)。

A. {body:color=black(body)}　　B. body:color=black

C. body{color:black}　　D. {body;color:black}

5. 样式表定义中的注释语句的格式是(　　)。

A. /*注释语句*/　　B. //注释语句//

C. //注释语句　　D. '注释语句

6. (　　)不是 ASP. NET 中使用的导航技术。

A. TreeView　　B. Menu　　C. CSS　　D. SiteMapPath

## 三、综合题

1. CSS 样式有哪几种定义方式?
2. CSS 样式有哪几种选择器?
3. 母版页有何作用?
4. 如何创建一个母版页,并利用其使多个页面风格统一?
5. 母版页与内容页是如何融合在一起的?
6. 如何在内容页中访问母版页中的控件?
7. 比较多种导航方式的优缺点及其使用场合。
8. TreeView 控件有哪几种生成导航树的方式?