“上海紧缺人才培训工程”教学系列丛书

ASP 使用基础

上海市计算机应用能力考核办公室　编

·上　海·

“上海紧缺人才培训工程”计算机应用能力教学系列丛书,由上海市教育委员会、上海市成人教育委员会、中共上海市委组织部、上海市人事局、上海市国民经济和社会信息化领导小组办公室联合组织编写。

上海市计算机应用能力考核专家组成员

组长：施伯乐　复旦大学教授
组员：白英彩　上海交通大学教授
　　郑衍衡　上海大学教授
　　汪燮华　华东师范大学教授
　　俞时权　上海师范大学教授
　　高毓乾　上海市科委高级工程师
　　陶　霖　上海第二工业大学教授
　　许永兴　上海电视大学教授

本书编撰人员
编　者：张博锋　王黎阳
审稿者：许永兴

致 读 者

中华人民共和国教育部部长　陈至立

高科技及其产业是当代经济发展的火车头。在当代科学技术革命中，计算机信息处理技术居于先导地位。在90年代的今天，世界科学技术已经进入了信息革命的新纪元。

上海的振兴正处于这一信息革命的时代。上海要在本世纪末、下世纪初跻身国际经济、金融、贸易中心城市之列，就必须牢牢把握机遇，大力发展计算机应用及其产业。市委、市政府决定尽快发展计算机产业，使其成为上海新一代的支柱产业。这是从上海产业结构调整、城市功能发挥、技术革命发展的战略高度出发作出的战略决策。今后几年，上海计算机产业的销售额将每年翻一番，到本世纪末形成年销售额达数百亿元的产业规模。金融电子化、商业电子化、个人用电脑的普及、机电一体化、城市管理、工业管理以及办公自动化、智能化大楼的建设、软件开发应用及系统集成等，将使上海的经济和社会生活发生深刻的变化，并为上海成为国际经济、金融、贸易中心城市提供必不可少的技术支撑。计算机产业不仅将成为上海工业发展的新的生长点，并将带动一批相关产业的发展。可以预计，不久的将来，计算机在上海将被广泛应用，渗透到各行各业，使上海的现代化水平向前迈进一大步。

发展计算机产业对计算机专业人才的培养及应用人才的培训提出了紧迫要求，一方面要培养一大批能够从事计算机研究开发的高级专业人才，另一方面要培训成千上万的计算机操作人员，普及计算机应用技术。只有各行各业的从业人员都学会计算机操作和应用，计算机的广泛使用和产业发展才能真正实现。因此，上海市“90年代紧缺人才培训工程”和上海市“三学”（学知识、学科学、学技术）活动都把计算机应用技术的普及作为其重要内容。上海市计算机应用能力考核则是在广大市民中普及计算机应用技术的一项重要举措。这项考核的独创性和实用性使其独具特点，受到应考者及用人单位的广泛欢迎。

希望上海广大市民顺应新技术革命的潮流，努力掌握计算机应用技术，为上海的振兴作出更大贡献！

1994年7月

注：本文发表时，作者任中共上海市委副书记、上海市计算机应用与产业发展领导小组组长。

序

中共上海市委副书记　龚学平

“90 年代上海紧缺人才培训工程”实施三年来，取得了较大的成绩。这一成绩表现在下列诸多方面：一、以系统或行业为依托，建立了以十大紧缺人才培训中心为主体的紧缺人才培训体系，分别承担现代企业高级经理、现代企业高级营销经理、房地产开发、涉外商务、涉外法律等 26 类岗位的紧缺人才培训考核工作。二、建立了计算机应用能力考核制和通用外语水平等级考试制，参加计算机应用能力考核的有 93 万人，经考核合格的有近 59 万人；参加通用外语水平等级考试的达 13 万人，经考试合格的有 8.4 万人，较好地提高了市民计算机应用能力和外语水平。三、建立了上海教育电视台，在交流教育信息、传播科学知识、弘扬优秀文化、提高市民素质等方面发挥了积极的作用。

“90 年代上海紧缺人才培训工程”进展顺利的原因是多方面的，其中最根本的是，它顺应了上海经济建设和社会发展的需要。具体地说，它的成功有赖于市委、市府的正确领导，有赖于这一培训工程的组织者——市教委、成人教委、市委组织部和市人事局的通力协作，有赖于中央和市有关部门的支持，有赖于从事这一工程的全体同志坚持不懈的努力。这里值得一提的是，这一培训工程的教学系列丛书从内容到形式，具有实用性强、应变性强、适用面宽的特点，与以往教材相比体现了“紧缺”之意，它是本市许多专家、学者与实际工作者共同心血的结晶。现在，其中的某些教材已经出新版本了，表明它们在“紧缺”方面有更进一步的追求。

从现在到 2010 年，是建设有中国特色社会主义承前启后、继往开来的重要时期。上海要努力建设成为国际经济、金融和贸易中心城市之一。在机遇与挑战并存的形势下，继续努力搞好“90 年代上海紧缺人才培训工程”，培养一大批社会主义现代化建设的急需人才，必将对上海的腾飞产生巨大的现实意义与深远的历史意义。

上海的改革和发展为我们提供了实施“90 年代上海紧缺人才培训工程”的广阔舞台。市各有关方面一定要进一步加强领导，团结协作，深化改革，扎实工作，努力在这个舞台上大显身手。我们也期待着更多的优秀教材面世，推进这一培训工程的进一步发展，为迎接 21 世纪的到来作出更大的贡献。

1997 年 4 月

序

上海市政协副主席　谢丽娟

由上海市人民政府教育卫生办公室、市成人教育委员会、中共上海市委组织部、市人事局联合组织编写的“90年代上海紧缺人才培训工程”教学系列丛书将陆续出版。编写、出版这套丛书是实施上海紧缺人才培训工程的基础工作之一，对推动培养和造就适应上海经济建设和社会发展急需的专业技术人才必将起到积极的作用。

90年代是振兴上海、开发浦东关键的十年。上海要成为国际经济、金融、贸易中心之一，成为长江流域经济发展的“龙头”，很大程度上取决于上海能否有效地提高上海人的整体素质，能否培养和造就出一大批坚持为上海经济建设和社会发展服务，既懂经济，懂法律，懂外语，又善于经济管理，擅长国际竞争，适应社会主义市场经济新秩序的多层次专业人才。这已越来越成为广大上海人民的共同认识。

目前上海人才的状况与经济建设、社会发展的需求矛盾日趋显著。它集中表现在：社会主义市场经济的逐步确立，外向型经济的迅速发展，新兴产业的不断崛起，产业产品结构的适时调整，城市建设和管理任务的日益繁重，使原来习惯于在计划经济体制下工作的各类专业技术人才进入了一个颇感生疏的境地，使原来以面向国内市场为主的各类专业技术人才进入一个同时面向国内外市场并参与国际竞争的新天地，金融、旅游、房地产、城市建设和管理等以及许多高新技术产业又急切地呼唤一大批新的专业技术人才。这就加剧了本市专业人才总量不足、结构不合理的矛盾。此外，本市的从业人员和市民的外语水平与计算机的应用能力普遍不高。这种情况如不迅速改变，必将影响上海的经济走向世界，必将影响上海在国际经济、金融、贸易中的地位和在长江流域乃至我国经济发展中的作用。紧缺人才培训问题已引起市委、市政府的高度重视。

“机不可失，时不再来。”我们要大力加强紧缺人才的培训工作和外语、计算机的推广普及工作。鉴于此，及时编写、出版本丛书，是当前形势之急需，其意义是现实的和深远的。诚然，要全面组织实施90年代上海紧缺人才培训工程，还有待于各有关方面的共同努力。

在“90年代紧缺人才培训工程”教学系列丛书开始出版之际，感触颇多，简述代序。

1993年8月

编 者 的 话

本书是 Active Server Pages (ASP)的使用基础教程,由上海市计算机应用能力考核办公室编,为“上海紧缺人才培训工程”教学系列丛书之一。在新推出的上海市信息技术认证证书考核中,要获取网页网站开发工程师认证证书,本书是其中的一门课程。通过本书的学习和实践,读者可以掌握 ASP 的基础知识和有关动态网站建设的基本内容。本书适合具有计算机初级应用能力的读者学习,适用于网站开发者制作动态、交互、高效的网站。

当今的世界是网络的世界,网络逐渐成为人们日常生活中重要的一部分。随着 Internet 功能的日益强大,用户不仅可以从 Internet 上获取新闻和邮件,还可以进行网上购物、网上聊天、网上点播等等。可以说,网络在 21 世纪将无处不在,并且会越来越完善。作为跨世纪的人才,当然不能徘徊在网络殿堂之外,而应该深入地了解网络的奥妙之处。

随着 Web 应用的发展,用户希望能够看到根据要求而动态生成的主页,例如响应用户数据库查询、动态生成报表等。ASP 是一种由 Microsoft 公司开发的服务器端的脚本语言运行环境。它可以结合 HTML 语言和 ActiveX 组件建立动态、交互、高效的 Web 服务器端应用程序。

所谓‘动态’网站,并不是指在网页上显示几个 GIF 动态图片,而是指应用程序会根据用户的要求和选择作出动态响应,不用修改 ASP 程序,便会自动生成新的页面,这样不同的人在不同的时间访问同一网址时,会产生不同的页面,因此可大大节省网站维护的工作量,并能很好地体现网站与客户端用户的交互性。

ASP 属于 ActiveX 技术中的服务器端技术,与常见的在客户端实现动态主页的技术,如 Java Applet、ActiveX Control、VBScript、JavaScript 等不同。客户端技术的 Script 语句是由浏览器解释执行的,而 ASP 中的 Script 语句是由服务器端解释执行的,执行结果产生动态生成的 Web 页面并送到浏览器,从而减轻了客户端浏览器的负担,提高了网站浏览的效率。这就是 ASP 网页的高效性。

由于 ASP 在服务器端解释执行,开发者可以不必考虑浏览器是否支持 ASP ;同时由于它在服务器端执行,在客户端的浏览器上看不到 ASP 源代码,看到的仅仅是执行后的结果,开发者也不必担心别人下载程序代码,增加了网站的安全性。

本书共分十章,主要内容有 ASP 简介,HTML 文档设计,脚本语言 VBScript,数据动态变

换，HTML 表单的处理，Application、Server 和 Session 对象，ASP 组件应用，ASP 与数据库，制作实例等。第一章简要介绍了 ASP 的运行环境和基本组成；HTML 语言是 ASP 的基础，因而第二章简述了 HTML 的基本用法；ASP 中使用的脚本语言为最常用的 VBScript，因此第三章着重讲解了 VBScript 的用法；第四章介绍了 ASP 的基础知识以及 SSI 和脚本语言的使用方法；ASP 非常注重客户端和服务器端之间的交互，因而第五章主要讲解了利用 Request、Response 对象实现交互的方法；第六章对表单的处理进行了详细的论述；第七章较为全面地说明了三大对象 Application、Server 和 Session 的用法；为了使网站的功能变得更强大，还需要使用其他组件以及第三方组件，有关这方面的用法在第八章中进行了详尽的解释；网站的建立离不开数据库的支持，因而第九章阐述了有关数据库在网站中的使用方法；第十章给出了两个动态网页的制作实例。另外，为了便于读者学习，在附录中列出了 VBScript 中的常用函数。

本书在每章后均附有习题，读者可以通过这些习题巩固所学到的基本知识和操作方法。

由于本书编写时间仓促，书中难免出现疏漏，祈请各位同行、专家和读者指正。

上海市计算机应用能力考核办公室

2001 年 5 月

内 容 提 要

本书是 Active Server Pages (ASP)的使用基础教程,由上海市计算机应用能力考核办公室编,为“上海紧缺人才培训工程”教学系列丛书之一。

本书共分十章,主要内容有 ASP 简介,HTML 文档设计,脚本语言 VBScript,数据动态交换,HTML 表单的处理,Application、Server 和 Session 对象,ASP 组件应用,ASP 与数据库,制作实例等。各章后均配有习题,读者可以通过这些习题巩固所学到的基本知识和操作方法。本书通过一些简单易懂的例子进行讲解,使读者能较快地掌握这些内容。第十章给出了两个动态网页的制作实例,附录列出了 VBScript 中的常用函数,可供读者参考。

本书可作为具有计算机初级应用能力的读者进行网页设计的入门教材,也可作为广大动态网页设计者的参考书。

图书在版编目(CIP)数据

ASP 使用基础/张博锋主编. —上海:上海大学出版社,2001.5
ISBN 7-81058-329-8

Ⅰ.A… Ⅱ.张… Ⅲ.主页制作-应用软件,ASP Ⅳ.TP393.092

中国版本图书馆 CIP 数据核字(2001)第 26102 号

责任编辑/李顺祺
封面设计/王春杰

上海大学出版社出版发行
(上海市延长路 149 号 邮政编码:200072)
江苏句容市排印厂印刷 各地新华书店经销
开本 787×1092 1/16 印张 15.25 字数 350 000
2001 年 6 月第 1 版 2001 年 6 月第 1 次印刷
印数 1~5 100
定价:27.00 元

目 录

第一章 ASP 简 介

1.1 ASP 概述

目前,在互联网网站中,有相当一部分网页是‘静态’的。所谓‘静态’指的就是网站的网页内容固定不变,当用户浏览器通过互联网的 HTTP (Hypertext Transfer Protocol)协议向 Web 服务器请求提供网页内容时,服务器仅仅是将已经设计好的静态 HTML 标准代码传送给用户浏览器,最多再加上流行的 GIF89A 格式的动态图片,比如产生几只小狗小猫跑来跑去的动画效果。若网站维护者要更新网页的内容,就必须用手工更新所有的 HTML 文档。‘静态’网站的致命弱点就是不易维护,为了不断更新网页内容,就必须不断地重复制作 HTML 文档,随着网站内容和信息量的日益扩增,网页维护的工作量无疑将是非常巨大的。

随着 Web 应用的发展,用户希望能够看到根据要求而动态生成的主页,例如响应用户数据库查询、动态生成报表等。ASP (Active Server Pages)是一种由 Microsoft 公司开发的服务器端的脚本语言运行环境。它可以结合 HTML 语言和 ActiveX 组件建立动态、交互、高效的 Web 服务器端应用程序。

所谓‘动态’网站,并不是指在网页上显示几个 GIF 动态图片,而是指应用程序会根据用户的要求和选择作出动态响应,不用修改 ASP 程序,便会自动生成新的页面,这样不同的人在不同的时间访问同一网址时,会产生不同的页面,因此可大大节省网站维护的工作量,并能很好地体现网站与客户端用户的交互性。

ASP 属于 ActiveX 技术中的服务器端技术,与常见的在客户端实现动态主页的技术,如 Java Applet、ActiveX Control、VBScript、JavaScript 等不同。客户端技术的 Script 语句是由浏览器解释执行的,而 ASP 中的 Script 语句是由服务器端解释执行的,执行结果产生动态的 Web 页面并送到浏览器,从而减轻了客户端浏览器的负担,提高了网站浏览的效率。这就是 ASP 网页的高效性。

由于 ASP 在服务器端解释执行,开发者可以不必考虑浏览器是否支持 ASP ;同时由于它在服务器端执行,在客户端的浏览器上看不到 ASP 源代码,看到的仅仅是执行后的结果,开发者也不必担心别人下载程序代码,增加了网站的安全性。

总的来说,ASP 大体上具有以下特点:

① 使用 VBScript、JScript 等简单的脚本语言,结合 HTML 语言就可以快速生成网站的应用程序。

② 可以在服务器端直接执行,不需要编译。

③ 通过各种文本编辑器就可以进行编辑设计,给程序设计人员提供了很大的方便。

④ 与浏览器无关,客户端只要使用可执行 HTML 的浏览器,即可浏览 ASP 所设计的网页内容。ASP 所使用的脚本语言(VBScript、Jscript)均在 Web 服务器端执行,不需要客户端浏览器支持这些脚本语言的引擎。

⑤ ASP 能与任何 ActiveX Scripting 语言相容,除了可使用 VBScript 或 JScript 语言设计外,还可以使用由第三方所提供的其他脚本语言,譬如 REXX、Perl、Tcl 等。

⑥ 不会把 ASP 的源程序传到客户浏览器上,因而可以避免源程序被他人剽窃,提高了程序的安全性。

1.1.1 ASP 的运行环境

ASP 本身并不是一种脚本语言,它只是提供了一种使镶嵌在 HTML 页面中的脚本程序得以运行的环境。ASP 应用程序以 .asp 文件的形式保存,而且不需要进行编译。当执行 ASP 程序时,脚本程序将一整套命令发送给脚本解释器(即脚本引擎),由脚本引擎进行翻译并将其转换成服务器所能执行的命令。在服务器端使用脚本语言,需要在服务器端安装脚本引擎。脚本引擎是用于处理脚本的 COM(Component Object Model)组件。ASP 为脚本引擎提供主机环境,并把 .asp 文件中的脚本交给脚本引擎处理。

在安装 ASP 时,系统提供了两种脚本语言:VBScript 和 JScript。VBScript 被作为系统默认的脚本语言。但是要使用一些不太常用的脚本语言的话,可能需要安装相应的脚本引擎。

Microsoft 公司对不同的操作系统环境推出了不同的组件,例如:

① 在 Windows NT Server 上需要安装 IIS:Microsoft Internet Information Server Version 3.0/4.0。

② 在 Windows NT Workstation 上需要安装 Microsoft Peer Web Services Version 3.0。

③ 在 Windows 95/98 上需要安装 PWS:Microsoft Personal Web Server。

另外,ASP 应用程序需要保存在服务器上指定的文件夹下,这个文件夹叫做 Web 服务器虚拟目录,只要该目录有 ASP 的可执行权,客户端浏览器就可以通过 WWW 的方式访问 ASP 程序了。当然允许用户修改这个虚拟目录,关于这一点,在下面 ASP 安装过程中有详细的说明。

本书着重以 Microsoft Personal Web Server 为例对 ASP 进行讲解。

这里,先介绍一下 Microsoft Personal Web Server 的安装过程。

① 将 Windows 98 的安装盘插入光驱中,例如光驱对应的盘符为"F:"。

② 在 PWS 所在文件夹"F:\add-ons\pws"下运行 setup.exe 程序,即进入安装过程;当提示输入 ASP 程序文件夹时,可以对 ASP 程序所存贮的文件夹进行设置,例如可以设置为"D:\asp文档"。安装完毕后,如果计算机名叫 zbf1,那么主页位置就在 http://zbf1 下,执行文件应该保存在"D:\asp文档"下,如图 1-1 所示。

如果需要改动 ASP 文件所在的虚拟目录,那么单击窗口左下角的'高级'图标,打开高级选项窗口进行设置,如图 1-2 所示。

1. 虚拟目录

单击虚拟目录中的"< Home >",然后单击'编辑属性'按钮,即出现编辑目录对话框,如图 1-3 所示。

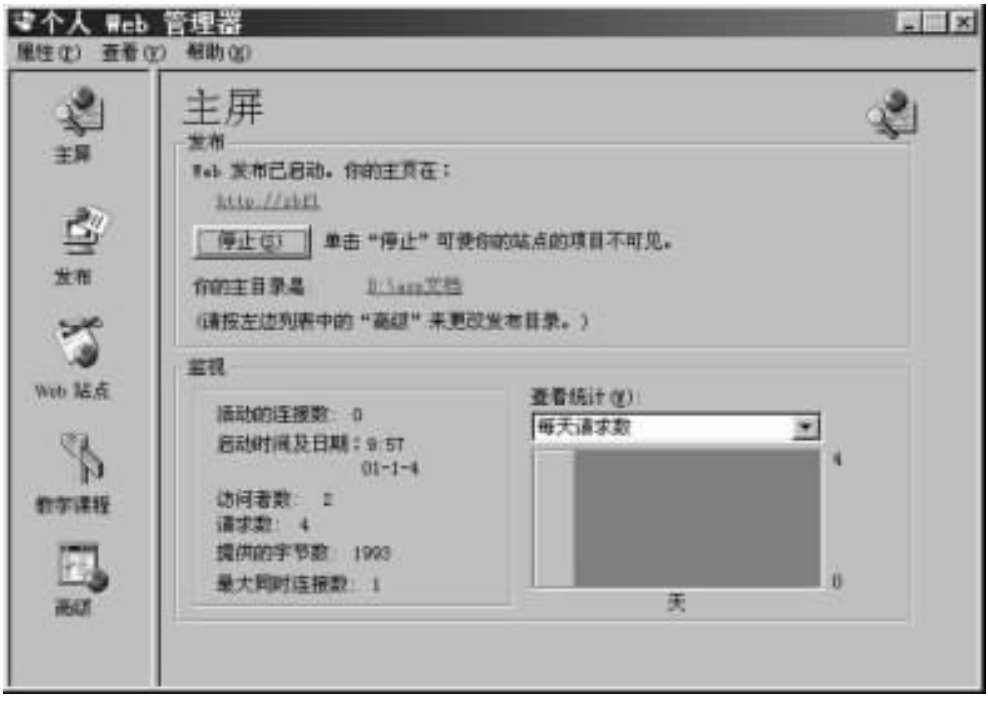

图 1－1 “个人 Web 管理器”主屏窗口

图 1－2 “个人 Web 管理器”高级选项窗口

图 1－3　‘编辑目录 ’对话框

在目录栏中输入虚拟目录所对应的物理路径,可用 ‘浏览 ’按钮进行选择,如图 1 - 4 所示。 ‘访问 ’一栏用于设置用户对 ASP 文件的访问权限 ;‘读取 ’表明用户有权查看 . asp 文件的内容,为了安全起见,建议不赋予用户这一权限 ;‘执行 ’表明用户有权执行 ASP 文件 ;‘脚本 ’表明可以执行脚本语句。要赋予用户相应的权限,只要在该选项前的复选框中打勾即可。一般来说,应选取 ‘执行 ’和 ‘脚本 ’两项。

图 1－4　‘浏览文件夹 ’对话框

2. 启用默认文档

选取 ‘启用默认文档 ’项,表示如果用户请求页面时只输入了网址或网址和目录,而没有指定具体的文件,浏览器将加载默认文档。在 ‘默认文档 ’后的文本框中输入默认文档的名称,当文档数超过一个时,各文档名之间用逗号隔开。服务器在接到没有指定文件名的页面请求时,将按次序搜寻本目录下的默认文档,然后将第一个找到的默认文档传送到浏览器。

3. 允许浏览目录

选取 ‘允许浏览目录 ’项,表示当没有启用默认文档或服务器找不到规定的默认文档时,将返回该目录下所有子目录的文件的超文本列表。为了网站的安全,建议不要选取此选项。

4. 保留 Web 站点活动日志

若选取 ‘保留 Web 站点活动日志 ’选项,将建立一个日志文件,以保存用户访问网站的活

动情况。

到现在为止,已经建立了 ASP 程序运行的环境。在 1.2 节中会给出第一个 ASP 程序的建立过程。

1.1.2 ASP 与脚本语言的关系

ASP 需要一种真正的程序语言作为脚本语言来实现,而 VBScript 或 JScript 是目前最常用的两种语言,其中 VBScript 是默认的脚本语言。脚本语言是一种介于 HTML 语言和 Java、C++、Visual Basic 等程序设计语言之间的语言。它在形式和功能上可能更接近后者,但是它的语法规则没有一般的编程语言那样严格和复杂,也无须编译。VBScript 基于 Visual Basic 语言,而 JScript 基于 Java 语言。本书以 VBScript 为例介绍 ASP。当然,读者可以选择自己熟悉的语言作为 ASP 的脚本语言。

VBScript 或 JScript 是一种解释性语言,主要由 Web 服务器或浏览器解释执行。ASP 程序可以使用各种编辑文本的工具进行编辑,例如写字板等。为此,Microsoft 公司提供了多种工具,例如 Microsoft FrontPage 和 Visual InterDev 等,都是用来开发 ASP 应用的常用工具。不过每个工具都具有自己的风格,读者可以根据自己的习惯选择编辑环境。

1.1.3 ASP 的基本组成

ASP 是通过多种语言如 HTML、VBScript 等结合使用的,一般地,ASP 程序主要包含下列四部分:

① 普通 HTML 文件的标记等内容。

② 客户端 Script 程序代码,用标记"< Script > "和"< /Script > "将该程序段包含在文件中。

③ 服务器端 ASP Script 程序代码,用标记"< % "和"% > "将该程序段包含在文件中。

④ Server - Side Include 语句,使用标记 # include 在本页中嵌入其他 Web 页面。

其中,②与③的区别在于它们的执行地方不同,客户端 Script 程序代码在客户端执行,不需要占有服务器端的处理时间;而服务器端 ASP Script 程序代码在服务器端执行。如果可以利用这两种语句合理分配客户端和服务器端的执行任务,那么将会大大改善网站的访问数据流量和访问效率。

1.2 一个简单的例子

下面以一个简单的 ASP 程序为例说明 ASP 程序的建立过程。该例子主要用来实现客户端表单的提交,在服务器端进行填写内容的提取,并将填写结果返回给客户端让访问者确认。这个功能仅用 HTML 语言很难实现。

首先我们编写客户端的表单程序 11.htm,并将它保存到对应目录"D:\asp 文档"中,其中在 HTML 代码中,标记"< ! - - "和"- - > "之间的语句一般为注释语句,有时它也可以用于特殊的用法,例如 < ! - - # include file = "……" - - > 表示该文件调用的文件名。使用任何的文本编辑器都可以创建 .asp 文件,但是使用那些带有 ASP 增强支持的编辑器将更能提高效率,如 Microsoft Visual InterDev 等。如果从未使用过 HTML,可考虑先使用 Microsoft FrontPage。使用 FrontPage 创建文档和格式化文本就像使用文字处理工具一样简单。

文件详细内容如下：

```
<html>                                    <！--HTML 开始标记-->
<head>                                    <！--HTML 头标记-->
<title>选择</title>                       <！--HTML 标题标记-->
</head>                                   <！--HTML 头编写结束-->
<body bgcolor="#d0d0d0">                  <！--HTML 程序主体开始标记-->
<b>请做出你的选择</b>
        <！--开始编写表单，并把相应的表单处理程序 answer.asp 引用进去，该程序在后面进行编写。该表单包含三个选择爱好的选择框-->
<form action="//response.asp" name=choice method=post>
        <！--编写选择框-->
<input type=checkbox name=play1 value="足球">足球<br>
<input type=checkbox name=play2 value="篮球">篮球<br>
<input type=checkbox name=play3 value="排球">排球<br>
        <！--编写提交按钮，用于触发表单处理程序的执行-->
<input type=submit name=bt value="递交">
        <！--编写重填按钮，用于清除填写内容-->
<input type=reset name=bt value="重填">
</form>                                   <！--表单编写结束-->
</body>                                   <！--HTML 程序体编写结束-->
</html>                                   <！--HTML 文件编写结束-->
```

我们在客户端浏览器地址栏中输入 http://zbf1/11.htm 以执行该程序，并进行选择，如图 1-5 所示。

图 1-5 客户端程序执行结果

然后编写用于处理表单的 ASP 程序 11response.asp。在 HTML 代码中，注释语句的标记为“<！--”和“-->”，但是在 ASP 代码中，标记“'”后的语句为注释语句，也就是 VB 中的注释语法。文件详细内容如下：

```
<html>
```

```
<head>
<title>选择</title>
</head>
<body bgcolor="#d0d0d0">
<b>你的选择</b>
<!--利用 Request 对象将客户端表单的填写结果获取到服务器端-->
<!--引入 ASP 代码端标记-->
<%
        '将选择框 play1 的内容取回到变量 nm1 中,如果该选择框被选择,则返回对应值,否则返回空值
        '由于选择框 play1 被选择,那么从客户端选择框 play1 返回到 nm1 中的值为"足球"
nm1 = request.form("play1")
        '由于选择框 play2 没有被选择,那么从客户端选择框 play2 返回到 nm2 中的值为""
nm2 = request.form("play2")
        '由于选择框 play3 被选择,那么从客户端选择框 play3 返回到 nm3 中的值为"排球"
nm3 = request.form("play3")
        '利用 Response 对象返回客户端浏览器上选择结果
response.write(nm1&" "&nm2&" "&nm3)
%>
</body>
</html>
```

当访问者在客户端浏览器上点击‘提交’按钮,会将表单提交到服务器端,服务器端执行相应的 ASP 程序进行处理,然后将处理结果返回到客户端,如图 1-6 所示。

图 1-6 ASP 程序执行结果

1.3 习题

1. 基础题

① ASP 程序在 Windows 98 中的运行环境为（ ）。

② ASP 程序主要包含哪几部分？

③ ASP Script 程序代码在（ ）端执行。

2. 操作题

① 安装 PWS,建立 ASP 运行环境。

② 实现一个类似于 1.2 节中的例子。

第二章 HTML 文档设计

由于 ASP 代码的实现离不开 HTML 语言，一般来说 ASP 代码嵌在 HTML 中，因此我们需要先对 HTML 语言作较为全面的了解，掌握它的基本语法，为进一步学习 ASP 奠定基础。

2.1 HTML 概述

在 WWW 中，单个文档称为一个页面，通常网站由多个页面组成，它们之间的联系通过彼此之间的链接来完成。这样读者可以在他们所需要的页面中寻找信息，而不需要阅读他们并不关心的内容。其中这些互相关联的 WWW 页中的第一页称为主页（Home Page），当读者利用统一资源定位器（URL，Uniform Resource Locator）链接某一 WWW 服务器时，所看到的第一个页面就是主页。读者可以通过它链接到其他所需要的页面。

URL 指出了文件在 Internet 中的位置。URL 由协议名、Web 服务器地址、文件在服务器中的路径（即目录）和文件名四部分构成，例如，http://www.microsoft.com/intl/cn/index.html。其中 http://是协议名，表示文件的运行是遵循超文本传输协议的。www.microsoft.com 是 Web 服务器的地址，/intl/cn/是文件所在的目录，index.html 则是文件名。

HTML 是 Hype Text Markup Language（超文本标记语言）的缩写。用户可以在一般的编辑器中进行编写，如 Edit、Notepad 等文本编辑工具，当然也可以用专用的 HTML 语言写作工具，如 FrontPage、InterDev 等。

HTML 文件的执行很简单，只需要在网页浏览器下浏览即可，常用的网页浏览器有 IE（Internet Explorer）或 Netscape。只要在执行的文件上双击，它会自动地在网页浏览器中执行，当然文件名后缀必须是.htm 或.html。

2.2 常用的 HTML 描述标记

我们对 HTML 已有了初步了解，下面来学习一些 HTML 中常用的描述标记。要说明的是，HTML 语言与字符的大小写无关，本书在程序中一般采用小写。

2.2.1 页面标记

文本格式 <html>……</html> 标识 HTML 文本的开头与结尾）
文件头 <head>……</head> 标识文件头区域）
文件主题 <title>……</title> 标识文件主题的开头与结尾）
文件主体 <body>……</body> 标识 HTML 主体内容的开头与结尾）
其中文件头的说明必须在文件主体前，而文件主题的说明必须在文件头中。

1. HTML 文件结构

HTML 的总体结构如下：

```
< html >
  < head >
       < title > …… < /title >
  < /head >
  < body >
       HTML 文件的正文内容
  < /body >
< /html >
```

2. 背景色彩与文字色彩的设置

一般地，网页用丰富的色彩区分不同的内容或不同的状态。例如用 bgcolor、text、link、alink、vlink 等分别表示背景色彩、非可链接文字的色彩、可链接文字的色彩、正被点击的可链接文字的色彩和已经访问过的可链接文字的色彩。

例如：

```
< body bgcolor = green text = black link = red alink = blue vlink = yellow >
```

表示文本背景色彩设为绿色，非可链接文字色彩设为黑色，可链接文字色彩设为红色，正被点击的可链接文字的色彩设为蓝色，而已经访问过的可链接文字的色彩设为黄色。

当然其中的色彩值也可用六位的 16 进制的红—绿—蓝值表示。如 # ffffff 表示白色，# ffff00表示黄色等等。

2.2.2 字体标记

1. 标题字体

```
< h ?> …… < /h ?>
```

其中“?”可以是 1，2，3……

例如：

```
< h1 > 这是 h1 标题 < /h1 >
```

2. 字体大小

```
< font size = ?> …… < /font >
```

其中“?”代表字体的大小。

例如：

```
< font size = 12 > 尺寸为 12 < /font >
```

3. 字型

< b > …… < /b > 表示写在其中的内容字体被加粗；

< I > …… < /I > 表示写在其中的内容字体是斜体；

< u > …… < /u > 表示写在其中的内容字体添加下划线；

< strike > …… < /strike > 表示写在其中的内容字体添加删除线。

下面是几个简单的例子（左边为程序代码，右边为显示结果）：

< b > 你好，朋友！ < /b >	你好，朋友！
< I > 你好，朋友！ < /I >	*你好，朋友！*
< u > 你好，朋友！ < /u >	你好，朋友！
< strike > 你好，朋友！ < /strike >	~~你好，朋友！~~

4. 字体颜色

```
< font color = ? > …… < /font >
```

表示写在其中的内容字体颜色为指定颜色，其中的颜色设置同背景颜色的设置一样。

2.2.3 文字布局标记

1. 行的控制

可以用 < br > 实现换行，也可以用 < p > …… < /p > 通过分段实现换行。

2. 文字的对齐格式

```
< p align = ?> …… < /p>
```

其中“?”为 left、center、right，表示写在其中的内容在水平方向上的格式：左对齐、居中、右对齐，其中默认值为 left。

```
< p valign = ?> …… < /p >
```

其中“?”为 top、middle、bottom，表示写在其中的内容在垂直方向上的格式：顶部对齐、居中、底部对齐，其中默认值为 bottom。

另外，

```
< center > …… < /center >
```

表示写在其中的内容在水平方向上为居中格式。

3. 文字的分区显示

```
< div align = ?> …… < /div>
```

其中“?”为 left、center、right，表示写在其中的内容作为一个整体，并称为一个分区，格式如上所述。

当然文字的布局标记还有很多，在此只提及最常用的几种，当涉及到深层次的应用时，可查阅相关的书籍。

2.2.4 表单标记

在客户端一般通过使用表单形式将填写结果提交到服务器端，由服务器端进行相应的处理。表单可以看做一个整体，其标记为 < form > …… < /form >，在其中可以有不同的内容，如文本框、单选框、提交按钮等。

1. 基本语法

表单的基本语法如下：

```
< form action = "……" method = ?name = "……" >
      ……
      ……
      < input type = submit name = "……" value = "……" >
      < input type = reset name = "……" value = "……" >
< /form >
```

其中 action 选项对应的字符串是 form 所要提交的处理程序的路径，即它的 URL；method 选项为表单提交方法，“?”为 post 或 get，其区别可参见 5.1 节；name 为表单名称；对表单进行填充，并按下“submit”按钮，便启动了相应的 action 处理；“reset”按钮对应的是所填内容的清空操作。

例如在下列 HTML 代码中就有一个表单，要求提交用户名到服务器端。详细内容如下：

```
< html >
< head >
< title > New Page 1 < /title >
< /head >
< body >
< form method = "post" action = "ch6resp.asp" name = Form1 >
    你的用户名：< input type = "text" name = "nm" size = "15" > < br >
    < br > < input type = "submit" name = "st" value = "提交" >
    < input type = "reset" name = "cl" value = "清除" >
< /form >
< /body >
< /html >
```

2．文本框

```
< input type = "text" size = ?name = "……" value = "……" >
```

上式为一般文本输入框的描述，其中 size 表示文本框的大小，表示可以显示的字符数。其中 name 表示所赋予的变量名，value 表示打开本页时所赋予的初始值。

密码输入框是一个特殊的文本输入框，它以特殊的方式显示用户输入的字符。密码输入框的描述如下：

```
< input type = "password" size = ?name = "……" >
```

3．复选框和单选框

（1）复选框

```
< input type = checkbox name = "……" value = "……" >
```

其中 name 表示对应复选框的变量名，value 表示该选项所对应的值，也就是表示当该项在客户端被选定后传送到服务器端的值。

请看下面的例子。为使代码结构清晰直观，这里采用缩进形式书写。但是，一般的编辑器不支持这种形式，读者可以根据自己的书写习惯。无论如何，清晰的代码结构是非常重要的。

```
<html>
    <head>
        <title>选择</title>
        </head>
        <body bgcolor="#d0d0d0">
        <b>请做出你的选择</b>
        <form action="answer.asp" name=choice method=post>
        <input type=checkbox name=play1>足球<br>
        <input type=checkbox name=play2>篮球<br>
        <input type=checkbox name=play3>排球<br>
        <input type=submit name=bt value="递交">
        <input type=reset name=bt value="重填">
        </form>
    </body>
</html>
```

其中 action 选项对应的字符串“answer.asp”表示对应的表单处理程序名。上述程序的执行结果，如图 2－1 所示。

图 2－1 复选框使用示例

(2) 单选框

```
<input type=radio name="……" value="……">
```

其中 name 表示对应单选框所赋予的变量，value 表示该选项所对应的值，也就是表示当该项在客户端被选定后传送到服务器端的值。

例如：

```
<html>
<head>
<title>选择</title>
</head>
<body bgcolor="#d0d0d0">
```

```
<b>请做出你的选择</b>
<form action="answer.asp" name=choice method=post>
    <input type=radio name=play>足球<br>
    <input type=radio name=play>篮球<br>
    <input type=radio name=play>排球<br>
    <input type=submit name=bt value="递交">
    <input type=reset name=bt value="重填">
</form>
</body>
</html>
```

其中 answer.asp 程序表示对应的表单处理程序。上述程序的执行结果，如图 2-2 所示。

图 2-2 单选框使用示例

对上述两程序进行对比，就会发现它们之间的区别，读者可以根据具体情况选择使用。

4. 按钮

```
<input type="button" name="……" value="……" onclick="……">
```

其中 name 表示对应按钮的登记名，value 表示该按钮在客户端所显示的名称，onclick 表示对应该按钮的响应事件。

例如：

```
<html>
<head>
<title>消息框的弹出</title>
</head>
<body>
<script language="vbscript">
<!--
sub welcome                    '响应事件
    msgbox ("你好")
```

```
end sub
- - >
</script>
<input type = button name = "welcome" value = "welcome" onclick = "welcome" >
</body>
</html>
```

在客户端执行上述程序的结果,如图 2 - 3 所示。

图 2 - 3 按钮使用示例

在客户端点击该按钮可激发一个消息框的产生。

2.2.5 表格标记

一般地,网页中有丰富的表格形式。那么,如何在网页中作出美观的表格呢?

1. 表格的基本用法

表格分为带边框的表格和不带边框的表格,当需要带边框时,用

```
<table border> …… <table>
```

表示;当不需要边框时,用

```
<table> …… </table>
```

表示即可。

```
<th> …… </th>  定义表头;
<tr> …… </tr>  定义表格中的一行;
<td> …… </td>  定义表格中的一个表格单元。
```

2. 跨多行、跨多列的表格单元

在表格中,当一个表格单元需要占几个列单元时,用下述语句表示:

```
<th colspan = ?> …… </th>
<td colspan = ?> …… </td>
```

其中“?”表示所占的列单元数。

当一个表格单元需要占几个行单元时,用下述语句表示:

```
<th rowspan = ?> …… </th>
```

```
< td rowspan = ?> …… < /td >
```

其中“?”表示所占的行单元数。

3. 表格尺寸设置

(1) 边框尺寸设置

```
< table border = ?>
```

其中“?”表示表格的边框宽度。

(2) 表格尺寸设置

```
< table border width = ?height = ?>
```

其中“?”表示表格的宽度和高度。

(3) 表格单元间隙设置

```
< table border cellspacing = ?>
```

其中“?”表示像素大小。

(4) 表格单元内部空白设置

```
< table border cellpadding = ?>
```

其中“?”表示像素大小。

4. 表格内的文字布局

```
< tr align = ?>
< th align = ?>
< td align = ?>
```

其中“?”为 left、center 或 right,分别表示表格中的一行、表头和单元表格分别在水平方向上的对齐方式。

```
< tr valign = ?>
< th valign = ?>
< td valign = ?>
```

其中“?”为 top、middle 或 bottom,分别表示表格中的一行、表头和单元表格分别在垂直方向上的对齐方式。

5. 表格在页面上的布局

表格在页面上的水平对齐方式用下面的形式表示:

```
< table align = ?>
```

其中“?”为 left、center 或 right。

另外,Netscape 中还允许定义表格在页面中垂直方向上的位置,表示为:

```
< table vspace = ?hspace = ?>
```

其中“?”为 space value (仅用于 Netscape)。

6. 表格的标题

表格标题在水平方向和垂直方向上的对齐方式用下面的形式表示：

```
< caption align = ?> …… < /caption >
```

其中“?”为 left、center 或 right。

```
< caption valign = ?> …… < /caption >
```

其中“?”为 top、middle 或 bottom。

7. 表格的色彩

< th bgcolor = ?> 表示设置表格单元的背景色彩；

< th background = "URL" > 表示设置表格单元的背景图案，其中 URL 表示对应的图像文件名路径；

< table bordercolor = ?> 表示表格边框的色彩；

< table bordercolordark = ?> 和 < table bordercolorlight = ?> 表示设置边框色彩的亮度控制。

8. 表格中边框的显示

(1) 显示四个边框

```
< table frame = box > …… < /table >
```

(2) 只显示上边框

```
< table frame = above > …… < /table >
```

(3) 只显示下边框

```
< table frame = below > …… < /table >
```

(4) 只显示上、下边框

```
< table frame = hsides > …… < /table >
```

(5) 只显示左、右边框

```
< table frame = vsides > …… < /table >
```

(6) 不显示任何边框

```
< table frame = void > …… < /table >
```

9. 表格设计举例

程序如下：

```
< html >
< head >
< title > a example < /title >
< /head >
< body >
< table frame = box bgcolor = " # d0d0d0" border = "1" width = "100 %" >
< caption align = center > 成绩 < /caption >
```

```
<th colspan=5>上海大学</th> 0
<tr align=center>
<td width="20%">学号 </td>
<td width="20%">姓名 </td>
<td width="20%">数学 </td>
<td width="20%">物理 </td>
<td width="20%">总分 </td>
</tr>
<tr align=left>
<td width="20%">9601 </td>
<td width="20%">王璇 </td>
<td width="20%">98</td>
<td width="20%">89 </td>
<td width="20%">187 </td>
</tr>
<tr align=center valign=middle>
<td width="20%">9602 </td>
<td width="20%">陈东 </td>
<td width="20%">97 </td>
<td width="20%">76</td>
<td width="20%">173 </td>
</tr>
<tr align=right>
<td width="20%">9603 </td>
<td width="20%">郭洪 </td>
<td width="20%">95 </td>
<td width="20%">80 </td>
<td width="20%">175 </td>
</tr>
</table>
</body>
</html>
```

上述程序运行结果,如图 2-4 所示。

2.2.6 多窗口页面

可以把一个页面分为几部分,每一部分对应一个 HTML 文件。这样的页面成为多窗口页面。一般地,在较为复杂的页面中,都是这种多窗口页面。

1. 基本语法

```
<frameset>……</frameset>
```

2. 各窗口的尺寸设置

以下(1)、(2)、(3)中的 ww1.htm、ww2.htm 和 ww3.htm 等 HTML 文件都是已编写好的

图 2-4 学生成绩统计表格

HTML 文件。

(1) 纵向排列多个窗口

例如：

```
<frameset cols=30%,20%,50%>
<frame src="ww1.htm">
<frame src="ww2.htm">
<frame src="ww3.htm">
</frameset>
```

(2) 横向排列多个窗口

例如：

```
<frameset rows=25%,25%,50%>
<frame src="ww1.htm">
<frame src="ww2.htm">
<frame src="ww3.htm">
</frameset>
```

(3) 纵横向排列多个窗口

例如：

```
<frameset cols=25%,75%>
<frame src="ww1.htm">
<frameset rows=50%,50%>
<frame src="ww2.htm">
<frame src="ww3.htm">
</frameset>
</frameset>
```

3．frame 的外观

frame 表示单窗口页面的设置，而 frameset 表示多窗口页面的设置，在学习下述内容前需要对它们的区别有所认识。

（1）各窗口边框的设置

```
< frame frameborder = ?>
```

其中“?”为 yes 或 no。

（2）各窗口间空白区域的设置

```
< frameset framespacing = ?>
```

其中“?”为空白区域的像素大小。

（3）各窗口边框色彩的设置

```
< frameset bordercolor = ?>
```

其中“?”是预定义的色彩值。

（4）窗口的页面空白设置

```
< frame marginwidth = ?marginheight = ?>
```

其中“?”为空白区域宽度和高度的像素大小。

（5）窗口的卷滚条设置

```
< frame scrolling = ?>
```

其中“?”为 yes、no 或 auto，缺省值为 auto。

4．多窗口页面举例

25．htm 程序如下：

```
< html >
< head >
< meta http - equiv = "Content - Type" content = "text/html；charset = gb2312">
< meta name = "GENERATOR" content = "Microsoft FrontPage 4.0" >
< meta name = "ProgId" content = "FrontPage.Editor.Document" >
< title > 多页面例子 < /title >
< /head >
< frameset rows = "50，* ，100" >
< frame src = "page 1.htm" >
< frameset cols = "50 % ，* " >
< frame src = "page 2.htm" scrolling = "auto" noresize >
< frame src = "page 3.htm" >
< /frameset >
< frame src = "page 4.htm" >
< /frameset >
< /html >
```

page1．htm 程序如下：

```
<html>
<head>
<title>页面1</title>
</head>
<body bgcolor="#d0d0d0">
<h2 align="center">这是窗口1</h2>
</body>
</html>
```

page2.htm、page3.htm、page4.htm 的程序代码与 page1.htm 相似，其运行结果，如图 2-5 所示。

图 2-5 简单的多窗口页面

2.2.7 多媒体页面

1. 嵌入多媒体文本

```
<embed src=?>
```

其中“?”表示所嵌入文本的 URL。本标记可以在主页中嵌入电影、声音等多媒体文件。

2. 背景音乐

```
<bgsound src=?>
```

其中“?”表示所嵌入文本的 URL。

```
<bgsound loop=?>
```

其中“?”表示循环次数。

2.2.8 总结

本节只对 HTML 的基本用法作了简要的概述,读者在学习 ASP 之前,应该对 HTML 有一定的了解,它是进一步学习 ASP 的基础。

2.3 习题

1. 基础题

① HTML 文档在总体结构上分为哪几部分?

② 表单的基本格式是什么?

③ 在处理表格时,表格单元在水平方向和垂直方向上的格式用哪几个标识确定?

④ 你能对 HTML 的用法做一些总结吗?

2. 操作题

试用 HTML 语言建立一个简单的文档,并满足以下要求:

① 用黑体、一号字显示文档标题;

② 建立一个表单,其中含有对姓名的填写,对性别的单项选择和对个人爱好的多项选择。

③ 建立一个表格,仿照书中的例子显示一些数据。

④ 在页面中添加背景音乐。

第三章 ASP 脚本语言 VBScript

ASP 就是嵌入某种脚本语言的 HTML 文档,常用的脚本语言有 VBScript 和 JScript。这里我们仅介绍 VBScript,对 JScript 有兴趣的读者可以查阅有关资料。

3.1 VBScript 概述

VBScript 脚本语言是 Visual Basic 的一个子集,但它的内容还是很丰富的。在本章中,我们主要介绍它的用法,让初学者对它有一个较为全面的了解。

在 Web 页面中增加 VBScript,使得在调用表单数据处理程序之前可以先进行数据项处理和检验,动态地创建新的 Web 内容,甚至可以编写完全在客户端运行的应用程序,可以扩展客户端的功能,减轻服务器端的压力,对用户的操作立即作出反应,提高网站的运行速度。这是在应用程序中引入 VBScript 的最重要的一点。

但 VBScript 是受限制的,它不能调用 API,不能直接操纵客户机上的文件,当然也不能拥有文件系统之上的控件。

如果读者需要进一步了解有关 VBScript 的详细内容,可以查阅相关信息。

3.2 VBScript 基本语法

3.2.1 数据类型

在 VBScript 中只有一种数据类型 Variant,它可以包含不同类别的信息,可以是整型或是字符串型等。这对用户而言可能是一种好消息 在定义数据时不用考虑它的实际表示类型,在使用时根据赋予它的值来确定它的类型,而且在使用时可以随时改变它的类型。当然,这样使用容易产生混乱,可能造成数据的丢失,所以使用时要分清它的使用类型,最好不要在使用时改变它的类型。

Variant 中常用的数据子类型及其含义,如表 3-1 所示。

表 3-1 常用的数据子类型及其含义

子类型	含 义
Empty	未初始化的 Variant。数值变量 (0),字符串变量 ("")
Null	不包含任何有效数据的 Variant
Boolean	True 或者 False
Integer	包含 - 32 768 ～ 32 767 之间的整数

续表）

子类型	含　义
Long	包含 - 2 147 483 648～2 147 483 647 之间的整数
Single	包含单精度浮点数
Double	包含双精度浮点数
Date	包含表示日期的数字
String	包含变长字符串

当然在 Variant 中还含有一些不常用的类型，在此就不一一列举了。

3.2.2 变量、常量

1. 变量

变量名是一种指向变量内存地址的指针，可以通过变量名方便地改变和引用变量中的值。在 VBScript 中，所用变量的数据类型都是 Variant。

（1）变量声明

变量声明的一种普通方式是使用 Dim 语句。

例如：

```
dim name                          '定义单个变量
dim age, class, score             '定义多个变量
```

（2）变量命名法则

当定义变量名时，需遵循以下的命名法则：

① 第一个字符必须是英文字母。

② 变量名一般由字母 a～z、A～Z、数字 0～9 和字符"_"构成。

③ 变量名长度不可超过 255 个字符。

④ 在作用域内变量名必须惟一。

（3）变量的分类

变量一般分为过程级变量（局部变量）和 Script 级变量（全局变量），如此分的依据主要是它们的作用域不同。其中过程级变量在过程中声明，它的生存期只是变量所在过程的运行期，并且随着过程的结束而释放它所占有的空间；Script 级变量在过程之外声明，它的生存期是它所在页面的生存期，相对于过程级变量，它的作用域是较大的。如果读者学习过其他语言的话，对这一点会有较深刻的理解。

2. 常量

常量表示固定值，只可以在过程中引用，且不可改变它的值。

当定义一个常量时，可以使用 Const 语句在 VBScript 中创建。一般用它表示具有一定含义的字符串和数值型常数等。例如：

```
const MyHome = "shanghai"
const MyRoom = 130
const Today = #12-20-2000#
```

其中定义字符串文字时,必须将其包含在“……”之间,这样就可以区分字符串常量和数值型常量。当定义日期时间型常量时,必须将其包含在“# …… # ”之间,如第三句所示。

3.2.3 运算符

在 VBScript 中,运算符包括算术运算符、比较运算符、连接运算符和逻辑运算符。当表达式包含多种运算符时,它们的优先级是:算术运算符 > 连接运算符 > 比较运算符 > 逻辑运算符(优先级自左向右逐渐降低)。

表 3-2 表示常用的运算符(优先级自上向下逐渐降低,但所有的比较运算符优先级都相同)。

表 3-2 常用的运算符

算术运算符		连接运算符		比较运算符		逻辑运算符	
描述	符号	描述	符号	描述	符号	描述	符号
求幂		字符串连接	&	等于	=	逻辑非	Not
负号	-			不等于	< >	逻辑与	And
乘	*			小于	<	逻辑或	Or
除	/			大于	>	逻辑异或	Xor
整除	h			小于等于	< =	逻辑等价	Eqv
求余	Mod			大于等于	> =	逻辑隐含	Imp
加	+						
减	-						

这里,对算术运算符不再介绍;连接运算符“&”是字符串之间的连接,返回一个将符号“&”两端的字符串首尾依次连接起来的字符串;比较运算符都是双目运算符,是对符号两端的数据进行比较并返回布尔值,符号两端的数据类型要求一致;逻辑运算符是进行布尔型之间的运算,并且也返回布尔型。

3.3 控制语句

在 VBScript 的流程控制中,一般通过循环语句和条件语句来实现。如果读者要掌握 VBScript,那么必须熟练运用这些控制语句。本节将对循环语句和条件语句进行详细的介绍,希望读者能够灵活掌握。

3.3.1 循环语句

循环语句主要用于重复执行一组语句,循环语句一般分为两类。

1. 执行语句次数不可预先指定的重复执行语句

(1) Do……Loop

在 VBScript 中,它可分为两类:

① Do While 条件语句……Loop 和 Do……Loop While 条件语句。

前者在进入循环之前检查条件,而后者在循环至少运行完一次后检查条件。

② Do Until 条件语句……Loop 和 Do……Loop Until 条件语句。

这两者之间的区别与①中所述一样,在此不再叙述。

①和②的区别在于,在①中当条件语句为 True 时重复执行语句,而②中当条件语句为 False 时重复执行语句。

例 1:实现一个从 1 到 100 的累加运算,分别用以上几种语句实现。

使用 'Do While 条件语句……Loop '实现,其中 Msgbox 函数的作用是将参数字符串以消息框的形式显示。

```
sub Sum11 ( )
dim Total, Count
    total = 0
    count = 1
    do while Count < = 100
        total = Total + Count
        count = Count + 1
    loop
    msgBox "累加和为"&Total
end Sub
```

使用 'Do……Loop While 条件语句 '实现。

```
sub Sum12 ( )
dim Total, Count
    total = 0
    count = 1
    do
        total = Total + Count
        count = Count + 1
    loop while Count < = 100
    msgbox "累加和为"&Total
end Sub
```

使用 'Do Until 条件语句……Loop '实现。

```
sub Sum21 ( )
    dim Total, Count
    total = 0
    count = 1
    do until Count > 100
        total = Total + Count
        count = Count + 1
    loop
    msgbox "累加和为"&Total
```

```
end sub
```

使用“Do……Loop Until 条件语句”实现。

```
sub Sum22 ( )
    dim Total, Count
    total = 0
    count = 1
    do
        total = Total + Count
        count = Count + 1
    loop until Count > 100
    msgbox "累加和为"&Total
end sub
```

以上语句说明实现某一个编程目的,语句的编写不是惟一的。应该根据需要选择适当的方法,不要局限于一种模式。当然不仅局限于这些,在以后的 ASP 内容中,还会进一步展开。

(2) While……Wend

该语句的实现形式是:While 条件语句……Wend。当条件语句为真时,继续执行循环内的语句。例如当我们要实现例 1 任务的子过程时,使用“While 条件语句……Wend”实现。

```
sub Sum31 ( )
    dim Total, Count
    total = 0
    count = 1
    while Count < = 100
        total = Total + Count
        count = Count + 1
    wend
    msgbox "累加和为"&Total
end sub
```

尽管该语句也能实现,但在 VBScript 编写中,由于缺乏灵活性,因此我们不鼓励使用它,而建议最好使用 Do……Loop 语句。

2. 执行语句次数可以预先指定的重复执行语句

(1) For……Next

For……Next 语句用于将语句运行指定的次数。实现形式为:For 计数器变量 = 初始值 to 终止值 Step 步长……Next。在循环中使用计数器变量,可以预先设定每一次循环时计数器的变化量(步长),它可以是正变化量或负变化量,当计数器值等于终止值时,便退出循环。

下面我们分别用两种方法实现以上例 1 中的任务。

使用“For……Next”实现,其中步长 Step 为正值(默认值为 1)。

```
sub Sum41 ( )
```

```
    dim Total, Count
    total = 0
    for Count = 1 to 100
        total = Total + Count
    next
    msgbox "累加和为"&Total
end sub
```

使用“For……Next”实现，其中计数器变化量 Step 为负值。

```
sub Sum42 ( )
    dim Total, Count
    total = 0
for Count = 100 to 1 Step -1
        total = Total + Count
    next
    msgbox "累加和为"&Total
end sub
```

如果要在计数器值达到其终止值之前退出 For……Next 循环，可以使用 Exit For 语句。当然这一般要在某些特殊情况下才使用，例如当发生严重错误时。一般要和条件语句“IF 判断语句 Then……Else……End If”结合使用，我们会在下一节中对 IF 判断语句进行详细讲述。

(2) For Each……Next

For Each……Next 循环其实与 For……Next 循环相似。不同之处在于，For Each……Next 不像 For……Next 一样将循环语句运行指定的次数，而是对于数组中的每个元素或对象集合中每一项重复一次循环语句，循环次数应该与数组中的元素个数或对象集合中的项数对应。

下面，我们再利用 For Each……Next 实现例 1 中的任务。在处理中，先使用 For……Next 对数组 num 进行赋值，然后再使用 For Each……Next 进行数组求和。

```
sub Sum51 ( )
    dim I, Total
    dim num (9) 'VBScript 中数组下标是从 0 开始的，所以 num 数组含有 100 个元素
    for I = 0 to 99
        num (I) = I + 1
    next
    total = 0
    for each I in num
        total = Total + I
    next
    msgbox "累加和为"&Total
end sub
```

当然没有必要如此繁琐地处理如此简单的一个任务，它的用武之地并不在这里。

3.3.2 条件语句

条件语句主要用于控制流程，它和循环语句不同，是根据不同情况产生不同的处理分支。一般来说，条件语句分为两种：

① If……Then……Else……End If 语句。

② Select Case 语句。

下面我们分别进行讲解。

1. If……Then……Else……End If 语句

语句格式为：

```
if 条件语句 then
    语句 1
else
    语句 2
end If
```

主要根据条件语句的真假进行相应的操作。如果条件语句为 True 时，则执行语句 1，否则，执行语句 2。

例 2：

```
sub Judge (con)
    if con = 0 then                    '对应的条件语句
        msgbox "你未输入值"             '对应的语句 1
    else
        msgbox "你好，欢迎你！"         '对应的语句 2
    end If
end sub
```

当 con = 0 时，即条件语句为 True 时，弹出消息框提示“你未输入值”；当 con 不等于 0 时，即条件语句为 False 时，弹出消息框提示“你好，欢迎你！”

如果只进行条件语句为真的操作，那么用到的语句格式为：

```
if 条件语句
then
    语句 1
end If
```

如果需要对多个条件语句进行判断时，那么就需要添加 ElseIf 子句，从而对 If……Then……Else……End If 语句的功能进行扩充，使其可以用于多种可能的程序流程控制。

例 3：

```
sub Judge (con)
    if con = 0 Then                    '对应的条件语句 1
        msgbox "你未输入值"             '语句 1
    elseif con = 1 Then                '条件语句 2
```

```
        msgbox "用户不合法"                    '语句 2
    else
        msgbox "你好,欢迎你 !"                 '语句 3
    end if
end sub
```

当 con = 0 时,即条件语句 1 为 True 时,弹出消息框提示 "你未输入值 ";当 con = 1 时,即条件语句 1 为 False,但条件语句 2 为 True 时,弹出消息框提示 "用户不合法 ";否则,弹出消息框提示 "你好,欢迎你 !"。

其中应该注意的是,当使用多条件进行判断时,End If 语句的个数应该与 If 语句的个数一致。由于使用多个 ElseIf 子句经常会使程序的可读性很差,因此在对一个参数值进行多选择时,更好的办法是使用 Select Case 语句。

2. Select Case 语句

Select Case 结构在语句块执行之前,对条件表达式语句进行计算,根据表达式的计算结果与结构中的每个 Case 的值进行比较。如果匹配,则执行与该 Case 关联的语句块。

语句格式如下:

```
select 条件表达式语句
    case 分支值 1
        语句块 1
    case 分支值 2
        语句块 2
    ……
    case 分支值 n
        语句块 n
    case else
        语句块 n + 1
end select
```

当然 Select Case 语句也可以通过 If 语句实现,但在 Select Case 语句中只计算一次表达式值,在 If 语句中要计算若干个条件。

下面我们用 Select Case 语句实现例 3。

例 4:

```
sub Judge (con)
    select con
        case 0
            msgbox "你未输入值"                '语句块 1
        case 1
            msgbox "用户不合法"                '语句块 2
        case Else
            msgbox "你好,欢迎你 !"             '语句块 3
    end select
end sub
```

总之,由于两种条件语句各有特点,应该根据不同情况灵活使用。

3.4 VBScript 过程

在程序中,为了调试和编写的方便,经常使用过程。通过过程,可以使程序可读性好,便于调试,并且可以避免程序段重复编写。因此,我们应该掌握过程的编写和使用。

在 VBScript 中,过程分为两类:Sub 过程和 Function 过程。

3.4.1 Sub 过程

实际上在以上举的例子里,我们已经使用了 Sub 过程,现在我们再在这里作一系统阐述。

Sub 过程的语句格式如下:

```
sub 过程名 参数值1,……)
    过程语句块
end sub
```

Sub 过程执行时,不返回值。Sub 过程可以使用参数,如果没有参数,在过程名后也必须有空括号( )。

3.4.2 Function 过程

Function 过程与 Sub 过程类似,它们之间的区别是 Function 过程可以返回值,而 Sub 过程不可以。它的语句格式如下:

```
function 过程名 参数值1,……)
    过程语句块
end function
```

与 Sub 过程一样,Function 过程可以使用参数,如果 Function 没有参数时,在过程名后也必须包括空括号( )。在 Function 过程中,可以通过 Function 过程名返回值,这个值是在过程语句块中赋予过程名的,Function 返回值的数据类型是 Variant。如果需要调用该 Function 过程,则可使用赋值语句。下面用一个例子进行说明。

例 5:

```
sub Showtotal (ov)
    msgbox "总体和是"&Totalnum (ov))
end sub
function Totalnum (num)
    dim I, Total
    total = 0
    for I = 1 to num
        total = Total + I
    next
    totalnum = Total
```

```
    end function
```

在例 5 中，Function 过程计算出从 1 到参数 num 的累加和，Sub 过程 Showtotal（tov）调用 Function 函数返回从 1 到参数 tov 的累加和，并以消息框的形式给出。

3.5 总结

在本章节中，我们介绍了 VBScript 的基本语法、控制语句和过程的使用等，这些是 ASP 中用到的最基本的内容。

下面我们在例 4 的基础上完成一个完整的综合实例。

例 6：

```
< html >
< head >
< title > 消息框的弹出 < /title >
< /head >
< body >
< script language = "vbscript" > < ! - -
sub Judge (con)
if con = 0 then
    Msgbox ("你未输入值")
elseif con = 1 Then
    msgbox ("用户不合法")
else
    msgbox ("你好，欢迎你！")
end if
end Sub
- - >
< /script >
< input type = "button" name = "0" value = " 0 " onclick = "judge (0)" >
< input type = "button" name = "1" value = " 1 " onclick = "judge (1)" >
< input type = "button" name = "2" value = " 2 " onclick = "judge (2)" >
< /body >
< /html >
```

在显示本页面时，会出现三个按钮，如图 3 - 1 所示。分别按下这三个按钮，会显示不同的消息显示框。

本章对 ASP 的基础——VBScript 作了简要介绍。如果读者需要深入研究 VBScript 的话，请参考有关书籍。

图 3－1　‘消息框的弹出’使用示例

3.6　习题

1．基础题

① 在 VBScript 中只有一种数据类型（　　）。

A．Char　　B．Variant　　C．Integer　　D．Date

② 在控制语句的循环语句（　　）中，循环语句不用计数器变量重复执行循环语句，但可以预先指定循环次数。

A．Do……Loop　　B．While……Wend

C．For……Next　　D．For Each……Next

③ Sub 过程和 Function 过程之间的主要区别是什么？

2．操作题

① 编写一个 Sub 过程，参数可以有年份和月份。在其中用 Select Case 语句进行判断并给出给定年份的给定月份的天数，在二月的天数判断上，可以用 If……Then……Else……End if 语句判断是否闰年，进而给出二月份的天数。

② 编写一个 Sub 过程 Outp1（1）和 Function 过程 Total1（2），在 Function 过程 Total（2）中计算从 1 到 c2 的乘积，并在 Outp1（1）中调用 Total1（1），将 Total1（1）的结果以消息框的形式给出。

第四章　ASP 应用基础

4.1　基础知识

4.1.1　文件结构

前面已经讲过，ASP 文件不是一个编译过的文件，它是一个文本文件，可以用文本文件编辑器如记事本打开和编辑 ASP 文件。

一般一个 ASP 文件包括以下内容：

① 标准的 HTML 标记。

② 服务器端执行的位于 < %……% > 界定符之间的 ASP 代码。

③ 客户端执行的位于 < Script > 与 < /Script > 之间的脚本代码。

④ 包含文件语句 # include，将程序中用到的子过程所在的程序文件包含进去。

目前可以在 ASP 中使用 VBScript 脚本语言和 JScript 脚本语言，其中系统默认的是 VBScript 语言。

ASP 文件名需要以 . asp 为扩展名，这样服务器才可知道这是需要在服务器上解释执行的代码文件。当然一个 HTML 文件也可以保存为以 . asp 为后缀结尾的文件。. asp 文件中的 ASP 源代码在客户端是不可见的，服务器端将 ASP 代码的执行结果以 HTML 的格式送至客户端，因此在客户端浏览器上看到的 . asp 文件的代码是标准的 HTML 文件。

例如 在服务器端执行如下 ASP 语句时，

```
< html >
< head >
< title > New Page 1 < /title >
< /head >
现在时间为
< % = now ( ) % >                                    < ! - - ASP 源代码 - - >
< /body >
< /html >
```

在客户端上的显示页面，如图 4 - 1 所示。

在客户端看到的代码为：

```
< html >
< head >
< title > New Page 1 < /title >
```

```
</head>
<body>
现在时间为
01-1-4  10:29:41                    <!--ASP 源代码-->
</body>
</html>
```

图 4-1　时间显示示例

这样，就可以避免别人看到自己的 ASP 程序源代码了，在某种意义上讲，还保护了网站的安全性。

4.1.2　基本语法

在 ASP 中有六个功能强大的内部对象，利用它们以及其他第三方组件可以实现大部分功能，表 4-1 列出了 ASP 内部对象名称及其功能。

表 4-1　ASP 内部对象名称及其功能

名　称	功　能
Response 对象	服务器端使用它向客户端传送信息
Request 对象	服务器端使用它从客户端取回信息
Server 对象	提供大量的服务器功能
Application 对象	存储用户个人的信息，并且只能被用户本人访问
Session 对象	公共存储区域，可以由不同用户来访问
ObjectContext 对象	用来配合 Microsoft Transaction 服务器进行分布式事务处理

在第五章和第七章中将对它们进行详细论述，在此仅作简单说明。Response 对象和 Request 对象主要用于服务器端和客户端之间的交互，实现 C/S 模式；Server 对象可以提供有关服务器的功能，可以用它建立有关的连接；Application 对象可以存储用户个人的消息，其中最重要的是它只能让用户本人访问，增加了健壮性，它的典型使用是在聊天室的建立上，利用 Application 对象传递信息；Session 对象与 Application 对象的不同之处在于它是一个公共存储区域，可以存储不同用户存放的公共信息，也可以让不同用户进行访问；而 Objectcontext 对象没有前五个对象那样常用，这里不再赘述。

在 ASP 文件中，一般包括三种标记：HTML 标记、< Script > 标记和 < %……% > 标记。

1．HTML 标记

在 HTML 标记的两端要加上"< "和"> "分隔符，例如：

```
< font size = 15 > 欢迎你，朋友！ < /font >
```

定义了分隔符之间的字符以 15 号字体显示，这在第二章中已作了详细的解释，在此不再赘述。

2．< Script > 标记

用它来标记脚本语言的程序块，当然可以使用多种脚本语言，如 VBScript 和 JScript 等。需要说明的是，如果要编写服务器端处理的模块，则需要在 < Script > 标记内指定属性 Runat = Server 如果忽略了这个属性，那么它默认为脚本语言是在客户端执行的。至于有关 VBScript 的详细内容，已在第三章中作了讲述。

3．< %……% > 标记

在网站的编写中，用“< % ”和“% > ”将在服务器端执行的 ASP 脚本包含起来，以区别于其他语句。例如：

```
<% response.write "hello！world！" %>
```

在“<% ”和“%> ”之间的便是 ASP 脚本，服务器端使用它达到在客户端显示“hello！world！”的目的。

4.1.3 声明脚本语言

ASP 中使用的脚本语言默认为 VBScript。如果在使用“< % ”和“% > ”前不作任何有关使用语言的声明，那么其中的语句就当作 VBScript 来处理。但是现在使用 JScript 的也越来越多。我们可以以不同的方式指定使用的脚本语言，如果读者用的是 PWS（Personal Web Server），可以使用以下两种方法在文档中改变脚本语言。

第一种方法是直接在 .asp 文件中加以说明，这样可以指定在特定的主页中使用的脚本语言。一般将说明语句“< % @ Language = ?% > ”其中“?”表示使用的脚本语言）放在主页文件的第一行。例如：

```
<% @ Language = jscript %>
< html >
< head >
< title > 我的主页 < /title >
< /head >
< body >
<%
for (i = 1;i < 11;i + +) {
    for (j = 1;j < = i;j + +) {%>
    < % = j %>
    < %} %>
< br >
<%} %>
```

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

在执行该脚本时,将会在浏览器上看到一个由数字组成的三角形。正如上面所述,ASP 文件的第一行声明了在该文件中的“<%”和“%>”之间所用的默认语言是 JScript,这样就可以在“<%”和“%>”之间使用 JScript 脚本语言编写自己的文档。要注意的是要在@和保留字 Language 之间留出一个空格。

第二种方法是利用 HTML 中的 <Script> 标记说明,通常格式为:

```
<script language="……" runat="……">
```

一般用 Language 属性限定使用了哪一种语言,用 Runat 属性指明该脚本是在服务器端还是在客户端执行。例如:

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<script language="jscript" runat="server">
function reply()
 {esponse.write"hello!world!"
}
</script>
我的问候是:<%reply()%>
</body>
</html>
```

在本 ASP 文件中,由于在文件头没有限定使用的脚本语言,所以默认使用的脚本语言是 VBScript。但在文件中又需要使用 JScript 脚本语言,因此我们需要用 <Script> 重新指定脚本语言。当该文件被执行到开始调用 Reply 函数时,脚本语言就改变成为 <Script> 中 Language 属性指定的 JScript 语言,执行结束后浏览器上会显示字符串‘我的问候是:hello!world!”。

如果在应用程序中使用同一种脚本语言,那么通常用第一种方法指定使用的脚本语言;如果在同一个应用程序中使用多种语言,那么用第二种方法实现比较灵活。

4.1.4 <Script> 标记和 <%……%> 标记的区别

可以说 <Script> 标记和 <%……%> 标记的使用是相辅相成的。如果想编写一个不错的网站,就应该灵活使用这两种标记。通常,在开发项目时很少使用 <Script> 标记。

使用 <Script> 标记的作用主要有两个:第一个作用是它可以调整输出的先后程序,例如:

```
<html>
<head>
<title>我的主页</title>
</head>
```

```
< body >
< br > 我是第一个字符 < br >
< script language = "jscript" runat = "server" >
response.write "(我是第二个字符")
< /script >
< /body >
< /html >
```

运行结果,如图 4 - 2 所示。

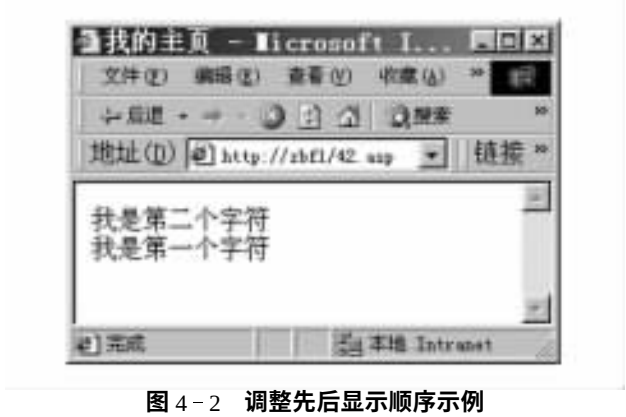

图 4 - 2 调整先后显示顺序示例

实际上,这个文件被 ASP 解释如下:

```
我是第二个字符 < html >
< head >
< title > 我的主页 < /title >
< /head >
< body >
< br > 我是第一个字符 < br >
< /body >
< /html >
```

读者或许会感到很奇怪。实际上,由于 < Script > 中的语句要由服务器执行,因此其中的语句会被首先执行,所以在浏览器上会先显示 '我是第二个字符 ",然后在客户端执行其他语句,显示 '我是第一个字符 "。浏览器上会产生图 4 - 2 所示的结果。

第二个作用是它允许在一个主页中使用多种 Script 脚本,在 4.1.3 节中已经对它作了一定的介绍,在此不再作讲解。

4.2 使用 SSI

SSI 是 Server-side Include 的简称,使用它可以将文件内容以及有关文件信息包含到 HTML 中。当然还可以在 ASP 中使用包含指令。它所包含的主要指令及其功能,如表 4 - 2 所示。

表 4－2 SSI 包含的主要指令及其功能

名 称	功 能
# config	指定返回到客户端浏览器的错误信息、日期及文件大小所使用的格式
# echo	在 HTML 页中插入环境变量的值
# exec	运行一个应用程序或 shell 命令并将输出插入到 HTML 中
# flastmod	把文件修改日期插入到 HTML 中
# fsize	把文件的大小插入到 HTML 中
# include	把文件包含到 HTML 或 ASP 中

4.2.1 # config

config 指令主要用来指定返回到客户端浏览器的错误信息、日期及文件大小所使用的格式。值得注意的是，该指令只能在 HTML 中使用，不可在 ASP 中使用。

参数 Errmsg 用来指定在处理 SSI 指令过程中发生错误时返回客户端的信息。例如可以用

```
<!-- #config errmsg="sorry, you are wrong."-->
```

屏蔽掉详细的调试信息，而把返回信息设定为“sorry, you are wrong.”。

参数 Timefmt 用来设置返回到客户端浏览器的日期格式。在设置时，需要用到其他格式，其名称及含义，如表 4－3 所示。

表 4－3 日期设置格式及其含义

名 称	含 义
%a	一周中某天的缩写（例如：Mon）
%A	一周中某天的全称（例如：Monday）
%b	月份的缩写（例如：Jan）
%B	月份的全称（例如：January）
%c	当地的日期和时间表示
%d	以十进制数字表示一个月中的某一天（1～31）
%H	24 小时格式
%I	12 小时格式
%j	以十进制数字表示一年中的某一天（001～365）
%m	以十进制数字表示的月份（1～12）
%M	以十进制数字表示的分钟（0～59）
%p	当地的上午或下午表示符（例如：PM，AM）
%S	以十进制数字表示的秒（0～59）
%U	以十进制数字表示一年中的某一周，星期日作为一周的开始（0～51）
%w	以十进制数字表示一周中的某一天，星期日作为第一天（1～6）
%W	以十进制数字表示一年中的某一周，星期日作为一周的开始（0～51）
%x	当地的日期表示
%X	当地的时间表示
%y	以十进制数字表示的不带有世纪的年（例如：00）
%Y	以十进制数字表示的带有世纪的年（例如：2000）

续表）

名　称	含　义
%z	时区的全称
%Z	时区的缩写
%%	表示百分号

例如：

```
<!-- #config timefmt="%b;%d;%I;%M;%p"-- >
```

设定了一种日期格式，读者可以根据表 4－3 来设置返回客户端的日期格式。

参数 Sizefmt 用来指定文件的大小，表示单位，用 Abbrev 表示 KB，用 Byte 表示字节。例如：

```
<!-- #config sizefmt="bytes" -- >
```

表示文件大小，用字节表示。

4.2.2　# exec

当需要运行应用程序或 Shell 命令时，可以使用 # exec 指令，执行的应用程序可以是 CGI 程序、ASP 程序或 ISAPI 应用程序。其中应用程序的路径必须是完整的虚拟路径或者 URL。如果应用程序需要参数，那么可以在应用程序名后跟上一个问号“?”和一系列由加号“+”连接起来的参数值。

例如：

```
<!-- #exec /total.asp -- >
```

其中假定在当前目录下存在一文件 total.asp。

4.2.3　# flastmod

用于将指定文件的修改时间插入到 HTML 中。

例如：

```
<!-- #flastmod file="……" -- >
```

其中“……”表示指定文件的全称。

```
<!-- #flastmod virtual="……" -- >
```

其中“……”表示相对于服务器基本目录路径中的文件。

4.2.4　# size

用于将指定文件的大小插入到 HTML 中。其中在默认情况下，文件大小用 KB 来表示，但可以用 # config 中的 sizefmt 选项改变计量单位，参数设置同 # flastmod。

4.2.5　# include

当 SSI 需要在脚本中使用另一个文件的内容时，就需要使用 # include 将相应的文件包含

进来。当服务器处理该文件时,先根据要处理的脚本查找 IIS,并将其对应内容插入到脚本中。这样当服务器查找 IIS 结束后,就会像处理单个 ASP 脚本一样。

在 ASP 中使用 # include 格式:

```
<!-- #include virtual = path -->
```

其中 path 表示相对于服务器基本目录路径中的文件,即虚拟路径。

```
<!-- #include file = ?-->
```

其中"?"表示指定文件的全称或相对于当前目录上的 ASP 文件。

在编写 ASP 文档时,经常将一些常用的函数或过程写入一个或几个 ASP 文件中。这样在以后使用时,可用 # include 语句将相应文件包含进来就可以了。

4.2.6 总结

最后,需要指出的是,除 # include 可以使用在 ASP 页面上外,其他的 SSI 指令只能用在 HTML 页面上,而不能用在 ASP 中。经常用到的指令仅是 # include 和 # config,其他指令一般很少用到。

4.3 ASP 使用的基本脚本语言

ASP 默认的脚本语言是 VBScript,这里主要学习 VBScript 在 ASP 中的运用方法。

4.3.1 循环语句

1. Do……Loop 语句

在此我们仅以"Do While 条件语句……Loop"语句举例。

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
利用循环语句产生三个文本框<br>
<%
dim string(2)
string(0)="姓名"
string(1)="学号"
string(2)="地址"
counter=0
do while counter<3
    response.write(string(counter)&":<input type=text size=20>")
    counter=counter+1
loop
%>
```

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

程序在客户端浏览器上的执行结果，如图 4－3 所示。

图 4－3　循环显示文本框示例

2．For……Nnext 语句

再利用 for……next 语句实现上述目的的例子。

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
利用循环语句产生三个文本框<br>
<%
dim string(2)
string(0)="姓名"
string(1)="学号"
string(2)="地址"
for counter=0 to 2
    response.write(string(counter)&":<input type=text size=20>")
next
%>
</body>
</html>
```

程序执行结果，如图 4－3 所示。

4.3.2　格式化输出

1．格式化日期

格式化日期的函数为 FormatDateTime（参数 1，参数 2），参数 1 表示要显示的日期、时间，参数 2 表示显示格式，其显示格式及示例，如表 4－4 所示。

表 4－4 日期显示格式及示例

显示格式	值	示 例
VBGeneralDate	0	00－12－30 22:39:02
VBLongDate	1	2000 年 12 月 30 日
VBShortDate	2	00－12－30
VBLongTime	3	22:39:02
VBShortTime	4	22:39

下面,我们举一个显示日期的例子：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
for count = 0 to 4
    response.write (formatdatetime (now(),count))
    %><br>
<% next %>
</body>
</html>
```

程序执行结果,如图 4－4 所示。

图 4－4 日期显示格式化示例

2. 格式化数字

格式化数字的函数为 FormatNumber (参数 1,参数 2),其中参数 1 为要显示的数字,而参数 2 为小数位数。

现在举一个例子,程序如下：

```
<html>
```

```
<head>
<title>New Page 1</title>
</head>
<body>
<%
number = 12345.67890
response.write (number&"显示为<br>")
for count = 0 to 5
    response.write (formatnumber (number,count))
    %><br>
<%
next
%>
</body>
</html>
```

程序执行结果,如图 4-5 所示。

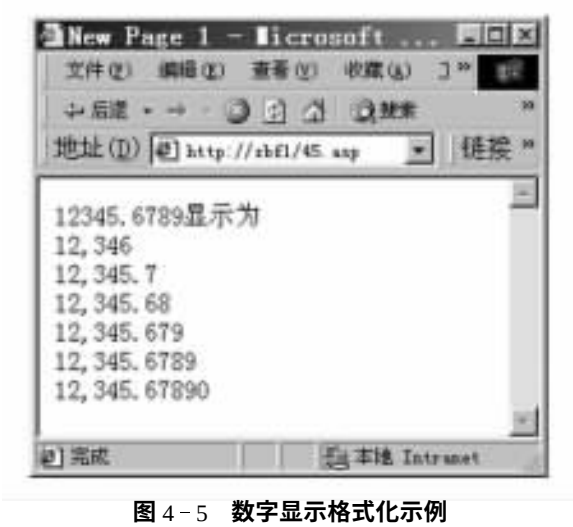

图 4-5 数字显示格式化示例

4.3.3 使用数学函数

在此,我们需要略提一下函数 eval(参数 1),其中参数 1 表示要执行的数字计算串。该函数的功能是计算参数所表示的表达式的值。例如:

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<%
maths = "(23 + 321)/3 + 12"
```

```
response.write (eval (maths))
%>
</body>
</html>
```

程序执行结果，如图 4－6 所示。

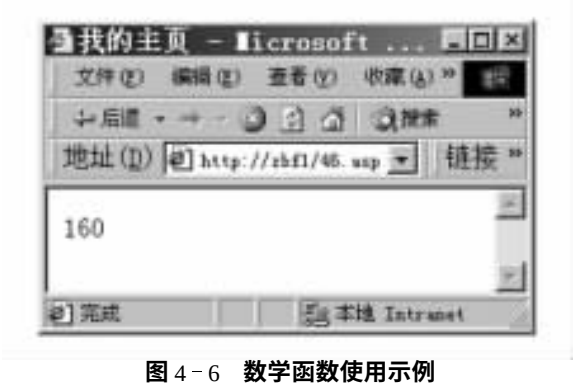

图 4－6 数学函数使用示例

4.3.4 使用 Split 函数

Split 函数的作用主要是将一个字符串赋予一个数组，它把字符串以某个确定字符进行分段。

现在举一个例子，程序如下：

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<%
dim array
sports = "football ;basketball ;volleyball"
array = split (sports," ;")
count = ubound (array)
response.write (sports&"<br>")
for I = 0 to count
    response.write (array (I)&"<br>")
next
%>
</body>
</html>
```

其中程序中所调用的函数 ubound （数组名）返回的值比数组的长度小 1，程序执行结果，如图 4－7所示。

图 4-7 字符串拆分示例

4.3.5 使用 With 语句

With……EndWith 语句的作用是减少代码的重复,优化程序代码。

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<form>
<%
with response
    .write"请输入代号:"
    .write"<input type=text size=20><br>"
    .write"请输入密码:"
    .write"<input type=password size=20><br>"
    .write"<input type=submit value='提交'>"
end with
%>
</form>
</body>
</html>
```

在上述程序中创建了一个文本框、一个密码输入框和一个提交按钮,程序执行结果,如图 4-8 所示。

4.3.6 调用过程

如果被调用的过程与调用语句不在同一个文件中,可以用 SSI 命令集中 # include 指令将被调用过程所在的文件包含进去。

一般可以用 Call 命令调用过程。如果过程是 Function 过程,也可以使用赋值命令调用过程,并且将返回值赋予一变量。下面用一个例子加以说明。

图 4-8 登陆界面

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<%
call show("hello!welcome you!")
%>
<br>从 1 到 10 的累计和为：
<%
'使用 call 命令调用函数
call show("hello!welcome you!")
response.write("<br>"&total(0))
sum=total(0)
response.write("<br>"&sum)

sub show(str)
    response.write(str)
end Sub
function total(n)
    sum=0
    for i=1 to n
        sum=sum+i
    next
    total=sum
end function
%>
</body>
</html>
```

在上述程序中，show（字符串）是一个 Sub 过程，用来将参数字符串显示在客户端浏览器

上 ;total (n)是一个 Function 过程,用来计算从 1 到 n 的累加和。程序执行结果,如图 4－9 所示。

图 4－9　过程使用示例

4.4　习题

1. 基础题

① 使用（　　）标志将 VBScript 代码添加到 HTML 页面中。

② 由于浏览器能够使用多种 Script 语言,所以必须用（　　）指定程序中所使用的 Script 语言种类。

③ 在 ASP 文件中,默认的脚本语言是（　　）。

④ 在一个 ASP 文件中,主要包含哪几方面内容?

⑤ 怎样实现在一个 ASP 文件中使用不同的脚本语言?

⑥ SSI 功能主要由哪些命令来实现?当我们要将一个文件包含到 HTML 页或 ASP 页中,用哪条指令来实现?怎样实现?

⑦ ASP 有哪几个内部对象?它们的基本功能是什么?

2. 操作题

① 编写一个函数文件,在其中编写求阶乘函数。然后再编写一个调用该函数的 ASP 文件,利用循环语句计算从 1 到 100 的阶乘并显示在客户端上,其中首先必须利用 # include 将函数文件包含进去。

② 编写一个 ASP 文件,要求取出系统时间,并用一定形式显示在客户端上,并且根据系统时间判断是上午、中午、下午还是晚上 可以预设对应的时间段)。

第五章　数据动态交换

如今,面向对象技术的概念已深入人心。我们经常使用的程序语言,如 Visual Basic、C++、Power Builder 等都是面向对象的编程语言,ASP 也不例外。在以后各章的学习中,我们将频繁接触到对象的事件、属性和方法的概念。

对象就是具有一定特性且能进行相关操作的实体。有的对象还可以对外部环境的变化(触发事件)作出响应。对象的属性就是对象所具有的特性,对象的方法就是对象能够进行的操作,而对象的事件则是对象对外部环境变化作出的响应。

在第四章中,曾经提到 ASP 中的六大内部对象。其中 Request 和 Response 对象主要用于在服务器端和客户端之间实现数据动态交换。因此本章主要介绍 Request 对象、Response 对象和 Cookie 对象的属性及方法等。

5.1　Request 对象

Request 对象是 ASP 中最有力的对象之一,在服务器端与客户端之间的交互中起着非常重要的作用。实际上,Request 对象的功能是单向的,它只能接受从客户端 Web 页面传来的消息。与之相对的是 Response 对象,它的功能也是单向的,将服务器端的数据传送到客户端。在实现服务器端和客户端之间的数据交互时,就必须将两者结合起来使用。

一般来说,客户端以递交表单的形式作为传递消息的常用手段,而且利用表单的形式也是最可靠的,其中重要的原因是因为它适用于任何浏览器。当然也可以选择使用其他方法,如 ActiveX 控件和 Java Applet 控件等。

利用 Request 对象可以读取客户端提交表单中的数据或 Cookies 中的数据,可以让服务器端获得客户端的以下 HTTP 请求:

① 利用 POST 方法传递的参数。

② 利用 GET 方法送出的请求。

③ Server 变量集合中大量的标准信息。

④ 客户端的 Cookies 信息。

在 Request 对象的集合中主要有 QueryString、Form、Cookies、ServerVariable 等。下面分别对它们进行较为详细的说明。

5.1.1　QueryString 集合

Request 对象中的 QueryString 数据集合可以直接从客户端取得有关提交的信息或 Cookies 中的数据。使用形式如下:

```
request.querystring("数据项名")
```

返回值为递交表单中对应数据项的值。

这里,用一个例子进行说明。客户端表单程序如下:

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<form action="51response.asp" method="get" name="login">
姓名<input type="text" name="nm" size=20><br>
性别<input type="radio" name="sx" value="male">男
<input type="radio" name="sx" value="female">女<br>
密码<input type="password" name="pwd" size=20>
<br><input type="submit" name="sd" value="递交">
<input type="reset" name="sd" value="重置">
</form>
</body>
</html>
```

在程序中,分别利用文本框、单选框和密码填写框实现姓名、性别和密码的输入。这样当访问者访问该主页时,首先显示出来的是表单界面。填写后的表单界面,如图 5-1 所示。

图 5-1 登录表单

现在开始编写服务器端表单处理程序 51response.asp。

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
nm=request.querystring("nm")
sx=request.querystring("sx")
```

```
response.write nm
if sx = "female" then
    response.write "女士"
else
    response.write "先生"
end if
response.write ",你好"
%>
</body>
</html>
```

在程序中，先利用 Request 对象中的 QueryString 集合从客户端取回表单信息中姓名填写文本框 nm 和性别单选框选择值 sx 中的内容，并且分别赋值予两个变量 nm 与 sx。然后对取回的值进行处理，根据不同情况返回不同的结果。单击‘递交’按钮（图 5－1）后，服务器端利用 Response 对象将处理信息送回客户端浏览器，显示结果，如图 5－2 所示。

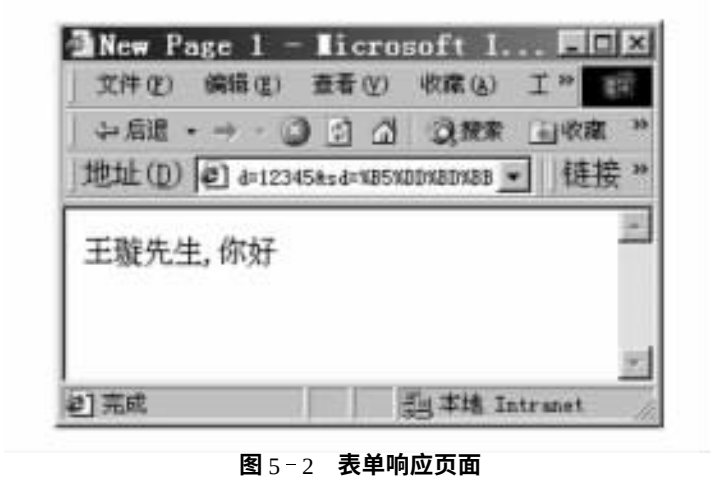

图 5－2　表单响应页面

在上面的例子中使用了 GET 的方法，当访问者在客户端填写完表单并且按下‘递交’按钮后，表单中的数据就发送到 URL 所指的服务器端上。在图 5－2 中可以看到 URL 中的内容为：

```
http://zbf1/response1.asp?nm=%CD%F5%E8%AF&sx=male&pwd=12345&sd=%B5%
    DD%BD%BB
```

表单处理程序名之后有符号“?”，表示它后面的是送到服务器端的数据项值，如姓名 nm、性别 sx 和密码 pwd 等。

我们也可以使用同一个名称传送多个数据项值，并且可以为表单处理带来一定的方便。但这仅限于某些用途，在多数情况还是要用不同的名字为好。当我们需要填写一系列含义相近的数据项时，就可以用这种方法。

例如，我们要在网上设计一个‘爱好’调查页面，那么可以用下面这种方法来实现。

```
<html>
<head>
```

```
<title>我的主页</title>
</head>
<body>
请填写你的爱好：
<form action="52response.asp" method="get" name="login">
爱好 1：<input type="text" name="lv" size=20><br>
爱好 2：<input type="text" name="lv" size=20><br>
爱好 3：<input type="text" name="lv" size=20><br>
<input type="submit" name="sj" value="递交">
<input type="reset" name="sj" value="重置">
</form>
</body>
</html>
```

在程序中，调查方利用文本框作为填写调查内容的方式。在访问者填写结束后，调查表单页面，如图 5－3 所示。

图 5－3　爱好调查表单页面

服务器端的表单处理程序 52response.asp 内容如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
你填写的爱好为：<br>
<%
for each item in request.querystring("lv")
    response.write item&"<br>"
next
```

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

在程序中，利用 Request.QueryString（"lv"）取回访问者填写的信息，并进行处理。这样当表单处理程序执行结束后，服务器端就会返回客户端浏览器上客户所填写的信息，如图 5－4 所示。

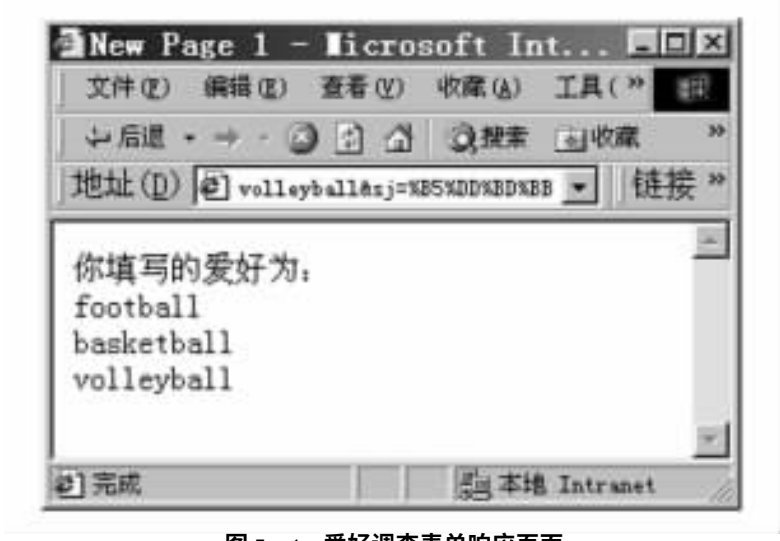

图 5－4　爱好调查表单响应页面

在图 5－4 中，可以看到 URL 中的内容为：

```
http://zbf1/52response.asp?lv=football&
      lv=basketball&lv=volleyball&sj=%B5%DD%BD%BB
```

说明客户端利用一个 "lv" 名字传送了三个文本值，这样在处理时服务器端利用一个 "for each……in ……next" 循环语句进行表单处理，省去了不少麻烦。

另外，对图 5－4 中的表单进行处理时，表单处理程序 response1.asp 还可以用以下语句进行处理：

```
<%
for i=1 to request.querystring("lv").count
    response.write request.querystring("lv")(i)&"<br>"
next
%>
```

在使用 Request 对象中的 QueryString 集合对表单进行处理时，应该根据不同情况灵活使用不同的方法。

5.1.2　Form 集合

现在，我们开始利用 Request 对象中的另一种集合 Form 实现表单的处理，它可以获得客

户端在表单中所填的信息。此时客户端更多地需要利用 POST 方法提交数据，因为使用 GET 方法只能传输较小的数据量，而传送大量的数据时，一般使用 POST 方法。其中使用形式如下：

```
request.form ("数据项名")
```

返回值为递交表单中对应数据项的值。

下面举一个例子进行说明。先编制一个客户端的表单程序 53.htm，分别利用文本框、选择框和文本框实现姓名、性别和地址的输入，表单中还有一个表单 '提交 '按钮 如图 5－5 所示)。

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<form method="post" action="53response.asp" name="form1">
姓名：<input type="text" name="nm" size=20><br>
性别：<select name="sex">
<option value="male">男
<option value="female">女
</select><br>
地址：<input type="text" name="address" size=20><br>
<input type="submit" value="提交">
</form>
</body></html>
```

图 5－5　客户信息表单页面

服务器端的表单处理程序 53response.asp 如下：

```
<html>
<head>
<title>New Page 1</title>
```

```
</head>
<body>
<%
nm = request.form ("nm")
sx = request.form ("sex")
ad = request.form ("address")
response.write nm
if sx = "female" then
    response.write "女士"
else
    response.write "先生"
end if
response.write ",你好"
response.write"<br>你的地址为"&ad
%>
</body>
</html>
```

在服务器端利用 Form 集合从客户端取回信息后，对其填写的信息进行处理，例如根据访问者填写的性别决定称呼：先生还是女士。然后利用 Response 对象把提示信息返回到客户端。此时客户端浏览器上显示的页面，如图 5－6 所示。

图 5－6　对应图 5－5 的表单处理结果页面（之一）

需要指出的是，在 HTML 中的 Form 文本框是不允许空的，否则服务器端的脚本执行将会出现错误信息。我们可以在表单的编制中对文本框的属性进行定义，例如在表单域中对限制有效性的最小长度中进行设置（例如将其置为 1），此时文本框的属性如下：

```
姓名：<！－－ webbot bot ="Validation" S－Display－Name ="name" B－Value－
Required ="TRUE" I－Minimum－Length ="1" －－ > < input type ="text" name ="nm"
```

```
size = 20 > < br >
```

我们也可以在服务器端表单处理程序中进行判断,只需将其返回值与空串进行比较即可,再根据不同的情况进行不同的处理。

当然,可以通过其他方法使用 Form 集合。例如,当我们需要将访问者的填写内容重新显示一遍让访问者确认时,可以利用 for each……next 循环实现。

此时,服务器端的处理程序 53response2.asp 如下 (53.htm 中的 action 也要作相应改动):

```
< html >
< head >
< title > New Page 1 < /title >
< /head >
< body >
< %
for each item in request.form
    response.write "< br >"&item&":"&request.form (tem)
next
% >
< /body >
< /html >
```

在程序中,利用 for each……next 循环将客户端递交表单中的每个数据项名取出,然后在循环体中将数据项名及其对应值显示出来,如图 5-7 所示。

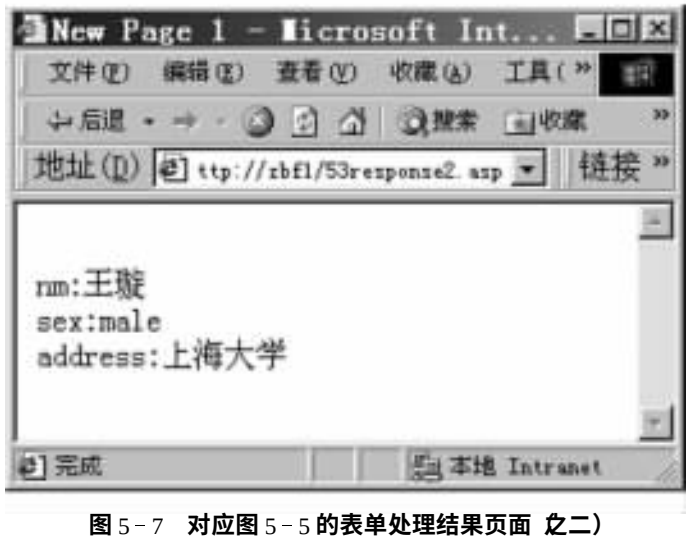

图 5-7 对应图 5-5 的表单处理结果页面 (之二)

也可以利用 for……next 循环使用 Form 集合,程序 53response3.asp 如下 (53.htm 中的 action 也要作相应改动):

```
< html >
< head >
< title > New Page 1 < /title >
```

```
</head>
<body>
<%
for i = 1 to request.form.count
    response.write "<br>"&request.form (i)
next
%>
</body>
</html>
```

在程序中，利用 Request.Form.Count 可以得出客户端表单的数据项数目，然后就可以使用预先知道循环次数的 For……Next 循环，在循环体中再利用 Request.Form (i) 取出该数据项所对应的值。程序执行结果，如图 5-8 所示。

图 5-8　对应图 5-5 的表单处理结果页面（之三）

像 QueryString 集合一样，Form 集合也可以通过每个元素返回多个值，这样对于属性相似的填写项的处理带来很大的方便。在处理对于某项调查中的多项选择时，就可以使用这种方法。例如当我们调查访问者的爱好时，就可编写出如下的表单程序：

```
<html>
<head>
<title>
我的主页
</title>
</head>
<body>
<form method="post" action="54response.asp" name="Form1">
姓名：<input type="text" name="nm" size=20><br>
性别：<select name="sex">
```

```
< option value = "male" > 男
< option value = "female" > 女
< /select > < br >
请选择你的爱好：< br >
< input type = checkbox name = "hobby" value = "football" > 足球 < br >
< input type = checkbox name = "hobby" value = "basketball" > 篮球 < br >
< input type = checkbox name = "hobby" value = "volleyball" > 排球 < br >
< input type = "submit" value = "提交" >
< /form >
< /body >
< /html >
```

在程序中，分别使用文本框、选择框、多选框进行姓名的输入、性别和爱好的选择。运行后的表单界面，如图 5－9 所示。

图 5－9　信息登记及爱好选择的页面

表单处理程序 54response.asp 如下：

```
< html >
< head >
< title >
New Page 1
< /title >
< /head >
< body >
< %
nm = request.form ("nm")
sx = request.form ("sex")
response.write nm
```

```
if sx = "female" then
    response.write "女士"
else
    response.write "先生"
end if
response.write "？你好 < br > 你的爱好是："
for each hobby in request.form ("hobby")
    response.write (hobby&" ")
next
%>
</body >
</html >
```

在服务器端的表单处理程序中，可使用前面所述的 Form 方法。在对多选框进行处理时，利用 For each……Next 循环就可进行方便的处理。表单处理结果，如图 5－10 所示。

图 5－10　对应图 5－9 的表单处理结果页面

本节对 Form 集合的使用方法进行了说明，由于使用方法较多，在实际应用中应该根据不同情况选择方便的方法进行处理。QueryString、Form 集合是 Request 对象中最常用的集合，希望读者能够熟练掌握。

5.1.3　ServerVariable 集合

ServerVariables 集合可以用来提供随 HTTP 请求一起传递的 HTTP 头信息，其中包含系统的环境变量等。客户端的请求和服务器端的响应都包含在 headers 中。

使用方式是：

```
request.servervariables ("环境变量名")
```

其中，使用不同的环境变量名可以得到系统的各种环境变量。表 5－1 列出了主要环境变量名称及其含义。

表 5－1　主要环境变量名称及其含义

名　称	含　义
Auth _Type	当用户试图存取保护类型的文件时,服务器用来验证其权限
Content _Length	客户端提交的正文的长度
Content _Type	客户端提交的正文的类型
Gateway _Interface	服务器端所使用的 CGI 修订版本
Logon _User	用户是否是以 Windows NT 的账户登录
Path _Info	客户端的路径信息 虚拟路径)
Query _String	在一个 HTTP 请求中的查询信息,就是一个 HTTP 请求中所传送的参数值
Remote _Addr	包含发出请求的远程主机的 IP 地址,用这个地址可以知道访问者的初始来源
Remote _Host	发出请求的远程主机 Domain Name 网址
Request _Method	数据请求的方法,对于 HTTP 可以是 GET、POST 或其他方法
Script _Map	给出 URL 的基本部分
Script _Name	执行脚本的虚拟路径,这个变量包含的是当前 ASP 的虚拟路径
Server _Name	该服务器端的 IP 地址或 Domain Name
Server _Port	数据请求所使用的端口
Server _Port _Secure	端口的安全性。如果 Sever _Port 是被保护的,返回字符“1”,否则返回字符“0”
Server _Protocol	服务器端的协议及修订版本
Server _Software	服务器端运行的软件名称及版本号
URL	系统的 URL 路径
HTTP _Referer	当访问者通过其他网站链接到当前页时,通过 HTTP _Referer header 保存用户的来源
HTTP _User _Agent	这个 header 指示了访问者访问本网站时所用的网络浏览器的类型
Path _Translated	用来呈现文件的实际路径
HTTP _UA _Color	浏览器可以显示颜色的数目
HTTP _UA _OS	运行浏览器的操作系统的类型
HTTP _UA _CPU	执行浏览器电脑的 CPU 型号
HTTP _UA _Pixels	浏览器所用的显示器的分辨率

现在,给出使用 ServerVariables 的例子。

1. 将所用的环境变量名称及内容显示出来

实现程序如下：

```
<html>
<head>
<title>
New Page 1
</title>
</head>
```

```
<body>
<table width="300">
<% for each item in request.servervariables %>
    <tr><td width="20%">
    <% =item %></td>
    <td><% = request.servervariables (item) %></td>
    </tr><%
next %>
</table>
</body>
</html>
```

程序执行结果，如图 5－11 **所示。**

图 5－11　**全部环境变量显示页面**

2．**显示特定服务器端环境变量的内容**

当我们需要了解 All_HTTP、Content_Type、Server_Name **等内容时，可用下列程序实现：**

```
<html>
<head>
<title>
New Page 1
</title>
</head>
<body>
```

```
ALL_HTTP:<% = request.servervariables ("ALL_HTTP") %> < BR > < BR >
CONTENT_TYPE:<% = request.servervariables ("CONTENT_TYPE") %> < BR > < BR >
SERVER_NAME:<% = request.servervariables ("SERVER_NAME") %> < BR > < BR >
SERVER_PORT:<% = request.servervariables ("SERVER_PORT") %> < BR > < BR >
REMOTE_HOST:<% = request.servervariables ("REMOTE_HOST") %> < BR > < BR >
HTTP_HOST:<% = request.servervariables ("HTTP_HOST") %>
</body>
</html>
```

上述程序执行的结果,如图 5－12 **所示。**

图 5－12　**特定环境变量显示页面**

5.1.4　对 Header 的授权操作

当有一些主页需要密码保护时,ServerVariables **集合中的** Auth _Type、Auth _User、Logon _User **和** Auth _Password **可以用来获得客户端访问该页面的信息。其中** AUTH _TYPE Header **表示用户进入主页的授权方式,**Auth _User Header **和** Logon _User Header **包含了** Windows NT **用户的用户名信息,**Auth _Password **包含了用户的密码信息。对** Header **的授权操作,这里不再作进一步说明,如果有需要可以查阅有关书籍。**

5.2　Response 对象

Response 对象主要用于服务器端向客户端发送数据，其中可以利用 Request.Write 直接向客户端发送信息，也可以利用 Response.Redirect 进行重新定位客户端到另一个 URL 上去，当然也可以利用 Response.Cookies 来设置 Cookies 的值。因此，将 Request 对象和 Response 对象结合起来，可以使服务器端和客户端的联系更灵活，实现了动态交换的目的，以及从 B/S（Browser/Server）到 C/S（Client/Server）的转变。

5.2.1　Response 对象的属性

Response 对象的属性及其含义，如表 5－2 中。

表 5－2　Response 对象的属性及其含义

属　性	含　义
Buffer	用来设置页面输出时，是否需要缓冲区（值为 true 或 false）
ContentType	用来指定 Response 对象的 HTTP 属性
Expires	用来设置页面保存在客户端浏览器上的时间长度
ExpireeAbsolute	用来刷新被缓存页面的具体时间和日期
CacheControl	用来设置服务器端的输出结果是否由代理服务器保存
Charset	用来设置 ContentType 的表头
Status	用于传递服务器 HTTP Response 的状态
PICS	设置 PICS 标记
IsClientConnected	用来检查客户端是否与服务器端还有连接

1．Buffer 属性

该属性用来设置（或获得）页面输出时是否要用缓冲区，这样可以改变执行结果的输出方法。如果将 Buffer 属性的值设为 True，那么服务器端 Response 的内容会被写入缓冲区中，当脚本处理结束后再显示给访问者；如果将 Buffer 属性的值设为 False，那么在服务器端处理脚本时，服务器端 Response 的内容将顺序地发送给客户端的浏览器。如果不设置 Buffer 值，则 Buffer 属性默认为 False，如以下程序所示：

```
<% response.buffer = true %>
<html>
<head>
<title>我的网站</title>
</head>
<body>
<%
for I = 1 to 100
    response.write "(我是第"&I&"个！<br>")
next
%>
```

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

读者不妨将 Buffer 属性值设为不同的值,试着浏览一下,就会发现不同的显示结果。当将 Buffer 属性值设为 False 时,可以看到在客户端浏览器上自上而下逐条显示服务器端利用 Response 对象传回的字符串;而当将 Buffer 属性值设为 True 时,客户端会将服务器端执行结果同时显示出来。

值得注意的是,当定义 Buffer 属性值时,需要将其定义语句放在服务器端 ASP 脚本输出的前面,通常放在 ASP 脚本的最前面定义(像上述程序一样),不允许在脚本输出后修改 Buffer 属性值。

2. ContentType 属性

当需要设置 Response 的 HTTP 类型,也就是控制送出文件的 MIME 数据类型时,就需要利用 Response 的 ContentType 属性进行设置。通常用到的 MIME 数据类型,如表 5-3 所示。

表 5-3 MIME 数据类型

MIME 类型	文 件 类 型	后缀名
Application/Pdf	PDF 格式(用 Adobe Acrobat 阅读)	.pdf
Application/Rtf	Rich Text Format	.rtf
Image/Gif	GIF 图像	.gif
Image/Jpeg	JPEG 图像	.jpeg
Text/Html	HTML 超文本格式	.html
Text/Plain	Plain 文本	.txt
Video/Mpeg	Mepg video	.mpeg 或 .mpg

使用方式为:

```
<% response.contenttype = "MIME 类型" %>
```

并且像 Buffer 属性一样,必须定义在服务器端 ASP 脚本输出的前面。

3. Expires 属性

该属性可以用来设置 Web 页面保存在客户端浏览器 Cache 上的时间长度(单位为分)。当在客户端选择 Web 页面时,客户端浏览器会检查 Cache 数据内是否有相同的页面,如果有就显示出来。

但由于 ASP 的动态特性会给缓冲区带来影响,因此一般将 Expires 属性设置为 0,就可较容易解决这个问题。使用方式为:

```
<% response.expires = 0 %>
```

这样,当服务器端收到请求时总是刷新客户端页面,原因是服务器端一收到页面就过了期。

4. ExpiresAbsolute 属性

与 Expires 属性比较相近的是 ExpiresAbsolute 属性,它也影响缓存页面的时间期限。但该属性是用来设置缓存中页面保存或者页面到期的绝对时间,而不是设置两次刷新之间的时间数。例如当要将所有客户程序缓存版本页面在 2001 年 1 月 1 日零点零分零秒到期并进行

刷新，那么应该在 ASP 脚本头添加如下语句来实现：

```
<% response.expireabsolute = #Jan 1,2001 00:00:00# %>
```

值得注意的是，根据 HTTP 协议，期限不能超过一年。

5．Status 属性

该属性用来设置当客户端浏览器在浏览过程中出现错误时所要执行的状态值。读者也许知道服务器端返回的状态代码由三位数字组成，客户端可以确定服务器端是如何处理请求的。但实际上，除状态代码外，Status 还返回一个有关状态代码的解释语句。因此在定义 Status 属性时，一般是一个三位数字加上一段解释语句。实现语句如下所示：

```
<% response.status = "101 对不起，你输入错误！" %>
```

一般来说，用五种状态代码，下面作分别解释（其中 X 代表 0～9 内任一数字）。

（1）1XX（Message）

本状态码主要用于实验。

（2）2XX（Success）

本状态码表示 HTTP 请求已经成功，例如状态码 200 表示主页请求被成功接受。

（3）3XX（Redirection）

本状态码表示在接受请求前应该采取进一步的行动，例如状态码 301 表示该主页已经转移到了其他地址，并且浏览器会自动转向新的地址。

（4）4XX（Client Error）

本状态码表示客户端浏览器发出的请求为错误时所返回的状态码，例如状态码 404 表示浏览器请求的主页并不存在。

（5）5XX（Server Error）

本状态码用来表示服务器端出现了问题，例如状态码 503 表示服务器端超时。

6．IsClientConnected 属性

当客户端在浏览 Web 页面时，可能由于某种原因而关闭了该网页，即客户端已不再和服务器端联系。但在服务器端对此却不知，仍然在继续处理 ASP 脚本，这纯粹是一种对系统资源的浪费。在这种的情况下，可以利用 IsClientConnected 属性解决这个问题。当客户端与服务器端处于连接状态时，属性值为 True，否则为 False，这样服务器端可以知道客户端已离开并停止 ASP 脚本的执行。

不过，该属性有一个限制，只有在利用 Response.Write 调用客户端浏览器时才可以使用该属性进行判断。下面举一个例子说明。

```
<html>
<head>
<title>示例</title>
</head>
<body>
<% for I = 1 to 100
    response.write I&"你还在吗？<br>"
    if not (response.isclientconnected) then
```

```
            response.end
        end if
    next
    %>
    </body>
    </html>
```

7. CacheControl 属性

本属性用来设置服务器端是否将 ASP 脚本输出结果暂存到代理服务器上。

现在,为了减少在 Internet 上接受主页的时间,许多人都使用了代理服务器 (大多是免费的)。当该属性值设置为 Private (默认值)时,将会通知代理服务器其返回结果对于访问者来说是私人的,不能设置缓存;当属性值设置为 Public 时,通知代理服务器可以进行缓存。在进行设置时,使用方式为:

```
<% response.cachecontrol = "public" "private" %>
```

8. Charset 属性

该属性用来设置响应文件的语种。当进行属性设置时,使用方式为:

```
<% response.charset (charstring) %>
```

其中 CharString 代表字符串,表示一语种。

例如当我们需要把文件中的语种定义为"BIG5"时,那么需要在 ASP 脚本文件头加上 Response.Charset ("BIG5")语句。如果服务器端 ASP 程序内未包含该属性设置,那么响应的客户端 HTML 文件头为"Content Type:Text/Html"如果包含上该属性设置,那么响应的文件头将变为"Content Type:Text/Html;Charset = BIG5"。

5.2.2 Response 对象的方法

在对 Response 对象的属性说明之后,接下来对 Response 对象的方法进行详细的讲解。Response 对象的方法及其功能,如表 5-4 中所示。

表 5-4 Response 对象的方法及其功能

方 法	功 能
AddHeader	用来设置 HTML 文件的 HTTP 标头
AppendToLog	在 Web Server 记录文件 (Log)中加上访问者数据记录
BinaryWrite	不使用任何字符转换,写入 HTTP 输出
Clear	清除在缓冲区的 HTML 输出数据
End	停止处理任何 ASP 文件,并返回当时的状况
Flush	立即送出 Server 缓冲区中的 HTML 数据
Redirect	指引客户端浏览器至新的 Web 页面
Write	写入字符串到最近的 HTTP 输出

1. AddHeader 方法

Response 对象的 AddHeader 方法用来设置 HTML 文件的 HTTP 标头。如果需要得知关于

HTTP 标头的信息，可以利用 Request. ServerVariables（"MY _HEADER"）方法，其中 MY __HEADER 代表标头变量名。当需要设置标头时，使用方式为：

```
response.addHeader 标头变量名，标头变量值
```

需要指出的是，在服务器端 ASP 脚本中的 Buffer 属性需要设置为 True，也就是将所有输出都存储在服务器端的缓存区中，这样在 ASP 脚本中使用 AddHeader 方法才可以，否则该方法必须写在 ASP 脚本文件头中。如果不按照上述规则，就会出错。

例如在下例中，就会出错。

```
<% response.buffer = false %>
<html>
<head>
<title>我的网站</title>
</head>
<body>
<% response.addheader "myheader","王璇" %>
……
</body>
</html>
```

对上例进行修改，可使用两种方法：首先，可以将 Response. Buffer 属性值改为 True；其次，可以将 Response. AddHeader 方法写在 ASP 文件头中，也就是"< html > "标记前。

2．AppendToLog 方法

该方法主要用于在 Web Server 记录文件（Log）尾端加上用户数据记录，以方便对访问者进行跟踪。当访问者有违反规定的动作时，可以利用记录进行查询。当然该方法的使用并不局限于这些，也可以添加其他记录。

在需要登记某个访问者的访问次数时，需要在 Web 服务器端日志上添加一项访问者的名字。对此可以利用下面的方法实现：

```
<% nm = request.cookies("visitername")
   response.appendtolog nm
%>
```

在本例中，先利用 Request 对象中的 Cookies 属性取出访问者的名字，然后再利用 Response 对象中的 AppendToLog 方法在 Web 日志末尾添加项，即访问者的名字。可以用它记录访问者的属性，例如访问次数等。

3．BinaryWrite 方法

BinaryWrite 方法可以在不使用任何字符转换的情况下，把信息写入到最近的 HTTP 输出上，该方法对于输出二进制数据是十分方便的。

使用方式为：

```
<% response.binarywrite 变量名 %>
```

其中变量名表示要转换的数值内容。

4．Clear 方法

Clear 方法用来清除在 Web Server 缓冲区内的页面。

使用方式为：

```
<% response.clear %>
```

值得注意的是，如果 Response 对象的 Buffer 属性被设置为 True，那么 Clear 方法将会清除所有在缓冲区内的页面；否则，如果 Response 对象的 Buffer 属性被设置为 False，由于未使用缓冲区，采用 Clear 方法将会产生一个运行模式的错误。所以在使用该方法时，一定要注意 Response 对象的 Buffer 属性的设置值。

5．End 方法

当在 ASP 脚本中使用 End 方法时，ASP 服务器端将会停止处理任何 ASP 文件，并返回当时的状况。

使用方式为：

```
<% response.end %>
```

如果在 Response 对象的 Buffer 属性被设置为 True 时使用该方法，那么服务器端立即将缓存区中的页面发送到客户端上，并清除缓冲区。因此，在使用 End 方法前应先利用 Clear 方法将缓冲区清空。使用方式如下：

```
<% response.clear
response.end %>
```

6．Flush 方法

Flush 方法主要用来让服务器端立刻送出缓冲区内的 Web 页面。本方法的使用前提是 Response 对象的 Buffer 属性被设置为 True，否则将会报错。

使用方式为：

```
<% response.flush %>
```

本方法与 End 方法不同的是，在调用了 Flush 方法后，服务器端仍然会继续处理页面，而不是像 End 方法那样会停止处理所有页面。因此，这两种方法适合在不同的场合使用。

7．Redirect 方法

Redirect 方法主要用来指引客户端浏览器到新的 Web 页面。它会类似于在第二章中提及的“< a href = 网页地址 > …… < /a > ”的超链接动作。

使用方式为：

```
<% response.redirect"网页名" %>
```

其中网页名表示要切换到的网页文件名称。使用本方法较灵活和方便，并且可以根据用户的不同选择而定位到不同的页面上去，当然还可以在处理程序中加入其他的处理动作，因此相对来说本方法的使用范围更广。

例如，在一个网站的聊天室建立中，必然在消息输入表单中出现‘离开聊天室’的按钮，其处理事件就是要从聊天室画面切换到主页面，在该事件程序 left.asp 中就使用了本方法。

```
<%
```

```
nm = request.cookies("custom")("name2")          '取出访问者昵称
application.lock                                 '禁止其他访问者改变其内容
flag = 1                                         '设标记位,以识别访问者的存放位置
member = application("member")
for i = 0 to application("num") - 1              '在访问者存放数组中删除该访问者
    if flag = 0 then
        member(i - 1) = member(i)
    end if
    if nm = member(i) then
        flag = 0
    end if
next
application("member") = member
if flag = 0 then
    application("num") = application("num") - 1  '访问者数目减 1
end if
if application("counter") > 19 then              '使循环语句数目为 20
    application("counter") = 0
end if
color = application("color")
message = application("message")
message(application("counter")) = "< font color = "&color&" > "&nm&"轻轻地走了……"
     &"< /font > "
application("message") = message
application("counter") = application("counter") + 1
count = application("count")
if count < 20 then
    application("count") = application("count") + 1
end if
application.unlock                               '允许其他访问者进行修改
response.redirect"heartmemo.htm"                 '切换到主页面
%>
< html >
< head >
< meta http-equiv = "Content-Type" content = "text/html;charset = gb2312" >
< meta name = "GENERATOR" content = "Microsoft FrontPage 4.0" >
< meta name = "ProgId" content = "FrontPage.Editor.Document" >
< title > New Page 1 < /title >
< /head >
< body >
< /body >
< /html >
```

8．Write 方法

Write 方法在前面的介绍中已经多次提到，它可以把变量值、Cookie 值等发送到用户端的页面上，其功能是非常强大的。

在 Write 方法中，可以使用各种 HTML 标记，可以将字符串的限制标记包含进去，从而使客户端的页面上以特定的方式显示出来。

使用方式为：

```
response.write (字符串 1"& "字符串 2"……)
```

其中，字符串 1、字符串 2 等表示要显示的符号串或显示格式标记，当然在上述格式中也可以是一个字符串或者用连接符 & 连接起来的多个字符串。

例如，在聊天室的显示处理程序 show. asp 中，就多次使用到了该方法。程序如下：

```
<% response.buffer = True %>
<%
server1 = request.servervariables ("Server _name")
path = request.servervariables ("Script _name")
self = "http ://"&server1&path
%>
< html >
< head >
< meta http-equiv = "refresh" content = "10 ;<% = self %>" >
< title > New Page 1 < /title >
< base target = "_self" >
< /head >
< body bgcolor = " # C0C0C0" >
< font size = "2" >
<%
message = application ("message")
color = application ("color")
count = application ("count")
if count = 20 then
    pos = application ("counter") - 1
    state = pos
    do
        response.write ("< br > "&" < font color = "&request.cookies
        ("custom") ("color") &" > "&message (pos)&" < /font > ")        '显示语句信息
        if pos = 0 then
            pos = 19
        else
            pos = pos - 1
        end if
```

```
    loop Until pos = state
else
    for i = application ("counter") - 1 to 0 step -1
        response.write ("<br>"&"<font color="&request.cookies ("custom") (
            color") &">"&message ()&"</font>")          '显示语句信息
    next
end if
%>
</font>
</body>
</html>
```

5.3　Cookies 集合

Cookies 集合是一种服务器端发送到客户端上的文本串句柄，并且保存在客户端的硬盘上，用来在 Web 站点间持久地保持必要的数据。例如某一网站具有聊天室的功能，当访问者以某一登记身份进入主网站后，又决定进入聊天室时就不应该再次输入用户名及密码等，这些信息可以利用 Cookies 集合保存起来，当在必要的时候取出。

Request 和 Response 对象都有一组 Cookies，利用 Request 对象可以取出 Cookies 集合中的值，利用 Response 对象可以创建 Cookies 集合中的某属性。

服务器端利用客户端一个或多个指定的文件支持 Cookie，这必然会带来一定的副作用。读者不妨找一找 Cookies 目录，会发现在该目录下面有许多 Cookies 文件，不过这些不会占用太大的空间，更不会使某些人利用这些文件窃取信息。目前一些浏览器提供了屏蔽 Cookie 的功能，或在一些网站上提供了屏蔽 Cookie 的方法，因此会对 ASP 程序的执行带来一定的影响。

5.3.1　Cookies 的属性

表 5－5 列出了 Cookies 集合的属性。

表 5－5　Cookies 集合的属性

属　性	含　义	示　例
ExpiresAbsolute	可以赋予一个日期值，该日期之后不可使用	resposne.cookies ("buy"). expiresabsolute="1/10/2001"
Domain	用来定义 Cookie 要传送的惟一域	response.cookies ("buy"). domain="……"
HasKeys	用来判断是否包含关键字	if request.cookies ("buy").haskeys …… end if

续表)

属 性	含 义	示 例
Path	用来指定 Cookie 要发给请求的路径	response.cookies ("buy").path = 路径名
Secure	用来设置 Cookie 是否能被客户读取	response.cookies ("buy").secure = true

5.3.2 Cookies 的建立和引用

首先,在 Cookie 的建立中,可以直接利用 Response 对象在客户端建立 Cookie 值。使用方式为:

```
response.cookies ("cookie 名") ("子键名") = 初始值
```

当然要注意的是,由于 Cookie 是作为 HTTP 传输的头信息的一部分发送给客户的,因此该语句要出现在 ASP 程序中标记 "< HTML > "的前面,并且还要写在其他类型的 Response 语句的前面,这样才可以保证 ASP 程序的正确性。

例如当某访问者以某一登记身份进入一个网站时,该网站服务器端就可以利用上述方式将该用户用关信息保存起来。在 ASP 程序 enterclub.asp 中的验证访问者身份的开头部分语句为:

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
set rs = server.createobject ("adodb.recordset")
bname2 = trim (request.form ("name2"))
psw = trim (request.form ("key"))
if bname2 = "" then                          '处理用户名未输入时的情况
    response.cookies ("custom") ("name2") = ""
    response.write ("你的用户名未输入!请重新输入!")
else
    if psw = "" then                         '处理用户密码未输入时的情况
        response.cookies ("custom") ("name2") = ""
        response.write ("你的口令未输入!请重新输入!")
    else                                     '检查访问者的合法性
        find1 = "select * from member where name2 = '"&bname2
        find1 = find1&"' and password = '"&psw&"'"
        set rs = mydata.execute (find1)
        if rs.eof and rs.bof then  '访问者输入有误
            response.cookies ("custom") ("name2") = ""
            response.write ("对不起,你的输入有误,请检查你的用户名和口令!")
        else                                 '访问者合法
```

```
                bname2 = trim (request.form ("name2"))
                '利用 Cookies 集合保存用户名等信息
                response.cookies ("custom") ("name2") = bname2
                response.write ("< marquee direction = right > 欢迎你进入俱乐部！
                    < /marquee > ")
%>
                < html >
                < head >
                < title >
                欢迎进入会员俱乐部
                < /title >
                < /head >
                < body bgcolor = " # FF6666" >
                < font face = "华文行楷" > 登录成功 < /font >
                < div align = "center" >
                < img src = "山林 2.jpg" width = "70 % " height = "70 % " >
<%
        end if
        rs.close
        set rs = nothing
    end if
end if
mydata.close
set mydata = nothing
%>
< /div >
< /body >
< /html >
```

在 ASP 程序中引用某个 Cookie 值时，就需要使用 Request 对象中的 Cookies 集合。使用方式为：

```
<% 变量名 = request.cookies ("cookie 名") [("子键名")] %>
```

在 3.2 节所举的示例程序 left.asp 中，利用语句

```
nm = request.cookies ("custom") ("name2")
```

取出访问者昵称，并存放到变量 nm 中。

另外，也可以利用

```
<% = request.cookies ("cookie 名") [("子键名")] %>
```

直接进行显示，也可以利用其他属性进行有关处理。

5.3.3 Cookies 的释放

由于 Cookies 文件要保存在客户端的硬盘上，要占用一定量的空间，因此要释放不需要的 Cookie 变量。经常采取的方法是为其 ExpiresAbsolute 属性设置一个过时时间。ExpiresAbsolute 属性 或 Expires 属性）的使用方法已经在 5.3.1 节中有所描述。

有一种常用的使用方式为：

```
<%
response.cookies（cookie 名″）.expires = Date - 365
%>
```

5.4 习题

1. 基础题

① （　　）对象用于服务器端从客户端提取数据；（　　）对象用于服务器端向客户端发送数据。

② 当要利用 Request 对象中的 QueryString 集合从客户端提取数据时，提取表单中需要使用的方法为（　　）；当要利用 Form 集合从客户端提取数据时，提取表单中需要使用的方法为（　　）。

③ 当我们要指定 Response 对象的 HTTP 类型时，需要用（　　）属性来指定。

④ Cookie 被保存在（　　）的硬盘上。

⑤ Cookies 集合的主要作用是什么？

2. 操作题

请利用 Request、Response 对象在客户端和服务器端之间进行交互。编写一个表单让访问者填写姓名、口令等信息，然后利用 Cookies 集合在表单处理程序中将其内容保存起来，同时在表单处理页面上加入显示按钮，当该按钮按下后，利用 Cookies 集合取出保存信息并显示出来。

第六章　处理 HTML 表单

在网站的编写中，实际上许多内容的填写都是利用表单的形式提交的。因此，应该熟练掌握有关表单的处理方式。

6.1　处理表单数据

在表单填写方式中，常见的有文本框、文本区、单选框、复选框等。

6.1.1　文本框和文本区

当需要访问者填写有关的文本信息时，就需要用到文本框和文本区。文本框标记为 Text，文本区标记为 TextArea。在使用方法上，两者是基本上相同的，只是接受信息的多少不同而已。

下面，我们举一个关于文本框和文本区的例子进行讲解。表单程序 61.htm 如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<form method="POST" action="61response.asp" name=Form1>
    你的用户名：<input type="text" name="nm" size="15"><br>
    简介：<textarea rows="2" name="intro" cols="20"></textarea>
    <br><input type="submit" name="st" value="提交">
    <input type="reset" name="cl" value="清除">
</form>
</body>
</html>
```

在上述程序中，利用文本框实现用户名的输入，然后利用文本区实现用户简介的输入，如图 6-1 所示。关于它们的使用格式可以从第二章的有关内容或其他地方获得。

对应的表单处理程序 61response.asp 如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
```

```
<% name = trim (request.form ("nm"))
intro = trim (request.form ("intro"))
response.write ("亲爱的"&name&"朋友,"&"你的简介为 :"&" < br > "&intro)
%> < /body >
< /html >
```

图 6－1　文本框和文本区使用示例

在以上程序中,利用 Request 对象的 Form 集合从客户端取回有关信息,当然也可以利用 Request 对象的 QueryString 集合实现该任务。然后利用 Trim 方法进行消除空格的处理。利用 Response 对象的 Write 方法将访问者信息显示在客户端。表单处理结果,如图 6－2 所示。

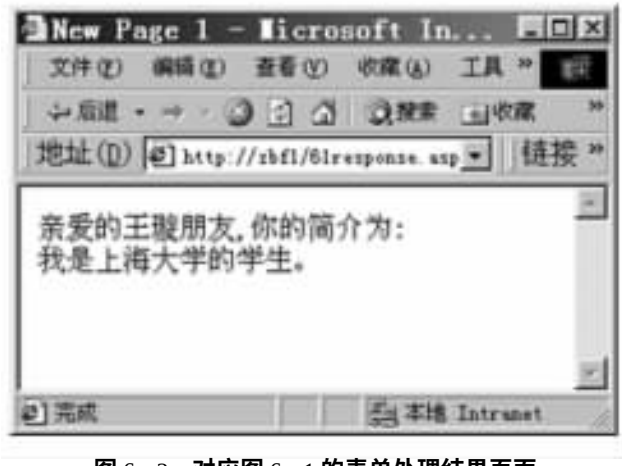

图 6－2　对应图 6－1 的表单处理结果页面

6.1.2　单选框和复选框

在询问访问者有关信息时,也经常让用户在一些给定选项中进行选择。此时就需要用到单选框和复选框等有关内容。单选框标记为 Radio,复选框标记为 Checkbox。它们的使用方法基本相似,只是单选框只能选择一项,而复选框同时可以选择多项。

同样来看一个例子，表单程序 62.htm 如下：

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<form method="post" action="62response.asp" name="Form1">
姓名：<input type="text" name="nm" size=20><br>
性别：<input type=radio name="sex" value="male">男
      <input type=radio value="female">女<br>
请选择你的爱好：<br>
<input type=checkbox name="hobby" value="football">足球<br>
<input type=checkbox name="hobby" value="basketball">篮球<br>
<input type=checkbox name="hobby" value="volleyball">排球<br>
<input type="submit" value="提交">
</form>
</body>
</html>
```

在上述程序中，利用单选框进行性别的选择，利用复选框进行访问者爱好的选择。表单显示界面，如图 6－3 所示。

图 6－3　单选框和复选框使用示例

对应的表单处理程序 62response.asp 如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
```

```
<body>
<%
nm = trim (request.form ("nm"))
sex = request.form ("sex")
hobby = request.form ("hobby")
if sex = "male" then
    nm2 = "先生"
else
    nm2 = "女士"
end if
response.write ("亲爱的"&nm&nm2&",你的喜好为"&hobby)
%>
</body>
</html>
```

在上述程序中,利用 Request 对象中的 Form 集合将图 6-3 中的表单内容取回来,并对它进行一系列处理后将有关信息显示在客户端的浏览器上。

当访问者填写有关信息并提交后,服务器端发送到客户端浏览器上的处理结果,如图 6-4 所示。

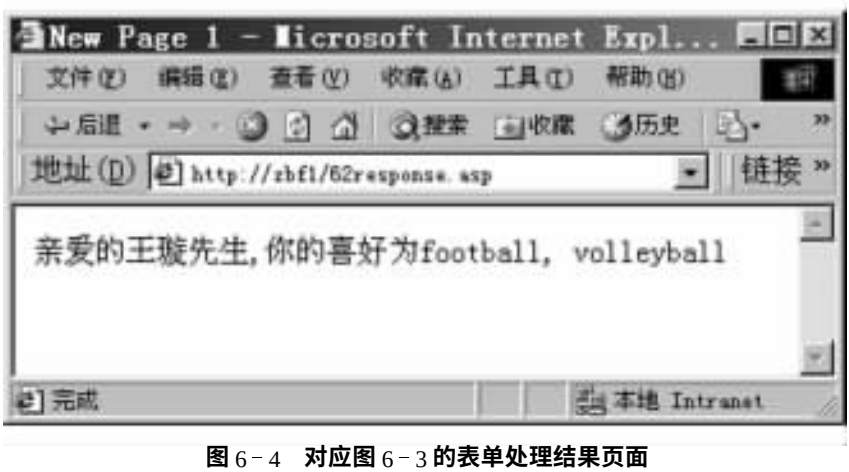

图 6-4 对应图 6-3 的表单处理结果页面

6.1.3 选择列表

利用单选框可以解决表单中的一些填写问题,但当选项较多时,必然要占去页面上的较大空间。选择列表并不占用页面上的很大空间,但却能够容纳较多信息。

选择列表的使用方式为:

```
<select name = 列表名>
    <option name = 选项名 value = 选项对应值> …… </option>
    <option name = 选项名 value = 选项对应值> …… </option>
    ……
</select>
```

其中 option 是用来设定列表元素的。

下面举一个示例程序 63.htm：

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<form method="post" action="63response.asp" name="Form1">
姓名：<input type="text" name="nm" size=20><br>
性别：<select name="sex">
<option value="male">男</option>
<option value="female">女</option>
</select>
<br>请选择你的爱好：
<select name="hobby">
<option name="football" value="足球">足 球</option>
<option name="basketball" value="篮球">篮 球</option>
<option name="volleyball" value="排球">排 球</option>
</select><br>
<input type="submit" name="tj" value="提交">
</form>
</body>
</html>
```

在上述表单程序中，用选择列表分别实现了访问者性别和爱好的填写。但要注意的是，在性别的填写中，对于选项并没有命名；不过在爱好的填写中，却对每一项选项都有命名。这两种用法并不矛盾，都可以使用。表单界面显示，如图 6－5 所示。

对应的表单处理程序 63response.asp 如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
nm=trim(request.form("nm"))
sex=request.form("sex")
hobby=request.form("hobby")
if sex="male" then
    nm2="先生"
else
    nm2="女士"
end if
```

```
response.write "亲爱的"&nm&nm2)
for each hobby in request.form ("hobby")
    response.write (",你的喜好为"&hobby)
next
%>
</body>
</html>
```

图 6-5　选择列表使用示例

在上述程序中,用两种不同的方法分别对性别选择列表和爱好选择列表进行处理。对于性别列表框,直接利用 Request 对象的 Form 集合来提取;对于爱好列表框,则利用 For Each……Next 循环进行处理。其显示结果与图 6-4 不同,因为爱好列表框也只能选择一项,显示结果,如图 6-5 所示。

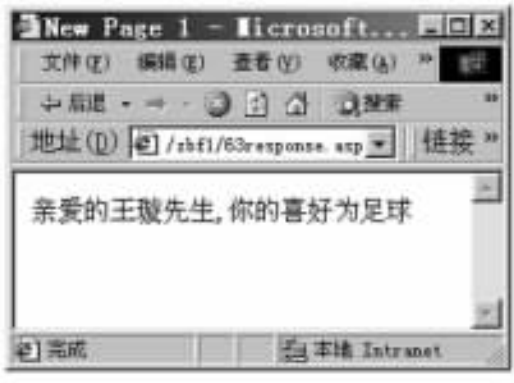

图 6-6　对应图 6-5 的表单处理结果页面

6.2　表单应用实例

表单数据的提取主要依赖于 Request 对象中的 Form 集合或者 QueryString 集合。在此举

一个例子进行系统说明。表单程序 regis1.asp 如下：

```
<html>
<head>
<title>
加入会员
</title>
</head>
<h2 align=center><font face="华文行楷">加入会员</font></h2>
<font face="华文行楷" size=4 color="#008000">请输入你的个人资料 带*的文本框为必填框）：</font>
<form method="POST" action="regsuccess.asp" name="FrontPage_Form1"
onsubmit="return FrontPage_Form1_Validator(this)">
<font face="华文行楷" color="#008000">
            
            
          
姓名：        
<!--webbot bot="Validation" S-Display-Name="用户名" B-Value-Required
="TRUE"Minimum-Length="1"-->
<input type="text" name=name1 size="20">*
<p>            
            
          
昵称：        
<!--webbot bot="Validation" S-Display-Name="昵称" B-Value-Required="
TRUE"I-Minimum-Length="1"-->
<input type="text" name="name2" size="20" value="">*
</p>
<p>            
            
          
电子邮箱：<input type="text" name="email" size="20" value="">
</p>
<p>            
            
          
工作地址：<input type="text" name="address" size="20" value="">
</p>
<p>            
            
          
密码：            
```

```
<！- - webbot bot = "Validation" S - Display - Name = "密码" B - Value - Required = "
        TRUE"I - Minimum - Length = "5" I - Maximum - Length = "10" - - >
<input type = "password" name = "psw1" size = "20" maxlength = "10" value = "" > * </
        p >
<p >   ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
          ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
          ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
确认密码 ;  ;  ;  ;
<！- - webbot bot = "Validation"S - Display - Name = "确定密码" B - Value - Required
        = "TRUE" I - Minimum - Length = "5"I - Maximum - Length = "10" - - >
<input type = "password" name = "psw2" size = "20" maxlength = "10" value = "" > * </p
        > </font >
<p >   ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
          ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
<input type = "submit" value = "提交" name = "B1" style = "color： # CC3300 ; font-
      family ：华文行楷 ; border-style ： ridge ; border-color ： # 008000" >
  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
<input type = "reset" value = "全部重写" name = "B2" style = "font-family ：华文行楷 ;
        color ：# 993300 ;border-style ：ridge ;border-color ：# 008000" > </p >
</form >
</body >
</html >
```

在表单程序中，“ ;”标记用于表示空格，用来调整表单显示形式 ;另外，表单项的一些属性用于确保表单项的合法填写。当在客户端运行本 ASP 程序并进行填写后，客户端浏览器上的显示界面，如图 6 - 7 所示。

表单处理程序 regsuccess. asp 如下：

```
<html >
<head >
<title >
登记成功
</title >
</head >
<%
set mydata = server.createobject ("adodb.connection")
set rs = server.createobject ("adodb.recordset")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
mydata.begintrans
bname = trim (request.form ("name1"))
bname2 = trim (request.form ("name2"))
```

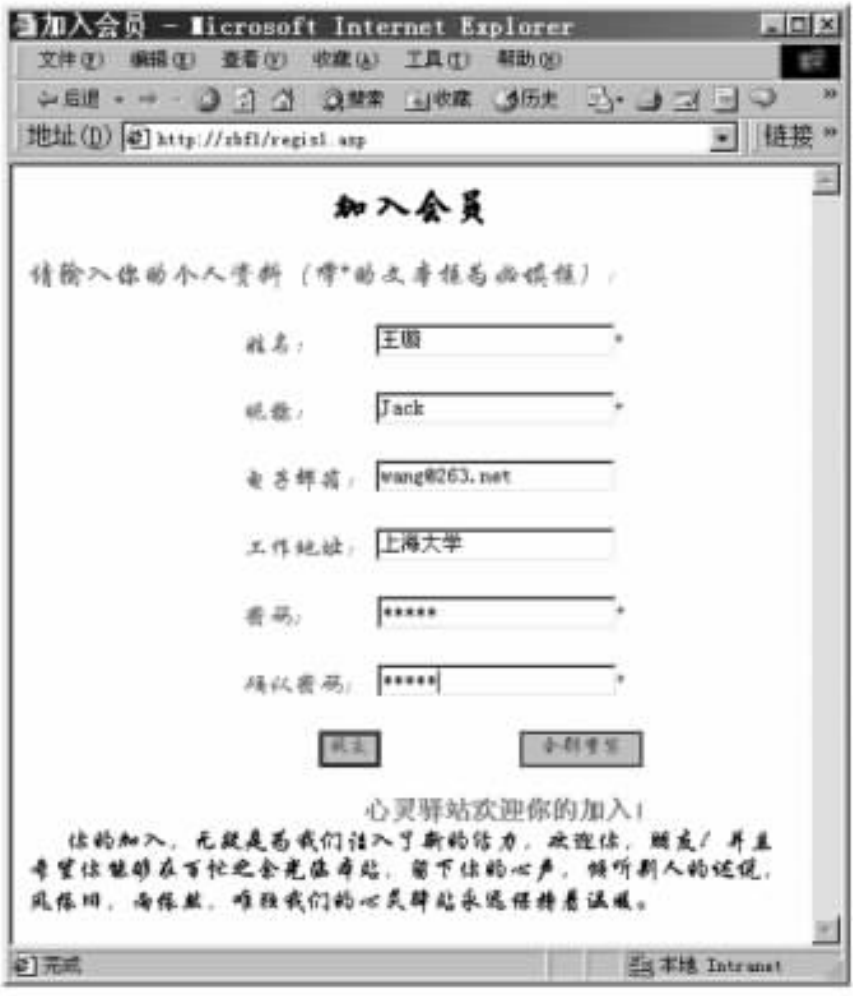

图 6-7　“心灵驿站”会员登记表单

```
bemail = trim (request.form ("email"))
baddress = trim (request.form ("address"))
bpsw1 = trim (request.form ("psw1"))
bpsw2 = trim (request.form ("psw2"))
if (bemail = "") then
    bemail = "    "
end if
if (baddress = "") then
    baddress = "    "
end if
if (bname = "") or (bname2 = "") then
    response.write ("对不起,你还未输入有关必要信息,请返回 !")
else
    if (bpsw1 = bpsw2) then
        find1 = "select * from member where name2 = '"&bname2&"'"
```

```
        set rs = mydata.execute (sql1)
        if rs.eof and rs.bof then
            wsd = "insert into member (name,name2,email,address,password) "
            wsd = wsd&"values ('"&bname&"','"& bname2 &"','"& bemail
    &"','"& baddress &"','"& bpsw1 &"')"
            mydata.execute (wsd)
            response.write "亲爱的"&bname&",祝贺你登记成功！并且希望你能够常回家
        看看。")
%>
            <div align = "center" >
            <img src = "花 1.jpg" align = "center" width = "250" height = "320" >
            </div >
<%
        else
            response.write "对不起,该昵称已存在,请更换！")
        end if
        rs.close
        set rs = nothing
    else
        response.write "你的确认密码与密码不同,请重新确认你的密码！")
    end if
end if
mydata.committrans
mydata.close
set mydata = nothing
%>
<p align = "center" >
<a href = "http://zbf1/regis1.asp" > 返回登记界面 </a >
<a href = "http://zbf1/heartmemo.htm" > 返回主界面 </a >
</p >
</body >
</html >
```

在表单处理程序中，trim（）函数用于将所输字符串中的空格删除掉。在程序中，利用 Request 对象的 Form 集合将访问者所填写信息取回来。然后对其填写进行合法性检查后，将其添加到用户数据库中。客户端将填写结果提交到服务器端，服务器端进行处理后的客户端显示界面，如图 6－8 所示。

6.3 习题

1. 基础题

① 当在客户端进行简单文本信息的输入时，常用到的表单组件为（　　）。

图 6-8　“心灵驿站”会员登记表单处理结果页面

② 当要求访问者在给定的选项中作出多项适合的选择时,常用的表单组件为(　　)。

③ 如编写某网站时要求访问者在给定选择项中作出某一项选择,并所占页面空间尽量小,则选择的表单组件为(　　)。

2. 操作题

编写一份网上测试问卷,要求访问者利用文本框填写用户名、用户地址等内容,利用选择列表填写性别,再考虑其他有关信息。问卷内容包括五道单项选择题和三道多项选择题。当访问者填写结束并提交后,取出有关信息进行处理,并返回问卷得分。

第七章 Application、Server 和 Session 对象

Application、Server 和 Session 对象在网站的建设中会经常用到，它们会使网站的功能变得更加强大。

7.1 创建 ASP 应用程序

在网站的编写中，许多网页之间都有许多公用的消息互相传播，而不仅仅是一组孤立的 Web 页面。例如当编制一个网上商店时，我们应该记录下用户选择购买的商品信息以便于结算，也就是说，在商品出售页面和商品结算页面之间要有许多共享信息。

7.1.1 Global.asa 文件

在安装 PWS 后，在存放 ASP 程序的目录下有一个 Global.asa 文件，这个文件对于网站的建设来说是非常重要的。在下列条件下要访问该文件：

① Web 服务器启动后，当一组应用程序中的任何一个 ASP 文件提出第一个 HTTP 请求时，服务器端就会读该文件。

② 不具有 Session 的客户向服务器端请求 ASP 文件时，也要访问该文件。

在该文件中，主要存放如下内容：

① Application 或者 Session 的开始事件（Application_OnStart、Session_OnStart）。

② Application 或者 Session 的结束事件（Application_OnEnd、Session_OnEnd）。

③ < object > 标记。

在记事本中打开这个文件，可以看到两个子程序：Application_onStart 和 Application_OnEnd，它们经常用于聊天室的应用中。在聊天室页面中，所有的访问者可以看到所有的公共信息，其中就需要用到 Global.asa 这个文件。

我们可用 < object > 标记来建立应用程序变量。例如在 Global.asa 文件中，包含如下有关代码：

```
< object runat = server scope = session id = myinfo progid = "MSWC.MyInfo" >
< /object >
```

有关 Global.asa 的详细用法，在以后的章节中还会进一步讲解。

7.1.2 Application 对象的简介

Application 对象是一个应用程序级的对象，用来在所有用户之间传递共享信息，当然也可以在 Web 页面运行期间持久保持数据。

下面我们列出有关 Application 对象的一些特性：

① 在 Application 内部共享数据，可以让多个访问者访问。

② 一个对象可以在整个 Application 中共享。

③ 一个 Application 包含事件，可以触发一些 Application 脚本。

④ 当某一个人的 Application 出现错误时，一般不会影响他人。

⑤ 服务器端可以停止一个 Application，而不会影响其他的应用。

一个网站并不只是拥有一个 Application。可以针对一些特殊的应用创建 Application。

1. Application 对象的属性

实际上，Application 对象并没有内置属性，但是开发者可以自己设置它的属性。

使用方式为：

```
<%
application("属性名") = 初始值
%>
```

这样，当我们创建了 Application 对象的一个属性后，在整个应用程序运行期间它都会存在，它的值也可以让所有合法用户访问。有时可以通过该对象计算访问网站的人数，如果保存访问人数的变量 visit 已经创建，那么在访问者进入网站的 ASP 程序中需要增加下述语句：

```
<%
application("visit") = application("visit") + 1
%>
```

上述语句有它的弊端，当访问者刷新访问页面时，服务器端则又一次增加访问次数。因此该语句被应用时，还需作其他处理。

在访问上述 Application 对象的有关属性时，还可以使用下述方式：

```
<%
application.contents("visit") = 0
%>
```

它的使用效果与下述语句相同：

```
<%
application("visit") = 0
%>
```

读者或许会有一点不理解，我们已经讲过 Application 对象没有内置属性，那么 Contents 又是怎么一回事呢？实际上，大多数 Application 变量（Application 对象的属性）都保存在 Contents 集合中。也就是说，一旦创建了新的 Application 变量，实际上就是在 Contents 集合中添加了一项。因此，也就出现了上述访问 Application 变量的方法。

在需要显示所有的 Application 变量时，可以使用 Contents 集合。

```
<%
count = 0
for each var in application.contents
```

```
    count = count + 1
    response.write "第" &count&"个 Application 变量是" & application.contents
(var))
    response.write "< br >"
next
%>
```

2. Application 对象的方法

在 Application 对象中,经常用到的方法有 lock 方法和 unlock 方法。读者试想一下 如果很多人访问一个网站,有可能多个访问者同时进入,那么当每个人同时执行了 Application 变量 visit 加 1 的操作后,返回的结果只是 Application 变量 visit 增加了一次。为了避免出现这样的情况,需要使用 lock 方法和 unlock 方法。

使用 lock 方法可以防止其他访问者同时修改 Application 变量,这样也就保持了数据的一致性和完整性。因此当访问者在修改 Application 变量前,需要使用 lock 方法将 Applicating 变量锁定。当访问者对 Application 变量修改后,要对 Application 变量解锁,即可使用 unlock 方法。可以看出,lock 方法和 unlock 方法是成对出现的,而且它们的使用顺序也不可颠倒。

使用方式为:

```
<%
application.lock
application("visit") = application("visit") + 1
application.unlock
%>
```

当然这些方法不只局限于对访问人数的变量修改上,在对其他变量进行修改时,也可以使用这种方法。

3. Application 对象的事件

在 Application 对象中,一般有两个事件:Application_OnStart()和 Application_OnEnd()。前者在应用程序启动时触发,后者在应用程序结束时触发。这两个事件一般被定义在 Global.asa 文件中。

7.1.3 聊天室页面的建立

目前,一般网站都具有聊天室这个内容,本节简单叙述建立一个聊天室的方法。由于本节所讲述的聊天室是包含在某主网站中的,因而需要对它进行一些处理才可使用。

在主框架文件 freetalk.asp 中,主要定义聊天室的页面框架,程序如下:

```
<%
nm2 = request.cookies("custom")("name2")
if nm2 = "" then
    response.write "对不起,请你加入会员俱乐部再进入本聊天室 !"
else
    set mydata = server.createobject("adodb.connection")
    mydata.connectionstring = "dsn = mydata;uid = wly;pwd = pwd"
    mydata.open
```

```
    modifystr = "update member set atnet = true where name2 = '"&nm2&"'"
    mydata.execute modifystr
    mydata.close
    set mydata = nothing
%>
<% nm = request.cookies("custom")("name2")
    flag = 1
    member = application("member")
    for i = 0 to application("num") - 1
        if nm = member(i) then
            flag = 0
        end if
    next
    if application("num") >= 20 then
          response.write("对不起！本聊天室已满,请稍候再进入。")
    else
        if flag = 1 then
            application.lock
            member = application("member")
            member(application("num")) = request.cookies("custom")("name2")
            application("member") = member
            application("num") = application("num") + 1
            color = request.form("color")
            response.cookies("custom")("color") = color
            application("color") = color
            if application("counter") > 19 then
                application("counter") = 0
            end if
            message = application("message")
            message(application("counter")) = "<font color = "&color&">"&nm&"快
                乐地跑来了！</font>"
            application("message") = message
            application("counter") = application("counter") + 1
            count = application("count")
            if count < 20 then
                application("count") = application("count") + 1
            end if
            application.unlock
        end if
%>
        <html>
        <head>
        <meta http-equiv = "Content-Type" content = "text/html;charset =
            gb2312">
```

```
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>心灵驿站聊天室</title>
</head>
<frameset rows="50,*,100">
<frame src="list1.asp" target="_self">
<frameset cols="75%,*">
<frame src="show.asp" target="_self" scrolling="auto" noresize>
<frame src="member.asp">
</frameset>
<frame src="message.asp">
</frameset>
</html>
<%
    end if
end if
%>
```

在上述程序中，首先利用 Request 对象中的 Cookies 集合将访问者的用户名取出作为聊天室中的用户名，利用用户数据库进行相关处理。然后利用 Application 对象进行有关信息的处理，例如访问人数的累加，聊天室信息的记录等。在对 Application 变量进行修改时，首先使用 lock 方法将 Application 变量锁定，禁止其他访问者修改，在修改结束后，使用 unlock 方法对 Application 变量解锁。在主页面中，主要有四个基本框架：主题框架、聊天显示页面、在线人员页面和对话输入页面。它们分别利用 list1.asp、show.asp、member.asp 和 message.asp 实现。该程序显示页面，如图 7-1 所示。

建立聊天室，不可避免地要用到 Global.asa 文件，并利用该文件存储一些应用程序的共享信息。例如在上述聊天室的建立中，对 Global.asa 文件所作的修改如下：

```
<script language=vbscript runat=server>
sub application_onstart
dim message(0)
dim member(0)
application("message")=message
application("counter")=0
application("count")=0
application("member")=member
application("num")=0
application("color")=black
end sub
</script>
```

在上述文件中，利用 Application 对象在 Application_OnStart 中声明了一系列变量。

在聊天室的建立中，首先用到的主题框架文件 list1.asp 如下所示：

图 7-1 “心灵驿站聊天室”页面

```
<html>
<head>
<title>聊天室</title>
<base target="_self">
</head>
<body topmargin="0" leftmargin="0" bgcolor="#CC6699">
<h2 align="center"><font color="#804040">心灵驿站聊天室</font></h2>
</body>
</html>
```

而在聊天显示页面 show.asp 中,利用 META 标记定义页面刷新的时间间隔,从而使页面得以重写。

```
<% response.buffer=true %>
<%
server1=request.servervariables("Server_name")
path=request.servervariables("Script_name")
self="http://"&server1&path
%>
```

```
<html>
<head>
<meta http-equiv="refresh" content="10;<%=self%>">
<title>New Page 1</title>
<base target="_self">
</head>
<body bgcolor="#C0C0C0">
<font size="2">
<%
message=application("message")
color=application("color")
count=application("count")
if count=20 then
    pos=application("counter")-1
    state=pos
    do
        response.write("<br>"&"<font color="&request.cookies("custom")("
            color")&">"&message(pos)&"</font>")
        if pos=0 then
            pos=19
        else
            pos=pos-1
        end if
    loop Until pos=state
else
    for i=application("counter")-1 to 0 step -1
        response.write("<br>"&"<font color="&request.cookies("custom")("
            color")&">"&message(i)&"</font>")
    next
end if
%>
</font>
</body>
</html>
```

而在线人员页面 member.asp 主要用来列举在线人员的信息。

```
<html>
<head>
<meta http-equiv="refresh" content="10;<%=self%>">
<title>New Page 1</title>
</head>
<body bgcolor="#FF6666">
```

在线人数 <%=application("num")%> **人**

```
<div align=center>
<table>
<%
member=application("member")
for i=0 to application("num")-1
%>
    <tr><td>
    <% =member(i) %>
    </td></tr>
<%
next
%>
</table>
</div>
</body>
</html>
```

对话输入页面 message.asp 用来输入访问者的谈话,程序如下:

```
<%
if not request.form("message")="" then
    application.lock
    if application("counter")>19 then
        application("counter")=0
    end if
    message=application("message")
    tonm=trim(request.form("nm1"))
    if tonm="" then
        message(application("counter"))=request.cookies("custom")("name2")&"对
            所有人"&request.form("way")&"说:"&request.form("message")
    else
        message(application("counter"))=request.cookies("custom")("name2")&"
            对"&request.form("nm1")&request.form("way")&"说:"&request.form
            ("message")
    end if
    application("message")=message
    application("counter")=application("counter")+1
    count=application("count")
    if count<20 then
        application("count")=count+1
    end if
    application.unlock
end if
%>
```

```
<html>
<head>
<title>New Page 1</title>
</head>
<body bgcolor="#008080">
<font size=2>
<form method="post" action="message.asp" onsubmit="return FrontPage_ Form1
      _Validator(this)" name="FrontPage_Form1">
<%=request.cookies("custom")("name2")%>
对<input name="nm1" type="text" size="2">
讲话方式<select name="way">
<option value="微笑着">微笑着
<option value="开怀大笑着">开怀大笑着
<option value="冷笑着">冷笑着
<option value="宛然一笑">宛然一笑
<option value="阴转多云">阴转多云
<option value="嚎啕大哭">嚎啕大哭
<option value="凄然泪下">凄然泪下
<option value="埋怨着">埋怨着
</select>说：<!--webbot bot="Validation" B-Value-Required="TRUE"
I-Minimum-Length="1" -->
<input name="message" type="text" size=30>
<input type="submit" value="发　送">
</form>
</font>
<div align="center">
<form method="post" action="left.asp" target="_top">
<input type="submit" name="left" value="离开聊天室">
</form>
</div>
</body>
</html>
```

这四个 ASP 程序与 freetalk.asp 程序一起实现聊天室的功能。读者在上述程序的基础上，可以利用自己学过的知识进一步对它进行美化，建立一个个性化的聊天室。

7.2 Server 对象

Server 对象在 ASP 中被认为是一个非常重要的对象，当一个网站需要实现某些高级功能时，经常需要用到 Server 对象。

7.2.1 Server 对象的属性

Server 对象的一个属性 ScriptTimeOut 可以指定一个脚本文件执行的最长时间期限。换

句话说,如果在设置时间限度内脚本还没有执行完毕,则将中止并显示超时错误;但如果一个 Server 组件还正在执行,那么该项属性的设置将无效。另外,要注意的是,该属性的单位是“秒”,默认值为 90 秒。

当不希望主页由于在 90 秒内没有执行结束而停止执行时,可以把 Server 对象的 ScriptTimeOut 属性设定成所需要的等待时间。

使用方式为:

```
<%
server.scripttimeout = 200
%>
```

其中 200 秒是希望设置的限制时间。

当然,要利用该属性将限制时间减少,同样也可采用上述方法。

需要指出的是,在使用该属性修改限制时间时,需要将上述内容说明在 ASP 文件的开头,并出现在“< HTML > ”的前面。

7.2.2 Server 对象的方法

1. HTMLEncode 方法

如果希望在某一页面上显示一段 HTML 源代码,就需要用到该方法。要在某一页面上显示“I come from ShanXi province < br > < p > ”的字样,该怎样实现呢?先看一下下面这段代码:

```
<html>
<head>
<title>message</title>
</head>
<body>
<%
response.write "I come from ShanXi province<br><p>"
%>
</body>
</html>
```

读者可以自己实验一下,当这段代码执行后,并没有出现期望的结果。出现在客户端上的只是下述结果:

I come from ShanXi province

原因是什么呢?其实,在执行 Response 对象的 Write 方法,服务器端遇到有关 HTML 的标记符时,服务器端总是要在解释后再在客户端显示,其结果使得“< br > ”和“< p > ”在客户端显示时变成了两个控制符。这时,就可以利用 HTMLEncode 方法实现。解决此问题的 ASP 文件 71.asp 如下:

```
<html>
<head>
<title>我的主页</title>
```

```
</head>
<body>
<%
response.write "I come from ShanXi province<br><p>"
response.write server.htmlencode "I come from ShanXi province<br><p>"
%>
</body>
</html>
```

在该文件中利用 Server.HTMLEncode ("……")显示 HTML 源代码。当在客户端执行该文件后的显示结果,如图 7-2 所示。

图 7-2 HTMLEncode 使用示例

2. URLEncode 方法

该方法有点类似 HTMLEncode 方法,它将一个指定的字符串按 URL 的格式显示出来。在字符串显示在客户端时,不会显示空格或其他特殊字符。

程序示例如下:

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<%
response.write "I come from ShanXi province<br><p>"
response.write server.urlencode "I come from ShanXi province<br><p>"
%>
</body>
</html>
```

上述 ASP 程序的执行结果,如图 7-3 所示。

3. MapPath 方法

该方法主要用于返回指定文件的相对路径或物理路径。

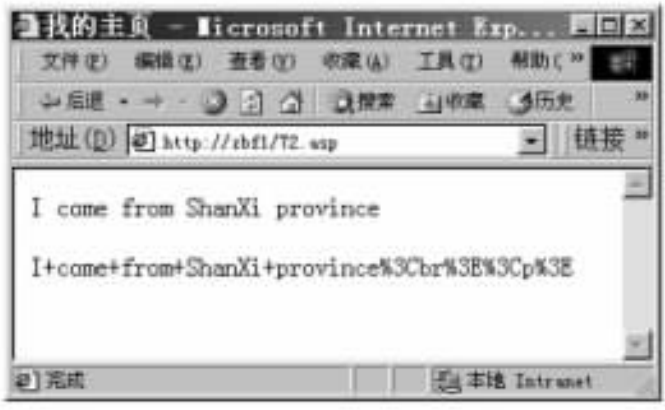

图 7－3　URLEncode 使用示例

使用方式为：

```
<%
server.mappath (文件名)
%>
```

下面举一个显示文件 73.asp 所在目录的例子，文件内容如下：

```
<html>
<head>
<title>我的主页</title>
</head>
<body>
<%
response.write "文件 73.asp 所在目录为："
response.write server.mappath "73.asp"
%>
</body>
</html>
```

在文件中利用 Server 对象的 MapPath 方法返回 73.asp 所在目录，并利用 Response 对象的 Write 方法显示在客户端上。程序执行结果，如图 7－4 所示。

因此，当访问者需要知道某个文件所在目录时，就可以使用该方法。

4．CreateObject 方法

在 Server 对象中，最常用的方法就是 CreateObject 方法。实际上，在前几个章节的叙述中也用到了该方法，例如在聊天室的建立中，就使用下述方法创建了一个数据库对象：

```
<%
set mydata = server.createobject "adodb.connection"
%>
```

CreateObject 方法用于创建已经注册到服务器上的 ActiveX 组件，这样可以通过使用 ActiveX 组件来扩展 ASP 的能力。例如在数据库访问、文件创建中都用到了该方法。

图 7－4　MapPath 使用示例

创建某对象实例的使用方式为：

```
<%
set 实例名 = server.createobject ("组件名")
%>
```

当创建对象后，如果不再需要，那么可以利用下述方法释放所占用的资源。使用方式如下：

```
<%
Set 实例名 = nothing
%>
```

需要指出的是，利用 CreateObject 方法建立对象实例时，不能利用它创建内部对象的实例，例如下述语句是不合法的：

```
set w1 = server.createobject ("Request")
set w2 = server.createobject ("Response")
set w3 = server.createobject ("Server")
set w4 = server.createobject ("Session")
set w5 = server.createobject ("Application")
```

7.3　Session 对象

Session 对象主要用来存储访问者的用户信息，它的存在期限为从到达该网站到离开为止这段时间，每个访问者都会得到一个 Session。

例如，当访问者进入网上商店购买货物时，需要将他的购买信息保存起来，但又不可让其他访问者看到。在这种情况下，就可以通过 Session 对象来实现。Session 对象可以在页面切换时继续保存信息，也可以供这个访问者在 Web 应用的所有页面中共享数据。

Session 对象与 Application 对象有点相似，不同点是，Session 对象不可以在两个访问者之间共享信息，而 Application 对象却可以在访问该网站的不同用户间共享信息。

7.3.1 Session 对象的属性

1. SessionID 属性

当访问者访问某网站时，会分配一个不同的标识符，可以利用 SessionID 属性返回这个标识符。

使用方式为：

```
<% = session.sessionid %>
```

或者

```
<%
response.write session.sessionid
%>
```

现在举一个例子，创建访问者的 Session 对象并利用该属性返回 Session 属性的标识符，程序示例如下：

```
<%
if isobject (session ("cart")) then
    set dictcart = session ("cart")
else
        '利用 Server 对象的 createobject 方法建立 Session 实例
    set dictcart = server.createobject ("scripting.dictionary")
end if
%>
<html>
<head>
<title>我的主页</title>
</head>
<body>
<%
response.write ("你的 Session 编号为：")
%>
<% = session.sessionID %>
</body>
</html>
```

上述程序的执行结果，如图 7－5 所示。

2. TimeOut 属性

该属性定义了访问者 Session 对象的时限。它的时间单位不是秒，而是分。它的作用是当访问者在 TimeOut 规定的时间内没有刷新时，Session 对象就会中止。在默认情况下，该属性的值为 20 分钟。

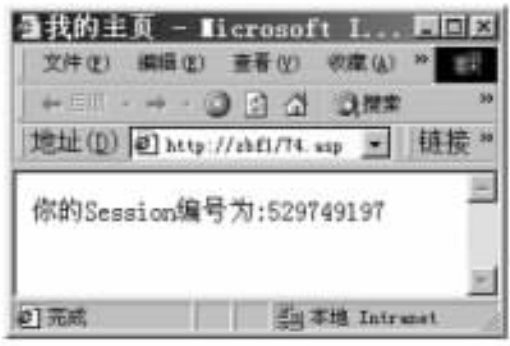

图 7－5　Session 使用示例

使用方式为：

```
<%
session.timeout = 30
%>
```

3．Session 实例

Session 对象与 Application 对象一样，可以由开发人员自己定义它的属性（Session 变量）。它的定义方法为：

```
<%
session("Session 实例名") = 初始值
%>
```

在定义了 Session 对象的属性后，属性数据就可以被访问者使用。

使用方法类似于 Application 对象：

```
<%
response.write session("Session 实例名")
%>
```

大多数 Session 变量（Session 对象的属性）都保存在 Contents 集合中。也就是说，在创建了新的 Session 变量时，实际上就是在 Contents 集合中添加了一项。因此也就出现了下述访问 Session 变量的方法：

```
<%
response.write session.contents("Session 实例名")
%>
```

当需要将所有保存在 Session 对象中的 Contents 集合内的属性全部显示出来时，可以利用下述方法：

```
<%
i = 1
for each var in session.contents
```

```
        response.write "<br>第"&i&"个 Session 变量是"&response.contents(var))
        i = i + 1
    next
    %>
```

当需要将其他对象直接保存在 Session 对象中时,就可以利用下述方法:

```
    <%
    set session("Session 实例名") = server.createobject "组件名")
    %>
```

当上述方法建立后,就可以直接调用该 Session 变量的属性和方法。不过还是不能建立 Request、Response、Session 等内置对象的实例。

7.3.2 Session 对象的方法

在 Session 对象中,用到的大概只有一个方法:Abandon 方法。该方法主要用于释放用户 Session 对象所占用的资源。

使用方式为:

```
    <%
    session.abandon
    %>
```

由于在访问者离开本网站后,其 Session 对象所占用的资源就会自动释放。所以,该方法一般用不到。

7.3.3 Session 对象的事件

如同 Application 对象一样,在 Session 对象中一般拥有两个事件:Session_OnStart 和 Session_OnEnd,其有关内容在此不再介绍。

7.4 习题

1. 基础题

① Server 对象的 ScriptTimeOut 属性用来指定(　　),该属性的单位是(　　),它的默认值为(　　)。

② 在 Server 对象中,利用(　　)方法显示 HTML 源文件。

③ 在 Application 对象中,其事件分别为(　　)。

④ 在 Application 对象中,所定义的绝大部分变量都存放在(　　)中。

⑤ 当我们需要在用户访问网站期间保存用户的个人信息,但又不可让其他访问者查阅时,需要用到的对象为(　　)。

⑥ 当我们需要保存用户之间的共享信息,让访问网站的用户查阅时,需要用到的对象为(　　)。

2．操作题

① 利用 Server 对象实现访问者检索文件路径的功能。要求当访问者输入所要查询的文件名时，服务器端可以给出文件的物理路径。

② 利用 Application 对象实现网站访问人数计数器的功能。

③ 编写一简单的网上商店，用 Session 对象存放用户的购买信息，然后利用这些信息给用户结账。

第八章　ASP 组件的应用

在 ASP 文件的编制中，可以通过使用服务器的组件进一步扩展 ASP 的能力。服务器组件实际上是运行在服务器上的 DLL 文件，该文件由 ASP 页面调用。这样以 Web 页面为交互对象，可以通过 Web 页面读入用户的输入，经服务器处理后再将处理结果返回到 Web 页面上。

在 ASP 中有它的内置组件，当然也可以在其他网站上下载第三方组件，或者利用 VB、VC、Delphi 等语言开发自己的组件。下面我们分别描述各类组件的运用。

8.1　ASP 的内置组件

ASP 的内置组件名称及其功能，如表 8－1 所示。

表 8－1　ASP 的内置组件名称及其功能

名　　称	功　　能
广告轮显组件（AdRotator）	使用独立数据文件的方式，构建广告 Web 页面
浏览器兼容组件（BrowerCap Object）	判别客户端的浏览器类型或使用设置，以便送出正确的 Web 页面
数据库存取组件（Data Access Component）	提供存取数据库的最好方法
文件超链接组件（Content Linking）	建立 Web 站点的索引，连接 Web 页面
文件存取组件（File Access Component）	提供文件的输入、输出方法

上述组件，其实有一些在前面几章的学习中已经提到，例如数据库存取组件，在前面所涉及到数据库的程序示例中，我们就曾经利用它们将访问者的登记信息存储在数据库中。相信读者在学习本章后，会在 ASP 水平上有更大的提高。

8.1.1　Ad Rotator 组件

广告轮显组件可以帮助我们制作精彩的广告 Web 页面，并以此吸引访问者的再次点击，并且它可以增加页面的可变性，达到宣传的目的。

1. Ad Rotator 组件的属性

在 Ad Rotator 组件中，共有三个属性：

（1）Border 属性

该属性主要用来设置广告图片的边界宽度。

使用方式为：

```
<%
set ad = server.createobject ("MSWC.adrotator")
ad.bordersize (1)          '设置广告图片的边界宽度位两个像素
%>
```

(2) Clickable 属性

该属性主要用来指定广告是否提供超链接属性。设置值为 True、False，默认值为 True。

使用方式为：

```
<%
set ad = server.createobject ("MSWC.adrotator")
ad.clickable ("true")
%>
```

(3) Targetframe 属性

该属性主要用来指定超链接后所要浏览的 Web 页面显示方式。其参数可以为 _TOP、_NEW、_CHILD、_SELF、_PARENT、_BLANK 等。

使用方式为：

```
<%
set ad = server.createobject ("MSWC.adrotator")
ad.targetframe ("SELF")
%>
```

2. Ad Rotator 组件的方法

在 Ad Rotator 组件中，只有一个 GetAdvertisement ()方法，用来取得广告信息文件。

使用方式为：

```
<%
set ad = server.createobject ("MSWC.adrotator")
ad.getadvertisement ("ad.txt")
%>
```

这里，"ad.txt"是一个广告信息文件，它虽然是一个纯文本文件，但它的格式是一定的。因此当我们要修改显示内容时，只需修改所对应的广告信息文件。

3. 广告信息文件的格式

使用方式为：

```
Redirect    处理点击广告时的超链接地址；
Width       广告图片的宽度，默认值为 440；
Height      广告图片的高度，默认值为 60；
Border      边框的宽度。
*
广告的图片名
广告图片的链接 URL
广告的备注
广告出现概率
```

广告信息文件 ad.txt 如下：

```
Redirect /81respond.asp
Width 200
height 30
Border 2
*
logo11.gif
http://www.shu.edu.cn
上海大学站点
100
```

其中，广告信息文件的编写要注意如下几点：

① 前面五行有关描述不可省略。

② 有关广告显示描述都保存在“*”后面。

③ 每一项广告描述都必须包含四项。

4. 利用 Ad Rotator 组件的示例

如果我们希望将上海大学站点和搜狐站点进行宣传，并且两者的宣传显示概率各为 50%，那么我们先来编写有关广告信息文件 ad.txt，其内容如下：

```
Redirect /81respond.asp
Width 200
height 50
Border 2
*
logo11.gif
http://www.shu.edu.cn
上海大学站点
50
图标 1.gif
http://www.sohu.com
搜狐站点
50
```

在该文件编写结束后，再来编写对应的 ASP 文件 81.asp，其内容如下：

```
<html>
<head>
<title>
adfdf
</title>
</head>
<body>
<center>
<%
```

```
set ad = server.createobject ("MSWC.AdRotator")
%>
<% = ad.getadvertisement ("ad.txt") %>
</body>
</html>
```

当然还有一 ASP **文件** 81respond.asp**,其内容如下：**

```
<% response.redirect (request.querystring ("url")) %>
```

该文件执行结果,如图 8－1 **所示。单击该图片可以链接到** http：//www.sohu.com。

图 8－1　**搜狐站点宣传页面**

单击 '刷新 '按钮后的显示画面,如图 8－2 **所示。单击该图片可以链接到** http：//www.shu.edu.cn。

图 8－2　**上海大学站点宣传页面**

8.1.2　浏览器兼容组件

浏览器兼容组件可以认为是一个很方便的工具,它可以了解客户端的浏览器配置。当客户端连接到服务器端时,会自动送出一个用户信息头给服务器。在该信息头中,包含了客户端的浏览器种类及版本信息。而浏览器兼容组件会将该表头与 Browscap.ini 进行比较。假如找到相互匹配的浏览器,浏览器组件会假定客户端浏览器所有性质都与该文件记录的一致;假如找不到与 Browscap.ini 相匹配的信息,那么浏览器兼容组件将会获得 Browscap.ini 内设定的浏览器信息,并将其作为客户端浏览器的相关信息。

构成浏览器兼容组件的成员有 Browscap.dll 和 Browscap.ini 文件。其中 Browscap.dll 文件是浏览器兼容组件的动态链接数据库,Browscap.ini 文件包含很多的浏览器信息,用来映射客户端浏览器的 HTTP 信息头。

1. Browser 组件的功能

在客户端的计算机内找到文件 Browscap.ini,由于它是一个纯文本文件,可以用写字板浏览其内容。

详细内容如下:

```
;Last update 16-Jun-97

;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;Microsoft Browsers ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;IE 1.x
[IE 1.5]
browser=IE
version=1.5
majorver=1
minorver=5
frames=FALSE
tables=TRUE
cookies=TRUE
backgroundsounds=FALSE
vbscript=FALSE
javascript=FALSE
javaapplets=FALSE
beta=False
Win16=False
```

Platform 属性在上述文件中没有给出,浏览器属性的名称及其含义,如表 8-2 所示。

表 8－2　浏览器属性的名称及其含义

名　称	含　义
Browser	浏览器类型
Version	浏览器当前版本
Majorver	浏览器的主版本
Minorver	浏览器的辅版本
Frames	指示浏览器是否支持框架功能
Tables	指示浏览器是否支持表格
Cookies	指示浏览器是否支持 Cookies
BackGroundSounds	指示浏览器是否支持 < bgsound > 标记
VbScript	指示浏览器是否支持 VbScript
JavaScript	指示浏览器是否支持 JavaScript
JavaApplets	指示浏览器是否支持 JavaApplets
ActiveX Controls	指示浏览器是否支持 ActiveX Controls
Beta	指示浏览器是否为测试版
Platform	检测目前用户的操作平台
Win16	检测用户的视窗系列是 16 位还是 32 位

2．Browser 组件的应用

可以说，有关 Browser 组件的属性实际上也就是表 8－2 中所列出的项。可以通过调用 Browser 组件获得有关客户端浏览器的属性。

使用方式为：

```
<%
set browser 实例名 = server.createobject ("MSWC.BrowserType")
%>
```

例如，当我们需要知道有关客户端浏览器的信息时，就可以使用以下的 ASP 程序：

```
<html>
<head>
<title>
浏览器组件
</title>
</head>
<body>
<%
set brow = server.createobject ("MSWC.BrowserType")
%>
<br>你的操作平台为：
<% = brow.platform %>
<br>你的浏览器类型为：
<% = brow.browser %>
<br>你的浏览器版本为：
```

```
<% = brow.version %>
<br> 你的浏览器是否支持分屏：<！-- 分屏表示多页面窗口 -- >
<% = brow.frames %>
<br> 你的浏览器是否支持 Cookies：
<% = brow.cookies %>
<br> 你的浏览器是否支持表格：
<% = brow.tables %>
<br> 你的浏览器是否支持 bgsound 标记：
<% = brow.backgroundsounds %>
<br> 你的浏览器是否支持 VBScript：
<% = brow.vbscript %>
<br> 你的浏览器是否支持 JavaScript：
<% = brow.javascript %>
</body>
</html>
```

程序执行后的显示结果，如图 8－3 所示。

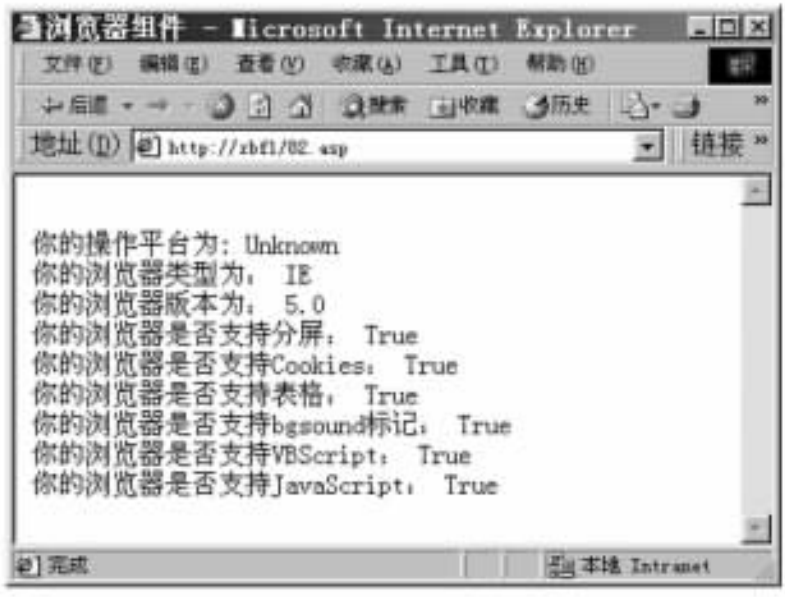

图 8－3　‘浏览器组件’显示页面

由于在 browscap.ini 中没有关于操作平台的描述，因此操作平台属性的返回值为 Unknown。

当需要根据有关浏览器的某个属性值的不同而作出不同的处理时，可以利用该组件实现这个要求。例如当需要在客户端建立 Cookies 文件，但又不知客户端是否支持 Cookies 功能，因此需要根据浏览器组件 Cookies 属性的不同作出不同的处理。

程序如下：

```
<%
set brow = server.createobject ("MSWC.BrowserType")
```

```
if brow.cookies then
    response.cookies ("count") = 10
else
    response.write ("对不起,不支持 Cookies")
end if
%>
<html>
<head>
<title>浏览器组件
</title>
</head>
<body>
</body>
</html>
```

在上述程序中,首先建立了浏览器组件的一个实例,然后根据 Cookies 属性值的不同,确定是否建立 Cookies。这只不过是该组件的一个简单应用而已,读者可以根据实际需要编写出能在各种浏览器类型的不同设置下都能正确显示的页面。

8.1.3 文件超链接组件

该组件主要用来管理 URL 网址,并可以帮助我们构建一个 Web 站点的索引。构建该组件的成员文件为 nextlink.dll 文件和 url * 文件。前者是文件超链接组件的动态链接库文件,后者存放要显示的所有 url 数据。

在编写有关文件超链接组件的 ASP 文件时,还要编写有关 URL 的文本文件。在这方面,该组件或许有点像 AdRotator 组件。下面我们详细地描述有关该组件的内容。

1. 建立文件超链接组件的实例

使用方式为:

```
<%
set 文件超链接组件实例名 = server.createobject ("MSWC.Nextlink")
%>
```

2. URL 数据文件

在该数据文件中,基本上每一行表示一个 Web 站点的索引。它的行格式如下:

```
Web 站点 (按下 Tab 键)站点描述
```

在本节中需要用到的 URL 数据文件 myurl.txt 的内容如下:

```
http://www.shu.edu.cn  上海大学
http://www.sohu.com    搜狐站点
http://www.5460.net    同学录
```

要特别指出的是,Web 站点和站点描述之间不是空格,而是制表符 Tab 键。

3. 文件超链接组件的方法

表 8-3 为文件超链接组件的方法及其功能。

表 8－3 文件超链接组件的方法及其功能

方 法	功 能
GetListCount	返回 URL 数据文件的行数
GetNextURL	返回 URL 数据文件中的下一行数据
GetPreviousDescription	返回 URL 数据文件前一行超链接描述
GetListIndex	返回所读取行的索引号
GetNthDescription	返回所读取行的超链接描述
GetPreviousURL	返回 URL 数据文件的上一行数据
GetNextDescription	返回下一行数据的超链接描述
GetNthURL	返回所读取行的超链接网址

上述方法的使用前提必须是文本超链接组件已经实例化。下面对各方法的描述，默认 myurl 已被实例化。

(1) GetListCount 方法

使用方式为：

```
<%
count = myurl.getlistcount (URL 数据文件名)
%>
```

(2) GetNextURL

使用方式为：

```
<%
url1 = myurl.getnextURL (URL 数据文件名)
%>
```

(3) GetPreviousDescription

使用方式为：

```
<%
des1 = myurl.getpreviousdescription (URL 数据文件名)
%>
```

(4) GetListIndex

使用方式为：

```
<%
num = myurl.getlistindex (URL 数据文件名)
%>
```

(5) GetNthDescription

使用方式为：

```
<%
des1 = myurl.getnthdescription (URL 数据文件名,索引序号)
%>
```

(6) GetPreviousURL

使用方式为：

```
<%
url1 = myurl.getpreviousurl (URL 数据文件名)
%>
```

(7) GetNextDescription

使用方式为：

```
<%
des1 = myurl.getnextdescription (URL 数据文件名)
%>
```

(8) GetNthURL

使用方式为：

```
<%
url1 = myurl.getnthurl (URL 数据文件,索引序号)
%>
```

以上,需要指出的是,索引序号是从 1 开始计数的。

4. 文件超链接组件的应用

有时,我们需要将几个友情站点链接显示在自己的网站上,为了维护方便,建议使用文件超链接组件。这样,在需要增加或删除友情站点时,就可以简单地通过修改 myurl.txt 文件来实现,从而减轻了网站维护的负担。

程序示例如下：

```
<html>
<head>
<title>
New Page 1
</title>
</head>
<body>
<%
set myurl = server.createobject ("MSWC.Nextlink")
count = myurl.getlistcount ("myurl.txt")
%>
<ol><ul>
<%
for i = 1 to count
%>
    <li>
    <a href = "<% = myurl.getnthurl ("myurl.txt",i) %>">
```

```
    <% = myurl.getnthdescription ("myurl.txt",i) %>
    </a> </li>
<%
next
%>
</ul>
</ol>
</body>
</html>
```

其中 myurl.txt 的文件内容以上已述。在建立了该组件的实例 myurl 后，利用 count = myurl.getlistcount ("myurl.txt") 获得站点数目，然后利用 For……Next 循环逐次显示。程序执行结果，如图 8－4 所示。

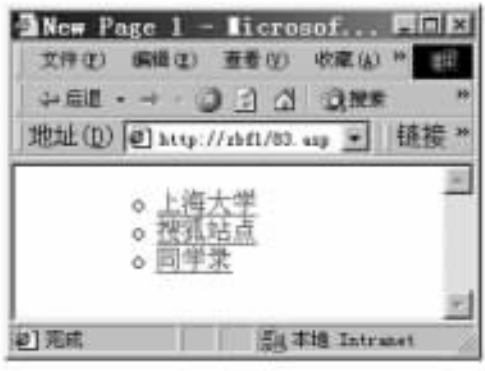

图 8－4　文件超链接组件使用示例

8.1.4　文件操作组件

在网站的建立中，许多重要的信息需要通过文件或者数据库保存起来。因此，文件存取组件对于网站建设来说也是一个重要的组件。像其他组件一样，需要利用 Server 对象的 CreateObject 方法来实现。它主要有四种用法，下面分别给出不同用法的有关属性、方法及示例。

1. 利用 FileSystemObject 组件读写文件

(1) 建立 FileSystemObject 组件的实例

使用方式为：

```
<%
set 对象实例名 = server.createobject ("Scripting.FileSystemObject")
%>
```

(2) FileSystemObject 组件的方法

假设已经建立 FileSystemObject 组件的实例 myfile，那么就可以通过 myfile 使用该组件的方法。在该组件中主要有两种方法：

① CreateTextFile 方法。它的功能是新建一个文件名称,并返回一个文件句柄供文件读写。

使用方式为:

```
<%
set mytxt = myfile.createtextfile 文件名 [,overwrite [,Unicode]])
……
%>
```

其中,overwrite 项表示是否允许覆盖文件,其值可以是 True 或 False。True 表示允许覆盖,而 False 表示不允许覆盖,默认值为不允许。Unicode 表示指定文件的格式,其值可以是 True 或 False。True 表示为 Unicode 格式,False 表示 ASCII 格式,默认值为 ASCII 格式。mytxt 表示返回的 TextStream 文件句柄。

② OpenTextFile 方法。它的功能是打开一个已经存在的文件,并返回一个文件句柄供文件读写。

使用方式为:

```
<%
set mytxt = myfile.opentextfile 文件名 [,iomode [,create,format]])
%>
```

其中,iomode 表示打开文件的目的,其值可以是 ForReading (读取,整数值为 1)或者 ForAppending 编辑,整数值为 8)。create 表示当指定文件不存在时可否自动建立新文件,其值可以是 True 或 False,True 表示可以自动建立新文件,False 表示不可以,默认值为 False。用 format 设置文件的打开方式,其值有三个:TristateTrue (以 Unicode 方式打开)、TristateFalse (以 ASCII 方式打开)、TristateUseDefault (以文件的默认方式打开)。mytxt 表示返回的 TextStream 文件句柄。

(3) TextStream 对象

当用上述方法建立了一个 TextStream 对象 (TextStream 文件句柄)时,就可以利用该对象读写文件内容了。有关该对象属性的名称及其含义,如表 8-4 所示。

表 8-4 TextStream 对象属性的名称及其含义

名 称	含 义
AtEndOfLine	用于判断当前字符是否处在行末,当其值为 True 时,表示处于行末,否则,不处于行末。
AtEndOfStream	用于判断当前字符是否处在文件结尾,当其值为 True 时,表示处于文件结尾,否则,不处于文件结尾。
Column	返回当前字符所处列号
Line	返回当前字符所处行号

有关该对象方法的名称及其功能,如表 8-5 所示。

表 8－5　TextStream 对象方法的名称及其功能

名　称	功　能
Close	关闭一个已打开的文件
Read (n)	读取已打开文件内容中的 n 个字符
ReadAll	读取已打开文件内容中的所有数据
ReadLine	读取已打开文件内容中的一行数据
Skip (n)	跳过已打开文件内容中的 n 个字符
SkipLine	跳过已打开文件内容中的一行数据
Write (string)	写入 String 到已打开文件中
WriteLine ([string])	写入一整行数据 String 到已打开文件中，如果 String 省略，则在该文件中加入换行符
WriteBlankLines (n)	在已打开文件中写入 n 行新行

(4) 利用 FileSystemObject 组件的示例

下面通过一个例子说明如何用 FileSystemObject 组件实现文件的读写操作。文件 84.asp 的程序如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
            '映射 ad.txt 文件的全路径
whichfile=server.mappath("ad1.txt")
'建立 FileSystemObject 组件的实例
set myfile = server.createobject("Scripting.FileSystemObject")
'新建一个允许覆盖的 ASCII 格式文件
set mytxt = myfile.createtextfile(whichfile,true)
for i=1 to 5
    mytxt.writeline("welcome you!")
next
mytxt.writeblanklines(2)                    '写入两行空行
mytxt.close                                 '关闭文件
'重新以可编辑方式打开该文件，8 表示 ForAppending
set mytxt = myfile.opentextfile(whichfile,8)
mytxt.writeline("the end of text!")
mytxt.close
'再以读取方式打开该文件，1 表示 ForReading
set mytxt = myfile.opentextfile(whichfile,1)
i=1
do while not mytxt.atendofstream    'atendofstream 为 True 表示文件结束
    lines=mytxt.readline
```

```
        response.write "<br>"&i&" "&lines)
        i = i + 1
    loop
    mytxt.close
    set mytxt = nothing
    set myfile = nothing
    %>
    </body>
    </html>
```

当文件执行结束后,在当前目录下建立文件 ad1.txt**,文件内容如下:**

```
1 welcome you!
2 welcome you!
3 welcome you!
4 welcome you!
5 welcome you!
the end of text!
```

客户端浏览器的显示界面,如图 8-5 **所示。**

图 8-5　FileSystemObject **组件使用示例**

(5) 计数器示例

一般网站上都有记录访问人数的计数器功能。可以通过文件实现这种功能。首先在服务器上建立计数文件 count.txt**,以保存访问人数,文件中初始内容为"0"。**

程序示例如下:

```
<html>
<head>
```

```
<title>counter</title>
</head>
<body>
<%
dim visitors
dim mailobject
whichfile = server.mappath ("count.txt")
set fs = createobject ("scripting.filesystemobject")
set thisfile = fs.opentextfile (whichfile)
visitors = thisfile.readline
thisfile.close
visitors = visitors + 1
set thisfile = fs.createtextfile (whichfile)
thisfile.writeline (visitors)
thisfile.close
set fs = nothing
%>
```

欢迎你，你是第 <% = visitors %> **个访问者！**

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

程序执行结束后返回客户端浏览器上的结果，如图 8－6 **所示。**

图 8－6　**访问者计数器**

2．**利用** FileSystemObject **组件操作文件**

在上一节中，我们着重讲述利用 FileSystemObject **组件进行文件的创建和读写。在本节中，讲述如何进行文件的拷贝、移动、删除等操作。当然实现这些操作要通过该组件的一系列方法。**

（） File **对象的属性**

在处理文件前，有必要了解有关文件的一些基本情况。假定 myfile **已建立为该组件的实例，那么可以利用** GetFile **（）查询目标文件的部分属性，其名称及其含义如表** 8－6 **所示。**

表 8－6　File 对象部分属性的名称及其含义

名　称	含　义
Attributes	显示当前文件的系统属性,返回值为整数：
	0　Normal
	1　Read-Only
	2　Hidden
	4　System
	8　Volume
	16　Directory
	32　Archive
	64　Alias
	128　Compressed
DateCreated	返回该文件被创建的日期和时间
DateLastAccessed	返回该文件上一次被访问的日期及时间
DateLastModified	返回该文件上一次被修改的日期及时间
Drive	返回该文件所在驱动器
Name	返回该文件的文件名
ParentFolder	返回包含该文件的文件夹名称
Path	返回该文件所在的完整物理路径
Size	返回该文件的大小
Type	返回该文件的类型

下面通过一个示例来说明使用方法。要了解已知文件 ad.txt 的属性,编写的 ASP 文件 85.asp 程序如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
filepath=server.mappath("ad.txt")
set myfile=server.createobject("Scripting.FileSystemObject")
set mytxt=myfile.getfile(filepath)
response.write"<br>文件的上一次使用日期为"&mytxt.datelastaccessed
response.write"<br>文件的上一次修改日期为"&mytxt.datelastmodified
response.write"<br>文件所在驱动器为"&mytxt.drive
response.write"<br>文件的大小为"&mytxt.size&"字节"
%>
</body>
</html>
```

在上述程序中,首先利用 Server 对象的 mappath 方法映射出文件 ad.txt 的完整物理路径,然后利用 CreateObject 方法建立 FileSystemObject 组件的实例,再利用该实例的 GetFile（）方法

查询文件的属性。程序执行结果，如图 8－7 所示。

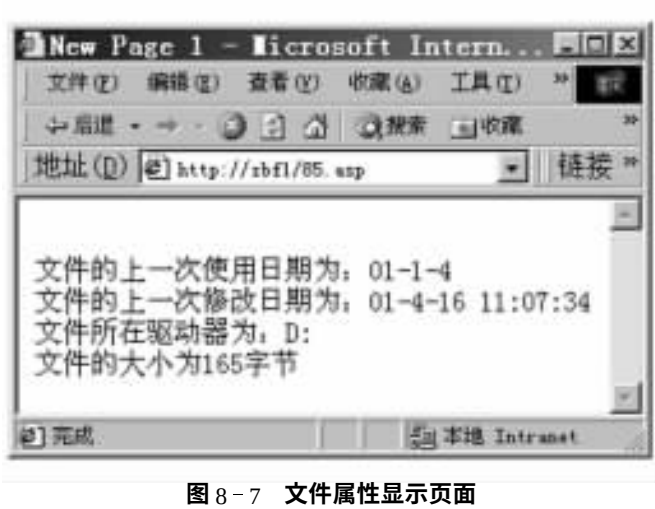

图 8－7　文件属性显示页面

（2）FileSystemObject 组件的方法

有关该组件在这方面的方法主要有三种：

① "CopyFile 源文件路径名 目的文件路径名 [Overwrite]"，该方法的主要作用是复制文件，其中 Overwrite 为可选项，用来决定当目的文件已经存在的前提下是否进行覆盖操作，当 Overwrite 参数值为 True 时可以进行覆盖操作，否则不可覆盖，默认值为 False。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
myfile.copyfile "d:/tree1/w1.txt" "d:/tree2/w2.txt"
……
myfile.close
set myfile = nothing
%>
```

② "MoveFile 源文件路径名 目的文件路径名"，该方法的主要作用是移动文件。当目的文件已经存在时会报错，不允许进行覆盖操作。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
myfile.movefile "d:/tree1/w1.txt" "d:/tree2/w2.txt"
……
myfile.close
set myfile = nothing
%>
```

③ "DeleteFile 指定删除文件名"，该方法的主要作用是删除指定文件。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
myfile.deletefile "d:/tree1/w1.txt"
……
myfile.close
set myfile = nothing
%>
```

当然也可以通过调用 GetFile ()方法利用 File 对象实现以上操作，下面就介绍一下这种用法。

① "Copy 新文件路径名 [,Overwrite]"，利用 Copy 方法实现文件复制操作，其中 Overwrite 为可选项，用来决定当目的文件已经存在的情况下是否进行覆盖操作。当 Overwrite 参数值为 True 时可以进行覆盖操作，否则不可覆盖，默认值为 False。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
set mytxt = myfile.getfile ("d:/tree1/w1.txt")
mytxt.Copy "d:/tree2/w2.txt"
……
mytxt.Close
set mytxt = nothing
myfile.Close
set myfile = nothing
%>
```

② "Move 新文件路径名"，利用 Move 方法实现文件移动操作。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
set mytxt = myfile.getfile ("d:/tree1/w1.txt")
mytxt.move "d:/tree2/w2.txt"
……
mytxt.Close
set mytxt = nothing
myfile.close
set myfile = nothing
%>
```

③ "Delete"，利用 Delete 方法实现文件删除操作。特别应该指出的是，在该方法中没有

参数。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
set mytxt = myfile.getfile ("d:/tree1/w1.txt")
mytxt.delete
……
mytxt.close
set mytxt = nothing
myfile.close
set myfile = nothing
%>
```

(6) 利用该组件检测某文件是否存在

在对某个文件进行处理前，经常需要判断该文件是否存在。这时可以通过调用 FileSystemObject 对象中的 FileExists ()方法实现。

当调用该方法时，它的返回值为 True 或 False。当返回值为 True 时，表示该文件存在，否则表示该文件不存在。

使用示例如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312">
<title>New Page 1</title>
</head>
<body>
<%
filepath = server.mappath ("ad.txt")
set myfile = server.createobject ("Scripting.FileSystemObject")
if myfile.fileexists (filepath) then
    response.write ("该文件存在")
else
    response.write ("该文件不存在")
end if
myfile.close
set myfile = nothing
%>
</body>
</html>
```

当指定文件存在时，在客户端浏览器上显示 '该文件存在 '的字样 ;否则，在客户端浏览

器上显示‘该文件不存在’的字样。

3. 利用 FileSystemObject 组件进行文件夹的操作

在 ASP 文件的处理中,有时需要了解有关文件夹的属性。现在应该学习一下该组件在这方面的应用。这一般是通过调用 GetFolder ( )方法实现的。

(1) Folder 对象的属性

有关 Folder 对象属性的名称及其含义,如表 8-7 所示。

表 8-7 Folder 对象属性的名称及其含义

名 称	意 义
Files	返回所有该文件夹下文件的集合,隐文件不显示
IsRootFolder	用来判断该文件夹是否为根目录,如果返回值为 True,则为根目录,否则不是根目录
Name	返回当前文件夹的名称
ParentFolder	返回当前文件夹的上一级目录
Size	返回当前文件夹所占空间大小
SubFolder	返回该文件夹下所有子文件夹的集合

下面举例说明对文件夹的操作方法。程序 86.asp 内容如下:

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
set myfile = server.createobject("Scripting.FileSystemObject")
set mytree =myfile.getfolder("d:/pda/")
response.write("<br>当前目录为"&mytree.name)
response.write("<br>当前目录所占空间大小为"&mytree.size&"字节")
response.write("<br>当前目录的上级目录为"&mytree.parentfolder)
set files=mytree.files
response.write("<br>当前目录下文件为")
for each file in files
    response.write("<br>"&file.name)
next
%>
</body>
</html>
```

上述程序的执行结果,如图 8-8 所示。

(2) FileSystemObject 组件的方法

图 8-8　文件夹属性显示页面

像 FileSystemObject 组件在文件操作中的应用一样，可以直接通过 FileSystemObject 组件对文件夹进行操作。

① "CopyFolder 源文件夹路径名 目的文件夹路径名 [,Overwrite]"，该方法用于文件夹的复制，Overwrite 参数为可选项，用来设定当目的文件夹已存在时，是否进行覆盖。如果该参数为 True，表示允许覆盖，否则表示不允许覆盖，默认值为 False。

使用方式为：

```
<%
set myfile = server.createObject ("Scripting.FileSystemObject")
myfile.copyfolder "d:/tree1/" "d:/tree2/"
……
myfile.close
set myfile = nothing
%>
```

② "CreateFolder 新建文件夹路径名"，该方法用于创建一个指定的新文件夹。

③ "DeleteFolder 指定文件夹路径名"，该方法用于删除一个指定的文件夹。

④ "GetParentFolderName 指定文件夹路径名"，该方法用于返回指定文件夹的上一级文件夹名。

⑤ "MoveFolder 源文件夹路径名 目的文件夹路径名"，该方法用于移动文件夹。

其中②～⑤的使用方法与（）所述相似，在此不再赘述。

上述的一系列操作也可以通过 Folder 对象来实现（Folder 对象可以通过 GetFolder（）方法来创建）。使用下述方法的前提是 Folder 对象已经利用 GetFolder（）方法建立。

① "CopyFolder 目的文件夹路径名 [,Overwrite]"，利用 CopyFolder 方法实现文件夹复制操作，其中 Overwrite 为可选项，决定当目的文件夹已经存在的情况下是否进行覆盖操作。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
set mytxt = myfile.getfolder ("d:/tree1/")
mytxt.copyfolder "d:/tree2/"
……
mytxt.close
set mytxt = nothing
myfile.close
set myfile = nothing
%>
```

② "DeleteFolder",利用 DeleteFolder 方法实现文件夹删除操作。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
set mytxt = myfile.getfolder ("d:/tree1/")
mytxt.deletefolder "d:/tree2/"
……
mytxt.close
set mytxt = nothing
myfile.close
set myfile = nothing
%>
```

③ "MoveFolder 目的文件夹路径名",利用 MoveFolder 方法将当前目录移动到目的文件夹。

使用方式为：

```
<%
set myfile = server.createobject ("Scripting.FileSystemObject")
set mytxt = myfile.getfolder ("d:/tree1/")
mytxt.movefolder "d:/tree2/"
……
mytxt.close
set mytxt = nothing
myfile.close
set myfile = nothing
%>
```

4. 利用 FileSystemObject 组件进行驱动器的有关操作

在 ASP 文件的处理中,有时需要了解有关驱动器的属性。当需要使用有关该组件进行驱动器操作时需要通过调用 GetDrive ()方法来实现。有关该对象属性或方法的名称及其含义,如表 8－8 所示。

表 8－8　驱动器对象属性或方法的名称及其含义

名　称	含　义
AvailableSpace	返回当前驱动器的总空间
DriveLetter	返回当前驱动器的盘符
DriveType	返回当前驱动器的类型
FreeSpace	返回当前驱动器的可使用空间大小（字节）
Path	表示目前驱动器的路径
RootFolder 方法	将目前驱动器映射为一个文件夹对象
SerialNumber	返回当前驱动器的序列号
ShareName	返回当前驱动器的共享名称
TotalSize	返回当前驱动器的总容量
VolumeName	返回当前驱动器的卷标名字符串

当需要了解有关某驱动器的有关属性时，就可以利用该对象。例如要了解 C：盘的情况，实现该目的的 ASP 文件内容如下：

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
testdriver = "c:"
set myfile = server.createobject("Scripting.FileSystemObject")
set mydriver = myfile.getdrive("C:")
response.write("<br>当前盘符为"&mydriver.DriveLetter)
if mydriver.drivetype = 1 then
    response.write("<br>当前驱动器类型为可移动的")
else
    if mydriver.drivetype = 2 then
        response.write("<br>当前驱动器类型为固定的")
    else
        response.write("<br>当前驱动器类型未知")
    end if
end if
response.write("<br>当前驱动器总空间大小为"&mydriver.totalsize&"字节")
response.write("<br>当前驱动器可用空间为"&mydriver.freespace&"字节")
response.write("<br>当前驱动器序列号为"&mydriver.serialnumber)
response.write("<br>当前驱动器卷标名为"&mydriver.volumename)
set mydriver = nothing
set myfile = nothing
%>
</body>
```

```
</html>
```

上述程序的执行结果，如图 8－9 所示。

图 8－9　驱动器属性显示页面

8.2　第三方组件

在本书前几个章节中用到的基本上都是系统自带的组件。为了扩充网站的功能，有时需要从其他有关网站上下载第三方组件。当它们安装后会自动在注册表中登记。下面详细介绍几个第三方组件。

8.2.1　Mailer 组件

该组件可以从 http：//www.serverobjects.com 上下载，其作用主要是为了发送电子邮件。下载后，在 MS-DOS 方式下，运行 Regsvr32 Smtpsvg.dll 命令可以将该组件登记。

有关发邮件的表单程序 SendMailer.htm 内容如下：

```
<html>
<body bgcolor=white>
<h2 align=center>Form Mailer-Test Page</h2>
<form method=post action="sendmail.asp">
Enter Mail To Address:<input type=text name="addressto"><br>
Enter Mail To Name:<input type=text name="nameto"><br>
Enter Mail Message:<textarea name="txtmsg" rows=10
      cols=60>Hello!</textarea>
<input type="submit">
</form></body>
</html>
```

上述程序的执行结果，如图 8－10 所示。

图 8－10　邮件发送表单

表单处理程序 sendmail.asp 内容如下：

```
<html>
<head>
<title>ASP Mailer Form Test</title>
<head>
<body bgcolor="white">
<h3 align="center">ASP Mailer Form Test</h3>
Mail To:<% = request.querystring("addressto") %><br>
<p>
<%
'建立一个 Mailer 组件
set mailer = server.createobject("SMTPsvg.Mailer")
mailer.remotehost = request.form("addressto")
'将发件人的姓名、邮件地址赋予邮件相应的属性
mailer.fromname = "Joe's Shrimp Shack"
mailer.fromaddress = "anyone@unknownhost.net"
'将收件人的姓名、邮件地址赋予邮件相应的属性
mailer.addrecipient request.querystring("nameto"),
request.querystring("addressto")
'将邮件的主题、内容赋予相应属性中
mailer.subject = "Stock Split Announced!"
mailer.bodytext = request.form("txtmsg")
'将邮件进行发送,并根据不同情况进行不同的处理
if not mailer.sendmail then
```

```
    response.write "Mailing Failed... Error is:<br>"
    response.write mailer.response
else
    response.write "sent successfully...<p>"
end if
response.write "This component expires:" & mailer.expires & "<p>"
%>
</body>
</html>
```

8.2.2 常见浏览器组件

1. PageCounter 组件

该组件主要为了显示页面被访问的次数,利用它可以免去许多语句的编写。可以在 http://www.microsoft.com/iis 上下载,不过在 IIS4 中默认有这个组件。其组件登记步骤与 Mailer 组件的登记方式相似。

创建方式为:

```
<%
set pagecounter 对象实例 = server.createobject("IISSample.PageCounter")
%>
```

表 8-9 列出了该组件的主要方法及其功能。

表 8-9 PageCounter 组件的主要方法及其功能

方 法	功 能
Hits (path)	该方法返回 path 所表示页面的点击次数,如果没有设置参数 path,则默认为当前页
Reset (path)	该方法主要用于将 path 所表示页面的点击次数设置为 0 如果没有设置参数 path,则默认为当前页

若要在某页面上添加该组件,以显示点击次数,不妨参照下述方法:

ASP 文件内容如下:

```
<html>
<head>
<title>New Page 1</title>
</head>
<body>
<%
set mycounter = server.createobject("IISSample.PageCounter")
myhits = mycounter.hits
response.write "你是第"&myhits&"个访问者"
%>
```

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

要将某页面或当前页的点击次数设置为 0，需要用到 Reset（ ）方法。

程序内容如下：

```
<html>
<head>
<title>我的网站</title>
</head>
<body>
<%
Set mycounter = server.createobject ("IISSample.PageCounter")
mycounter.reset
mycounter.reset ("http://zbf1/heartmemo.htm")
%>
</body>
</html>
```

2．PageFormat 组件

该组件可以把服务器端的纯文本文件在客户端浏览器上打开。该组件的注册步骤与 Mailer 组件的注册类似。

使用方式为：

```
<%
set pageformat 组件实例 = server.createobject ("IISSample.TextFMT"&".C++")
%>
```

表 8－10 列出了该组件的主要方法及其功能。

表 8－10　PageFormat 组件的主要方法及其功能

方　法	功　能
About	返回版本信息
WrapTextFromFile	返回要打开文件的纯文本格式

当需要利用该组件显示纯文本内容时，可以使用上述组件，并且可以将其内容显示在 < textarea > 与 < /textarea > 标签之间。

3．HTTP 组件

在网站 http://www.genusa.com/asp/上可以下载名叫 ASPHTTP 的组件。该组件符合 HTTP 的 GET 标准，不过这只是一个简易版本（.0 版），不提供显示图形的功能，主要用于显示网页的文字。

建立方式为：

```
<%
set HTTP 对象实例 = server.createobject ("ASPSVG.HTTP")
```

```
%>
```

该组件的登记步骤，与 Mailer 组件的登记步骤相似。在 MS-DOS 方式下，运行 Regsvr32 AspHttp.dll 命令可以将该组件登记。

表 8－11 列出了该组件属性或方法的名称及其含义。

表 8－11　HTTP 组件属性或方法的名称及其含义

名　称	含　义
URL	指向欲链接的 URL 地址
Port	表示欲链接的端口（Port）
TimeOut	描述欲链接的网页最长时间延迟
GetURL 方法	获取网页，返回其中内容

当我们需要了解有关某网页的内容时，可以使用该组件。程序内容如下：

```
<html>
<head>
<title>HTTP 范例</title>
</head>
<body>
<%
set myhttp = server.createobject("ASPSVG.HTTP")
myhttp.URL = "http://www.shu.edu.cn"
myhttp Port = 80
myhttp.timeout = 30
myweb = myhttp.geturl
response.write myweb
set myhttp = nothing
%>
</body>
</html>
```

读者可以将上述程序亲自实践一下，以考察它的运行效果。

4．ImageSize 组件

该组件可以在网站 www.ServerObjects.com 上下载，它能够获取某个图像的高度、宽度等属性，它支持的文件有 bmp、jpeg（jpg）、gif 等格式。它的登记方法同以上控件相似，即在 MS-DOS 命令界面上执行 regsvr32 imgsize.dll 命令。

创建方式为：

```
<%
set imagesize 对象组件 = server.createobject("ImageSize.Check")
%>
```

该组件属性的名称及其含义，如表 8－12 所示。

表 8－12　ImageSize 组件属性的名称及其含义

名　称	含　义
FileName	表示该组件显示的图像文件名
Height	表示显示图像的高度
Width	表示显示图像的宽度
Error	返回出现的错误信息

当我们需要显示某一图像文件信息时，可以使用该组件。程序内容如下：

```
<html>
<head>
<title>ImageSize 示例</title>
</head>
<body>
<%
set myimg = server.createobject ("ImgSize.Check")
myimg.filename = "d:\ad.gif"
if myimg.err<>"" then
    response.write "错误发生，错误信息为"
    response.write myimg.err
else
    response.write "<br>图像文件名为"&myimg.Filename
    response.write "<br>图像的高度为"&myimg.Height
    response.write "<br>图像的宽度为"&myimg.Width
end if
set myimg = nothing
%>
</body>
</html>
```

8.2.3　获得 ASP 组件的网站

当编制网站时，需要用到的组件很多很多，前面描述的组件只不过是 ASP 组件汪洋大海中的一朵浪花。表 8－13 中列出了提供 ASP 组件的部分网站，在这些网站上或许有读者所需的组件。

表 8－13　提供 ASP 组件的部分网站

网　站	提供的组件
http://www.microsoft.com	在该网站上提供大量的 ASP 组件
http://www.activestate.com	提供 Perl Script 引擎，使得 ASP 可以使用 Perl 语言作为脚本语言
http://www.renetusa.com/asc	提供文件的拷贝、删除和移动等组件

续 表）

网 站	提供的组件
http://www.aufrance.com	提供日期组件等
http://www.codewarriors.com	提供送信组件等
http://www.digitalfoundry.com	提供聊天室组件
http://www.dimac.net	提供目录树、Email 组件、Socket 组件等
http://www.softwarex.com/cfxie	提供图表组件

8.3 创建自己的组件

为了使自己的网站具有新意，有时需要创建自己的组件，这可通过 Microsoft Visual Basic、Micorsoft Visual C++、Microsoft Visual J++ 等语言来实现。至于创建组件的具体方法，在相关语言中均有介绍，有兴趣的读者可以参阅有关书籍，这里不再赘述。

8.4 习题

1. 基础题

① 利用（ ）组件实现广告图片的轮换显示，利用（ ）组件实现文件的一系列操作，利用浏览器组件可以实现（ ）。

② 在下述组件中，（ ）组件需要利用配置文本。

A. AdRotator 组件　　B. BrowserType 组件

C. 文件超链接组件　　D. 文件存取组件

③ 在下述组件中，（ ）组件不是内置组件。

A. AdRotator 组件　　B. BrowserType 组件

C. Mailer 组件　　D. 文件存取组件

④ 在相关网站上下载组件后，需要执行（ ）命令登记组件信息。

2. 操作题

建立一个网站，实现以下目的：

① 在网站主页上，显示访问者是第几个访问者，并且显示访问者是第几次访问本站。

② 利用浏览器组件显示有关站点的链接。

③ 利用 AdRotator 组件显示站点的宣传。

④ 在网站上添加发送邮件的功能。

⑤ 发挥你的想像，为网站添加新的功能。

第九章　ASP 与数据库

一个成功的网站必然有许多网站信息和用户信息需要保存，所有这些必然离不开数据库的帮助。因此，在本章中着重讲解有关数据库在 ASP 中的应用。这一章的内容比较多，读者可以根据自己的情况进行取舍。

9.1　应用前提

首先，有必要了解一些有关 ODBC（Open Database Connectivity）和 SQL（Structured Query Language）语言的预备知识，为今后数据库的学习奠定基础。

9.1.1　ODBC 应用简介

当数据库建立后，还需要利用 ODBC 数据源管理器应用程序将已建数据库进行登记，以方便数据库的使用。

打开控制面板，会发现“ODBC 数据源管理器”应用程序。双击该图标激活该应用程序，如图 9－1 所示。

图 9－1　“ODBC 数据源管理器”页面

如果已经利用 Microsoft Access 建立了数据库文件 employee.mdb，该数据库包含的表可以根据需要来设计，那么可以利用以下步骤进行登记：

① 启动 ODBC 应用程序，点击‘系统 DSN’标签，如图 9－2 所示。

图 9－2　点击‘系统 DSN’后的页面

② 点击‘添加’按钮，激活添加数据库的操作界面，如图 9－3 所示。

图 9－3　‘创建新数据源’页面

③ 选择所创建数据库的类型。例如 employee.mdb 是利用 Microsoft Access 建立的，那么需要选中第二项（Microsoft Access Driver (*.mdb)），点击‘完成’按钮，激活下一步操作画面，如图 9－4 所示。

④ 这时，点击‘选取’按钮，启动‘选定数据库’对话框，进行数据库的选择，如图 9－5 所示。

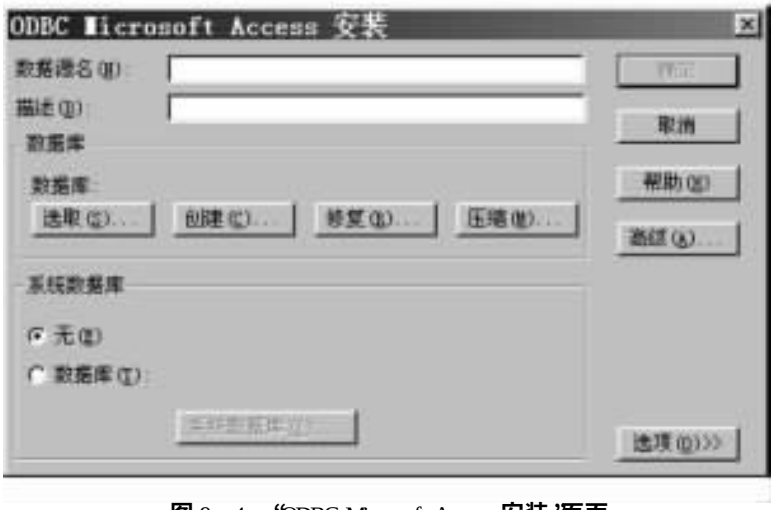

图 9－4　"ODBC Microsoft Access 安装"页面

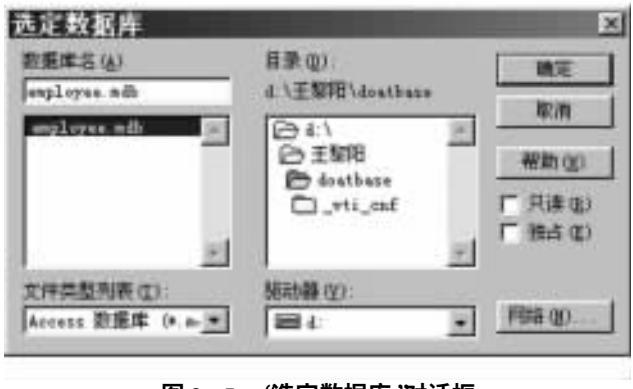

图 9－5　"选定数据库"对话框

⑤ 当选择结束后，点击"确定"按钮，将会返回到图 9－4 所示的界面。然后在数据源名输入框中填写数据源名，例如 dbasp，如图 9－6 所示。

图 9－6　数据源属性设置结束

⑥ 当数据源名填写完毕后，点击“确定”按钮返回，界面如图 9－7 所示。

图 9－7　数据源 mydata 添加后的“ODBC 数据源管理器”页面

可以看出，在系统 DSN 页面上添加了一项 dbasp（Microsoft Access Driver）。至此，数据库的登记操作结束。

9.1.2　SQL 语言简介

SQL 语言是一种专门用于关系数据库结构化的查询语言，是 1974 年由 Royee 和 Chamberlin 提出的。并于 1986 年由美国国家标准局的数据库委员会 X3H2 批准为关系数据库语言的美国标准，并于同年公布了标准的 SQL 文本。不久以后，国际化标准组织也作出了同样的决定。

1．SQL 概述

通常所说的 SQL 语言是符合 ANSI 标准的。ANSI SQL 可以分成六个基本类。

⑴ 数据查询语言 DQL

这类语句一般包括 SELECT 语句，可以从表中获得数据，并决定如何提交给应用程序。

⑵ 数据操作语言 DML

这类语句一般包括 INSERT、UPDATE 和 DELETE 语句，可以对表中的数据进行添加、修改和删除等操作。

⑶ 事务处理语言 TPL

这类语句一般用来确保 DML 语言对数据库所进行的操作，可以立即执行。

⑷ 数据控制语言 DCL

这类语句一般包括 GRANT 和 REVOKE 语句，用来设置和取消某用户对某数据库的访问权限。

⑸ 数据定义语言 DDL

这类语句可以在数据库中创建新表,也可以为表设置索引。

(6) 光标控制语言 CCL

这类语句一般用来进行光标设置。

虽然 ANSI SQL 可分为六个基本类,但最常用的是前两类。下面对 SQL 语言进行较为详细的描述。

2. SQL 查询语句

在 SQL 的应用中,查询语句可以说是最常用、最核心的语句。它的使用形式多种多样,应该熟练掌握。当然 SQL 语言不仅仅应用于 ASP 中,在其他语言中也常常用到它。该语句的一般格式如下:

```
SELECT 字段名 1 [字段名 2]
FROM 基本表名 1 视图名 1) [基本表名 2 视图名 2)]
[WHERE 条件表达式]
[GROUP BY 字段名 3]
[ORDER BY 字段名 4]
```

上述表达式中,"[ ]"和"[ ]"之间的内容表示可有可没有,"( )"和"( )"之间的内容表示可代替括号外的内容。

例 1:在职员表 employee 中要将工资(在表中的域名为 income)高于 5 000 元的职员姓名(域名为 name)列出来,则可以通过以下语句来实现:

```
SELECT name
FROM employee
WHERE income > = 5000;
```

例 2:在职员表 employee 中要将工资高于 5 000 元的职员的所有信息列出来,则可以通过以下语句来实现:

```
SELECT *
FROM employee
WHERE income > = 5000;
```

其中,"*"表示所调用表中的列名所对应的值。

例 3:在职员表 employee 中要将工资高于 5 000 元并且性别(在表中的域名为 sex)为 female 的职员姓名、职务(在表中的域名为 position)列出来,则可通过以下语句来实现。

```
SELECT name
FROM employee
WHERE income > = 5000 and sex = 'female';
```

如果要在多个表中进行查询,并且不同的表中有相同的域名,那么必须在该域名前加上表名。

例 4:现在要在教师表 teacher 和班级表 class 两个表中进行查询,将教"9602"班的班主任的收入查询出来。在班级表中必然要有班主任的名字(teachername),还有班级名(classname)。教师表中也有班主任的名字,还有班主任的收入(income)。实现上述目的的语句如下:

```
SELECT classname, teacher.teachername, income
FROM teacher, class
WHERE class.classname = '9602' and class.teachername = teacher.
    teachername;
```

数据库的查询语句 SELECT 的用法很多,以上提到的只是最常用的一部分,希望读者在今后的学习中能灵活使用,也可查阅有关资料。

3. SQL 更新语句

SQL 更新语句,主要包括 UPDATE、DELETE、INSERT 语句。这一系列语句用于对数据库进行修改操作。在使用前一般需要进行检验,防止误操作。

(1) UPDATE 语句

该语句的一般格式为:

```
UPDATE 表名
SET 字段名 1 = 表达式 1 [字段名 2 = 表达式 2]
[WHERE 条件表达式]
```

上述语句可以结合 SELECT 语句,以完成较为复杂的操作。

例 1:单记录修改。如果要将学号 num 为 S1 的学生的姓名改为 Mary,可以用下列方法来实现:

```
UPDATE Student
SET name = 'Mary'
WHERE num = 'S1'
```

例 2:多记录修改。将所有学生的学龄加 1。

```
UPDATE Student
SET studyage = studyage + 1
```

例 3:结合 Select 语句实现复杂查询,将所有计算机系学生的成绩置 0,其中,SCORE 表示学生的成绩表,STUDENT 表示学生的信息表。

```
UPDATE SCORE
SET score = 0
WHERE 'computer' =
    (SELECT department
    FROM STUDENT
    WHERE SCORE.num = STUDENT.num)
```

(2) DELETE 语句

删除语句的一般格式为:

```
DELETE
FROM 表名
[WHERE 条件表达式]
```

例 1：单记录删除。将学号为 S1 的学生从表 STUDENT 中删除。

```
DELETE
FROM STUDENT
WHERE num = 'S1'
```

例 2：多记录删除。将计算机系的学生都删除。

```
DELETE
FROM STUDENT
WHERE department = 'computer'
```

(3) INSERT 语句

插入语句的一般格式为：

```
INSERT
INTO 表名 (字段名 1 [,字段名 2])
VALUES (值 1 [,值 2])
```

例 1：单记录插入。将新生 John 登记注册入学生信息表中。

```
INSERT
INTO STUDENT
VALUES ('S21','John','Computer',23)
```

例 2：多记录插入。将表 STUDENT1 中的数据写入表 STUDENT 中。

```
INSERT
INTO STUDENT
SELECT *
FROM STUDENT1
```

本节简单介绍了有关 SQL 的常用知识。如果需要对其进行深入了解，可参阅有关书籍。

9.2 ADO 对象模型

ADO (ActiveX Data Objects) 是 Microsoft 公司为 Web 开发者使用数据库而提供的开发工具。它可以兼容几乎所有的数据库系统，例如 Foxpro、MS Access、MS SQL Server、Oracle、Sybase 等类型的数据库。而且，它可以供许多语言开发环境使用，例如 VB、C++、Java 等语言。如果服务器端操作系统支持 COM 以及 OLE，如 Microsoft 公司推出的操作系统系列等等，那么 ADO 就可以使用。通过它使用数据库，可以为各种类型的数据库提供一个公共的调用端口，使得 ASP 程序与数据库的联系变得更自由。

9.2.1 ADO 的原理

ADO 对数据提供了应用级编程接口 (Application-Level Programming Interface)，对网站开发

者提供了存取有关数据、保存网站有关信息的技术。在数据库的支持下，ASP 的功能变得更强大。

它不同于 OLE DB，因为 OLE OB 是 Microsoft 仅仅提供给数据库系统级的程序接口（System-Level Programming Interface）。换句话说，ADO 并不与数据资源直接打交道，而是通过 OLE DB 实现数据的传递。OLE DB 有两种方式访问数据库：一种是通过 ODBC 驱动器，另一种是直接通过原始的 OLE DB 提供程序。通常使用的是前一种。不过 ODBC 也有它的不足之处，例如它仅能处理支持 SQL 语言的数据库。但总的来说，利用 ADO 可以开发更有效的 Web 应用程序。

9.2.2 ADO 的使用

ADO 主要通过 ODBC 与数据库建立联系。ADO 组件由七个对象和四个集合组成，其对象模型分为四级，如图 9－8 所示。

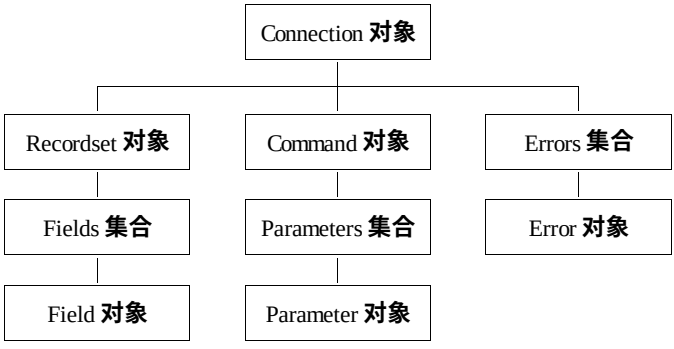

图 9－8　ADO 组件总体结构图

在图 9－8 中，只包括了六个对象和三个集合，另外还有 Properties 集合和 Property 对象是 Connection、Command、Recordset 和 Field 对象所共同具有的，如图 9－9 所示。

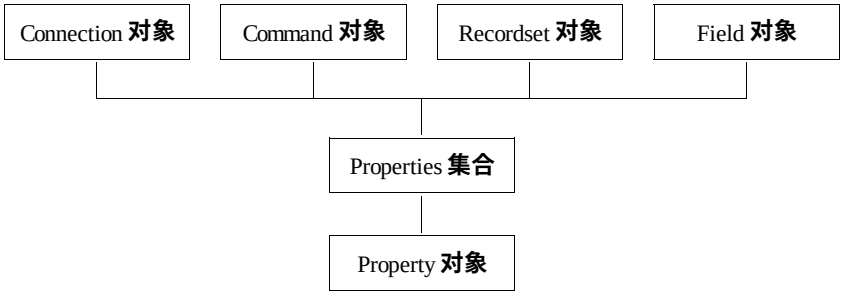

图 9－9　Properties 集合和 Property 对象的关系图

ADO 对象的名称及其功能，如表 9－1 所示，其中最主要的是三个对象——Connection、Command 和 Recordset 对象。通过这三个对象，我们在 ASP 脚本中可以与数据库连接，查询数据库中的数据，以及插入、删除和修改数据库中的数据。

表 9－1 ADO 对象的名称及其功能

名 称	功 能
Connection	提供对数据库服务器的连接
Command	对数据库服务器提供数据查询
Recordset	由数据库服务器返回的记录集合
Fields	由多个返回的数据字段形成的集合
Field	由数据库服务器返回的单一数据字段
Properties	由多个 Property 对象构成的集合
Property	单独的一个 Property 对象，提供属性功能
Parameters	有多个 Parameter 对象构成的集合
Parameter	单独的一个 Parameter 对象，提供属性功能
Errors	由多个 Error 对象构成的集合
Error	单独的一个 Error 对象，提供处理错误的功能

Connection 对象：Connection 对象用于创建应用程序和数据库之间的连接，Command 对象和 Recordset 对象是在 Connection 对象的基础上完成查询和更新数据库操作的，由于在和数据库交互的过程中可能包含了一个或者多个从数据库返回的错误，所以在 Connection 对象中还包括了 Errors 集合。

Command 对象：Command 对象用于定义数据库的操作，通常情况数据库操作以 SQL 指令的形式表现出来，现在绝大多数数据库都支持 SQL 语言，除了 SQL 指令之外，Command 对象还允许使用特定数据库自身的指令，比如说，当我们访问 SQL Server 数据库的时候，可以使用 SQL Server 中特殊的指令。

Recordset 对象：Recordset 对象也许是 ADO 对象中最复杂的一个对象，不过它的功能十分强大，在 Recordset 对象中包含了从数据库中查询的结果集合。使用 Recordset 对象，我们可以每次取出结果集合中的一条记录，独立地访问记录中的每一个字段。通过服务器端的脚本环境，我们还可以对结果集合中的记录进行分析和统计。

在这个组件中还有另外四个对象——Field、Property、Parameter 和 Error 对象。

Field 对象：Field 对象依赖于 Recordset 对象，它代表着 Recordset 对象中结果集合的某一列数据，Fields 集合则代表了 Recordset 对象中所有的域。

Parameter 对象：Parameter 对象代表传给 Command 对象的参数，比如说，如果我们要调用 SQL Server 的存储过程，我们可以将其参数用 Parameter 对象存储起来，然后再使用 Command 对象对其进行调用。Parameter 对象存储的是一个参数，而 Parameters 集合则代表了用于 Command 对象的所有参数。

Error 对象：Error 对象代表从数据源中返回的错误信息，运用 Error 对象，我们可以判断错误的准确原因。同样，Error 对象代表来自数据源的一个错误，Errors 集合则代表了来自数据源的所有错误。

Property 对象：Property 对象代表着 ADO 中 Connection、Command、Recordset 和 Field 对象的属性，根据调用对象的不同，Property 对象也反映了不同的属性，其中，属性分为内部属性（ADO 对象中的预定义属性）和外部属性（OLE DB 供应者添加到 ADO 对象中的属性）。Properties 集合代表了一个指定 ADO 对象的所有属性的集合。

下面详细介绍有关 ADO 的具体使用方法。

9.3 Connection 对象

Connection 对象主要负责 ASP 应用程序与服务器端数据库之间的实际连接操作，其他两个对象：Command 对象和 Recordset 对象与数据库之间进行的一系列操作需要由 Connection 对象来完成。对于数据库来说，Connection 对象是它与应用程序之间的惟一通道。因此 Connection 对象在数据库的应用中扮演着一个十分重要的角色。

9.3.1 创建 Connection 对象实例

创建方式为：

```
<%
Set mycon = Server.CreateObject ("ADODB.Connection")
%>
```

在使用该对象前必须首先创建一个 Connection 对象实例。但在创建了一个 Connection 对象后，仅仅是定义了一个变量，与数据源之间的连接还没有真正建立，还需要通过 Open 方法来打开。

下面实现留言簿功能的例子就说明了如何使用该对象，其中与留言簿对应的表结构，如图 9－10 所示。

图 9－10　留言簿表结构设计

程序 Sendnote.asp 如下。

```
    '设置 Response 对象的 buffer 属性为 True
<% response.buffer = true %>
<html>
<head>
<title>留言簿</title>
<bgsound src = "C05－06.mp3" loop = "－1">
</head>
<body bgcolor = "#d0d0d0">
<p align = "center">
<img src = "娃娃1.gif">
<font size = 20><font face = "华文行楷">留言簿</font></font>
```

```
</p>
<div align=center>
<form action="sendrespond.asp" method="post" name = "FrontPage_ Form1"
     onsubmit = "return FrontPage_ Form1_ Validator (this)">
<p>
留言对象：<!--webbot bot="Validation" S-Display-Name="留言对象"
B-Value-Required="TRUE" I-Minimum-Length="1" -->
<input type="text" name="nm" size="20" style="background-color:
  #ffffff">
</p>
<p align="center">
<textarea rows="8" cols="50" name="ly" style="background-color:
  #ffffff">
</textarea>
<br><br>
<input type=submit name=bt value="确认提交" style="border-style:solid;
       border-color:#FF6666">   
<input type=reset name=br value="重新输入" style="border-style:solid;
  border-color:#FF6666">
</p>
</form>
</div>
</body>
</html>
```

上述 ASP 文本的表单中主要有两个填写项：留言对象和留言内容，如图 9-11 所示。响应该表单的表单处理程序 sendrespond.asp 如下：

```
<html>
<head>
<title>
欢迎你的留言
</title>
</head>
<body bgcolor="#008080">
<p align="center">
<%
'创建 Connection 对象
set mydata=server.createobject ("adodb.connection")
'连接数据源
mydata.connectionstring="DSN=mydata;UID=uid;PWD=pwd"
mydata.open
'对数据开始进行处理
mydata.begintrans
```

图 9-11 “留言簿”页面

```
'取出时间
sj = date ()
'cint 函数对数据类型进行强制转换
if cint (minute (time ())) > 9 then
    sj = sj&" "&hour (time ())&":"&minute (time ())
else
    sj = sj&" "&hour (time ())&":0"&minute (time ())
end if
'将保存到 Cookies 中的用户登陆信息取出来
if request.cookies ("custom") ("name2") < > "" then
    bname2 = request.cookies ("custom") ("name2")
    notes = trim (request.form ("ly"))
    nm1 = trim (request.form ("nm"))
    if notes = "" then
        response.write ("你的留言不可为空,请重新输入")
      else
        response.write ("留言成功！你的留言将成为我们最宝贵的财富。")
    %>
    < img src = "雪山 1.jpg" width = "75 %" height = "75 %" > < br > < br >
    <%
```

```
            '对数据库进行插入操作,其中 insertrec 表示对应的 sql 语句
            insertrec = "insert into 留言登记 (date1,name2,note3,toname)"
            insertrec = insertrec & " values ('" & sj & "','" & bname2 & "','" & notes &
            "','" & nm1 & "')"
        mydata.execute insertrec
        end if
else
        response.write "对不起,你不是心灵驿站的会员,不可留言 !"
end if

'对数据库进行更新
mydata.committrans
'关闭数据库
mydata.close
set mydata = nothing
%>
<a href = "./heartmemo.htm" > 返回主界面 </a >
<p align = "center" >
</p >
</body >
</html >
```

该程序中,主要利用 Connection 对象对数据库进行操作,其中插入操作是利用 SQL 语言中的 Insert 语句来实现的。这里只用到了这些对象的一部分属性和方法,这些对象有很多的属性和方法可以完成对数据库的操作。下面对这些对象的属性和方法作较为详细的介绍。

9.3.2　Connection 对象的属性

Connection 对象属性的名称及其含义,如表 9－2 所示。

表 9－2　Connection 对象属性的名称及其含义

名　称	含　义
Attributes	定义事务的处理方法
CommandTimeOut	允许数据源操作等待的最长时间
ConnectionString	设置数据库连接的信息
ConnectionTimeOut	设置利用 Open 方法连接的最长等待时间
CursorLocation	设置指针的处理位置
DefaultDatabase	指定 Connection 对象的默认数据库
IsolationLevel	指定事务的处理时间
Mode	控制 Connection 对象共享数据库的模式
Provider	返回或设置数据提供者的名称
Version	返回 ADO 对象的版本号

下面具体描述表 9－2 中的属性。

1．Attributes 属性

该属性定义了 Connection 对象的事务处理方法，控制着事务失败或成功后向数据库写数据的方式。该属性的名称及其含义，如表 9－3 所示。

表 9－3　Attributes 属性的名称及其含义

名　称	参数值	含　义
AdXactCommit Retaining	131072，对应的十六进制数为 0X00020000	表示当调用 CommitTrans 方法后不需要重新调用 BeginTrans 方法，Connection 对象会自动传送其他的运行请求
AdXactAbort Retaining	262144，对应的十六进制数为 0X00040000	表示当调用 RollbackTrans 方法后不需要重新调用 BeginTrans 方法，Connection 对象会自动传送其他的运行请求

2．CommandTimeOut 属性

该属性主要用来设置使用 Connection 对象的 Execute 方法运行的最长等待时间。可以利用该属性确定当链接业务量很大或服务器很忙时如何进行处理。该属性是一个长整形变量，其默认值为 30 秒，当其值设为 0 时，Connection 对象的操作将会运行到结束为止，相当于它没有时间限制。

需要指出的是，该属性的设置必须出现在运行之前，由于当数据库开始运行后，该属性会被指定为只读。

使用方式为：

```
<%
set mycon = server.createobject ("ADODB.Connection")
mycon.commandtimeout = 40
%>
```

3．ConnectionString 属性

ConnectionString 属性用来设置 Connection 对象的数据库链接信息。该属性的参数及其含义，如表 9－4 所示。

表 9－4　ConnectionString 属性的参数及其含义

参　数	含　义
DSN	数据源名
UID	访问数据源的账号
PWD	访问数据源口令
Provider	数据源提供者
File Name	指定数据源的某个特定文件

在 Connection 对象被打开之前，ConnectionString 属性是可读写的，但在 Connection 对象被打开后，它就变成“只读”了。

使用方式为：

```
<%
set mycon = server.createObject ("ADODB.Connection")
mycon.connectionstring = "DSN = 数据源名；UID = 账号；PWD = 口令；
provider = 提供者；file name = 文件名"
%>
```

4．ConnectionTimeOut 属性

该属性用来设置使用 Connection 对象的 Open ()方法执行的最长等待时间，也就是链接数据源的最长等待时间。该属性也是一个长整形变量，默认值为 15 秒，当其值设为 0 时，等于没有时间限制。使用方式同 CommandTimeOut 属性。

5．CursorLocation 属性

该属性主要用来设置数据库指针的处理位置，其属性的名称及其含义，如表 9－5 所示。

表 9－5　CursorLocation 属性的名称及其含义

名　称	参数值	含　义
AdUseClient	1	把数据库服务器的处理结果下载到客户端缓存保存
AdUseServer	2	把数据库服务器的处理结果保留在服务器端
AdUseClientBatch	3	可以动态地处理，当数据库服务器有更新或重新获得数据请求时，能重新与数据库建立链接

使用方式为：

```
<%
set mycon = server.createobject ("ADODB.Connection")
mycon.cursorLocation = aduseclient
%>
```

6．DefaultDatabase 属性

该属性用来指定 Connection 对象链接时的默认数据库，如果缺省数据库在 Connection 对象中的 DefaultDatabase 属性中定义，那么 SQL 语句就可以利用非限定的语法（SQL 语句不一定要数据库名）对数据库中的对象进行访问。

使用方式为：

```
<%
set mycon = server.createobject ("ADODB.Connection")
mycon.defaultdatabase = "mydata"
%>
```

7．IsolationLevel 属性

该属性用来设置 Connection 对象事务处理（Transaction）的时机。表 9－6 列出了它的属性的名称及其功能，其默认值为 AdXactCursorStability。

表 9－6　IsolationLevel 属性的名称及其功能

名　称	参数值（十进制）	功　能
AdXactUnspecified	－1	调用 BeginTrans 后立即开始事务处理工作
AdXactChaos	16	表示该 Connection 对象的传送可以覆盖掉其他层级的事务处理修改
AdXactBrowse	256	表示无法修改或覆盖较高级别的事务处理，有异常时，立即保存到数据库，并且其他用户可以实时取得修改后的数据
AdXactCursorStability	4096	所有的数据改变会自动处理，该项为默认选项
AdSactRepeatableRead	65536	表示当数据库有异常时，立即将数据保存到数据库，但用户无法取得修改后的数据，除非调用 Requery 方法
AdXactIsolated	1048576	表示当数据库有异常时，立即将数据保存到数据库，但用户无法取得修改后的数据，也不能调用 Requery 方法

8．Mode 属性

该属性用来设定用户用 Connection 对象共享数据库的模式。它的名称及其含义见表 9－7，默认值为 AdModeUnknown。

表 9－7　Mode 属性的名称及其含义

名　称	参数值	含　义
AdModeUnknown	0	表示未定义任何权限
AdModeRead	1	可提供其他 Connection 对象读权限
AdModeWrite	2	可提供其他 Connection 对象写权限
AdModeReadWrite	3	可提供其他 Connection 对象读写权限
AdModeShareDenyRead	4	拒绝其他 Connection 对象以读模式建立连接
AdModeShareDenyWrite	8	拒绝其他 Connection 对象以写模式建立连接
AdModeShareExclusive	12	拒绝其他 Connection 对象以读写模式建立连接
AdModeShareDenyNone	16	拒绝其他任何 Connection 对象建立连接

9．Provider 属性

该属性用来设置指定数据库管理者的名称，数据类型为字符串型。其中 ADO 的默认数据库管理者为 MSDASQL。

10．Version 属性

该属性主要用来标志 ADO 对象的版本，数据类型为字符串型。

9.3.3　Connection 对象的方法

Connection 对象方法的名称及其功能，如表 9－8 所示。

在 9.3 节的示例程序中，就用到了 BeginTrans、Open、Execute、CommitTrans 和 Close 方法。而 RollbackTrans 方法一般用于数据操作失败后所进行的操作。下面分别进行详细说明。

表 9－8 Connection 对象方法的名称及其功能

名 称	功 能
BeginTrans	开始处理事务
Close	关闭 Connection 对象和数据库的连接
CommitTrans	提交所有事务的处理结果
Execute	对数据库进行处理
RollbackTrans	放弃所有事务处理结果
Open	打开一个新的 Connection 对象

1．BeginTrans 方法

该方法可以方便地在同一时刻打开一个新的事务处理。所谓事务处理，就是打开了与数据库沟通的交互管道。只有当一个事务处理管道打开时，应用程序才可以对数据库中的记录进行更新。不过对数据库的更新并没有在磁盘上的数据库进行实际操作，直到提交事务处理（即 CommitTrans 方法执行）时才会对数据库进行修改。

当调用该方法时，返回一个长整型数值，以表明这个事务的等级。

使用方式为：

```
<% n = mydata.begintrans %>
```

不需要对返回值进行处理时，也可以利用下述方法：

```
<% mydata.begintrans %>
```

2．Close 方法

该方法主要用来关闭一事务处理与数据库的连接，会切断 Connection 对象与数据库之间的传送通道，同时所有依赖于该 Connection 对象的 Command 对象和 Recordset 对象也会立即被切断连接。该对象的操作与 Open 方法执行的操作是相反的，一般在对数据库操作完毕后，除执行 Close 方法操作外，还要利用“Set mydata = nothing”释放该 Connection 对象实例。

使用方式为：

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
mydata.begintrans
……
mydata.close
set mydata = nothing
%>
```

3．CommitTrans 方法

该方法主要用来提交所有事务处理的结果，对数据库进行的操作真正地反映在对数据库的实际修改上。该方法与 BeginTrans 方法相对应。

使用方式为：

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
mydata.begintrans
……
mydata.committrans
%>
```

利用上述方法的场合主要是防止 ASP 代码在执行时由于 SQL 语句的执行中断或异常而带来错误的执行结果。利用该语句结束事务时,如果这个事务的一个语句出现异常,那么所有的语句都将不执行。

4. Execute 方法

该方法主要用来进行数据库的数据操作。

使用方式为:

```
set mytab = mydata.execute (Query [Count [Options]])
```

其中,"["和"]"之间的项可选可不选;mytab 表示 Connection 对象在执行 Execute 方法后返回的 Recordset 结果;Query 表示对数据库执行的操作语句;Count 指定执行 Query 所返回或影响的记录数;Options 控制 Query 参数的数据查询类型。

例如:

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
mydata.begintrans
insertrec = "insert into 留言登记 (date1,name2,note3,toname)"
insertrec = insertrec & "values ('" & sj & "','" & bname2 & "','" & notes
    & "','" & nm1 & "')"
mydata.execute insertrec          '执行插入操作
mydata.comittrans
mydata.close
set mydata = nothing
%>
```

5. RollbackTrans 方法

该方法用于放弃所有事务处理,可以用来结束某个事务处理或取消所有更新数据的操作。

使用方式为:

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
```

```
mydata.open
mydata.begintrans
insertrec = "insert into 留言登记 (date1,name2,note3,toname)"
insertrec = insertrec & "values ('"& sj &"','"& bname2 &"','"
    & notes&"','"&nm1&"')"
mydata.execute insertrec
if mydata.errors.count > 0 then
    mydata.rollbacktrans          '取消插入操作
    response.write "操作有误"
else
    mydata.committrans            '提交操作
    response.write "操作成功"
end if
set mydata = nothing
%>
```

6. Open 方法

该方法主要用来打开一个新的 Connection 对象，即 Connection 对象与远端数据库服务器之间的连接操作。一旦建立连接后，相应的 Recordset 和 Command 对象的 ActiveConnection 属性值就是该 Connection 对象的名称。

使用方式为：

connection 对象实例名 .open (Data Source [User [Password]])

其中各参数及其含义，如表 9－9 所示。

表 9－9　open 方法的参数及其含义

参　数	含　义
Data Source	字符串型值，用于表示数据源识别字，可以自动写入 Connection 对象的属性之中
User	指定数据库用户名
Password	指定数据库用户的登陆密码

使用示例如下：

```
<%
set mydata = server.createobject "adodb.connection"
mydata.open"DSN = mydata ;UID = uid ;PWD = pwd"
mydata.begintrans
insertrec = "insert into 留言登记 (date1,name2,note3,toname)"
insertrec = insertrec & "values ('"& sj &"','"& bname2 &"','"
    & notes&"','"&nm1&"')"
mydata.execute insertrec
mydata.comittrans
```

```
mydata.close
Set mydata = nothing
%>
```

9.3.4 执行 SQL 语句

当数据库建立连接后，需要用 SQL 语句进行一系列操作。本节着重介绍在数据库中利用 SQL 语句进行查询、插入、删除等操作。

1. Select 语句

该语句的语法已经在前面有所介绍，下面着重介绍 Select 语句的应用。

使用示例如下：

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
command = "select * from 留言登记"
set savelb = mydata.execute (command)
while not savelb.eof
    response.write ("姓名为"&df ("name2"))
    savelb.movenext
wend
%>
```

2. Update 语句

如果需要对数据库进行修改，可以通过 SQL 语言来实现。

示例如下：

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
command = "update 留言登记 Set name = 'Mary' "
command = command&"where name = 'John' "
set savelb = mydata.execute (command)
while not savelb.eof
    response.write ("姓名为 "&df ("name2"))
    savelb.movenext
wend
%>
```

3. Delete 语句

如果要删除记录，也可以通过 SQL 语言来实现。

示例如下：

```
<%
set mydata = server.createobject ("adodb.connection")
mydata.connectionstring = "DSN = mydata ;UID = uid ;PWD = pwd"
mydata.open
command = "Delete 留言登记"
command = command&"where name = 'John' "
set savelb = mydata.execute (command)
while not savelb.eof
    response.write ("姓名为"&df ("name2"))
    savelb.movenext
wend
%>
```

9.4 Command 对象

ADO 的 Command 对象主要用来控制对数据库发出的请求信息，它提供了一种简单而有效的方法来处理对数据库的操作过程。本节详细介绍有关 Command 对象的方法、属性等内容。

9.4.1 创建 Command 对象

Command 对象可以利用已建立的 Connection 对象来创建，也可以单独创建。首先介绍一下如何利用 Connection 对象建立 Command 对象。

使用方式为：

```
<%
set mycon = server.createobject ("adodb.connection")
mycon.open"DSN = mydata ;UID = uid ;PWD = pwd"
……
set mycom = server.createobject ("adodb.command")
mycom.activeconnection = mycon
%>
```

再介绍单独创建的方法：

```
<%
set mycom = server.createobject ("adodb.command")
mycom.activeconnection = "DSN = mydata ;UID = uid ;PWD = pwd"
……
%>
```

9.4.2 Command 对象的属性

Command 对象的属性控制着对数据源进行操作的一切特性。表 9－10 简要介绍了有关 Command 对象属性的名称及其含义。

表 9－10 Command 对象属性的名称及其含义

名 称	含 义
ActiveConnection	建立与 Connection 对象的连接
CommandText	用来设定数据操作字符串
CommandTimeOut	开始执行查询信息后允许继续执行的最长时间
CommandType	指定数据查询信息的类型
Prepared	表示在命令执行前是否创建一预备语句

下面详细介绍它们的用法。

1．ActiveConnection 属性

该属性定义了 Command 对象的连接信息。该属性可以被设置或返回一个字符串，或者可以被指向一个已打开的 Connection 对象，也可以定义为一个新连接。

2．CommandText 属性

该属性为数据查询字符串，当 Execute 方法执行的时候，Command 对象把 CommandText 属性值作为默认的数据查询信息。

使用方式为：

```
<%
set mycom = server.createobject ("adodb.command")
mycom.activeconnection = "DSN = mydata ;UID = uid ;PWD = pwd"
……
mycom.commandtext = "select * from 留言登记"
set myrec = mycom.execute
%>
```

值得注意的是，当 Command 对象的 CommandText 属性为 SQL 语句或是一存储过程时，数据源提供者将会自动将该参数加到 Command 对象的 Parameters 集合中。

3．CommandTimeout 属性

该属性用来定义 Command 对象开始执行查询信息后允许继续执行的最长时间，它可以继承与它连接的 Connection 对象的这个属性值，但在自行设置后，与它连接的 Connection 对象的这个属性值将不会对它再起作用。

4．CommandType 属性

该属性用来指定数据查询信息的类型，用于优化数据源提供者的执行速度，其属性的名称及其含义，如表 9－11 所示。

表 9－11　CommandType 属性的名称及其含义

名　称	参数值	含　义
AdCmdText	1	指定数据查询信息类型为 SQL 语句
AdCmdTable	2	指定数据查询信息类型为数据库表名
AdCmdStore	4	指定数据查询信息类型为一存储过程
AdCmdUnknown	8	不可识别，该值为缺省值

5．Prepared 属性

Prepared 属性表示在执行命令之前是否要用命令创建一个预备语句。如果此属性被设置为 True，那么在 Command 命令执行前，CommandString 会被编译、优化和存储。尽管语句开始执行速度较慢，但在下一次使用该命令时，将会执行编译和优化过的语句。这样总的查询速度就提高了。

9.4.3　Command 对象的方法

Command 对象方法的名称及其功能，如表 9－12 所示，其作用是创建执行命令过程中所用到的对象。

表 9－12　Command 对象方法的名称及其功能

名　称	功　能
CreateParameter	用于建立一个新的 Parameter 对象
Execute	用于对数据库进行查询

1．CreateParameter 方法

该方法用来建立一个新的 Parameter 对象，并在执行前加到 Command 对象的 Parameter 集合中。Parameter 对象表示传递给 SQL 语句或存储进程的一个参数。

使用方式为：

```
<%
set mypara = mycom.createparameter (Name,Type,Direction,Size,Value)
%>
```

其中各参数及其含义，如表 9－13 所示。

表 9－13　CreateParameter 方法中的参数及其含义

参　数	含　义
Name	要创建的参数对象名称，字符串型
Type	定义了指定参数传递数据的类型
Direction	指定参数的方向
Size	指定允许传入参数的最大值
Value	指定传递给 Connection 对象的参数值

表 9－13 中的 Type 参数的名称及其含义，如表 9－14 所示。

表 9－14　Type 参数的名称及其含义

名　称	参数值	含　义
AdBigInt	20	八位的符号整数
AdBinary	128	二进制值
AdBoolean	11	布尔值
AdBSTR	8	以空值结束的字符串
AdChar	129	字符串值
AdCurrency	6	货币值
AdDate	7	日期值
AdDBDate	133	日期值（形式为 yyyymmdd）
AdDBTime	134	时间值（形式为 hhmmss）
AdDBTimeStamp	135	日期-时间值（形式为 yyyymmddhhmmss）
AdDBDecimal	14	有固定精确度和标度的数值
AdDouble	5	双精度浮点值
AdEmpty	0	无指定值
AdError	10	32 位出错代码
AdGUID	72	全局惟一指示符（GUID）
AdIDispatch	9	指向 OLE 对象中的接口指针
AdInteger	3	四位符号整数
AdUnknown	13	指向 OLE 对象的接口指针
AdLongVarBinary	205	一长二进制值（只适合于 Parameter 对象）
AdLongVarChar	201	一长字符串值（只适合于 Parameter 对象）
AdLongVarWchar	203	一以空值结束的长字符串值（只适合于 Parameter 对象）
AdNumeric	141	有固定精度和标度的值
AdSingle	4	单精度浮点值
AdSmallInt	2	两位符号整数
AdTinyInt	16	一位符号整数
AdUnsignedBigInt	21	八位无符号整数
AdUnsignedInt	19	四位无符号整数
AdUnsignedSmallInt	18	两位无符号整数
AdUnsignedTinyInt	17	一位无符号整数
AdUserDefined	132	用户定义的常量
AdVarBinary	204	二进制值（只适合于 Parameter 对象）
AdVarChar	200	字符串值（只适合于 Parameter 对象）
AdVariant	12	OLE 自制变量
AdVarWChar	202	以空值结束的字符串值（只适合于 Parameter 对象）
AdWchar	130	以空值结束的字符串值

Direction 参数的名称及其含义，如表 9－15 所示。

表 9－15 Direction 参数的名称及其含义

名 称	参数值	含 义
AdParamInput	1	输入参数，向存储进程传递信息
AdParamOutput	2	输出参数，从存储进程作为 SQL 输出的参数信息
AdParamInputOutput	3	输入和输出参数
AdParamReturnValue	4	用来读取从存储进程返回的状态值

2．Execute 方法

该方法用于向数据源提出数据查询。

使用方式为：

```
<%
Set myrec = mycom.execute (count,parameter,option)
%>
```

或者为：

```
<%
mycom.execute (count,parameter,option)
%>
```

其中各参数的名称及其含义，如表 9－16 所示。

表 9－16 Execute 方法中的参数的名称及其含义

名 称	含 义
Count	用于指定在数据查询发出后，数据库符合请求的所有数据总数，不过本属性可以省略不写
Parameter	参数值数组，也可以省略不写
option	该参数为 CommandType 属性值，本属性也可省略不写

下面举一个例子来说明：

```
<%
set mycon = createobject ("adodb.connection")
mycon.open "DSN = mydata;UID = wly;PWD = pwd"
set mycom = createobject ("adodb.command")
mycom.activeconnection = mycon
set mypara = mycom.createparameter "p1",adchar,adparaminput,4,"1000")
mycom.parameters.append mypara
set mypara = mycom.createparameter "p2",adchar,adparaminput,20,"上海")
mycom.parameters.append mypara
mycom.execute
mycom.parameters ("p1") = "2300"
```

```
mycom.parameters ("p2") = "太原"
mycom.execute
set mycon = nothing
set mycom = nothing
%>
```

9.4.4 Parameters 数据集合

Command 对象是通过 Parameters 集合传递参数的。

1. Parameters 集合的属性

在 Parameters 数据集合中，具有两个属性：Count 和 Item。

(1) Count 属性

该属性主要用来返回某个 Command 对象的参数个数。

使用方式为：

```
<%
n = mypara.count
%>
```

(2) Item 属性

该属性用来取得 Parameters 数据集合中所包含的 Parameter 对象。

使用方式为：

```
<%
set parameter 对象实例 = parameters.item (ndex)
%>
```

其中，index 是一整数值或字符串名称，作为 Parameters 数据集合内的索引。

不过程序设计人员也可以使用下述方法来实现：

```
set myp = mycom ()
set myp = mycom.parameters ()
set myp = mycom.parameters.item ()
```

上述三种方法的表示意义是一样的。当然也可以利用 Response 对象显示该内容。

使用示例如下：

```
<%
response.write mycom.parameters.item ()
%>
```

2. Parameters 集合的方法

Parameters 集合的方法及其功能，如表 9－17 所示。

表 9－17 Parameters 集合的方法及其功能

方 法	功 能
Append	该方法用于加入一个新的 Parameter 对象到数据集合中
Delete	该方法用于删除一个 Parameter 对象
Refresh	重新整理 Parameters 数据集合

(1) Append 方法

该方法用于加入一个新的 Parameter 对象到数据集合中。它的参数就是要添加的 Parameters 对象名。

使用方式为：

```
<%
parameters.append (parameter)
%>
```

当使用该方法把一个新的 Parameter 对象加入到 Parameters 数据集合中时，是放在该数据集合的末尾。

(2) Delete 方法

该方法可以把 Parameters 数据集合中的 Parameter 对象删除，它的参数就是要删除的 Parameters 对象名。

使用方式为：

```
<%
parameters.delete (parameter)
%>
```

当使用该方法删除一个 Parameters 对象时，我们可以把 Index 参数作为指向 Parameter 对象的索引值。

(3) Refresh 方法

该方法用于重新整理 Parameters 数据集合。

使用方式为：

```
<%
parameters.refresh
%>
```

当该方法被调用后，Command 对象会自动地对数据库控制器进行连接，并取回符合 CommandText 属性的所有 parameters 对象。

9.5 Parameter 对象

一个 Parameter 对象表示一个 Command 对象的相关参数，该对象主要用来负责记录所要

传递参数的相关属性。它的属性可以从传递给 Command 对象的 CreateParameter 方法的参数中继承下来。不过有一些属性不能从 CreateParameter 方法的参数中继承,因此一些 Parameter 对象的属性需要在对象建立后进行设置,才可以加入到 Parameters 数据集合中。

9.5.1 Parameter 对象的属性

该对象属性的名称及其含义,如表 9－18 所示。

表 9－18 Parameter 对象属性的名称及其含义

名 称	含 义
Attribute	指定该参数对象属性值的数值性质
Direction	控制该参数对象允许的读写模式
Name	指定该参数对象的名称
NumericScale	指定参数对象值小数点后允许的小数位数
Precision	允许参数值的最大数值
Size	允许参数值的字节组值大小
Type	允许参数值的数据类型
Value	参数值

1. Attribute 属性

该属性用于返回或设置它所能够接受的数据,该属性的值可以为表 9－19 中的某一值或多个值的和,其中该参数的默认值为 AdParamSigned。

表 9－19 Attribute 属性的名称及其含义

名 称	参数值	含 义
AdParamLong	128	允许参数值拥有相当大的值
AdParamNullable	64	该参数也可以接受空值
AdParamSigned	16	该参数可以接受有符号的数

2. Direction 属性

该属性用于控制该参数的读写模式,为整型数据。它的名称及其含义,如表 9－20 所示。

表 9－20 Direction 属性的名称及其含义

名 称	参数值	含 义
AdParamInput	1	允许数据输入该参数中
AdParamOutput	2	允许从该参数输出数据
AdParamInputOutput	3	允许数据输入、输出到该参数中
AdParamReturnValue	4	允许从子程序中返回数据到该参数中

3. Name 属性

该属性是一字符串值,用来定义 Parameters 集合中的 Parameter 对象的名称。在没有将该 Parameter 对象加入 Parameters 集合中时,它是可读写的。

4．NumericScale 属性

该属性包含字节值，用于指定参数对象值小数点后允许的小数位数。

5．Precision 属性

该属性包含字节值，用来设置或返回 Parameter 值的位数。

6．Size 属性

该属性为一整数值，可以读写，表示允许 Parameter 值的字节组值大小，即该对象的最大范围。需要指出的是，当 Parameter 对象加入 Parameters 集合时，Size 属性仍然是可读写的。

7．Type 属性

该属性是一整数类型值，可以读写，用来表示允许 Parameter 值的数据类型。当然该属性也可以从 Command 对象中继承而来。它的值可以为表 9－14 中的值。

8．Value 属性

该属性为一变量值，可以读写，用来定义要传递给 Command 对象绑定的查询或存储过程的值。

9.5.2　Parameter 对象的方法

Parameter 对象方法的名称及其功能，如表 9－21 所示。

表 9－21　Parameter 对象方法的名称及其功能

名　称	功　能
AppendChunk	用于加入新的数据到 Parameter 对象尾端
GetChunk	用于取得 Parameter 对象内的部分数据值

1．AppendChunk 方法

该方法用于加入新的数据到 Parameter 对象尾端，可以用来传递一个长文本信息或二进制数据。

使用方式为：

```
<%
parameter 对象 .appendchunk (data)
%>
```

其中 data 表示一个要传送至 Parameter 对象尾端的长文本信息或二进制数据。要说明的是，在使用该方法前最好把 Parameter 对象的 Attribute 属性设置为 AdParamLong，这样 Parameter 对象才能够接受该方法加入的长文本信息或二进制数据。另外，该方法仅允许同一时间内使用于一个 Parameter 对象。

2．GetChunk 方法

该方法主要用来取得 Parameter 对象内的部分数据值。

使用方式为：

```
<%
set vari = parameter 对象 .getchunk (count)
%>
```

其中，vari 表示返回数据的变量名称。count 用来指定想从 Parameter 对象中取得数据的大小，其为一长整型。

当使用该方法从 Parameter 对象中取回数据时，第一次调用会取回对象中的第一行，第二次调用会取回对象中的第二行，以此类推。

9.6 Errors 集合和 Error 对象

在 Connection 对象中有一个 Errors 集合，用来返回 ADO 在操作过程中所发生的错误。而这样的错误一般指某个操作所引起的错误。当在 ADO 运行期间发生了一个操作错误，则该错误对象将会被加入到 Errors 集合中，当另一个 ADO 操作引起错误时，原先的 Errors 集合将会被删除，新的 Errors 集合将会在 Connection 对象中建立。

9.6.1 Errors 集合的属性

在 Errors 集合中，常用到的属性的名称及其含义，如表 9－22 所示。

表 9－22 Errors 集合的属性的名称及其含义

名 称	含 义
Count	用来返回 Errors 集合中 Error 对象的数目
Item	用来返回 Errors 集合中某个具体的 Error 对象

使用示例如下：

```
<html>
<head>
<title>
检查
</title>
</head>
<body>
<%
set mycon=server.createobject("ADODB.Connection")
Mycon.open "DSN=mydata;UID=wly;PWD=pwd"
Set myerr=mycon.errors
Response.write "系统发现了"&myerr.count&"个错误<br>"
For I=0 to myerr.count-1
    Response.write "第"&I&"错误为"&myerr(I).description
Next
set mycon=nothing
%>
</body>
</html>
```

运行结果，如图 9－12 所示。

图 9－12　Errors 集合使用示例

9.6.2　Errors 集合的方法

在 Errors 集合中，常用到的方法只有一个：Clear 方法。
该方法用来清除 Errors 集合中的所有成员。
使用方式为：

```
<%
set myerr = mycon.errors
myerr.clear
%>
```

9.6.3　Error 对象的属性

该对象主要负责存储系统运行时发生的错误或警告信息，其属性的名称及其含义，如表 9－23 所示。

表 9－23　Error 对象属性的名称及其含义

名　称	含　义
Description	用来表示错误或警告信息发生的原因
Number	用来表示错误或警告信息的代码
Source	用来表示错误或警告信息的来源
NativeError	用来表示造成错误或警告信息的代码，指 Provider 默认的错误代码
SQLState	表示最近一次 SQL 命令运行状态
HelpContext	表示错误或警告信息的相应解决方法的描述
HelpFile	表示错误或警告信息的相应解决方法的帮助文件

9.7　Recordset 对象的处理

Recordset 对象是 ADO 中一个非常重要的对象，可以利用它操纵客户端的数据库接口。

就像在前面的例子一样，当需要察看数据库中符合条件的记录时，就是利用 Recordset 对象来处理的。因此，需要熟练地掌握它的用法。

9.7.1 Recordset 对象使用示例

在需要对留言簿中的内容进行浏览时，就可以使用以下方法（由于下面所述程序是镶在网站程序中的，因此不能单独运行，如果要运行，请稍作改动）。

程序 look.asp 如下：

```
<html>
<head>
<title>查看留言</title>
</head>
<body bgcolor="#d0d0d0">
<h2 align="center"><font face="华文行楷" color="#FF9900" size="7">留言表
    </font></h2>
<div align="center">
<%
nm1=request.cookies("custom")("name2")
'利用 Connection 对象建立与数据库的连接
set mydata=server.createobject("adodb.connection")
mydata.connectionstring="DSN=mydata;UID=uid;PWD=pwd"
mydata.open
set rs=server.createobject("adodb.recordset")
finds="select * from 留言登记 where toname='"&nm1&"' or toname='all'"
'利用 Recordset 对象保存 finds 执行的结果
set rs=mydata.execute(finds)
rs.movefirst
i=0
do while (not rs.eof and i<10)          '对 Recordset 对象内容进行显示
%>
    <table bordercolor="#999966" border="5" width=502 height=150
        style="background-color:#ffffff">
    <tr height=10%">
    <td width="60%"> 留言时间：<%=rs("")%></td>
    <td>留言人：<%=rs("")%></td>
    </tr>
    <tr>
    <td colspan=2 valign="top">留言：<%=rs("")%></td>
    </tr>
    </table><br>
<%
```

```
    i = i + 1
    rs.movenext
loop
rs.close
set rs = nothing
mydata.close
set mydata = nothing
%>
<a href = "heartmemo.htm" > 返回主目录 </a >
</div > </body >
</html >
```

在程序中，首先利用 Connection 对象建立与数据库的连接，然后将 SQL 语句的执行结果返回到 Recordset 对象中进行显示。程序执行结果，如图 9－13 所示。这里仅通过一个例子，使读者对 Recordset 的使用方法有一个大概了解。下面详细介绍该对象的属性。

图 9－13　‘查看留言’页面

9.7.2 Recordset 对象的属性

Recordset 对象属性的名称及其含义，如表 9－24 所示。

表 9－24 Recordset 对象属性的名称及其含义

名 称	含 义
AbsolutePage	当前页面的绝对位置
AbsolutePosition	当前数据库的绝对位置
ActiveConnection	与 Command、Connection 对象的连接
Bof	判断是否处于数据库记录集的开头
Bookmark	提供游标位置的书签功能
CacheSize	数据缓存区的大小
CursorLocation	用来控制游标的类型 :服务器端还是客户端
CursorType	控制从服务器端数据库取数据的类型
Eof	用来判断处于数据库记录集的结尾
EditMode	控制目前数据的处理状态
Filter	控制要显示或不显示的记录
LockType	控制数据更新的模式
MaxRecords	用来设定从服务器端取回的记录的最大数目
PageCount	用来设置显示数据页的数目
PageSize	用来控制每一数据页面的记录数
RecordCount	返回由服务器端数据库取回的记录总数
Source	数据查询信息
Status	在实施最近一次更改后，用来查询 Recordset 的状态

下面具体介绍这些属性的用法，供读者查阅。

1. Absolutepage 属性

该属性为一整数值，可以读写，表示的是当前页的绝对页数。它可以是从第一页到最后一页的页数值。另外，它还包括三个特殊值：

“- 1”表示无法取得该记录所在的页数的绝对位置；

“- 2”表示目前位置在所有数据行的开头；

“- 3”表示目前位置在所有数据行的结尾。

2. AbsolutePosition 属性

该属性为一整型数据类型，可以读写，用来表示当前记录行的绝对位置。它可以是 Recodset 对象内所有记录的每一行的位置，即从第一行到最后一行的对应的整数值。另外，它还可以为下列值：

“- 1”表示无法得知该记录行的绝对位置，对应名称为 AdPositionUnknown；

“- 2”表示目前记录行在所有记录行的前面，对应名称为 AdPosBOF；

“- 3”表示目前记录行在所有记录行的结尾，对应名称为 AdPosEOF。

当该属性被指定一新值时，Recordset 对象会把指针从目前记录行的位置移动到指定目标。

3．ActiveConnection 属性

该属性为字符串型数值，可以读写，用来定义 Recordset 对象与数据库提供者和数据库的连接。

使用方式为：

```
<%
set rs = server.createobject ("ADODB.Recordset")
rs.activeconnection = "DSN = mydata"
rs.open
……
%>
```

在 RecordSet 对象中，可以使用 Open 方法来完成数据库连接。当该方法执行时，如果不带任何参数，ActiveConnection 属性将会生效；但当在该方法内部带上 ActiveConnection 属性时，那么 ActiveConnection 属性的预设定将会被覆盖。

4．BOF 属性

该属性为布尔类型，可以为 True 或 False，只允许读，用来判断目前记录行是否在 Recordset 内部所有记录行的开头。

使用方式为：

```
<%
set rs = server.createobject ("ADODB.Recordset")
rs.activeconnection = "DSN = mydata"
rs.open
rs.movelast                              '指针移动到结尾
do while not rs.bof                      '显示记录
    response.write ("姓名为："&rs ("bname"))
    rs.moveprivious
loop
rs.close
set rs = nothing
……
%>
```

5．CacheSize 属性

该属性为一整数数据类型，可以读写，用来决定客户端每次从服务器端数据库取回的数据大小。可以通过设定客户端存储数据的内部缓冲 Cache 大小来决定。

6．CursorLocation 属性

该属性可以为从 1 到 3 之间的整数，可以读写，用来控制服务器端数据库返回数据的位置。该属性的名称及其含义，如表 9－25 所示。

表 9－25　CursorLocation 属性的名称及其含义

名　称	参数值	含　义
AdUseClient	1	下载服务器端数据库数据到客户端 Cache 中进行存取处理
AdUseServer	2	保存服务器端数据库的数据到服务器端进行存取处理
AdUseClientBatch	3	服务器端数据库的数据可以动态返回到客户端，停止连接进行数据处理，然后再连接数据库进行数据更新

7．CursorType 属性

该属性可以为从 1 到 4 之间的整数，可以读写，用来控制数据更新后过滤下载的数据类型。该属性的名称及其含义，如表 9－26 所示。

表 9－26　CursorType 属性的名称及其含义

名　称	参 数 值	含　义
AdOpenForwartOnly	1	为该属性的默认值，这种类型的游标只可以使用 Movenext 方法向后检索数据，暂时保存在 BookMark 中的数据无法再次获得
AdOpenKeySet	2	该类型游标可以见到 Recordset 对象创建后其他用户所做的修改，但是见不到其他用户增加和删除的记录
AdOpenDynamic	3	该类型游标是功能最强大、占资源最多的一种游标，所有数据库的操作都会立即反映在各个客户端上
AdOpenStatic	4	该类型游标无法见到 RecordSet 对象创建后其他用户所做的编辑、增加和删除

8．EditMode 属性

该属性可以为从 0 到 2 之间的整数，可以读取，用来显示目前数据的处理状态。该属性的名称及其含义，如表 9－27 所示。

表 9－27　EditMode 属性的名称及其含义

名　称	参数值	含　义
AdEditNone	0	该数据未被处理更改过
AdEditInProgress	1	该数据正在被处理，但还未被提交到数据源
AdEditAdd	2	该数据为刚存入数据库的记录行

9．EOF 属性

该属性与 BOF 属性一样，为布尔类型，可以为 True 或 False，用来判断目前的记录是否在结尾，即所有记录的后面。

使用方式为：

```
<%
set rs = server.createobject ("ADODB.Recordset")
rs.activeconnection = "DSN = mydata"
rs.open
'显示记录
do while not rs.eof
response.write ("姓名为:"&rs ("bname"))
rs.movenext
loop
rs.close
ser rs = nothing
……
%>
```

10．Filter 属性

该属性可以用一种简单的字符串来筛选记录，一般使用 SQL 语言来实现。

使用方式为：

```
<%
set rs = server.createobject ("ADODB.Recordset")
rs.activeconnection = "DSN = mydata"
rs.filter = "bname = Mary"
rs.open
……
%>
```

该属性也可以为从 0 到 3 之间的整数，用来控制数据的显示方式，该属性的名称及其含义，如表 9－28 所示。

表 9－28　Filter 属性的名称及其含义

名　称	参数值	含　义
AdFilterNone	0	表示无过滤功能，所有数据均可显示
AdFilterPendlingRecords	1	只允许没有更新、插入或删除的记录
AdFilterAffectedRecords	2	只显示最近一次使用 UpdateBatch、CancelBatch、Delete 或者 Resync 方法所更新的记录
AdFilterFetchedRecords	3	只显示目标存储在客户端 Cache 内的数据

11．LockType 属性

该属性可以为从 1 到 4 之间的整数，用来控制对数据库更新数据时的锁定机制。该属性的名称及其含义，如表 9－29 所示。

表 9－29　LockType 属性的名称及其含义

名　称	参数值	含　义
AdLockReadonly	1	规定 Recordset 对象以只读方式打开，即不允许作更新、插入或删除操作
AdLockPessimistic	2	规定数据在更新时需要锁定其他所有动作，也是最安全的锁定机制
AdLockOptimistic	3	规定数据在更新时并不锁定其他用户的动作，仍可进行操作
AdLockBatchOptimistic	4	规定数据在更新时并不锁定其他用户的动作，不过数据处理时需在 AdUseClientBatch 模式下才可进行操作。该方法可以加快更新 Recordset 对象

12．MaxRecords 属性

该属性为整数类型，可以读写，用于控制当 RecordSet 对象打开后，向数据库提出数据查询所返回的最大记录行。它的默认值为 0，表示返回符合条件的所有记录行。需要指出的是，在 Recordset 对象打开前，该属性是可读写的，但在打开后就变成只读的了。

13．PageCount 属性

该属性为整数类型，可以读取，表示目前 Recordset 对象获得的数据页总数目。如果 Recordset 对象的最后一页未满，那么其中的记录以附加页计算。

14．PageSize 属性

该属性为整数类型，可以读写，用来指定 Recordset 对象中每一数据页内允许的数据记录数目。

15．RecordCount 属性

该属性为整数类型，可以读取，表示返回 Recordset 对象中的记录数。不过该属性表示的记录数并不准确，因此一般不使用该属性。

上述三个属性，存在以下换算关系：

PageCount = (RecordCount + PageSize - 1) / PageCount。

16．Source 属性

该属性用于表示 Command 对象或数据查询字符串，可以读写，用来表示在服务器端数据库检索数据的方法。

使用方式为：

```
<%
set rs = server.createobject ("ADODB.RecordSet")
rs.activeconnection = "DSN = mydata"
rs.source = "Select * from 留言簿"
rs.open
……
%>
```

17. Status 属性

该属性为整数值,可以读取,用来表示目前数据的状态,对于多用户连接管理是特别有用的。该属性的名称及其含义,如表 9－30 所示。

表 9－30 Status 属性的名称及其含义

名 称	参数值	意 义
AdRecOK	0	记录更新成功
AdRecNew	1	表示该行数据是新加入的数据,不过还未加入到数据库中
AdRecModified	2	表示该行数据已被修改,不过还未反映到数据库中
AdRecDeleted	4	该记录已被修改
AdRecUnmodified	8	该记录尚未被改动过
AdRecInvalid	16	无法获得记录,故更新失败
AdRecMultipleChanges	64	更新数据库失败,需要通过多重记录才可编辑
AdRecPendingChanges	128	更新数据库失败,由于该记录的修改要依靠其他待改变的记录
AdRecCanceled	256	由于放弃数据库连接而导致数据库更新失败
AdRecCantRelease	1024	由于数据库锁定而导致更新失败
AdRecConCurrencyViolation	2048	由于数据库类型为 AdLockOptimistic,导致数据库更新失败
AdRecIntegrityViolation	4096	由于记录违背了完整性规则,导致数据库更新失败
AdRecMaxChangsExceeded	8192	由于许多记录要更新或加入,导致数据库更新失败
AdRecObjectOpen	16384	由于与其他的 Recoreset 对象冲突导致失败
AdRecOutOfMemory	32768	由于服务器端内存不足,导致数据库更新错误
AdRecPermissionDenied	65536	由于访问者缺少权限导致数据库更新错误
AdRecSchemaViolation	131072	该记录与数据库结构不符合而导致数据库更新错误
AdRecDBDeleted	262144	由于该记录已被删除,导致数据库更新失败

当我们要判断 Recordset 对象的状态是否是 AdRecDeleted 时,可以使用下列方式:

```
<%
set rs = server.createobject ("ADODB.RecordSet")
rs.activeconnection = "DSN = mydata"
rs.source = "Select * from 留言簿"
rs.open
if rs.status and adrecdeleted then
  ……
```

```
end if
……
%>
```

9.7.3 Recordset 对象的方法

Recordset 对象方法的名称及其功能，如表 9－31 所示。

表 9－31 Recordset 对象方法的名称及其功能

名 称	功 能
AddNew	添加一个新的记录行
CancelBatch	取消批量（batch）上传的数据更新模式
CancelUpdate	取消数据更新的操作
Clone	用来建立一个相同的 Recordset 对象
Close	用于关闭目前的 Recordset 对象
Delete	用于删除目前的记录行
GetRows	用来取得多条记录
Move	将指针移动到指定的记录行位置
MoveNext	将指针移动到下一行记录行位置
MovePrevious	将指针移动到上一行记录行位置
MoveFirst	将指针移动到第一个记录行位置
MoveLast	将指针移动到最后一行记录行位置
NextRecordset	允许读取另外一个 Recordset 对象
Open	执行数据库查询数据的动作
Requery	重新获得 Recordset 对象的内容
Resync	用来设置与数据库服务器进行数据同步的模式
Supports	可以用来判断当前 Recordset 对象支持的功能
Update	更新目前的数据到服务器数据库上
UpdateBatch	可以进行批量的数据更新动作

1．AddNew 方法

该方法可以将一个或数行数据添加到数据库中。

使用方式为：

```
<%  myrec.addnew (fields,values)  %>
```

其中，fields 表示单一的字段名或由多个字段名构成的数组，values 表示单一的数据值或由多个数据值构成的数组。

下面举一个例子来进行讲解：

```
<%
dim fields (2)
dim values (2)
set rs = server.createobject ("ADODB.RecordSet")
rs.activeconnection = "DSN = mydata"
```

```
rs.source = "Select * from 留言簿"
rs.open
'在 rs 中添加记录
fields (0) = "姓名"
fields (1) = "学号"
values (0) = "郑洁"
values (1) = "9628"
rs.addnew fields,values
'利用 update 方法将数据更改传到数据库
rs.update
……
%>
```

2．CancelBatch 方法

该方法用来取消批量传送模式的数据更新动作。

使用方式为：

```
<% myrec.cancelbatch %>
```

需要声明的是，Recordset 对象的 LockType 需要设定为 AdLockBatchOptimistic，这样当 UpdateBatch 方法执行后，在 Recordset 对象内进行的数据更新、删除等操作才可以用 CancelBatch 方法来取消。

3．CancelUpdate 方法

该方法用来取消数据更新的操作。

使用方式为：

```
<% myrec.cancelupdate %>
```

在执行 Update 方法前，可以使用 CancelUpdate 方法取消已更新数据的操作。

4．Clone 方法

该方法主要用来建立一个与当前 Recordset 对象相同的 Recordset 对象。

使用方式为：

```
<% set newrec = myrec.clone %>
```

需要指出的是，newrec 与 myrec 的属性分别进行保持，但其中任何一个 Recordset 对象有改动，都会在另外一个 Recordset 对象上有所体现。但当关闭 Recordset 对象时，需要分别对它们进行关闭。

5．Close 方法

该方法用来将已打开的 Recordset 对象关闭。其实该方法在前几节的讲述中已有所使用。

使用方式为：

```
<% myrec.close %>
```

6．Delete 方法

该方法用来删除目前位置的记录行。

使用方式为：

```
<% myrec.delete [AdAffectCurrent | AdAffectGroup] %>
```

其中所用到的参数名称及其含义，如表 9－32 所示。

表 9－32　Delete 方法的参数名称及其含义

名　称	参数值	含　义
AdAffectCurrent	1	设定该属性会删除当前位置索引的记录行
AdAffectGroup	2	删除符合 Filter 属性指定的记录行子集合

其中，前者为默认值。需要说明的是，当 Recordset 对象的 LockType 设定为 AdLockBatchOptimistic，而且 UpdateBatch 还没有执行，那么已被删除的内容可以被恢复。

7．GetRows 方法

该方法可以用来读取多个数据行。

使用方式为：

```
<% set vararray = myrec.getrows (count,start,fields) %>
```

其中，用到的参数名称及其含义，如表 9－33 所示。

表 9－33　GetRows 方法中的参数名称及其含义

名　称	含　义
VarArray	由 GetRows 返回的由多行记录构成的数组
Count	设定要取得记录行数据的行数，可以不设定，默认值为 －1，表示取回 Recordset 对象内全部记录行数据
Start	设定要取得数据的开始位置，可以不设定，默认为从当前位置索引记录行开始
Fields	设定要取回的数据字段，可以不设定，默认值为所有字段

8．Move 方法

该方法用来将指针移动到指定记录行位置。

使用方式为：

```
<% myrec.move count,start %>
```

其中各参数的名称及其含义，如表 9－34 所示。

表 9－34　Move 方法中的参数的名称及其含义

名　称	含　义
Count	表示要移动的相对于基准位位置的相对位置
Start	表示要定位位置，可选择的，默认值为当前位置，例如 Start 为 1 时，表示以下一记录行为基准位指针

使用该方法前要注意，Recordset 对象的 CursorType 属性不可以设为 AdOpenForward。

9．MoveNext 方法

该方法用来将指针移动到下一行记录行处。

使用方式为：

```
<% myrec.movenext %>
```

在使用该方法之前要判断是否处于 Recordset 对象内的最后一行，这可以利用 Eof 属性来实现。

10．MovePrevious 方法

该方法用来将指针移动到上一行记录行处。

使用方式为：

```
<% myrec.moveprevious %>
```

在使用该方法之前要判断是否处于 Recordset 对象内的开头，可以利用 Bof 属性来实现。要使用该方法，Recordset 对象的 CursorType 属性不可以设定为 AdOpenForward。

11．MoveFirst 方法

该方法用来将指针移动到第一行记录行处。

使用方式为：

```
<% myrec.movefirst %>
```

注意，这时 Recordset 对象的 CursorType 属性不可以设定为 AdOpenForward。

12．MoveLast 方法

该方法用来将指针移动到最后一行记录行处。

使用方式为：

```
<% myrec.movelast %>
```

13．NextRecordset 方法

该方法用来移动多重数据查询时取得的另外一个 Recordset 对象。

使用方法为：

```
<% set newrec = myrec.nextrecordset (affected) %>
```

其中，参数 Affected 表示要移动到的那个 Recordset 对象的索引整数数字。另外，当下一个 Recordset 对象不存在时，将会返回 nothing，并赋予 newrec 上。

14．Open 方法

该方法用于打开一个向数据库执行数据查询的动作。

使用方式为：

```
<% myrec.open (source,connection,cursor,lock,type) %>
```

其中各参数的名称及其含义，如表 9－35 所示。

表 9－35　Open 方法中的参数的名称及其含义

名　称	含　义
Source	可以是 Command 对象名或数据查询字符串,可以省略不写
Connection	可以是 Connection 对象名或数据库链接信息字符串,可以省略不写
Cursor	用来设定 CursorType 属性值,可以省略不写
Lock	用来设定 LockType 属性值,可以省略不写
type	用来设定 Command 对象的 CommandType 属性,可以省略不写

当上述参数省略不写时,都认为是预设定的该 Recordset 对象的对应属性内容。

15．Requery 方法

该方法用来重复执行数据库查询的动作。

使用方式为：

```
<% myrec.requery %>
```

使用该方法后,所有已存在于 Recordset 对象内的数据将会更新。通过这种方法,可以看到自上一次数据查询以来进行的一系列操作。该命令相当于该 Recordset 对象被关闭后又重新打开。

16．Resync 方法

该方法主要用来设置与服务器数据库同步的更新模式。

使用方式为：

```
<% myrec.resync [adAffectCurrent | adAffectGroup | adAffectAll] %>
```

其中用到的参数的名称及其含义,如表 9－36 所示。

表 9－36　Resync 方法的参数的名称及其含义

名　称	参数值	含　义
AdAffectCurrent	1	只对当前记录行做同步更新的操作
AdAffectGroup	2	对所有合乎条件的记录行做同步更新的操作
AdAffectAll	3	对所有 Recordset 对象内的数据做同步更新的操作

使用 Resync 方法可以容易地进行重复的数据查询操作。

17．Supports 方法

该方法用来显示目前 Recordset 对象支持的功能。

使用方式为：

```
<% mysup = myrec.supports (option) %>
```

在上述方式中,mysup 是调用该方法的返回值,为布尔型（True 或 False）。Option 表示 Support 方法支持的参数设定功能,可以设定的参数的名称及其含义,如表 9－37 所示。

表 9－37 Supports 方法的参数的名称及其含义

名 称	参数值	含 义
AdAddNew	16778240	支持 AddNew 方法
AdApproxPosition	16384	可以读取并设置 AbsolutePosition 和 AbsolutePage 属性
AdBookMark	8192	支持 Bookmark 功能
AdDelete	16779264	支持 Delete 方法
AdHoldRecords	256	支持多重分离记录行区块存取
AdMovePrevious	512	支持 MovePrevious 方法
AdResync	131072	允许使用 Resync 方法
AdUpdate	16809984	允许使用 Update 方法
AdUpdateBatch	65536	允许使用 UpdateBatch 方法

例如，要察看目前的 Recordset 对象是否支持 AddNew 和 Bookmark 功能，可以通过 Support 方法来实现。

使用方式为：

```
<%
set rs = server.createobject ("ADODB.RecordSet")
rs.activeconnection = "DSN = mydata"
rs.source = "Select * from 留言簿"
rs.open
……
if rs.supports (AdAddNew + AdBookmark) then
  ……
end if
……
%>
```

18．Update 方法

该方法用来更新目前的数据到服务器端数据库上。

使用方式为：

```
<%
myrec.update (fields,values)
%>
```

其中，fields 表示单一的字段名或由多个字段名构成的数组，values 表示单一的数据值或由多个数据值构成的数组。不过这两个参数可以被省略，这时该方法将会把对数据库进行所有的修改传到数据库中。

19．UpdateBatch 方法

该方法用来在批量传送模式下进行数据更新的操作。

使用方式为：

```
<%
```

```
myrec.updatebatch [adAffectCurrent | adAffectGroup | adAffectAll]
%>
```

其中各参数的名称及其含义,如表 9－38 所示。

表 9－38　UpdateBatch 方法的参数的名称及其含义

名　称	参数值	含　义
AdAffectCurrent	1	只对当前记录行做数据更新的操作
AdAffectGroup	2	对所有合乎条件的记录行做数据更新的操作
AdAffectAll	3	对所有的 Recorset 对象内的记录行做数据更新的操作,该参数为缺省值

9.7.4　Fields 集合

其实当 Recordset 对象实例创建后,它的每一列都对应着一个字段。而 Fields 集合就对应地包含着所创建的 Recordset 对象实例中的所有字段。

该集合主要包含两个属性：

1．Count 属性

该属性表示 Recordset 对象实例中的字段个数。

例如,当要显示 Recordset 对象中所有字段所对应的值,可以使用以下方式：

```
<%
for i = 0 to myrec.fields.count
  response.write myrec ()
next
……
%>
```

2．Item 属性

该属性可以取得 Fields 集合中的指定字段内容。以下方式可以显示数据库中某个字段的内容：

```
myrec.fields.item ()
myrec.fields.item "bname"
myrec.fields ()
myrec.fields "bname"
```

以上方式是等效的。

该集合只有一种方法：Refresh 方法,用来强制 Fields 集合重新取得所需字段内容。

使用方式为：

```
<%
myrec.fields.refresh
%>
```

9.7.5 Field 对象

1. Field 对象的属性

Field 对象属性的名称及其含义，如表 9－39 所示。

表 9－39 Field 对象属性的名称及其含义

名 称	含 义
ActualSize	表示当前字段内的数据值长度
Attributes	表示当前字段内的数据值属性
DefinedSize	表示当前字段内所定义的长度
Name	表示当前字段对象所指的字段名称
NumericScale	表示当前字段对象所指的字段数据类型
OraginalValue	表示当前字段对象修改前的字段名值
Precision	表示当前字段对象所允许存放数字的最大值
UnderlyingValue	表示当前字段对象在数据源内的字段值
Value	表示当前字段对象所指的字段值

(1) ActualSize 属性

该属性是只读属性，可以返回 Recordset 对象的当前记录一字段的实际长度。使用方式为：

```
<%
mylen = myrec.fields ("bname").actualsize
%>
```

(2) Attributes 属性

该属性的名称及其含义，如表 9－40 所示。

表 9－40 Attributes 属性的名称及其含义

名 称	参数值	含 义
AdFldMayDefer	2	不允许重新由数据库更新数据
AdFldUpdatable	4	不允许被修改
AdFldUnknownUpdatable	8	不可决定是否可被更改
AdFldFixed	16	允许的数值长度已被修改
AdFldIsNullable	32	可以被设为空值
AdFldMayBeNull	64	可以从该字段中读取空值
AdFldLong	128	允许存放相当大的数据类型值
AdFldRowID	256	可存放某种记录的 ID
AdFldRowVersion	512	显示最后一次的修改时间
AdFldCacheDeferred	4096	当存取时可由 Cache 内取出

使用方式为：

```
<%
```

```
……
if rs ("banme").attributes and adfldlong then
……
end if
……
%>
```

(3) DefinedSiz 属性

该属性是一整数值,可以读取,表示目前记录指定字段所定义的长度。

(4) Name 属性

该属性是一字符串值,可以读取,表示目前 Field 对象所指定的字段名称。当要取出所调用数据库某表中的字段名时,可以使用以下方式:

```
<%
for i = 0 to myrec.fields.count - 1
    response.write myrec ().name
next
%>
```

(5) NumericScale 属性

该属性是一个字节值,可读取,用来表示目前 Field 对象所指字段内所允许存放数字的大小数目限制,即用来设定字段内允许数字的位数。

(6) OriginalValue 属性

该属性用来返回修改前的 Field 字段值。需要说明的是,使用 CancelUpdate 和 CancelBatch 方法实际上就是用该属性值进行恢复。

(7) Precision 属性

该属性是一个字节值,可读写,用来表示目前 Field 对象所指字段内所允许存放数字的最大值。该值需要与 NunericScale 属性值对应起来,不可发生冲突。

(8) UnderlyingValue 属性

该属性用来在单个字段实现与 Resync 方法同样的功能,可以参阅 Resync 方法。

(9) Value 属性

该属性可以读写,用来返回当前记录行在对应字段内的值,并且该属性是 Field 对象的缺省属性。

当要对某 Recordset 对象进行显示字段值时,可以通过该方法来实现。

使用方式为:

```
<%
set rs = server.createobject ("ADODB.RecordSet")
rs.activeconnection = "DSN = mydata"
rs.source = "Select * from 留言簿"
rs.open
do while not Rs.eof
    for i = 0 to rs.fields.count - 1
```

```
        response.write rs().value
    next
    rs.movenext
loop
……
%>
```

2．Field 对象的方法

在 Field 对象中，常用到的方法的名称及其功能，如表 9－41 所示。

表 9－41 Field 对象方法的名称及其功能

名 称	功 能
AppendChunk 方法	用于合并数据到现有数据尾端
GetChunk 方法	用于取得数据内的合并数据

（1）AppendChunk 方法

该方法用于合并数据到现有数据尾端。

使用方式为：

```
<%
myrec().appendchunk Type
%>
```

其中，参数 Type 表示欲传送到目前字段数据尾端的数据类型。另外，需要注意的是，仅可在同一时间对一字段进行 AppendChunk 操作，否则，将会带来数据混乱。

（2）GetChunk 方法

该方法可以用来从字段内取出合并数据。

使用方式为：

```
<%
myrec().getchunk(count)
%>
```

其中，参数 count 表示要从指定字段内取回的字节数。

9.8 与 Access 和 SQL Server 数据库连接

本章的前面，着重讲解了 ADO 五大对象的使用方法。由于数据库的创建方法不同，所以与数据库进行连接也有所不同。本节仅就 MS Access 和 MS SQL Server 的连接进行讲解。

9.8.1 与 Access 数据库连接

当在计算机上装载 Office 办公自动化软件时，将会为系统安装 MS Access 数据库软件。安装完毕后，就会在‘开始’菜单中的‘程序’选项中发现‘Microsoft Access’选项。

在利用该软件创建一个数据库后，将会以 *.mdb 形式保存。然后需要利用 ODBC 将 ADO 与数据库连接起来，具体操作方法见 9.1 节。

不过这个过程也可以通过下述代码来实现（其实在代码中使用了原始的由 OLE DB 提供的驱动程序）：

```
<%
set mycon = server.createObject ("ADODB.Connection")
mycon.open "(PROVIDER = MICROSOFT.JET.OLEDB.4.0;
DATA SOURCE = D:/mydatabase/mydata.mdb")
……
%>
```

当需要在程序中创建一个 Access 数据库时，需要使用 ADOX。例如，要在 D：\mydatabase\下创建数据库文件 mydata1.mdb 时，可以使用下列代码来实现：

```
<%
dbnm = "d:/mydatabase/mydata1.mdb"
'创建 ADOX.Catalog 对象实例
set mydb = server.createobject ("ADOX.Catalog")
'利用 Create 方法创建新的 Access 数据库
mydb.create "(PROVIDER = MICROSOFT.JET.OLEDB.4.0;DATA SOURCE = "&dbnm
'利用 Connection 对象打开与数据库的连接
set mycon = server.createobject ("ADODB.Connection")
mycon.open "(PROVIDER = MICROSOFT.JET.OLEDB.4.0;DATA SOURCE = "&dbnm
……
%>
```

9.8.2 与 SQL Server 数据库连接

目前较大的网站都使用 SQL Server 数据库软件创建数据库。在使用 SQL Server 数据库前需要利用 ODBC 进行连接。

例如，可以使用下述代码建立与 SQL Server 数据库的连接：

```
<%
set mycon = server.createobject ("ADODB.Connection")
mycon.open "(PROVIDER = SQLOLEDB;DATA SOURCE = dataserver;
    UID = sa;PWD = pwd;DATABASE = mydata")
%>
```

在上述使用示例中用到了一些参数，各参数的名称及其含义，如表 9－42 所示。

表 9－42 数据源连接时各参数的名称及其含义

名 称	含 义
Provider	用来规定这次连接使用到的由 OLE DB 提供的程序名称
Data Source	用来表示数据库所处位置,如果数据库在本机器上,设为 Local Server,如果位于 DataServer 上,则设定为 DataServer
UID	用来表示访问数据库的访问名
PWD	提供对应访问名的注册密码
Database	用来设定位于数据库服务器上的一特定数据库名

9.9 习题

1. 基础题

① （ ）是 Microsoft 公司开发的与数据库进行连接通信的接口,它提供了用（ ）语言和数据库通信的一般标准。

② 当我们对数据库进行查询时,在 SQL 语言中用到的语句为（ ）语句,进行插入用到的是（ ）语句。

③ （ ）对象是负责针对数据库的实际连接操作;它是利用 Server 对象的（ ）方法来建立的。

④ 在 Connection 对象中,（ ）属性是用来定义 Open 方法最长的执行时间,（ ）是用来定义 Execute 方法最长的执行时间。

⑤ 在 Connection 对象中,可以将出现的错误保存在（ ）集合中。

2. 操作题

① 利用 MS Access 建立一数据库,并利用 ODBC 将 ADO 与数据库连接起来。

② 利用 Connection 对象连接数据库,并利用 Recordset 对象将符合条件的记录保存起来,并显示在客户端的浏览器上。

③ 利用 SQL 语句实现数据项的添加或删除操作。

④ 利用以上的语句实现数据库的管理。

第十章　制 作 实 例

10.1　数据库的管理

在一般的网站中,经常需要管理许多数据库。下面我们来看一个简单的例子。

1. **主程序** LoginRecord.asp

```
'计算访问次数
<%
dim num
num = request.cookies("visit_num")
if num <> "" then
    num = cint(num) + 1
    num = cstr(num)
    response.cookies("visit_num") = num
    response.write"你是第"&num&"次访问本站点"
else
    num = "1"
    response.cookies("visit_num") = num
    response.write"欢迎你首次访问本站点"
end if
%>
<html>
<head>
<title>选择方式</title>
<bgsound src="C05-08.mp3" loop="-1">
</head>
<body style="border-style:ridge;border-color:#008000"
    bgcolor="#FFFFFF">
<p> </p>
<p>        
<div align="center">
<center>
<table border=2>
<a href="http://zbf1/recordren.asp">登记进入</a>
<b><font color="#008000">欢迎你加盟到我们的集体中</font></b><br>
```

```
<a href="http://zbf1/browserecord.asp">浏览记录</a>
<b><font color="#008000">欢迎你浏览我们的记录</font></b><br>
<a href="http://zbf1/searchrecord.asp">查询记录</a>
<b><font color="#008000">欢迎你查询你所需要的记录</font></b><br>
<a href="http://zbf1/modify1.asp">修改记录</a>
<b><font color="#008000">欢迎你进行你信息的修改</font></b>
</table>
</center>
</div>
<p>  </p>
<p><marquee direction=right width="480" height="20" >
<b>我是你永远的朋友
</b></marquee></p>
</body>
```

程序显示结果，如图 10－1 **所示。**

图 10－1　**数据库操作主页面**

2．**登记程序** RecordRen.asp

```
<html>
<head>  <title>人员登记</title>  </head >
<body bgcolor="#d0d0d0">
```

```
<b>请你开始登记：</b><br>
<form action="modifydata.asp" method="get" name="mytable"
    style="position:relative">
工号：<input type="text" name="numb" value=""><br>
姓名：<input type="text" name="nm" value=""><br>
性别：<input type="radio" name="xb" value="male">男
    <input type="radio" name="xb" value="female">女<br>
工龄：<input type="text" name="gl" ><br>
基本工资：<input type="text" name="jbgz" ><br>
总津贴：<input type="text" name="zjt"><br>
<input type="submit" name="br" value="提交">
<input type="reset" name="br" value="重置"><br>
<a href="http://zbf1/loginrecord.asp">返回主界面</a>
</form>
</body>
</html>
```

对应的 ModifyData.asp 程序如下：

```
<html>
<head> <title>数据库访问</title> </head>
<body>
<%
set mycon=server.createobject("adodb.connection")
mycon.connectionstring="DSN=mydata;UID=uid;PWD=pwd"
mycon.open
mycon.begintrans
gh=request.querystring("numb")
if gh="" then
    response.write("对不起,你未输入工号")
else
    xm=request.querystring("nm")
    xb=request.querystring("xb")
    gl=request.querystring("gl")
    jbgz=cdbl(request.querystring("jbgz"))
    zjt=cdbl(request.querystring("zjt"))
    zsr=jbgz+zjt
    response.write(xm&"的总收入是"&zsr&"元"&"<br>")
    response.write(xm&"的基本工资是"&jbgz&"元"&"<br>")
    response.write(xm&"的总津贴是"&zjt&"元")
    sqlstmt="insert into 职员收入表"
    sqlstmt=sqlstmt&" values("&gh&"','"&xm&"','"&xb&"','"&gl&"','"
        &jbgz&"','"&zjt&"','"&zsr&"')"
    set rs=mycon.execute(sqlstmt)
```

```
        set rs = nothing
        if err.number > 0 then
            response.write"illegal error number"&err.number&"< br >"
        end if
    end if
    mycon.committrans
    mycon.close
    set mycon = nothing
    %>
    < br > < a href = "http ://zbf1/recordren.asp" > 返回 < /a >
    < marquee scrollamount = 5 align = botton > 欢迎你！ <% = xm %> ！ < /marquee >
    < /body > < /html >
```

显示程序登记页面，如图 10－2 所示。

图 10－2　“人员登记 ’页面

3．浏览记录的程序 BrowseRecord.asp

```
    < html >
    < head >
    < title > 浏览记录 < /title >
    < bgsound src = "C04 - 10.mp3" loop = " - 1" >
    < /head >
    < body bgcolor = " # d0d0d0" >
    <%
    set mycon = server.createobject ("adodb.connection")
    mycon.open"DSN = mydata ;UID = uid ;PWD = pwd"
    set rstemp = mycon.execute ("select * from 职员收入表")
```

```
count = rstemp.fields.count - 1
%>
< table border = 1 > < tr >
<% for i = 0 to count %>
    < td > < b >
    <% = rstemp (i).name %>
    < /b > < /td >
<% next %>
< /tr >
<%
do while not rstemp.eof
%>
< tr >
<%
    for i = 0 to count
        thisvalue = rstemp (i)
        if isnull (thisvalue) then
            thisvalue = " "
        end if
%>
        < td valign = top > <% = thisvalue %> < /td >
<%
    next %>
    < /tr >
<%
    rstemp.movenext
loop
%>
< /table >
<% rstemp.close
set rstemp = nothing
mycon.close
set mycon = nothing
%>
< a href = "http://zbf1/loginrecord.asp" > 返回 < /a >
< /body >
< /html >
```

显示页面，如图 10－3 **所示。**

4．**查询记录的程序** SearchRecord.asp

```
< html >
< head >
< title > 查询记录 < /title >
```

图 10－3　'浏览记录 '页面

```
</head>
<body bgcolor="#d0d0d0">
<form action="search.asp" method="get" name="mysearch">
请输入职工工号下限：<input type="text" name="nn1" value=""><br>
请输入职工工号上限：<input type="text" name="nn2" value=""><br>
<input type="submit" name="bt" value="提交">
<input type="reset" name="bt" value="重置"><br>
</form>
<a href="http://zbf1/loginrecord.asp">返回主界面</a>
</body>
</html>
```

相应的查询实现程序 Search.asp 如下：

```
<html>
<head>
<title>回应查找记录</title>
</head>
<body>
<% num1=trim(request.querystring("nn1"))
  num2=trim(request.querystring("nn2"))
```

```
  set mycmd = server.createobject ("adodb.recordset")
  mycmd.activeconnection = "dsn = mydata ;UID = uid ;PWD = pwd"
  mycmd.source = "select * from 职员收入表 where 职员工号 > = '"&num1&"' and 职员工
     号 < = '"&num2&"'"
  mycmd.cursortype = adopenkeyset
  mycmd.open
  counter = mycmd.fields.count - 1
  recnum = 0
  if mycmd.eof and mycmd.bof then
     response.write ("对不起！本记录不存在！"&" < br > ")
else
%>
    < br > < table border = 1 > < tr >
<%
    for i = 0 to counter
%>
        < td > < b >
        <% = mycmd (i).name %>
        < /b > < /td >
    <% next %>
    < /tr >
    <% do while not mycmd.eof %>
        < tr >
        <% for i = 0 to counter %>
            < td valign = top > <% = mycmd.fields (i).value %>   < /td >
        <% next %>
        < /tr >
        <% mycmd.movenext
           recnum = recnum + 1
    loop
    %>
    < /table >
    < br > < b > 本查询共有 <% = recnum %> 条记录 < /b > < br >
<% end if
mycmd.close
set mycmd = nothing
%>
< a href = "http ://zbf1/searchrecord.asp" > 重新输入 < /a >
< /body >
```

'查询记录'页面，如图 10－4 所示。

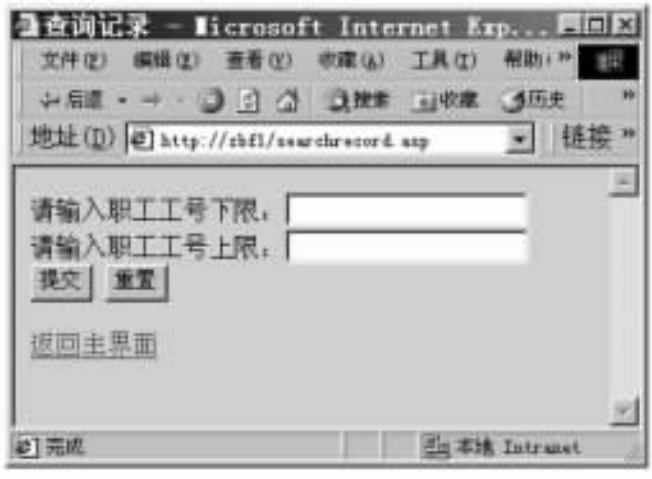

图 10－4　‘查询记录’页面

5．**修改记录的程序** Modify1.asp

```
<html>
<head>
<title>修改记录</title>
</head>
<body bgcolor="#d0d0d0">
<form action="modifystr.asp" method="get" name="mysearch">
请输入修改的职工工号：<input type="text" name="nnn" value=""><br>
<input type="submit" name="bt" value="提交">
<input type="reset" name="bt" value="重置"><br>
</form>
<a href="http://zbf1/loginrecord.asp">返回主界面</a>
</body>
</html>
```

相应的数据处理程序 ModifyStr.asp **如下：**

```
<html>
<head>
<title>回应修改记录</title>
</head>
<body>
<%
dim nu3
nu3=trim(request.querystring("nnn"))
set mycom1=server.createobject("adodb.recordset")
mycom1.activeconnection="DSN=mydata;UID=uid;PWD=pwd"
mycom1.cursortype=adopenkeyset
```

```
mycom1.source = "select * from 职员收入表 where 职员工号 = '"&nu3&"'"
mycom1.open
recnum = mycom1.fields.count - 1
if mycom1.eof and mycom1.bof then
    response.write"不存在查找记录"
else
    for i = 0 to recnum
%>
        <% = mycom1 ().name %>
        < input type = "text" name = nm value = <% = mycom1.fields ().value %> >
        < br >
  <% next
end if
%>
< input type = "submit" name = qd value = "完成" >
<%
mycom1.close
set mycom1 = nothing
%>
< br > < a href = "http ://zbf1/modify1.asp" > 返回输入 < /a >
< /body >
< /html >
```

‘修改记录 ’页面,如图 10 - 5 **所示。**

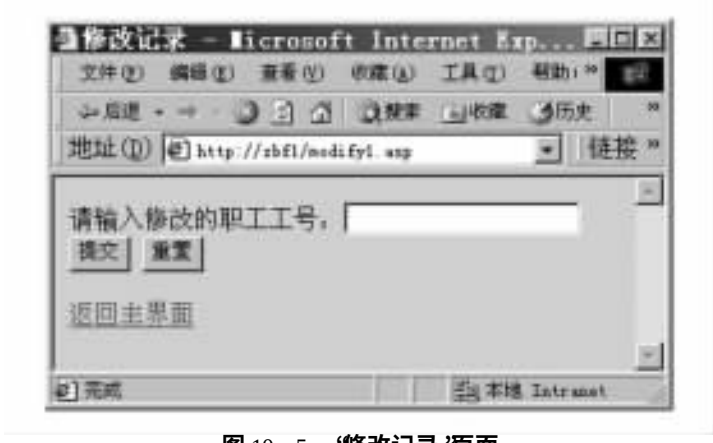

图 10 - 5 **‘修改记录 ’页面**

10.2 网上商店的实现

下面给出的是一个 ‘网上商店 ’的例子。程序执行时的页面,如图 10 - 6 **所示。**

图 10－6　**‘网上商店’主页面**

购物车中物品的查询结果，如图 10－7 **所示。**

图 10－7　**购物车页面**

访问者购买结束后会进入结账处，结账处显示页面，如图 10－8 所示。

图 10－8　结账处页面

不过需要事先介绍一下数据库的结构，该数据库是使用 MS Access 数据库软件来创建的。‘商品仓库’的页面，如图 10－9 所示。

图 10－9　‘商品仓库’页面

程序实例如下：

```
<% response.buffer = True %>
<html>
<head>
```

```
<title>网上商店</title>
<base target="_parent">
<style fprolloverstyle>A:hover {color:#008000;font-size:1em;text-
    decoration:blink}
</style>
</head>
<body style="color:#008000;font-family:华文新魏;border:5 ridge
    #008000;padding:2" topmargin="2" leftmargin="2" background=
    "images/200006053553.jpg">
<img src="logo11.gif" width="130" height="35">
<%

'添加选择物品到购物车中
sub additemtocart(Itemid,iItemcount)
    getitemparameters(Itemid)
    if flag=1 then
        '将数据库中对应的物品数量减1
        mycon.begintrans
        sqlstr="update 商品仓库 set 商品数量=商品数量-"&iItemcount&"where 商
        品序号="&cstr(Itemid)
        set rm=mycon.execute(sqlstr)
        set rm=nothing
        mycon.committrans
        '将选择物品加入购物车中
          if dictcart.exists(Itemid) then
              dictcart(Itemid)=dictcart(Itemid)+iItemcount
          else
              dictcart.add iItemid,iItemcount
          end if
    else
          response.write("对不起,本商品脱销")
    end if
end sub

'取消某物品的购买
sub removeitemfromcart(Itemid,iItemcount)
    '将数据库中对应物品数量加1
    mycon.begintrans
    sqlstr="update 商品仓库 set 商品数量=商品数量+"&iItemcount&" where 商品序
        号="&cstr(Itemid)
    set rm=mycon.execute(sqlstr)
    set rm=nothing
    mycon.committrans
```

```
    '将购物车中对应物品去掉
        if dictcart.exists (Itemid) then
          if dictcart (Itemid) < = iItemcount then
            dictcart.remove iItemid
        else
            dictcart (Itemid) = dictcart (Itemid) - iItemcount
        end if
        response.write iItemcount&"of item # "&iItemid&" have been removed,
            < br > "
    else
        response.write "can not find any item"
    end if
end sub

'显示购物车中已购买物品信息
sub showitemsincart ()
dim key
dim apara
dim stotal,shipping
% >
< div valign = middle align = center >
< font size = 10 > 你的购物车 < /font >
< table border = 2 bordercolor = " # 008080" bordercolorlight = " # 000000" >
< tr >
< td > 商品序号 < /td >
< td > 商品名称 < /td >
< td > 购买数量 < /td >
< td > 取消购买 < /td >
< td > 商品价格 < /td >
< td > 总价位 < /td >
< /tr >
< %
stotal = 0
for each key in dictcart
    apara = getitemparameters (key)
% >
     < tr >
    < td > < % = key % > < /td >
    < td > < % = apara (0) % > < /td >
    < td > < % = dictcart (key) % > < /td >
    < td >
    < a href = "./shopping.asp ? action = del&item = < % = key % > &count = 1" > 取消
        一件 < /a >   ;  ;
```

```
    <a
    href="./shopping.asp?action=del&item=<%=key%>&count=<%=
    dictcart(key)%>">取消所有</a>
    </td>
    <td>$<%=apara()%></td>
    <td>$<%=formatnumber(dictcart(key)*apara())%></td>
    </tr>
    <%
    stotal=stotal+(dictcart(key)*apara())
next
if stotal<>0 then
    shipping=7.5
else
    shipping=0
end if
stotal=stotal+shipping
%>
<tr>
<td colspan=5 align="right"><b>B+H</b></td>
<td>$<%=formatnumber(shipping,2)%></td>
</tr>
<tr>
<td colspan=5 align="right"><b>total:</b></td>
<td>$<%=formatnumber(stotal,2)%></td>
</tr>
</table>
<%end sub

'显示所有网上商店商品信息
sub showfullcatalog %>
<div valign=middle align="center">
<font size=12>网上商店</font>
<table border=2 bordercolor="#008080">
<tr>
<td>名称</td>
<td>价格</td>
<td>产地</td>
<td>购买确认</td>
</tr>
<%
i=1
myrecordset.movefirst
do while not myrecordset.eof
```

```
    apara = getitemparameters (i)
    %>
    <tr>
    <td> <% = apara (0) %> </td>
    <td> $ <% = apara (1) %> </td>
    <td> <% = apara (2) %> </td>
    <td > <a href = "./shopping.asp?action = add&item = <% = i %> &count = 1" >
        购买一件 </a> </td>
    </tr>
    <%
    i = i + 1
    myrecordset.movenext
loop %>
</table>
<% = SessionID %>
<a href = "./shopping.asp?action = viewcart" > 购物车预览 </a>
<a href = "heartmemo.htm" > 返回主界面 </a>
</div> <br>
<marquee direction = right > <b > 欢迎你的大驾光临 </b> </marquee>
<%
end sub

'进行结账处理
sub placeorder ()
dim key
dim apara
dim stotal,shipping
%>
<div valign = middle align = "center" >
<font size = 10 > 结账处 </font>
<table border = 2 bordercolor = "#008080" >
<tr>
<td> 商品序号 </td>
<td> 商品名称 </td>
<td> 商品数量 </td>
<td> 商品价格 </td>
<td> 总价位 </td>
</tr>
<% stotal = 0
for each key in dictcart
    apara = getitemparameters (key)
%>
    <tr>
```

```
    <td><%=key%></td>
    <td><%=apara(0)%></td>
    <td><%=dictcart(key)%></td>
    <td>$<%=apara(1)%></td>
    <td>$<%=formatnumber(dictcart(key)*apara(1))%></td>
    </tr>
    <%stotal=stotal+(dictcart(key)*apara(1))
next
if stotal<>0 then
    shipping=7.5
else
    shipping=0
end if
stotal=stotal+shipping
%>
<tr>
<td colspan=4><b>S+H</b></td>
<td>$<%=formatnumber(shipping,2)%></td>
</tr>
<tr>
<td colspan=4><b>total</b></td>
<td>$<%=formatnumber(stotal,2)%></td>
</tr>
</table>
<%
session.abandon
end sub

'得到商品消息
function getitemparameters(Itemid)
dim apara
do while myrecordset(0)<>iItemid
    myrecordset.movenext
    if myrecordset.eof then
        myrecordset.movefirst
    end if
loop
apara=array(myrecordset(0),myrecordset(1),myrecordset(2))
if myrecordset(4)<=0 then
    flag=0
    else flag=1
end if
getitemparameters=apara
```

```
end function
%>
<%
dim dictcart
dim saction
dim iItemid
dim iItemcount
dim flag
set mycon = server.createobject ("adodb.connection")
mycon.connectionstring = "dsn = mydata ;pwd = pwd ;uid = wly"
mycon.open
set myrecordset = server.createobject ("adodb.recordset")
myrecordset.cursortype = adopenkeyset
myrecordset.activeconnection = "dsn = mydata ;pwd = pwd ;uid = wly"
myrecordset.source = "select * from 商品仓库"
myrecordset.open
if isobject (session ("cart")) then
    set dictcart = session ("cart")
else
    set dictcart = server.createobject ("scripting.dictionary")
end if
saction = cstr (request.querystring ("action"))
iItemid = cint (request.querystring ("item"))
iItemcount = cint (request.querystring ("count"))
%>
<%
select case saction
    case "add"
        additemtocart iItemid,iItemcount
        showfullcatalog
    case "del"
        removeitemfromcart iItemid,iItemcount
        showitemsincart
        %>
  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;  ;
        < a href = "./shopping.asp ? action = " > 继续购物 < /a >
        < a href = "./shopping.asp ? action = checkout" > 结账处 < /a >
        < br >
        <%
    case "viewcart"
        showitemsincart
        %>
        < a href = "./shopping.asp ? action = " > 继续购物 < /a >
```

```
        <a href="./shopping.asp?action=checkout">结账处</a><br>
        </div>
        <marquee direction=right><b>你的满意是我们服务的宗旨！</b>
        </marquee>
        <%
    case "checkout"
        placeorder
%>
        <a href="heartmemo.htm"><img src="images/3.gif"></a>
        <br>
        <h2>欢迎你的光临,谢谢！</h2>
        </div>
        <marquee direction=right><b>一路走好！</b></marquee>
        <%
        myrecordset.close
        set myrecordset=nothing
    case else
        showfullcatalog
end select
set session("cart")=dictcart
%>
</body>
</html>
```

附录　VBScript 中的常用函数

1．Abs 函数

语法：

```
Abs (num)
```

参数 num 表示一数值，该函数返回 num 的绝对值。如果 num 的值为 Null，则返回 Null；如果 num 是没有初始化的变量，则返回 0。

2．Array 函数

语法：

```
Array (arglist)
```

参数 arglist 是一系列用逗号隔开的数值，这些值是被指定为 Variant 中数组的元素。如果没有指定参数，将会建立零长度的数组。

注解：

```
'建立一个名为 A 的 Variant 量
Dim A
'指定一个数组给变量 A
A = Array (0,20,30)
'将数组中第二个元素给 B
B = A (1)
```

3．Asc 函数

语法：

```
Asc (string)
```

参数 string 是任何可用的字符串表达式，并且其长度必须大于零。该函数用于返回字符串中第一个字母的 ANSI 码。

4．Atn 函数

语法：

```
Atn (num)
```

参数 num 可以是任何有效的数值表达式，该函数用于返回参数的反正切值（arctangent），不过函数返回值是以弧度为单位的。

5．CBool 函数

语法：

```
CBool (exp)
```

参数 exp 可以是任何有效的数值表达式。

该函数可以将参数 exp 转换为 Boolean 型。如果 exp 为 0,则返回 False ;否则返回 True。

6. CByte 函数

语法:

```
CByte (exp)
```

参数 exp 可以是任何有效的数值表达式。

该函数用于强制进行 Byte 计算,即将任何一种数据转换至 Byte 型。不管单精度、双精度数还是整数,都可以正常进行。不过不同的小数点分隔符号会依据操作系统中的国别设定来自动确认,千分位分隔符号也一样。

7. CCur 函数

语法:

```
CCur (exp)
```

参数 exp 可以是任何有效的数值表达式。

该函数用于返回一个转换为 Currency 类型的表达式。

我们可以使用该函数来强制执行 Currency 计算,以免被当作整数运算。该函数可以使用在任何语言的版本中,让任何一种数据转换为 Currency 类型,小数点分隔符号、千分位分隔符号和货币符号会依据操作系统中的国别设定来自动确认。

8. CDate 函数

语法:

```
CDate (date)
```

参数 date 为任何有效的日期表达式。

该函数用于返回一个转换为 Date 类型的表达式。

注解:

可以使用 IsDate 函数来检查 date 是否可以被转换为日期或时间。而 CDate 接受日期文字和时间文字,以及日期/时间有效范围内的适当数值。当转换一个数字为日期时,是将数字部分转换为日期,任何数字中的小数部分将转换为从午夜起算得的时间。另外它将会自动依据操作系统中的设定来作适当的格式设置。

9. CDbl 函数

语法:

```
CDbl (exp)
```

参数 exp 表示一个有效的表达式。

该函数用于返回一个已转换为 Double 类型的表达式。

10. Chr 函数

语法:

```
Chr (char)
```

参数 char 表示一个数值，用来识别某个字符。

该函数用于返回 char 所代表的字符。

注解：

0 到 31 之间的数字与一般非打印的 ASCII 码相同，例如 Chr（0）代表换行字符。

11．CInt 函数

语法：

```
CInt (exp)
```

参数 exp 表示任何有效的表达式。

该函数用于返回一个转换为 Integer 类型的表达式。

注解：

可以使用 CInt 或 CLong 函数来强制执行整数运算，以免当作货币值、单精度或双精度运算。如果 exp 超出 Integer 类型允许的范围，将会发生错误。

另外需要注意的是：CInt 函数与 Fix 及 Int 函数不同，Fix 及 Int 函数会将小数部分去掉，然后返回整数值。但当小数部分为 0.5 时，CInt 函数将会转换为最接近的偶数值，例如，CInt（0.5）= 0，CInt（1.5）= 2。

12．CLong 函数

语法：

```
CLong (exp)
```

参数 exp 表示任何有效的表达式。

该函数用于返回一个转换为 Long 类型的表达式。

注解：

可以使用 CInt 或 CLong 来强制执行整数运算，以免当作货币值、单精度或双精度运算。如果 exp 超出 Long 类型允许的范围，将会发生错误。

13．Cos 函数

语法：

```
Cos (num)
```

参数 exp 表示任何有效的数值表达式。其值代表一个角度，以弧度为单位。

该函数用于返回一角度的余弦值。返回值在 - 1 和 1 之间。

14．CreateObject 函数

语法：

```
CreateObject (class)
```

参数 class 使用 Servername.TypeName 语法，并且具有下列几个部分：

ServerName：提供该对象的应用程序的名称；

TypeName：表示所要建立的类型或对象名称。

例如：

```
Dim ExcelSheet
```

```
'建立 MS Excel 表的对象
Set ExcelSheet = CreateObject("Excel.Sheet")
'通过 Application 对象来显现 Excel
ExcelSheet.Application.Visible = True
'输入表格中内容
ExcelSheet.Cells(,1).Value = "Welcome you !"
'将这个表格存档
ExcelSheet.Saveas "C:\Excel1.Xls"
'使用 Application 对象的 Quit 方法来关闭 Excel
ExcelSheet.Application.Quit
'释放 Excel 对象变量
Set ExcelSheet = nothing
```

15. CSng 函数

语法：

```
CSng(exp)
```

参数 exp 表示任何有效的数值表达式。

该函数用于返回一个转换为 Single 类型的表达式。

注解：

使用 CDbl 或 CSng 方法强制进行双精度或单精度运算，以免被当作货币值或整数运算。如果 exp 超过 Single 类型允许的范围，将会发生错误。

16. CStr 函数

语法：

```
CStr(exp)
```

参数 exp 表示任何有效的表达式。

该函数用于返回一个转换为 Single 类型的表达式。

注解：

可以使用 CStr 函数强制进行 String 类型转换。参数 exp 将会决定函数返回值是什么值。可以参考如下说明：

如果 retexp 内容为返回值，则：

Boolean 包含 "True" 或 "False" 的字符串；

Date 包含一个日期的字符串；

Null 表示产生执行时期错误；

Empty 是一个长度为零的字符串（""）；

Error 包含一个英文字 "Error" 以及一个错误代码的字符串。

17. DateAdd 函数

语法：

```
DateAdd(nterval,number,date)
```

该函数将会返回一个基准日期加上或减去数个时间间隔单位后的日期。

各参数表示下列含义：

interval 为必选项，为一个字符串表达式，表示要加上去的时间间隔单位。

其设定值如下：

yyyy 表示年；

q 表示季；

m 表示月；

y 表示一年的日数；

d 表示日；

w 表示一周的日数；

ww 表示周；

h 表示时；

n 表示分；

s 表示秒。

number 为必选项，为一个数值表达式，表示要加上去的时间间隔次数。

date 为必选项，其类型为 Variant (Date) 的变量或正确表示日期的字符串，代表基准日期，将此基准日期加上 number 次的 interval 后就是返回值日期。

注解：

可以使用 DateAdd 函数计算基准日期加上或减去所指定的时间间隔后的结果。返回值一定是有效且准确的。

例如：

```
NewDate = DateAdd ("m",1,"31 - Jan - 1995")
```

DateAdd 将会返回 28-Feb-1995，而不会是 31-Feb-1995。

18．DateDiff 函数

语法：

```
DateDiff (nterval,date1,date2",firstdayofweek",firstweekofyear"")
```

该函数将会返回两个日期间相差的时间间隔单位数目。

各参数含义如下：

interval 为必选项，为字符串表达式，用来表示计算 date1 和 date2 之间时间差的时间间隔单位。

其设定值如下：

yyyy 表示年；

q 表示季；

m 表示月；

y 表示一年的日数；

d 表示日；

w 表示一周的日数；

ww 表示周；

h 表示时；

n 表示分；

s 表示秒。

参数 firstdayofweek 的设定值，如表附-1 所示。

表附-1 参数 firstdayofweek 的设定值

名 称	参数值	含 义
VBUseSystem	0	使用 NLS API 设定
VBSunday	1	星期日（默认值）
VBMonday	2	星期一
VBTuesday	3	星期二
VBWednesday	4	星期三
VBThursday	5	星期四
VBFriday	6	星期五
VBSaturday	7	星期六

参数 firstweekofyear 的设定值，如表附-2 所示。

表附-2 参数 firstweekofyear 的设定值

名 称	参 数 值	含 义
VBUseSystem	0	使用 NLS API 设定
VBFirstJan1	1	包含一月一日的星期（默认值）
VBFirstFourDays	2	第一个至少包含此年度四天的星期
VBFirstFullWeek	3	第一个完整的星期

19．DatePart 函数

语法：

```
DatePart (interval,date",firstdayofweek",firstweekofyear"")
```

该函数用于返回指定日期的某个时间部分。

各参数设置与 DateDiff 函数类似。

20．DateSerial 函数

语法：

```
DateSerial (year,month,day)
```

该函数用于返回一 Variant (date)中内容为指定的年、月、日的日期表达式。

各参数的含义如下：

year 为必选项，为从 100 到 9999 的整数，或是一整数表达式；

month 为任何数值表达式；

day 为任何数值表达式。

注解：

不过需要说明的是，日和月所赋予的值必须是合理的。

21．DateValue 函数

语法：

```
DateValue (date)
```

该函数用来返回一 Variant (date)类型的日期表达式。

注解：

如果 date 包含时间，则 DateValue 会把时间部分舍去。但如果 date 只含有时间而无日期，会产生程序错误。

22．Day 函数

语法：

```
Day (date)
```

该函数用于返回一个从 1 到 31 之间的值，表示一个月中的某一日。

Date 可以是任何表达式，只要能够表示一个合理的日期即可。如果 date 为 Null，则返回 Null。

23．Exp 函数

语法：

```
Exp (num)
```

参数 num 可以是任何的数值表达式。

该函数将会返回 e (自然对数的底数)的 num 次方。

24．Filter 函数

语法：

```
Filter (nputstr,value",include",compare"")
```

该函数将会返回一个以零为基数的数组，其内容为符合条件的字符串子集合。

各参数的含义如下：

inputstr 为必选项，为被搜寻的一维字符串；

value 为必选项，为所要寻找的字符串；

include 为可选项，以 Boolean 值表示是否返回包含或不包含 Value 的字符串；

compare 为可选项，以数值来指定字符串的比较方式，其取值如表附-3 所示。

表附-3　compare 参数的取值

名　称	参数值	含　义
VBBinaryCompare	0	进行二进制比较
VBTextCompare	1	进行纯文字比较
VBDatabaseCompare	2	进行数据比较

注解：

如果不包含 Value 所表示的字符串，那么在 inputstr 中 Filter 函数将会返回一个空字符串。如果 inputstr 为 Null 或不是一维数组，则会发生错误。

25. Int、Fix 函数

语法：

```
Int (num)
Fix (num)
```

该函数用于返回 num 的整数部分。

参数 num 表示任何有效的数值表达式。如果 num 为 Null，则返回 Null。

26. FormatCurrency 函数

语法：

```
FormatCurrency (exp",numdigits ",includedigit",useparens ",groupdigits"")
```

该函数用于返回一个以系统控制中设定货币符号来格式化的货币值表达式。

各参数含义如下：

exp 为必选项，表示欲被格式化的表达式；

numdigits 为可选项，表示小数位数，默认值为 - 1，表示使用系统设定值；

includedigit 为可选项，以 Tristate 常数来表示是否显示‘前导零’，可以设定的值如表附-4所示；

useparens 为可选项，以 Tristate 常数来表示负数值是否带有括号，可以设定的值如表附-4所示；

groupdigits 为可选项，也以 Tristate 常数来表示是否以‘数位群组符号’进行分隔，视不同的系统设定而有不同。

表附-4 Tristate 常数

名 称	参数值	含 义
TristateTrue	- 1	True
TristateFalse	0	False
TristateUseDefault	- 2	使用系统设定值

27. FormatDateTime 函数

语法：

```
FormatDateTime (Date",NameFormat")
```

该函数用于返回日期或时间格式的表达式。

各参数的含义如下：

Date 为必选项，为欲格式化的日期表达式；

NameFormat 为必选项，以数值来表示所使用的日期/时间格式，可以设定的值如表附-5所示。

表附-5 NameFormat 参数的取值

名 称	参数值	含 义
VBGeneralDate	0	显示日期和(或)时间,将以简短日期格式显示日期,以完整时间显示时间,可以两者都显示
VBLongDate	1	以系统完整日期格式设定值来显示日期
VBShortDate	2	以系统简短日期格式设定值来显示日期
VBLongTime	3	以系统时间格式设定值来显示时间
VBShortTime	4	以 24 小时制格式(hh:mm)来显示时间

28. FormatNumber 函数

语法:

```
FormatNumber (exp",numdigits ",includedigit",useparens ",groupdigits""""")
```

该函数用于返回一个以系统控制中格式化的数字。

各参数含义如下:

exp 为必选项,表示欲被格式化的表达式;

numdigits 为可选项,表示小数位数,默认值为 - 1,表示使用系统设定值;

includedigit 为可选项,以 Tristate 常数来表示是否显示'前导零",可以设定的值如表附-6所示;

useparens 为可选项,以 Tristate 常数来表示负数值是否带有括号,可以设定的值如表附-6所示;

groupdigits 为可选项,也以 Tristate 常数来表示是否以'数位群组符号'进行分隔,视不同的系统设定而有不同。

表附-6 Tristate 常数

名 称	参数值	含 义
TristateTrue	- 1	True
TristateFalse	0	False
TristateUseDefault	- 2	使用系统设定值

29. FormatPercent 函数

语法:

```
FormatPercent (exp",numdigits ",includedigit",useparens ",groupdigits""""")
```

该函数用于返回带有"%"的格式化百分比表达式。

各参数含义如下:

exp 为必选项,表示欲被格式化的表达式;

numdigits 为可选项,表示小数位数,默认值为 - 1,表示使用系统设定值;

includedigit 为可选项,以 Tristate 常数来表示是否显示'前导零",可以设定为表附-7 中的值;

useparens 为可选项，以 Tristate 常数来表示负数值是否带有括号，可以设定为表附-7 中的值；

groupdigits 为可选项，也以 Tristate 常数来表示是否以‘数位群组符号’进行分隔，视不同的系统设定而有不同。

表附-7 Tristate 常数

名 称	参数值	含 义
TristateTrue	-1	True
TristateFalse	0	False
TristateUseDefault	-2	使用系统设定值

30. Hex 函数

语法：

```
Hex (num)
```

该函数将 num 以十六进制表示，用 String 来返回。

参数 num 可以是任何的合法表达式。如果 num 不是一个整数，将会进行四舍五入计算。可以将十六进制数以 &H 开头来表示。

31. Hour 函数

语法：

```
Hour (time)
```

该函数用于返回一个在 0 到 23 之间的值，表示一天中的某时。

32. InputBox 函数

语法：

```
InputBox (prompt",title"",default"",xpos"",ypos"",helpfile,context")
```

该函数用于显示一个对话框来让访问者输入信息，并将返回输入的内容。

各参数如下所示：

prompt 是一个字符串表达式，用来作为对话框信息的字符串表达式，其最大长度大约为 1024 个字符。换行可以通过 chr(13)、chr(0)来实现。

title 显示为对话框标题。缺省值为应用程序的名称。

default 为显示在文字方块中的字符串表达式。当没有 default 时，则文字方块为空白的。

xpos 为数值表达式，用来指定对话框的左边和屏幕的左边的水平距离。如果无设置，则对话框会出现在水平方向的中间。

ypos 为数值表达式，用来指定对话框的上边缘和屏幕的上边缘的垂直距离。如果无设置，则对话框会出现在屏幕垂直方向的三分之一的位置。

helpfile 为字符串表达式，用来指定对话框的帮助文件。如果 helpfile 被指定，则 context 也必须被指定。

context 为数值表达式，由帮助文件来指定给某个说明主题的代码。与 helpfile 必须一同指定，并且对话框会指定一个帮助按钮。

33．InStr 函数

语法：

```
InStr ([start,]string1,string2[,compare])
```

该函数将会返回在某字符串中一字符串最先出现的位置。

各参数的含义如下：

start 为选择项，为一数值表达式，用来设定每次搜寻的起点，如果省略，将会从第一个字符开始，并且如果有 compare 参数，则一定会有 start 参数；

string1 为必选项，欲进行搜寻的字符串；

string2 为必选项，欲进行搜寻的另一个字符串；

compare 为选择项，设定字符串比较种类，如果省略，将会进行二进制比较。

参数 compare 的设定值，如表附-8 所示。

表附-8　参数 Compare 的设定值

名　称	参数值	含　义
VBBinaryCompare	0	进行二进制比较
VBTextCompare	1	进行字符比较
VBDatabaseCompare	2	执行数据内容比较

Instr 函数在不同的条件下有不同的返回值，如表附-9 所示。

表附-9　Instr 函数的返回值

条　件	返　回　值
String1 长度为 0	0
String1 为 Null	Null
String2 长度为 0	Start
String2 为 Null	Null
String2 找不到	0
在 String1 中找到 String2	找到的位置
Start > Len (String2)	0

34．InstrRev 函数

语法：

```
InstrRev (string1,string2[,start[,compare]])
```

该函数将会返回在某字符串中一字符串最先出现的位置，从尾端开始搜寻。

各参数的含义如下：

string1 为必选项，欲进行搜寻的字符串；

string2 为必选项，欲进行搜寻的另一个字符串；

start 为选择项，为一数值表达式，用来设定每次搜寻的起点，如果省略，为 -1，将会从最后一个字符开始，如果 start 为 Null，将会发生错误；

compare 为选择项，为一数值表达式，设定字符串比较种类，如果省略，将会进行二进比较。

参数 compare 的设定值，如表附-10 所示。

表附-10 参数 compare 的设定值

名 称	参数值	含 义
VBBinaryCompare	0	进行二进制数据比较
VBTextCompare	1	进行文字数据比较
VBDatabaseCompare	2	执行数据内容比较

InstrRev 函数在不同的条件下有不同的返回值，具体描述如表附-11 所示。

表附-11 InstrRev 函数的返回值

条 件	返回值
String1 长度为 0	0
String1 为 Null	Null
String2 长度为 0	Start
String2 为 Null	Null
String2 找不到	0
在 String1 中找到 String2	找到的位置
Start > Len（string2）	0

需要说明的是，该函数与 InStr 函数的语法并不一样，使用时应该有所区别。

35. IsDate 函数

语法：

```
IsDate (exp)
```

参数 exp 可以是任何日期表达式，或其他可以被认为是日期、时间的字符串表达式。

该函数的返回值为布尔型，经常被用来判断 exp 是否是日期值。如果 exp 为日期值，则返回 True 否则将返回 False。

36. IsEmpty 函数

语法：

```
IsEmpty (exp)
```

参数 exp 可以是任何表达式。

该函数主要用来指出变量是否已经初始化，将会返回一个布尔值。如果变量 exp 未被初始化，或者已明确设定为 Empty，将会返回 True 否则将会返回 False。如果 exp 含有一个以上的变量时，将会返回 False。

37. IsNull 函数

语法：

```
IsNull (exp)
```

参数 exp 可以是任何表达式。

该函数将会返回 Boolean 值，用来指定表达式是否未含任何有效数据。

注解：

如果 exp 为 Null，则该函数将会返回 True。否则该函数将会返回 False。如果 exp 中存在不只一个变量，如果其中任一变量为 Null，则会返回 True。

该函数与 IsEmpty 函数有所不同，前者用来判断未初始化的变量，这和空字符串有所不同。

38．IsNumeric 函数

语法：

```
IsNumeric (exp)
```

参数 exp 可以是任何表达式。

该函数将会返回 Boolean 值，用来指定表达式的结果是否为数字。如果整个 exp 的运算结果为数字，将会返回 True；否则将会返回 False。

39．IsObject 函数

语法：

```
IsObject (exp)
```

参数 exp 可以是任何表达式。

该函数将会返回 Boolean 值，用来指定 exp 是否为一个对象。如果整个 exp 是对象类型，将会返回 True；否则将会返回 False。

40．Join 函数

语法：

```
Join (list",delimiter")
```

该函数将返回一字符串，结果是由数组中一些子字符串连接而成。

各参数的含义如下：

list 为必选项，包含欲连接的字符串的一维数组；

delimiter 为选择项，使用于返回字符串中分隔子字符串的字符，如省略，则将使用空白字符""，如果 delimiter 为空字符串，那么所有的连接将无分隔符。

41．Lbound 函数

语法：

```
Lbound (arraynm",dimension")
```

该函数用于返回指定数组某维最小可使用的数组索引。

各参数的含义如下：

arraynm 为数组变量的名称，遵循标准变量命名规格；

dimension 表示返回的是某一维的下限，1 表示第一维，2 表示第二维，以此类推，缺省值为 1。

注解：

利用本函数和 Ubound 函数可以决定数组的大小。使用 Ubound 函数可以找出数组的某

一维的上限。

42. Lcase 函数

语法：

```
Lcase (str)
```

该函数用于返回一转换为小写的 str。

参数 str 可以是任何字符串表达式。如果 str 为 Null，将返回 Null。

注解：

只有大写的字母才转换为小写，所有的小写字母及其他字符保持不变。

43. Left 函数

语法：

```
Left (str,len)
```

该函数用于返回一字符串从左数起某位置的字符。

各参数含义如下：

str 表示字符串表达式；

len 为数字表达式，表示返回从左数多少个字符，如果为零，返回“”，如果大于字符串的长度，将返回整个字符串。

44. Len 函数

语法：

```
Len (str | varnm)
```

该函数用于返回字符串中字符的数目，或存储某变量的位数。

各参数含义如下：

str 表示字符串表达式，如果字符串为 Null，将返回 Null；

varnm 表示变量名称，如果 varnm 为 Null，将返回 Null。

45. LoadPicture 函数

语法：

```
LoadPicture (picturenm)
```

该函数用于返回一图形对象。

各参数含义如下：

picturenm 是一个字符串表达式，表示图形文件名。图形格式由该函数管理，包括位图文件、图表文件等。

46. Log 函数

语法：

```
Log (num)
```

该函数返回参数 num 对应的自然对数值。

参数 num 可以是任何数值表达式，条件是参数值 num 必须大于 0。

47．Ltrim、Rtrim 与 Trim 函数

语法：

```
Ltrim (str)
Rtrim (str)
Trim (str)
```

这些函数分别将给定字符串的前面空白、后面空白或前后空白删除后返回。

参数 str 可以是任何字符串表达式。如果 str 为 Null，则返回 Null。

48．Mid 函数

语法：

```
Mid (str,start",len")
```

该函数用于返回特定数量的字符。

各参数含义如下：

str 是一个字符串表达式，如果 str 为 Null，则返回 Null；

start 代表要返回字符串在 str 的开头位置，如果 start 超过 str 的范围，将返回零长度字符串（""）；

len 表示返回的字符数，如果省略或 len 超过可以返回的字符数，就将返回从 start 到尾端的所有字符数。

49．Minute 函数

语法：

```
minute (time)
```

该函数返回一值，从 0 到 59，表示一小时中的某分钟。

参数 time 表示任何可以表示合理时间的表达式。

50．Month 函数

语法：

```
Month (date)
```

该函数用于返回一值，从 1 到 12，表示一年中的某月。

参数 date 可以是任何一个表示合理日期的表达式。

51．MonthName 函数

语法：

```
MonthName (month",abbreviate")
```

该函数用于返回指定月份的字符串。

各参数含义如下：

month 为必选项，代表月份的数值，例如一月是 1、二月是 2 等等；

abbreviate 为选择项，为一个 Boolean 值，用来指定月份名称是否为缩写，如果省略，则默认为 False，即表示月份名称不可缩写。

52. MsgBox 函数

语法：

```
MsgBox (prompt",buttons"",title"",helpfile,context")
```

该函数用来将信息显示在消息框中，并等待访问者确认。

各参数的含义如下：

prompt 是一个字符串表达式，用来作为对话框信息的字符串表达式，其最大长度大约为 1024 个字符，换行可以通过 chr（3）、chr（0）来实现；

buttons 为数值表达式，用来指定显示按钮的数目及形状、使用的图表样式、预设按钮等，如果没有指定，则 Buttons 的默认值为 0，其取值，如表附-12 所示。

表附-12 Buttons 参数的取值

名 称	参数值	含 义
VBOkonly	0	只显示 'OK' 按钮
VBOkCancel	1	显示 '确定' 及 '取消' 按钮
VBAbortRetryIgnore	2	显示 '放弃'、'重试'、'忽略' 按钮
VBYesNoCancel	3	显示 '是'、'否'、'取消' 按钮
VBYesNo	4	显示 '是'、'否' 按钮
VBRetryCancel	5	显示 '重试'、'取消' 按钮
VBCritical	16	显示 '重试信息' 对话框
VBQuestion	32	显示 '问号符号' 对话框
VBExclamation	48	显示 '警告符号' 对话框
VBInformation	64	显示 '信息符号' 对话框
VBDefaultButton1	0	第一个按钮为默认值
VBDefaultButton2	256	第二个按钮为默认值
VBDefaultButton3	512	第三个按钮为默认值
VBDefaultButton4	768	第四个按钮为默认值
VBApplicationModal	0	模拟对话框
VBSystemModal	4096	系统强制响应，所有的应用程序都会暂停，直到使用者响应此消息框

title 显示为对话框标题，缺省值为应用程序的名称；

helpfile 为字符串表达式，用来指定对话框的帮助文件，如果 helpfile 被指定，则 context 也必须被指定；

context 为数值表达式，由帮助文件来指定给某个说明主题的代码。与 helpfile 必须一同指定。并且对话框会指定一个帮助按钮；

msgBox 函数的返回值，如表附-13 所示。

表附-13 MsgBox 函数的返回值

名 称	参数值	按 钮	名 称	参数值	按 钮
VBOK	1	确 定	VBIgnore	5	忽 略
VBCancel	2	取 消	VBYes	6	是
VBAbort	3	放 弃	VBNo	7	否
VBRetry	4	重 试			

53. Now 函数

语法：

```
Now
```

该函数用于返回系统现在的日期及时间。

54. Oct 函数

语法：

```
Oct (num)
```

参数 num 可以为任何正确的表达式，并且 num 如果不是整数，将会先进行四舍五入后进行处理。

该函数将会返回一代表 num 的八进制的字符串。

55. Replace 函数

语法：

```
Replace (exp,find,repwidth",start",count",compart""")
```

该函数用于返回字符串，用来表示字符串中的一特定字符串被另一个字符串取代的次数。

各参数的含义如下：

exp 为必选项，表示一个字符串表达式，以及要被取代的子字符串；

find 为必选项，表示要搜寻的字符串；

repwidth 为必选项，表示用来取代的子字符串；

start 为必选项，用来表示搜寻的开始位置；

count 为选择项，用来表示取代次数；

compare 为选择项，表示子字符串比较种类。

参数设定值如表附-14 所示。

表附-14 Compare 参数的设定值

名 称	参数值	含 义
VBBinaryCompare	0	二进制比较
VBTextCompare	1	字符比较
VBDatabaseCompare	2	数据内容比较

该函数对参数的不同取值有不同的返回值，如表附-15 所示。

表附-15 Replace 函数的返回值

条 件	返 回 值
Exp 的长度为 0	空字符串
Exp 为 Null	错误
Find 的长度为 0	一份 Exp 的复制
Repwidth 的长度为 0	Find 的移除项目
Start > Len (exp)	空字符串
Count 为 0	一份 Exp 的复制

56. RGB 函数

语法：

```
RGB (red,green,blue)
```

参数 red、green、blue 分别表示红、绿、蓝的份值。

该函数用来设定颜色值。

57. Right 函数

语法：

```
Right (str,len)
```

该函数用于返回字符串 str 右边 len 个字符。

各参数的含义如下：

str 为字符串表达式；

len 为数值表达式，指出从右边返回多少字符。

58. Rnd 函数

语法：

```
Rnd" (num)"
```

该函数用于返回一随机数。

参数 num 可以是任何的数值表达式。

注解：

Rnd 函数返回的随机数介于 0 和 1 之间。可以等于 0，但不能等于 1。

59. Round 函数

语法：

```
Round (exp",numdec")
```

该函数用于返回一个被四舍五入的数值。

各参数的含义如下：

exp 为必选项，表示被四舍五入的数值表达式；

numdec 为选择项，用来表示要四舍五入到小数点后第几位，如果省略，将返回整数。

60．ScriptEngine 函数

语法：

```
ScriptEngine
```

该函数用于返回一字符串，表示所使用的 Script 程序语言。

具体函数的返回值，如表附-16 所示。

表附-16　ScriptEngine 函数的返回值

返 回 值	含　义
VBScript	表示 Script Edition 是目前的 Script 引擎
JScript	表示 Microsoft JScript 是目前的 Script 引擎
VBA	表示 Microsoft VB for Application 是目前的引擎

61．ScriptEngineMajorVersion 函数

语法：

```
ScriptEngineMajorVersion
```

该函数将会返回所使用的 Script 引擎的主要版本编号。

62．ScriptEngineMinorVersion 函数

语法：

```
ScriptEngineMinorVersion
```

该函数将会返回所使用的 Script 引擎的次要版本编号。

63．Second 函数

语法：

```
Second (time)
```

该函数返回一值，从 0 到 59，表示某时间内的秒值。

64．Sgn 函数

语法：

```
Sgn (num)
```

该函数返回一个整数代表参数的正负号。

参数 num 可以是任何的数值表达式。

Sgn 将会根据不同条件返回不同的值，如表附-17 所示。

表附-17　Sgn 函数的返回值

条　件	返 回 值
Num > 0	1
Num = 0	0
Num < 0	- 1

65. Sin 函数

语法：

```
Sin (num)
```

该函数返回参数 num 的正弦值。

参数 num 是任何的数值表达式，代表一个角度，以弧度为单位。

66. Space 函数

语法：

```
Space (num)
```

该函数返回 num 个空格的字符串。

参数 num 表示想要的空格数。

67. Split 函数

语法：

```
Split (exp",delimiter",count",compare"")
```

该函数返回一个以 0 为基数的一维数组，包含特定个数的子字符串。

各参数的含义如下：

exp 为必选项，为一字符串表达式，它应该包含子字符串及分隔符号；

delimiter 为选择项，用来表示字符串界限的字符串字符，缺省值为空白字符（" "）；

count 为选择项，返回的子字符串的个数，- 1 表示所有的子字符串将被返回，为缺省值；

compare 为选择项，指定字符串比较的种类。

其取值，如表附- 18 所示。

表附- 18 Compare 选项的取值

名 称	参 数 值	含 义
VBBinaryCompare	0	二进制比较
VBTextCompare	1	字符比较
VBDatabaseCompare	2	数据内容比较

68. Sqr 函数

语法：

```
Sqr (num)
```

该函数将会返回参数的平方根。

参数 num 可以是任何数值表达式，只要大于或等于 0 即可。

69. StrComp 函数

语法：

```
StrComp (str1,str2",compare")
```

该函数返回一值，代表字符串比较的结果。

各参数的含义如下：

str1 为必选项，为任何正确的字符串表达式；

str2 为必选项，为任何正确的字符串表达式；

compare 为选择项，指定比较类型，如表附-18 所示。

该函数可以根据不同条件返回不同的值，如表附-19 所示。

表附-19　StrComp 函数的返回值

条　件	返 回 值
str1 < str2	-1
str1 = str2	0
str1 > str2	1
str1 或 str2 为 null	Null

70．StrReverse 函数

语法：

```
StrReverse (str)
```

该函数将特定字符串的字符顺序颠倒后返回。

Str 可以是任何要颠倒的字符串。

71．String 函数

语法：

```
String (num,character)
```

该函数用于返回一特定长度的重复字符的字符串。

各参数的含义如下：

num 为数值表达式，表示要返回字符串的长度；

character 为字符码或字符串表达式（用其第一个字符），用来构建返回字符串的特定字符。

72．Tan 函数

语法：

```
Tan (num)
```

该函数用于返回参数的正切值。

参数 num 可以是任何的数值表达式。代表一个角度，以弧度为单位。

73．Time 函数

语法：

```
Time
```

该函数将返回系统时间。

74．TimeSerial 函数

语法：

```
TimeSerial (hour,minute,second)
```

该函数返回内容为指定的时、分、秒的日期表达式。

各参数的含义如下：

hour 从 0 到 23 的数值表达式；

minute 从 0 到 59 的数值表达式；

second 从 0 到 59 的数值表达式。

75. TimeValue 函数

语法：

```
TimeValue (time)
```

该函数返回一个 Variant (Date)的时间表达式。

参数 time 是一个字符串表达式，用来表示一个时刻。从 0:00:00 到 23:59:59。

76. TypeName 函数

语法：

```
TypeName (varnm)
```

该函数返回一个 String，提供了某个变量的相关信息。

参数项 varnm 可以是任何变量，其取值如表附-20 所示。

表附-20 TypeName 函数的参数取值

名 称	含 义	名 称	含 义
Byte	位元值	Boolean	布尔型
Integer	整数	Empty	未初始化
Long	长整数	Null	未含有效数据
Single	单精度整数	< Object Type >	对象实际类型名称
Double	双精度整数	Object	一般组件
Currency	货币	Unknown	未知类型组件
Decimal	十进制	Nothing	不再应用任何对象
Date	日期或时间	Error	错误
String	字符串		

77. Ubound 函数

语法：

```
Ubound (arraynm",dimension")
```

该函数将返回指定数组中某维可使用的最大索引。

各参数的含义如下：

arraynm 为必选项，数组变量名称。

dimension 为选择项，表示返回那一维的上限 缺省值为 1。

78. Ucase 函数

语法：

```
Ucase (str)
```

该函数将要返回以转换为大写的字符串。

参数 str **可以是任何字符串表达式。**

79．VarType **函数**

语法：

```
VarType (varnm)
```

该函数返回变量的类型。

参数 varnm **可以是任何变量。**

80．Weekday **函数**

语法：

```
Weekday (date",firstdayofweek")
```

该函数返回一整数，代表某日为星期几。

各参数的含义如下：

date **表示任何一合理日期的表达式；**

firstdayofweek **为常数，表示一星期的第一天，缺省值为** VBSunday**，表示星期天为第一天。**

81．WeekDayName **函数**

语法：

```
Weekdayname (week,abbreviate,firstdayofweek)
```

该函数用于返回一字符串，表示一星期中的某一天。

各参数的含义如下：

weekday **为必选项，表示一星期中的某一天的数值，依据** firstdayofweek **来设定；**

abbreviate **为必选项，为布尔型，表示是否可为缩写；**

firstdayofweek **为选择项，与上一个函数的所述相同。**

82．Year **函数**

语法：

```
Year (date)
```

该函数返回一数值，表示那个年份。

参数 date **可以是任何表达式，只要是一个合理的日期即可。**

主要参考书目

1 李世杰. Active Server Pages (ASP)2.0 网页设计手册.北京：清华大学出版社，1999
2 肖金秀，冯沃辉，施鸿翔，香文斌等.ASP 动态网页培训教程.北京：冶金工业出版社，1999
3 汪晓平，吴勇强，张宏林等 .ASP 网络开发技术 . 北京：人民邮电出版社，2000
4 徐罕，吴玉新 . 网站 ASP 后台解决方案 . 北京：人民邮电出版社，2001
5 武延军，赵彬 . 精通 ASP 网络编程 . 北京：人民邮电出版社，2000
6 蒋光帅 .Frontpage2000&ASP 网页设计技巧与网站维护 . 北京：清华大学出版社，2000
7 BRYAN 著，杨亚平等译 .HTML4 实用大全 . 北京：中国水利水电出版社，1999
8 廖信彦 .ACTIVE SERVER PAGES 应用大全 .ASP 与数据库的整合 . 北京：清华大学出版社，2000
9 本社编 . 动态 WEB 应用高级开发指南——ASP ADO 和 DHTML 编程 . 北京：人民邮电出版社，1999
10 Richard Anderson，Chris Blexrud 等著，刘福太等译 .ASP3 高级教程 . 北京：机械工业出版社，2000
11 李劲 . 精通 ASP 数据库程序设计 . 北京：科学出版社，2001
12 邓文渊，陈惠贞，陈俊荣 .ASP 与网页数据库设计 . 北京：中国铁道出版社，2001
13 林风 . 动态网站设计捷径——ASP. 西安：西安电子科技大学出版社，1999
14 王国荣 .ASP 网页制作教程 . 北京：人民邮电出版社，2000
15 Alex Homer 著，刘爱民、高德辉等译 .ASP3.0 专业 Web 技术 . 北京：人民邮电出版社，2000