

创建简单的 Web 网站

随着 Web 技术的发展,人们不再满足于建立各种静态发布信息的网站,更多的时候需要能与用户进行交互,提供后台数据库管理和控制等服务的动态网站。在这种情况下,产生了许多动态脚本技术。微软公司推出的 ASP.NET 是其中比较优秀的一种,旨在减轻开发人员的压力,使程序员有更多的时间思考核心的业务逻辑。本项目重点介绍 ASP.NET 中基本的开发方式,帮助读者了解 ASP.NET 的运行原理和过程。

【工作任务】

- ④ 搭建 ASP.NET 开发环境。
- ④ 创建第一个 ASP.NET 应用程序。
- ④ 显示 Northwind 数据库的雇员表,并实现按雇员编号查询雇员信息。

【技能目标】

- ④ 能按照 ASP.NET 程序的开发步骤开发 Web 程序。
- ④ 熟练应用 ASP.NET“控件+事件”的开发方式开发 Web 程序。
- ④ 能熟练使用 Label、TextBox、Button 等简单控件。

【常用术语】

IIS、虚拟目录、代码后置、代码内嵌、ASP.NET 运行机制



1.1 任务 1: 创建第一个 ASP.NET 应用程序

1.1.1 任务描述

编写简单的问候程序,要求输入用户名后单击“确定”按钮即显示欢迎信息,如图 1-1 所示。



图 1-1 问候程序效果图

1.1.2 安装和配置 IIS 服务器

ASP.NET 需要使用 Web 服务器作为发布平台,一般使用 IIS 作为 Web 服务器。IIS (Internet information services, Internet 信息服务)是微软公司开发的 Web 服务器,它基于 Windows 操作系统,操作方便,功能强大,为 ASP.NET 的稳定运行提供了有力保障。

1. IIS 的安装

在 Windows 操作系统的安装光盘中一般都有 IIS 程序,但默认是不安装的,而且 IIS 根据操作系统版本的不同也有些区别,所以要顺利安装 IIS,最好找到与当前操作系统一致的系统安装光盘。在 Windows 2003 中,IIS 的安装步骤如下。

(1)打开“控制面板”,双击“添加/删除程序”图标,在弹出的对话框中单击“添加/删除 Windows 组件”按钮。

(2)在弹出的“Windows 组件向导”对话框中的“组件”列表框中选“应用程序服务器”复选框,如图 1-2 所示。

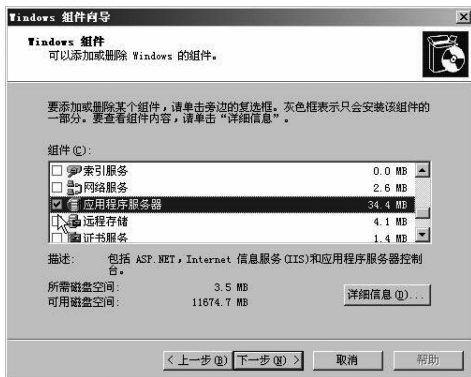


图 1-2 “Windows 组件向导”对话框

(3)单击“详细信息”按钮,弹出“应用程序服务器”对话框,在列表框中选中“ASP.NET”和“Internet 信息服务(IIS)”复选框,如图 1-3 所示。

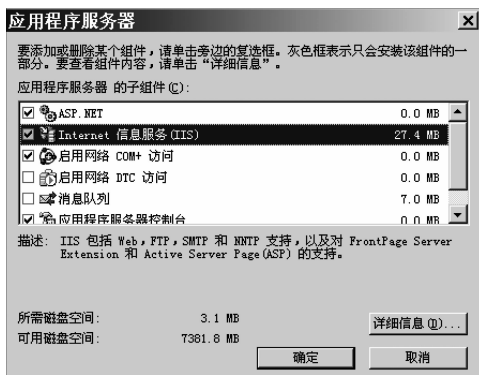


图 1-3 “应用程序服务器”对话框

(4)选择“Internet 信息服务(IIS)”选项,单击“详细信息”按钮,在弹出的“Internet 信息服务(IIS)”对话框中可选择安装相关的组件和服务,如图 1-4 所示。其中,“万维网服务”选项默认为选中状态,在它的详细信息配置中可以选择安装 ASP、服务器安装文件等子组件,如图 1-5 所示。

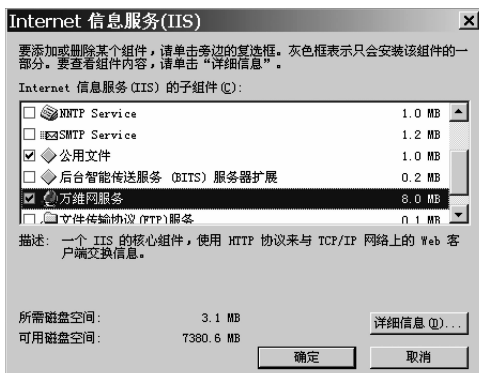


图 1-4 “Internet 信息服务(IIS)”对话框

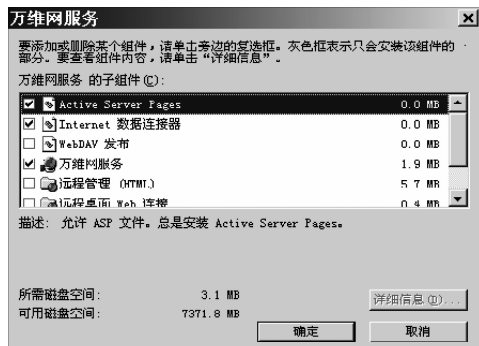


图 1-5 “万维网服务”对话框

(5)单击“确定”按钮,进行下一步操作,Windows 会自动寻找安装光盘中所需组件进行安装。安装完成时,Windows 组件向导会提示安装成功,如图 1-6 所示。



图 1-6 完成安装

2. 设置虚拟目录

IIS 安装完成后,要为 ASP.NET 应用程序设置站点。IIS 中完全是可视化操作,只需要在引导下设置即可。

一般情况下,当为网站创建了主目录后,还需要创建虚拟目录。所谓“虚拟目录”,是指该目录物理上并不在 Web 服务器的主目录中,而是在本地计算机中的磁盘的其他位置或网络中的其他计算机中,为了让这个目录中的内容可以被用户访问到,就使用虚拟目录的方式建立一种逻辑上的链接,使这个目录就像位于主目录中一样。例如,已经设定主目录为 c:\inetpub\wwwroot,该目录下存放的是网站的主文件,而程序文件存放在 d:\aaa 目录中,这时,若用户在浏览器中输入 http://localhost,则只能浏览 c:\inetpub\wwwroot 中的网页,而无法运行 d:\aaa 目录中的程序。如果要浏览 d:\aaa 目录中的内容,就可以设定 d:\aaa 目录为 http://localhost 的一个虚拟目录,这样当用户在浏览器中输入 http://localhost/aaa 后就可以浏览 d:\aaa 目录中的内容。

创建一个虚拟目录的步骤如下。

(1)打开 Internet 信息服务(IIS)管理器,如图 1-7 所示。



图 1-7 IIS 管理器

(2) 右击“网站”下面的“默认网站”，在弹出的快捷菜单中选择“新建”→“虚拟目录”命令，如图 1-8 所示。

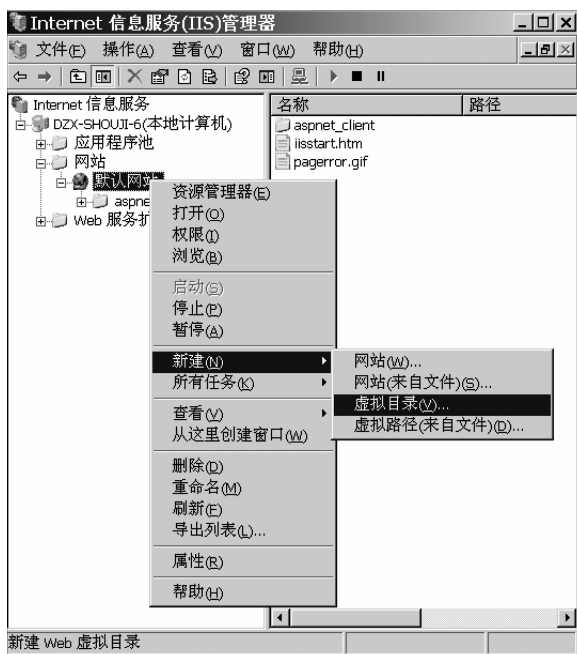


图 1-8 新建虚拟目录

(3) 在向导中首先需要填写该虚拟目录的名称。

(4) 根据需要选择路径，即选择应用程序放置的路径。

(5) 设置该虚拟路径的访问权限。此处至少要设置为可以读取，根据需要还可以设置为允许运行脚本。

3. 开发服务器

一般会把开发好的 ASP.NET 程序部署到 IIS 上，因此需要考虑虚拟目录等细节问题。实际上在程序开发阶段并没有用到很多 IIS 的特征，因此微软公司提供了一种解决方案：在 Visual Studio 中内置一个轻量级的 Web 服务器，开发人员只需要像 WinForms 开发那样编码，然后运行就可以了。该应用程序运行时在状态栏中有一个图标，如图 1-9 所示。



图 1-9 状态栏中显示的开发服务器图标

双击该图标，可以查看当前运行的站点信息，如图 1-10 所示。

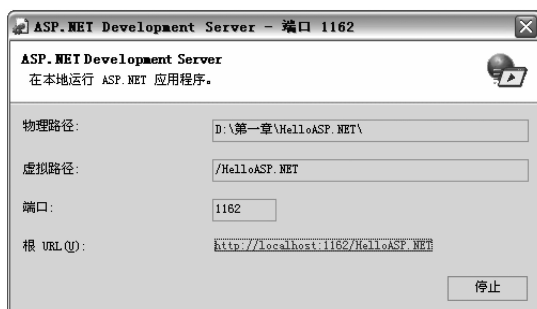


图 1-10 查看开发服务器当前运行的站点信息

1.1.3 ASP.NET 的运行环境

ASP.NET 的运行需要一系列的运行环境支持。因为 ASP.NET 是运行在服务器上的程序,所以下面介绍的这些开发环境均为服务器端环境。

1. 操作系统的支持

ASP.NET 被推荐运行在 Windows 操作系统上,包括以下操作系统。

- Windows 2000(包含 Professional、Server 和 Advanced Server 三个版本)
- Windows XP Professional
- Windows Server 2003

提示: 操作系统需要安装最新的 Server Pack,如 Windows 2000 系列需要安装 SP4。同时,操作系统还需要安装 Internet Explorer 5.5 或以上版本的浏览器。

2. .NET 环境

要正常运行 ASP.NET 还需要安装 .NET 运行环境,即 .Net Framework。ASP.NET 的运行环境是 .NET Framework,开发环境是 Visual Studio 2005。一般的开发环境,安装程序中已经集合了运行环境,安装 Visual Studio 2005 时安装程序会提示自动安装 .NET Framework。但开发环境不一定要使用 Visual Studio 2005,使用记事本等任意一个文本编辑器即可,此时只需要安装 .NET Framework。另外,Windows 2003 不需要此步骤,系统已自带了 ASP.NET 的运行环境。

3. 其他软件要求

ASP.NET 的运行需要 Web 服务器的支持。在 Windows 系统下使用的 Web 服务器是 IIS,前面已经介绍。

ADO.NET 需要微软数据访问组件(MDAC)2.7 以上版本,该软件可以从微软公司官方网站下载。

1.1.4 ASP.NET 的工作原理

1. ASP.NET 的运行机制

ASP.NET 运行时首先是来自客户端的请求,Web 服务器 IIS 根据相应条件(后缀名)将请求提交给解析器,经过编译、运行,最终将请求结果返回给客户端。ASP.NET 的运行

机制如图 1-11 所示。我们知道,页面由.aspx 文件和.cs 文件构成,用户访问.aspx 文件时,ASP.NET 引擎会编译.aspx 文件和.cs 文件,生成页面类,用户请求经过处理后返回处理结果,这是第一次请求的处理过程。当第二次请求该页面时,情况就不一样了,因为该页面类已经存于内存了,所以就省去了编译环节,只剩下执行和输出。

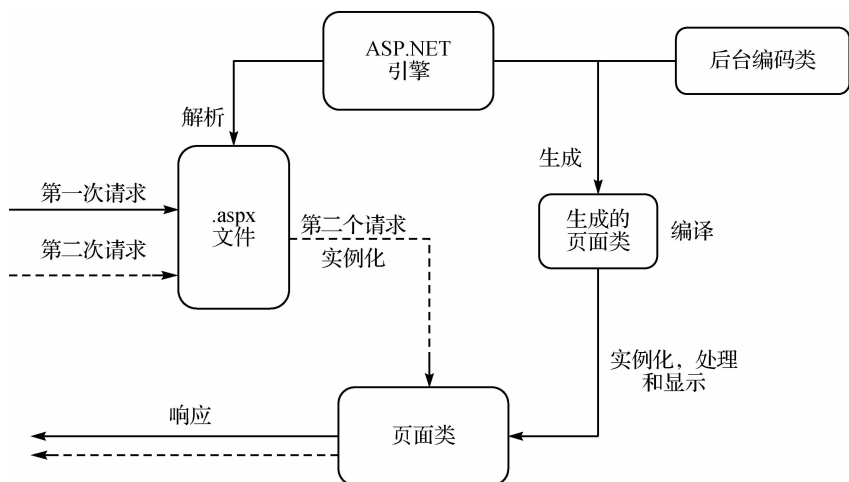


图 1-11 ASP.NET 的运行机制

2. ASP.NET 包含的文件类型

由于 ASP.NET 支持多种语言开发及控件开发,因此文件类型较多,表 1-1 列举了常见的文件类型。

表 1-1 ASP.NET 文件类型

文件扩展名	用途及说明
Global.asax	ASP.NET 应用程序级别的逻辑和事件处理代码都存放在这个文件中,不处理 UI 的生成,并且一般不为响应个别页面请求而被调用
Web.config	系统环境设置文件,相当于 ASP 中的 Global.asa 文件
.aspx	包含 ASP.NET 程序代码的文件,浏览器可执行此类文件,向服务器提出请求
.vb 或 .cs	.vb 文件是存放 VB.NET 代码的文件,.cs 文件是存放 C# 代码的文件
.aspc	可重复使用在多个.aspx 文件中,此文件可含有控件
.ascx	用户控件文件,可包含在多个.aspx 文件中
.asmx	制作 Web Service 的原始文件
.sdl 或 wsdl	Web Service 的描述文件,以 XML 格式保存

1.1.5 实现第一个 ASP.NET 应用程序

1. 创建 ASP.NET 应用程序

打开 Visual Studio,执行“文件”→“新建”→“网站”命令,在打开的如图 1-12 所示的对话框中可以看到,新建网站时不一定需要指定网站路径,直接指定文件路径即可;“位置”可

以指定 3 种类型,常选择“文件系统”;单击“浏览”按钮可以设置文件夹所在位置,避免手动输入路径时产生错误。

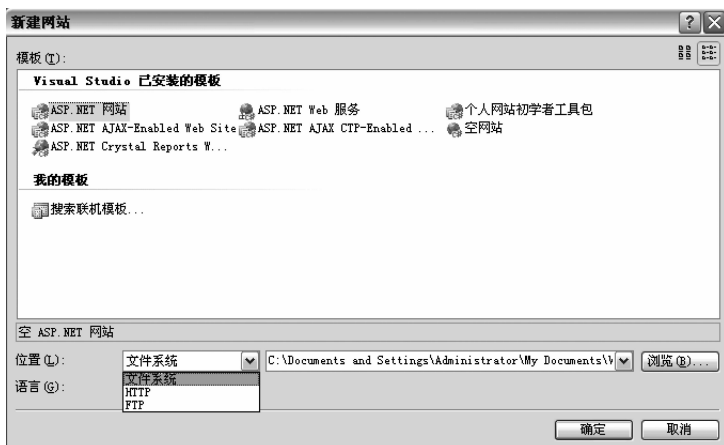


图 1-12 “新建网站”对话框

设置好路径后单击“确定”按钮就新建了一个站点。当然还必须设置开发应用程序使用的编程语言为 C#。

事实上,Visual Studio 提供了 4 种创建 Web 站点的方式,分别是文件系统、本地 IIS、FTP 站点和远程站点,如图 1-13 所示。



图 1-13 创建 Web 站点的位置选择

- 文件系统:这种方式允许将站点文件储存在本地计算机的一个指定文件夹中,或者存储在局域网中的一个共享位置。这就意味着不需要将站点作为 IIS 应用程序来创建就可以进行开发测试等工作。该方式的缺点就是无法使用 IIS 中的某些功能特征,毕竟开发的服务器是简化的 Web 服务器。
- 本地 IIS:本地 IIS 方式是早期的一种创建方式,比文件系统的方式稍微复杂,但可以使用 IIS 的高级功能,所以常常在需要做 Web 测试时使用。该方式的缺点是由于 IIS 服务只有一个进程,调试时也只能有一个用户参与调试过程。

- FTP 站点:FTP 站点方式可以在服务器上保存文件,通过 FTP 访问它们。这是共享环境的一种可能配置,在共享环境下,许多人可以同时使用项目。另外,还可以使用 FTP 设置远程编辑文件。其中,远程服务器有一个虚拟目录映射到 FTP 文件驻留的位置。该方式的最大缺点是,不能使用源代码管理器(如 VSS),团队中的多个人可能对项目进行相互矛盾的改动。
- 远程站点:远程站点方式类似于 FTP 站点方式,所不同的是它不再使用 FTP 方式连接,而是使用 HTTP 方式。这种方式配置站点相当复杂,而且缺点很多,一般很少使用。

综上所述,可以根据实际需要选择适合的开发方式,这里使用的是文件系统方式。站点建好后,会发现 IDE 自动添加了一个页面 default.aspx,可以看到它有两种编辑方式:设计视图和源视图。此时如果单击“运行”按钮(或者按 F5 键),即可运行该应用程序。

提示:第一次运行新建的站点时,会看到如图 1-14 所示的提示对话框,提示未启用调试。单击“确定”按钮,即可自动添加启用了调试的 Web.config 文件,激活调试功能。

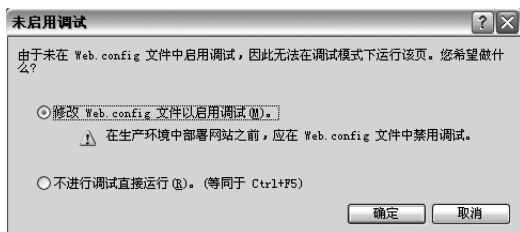


图 1-14 启用调试

2. 解决方案组成

观察刚才创建好的 ASP.NET 应用程序,看看 Visual Studio 自动产生了哪些内容。通过如图 1-15 所示的“解决方案资源管理器”,可以看到如下效果。

项目名称位置显示的是解决方案的路径,下面还默认创建了一个文件夹和三个文件。

- App_Data 文件夹是存放数据的文件夹。
- Default.aspx 就是刚才看到的那个空白页面,注意后缀名是.aspx。
- Default.aspx.cs 文件与 Default.aspx 文件有关系,是代码后置的文件。
- Web.Config 文件是站点配置文件,可以设置是否允许调试等信息。



图 1-15 解决方案资源管理器

3. 代码后置和代码内嵌

1) 代码后置

代码后置是微软公司的一项技术,也是编写 ASP.NET 常用的编码方式。具体方式就像前面所说的页面文件(.aspx)和代码文件(.cs)两个文件相互关联构成一个页面。一般情



况下,.aspx 中没有代码,只有控件和 HTML,而在.cs 文件中编写相关的代码。这样做的好处就是代码和页面内容分离,使代码更清晰。

2) 代码内嵌

与代码后置相对应的是代码内嵌,可以不使用后置的.cs 文件,而完全在.aspx 文件中编写代码。只需要在创建页面时,在如图 1-16 所示的对话框中取消选中“将代码放在单独的文件中”复选框,把代码写在“<%”和“%>”之间就行了。



图 1-16 “添加新项”对话框

4. 控件和事件

ASP.NET 沿用了 WinForms 编程中的事件驱动概念,并将很多复杂的功能封装到控件中。本节将使用 Label、TextBox 和 Button 三种控件。Label 控件用来显示文本,它的 Text 属性包含了要显示的文本内容。TextBox 控件用于用户输入或显示文本,表 1-2 列出了 TextBox 控件常用的属性和事件。Button 控件可以将表单提交给服务器,从而激活服务器端处理。

表 1-2 TextBox 控件常见的属性和事件

属性和事件	类 型	值	说 明
AutoPostBack 属性	Boolean	True、False	指示如果用户更改了控件的内容,是否自动回发到服务器,默认值为 False
ReadOnly 属性	Boolean	True、False	指示该控件内容是否可编辑,默认值为 True
Text 属性	String	—	设置或者获取控件的文本内容
TextMode 属性	TextBoxMode	Singleline、Multiline、Password	控件的模式:单行、多行或密码。设置为密码时,所有的输入显示为“*”
TextChanged 事件	—	—	当文本发生变化且失去焦点时触发的事件。该事件需要配合 AutoPostBack 使用,因为只有自动回发到服务器,文本改变的事件才会执行

Button 控件也有 Text 属性,它只是用来设计按钮上显示的文字。Click 事件在控件单击时触发,可以在设计模式下双击 Button 控件来添加单击事件的代码,同时在属性窗口中双击 Click 后面的空白区域,也可以添加事件的代码。

5. 问候程序的实现

- (1)新建页面 HelloASP.NET.aspx。
- (2)从工具栏拖入两个 Label、一个 TextBox 和一个 Button 控件,如图 1-17 所示。

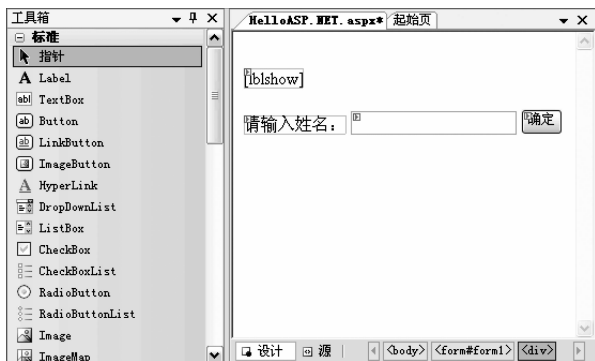


图 1-17 控件布局

提示: 在 ASP.NET 中,控件在页面上的位置默认是相对定位,如果想要像 WinForms 中那样,实现控件被拖到哪里就显示在哪里,可以在选中控件的情况下,执行菜单栏中的“布局”→“位置”→“绝对定位”命令,即可实现位置拖动。

(3)编写事件方法。这里要实现的功能就是在输入用户名后,单击“确定”按钮,显示相应的欢迎信息,因而所要编写的事件就是单击“确定”按钮的事件。获取和设置用户控件值的方式就是常说的“控件+事件”的编码方式。代码如下。

```
protected void btnOK_Click (object sender, EventArgs e)
{
    lblshow.Text = "欢迎" + username.Text + "!";
}
```

1.2 任务 2:体验 ASP.NET 应用程序的快速开发特性

1.2.1 任务描述

编写一个显示 Northwind 数据库中雇员表信息的页面,要求显示所有雇员的信息,包括 EmployeeID、LastName、FirstName、BirthDate、Address 这 5 个字段,表头用中文显示,每页显示 6 条记录,如图 1-18 所示。另外,还要有排序、分页的功能。

雇员编号	名	姓	出生日期	家庭地址
1	Davolio	Nancy	1948-12-8 0:00:00	507 - 20th Ave. E. Apt. 2A
2	Fuller	Andrew	1952-2-19 0:00:00	908 W. Capital Way
3	Leverling	Janet	1963-8-30 0:00:00	722 Moss Bay Blvd.
4	Peacock	Margaret	1937-9-19 0:00:00	4110 Old Redmond Rd.
5	Buchanan	Steven	1955-3-4 0:00:00	14 Garrett Hill
6	Suyama	Michael	1963-7-2 0:00:00	Coventry House Miner Rd.

图 1-18 显示雇员信息列表

1.2.2 用 GridView 显示 Northwind 数据库的雇员表

1. 将雇员信息表拖入页面

执行“视图”→“服务器资源管理器”命令,打开服务器资源管理器,如图 1-19 所示。



图 1-19 服务器资源管理器

选中“数据连接”,右击,在弹出的快捷菜单中选择“添加连接”命令,在打开的“选择数据源”对话框中选择“Microsoft SQL Server”选项(如图 1-20 所示),单击“继续”按钮打开“添加连接”对话框(如图 1-21 所示),连接到 Northwind 数据库。在“服务器资源管理器”的“数据连接”中找到雇员信息表 Employees 并将其拖入页面,如图 1-22 所示。

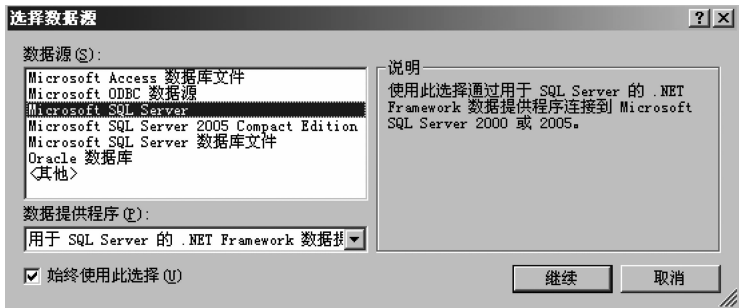


图 1-20 “选择数据源”对话框

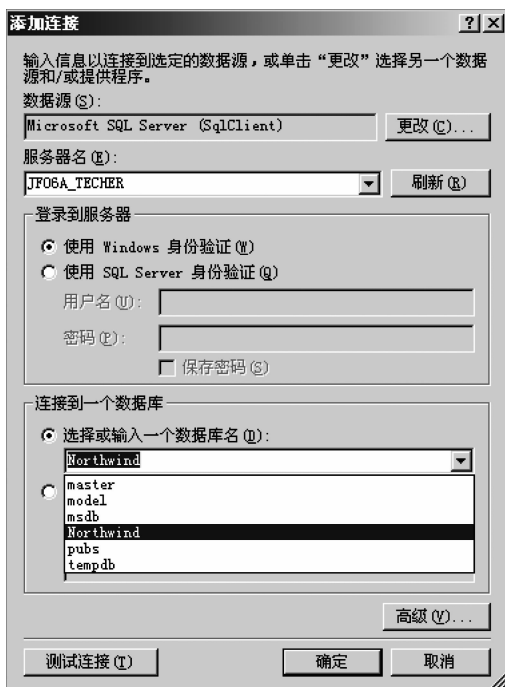


图 1-21 “添加连接”对话框

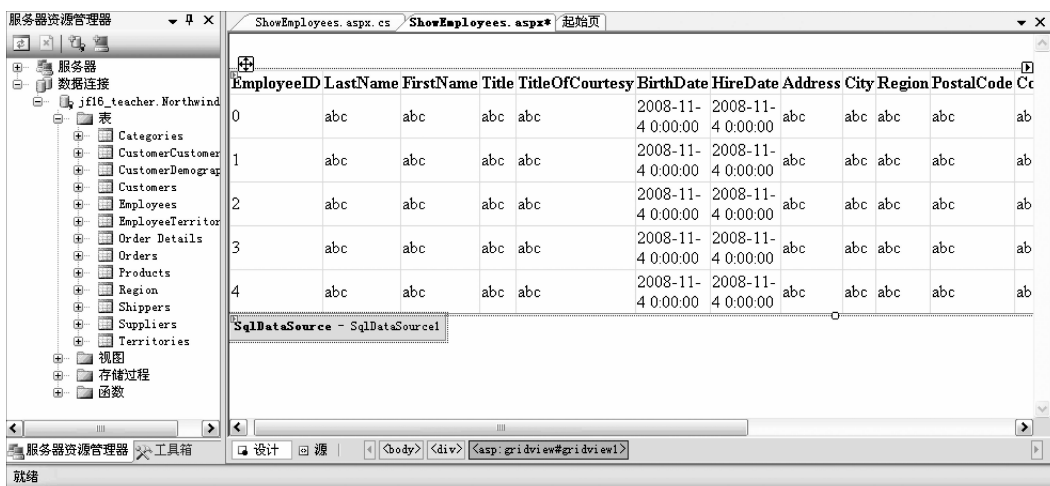


图 1-22 将雇员信息表拖入页面

2. 显示内容的选择

把雇员信息表拖入页面后,所有的字段都显示出来了,但只需要显示 EmployeeID、LastName、FirstName、BirthDate、Address 这 5 个字段,可以在智能标记中单击“编辑列”,如图 1-23 所示,打开“字段”对话框。

在“字段”对话框中可以设置显示的内容、标题以及显示顺序,如图 1-24 所示。单击对话框中的 和 按钮可以调整显示顺序,单击 按钮可删除选定字段。当选定某字段时,在右侧的字段属性列表框中可设置其属性。这里只需要修改显示的标题头即可,即在 HeaderText 栏的属性值中填写对应的中文标题。



图 1-23 编辑列

图 1-24 “字段”对话框

1.2.3 设置分页和排序

显示雇员信息列表内容一般比较多,需要设置分页,只需选中“启用分页”复选框就可以实现,如图 1-25 所示。此外还要设置一页显示多少条,可以通过 GridView 属性栏中的分页设置来实现,如图 1-26 所示。其中 AllowPaging 属性是设置是否打开分页功能;PageIndex 设置的是当前显示第几页,0 页是第一页;PageSize 属性是设置每页显示多少条,默认是 10 条。

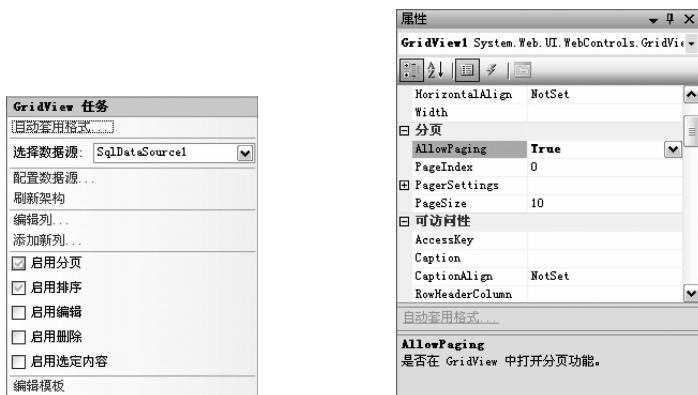


图 1-25 设置分页

图 1-26 分页和排序

在设置分页时,还有个“启用排序”复选框,启用排序就是单击标题头时,如单击“出生日期”时就会按照出生日期所对应的字段进行排序。

最后希望这个页面显示得好看些。Visual Studio 提供了很多样式,通过选择“自动套用

格式”命令(如图 1-27 所示)在打开的“自动套用格式”对话框(如图 1-28 所示)中可以方便地套用。



图 1-27 选择“自动套用格式”命令



图 1-28 “自动套用格式”对话框

至此,页面设计完成,其运行效果如图 1-29 所示。这期间没有编写一行代码,不仅实现了显示、排序和分页的功能,还有不错的显示效果,这就是使用 ASP. NET 开发的简便之处。



图 1-29 运行效果

1.3 项目小结

本项目完成了搭建开发环境、编写问候程序和显示雇员信息三个任务,应掌握以下技能。

- 能正确完成 IIS 的安装和虚拟目录的设置。
- 理解并体会 ASP. NET 的特色和优势。



- ## 1.4 思考与练习

- 16

1.5 项目实战

1. 根据用户输入的年龄显示相应信息

1) 任务说明

编写一个 Web 应用程序,接收用户输入的姓名和年龄,根据用户输入的年龄显示相应信息。若年龄大于 40 岁,显示“对不起 * * *,你不是年青人!”;若年龄小于等于 40 岁,则显示“欢迎 * * *,你风华正茂!”。

实现的效果如图 1-30 所示。



图 1-30 程序运行效果图

2) 技能要点

(1) 创建 ASP.NET 站点。

(2) 创建动态页面。

(3) 代码后置的开发模式。

3) 实现思路

(1) 新建一个 Web 应用程序。

(2) 定义三个 Label 控件,分别显示姓名提示信息、年龄提示信息和结果信息。

(3) 定义两个 TextBox 控件,用来接收姓名和年龄。

(4) 定义一个 Button 控件,用来提交信息。

(5) 使用 if 结构根据用户输入的年龄输出相应信息。

2. 显示考试管理系统中所有教师的信息

1) 任务说明

MySchool 是考试管理系统的数据库,现要求显示学校所有教师的信息,包括 LoginID、TeacherName 和 Sex 三个字段,表头用中文显示,每页显示 10 条记录。

输出的效果如图 1-31 所示。



图 1-31 显示所有教师的信息



2) 技能要点

(1) ASP.NET 的高效开发。

(2) 使用 GridView 控件完成数据的显示。

3) 实现思路

(1) 利用服务器资源管理器连接 MySchool 数据库。

(2) 把 Teacher 表拖入页面。

(3) 单击 GridView 控件的智能标记, 在显示的“GridView 任务”窗格中进行相应的设置。

(4) 单击“编辑列”, 在弹出的对话框中进行表头的设置。

实现简单的用户注册登录系统

项目 1 中介绍了 IIS 的安装和虚拟目录的配置,如何使用 Visual Studio 创建 ASP.NET 应用程序,体验了 ASP.NET 开发的方便和快捷,了解了站点的开发也可以像 WinForms 那样用“控件+事件”的方式实现。本项目将学习 ASP.NET 中常见的系统对象 Page、Request、Response、Cookie、Session、Application 的常用属性和方法,页面数据传递的方式,Session、Cookie、Application 三种状态保持方式的区别及使用它们可以解决的问题等。

【工作任务】

- ④ 实现用户注册功能。
- ④ 实现用户登录功能,登录成功后跳转到欢迎页面。
- ④ 统计在线人数。

【技能目标】

- ④ 熟练掌握 ASP.NET 中常用的系统对象。
- ④ 熟练掌握 ASP.NET 中的状态保持方式。
- ④ 熟练掌握 ASP.NET 中数据传递的方式。

【常用术语】

Page 对象、Request 对象、Response 对象、Application 对象、Session 对象、Cookie 对象、Server 对象、PostBackUrl 属性、IsCrossPageBack 属性

2.1 任务 1: 用户注册

2.1.1 任务描述

编写一个用户注册程序。用户填写完注册信息后单击按钮提交,跳转到确认信息页面,用户确认信息无误后单击“确认”按钮即完成注册。实现效果如图 2-1 和图 2-2 所示。



图 2-1 填写注册信息页面



图 2-2 确认注册信息页面

2.1.2 系统对象

在 ASP.NET 页面中包含一系列类,在页面中可以直接使用的称为系统对象。表 2-1 列出了 ASP.NET 中常见的系统对象。



表 2-1 ASP.NET 中常见的系统对象

对象名	说明
Page	该对象指向页面自身,在整个页面的执行期内都可以使用该对象
Request	该对象封装了由 Web 浏览器或其他客户端生成的 HTTP 请求的细节(参数、属性和数据),提供从浏览器读取信息或者读取客户端信息等功能,用于页面请求期
Response	该对象封装了返回到 HTTP 客户端的输出,提供向浏览器输出信息或者发送指令,用于页面执行期
Server	该对象提供了服务器端的一些属性和方法,如页面文件的绝对路径等
Application	为所有用户提供共享信息,作用于整个应用程序运行期
Session	为某个用户提供共享信息,作用于用户会话期
Cookie	保持于客户端的共享信息方式

2.1.3 Page 对象

ASP.NET 的页面类继承自 System.Web.UI.Page 类,每一个 ASP.NET 的页面对应一个页面类。Page 对象就是页面类的实例,表示客户机请求的页面,即已编译运行的页面(内存中的实例),该页面来自于.aspx 文件。

1. @ Page 指令

在 Visual Studio 的 IDE 中,新建页面的.aspx 文件有两种视图:设计视图和源视图。打开源视图,在第一行可以看到这样一行代码:

```
<% Page Language="C#" AutoEventWireup="True" CodeFile="default.aspx.cs" Inherits="_default"%>
```

这就是常说的页面指令,@ Page 指令定义了 ASP.NET 页用于编译和解析的属性,每个.aspx 页面只能有一个@ Page 指令。从代码中可以看到@ Page 指令的写法,其中的一些参数含义如下。

- Language:指定页面代码和后置代码使用的语言,需要注意的是,这里只支持.NET 框架中的语言。
- AutoEventWireup:设置页面是否自动调用网页事件,默认(不写时)为 True。
- CodeFile:指定代码后置文件名,后置代码,该后置代码与页面是局部关系。
- Inherits:页面类。

为了说明 AutoEventWireup 的作用,可以做一个试验:在页面中放置按钮,并在 Page_Load 事件(页面加载事件)中设置断点(不需要编码),通过调试可以看到,默认情况下 Page_Load 事件是执行的,当该属性设置为 False 时 Page_Load 事件没有执行。

2. ASP.NET 页面传值

1) 页内数据传递

页内数据传递是最简单的页面数据传递形式。如图 2-3 所示,对于用户单击按钮等引起回传的控件,所有页面上的服务器端控件的值都要回传,可以使用 Request 方式获得所有

提交过来的内容。在 ASP.NET 中这些是不需要处理的,ASP.NET 都已经封装好,可以使用“控件.属性”的方式直接访问控件的相关内容。总之,因为有了页面 PostBack 机制,使得编写 ASP.NET 页面可以像 WinForms 那样直观快捷。不需要在 HTML 标签中设置 Post 或 Get 方式提交,也不用编写 Request 代码,就可以直接使用控件的属性值。

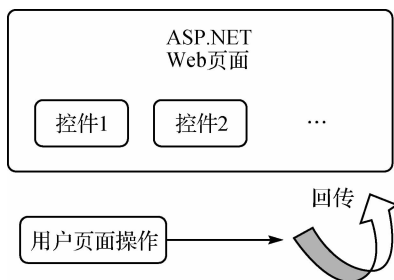


图 2-3 页面数据的回传

但是,页面回传机制也带来了一些问题,比如有这样一个页面,如图 2-4 所示。

登录窗口	
用户名	
密码	
登录	

图 2-4 示例页面

该页面中有用户名和密码两个数据框,并且当用户单击“登录”按钮时,后台代码获得用户输入的信息,可以用来验证用户的身份。为了让用户更清楚用户名和密码的输入位置,在页面中编写如下代码。

```
protected void Page_Load(object sender,EventArgs e)
{
    txtLoginId.Text="请在此处输入用户名!";
    txtLoginPwd.Text="请在此处输入密码!";
}
```

这段代码的本意是希望提示用户名和密码的输入位置,但 Page_Load 是页面加载事件,当页面加载时该代码就能执行。也就是说,当用户单击“登录”按钮后,页面回传,这段代码又一次被执行,结果就无法获取到用户输入的内容。

为了解决这个问题,ASP.NET 提供了一个特殊的属性——Page.IsPostBack 来进行判断,使用过程中可以直接写成 IsPostBack。它是一个布尔值,当该值为真时则页面为回传,否则就是首次加载,所以将上面的代码修改如下。

```
protected void Page_Load(object sender,EventArgs e)
{
    if(!Page.IsPostBack)
```

若页面首次加载,则文本框中显示相应信息

```

txtLoginId.Text="请在此处输入用户名!";
txtLoginPwd.Text="请在此处输入密码!";
}
}

```

2) 跨页数据传递

处理完页内数据的传递,如果需要把控件的值传给另外一个页面该怎么办呢?这种情况在 Web 开发中是常见的现象。ASP.NET 中有一种跨页发送的形式,就是将页面上所有控件的值发送到另一个页面,如图 2-5 所示。



图 2-5 跨页发送

当然,只有源页面和目标页面都在同一个应用程序,目标页面才可以访问该页面上控件的内容。在目标页面访问源页面中的控件可以使用:

```
this.PreviousPage.FindControl("控件 ID")
```

其中,PreviousPage 指向源页面,FindControl 方法则通过控件 ID 找到相应控件对象。

下面通过一个例子实现跨页发送——利用 IsCrossPagePostBack 属性实现两个页面之间的数据传送。

首先创建两个页面,一个是发起提交的页面(源页面)Source.aspx,一个是获取内容的页面(目标页面)Target.aspx。在目标页面上放置一个 Label 用于显示内容,在源页面上放置文本框和按钮(运行效果如图 2-6 所示),并设置按钮 PostBackUrl 的属性为目标页面。



图 2-6 源页面

提示: 目前具有 PostBackUrl 属性的控件只有 Button、LinkButton 和 ImageButton 三种,所以要实现跨页传递,一定要使用这三种按钮才行。这里使用 Button 控件,并设置其 PostBackUrl 的属性为 Target.aspx。

为了更好地理解跨页发送的过程,在 Source.aspx 页面上做的处理代码如下。

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {

```

```

        this.tbKeyWord.Text="请输入查询内容";
    }
    if (Page.IsCrossPagePostBack)//判断页面是否使用跨页提交
    {
        this.tbKeyWord.Text="跨页数据传送!";
    }
}

```

代码中使用了 IsCrossPagePostBack 属性,与 IsPostBack 类似,它也是一个布尔值。当跨页提交时,IsCrossPagePostBack 属性值为 True。

然后在目标页面编写代码即可,Target.aspx 的代码如下(此处假设 Source 页面上文本框的 ID 是 txtKeyWord,Target.aspx 页面上的 Label 控件的 ID 是 lblShowURL)。

```

protected void Page_Load(object sender,EventArgs e)
{
    if (Page.PreviousPage!=null)
        if (PreviousPage.IsCrossPagePostBack==True)
        {
            this.lblShow.Text="您的查询条件为:"+
            ((TextBox)this.PreviousPage.FindControl("tbKeyWord")).Text;
        }
}

```

运行时当单击源页面上的“查询”按钮时,提交到目标页面,目标页面执行结果如图 2-7 所示。

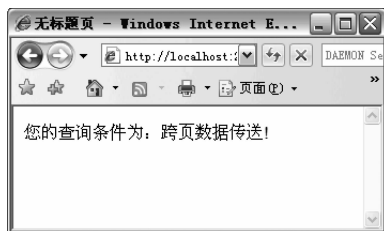


图 2-7 目标页面运行结果

显然,源页面的“this.txtKeyWord.Text="跨页数据传送!"”这段代码被执行了。如果设置断点跟踪调式,会发现如下几点。

- (1)当跨页提交发生后,程序直接进入目标页面代码,并不执行源页面的代码。
- (2)执行目标页面中的 Page.PreviousPage 时,程序转入源页面,并执行该页面内容。
- (3)之后,程序回到目标页面中的 Page.PreviousPage 处,继续执行该页面的代码。

2.1.4 Response 和 Request 对象

页面输入/输出涉及 Request 和 Response 对象。Response 被用于回应客户浏览器,告诉浏览器回应内容的报头、服务器端的状态信息以及输出定制的内容。

Response 对象的常用属性是 Buffer 属性,用来设置是否在服务器端开启缓存功能。当 Buffer 属性设置为 True 时,启用服务器端的缓存功能,此时服务器会将整个页面处理后再传给客户端浏览器;否则服务器将边处理页面边将处理完毕的数据传递给客户端浏览器,用户看到的网页可能是不连续的。

Response 对象的常用方法如下。

- Write(): 用于向浏览器发送指定的文本内容。
- End(): 可以使 Web 服务器停止当前的程序并返回结果。
- Redirect(): 用于将客户端重定向到新的 URL。该方法常被用来根据某条件将用户引向不同的页面。

当客户端浏览器请求 ASP.NET 应用程序时,CLR 将客户端请求信息封装在 Request 对象中,包括请求报头(Header、浏览器类型、浏览器版本号、用户使用的语言及编码方式等)、请求方法(Post 或 Get)、参数名称、参数值等。

Request 对象的常用属性如下。

- QueryString 集合: 主要用于收集 HTTP 中的 Get 请求发送的数据,如果一个请求事件中被请求的程序 URL 中出现“?”后的数据,这些数据被称为“URL 的附加信息”,通常用来在不同网页之间传送数据。
- Form 集合: 与 QueryString 类似,但它用于收集 Post 方法发送的请求数据(Get 方法一般只能传递 256B 的数据,而 Post 方法可以达到 2 MB)。
- Server Variable(环境变量)集合: 其中包含了服务器和客户端的系统信息。
- Params 集合: 是以上三种方式的集合,它不区分是由哪种方式传递的参数。相比而言,该方法需要在最大的范围(如 URL 参数、表单、环境变量等)寻找匹配字段,所以效率较低。

提示: (1)Request.Server Variable["HTTP_USER_AGENT"]: 可获取当前客户端浏览器的版本信息。

(2)Request.Server Variable["HTTP_ACCEPT_LANGUAGE"]: 可获取当前客户端浏览器使用的语言。

2.1.5 用户注册功能的实现

1. 页面设计

1) 注册页 RegisterPage.aspx

注册页实现用户输入个人信息,并需要有一个提交按钮,控件布局如图 2-1 所示。从工具箱中拖入 1 个 Table、4 个 TextBox 和 1 个 Button 控件,设置“密码”和“确认密码”文本框的 TextMode 属性值为 Password,设置提交按钮的 PostBackUrl 属性为确认信息页面的 URL。

2) 确认信息页 ValidatePage.aspx

确认信息页显示注册页中用户输入的个人信息,并需要“确认”和“返回”两个按钮,控件布局如图 2-2 所示。从工具箱中拖入 1 个 Table、3 个 Label 和 2 个 Button 控件,设置“返回”按钮的 PostBackUrl 属性为注册页面的 URL。

2. 实现思路

- (1) 将注册页面中提交 Button 控件的 PostBackUrl 属性设置为确认信息页面的 URL。
- (2) 通过确认信息页面的 IsCrossPageBack 属性来判断是否为跨页请求。
- (3) 利用 FindControl 方法在确认信息页面中查找注册页面的控件。

3. 关键代码

注册页无代码设计, 信息确认页面中的源代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Page.PreviousPage != null)
    {
        if (PreviousPage.IsCrossPagePostBack == True)
        {
            TextBox SourceName =
                (TextBox)Page.PreviousPage.FindControl("txtName");
            if (SourceName != null)
            {
                lblName.Text = "<br>姓 名:" + SourceName.Text;
            }
            TextBox SourcePsd =
                (TextBox)Page.PreviousPage.FindControl("txtPsd");
            if (SourcePsd != null)
            {
                lblPsd.Text = "<br>密 码:" + SourcePsd.Text;
            }
            TextBox SourceEmail =
                (TextBox)Page.PreviousPage.FindControl("txtEmail");
            if (SourceEmail != null)
            {
                lblEmail.Text = "<br>电子邮件:" + SourceEmail.Text;
            }
        }
    }
}

protected void btnSubmit_Click(object sender, EventArgs e)
{
    Response.Write("<script>alert(\"用户注册成功!\")</script>");
}

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

```

{
    Response.Redirect("RegisterPage.aspx");
}
}

```

2.2 任务 2: 用户登录

2.2.1 任务描述

编写一个用户登录程序, 界面如图 2-8 所示。如果用户输入的用户名和密码正确, 则记录用户登录状态和登录时间, 并跳转到欢迎页面。欢迎页面显示用户名、登录信息和浏览器信息。



图 2-8 “用户登录”窗口

2.2.2 Cookie 对象

在 Internet 中, Cookie 是指小型文本文件, 它是由网络服务器发送出来, 存储在网络浏览器上的数据。当访客下次浏览该站点时, 可以通过该浏览器获得这些信息, 如上次访问的位置、花费的时间或用户选项(如样式表所控制的页面风格)等。Cookie 作为 HTTP 头部文件的一部分在浏览器和服务器之间互相传递, 这些数据和它们的传递过程对于用户来说不是直接可见的。

Cookie 对象为 Web 应用程序保存用户的相关信息提供了一种有效的方法, 可以使用 Cookie 来保存用户的登录信息。Cookie 对象分别属于 Request 对象和 Response 对象, 每一个 Cookie 对象都属于集合 Cookies, 所以访问 Cookie 的方式就可以使用索引器的方式。语法如下。

```

Response.Cookies[Cookie 的名称].Value=变量值;           //写入 Cookie
String 变量名=Request.Cookies[Cookie 的名称].Value; //读取 Cookie
Cookie 所对应的类名是 HttpCookie, 所以添加新 Cookie 还有一种方法。
HttpCookie hcCookie=new HttpCookie("Cookie 的名称","值");
Response.Cookies.Add(hcCookie);

```




由于 Cookie 存储于客户端,出于安全考虑,一般不使用它进行客户登录状态的判断。Cookie 对象的常用属性如下。

- Name: Cookie 变量的名称。
- Value: 取得或者设置 Cookie 变量的内容(值)。
- Expires: 设定 Cookie 变量的有效时间。

提示: (1) 如果没有设置 Cookie 的有效日期,它们仅保存到关闭浏览器程序为止;如果将 Cookie 对象的 Expires 属性设置为 MaxValue,则表示 Cookie 永远不会过期。

(2) 由于 Cookie 是存储在客户端的,所以它的使用就受到客户端浏览器的限制。大多数浏览器对 Cookie 的大小限制为 4 KB,所以在 Cookie 中存储大量数据是不现实的。

(3) 用户可以通过修改浏览器设置将 Cookie 禁用。

Cookie 一般用于在用户未登录的情况下记录用户的信息,如用户的浏览记录、用户上次登录时间等内容,进而可分析用户多久访问一次网站及用户的关注内容等。

2.2.3 Session 对象

与 Cookie 不同的是,Session 是存储在服务器端的数据。针对每一个连接,系统自动分配一个 ID 来标识每一个不同的用户,该 ID 在客户端和服务端间传递,达到唯一标识某一个用户的目的。要得到该 ID,可以使用 Session.SessionID 属性。每一次访问都会单独获得一个 SessionID。在 Web 应用程序中,当一个用户访问该应用程序时,Session 类型的变量可以供这个用户在该 Web 应用的所有页面中共享数据。

Session 也是集合,同样可以使用索引器方式进行访问。不过 Session 比 Cookie 使用更加方便,直接进行读取即可。语法格式如下。

```
Session["Session 名称"]=值;
```

```
变量=Session["Session 名称"];
```

Session 对象的特点如下。

- Session 对象包含某一个用户的状态信息,此信息仅面向该连接,不与其他用户共享。
- 会话超时或者过期,服务器即刻清除 Session 对象,释放其所占资源。
- 会话期通过 SessionID 传递状态信息,不像 Cookie 那样将所有内容传输,客户端仅对 SessionID 可见,而对状态信息的内容不可见。

Session 对象的常见属性和方法见表 2-2。

表 2-2 Session 对象的常见属性和方法

属性和方法	说 明
SessionID 属性	包含唯一的用户会话标识符,它可用于在整个会话过程中记录用户信息
Timeout 属性	设置用户超时的时间,单位为分钟。在该段时间内,即使没有任何操作,用户状态也不会被清除。注意,此处的设置是针对服务器清除 Session 的时间而言的,因为如果一段时间内没有操作,服务器无法知道用户是否在线,所以以超时时间为限。而客户端有自己的处理机制,不同的浏览器也不相同,一般的,关闭浏览器窗口就会造成 Session 丢失



续表

属性和方法	说 明
IsNewSession 属性	如果已按当前请求创建会话,那么该属性将返回 True
Clear 方法	从会话状态集合中移除所有的键和值
Abandon 方法	结束 Session,取消当前会话

虽然 Session 有很多优点,但并不表示 Cookie 就没用了。Session 占用服务器的资源,如果存储大量信息,当站点访问量大时将影响服务器的性能。对比两者的优缺点,一般将登录状态信息等存在安全需求的内容存储在 Session 中,而用户的浏览记录、上次访问时间等内容存储在 Cookie 中。

Session 的典型应用是存放用户的登录信息,如用户名、密码、权限、角色等,应用程序(如 E-mail 服务、网上银行等系统)根据这些信息可以进行身份验证和权限验证。

2.2.4 Server 对象

Server 对象封装了一些 Web 服务器相关的常用方法,这些常用方法及其说明见表 2-3。

表 2-3 Server 对象的常用方法

方 法	说 明
HtmlEncode	此方法带一个字符串参数,可将其编码,使其在浏览器中正确显示。例如,希望在页面输入一个“ ”,又不想在浏览器中显示成换行,可以使用此方法
HtmlDecode	此方法与 HtmlEncode 方法相反,对已编码内容解码
MapPath	此方法带有一个虚拟路径,可返回该路径在物理磁盘中的准确位置。在 Web 开发过程中使用频率比较高,一般实现文件读写时使用
UrlEncode	对 URL 地址进行编码,对于 URL 传输含有“#”“&.”等特殊字符的参数时,需要进行编码,否则后面的内容不会被识别
UrlDecode	此方法与 UrlEncode 方法相反,对 URL 地址进行解码
Execute	在当前页面执行参数指定的页面,执行完成后继续执行本页面
Transfer	参数指定的页面处理完成后,页面执行就结束,不像 Execute 那样还要返回控制权给先前的页面

MapPath 的语法格式如下。

Server.MapPath(string Path)

其中,Path 参数为 Web 服务器上的虚拟路径,如果 Null 作为参数,则返回应用程序所在目录的物理路径。

2.2.5 用户登录功能的实现

用户登录显然要使用 Session 记录用户状态,登录时间之类的内容使用 Cookie 比较合

适。创建一个用户的类,用于记录用户名和密码。

1. 用户类 User. CS

```
using System;

public class User
{
    private string username;           //用户名
    private string password;          //密码
    public User()
    {
    }
    public string Username
    {
        get{return username;}
        set{username=value;}
    }
    public string PassWord
    {
        get{return password;}
        set{password=value;}
    }
}
```

该用户类没有赋值的方法,仅用来存储用户名和密码。注意在 Web 程序开发中添加类时,会出现如图 2-9 所示的提示。

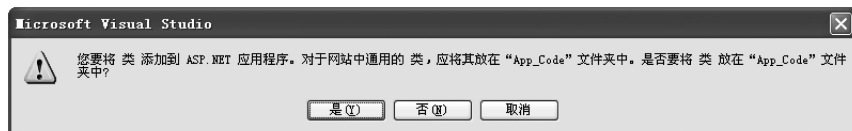


图 2-9 添加类时的提示

该提示是说对于网站中的通用类,应该放置于 App_Code 文件夹中,如果没有该文件夹,则会自动创建。

2. 登录页面 Login.aspx

登录页面需要用户输入用户名和密码,并需要有个提交按钮,以及输入错误时显示提示信息的 Label。登录界面如图 2-8 所示。

3. 后台代码 Login.aspx.cs

该页面的后台代码只有一个方法 btnSubmit_Click。

```
protected void btnSubmit_Click(object sender,EventArgs e)
{
}
```


4. 后台代码 Welcome.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //判断 Session 中用户名是否已被记录
    if (Session["User"] == null)
        Response.Redirect ("Login.aspx");
    //将 Session 中的对象取出来
    User user = Session["user"] as User;
    //浏览器及版本
    Response.Write("欢迎," + user.Username + "<br>");
    Response.Write("您的浏览器版本是:");
    Response.Write(Request.ServerVariables["HTTP_USER_AGENT"].ToString() +
"<br>");
    //输出 Cookie 中的登录时间
    Response.Write("您的登录时间为:" + Request.Cookies["LoginTime"].Value +
"<br>");
}
```

该代码对用户状态进行判断,以确定用户是否登录,未登录则跳转到登录页

将Session中保存的用户信息读取出来的方式,注意as的用法

2.3 任务 3:统计在线人数

2.3.1 任务描述

编写一个统计在线人数的程序。统计在线人数是为了网站管理者可以知道当前在线用户人数的数量,然后根据用户数量来观察服务器或者程序的性能,从而可以直观地了解网站的吸引力或者网站程序的运行效率。

统计在线人数的程序运行效果如图 2-11 所示。



图 2-11 统计在线人数的程序运行效果图

2.3.2 Application 对象

Application 对象是应用程序级的对象,用来在所有用户间共享信息,并可以在 Web 应用程序运行期间持久地保持数据。当应用程序第一次启动时,会创建 Application 对象。创建成功后,在整个应用程序中都可以访问该对象的值,直到应用程序结束。Application 对象的用法和 Session 一样,语法格式如下。

```
Application["Application 名称"]=值;
```

```
变量=Application["Application 名称"];
```

表 2-4 列出了 Application 对象的常用属性,表 2-5 列出了 Application 对象的常用方法。

表 2-4 Application 对象的常用属性

属 性	说 明
All	返回全部 Application 对象变量到一个对象数组
AllKeys	返回全部 Application 对象变量到一个字符串数组
Count	取得 Application 对象变量的数量
Item	允许使用索引或 Application 变量名称传回内容值

表 2-5 Application 对象的常用方法

方 法	说 明
Add	新增一个 Application 对象变量
Clear	清除全部的 Application 对象变量
Get	使用索引值或变量名称回传变量值
Set	使用变量名称更新一个 Application 对象变量的内容
Lock	锁定全部 Application 变量
UnLock	解除锁定 Application 变量

由于 Application 是应用程序级别的状态保持方式,所以能够保存超越会话级别的状态信息,如在线人数等。

Session、Cookie 和 Application 的区别如下。

(1)Session 能够保存相对少量的、简单的数据,这是面向单个用户、单个连接的信息,保存于服务器端(会占用服务器资源)。当 Session 超时或者被关闭时将清空保存数据。由于用户停止使用应用程序后仍然会存在一段时间(一般是 20 min),所以 Session 对象保存数据的效率比较低,适用于保存少量数据。

(2)Cookie 保存少量的、简单的数据,一般不允许超过 4 KB(与浏览器有关),使用期限可以进行灵活设定。面向单个用户,保存于浏览器中(也就是客户端)。由于信息存储在客户端,最好不要保存敏感的信息。

(3)Application 能够保存任意大小的信息,在整个应用程序运行周期中有效,是面对所有应用程序用户的,保存于服务器端。当网站访问量过大时,可能造成性能上的瓶颈。因此

不推荐使用该对象来保存大的数据集合。

2.3.3 统计在线人数功能的实现

1. 实现思路

结合 Global.asax 文件,用 Application 和 Session 实现在线人数统计。由于程序中只用到一个 Application,所以程序占用的系统资源可以忽略不计。

2. 关键代码

1)Global.asax 文件

当网站开始服务的时候(Application 开始的时候),程序设置 Application["online"]为 0。当用户进入网站(Session 开始的时候)的时候,锁定 Application,将 Application("online")加 1。用户退出网站的时候,Application("online")减 1。这样就实现了在线人数的统计。关键代码如下。

```
<%@ Application Language="C#" %>
<script runat="server">
void Application_Start(object sender,EventArgs e)
{
    //应用程序启动时运行的代码
    Application["online"]=0;
}
void Session_Start(object sender,EventArgs e)
{ //新会话启动时运行的代码
    Application.Lock();
    Application["online"]=Convert.ToInt32(Application["online"])+1;
    Application.Unlock();
}
void Session_End(object sender,EventArgs e)
{ //会话结束时运行的代码
    Application.Lock();
    Application["online"]=Convert.ToInt32(Application["online"])-1;
    Application.Unlock();
}
</script>
```

2)在线人数显示页面

```
protected void Page_Load(object sender,EventArgs e)
{
    if (!IsPostBack)
    {
        this.lblResult.Text="在线人数:"+Application["online"].ToString();
    }
}
```



```

    }
}

```

2.4 项目小结

本项目介绍了 ASP.NET 中常用的系统对象、状态保持方法以及数据传递方式,主要包括以下内容。

- Page 对象是页面类的示例,页面类继承自 System.Web.UI.Page。
- @ page 指令定义了 ASP.NET 页用于编译和解析的属性,每个.aspx 页面只能有一个@ page 指令。
- ASP.NET 页内数据传递是通过回传机制 PostBack 实现的。在页面实现回传的过程中,为判断页面是否是第一次加载要用到 Page 对象的 IsPostBack 属性。
- 跨页提交中,在目标页面中 Page 对象的 PreviousPage、IsCrossPagePostBack 属性的正确使用方法,以及源页面中服务器控件 PostBackUrl 属性的设置。
- ASP.NET 系统对象 Page、Request、Response、Cookie、Session、Application 的常用属性和方法。
- Session、Cookie 和 Application 三种状态保持方式的区别,以及使用它们可以解决的问题。
- Server 对象封装了 Web 服务器相关的一些方法,常用的是返回实际物理路径的 MapPath 方法。

2.5 思考与练习

1. App_data 目录用来放置()。
 - A. 专用数据文件
 - B. 共享文件
 - C. 被保护的文件
 - D. 代码文件
2. 当跨页传递数据时,哪个属性标识源页面是跨页提交?()
 - A. IsPostBack
 - B. IsCrossBack
 - C. PreviousPage
 - D. IsCrossPagePostBack
3. 以下是跨页提交的代码,下列说法正确的是()。

```

Protected void Page_Load(object sender,EventArgs e)
{
    if(PreviousPage.IsCrossPagePostBack==True)
        this.lblshowURL.Text="您的查询条件为:"+
        ((TextBox)this.PreviousPage.FindControl("txtKeyWord")).Text;
}

```

- A. 由源页面提交后,Label 控件显示“您的查询条件为:×××”(×××代表源页面相

关控件值)

B. 由源页面提交后,Label 控件不显示任何结果

C. 直接打开本页时报错

D. 直接打开本页时,Label 控制显示“您的查询条件为:”

4. 下列有关状态保持的说法,正确的是()。(多选)

A. Session 保存在服务器端,占用服务器资源

B. Cookie 保存在客户端,使用时占用带宽资源

C. Application 是应用程序级的变量,所以在应用程序的任何部分代码都可以直接写成:String appstr=Application("Name");

D. Cookie 比 Session 安全

5. 下面可以获取网站根目录的是()。(多选)

A. Request. ApplicationPath

B. Server. MapPath

C. Response. Redirect

D. Server. Execute

2.6 项目实战

1. 为校友录创建班级注册页面

1)任务说明

为校友录创建班级注册页面,如图 2-12 所示,实现单击“创建新班级”按钮即跳转到确认注册信息页面,如图 2-13 所示,用户确认信息无误后单击“确认”按钮,完成班级注册。



图 2-12 班级注册页面



图 2-13 确认注册信息页面

2)技能要点

(1)熟练使用 PostBackUrl 属性。

(2)熟练使用 IsCrossPageBack 属性。

(3)熟练使用 FindControl 方法。

3) 实现思路

- (1) 将“创建新班级”按钮的 PostBackUrl 属性设置为 ValidateClass.aspx 页面的 URL。
- (2) 通过 ValidateClass.aspx 页面的 IsCrossPageBack 属性判断是否为跨页请求。
- (3) 利用 FindControl 方法在 ValidateClass 页面获取 RegisterClass 页面的 TextBox 控件。

2. 为校友录创建班级留言页面

1) 任务说明

为校友录创建班级留言页面,如图 2-14 所示。要求留言发布的时间只显示页面首次加载的时间,即使一次提交多条留言,发表时间仍为初次加载时间。

2) 技能要点

- (1) 熟练使用 IsPostBack 属性。
- (2) 熟练使用 Cookie。

3) 实现思路

- (1) 使用 IsPostBack 属性。
- (2) 使用 Cookie 保存用户登录时间。

3. 为校友录创建后台管理登录页面

1) 任务说明

为校友录创建后台管理登录页面,如图 2-15 所示。用 Session 保存用户登录信息,然后跳转到后台管理页面,再在后台管理页面读取 Session 的值并进行验证,要求只有用户名为“admin”,密码为“12345”的用户才能登录(登录成功后显示如图 2-16 所示的界面),登录失败则返回登录页面。

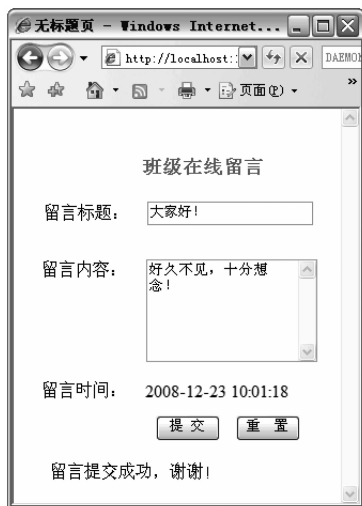


图 2-14 班级留言页面效果图



图 2-15 后台管理登录页面



图 2-16 登录成功后的显示页面

2) 技能要点

- (1) 熟练使用 IsPostBack 属性。
- (2) 熟练使用 Session 对象。

3) 实现思路

- (1) 利用 Session 对象保存用户信息。
- (2) 在后台管理页面读取 Session 的值,并进行验证。

构建网站前台页面模板

在前面的项目中,对 ASP.NET 的系统对象和状态管理有了基本的了解,从本项目开始,从建立“迅捷”网站前台母版页开始逐步搭建“迅捷”网站的整体架构。本项目的任务是完成网站前台页面母版页的制作,在母版页中实现网站导航及树型目录结构功能,并通过母版页创建内容页。

【工作任务】

- ④ 完成网站前台母版页的创建。
- ④ 实现网站导航功能,将网站导航加入母版页。
- ④ 实现网站的树型目录结构功能。

【技能目标】

- ④ 熟练创建母版页。
- ④ 熟练使用 SiteMapPath 控件实现网站的导航功能。
- ④ 熟练使用 TreeView 控件绑定 XML 数据源完成网站树型目录结构。

【常用术语】

母版页、站点地图、SiteMapPath 控件、TreeView 控件

3.1 任务1:创建网站前台母版页

3.1.1 任务描述

创建“迅捷”网站前台母版页,效果如图 3-1 所示。通过母版页创建内容页,实现网页风格一体化。



图 3-1 迅捷网站前台显示效果图

3.1.2 网页典型布局方式

在介绍网页页面的风格前,了解一下页面的布局。典型的页面布局有栏式结构和区域结构两种。

1. 栏式结构

栏式结构是常见的页面结构,其特点是简单实用、条理分明、格局清晰严谨,适合信息量大的页面。图 3-2 列出了几种常见的栏式结构。

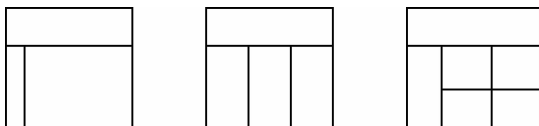


图 3-2 网页布局的栏式结构

2. 区域结构

区域结构使用得比较少,其特点是页面精美、主题突出、空间感很强,但仅适合信息量比较小的页面。图 3-3 是一个区域结构的样例,可以看到,页面被分成多个区域,有很多的空白区域作为背景。

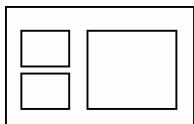


图 3-3 网页布局的区域结构

要实现页面布局,现在一般使用 DIV+CSS。早期曾大量使用 Table 作为布局方式,由于 DIV+CSS 方式具有代码简洁等优势,Table 布局方式逐渐被淘汰。

提示: Table 用于页面布局越来越少,但并不代表 Table 标签过时,在 DIV 布局的页面上,常常会有 Table 的表格。在许多项目中,往往使用 DIV 实现页面的整体布局,而页面中列表的显示常常还是使用 Table 标签。

3.1.3 母版页的制作

同一个站点的多个页面如果风格不一致,会给人不好的印象。ASP.NET 提供了一种比较简单的方法保持统一的风格,就是母版页。

1. 什么是母版页

母版页是一个以 .master 作为后缀名的文件,它可以将页面上的公用元素(如网站 Logo、广告条、导航条等)整合到一起。母版页对应内容页,两者的关系如图 3-4 所示。

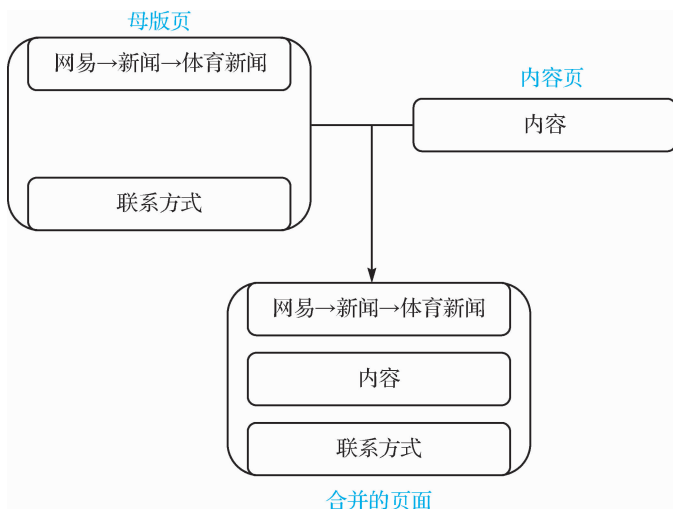


图 3-4 母版页与内容页的关系示例

2. 母版页的特点

使用母版页,可以为 ASP.NET 应用程序页面创建一个通用的外观。开发人员可在母版页中进行页面布局,然后在多个内容页面中应用该母版页布局。母版页具有以下 4 个优点。

(1)有利于站点修改和维护,减小开发人员的工作强度。由于公共内容被集中于母版页中,因此,只要修改母版页,就可以快速完成站点修改和维护任务,这在很大程度上提高了工作效率。

(2)具备高效的内容整合能力。在母版页中允许添加文字、控件等 Web 元素,同时也可以为这些 Web 元素添加相应的事件处理程序等。例如,只需要在母版页中添加一个服务器控件事件处理程序,站点内所有引用该母版页的网页都将获得同样的应用效果。

(3)有利于实现页面布局。内容页对应于母版页中的一处位置,这在很大程度上有利于页面的布局工作。

(4)提供一种便于利用的对象模型。由内容页和母版页组成的对象模型,能够为应用程序提供一种高效、易用的实现方式,并且这种对象模型的执行效率较以往的处理方式有了很大的提高。

3. 母版页的使用

1) 创建母版页

如图 3-5 所示,在“添加新项”对话框中选择母版页,本例取名为 MyMaster.master。



图 3-5 “添加新项”对话框

可以看到,母版页创建方式和新建页面方式是一样的,也有“将代码放在单独的文件中”复选框。由于很多时候并不需要在母版页中编写后台代码,所以取消选中此复选框。

新建的母版页上只有一个 ContentPlaceHolder 控件,它是预留给内容页显示信息的控件。可以在母版页上添加显示网站公共信息的控件,例如,在页头和页尾分别放置一个 Label 控件,如图 3-6 所示。



图 3-6 母版页

在实际开发中,美工人员常常提供静态页面文件,只需要将这些静态页面文件代码粘贴到母版页即可,注意至少保留一个 ContentPlaceHolder 控件。这里给 ContentPlaceHolder 控件设置的 ID 为 cphContent。

2) 创建内容页

添加内容页有两种方式:右击母版页中的 ContentPlaceHolder 控件,在弹出的快捷菜单中选择“添加内容页”命令(如图 3-7 所示),或者在创建页面时在“添加新项”对话框中选中“选择母版页”复选框(如图 3-8 所示),然后在弹出的“选择母版页”对话框中选择相应的母版页文件即可(如图 3-9 所示)。



图 3-7 添加内容页

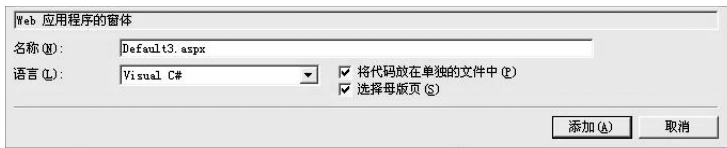


图 3-8 新建网页时选择母版页



图 3-9 “选择母版页”对话框

新添加的内容页在设计模式下可以看到母版页上的内容,如图 3-10 所示。内容页代码比较简单,代码如下。

```
<% @ Page Language = "C#" MasterPageFile = "~/MyMaster.master" AutoEven-
```

```
tWireup="true" CodeFile="Default.aspx.cs" Inherits="Default" Title="Untitled
Page" % >
    <asp:Content ID="Content1" ContentPlaceHolderID="cphContent" Runat="Serv-
er">
    </asp:Content>
```



图 3-10 新添加的内容页前台显示

它甚至没有 HTML 元素,下面解释一些代码的含义。

- MasterPageFile:用于指定所使用的母版页的路径。
- Title:用于设置内容页显示的标题。
- ContentPlaceHolderID:指定所对应的母版页中 ContentPlaceHolder 控件的 ID,如果指定的 ID 在母版页中不存在,将会发生错误。

3) 将现有的页面改为内容页

若已有一个 .aspx 文件,要将该文件改为某个母版的内容页,修改代码的步骤如下。

- (1) 在 @ Page 标记中指定 MasterPageFile 的位置。
- (2) 去除现有文件中的多余 HTML 标签。
- (3) 去除 Form 标记。
- (4) 创建 <asp:Content> 标签,设置 ID 属性和 ContentPlaceHolderID 属性。
- (5) 设置服务器端运行,添加 Runat="Server"。

4. 母版页和普通页的区别

母版页与普通页有些类似,区别如下。

1) 声明

普通页声明如下。

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

母版页声明如下。

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

2) 扩展名

普通页面文件后缀为.aspx,而母版页文件后缀为.master。

3) ContentPlaceHolder 控件

母版页可以使用一个或多个 ContentPlaceHolder 控件,而普通页不可以使用该控件。

3.1.4 网站前台母版页的实现

“迅捷”网站前台功能有我的购物车、用户登录、管理员界面、商品搜索。

(1)设计前台页面的布局。“迅捷”网站的页面布局根据用户需求设计,页面名称是 WebSite.html,布局如图 3-11 所示。



图 3-11 前台页面的布局

(2)建立前台母版页文件 WebSite.Master,将设计好的 WebSite.html 文件中的“<html>”和“</html>”之间的 HTML 代码复制到 WebSite.master 母版文件声明信息“<% @ Master Language="C#" AutoEventWireup="true" CodeFile="..." Inherits="..." %>”的后面,但需注意保留母版页中的 ContentPlaceHolder 控件。完成后的设计界面如图 3-11 所示。母版页不可以单独运行,可通过创建该母版的内容页进行浏览。

(3)创建该母版的内容页,右击母版页中的 ContentPlaceHolder 控件,在弹出的快捷菜单中选择“添加内容页”命令,将内容页命名为 WebSite.aspx。内容页可运行,母版页的设计效果通过运行内容页查看,内容页的效果如图 3-1 所示。

3.2 任务 2:制作网站导航

3.2.1 任务描述

在每一页上面显示导航条,让网页浏览者能清楚地了解自己所处的位置,如图 3-12 所示,显示“您现在的位置:迅捷”。



图 3-12 迅捷网站前台导航效果图

此网站导航效果的实现是由站点地图 Web. sitemap 文件和导航控件 SiteMapPath 联合实现的。站点地图 Web. sitemap 文件包含整个网站各个页面间的层次关系,例如,“首页-新闻-国内”;SiteMapPath 控件用于将站点地图文件中设置的层次关系显示出来,其显示形式是“一级页面(分隔符)二级页面(分隔符)……”,其中分隔符可以通过 SiteMapPath 控件的 PathSeparator 属性设置。

3.2.2 站点地图

站点地图是描述站点逻辑结构的文件,该文件实际上是一个 XML 结构的文件。在网站中添加和删除页面的时候,开发人员只要更改站点地图文件就可以管理页面的导航,而不需要修改各个页面本身的导航链接。

站点地图文件名必须是 Web. sitemap 文件,而且需要放置于应用程序的根目录。它是一个 XML 文件,节点的描述如下。

- 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 标签没有任何联系,虽然它们的值可以相同)。
- url:文件在解决方案的位置。
- description:说明性文本。

下面结合具体的网站结构说明网站站点地图的创建,在图 3-13 中列出了一个示例网站的树型结构图。

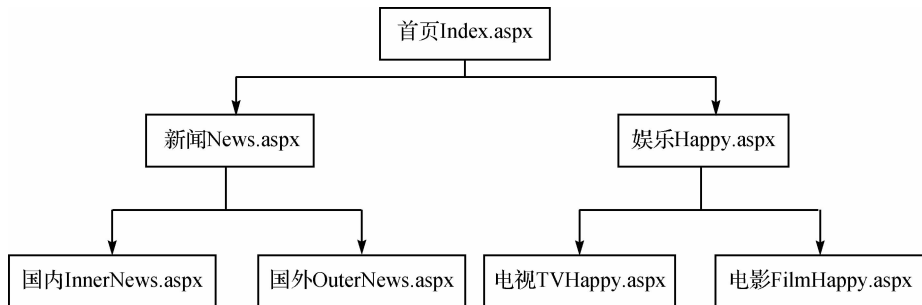


图 3-13 示例网站架构

可以看到,该网站定义了三级目录,网站的首页是 Index.aspx,是网站的根节点。在根节点下,分别定义了两个二级页面节点,分别是新闻 News.aspx 和娱乐 Happy.aspx。在新闻节点下包括了国内新闻和国外新闻两个节点,在娱乐节点下包括了电视娱乐和电影娱乐两个节点。要实现该网站架构的站点地图文件,首先在站点的根目录下创建站点地图文件。

在“项目”菜单中,选择“添加新项”命令,打开“添加新项”对话框,如图 3-14 所示。



图 3-14 创建站点地图文件

站点地图文件中的具体代码如下。

```
<? 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="网站首页">
    <siteMapNode url="~/News/news.aspx"title="新闻首页"description="新闻首页">
      <siteMapNode url="~/News/InnerNews.aspx"title="国内新闻"/>
      <siteMapNode url="~/OuterNews.aspx"title="国外新闻"/>
    </siteMapNode>
    <siteMapNode url="~/Happy/Happy.aspx"title="娱乐首页"description="娱乐首页">
      <siteMapNode url="~/Happy/TVHappy.aspx"title="电视娱乐"/>
      <siteMapNode url="~/Happy/FilmHappy.aspx"title="电影娱乐"/>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

上述 XML 代码实现了图 3-13 所示的站点结构。将该文件保存为 Web.sitemap 文件即可。编写站点地图的注意事项如下。



- (1) 站点地图的根节点为<sitemap>元素,一个网站地图中只能有一个根节点。
 - (2) 在<sitemap>下一级有且只有一个<siteMapNode>节点。
 - (3) <siteMapNode>节点是具体描述网站导航页面的节点,<siteMapNode>节点下面可以包含多个新的<siteMapNode>节点,从而完成整个网站的结构定义。
 - (4) 在站点地图中,同一个 URL 仅能出现一次。
- 提示:** 站点地图文件指定的页面关系是逻辑关系,而不是存储位置间的关系,所以具有很大的灵活性。

3.2.3 SiteMapPath 控件

1. 概述

SiteMapPath 控件可以为站点提供“面包屑导航”的功能,用于显示从站点首页(根节点)到当前页面之间的路径。如图 3-15 所示,此控件可以为每一个非末级页面提供链接,供用户快速定位到当前页面的任意上级或顶级页面。

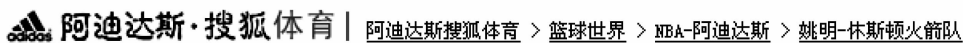


图 3-15 建立站点地图示例

SiteMapPath 控件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,首先要有站点地图文件。

2. SiteMapPath 控件的常用属性

SiteMapPath 控件的常用属性见表 3-1。

表 3-1 SiteMapPath 的常用属性

属性名称	说 明
CssClass	获取或设置由 Web 服务器控件在客户端呈现的级联样式表 (CSS) 类
PathDirection	获取或设置导航路径节点的呈现顺序
PathSeparator	获取或设置一个字符串,该字符串在呈现的导航路径中分隔 SiteMapPath 节点

3. SiteMapPath 的应用

用 SiteMapPath 控件实现图 3-13 所示的网站结构的页面导航,在该站点中建立页面文件 Index.aspx,根据站点地图文件的路径要求,分别建立 News 和 Happy 文件夹,在 News 文件夹内添加 News.aspx、InnerNews.aspx 和 OuterNews.aspx 文件;在 Happy 文件夹内添加 Happy.aspx、TVHappy.aspx、FilmHappy.aspx 文件,并在以上建立的页面中拖入 SiteMapPath 控件。其中,TVHappy.aspx 页面的导航效果如图 3-16 所示。

首页 > 娱乐首页 > 电视娱乐

图 3-16 TVHappy.aspx 页面的导航效果

提示: (1) SiteMapPath 的 PathSeparator 属性用于设置 SiteMapPath 的分隔符显示样式, 这里是用“>”分隔的, 如要用“-”分隔, 可做如下设置: <asp: SiteMapPath ID = "SiteMapPath1" runat = "server" PathSeparator = "-">。

(2) PageLevelsDisplayed 属性用于控制导航显示的级数, 如果导航过深, 影响美观, 可通过该属性控制级数。

(3) Web. sitemap 中必须包含当前页面的 URL, 否则当前页的 SiteMapPath 不会被正确显示。

3.2.4 网站前台母版页导航的实现

“迅捷”网站的前台页面结构如图 3-17 所示。

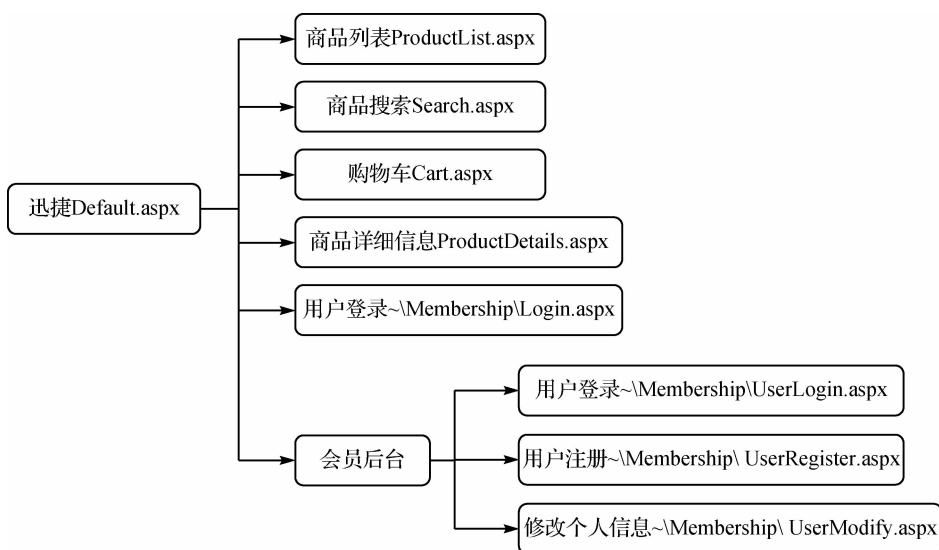


图 3-17 “迅捷”网站的前台页面结构图

(1) 根据结构图建立网站的站点地图 Web. sitemap。

站点地图文件的代码如下。

```

<? xml version="1.0" encoding="utf-8"? >
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="~\Default.aspx" title="迅捷" description="">
        <siteMapNode url="ProductList.aspx" title="商品列表页" description=
""/>
        <siteMapNode url="Search.aspx" title="搜索页" description=""/>
        <siteMapNode url="Cart.aspx" title="购物车" description=""/>
    
```

```

        <siteMapNode url = "~\ProductDetails.aspx" title = "商品详细页"
description="" />
        <siteMapNode url = "~\Membership\Login.aspx" title = "用户登录"
description="" />
        <siteMapNode Id="" url="" title = "会员后台" description="">
        <siteMapNode url = "~\Membership\UserRegister.aspx" title = "用户注
册" />
        <siteMapNode url = "~\Membership\UserLogin.aspx" title = "用户登录" />
        <siteMapNode url = "~\Membership\UserModify.aspx" title = "修改个人信
息" />
    </siteMapNode>
</siteMapNode>
</siteMap>

```

(2) 在母版页 Website.master 中添加导航控件 SiteMapPath。

拖动 SiteMapPath 至母版页的相应位置, 各页导航自动完成。HTML 视图中的代码如下。

```

<div style = "padding-left: 30px; font-size: 13px; line-height: 30px; font-
weight: bold;">
    <table border = "0" cellpadding = "5" cellspacing = "0" width = "90%">
    <tr>
        <td style = "text-align: left;">
            您现在的位置:<asp:SiteMapPath ID = "SiteMapPath1" runat = "server">
        </asp:SiteMapPath>
        </td>
        <td style = "text-align: right;">
            <asp:Label ID = "lblUserInfo" runat = "server" Text = ""></asp:Label>
        </td>
    </tr>
    </table>
</div>

```

3.3

任务 3: 制作网站树型目录结构

3.3.1 任务描述

实现母版页左侧的树型目录结构导航菜单, 效果如图 3-18 所示。



图 3-18 “迅捷”网站前台树型目录结构导航菜单效果图

3.3.2 TreeView 导航控件

1. 概述

TreeView 服务器控件主要以树型结构的形式展现分层数据,如站点的导航结构或者文件的目录结构等。TreeView 控件主要支持的功能有:数据自动绑定、站点导航、显示数据信息或者超链接、通过编程动态加载树型结构、添加附加的控件(如复选框控件)实现对节点的选择。

TreeView 控件由一个或者多个节点构成,树中的每一项都是一个节点。根据节点位置的不同,可以分成根节点、父节点和叶节点三种。根节点是树的顶级节点,它没有父节点;叶节点是底层的节点,它没有子节点。根节点和叶节点之外的节点都可以称为父节点。每一个节点对象都有 Text 和 Value 属性,Text 属性可用来显示节点的名称,而 Value 属性用来存放节点的其他任何附加信息。

TreeView 控件的数据显示有三个方法。

- (1)通过静态数据来填充。
- (2)通过将 TreeView 控件与数据源控件进行绑定动态地获取数据。
- (3)通过 TreeView 控件事件驱动模型,根据用户的选择动态地加载树上的节点信息。

本节主要介绍第二种方法即与数据源的绑定来获取数据,根据数据源类型的不同,可以将 TreeView 的使用方式分为两种:使用站点地图文件作为数据源和使用 XML 文件作为数据源。

TreeView 控件的常用属性见表 3-2。



表 3-2 TreeView 控件的常用属性

属性名称	说 明
CollapseImageUrl	获取或设置自定义图像的 URL,该图像用作可折叠节点的指示符
ExpandDepth	获取或设置第一次显示 TreeView 控件时所展开的级数
ImageSet	获取或设置用于 TreeView 控件的图像组
NoExpandImageUrl	获取或设置自定义图像的 URL,该图像用作不可展开节点的指示符
PathSeparator	获取或设置用于分隔由 ValuePath 属性指定的节点值的字符
Target	获取或设置要在其中显示与节点相关联的网页内容的目标窗口或框架

2. 使用站点地图文件作为 TreeView 控件的数据源

可以用站点地图文件作为数据源,由于刚才已经创建了“迅捷”网站的站点地图和不少页面文件,这里就新建页面 WebForm1.aspx,并拖入 TreeView 控件,右击该控件,在弹出的快捷菜单中选择“显示智能标记”命令,弹出“TreeView 任务”智能提示信息,如图 3-19 所示。



图 3-19 “TreeView 任务”对话框

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,可以看到 TreeView 控件可以使用两种形式的数据源。

此时选择“站点地图”数据源,并为数据源指定 ID,如图 3-20 所示。

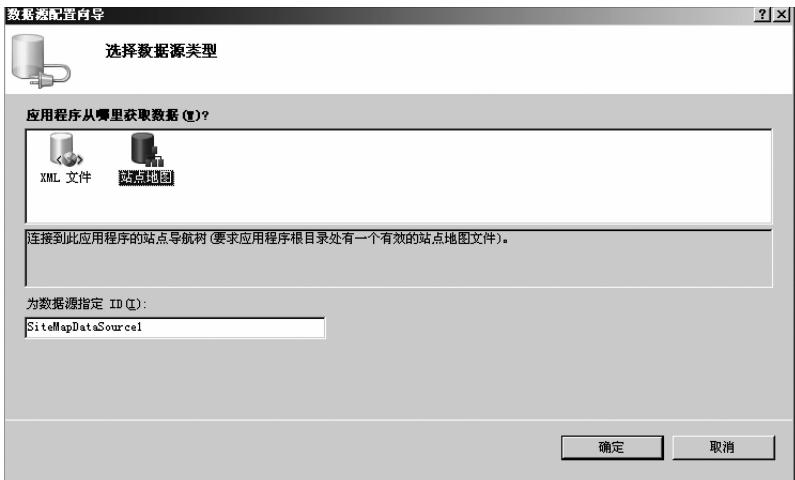


图 3-20 “数据源配置向导”对话框

单击“确定”按钮,树型目录导航就可以实现了,如图 3-21 所示。

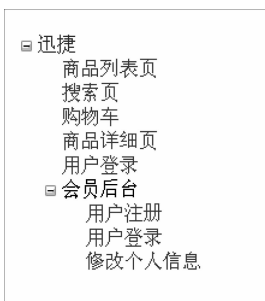


图 3-21 “迅捷”网站的树型目录

3. 使用 XML 文件作为 TreeView 控件的数据源

刚才使用了站点地图文件作为 TreeView 控件的数据源,但由于一般的树型目录不会把所有页面列出来,在这种情况下,应使用 XML 文件作为数据源,显示用户所需的导航信息。下面通过一个例子说明 TreeView 控件与 XML 文件的数据绑定。如图 3-22 所示是要显示的树型目录。

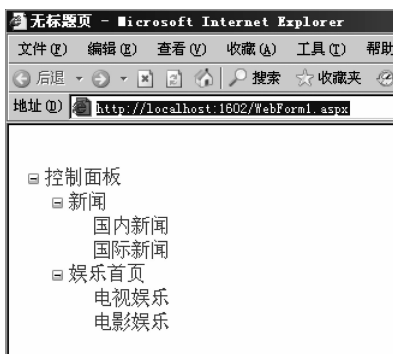


图 3-22 树型目录

其实现步骤如下。

(1) 编写 XML 文件。首先需要编写一个 XML 文件——TreeViewXmlSourceDemo.xml,该 XML 文件的格式可参考站点地图文件的内容。与站点地图相比,这里需要的 XML 文件没有过多的限制条件,从站点地图中复制部分代码即可。具体代码如下。

```
<? xml version="1.0"encoding="utf-8"? >
<siteRoot Id="root"url=""title="控制面板"description="">
  <siteMapNode url="~/News/news.aspx"title="新闻"description="">
    <siteMapNode url="~/News/InnerNews.aspx"title="国内新闻"description=
"/>
    <siteMapNode url="~/OuterNews.aspx"title="国际新闻"description=
"/>
  </siteMapNode>
  <siteMapNode url="~/Happy/Happy.aspx"title="娱乐首页"description=
"/>
```



```

<siteMapNode url = "~/Happy/TVHappy.aspx" title = "电视娱乐" description =
""/>
<siteMapNode url = "~/Happy/FilmHappy.aspx" title = "电影娱乐" descrip-
tion = ""/>
</siteMapNode>
</siteRoot>

```

(2) 建立新的页面文件, 命名为 TreeViewXmlSourceDemo.aspx, 进入该页的设计视图, 在页面中添加一个 TreeView 控件。

(3) 设置 XML 文件数据源, 这一步和使用站点地图文件作为数据源的方式一样, 只是要选择“XML 文件”作为数据源即可。单击“确定”按钮后, 弹出“配置数据源”对话框, 如图 3-23 所示。

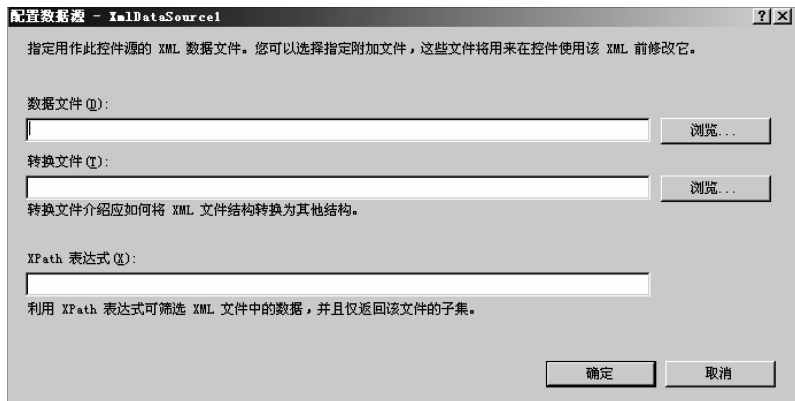


图 3-23 “配置数据源”对话框

其中, “数据文件”文本框用于设置 XML 文件的路径, 可单击后面的“浏览”按钮, 选择需要的 XML 文件。设置完成后单击“确定”按钮, 显示如图 3-24 所示的结果。

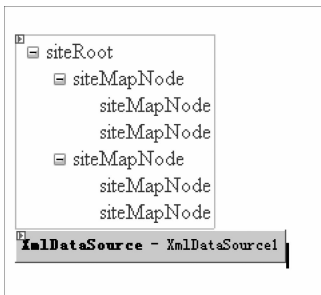


图 3-24 配置数据源后的显示结果

(4) 编辑数据的绑定。很显然, TreeView 控件并未识别用户的意图, 用户希望用 siteMapNode 元素的 title 属性作为显示字段, url 属性作为链接, 这就需要做一些设置。

如图 3-25 所示, 单击智能提示信息中的“编辑 TreeNode 数据绑定”, 打开“TreeView Data-Bindings 编辑器”对话框。



图 3-25 编辑 TreeNode 数据绑定

在“TreeView DataBindings 编辑器”对话框中,添加要绑定的节点,然后在右侧设置绑定的元素,如图 3-26 所示。

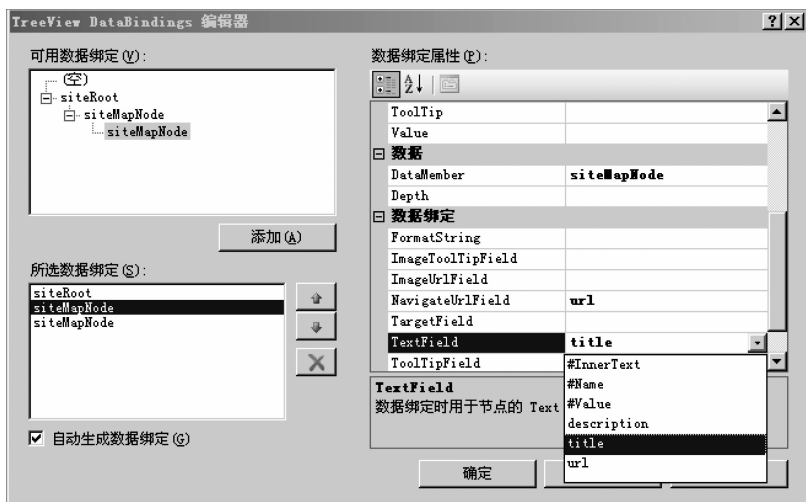


图 3-26 “TreeView DataBindings 编辑器”对话框

其中常用的数据绑定属性见表 3-3。

表 3-3 TreeView 控件常用的数据绑定属性

属 性	说 明
TextField	设置显示的文字所绑定的字段或元素
NavigateUrlField	设置链接对应的字段或元素值

(5) 设置完成后运行,效果如图 3-22 所示。

3.3.3 网站前台母版页树型目录结构的实现

1. 创建 XML 文件

“迅捷”网站前台的树型目录结构如图 3-27 所示。

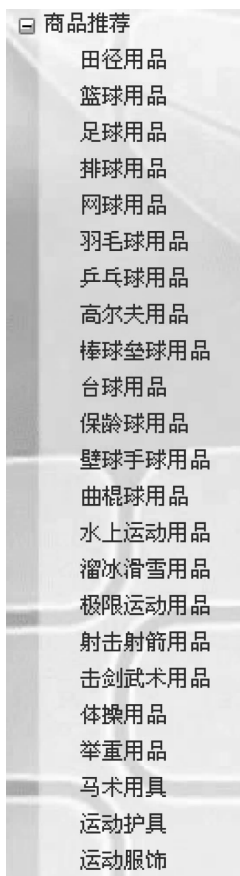


图 3-27 “迅捷”网站前台树型目录结构

要实现该树型目录,首先应新建一个 XML 文件(TreeView.xml),该文件的代码如下。

```
<? xml version="1.0"encoding="utf-8"? >
<siteMapNode url="default.aspx" title="商品推荐" description="">
  <siteMapNode title="田径用品" url="ProductList.aspx? typeid=48"/>
  <siteMapNode title="篮球用品" url="ProductList.aspx? typeid=47"/>
  <siteMapNode title="足球用品" url="ProductList.aspx? typeid=46"/>
  <siteMapNode title="排球用品" url="ProductList.aspx? typeid=38"/>
  <siteMapNode title="网球用品" url="ProductList.aspx? typeid=35"/>
  <siteMapNode title="羽毛球用品" url="ProductList.aspx? typeid=31"/>
  <siteMapNode title="乒乓球用品" url="ProductList.aspx? typeid=29"/>
  <siteMapNode title="高尔夫用品" url="ProductList.aspx? typeid=28"/>
  <siteMapNode title="棒球垒球用品" url="ProductList.aspx? typeid=27"/>
  <siteMapNode title="台球用品" url="ProductList.aspx? typeid=26"/>
  <siteMapNode title="保龄球用品" url="ProductList.aspx? typeid=25"/>
  <siteMapNode title="壁球手球用品" url="ProductList.aspx? typeid=22"/>
  <siteMapNode title="曲棍球用品" url="ProductList.aspx? typeid=21"/>
```

```
<siteMapNode title="水上运动用品"url="ProductList.aspx? typeid=20"/>
<siteMapNode title="溜冰滑雪用品"url="ProductList.aspx? typeid=17"/>
<siteMapNode title="极限运动用品"url="ProductList.aspx? typeid=16"/>
<siteMapNode title="射击射箭用品"url="ProductList.aspx? typeid=15"/>
<siteMapNode title="击剑、武术用品"url="ProductList.aspx? typeid=6"/>
<siteMapNode title="体操用品"url="ProductList.aspx? typeid=5"/>
<siteMapNode title="举重用品"url="ProductList.aspx? typeid=4"/>
<siteMapNode title="马术用具"url="ProductList.aspx? typeid=3"/>
<siteMapNode title="运动护具"url="ProductList.aspx? typeid=2"/>
<siteMapNode title="运动服饰"url="ProductList.aspx? typeid=1"/>
</siteMapNode>
```

2. 在母版中添加树型导航控件,并设置数据源

为了使“迅捷”网站前台各页面风格一致,即前台各页面的左侧都是树型目录导航,需在前台母版中添加树型导航控件,打开前面创建的母版页文件 Website.master,在左侧添加 TreeView 控件,设置该控件的数据源为 TreeView.xml,方法同前。

3. 编辑数据的绑定

添加要绑定的节点,然后设置节点的 TextField 属性为“title”,NavigateUrlField 的属性为“url”。

4. 添加内容页

新建内容页,选择母版为 Website.master,运行效果如图 3-28 所示。



图 3-28 “迅捷”网站前台树型目录结构显示效果



3.4 项目小结

本项目介绍了在 ASP.NET 中如何使用母版页建立网站统一风格,并完成了网站前台的制作,内容如下。

- 母版页提供一个页面框架,而内容页提供页面内容。
- 母版页可以放置导航控件,需要说明的是,SiteMapPath 控件可根据内容页的路径显示当前的路径,而不是依据母版页。
- 在站点地图中写明某页面的路径,该页面才能显示 SiteMapPath 导航。
- SiteMapPath 控件只能绑定站点地图,而 TreeView 控件还可以绑定 XML 文件。

3.5 思考与练习

1. 有关母版页的说法,正确的是()。

- A. 一个站点只能有一个母版页
- B. 一个内容页对应母版页中的一个位置 ContentPlaceHolder
- C. 内容页相当于 HTML 中的 iframe 页,浏览地址显示母版页地址
- D. 母版页文件的后缀是.master

2. 下面是.aspx 页面中的一段代码:

```
<% @ Page Language="C#" MasterPageFile="~/Admin/Admin.Master" Auto-  
EventWireup="true" Codebehind="Default.aspx.cs" Inherits="Web.Admin.Default"  
Title="迅捷|后台管理中心"%>
```

关于“~/Admin/Admin.Master”,下面说法正确的是()。

- A. 母版页在应用程序中的根目录
- B. 母版页在当前目录
- C. 站点地图在应用程序中的根目录
- D. 站点地图在当前目录

3. 有关导航控件的说法,正确的是()。

- A. 作为服务器控件,导航控件将生成浏览器可执行的 JavaScript 脚本和 HTML 代码
- B. SiteMapPath 控件只要拖入当前页中就可以发挥作用
- C. SiteMapPath 控件可以使用后缀为.xml 的文件作为数据源
- D. TreeView 控件所使用的数据源一定是后缀为.xml 的文件

4. 下面是 TreeView 控件的一段代码:

```
<asp:TreeNode Text="节点 1" SelectAction="Expand">  
    <asp:TreeNode Text="节点 2"...>  
        <asp:TreeNode Text="节点 3"...>  
</asp:TreeNode>
```

当单击“节点 1”时只进行()的操作。

- A. 调用网页
- B. 显示或隐藏本节点
- C. 折叠或展开子节点
- D. 恢复原状

5. 关于 Web 服务器控件,说法正确的是()。(选择两项)

- A. 运行时输出客户端的是 HTML 标签
- B. 运行时输出客户端的是微软特定内容,因为这样才可以在服务器识别
- C. 所有支持 HTML 标准的浏览器都可以识别服务器控件产生的内容
- D. 浏览器需支持微软标准才能识别服务器控件

3.6 项目实战

完成后台母版页的创建及应用,包括后台网站导航及树型目录结构。

1. 任务说明

在前面,网站前台母版页已经搭建好了,本项目实战将搭建一个后台母版页面,在后台母版页中实现与前台页面基本一致的功能,效果如图 3-29 所示。

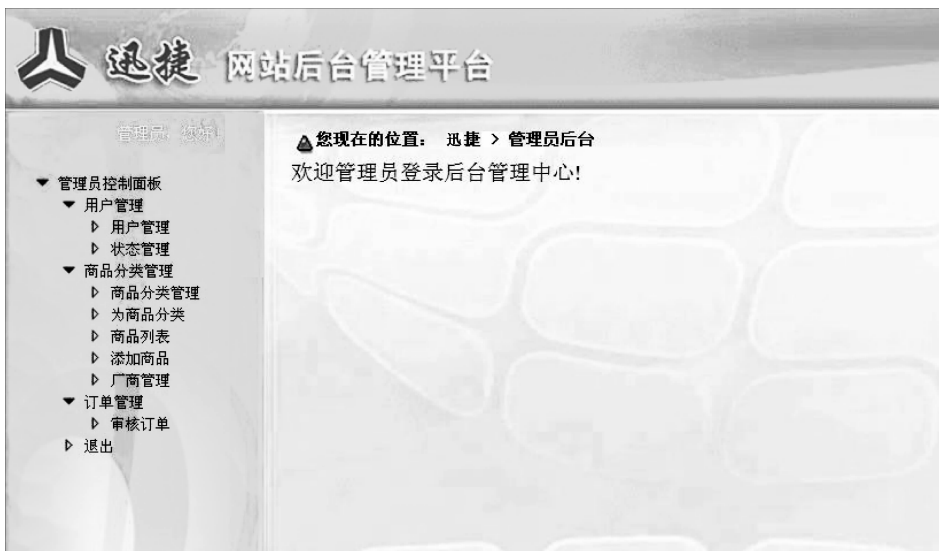


图 3-29 “迅捷”网站后台首页

2. 技能要点

- (1) 母版页的创建及使用。
- (2) 熟练掌握 SiteMapPath 控件的使用。
- (3) 熟练掌握 TreeView 控件的使用。



3. 实现思路

- (1) 创建母版页 Admin.master, 根据美工设计, 做好后台网站框架结构。
- (2) 加入 SiteMapPath 控件到相应位置。
- (3) 建立 XML 文件 admin_menu.xml, 为左侧树型结构提供数据。
- (4) 从工具箱中将 XmlDataSource 数据源拖入 Admin.master 中, 设置其 ID 为 xdsAdmin, DataFile 属性设置为“~/Admin/admin_menu.xml”。
- (5) 将 TreeView 控件拖入相应位置, 选择其数据源为 xdsAdmin。
- (6) 在相应位置加入 ContentPlaceHolder 控件。