

高等学校计算机科学与技术教材

ASP 精解案例教程

石志国 王志良 薛为民 编著

清华大学出版社
北方交通大学出版社
· 北京 ·

内 容 提 要

本书对B/S架构ASP编程进行了系统的介绍,最大特色是:程序和案例都来自教学实践,全书有20个完整的案例和超过150个基本程序,都是网站应用中常用的程序。本书介绍了B/S架构的编程体系:客户端采用HTML, CSS和JavaScript脚本语言,服务器端采用ASP+SQL Server体系。全书分成四大部分:第一部分,环境配置与Web编程基础,介绍ASP+SQL Server平台的配置、HTML, CSS和JavaScript语言的使用;第二部分,ASP对象与组件,介绍ASP五大常用内置对象、ASP的内置组件和常用的外部组件;第三部分,ASP操作数据库,介绍了ASP操作Access和SQL Server数据库的三大基本格式及如何读写XML文件;第四部分,工程实践,从工程的角度介绍在线考试系统的设计与开发。

版权所有,翻印必究。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

(本书防伪标签采用清华大学核研院专有核径迹膜防伪技术,用户可通过在图案表面涂抹清水,图案消失,水干后图案复现;或将表面膜揭下,放在白纸上用彩笔涂抹,图案在白纸上再现的方法识别真伪)。

图书在版编目(CIP)数据

ASP精解案例教程/石志国,王志良,薛为民编著. —北京:清华大学出版社;北方交通大学出版社,2004.1

(高等学校计算机科学与技术教材)

ISBN 7-81082-221-7

. A... . 石... . 王... . 薛... . 主页制作-程序设计-高等学校-教材
. TP393.092

中国版本图书馆CIP数据核字(2003)第093287号

责任编辑:谭文芳

印刷者:北京东光印刷厂

出版发行:清华大学出版社 邮编:100084 电话:010-62786969

北方交通大学出版社 邮编:100044 电话:010-51686414

经 销:各地新华书店

开 本:787×1092 1/16 印张:17.5 字数:448千字 附光盘1张

版 次:2004年1月第1版 2005年1月第3次印刷

印 数:11001~16000册 定价:29.00元(含光盘)

目 录

第一部分 环境配置与 Web 编程基础

第1章	配置 ASP 运行环境	(1)
1.1	软件编程体系简介	(1)
1.1.1	B/S 架构编程体系	(1)
1.1.2	C/S 架构编程体系	(2)
1.2	ASP 基本原理	(2)
1.2.1	ASP 的发展历史	(2)
1.2.2	ASP 相关技术概览	(3)
1.3	ASP 运行环境	(3)
1.3.1	安装 IIS 5.0	(3)
1.3.2	配置 IIS 5.0	(5)
1.3.3	测试 ASP 运行环境	(9)
1.3.4	SQL Server 的安装	(10)
1.3.5	测试 ASP + SQL Server 的开发平台	(14)
	小结	(14)
	课后习题和上机练习	(15)
第2章	Web 编程基础	(16)
2.1	HTML 概述	(16)
2.1.1	HTML 发展概述	(16)
2.1.2	建立 HTML 网页框架	(16)
2.2	HEAD 头元素	(17)
2.3	HTML 的常用标记	(18)
2.3.1	字体标记	(18)
2.3.2	图片格式	(19)
2.3.3	超级链接	(19)
2.3.4	列表	(20)
2.3.5	表格	(22)
2.3.6	表单	(26)
2.3.7	块级元素	(29)
2.3.8	预排版标记	(30)
2.3.9	设计网页框架	(31)
2.4	CSS 概述	(34)
2.5	加载 CSS 样式的三种方式	(35)

2.5.1	HEAD 内引用	(35)
2.5.2	BODY 内引用	(36)
2.5.3	文件外引用	(37)
2.6	CSS 与标记对应的三种方式	(39)
2.6.1	标记选择符	(39)
2.6.2	类选择符	(39)
2.6.3	ID 选择符	(40)
2.7	定义超级链接样式	(41)
	小结.....	(41)
	课后习题和上机练习.....	(42)
第3章	JavaScript 语言简介	(43)
3.1	JavaScript 简介	(43)
3.1.1	JavaScript 概述	(43)
3.1.2	JavaScript 与Java 的区别	(43)
3.1.3	JavaScript 的运行环境	(43)
3.2	网页中引入JavaScript	(44)
3.3	变量与数组	(44)
3.3.1	变量	(44)
3.3.2	数组	(46)
3.4	表达式与运算符	(47)
3.4.1	算术运算符	(47)
3.4.2	逻辑运算符	(48)
3.4.3	字符串运算符	(49)
3.4.4	条件表达式	(50)
3.5	流控制语句之条件语句	(51)
3.5.1	if 语句.....	(51)
3.5.2	switch 语句	(52)
3.6	流控制语句之循环语句	(53)
3.6.1	for 语句	(54)
3.6.2	while 语句	(55)
3.6.3	break 与continue 语句	(55)
3.7	JavaScript 函数	(57)
3.7.1	函数定义.....	(57)
3.7.2	函数调用.....	(58)
3.8	事件的概念	(59)
3.8.1	单击事件.....	(60)
3.8.2	处理下拉列表	(60)
3.9	对象处理语句	(63)
3.9.1	this 语句	(63)

3.9.2	for...in 语句	(64)
3.9.3	with 语句	(65)
3.10	JavaScript 内置对象	(66)
3.10.1	时间对象	(66)
3.10.2	Math 对象	(69)
3.10.3	String 对象	(70)
3.11	JavaScript 的常用函数	(72)
3.11.1	eval() 函数	(72)
3.11.2	parseInt() 函数和parseFloat() 函数	(73)
3.12	对象层次及DOM 模型	(74)
3.12.1	window 对象	(76)
3.12.2	history 对象	(78)
	小结	(89)
	课后习题和上机练习	(89)

第二部分 ASP 内置对象与组件

第4 章	ASP 内置对象	(91)
4.1	内置对象概述	(91)
4.2	Response 对象	(91)
4.2.1	输出数据	(91)
4.2.2	网页转向	(94)
4.2.3	停止输出	(95)
4.3	Request 对象	(95)
4.3.1	获得表单数据	(96)
4.3.2	获得服务器信息	(102)
4.4	Application 对象	(102)
4.4.1	自定义属性	(103)
4.4.2	实现聊天室	(104)
4.5	Session 对象	(107)
4.5.1	对Session 的理解	(107)
4.5.2	自定义属性	(108)
4.6	Server 对象	(110)
4.6.1	输出HTML 代码	(110)
4.6.2	获取物理路径	(110)
4.7	Cookie 集合	(111)
4.7.1	写入Cookie	(111)
4.7.2	读取Cookie	(112)
4.8	global.asa 文件	(112)
	小结	(121)

课后习题和上机练习	(122)
第5章 ASP 内置组件	(123)
5.1 使用内置文件组件	(123)
5.1.1 使用对文件操作的组件	(123)
5.1.2 对文件进行处理	(127)
5.1.3 对文件夹和驱动器进行操作	(132)
5.2 广告的处理	(141)
小结.....	(143)
课后习题和上机练习	(143)
第6章 在ASP中使用外置组件	(144)
6.1 利用ASP的外部组件.....	(144)
6.1.1 组件概述	(144)
6.1.2 组件的调用方法	(144)
6.2 实现文件上传	(144)
6.2.1 上传组件简介	(145)
6.2.2 组件提供的方法	(146)
6.2.3 组件提供的属性	(147)
6.3 E-mail 组件	(149)
6.4 自己编写组件	(152)
小结.....	(157)
课后习题和上机练习	(157)

第三部分 ASP 操作数据库

第7章 ADO 数据访问接口	(158)
7.1 ADO 概述	(158)
7.2 Connection 数据对象	(158)
7.2.1 打开和关闭数据库连接	(160)
7.2.2 向浏览器输出数据库内容	(162)
7.2.3 以表格的形式输出	(163)
7.3 使用SQL 语句.....	(166)
7.3.1 Select 的三大基本格式	(166)
7.3.2 Like 子句	(168)
7.3.3 使用SQL 语句操作数据库	(171)
7.4 RecordSet 数据对象	(176)
7.4.1 RecordSet 对象的属性及方法	(176)
7.4.2 使用 RecordSet 对象打开数据库	(177)
7.4.3 实现数据库的分页显示	(180)

7.5	Command 数据对象	(188)
	小结	(198)
	课后习题和上机练习	(198)
第8章	ASP 操作SQL Server 数据库	(199)
8.1	SQL Server 简介	(199)
8.2	SQL Server 的集成环境介绍	(199)
8.2.1	SQL 服务管理器	(199)
8.2.2	企业管理器	(200)
8.2.3	查询分析器	(201)
8.2.4	事件探查器	(202)
8.2.5	联机帮助	(202)
8.3	创建数据库	(203)
8.3.1	创建数据库	(203)
8.3.2	删除数据库	(205)
8.3.3	数据类型	(205)
8.3.4	表	(206)
8.3.5	修改表	(207)
8.3.6	删除表	(208)
8.4	数据完整性	(208)
8.4.1	使用Identity 属性	(208)
8.4.2	使用 Uniqueidentifier 类型	(210)
8.4.3	使用约束	(210)
8.5	ADO 操作SQL Server 数据库	(216)
8.5.1	格式一的SQL Server 版本	(218)
8.5.2	格式二的SQL Server 版本	(221)
8.5.3	格式三的SQL Server 版本	(224)
8.6	SQL Server 存储过程	(225)
8.6.1	存储过程的概念	(225)
8.6.2	存储过程的例子	(225)
8.7	ADO 操作SQL Server 存储过程	(231)
8.7.1	调用无输入输出参数存储过程	(231)
8.7.2	调用带输入输出参数的存储过程	(233)
	小结	(240)
	课后习题和上机练习	(241)
第9章	ASP 操作XML 文件	(242)
9.1	XML 的概念	(242)
9.2	编写XML 文档	(243)
9.2.1	定义基本元素	(243)

9.2.2	使用属性	(243)
9.3	XML 文档结构	(244)
9.3.1	XML 声明	(245)
9.3.2	注释	(246)
9.3.3	字符和实体引用	(246)
9.4	XML 的三种显示格式	(247)
9.4.1	CSS 样式表	(247)
9.4.2	XSL 样式语言	(248)
9.4.3	XML 的数据岛技术	(250)
9.5	使用XML 组件	(252)
9.5.1	创建 DOM 对象	(252)
9.5.2	读取XML 文件	(253)
	小结.....	(259)
	课后习题和上机练习.....	(259)

第四部分 工 程 实 践

第10 章	项目分析：在线考试系统	(260)
10.1	在线考试系统的数据结构.....	(260)
10.2	考试系统的实现.....	(261)
	小结.....	(267)
	课后习题和上机练习.....	(267)
	参考文献.....	(269)

前言

ASP 技术是目前网站应用中的核心技术,也是流行的 3P 技术中应用最广泛的一种。3P 技术分别是:ASP(Active Server Pages),PHP(Personal HomePage)和 JSP(Java Server Pages)。ASP 是微软公司的产品,JSP 最初是 SUN 公司推出的,PHP 是由一个网络小组开发和维护的。目前最常用的是 ASP 和 JSP。□

在企业应用中,操作系统使用 Windows 2000 Server,Web 服务器使用操作系统自带的 IIS 5.0 (Internet Information Server),服务器端语言使用 ASP,数据库服务器使用 SQL Server。这种搭配已经成为目前开发领域中的标准配置。□

ASP 可以使用 JScript (JavaScript 的微软版)或者 VBScript 作为脚本语言,本书全部程序都使用 JScript。使用该脚本有以下四个优点。□

1. JScript 和 C 语言的语法很类似,几乎所有的高校都开 C 语言这门课程。这样有利于利用已有的知识,或者为学习其他课程提供良好的基础。□

2. 客户端编程考虑浏览器兼容的问题,目前一般采用 JavaScript 语言。这样客户端和服务端使用语言的语法就一致了,不至于将语法弄混,有利于学习。□

3. 在 ASP.NET 中,一般采用 C#作为脚本,C#的语法和 JScript 一致,这样就为学习新技术提供了必要的基础。□

4. 在实际应用中,大部分企业采用 JScript 作为脚本,如联想、方正和用友软件等知名企业。这样使学习更加贴近于实际的工程应用。□

本书全面介绍客户端和服务端编程技术,全书从体系上分成以下四大部分。□

第一部分:环境配置与 Web 编程基础。□

第 1 章配置 ASP 运行环境:介绍如何配置 ASP+SQL Server 的运行平台,并给出测试程序。

第 2 章 Web 编程基础:介绍标记语言 HTML 和样式语言 CSS 以及如何将这两种语言搭配使用。□

第 3 章 JavaScript 语言简介:介绍 JavaScript 语言的语法、控制语言以及如何与 HTML 搭配使用等。□

第二部分:ASP 内置对象与组件。□

第 4 章 ASP 内置对象:介绍 ASP 的常用五大对象、一个集合和一个文件的使用方法。□

第 5 章 ASP 内置组件:介绍 ASP 的两大常用内置组件,文件系统组件和广告组件。□

第 6 章在 ASP 中使用外置组件:介绍两个外置组件,文件上传组件和 E-mail 组件的使用方法以及如何自己写一个组件。□

第三部分:ASP 操作数据库。□

第 7 章 ADO 数据访问接口:介绍 ADO 的三个常用对象和操作数据库的三个基本格式及如何操作 Access 数据库。□

第 8 章 ASP 操作 SQL Server 数据库:介绍 SQL Server 数据库的基本使用及如何使用 ASP 操作 SQL Server 数据库。□

第 9 章 ASP 操作 XML 文件:介绍 XML 文件的基本概念及如何使用 ASP 操作 XML 文件。

第四部分：工程实践。

第 10 章项目分析：在线考试系统，介绍一个在线考试系统的规划和设计。□

本书所有案例和程序都来自教学实践，作为讲义用了三年有余。所有程序力求最精，每一个程序说明一个知识点，尽量不涉及其他的知识点，每一个案例是一个小综合，将几个程序的知识点综合起来实现一个应用。由于时间和作者水平有限，难免出现错误，对于本书的任何问题请使用 E-mail 发送到作者邮箱：shizhiguo@tom.com，本书的支持信息将在<http://www.gettop.net> 上发布。□

在本书的编写过程中，得到众多老师的指导和帮助。感谢北京大学计算机研究所的陈晓鸣教授、吴於茜老师、王学武老师、章丰老师、曾建平老师、徐申鑫老师、张大力老师、姜每英老师和涂哲民老师，他们为本书提供了良好的技术支持和指导。感谢北京科技大学的刘冀伟教授和贾文静老师为本书提供了技术指导。感谢新东方 IT 教育栗松涛老师和刘宏伟老师，感谢他们在大纲制定的过程中提供的帮助。感谢本书的编辑北方交通大学出版社谭文芳老师，没有她的辛勤劳动，本书不可能出版。感谢我的父母，在写作的过程中给了我无微不至的关心。感谢我的学生们，他们的每一个问题，都是本书要强调的知识点，他们的笑容是我最大的动力，本书献给他们和最广大的读者。□□

石志国□

2004 年 1 月于北京

第一部分 环境配置与 Web 编程基础

第 1 章 配置 ASP 运行环境

本章要点

本章首先介绍 ASP 在整个应用程序开发体系中的位置，介绍两大编程架构。然后介绍 ASP 的发展历史及其 ASP 的相关技术。最后介绍如何配置 ASP 的运行环境、安装 SQL Server2000 的注意事项。利用两个案例分别来测试 ASP 的运行环境和 ASP+SQL Server 的开发平台。

1.1 软件编程体系简介

目前在程序开发领域中，主要分成两大编程体系，一种是基于浏览器的 B/S（Browser/Server）结构，另一种是 C/S（Client/Server）结构。如图 1-1 所示。

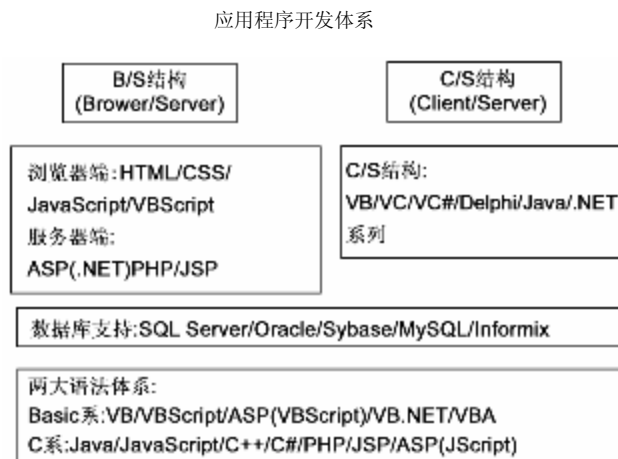


图 1-1 程序开发体系

开发基于 B/S 结构的项目，目前主要采用两种服务器端语言：ASP（Active Server Pages）和 JSP（Java Server Pages），这两种语言构成两大开发体系：ASP+SQL Server2000 体系和 ASP+Oracle 9i 体系。后面的章节将全面展示 ASP+SQL Server2000 的强大功能。

1.1.1 B/S 架构编程体系

从图 1-1 中可以看出，B/S 结构的编程语言分成浏览器端编程语言和服务器端编程语言。

浏览器端包括：HTML（Hypertext Markup Language，超文本标记语言）、CSS（Cascading Style Sheets，层叠样式表单）、JavaScript 语言和 VBScript 语言。所谓的浏览器端编程语言就是这些语言都是被浏览器解释执行的。HTML 和 CSS 都是由浏览器解释的，JavaScript 语言和 VBScript 语言是在浏览器上执行的。后面的章节将展示 HTML+CSS+JavaScript 构成的浏览器端的编程体系。

为了实现一些复杂的操作，比如：连接数据库，操作文件等，需要使用服务器端编程语言。目前主要是 3P（ASP、JSP 和 PHP（Personal Homepage））技术。ASP 是微软公司推出的，在这三种语言中是用得最为广泛的一种。JSP 是 SUN 公司推出的是 J2EE（Java 2 Enterprise Edition，Java2 企业版）十三种核心技术中最重要的一种。这两种语言是目前应用开发体系的主流。PHP 在 1999 年的下半年和 2000 年用得非常广泛，因为 Linux+PHP+MySQL（一种小型数据库管理系统）构成全免费的而且非常稳定的应用平台。但更新的速度比较慢，而且没有良好的技术支持，所以 PHP 逐渐淡出开发领域。

做应用开发，数据库支持是必须的，目前应用领域的数据库系统全部采用关系型数据库（Relation Database Management System, RDBMS）。在企业级的开发领域中，目前主要采用三大厂商的数据库关系系统：微软公司的 SQL Server2000、Oracle 公司的 Oracle 9i 和 IBM 公司的 DB2 7.2。

在浏览器端使用 JavaScript 编写程序，在服务器端采用 ASP，数据库采用 SQL Server2000，这样 B/S 体系就完整了。这样对做一个项目来说，知识体系也就完整了。后面的章节将致力介绍这样一个项目开发体系。

1.1.2 C/S 架构编程体系

在 2000 年以前，C/S 结构占据开发领域的主流，随着 B/S 结构的发展，C/S 结构已经逐步被 B/S 结构取代。值得一提的是两门经典的开发语言：C++ 和 Java，这两门语言覆盖了该领域 85% 以上的项目。虽然 Java 如日中天，但是 C++ 在开发领域中老大的位置，始终不变。

那么多语言，学习起来也是有规律可寻的。图 1-1 最下面的方框将目前常用的开发语言分成两大语系：Basic 语系和 C 语系，目前占主流的是 C 语系。语系中的语言所有的流程控制语言都是一样的，常用的函数也是大同小异的。所以只要精通其中任何一门语言，该语系中的其他语言也就不攻自破了。

1.2 ASP 基本原理

1.2.1 ASP 的发展历史

ASP 的第一个版本是 0.9 测试版，它能够将代码直接嵌入 HTML，使得设计 Web 页面变得简单更强大，并且通过内置的组件能够实现强大功能，最明显的就是 ActiveX Data Objects（ADO，数据访问接口）。

ASP 1.0 作为 IIS（Internet Information Server，Internet 信息服务器）的附属产品免费发送，

并且不久就在 Windows 平台上广泛使用。ASP 与 ADO 的结合使开发者很容易地在一个数据库中建立和打开一个记录集。这是它如此快就被大众接受的原因。

1998 年,微软公司又发布了 ASP 2.0。ASP 1.0 和 ASP 2.0 主要区别是外部组件。有了 ASP 2.0 和 IIS 4.0,就可以建立 ASP 应用了。

微软公司接着开发了 Windows 2000 操作系统。这个 Windows 版本给带上了 IIS 5.0 及 ASP 3.0。虽然到目前 Windows 已经发展到比较高的版本,但是开发领域中依然百分之百采用 Windows 2000 Server。Windows 2000 包括三个不同的版本:Professional, Server 和 Advanced Server。按照默认设置安装 Windows 2000 Server,安装时不用作任何改动,就配置好了 ASP 的运行环境。

1.2.2 ASP 相关技术概览

ASP 可以使用两种脚本语言:VBScript 和 Jscript。所谓的 Jscript 语言就是微软版本的 JavaScript 语言。本书所有案例程序基于 C 语系下的 Jscript 语言。

ASP 包含内置对象,最常用的是五大对象、一个集合和一个文件。五大对象分别是:Response, Request, Session, Application 和 Server,一个集合是 Cookies,一个文件是 Global.asa。

ASP 最常用的内置组件是操作文件的组件和操作广告条的组件。ASP 最强大的功能还是使用外置组件,比如使用外置组件实现文件上传,发送 E-mail,等等。

通过 ADO 数据访问接口可以方便地操作各种数据库。通过 ADO 访问数据库有三种标准的访问格式。

1.3 ASP 运行环境

建议的配置环境为:Windows 2000 Server SP2 + IE5.5/IE6.0+SQL Server 2000 企业版,这也是本书完成的环境。

1.3.1 安装 IIS 5.0

如果操作系统是 Windows 2000 Server 或者是 Windows 2000 Advanced Server 的话,IIS 5.0 已经是默认安装上的。如果是 Windows 2000 Professional,则需要安装 IIS 5.0。从操作系统“控制面板”中找到并双击“添加/删除程序”,单击“添加/删除 Windows 组件”,选择“Internet 信息服务”,然后选择详细信息,如图 1-2 所示。

选中“World Wide Web 服务器”,这就是需要安装的 Web 服务器,如图 1-3 所示。

单击两次“确定”以后,放入 Windows 2000 的安装盘,出现安装界面,进行安装。完成界面如图 1-4 所示。



图 1-2 安装 IIS 5.0



图 1-3 选择 World Wide Web 服务器



图 1-4 完成 IIS 5.0 的安装

安装完毕后，可以测试一下是否安装成功。打开浏览器，在浏览器的地址栏中输入 `http://localhost` 或者 `http://127.0.0.1`，如果安装成功的话，将会出现欢迎界面，如图 1-5 所示。

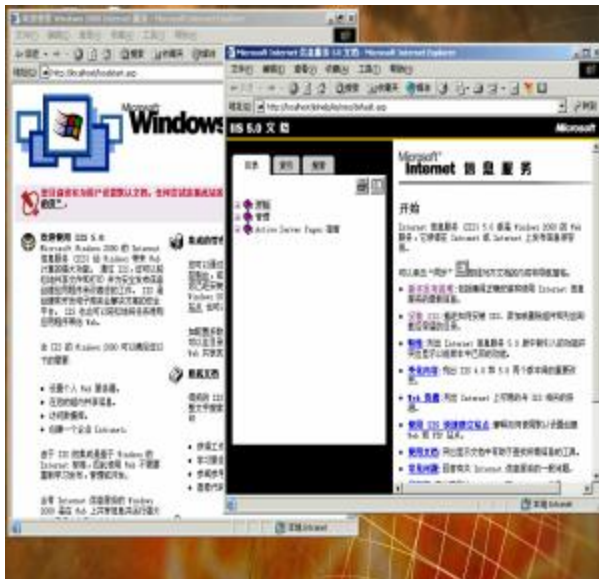


图 1-5 测试 IIS 5.0 的安装

进入 IIS 的管理界面可以在“控制面板”的“管理工具”中打开“Internet 服务管理器”。IS 的管理界面如图 1-6 所示。

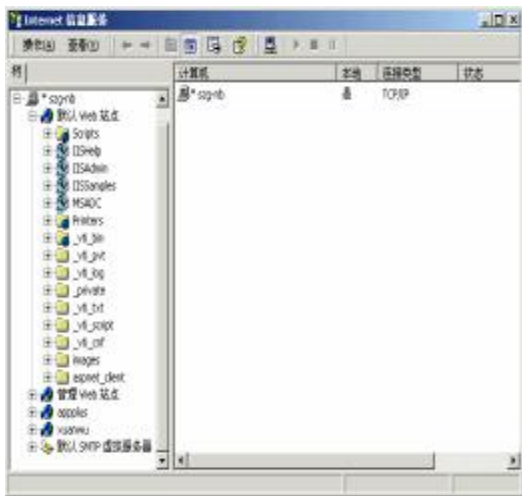


图 1-6 IIS 5.0 的管理界面

现在 IIS 5.0 的环境就建立好了，这个时候其实也已经建立了 ASP 的运行环境。

1.3.2 配置 IIS 5.0

如图 1-5 所示，在浏览器中输入 `http://localhost` 后，自动打开一个页面文件，这个文件其实是某一个目录下的一个文件。`http://localhost` 取的是网站的根目录，这个目录对应本地的一个目录，这个目录是可以改变的。首先打开 IIS 5.0 管理界面，右击“默认 Web 站点”，如图 1-7 所示。

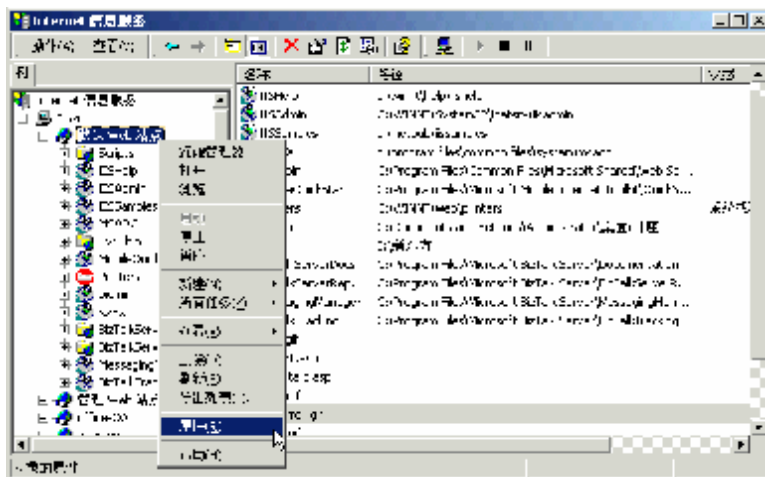


图 1-7 配置网站路径

单击“属性”菜单项，在出现的对话框中选择“主目录”选项卡，如图 1-8 所示。

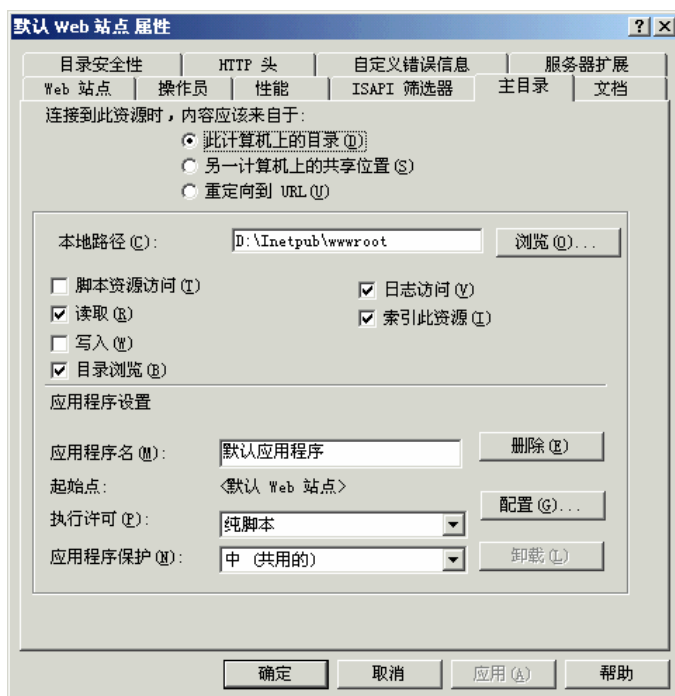


图 1-8 配置主目录

可以看到当前网站的本地路径是“D:\Inetpub\wwwroot”。一般，操作系统安装哪个盘，该路径就在哪个盘上。可以单击“浏览”按钮将网站指到本地的任何路径。下面有几个选项，一般调试程序时选择“目录浏览”选项。该选项的意义是：如果 IIS 5.0 找不到默认打开的文件，就将该目录下所有文件列出来。

单击“文档”卡，设置默认打开的文件如图 1-9 所示。

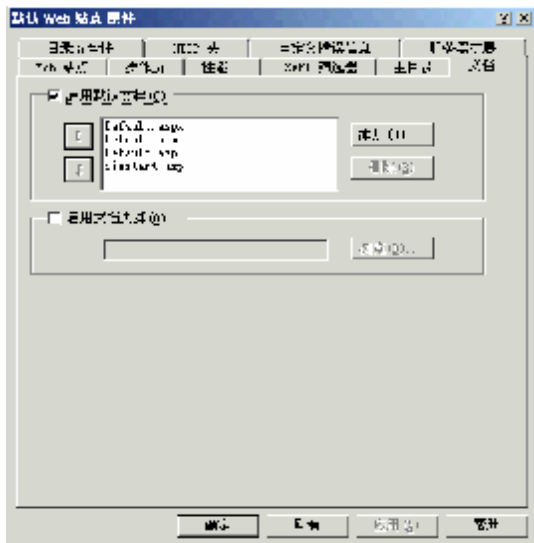


图 1-9 配置默认打开的文档

可以看到，当浏览网站时，IIS 5.0 自动在网站的主目录下寻找这些文件，从前到后依次寻找，如果找到了，就显示该文件。如果找不到这几个文件，判断是否可以目录浏览，如果可以目录浏览，则将该目录下所有文件列出来。

首先在 C 盘根目录下建立一个名为“asproot”的文件夹，将网站的主目录设置到该目录下，并将“目录浏览”打开，如图 1-10 所示。



图 1-10 设置主目录

在 C:\asproot 目录下新建一个文件“test.txt”，依然可以在浏览器地址栏中输入 <http://localhost> 看到该文件。这时可以在 IIS 5.0 中看到该文件，首先刷新 IIS 5.0，如图 1-11 所示。

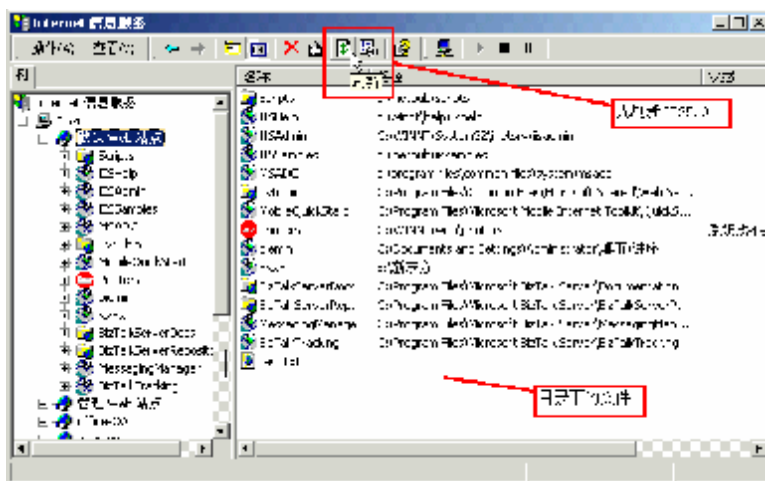


图 1-11 查看文件

右击“默认 Web 站点”选择“浏览”选项卡，如图 1-12 所示。

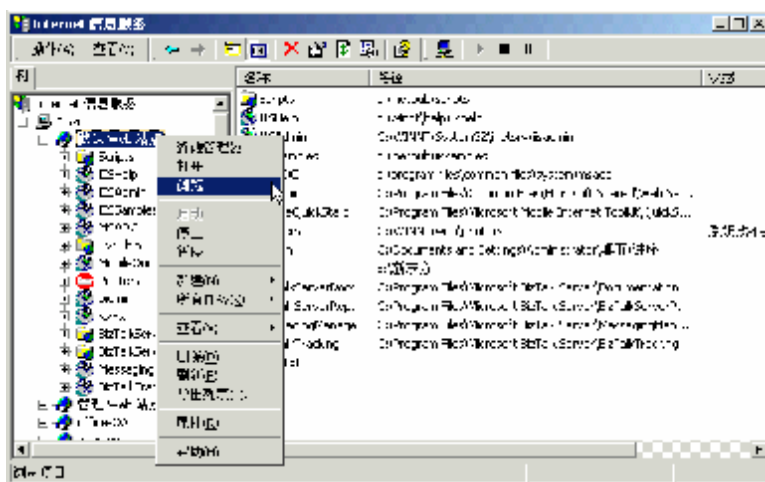


图 1-12 浏览 Web 站点

IIS 5.0 自动打开浏览器，如图 1-13 所示。

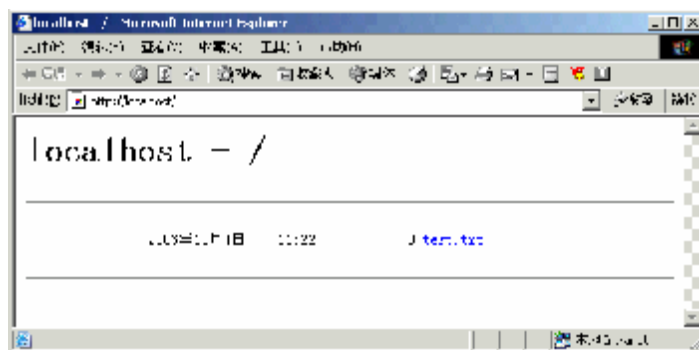


图 1-13 目录浏览

从图中可以看到，浏览器地址栏的地址也是 `http://localhost`，将 ASP 文件放在该目录下就可以执行 ASP 文件了。

1.3.3 测试 ASP 运行环境

第一个程序依然是经典的“Hello World”的例子，让程序输出字符串“你好，中国！我的祖国！”，如程序 1-01.asp 所示。

案例名称：测试 ASP 运行环境

程序名称：1-01.asp

```
<%@language="Jscript" %>
<%
Response.Write("你好，中国！我的祖国！");
%>
```

利用 Windows 自带的记事本编辑上面的文件，这里需要注意的是：Jscript 区分大小写，在输入的时候，一定要注意大小写。把文件命名为 1-01.asp，并保存到 `C:\asproot` 目录下。在浏览器中输入 `http://localhost`，可以看到程序列表，如图 1-14 所示。

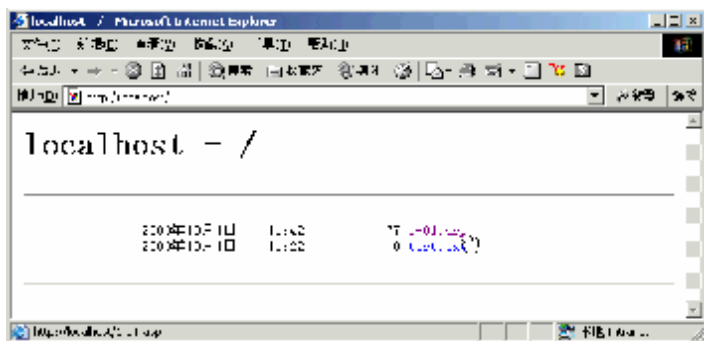


图 1-14 程序列表

单击该 ASP 文件，如果程序没有输入错误的话，就可以看到输出的字符串，如图 1-15 所示。

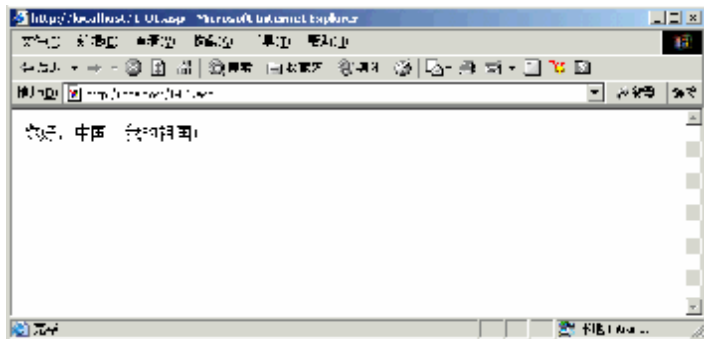


图 1-15 测试 ASP 运行环境

本书后面所有 ASP 文件的运行方式都按照这种方式。只要看到了该字符串，就说明运行环境没有问题。

任何应用程序都需要数据库的支持，下面介绍如何安装 SQL Server 2000 数据库系统。

1.3.4 SQL Server 的安装

总的来说，安装 SQL Server 2000 比较简单。但是其中有一些小的细节需要明确，不然可能会影响使用。放入 SQL Server 2000 的安装盘，第一步：选择安装“SQL Server 2000 组件”，如图 1-16 所示。



图 1-16 SQL Server 安装步骤一

第二步：选择“安装数据库服务器”，如图 1-17 所示。



图 1-17 SQL Server 安装步骤二

第三步：出现“软件许可协议”对话框，单击“是”按钮，如图 1-18 所示。

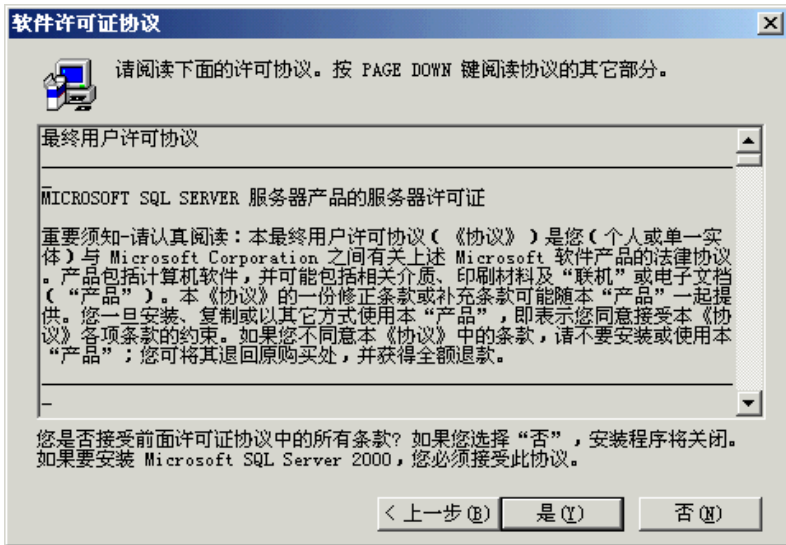


图 1-18 SQL Server 安装步骤三

第四步：出现“安装定义”对话框，选择“服务器和客户端工具”，并单击“下一步”。如图 1-19 所示。



图 1-19 SQL Server 安装步骤四

第五步：出现“服务帐户”对话框，这里注意：选择“使用本地系统帐户”，并单击“下一步”。如图 1-20 所示。

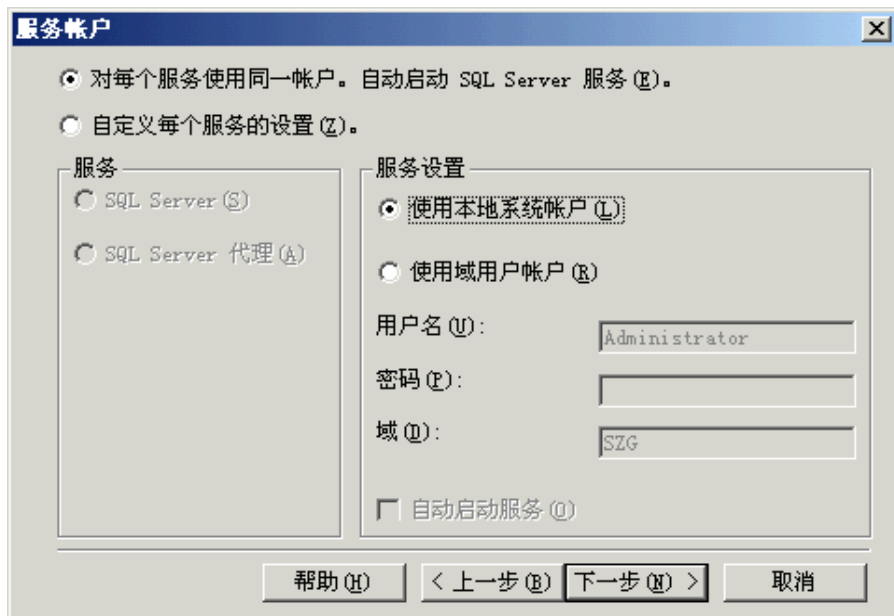


图 1-20 SQL Server 安装步骤五

第六步：设置身份验证模式，默认是 Windows 身份验证模式。这里改为混合模式。如果是初学或者只是自己调试程序使用，这里建议使用空密码，因为后面涉及 SQL Server 登录的程序都采用空密码，如图 1-21 所示。单击“下一步”。

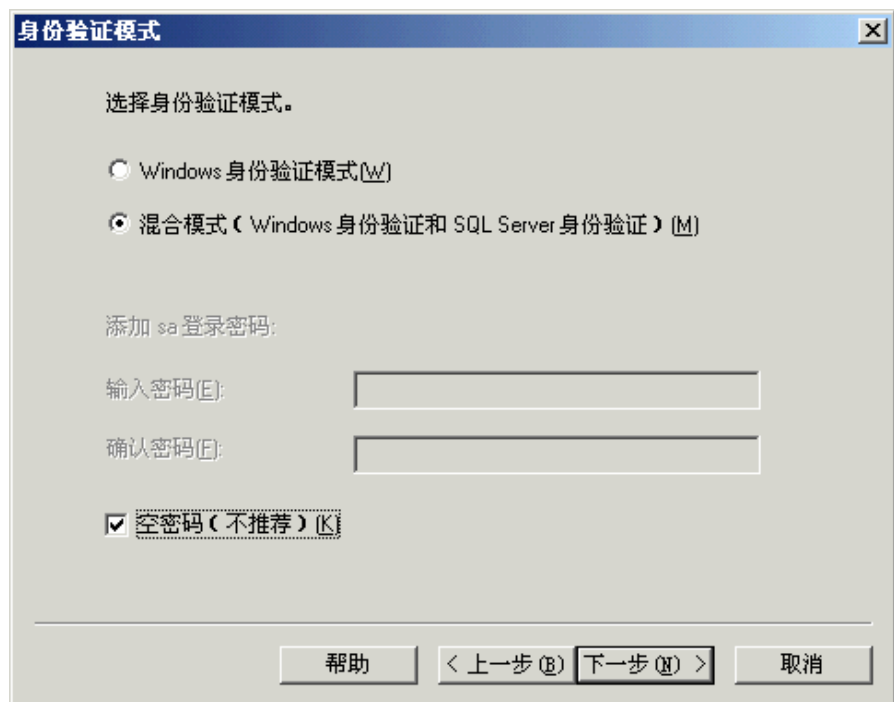


图 1-21 SQL Server 安装步骤六

第七步：进入 SQL Server 2000 的安装进程界面，如图 1-22 所示。

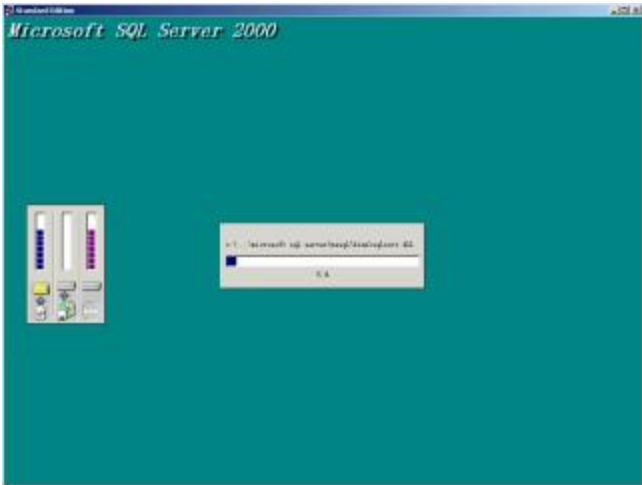


图 1-22 SQL Server 安装步骤七

安装完毕后，系统报告安装成功。选择“开始”→“程序”→“Microsoft SQL Server”→“服务管理器”选项，如图 1-23 所示。

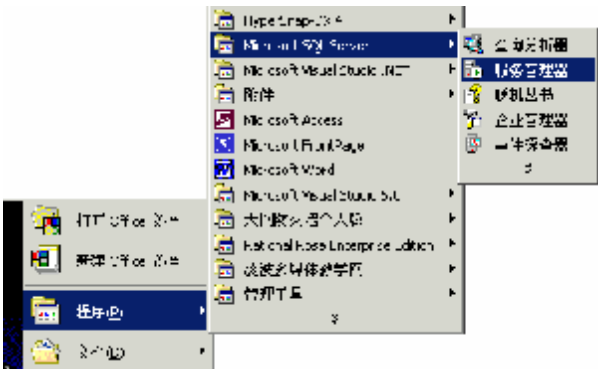


图 1-23 选择服务管理器

单击“开始/继续”启动本地的 SQL Server 服务。启动以后的界面如图 1-24 所示。



图 1-24 启动 SQL Server

1.3.5 测试 ASP+SQL Server 的开发平台

利用 1-02.asp 文件来测试该开发平台。先要装上 SQL Server 数据库。按照上面的安装步骤并将 SQL Server 启动，不用对 SQL Server 做任何的设置。测试程序如程序 1-02.asp 文件所示。

案例名称：测试 ASP+SQL Server 运行环境

程序名称：1-02.asp

```
<%@ language = "Jscript" %>
<%
    var Conn = Server.CreateObject("ADODB.Connection");
    Conn.Open("driver={SQL Server};database=pubs;server=localhost; uid=sa;
    pwd="");
    var rs = Conn.Execute("Select * From Titles");
    for (i = 0; i < rs.Fields.Count; i++)
    {
        Response.Write(rs(i).Name + "<br>");
    }
%>
```

程序用六条语句读取了 SQL Server 中 pubs 数据库的 Titles 表的字段名，该程序每条语句的含义将在后面详细介绍。将该程序依然放在网站的主目录下，程序执行的结构如图 1-25 所示。

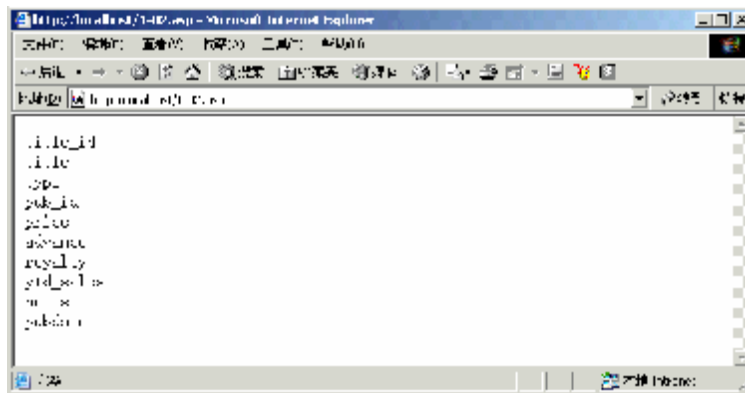


图 1-25 读取 Titles 表的字段名

到此，所有相关配置都已完成，后面章节的 ASP 程序都在该环境下运行。

小结

本章需要重点理解的是：B/S 和 C/S 两大编程体系及其 ASP 在应用程序开发体系中的地

立, 了解 ASP 的发展历史和相关技术。需要熟练掌握的是如何建立 ASP+SQL Server 的开发平台。

课后习题和上机练习

1. ASP, PHP 和 JSP 分别是哪个公司推出的? 各有什么特点?
2. 简要说明 ASP 的发展历史。
3. ASP 支持哪些脚本语言, 分别属于哪个语系?
4. 建立 ASP+SQL Server2000 开发平台, 并编写程序测试。(上机完成)

第 2 章 Web 编程基础

本章要点

本章主要介绍浏览器端的 HTML 和 CSS 编程。首先介绍 HTML 的发展历史，然后介绍 HTML 的基本框架，然后详细介绍了 HTML 的各种常用标记：文字标记、图片标记、超级链接标记，等等。最后介绍 CSS 的基本使用方法，如何让 CSS 与 HTML 协同工作，如何利用 CSS 样式表实现页面元素的显示和隐藏。

2.1 HTML 概述

要把信息发布到全球，就必须使用能够被大众接受的语言，也就是使用一种大多数计算机能够识别的语言。在 Internet 上，通常使用的发布语言是 HTML，即超文本标识语言。

2.1.1 HTML 发展概述

在 20 世纪 90 年代 Web 网络的迅速兴起，使得 HTML 空前繁荣。当时，HTML 被发展成了许多不同的版本。出于解决这种混乱局面的考虑，迫切需要制定一个公认的 HTML 语言规范。

1995 年 11 月，Internet Engineering Task Force (IETF) 整理了以前的各种版本，倡导并主持开发 HTML2.0 规范，同年推出 HTML3.0 技术规范。

1996 年，World Wide Web Consortium (W3C) 的 HTML Working Group 开始组织编写新的规范，于 1997 年 1 月推出了 HTML3.2。在 HTML3.2 中做了许多重要改动。到 1999 年下半年推出到现在依然使用的 HTML4.0。

2.1.2 建立 HTML 网页框架

任意打开一个网页，查看它的源代码，就可以看到这个网页的 HTML 代码。无论该网页如何复杂，都是由一个最基本的框架组成的，HTML 的基本框架如程序 2-01.htm 所示。

案例名称：HTML 网页框架

程序名称：2-01.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
```

```
</BODY>
```

```
</HTML>
```

这是一段最基本的 HTML 标识,任何 HTML 文档都是由一个<HTML>和</HTML>标记包含的,然后分为<HEAD>和<BODY>两大部分,页面的标识一般都是在<BODY>标识中定义的。

注意:HTML 文件的扩展名可以是.htm 或者.html 都可以,现在已经没有区别了。原来在 Linux 操作系统上用.html 作为文件的扩展名,而在 Windows 操作系统上用.htm,因为早期的 Windows 不支持三个以上字母的文件扩展名。

2.2 HEAD 头元素

HEAD 头元素主要包括该页面的一些基本描述语句。

META 的属性包括:Description,网页的描述信息;Keywords,关键字,当搜索引擎查找时,按此关键字查找;Content-type,用来设置该网页的编码;Author,用来设置该网页的作者姓名;Refresh,用来设置网页的自动更新。当CONTENT="3; URL=http://www.263.net"时,该网页打开 3 秒钟后,就自动的转到 www.263.net 网站,使用方法如程序 2-02.htm 所示。

案例名称:HTML 网页框架

程序名称:2-02.htm

```
<HTML>
```

```
<HEAD>
```

```
<META NAME="Description" CONTENT="The Page Of HTML">
```

```
<META NAME="Keywords" CONTENT="Good,Better,Best">
```

```
<META HTTP-EQUIV="Content-type" CONTENT="Text/html; charset=gb2312">
```

```
<META NAME="Author" CONTENT="Zhou RunFa">
```

```
<META HTTP-EQUIV="Refresh" CONTENT="3; URL=http://www.263.net">
```

```
<TITLE>我的第一页面</TITLE>
```

```
</HEAD>
```

```
<BODY></BODY>
```

```
</HTML>
```

当该页面被打开,3 秒后就会自动转到 www.263.net 网站,Title 标记显示在浏览器的标题栏上,如图 2-1 所示。



图 2-1 使用 META 元素建立页面

2.3 HTML 的常用标记

HTML 的常用标记有一些共同特点：都放在 BODY 标记里面。

2.3.1 字体标记

处理文字时通常利用如“ xx”的标记，定义字符 xx 的字体显示为隶书，字号是 40，颜色是红色。程序 2-03.htm 说明如何使用文字格式。

案例名称：使用字体标记

程序名称：2-03.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <FONT FACE="隶书" SIZE="5" COLOR="Blue"> 本书主要对 ASP 网页编程作了系统的介绍，
    本书的特色是以案例为主，以知识点为主线。全书有 30 个完整的案例和超过 200 个基本程序。
  </FONT>
</BODY>
</HTML>
```

程序 2-03.htm 利用标记定义的文字显示结果如图 2-2 所示。

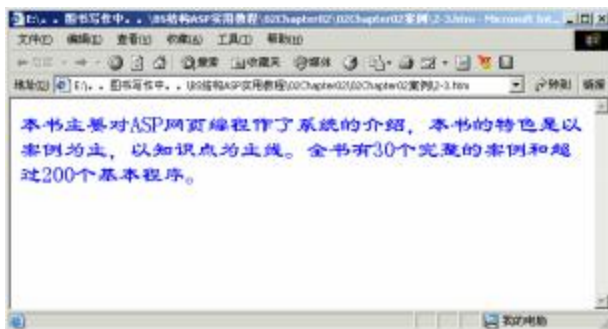


图 2-2 使用字体标记

文字标记中，Face 属性的功能是设置字体，双击“控制面板”中的“字体”，可以看到可以引用字体的名称，这些字体的名称都可以给 FACE 属性赋值。

注意：

- (1) HTML 文件不区分大小写，书写起来比较灵活；
- (2) HTML 文件可以在资源管理器中双击该文件来执行。

2.3.2 图片格式

利用 “” 格式可以插入一张图片，myimage.jpg 文件必须和该 HTML 文件放在同一个目录下。IMG 是 HTML 的一个标记，是 IMAGE 的缩写；SRC 属性给出要连接的图片的路径和文件名，使用方法如程序 2-04.htm 所示。

案例名称：使用图片标记

程序名称：2-04.htm

```
<HTML>
<HEAD>
  <TITLE></TITLE>
</HEAD>
<BODY>
  <IMG SRC="myimage.jpg" WIDTH="300" HEIGHT="200" BORDER="10">
</BODY>
</HTML>
```

使用 IMG 标记将 myimage.jpg 图片插入到 HTML 的文档中，WIDTH 属性和 HEIGHT 属性分别设置该图片的宽度和高度，单位是像素。要去掉图片的黑框，只要将 BORDER 属性设置为 0 就可以了，显示的结果如图 2-3 所示。



图 2-3 使用图片标记

2.3.3 超级链接

使用超级链接的基本的语法是：XX。XX 是一个超级链接，连接到 Address.htm 文件；<A>是单词 Anchor 的缩写，中文的意思是“锚”，功能是从一个页面链接到另一个页面；属性 HREF 定义的是链接到哪一页。具体使用方法如程序 2-05.htm 所示。

案例名称：使用超级链接

程序名称：2-05.htm

```
<HTML>
<HEAD>
    <TITLE></TITLE>
</HEAD>
<BODY>
    其他文件<A HREF="2-05.htm">上一个页面</A><BR>
    位于北京的<A HREF="http://www.tsinghua.edu.cn">清华大学</A>
</BODY>
</HTML>
```

当单击“上一个页面”时，就自动转到程序 2-05.htm 文件。当单击“清华大学”超级链接时，就自动连接到清华大学的网站，显示的结果如图 2-4 所示。

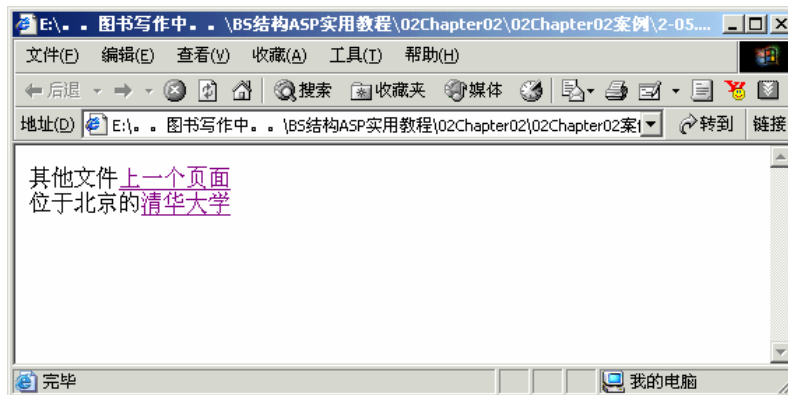


图 2-4 使用超级链接

2.3.4 列表

列表有两种方式，一种是有序列表，另一种是无序列表。

1. 有序列表

无序列表是所有的行之前都有一个小方块，而有序列表是自动排序的，前面有序号，使用方法如程序 2-06.htm 所示。

案例名称：使用有序列表

程序名称：2-06.htm

```
<HTML>
<HEAD>
    <TITLE></TITLE>
```

```
</HEAD>
<BODY>
  <OL>
    <LI>热爱祖国</LI>
    <LI>热爱人民</LI>
    <LI>热爱党</LI>
  </OL>
</BODY>
</HTML>
```

有序列表显示时，会自动按照顺序编号，显示结果如图 2-5 所示。

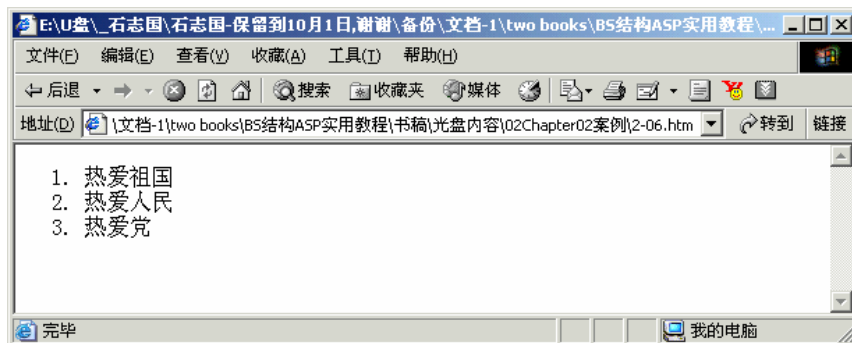


图 2-5 有序列表的效果

2. 无序列表

无序列表在每一行前面会自动出现一个小方块。与有序列表不同的地方是有序列表是以开始和结尾的，而无序列表是以开始和结尾的，使用方法如程序 2-07 所示。

案例名称：使用无序列表

程序名称：2-07.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
<BODY>
  <UL>
    <LI>热爱祖国</LI>
    <LI>热爱人民</LI>
    <LI>热爱党</LI>
  </UL>
</BODY>
```

</HTML>

无序列表显示的结果是，所有行前面都有一个小方块，不分先后。显示的结果如图 2-6 所示。

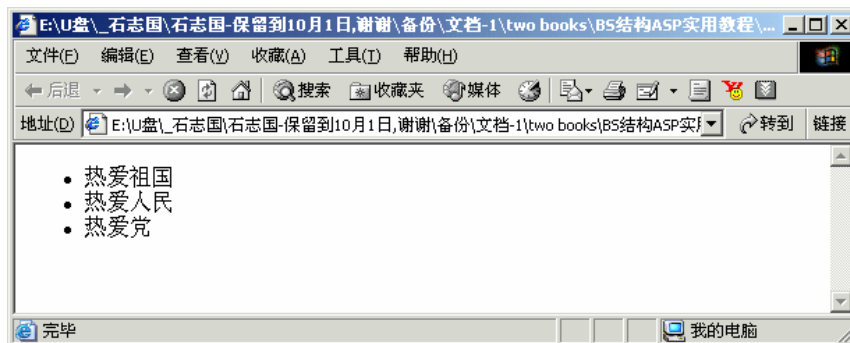


图 2-6 无序列表的效果

2.3.5 表格

表格的属性很多，也是网页设计上最常用的元素。

1. 基本表格

<TABLE>是表格的基本标记。<TR>代表表格的行，<TD>代表表格的列。定义一个三行两列的表格如程序 2-08.htm 所示。

案例名称：基本表格

程序名称：2-08.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER="1" >
      <TR>  <TD>第一行第一列</TD><TD>第一行第二列</TD> </TR>
      <TR>  <TD>第二行第一列</TD><TD>第二行第二列</TD> </TR>
      <TR>  <TD>第三行第一列</TD><TD>第三行第二列</TD> </TR>
    </TABLE>
  </BODY>
</HTML>
```

显示的结果如图 2-7 所示。



图 2-7 基本表格

可以将 BORDER 属性设置为 0，网页中就看不见表格边框。

2. 表格的灵活应用

(1) ROWSPAN 和 COLSPAN 属性的使用方法。利用 ROWSPAN 属性设置该单元格占用多行，利用 COLSPAN 属性设置该单元格是占用多列。使用方法如程序 2-09.htm 所示。

案例名称：跨行和跨列

程序名称：2-09.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER="1" >
      <TR>
        <TD ROWSPAN="2">跨两行</TD>
        <TD COLSPAN="2">跨两列</TD>
      </TR>
      <TR>
        <TD>1000</TD>
        <TD>1000</TD>
      </TR>
      <TR>
        <TD>3000</TD>
        <TD>2000</TD>
        <TD>4000</TD>
      </TR>
    </TABLE>
  </BODY>
```

</HTML>

显示的结果如图 2-8 所示。

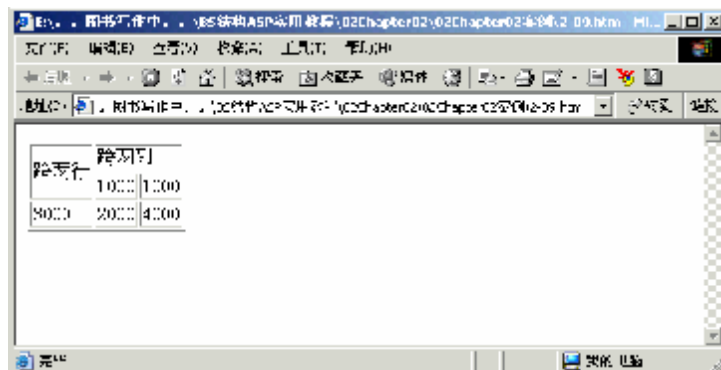


图 2-8 基本表格的灵活应用

(2) Cellpadding 和 Cellspacing 属性的使用方法。Cellpadding 的意思是单元格的边距，指的是字与单元格边框的距离。Cellspacing 的意思是单元格间距，指的是单元格之间的距离。使用方法如程序 2-10.htm 所示。

案例名称：边距和间距

程序名称：2-10.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
<BODY>
<TABLE BORDER="1" CELLPADDING="10" CELLSPACING="0">
  <TR>
    <TD>第一行第一列</TD>
    <TD>第一行第二列</TD>
  </TR>
  <TR>
    <TD>第二行第一列</TD>
    <TD>第二行第二列</TD>
  </TR>
</TABLE><BR>
<TABLE BORDER="1" CELLPADDING="0" CELLSPACING="10">
  <TR>
    <TD>第一行第一列</TD>
    <TD>第一行第二列</TD>
  </TR>
  <TR>
```

```
<TD>第二行第一列</TD>
<TD>第二行第二列</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

显示的结果如图 2-9 所示。

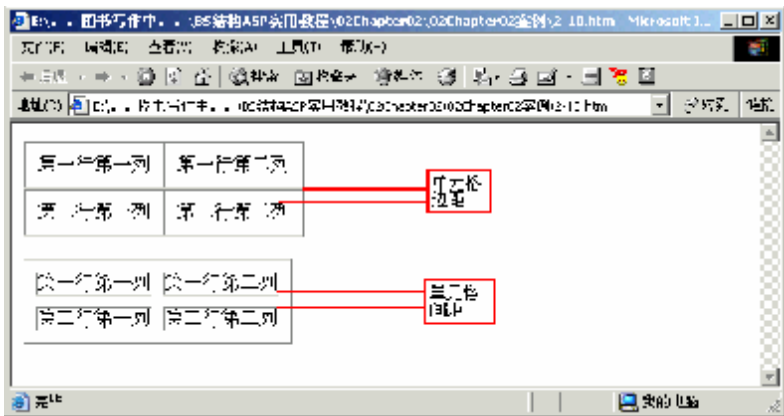


图 2-9 边距和间距

3. 案例 2-1：表格的样式

可以通过配色，使表格更漂亮更和谐，比如下面的表格。如程序 StyleTable.htm 所示。

案例名称：表格的样式

程序名称：StyleTable.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER="4" BORDERCOLOR="ORANGE">
      <TR BGCOLOR="ORANGE" ALIGN="MIDDLE">
        <TD><FONT COLOR="WHITE"><B>序号</B></FONT></TD>
        <TD><FONT COLOR="WHITE"><B>大学</B></FONT></TD>
        <TD><FONT COLOR="WHITE"><B>师资评分</B></FONT></TD>
        <TD><FONT COLOR="WHITE"><B>学生评分</B></FONT></TD>
        <TD><FONT COLOR="WHITE"><B>设备评分</B></FONT></TD>
      </TR>
      <TR ALIGN="MIDDLE">
        <TD>1</TD><TD>清华大学</TD><TD>100</TD><TD>100</TD><TD>100</TD>
```

```
</TR>
<TR ALIGN="MIDDLE">
<TD>2</TD><TD>北京大学</TD><TD>100</TD><TD>100</TD><TD>97</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

其中 `BorderColor` 属性设置表格边框的颜色, `BGColor` 属性设置背景颜色, `Align` 属性设置表格的对齐方式, `` 标记是将内部的文字加粗显示。显示的结果如图 2-10 所示。

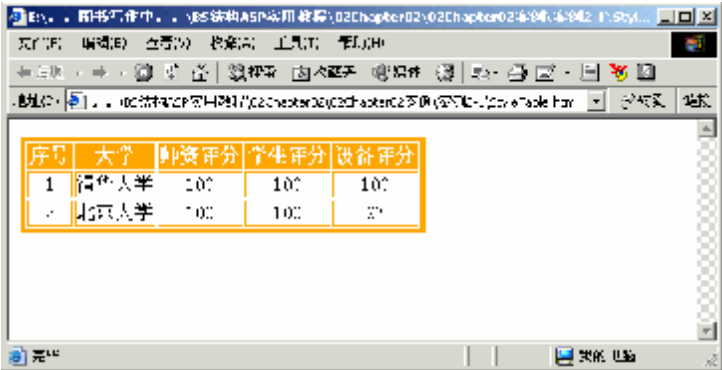


图 2-10 表格的样式

2.3.6 表单

表单的功能是收集用户信息实现系统与用户交互。比如 E-mail 信箱的注册页面就是一个十分典型的表单页面。

表单信息的处理过程如下：当单击表单中的提交按钮时，表单中的信息就会上传到服务器中，然后由服务器端的应用程序（例如 CGI，ASP，PHP，JSP 等）进行处理，处理后将用户提交的信息存储在服务器端的数据库中，或者将有关信息返回到客户端浏览器上。

1. 表单头及其属性

表单的通用格式是：`<FORM METHOD="Post" ACTION="do_submit.asp">XX </FORM>`。表单元素包含在`<FORM>`标记中，有两个重要的属性：`METHOD=“Post”`或`“Get”`，`Post`和`Get`方式的区别在于 `Post` 是一种邮寄的方式，在浏览器的地址栏中不显示提交的信息，这种方式传送的数据量的大小没有限制；`Get` 方式将信息传递到浏览器的地址栏上，最大传输的信息量是 2 KB。当不指明是哪种方式时，默认为 `Get` 方式。`Action` 属性是设置利用哪个文件来处理所提交的数据。使用方法如程序 2-11.htm 所示。

案例名称：表单的基本使用方法
程序名称：2-11.htm

```
<HTML>
```

```

<HEAD>

    <TITLE></TITLE>

</HEAD>

<BODY>

    <FORM METHOD="Post" ACTION="do_submit.asp" >
    用户名: <INPUT TYPE="Text" NAME="UserID"><BR>
    密码:   <INPUT TYPE="Password" NAME="UserPWD"><BR><BR>
           <INPUT TYPE="Submit" VALUE="提交" NAME="B1">
           <INPUT TYPE="Reset" VALUE="重写" NAME="B2">

    </FORM>

</BODY>

</HTML>

```

在表单中加入了一个文本框和一个密码框，单击提交按钮以后就会自动调用当前目录下的 do_submit.asp 文件来处理，显示的结果如图 2-11 所示。

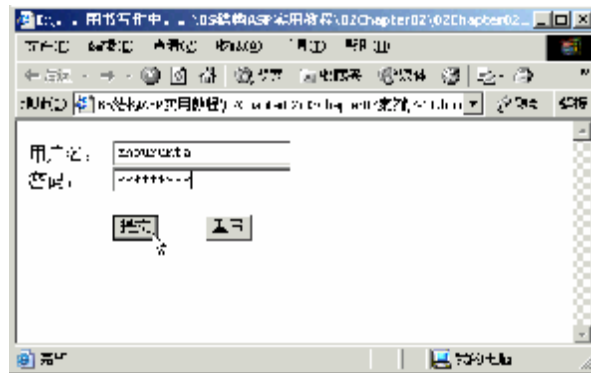


图 2-11 表单头及其属性

下面介绍如何设计表单的元素。

2. 表单中常用控件

在常用的表单制作过程中，经常遇到的是按钮制作、输入元素的制作等。常见的表单控件包括文本框、文本域、密码框、多选框、单选框和下拉列表框，等等。除了文本域和下拉列表，其他只要修改 TYPE 属性就可以了，使用方法如程序 2-12.htm 所示。

案例名称：表单中常用控件

程序名称：2-12.htm

```

<HTML>

    <HEAD>

        <TITLE></TITLE>

    </HEAD>

    <BODY>

        <FORM ACTION="do_submit.asp" METHOD="POST">

```

```

姓名: <INPUT TYPE="TEXT" NAME="USERNAME"><BR>
密码: <INPUT TYPE="PASSWORD" NAME="USERPWD"><BR>
性别: <INPUT TYPE="RADIO" NAME="SEX" CHECKED>男
      <INPUT TYPE="RADIO" NAME="SEX">女 <BR>
血型: <INPUT TYPE="RADIO" NAME="BLOOD" CHECKED>O
      <INPUT TYPE="RADIO" NAME="BLOOD">A
      <INPUT TYPE="RADIO" NAME="BLOOD">B
      <INPUT TYPE="RADIO" NAME="BLOOD">AB <BR>
性格: <INPUT TYPE="CHECKBOX" CHECKED>热情大方
      <INPUT TYPE="CHECKBOX">温柔体贴
      <INPUT TYPE="CHECKBOX">多情善感<BR>
文件: <INPUT TYPE="FILE"><BR>
简介: <TEXTAREA ROWS="8" COLS="30"></TEXTAREA><BR>
城市: <SELECT SIZE=1>
      <OPTION>北京市</OPTION>
      <OPTION>上海市</OPTION>
      <OPTION>南京市</OPTION>
</SELECT><BR>
<INPUT TYPE="BUTTON" VALUE="提交">
<INPUT TYPE="SUBMIT" VALUE="提交">
<INPUT TYPE="RESET" VALUE="RESET">

</FORM>
</BODY>
</HTML>

```

需要注意的是：程序中单选框分成两个组，一个是性别，另一个是血型，同一组必须用同样的名字，如果某一个默认选中，只要加上 **CHECKED** 属性就可以了。两个比较特殊的元素是下拉列表和文本域，它们使用的 **HTML** 标记是 **TEXTAREA** 和 **SELECT**。程序显示的结果如图 2-12 所示。



图 2-12 表单中常用控件

2.3.7 块级元素

块级元素包括 DIV 和 SPAN 两种标记。

1. DIV 标记

DIV 称为层标记，有时也称为块标记。利用 DIV 标记和 CSS 的定位技术可以做出相当出色的效果，DIV 标记的使用方法如程序 2-13.htm 所示。

案例名称：使用 DIV 标记

程序名称：2-13.htm

```
<HTML>
<HEAD>
  <TITLE></TITLE>
</HEAD>
<BODY>
  <DIV ID="MYDIV" STYLE="Background:Blue">I am a layer!</DIV>
</BODY>
</HTML>
```

程序 2-13.htm 显示结果如图 2-13 所示。

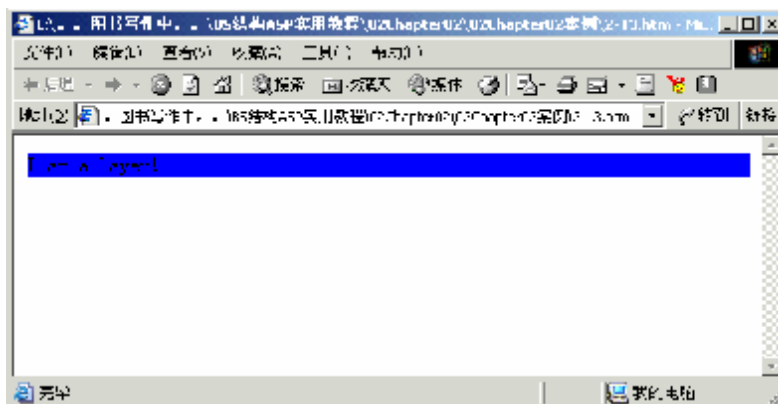


图 2-13 使用 DIV 标记

2. SPAN 标记

SPAN 标记和 DIV 标记的基本语法是一样的，但 SPAN 标记和 DIV 标记的区别还是很大的，如程序 2-14.htm 所示。

案例名称：使用 SPAN 标记

程序名称：2-14.htm

```
<HTML>
```

```
<HEAD>
  <TITLE></TITLE>
</HEAD>
<BODY>
  <SPAN ID="MYDIV" STYLE="Background:Blue">I am a Span!</SPAN>
</BODY>
</HTML>
```

显示的结果如图 2-14 所示。

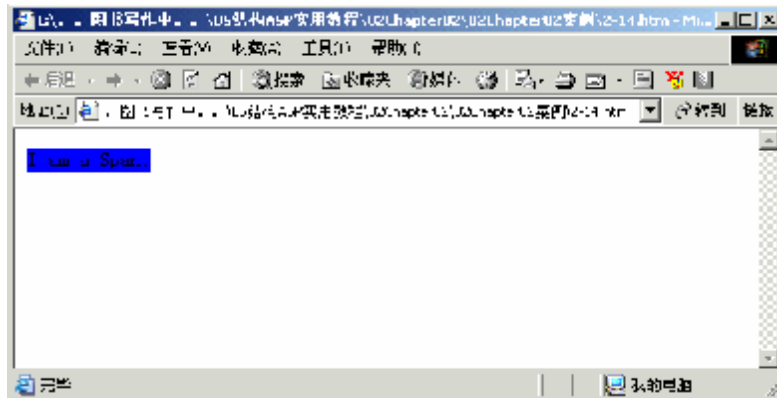


图 2-14 使用 SPAN

使用 SPAN 标记的行只限制到有字的区域，而使用 DIV 标记的行，限制到有字的一整行，不管有没有字符。

2.3.8 预排版标记

包含在预排版标记中的字符会按照 HTML 原码的格式输出到浏览器上。HTML 文件中的英文空格一般不起作用，但是在预排版标记中空格可以显示出来。语法如程序 2-15.htm 文件所示。

案例名称：预排版标记

程序名称：2-15.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <PRE>
      白日依山尽，
      黄河入海流。
    </PRE>
```



```
</BODY>
</HTML>
```

显示到浏览器上时，会按照和源码一样的格式输出，显示的结果如图 2-15 所示。

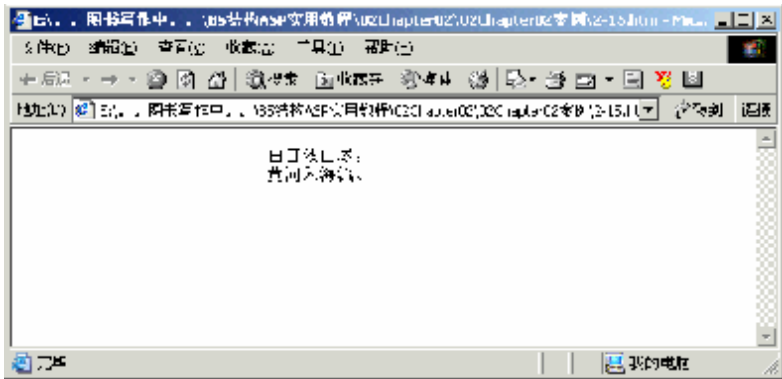


图 2-15 使用 PRE

2.3.9 设计网页框架

设计网页时，经常用到的一种格式是框架。基本网页框架分成两种，左右框架和上下框架。

1. 上下框架

案例名称：上下框架
程序名称：2-16.htm

```
<HTML>
  <HEAD>
    <TITLE> </TITLE>
  </HEAD>
  <FRAMESET ROWS="20%,*" >
    <FRAME NAME="TOP" SRC="TOP.HTM" NORESIZE>
    <FRAME NAME="BOTTOM" SRC="BOTTOM.HTM" NORESIZE>
  </FRAMESET>
</HTML>
```

<FRAMESET ROWS="20%,*">的意思是：基本框架是上下框架，上面占 20%，下面占 80%。
<FRAME NAME="TOP" SRC="TOP.HTM" NORESIZE>的意思是：框架的名称是 top，放入的 HTML 网页是 top.htm，而且不可改变大小。

注意：如果有了 FRAMESET 标记，不可以有 Body 标记，否则无法显示。

在当前的目录下如果有 Top.htm 和 Bottom.htm 两个文件，就可以正常的显示，如图 2-16 所示。

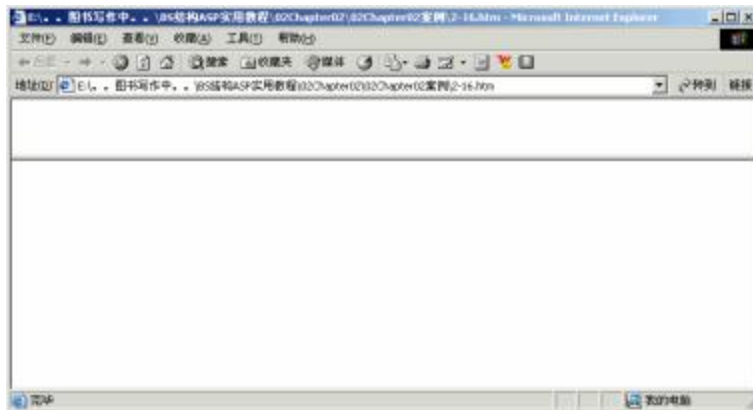


图 2-16 使用上下框架

2. 左右框架

如果使用左右框架，只要将 ROWS 改成 COLS 就可以了。

案例名称：左右框架

程序名称：2-17.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <FRAMESET COLS="20%,*" >
    <FRAME NAME="Left" SRC="Left.htm" NORESIZE>
    <FRAME NAME="Right" SRC="Right.htm" NORESIZE>
  </FRAMESET>
</HTML>
```

本程序使用的是：COLS="20%,*"，意思是框架左边占总宽度的 20%，右边占余下的 80%，显示的结果如图 2-17 所示。

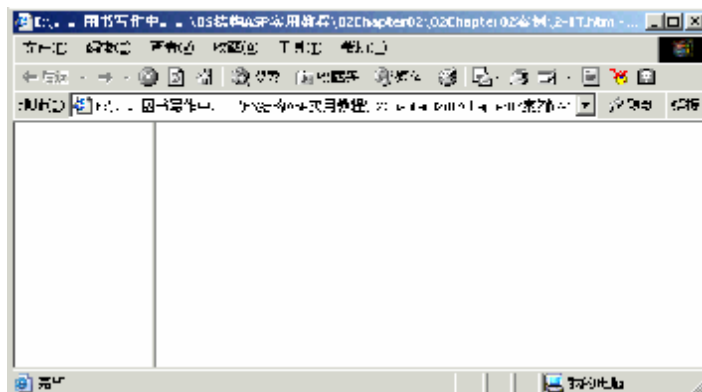


图 2-17 使用左右框架

在实际的工程应用中，一般需要框架套装，但是基本的语法都是上面的两种基本框架格式。

3. 案例 2-2：使用框架

一般在工程应用中，都是由三个页面组成的框架组合，分成上框架，右框架和左框架。

如案例 Frame.htm 文件。

案例名称：使用框架组合

程序名称：Frame.htm

```
<HTML>

  <HEAD>

    <TITLE></TITLE>

  </HEAD>

  <FRAMESET BORDERCOLOR="#8E8E8E" ROWS="96,*" FRAMEBORDER="10">

    <FRAME STYLE="BORDER-BOTTOM: #E7E6F8 2PX SOLID" NAME="TOPFRAME"
      SRC="HEAD.HTM" FRAMEBORDER="0" NORESIZE SCROLLING="NO">

    <FRAMESET STYLE="BORDER-RIGHT: #8E8E8E 4PX SOLID;
      BORDER-TOP: #8E8E8E 4PX SOLID;
      BORDER-LEFT: #8E8E8E 4PX SOLID;
      BORDER-BOTTOM: #8E8E8E 4PX SOLID" FRAMESPACING="4"
      BORDERCOLOR="#8E8E8E" FRAMEBORDER="10" COLS="233,*">

      <FRAME STYLE="BORDER-RIGHT: #DFDFDF 6PX SOLID;
        BORDER-LEFT: #DFDFDF 6PX SOLID;
        BORDER-BOTTOM: #DFDFDF 6PX SOLID" NAME="LEFTFRAME"
        SRC="LEFT.HTM" FRAMEBORDER="0" NORESIZE>

      <FRAME STYLE="BORDER-RIGHT: #DFDFDF 6PX SOLID;
        BORDER-LEFT: #DFDFDF 6PX SOLID;
        BORDER-BOTTOM: #DFDFDF 6PX SOLID" NAME="RIGHTFRAME"
        SRC="RIGHT.HTM" NORESIZE SCROLLING="YES"></FRAMESET>

    </FRAMESET>

  </HTML>
```

程序在框架中加上了一些样式，比如：FRAMEBORDER 代表是否右边框，BORDER-LEFT 代表表格框架左边的颜色，程序显示的结果如图 2-18 所示。

一般情况下工程的主页面都采用这种框架格式，在上面的页面放置网站的图片，右边放置主页面，左边放一个下拉菜单。

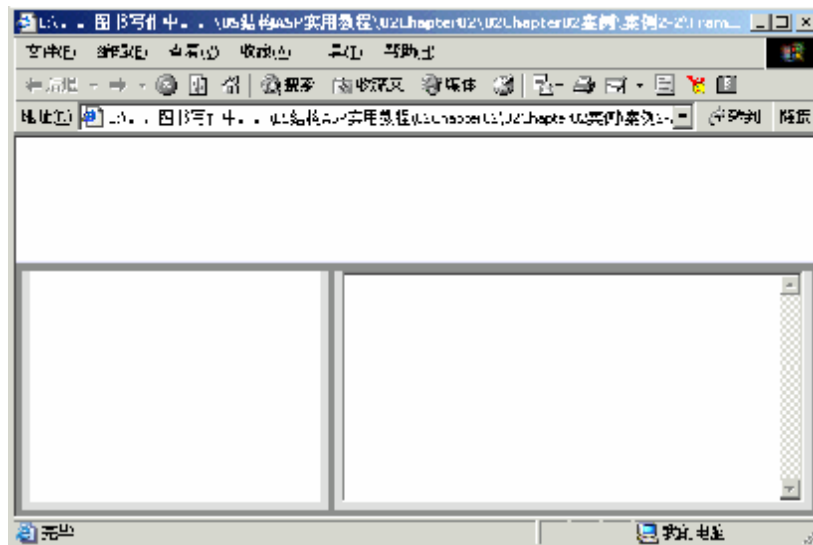


图 2-18 使用框架

2.4 CSS 概述

CSS (Cascading Style Sheets) 中文翻译为层叠样式表单, 简称样式单, 是近几年才发展起来的新技术。1998 年 5 月 12 日, CSS level 2 才成为 W3C 的标准, 它是一组样式, 样式中的属性在 HTML 元素中依次出现, 并显示在浏览器中。样式可以定义在 HTML 文档的标志里, 也可以在外部附加文档作为外加文档。CSS 的功能无比强大, W3C 也极力向世界推广这一先进技术。

为什么要使用 CSS 呢? 原因主要有如下四点。

- (1) 弥补 HTML 对网页格式化功能的不足, 比如段落间距, 行距等。
- (2) 字体变化和大小。
- (3) 页面格式的动态更新。
- (4) 排版定位等。

简单来说, HTML 是一种标记语言, 而 CSS 是这种标记的一种重要扩展, 可以进一步美化页面。换句话说, CSS 是一种用来装饰 HTML 的标记集合。

CSS 样式规则组成如下:

选择符 { 属性: 值 }

单一选择符的复合样式声明应该用分号隔开:

选择符 { 属性 1: 值 1; 属性 2: 值 2 }

程序 2-18.htm 定义了 H1 和 H2 元素的颜色和字体大小属性。

案例名称: 使用 CSS

程序名称: 2-18.htm

```
<HTML>
```

```
<HEAD>
```

```

<TITLE></TITLE>

<STYLE TYPE="TEXT/CSS">
    H1 { FONT-SIZE: X-LARGE; COLOR: RED }
    H2 { FONT-SIZE: LARGE; COLOR: BLUE }
</STYLE>

</HEAD>

<BODY>

    <H1>中国，我的祖国！H1 显示的</H1>

    <H2>中国，我的祖国！H2 显示的</H2>

</BODY>

</HTML>

```

在 HEAD 标记中加上<STYLE TYPE="TEXT/CSS">标记，告诉浏览器内容是 CSS 样式表。H1 标记在浏览器中已经内置了显示样式，在样式表中被重定义了。样式表告诉浏览器用加大、红色字体去显示 H1 标记，用大、蓝色字体去显示二级标题。显示如图 2-19 所示。



图 2-19 使用 CSS 样式表

2.5 加载 CSS 样式的三种方式

使用 CSS 来格式化网页，共有三种方式：在 HEAD 中引用、在 BODY 中引用和作为文件来引用。

2.5.1 HEAD 内引用

这种方式只要在 HEAD 标记中加上 STYLE 标记就可以了，然后在其中定义各种标记的显示样式。

案例名称：HEAD 内引用

程序名称：2-19.htm

```

<HTML>

    <HEAD>

        <TITLE></TITLE>

```

```

<STYLE TYPE="TEXT/CSS">
    H1 {COLOR:GREEN;FONT-SIZE:37PX;}
    P {BACKGROUND:YELLOW;}
</STYLE>
</HEAD>
<BODY>
    <H1>北京大学，清华大学</H1>
    <P>南京大学，复旦大学</P>
</BODY>
</HTML>

```

在样式表中，对 HTML 中的 H1 标记和 P 标记进行了重定义，COLOR 属性设置字体的颜色，FONT-SIZE 属性设置字体的大小。显示的结果如图 2-20 所示。

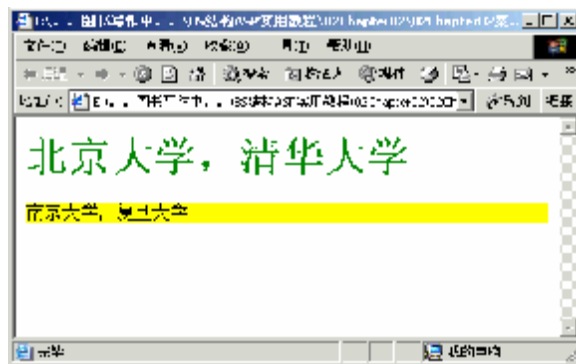


图 2-20 在 HEAD 中使用 CSS

2.5.2 BODY 内引用

在 BODY 中实现主要是在标记中引用，只要将定义在 STYLE 标记中的值拿到对应的标记中就可以了，如程序 2-20.htm 所示。

案例名称：BODY 内引用

程序名称：2-20.htm

```

<HTML>
    <HEAD>
        <TITLE></TITLE>
    </HEAD>
    <BODY>
        <H1 STYLE="COLOR:GREEN;FONT-SIZE:37PX;">北京大学，清华大学</H1>
        <P STYLE="BACKGROUND:YELLOW;">南京大学，复旦大学</P>
    </BODY>
</HTML>

```

显示的结果与图 2-20 相同,但是如果有多个标记需要定义的话,一般还是采用 HEAD 中引用的方式,显示如图 2-21 所示。

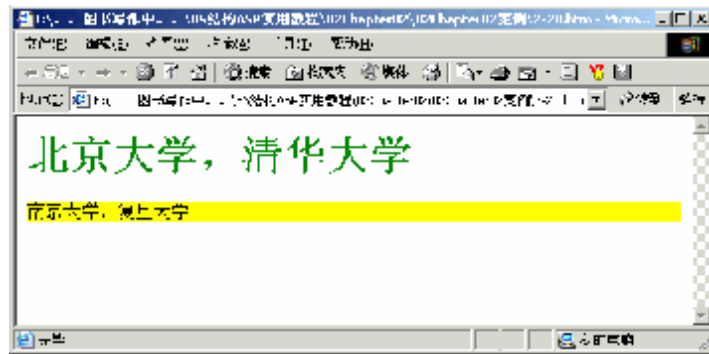


图 2-21 在 Body 中使用样式

2.5.3 文件外引用

可以将 HEAD 中定义的样式另存成一个文件。CSS 作为一个外部文件引入的方式有两种,一种是做链接,另一种是导入。

首先将 STYLE 标记中的内容存成一个文件。

案例名称: 样式表文件

程序名称: mystyle.css

```
H1 {COLOR:GREEN;FONT-SIZE:37PX;}
```

```
P {BACKGROUND:YELLOW;}
```

这里定义了样式,在 HTML 文件中采用 LINK 标记将该样式表链接进入该 HTML 文件。

如程序 2-21.htm 所示。

案例名称: 链接 CSS 文件

程序名称: 2-21.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <LINK REL=STYLESHEET HREF="mystyle.css" TYPE="TEXT/CSS">
  </HEAD>
  <BODY>
    <H1>北京大学, 清华大学</H1>
    <P>南京大学, 复旦大学</P>
  </BODY>
</HTML>
```

显示的结果如图 2-22 所示。

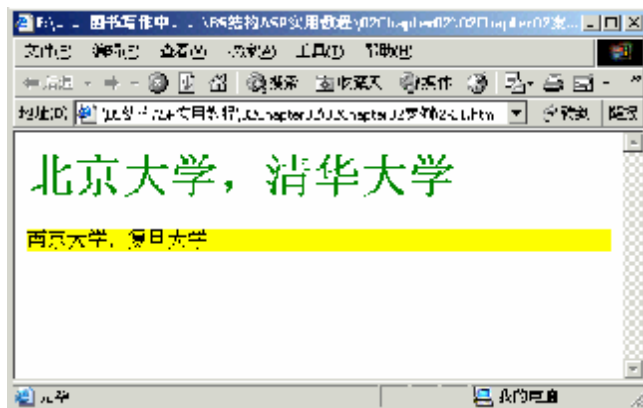


图 2-22 引用 CSS 文件

第二种方法是导入 CSS 文件，如程序 2-22.htm 所示。

案例名称：导入 CSS 文件

程序名称：2-22.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <STYLE TYPE="TEXT/CSS">
      @IMPORT URL(MYSTYLE.CSS);
    </STYLE>
  <BODY>
    <H1>北京大学，清华大学</H1>
    <P>南京大学，复旦大学</P>
  </BODY>
</HTML>
```

使用@IMPORT 关键字将外部文件导入，注意@IMPORT 必须写在 STYLE 标记中。显示的结果如图 2-23 所示。

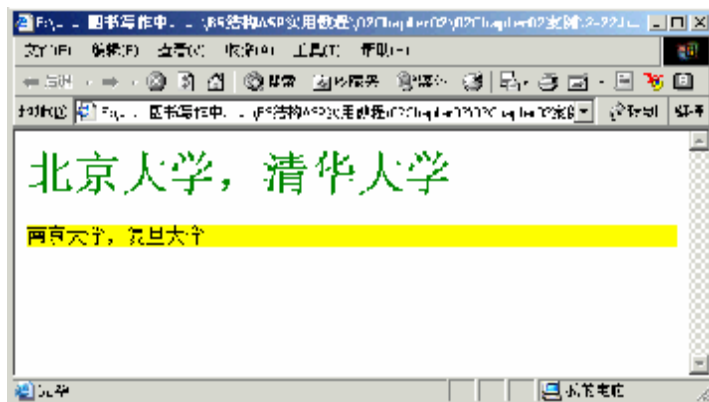


图 2-23 导入 CSS 样式

当某个标记重复定义的时候，按照引用样式的先后，后定义的优先级最高。换句话说，那个标记离要格式化文本最近，就起作用。

2.6 CSS 与标记对应的三种方式

匹配一个 HTML 标记和 CSS 样式表有三种方式：标记选择符、类选择符和 ID 选择符。

2.6.1 标记选择符

任何 HTML 元素都可以是一个 CSS 的选择符。上面所有的样式表都是采用标记选择符引入的。例如：P { BACKGROUND:YELLOW;}，这里用的标记选择符是 P。

2.6.2 类选择符

在 STYLE 标记定义一个“.类名”，然后在 HTML 标记中使用 CLASS=“类名”就可以引入该样式，如程序 2-23.htm 所示。

案例名称：类选择符

程序名称：2-23.htm

```
<HTML>
<HEAD>
  <TITLE></TITLE>
  <STYLE TYPE="TEXT/CSS">
    .LITTLERED{COLOR:RED;FONT-SIZE:9PT}
    .LITTLEGREEN{COLOR:GREEN;FONT-SIZE:9PT}
  </STYLE>
</HEAD>
<BODY>
  <DIV CLASS="LITTLERED">这是红色，而且比较小！</DIV>
  <SPAN CLASS="LITTLEGREEN">这是绿色，而且比较小！</SPAN>
</BODY>
</HTML>
```

这里使用 CLASS 关键字引入定义好的样式（注意样式名称前面有一个英文的“.”）。显示的结果如图 2-24 所示。

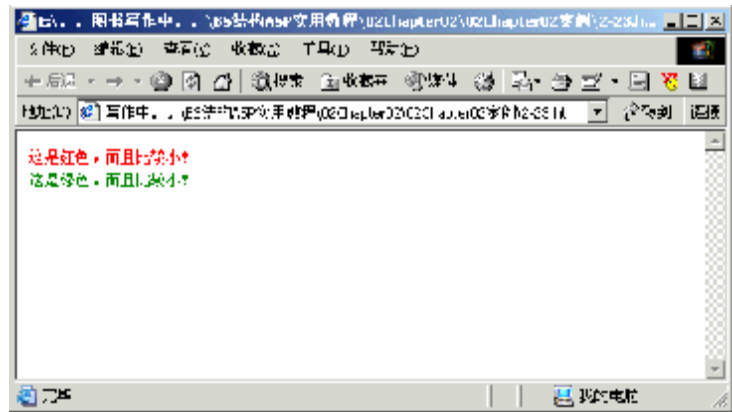


图 2-24 使用类选择符

2.6.3 ID 选择符

定义 ID 选择符时, 在样式名之前加“#名字”, 引用的时候使用“ID=名字”。如程序 2-24.htm 所示。

案例名称: ID 选择符
程序名称: 2-24.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <STYLE TYPE="TEXT/CSS">
      #SZG { COLOR:RED }
    </STYLE>
  </HEAD>
  <BODY>
    <P ID=SZG>这是 ID 选择符号! </P>
  </BODY>
</HTML>
```

这样 P 标记就会按照定义的样式显示成红色, 显示的结果如图 2-25 所示。

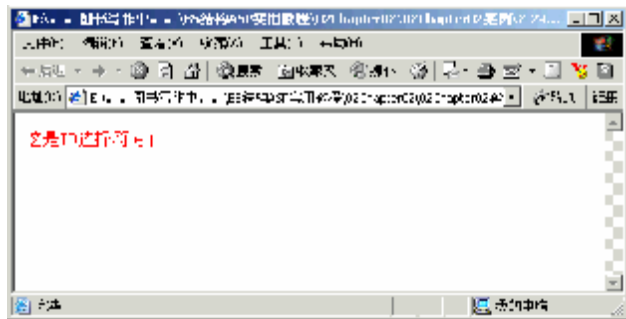


图 2-25 ID 选择符

2.7 定义超级链接样式

可以指定 A 标记以不同的方式显示。一个超级链接有几种不同的状态：未被访问链接（Link）、已访问链接（Visited）和鼠标移动过（Hover）。可以定义超级链接文字的颜色，可以定义字体的大小，一般超级链接都有下划线，可以利用“TEXT-DECORATION:NONE”将超级链接的下划线去掉。

案例名称：定义超级链接样式

程序名称：2-25.htm

```
<HTML>
<HEAD>
  <TITLE></TITLE>
  <STYLE TYPE="TEXT/CSS">
    A:LINK{ COLOR:RED ;FONT-SIZE:9PT;TEXT-DECORATION:NONE}
    A:VISITED{COLOR:BLUE;FONT-SIZE:9PT;TEXT-DECORATION:NONE}
    A:HOVER{COLOR:GREEN;FONT-SIZE:15PT;TEXT-DECORATION:UNDERLINE}
  </STYLE>
</HEAD>
<BODY>
  <A HREF="#">这是超级链接</A>
</BODY>
</HTML>
```

当鼠标移动到其之上时，自动变大并加一个下划线，显示的结果如图 2-26 所示。



图 2-26 定义超级链接

小结

本章作为编程基础，主要介绍两大部分的内容：HTML 的常用标记和 CSS 样式表的使用方法。需要重点掌握的是表格和表单的使用方式、加载 CSS 样式的三种方式和 CSS 与标记对应的三种方式。

课后习题和上机练习

1. 如何在网页中设置字体？有哪些字体可以使用？
2. 如何引入一张图片？如何给图片加上边框？
3. 如何使用超级链接？如何将超级链接的下划线去掉？
4. 如何定义跨行的表格？如何将表格的字体和边框的距离加大？
5. 框架有几种基本形式？如何使用？
6. 加载 CSS 样式的方式有哪些？如何使用？
7. 编写 E-mail 注册的表单。（上机练习）

第 3 章 JavaScript 语言简介

本章要点

本章主要介绍 JavaScript 的基本语法，如何在网页中引入 JavaScript 并实现 JavaScript 和 HTML 交互。介绍 JavaScript 中的变量、数组、表达式、运算符、流程控制语句、JavaScript 的函数、内置对象、浏览器对象的层次和 DOM 模型的建立和使用。

3.1 JavaScript 简介

JavaScript 是一种通用的、基于原型的、面向对象的脚本语言。JavaScript 设计目标是在不与用很多系统和网络资源的情况下，可以在页面做完整的程序。

3.1.1 JavaScript 概述

JavaScript 是一种 Script 脚本语言，所谓的脚本语言就是可以和 HTML 语言混合使用的语言。VBScript 也是 Script 语言中的一种，但是 VBScript 只有微软的浏览器 Internet Explore (IE) 才能完全支持。而 JavaScript 则不管是什么浏览器都可以运行，这也是 JavaScript 的一个优点。

JavaScript 是一种高级的脚本描述性语言，并不需要依赖于特定的机器和操作系统，所以说它是独立于操作平台的。JavaScript 1.0 最初是在 Netscape Navigator 2.0 及 Netscape LiveWire .0 上实现的，目前 JavaScript 的版本是 JavaScript 1.2。

3.1.2 JavaScript 与 Java 的区别

从本质上说 JavaScript 和 Java 没有什么联系，但是同时作为语言，可以从下面的角度来区别。

(1) JavaScript 是解释型的语言，当程序执行的时候，浏览器一边解释一边执行。而 Java 是编译型的语言，必须经过编译才能执行。

(2) 代码格式不一样，Java 代码经过编译后成为二进制文件，而 JavaScript 是纯文本的文件。

(3) 在 HTML 中的嵌入方式不一样。Java 可以通过小应用程序嵌入 HTML 文件，而 JavaScript 可直接写入一个文本文件或 HTML 文件中。

3.1.3 JavaScript 的运行环境

最常用的两种浏览器 IE 和 Netscape Communicator 都支持 JavaScript, IE 3.0 以上和 Netscape

Communicator 3.0 以上都没有问题。

3.2 网页中引入 JavaScript

在页面中引入 JavaScript，只要加上<SCRIPT>标记然后再设置一下所用的语言就可以了，如程序 3-01.htm 文件所示。

案例名称：第一个 JavaScript 程序

程序名称：3-01.htm

```
<HTML>
  <HEAD></HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      document.write("这是以 JavaScript 输出的!")
    </SCRIPT>
  </BODY>
</HTML>
```

给 Script 标记的 Language 属性赋值为：JavaScript。该程序中只有一条语句“document.write()”，该语句的功能是向浏览器输出括号中的字符串。显示如图 3-1 所示。



图 3-1 第一个 JavaScript 程序

注意：JavaScript 区分大小写，HTML 不区分大小写。

3.3 变量与数组

变量和数组是 JavaScript 的基础，JavaScript 和 C 语言属于同一语系，许多基本语法一样。不管是在 JavaScript 中还是在其他程序语言中，最基本的概念是变量。

3.3.1 变量

JavaScript 定义变量只有一个关键字“var”，在 JavaScript 中定义一个用户名变量的语法为：

“var strUserName;”。

注意：

- (1) JavaScript 的变量是弱变量，不经过申明就可以使用，建议先利用 var 申明；
- (2) JavaScript 语句后面可以加分号，也可以不加。但是 C 语系下其他语言都是严格要求加分号的，所以建议加上。

定义字符串变量和定义整型变量的语法如程序 3-02.htm 所示。

案例名称：使用变量

程序名称：3-02.htm

```
<HTML>
<HEAD>
<TITLE></TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    var strWelcome = "欢迎您！<BR>";
    var iCounter = 10;
    iCounter = iCounter + 1;
  </SCRIPT>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    document.write(strWelcome);
    document.write(iCounter);
  </SCRIPT>
</BODY>
</HTML>
```

使用 var 关键字定义一个变量，如果赋值是字符串，那么该变量就是字符串变量，如果赋值是整数，则该变量就是整型变量。显示的结果如图 3-2 所示。

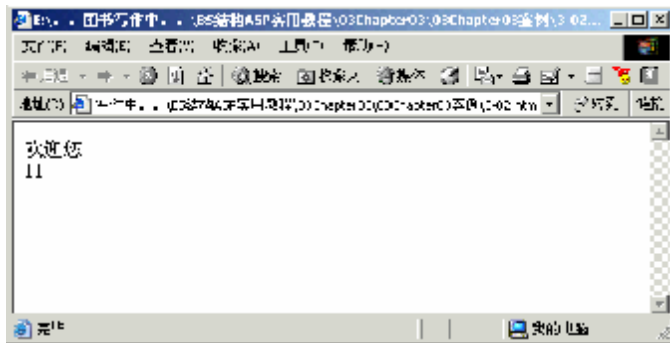


图 3-2 使用变量

变量命名需要遵守以下几个规则。

- (1) 变量命名必须以一个英文字母或是下划线为开头，也就是变量名第一个字符必须是

A 到 Z 或是 a 到 z 之间的字母或是 “_”。

(2) 变量名长度在 0~255 字符之间。

(3) 除了首字符，其他字符可以使用任何字符、数字及下划线，但是不可以使用空格。

(4) 不可以使用 JavaScript 的运算符，例如：+，-，*，/ 等。

(5) 不可以使用 JavaScript 用到的保留字，例如：sqrt（开方），parseInt（转换成整型）等。

(6) 在 JavaScript 中，变量名大小写是有所区别的，例如：变量 s12 和 S12 是不同的两个变量。

3.3.2 数组

简单的说，数组就是由一组数值按照顺序排列在一起，放在同一个变量中，而每个数值都可以利用索引（Index）来得到数组中所存储的信息。

比如使用 “var arrUserName = new Array(2);” 定义含有两个数的一维数组，第一个数的 index 下标是 0，第二个数是 1。C 语系中数组的下标是从 0 开始的，最大值是 N-1（如果定义 Array(N)），所以共有 N 个数。

声明数组时，用 new 和 Array 关键字，new 代表建立一个新的对象，Array 是 JavaScript 内置的一个对象，由于 JavaScript 区分大小写，所以 Array 的首字母必须是大写。使用方法如程序 3-03.htm 所示。

案例名称：使用数组

程序名称：3-03.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      var arrUserName = new Array(2);
      arrUserName[0] = "Bill";
      arrUserName[1] = "Bob";
      document.write(arrUserName[0]);
      document.write("<br>");
      document.write(arrUserName[1]);
      document.write("<br>");
    </SCRIPT>
  </BODY>
</HTML>
```

该程序声明了一个包含两个元素的一维数组，下标是 0 和 1。然后程序将数组中的两个数输出到浏览器上。显示的结果如图 3-3 所示。

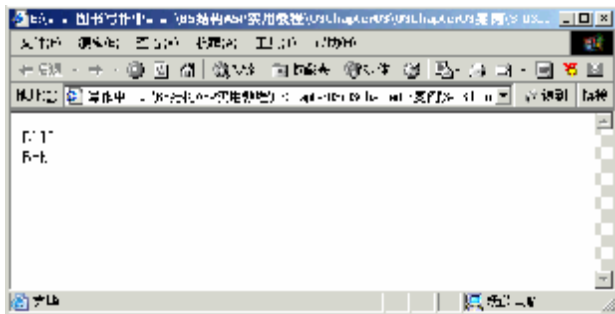


图 3-3 使用数组

如果没有对数组的某个元素赋值，输出这个数组元素，会显示 `undefined`。

3.4 表达式与运算符

程序主要功能是运算，例如加、减、乘、除等基本操作。

3.4.1 算术运算符

算术运算符主要提供加、减、乘、除等操作，计算机中没有通常的乘号，用“*”代替。其余操作用“%”，使用方法如程序 3-04.htm 所示。

案例名称：算术运算符

程序名称：3-04.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      document.write(3+2);
      document.write("<br>");
      document.write(3*2);
      document.write("<br>");
      document.write(3/2);
      document.write("<br>");
      document.write(3-2);
      document.write("<br>");
      document.write(3%2);//取余数
    </SCRIPT>
  </BODY>
</HTML>
```

JavaScript 的行注释语句用 “//注释” 形式，块注释语句利用 “/*注释*/” 形式。该程序计算结果如图 3-4 所示。

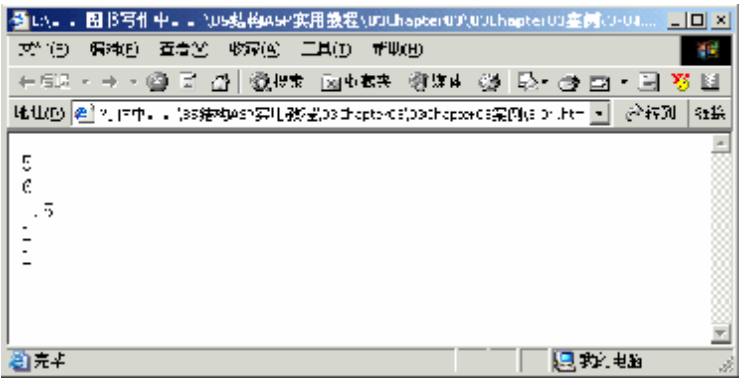


图 3-4 算术运算符

3.4.2 逻辑运算符

逻辑运算符包括：与运算符 “&&”、或运算符 “||” 和取反运算符 “!”。

- (1) 逻辑与运算符&&：在 “A&&B” 中，只有当两个条件 A 和 B 都为真的时候，整个表达式值才为真。
- (2) 逻辑或表达式||：在 “A||B” 中，只要两个条件 A 和 B 有一个成立，整个表达式值就为真。
- (3) 逻辑反表达式!：在!A 中，当 A 为真时，表达式的值为假；当 A 为假时，表达式的直为真。

下面通过案例来说明。如程序 3-05.htm 所示。

案例名称：逻辑运算符
程序名称：3-05.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      document.write(true&&false);
      document.write("<br>");
      document.write(false&&false);
      document.write("<br>");
      document.write(true||false);
      document.write("<br>");
```

```
document.write(!false);  
</SCRIPT>  
</BODY>  
</HTML>
```

程序显示的结果如图 3-5 所示。

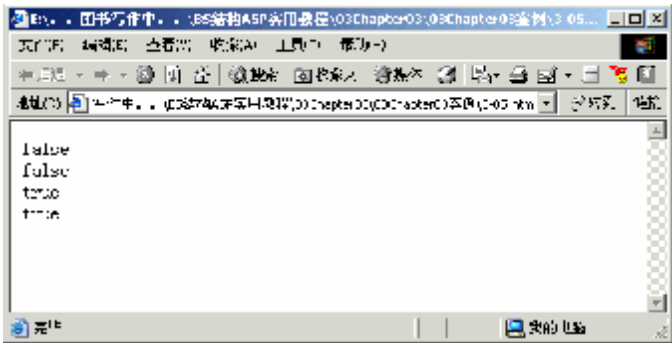


图 3-5 逻辑预算符

3.4.3 字符串运算符

字符串运算符主要是字符串运算符“+”和它的变形形式：“+=”。比如“A=A+B”可以省略一个 A，写成“A+=B”，在 C 语系中这两个表达式是一样。

在程序表达式中，操作符是有一定优先顺序的，叫做运算符的优先级。表 3-1 由上至下列出了操作符从低到高的优先级。

表 3-1 操作符的优先级	
赋值操作符	=, +=, *=, /=, %=, <=, >=, &=, =
条件表达式	? :
逻辑或	
逻辑与	&&
按位或	
按位异或	^
按位与	&
比较操作符	==, !=
关系操作符	<, <=, >, >=
算术操作符	+, -
算术操作符	*, /, %
增量操作符	!, ~, ++, --

字符串运算符的使用方法如程序 3-06.htm 所示。

案例名称：字符串运算符

程序名称：3-06.htm

```
<HTML>  
  <HEAD>  
    <TITLE></TITLE>
```

```
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    var strHello = "网页编程";
    var strResult = "你好, ";
    strResult += strHello;
    //等价于: strResult = strResult + strHello;
    document.write(strResult);
  </SCRIPT>
</BODY>
</HTML>
```

程序显示的结果如图 3-6 所示。

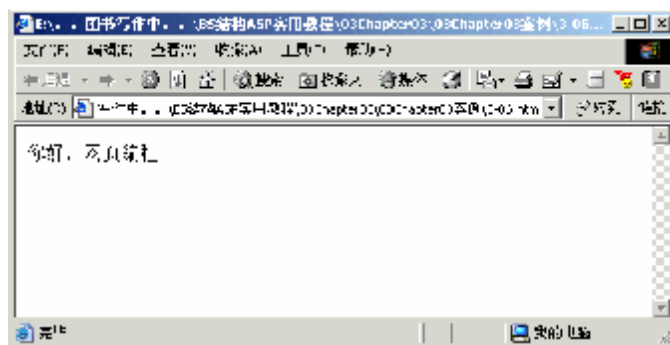


图 3-6 字符串运算符

3.4.4 条件表达式

条件表达式的基本语法是：“（条件）？ A： B”，如果表达式中的条件为真，则表达式取 A 的值；否则，表达式取 B 的值。使用方法如程序 3-07.htm 所示。

案例名称：条件表达式

程序名称：3-07.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      a = (4>3) ? 5 : 7;
      b = (4<3) ? 5 : 7;
      document.write(a);
```

```
document.write("<br>");  
document.write(b);  
</SCRIPT>  
</BODY>  
</HTML>
```

在第一个表达式中, $4 > 3$ 成立, 则 a 的值取 5; 在第二个表达式中, $4 < 3$ 不成立, b 的值仅 7; 所以输出的结果为 5 和 7, 实际上该表达式实现二重选择分支。显示的结果如图 3-7 所示。

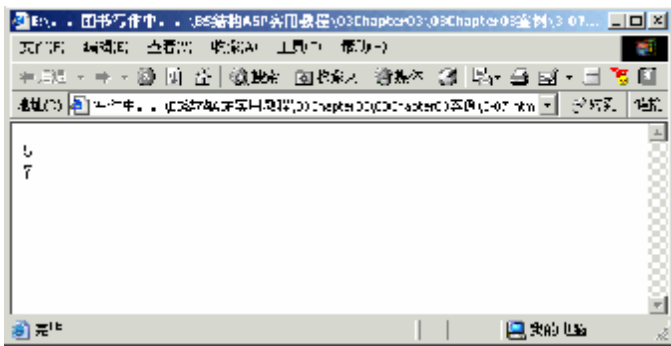


图 3-7 条件表达式

3.5 流控制语句之条件语句

JavaScript 提供的语句可以分为以下几大类。

- (1) 条件和分支语句: If...else 语句, switch 语句。
- (2) 循环语句: for 语句, do...while 语句, break 语句和 continue 语句。
- (3) 对象操作语句: new, this 和 with。
- (4) 注释语句: “//” 或 “/* */”。

3.5.1 if 语句

if 语句完成程序流程中的分支功能, 如果条件成立, 则执行相应的语句。二重分支结构的基本语法如程序 3-08.htm 所示。

案例名称: if 语句

程序名称: 3-08.htm

```
<HTML>  
  <HEAD>  
    <TITLE></TITLE>  
  </HEAD>  
<BODY>
```

```
<SCRIPT LANGUAGE="JavaScript">
    var iHour = 13;
    if (iHour < 12)
    {
        document.write("早上好! ");
    }
    else
    {
        document.write("下午好! ");
    }
</SCRIPT>
</BODY>
</HTML>
```

程序中, 因为 iHour 为 13, 条件 “iHour < 12” 不成立, 执行 else 分支中的语句, 所以输出了 “下午好”。显示的结果如图 3-8 所示。

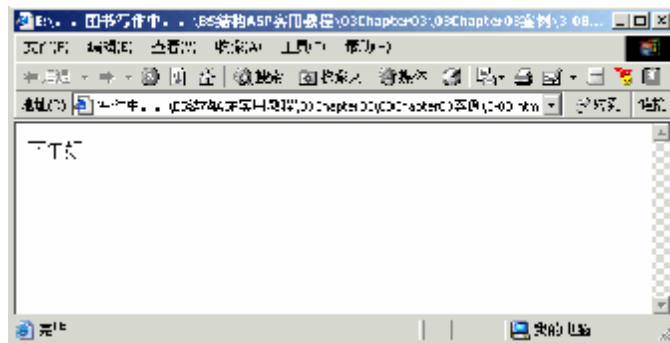


图 3-8 if 语句

3.5.2 switch 语句

分支语句 switch 可以根据一个变量的不同取值而采取不同的处理方法, 这样的语句叫做多重分支语句。如程序 3-09.htm 所示。

案例名称: switch 语句

程序名称: 3-09.htm

```
<HTML>
<HEAD>
    <TITLE></TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
```

```
var val = "";
var i = 5;
switch(i)
{
    case 1:
        val = "一"; break;
    case 2:
        val = "二"; break;
    case 3:
        val = "三"; break;
    case 4:
        val = "四"; break;
    case 5:
        val = "五"; break;
    default:
        val = "不知道";
}
document.write(val);
</SCRIPT>
</BODY>
</HTML>
```

程序中，通过“switch(i)”中的 i 和后面 case 后的数值比较，如果相同就会进入该分支。注意 break 语句的意思：如果不加 break 语句，当进入该分支后，就会继续执行后面的分支语句。如果没有相同的值就会执行 default 分支。显示的结果如图 3-9 所示。

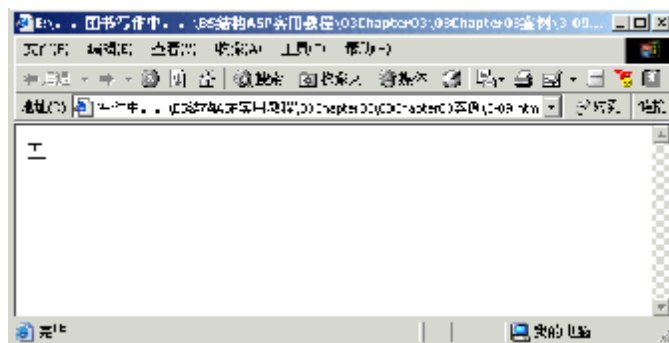


图 3-9 switch 语句

3.6 流控制语句之循环语句

循环语句包括：for 语句、while 语句、break 语句和 continue 语句。

3.6.1 for 语句

for 语句的基本语法如下。

```
for (初始化部分; 条件部分; 更新部分)
{
    语句块
}
```

这里的三个部分，都是可以省略的，但是两个分号不可以省略！

案例名称：for 语句

程序名称：3-10.htm

```
<HTML>
<HEAD>
    <TITLE></TITLE>
</HEAD>
<BODY>
    <SCRIPT LANGUAGE="JavaScript">
        var iSum = 0;
        for(var i = 0; i <= 100; i++)
        {
            iSum += i;
        }
        document.write(iSum);
    </SCRIPT>
</BODY>
</HTML>
```

程序中的 for 循环运行了 101 次，统计 1~100 所有整数的和，语句“i++”功能和“i=i+1”的功能一样。结果如图 3-10 所示。

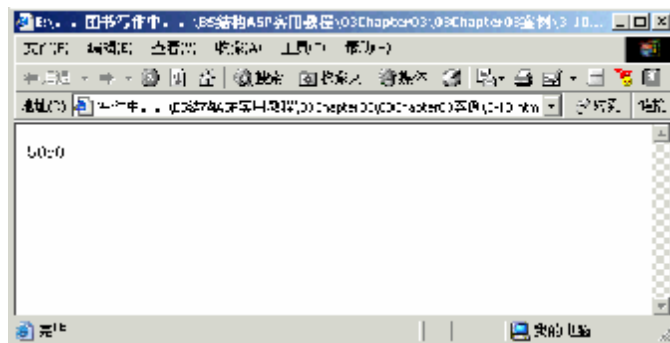


图 3-10 for 语句

3.6.2 while 语句

while 语句也可以控制循环，语法如程序 3-11.htm 所示。

案例名称：while 语句

程序名称：3-11.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      var iSum = 0;
      var i = 0;
      while( i<=100 )
      {
        iSum += i;
        i++;
      }
      document.write(iSum);
    </SCRIPT>
  </BODY>
</HTML>
```

程序依然输出了 1~100 所有整数的和。当“while(i<=100)”中的条件不成立时就退出循环。程序显示结果如图 3-11 所示。



图 3-11 while 语句

3.6.3 break 与 continue 语句

break 语句的功能是结束整个循环，continue 语句的功能是结束当前这一次循环，但是并

没有跳出整个的循环，如程序 3-12.htm 所示。

案例名称: break 语句

程序名称: 3-12.htm

```
<HTML>

<HEAD>

<TITLE></TITLE>

</HEAD>

<BODY>

<SCRIPT LANGUAGE="JavaScript">

    for(i = 1; i < 20; i++)

    {

        if(i%5 == 0)

        {

            break;

        }

        document.write(i + "<br>");

    }

</SCRIPT>

</BODY>

</HTML>
```

i 循环到 5 的时候，条件成立执行 break 语句跳出整个循环，所以只输出 1 到 4。显示的结果如图 3-12 所示。

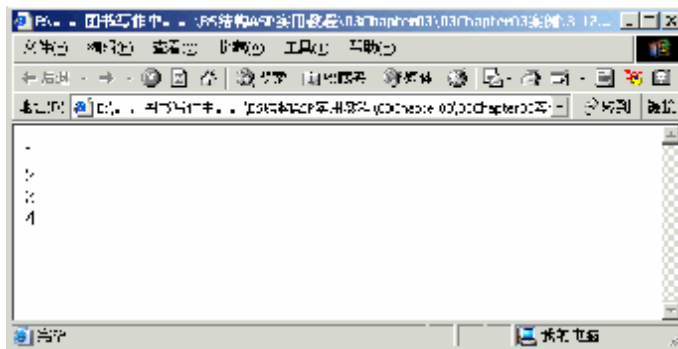


图 3-12 break 语句

将程序 3-12.htm 中 break 语句修改成 continue 语句，如程序 3-13.htm 所示。

案例名称: continue 语句

程序名称: 3-13.htm

```
<HTML>

<HEAD>
```

```
<TITLE></TITLE>

</HEAD>

<BODY>

  <SCRIPT LANGUAGE="JavaScript">
    for(i = 1; i < 20; i++)
    {
      if(i%5 == 0)
      {
        continue;
      }
      document.write(i + "<br>");
    }
  </SCRIPT>

</BODY>

</HTML>
```

当条件成立时执行 `continue` 语句，只结束当前的循环，所以 5 的倍数都没有输出。程序执行结果如图 3-13 所示。

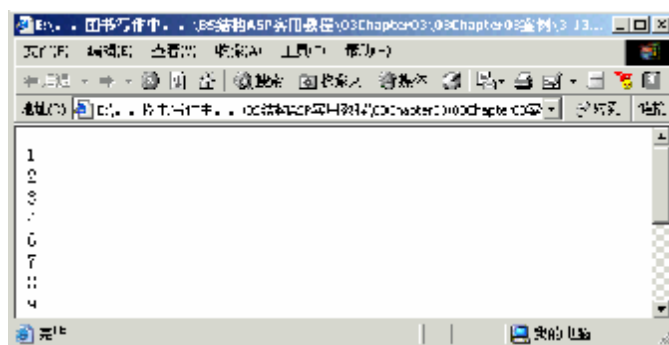


图 3-13 使用 `continue` 语句

3.7 JavaScript 函数

3.7.1 函数定义

函数在定义时并没有被执行，只有函数被调用时，其中的代码才真正被执行。为了实现函数的定义和调用，JavaScript 语句提供了两个关键字：`function` 和 `return`。JavaScript 函数的基本语法如下：

```
function 函数名称 (参数表)
{
  语句块;
```

```
}
```

return 语句将结束函数并返回后面表达式的值, return 语句的语法为:“return 表达式;”函数结束时可以没有 return 语句,但是只要遇到 return 语句函数就结束。函数的定义和调用如程序 3-14.htm 所示。

案例名称: 函数定义和调用

程序名称: 3-14.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      function getSqrt(iNum)
      {
        var iTemp = iNum * iNum;
        document.write(iTemp);
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      getSqrt(8);
    </SCRIPT>
  </BODY>
</HTML>
```

程序定义了一个函数,该函数没有返回值。每次调用就会将相应的内容显示到浏览器上。显示的结果如图 3-14 所示。

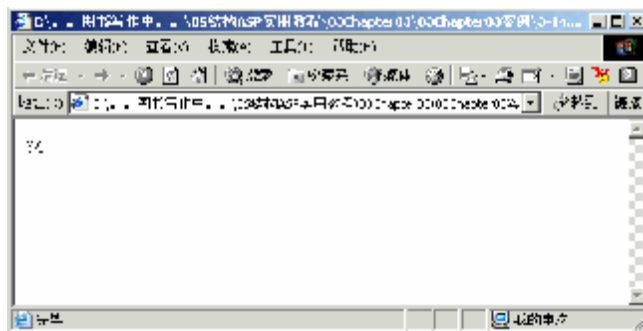


图 3-14 函数定义

3.7.2 函数调用

使用 return 语句返回函数的返回值,如程序 3-15.htm 所示。

案例名称：函数的返回值

程序名称：3-15.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      function f(y)
      {
        var x = y * y;
        return x;
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      for(x = 0; x < 10; x++)
      {
        y = f(x);
        document.write(y);
        document.write("<br>");
      }
    </SCRIPT>
  </BODY>
</HTML>
```

函数调用的结果如图 3-15 所示。



图 3-15 函数调用

3.8 事件的概念

JavaScript 事件主要包括三大类的事件：超级连接事件，浏览器事件和界面事件。界面事

牛包括：Click（单击）、MouseOut（鼠标移出）、MouseOver（鼠标移过）和MouseDown（鼠标按下）等。

3.8.1 单击事件

鼠标单击事件是常见的事件，语法非常简单：“onclick=函数或是处理语句”，如程序 3-16.htm 所示。

案例名称：单击事件

程序名称：3-16.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <FORM>
      <INPUT TYPE="BUTTON" VALUE="单击" ONCLICK="alert('鼠标单击')">
    </FORM>
  </BODY>
</HTML>
```

当鼠标单击按钮的时候，自动弹出一个对话框，显示的结果如图 3-16 所示。

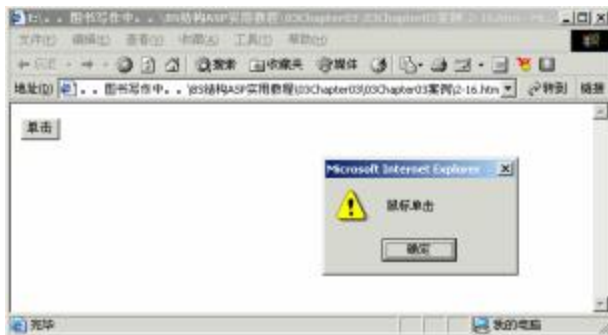


图 3-16 单击事件

3.8.2 处理下拉列表

下拉列表是常用的一种网页元素，一般利用 ONCHANGE 事件来处理。如程序 3-17.htm 所示。

案例名称：处理下拉列表

程序名称：3-17.htm

```
<HTML>
```

```
<HEAD>

</HEAD>

<BODY>

  <SELECT NAME="selAddr" SIZE="1" ONCHANGE="func()">

    <OPTION SELECTED VALUE="北京">北京</OPTION>

    <OPTION VALUE="上海">上海</OPTION>

    <OPTION VALUE="广州">广州</OPTION>

  </SELECT>

  <SCRIPT LANGUAGE="JavaScript">

    function func()

    {

      alert("你选择了" + selAddr.value);

    }

  </SCRIPT>

</BODY>

</HTML>
```

每个下拉列表的 **OPTION** 项都有一个 **VALUE** 值，读出来的是 **VALUE** 属性的值。执行的结果如图 3-17 所示。

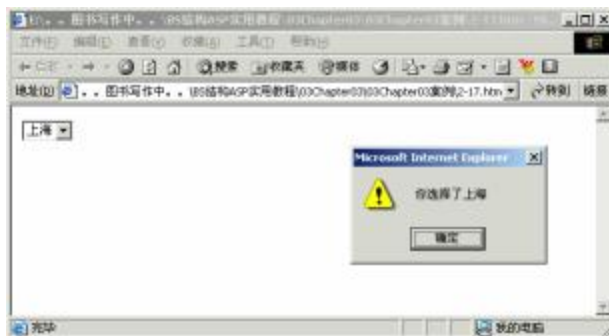


图 3-17 使用下拉列表

案例 3-1：动态按钮

当鼠标移动到按钮上的时候，按钮就会凸起来。这通过 JavaScript 的事件实现起来非常方便。首先必须准备两张图片，当鼠标移上去的时候，自动切换成另一张图片就可以了。实现的方法如程序 `hoverbutton.htm` 文件所示。

案例名称：动态按钮

程序名称：hoverbutton.htm

```
<HTML>

<HEAD>

</HEAD>
```

```
<BODY>

<SCRIPT LANGUAGE="JavaScript">

    function DoOver(oimg)
    {
        var imgSRC;
        imgSRC = 'Edit_' + oimg.name + '_Over.gif';
        oimg.src = 'images/' + imgSRC;
    }

    function DoOut(oimg)
    {
        var imgSRC;
        imgSRC = 'Edit_' + oimg.name + '.gif';
        oimg.src = 'images/' + imgSRC;
    }

</SCRIPT>

<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0">

    <tr>

        <td><img name="New" SRC="images/Edit_New.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)"
            alt="新建" onclick="alert('新建')" style="CURSOR: hand"></td>

        <td><img name="Open" SRC="images/Edit_Open.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)"
            alt="打开" onclick="alert('打开')" style="CURSOR: hand"></td>

        <td><img name="Save" SRC="images/Edit_Save.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="保存"
            onclick="alert('保存')" style="CURSOR: hand"></td>

        <td><img name="SaveAs" SRC="images/Edit_SaveAs.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="另存"
            onclick="alert('另存')" style="CURSOR: hand"></td>

        <td></td>

        <td><img name="Print" SRC="images/Edit_Print.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="打印"
            onclick="alert('打印')" style="CURSOR: hand"></td>

        <td><img name="PrintSetup" SRC="images/Edit_PrintSetup.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="打印设置"
            onclick="alert('打印设置')" style="CURSOR: hand"></td>

        <td></td>

        <td><img name="Cut" SRC="images/Edit_Cut.gif"
            onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="剪切"
            onclick="alert('剪切')" style="CURSOR: hand"></td>
```



```

<td><img name="Copy" SRC="images/Edit_Copy.gif"
onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="复制所选内容"
onclick="alert('复制')" style="CURSOR: hand"></td>
<td><img name="Paste" SRC="images/Edit_Paste.gif"
onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="粘贴"
onclick="alert('粘贴')" style="CURSOR: hand"></td>
<td><img name="Undo" SRC="images/Edit_Undo.gif"
onmouseover="DoOver(this)" onmouseout="DoOut(this)" alt="撤销上次操作"
onclick="alert('撤销')" style="CURSOR: hand"></td>
<td><img name="Redo" SRC="images/Edit_Redo.gif"
onmouseover="DoOver(this)" onmouseout="DoOut(this)"
alt="重新进行已经撤销的操作" onclick="alert('重新进行已经撤销的操作')"
style="CURSOR: hand"></td>
</TR>
</TABLE>
</BODY>
</HTML>

```

程序首先将图片全部放在 `images` 目录下, 图片命名也要按照一定的规则。当鼠标移过的时候, 图片就会自动凸起来, 如图 3-18 所示。



图 3-18 动态按钮

3.9 对象处理语句

3.9.1 this 语句

`this` 语句符总是指的是当前的对象, 这同其他面向对象语言如 C++ 中的 `this` 功能一样。如程序 3-16.htm 所示。

案例名称: `this` 语句

程序名称: 3-16.htm

```
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      function imgclick(oimg)
      {
        alert("图片名称是: " + oimg.name + "\r\n 图片地址是: " + oimg.src );
      }
    </SCRIPT>
    <IMG src="Edit_Commit.gif" NAME="img1" ONCLICK="imgclick(this)">
  </BODY>
</HTML>
```

程序中的函数 `imgclick` 传递了一个 `this`，该 `this` 指的是当前的 `img` 对象，取对象的 `name` 和 `src` 属性，可以得到该对象的名称和地址，“`\r\n`”的功能是回车换行。当单击图片时，就会显示该图片的名称和地址，如图 3-19 所示。

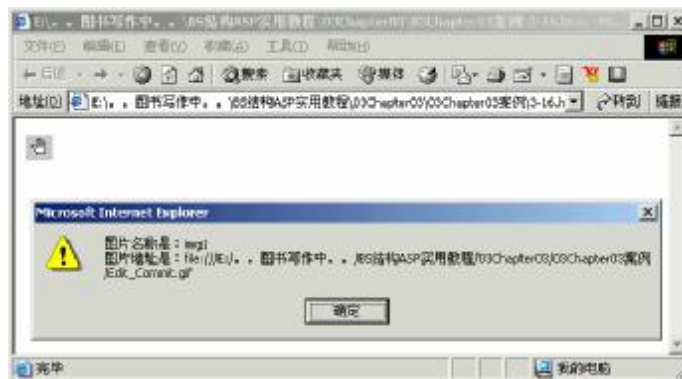


图 3-19 使用 `this` 语句

3.9.2 for...in 语句

如果不知道一个对象中包含多少元素，可以 `for...in` 语句实现循环。`in` 后面跟一个对象，循环的次数是该对象中的所有元素的总和，以数组对象为例，如程序 3-17.htm 所示。

案例名称: `for...in` 语句

程序名称: 3-17.htm

```
<HTML>
  <HEAD>
```

```
<TITLE></TITLE>

</HEAD>

<BODY>

  <SCRIPT LANGUAGE="JavaScript">
    var arr = new Array(3);
    arr[0] = "Jack";
    arr[1] = "Mike";
    arr[2] = "Rose";
    for(i in arr)
    {
      document.write("<br>第" + i + "个为: " + arr[i]);
    }
  </SCRIPT>

</BODY>

</HTML>
```

这里的 Array 是 JavaScript 的内置的对象，利用 new 关键字创建一个实例。Arr 对象中含有三个值，通过循环将三个值全部取出来，如图 3-20 所示。

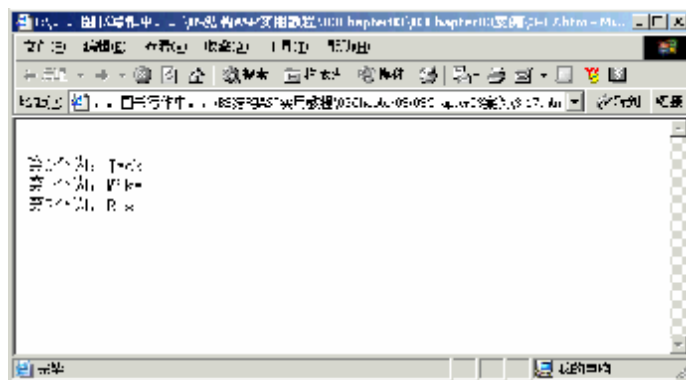


图 3-20 使用 for...in

3.9.3 with 语句

with 语句的作用范围内，如果没有指出所选择的对象，而是使用默认的对象，默认的对象使用 with 语句给出。程序 3-18.htm 利用 with 语句省略了一些 document 对象。

案例名称：with 语句

程序名称：3-18.htm

```
<HTML>

  <HEAD>

    <TITLE></TITLE>
```

```
</HEAD>

<BODY>
<SCRIPT LANGUAGE = "JavaScript">
    with(document)
    {
        write("你好世界<br>");
        write("你好中国<br>");
        write("再见");
    }
</SCRIPT>
</BODY>
</HTML>
```

在 with 语句块内，全部省略了 document 对象，显示的结果如图 3-21 所示。

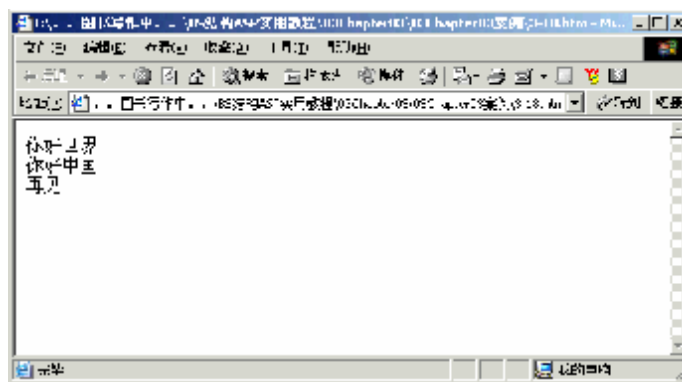


图 3-21 使用 with

3.10 JavaScript 内置对象

内置对象都有自己的方法和属性，访问属性的语法是：“对象名.属性名称”。访问方法的语法是：“对象名.方法名称(参数表)”。所谓的方法就是一个普通的函数被封装到一个对象中。

3.10.1 时间对象

时间对象是 JavaScript 的内置对象，使用前必须先申明：“var curr=new Date();”，注意 Date() 的首字母必须大写。使用时间对象的语法如程序 3-19.htm 所示。

案例名称：使用时间对象

程序名称：3-19.htm

```
<HTML>
<HEAD>
```

```
<TITLE></TITLE>

</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    var curr=new Date();
    document.write("今天是");
    switch(curr.getDay()){
      case 0:document.write("周日,休息了!");break;
      case 1:document.write("周一,需要工作!");break;
      case 2:document.write("周二,需要工作!");break;
      case 3:document.write("周三,需要工作!");break;
      case 4:document.write("周四,需要工作!");break;
      case 5:document.write("周五,需要工作!");break;
      case 6:document.write("周六,休息了!");break;
    }
  </SCRIPT>
</BODY>
</HTML>
```

getDay()是对象 Date 的一个方法。getDay()方法得到今天是星期几，如果返回 0 代表星期日，结果如图 3-22 所示。



图 3-22 使用时间对象

Date 对象提供了两类方法：从系统中获得当前的时间和日期；设置当前时间和日期。表 3-2 列出了常用的方法。

表 3-2 Date 对象中的方法

名 称	描 述
getDate	获得当前的日期
getDay	获得当前的星期
getHours	获得当前的小时
getMinutes	获得当前的分钟
getMonth	获得当前的月份
getSeconds	获得当前的秒
getTime	获得当前的时间（毫秒为单位）
getYear	获得当前的年份

案例 3-2: 网页时钟

利用 Date 对象可以方便的做一个网页时钟, 如程序 time.htm 所示。

案例名称: 网页时钟

程序名称: time.htm

```
<HTML>
<HEAD>
    <SCRIPT LANGUAGE="JavaScript">
        var strTime, strDate;
        function webClock()
        {
            var dNow = new Date();
            var dHours = dNow.getHours();
            var dMinutes = dNow.getMinutes();
            var dSeconds = dNow.getSeconds();
            strTime = dHours;
            strTime += ((dMinutes<10) ? ":0" : ":") + dMinutes;
            strTime += ((dSeconds<10) ? ":0" : ":") + dSeconds;
            clock.time.value = strTime;

            var dDate = dNow.getDate();
            var dMonth = dNow.getMonth() + 1;
            var dYear = dNow.getFullYear();
            strDate = dMonth;
            strDate += ((dDate<10) ? "/0" : "/") + dDate;
            strDate += "/" + dYear;
            clock.date.value = strDate;
            setTimeout("webClock()",1000);
        }
    </SCRIPT>
    <TITLE></TITLE>
</HEAD>
<BODY ONLOAD="webClock()">
    <FORM NAME="clock">
        时间: <INPUT TYPE="TEXT" NAME="time" SIZE="10"><BR>
        日期: <INPUT TYPE="TEXT" NAME="date" SIZE="10"><BR>
    </FORM>
</BODY>
</HTML>
```

程序首先定义了一个时间对象 dNow, 然后利用 Date 对象的方法 getHours()、getMinutes() 和 getSeconds() 分别得到当前时间的小时, 分钟和秒。

语句 “((dMinutes<10) ? "0" : "") + dMinutes;” 的功能是: 如果 dMinutes 小于 10, 就在前面加上一个 “0”。目的是为了显示时总是保持两位的分钟数, 如果 dMinutes 为 “8”, 则显示 “08”。

语句 “setTimeout("webClock()",1000);” 的含义是: 1000 毫秒以后, 再调用 webClock () 函数一次, 因为该语句是在 webClock 函数内部, 所以 webClock 就会被一直调用下去。

“clock.date.value = strDate;” 和 “clock.time.value = strTime;” 是给文本框赋值, clock 是 form 的名字, date 和 time 分别是两个文本框的名字, 分别把程序变量 strDate 和 strTime 的值写到文本框中去, 程序显示的结果如图 3-23 所示。

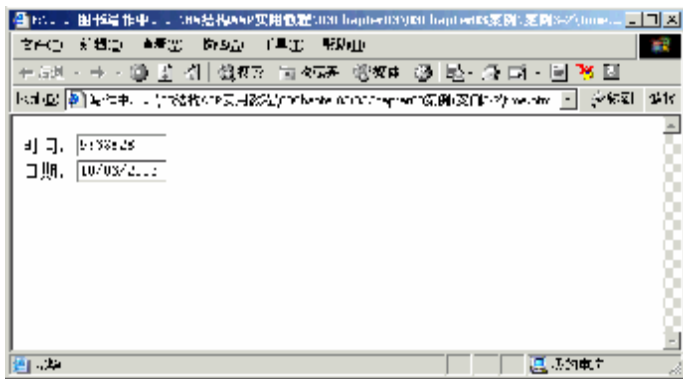


图 3-23 网页时钟

3.10.2 Math 对象

Math 对象可以用来处理各种数学运算。Math 对象的内置方法定义了各种数学运算、可以直接调用的 Math 对象的方法, 如: “Math.数学函数(参数)”。如程序 3-20 所示的取正函数的值。

案例名称: 使用 Math 对象

程序名称: time.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      var a = Math.sin(1);
      document.write(a)
    </SCRIPT>
  </BODY>
```

</HTML>

这样弧度为 1 的角度的 sin 值就求出来了，显示的结果如图 3-24 所示。

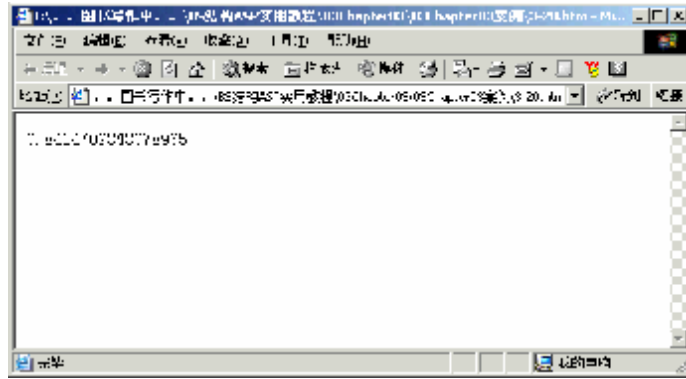


图 3-24 使用 Math 对象

3.10.3 String 对象

一般利用 String 对象提供的函数来处理字符串。String 对字符串的处理主要提供了下列方法。

- (1) charAt(idx): 第一个字符位置是“0”，返回指定位置处的字符。
- (2) indexOf(chr): 返回指定子字符串的位置，从左到右，找不到返回-1。
- (3) lastIndexOf(chr): 返回指定子字符串的位置，从右到左，找不到返回-1。
- (4) toLowerCase(): 将字符串中的字符全部转化成小写。
- (5) toUpperCase(): 将字符串中的字符全部转化成大写。

使用 String 方法如程序 3-21.htm 所示。

案例名称：使用字符串处理函数

程序名称：3-21.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <script language="JavaScript">
      var str = "I am a boy, I like programming!";
      a = str.charAt(7);
      b = str.indexOf("a");
      c = str.lastIndexOf("a");
      d = str.toUpperCase();
      document.write(a + "<br>");
```



```

        document.write(b + "<br>")
        document.write(c + "<br>")
        document.write(d + "<br>")
    </script>
</BODY>
</HTML>

```

显示的结果如图 3-25 所示。

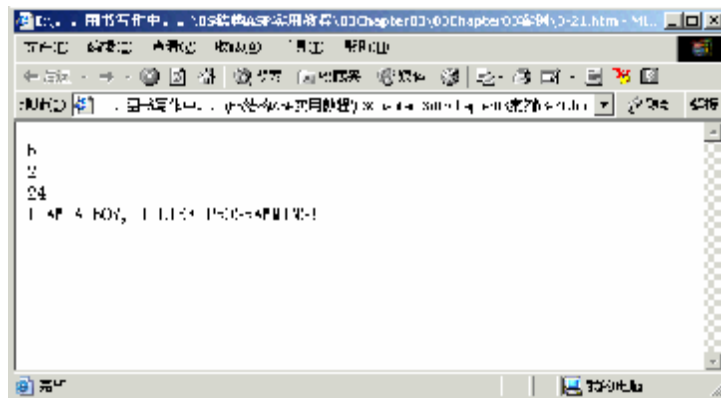


图 3-25 使用 string 对象

案例 3-3：字符串扫描统计

程序 string.htm 实现的功能是，统计字符串中字符“a”出现的次数，然后将次数输出到浏览器上。

案例名称：使用字符串处理函数

程序名称：string.htm

```

<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      var str = "I am a girl, I like cat"
      var iCount = 0;
      for( i = 0; i < str.length; i++)
      {
        if(str.charAt(i) == "a")
        {
          iCount++;
        }
      }
    </SCRIPT>
  </BODY>
</HTML>

```

```
}  
document.write(iCount);  
</SCRIPT>  
</BODY>  
</HTML>
```

程序执行的结果如图 3-26 所示。

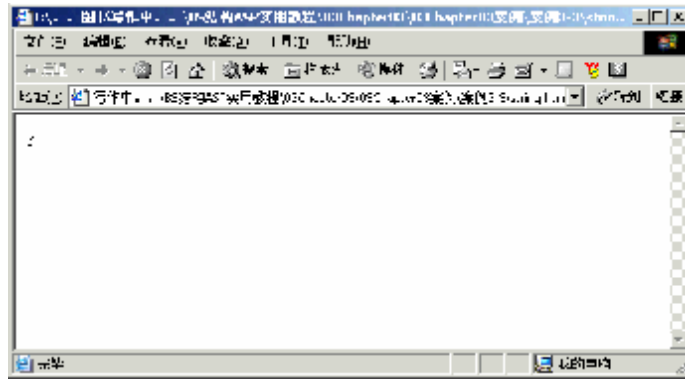


图 3-26 字符统计

3.11 JavaScript 的常用函数

介绍几个常用的函数：eval()函数，parseInt()函数和 parseFloat 函数。

3.11.1 eval()函数

eval()函数接受一个字符串形式的表达式，并执行该表达式。可以这样理解：eval()函数的参数是一个字符串变量，经过 eval()函数处理后就变成一条语句了。使用方法如程序 3-22.htm 所示。

案例名称：使用 eval()函数

程序名称：3-22.htm

```
<HTML>  
  <HEAD>  
    <TITLE></TITLE>  
  </HEAD>  
  <BODY>  
    <SCRIPT LANGUAGE="JavaScript">  
      var str = "1+2+3";  
      document.write(eval(str));  
    </SCRIPT>
```

```
</BODY>
```

```
</HTML>
```

可以看出 `eval()` 函数参数是一个字符串 “1+2+3”，经过 `eval()` 函数处理以后，变成一个算术表达式。显示的结果如图 3-27 所示。

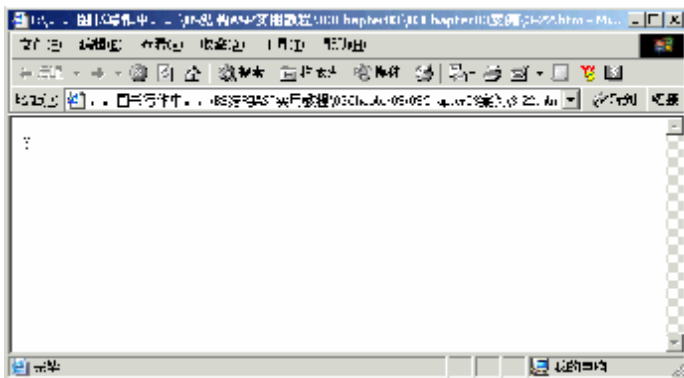


图 3-27 使用 eval 函数

3.11.2 parseInt()函数和 parseFloat()函数

`parseInt()` 函数功能是从一个字符串中提出一个整数，如果遇到字符串中除了数字以外的字符，`parseInt()` 就停止转换，返回已有的结果。如果第一个字符不是数字，`parseInt()` 就返回 “NaN”。`parseFloat()` 函数和 `parseInt()` 函数相似，区别 `parseFloat()` 可以提取小数。这两个函数的使用方法如程序 3-23.htm 所示。

案例名称：使用 `parseInt` 函数和 `parseFloat` 函数

程序名称：3-23.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      var a = parseInt("123China");
      var b = parseFloat("123.12China");
      document.write(a + "<br>");
      document.write(b);
    </SCRIPT>
  </BODY>
</HTML>
```

`parseInt()` 函数的功能是从参数的左边找数字，直到找到的不是数字为止，`parseFloat()` 函数

是从参数左边找小数，就是包括小数点，直到找到的不是数字为止。显示的结果如图 3-28 所示。

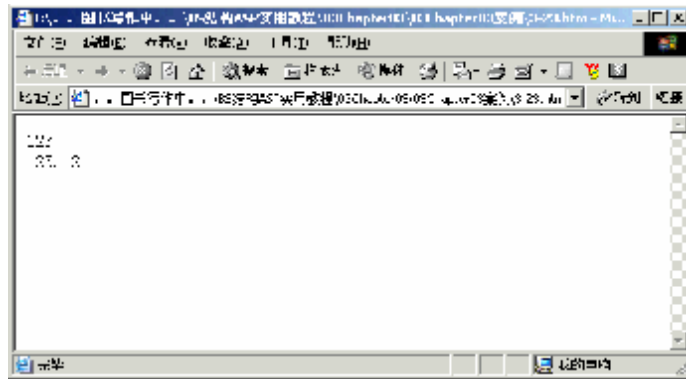


图 3-28 使用 parseInt 函数

3.12 对象层次及 DOM 模型

浏览器对象有一定的层次，也就是一定的从属关系。在从属关系中浏览器对象 window 反映的是一个完整的浏览器窗口，它是其他大部分对象的祖先。window 对象包括 location、history 和 document 等。document 对象之下还有下一级的对象，包括：forms、links 及 anchors，等等，层次结构如图 3-29 所示。

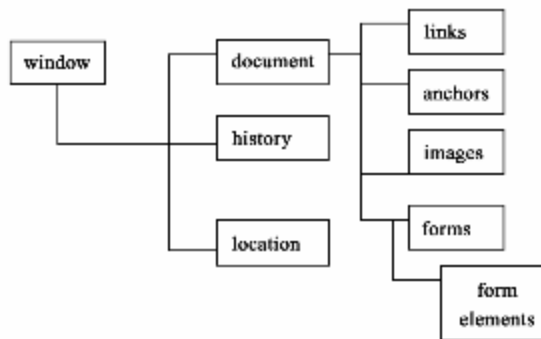


图 3-29 对象的层次

注意：所有的对象是浏览器的一些对象，与 JavaScript 语言本身并没有太大的关系。在其他的脚本语言中调用的方法是一样的。

在使用 IE 浏览器载入一个网页时，浏览器会根据网页的内容产生一些相应的对象提供给 JavaScript 使用，这就是浏览器对象的一部分。另外，根据浏览器本身的配置和属性，还有一些其他的对象可以提供给 JavaScript 程序使用。浏览器对象就是网页和浏览器本身各种实体元素在 JavaScript 程序中的体现。这样的浏览器对象主要包括如下几类。

(1) window 对象：处于整个从属表的最顶级位置。每一个这样的对象代表一个浏览器窗口。

- (2) document 对象：含有当前网页的各种特性，例如标题、背景及使用的语言等。
- (3) location 对象：含有当前网页的 URL 地址。
- (4) history 对象：含有以前访问过的网页的 URL 地址。
- (5) forms 对象：是从属于浏览器对象 document 的一个数组，为处理表单及其中的界面对象提供属性和方法，每一个表单是这个数组中的一个单独元素。
- (6) anchors 对象：是从属于浏览器对象 document 的一个数组，为处理锚提供属性和方法，每一个锚就是这个数组中的一个元素。
- (7) links 对象：是从属于浏览器对象 document 的一个数组，为处理超级连接提供属性和方法，每一个超级连接就是这个数组中的一个元素。

DOM (Document Object Model) 是文档对象模型的缩写，文档对象模型提供了文档的定立模型。当读取 Form 表单中的某一个元素的时候，就要用到 DOM 来定位了。定义的方法如程序 3-24.htm 所示。

案例名称：DOM 定位模型

程序名称：3-24.htm

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
      function do_Copy()
      {
        var str = frm1.txtBox1.value;
        frm2.txtBox2.value= str;
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <FORM NAME="frm1">
      <INPUT TYPE="TEXT" NAME="txtBox1" >
      <INPUT TYPE="BUTTON" VALUE="复制" ONCLICK="do_Copy()">
    </FORM>
    <FORM NAME="frm2">
      <INPUT TYPE="TEXT" NAME="txtBox2">
    </FORM>
  </BODY>
</HTML>
```

在上面的文本框中输入字符串后，单击按钮字符串就会自动复制到第二个文本框中，显示的结果如图 3-30 所示。

Form 表单中所有的元素都可以采用这样的定位技术来得到它们的值。语句“frm1.txtBox1.value”中，frm1 是 Form 的名字，txtBox1 是文本框的名字。

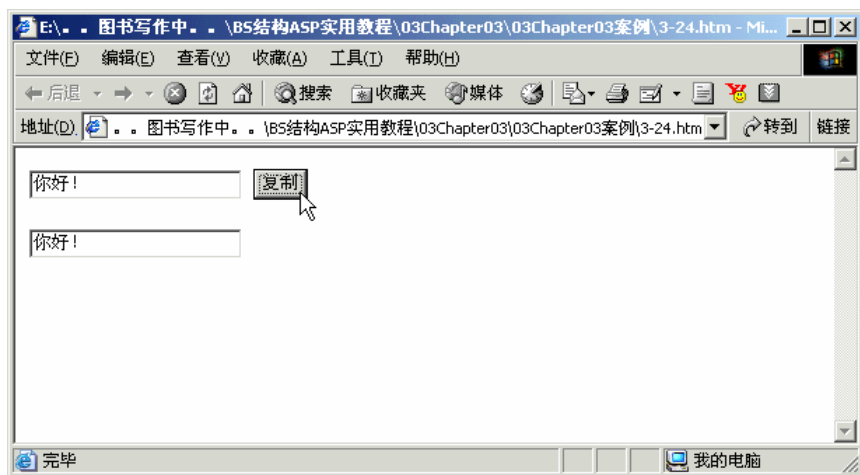


图 3-30 使用 DOM

3.12.1 window 对象

window 对象是浏览器对象中大部分对象的祖先,下面列出一些常用 Window 对象的方法。

(1) open(URL,windowName,parameterList): open()方法创建一个新的浏览器窗口,并在新窗口中载入一个指定的 URL 地址。

(2) close(): close()方法关闭一个浏览器窗口。

(3) alert(): 弹出一个消息框。

(4) confirm(): 弹出一个确认框。

(5) prompt(): 弹出一个提示框。

(6) setTimeout(expression, time): 定时设置,在一定的时间后自动执行 expression 的代码,使用 time 设置时间,单位是毫秒。

(7) clearTimeout(timer): 取消利用 setTimeout 的定时设置。

(8) setInterval(expression, time): 设定一个时间间隔,可以定时反复的自动执行 expression 描述的代码,使用 time 设置时间,单位是毫秒。

使用 window 对象来打开一个新窗口,如程序 3-25.htm 所示。

案例名称: 使用 window 对象

程序名称: 3-25.htm

```
<HTML>
<HEAD>
  <TITLE></TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    function new_win( )
    {
      window.open( "new.htm", "my", "toolbar=no, left=150, top=200, menubar=no,
width=150,height=150" );
    }
  </SCRIPT>
</HEAD>
```

```

    }
  </SCRIPT>
</HEAD>
  <BODY onload="new_win()">
  </BODY>
</HTML>

```

当打开该页面的时候，就会自动打开一个小窗口，这种技术目前十分常用，一般在小窗口显示广告。Open()方法有三个参数：第一个参数是“new.htm”，设置要打开的网页；第二个参数是打开窗口在操作系统中的名称，这里任何名称都可以，只是兼容以前的版本，程序中用不到；第三个参数设置打开浏览器的样式。该方法在页面加载的时候调用，执行结果如图 3-31 所示。

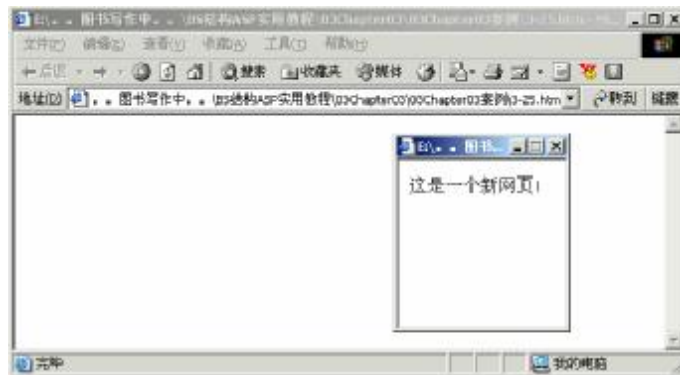


图 3-31 使用 window 对象

location 是 window 对象的一个属性。可以设置当前网页的地址。可以使用 location 属性让浏览器打开某页，具体的语法如下：

```
window.location="页面地址"
```

location 属性的使用方法如程序 3-26.htm 所示。

案例名称：使用 location 属性

程序名称：3-26.htm

```

<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
  function test_location()
  {
    window.location="new.htm";
  }
</SCRIPT>
</HEAD>
  <BODY>

```

```
<FORM NAME="frm">
  <INPUT TYPE="BUTTON" VALUE="超级链接" ONCLICK="test_location()">
</FORM>
</BODY>
</HTML>
```

当单击按钮时,调用函数 test_location()使当前浏览器转到 new.htm 文件。利用按钮实现了超级链接的功能,如图 3-32 所示。

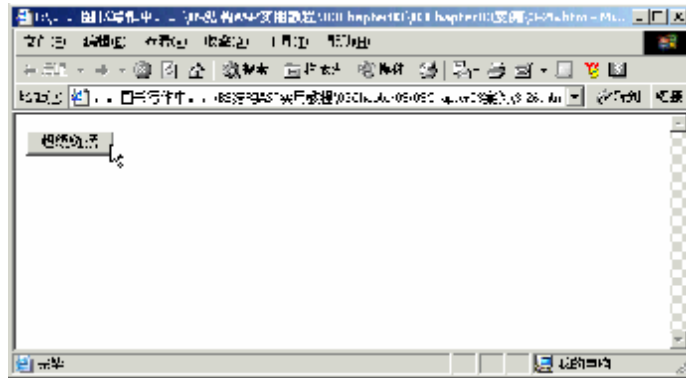


图 3-32 Location 属性

3.12.2 history 对象

history 对象含有以前访问过的网页的 URL 地址。可以利用 history 对象实现网页的前进和后退,如程序 3-27.htm 所示。

案例名称: 使用 history 对象

程序名称: 3-27.htm

```
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="frm">
      <INPUT TYPE="BUTTON" VALUE="后退" ONCLICK="goback()">
      <INPUT TYPE="BUTTON" VALUE="前进" ONCLICK="goforward()">
    </FORM>
    <SCRIPT LANGUAGE="JavaScript">
      function goforward()
      {
        history.go(1);
      }
    </SCRIPT>
  </BODY>
</HTML>
```



```

    }
    function goback()
    {
        history.go(-1)
    }
</SCRIPT>
</BODY>
</HTML>

```

要实现前进和后退的功能，必须在浏览器中可以前进和后退，首先打开浏览器其他的 HTML 文件，然后在浏览器中修改打开文件为 3-27.htm。这时候发现浏览器上后退按钮是激活的，如图 3-33 所示。单击页面上的“后退”按钮，就可退到上一页面。

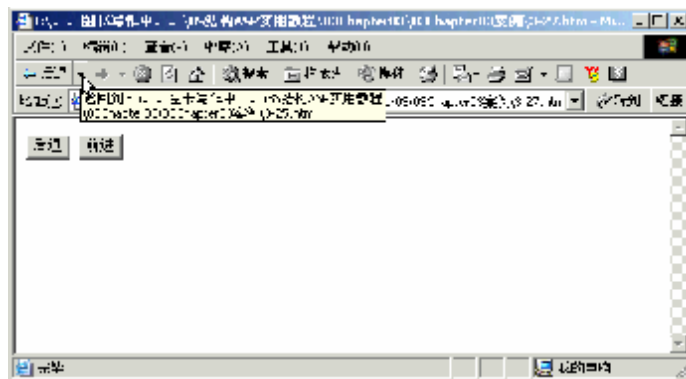


图 3-33 History 对象

案例 3-4：网页计算器

网页计算器是一种常见的应用，可以利用现在所学的知识，实现一个简易的计算器。如程序 cal.htm 所示。

案例名称：网页计算器

程序名称：cal.htm

```

<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
    <CENTER>
      <FORM NAME="myform">
        <input type="button" value="1" name="b1" onClick="add1()">
        <input type="button" value="2" name="b2" onClick="add2()">
        <input type="button" value="3" name="b3" onClick="add3()">

```

```
<input type="button" value="+" name="b_add" onClick="add_add()">
<br><br>
<input type="button" value="4" name="b4" onClick="add4()">
<input type="button" value="5" name="b5" onClick="add5()">
<input type="button" value="6" name="b6" onClick="add6()">
<input type="button" value="-" name="b_minus" onClick="add_minus()">
<br><br>
<input type="button" value="7" name="b7" onClick="add7()">
<input type="button" value="8" name="b8" onClick="add8()">
<input type="button" value="9" name="b9" onClick="add9()">
<input type="button" value="*" name="b_mul" onClick="add_mul()">
<br><br>
<input type="button" value="0" name="b0" onClick="add0()">
<input type="button" value="." name="b_pot" onClick="add_pot()">
<input type="button" value="=" name="b_value" onClick="calc()">
<input type="button" value="/" name="b_divide" onClick="add_divide()">
<br><br>
<input type="button" value="清空" name="b_clear" onClick="clear_all()">
<hr></hr>
<input type="text" name="sour_txt" value="">
<br><br>
<input type="text" name="dest_txt" value="运算结果">
</form></CENTER>
<Script Language="JavaScript">
    //msg: 运算字符串
    var msg = "";
    function clear_all()
    {
        msg = "";
        document.myform.sour_txt.value = "";
        document.myform.dest_txt.value = "";
    }
    function add1()
    {
        msg += document.myform.b1.value;
        document.myform.sour_txt.value = msg;
    }
    function add2()
    {
        msg += document.myform.b2.value;
```

```
document.myform.sour_txt.value = msg;
}
function add3()
{
    msg += document.myform.b3.value;
    document.myform.sour_txt.value = msg;
}
function add4()
{
    msg += document.myform.b4.value;
    document.myform.sour_txt.value = msg;
}
function add5()
{
    msg += document.myform.b5.value;
    document.myform.sour_txt.value = msg;
}
function add6()
{
    msg += document.myform.b6.value;
    document.myform.sour_txt.value = msg;
}
function add7()
{
    msg += document.myform.b7.value;
    document.myform.sour_txt.value = msg;
}
function add8()
{
    msg += document.myform.b8.value;
    document.myform.sour_txt.value = msg;
}
function add9()
{
    msg += document.myform.b9.value;
    document.myform.sour_txt.value = msg;
}
function add0()
{
    msg += document.myform.b0.value;
```

```
document.myform.sour_txt.value = msg;
}
function add_add()
{
    msg += document.myform.b_add.value;
    document.myform.sour_txt.value = msg;
}
function add_minus()
{
    msg += document.myform.b_minus.value;
    document.myform.sour_txt.value = msg;
}
function add_mul()
{
    msg += document.myform.b_mul.value;
    document.myform.sour_txt.value = msg;
}
function add_divide()
{
    msg += document.myform.b_divide.value;
    document.myform.sour_txt.value = msg;
}
function add_pot()
{
    msg += document.myform.b_pot.value;
    document.myform.sour_txt.value = msg;
}
function calc()
{
    document.myform.dest_txt.value = eval(msg);
}
</SCRIPT>
</BODY>
</HTML>
```

程序将操作存储在 msg 变量中，最后利用 eval() 函数将结果计算出来。程序执行结果如图 i-34 所示。



图 3-34 网页计算器

案例 3-5：表单的数据合法性验证

JavaScript 的主要应用之一是验证 Form 表单输入数据的合法性。表单可以有两种基本的验证方式，一种是在服务器端利用 JSP/ASP 来验证，另一种是利用 JavaScript 进行验证。通常都是采用客户端的 JavaScript 验证，如一个录入系统的表单。如图 3-35 所示。



图 3-35 输入表单

需要验证学号、姓名等文本框不能为空，而且学号必须是 13 位。利用 JavaScript 和 DOM 技术可以编写出验证程序。如程序 Verify.htm 所示。

案例名称：表单验证

程序名称：Verify.htm

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function mycheck(){
    if (document.myform.xuehao.value=="") {
        alert("请输入学号!");
        document.myform.xuehao.focus();
    }
}
```

```

        return false;
    }
    if (document.myform.xuehao.value.length!=13) {
        alert("标准学号为 13 位, 您目前输入的学号为" +
document.myform.xuehao.value.length + "位");
        document.myform.xuehao.focus();
        return false;
    }
    if (document.myform.xingming.value=="") {
        alert("请输入姓名!");
        document.myform.xingming.focus();
        return false;
    }
    if (document.myform.pingyin.value=="") {
        alert("请输入姓名的拼音!");
        document.myform.pingyin.focus();
        return false;
    }
    if (document.myform.zhengjian.value=="") {
        alert("请输入证件号码!");
        document.myform.zhengjian.focus();
        return false;
    }
}
</SCRIPT>
<BODY>
<FORM METHOD="POST" ACTION="do_submit.htm" onSubmit="return mycheck()"
name="myform">
<font face="华文行楷" size="6">学员档案录入系统</font></p>
<table border="0" cellpadding="0" cellspacing="0" width="631">
<tr>
<td width="69" height="31"> </td>
<td width="148" height="31">
<span style="font-family: 宋体; font-size: 10.5pt">学号</span></td>
<td width="414" height="31">
<input type="text" name="xuehao" size="20">
<font color="#FF0000">** 13 位学员学号</font></td>
</tr>
<tr>
<td width="69" height="31"> </td>

```

```
<td width="148" height="31">姓名</td>
<td width="414" height="31">
  <input type="text" name="xingming" size="20">
  <font color="#FF0000">** 3-4 个汉字中文汉字</font></td>
</tr>
<tr>
  <td width="69" height="31"> </td>
  <td width="148" height="31"> 姓名拼音</td>
  <td width="414" height="31">
    <input type="text" name="pingyin" size="20">
    <font color="#FF0000">** 限 20 个英文字母</font></td>
</tr>
<tr>
  <td width="69" height="31"> </td>
  <td width="148" height="31"> 性别</td>
  <td width="414" height="31">
    <input type="radio" value="男" checked name="xingbie">男
    <input type="radio" name="xingbie" value="男">女
    <font color="#FF0000">**</font></td>
</tr>
<tr>
  <td width="69" height="31"> </td>
  <td width="148" height="31"> 身份证号/军官证</td>
  <td width="414" height="31"><font color="#FF0000">
    <input type="text" name="zhengjian" size="20"> ** 身份证编号</font></td>
</tr>
<tr>
  <td width="69" height="31"> </td>
  <td width="148" height="31">文化程度</td>
  <td width="414" height="31">
    <select size="1" name="wenhua">
      <option value="高中以下">高中以下</option>
      <option selected value="高中或中专">高中或中专</option>
      <option value="大专或高职">大专或高职</option>
      <option value="本科">本科</option>
      <option value="本科以上">本科以上</option>
    </select><font color="#FF0000">**</font></td>
</tr>
<tr>
  <td width="69" height="31"> </td>
```

```
<td width="148" height="31"> 专业</td>
<td width="414" height="31">
    <select size="1" name="zhuanye">
<option selected value="计算机及相关专业">计算机及相关专业</option>
<option value="其他">其他</option>
    </select><font color="#FF0000">*</font></td>
</tr>
<tr>
    <td width="69" height="31"> </td>
    <td width="148" height="31"> 毕业学校</td>
    <td width="414" height="31">
        <input type="text" name="xuexiao" size="20"></td>
</tr>
<tr>
    <td width="69" height="31"> </td>
    <td width="148" height="31"> 工作状态</td>
    <td width="414" height="31">
        <select size="1" name="gongzuozhuangtai">
<option value="在读学生">在读学生</option>
<option value="在职" selected>在职</option>
<option value="暂无职业">暂无职业</option>
        </select><font color="#FF0000">*</font></td>
</tr>
<tr>
    <td width="69" height="31"> </td>
    <td width="148" height="31"> 工作单位</td>
    <td width="414" height="31">
        <input type="text" name="gongzuodanwei" size="20"></td>
</tr>
<tr>
    <td width="69" height="31"> </td>
    <td width="148" height="31">工作种类</td>
    <td width="414" height="31">
        <select size="1" name="gongzuozhonglei">
<option value="计算机技术人员" selected>计算机技术人员</option>
<option value="其它">其它</option>
        </select><font color="#FF0000">*</font></td>
</tr>
<tr>
    <td width="69" height="31"> </td>
```



```
<td width="148" height="31"> 通信地址</td>
<td width="414" height="31">
  <input type="text" name="dizhi" size="20"></td>
</tr>
<tr>
  <td width="69" height="25"> </td>
  <td width="148" height="25"> 邮编</td>
  <td width="414" height="25">
    <input type="text" name="youbian" size="20"></td>
</tr>
<tr>
  <td width="69" height="32"> </td>
  <td width="148" height="32">学员状态</td>
  <td width="414" height="32">
    <select size="1" name="xueyuanzhuangtai">
      <option value="在读" selected>在读</option>
      <option value="但未获认证">但未获认证</option>
      <option value="已获认证">已获认证</option>
      <option value="退学">退学</option>
      <option value="休学">休学</option>
      <option value="已毕业">已毕业</option>
    </select><font color="#FF0000">*</font></td>
</tr>
<tr>
  <td width="69" height="32"> </td>
  <td width="148" height="32">入学时间</td>
  <td width="414" height="32">
    <select size="1" name="ruxuenian">
      <option selected value="2004">2004</option>
      <option value="2005">2005</option>
      <option value="2006">2006</option>
    </select>年<select size="1" name="ruxueyue">
      <option value="1" selected>1</option>
      <option value="2">2</option>
      <option value="3">3</option>
      <option value="4">4</option>
      <option value="5">5</option>
      <option value="6">6</option>
      <option value="7">7</option>
      <option value="8">8</option>
```

```

<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
</select>月<select size="1" name="ruxueri">
<option value="1" selected>1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
</select>日<font color="#FF0000">**</font></td>
</tr>
<tr>
<td width="69" height="32"> </td>

```

```
<td width="148" height="32"> 备注</td>
<td width="414" height="32">
    <textarea rows="7" name="beizhu" cols="41"></textarea></td>
</tr>
<tr>
    <td width="69" height="32"> </td>
    <td width="148" height="32"> </td>
    <td width="414" height="32"><font color="#FF0000">注意：带**号的选项为必填
字段</font></td>
</tr>
</table>
<p align="center"><input type="submit" value="提交" name="B1">
<input type="reset" value="重置" name="B2"></p>
</form>
</body>
</html>
```

当单击“提交”按钮后，程序首先调用“return mycheck()”语句。该语句的意思是如果 mycheck() 函数返回 true，则提交表单，如果函数返回 false，则不提交表单。当学号不填写任何字符的时候，条件“if (document.myform.xuehao.value=="")”为 true，显示提示框，返回 false，表单就不提交。当用户输入学号为 7 位时，条件“document.myform.xuehao.value.length != 13”成立，显示提示框，如图 3-36 所示。



图 3-36 出错提示框

对其他 Form 元素也可以采用这种方法。目前客户端的验证都采用这种方法。

小结

本章是编程基础，需要重点掌握的是分支和循环流程控制语句、如何定义和调用一个函数、JavaScript 内置对象和常用函数的使用方法。重点理解本章的四个案例：动态按钮、网页时钟、字符串统计和网页计算器。

课后习题和上机练习

1. 编写程序统计 1 到 50 中所有偶数的和。（分别用 for 和 while 语句实现）

2. 编写程序实现：取系统时间，如果时间在 6:00—12:00 之间，输出“早上好”；如果时间在 12:00—18:00，输出“下午好”；如果时间在 18:00—24:00 之间，输出“晚上好”；如果时间在 0:00—6:00，输出“凌晨好”。

3. 在字符串“I am a girl, I like dancing!”的每个字符之间加上一个字符“#”，输出字符为：“I# #a#m# #a# #girl#,# #I# #like# #d#a#n#c#i#n#g#!”，并统计“#”的个数。

4. 改写案例 3-4，尽量将函数缩减到最小。（上机完成）

第二部分 ASP 内置对象与组件

第 4 章 ASP 内置对象

本章要点

本章主要介绍 ASP 内置的五大常用对象、一个集合和一个文件。五大对象分别是：Response, Request, Application, Session 和 Server, 一个集合是 Cookie, 一个文件是 global.asa。

4.1 内置对象概述

为了实现网站的常见功能, ASP 提供了内置对象, 内置对象的特点是: 不需要先创建一个实例, 可以直接使用。常用的内置对象及其功能如下。

- (1) Response 对象: 将信息发送回给浏览器。
- (2) Request 对象: 获取客户端的信息。
- (3) Application 对象: 存储一个应用中所有用户共享的信息。
- (4) Session 对象: 存储一个普通用户其滞留期间的用户信息。
- (5) Server 对象: 提供许多服务器端的应用函数。

4.2 Response 对象

Response 对象的主要功能是向浏览器输出信息。

4.2.1 输出数据

Response.Write()的功能是向浏览器输出信息, 与 JavaScript 中的 document.write()的功能相斥。但是必须了解其区别: Response 是 ASP 的对象, 输出的方式是从服务器端向客户端的浏览器输出。如程序 4-01.asp 所示。

案例名称: 输出数据

程序名称: 4-01.asp

```
<%@ Language=Jscript %>
<%
    var str = "你好, 中国! ";
    Response.Write(str);
%>
```

程序首先将该文件放入网站的主目录下，然后在浏览器中输入“http://localhost/4-01.asp”，这样就将字符串的内容输出到浏览器上了。显示结果如图 4-1 所示。

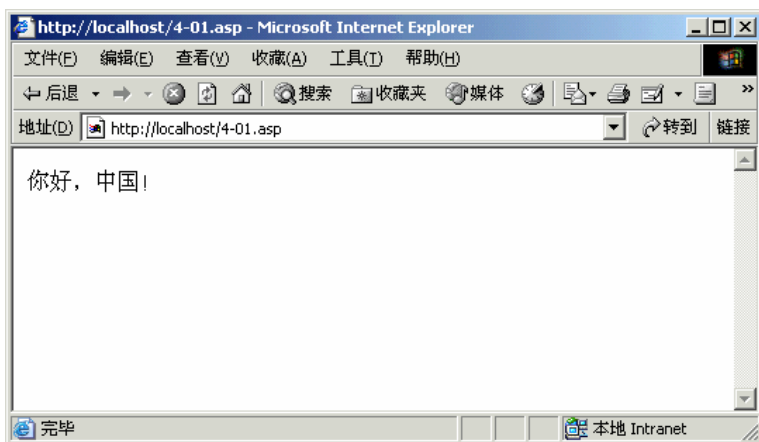


图 4-1 程序显示结果

所有<%...%>内的代码都被 IIS 解释执行，IIS 解释执行后把程序的结果发送给浏览器。<Script>内的代码都是由浏览器解释执行的，如程序 4-02.asp 所示。

案例名称：理解服务器端执行

程序名称：4-02.asp

```
<%@ Language=Jscript %>
<%
    var dnow = new Date();
    dhours = dnow.getHours();
    dminutes = dnow.getMinutes();
    dseconds = dnow.getSeconds();
    Response.Write("服务器时间: " + dhours + ":" + dminutes + ":" + dseconds);
%>
<SCRIPT LANGUAGE="JavaScript">
    var dnow = new Date();
    dhours = dnow.getHours();
    dminutes = dnow.getMinutes();
    dseconds = dnow.getSeconds();
    document.write("<br>浏览器时间: " + dhours + ":" + dminutes + ":" + dseconds);
</SCRIPT>
```

该程序的代码基本一样，都是取时间，但是<%...%>内的代码是 ASP 程序，取的是服务器端时间，程序执行的结果如图 4-2 所示。

这两个程序的执行结果是一样的，因为浏览器和服务器在同一台计算机上，如果用另一台计算机的 IE 通过 IP 地址链接该页面，取的就是不同计算机的系统时间。右击浏览器查看源代码如图 4-3 所示。

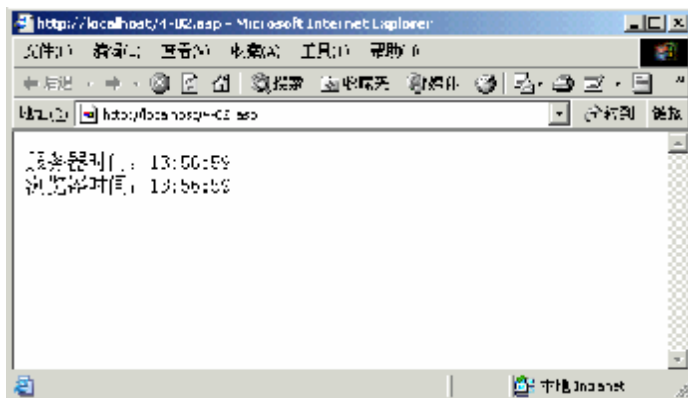


图 4-2 理解服务器端执行

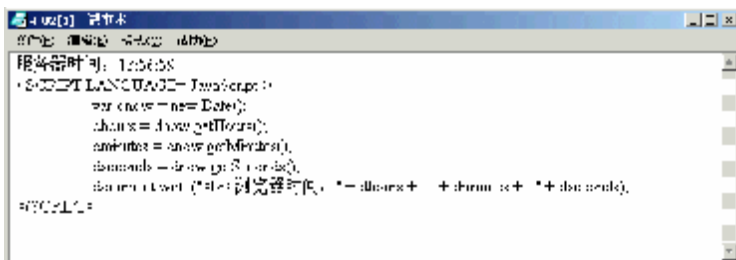


图 4-3 查看源代码

可以看出<%...%>内的程序只有结果，而<Script>标记可以看到程序。因为<%...%>内的程序是由服务器 IIS 执行的，而<Script>中的程序是客户端浏览器解释执行的。

因为 `Response.Write` 使用非常频繁，可将它简化为“`=`”。如程序 4-03.asp 所示。

案例名称：简写形式

程序名称: 4-03.asp

```
<%@ Language=Jscript %>
```

<%="简写形式"%>

简写的时候需要注意：如果`<%`和`Response.Write`之间还有其他语句，就不能简写。也就是说，`<%`和`Response.Write`连着的时候，才可以简写成“`=`”，显示的结果如图4-4所示。

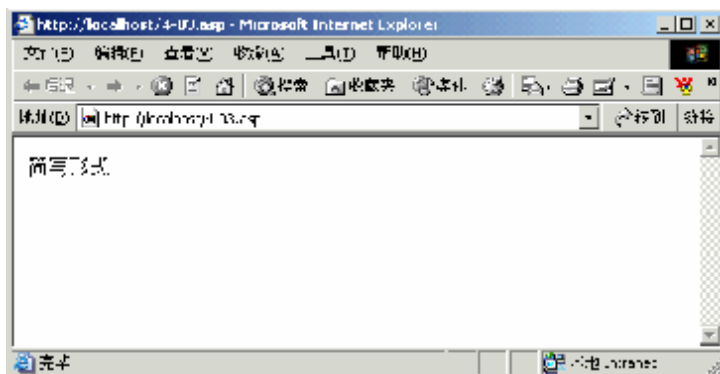


图 4-4 简写形式

可以利用 Response.Write 输出 HTML 标记，如程序 4-04.asp 所示。

案例名称：输出 HTML 标记

程序名称：4-04.asp

```
<%@ Language=Jscript %>
<%
    Response.Write("<html>");
    Response.Write("<body>");
    Response.Write("输出图片<hr>");
    Response.Write("<IMG SRC='myimage.jpg' WIDTH='300' HEIGHT='200'>");
    Response.Write("</body>");
    Response.Write("</html>");
%>
```

Response.Write 输出的是一个标准的 HTML 文件的内容，在 Body 标记中引入一张图片。这里需要注意的是必须将 HTML 标记中的双引号改成单引号。当前的文件夹下必须有一张名为 myimage.jpg 的图片，程序执行的结果如图 4-5 所示。

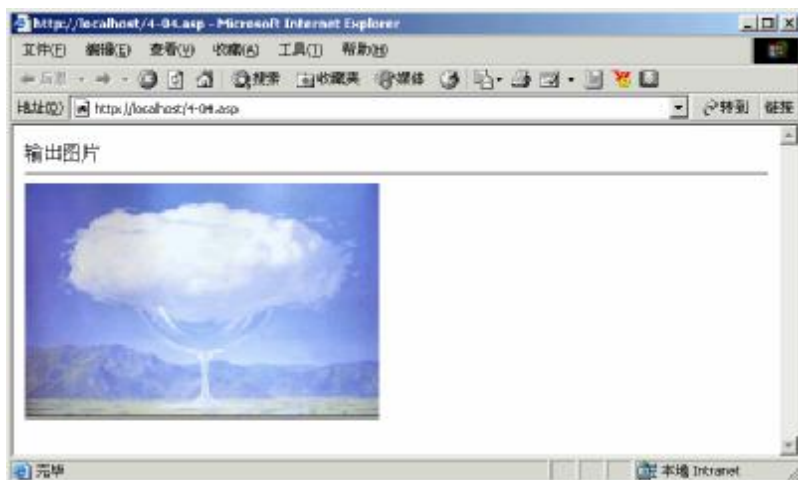


图 4-5 输出 HTML 标记

1.2.2 网页转向

访问一个 ASP 页面，有时并不是出现需要的网页，而是转到另外一个页面上去了。Response 中有一个 Redirect 方法的功能能实现转向，如程序 4-05.asp 所示。

案例名称：网页转向

程序名称：4-05.asp

```
<%@ Language=Jscript %>
<%
```



```
Response.Redirect("4-02.asp");
```

```
%>
```

执行这个 ASP 程序, 就会自动转到并执行 4-02.asp 文件。

1.2.3 停止输出

在程序的执行过程中, 如果遇到了 `Response.End()` 语句, 下面所有的 `Response.Write` 就不再被执行了, 如程序 4-06.asp 所示。

案例名称: 停止输出

程序名称: 4-05.asp

```
<%@ Language=Jscript %>
<%
for(var i = 0; i < 100; i++)
{
    Response.Write(i + "<br>");
    if(i == 5)
    {
        Response.End();
    }
}
%>
```

当 `i` 循环到 5 的时候, 条件成立, 执行 “`Response.End();`” 语句, 所以只输出 0~5。执行结果如图 4-6 所示。



图 4-6 停止输出

1.3 Request 对象

`Request` 对象主要的功能是从客户端得到数据, 常用的三种取得数据的方法是:

Request.Form, Request.QueryString 和直接使用 Request。Request 是前两种方法的缩写。前两种方法主要对应的是 Form 提交时的两种不同提交方法：Post 方法和 Get 方法。

4.3.1 获得表单数据

首先准备一个 HTML 表单，这里提供两个输入框，一个输入用户名，一个输入密码。如程序 4-06.htm 所示。

案例名称：HTML 表单

程序名称：4-06.htm

```
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <FORM ACTION="4-07.asp" METHOD="POST">
      <P>姓名: <INPUT TYPE="TEXT" SIZE="20" NAME="UserID"></P>
      <P>密码: <INPUT TYPE="PASSWORD" SIZE="20" NAME="UserPWD"></P>
      <P><INPUT TYPE="SUBMIT" VALUE="提交"> </P>
    </FORM>
  </BODY>
```

Form 表单 Action 属性是文件 4-07.asp，意思是当用户提交时，用 4-07.asp 来处理提交的数据。METHOD 属性说明提交的方式，这里设置为 Post 方式，需要使用 Request.Form 来读取。如程序 4-07.asp 所示。

案例名称：读取表单数据

程序名称：4-07.asp

```
<%@ Language=Jscript %>
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    用户名: <%=Request.Form("UserID")%><br>
    密码为: <%=Request.Form("UserPWD")%>
  </BODY>
</HTML>
```

UserID 和 UserPWD 分别是程序 4-06.htm 中文本框的名字。首先执行 4-06.htm 文件（放在网站中打开），在文本框中输入用户名和密码，如图 4-7 所示。

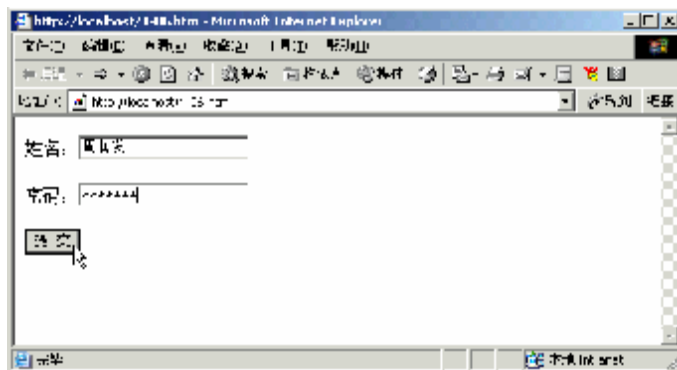


图 4-7 在表单中输入内容

输入完毕，单击“提交”按钮，调用 4-07.asp 来处理提交的内容。程序将文本框的内容读取出来，再输出到浏览器上，如图 4-8 所示。

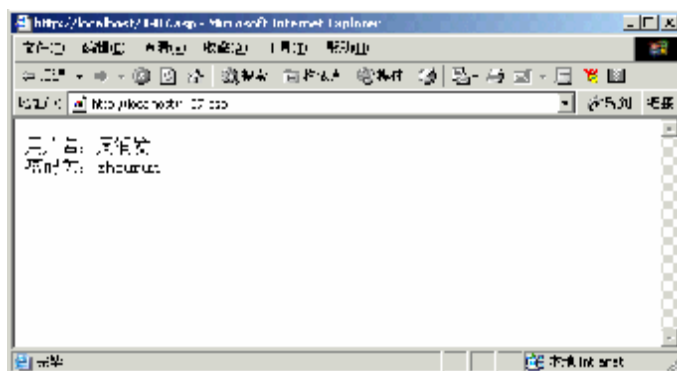


图 4-8 显示读取的数据

修改 4-06.htm 文件中 Form 表单的 METHOD 属性为 Get，如程序 4-08.htm 所示。

案例名称：HTML 表单

程序名称：4-08.htm

```
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <FORM ACTION="4-09.asp" METHOD="Get">
      <P>姓名: <INPUT TYPE="TEXT" SIZE="20" NAME="UserID"></P>
      <P>密码: <INPUT TYPE="PASSWORD" SIZE="20" NAME="UserPWD"></P>
      <P><INPUT TYPE="SUBMIT" VALUE="提交"> </P>
    </FORM>
  </BODY>
```

要读取 Get 方法提交的数据必须采用 Request.QueryString，如程序 4-09.asp 所示。

案例名称：读取表单数据

程序名称：4-09.asp

```
<%@ Language=Jscript %>
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    用户名: <%=Request.QueryString("UserID")%><br>
    密码为: <%=Request.QueryString("UserPWD")%>
  </BODY>
</HTML>
```

首先在程序 4-08.htm 的表单输入用户名和密码, 如图 4-9 所示。

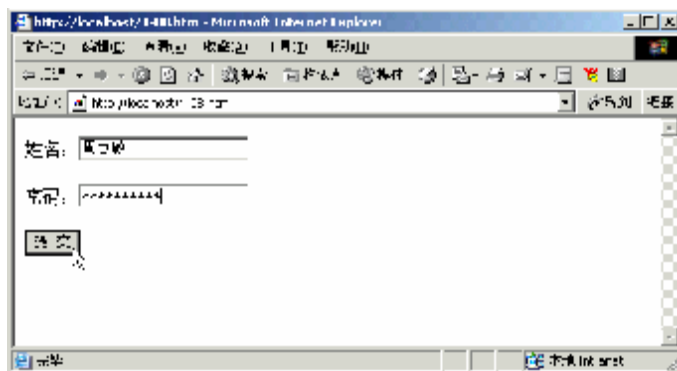


图 4-9 输入数据

当表单提交后, 可以看到 Get 方式发送的数据在浏览器地址栏上显示, 如图 4-10 所示。

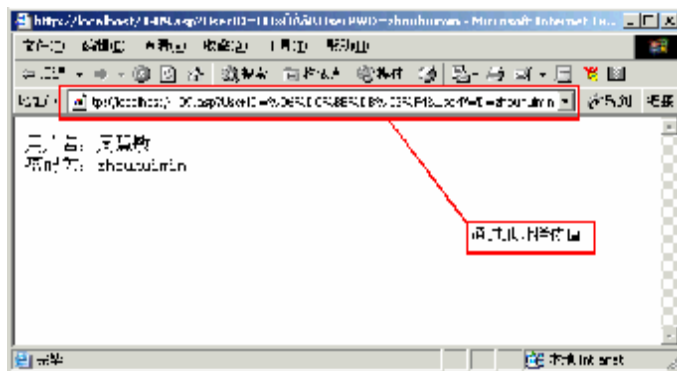


图 4-10 读取数据

提交数据和读取数据的对应关系如表 4-1 所示。

表 4-1 提交数据和读取数据的对应关系

提交方式	读取方式
Method = Post	Request.Form
Method = Get	Request.QueryString

通过 Get 方式传递的数据直接显示在地址栏上, 仿照这样的格式在超级链接后面加上数

居，Request 对象是否可以读取出来呢？答案是可以的，如程序 4-10.asp 所示。

案例名称：读取超级链接后面的参数

程序名称：4-10.asp

```
<%@ Language=Jscript %>
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <A HREF="4-09.asp?UserID=王二小&UserPWD=wex">传递参数</A></P>
  </BODY>
</HTML>
```

鼠标移动到超级链接上，在状态栏上显示出详细信息。如图 4-11 所示。

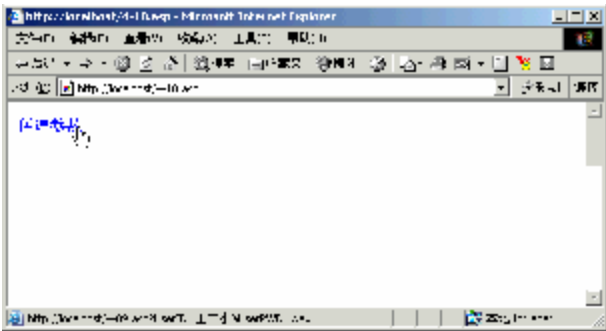


图 4-11 超级链接传递参数

单击超级链接，调用 4-09.asp 文件处理信息，将用户名和密码显示出来，如图 4-12 所示。

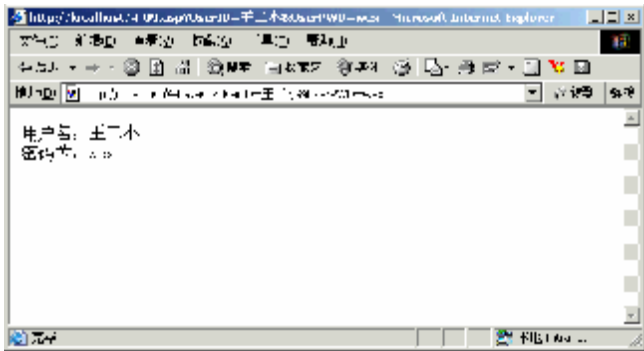


图 4-12 读取超级链接后面的参数

可以将两种方法简化为一种方法，就是不管是 Post 方法还是 Get 方法，一律用 Request(“元素名”)来接收。

案例 4-1：调查表

该案例实现的是一个含有多种 Form 元素的输入表单，利用 Request 对象读取到服务器端，

用 Response 对象输出到浏览器上。输入程序如 input.htm 所示。

案例名称：调查表输入页面

程序名称：input.htm

```
<HTML>

  <HEAD>

    <TITLE></TITLE>

  </HEAD>

<BODY>

  <FORM ACTION="handle.asp" METHOD="POST">

    姓名： <INPUT TYPE="TEXT" NAME="USERNAME"><BR>

    密码： <INPUT TYPE="PASSWORD" NAME="USERPWD"><BR>

    性别：

    <INPUT TYPE="RADIO" NAME="SEX" VALUE="男">男

    <INPUT TYPE="RADIO" NAME="SEX" VALUE="女">女 <BR>

    血型：

    <INPUT TYPE="RADIO" NAME="BLOOD" VALUE="O">O

    <INPUT TYPE="RADIO" NAME="BLOOD" VALUE="A">A

    <INPUT TYPE="RADIO" NAME="BLOOD" VALUE="B">B

    <INPUT TYPE="RADIO" NAME="BLOOD" VALUE="AB">AB <BR>

    性格：

    <INPUT TYPE="CHECKBOX" Name="CHATACTER" VALUE="热情大方">

    热情大方

    <INPUT TYPE="CHECKBOX" Name="CHATACTER" VALUE="温柔体贴">

    温柔体贴

    <INPUT TYPE="CHECKBOX" Name="CHATACTER" VALUE="多情善感">

    多情善感<BR>

    简介：

    <TEXTAREA ROWS="8" COLS="30" NAME="MEMO"></TEXTAREA><BR>

    城市： <SELECT SIZE="1" NAME="CITY">

    <OPTION VALUE="北京">北京市</OPTION>

    <OPTION VALUE="上海">上海市</OPTION>

    <OPTION VALUE="南京">南京市</OPTION>

    </SELECT><BR>

    <INPUT TYPE="SUBMIT" VALUE="提交">

    <INPUT TYPE="RESET" VALUE="RESET">

  </FORM>

</BODY>

</HTML>
```

复选框（Checkbox）、单选框（Radio）和下拉列表（Select）比较特殊，利用 Request 对

象读取的是其 Value 属性的值，一般将 Value 值设置和其显示的值相同。在调查表中输入一些数据，如图 4-13 所示。

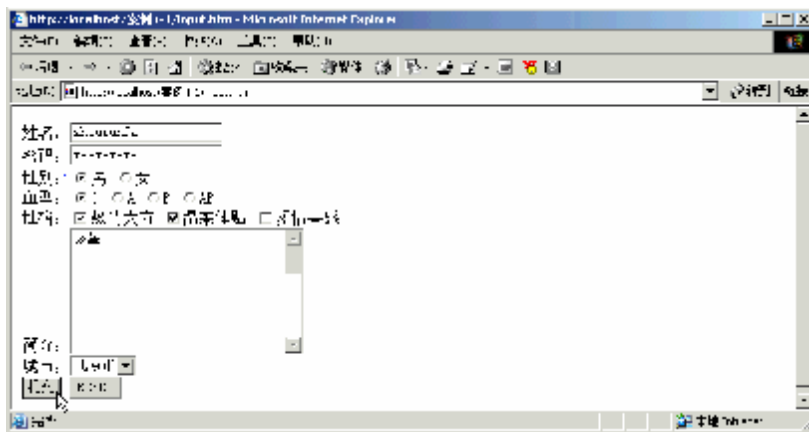


图 4-13 调查表输入界面

利用 Request 对象可以将所有数据都读出来，注意和输入表单元素的 Name 属性相同。如程序 handle.asp 所示。

案例名称：调查表处理页面

程序名称：handle.asp

```
<%@ Language=Jscript %>
<%
    var strUserName = Request("USERNAME");
    var strUserPWD = Request("USERPWD");
    var strUserSex = Request("SEX");
    var strUserBlood = Request("BLOOD");
    var strUserChar = Request("CHATACTER");
    var strUserMemo = Request("MEMO");
    var strUserCity = Request("CITY");
%>
```

用户名是: <%=strUserName%>

用户密码: <%=strUserPWD%>

你的性别: <%=strUserSex%>

你的血型: <%=strUserBlood%>

你的性格: <%=strUserChar%>

你的简介: <%=strUserMemo%>

所在城市: <%=strUserCity%>

复选框的值以逗号分隔读取出来，如图 4-14 所示。

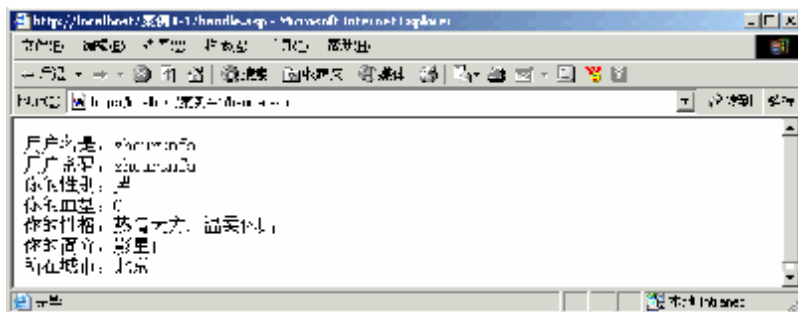


图 4-14 读取表单内容

1.3.2 获得服务器信息

可以通过 Request 对象的 ServerVariables 方法得到一些服务器端的信息, 比如当前 ASP 的文件名、客户端的 IP 地址等。如程序 4-11.asp 所示。

案例名称: 获得服务器的信息

程序名称: 4-11.asp

```
<%@ Language=Jscript %>
```

PATH_INFO 返回:

```
<%=Request.ServerVariables("PATH_INFO")%><br>
```

REMOTE_ADDR 返回:

```
<%=Request.ServerVariables("REMOTE_ADDR")%><br>
```

SERVER_NAME 返回:

```
<%=Request.ServerVariables("SERVER_NAME")%><br>
```

利用该程序可以将所有链接过本网站用户的 IP 地址记录下来, 如图 4-15 所示。

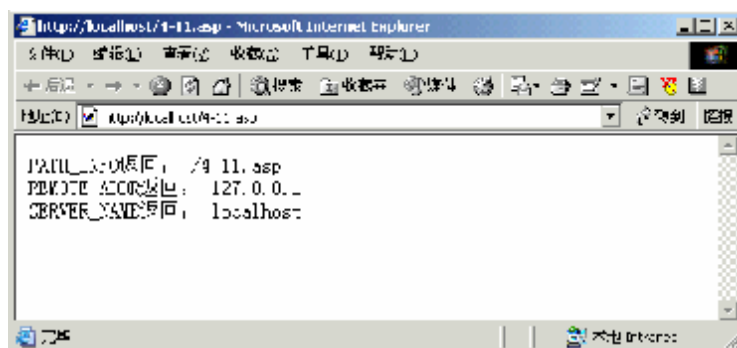


图 4-15 服务器信息

1.4 Application 对象

Application 对象是一个比较重要的对象, 对 Application 对象的理解关键是: 网站所有的

用户公用一个 Application 对象，当网站服务器开启的时候，Application 就被创建。利用 Application 这一特性，可以方便地创建聊天室和网站计数器等常用站点应用程序。

4.4.1 自定义属性

Application 对象没有自己的属性，用户可以根据自己的需要定义属性，来保存一些信息，其基本语法是：Application（“自定义属性名”），如程序 4-12.asp 所示。

案例名称：Application 属性的自定义属性

程序名称：4-12.asp

```
<%@ Language=Jscript %>
<%
Application("Greeting")="你好！"
%>
<%=Application("Greeting")%>
```

首先对自定义属性 Application("Greeting")赋值，然后程序将其输出。程序执行的结果如图 4-16 所示。

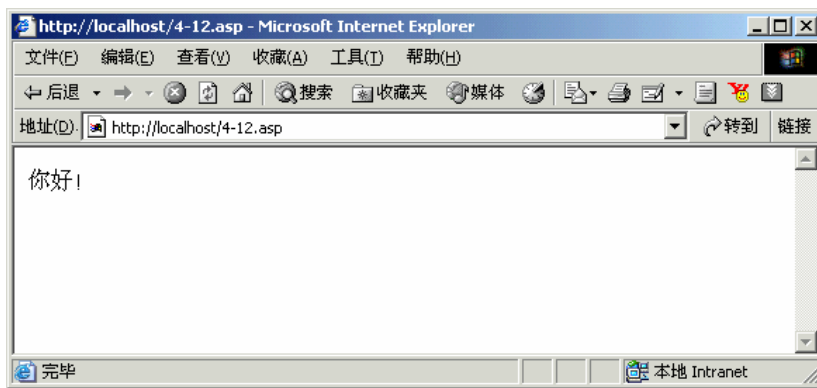


图 4-16 自定义属性

执行完以后，该对象就被保存在服务器上。执行程序 4-13.asp 时依然可以输出原先保存的值。

案例名称：Application 属性的自定义属性

程序名称：4-13.asp

```
<%@ Language=Jscript %>
<%=Application("Greeting")%>
```

虽然在该程序没有赋值，但是依然可以输出，因为 4-12.asp 文件已经给 Application("Greeting")赋值，如图 4-17 所示。

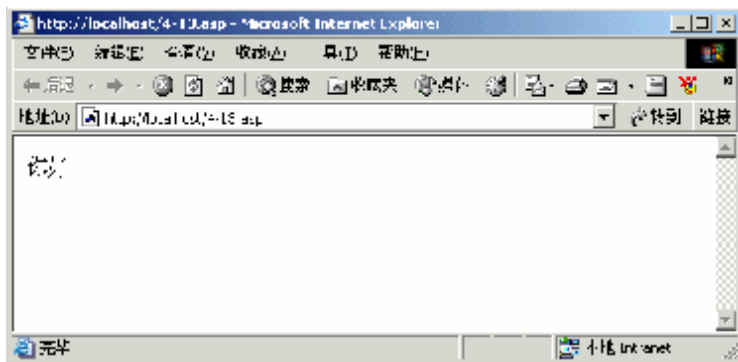


图 4-17 读取 Application 属性的值

Application 变量不会因为某一个甚至全部用户离开就消失，一旦建立 Application 变量，那么它就一直存在到网站关闭或者这个 Application 对象被卸载，这经常可能是几周或者几个月。

1.4.2 实现聊天室

聊天室允许多用户实时进行信息交流，所有用户可以看到彼此的信息，这与 Application 对象的特点正好符合，所以可以方便地利用 Application 实现聊天室。如程序 4-14.asp 所示。

案例名称：简易聊天室

程序名称：4-14.asp

```
<%@ Language=Jscript %>
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <%
      var mywords = Request("mywords");
      Application("chat") = Application("chat") + "<br>" + mywords;
      Response.Write(Application("chat"));
    %>
    <FORM ACTION="4-14.asp" METHOD="get">
      <INPUT TYPE="TEXT" SIZE="30" NAME="mywords" VALUE="I LIKE CHAT">
      <INPUT TYPE="SUBMIT" name="submit" VALUE="提交">
    </FORM>
  </BODY>
</HTML>
```

这时就可邀请一个朋友进入聊天室，虽然比较简易，不过已经实现了聊天室的功能。执行的结果如图 4-18 所示。

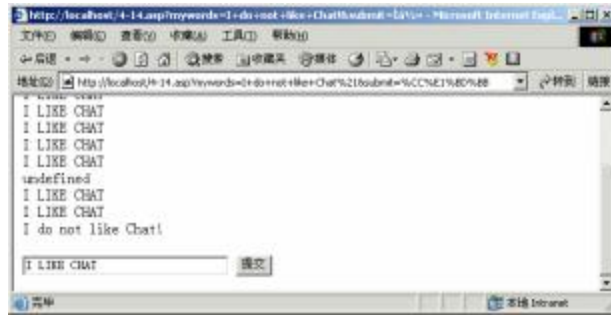


图 4-18 简易聊天室

案例 4-2：网页计数器

网页计数器是 Application 对象的又一个用途，因为 Application 是所有的用户所共有的，所以可以存储计数器的值，当有新用户访问网页时自动增加计数器的值。如程序 CountV1.asp 所示。

案例名称：网页计数器版本一

程序名称：CountV1.asp

```
<%@ Language=Jscript %>
<%
    if(Application("Counter")!=null)
        Application("Counter") = parseInt(Application("Counter")) + 1;
    else
        Application("Counter") = 1;
%>
<HTML>
<BODY>
    <P ALIGN="CENTER">您是本站点第<%=Application("Counter")%>位贵宾! </P>
</BODY>
</HTML>
```

以上程序只有两条 ASP 语句，但是已经可以实现计数了。这个计数器还不完善，下面分成几个版本来完善它，在版本二中加入 Application 的锁定语句。如程序 CountV2.asp 所示。

案例名称：网页计数器版本二

程序名称：CountV2.asp

```
<%@ Language=Jscript %>
<%
    Application.Lock();
    if(Application("Counter")!=null)
        Application("Counter") = parseInt(Application("Counter")) + 1;
    else
        Application("Counter") = 1;
```

```

        Application.Unlock();

%>
<HTML>
<BODY>
    <P ALIGN="CENTER">您是本站点第<%=Application("Counter")%>位贵宾! </P>
</BODY>
</HTML>

```

这时在版本一的基础上增加了 `Application_Lock` 和 `Application_Unlock` 两条语句, 这两条语句是防止两个用户几乎同时浏览页面, 当前一个正在修改 `Application` 的值, 第二个人也想修改时, 这样就会出现一个用户没有统计, 所以当第一个人开始修改时, 用 `Application_Lock` 语句先锁定, 当操作完毕时, 再解除锁定。执行结果如图 4-19 所示。



图 4-19 计数器显示的结果

一般网站的计数器都是图形界面, 这个计数器也可以变成具有图形界面的计数器。如程序 `CountV3.asp` 所示。

案例名称: 网页计数器版本三

程序名称: `CountV3.asp`

```

<%@ Language=Jscript %>
<%
    Application.Lock();
    if(Application("Counter")!=null)
        Application("Counter") = parseInt(Application("Counter")) + 1;
    else
        Application("Counter") = 1;
Application.Unlock();
function G( counter )
{
    var S, i, myimage;
    myimage = "";
    S = counter + "";
    for(i = 0; i<S.length; i++)

```

```

    {
        myimage = myimage + "<IMG SRC=" + S.charAt(i) + ".gif>";
    }
    return myimage;
}
%>
<HTML>
    <HEAD>
    </HEAD>
    <BODY>
        <P ALIGN="CENTER">您是本站第 <%=G(Application("Counter"))%> 位贵宾! </P>
    </BODY>
</HTML>

```

程序原理和前面的两个版本是一样的，只是将数字做了一些转化，函数 G 的功能就是实现这个转化。如何转化的呢？首先取出 Application("Counter") 的值，然后赋值给变量 S，再执行循环语句，S.length 功能是取字符串的长度，S.charAt(i) 的意思是从字符串 S 的第 i 个位置开始取 1 个字符。执行完后就将原先的字符数字转化成以图形显示的图形计数器。本程序执行需要有 0~9 的十个 Gif 图片，运行的结果如图 4-20 所示。

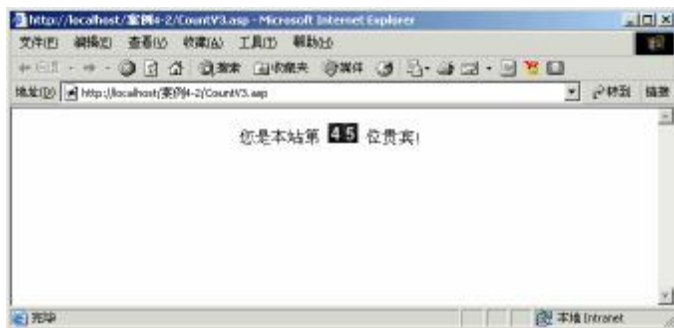


图 4-20 基于图形界面的计数器

4.5 Session 对象

Session 指的是访问者从到达某个特定主页到离开为止的那段时间网站为用户分配的用来保存用户信息的对象。

可以使用 Session 对象存储用户登录网站时候的信息。当用户在页面之间跳转时，存储在 Session 对象中的变量不会被清除。

4.5.1 对 Session 的理解

当用户登录网站的时候，系统会自动分配给用户一个 Session。可以使用 SessionID 属性得到该 Session 的 ID，如程序 4-15.asp 所示。

案例名称: 使用 SessionID 属性

程序名称: 4-15.asp

```
<%@ Language=Jscript %>
<HTML>
  <HEAD>
  </HEAD>
<BODY>
  你的 SessionID: <%=Session.SessionID%>
</BODY>
</HTML>
```

这个 ID 是惟一的, 用来标识每一个用户, 结果如图 4-21 所示。

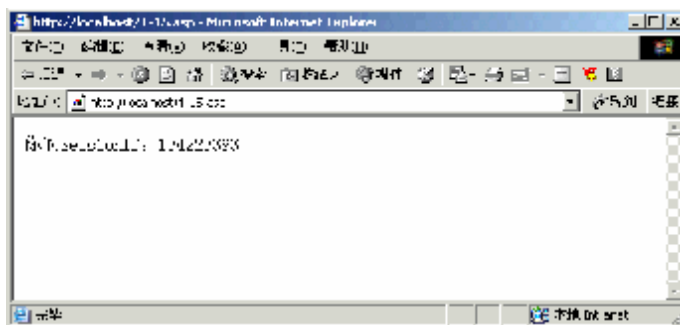


图 4-21 得到 SessionID 的值

4.5.2 自定义属性

Session 对象的主要用途也是保存信息, 当用户第一次到达网站时, 系统为其分配一个 session。Session 和 Application 一样也使用自己的自定义属性, 如程序 4-16.asp 所示。

案例名称: 使用 Session 的自定义属性

程序名称: 4-16.asp

```
<%@ Language=Jscript %>
<HTML>
  <HEAD></HEAD>
<BODY>
  <%
    Session("Greeting")="欢迎!";
    Response.Write(Session("Greeting"));
  %>
  <br>
  <a href="4-17.asp">下一页</a>
```

```
</BODY>
```

```
</HTML>
```

首先给自定义属性赋值，然后将该属性值读取并显示出来，如图 4-22 所示。

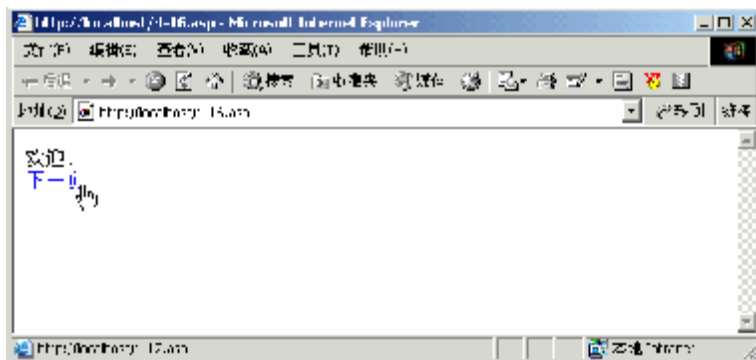


图 4-22 Session 的自定义属性

单击超级链接进入 4-17.asp 页面。

案例名称：使用 Session 的自定义属性

程序名称：4-17.asp

```
<%@ Language=Jscript %>
<HTML>
    <HEAD></HEAD>
<BODY>
    <%
        Response.Write(Session("Greeting"));
    %>
</BODY>
</HTML>
```

在该页面中没有对 Session 赋值，同样也可以得到该 Session 的值，如图 4-23 所示。注意这里是通过 4-16.asp 文件的超级链接打开的该文件。如果打开一个新的浏览器直接执行 4-17.asp 文件，就取不到值了。而 Application 的自定义属性则可以。

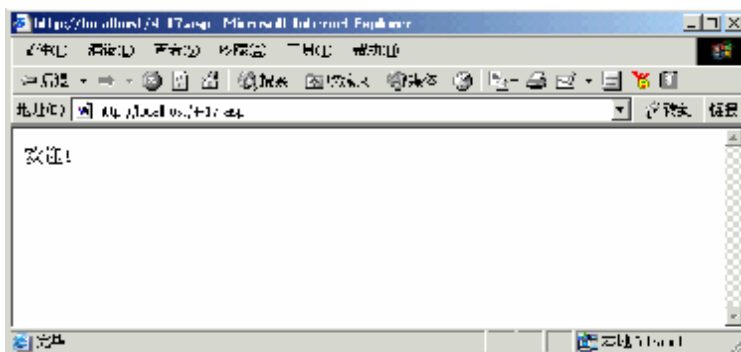


图 4-23 使用 Session 自定义属性

1.6 Server 对象

利用 Server 对象可以方便地访问服务器上的方法和属性，最常用的是利用 `Server.CreateObject` 创建组件的实例。

1.6.1 输出 HTML 代码

通常情况下，浏览器将“<”和“>”中间的符号作为系统标记，不会显示在浏览器上，如果想在浏览器上显示时，可以使用 Server 对象的 `HTMLEncode` 方法，如程序 4-18.asp 所示。

案例名称：输出 HTML 代码

程序名称：4-18.asp

```
<%@ Language=Jscript %>  
<%= Server.HTMLEncode( "HTML 段落标记:<P></P>" )%>
```

加上 `HTMLEncode` 方法以后，就可以将 HTML 标记显示在浏览器上。程序执行的结果如图 4-24 所示。

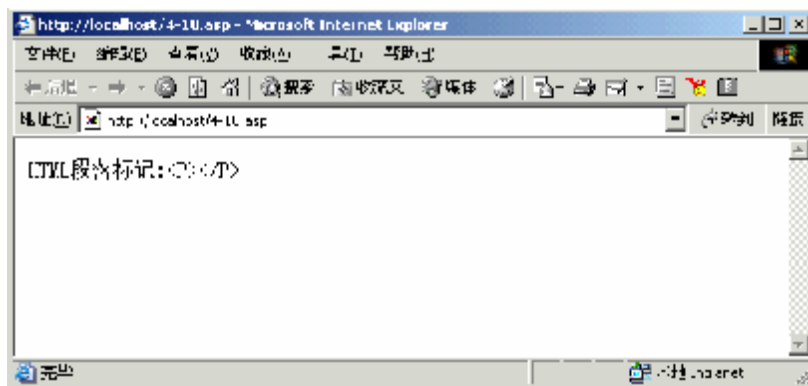


图 4-24 输出 HTML 代码

1.6.2 获取物理路径

`Server.MapPath` 的功能是把“网络路径”转换为服务器上的物理路径。如程序 4-19.asp 所示。

案例名称：获取物理路径

程序名称：4-19.asp

```
<%@ Language=Jscript %>  
Server.MapPath("/") 传回 <%=Server.MapPath("/")%><P>  
Server.MapPath("/file1.txt")传回 <%=Server.MapPath("/file1.txt")%><P>
```



```
Server.MapPath("/fololer") 传回 <%=Server.MapPath("/folder")%><P>
```

```
Server.MapPath("file2.txt") 传回 <%=Server.MapPath("file2.txt")%><P>
```

file1.txt 文件和 folder 文件夹是不存在的, 这里只是将这些文件名作为字符串来考虑, 到周用时才会去检查它到底存不存在。Server.MapPath("/")获得的永远是网站的主目录, server.MapPath("/file1.txt")获取物理路径时, 网站主目录加上字符串“file1.txt”, 如果不加“/”, 则得到的是当前 ASP 文件所在目录, 程序执行结果如图 4-25 所示。



图 4-25 获取物理路径

4.7 Cookie 集合

Cookie 和 Session 一样都可以保存用户信息, 区别是 Cookie 将信息保存在客户端, 而 session 将信息保存在服务器。

4.7.1 写入 Cookie

可以将 Cookie 写到浏览器中, 让浏览器来保存 Cookies 的值。如程序 4-20.asp 所示。

案例名称: 写入 Cookie

程序名称: 4-20.asp

```
<%@ Language=Jscript %>
<%
    Response.Cookies("User")("Name1")="Jack"
    Response.Cookies("User")("Password")="password"
%>
```

写入 Cookies

查看

程序将字符串“Jack”和“Password”写入 Cookie 集合, 程序执行的结果如图 4-26 所示。

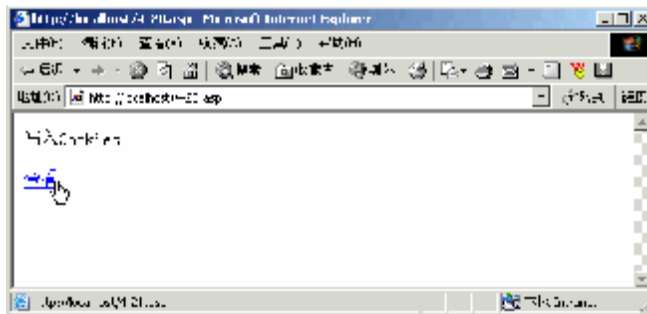


图 4-26 写入 Cookies

1.7.2 读取 Cookie

程序 4-20.asp 是将 Cookie 写入浏览器,也就是将 Cookie 保存起来,程序 4-21.asp 将 Cookie 读出来,并且显示到浏览器上。

案例名称: 读出 Cookie

程序名称: 4-21.asp

```
<%@ Language=Jscript %>  
读出 Cookies<br>  
<%=Request.Cookies("User") %><br>  
<%=Request.Cookies("User")("Name1")%><br>  
<%=Request.Cookies("User")("Password")%><br>
```

该程序不能直接执行,否则不会输出 Cookie,必须通过 4-20.asp 文件的超级链接执行该文件。显示结果如图 4-27 所示。

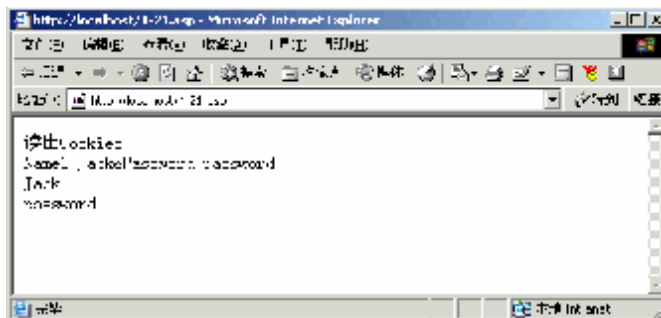


图 4-27 读取 Cookies

1.8 global.asa 文件

Application 的两个事件 (Application_OnStart 和 Application_OnEnd) 和 Session 的两个事件 (Session_OnStart 和 Session_OnEnd) 都放在 global.asa 文件中。global.asa 文件必须位于网站的根目录才能起作用。global.asa 文件有如下结构。

案例名称: global.asa 文件的结构

程序名称: global.asa

```
<SCRIPT LANGUAGE="JavaScript" RUNAT="SERVER">
function Application_OnStart()
{
}
function Session_OnStart()
{
}
function Session_OnEnd()
{
}
</SCRIPT>
```

注意: global.asa 使用<SCRIPT>标记语法来限制脚本, 必须用<SCRIPT>标记来引用而不能使用<%和%>符号引用, 在 global.asa 中不能有任何输出语句, 无论是 HTML 的语法还是 Response.Write()方法都是不行的。

一个 Application 的 OnStart 事件肯定是在 Session_OnStart 事件之前。Application 不会像 Session 那样在一个新用户登录后就触发, Application 只触发一次, 即是第一个用户登录网站时。

案例 4-3: 实现动态在线人数统计

当网站被第一个人访问时就会自动调用 Global.asa 文件, 但是需要注意的是: global.asa 文件必须放在网站的根目录下。程序 CountOnLine.asp 文件只用了一条语句实现在线人数统计。

案例名称: 在线人数统计

程序名称: CountOnLine.asp

```
<%@ Language=Jscript %>
<%Response.Write("现在有" + Application("whoson")+"人在线")%>
```

具体实现统计功能的是 global.asa 文件。

案例名称: 在线人数统计的 global.asa 文件

程序名称: CountOnLine.asp

```
<SCRIPT LANGUAGE="JavaScript" RUNAT="SERVER">
function Application_OnStart()
{
    Application.Lock();
    Application("whosOn") = 0;
    Application.Unlock();
}
```

```
function Session_OnStart()  
{  
    Application.Lock();  
    Application("whoson") = parseInt(Application("whoson")) + 1;  
    Application.Unlock();  
}  
function Session_OnEnd()  
{  
    Application.Lock();  
    Application("whoson") = parseInt(Application("whoson")) - 1;  
    Application.Unlock();  
}  
</SCRIPT>
```

当网站开启时就自动调用 Global.asa 文件, 此时其中的 Application_OnStart 首先被调用, 然后执行其中的语句, Application("whoson")被自动清零。然后当第一个用户登录网站时, Session_OnStart 被调用, 此时 Application("whoson")被自动加一。当第二位、第三位用户登录时, 又被加一, 所以显示的是在线的人数。当有用户退出时, Session_OnEnd 事件被自动调用, 将 Application("whoson")的值减一, 所以显示的始终是网站上在线人数的数目, 从而实现动态也统计在线人数, 结果如图 4-28 所示。

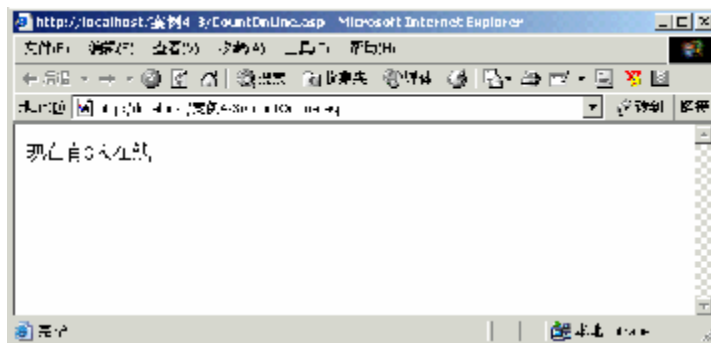


图 4-28 在线人数统计

案例 4-4: 聊天室研究

利用 Application 和 Session 可以做出比较专业的聊天室。这个聊天室对上面的聊天室进行了功能上的扩充, 由以下 5 个文件组成。

- (1) index.asp: 聊天室的登录界面。
- (2) do_login.asp: 登录处理界面。
- (3) chatpage.htm: 聊天室的框架文件。
- (4) message.asp: 聊天室的信息输入界面。
- (5) display.asp: 聊天信息显示页面。

首先是登录界面, 如程序 index.asp 所示。

案例名称: 登录界面

程序名称: index.asp

```
<%@ Language=Jscript %>
<HTML>
    <HEAD>
    </HEAD>
<BODY>
<FORM ACTION="do_login.asp" METHOD="post">
请输入你的大名: <input name="txtUserID" type="text" size="10" value="采花大盗"
><br>
请输入你的密码: <input name="txtUserPWD" type="Password" size="8" value="n22">
<INPUT TYPE="SUBMIT" VALUE="提交">
</FORM>
</BODY>
</HTML>
```

该文件提供了两个文本框, 一个输入姓名, 另一个输入密码。当单击“提交”按钮时, 周用 do_login.asp 文件进行处理, 如下所示。

案例名称: 登录处理程序

程序名称: do_login.asp

```
<%@ Language=Jscript %>
<%
    var strUserID = Request("txtUserID")(1);
    var strUserPWD = Request("txtUserPWD")(1);
    Session("username") = strUserID;
    Session("no")= strUserPWD;
    Response.Redirect ("chatpage.htm");
%>
```

该程序首先利用 Request 对象得到用户名和密码, 然后将用户名和密码赋值给两个 Session 对象, 语句 Request("txtUserID")(1)的功能是返回 Request 变量中字符串部分, 然后转到聊天框架页面 chatpage.htm。如程序 chatpage.htm 所示。

案例名称: 实现聊天框架

程序名称: chatpage.htm

```
<HTML>
    <HEAD>
        <TITLE></TITLE>
    </HEAD>
    <FRAMESET ROWS="*,100">
```

```

<FRAME SRC="DISPLAY.ASP">
<FRAME SRC="MESSAGE.ASP">
</FRAMESET>
</HTML>

```

程序 chatpage.htm 实现了一个上下框架，让用户在如程序 message.asp 所示的文件中输入数据，然后显示在 display.asp 文件中。

案例名称：发送聊天信息

程序名称：message.asp

```

<%@ Language=Jscript %>
<%
    var mywords = Request("message");
    var oneSentence = "姓名: " + Session("username") ;
    oneSentence += "机器号:" + Session("no")+ "说:    " + mywords;
    Application.Lock();
    Application("talk") = Application("talk") + oneSentence + "<br>";
    Application.Unlock();
%>
<HTML>
    <HEAD>
    </HEAD>
    <BODY BGCOLOR="LIGHTBLUE">
        <FORM METHOD="POST" ACTION="MESSAGE.ASP">
            <INPUT NAME="message" TYPE="TEXT" SIZE="50">
            <INPUT TYPE="SUBMIT" VALUE="SEND">
        </FORM>
    </BODY>
</HTML>

```

程序 message.asp 主要实现用户聊天信息的输入，当用户单击“提交”按钮时，将信息保存到 Application("talk")中。如程序 display.asp 所示。

案例名称：显示聊天信息

程序名称：display.asp

```

<%@ Language=Jscript %>
<HTML>
<HEAD>
<META HTTP-EQUIV="REFRESH" CONTENT="3;URL=display.asp">
<SCRIPT LANGUAGE="JavaScript">
    function scrollWindow()
    {

```

```

        this.scroll(0,65000);
        setTimeout('scrollWindow()',200);
    }
    scrollWindow();
</SCRIPT>
</HEAD>
<BODY>
<%
    Response.Write(Application("talk"));
%>
</BODY>
</HTML>

```

文件实现聊天室自动更新用户聊天信息的功能, 关键的语句如下:

```
<META HTTP-EQUIV="REFRESH" CONTENT="3;URL=display.asp">
```

这条语句的功能是每隔三秒钟重新调用一次 `display.asp` 文件, 也就实现了聊天室自动刷新的功能。下面的语句是实现自动滚屏, 使用户见到的内容始终是最近提交的内容。

```

<SCRIPT LANGUAGE="JavaScript">
function scrollWindow()
{
    this.scroll(0,65000);
    setTimeout('scrollWindow()',200);
}
scrollWindow();
</SCRIPT>

```

聊天室的登录界面如图 4-29 所示。

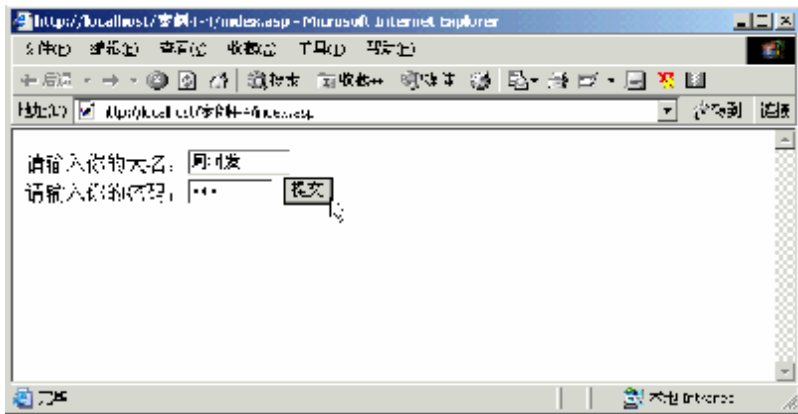


图 4-29 聊天室登录界面

当登录到聊天室以后, 就可以进行聊天了, 如图 4-30 所示。

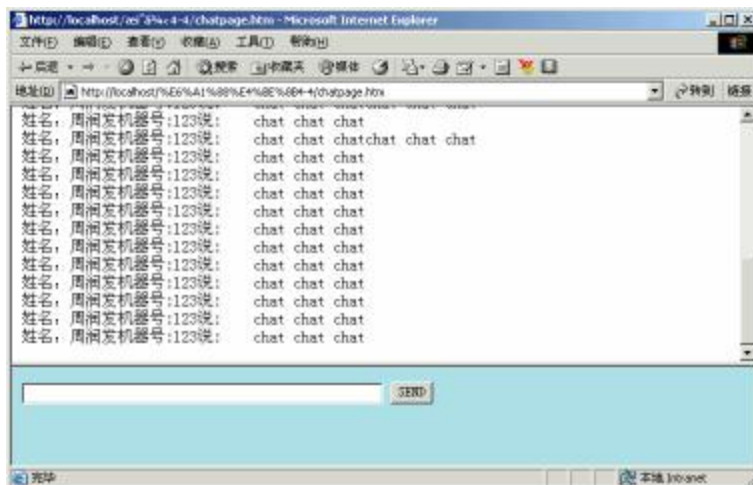


图 4-30 聊天室主界面

案例 4-5: 简易 Session 版购物车

利用 Session 保存用户选购的商品信息。本购物车程序包含三个程序:

- (1) buy1.asp: 购物网页一。
- (2) buy2.asp: 购物网页二。
- (3) display.asp: 查看购物车程序。

购物网页一如程序 buy1.asp 所示。

案例名称: 购物网页一

程序名称: buy1.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
if (Request("c1").Count > 0)
{
    Session("s1") = Request("c1")(1);
}
if (Request("c2").Count > 0)
{
    Session("s2") = Request("c2")(1);
}
if (Request("c3").Count > 0)
{
    Session("s3") = Request("c3")(1);
}
%>
```


各种肉大甩卖,一律十块:

```
<form method="POST" action="buy1.asp">
  <p> </p>
  <p><input type="checkbox" name="c1" value="猪肉">猪肉</p>
  <p><input type="checkbox" name="c2" value="牛肉">牛肉</p>
  <p><input type="checkbox" name="c3" value="羊肉">羊肉</p>
  <p><input type="submit" value="提交" name="B1">
  <input type="reset" value="全部重写" name="B2">
  <a href="buy2.asp">买点别的</a>
  <a href="display.asp">查看购物车</a>
</P>
</FORM>
</BODY>
</HTML>
```

程序中“if (Request("c1").Count > 0)”是判断 c1 复选框是否选中,如果提交的时候 c1 被选中,Count 属性为 1,如果第一次执行该网页或者 c1 没有选中,Count 属性均为 0。Request 读取数据的时候,是一集合(数组)传递的,可以利用“Request("c1")(1);”得到读取的文本数据。程序显示如图 4-31 所示。

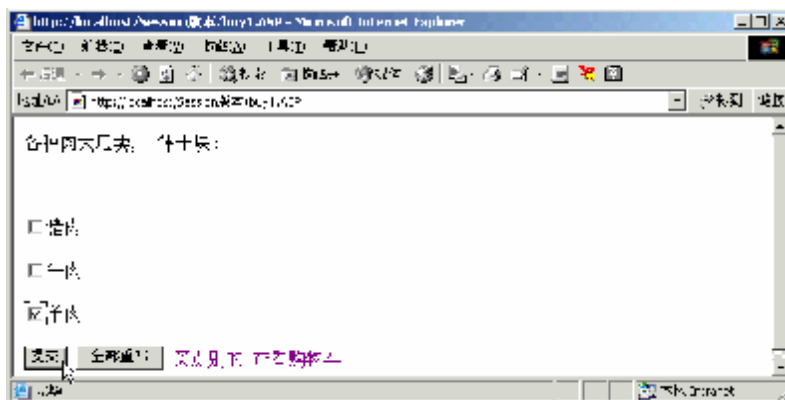


图 4-31 购物网页一

购物网页二和这个程序类似,如程序 buy2.asp 所示。

案例名称: 购物网页二

程序名称: buy2.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
if (Request("b1").Count > 0)
{
```

```
Session("s4") = Request("b1")(1);  
}  
if (Request("b2").Count > 0)  
{  
    Session("s5") = Request("b2")(1);  
}  
if (Request("b3").Count > 0)  
{  
    Session("s6") = Request("b3")(1);  
}  
%>
```

各种球大甩卖,一律八块:

```
<form method="POST" action="buy2.asp">  
    <p> </p>  
    <p><input type="checkbox" name="b1" value="篮球">篮球</p>  
    <p><input type="checkbox" name="b2" value="足球">足球</p>  
    <p><input type="checkbox" name="b3" value="排球">排球</p>  
    <p><input type="submit" value="提交" name="x1">  
    <input type="reset" value="全部重写" name="B2">  
    <a href="buy1.asp">买点别的</a>  
    <a href="display.asp">查看购物车</a>  
    </P>  
</FORM>  
</BODY>  
</HTML>
```

程序显示的结果如图 4-32 所示。

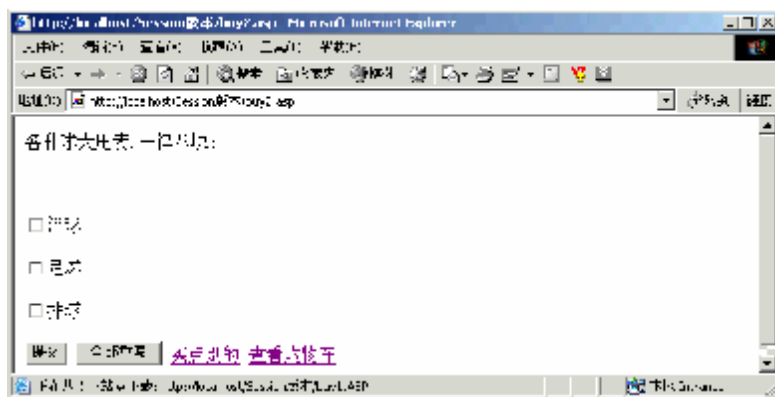


图 4-32 购物网页二

选择几个商品提交,程序将商品信息保存到 Session 中,可以单击“查看购物车”按钮,结果如图 4-33 所示。

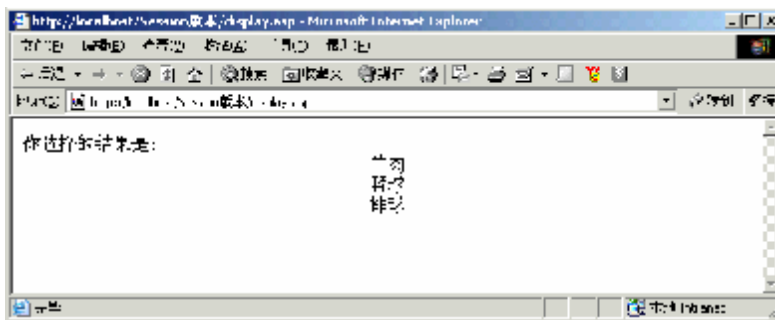


图 4-33 查看购物车

显示结果程序如程序 display.asp 文件所示。

案例名称：显示购物车程序

程序名称：display.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
你选择的结果是：
    <center>
    <%
    if(Session("s1")!=undefined)
        Response.Write(Session("s1")+"<br>");
    if(Session("s2")!=undefined)
        Response.Write(Session("s2")+"<br>");
    if(Session("s3")!=undefined)
        Response.Write(Session("s3")+"<br>");
    if(Session("s4")!=undefined)
        Response.Write(Session("s4")+"<br>");
    if(Session("s5")!=undefined)
        Response.Write(Session("s5")+"<br>");
    if(Session("s6")!=undefined)
        Response.Write(Session("s6")+"<br>");
    %>
    </center>
</BODY>
</HTML>
```

该程序中做了一个判断，如果 Session 中有值的话（不是 undefined）就输出到浏览器上。

小结

本章重点理解五大对象、一个集合和一个文件的概念和使用方法。理解 Response 对象提

共的 Write 方法、Redirect 方法和 End 方法。如何利用 Request 获得 Form 表单中的信息，如何获得超级链接传递的变量。理解 Application 对象，如何利用 Application 对象实现聊天室和计数器。理解 Session 对象及其自定义属性。使用 Server 对象获取网站的物理路径，如何向浏览器写入 Cookie 和如何将 Cookie 读取出来。理解 global.asa 文件的特点和功能。

课后习题和上机练习

1. Response 对象有什么功能，Response.Wrtie 和 document.write 有什么区别？
2. Request.Form 和 Request.QueryString 有什么异同点？
3. 如何获得获得客户端的 IP 地址？
4. Application 对象有什么特点？和 Session 对象有什么联系和区别？
5. 如何利用程序获得某网站的根路径？
6. 程序如何向浏览器写入 Cookie 集合，如何从浏览器端读取 Cookie 集合。
7. 改写案例 4-3，实现人数统计的图形显示。（上机练习）。
8. 改写案例 4-3，给聊天室添加发言的颜色选择，发言时可以选择三种颜色（红色 Red，蓝色 Blue 和黑色 Black），发言在显示时显示成所选颜色。（上机练习）。

第 5 章 ASP 内置组件

本章要点

本章主要介绍两种常用的内置组件：文件组件和广告组件。操作一个文件系统分成三个层次：操作文件的内容、操作文件和操作文件夹。利用文件组件实现个人主页编辑器和文件反的留言簿。利用广告组件实现广告处理。

5.1 使用内置文件组件

FileSystemObject (FSO, 文件系统对象) 是 IIS 自带的一个组件, 利用这个组件的一些方法可以在服务器上操作文件的内容、操作文件和操作文件夹。

5.1.1 使用对文件操作的组件

利用 IIS 5.0 自带的 FileSystemObject 组件, 几乎可以控制服务器的所有文件系统。为了实现这些功能, 需要使用下面对象。

(1) FileSystemObject: 它包括一些基本的对文件系统进行操作的方法, 比如复制和删除文件夹或者文件。

(2) TextStream: 它用来读写文件。

(3) File: 它的方法和属性被用来处理单独的文件。

(4) Folder: 它的方法和属性被用来处理文件夹。

要创建一个文本文件并且写入一些内容, 可以使用 FileSystemObject 和 TextStream 对象。首先创建一个 FileSystemObject 对象的实例, 然后, 再利用 CreateTextFile() 方法创建一个 TextStream 对象的实例, 最后利用 TextStream 对象的 WriteLine() 方法来写入文件。如程序 5-01.asp 所示。

案例名称: 创建文本文件

程序名称: 5-01.asp

```
<%@ Language=Jscript %>
<%
var path = Server.MapPath("test.txt");
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
var MyTextFile = MyFileObject.CreateTextFile(path);
MyTextFile.WriteLine("我在当前的目录创建了一个叫 test.txt 的文件");
MyTextFile.Close();
%>
```

程序执行完后，在此 ASP 文件所在的目录下就会出现一个名为 test.txt 的文本文件，文件内容是：“我在当前的目录创建了一个叫 test.txt 的文件”，如图 5-1 所示。

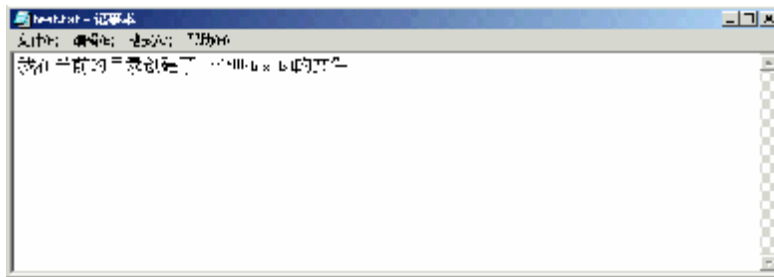


图 5-1 创建生成的文本文件

CreateTextFile()方法用来创建一个新的文本文件，当这个方法被调用，那么就返回一个 TextStream 对象，可以连续写入多行内容，如程序 5-02.asp 所示。

案例名称：连续写入多行信息

程序名称：5-02.asp

```
<%@ Language=Jscript %>
<%
var path = Server.MapPath("test.txt");
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject")
var MyTextFile = MyFileObject.CreateTextFile(path)
for ( var i = 0;i < 30; i++)
{
MyTextFile.WriteLine("Hello World!")
}
MyTextFile.Close();
%>
```

这样，执行程序就向文本文件中自动写入了 30 行“Hello World!”，如图 5-2 所示。

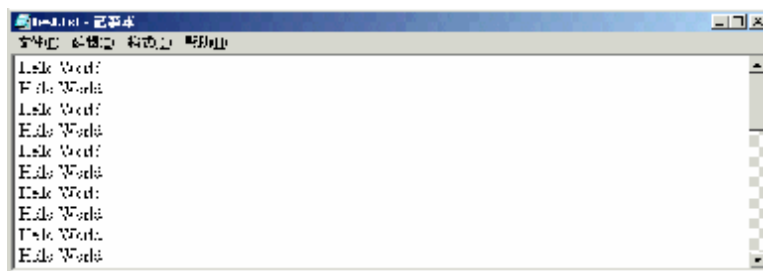


图 5-2 写入多行的信息

从文本文件中读取和向文本文件中追加数据时，首先要创建一个 FileSystemObject 对象的实例，然后利用 OpenTextFile()方法来创建一个 TextStream 对象的实例，最后利用 TextStream 对象的 ReadLine 方法来读取文件的内容，如程序 5-03.asp 所示。

案例名称: 读取文件的内容

程序名称: 5-03.asp

```
<%@ Language=Jscript %>
<%
var path = Server.MapPath("test.txt");
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
var MyTextFile = MyFileObject.OpenTextFile(path);
while(!MyTextFile.AtEndOfStream)
{
    Response.Write(MyTextFile.ReadLine() + "<br>")
}
MyTextFile.Close();
%>
```

程序将 test.txt 文件读出输出到浏览器上, ReadLine 方法每次读取一行, 直到文件的结束, 程序执行结果如图 5-3 所示。

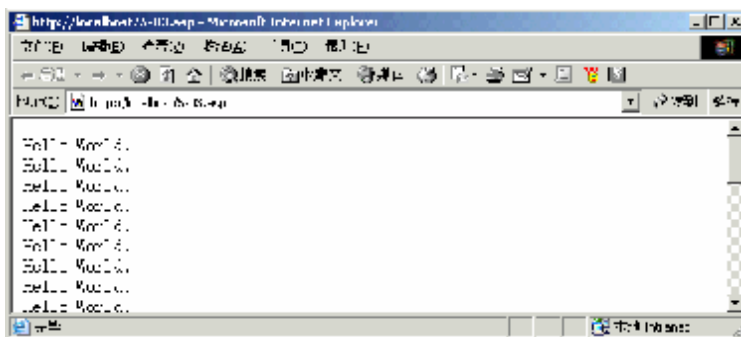


图 5-3 读取文件的内容

程序将文本文件 test.txt 中所有的内容读出来并且显示在浏览器上。如果文件不存在, 会显示错误信息。

While 循环将文件内容逐行读取, 到文件末尾时, AtEndOfStream 属性就会变为 true, 从而退出循环。

除了使用 ReadLine 方法以外, 还可以使用 Read()方法。Read()方法会从指定打开的文本文件中返回指定数目的字符。如程序 5-04.asp 所示。

案例名称: Read 方法的使用

程序名称: 5-04.asp

```
<%@ Language=Jscript %>
<%
var path = Server.MapPath("test.txt");
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
```

```
var MyTextFile = MyFileObject.OpenTextFile(path);  
while (!MyTextFile.AtEndOfLine)  
{  
    Response.Write(MyTextFile.Read(1))  
}  
MyTextFile.Close();  
%>
```

Read(1)每次读取一个字符，一直到第一行结尾为止，所以读出了第一行内容，结果如图 5-4 所示。

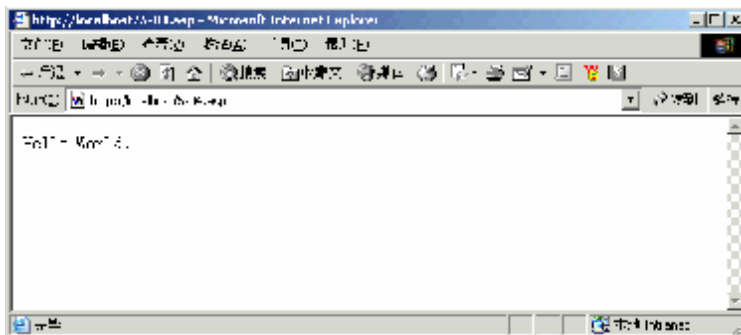


图 5-4 Read 函数的使用

读取文本文件的方法如下。

- (1) Read(Number): 从文本文件中读取限定数目个字符。
- (2) ReadLine(): 从文本文件中读取一行。
- (3) ReadAll(): 这个方法接受 TextStream 文件的所有内容。

一般说来，OpenTextFile()方法用来读取数据，也可以用它追加数据信息，例如，在某文本文件后面追加一些新内容，如程序 5-05.asp 所示。

案例名称：追加数据

程序名称：5-05.asp

```
<%@ Language=Jscript %>  
<%  
var path = Server.MapPath("test.log");  
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject")  
var MyTextFile = MyFileObject.OpenTextFile(path,8,true)  
MyTextFile.WriteLine(Request.ServerVariables("REMOTE_ADDR"))  
MyTextFile.Close();  
%>
```

OpenTextFile(path,8,true)中有三个参数，第一个参数“path”打开文件的地址，第二个参数“8”是打开方式，这里是以追加的形式打开，第三个参数“true”的意思是如果该文件不存在就创建一个新文件。Request.ServerVariables("REMOTE_ADDR")为客户端的 IP 地址，这

所有访问过该页面的 IP 地址就被记录下来了，创建的文件如图 5-5 所示。

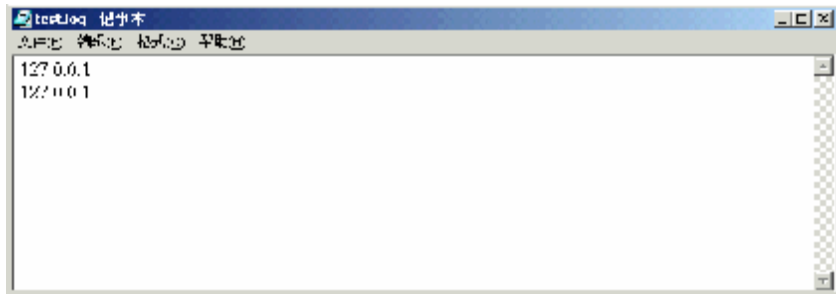


图 5-5 创建的 log 文件显示的结果

5.1.2 对文件进行处理

有多种方法可对文件进行复制、移动及删除工作。可以利用 `FileSystemObject` 对象的方法也可以利用 `File` 对象的方法。`FileSystemObject` 对象对文件操作的常用方法如下。

(1) `CopyFile(source, destination,[Overwrite])`: 这个方法进行复制操作，可以使用 `source` 参数通配符在一个时刻进行多个文件的复制。`OverWrite` 参数将在目标文件已经存在的情况下进行覆盖操作。

(2) `MoveFile(source, destination)`: 这个方法对文件进行移动操作，同样可以使用通配符来移动多个文件，如果目的文件已经存在，则会报错。

(3) `DeleteFile(FileSpecifier)`: 这个方法功能是删除指定文件，同样还可以利用通配符实现多文件的删除。如果没有符合通配符的文件，将会报错。

使用这些方法之前，首先创建一个 `FileSystemObject` 对象的实例。如程序 5-06.asp 所示。

案例名称：复制文件

程序名称：5-06.asp

```
<%@ Language=Jscript %>
<%
//创建一个 FileSystemObject 的实例
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
//创建一个要进行操作的文件
var MyFile = MyFileObject.CreateTextFile("c:\\test.txt");
MyFile.WriteLine("Hello")
MyFile.Close();
//复制文件操作
MyFileObject.CopyFile("c:\\test.txt", "c:\\test2.txt");
//移动文件操作
MyFileObject.MoveFile("c:\\test.txt", "c:\\test3.txt");
//删除这些文件
```

```
//MyFileObject.DeleteFile("c:\\test.txt");  
//MyFileObject.DeleteFile("c:\\test3.txt");  
%>
```

运行完上面的程序后就在 C 盘根目录下创建了一个名为 Test.txt 的文件, 并且复制为 test2.txt 文件, 如图 5-6 所示。

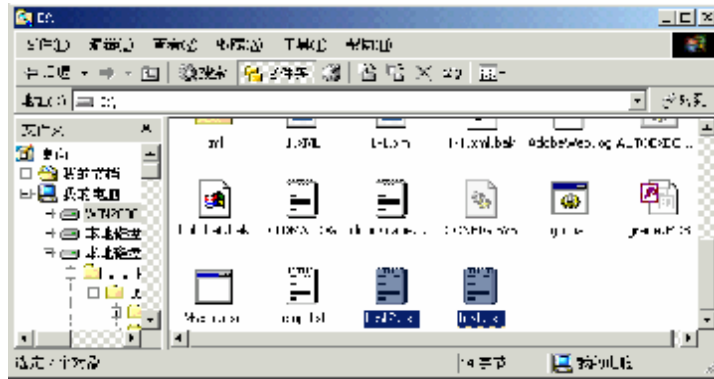


图 5-6 创建文件的结果

除了使用 FileSystemObject 对象的方法来操作文件以外, 还可以使用 FILE 对象提供的如下方法。

(1) Copy (newcopy, [Overwrite]): 该方法给当前文件创建备份, 当可选的 OverWrite 参数为 true 时, 如果存在同名的文件, 则覆盖。

(2) Move (newcopy): 该方法功能是移动当前文件。

(3) Delete(): 删除当前文件。

首先创建 File 对象的一个实例, 如程序 5-07.asp 所示。

案例名称: 复制文件

程序名称: 5-07.asp

```
<%@ Language=Jscript %>  
<%  
//创建一个 FileSystemObject 对象的实例  
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject")  
//创建一个 File 对象的实例  
var MyFile = MyFileObject.CreateTextFile("c:\\test.txt")  
MyFile.WriteLine("www.gettop.net")  
MyFile.Close();  
var myfileobject = MyFileObject.GetFile("c:\\test.txt")  
//复制文件  
myfileobject.Copy("c:\\test4.txt");  
//移动文件  
//myfileobject.Move("c:\\test5.txt");
```

```
//删除文件
//myfileobject.Delete();
%>
```

首先在 C 盘根目录下创建一个 test.txt 文件，程序首先赋值复制出一个 test4.txt 文件，又将该文件移动成为 Test5.txt，如图 5-7 所示。

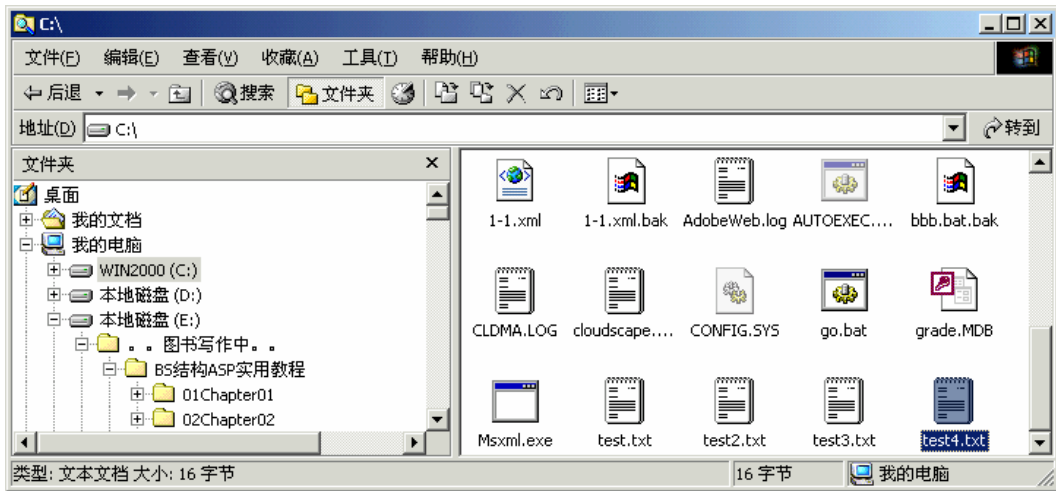


图 5-7 使用文件对象的操作结果

如果需要确定一个特定的文件是否存在，需要使用 FileSystemObject 对象的 FileExists() 方法，从返回值是 true 还是 false 来判断文件是否存在，如程序 5-08.asp 所示。

案例名称：检测文件是否存在

程序名称：5-08.asp

```
<%@ Language=Jscript %>
<%
//创建一个 FileSystemObject 对象实例
var MyFileObject=Server.CreateObject("Scripting.FileSystemObject");
if( MyFileObject.FileExists("c:\\test.txt") )
{
Response.Write("存在这个文件");
}
else
{
Response.Write("不存在这个文件");
}
%>
```

程序执行的结果是判断 C:\test.txt 文件是否存在，如果存在则返回存在，若不存在则返回不存在，如图 5-8 所示。

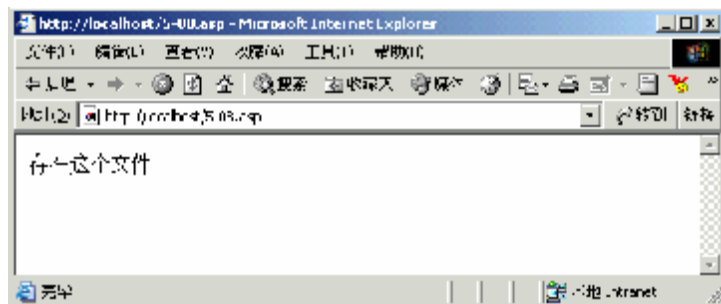


图 5-8 测试文件是否存在

还可以得到文件的相关属性，首先创建一个 File 对象的实例，显示当前目录下 test.txt 文件的属性。如程序 5-09.asp 所示。

案例名称：得到文件的相关属性

程序名称：5-09.asp

```
<%@ Language=Jscript %>
<%
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
var path          = Server.mappath("test.txt");
var MyFile        = MyFileObject.GetFile(path);
%>
<pre>
名称:      <%=MyFile.Name %>
路径:      <%=MyFile.Path %>
驱动器:    <%=MyFile.Drive %>
大小:      <%=MyFile.size %>
类型:      <%=MyFile.type %>
属性:      <%=MyFile.Attributes %>
创建日期: <%=MyFile.DateCreated %>
</pre>
```

显示结果如图 5-9 所示。

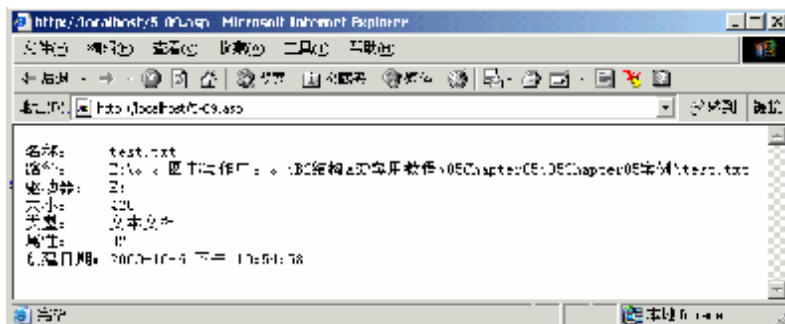


图 5-9 显示文件的属性

Attributes 属性需要解释，它的返回值如表 5-1 所示。

表 5-1 文件的属性值

Attribute 属性	值
Normal (正常)	0
Read-Only (只读)	1
Hidden (隐藏)	2
System (系统文件)	4
Volume (驱动器)	8
Directory (目录)	16
Archive (存档)	32
Alias (快捷方式)	64
Compressed (压缩)	128

由于 test.txt 文件的属性为 Archive，所以属性就是 32。也可以利用操作系统查看文件的属性，如图 5-10 所示。

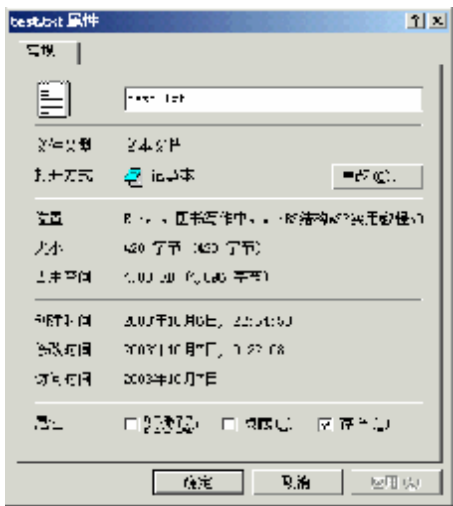


图 5-10 查看文件的属性

属性还可以进行设置，例如，使 c:\test.txt 具有 Archive 和 Read-Only 两种属性，如程序 5-10.asp 所示。

案例名称：修改文件的属性

程序名称：5-10.asp

```
<%@ Language=Jscript %>
<%
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
var path          = Server.mappath("test.txt ");
var MyFile        = MyFileObject.GetFile(path);
MyFile.attributes = 33;
%>
```

程序将 test.txt 文件的属性修改为只读和存档的类型，文件的属性如图 5-11 所示。

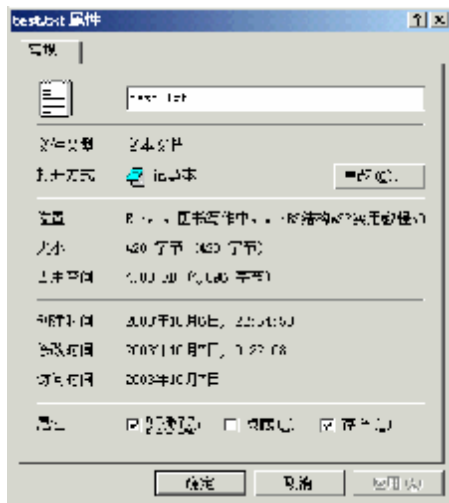


图 5-11 修改后文件的属性

5.1.3 对文件夹和驱动器进行操作

FileSystemObject 提供对文件夹和驱动器进行操作的一些方法，获取有关驱动器的信息及对目录的创建、删除移动和显示内容操作。用户在使用这些集合和方法之前，必须首先创建一个 Driver 对象的实例。可以使用 FileSystemObject 对象的 Getdrive()方法来创建，如程序 5-11.asp 所示，将返回 C 盘的容量。

案例名称：获取 C 盘的容量

程序名称：5-11.asp

```
<%@ Language=Jscript %>
<%
var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
var MyDrive      = MyFileObject.Getdrive("C:");
Response.Write("c:的容量是" + MyDrive.totalsize);
%>
```

程序返回服务器端 C 盘容量的大小，显示的结果如图 5-12 所示。

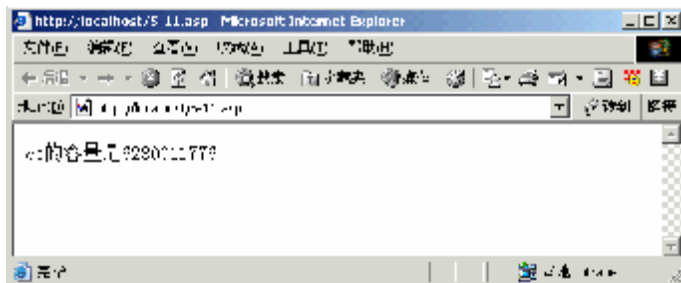


图 5-12 读取 C 盘的容量

FileSystemObject 对象包括如下处理文件夹的方法。

(1) CopyFolder(source, destination [,Overwrite]): 用来进行文件夹的复制, 可以使用通配符来进行多目录的复制, 如果目的目录已经存在, 可以通过将 Overwrite 参数设为 true 进行覆盖, 默认值为 true。

(2) CreateFolder(FolderSpecifier): 创建一个指定的文件夹。

(3) DeleteFolder(FolderSpecifier): 删除一个指定的文件夹。

(4) FolderExists(FolderSpecifier): 如果该指定文件夹存在, 返回 true, 否则返回 false。

(5) GetFolder(FolderSpecifier): 由指定的文件夹创建一个 Folder 对象。

(6) GetParentFolderName(Path): 返回包含该路径的上一级目录名。

(7) MoveFolder(source, Destination): 将指定目录进行移动, 可以利用通配符来移动多个文件夹。

对文件夹的操作如程序 5-12.asp 所示。

案例名称: 操作文件夹

程序名称: 5-12.asp

```
<%@ Language=Jscript %>
<%
    var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");
    MyFileObject.CreateFolder("C:\\\\ToBeDelete");
    MyFileObject.CopyFolder("C:\\\\ToBeDelete","C:\\\\ToBeDelete3");
    // MyFileObject.MoveFolder("C:\\\\ToBeDelete","C:\\\\ToBeDelete2");
    // MyFileObject.DeleteFolder("C:\\\\ToBeDelete2");
%>
```

运行程序后就在 C 盘根目录下创建了 ToBeDelete 目录, 然后将其复制成 ToBeDelete3, 如图 5-13 所示。

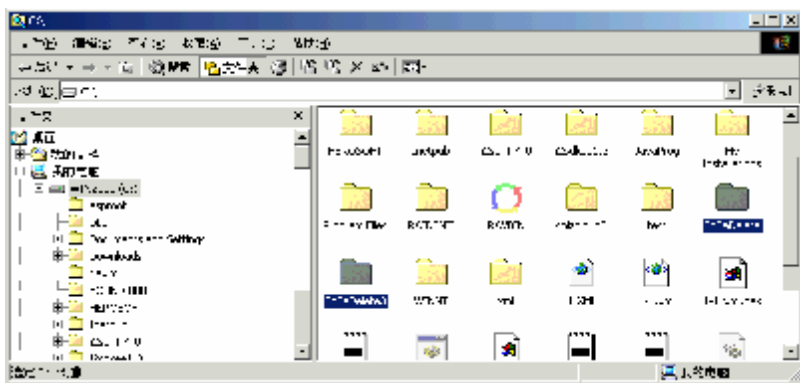


图 5-13 操作文件夹

Folder 对象也提供一些重要的方法和属性。如下所示。

(1) CopyFolder(new copy [,overwrite]): 将当前文件夹复制到新的位置。

(2) DeleteFolder: 删除当前文件夹。

- (3) Files: 返回所有该目录下文件的集合。其中隐含文件不显示。
- (4) IsRootFolder: 如果是根目录返回 true。
- (5) MoveFolder(FolderSpecifier): 移动当前目录到另外的位置。
- (6) Name: 返回当前目录名称。
- (7) ParentFolder: 返回到上一级目录。
- (8) Size: 显示目前目录及子目录的所有文件大小的总和。
- (9) SubFolders: 返回为所有这个文件夹下面子目录的集合。

程序 5-13.asp 将返回这个文件夹内容的大小, 如下所示。

案例名称: 创建文件夹

程序名称: 5-13.asp

```
<%@ Language=Jscript %>
<%
    var MyFileObject = Server.CreateObject("Scripting.FileSystemObject")
    var MyFolder = MyFileObject.GetFolder("c:\\inetpub")
    Response.Write(MyFolder.Size);
%>
```

程序执行的结果如图 5-14.asp 所示。

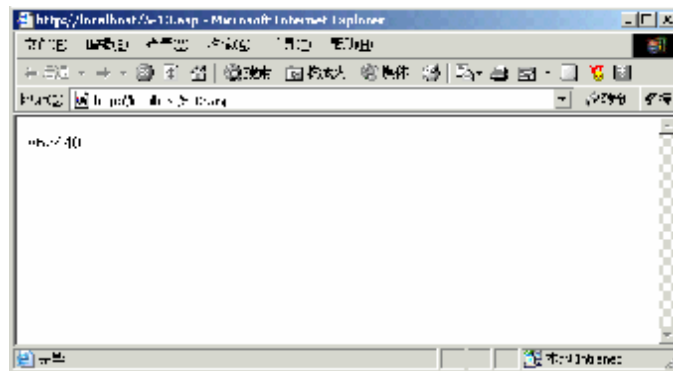


图 5-14 显示所有的子文件

案例 5-1: 在线个人主页编辑器

利用前面的知识可以创建个人主页编辑器。该编辑器由两个文件组成, 一个是编辑的界面, 另一个是显示的界面。编辑界面如程序 edit.asp 所示。

案例名称: 编辑页面

程序名称: edit.asp

```
<%@ Language=Jscript %>
<%
    var fs = Server.CreateObject("Scripting.FileSystemObject");
    var File = Server.MapPath( "page.txt" );
```



```

var txtf = fs.OpenTextFile(File,1);
if (!txtf.atEndOfStream)
{
    Content = txtf.ReadAll();
}
else
{
    Content = "abc";
}
txtf.Close();
%>
<HTML>
<BODY>
<H2>个人主页编辑器<HR></H2>
<FORM Action="display.asp" Method="POST">
    <TEXTAREA Rows="10" Cols="60" Name="Sample">
        <%=Content%>
    </TEXTAREA><P>
    <INPUT Type="Submit" Value="提交">
</FORM>
</BODY>
</HTML>

```

程序从文件中将内容读出来，显示到多行文本框中，留给用户修改。OpenTextFile(File,1)方法中的第二个参数“1”的意思是以只读的形式打开文件，如果是“2”的话是以只写的方式打开文件，“4”是以读写的方法打开，“8”是以追加的方式打开文件。程序 display.asp 是主页的显示文件。

案例名称：显示页面

程序名称：display.asp

```

<%@ Language=Jscript %>
<%
Content = Request("Sample")(1);
var fs = Server.CreateObject("Scripting.FileSystemObject");
var File = Server.MapPath( "page.txt" );
var txtf = fs.OpenTextFile( File, 2, true );
txtf.Write(Content);
%>
<HTML>
<BODY>
<H2>个人主页<HR></H2>

```

```
<%=Content%>  
<br><br>  
<A HREF="EDIT.ASP">返回</A>  
</BODY>  
</HTML>
```

这时可进入编辑界面，用户可以在此输入文本和 HTML 代码，如图 5-15 所示。



图 5-15 个人主页编辑界面

编辑完毕后，系统自动保存到一个文本文件，并且将结果显示出来，如图 5-16 所示。

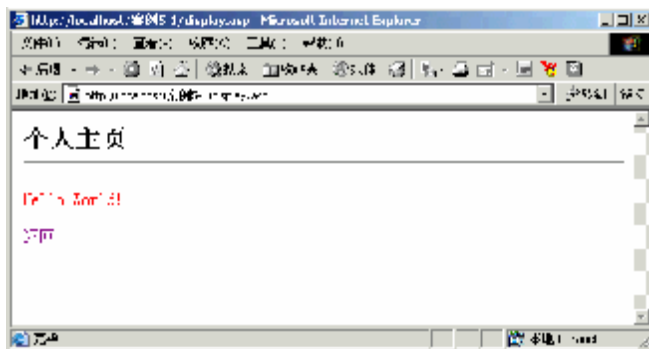


图 5-16 个人主页显示的内容

案例 5-2：文件版留言簿

该留言簿的功能是用户可以在网站上留一些信息，它由四个文件组成。

- (1) input.htm：留言输入界面。
- (2) handle.asp：留言处理保存页面。
- (3) display.asp：留言显示页面。
- (4) lyb.txt：留言保存文件。

用户输入界面是一个标准的表单，如程序 input.htm 所示。

案例名称：留言簿输入界面

程序名称：input.htm

```
<HTML>

  <HEAD>
    <TITLE>留言簿</TITLE>
  </HEAD>

  <BODY>
    <TABLE BORDER="1" CELLSPACING="0" CELLPADDING="4" WIDTH="100%">
      <TR>
        <TD ALIGN="CENTER"><FONT SIZE="+2" COLOR="#000080">
          <B>留言表单</B></FONT></TD>
      </TR>
    </TABLE>

    <FORM ACTION="handle.ASP" METHOD="POST">
      <TABLE BORDER="1" WIDTH="100%">
        <TR>
          <TD><FONT COLOR="#000080">姓名: </FONT></TD>
          <TD><INPUT TYPE="TEXT" NAME="Name" SIZE="40"></TD>
        </TR>
        <TR>
          <TD><FONT COLOR="#000080">E-MAIL:</FONT></TD>
          <TD><INPUT TYPE="TEXT" NAME="E-mail" SIZE="40"></TD>
        </TR>
        <TR>
          <TD><FONT COLOR="#000080">主题: </FONT></TD>
          <TD><INPUT TYPE="TEXT" NAME="Subject" SIZE="40"></TD>
        </TR>
        <TR>
          <TD VALIGN="TOP"><FONT COLOR="#000080">留言: </FONT> </TD>
          <TD><TEXTAREA NAME="Memo" ROWS="6" COLS="39">
            </TEXTAREA></TD>
        </TR>
      </TABLE>

      <DIV ALIGN="CENTER"><CENTER><TABLE WIDTH="100%">
        <TR ALIGN="CENTER">
          <TD><INPUT TYPE="SUBMIT" NAME="SEND" VALUE="提交留言"></td>
          <TD><A HREF="display.asp">浏览留言</A></TD>
          <TD><INPUT TYPE="RESET" VALUE="清除重写"></TD>
        </TR>
      </TABLE>
    </CENTER></DIV>
    </FORM>
```

```
</BODY>
```

```
</HTML>
```

用户可在该页面中输入相关留言信息，如图 5-17 所示。



图 5-17 留言输入界面

输入完毕后，单击“提交”按钮，调用 `handle.asp` 文件处理留言信息，它是留言簿的核心文件。如程序 `handle.asp` 所示。

案例名称：留言处理文件

程序名称：handle.asp

```
<%@ Language="Jscript"%>
<%
    Name    = Request("Name")(1);
    E-mail  = Request("E-mail")(1);
    Subject = Request("Subject")(1);
    Memo    = Request("Memo")(1);

    if ("" == Name)
    {
        Response.Write("姓名不能为空白!");
        Response.End();
    }
    if ("" == E-mail)
    {
        Response.Write("E-mail 不能为空白!");
        Response.End();
    }
    if ("" == Subject)
    {
        Response.Write("主题不能为空白!");
    }
}
```

```
        Response.End();
    }
    if ("" == Memo)
    {
        Response.Write("留言不能为空白!");
        Response.End();
    }

    // 第一行包含姓名
    Line1 = "留言人: " + Name + "<br>";
    E-mail = "<A HREF=mailto:" + E-mail + ">" + E-mail + "</A>";
    // 第二行包含 E-mail
    Line2 = "E-mail □" + E-mail + "<BR>";
    // 第三行包含主题
    Line3 = "主 题: " + Subject + "<BR>";
    //为了显示不同的背景颜色, 采用表格输出
    Line4 = "<TABLE BORDER=0 BGCOLOR='ORANGE'><TR><TD>";
    Line4 = Line4 + Memo + "</TD></TR></TABLE>";
    // 第五行为留言时间
    var now = new Date();
    Line5 = now.toLocaleString();

    try
    {
        // 建立 FileSystemObject 对象
        fso = Server.CreateObject("Scripting.FileSystemObject");
        // 取得 lyb.txt 及 lybold.txt 的完整路径
        FilePath = Server.MapPath("lyb.txt");
        OldFilePath = Server.MapPath("lybold.txt");

        // 将 lyb.txt 更名为 lybold.txt
        fso.MoveFile(FilePath, OldFilePath);
        // 打开 lybold.txt
        fin = fso.OpenTextFile(OldFilePath);
        // 建立 lyb.txt
        fout = fso.CreateTextFile(FilePath);
        // 写入留言
        fout.WriteLine(Line1);
        fout.WriteLine(Line2);
        fout.WriteLine(Line3);
    }
```

```

        fout.WriteLine(Line4);
        fout.WriteLine(Line5);
        fout.WriteLine("<HR>");
        // 一次读取整个 lybold.txt, 然后写入 lyb.txt
        fout.WriteLine(fin.ReadAll());
        // 关闭 lyb.txt
        fin.Close();
        fout.Close();
        // 删除 lybold.txt
        fso.DeleteFile(OldFilePath);

        Response.Redirect("display.asp");
    }
    catch(e)
    {}
    %>

```

该程序包含四大部分：第一部分读取 Form 表单的内容；第二部分做验证，如果有输入框为空则停止执行程序；第三部分将读取的信息设置成固定的格式，分别保存在几个字符串中；第四部分将新增的留言写入文件。为了将最新的留言写到文件的最上面，首先将 lyb.txt 文件改名成 lybold.txt，再创建一个 lyb.txt 文件并向该文件中写入最新的留言信息，然后再将 lybold.txt 文件的内容追加到 lyb.txt 文件中，注意这种算法。

处理完以后转到 display.asp 文件。如下所示。

案例名称：留言显示文件

程序名称：display.asp

```

<HTML>
    <HEAD>
        <TITLE>留言簿</TITLE>
    </HEAD>
    <BODY>
        <H2 ALIGN="CENTER">留言簿</H2>
        <HR WIDTH="100%">
        <!--#include file="lyb.txt" -->
        <A HREF="INPUT.HTM"><P ALIGN="CENTER">
            返回留言表单</A></P>
    </BODY>
</HTML>

```

这个文件没有用一条代码就实现了留言的输出，语句“<!--#include file="lyb.txt" -->”的力能是将 lyb.txt 文件内容全部包含到当前文件中来。程序显示的结果如图 5-18 所示。



图 5-18 留言簿的显示页面

5.2 广告的处理

利用内置组件 AD ROTATOR 可以实现动广告图片的动态显示。使用该组件，可以在每次访问中显示不同的图标，可以设置广告的不同权重使得显示频率不同。

案例 5-3：广告图片显示

Ad Rotator 组件只有一个方法，getAdvertisement()方法。使用方法如程序 ad.asp 所示。

案例名称：实现广告

程序名称：ad.asp

```
<%@ Language=Jscript %>
<HTML>
  <BODY>
    <%
      var MyAdvertise = Server.CreateObject("MSWC.AdRotator")
    %>
    <CENTER><%=MyAdvertise.GetAdvertisement("ad.txt") %></CENTER>
  </Body>
</HTML>
```

调用 ad.txt 文件显示的结果如图 5-19 所示，每次刷新浏览器的时候，显示的图片可能是不同的。



图 5-19 广告显示的内容

GetAdvertisement()方法需要一个参数，这个参数是一个文件，该文件包含有关广告图标的显示信息和连接信息以及显示权重，如程序 ad.txt 所示。

案例名称：广告的配置文件

程序名称：ad.txt

```
REDIRECT ad_redir.asp
WIDTH 226
HEIGHT 62
BORDER 0
*
gettop.gif
http://www.gettop.net
卓越信息
50
xdf.jpg
http://www.neworiental.cn
新东方学校
30
kxt.gif
http://www.kuaxuetang.com.cn
快学堂
20
```

两部分信息用*号分隔，第一部分是 4 个通用的参数，解释如下。

- (1) REDIRECT：当单击广告后，调用该文件进行处理。
- (2) WIDTH：指示该图标文件的宽度，默认值为 440。
- (3) HEIGHT：指示图标文件的高度，默认值为 60。
- (4) BORDER：广告图标文件的边界厚度，默认值为 0。

在 ad.txt 文件中，AD_REDIREC 参数指示重定向文件为 ad_redir.asp。而 Width 和 Height 参数分别为 226 和 62 像素，图标边界设为 0。

第二部分是针对每一个广告图标的信息。对于每一个广告图标，有如下 4 行信息。

- (1) 第一行是该图标的文件名及其位置，文件可以在当前服务器上，也可以在互联网的其他位置。
- (2) 第二行是该广告的连接位置。
- (3) 第三行设置的是鼠标移动到图标上时，显示的提示信息。
- (4) 第四行限定了广告图标的显示频率，例如在这个例子中，三个图标被显示的概率依次为 50%，30%，20%，当然概率越高，显示次数就会越多。

当用户单击了广告图标后，就进行文件处理，该文件主要功能是转到广告图标指向的站点。如程序 ad_redir.asp 所示。

案例名称：广告的转向文件

程序名称：ad_redir.asp


```
<%@ Language=Jscript %>
<%
    Response.Redirect(Request("url"));
%>
```

首先利用 Request 对象得到要转向的 URL，然后利用 Redirect 方法转到该网站。

小结

本章重点理解文件和广告组件的使用方法。掌握操作文件内容、操作文件和操作文件夹的方法，重点理解文件版留言簿的原理。了解广告组件的概念及其广告配置文件的含义，会使用广告组件。

课后习题和上机练习

1. 如何使用文件组件？文件组件提供哪些功能？
2. 如何向已经存在的文件中追加内容？
3. 打开文件有哪几种方式，有哪些参数？各是什么意思？
4. 如何利用复制、删除和赋值文件？
5. 广告组件的配置文件的功能是什么？
6. 改写案例 5-2，添加留言的表情，提供下拉列表框（笑着说和苦着说）供用户选择。（上机完成）

第 6 章 在 ASP 中使用外置组件

本章要点

本章主要介绍外置组件的基本概念、如何注册外部组件，如何在程序中调用。着重介绍两个网站常用的外置组件：文件上传和 E-mail 组件。为了加深对组件的理解，最后利用 VB 6.0 编写一个组件，并编写 ASP 文件调用。

5.1 利用 ASP 的外部组件

ASP 之所以功能强大，主要因为它可以调用外部组件。

5.1.1 组件概述

组件通过指定的接口函数提供一些功能。可以把组件理解为一种程序，通过调用这种程序，实现在 ASP 程序中无法实现或者很难实现的功能，组件提供一种很好的代码重用的方法。

可以利用 ASP 设计制作动态、交互的 Web 页面，但是会发现 ASP 在某些方面功能不强，甚至如果不借助服务器端组件就很难实现某些功能，例如文件上传、数据库操作，邮件功能，文件系统操作等。但是幸运的是可以找到很多组件来提高 ASP 的编程应用能力，

5.1.2 组件的调用方法

服务器组件和 ASP 内置对象不同，不能直接使用，必须首先被实例化。利用 Server 对象的 CreateObject 方法创建一个对象并返回这个对象的引用。语法如下：

```
var objVar = Server.CreateObject(Class)
```

Class 表示创建什么样的对象，Class 的格式一般为：

工程名.类名

例如：var MyFileObject = Server.CreateObject("Scripting.FileSystemObject");中，工程名是 Scripting，类名是 FileSystemObject。

5.2 实现文件上传

文件上传的组件很多，笔者手中就有三个上传的组件，其中 LyfUpload 功能比较全，而且使用起来比较方便。在此向组件的作者表示最深刻的敬意！

5.2.1 上传组件简介

LyfUpload 是一个免费的 ASP 组件，可以在 ASP 页面中接收客户端浏览器使用 `encType= "multipart/form-data"` 的 Form 上传的文件。该组件是一个 DLL（Dynamic Link Library，动态链接库）文件，名称为 `lyfUpload.dll`，如图 6-1 所示。



图 6-1 上传组件

该组件支持单文件上传、多文件上传、限制文件大小上传、限制某一类型文件上传、及文件上传重命名等功能。

上传组件的功能如下。

- (1) 支持上传多个文件。
- (2) 可以将上传的文件改名保存。
- (3) 可以同时使用其他的 form 元素的信息。
- (4) 可以限制上传文件的大小。
- (5) 支持限制文件上传的类型。
- (6) 可以得到上传文件的大小。
- (7) 可以得到上传文件的类型，如 gif 文件为 `images/gif`。
- (8) 本版本完全免费，没有任何限制。

ASP 要使用一个外置组件，必须首先在服务器上注册。将 `lyfUpload.dll` 文件复制到系统盘 WINNT 目录的 `system32` 目录下，单击“开始”菜单，打开“运行”窗口，如图 6-2 所示。



图 6-2 打开运行窗口

在运行窗口中输入“`regsvr32 lyfupload.dll`”，注意必须将 `lyfupload.dll` 文件复制到 `system32` 目录下，否则就找不到。如图 6-3 所示。



图 6-3 注册组件

注册成功后弹出对话框，说明注册成功，如图 6-4 所示。



图 6-4 注册成功

注册完以后，就可以在 ASP 程序调用使用。

5.2.2 组件提供的方法

组件利用提供如下方法。

1. Request 方法

- (1) 功能：得到提交页面中表单元素的值。
- (2) 声明：Public Function Request(nm As String)。
- (3) 返回值：元素的值，字符串类型。

2. FileType 方法

- (1) 功能：得到上传文件的 Content-Type。
- (2) 声明：Public Function FileType(strTag As String)。
- (3) 参数介绍：strTag 为 Form 中文件元素的名字，如“<input type="file" name="myfile">”中的 name 属性 myfile。
- (4) 返回值：文件上传成功，返回文件的 Content-Type 类型，不成功，返回空字符串。

3. SaveFile 方法

- (1) 功能：上传客户端选择的文件。
- (2) 声明：SaveFile(strTag As String, strPath As String, strway as boolean, Optional DestFileName As String) As String。
- (3) 参数介绍：
 - ① strTag 为 Form 中文件元素的名字，如"myfile"。
 - ② strPath 为要文件保存在服务器的目录。
 - ③ strway 为上传文件方式，覆盖方式上传为 true，不覆盖上传为 false。
 - ④ DestFileName(可选参数)，代表文件上传后重命名保存的名字。
- (4) 返回值：
 - ① 成功，返回上传的文件的名字；
 - ② 不成功，如果上传失败，返回为空字符串；
 - ③ 不成功，如果上传文件后缀不对，返回为"0"（当设置了 extName 属性时有效）；
 - ④ 不成功，如果上传文件的大小太大，返回为"1"（当设置了 MaxSize 属性时有效）；
 - ⑤ 不成功，如果上传文件同服务器上已有文件相同，返回为"2"（当设置了参数 strway 为 false 时有效）。

4. About 方法

About 方法显示 LyfUpload 组件的作者及版本号等信息调用，如下所示。

```
<%  
var ss = Server.CreateObject("LyfUpload.UploadFile");//创建 LyfUpload 组件对象  
Response.Write(ss.About());  
%>
```

5.2.3 组件提供的属性

组件提供如下属性。

1. ExtName 属性

(1) 功能：限制上传文件的类型。

(2) 调用方法如下。

```
var obj = Server.CreateObject("LyfUpload.UploadFile");  
obj.extname="gif"; //设置文件上传只能是 gif 文件  
obj.extname="gif,jpg,bmp"; //多文件类型请用","隔开
```

2. MaxSize 属性

(1) 功能：限制上传文件的大小。

(2) 调用方法如下。

```
var obj = Server.CreateObject("LyfUpload.UploadFile");  
obj.MaxSize = 2048; //设置文件上传的最大为 2048 个字节(2K)
```

3. FileSize 属性

(1) 功能：得到上传文件的大小。

(2) 调用方法如下。

```
var obj = Server.CreateObject("LyfUpload.UploadFile");  
Response.Write( obj.FileSize);
```

案例 6-1：文件上传

案例实现的功能是将客户端的文件上传到 ASP 所在路径下的 Files 目录中。首先是文件选择程序，如程序 FileUpload.htm 所示。

案例名称：文件上传输入界面

程序名称：FileUpload.htm

```
<HTML>  
  <HEAD>  
  </HEAD>  
  <BODY>  
    <FORM METHOD="POST" ENCTYPE="MULTIPART/FORM-DATA"  
      ACTION="FileUpload.asp">  
      文本框 1: <INPUT TYPE="TEXT" NAME="text1" SIZE="20"><BR>  
      选择文件: <INPUT TYPE="FILE" NAME="file1"><BR>  
      <INPUT TYPE="SUBMIT" VALUE="上传">  
    </FORM>
```

```
</BODY>
```

```
</HTML>
```

表单属性中一定要包含 `enctype="multipart/form-data"` 语句, 此时处理的文件是一种 Form 数据, 所以一定要加上这条语句, 否则文件不能上传。程序中 “`<INPUT TYPE="FILE" NAME="file1">`” 文件框的名字是 file1。将程序放到网站的某个目录下, 程序显示如图 6-5 所示。

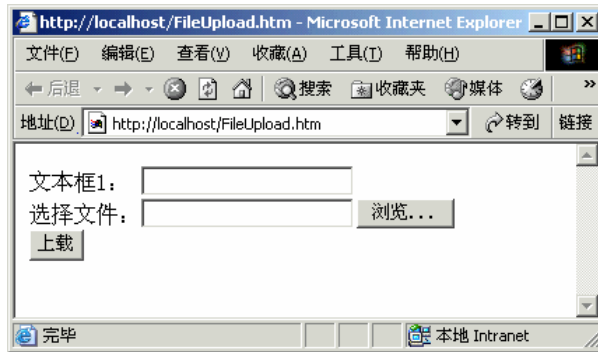


图 6-5 输入界面

当单击“提交”按钮时, 调用程序 FileUpload.asp 文件处理, 如下所示。

案例名称: 文件上传处理程序

程序名称: FileUpload.htm

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    var obj = Server.CreateObject("LyfUpload.UploadFile");
    txt = obj.Request("text1") ;//得到 form 元素的值
    Response.Write( "文本框 1 的输入值是: " + txt) ;
    Response.Write("<br>");
    //保存文件到 ASP 所在路径的 Files 目录下, file1 为 Form 表单中文件框的名字
    strFileName = obj.SaveFile("file1", Server.MapPath("Files"),true);
    //得到文件类型
    strFileType = obj.FileType("file1");
    if (strFileName != "")
    {
        Response.Write("选择的文件已经上载到服务器! ");
        Response.Write("<br>文件名: " + strFileName)
        Response.Write("<br>文件类型: " + strFileType)
        Response.Write("<br>文件大小: " + obj.FileSize)
        //Response.Write( obj.About())
    }
}
```

```
}  
%>  
</BODY >  
</HTML>
```

为什么在输入界面上加上文本框 1 呢？其实没有什么作用，当上传文件的时候，如果处理页面要读取文本框的内容的话，不可以直接使用 `Request("文本框的名称")`，而只能使用组件提供的方法 `Obj.Request("文本框名称")`。

程序首先创建一个上传组件的一个对象实例，然后使用语句“`strFileName = obj.SaveFile("file1", Server.MapPath("Files"),true);`”将上一页 file1 文件框中选择的文件上传到服务器当前目录的 Files 目录下，true 的意思是如果有同名文件，就覆盖该文件。

首先选择一个文件，如图 6-6 所示。

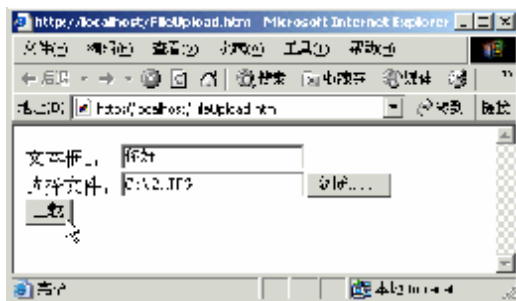


图 6-6 选择文件上传

上传成功以后，返回上传文件信息，如图 6-7 所示。

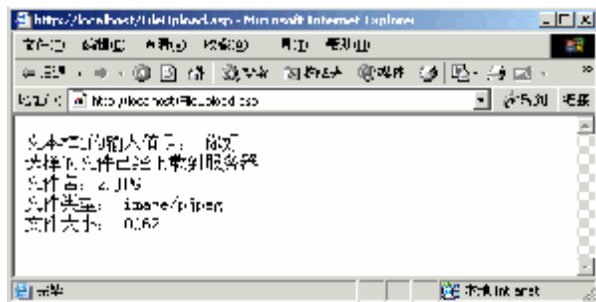


图 6-7 返回文件信息

这时可以查看 Files 目录，该文件已经上传。在网站应用中通常使用该组件上传图片，比如 BBS 和报名等系统的上传图片功能，都可以通过该组件实现。

5.3 E-mail 组件

网上贺卡非常流行，利用 ASP 就可以发送网上贺卡。程序发送 E-mail 的另一个应用领域是：会员信息的传送。在一个网站注册后，会收到一封系统自动发送的信，ASP 可以实现此类功能。这里介绍利用 Jmail 组件实现 E-mail 的发送。

W3 JMail 是一个发送邮件的组件，支持 HTML 格式的邮件，最新的版本可以从 <http://www.dimac.net> 下载。W3 JMail 发送邮件速度快、功能丰富并且是免费的。Jmail 组件使用方法如下。

```
var msg = Server.CreateObject( "JMail.Message" );
```

不管用哪个外置组件首先都必须创建一个基于该组件的对象，然后才能调用该组件提供的功能。

案例 6-2：主页发送 E-mail

首先安装 Jmail 组件，安装完以后自动注册到服务器上。Jmail 发送邮件的使用方法如程序 SendMail.asp 所示。

案例名称：主页发送 E-mail

程序名称：SendMail.asp

```
<%@ Language=Jscript %>
<%
    //创建 JMail 对象
    var msg = Server.CreateObject( "JMail.Message" );
    //启用日志记录
    msg.Logging = true;
    //设置邮件的编码
    msg.Charset = "gb2312";
    //邮件的发送人和姓名
    msg.From = "发送人邮箱地址";
    msg.FromName = "发送人姓名";
    //添加邮件接收人
    msg.AddRecipient( "接收人的 E-mail 地址", "接收人姓名" );
    //邮件主题
    msg.Subject = "How are you?";
    msg.Body = "Mail Test";
    if ( ! msg.Send( "用户名:密码@邮箱的 SmtP 地址" ) )
    {
        Response.Write("<pre>" & msg.log & "</pre>");
    }
    else
    {
        Response.Write("邮件发送成功");
    }
%>
```

这是本书惟一需要修改才可以执行的程序。需要修改的地方有如下 4 个。

(1) 修改 “msg.From = “发送人邮箱地址”；”语句，填入自己的邮件地址，如 “msg.From

= "zhangsan@mail.com";”。

(2) 修改 “msg.FromName = "发送人姓名";” 语句，填入自己的大名，如 “msg.FromName = "zhourunfa";”。

(3) 修改 “msg.AddRecipient("接收人的 E-mail 地址", "接收人姓名");” 语句，填入自己要发送的地址，如 “msg.AddRecipient("zhangsan@mail.com", "zhangsan");”。

(4) 修改 “if (! msg.Send("用户名:密码@邮箱的 SmtP 地址"))”，用户名和密码是发送人邮箱的用户名和密码，注意中间用分号隔开，@ 符号后面是邮件的 SMTP 地址，现在发送邮件的服务器都需要验证是不是本系统的用户，这条语句最关键。

比如现在张三有一个邮箱是 zhangsan@163.net，邮箱密码是 mamahao，现在他要给周润发发邮件，周润发的邮箱是 zhourunfa@163.com。改完的四句话如下面的代码段。

```
msg.From = "zhangsan@163.net";
msg.FromName = "zhangsan";
//添加邮件接收人
msg.AddRecipient("zhourunfa@163.com", "zhourunfa");
//邮件主题
msg.Subject = "How are you?";
msg.Body = "Mail Test";
if (! msg.Send( "zhangsan:mamahao@smtp.163.net" ))
```

其中 smtp.163.net 是 163.net 提供的发送邮件服务器地址，一般都可以在网站上查到自己邮箱的 SMTP 地址，表 6-1 列出常用邮箱的 SMTP 地址。

表 6-1 常见邮箱的 SMTP 地址

邮 箱 后 缀	SMTP 地址
21cn.com	smtp.21cn.com
sina.com	smtp.sina.com.cn
163.net	smtp.163.net
163.com	smtp.163.com
fin365.com	mail.fin365.com
263.net	smtp.263.net

填写好信息后，就可以发送了。如果参数都正确的话，系统就出现信息发送成功界面，如图 6-8 所示。

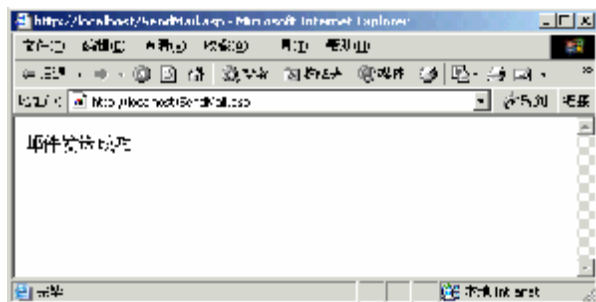


图 6-8 发送成功返回信息

拨号上网的用户可以高枕无忧地使用，如果在局域网中利用代理上网，可能会出现一些问题。为了确认发送是否真的成功，查看目标邮箱，发现果然出现了许多垃圾邮件，说明 Jmail

组件成功发送了邮件，如图 6-9 所示。

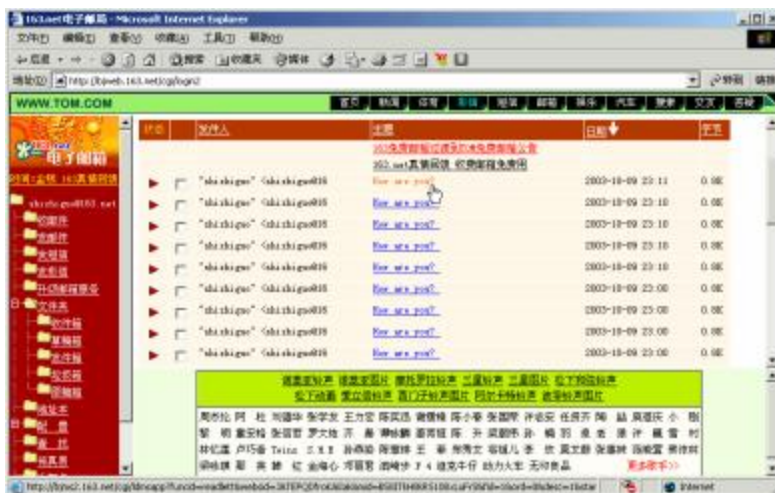


图 6-9 收到的邮件

看完信的标题，可以看看邮件的内容，所有的信息都传过来了。如图 6-10 所示。



图 6-10 邮件内容

5.4 自己编写组件

现在从事 ASP 编程的开发人员一般都要编写自己的组件，为了满足这种工作需要和深入了解 ASP 的组件，现在通过实例自己动手编写一个组件。虽然比较简单，但是可以加深对组件的理解。

案例 6-3：利用 VB 6.0 写服务器端组件

第一步：打开 VB 6.0，选择新建工程中的 ActiveX DLL，单击“打开”按钮，如图 6-11 所示。



图 6-11 新建工程

第二步：出现工程代码输入界面，右击工程窗口右边的“工程 1”，选择“工程 1 属性”如图 6-12 所示。

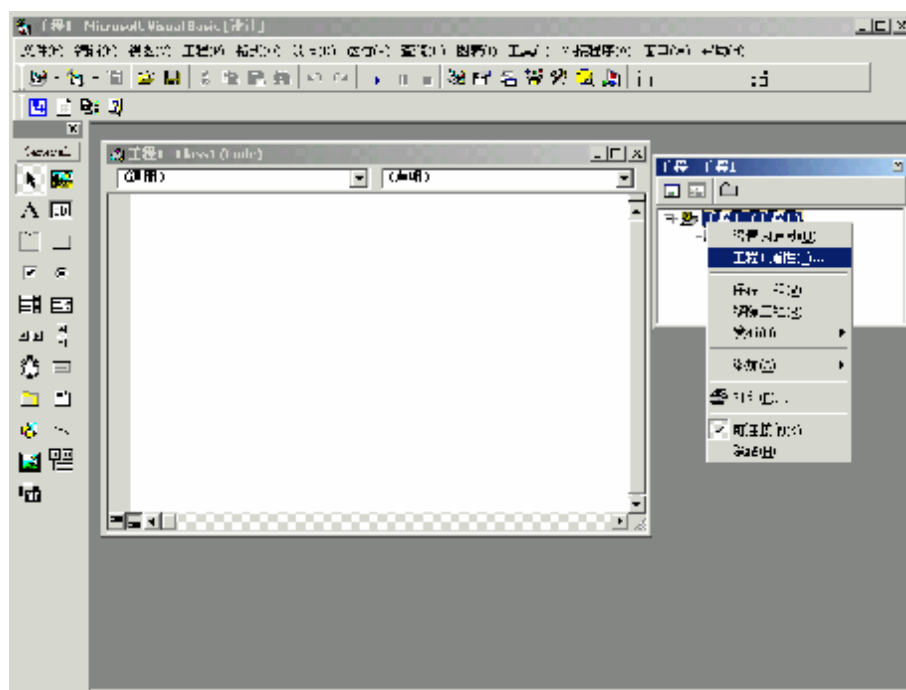


图 6-12 选择工程一

第三步：出现“工程属性”对话框，修改工程名为：aaa，如图 6-13 所示。

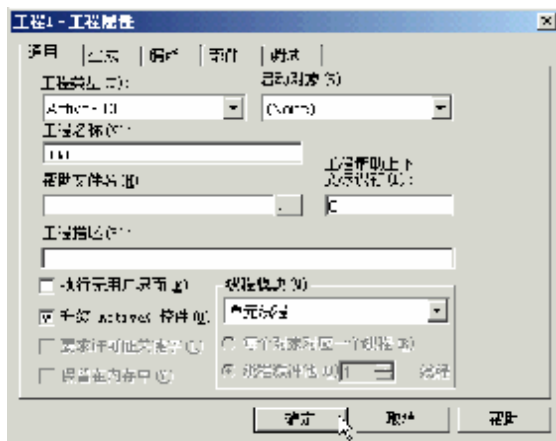


图 6-13 修改工程名

这里要修改的工程名称可以是任何名称，还记得在调用 Jmail 组件时，使用“var msg = server.CreateObject("JMail.Message")”，在 Server.CreateObject 后面的括弧中有 JMail.Message，如果在上图中将工程命名为相当于这里的 Jmail，这里将工程命名为 aaa。

第四步：工程名称 aaa 下面有一个叫 Class1 的图标，右击选择“属性窗口”，如图 6-14 所示。



图 6-14 修改类名

第五步：出现“属性”窗口，修改类名为 bbb，如图 6-15 所示。

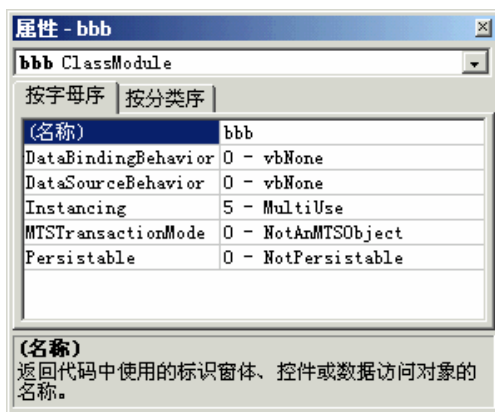


图 6-15 修改类名

根据上面的规则，创建该组件使用的语法是：Server.CreateObject(aaa.bbb)，下面开始编写

处理函数。

第六步：双击类名 bbb，在出现的窗口中编写处理函数，该函数实现的功能是输入一个整数，返回该数的平方。处理函数如程序 bbb.cls 所示。

案例名称：编写组件方法

程序名称：bbb.cls

```
Public Function ccc(ddd As Integer)
    ccc = ddd * ddd
End Function
```

程序执行结果如图 6-16 所示。

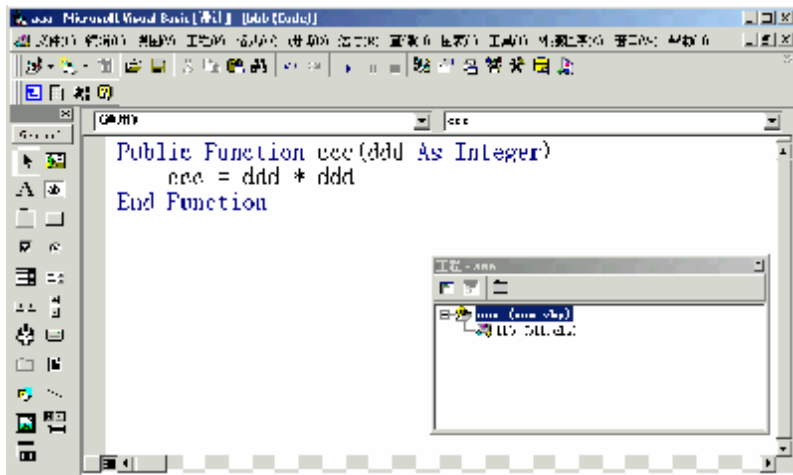


图 6-16 编写处理函数

在 Basic 语法体系中函数的返回值是通过给函数名赋值实现的。如语句“ccc = ddd * ddd”中，ccc 是函数名。这条语句相当于 JavaScript 中的语句“return ddd * ddd;”。

第七步：编译程序，选择菜单栏“文件”中的“生成 aaa.dll”菜单项，如图 6-17 所示。

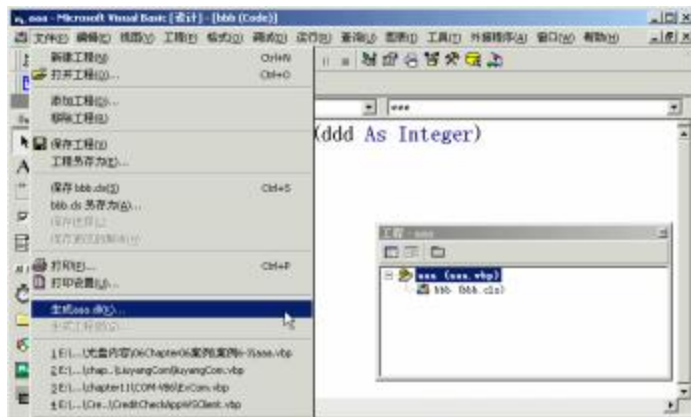


图 6-17 编译组件

新建一个目录，将工程保存其中，保存完毕后，就会有如图 6-18 所示的 DLL 文件。

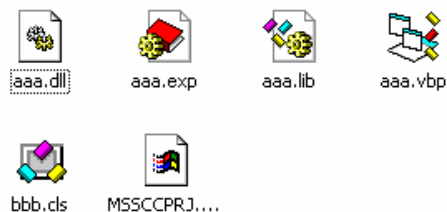


图 6-18 编译成 DLL 文件

第八步:注册 DLL 文件,将生成的 aaa.dll 文件复制到 WINNT 目录下的 System32 目录下,然后在运行窗口中执行 Regsvr32 aaa.dll 命令。(这步可以省略,因为 VB 在编译的过程已经自动注册了。)

第九步:现在开始编写 ASP 代码,通过该代码调用刚才编写的 DLL 文件。创建一个 ASP 文件的内容如程序 dll.asp 所示。

案例名称:调用自己编写的 DLL 文件

程序名称: dll.asp

```
<%@ Language=Jscript %>
<%
    var kkkk = Server.CreateObject( "aaa.bbb" );
    Response.Write(kkkk.ccc(10) );
%>
```

程序中, aaa.bbb 分别是工程名和类名, kkkk 是随便起的实例名, ccc 方法是 bbb 提供的, 该方法有一个输入值, 返回该数的平方, 这里输入值为 10, 应该返回 100, 执行的结果如图 6-19 所示。



图 6-19 调用自己编写的组件

到此为止, 全部的开发工作结束。案例虽然简单, 但是任何复杂的组件都可通过这样的方法开发出来。

小结

本章重点理解组件的基本概念、如何注册和调用一个组件。掌握文件上传组件常用的方法和属性，掌握 Jmail 组件的使用方法和如何使用 VB 6.0 编写一个组件以及如何在 ASP 文件周用。

课后习题和上机练习

1. 内置组件和外置组件有什么区别？
2. 得到一个新的外置组件，如何在服务器上注册？
3. 文件上传组件有哪些属性和方法？功能是什么？
4. 文件上传程序的文件输入表单和一般的表单有什么区别？
5. 新建一个 Jmail 的 Message 对象为 msg，语句“msg.Send("用户名:密码@邮箱的 Smtplib 地址")”中的用户名、密码和 SMTP 的地址分别是什么含义？如果某人的邮箱是 isi@fm365.com，密码是 aaabbbccc，如何设置这三个参数？
6. 根据自己的情况，改编案例 6-2 中的程序，使之能发送邮件。（上机练习）
7. 编写一个组件，包含一个方法，该方法实现求一个输入值的立方，方法名为 zzz。该组件调用的方法必须是：“Server.CreateObject("abc.def");”。（上机练习）

第三部分 ASP 操作数据库

第 7 章 ADO 数据访问接口

本章要点

本章介绍 ADO 的基本概念，以及如何在 ASP 程序中使用 ADO 的对象。介绍 ADO 的对象 Connection, RecordSet 和 Command 的使用。介绍 SQL 语句的基本概念及如何利用 SQL 语句操作数据库。介绍访问数据库的三个基本格式。

7.1 ADO 概述

使用 ADO (ActiveX Data Object, ActiveX 数据对象)，可以对几乎所有数据库进行读取和写入操作。可以使用 ADO 来访问 Microsoft Access, Microsoft SQL Server 和 Oracle 等数据库。

ADO 常用的四种对象及其功能如下。

- (1) 连接对象 (Connection): 用来连接数据库。
- (2) 记录集对象 (RecordSet): 用来保存查询语句返回的结果。
- (3) 命令对象 (Command): 用来执行 SQL (Structured Query Language) 语句或者 SQL server 的存储过程。
- (4) 参数对象 (Parameter): 用来为存储过程或查询提供参数。

下面介绍常用的连接对象 (Connection)、记录集对象 (RecordSet)、命令对象 (Command)。

7.2 Connection 数据对象

与数据库的所有通信都通过一个打开的 Connection 对象进行。对一个数据库进行数据的插入和读取之前，必须先打开与这个数据库的连接。

如何打开和关闭一个数据库连接？首先创建一个 Access 数据库表，打开 Access2000/XP，选择“新建数据库”，数据库命名为：“person.mdb”，出现如图 7-1 所示界面。

单击“使用设计器创建表”出现输入界面，在其中输入表的结构，如图 7-2 所示。

输入完以上的信息后，单击保存图标，并命名为“grade”，右击刚创建的“grade”表，选择“打开”，将产生一个空表，如图 7-3 所示。

在出现的界面中输入人员的信息，如图 7-4 所示。

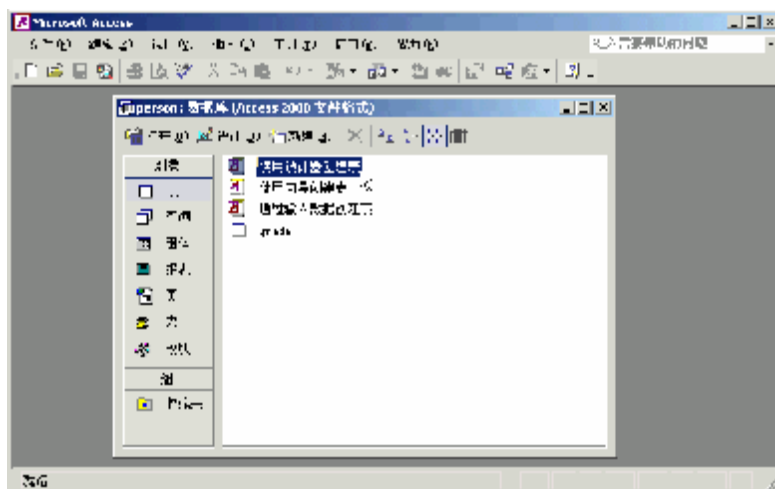


图 7-1 新建数据库表

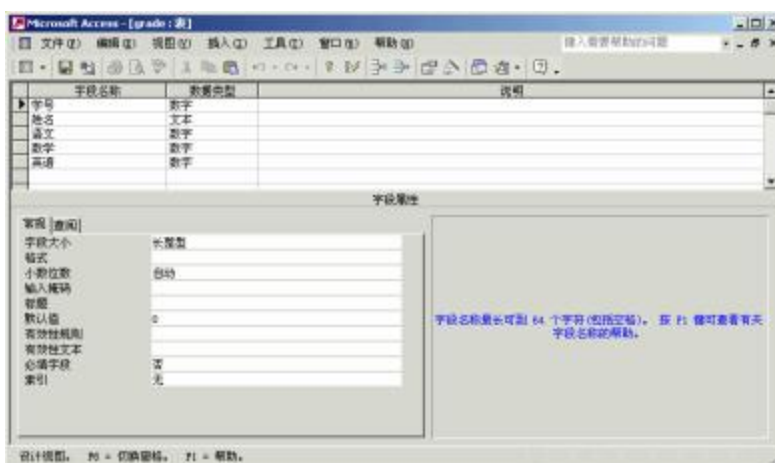


图 7-2 输入数据字段名

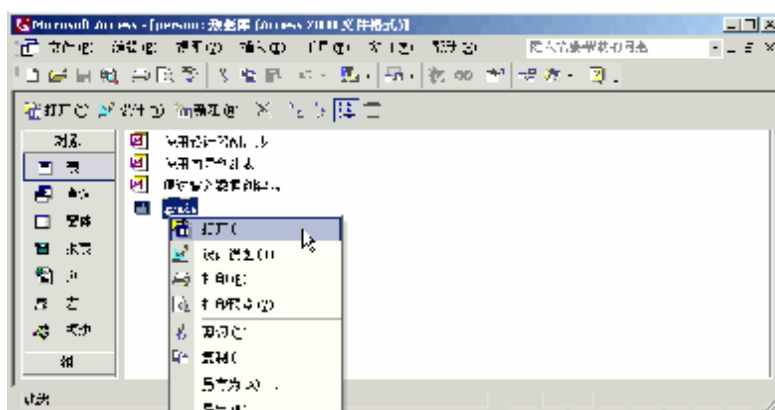


图 7-3 产生一个空表

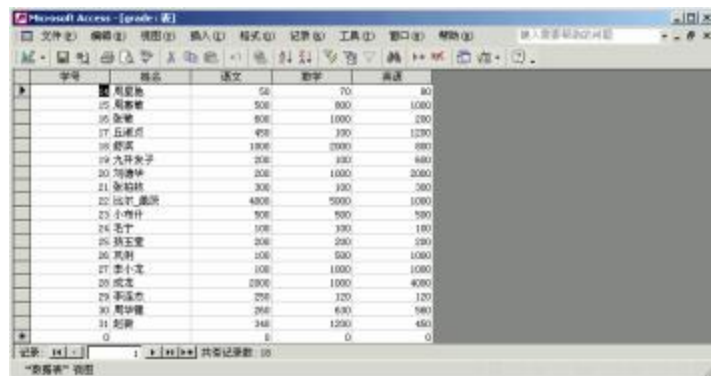


图 7-4 输入数据库表的内容

输入完毕后，保存关闭窗口，表命名为“grade”。下面对其结构进行简要的分析。

假定建立了一个 RecordSet 对象的实例 rs，rs 的内部结构和数据库表一样具有二维结构。s 对象包含一个无形的指针，默认的情况下指向第一行，结构如图 7-5 所示。

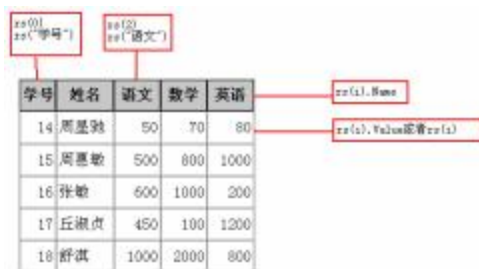


图 7-5 关系型数据库表与 RecordSet 对象

其中：写法 rs(0)、rs(“学号”)、rs.Fields(0)和 rs.Fields(“学号”)是相同的，写法 rs(2)、rs(“语文”)、rs.Fields(2)和 rs.Fields(“语文”)是相同的。其余类推。

根据上表和注释：

```
rs(3).Name = rs.Fields(3).Name = "数学";
```

```
rs(4).Name="英语";
```

假设当前的记录指向第三条，那么

```
rs(1).Value=rs("姓名")=rs(1)= "张敏"
```

```
rs(4).Value=rs(4)=rs("英语")=200
```

有了这些基础，就可以进行数据库操作。

7.2.1 打开和关闭数据库连接

要建立一个数据库的连接，首先创建 Connection 对象的一个实例，然后调用 Connection 对象的 Open 方法打开一个连接，首先通过程序 7-01.asp 输出数据库的表头。

案例名称：输出数据库的表头

程序名称：7-01.asp

```
<%@ Language=Jscript %>
```

```

<HTML>
  <BODY>
    <UL>
      <%
        var conn = Server.CreateObject("ADODB.Connection");
        conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
          Server.MapPath("person.mdb"));
        rs = conn.Execute("select * from grade");
        for (i=0; i<rs.Fields.Count; i++)
        {
          Response.Write("<LI>" + rs(i).Name);
        }
        conn.close();
      %>
    </UL>
  </BODY>
</HTML>

```

显示的结果如图 7-6 所示。

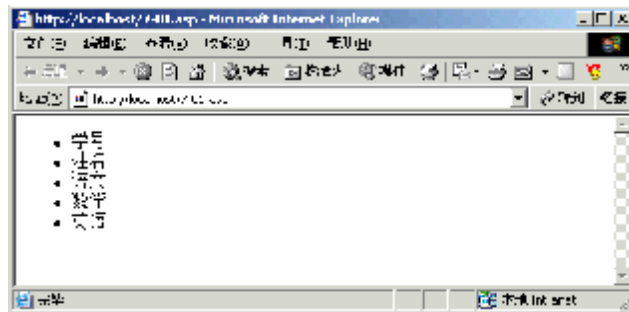


图 7-6 输出数据表的表头

在解释程序之前，首先介绍一个通用的打开数据库的格式。

格式一：数据库调用的基本格式

格式说明：利用 Execute 方法建立 RecordSet 对象

//第一步：建立 Connection 对象

```
var conn = Server.CreateObject("ADODB.Connection");
```

//第二步：使用 Connection 对象的 Open 方法建立数据库连接

```
conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
  Server.MapPath("Access 数据库"));
```

//第三步：使用 Connection 对象的 Execute 方法执行 SQL 语句

//如果执行查询语句

```
rs = conn.Execute("数据查询语句");
```

```
//如果执行数据操纵语句
```

```
conn.Execute( "数据操纵语句" );
```

本章总共将介绍连接数据库的三大基本格式。格式一的特点是：**RecordSet** 对象是利用 **Connection** 对象的 **Execute** 方法建立的，建立的 **rs** 对象的指针只能向后，不能向前移动。第一步建立了 **Connection** 对象的一个实例，第二步利用 **Open** 方法和用 **Access** 数据驱动程序打开服务器上的某个 **Access** 数据库，第三步执行 **SQL** 语句。

程序 7-01.asp 中利用 **conn** 的 **Open** 方法打开当前目录下的 **person.mdb** 文件，然后执行一个 **SQL** 语句“**Select * from grade**”，其中*表示所有列，**grade** 是数据库表名。“**rs.Fields.Count**”返回数据库表的列数。

7.2.2 向浏览器输出数据库内容

已经成功地向浏览器输出了表头后，再输出内容就比较容易，如程序 7-02.asp 所示。

案例名称：输出第一条记录

程序名称：7-02.asp

```
<%@ Language=Jscript %>
<HTML>
  <BODY>
    <UL>
      <%
        var conn = Server.CreateObject("ADODB.Connection");
        conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
          Server.MapPath("person.mdb"));
        rs = conn.Execute( "select * from grade" );
        for (i=0; i<rs.Fields.Count; i++)
        {
          Response.Write ("<LI>" + rs(i).Name + " = " + rs(i));
        }
        conn.close();
      %>
    </UL>
  </BODY>
</HTML>
```

显示的内容如图 7-7 所示。

到此必须建立一个概念：**RecordSet** 对象有一个无形的指针。关系型数据库表是一张二维表，每一行代表一个实体的信息。**RecordSet** 指针可以指向不同的行，当数据库第一次打开时，这个指针定位在第一行，当需要向后移动的时候，可以用 **movenext()**来移动这个指针。这时已经实现向浏览器输出数据库表的第一条内容。

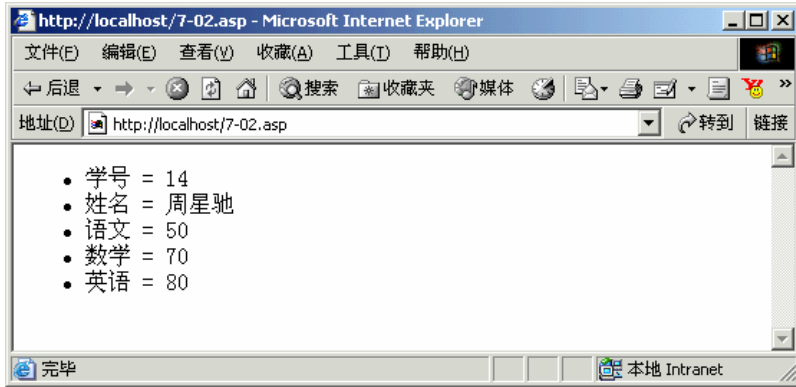


图 7-7 输出第一行

7.2.3 以表格的形式输出

下面用一个循环输出数据库中的所有数据，遇到一个陌生的语法是：`rs.movenext()`。当数据库打开时，`rs` 对象定位在数据库表的第一条记录上，输出第一条记录的内容；要想输出第二条记录，必须执行 `rs.movenext()` 指令，让它移动到下一条记录。

其次还要介绍的语法是：`rs.Bof` (Begin of File: 文件开头) 和 `rs.Eof` (end of File: 文件结尾)，这两条指令判断记录指针是否移动最前面和最后面，理解它的一个关键的地方是：`Bof` 的位置是在第一条记录之前，`Eof` 是在最后一条记录之后。如果 `rs` 指针在最后一条记录上，再执行一次 `movenext` 时，`rs.Eof` 为真。一般不能让 `rs.Eof` 或者 `rs.Bof` 为真，因为这时，读取数据会出错。

案例名称：以表格的形式输出

程序名称：7-03.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    var conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = conn.Execute("Select * from grade");

    Response.write("<table border=1>");
    Response.write("<tr>");
    //输出表头
    for (i=0; i<rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs(i).Name + "</td>");
```

```
}  
Response.write ("</tr>");  
  
//输出表内容  
while (!rs.EOF)  
{  
    Response.write ("<tr>");  
    for (i=0; i<rs.Fields.Count; i++)  
    {  
        Response.Write ("<td>" + rs(i) + "</td>");  
    }  
    Response.write ("</tr>");  
    rs.movenext();  
}  
Response.write ("</table>");  
conn.close();  
%>  
</BODY>  
</HTML>
```

执行上面的程序，程序以表格的形式输出到浏览器上，程序实现一个循环将所有的数据都显示出来，每次循环让 RecordSet 对象向下移动一次，直到移动到最后一记录为止，所以显示出来的是全部数据，而且以表格的形式输出到浏览器上，如图 7-8 所示。

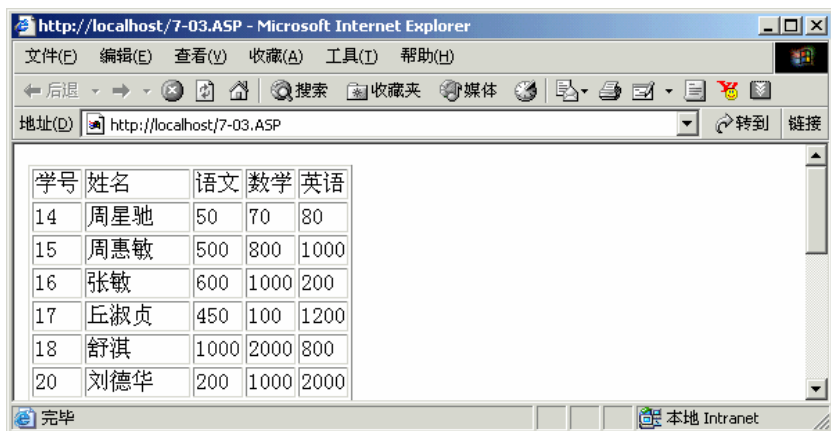


图 7-8 表格输出

可以将输出的程序写成一个子过程的形式，便于以后的调用。

案例名称：编写输出函数

程序名称：7-04.asp

```
<%@ Language=Jscript %>
```

```

<HTML>
<BODY>
<%
    var conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = conn.Execute( "Select * from grade" );
    rstotab(rs);
    conn.close();
%>

<%
function rstotab(rs)
{
    Response.write("<table border=1>");
    Response.write("<tr>");
    //输出表头
    for (i=0; i<rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs(i).Name + "</td>");
    }
    Response.write("</tr>");

    //输出表内容
    while (!rs.EOF)
    {
        Response.write("<tr>");
        for (i=0; i<rs.Fields.Count; i++)
        {
            Response.Write("<td>" + rs(i) + "</td>");
        }
        Response.write("</tr>");
        rs.movenext();
    }
    Response.write("</table>");
}
%>
</BODY>
</HTML>

```

显示的结果依然将所有记录输出来。当要向浏览器输出信息时，只要包含这个函数然后

周用函数就可以了，可以提高编程效率。

7.3 使用 SQL 语句

SQL 语句主要包括数据查询语言（Data Query Language, DQL）和数据操作语言（Data Manipulation Language, DML）。首先介绍 DQL 的使用方法。

7.3.1 Select 的三大基本格式

1. 基本句型一：（最简单的 select 语句）

Select 字段名 From 数据表

测试句型如下。

（1）Select * From grade

功能说明：将 grade 表中的所有字段取出来。

（2）Select 学号,姓名 from grade

功能说明：将 grade 表中学号和姓名字段取出来。

（3）Select 学号,姓名,语文+数学+英语 as 总成绩 from grade

功能说明：将 grade 表中的学号和姓名取出来，并将语文、数学和英语成绩相加产生虚拟列总成绩。

2. 基本句型二：（使用条件查询）

Select 字段名 From 数据表 where 筛选条件

测试句型如下。

（1）Select * from grade where 数学>60

功能说明：把所有数学成绩大于 60 分的记录选出来。

（2）Select * from grade where 数学=300 or 语文=300

功能说明：把数学成绩等于 300 分或者语文成绩等于 300 分的人选出来。

3. 基本句型三：（进行排序）

Select 字段名 From 数据表 Order by 字段名

测试句型如下。

（1）Select * from grade order by 数学 注：从低到高排序

功能说明：从 grade 表中取出所有字段，并按数学成绩排序。

（2）Select * from grade order by 数学,语文

功能说明：从 grade 表中取出所有字段，并按数学成绩排序，如果数学成绩相同则按照语文成绩排序。

（3）Select * from grade order by 数学 desc 注：从高到低排序

功能说明：从 grade 表中取出所有字段，并按数学成绩倒序。

（4）Select top 5 * from 成绩单

功能说明：从 grade 表中取出前五条记录的所有字段。

测试上面的 SQL 语句，如程序 7-05.asp 所示。

案例名称：测试 SQL 语句

程序名称：7-05.asp

```
<%@ Language=Jscript %>
<HTML>

<BODY>
    <%
    var SQL = "";
    if (Request("SQL").Count > 0)
    {
        var conn = Server.CreateObject("ADODB.Connection");
        conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
            Server.MapPath("person.mdb"));
        SQL = Request("SQL")(1);
        rs = conn.Execute(SQL);
        rstotab(rs);
        conn.close();
    }
    %>

    <%
    function rstotab(rs)
    {
        Response.write("<table border=1>");
        Response.write("<tr>");
        //输出表头
        for (i=0; i<rs.Fields.Count; i++)
        {
            Response.Write("<td>" + rs(i).Name + "</td>");
        }
        Response.write("</tr>");

        //输出表内容
        while (!rs.EOF)
        {
            Response.write("<tr>");
            for (i=0; i<rs.Fields.Count; i++)
```

```
{
    Response.Write ("<td>" + rs(i) + "</td>");
}
Response.write ("</tr>");
rs.movenext();
}
Response.write ("</table>");
}
%>
<FORM Action="7-05.asp" Method=POST>
指令: <INPUT Type="Text" Name="SQL" Size="60" Value="<%=SQL%>"><P>
<INPUT Type=Submit Value=" 执行">
</FORM>
</BODY>
</HTML>
```

程序显示的结果如图 7-9 所示。

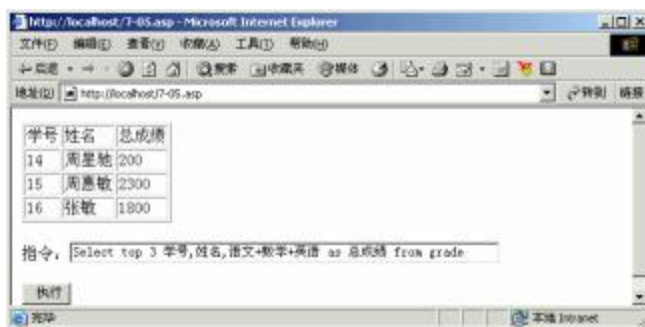


图 7-9 SQL 语句测试结果

可以在文本框内输入测试的 Select 语句。

7.3.2 Like 子句

在实际应用系统中，模糊查询用得比较多，比如在一些网站上经常会提供按照关键字查询，可以利用 Select 语句的 Like 子句方便地实现模糊查询。

首先分析一下 Like 子句的使用方法。

(1) 基本格式一：“_”匹配。

说明：每个下划线匹配一个任意字符，注意只匹配一个字符。比如：姓名 like '_敏'，匹配姓名以“敏”字结尾且字数等于二的所有数据记录，如：“张敏”。

(2) 基本格式二：“%”匹配。

比如：姓名 Like '%敏%', 匹配姓名中出现“敏”的所有数据记录，如：“周惠敏”，“于敏”、“敏大”、“敏二”等。

有了这个基础，就可以利用 Like 子句来实现网站的模糊查询系统了，比如要在数据库中

查询姓江的人，只要利用一条 SQL 语句就可以了。

```
Select * from 数据库表 where 姓名 Like '江%'
```

案例 7-1：模糊查询系统

本案例实现了网站的查询系统，当用户输入姓名的关键字的时候，系统就会从数据库中自动将所有的符合姓名中有这个字的人都显示出来，如程序 mb.asp 所示。

案例名称：模糊查询系统

程序名称：mb.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
    <%
        var Key = "";
        if (Request("keywords").Count > 0)
        {
            var conn = Server.CreateObject("ADODB.Connection");
            conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
                Server.MapPath("person.mdb"));
            Key = Request("keywords")(1);
            SQL = "Select * from grade where 姓名 like '%" + Key + "%'";
            Response.Write (SQL);
            rs = conn.Execute(SQL);
            rstotab(rs);
            conn.close();
        }
    %>
    <%
        function rstotab(rs)
        {
            Response.write("<table border=1>");
            Response.write("<tr>");
            //输出表头
            for (i=0; i<rs.Fields.Count; i++)
            {
                Response.Write("<td>" + rs(i).Name + "</td>");
            }
            Response.write("</tr>");
            //输出表内容
            while (!rs.EOF)
```

```

    {
        Response.write ("<tr>");
        for (i=0; i<rs.Fields.Count; i++)
        {
            Response.Write ("<td>" + rs(i) + "</td>");
        }
        Response.write ("</tr>");
        rs.movenext();
    }
    Response.write ("</table>");
}
%>
<FORM Action="mb.asp" Method=POST>
    输入姓名关键字:
    <INPUT Type="Text" Name="keywords" Size="60" Value="<%=Key%>"><P>
    <INPUT Type=Submit Value=" 执行 ">
</FORM>
</BODY>
</HTML>

```

程序中最关键的是如何将变量加到 SQL 语句中去。“SQL = "Select * from grade where 姓名 like '%" + Key + "%'" 语句中 Key 是变量, 要得到正确的格式只要按照下面的两个步骤进行操作。

(1) 写出正确的 SQL 语句, SQL = "Select * from grade where 姓名 like '%敏%'", 因为姓名是文本型变量, 所以必须加上单引号, 在 SQL 语句中, 用单引号表示字符型变量。

(2) 确定要替换的变量, 这里是要将“敏”替换成变量 Key。替换的规则是: 删除“敏”字, 在原字符串“敏”的位置, 首先加上两个双引号, 然后在两个双引号之间加上两个加号, 最后将变量加到两个加号中间。

按照两个步骤, 首先删除敏字, 加上两个引号得到 "Select * from grade where 姓名 like '%" + " + "%'", 然后在刚才加的两个双引号之间加上两个加号, 得到 "Select * from grade where 姓名 like '%" ++ "%'", 最后再将变量加到两个加号之间, 得到 "Select * from grade where 姓名 like '%" + Key + "%'。可以看出按照这两个步骤写出的 SQL 语句和程序中一样, 是正确的。程序执行的结果如图 7-10 所示。



图 7-10 模糊查询结果

7.3.3 使用 SQL 语句操作数据库

基本 SQL 语言分成三大体系。

- (1) 数据定义语言 DDL: 用来定义数据。
- (2) 数据查询语言 DQL: 数据检索语言。
- (3) 数据操作语言 DML: 包括 INSERT 语句、UPDATE 语句和 DELETE 语句。

SQL 语言的基本操作如下。

- (1) DELETE 指令: 删除数据记录。

基本语法:

```
DELETE FROM 数据表 WHERE 条件
```

例:

```
delete from grade where 数学=0
```

说明: 删除所有数学成绩为零的记录

- (2) UPDATE 指令: 更新数据记录。

基本语法:

```
UPDATE 数据表 SET 字段值=新值 where 条件
```

例 1:

```
update grade set 数学=数学+10 说明: 将 grade 表中所有人的成绩加 10 分
```

例 2:

```
update grade set 数学=100 where 姓名 like '%敏%'
```

说明: 将姓名中含有敏的人的数学成绩更新为 100 分

- (3) INSERT INTO 指令: 添加数据记录。

基本格式 1:

```
INSERT INTO 数据表 VALUES (字段新值)
```

基本格式 2:

```
INSERT INTO 数据表(字段一, 字段二, ……) VALUES (字段新值)
```

两种格式的区别是: 当 values 含有数据库表所有字段的值, 并且顺序和数据库字段一致时, 就可以省略数据库表后面的字段名称。

例 1:

```
Insert into grade(学号, 姓名, 数学)values (1234, '周润发',70)
```

例 2:

```
Insert into grade values(5678, '周润发',70,80,90)
```

说明: 该语句等价于:

```
Insert into grade(学号, 姓名, 语文, 数学, 英语) values(5678, '周润发',70,80,90)
```

因为包含数据库所有字段, 而且顺序和数据库一致, 所以字段列表可以省略。

测试以上 SQL 操作语言, 如程序 7-06.asp 所示。

案例名称: SQL 操作语言测试程序

程序名称: 7-06.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
    <%
        var SQL = "";
        if (Request("SQL").Count > 0)
        {
            var conn = Server.CreateObject("ADODB.Connection");
            conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
                Server.MapPath("person.mdb"));
            SQL = Request("SQL")(1);
            Response.Write(SQL);
            conn.Execute(SQL);
            conn.close();
        }
    %>
    <FORM Action="7-06.asp" Method=POST>
    指令: <INPUT Type="Text" Name="SQL" Size="60" Value="<%=SQL%>"><P>
    <INPUT Type="Submit" Value=" 执行">
    </FORM>
    <BR><A HREF="7-04.asp">查看</A>
</BODY>
</HTML>
```

可以看出这里依然利用格式一提供的语法,可以将 SQL 测试语句输入,然后单击“查看”,执行完毕后,可以查看执行后的情况。执行情况如图 7-11 所示。

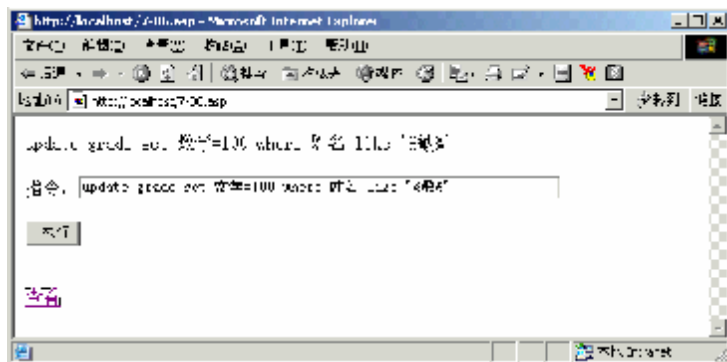


图 7-11 测试 SQL 操作语句

案例 7-2: 数据库版本留言簿

上一个版本利用文件实现留言簿,但利用数据库系统实现要比文件版本简单。首先设计留言簿所有的数据库,利用 Access2000/XP 新建一个名为 lyb.mdb 的数据库文件,新建 lyb

数据表，结构如图 7-12 所示。

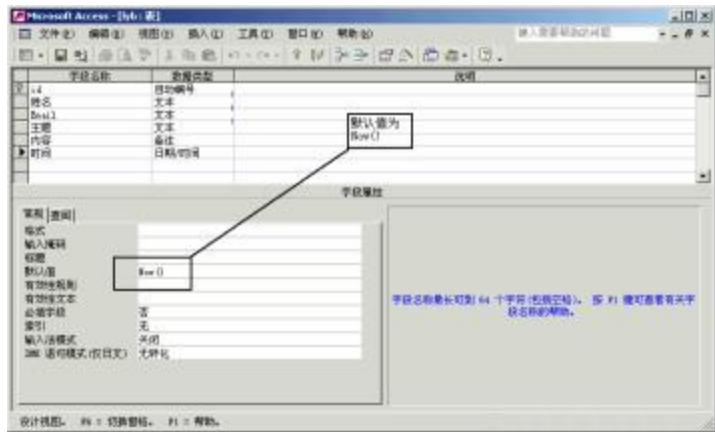


图 7-12 数据库表结构

注意：这里时间字段默认值为“Now()”，该默认值自动取系统的时间。该留言簿依然含有三个文件：input.htm，留言簿的输入界面；handle.asp，处理程序；display.asp，显示程序。input.htm 文件和文件版本的一样，这里就不列出了。当用户提交留言的时候调用程序 handle.asp 处理。如下所示。

案例名称：留言簿处理程序

程序名称：handle.asp

```
<%@ Language="Jscript"%>
<%
    Name    = Request("Name")(1);
    E-mail   = Request("E-mail")(1);
    Subject  = Request("Subject")(1);
    Memo     = Request("Memo")(1);

    if ("" == Name)
    {
        Response.Write("姓名不能为空白!");
        Response.End();
    }
    if ("" == E-mail)
    {
        Response.Write("E-mail 不能为空白!");
        Response.End();
    }
    if ("" == Subject)
    {
```

```

        Response.Write("主题不能为空白!");
        Response.End();
    }
    if (" " == Memo)
    {
        Response.Write("留言不能为空白!");
        Response.End();
    }
%>
<%
//插入到数据库
var conn = Server.CreateObject("ADODB.Connection");
conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
Server.MapPath("lyb.mdb"));
SQL = "insert into lyb(姓名,E-mail,主题,内容) values('" + Name + "','" +
E-mail + "','" + Subject + "','" + Memo + "')";
//Response.Write(SQL);
conn.Execute(SQL);
conn.close();
Response.Redirect("display.asp");
%>

```

程序依然读取留言信息，然后判断是否为空，如果不为空，使用 **Insert** 语句将信息插入到数据库中。最后转到显示界面 **display.asp** 文件，如程序 **display.asp** 所示。

案例名称：留言簿显示程序

程序名称：display.asp

```

<%@ Language="Jscript"%>
<%
//连接数据库，做查询
var conn = Server.CreateObject("ADODB.Connection")
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("lyb.mdb"));
sql = "select * from lyb order by 时间 desc";
rs = conn.Execute( sql )
%>
<%
//按照样式输出
while(!rs.EOF)
{
%>

```



```

<table border="0" bordercolor="#111111">
  <tr>
    <td><b><font size="4" color="#008080">姓名: </font></b></td>
    <td><%=rs("姓名")%></td>
  </tr>
  <tr>
    <td><b><font size="4" color="#008080">
      E-mail:</font></b></td>
    <td><a href="mailto:<%=rs("E-mail")%>">
      <%=rs("E-mail")%></a></td>
  </tr>
  <tr>
    <td><b><font size="4" color="#008080">主题: </font></b></td>
    <td width="542" height="17"><%=rs("主题")%></td>
  </tr>
  <tr>
    <td><b><font size="4" color="#008080">时间: </font></b></td>
    <td><%=rs("时间")%></td>
  </tr>
  <tr>
    <td><b><font size="4" color="#008080">内容</font></b></td>
    <td><%=rs("内容")%></td>
  </tr>
</table>
<HR>
<%
  rs.movenext( );
}>
%>

```

程序执行 SQL 语句“select * from lyb order by 时间 desc”将留言信息按照时间的倒序输出来，这样最新的留言总是在上面。首先在输入界面中输入一些留言，如图 7-13 所示。



图 7-13 输入留言

输入完以后，提交表单，就看到刚才的留言，程序执行的结果如图 7-14 所示。



图 7-14 显示留言

7.4 RecordSet 数据对象

记录集可以用来代表表中的记录。一个记录集包含一条或多条记录（行），每条记录包括一个或多个域（字段）。在任何时刻，只有一条记录是当前记录。

创建记录集对象的一个实例，可以使用 Connection 对象的 Execute() 方法，用 Execute() 创建的记录集指针只能向下，而不能向上移动，即不能执行 movefirst() 和 moveprevious() 指令。

7.4.1 RecordSet 对象的属性及方法

利用 RecordSet 对象的属性和方法可以完全操作一个数据库，理论上只要 SQL 语句可以完成的操作，利用 RecordSet 对象都可以实现。RecordSet 对象的属性和方法很多，下面是常用的属性和方法。

- (1) Move NumRecords: 在记录集中向前或向后移动指定数目的记录数。
- (2) MoveFirst: 移动到记录集的第一条记录。
- (3) MoveNext: 移动到记录集的下一条记录。
- (4) MovePrevious: 移动到记录集中的上一条记录。
- (5) MoveLast: 移动到记录集的最后一条记录。
- (6) AbsolutePage: 指定当前的页。
- (7) PagePount: 返回记录集中的逻辑页数。
- (8) PageSize: 指定一个逻辑页中的记录个数，默认值是 10。
- (9) AddNew: 向记录集中添加一条新记录。
- (10) CancelBatch: (当记录集处在批量更新模式时) 取消一批更新。
- (11) CancelUpdate: (调用 Update 之前) 取消对当前记录所做的所有修改。
- (12) Delete: 从记录集中删除一条记录。
- (13) Update: 保存对当前记录所做的修改。

(14) **UpdateBatch**: (当记录集处于批量更新模式时) 保存对一个或多个记录的修改。在此只做一般的解释, 下面利用 **RecordSet** 对象的属性和方法来操作数据库。

7.4.2 使用 RecordSet 对象打开数据库

前面介绍了访问数据库的格式一, 格式一的不足是数据记录指针只能向下移。下面介绍功能强大的格式二。

格式二: 数据库调用的基本格式

格式说明: 利用 RecordSet 对象打开数据库表

```
//第一步: 建立 Connection 对象
var conn = Server.CreateObject("ADODB.Connection");
//第二步: 使用 Connection 对象的 Open 方法建立数据库链接
conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
Server.MapPath("Access 数据库"));
//第三步: 建立 RecordSet 对象
var rs = Server.CreateObject("ADODB.Recordset");
//第四步: 利用 RecordSet 对象的 Open 方法打开数据库
rs.Open("SQL 语句", conn, 打开方式, 锁定类型);
```

可以看出, 格式二与格式一的区别在于最后两条语句, 首先建立一个 **RecordSet** 对象, 然后利用 **rs.Open** 方法打开数据库表。**rs.Open** 方法有四个参数, 后面的两个参数即打开方式和锁定类型可以省略。第一个参数是 SQL 语句, 第二个参数是前面建立的 **Connection** 对象。下面着重解释后面两种参数。

打开类型的四个参数如下。

- (1) **adOpenFowardOnly** (默认值): 对应的数字是 0, 记录集只能向前移动。
- (2) **adOpenKeyset**: 对应的数字是 1, 记录集可以向前或向后移动。如果另一个用户删除或改变一条记录, 记录集中将反映这个变化。但是, 如果另一个用户添加一条新记录, 新记录不会出现在记录集中。
- (3) **adOpenDynamic**: 对应的数字是 2, 使用动态游标, 可以在记录集中向前或向后移动。其他用户造成的记录的任何变化都将在记录集中有所反映。
- (4) **adOpenStatic**: 对应的数字是 3, 使用静态游标, 可以在记录集中向前或向后移动。但是, 静态游标不会对其他用户造成的记录变化有所反映。

锁定类型的参数如下。

- (1) **adLockReadOnly**: 只读锁定, 对应的数字是 1 (默认值), 不能修改记录集中的记录。
- (2) **adLockPessimistic**: 悲观锁定, 对应的数字是 2, 指定在编辑一个记录时, 立即锁定之。

```
进入锁定---rs("数学")=rs("数学")+100
rs("语文")=rs("语文")+100
rs.Update()-----解除锁定
```

- (3) **adLockOptimistic**: 乐观锁定, 对应的数字是 3, 指定只有调用记录集的 **Update** 方法

时，才锁定记录。

```
rs("数学")=rs("数学")+100
rs("语文")=rs("语文")+100
进入锁定----rs.Update()-----解除锁定
```

(4) adLockBatchOptimistic: 批次乐观锁定，对应的数字是 4，指定记录只能成批地更新。

```
for(i = 0; i < 10; i++)
{
    rs("数学")=rs("数学")+100
    rs("语文")=rs("语文")+100
    rs.movenext()
}
进入锁定----rsUpdate()-----解除锁定
```

上面的打开方式和锁定类型全部定义在 adojavas.inc（微软的定义文件）文件中。

格式二的功能非常强大，首先学习非常重要的属性 **AbsolutePosition**，当给 **AbsolutePosition** 赋值时，记录指针就会定位到相应的记录位置上。

当数据库第一次打开时，**RecordSet** 指针定位在第一条记录上，可以利用 **AbsolutePosition** 直接定位到某条记录上，基本的语法是：**AbsolutePosition=N**，如程序 7-07.asp 文件所示。

案例名称：使用 **AbsolutePosition** 属性

程序名称：7-07.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "Select * from grade";
    rs.Open(sql, conn, 3);
    rs.AbsolutePosition = 2;
    Response.Write(rs(1));
%>
```

程序执行完毕后，直接定位到数据库中第二条记录上，读取第二个字段的值，显示的结果如图 7-15 所示。

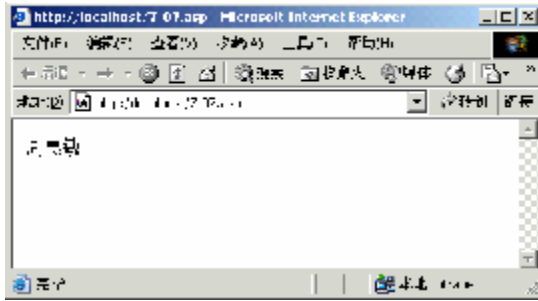


图 7-15 使用 AbsolutePosition 属性

还有一个属性是 AbsolutePage，当调用 AbsolutePage 时，系统将对数据记录进行分页，默认每页为 10 条记录，AbsolutePage 为几，记录指针就自动定位到第几页的第一条记录上。比如说，AbsolutePage=3，则记录指针就自动定位到第 21 条记录上去了，此时 AbsolutePosition=21。计算公式为：AbsolutePosition=(AbsolutePage-1)*PageSize，PageSize 默认是 10，也可以制定为其他的值。使用方法如程序 7-08.asp 所示。

案例名称：使用 AbsolutePage 属性

程序名称：7-08.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "Select * from grade";
    rs.Open(sql, conn, 3);
    rs.PageSize = 4;
    rs.AbsolutePage = 2;
    Response.Write(rs(1));
%>
```

程序执行后，就把数据库中第五条记录的第二个字段的内容输出到浏览器上。如图 7-16 所示。

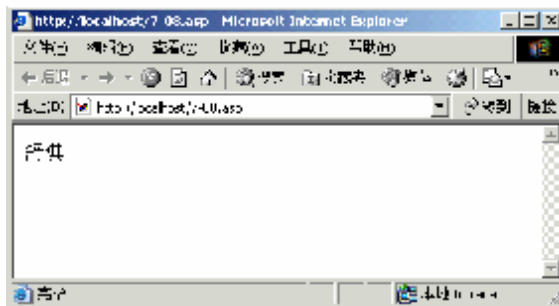


图 7-16 使用 AbsolutePage 属性

7.4.3 实现数据库的分页显示

数据库中可能有成千上万条记录，如果一次都显示出来的话消耗的时间太多。下面通过一个案例分成六个版本实现数据库的分页显示。

案例 7-3：分页显示

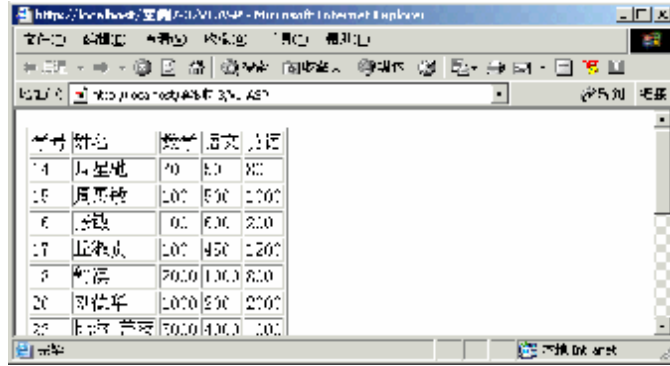
首先利用格式二将数据库表中的所有记录输出，如程序 V1.asp 所示。

案例名称：版本一输出所有记录

程序名称：V1.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "Select * from grade";
    rs.Open(sql, conn, 3);
%>
<table border=1>
<tr>
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
</tr>
<%
while(!rs.EOF)
{
    Response.Write("<tr>");
    Response.Write("<td>" + rs("学号") + "</td>");
    Response.Write("<td>" + rs("姓名") + "</td>");
    Response.Write("<td>" + rs("数学") + "</td>");
    Response.Write("<td>" + rs("语文") + "</td>");
    Response.Write("<td>" + rs("英语") + "</td>");
    Response.Write("</tr>");
    rs.movenext();
}
%>
</table>
```

利用循环将所有的数据全部读取并输出到浏览器上，如图 7-17 所示。



学号	姓名	数学	语文	英语
14	周星驰	70	80	90
15	周星驰	100	90	100
16	周星驰	00	90	200
17	周星驰	100	450	1200
18	周星驰	2000	1000	600
20	周星驰	1000	200	200
22	周星驰	2000	400	100

图 7-17 输出所有记录

实现输出某页的所有记录，如程序 V2.asp 所示。

案例名称：版本二输出某页所有记录

程序名称：V2.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "Select * from grade";
    rs.Open(sql, conn, 3);
    rs.PageSize = 4;
    rs.AbsolutePage = 2;

%>
<table border=1>
<tr>
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
</tr>
<%
for( i = 0; i < rs.PageSize; i++)
{
    if(!rs.Eof)
    {
        Response.Write("<tr>");
        Response.Write("<td>" + rs("学号") + "</td>");
        Response.Write("<td>" + rs("姓名") + "</td>");
        Response.Write("<td>" + rs("数学") + "</td>");
        Response.Write("<td>" + rs("语文") + "</td>");
```

```
Response.Write("<td>" + rs("英语") + "</td>");  
Response.Write("</tr>");  
rs.movenext();  
}  
}  
conn.close();  
%>  
</table>
```

程序设置 `PageSize` 属性为 4, 这样每页显示 4 条记录, 利用循环将该页的所有记录输出, 如图 7-18 所示。

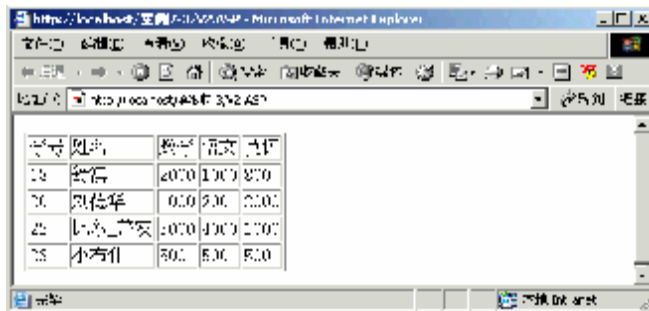


图 7-18 显示某页信息

在程序中加上翻页的超级连接就可以实现翻页, 如程序 V3.asp 所示。

案例名称: 版本三实现翻页

程序名称: V3.asp

```
<%@ Language=Jscript %>  
<%  
    conn = Server.CreateObject("ADODB.Connection");  
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +  
        Server.MapPath("person.mdb"));  
    rs = Server.CreateObject("ADODB.Recordset");  
    sql = "Select * from grade";  
    rs.Open(sql, conn, 3);  
    rs.PageSize = 4;  
    if (Request("page").Count > 0)  
    {  
        iPage = parseInt(Request("page"));  
    }  
    else  
    {  
        iPage = 1;  
    }  
}
```



```

    }
    Response.Write("当前第" + iPage + "页");
    rs.AbsolutePage = iPage;

%>
<table border=1>
<tr>
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
</tr>
<%
for( i = 0; i < rs.PageSize; i++)
{
    if(!rs.EOF)
    {
        Response.Write("<tr>");
        Response.Write("<td>" + rs("学号") + "</td>");
        Response.Write("<td>" + rs("姓名") + "</td>");
        Response.Write("<td>" + rs("数学") + "</td>");
        Response.Write("<td>" + rs("语文") + "</td>");
        Response.Write("<td>" + rs("英语") + "</td>");
        Response.Write("</tr>");
        rs.movenext();
    }
}

conn.close();

%>
</table>
<br><br><br>
<a href="#">第一页</a><a href="V3.asp?page=<%=iPage-1%>">上一页</a>
<a href="V3.asp?page=<%=iPage+1%>">下一页</a><a href="#">最后页</a>

```

程序“iPage = parseInt(Request("page"));”得到当前的页面,当单击上一页时,iPage 减一,从而向前翻页,如图 7-19 所示。

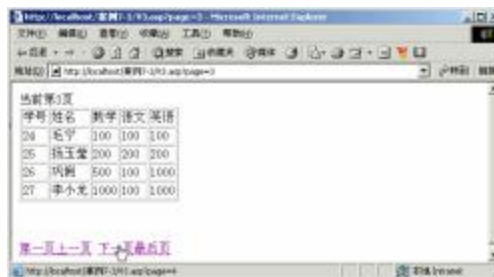


图 7-19 实现翻页

数据库中总共只有四页，但是 V3.asp 文件中的 iPage 的值可以无限增加，也可以等于零，这时程序出错。可以利用 PageCount 属性得到总页数，修正以上的 Bug，得到版本四。如程序 /4.asp 所示。

案例名称：版本四消除 Bug

程序名称：V4.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("person.mdb"));
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "Select * from grade";
    rs.Open(sql, conn, 3);
    rs.PageSize = 4;
    if (Request("page").Count > 0)
    {
        iPage = parseInt(Request("page"));
        if(iPage < 1)
            iPage = 1;
        if(iPage > rs.PageCount)
            iPage = rs.PageCount;
    }
    else
    {
        iPage = 1;
    }
    Response.Write("当前第" + iPage + "页，共" + rs.PageCount + "页");
    rs.AbsolutePage = iPage;

%>

<table border=1>
<tr>
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
</tr>
<%
for( i = 0; i < rs.PageSize; i++)
{
    if(!rs.Eof)
    {
```

```

        Response.Write("<tr>");
        Response.Write("<td>" + rs("学号") + "</td>");
        Response.Write("<td>" + rs("姓名") + "</td>");
        Response.Write("<td>" + rs("数学") + "</td>");
        Response.Write("<td>" + rs("语文") + "</td>");
        Response.Write("<td>" + rs("英语") + "</td>");
        Response.Write("</tr>");
        rs.movenext();
    }
}
%>
</table>
<br><br><br>
<%
    if (iPage != 1)
    {
        %>
        <a href="V4.asp?page=1">第一页</a>
        <a href="V4.asp?page=<%=iPage-1%>">上一页</a>
        <%
    }
    if (iPage != rs.PageCount)
    {
        %>
        <a href="V4.asp?page=<%=iPage+1%>">下一页</a>
        <a href="V4.asp?page=<%=rs.pageCount%>">最后一页</a>
        <%
    }
    conn.close();
    %>

```

到此，分页的基本功能实现了。在实际应用时，需要在某列上加上超级链接，当单击该超级链接时，显示该行的详细信息。加入超级链接的版本五如程序 V5.asp 所示。将姓名一列加上超级链接。

案例名称：版本五加上超级链接

程序名称：V5.asp

节选自 V5.asp，其他和 V4 相同

```

<%
for( i = 0; i < rs.PageSize; i++)
{

```

```

if(!rs.EOF)
{
    Response.Write("<tr>");
    Response.Write("<td>" + rs("学号") + "</td>");
%>

<TD><A HREF='detail.asp?xuehao=<%=rs("学号")%>' TARGET='_blank'>
<%=rs("姓名")%></A></TD>

<%

    Response.Write("<td>" + rs("数学") + "</td>");
    Response.Write("<td>" + rs("语文") + "</td>");
    Response.Write("<td>" + rs("英语") + "</td>");
    Response.Write("</tr>");
    rs.movenext();
}
}
%>

```

版本五只在版本四的基础上做了简单修改,在输出时给姓名字段加上了超级链接,如图 7-20 所示。



图 7-20 加上超级链接

当单击该超级链接时,调用另外一个文件 detail.asp,并将该学员的学号传递过去。Detail.asp 文件的功能是显示该行的详细信息,如程序 V5.asp 所示。

案例名称: 版本五加上超级链接

程序名称: V5.asp

```

<%@ Language=Jscript %>
<%
    strNo = Request("xuehao")(1)
%>
<%

```

```

conn = Server.CreateObject("ADODB.Connection");
conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
Server.MapPath("person.mdb"));
strSQL = "select * from grade where 学号="+strNo;
rs = conn.Execute(strSQL)

%>

<HTML>
<BODY>
<h1>姓名: <%=rs("姓名")%></h1>
他的成绩为: <br>
数学= <%=rs("数学")%><br>
英语=<%=rs("英语")%><br>
语文=<%=rs("语文")%><br>
</BODY>
</HTML>
</HTML>

```

程序利用上页传递过来的学号作为查询条件查询数据库, 显示该记录的详细信息。显示结果如图 7-21 所示。



图 7-21 显示详细信息

这个版本功能比较强大, 但是显示起来还比较难看, 一般在显示成表格时会加上一些样式, 而且偶数行和奇数行可显示不同的背景颜色。版本六将实现这些功能, 如程序 V6.asp 所示。

案例名称: 版本六加上显示样式

程序名称: V6.asp

节选自 V6.asp, 其他和 V4 相同

```

<table cellpadding="2" bordercolor="Black" border="1">
<tr style="background-color:#AAAADD;">
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
</tr>
<%
for( i = 0; i < rs.PageSize; i++)

```

```

{
    if(!rs.EOF)
    {
        if(i%2==0)
            Response.Write("<tr style='background-color:#FFFFCD;'>");
        else
            Response.Write("<tr>");
        Response.Write("<td>" + rs("学号") + "</td>");

%>
<TD><A HREF='detail.asp?xuehao=<%=rs("学号")%>' TARGET='_blank'>
<%=rs("姓名")%></A></TD>

<%
        Response.Write("<td>" + rs("数学") + "</td>");
        Response.Write("<td>" + rs("语文") + "</td>");
        Response.Write("<td>" + rs("英语") + "</td>");
        Response.Write("</tr>");
        rs.movenext();
    }
}
%>

```

条件 `if(i%2==0)` 判断当前行是否是奇数行，这里实现的是奇数行和偶数行显示不同的样式。程序执行的结果如图 7-22 所示。

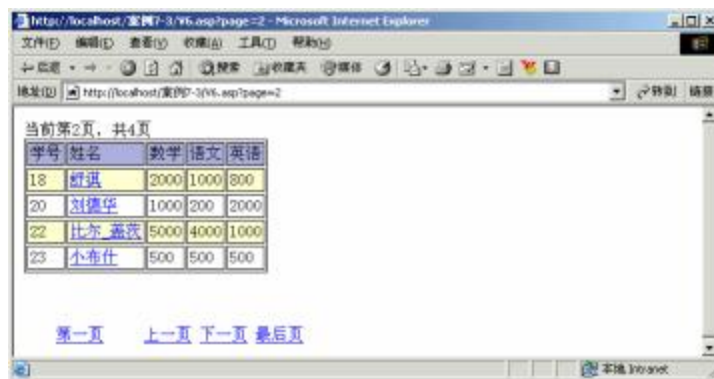


图 7-22 加上显示样式

到此，六个版本的分页显示完成。这里主要利用了格式二提供的功能。

7.5 Command 数据对象

单纯地从这个名称上来理解，可以理解为“命令”的意思，也就是说可以执行 SQL 语句。

Command 对象的功能是：执行 SQL 语句和调用 SQL Server 的存储过程。如基本格式三。

格式三：数据库调用的基本格式

格式说明：利用 Command 对象的 Execute 方法建立 RecordSet 对象

```
//第一步：建立 Connection 对象
var conn = Server.CreateObject("ADODB.Connection");
//第二步：使用 Connection 对象的 Open 方法建立数据库连接
conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
Server.MapPath("Access 数据库"));
//第三步：建立 Command 对象
cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
cmd.CommandText = sql;
//第四步：使用 Command 对象的 Execute 方法执行 SQL 语句
//如果执行查询语句
rs = cmd.Execute();
//如果执行数据操纵语句
cmd.Execute();
```

Command 对象的几个重要属性如下。

(1) ActiveConnection 属性：定义 Command 对象的连接信息。这个属性一般指向一个当前打开的 Connection 对象。

(2) CommandText 属性：为 SQL 语句、查询、表名或者 SQL Server 存储过程的名字。

(3) CommandType 属性：优化数据提供者的执行速度。通过对 CommandText 属性中所定义的命令类型，数据提供者就不用花时间去分析是何种类型的数据，如下所示。

AdCmdText：代表数字 1，表示处理的是一个 SQL 语句。

AdCmdTable：代表数字 2，表示处理的是一个表。

adCmdStoredProc：代表数字 4，表示处理的是一个存储过程。

首先介绍利用 Command 来执行 SQL 语句。使用 Command 对象来执行 SQL 语句，只要套用格式三就可以了，如程序 7-09.asp 所示。

案例名称：使用 Command 对象操作数据库

程序名称：7-09.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    conn = Server.CreateObject("ADODB.Connection");
    DBPath = Server.MapPath("person.mdb");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" + DBPath);
```

```
cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
sql = "Select * From grade where 数学 < 200";
cmd.CommandText = sql;
rs = cmd.Execute();
rstotab(rs);

%>
<%
function rstotab(rs)
{
    Response.write("<table border=1>");
    Response.write("<tr>");
    //输出表头
    for (i=0; i<rs.Fields.Count; i++)
    {
        Response.Write("<td>" + rs(i).Name + "</td>");
    }
    Response.write("</tr>");

    //输出表内容
    while (!rs.EOF)
    {
        Response.write("<tr>");
        for (i=0; i<rs.Fields.Count; i++)
        {
            Response.Write("<td>" + rs(i) + "</td>");
        }
        Response.write("</tr>");
        rs.movenext();
    }
    Response.write("</table>");
}
%>
</BODY>
</HTML>
```

首先建立一个 Command 对象名为 cmd, 然后再将和数据库已经建立联系的 conn 对象赋值给 cmd 的 ActiveConnection 属性。

再执行 SQL 语句将查询的结果赋值给 rs 对象, 当 Command 对象执行 Select 语句时, 自动返回一个 rs 对象。然后调用 rstotab 函数将返回的数据显示到浏览器上, 如图 7-23 所示。

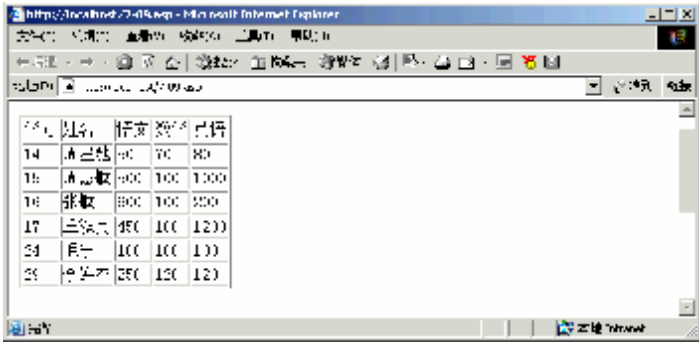


图 7-23 使用 Command 对象执行 SQL 语句

利用 Command 对象同样可以执行 Insert 语句、Update 语句和 Delete 语句，只是执行时不返回 RecordSet 对象。

案例 7-4：分页列表的删除和修改

本案例实现的功能是：修改数据和删除数据功能。首先为列表添加两列：修改和删除。显示的主页面如图 7-24 所示。



图 7-24 添加功能

修改后的程序如 V6.asp 所示。

案例名称：添加分页列表的功能

程序名称：V6.asp

节选自 V6.asp，其他和 V4 相同

```
<table cellpadding="2" bordercolor="Black" border="1">
<tr style="background-color:#AAAADD;">
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
    <td>修改</td><td>删除</td>
</tr>
<%
for( i = 0; i < rs.PageSize; i++)
{
```

```

        if(!rs.EOF)
        {
            if(i%2==0)
                Response.Write("<tr style='background-color:#FFFFCD;'>");
            else
                Response.Write("<tr>");
            Response.Write("<td>" + rs("学号") + "</td>");

%>
<TD><A HREF='detail.asp?xuehao=<%=rs("学号")%>' TARGET='_blank'>
<%=rs("姓名")%></A></TD>

<%
        Response.Write("<td>" + rs("数学") + "</td>");
        Response.Write("<td>" + rs("语文") + "</td>");
        Response.Write("<td>" + rs("英语") + "</td>");

%>
<td><a href="do_modify.asp?xuehao=<%=rs("学号")%>">修改</a></td>
<td><a href="do_del.asp?xuehao=<%=rs("学号")%>">删除</a></td>

<%
        Response.Write("</tr>");
        rs.movenext();
    }
}
%>
</table>

```

程序添加了两列，并将两列做成超级链接，将学号作为参数传递到链接的页面。当单击修改的按钮后，打开 do_modify.asp 文件，如下所示。

案例名称：处理修改操作

程序名称：do_modify.asp

```

<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    var str = Request("xuehao")(1);
    conn = Server.CreateObject("ADODB.Connection");
    DBPath = Server.MapPath("person.mdb");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" + DBPath);

    cmd = Server.CreateObject("ADODB.Command");

```

```

cmd.ActiveConnection = conn;
sql = "select * from grade where 学号="+str;
cmd.CommandText = sql;
rs = cmd.Execute();

%>
<%=rs("姓名")%>的成绩为:<br>
<form action="do_mdysmt.asp" method="post">
<%
    //输出表内容
    if(!rs.EOF)
    {
%>
<input type="hidden" name="xuehao" value="<%=rs("学号")%>"><br>
语文:<input type="text" name="yuwen" value="<%=rs("语文")%>"><br>
数学:<input type="text" name="shuxue" value="<%=rs("数学")%>"><br>
英语:<input type="text" name="yingyu" value="<%=rs("英语")%>"><br>
<input type="submit" value="修改">
<%
    }
%>
</form>
</BODY>
</HTML>

```

程序将该学号人员的成绩读到文本框中,这样就可以修改了。程序显示的结果如图 7-25 所示。

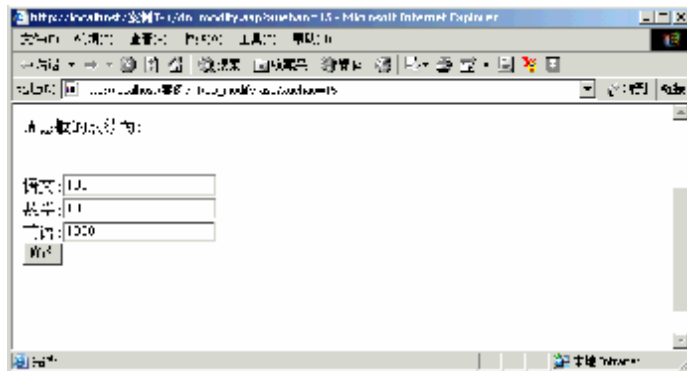


图 7-25 修改数据

修改某一门的成绩,单击“修改”按钮,调用 do_mdysmt.asp 文件,将修改的结果保存到数据库中,如下所示。

案例名称: 处理修改提交操作

程序名称: do_mdysmt.asp

```

<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    var str = Request("xuehao")(1);
    var yuwen = Request("yuwen")(1);
    var shuxue = Request("shuxue")(1);
    var yingyu = Request("yingyu")(1);
    conn = Server.CreateObject("ADODB.Connection");
    DBPath = Server.MapPath("person.mdb");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" + DBPath);

    cmd = Server.CreateObject("ADODB.Command");
    cmd.ActiveConnection = conn;
    sql = "update grade set 数学="+shuxue+",英语="+yingyu+",语文="+
    yuwen+" where 学号=" + str;
    Response.Write(sql);
    cmd.CommandText = sql;
    cmd.Execute();
    Response.Redirect("V6.asp");
%>
</BODY>
</HTML>

```

程序利用 Update 语句更新数据库表的内容。当单击“删除”超级链接时，程序自动调用 lo_del.asp 文件将该记录删除，如图 7-26 所示。



图 7-26 删除数据

程序 do_del.asp 文件如下。

案例名称：处理删除操作

程序名称：do_del.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    var str = Request("xuehao")(1);
    conn = Server.CreateObject("ADODB.Connection");
    DBPath = Server.MapPath("person.mdb");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" + DBPath);

    cmd = Server.CreateObject("ADODB.Command");
    cmd.ActiveConnection = conn;
    sql = "delete from grade where 学号=" + str;
    Response.Write(sql);
    cmd.CommandText = sql;
    cmd.Execute();
    Response.Redirect("V6.asp");
%>
</BODY>
</HTML>
```

这种修改和删除的技术在网站开发中应用的非常广泛，需要掌握。

案例 7-5：分页版本的留言簿

这里再对留言簿作一次升级，在案例 7-2 的基础上加入分页功能。只要更改 display.asp 文件就可以，如下所示。

案例名称：留言的分页显示

程序名称：display.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" +
        Server.MapPath("lyb.mdb"));
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "Select * from lyb order by 时间 desc";
    rs.Open(sql, conn, 3);
    rs.PageSize = 1;
    if (Request("page").Count > 0)
    {
        iPage = parseInt(Request("page"));
        if(iPage < 1)
```

```
        iPage = 1;
        if(iPage > rs.PageCount)
            iPage = rs.PageCount;
    }
    else
    {
        iPage = 1;
    }
    Response.Write("当前第" + iPage + "页, 共" + rs.PageCount + "页");
    rs.AbsolutePage = iPage;

%>

<%
for( i = 0; i < rs.PageSize; i++)
{
    if(!rs.Eof)
    {
%>
<table border="0" bordercolor="#111111">
    <tr>
        <td><b><font size="4" color="#008080">姓名: </font></b></td>
        <td><%=rs("姓名")%></td>
    </tr>
    <tr>
        <td><b><font size="4" color="#008080">
            E-mail:</font></b></td>
        <td><a href="mailto:<%=rs("E-mail")%>">
            <%=rs("E-mail")%></a></td>
    </tr>
    <tr>
        <td><b><font size="4" color="#008080">主题: </font></b></td>
        <td width="542" height="17"><%=rs("主题")%></td>
    </tr>
    <tr>
        <td><b><font size="4" color="#008080">时间: </font></b></td>
        <td><%=rs("时间")%></td>
    </tr>
    <tr>
        <td><b><font size="4" color="#008080">内容</font></b></td>
```

```

        <td><%=rs("内容")%></td>
    </tr>
</table>
<HR>
<%
    rs.movenext();
}
}
%>

<br><br><br>
<%
    if (iPage != 1)
    {
        %>
        <a href="display.asp?page=1">第一页</a>
        <a href="display.asp?page=<%=iPage-1%>">上一页</a>
        <%
    }
    if (iPage != rs.PageCount)
    {
        %>
        <a href="display.asp?page=<%=iPage+1%>">下一页</a>
        <a href="display.asp?page=<%=rs.pageCount%>">最后页</a>
        <%
    }
    conn.close();
%>

```

设置每页显示一条记录，和 V4.asp 文件基本相同。留言簿显示的结果如图 7-27 所示。

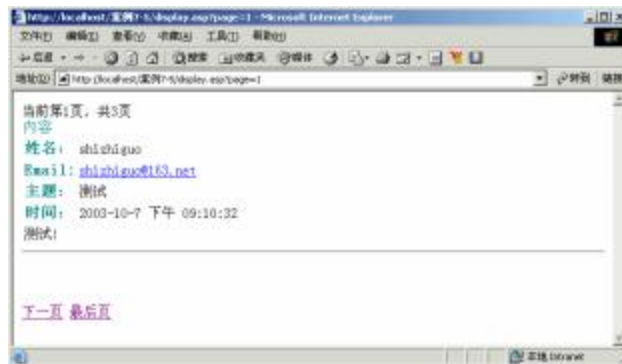


图 7-27 留言簿的分页显示

小结

本章重点理解访问数据库的三大基本格式及各自的特点。掌握 Connection, RecordSet 和 Command 对象的使用方法。熟练掌握利用格式二实现数据库的分页显示和利用 SQL 语句操作数据库的方法。

课后习题和上机练习

1. 简述 ADO 的功能及常用的三大对象的用途。
2. 访问数据库格式一有什么特点？可以执行哪些 SQL 语句？
3. 如何实现模糊查询？
4. 如何用变量替换 SQL 语句的值？
5. 格式二的数据打开方式和锁定方式有几种？各有什么含义？
6. 比较访问数据库的三个基本格式的异同。
7. 在案例 7-4 的基础上，增加添加数据功能。（上机练习）

第 8 章 ASP 操作 SQL Server 数据库

本章要点

本章主要介绍 SQL Server2000 的基本使用，介绍 SQL Server2000 集成开发环境的使用，介绍如何在 SQL Server2000 中使用查询分析器建立数据库和数据库表，以及如何使用 ADO 操作数据库表。重点介绍如何使用 ADO 操作 SQL Server2000 的存储过程。

3.1 SQL Server 简介

Microsoft SQL Server2000 是为当前的分布式客户机—服务器环境特别设计的产品，SQL Server 数据库能够轻松通过 Internet 和内部网络加以访问。SQL Server 提供一系列方法将数据真充到 Web 服务器，同时提供对数据的快捷访问。它是一个可伸缩、高性能的数据库管理系统，具备内置的复制能力、Internet 集成、开放系统体系结构及强大的基于 GUI 的管理工具。它的一个独特功能是基于服务器的作业调度系统，允许直观地控制多个服务器和远程操作。这极大地增强了数据库的性能和可靠性。与 Access 相比较，它具有更好的应用特征，如下所示。

- (1) 支持企业级运算、支持 C/S 模型、更好的性能和更方便的操作。
- (2) 功能增强：海量数据存储、数据复制、数据转换服务、分布式事务、全文检索。
- (3) 支持多种协议（TCP/IP、NETBEUI）和支持分布式计算，分布式计算模型。
- (4) ANSI/92 标准兼容并进行 T-SQL 的增强。

3.2 SQL Server 的集成环境介绍

SQL Server2000 提供强大的 GUI（Graphic User Interface）界面，用户可以直接通过界面或者通过 T-SQL 语句操作数据库。常用的有 SQL Server 管理器、企业管理器、查询分析器、事件查看器和联机帮助，下面逐个介绍。

3.2.1 SQL 服务管理器

打开菜单选项，找到 Microsoft SQL Server 下的服务管理器，如图 8-1 所示。

单击“开始/继续”旁边的绿色三角的按钮，启动服务器，如图 8-2 所示。

这时在任务栏可以看到 SQL Server 的图标，如图 8-3 所示。

从图 8-2 中可以看出，SQL Server 的名称是 SZG-NB，启动的服务是 SQL Server，下面有三个按钮控制 SQL Server 运行。启动和停止的功能一目了然，但暂停和停止有什么区别呢？区别在于当服务器暂停的时候，已经登录到系统的用户依然可以继续得到 SQL Server 提供的数据服务，但是现在已经不能登录 SQL Server 了。当服务器停止的时候，所有的用户，包括登录与没有登录的都不能继续访问 SQL Server 了。



图 8-1 选择菜单



图 8-2 服务管理器



图 8-3 任务栏图标

3.2.2 企业管理器

企业管理器是图形化管理界面的核心。在“开始”菜单中选择“企业管理器”，打开企业管理器，如图 8-4 所示。

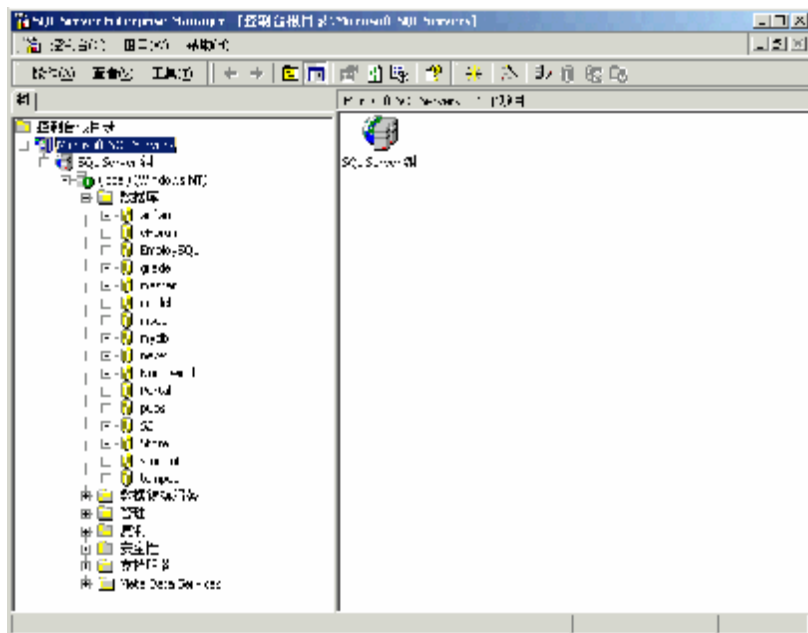


图 8-4 企业管理器界面

可以看到有许多选项,在这个界面上可以利用图形工具创建数据库、表、视图和存储过程等数据库项目。通常用得最多的就是企业管理器和查询分析器。

3.2.3 查询分析器

选择“开始”菜单中的“查询分析器”,出现登录对话框,如图 8-5 所示。

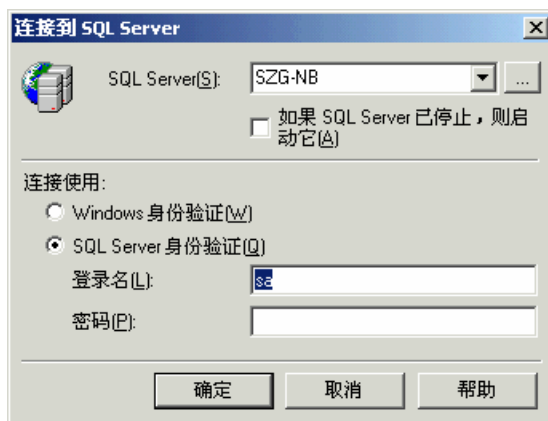


图 8-5 登录对话框

利用“sa”作为登录名登录,如果是自己练习的话,在安装数据库时建议使用空密码,因为很多测试程序都为空。但是作为工程项目的数据库,千万不要将“sa”的密码设置为空。登录系统以后,就可以测试 SQL 语句了,如图 8-6 所示。



图 8-8 联机丛书

3.3 创建数据库

可以使用 Create Database 命令来创建新的数据库和相关的日志文件。

3.3.1 创建数据库

创建一个数据文件、一个日志文件的基本语法，如程序 8-01.sql 所示。

案例名称：创建单数据文件的数据库

程序名称：8-01.sql

```
CREATE DATABASE MySales
ON
( NAME = Sales_dat,
  FILENAME = 'c:\program files\microsoft sql server\mssql\ data\ Mysaledat.
idf',
  SIZE = 2,
  MAXSIZE = 2,
  FILEGROWTH = 2 )
LOG ON
( NAME = 'Sales_log',
  FILENAME = 'c:\program files\microsoft sql server\mssql\data\ Mysalelog.
df',
  SIZE = 1MB,
  MAXSIZE = 1MB,
  FILEGROWTH = 1MB )
GO
```

在查询分析器里面执行程序，结果如图 8-9 所示。

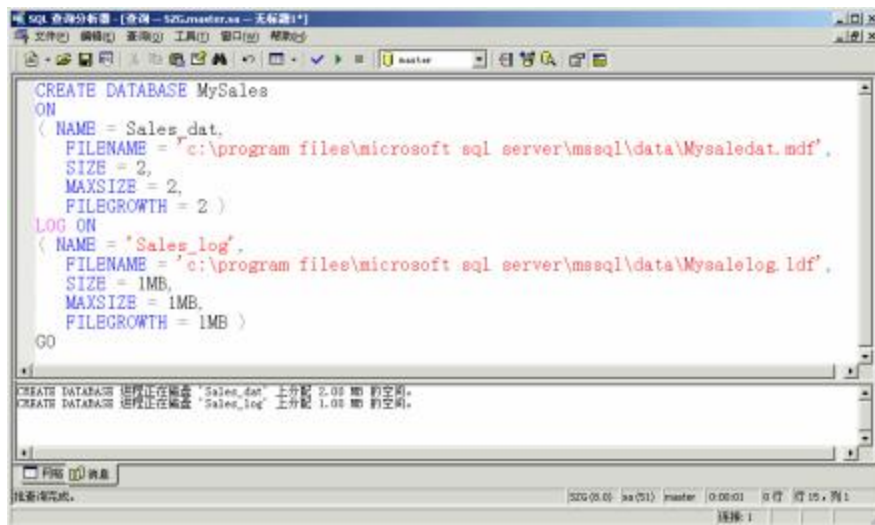


图 8-9 创建数据库

注意：SQL 语句不区分大小写。

这样数据库就创建出来了。也可以创建多数据文件和多日志文件的数据库，如程序 8-02.sql 所示。

案例名称：创建多数据文件的数据库

程序名称：8-02.sql

```
CREATE DATABASE Archive
ON
PRIMARY ( NAME = Arch1,
          FILENAME = 'c:\program files\microsoft sql server\mssql\data\ archdat1.
          ndf',
          SIZE =1MB,
          MAXSIZE =1,
          FILEGROWTH = 1),
( NAME = Arch2,
  FILENAME = 'c:\program files\microsoft sql server\mssql\data\ archdat2.ndf',
  SIZE = 1MB,
  MAXSIZE = 1,
  FILEGROWTH = 1),
( NAME = Arch3,
  FILENAME = 'c:\program files\microsoft sql server\mssql\data\archdat3.ndf',
  SIZE = 1MB,
  MAXSIZE = 1,
  FILEGROWTH = 1)
LOG ON
( NAME = Archlog1,
```

```
FILENAME = 'c:\program files\microsoft sql server\mssql\data\ archlog1.ldf',  
SIZE = 1MB,  
MAXSIZE = 1,  
FILEGROWTH =1),  
( NAME = Archlog2,  
FILENAME = 'c:\program files\microsoft sql server\mssql\data\ archlog2.ldf',  
SIZE = 1MB,  
MAXSIZE = 1,  
FILEGROWTH =1)  
GO
```

如果有多个数据文件，必须有一个主数据库文件，该文件的扩展名为 **mdf**，其他的数据文件为辅数据库文件，扩展名为 **ndf**。Log 文件的扩展名为 **ldf**。程序执行结果如图 8-10 所示。

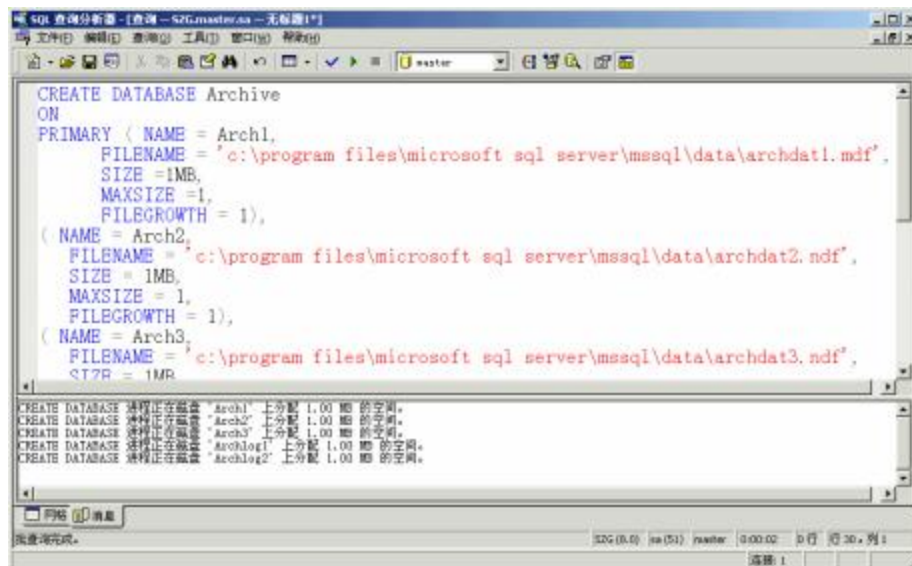


图 8-10 创建多数据文件的数据库

3.3.2 删除数据库

删除数据库的关键词是：**DROP DATABASE**。

语法如下：

```
DROP DATABASE TEST1
```

说明：删除名为 **Test1** 的数据库。

注意：不要轻易删除数据库，将导致所有数据的完全丢失！

3.3.3 数据类型

数据类型指列、存储过程参数和局部变量的数据特性。数据按照数据类型存储在列中。

例如, 日期与文本及数字分别存储在不同列中。创建表以存储数据时, 必须指定每一列的数据类型。为此, 需要提前规划表的结构。

SQL Server 提供了大量的数据类型。常用的数据库类型如表 8-1 所示。

表 8-1 SQL Server 提供的数据类型

数据类型	描述	需要空间
Binary	固定长度的二进制数据, 最大长度为 8000 字节	0 到 8000 字节, 具体取决于定义
Char	固定长度的非 Unicode 字符数据, 最大长度为 8000 个字符	0 到 8000 字节, 具体意义取决于定义
Datetime	日期和时间数据	8 字节
Int	整型数据, 从 -2^{31} 到 $2^{31}-1$	4 个字节
Money	货币数据值, 从 -2^{63} 到 $2^{63}-1$	8 字节
Smallint	整型数据, 从 -2^{15} 到 $2^{15}-1$	2 字节
Varchar	可变长度的非 Unicode 数据, 最大长度为 $2^{31}-1$ 个字符	存储大小是输入数据的实际长度
Uniqueidentifier	存储作为全局唯一标识(GUID)的 16 字节的二进制数值。GUID 是确保唯一性的二进制数字	16 字节

这些数据类型将在下面建表的语句中使用。

3.3.4 表

表是关系型数据库中的逻辑单元, 该数据库用于存储实体数据。表由行和列组成。行描述实体的实例, 列定义实体的属性。为表命名时必须小心, 应确保表名称在数据库中是唯一的, 并且应遵循标识符命名规则。

SQL Server 有下列几个针对表的命名约定, 这些约定对所有其他数据库对象也适用。

- (1) 可以包含 1 到 128 个字符, 包括字母、符号和数字。
- (2) 第一个字符必须是字母、下划线(_)、@符号。
- (3) 首字母之后的字符可以包括字母、数字或#、\$符号及其_。
- (4) 除非在引号内定义对象名称, 否则不允许有空格。

可以使用“企业管理器”或 Create Table 语句创建表。Create Table 语句可以用来创建数据库表。可在数据库中创建数据库表, 如程序 8-03.sql 的语法。

案例名称: 创建数据库表

程序名称: 8-03.sql

```
CREATE TABLE MyTable
(
    MyName CHAR(10) NOT NULL,
    MyBorthDay DATETIME,
)
```

程序执行的结果如图 8-11 所示。

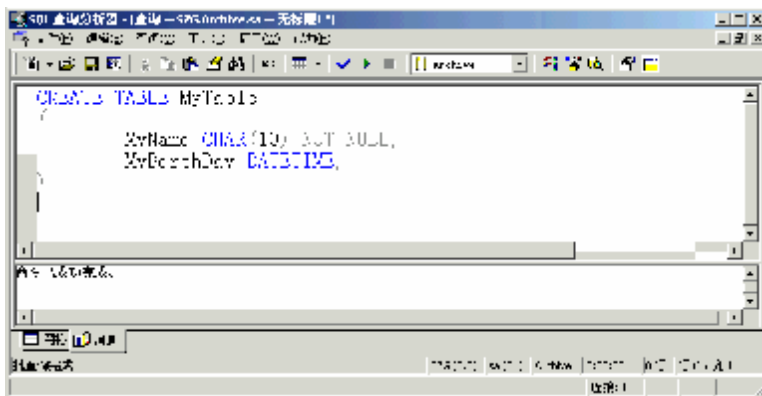


图 8-11 创建数据库表

3.3.5 修改表

修改表是指修改表的结构，调整增减列，也就是调整实体的属性的过程。

1. 增加列

在 myTable 中增加一列 MySistName，属性是 CHAR(20)，如程序 8-04.sql 所示。

案例名称：添加列

程序名称：8-04.sql

```
ALTER TABLE MyTABLE ADD MySistName CHAR(20)
```

程序执行的结果如图 8-12 所示。

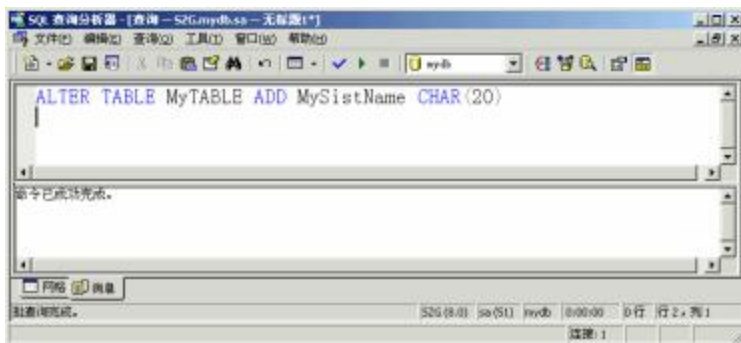


图 8-12 添加列

2. 删除列

删除新增的 MysistName 列，如程序 8-05.sql 所示。

案例名称：删除列

程序名称：8-05.sql

```
ALTER TABLE MyTABLE DROP COLUMN MySistName
```

程序执行的结果如图 8-13 所示。

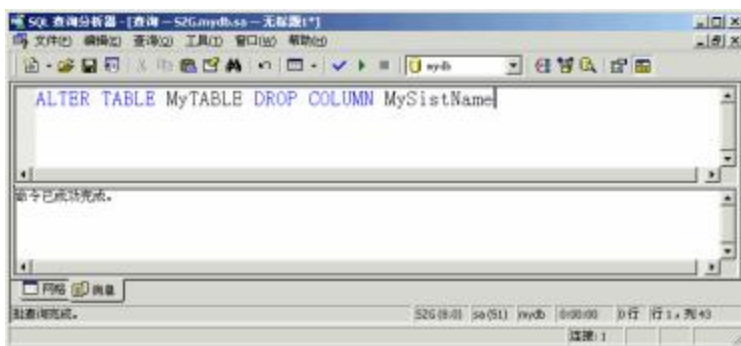


图 8-13 添加数据库表的列

3.3.6 删除表

可以使用“企业管理器”或者用 Drop Table 语句删除 SQL Server 中的表。

语法：

```
Drop Table table_name
```

说明：删除一个表，表名为 table_name。

3.4 数据完整性

一旦创建并填充完数据库，就应确保存储数据的可靠性，这对于任何企业都很关键。因此必须在设计数据库的时考虑数据完整性。

数据完整性指数据库中存储数据的一致性。常规数据库管理系统需要在每个应用程序中编码实现数据完整性逻辑。

实现数据完整型可以利用下面的三种方法。

- (1) 使用 Identity 属性。
- (2) 使用 Uniqueidentifier 数据类型和 NEWID() 函数。
- (3) 使用六大约束。

3.4.1 使用 Identity 属性

表中一般会包含连续值的列，将 Identity 属性添加到该列上，SQL Server 可自动生成这些值。Identity 属性生成的值唯一地标识表中的每一行，每次表中插入一行时，该属性就会自动生成值。

在创建表的时候创建 Identity 列，定义 Identity 列的语法如下：

```
Identity [ (Seed, Increment) ]
```

参数说明如下。

Seed: 指定 Identity 列的初始值。

Increment: 每次自动增加多少。

注意: Seed 和 Increment 参数是可选的, 如果没有指定, 则使用默认为 1。

创建一个 Student 表, 其中的 StudID 列具有 Identity 属性, Seed 值为 101, Increment 的值为 5, 如程序 8-06.sql 所示。

案例名称: 创建 Identity 列

程序名称: 8-06.sql

```
CREATE TABLE Student
(
    StudID int Identity (101, 5),
    FirstName Varchar(20),
    LastName varchar(20)
)

Insert into Student(FirstName, LastName)
Values('runfa','zhou')

Select * from student
```

如果已经创建了 Student 表, 并且希望将 Identity 属性附加到 StudID 列, 则可以使用下面的语句。

```
Alter Table Student add StudID int Identity (101,5)
```

程序 8-06.sql 在数据库中创建一个 Student 表, 利用 Insert 语句插入一条记录, 然后利用 select 语句查看表的内容, 可以看到 StudID 列自动为 101, 如图 8-14 所示。

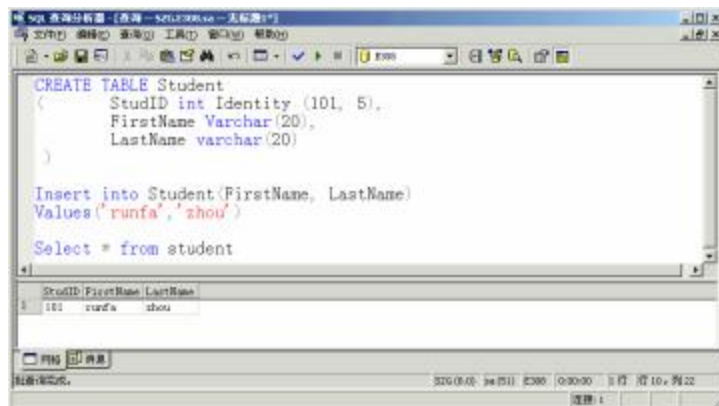


图 8-14 使用 Identity 列

注意: 在执行 Insert 语句时, 该列不能插入一个数, 因为该列是由 SQL Server 控制的。如果再插入一条记录该列就变成 106。

注意: 在查询分析器中, 用鼠标选中一条完整的 SQL 语句, 查询分析器就执行该语句, 不执行其他语句, 如果没有任何一条语句被选中, 就全部执行。

3.4.2 使用 Uniqueidentifier 类型

可以使用 Uniqueidentifier 数据类型和 NEWID()函数来生成列的惟一值。如果创建的列是 Uniqueidentifier 类型, 需使用 NEWID()函数为该列生成新值。利用 Create table 来创建 Uniqueidentifier 列, 如程序 8-07.sql 所示。

案例名称: 创建 Uniqueidentifier 类型

程序名称: 8-07.sql

```
CREATE TABLE MYFRIEND
(
    NID UNIQUEIDENTIFIER,
    STUDENTXING VARCHAR(20),
    STUDENTMING VARCHAR(20)
)
GO
INSERT MyFriend Values(NEWID(),'周','润发 ')
GO
INSERT MyFriend Values(NEWID(),'周','敏')

Select * from MyFriend
```

创建完以后, 可以使用 NEWID()函数通过下面的命令向表中插入数据。程序执行的结果如图 8-15 所示。

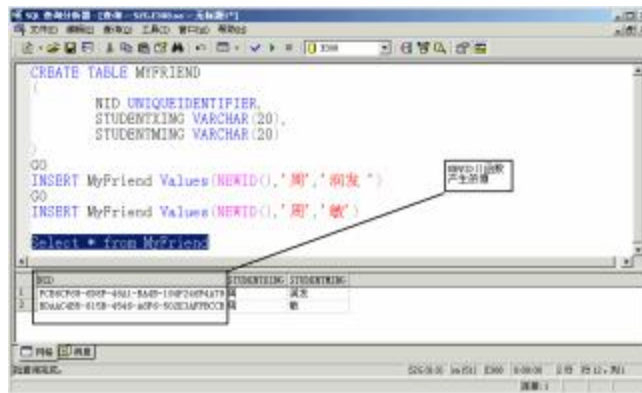


图 8-15 使用 Uniqueidentifier 类型

利用 NEWID()函数产生的列永远不会重复。

3.4.3 使用约束

最常用的对表强制执行完整性的方法是使用约束, 限制表或列中的值。

约束有六种,分别是:主键约束(Primary Key)、外键约束(Foreign key)、惟一约束(Unique)、非空约束(Not Null)、检查约束(Check)和默认约束(Default)。

1. 主键约束

主键约束具有如下特性。

- (1) 一个表中只可定义一个主键。
- (2) 不能在主键列中输入 null 值和重复的值。
- (3) 最多可定义 16 列作为主键。

创建一个新的 Student 表,并将表的 StudID 列设置为主键,如程序 8-08 所示。

案例名称:使用主键

程序名称:8-08.sql

```
CREATE TABLE STUDENT
(
    STUDID          INT PRIMARY KEY,
    FIRSTNAME       VARCHAR(20),
    LASTNAME        VARCHAR(20),
)
```

```
Insert into student Values(1001, 'runfa', 'zhou')
```

首先创建 Student 表,然后执行 Insert 语句,第一遍执行没有问题。当再执行一次同样的 insert 语句时,违反了主键不能重复的约束,显示错误信息,如图 8-16 所示。

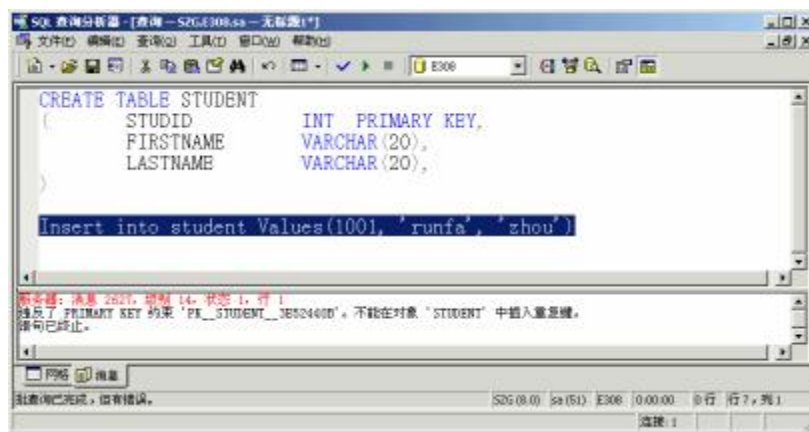


图 8-16 使用主键约束

2. 外键约束

Foreign key 约束定义列值与另一个表的主键相匹配。使用外键时应该注意: Foreign key 约束必须引用另一个表的主键列或者 Unique 列。

下面创建两个表,并且建立两张表的外键关系。一个是学生的基本信息表,另一个是历史表。学生的基本信息是基本固定的,学员每学习完一年就升学到高年级学习,这样,一个

学员可能对应多条历史记录。有这样的一个关系，所有历史表中的学员一定在基本信息表中可以找到，因为学员总是先注册然后上课学习的。外键约束就是要实现这样的约束关系。如程序 8-09.sql 所示。

案例名称：使用外键

程序名称：8-09.sql

```
Create Table basicinfo
(
    stu_id      int Identity(1001,1) Primary Key,
    Firstname   Varchar(10) ,
    Lastname    Varchar(10)
)

Create Table history
(
    historyid    int Primary Key,
    stu_id       int,
    stu_grade    int
    foreign key(stu_id) references basicinfo(stu_id)
)
```

这里利用语句“foreign key(stu_id) references basicinfo(stu_id)”建立的外键约束关系是：history 表的 stu_id 列，对应 basicinfo 表的主键 stu_id。

下面利用语句来测试外键关系。首先执行“Insert into history Values(100,1001,98)”语句，插入学号为 1001 同学的成绩，因为此时 Basicinfo 表中还没有学号为 1001 的同学，所以违反了约束的关系，执行的结果如图 8-17 所示。

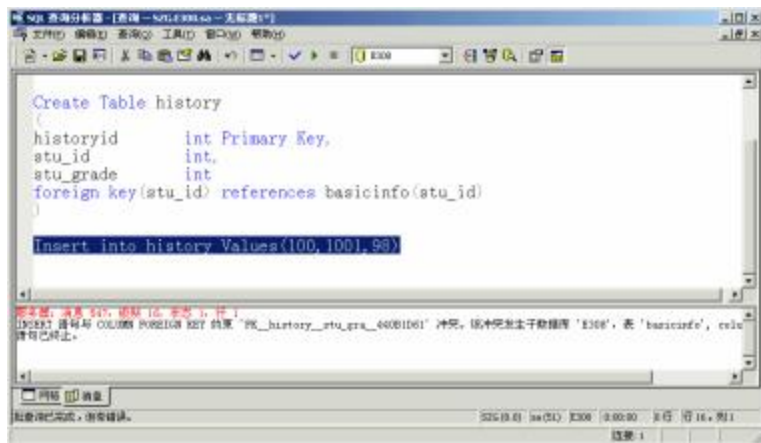


图 8-17 执行插入操作

先在 basicinfo 表中插入一个学员，语句是“Insert into basicinfo values('runfa', 'zhou)”，该学员的学号为 1001，再向 History 表中插入该记录就可以了，如图 8-18 所示。

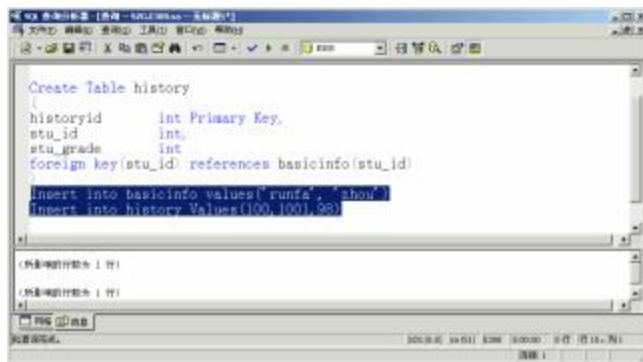


图 8-18 执行插入操作

可以看出两条语句都成功执行。

3. 惟一约束

在列中应用 Unique 约束是为了确保列中不输入重复的值。使用 Unique 约束应该注意：Unique 约束的列可以出现一个空值的行，只要不重复就不违反约束。下面的案例向 Student 表中的 TelNo 列添加 Unique 约束。如程序 8-10.sql 所示。

案例名称：使用惟一约束

程序名称：8-10.sql

```
Create Table testUnique
(
    stu_id          int Identity(1001,1) Primary Key,
    Firstname       Varchar(10) Unique,
    Lastname        Varchar(10)
)
```

```
Insert into testUnique Values('runfa', 'zhou')
```

创建表 testUnique，惟一性列是 Firstname，当第一次执行 Insert 语句时，没有出错，但是第二次执行时，违反了惟一性约束。如图 8-19 所示。

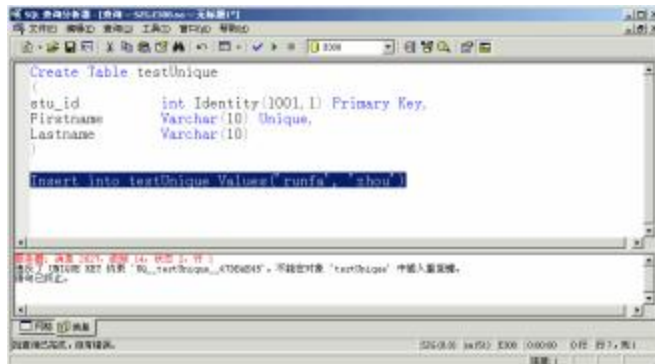


图 8-19 惟一性约束

4. 非空约束

如果一个列被附加了非空约束，该列就不能为空值。每次插入数据时必须插入数据，如程序 8-11.sql 所示。

案例名称：使用非空约束

程序名称：8-11.sql

```
Create Table testNotNull
(
    Firstname    Varchar(10) Not Null,
    Lastname     Varchar(10)
)
```

```
Insert into testNotNull(lastname) Values('zhou')
```

列 Firstname 是非空列，当执行 Insert 语句的时候，因为没有插入 Firstname 的值，这样就要写空值到该列中了，这样违反了非空约束，如图 8-20 所示。

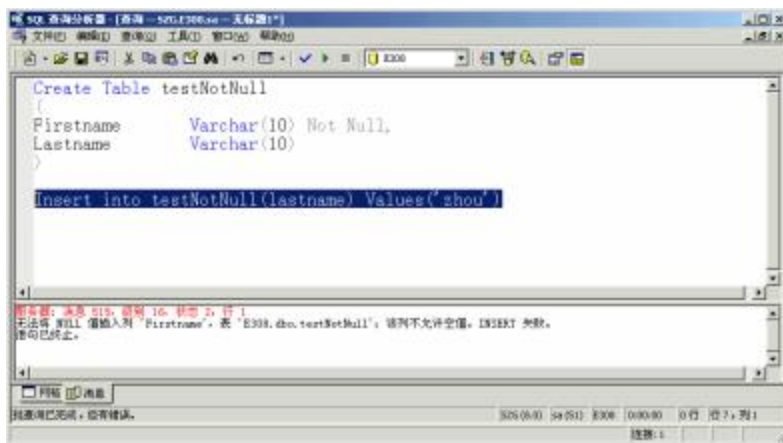


图 8-20 非空约束

5. 检查约束

CHECK 约束根据指定值测试列中的输入值。每次在列中插入或更新数据时都需要进行这一测试。如程序 8-12.sql 所示。

案例名称：使用检查约束

程序名称：8-12.sql

```
Create Table testCheck
(
    stu_id       int IDENTITY(100000,1) Primary Key,
    Firstname     Varchar(10) not null,
```



```

lastname      Varchar(10) not null,
age           int CHECK (age > 6),
sex           Varchar(30) CHECK(sex in('M','F'))
)

```

```
Insert Into testCheck Values('runfa','zhou',5,'M')
```

```
Insert Into testCheck Values('runfa','zhou',7,'A')
```

CHECK 约束限制 age 列的输入值必须大于 6，限制 sex 列的输入值必须为 M 或者 F，这样，当执行下面两条 Insert 语句时就违反了约束，如图 8-21 所示。

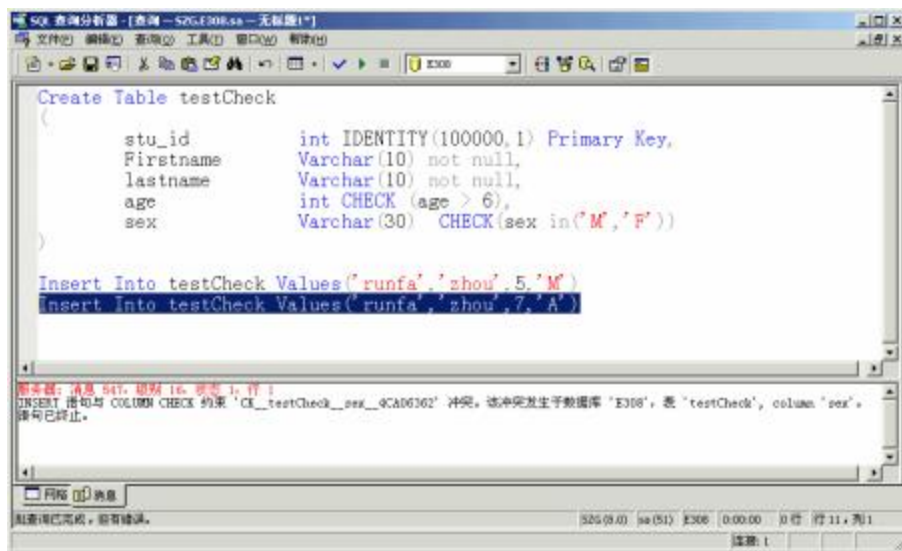


图 8-21 使用检查约束

将第一条 Insert 语句的 5 改成大于 6 的值就可以成功执行了。

6. 默认约束

Default 约束用于在用户未提供列值的情况下，提供一个自动添加的列值。testDefault 表中的 Sex 列被添加 Default 约束，默认值为“M”，如程序 8-13.sql 所示。

案例名称：使用默认约束

程序名称：8-13.sql

```

Create Table testDefault
(
    stu_id      int IDENTITY(100000,1) Primary Key,
    Firstname   Varchar(10) not null,
    Lastname    Varchar(10) not null,
    Sex         Varchar(30) Default 'M'
)

```

```
Insert into testDefault(Firstname,Lastname) Values('runfa','zhou')
```

```
Select * from testDefault
```

程序执行的结果如图 8-22 所示。

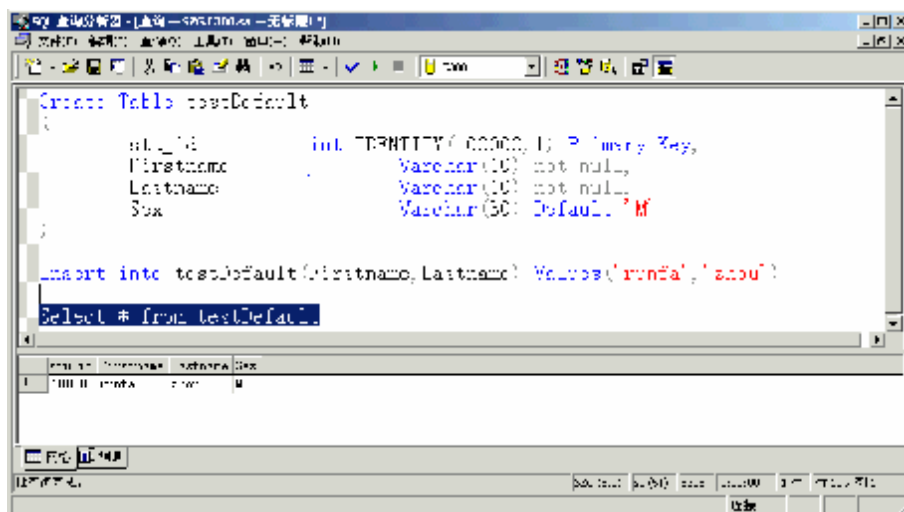


图 8-22 使用默认约束

3.5 ADO 操作 SQL Server 数据库

首先建立 SQL Server 的数据库表。在 pubs 数据库中建立一个 grade 数据表，语句如程序 8-14.sql 所示。

案例名称：新建数据库表

程序名称：8-14.sql

```
use pubs
go
Create Table grade
(
    学号          int Primary Key,
    姓名          Varchar(20) not null,
    语文          int,
    数学          int,
    英语          int
)
```

程序在查询分析器中执行结果如图 8-23 所示。

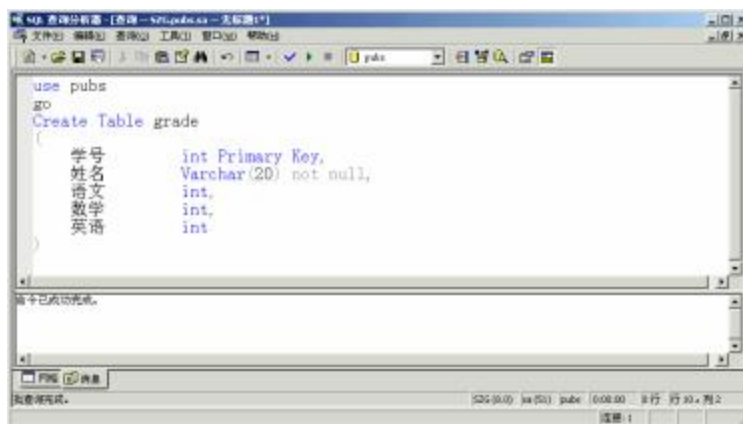


图 8-23 创建数据库表

表建立后，为了查询方便还要添加一些测试数据，如程序 8-15.sql 所示。

案例名称：添加测试数据

程序名称：8-15.sql

```
Insert into grade Values(14,'周星驰',50,90,90)
Insert into grade Values(15,'张敏',50,90,90)
Insert into grade Values(16,'丘淑贞',50,90,90)
Insert into grade Values(17,'舒淇',50,90,90)
Insert into grade Values(18,'刘德华',50,90,90)
Insert into grade Values(19,'比尔_盖茨',50,90,90)
Insert into grade Values(20,'小布什',50,90,90)
Insert into grade Values(21,'巩俐',50,90,90)
Insert into grade Values(22,'李小龙',50,90,90)
Insert into grade Values(24,'成龙',50,90,90)
Insert into grade Values(25,'李连杰',50,90,90)
Insert into grade Values(26,'周华键',50,90,90)
```

程序在查询分析器中执行，结果如图 8-24 所示。

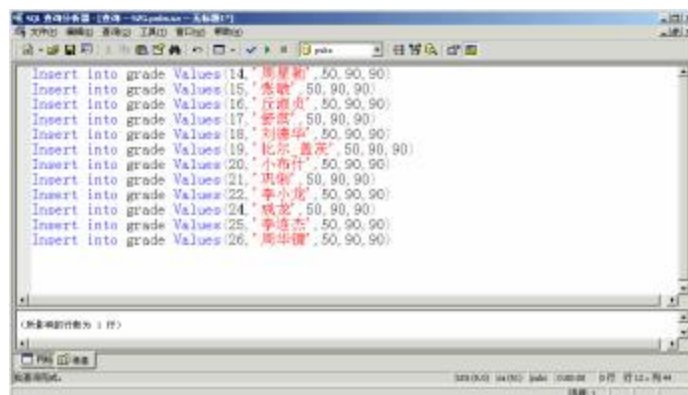


图 8-24 插入测试数据

第 7 章介绍了操作 Access 数据库的三大基本格式，这些格式可以非常方便地用到 SQL Server 数据库表上。

3.5.1 格式一的 SQL Server 版本

其实操作数据库的 SQL 语句都是一样的，链接格式也大同小异。只要格式一的第二步就可以连接并操作 SQL Server 数据库了。

格式一：数据库调用的基本格式

格式说明：利用 Execute 方法建立 RecordSet 对象

```
//第一步：建立 Connection 对象
var conn = Server.CreateObject("ADODB.Connection");
//第二步：使用 Connection 对象的 Open 方法建立数据库连接
conn.Open("driver={SQL Server};database=数据库名;server=服务器名;uid=sa;pwd=");
//第三步：使用 Connection 对象的 Execute 方法执行 SQL 语句
//如果执行查询语句
rs = conn.Execute("数据查询语句");
//如果执行数据操纵语句
conn.Execute("数据操纵语句");
```

可以看出，只有方法 conn.Open 的参数改变了，连接串“driver={SQL Server};database=数据库名;server=服务器名;uid=sa;pwd=”中，driver 是连接数据库的驱动程序名，database 是连接数据库名，server 是连接数据库服务器名，如果连接本地的服务器，服务器名是“localhost”，uid 是用户名，pwd 是密码。改写程序 7-04.asp，连接 SQL Server 数据库，把 pubs 数据库中的 grade 数据表显示出来。如程序 8-16.asp 所示。

案例名称：连接 SQL Server 并输出记录

程序名称：8-16.asp

```
<%@ Language=Jscript %>
<HTML>
<BODY>
<%
    var conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={SQL Server};database=pubs;server=localhost;uid=sa;pwd=");
    rs = conn.Execute("Select * from grade");
    rstotab(rs);
    conn.close();
%>
```

```
<%  
function rstotab(rs)  
{  
    Response.write ("<table border=1>");  
    Response.write ("<tr>");  
    //输出表头  
    for (i=0; i<rs.Fields.Count; i++)  
    {  
        Response.Write ("<td>" + rs(i).Name + "</td>");  
    }  
    Response.write ("</tr>");  
  
    //输出表内容  
    while (!rs.EOF)  
    {  
        Response.write ("<tr>");  
        for (i=0; i<rs.Fields.Count; i++)  
        {  
            Response.Write ("<td>" + rs(i) + "</td>");  
        }  
        Response.write ("</tr>");  
        rs.movenext();  
    }  
    Response.write ("</table>");  
}  
%>  
</BODY>  
</HTML>
```

注意：应该启动 SQL Server 数据服务，程序和 7-04.asp 文件只有连接串不一样，其他都一样，执行的结果是把 grade 表的所有记录输出来，如图 8-25 所示。



学号	姓名	语文	数学	英语
14	周星驰	50	90	90
15	张敏	50	90	90
16	丘淑贞	50	90	90
17	舒淇	50	90	90
18	刘德华	50	90	90
19	比尔·盖茨	50	90	90
20	小布什	50	90	90

图 8-25 输出 SQL Server 数据表

这样可以将第 7 章中所有利用格式一的程序改成 SQL Server 版本。因为函数 rstotab 很常用, 可以将函数拿到另一个 ASP 文件中, 在程序中包含这个文件就可以了。如程序 rstotab.asp 所示。

案例名称: 输出记录的函数

程序名称: rstotab.asp

```
<%  
function rstotab(rs)  
{  
    Response.write("<table border=1>");  
    Response.write("<tr>");  
    //输出表头  
    for (i=0; i<rs.Fields.Count; i++)  
    {  
        Response.Write("<td>" + rs(i).Name + "</td>");  
    }  
    Response.write("</tr>");  
  
    //输出表内容  
    while (!rs.Eof)  
    {  
        Response.write("<tr>");  
        for (i=0; i<rs.Fields.Count; i++)  
        {  
            Response.Write("<td>" + rs(i) + "</td>");  
        }  
        Response.write("</tr>");  
        rs.movenext();  
    }  
    Response.write("</table>");  
}  
%>
```

当程序需要输出记录时, 首先将该文件包含进来, 然后调用函数就可以了。如程序 8-17.asp 所示。

案例名称: 包含文件

程序名称: 8-17.asp

```
<%@ Language=Jscript %>  
<!--#include file="rstotab.asp"-->  
<HTML>
```

```

<BODY>
<%
    var conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={SQL Server};database=pubs;server=localhost;uid=sa;
    >wd=");
    rs = conn.Execute("Select * from grade");
    rstotab(rs);
    conn.close();
%>

```

执行的结果依然将全部数据输出到浏览器上。利用格式二同样可以执行 SQL 语句的 Insert 语句、Update 语句和 Delete 语句。

3.5.2 格式二的 SQL Server 版本

格式二也只要修改第二步连接串就可以了，其他的不用改变，如下所示。

格式二：数据库调用的基本格式

格式说明：利用 RecordSet 对象打开数据库表

```

//第一步：建立 Connection 对象
var conn = Server.CreateObject("ADODB.Connection");
//第二步：使用 Connection 对象的 Open 方法建立数据库连接
conn.Open("driver={SQL Server};database=数据库名;server=服务器名;uid=sa;
>wd=");
//第三步：建立 RecordSet 对象
var rs = Server.CreateObject("ADODB.Recordset");
//第四步：利用 RecordSet 对象的 Open 方法打开数据库
rs.Open("SQL 语句", conn, 打开方式, 锁定方式);

```

案例 8-1：分页显示的 SQL Server 版本

下面将分页显示版本六 V6.asp 文件改成 SQL Server 版本，让程序连接 SQL Server 数据库 pubs，并将 grade 表分页显示出来，如程序 V6.asp 所示。

案例名称：分页显示的 SQL Server 版本

程序名称：V6.asp

```

<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={SQL Server};database=pubs;server=localhost; uid= sa;
    >wd=");
    rs = Server.CreateObject("ADODB.Recordset");

```

```

sql = "Select * from grade";
rs.Open(sql, conn, 3);
rs.PageSize = 4;
if (Request("page").Count > 0)
{
    iPage = parseInt(Request("page"));
    if(iPage < 1)
        iPage = 1;
    if(iPage > rs.PageCount)
        iPage = rs.PageCount;
}
else
{
    iPage = 1;
}
Response.Write("当前第" + iPage + "页, 共" + rs.PageCount + "页");
rs.AbsolutePage = iPage;

%>
<table cellpadding="2" bordercolor="Black" border="1">
<tr style="background-color:#AAAADD;">
    <td>学号</td><td>姓名</td><td>数学</td><td>语文</td><td>英语</td>
</tr>
<%
for( i = 0; i < rs.PageSize; i++)
{
    if(!rs.Eof)
    {
        if(i%2==0)
            Response.Write("<tr style='background-color:#FFFFCD;'>");
        else
            Response.Write("<tr>");
        Response.Write("<td>" + rs("学号") + "</td>");
%>
<TD><A HREF='detail.asp?xuehao=<%=rs("学号")%>' TARGET='_blank'>
<%=rs("姓名")%></A></TD>

<%
        Response.Write("<td>" + rs("数学") + "</td>");
        Response.Write("<td>" + rs("语文") + "</td>");

```



```

        Response.Write("<td>" + rs("英语") + "</td>");
        Response.Write("</tr>");
        rs.movenext();
    }
}
%>
</table>
<br><br><br>
<%
if (iPage != 1)
{
%>
<a href="V6.asp?page=1">第一页</a>
<a href="V6.asp?page=<%=iPage-1%>">上一页</a>
<%
}
if (iPage != rs.PageCount)
{
%>
<a href="V6.asp?page=<%=iPage+1%>">下一页</a>
<a href="V6.asp?page=<%=rs.pageCount%>">最后页</a>
<%
}
conn.close();
%>

```

依然只有一条语句发生了变化,即连接串。程序显示的结果如图 8-26 所示。



图 8-26 分页显示

当单击超级链接时,就会显示该行的详细信息,调用 detail.asp 文件,当然 detail.asp 也必须将连接串变过来。如图 8-27 所示。

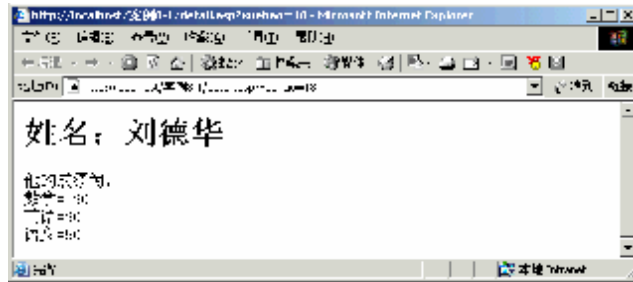


图 8-27 详细信息

3.5.3 格式三的 SQL Server 版本

格式三的格式也是一样的，只需要修改其中的一条语句，如下所示。

格式三：数据库调用的基本格式

格式说明：利用 Command 对象的 Execute 方法建立 RecordSet 对象

```
//第一步：建立 Connection 对象
var conn = Server.CreateObject("ADODB.Connection");
//第二步：使用 Connection 对象的 Open 方法建立数据库连接
conn.Open("driver={SQL Server};database=数据库名;server=服务器名;uid=sa;pwd=");
//第三步：建立 Command 对象
cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
cmd.CommandText = sql;
//第四步：使用 Command 对象的 Execute 方法执行 SQL 语句
//如果执行查询语句
rs = cmd.Execute();
//如果执行数据操纵语句
cmd.Execute();
使用方法如程序 8-18.asp 所示。
```

案例名称：使用 Command 对象操作 SQL Server 数据库

程序名称：8-18.asp

```
<%@ Language=Jscript %>
<!--#include file="rstotab.asp"-->
<HTML>
<BODY>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={SQL Server};database=pubs;server=localhost; uid= sa;
```

```

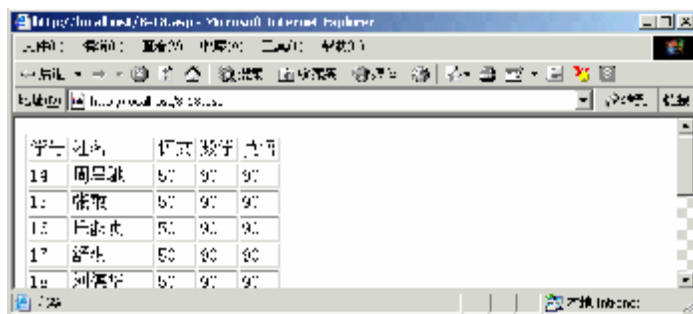
>wd="");

cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
sql = "Select * From grade where 数学 < 200";
cmd.CommandText = sql;
rs = cmd.Execute();
rstotab(rs);

%>

```

程序执行相同，都将显示所有记录，如图 8-28 所示。



学号	姓名	数学	语文	英语
14	周昌洪	50	90	90
15	张聚	50	90	90
16	王洪成	50	90	90
17	潘伟	50	90	90
18	刘喜华	50	90	90

图 8-28 利用格式三连接 SQL Server 数据库

格式三还有一个非常重要的用途是调用 SQL Server 的存储过程。

3.6 SQL Server 存储过程

存储过程对任何数据库来说都是非常重要的。数据库开发人员和数据库管理员会经常编写自己的存储过程，以便运行一般的管理任务或者应用复杂的业务规则。这些类型的过程中可以包括流程控制结构、数据更改或者数据检索语句及错误处理语句。

3.6.1 存储过程的概念

存储过程是 SQL 语句和可选控制流语句的预编译集合，以一个名称存储并作为一个单元处理。存储过程存储在数据库内，可由应用程序通过一个调用执行，而且允许用户声明变量、与条件执行及其他强大的编程功能。

3.6.2 存储过程的例子

可以使用 Create Procedure 关键字创建存储过程。程序 8-19.sql 创建了一个带三个输入参数的存储过程，三个输入参数都含有默认值。

案例名称：创建存储过程

程序名称：8-19.sql

```
use pubs
GO
CREATE PROCEDURE demo_proc
(@name char(16)='SQL Server',@major int = 7,@minor int =0)
AS
```

```
PRINT @name + STR(@major,5) + '.' + STR(@minor,5)
```

程序执行的结果如图 8-29 所示。

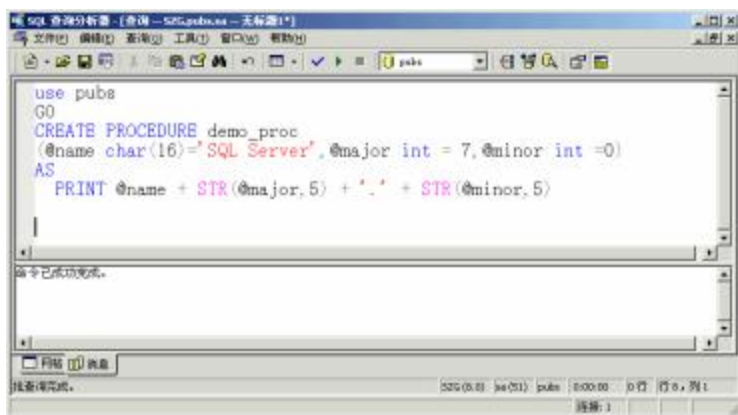


图 8-29 创建存储过程

该存储过程包含三个输入参数，而且三个参数都有默认值。默认值有什么用呢？如果在调用存储过程的时候没有给参数，就用默认值。存储过程创建完以后可以在查询分析器中调用。如程序 8-20.sql 所示。

案例名称：调用存储过程

程序名称：8-20.sql

```
use pubs
GO
demo_proc
EXECUTE demo_proc DEFAULT,7
EXECUTE demo_proc 'Oracle',8
EXECUTE demo_proc DEFAULT,7,DEFAULT
EXECUTE demo_proc 'Oracle',8,DEFAULT
EXECUTE demo_proc 'Oracle',8,1
EXECUTE demo_proc @major=8,@name='Oracle',@minor=0
EXECUTE demo_proc @major=7
```

上面的每行语句都是调用存储过程的语句。调用存储过程一般可以用 **Execute** 命令来调用。如果指示调用时使用默认值，可以使用 **Default** 关键字，但是参数的顺序必须和原存储过程的顺序一致。利用上面的每一条语句调用，体会其调用方法。其中一条的执行结果如图 8-30 所示。

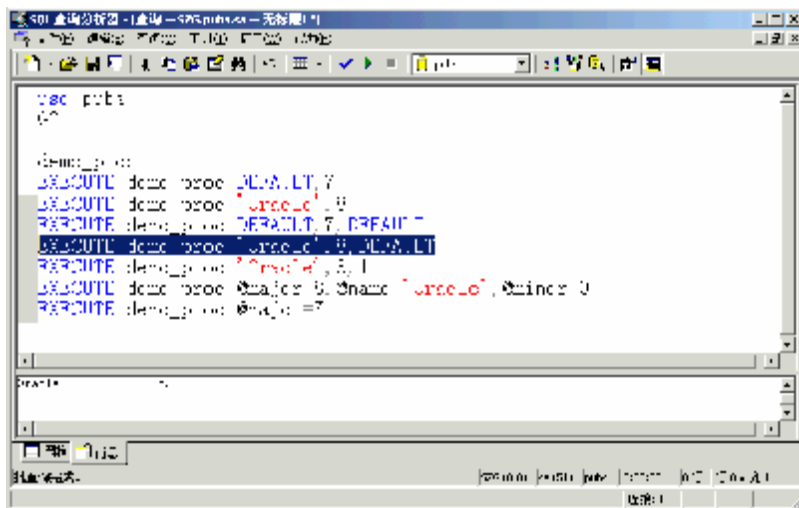


图 8-30 调用存储过程

存储过程可以使用 SQL 中的语句。下面的存储过程中使用了 Select 语句和一个输入参数。
 程序 8-21.sql 所示。

案例名称：创建带有 Select 语句的存储过程

程序名称：8-21.sql

```
use pubs
GO
Create proc GetEmployeeCount
@v_hiredate DateTime
as
Print 'Number of Employees recruited after the input date'
select count(*) from Employee where hire_date>@v_hiredate
```

“select count(*)” 统计符合条件记录的条数，存储过程创建的结果如图 8-31 所示。

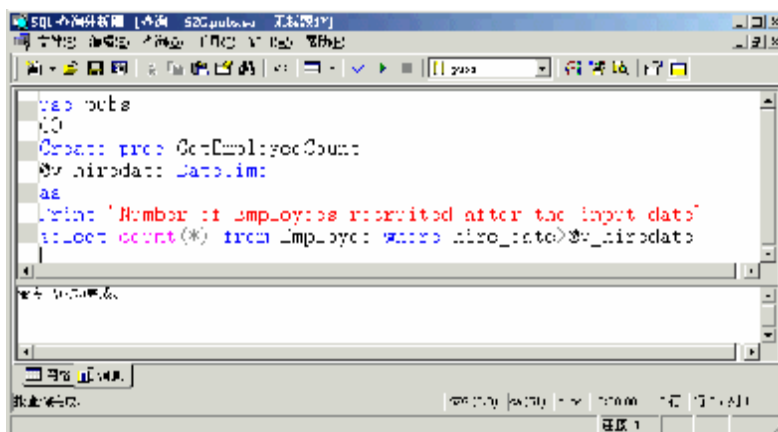


图 8-31 带 Select 语句的存储过程

利用程序 8-22.sql 调用创建好的存储过程。

案例名称：调用存储过程

程序名称：8-22.sql

--调用存储过程

```
execute GetEmployeeCount '01/01/1993'
```

该存储过程的功能是统计 1993 年 1 月 1 日以后入职职工的人数。程序成功调用存储过程，如图 8-32 所示。

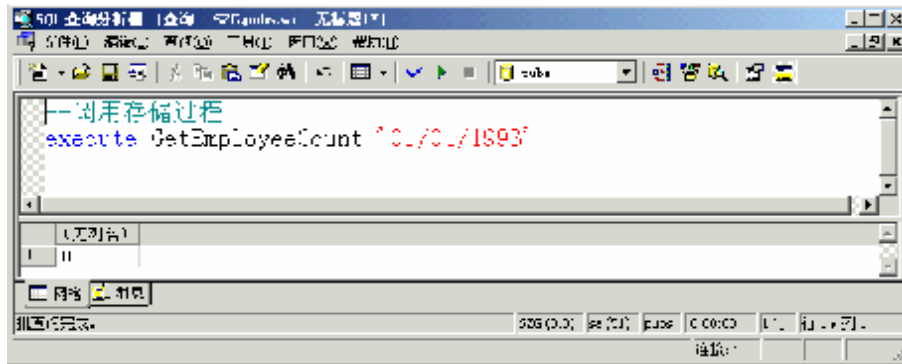


图 8-32 执行存储过程

如存储过程以后不再使用，可以利用下面程序提供的方法删除。注意在查询分析器中“--”是注释的意思。如程序 8-23.sql 所示。

案例名称：删除存储过程

程序名称：8-23.sql

--删除存储过程

```
drop proc GetEmployeeCount
```

程序执行结果如图 8-33 所示。

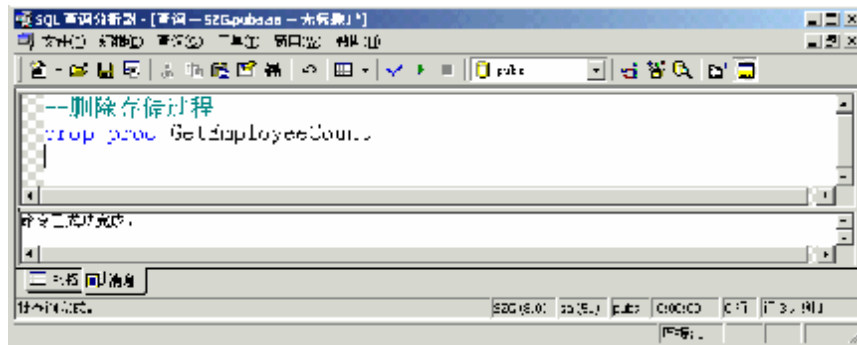


图 8-33 删除存储过程

存储过程可以带输入输出参数，利用下面的存储过程来说明如何同时使用输入和输出参数，该存储过程十分常用，可以用来做密码验证。首先创建用户表，如程序 8-24.sql 所示。

案例名称：创建数据表

程序名称：8-24.sql

```
--创建表
use pubs
go
create table WebUsers
(
    username varchar(20),
    userpass varchar(10)
)
--向表中添加数据
insert into webusers values('aa','aa')
insert into webusers values('bb','bb')
```

两条 Insert 语句分别插入了两条记录，用户“aa”的密码是“aa”，执行的结果如图 8-34 所示。

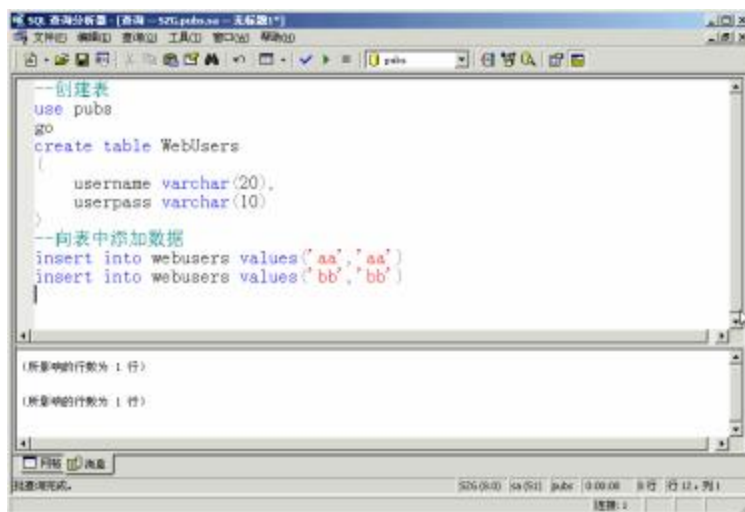


图 8-34 创建数据库表

有了这个表作为基础，下面利用程序 8-25.sql 创建存储过程。

案例名称：创建存储过程

程序名称：8-25.sql

```
--创建存储过程
CREATE PROCEDURE sp_CheckPass
(@CHKName VARCHAR(30),@CHKPass VARCHAR(30),@ISValid varchar(12) OUTPUT)
AS
IF EXISTS(SELECT UserName FROM WebUsers WHERE UserName=@CHKName AND UserPass=
)CHKPass)
```

```
SELECT @ISValid='Good'  
ELSE  
SELECT @ISValid='Bad'  
程序执行的结果如图 8-35 所示。
```

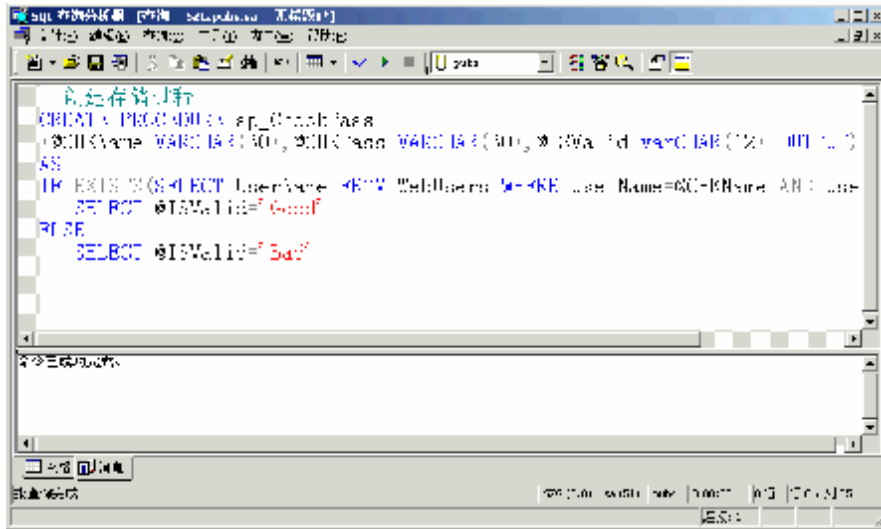


图 8-35 创建带输入输出参数的存储过程

存储过程创建成功后，利用程序 8-26.sql 来测试存储过程。

案例名称：测试存储过程

程序名称：8-26.sql

```
--调用存储过程  
declare @aa Varchar(12)  
exec sp_CheckPass 'aa','aa',@aa output  
select @aa '返回值'  
测试的结果如图 8-36 所示。
```

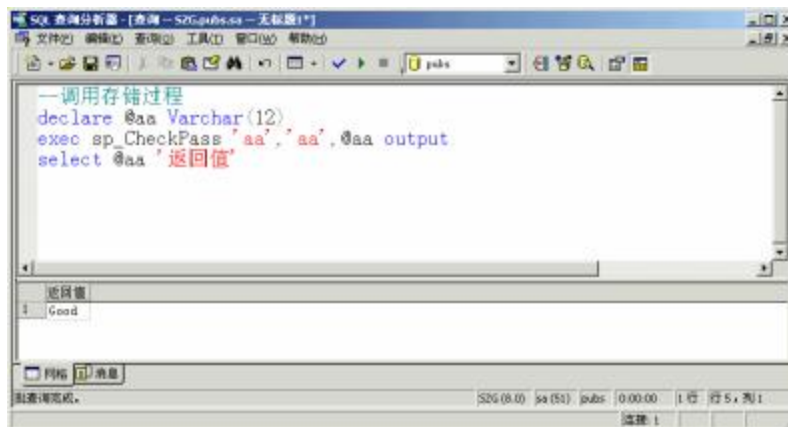


图 8-36 执行存储过程

可以看出,如果能在表中查到这个用户的信息,返回的值就是 Good,反之则返回 Bad。最后调用存储过程。如果用户名和密码不正确,这时则返回 Bad,返回 Bad 的结果如图 8-37 所示。

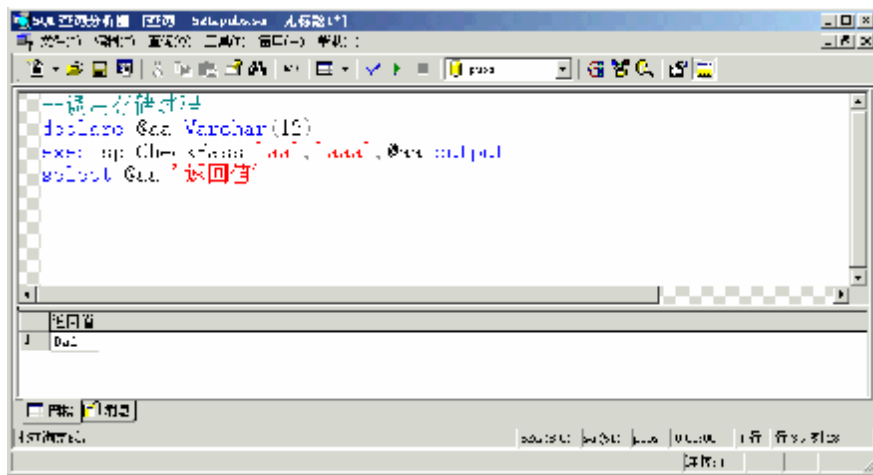


图 8-37 调用存储过程

3.7 ADO 操作 SQL Server 存储过程

存储过程是 SQL Server 数据库的一个最重要的特色,可以利用 Command 对象方便地调用 SQL Server 的存储过程,为什么要利用存储过程呢?

SQL 存储过程执行起来比 SQL 命令文本快得多。当一个 SQL 语句包含在存储过程中时,服务器不必在每次执行它时都要分析和编译它。

可以在多个网页中调用同一个存储过程,使站点易于维护。如果一个 SQL 语句需要做某些改动,只要做一次即可。

可以在存储过程中利用 T-SQL 的强大功能。一个 SQL 存储过程可以包含多个 SQL 语句。也可以使用变量和条件。这样就意味着可以建立非常复杂的检索或者操作数据库的方法。

3.7.1 调用无输入输出参数存储过程

简而言之,能用存储过程时就要用存储过程。存储过程有着极大的优点,也是 SQL Server 数据库的生命力所在。应学会如何利用 Command 来调用存储过程。

现在开始利用 Command 对象来执行一个现有的 SQL Server 的存储过程,在安装的 SQL Server 时,微软有一个作为测试的 pubs 数据库,里面有几个存储过程。打开 SQL Server 的数据库,里面有一个叫 pubs 的库,pubs 库里面有一个叫 Stored procedure 的选项,打开有一个叫 spTq1 的存储过程,双击它显示存储过程的内容如图 8-38 所示。

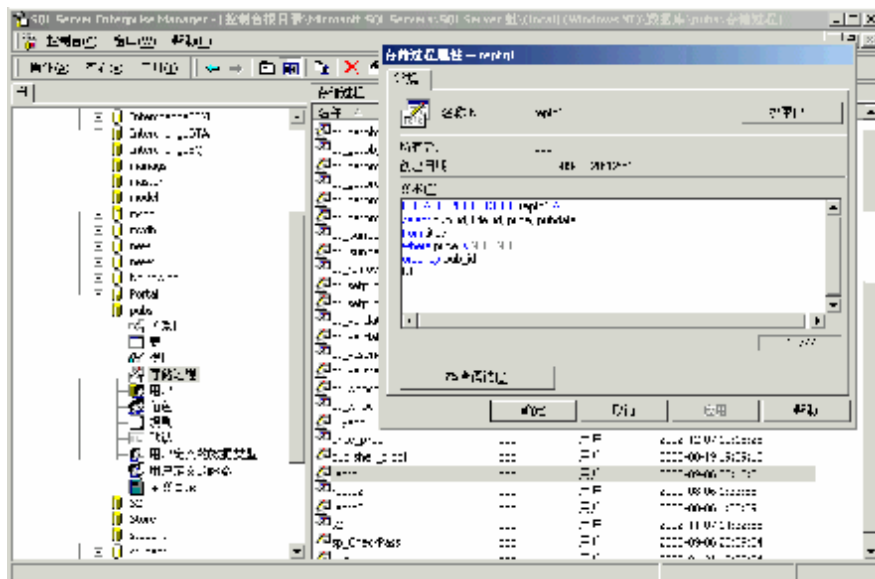


图 8-38 存储过程

可以看到该存储过程的内容如程序 8-27.sql 所示。

案例名称: SQL Server 的存储过程

程序名称: 8-27.sql

```
CREATE PROCEDURE reptql AS
select pub_id, title_id, price, pubdate
from titles
where price is NOT NULL
order by pub_id
GO
```

使用下面的程序来调用这个存储过程。在调用该存储过程时, 需要告诉 Command 对象要调用的是存储过程。利用程序 8-28.asp 来说明如何利用 ADO 实现存储过程的调用。

案例名称: 调用 SQL Server 的存储过程

程序名称: 8-28.sql

```
<%@ Language=Jscript %>
<!--#include file="rstotab.asp"-->
<%

conn = Server.CreateObject("ADODB.Connection");
conn.Open("driver={SQL Server};database=pubs;server= localhost;uid =sa;
pwd=");

cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
cmd.CommandType=4;
cmd.CommandText = "reptql";
```

```
rs = cmd.Execute();
rstotab(rs);
%>
```

“cmd.CommandType=4;”中，4 代表要连接的是 SQL Server 的存储过程。
“cmd.CommandText = "reptq1";”中将存储过程的名字赋值给 CommandText 属性。程序显示的结果如图 8-39 所示。

pub id	title id	price	concatenate
0785	P22273	11.95	Fri Jan 31 00:00:00 UTC+0800 1991
0785	P22273	45.2	Sat Jan 12 00:00:00 UTC+0800 1991
0785	P22003	23	Fri Dec 5 00:00:00 UTC+0800 1991
0785	P22032	79.99	Sat Jan 12 00:00:00 UTC+0800 1991
0785	P22004	30.95	Sat Jan 12 00:00:00 UTC+0800 1991
0785	P22013	23.5	Fri Dec 2 00:00:00 UTC+0800 1991
0785	P22002	41.3	Sat Jan 12 00:00:00 UTC+0800 1991

图 8-39 调用 SQL Server 存储过程

3.7.2 调用带输入输出参数的存储过程

8.7.1 节中只是调用了一个没有输入输出参数的存储过程，如何在调用存储过程的时候使用输入输出参数？如下所示，程序 8-29.asp 调用了 8.6.2 小节中的存储过程 “sp_CheckPass”。

案例名称：调用存储过程的输入和输出参数

程序名称：8-29.asp

```
<%@ Language=Jscript %>
<!--#include file="adojavas.inc"-->
<%

conn = Server.CreateObject("ADODB.Connection");
conn.Open("driver={SQL Server};database=pubs;server=localhost;uid =sa;
pwd=");

cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
cmd.CommandType=4;
cmd.CommandText = "sp_CheckPass";
//创建存储过程的两个输入参数
//adVarChar 和 adParamInput 定义在文件 adojavas.inc 中
param = cmd.CreateParameter("CHKName", adVarChar, adParamInput, 30);
cmd.Parameters.Append(param);
param = cmd.CreateParameter("CHKPass", adVarChar, adParamInput, 30);
```

```
cmd.Parameters.Append(param);  
//给存储过程两个输入参数赋值  
cmd("CHKNAME") = "aa";  
cmd("CHKPass") = "aa";  
//创建存储过程输出参数  
param = cmd.CreateParameter("ISValid", adVarChar, adParamOutput, 30);  
cmd.Parameters.Append(param);  
//执行存储过程  
cmd.Execute();  
//输出返回值  
Response.Write(cmd("ISValid"));
```

%>

用户名和密码的参数都是“aa”，这时存储过程的返回值是“Good”，程序执行的结果如图 8-40 所示。

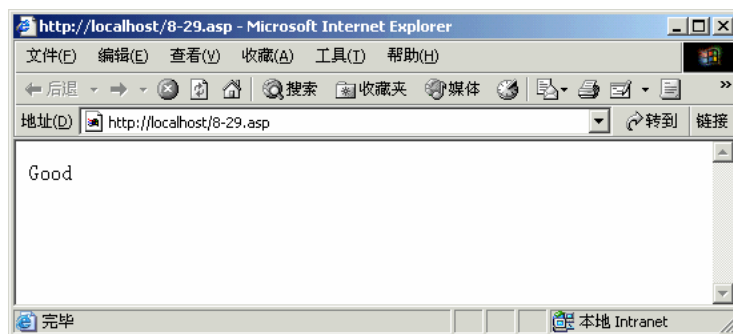


图 8-40 调用带输入输出参数的存储过程

修改密码为其他的字符串，可以看出这时返回的值的“Bad”，其他的程序就可以根据返回值的不同的判断该用户是否合法。如图 8-41 所示。

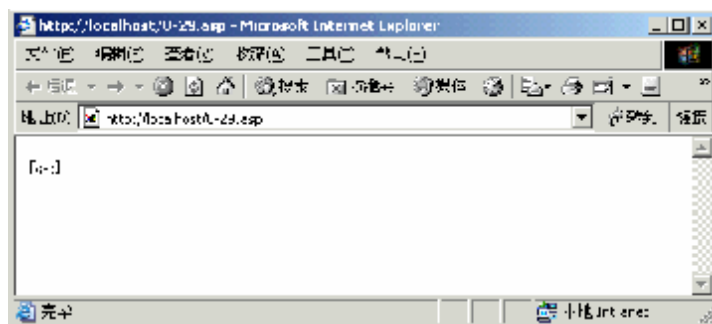


图 8-41 调用带输入输出参数的存储过程

案例 8-2：通讯录

通讯录的功能是保存手机号码、E-mail 号码、住宅电话等常用的联系信息，并提供录入的界面增加用户，实现信息的分页显示，每页显示一个人的信息。为了实现这些功能，首先编

号 SQL 脚本，如程序 txl.sql 所示。

案例名称：通讯录数据库的 SQL 脚本

程序名称：txl.sql

```
use pubs
go
Create Table txl
(
    tID int identity(10000,1) primary key, --惟一标识一行
    tName Varchar(50) not null,
    tE-mail Varchar(50) ,
    tMobile Char(11),
    tHomePhone Varchar(12),
    tMemo Varchar(2000)--备注
)

Insert into txl
values('岳明胜','yms@sohu.com','13681246528','62349087','一个好人')
```

打开查询分析器，执行的结果如图 8-42 所示。

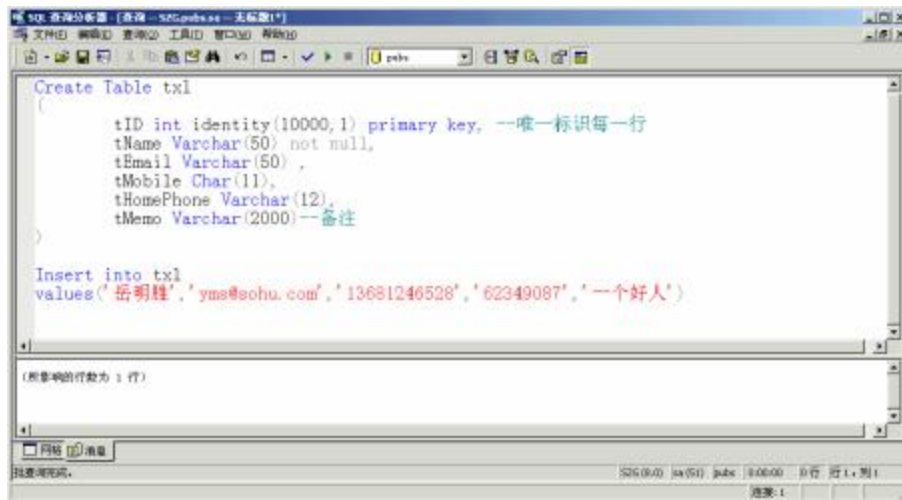


图 8-42 执行 SQL 脚本

通讯录包含以下三个文件。

- (1) index.asp: 通讯录的首页，实现分页显示。
 - (2) do_add.asp: 通讯录的信息录入界面。
 - (3) do_addsumit.asp: 处理录入界面的数据，并保存到数据库中。
- index.asp 实现信息的分页显示，如图 8-43 所示。



图 8-43 通讯录首页

首页程序如程序 index.asp 所示。

案例名称：通讯录首页文件

程序名称：index.asp

```
<%@ Language=Jscript %>
<%
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={SQL Server};database=pubs;server=localhost;uid =sa;
pwd=");
    rs = Server.CreateObject("ADODB.Recordset");
    sql = "select * from txl order by tID desc";
    rs.Open(sql, conn, 3);
    rs.PageSize = 1;
    if (Request("page").Count > 0)
    {
        iPage = parseInt(Request("page"));
        if(iPage < 1)
            iPage = 1;
        if(iPage > rs.PageCount)
            iPage = rs.PageCount;
    }
    else
    {
        iPage = 1;
    }
    rs.AbsolutePage = iPage;

%>
<HTML>
<HEAD>
```

```

<TITLE>通讯录</TITLE>

</HEAD>

<BODY>

<P ALIGN="CENTER"><FONT SIZE="7" COLOR="#008000">通讯录</FONT></P>

<P ALIGN="LEFT"><A HREF="DO_ADD.ASP">添加</A></P>

<%
for( i = 0; i < rs.PageSize; i++)
{
    if(!rs.EOF)
    {
        %>
<TABLE BORDER="1" CELSPACING="1" BORDERCOLOR="#111111" WIDTH="100%">
    <TR>
        <td width="25%">姓名</td>
        <td width="25%"><%=rs( "tName" )%></td>
        <td width="25%">手机号码:</td>
        <td width="25%"><%=rs( "tMobile" )%></td>
    </tr>
    <tr>
        <td width="25%">电子邮件</td>
        <td width="25%">
            <a href="mailto:<%=rs( "tE-mail" )%>"><%=rs( "tE-mail" )%></a></td>
        <td width="25%">住宅电话:</td>
        <td width="25%"><%=rs( "tHomePhone" )%></td>
    </tr>
    <tr>
        <td width="25%">备注</td>
        <td width="25%"> </td>
        <td width="25%"> </td>
        <td width="25%"> </td>
    </tr>
    <tr>
        <td width="100%" colspan="4"><%=rs( "tMemo" )%></td>
    </tr>
</TABLE>
<BR><HR><BR>

<%
        rs.movenext( );
    }
}

```

```

%>

<%
if (iPage != 1)
{
%>
<a href="index.asp?page=1">第一页</a>
<a href="index.asp?page=<%=iPage-1%>">上一页</a>
<%
}
if (iPage != rs.PageCount)
{
%>
<a href="index.asp?page=<%=iPage+1%>">下一页</a>
<a href="index.asp?page=<%=rs.pageCount%>">最后一页</a>
<%
}
conn.close();
%>
</BODY>
</HTML>

```

单击“添加”按钮，可以输入新用户的信息。在文本框中输入信息，如图 8-44 所示。



图 8-44 信息录入界面

信息录入界面是一个标准的 HTML 文件，如程序 do_add.asp 所示。

案例名称：通讯录信息录入界面

程序名称：do_add.asp

```

<HTML>

<HEAD>

</HEAD>

<BODY>

<FORM METHOD="POST" ACTION="DO_ADDSUMIT.ASP">
<TABLE BORDER="1" CELLSPACING="1" BORDERCOLOR="#111111"

```



```

WIDTH="100%" >
<TR>
  <TD WIDTH="25%">姓名</TD>
  <TD WIDTH="25%"><INPUT TYPE="TEXT" NAME="tusername" SIZE="20">
  </TD>
  <TD WIDTH="25%">手机号码:</TD>
  <TD WIDTH="25%"><INPUT TYPE="TEXT" NAME="tmobile" SIZE="20"></TD>
</TR>
<TR>
  <TD WIDTH="25%">电子邮件</TD>
  <TD WIDTH="25%"><INPUT TYPE="TEXT" NAME="tE-mail" SIZE="20"></TD>
  <TD WIDTH="25%">住宅电话:</TD>
  <TD WIDTH="25%"><INPUT TYPE="TEXT" NAME="tphone" SIZE="20"></TD>
</TR>
<TR>
  <TD WIDTH="25%">备注</TD>
  <TD WIDTH="25%"> </TD>
  <TD WIDTH="25%"> </TD>
  <TD WIDTH="25%"> </TD>
</TR>
<TR>
  <TD WIDTH="100%" COLSPAN="4">
    <TEXTAREA ROWS="5" NAME="tmemo" COLS="40"></TEXTAREA></TD>
</TR>
</TABLE>
<P><INPUT TYPE="SUBMIT" VALUE="提交" NAME="B1">
<INPUT TYPE="RESET" VALUE="重置" NAME="B2"></P>
</FORM>
</BODY>
</HTML>

```

添加完数据以后,单击“提交”按钮调用 do_addsumit.asp 处理数据,并提交到数据库,如程序 do_addsumit.asp 所示。

案例名称: 通讯录信息录入处理程序

程序名称: do_addsumit.asp

```

<%@ Language=Jscript %>
<%
  //读取文本框的值
  tusername = Request("tusername")(1);
  tmobile    = Request("tmobile")(1);

```

```
tE-mail      = Request("tE-mail")(1);
tphone       = Request("tphone")(1);
tmemo        = Request("tmemo")(1);
//连接数据库并插入数据
conn = Server.CreateObject("ADODB.Connection");
conn.Open("driver={SQL Server};database=pubs;server=localhost; uid= sa;
pwd=");

cmd = Server.CreateObject("ADODB.Command");
cmd.ActiveConnection = conn;
sql = "Insert into txl values('" + tusername + "','" + tE-mail +
      "','" + tmobile + "','" + tphone + "','" + tmemo + "')"
cmd.CommandText = sql
cmd.Execute();
//返回
Response.Redirect("index.asp");

%>
```

程序首先将数据读取出来,利用 Insert 语句将信息写到数据库中,然后自动转到显示页面,则可以看到刚才添加的数据。如图 8-45 所示。

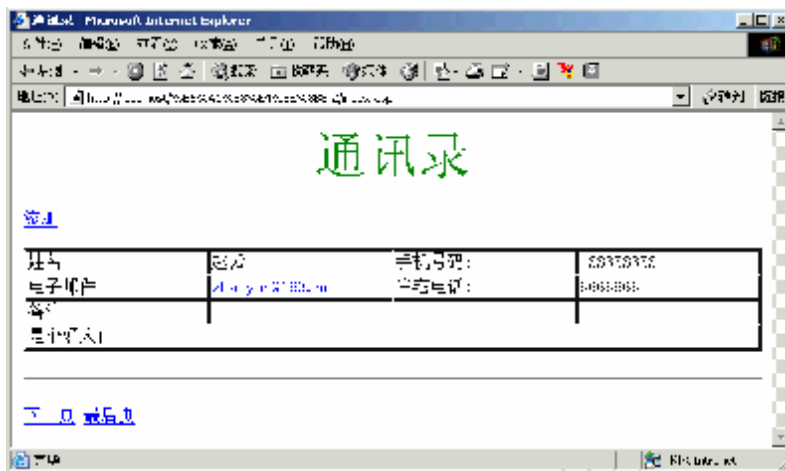


图 8-45 通讯录首页

小结

本章在第 7 章的基础上继续介绍使用 ADO 操作数据库。本章需要掌握如何在 SQL Server 系统建立数据库和数据库表。熟练掌握保持数据库完整性方法、三种操作 SQL Server 数据库的格式和操作 Access 数据库的差异、如何建立和调用 SQL Server 存储过程、如何利用 ADO 周用存储过程。

课后习题和上机练习

1. SQL Server 与 Access 的联系和区别？
2. 如何在 SQL Server 查询分析中建立数据库和数据库表？
3. 简述 Identity 属性的功能。
4. 六大约束包括哪些？如何使用？外键约束的功能是什么？
5. 比较操作 SQL Server 数据库的三大基本格式和操作 Access 的三大基本格式的异同。
6. 存储过程有什么作用？如何建立和调用存储过程？
7. 如何利用 ADO 调用带参数的存储过程？程序如何与存储过程传递参数？
8. 完善案例 8-2，添加功能：（1）模糊查找某用户；（2）修改某人的信息；（3）删除某人的信息。（上机练习）

第 9 章 ASP 操作 XML 文件

本章要点

可扩展标记语言 (eXtensible Markup Language, XML) 是目前应用开发领域中的热门技术。本章主要讲述 XML 基本概念, XML 的三种显示样式: CSS (Cascading Style Sheet, 层叠式样式表单)、XSL(Extensible Style Language, 扩展的标记语言)和 Data Island (数据岛) 及如何利用 ASP 操作 XML 文件。

9.1 XML 的概念

XML 即可扩展的标记语言, 可以定义语义标记, 是元标记语言。

XML 不像超文本标记语言 HTML, HTML 只能使用规定的标记, 对于 XML, 用户可以定义自己需要的标记。假如用户定义和简历的相关信息, 需要描述的是姓名、性别、工作经历, 学历, 等等, 就可以为每项信息定义一个标记。如程序 9-01.xml 所示。

案例名称: 元标记语言

程序名称: 9-01.xml

```
<?xml version="1.0" encoding="gb2312"?>
<老师>
    <姓名>zhourunfa</姓名>
    <性别>male</性别>
    <职业>Teacher Peking Univ.</职业>
</老师>
```

直接用浏览器打开该 XML 文件, 显示成默认的树状结构, 如图 9-1 所示。



图 9-1 XML 文件显示样式

注意: 浏览器的版本号必须是 IE 5.0 以上。

9.2 编写 XML 文档

使用 XML 可以方便地格式化数据，但是必须符合 XML 文件的格式。

9.2.1 定义基本元素

为了把数据库记录转换为 XML 文档，首先要确定一个根元素，在这里可以使用<老师记录>作为文档元素，其中包含一个员工的所有信息内容。接着，可以把员工的姓名放到<姓名>元素中，把地址放到<地址>元素中。如程序 9-02.xml 文件所示。

案例名称：定义基本元素

程序名称：9-02.xml

```
<?xml version="1.0" encoding="gb2312"?>
<老师记录>
  <姓名>周润发</姓名>
  <住址>北京大学</住址>
  <职位>计算机系老师</职位>
  <工资>2000</工资>
</老师记录>
```

利用浏览器打开程序，仍然显示成标准的树状结构，如图 9-2 所示。

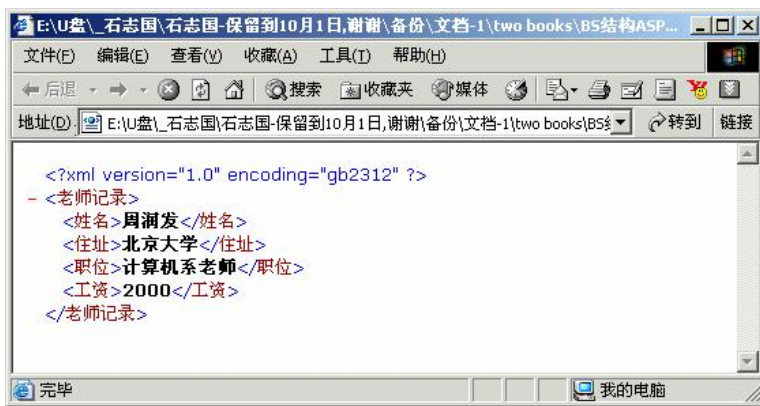


图 9-2 定义基本元素

9.2.2 使用属性

如果有不属于文档的内容或者不需要使用元素进一步表达的内容时，就需要使用属性。比如，如果使用不止一种货币发放工资，就需要在<工资>元素上表明是哪一种币制。可以添加一个名为“货币”的属性来表达这个消息。如果教师分为专职和兼职，如何表示呢？如程序 9-03.xml 所示。

案例名称：添加属性

程序名称：9-03.xml

```
<?xml version="1.0" encoding="gb2312"?>
<老师记录>
    <姓名 类别="专职">周润发</姓名>
    <住址>北京大学</住址>
    <职位>计算机系教师</职位>
    <工资 货币="美元">2000</工资>
</老师记录>
```

显示的结果如图 9-3 所示。



图 9-3 添加属性

9.3 XML 文档结构

一个典型 XML 文件中除了会出现元素和属性外，还会出现其他内容，比如 XML 声明，注释等。程序 9-04.xml 文件给出了一个典型的 XML 文档的结构。

案例名称：典型的 XML 文档结构

程序名称：9-04.xml

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<老师记录>
    <姓名>周润发</姓名>
    <职业>教师</职业>
    <家属>
        <妻子>None</妻子>
    </家属>
    <兴趣>唱歌、跳舞</兴趣>
    <!--这是注释-->
    <电子邮件>shizhiguo@163.net</电子邮件>
</老师记录>
```

利用浏览器打开程序，显示的结果如图 9-4 所示。



图 9-4 XML 文档的显示

XML 文档总体上包括两部分：序言（Prolog）和文档元素（Document Elements）。序言中包含 XML 声明（XML Declaration）、处理指令（Processing Instructions）和注释（Comments）；而文档元素中包含各种元素（Elements）、属性（Attributes）、文本内容（Textual Content）、字符和实体引用（Character and Entity References）、CDATA 区等。

9.3.1 XML 声明

XML 声明是以“<?xml”开始的，程序 9-04.xml 的声明为：

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
```

XML 声明的作用是告诉浏览器将要处理的文档是 XML 文件。一个 XML 文档最好是以一个 XML 声明作为开始。之所以说“最好”是因为 XML 声明在 XML 文档中是可选内容，可加可不加，但 XML 标准强烈推荐加入这一行声明。XML 声明中可以包含 version，encoding 和 standalone 三个属性。

1. version 属性

在 XML 声明中必须包含 version 属性，指明以下文档遵循哪个版本的 XML 规范。该属性必须排在 XML 声明中其他属性之前。由于当前的 XML 最新版本为 1.0，所以在 XML 声明中出现的版本说明无一例外地都是 version="1.0"。

2. encoding 属性

该属性指示文档中字符使用的编码标准。如果文档中使用其他编码规则，则必须使用 encoding 属性指明。在 XML 规范中列出了很多编码类型，一般情况下很多编码用不到，只要知道下面几个常见的编码就可以了。

- (1) GB2312 或者 GBK：简体中文编码。
- (2) BIG5：繁体中文编码。
- (3) UTF-8：压缩的 Unicode 编码。

如果 XML 文档使用中文标记或出现中文内容，就应该在 XML 声明中使用 encoding="gb2312" 属性。

3. standalone 属性

该属性表明该 XML 文档是否和一个外部文档配套使用。如果把这个属性值设为 “yes”。如 “<?xml version="1.0" encoding="gb2312" standalone="yes"?>”，说明这是一个独立的 XML 文档，与外部文件无关联。

1.3.2 注释

注释是对文档结构或内容的解释，不属于 XML 文档的内容，所以 XML 解释器不会处理它们。注释以 “<!--” 开始，以 “-->” 结束。下面一行是注释：

```
<!--最后更新于 2004 年 1 月 1 日 -->
```

解析器碰到 “-->” 时就看做一个注释的结束，接着把后面的内容作为普通 XML 文档处理。所以，字符串 “-->” 不能出现在注释的内部。除了这个限制外，所有其他合法的 XML 字符都可以出现在注释中。

有时候为了暂时不让 XML 解析器处理 XML 文档中的某些内容，就可以在它们的前后加上注释标记。

1.3.3 字符和实体引用

字符和实体引用可以向 XML 文档中引入其他信息，而不需要直接在文档中输入它们。字符和实体引用通常用于以下情况。

- (1) 字符不能直接出现在文档中，因为它们会被解释为标记。
- (2) 由于输入设备的限制，字符不能直接输入到文档中。
- (3) 由于单字节字符的限制，字符不能可靠地经过处理程序。
- (4) 相同的字符串或文档片断在文档中多次使用。

在 XML 中，字符和实体引用以 “&” 开始，以 “;” 结束。例如，要在文档中使用欧元的标记，因为这是键盘上不能直接输入该字符，所以可以向文档插入 “€” 或者 “€”。表 9-1 列出了 XML 规范中常用的实体引用。

表 9-1 XML 常用的实体引用

实 体	实 体 引 用	意 义
lt	<	< (小于)
gt	>	> (大于)
amp	&	& (和)
apos	'	' (单引号)
quot	"	" (双引号)

需要直接输出特殊字符时，如 “<周润发>”，不能直接输出字符，需要使用实体引用。

案例名称：使用特殊字符

程序名称：9-05.xml

```
<?xml version="1.0" encoding="gb2312"?>
<老师记录>
```



```

<姓名 类别="专职">&lt;周润发&gt; </姓名>
<住址>北京大学</住址>
<职位>教师</职位>
<工资 货币="美元">2000</工资>
</老师记录>

```

显示结果如图 9-5 所示。

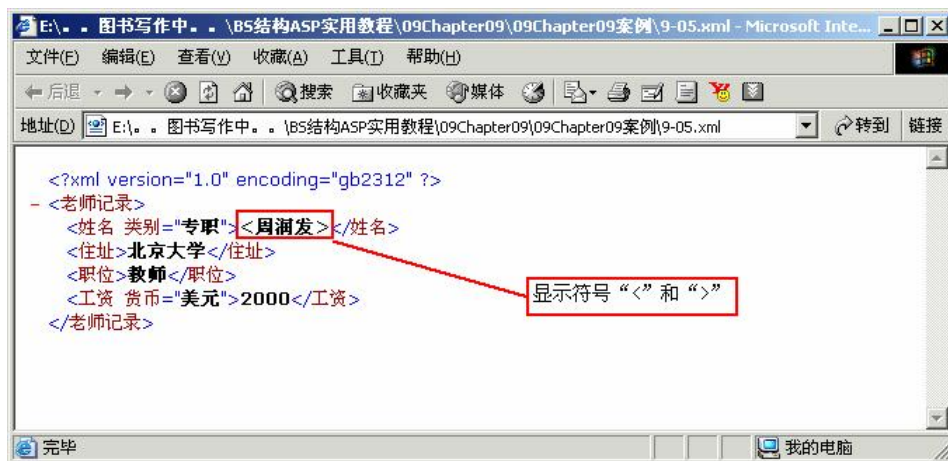


图 9-5 使用特殊字符

9.4 XML 的三种显示格式

显示 XML 文件常见的有三种方式：使用 CSS 样式表，使用 XSL 样式单和使用 XML 的数据岛技术。

9.4.1 CSS 样式表

对 XML 的标记 GREETING 定义显示样式，定义字体的大小为 48pt 和字体加粗。CSS 样式单如程序 9-06.css 所示。

案例名称：CSS 样式表

程序名称：9-06.css

```

GREETING
{
    font-size: 48pt;
    font-weight: bold;
}

```

将该 CSS 文件和 9-07.xml 文件存储到同一个文件文件夹下。如下所示。

案例名称：调用 CSS 的 XML 文件

程序名称: 9-07.xml

```
<?xml version="1.0" encoding="GBK"?>
<?xml-stylesheet type="text/css" href="9-06.css"?>
<GREETING>
    China
</GREETING>
```

利用浏览器打开该 XML 文件, 显示的结果不再是树状结构, 而是按照 CSS 样式表中定义的样式显示, 如图 9-6 所示。



图 9-6 利用 CSS 样式显示 XML 文件

9.4.2 XSL 样式语言

XSL 语言和 CSS 的功能一样, 都是定义样式输出 XML 文件的内容。调用 XSL 样式单的 XML 文件如程序 9-08.xml 文件所示。

案例名称: 调用 XSL 的 XML 文件

程序名称: 9-08.xml

```
<?xml version="1.0" encoding="gb2312"?>
<?xml-stylesheet type="text/xsl" href="9-09.xsl"?>
<persons>
    <person>
        <name>周润发</name>
        <age>25</age>
        <tel>66666666</tel>
    </person>
    <person>
        <name>周慧敏</name>
        <age>26</age>
```

```

        <tel>666666667</tel>
    </person>
    <person>
        <name>周星驰</name>
        <age>28</age>
        <tel>666666669</tel>
    </person>
</persons>

```

要想格式化显示该 XML 文件，需要编写 XSL 样式表，如程序 9-09.xsl 所示。

案例名称：XSL 文件

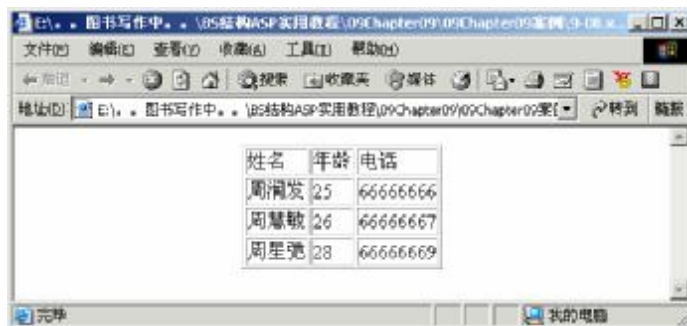
程序名称：9-09.xsl

```

<?xml version="1.0" encoding="GB2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
    <HTML>
    <BODY>
        <CENTER>
            <TABLE BORDER="1">
                <TR>
                    <TD>姓名</TD>
                    <TD>年龄</TD>
                    <TD>电话</TD>
                </TR>
                <xsl:for-each select="persons/person">
                    <TR>
                        <TD><xsl:value-of select="name"/></TD>
                        <TD><xsl:value-of select="age"/></TD>
                        <TD><xsl:value-of select="tel"/></TD>
                    </TR>
                </xsl:for-each>
            </TABLE>
        </CENTER>
    </BODY>
    </HTML>
</xsl:template>
</xsl:stylesheet>

```

将上面的两个文件存储到同一个文件夹下，用浏览器打开该 XML 文件，显示的结果如图 1-7 所示。



The screenshot shows a web browser window with a table containing three rows of teacher data. The table has three columns: 姓名 (Name), 年龄 (Age), and 电话 (Phone). The data is as follows:

姓名	年龄	电话
周润发	25	66666666
周慧敏	26	66666667
周星驰	28	66666669

图 9-7 显示结果

9.4.3 XML 的数据岛技术

IE 5.0 以上的版本才支持 XML 文件的显示, 其实 IE 4.0 的时候就支持 XML 的数据岛技术了, 该技术可以有效的将显示和数据分离, 如程序 9-10.htm 所示。

案例名称: 使用 XML 文件数据岛

程序名称: 9-10.htm

```
<HTML>
<BODY>
  <xml id="xmlid">
    <教师队伍>
      <教师>
        <名字>周润发</名字>
        <课程>编程基础</课程>
        <结论>是个好老师</结论>
      </教师>
      <教师>
        <名字>周慧敏</名字>
        <课程>ASP 技术</课程>
        <结论>是个好老师</结论>
      </教师>
    </教师队伍>
  </xml>

  <TABLE BORDER="0" datasrc="#xmlid" ALIGN="CENTER" WIDTH="443">
    <THEAD>
      <TD BGCOLOR="#99FF99">名字: </TD>
      <TD BGCOLOR="#3399CC">课程: </TD>
      <TD BGCOLOR="#CC99CC">结论: </TD>
    </THEAD>
    <TR>
```

```

<TD BGCOLOR="#99FF99"><SPAN datafld="名字"></SPAN></TD>
<TD BGCOLOR="#3399CC"><SPAN datafld="课程"></SPAN></TD>
<TD BGCOLOR="#CC99CC"><SPAN datafld="结论"></SPAN></TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

注意：该文件的扩展名为 HTML，利用<xml id="xmlid">引入一个 XML 格式的文件，利用 datasrc="#xmlid"将 XML 和表格绑定，必须加上绑定标记“#”，利用绑定到表格的每一行，显示的结果如图 9-8 所示。

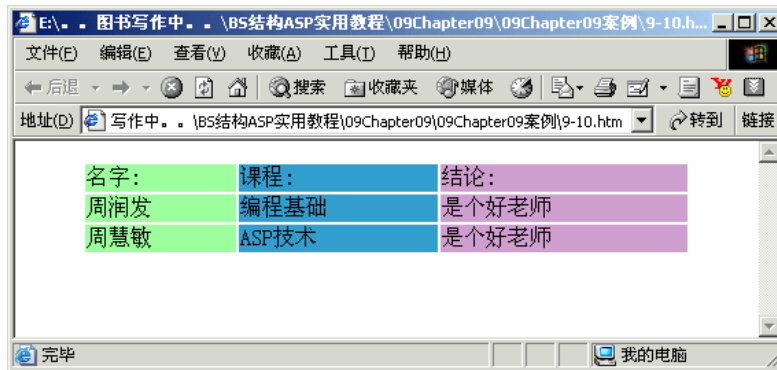


图 9-8 数据岛技术

可以将 XML 标记部分分离出来，变成一个文件，如程序 9-11.xml 所示。

案例名称：使用 XML 数据岛

程序名称：9-11.xml

```

<?xml version="1.0" encoding="gb2312"?>
<教师队伍>
<教师>
  <名字>周润发</名字>
  <课程>编程基础</课程>
  <结论>是个好老师</结论>
</教师>
<教师>
  <名字>周慧敏</名字>
  <课程>.NET 技术</课程>
  <结论>是个好老师</结论>
</教师>
</教师队伍>

```

利用程序 9-12.htm 调用 9-11.xml 文件。如下所示。

案例名称：使用 XML 数据岛

程序名称: 9-12.htm

```
<HTML>
<BODY>
<xml id="xmlid" src="9-11.xml">
</xml>
<TABLE BORDER="0" datasrc="#xmlid" ALIGN="CENTER" WIDTH="443">
  <THEAD>
    <TD BGCOLOR="#99FF99">名字: </TD>
    <TD BGCOLOR="#3399CC">课程: </TD>
    <TD BGCOLOR="#CC99CC">结论: </TD>
  </THEAD>
  <TR>
    <TD BGCOLOR="#99FF99"><SPAN datafld="名字"></SPAN></TD>
    <TD BGCOLOR="#3399CC"><SPAN datafld="课程"></SPAN></TD>
    <TD BGCOLOR="#CC99CC"><SPAN datafld="结论"></SPAN></TD>
  </TR>
</TABLE>
</BODY>
</HTML>
```

显示的结果如图 9-9 所示。



图 9-9 使用 XML 数据岛技术

9.5 使用 XML 组件

在 IE 5.0 中, 包含了 Microsoft XML 2.0 类库, 其中包含了 DOMDocument 对象。在 ASP 中, 可以通过该对象方便地对 XML 对象进行操作。

9.5.1 创建 DOM 对象

要在 ASP 中使用 Microsoft XML 2.0, 必须在服务器端安装 IE 5.0 或 XML 2.0 的插件。如

果使用的是 Win2000 或 win2001 的话, 不需要安装, 要是使用 Windows98/NT 的话, 只要安装一个 IE 5.0 以上的版本就行了。在 ASP 中创建 DOMDocument 对象的基本语法为:

```
"var objXML = Server.CreateObject("Microsoft.XMLDOM");"
```

创建完这个对象后就可以来使用其内部的函数了。

9.5.2 读取 XML 文件

下面的程序说明利用 ASP 读取 XML 文档中的数据的基本的语法格式。首先使用上面的各式创建 Microsoft XMLDom 对象, 如程序 9-12.htm 所示。

案例名称: 使用 DOM 读取 XML 文件

程序名称: 9-12.htm

```
<%@ Language=Jscript %>
<%
    var objXML = Server.CreateObject("Microsoft.XMLDOM");
    objXML.load(Server.MapPath("9-08.xml"));
    var objLst = objXML.getElementsByTagName("person");
    intNoOfHeadlines = objLst.length;
    Response.Write(intNoOfHeadlines);
%>
```

利用 Load 函数装载 XML 文件, 装载完毕后就可以读取了。程序取得 XML 文件中 person 节点的个数。该文件有三个 person 节点, 程序显示结果如图 9-10 所示。

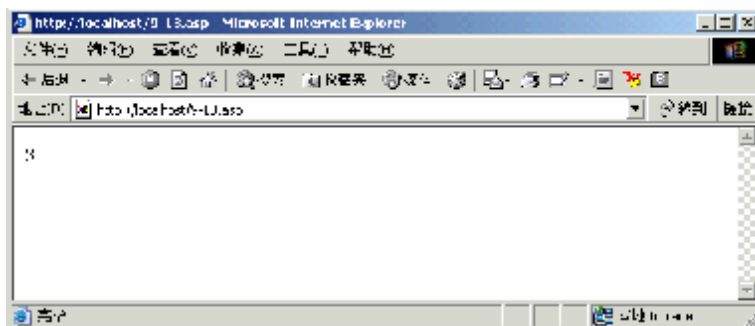


图 9-10 读取节点的个数

案例 9-1: XML 文件的读写

这个案例利用 XMLDom 组件来读取 XML 文件, 实现 XML 文件的信息的读取, 数据载体是 mysites.xml 文件, 如程序 mysites.xml 所示。

案例名称: 数据载体

程序名称: mysites.xml

```
<?xml version="1.0" encoding="gb2312"?>
```

```
<mysites>
  <site>
    <topic>新闻</topic>
    <name>sina</name>
    <url>http://www.sina.com</url>
  </site>
  <site>
    <topic>体育</topic>
    <name>nease</name>
    <url>http://www.163.com</url>
  </site>
  <site>
    <topic>新闻</topic>
    <name>sohu</name>
    <url>http://www.sohu.com</url>
  </site>
  <site>
    <topic>计算机</topic>
    <name>COL</name>
    <url>http://www.263.net</url>
  </site>
  <site>
    <topic>计算机</topic>
    <name>Microsoft</name>
    <url>http://www.microsoft.com</url>
  </site>
  <site>
    <topic>体育</topic>
    <name>sports</name>
    <url>http://www.sports.com</url>
  </site>
  <site>
    <topic>计算机</topic>
    <name>Computer</name>
    <url>http://www.Computer.com</url>
  </site>
  <site>
    <topic>聊天</topic>
    <name>chat</name>
    <url>http://chat.263.net</url>
```



```
</site>
</mysites>
```

根节点是 `mysites`，它有许多一级子节点，每个子节点又包含三个叶节点，分别存储网站的主题，网站的名称和网站的网址。用浏览器打开如图 9-11 所示。



图 9-11 文件显示的结构

程序 `index.asp` 利用 DOM 技术实现 XML 文件的读取，如下所示。

案例名称：读取 XML 文件并以列表显示

程序名称：`index.asp`

```
<%@ Language=Jscript %>
<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY BGCOLOR="BEIGE">
    <%
      var objXML = Server.CreateObject("Microsoft.XMLDOM");
      objXML.load(Server.MapPath("DATA/MYSITES.XML"));
      var objLst = objXML.getElementsByTagName("site");
      intNoOfHeadlines = objLst.length;
    %>
    <TABLE BORDER="1" >
    <%
      for (i=0; i< intNoOfHeadlines; i++)
      {
        objHdl = objLst.item(i);
      }
    %>
    <TR>
      <TD><%=objHdl.childNodes(0).text%></TD>
      <TD><%=objHdl.childNodes(1).text%></TD>
      <TD><%=objHdl.childNodes(2).text%></TD>
    </TR>
```

```

<%
}
%>
</TABLE>
</BODY>
</HTML>

```

程序中，“objHdl.childNodes(0).text”得到 sites 节点下第一个子节点的值，将所有 sites 节点下的内容全部以表格的形式输出来，执行结果如图 9-12 所示。

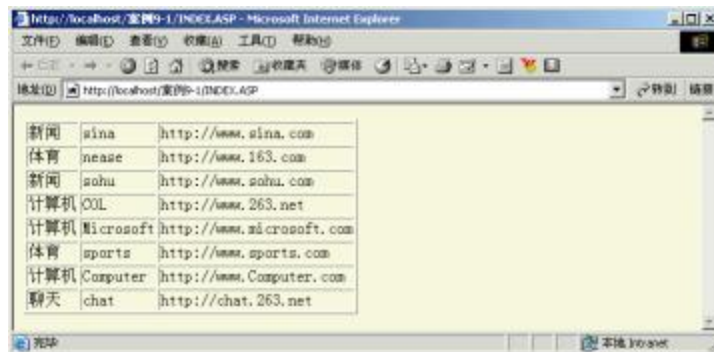


图 9-12 XML 文件得列表显示

利用 DOM 技术也可以为一个 XML 文件追加信息，如程序 add.asp 所示。

案例名称：利用 DOM 写 XML 文件

程序名称：add.asp

```

<%@ Language=Jscript %>
<%
//add.asp 为一自提交页面，第一次进入，要求输入站点信息，然后提交结果给自己
if (Request("action").Count > 0)
{
    if ("add" == Request("action")(1))
    {
        var XmlDoc, Flag, FileName;
        FileName = Server.MapPath("data/mysites.xml");
        XmlDoc = Server.CreateObject("Microsoft.XMLDOM");

        Flag = XmlDoc.load(FileName);
        if (Flag)
        {
            var Element, El2;
            Element = XmlDoc.createElement("site");
            El2 = XmlDoc.createElement("topic");

```

```

        E12.text = Request.form("topic");
        Element.appendChild(E12);

        E12 = XmlDoc.createElement("name");
        E12.text = Request.form("name");
        Element.appendChild(E12);

        E12 = XmlDoc.createElement("url");
        E12.text = Request.form("url");
        Element.appendChild(E12);
        XmlDoc.documentElement.appendChild(Element);
        //保存修改结果
        XmlDoc.save(FileName);
    }
}
%>
<HTML>
    <HEAD>
        <SCRIPT LANGUAGE="JavaScript">
            function valid()
            {
                //输入合法性检测
                var name = document.addform.name.value;
                var url = document.addform.url.value;
                var topic = document.addform.topic.value;
                if (name==" " || url==" " || topic==" ")
                {
                    alert ("请输入站点信息!")
                    return false
                }
            }
        </SCRIPT>
        <TITLE>ADD</TITLE>
    </HEAD>
    <BODY BGCOLOR="BEIGE">
        <TABLE>
            <FORM NAME="addform" ACTION="add.asp?action=add"
            METHOD="post" ONSUBMIT="return valid()">
            <TR>

```

```
<TD>
    TOPIC.:</TD><TD><INPUT TYPE="text" name="topic" size=20>
</TD>
</TR>
<TR>
<TD>
    NAME:</TD><TD><INPUT TYPE="TEXT" NAME="name" SIZE=20>
</TD>
</TR>
<TR>
<TD>URL:</TD>
<TD><INPUT TYPE="TEXT" NAME="url" SIZE=60 VALUE="http://"></td>
</TR>
<TR>
<TD COLSPAN=2 ALIGN="CENTER"><INPUT TYPE="SUBMIT" VALUE="add">
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

程序提供三个输入框，分别对应 XML 文件的三个叶节点，在文本框中输入合适的参数，如图 9-13 所示。

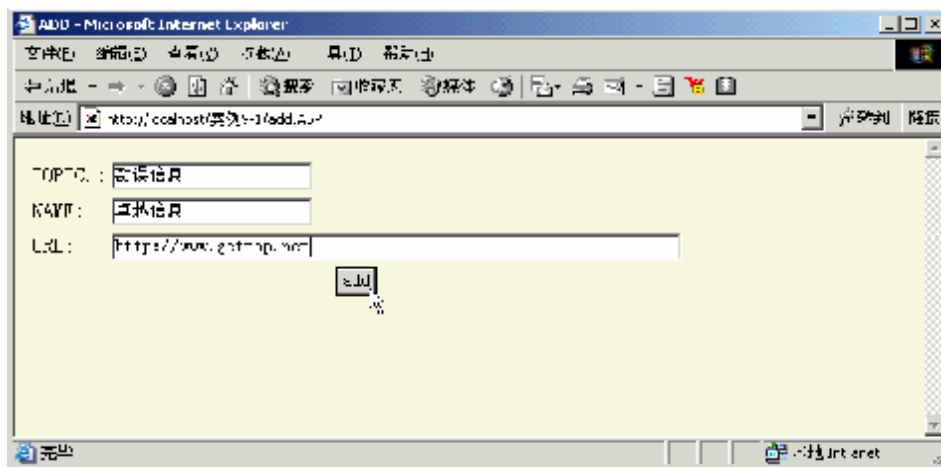


图 9-13 输入界面

单击“add”按钮，就将输入的值写到 XML 文件中去了。可以打开 index.asp 文件来查看输入的值，如图 9-14 所示。

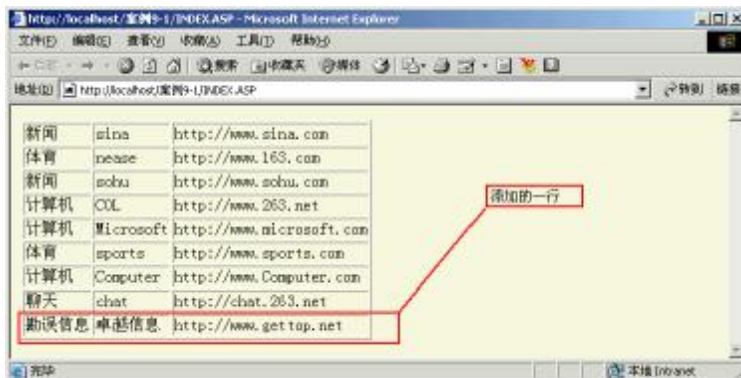


图 9-14 查看 XML 文件的内容

小结

本章需要重点理解的是 XML 的概念, XML 的优越性主要表现在可以定义标记, 配合 CSS 和 XSL 技术可以自己定义一套标记语言。掌握 XML 文件的三种显示方法: 利用 CSS 样式表、利用 XSL 样式表和利用 XML 数据岛技术。重点掌握利用 DOM 组件对 XML 文件进行读取和追加。

课后习题和上机练习

1. 简述 XML 和 HTML 的区别与联系。
2. XML 文档由哪些部分组成? 各有什么作用? 如何在 XML 文件中输出大于号?
3. XML 文件如何调用 CSS 样式文件? 如何调用 XSL 样式文件? 如何使用 XML 数据岛技术?
4. 利用 DOM 技术, 编写 ASP 程序对 9-08.xml 文件实现读取和追加操作。(上机完成)

第四部分 工程实践

第 10 章 项目分析：在线考试系统

本章要点

本章介绍一个在线考试系统的设计与开发。利用这个案例进一步巩固本书前面各章节的内容。分别介绍考试系统的数据库结构、考试系统的登录界面、考试系统的考试界面和考试的评分程序。

10.1 在线考试系统的数据结构

考试系统的主要功能是实现在线考试和客观题的自动评分。

首先介绍考试系统数据库的设计。该考试系统包含一个用户的成绩表和考试题目表，每个考试科目有一个表，保存考题信息。本考试系统实现两门科目的考试，加上成绩表，总共 3 个表。表结构如表 10-1 和表 10-2 所示。

表 10-1 数据库表“成绩单”的结构

字段名	类型	说明
学号	Varchar(20)	学员的学号
姓名	Varchar(20)	学员的姓名，学号和姓名用来验证用户的合法性
ASP	Int	ASP 科目的成绩，如果没有参加考试该字段为-1
XML	Int	XML 科目的成绩，如果没有参加考试该字段为-1

表 10-1 数据库表“ASP”的结构

字段名	类型	说明
类型	Varchar(20)	题目类型，单选或者多选
题号	Varchar(20)	题号
题目	Varchar(1000)	考题的题干
选项 1	Varchar(500)	选项一
选项 2	Varchar(500)	选项二
选项 3	Varchar(500)	选项三
选项 4	Varchar(500)	选项四
分数	Varchar(20)	改题目的分值
解答	Varchar(20)	该题目的解答

其他科目的数据库表和表 10-2 的结构一样。考试系统由四个文件组成，功能如下。

- (1) TEST.MDB: 考试系统的数据库文件（Access2000/XP 格式）。
- (2) Login.asp: 考试系统的登录界面。
- (3) Test.asp: 考试系统的考试界面。
- (4) Score.asp: 考试系统的评分界面。

10.2 考试系统的实现

首先进入的登录程序，其界面如图 10-1 所示。



图 10-1 考试系统的登录界面

登录程序主要用于密码的验证，检查登录的用户是否在考试系统的用户表中，如果在而且还没有参加过本科目的考试则可以进入考试。如程序 login.asp 所示。

案例名称：在线考试系统登录界面

程序名称：login.asp

```
<%@ Language=Jscript %>
<%
    var No = "";
    var Name = "";
    var Lesson = "";
    var Msg = "";
    if (Request("Name").Count > 0)
    {
        No = Request("No")(1);
        Name = Request("Name")(1);
        Lesson = Request("Lesson")(1);
        SQL = "Select * From 成绩单 ";
        SQL = SQL + "Where 学号=" + "'" + No + "'" + " And 姓名='" + Name + "'";
        var conn = Server.CreateObject("ADODB.Connection");
        conn.Open ("driver={Microsoft Access Driver (*.mdb)};dbq=" + Server.
            MapPath("test.mdb"));
        var rsScore = conn.Execute(SQL);

        if (rsScore.EOF)
        {
```

```
        Msg = "你不是合法考生!";
    }
    else
    {

        if (rsScore(Lesson) != -1)
        {
            Msg = "你已经考过本科目了!";
        }
        else
        {
            rsScore.Close();
            conn.Close();
            Response.Redirect("Test.asp?" + Request.QueryString);
        }
    }
    rsScore.Close();
    conn.Close();
}

%>
<HTML>
    <HEAD>
        <TITLE></TITLE>
    </HEAD>
    <BODY BGCOLOR="BEIGE">
        <H1 ALIGN="CENTER">在线考试系统</H1>
        <HR>
        <FORM ACTION="login.asp" METHOD="GET">
            <P>考试科目: <SELECT NAME="Lesson" SIZE="1">
                <OPTION VALUE="ASP">ASP</OPTION>
                <OPTION VALUE="XML">XML</OPTION>
            </SELECT></P>
            <P>姓名:
            <INPUT TYPE="TEXT" SIZE="20" NAME="Name" VALUE="<%=Name%>"></P>
            <P>学号:
            <INPUT TYPE="TEXT" SIZE="20" NAME="No" VALUE="<%=No%>"></P>
            <P><INPUT TYPE="SUBMIT" NAME="SEND" VALUE=" 开始考试"> </P>
        </FORM>
        <HR>
        <FONT Color="red"><%=Msg%></FONT>
```


</BODY>

</HTML>

首先利用 SQL 语句在数据库中查询，验证是否有此考生的信息。如果在数据库中找不到，说明该用户没有注册。找不到当前用户时，rsScore.EOF 为真。如果在数据库中找到了这个人的信息，但是所选考试科目的分数不为-1，此时说明考生已经考完了。

如果考生还没有参加考试，转到 test.asp 文件，利用 Request.QueryString 将浏览器地址栏的参数全部读取并传递到 test.asp，显示如图 10-2 所示。

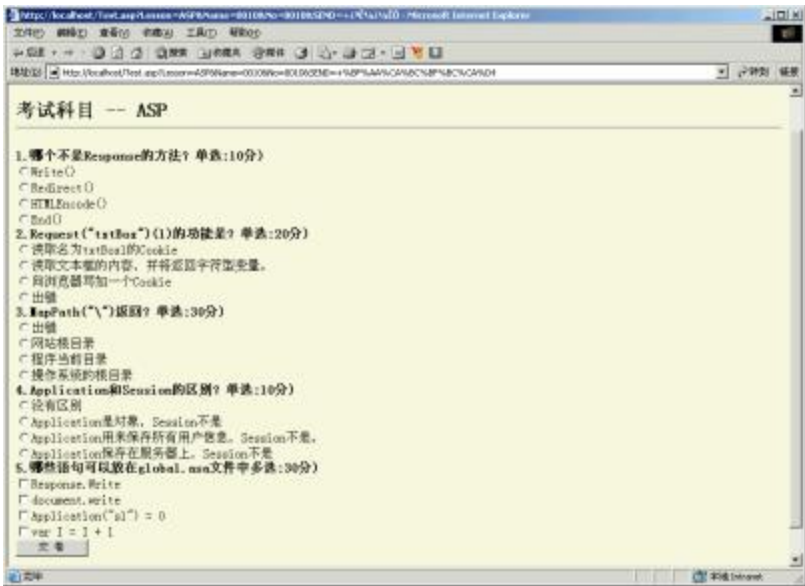


图 10-2 考试主界面

考题有单选题和多选题，所有试题都是从该科目的题库中读取的，程序如 test.asp 所示。

案例名称：在线考试系统考试界面

程序名称：test.asp

```
<%@ Language=Jscript %>
<%
    Lesson = Request("Lesson");
    No = Request("No");
    Name = Request("Name");
    SQL = "Select * From " + Lesson + " Order By 题号";
    conn = Server.CreateObject("ADODB.Connection");
    conn.Open ("driver={Microsoft Access Driver (*.mdb)};dbq=" + Server.
    MapPath("test.mdb"));
    rs = conn.Execute( SQL );
%>

<HTML>
```

```

<BODY BGCOLOR=BEIGE>
<CENTER>
<H1><FONT COLOR=#6699DD>在线考试系统</FONT></H1>
</CENTER>
<H2>考试科目 -- <%=Lesson%><HR></H2>
<FORM Action="Score.asp" Method="GET">
<INPUT Type="Hidden" Name="Lesson" Value=<%=Lesson%>>
<INPUT Type="Hidden" Name="No" Value=<%=No%>>
<INPUT Type="Hidden" Name="Name" Value=<%=Name%>>
<%
while (!rs.EOF)
{
    Response.Write("<B>" + rs("题号") + "." + Server.HtmlEncode(rs("题目")) +
rs("类型") + ":" + rs("分数") + "分)</B>")
    Response.Write("<div>");
    for (i=1; i<5; i++)
    {
        if (rs("类型") == "单选")
        {
            TestType = "Radio";
        }
        else
        {
            TestType = "CheckBox";
        }
        Response.Write("<INPUT Type=" + TestType + " Name=" + rs("题号") +
" Value=" + i + ">" + Server.HtmlEncode(rs("选项" + i)) + "<BR>");
    }
    Response.Write("</div>");
    rs.MoveNext();
}
%>
<INPUT Type=Submit Value=" 交 卷 ">
</FORM>
<HR>
</BODY>
</HTML>

```

程序首先将上一页提交过来的考生信息和考试科目读出来放到一个变量中，然后执行 SQL 语句从科目的数据表中读出考题的内容。

后面的页面还需要考生的信息，所以这里利用语句 “<INPUT Type="Hidden"

Value="Lesson" Value=<%=Lesson%>>" 将考生的信息保存起来，当考生提交表单时，信息同时被提交。

下面利用循环将题目读取出来，并显示成考题的形式，首先从数据库科目表中读出题号、题目、类型和该题的分值，然后利用一个循环，将四个选项显示出来，判断是多选题还是单选题。因为单选多选输出在显示时，type 属性不一样，如果是单选输出 Radio，如果是多选输出 CheckBox。

当用户提交时，系统就会调用 Score.asp 文件来对考生进行自动判分，并将考生的考试信息显示出来，如图 10-3 所示。

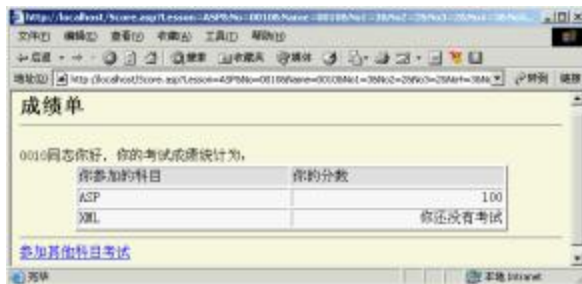


图 10-3 考生的得分显示

如果考生按“返回”按钮再重新答卷的话，系统会显示考过的信息如图 10-4 所示。



图 10-4 出错的信息

利用程序 score.asp 实现自动评分。

案例名称：在线考试系统成绩单界面

程序名称：score.asp

```
<%@ Language=Jscript %>
<%
    Lesson = Request("Lesson")(1);
    No = Request("No")(1);
    Name = Request("Name")(1);
    var conn = Server.CreateObject("ADODB.Connection");
    conn.Open("driver={Microsoft Access Driver (*.mdb)};dbq=" + Server.
    MapPath("test.mdb"));
    var rs = conn.Execute("Select * from " + Lesson);
%>
```

```

<HTML>
  <BODY BGCOLOR="BEIGE">
    <H2>成绩单<HR></H2>
  <%
    Score = 0
    while (!rs.EOF)
    {
      if (Request("No" + rs("题号")).Count>0)
      {
        Sel = Request("No" + rs("题号")) + "";
        Ans = rs("解答");
        //Response.Write(Sel + Ans + "<br>");
        if (Ans == Sel)
        {
          Score = Score + parseInt(rs("分数"));
          //Response.Write("OK");
        }
      }
      rs.MoveNext();
    }
    SQL = "Select * From 成绩单 ";
    SQL = SQL + "Where 学号=" + "'" + No + "'" + " And 姓名='" + Name + "'";
    rsScore = conn.Execute(SQL);
    if (rsScore(Lesson)=="-1")
    {
      conn.Execute("update 成绩单 set " + Lesson + " = " + parseInt(Score)
+ " where 学号=" + "'" + No + "'" + " And 姓名='" + Name + "'");
      rsScore= conn.Execute(SQL);
    }
    else
    {
      Response.Write ("<script>{alert('你已经考过了')}</script>");
    }
  %>

  <%=Name%>同志你好，你的考试成绩统计为：<BR>
  <CENTER>
    <TABLE BORDER=2 BGCOLOR="#F1F1F1" WIDTH=80%>
      <TR BGCOLOR="#DDDDDD"><TD>你参加的科目</TD><TD>你的分数</TD></TR>
      <TR><TD>ASP</TD><TD>
        align=Right><%=TestResult(rsScore("ASP"))%></TD></TR>

```

```

        <TR><TD>XML</TD><TD
Align=Right><%=TestResult(rsScore("XML"))%></TD></TR>
    </TABLE></center>

    <HR>

    <A HREF="login.asp?No=<%=No%>+Name=<%=Name%>">参加其他科目考试</A>

</BODY>
</HTML>
<%
function TestResult(Score)
{
    if (-1 == Score)
    {
        return "你还没有考试";
    }
    else if(Score < 60)
    {
        return "<FONT Color=Red>" + Score + "</FONT>";
    }
    else
    {
        return Score;
    }
    conn.close();
}
%>

```

程序首先将变量 `Score` 设置为 0, 然后利用一个循环从数据库中分别读出每题的标准答案, 与考生提交的答案比较, 如果相同则将该题的分值从数据库中读出来, 并累加到 `Score` 变量。判完试卷后, `Score` 的值就是考生的考试分数。

程序做了一个判断, 只有当考生数据库中该科目的成绩是 -1 时, 才能将信息提交到数据库中, 如果不是 -1 说明考生已经考过该科目了, 显示“你已经考过”的对话框。本考试系统充在 100 人以下的考场环境中表现良好。

小结

本章重点理解考试系统的数据库表结构, 如何判断一个用户是否合法, 如何在几个页面之间传递参数。掌握考试系统的登录程序和考试系统的评分程序, 并能对考试系统进行扩充。

课后习题和上机练习

1. 在 SQL Server 中编写 SQL 脚本创建如表 10-1 和表 10-2 所示的数据库表。

2. 将 Access 版本的考试系统改编成 SQL Server 版本。(上机完成)

3. 扩充考试系统功能。

(1)【课程设计】需求一：为考试添加用户管理功能模块，管理员可以通过管理界面对用户进行添加删除和修改。

(2)【课程设计】需求二：为考试系统添加题库管理功能，管理员可以通过管理界面对题库进行添加删除和修改。

参 考 文 献

- 1 石志国. ASP 动态网站编程. 北京: 清华大学出版社, 2001
- 2 栗松涛. XML 程序设计. 北京: 清华大学出版社, 2001
- 3 石志国. ASP.NET 实用案例教程. 北京: 清华大学出版社, 2003
- 4 石志国. 网页编程基础. 北京: 清华大学出版社, 2001
- 5 王国荣. ASP 网页制作教程. 北京: 人民邮电出版社, 2000
- 6 黄斯伟, 王伟. HTML 4.0 使用详解. 北京: 人民邮电出版社, 2001
- 7 闫华文. SQL Server2000 与 ASP Web 数据库编程技术. 北京: 北京大学出版社, 2001
- 8 贾佳, 郝洪明. ASP 与 SQL Server 网站架设. 北京: 机械工业出版社, 2001
- 9 Buser D, Kauffman J. ASP 3 初级编程. 北京: 机械工业出版社, 2001
- 10 Esposito D. ASP 数据访问高级编程. 北京: 机械工业出版社, 2001
- 11 Anderson R, Blexrud C. ASP 3 高级编程. 北京: 机械工业出版社, 2000
- 12 肖金秀. ASP 网络编程技术. 北京: 清华大学出版社, 2001