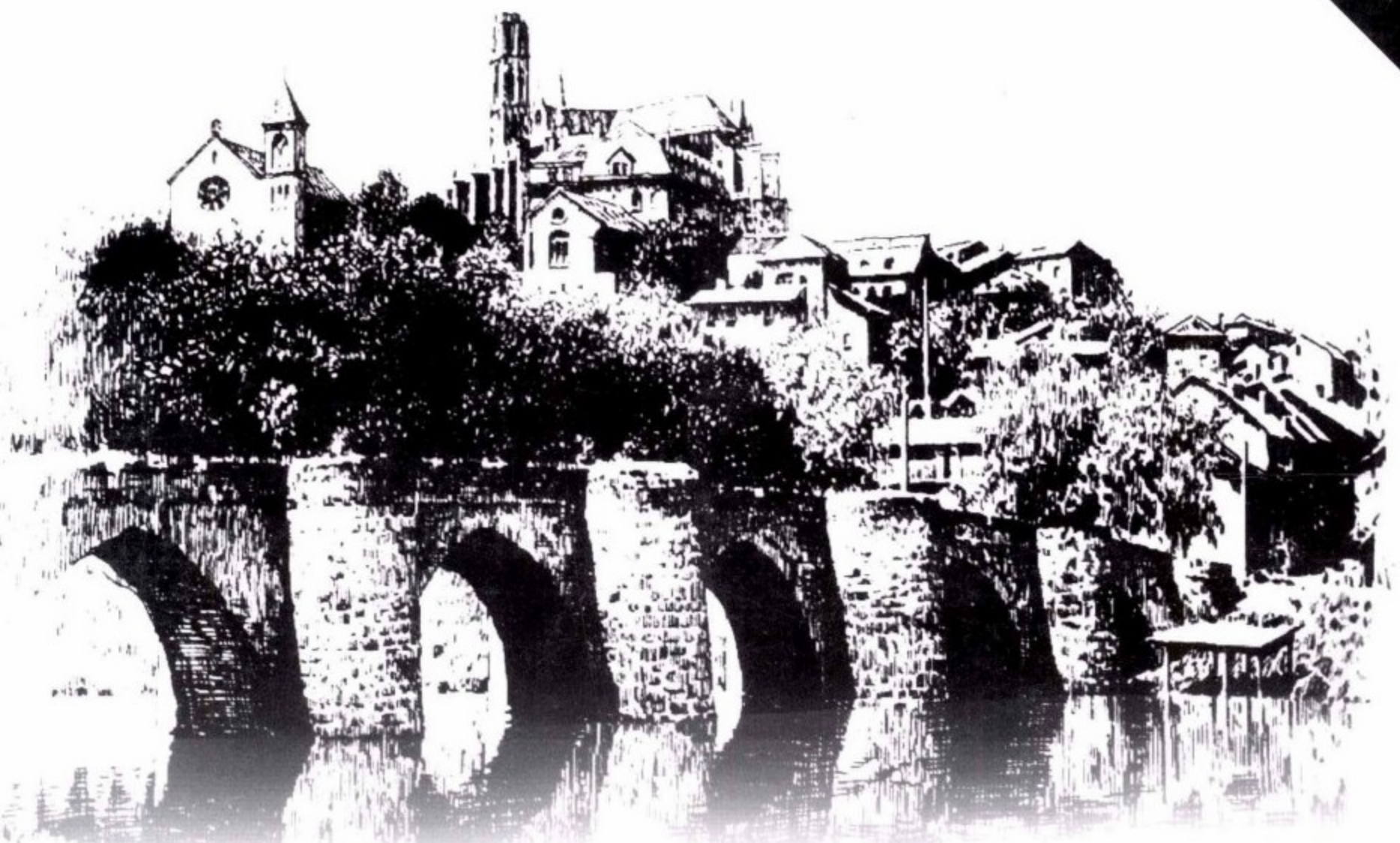
Broadview®
www.broadview.com.cn

博学 · 慎思 · 明辨 · 笃行

编程成功之路
从此开始！



ASP.NET 学习手册

DVD多媒体教学光盘：

- 19小时视频专题讲座
- 书中145个典型实例和5个实战项目
- 赠送ASP.NET项目源码
- 赠送多套学习示例关键代码

本书特色：

- 基础、易学、快速入门
- 内容讲解简单化、形象化、生活化
- 应用性与实践性相结合
- 精彩、详细的专家视频讲座

全程技术指导：

- 答疑网站：www.mingribook.com
- 学习社区：www.mrbccd.com

明日科技 吕双 房大伟 刘云峰 等编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

ÓÉ Foxit PDF Editor ±à¼-
°æÈ¨ËùÓÐ (c) by Foxit Software Company, 2004
½öÓÃÓÚÆÀ¹À¡£

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ⇕

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

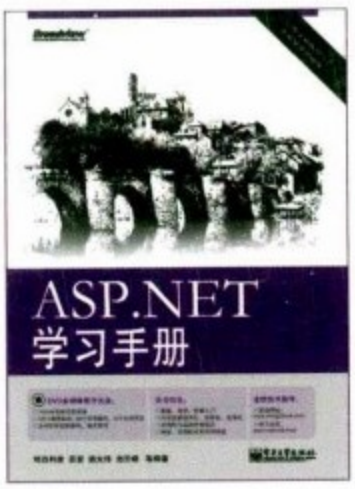
- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

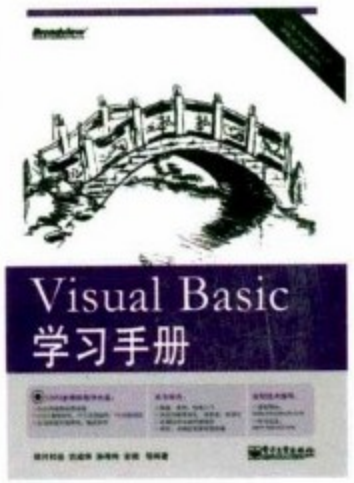
Total: 77 1 2 3 4 5 6 ›

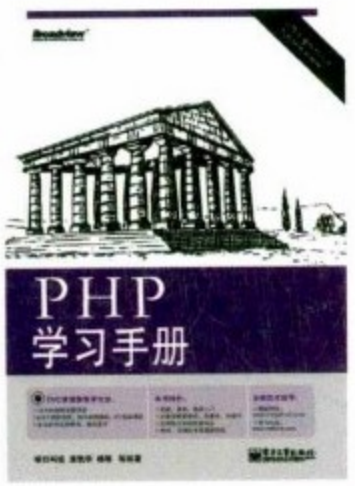
HTTP://WLSAM168.400GB.COM

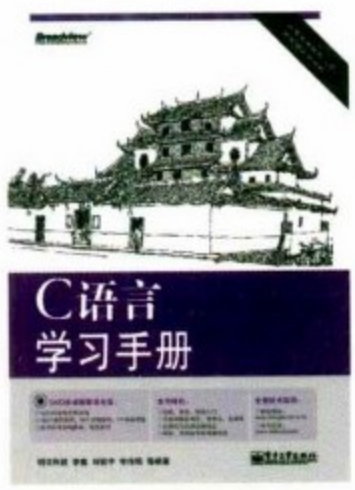
Broadview®
www.broadview.com.cn
博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版

系列全景









推荐阅读

上架建议：编程语言>ASP.NET

ISBN 978-7-121-12862-2

9 787121 128622 >

定价：59.00元（含DVD光盘1张）

责任编辑：胡辛征　江　立
封面设计：李　玲

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

ASP.NET 学习手册

明日科技 吕双 房大伟 刘云峰 等编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书以初学者为核心，全面介绍了使用 ASP.NET 进行程序开发的各种技术。在内容排列上由浅入深，让读者循序渐进掌握编程技术；在内容讲解上结合丰富的图解和形象的比喻，帮助读者理解“晦涩难懂”的技术；在内容形式上附有大量的提示、技巧、说明、情景应用、实战练习等栏目，夯实读者编程技术，丰富编程经验。全书共分 22 章，其中，主要讲述了 ASP.NET 的开发基础、开发环境、字符与字符串、运算符、数组和集合、内置对象、ADO.NET 技术、标准服务器控件、数据绑定控件、Web 用户控件、数据验证控件、GDI+图形图像技术、AJAX 无刷新技术、调试与错误处理等知识，并详细介绍了面向对象编程的相关内容。最后，通过几个实战项目使读者将所学知识更好地应用到实际开发当中。

本书附有配套光盘。光盘中包含本书所有程序的源代码、重点难点的实例视频录像。其中，源代码全部经过精心测试，能够在 Windows 2000、Windows XP、Windows 2003、Windows 7 系统中编译和运行。

本书适用于 ASP.NET 的爱好者、初学者和中级开发人员，也可以作为大中专院校和培训机构的教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

ASP.NET 学习手册 / 吕双等编著. —北京：电子工业出版社，2011.3
ISBN 978-7-121-12862-2

Ⅰ. ①A… Ⅱ. ①吕… Ⅲ. ①主页制作－程序设计－技术手册 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字（2011）第 013874 号

责任编辑：胡辛征　江　立
印　　刷：
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：32　字数：1113 千字
印　　次：2011 年 3 月第 1 次印刷
印　　数：4000 册　定价：59.00 元（含 DVD 光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

让想要学习编程的人员都能够学会编程！

我可以学会编程吗？

当然可以！即使你没有基础，即使你非计算机专业毕业，即使你已过而立之年，甚至是只有初中毕业。我国最早的计算机反病毒专家、江民杀毒软件创始人王江民，初中毕业，38 岁开始学计算机，因为英语基础不好，很多人认为他根本没有学会编程的可能性。但王江民没有感觉自己不行，硬是克服各种困难，成为了我国最成功的程序开发人员之一。

学习编程的诀窍是什么？

是实践。王江民说过：“计算机是实践性非常强的学科。我搞计算机是用计算机，不是学计算机。”王江民首先学的是 Basic 语言。当时，为了辅导上小学的孩子，对软件一片空白的王江民竟然决定编写程序代替家长辅导。短短 3 个月，王江民就边学边实践，编好了一套数学语文教学软件，试过后效果奇佳。后来，该软件在电脑报软件交流会中被评为第二名，誉为“教育软件第一”。因为这次成功，极大地激发了王江民对编程的兴趣，使他从此把精力转到了软件领域。

所以说：实践是学习编程的最好方法，也是培养编程乐趣、捕获发展机遇的最佳途径。值得注意的是，不要等学完全部知识再去实践，要随时学习随时实践，哪怕只学了一点点。

这本书适合我吗？

非常适合！本书是学习 ASP.NET 编程的绝佳选择。所讲内容通俗易懂，易于学习，贴近实践。对于学习中的难点、重点，注意结合实际开发，采用情景应用的方式进行介绍和练习。本书光盘提供了完整的视频讲座，可以对照本书内容循序渐进地进行学习。利用本书，不但可以学会编程，更能激发学习者理论联系实际的热情，开发出符合市场需求的软件和项目。

「本书内容」

本书分为基础篇、核心篇、高级篇和实战篇，共计 22 章内容。内容安排按照“从零起步，循序渐进”的学习顺序，全面提高读者学、练、用能力。本书知识结构如图 1 所示。

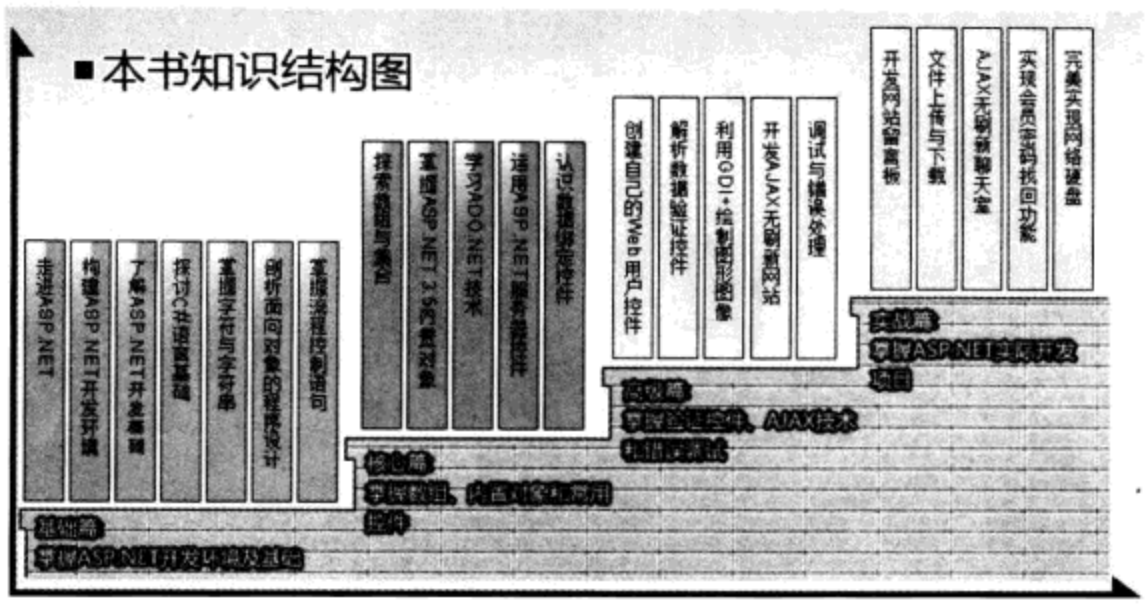


图 1　本书知识结构图

「本书特色」

☑ 最基础、最易学

为了让初学者易于学习，本书力求内容通俗易懂，讲解寓教于乐。对于初学者难以理解的专业术语，本书都进行了形象的解释，有些还提供了例图。书后附录中又单独将专业术语提炼出来，供读者学习时参考。例如：数据库好比水源；Connection 好比伸入水中的进水笼头，保持与水的接触，只有它与水进行了“连接”，其他对象才可以抽到水；Command 则像抽水机，为抽水提供动力和执行方法；DataAdapter、DataReader 就像输水管，担任着水的传输任务；DataSet 则是一个大水库，把抽上来的水按一定关系进行存放，即使撤掉“抽水装置”（断开连接，离线状态），也可以保持“水”的存在，如图 2 所示。这也正是 ADO.NET 的核心。

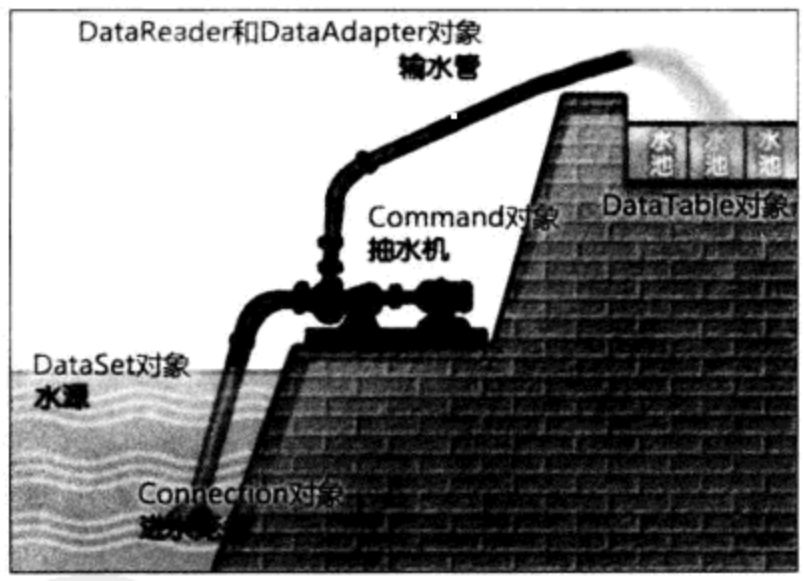


图 2　ADO 工作原理演示图

☑ 语音视频讲座

为了方便读者学习本书内容，本书附赠光盘中提供了 19 小时视频讲座。其讲解细腻、层次清楚、互动性强，不但可以加深对书中内容的理解，还可以引导学习者掌握“是什么”、“为什么”、“怎么办”、“应注意些什么”的编程思维能力。所讲视频不仅对书中的内

容进行了详细介绍，还对相关技术进行了有效扩展，对开发中易犯的各种错误做出了视频解决方案。

☑ 注重实用性

初学者经常会遇到这样的情况，书中讲解的技术能够理解，但不知道如何应用。例如，在学习抽象类时，读者能够自己定义抽象类并且能够从抽象类派生子类，但是不知道抽象类能够做什么、在哪里应用抽象类。本书在介绍技术时，注意强调技术的实用性，并且通过实例突出技术的应用价值。对于一些典型的应用，书中通过“情景应用”栏目进行重点介绍。

☑ 实战互动练习

要掌握一项技术，最佳的方式就是多练习、多实践。本书每章都提供了多个供读者实践的实例任务，读者可以对照检验对知识的掌握情况。每个实例任务都分为“题目描述”、“技术指导”和“紧急救援”3 部分。其中“题目描述”给出了实例功能的详细描述和效果图；“技术指导”给出了开发思路和关键技术；“紧急救援”提供了下载实例源码和开发文档的地址链接。

☑ 思想与激励

本书每章都提供了一个与软件行业相关的励志故事，主要有两个目的：一是读者在学习疲惫的时候，可以阅读这些人物故事，放松一下；二是希望读者能从这些人物事迹中有所感悟。这些人物都具有一些共同的特征，他们都能将学习与实践完美结合，都能发现生活中的一些小机会，借以发扬光大，都具有专注、坚持不懈的精神……。希望这些故事能成为你编程路上的精神食粮，伴你度过编程学习的日日夜夜。

「超值 DVD 光盘」

为了帮助读者学习和使用书中的实例，本书附赠 DVD 光盘 1 张，里面不仅包含书中所有实例项目的源代码、素材、光盘使用说明书，还提供了 19 小时视频专题录像以及 5 个项目源码。光盘目录如图 3 所示。

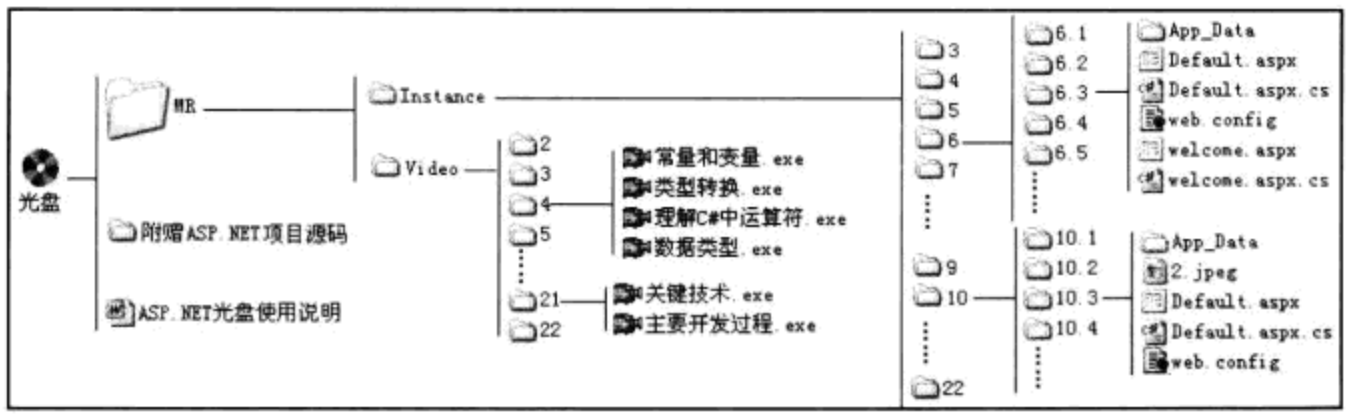


图 3　本书配套光盘目录

「本书适用人群」

本书是一部零基础编程图书，非常适合以下人员阅读：

☑ 零基础的初学编程人员

☑ 初中级程序开发人员

☑ 编程爱好者
☑ 大中专院校的教师和学生
☑ 想学编程的各企事业在职人员
☑ 相关培训机构的教师和学员

『在线互动答疑』

在学习本书过程中，如有任何疑问可以通过如下方式与我们联系：

☑ 服务网站：www.mingribook.com
☑ 学习社区：www.mrbccd.com
☑ 服务信箱：mingrisoft@mingrisoft.com
☑ 服务电话：0431-84978981/84978982

我们承诺将在5个工作日内给您提供满意的回答。

『本书作者』

本书由明日科技组织编写，参加编写的有吕双、房大伟、刘云峰、杨丽、顾丽丽、刘玲玲、董大永、李继业、尹强、张磊、王小科、王军、安剑、孙秀梅、赛奎春等。由于作者水平有限，疏漏和不足之处在所难免，欢迎广大读者提出宝贵意见。

非学无以广才，非志无以成学。

祝大家读书快乐！

明日科技

2011年1月

目　录

第 1 篇　基础篇

第 2 篇 核心篇

第 3 篇　高级篇

第 4 篇　实战篇

PDG

第 1 篇

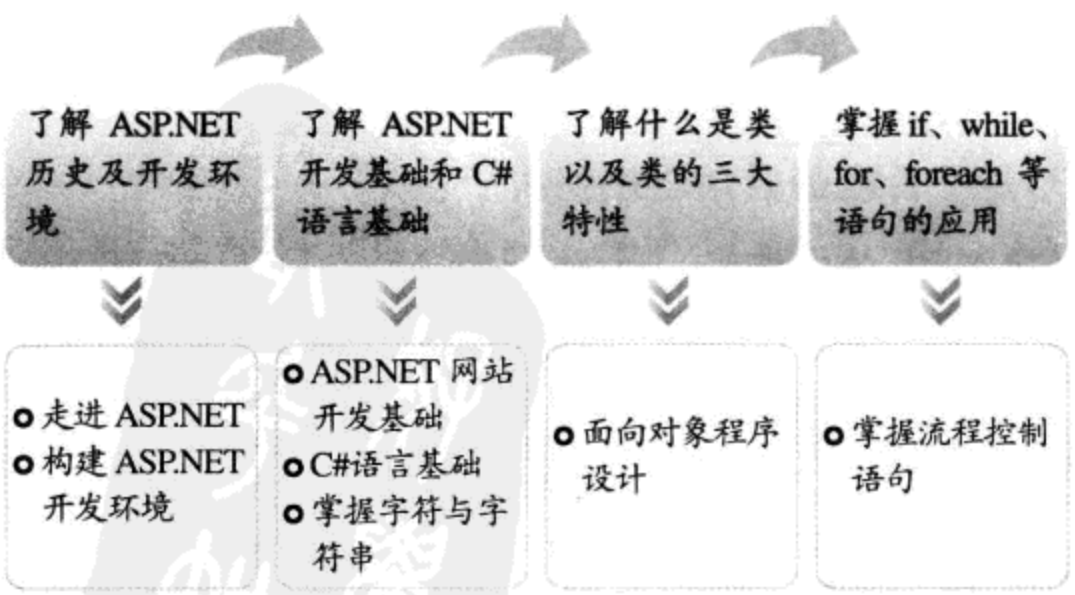
基础篇

书山有路勤为径
学海无涯苦作舟

本篇主要内容：

- 第 1 章　走进 ASP.NET
- 第 2 章　构建 ASP.NET 开发环境
- 第 3 章　ASP.NET 网站开发基础
- 第 4 章　C#语言基础
- 第 5 章　掌握字符与字符串
- 第 6 章　面向对象程序设计
- 第 7 章　掌握流程控制语句

本篇学习流程：



第 1 章

走进 ASP.NET

每个人都是从零开始成长的。程序员当然也不例外，从开始接触编程，到开始学习编程，之后以一个初级程序员的身份参加工作，一两年之后成为一名高级程序员，再之后就是做一名出色的系统分析师，来策划整个项目并分配任务。从开始学习编程到成为一名系统分析师，大概就需要花费 5 年的时间。当然这需要程序员在工作中，不断地学习、研究。系统分析师的工作是令人羡慕的，也是很多公司的挖掘对象。这也就是为什么说 IT 业是一个很有发展空间的职业，也许在不久的将来，你就是一个优秀的系统分析师。为了这个目标，从本章开始努力学习吧。通过本章的学习，你可以掌握以下知识：

- 了解 ASP.NET 到底是什么
- 明白自己选择 ASP.NET 的理由
- 欣赏由 ASP.NET 开发的成功案例图片
- 掌握学好 ASP.NET 的方法

1.1　接触 ASP.NET

从本节开始，读者将会踏上奇妙的 ASP.NET 之旅。本节能够使读者了解什么是 ASP.NET、什么是.NET Framework 框架、C#语言的诞生以及这三者之间的关系。

1.1.1　ASP.NET 是什么

ASP.NET 是一种开发动态网站的技术。它是.NET 框架的一部分，可以使用任何.NET 兼容的语言（如 Visual Basic .NET、C#、J#等语言）来编写 ASP.NET 网站。ASP.NET 是作为.NET 框架体系结构的一部分推出的，其更新顺序如下：

☑ 2000 年 ASP.NET 1.0 正式发布。

☑ 2003 年 ASP.NET 升级为 1.1 版本。

☑ 2005 年 11 月微软公司发布了 ASP.NET 2.0。

☑ 2008 年 2 月微软公司发布了 ASP.NET 3.5。

ASP.NET 3.5 开发网站时，用“简化”来形容一点不为过。因为其设计目标是将应用程序代码数减少 70%，改变过去那种需要编写很多重复性代码的状况，尽可能做到写很少的代码就能完成任务的效果。对于应用构架师和开发人员而言，ASP.NET 3.5 是 Microsoft Web 开发史上的一个重要的里程碑！

『名词解析』动态网站和静态网站

在此读者可能会有疑惑，什么是动态网站？什么是静态网站？所谓的动态网站就是网站中网页网址的后缀名不是.htm、.html、.shtml、.xml 等静态网页的常见形式，而是以 aspx、asp、.jsp、.php 等形式为后缀，动态网页可以与数据库交互，并且在动态网页网址中可以使用“?”传递参数；静态网页是相对于动态网页而言的，是指没有后台数据库、不含程序和不可交互的网页，编的是什么它显示的就是什么、不会有任何改变。静态网页相对更新起来比较麻烦，适用于一般更新较少的展示型网站。

1.1.2　.NET Framework

.NET Framework 是微软公司推出的完全面向对象的软件开发与运行平台。.NET Framework 具有两个主要组件：公共语言运行时（Common Language Runtime，简称 CLR）和类库。

☑ 公共语言运行时：公共语言运行时（CLR）负责管理和执行由.NET 编译器编译产生的中间语言代码（.NET 程序

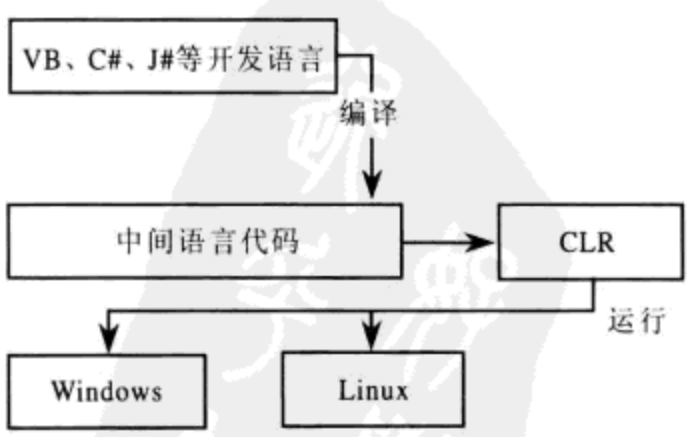


图 1.1　.NET 程序执行原理

执行原理如图 1.1 所示）。由于公共语言运行库的存在，解决了很多传统编译语言的一些致命缺点，如垃圾内存回收、安全性检查等。

☑ 类库：类库我们比较好理解，就好比一个大仓库里装满了工具。类库里有很多现成的类，可以拿来直接使用。例如，文件操作时，可以直接使用类库里的 IO 类。

『名词解析』中间语言代码

所谓的中间语言代码是一种类似于汇编的程序语言，起到中介或翻译的作用。例如，有汉语、日语和法语三位学者，他们正在交流一个问题，他们怎么交流呢？最后，他们选择用英语进行交流，因为英语是世界上普及较广的语言。这样，他们彼此才能明白对方说的是什么。此处的英语就相当于中间语言，任何语言都会被编译成中间语言。

1.1.3 C#语言

Visual C#（以下简称 C#）是微软公司推出的一种语法简洁、类型安全的面向对象的编程语言，开发人员可以通过它编写在.NET Framework 上运行的各种安全可靠的应用程序，使用 C#可以创建很多程序，例如，中国移动飞信系统、安华农业保险系统等。既然 C#功能这么强大，并且很多程序员都使用 C#开发 ASP.NET，那么 C#的开发者是谁？C#是什么时期诞生的？我们带着这些疑问来学习下面的内容。

1998 年，安德斯•海尔斯伯格（Delphi 和 Turbo Pascal 语言的设计者）以及他的微软开发团队开始设计 C#语言的第一个版本。2000 年 9 月，ECMA（国际信息和通信系统司标准化组织）成立了一个任务组，着力为 C#编程语言定义一个 Microsoft 建议标准。据称，其设计目标是制定“一个简单、现代、通用、面向对象的编程语言”，于是出台了 ECMA-334 标准，这是一种令人满意的简洁的语言，它有类似 Java 的语法，但显然又借鉴了 C++和 C 的风格。设计 C#语言是为了增强软件的健壮性，为此提供了数组越界检查和“强类型”检查，并且禁止使用未初始化的变量。C#语言自从 2002 年随着 Visual Studio 一起推出，其发展趋势如图 1.2 所示。

学习手册 说明 从图 1.2 中可以看出，C#自从 2002 年正式发布以来，一直呈现稳定的上升趋势，而且作为微软全力推广的一种新语言，伴随着 Windows 7 操作系统（内嵌.NET Framework 3.5）的发布与普及，它的发展前景被世界的编程人员所瞩目，前途不可限量。

1.1.4 三者之间的关系

通过以上几节的学习，读者对 ASP.NET、.NET Framework 和 C#已经有所了解。那么，这三者之间是否存在某种关系呢？答案是肯定的。通过 C#或其他语言可以开发出 ASP.NET 应用程序，但是 ASP.NET 应用程序还必须运行在.NET Framework 平台上，因为 C#的命名空间和类库都是在.NET Framework 中，对于这三者之间的关系，图 1.3 可以很好

地进行诠释。

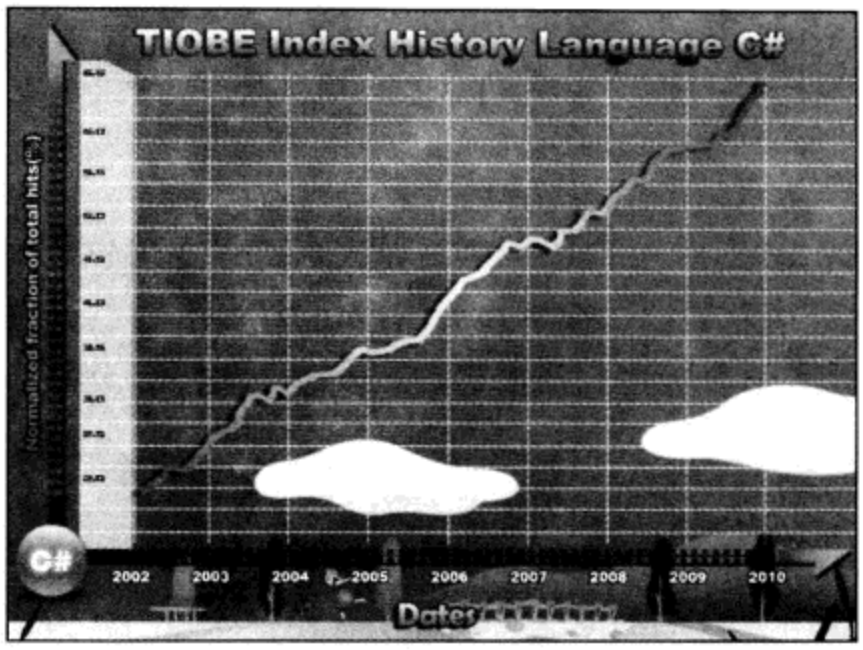


图 1.2　C#语言发展趋势

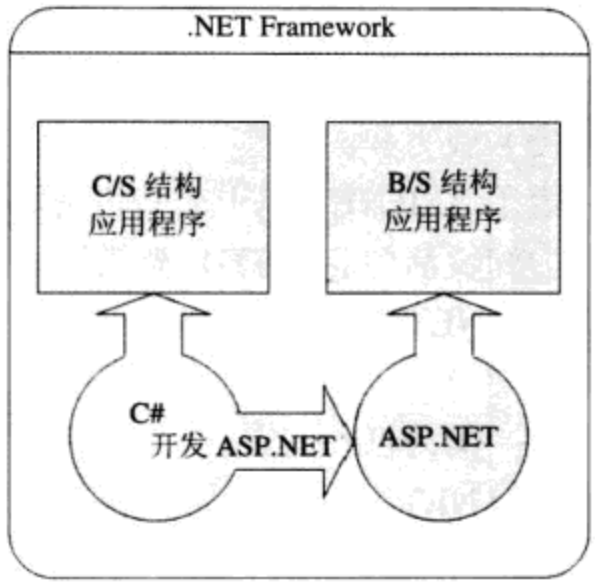


图 1.3　ASP.NET、.NET Framework 和 C# 三者之间的关系

1.2　选择 ASP.NET 的理由

编程语言有很多，为什么选择 ASP.NET 呢？之所以 ASP.NET 受到众多编程爱好者的喜爱，一定有其自己的特性和优点，通过本节，读者可以对 ASP.NET 的特性和优点有所了解，揭开 ASP.NET 受宠之谜。

1.2.1　ASP.NET 特性

与其他语言相比，ASP.NET 开发网站的速度是非常惊人的，维护起来也相当方便，且使用代码少。同时，还可以根据自己的需求向 ASP.NET 添加自定义功能。ASP.NET 特性主要包括以下几方面：

- ☑ 开发效率：使用新增的 ASP.NET 服务器控件和包含新增功能的现有控件，可以轻松、快捷地创建 ASP.NET 网站。
- ☑ 灵活性和可扩展性：很多 ASP.NET 功能都可以扩展，这样可以轻松地将自定义功能集成到程序中。例如，ASP.NET 为不同数据源提供插入支持。
- ☑ 性能：使用缓存和 SQL 缓存失效等功能，可以优化网站的性能。
- ☑ 安全性：向网站程序中添加身份验证和授权比以往任何时候都简单。
- ☑ 利用 ASP.NET AJAX 可以创建更有效、更具交互性、高度个性化的 Web 体验，这些体验在所有最流行的浏览器上都能实现。
- ☑ Visual Studio 2008 对 WF、WCF 和 WPF 的完美支持。

1.2.2　ASP.NET 的优点

ASP.NET 与以前的 Web 开发相比，增加了许多功能，其优点也很明显。所以，下面

介绍 ASP.NET 的几个重要的优点。

（1）增强的性能

ASP.NET 是将编译好的公共语言运行库代码运行在服务器上。它可以利用早期绑定、实时编译、本机优化和缓存服务，相当于在编写代码前就提高了应用程序性能。

（2）威力和灵活性

由于 ASP.NET 基于公共语言运行库，因此 Web 应用程序开发人员可以使用整个平台的威力和灵活性。.NET 框架类库、消息处理和数据访问解决方案都可从 Web 进行无缝访问。ASP.NET 与语言无关，所以可以选择最适合应用程序的语言，或跨多种语言分割应用程序。

（3）简易性

ASP.NET 使执行常见任务变得容易，从简单的窗体提交和客户端身份验证到部署和站点配置。另外，公共语言运行库利用托管代码服务（如自动引用计数和垃圾回收）简化了程序的开发过程。

（4）可管理性

ASP.NET 采用基于文本的分层配置系统，简化了设置应用于服务器环境和 Web 应用程序的工作。

（5）自定义性和扩展性

ASP.NET 使开发人员可以用自己编写的自定义组件扩展或替换 ASP.NET 运行库的任何子组件。

（6）安全性

内置 Windows 身份验证和基于每个应用程序的配置保证了应用程序是安全的。

1.3 欣赏 ASP.NET 成功案例

ASP.NET 作为微软全力推出的一个崭新的网站开发技术，经过最近几年的发展，在实际生活中已经有了很多成功的项目案例，比如世界饮食行业的龙头老大 KFC、中国最成功的游戏之一问道、中华人民共和国人力资源和社会保障部以及中国最著名的汽车产商之一东风汽车公司等，它们的官方网站都是用 ASP.NET 开发的。

1.3.1 龙头老大 KFC

肯德基（KFC）的名字家喻户晓，它有着悠久的历史，已经成为世界饮食行业的龙头老大，其食品的卫生和口味也得到认可，更是以方便和快捷征服广大消费者。其官方网站就是由 ASP.NET 开发的，如图 1.4 所示。

1.3.2 问道官网

问道是历时两年时间开发而成的 2D 回合制网络游戏。它是一款以深厚博大的道教文化为切入点，以《道德经》、《庄子》中的人物为衍生，用幽默搞笑的工笔画风格再现了

道家风骨的回合制网游。其官方网站也是用 ASP.NET 开发的，如图 1.5 所示。

1.3.3　中华人民共和国人力资源和社会保障部

中华人民共和国人力资源和社会保障部是统筹机关企事业单位人员管理以及城乡就业和社会保障政策的中国国家权力机构。十一届全国人大一次会议第四次全体会议（2008.03.11）“国务院机构改革方案”审议通过组建，同时组建国家公务员局，由人力资源和社会保障部管理。不再保留人事部、劳动和社会保障部。2008 年 3 月 31 日正式挂牌，而其官方网站也于同日开始运行。其官方网站也是用 ASP.NET 开发的，如图 1.6 所示。

图 1.4　KFC 官方网站

图 1.5　问道游戏官方网站

1.3.4　东风汽车公司

东风汽车公司始建于 1969 年，是中国汽车行业的骨干企业之一。公司主要业务分布在十堰、襄樊、武汉、广州四大基地，形成了“立足湖北，辐射全国，面向世界”的事业

布局。公司总部设在“九省通衢”的武汉，主营业务涵盖全系列商用车、乘用车、发动机及汽车零部件和汽车水平事业。其官方网站同样使用 ASP.NET 开发，如图 1.7 所示。

图 1.6　中华人民共和国人力资源和社会保障部官方网站

图 1.7　东风汽车公司官方网站

1.4　学好 ASP.NET 的方法

学习编程对大多数 IT 人员来说都是非常重要的，学编程，做一名编程人员，从个人角度讲，可以解决在软件使用中遇到的问题，改进现有软件，还可以为自己找到一份理想的工作添加重要的筹码，有利于在求职道路上谋得一个好的职位，一名优秀的程序员永远是被争夺的对象。

笔者也是从一个初学者一路走来的，通过一路上的自学探究，深深体会到对于任何一门技术来说往往是两头难，也就是入门难、精通难。对于学习 Web 开发来说更是如此，Web 开发不是一门语言，而是多种技术、多个语言的集合。

通过本章的学习，读者对 ASP.NET 已经有了深刻的认识。那么，接下来就应该考虑如何才能学好 ASP.NET。在整个学习过程中，要学会使用一些工具来解答问题（例如，Microsoft Visual Studio 的 MSDN 帮助文档，在以后章节中会做详细介绍）。因为不是每一个初学者的周围都会有高手回答自己的问题，所以我们要学会自己解决问题，同时需要注意以下几个问题。

1.4.1　明确学习目的

在确定学习编程之前，先问问自己为什么要学习编程，有没有信心学好。这一点很重要，不要随波逐流，看大家都在学就去学，这样盲目的学习对自己没有好处。

1.4.2　打好基础，盖好“房子”

学习编程如同盖房子一样，只有良好的基础才能把大楼盖得更高、更好。程序员也是如此，想要有良好的发展，必须有扎实的基础。在学习编程之初，一定要注重逻辑思维能力的培养，要想成为一名优秀的程序员，最重要的是掌握编程思想。要做到这一点必须在反复的实践、观察、分析、比较、总结中逐渐地积累。因此在学习编程的过程中，我们不必等到什么都完全明白了再去动手实践，先尝试做出一些东西，然后再去探究原因，这种曲折式的学习方法更适合初学者。有些问题只有通过实践后才能明白，也只有实践才能把老师和书上的知识变成自己的，高手都是这样成才的。

1.4.3　多请教、多交流

在初学编程的过程中，一定会遇到很多的问题，当遇到问题时，一定要多和同学交流、多向老师请教。千万不要问代码应该怎么写，应该问解决问题的思路。

另外，对于初学者一定还要利用好大型搜索引擎网站（百度、谷歌等）和比较知名的社区论坛（如明日科技编程词典学习社区 bbs.mrbccd.com 等）。

一定要学会搜索，全世界这么多人在学习 ASP.NET，如果你遇到问题，这个问题 99% 别人也遇到过。现在网络这么发达，无数人在 Blog、论坛分享自己的解决方案，因此如果大家在学习过程中遇到难题，身边没有其他人可以帮助，通过搜索引擎在网络上寻找答案是一种较好的学习方式。

1.5　本 章 小 结

本章主要带领读者了解什么是 ASP.NET、.NET Framework 和 C#，并通过直观的示意图展示这三者之间的关系。学习本章之后，读者能够更好地了解 ASP.NET 的特性和优点，并且欣赏 ASP.NET 的成功案例。对于初学者来说，如何学好 ASP.NET 已经成为首要问题。作者在本章最后概括了自己从初学者到开发者的过程中总结出来的学习方法，希望对读者能够起到点拨的作用。

第2章

构建 ASP.NET 开发环境

（名师课堂：43 分钟）

本章的主要内容是介绍构建 ASP.NET 开发环境的基本条件，并以图文并茂的形式介绍如何安装 Visual Studio 2008 开发工具和 MSDN 帮助文档。通过本章的学习，读者会对 ASP.NET 开发工具有更深的认识，并掌握以下知识：

- 了解构建 ASP.NET 3.5 的软件和硬件要求
- 熟悉如何安装 IIS
- 掌握如何安装 Visual Studio 2008 开发工具
- 熟悉如何创建网站
- 掌握如何安装和使用 MSDN 帮助文档

2.1　构建 ASP.NET 3.5 开发环境

为了更好地学习 ASP.NET，首先要了解如何构建 ASP.NET 的开发环境，以及对软件和硬件的要求。如果想在某台计算机中架设 ASP.NET 网站，必须在计算机中安装 IIS 和.NET Framework。通过本节的学习，读者能了解 IIS 的详细安装过程，为以后的学习奠定基础。

2.1.1　软件和硬件要求

构建 ASP.NET 3.5 开发环境时，对软件和硬件都有一定的要求。如果你的配置低于最低要求，程序运行时可能会影响开发，具体要求如图 2.1 所示。

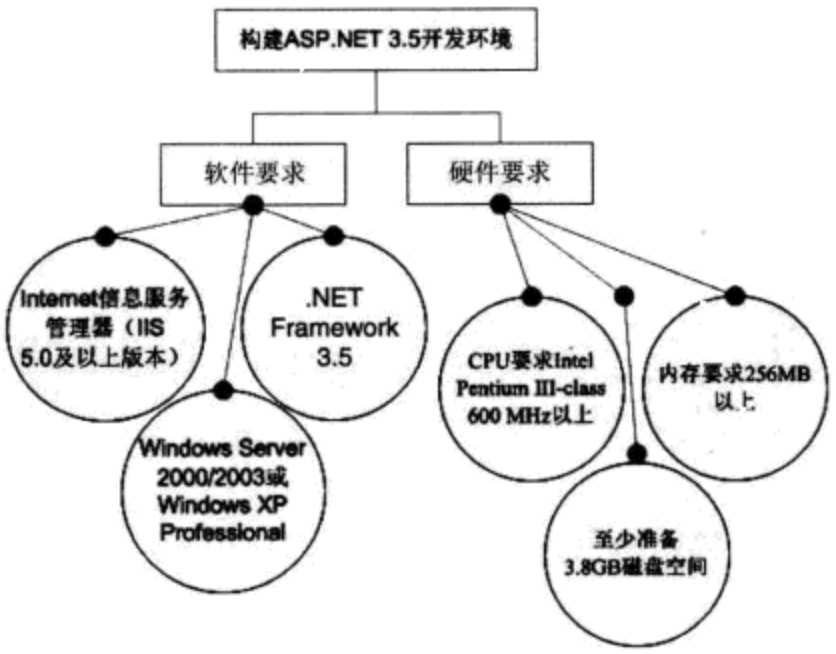


图 2.1　软件和硬件要求

（1）软件要求

☑ Internet 信息服务管理器（IIS 5.0 及以上版本）。

☑ .NET Framework 3.5。

☑ 操作系统方面，最低版本要求是 Windows Server 2003 SP1（其中 SP 为 Windows 操作系统的补丁，1 代表补丁的版本）、Windows XP Professional SP2 或 Windows Server 2000 SP4 系列。

（2）硬件要求

☑ CPU：CPU 要求 Intel Pentium III-class 600 MHz 以上。

☑ 内存：内存要求 256MB 以上。

☑ 显示器：显示器要求至少 800×600 像素，256 色（建议 1024×768 像素，增强色 16 位）。

如果是最近一年内配置的计算机，其硬件标准基本上都符合要求。

2.1.2 安装 IIS

专题讲座：光盘\MR\Video\2\安装 IIS 服务器.exe

➤➤➤视频速递：本视频详细地演示了如何安装 IIS

IIS 是 Internet Information Server 的缩写，是微软公司主推的 Web 服务器。IIS 提供了最简捷的方式来共享信息，建立并部署企业应用程序，以及建立和管理 Web 上的网站。通过 IIS，用户可以轻松地测试、发布、应用和管理自己的 Web 页和 Web 站点。

Windows 系统安装时默认没有安装 IIS，需要用户自行安装。安装 IIS 很简单，花费大约几分钟的时间就可以完成。下面介绍在 Windows 7 操作系统中 IIS 的安装，具体的安装步骤如下。

（1）将 Windows 7 系统光盘放到光盘驱动器中。

（2）依次选择“开始”/“控制面板”/“程序和功能”选项，打开“卸载或更改程序”对话框，如图 2.2 所示。

（3）单击对话框左侧的“打开或关闭 Windows 功能”按钮，弹出“Windows 功能”对话框，如图 2.3 所示。

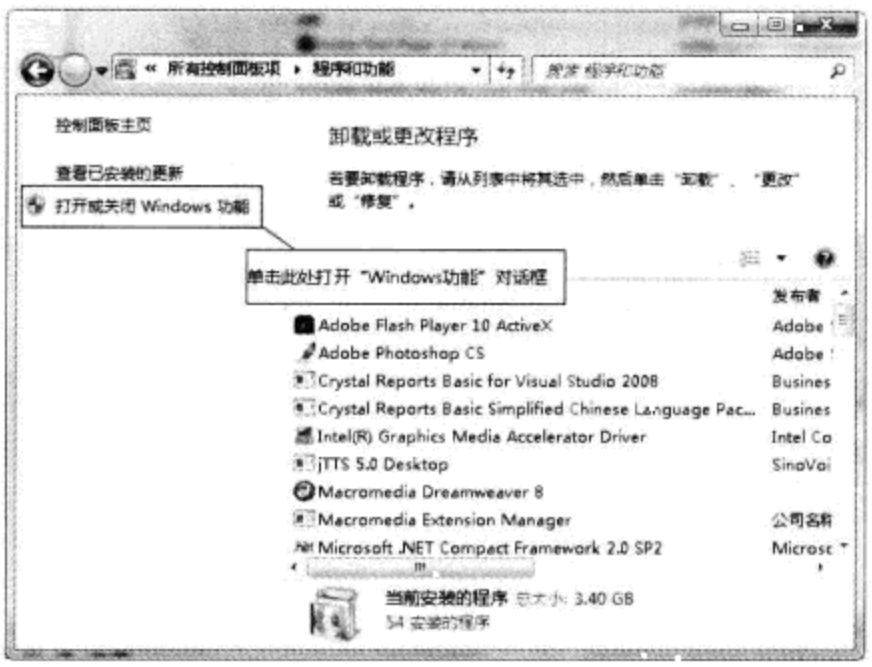


图 2.2 “卸载或更改程序”对话框

图 2.3 “Windows 功能”对话框

图 2.4 选中全部子节点

（4）单击“Internet 信息服务”下面子节点前的“+”号，展开子节点，并将子节点全部选中，如图 2.4 所示。

（5）单击“确定”按钮，开始安装 IIS，如图 2.5 所示。

（6）IIS 安装完成之后，选择“开始”/“控制面板”/“管理工具”/“服务”选项，弹出“服务”窗口，如图 2.6 所示。

（7）在“服务”窗口中，选择“IIS Admin Service”，双击鼠标左键，打开“IIS Admin Service 的属性”对话框，如图 2.7 所示。在该对话框中启动类型选择“自动”，然后单击“启动”按钮来启动服务。

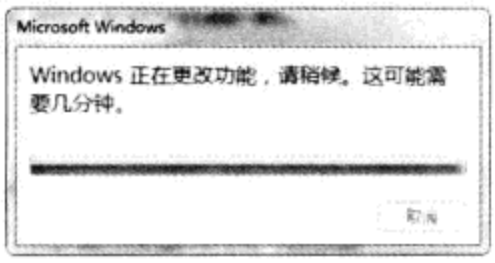


图 2.5　开始安装 IIS

图 2.6　“服务”窗口

（8）启动“IIS Admin Service”之后，打开 Internet 信息服务（IIS）管理器，看看其运行界面，如图 2.8 所示。

图 2.7　启动 IIS Admin Service

图 2.8　Internet 信息服务（IIS）管理器

2.2　安装 Visual Studio 2008 集成开发工具

Visual Studio 2008 是微软为了配合.NET 战略推出的开发工具，可以开发 ASP.NET 2.0、ASP.NET 3.0、ASP.NET 3.5 网站程序。当然，用来开发其他 Windows 应用程序也是游刃有余的，例如中国移动飞信系统。在开发 ASP.NET 应用程序时，需要安装 Visual Studio 2008 集成开发环境。本节将对 Visual Studio 2008 的安装与使用进行详细讲解。

2.2.1　安装 Visual Studio 2008

专题讲座：光盘\MR\Video\2\安装 Visual Studio 2008 集成开发工具.exe

➤➤➤视频速递：本视频将详细地演示如何安装 Visual Studio 2008 集成开发工具。

（1）将 Visual Studio 2008 安装盘放入计算机光驱，光盘运行后会自动弹出安装程序界面，如图 2.9 所示，该界面中共有 3 个安装选项供用户选择，分别为安装 Visual Studio 2008、安装产品文档和检查 Service Release。此处选择“安装 Visual Studio 2008”。

（2）单击“安装 Visual Studio 2008”选项后，安装程序会弹出如图 2.10 所示的 Visual Studio 2008 安装向导界面。

图 2.9　Visual Studio 2008 安装程序界面

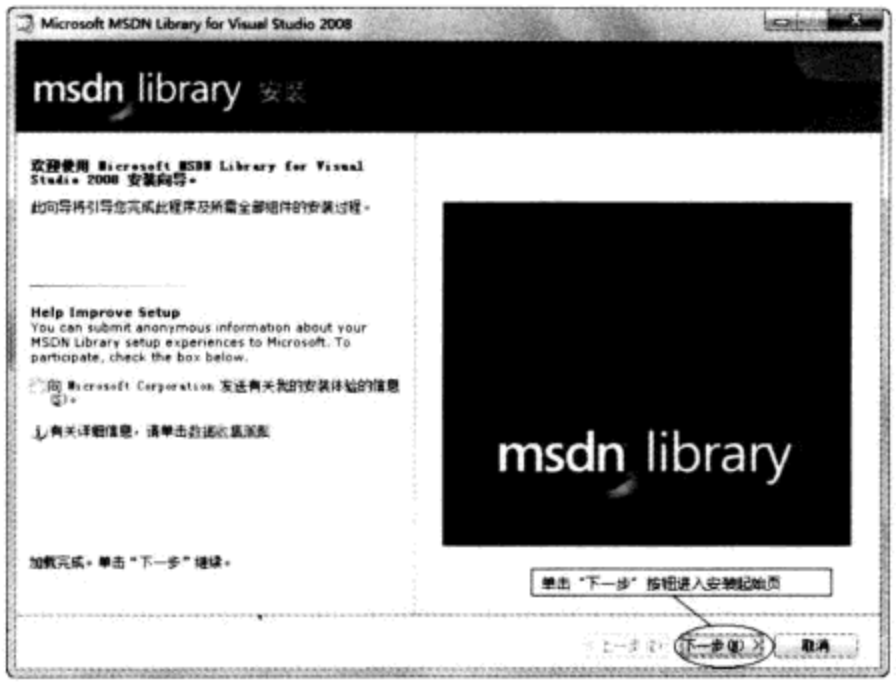


图 2.10　Visual Studio 2008 安装向导

（3）单击“下一步”按钮，安装程序会跳转到“Visual Studio 2008 安装程序-起始页”界面。该界面左侧显示关于 Visual Studio 2008 安装程序的所需组件信息，右侧显示用户许可协议，如图 2.11 所示。

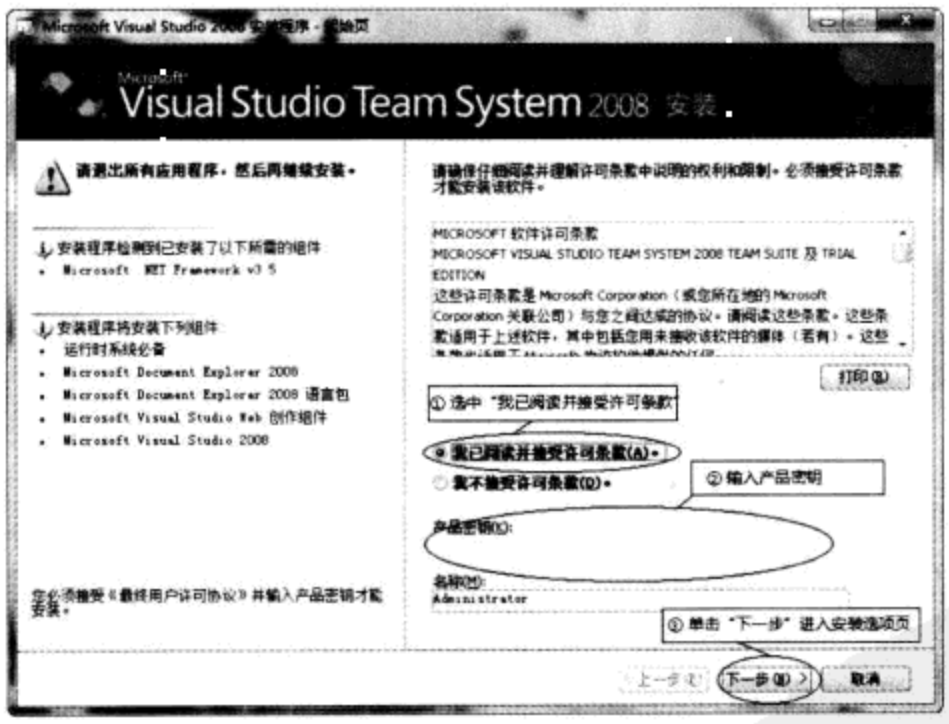


图 2.11　Visual Studio 2008 安装程序-起始页

（4）选中“我已阅读并接受许可条款”单选框，单击“下一步”按钮，安装程序会跳转到“Visual Studio 2008 安装程序-选项页”界面，用户可对自己需要的功能进行选择，并

对产品安装路径进行设置，通常用户选择要安装的功能为默认值，产品默认路径为“C:\Program Files\Microsoft Visual Studio 9.0\”，如图 2.12 所示。

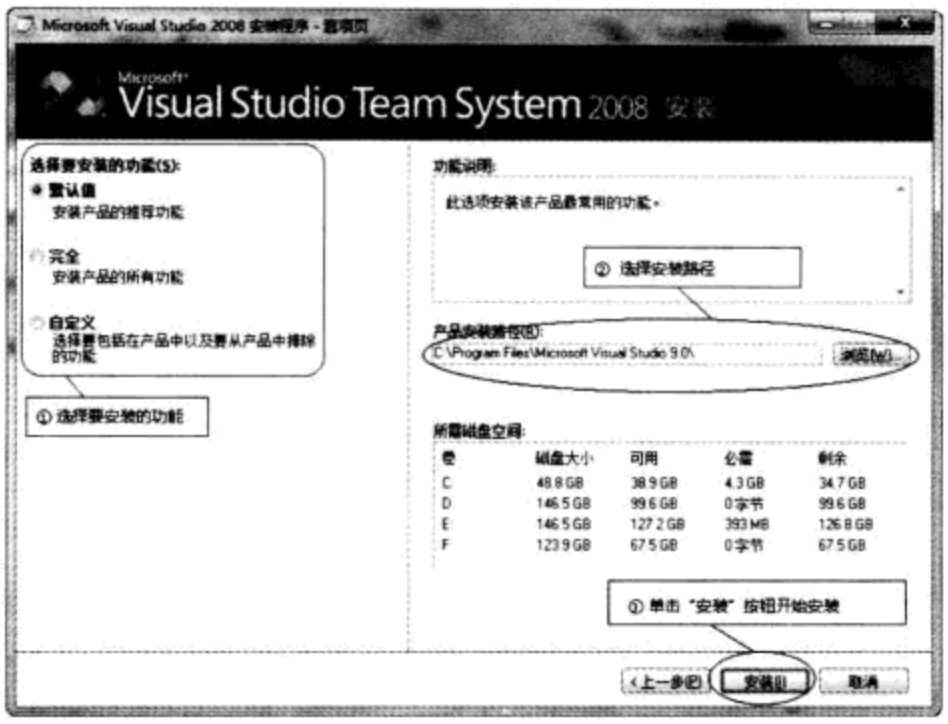


图 2.12　Visual Studio 2008 安装程序-选项页

（5）如果用户想手动设置安装的项目，可以选择“自定义”。然后，单击“安装”按钮进入自定义安装界面，在该界面中选择自己希望安装的项目，如图 2.13 所示。

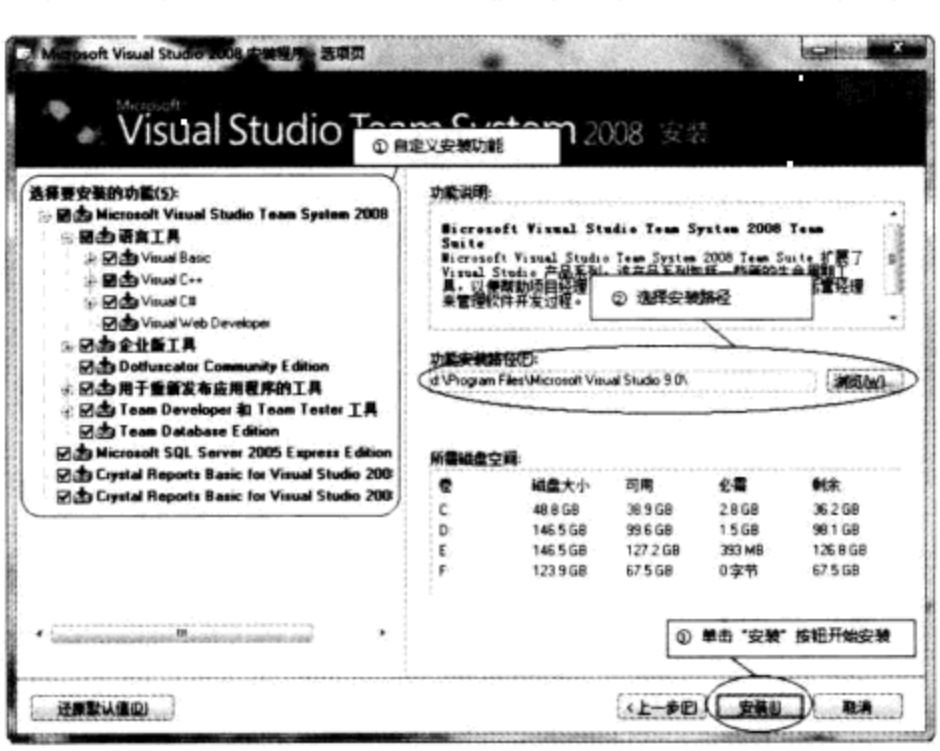


图 2.13　自定义安装界面

（6）安装路径设置完毕后，单击“安装”按钮，程序会跳转到“Visual Studio 2008 安装程序-安装页”界面，该界面中显示正在安装组件，如图 2.14 所示。

（7）等待 Visual Studio 2008 安装程序自动完成后，程序会打开“Visual Studio 2008 安装程序-完成页”界面，单击“完成”按钮，完成 Visual Studio 2008 开发环境的安装，如图 2.15 所示。

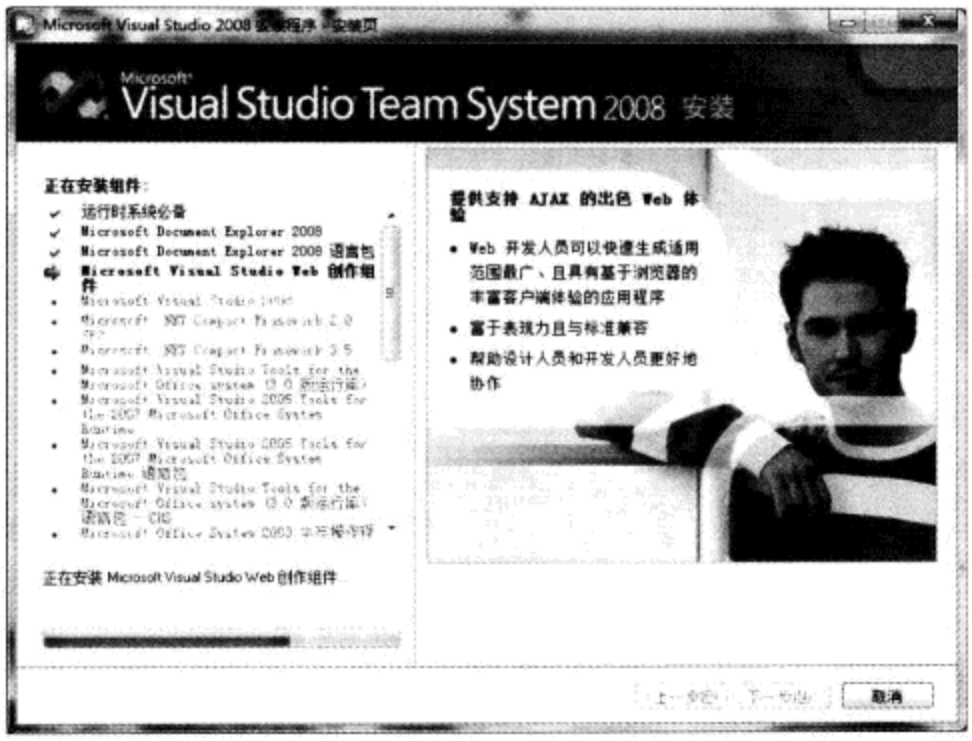


图 2.14　Visual Studio 2008 安装程序-安装页

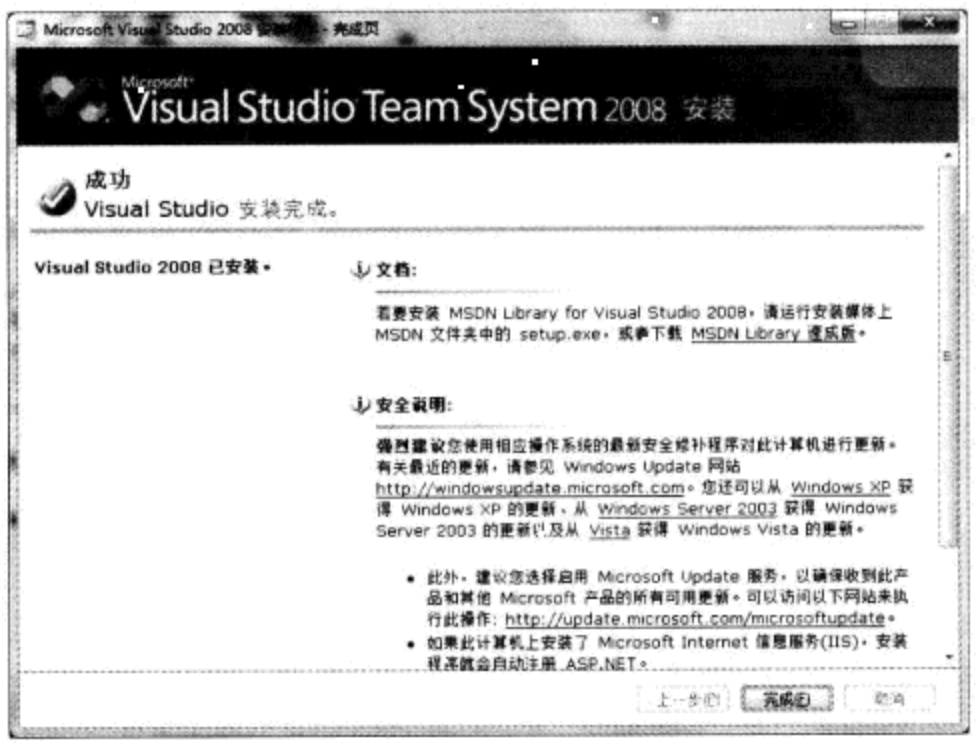


图 2.15　Visual Studio 2008 安装程序-完成页

2.2.2　创建第一个“Hello Word”

安装 Visual Studio 2008 之后，现在开始开发自己的第一个 Web 程序，详细步骤说明如下。

（1）依次选择“开始”/“所有程序”/“Microsoft Visual Studio 2008”/“Microsoft Visual Studio 2008”选项，便可启动 Visual Studio 2008，如图 2.16 所示。

（2）在“模板”列表中，首先选择新建网站类型，然后选择框架类型和开发语言，最后指定新建网站存储的路径与名称，如图 2.17 所示，单击“确定”按钮。这样一个 ASP.NET 网站便创建成功了。

图 2.16　启动 Visual Studio 2008

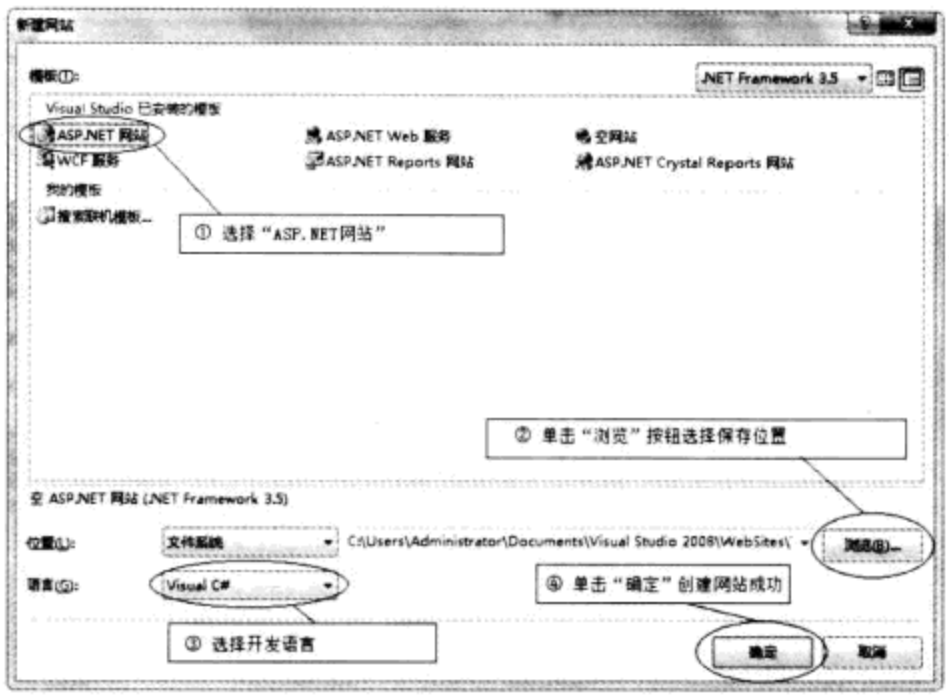


图 2.17　“新建网站”对话框

（3）网站创建成功后，是一个空白网站，我们可以在此基础上进行网站的开发，网站创建成功后如图 2.18 所示。

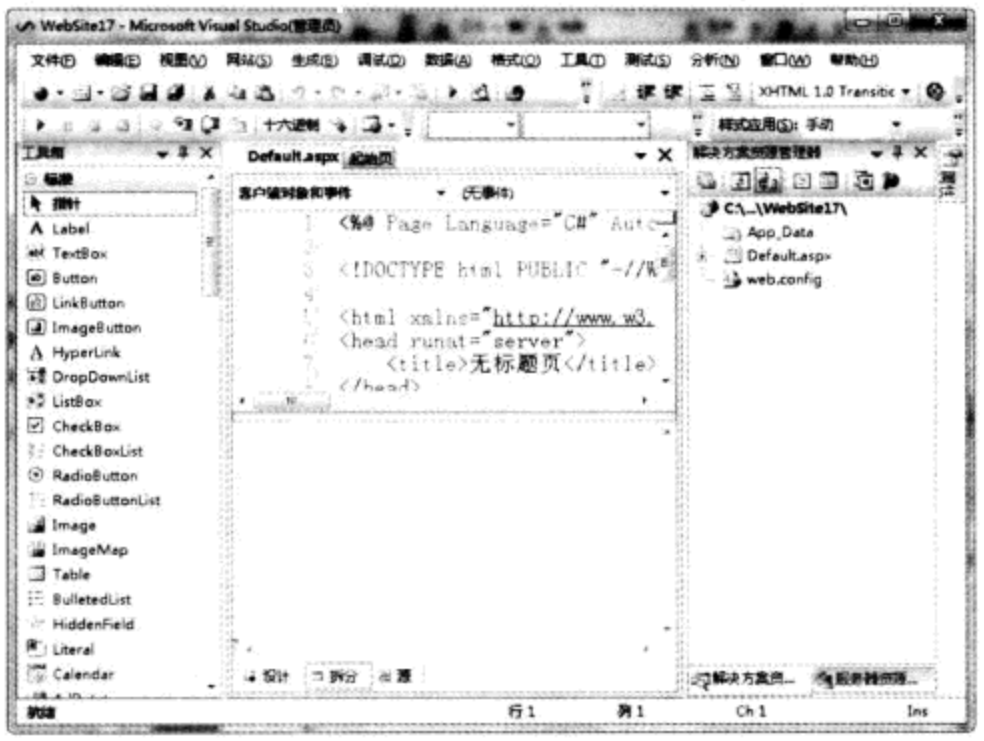


图 2.18　网站创建成功后的界面

（4）双击空白区域，进入后台代码编写界面。在 Page_Load 事件中编写代码，实现输出“Hello Word”，如图 2.19 所示。

（5）编写代码之后，运行程序，效果如图 2.20 所示。

2.2.3　Visual Studio 2008 功能介绍

Visual Studio 2008 的功能窗口如图 2.21 所示。

下面将对 Visual Studio 2008 集成开发环境窗口中的各项功能进行详细介绍。

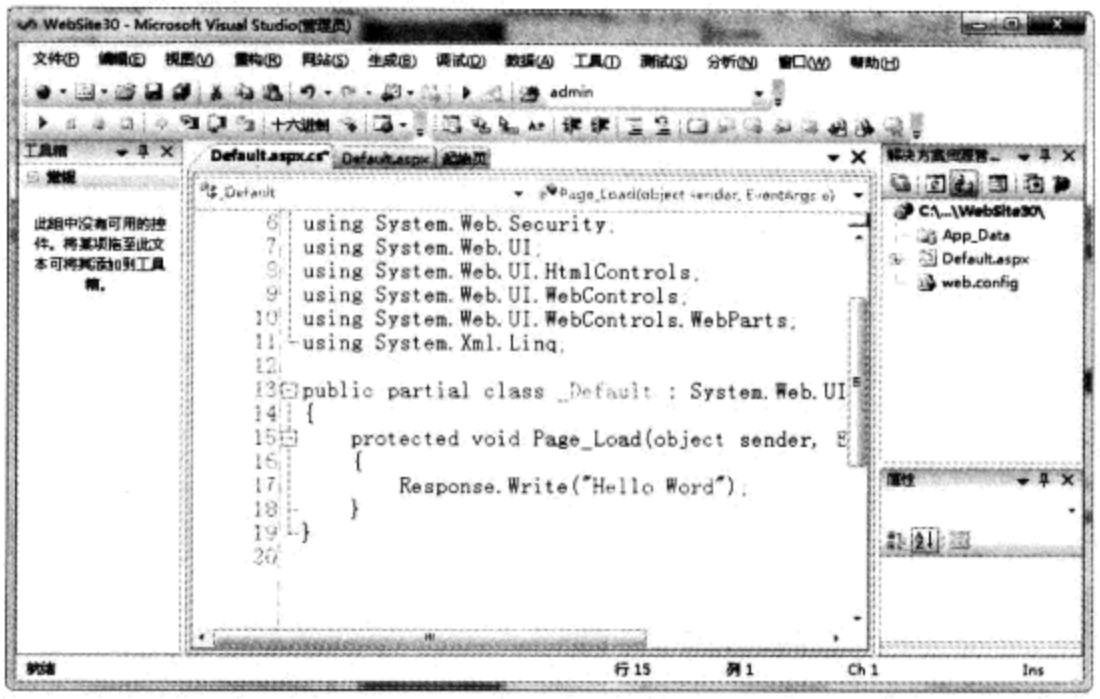


图 2.19　编写代码输出“Hello Word”

图 2.20　输出“Hello Word”

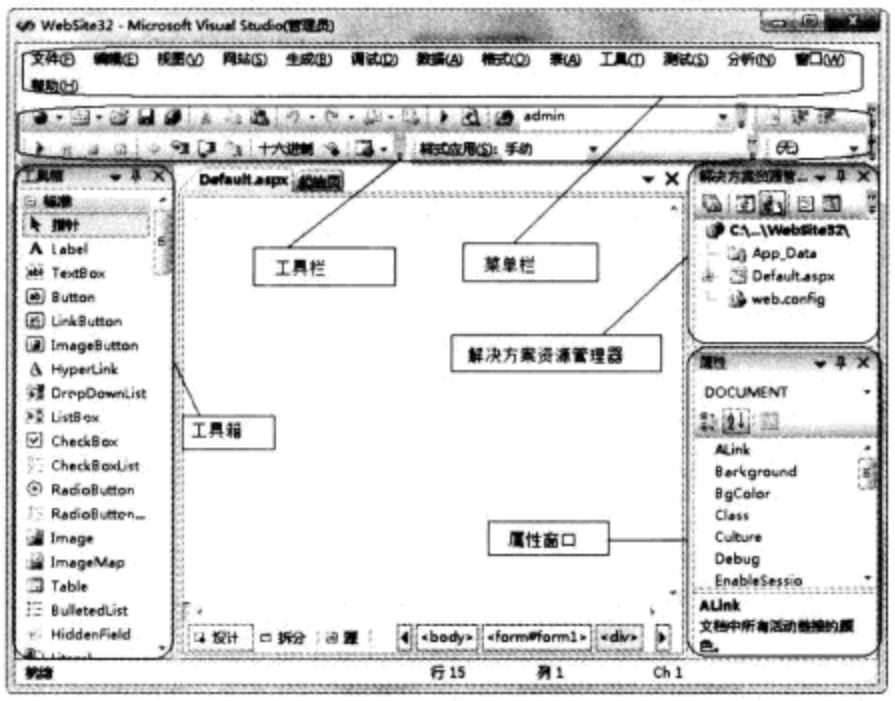


图 2.21　Visual Studio 2008 开发环境

1. 菜单栏

菜单栏显示了所有可用的 Visual Studio 2008 命令，除了“文件”、“编辑”、“视图”、“窗口”和“帮助”菜单之外，还提供编程专用的功能菜单，如“网站”、“生成”、“调试”、“工具”和“测试”等，如图 2.22 所示。

图 2.22　Visual Studio 2008 菜单栏

每个菜单项中都包含若干个菜单命令，分别执行不同的操作。下面详细介绍主要菜单的功能。

（1）“文件”菜单

“文件”菜单包括“新建”、“打开”、“添加”、“关闭”、“保存 Default.aspx”、“Default.aspx 另存为”、“全部保存”、“导出模板”、“页面设置”、“打印”、“最近的文件”、“最近的项目”和“退出”命令。通过“文件”菜单可以打开和保存项目、网站和文件，以及导出项目模板。另外，它还列出了最近打开的文件和最近打开的项目，详细介绍如图 2.23 所示。

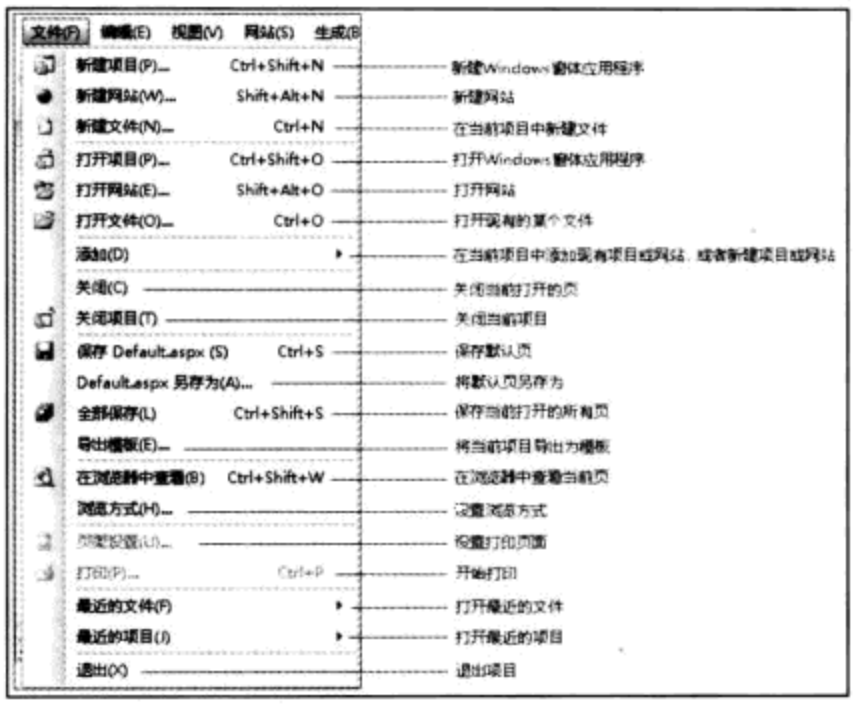


图 2.23　“文件”菜单下的所有子菜单

在“文件”菜单中，比较常用的是“新建项目”、“新建网站”、“打开项目”和“打开网站”命令。例如，通过选择“文件”/“打开网站”命令，可以打开一个现有的网站，如图 2.24 所示。

图 2.24　“打开网站”对话框

（2）“视图”菜单

“视图”菜单包括用于显示或隐藏集成开发环境中的各种窗口、工具栏及其他组成部

分的命令，详细介绍如图 2.25 所示。

（3）“生成”菜单

“生成”菜单主要包含编译项目及网站发布的各种命令，详细介绍如图 2.26 所示。

图 2.25 “视图”菜单下的所有子菜单

图 2.26 “生成”菜单下的所有子菜单

（4）“调试”菜单

“调试”菜单包括调试工程的各种命令，如“启动调试”、“开始执行”和“新建断点”等，详细介绍如图 2.27 所示。

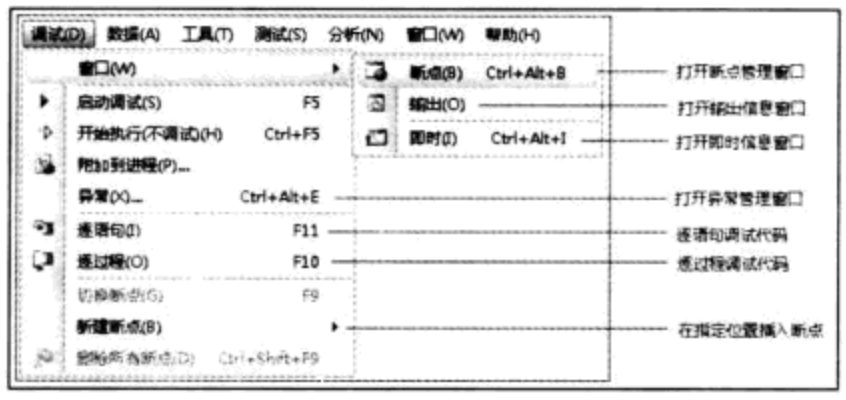


图 2.27 “调试”菜单下的所有子菜单

例如，当选择“调试”/“窗口”/“断点”命令时，会打开断点管理窗口，在该窗口中会显示当前插入的断点的相关信息，如图 2.28 所示。

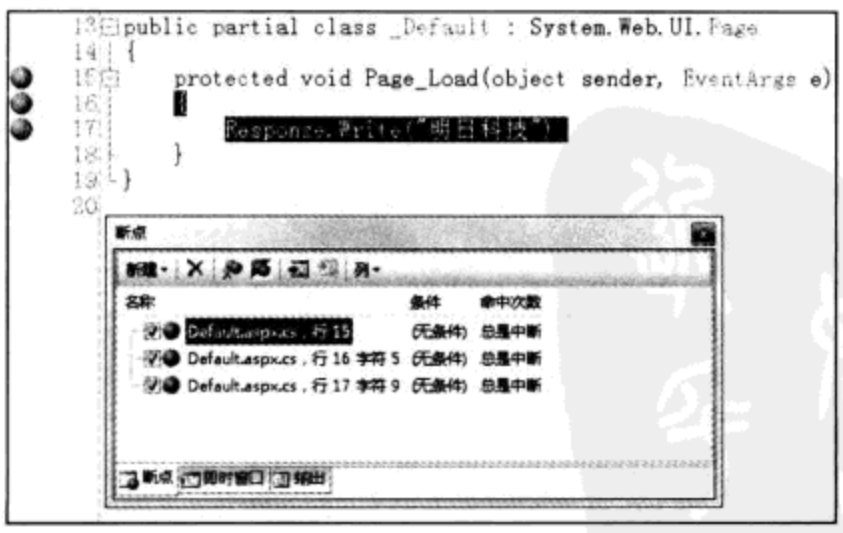


图 2.28 断点管理窗口

2. 工具栏

通过工具栏可以迅速访问常用的菜单命令。一般情况下，集成环境中只显示标准工具栏，要显示其他工具栏，可以通过“视图”/“工具栏”命令或鼠标右键单击工具栏实现，如图 2.29 所示。

图 2.29　Visual Studio 2008 工具栏

3. 工具箱

Visual Studio 2008 的窗口左侧有一个隐藏的工具箱，当用户将鼠标放置在工具箱标签上时，会弹出一个“工具箱”对话框，如图 2.30 所示。

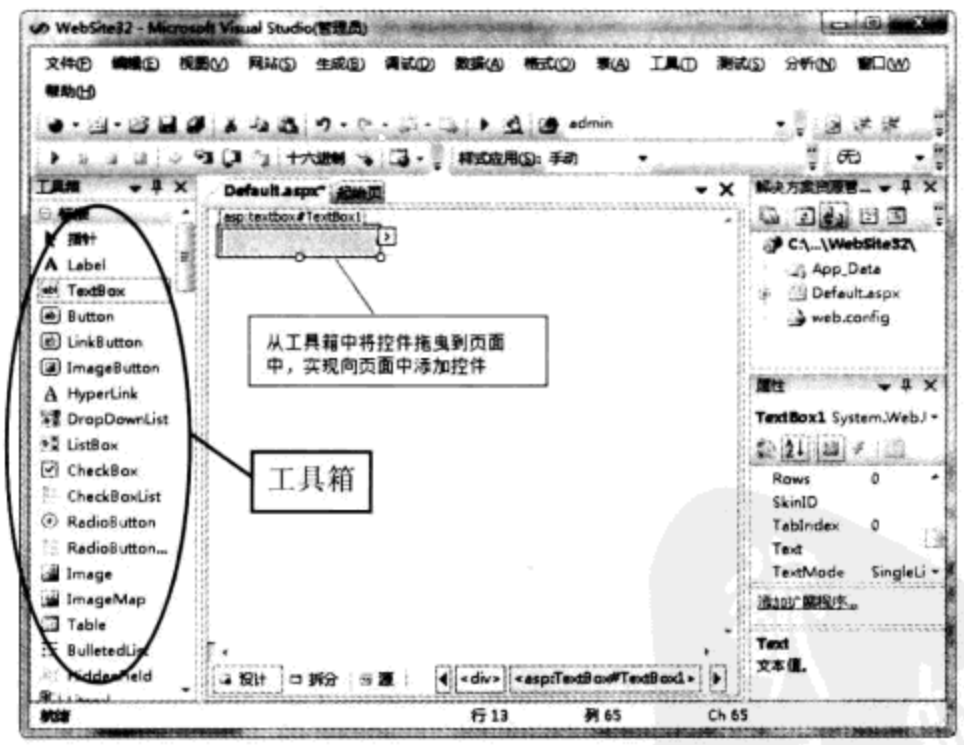


图 2.30　“工具箱”对话框

与 Visual Studio 2005 不同的是，“工具箱”对话框把 AJAX 服务器控件添加到工具箱中并分类列出，用户可以直接使用这些控件，大大节省了编写代码的时间，加快程序开发的进度，如图 2.31 所示。

图 2.31 “工具箱”对话框

学习手册 说明 将 AJAX 控件拖曳到页面后，就可以开发绚丽多彩的 AJAX 程序了。

4. 属性窗口

在进行界面设计时，需要使用“属性”窗口。在该对话框中，用户可以对页面的一些属性值进行设置，这些设置的属性值也会自动添加到 HTML 源码中，属性值会随着标签值的改变而改变，如图 2.32 所示。

5. 窗体设计器

所谓窗体设计器，是指在“解决方案资源管理器”中双击鼠标打开一个后缀名为.aspx 的文件，该文件所显示的内容就是 Web 窗体设计器。开发人员可在窗体设计器中按照页面的实际需要进行设计，如图 2.33 所示。

图 2.32 “属性”对话框

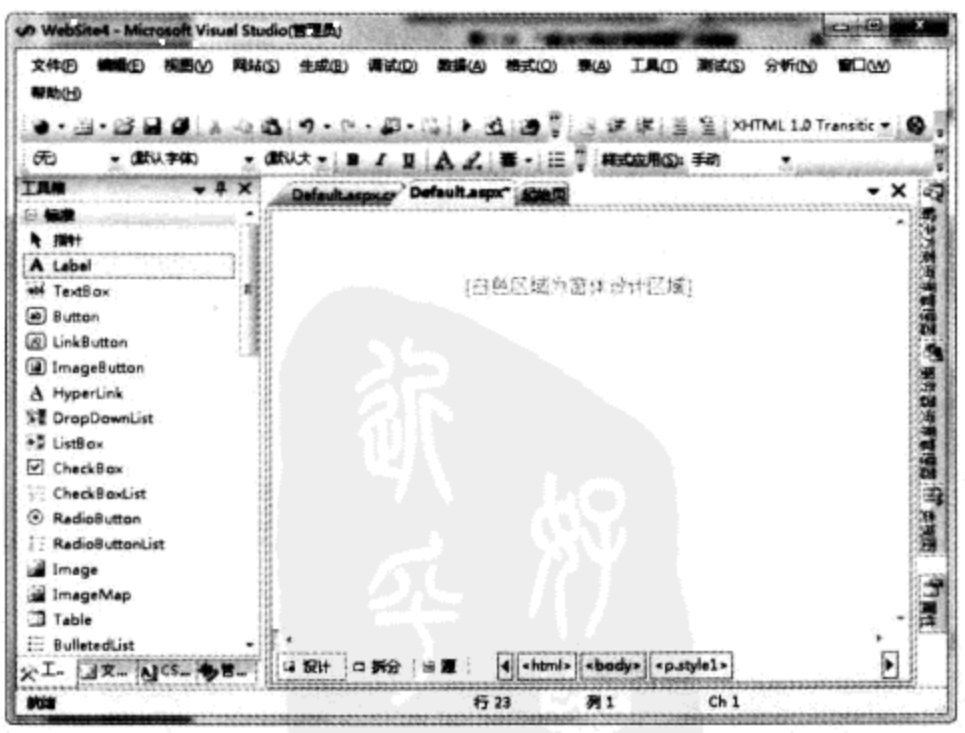


图 2.33 窗体设计器

6. 代码编辑器窗口

代码编辑器窗口是指在窗体设计器上任意处双击鼠标进入到另一个后缀名为.aspx.cs 文件中，用户在该文件中编写服务器代码，实现各种功能，如图 2.34 所示。

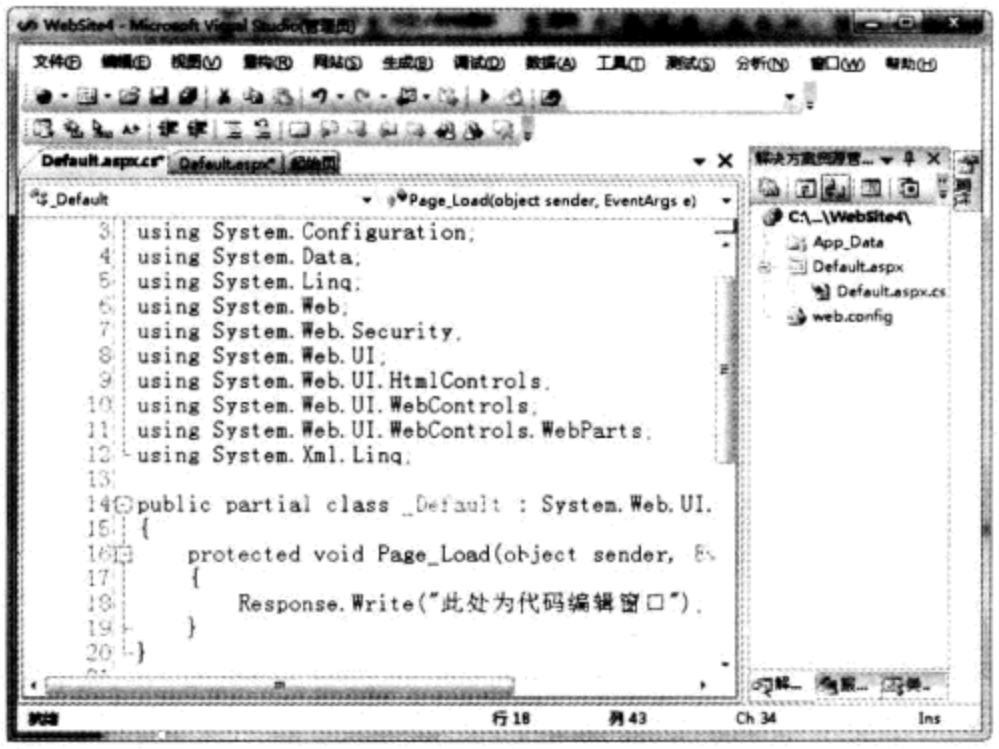


图 2.34　代码编辑器窗口

2.3　安装 MSDN 帮助系统

MSDN 是 Microsoft Visual Studio 2008 自带的一种帮助文档，开发人员在编写应用程序时可以直接在 MSDN 中查找帮助，以解决在开发过程中遇到的疑难问题。本节主要讲述 MSDN 帮助系统的安装与使用。

2.3.1　安装 MSDN 帮助系统

专题讲座：光盘\MR\Video\2\安装 MSDN 帮助系统.exe

➤➤➤视频速递：本视频将详细地演示如何安装 MSDN 帮助系统。

Visual Studio 2008 中提供了一个强大的帮助工具，简称 MSDN，MSDN 是开发人员在开发项目过程中最好的帮手，它包含了对 C#语言各方面知识的讲解，并附有示例代码。如果将开发人员视为一名剑客，Visual Studio 2008 就是剑客手中的一把宝剑，而 MSDN 则是剑客身上的铠甲。有了宝剑和铠甲，就可以纵横于江湖。本节将对 MSDN 的安装步骤进行详细讲解，安装 MSDN 的步骤如下。

（1）把 Visual Studio 2008 MSDN 安装盘放入光驱中，光盘自动运行后会进入安装程序文件界面，如果光盘不能自动运行，双击 setup.exe 可执行文件，应用程序自动跳转到如图 2.35 所示的 Visual Studio 2008 MSDN 安装向导界面。

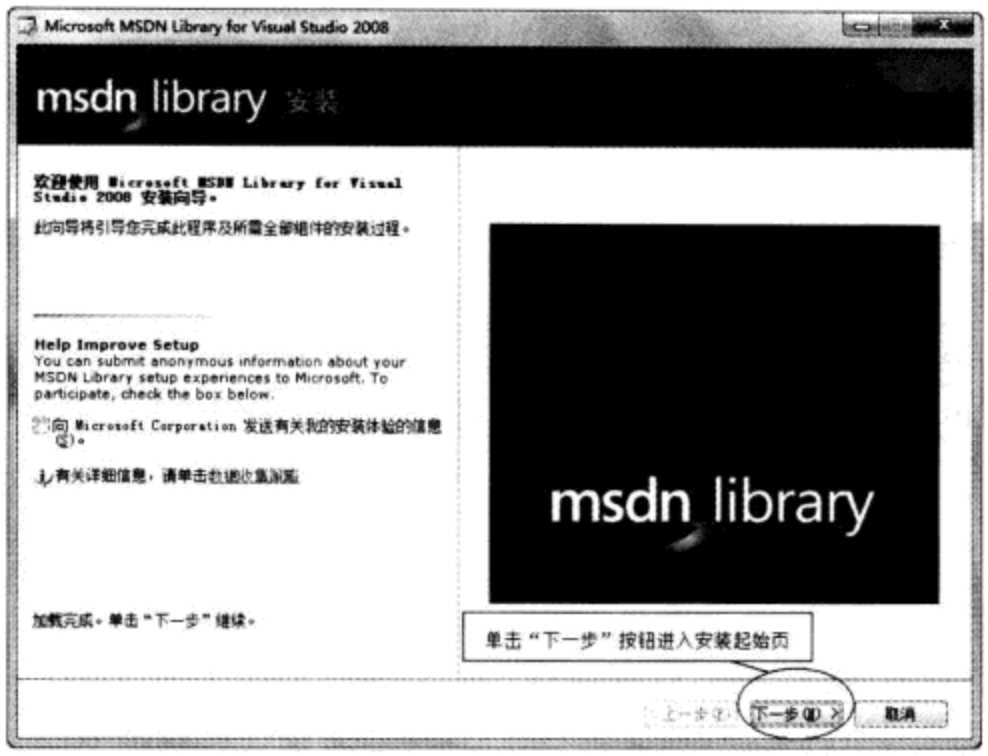


图 2.35　Visual Studio 2008 MSDN 安装向导界面

（2）单击“下一步”按钮，进入 Visual Studio 2008 MSDN 安装程序起始页，如图 2.36 所示。在这里如果用户单击“取消”按钮，则会退出 Visual Studio 2008 MSDN 安装程序。

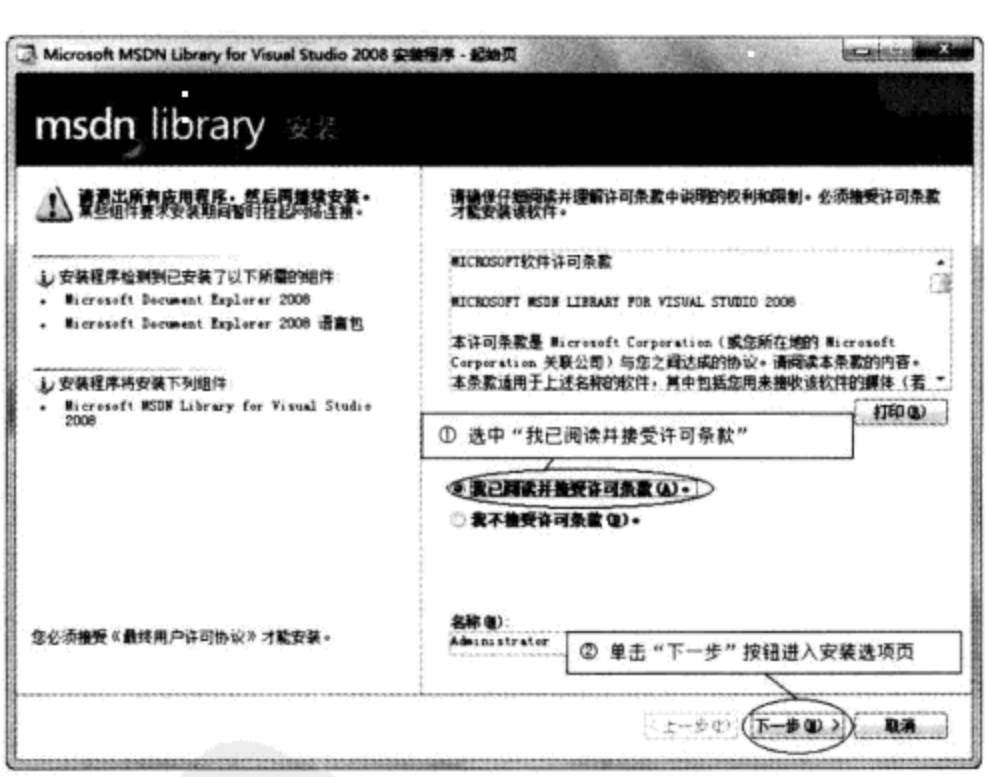


图 2.36　Visual Studio 2008 MSDN 安装程序起始页

（3）选中图 2.36 中“我已阅读并接受许可条款”单选框，单击“下一步”按钮，进入 Visual Studio 2008 MSDN 安装程序选项页，用户可以自己选择 Visual Studio 2008 MSDN 的 3 种安装类型（完全、最小和自定义），默认为完全安装，如图 2.37 所示。

（4）如果用户不想完全安装 Visual Studio 2008 MSDN，可以选择“自定义”，然后单击“安装”按钮进入自定义安装界面，如图 2.38 所示。在自定义安装界面中，用户可以有选择性地安装自己需要的项目。

（5）在图 2.37 或图 2.38 界面中，用户可以单击“浏览”按钮选择 Visual Studio 2008

MSDN 的安装路径（默认安装路径为“C:\Program Files\MSDN\MSDN 9.0\”）。单击“安装”按钮，进入 Visual Studio 2008 MSDN 安装界面，如图 2.39 所示。

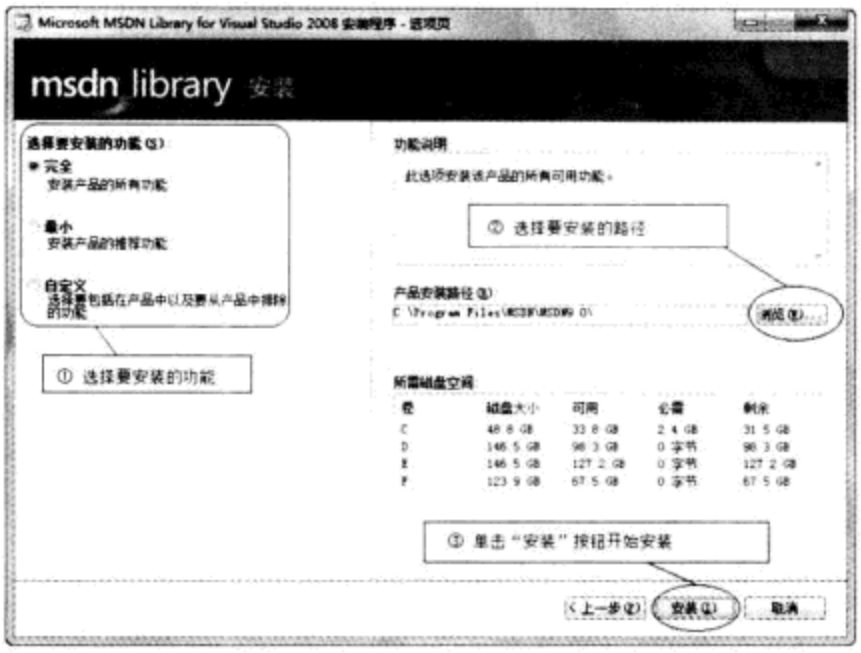


图 2.37　Visual Studio 2008 MSDN 安装程序选项页

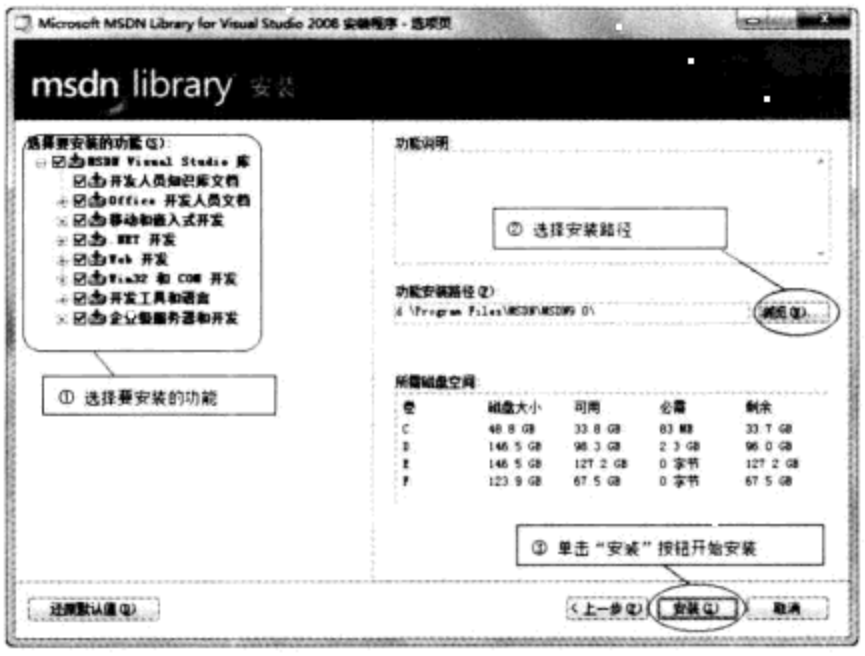


图 2.38　自定义安装 MSDN

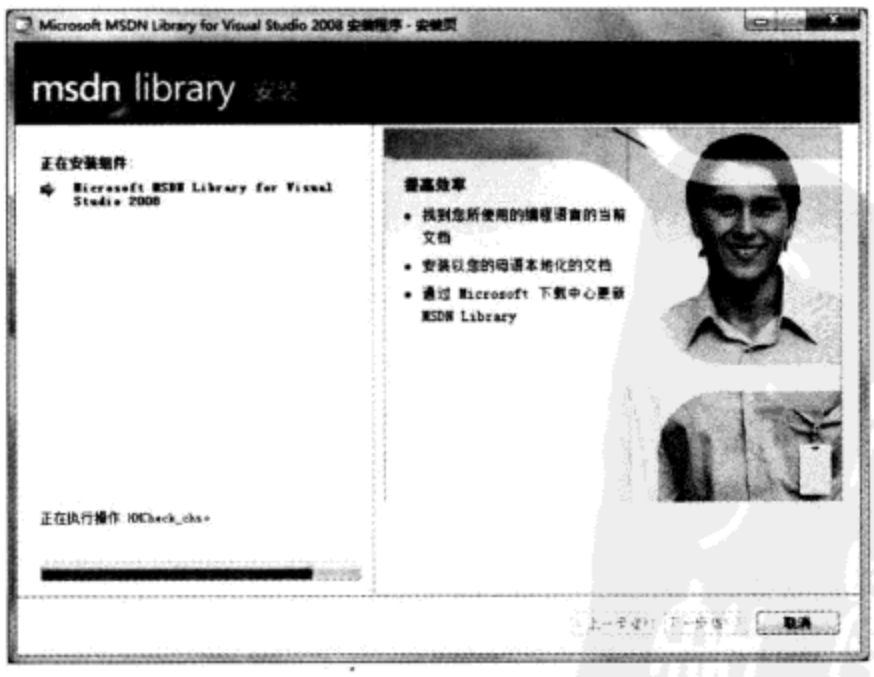


图 2.39　Visual Studio 2008 MSDN 安装界面

(6)安装完成之后，单击“完成”按钮，如图 2.40 所示，完成对 Visual Studio 2008 MSDN 的安装。

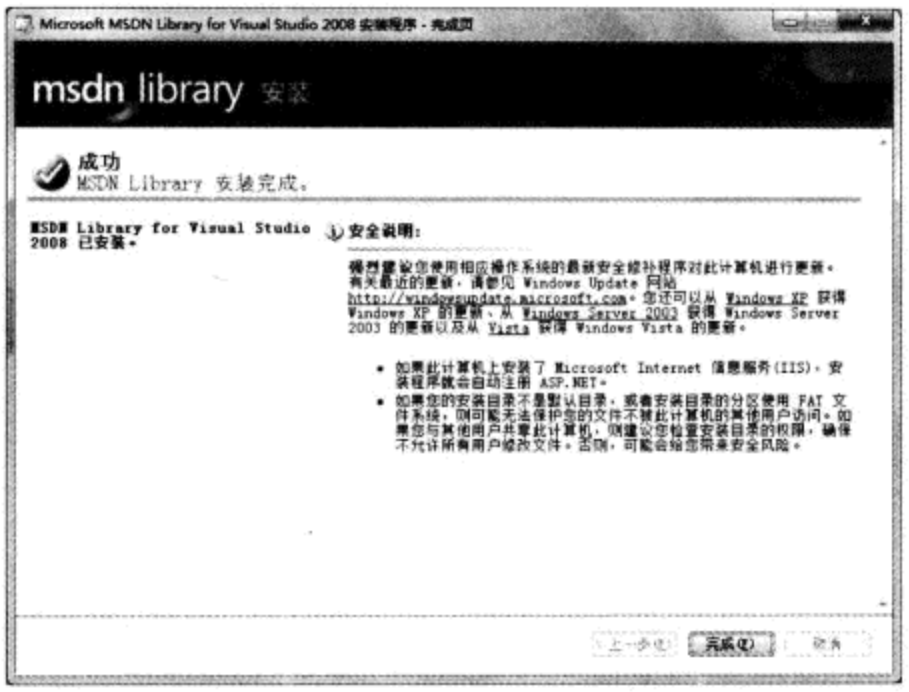


图 2.40　Visual Studio 2008 MSDN 安装完成界面

2.3.2　使用 MSDN 帮助系统

用户安装完 MSDN 之后，可以依次选择“开始”/“所有程序”/“Microsoft Developer Network”/“MSDN Library for Visual Studio 2008 简体中文”打开 Visual Studio 2008 MSDN 帮助文档，其起始页如图 2.41 所示。

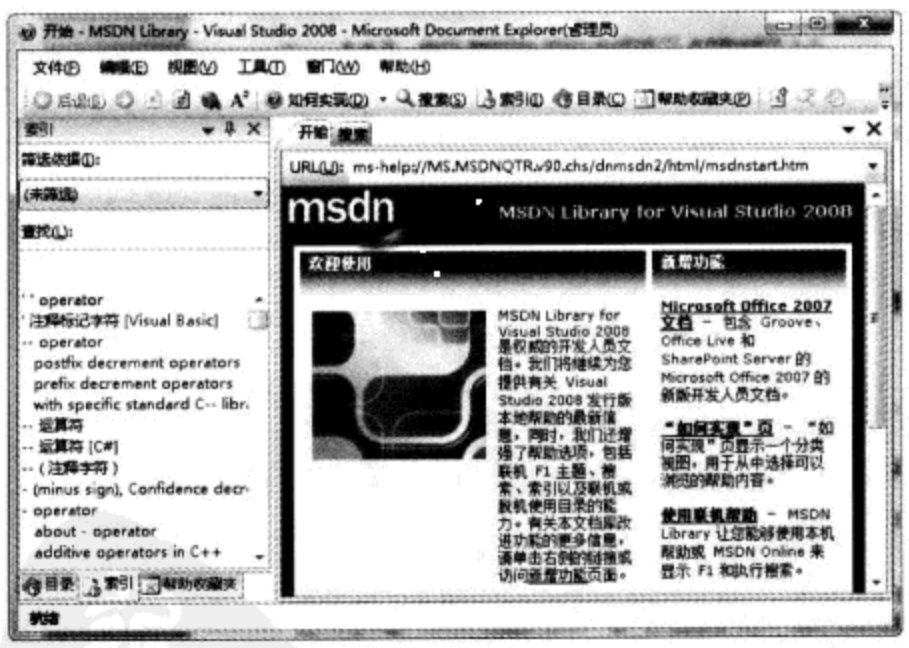


图 2.41　MSDN 帮助起始页

使用 MSDN 帮助文档可以大大提高开发人员开发程序的效率，下面就来简单了解一下如何使用 MSDN 帮助文档。使用 MSDN 帮助文档时可以采用搜索和索引两种方式查找所需要的内容，如图 2.42 所示。

例如，下面以通过“搜索”查找 split 方法为例，说明如何使用 MSDN 帮助文档，选择“搜索”选项卡，在查找文本框中输入“split”，然后按〈Enter〉键或者单击“搜索”按钮，这时就会在 MSDN 帮助文档左侧显示其有关信息，用户可以双击查看其详细信息，

具体如图 2.43 所示。

图 2.42　MSDN 帮助文档查找方式

图 2.43　使用 MSDN 帮助文档

2.4　本 章 小 结

本章主要介绍构建 ASP.NET 开发环境时软件和硬件的要求，并详细介绍如何安装 IIS 以及 Visual Studio 2008 开发工具，同时为读者演示如何创建一个网站，输出“Hello Word”。最后，介绍了如何安装 MSDN 帮助文档及其用法。

第 3 章

ASP.NET 网站开发基础

（名师课堂：44 分钟）

学习 ASP.NET 前读者需要对 ASP.NET 有一个初步的了解。ASP.NET 是由 Microsoft 公司推出的新一代 Web 开发平台，程序开发人员可通过 ASP.NET 实现动态或静态网站的开发。本章将详细介绍 Visual Studio 2008 开发环境中设计 ASP.NET 网页的三种视图，以及如何运行程序和配置 IIS。同时本章也对网页扩展名和页面指令等内容做了详细的介绍。通过本章的学习，可以掌握以下知识：

- 了解设计 Web 网页的三种视图
- 了解如何运行 ASP.NET 程序
- 学习配置 IIS 虚拟目录
- 了解 ASP.NET 网页扩展名
- 了解什么是页面指令及其作用
- 掌握如何注释 ASPX 文件中的代码

3.1　设计 ASP.NET 网站

通过第 2 章的学习，读者对 Visual Studio 2008 已经大致有所了解，并且知道如何创建第一个“Hello World”程序了。从本节开始，我们将带领读者学习如何设计 Web 页面，以及了解网站中常用的项目。

3.1.1　设计 Web 页面

专题讲座：光盘\MR\Video\3\设计 Web 页面.exe

➤➤➤视频速递：通过本视频读者可以更快地了解、设计 Web 页面。

创建了一个网站后，接下来的工作就是设计 Web 页面。在 Visual Studio 2008 中一共有三种视图模式：“设计”视图、“拆分”视图和“源”视图。下面分别介绍一下这三种视图的不同及其使用方法。

1.“设计”视图

在“设计”视图中，用户可以从“工具箱”选项卡中直接选择各种控件拖曳到 Web 页面上，在该视图下可以直接看到控件添加到页面后的效果。例如，在“设计”视图中添加一个 TextBox 控件如图 3.1 所示。

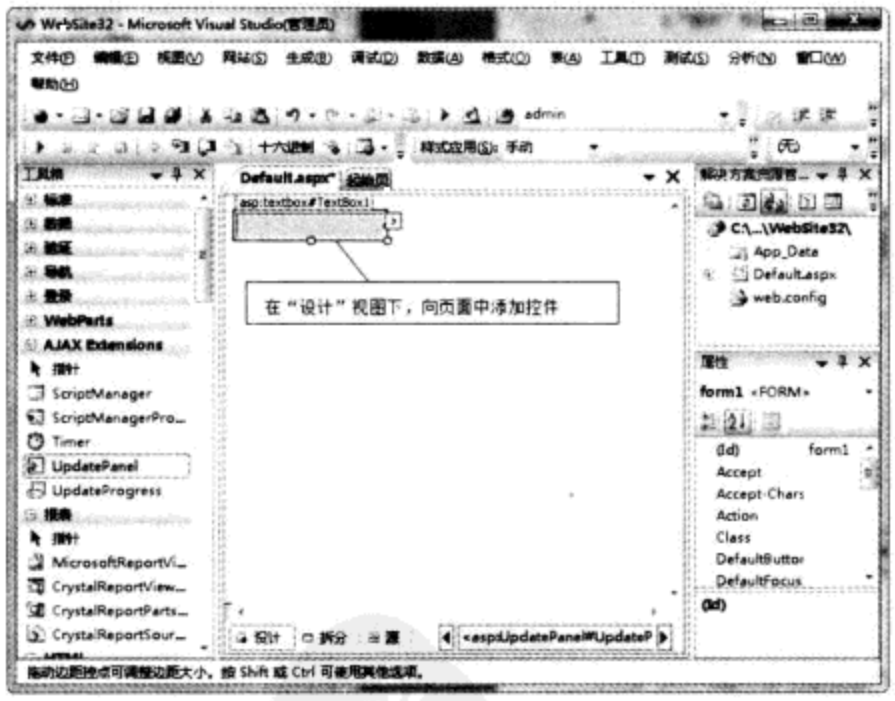


图 3.1　在“设计”视图中添加控件

2.“拆分”视图

在“拆分”视图中用户不但可以从“工具箱”选项卡中拖曳控件到 Web 页面上，也可以使用代码添加控件，在该视图下可以直接看到控件添加到页面后的效果。例如，页面中添加一个 TextBox 控件后，在“拆分”视图下就能看到页面中的控件和控件的 HTML 代码。并且，在“拆分”视图下可以编写 HTML 代码和 JavaScript 脚本等，如图 3.2 所示。

图 3.2　在“拆分”视图中添加控件

3. “源”视图

在“源”视图中用户可以从“工具箱”选项卡中拖曳控件到 Web 页面上，也可以使用代码进行添加控件，但是在该视图中并不能看到控件被添加到 Web 页面后的效果。例如，在“源”视图中使用代码添加一个 Label 控件如图 3.3 所示。

图 3.3　“源”视图

通过这几种视图可以设计 Web 页面，例如，我们在“设计”视图下设计一个用户登录界面，如图 3.4 所示。

图 3.4　“设计”视图下设计“用户登录界面”

3.1.2　运行 Web 网站

当我们设计 Web 页面之后，该如何运行程序呢？Visual Studio 2008 中有多种方法运行应用程序。可以单击“调试”菜单，选择“启动调试”命令运行应用程序，如图 3.5 所示；也可以单击工具栏上的“▶”按钮运行应用程序，如图 3.6 所示，还可以直接按〈F5〉键运行程序。

上述设计的“用户登录界面”的运行结果如图 3.7 所示。

图 3.5 “调试”菜单运行应用程序

图 3.6　工具栏运行应用程序

图 3.7　运行结果

3.1.3　配置 IIS 虚拟目录

专题讲座：光盘\MR\Video\3\配置 IIS 虚拟目录.exe

➤➤➤视频速递：通过本视频读者可以直观地学习如何配置 IIS 虚拟目录。

在网站完成之后，就需要在 IE 浏览器中进行浏览了。IIS 作为当今流行的 Web 服务器之一，提供了强大的 Internet 服务功能，可以发布、测试和维护自己的 Web 页和 Web 站点。下面以 Windows 7 系统为例，介绍如何在 IIS 管理器中配置网站，操作步骤说明如下：

（1）单击“开始”/“控制面板”命令，打开“控制面板”窗口。

（2）单击“管理工具”，打开“管理工具”窗口，双击“Internet 信息服务（IIS）管理器”图标，弹出如图 3.8 所示的“Internet 信息服务（IIS）管理器”窗口。

图 3.8 “Internet 信息服务（IIS）管理器”窗口

（3）在“Internet 信息服务（IIS）管理器”窗口中，打开“网站”节点，右键单击“Default Web Site”节点，在弹出的快捷菜单中选择“添加应用程序”选项，如图 3.9 所示。

图 3.9　选择“添加应用程序”选项

（4）弹出如图 3.10 所示的“添加应用程序”对话框，在“别名”文本框中输入虚拟目录别名，本例输入 MyWeb，这个名称是访问网页的时候需要输入的名称。在“物理路径”文本框中输入网站的路径，或者单击“ ... ”按钮选择路径。

（5）单击“确定”按钮，完成网站的架设，如图 3.11 所示。

图 3.10　“添加应用程序”对话框

图 3.11　完成网站架设

（6）架设完网站后，我们应该如何浏览自己的网站呢？如果已经为网站绑定了 IP 地址，则可以在浏览器中输入 http://192.168.1.188/MyWeb/进行浏览，其中 IP 地址是主机服务器的 IP 地址。如果没有绑定 IP 地址，则在浏览器中输出 http://localhost/MyWeb 进行浏览，两种方式浏览网页的效果如图 3.12 所示。

介绍到这里，读者可能会有个疑问，到底该如何为网站绑定 IP 呢？为什么要绑定 IP 呢？之所以绑定 IP 主要是因为绑定之后在浏览器中可以通过 IP 地址访问网站。绑定 IP 的方法非常简单，下面作者就详细地介绍一下如何为网站绑定 IP。首先，选中“Default Web Site”，单击其右侧“操作”面板中的“绑定...”选项，如图 3.13 所示。

图 3.12　两种方式浏览网页

图 3.13　“操作”面板中的“绑定...”选项

单击“绑定...”后，会打开“网站绑定”窗口，在该窗口中可以添加、编辑和删除网站绑定信息，如图 3.14 所示。

在该窗口中单击“编辑”按钮，打开“编辑网站绑定”窗口，在该窗口中可以选择网站要绑定的 IP 地址，如图 3.15 所示。选择 IP 地址之后，单击“确定”按钮完成 IP 地址的绑定。

图 3.14　“网站绑定”窗口

图 3.15　设置绑定的 IP 地址

3.2　ASP.NET 网页语法

专题讲座：光盘\MR\Video\3\学习 ASP.NET 网页语法.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 ASP.NET 网页语法。

本节将会对 ASP.NET 中常见的扩展名进行介绍，同时，了解页面的常用指令、如何注释代码以及如何在网页中使用代码块语法。这些内容在日常开发中经常用到，所以请读者认真学习本节的内容。

3.2.1　ASP.NET 网页扩展名

可以把 ASP.NET 网页扩展名理解为 ASP.NET 文件的“身份证”，不同的扩展名决定了不同文件的类型和作用。通过 Internet 信息服务（IIS）将文件扩展名映射到 ASP.NET 运行处理。例如，Web 页面的扩展名为.aspx、母版页的扩展名为.master 等。在 ASP.NET

网页中包含很多种文件类型，其扩展名及具体描述如表 3.1 所示。

表 3.1　ASP.NET 网页扩展名

文 件 类 型	说　明
.asax	通常是 Global.asax 文件，该文件包含从 HttpAppliction 类派生并表示该应用程序的代码
.ascx	Web 用户控件文件，可重复使用
.ashx	一般处理程序文件，该文件包含实现 IhttpHandler 接口以处理所有传入请求的代码
.asmx	XML Web Services 文件，该文件包含通过 SOAP 方式可用于其他 Web 应用程序的类和方法
.aspx	ASP.NET Web 窗体文件，该文件可包含 Web 控件和其他业务逻辑
.config	通常是 Web.config 配置文件，该文件包含其配置各种 ASP.NET 功能的 XML 元素
.dll	已编译的类库文件，可以将类的源代码放在 App_Code 子目录下
.master	母版页，定义应用程序中引用母版页的其他网页的布局
.sitemap	站点地图文件，该文件包含网站的结构。ASP.NET 中附带了一个默认的站点地图提供程序，它使用站点地图文件可以很方便地在网页上显示导航控件
.skin	用于确定显示格式的外观文件
.css	用于确定 HTML 元素格式的样式表文件
.mdb、.ldb	Access 数据库文件
.mdf	SQL 数据库

【例 3.1】 创建网站之后，在其根目录下就会出现一个扩展名为.aspx 的文件和一个扩展名为.aspx.cs 的文件。.aspx 是创建的 Web 窗体文件，.cs 是运行时要编译的类源代码文件，如图 3.16 所示。

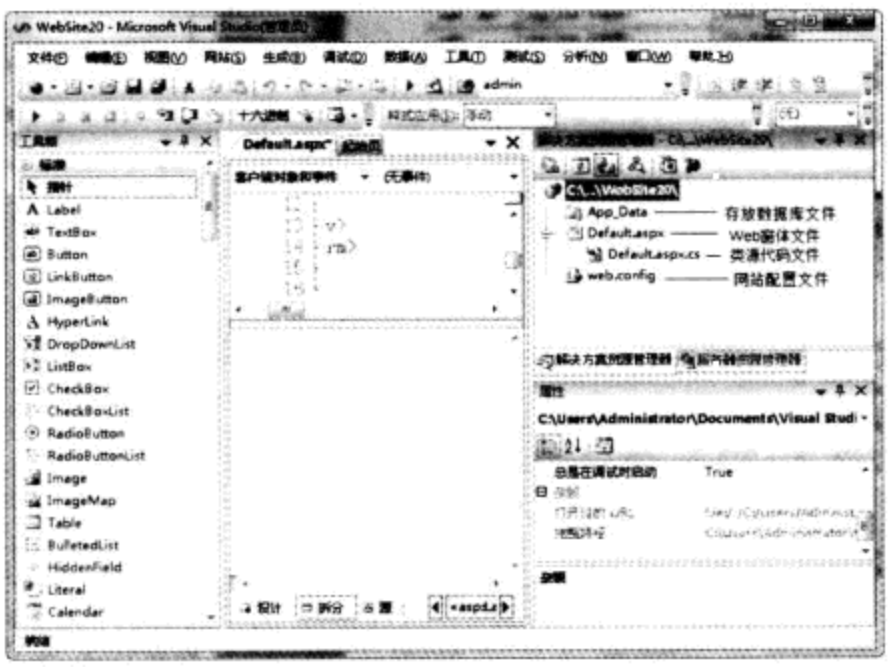


图 3.16　创建网站后根目录下包含的文件

3.2.2　ASP.NET 页面指令

ASP.NET 页面的 HTML 代码中通常包含一些类似于<%@…%>的代码，被称为页面指令。这些指令允许为相应页指定页属性和配置信息，并由 ASP.NET 用作处理页面的指令，但不作为发送到浏览器标记的一部分呈现。当使用页面指令时，虽然标准的做法是将指令包括在文件的开头，但是它们也可以位于.aspx 或.ascx 文件的任何位置。每个指令都可以包含一个或多个特定于该指令的属性（与值成对出现）。

【例 3.2】 通过页面指令设置当前页面加载的主题，主题读者也可以理解为网页的皮肤，代码如下所示：

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

在 Web 页面中常用的页面指令及其作用如表 3.2 所示。

表 3.2　ASP.NET@指令及其作用

指　令	作　用
@Page	定义 ASP.NET 页分析器和编译器使用的页特定（.aspx 文件）属性
@OutputCache	以声明的方式控制 ASP.NET 页或页中包含的用户控件的输出缓存策略
@Register	将别名与命名空间及类名关联起来，以便在自定义服务器控件语法中使用简明的表示法

为了使读者能够更好地理解页面指令，下面详细介绍 ASP.NET 中指令的作用。

1. @Page 指令

@Page 指令允许开发人员为页面指定多个配置选项，并且该指令只能在 Web 窗体页中使用。每个.aspx 文件只能包含一条@Page 指令，@Page 指令语法中各属性的说明如表 3.3 所示。

表 3.3　@Page 指令属性说明

属　性	描　述
Buffer	确定是否启用了 HTTP 响应缓冲。如果启用了页缓冲，则为 true；否则为 false。默认值为 true
CodeBehind	指定包含与页关联的类的已编译文件的名称。该属性不能在运行时使用
CodeFile	指定指向页引用的代码隐藏文件的路径
CodePage	指示用于响应的编码方案的值，该值是一个用做编码方案 ID 的整数
Debug	指示是否应使用调试符号编译该页。如果应使用调试符号编译该页，则为 true；否则为 false。由于此设置影响性能，因此只应在开发期间将此属性设置为 true
EnableTheming	指示是否在页上使用主题。如果使用主题，则为 true；否则为 false。默认值为 true
EnableViewState	指示是否在页请求之间保持视图状态。如果要保持视图状态，则为 true；否则为 false。默认值为 true
Language	指定在对页中的所有内联呈现（<% %> 和 <%= %>）和代码声明块进行编译时使用的语言
Src	指定包含链接到页的代码的源文件的路径。在链接的源文件中，可以选择将页的编程逻辑包含在类中或代码声明块中。可以使用 Src 属性将生成提供程序链接到页
Theme	指定在页上使用的有效主题标识符。如果设置 Theme 属性时没有使用 StyleSheetTheme 属性，则将重写控件上的单独的样式设置，允许用户创建统一而一致的页外观

为了使读者更好地理解如何使用@Page 指令，下面对常用属性进行说明。

（1）CodeFile 属性

该属性指定代码隐藏文件的路径，也就是编写后台代码的文件，扩展名是.cs 的文件。此属性仅对编译的页有效。

【例 3.3】 新添加一个.aspx 页时，设置该页面代码隐藏文件的路径为“Default2.aspx.cs”，代码如下所示：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits= "Default2" %>
```

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

（2）Language 属性

该属性指定编译页面使用的语言，每页只能使用和指定一种语言。

【例 3.4】 指定 ASP.NET 页编译器使用 C#作为页的服务器端代码语言，代码如下所示：

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

2. @OutputCache 指令

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

页输出缓存对于那些不经常更改但需要大量处理才能创建的页特别有用。例如，创建大通信量的网页来显示不需要频繁更新的数据，页输出缓存则可以极大地提高该页的性能。可以分别为每个页配置页缓存，也可以在 Web.config 文件中创建缓存配置文件。利用缓存配置文件，只定义一次缓存设置就可以在多个页中使用这些设置，@OutputCache 指令的属性说明如表 3.4 所示。

表 3.4 @OutputCache 指令的属性说明

属　性	描　述
Duration	设置页或用户控件进行缓存的时间（以秒计）
Location	设置控制资源的输出缓存 HTTP 响应的位置，它的属性值为 OutputCacheLocation 枚举值之一，默认值为 Any
CacheProfile	与该页关联的缓存设置的名称。这是可选属性，默认值为""
NoStore	设置是否阻止敏感信息的二级存储
Shared	设置用户控件输出是否可以由多个页共享。默认值为 false
SqlDependency	设置缓存与数据库之间的对应关系。其值为字符串值，用以辨识网页或控件的输出缓存所依赖的数据库集和数据库表名称组。需要特别注意，SqlCacheDependency 类会监视输出缓存所依赖的数据库中的数据表，因此当更新表中的项时，使用基于表的轮询时将从缓存中移除这些项
VaryByCustom	设置自定义输出缓存要求的任意文本。如果属性值为 browser，则缓存将随浏览器名称和主要版本信息的不同而异。如果为自定义字符串，则必须在应用程序的 Global.asax 文件中重写 GetVaryByCustomString 方法
VaryByHeader	设置分号分隔的 HTTP 标头列表，用于使输出缓存发生变化。将该属性设为多标头时，对于每个指定标头组合，输出缓存都包含一个不同版本的请求文档
VaryByParamr	分号分隔的字符串列表，用于使输出缓存发生变化
VaryByControl	一个分号分隔的字符串列表，用于更改用户控件的输出缓存。这些字符串代表用户控件中声明的 ASP.NET 服务器控件的 ID 属性值

为了使读者更好地理解如何使用@OutputCache 指令，下面对常用属性进行说明。

（1）Duration 属性

该属性指定页或用户控件进行缓存的时间，以秒为单位。

【例 3.5】 设置页或用户控件进行输出缓存的持续时间为 100 秒，代码如下所示：

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

这是必选属性。如果未包含该属性，将出现分析器错误。

（2）VaryByParam 属性

该属性为分号分隔的字符串列表，用于使输出缓存发生变化。在默认情况下，这些字符串与随 GET 方法发送的查询字符串值对应，或与使用 POST 方法发送的参数对应。

在 ASP.NET 页和用户控件上使用 @OutputCache 指令时，需要包含 VaryByParam 属性或 VaryByControl 属性。如果没有包含 VaryByParam 属性或 VaryByControl 属性，则发生分析器错误。如果不希望通过指定参数来改变缓存内容，可将 VaryByParam 属性值设置为 none。如果希望通过所有的参数值改变输出缓存，可将属性设置为星号（*）。

3.2.3　注释 ASPX 文件中代码

服务器端注释（<%--注释内容--%>）允许开发人员在 ASP.NET 应用程序文件的任何部分（除了<script>代码块内部）嵌入代码注释。所谓服务端注释就是为服务器端代码添加注释。服务器端注释元素的开始标记和结束标记之间的任何内容，不管是 ASP.NET 代码还是文本，都不会在服务器上进行处理或呈现在结果页上。注释内容有利于帮助开发者理解代码，或者不希望某些代码运行时，也可以将其注释掉。

【例 3.6】 使用服务器端注释 TextBox 控件，代码如下所示：

```
<%--
   <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
--%>
```

执行后，浏览器上将不显示此文本框。

如果<script>代码块中的代码需要注释，则使用 HTML 代码中的注释（<!--注释-->）。此标记用于告知浏览器忽略该标记中的语句。

【例 3.7】 注释 HTML 中的客户端代码，可以通过下面的代码实现。

```
<script language ="javascript" runat ="server">
    <!--
        注释内容
    -->
</script>
```

3.3　本 章 小 结

Visual Studio 2008 是微软公司推出的集成化互联网开发平台，利用它可以设计、开发以及调试部署 Web 应用程序。本章对 Visual Studio 2008 开发环境的三种设计视图做了详细介绍，并讲解了如何运行程序和配置 IIS，为以后的项目开发打下了坚实的基础。

第4章

C#语言基础

（名师课堂：1小时58分钟）

学习任何一门语言都不能一蹴而就，必须遵循一个客观的原则。从基础学起，有了坚固的基础，再进阶学习有一定难度的技术就会很轻松。学习的过程，就好比一个婴儿的成长过程，不可能还没学习走路，就去参加世界锦标赛进行百米跨栏。所以一门语言的基础是一个人技术实力的根基，也好比一棵大树的树根，掌握的基础知识越牢固，树根扎得越深，面对再大的暴风骤雨也不会畏惧，正是这样的道理。本章将从初学者的角度考虑，将知识与实例配合，轻松了解C#语言基础，快速入门。通过本章的学习，可以掌握以下知识：

- 了解C#中数据类型、运算符的分类
- 掌握C#中数据类型的使用及类型转换
- 掌握常量和变量的使用
- 掌握各种运算符的使用方法

4.1　掌握数据类型

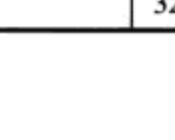
专题讲座：光盘\MR\Video\4\数据类型.exe

➤➤➤视频速递：为了更好地理解数据类型，读者可以参考本视频。

顾名思义，数据类型的表面含义是数据是属于哪种类型，就好像一个人是属于哪个民族一样。数据类型可分为值类型和引用类型。在程序执行时，操作系统都会分配一块内存给应用程序存储相关的数据。内存是有限的资源，若已填满数据，其他要存储的数据必须等某些数据不用、有多余的空间时才能存入，所以要根据数据特性及预估计范围选择一个适合的数据类型。

4.1.1　值类型

所有的值类型都隐式地继承自 System.ValueType 类，该类又继承自 object 类。任何类型都不可能派生自值类型，因此所有的值类型都是隐式密封的。



System.ValueType 本身并不是值类型。相反，它是一个类类型，所有的值类型都自动派生自该类型。

值类型直接存储数据值，它主要包含整数类型、浮点类型以及布尔类型等。值类型在堆栈中进行分配，因此效率很高，使用值类型的主要目的是为了提高性能。值类型具有如下特性。

- ☑ 值类型变量都存储在堆栈中。
- ☑ 访问值类型变量时，一般都是直接访问其实例。
- ☑ 每个值类型变量都有自己的数据副本，因此对一个值类型变量的操作不会影响其他变量。
- ☑ 复制值类型变量时，复制的是变量的值，而不是变量的地址。
- ☑ 值类型变量不能为 null，必须具有一个确定的值。

简单类型是直接列出一系列元素构成的数据类型。C#语言中提供了一组已经定义好的简单类型。所有的简单类型均为.NET Framework 系统类型的别名。例如，int 是 System.Int32 的别名。简单类型可以分为整数类型、布尔类型、实数类型、结构类型和枚举类型等。

（1）整数类型

整数类型变量的值为整数，它的值有一定范围限制。C#中有几种整数类型，如表 4.1 所示。

表 4.1　C#内置的整数类型

类　型	说　明	范　围
sbyte	8 位有符号整数	−128～127
short	16 位有符号整数	−32768～32767
Int	32 位有符号整数	−2147483648～2147483647

续表

类　型	说　明	范　围
long	64 位有符号整数	−9223372036854775808～9223372036854775807
byte	8 位无符号整数	0～255
ushort	16 位无符号整数	0～65535
unit	32 位无符号整数	0～4294967295
ulong	64 位无符号整数	0～18446744073709551615

byte 类型以及 short 类型是范围比较小的整数，如果正整数的范围没有超过 65535，声明为 ushort 类型即可，当然更小的数值直接以 byte 类型进行处理即可。例如，如果要存储班级人数，范围在 30～50 人之间用 byte 即可。一是因为人数一定是“整数”，不可能有 33.5 人的情况，二是范围在 30～50 人之间，若用 int（可存储−2147483648～2147483647 之间的整数），存放班级人数绝对是绰绰有余，但一定会浪费内存空间。

【例 4.1】 整数类型在实际开发中应用非常广泛，例如，在开发会员注册时，需要用户输入年龄、电话号码等。那么，年龄和电话号码就必须是整数类型，运行效果如图 4.1 所示。

学习手册 说明 在输入数据时，要对用户输入的数据进行验证，验证是否是整数类型。相关的内容将在第 9 章中进行讲解。

实例位置：光盘\MR\Instance\4\4.1

【例 4.2】 为了让读者更好地理解整数类型，下面举个简单的例子。例如，第一章 40 页，第二章 50 页，计算总页码是多少？运行结果如图 4.2 所示。

图 4.1　会员注册页面中需要输入整数类型的电话号码

图 4.2　整数类型的应用

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int UltimoMoney = 40;                                    //声明整型变量并赋值
    int NowMoney = 50;
    int result;
    result = UltimoMoney + NowMoney;                         //计算两个整型变量相加
    Response.Write("第一章页码：" + UltimoMoney.ToString() + "<br>");
                                                             //将数据输出到页面中
   Response.Write("第二章页码：" + NowMoney.ToString() + "<br>");
```

```
    Response.Write("总页码：" + result.ToString());
}
```

学习手册 说明 整数类型的应用还有很多，例如，计算仓库货物总数、统计员工工资、统计考勤人数和计算员工保险金额等，在本书以后的几章中整数类型也会经常出现。

（2）布尔类型

布尔类型主要用来表示 true/false 值，一个布尔类型的变量，其值只能是 true 或者 false，不能将其他的值指定给布尔类型变量，布尔类型变量不能与其他类型进行转换。

【例 4.3】 将 927 赋值给布尔类型变量 x，代码如下所示。

```
bool x = 927;
```

这样赋值显然是错误的，编译器会返回错误提示“常量值 927 无法转换为 bool”。布尔类型大多数被应用到流程控制语句当中。例如，循环语句或者 if 语句等。

实例位置：光盘\MR\Instance\4\4.2

【例 4.4】 布尔类型在开发中被经常用到，例如，通过布尔类型可以判断登录用户身份等。本实例创建一个布尔类型的变量 flag，并初始化为 false。根据这个布尔型变量执行相应的 if...else 语句分支，运行结果如图 4.3 所示。

图 4.3　布尔类型变量的应用

实现的代码如下所示：

```
bool  flag=false;                                    //创建并初始化布尔类型变量
if(flag)                                             //如果为 true，则执行 if 分支
{
    Response.Write ("我是中国人！");                  //输出信息
}
else                                                 //如果为 false，则执行 else 分支
{
    Response.Write ("你是美国人！");                  //输出信息
}
```

（3）实数类型

实数在 C#中有单精度（float）和双精度（double）之分，它们的区别在于取值范围和精度不同。表 4.2 列出了这两种数值类型的描述信息。

表 4.2　实数类型及描述

类　型	说　明	范　围
float	精确到 7 位数	1.5×10^{-45}~3.4×10^{38}
double	精确到 15~16 位数	50×10^{-324} ~1.7×10^{308}

对于这两种浮点类型可以根据实际应用情况而定，比如贷款利率，即使是现金卡或地下钱庄，利率也不可能是天文数字，且小数位数顶多两三位，所以用 float 存储利率即可。

C#还专门定义了一种十进制（decimal）类型，它是适合金融和货币计算的 128 位数据

类型。decimal 类型可以表示从 1.0×10^{-28} 到 7.9×10^{28} 的范围内且具有 28 到 29 位有效数字的值。当定义一个变量并进行赋值的时候，使用 m 后缀来表示它是一个 decimal 类型，如下示例代码所示。

```
decimal i =50.20m
```

学习手册 说明 这里如果省略了 m 后缀，那么变量被赋值之前将被编译器当成 double 型。

如果不做任何设置，包含小数点的数值都被认为是 double 类型，例如 9.27，如果没有特别指定，这个数值的类型是 double 类型。如果要将数值以 float 类型来处理，就应该通过强制使用 f 或 F 将其指定为 float 类型。

【例 4.5】 下面的代码用来将数值强制指定为 float 类型。

```
float theMySum = 9.27f;                    //使用 f 强制指定为 float 类型
float theMuSums = 1.12F;                   //使用 F 强制指定为 float 类型
```

如果要将数值强制指定为 double 类型，则需要使用 d 或 D 进行设置。

【例 4.6】 下面的代码用来将数值强制指定为 double 类型。

```
double myDou = 927d;                       //使用 d 强制指定为 double 类型
double mudou = 112D;                       //使用 D 强制指定为 double 类型
```

学习手册 注意 程序中使用 float 类型时，必须在数值的后面跟随 f 或 F，否则编译器会直接将其作为 double 类型处理。

（4）结构类型

利用上面介绍过的简单类型，我们在进行一些常用的数据运算、文字处理似乎已经足够了。但是我们会经常碰到一些更为复杂的数据类型。比如，通信录的记录中可以包含他人的姓名、电话和地址。如果按照简单类型来管理，每一条记录都要存放到三个不同的变量当中，这样工作量很大，也不够直观。有没有更好的办法呢？

在实际生活中，我们经常把一组相关的信息放在一起。把一系列相关的变量组织成为一个单一实体的过程，我们称为生成结构的过程。这个单一实体的类型就叫做结构类型，每一个变量称为结构的成员。前面我们讲的简单类型（int、double 和 bool）实际上都是结构类型，在 C#中，结构类型采用关键字“struct”来声明。

实例位置：光盘\MR\Instance\4\4.3

【例 4.7】 首先应用关键字“struct”创建一个结构类型 Employee，分别存储员工姓名、年龄、部门和积分信息，之后在 Page_Load 事件中声明创建的结构类型 Employee 的变量 E，最后输出指定的员工信息，运行结果如图 4.4 所示。

图 4.4 结构类型应用实例

实现的代码如下所示：

```
public partial class _Default : System.Web.UI.Page
{
```

```
    struct Employee                                    //创建一个结构类型变量
    {
        public string empName;                         //创建字段存储员工姓名
        public uint Age;                               //创建字段存储年龄
        public string Aparterment;                     //创建字段存储部门
        public uint Score;                             //创建字段存储积分
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        Employee E;                                    //声明结构类型 Employee 的变量 E
        E.empName = "红尘倒影";
        E.Age = 24;
        E.Aparterment = "ASP.NET 部门";                 为字段赋值
        E.Score = 98;
        Response.Write(string.Format("{0}员工，今年{1}岁，{2}新员工，绩效成绩{3},优
        秀！", E.empName, E.Age, E.Aparterment, E.Score));
    }
}
```

（5）枚举类型

枚举类型是一种具有命名常量的独特的类型，枚举声明以关键字 enum 开始，然后定义枚举的名称、可访问性、基础类型和成员。其基本语法格式如下所示：

```
访问修辞符 enum 枚举名:基础类型
 {
    枚举成员
 }
```

上述语法中的基础类型必须能够表示该枚举中定义的所有枚举数值。枚举声明可以显式地声明 byte、sbyte、short、ushort、int、uint、long 或 ulong 类型作为对应的基础类型。没有显式地声明基础类型的枚举声明意味着所对应的基础类型是 int（整型）。

枚举成员是该枚举类型的命名常数。任意两个枚举成员不能具有相同的名称。每个枚举成员均具有相关联的常数值。此值的类型就是枚举的基础类型。每个枚举成员的常数值必须在该枚举的基础类型的范围之内。

【例 4.8】 创建一个枚举类型，声明三个枚举成员并分别赋值，代码如下所示：

```
public enum ColorOfCar: uint                           //创建枚举类型
{
    Red=-3,                                            //声明三个枚举成员，并赋值
    Green=-2,
    Blue=-1
}
```

学习手册 说明 运行这样的示例代码会产生编译时错误，原因是-1、-2 和-3 不在基础整型 uint（无符号整型）的范围内。

在枚举类型中声明的第一个枚举成员的默认值为 0。以后的枚举成员值是将前一个枚举成员（按照文本顺序）的值加 1 得到的。这样增加后的值必须在该基础类型可表示的值的范围内；否则，会出现编译时错误。

【例 4.9】 创建一个枚举类型，声明三个枚举成员，并输出枚举成员的值，代码如下所示：

```
public enum ColorOfCar: uint                    //创建枚举类型
{
    Red,                                        //声明三个枚举成员
    Green,
    Blue
}
```

枚举成员默认值分别为：Red 的值为 0；Green 的值为 1；Blue 的值为 2。

图 4.5 枚举类型应用

实例位置：光盘\MR\Instance\4\4.4

【例 4.10】 首先应用关键字“enum”创建一个枚举类型 TimeofDay，分别存储一天中早中晚三个时间段信息；之后声明一个 Text 类，在该类中分对 TimeofDay 枚举类型中的成员进行赋值操作，最后在 Main 函数中调用所创建的 Text 类输出相应的时间段信息，运行效果如图 4.5 所示。

实现的代码如下所示：

```
public enum TimeofDay                           //创建枚举类型
 {
    Morning,                                    //声明三个枚举成员
    Afternoon,
    Evening
 }
 class Test                                     //创建类
 {
    public static string WriteTimeDay(TimeofDay timeofDay)    //类中创建方法
    {
        string result="";
        switch (timeofDay)                      //根据参数执行分支中的代码
        {
            case TimeofDay.Morning:
                result="Good Morning!";
                break;
            case TimeofDay.Afternoon:
                result="Good Afternoon!";
                break;
            case TimeofDay.Evening:
                result="Good Evening!";
                break;
        }
        return result;
    }
 }
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Test.WriteTimeDay(TimeofDay.Morning));
    Response.Write(Test.WriteTimeDay(TimeofDay.Afternoon));

    Response.Write(Test.WriteTimeDay(TimeofDay.Evening));
}
```

根据参数执行分支中的代码

调用类中的方法，并将结果输出到页面

4.1.2　引用类型

引用类型是构建 C#应用程序的主要类型，引用类型的变量又称为对象，可存储对实际数据的引用。C#支持两个预定义的引用类型 object 和 string，其说明如表 4.3 所示。

表 4.3　C#预定义的引用类型及说明

类　型	说　明
object	object 类型在.NET Framework 中是 Object 的别名。在 C#的统一类型系统中，所有类型（预定义类型、用户定义类型、引用类型和值类型）都是直接或间接从 Object 继承的
string	string 类型表示零或更多 Unicode 字符组成的序列

学习手册 说明　尽管 string 是引用类型，但如果用到了相等运算符（==和!=），则表示比较 string 对象（而不是引用）的值。

在应用程序执行的过程中，引用类型以 new 创建对象实例，所有被称为“类”的都是引用类型，主要包括类、接口、数组和委托等。下面通过一个实例来演示如何使用引用类型。

实例位置：光盘\MR\Instance\4\4.5

【例 4.11】 在平时的开发中，类是经常被用到的。所以，引用类型也变得尤为重要，读者要掌握创建引用类型对象的方法。例如，本实例中创建一个类 C，在此类中建立一个字段 Value，并初始化为 0，然后在程序的其他位置通过 new 关键字创建对此类的引用类型变量，最后输出，如图 4.6 所示。

图 4.6　如何使用引用类型

实现的代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    int v1 = 0;                                    //声明一个 int 类型的变量 v1，并初始化为 0
    int v2 = v1;                                   //声明一个 int 类型的变量 v2，并将 v1 赋值给 v2
    v2 = 927;                                      //重新将变量 v2 赋值为 927
    C r1 = new C();                                //使用 new 关键字创建引用对象
    C r2 = r1;                                     //使 r1 等于 r2
    r2.Value = 112;                                //设置变量 r2 的 Value 值
    Response.Write("Values:"+ v1+","+ v2);          //输出变量 v1 和 v2
    Response.Write("Refs:"+r1.Value+","+r2.Value);//输出引用类型对象的 Value 值
}
class C                                            //创建一个类 C
{
    public int Value = 0;                          //声明一个公共 int 类型的变量 Value
}
```

学习手册 注意　关于类我们在以后的章节中会进行详细介绍，读者可以参考本书第 6 章的内容。

4.2 认识常量和变量

专题讲座：光盘\MR\Video\4\常量和变量.exe

➤➤➤视频速递：常量和变量是开发中经常用到的，本视频会实际演示其用法。

常量又叫常数，它主要用来存储在程序运行过程中值不改变的数据。变量也就是可以变化的数据，每个变量都具有一个类型。麻雀虽小却五脏俱全，常量和变量的作用不可忽视。通过本节的学习，读者能够了解什么是常量和变量，以及如何使用常量和变量。

4.2.1 什么是常量

常量又叫常数，它主要用来存储在程序运行过程中值不改变的数据。常量也有数据类型，C#语言中，常量的数据类型有多种，主要有 sbyte、byte、short、ushort、int、uint、long、ulong、char、float、double、decimal、bool 和 string 等。

4.2.2 定义并使用常量

常量通过 const 关键字声明，常量必须在声明时初始化。例如，通过 const 关键字声明一个常量，代码如下所示：

```
class Calendar1
{
    public const int months = 12;
}
```

上述代码中，常量 months 将始终为 12，不能更改，即使是该类自身也不能更改它。常量必须属于整型（sbyte、byte、short、ushort、int、uint、long、ulong、char、float、double、decimal、bool 或 string）枚举或对 null 的引用。

【例 4.12】 可以同时声明多个相同类型的常量，代码如所示：

```
class Calendar2
{
    const int months = 12, weeks = 52, days = 365;
}
```

实例位置：光盘\MR\Instance\4\4.6

【例 4.13】 在开发中，一些频繁用到的且值不改变的经常被定义为常量。例如，在代码中经常将算术中的 π 定义为常量，因为这个值是不改变的。在本实例中计算圆的周长，运行结果如图 4.7 所示。

图 4.7 计算圆的周长

实现的代码如下所示。

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

```
    const double PI = 3.1415926;                    //声明常量，存储π的值
    int r = 112;                                    //声明圆的半径
    double perimeter;                               //存储圆周长
    perimeter = 2 * PI * r;                         //调用常量和半径计算圆周长
    Response.Write("圆的周长是："+perimeter);        //输出周长
}
```

常量可标记为 public、private、protected 或 internal，这些访问修饰符定义用户访问该常量的权限。尽管常量不能使用 static 关键字声明，但可以像访问静态字段一样访问常量。未包含在定义常量类中的表达式必须使用类名、一个句点和常量名来访问该常量。

若要创建在运行时初始化的常量值，请使用 readonly 关键字。

编程准则：尽可能多地使用 const

const 关键字可以防止开发程序时错误的产生。例如，对于一些不需要改变的对象，使用 const 关键字将其定义为常量，可以防止开发人员不小心修改对象的值，产生意想不到的结果。将函数返回的指针类型定义为 const 指针，可以向其他人员表明不应该修改函数的返回值，如果不进行类型转换，显式地进行修改，则会出现编译错误。如果在定义函数时，不希望在函数体中修改参数值，应使用 const 关键字将参数定义为常量参数，防止用户修改参数值。在定义类的方法时，例如 getXXX()等形式的方法用于获取类的信息时，应将方法定义为 const 方法，阻止用户在方法中修改成员变量的值，因为方法的作用是获取信息，而不是修改信息。总之，应尽可能多地使用 const 关键字（Use const whenever you need）。

4.2.3　什么是变量

变量也就是可以变化的数据，每个变量都具有一个类型，这个类型确定了该变量中可以存储哪些值。C#是一种类型安全的语言，C#编译器保证了存储在变量中的值总是具有合适的类型。变量的值可以通过进行赋值或++和--运算符来更改。

4.2.4　变量的声明和赋值

变量的作用非常重要，它代表一个特定的数据项或值。与常量不同，变量可以反复赋值。它就像一个盒子，存储着各种不同的数据。C#中规定，使用变量前必须声明。变量的声明同时规定了变量的类型和变量的名字。变量的声明采用如下规则。

```
<type><name>;
```

使用未声明的变量是不会通过程序编译的。C#中并不要求在声明变量的同时初始化变量，即为变量赋初值，但为变量赋初值通常是一个好习惯。C#中可以声明的变量类型并不仅限于C#预先定义的那些。因为 C#有自定义类型的功能，开发人员可以根据自己的需要

建立各种特定的数据类型以方便存储复杂的数据。

声明变量非常简单，下面的例子便声明了一个整型变量 a。

```
int a;
```

【例 4.14】 在声明变量的同时为变量赋初值，如下所示。

```
bool b=true;
```

上述代码在声明布尔型变量的同时将其初始值设置为真（true）。

【例 4.15】 在同一行中同时声明多个变量，如下所示。

```
int b,c;
int d=1,e=2;
```

每一个变量都有自己的名称，但 C#规定不能用任意的字符作为变量名。变量的命名规则有以下几条，希望读者能够牢记。

☑ 变量的第一个字符必须是字母、下画线“_”或“@”。

☑ 后面的字符可以是字母、下画线或数字。

☑ 变量的名称不能使用 C#的关键字。

C#的关键字如表 4.4 所示。

表 4.4 C#关键字

关 键 字			
abstract	event	new	Struct
as	explicit	null	Switch
base	extern	object	This
bool	false	operator	Throw
break	finally	out	True
byte	fixed	override	Try
case	float	params	Typeof
catch	for	private	Uint
char	foreach	protected	Ulong
checked	goto	public	Unchecked
class	if	readonly	Unsafe
const	implicit	ref	Ushort
continue	in	return	Using
decimal	int	sbyte	Virtual
default	interface	sealed	Volatile
delegate	internal	short	Void
do	is	sizeof	While
double	lock	Stackalloc	
Else	lock	static	

另外需要注意的是 C#区分大小写，使用变量时必须按照正确的大小写引用。如下面

的变量虽然仅有大小写的区别，但在 C#中代表不同的变量：mydata、Mydata、MYDATA。

编程信条：从变量命名开始培养写程序的好习惯

如果写程序的习惯不好，造成程序的可读性差，日后要维护或调试时，别人看不懂就算了，即使是你自己，隔个三五天再来看自己编写的程序可能就已经看不懂，更不要说修改或维护。英文有句话说：“Eat your dog food”，大体翻译过来就应该是“自食其果”的意思，所以培养一个好的编程习惯就从为变量取个好名字开始吧。基本上为变量选择名称时，最好使用有意义的名称，而不要为了少打几个字用简写或缩写。比如，与其用 hAddr 还不如用 homeAddress 更加明白、正确。

在声明了变量之后就可以在后面的程序中使用它们。无论在声明变量时是否对其初始化，之后都可以对其进行再次赋值。

【例 4.16】 声明两个变量，并对其进行赋值。

```
Int a=0;                                    //定义整型变量 a，并为其赋初值 0
Double b;                                   //定义双精度型变量 B，未赋初值
a=256;                                      //为整型变量 a 赋值为 256
b=12.28;                                    //为双精度型变量度 b 赋值为 12.28
```

实例位置：光盘\MR\Instance\4\4.7

【例 4.17】 常量和变量在开发中都是不可缺少的，在编程中至关重要。例如，声明一个变量并初始化为 0，然后通过一个循环改变其值，并输出到页面，运行结果如图 4.8 所示。

图 4.8　改变变量的值

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int flag = 0;                                   //声明一个变量并初始化
    for (int i = 0; i < 5; i++)                     //通过 for 循环改变变量的值
    {
        Response.Write(flag.ToString());            //输出改变后变量的值
        flag++;                                     //变量加 1
    }
}
```

学习手册 说明 关于 for 语句，我们将在第 5 章进行介绍，请读者参看第 5 章的内容。

4.2.5　变量的作用域

变量的作用域就是可以访问该变量的代码区域。一般情况下，可以通过以下规则确定变量的作用域。

☑ 只要字段所属的类在某个作用域内，其字段也在该作用域内。

☑ 局部变量存在于表示声明该变量的块语句或方法结束的封闭花括号之前的作用域内。

☑ 在 for、while 或类似语句中声明的局部变量存在与该循环体内。

【例 4.18】 使用 for 循环将 0～20 的数字显示出来。然后在 for 语句中声明变量 i，此时 i 就是局部变量，其作用域只限于 for 循环体内，代码如下所示。

```
static void Main(string[] args)
{
    //调用 for 语句循环输出数字
    for (int i = 0; i <= 20; i++)                          //for 循环内的局部变量 i
    {
        Response.Write (i.ToString());  } 变量作用域
    }
}
```

4.3 如何实现类型转换

专题讲座：光盘\MR\Video\4\类型转换.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 ASP.NET 中的类型转换。

类型转换就是将一种类型转换成另一种类型，转换可以是隐式转换或者显式转换，本节将详细介绍这两种转换方式。

4.3.1 隐式类型转换

隐式类型转换就是不需要声明就能进行的转换。进行隐式类型转换时，编译器不需要进行检查就能安全地进行转换。

【例 4.19】 整型转换为货币类型和单精度类型，代码如下所示：

```
int i=123;
decimal money=i;                                           //数字转货币
float f=i;                                                 //整型转单精度
```

下面是隐式类型转换，如表 4.5 所示。

表 4.5 隐式类型转换表

源 类 型	目 标 类 型
sbyte	short、int、long、float、double、decimal
byte	short、ushort、int、uint、long、ulong、float、double 或 decimal
short	int、long、float、double 或 decimal
ushort	int、uint、long、ulong、float、double 或 decimal
int	long、float、double 或 decimal
uint	long、ulong、float、double 或 decimal
char	ushort、int、uint、long、ulong、float、double 或 decimal

续表

源 类 型	目 标 类 型
Float	double
ulong	float、double 或 decimal
long	float、double 或 decimal

从 int、uint、long 或 ulong 到 float，以及从 long 或 ulong 到 double 的转换可能导致精度损失，但不会影响它的数量级。其他的隐式转换不会丢失任何信息。

【例 4.20】 将 int 类型的值隐式转换成 long 类型，代码如下所示。

```
int i = 123;                    //声明一个整型变量 i 并初始化为 123
double b = i;                   //隐式转换成 double 类型
```

4.3.2　显式类型转换

显示类型转换也可以称为强制类型转换，它需要在代码中明确地声明要转换的类型。如果在不存在隐式转换的类型之间进行转换，就需要使用显式类型转换。表 4.6 列出了需要进行显式类型转换的数据类型。

表 4.6　显式类型转换表

源 类 型	目 标 类 型
sbyte	byte、ushort、uint、ulong 或 char
byte	sbyte 和 char
short	sbyte、byte、ushort、uint、ulong 或 char
ushort	sbyte、byte、short 或 char
int	sbyte、byte、short、ushort、uint、ulong 或 char
uint	sbyte、byte、short、ushort、int 或 char
char	sbyte、byte 或 short
float	sbyte、byte、short、ushort、int、uint、long、ulong、char 或 decimal
ulong	sbyte、byte、short、ushort、int、uint、long 或 char
Long	sbyte、byte、short、ushort、int、uint、ulong 或 char
double	sbyte、byte、short、ushort、int、uint、ulong、long、char 或 decimal
decimal	sbyte、byte、short、ushort、int、uint、ulong、long、char 或 double

在 C#语言中，若要将某个表达式显式转换为特定数据类型，可使用显式强制转换调用转换运算符，将数据从一种类型转换为另一种类型。

【例 4.21】 使用显式转换将一个单精度值转换为一个整数值。

```
float f = 123.45;
int i = (int)f;
```

或

```
float f = 123.45
int i =Convert.ToInt32(f);
```

学习手册 注意 在进行数据类型的转换编程时，最好显式地给出转换的类型。这样既方便程序的阅读和维护，也不易导致错误。

4.4 使用 C#中运算符

专题讲座：光盘\MR\Video\4\C#中运算符.exe

➤➤➤视频速递：本视频主要对 C#中的运算符进行详细讲解，读者能够更直观地了解其用法。

C#中的运算符是用来对变量、常量或数据进行计算的符号，它主要用来指挥计算机进行什么样的操作。可以将运算符理解为交通警察的命令，用来指挥行人或车辆等不同的运动实体（运算数据），最后达到一定的目的。C#中的运算符包括算术运算符、赋值运算符和关系运算符等，本节将对这几种运算符进行详细介绍。

4.4.1 算术运算符

C#中有 6 种算术运算符，如表 4.7 所示。

表 4.7 算术运算符

加法	+	除法	/
减法	–	求余（取模）	%
乘法	*	取负	–

1. 加法运算符

加法运算符（+）通过两个数相加来执行标准的加法运算。

实例位置：光盘\MR\Instance\4\4.8

【例 4.22】 在实际开发中，使用加法运算符可以进行标准的加法运算，例如，开发网页计算机器、计算工资等。在本实例中，声明两个整型变量 M1 和 M2，并将 M1 赋值为 927，然后使 M2 的值为 M1 与 M1 相加之后的值，运行结果如图 4.9 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int M1 = 927;                                    //声明整型变量 M1，并赋值为 927
    int M2;                                          //声明整型变量 M2
    M2 = M1 + M1;                                    //M2 的值为 M1 与 M1 相加之后的值
    Response.Write (M2.ToString());
}
```

2. 减法运算符

减法运算符（－）通过从一个表达式中减去另外一个表达式的值来执行标准的减法运算。

实例位置：光盘\MR\Instance\4\4.9

【例 4.23】 在开发库存管理系统时，就是用减法运算符计算出库后的库存数量。本实例首先声明两个 decimal 类型的变量 R1 和 R2，并分别赋值为 1112.82 和 9270.81，然后再声明一个 decimal 类型的变量 R3，使其值为 R2 减去 R1 之后得到的值，运行结果如图 4.10 所示。

图 4.9　加法运算符计算两数之和

图 4.10　减法运算符计算两数之差

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    decimal R1 = (decimal)1112.82;                    //声明整型变量 R1，并赋值为 1112.82
    decimal R2 = (decimal)9270.81;                    //声明整型变量 R2，并赋值为 9270.81
    decimal R3;                                       //声明整型变量 R3
    R3 = R2 - R1;                                     //R3 的值为 R2 减去 R1 得到的值
    Response.Write (R3.ToString());
}
```

3. 乘法运算符

乘法运算符（*）将两个表达式进行乘法运算并返回它们的乘积。

实例位置：光盘\MR\Instance\4\4.10

【例 4.24】 在开发人力资源管理系统时，通过乘法运算符计算员工月工资。本实例首先声明两个整型变量 int1 和 int2，并分别赋值为 10 和 20。再声明一个整型变量 Mul，使其值为 int1 和 int2 的乘积，运行结果如图 4.11 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int int1 = 10;                                    //声明整型变量 int1，并赋值为 10
    int int2 = 20;                                    //声明整型变量 int2，并赋值为 20
    int  Mul;                                         //声明整型变量 Mul
    Mul = int1 * int2;                                //使 Mul 的值为 int1 和 int2 的乘积
    Response.Write (Mul.ToString());
}
```

4. 除法运算符

除法运算符（/）执行算术除运算，它用除数表达式除以被除数表达式而得到商。

被除数表达式的结构不能为 0，否则将会出现异常。

实例位置：光盘\MR\Instance\4\4.11

【例 4.25】 在开发人力资源系统时，作者就是通过除法运算符计算公司员工的人均收入的。在本实例中首先声明两个整型变量 cs1 和 cs2，并分别赋值为 45 和 5。再声明一个整型变量 result，使其值为 cs1 除以 cs2 得到的值，运行结果如图 4.12 所示。

图 4.11 乘法运算符计算两数之积

图 4.12 除法运算符计算两数之商

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int cs1 = 45;                          //声明整型变量 cs1，并赋值为 45
    int cs2 = 5;                           //声明整型变量 cs2，并赋值为 5
    int result;                            //声明整型变量 result
    result = cs1 /cs2;                     //使 result 的值为 cs1 除以 cs2 得到的值
    Response.Write (result.ToString());
}
```

5. 求余运算符

求余运算符（%）返回除数与被除数相除之后的余数，通常用这个运算符来创建余数在特定范围内的等式。

图 4.13 求余运算符判断是否为偶数

实例位置：光盘\MR\Instance\4\4.12

【例 4.26】 求余运算符在实际开发中应用还是比较广泛的，例如，通过求余运算符判断某个数字是奇数还是偶数。本实例通过求余运算符挑选出 0~30 之内的所有偶数，并输出显示，运行结果如图 4.13 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    for (int i = 0; i <= 30; i++)          //for 语句循环 0~30 的所有数字
    {
        if (i % 2 == 0)                    //通过求余运算符判断数字与 2 的余数是否为 0
        {
            Response.Write(i.ToString()+",");   //如果余数是 0 则说明是偶数
        }
    }
}
```

知识解读：什么是一元运算符和二元运算符

在 C#中，接受一个操作数的运算符称为“一元”运算符，例如增量运算符（++）或 new。接受两个操作数的运算符称为“二元”运算符，例如算术运算符 +、-、*、/。条件运算符 ?: 接受三个操作数，是 C# 中唯一的三元运算符。

4.4.2　赋值运算符

赋值运算符有= 、 += 、-= 、*= 、/= 、 %= 、&= 、|= 、^= 、<<= 、>>=、??等。C#语言中可以对变量进行连续赋值，这时赋值操作符是右关联的，这意味着从右向左操作符被分组，如 x=y=z 等价于 x=(y=z)。如果操作符两侧的操作数类型不一致，应先进行类型转换。给变量赋值的式子称为赋值表达式，赋值表达式的值用于给变量赋的值。常用的赋值操作符的运算对象、运算法则及运算结果如表 4.8 所示。

表 4.8　赋值操作符的运算规则

名　称	操 作 符	运 算 规 则	意　义	运 算 对 象	运 算 结 果
赋值	=	将表达式赋值给变量	将右边的值给左边	任意类型	任意类型
加赋值	+=	a+=b	a =a+b	数值型（整型、实数型等）	数值型（整型、实数型等）
减赋值	-=	a-=b	a =a-b		
除赋值	/=	a/=b	a =a/b		
乘赋值	*=	a*=b	a =a*b		
模赋值	%=	a%=b	a =a%b	整型	整型

在 C#语言中，所有赋值和自反赋值操作符的优先级都一样，比所有其他操作符的优先级都低，是优先级最低的操作符。

实例位置：光盘\MR\Instance\4\4.13

【例 4.27】 赋值运算符应用非常广泛。本实例实现的是当输入用户名之后，通过赋值运算符“=”将输入的用户名赋值给字符串变量，并输出，运行结果如图 4.14 所示。

图 4.14　赋值运算符为变量赋值

实现的代码如下所示：

```
protected void btnlogin_Click(object sender, EventArgs e)
{
    string name = string.Empty;                    //声明一个字符串变量
    name = txtname.Text.Trim();                    //通过赋值运算符将输入的用户名赋值给变量
    Response.Write("登录用户："+name);              //输出字符串变量存储的值
}
```

4.4.3 关系运算符

关系运算符可以实现对两个值的比较运算，并且在比较运算之后会返回一个代表运算结果的布尔值。常见的关系运算符及说明如表 4.9 所示。

表 4.9 关系运算符及说明

关系运算符	说　明	关系运算符	说　明
==	等于	!=	不等于
>	大于	>=	大于等于
<	小于	<=	小于等于

下面对这几种关系运算符进行详细讲解。

1. 相等运算符

要查看两个表达式是否相等，可以使用相等运算符（==）。相等运算符对整型、浮点型和枚举类型数据的操作是一样的，它只简单地比较两个表达式，并返回一个布尔结果。

实例位置：光盘\MR\Instance\4\4.14

【例 4.28】 通过相等运算符判断登录用户是否为合法用户。主要思路是通过相等运算符将输入的用户名和密码与合法的进行比对，如果相等，则说明合法。运行结果如图 4.15 所示。

实现的代码如下所示：

```
protected void btnlogin_Click(object sender, EventArgs e)
{
    string name = txtname.Text;                          //获取输入的用户名
    string pwd = txtpwd.Text;                            //获取输入的密码
    if (name == "mr" && pwd == "mrsoft")                 //通过相等运算符判断是否合法
    {
        Response.Write("登录成功");                       //如果合法输出提示信息
    }
    else                                                 //如果不合法
    {
        Response.Write("登录失败");                       //输出错误提示
    }
}
```

2. 不等运算符

不等运算符（!=）是与相等运算符相反的运算符，有两种格式的不等运算符可以应用到表达式中，一种是普通的不等运算符（!=），另外一种是相等运算符的否定!(a= =b)。通常，这两种格式可以计算出相同的值。

实例位置：光盘\MR\Instance\4\4.15

【例 4.29】 在会员注册时，登录用户名一定不能为空，那么，在后台代码中如何判断其是否为空呢？此时，不等运算符就派上用场了，只有当登录用户名不为空的情况才允许注册，运行结果如图 4.16 所示。

图 4.15　判断登录用户是否合法

图 4.16　不等运算符判断是否输入用户名

实现的代码如下所示：

```
protected void btnreg_Click(object sender, EventArgs e)
{
    if (txtname.Text.Length != 0)  //如果文本框中文本的长度不等于 0 说明输入了用户名
    {
                                   //此处是用户注册的相关代码
    }
    else                           //否则，说明没有输入用户名
    {
        txtname.Text="请输入用户名";
    }
}
```

3. 小于运算符

如果要比较一个值是否小于另外一个值，可以使用小于运算符（<）。当左边的表达式的值小于右边表达式的值时，结果是真，否则结果是假。

实例位置：光盘\MR\Instance\4\4.16

【例 4.30】 会员注册时，输入的年龄不能太小。例如，年龄不能为 18 岁以下，因为 18 岁才符合成年人的年龄，所以对输入的年龄要进行判断。本实例就是使用小于运算符判断输入的年龄是否小于 18，运行结果如图 4.17 所示。

实现的代码如下所示：

```
protected void btnreg_Click(object sender, EventArgs e)
{
    if (int.Parse(txtage.Text) < 18)                    //判断输入的年龄是否小于 18
    {
        Response.Write("年龄输入有误");                  //如果小于 18 则输出提示信息
    }
}
```

4. 大于运算符

如果比较一个值是否大于另外一个值，可以使用大于运算符（>）。当左边的表达式的值大于右边的表达式的值时，结果是真，否则结果是假。

实例位置：光盘\MR\Instance\4\4.17

【例 4.31】 会员注册时，输入的年龄不仅不能太小，同时也不能无限大，所以要对其进行限制。例如，年龄不能大于 80，当然年龄的上限读者可以自己根据实际情况设置，本实例就是使用大于运算符判断输入的年龄是否大于 80，运行结果如图 4.18 所示。

图 4.17 小于运算符判断输入的年龄是否小于 18

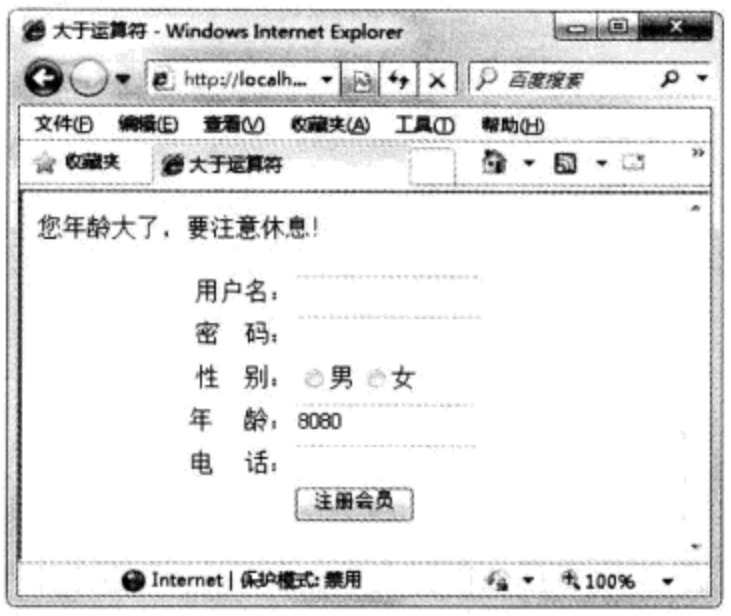


图 4.18 大于运算符判断输入的年龄是否大于 80

实现的代码如下所示：

```
protected void btnreg_Click(object sender, EventArgs e)
{
    if (int.Parse(txtage.Text) > 80)                    //判断输入的年龄是否大于 80
    {
        Response.Write("您年龄大了，要注意休息！");      //如果大于 80 输出提示信息
    }
}
```

4.4.4 情景应用 1：开发简单计算器

实例位置：光盘\MR\Instance\4\4.18

【例 4.32】 本实例通过算术运算符开发一个简单的加法运算器，输入两个数字之后，单击“计算”按钮，会将两个数字之和显示出来，如图 4.19 所示。

实现的代码如下所示：

```
protected void btnresult_Click(object sender, EventArgs e)
{
    int num1 = int.Parse(txtnum1.Text.Trim());          //获取输入的第一个数字
    int num2 = int.Parse(txtnum2.Text.Trim());          //获取输入的第二个数字
    txtresult.Text = (num1 + num2).ToString();          //通过加法运算符计算两数之和
}
```

4.4.5 情景应用 2：开发简单登录

实例位置：光盘\MR\Instance\4\4.19

【例 4.33】 会员注册和登录对于经常上网的朋友来说并不陌生，本实例根据关系运算符开发一个简单的登录功能，如图 4.20 所示。

图 4.19　简单的加法计算器

图 4.20　简单的登录功能

实现的代码如下所示：

```
protected void btnlogin_Click(object sender, EventArgs e)
{
    if (txtname.Text == "mr" && txtpwd.Text == "mrsoft")
                                          //判断输入的用户名和密码是否正确
    {
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('登录成功');",
        true);
    }
}
```

4.5　实 战 练 习

4.5.1　输入出生年份判断生肖属相

➤➤➤题目描述

平时上网时，有很多网站都可以在线根据输入的出生年份查找相应的生肖属相。那么，这个功能是如何开发的呢？其实，开发起来并不复杂。读者可以尝试自己开发该功能，作者开发了一个小程序，能够实现该功能，仅供读者参考，效果如图 4.21 所示。

➤➤➤技术指导

开发本实例时，首先声明字符串数组变量，用来存储生肖属相（存储顺序为猴鸡狗猪鼠牛虎兔龙蛇马羊）。输入的出生年份除以 12 取模，余数是几则在字符串数组中找到对应的属相。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/4.5.1**

4.5.2 求最大公约数

➤➤➤题目描述

最大公约数即能被两个整数整除的最大整数，例如，24 与 15 两个数的最大公约数为 3，本实例实现在两个文本框中输入数字，然后单击“求最大公约数”按钮，输出计算出来的最大公约数，效果如图 4.22 所示。

图 4.21 输入出生年份判断生肖属相

图 4.22 求最大公约数

➤➤➤技术指导

求最大公约数可以用求余法实现。即用两个数中较大的数除以较小的数求余，然后使用除数除以余数求余，直到余数为 0 时，之前的除数也就是两个数的最大公约数。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/4.5.2**

4.5.3 求最小公倍数

图 4.23 求最小公倍数

➤➤➤题目描述

本实例通过多种运算符相结合，实现计算最小公倍数。运行程序，在两个文本框中输入数字，然后单击“求最小公倍数”按钮，输出计算出来的最小公倍数，效果如图 4.23 所示。

➤➤➤技术指导

最小公倍数可以通过两个数的乘积除以这两个数的最大公约数得到。例如，12 与 9 的最大公约数为 3，则这两个数最小公倍数的计算方法为 12*9/3，计算的结果就是这两个数的最小公倍数。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/4.5.3**

4.6　本 章 小 结

本章首先对 C#语言进行了简单介绍，接着对 C#语言中的基本语法进行了讲解，其中包括数据类型、常量、变量、数据类型转换、运算符等内容。学习完本章后，读者应该对 C#基础语法有一个清晰的认识，并能在实际中简单应用。

第5章

掌握字符与字符串

（名师课堂：55分钟）

.NET Framework 类库中提供了强大的字符和字符串处理功能，其中 Char 类是 C#提供的字符类型，String 类是 C#提供的字符串类型，开发人员可以通过这两个类提供的方法对字符和字符串进行各种操作；另外，本章还讲解了 StringBuilder 类与 String 类的区别，通过本章的学习，读者可以掌握以下知识：

- 了解字符和字符串的区别
- 掌握 Char 类的概念及应用
- 掌握字符串的基本操作
- 掌握 StringBuilder 类的用法
- 了解 StringBuilder 类与 String 类的区别

5.1　Char 字符类应用

专题讲座：光盘\MR\Video\5\ Char 字符类.exe

➤➤➤视频速递：详细讲解字符类的概念及应用。

本节重点讲解 Char 类的概念及应用，通过本节的学习，读者完全能够掌握什么是字符，以及在开发中如何使用字符，希望读者能够认真学习本节的内容。

5.1.1　Char 类的概述

Char 在 C#中表示一个 Unicode 字符，正是这些 Unicode 字符构成了字符串，Unicode 字符是目前计算机中通用的字符编码，它为针对不同语言中的每个字符设定了统一的二进制编码，用于满足跨语言、跨平台的文本转换及处理的要求。其实字符离我们并不遥远，比如，计算机按键上的单个字母、单个数字都是字符。Char 的定义非常简单，可以通过下面的代码定义字符。

```
char ch1='a';
char ch2='B';
```

Char 只定义一个 Unicode 字符。

5.1.2　Char 类的应用

Char 类为开发人员提供了许多方法，可以通过这些方法灵活地操控字符。Char 类的常用方法及说明如表 5.1 所示。

表 5.1　Char 类的常用方法及说明

方　法	说　明
IsControl	指示指定的 Unicode 字符是否属于控制字符类别
IsDigit	指示某个 Unicode 字符是否属于十进制数字类别
IsLetter	指示某个 Unicode 字符是否属于字母类别
IsLower	指示某个 Unicode 字符是否属于小写字母类别
IsNumber	指示某个 Unicode 字符是否属于数字类别
IsPunctuation	指示某个 Unicode 字符是否属于标点符号类别
IsSeparator	指示某个 Unicode 字符是否属于分隔符类别
IsSymbol	指示某个 Unicode 字符是否属于符号字符类别
IsUpper	指示某个 Unicode 字符是否属于大写字母类别
IsWhiteSpace	指示某个 Unicode 字符是否属于空白类别
Parse	将指定字符串的值转换为它的等效 Unicode 字符
ToLower	将 Unicode 字符的值转换为它的小写等效项
ToString	将此实例的值转换为其等效的字符串表示

续表

方　法	说　明
ToUpper	将 Unicode 字符的值转换为它的大写等效项
TryParse	将指定字符串的值转换为它的等效 Unicode 字符

可以看到 Char 提供了非常多的实用方法，其中以 Is 和 To 开头的比较重要。以 Is 开头的方法大多是判断 Unicode 字符是否为某个类别，以 To 开头的方法主要是转换为其他 Unicode 字符。

图 5.1　Char 类中常用方法的使用

实例位置：光盘\MR\ Instance\5\5.1

【例 5.1】 创建一个网站，在网站的后台代码中声明几个字符变量并赋值。然后，使用 Char 类中的多个方法来判断字符变量的相关内容。例如，判断字符是否为字母、是否是大写字母等，运行结果如图 5.1 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    char a = 'a';
    char b = '8';
    char c = 'L';
    char d = '.';
    char f = '|';
    char g = ' ';
    //使用 IsLetter 方法判断 a 是否为字母
    Response.Write(string.Format("IsLetter 方法判断 a 是否为字母：{0}<br>", Char.
    IsLetter(a)));
    //使用 IsDigit 方法判断 b 是否为数字
    Response.Write(string.Format("IsDigit 方法判断 b 是否为数字：{0}<br>", Char.
    IsDigit(b)));
    //使用 IsLower 方法判断 a 是否为小写字母
    Response.Write(string.Format("IsLower 方法判断 a 是否为小写字母:{0}<br>", Char.
    IsLower(a)));
    //使用 IsUpper 方法判断 c 是否为大写字母
    Response.Write(string.Format("IsUpper 方法判断 c 是否为大写字母:{0}<br>", Char.
    IsUpper(c)));
    //使用 IsPunctuation 方法判断 d 是否为标点符号
    Response.Write(string.Format("IsPunctuation方法判断d是否为标点符号:{0}<br>", Char.
    I sPunctuation(d)));
    //使用 IsSeparator 方法判断 e 是否为分隔符
     Response.Write(string.Format("IsSeparator 方法判断 f 是否为分隔符:{0}<br>", Char.
     IsSeparator(f)));
     //使用 IsWhiteSpace 方法判断 f 是否为空白
     Response.Write(string.Format("IsWhiteSpace 方法判断 g 是否为空白：{0}", Char.
     IsWhiteSpace(g)));
 }
```

声明字符变量

5.1.3　转义字符

C#采用字符“\”作为转义字符。例如，定义一个字符，而这个字符是单引号，如果不使用转义字符，则会产生错误。

转义字符就相当于一个电源变换器，电源变换器就是通过一定的手段获得所需的电源形式，如交流变为直流、高电压变为低电压、低频变为高频等。转义字符是将字符转换成另一种操作形式，或是将无法一起使用的字符进行组合。

转义符\（单个反斜杠）只针对后面紧跟着的单个字符进行操作。

【例 5.2】 不使用转义字符，直接将单引号赋值给字符变量，会产生错误，代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    char M='';                                          //声明一个字符变量，值为单引号
}
```

程序的运行结果如图 5.2 所示。

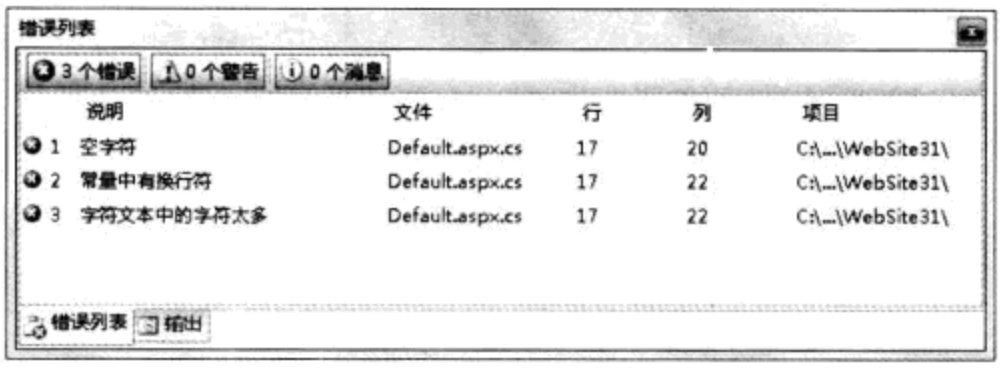


图 5.2　错误提示

【例 5.3】 为了避免此类错误，应该使用转义字符，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    char a='\'';                                        //使用转义字符定义字符的值为单引号
}
```

另外还有其他转义字符，如表 5.2 所示。

表 5.2　转义字符及说明

转 义 字 符	说　　明
\n	回车换行
\t	横向跳到下一制表位置
\v	竖向跳格
\b	退格
\r	回车
\f	换页
\\	反斜线符
\'	单引号符
\ddd	1~3 位八进制数所代表的字符
\xhh	1~2 位十六进制数所代表的字符

学习手册 技巧 通过以下两种方式，可以让转义字符不发生转义。

图 5.3 转义字符的使用

① 通过@符实现，如：@“\n”。

② 通过逐字指定字符串字面值（两个反斜杠）实现，如：“\\n”。

实例位置：光盘\MR\Instance\5\5.2

【例 5.4】 创建一个网站，在 Page_Load 事件中编写代码。首先，不使用转义字符，正常输出四句话。然后，使用转义字符为每句话加上引号，并输出到页面，运行结果如图 5.3 所示。

实现的代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("正常输出：<br>");
    Response.Write("十里平湖霜满天<br>");
    Response.Write("寸寸青丝愁华年<br>");
    Response.Write("对月形单影相互<br>");
    Response.Write("直线鸳鸯不羡仙<br><br>");
    Response.Write("使用转义字符后：<br>");
    Response.Write("\"十里平湖霜满天\"<br>");
    Response.Write("\"寸寸青丝愁华年\"<br>");
    Response.Write("\"对月形单影相互\"<br>");
    Response.Write("\"直线鸳鸯不羡仙\"");
}
```

没有使用转义字符

使用转义字符输出带引号的字符串

5.2 使用静态字符串类 String

专题讲座：光盘\MR\Video\5\ String 字符串类.exe

➤➤➤视频速递：详细讲解字符串类的概念及多种应用。

在 ASP.NET 中提供了 String 类来对字符串进行操作。这些操作在很大程度上方便了开发人员，而且编写程序的灵活性也得到大大增强。下面介绍常用的字符串处理。

5.2.1 字符串的概念

字符串型表示一系列字符，是使用 string 关键字声明的一个字符数组。.NET Framework 中表示字符串的关键字为 string。例如，“我们是中国人”，这就可以成为一个字符串。甚至还可以将数字定义为字符串。例如，下边的代码就是将数字定义为字符串。

```
string s_shj="0112";
```

所以说，字符串的变化是多种多样的，读者在学习中不要拘泥于字符串的概念，要根据实际情况灵活运用。

5.2.2　比较字符串

比较字符串是指按照字典排序的规则，判断两个字符串的大小，在英文字典中，在前面的单词小于在后面的单词。String 类中，常用的比较字符串的方法包括 Compare、CompareTo 以及 Equals 等，下面对它们进行详细介绍。

1．Compare 方法

Compare 方法是 String 类的静态方法，用于全面比较两个字符串对象。

语法：

```
Int compare (string strA, string strB)
```

参数 strA 和 strB 为待比较的两个字符串。

Compare 方法的返回值如表 5.3 所示。

表 5.3　Compare 方法的返回值

参 数 条 件	返　回　值
strA 小于 strB	负整数
strA 大于 strB	正整数
strA 等于 strB	0

实例位置：光盘\MR\Instance\5\5.3

【例 5.5】 声明两个字符串变量并初始化，然后通过 Compare 方法比较两个字符串，得到如图 5.4 所示的结果。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    System.String Str1 = "您好";              //声明字符串并赋值
    System.String Str2 = "您好吗";            //声明字符串并赋值
    //字符串比较
    Response.Write(String.Compare(Str1, Str2)+"<br>");
                                              //如果第一个字符串小于第二个则返回负整数
    Response.Write(String.Compare(Str1, Str1) + "<br>");
                                              //如果两个字符串相同则返回 0
    Response.Write(String.Compare(Str2, Str1) + "<br>");
                                              //如果第一个字符串大于第二个则返回正整数
}
```

学习手册 注意　CompareTo 方法将当前字符串对象与另一个字符串对象作比较，其作用与 Compare 类似，返回值也是相同的。

2．Equals 方法

Equals 方法用于判断两个字符串是否相同。

语法：

```
Public bool Equals(string)
```

如果两个字符串相等，Equals()返回值为 True；否则返回 False。

实例位置：光盘\MR\Instance\5\5.4

【例 5.6】 在开发会员登录时，除了使用相等运算符判断用户名和密码是否正确外，还可以使用 Equals 方法进行判断，运行结果如图 5.5 所示。

图 5.4 通过 Compare 方法比较字符串

图 5.5 Equals 方法的使用

实现的代码如下所示：

```
protected void btnlogin_Click(object sender, EventArgs e)
{
    if (txtname.Text.Equals("mr") && txtpwd.Text.Equals("mrsoft"))
                                                        //判断用户名和密码是否正确
    {
            //如果输入的与指定的用户名和密码相同，说明用户名和密码合法，则弹出提示信息
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('登录成功');",
        true);
    }
    else                                                //如果不相等
    {
            //则弹出错误提示信息
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('登录失败');",
        true);
    }
}
```

学习手册 注意 Equals 方法区分大小写，ClientScript.RegisterStartupScript 方法用于在当前网页中弹出提示窗口。

5.2.3 格式化字符串

String.Format 方法能够将任何数值、枚举以及日期和时间等基本数据类型表示为指定格式的字符串，即格式化。

语法：

```
String Format(String,Object);
```

将指定的 String 中的格式项替换为指定的 Object 实例的值的文本等效项。

实例位置：光盘\MR\Instance\5\5.5

【例 5.7】 在数据绑定时，因为有时候会将数据格式化成自己想要的格式，所以经常格式化字符串。所谓的数据绑定就是将数据库中的数据提取出来显示在数据绑定控件上，

关于数据绑定我们在以后的章节中会有所介绍。本实例主要是将指定的数字格式化为 Currency 类型，将当前时间格式化成 ShortDate 类型，运行结果如图 5.6 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    //格式化为 Currency 类型
    string str1 = String.Format("(C) Currency:{0:C}\n", -123.45678f);
    //格式化为 ShortDate 类型
    string str2 = String.Format("(d) Short date: {0:d}\n", DateTime.Now);
    Response.Write(str1+"<br>");
    Response.Write(str2);
}
```

5.2.4　截取字符串

Substring 方法可以从指定字符串中截取子串。

语法：

```
String.Substring(Int32,Int32);
```

参数 1 表示子串的起始位置；参数 2 表示子字符串的长度。

实例位置：光盘\MR\Instance\5\5.6

【例 5.8】 截取字符串在开发中是非常有用的，很多时候都需要截取字符串。举个最简单的例子，在页面中显示数据时，如果数据过长可能会破坏页面布局，此时就需要对数据进行截取，长度超过指定长度时就用“...”代替。运行结果如图 5.7 所示。

图 5.6　格式化字符串

图 5.7　截取字符串

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    string str = "十里平湖霜满天，寸寸青丝愁华年。对月形单影相互，只羡鸳鸯不羡仙。";
    string substr = str.Substring(0, 15)+"...";                    //截取字符串
    Response.Write(substr);                                        //输出截取后的字符串
}
```

5.2.5　分割字符串

Split 方法可以把一个字符串按照某个分隔符分割成一系列小的字符串。

语法：

```
String[] Split(Char[]);
```

其中，参数为分隔字符串的分隔符数组。

实例位置：光盘\MR\Instance\5\5.7

【例 5.9】 在实际开发中，经常使用 Split 方法对用户权限、用户注册时选择的爱好等进行分割，从而将其绑定到相应的控件中。本实例首先创建一个字符串赋值为“Hello·World!”，通过 Split 方法将其在“.”和“！”处进行分割，分割之后返回字符串数组，然后将数组中的字符串输出，运行结果如图 5.8 所示。

图 5.8 分割字符串

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    string str = "Hello.World!";                                    //声明字符串并赋值
    string[] split = str.Split(new Char[] { '.', '!' });            //根据指定的字符进行分割
    foreach (string s in split)                                     //遍历分割后的字符数组
    {
        if (s.Trim() != "")
            Response.Write(s+"<br>");                               //输出数组中的内容
    }
}
```

5.2.6 替换字符串

Replace 方法可以替换掉一个字符串中的某些特定字符或者子串。

语法：

```
String Replace(String String);
```

图 5.9 替换字符串

其中，第一个参数为待替换的子串；第二个参数为替换后的新子串。

实例位置：光盘\MR\Instance\5\5.8

【例 5.10】 在实际开发中，经常使用 Replace 方法过滤脏字。其原理就是将字符串中的脏字替换成某个字符。例如，本实例中首先创建一个字符串并赋值为“他爷爷的年纪大了，身体很不好”，但是，很可能这个字符串中的某些语言涉及脏字。所以，对其进行过滤。运行结果如图 5.9 所示。

实现的代码如下所示：

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

```
    string str = "他爷爷的年纪大了，身体很不好";          //声明一个可能含有脏字的字符串
    string repstr = str.Replace("他爷爷的","*****");  //替换指定的字符串
    Response.Write(repstr);                          //输出替换后的字符串
}
```

5.3　使用可变字符串类 StringBuilder

专题讲座：光盘\MR\Video\5\StringBuilder 字符串类.exe

➤➤➤视频速递：讲解 StringBuilder 类的概念、应用及与 String 类的区别。

本章重点介绍什么是 StringBuilder 类，并且通过实例演示如何使用 StringBuilder 类。StringBuilder 类在开发时应用还是比较广泛的，可能读者对 StringBuilder 类和 String 类的区别感到模糊，本节也会详细介绍两者的区别。

5.3.1　StringBuilder 类的概述

StringBuilder 类亦称为可变字符串，它位于 System.Text 命名空间下，之所以称为可变字符串，是因为该类的对象在通过追加、移除、替换或插入字符串后不会生成新的字符串对象，即字符串变量始终指向同一个对象。

StringBuilder 类有 6 种不同的构造方法，这里列出程序开发中最常用 4 种，其语法格式如下。

```
public StringBuilder()
public StringBuilder(string value)
public StringBuilder(int capacity)
public StringBuilder(string value,int capacity)
```

其中，value 表示 StringBuilder 对象引用的字符串。capacity 设定 StringBuilder 对象的初始大小，即 StringBuilder 对象的容量。

【例 5.11】 创建一个 StringBuilder 对象，其初始引用的字符串为“欢乐斗地主！”，代码如下。

```
StringBuilder sb = new StringBuilder("欢乐斗地主！");
```

5.3.2　StringBuilder 类的应用

StringBuilder 类存在于 System.Text 命名空间中，如果要创建 StringBuilder 对象，首先必须引用此命名空间。StringBuilder 类中常用的方法及说明如表 5.4 所示。

表 5.4　StringBuilder 类中常用的方法及说明

方　法	说　明
Append	将文本或字符串追加到指定对象的末尾
AppendFormat	自定义变量的格式并将这些值追加到 StringBuilder 对象的末尾
Insert	将字符串或对象添加到当前 StringBuilder 对象中的指定位置

续表

方　法	说　明
Remove	从当前 StringBuilder 对象中移除指定数量的字符
Replace	用另一个指定的字符来替换 StringBuilder 对象内的字符

图 5.10　StringBuilder 类中常见方法的应用

下面通过实例来演示如何使用 StringBuilder 类中的这 5 种方法。

实例位置：光盘\MR\Instance\5\5.9

【例 5.12】 首先实例化一个 StringBuilder 类型的对象 sb，其初始值为“出塞”，初始大小为 100，然后使用 StringBuilder 类的 Append、AppendFormat、Insert、Remove 和 Replace 方法操作 StringBuilder 对象，运行结果如图 5.10 所示。

实现的代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder("无题", 100); //创建 StringBuilder 类的对象
    sb.Insert(2, "(作者：红尘倒影)");                    //在索引为 2 的位置添加字符串
    sb.Append("长春<br>十里平湖霜满天" + "<br>");        //添加字符串(诗句)
    sb.Append("寸寸青丝愁华年" + "<br>");                //添加字符串(诗句)
    sb.Append("对月形单影相互" + "<br>");                //添加字符串(诗句)
    sb.Append("只羡鸳鸯不羡仙" + "<br>");                //添加字符串(诗句)
    sb.Remove(11, 2);                          //移除从索引 11 开始的 2 个字符(即“长春”)
    sb.Replace("(作者：红尘倒影)", "(作者：无名氏)");    //替换掉错误的作者名称
    sb.AppendFormat("{0:D}", DateTime.Now.ToString());      //添加当前时间
    Response.Write(sb);
}
```

5.3.3　StringBuilder 类与 String 类的区别

String 类和 StringBuilder 类都用来处理字符串，它们之间有着较多相似的常规用法，并且这二者之间可以相互转换，这导致许多人在程序设计中不严格区分这二者，甚至认为这二者基本相同，是可以通用的。但实际情况不然，而是这二者之间的内在差别非常大。下面将对这两个类进行详细区分。

- ☑ String 类型表示 Unicode 字符的字符串，该类型的字符串对象是只读的，也就是说，一旦创建了某个字符串对象，那么该字符串对象就不能够修改。表面看来能够修改字符串的所有方法实际上并没有修改原有的字符串，而是生成另外一个全新的字符串对象。
- ☑ StringBuilder 类型表示值为可变字符序列的类似字符串的对象。之所以说值是可变的，是因为可以对 StringBuilder 对象进行追加、移除、替换或插入字符来对其进行修改，这点与 String 类型恰恰相反。大多数修改 StringBuilder 实例的方法都返回对同一实例的引用。由于返回的是对实例的引用，因此可以调用该引

用的方法或属性。如果想要编写将连续操作依次连接起来的单个语句，这将很方便。

【例 5.13】 下面通过一个例子来具体看一下 StringBuilder 类与 String 类的区别，首先看下面一段代码：

```
string str="QQ"+"农场";
stringbuilder sb=new stringbuilder();
sb.append("QQ");
sb.append("农场");
```

上面的代码分别使用 string 类和 StringBuilder 类连接字符串，但这两种方法在内存中的操作是不同的，第一种方法在内存中操作时，有 3 个 string（分别为"QQ"、"农场"、"QQ 农场"）变量；而第二种方法在内存中操作时只有一个（"QQ 农场"）变量，所以它们的性能是完全不同的。

当程序中需要对某个字符串进行大量操作时，应该考虑应用 StringBuilder 类处理该字符串，其设计目的就是针对大量 string 操作的一种改进办法，避免产生太多的临时对象。当程序中只是对某个字符串进行一次或几次操作时，采用 string 类即可。

5.4 实 战 练 习

5.4.1 巧截字符串的数字

➤➤➤题目描述

本实例主要实现的是在包含数字的字符串中剥离数字，并输出到页面，这个功能是如何开发的呢？其实，开发起来并不复杂。读者可以尝试自己开发该功能，作者开发了一个小程序，能够实现该功能，仅供读者参考，效果如图 5.11 所示。

图 5.11　巧截字符串的数字

➤➤➤技术指导

截取字符串中的数字时，可以先使用 CharEnumerator 对象的 MoveNext 方法循环访问字符串中的每个字符，并将字符用 System.Text.Encoding 类中 ASCII 编码方式的 GetBytes 方法进行编码，然后判断经过编码之后的字符的 ASCII 码值是否介于 48 和 57 之间，如果是，则说明是数字。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/5.4.1**

5.4.2 在字符串中查找指定的字符或字符串

➤➤➤题目描述

我们在数据库中可以检索某个数据，在数组中也可以搜索数据。那么，是否在字符串中也可以查找某个字符或字符串呢？答案是肯定的。本实例就要求在字符串中查找指定的字符或字符串，效果如图 5.12 所示。

➤➤➤技术指导

开发本实例时，首先声明 string 类型变量，为其随意设置初始字符串及用来进行查询的数据。然后通过 IndexOf 方法在字符串中查找指定的字符或字符串，将查询结果输出到页面。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/ loveASP.NET/5.4.2**

5.4.3 颠倒字符串

➤➤➤题目描述

颠倒输出字符串时，可以先将要输出的字符串保存到一个 char 类型的数组中，然后使用 Array 类的 Reverse 方法将数组中的元素颠倒输出，效果如图 5.13 所示。

图 5.12 在字符串中查找指定的字符或字符串

图 5.13 颠倒字符串

➤➤➤技术指导

反转字符串可以考虑使用 Array 类的 Reverse 方法，它可以反转一个一维数组中所有元素的顺序，该方法的参数为一个一维数组，这样可以考虑把要反转的字符串转换为字符型的一维数组，这需要使用 String 类的 ToCharArray 方法，该方法可以把指定字符串转换为一维字符数组。参考代码如下所示：

```
char[] charStr = strRead.ToCharArray();    //把字符串 strRead 转换为一维字符数组
Array.Reverse(charStr);                    //反转一维字符数组中的所有元素
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/5.4.3**

5.5　本 章 小 结

本章介绍了用于文本处理的 Char、String 和 StringBuilder 类。在介绍这 3 个类时，结合了大量的实例进行讲解，使读者能够从实例中掌握每种类的用法。在阅读本章时，读者要重点掌握 String 类中处理字符串的一些方法，这些方法在开发程序时会经常用到。StringBuilder 类允许使用同一个字符串对象进行字符串的维护操作，这样，可以在操作字符串数据的过程中提高效率，尤其是处理大量文字数据时。另外，还要注意 String 类与 StringBuilder 类的本质区别。

PDG

第 6 章

面向对象程序设计

（名师课堂：1 小时 39 分钟）

面向对象程序设计是在面向过程程序设计的基础上发展而来的，它将数据和对数据的操作看做是一个不可分割的整体，力求将现实问题简单化，因为这样不仅符合人们的思维习惯，同时也可以提高软件的开发效率，并方便后期的维护。所有的.NET 程序语言都是面向对象的程序语言，而且在.NET 里所有的东西都是对象，ASP.NET 也是如此。因此，要成为一个 ASP.NET 的程序开发人员，一定要具备面向对象程序设计的基本概念。通过本章的学习，读者可以掌握以下知识：

- 了解什么是面向对象编程
- 了解面向对象编程的特点
- 了解类与类成员的概念及用法
- 了解并掌握类的封装
- 了解并掌握类的继承
- 了解并掌握类的多态

6.1　理解面向对象编程

专题讲座：光盘\MR\Video\6\面向对象编程.exe

➤➤➤视频速递：了解什么是面向对象编程及其特点。

面向对象思想来源于对现实世界的认知。现实世界缤纷复杂，事物种类繁多，难于认识和理解。但是聪明的人们学会了把这些错综复杂的事物进行分类，从而使世界变得井井有条。比如我们由各式各样的汽车抽象出汽车的概念、由形形色色的鸟抽象出鸟的概念、由五彩斑斓的鲜花抽象出花的概念等。本节将讲述什么是面向对象编程及其特点。

6.1.1　面向对象编程概述

面向对象编程（Object-Oriented Programming）简称 OOP 技术，是开发应用程序的一种新方法、新思想，也是软件开发的主流。图 6.1 以简单示意图的方式，展示了面向对象开发的一个标准过程。

在面向对象世界里都会涉及两个非常重要的概念——“类”与“对象”，类与对象是面向对象观念的核心。类（Class）在面向对象观念中可视为对象的“模板”、“蓝图”。只要是同一个类所产生的对象，一定都会有相同的特性。而将抽象化的类具体化，就成了对象（Object）。以日常生活中具体事物为例，我们都比较喜欢狗，那么就可以为狗建立一个类，具体到如松狮犬、萨摩犬和西施犬等就是狗这个类的对象。类与对象之间的关系可形象地如图 6.2 所示。

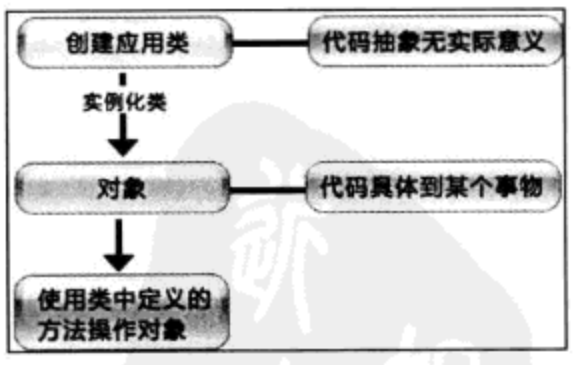


图 6.1　面向对象开发的过程

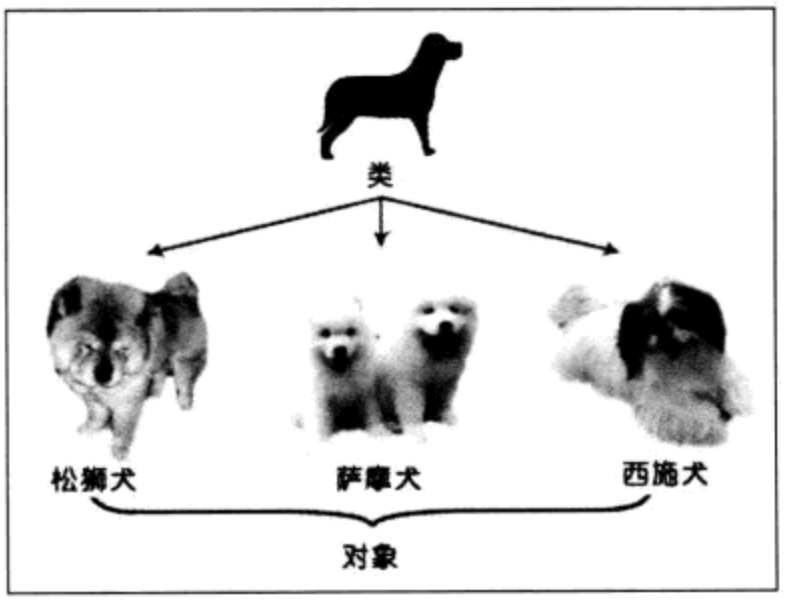


图 6.2　类和对象的关系

学习手册 说明　另外，狗这个类还具有一些特征，如颜色、高度和重量；还有一些行为，如跑和咬等。我们称这些特征为属性，称这些行为为方法。

求知问道：万物皆对象?

简单理解类与对象的关系：类是抽象的概念，对象是真实的个体。我们可以说米老鼠的体重是 0.5kg，而不能说鼠类的体重是 0.5kg；可以说刘翔在奥运会 110 米跨栏比赛中夺冠，而不能说人类在跨栏比赛中夺冠。现实生活中到处充实着对象，小到一粒沙，大到中国的万里长城、埃及的金字塔，乃至一种语言、一种方法都可以称为对象（即万物皆对象）。

本书所使用的语言为 C#，C#就是一门完全面向对象的开发语言，它建立在成熟的面向对象基础之上，充分体现了面向对象技术的优势。

6.1.2 面向对象编程的特点

面向对象之所以流行，是因为其达到了软件工程的三个主要目标：重用性、灵活性和扩展性。面向对象编程的基本思想是使用对象、类、继承、封装、多态等来进行程序开发，这里就体现了面向对象编程语言的三个重要特点：继承、封装和多态。

下面对面向对象编程三大支柱进行简单的介绍。

（1）封装性：类是属性和方法的集合，为了实现某项功能而定义类后，开发人员并不需要了解类体内每句代码的具体含义，只需通过对象来调用类内某个属性或方法即可实现某项功能，这就是类的封装性。

比如，使用电脑时，并不需要将电脑拆开了解每个部件的具体用处，用户只需按下主机箱上的 Power 按钮就可以启动电脑，在键盘上敲打就可以将文字输入到电脑中，但对于电脑内部的构造，用户可能根本不了解，这就是封装的具体表现。效果如图 6.3 所示。

（2）继承性：通过继承可以创建子类（派生类）和父类之间的层次关系，子类（派生类）可以从其父类中继承属性和方法，通过这种关系模型可以简化类的操作。假如已经定义了 A 类，接下来准备定义 B 类，而 B 类中有很多属性和方法与 A 类相同，那么就可以使 B 类继承于 A 类，这样就无须再在 B 类中定义 A 类已有的属性和方法，从而可以在很大程度上提高程序的开发效率。

例如可以将水果看成一个父类，那么水果类具有颜色属性，然后再定义一个苹果类，在定义苹果类时完全可以不定义苹果类的颜色属性，通过如下继承关系完全可以使苹果类具有颜色属性，效果如图 6.4 所示。

（3）多态性：类的多态性指不同的类进行同一操作可以有不同的行为。例如，定义一个火车类和一个汽车类，火车和汽车都可以移动，说明两者在这方面可以进行相同的操作，然而，火车和汽车移动的行为是截然不同的，因为火车必须在铁轨上行驶，而汽车在公路上行驶，这就是类多态性的形象比喻，如图 6.5 所示。

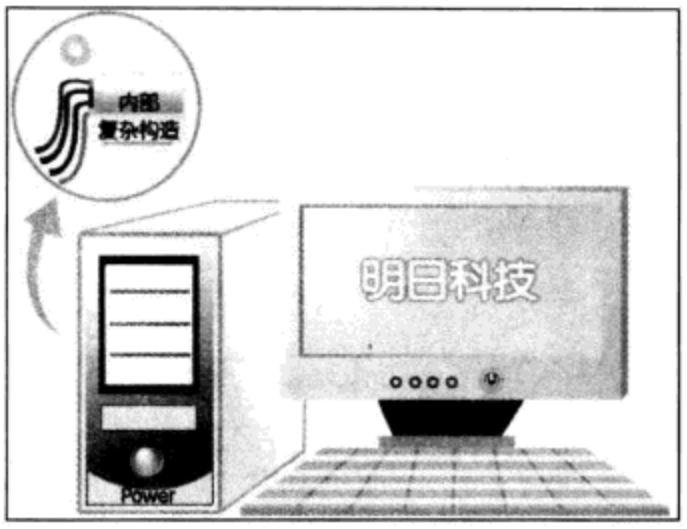


图 6.3　封装特性效果示意图

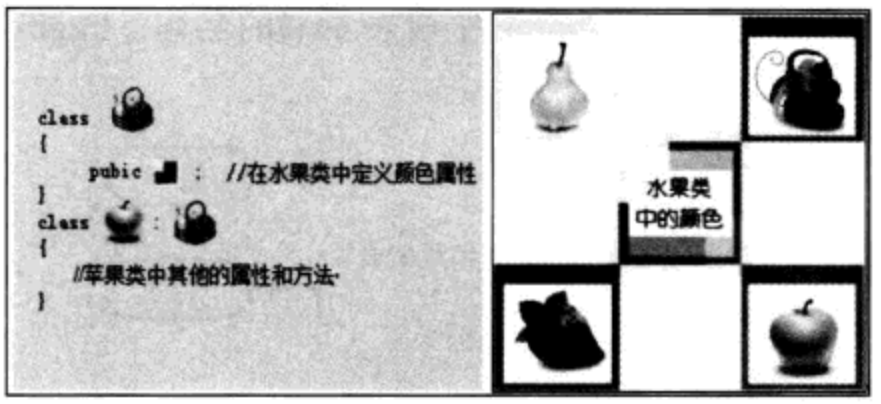


图 6.4　继承特性效果示意图

图 6.5　多态性在生活中的体现

6.2　面向对象中类与类成员

专题讲座：光盘\MR\Video\6\类与类成员.exe

➤➤➤视频速递：通过对本视频的学习可以掌握如何创建类及其成员。

对于一个.NET 或 ASP.NET 程序开发人员来说，了解类的设计方式，对于如何“妥善”、“正确”地使用它，将会十分有帮助，且可避免许多不必要的误解及错误。本节将对类和类成员进行详细讲解。

6.2.1　类的概念

类是对象在面向对象编程语言中的反映，是相同对象的集合。类描述了一系列在概念上有相同含义的对象，并为这些对象统一定义了编程语言上的属性和方法。例如，车是一个类，自行车、汽车、火车也是类，但是自行车、汽车、火车都属于车这个类的子类（派生类），因为它们有共同的特点——都是交通工具、都有轮子、都可以运输；而汽车有颜色、车轮、车门和发动机，这是它和自行车、火车不同的地方，是汽车类自己的属性，也是所有汽车共同的属性，所以汽车也是一个类；而具体到某个汽车就是一个对象了，例如：车牌照为吉 A7777****的黑色奥迪车。

简而言之，类是 C#中功能最为强大的数据类型，像结构一样，类也定义了数据类型的数据和行为，然后，程序开发人员可以创建作为此类的实例的对象。C#中，类是使用 class

关键字来声明的，如图 6.6 所示。

图 6.6 中定义了一个类，类名为 Book，类体为两个变量 Name 和 Page，在类中定义的变量叫做字段，Name 字段表示书的名称；Page 字段表示书的页码。也可以在类中定义对象行为，如图 6.7 所示。

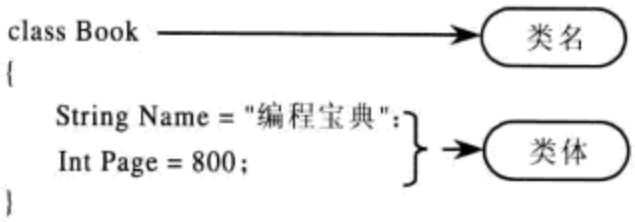


图 6.6　类的定义

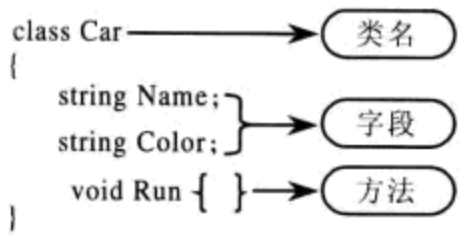


图 6.7　类的字段和方法

6.2.2　创建类

（1）打开 Visual Studio 2008 开发环境，新建一个网站，并将其命名为 MRText，然后右键单击网站名称，在弹出的快捷菜单中选择“添加新项”选项，如图 6.8 所示。

（2）单击“添加新项”之后，便可打开如图 6.9 所示的“添加新项”窗口，单击“模板”中的“类”图标，在“名称”栏中输入这个类文件的名称。

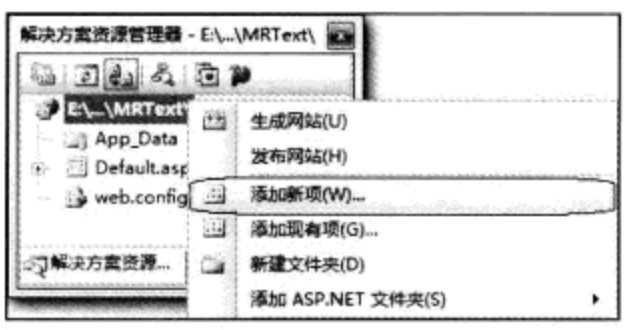


图 6.8 “添加新项”选项

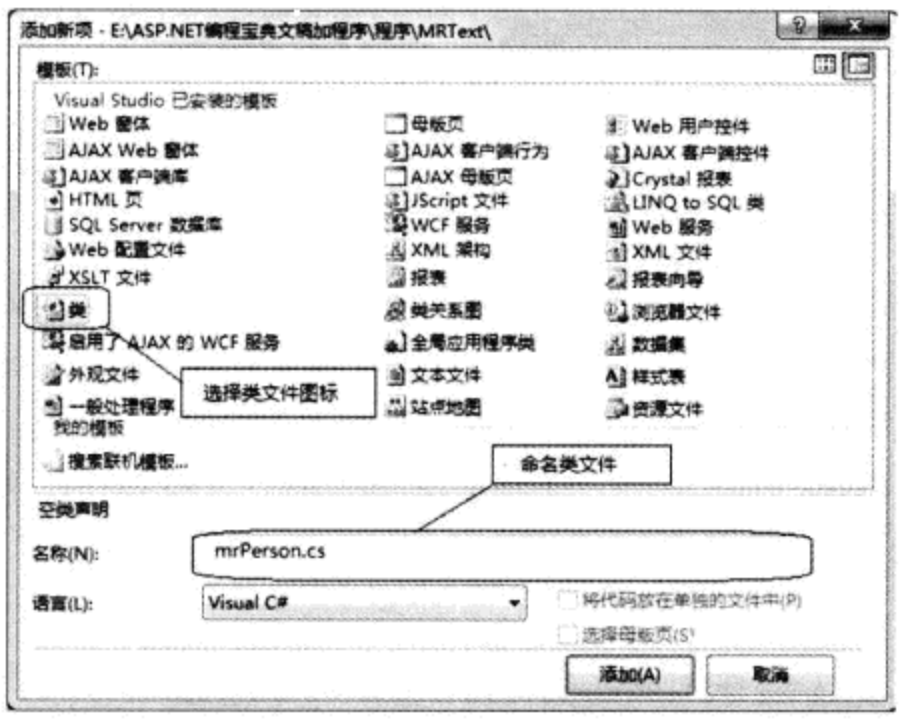


图 6.9　添加类

（3）完成以上步骤后，单击“添加”按钮，将会弹出如图 6.10 所示的一个对话框向用户提问是否要将类放在 App_Code 文件夹内，这里请单击“是”按钮。

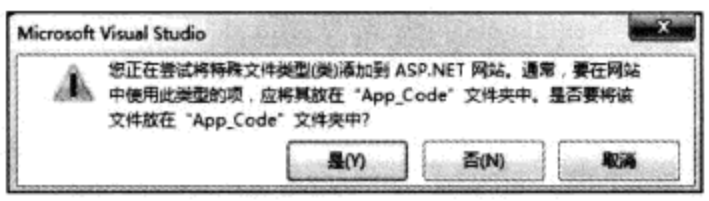


图 6.10　是否要将类放在 App_Code 文件夹内

学习手册 说明 App_Code 是 ASP.NET 特别为程序代码文件而设的一个目录，如其名所述就是用来存放程序代码（Code）的，所有外界对 App_Code 目录的浏览要求都会被 ASP.NET 挡下来，保护程序代码不被外界轻意地浏览到。

（4）在“解决方案资源管理器”窗口应可以看到 mrPerson.cs 被放在 App_Code 目录下，

如图 6.11 所示。

打开 mrPerson.cs，其内容分为两大部分，上部分主要为默认网页所引用的命名空间，下部分为要编写的类的程序代码，如图 6.12 所示。

图 6.11　类文件应存放在 App_Code 文件夹中

图 6.12　mrPerson.cs 类文件

学习手册 说明　mrPerson.cs 类文件中的上部分（第 1 行~第 11 行）为所引用的命名空间 System 等，主要是方便程序简洁地调用 C#内建的一些功能；第 16 行~第 24 行是对创建的 mrPerson 类的声明。

6.2.3　定义类

在创建的 mrPerson.cs 类文件中可以看到使用 class 关键字来对名为 mrPerson 的类进行定义，并且该类为公有的访问类（public 修饰符），具体代码如下所示：

```
public class mrPerson
{
    public mrPerson()
    {
            //
            //TODO: 在此处添加构造函数逻辑
            //
    }
   //定义类成员
}
```

学习手册 提示　在上述代码中，public mrPerson()程序块是创建 mrPerson 类时自动生成的，它可自动帮助开发人员显示出默认的构造函数，关于构造函数后面章节将会给予说明。

另外，从上述代码中可以总结出定义类的语法如下：

```
类修饰符 class 类名
{
    //定义类成员
```

```
}
```

其中，类修饰符可以使用 public 访问修饰符（访问修饰符用来定义类或类成员的可访问性），这样被定义的类就成为公共类。除了 public 之外，还有 4 个常用的访问修饰符，具体说明如下。

- ☑ protected：受保护访问修饰符，只能被本类及子类访问。
- ☑ Internal：仅限于内部才能访问。
- ☑ protected internal：内部保护访问修饰符，只能被项目内的所有类和这个类的子类访问。
- ☑ private：私有访问修饰符，只能被本类访问。

除了使用以上常用访问修饰符，还可以使用 abstract 和 sealed。其中 abstract 修饰符用来指定类是抽象的，抽象类的特点是不能被实例化，只能被继承。sealed 关键字用来指定类是密封的，密封类的特点是不被继承。

如果一个类中包含敏感数据，有些人可以访问，有些人不能访问，如果不对这些数据的访问加以限制，后果将会非常严重。所以在编写程序时，要对类的成员使用不同的访问修饰符，从而定义它们的访问权限。

类建立完成后，接下来便是设计类的内容，在类中可以包含以下几种类成员。

- ☑ 常量：代表与类相关联的常量值。
- ☑ 字段：在类中定义的变量。
- ☑ 方法：执行类中的计算和其他操作。
- ☑ 属性：提供对类的字段进行安全访问。
- ☑ 事件：用来说明发生了什么事情。
- ☑ 索引器：允许像使用数组那样访问类的数据。
- ☑ 运算符：定义类的特有运算。
- ☑ 构造函数和析构函数：分别用来对类的实例进行初始化和销毁。

【例 6.1】 实现类及其的数据成员的设定。

```
public class mrPerson
{
    //定义类的数据成员
    public string Name;
    public string Address;
    private DateTime _birthday;
}
```

上述代码中定义了三个类的数据成员：Name、Address 和_birthday，分别用来记录人的姓名、地址及生日，从示例代码中可以看到，类的数据成员（Data Member）其实就是在类里声明一个变量，用来存储数据。

定义类的时候所使用的访问修饰符及其使用范围，同样也对类的数据成员有效。

6.2.4　实例化类对象

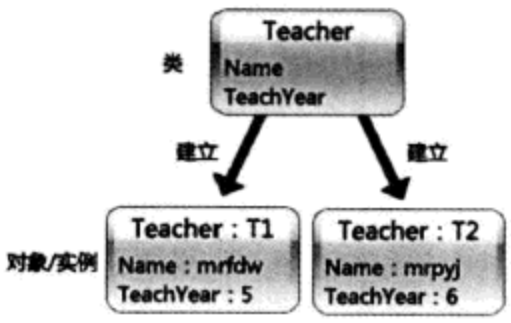


图 6.13　T1、T2 都是套用 Teacher 类建立的对象

尽管有时类和对象可以互换，但它们是不同的概念。类定义对象的类型，但它不是对象本身。对象是基于类的具体实体，有时称为类的实例。

根据前面介绍过的类与对象之间的关系：每一个对象都是依照某个类创建的实例，如图 6.13 所示，Teacher 是类，以它为模板建立的两个 Teacher 对象（T1、T2）都具有 Name 及 TeachYear 成员，但是其中的值可能不一样。

若要建立对象，必须使用 new 关键字，通过使用 new 关键字，后跟类的名称，即可创建对象（也称实例化类对象），其基本语法可表示如下：

```
类对象变量名 = new 类名();
```

【例 6.2】 创建类的对象，示例代码如下所示：

```
Teacher teacher1=new Teacher ();
```

学习手册 说明 其中，teacher1 为 Teacher 类对象定义的变量名，new Teacher ()这个过程称之为类对象的实例化。

创建类的实例后，将向开发人员传递回对该对象的引用。此处只是引用新对象，但不包含对象数据本身（例如不包含示意图中的 Name 属性值 mrfdw 等）。实际上，可以在根本不创建对象的情况下创建对象引用（即只声明对象变量）：

```
Teacher teacher2
```

建议不要创建不引用对象的对象引用，因为在运行时通过这样的引用来访问对象的尝试将会失败。但是，可以创建这样的引用来引用对象，方法是创建新对象，或者将它分配给现有的对象，如下所示：

```
Teacher teacher3= new Teacher ();
Teacher teacher4= teacher3; //将声明的类对象变量 teacher4 分配给 teacher3
```

学习手册 说明 上述示例代码创建了两个对象引用，它们引用同一个对象。因此，通过对 teacher3 对象所做的任何更改都将反映在随后使用的 teacher4 中。

6.2.5　类的成员字段

1．声明成员字段

字段也称成员变量，它表示存储位置，是 C#类不可缺少的一部分，字段的类型可以是 C#中的任何数据类型。

声明字段用标准的变量声明格式和访问修饰符来声明，并且可以对其初始化。例如，下列示例代码中声明了两个字段——Name 和_birthday：

```
public class mrPerson
{
    public string Name;
    private DateTime _birthday;
}
```

前面所讲的类的数据成员其实也就是这里所要讲解的类的成员字段。

2. 访问成员字段

要访问类的成员字段，直接通过成员名称即可，即类对象变量名.字段。

【例 6.3】 创建一个网站，默认主页名为 Default.aspx，接着创建一个名为 mrPerson 的类，在该类中声明一个公有的类成员字段 Name，并赋予初始值“明日科技”，代码如下：

```
public class mrPerson
{
    //定义成员字段
    public string Name="明日科技";
}
```

最后，在默认主页的后台代码 Page_Load 事件中输出类中 Name 字段的值“明日科技”，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    mrPerson person1 = new mrPerson();          //实例化类对象
    Response.Write(person1.Name);               //访问类中成员字段
}
```

最终输出结果为“明日科技”。

除了以上定义的常见的类成员字段外，C#类中还有两个特殊的字段：只读字段和静态字段。

☑ 只读字段

使用 readonly 关键字可以修饰字段为只读，只读字段不能被修改。只读字段可以由初始化赋值语句赋值，或者在同一类的构造函数中赋值。

☑ 静态字段

图 6.14 访问创建的类 mrPerson 中字段 Name 的值

使用关键字 static 修饰的字段称为静态字段，静态字段属于类，为类的所有对象共用。而非静态字段属于某个具体的对象，只能被特定对象专有。

实例位置：光盘\MR\Instance\6\6.1

【例 6.4】 本实例主要演示如何访问类成员的静态与非静态字段。实例首先创建一个名为 MyClass 的类文件，在该文件中定义静态字段和非静态字段。然后，在 Page_Load 事件中调用静态和非静态字段并输出，运行效果如图 6.14 所示。

实现的代码如下所示：

```
public class MyClass
{
    public int x;                                          //定义一个非静态字段
    public static int y;                                   //定义一个静态字段
    public int add_x()                                     //定义一个非静态方法
    {
        x++;
        return x;
    }
    public static int add_y()                              //定义一个静态方法
    {
        y++;
        return y;
    }
}
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("对象访问：" + "<br/>");
    MyClass class1 = new MyClass();
    Response.Write(class1.add_x() + "<br/>");              //输出结果为 1
    MyClass class2 = new MyClass();
    Response.Write(class2.add_x() + "<br/>");              //输出结果为 1
    Response.Write("类访问：" + "<br/>");
    Response.Write(MyClass.add_y() + "<br/>");             //输出结果为 1
    Response.Write(MyClass.add_y() + "<br/>");             //输出结果为 2
}
```

C#严格规定非静态字段只能通过对象来访问，静态字段只能通过类来访问。上述实例中的运行结果体现了类中静态字段和非静态字段的不同。类的静态字段属于类的对象共用，且必须被类访问；而非静态字段的值为每个对象所专有。从运行结果中还可以看到两次引用 add_x()所得到的结果是相同的，因为它们分别属于两个不同对象；而方法 add_y()的结果属于类的对象共用，所以两次调用的结果将被累加起来。

6.2.6　类的成员方法

1．声明方法

类的成员字段用来记录数据，若要执行一段程序或进行数据处理、逻辑判断等，则必须添加方法。方法在类中声明，声明时需要指定访问级别、返回值类型、方法名称以及方法参数。多个方法参数放在括号中，并用逗号隔开。空括号表示无参数方法。方法声明大体如下：

```
修饰符 返回值类型 方法名称（方法参数）
{
  //程序代码
}
```

2. 方法类型

定义类成员方法大体上可分为三种，分别为无返回值的方法、有返回值的方法和带参数的方法。

☑ 无返回值的方法

一些方法并不要求返回数据，此时可以将返回值类型声明为 void，不需要在方法中使用 return 关键字返回数据。

【例 6.5】在创建的 mrPerson 类中使用 void 定义一个无返回值类型的 CheckName 方法。

```
public class mrPerson
{
    //定义类成员
    public string Name;
    public string Address;
    private DateTime _birthday;
    public void CheckName()                          //定义一个无返回值类型的成员方法
    {
        if (string.IsNullOrEmpty(Name))              //判断字段是否为空
        {
            Name = "员工姓名未登记";                  //如果为空则重新为字段赋值
        }
    }
}
```

学习手册 说明 在上述代码中，用了一个 if 语句判断 Name 成员字段的值是否为空字符串，如果为空字符串，就将 Name 的值设定为“员工姓名未登记”。

☑ 有返回值的方法

C#应用程序使用 return 关键字将运算的结果传回原调用程序，return 语句所返回的数据必须和声明类的成员方法时返回值数据类型相同。

【例 6.6】 定义字符类型的方法 AddName()用来返回“新增员工姓名：……”，例如：“新增员工姓名：GLL”。

```
public class mrPerson
{
    //定义类成员
    public string Name;
    public string Address;
    private DateTime _birthday;
    public void CheckName()                          //创建方法检查字段是否为空
    {
        if (string.IsNullOrEmpty(Name))
        {
            Name = "员工姓名未登记";
        }
    }
    public string AddName()                          //定义有返回值类型的方法
    {
        return "新增员工姓名：" + Name;
    }
}
```

使用 return 语句时应注意：若方法返回值的类型不是 void，则方法中至少要有一个 return 语句；方法执行到 return 语句就结束了，因此 return 语句后面的程序将不会被执行。

☑　带参数的方法

无论是否有返回值的方法，为了让方法执行时更灵活，还可以在方法中设定参数。

【例 6.7】 对前面示例中编写过的 CheckName 方法进行改写。

```
public class mrPerson
{
    //定义类成员
    public string Name;
    public string Address;
    private DateTime _birthday;
    public void CheckName(string defaultName)          //定义带有参数类型的方法
    {
        Name = defaultName;
    }
}
```

在类的成员方法中定义的参数可以有一个以上，若有多个参数，则参数之间需要用逗号分隔开。

【例 6.8】 在类的成员方法中定义多个参数。

```
public void AddName(string defaultName,int Age)
{
    //程序代码
}
```

3. 调用方法

在类中定义声明方法之后，就可以在程序中通过方法名称调用方法。调用对象的方法类似于访问字段。在对象名称之后，依次添加句点、方法名称和括号。参数在括号内列出，并用逗号隔开。

实例位置：光盘\MR\Instance\6\6.2

【例 6.9】 首先创建一个名为 mrPerson 的类，在类中创建一个 Name 字段和一个 CheckName 方法，该方法用于检查 Name 字段是否为空，如果为空则设置 Name 字段的值为参数 defaultName 的值。然后，在页面的 Page_Load 事件中实例化 mrPerson 类，并设置该类中的 Name 字段为空，最后输出 Name 字段的值，运行结果如图 6.15 所示。

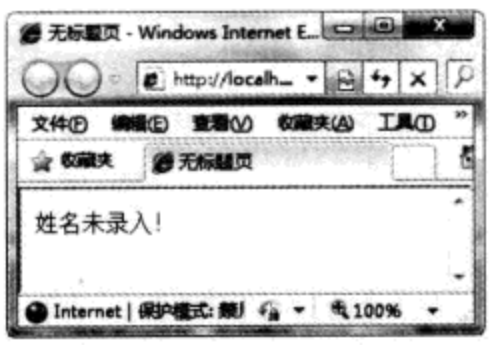


图 6.15　调用创建的类 mrPerson 中的 CheckName 方法检查用户姓名

实现的代码如下所示：

```
public class mrPerson                                  //定义类
{
    public string Name;                                //定义字段
    public void CheckName(string defaultName)          //创建检查字段是否为空的方法
```

```
    {
        if(string.IsNullOrEmpty(Name)){
            Name=defaultName;
        }
    }
}
protected void Page_Load(object sender, EventArgs e)
{
    mrPerson person2 = new mrPerson();          //实例化类对象
    person2.Name = "";
    person2.CheckName("姓名未录入！");          //调用类中定义的 CheckName()方法
    Response.Write(person2.Name);               //输出方法中值
}
```

挑战自我：如何调用有返回值的方法?

读者朋友们可以尝试开发一个 Web 程序，在该程序中看看如何调用有返回值的方法。前面在讲解“有返回值的方法”时编写了一个名为 mrPerson 的类，在该类中创建了一个有返回值类型的 AddName()方法，那么可以根据该方法来实现如何调用有返回值的方法。

6.2.7 类的成员属性

在面向对象程序设计语言中，属性（property）是指对象的特征和状态，具体地说就是指对象的数据成员。用户可以指定数据成员能否被外界直接访问，如果数据成员被指定为 public 的，外界就可以用“对象名.公有数据成员名”访问该成员。C#是完全面向对象的语言，C#倡导一种新途径，对数据成员能够更好地封装和保护，同时又向外界提供更有效的访问形式。C#中用来达到这个目标的就是“属性”，而那些数据成员在 C#中称为“字段”或“域”。

属性的定义方式与字段的定义方式类似，但定义属性比较复杂，属性拥有两个花括号（{}）代码块，一块用于获取属性值，另一块用于设置属性值，这两块又称访问器，分别用 get 和 set 关键字来定义，可以控制对属性的访问级别。属性的基本结构包括访问修饰符、类型、属性名、get 块和 set 块，声明属性的一般形式为：

```
修饰符 类型 属性名
{
    get
    {
        //获取属性代码
    }
    set
    {
        //设置属性代码
    }
}
```

属性的修饰符可以是任何访问控制符（如 public、private 等），还可以被定义为静态。在属性声明中，get 和 set 访问器都必须在属性体的内部声明。

get 和 set 是一种特定的方法，get 必须有一个属性类型的返回值，简单的属性一般与一个私有字段相关联，也控制对个这个字段的访问，此时 get 可以直接返回该字段的值，而 set 用来向字段写入数据，将外界的数据写入字段时，C#使用 value 表示输入的数据，其中 value 为 C#中的关键字，意义为引用用户提供的属性值。

【例 6.10】 假设 Bank 类是一个银行，既能存钱也能取钱。那么，应该通过 get 和 set 访问器将 GetMoney 属性设为可读、可写，如下代码：

```
protected Class Bank()
{
  private int m_money;                                    //私有字段
  public int GetMoney()                                   //可公开的字段
{
      get { return m _money ;}                            //为字段添加可读属性
      set {m _money = value ;}                            //为字段添加可写属性
  }
}
```

那么 m_money 就像银行里的自动存取款机，你看不见里面的 money，但你能用 set(存钱)，也能用 get(取钱)。m_money 是一个私有字段，是分装在类中的，类以外的程序不能直接访问，类的 set 和 get 成员是外部程序访问类内部属性的唯一方法，就像你去银行取钱，你不能直接从银行的保险柜里拿到钱，而是银行营业人员把钱取出来给你的。

编程锦囊：类属性的巧妙设置

类属性的读写设置：属性中的 get 和 set 可以只用一个，如果只有 get 而没有 set，那么这个属性只可读出，不可写入；如果只有 set 而没有 get，那么这个属性只可写入，不可读出，当然只写的属性是没有任何意义的。在 set 访问器中，内建了一个隐含变量 value，可以用来接收外部程序传到属性上的数据，因此在数据设定到变量之前，可以先检查数据的正确性再存到对象中。

get 和 set 有一个合理的分工是：设计字段是为了便于内部方法使用，而尽量与外界隔绝；设计属性考虑的是方便外界的使用，但是不让外界知道的数据一律不给，下面以一个实例讲解来体现。

实例位置：光盘\MR\Instance\6\6.3

【例 6.11】 首先，创建一个名为 EmployeeInfo 的类，在该类中定义三个私有的成员变量 empName、empAge 和 empCompany，然后通过 get/set 或 get 分别设置类中的属性：employeeName（读写属性）、employeeAge（读写属性）、employeeCompany（只读属性）和 employeeInfo（只读属性）来相应地封装该类中定义的那三个私有成员变量。然后，

在默认主页 Default.aspx 的后台代码中，编写如下代码，输出指定用户的信息。运行结果如图 6.16 所示。

图 6.16　定义成员属性，并输出相应的属性值

实现的代码如下所示：

```
public class EmployeeInfo
{
    #region 私有成员
    private string empCompany = "吉林省明日科技";
                                    //公司名称不可修改
    private string empName = "红尘倒影";
    private int empAge = 29;
    #endregion 私有成员
    #region 属性
    public string employeeName                          //通过 get 和 set 设置读写属性
    {
        get { return empName; }
        set { empName = value; }
    }
    public int employeeAge                              //通过 get 和 set 设置读写属性
    {
        get { return empAge; }
        set { empAge = value; }
    }
    public string employeeCompany                       //通过 get 设置只读属性
    {
        get { return empCompany; }
    }
    public string employeeInfo                          //通过 get 设置只读属性
    {
        get { return "公司：" + empCompany + "/名字：" + empName + "/年龄：" + empAge; }
    }
    #endregion 属性
}
protected void Page_Load(object sender, EventArgs e)
 {
     EmployeeInfo employee = new EmployeeInfo();
     Response.Write(employee.employeeCompany);
                                    //输出 employee 类对象中的 employeeCompany 属性值
     Response.Write("<br>");
     employee.employeeAge = 28; //对 employee 类对象中的 employeeAge 属性重新赋值
     Response.Write(employee.employeeInfo);
                                    //输出 employee 类对象中的只读属性 employeeInfo 的值
 }
```

在类 EmployeeInfo 中对私有成员变量 empAge 赋予了初始值 29，而在页面的 Page_Load 事件中却显式地对 employeeAge 赋予了初始值 28，最终运行结果中输出的年龄为 28，主要是因为在该类中对 employeeAge 设置了读写属性，可更改其在类中设置的初始值，而像 employeeCompany 设置的是只读属性，更改不了其值，也不能在页面的 Page_Load 事件中对其显式地进行赋值。

从本实例中可以看出，属性在调用者看来就像一个普通的变量，普通变量怎么用，它就怎么用，但作为类的设计者，可以利用属性来隐藏类中的一些字段，使外界只能通过属性来访问用户设置的字段。在属性声明中，get 和 set 访问器需要注意以下两点。

☑ get 访问器必须在 return 或 throw 语句中终止，并且控制不能超出访问器的有效区域。

使用 get 访问器更改对象的状态是一种错误的编程模式。

【例 6.12】 以下访问器在每次访问 number 字段时都产生更改对象状态的副作用。

```
public int Number
{
    get {
    return number++;                                    //错误操作
   }
}
```

可以将 get 访问器用于返回字段值，或用于计算字段值并将其返回。例如：

```
public string Name
{
   get {
    return name != null ? name : "mrfdw";              //将最后的值返回给字段的调用方
   }
}
```

在上述代码段中使用了三目运算符，如果不对 Name 属性赋值，它将返回值 mrfdw。

☑ 在 set 访问器中对局部变量声明使用隐式参数名（va!ue）是错误的。

【例 6.13】 set 访问器中局部变量声明使用隐式参数，出现错误，示例代码如下所示：

```
public class MyClass
{
   private int value;                                  //声明一个私有字段 value
   public int Value
   {
      set  {
         value = value;                                //局部变量声明使用隐式参数名 (value)是错误的
      }
   }
}
```

6.2.8　构造函数

细心的读者会发现前面在用 ASP.NET 创建 mrPerson 类时会自动生成 public mrPerson() 程序块，如图 6.17 所示。

public mrPerson()程序块就是自动帮助开发人员显示出默认的构造函数（无参数）。构造函数是在创建给定类型的对象时执行的类的一种特殊方法，它具有如下特性：

☑ 构造函数的方法名和类名相同；

☑ 构造函数没有返回类型；

☑ 构造函数的主要作用是实现对类对象的初始化；

☑ 当使用 new 关键字创建一个对象时，系统会自动调用该类的构造函数初始化对象。

实例位置：光盘\MR\Instance\6\6.4

【例 6.14】 创建一个名为 Person 的类，在该类的构造函数中对所定义的公有成员字段 Name 和 Address 进行初始化，并对 decimal 类型的私有成员字段_charge 添加类型为 decimal 的 Change 属性，然后，在 Page_Load 事件中输出在 Person 类的构造函数中对类成员字段 Name 及 Address 的初始值，运行结果如图 6.18 所示。

```
/// <summary>
///mrPerson 的摘要说明
/// </summary>
public class mrPerson
{
    public mrPerson()
    {
        //
        //TODO: 在此处添加构造函数逻辑
        //
    }
}
```

图 6.17 创建 mrPerson 类时自动生成的默认构造函数 public mrPerson()

图 6.18 构造函数的应用

实现的代码如下所示：

```
public class Person
{
public Person()                                          //构造函数
{
        Name = "明日科技";                                //初始化字段
        Address = "吉林省长春市东盛大街 89 号亚泰广场";    //初始化字段
}
 public string Name;
 public string Address;
 private decimal _charge=300000;
 public decimal Change
 {
     get
     {
         return _charge;
     }
     set
     {
         if (_charge > 0)
         {
             _charge = value;
         }
         else
         {
             _charge = 0;
         }
```

创建字段

为 Change 字段设置可读写的属性

```
    }
}
```

学习手册 说明 基本上，构造函数的访问修饰符都是 public，这样外部程序才能通过构造函数建立对象，若为 private，则外部程序无法使用此构造函数，换句话说，无法建立对象。

```
protected void Page_Load(object sender, EventArgs e)
{
    Person p1 = new Person();                                    //实例化类对象
    Response.Write("公司名称："+p1.Name+"<br/>");
    Response.Write("公司地址：" + p1.Address + "<br/>");
    Response.Write("固定资产：" + p1.Change + "<br/>");
}
```

6.2.9　析构函数

析构函数是以类名加~来命名的。.NET Framework 类库有垃圾回收功能，当某个类的实例被认为不再有效，并符合析构条件时，.NET Framework 类库的垃圾回收功能就会调用该类的析构函数实现垃圾回收。

【例 6.15】 构建一个构造函数后，再构造一个析构函数来释放非托管资源。

```
public DataBase()//构造函数
{
   ConnectionString    =    "Data    Source=(lccal);DataBase=mrOnLineExam;User
ID=sa;Password=;";
}
~DataBase()                                          //析构函数，释放非托管资源
{
try {
    if (Connection != null)
    Connection.Close();
}
catch{}
try {
    Dispose();
}
catch{}
}
```

学习手册 说明 一个类中只能有一个析构函数，并且无法调用析构函数，它是被自动调用的。在析构函数中，仅释放类直接拥有的非托管资源，如果类不拥有任何非托管资源，则不要在类中包含析构函数。

6.2.10　情景应用：访问商品类的成员

实例位置：光盘\MR\Instance\6\6.5

【例 6.16】 创建一个网站，在后台代码中首先定义一个类 myfood，该类用来描述药

品的相关信息（包括名称、单价、产地），然后在 Page_Load 事件中创建 myfood 类的对象（创建类的对象请参见 6.2.4 节），然后调用 myfood 类中返回商品信息的方法 foodinfo，并将查找的商品名称作为参数传递给 foodinfo，最后输出该种商品的单价和产地等信息，运行结果如图 6.19 所示。

图 6.19　类成员的访问及应用

实现的代码如下所示：

```
public class myfood
{
    public string strFoodName="香辣鸡腿堡";              //定义公共变量表示商品名称
    private int decPrice = 18;                          //定义商品价格
    protected string strfrom = "吉林省长春市";           //定义商品产地
    //定义公共方法，获取关于商品的相关信息，访问不受限制
    public string foodinfo (string  name)
    {
        string result="";
        if (name == strFoodName)                        //判断传递的参数是否等于商品名称字段
            result = "名称：" + strFoodName + "<br>产地：" + strfrom + "<br>单价："
            + decPrice;
        else
            result = "没有查到商品";
        return result;
    }
}
protected void Page_Load(object sender, EventArgs e)
{
    myfood fd = new myfood();                           //创建 myfood 类的实例
    Response.Write(fd. foodinfo ("香辣鸡腿堡")); //调用 foodinfo 方法返回相关信息
}
```

6.3　面向对象特性之封装

专题讲座：光盘\MR\Video\6\类的封装.exe

➤➤➤视频速递：了解什么是类的封装以及如何实现封装。

C#中可以使用类来达到数据封装的效果，这样就可以使数据与方法封装成单一元素，以便于通过方法存取数据。通过本节的学习要掌握类的封装特性，并学会如何实现它。

6.3.1　封装概述

面向对象编程中，大多数都是以类作为数据封装的基本单位。类将数据和操作数据的方法结合成一个单位。设计类时，不希望直接存取类中的数据，而是希望通过方法来存取数据，这样就可以达到封装数据的目的，方便以后的维护升级，也可以在操作数据时多一层判断。

制造汽车的过程中什么人最牛？当然不是焊钢板的，也不是装轮胎的，更不是拧螺丝

的，而是设计汽车的工程师，因为他知道汽车的运行原理。但是我们开车时，需要知道汽车的运行原理吗？答案是否定的。汽车的运行原理已经被伟大的工程师封装在汽车内部，提供给司机的只是一个简单的使用接口，司机操纵方向盘和各种按钮就可以灵活自如地开动汽车了。

与制造汽车相似，面向对象技术把事物的状态和行为的实现细节封装在类中，形成一个可以重复使用的“零件”。类一旦被设计好，就可以像工业零件一样，被成千上万的对其内部原理毫不知情的程序员使用。类的设计者相当于汽车工程师，类的使用者相当于司机。这样程序员就可以充分利用他人已经编写好的“零件”，而将主要精力集中在自己的专属领域。

6.3.2　封装的实现

C#中可以使用类来达到数据封装的效果，这样就可以使数据与方法封装成单一元素，以便于通过方法存取数据。除此之外，还可以控制数据的存取方式。下面通过一个实例演示如何实现面向对象编程的封装性。

实例位置：光盘\MR\Instance\6\6.6

【例 6.17】 创建一个网站，在后台代码中自定义一个 MyClass 类，该类用来封装加数和被加数属性；然后自定义一个 Add 方法，该方法用来返回该类中两个整数类型属性的和；然后在 Page_Load 事件中实例化自定义类的对象，并分别为 MyClass 类中的两个属性赋值，最后调用 MyClass 类中的自定义方法 Add 返回两个属性的和。运行结果如图 6.20 所示。

图 6.20　类封装属性及算法

实现的代码如下所示：

```
class MyClass                                          //自定义类，封装加数和被加数属性
{
    private int x = 0;                                 //定义 int 型变量，作为加数
    private int y = 0;                                 //定义 int 型变量，作为被加数
    /// <summary>
    /// 加数
    /// </summary>
    public int X                                       //该字段设为可读写
    {
        get{return x;}
        set{x = value;}
    }
/// <summary>
/// 被加数
/// </summary>
public int Y                                           //该字段设为可读写
{
    get{return y;}
    set{y = value;}
}
```

```
    /// <summary>
    /// 求和
    /// </summary>
    /// <returns>加法运算和</returns>
    public int Add()                                    //创建一个方法计算两个数字之和
    {
        return X + Y;
    }
}
protected void Page_Load(object sender, EventArgs e)
{
    MyClass myclass = new MyClass();                    //实例化 MyClass 的对象
    myclass.X = 15;                                     //为 MyClass 类中的属性赋值
    myclass.Y = 10;                                     //为 MyClass 类中的属性赋值
    Response.Write("一个汉堡: "+myclass.X+"<br>");
    Response.Write("一个鸡块: " + myclass.Y + "<br>");
    Response.Write("消费总额: " + myclass.Add()); //调用 MyClass 类中的 Add 方法求和
}
```

6.4 面向对象特性之继承

专题讲座：光盘\MR\Video\6\类的继承.exe

➤➤➤视频速递：了解如何实现类的继承。

利用类的继承机制，用户可以通过增加、修改或替换类中的方法对这个类进行扩充，以适应不同的应用要求。本节将对继承的概念和应用做详细的讲解。

6.4.1 继承概述

在日常生活中的继承一般是说从父母或长辈那里得到或获得某些东西，比如小明的眼睛像母亲，并继承了父亲坚强、勇敢、诚实、善良的性格。在面向对象的世界中也存在继承特性，继承是面向对象程序设计的重要特性之一，继承的最大优点就是提供了代码的重用，在 C#中类只可以单继承，比如定义三个类：A 类、B 类和 C 类，B 类继承于 A 类，所以 B 类不可以再继承于 C 类，如图 6.21、图 6.22 和图 6.23 所示。

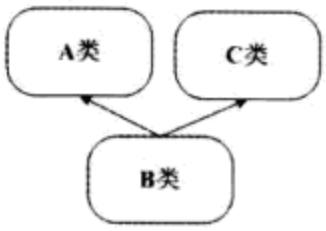
从图 6.21 中可以看到 B 类同时继承于 A 类和 C 类，这样定义是错误的。

从图 6.22 中可以看到 C 类继承于 A 类，而 B 类又继承于 C 类，这样的继承是单继承，是正确的。

从图 6.23 中可以看到 B 类继承于 A 类，C 类也继承于 A 类，因为 B 类继承于 A 类，C 类继承于 A 类，所以这样的继承是正确的。

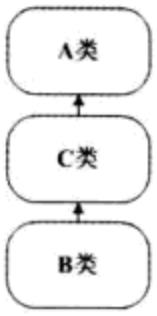
前面曾提到过继承提供了代码的重用，比如开发某公司的人事管理系统，定义员工类可以有姓名和考勤，定义管理者类可以有姓名、考勤和分配任务。如果按原计划我们可以定义一个员工类和一个管理者类，现在有了继承的概念，我们可以为员工类定义姓名字段和考勤方法，然后再定义管理者类，使管理者类继承于员工类，由于继承于员工类，所以管理者类已经从员工类继承到姓名字段和考勤方法，不需要再写姓名字段和考勤方法部分

的代码，只需要写出实现分配任务部分的代码即可，这样会很方便。



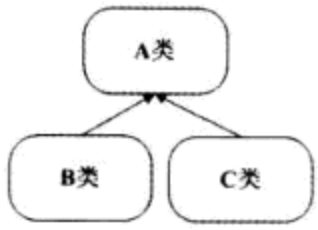
错误

图 6.21　多继承



正确

图 6.22　单继承一



正确

图 6.23　单继承二

类可以从另外一个类继承，这就是说，这个类拥有它继承的类的所有成员（除构造方法外）。利用类的继承机制，程序开发人员可以在已有类的基础上构造新类。

6.4.2　继承的实现

继承可以提供代码重用，继承是怎样使用的呢？继承的方法是在类名后面加上一个冒号（：），然后在其后添加基类名，如图 6.24 所示。

从图 6.24 中可以看到，定义继承很简单，定义 B 类时，可以在 B 类的后面加冒号再加 A 类的名称，表示 B 类继承于 A 类，这时 B 类叫做派生类或子类，A 类叫做基类或父类，A 类与 B 类之间是继承关系，B 类拥有 A 类所有的数据成员，下面结合实例演示继承的用法。

实例位置：光盘\MR\Instance\6\6.7

【例 6.18】 创建一个网站，在后台代码中自定义一个 MyClass1 类；然后再自定义一个 MyClass2 类，该类继承于 MyClass1 类，这时，MyClass2 类就拥有 MyClass1 类中的所有公有成员，并且可以扩展其成员；在 Page_Load 事件中通过 MyClass2 类的对象调用 MyClass1 类中的方法。运行结果如图 6.25 所示。

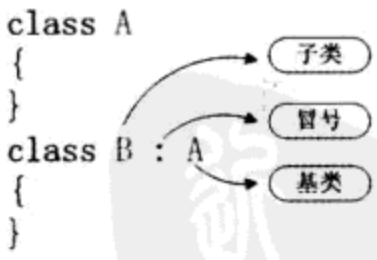


图 6.24　定义继承

图 6.25　子类调用基类中的成员变量

实现的代码如下所示：

```
/// <summary>
/// 自定义类
/// </summary>
class MyClass1
```

```
{
    private int x = 0;                                    //定义 int 型变量，作为加数
    private int y = 0;                                    //定义 int 型变量，作为被加数
    /// <summary>
    /// 加数
    /// </summary>
    public int X                                          //设置该字段为可读写
    {
        get {return x;}
        set{x = value;}
    }
    /// <summary>
    /// 被加数
    /// </summary>
    public int Y                                          //设置该字段为可读写
    {
        get{return y;}
        set{y = value;}
    }
    /// <summary>
    /// 求和
    /// </summary>
    /// <returns>加法运算和</returns>
    public int Add1()                                     //创建该方法用于计算两数之和
    {
        return X + Y;
    }
}
/// <summary>
/// 自定义类，该类继承自 MyClass1
/// </summary>
class MyClass2 : MyClass1                                 //继承第一个类，那么就可以访问 x 和 y
{
    private int z = 0;                                    //定义 int 型变量，作为第 2 个被加数
    /// <summary>
    /// 被加数
    /// </summary>
    public int Z                                          //设置该字段为可读写
    {
        get{return z;}
        set{z = value;}
    }
    /// <summary>
    /// 求和
    /// </summary>
    /// <returns>加法运算和</returns>
    public int Add2()                                     //创建该方法用于计算数字相加之和
    {
        return X + Y + Z;
    }
}
protected void Page_Load(object sender, EventArgs e)
 {
      MyClass1 myclass1 = new MyClass1();                 //实例化 MyClass1 的对象
      MyClass2 myclass2 = new MyClass2();                 //实例化 MyClass2 的对象
```

```
        myclass1.X = 3;                    //为 MyClass1 类中的属性赋值
        myclass1.Y = 4;                    //为 MyClass1 类中的属性赋值
        myclass2.X = 5;                    //使用 MyClass2 类对象调用基类中的属性并赋值
        myclass2.Y = 6;                    //使用 MyClass2 类对象调用基类中的属性并赋值
        myclass2.Z = 7;                    //为 MyClass2 类中的属性赋值
        //调用 MyClass1 类中的 Add1 方法求和
        Response.Write("基类中的 X 和 Y 相加：  "+myclass1.Add1()+"<br>");
        //使用 MyClass2 类对象调用基类中的方法
        Response.Write("子类中调用基类的 X、Y 相加：  "+myclass2.Add1()+"<br>");
        //调用 MyClass2 类中的 Add2 方法求和
        Response.Write("子类中调用基类中的 X、Y 与子类的 Z 相加：  " + myclass2.Add2());
}
```

6.5　面向对象特性之多态

专题讲座：光盘\MR\Video\6\类的多态.exe

➤➤➤视频速递：了解什么是类的多态以及如何实现多态。

通过前边的学习，我们对封装和继承已经有所了解，接下来就认识一下类的多态。多态，顾名思义是多种形态，可以简单地描述成“一个接口，多种方法”。通过对本节的学习，读者能够了解多态的概念及其应用。

6.5.1　认识多态

封装可以隐藏实现细节，使得代码模块化；继承可以扩展已经存在的代码模块，它们的目的都是代码重用；而多态则可简单地概括为“一个接口，多种方法”，它是在程序运行的过程中才决定调用的方法，其原理建立在“从父类继承而来的子类可以转换为其父类”这个规则之上。多态是一种概念，也是一种思想。重载（overload）和重写（override）都是多态的体现。

1. 重载

重载指可以定义一些名称相同的方法，通过定义不同的输入参数来区分这些方法，然后在调用时，CLR 就会根据不同的参数样式来选择合适的方法执行。

【例 6.19】 重载 myTest 方法，只改变参数的数量或类型，代码如下：

```
public  void  myTest(int  x ,int y){}
public  void  myTest (int  x ,ref  int y){}
public  void  myTest (int  x ,int y ,string a){}
```

从上述示例代码中可以总结重载特征如下：

☑ 方法名必须相同。

☑ 参数列表必须不相同，与参数列表的顺序无关。

☑ 返回值类型可以不相同。

2. 重写

重写也称为覆盖，从字面就可以知道，它是覆盖了一个方法并且对其重写，以求达到

不同的作用。override 是重写基类的方法，在基类中的方法必须有修饰符 virtual（方法的声明中加上关键字 virtual，这种方法称为虚方法）。同时，在派生类中，重写方法中需要加上关键字 override。

【例 6.20】 在父类中创建一个虚方法，其功能是返回指定字符串。然后，在继承父类的子类中重写父类中的这个方法，使其返回不同的字符串，代码如下：

```
class BaseClass                                    //父类
{
    public virtual string GetName()                //应用 virtual 关键字创建父类中虚方法
    {
        return "父级名称";
    }
}
class SubClass : BaseClass                         //子类继承父类
{
    public override string GetName()               //重写父类中的虚方法
    {
        return "子级名称";
    }
}
```

重写以后，用基类对象和子类对象访问 GetName ()方法，结果都是访问在子类中重新定义的方法，基类的方法相当于被覆盖掉了。

在使用重写时要注意以下的几点：

① virtual 修饰符不能与 private、static、abstract 或者 override 修饰符同时使用。

② override 修饰符不能与 new、static 或者 virtual 修饰符同时使用，并且重写方法只能用于重写基类中的虚方法。

『名词解析』什么是虚方法？

“虚”这个字可以理解为“不真实的”或“虚幻的”，虚方法用在哪里呢？比如 B 类继承于 A 类，B 类可以使用 A 类中定义的方法和字段，如果 A 类中有一个方法并不适合 B 类使用，那就要用到虚方法了。可以将 A 类中的这个方法定义为虚方法，这时在 B 类就可以重写这个虚方法，将该方法变为自己的方法。

6.5.2 重载

从前面的介绍可知，重载是面向对象多态性的一个重要特征，它表示在一个类中定义多个同名方法，这些方法通过不同的参数类型或者参数个数进行区分。

实例位置：光盘\MR\Instance\6\6.8

【例 6.21】 创建一个网站，在后台代码中创建一个名为 MyClass 的类，然后在该类中定义一个 isEqual 方法的两个重载，分别用来判断数值和字符串是否相等。在 Page_Load 事件中，调用 MyClass 类中定义的 isEqual 方法的两个重载方法分别判断给定的两组数据是否相等，运行结果如图 6.26 所示。

实现的代码如下所示：

```
class MyClass     //定义类
{
    public static bool isEqual(int a, int b)
                  //创建带两个整型参数的方法
    {
        if (a == b)          //如果 a 等于 b
        {
            return true;     //返回 true
        }
        else                 //否则
        {
            return false;    //返回 false
        }
    }
    public static bool isEqual(string a, string b) //重载该方法，参数是字符串类型
    {
        if (a == b)          //如果 a 等于 b
        {
            return true;     //返回 true
        }
        else                 //否则
        {
            return false;    //返回 false
        }
    }
}
protected void Page_Load(object sender, EventArgs e)
{
    bool x = MyClass.isEqual(927,112);          //调用 isEqual 方法比较两个整数
    bool y = MyClass.isEqual("1314", "1314");   //调用 isEqual 方法比较字符串
    Response.Write("927 和 112 比对的结果："+x + "<br/>");    //输出结果
    Response.Write("1314 和 1314 比对的结果："+y);
}
```

图 6.26　重载方法比对数据是否相等

6.5.3　重写

C#中，类的多态性是通过在子类（派生类）中重写基类的虚方法或函数成员来实现的。在派生于同一个类的不同对象上执行任务时，多态是一种极为有效的技巧，使用的代码最少。可以把一组对象放到一个数组中然后调用它们的方法，在这种情况下多态的作用就体现出来了，这些对象不必是相同类型的对象。当然如果它们都继承自某个类，可以把这些子类（派生类）都放到一个数组中。如果这些对象都有同名方法，就可以调用每个对象的同名方法。

实例位置：光盘\MR\Instance\6\6.9

【例 6.22】 本实例主要介绍如何实现多态中的重写。本实例首先创建一个虚方法，其返回指定的字符串。然后在创建的三个类中分别重写这个虚方法，使其返回不同的值，从而为读者形象地演示了如何实现重写虚方法，运行结果如图 6.27 所示。

图 6.27　重写基类的虚方法输出不同的歌曲

（1）新建一个网站，默认主页（Defautl.aspx）的后台代码中创建一个名为 mysong 的类作为其他对象继承的基类，在该类中定义一个名为 lovesong ()的方法，lovesong ()方法带有一个 virtual 修饰符，该修饰符表明该基类的派生类可以重写该方法，mysong 类的 lovesong ()方法将输出如下信息“我最爱的歌曲排名”，代码如下：

```
public class mysong
{
    public virtual string lovesong()          //定义一个虚方法，以便子类中重写
    {
        return "我最爱的歌曲排名";
    }
}
```

（2）定义三个类，这三个类都派生自 mysong 类。每个类都有一个同名的 lovesong ()方法，这些 lovesong ()方法都有一个重写修饰符。重写修饰符可让该方法在运行时重写其基类的虚方法，具体代码如下：

```
public class song1 : mysong                   //最喜欢的歌曲 1
 {
    public override string lovesong()         //重写虚方法
    {
        return "凡人歌";
    }
}
public class song2 : mysong                   //最喜欢的歌曲 2
{
    public override string lovesong()         //重写虚方法
    {
        return "想起";
    }
}
public class song3 : mysong                   //最喜欢的歌曲 3
{
    public override string lovesong()         //重写虚方法
    {
        return "找个好人嫁了吧";
    }
}
```

（3）在默认主页（Defautl.aspx）的后台代码 Page_Load 事件中，首先创建一个数组，数组元素是 mysong 类的对象，该数组名为 dObj，是由四个 mysong 类型的对象组成。接着，初始化 dObj 数组，由于 song1、song2 和 song3 类都是 mysong 类的派生类，所以这些类可以作为 dObj 数组元素的类型，最后执行 foreach 循环，并输出数组中的每个元素，代码如下：

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

```
{
    mysong[] dObj = new mysong[4];                          //创建一个类的数组
    dObj[0] = new song1();                                  //实例化子类
    dObj[1] = new song2();
    dObj[2] = new song3();
    dObj[3] = new mysong();
    foreach (mysong drawObj in dObj)                        //遍历类的数组
    {
        Response.Write(drawObj.lovesong() + "<br/>"); //输出子类中重写方法后的返回值
    }
}
```

6.6　实 战 练 习

6.6.1　使用面向对象的思想查找数字

➤➤➤题目描述

面向对象的一个经常使用的特性是封装，可以把要实现的功能封装到某个方法中，当需要使用该功能时，只需调用这个方法并传入实参即可。本实例尝试开发一个程序，要求使用面向对象的思想实现在字符串中查找所有的数字，程序运行效果如图 6.28 所示。

图 6.28　在字符串中查找数字

➤➤➤技术指导

本练习的关键技术点是定义一个用来判断某个字符是否为数字的方法，首先在该方法中定义一个布尔型变量并初始化为 false，作为逻辑判断标记，定义一个由数字（0～9）组成的字符串数组，作为判断字符是否为数字的参照对象，然后通过 foreach 语句遍历数组中的元素，并与要判断的字符作比较，若在数组中可以找到该字符，则该方法返回 true，否则返回 false。实现判断某个字符是否为数字的方法的参照代码如下：

```
public static bool getNumeric(string str)
{
    bool b = false;                                         //定义逻辑判断标记
    string[] ArrayInt = new string[] { "0", "1", "2", "3", "4", "5", "6", "7",
    "8", "9" };                                             //定义由数字组成的字符串数组
    foreach (string n in ArrayInt)                          //判断字符是否被包含在数组中
    {
        if (n == str)                                       //若找到对应的数字字符
        {
            b = true;                                       //逻辑标记设为 true
            break;                                          //跳出循环
        }
    }
    return b;                                               //若为 true，则判断该字符是数字
}
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/6.6.1**

6.6.2 使用面向对象的思想实现简单计算器

➤➤➤题目描述

计算器是编程初学者最常见的一个例子，很多高校通常都是用编写计算器代码来考查学生的编程能力。本实例使用面向对象编程思想来编写一个简单的计算器，仅供读者参考，程序运行结果如图 6.29 所示。

图 6.29 简单计算器

➤➤➤技术指导

开发本实例时，首先需要创建一个名为 Ccount 的类，在该类中创建一个名为 Sum 的方法，该方法有三个参数，分别代表需要进行运算的两个数以及运算符，在该方法中通过 switch 语句根据不同的运算符进行相应的运算，关键代码如下：

```
public class CCount
{
    public int Sum(int a, int b, string type)
    {
        switch (type)                          //switch 根据传递过来的运算符执行相应运算
        {
            case "+":                          //加法运算
                return a + b; break;
            case "-":                          //减法运算
                return a - b; break;
            case "*":
                return a * b; break;           //乘法运算
            case "/":                          //除法运算
                return a / b; break;
            default:                           //返回默认值
                return 0; break;
        }
    }
}
```

选择运算符执行的操作

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/6.6.2**

6.7　本 章 小 结

本章主要介绍了面向对象程序设计的基本知识，其中，首先对属性、方法和结构的使用进行了讲解，然后重点介绍了类与对象的使用，并对面向对象程序设计的 3 个特性（封装、继承和多态）及其使用进行了详细讲解。学习完本章，读者应该熟练掌握面向对象程序设计的基本知识，并能够在实际应用中使用面向对象技术进行程序设计。

第 7 章

掌握流程控制语句

（名师课堂：56 分钟）

C#程序设计中，流程控制语句是最常用的技术之一，语句是程序完成一次完整操作的基本单位，在默认情况下，程序的语句是顺序执行的。但是，如果一个程序只能顺序执行各语句，那么程序功能会受到很大的限制。C#中有很多流程控制语句，通过这些语句可以控制程序代码的执行次序，提高程序的灵活性，从而实现比较复杂的程序。本章将从初学者的角度考虑，详细讲解 C#流程控制语句，为了便于读者理解，结合了大量的实例。通过本章的学习，读者可以掌握以下知识：

- 掌握 if 条件语句的用法
- 掌握 switch 多分支语句的用法
- 掌握 while 语句的用法
- 掌握 for 语句的用法
- 掌握 foreach 语句的用法

7.1　接触条件语句

专题讲座：光盘\MR\Video\7\条件语句.exe

➤➤➤视频速递：通过本视频能够更好地的掌握 if 语句、switch 语句的用法。

条件语句可根据不同的条件执行不同的语句。条件语句主要包括 if 条件语句和 switch 多分支语句，本节将向读者详细介绍如何使用条件语句。

7.1.1　if 条件语句

英文单词 if 可以翻译成“如果”，例如，这里提供一句话，大家可以考虑如何通过 if 语句来实现，“如果你能坚持不懈地努力，那么就会成功”，这句话如果使用 if 语句表达就应该是下面的形式：

```
if(你能坚持不懈地努力)
{
        就会成功;
}
```

也就是说，如果你选择坚持不懈地努力，那么就会实现自己的梦想，坚持不懈地努力是成功的先决条件，所以“()”里的内容是前提条件，只有满足了“()”里的内容，才能执行“{}”号里的代码，这便是 if 语句的基本用法。

if 语句的基本格式如图 7.1 所示。

其中，if 语句中的条件可以是任何关系表达式，当条件为真时执行 if 语句后边的语句块，当条件为假时则执行 else 后边的语句块。语句块就是大括号里边的内容，在语句块里可以编写任何行数的 C#语句。

if 语句用于根据一个布尔表达式的值选择一条语句来执行，其执行流程如图 7.2 所示。

语　法	示　例

if（条件）
{
条件成立时的操作
}
else
{
条件不成立时的操作
}
	if（今天下雨）
	{
	带伞
	}
	else
	{
	不带伞
	}

图 7.1　if 语句的基本格式

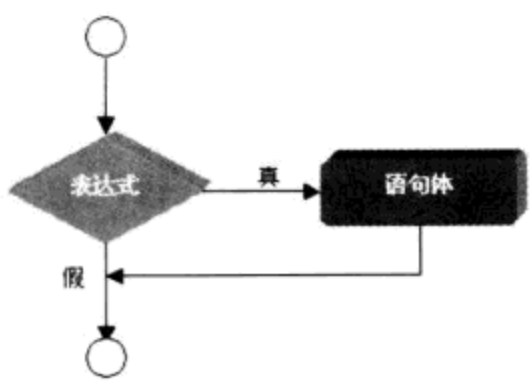


图 7.2　if 语句执行流程

实例位置：光盘\MR\Instance\7\7.1

【例 7.1】 if...else 语句在开发中应用非常广泛，例如，在会员登录、会员注册及其他

很多方面都会出现它的影子。本实例主要通过 if...else 语句判断注册用户选择的性别，运行结果如图 7.3 所示。

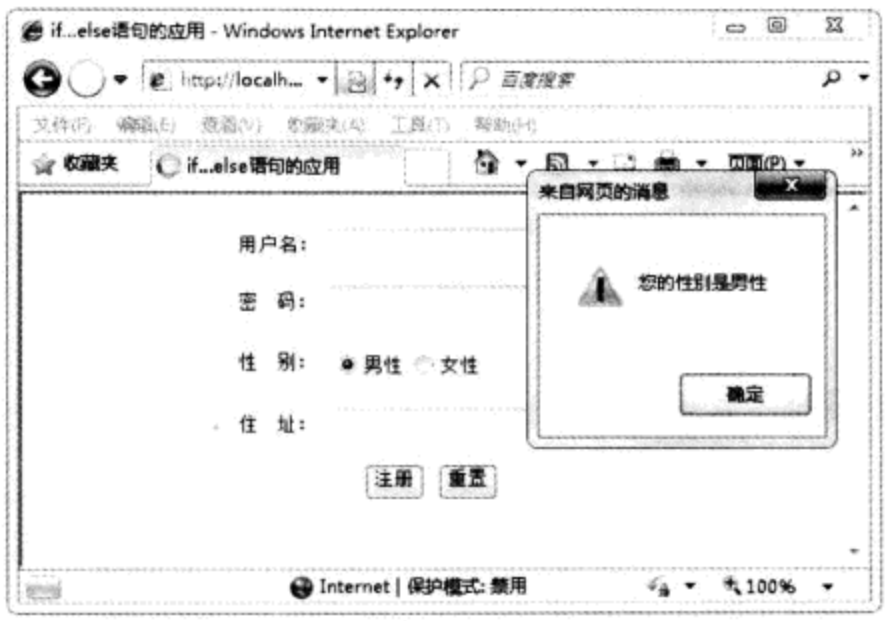


图 7.3 if...else 语句判断选择性别

实现的代码如下所示：

```
protected void btnreg_Click(object sender, EventArgs e)
{
    if (rblsex.SelectedIndex == 0)                    //如果选择了第一项则执行 if 分支
    {
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('您的性别
        是男性');", true);
    }
    else                                              //否则，执行 else 分支
    {
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('您的性别
        是女性');", true);
    }
}
```

学习手册 技巧 编写程序时，读者要养成良好的编写习惯。在使用 if 语句时，通常是在 if 语句和 else 语句后使用大括号，甚至在只有一条语句的时候也使用大括号，并且对大括号内的语句使用缩进，这样在以后添加其他语句时，会变得很容易，同时也增加了代码的可读性，有助于避免错误。

除了上述的 if...else 语句外，if...else 语句和 if 语句、if...else 语句之间都可以互相嵌套使用。所以，if...else 语句的应用是非常灵活的，读者要根据实际情况选择如何使用 if...else 语句和 if 语句。

多重 if 语句的基本格式如图 7.4 所示。

下面通过实例演示多重 if 语句在实际开发中的应用。

实例位置：光盘\MR\Instance\7\7.2

【例 7.2】为了让读者更好地了解 if...else 语句之间的嵌套，下面通过一个实例来演示。本实例主要是对会员输入的注册信息进行验证，判断用户是否输入完整信息，运行结果如

图 7.5 所示。

语　　法	示　　例

If（条件 1）{
满足条件 1 执行的语句
}
else if（条件 2）{
满足条件 2 执行的语句
}
else if（条件 3）{
满足条件 3 执行的语句
}
……可以有很多个 else if
else{
上面条件都不满足时执行
}
	给女朋友送花
	if（玫瑰花 11 朵）
	{
	一心一意
	else if（玫瑰花 1314 朵）
	{
	一生一世
	}
	else if（玫瑰花 3344）
	{
	生生世世
	}
	else{分手吧}

图 7.4　多重 if 语句的基本格式

图 7.5　if...else 语句嵌套使用验证用户输入信息是否完整

实现的代码如下所示：

```
protected void btnreg_Click(object sender, EventArgs e)
{
    if (txtname.Text.Length == 0)   //如果没输入用户名，则执行 if 分支
    {
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请输入
        用户名');", true);
    }
    else                            //否则，执行下面的 else 分支
    {
        if (txtpwd.Text.Length == 0) //如果没输入密码，则执行嵌套 if…else 语句的 if 分支
        {
            ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请输入
            密码');", true);
        }
        else                        //否则，执行 else 分支
        {
            //如果没有输入住址，则继续执行嵌套的 if…else 语句的 if 分支
            if (txtaddr.Text.Length == 0)
```

```
            {
                ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请
                输入您的住址');", true);
            }
            else                                    //否则，执行 else 分支
            {
                ClientScript.RegisterStartupScript(this.GetType(), "", "alert('信息
                输入已经完成');", true);
            }
        }
    }
}
```

7.1.2 switch 多分支语句

switch 语句是多路选择语句，它根据某个值来使程序从多个分支中选择一个用于执行，在许多情况下，使用 switch 语句可以简化 if…else 语句，并且执行效率更高。为了能让读者更好地理解，请参看如图 7.6 所示的 switch 语句执行流程。

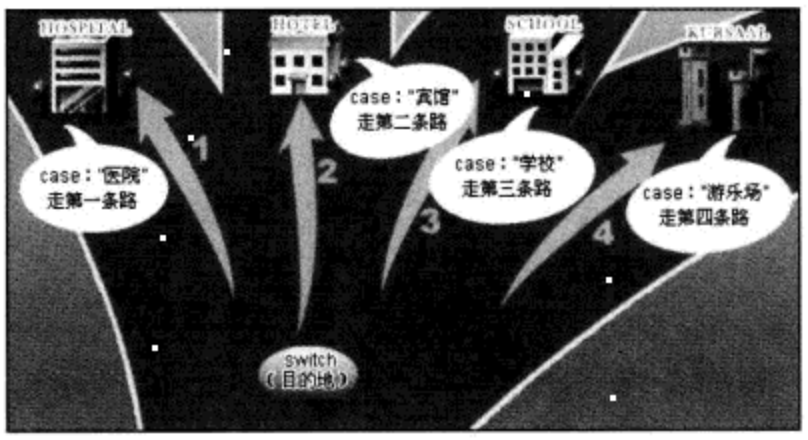


图 7.6　switch 语句执行流程

通过图 7.6，读者能直观地了解 switch 语句的大致用法，接下来我们详细地了解 ASP.NET 中 switch 语句的基本格式，如图 7.7 所示。

语　法	示　例

Switch(变量)
{
case 值 1：
如果变量等于值 1 则执行该语句块
break；
case 值 2：
如果变量等于值 2 则执行该语句块
break；
case 值 3：
如果变量等于值 3 则执行该语句块
break；
……
default：
如果上面条件都不满足则执行文该语句块
}
	还是以给女朋友送 x 多花为例
	switch(x)
	{
	case 11：一心一意 break；
	case 1314：一生一世 break；
	case 3344：生生世世 break；
	……
	default：
	分手吧
	}

图 7.7　switch 语句的基本格式

switch 语句的类型主要包括 sbyte、byte、short、ushort、int、uint、long、ulong、char、string 和枚举类型。每个 case 后面的值必须是与图 7.7 中变量类型相同的一个常量，不能是变量。同一个 switch 语句中的两个或多个 case 标签中指定同一个常数值，就会导致编译出错，一个 switch 语句中最多只能有一个 default 标签，并且每一个标签后边都需要一个 break 语句跳过 switch 语句的其他标签。读者也可以参照 switch 语句的流程图来理解 switch 语句的用法，如图 7.8 所示。

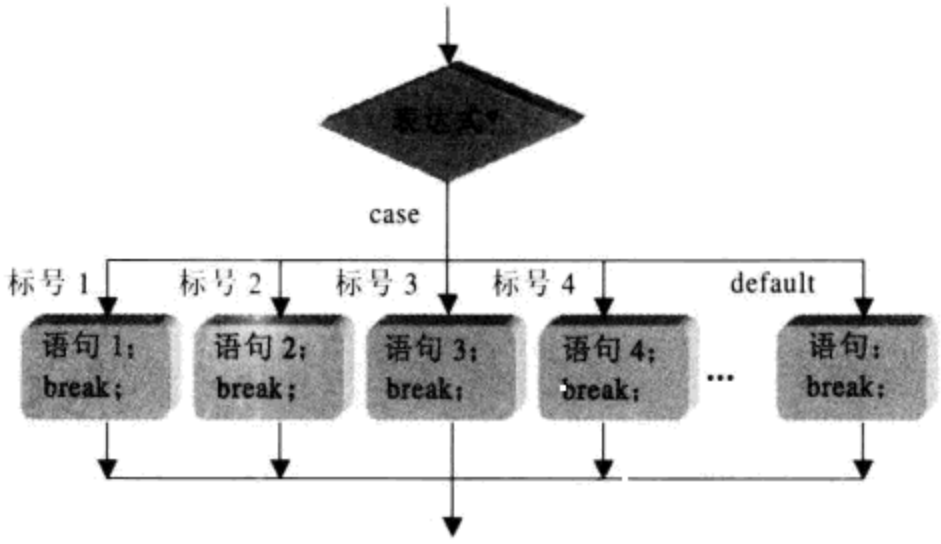


图 7.8　switch 语句的流程图

下面通过实例演示如何使用 switch 语句实现多分支选择。

实例位置：光盘\MR\Instance\7\7.3

【例 7.3】 通过 switch 语句来判断输入的月份属于哪个季节，运行结果如图 7.9 所示。

图 7.9　switch 语句判断输入的月份对应的季节

实现的代码如下所示：

```
protected void btnresult_Click(object sender, EventArgs e)
 {
      bool isnum = Information.IsNumeric(txtmonth.Text.Trim());
                                                    //判断输入的是否是数字
      if (isnum)                                    //如果是数字
      {
          //判断数字是否在 1 和 12 之间
          if (Convert.ToInt32(txtmonth.Text.Trim()) >= 1 && Convert.ToInt32
          (txtmonth.Text.Trim()) <= 12)
```

```
{
    string season = string.Empty; //声明变量存储根据月份推算出来的季节名称
    int month = Convert.ToInt32(txtmonth.Text.Trim()); //获取输入的鱼粉
    switch (month)                    //使用 switch 语句，判断输入的月份所属的分支
    {
        case 12:
        case 1:
        case 2:
            season = "冬季"; break;  //如果输入的月份是 12、1 或者 2，则属于冬季
        case 3:
        case 4:
        case 5:
            season = "春季"; break; //如果输入的月份是 3、4 或者 5，则属于春季
        case 6:
        case 7:
        case 8:
            season = "夏季"; break; //如果输入的月份是 6、7 或者 8，则属于夏季
        case 9:
        case 10:
        case 11:
            season = "秋季"; break;  //如果输入的月份是 9、10 或者 11，则属于秋季
        default:
            season = "不存在"; break;
                                //如果不属于任何一个分支，则字符串变量赋值为“不存在”
    }
    txtShow.Text = season;
}
        txtShow.Text = season;
    }
    else                                          //如果输入的月份不在 1 和 12 之间
    {
        txtShow.Text = "您输入的月份错误，请重新输入";
    }
}
else                                              //如果输入的不是数字
{
    txtShow.Text = "请输入月份";
}
}
```

学习手册 说明 在许多情况下，switch 语句可以简化 if...else 语句结构，而且比 if...else 语句执行效率更高。IsNumeric 是 VB 库里的一个函数，其功能是判断数据是否是数字。首先要添加 Web 引用，具体方法是在项目名称上单击鼠标右键，在右键菜单中选择“添加引用”，在打开的添加引用窗口中选择“.NET”选项卡，然后选择“Microsoft.VisualBasic”。最后，在后台代码中引入命名空间“using Microsoft.VisualBasic;”，这样就可以使用 IsNumeric 方法了。

7.1.3 情景应用：判断数字奇偶性

实例位置：光盘\MR\Instance\7\7.4

【例 7.4】 使用多个 if 语句验证用户输入的数据是否是大于 0 的数字，同时，通过 if 语句判断输入的数字是否为偶数，并弹出相应的提示信息，实例效果如图 7.10 所示。

图 7.10　判断数字奇偶性

实现的代码如下所示：

```
protected void btnjuge_Click(object sender, EventArgs e)
{
    if (txtNum.Text.Trim().Length == 0)          //如果没有输入数据
    {
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请输入
        数据！');", true);                        //提示信息
        return;
    }
    if (!Regex.IsMatch(txtNum.Text.Trim(), @"^\+?[1-9][0-9]*$"))
                                                  //判断输入的是否是大于 0 的数字
    {
        //弹出提示信息
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请输入
        大于 0 的数字！');", true);
        return;
    }
    if (Convert.ToInt32(txtNum.Text.Trim()) % 2 == 0)     //判断是否为偶数
    {
        //弹出提示信息
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('您输入
        的是偶数！');", true);
        return;
    }
    if (Convert.ToInt32(txtNum.Text.Trim()) % 2 != 0)     //判断是否是奇数
    {
        //弹出提示信息
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('您输入
        的是奇数！');", true);
        return;
    }
}
```

7.2　掌握循环语句

专题讲座：光盘\MR\Video\7\循环语句.exe

➤➤➤视频速递：能够更好地掌握 while 语句、do…while 语句、for 语句和 foreach 语句的用法。

循环语句就是在满足一定条件的情况下反复执行某一个操作。在 C#中提供了四种常见的循环语句，分别是 while 语句、do...while 语句、for 语句和 foreach 语句。本节分别对这 4 种循环语句进行介绍，希望读者能够认真学习。

7.2.1 while 语句

while 语句用于根据条件值执行一条语句零次或多次，当每次 while 语句中的代码执行完毕时，将重新查看是否符合条件值，若符合则再次执行相同的程序代码，否则跳出 while 语句，执行其他程序代码。while 语句的基本格式如图 7.11 所示。

在条件为真的情况下，会重复执行 while 循环体中的程序代码。由于 while 表达式的测试在每次执行循环前发生，因此 while 循环执行零次或多次，这与执行一次或多次的 do 循环不同。while 循环非常类似于 do 循环，但有一个非常重要的区别：while 循环中的条件测试是在循环开始时进行，而不是最后。如果测试结果为 false，就不会执行循环。程序会直接跳转到循环后面的代码。while 语句的执行流程如图 7.12 所示。

语　法	示　例

while（条件）
{
　循环体中的语句
}
	对不起我爱你，一万句我爱你表达我的心
	int love=0;
	
	while (love<10000)
	{
	　　我爱你
	　　love++;
	}

图 7.11　while 语句的基本格式

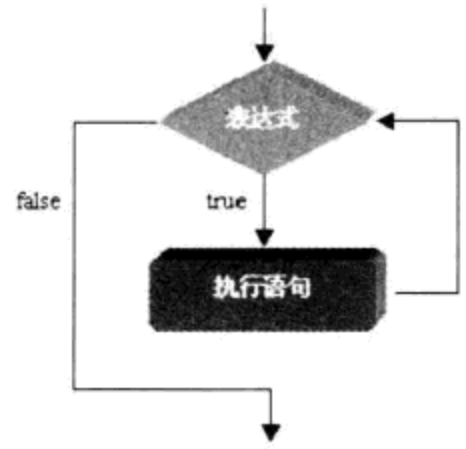


图 7.12　while 语句的执行流程

下面通过实例演示如何使用 while 语句实现代码的循环执行。

实例位置：光盘\MR\Instance\7\7.5

【例 7.5】 声明两个 int 类型的变量 s 和 num，分别初始化为 0 和 80。然后通过 while 语句循环输出，当 s 大于 40 时，使用 break 语句终止循环。当 s 是偶数时，使用 continue 语句将程序转到下一次循环。从而实现输出 40 以内的所有奇数，效果如图 7.13 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int s = 0, num = 80;                    //声明两个 int 类型的变量并初始化
    while (s < num)                         //调用 while 语句，当 s 小于 num 时执行
    {
        s++;                                //s 自增 1
        if (s > 40)                         //使用 if 语句判断 s 是否大于 40
        {
            break;                          //使用 break 语句终止循环
```

图 7.13　输出 40 以内的所有奇数

```
        }
        if ((s % 2) == 0)                    //调用 if 语句判断 s 是否为偶数
        {
            continue;                        //使用 continue 语句将程序转到下一次循环
        }
        Response.Write(s + "<br>");          //输出 s
    }
}
```

学习手册 技巧　在 while 语句的嵌入语句块中，可以使用 break 语句将控制转到 while 语句的结束点，而 continue 语句则可用于将控制直接转到下一次循环。

7.2.2　do…while 语句

do...while 循环与 while 循环类似，只要条件表达式为 true，循环体就会不断地重复执行，但 do...while 语句会先执行一次循环体，然后判断条件表达式是 true 还是 false。它对应的循环体执行一次（至少一次）或若干次。与 while 循环方式相比，do...while 循环是“先斩后奏”。其基本格式如图 7.14 所示。

学习手册 注意　while 条件表达式后的分号一定要写，否则会出现语法错误。

do…while 语句的执行流程如图 7.15 所示。

语　法	示　例

do
{
　循环体语句
}while（条件）；
	对不起我爱你，一万句我爱你表达我的心
	int love=0;
	do
	{
	　我爱你
	　love++;
	}while (love<10000)；

图 7.14　do...while 语句基本格式

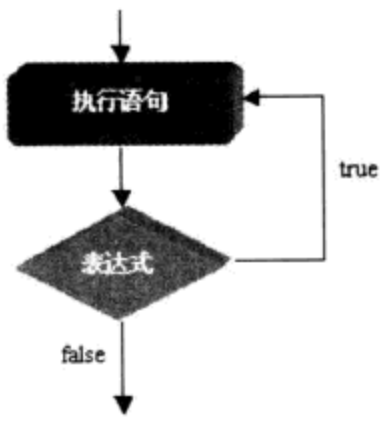


图 7.15　do…while 语句的执行流程

图 7.16　do…while 语句的应用

为了加深读者对 do…while 语句的理解，下面通过一个实例来演示一下。

实例位置：光盘\MR\Instance\7\7.6

【例 7.6】 声明一个 bool 类型的变量 term，并初始化为 false。再声明一个 int 类型的数组，并初始化该数组。然后调用 do…while 语句，通过 for 语句循环输出数组中的值，效果如图 7.16 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    bool term = false;                          //声明一个 bool 型变量 term 并初始化为 false
    int[] myArray = new int[5] { 0, 1, 2, 3, 4 };      //声明一个 int 数组并初始化
    do                                          //调用 do…while 语句
    {
        for (int i = 0; i < myArray.Length; i++) //调用 for 语句输出数组中的所有数据
        {
            Response.Write(myArray[i]);          //输出数组中数据
        }
    } while (term);                             //设置 do…while 语句的条件
}
```

从代码中不难看出，bool 类型变量 term 被初始化为 false，但是 do…while 语句依然执行了一次 for 循环，将数组中的值输出。由此可以说明，do…while 语句至少要执行代码一次，无论最后的条件是 true 还是 false。

7.2.3　for 语句

for 语句用于重复执行一个语句或者语句块，直到指定的条件不成立，或者表达式计算为 false 时，才退出循环。for 语句将初始值、布尔判断和更新值都编写在同一行程序中，其基本格式如图 7.17 所示。

语　法	示　例

for(表达式1；表达式2；表达式3)
{
　　循环体语句
}
	对不起我爱你，一万句我爱你表达我的心
	for (int x=0; x<10000; x++)
	{
	　　我爱你
	}

图 7.17　for 语句的基本格式

for 语句的三个参数都是可选的，理论上并不一定完全具备。但是如果不设置循环条件，程序就会产生死循环。

for 语句的执行流程如图 7.18 所示。

【例 7.7】 通过 for 语句遍历字符串数组中的所有字符串，并输出字符串，代码如下所示：

```
string[] files = new string[10] { "红尘倒影","洪杰","小梁","小剑","小彬" ,"玲玲"};
                                                          //创建并初始化一个字符串数组
for (int i=0;i<files.Length ;i++ )                        //使用 for 语句遍历数组
{
    Response.Write(files[i]+"  ");                        //输出数组中的内容
}
```

实例位置：光盘\MR\Instance\7\7.7

【例 7.8】《九九乘法歌诀》又常称为“小九九”，学生学的“小九九”口诀就是由此而来。现在人们把那些有心计、会算计、善谋划的人形容为心里有“小九九”，中国使用《九九乘法歌诀》的时间较早，在《荀子》、《管子》、《淮南子》和《战国策》等书中就能找到“三九二十七”、“六八四十八”、“四八三十二”和“六六三十六”等句子，由此可见，早在“春秋”和“战国”时期，《九九乘法歌诀》就已经开始流行了。本例通过 for 语句实现输出《九九乘法歌诀》表，效果如图 7.19 所示。

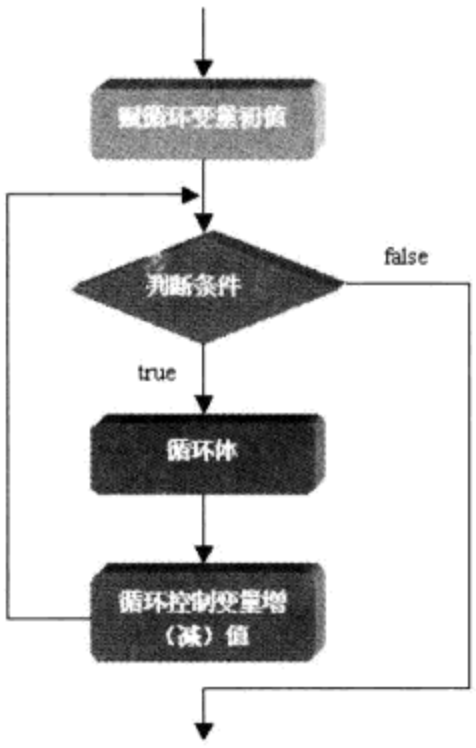


图 7.18　for 语句的执行流程

九九乘法歌诀

```
1×1=1
1×2=2   2×2=4
1×3=3   2×3=6   3×3=9
1×4=4   2×4=8   3×4=12  4×4=16
1×5=5   2×5=10  3×5=15  4×5=20  5×5=25
1×6=6   2×6=12  3×6=18  4×6=24  5×6=30  6×6=36
1×7=7   2×7=14  3×7=21  4×7=28  5×7=35  6×7=42  7×7=49
1×8=8   2×8=16  3×8=24  4×8=32  5×8=40  6×8=48  7×8=56  8×8=64
1×9=9   2×9=18  3×9=27  4×9=36  5×9=45  6×9=54  7×9=63  8×9=72  9×9=81
```

图 7.19　for 语句实现输出《九九乘法歌诀》表

实现的代码如下所示：

```
//初始化一个 Bitmap 对象，设置画布大小
System.Drawing.Bitmap image = new System.Drawing.Bitmap(600,250);
Graphics g = Graphics.FromImage(image);                   //实例化 Graphics 对象
try
{
    int x = 5;                                            //x 坐标
    int y = 50;                                           //y 坐标
g.Clear(Color.White);                                     //清空图片背景色
     //实例化 Font 对象，设置画笔
    Font font = new System.Drawing.Font("Arial", 12, (System.Drawing.FontStyle.
```

```
    Bold));
    System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing.
    Drawing2D. Linear- GradientBrush(new Rectangle(0, 0, image.Width, image.
    Height), Color.Blue, Color.DarkRed, 1.2f,  true);      //设置笔刷
    g.DrawString("九九乘法歌诀", font, brush,200, 5);        //绘制标题
    g.DrawLine(new Pen(Color.Silver), 0, 30, 600, 30);     //绘制横线
    for (int i = 1; i <= 9; i++)              //使用 for 语句嵌套实现打印九九乘法歌诀
    {
        for (int j = 1; j <= i; j++)
        {
            string str = j + "×" + i + "=" + j * i + "  "; //设置绘制的字符串
            g.DrawString(str, font, brush,x, y);           //绘制字符串
            x = x + str.Length + 60;                       //设置每列的间隔
        }
        y = y + 20;                                        //设置行的间隔
        x = 5;                                             //恢复 x 坐标的默认值
    }
    //画图片的边框线
    g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.Height - 1);
    System.IO.MemoryStream ms = new System.IO.MemoryStream(); //创建一个内存流
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                              //将绘制的图片保存到内存流中
    Response.ClearContent();
    Response.ContentType = "image/Gif";                    //设置图片格式
    Response.BinaryWrite(ms.ToArray());                    //输出图片
}
finally
{
    g.Dispose();
    image.Dispose();
}
```

学习手册 注意 在输出某一项后要设置它与第二项之间的间距，包括列间距和行间距，否则《九九乘法歌诀》都绘制到一起了。

7.2.4 foreach 语句

foreach 语句用于循环列举一个集合的元素，并对该集合中的每个元素执行一次相关的语句，其基本格式如图 7.20 所示。

语　法	示　例

foreach(类型 变量 in 集合类型表达式)
{
　语句块
}
	对不起我爱你，一万句我爱你表达我的心
	string[] loves={我爱你，我爱你....我爱你};
	foreach (string s in loves)
	{
	　输出s
	}

图 7.20　foreach 语句基本格式

变量的类型一定要与集合类型相同，例如，如果想遍历一个字符串数组中的每一项，那么此处变量的类型就应该是 string 类型，以此类推。

foreach 语句的执行流程如图 7.21 所示。

【例 7.9】 初始化一个字符串数组，然后使用 foreach 语句遍历数组，并输出数组中的每一项，代码如下所示：

```
string[] files = new string[6] {"红尘倒影","洪杰","玲玲","小剑","小科"};
                                        //创建并初始化数组
foreach(string s in files)              //foreach 语句遍历数组
{
    Response.Write(s+"  ");             //输出数组中的内容
}
```

实例位置：光盘\MR\Instance\7\7.8

【例 7.10】 foreach 语句在开发中使用的几率还是比较高的，例如，使用 foreach 语句遍历窗体中所有控件或遍历指定文件夹下的所有文件。本例使用 foreach 语句遍历窗体中所有控件并清空窗体中所有文本框，效果如图 7.22 所示。

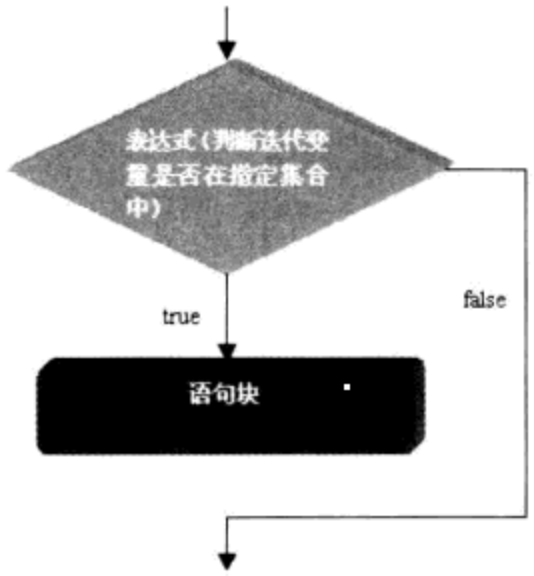


图 7.21　foreach 语句的执行流程

图 7.22　遍历页面中控件，并清空文本框

实现的代码如下所示：

```
for(int i=0;i<Page.Controls.Count;i++)     //使用 for 语句遍历窗体中的所有控件
{
    foreach(Control c in Page.Controls[i].Controls)
                                           //使用 foreach 语句遍历窗体中的控件集合
    {
        if (c is TextBox)                  //判断当前遍历的控件是否是 TextBox 控件
        {
            TextBox tb = (TextBox)c;       //如果是 TextBox 控件则创建一个 TextBox 对象
            tb.Text = "";                  //将该对象的 Text 属性设为空
        }
    }
}
```

当判断 c 是 TextBox 控件以后，要根据 c 创建一个 TextBox 对象。此时，需要将 c 强制转换为 TextBox 对象。

7.3 实 战 练 习

7.3.1 实现简单的会员注册功能

➤➤➤题目描述

用流程语句开发一个简单的会员注册功能，通过流程语句判断用户输入的信息是否符合规则，如果输入无误，则弹出用户的注册信息，效果如图 7.23 所示。

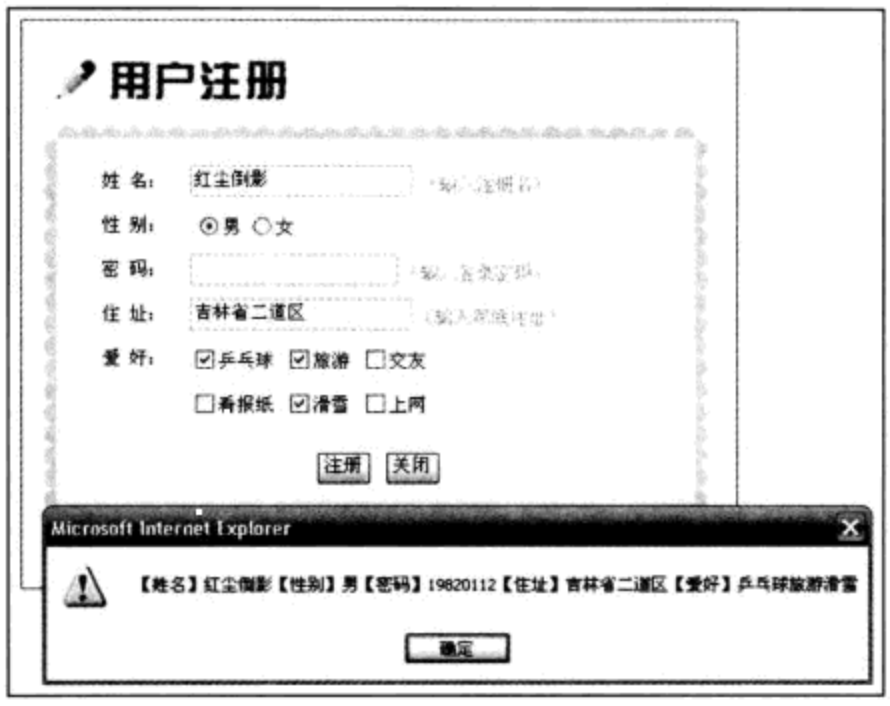


图 7.23 会员注册界面

➤➤➤技术指导

使用流程语句开发会员注册功能，其中包括注册用户的“姓名”、“性别”、“密码”、“住址”和“爱好”等信息，读者可以参考“会员注册流程图”进行开发，如图 7.24 所示。

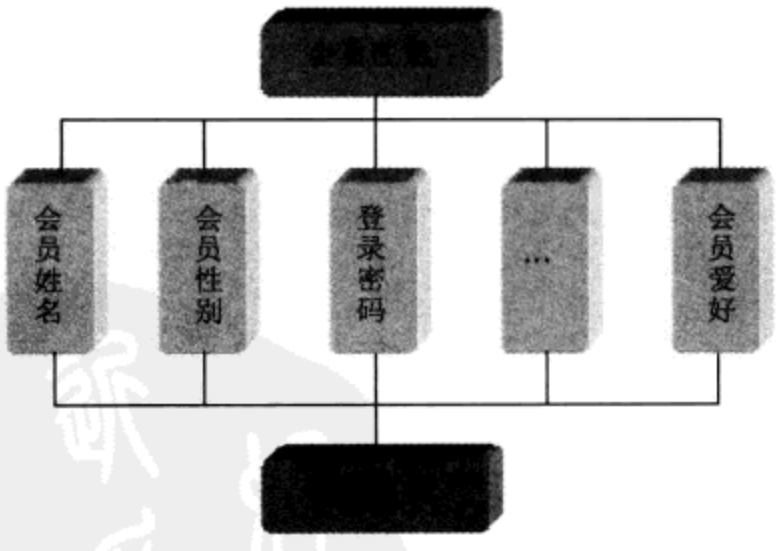


图 7.24 会员注册流程图

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取

本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/7.3.1**

7.3.2　遍历指定文件夹

➤➤➤题目描述

遍历指定文件夹中的所有.txt 文件时，使用了 if 语句之间的嵌套。通过本实例，可以将指定文件夹下的所有文件遍历出来，遍历出来的是文件的全路径，包括文件名和扩展名等。实例运行结果如图 7.25 所示。

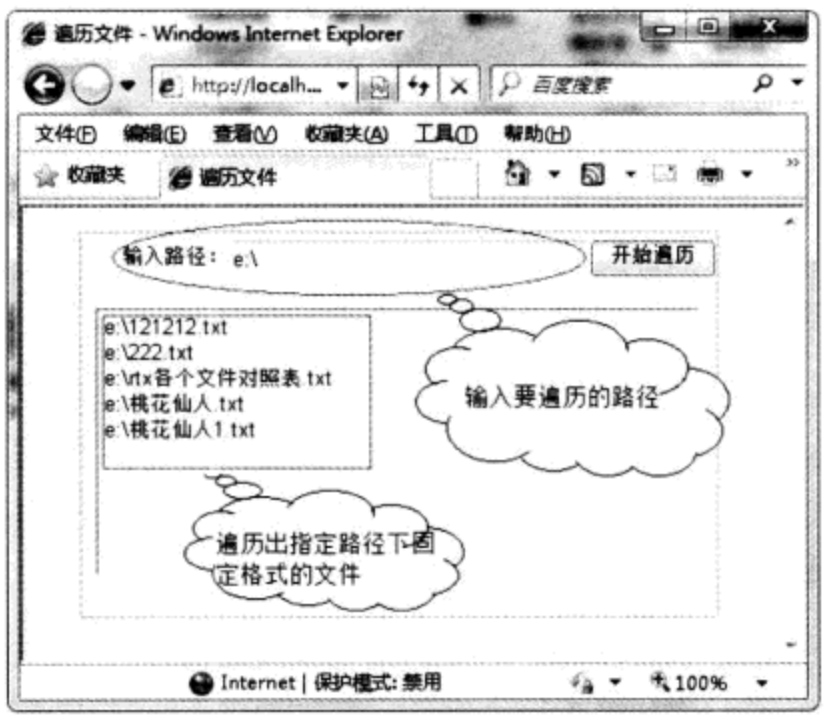


图 7.25　遍历指定文件夹下的所有.txt 文件

➤➤➤技术指导

本实例的关键技术是首先通过 System.IO.Directory.GetFiles 方法获取指定目录下的所有文件，其返回值是字符串数组。然后使用 for 语句或者 foreach 语句遍历这个数组，判断数组中存储的文件扩展名是否是 txt，如果是 txt 格式则添加到 ListBox 控件中，关键代码如下所示：

```
string[] files = System.IO.Directory.GetFiles(txtPath.Text.Trim());
                                                    //获取文件夹中所有文件
for (int i = 0; i < files.Length; i++)              //遍历文件夹中所有文件的数组
{
    System.IO.FileInfo fi = new System.IO.FileInfo(files[i]);
                                                    //实例化 FileInfo 对象
    if (fi.Extension == ".txt")                     //获取文件扩展名,判断是否是.txt 文件
    {
        lbinfo.Items.Add(files[i]);                 //将文件全路径添加到 ListBox 控件中
    }
}
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/7.3.2**

7.3.3 递归算法的经典面试题

图 7.26 使用递归算法的经典面试题

➤➤➤题目描述

在编程的世界中，充斥着大量的数据，怎样方便有效地处理数据呢？这时算法就显得尤为重要了，下面的实例中介绍了一个经典的面试题，有一组数 1、1、2、3、5、8、13、21、34…要求用递归算法算出这组数的第 30 个数是多少？实例运行结果如图 7.26 所示。

➤➤➤技术指导

本实例实现时主要用到了递归算法，下面对其进行详细讲解。递归算法在程序设计中被广泛应用，递归是方法在运行中直接或间接调用自身而产生的重入现象，它是计算机学科的一个重要概念，而且在算法的世界中占有重要的位置，使用递归方法可以使代码变得更加简洁和清晰。在生活中，我们可以从视觉的角度了解递归的概念，准备两面平行的反光镜，人站在两面反光镜的中间，向其中的一面反光镜看，会看到自己和反光镜，而反光镜中还有自己和反光镜，就是这样一直递归下去。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/7.3.3**

7.4 本 章 小 结

本章详细介绍在程序当中有效使用分支语句、循环语句的方法，语句是程序完成一次操作的基本单位，而流程语句控制语句执行的顺序，在讲解流程语句过程中，通过实例演示每种语句的用法；学习本章时，读者要重点掌握 if 语句、switch 语句、for 语句、while 语句的用法，因为流程控制语句在程序开发中会经常用到，希望通过对本章的学习，读者能够对流程控制语句有一个深刻的认识。

第 2 篇

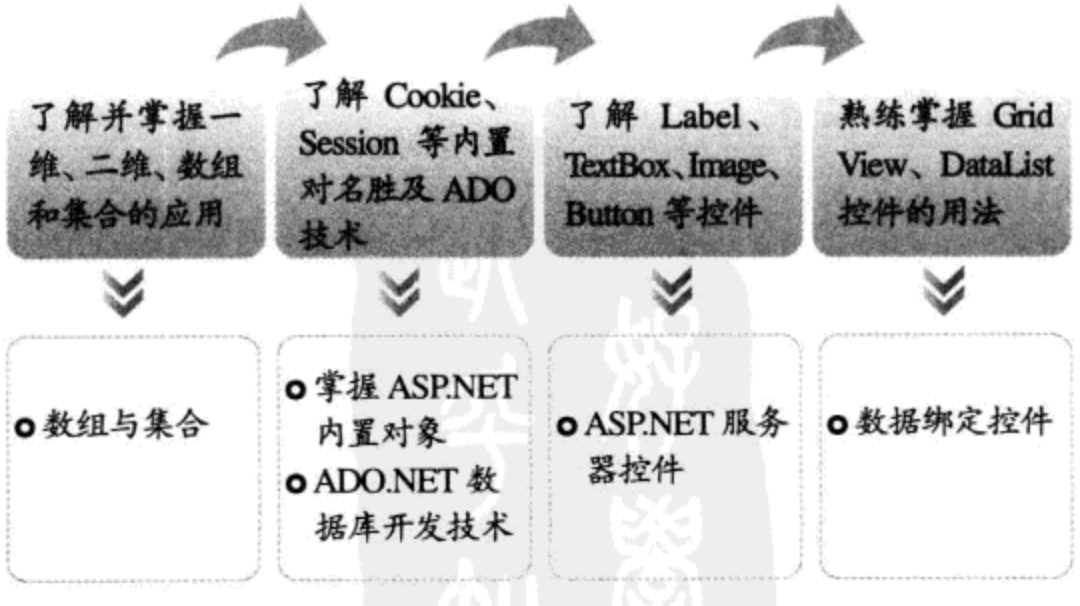
核心篇

书山有路勤为径
学海无涯苦作舟

本篇主要内容：

- 第 8 章　数组与集合
- 第 9 章　掌握 ASP.NET 内置对象
- 第 10 章　ADO.NET 数据库开发技术
- 第 11 章　ASP.NET 服务器控件
- 第 12 章　数据绑定控件

本篇学习流程：



第 8 章

数组与集合

（名师课堂：52 分钟）

C#程序设计中，数组与字符串一样，是最常用的类型之一，数组能够按一定规律把相关的数据组织在一起，并能通过“索引”或“下标”快速地管理这些数据；另外，C#中还提供了 ArrayList 集合，也可以存储多个数据，本章也将对其进行详细讲解。本章将从初学者的角度考虑，详细讲解数组和集合，讲解过程中为了便于读者理解，结合了大量的实例。通过本章的学习，读者可以掌握以下知识：

- 了解数组的基本概念
- 掌握一维数组和二维数组的声明及使用
- 掌握数组的各种基本操作
- 掌握常用的数组排序算法
- 掌握 ArrayList 集合类的使用及操作

PDG

8.1　了解数组从何而来

数组是包含若干相同类型的变量的集合。这些变量可以通过索引进行访问。数组的索引从 0 开始。数组中的变量称为数组的元素。数组中的每个元素都具有唯一的索引与其对应。数组主要包括一维数组、二维数组等，下面将会详细介绍其概念及用法。

8.1.1　数组概念

在日常生活中，我们经常使用成组的容器来存放东西，比如超市的储物柜、家里的书架及抽屉等，这些容器都由很多相同的容器组成，可以存放大量类似的东西，在 C#中也有类似的存储结构——数组。

数组是相同类型的对象的集合。由于数组几乎可以为任意长度，因此可以使用数组存储数千乃至数百万个对象，但必须在创建数组时就确定其大小。数组中的每项都按索引进行访问，索引是一个数字，指示对象在数组中的存储位置。

数组类型继承于抽象基类 System.Array 而后者又继承于 System.Object，所以数组类型是引用类型，它被分配在托管堆内。数组既可用于存储引用类型，也可用于存储值类型。可以使用 new 操作符来创建一个数组对象实例。

8.1.2　数组的用途

数组的使用非常广泛，在处理大量数据时，数组的确是很好的选择，数组是集合的一个子集。在 C#中数组几乎与字符串--样常用。而在本书后面章节中介绍的 Windows 窗体编程中很多 Windows 控件内部都使用了数组或集合，第 12 章介绍的数据库开发中也会相应地用到数组和集合，读者朋友听到这里一定迫不及待了吧，一起来体验数组给我们带来的方便、快捷与高效吧！

8.2　走进一维数组

专题讲座：光盘\MR\Video\8\一维数组.exe

➤➤➤视频速递：通过本视频读者可以对一维数组的概念和应用有所了解。

本节的重点内容是使读者能够了解什么是一维数组，并且掌握如何创建、初始化和遍历一维数组。一维数组在软件开发中的应用相当频繁，所以希望读者能够认真学习本节的内容。

8.2.1 创建一维数组

数组可以分为一维数组、二维数组、三维数组及多维数组。其中一维数组是最常用的，而且也是经过了特殊优化的。后面的小节中会详细介绍怎样使用 for 和 foreach 循环来遍历数组。一维数组的声明格式如图 8.1 所示。

从图 8.1 中可以看到，一维数组的声明包括了数组存储的数据类型和一个中括号[]还有数组的名称，这个声明语句已经声明了数组的类型及引用数组的名称 Name，前面已经介绍过，数组类型隐式继承于 System.Array 抽象类，而 System.Array 又继承自 System.Object，所以数组是引用类型，现在我们再使用 new 操作符来得到一个数组对象，如图 8.2 所示。

图 8.1 一维数组的声明格式

图 8.2 创建一维数组对象

从图 8.2 中可以看到，操作中使用 new 操作符创建了一个数组对象，数组对象的元素数量是 3，元素的类型为 string。现在数组的对象已经创建完成了，可以使用数组对象了，但是数组内还没有数据，要怎样才可以向数组中添加数据呢，我们在 8.22 节中进行详细介绍。

8.2.2 一维数组初始化

本节主要讲解一维数组的初始化，可以在创建数组对象时对其进行初始化，也可以在创建数组对象后使用 for 循环对其进行初始化。创建数组对象时对数组初始化一般用于数组内部元素数量较少的情况，所以在创建数组对象时，在图 8.2 的基础上稍作更改就可以实现对数组进行初始化，如图 8.3 所示。

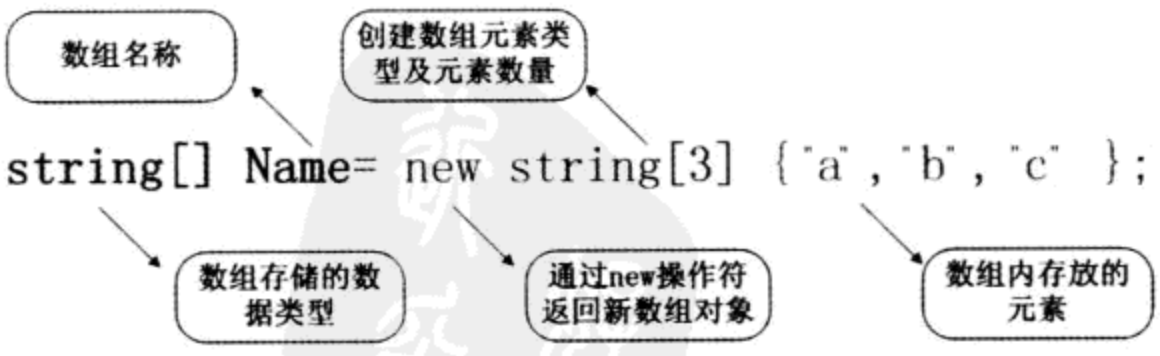


图 8.3 创建一维数组对象并初始化

从图 8.3 中可以看到大括号中包括了 3 个字符串元素，这 3 个元素已经被初始化到 Name 数组中。

string[] Name=new string[3]{"a","b","c"} 中数组的大小必须要与大括号中的元素个数相同，否则会产生编译错误。如数组中的元素是 3 个则大括号内的字符串数量也要是 3 个。

在定义数组并对数组进行初始化时也可以不使用 new 操作符，简写如下：

```
string[] Name = {"a","b","c" };
```

上面的代码与图 8.3 执行后的结果是一样的，都是 Name 数组中有 3 个字符串元素分别为"a"、"b"和"c"。在 8.2.3 节中会介绍使用 for 循环来遍历数组、对数组进行初始化或取值操作。

8.2.3　遍历一维数组

前面介绍了在创建数组对象时对数组进行初始化，现在我们来建立一个元素数量为 200 的数组。如果还是在创建数组对象时对数组进行初始化的话要手动写出 200 个字符串会很累，所以这一次使用 for 循环遍历数组并对其进行初始化。

图 8.4　遍历数组并输出

实例位置：光盘\MR\Instance\8\8.1

【例 8.1】 创建一个网站，在后台代码中首先创建一个一维数组，然后通过 for 语句初始化数组。最后使用 for 语句遍历这个一维数组，并将数组中各元素输出到页面中，运行结果如图 8.4 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] Name;                                    //定义数组类型变量 Name
    Name = new string[200];                           //数组变量 Name 得到新的数组对象引用
    for (int i = 0; i < 200; i++)                     //开始 for 循环
    {
       Name[i] = i.ToString();                        //对数组元素进行赋值
    }
    for (int j = 0; j < Name.Length; j++)             //遍历数组
    {
        Response.Write(Name[j].ToString()+",");       //输出数组元素
    }
}
```

Name[i]中的 i 值为 Name 数组的索引。可以使用索引设置或获取数组中的元素。格式为 string s= Name[0]; 即获取 Name 数组中的第 1 个元素给字符串变量 s。Name[0]="abc"; 更改 Name 数组中的第 1 个元素的字符串为"abc"。一般情况下数组的索引都是从 0 开始依次向后排，如建立了 200 个元素的数组，那么元素的索引就是从 0 到 199，如图 8.5 所示（由于没有初始化数组，所以数组内元素的引用都为空），在实例 8.1 中使

用循环为 Name 数组的每一个元素赋值，赋值后如图 8.6 所示。

索引	数组元素的值
0	NULL
1	NULL
2	NULL
3	NULL
…	NULL
199	NULL

图 8.5　一维数组的结构图

索引	数组元素的值
0	"0"
1	"1"
2	"2"
3	"3"
…	"…"
199	"199"

图 8.6　一维数组赋值后的结构

8.2.4　情景应用：尝试使用 foreach 语句遍历数组

专题讲座：光盘\MR\Video\8\foreach 语句遍历一维数组.exe

➤➤➤视频速递：通过本视频读者可以对一维数组的概念和应用有所了解。

实例位置：光盘\MR\Instance\8\8.2

【例 8.2】 定义并初始化一个一维数组，然后使用 foreach 语句遍历该数组，并将结果显示在页面中，效果如图 8.7 所示。

图 8.7　初始化数组和遍历数组

实现代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
    {
        string[] str = { "红尘倒影", "洪杰", "玲玲", "小剑", "小科", "小强" };
                                                        //创建并初始化数组
        Response.Write("本月优秀员工：<br>");
        foreach (string s in str)                       //foreach 语句遍历数组
        {
            Response.Write(s + "<br>");                 //输出数组中的内容
        }
    }
```

8.3　迈向二维数组

专题讲座：光盘\MR\Video\8\二维数组.exe

➤➤➤视频速递：通过本视频读者可以对二维数组的概念和应用有所了解

通过以上章节的学习，读者已经能够熟练掌握一维数组的应用。那么，从本节开始学习什么是二维数组，以及如何创建、初始化和遍历二维数组。

8.3.1　创建二维数组

用抽象的比喻形容一维数组是一条线的话，那么二维数组就是一个表格。我们可以对

比一下图 8.8 一维数组的存储结构与图 8.9 二维数组的存储结构。

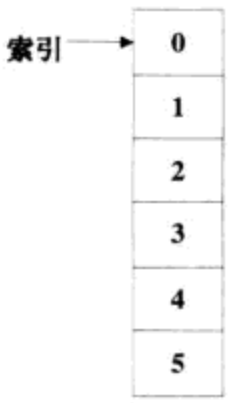


图 8.8　一维数组的存储结构

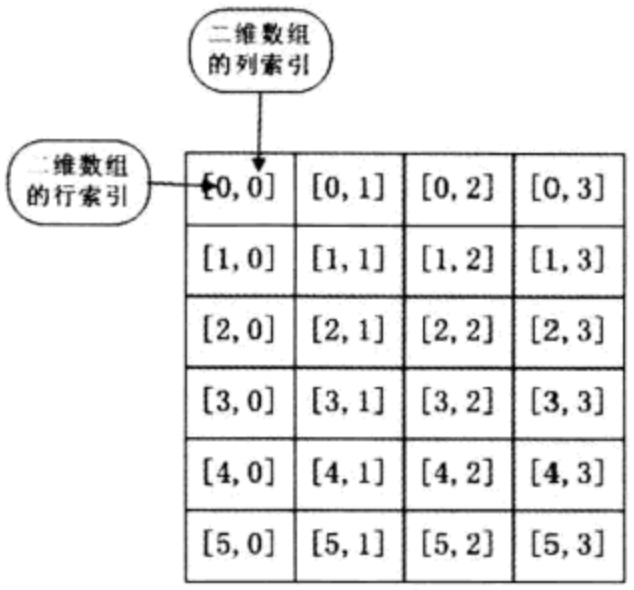


图 8.9　二维数组的存储结构

从图 8.8 中可以看到一维数组中共有 6 个元素，元素的索引从 0 到 5。

学习手册 注意　我们也可以显式建立起始索引非 0 的数组，非 0 基数组不太常用，而且遍历非 0 基数组的效率不高，所以在本文中不会对非 0 基数组做过多介绍。

从图 8.9 中可以看到二维数组更像是一个表格，表格中的数字如：[0,0]标明了每一个数组元素在二维数组中的索引。现在仔细分析一下图 8.9 中的二维数组，它包括 6 行 4 列，共有 24 个元素，每一个数组元素都包括了行索引和列索引。[0,0]中第一个 0 是行索引，第二个 0 是列索引，在表格中第一行所有数组元素的行索引都为 0，第二行所有数组元素的行索引都为 1，同样的，表格中第一列所有数组元素的列索引都为 0，第二列所有元素的列索引都为 1。以此类推可以更好地理解二维数组的索引。

在前面已经简单了解了二维数组的索引方式，现在开始创建二维数组。

二维数组的声明格式如图 8.10 所示。

图 8.10 中二维数组的声明格式与一维数组差不多只是多了一个逗号“，”，由于数组是引用类型，我们可以使用 new 操作符创建一个二维数组的对象，如图 8.11 所示。

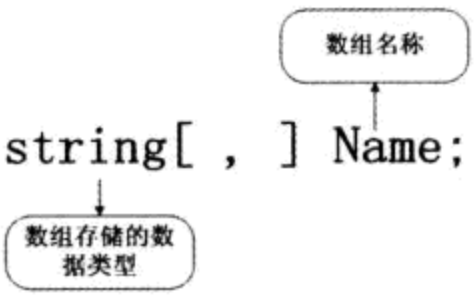


图 8.10　二维数组的声明格式

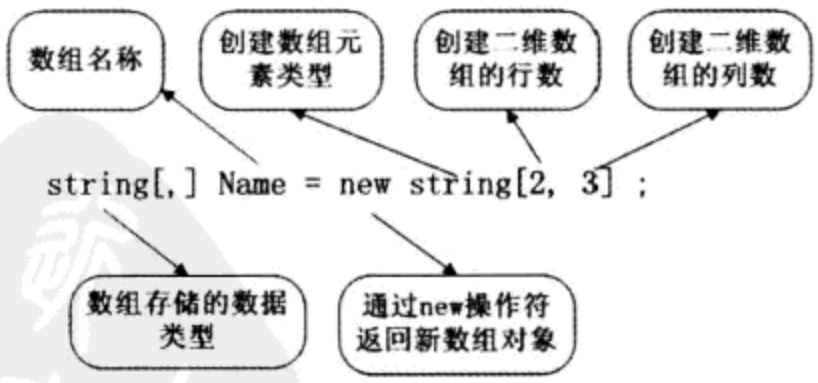


图 8.11　创建二维数组对象

学习手册 注意　图 8.11 中使用 new 操作符创建了一个 2 行 3 列的二维数组对象，该二维数组对象还没有被初始化。在 8.3.2 节中会详细介绍二维数组的初始化。

8.3.2 初始化二维数组

我们在创建二维数组时可以同时对二维数组初始化，也可以在创建二维数组对象后使用 for 循环对数组进行初始化。图 8.12 演示了创建二维数组对象并初始化。

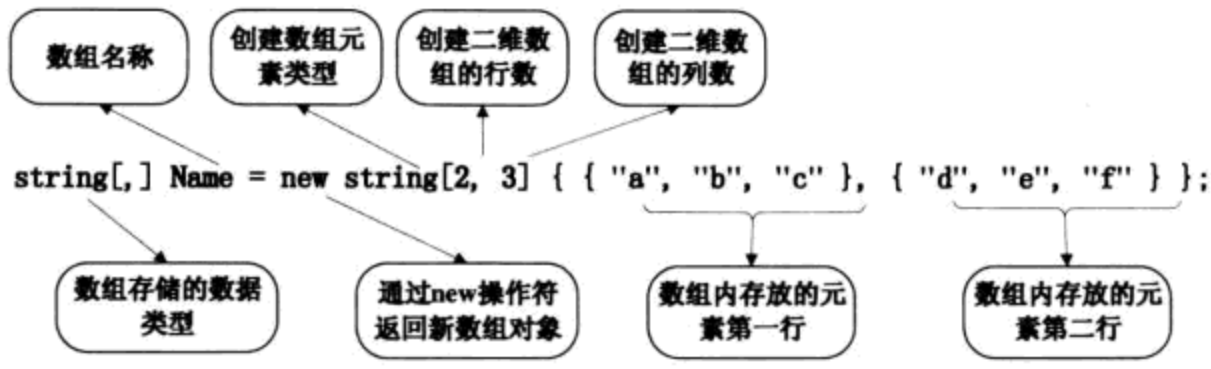


图 8.12 创建二维数组对象并初始化

从图 8.12 中可以看到建立了一个 2 行 3 列的二维数组，并完成了二维数组的初始化。在二维数组初始化部分中，最外层的大括号中包含两个大括号（也就是二维数组中的 2 行），其中每一个大括号内的数据都是这个二维数组中的一行，而每一个大括号内又有 3 个字符串，这 3 个字符串对应着二维数组中的每一列。完成二维数组的初始化后图 8.13 演示了二维数组内元素的存储结构。

学习手册 提示 图 8.13 中数组对象已经创建，并已经进行初始化，我们可以在代码中根据数组元素的索引来访问数组元素，如图 8.14 所示。

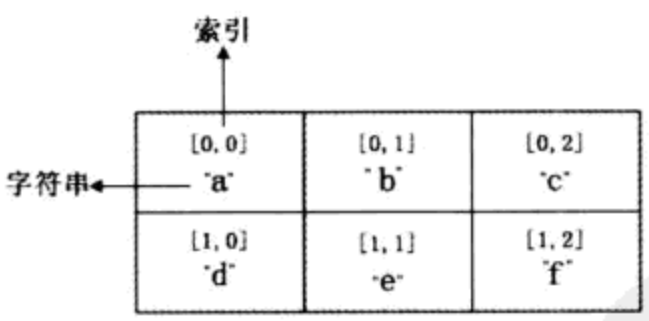


图 8.13 二维数组内元素的存储结构

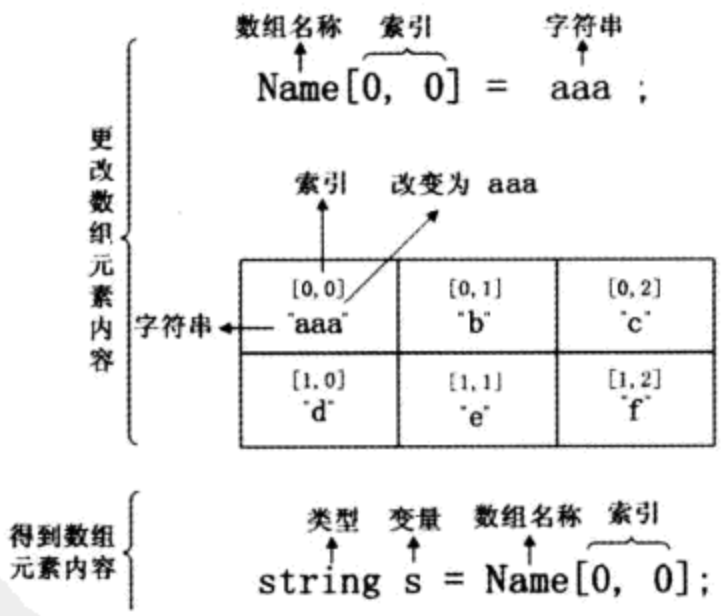


图 8.14 根据数组元素的索引来访问数组元素

下面通过一个实例演示如何创建并初始化二维数组。

实例位置：光盘\MR\Instance\8\8.3

【例 8.3】 创建网站，在后台代码中声明一个二维数组，并初始化该二维数组，然后在页面中输出指定索引的值，如图 8.15 所示。

实现的代码如下所示：

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

ÓÉ Foxit PDF Editor ±à¼-
°æÈ¨ËùÓÐ (c) by Foxit Software Company, 2004
½öÓÃÓÚÆÀ¹À¡£

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ⇕

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 ›

HTTP://WLSAM168.400GB.COM

图 8.15　初始化二维数组并获取数组中内容

```
{
    //定义二维数组并初始化
    string[,] Name = new string[2, 3] { { "小张", "Java", "软件工程师" }, {"小剑",
    "VB", "软件工程师" } };
    string name = Name[0, 0];              //根据二维数组对象的索引获取数组元素的字符串
    string dep = Name[0, 1];
    string job = Name[0, 2];
    Response.Write("Java 软件工程师：<br>");
    Response.Write("姓名：" +name);
    Response.Write("<br>部门：" +dep);
    Response.Write("<br>职务：" + job);
}
```

输出二维数组中的内容

本节我们介绍了创建二维数组并对数组进行初始化，这种情况适合于数组内元素数量较少的情况，如果数组中元素数量较多，我们对数组进行初始化就要用到 for 循环，在 8.3.3 节中将会介绍使用 for 循环初始化二维数组。

8.3.3　遍历二维数组

在 8.2.3 节中我们已经能够使用 for 循环来遍历一维数组，本节我们将使用嵌套的 for 循环来遍历和初始化二维数组，通过下面的例子读者能够掌握如何遍历二维数组。

实例位置：光盘\MR\Instance\8\8.4

【例 8.4】 创建一个网站，在后台代码中首先创建一个二维数组，然后通过嵌套的 for 语句初始化此二维数组，最后再通过嵌套的 for 语句遍历二维数组，并将数组中的元素输出到页面中，运行结果如图 8.16 所示。

图 8.16　遍历二维数组

实现的代码如下所示：

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

```
    string[,] Names = new string[6, 4];          //创建数组对象
    for (int i = 0; i < 6; i++)                  //开始第一层 for 循环
    {
       for (int j = 0; j < 4; j++)               //开始第二层 for 循环
        {
            Names[i, j] = "[" + i.ToString() + "," + j.ToString() + "]  ";
                                                 //初始化二维数组中的元素
        }
    }
    for (int i = 0; i < 6; i++)                  //开始第一层 for 循环
    {
        for (int j = 0; j < 4; j++)              //开始第二层 for 循环
        {
            Response.Write(Names[i,j]+"<br>");   //在控制台输出二维数组中的所有元素
        }
    }
}
```

8.4 掌握数组的基本操作

专题讲座：光盘\MR\Video\8\数组的基本操作.exe

➤➤➤视频速递：通过本视频读者可以掌握数组的基本操作。

System.Array 类中包含用来操作数组（比如查找和合并）的各种方法，本节将向读者介绍数组最常用的一些基本操作。

8.4.1 清空数组

数组不只有着批量数据存储能力，它还有着强大的批量数据处理能力，可以使用多种方法清空数组内元素，本节介绍两种方法：清空数组内元素的值一种方法是使用 for 循环；另一种方法是使用 Clear 方法。

1. for 循环清空

下面通过一个实例演示如何使用 for 循环清空数组内元素的值。

实例位置：光盘\MR\Instance\8\8.5

【例 8.5】 创建一个网站，在后台代码中创建一个数组并初始化，然后使用 for 循环清空数组内所有元素的值，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] Name = new string[5] { "a", "b", "c", "d", "e" };
                                                 //定义一维数组并初始化
    for (int i = 0; i < 5; i++)                  //使用 for 循环
    {
        Name[i] = null;                          //清空数组元素内容
    }
}
```

使用 for 循环遍历数组中每一个元素，并清空元素的值。

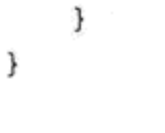
2. Clear 方法

上面我们已经介绍了一种清空数组的方法，下面的一种方法是使用数组基类中的 Clear 方法清空数组。

实例位置：光盘\MR\Instance\8\8.6

【例 8.6】 创建一个网站，在后台代码中创建一个数组并初始化，然后将数组对象转换为 IList 接口对象，最后调用接口对象的 Clear 方法清空数组元素的值，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] Name = new string[5] { "a", "b", "c", "d", "e" };
                                                    //定义一维数组并初始化
    System.Collections.IList il = Name;             //将数组对象转换为 IList 接口对象
    il.Clear();                                 //调用接口对象的 Clear 方法清空数组元素内数据
}
```



上面的两个例子演示清除数组内元素，实际并没有删除数组内元素的数量，只是将数组内数组元素的值清除。

8.4.2　合并数组

下面实例将会使用代码演示数组合并。

实例位置：光盘\MR\Instance\8\8.7

【例 8.7】 创建一个网站，在后台代码中定义两个数组，然后合并两个数组到第三个数组中，并输出第三个数组的元素，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] s1 = new string[5] { "a", "b", "c", "d", "e" };   //定义第一个数组
    string[] s2 = new string[2] { "f", "g" };                  //定义第二个数组
    string[] s3 = new string[s1.Length + s2.Length];           //定义第三个数组
    for (int i = 0; i < s1.Length; i++)                        //开始 for 循环
    {
        s3[i] = s1[i];                          //将第一个数组的内容添加到第三个数组中
    }
    for (int i = 0; i < s2.Length; i++)        //开始 for 循环
    {
        s3[i + s1.Length] = s2[i];              //将第二个数组的内容添加到第三个数组中
    }
    foreach (string s in s3)                   //使用迭代器
    {
        Response.Write(s + ",");                //输出合并后数组的元素
    }
}
```

程序运行结果为：a,b,c,d,e,f,g。

学习手册 注意 上述代码中第 3 个数组的长度等于第一个数组的长度与第二个数组的长度之和，所以才可以将第一个数组内的元素与第二个数组内的元素都放入第 3 个数组内。

8.4.3 拆分数组

图 8.17 拆分数组的执行结果

在 8.4.2 节中演示了合并数组，在使用数组时也可以对数组进行拆分，下面的例子演示了数组的拆分。

实例位置：光盘\MR\Instance\8\8.8

【例 8.8】 创建一个网站，在后台代码中定义一个字符串类型数组并初始化，再将字符串数组拆分为两个字符串数组，并输出两个数组的内容，运行结果如图 8.17 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] s1 = new string[] { "a", "b", "c", "d", "e", "f" };
                                                          //定义将要被拆分的数组
    string[] s2 = new string[s1.Length - 3];              //定义第一个数组
    string[] s3 = new string[s1.Length - s2.Length];      //定义第二个数组
    for (int i = 0; i < s2.Length; i++)                   //遍历第一个新数组
    {
        s2[i] = s1[i];                                    //将元素添加到第一个新数组
    }
    for (int i = 0; i < s3.Length; i++)                   //遍历第二个新数组
    {
        s3[i] = s1[i + s2.Length];                        //将剩余元素添加到第二个新数组
    }
    Response.Write("第一个数组内的元素：<br>");
    foreach (string item in s2)
    {
        Response.Write(item+"<br>");                      //输出第一个数组的元素
    }
    Response.Write("第二个数组内的元素：<br>");
    foreach (string item in s3)
    {
        Response.Write(item+"<br>");                      //输出第二个数组的元素
    }
}
```

学习手册 注意 上述代码中并没有将被拆分的数组分解成两个独立的数组，而是将被拆分的数组元素分别放入两个新创建的数组中。

8.4.4　查找数组元素

可以使用数组抽象基类 System.Array 定义的静态方法 IndexOf 查找数组内的元素。下面通过一个实例来演示如何查找数组元素。

实例位置：光盘\MR\Instance\8\8.9

【例 8.9】创建一个网站，在后台代码中创建一个数组并初始化，然后使用 System.Array 类的静态方法 IndexOf 查找数组内指定元素的索引，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] s = new string[] { "a", "b", "c", "d", "e" };      //定义数组
    int i = System.Array.IndexOf(s, "c");                       //使用静态方法查找字符串的索引
    Response.Write("c 在数组中的索引: "+i);                      //输出索引
}
```

程序运行结果为：2。

学习手册 说明 上述代码中使用 System.Array 类的静态方法 IndexOf()查找指定字符串的索引，并在控制台输出索引值。如果在数组中未能找到指定字符串，那么 IndexOf 方法将返回索引值-1。System.Array 类的静态方法 IndexOf()提供了很多方法的重载，可以支持从指定索引位置开始执行查找。也可以使用 System.Array 类的静态方法 LastIndexOf 方法从数组的最后索引向前查找。

8.4.5　数组排序

排序是编程中最常用的算法之一，排序的方法有很多种，在实际开发程序时，可以用算法对数组进行排序，也可以用 Array 类的 Sort 方法和 Reverse 方法对数组进行排序。本节将详细讲解如何对数组进行排序。

1．冒泡排序法

冒泡排序是一种最常用的排序方法，其过程很简单，就像气泡一样越往上走越大，因此被人们形象地称为冒泡排序法。冒泡排序的过程很简单，首先将第一个记录的关键字和第二个记录的关键字进行比较，若为逆序，则将两个记录交换，然后比较第二个记录和第三个记录的关键字，依次类推，直至第 n-1 个记录和第 n 个记录的关键字进行比较为止，上述过程称为第一趟冒泡排序，执行 n-1 次上述过程后，排序即可完成。

实例位置：光盘\MR\Instance\8\8.10

【例 8.10】 创建一个网站，在后台代码中创建并初始化一个数组，然后使用冒泡法对数组中的元素从小到大进行排序，运行结果如图 8.18 所示。

图 8.18　冒泡排序

实现代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
 {
    int[] arr = new int[] { 3, 9, 27, 6, 18, 12, 21, 15 };
                                          //定义一个一维数组，并赋值
    Response.Write("初始数组：");
    foreach (int m in arr)                //循环遍历定义的一维数组，并输出其中的元素
       Response.Write(m + " ");
    Response.Write("<br>");
    //定义两个 int 类型的变量，分别用来表示数组下标和存储新的数组元素
    int j, temp;
    for (int i = 0; i < arr.Length - 1; i++)      //根据数组下标的值遍历数组元素
    {
       j = i + 1;
    id:                                   //定义一个标识，以便从这里开始执行语句
       if (arr[i] > arr[j])               //判断前后两个数的大小
       {
          temp = arr[i];                  //将比较后大的元素赋值给定义的 int 变量
          arr[i] = arr[j];                //将后一个元素的值赋值给前一个元素
          arr[j] = temp;                  //将 int 变量中存储的元素值赋值给后一个元素
          goto id;                        //返回标识，继续判断后面的元素
       }
       else
          if (j < arr.Length - 1)         //判断是否执行到最后一个元素
          {
             j++;                         //如果没有，则再往后判断
             goto id;                     //返回标识，继续判断后面的元素
          }
    }
    Response.Write("排序后的数组：");
    foreach (int n in arr)                //循环遍历排序后的数组元素并输出
       Response.Write(n + " ");
 }
```

2. 直接插入排序法

直接插入排序是一种最简单的排序方法，其基本操作是将一个记录插入到已排好序的有序表中，从而得到一个新的、记录数增 1 的有序表，然后再从剩下的关键字中选取下一个插入对象，反复执行直到整个序列有序。

图 8.19　直接插入排序法

实例位置：光盘\MR\Instance\8\8.11

【例 8.11】 创建一个网站，在后台代码中创建并初始化一个数组，然后使用直接插入法对数组中的元素从小到大进行排序，运行结果如图 8.19 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int[] arr = new int[] { 3, 9, 27, 6, 18, 12, 21, 15 }; //定义一个一维数组，并赋值
```

```
    Response.Write("初始数组：");
    foreach (int n in arr)                    //循环遍历定义的一维数组，并输出其中的元素
        Response.Write(string.Format("{0}", n + " "));
    Response.Write("<br>");
    for (int i = 0; i < 8; ++i)               //循环访问数组中的元素
    {
        int temp = arr[i];                    //定义一个 int 变量，并使用获得的数组元素值赋值
        int j = i;
        while ((j > 0) && (arr[j - 1] > temp)) //判断数组中的元素是否大于获得的值
        {
            arr[j] = arr[j - 1];                        //如果是，则将后一个元素的值提前
            --j;
        }
        arr[j] = temp;                        //最后将 int 变量存储的值赋值给最后一个元素
    }
    Response.Write("排序后的数组：");
    foreach (int n in arr)                    //循环访问排序后的数组元素并输出
        Response.Write(string.Format("{0}", n + " "));
}
```

3. 选择排序法

选择排序的基本思想是，每一趟在 n 个记录中选取关键字最小的记录作为有序序列的第 I 个记录，并且令 I 从 1 至 n-1，进行 n-1 趟选择操作。

实例位置：光盘\MR\Instance\8\8.12

【例 8.12】 创建一个网站，在后台代码中创建并初始化一个数组，然后使用选择排序法对数组中的元素从小到大进行排序，运行结果如图 8.20 所示。

图 8.20　选择排序法

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int[] arr = new int[] { 3, 9, 27, 6, 18, 12, 21, 15 };//定义一个一维数组，并赋值
    Response.Write("初始数组：");
    foreach (int n in arr)                    //循环遍历定义的一维数组，并输出其中的元素
        Response.Write(string.Format("{0}", n + " "));
    Response.Write("<br>");
    int min;                                  //定义一个 int 变量，用来存储数组下标
    for (int i = 0; i < arr.Length - 1; i++)  //循环访问数组中的元素值（除最后一个）
    {
        min = i;                                        //为定义的数组下标赋值
        for (int j = i + 1; j < arr.Length; j++)//循环访问数组中的元素值（除第一个）
        {
            if (arr[j] < arr[min])                      //判断相邻两个元素值的大小
                min = j;
        }
        int t = arr[min];                     //定义一个 int 变量，用来存储比较大的数组元素值
        arr[min] = arr[i];                    //将小的数组元素值移动到前一位
        arr[i] = t;                           //将 int 变量中存储的较大的数组元素值向后移
    }
    Response.Write("排序后的数组：");
```

```
    foreach (int n in arr)                          //循环访问排序后的数组元素并输出
        Response.Write(string.Format("{0}", n + " "));
}
```

8.5 使用 ArrayList 集合

专题讲座：光盘\MR\Video\8\ ArrayList 集合.exe

➤➤➤视频速递：通过本视频读者可以对 ArrayList 集合的概念和应用有所了解。

相同的对象组合在一起就叫做集合，日常生活中我们经常用到集合，比如读者手里拿的书就是一个集合，书中的每一个页面相当于集合中的每一个元素，页面的页码相当于集合中元素的索引。如果再抽象一点比喻，学校的班级也是一个集合，所有的学生都是这个集合中的元素；一个公司也是一个集合，所有的员工都是这个集合中的元素。本节将详细描述集合的概念、成员及应用。

8.5.1 ArrayList 集合概述

ArrayList 动态数组也叫数组列表，它之所以被称做动态数组，是因为 ArrayList 的长度是可以动态改变的。比如你现在要为某公司开发一个人事管理系统，需要使用数组来存放每一个员工的信息，这样做显然很合理，但是如果公司在某段时间内来了新员工，这时应用数组就会很麻烦，因为数组不会自动改变其长度。如果此时使用 ArrayList 就会轻松很多，ArrayList 会根据添加元素的数量自动更改自身容量的大小。这一特性将在下面两节中详细介绍。

8.5.2 ArrayList 成员

ArrayList 集合有着很多属性和方法，可以方便快捷地对集合进行操作。ArrayList 类最常用的属性及说明如表 8.1 所示。

表 8.1 ArrayList 类常用的属性及说明

属　性	说　明
Capacity	ArrayList 集合的元素容量，ArrayList 可包含的元素数
Count	ArrayList 集合的元素数量，实际包含的元素数
Item	获取或设置指定索引处的元素

ArrayList 类最常用的方法及说明如表 8.2 所示。

表 8.2 ArrayList 类常用的方法及说明

方　法	说　明
Add	将对象添加到 ArrayList 的结尾处
AddRange	向 ArrayList 添加数组元素
Insert	在 ArrayList 集合内指定索引位置添加

续表

方　法	说　明
ToArray	将 ArrayList 元素复制到指定的数组中
Remove	删除 ArrayList 某个元素
RemoveAt	根据元素的索引，在 ArrayList 集合中删除元素
Clear	清空 ArrayList 集合中的元素
IndexOf	正向查找指定元素在 ArrayList 集合中第一次出现的索引
LastIndexOf	从后向前查找指定元素在 ArrayList 集合中第一次出现的索引

8.5.3　添加 ArrayList 集合元素

向 ArrayList 集合中添加元素时，可以使用 ArrayList 类提供的 Add 方法和 Insert 方法，下面对这两种方法进行详细介绍。

1. Add 方法

该方法用来将对象添加到 ArrayList 集合的末尾处，其语法格式如下。

```
public virtual int Add (Object value)
```

参数 value：要添加到 ArrayList 的末尾处的 Object，该值可以为空引用。

返回值：ArrayList 索引，已在此处添加了 value。

【例 8.13】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Add 方法为该 ArrayList 对象添加元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr); //使用声明的一维数组实例化一个 ArrayList 对象
List.Add(7);                         //为 ArrayList 对象添加元素
```

2. Insert 方法

该方法用来将元素插入 ArrayList 集合的指定索引处，其语法格式如下。

```
public virtual void Insert (int index,Object value)
```

index：从零开始的索引，应在该位置插入 value。

value：要插入的 Object，该值可以为空引用。

【例 8.14】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Insert 方法在该 ArrayList 对象的指定索引处添加一个元素，代码如下所示。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr); //使用声明的一维数组实例化一个 ArrayList 对象
List.Insert(3, 7);                   //在 ArrayList 集合的指定位置添加一个元素
```

下面通过一个实例来演示如何使用 Add 方法和 Insert 方法。

实例位置：光盘\MR\Instance\8\8.13

【例 8.15】 创建一个网站，在后台代码中定义一个 int 类型的一维数组，并使用该数

组实例化一个 ArrayList 对象，然后分别使用 ArrayList 对象的 Add 方法和 Insert 方法向 ArrayList 集合的末尾处和指定索引处添加元素，运行结果如图 8.21 所示。

图 8.21　添加 ArrayList 集合元素

实现的代码如下所示：

```
    protected void Page_Load(object sender,
EventArgs e)
      {
          int[] arr = new int[] { 1, 2, 3, 4, 5,
6 };
          ArrayList List = new ArrayList(arr);
                                        //使用声明的一维数组实例化一个 ArrayList 对象
          Response.Write("原 ArrayList 集合: ");
          foreach (int i in List)       //遍历 ArrayList 集合并输出
          {
              Response.Write(i + " ");
          }
          Response.Write("<br>");
          for (int i = 1; i < 5; i++)
          {
              List.Add(i + arr.Length);     //为 ArrayList 集合添加元素
          }
          Response.Write("<br>使用 Add 方法添加: ");
          foreach (int i in List)       //遍历添加元素后的 ArrayList 集合并输出
          {
              Response.Write(i + " ");
          }
          Response.Write("<br>");
          List.Insert(6, 6);            //在 ArrayList 集合的指定位置添加元素
          Response.Write("使用 Insert 方法添加: ");
          foreach (int i in List)       //遍历最后的 ArrayList 集合并输出
          {
              Response.Write(i + " ");
          }
    }
```

8.5.4　删除 ArrayList 集合元素

删除 ArrayList 集合中的元素时，可以使用 ArrayList 类提供的 Clear 方法、Remove 方法、RemoveAt 方法和 RemoveRange 方法，下面对这 4 种方法进行详细介绍。

1．Clear 方法

该方法用来从 ArrayList 中移除所有元素，其语法格式如下。

```
public virtual void Clear ()
```

【例 8.16】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Clear 方法清除 ArrayList 中的所有元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.Clear();
```

2. Remove 方法

该方法用来从 ArrayList 中移除特定对象的第一个匹配项，其语法格式如下。

```
public virtual void Remove (Object obj)
```

参数 obj：要从 ArrayList 中移除的 Object，该值可以为空引用。

【例 8.17】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Remove 方法从声明的 ArrayList 对象中移除与“3”匹配的元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.Remove(3);
```

3. RemoveAt 方法

该方法用来移除 ArrayList 中指定索引处的元素，其语法格式如下。

```
public virtual void RemoveAt (int index)
```

index：要移除元素的从零开始的索引。

【例 8.18】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 RemoveAt 方法从声明的 ArrayList 对象中移除索引为 3 的元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.RemoveAt(3);
```

4. RemoveRange 方法

该方法用来从 ArrayList 中移除一定范围的元素，其语法格式如下。

```
public virtual void RemoveRange (int index,int count)
```

index：要移除元素的范围。

count：要移除元素数。

【例 8.19】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后在该 ArrayList 对象中使用 RemoveRange 方法从索引 3 处删除两个元素，代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
List.RemoveRange(3,2);
```

为了使读者更好地掌握如何删除 ArrayList 集合元素，下面通过一个实例进行演示。

实例位置：光盘\MR\Instance\8\8.14

【例 8.20】 创建一个网站，在后台代码中创建并初始化一个数组，然后用该数组创建一个 ArrayList 对象。然后使用 RemoveRange 方法删除 ArrayList 集合中的一批元素，运行结果如图 8.22 所示。

图 8.22　RemoveRange 方法删除 ArrayList 集合中的一批元素

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int[] arr = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    ArrayList List = new ArrayList(arr); //使用声明的一维数组实例化一个 ArrayList 对象
    Response.Write("原 ArrayList 集合：");
    foreach (int i in List)              //遍历 ArrayList 集合中的元素并输出
    {
        Response.Write(i.ToString() + " ");
    }
    Response.Write("<br>");
    List.RemoveRange(0, 5);              //从 ArrayList 集合中移除指定下标位置的元素
    Response.Write("删除元素后的 ArrayList 集合：");
    foreach (int i in List)              //遍历删除元素后的 ArrayList 集合并输出其元素
    {
        Response.Write(i.ToString() + " ");
    }
}
```

8.5.5 遍历 ArrayList 集合

ArrayList 集合的遍历与数组类似，都可以使用 foreach 语句，下面通过一个实例说明如何遍历 ArrayList 集合中的元素。

图 8.23 遍历 ArrayList 集合

实例位置：光盘\MR\Instance\8\8.15

【例 8.21】 创建一个网站，在后台代码中实例化了一个 ArrayList 对象，并使用 Add 方法向 ArrayList 集合中添加了两个元素，然后使用 foreach 语句遍历 ArrayList 集合中的各个元素并输出。运行结果如图 8.23 所示。

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    ArrayList list = new ArrayList();          //实例化一个 ArrayList 对象
    list.Add("ASP.NET 编程词典");               //向 ArrayList 集合中添加元素
    list.Add("ASP.NET 自学手册");
    foreach (string str in list)               //遍历 ArrayList 集合中的元素并输出
    {
        Response.Write(str+" <br>");
    }
}
```

8.5.6 查找 ArrayList 集合元素

查找 ArrayList 集合中的元素时，可以使用 ArrayList 类提供的 Contains 方法、IndexOf 方法和 LastIndexOf 方法，下面对这 3 个方法进行详细介绍。

1. Contains 方法

Contains 方法用来确定某元素是否在 ArrayList 集合中，其语法格式如下。

```
public virtual bool Contains (Object item)
```

参数 item：要在 ArrayList 中查找的 Object，该值可以为空引用。

返回值：如果在 ArrayList 中找到 item，则为 true；否则为 false。

【例 8.22】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 Contains 方法判断数字 2 是否在 ArrayList 集合中。代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
Response.Write(List.Contains(2));          //判断ArrayList集合中是否包含指定的元素
```

运行结果为 True。

2. IndexOf 方法

IndexOf 方法用来搜索指定的 Object，并返回整个 ArrayList 中第一个匹配项从零开始的索引，其语法格式如下。

```
public virtual int IndexOf(Object value)
```

参数 value：要在 ArrayList 中查找的 Object，该值可以为空引用。

返回值：如果在整个 ArrayList 中找到 value 的第一个匹配项，则为该项从零开始的索引；否则为-1。

【例 8.23】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 IndexOf 方法在 ArrayList 集合中顺序查找数字 2 的索引位置。代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
Response.Write(List. IndexOf (2));          //顺序查找数字 2 的索引位置
```

运行结果为 1。

3. LastIndexOf 方法

LastIndexOf 方法用来搜索指定的 Object，并返回整个 ArrayList 中最后一个匹配项从零开始的索引，其语法格式如下。

```
public virtual int LastIndexOf(Object value)
```

参数 value：要在 ArrayList 中查找的 Object，该值可以为空引用。

返回值：如果在整个 ArrayList 中找到 value 的最后一个匹配项，则为该项从零开始的索引；否则为-1。

【例 8.24】 声明一个包含 6 个元素的一维数组，并使用该数组实例化一个 ArrayList 对象，然后使用 LastIndexOf 方法在 ArrayList 集合中倒序查找数字 2 的索引位置。代码如下。

```
int[] arr = new int[] { 1, 2, 3, 4, 5, 6 };
ArrayList List = new ArrayList(arr);
Response.Write(List. LastIndexOf (2));     //倒序查找数字 2 的索引位置
```

运行结果为 1。

8.6 实 战 练 习

8.6.1 经典问题之约瑟夫环问题

➤➤➤题目描述

约瑟夫环问题即设有 n 个人坐成一个圈，从某个人开始报数，数到 m 的人出列，接着从出列的下一个人开始重新报数，数到 m 的人再次出列，如此反复循环，直到所有人都出列位置，最后按出列顺序输出，运行结果如图 8.24 所示。

图 8.24 经典问题之约瑟夫环问题

➤➤➤技术指导

本实例的关键技术要点就是如何理解约瑟夫环问题算法。该程序中首先自定义一个返回值类型为 int 数组类型的 Jose 方法，用来实现约瑟夫环问题的算法，该方法中有 3 个 int 类型的参数，分别用来表示总人数、开始报数的人和要出列的人，具体实现代码如下所示：

```
#region 约瑟夫环问题算法
/// <summary>
/// 约瑟夫环问题算法
/// </summary>
/// <param name="total">总人数</param>
/// <param name="start">开始报数的人</param>
/// <param name="alter">要出列的人</param>
/// <returns>返回一个 int 类型的一维数组</returns>
static int[] Jose(int total, int start, int alter)
{
    int j, k = 0;
    int[] intCounts = new int[total + 1];
                          //intCounts 数组存储按出列顺序的数据，以当结果返回
    int[] intPers = new int[total + 1];    //intPers 数组存储初始数据
    for (int i = 0; i < total; i++)
                          //对数组 intPers 赋初值,第一个人序号为 0,第二人为 1，依此下去
    {
        intPers[i] = i;
    }
    for (int i = total; i >= 2; i--)       //按出列次序依次存于数组 intCounts 中
    {
        start = (start + alter - 1) % i;
        if (start == 0)
            start = i;
        intCounts[k] = intPers[start];
        k++;
        for (j = start + 1; j <= i; j++)
            intPers[j - 1] = intPers[j];
    }
```

```
    intCounts[k] = intPers[1];
    return intCounts;                    //结果返回
}
#endregion
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/8.6.1**

8.6.2　数组之冒泡排序

➤➤➤题目描述

冒泡排序是一种最常用的排序方法，其过程很简单，就像气泡一样越往上走越大，因此被人们形象地称为冒泡排序法。本实例用 C#实现了采用冒泡排序法对一组数据进行排序。运行结果如图 8.25 所示。

图 8.25　数组之冒泡排序

➤➤➤技术指导

冒泡排序的过程很简单。首先将一个记录的关键字和第二个关键字进行比较，若为逆序，则将两个记录交换，然后比较第二个记录和第三个记录的关键字。依次类推，直至第 *n*-1 个记录和第 *n* 个记录的关键字进行过比较为止。上述过程称为第一趟冒泡排序，执行 *n*-1 次上述过程后，排序完成。

另外，本实例还应用了一个关键技术就是将数组作为参数。创建数组后，可以将它作为参数传递，也可以将它作为成员返回值接收。例如本实例中将数组冒泡的算法定义在了一个 Sort()方法中，该方法带有一个传入的字符串数组，该方法的代码如下所示：

```
static void Sort(int[] list)                          //数组作为参数
{
    int i, j, temp;
    bool done = false;
    j = 1;
    while ((j < list.Length) && (!done))              //判断长度
    {
        done = true;
        for (i = 0; i < list.Length - j; i++)
        {
            if (list[i] > list[i + 1])
            {
                done = false;
                temp = list[i];
                list[i] = list[i + 1];                //交换数据
                list[i + 1] = temp;
            }
        }
```

```
            j++;
        }
        Response.Write("排序结果如下：<br>");
        for (int d = 0; d < list.Length; d++)
            Response.Write(list[d] + "<br>");
    }
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/8.6.2**

8.6.3 数组快速排序

图 8.26 数组快速排序

➤➤➤题目描述

快速排序算法是诸多排序算法中性能较好的一种，也是很多内建排序类型所采用的算法，本实例将详细介绍如何对数组进行快速排序，实例运行结果如图 8.26 所示。

➤➤➤技术指导

快速排序是公认的最为优秀的内部排序算法之一，其实现思想很简单，并且在一般情况下性能较高。本实例中将快速排序算法设计在自定义的 QuickRun()方法中，代码如下所示：

```
/// <summary>
/// 快速排序算法
/// </summary>
/// <param name="data">排序数组</param>
/// <param name="low">排序上限</param>
/// <param name="high">排序下限</param>
static void QuickRun(int[] data, int low, int high)
{
    //简单设定中间值，并以此为一趟快排的分割点
    //注意这里是一个简单的算法，如果想对这个算法优化的话，可以采取随机的方法来获取分割点
    int middle = data[(low + high) / 2];
    int i = low;                                    //设定移动上标
    int j = high;                                   //设定移动下标
    do//直至分割出两个序列
    {
        while (data[i] < middle && i < high)        //扫描中值左边元素
            i++;
        while (data[j] > middle && j > low)         //扫描值右边元素
            j--;
        if (i <= j)                                 //找到了一对可以交换的值
        {
            int temp = data[i];
            data[i] = data[j];
```

```
            data[j] = temp;
            i++;
            j--;
        }
    } while (i<=j);
    if (j > low)                          //递归对比分割点元素都小的那个序列时行快速排序
        QuickRun(data, low, j);
    if (i < high)                         //递归对比分割点元素都大的那个序列时行快速排序
        QuickRun(data, i, high);
}
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/8.6.3**

8.7　本 章 小 结

本章首先对数组进行了详细讲解，然后介绍了 ArrayList 集合。讲解数组时，将数组分为一维数组和二维数组进行了讲解，然后对数组的各种操作，比如遍历、删除、排序、合并和拆分等进行了详细讲解。数组讲解完毕之后，又通过概述、元素的添加、删除、遍历和查找等详细讲解了 ArrayList 集合。通过本章的学习，读者应该能够熟练掌握数组、ArrayList 集合的使用，并能将其应用于实际开发中。

PDG

第 9 章

掌握 ASP.NET 内置对象

（名师课堂：1 小时 7 分钟）

ASP.NET 的内置对象是程序设计中最频繁使用的元素，它通过向用户提供基本的请求、响应、会话等处理功能实现了 ASP.NET 的绝大多数功能。这些对象主要有 Response 对象、Request 对象、Server 对象、Session 对象、Application 对象和 Cookie 对象，本章将通过实例详细介绍这些内置对象的使用方法和技巧。通过本章的学习，读者可以掌握以下知识：

- 学习 Response 对象的常用属性、方法
- 掌握 Response 对象页面导向和文件写入技术
- 学习 Request 对象的常用属性、方法
- 掌握 Request 对象获取地址栏传值及主机信息等
- 学习 Application 对象的常用集合、属性和方法
- 掌握 Application 对象实现网上在线人数统计
- 学习 Session 对象的常用集合、属性和方法
- 掌握 Session 对象存储登录信息
- 学习 Cookie 对象的常用属性、方法
- 掌握如何使用 Server 对象的各种属性和方法

9.1　程序响应对象 Response

Response 对象可形象地称之为响应对象，用于将数据从服务器发送回浏览器。为了更好地理解 Response 对象，本节具体介绍该对象的常用属性和方法及在实际开发中的典型应用。

9.1.1　Response 对象概述

Response 对象用于将数据从服务器发送回浏览器。比如说，我们去买电影票，将钱递给售票员，售票员给你电影票，如图 9.1 所示。Response 对象允许将数据作为请求的结果发送到浏览器中，并提供有关响应的信息。它可以用来在页面中输入数据、在页面中跳转，还可以传递各个页面的参数。它与 HTTP 的响应消息相对应。

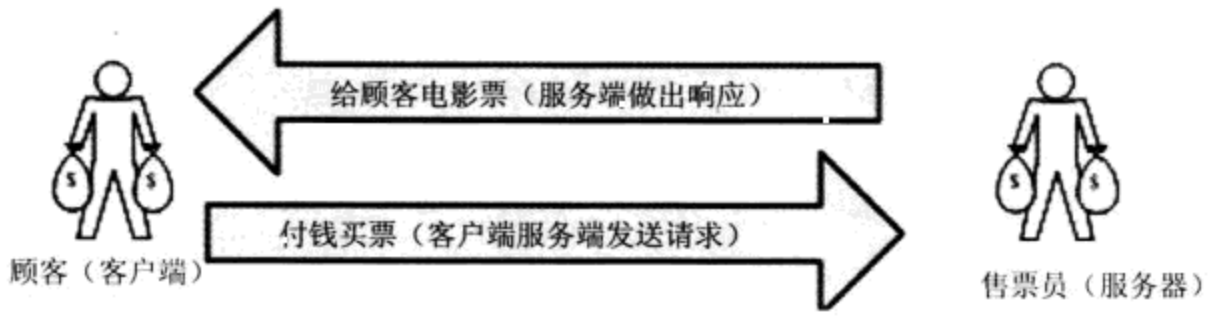


图 9.1　服务器通过 Response 对象将数据发送回客户端

9.1.2　Response 对象常用属性

Response 对象常用属性及说明如表 9.1 所示。

表 9.1　Response 对象常用属性及说明

属　性	说　明
Buffer	获取或设置一个值，该值指示是否缓冲输出，并在完成处理整个响应之后将其发送
Cache	获取 Web 页的缓存策略，如：过期时间、保密性、变化子句等
Charset	设定或获取 HTTP 的输出字符编码
Expires	获取或设置在浏览器上缓存的页过期之前的分钟数
BufferOutput	获取或设置一个值，该值指示是否缓冲输出，并在完成处理整个页之后将其发送
Cookies	获取当前请求的 Cookie 集合
IsClientConnected	传回客户端是否仍然和 Server 连接
SuppressContent	设定是否将 HTTP 的内容发送至客户端浏览器，若为 True，则网页将不会传至客户端

下面详细介绍 Response 对象的常用属性。

（1）BufferOutput 属性

获取或设置一个值，该值指示是否缓冲输出，并在完成处理整个页之后将其发送。

语法：

```
public bool BufferOutput { get; set; }
```

属性值：如果缓冲了到客户端的输出，则为 true；否则为 false。默认为 true。

【例 9.1】 通过设置 BufferOutput 属性，使页面内容缓存后再输出，程序代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.BufferOutput = true;                    //设置缓存后输出
    Response.Write("测试");                          //输出字符串
}
```

（2）IsClientConnected 属性

获取一个值，通过该值指示客户端是否仍连接在服务器上。

语法：

```
public bool IsClientConnected { get; }
```

属性值：如果客户端当前仍在连接，则为 True；否则为 False。

【例 9.2】 通过使用 IsClientConnected 属性来判断客户端是否连接在服务器上，具体代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Response.IsClientConnected)                  //判断客户端是否连接
        Response.Redirect("Default2.aspx", false);   //实现页面跳转
    else
        Response.End();                  //应用 Response.End 方法停止当前页面的执行
}
```

9.1.3 Response 对象常用方法

Response 对象常用方法及说明如表 9.2 所示。

表 9.2 Response 对象常用方法及说明

方　法	说　明
AddHeader	将一个 HTTP 头添加到输出流
AppendToLog	将自定义日志信息添加到 IIS 日志文件
Clear	将缓冲区的内容清除
End	将目前缓冲区中所有的内容发送至客户端然后关闭
Flush	将缓冲区中所有的数据发送至客户端
Redirect	将网页重新导向另一个地址
Write	将数据输出到客户端
WriteFile	将指定的文件直接写入 HTTP 内容输出流

下面详细介绍 Response 对象的一些常用方法。

（1）Redirect 方法

将网页重新导向另一个地址。

语法：

```
public void Redirect (string url)
```

参数 url：客户端应用来定位请求的资源的 URL。

在构建 Web 页面时，需要用户导向到另外一个页面，比如用户查询资料时就需要页面跳转，这时，就可以使用 Response 对象的 Redirect 方法实现。

【例 9.3】 用户从登录页面进入网站主页时，输入合法的用户名和密码之后，通过调用 Response 对象的 Redirect 方法就可以跳转到网站主页面，代码如下所示。

```
Response.Redirect("Index.aspx");
```

（2）Write 方法

Write 方法用来将数据输出到客户端。

语法：

```
Response.Write("输出字符串");
```

实例位置：光盘\MR\Instance\9\9.1

【例 9.4】 本实例演示如何通过 Write 方法将数据输出到客户端。启动 Visual Studio 2008，新建一个 Web 应用程序，默认主页为 Default.aspx。在这个新建的应用程序中，创建一个新的 Web 窗体，将其命名为 response.aspx。在 response.aspx 页面的前台代码中，输入以下代码。

```
<head runat="server">
    <title>Response test</title>
</head>
<body>
    <%
        for (int i = 0; i < 50; i++)
        {
            Response.Write("X");                    //向浏览器中写入数据
        }
    %>
</body>
```

（3）WriteFile 方法

将指定的文件直接写入 HTTP 内容输出流。

语法：

```
public void WriteFile (string filename)
```

参数 filename：要写入 HTTP 输出的文件名。

在应用程序中可借助于 Response 对象的 WriteFile 方法来实现直接输出文本的方法，此方法与地址重定向方法类似。

实例位置：光盘\MR\Instance\9\9.2

【例 9.5】 本实例演示如何通过 WriteFile 方法将文件中的内容输出到页面。运行结果如图 9.2 所示。

图 9.2　输出文件中的内容

实现的代码如下所示：

```
Response.WriteFile(Server.MapPath(@"TextFile.txt"));
```

学习手册 说明 Server.MapPath(@"TextFile.txt")的作用是获取 TextFile.txt 文件的物理路径，以便将其中的内容输出到页面中。

9.1.4 情景应用 1：页面跳转

专题讲座：光盘\MR\Video\9\页面跳转.exe

➤➤➤视频速递：理解如何在实际中应用 Response 对象。

实例位置：光盘\MR\Instance\9\9.3

【例 9.6】 通过 Response 对象的 Redirect 方法实现页面跳转并传递参数。执行程序，在 TextBox 文本中输入姓名并选择性别，单击“确定”按钮，跳转到“welcome.aspx”页，实例效果如图 9.3 和图 9.4 所示。

图 9.3 页面跳转传递参数

图 9.4 重定向的新页

实现的代码如下所示：

```
Response.Redirect("welcome.aspx?name=" + txtName.Text + "&sex=" + rblsex.
SelectedValue);
```

学习手册 注意 在 welcome.aspx 页面中，用到了 Request 对象的相关属性。关于 Request 对象我们在 9.2 节中会进行详细介绍。

9.1.5 情景应用 2：输出二进制图像

专题讲座：光盘\MR\Video\9\输出二进制图像.exe

➤➤➤视频速递：理解如何在实际中应用 Response 对象。

实例位置：光盘\MR\Instance\9\9.4

【例 9.7】 BinaryWrite 方法输出二进制图像在实际中应用还是比较广泛的，例如，在开发图形验证码时，首先生成验证码，然后绘制成图像，最后通过该方法输出到页面中。所以，读者朋友应该熟练地掌握该方法，以便为以后开发图形验证码奠定基础。本实例通过 BinaryWrite 方法输出二进制图像，运行结果如图 9.5 所示。

图 9.5　BinaryWrite 方法输出二进制图像

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 页的 Page_Load 事件中，首先定义一个字符串，然后根据字符串的长度创建一个画布 Bitmap，然后在 Bitmap 对象上绘制边框、背景颜色、背景噪音线和前景噪音点，并且将绘制后的二进制图像保存到内存流中，最后通过 Response 对象的 BinaryWrite 方法输出到浏览器中，程序代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    string checkCode = "红尘倒影"; //声明字符串并赋值
    System.Drawing.Bitmap image = new System.Drawing.Bitmap((int)Math.Ceiling
    ((checkCode.Length * 20.5)), 22);                                    //创建画板
    Graphics g = Graphics.FromImage(image);                              //创建 Graphics 对象
    try
    {
        Random random = new Random();                                    //生成随机生成器
        g.Clear(Color.White);                                            //清空图片背景色
        //画图片的背景噪音线
        for (int i = 0; i < 2; i++)
        {
            int x1 = random.Next(image.Width);
            int x2 = random.Next(image.Width);
            int y1 = random.Next(image.Height);
            int y2 = random.Next(image.Height);
            g.DrawLine(new Pen(Color.Black), x1, y1, x2, y2);
        }
        Font font = new System.Drawing.Font("Arial", 12, (System.Drawing.
        FontStyle. Bold));
        System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing.
        Drawing2D. LinearGradientBrush(new Rectangle(0, 0, image.Width, image.
        Height), Color.Blue, Color. DarkRed, 1.2f, true);
        g.DrawString(checkCode, font, brush, 2, 2);
        //画图片的前景噪音点
        for (int i = 0; i < 100; i++)
        {
            int x = random.Next(image.Width);
            int y = random.Next(image.Height);
            image.SetPixel(x, y, Color.FromArgb(random.Next()));
        }
        //画图片的边框线
        g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.
        Height - 1);
        System.IO.MemoryStream ms = new System.IO.MemoryStream();
        image.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
        Response.ClearContent();
        Response.ContentType = "image/Gif";
        Response.BinaryWrite(ms.ToArray());
    }
    finally
    {
        g.Dispose();
        image.Dispose();
    }
}
```

9.2 程序请求对象 Request

Request 对象可称为请求对象，Request 对象是 HttpRequest 类的一个实例，它提供对当前页请求的访问，其中包括标题、Cookie、查询字符串等，用户可以使用此类来读取浏览器已经发送的内容。为了更好地理解 Request 对象，本节将具体介绍该对象的常用属性和方法及在实际开发中的典型应用。

9.2.1 Request 对象概述

当用户打开 Web 浏览器，并从网站请求 Web 页时，Web 服务器就接收一个请求，此请求包含用户、用户的计算机、页面以及浏览器的相关信息，这些信息将被完整地封装，并在 Request 对象中利用它们，以上这些信息都是通过 Request 对象一次性提供的。

9.2.2 Request 对象常用属性

该对象使你获得 Web 请求的 HTTP 数据包的全部信息，其常用属性如表 9.3 所示。

表 9.3 Request 对象常用属性及说明

属　性	说　明
ApplicationPath	获取服务器上 ASP.NET 应用程序虚拟根目录路径
Browser	获取或设置有关正在请求的客户端浏览器的功能信息
ContentLength	指定客户端发送的内容长度（以字节计）
Cookies	获取客户端发送的 Cookie 集合
FilePath	获取当前请求的虚拟路径
Files	获取采用多部分 MIME 格式的由客户端上载的文件集合
Form	获取窗体变量集合
Item	从 Cookies、Form、QueryString 或 ServerVariables 集合中获取指定的对象
Params	获取 QueryString、Form、ServerVariables 和 Cookies 项的组合集合
Path	获取当前请求的虚拟路径
QueryString	获取 HTTP 查询字符串变量集合
UserHostAddress	获取远程客户端 IP 主机地址
UserHostName	获取远程客户端 DNS 名称

下面详细介绍 Request 对象的常用属性。

（1）Browser 属性

用于获取或设置有关正在请求的客户端浏览器的信息。

语法：

```
public HttpBrowserCapabilities Browser { get; set; }
```

其中的属性值：列出客户端浏览器功能的 HttpBrowserCapabilities 对象。

实例位置：光盘\MR\Instance\4\9.5

【例 9.8】本实例演示 Request 对象的 Browser 属性的具体应用。首先，启动 Visual Studio 2008，创建一个新的 ASP.NET 应用程序，默认主页为 Default.aspx。在 Default.aspx 页面的 Page_Load 事件中，调用 Request 对象的 Browser 属性输出与浏览器相关的一些信息，运行结果如图 9.6 所示。

图 9.6　获取浏览器信息

实现代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
Response.Write("浏览器使用的平台：" + Request.Browser.Platform + "<br>" + "浏览器类型：" + Request.Browser.Type + "<br>" + "浏览器版本："+ Request.Browser.Version);
}
```

（2）QueryString 属性

用于获取 HTTP 查询字符串变量的集合。

语法：

```
public NameValueCollection QueryString { get; }
```

属性值：NameValueCollection，包含由客户端发送的查询字符串变量的集合。

关于 QueryString 属性的用法在 9.1.4 节的实例代码中使用过，请读者参看其源码。

9.2.3　Request 对象常用方法

Request 对象常用方法及说明如表 9.4 所示。

表 9.4　Request 对象常用方法及说明

方　法	说　明
MapPath	请求的 URL 中的虚拟路径映射到服务器上的物理路径
SaveAs	将 HTTP 请求保存到磁盘

下面介绍 Request 对象的一些常用方法。

（1）MapPath 方法

MapPath 方法用于接收一个字符串类型的参数，并返回当前文件所在的实际路径。该方法主要应用在需要使用实际路径的位置。例如，建立数据源连接时，必须指定完整的路径，这时就可以使用 MapPath 方法取回。

语法：

```
public string MapPath (string virtualPath)
```

参数 virtualPath：当前请求的虚拟路径（绝对路径或相对路径）。

返回值：由 virtualPath 指定的服务器物理路径。

【例 9.9】 下面的代码用来取得页面 Default.aspx 的物理路径。

```
string str = Request.MapPath("Default.aspx");
```

（2）SaveAs 方法

SaveAs 方法用于将 HTTP 请求保存到硬盘上，在调试过程中非常有用，该方法常用格式如下。

```
Request.SaveAs("硬盘上的文件路径",true/false);
```

【例 9.10】 在应用程序中调用 Request 对象的 SaveAs 方法，将 HTTP 请求保存到文本文件中，关键代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    Request.SaveAs(Server.MapPath("test.txt"), true);
}
```

9.2.4 情景应用 1：获取地址栏传递的数据

专题讲座：光盘\MR\Video\9\获取页面间传送的值.exe

➤➤➤视频速递：理解如何在实际中应用 Request 对象。

实例位置：光盘\MR\Instance\9\9.6

【例 9.11】 本实例主要使用 QueryString 属性实现地址栏传值，当程序运行的时候，在页面 Default.aspx 的 TextBox 文本框中输入一个值，单击“搜索”按钮，将 TextBox 文本框中的值传到 Request.aspx 页面的地址栏中。实例运行结果如图 9.7 和图 9.8 所示。

图 9.7 页面传值

图 9.8 接收上个页面传值

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加一个 TextBox 控件和一个 Button 控件，分别用于输入相关信息和实现页面传值操作。

在 Default.aspx 页面中，单击“页面传值”按钮，程序跳转到 Request.aspx 页面，并使用标识 id 将 TextBox 文本框中的值传到 Request.aspx 页面中。“页面传值”按钮的 Click

事件代码如下所示。

```
protected void btnsearch_Click(object sender, EventArgs e)
{
    Response.Redirect("Request.aspx?id=" + txtkey.Text);
}
```

（2）右键单击解决方案名称，在弹出的快捷菜中选择“添加新项”命令，在弹出的对话框中选择“Web 窗体”，创建一个名为 Request.aspx 的页。此页主要用来接收主页传来的值。程序代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("地址栏传值 id 为：" + Request.QueryString["id"]);
}
```

9.2.5　情景应用 2：获取浏览器和主机信息

专题讲座：光盘\MR\Video\9\获取客户端浏览器信息.exe

➤➤➤视频速递：理解如何在实际中应用 Request 对象。

实例位置：光盘\MR\Instance\9\9.7

【例 9.12】 本实例主要通过 Request 对象的 Browser 属性获取客户端浏览器信息、UserHostName 属性获取客户端主机名称和 UserHostAddress 属性获取客户端 IP 地址等。执行程序，实例运行结果如图 9.9 所示。

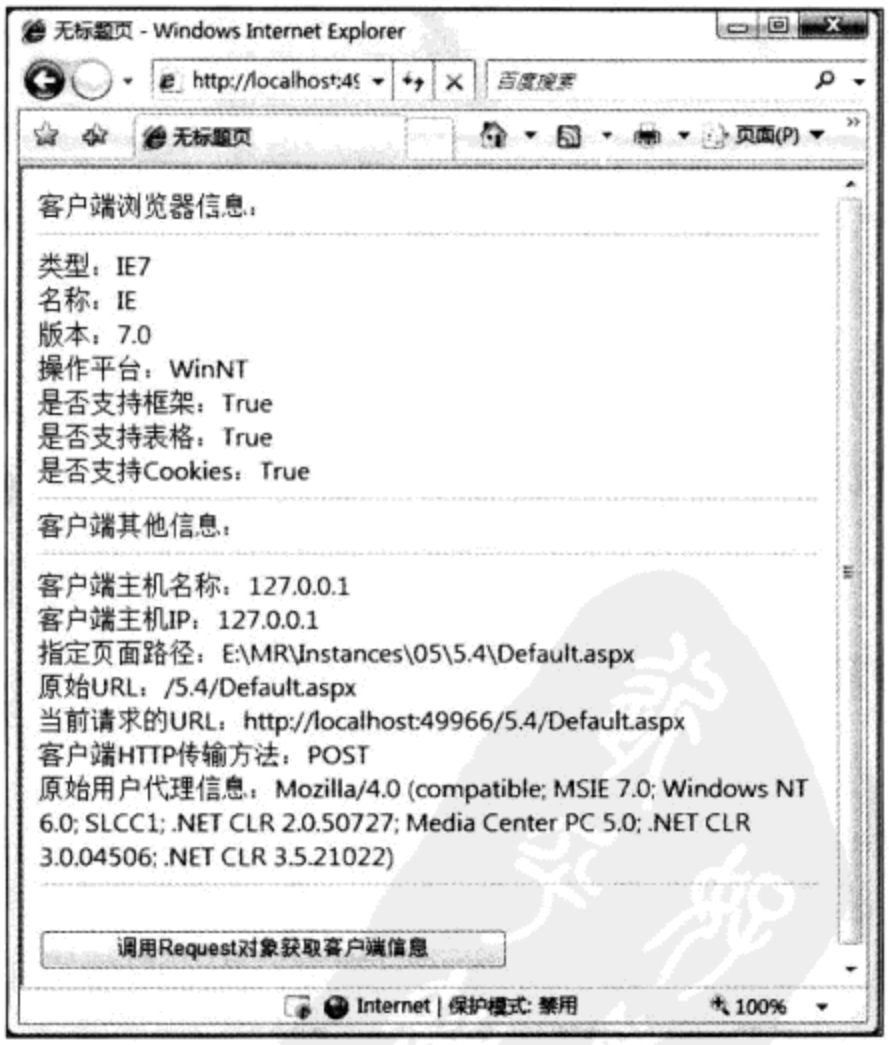


图 9.9　获取客户端浏览器等信息

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 的 Page_Load 事件中先定义 HttpBrowser- Capabilities 的类对象用于获取 Request 对象的 Browser 属性的返回值；另外，还调用了该对象中的如 UserHostName 属性来获取主机对应的名称等客户端信息，代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
    HttpBrowserCapabilities b = Request.Browser;
    Response.Write("客户端浏览器信息：");
    Response.Write("<hr>");
    Response.Write("类型：" + b.Type + "<br>");
    Response.Write("名称：" + b.Browser + "<br>");
    Response.Write("版本：" + b.Version + "<br>");
    Response.Write("操作平台：" + b.Platform + "<br>");
    Response.Write("是否支持框架：" + b.Frames + "<br>");
    Response.Write("是否支持表格：" + b.Tables + "<br>");
    Response.Write("是否支持 Cookies：" + b.Cookies + "<br>");
    Response.Write("<hr>");
    Response.Write("客户端其他信息：");
    Response.Write("<hr>");
    Response.Write("客户端主机名称：" + Request.UserHostName + "<br>");
    Response.Write("客户端主机 IP：" + Request.UserHostAddress + "<br>");
    Response.Write("指定页面路径：" + Request.MapPath("Default.aspx") + "<br>");
    Response.Write("原始 URL：" + Request.RawUrl + "<br>");
    Response.Write("当前请求的 URL：" + Request.Url + "<br>");
    Response.Write("客户端 HTTP 传输方法：" + Request.HttpMethod + "<br>");
    Response.Write("原始用户代理信息：" + Request.UserAgent + "<br>");
    Response.Write("<hr>");
}
```

9.3 全局变量应用对象 Application

Application 对象可称为记录应用程序参数的对象，该对象用于共享应用程序级信息。为了更好地理解此对象，本节将具体介绍该对象的常用属性和方法及在实际开发中的典型应用。

9.3.1 Application 对象概述

Application 对象用于共享应用程序级信息，即多个用户共享一个 Application 对象。在第一个用户请求 ASP.NET 文件时，将启动应用程序并创建 Application 对象。一旦 Application 对象被创建，它就可以共享和管理整个应用程序的信息。在应用程序关闭之前，Application 对象将一直存在。所以，Application 对象是用于启动和管理 ASP.NET 应用程序的主要对象。

9.3.2　Application 对象常用属性

Application 对象的常用属性及说明如表 9.5 所示。

表 9.5　Application 对象常用属性及说明

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

下面详细介绍 Application 对象的常用属性。

（1）Count 属性

用于获取 Application 对象变量的数量。

语法：

```
public override int Count { get; }
```

属性值 get：集合中的 Item 对象数，默认值为 0。

为了进一步理解这一属性的应用，以下小示例实现了当程序运行的时候，在 Default.aspx 页面中输出 Application 对象的数量及每一项的值。

【例 9.13】 使用 Application 对象的 Count 属性获取 Application 对象变量的数量，并输出到页面上，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
        Application["app1"] = "app1";                          //Application 对象赋值
        Application["app2"] = "app2";
        Application["app3"] = "app3";
        Response.Write("Application 对象数量为：" + Application.Count.ToString()
        + "个，分别为：<br>" + Application["app1"] + ", " + Application["app2"]+
        "和" + Application ["app3"]);
}
```

（2）Item 属性

该属性可重载，它有以下两种格式，分别通过索引和名称获取单个 Application 对象的值。

语法：

```
public Object this [int index] { get; }                    //通过索引获取
public Object this [string name] { get; set; }              //通过名称获取
```

index：集合中对象的数字索引。

name：集合中的对象名。

【例 9.14】 本示例使用 Application 对象 Item 属性的两种重载格式访问 Application 对象集合中的项，并输出到 Web 页面中，代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Application["app1"] = "App1";
```

```
        Application["app2"] = "App2";
        Application["app3"] = "App3";
        Response.Write(Application["app1"].ToString());
        Response.Write("</br>");
        Response.Write(Application[2].ToString());
    }
```

9.3.3 Application 对象常用方法

Application 对象常用方法及说明如表 9.6 所示。

表 9.6 Application 对象常用方法及说明

方　法	说　明
Add	新增一个 Application 对象变量
Clear	清除全部 Application 对象变量
Lock	锁定全部 Application 对象变量
Remove	使用变量名称移除一个 Application 对象变量
RemoveAll	移除全部 Application 对象变量
Set	使用变量名称更新一个 Application 对象变量的内容
UnLock	解除锁定的 Application 对象变量

下面介绍 Application 对象的一些常用方法。

（1）Add 方法

Add 方法将新变量添加到 Application 集合中。

语法：

```
public void Add (string name,Object value)
```

name：要添加到 Application 对象中的变量名。

value：变量的值。

【例 9.15】 下面的代码为将名为 mr 和 mrsoft 的应用程序变量添加到 Application 集合中。

```
Application.Add("mr", "mr");
Application.Add("mrsoft", "mrsoft");
```

（2）Lock 方法

Lock 方法用来锁定全部的 Application 对象变量。

语法：

```
public void Lock ()
```

【例 9.16】 统计在线人数时，就应该先对 Application 对象加锁，以防止因为多个用户同时访问页面而造成并行，常用格式如下。

```
Application.Lock();                          //锁定
Application("变量")=表达式;                    //赋值
Application.UnLock();                        //开锁
```

① Lock 方法可以阻止其他用户修改存储在 Application 对象中的变量，以确保在同一时刻仅有一个用户可以修改和存取 Application 变量。如果用户没有明确用 Unlock 方法，则服务器将在页面文件结束或超出时解除对 Application 对象的锁定。

② Unlock 方法可以使其他客户端在使用 Lock 方法锁住 Application 对象后，修改存储在该对象中的变量。如果未显式地调用该方法，Web 服务器将在页面文件结束或超出后解锁 Application 对象。

（3）Remove 方法

Remove 方法用来将指定变量从 Application 集合中移除。

语法：

```
public void Remove (string name)
```

参数 name：要从 Application 对象中移除的变量名。

【例 9.17】 下面的代码为将名为 mr 和 mrsoft 的应用程序变量从 Application 集合中移除。

```
Application.Remove("mr");
Application.Remove("mrsoft");
```

如果要移除 Application 集合中的所有变量，可直接调用其 RemoveAll 方法，实现代码如下所示。

```
Application.RemoveAll();
```

9.3.4　Application 对象常用事件

1．Application_Start 事件

Application_Start 事件在首次创建新的会话（即事件）之前发生，只有 Application 和 Server 内置对象可使用。在 Application_Start 事件中引用 Session、Request 或 Response 对象将导致错误。

Application 对象的 Start 事件肯定发生在 Session_Start 事件之前，不过，Application 对象不会像 Session 对象那样在一个新用户请求后触发，Application 对象只触发一次，即第一个用户的第一次请求。Application_Start 事件触发唯一一个脚本程序，它存在于 Global.asax 文件中。

由于 Application 对象是多用户共享的，因此它与 Session 对象有着本质的区别，同时 Application 对象不会因为某一个甚至全部用户的离开而消失，一旦建立了 Application 对象，那么它就会一直存在直到网站关闭或者该 Application 对象被卸载，这通常需要很长时间。

由于 Application 对象创建之后不会自己注销，因此一定要特别小心。另外，它会占用内存，要避免降低服务器对其他工作的响应速度。中止 Application 对象有 3 种方法，分别

为服务被中止、Global.asax 文件被改变或者该 Application 对象被卸载。

2. Application_End 事件

Application_End 事件在应用程序退出时于 Session_End 事件之后发生，只有 Application 和 Server 内置对象可使用。Application_End 事件只有在服务中止或者该 Application 对象卸载时才会触发，如果单独使用 Application 对象，该事件可以通过 Application 对象在利用 Unload 事件卸载时进行触发。一个 Application_End 事件肯定发生在 Session_End 事件之后。Application_End 事件触发唯一一个脚本程序，它存在于 Global.asax 文件中。

【例 9.18】 建立一个新的 ASP.NET 应用程序，默认主页为 Default.aspx。右键单击该应用程序名称，在弹出的快捷菜单中选择“添加新项”命令，在弹出的对话框中选择“全局应用程序类”项，创建一个名为 Global.asax 的文件，并在 Application_Start 事件中输入如下代码，实现当程序启动时，对 Application 对象进行赋值。

```
void Application_Start(object sender, EventArgs e)
{
    // 在应用程序启动时运行的代码
    Application["Name"] = "欢迎您点击明日科技公司网址：www.mingrisoft.com";
}
```

9.3.5 情景应用 1：简单聊天室

专题讲座：光盘\MR\Video\9\开发聊天室.exe

➤➤➤视频速递：理解如何在实际中应用 Application 对象。

实例位置：光盘\MR\Instance\9\9.8

【例 9.19】 本实例主要是利用 Application 对象实现聊天室功能，首先登录聊天室，在如图 9.10 所示的“昵称”文本框中输入用户登录的昵称，单击“进入聊天室”按钮登录聊天室开始聊天。此时就可以看到在线的所有人，以及聊天的相关信息。程序运行结果如图 9.11 所示。

图 9.10 聊天室登录界面

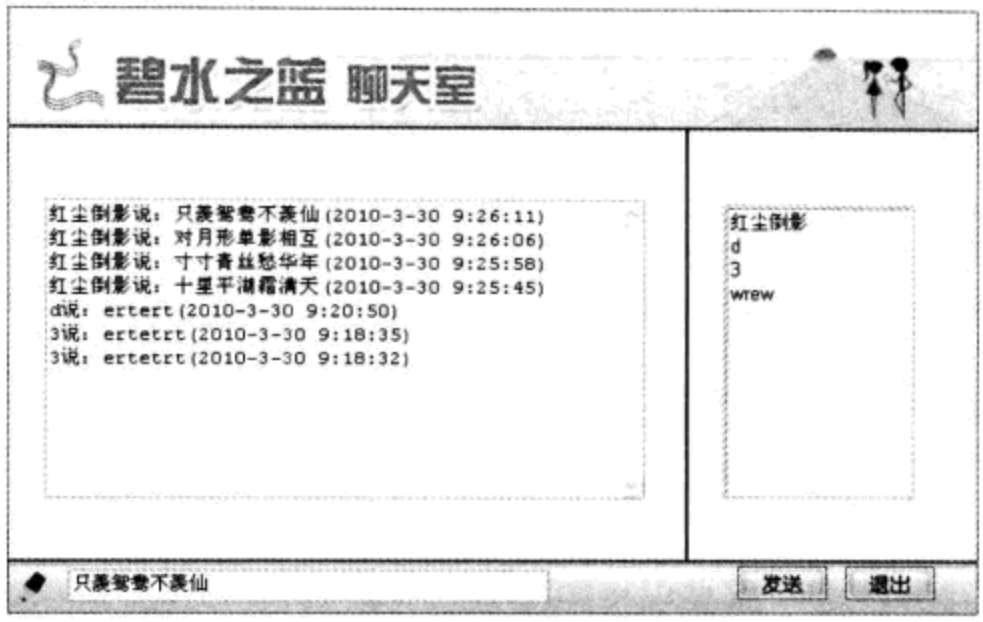


图 9.11　聊天室主界面

该聊天室是使用 Application 对象实现的，在应用程序启动时，应在 Application 对象的 Application_Start 事件中将所有数据初始化，代码如下所示：

```
void Application_Start(object sender, EventArgs e)
{
    // 在应用程序启动时运行的代码
    string user = "";                                          //用户列表
    Application["user"] = user;
    Application["userNum"] = 0;
    string chats = "";                                         //聊天记录
    Application["chats"] = chats;
    Application["current"] = 0;                                //当前的聊天记录数
}
```

学习手册 说明 Application_Start 事件在 Global.asax 文件中，创建网站时，该文件不会自动创建，如果需要该文件，必须手动添加。

在聊天室主页面中，单击“发送”按钮时，首先调用 Application 对象的 Lock 方法对所有 Application 对象进行锁定，然后判断当前聊天信息的记录数是否大于 20，如果大于，则清空聊天记录，并重新加载用户的聊天记录；否则，将聊天内容、用户名、发信息时间保存在 Application 对象中，代码如下所示：

```
    int P_int_current = Convert.ToInt32(Application["current"]);
                                                   //获取当前的聊天记录数
    Application.Lock();                            //Application 对象进行锁定
    if (P_int_current == 0 || P_int_current > 20)  //判断当前聊天信息的记录数是否大于 20
    {
    P_int_current = 0;                             //清空聊天记录数
    Application["chats"] = Session["userName"].ToString() + "说：" + txtMessage.
Text.Trim() + "(" +      DateTime.Now.ToString() + ")";    //设置聊天内容
    }
    else                                           //如果没有超出限制
    {
    Application["chats"]  =  Application["chats"].ToString()  +  ","  +  Session
["userName"].ToString() + "说：" +    txtMessage.Text.Trim() + "(" + DateTime.Now.
ToString() + ")";                                  //设置聊天内容
```

```
}
P_int_current += 1;                          //聊天信息数加一
Application["current"] = P_int_current;      //将聊天信息数存储到 Application 对象
Application.UnLock();                        //Application 对象进行解锁
```

显示聊天信息页面 Content.aspx 加载时，从 Application 对象中读取保存的聊天信息，并将其显示在 TextBox 文本框中。Content.aspx 页面的 Page_Load 事件代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    int P_int_current = Convert.ToInt32(Application["current"]);
                                                 //获取当前的聊天记录数
    Application.Lock();                          //Application 对象进行锁定
    string P_str_chats = Application["chats"].ToString(); //读取保存的聊天信息
    string[] P_str_chat = P_str_chats.Split(',');          //聊天信息拆分为数组
    for (int i = P_str_chat.Length - 1; i >= 0; i--)       //遍历聊天信息数组
    {
        if (P_int_current == 0)                  //如果聊天记录数为 0 说明刚开始发言
        {
            txtContent.Text = P_str_chat[i].ToString();    //显示发言信息
        }
        else                                     //如果不为 0 说明已经发言多次
        {
      //在原有的聊天信息中加入当前发送的信息，并显示出来
            txtContent.Text = txtContent.Text + "\n" + P_str_chat[i].ToString();
        }
    }
    Application.UnLock();                        //Application 对象进行解锁
}
```

由于篇幅有限，以上代码只是部分代码。具体代码读者可以参考附带光盘中的源码。

9.3.6 情景应用 2：在线访问人数统计

专题讲座：光盘\MR\Video\9\在线访问人数统计.exe

➤➤➤视频速递：理解如何在实际中应用 Application 对象。

实例位置：光盘\MR\Instance\9\9.9

【例 9.20】 本实例主要是对 Global.asax 文件的配置，并在该文件中对在线访问人数进行统计，然后将访问人数在 Default.aspx 文件中显示出来。实例运行效果如图 9.12 所示。

程序实现步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，然后右键单击该网站名称，在弹出的快捷菜单中单击“添加新项”选项，添加一个“全局应用程序类（即 Global.asax 文件）”，如图 9.13 所示。

图 9.12　在线访问人数统计

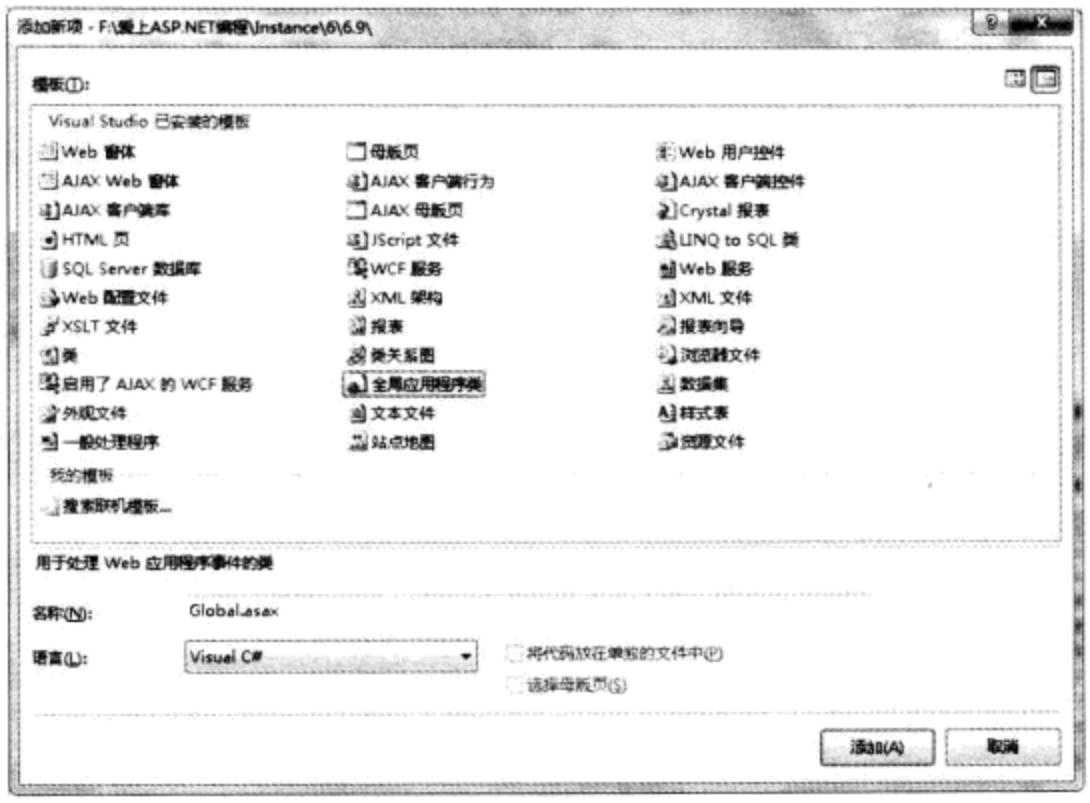


图 9.13　选择“全局应用程序类”选项

（2）在 Global.asax 文件的 Application_Start 事件中首先将访问数初始化为 0，代码如下：

```
void Application_Start(object sender, EventArgs e)
{
    // 在应用程序启动时运行的代码
    Application["count"] = 0;                                    //创建变量并初始化为 0
}
```

当有新的用户访问网站时，将建立一个新的 Session 对象，并在 Session 对象的 Session_Start 事件中对 Application 对象加锁，以防止因为多个用户同时访问页面造成并行，同时将访问人数加 1；当用户退出该网站时，将关闭该用户的 Session 对象，同时对 Application 对象加锁，然后将访问人数减 1。代码如下：

```
void Session_Start(object sender, EventArgs e)
{
    //在新会话启动时运行的代码
    Application.Lock();                                          //加锁
    Application["count"] = (int)Application["count"] + 1;        //赋值使人数增加 1
    Application.UnLock();                                        //解锁
}
void Session_End(object sender, EventArgs e)
{
    //在会话结束时运行的代码。
    // 注意：只有在 Web.config 文件中的 sessionstate 模式设置为
    // InProc 时，才会引发 Session_End 事件。如果会话模式
    //设置为 StateServer 或 SQLServer，则不会引发该事件。
    Application.Lock();                                          //加锁
    Application["count"] = (int)Application["count"] - 1;        //赋值使人数减 1
    Application.UnLock();                                        //解锁
}
```

（3）对 Global.asax 文件进行设置后，需要将访问人数在网站的默认主页 Default.aspx 中显示出来。在 Default.aspx 页面上添加了一个 Label 控件，用于显示访问人数。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    lblinfo.Text = "您是该网站的第" + Application["count"].ToString() + "个访问者!";
}
```

9.4 会话信息处理对象 Session

Session 对象可称为记录浏览器端的变量对象，用来存储跨网页程序的变量或者对象。为了更好地理解该对象，本节具体介绍该对象的常用属性和方法及在实际开发中的典型应用。

9.4.1 Session 对象的概述

Session 对象用来存储跨网页程序的变量或者对象，Session 对象只针对单一网页使用者，也就是说服务器会为连接的客户端分配各自的 Session 对象，不同的客户端无法互相存取。Session 对象超过设置的有效时间时就会消失。

Session 对象和 Application 对象一样都是 Page 对象的成员，因此可以直接在网页中使用。使用 Session 对象存放信息的语法如下：

```
Session["变量名"]= "内容";
```

从会话中读取信息的语法如下：

```
VariablesName=Session["变量名"];
```

9.4.2 Session 对象常用属性

Session 对象的常用属性及说明如表 9.7 所示。

表 9.7 Session 对象常用属性及说明

属　性	说　明
Contents	获取对当前会话状态对象的引用
Item	获取或设置会话值
TimeOut	传回或设定 Session 对象变量的有效时间，当使用者超过有效时间没有动作，Session 对象就会失效。默认值为 20 分钟

下面详细介绍 Session 对象的常用属性。

（1）Contents 属性

获取对当前会话状态对象的引用。

语法：

```
public HttpSessionState Contents { get; }
```

属性值：当前的 HttpSessionState。

【例 9.21】 本实例实现的是当程序运行的时候，通过调用 Session 对象的 Contents 属性，循环输出 Session 变量的值。

```
protected void Page_Load(object sender, EventArgs e)
{
    //Session 对象赋值
    Session["id1"] = "ID1";
    Session["id2"] = "ID2";
    Session["id3"] = "ID3";
    foreach(string str in Session.Contents)          //遍历对 Session 对象应用的集合
    {
        Response.Write(Session[str].ToString());     //输出引用值
        Response.Write("</br>");
    }
}
```

（2）TimeOut 属性

获取并设置在会话状态提供程序终止会话之前各请求之间所允许的时间（以分钟为单位）。

语法：

```
public int Timeout { get; set; }
```

属性值 get、set：表示超时期限（以分钟为单位）。

在应用程序开发过程中，要更改 Session 对象的有效期限，只要设定 TimeOut 属性即可，TimeOut 属性的默认值是 20 分钟。

可以在应用程序的 Web.config 文件中使用 sessionState 配置元素的 timeout 属性来设置 Timeout 属性，也可以使用程序代码来直接设置 Timeout 属性值。Timeout 属性不能设置为超过 525600 分钟（1 年）的值。

【例 9.22】 在 Web.Config 文件中设置 Session 对象的有效时间为 30 分钟，代码如下。

```
<sessionState mode="InProc"  timeout="30"/>
```

9.4.3　Session 对象常用方法

Session 对象常用方法及说明如表 9.8 所示。

表 9.8　Session 对象常用方法及说明

方　法	说　明
Abandon	此方法结束当前会话，并清除会话中的所有信息。如果用户随后访问页面，可以为它创建新会话（“重新建立”非常有用，这样用户就可以得到新的会话）
Add	用于向 Session 对象集合中添加一个新项
CopyTo	将会话状态值的集合复制到一维数组中（从数组的指定索引处开始）
Clear	此方法清除全部的 Session 对象变量，但不结束会话

下面详细介绍 Session 对象的常用方法。

（1）Clear 方法

Clear 方法用来清除全部 Session 对象变量。

语法：

```
public void Clear ()
```

【例 9.23】 移除 Session 对象集合中所有的键值。

```
Session.Clear();
```

（2）Add 方法

用于向 Session 对象集合中添加一个新项。

语法：

```
public void Add (string name,Object value)
```

name：要添加到 Session 对象集合中的项的名称。

value：要添加到 Session 对象集合中的项的值。

【例 9.24】 在 Session 对象集合中添加一个名称为“test”、值为“test 文本”的项。

```
Session.Add("test", "test 文本");
```

9.4.4 情景应用：Session 对象存储登录信息

专题讲座：光盘\MR\Video\9\Session 对象存储登录信息.exe

➤➤➤视频速递：理解如何在实际中应用 Session 对象

实例位置：光盘\MR\ Instance\9\9.10

【例 9.25】 用户登录后有时会根据其登录身份（如管理员）来记录该用户相关信息，而该信息是其他用户不可见且不可访问的，这就需要使用 Session 对象进行存储。下面通过实例介绍如何使用 Session 对象保存当前登录用户的信息。执行程序，运行结果分别如图 9.14 和图 9.15 所示。

图 9.14 用户登录界面

图 9.15 应用 Session 对象记录用户登录名

程序实例的具体步骤如下。

（1）新建一个网站，默认主页 Default.aspx，将其命名为 Login.aspx。在 Login.aspx 页面上添加两个 TextBox 控件和两个 Button 控件，它们的属性设置如表 9.9 所示。

表 9.9　Login.aspx 页面中控件属性设置及其用途.

控 件 类 型	控 件 名 称	主要属性设置	用　途
标准/TextBox 控件	txtName	无	输入用户名
	txtPwd	TextMode 属性设置为“Password”	输入密码
标准/Button 控件	btnLogin	Text 属性设置为“登录”	登录按钮
	btnCancel	Text 属性设置为“取消”	取消按钮

（2）双击 Login.aspx 页面中的“登录”按钮，触发其 Click 事件，实现将用户登录名及登录时间存储到 Session 对象中，代码如下所示：

```
if (txtName.Text == "mr" && txtPwd.Text == "mrsoft")
{
    Session["UserName"] = txtName.Text;   //使用 Session 变量记录用户名
    Session["TimeLogin"] = DateTime.Now; //使用 Session 变量记录用户登录系统的时间
    Response.Redirect("~/UserPage.aspx"); //跳转到主页
}
else
{
    Response.Write("<script>alert('登录失败！请返回查找原因');location='Login.
    aspx'</script>");
}
```

（3）在该网站中，添加一个新页，将其命名为 UserPage.aspx。在页面 UserPage.aspx 的初始化事件中，将登录页中保存的用户登录信息显示在页面上，并同时将在线访问人数显示在该页面中，代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
   Response.Write("欢迎用户" + Session["UserName"].ToString() + "登录本系统!
   <br>");
   Response.Write("您登录的时间为：" + Session["TimeLogin"].ToString());
}
```

9.5　缓存对象 Cookie

Cookie 对象也称缓存对象，该对象用于保存客户端浏览器请求的服务器页面，也可用它存放非敏感性的用户信息。为了更好地理解，本节具体介绍该对象的常用属性和方法及在实际开发中的典型应用。

9.5.1　Cookie 对象概述

Cookie 对象用于保存客户端浏览器请求的服务器页面，也可用它存放非敏感性的用户信息，信息保存的时间可以根据用户的需要进行设置。并非所有的浏览器都支持 Cookie，

并且数据信息是以文本的形式保存在客户端计算机中。

9.5.2 Cookie 对象常用属性

Cookie 对象的常用属性及说明如表 9.10 所示。

表 9.10 Cookie 对象常用属性及说明

属　性	说　明
Clear	清除所有的 Cookie
Expires	设定 Cookie 变量的有效时间，默认为 1000 分钟，若设为 0，则可以实时删除 Cookie 变量
Name	取得 Cookie 变量的名称
Value	获取或设置 Cookie 变量的内容值
Path	获取或设置 Cookie 适用于的 URL

下面详细介绍 Cookie 对象的常用属性。

（1）Expires 属性

获取或设置 Cookie 的过期日期和时间。

语法：

```
public DateTime Expires { get; set; }
```

属性值：Cookie 的过期时间（在客户端）。

【例 9.26】 将 Cookie 的过期时间设置为当前时间之后 20 秒钟，代码如下。

```
HttpCookie cookie = new HttpCookie("userName");          //声明一个 Cookie 变量
cookie.Value = "明日科技";                                //赋值给这个 Cookie 变量
DateTime time = DateTime.Now;                            //获取当前时间
TimeSpan TSpan = new TimeSpan(0, 0, 0, 20);              //设置时间间隔
cookie.Expires = time.Add(TSpan);                        // Cookie 过期时间
```

Expires 属性必须被设置，若没有指定，则 Cookie 变量将不会被存储。

（2）Path 属性

获取或设置要与当前 Cookie 一起传输的虚拟路径。

语法：

```
public string Path { get; set; }
```

属性值：要与当前 Cookie 一起传输的虚拟路径。默认值为当前请求的路径。

【例 9.27】 要获得与当前 Cookie 一起传输的虚拟路径可通过调用 Cookie 对象的 Path 属性来实现。要实现以上操作，可在创建的 Web 窗体的后台代码中输入如下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie = new HttpCookie("test");          //声明一个 Cookie 变量
    cookie.Value = "cookieTest";                         //赋值给这个 Cookie 变量
    Response.Cookies.Add(cookie);                        //添加 Cookie
    Response.Write(Request.Cookies["test"].Path);        //页面输出
}
```

9.5.3　Cookie 对象常用方法

Cookie 对象常用方法及说明如表 9.11 所示。

表 9.11　Cookie 对象常用方法及说明

方　法	说　明
Equals	确定指定 Cookie 是否等于当前的 Cookie
ToString	返回此 Cookie 对象的一个字符串表示形式

（1）ToString 方法

返回表示当前 Object 的 String。

语法：

```
public virtual string ToString ()
```

返回值：String，表示当前的 Object。

（2）Equals 方法

确定指定的 Object 是否等于当前的 Object。

语法：

```
public virtual bool Equals (Object obj)
```

obj：与当前的 Object 进行比较的 Object。

返回值：如果指定的 Object 等于当前的 Object，则为 True；否则为 False。

下面给合一个实例介绍 Equals 方法的具体应用。

实例位置：光盘\MR\ Instance\9\9.11

【例 9.28】 本实例通过调用 Cookie 对象的 Equals 方法来判断已经定义的两个 Cookie 对象的值是否相等。运行结果如图 9.16 所示。

图 9.16　比较 Cookie 对象的值是否相等

启动 Visual Studio 2008，新建一个 ASP.NET 应用程序，默认主页名为 Default.aspx。在该主页的 Page_Load 事件中输入以下代码，实现通过调用 Cookie 对象的 Equals 方法来判断已经定义的两个 Cookie 对象的值是否相等，代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
        //声明一个 HttpCookie 实例 cookie1
        HttpCookie cookie1 = new HttpCookie("userName1");
        cookie1.Value = "明日科技";                          //赋值给 cookie1
        //声明一个 HttpCookie 实例 cookie1
        HttpCookie cookie2 = new HttpCookie("UserName2");
        cookie2.Value = "明日科技";                          //赋值给 cookie1
        Response.Cookies.Add(cookie1);                       //添加 cookie1
        Response.Cookies.Add(cookie2);                       //添加 cookie2
        if (Equals(Request.Cookies["userName1"].Value, Request.Cookies
        ["userName2"]. Value))
```

```
            Response.Write("Cookie 值相等");                    //两个 Cookie 对象的值是否相等
        else
        Response.Write("Cookie 值不相等");
}
```

9.5.4 情景应用：Cookie 对象存储登录用户名和密码

专题讲座：光盘\MR\Video\9\Cookie 对象记录登录信息.exe

➤➤➤视频速递：理解如何在实际中应用 Cookie 对象。

实例位置：光盘\MR\ Instance\9\9.12

【例 9.29】 本实例实现的是通过使用 Cookie 对象来存储登录用户名及密码信息，登录成功后，跳转到 Main.aspx 页面，在该页面中显示登录用户名信息。程序运行结果如图 9.17 所示，成功登录后，运行效果如图 9.18 所示。

图 9.17 用户登录界面

图 9.18 登录成功页

程序实例的步骤如下。

（1）新建一个网站，默认主页为 Default.aspx，在该主页中添加两个 TextBox 控件和两个 Button 控件，分别用来实现输入用户名及密码、提交和重置登录信息。

（2）双击 Default.aspx 页面中的“登录”按钮，触发其 Click 事件，实例应用 Cookie 对象将用户登录信息存储起来，实现的代码如下所示：

```
protected void imb_login_Click(object sender, EventArgs e)
{
    if (t_AdminCode.Text.Trim() != "" && t_AdminPwd.Text.Trim()!= "")
    {
        //声明一个 Cookie 对象,存储用户输入的用户名
        HttpCookie cookieAdminCode = new HttpCookie("CookAdminCode");
        //赋值给定义的 Cookie 值
        cookieAdminCode["AdminCode"] = t_AdminCode.Text.Trim();
        //设定 Cookie 过期时间
        cookieAdminCode.Expires.AddDays(1);
        //添加 Cookie 值
        Response.Cookies.Add(cookieAdminCode);
        //声明一个 Cookie 对象,存储用户输入的密码
        HttpCookie cookieAdminPwd = new HttpCookie("CookAdminPwd");
```

```
            cookieAdminPwd["AdminPwd"] = t_AdminPwd.Text.Trim();
            cookieAdminPwd.Expires.AddDays(1);
            Response.Cookies.Add(cookieAdminPwd);
            Response.Redirect("Main.aspx");
        }
        else
        {
            MessageBox("对不起，请输入用户名及密码！");
        }
    }
```

（3）在 Default.aspx 页面中用户输入的用户名及密码是否正确将在新创建的 Main.aspx 页面中进行验证。这里主要是通过 Cookie 对象来接收登录页面中传过来的用户名及密码，具体代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookieAdminCode = Request.Cookies["CookAdminCode"];
    HttpCookie cookieAdminPwd = Request.Cookies["CookAdminPwd"];
    string AdminCode = cookieAdminCode.Values["AdminCode"].ToString().Trim();
    string AdminPwd = cookieAdminPwd.Values["AdminPwd"].ToString().Trim();
    if (AdminCode == "mr" && AdminPwd == "mrsoft")
    {
        MessageBox("登录成功！");
        lblinfo.Text = "欢迎" + AdminCode + "登录本系统！";
    }
    else
    {
        MessageBox("对不起，身份验证失败请重试！");
        Response.Write("<script language=javascript>window.location.href=
        'Default.aspx'</script>");
    }
}
```

另外，对于本实例中弹出提示框信息编写了一个自定义方法 MessageBox，代码如下所示：

```
private void MessageBox(string str)
{
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('" + str + "');",
true);
}
```

9.6　服务器信息处理对象 Server

Server 对象又称为服务器对象。为了更好地理解，本节具体介绍该对象的常用属性和方法及在实际开发中的典型应用。

9.6.1　Server 对象概述

Server 对象定义了一个与 Web 服务器相关的类提供对服务器上方法和属性的访问。

9.6.2 Server 对象常用属性

Server 对象的常用属性及说明如表 9.12 所示。

表 9.12 Server 对象常用属性及说明

属　性	说　明
MachineName	获取服务器的计算机名称
ScriptTimeout	获取和设置请求超时值（以秒计）

下面对这两个属性进行详细介绍。

（1）MachineName 属性

获取服务器的计算机名称。

语法：

```
public string MachineName { get; }
```

图 9.19 使用 MachineName 属性获取服务器名称

属性值：本地计算机的名称。

实例位置：光盘\MR\ Instance\9\9.13

【例 9.30】 本例介绍在实际开发中如何使用 Server 对象的 MachineName 属性获取计算机名称，运行效果如图 9.19 所示。

本实例中调用 Server 对象的 MachineName 属性获取服务器名称，然后使用 ToLower 方法将得到的服务器名称转换为小写，并输出到页面中，实现代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Server.MachineName.ToLower());
}
```

（2）ScriptTimeout 属性

获取和设置请求超时值（以秒计）。

语法：

```
public int ScriptTimeout { get; set; }
```

属性值：请求的超时值设置。

【例 9.31】 将请求超时期限设置为 30 秒，代码如下所示。

```
Server.ScriptTimeout = 30;
```

9.6.3 Server 对象常用方法

Server 对象常用方法及说明如表 9.13 所示。

表 9.13　Server 对象常用方法及说明

方　法	说　明
HtmlDecode	对已被编码以消除无效 HTML 字符的字符串进行解码
HtmlEncode	对要在浏览器中显示的字符串进行编码
MapPath	返回与 Web 服务器上的指定虚拟路径相对应的物理文件路径
UrlDecode	对字符串进行解码，该字符串为了进行 HTTP 传输而进行编码并在 URL 中发送到服务器
UrlEncode	编码字符串，以便通过 URL 从 Web 服务器到客户端进行可靠的 HTTP 传输

下面介绍 Server 对象的一些常用方法。

（1）MapPath 方法

MapPath 方法用来返回与 Web 服务器上的指定虚拟路径相对应的物理文件路径，其常用格式如下。

```
Server.MapPath("path");
```

参数 path：表示 Web 服务器上的虚拟路径，如果 path 值为空，则该方法返回包含当前应用程序的完整物理路径。

【例 9.32】 在浏览器中输出指定文件“Default.aspx”的物理文件路径，代码如下。

```
Response.Write(Server.MapPath("Default.aspx"));
```

在 IIS 5.x 版本中，MapPath 方法一般不会出现什么问题。但在 IIS 9.x 版本中可能会出现无法返回路径的情况，例如，可能的报错信息有“在 MapPath 的 Path 参数中不允许字符'../' ”。这是因为 IIS 9.x 在默认情况下是不能访问父路径的，如果使用'../'就会报错。启用父路径访问的方法如下：启动 ISS，在左侧窗格中选择“网站”下的“默认网站”节点，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，打开“网站属性”对话框。选择“主目录”选项卡，单击“配置”按钮，打开“应用程序配置”对话框。选择“选项”选项卡，选中“启用父路径”复选框即可。

（2）HtmlEncode 方法

用于对字符串进行 HTML 编码，并返回已编码的字符串。

语法：

```
public string HtmlEncode (string s)
```

实例位置：光盘\MR\Instance\9\9.14

【例 9.33】 本实例实现的是当程序运行的时候，在页面 Default.aspx 中使用 Server 对象的 HtmlEncode 方法编码字符串。运行结果如图 9.20 所示。

图 9.20　输出编码后的字符串

实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("使用 HtmlEncode 方法为指定字符串编码：" + Server.HtmlEncode
    ("<B>浏览器中字符串</B>") + "<br>");
}
```

（3）HtmlDecode 方法

HtmlEncode 方法用来对字符串进行 HTML 编码并返回已编码的字符串。

语法：

```
public string HtmlDecode (string s)
```

参数 s：要解码的 HTML 字符串。

返回值：已解码的文本。

【例 9.34】 将一个名为“mr”的字符串进行 HTML 编码并显示在浏览器上，代码如下。

```
Response.Write(Server.HtmlEncode("mr"));
```

9.6.4 情景应用：获取文件或文件夹在服务器中的物理地址

专题讲座：光盘\MR\Video\9\获取文件或文件夹在服务器中的物理地址.exe

➤➤➤视频速递：理解如何在实际中应用 Server 对象。

图 9.21 使用 Server.MapPath 方法获取服务器的物理地址

实例位置：光盘\MR\Instance\9\9.15

【例 9.35】 本实例实现的是用 Server 对象的 MapPath 方法获取虚拟路径所对应的物理路径。运行结果如图 9.21 所示。

程序主要开发步骤如下。

（1）启动 Visual Studio 2008，创建一个新的 ASP.NET 应用程序，默认主页名为 Default.aspx。

（2）在 Default.aspx.cs 的 Page_Load 方法中输入以下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Server.MapPath("2.jpeg"));                    //获取物理地址
}
```

9.7 实 战 练 习

9.7.1 使用 Response 对象设置页面缓存

➤➤➤题目描述

ASP.NET 缓存技术是一项非常重要的技术。当某个页面被频繁访问，如果不使用缓存技术，那么每访问一次就要回发一次服务器，显然这样对服务器造成很大的负担。所以，可以在被频繁访问的页面中设置缓存。例如，网站的注册页面就可以设置缓存，因为它可能被频繁访问。所以设置页面缓存是至关重要的，本实例通过 Response 对象设置页面缓存，运用缓存技术的前后对比如图 9.22 所示。

图 9.22　Response.Cache 属性设置页面缓存的前后对比

➤➤➤技术指导

本实例主要使用 Response 对象的 Cache 属性设置页面缓存，它用以取得缓存策略（Caching Policy），包括过期、私密性与 Vary 子句等，其类型浅显地说就是 HttpCachePolicy 类型。设置缓存的关键方法是 HttpCachePolicy 类的 SetExpires 方法。

该方法用于设置缓存过期的绝对时间。

语法：

```
public void SetExpires (DateTime date)
```

参数 date：是一个 DataTime 类的实例，表示过期的绝对时间。如果过期日期违反限制原则，此方法将失败。

【例 9.36】 将过期时间设置为当前过期日的本地时间 6:00 PM，代码如下所示：

```
Response.Cache.SetExpires(DateTime.Parse("6:00:00PM"));
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/love ASP.NET /9.7.1**

9.7.2　利用 Cookie 统计 IP 地址登录次数

➤➤➤题目描述

Cookie 提供了一种在 Web 应用程序中存储用户特定信息的方法。例如，当用户访问当前的站点时，可以使用 Cookie 存储用户的 IP 地址或其他信息。当该用户再次访问当前的网站时，应用程序便可以检索以前存储的信息。本程序利用 Cookie 存储用户特定信息，实现统计用户 IP 地址的登录次数。程序运行结果如图 9.23 所示。

图 9.23　利用 Cookie 统计 IP 地址登录次数

➤➤➤技术指导

开发本实例时主要应该了解如何对 Cookie 对象进行存储和读取操作。首先应该通过下面的代码来创建 Cookie 并存储数据：

```
Response.Cookies["UserSettings"].Value = lastVisitCounter.ToString();
Response.Cookies["UserSettings"].Expires = DateTime.MaxValue;
```

那么，我们应该如何读取 Cookie 中的值呢？当浏览器向服务器发出请示时，会随请求一起发送该服务器的 Cookie。在 ASP.NET 应用程序中，可以使用 Cookie 名称作为主键从 Cookies 集合中读取字符串。其代码如下所示。

```
if (Request.Cookies["UserSettings"] != null)
{
   string userSettings= Server.HtmlEncode(Request.Cookies["UserSettings"].Value);
}
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/love ASP.NET /9.7.2**

9.7.3 Session 对象判断用户登录状态

➤➤➤题目描述

在开发网站时，大多数页面只有当用户登录之后才可以访问。那么，我们该如何判断用户是否已经成功登录呢？答案是 Session 对象。在某个页面通过判断存储用户名的 Session 对象是否为空，来判断用户是否成功登录。在登录界面中，如果用户成功登录则创建一个 Session 对象，并存储登录用户名。在另一个页面中，通过判断 Session 对象是否为空来判断用户是否成功登录，如果为空则将页面转向如图 9.24 所示的登录界面。

图 9.24　如果 Session 对象为空则返回登录窗口

➤➤➤技术指导

开发本实例的关键技术是对 Session 对象的写入和读取。Session 对象和 Application 对象一样都是 Page 对象的成员，因此可直接在网页中使用。使用 Session 对象存放信息的语法如下：

```
Session["变量名"]= "内容";
```

从会话中读取信息的语法如下：

```
VariablesName=Session["变量名"];。
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/love ASP.NET/9.7.3**

9.8　本 章 小 结

本章重点介绍了 ASP.NET 3.5 的内置对象，这些对象主要有 Request 对象、Response 对象、Server 对象、Session 对象和 Application 对象等。其中 Request 对象通过 HTTP 请求可以得到客户端的信息，Response 对象控制发送给客户端的信息，而 Server 对象提供了服务器端的基本属性与方法。另外，本章介绍了 Session 对象和 Cookie 对象的使用方法。通过这些内容的学习，读者应能够熟练地掌握和运用各种 ASP.NET 3.5 对象进行基本的程序设计。

第 10 章

ADO.NET 数据库开发技术

（名师课堂：1 小时 21 分钟）

随着网络技术的飞速发展，网络信息的不断增加，数据库访问技术越来越受到大家的关注，ADO.NET 数据访问技术受到了大家广泛的好评。以前，大多数 Web 页都由静态信息组成，Web 站点仅允许访问者读取数据，其交互性不强，并且不存储访问者的信息，如果必须存储访问者的信息，则需要使用数据库并制定数据访问策略，该数据访问策略允许程序员与数据库建立连接，并且提供检索、插入、更新和删除数据的命令。通过本章的学习，读者可以掌握以下知识：

- 运用各种数据库的连接技术实现与数据库的关联
- 使用 Command 对象操作数据库，实现添加、修改和删除
- 运用节省内存的 DataReader 对象读取数据库中的数据
- 使用 ADO.NET 核心对象 DataSet 将数据存储到内存中
- 应用 DataSet 对象和 DataAdapter 对象查询数据
- 使用 DataAdapter 更新数据库中的数据（支持批量更新）

10.1　了解什么是 ADO.NET

数据库的应用在我们的生活和工作中可以说已经无处不在，无论是一个小型的企业办公自动化系统，还是像中国移动的大型运营系统，似乎都离不开数据库的应用。对于大多数应用程序来说，不管它们是 Windows 桌面应用程序，还是 Web 应用程序，存储和检索数据都是其核心功能。所以针对数据库的开发已经成为软件开发的一种必备技能。如果说过去是“学好数理化，走遍天下都不怕”，那么对于今天的软件开发者而言就是“学好数据库，走到哪儿都不怵”！

ADO.NET 是微软新一代.NET 数据库的访问架构，ADO 是 ActiveX Data Objects 的缩写。ADO.NET 是数据库应用程序和数据源之间沟通的桥梁，主要提供一个面向对象的数据访问架构，用来开发数据库应用程序。为了更好地理解 ADO.NET 架构模型的各个组成部分，我们可以对 ADO.NET 中的相关对象进行图示理解，如图 10.1 所示的是 ADO.NET 中数据库对象的关系图。

ADO.NET 主要包括 Connection、Command、DataReader、DataSet 和 DataAdapter 对象。详细介绍如表 10.1 所示。

表 10.1　ADO.NET 的主要对象及说明

对 象 名 称	说　　明
Connection	对象主要提供与数据库的连接功能
Command	对象用于返回数据、修改数据、运行存储过程以及发送或检索参数信息的数据库命令
DataReader	对象通过 Command 对象提供从数据库检索信息的功能。DataReader 对象以一种只读的、向前的、快速的方式访问数据库
DataSet	是 ADO.NET 的中心概念，它是支持 ADO.NET 断开式、分布式数据方案的核心对象。它是一个数据库容器，可以把它当做是存在于内存中的数据库。DataSet 是数据的内存驻留表示形式，无论数据源是什么，它都会提供一致的关系编程模型；它可以用于多种不同的数据源。例如，用于访问 XML 数据，或用于管理本地应用程序的数据
DataAdapter	提供连接 DataSet 对象和数据源的桥梁，DataAdapter 对象使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并确保 DataSet 中数据的更改与数据源保持一致

这里可以用趣味形象化的方式理解 ADO.NET 对象模型的各个部分，如图 10.2 所示，对比 ADO.NET 的数据库对象关系图，我们可以形象地理解每个对象的作用。

轻松一刻：趣味理解 ADO.NET

☑ 数据库好比水源，存储了大量的数据。

☑ Connection 好比伸入水中的进水笼头，保持与水的接触，只有它与水进行了“连接”，其他对象才可以抽到水。

☑ Command 则像抽水机，为抽水提供动力和执行方法，通过“水龙头”把水返给

上面的“水管”。

- ☑ DataAdapter、DataReader 就像输水管，担任着水的传输任务，并起着桥梁的作用。DataAdapter 像一根输水管，通过抽水机，把水从水源输送到水库里进行保存。DataReader 也是一种水管，和 DataAdapter 不同的是，DataReader 不把水输送到水库里面，而是单向地直接把水送到需要水的用户那里或田地里，所以要比在水库中转一下（速度）更快。
- ☑ DataSet 则是一个大水库，把抽上来的水按一定关系的池子进行存放。即使撤掉“抽水装置”（断开连接，离线状态），也可以保持“水”的存在。这也正是 ADO.NET 的核心。
- ☑ DataTable 则像水库中的每个独立的水池子，分别存放不同种类的水。一个大水库由一个或多个这样的水池子组成。

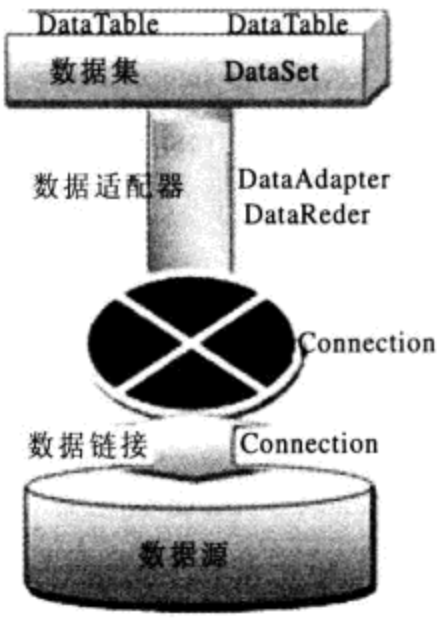


图 10.1　ADO.NET 对象模型

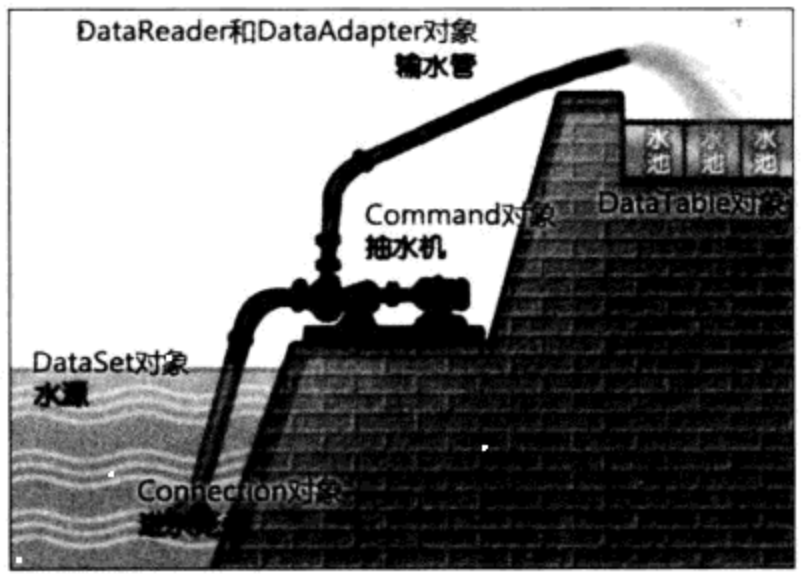


图 10.2　趣味理解 ADO.NET

10.2　使用 Connection 连接数据库

专题讲座：光盘\MR\Video\10\Connection 建立连接数据库.exe

➤➤➤视频速递：通过本讲座读者可以轻松掌握如何连接数据。

所有对数据库的访问操作都是从建立数据库连接开始的。在打开数据库之前，必须先设置好连接字符串（ConnectionString），然后再调用 Open 方法打开连接，此时便可对数据库进行访问，最后调用 Close 方法关闭连接，本节将详细介绍如何与数据库建立连接。

10.2.1　熟悉 Connection 对象

所有对数据库的访问操作都是从建立数据库连接开始的。Connection 对象就是用于连接数据库和管理数据库事务。Connection 对象还提供一些方法允许程序员与数据源建立连接或者断开连接。微软公司提供了 4 种连接对象，以便针对不同的数据库提供最佳的访问效能，如图 10.3 所示。具体说明如表 10.2 所示。

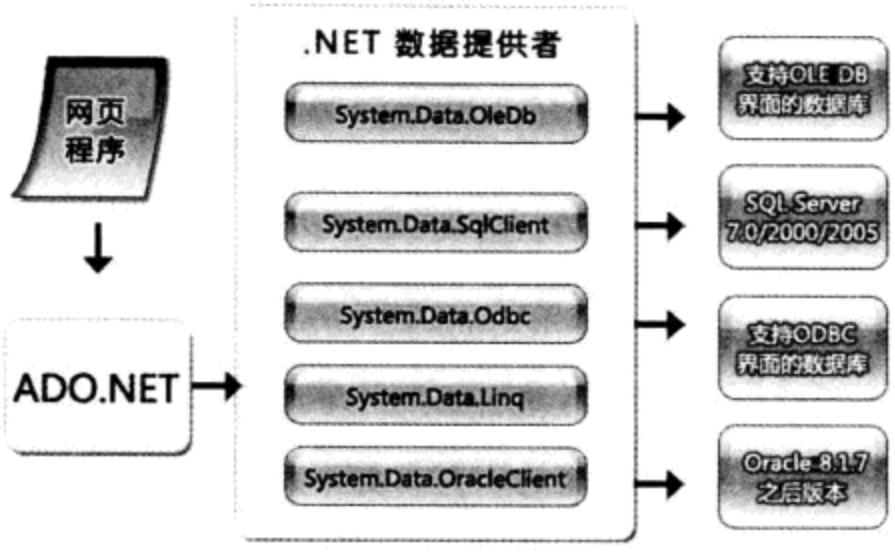


图 10.3　ADO.NET 针对不同数据库提供不同连接对象

表 10.2　4 种数据提供程序的连接对象及说明

连 接 对 象	说　明
SQL Server	.NET 数据提供程序的 SqlConnection 连接对象，命名空间 System.Data.SqlClient.SqlConnection
OLE DB	.NET 数据提供程序的 OleDbConnection 连接对象，命名空间 System.Data.OleDb.OleDbConnection
ODBC	.NET 数据提供程序的 OdbcConnection 连接对象，命名空间 System.Data.Odbc.OdbcConnection
Oracle	.NET 数据提供程序的 OracleConnection 连接对象，命名空间 System.Data.OracleClient.OracleConnection

本章所讲的关于 ADO.NET 相关技术的所有实例都将以 SQL Server 数据库为例，引入的命名空间即为 System.Data.SqlClient!

10.2.2　连接数据库字符串

为了让连接对象知道欲访问的数据库文件在哪里，用户必须将这些信息用一个字符串加以描述。连接字符串中需要提供的必要信息包括服务器的位置、数据库的名称和数据库的身份验证方式（Windows 集成身份验证或 SQL Server 身份验证）。另外，还可以指定其他操作的信息，诸如连接超时等。下面详细介绍数据库连接字符串常用的参数及描述，如表 10.3 所示。

表 10.3　数据库连接字符串常用的参数及描述

参　数	说　明
Provider	这个属性用于设置或返回连接提供程序的名称，仅用于 OleDbConnection 对象
Connection Timeout	在终止尝试并产生异常前，等待连接到服务器的连接时间长度（以秒为单位）。默认值是 15 秒
Initial Catalog 或 Database	数据库的名称
Data Source 或 Server	连接打开时使用的 SQL Server 名称，或者是 Microsoft Access 数据库的文件名
Password 或 pwd	SQL Server 账户的登录密码
User ID 或 uid	SQL Server 登录账户
Integrated Security	此参数决定连接是否是安全连接。可能的值有 True、False 和 SSPI（SSPI 是 True 的同义词）

连接字符串通常由分号隔开的名称和值组成，它指定数据库运行库的设置。在连接数据库时只要使用几个主要的参数就可以完成连接数据库的操作。下面分别以连接 SQL Server 2000/2005 数据库、Access 数据库和 Oracle 数据库为例，看一下其连接字符串的设置。

（1）连接 SQL Server 2000/2005 数据库字符串

字符串连接基本语法格式如下：

```
string connectionString="Ser ver=服务器名;User Id=用户;Pwd=密码;DataBase=数据库名称"
```

【例 10.1】 连接 SQL Server 2005 的字符串如下，实现通过 ADO.NET 连接本地 SQL Server 2005 中的 master 数据库：

```
//创建连接数据库的字符串
string SqlStr = "Server= DONET\\DONET2005;User Id=sa;Pwd=;DataBase=master";
```

学习手册 说明 在连接 SQL Server 2005 数据库服务器时，server 参数需要指定服务器所在的机器名称（IP 地址）和数据库服务器的实例名称。例如上述代码：DONET 为计算机名称；DONET2005 为数据库服务器的实例名称。

（2）连接 Access 数据库字符串

字符串连接基本语法格式如下：

```
string connectionString=“provide=提供者; Data Source=Access 文件路径”;
```

学习手册 说明 使用 OleDb 方式连接 Access 数据库时，需要指定 Provide 和 DataSource 两个参数。Provide 指数据提供者；DataSource 指 Access 文件路径。

【例 10.2】 本示例代码实现的是一个连接 Access 数据库的字符串：

```
String  connectionStirng="provide=Microsoft.Jet.OLEDB.4.0;"+@"Data  Source=
C:\myData\ db_access. mdb";
```

（3）连接 Oracle 数据库字符串

【例 10.3】 本示例演示了如何在 ASP.NET 应用程序中连接 Oracle 数据库的字符串。

```
string strCon = " Data Source=Oracle9i;Integrated Security=yes";
```

学习手册 注意 使用 Oracle.NET Framework 数据提供程序，要求必须先在系统上安装 Oracle 客户端软件（9.1.7 版或更高版本），才能连接到 Oracle 数据源。

很多时候，用户指定连接字符串的时候，并不会像以上所给的示例那样直接定义在一个字符串里，就传给 Connection 对象，更好的做法是将这个字符串写到项目的 Web.Config 配置文件中。这样在需要修改这个字符串的时候，就不用修改任何代码，而直接从 Web.Config 文件中修改就可以了。

将数据库连接字符串存放在应用程序的配置文件（即 Web.Config）中，代码如下所示。

```
<configuration>
//在 Web.Config 文件中配置数据库连接字符串
<appSetting>
```

```
<add key="strconnection" value=" server=(local);database=stu;uid=sa;pwd=">
</appSetting>
</configuration>
```

那么在上述典型的代码中，连接的字符串将改写如下（其他代码不变）。

```
string connectionString=ConfigurationSetting.AppSetting["strconnection"];
```

10.2.3　使用 SqlConnection 对象连接数据库

调用 Connection 对象的 Open 或 Close 方法可以打开或关闭数据库连接，而且必须在设置好连接字符串后才能调用 Open，否则 Connection 对象不知道要与哪一个数据库建立连接。这里一定要明确一点：必要时再打开连接并尽早关闭连接。

数据库联机资源是有限的，因此在需要的时候再打开连接，且一旦使用完就应该尽早地关闭连接，把资源归还给系统。打个简单的比喻：就像全家人只有一台小轿车，若没有用车就应该尽早地开回来，以方便其他有需要的家人使用。否则若某人开出去游玩一周，那这个星期其他所有的家人就都没有车可用了。下面以 SqlConnection 对象连接 SQL Server 2000/2005 数据库为例说明如何打开相应的数据库连接。

ADO.NET 专门提供了 SQL Server .NET 数据提供程序用于访问 SQL Server 数据库。SQL Server .NET 数据提供程序提供了专用于访问 SQL Server 10.0 及其更高版本数据库的数据访问类集合，例如 SqlConnection、SqlCommand、SqlDataReader 及 SqlDataAdapter 等数据访问类。

SqlConnection 类是用于建立与 SQL Server 2000/2005 服务器连接的类。其语法格式如下：

```
SqlConnection con = new SqlConnection("Server=服务器名;User Id=用户;Pwd=密码;DataBase=数据库名称");
```

实例位置：光盘\MR\Instance\10\10.1

【例 10.4】 创建一个数据库连接字符串，并通过 SqlConnection 对象连接到本地 SQL Server 2005 数据库中的 Master 数据库，同时应用 SqlConnection 对象的 State 属性判断数据库的连接状态。实例运行效果如图 10.4 所示。

图 10.4　用 SqlConnection 对象的 State 属性判断连接状态

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面的 Page_Load 事件中应用 SqlConnection 对象的 State 属性判断数据库的连接状态，代码如下所示：

```
//引入命名空间
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //创建连接数据库的字符串
```

```
            string SqlStr = @"Server=WIN-GI7E47AND9R\LS;uid=sa;Pwd=;DataBase=master";
            SqlConnection con = new SqlConnection(SqlStr); //创建 SqlConnection 对象
            con.Open();                                    //打开数据库的连接
            if (con.State == System.Data.ConnectionState.Open) //判断连接状态
            {
                Response.Write("SQL Server 数据库连接开启！<p/>");
                con.Close();                               //关闭数据库的连接
            }
            if(con.State==System.Data.ConnectionState.Closed)  //判断连接是否关闭
            {
                Response.Write("SQL Server 数据库连接关闭！<p/>");
            }
        }
    }
```

在连接 SQL Server 2005 数据库服务器时，server 参数需要指定服务器所在的机器名称（IP 地址）和数据库服务器的实例名称。例如上述代码中 server 参数是“WIN-GI7E47AND9R\LS”；WIN-GI7E47AND9R 为计算机名称；LS 为数据库 2005 服务器的实例名称。

10.2.4 使用 OleDbConnection 对象连接数据库

（1）使用 OleDb 方式连接 Access 数据库

OLE DB 数据源包含具有 OLE DB 驱动程序的任何数据源，如 SQL Server、Access、Excel、Oracle 等数据源。OLE DB 数据源连接字符串必须提供 Provide 属性及其值。其语法格式参见 10.2.2 节“连接数据库字符串”中连接 Access 数据库的字符串。

图 10.5 用 OleDbConnection 对象的 State 属性判断连接状态

实例位置：光盘\MR\Instance\10\10.2

【例 10.5】 创建一个数据库连接字符串，并通过 OleDbConnection 对象连接到本地 Access 数据库，同时应用 OleDbConnection 对象的 State 属性判断数据库的连接状态。实例运行效果如图 10.5 所示。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面的 Page_Load 事件中应用 OleDbConnection 对象的 State 属性判断数据库的连接状态，代码如下所示：

```
//引入命名空间
using System.Data.OleDb;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string StrLoad = Server.MapPath("db.mdb");    //获取指定数据库文件的路径
        OleDbConnection myConn = new OleDbConnection("Provider=Microsoft.
        Jet.OLEDB.4.0; Data Source=" + StrLoad + ";"); //创建 OleDbConnection 对象
```

```
            myConn.Open();                                          //打开数据库连接
            if (myConn.State == System.Data.ConnectionState.Open)
            {
                Response.Write("Access 数据库连接开启！<p/>");
                myConn.Close();                                     //关闭数据库的连接
            }
            if (myConn.State == System.Data.ConnectionState.Closed)
            {
                Response.Write("Access 数据库连接关闭！<p/>");
            }
        }
    }
```

（2）使用 OleDb 方式连接 SQL Server 数据库

使用 OleDb 方式连接 SQL Server 数据库的语法格式如下：

```
OleDbConnection myConn = new OleDbConnection("Provider=OLE DB 提供程序的名称;Data Source=存储要连接数据库的 SQL 服务器; Initial Catalog=连接的数据库名;Uid =用户名;Pwd=密码");
```

实例位置：光盘\MR\Instance\10\10.3

【例 10.6】创建一个数据库连接字符串，并通过 OleDbConnection 对象连接到本地 SQL Server 数据库，运行结果如图 10.6 所示。

图 10.6　打开数据库并输出其状态信息

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面的 Page_Load 事件中应用 OleDbConnection 对象的 State 属性判断数据库的连接状态，代码如下所示：

```
//引入命名空间
using System.Data.OleDb;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        OleDbConnection myConn = new OleDbConnection(); //创建 OleDbConnection 对象
        myConn.ConnectionString = @"provider=SQLOLEDB;Data Source=WIN-
        GI7E47AND9R\LS; Initial Catalog= master;User Id=sa;pwd="; //设置连接字符串
        myConn.Open();                                          //打开连接
        Response.Write(myConn.State);                           //将数据库连接状态输出到页面
        myConn.Close();                                         //关闭连接
    }
}
```

10.3　使用 Command 对象操作数据

专题讲座：光盘\MR\Video\10\使用 Command 对象操作数据.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握如何对数据进行添加、修改和删除等

操作。

Command 对象是一个对 SQL 语句和存储过程进行执行的类。通过它，用户可以实现对数据的添加、删除、更新、查询等各种操作。本节作者将带领大家熟悉 Command 对象，并通过该对象操作数据。

10.3.1 熟悉 Command 对象

使用 Connection 对象与数据源建立连接后，可使用 Command 对象对数据源执行查询、添加、删除和修改等各种操作，操作实现的方式可以是使用 SQL 语句，也可以是使用存储过程。根据所用的.NET Framework 数据提供程序的不同，Command 对象也可以分成 4 种，分别是 SqlCommand 、 OleDbCommand 、 OdbcCommand 和 OracleCommand，在实际的编程过程中应根据访问的数据源不同，选择相应的 Command 对象。以操作 SQL Server 数据库为例，SqlCommand 对象应用模型如图 10.7 所示。

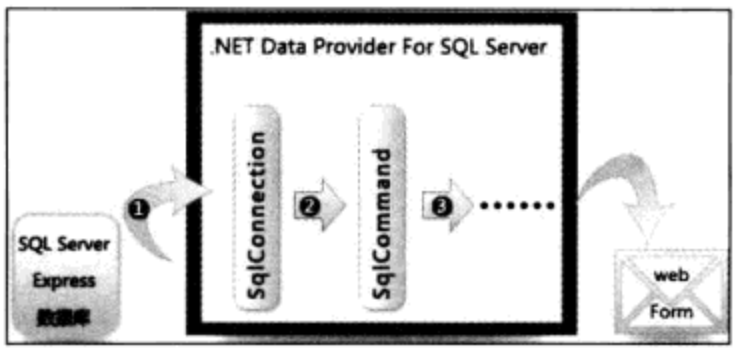


图 10.7 SqlCommand 对象的应用模型

Command 对象常用属性如表 10.4 所示。

表 10.4 Command 对象常用属性及说明

属 性	说 明
CommandType	获取或设置 Command 对象要执行命令的类型
CommandText	获取或设置要对数据源执行的 SQL 语句或存储过程名或表名
CommandTimeOut	获取或设置在终止对执行命令的尝试并生成错误之前的等待时间
Connection	获取或设置此 Command 对象使用的 Connection 对象的名称
Parameters	获取 Command 对象需要使用的参数集合

下面对创建 Command 对象必须指定的三个属性（Connection、CommandType 和 CommandText）详细介绍如下。

（1）Connection 属性

以操作 SQL Server 数据库为例，执行命令操作的对象为 SqlCommand，然而 SqlCommand 本身无法建立与数据库的连接，若要对数据库的内容进行访问，只能通过 SqlConnection 对象建立连接，然后再利用 Connection 属性进行记录。

（2）CommandType 和 CommandText 属性

从表中可以知道 CommandText 既可以是 SQL 语句，也可以是存储过程，还可以是表的名字。要使用不同种类的 CommandText，只需要设置相应的 CommandType 即可。表 10.5 列出了 3 种不同的 CommandType。

表 10.5 3 种不同的 CommandType

类 型	描 述
CommandType.Text	这是 CommandType 默认值，它指示执行的是 SQL 语句，为 CommandText 指定 SQL 字符串。默认情况下 Command 对象使用这种类型

续表

类　型	描　述
CommandType.StoredProcedure	这个值指示执行的是存储过程，需要为 CommandText 指定一个存储过程的名称
CommandType.TableDirect	为 CommandText 指定一个数据表名称，这个值指示用户将得到这个表中的所有数据（建议用户还是直接写 SQL 语句来实现这个功能比较可靠）

【例 10.7】下面的示例代码展示了如何使用 Command 对象执行一个 SQL 语句的查询操作（以 SQL Server 数据库为例）。

```
//引入命名空间
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //创建数据库连接对象
        SqlConnection conn = new SqlConnection("Server=WIN-GI7E47AND9R\LS;
        User Id=sa; Pwd=; DataBase=master");
        SqlCommand comm = new SqlCommand();    //创建命令对象 SqlCommand
        comm.Connection = conn;                //设置连接对象
        comm.CommandType = CommandType.Text;   //设置 Command 对象执行语句的类型
        comm.CommandText = "select * from Employee";   //设置执行的语句
        ……
    }
}
```

【例 10.8】如果熟悉了 Command 对象后可以使用一种更简单的方式完成上面的操作：

```
//创建数据库连接对象
SqlConnection conn = new SqlConnection("Server=WIN-GI7E47AND9R\LS;User
Id=sa;Pwd=; DataBase=master");                    //创建数据库连接对象
//创建命令对象 SqlCommand
SqlCommand comm = new SqlCommand("select * from Employee", conn);
```

同样地，需要执行存储过程的时候，可以使用下面的这种方式：

```
//创建数据库连接对象
SqlConnection conn = new SqlConnection("Server=WIN-GI7E47AND9R\LS;User Id=sa;
Pwd=; Data-Base=master");
//创建命令对象 SqlCommand
SqlCommand comm = new SqlCommand("GetEmployee", conn);
comm.CommandType = CommandType.StoredProcedure;   //设置执行的方式是执行存储过程
```

上面的这些示例只是简单地定义出了 Command 对象，并为它们设置了一些必要属性如 Connection、CommandText 等，而并没有真正地执行这些操作。如果想执行 Command 操作，有 3 种方法可以使用，如表 10.6 所示，这 3 种方法的主要不同之处在于它们的返回值。

表 10.6　Command 对象常用的执行方式

类　型	描　述
ExecuteNonQuery()	用于执行非 SELECT 命令，比如 INSERT、DELETE 或者 UPDATE 命令，返回三个命令所影响的数据的行数。也可以用 ExecuteNonQuery()来执行一些数据定义命令，比如新建、更新、删除数据库对象（如表、索引等）

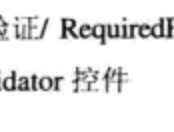
续表

类　型	描　述
ExecuteScalar()	用于执行 SELECT 查询命令，返回数据中第一行第一列的值。这个方法通常用来执行那些用到 COUNT()或 SUM()函数的 SELECT 命令
ExecuteReader()	执行 SELECT 命令，并返回一个 DataReader 对象。这个 DataReader 是向前只读的数据集

表 10.6 中这 3 种方法非常重要，如果要使用 ADO.NET 完成某种数据库操作，一定会用到上面这些方法，这 3 种方法没有任何的优劣之分，只是使用场合不同罢了，所以一定要弄清楚它们的返回值类型以及使用方法，以适当地使用它们。

10.3.2　使用 Command 对象添加数据

以操作 SQL Server 数据库为例，向数据库中添加记录时，首先要创建 SqlConnection 对象连接数据库，然后定义添加记录的 SQL 字符串，最后调用 SqlCommand 对象的 ExecuteNonQuery 方法执行记录的添加操作。



再次强调一下，本章以下所有实例都将以操作 SQL Server 数据库为例，在后面列举实例时将不再给予提示。

图 10.8　添加记录后的结果图

实例位置：光盘\MR\Instance\10\10.4

【例 10.9】 本实例主要讲解在 ASP.NET 应用程序中如何向数据库添加记录。执行程序，实例运行结果如图 10.8 所示；输入用户信息和年龄后，单击“添加到数据库”按钮，将用户信息和用户年龄添加到数据库中。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上分别添加两个 TextBox 控件、一个 Button 控件和一个 RequiredFieldValidator 控件，它们的属性设置如表 10.7 所示。

表 10.7　Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用　途
标准/TextBox 控件	txtuser	均为默认设置	输入用户信息
	txtage	均为默认设置	输入用户年龄
标准/Button 控件	btninsert	Text 属性设置为“添加到数据库”	执行页面提交的功能
验证/ RequiredField-Validator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为“txtuser”	要验证的控件的 ID 为 txtuser
	ErrorMessage 属性设置为“*”	显示的错误信息为“*”	ErrorMessage 属性设置为“*”
	SetFocusOnError 属性设置为 True	验证无效时，在该控件上设置焦点	SetFocusOnError 属性设置为 True

（2）在“添加到数据库”按钮的 Click 事件下，使用 Command 对象将文本框中的值添加到数据库中，其代码如下所示：

```
protected void btninsert_Click(object sender, EventArgs e)
 {
    //创建数据库连接对象
    SqlConnection conn = new SqlConnection(@"Server=WIN-GI7E47AND9R\LS;User
    Id=sa;Pwd=; DataBase=db_AspnetLearn");
    string strsql = "insert into tb_user_07(username,userage) values('" +
    txtuser.Text + "','" + txtage. Text + "')";
    SqlCommand comm = new SqlCommand(strsql, conn);     //创建 SqlCommand 对象
    if (conn.State.Equals(ConnectionState.Closed))      //打开数据库连接
    { conn.Open(); }
    //判断 ExecuteNonQuery 方法返回的参数是否大于 0，大于 0 表示添加成功
    if (Convert.ToInt32(comm.ExecuteNonQuery()) > 0)    //如果返回值大于 0
    {
         lblinfo.Text="添加成功！";                      //说明操作成功
    }
    else                                                //如果返回值不大于 0
    {
       lblinfo.Text="添加失败！";                       //说明操作失败了
    }
    if (conn.State.Equals(ConnectionState.Open))        //关闭数据库连接
       conn.Close();
}
```

在讲解 Connection 对象时，我们提到一点：必要时才打开连接并尽早关闭连接。依照这个原则，在还没有执行 SQL 语句之前，可以先建立并设置 SqlConnection 对象，但不需要马上打开连接。本实例是建立 SqlCommand 对象之后才显式地打开数据库连接。

程序执行添加操作完成后，读者可以在本实例所应用的 SQL Server 2005 数据库管理器中查看是否真正地向数据库 db_AspnetLearn 的 tb_user_07 表中添加了一条数据信息，查看操作分别如图 10.9 和图 10.10 所示。

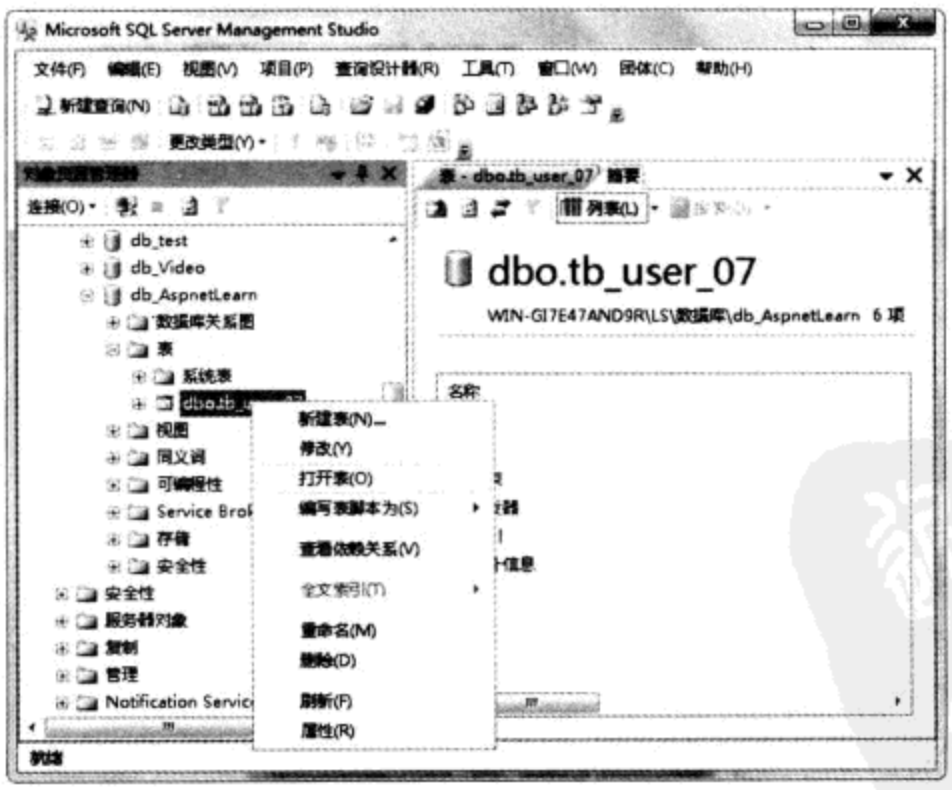


图 10.9　打开 tb_user_07 数据表

图 10.10　新增一条数据信息

10.3.3 使用 Command 对象修改数据

10.3.2 节的实例实现了向数据库添加一条数据信息，接下来看一下如何对添加的这条信息进行修改操作。修改数据库中的记录时，首先创建 SqlConnection 对象连接数据库，然后定义修改数据的 SQL 字符串，最后调用 SqlCommand 对象的 ExecuteNonQuery 方法执行记录的修改操作。

实例位置：光盘\MR\Instance\10\10.5

【例 10.10】下面通过一个实例讲解一下在 ASP.NET 应用程序中如何修改数据表中的记录。示例运行结果如图 10.11 所示，向文本框中输入修改内容，单击“执行修改操作”按钮。修改信息。打开 SQL Server 2005 数据库，可以查看到修改后数据库表中的数据结果如图 10.12 所示。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上分别添加 3 个 TextBox 控件、一个 Button 控件和两个 RequiredFieldValidator 控件，它们的属性设置如表 10.8 所示。

图 10.11 修改数据的成功

表 - dbo.tb_user_07 摘要 （修改之前的数据）

ID	username	userage
1	洪杰	29

表 - dbo.tb_user_07 摘要 （修改之后的数据）

ID	username	userage
1	洪杰	28

图 10.12 数据表中的数据

表 10.8 Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用途
标准/TextBox 控件	txtuser	均为默认设置	输入用户信息
	txtage	均为默认设置	输入用户年龄
	txtid	均为默认设置	输入要修改信息的 ID 值
标准/Button 控件	btnupdate	Text 属性设置为“执行修改操作”	执行页面修改的功能
验证/ RequiredFieldValidator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为“txtuser”	要验证的控件的 ID 为 txtuser
	ErrorMessage 属性设置为“*”	显示的错误信息为“*”	ErrorMessage 属性设置为“*”
	SetFocusOnError 属性设置为 True	验证无效时，在该控件上设置焦点	SetFocusOnError 属性设置为 True

（2）在 Web.config 文件中配置数据库连接字符串。在配置节<configuration>下的子配置节<appSettings>中添加连接字符串，其代码如下所示：

```
<appSettings>
    <add key="conStr" value="server=WIN-GI7E47AND9R\LS;database=db_AspnetLearn;
    uid=sa;pwd=;"/>
</appSettings>
```

（3）当输入要修改的信息后，单击“执行修改操作”按钮修改数据，在按钮的 Click 事件下，编写如下代码。

```
protected void btnupdate_Click(object sender, EventArgs e)
{
    //获取配置文件中的数据库链接配置
    string myStr = ConfigurationManager.AppSettings["conStr"].ToString();
    SqlConnection myConn = new SqlConnection(myStr);        //建立数据库连接
    string sqlStr = "update tb_user_07 set username='" + txtuser.Text + "',
    userage='" + txtage.Text + "' where ID="+int.Parse(txtid.Text.Trim());
                                                            //定义查询 SQL 语句
    SqlCommand myCmd = new SqlCommand(sqlStr, myConn);      //初始化查询命令
    if (myConn.State == ConnectionState.Closed)             //打开数据库连接
    { myConn.Open(); }
    int records = Convert.ToInt32(myCmd.ExecuteNonQuery()); //执行 SQL 语句并返回值
    if (records > 0)
        Response.Write("修改成功！更新了" + records.ToString() + "条数据！");
    else
        Response.Write("信息提示：修改失败！");
    myCmd.Dispose();
    myConn.Close();                                         //关闭数据库连接
}
```

10.3.4　使用 Command 对象删除数据

删除数据库中的记录时，首先创建 SqlConnection 对象连接数据库，然后定义删除字符串，最后调用 SqlCommand 对象的 ExecuteNonQuery 方法完成记录的删除操作。

可以看出删除的操作步骤比数据的查询和保存更简单，只要一个简单的 SQL 语句就能轻松完成。事实上，删除操作并不是那么简单，通常需要花一定时间在删除验证上，因为删除是一件对数据影响比较大的事情。比如要避免用户的误删除、误操作，大多数应用程序在用户要删除一条记录时都会弹出一个对话框，询问用户是否真的要删除。这虽然对用户和开发者来说都有点麻烦，但这个麻烦是值得的，它能避免很多误操作的发生。

另外一种避免数据损失的方式是在执行删除操作时不使用 DELETE 语句，而使用 UPDATE 语句。这里所说的删除并不是真正的删除，而是通过 UPDATE 语句更改数据的某个状态，使数据变成“透明的”，用户在查询的时候检索不到这些“透明的”数据，给用户的感觉是这些数据被删除了，而实际上它们还存在于数据库中，当有特殊需要时可以恢复这些数据，供用户使用，这里通过一个实例来理解这种删除方式。

实例位置：光盘\MR\Instance\10\10.6

【例 10.11】 下面通过一个实例讲解在 ASP.NET 应用程序中删除数据库中指定的数据。在文本框中输入要删除的数据在数据表中的 ID 号，然后单击“执行删除操作”按钮

删除该数据，运行结果如图 10.13 所示，数据库的数据表变化如图 10.14 所示。

图 10.13　删除数据库中的数据

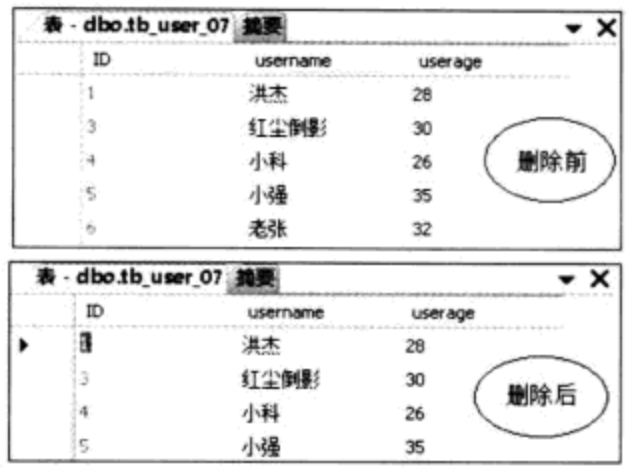


图 10.14　删除数据前后的表中数据变化

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加一个文本框控件、一个 Button 按钮和一个 Label 控件，分别用来输入要删除的数据的 ID 号、执行删除功能和显示提示信息。

（2）当在文本框输入 ID 号码后，单击“执行删除操作”按钮，将会触发 Button 控件的 Click 事件，在该事件下，编写如下代码删除指定信息。

```
protected void btndelete_Click(object sender, EventArgs e)
{
    //创建数据库连接对象
    SqlConnection con=new SqlConnection(ConfigurationManager.AppSettings
    ["conStr"].ToString());
    try
    {
        int id = Convert.ToInt32(this.txtid.Text.Trim());
        if (id > 0)                                          //简单的数据验证
        {
            SqlCommand cmd = new SqlCommand();       //创建 SqlCommad 命令对象
            cmd.Connection = con;
            cmd.CommandText = string.Format("delete from tb_user_07 where ID=
            {0}", id);
            con.Open();                          //打开数据库连接
            int affectRow = cmd.ExecuteNonQuery();
                                                 //执行 ExecuteNonQuery()，返回影响的数据行数
            if(affectRow>0)
            this.lblinfo.Text = "删除成功！";
        }
        else
        {
            lblinfo.Text = "编号必须是正整数！"; //当用户输入的编号不为正整数时给予提示
        }
    }
    catch
    {
        lblinfo.Text = "编号只能是数字！";       //显示错误提示
    }
    finally
    {
```

```
            con.Close();                                    //关闭数据库连接
        }
    }
```

10.3.5　使用 Command 对象调用存储过程

存储过程可以使管理数据库和显示数据库信息等操作变得非常容易。它存储在数据库内，在程序中可以通过 SqlCommand 对象来调用，其执行速度比 SQL 语句快，同时还保证了数据的安全性和完整性，下面将通过实例演示如何调用存储过程。

图 10.15　使用 Command 对象调用存储过程

实例位置：光盘\MR\Instance\10\10.7

【例 10.12】 在前面讲解使用 Command 对象添加数据实例时用 SQL 语句直接向 tb_user_07 表中添加了一条数据，下面我们将通过调用存储过程的方式来重新实现这个功能，以便说明存储过程的调用方式。实例运行结果如图 10.15 所示。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上分别添加两个 TextBox 控件、一个 Button 控件和一个 RequiredFieldValidator 控件，它们的属性设置如表 10.9 所示。

表 10.9　Default.aspx 页控件属性设置及说明

控 件 类 型	控 件 名 称	主要属性设置	用　途
标准/TextBox 控件	txtuser	均为默认设置	输入用户信息
	txtage	均为默认设置	输入年龄
标准/Button 控件	btninsert	Text 属性设置为“调用存储过程添加数据”	执行页面提交的功能

（2）在“调用存储过程添加数据”按钮的 Click 事件下，使用 Command 对象调用存储过程将文本框中的值添加到数据库中，其代码如下所示：

```
protected void btninsert_Click(object sender, EventArgs e)
{
    string strCon = ConfigurationManager.AppSettings["conStr"].ToString();
                                                        //定义数据库连接字符串
    SqlConnection con = new SqlConnection(strCon); //创建数据库连接对象
    SqlCommand cmd = new SqlCommand("insertuser", con);
                                                        //创建命令对象并指定存储过程名称
    cmd.CommandType = CommandType.StoredProcedure; //指定命令类型为存储过程
    //先清空参数数组，然后再逐项为存储过程中的变量赋值
    cmd.Parameters.Clear();
    //添加参数
    cmd.Parameters.Add(new SqlParameter("@username", SqlDbType.VarChar, 50));
    cmd.Parameters["@username"].Value =txtuser.Text; //参数赋值
    cmd.Parameters.Add(new SqlParameter("@userage", SqlDbType.Int,20));
    //假设用户添加的价格为纯数字，执行以下代码
```

```
    cmd.Parameters["@userage"].Value = int.Parse(txtage.Text);
    if (con.State == ConnectionState.Closed)              //获取连接状态
    { con.Open(); }                                       //如果数据库处于打开状态
    int records = Convert.ToInt32(cmd.ExecuteNonQuery()); //执行存储过程
    if (records > 0)                                      //如果返回值大于 0
       lblinfo.Text="添加数据成功！增加了" + records.ToString() + "条数据！";
                                                          //说明操作成功
    else                                                  //否则
        lblinfo.Text="信息提示：添加失败！";               //操作失败
    cmd.Dispose();
    con.Close();                                          //关闭数据连接
}
```

在数据库中编写存储过程，实现在网页中向存储过程传递参数，然后在存储过程中实现插入操作，代码如下所示：

```
CREATE PROCEDURE insertuser                    // insertuser 是存储过程名
@username varchar(50),                         //参数
@userage int                                   //参数
AS
BEGIN
insert into tb_user_07(username,userage) values(@username,@userage) //执行的操作
END
GO
```

10.3.6 情景应用 1：使用 Command 对象添加联系人管理数据

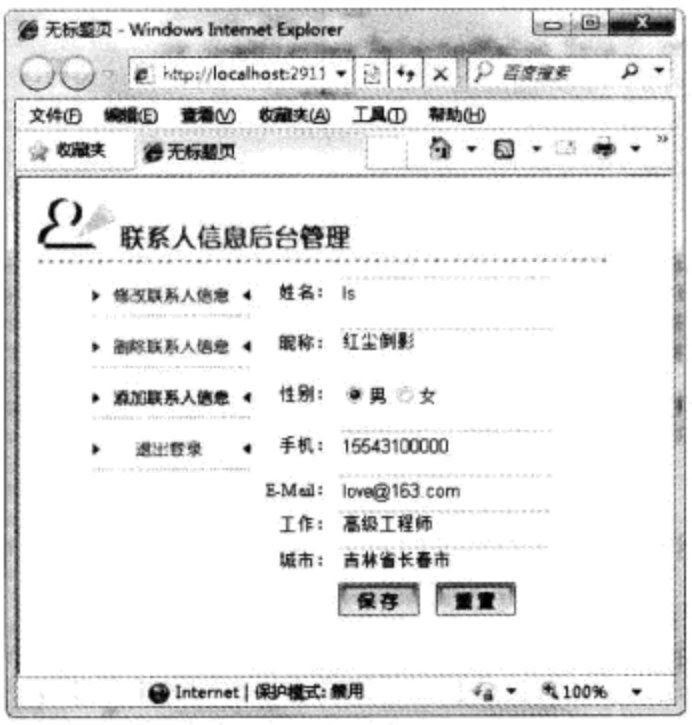


图 10.16 使用 Sqlcommand 对象添加联系人管理数据

为了使读者更好地掌握如何使用 Command 对象向数据库中添加数据，下面通过一个实例进行详细演示，希望读者能够认真学习。

实例位置：光盘\MR\Instance\10\10.8

【例 10.13】 创建一个 Web 网站，添加一个名为 LinkManManage.aspx 的 Web 窗体，主要是用来实现添加联系人信息，如联系人的姓名、昵称和性别等信息，其运行结果如图 10.16 所示。

(1)在 LinkManManage.aspx 前台页面中添加一个 ID 为 imgAdd 的 ImageButton 控件，用来执行添加操作。

(2)双击该按钮触发其 imgAdd_Click 事件，实现添加联系人详细信息，代码如下所示：

```
protected void imgAdd_Click(object sender, ImageClickEventArgs e)
{
    string userName = this.txtName.Text;                  //获取用户名
    string nickName = this.txtNickName.Text;              //获取昵称
    string sex = "";
    if (radlistSex.SelectedValue.Trim() == "男")          //获取联系人性别
```

```
{
    sex = "男";
}
else   {
    sex = "女";
}
string phone = this.txtphone.Text;                          //获取手机号
string email = this.txtMail.Text;                           //获取邮件地址
string work = this.txtWork.Text ;                           //获取地址信息
string city = this.txtCity.Text;                            //获取所在城市
//创建 SQL 语句，用来添加用户详细信息
string sqlInsert="insert into AddLinkMen values('"+userName+"',
'"+nickName+"','"+sex+"', '"+phone+"','"+email+"','"+work+"','"+city+"')";
OperateDataBase odb = new OperateDataBase();                //实例化类对象
bool add = odb.ExceSql(sqlInsert);      //调用 odb 类的 ExceSql 方法执行添加操作
if (add == true)                        //如果返回值为 true 说明操作成功
{
   Response.Write("<script language=javascript>alert('添加成功!');location=
   'LinkManManage. aspx'</script>");
}
else
{
   Response.Write("<script language=javascript>alert('添加失败!');location=
  'javascript: history. go(-1)'</script>");
    }
}
```

10.3.7　情景应用 2：使用 Command 对象修改联系人管理数据

添加数据之后，如何修改数据呢？本实例主要为读者演示如何使用 Command 对象修改指定的数据，使读者更好地掌握 Command 对象的用法。

实例位置：光盘\MR\Instance\10\10.9

【例 10.14】实现了联系人的添加操作后，接下来就可以对联系人的信息进行修改操作，创建一个名为 UpdateLinkMan.aspx 的页，主要用来实现联系人信息的修改操作，其实际运行结果如图 10.17 所示。

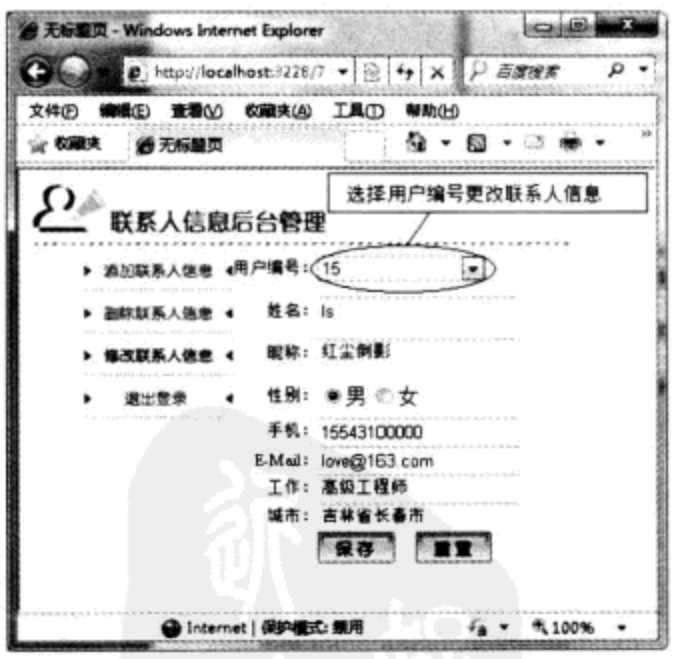


图 10.17　使用 SqlCommand 对象修改联系人管理数据

（1）在 UpdateLinkMan.aspx 前台页面中，添加一个 DropDownList 控件和一个 ImageButton 控件，分别用于选择用户编号和执行修改操作。

（2）双击 UpdateLinkMan.aspx 页面中用于执行修改操作的 ImageButton 控件，触发其 imgUpdate_Click 事件代码，主要程序代码如下所示：

```
protected void imgUpdate_Click(object sender, ImageClickEventArgs e)
{
    string userName = this.txtName.Text;                    //获取用户名
    string nickName = this.txtNickName.Text;                //获取昵称
```

```
string sex = "";
if (radlistSex.SelectedValue.Trim() == "男")              //获取联系人性别
{
    sex = "男";
}
else
{
    sex = "女";
}
string phone = this.txtphone.Text;                        //获取手机号
string email = this.txtMail.Text;                         //获取邮件地址
string work = this.txtWork.Text;                          //获取地址信息
string city = this.txtCity.Text;                          //获取所在城市
string update_sql = "update AddLinkMen set userName='" + userName + "',
UserNickName='" + nickName + "',UserSex='" + sex + "',UserPhone='" + phone + "'"
       + ",UserEmail='" + email + "',UserAdress='" + work + "',UserCity='" +
       city + "' where UserID="+int.Parse(DropDownList1.SelectedItem.Text);";
bool update_data = odb.ExceSql(update_sql);    //根据返回值判断操作是否成功
if (update_data == true)                                  //如果返回 true 则操作成功
{
    Response.Write("<script language=javascript>alert('修改成功!');
    location='UpdateLinkMan. aspx'</script>");
}
else                                                      //返回 false 操作失败
{
   Response.Write("<script language=javascript>alert('修改失败!');
   location='javascript:history. go(-1)'</script>");
    }
}
```

学习手册 说明 修改数据库中记录时，首先创建 SqlConnection 对象连接数据库，然后定义修改数据的 SQL 字符串，最后调用 SqlCommand 对象的 ExecuteNonQuery 方法执行记录的修改操作。

10.3.8 情景应用 3：使用 Command 对象删除联系人管理数据

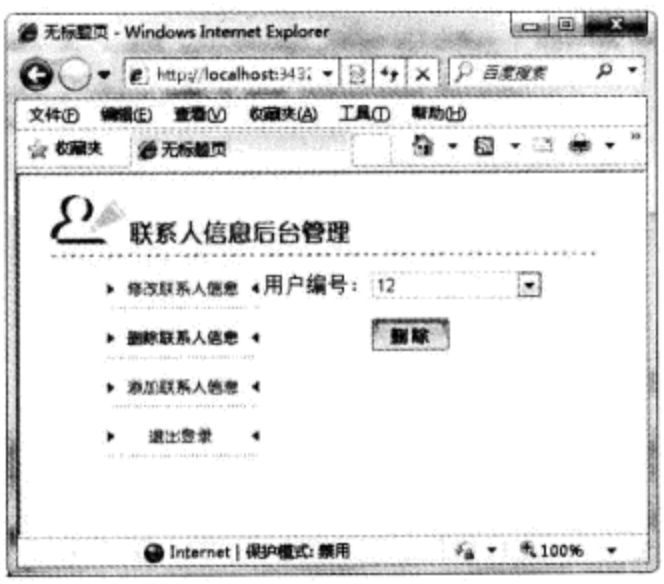


图 10.18 使用 SqlCommand 对象删除联系人管理数据

删除数据时，无论是 SQL 语句还是执行方式都相对简单，主要也是应用 Command 对象实现的。本实例还是以联系人信息后台管理系统为例，演示如何删除数据。

实例位置：光盘\MR\Instance\10\10.10

【例 10.15】 创建一个网站，并将首页改成 DeleteLinkMan.aspx。然后在该页中执行删除联系人信息的操作，运行结果如图 10.18 所示。

（1）在 DeleteLinkMan.aspx 页面中，添加一个 DropDownList 控件和一个 ImageButton 控件，分别用于选择用户编号和执行删除操作。

（2）双击页面中添加的 ImageButtion 按钮，触发其 imgDel_Click 事件，根据所选择的用户编号来对联系人信息进行删除操作，代码如下所示：

```
protected void imgDel_Click(object sender, ImageClickEventArgs e)
{
  if (DropDownList1.SelectedValue != "" && DropDownList1.SelectedIndex != 0)
                                                          //判断是否选择编号
  {
     string delete_sql = "delete from AddLinkMen where UserID='" + Convert.
     ToInt32(DropDownList1.       SelectedValue) + "'"; //根据选择的编号进行删除
      bool delete_data = odb.ExceSql(delete_sql); //执行 SQL 语句并获取返回值
      if (delete_data == true)                          //返回值为 true 则操作成功
      {
         Response.Write("<script language=javascript>alert('删除成功!');
         location='DeleteLinkMan. aspx'</script>");
      }
      else                                              //返回值为 false 则操作失败
      {
        Response.Write("<script language=javascript>alert('删除失败!');
        location='DeleteLinkMan. aspx'</script>");
          }
       }
else
{
       Response.Write("<script language=javascript>alert('暂无数据!');
       location='DeleteLinkMan. aspx'</script>");};
  }
}
```

ExceSql 方法是公共类中的自定义方法，读者可以参看源代码中公共类。

10.4　使用 DataReader 对象读取数据

专题讲座：光盘\MR\Video\10\使用 DataReader 对象读取数据.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握如何使用 DataReader 对象读取数据。

在介绍 Command 对象时，实现了对数据库进行添加、修改和删除操作，而没有对数据库中的数据进行查询显示操作，这里就可以使用 DataReader 对象来实现。当 Command 对象返回结果集时，需要使用 DataReader 对象来检索数据。DataReader 对象返回一个来自 Command 的只读的、只能向前的数据流。通过本节的学习，读者对 DataReader 对象会有一个彻底的认识。

10.4.1　理解节省内存的 DataReader 对象

DataReader 对象是一个简单的数据集，如其名一样，用于从数据源中读取只读的数据集，并常用于检索大量数据。根据.NET Framework 数据提供程序的不同，DataReader 也可以分成 SqlDataReader、OleDbDataReader 等几类。DataReader 每次只能在内存中保留一行，

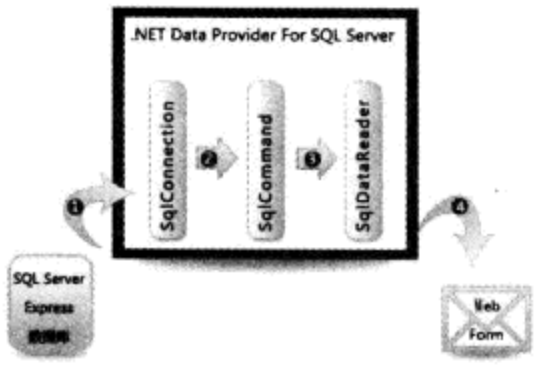


图 10.19 SqlDataReader 读取数据时，必须一直保持与数据库的连接

所以开销非常小。DataReader 对象可以形象地比喻成一根水管，直接把水送到需要水的用户那里或田地里。使用 SqlDataReader 对象为例读取数据时，必须一直保持与数据库的连接，所以也被称为连线模式，其架构如图 10.19 所示。

DataReader 是一个轻量级的数据对象，如果只需要将数据读出并显示，那它是最合适的工具，它的读取速度比稍后要讲解的 DataSet 对象要快，占用的资源也比 DataSet 少。但是，一定要铭记 DataReader 在读取数据的时候要求数据库保持在连接状态，读完数据之后才能断开连接。

通过 Command 对象的 ExecuteReader 方法从数据源中检索数据来创建 DataReader 对象。下面介绍 DataReader 对象的常用属性和方法。DataReader 对象常用属性如表 10.10 所示。

表 10.10 DataReader 对象常用属性及说明

属　性	说　明
HasRows	判断数据库中是否有数据
FieldCount	获取当前行的列数
RecordsAffected	获取执行 SQL 语句所更改、添加或删除的行数

DataReader 对象常用方法如表 10.11 所示。

表 10.11 DataReader 对象常用方法及说明

方　法	说　明
Read	使 DataReader 对象前进到下一条记录
Close	关闭 DataReader 对象
Get	用来读取数据集当前行的某一列的数据

若要判断是否有数据可供读取，可以先用 DataReader 对象的 HasRows 属性判断是否有数据可以回传，若有数据就传回 True，否则就传回 False，接着再调用 DataReader 的 Read 方法：它往下读取一条数据，若有数据就会传回 True，否则就传回 False，如图 10.20 所示。

10.4.2 使用 DataReader 对象读取数据

实例位置：光盘\MR\Instance\10\10.11

【例 10.16】 本实例使用一个 SqlDataReader 对象来读取数据库中表 tb_mrbccd 中所有的内容，并使用 Write 方法将所读取的内容显示在表格中，最后关闭 SqlDataReader 对象。实例运行结果如图 10.21 所示。

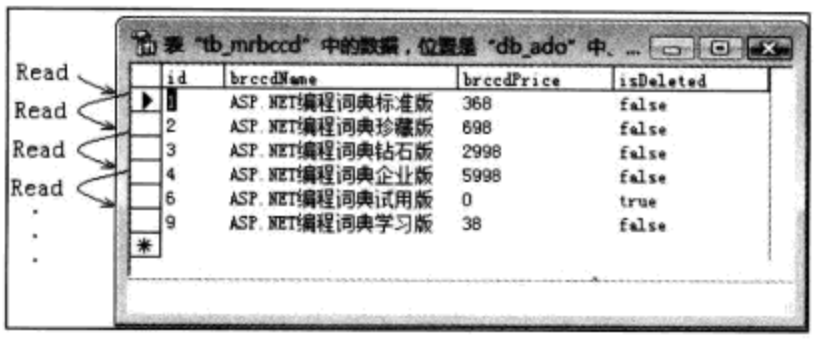


图 10.20　调用 SqlDataReader 的 Read 方法读取下一条数据

图 10.21　用 SqlDataReader 对象读出 SELECT 语句的查询结果

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）当页面加载时，在 Default.aspx 页的 Page_Load 事件下，使用 SqlDataReader 对象读取数据库中的信息，并将读取的数据信息显示出来，编写如下代码。

```
//引入命名空间
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //创建 SqlConnection 对象
        SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings
        ["conStr"]. ToString());
        //创建 SqlCommand 对象
        SqlCommand cmd = new SqlCommand("select * from AddLinkMen", conn);
        if (conn.State == ConnectionState.Closed)
        { conn.Open(); }                                    //打开数据库连接
        //接收 ExecuteReader 方法返回的 SqlDataReader 对象
        SqlDataReader sdr = cmd.ExecuteReader();
        Response.Write("<table border=1 align='center' "); //输出表格
        Response.Write("<tr><th>编号</th><th>用户名</th><th>昵称</th></tr>");
        try
        {
            if (sdr.HasRows)                                //判断是否有数据
            {
                //显示编程词典信息
                while (sdr.Read())                          //遍历 DataReader 时最常用的方式
                {
                    Response.Write("<tr>");
                    //按照顺序以列名指定要读取的项
                    Response.Write("<td align='center'>" + sdr["UserID"].ToString()
                    + "</td>");
                    Response.Write("<td align='center'>" + sdr["userName"].
                    ToString() + "</td>");
                    Response.Write("<td align='left'>" + sdr["UserNickName"].
                    ToString() + "</td>");
                    Response.Write("</tr>");
                }
            }
```

```
            Response.Write("</table>");
        }
        catch (SqlException ex)
        {
            Response.Write(ex.ToString());                    //异常处理
        }
        finally
        {
            sdr.Close();                                      //关闭 DataReader 对象
            conn.Close();                                     //断开数据库连接
        }
    }
}
```

读完数据后，务必把 DataReader 对象关闭，否则当 DataReader 对象尚未关闭时，DataReader 所使用的 Connection 对象就无法执行其他的动作了。

10.5 使用 DataSet 和 DataAdapter 查询数据

专题讲座：光盘\MR\Video\10\使用 DataSet 和 DataAdapter 查询数据.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握如何使用 DataSet 和 DataAdapter。

DataSet 可以将其看做是在内存中创建的一个小型关系数据库，它将数据库中的数据复制了一份放到了用户本地的内存中，供用户在不连接数据库的情况下读取数据。DataAdapter 是一种用来充当 DataSet 对象与实际数据源之间桥梁的对象，专门为 DataSet 服务的。本节将详细介绍 DataSet 和 DataAdapter 对象的相关知识。

10.5.1 离线模式核心对象——DataSet 对象

DataSet 是 ADO.NET 最核心成员之一，它是支持 ADO.NET 断开式、分布式数据方案的核心对象，也是实现基于非连接的数据查询的核心组件。对于 DataSet 对象可以将其看做是在内存中创建的一个小型关系数据库，它将数据库中的数据复制了一份放到了用户本地的内存中，供用户在不连接数据库的情况下读取数据，充分利用了客户端资源，大大降低了数据库服务器的压力。

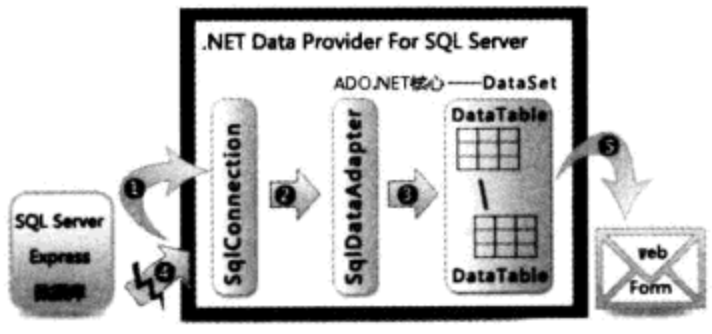


图 10.22 ADO.NET 数据库访问：离线模式（也称非连接模式）

就像前面我们将 DataSet 对象比喻成一个大水库，把抽上来的水按一定关系的池子进行存放，即使撤掉“抽水装置”（断开连接，离线状态），也可以保持“水”的存在。这也正是 ADO.NET 的核心。

如图 10.22 所示，当把 SQL Server 数据库的数据通过起“桥梁”作用的 SqlDataAdapter 对象（主要为该对象的 Fill 方法）填充到 DataSet 数据集中后，就可以对数据库进行一个断开连接、离线状态的操作，所以“标记④”这一步骤

就可以忽略不使用。

学习手册 注意　离线模式同连线模式一样也要建立数据库连接访问，不同的是：在离线模式下，一旦将数据整批取回便可切断连接，后续的数据读取操作可不再依赖数据库连接读取。

10.5.2　桥梁架设工程师——DataAdapter 对象

DataAdapter 对象（通常称为数据适配器）是一种用来充当 DataSet 对象与实际数据源之间桥梁的对象，可以说只要有 DataSet 的地方就有它，它也是专门为 DataSet 服务的。

DataAdapter 对象的工作步骤一般有两种：一种是通过 Command 对象执行 SQL 语句从数据源中检索数据，将获取的结果集填充到 DataSet 对象的表中；另一种是把用户对 DataSet 对象作出的更改写入到数据源中。

学习手册 说明　在.NET Framework 中主要使用两种 DataAdapter 对象，即 OleDbDataAdapter 和 SqlDataAdapter。OleDbDataAdapter 对象适用于 OLEDB 数据源，SqlDataAdapter 对象适用于 SQL Server 10.0 或更高版本。

（1）DataAdapter 对象常用属性如表 10.12 所示。

表 10.12　DataAdapter 对象常用属性及说明

属　性	说　明
SelectCommand	获取或设置用于在数据源中选择记录的命令
InsertCommand	获取或设置用于将新记录插入到数据源中的命令
UpdateCommand	获取或设置用于更新数据源中记录的命令
DeleteCommand	获取或设置用于从数据集中删除记录的命令

先前所讲解的 DataSet 对象是一个非连接的对象，它与数据源无关，也就是说该对象并不能直接跟数据源产生联系，而这里所介绍的 DataAdapter 则正好负责填充它并把它的数据提交给一个特定的数据源，它与 DataSet 配合使用才可以执行添加、修改和删除操作。

学习手册 说明　这里只需要将 DataAdapter 的 Command 属性设置成表 10.12 中给出的相应的 InsertCommand、UpdateCommand 和 DeleteCommand 即可。如果想填充一个 DataSet，则需要将这个属性设置成 SelectCommand。

【例 10.17】 对 DataAdapter 对象的 SelectCommand 属性赋值。

```
SqlConnection con=new SqlConnection(strCon);          //创建数据库连接对象
SqlDataAdapter ada = new SqlDataAdapter();            //创建 SqlDataAdapter 对象
//给 SqlDataAdapter 的 SelectCommand 赋值
 ada.SelectCommand=new SqlCommand("select * from authors",con);
……//省略后继代码
```

【例 10.18】可以使用上述方式给 InsertCommand、UpdateCommand 和 DeleteCommand 属性赋值，如以下示例代码对 DataAdapter 对象的 UpdateCommand 属性赋值执行更新操作：

```
    SqlConnection con=new SqlConnection(strCon);          //创建数据库连接对象
    SqlDataAdapter da = new SqlDataAdapter();             //创建 SqlDataAdapter 对象
    //给 SqlDataAdapter 的 UpdateCommand 属性指定执行更新操作的 SQL 语句
    da.UpdateCommand = new SqlCommand("update tb_mrbccd set brccdName = @brccdName
where id=@id", con);
    //添加参数并赋值
    da.UpdateCommand.Parameters.Add("@brccdName",    SqlDbType.NVarChar,    900,
"brccdName");
    SqlParameter prams_ID = da.UpdateCommand.Parameters.Add("@id", SqlDbType.Int);
    ……//省略后继代码
```

（2）DataAdapter 对象常用方法如表 10.13 所示。

表 10.13　DataAdapter 对象常用方法及说明

方　法	说　明
Fill	从数据源中提取数据以填充数据集
Update	更新数据源

根据表 10.13，当调用 Fill 方法时，它将向数据存储区传输一条 SQL SELECT 语句，该方法主要用来填充或刷新 DataSet，返回值是影响 DataSet 的行数。该方法的常用定义如下：

☑ int Fill(DataSet dataset)：添加或更新参数所指定的 DataSet，返回值是影响的行数。

☑ int Fill(DataTable datatable)：将数据填充到一个数据表中。

☑ int Fill(DataSet dataset，String tableName)：填充指定的 DataSet 中特定的表。

10.5.3　填充并访问 DataSet 表中数据

通过 DataAdapter 对象查询数据之后，需要把数据填充到 DataSet 中。

具体操作流程是首先使用 DataAdapter 取出数据，然后调用 DataAdapter 的 Fill 方法，将取得的数据导入 DataSet 中。

图 10.23　使用 SqlDataAdapter 对象以只读方式读取 DataSet 表中数据

实例位置：光盘\MR\Instance\10\10.12

【例 10.19】 使用 SqlDataAdapter、DataSet 对象访问 db_AspnetLearn 数据库中的 AddLinkMen 表，应用 foreach 循环语句把该表中的信息读取出来。实例运行结果如图 10.23 所示。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）当页面加载时，在 Default.aspx 页的 Page_Load 事件下应用一个 foreach 循环，声明 DataRow 类型的变量 mydr，将数据集 ds 里名称为 tb_user 表的 Rows 集合的内容逐一读取出来并把它们显示在前台页面中的 Label 控件中，实现代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    lblinfo.Text = "";
    SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings
```

```
        ["conStr"]. ToString());
        SqlDataAdapter ada = new SqlDataAdapter("select * from AddLinkMen", conn);
                                                    //创建 SqlDataAdapter 对象
        DataSet ds = new DataSet();                 //创建 DataSet 对象
        int counter = ada.Fill(ds, "tb_user");      //填充数据集
        Response.Write("获得: " + counter.ToString() + "条数据!" + "<br/>");
                                                            //输出信息
        foreach (DataRow mydr in ds.Tables["tb_user"].Rows)     //遍历数据行
        {
            lblinfo.Text += mydr["UserID"].ToString() + "-"+ mydr["userName"].
            ToString() + "--昵称: "+ mydr ["UserNickName"].ToString() + "<br/>";
        }
    }
```

10.5.4　修改 DataSet 并更新数据源（批量更新）

DataAdapter 可谓一把“双刃剑”，不但可以用来填充 DataSet，还可以调用其 Update() 方法将 DataSet 中的更改解析回数据库，从而可以更新数据库，包括插入、更新和删除。DataAdapter 对象的 Update()方法常用定义如下。

- ☑ Update(DataSet)：根据指定的数据集中的数据表更新数据源。
- ☑ Update(DataTable)：根据指定的数据表更新数据源。
- ☑ Update(dataRows)：根据指定的数据行数组更新数据源。

当调用 Update 方法时，DataAdapter 将分析已作出的更改并执行相应的命令（Insert、Update 或 Delete）。当 DataAdapter 遇到 DataRow 的更改时，它将使用 InsertCommand、UpdateCommand 和 DeleteCommand 来处理更改。

学习手册 说明　可以通过在设计时制定命令语法并在可能时通过使用存储过程来尽量提高 ADO.NET 应用程序的性能。在调用 Update 命令前，保证用于更改数据的 3 个 Command 存在，否则将引发异常。

实例位置：光盘\MR\Instance\10\10.13

【例 10.20】 实现使用数据适配器 DataAdapter 从数据库中读取“AddLinkMen”表中的内容并填充到 DataSet 数据集中，使用 CommandBuilder 对象及 Update 方法将数据集中的用户所在城市修改为“吉林省长春市”，然后将最终结果输出到页面中，最后将对数据集 DataSet 所做的更改保存到 SQL Server 2005 数据库中。实例运行结果如图 10.24 所示。

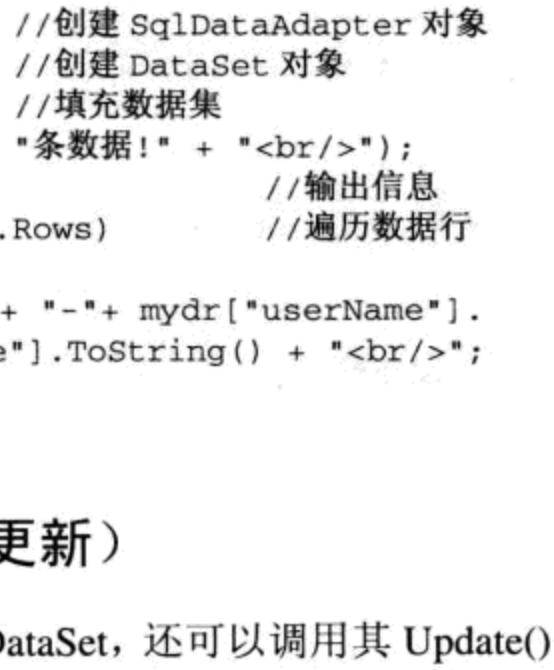


图 10.24　批量修改前后数据对比

学习手册 注意　在使用 CommandBuilder 时，数据库表必须定义主键，否则会出现异常信息。

程序开发的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在页面的 Paga_Load 事件中，应用 CommandBuilder 对象及 Update 方法修改 DataSet 并更新数据源中指定表中的数据，实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    //创建数据库连接对象
    SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings
    ["conStr"]. ToString());
    //创建数据适配器
    SqlDataAdapter dap = new SqlDataAdapter("select UserID,userName,UserCity
    from AddLinkMen", conn);
    DataSet ds = new DataSet();                              //创建数据集
    dap.Fill(ds, "user");                                    //填充数据集
    ShowDsTable(ds.Tables[0]);                               //显示未更新前的数据信息
    for (int i = 0; i <= ds.Tables["user"].Rows.Count - 1; i++)//更改数据操作
    {
        ds.Tables["user"].Rows[i]["UserCity"] = "吉林省长春市";
    }
    //使用 SqlCommandBuilder 对象，并和 SqlDataAdapter 关联，自动创建 UpdateCommand
    SqlCommandBuilder builder = new SqlCommandBuilder(dap);
    dap.Update(ds, "user");          //应用 SqlDataAdapter 和的 Update 方法更新数据
    ShowDsTable(ds.Tables[0]);       //调用自定义方法 ShowDsTable 显示更新后的数据
    dap.Dispose();
    ds.Dispose();
}
```

如果想使用最新的数据，那么必须重新调用 Fill()方法刷新 DataSet，更新的信息将并入现有行。Fill()方法通过检查 DataSet 中行的主键值及 SelectCommand 返回的行来确定是要添加一个新行还是更新现有行。

在上述事件代码中调用了一个自定义方法 ShowDsTable，用于将更改前的原始数据及更改后的数据分别显示在页面中，该方法的具体代码如下所示：

```
public void ShowDsTable(DataTable dataTable)
{
    Response.Write("<table border=1 align='center' ");          //输出表格
    Response.Write("<tr><th>编号</th><th>用户名</th><th>所在城市</th></tr>");
    Response.Write("<tr>");
    foreach (DataRow row in dataTable.Rows)                     //遍历数据行
    {
        for (int i = 0; i < dataTable.Columns.Count; i++)
        {
            Response.Write("<td align='center'>" + row[i] + "</td>");
                                                    //按照顺序以列名指定要读取的项
        }
        Response.Write("</tr>");
    }
    Response.Write("</table>");
}
```

10.6 实 战 练 习

10.6.1 使用参数化查询预防 SQL 注入式攻击

➤➤➤题目描述

在与数据库交互的 Web 应用程序中严重的风险之一就是 SQL 注入式攻击，该攻击是

指利用 SQL 语句设计上的漏洞，在目标服务器上运行 SQL 命令以及进行其他方式的攻击。比较有效的防范措施就是使用参数化查询预防 SQL 注入。运行效果如图 10.25 所示。

图 10.25　使用参数化查询防止 SQL 注入式攻击

➤➤➤技术指导

在讲解如何预防 SQL 注入式攻击之前，举个例子来看一下基本的 SQL 注入式攻击。假如在登录页面里添加一个文本框用来输入会员名，再添加一个按钮用来执行登录操作。在文本框中输入会员名“mr”，然后用 SQL 语句查找出数据库中符合条件的记录条数。实现的 SQL 语句代码如下所示：

```
select * from tb_LoginUser where UserName='mr'--构建普通的 SQL 语句
```

通过上面的语句可以在数据库中查询出一条 UserName 字段为 mr 的用户信息。如果在文本框中输入“'mr'or'1'='1”（简单的 SQL 注入式攻击），那么将会把其他与 mr 相关的用户也搜索出来（用户登录信息一般都为较敏感信息，这样暴露很危险）。简单的 SQL 攻击语句如下：

```
select * from tb_LoginUser where UserName='mr' or '1'='1'--简单的 SQL 注入式攻击
```

打开 SQL Server 2005 查询分析器，选择数据库名为 db_AspnetLearn，以上两条 SQL 语句在查询分析器中的运行结果如图 10.26 所示。

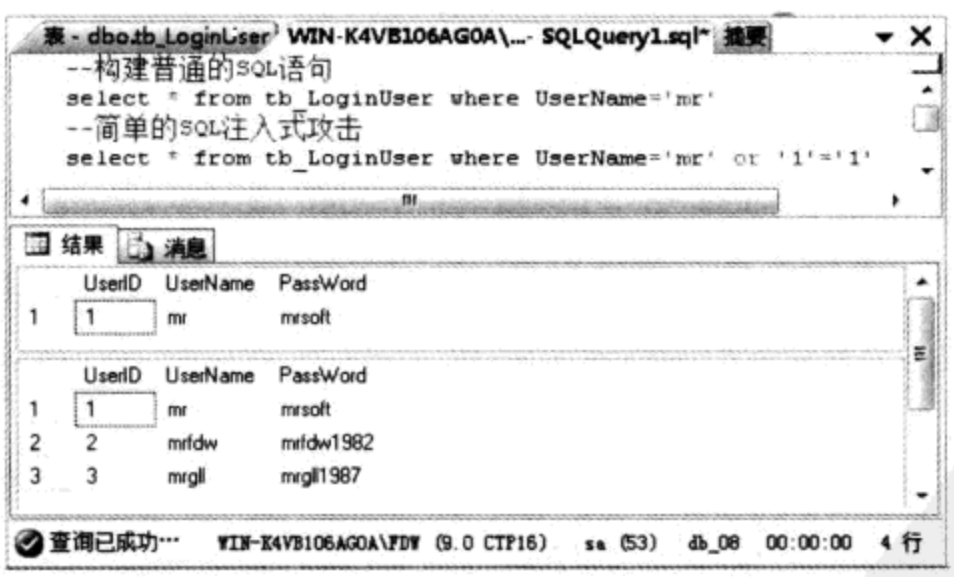


图 10.26　普通的 SQL 语句与注入式攻击的 SQL 对比

可以通过使用 SqlCommand.Parameters 属性的参数传值防止这样的 SQL 语句的注入式攻击（以操作 SQL Server 为例），将非法字符过滤掉，这样 or 语句就不起作用了。

【例 10.21】　下面是正常的 SQL 语句：

```
select * from tb_LoginUser where UserName='mr'
```

可以写改成：

```
select * from tb_LoginUser where UserName=@UserName
```

然后，使用 Command 对象的 Parameters 集合来进行值的传递，就可防止简单的 SQL 注入式攻击了。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/10.6.1**

10.6.2 应用存储过程有效预防 SQL 注入式攻击

➤➤➤题目描述

调用存储过程防范 SQL 注入式攻击（非常有效的方式），参考的运行结果如图 10.27 所示。

图 10.27 调用存储过程防止 SQL 注入

➤➤➤技术指导

使用参数化命令即 SQL 文本中使用占位符的命令，基本上可以有效地预防 SQL 注入式攻击，但是更有效的方法就是调用存储过程。

学习手册 说明 存储过程有着良好的逻辑封装体，不仅可以接收和返回数据，还可以有效过滤 SQL 语句中的非法字符。参数化命令是调用完整功能存储过程的诸多命令中的一小部分。

打开 SQL Server 2005 数据库管理器，选择名为 db_AspnetLearn 的数据库，创建一个名为 getLoginUser 的存储过程执行查询操作，代码如下所示。

```
CREATE PROCEDURE getLoginUser
@UserName NvarChar(50),
@PassWord NvarChar(50)
AS
BEGIN
SELECT COUNT(*) FROM tb_LoginUser WHERE UserName=@UserName AND PassWord=
@PassWord
END
```

GO

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ ASP.NET/loveASP.NET/10.6.2**

10.7　本 章 小 结

本章围绕使用 ADO.NET 进行数据访问展开，介绍了在 Vistal Studio .NET 中创建数据库连接和执行数据库操作，并且介绍了如何能够将 ASP.NET 控件和 ASP.NET 的强大功能结合起来提供当今网站所需要的数据库功能，通过本章的学习，希望读者对 ADO.NET 能有更进一步的掌握。

第 11 章

ASP.NET 服务器控件

（名师课堂：54 分钟）

服务器控件在 ASP.NET 框架中起着举足轻重的作用，是构建 Web 应用程序最关键、最重要的组成元素。对于一个优秀的开发人员，掌握服务器控件的基础知识是非常重要的。本章将介绍服务器端标准控件、服务器端验证控件及导航控件的应用。通过本章的学习，读者可以掌握以下知识：

- 掌握文本类控件的使用
- 掌握按钮类控件的使用
- 掌握列表类控件的使用
- 掌握图形显示控件的使用
- 掌握文件上传控件的使用

11.1　服务器端控件概述

在 ASP.NET 中，服务器控件是指在服务器上执行程序逻辑的组件。这个组件可能生成一定的用户界面，也可能不包括用户界面。每个服务器控件都包含一些成员对象，以便于开发人员调用，如属性、事件、方法等。通常情况下，服务器控件都包含在 ASP.NET 页面中。当运行页面时，.NET 执行引擎将根据控件成员对象和程序逻辑定义完成一定的功能。例如在客户端呈现用户界面，这时用户可与控件发生交互行为，当页面被用户提交时，控件可在服务器端引发事件，并由服务器端根据相关事件处理程序来进行事件处理。服务器控件是 WebForm 编程模型的重要元素，它们构成了一个新的基于控件的表单程序的基础。通过这种方式可以简化 Web 应用程序的开发，提高应用程序的开发效率。向页面中添加控件的方法很简单，通常情况下只需将控件从工具箱中拖曳到页面中即可，如图 11.1 所示。

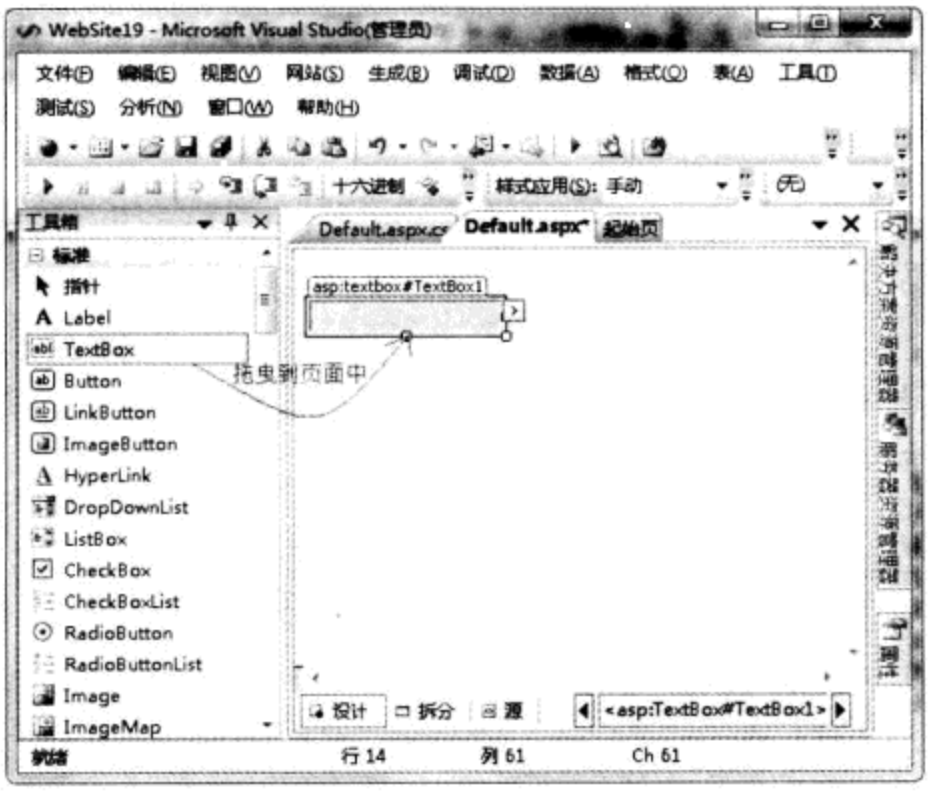


图 11.1　添加服务器控件

学习手册 说明　删除控件时，只需选中页面中的控件，按下键盘中的<Delete>键即可，或者在控件上单击鼠标右键，在右键菜单中选择“删除”选项。

11.2　文本类型控件

专题讲座：光盘\MR\Video\11\文本类型控件.exe

➤➤➤视频速递：通过本视频可以对 Label 控件和 TextBox 控件有更深的了解。

文本类型控件主要包括标签控件 Label 和文本框控件 TextBox，都是用来接收文本信

息的。本节将详细介绍如何添加文本类型控件及其用法。

11.2.1 使用 Label 控件显示文本

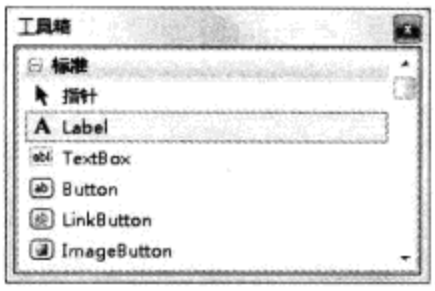


图 11.2 Label 控件

Label 控件主要用来在 Web 页面上显示静态文本，使用 Label 控件的好处是，可以在运行时使用代码改变它的显示文本或背景色等属性。如果只想显示静态文本而不想在运行时改变它，则可以用 HTML 进行显示，即直接在 aspx 文件中输入显示的内容。Label 控件如图 11.2 所示。

Label 控件的常用属性及说明如表 11.1 所示。

表 11.1 Label 控件的常用属性及说明

属　性	说　明
ID	Label 控件的 ID 名称
Text	Label 控件显示的文本
Width	Label 控件的宽度设置
BackColor	控件的背景颜色
BorderColor	控件的边框颜色
BorderWidth	控件的边框宽度

下面对比较重要的属性进行详细介绍。

（1）ID 属性

ID 属性用来标识 Label 控件，程序开发人员在编程过程中可以通过控件的 ID 来调用该控件，设置其属性、方法和事件。

语法：

```
public virtual string ID { get; set; }
```

【例 11.1】 设置 Label 控件的 ID 为 lblname，通常情况下，在 Label 属性对话框中设置即可，如图 11.3 所示。

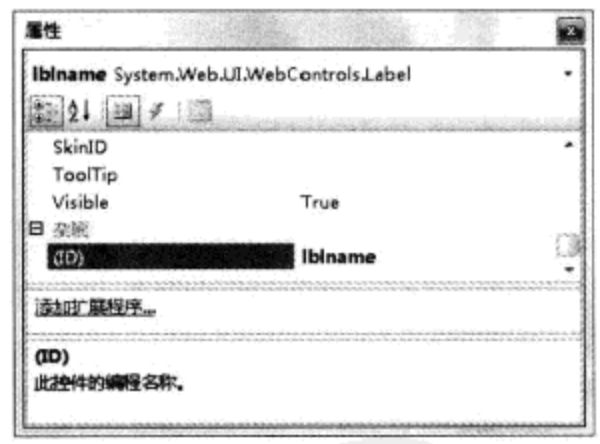


图 11.3 设置 Label 控件的 ID 属性

学习手册 说明 除了在属性窗口中设置 ID 属性，还可以在控件的 HTML 代码中设置 ID 属性。

（2）Text 属性

Text 属性用来设置 Label 控件所显示的文本内容。

语法：

```
public string Text { get; set; }
```

【例 11.2】 根据程序开发人员的需要可以通过 Label 属性对话框设置，也可以通过代码设置。在此通过代码来设置 Text 属性为“红尘倒影”，代码如下所示：

```
Label1.Text = "红尘倒影";
```

通过属性对话框进行属性值的设置，直接打开控件的属性对话框设置即可。

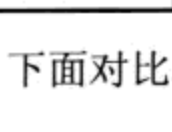
实例位置：光盘\MR\Instance\11\11.1

【例 11.3】 通过设置 Label 控件的 Text 属性，显示静态的文本信息，如显示“红尘百劫苦、虚幻总是甜”字样，实例运行结果如图 11.4 所示。

图 11.4　使用 Label 控件显示文本信息

程序代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    lblinfo.BackColor = System.Drawing.Color.Azure;          //设置背景颜色
    lblinfo.BorderWidth = 1;                                 //设置边框宽度
    lblinfo.BorderColor = System.Drawing.Color.Black;        //设置边框颜色
    lblinfo.Text = "红尘百劫苦、虚幻总是甜";                  //设置显示的文字
}
```



Label 控件也有一些常用的方法和事件，但是这些方法和事件在开发过程中基本上用不到，只是一些公用的方法和事件而已，最常用的是 Label 控件的相关属性，所以作者没有列出 Label 控件的方法和事件，读者只了解 Label 控件的常用属性即可。

11.2.2　使用 TextBox 控件输入数据

TextBox 控件又称文本框控件，主要作用是为用户提供输入文本的区域，在程序开发中是比较常用的服务器控件，应用程序利用 TextBox 控件使用户能够输入字符串或者密码，TextBox 控件如图 11.5 所示。

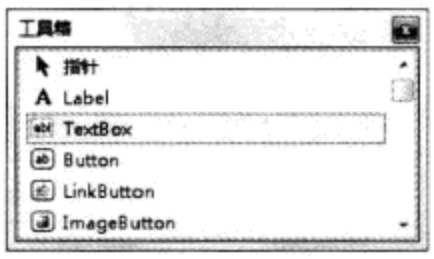


图 11.5　TextBox 控件

1．TextBox 控件属性

TextBox 控件的常用属性及说明如表 11.2 所示。

表 11.2　TextBox 控件的常用属性及说明

属　性	说　明
AutoPostBack	在文本修改之后，是否自动回发到服务器
ID	控件 ID
Text	控件要显示的文本
TextMode	设置 TextBox 控件的行为模式（单行、多行或密码）
Width	控件的宽度
BackColor	控件的背景颜色
Rows	设置多行文本框中显示的行数
Columns	设置多行文本框中显示的宽度

下面对比较重要的属性进行详细介绍。

（1）TextMode 属性

TextMode 属性主要用于控制 TextBox 控件的文本显示方式。

语法：

```
[ThemeableAttribute(false)]
public virtual TextBoxMode TextMode { get; set; }
```

该属性的设置选项有以下 3 种。

☑ 单行（SingleLine）：用户只能在一行中输入信息，还可以通过设置 TextBox 的 Columns 属性值限制文本的宽度；通过设置 MaxLength 属性值限制输入的最大字符数。

☑ 多行（MultiLine）：文本很长时，允许用户输入多行文本并执行换行，还可以通过设置 TextBox 的 Rows 属性值限制文本框显示的行数。

☑ 密码（Password）：将用户输入的字符用黑点（•）屏蔽，以隐藏这些信息。

【例 11.4】 在验证用户登录密码时，可以将 TextBox 控件的 TextMode 属性设置为 Password。在填写备注时，文字可能会很多，此时可将 TextBox 控件的 TextMode 属性设置 MultiLine。效果如图 11.6 和图 11.7 所示。

（2）AutoPostBack 属性

用于设置在文本修改之后，是否自动回发到服务器。如果回发，则设置为 true，否则为 false。可以在 TextBox 控件的属性窗口中进行设置，如图 11.8 所示。

图 11.6 隐藏登录密码

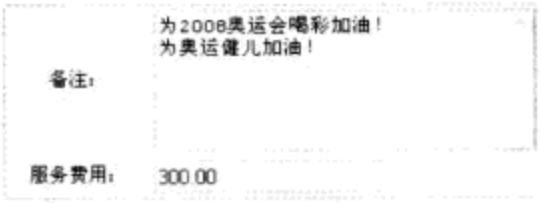


图 11.7 填写备注信息

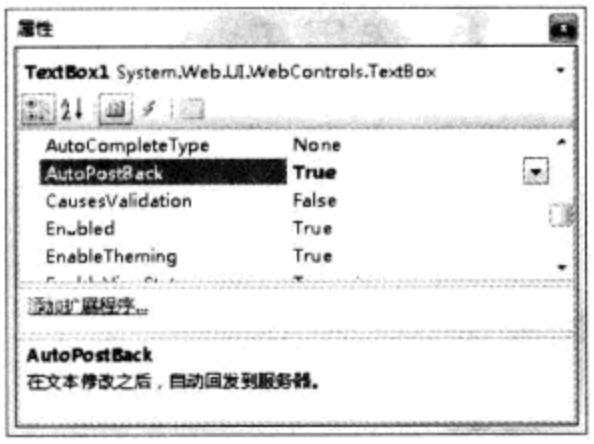


图 11.8 设置 AutoPostBack 属性

学习手册 说明 将 TextBox 控件的 AutoPostBack 属性设为 true 之后，当在控件中输入数据且控件失去焦点以后，控件会自动回发到服务器。

图 11.9 使用 TextBox 控件制作用户注册界面

实例位置：光盘\MR\Instance\11\11.2

【例 11.5】 通过设置 TextBox 控件的 TextMode 属性值，制作会员注册界面。执行程序，并在 TextBox 文本框中输入文字，实例运行结果如图 11.9 所示。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 4 个 TextBox 控件、一个 RadioButtonList 控件和一个 Button 控件，

它们的属性设置如表 11.3 所示。

表 11.3　Default.aspx 页面中控件的属性设置

控　件	说　明
abl TextBox	ID 属性设为“txtUser”，用于输入注册用户的姓名
	ID 属性设为“txtPassword”，TextMode 属性设为“Password”，用于输入密码
	ID 属性设为“txttel”，用于输入注册用户的移动电话号码
	ID 属性设为“txtinfo”，TextMode 属性设为“MultiLine”，用于输入用户简介
RadioButtonList	ID 属性设为“rblsex”，用于选择性别
ab Button	用于提交注册用户信息

学习手册 说明　由于本例主要是介绍控件的属性设置，所以程序中并没有编写代码，读者根据本例能够了解 TextBox 控件的几个主要属性设置。

2．TextBox 控件方法

TextBox 控件的常用方法及说明如表 11.4 所示。

表 11.4　TextBox 控件的常用方法及说明

方　法	说　明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
Focus	为控件设置输入焦点
GetType	获取当前实例的类型

TextBox 控件主要方法只有一个 Focus 方法，Focus 方法是为控件设置输入焦点。

语法：

```
public virtual void Focus ()
```

【例 11.6】 使 ID 为 txtTest 的 TextBox 控件获得焦点。代码如下所示：

```
This.txtTest.Focus ();
```

3．TextBox 控件事件

TextBox 控件的常用事件及说明如表 11.5 所示。

表 11.5　TextBox 控件的常用事件及说明

事　件	说　明
DataBinding	当服务器控件绑定到数据源时发生
Disposed	通过该事件将服务器控件从内存中释放出来
TextChanged	当用户更改 TextBox 的文本时发生
Unload	当服务器控件从内存中卸载时发生

TextBox 控件主要事件是 TextChanged 事件，当用户更改 TextBox 控件的文本时将触发 TextChanged 事件。

语法：

```
public event EventHandler TextChanged
```

在应用此事件之前要将 TextBox 控件的 AutoPostBack 属性设为 True。

实例位置：光盘\MR\Instance\11\11.3

【例 11.7】 页面上放置两个 TextBox 控件，在第一个 TextBox 控件的 TextChanged 事件中编写代码，判断数字是奇数还是偶数，并将判断结果显示到第二个 TextBox 控件中，实例运行结果如图 11.10 所示。

图 11.10 TextChanged 事件中判断数字奇偶性

程序代码如下所示：

```
protected void txtNum2_TextChanged(object sender, EventArgs e)
{
    if (txtNum2.Text.Length == 0)                          //判断是否输入第一个数字
    {
        //如果没有输入数字，则弹出提示信息
        Page.RegisterClientScriptBlock("", "<script>alert('请输入一个数');
        </script>");
        txtNum2.Focus();                                   //将光标定位到文本框中
    }
    else                                                   //如果输入数字
    {
        int result;
        bool n1 = Information.IsNumeric(txtNum2.Text.Trim());
                                                           //判断第一个数据是否是数字
        if (n1)
        {
            result=int.Parse(txtNum2.Text.Trim())%2;       //判断奇偶性
            if (result == 0)                               //如果结果为 0
            {
                txtSum.Text = "偶数";                      //说明是偶数
            }
            els    e                                       //如果不为 0
            {
                txtSum.Text = "奇数";                      //说明是奇数
            }
        }
    }
}
```

判断文本框中是否输入数据，可以通过判断 Text 属性是否为空，也可以通过判断文本框中数据的长度是否为 0 来判断是否输入数据。

在对 TextChanged 事件编程时，应先将该控件的 AutoPostBack 属性设为 True。AutoPostBack 属性用于控制 TextBox 控件的事件是否自动提交服务器，系统默认设置为 False。当属性设置为 True 时，若事件被触发则事件自动被提交到服务器，否则事件只有在下一次页面提交服务器时才被触发。

11.2.3　情景应用：简单的加法运算器

了解 TextBox 控件后，下面通过一个简单的实例演示如何使用 TextBox 控件和运算符开发简单的运算器程序。

专题讲座：光盘\MR\Video\11\简单的加法运算器.exe

➤➤➤视频速递：理解如何在实际中应用运算符和 TextBox 控件。

实例位置：光盘\MR\Instance\11\11.4

【例 11.8】 读者对运算器并不会感到陌生，通过运算器可以计算很多数学公式，如加法运算、减法运算和除法运算等。主要是通过不同的运算符执行相应的运算，本实例通过加法运算符“+”开发一个简单的加法运算器，效果如图 11.11 所示。

图 11.11　加法运算器

（1）新建一个网站，默认主页为 Default.aspx。

（2）向窗体中添加 3 个 TextBox 控件，分别用于输入两个数字和显示相加后的结果。

（3）将第二个 TextBox 控件 txtNum2 的 AutoPostBack 属性设为 true，使控件失去焦点时自动回发到服务器，触发 TextChanged 事件。

（4）在 txtNum2 控件的 TextChanged 事件中编写代码，计算两个数字的和，主要代码如下所示：

```
protected void txtNum2_TextChanged(object sender, EventArgs e)
{
    if (txtNum1.Text.Length == 0)                          //判断是否输入第一个数字
    {
        //如果没有输入数字，则弹出提示信息
        Page.RegisterClientScriptBlock("", "<script>alert('请输入第一个数');
        </script>");
        txtNum1.Focus();                                   //将光标定位到文本框中
    }
    else                                                   //如果输入数字
    {
        if (txtNum2.Text.Length == 0)                      //判断是否输入第二个数字
        {
            //如果没有输入，则弹出提示信息
            Page.RegisterClientScriptBlock("", "<script>alert('请输入第二个数');
            </script>");
            txtNum2.Focus();                               //将光标定位到文本框中
        }
        else                                               //如果输入第二个数字
        {
            bool n1= Information.IsNumeric(txtNum1.Text.Trim());
                                                           //判断第一个数据是否是数字
            bool n2 = Information.IsNumeric(txtNum2.Text.Trim());
                                                           //判断第二个数据是否是数字
            if (n1 == true && n2 == true)                  //如果两个输入的都是数字
            {
                //计算两个数字的和并显示出来
                txtSum.Text = (int.Parse(txtNum1.Text.Trim()) + int.Parse
                (txtNum2.Text.Trim())). ToString();
            }
```

```
            else                                          //如果输入的数据有的不是数字
            {
                //弹出提示信息
                Page.RegisterClientScriptBlock("", "<script>alert('请输入数字');
                </script>");
            }
        }
    }
}
```

11.3 按钮类型控件

专题讲座：光盘\MR\Video\11\按钮类型控件.exe

➤➤➤视频速递：通过本视频可以对 Button 控件和 ImageButton 控件有更深的了解。

本节主要讲解 Button 控件（分为提交按钮控件和命令按钮控件）和 ImageButton 控件。按钮类型控件在开发中的应用是非常广泛的，基本上每个页面都会有按钮控件的影子，所以读者要认真学习本节的内容。

11.3.1 通过 Button 控件提交表单

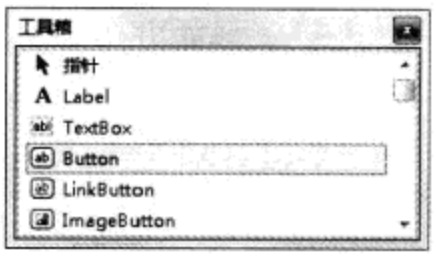


图 11.12 Button 控件

Button 控件可以分为提交按钮控件和命令按钮控件。提交按钮控件只是将 Web 页面回送到服务器，在默认情况下，Button 控件为提交按钮控件；而命令按钮控件一般包含与控件相关联的命令，用于处理控件命令事件，Button 控件如图 11.12 所示。

1. Button 控件属性

Button 控件常用属性及说明如表 11.6 所示。

表 11.6 Button 控件常用属性及说明

属　性	说　明
ID	设置分配给服务器控件的编程标识符
Text	设置在 Button 控件中显示的文本标题
AccessKey	该控件使用的键盘快捷键
CommandArgument	设置可选参数，该参数与关联的 CommandName 一起被传递到 Command 事件
CommandName	设置命令名，该命令名与传递给 Command 事件的 Button 控件相关联
PostBackUrl	设置单击 Button 控件时从当前页发送到的网页的 URL

下面对比较重要的属性进行详细介绍。

（1）Text 属性

Button 控件的 Text 属性用于设置在按钮上显示的文本。

语法：

```
public string Text { get; set; }
```

属性值：在 Button 控件中显示的文本内容，默认值为空。

【例 11.9】 将 ID 属性为 btnButton 的按钮控件的显示文本设置为“我是个按钮，你知道吗？”，运行结果如图 11.13 所示。

代码如下所示：

```
btnClick.Text=" 我是个按钮，你知道吗？";
```

（2）AccessKey 属性

使用 AccessKey 属性可以为 Button 控件指定键盘快捷键。设置该属性后通过按键盘上的 Alt 键和指定的字符键，即可快速触发该控件的 Click 事件。

语法：

```
public virtual string AccessKey { get; set; }
```

属性值：用于快速定位到 Web 服务器控件的访问键。默认值为空，表示未设置此属性。

【例 11.10】 如果将控件的访问键设置为字符串“A”，表示用户可以通过"Alt+A"组合键触发控件的 Click 事件，可以在属性窗口中设置该属性，如图 11.14 所示。

图 11.13　Button 控件的 Text 属性

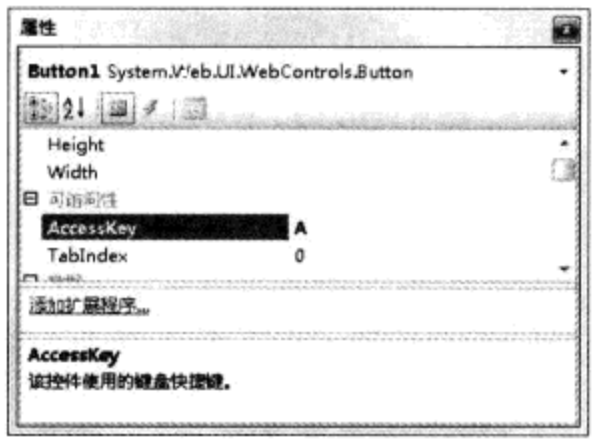


图 11.14　设置 AccessKey 属性

AccessKey 属性只允许设置为单个字符串。如果此属性设置既不是空引用，也不是单个字符串的值，将会发生异常。

2. Button 控件方法

Button 控件的常用方法及说明如表 11.7 所示。

表 11.7　Button 控件的常用方法及说明

方　法	说　明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
Focus	为控件设置输入焦点
ResolveUrl	将 URL 转换为在请求客户端可用的 URL

下面主要介绍 Button 控件的 Focus 方法，该方法为控件设置焦点，这样，当控件获得焦点后，直接单击回车键就可以触发 Button 控件的 Click 事件。

语法：

```
public virtual void Focus ()
```

【例 11.11】 使 Button 控件获得焦点。

```
protected void Page_Load(object sender, EventArgs e)
{
    Button1.Focus();
}
```

Button 控件获得焦点以后，按键盘上的回车键就会直接执行按钮的 Click 事件中的代码。

3. Button 控件事件

Button 控件的常用事件及说明如表 11.8 所示。

表 11.8　Button 控件的常用事件及说明

事　件	说　明
Command	在单击 Button 控件并定义关联的命令时激发
Click	在单击 Button 控件时引发的事件
PreRender	在加载 Control 对象之后、呈现之前发生
Unload	当服务器控件从内存中卸载时发生

用户如果要在 Button 控件的某个事件下实现某种功能，可以在属性对话框中单击事件按钮并找到相应事件，然后双击进入编写代码。此处，主要介绍 Button 按钮的 Click 事件，此事件为控件的单击事件，当单击控件时可触发控件的 Click 事件中的代码程序。

语法：

```
public event EventHandler Click
```

【例 11.12】用户要在单击按钮时弹出一个消息提示框，可直接在 Button 控件的 Click 事件下编写如下代码：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("<script>alert('弹出提示框！')
    </script>");
}
```

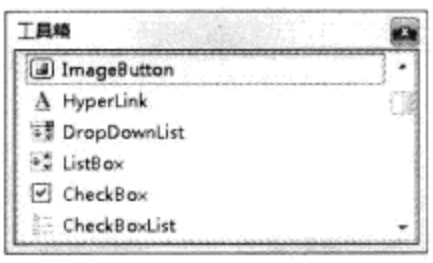


图 11.15　ImageButton 控件

11.3.2　显示图像的 ImageButton 控件

ImageButton 控件为图像按钮控件，它用于显示具体的图像，在功能上和 Button 控件相同，ImageButton 控件如图 11.15 所示。

ImageButton 控件的常用属性及说明如表 11.9 所示。

表 11.9　ImageButton 控件的常用属性及说明

属　性	说　明
ID	控件 ID
Text	设置在 ImageButton 控件中显示的文本标题
ImageUrl	在 ImageButton 控件中显示的图像的位置
PostBackUrl	单击 ImageButton 控件时所发送到的 URL

AlternateText	当图像不可用时，ImageButton 控件中显示的替换文本

下面对比较重要的属性进行详细介绍。

（1）ImageUrl 属性

用于设置在 ImageButton 控件中显示的图像的位置。

语法：

```
public virtual string ImageUrl { get; set; }
```

属性值：在 Image 控件中显示的图像的位置。

可以使用相对 URL 或绝对 URL。相对 URL 使图像的位置与网页的位置相关联，使得将整个站点移动到服务器上的其他目录变得更加容易。绝对 URL 提供完整路径，因此将站点移到其他目录时需要更新代码。

（2）AlternateText 属性

在 ImageUrl 属性指定的图像不可用时显示的文本。

语法：

```
public virtual string AlternateText { get; set; }
```

属性值：当图像不可用时，ImageButton 控件中显示的替换文本。

实例位置：光盘\MR\Instance\11\11.5

【例 11.13】 在页面中添加两个 ImageButton 控件，第一个只设置 AlternateText 属性，第二个只设置 ImageUrl 属性，运行结果如图 11.16 所示。

图 11.16　设置 ImageButton 控件的不同属性

程序代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    ImageButton1.AlternateText = "暂时没有图像";      //设置没有图像时显示的提示文字
    ImageButton2.ImageUrl = "login.gif";              //设置 ImageUrl 属性
}
```

由于 ImageButton 控件的事件与 Button 控件相同，所以，此处就不过多地介绍 ImageButton 控件的 Click 事件，读者可以参照 Button 控件的 Click 事件。

11.4　列表类型控件

专题讲座：光盘\MR\Video\11\列表类型控件.exe

➤➤➤视频速递：通过本视频可以对 ListBox 控件和 DropDownList 等控件有更深的了解。

本节主要讲解 ListBox 控件用于显示一组列表项和 DropDownList 控件从列表中选择一项，而且只在框中显示选定项。本节对这两个控件的用法会做详细的介绍。

11.4.1 呈现列表的 ListBox 控件

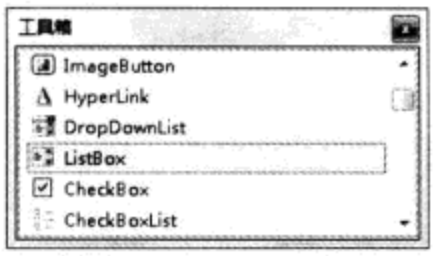


图 11.17 ListBox 控件

ListBox 控件用于显示一组列表项，用户可以从中选择一项或多项。如果列表项的总数超出可以显示的项数，则 ListBox 控件会自动添加滚动条。ListBox 控件如图 11.17 所示。

1. ListBox 控件属性

ListBox 控件的常用属性及说明如表 11.10 所示。

表 11.10 ListBox 控件的常用属性及说明

属　性	说　明
Items	获取列表控件项的集合
SelectionMode	设置 ListBox 控件的选择格式
SelectedIndex	设置列表中选定项的最低序号索引
SelectedItem	获取列表中索引最小的选中项
SelectedValue	获取列表控件中选定项的值，或选择列表控件中包含指定值的项
Rows	设置 ListBox 控件中显示的行数
Text	设置 ListBox 控件的 SelectedValue 属性

下面对比较重要的属性进行详细介绍。

（1）Items 属性

用于获取列表控件项的集合。

语法：

```
public virtual ListItemCollection Items { get; }
```

属性值：表示列表对象中的项的集合。

可以通过 ListBox 控件的 Items 属性添加、删除列表中的项。

【例 11.14】 向 ID 为 lbTest 的 ListBox 控件中添加项，代码如下所示：

```
lbTest.Items.Add("苹果");
lbTest.Items.Add("香蕉");
lbTest.Items.Add("西瓜");
lbTest.Items.Add("水蜜桃");
```

【例 11.15】 从 ID 为 lbTest 的 ListBox 控件中移除项，代码如下所示：

```
this.lbTest.Items.Remove("苹果");                    //从列表中移除指定项
this.lbTest.Items.RemoveAt(0);                       //从列表中移除索引为 0 的项
```

（2）SelectionMode 属性

用于设置 ListBox 控件的选择格式。

语法：

```
public virtual ListSelectionMode SelectionMode { get; set; }
```

属性值：将该属性设置为 ListSelectionMode.Single 表示只能从 ListBox 控件中选择一项，设置为 ListSelectionMode.Multiple 表示可选择多项。

【例 11.16】将 ID 为 lbTest 的 ListBox 控件的选择模式设为选择多项。代码如下所示：

```
this.lbTest.SelectionMode = ListSelectionMode.Multiple;
```

2. ListBox 控件方法

ListBox 控件最常用方法及说明如表 11.11 所示。

表 11.11　ListBox 控件最常用方法及说明

方　法	说　明
ClearSelection	清除列表选择并将所有项的 Selected 属性设置为 False
Equals	确定两个 Object 实例是否相等
Focus	为控件设置输入焦点
FindControl	在当前的命名容器中搜索指定的服务器控件
GetSelectedIndices	获取当前在 ListBox 控件中选定的项的索引值数组

下面对比较重要的方法进行详细介绍。

（1）ClearSelection 方法

清除列表选择并将所有项的 Selected 属性设置为 False。

语法：

```
public virtual void ClearSelection ()
```

使用此方法重置控件，以便不选定任何项。

（2）FindControl 方法

在当前的命名容器中搜索指定 ID 参数的服务器控件。

语法：

```
public virtual Control FindControl (string id)
```

参数 ID：要查找的控件的标识符。

返回值：指定的控件，或为空引用。

（3）GetSelectedIndices 方法

用 GetSelectedIndices 方法标识或访问 ListBox 控件中的选定项。返回数组中的每个元素表示一个选定列表项的索引。可以使用这些索引值访问 Items 集合中的项。

语法：

```
public virtual int[] GetSelectedIndices ()
```

返回值：一个整数数组，其中每个整数表示列表框中一个选定项的索引。

实例位置：光盘\MR\Instance\11\11.6

【例 11.17】对 ListBox 控件中的列表项进行多选和单选操作，选择 ListBox 控件中的一项或者多项后，单击按钮，可以将选择的内容添加到另一个 ListBox 控件中，实例运行结果如图 11.18 所示。

图 11.18　ListBox 控件基本应用

自定义一个 AddUser 方法，通过 ListBox 控件 Items

集合的 Add 方法向显示公司人员名单的 ListBox 控件中添加人员名称，代码如下所示：

```
private void AddUser()
{
    //通过ListBox控件Items集合的Add方法向显示公司人员名单的ListBox控件中添加人员名称
    lblist.Items.Add("红尘倒影");
    lblist.Items.Add("洪杰老师");
    lblist.Items.Add("小安老师");
    lblist.Items.Add("小科老师");
    lblist.Items.Add("小梅老师");
    lblist.Items.Add("玲玲老师");
    lblist.Items.Add("小刘老师");
    lblist.Items.Add("彬彬老师");
}
```

向 ListBox 控件中添加数据

向 ListBox 控件中添加项目，可以使用上述代码进行添加，也可以通过 ListBox 控件的属性窗口进行添加。

单击“添加”按钮，可以将选择的人员名添加到值班人员名单中。程序首先遍历所有人员名单，如果某一项被选择了，则将该项添加到值班人员名单中，使用的方法是 ListBox 控件 Items 集合的 Add 方法，代码如下所示：

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    for (int i = 0; i < lblist.Items.Count; i++)              //遍历控件所有项
    {
        if (lblist.Items[i].Selected)                         //如果项目被选中
        {
            ListItem li = lbStat.Items.FindByText(lblist.Items[i].Text);
                                                              //判断该项是否已被添加过
            if (li == null)                                   //如果没有被添加过
            {
                lbStat.Items.Add(lblist.Items[i].Text);       //将项目添加到值班名单中
            }
        }
    }
    lblist.ClearSelection();                                  //清除项目的选中状态
}
```

此处通过 FindByText 方法判断选择的员工名称是否已经添加过，如果已经添加过，则返回 ListItem 对象，否则对象为空。

如果想删除值班人员名单中的员工，可以选择员工名称之后，单击“移除”按钮。首先还是先遍历所有的值班人员名单，然后通过 Items 集合的 RemoveAt 移除选中的项目，代码如下所示：

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
    for (int i = 0; i < lbStat.Items.Count; i++)              //遍历控件所有项
    {
        if (lbStat.Items[i].Selected)                         //判断项目是否被选中
        {
            lbStat.Items.RemoveAt(i);                         //如果选中则删除该项
```

```
        }
    }
    lblist.ClearSelection();                    //清除项目的选中状态
}
```

3. ListBox 控件事件

ListBox 控件的常用事件及说明如表 11.12 所示。

表 11.12　ListBox 控件的常用事件及说明

事　件	说　明
DataBinding	当服务器控件绑定到数据源时引发的事件
DataBound	在服务器控件绑定到数据源后发生
SelectedIndexChanged	当列表控件的选定项在信息发往服务器之间变化时发生
TextChanged	当 Text 和 SelectedValue 属性更改时发生
Unload	当服务器控件从内存中卸载时发生

下面对比较重要的事件进行详细介绍。

（1）SelectedIndexChanged 事件

当列表控件中的选择项发生改变时，会引发 SelectedIndexChanged 事件。

语法：

```
public event EventHandler SelectedIndexChanged
```

【例 11.18】 为 ListBox 控件添加各项内容。

```
protected void Page_Load(object sender, EventArgs e)
{
    //向控件中添加项目
    ListBox1.Items.Add("春天");
    ListBox1.Items.Add("夏天");
    ListBox1.Items.Add("秋天");
    ListBox1.Items.Add("冬天");
}
```

在 SelectedIndexChanged 事件中编写代码，实现当选择项改变时弹出选择项的文本。

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    Response.Write(ListBox1.SelectedValue.ToString());//获取选择项的值
}
```

（2）TextChanged 事件

当用户对 ListBox 控件的 Text 和 SelectedValue 属性进行更改时引发 TextChanged 事件。

语法：

```
public event EventHandler TextChanged
```

【例 11.19】 当文本属性发生更改后，将输出提示“文本属性已更改”。

```
protected void ListBox1_TextChanged(object sender, EventArgs e)
{
```

```
        Response.Write("文本属性已更改");                    //文本属性更改后弹出提示
    }
```

学习手册 说明 在运行此程序之前应先将该控件的 AutoPostBack 属性设置为 True，表示只要用户更改列表中的选定内容就自动产生向服务器的回发。

11.4.2 实现下拉框的 DropDownList 控件

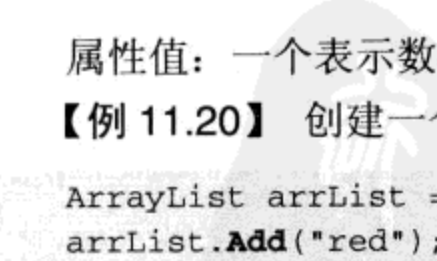


图 11.19 DropDownList 控件

DropDownList 控件与 ListBox 控件的使用类似，但 DropDownList 控件只允许用户每次从列表中选择一项，而且只在框中显示选定项，DropDownList 控件如图 11.19 所示。

1. DropDownList 控件属性

DropDownList 控件的常用属性及说明如表 11.13 所示。

表 11.13 DropDownList 控件的常用属性及说明

属　性	说　明
DataSource	设置对象，数据绑定控件从该对象中检索其数据项列表
DataTextField	设置为列表项提供文本内容的数据源字段
DataValueField	设置为各列表项提供值的数据源字段
ID	设置分配给服务器控件的编程标识符
Items	获取列表控件项的集合
SelectedItem	获取列表控件中索引最小的选定项
SelectedValue	获取列表控件中选定项的值，或选择列表控件中包含指定值的项

下面对比较重要的属性进行详细介绍。

（1）DataSource 属性

通过使用 DataSource 属性可以从数组或集合中获取列表项，并将其添加到控件中。当编程人员希望从数组或集合中填充列表时，可以使用此属性。

语法：

```
public virtual Object DataSource { get; set; }
```

属性值：一个表示数据源的对象，数据绑定控件从该对象中检索其数据。

【例 11.20】 创建一个数组 arrList。

```
ArrayList arrList = new ArrayList();                    //创建 ArrayList 集合数组
arrList.Add("red");                                     //添加内容
arrList.Add("blue");
arrList.Add("green");
arrList.Add("yellow");
```

使用 DataSource 属性将该数组附加到要填充的 DropDownList 控件中，代码如下所示：

```
this.ddlTest.DataSource = arrList;                      //绑定数据源
```

（2）DataTextField 属性

设置为列表项提供文本内容的数据源字段。

语法：

```
public virtual string DataTextField { get; set; }
```

属性值：指定为列表项提供文本内容的数据源字段，默认为空。

【例 11.21】 请读者参照下面的代码理解此属性的含义：

```
DropDownList1.DataTextField = "UserName";
```

学习手册 说明 UserName 是数据库中的字段名，如果 DropDownList 控件中显示数据库中某个字段的数据，则将控件的 DataTextField 属性设为相应的字段名。

2. DropDownList 控件方法

DropDownList 控件的最常用方法及说明如表 11.14 所示。

表 11.14　DropDownList 控件的最常用方法及说明

方　法	说　明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
FindControl	在当前的命名容器中搜索指定的服务器控件
ClearSelection	清除列表选择并将所有项的 Selected 属性设置为 False
OnSelectedIndexChanged	引发 SelectedIndexChanged 事件

下面对比较重要的方法进行详细介绍。

（1）DataBind 方法

DataBind 方法用于在 DropDownList 控件使用 DataSource 属性附加了数据源后，将数据源绑定到被调用的 DropDownList 控件。

语法：

```
public override void DataBind ()
```

【例 11.22】 将例 11.20 中的 arrList 数组绑定到 DropDownList 控件。

```
ArrayList arrList = new ArrayList();                    //实例化
arrList.Add("green");                                   //一个动态数组
arrList.Add("red");                                     //向动态数组中添加内容
arrList.Add("blue");
arrList.Add("yellow");
this.ddlTest.DataSource = arrList;                      //将数组作为控件数据源
this.ddlTest.DataBind();                                //对控件进行数据绑定
```

学习手册 注意 使用 DataBind 方法可以将数据库指定字段中的数据绑定到 DropDownList 控件中，但前提是要设置控件的 DataSource 属性。

（2）OnSelectedIndexChanged 方法

此方法会引发 SelectedIndexChanged 事件。

语法：

```
protected virtual void OnSelectedIndexChanged (EventArgs e)
```

参数 e：包含事件数据的 System.EventArgs。

【例 11.23】 当 DropDownList 控件设置完 SelectedIndexChanged 事件之后，在 HTML 代码中会自动添加 OnSelectedIndexChanged 方法。代码如下所示：

```
    <asp:DropDownList   ID="DropDownList1"   runat="server"   AutoPostBack="True"
OnSelectedIndex- Changed="DropDownList1_SelectedIndexChanged">
        <asp:ListItem Selected="True">春天</asp:ListItem>
        <asp:ListItem>夏天</asp:ListItem>
        <asp:ListItem>秋天</asp:ListItem>
        <asp:ListItem>冬天</asp:ListItem>
    </asp:DropDownList>
```

3. DropDownList 控件事件

DropDownList 控件的常用事件及说明如表 11.15 所示。

表 11.15 DropDownList 控件的常用事件及说明

事　件	说　明
DataBinding	控件绑定到数据源时发生
DataBound	控件绑定到数据源后发生
SelectedIndexChanged	当列表控件的选定项在信息发往服务器之间变化时发生
TextChanged	当 Text 和 SelectedValue 属性更改时发生
Unload	当服务器控件从内存中卸载时发生

下面对比较重要的事件进行详细介绍。

（1）SelectedIndexChanged 事件

当控件选择的项改变时，会引发 SelectedIndexChanged 事件。

语法：

```
public event EventHandler SelectedIndexChanged
```

【例 11.24】 当选择项改变时，输出选择项的文本。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
                                                                    //选择项改变
{
    Response.Write(DropDownList1.SelectedItem.Text.ToString());//输出选择项的值
}
```

（2）TextChanged 事件

在更改文本属性后将激发此事件。

语法：

```
public event EventHandler TextChanged
```

【例 11.25】 当控件的文本属性更改后，输出提示“文本属性已更改”。

```
protected void DropDownList1_TextChanged(object sender, EventArgs e)
                                                                    //文本属性改变
```

```
{
    Response.Write("文本属性已更改");                          //弹出提示信息
}
```

学习手册 说明　在使用 SelectedIndexChanged 事件之前要将 DropDownList 控件的 AutoPostBack 属性设为 True，使其自动提交到服务器，才能使 SelectedIndexChanged 事件生效。

实例位置：光盘\MR\Instance\11\11.7

【例 11.26】在控件的 SelectedIndexChanged 事件中，通过设置 BackColor 属性，实现当 DropDownList 控件列表项改变时，其背景色也做相应的改变，实例运行结果如图 11.20 所示。

图 11.20　动态改变 DropDownList 控件的背景色

程序代码如下所示：

```
protected void ddlcolor_SelectedIndexChanged(object sender, EventArgs e)
{
    string color = this. ddlcolor.SelectedItem.Value;
                                                    //获取 DropDownList 控件选择项的值
    switch (color)                                  //switch 语句根据选择的值设置背景
    {
        case "Red":                                 //选择红色
            this. ddlcolor.BackColor = System.Drawing.Color.Red;//背景设为红
            break;
        case "Green":                               //选择绿色
            this. ddlcolor.BackColor = System.Drawing.Color.Green;
                                                    //背景设为绿，以此类推
            break;
        case "Blue":
            this. ddlcolor.BackColor = System.Drawing.Color.Blue;
            break;
        case " LightGray ":
            this. ddlcolor.BackColor = System.Drawing.Color. LightGray;
            break;
        default :
            this. ddlcolor.BackColor = System.Drawing.Color.White;
            break;
    }
}
```

11.5　选择类型控件

专题讲座：光盘\MR\Video\11\选择类型控件.exe

➤➤➤视频速递：通过本视频可以对 RadioButton 控件和 CheckBox 等控件有更深的了解。

选择类型控件主要包括 RadioButton 控件（一种单选按钮控件）和 CheckBox 控件（用于在页面上创建出简单的复选框），本节将详细介绍这两种控件的用法。

11.5.1 实现单选的 RadioButton 控件

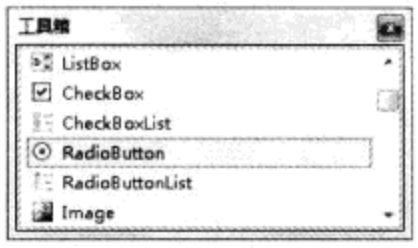


图 11.21 RadioButton 控件

RadioButton 控件是一种单选按钮控件，用户可以在页面中添加一组 RadioButton 控件，通过为所有的单选按钮分配相同的 GroupName（组名），来强制执行从给出的所有选项集中仅选择一个选项。RadioButton 控件如图 11.21 所示。

RadioButton 控件的常用属性及说明如表 11.16 所示。

表 11.16 RadioButton 控件的常用属性及说明

属　性	说　明
AutoPostBack	单击 RadioButton 控件时，是否自动回发到服务器
Checked	是否已选中 RadioButton 控件
GroupName	单选按钮所属的组名
Text	与 RadioButton 控件关联的文本标签
TextAlign	与 RadioButton 控件关联的文本标签的对齐方式
Enabled	控件是否启用
ID	分配给服务器控件的编程标识符

下面介绍一下 RadioButton 控件的一些重要属性。

（1）GroupName 属性

使用 GroupName 属性指定一组单选按钮，以创建一组互相排斥的控件。如果用户在页面中添加了一组 RadioButton 控件，可以将所有单选按钮的 GroupName 属性值设为同一个值，来强制执行所有选项集中仅有一个处于被选中状态。

语法：

```
public virtual string GroupName { get; set; }
```

属性值：单选按钮所属的组名。默认值为空字符串（""）。

只有将多个 RadioButton 控件的 GroupName 属性设为同一个名称后，才能实现单选按钮的效果，否则多个 RadioButton 控件可以同时被选中。

（2）Checked 属性

获取或设置一个值，该值指示是否已选中 CheckBox 控件。

语法：

```
public virtual bool Checked { get; set; }
```

属性值：如果为 true，则指示选中状态；否则为 false。默认为 false。

（3）ValidationGroup 属性

获取或设置在 CheckBox 控件回发到服务器时要进行验证的控件组。

语法：

```
public virtual string ValidationGroup { get; set; }
```

属性值：在 CheckBox 控件回发到服务器时要进行验证的控件组。默认值为空字符串

("")。

使用验证组可以将页上的验证控件归入特定类别。每个验证组都可以独立于页上的其他验证组进行验证。使用 ValidationGroup 属性指定当 CheckBox 控件回发到服务器时要进行验证的验证组名称。当 CausesValidation 属性值设置为 true 时，ValidationGroup 属性才起作用。如果为 ValidationGroup 属性指定了属性值，则当 CheckBox 控件回发到服务器时，只会验证属于指定组的验证控件。如果没有为 ValidationGroup 属性指定值，并且 CausesValidation 属性设置为 true，则该控件回发到服务器时，将对页上未指定验证组的所有验证控件进行验证。

实例位置：光盘\MR\Instance\11\11.8

【例 11.27】 通过设置 RadioButton 控件的 GroupName 属性值，模拟“考试系统”中的单选题，并在 RadioButton 控件的 CheckedChanged 事件下，判断用户选择的答案是否正确。执行程序并选择答案“D”，单击“提交”按钮，将会弹出“答案正确”的提示框，实例运行结果如图 11.22 所示。

图 11.22　使用 RadioButton 控件模拟考试系统

程序代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
    //如果没有选择任何一个答案
    if (rdbA.Checked == false && rdbB.Checked == false && rdbC.Checked == false
    && rdbD.Checked      == false)
    {
        //弹出提示窗口
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请选择
        答案');", true);
        return;
    }
    else                                                    //如果选择答案了
    {
        if (rdbD.Checked)                                   //判断是否选择第四项
        {
            //如果选择第四项，则答案正确，弹出提示信息
            ClientScript.RegisterStartupScript(this.GetType(),"","alert('答案
            正确');",true);
        }
    }
}
```

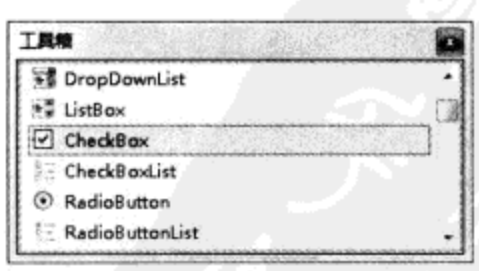


图 11.23　CheckBox 控件

11.5.2　实现多选的 CheckBox 控件

CheckBox 控件用来显示复选框，用户可以从一组 CheckBox 控件中选择一项或多项。CheckBox 控件如图 11.23 所示。

1. CheckBox 控件属性

CheckBox 控件的常用属性及说明如表 11.17 所示。

表 11.17　CheckBox 控件的常用属性及说明

属　性	说　明
AutoPostBack	获取或设置一个值，该值指示在单击 CheckBox 控件时，是否自动回发到服务器
Checked	获取或设置一个值，该值指示是否已选中 CheckBox 控件
Text	获取或设置与 CheckBox 关联的文本标签
TextAlign	获取或设置与 CheckBox 控件关联的文本标签的对齐方式
ID	获取或设置分配给服务器控件的编程标识符

下面介绍一下 CheckBox 控件的一些重要属性。

（1）AutoPostBack 属性

用于设置是否自动回发到服务器。

语法：

```
public virtual bool AutoPostBack { get; set; }
```

属性值：使用此属性来指定在单击时是否将 CheckBox 控件的状态回发到服务器。如果回发到服务器，则设置为 True，否则为 False。默认为 False。设置 AutoPostBack 属性值通常在控件的属性对话框中设置。

【例 11.28】 允许 CheckBox 控件自动回发到服务器。代码如下所示：

```
this.chkTest1.AutoPostBack = True;
```

（2）Checked 属性

使用该属性可以确定是否选中 CheckBox 控件，也可以使用该属性以编程方式设置 CheckBox 控件的状态。

语法：

```
public virtual bool Checked { get; set; }
```

属性值：如果为 True，则控件处于选中状态；否则为 False。默认为 False。

【例 11.29】 将 CheckBox 控件设为选中状态。代码如下所示：

```
this.chkTest1.Checked = True;
```

如果想让某个 CheckBox 控件被选中，可以通过代码将其 Checked 属性设为 true，也可以直接在其属性面板中将 Checked 属性设为 true。

（3）Text 属性

使用该属性来指定与 CheckBox 控件关联的文本标签。该属性也可以用于以编程方式设置与 CheckBox 控件关联的文本标签。

语法：

```
public virtual string Text { get; set; }
```

【例 11.30】 为四个 CheckBox 控件设置文本标签。代码如下所示：

```
this.chkTest1.Text = "唱歌";
this.chkTest2.Text = "跳舞";
this.chkTest3.Text = "看书";
```

```
this.chkTest4.Text = "上网";
```

运行结果如图 11.24 所示。

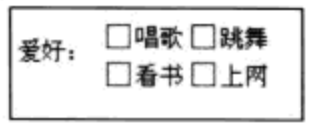


图 11.24　设置 CheckBox 控件的 Text 属性

2. CheckBox 控件事件

CheckBox 控件的常用事件及说明如表 11.18 所示。

表 11.18　CheckBox 控件的常用事件及说明

事　件	说　明
CheckedChanged	当 Checked 属性的值在向服务器进行发送期间更改时发生
DataBinding	当服务器控件绑定到数据源时发生
DataBind	当服务器控件绑定后发生
Unload	当服务器控件从内存中卸载时发生

下面主要介绍一下 CheckBox 控件的 CheckedChanged 事件。

语法：

```
public event EventHandler CheckedChanged
```

当 CheckBox 控件的选择状态发生更改时，将引发 CheckedChanged 事件。此事件不将页面回发到服务器，除非 AutoPostBack 属性被设置为 True。可以在 CheckBox 控件的 CheckedChanged 事件中编写程序。

实例位置：光盘\MR\Instance\11\11.9

【例 11.31】使用 CheckBox 控件选择个人爱好，可以选择某些爱好，也可以通过“全选”复选框选择所有爱好，并且显示选择的内容，主要应用到 CheckBox 控件的 Checked 属性、CheckedChanged 事件、AutoPostBack 属性和 Text 属性，实例运行结果如图 11.25 所示。

图 11.25　CheckBox 控件选择个人爱好

当单击“全选”按钮后，程序首先遍历窗体中的所有控件，判断控件是否是 CheckBox 控件，如果是 CheckBox 控件则通过将 Checked 属性设为 true 使其处于选中状态，程序代码如下所示：

```
protected void chbSelectAll_CheckedChanged(object sender, EventArgs e)
{
    for (int i = 0; i < Page.Controls.Count; i++)          //遍历页面控件集合
    {
        foreach (Control c in Page.Controls[i].Controls)//遍历所有控件
        {
            if (c is CheckBox)                          //判断是否是 CheckBox 控件
            {
                CheckBox cb = (CheckBox)c;              //如果是该控件则创建 CheckBox 对象
                if (chbSelectAll.Checked)               //判断“全选”复选框是否选中
                {
                    cb.Checked = true;
                        //如果“全选”复选框选中，则选中页面中的所有 CheckBox 控件
```

```
                }
                else                          //否则
                {
                    cb.Checked = false;   //取消页面中所有 CheckBox 控件的选中状态
                }
            }
        }
    }
}
```

学习手册 注意 将“全选”按钮的 AutoPostBack 属性设为 true，这样当选中“全选”按钮时，会自动回发服务器，触发 CheckedChanged 事件下的代码。

选择个人爱好之后，单击“显示”按钮，可以将选择的爱好显示在文本框中，主要是通过遍历窗体中所有 CheckBox 控件并获取控件的 Text 属性实现的，代码如下所示：

```
txtinfo.Text = "";
for (int i = 0; i < Page.Controls.Count; i++)                    //遍历页面
{
    foreach (Control c in Page.Controls[i].Controls)             //遍历控件集合
    {
        if (c is CheckBox)                          //判断控件是否为 CheckBox 控件
        {
            CheckBox cb = (CheckBox)c;              //如果是该控件则创建 CheckBox 对象
            if (cb.Checked&&cb.Text!="全选")        //排除“全选”按钮
            {
                txtinfo.Text += cb.Text + "\n";//获取其他所有 CheckBox 控件显示的文本
            }
        }
    }
}
```

11.6 图形控件显示图像

专题讲座：光盘\MR\Video\11\图形显示类型控件.exe

➤➤➤视频速递：通过本视频可以对 Image 控件和 ImageMap 控件有更深的了解。

图形显示类型控件主要包括 Image 控件和 ImageMap 控件。使用 Image 控件可以在设计或运行时以编程方式为 Image 对象指定图形文件。ImageMap 控件允许在图片中定义一些热点（HotSpot）区域。当用户单击这些热点区域时，将会引发超链接或者单击事件，学习本节之后，读者将会掌握这两个控件的用法。

11.6.1 显示图像的 Image 控件

使用 Image 控件可以在设计或运行时以编程方式为 Image 对象指定图形文件，还可以将控件的 ImageUrl 属性绑定到一个数据源，根据数据库信息显示图形，Image 控件如图 11.26 所示。

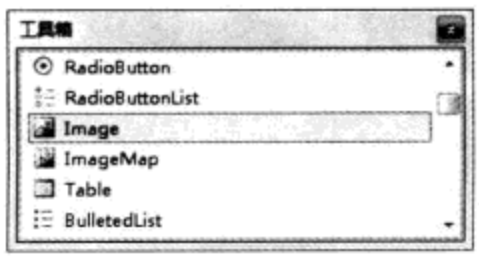


图 11.26 Image 控件

Image 控件的常用属性及说明如表 11.19 所示。

表 11.19　Image 控件的常用属性及说明

属　性	说　明
ID	获取或设置分配给服务器控件的编程标识符
AlternateText	在图像无法显示时显示的替换文字
ImageAlign	获取或设置 Image 控件相对于网页上其他元素的对齐方式
ImageUrl	获取或设置在 Image 控件中显示的图像的位置
Enabled	获取或设置一个值，该值指示是否已启用控件
DescriptionUrl	获取或设置图像详细说明的位置
ToolTip	获取或设置当鼠标指针悬停在 Web 服务器控件上时显示的文本

下面对一些比较常用的属性进行一下详细说明。

（1）ImageAlign 属性

ImageAlign 属性指定或确定图像相对于网页上其他元素的对齐方式。

语法：

```
public virtual ImageAlign ImageAlign { get; set; }
```

属性值：ImageAlign 值之一，默认为 NotSet，ImageAlign 属性值在表 11.20 中列出。

表 11.20　Image 控件的 ImageAlign 属性值

对 齐 方 式	说　明
Left	图像沿网页的左边缘对齐，文字在图像右边换行
Right	图像沿网页的右边缘对齐，文字在图像左边换行
Baseline	图像的下边缘与第一行文本的下边缘对齐
Top	图像的上边缘与同一行上最高元素的上边缘对齐
Middle	图像的中间与第一行文本的下边缘对齐
Bottom	图像的下边缘与第一行文本的下边缘对齐
AbsBottom	图像的下边缘与同一行中最大元素的下边缘对齐
AbsMiddle	图像的中间与同一行中最大元素的中间对齐
TextTop	图像的上边缘与同一行上最高文本的上边缘对齐

（2）ImageUrl 属性

ImageUrl 属性用于设置在 Image 控件中显示图像的位置（URL）。

语法：

```
public virtual string ImageUrl { get; set; }
```

属性值：在 Image 控件中显示图像的 URL 路径位置。

在设置 ImageUrl 属性值时，可以使用相对 URL，也可以使用绝对 URL。相对 URL 使图像的位置与网页的位置相关联，当整个站点移动到服务器上的其他目录时，不需要修改 ImageUrl 属性值；而绝对 URL 使图像的位置与服务器上的完整路径相关联，当改更站点路径时，需要修改 ImageUrl 属性值。笔者建议，在设置 Image 控件的 ImageUrl 属性值时，使用相对 URL。

（3）DescriptionUrl 属性

获取或设置图像详细说明的位置。DescriptionUrl 属性指定提供图像附加详细信息的 HTML 文件。

语法：

```
public virtual string DescriptionUrl { get; set; }
```

属性值：包含图像详细说明的文件的 URL。默认值为空字符串（""）。

（4）ToolTip 属性

获取或设置当鼠标指针悬停在 Web 服务器控件上时显示的文本。

语法：

```
public virtual string ToolTip { get; set; }
```

图 11.27　显示选择的头像

属性值：当鼠标指针悬停在 Web 服务器控件上时显示的文本，默认为 String.Empty。

实例位置：光盘\MR\Instance\11\11.10

【例 11.32】通过改变 Image 控件的 ImageUrl 属性值来动态显示用户头像，这样的例子随处可见，例如，网站的会员注册页面就会有选择头像的功能。本例主要是通过 Image 控件和 DropDownList 控件相结合实现的，实例运行结果如图 11.27 所示。

程序代码如下所示：

```
    <asp:DropDownList ID="ddlselect" runat="server" AutoPostBack="True" Height=
"22px" Width="98px"
          onselectedindexchanged="DropDownList1_SelectedIndexChanged">
          <asp:ListItem Selected="True" Value="1.jpg">头像 1</asp:ListItem>
          <asp:ListItem Value="2.jpg">头像 2</asp:ListItem>
          <asp:ListItem Value="3.jpg">头像 3</asp:ListItem>
          <asp:ListItem Value="4.jpg">头像 4</asp:ListItem>
    </asp:DropDownList>
```

在 DropDownList 控件的 SelectedIndexChanged 事件下编写代码，实现当选择某个头像后，设置 Image 控件的 ImageUrl 属性为 DropDownList 控件选择的图片路径，代码如下所示：

```
img.ImageUrl = ddlselect.SelectedValue;                    //动态改变控件显示的图像
```

学习手册 说明 在使用 DropDownList 控件选择头像时，首先要将控件的 AutoPostBack 属性设为 true，这样，当选择某个头像后，会自动回发到服务器，执行 DropDownList 控件 SelectedIndexChanged 事件下的代码。

11.6.2　设置热点区域的 ImageMap 控件

ImageMap 控件允许在图片中定义一些热点（HotSpot）区域。当用户单击这些热点区

域时，将会引发超链接或者单击事件。当需要对某幅图片的局部实现交互时，使用 ImageMap 控件。例如，以图片形式展示网站地图、流程图等，ImageMap 控件如图 11.28 所示。

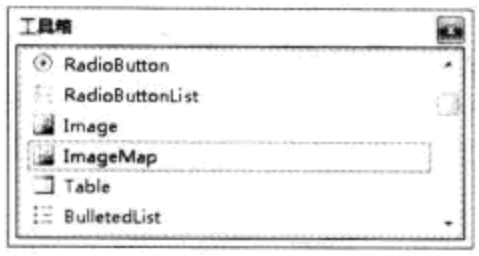


图 11.28　ImageMap 控件

ImageMap 控件的常用属性及说明如表 11.21 所示。

表 11.21　Image 控件的常用属性及说明

属　性	说　明
ID	获取或设置分配给服务器控件的编程标识符
AlternateText	在图像无法显示时显示的替换文字
HotSpotMode	获取或设置单击 HotSpot 对象时 ImageMap 控件的 HotSpot 对象的默认行为
HotSpots	获取 HotSpot 对象的集合，这些对象表示 ImageMap 控件中定义的作用点区域
ImageAlign	获取或设置 Image 控件相对于网页上其他元素的对齐方式
ImageUrl	获取或设置在 Image 控件中显示的图像的位置
Target	获取或设置单击 ImageMap 控件时显示链接到的网页内容的目标窗口或框架
Enabled	获取或设置一个值，该值指示是否已启用控件

下面对比较重要的属性进行详细介绍。

（1）HotSpotMode 属性

HotSpotMode 属性用于获取或者设置单击热点区域后的默认行为方式。

语法：

```
public virtual HotSpotMode HotSpotMode { get; set; }
```

属性值：HotSpotMode 枚举值之一。默认为 NotSet。使用 HotSpotMode 属性，可以指定 ImageMap 控件内 HotSpot 对象的默认单击行为。使用一个 HotSpotMode 枚举值设置此属性，在表 11.22 中列出了 HotSpotMode 属性的枚举值。

表 11.22　HotSpotMode 属性的枚举值

枚 举 值	说　明
Inactive	无任何操作，即此时形同一张没有热点区域的普通图片
NotSet	未设置项，同时也是默认项。虽然名为未设置，但是默认情况下将执行定向操作，即链接到指定的 URL 地址。如果未指定 URL 地址，则默认链接到应用程序根目录下
Navigate	定向操作项。链接到指定的 URL 地址。如果未指定 URL 地址，则默认链接到应用程序根目录下
PostBack	回传操作项。单击热点区域后，将触发控件的 Click 事件

HotSpotMode 属性虽然为图片中所有热点区域定义了单击事件的默认行为方式，然而在某些情况下，图片中热点区域的行为方式各不相同，需要单独为每个热点区域定义 HotSpotMode 属性及其相关属性。

（2）HotSpots 属性

获取 HotSpot 对象的集合，这些对象表示 ImageMap 控件中定义的作用点区域。

语法：

```
public HotSpotCollection HotSpots { get; }
```

属性值：一个 HotSpotCollection 对象，表示 ImageMap 控件中定义的作用点区域。

HotSpots 属性用于获取 HotSpots 对象集合。HotSpot 类是一个抽象类，它包含 CircleHotSpot（圆形热区）、RectangleHotSpot（方形热区）和 PolygonHotSpot（多边形热区）3 个子类。这些子类的实例称为 HotSpot 对象。创建 HotSpot 对象的步骤说明如下。

首先在 ImageMap 控件上单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，弹出属性窗口。在属性窗口中，单击 HotSpots 属性后面的 [...] 按钮，弹出“HotSpot 集合编辑器”窗口。在该窗口中，单击“添加”按钮后的 [·] 按钮，将会弹出一个下拉菜单，该下拉菜单中包括 CircleHotSpot（圆形热区）、RectangleHotSpot（方形热区）和 PolygonHotSpot（多边形热区）3 个对象，可以通过单击添加该对象，如图 11.29 所示。

然后在右侧窗口中为热点（HotSpot）区域设置属性。

在定义每个热点区域的过程中，主要设置两个属性：一个是 HotSpotMode 及其相关属性，用于指定图像映射是否导致回发或导航行为；另一个是热点区域坐标属性。对于 CircleHotSpot(圆形热区)，需要设置半径 Radius 和圆心坐标 X 和 Y。对于 RectangleHotSpot（方形热区），需要设置其左上右下的坐标，即 Left、Top、Right、Bottom 属性。对于 PolygonHotSpot（多边形热区），需要设置每一个关键点的坐标 Coordinates 属性。

最后单击“确定”按钮，创建完成。

实例位置：光盘\MR\Instance\11\11.11

【例 11.33】 本实例主要是使用 ImageMap 控件展示图片中的链接热区，在卡通人物的额头和嘴巴位置设置热区，当鼠标单击某个热区后，会弹出相应的提示，例如，单击卡通人物的嘴巴，则弹出“点到嘴巴啦！”的提示，实例运行结果如图 11.30 所示。

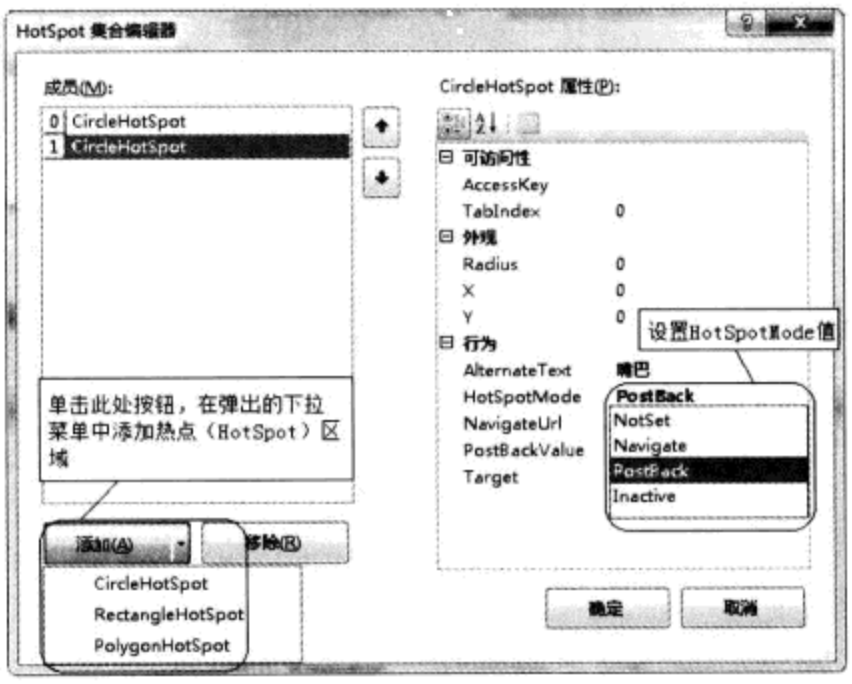


图 11.29 HotSpot 集合编辑器

图 11.30 单击 ImageMap 控件中的热区弹出提示信息

程序代码如下所示：

```
    <asp:ImageMap  ID="ImageMap1"  runat="server"  ImageUrl="~/2.jpg"  onclick=
"ImageMap1_Click">
        <asp:CircleHotSpot AlternateText="嘴巴" HotSpotMode="PostBack" PostBack
        Value="嘴巴"
```

```
        Radius="5" X="50" Y="60" />
    <asp:CircleHotSpot AlternateText="额头" HotSpotMode="PostBack" PostBack
    Value="额头"
        Radius="5" X="50" Y="40" />
</asp:ImageMap>
```

在 ImageMap 控件的 Click 事件中，通过 PostBackValue 属性获取单击某个热区后的返回值，然后弹出相应的提示，代码如下所示：

```
protected void ImageMap1_Click(object sender, ImageMapEventArgs e)
{
    string backValue = e.PostBackValue;          //获取所点热区的 PostBackValue 值
    string str = string.Empty;                   //声明字符串记录提示信息
    //switch 语句根据不同 PostBackValue 值，设置相应的提示信息
    switch (backValue)
    {
        //如果 PostBackValue 为“嘴巴”，则字符串 str 赋值为“点到嘴巴啦！”
        case "嘴巴": str = "点到嘴巴啦！"; break;
        //如果 PostBackValue 为“额头”，则字符串 str 赋值为“点到额头啦！”
        case "额头": str = "点到额头啦！"; break;
    }
    //弹出相应的提示信息
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('"+str+"');
    ", true);
}
```


设置热区的时候，比较难设置的是热区的坐标，这就需要凭借观察力先估算一个值，然后一点点地设置。

11.7　文件上传控件上传文件

ASP.NET 中如果要将文件等上传到服务器就要应用到 FileUpload 文件上传控件，该控件的功能非常强大，使用起来也非常简单。编写短短几行代码，就可以实现将选择的文件上传到服务器中。在没有特殊功能要求下，使用该控件上传文件游刃有余。

11.7.1　FileUpload 控件的概述

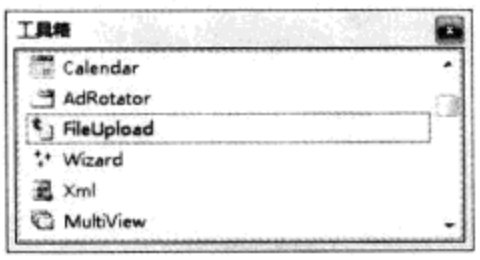


图 11.31　FileUpload 控件

FileUpload 控件的主要功能是向指定目录上传文件。该控件包括一个文本框和一个浏览按钮。用户可以在文本框中输入完整的文件路径，或者通过按钮浏览并选择需要上传的文件。FileUpload 控件不会自动上传文件，必须设置相关的事件处理程序，并在程序中实现文件上传。FileUpload 控件如图 11.31 所示。

11.7.2　FileUpload 控件的属性

FileUpload 控件部分常用属性及说明如表 11.23 所示。

表 11.23　FileUpload 控件部分常用属性及说明

属　性	说　明
ID	设置控件的编程标识符
Text	设置或返回控件的文本
Width	控件的宽度
FileBytes	从使用 FileUpload 控件指定的文件返回一个字节数组
FileContent	获取 Stream 对象，它指向要使用 FileUpload 控件上载的文件
FileName	获取客户端上使用 FileUpload 控件上载的文件的名称
HasFile	获取一个值，该值指示 FileUpload 控件是否包含文件
PostedFile	获取使用 FileUpload 控件上载的文件的基础 HttpPostedFile 对象

下面对比较重要的属性进行详细介绍。

（1）FileBytes 属性

用于从使用 FileUpload 控件指定的文件中返回一个字节数组。

语法：

```
public byte[] FileBytes { get; }
```

属性值：byte 数组，包含指定文件的内容。

FileUpload 控件不会自动从客户端读取该文件，必须显式提供一个控件或机制，使用户能提交指定的文件。

【例 11.34】 获得上传文件的大小。

```
FileUpload1.FileBytes.Length;
```

（2）FileName 属性

使用该属性获取客户端上使用 FileUpload 控件上传的文件的名称。

语法：

```
public string FileName { get; }
```

属性值：该字符串指定客户端上使用 FileUpload 上载的文件的名称。

FileName 属性返回的文件名不包含此文件在客户端上的路径。

【例 11.35】 获得上传文件的名称。

```
FileUpload1.FileName;
```

（3）HasFile 属性

HasFile 属性获取一个值，该值指示 FileUpload 控件是否包含要上传的文件。

语法：

```
public bool HasFile { get; }
```

属性值：如果 FileUpload 包含文件，则为 True；否则为 False。

【例 11.36】 判断文件是否存在，如果存在则输入文件名；如果不存在则提示“文件不存在”消息框。

```
if (this.ful.HasFile)                                          //判断是否选择文件
{
    Response.Write("文件名为："+FileUpload1.FileName);          //显示文件名
}
else                                                           //如果没有选择
{
    Response.Write("<script>alert('文件不存在！');</script>");  //弹出提示信息
}
```

在对文件执行操作之前，使用该属性来验证要上传的文件是否存在。如果 FileUpload 包含文件，则为 True；否则为 False。

11.7.3　FileUpload 控件的方法

FileUpload 控件的常用方法及说明如表 11.24 所示。

表 11.24　FileUpload 控件的常用方法及说明

方　法	说　明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
Focus	为控件设置输入焦点
ResolveClientUrl	获取浏览器可以使用的 URL
ResolveUrl	将 URL 转换为在请求客户端可用的 URL
SaveAs	使用 FileUpload 控件将上传的文件内容保存到 Web 服务器上的指定路径
ToString	返回表示当前对象类型的完全限定名

下面主要对 SaveAs 方法进行详细介绍。

SaveAs 方法将 FileUpload 控件上传的文件内容保存到服务器上指定的路径。

语法：

```
public void SaveAs (string filename)
```

filename：服务器上保存上传文件的位置。

11.7.4　情景应用：上传图片并获取相关信息

学习完文件上传控件后，读者一定跃跃欲试，想要自己开发一个上传文件的程序，别着急，本实例将会详细介绍文件上传的整个过程。

专题讲座：光盘\MR\Video\11\上传图片并获取相关信息.exe

视频速递：理解如何在实际中应用 FileUpload 控件上传文件。

实例位置：光盘\MR\Instance\11\11.12

【例 11.37】 使用 FileUpload 控件上传图片文件，并将原文件路径、文件大小和文件类型显示出来，实例运行结果如图 11.32 所示。

图 11.32　FileUpload 控件上传图片

程序代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)                                    //如果选择了上传文件
    {
      if (FileUpload1.PostedFile.ContentType.Substring(0,5) == "image")
                                                                //判断是否是图像文件
        {
          try
          {
             string serverPath = Server.MapPath("upLoad");
                                                        //获取服务器中的上传目录
             if (!System.IO.Directory.Exists(serverPath))     //如果不存在
             {
                System.IO.Directory.CreateDirectory(serverPath); //则创建该目录
             }
            string imgName = FileUpload1.FileName;    //获取上传文件的名称
             string newPath = serverPath + "\\" + imgName;
                                                        //设置图片在服务器端的新路径
             FileUpload1.SaveAs(newPath);
             ClientScript.RegisterStartupScript(this.GetType(), "", "alert('
             上传成功！');", true);
             txtinfo.Text = "";
             img.ImageUrl = "upLoad/"+imgName;          //显示上传的图片
             txtinfo.Text +="路径："+ FileUpload1.PostedFile.FileName+"\n"+"
             大小："+FileUpload1. PostedFile.ContentLength/1024+"\n"+"类型：
             "+FileUpload1.PostedFile.ContentType;
          }
          catch
          {
             ClientScript.RegisterStartupScript(this.GetType(), "", "alert
             ('上传失败！');", true);
          }
    }
    else
    {
       ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请
```

```
                选择图片！');", true);
            }
        }
        else
        {
            ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请选择
            要上传的图片！');", true);
        }
    }
```

学习手册 说明 ContentType 属性可以获得上传文件的类型，但是，此处要注意 ContentType 属性值的格式。例如，上传一个 gif 格式的图片，则该属性的属性值是 image/gif。所以本例只判断前五个字符是否是 image，来证明上传的文件是否是图片文件。FileUpload 控件的 FileName 属性只获取上传文件的名称和扩展名，不能获取文件在客户端的全路径。

11.8　实 战 练 习

11.8.1　限制文本框中输入的字符长度

➤➤➤题目描述

在开发程序的时候，经常会遇到限制文本框中输入的字符长度的情况。例如，开发会员注册页面，在该页面中用户注册的用户名的长度需要有一定的限制，而用户名是在文本框中输入的，所以，要设置文本框允许输入的字符长度，效果如图 11.33 所示。

图 11.33　限制文本框中输入的字符长度

➤➤➤技术指导

本实例主要设置 TextBox 控件的 MaxLength 属性。该属性用于设置文本框中最多允许的字符数，默认值为 0，表示没有长度限制。通过该属性可以很好地限制文本框中输入的字符长度，其语法格式如下：

```
[BindableAttribute(true)]
public int MaxLength { get; set; }
```

属性值：文本框中最多允许的字符数。

学习手册 说明 TextBox 控件的 MaxLength 属性可以通过控件的属性窗口进行设置，也可以通过代码进行设置。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/11.8.1**

11.8.2 显示验证码图片

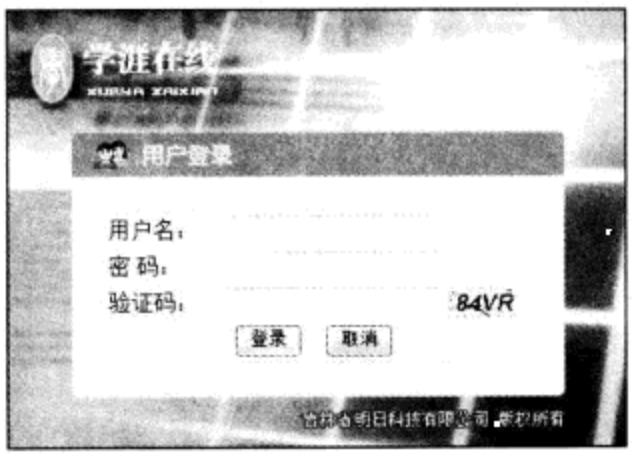


图 11.34 显示验证码图片

➤➤➤题目描述

现在网络安全已成为一个不可忽视的问题，任何一个网站都设有用户登录模块，在该模块中主要验证用户登录时填写的信息是否正确。在用户登录模块中除了验证用户登录信息外，还会提供验证码验证，主要是为了防止机器人登录。本实例将演示如何在用户登录模块中显示验证码图片，运行结果如图 11.34 所示。

➤➤➤技术指导

本实例使用 Image 客户端控件在网页中显示已经生成的验证码图片，程序代码如下。

```
<img id="IMG1" runat="server" src="checkcode.aspx" />
```

src 属性：用来设置生成验证码图片的页面。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/11.8.2**

11.8.3 带图像的登录按钮

➤➤➤题目描述

ImageButton 控件为图像按钮控件，它用于显示具体的图像，在功能上和 Button 控件相同。但是，ImageButton 控件可以显示指定的图片，例如，在开发会员登录界面时，就使用了 ImageButton 控件，如图 11.35 所示。

➤➤➤技术指导

开发本实例时主要是通过 ImageButton 控件的 ImageUrl 属性设置要显示的图像地址。

语法：

```
public virtual string ImageUrl { get; set; }
```

属性值：在 Image 控件中显示的图像的位置。

图 11.35　带图像的登录按钮

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/11.8.3**

11.9　本 章 小 结

服务器控件对于开发人员来说是必须要掌握的基础知识。通过本章读者学习到了服务器控件的基本概述及基本操作，并且通过实例掌握了服务器端标准控件及验证控件的应用。服务器控件是 Web 网站开发不可缺少的一部分，所以，其作用显得尤为重要，通过本章的学习读者能够完全掌握常用服务器控件的用法。

第 12 章

数据绑定控件

（名师课堂：1 小时 26 分钟）

ASP.NET 中提供了多种数据控件，用于在 Web 页中显示数据。这些控件具有丰富的功能，例如分页、排序、编辑等。开发人员只需要简单配置一些属性，就能够在几乎不编写代码的情况下，快速、正确地完成任务，通过本章的学习，读者可以掌握以下知识：

- 使用 GridView 控件绑定数据源
- 制定 GridView 控件的列
- 使用 GridView 控件分页显示数据及编辑数据
- 使用 DataList 控件删除（及批量删除）数据
- 使用 DataList 控件编辑数据

12.1　GridView 控件基本应用

专题讲座：光盘\MR\Video\12\ GridView 控件显示绑定数据.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握与 GridView 控件相关的操作。

GridView 控件可称之为表格控件，顾名思义，其数据显示形式是以表格形式来体现，并且该控件有自带的编辑、删除、排序等功能，通过对本节的学习，读者可以对 GridView 控件有一个基本的认识。

12.1.1　关于 GridView 控件

显示表格数据是软件开发中的一个周期性任务。ASP.NET 提供了许多工具来在网格中显示表格数据，例如 GridView 控件。通过使用 GridView 控件，用户可以显示、编辑和删除多种不同的数据源（例如数据库、XML 文件和公开数据的业务对象）中的数据。

既然是数据显示控件，那么就先来看一下如何将数据绑定到 GridView 控件上，GridView 控件提供了两个用于绑定到数据的选项。

☑ 使用 DataSource 属性进行数据绑定，此选项能够绑定到包括 ADO.NET 数据集和数据读取器在内的各种对象。使用此方法时，GridView 控件可以不提供内置的排序、分页和更新等功能，需要使用适当的事件代码提供此功能。

☑ 使用 DataSourceID 属性进行数据绑定，此选项能够将 GridView 控件绑定到数据源控件（如 SqlDataSource 等）。使用这种绑定方式，GridView 控件可以提供内置的排序、分页和更新功能。

当使用 DataSourceID 属性绑定到数据源时，GridView 控件支持双向数据绑定。除可以使该控件显示返回的数据之外，还可以使它自动支持对绑定数据的更新和删除操作。

12.1.2　GridView 控件分页绑定数据

GridView 控件有一个内置分页功能，可支持基本的分页功能。在启用其分页机制前需要设置 AllowPaging 和 PageSize 属性，AllowPaging 决定是否启用分页功能，PageSize 决定分页时显示几条数据（默认值为 12）。

实例位置：光盘\MR\Instance\12\12.1

【例 12.1】 本实例利用 GridView 控件的内置分页功能实现分页显示数据。执行程序，如果数据超过 6 条，则进行分页显示，运行结果如图 12.1 所示。

图 12.1　使用 GridView 控件分页显示数据

首先将 GridView 控件的 AllowPaging 属性设置为 true 表示允许分页；然后将 PageSize 属性设置为 5，表示每页最多显示 5 条数据，并在页面的 Page_Load 事件中，对页面进行初始化，将数据表中的数据绑定到 GridView 控件中，代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    //定义数据库连接字符串
    string strCon = ConfigurationManager.AppSettings["conStr"].ToString();
    string sqlstr = "select * from tb_user";          //定义执行查询操作的 SQL 语句
    SqlConnection con = new SqlConnection(strCon);          //创建数据库连接对象
    SqlDataAdapter da = new SqlDataAdapter(sqlstr, con);    //创建数据适配器
    DataSet ds = new DataSet();                             //创建数据集
    da.Fill(ds);                                            //填充数据集
    //设置 GridView 控件的数据源为创建的数据集 ds
    Gvinfo.DataSource = ds;
    //将数据库表中的主键字段放入 GridView 控件的 DataKeyNames 属性中
    Gvinfo.DataKeyNames = new string[] { "ID" };
    Gvinfo.DataBind();                                      //绑定数据库表中数据
}
```

学习手册 说明　要对 GridView 控件进行自定义列，必须先取消 GridView 自动产生字段的功能，这里只要将 GridView 的 AutoGenerateColumns 属性设置为 false 即可。

在 GridView 控件的属性窗口中单击事件图标“⚡”，找到 PageIndexChanging 事件，并双击对应的右边框任意空白处使其产生 GridView1_PageIndexChanging 事件处理程序，在事件处理程序中主要是设置当前页的索引值，并重新绑定 GridView 控件，具体代码如下所示：

```
protected void Gvinfo_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    Gvinfo.PageIndex = e.NewPageIndex;                      //获取当前分页的索引值
    Gvinfo.DataBind();                                      //重新绑定数据
}
```

若 GridView 控件启用了分页功能，假设每页显示的是 12 条数据，那么第二页显示的第一条数据的PageIndex是0而并非12。索引值(Index)指的是排序顺序，在.NET里索引值都是从 0 开始的。

12.1.3　以编程方式实现选中、编辑和删除 GridView 数据项

在 GridView 控件的按钮列中包括一种“编辑、更新、取消”的按钮，这 3 个按钮分别触发 GridView 控件的 RowEditing、RowUpdating、RowCancelingEdit 事件，从而完成对指定项的编辑、更新和取消操作功能。

实例位置：光盘\MR\Instance\12\12.2

【例 12.2】 利用 GridView 控件的 RowDeleting、RowCancelingEdit、RowEditing 和 RowUpdating 事件，对指定项的信息进行删除和编辑操作。执行程序，实例运行结果如图 12.2 所示。

新建一个网站，默认主页为 Default.aspx。添加一个 GridView 控件，并分别为 GridView 控件添加一个编辑列和一个嵌入 ImageButton 控件（执行删除操作）的模板列，如图 12.3 所示。

图 12.2　在 GridView 控件中对数据进行编辑和删除操作

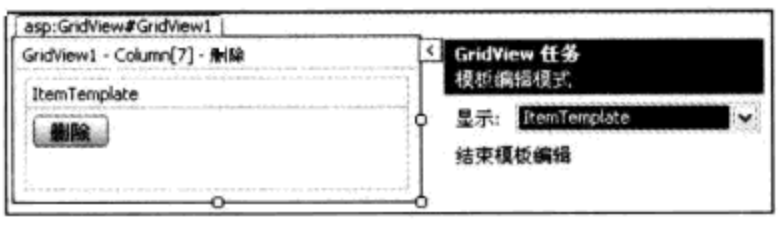


图 12.3　模板列中添加一个 ImageButton 控件用于执行删除操作

在页面的后台 Page_Load 事件中调用一个自定义方法 BindData()，该方法主要是用来将查询结果绑定到 GridView 控件上。编写的具体代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        BindData();                                    //调用自定义方法绑定数据到控件
    }
}
public void BindData()
{
    //定义数据库连接字符串
```

```
    string strCon = ConfigurationManager.AppSettings["conStr"].ToString();
    string sqlstr = "select * from tb_user";        //定义执行查询操作的 SQL 语句
    SqlConnection con = new SqlConnection(strCon);          //创建数据库连接对象
    SqlDataAdapter da = new SqlDataAdapter(sqlstr, con);    //创建数据适配器
    DataSet ds = new DataSet();                             //创建数据集
    da.Fill(ds);                                            //填充数据集
    //设置 GridView 控件的数据源为创建的数据集 ds
    Gvinfo.DataSource = ds;
    //将数据库表中的主键字段放入 GridView 控件的 DataKeyNames 属性中
    Gvinfo.DataKeyNames = new string[] { "ID" };
    Gvinfo.DataBind();                                      //绑定数据库表中数据
}
```

在自定义的 BindData 方法中，要注意设置 GridView 控件的 DataKeyNames 属性，只有设置了该属性，才能够通过 DataKeys 属性获取每行关键字段的值，读者可以参考下面的代码来掌握 DataKeyNames 属性的用法。

当用户单击“编辑”按钮时，将触发 GridView 控件的 RowEditing 事件。在该事件的程序代码中将 GridView 控件编辑项索引设置为当前选择项的索引，并重新绑定数据。代码如下所示：

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    Gvinfo.EditIndex = e.NewEditIndex;                      //设置编辑项的索引
    BindData();                                             //数据绑定
}
```

在“编辑”状态下，当用户单击“更新”按钮时，将触发 GridView 控件的 RowUpdating 事件。在该事件的程序代码中，首先获得编辑行的关键字段的值并取得各文本框中的值，然后将数据更新至数据库，最后重新绑定数据。代码如下所示：

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
     //取得编辑行的关键字段的值
    string id = GridView1.DataKeys[e.RowIndex].Value.ToString();
    //取得文本框中输入的内容
    string Nname=((TextBox)(GridView1.Rows[e.RowIndex].Cells[1].Controls[0])).
    Text.ToString().Trim();
    string Nsex=((TextBox)(GridView1.Rows[e.RowIndex].Cells[2].Controls[0])).
    Text.ToString().Trim();
    string Nage=((TextBox)(GridView1.Rows[e.RowIndex].Cells[3].Controls[0])).
    Text.ToString().Trim();
    string Naddr=((TextBox)(GridView1.Rows[e.RowIndex].Cells[4].Controls[0])).
    Text.ToString().Trim();
    //定义更新操作的 SQL 语句
    string update_sql = "update tb_user set name='" + Nname + "',sex='" + Nsex +
    "',age='" + Nage + "',addr='" + Naddr + "' where ID='" + id + "'";
    bool update = ExceSQL(update_sql);                  //调用 ExceSQL 执行更新操作
    if (update)                                         //如果返回值为 true
    {
        Response.Write("<script language=javascript>alert('修改成功！')
        </script>");
        //设置 GridView 控件的编辑项的索引为－1，即取消编辑
        Gvinfo.EditIndex = -1;
        BindData();
```

```
    }
    else
    {
        Response.Write("<script language=javascript>alert('修改失败！');
        </script>");
    }
}
```

在上述代码中，请读者注意“GridView1.DataKeys[e.RowIndex].Value”。这段代码用来获取编辑行关键字段的值。例如，我们将 GridView 控件的 DataKeyNames 属性设为 ID，那么 GridView1.DataKeys[e.RowIndex].Value 获取的是编辑行的数据在数据表中的 ID 值。

上述事件代码中调用了一个自定义方法 ExceSQL，用来执行本实例中为了更新和删除数据所定义的 SQL 语句，具体代码如下所示：

```
public bool ExceSQL(string strSqlCom)
{
    //定义数据库连接字符串
    string strCon = ConfigurationManager.AppSettings["conStr"].ToString();
    SqlConnection sqlcon = new SqlConnection(strCon);      //创建数据库连接对象
    SqlCommand sqlcom = new SqlCommand(strSqlCom, sqlcon);
    try
    {
        if (sqlcon.State == System.Data.ConnectionState.Closed)
                                                    //判断数据库是否为连接状态
        { sqlcon.Open(); }
        sqlcom.ExecuteNonQuery();                   //执行 SQL 语句
        return true;                                //SQL 语句执行成功，返回 true 值
    }
    catch
    {
        return false;                               //SQL 语句执行失败，返回 false 值
    }
    finally
    {
        sqlcon.Close();                             //关闭数据库连接
    }
}
```

本例使用的是 SQL Server 2005 数据库，读者在运行程序时，请自行更改数据库的连接字符串，以防止运行出错。

在 GridView 控件的 RowDeleting 删除事件中，首先获取删除行关键字段的值，然后编写一条 delete 语句，最后调用自定义方法 ExceSQL 执行 SQL 语句，并重新调用 BindData 方法绑定数据，代码如下所示。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string delete_sql = "delete from tb_user where ID='" + GridView1.DataKeys[e.
    RowIndex].Value . ToString() + "'";
    bool delete = ExceSQL(delete_sql);          //调用 ExceSQL 执行删除操作
    if (delete)
    {
```

```
            Response.Write("<script language=javascript>alert('删除成功！')
            </script>");
            BindData();                                    //调用自定义方法重新绑定控件中数据
        }
        else
        {
            Response.Write("<script language=javascript>alert('删除失败！')
            </script>");
        }
    }
```

学习手册 说明 这样设置后就可为程序的删除功能多加一道确认步骤。这一确认步骤是很有必要的，如果单击删除后，把数据全部删除了或删错了某条数据，那就天下大乱了！

12.2 GridView 控件 72 般绝技

专题讲座：光盘\MR\Video\12\ GridView 控件 72 般绝技.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 GridView 控件强大功能的实现。

本节将介绍 GridView 控件的一些具体应用。本节介绍的这些功能都是在实际开发中经常用到的，实用性很强。希望读者能够认真学习本节的内容。

12.2.1 一次性编辑 GridView 控件所有行中的数据

在实际项目中，为了减少往返提交数据，可以一次编辑 GridView 的所有行，这样大大改善了用户使用产品的体验。那么如何一次性更新所有修改过的记录呢？这里我们提供了两种方法，一种是使用 SqlDataSource 来更新所有记录；另一种是用组合 SQL 语句来进行的，该方法速度比较快，原理也比较容易理解！

实例位置：光盘\MR\Instance\12\12.3

【例 12.3】 本实例通过组合 SQL 语句来实现一次性编辑 GridView 控件的所有行中数据，当用户修改表格中的数据之后，单击“一次性修改所有行”按钮，可以批量修改表格中的所有数据，实例运行结果如图 12.4 所示。

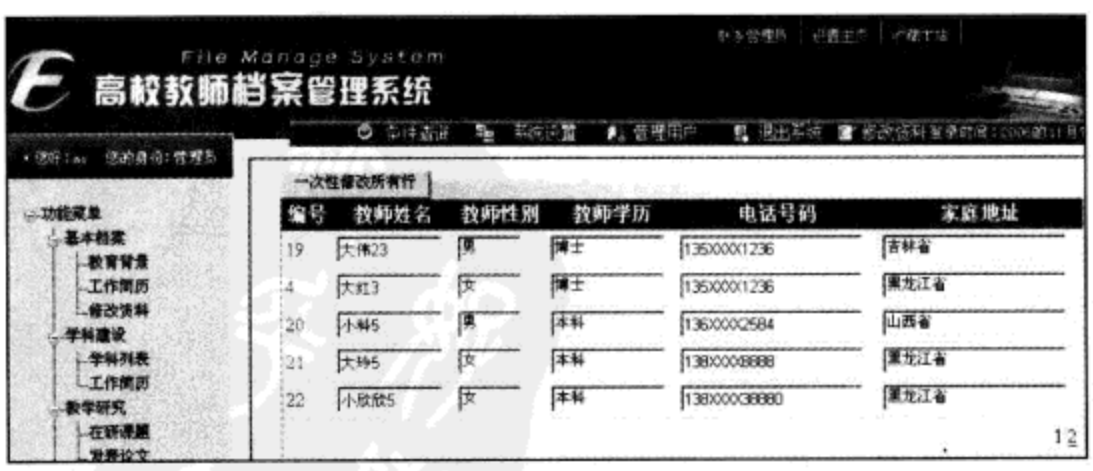


图 12.4 一次性编辑 GridView 控件的所有行中数据

首先自定义一个 bind 方法，用于检索数据表 tb_mrEmp 中的所有数据，并绑定到

GridView 控件中，代码如下所示：

```
public void bind()
{
    string sqlstr = "select * from tb_mrEmp"; //编写 SQL 语句，检索 tb_mrEmp 表
    SqlDataAdapter myda = new SqlDataAdapter(sqlstr, conn);
                                                //创建 SqlDataAdapter 对象
    DataSet myds = new DataSet();               //创建 DataSet 对象
    conn.Open();                                //打开数据库连接
    myda.Fill(myds, "tb_mrEmp");                //Fill 方法填充数据集
    Gvinfo.DataSource = myds;                   //设置 GridView 控件数据源
    Gvinfo.DataBind();                          //绑定数据
    conn.Close();                               //关闭数据库连接
}
```

读者在编写代码时，要养成及时关闭数据库连接的习惯，这样能更好地节省服务器的开销，提高程序的运行效率。

更改数据之后，单击“一次性修改所有行”按钮，可以修改当前分页中所有行的数据，其思路是遍历 GridView 控件的所有行并获取每行中文本框的内容，然后批量修改数据表中相应的数据，代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
    StringBuilder query = new StringBuilder();          //创建 StringBuilder 对象
    for (int i = 0; i < GridView1.Rows.Count; i++)      //遍历所有行
    {
        GridViewRow row = GridView1.Rows[i];            //实例化 GridViewRow 对象
        //获取每行中文本框的值
        string value1 = ((TextBox)row.Cells[1].FindControl("txtName")).Text.
        Replace("'", "''");
        string value2 = ((TextBox)row.Cells[2].FindControl("txtSex")).Text.
        Replace("'", "''");
        string value3 = ((TextBox)row.Cells[3].FindControl("txtState")).Text.
        Replace("'", "''");
        string value4 = ((TextBox)row.Cells[4].FindControl("txtPhone")).Text.
        Replace("'", "''");
        string value5 = ((TextBox)row.Cells[5].FindControl("txtAddress")).Text.
        Replace("'", "''");
        string value = GridView1.DataKeys[i].Value.ToString();//获取关键字段的值
        //自定义 update 语句并添加到 StringBuilder 对象中
        query.Append("UPDATE [tb_mrEmp] SET [au_lname] = '")
            .Append(value1).Append("' , [sex] = '")
            .Append(value2).Append("' , [state] = '")
            .Append(value3).Append("' , [phone] = '")
            .Append(value4).Append("' , [address] = '")
            .Append(value5).Append("' WHERE [ID] = '")
            .Append(value).Append("';\n");
    }
    conn.Open();                                        //打开数据库连接
    SqlCommand command = new SqlCommand(query.ToString(), conn);
                                                        //创建 SqlCommand 对象
    if (Convert.ToInt32(command.ExecuteNonQuery()) > 0)         //执行 sql 语句
    {
        Response.Write("<script>alert('一次修改数据成功！')</script>");//提示信息
    }
```

```
        else
        {
            Response.Write("<script>alert('一次修改数据失败！')</script>");
        }
        conn.Close();                                                    //关闭数据库连接
        bind();                                                          //关闭数据库连接
    }
```

学习手册 说明 由于篇幅有限，此处给出的只是部分代码。完整代码请读者参看图书附带的光盘。

12.2.2 在 GridView 控件中内嵌 DropDownList 设置考试试卷状态

通过使用 GridView 控件的模板列，可以指定在列中显示的控件。那么如何把从数据库中提取的数据和 GridView 控件模板列中的 DropDownList 控件进行绑定呢？下面通过实例演示如何实现在 GridView 控件中内嵌 DropDownList。

实例位置：光盘\MR\Instance\12\12.4

【例 12.4】 在实现在线考试试卷管理模块中，通过在 GridView 控件中添加一个模板列，并在该列中添加一个 DropDownList 控件来实现试卷状态的更改功能。执行程序，实例运行结果如图 12.5 所示。

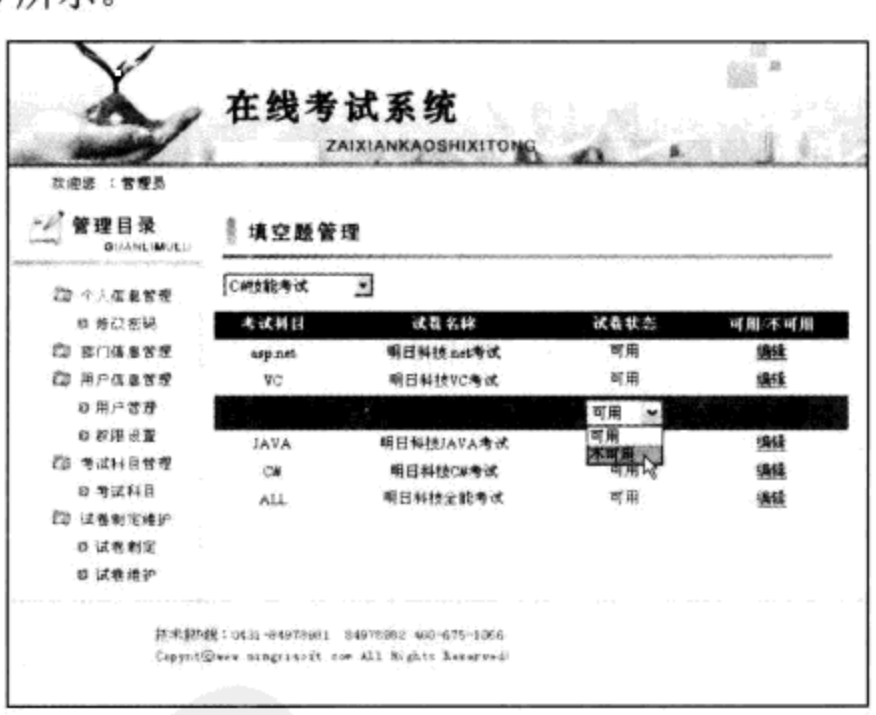


图 12.5 在 GridView 控件中应用 DropDownList 设置考试试卷状态效果图

自定义一个 InitData 方法，用于对 GridView 控件进行数据绑定。同时，通过 case 语句获取试卷状态，代码如下所示：

```
public void InitData()
{
    string strsql = "SELECT *,CASE PaperState WHEN 1 THEN '可用' ELSE '不可用'
    END AS state FROM Papermr";                                  //编写检索数据的 SQL 语句
    SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings
    ["conStr"].ToString());
    conn.Open();                                                 //打开数据库连接
```

```
    SqlDataAdapter myda = new SqlDataAdapter(strsql, conn);     //创建数据适配器
    DataSet myds = new DataSet();                               //创建数据集
    myda.Fill(myds);                                            //填充数据集
    if (myds.Tables[0].Rows.Count > 0)                          //当表格行数大于 0
    {
        Gvinfo.DataSource = myds;                               //设置控件数据源
        Gvinfo.DataBind();                                      //绑定数据
    }
}
```

学习手册 说明 CASE 语句计算一组布尔表达式以确定结果，试卷的状态在数据表中是布尔类型，但是在页面中要显示“可用”或“不可用”，所以，当 PaperState 字段为 true 时，使用 CASE 语句返回“可用”，否则返回“不可用”。此处获取 Count 属性，判断该属性是否大于 0，如果大于 0，说明存在数据，否则说明数据源中没有数据。

在 GridView 控件的 RowUpdating 事件中，选择试卷状态后，单击“更新”按钮进行更新操作。程序首先获取编辑行关键字段的值，然后编写 update 语句进行更新，最后还要调用 InitData 方法重新绑定数据，代码如下所示：

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
 //获取编辑行索引
    int ID=int.Parse(GridView1.DataKeys[e.RowIndex].Values[0].ToString());
    Byte PaperState = byte.Parse(((DropDownList)GridView1.Rows[e.RowIndex].
    FindControl ("ddlPaperState")). SelectedValue);
    string strsql = "UPDATE Papermr SET PaperState = @PaperState WHERE PaperID=
    @PaperID";
    SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings
    ["conStr"].ToString());
    conn.Open();                                                //打开数据库连接
    SqlCommand comm = new SqlCommand(strsql, conn);     //创建 SqlCommand 对象
    comm.Parameters.Add(new SqlParameter("@PaperID", SqlDbType.Int,4));
    comm.Parameters["@PaperID"].Value = ID;
    comm.Parameters.Add(new SqlParameter("@PaperState", SqlDbType.Bit, 1));
    comm.Parameters["@PaperState"].Value = PaperState;//设置参数值
    if (Convert.ToInt32(comm.ExecuteNonQuery()) > 0)   //如果返回值大于 0
    {
        Response.Write("<script language=javascript>alert('试卷状态修改成功!');
        location='Default.                aspx'</script>");      //说明修改成功
    }
    else                                                        //否则
    {
        Response.Write("<script language=javascript>alert('试卷状态修改失败!');
        location='Default.aspx'</script>");                     //修改失败
    }
    Gvinfo.EditIndex = -1;                                      //取消编辑操作
    //调用自定义方法 InitData()重新绑定 GridView 控件中信息
    InitData();
}
```

12.2.3　GridView 控件中高亮显示行数据

在开发网站的过程中，对数据库中的某行数据信息进行相关操作（如删除操作）有时需要高亮显示鼠标指定的行数据，以突出显示该行数据。那么该如何实现这个功能呢？下

面通过实例来具体演示实现的过程。

实例位置：光盘\MR\Instance\12\12.5

图 12.6 GridView 控件中高亮显示行数据效果图

【例 12.5】 在本实例中主要应用到了 GridView 控件的 RowDataBound 事件，该事件呈现在 GridView 控件之前，在该事件中设置鼠标进入和离开数据行时该行的背景颜色，从而实现 GridView 控件中高亮显示行数据，实例运行结果如图 12.6 所示。

首先自定义一个 BindData 方法，通过该方法将数据表中的数据绑定到 GridView 控件中，代码如下所示：

```
public void BindData()
{
    string strCon = ConfigurationManager.AppSettings["conStr"].ToString();
                                                          //创建数据库连接字符串
    string sqlstr = "select ID,name,sex,age from tb_user";     //声明 SQL 语句
    SqlConnection mycon = new SqlConnection(strCon);           //创建连接对象
    SqlDataAdapter myda = new SqlDataAdapter(sqlstr, strCon); //创建数据适配器
    DataSet myds = new DataSet();                              //创建数据集
    mycon.Open();                                              //打开数据库连接
    myda.Fill(myds);                                           //填充数据集
    Gvinfo.DataSource = myds;                                  //设置数据源
    Gvinfo.DataKeyNames = new string[] { " ID "};              //设置关键字段
    Gvinfo.DataBind();                                         //绑定数据
    mycon.Close();                                             //关闭连接
}
```

在 GridView 控件的 RowDataBound 事件中，设置鼠标进入和离开数据行时该行的背景颜色，从而实现 GridView 控件中高亮显示行数据，代码如下所示：

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)          //判断是否是数据行
    {
        e.Row.Attributes.Add("onMouseOver", "Color=this.style.backgroundColor;
        this.style.background-Color= 'lightBlue'");           //鼠标滑过时背景颜色
        e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=
        Color;");                                             //鼠标离开时背景颜色
    }
}
```

12.2.4 在 GridView 控件中排序数据

GridView 控件还提供了内置排序功能，无须任何编码，只要通过为列设置自定义 SortExpression 属性值，并使用 Sorting 和 Sorted 事件，进一步自定义 GridView 控件的排序功能。

实例位置：光盘\MR\Instance\12\12.6

【例 12.6】 本实例利用 GridView 控件的内置排序功能进行排序并显示数据。执行程序，实例运行结果如图 12.7 所示。

图 12.7　单击表头对数据进行排序

在 Default.aspx 页中添加一个 GridView 控件，在 Default.aspx 页的 Page_Load 事件中，用视图状态保存默认的排序表达式和排列顺序，然后对 GridView 控件进行数据绑定。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
 if (!IsPostBack)
 {
    ViewState["SortOrder"] = "stuID";                    //保存编号
    ViewState["OrderDire"] = "ASC";                      //保存排序规则
    GridViewBind();                                      //调用自定义的绑定方法
 }
}
```

该页的 Page_Load 事件中调用了自定义方法 GridViewBind()。该方法用来从数据库中取得要绑定的数据源，并设置数据视图的 Sort 属性，最后把该视图和 GridView 控件进行绑定。代码如下所示：

```
public void GridViewBind()
{
    SqlConnection sqlCon = new SqlConnection();    //实例化 SqlConnection 对象
    //实例化 SqlConnection 对象连接数据库的字符串
    sqlCon.ConnectionString = ConfigurationManager.AppSettings["conStr"].
    ToString();
    string SqlStr = "select * from tb_GoodsInfo"; //定义 SQL 语句
    //实例化 SqlDataAdapter 对象
    SqlDataAdapter da = new SqlDataAdapter(SqlStr, sqlCon);
    DataSet ds = new DataSet();                    //实例化数据集 DataSet
    da.Fill(ds);                                   //填充数据集
    DataView dv = ds.Tables[0].DefaultView;
    string sort = (string)ViewState["SortOrder"] + " " + (string)ViewState
    ["OrderDire"];
    dv.Sort = sort;                                //排序
    //绑定 DataList 控件
    Gvinfo.DataSource = dv;                        //设置数据源，用于填充控件中的项的值列表
    Gvinfo.DataBind();                             //将控件及其所有子控件绑定到指定的数据源
}
```

本例使用的是 SQL Server 2005 数据库，读者在运行程序时，请自行更改数据库的连接字符串，以防止运行出错。

在 GridView 控件的 Sorting 事件中，首先取得指定的表达式，然后判断是否是当前的排序方式。如果是，则改变当前的排序索引；如果不是，则设置新的排序表达式，并重新进行数据绑定，代码如下所示：

```
protected void Gvinfo _Sorting(object sender, GridViewSortEventArgs e)
{
    string sPage = e.SortExpression;                          //获取排序方式
    if (ViewState["SortOrder"].ToString() == sPage)           //判断是否是当前排序方式
    {
        if (ViewState["OrderDire"].ToString() == "Desc")      //如果排序方式是 Desc
            ViewState["OrderDire"] = "ASC";                   //更改为 ASC
        else                                                  //否则
            ViewState["OrderDire"] = "Desc";                  //更改为 Desc
    }
    else                                                      //如果不是当前排序方式
    {
        ViewState["SortOrder"] = e.SortExpression;            //设置新的排序表达式
    }
    GridViewBind();                                           //重新绑定数据
}
```

12.2.5 在 GridView 控件中实现全选和全不选功能

在 GridView 控件中添加一列 CheckBox 控件，并能通过在复选框的选择实现全选和取消全选的功能，该功能在批量删除时会经常用到。

实例位置：光盘\MR\Instance\12\12.7

【例 12.7】 利用 GridView 控件的模板列及 FindControl 方法实现全选和取消全选的功能。执行程序，实例运行结果如图 12.8 所示。

图 12.8 GridView 控件实现全选不全选功能

在 Default.aspx 页中添加一个 GridView 控件和一个 CheckBox 控件，CheckBox 控件的 AutoPostBack 属性设为 True。为 GridView 控件添加一列模板列，然后向模板列中添加 CheckBox 控件。GridView 控件的前台代码如下所示：

```
    <asp:GridView  ID="GridView1"  runat="server"  AutoGenerateColumns="False"
Width="328px">
     <Columns>
        <asp:TemplateField>
           <ItemTemplate>
              <asp:CheckBox ID="chkCheck" runat="server" />
           </ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="GoodsID" HeaderText="编号" />
        <asp:BoundField DataField="GoodsName" HeaderText="名称" />
        <asp:BoundField DataField="GoodsTypeName" HeaderText="类型" />
        <asp:BoundField DataField="GoodsPrice" HeaderText="价格" />
     </Columns>
    </asp:GridView>
```

改变“全选”复选框的选择状态时，将循环访问 GridView 控件中的每一项，并通过 FindControl 方法搜索 TemplateField 模板列中 ID 为 chkCheck 的 CheckBox 控件，并建立该控件的引用，实现“全选/不全选”功能，代码如下所示：

```
protected void chkAll_CheckedChanged(object sender, EventArgs e)
{
    for (int i = 0; i <= GridView1.Rows.Count - 1; i++)
    {
        //建立模板列中 CheckBox 控件的引用
        CheckBox chk = (CheckBox)GridView1.Rows[i].FindControl("chkCheck");
        if (chkAll.Checked == true)                    //如果选中“全选”复选框
        {
            chk.Checked = true;                        //将每一行前边的复选框都选中
        }
        else                                           //如果没有选择“全选”复选框
        {
            chk.Checked = false;                       //取消每一行前边复选框的选中状态
        }
    }
}
```

CheckBox 控件的 AutoPostBack 属性设为 True，否则当更改控件的选择状态时不能触发 CheckedChanged 事件。

12.3　灵活访问数据俊杰：DataList 控件

专题讲座：光盘\MR\Video\12\ DataList 控件的使用.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 DataList 控件强大功能的实现。

DataList 控件是一个常用的数据绑定控件，该控件能够以某种设定好的模板格式循环显示多条数据。这种模板格式是可以根据需要进行自定义的，比较于 GridView 控件，虽

然 GridView 控件功能非常强大，但它始终只能以表格的形式显示数据，而使用 DataList 控件则灵活性非常强，其本身就是一个富有弹性的控件，本节将对 DataList 控件进行详细讲解。

12.3.1 关于 DataList 控件

DataList 控件可以使用模板与定义样式来显示数据，并进行数据的选择、删除以及编辑。DataList 控件最大的特点是一定要通过模板来定义数据的显示格式。如果要设计出美观的界面，就需要花费一番心思。正因为如此，DataList 控件显示数据时却更具灵活性，开发人员个人发挥的空间也比较大，DataList 控件支持的模板说明如下。

- ☑ **AlternatingItemTemplate**：为 DataList 中的交替项提供内容和布局。如果未定义，则使用 ItemTemplate。
- ☑ **EditItemTemplate**：为 DataList 中当前编辑项提供内容和布局。如果未定义，则使用 ItemTemplate。
- ☑ **FooterTemplate**：为 DataList 的脚注部分提供内容和布局。如果未定义，将不显示脚注部分。
- ☑ **HeaderTemplate**：如果已定义，则为 DataList 的页眉节提供内容和布局。如果未定义，将不显示页眉节。
- ☑ **ItemTemplate**：为 DataList 中的项提供内容和布局所要求的模板。
- ☑ **SelectedItemTemplate**：为 DataList 中当前选定项提供内容和布局。如果未定义，则使用 ItemTemplate。
- ☑ **SeparatorTemplate**：为 DataList 中各项之间的分隔符提供内容和布局。如果未定义，将不显示分隔符。

12.3.2 分页绑定 DataList 控件中的数据（实现页面跳转功能）

DataList 控件绑定数据源的方法与 GridView 控件基本相似，但要将所绑定数据源的数据显示出来，就需要通过设计 DataList 控件的模板来完成。

实例位置：光盘\MR\Instance\12\12.8

【例 12.8】 使用 PagedDataSource 类实现 DataList 控件的分页功能，并实现页面的跳转功能。执行程序，实例运行结果如图 12.9 所示。

单击 DataList 控件右上方的“ ”按钮，在弹出的快捷菜单中的选择“编辑模板”选项。打开“DataList 任务—模板编辑模式”，在“显示”下拉列表框中选择“ItemTemplate”选项，然后向模板中添加表格、Image 控件并输入相关信息，如图 12.10 所示。

在 ItemTemplate 模板的 HTML 代码中，通过 Eval 绑定数据，代码如下所示：

```
<ItemTemplate>
    ……
 <asp:Image ID="Image4" runat="server" ImageUrl="~/images/ico2.gif" />
    空间主人:<a><%#Eval("PerHomeUser") %></a></td>
    ……
```

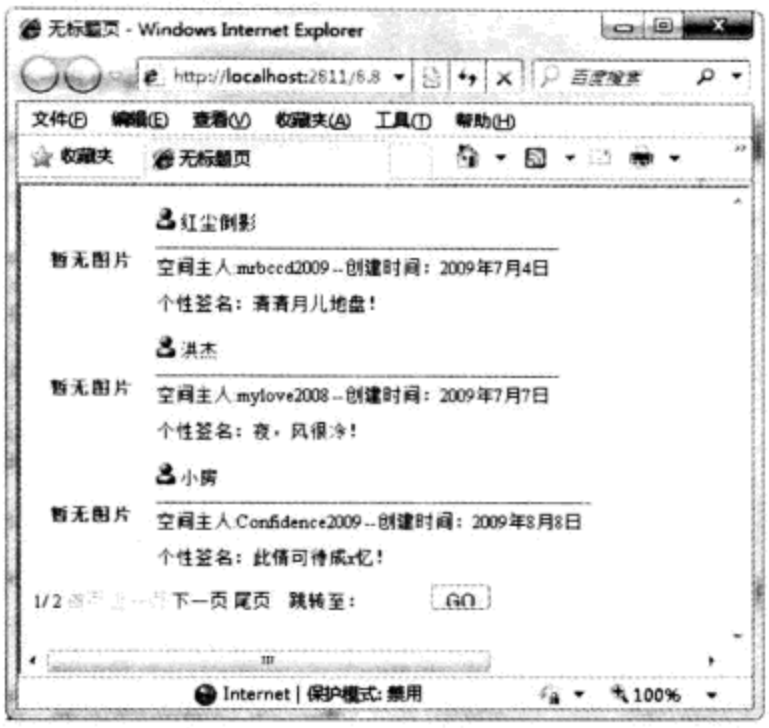


图 12.9　分页绑定 DataList 控件中的数据，并实现指定页面的跳转

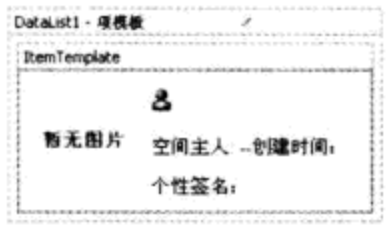


图 12.10　模板中的内容

```
        --创建时间：<a><%#Eval("PerHomeTime","{0:D}") %></a></td>
    ......
        个性签名：<a ><%#Eval("PerHomeSign").ToString().Length > 12 ? Eval
        ("PerHomeSign"). ToString().Substring(0, 12) + "..." : Eval("PerHomeSign")%>
        </a></td>
</ItemTemplate>
```

使用同样方式在“编辑模板”中选择 FootTemplate 模板，在该模板中添加两个 Label 控件和 4 个 LinkButton 控件。Label 控件的 ID 属性分别为 labPageCount 和 labCurrentPage，主要用来显示总页数和当前页；LinkButton 控件的 ID 属性分别为 lnkbtnFirst、lnkbtnFront、lnkbtnNext、lnkbtnLast，分别用来显示首页、上一页、下一页、尾页，如图 12.11 所示。

图 12.11　FootTemplate 模板中的内容

在后台代码中，首先定义两个全局变量对象，分别是分页数据源对象和数据库连接对象，声明代码如下所示：

```
//创建一个分页数据源的对象且一定要声明为静态
protected static PagedDataSource pds = new PagedDataSource();
public SqlConnection conn = new SqlConnection(ConfigurationManager.AppSettings
["conStr"]. ToString());
```

PagedDataSource 数据源分页对象 pds 一定要设置为 static 即静态对象变量，否则它将不能保存当前页面的索引值。

自定义一个 BindDataList 方法，在该方法中主要设置控件分页及数据绑定。其参数用于获取页面中的索引值，代码如下所示：

```
private void BindDataList(int currentpage)
{
```

```
        pds.AllowPaging = true;                                    //允许分页
        pds.PageSize = 3;                                          //每页显示 3 条数据
        pds.CurrentPageIndex = currentpage;                        //当前页为传入的一个 int 型值
        string strSql = "SELECT * FROM PerHomeDetail";      //定义一条 SQL 语句
        conn.Open();                                               //打开数据库连接
        SqlDataAdapter sda = new SqlDataAdapter(strSql,conn);
        DataSet ds = new DataSet();
        sda.Fill(ds);                                              //把执行得到的数据放在数据集中
        pds.DataSource = ds.Tables[0].DefaultView;    //把数据集中的数据放入分页数据源中
        dlinfo.DataSource = pds;                                   //绑定 Datalist
        dlinfo.DataBind();
        conn.Close();
    }
```


PagedDataSource 类封装数据绑定控件（如 DataGrid、GridView、DetailsView 和 FormView）的与分页相关的属性，以允许该控件执行分页操作。

在 DataList 控件的 ItemCommand 事件中主要对用户单击“第一页”、“上一页”、“下一页”和“尾页”及在文本框内输入页数时跳转到指定页时发生的事件进行处理，代码如下所示：

```
protected void dlinfo _ItemCommand(object source, DataListCommandEventArgs e)
{
    switch (e.CommandName)
    {
        //以下 5 个为 捕获用户单击 上一页 下一页等时发生的事件
        case "first":                                              //第一页
            pds.CurrentPageIndex = 0;                              //当前页为 0
            BindDataList(pds.CurrentPageIndex);                    //重新绑定
            break;
        case "pre":                    //上一页
            pds.CurrentPageIndex = pds.CurrentPageIndex - 1;       //当前页减 1
            BindDataList(pds.CurrentPageIndex);                    //重新绑定
            break;
        case "next":                   //下一页
            pds.CurrentPageIndex = pds.CurrentPageIndex + 1;       //总页数加 1
            BindDataList(pds.CurrentPageIndex);                    //重新绑定
            break;
        case "last":                   //最后一页
            pds.CurrentPageIndex = pds.PageCount - 1;              //总页数减 1
            BindDataList(pds.CurrentPageIndex);                    //重新绑定
            break;
        case "search":                 //页面跳转页
            if (e.Item.ItemType == ListItemType.Footer)
            {
                int PageCount = int.Parse(pds.PageCount.ToString());
                TextBox txtPage = e.Item.FindControl("txtPage") as TextBox;
                int MyPageNum = 0;
                if (!txtPage.Text.Equals(""))
                    MyPageNum = Convert.ToInt32(txtPage.Text.ToString());
                if (MyPageNum <= 0 || MyPageNum > PageCount)
                    Response.Write("<script>alert('请输入页数并确定没有超出总页数！')
                    </script>");
                else
                    BindDataList(MyPageNum - 1);
```

```
            }
            break;
        }
    }
```

DataList 控件的 ItemDataBound 事件主要是对在 DataList 控件的 FootTemplate 模板中添加的 Label 和 LinkButton 控件进行操作，代码如下所示：

```
protected void dlinfo _ItemDataBound(object sender, DataListItemEventArgs e)
{
    if (e.Item.ItemType == ListItemType.Footer)
    {
        //以下六个为脚模板中的控件,并创建变量
        Label CurrentPage = e.Item.FindControl("labCurrentPage") as Label;
        Label PageCount = e.Item.FindControl("labPageCount") as Label;
        LinkButton FirstPage = e.Item.FindControl("lnkbtnFirst") as LinkButton;
        LinkButton PrePage = e.Item.FindControl("lnkbtnFront") as LinkButton;
        LinkButton NextPage = e.Item.FindControl("lnkbtnNext") as LinkButton;
        LinkButton LastPage = e.Item.FindControl("lnkbtnLast") as LinkButton;
        CurrentPage.Text = (pds.CurrentPageIndex + 1).ToString();//绑定显示当前页
        PageCount.Text = pds.PageCount.ToString();                    //绑定显示总页数
        if (pds.IsFirstPage)              //如果是第一页,则首页和上一页不能用
        {
            FirstPage.Enabled = false;
            PrePage.Enabled = false;
        }
        if (pds.IsLastPage)               //如果是最后一页"下一页"和"尾页"按钮不能用
        {
            NextPage.Enabled = false;
            LastPage.Enabled = false;
        }
    }
}
```

当项被数据绑定到 DataList 控件后，将引发 ItemDataBound 事件。此事件提供了在客户端显示数据项之前访问该数据项的最后机会。

12.3.3　使用 DataList 删除数据（支持批量删除）

DataList 控件的功能非常强大，而且运行效率比较高。可能读者会将 DataList 控件和 GridView 控件进行比较，比较这两个控件到底哪个更加强大？其实，每个控件都有自己的优点和缺点，DataList 控件的显著优点是效率比较高，缺点则是开发起来没有 GridView 控件迅速。具体选择哪个控件要根据实现的功能进行选择。

实例位置：光盘\MR\Instance\12\12.9

【例 12.9】 利用 DataList 控件来实现单条数据的删除和批量数据删除的操作。执行程序，实例运行结果如图 12.12 所示。

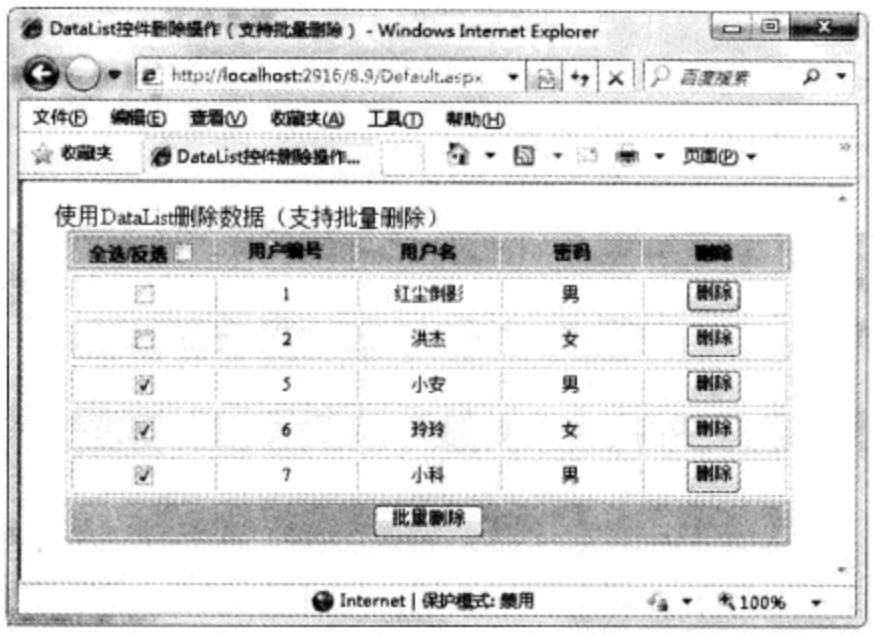


图 12.12　使用 DataList 删除数据（支持批量删除）

新建一个网站，默认主页为 Default.aspx 并添加 1 个 DataList 控件。单击 DataList 控件右上方的“▷”按钮，在弹出的快捷菜单中选择“编辑模板”选项。打开“DataList 任务—模板编辑模式”，在“显示”下拉列表框中选择“ItemTemplate”选项，如图 12.13 所示。

图 12.13　DataList 编辑模板列中设计的内容

学习手册 注意 在 ItemTemplate 模板列中添加执行单条数据删除操作的 Button 按钮，将其 CommandName 属性设置为 delete。用同样方式在 DataList 控件的 FootTemplate 模板列中添加一个 Button 控件用于执行批量删除操作，同时要设置其 CommandName 属性为 pldelete。

在后台代码的 Page_Load 事件中，调用了一个自定义方法 BindDataList，主要是用来在页面初始化时绑定 DataList 控件中的数据，代码如下所示：

```
    //得到 Web.config 中的连接放在变量中
    SqlConnection  Conn  =  new  SqlConnection(ConfigurationManager.AppSettings
["conStr"].ToString());
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            BindDataList();                                    //绑定数据
        }
    }
```

上述事件代码中调用了自定义方法 BindDataList，该方法主要是将从数据库中查询的信息绑定到 DataList 控件中，代码如下所示：

```
    public void BindDataList()
    {
        string strSql = "SELECT * FROM tb_User";        //定义一条 SQL 语句
```

```
        SqlDataAdapter sda = new SqlDataAdapter(strSql, Conn);
        DataSet ds = new DataSet();
        sda.Fill(ds);                                    //把执行得到的数据放在数据集中
        dlinfo.DataSource = ds;                          //绑定 Datalist
        dlinfo.DataBind();
    }
```

执行单条数据或批量数据删除操作时，触发 DataList 控件的 ItemCommand 事件，在该事件中根据 CommandName 属性设定的值来分别执行单条数据的删除和批量数据的删除操作，代码如下所示：

```
protected void dlinfo _ItemCommand(object source, DataListCommandEventArgs e)
{
    switch(e.CommandName)
    {
        //单条数据删除操作
        case "delete":
        int id=int.Parse(DataList1.DataKeys[e.Item.ItemIndex].ToString());
                                                         //取得当前 DataList 控件列
        string strsql="delete from tb_User where UserID='"+id+"'";
        if(Conn.State.Equals(ConnectionState.Closed))
        { Conn.Open();}                                  //打开数据库连接
        SqlCommand cmd=new SqlCommand(strsql,Conn);
        if(Convert.ToInt32(cmd.ExecuteNonQuery())>0)
        {
            Response.Write("<script>alert('删除成功！')</script>");
            BindDataList();                              //重新绑定控件数据
        }
        else
        {
            Response.Write("<script>alert('删除失败，请查找原因！')</script>");
        }
        Conn.Close();                                    //关闭连接
        break;
        //批量删除操作
        case "pldelete":
        Conn.Open();                                     //打开数据库连接
        DataListItemCollection dlic = DataList1.Items;
                                                         //创建一个 DataList 列表项集合对象
        //执行一个循环,删除所有用户选中的信息
        for (int i = 0; i < dlic.Count; i++)
        {
          if (dlic[i].ItemType == ListItemType.AlternatingItem || dlic[i].
          ItemType == ListItemType.Item)
          {
            CheckBox cbox = (CheckBox)dlic[i].FindControl("CheckBox2");
            if (cbox.Checked)
            {
              int id_pldelete = int.Parse(DataList1.DataKeys[dlic[i].ItemIndex].
              ToString());
              SqlCommand cmd_pldel = new SqlCommand("delete from tb_User where
              UserID=" + id_pldelete, Conn);
               cmd_pldel.ExecuteNonQuery();              //执行批量删除操作
            }
          }
    }
```

```
            Conn.Close();
            BindDataList();
            break;
        }
    }
```

学习手册 说明 ItemCommand 事件在单击 DataList 控件中的任一按钮时引发，并且在使用具有带自定义 CommandName 值的按钮控件时会用到 ItemCommand 事件。

12.3.4 在 DataList 控件中实现数据编辑操作

DataList 控件也可以像 GridView 控件一样，为特定项进行编辑操作。在 DataList 控件中是使用 EditItemTemplate 模板实现这一功能的。

实例位置：光盘\MR\Instance\12\12.10

【例 12.10】 使用 EditItemTemplate 模板对 DataList 控件中的数据项进行编辑。执行程序，实例运行结果如图 12.14 所示。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 页中添加一个 DataList 控件。打开 DataList 控件的项模板编辑模式。在 ItemTemplate 模板中添加一个 Label 控件和一个 Button 控件；在 EditItemTemplate 模板中添加两个 Button 控件和一个 Label 控件和 4 个 TextBox 控件。DataList 控件及各模板内控件的设计如图 12.15 所示。

图 12.14 在 DataList 控件中对数据进行编辑操作

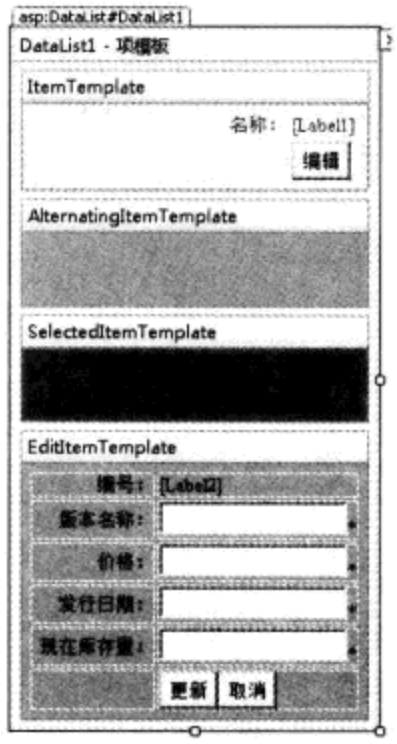


图 12.15 在 DataList 控件的 ItemTemplate 和 EditTemplate 模板中设计的内容

在前台页面 HTML 中，需要注意控件数据的绑定，代码如下所示：

```
<asp:DataList ID="DataList1" runat="server"……>
        <ItemTemplate>
    ……
        名称： <asp:Label ID="Label1" runat="server" Text='<%# Eval("BccdName")
```

```
            %>'></asp:Label>
            <asp:Button ID="btnEdit" runat="server" CommandName="edit" Text="编辑" />
        ……
            </ItemTemplate>
            <EditItemTemplate>
            编号: <asp:Label ID="Label2" runat="server" Text='<%# Eval("BccdID") %>'>
            </asp:Label>
            版本名称:  <asp:TextBox ID="tbBccdName" runat="server" Text='<%# Bind
            ("BccdName") %>'></asp:TextBox>
             ……
        价格: <asp:TextBox ID="tbBccdPrice" runat="server" Text='<%# Bind
        ("BccdPrice") %>'></asp: TextBox>
        ……
            发行日期: <asp:TextBox ID="tbDate" runat="server" Text='<%# Bind
            ("BccdSaleDate","{0:d}") %>'></asp:TextBox>
            ……
            现在库存量: <asp:TextBox ID="tbBccdInStock" runat="server" Text='<%# Bind
            ("BccdInStock") %>'></asp:TextBox>
            ……
            </EditItemTemplate>
    </asp:DataList>
```

当用户单击“编辑”按钮时，将触发 DataList 控件的 EditCommand 事件。在该事件的处理程序中，将用户选中的项设置为编辑模式，代码如下所示：

```
protected void dlinfo _EditCommand(object source, DataListCommandEventArgs e)
{
    //设置 DataList1 控件的编辑项的索引为选择的当前索引
    dlinfo.EditItemIndex = e.Item.ItemIndex;
    BindDataList();                                    //绑定数据
}
```

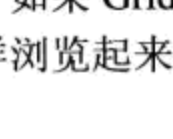
在编辑模式下，当用户单击“更新”按钮时，将触发 DataList 控件的 UpdataCommand 事件。在该事件的处理程序中，将用户的更改更新至数据库，并取消编辑状态。代码如下所示：

```
protected void dlinfo _UpdateCommand(object source, DataListCommandEventArgs e)
{
    //取得编辑行的关键字段的值
    int id =int.Parse(DataList1.DataKeys[e.Item.ItemIndex].ToString());
    string bccdName = ((TextBox)e.Item.FindControl("tbBccdName")).Text;
    string bccdPrice = ((TextBox)e.Item.FindControl("tbBccdPrice")).Text;
    string bccdSaleDate = ((TextBox)e.Item.FindControl("tbDate")).Text;
    string bccdBccdInStock = ((TextBox)e.Item.FindControl("tbBccdInStock")).
    Text;
    string strsql = "update mrbccd set BccdName='" + bccdName + "',BccdPrice='"
    + bccdPrice + "', BccdSaleDate='" + bccdSaleDate + "',BccdInStock='" +
    bccdBccdInStock + "'where BccdID=" + id + "";
    Conn.Open();//打开数据库
    SqlCommand cmd = new SqlCommand(strsql, Conn);
    try
    {
        cmd.ExecuteNonQuery();
        //取消编辑
        dlinfo.EditItemIndex = -1;
        BindDataList();
    }
    catch (Exception err)
```

```
        {
            //输出异常信息
            Response.Write(err.ToString());
        }
        finally
        {
            Conn.Close();
        }
    }
```

当用户单击“取消”按钮时，将触发 DataList 控件的 CancelCommand 事件。在该事件的处理程序中，取消处于编辑状态的项，并重新绑定数据。代码如下所示：

```
protected void dlinfo _CancelCommand(object source, DataListCommandEventArgs e)
{
    //设置 DataList1 控件的编辑项的索引为－1，即取消编辑
    dlinfo.EditItemIndex = -1;
    BindDataList();
}
```



DataList 控件在绑定数据时，应先将 DataKeyField 属性设置为数据表的主键。在程序中，可以由 DataKeys 属性获取每行数据的主键值。

12.4 实 战 练 习

12.4.1 GridView 控件实现用“…”代替超长字符

➤➤➤题目描述

GridView 控件实现用“…”代替超长字符，作者根据这个功能做了一个小程序，当单元格里的数据超过指定的长度时，使用“…”代替，运行效果如图 12.16 所示。

图 12.16 用“…”代替超长字符

➤➤➤技术指导

如果 GridView 控件单元格中的数据过多，则会影响整个 GridView 控件的布局。显然，这样浏览起来不是很美观。此时，就应该用“…”代替超长字符。本例主要是创建一个自

定义方法用于判断单元格中的数据是否过长，如果过长则截取后用“…”代替多余的部分。关键技术是我们自定义的 SubStr 方法，其参数分别是 sString（要截取的字符串）和 nLeng（截取的长度）。该方法用于从一个字符串中截取指定长度的字符串，代码如下所示：

```
public string SubStr(string sString, int nLeng)   //自定义方法用于截取字符串
{
    if (sString.Length <= nLeng)                   //判断字符串长度是否小于等于规定长度
    {
        return sString;                            //如果小于或者等于则不用截取
    }
    string sNewStr = sString.Substring(0, nLeng);
                                                   //如果大于规定长度则截取规定长度的字符串
    sNewStr = sNewStr + "...";                     //截取之后加上“…”
    return sNewStr;                                //返回截取后的字符串
}
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/12.4.1**

12.4.2　GridView 控件加入自动求和及平均值功能

➤➤➤题目描述

GridView 控件加入自动求和及平均值功能，作者根据这个要求做了一个小程序，可以自动求某一列的和及平均值，运行结果如图 12.17 所示。

图 12.17　GridView 控件加入自动求和及平均值功能

➤➤➤技术指导

在实际生活中，有很多数据需要计算总和与平均值。例如，学生的考试成绩、员工的工资等。GridView 控件的功能特别强大，我们完全可以在显示数据后，加入自动求和及平均值的功能，主要是在控件的 RowDataBound 事件中添加脚注行，在脚注行里添加显示总价格和平均价格的单元格。在 GridView 控件的 RowDataBound 事件中，首先获取指定列中价格的总和。然后判断数据行的类型是否是 Footer（脚注行），如果是 Footer 则在 Footer

中添加总价格和平均价格等信息，代码如下所示：

```
private double sum = 0;                                          //取指定列的数据和
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowIndex >= 0)                                     //如果数据行索引大于0
    {
        sum += Convert.ToDouble(e.Row.Cells[2].Text);            //计算价格的总和
    }
    else if (e.Row.RowType == DataControlRowType.Footer)         //如果是脚注行
    {
        e.Row.Cells[0].Text = "总价格：";                         //添加标题
        e.Row.Cells[1].Text = sum.ToString();                    //添加价格总和
        e.Row.Cells[2].Text = "平均价格：";                       //添加标题
        e.Row.Cells[3].Text = ((int)(sum / GridView1.Rows.Count)).ToString();
                                                                 //添加平均值
    }
}
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/12.4.2**

12.5 本章小结

ASP.NET 3.5 中提供了多种数据控件，用于在 Web 页中显示数据，这些控件具有丰富的功能，本章主要学习了对 GridView 控件和 DataList 控件的使用。通过实例对这两个控件有了进一步的了解。这两个控件在以后制作网页的过程中会经常使用，所以本章内容要熟练掌握。

第 3 篇

高级篇

书山有路勤为径
学海无涯苦作舟

本篇主要内容：

- 第 13 章　创建自己的 Web 用户控件
- 第 14 章　ASP.NET 验证控件
- 第 15 章　利用 GDI+绘制 Web 图形图像
- 第 16 章　AJAX 无刷新技术
- 第 17 章　调试与错误处理

本篇学习流程：

第 13 章

创建自己的 Web 用户控件

（名师课堂：35 分钟）

提高应用程序中的代码重用性，是网站开发过程中力求的原则。用户控件基本的应用就是把网页中经常用到且使用频率较高的程序封装到一个模块中，以便在其他页面中使用，以此提高代码的重用性和程序开发的效率。用户控件的应用始终融汇着一个高层的设计思想，即“模块化设计，模块化应用”的原则。本章将通过介绍如何创建及使用和设置用户控件进一步掌握这项开发网站的实用技术。通过本章的学习，读者可以掌握以下知识：

- 了解用户控件与普通 Web 页的区别
- 了解用户控件的一些优点
- 学习和掌握普通用户控件的创建
- 学习如何将 Web 用户控件添加至网页
- 掌握如何使用用户控件制作站点导航条
- 了解如何在用户控件中添加用户控件
- 掌握访问用户控件属性的方法
- 掌握访问用户控件中服务器控件的方法
- 学习将 ASP.NET 页转换为用户控件的方法

13.1　Web 用户控件的概述

通过对本节的学习，读者能够了解什么是 Web 用户控件、Web 用户控件与普通 Web 页的区别以及 Web 用户控件的优点，所以请读者认真阅读本节内容。

13.1.1　什么是 Web 用户控件

Web 用户控件是一种服务器控件，它与 ASP.NET 页面有着同样的“所见即所得”的特点和声明性样式，并以.ascx 为扩展名存储为文本文件。Web 用户控件与完整的 ASP.NET 网页（即.aspx 文件）非常相似，同时具有自己的用户界面页和代码。开发人员可以采取与创建 ASP.NET 页相似的方式创建 Web 用户控件，然后向其中添加所需的标记和子控件。Web 用户控件可以像页面一样包含对其内容进行操作（包括执行数据绑定等任务）的代码。

此外，用户控件还拥有自己的对象模型的类，页面开发人员可以对其编程，它比服务器端包含文件提供了更多的功能，为创建具有复杂用户界面元素的控件带来了极大方便。同时，编写 Web 用户控件的语言可以与包含它的页面语言有所不同，这意味着使用公共语言运行库支持的任何语言编写的 Web 用户控件可以在同一个页面中使用。

13.1.2　用户控件与普通 Web 页的比较

用户控件几乎与.aspx 文件相同，但是仍存在以下不同之处。

- ☑ 用户控件的文件扩展名为.ascx。
- ☑ 用户控件中没有@Page 指令，而是包含@Control 指令，该指令对配置及其他属性进行定义。
- ☑ 用户控件不能作为独立文件运行，而必须像处理任何控件一样，将它们添加到 ASP.NET 页中。
- ☑ 用户控件中没有 html、body 或 form 元素。

可以在 Web 用户控件上使用与在 ASP.NET 网页上所用相同的 HTML 元素（html、body 或 form 元素除外）和 Web 控件。例如，如果要创建一个用作工具栏的 Web 用户控件，可以将一系列 Button 服务器控件添加到该控件上，并创建这些按钮的事件处理程序。

13.1.3　用户控件的优点

通过 Web 用户控件，可以很好地重用公共用户界面（UI）功能，从而提升程序开发效率。用户控件提供了一个面向对象的编程模型，在一定程度上取代了服务器端文件包含（<!--#include-- >）指令，并且提供的功能比服务器端包含文件提供的功能多。使用用户控件的主要优点如下。

- ☑ 可以将常用的内容或者控件以及控件的运行程序逻辑设计为用户控件，然后便可以在多个网页中重复使用该用户控件，从而省略许多重复性的工作。例如网页上的导航栏，几乎每个页都需要相同的导航栏，这时可以将其设计为一个用户控件，在多个页中使用。
- ☑ 如果网页内容需要改变时，只需修改用户控件中的内容，其他添加使用该用户控件的网页会自动随之改变，因此网页的设计以及维护变得简单易行。

总之，对于页面上复用的元素，如导航条、站内搜索、用户注册和登录控件等，都可以将其代码封装到 Web 用户控件中，以此来减少每个页面上的代码量。此外，使用 Web 用户控件的高速缓存功能，高速缓存这些经常浏览的页面，可以提高页面的性能。

13.2 创建及使用 Web 用户控件

专题讲座：光盘\MR\Video\13\创建及使用 Web 用户控件.exe

➤➤➤视频速递：全面解读如何创建及使用 Web 用户控件。

ASP.NET 用户控件的创建方式很灵活，可以直接在项目中添加 Web 用户控件，也可以通过创建以.ascx 为扩展名的文本文件进行创建，还可以通过修改 Web 窗体页（.aspx 文件），将其转换为 ASP.NET 用户控件。本节主要讲解如何创建 Web 用户控件、如何将 Web 用户控件添加到网页以及在用户控件中添加用户控件等内容。

13.2.1 创建 Web 用户控件

虽然创建 Web 用户控件的方法有很多种，但是比较之后还是通过 Visual Studio 2008 开发环境创建 Web 用户控件最为简单，这是最常用的一种方法。所以，下面演示如何通过 Visual Studio 2008 创建一个用户控件。

（1）打开解决方案资源管理器，鼠标右键单击项目名称，在弹出的快捷菜单中选择“添加新项”选项，将会弹出如图 13.1 所示的“添加新项”对话框。在该对话框中，选择“Web 用户控件”项，并为其命名，然后单击“添加”按钮即可将 Web 用户控件添加到项目中。

（2）双击打开已创建好的 Web 用户控件（用户控件的文件扩展名为.ascx），在.ascx 文件中可以直接添加各种服务器控件以及静态文本、图片等，与向普通网页中添加控件是一样的。Web 用户控件成功添加后如图 13.2 所示。

（3）双击页面上的任何位置，或者直接按 F7 键，可以将视图切换到后台代码文件，程序开发人员可以直接在文件中编写代码。

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

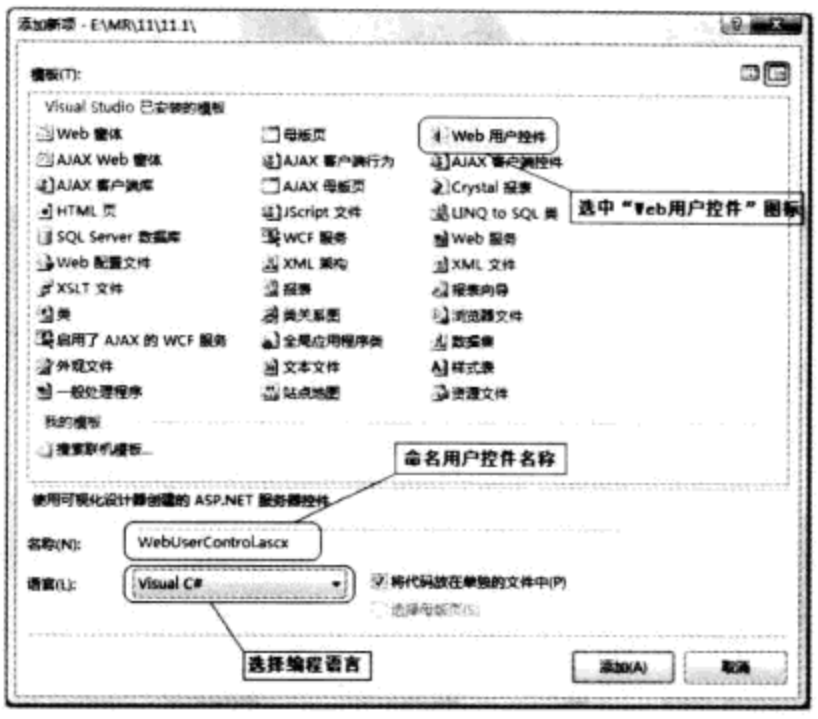


图 13.1　添加新项对话框

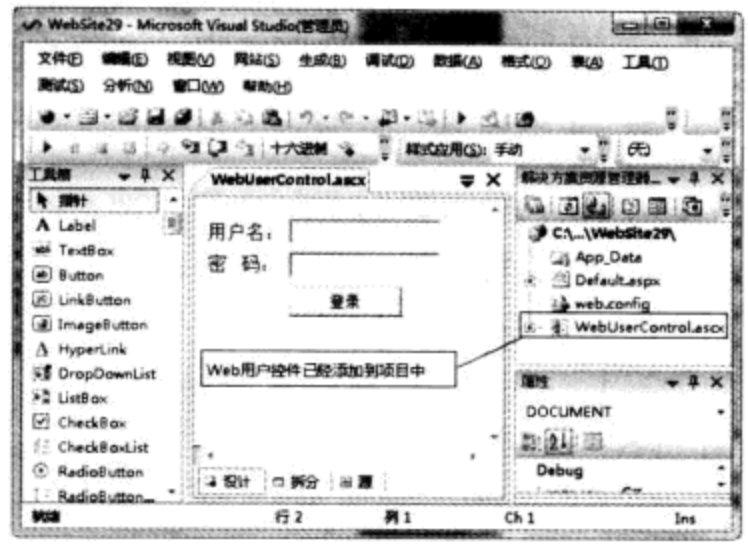


图 13.2　项目中成功添加 Web 用户控件

13.2.2　将 Web 用户控件添加至网页

如果已经设计好了 Web 用户控件，可以将其添加到一个或者多个网页中。在同一个网页中也可以重复使用多次，各个用户控件会以不同 ID 来标识。在“设计”视图下，将用户控件以拖放的方式直接添加到网页上即可，其操作与将内置控件从工具箱中拖放到网页上一样。在网页中添加用户控件的步骤如下。

（1）在解决方案资源管理器中，用鼠标单击要添加至网页的用户控件。

（2）按住鼠标左键，将用户控件拖曳到网页上，然后释放鼠标左键即可，如图 13.3 所示。

（3）在已添加的用户控件上，单击鼠标右键，在弹出的快捷菜单中选择“属性”项，打开用户控件的属性窗口，如图 13.4 所示，用户可以在属性窗口中修改用户控件的属性。

图 13.3　将 Web 用户控件添加至网页

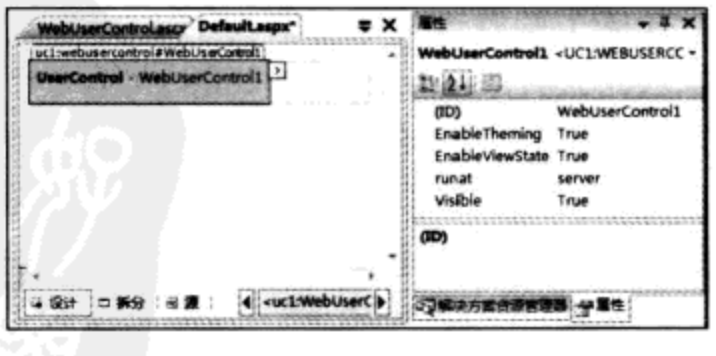


图 13.4　用户控件的属性窗口

13.2.3 情景应用 1：在用户控件中添加用户控件

用户控件的使用可以减少开发人员的工作量，在设计用户控件时，可以将已创建好的用户控件添加到某个用户控件中。通常称这种用户控件为复合式用户控件。

实例位置：光盘\MR\Instance\13\13.1

【例 13.1】 本实例是一个开发博客网站的实例，在开发过程中应用到了在用户控件中添加用户控件的技巧。首先对实例中创建的复合式用户控件的功能简单分析如下。

（1）Header.ascx 控件

Header.ascx 控件用来实现界面头部的设计，如图 13.5 所示。在这个界面的头部中主要实现用户登录、用户注册和两个超链接功能。

（2）VisitorNav.ascx 控件

VisitorNav.ascx 控件用来实现此博客网站的信息导航，如图 13.6 所示。通过使用 4 个 HypeLink 控件，分别显示“首页”、“博客文章”、“博客注册”和“帮助”，单击每个 HypeLink 控件，可链接到相应的 URL 地址。

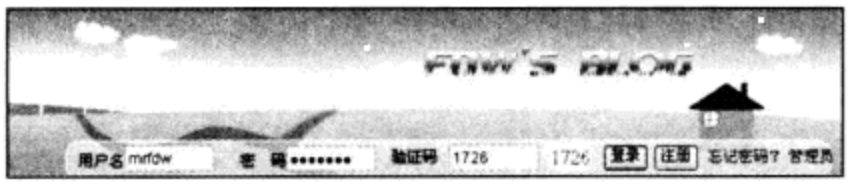


图 13.5 博客网站头部的用户控件

图 13.6 博客网站中的导航控件

同将 Web 用户控件添加至网页的方法一样，以拖曳的方式将 VisitorNav.ascx 控件拖曳至 Header.ascx 用户控件的适当位置。通过以上操作设计的复合式控件如图 13.7 所示。

除了上述将网站页面头做成用户控件（属于复合式用户控件）外，也可以应用普通的用户控件如将日历控件（Calendar）做成普通的用户控件，如图 13.8 所示。

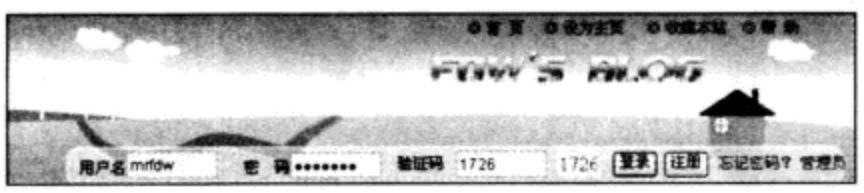


图 13.7 复合式用户控件

图 13.8 制作普通用户控件

根据以上用户控件的分析，用户控件组合后的实例运行效果如图 13.9 所示。

实例主要开发步骤说明如下。

（1）打开 Visual Studio 2008 新建一个网站，默认主页为 BlogIndex.aspx。

（2）在该网站的解决方案下，右键单击网站名称，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，选择“Web 用户控件”项，分别创建网站中所应用的用户控件，并创建如上所述的复合式控件和普通用户控件。

（3）新建一个 Web 页，并将其命名为 BlogIndex.aspx，此页主要用来设计这个博客网站的首页。将以上创建完的用户控件和一些复合式控件分别从应用程序中拖曳到这个 Web 页中，并作相应的合理布局。

图 13.9　应用用户控件制作的博客首页界面

完成以上主要实施步骤，便基本上完成了一个博客网站的首页界面的制作。

学习手册 技巧　在开发博客网站时，可以将博客网站的导航条设计为用户控件，由于浏览博客网站前后的用户身份有两种，即访客（还没有登录的用户）和博客（已登录的用户），对于不同身份的用户都需要在导航条中添加导航按钮，除此之外，访客用户还需要显示登录模块以方便其登录，而对于博客用户可以显示欢迎模块，因此在设计博客网站导航条时，可以设计为 3 个用户控件，然后将访客的登录模块用户控件（如 Login.ascx）和博客的欢迎模块用户控件（如 Welcome.ascx）添加到导航按钮用户控件（如 menu.ascx）中，这样可以根据用户的身份来显示不同的导航控件。

13.2.4　情景应用 2：使用 Web 用户控件制作博客导航条

对于页面上常用的内容或者控件，可以将它们设计成用户控件，然后便可以在多个网页中重复使用该用户控件，从而省去了编写许多重复的代码。例如网页的横幅，对于很多网站来说，几乎所有网页都有相同的网页横幅，可以将横幅设计成用户控件，然后添加到所有需要的网页中即可。

实例位置：光盘\MR\Instance\13\13.2

【例 13.2】 本实例主要利用 Web 用户控件制作一个博客导航条，实例运行结果如图 13.10 所示。

图 13.10　使用 Web 用户控件制作导航条

程序实现的主要步骤说明如下。

（1）打开 Visual Studio 2008 新建一个网站，默认主页为 Default.aspx。

（2）在该网站中添加一个 Web 用户控

件，默认名为 WebUserControl.ascx。在该 Web 用户控件中，添加 6 个 HyperLink 控件，分别设置其 ImageUrl（要显示图像的 URL）和 NavigateUrl（超链接页的 URL）属性值。将 Web 用户控件切换到“源”视图中，其完整的代码如下所示。

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.
cs" Inherits="WebUserControl" %>
<table height="157" width="759" align =center background="images/13.jpg" >
   <tr>     <td colspan =7 style="height: 116px">
      </td> </tr> <tr>  <td style="width: 185px; height: 23px"></td>
      <td style="width: 80px; height: 23px">
          <asp:HyperLink ID="HyperLink1" runat="server" Font-Underline="False"
          ImageUrl="~/images/3.jpg" NavigateUrl="~/Default.aspx"></asp:
          HyperLink></td>
      <td style="height: 23px; width: 83px;">
          <asp:HyperLink ID="HyperLink2" runat="server" Font-Underline="False"
          ImageUrl="~/images/4.jpg" NavigateUrl="~/Default.aspx">></asp:
          HyperLink></td>
      <td style="height: 23px; width: 66px;">
      <asp:HyperLink ID="HyperLink3" runat="server" Font-Underline="False"
      ImageUrl= "~/images/5.jpg" NavigateUrl= "~/Default.aspx">></asp:
      HyperLink></td>
      <td style="height: 23px; width: 83px;">
      <asp:HyperLink ID="HyperLink4" runat="server" Font-Underline="False"
      ImageUrl= "~/images/ 6.jpg" NavigateUrl= "~/Default.aspx">></asp:
      HyperLink></td>
      <td style="height: 23px; width: 70px;">
      <asp:HyperLink ID="HyperLink5" runat="server" Font-Underline="False"
      ImageUrl="~/images/ 7.jpg" NavigateUrl= "~/Default.aspx">></asp:
      HyperLink></td>
      <td style=" height: 23px">
      <asp:HyperLink ID="HyperLink6" runat="server" Font-Underline="False"
      ImageUrl="~/images/ 8.jpg" NavigateUrl= "~/Default.aspx">></asp:
      HyperLink></td>
   </tr>
</table>
```

（3）将制作好的 Web 用户控件拖曳到 Default.aspx 页中。切换到“源”视图，完整的代码如下所示。

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<%@ Register Src="WebUserControl.ascx" TagName="WebUserControl" TagPrefix=
"uc1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 13.0 Transitional//EN" "http://www.w3.
org/TR/ xhtml1/DTD/xhtml1- transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>使用 Web 用户控件制作导航条</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <uc1:WebUserControl ID="WebUserControl1" runat="server" />
     </div>
   </form>
</body>
```

```
</html>
```

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

```
<%@ Register Src="WebUserControl.ascx" TagName="WebUserControl" TagPrefix= "uc1" %>
```

在要使用该用户控件的页中，需通过@ Register 指令创建对该用户控件的引用。这里首先来了解一下@Register 指令。

此指令主要是创建标记前缀和自定义控件之间的关联，这为开发人员提供了一种在 ASP.NET 应用程序文件中引用自定义控件的简明方法。一般形式如下。

```
<%@ Register TagPrefix="tagprefix"  TagName="tagname"  Src="pathname" %>
```

此@ Register 指令的 3 个主要属性如下。

① TagPrefix 属性：为用户控件提供了标签前缀，该前缀可由用户定义。

② TagName 属性：提供了标签的名字。

③ Src 属性：用于指定用户控件的路径。这里特别注意的是 Src 属性指定的路径为虚拟路径，而不能为其指定绝对路径。

13.3　设置用户控件

专题讲座：光盘\MR\Video\13\设置用户控件.exe

➤➤➤视频速递：全面解读如何设置用户控件。

用户控件和 Web 控件一样可以被设置相关属性和方法。在设计用户控件时需要考虑到用户控件的用途，为用户控件公开一些能够使用的属性和方法，进而方便自定义控件的设计和编码。

13.3.1　访问用户控件的属性

ASP.NET 提供的各种服务器控件都有其自身的属性和方法，程序开发人员可以灵活地使用服务器控件中的属性和方法开发程序。

在用户控件中，属性是一种有效的并向类型使用者公开数据的字段，从类型使用者的角度来看，属性是一个 public 字段，通过实现一个属性，可以将使用者和实现细节相互隔离，同时还可以在属性被访问时提供数据有效检查、跟踪等处理手段。

创建用户控件的属性并没有什么特殊之处，其方法如同创建任何类的属性一样。

图 13.11　访问用户控件中定义的属性

实例位置：光盘\MR\Instance\13\13.3

【例 13.3】 本实例主要介绍如何访问用户控件中定

义的新属性，并将定义的用户控件的属性值（“ASP.NET 视频教学！”）显示在界面上，实例运行结果如图 13.11 所示。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，并在该页中添加一个 Label 控件，用于显示从用户控件中获取的属性值。

（2）在该网站中添加一个用户控件，默认名为 WebUserControl.ascx。

（3）在该用户控件中首先定义一个私有变量 txtName，并为其赋值，然后再定义一个公有变量，用来读取并返回私有变量的值，其代码如下所示：

```
private string txtName = "ASP.NET 视频教学！";  //私有变量，外部无法访问
public string str_txtName                        //定义公有变量来读取私有变量
{
    get { return txtName; }
    set { txtName = value; }
}
```

（4）将用户控件添加至 Default.aspx 页中，并在 Default.aspx 页的 Page_Load 事件下添加如下代码，获取用户控件的属性值，并将其显示出来。

```
protected void Page_Load(object sender, EventArgs e)
{
    //在当前页面中查找 WebUserControl1 用户控件
    WebUserControl wuc=(WebUserControl)Page.FindControl("WebUserControl1");
    //调用用户控件中设置的属性并输出其值
    Response.Write(wuc.str_txtName.ToString());
}
```

13.3.2 访问用户控件中的服务器控件

程序开发人员可以在用户控件中添加各种控件，如 Label 控件、TextBox 控件等，但当用户控件创建完成后，将其添加到网页时，在网页的后台代码中不能直接访问用户控件中的服务器控件的属性。为了实现对用户控件中的服务器控件的访问，必须在用户控件中定义公有属性，并且利用 GET 访问器与 SET 访问器来读取、设置控件的属性。

当用户控件包括在 Web 窗体页中时，此用户控件中包含的任何 ASP.NET 服务器控件的所有属性和方法都将提升为此用户控件的公共属性和方法。

图 13.12 访问用户控件中的服务器控件的属性

实例位置：光盘\MR\Instance\13\13.4

【例 13.4】本实例主要介绍如何访问用户控件中的服务器控件。执行程序，实例运行结果如图 13.12 所示，在该图中，当用户单击网页中的“登录”按钮时，首先获取用户控件中用户名和密码文本框的值，然后判断输入的值是否合法，如果是合法用户，则弹出欢迎用户光临的对话框。

程序实现的主要步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx，

并在该页中添加一个 Button 控件，用于判断用户输入的信息是否合法。

（2）在该网站中添加一个 Web 用户控件，默认名为 WebUserControl.ascx，并在该用户控件中添加两个文本框，分别用于输入用户名和密码。

（3）在该 Web 用户控件中定义两个公有属性，分别用于设置或读取各个文本框的 Text 属性，其代码如下所示：

```
public string str_Name                              //公有属性，访问“用户姓名”文本框
 {
    get { return this.TextBox13.Text; }             /返回“用户姓名”文本框的值
    set { this.TextBox13.Text = value; }            //设置“用户姓名”文本框的值
 }
 public string str_Pwd                              //公有属性，访问“密码”文本框
 {
    get { return this.TextBox2.Text; }              //返回“密码”文本框的值
    set { this.TextBox2.Text = value; }             //设置“密码”文本框的值
}
```

（4）将用户控件添加至 Default.aspx 页中，并在 Default.aspx 页的“登录”按钮的 Click 事件下添加如下代码，获取用户控件的文本框值，并判断用户输入是否合法。

```
protected void Button1_Click(object sender, EventArgs e)
{
   if (this.WebUserControl13.str_Name == "" || this.WebUserControl13.str_Pwd == "")
   {
      Response.Write("<script>alert('请输入必要的信息！')</script>");
   }
   else
   {
      if (this.WebUserControl13.str_Name == "mr" && this.WebUserControl13.str_
      Pwd == "mrsoft")
      {
         Response.Write("<script>alert('您是合法用户，欢迎您的光临！')</script>");
      }
      else
      {
         Response.Write("<script>alert('您的输入有误，请核对后重新输入！')
         </script>");
      }
   }
}
```

13.3.3　将 Web 网页转化为用户控件

将现有的 Web 页转换为用户控件。对于提高代码的重用性来说，这是最佳的选择方案。在程序开发过程中，当发现一个 Web 页会经常用到且使用的频率较高，并打算在整个应用程序中访问其功能，则可以对该页面略加改动，将其更改为一个用户控件，关键的转换技术如下。

将 Web 页转换为用户控件。这种技术一般分为两种情况。

（1）将单文件 ASP.NET 网页转换为用户控件

具体操作如下。

☑　重命名控件，使其文件扩展名为.ascx。

- ☑ 从该页面中移除 html、body 和 form 元素。
- ☑ 将@Page 指令更改为@Control 指令。
- ☑ 移除@Control 指令中除 Language、AutoEventWireup（如果存在）、CodeFile 和 Inherits 之外的所有属性。
- ☑ 在@Control 指令中包含 ClassName 属性。这允许将用户控件添加到页面时对其进行强类型化。

（2）将代码隐藏 ASP.NET 网页转换为用户控件

具体操作如下。

- ☑ 重命名.aspx 文件，使其文件扩展名为.ascx。
- ☑ 根据代码隐藏文件使用的编程语言，重命名代码隐藏文件使其文件扩展名为.ascx.cs。
- ☑ 打开代码隐藏文件并将该文件继承的类从 Page 更改为 UserControl。即将 System.Web.UI.Page 语句更改成 System.Web.UI.UserControl。

在.aspx 文件中，执行以下操作。

- ☑ 从该页面中移除 html、body 和 form 元素。
- ☑ 将@Page 指令更改为@Control 指令。
- ☑ 移除@Control 指令中除 Language、AutoEventWireup（如果存在）、CodeFile 和 Inherits 之外的所有属性。
- ☑ 在@Control 指令中，将 CodeFile 属性更改为指向重命名的代码隐藏文件。
- ☑ 在@Control 指令中包含 ClassName 属性。这允许将用户控件添加到页面时对其进行强类型化。

实例位置：光盘\MR\Instance\13\13.5

【例 13.5】 本实例实现使用@Reference 指令链接用户控件，并使用 LoadControl 方法将其加载到包含的 Web 窗体中。执行程序，实例运行结果如图 13.13 所示。

图 13.13 将 Web 网页转化为用户控件

程序开发的主要步骤说明如下。

（1）打开 Visual Studio 2008 新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页中分别添加一个 TextBox 控件、一个 Button 控件和一个 Lable 控件，分别用于输入用户名、执行相关事件的操作和显示特定信息。

（3）默认主页 Default.aspx 的源代码如下所示（此代码将与更改后的代码比较）。

```
    <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
    <html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
        <title>将 Web 页转换为 ASP.NET 用户控件</title>
    </head>
    <body>
    <script runat=server>
     protected void Button1_Click(object sender, EventArgs e)
    {
        Label13.Text = "Hi " + TextBox13.Text + " welcome to ASP.NET!";
```

```
}
</script>
    <form id="form1" runat="server">
    <div>
        Web Forms Page: <br />
        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="Enter" OnClick="Button1_
        Click" /><br />
        <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label>
</div>
  </form>
</body>
</html>
```

（4）重命名.aspx 页使其文件扩展名为.ascx。从该页面中移除 html、body 和 form 元素。

（5）将 ASP.NET 指令类型从@ Page 指令更改为@ Control 指令。

（6）移除@ Control 指令中除 Language、AutoEventWireup（如果存在）、CodeFile 和 Inherits 之外的所有属性。

更改后的代码如下所示。

```
<%@ Control  Language="C#" AutoEventWireup="true"%>
<script runat=server>
 protected void Button1_Click(object sender, EventArgs e)
{
    Label3.Text = "Hi " + TextBox3.Text + " welcome to ASP.NET!";
}
</script>
Web Forms Page: <br />
<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Enter" OnClick="Button1_Click" />
<br />
<asp:Label ID="Label3" runat="server" Text="Label"></asp:Label>
```

比较更改前与更改后的代码，在比较过程中要进一步理解如何将现有的 ASP.NET 页转化为用户控件这一编写用户控件最优的设计技术。

13.4　Web 用户控件的实际应用

专题讲座：光盘\MR\Video\13\ Web 用户控件应用.exe

➤➤➤视频速递：全面解读 Web 用户控件的应用。

通过以上章节的学习，读者对 Web 用户控件已经有了比较深的认识。那么，从本节开始，笔者将通过几个典型的实例来为读者演示 Web 用户控件的具体应用。

13.4.1　创建会员注册的 Web 用户控件

实例位置：光盘\MR\Instance\13\13.6

【例 13.6】 经常上网的读者对会员注册不会感到陌生，现在很多网站都提供会员注册

图 13.14　创建会员注册的 Web 用户控件

的功能，以便更好地提高网站人气并统计网站访问量。在庞大的网站里，可能在多个页面中出现会员注册的功能，如果都需要手动添加会员注册功能，这无疑是费时费力的，幸好 ASP.NET 中提供了 Web 用户控件，我们只需要开发一个具有会员注册功能的Web用户控件即可，需要的时候直接拖到页面中，实例运行结果如图 13.14 所示。

本实例的主要技术是如何在 Web 用户控件中实现会员注册功能，实现的基本思路与在普通页面中实现是一样的。首先需要使用 count 函数判断数据表中是否已经存在输入的用户名，具体代码如下所示：

```
    cmd.CommandText = "select count(*) from Tb_Login where UseName='"+txtUserName.
Text.Trim()+"'";
    int flag=int.Parse(cmd.ExecuteScalar().ToString());
```

如果返回值小于 0 说明可以注册，则使用 insert 语句将用户输入的注册信息添加到数据表中，代码如下所示：

```
    cmd.CommandText = "insert into Tb_Login(UseName,UsePassword,UserEmail,Question,
UserKey) values ('" + txtUserName.Text + "','" + txtPwd.Text + "','" + txtEmail.Text
+ "','" + txtquestion.Text + "','" + txtresult.Text + "')";
```

具体开发步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）向项目中添加一个 Web 用户控件，命名为 login.ascx，用于实现用户注册功能。

（3）在 login.ascx 中添加 5 个 TextBox 控件、两个 Button 控件、5 个 RequiredFieldValidator 控件和一个 RegularExpressionValidator 控件，分别用于输入注册信息、注册或重置、验证是否输入注册信息和验证输入的电子邮件地址是否合法。

（4）在“注册”按钮的 Click 事件中，首先判断输入的用户名是否可用。如果用户名可用，则使用 insert 语句将用户输入的注册信息添加到 Tb_Login 表中，代码如下所示：

```
    //获取 Web.config 中配置的数据库连接字符串
    string sqlstr = System.Configuration.ConfigurationManager.AppSettings["con"].
ToString();
    SqlConnection conn = new SqlConnection(sqlstr);    //连接数据库
    conn.Open();                                       //打开数据库
    SqlCommand cmd = new SqlCommand();                 //创建 SqlCommand 对象
    cmd.Connection = conn;                             //设置该对象使用 conn 连接数据库
    cmd.CommandType = CommandType.Text;                //设置类型
    //设置 sql 语句
    cmd.CommandText = "select count(*) from Tb_Login where UseName='"+txtUserName.
Text.Trim()+"'";
    int flag=int.Parse(cmd.ExecuteScalar().ToString()); //执行 SQL 语句并获取返回值
    if (flag > 0)                                      //如果大于 0
    {
        //说明用户已经存在，弹出提示信息
        Page.ClientScript.RegisterStartupScript(this.Parent.GetType(), "", "alert('
        该用户名已经存在');", true);
```

```
}
else                                                    //否则
{
     //执行 insert 语句，将注册信息插入数据表中
    cmd.CommandText = "insert into Tb_Login(UseName,UsePassword,UserEmail,
    Question,UserKey) values ('" + txtUserName.Text + "','" + txtPwd.Text + "','"
    + txtEmail.Text + "','" + txtquestion.Text +"','" + txtresult.Text + "')";
     if(cmd.ExecuteNonQuery()>0)                        //如果返回值大于 0，说明操作成功
{
//弹出提示信息
        Page.ClientScript.RegisterStartupScript(this.Parent.GetType(), "",
        "alert('用户注册成功');", true);
     }
     else                                               //否则
{
//提示用户注册失败
        Page.ClientScript.RegisterStartupScript(this.Parent.GetType(), "",
        "alert('用户注册失败');", true);
     }
}
conn.Close();                                           //关闭数据库连接
```

本实例使用的是 SQL Server 2005 数据库，连接数据库时，连接字符串是在 Web.config 中配置的，在代码中是通过 System.Configuration.ConfigurationManager.AppSettings["conn"].ToString()获取的，读者在自己电脑上配置数据库服务器实例名时，只需在 Web.config 文件中修改即可。

13.4.2　具有文件上传功能的 Web 用户控件

实例位置：光盘\MR\Instance\13\13.7

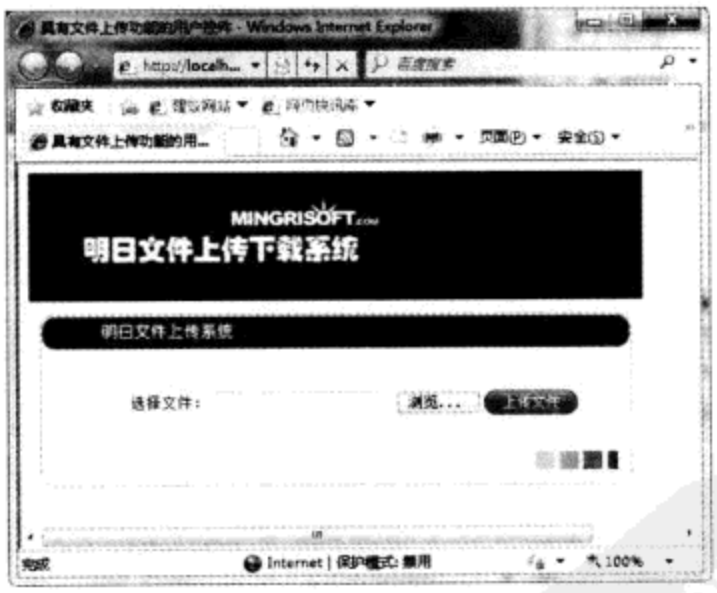


图 13.15　具有文件上传功能的 Web 用户控件

【例 13.7】 文件上传功能在网站中随处可见，可能会在网站的很多个页面中出现。如果每个页面都单独开发一个文件上传的功能，很显然这样做很不合理。幸好我们拥有 Web 用户控件，我们只需要做一个具有文件上传功能的 Web 用户控件，然后将其拖曳到需要文件上传功能的页面中即可，节省了时间，提高了效率。本实例的关键技术是 FileUpload 控件的基本应用，特别是 ContentLength 属性及 SaveAs 方法，分别用于获取上传文件的大小和将上传文件保存到服务器指定的目录下，如图 13.15 所示。

具体实现步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在该网站中添加一个 Web 用户控件，命名为 fileUpload.ascx，并在该用户控件中添加 FileUpload 控件和 Button 控件，分别用于选择上传文件和开始上传的功能。

（3）在 Web 用户控件 fileUpload.ascx 中 Button 按钮的 Click 事件下添加如下代码，用于将选择的文件上传到指定的目录中。

```
string serverPath = Server.MapPath("UpLoad");                    //获取服务器端目录绝对路径
if (!System.IO.Directory.Exists(serverPath))                     //如果不存在该目录
{
    System.IO.Directory.CreateDirectory(serverPath);             //创建该目录
}
if (FileUpload13.HasFile)                                        //判断是否选择上传的文件
{
    int filesize = FileUpload13.PostedFile.ContentLength / 1024 / 1024;
                                                                 //获取上传文件的大小
    if (filesize > 8)                                            //如果大于 8MB
    {
        Page.ClientScript.RegisterStartupScript(this.GetType(), "", "alert('只
        允许上传不大于 8 兆的文件');", true);                    //弹出提示信息
        return;
    }
    else                                                         //否则
    {
        //使用 SaveAs 方法将上传的文件存储到服务器中
        FileUpload13.SaveAs(serverPath + "\\" + FileUpload13.FileName);
        Page.ClientScript.RegisterStartupScript(this.GetType(), "", "alert('上
        传成功');", true);
    }
}
else                                                             //如果没有选择文件
{
    //弹出提示信息
    Page.ClientScript.RegisterStartupScript(this.GetType(), "", "alert('请选择
    文件');", true);
    return;
}
```

在 Web 用户控件中，ClientScript 必须通过引用 Page 类才能被编译器识别，例如，本例中 Page.ClientScript.RegisterStartupScript(this.Parent.GetType(), "", "alert('请选择文件');", true);。

13.4.3 创建在线投票的 Web 用户控件

实例位置：光盘\MR\Instance\13\13.8

【例 13.8】 一些大中型企业，经常在网站的首页设立一项在线投票功能，以便能够及时地了解本企业的产品或客户服务在广大市民心中的地位。为了在投票系统中确保准确率，防止重复投票是一项必不可少的功能。本实例将介绍如何在投票系统中防止重复投票。在线投票界面和投票结果界面如图 13.16 和图 13.17 所示。

为了在投票系统中确保准确率，防止重复投票是一项必不可少的功能，实现的方法有很多，本例主要通过 Cookie 对象来限制重复投票的现象。Cookie 提供了一种在 Web 应用程序中存储用户特定信息的方法。如利用 Cookie 存储用户登录的 IP 地址，只要用户在 Cookie 的有效期内登录网站，网站就可以识别该用户的身份。本实例就是利用这个特性，

自定义 Cookie 对象来防止网民们重复投票。下面介绍 Cookie 对象的用途和常用属性。

图 13.16　在线投票界面

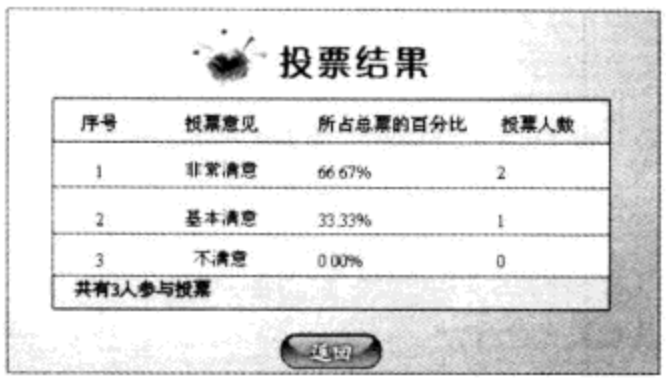


图 13.17　查看投票结果

由于 Cookie 与 Web 站点直接关联，因此只要用户发出浏览此 Web 站点中页面的请求，浏览器就会和服务器交换 Cookie 信息。Cookie 对象不隶属于 Page 对象，而分别隶属于 Request 对象和 Response 对象，每一个 Cookie 变量都被不同的 Cookie 对象所管理。

如果保存一个 Cookie 变量，需要使用 Response 对象的 Cookies 集合，语法如下。

```
Response.Cookies["变量名"].Value="表达式"
```

如果读取一个 Cookie 变量，需要使用 Request 对象的 Cookies 集合，语法如下。

```
username=Request.Cookies["变量名"].Value
```

Cookies 对象最常用的属性有以下几种。

☑ Expires 属性：设定 Cookie 变量的有效时间，默认为 1000 分钟，若设为 0 则可以实时删除 Cookie 变量。

☑ Name 属性：Cookie 变量的名称。

☑ Value 属性：Cookie 变量的值。

具体开发步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在该网站中添加一个 Web 用户控件，命名为 vote.ascx，并在该用户控件中添加一个 RadioButtonList 控件和两个 ImageButton 控件，分别用于选择投票项目和开始投票的功能。

（3）在 vote.ascx 的后台代码中，自定义 readCount 方法用来从 txt 文件中读取投票数量，addCount 方法用来向 txt 文本文件中写入投票数量。程序代码如下所示：

```
public static int readCount(string P_str_path)
{
    int P_int_count = 0;                                          //创建整型变量
    StreamReader streamread;                                      //创建流
    streamread = File.OpenText(P_str_path);                       //打开指定的文本文件
    while (streamread.Peek() != -1)                               //如果有数据
    {
        P_int_count = int.Parse(streamread.ReadLine()); //读取数据并赋值给整型变量
    }
    streamread.Close();                                           //关闭流
    return P_int_count;                                           //返回从文本文件中获取的值
}
public static void addCount(string P_str_path)
{
    int P_int_count = readCount(P_str_path);                      //读取现有投票数
```

```
    P_int_count += 1;                                          //记录数增加 1
    StreamWriter streamwriter = new StreamWriter(P_str_path, false);
                                                               //将数据记录写入文件
    streamwriter.WriteLine(P_int_count);                       //写入新的票数
    streamwriter.Close();                                      //关闭流
}
```

在该网站的虚拟目录下新建 3 个记事本文件 result1.txt、result2.txt 和 result3.txt，分别用来记录各投票选项的投票数量。

（4）单击“我要投票”按钮，程序首先判断用户是否已投过票，如果用户已投票，则弹出信息提示框，否则利用 Cookie 对象保存用户的 IP 地址，并弹出对话框提示用户投票成功。“我要投票”按钮的 Click 事件代码如下所示：

```
string P_str_IP = Request.UserHostAddress.ToString();
HttpCookie oldCookie = Request.Cookies["userIP"];
if (oldCookie == null)                                         //如果 Cookie 对象不存在
{
    int flag=RadioButtonList13.SelectedIndex;                  //获取投票项目的索引
    switch(flag)                                               //根据不同的选项执行分支下代码
    {
        case 0: addCount(Server.MapPath("result13.txt")); break;
                                                               //增加第一项的投票数
        case 1: addCount(Server.MapPath("result2.txt")); break;
                                                               //增加第二项的投票数
        case 2: addCount(Server.MapPath("result3.txt")); break;
                                                               //增加第三项的投票数
    }
    ClientScript.RegisterStartupScript(this.GetType(), "", "alert('投票成功，谢谢
    您的参与！');", true);
    HttpCookie newCookie = new HttpCookie("userIP");           //定义新的 Cookie 对象
    newCookie.Expires = DateTime.MaxValue;                     //设置 Cookie 有效时间
    //添加新的 Cookie 变量 IPaddress，值为 P_str_IP
    newCookie.Values.Add("IPaddress", P_str_IP);
    Response.AppendCookie(newCookie);                          //将变量写入 Cookie 文件中
}
else
{
   string P_str_oldIP = oldCookie.Values["IPaddress"];
   if (P_str_IP.Trim() == P_str_oldIP.Trim())                  //判断是否投过票
   {
      ClientScript.RegisterStartupScript(this.GetType(), "", "alert('一个 IP
      地址只能投一次票，谢谢您的参与！');", true);             //如果投过则弹出提示
    }
    else
    {
        HttpCookie newCookie = new HttpCookie("userIP");//创建 Cookie 对象
        newCookie.Values.Add("IPaddress", P_str_IP);
                                                               //将投票人的 IP 地址添加到 Cookie
        newCookie.Expires = DateTime.MaxValue;                 //设置 Cookie 有效时间
        Response.AppendCookie(newCookie);
        int rflag = RadioButtonList13.SelectedIndex;           //判断投票选项的索引
        switch (rflag)                                         //根据不同的选项执行分支下代码
        {
            case 0: addCount("result13.txt"); break;           //增加第一项的投票数
            case 1: addCount("result2.txt"); break;            //增加第二项的投票数
```

```
            case 2: addCount("result3.txt"); break;          //增加第三项的投票数
        }
        ClientScript.RegisterStartupScript(this.GetType(), "", "alert('投票成功,
        谢谢您的参与! ');", true);
    }
}
```

（5）在查看投票结果页面 result.aspx 的后台代码中，声明 3 个 string 类型的全局变量，用来记录各选项投票数量的百分比，再声明 3 个 int 类型的全局变量，用来记录各选项投票的具体数量，代码如下所示：

```
protected string M_str_rate1;                                    //第一项投票百分比
protected string M_str_rate2;                                    //第二项投票百分比
protected string M_str_rate3;                                    //第三项投票百分比
protected int P_int_count1;                                      //第一项投票数
protected int P_int_count2;                                      //第二项投票数
protected int P_int_count3;                                      //第三项投票数
```

为了更直观地显示投票结果，在 result.aspx 页面中将以百分比的形式来显示。实现此功能时，首先需要将用来记录各选项百分比的全局变量绑定到 Table 表格的单元格中，以绑定第一个投票选项所获票数的百分比为例，实现代码如下所示：

```
<td class="style8"><% =M_str_rate1%></td>
```

result.aspx 页面在加载时，获取各选项的投票数量和总的投票数量，然后判断总投票数量是否为 0，如果为 0，弹出信息提示框，说明还没有人投过票，否则计算各选项所占的百分比，并将其分别赋值给对应的全局变量。result.aspx 页面的 Page_Load 事件代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    P_int_count1 = readCount(Server.MapPath("result13.txt")); //获取第一项投票数
    P_int_count2 = readCount(Server.MapPath("result2.txt")); //获取第二项投票数
    P_int_count3 = readCount(Server.MapPath("result3.txt")); //获取第三项投票数
    int P_int_count = P_int_count1 + P_int_count2 + P_int_count3;
                                                             //计算投票总数
    if (P_int_count == 0)                                    //如果投票总数为 0
    {
        Response.Write("<script>alert('还没有人投过票! ')</script>");
                                                             //弹出提示信息
        lblresult.Text = "共有 0 人参与投票";                  //显示没有人投票
    }
    else                                                     //否则
    {
        M_str_rate1 = (Convert.ToDouble(P_int_count1) * 100 / Convert.ToDouble
        (P_int_count)). ToString("0.00") + "%";              //第一个选项得票百分比
        M_str_rate2 = (Convert.ToDouble(P_int_count2) * 100 / Convert.ToDouble
        (P_int_count)).ToString("0.00") + "%";               //第二个选项得票百分比
        M_str_rate3 = (Convert.ToDouble(P_int_count3) * 100 / Convert.ToDouble
        (P_int_count)).ToString("0.00") + "%";               //第三个选项得票百分比
        lblresult.Text = "共有" + P_int_count.ToString() + "人参与投票";
                                                             //统计并显示投票人数
    }
}
```

学习手册 说明 页面中也要自定义一个 readCount 方法，此方法与 vote.ascx 中的 readCount 方法是同一个方法，所以此处不再赘述。

13.5 实 战 练 习

13.5.1 利用 Web 用户控件实现选择日期的功能

图 13.18 利用 Web 用户控件实现选择日期的功能

➤➤➤题目描述

用户注册会员时，表单中大多数都需要输入出生日期、毕业日期或者工作日期等。对于日期的输入现在基本上存在两种形式：一种是直接手动输入；另一种是弹出一个日期选择器，在日期选择器中选择相应的日期。本实例开发了一个 Web 用户控件，实现选择日期的功能，如图 13.18 所示。

➤➤➤技术指导

本实例是通过 Web 用户控件和 jQuery 实现的，读者可以打开 jQuery 的官方网站 http://www.jquery.com，下载 datepick 插件和 jQuery 框架。解压缩后，将 jquery.datepick.js、jquery.datepick-zh-CN.js 和 jquery.1.3.2.min.js 复制到项目根目录的 js 文件夹下。同时，将压缩包里的 redmond.datepick.css 样式文件也复制到 js 文件夹下，压缩包里有很多样式文件，读者可以自己选择使用哪个样式。然后在 Web 用户控件中引入这些文件，之后初始化日期选择器插件 datepick，代码如下所示：

```
$('#txtdate').datepick({ dateFormat: 'yyyy-mm-dd'});
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/13.5.1**

13.5.2 网页软键盘

图 13.19 网页软键盘

➤➤➤题目描述

随着网络的普及，在网络上进行交易已经逐渐成为一种潮流。例如，在网上购物、网上转账等。大多数银行的网上银行系统在登录时都提供了一个虚拟的网页键盘，避免通过键盘输入，从

而增强了安全性，防止黑客攻击。本实例实现的是开发一个网页软键盘的 Web 用户控件，运行效果如图 13.19 所示。

➤➤➤技术指导

本实例是通过 Web 用户控件和 jQuery 实现的，读者可以打开 jQuery 的官方网站 http://www.jquery.com，下载 keypad 插件和 jQuery 框架。解压缩后，将 jquery.keypad.js 和 jquery.1.3.2.min.js 复制到项目根目录的 js 文件夹下。同时，将压缩包里的 jquery.keypad.alt.css 样式文件也复制到 js 文件夹下，压缩包里有其他样式文件，读者可以自己选择使用哪个样式。然后在 Web 用户控件中引入这些文件，之后初始化虚拟键盘插件 keypad，代码如下所示：

```
    $(function () {
    $('#defaultKeypad').keypad({prompt:'',keypadOnly:false,layout:$.keypad.qwer
tyLayout});
    });
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/13.5.2**

13.6　本 章 小 结

本章主要介绍了.NET 中代码复用的相关内容，即 Web 用户控件。事实上，对于初学者第一次使用用户控件很难理解这种控件的优点，用户控件的一个最突出的优势，是创建起来非常容易。它使得 ASP.NET 中代码复用能够以简单的方式来实现。可以通过本章的一些实例应用设计出自己所希望的各种用户控件。掌握 ASP.NET 用户控件的开发和使用，对今后网站的开发和制作都很有帮助。

第14章

ASP.NET 验证控件

（名师课堂：35 分钟）

为了提高 Web 开发人员的开发效率和降代错误出现的几率，ASP.NET 技术为 Web 开发人员提供了数据验证控件。这些验证控件可以实现非空验证、数据比较验证、数据类型验证、数据格式验证、数据范围验证、验证错误集合和自定义验证等，通过本章的学习，读者可以达到以下目的：

- 使用 RequiredFieldValidator 控件实现数据的非空验证
- 使用 CompareValidator 控件实现数据的比较验证
- 使用 RegularExpressionValidator 控件实现数据的输入格式验证
- 使用 RangeValidator 控件实现数据的范围验证
- 使用 ValidationSummary 控件实现验证错误信息提示
- 使用 CustomValidator 控件实现自定义验证

14.1　了解何谓验证控件

通过对本节的学习，读者能够了解什么是验证控件以及验证控件的工作原理。只有了解了原理，在以后的学习中才能够更加得心应手。

14.1.1　什么是验证控件

ASP.NET 提供了一组验证控件，只要通过属性设定，它们便会自动产生相关的 JavaScript，当用户输入时直接在浏览器中检查，响应速度快。基本上每个验证控件只能做一种检查，如有没有输入值、是否在某个范围内等。当用户输入的值不符合条件时，验证控件会提示报错信息，且中断后续的数据处理操作。ASP.NET 提供了以下 6 种验证控件。

- ☑ RequiredFieldValidator：检查某个字段是否输入。
- ☑ CompareValidator：将某个字段的内容与指定的对象进行比较。
- ☑ RangeValidator：检查某个字段的内容是否处在指定的范围内。
- ☑ RegularExpressionValidator：检查某个字段的内容是否符合指定的格式，如电话号码、身份证。
- ☑ CustomValidator：可自己编写一段程序，自定义验证条件。
- ☑ ValidationSummary：这个控件不提供验证功能，只负责显示所有的验证报错信息。

14.1.2　验证控件的工作原理

在对用户提交的数据进行验证方面，ASP.NET 有两道验证：用户的浏览器（即客户端）及 Web 服务器，如图 14.1 所示。

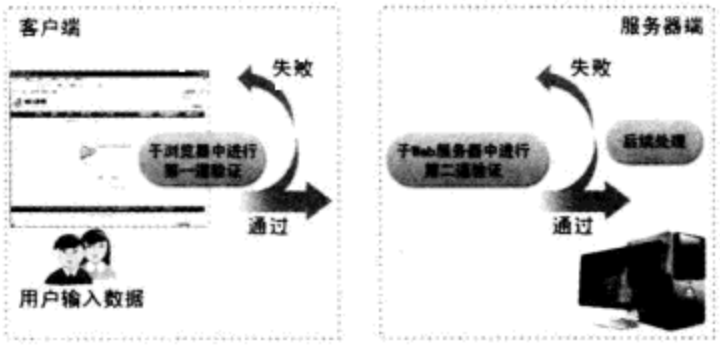


图 14.1　ASP.NET 两道验证

1．客户端验证

在客户端进行验证响应速度快，在最接近使用的地方进行检查，一旦有错会被浏览器中的 JavaScript 拦截下来，用户必须更正错误，通过验证，才能继续之后的处理。这里所说的执行拦截检查操作的 JavaScript 在 ASP.NET 会自动建立。若要取消客户端验证，可将验证控件的 EnableClientSideScript 属性设置为 False。

学习手册 说明　目前大部分浏览器都支持 JavaScript 的执行功能。若用户使用的浏览器不支持，或者考虑安全问题而关闭了 JavaScript 执行功能，ASP.NET 仍然会在服务器端进行验证。

2. 服务器端验证

无论是否通过客户端的检查，服务器端的 ASP.NET 程序会对所有提交的数据进行检查，如与服务器端的数据库内容进行核对。在服务器端再重复一次相同的检查仍是必要的，因为“有心”的用户可以通过一些手法，跳过或免除客户端的验证步骤，所以开发 Web 网站其安全性是非常重要的，这就像对重要的物品在存放到仓库中要有两道或多道防盗门一样设置多重防范措施。

14.2 验证是否输入数据

在网站的实际开发中，RequiredFieldValidator 验证控件用来验证输入文本中的信息内容是否为空。例如，注册会员信息时，用户名称、密码、电话、住址等较重要的信息为必填项，这时需要使用 RequiredFieldValidator 控件来进行验证，本节对该控件进行详细介绍。

14.2.1 RequiredFieldValidator 控件

网页上有些字段是一定要输入的，如在注册会员时的“用户名”及“密码”，至于家庭地址、电子邮件等不输入也没有关系。此时，可用 RequiredFieldValidato 检查必要字段是否被输入。Visual Studio 2008 工具箱中的 RequiredFieldValidator 控件如图 14.2 所示。

RequiredFieldValidator 控件的部分常用属性如表 14.1 所示。

表 14.1 RequiredFieldValidator 控件最常用的属性

属　性	描　述
ID	控件 ID，控件唯一标识符
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的控件必须和验证控件在相同的容器中
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	获取或设置一个值，该值指示控件验证的数据是否有效。默认值为 true
Display	设置错误信息的显示方式
Text	如果 Display 为 Static，则不出错时显示该文本

下面对比较重要的属性进行介绍。

（1）ControlToValidate 属性

该属性获取或设置要验证的输入控件。

语法：

```
public string ControlToValidate { get; set; }
```

属性值：要验证的控件 ID。

例如，要验证 ID 属性为 TextBox1 的 TextBox 控件，只要将 RequiredFieldValidator 控件的 ControlToValidate 属性设置为 TextBox1 即可，如图 14.3 所示。

其他属性用默认值或不设置值也没有关系，但 ControlToValidate 无论如何一定要设置，不可以是空白的，否则网页执行时就会报错。

图 14.2　RequiredFieldValidator 控件

图 14.3　设置 ControlToValidate 属性来指定验证对象

（2）ErrorMessage 属性

该属性用于指定页面中使用 RequiredFieldValidator 控件时显示的错误消息文本。

语法：

```
public string ErrorMessage { get; set; }
```

属性值：验证失败时 ValidationSummary 控件中显示的错误信息。

【例 14.1】 将 RequiredFieldValidator 控件的错误消息文本设为“*”。

```
this.RequiredFieldValidator1.ErrorMessage = "*";
```

14.2.2　情景应用：验证用户是否输入信息

专题讲座：光盘\MR\Video\14\验证用户是否输入信息.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 RequiredFieldValidator 控件。

实例位置：光盘\MR\Instance\14\14.1

【例 14.2】 本实例设计一个会员注册时使用 RequiredFieldValidator 控件验证用户是否输入用户名和密码，但 E-mail 地址可以不输入。如果用户名及密码内容为空时，单击“注册”按钮显示错误提示信息，运行结果如图 14.4 所示。

图 14.4　RequiredFieldValidator 检查必要的输入

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 3 个 TextBox 控件、两个 RequiredFieldValidator 控件和一个 Button 控件，它们的属性设置如表 14.2 所示。

表 14.2　Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用　途
标准/TextBox 控件	txtName	均为默认设置	输入姓名

续表

控件类型	控件名称	主要属性设置	用途
标准/TextBox 控件	txtPwd	TextMode 属性设为“Password”	输入密码
	txtEmail	均为默认设置	输入密码
标准/Button 控件	btnCheck	Text 属性设置为“注册”	执行页面提交的功能
验证/Required Field-Validator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为“txtName”	设置要验证的控件 ID
		ErrorMessage 属性设置为“输入用户名”	显示的错误信息为“输入用户名”
	RequiredFieldValidator2	ControlToValidate 属性设置为“txtPwd”；ErrorMessage 属性设置为“输入密码”	验证密码是否为空

14.3 比较数据是否一致

CompareValidator 控件称为比较验证控件，其功能是将输入的值与常数值或其他控件输入的值相比较，以确定这两个值是否与比较运算符（小于、等于、大于等）指定的关系相匹配。通过对本节的学习，读者可以完全掌握 CompareValidator 控件的使用。

14.3.1 CompareValidator 控件

图 14.5 CompareValidator 控件

CompareValidator 控件，顾名思义可用来作比较，用户可将指定的值与一个固定的常量或另一个值进行比较，Visual Studio 2008 工具箱中的 CompareValidator 控件如图 14.5 所示。

为了更好地掌握 CompareValidator 控件，首先要了解该控件的几个重要属性。CompareValidator 控件的部分常用属性如表 14.3 所示。

表 14.3 CompareValidator 控件最常用的属性

属性	描述
ID	控件的唯一标识
ControlToCompare	设置用于比较的控件 ID
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的空间必须和验证控件在相同的容器中
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	获取或设置一个值，该值指示控件验证的数据是否有效
Operator	获取或设置验证中使用的比较操作
Type	获取或设置比较的两个值的数据类型

下面对比较重要的属性进行介绍。

（1）ControlToCompare 属性

该属性指定要对其进行值比较的控件 ID。

语法：

```
public string ControlToCompare { get; set; }
```

属性值：要与所验证的输入控件进行比较的输入控件。

【例 14.3】 ID 属性为 txtRePwd 的 TextBox 控件与 ID 属性为 txtPwd 的 TextBox 控件进行比较验证。

```
CompareValidator1.ControlToCompare= "txtPwd";
CompareValidator1. ControlToValidate = "txtRePwd";
```

（2）Type 属性

该属性指定要对其进行比较的两个值的数据类型。

语法：

```
public ValidationDataType  Type { get; set; }
```

属性值：ValidationDataType 枚举值之一，默认值为 String。ValidationDataType 枚举值及说明如表 14.4 所示。

表 14.4　ValidationDataType 枚举值及说明

枚 举 值	描　述
Currency	货币数据类型。该值被视为 System.Decimal，但仍允许使用货币和分组符号
Date	日期数据类型。仅允许使用数字日期，不能指定时间部分
Double	双精度浮点数数据类型。该值被视为 System.Double
Integer	32 位有符号整数数据类型。该值被视为 System.Int32
String	字符串数据类型。该值被视为 System.String

【例 14.4】验证 ID 属性为 TextBox1 的 TextBox 控件与 ID 属性为 TextBox2 的 TextBox 控件中值的类型是否为 string 类型。

```
this.CompareValidator1.Type = ValidationDataType.String;
```

（3）Operator 属性

该属性指定比较验证中使用的比较操作。

语法：

```
public ValidationCompareOperatorOperator { get; set; }
```

属性值：ValidationCompareOperator 值之一，默认值为 Equal。ValidationCompareOperator 枚举值及说明如表 14.5 所示。

表 14.5　ValidationCompareOperator 枚举值及说明

枚 举 值	描　述
DataTypeCheck	只对数据类型进行比较
Equal	相等比较
GreaterThan	大于比较
GreaterThanEqual	大于或等于比较
LessThan	小于比较
LessThanEqual	小于或等于比较
NotEqual	不等于比较

【例 14.5】 使用 Operator 属性的 DataTypeCheck 值来检查用户输入的日期是否合法，例如，验证 2010/10/32 这样的日期是否合法。实际运行结果如图 14.6 所示。

CompareValidator 控件前台 HTML 代码设置如下：

```
<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToValidate="txtdate" Display="Dynamic" ErrorMessage="日期不合法" Operator="DataTypeCheck" Type="Date"></asp: CompareValidator>
```

（4）ValueToCompare 属性

该属性指定要比较的值。如果 ValueToCompare 和 ControlToCompare 属性都存在，则使用 ControlToCompare 属性的值。

【例 14.6】 网页中使用一个 CompareValidator 检查用户输入的订购数量是否大于 0，运行结果如图 14.7 所示。

图 14.6 检查日期是否合法

图 14.7 检查订购数量是否大于 0

CompareValidator 控件前台 HTML 代码设置如下：

```
<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToValidate="txtQty" ErrorMessage="订购数量要大于 0" Operator="GreaterThan" Type="Integer" ValueToCompare="0"></asp: CompareValidator>
```

14.3.2 情景应用：验证两次密码输入是否一致

专题讲座：光盘\MR\Video\14\验证两次密码输入是否一致.exe

视频速递：通过本视频读者可以更好地掌握 CompareValidator 控件。

实例位置：光盘\MR\Instance\14\14.2

图 14.8 注册时两次密码输入必须相符

【例 14.7】本实例是应用 CompareValidator 对网页上的两个字段进行比较。如图 14.8 所示，当会员注册时，为避免密码打错、记错，通常会要求用户输入两次密码，此时便可用 CompareValidator 控件检查两次输入的密码是否相同。

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 3 个 TextBox 控件、3 个 RequiredFieldValidator 控件、一个 CompareValidator 控件和一个 Button 控件，它们的属性设置如表 14.6 所示。

表 14.6　Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用　途
标准 /TextBox 控件	txtName	均为默认设置	输入姓名
	txtPwd	TextMode 属性设为 Password	输入密码
	txtRePwd	TextMode 属性设为 Password	确认密码
标准/Button控件	btnCheck	Text 属性设置为“注册”	提交页面
RequiredField-Validator 控件	RequiredFiel d Validator1	ControlToValidate 属性设为“txtName”	要验证的控件的 ID 为 txtName
		ErrorMessage 属性设为“姓名不能为空”	显示的错误信息为“姓名不能为空”
		SetFocusOnError 属性设置为 True	验证无效时，在该控件上设置焦点
RequiredField-Validator 控件	RequiredFiel d Validator2	ControlToValidate 属性设置为“txtPwd”	要验证的控件的 ID 为 txtPwd
		ErrorMessage 属性设置为“密码不能为空”	显示的错误信息为“密码不能为空”
		SetFocusOnError 属性设置为 True	验证无效时，在该控件上设置焦点
RequiredField-Validator 控件	RequiredFiel dValidator3	ControlToValidate 属性设置为“txtRePwd”	要验证的控件的 ID 为 txtRePwd
		ErrorMessage 属性设置为“请重新输入密码”	显示的错误信息为“请重新输入密码”
		SetFocusOnError 属性设置为 True	验证无效时，在该控件上设置焦点
Compare-Validator 控件	Compare-Validator1	ControlToValidate 属性设置为“txtRePwd”	要验证的控件的 ID 为 txtRePwd
		ControlToCompare 属性设置为“txtPwd”	进行比较的控件 ID 为 txtPwd
		ErrorMessage 属性设置为“密码不一致”	显示的错误信息为“密码不一致”

14.4　验证输入数据的范围

在实际开发过程中，RangeValidator 验证控件用来验证输入控件中的数据是否在指定的下限与上限之间。例如，年龄输入限制范围为 18~68，日期输入限制范围 2000-01-01～2010-01-01，这时需要使用 RangeValidator 控件来验证，本节对该控件会进行详细介绍。

14.4.1　RangeValidator 控件

使用 RangeValidator 控件验证用户输入是否在指定范围之内。可以通过对 RangeValidator 控件的上、下限属性以及指定控件要验证的值的数据类型的设置完成这一功能。如果用户的输入无法转换为指定的数据类型（如无法转换为日期）则验证将失败。如果用户将控件保留为空白，则此控件将通过范围验证。若要强制用户输入值，则还要添加 RequiredFieldValidator 控件。Visual Studio 2008 工具箱中的 RangeValidator 控件如图 14.9 所示。

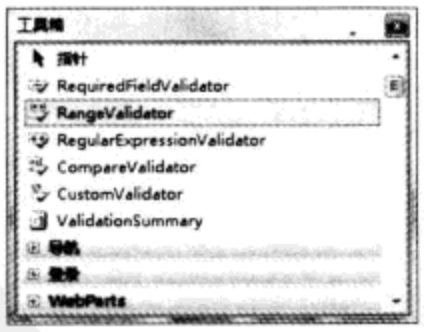


图 14.9　RangeValidator 控件

RangeValidator 控件的部分常用属性如表 14.7 所示。

表 14.7　RangeValidator 控件最常用的属性

属　性	描　述
ID	控件 ID，控件唯一标识符
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的控件必须和验证控件在相同的容器中

续表

属　性	描　述
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	控件验证的数据是否有效，默认值为 true
Display	设置错误信息的显示方式
MaximumValue	设置要验证的控件的值，该值必须小于或等于此属性的值
MinimumValue	设置要验证的控件的值，该值必须大于或等于此属性的值
Text	如果 Display 为 Static，则不出错时显示该文本
Type	获取或设置一种数据类型，用于指定如何解释要比较的值

下面对比较重要的属性进行介绍。

（1）MaximumValue 属性

该属性获取或设置验证范围的最大值。

语法：

```
public string MaximumValue { get; set; }
```

属性值：验证范围的最大值。

（2）MinimumValue 属性

该属性获取或设置验证范围的最小值。

语法：

```
public string MinimumValue { get; set; }
```

属性值：验证范围的最小值。

为了说明范围，RangeValidator 控件提供了这两个属性，它们分别用最小及最大值指定范围。如果是开发范围，如大于 0 或小于某个值，就改用 CompareValidator，而 RangeValidator 适用于封闭范围的检查。

图 14.10　用 RangeValidator 检查报名人数

【例 14.8】若要检查用户输入的报名人数是否在 1~50 之间，便可用 RangeValidator，如图 14.10 所示。

实现的代码如下所示：

```
<asp:RangeValidator ID="RangeValidator1" runat="server"
ErrorMessage="报名人数限制在 1~50 之间" MaximumValue="50" MinimumValue="1"
SetFocusOnError="True" ControlToValidate="txtnum" Display="Dynamic"
Type="Integer"></asp:RangeValidator>
```

（3）Type 属性

该属性设置在比较之前将所比较的值转换到的数据类型。

语法：

```
public ValidationDataTypeType { get; set; }
```

属性值：ValidationDataType 枚举值之一，默认值为 String。ValidationDataType 枚举值及说明如表 14.8 所示。

表 14.8　ValidationDataType 枚举值及说明

枚 举 值	说　明
Currency	货币数据类型。该值被视为 System.Decimal，但仍允许使用货币和分组符号
Date	日期数据类型。仅允许使用数字日期，不能指定时间部分
Double	双精度浮点数数据类型。该值被视为 System.Double
Integer	32 位有符号整数数据类型。该值被视为 System.Int32
String	字符串数据类型。该值被视为 System.String

Type 属性的应用请读者参看例 14.8 中的代码，此处就不再赘述。

14.4.2　情景应用：验证输入的日期是否在指定范围内

专题讲座：光盘\MR\Video\14\验证输入的日期是否在指定范围内.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 RangeValidator 控件。

实例位置：光盘\MR\Instance\14\14.3

【例 14.9】 本实例设计一个用户注册页面，在该页面中要求用户输入出生日期。出生日期并非随便输入的，对其格式和范围要进行验证。本实例中限制出生日期的范围是 1960/1/1 ~1992 /12/31，如果超过这个范围则显示提示信息，如图 14.11 所示。

图 14.11　用 RangeValidator 检查输入值是否在某个范围内

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 4 个 TextBox 控件、一个 RangeValidator 控件和一个 Button 控件，它们的属性设置如表 14.9 所示。

表 14.9　Default.aspx 页控件属性设置及说明

控 件 类 型	控 件 名 称	主要属性设置	用　途
标准/TextBox 控件	txtDate	无	输入日期
标准/Button 控件	btnCheck	Text 属性设置为“确定”	执行页面提交的功能
验证/ Range-Validator 控件	RangeValidator1	ControlToValidate 属性设置为“txtDate”	要验证的控件的 ID 为 txtDate
		ErrorMessage 属性设置为“日期只能在 1960/1/1~1992/12/31”	显示的错误信息为“日期只能在 1960/1/1~1992/12/31”
		MaximumValue 属性设置为 1992/12/31	最大日期值为 1992/12/31
		MinimumValue 属性设置为 1960/1/1	最小日期值为 1960/1/1
		Type 属性设置为“Date”	日期型比较

14.5　验证数据输入格式

在实际开发过程中，RegularExpressionValidator 验证控件用来验证输入控件的值是否

与某个正则表达式所定义的模式相匹配。如身份证号码、电子邮件地址、电话号码、邮政编码等中的字符序列，这时需要使用 RegularExpressionValidator 控件来验证，此控件的应用还是比较广泛的，希望读者能够认真学习。

图 14.12 RegularExpression Validator 控件

14.5.1 RegularExpressionValidator 控件

使用 RegularExpressionValidator 控件可以验证用户输入是否与预定义的模式相匹配，预定义的模式是通过正则表达式定义的。这样就可以对电话号码、邮编、网址等进行验证。Visual Studio 2008 工具箱中的 RegularExpressionValidator 控件如图 14.12 所示。

RegularExpressionValidator 控件的部分常用属性如表 14.10 所示。

表 14.10 RegularExpressionValidator 控件最常用的属性

属　性	描　述
ID	控件 ID，控件唯一标识符
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的空间必须和验证控件在相同的容器中
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	获取或设置一个值，该值指示控件验证的数据是否有效。默认值为 true
Display	设置错误信息的显示方式
Text	如果 Display 为 Static，则不出错时显示该文本
ValidationExpression	获取或设置被指定为验证条件的正则表达式。默认值为空字符串（""）

RegularExpressionValidator 控件的属性与 RequiredFieldValidator 控件大致相同，这里只对 ValidationExpression 属性进行具体介绍。

了解其用途不难，难在如何设置“正则表达式(Regular Expression)”。好在 Visual Studio 开发工具已经将常用的格式定义好，用户无须烦恼如何以 Regular Expression 定义格式，只要在设置 ValidationExpression 属性时，从预先定义好的格式中选取即可。设置 RegularExpressionValidator 控件验证模式的具体方法说明如下。

（1）从工具箱中将 RegularExpressionValidator 控件拖曳到页面中。

（2）选中 RegularExpressionValidator 控件，在属性窗口中选中 ValidationExpression 属性。

（3）单击 ValidationExpression 属性后面的“...”按钮，打开“正则表达式编辑器”，如图 14.13 所示。

图 14.13 正则表达式编辑器

（4）在“正则表达式编辑器”中选择要验证的模式。

RegularExpressionValidator 利用正则表达式定义格式，虽然 Visual Studio 开发工具已定义好了许多常用的格式，但在有些情况下，用户仍旧需要了解如何用正则表达式定义自定义的格式，如学生证编号、产品编程等。常用的正则表达式字符及其含义如表 14.11 所示。

表 14.11　常用正则表达式字符及其含义

正则表达式字符	含　义
[……]	匹配括号中的任何一个字符
[^……]	匹配不在括号中的任何一个字符
\w	匹配任何一个字符（a~z、A~Z 和 0~9）
\W	匹配任何一个空白字符
\s	匹配任何一个非空白字符
\S	与任何非单词字符匹配
\d	匹配任何一个数字（0~9）
\D	匹配任何一个非数字（^0~9）
[\b]	匹配一个退格键字母
{n,m}	最少匹配前面表达式 n 次，最大为 m 次
{n,}	最少匹配前面表达式 n 次
{n}	恰恰匹配前面表达式为 n 次
?	匹配前面表达式 0 或 1 次{0,1}
+	至少匹配前面表达式 1 次{1,}
*	至少匹配前面表达式 0 次{0,}
|	匹配前面表达式或后面表达式
(…)	在单元中组合项目
^	匹配字符串的开头
$	匹配字符串的结尾
\b	匹配字符边界
\B	匹配非字符边界的某个位置

根据上述表格读者认识了正则表达式中的关键词之后，现在来理解一下正则表达式“（[A-Z]|[a-z]）\d{9}”的含义。

（[A-Z]|[a-z]）：第一个格式单位，[A-Z]及[a-z]表示只能输入大小写的 A 到 Z 之间的字符。

\d{9}：\d 表示只能输入 0 到 9 之间的数字，而{9}表示一定要出现 9 次。

下面再来列举几个常用的正则表达式。

验证电子邮件：

☑　\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*。

☑　\S+@\S+\. \S+。

验证网址：

☑　HTTP://\S+\. \S+。

☑　HTTP://\S+\. \S+。

验证邮政编码：

☑　\d{6}。

其他常用正则表达式：

☑　[0-9]：表示 0~9 十个数字。

☑　\d*：表示任意个数字。

☑　\d{3,4}-\d{8,8}：表示中国大陆的固定电话号码。

☑ \d{2}-\d{5}：验证由两位数字、一个连字符再加 5 位数字组成的 ID 号。
☑ <\s*(\S+)(\s[^>]*)?>[\s\S]*<\s*\/\1\s*>：匹配 HTML 标记。

14.5.2 情景应用：验证邮件、生日等是否正确

专题讲座：光盘\MR\Video\14\验证邮件、生日等是否正确.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 RegularExpressionValidator 控件。

实例位置：光盘\MR\Instance\14\14.4

【例 14.10】 本实例主要通过 RegularExpressionValidator 控件的 ControlToValidate 属性、Operator 属性和 Type 属性验证用户输入的出生日期与日期类型是否匹配以及 E-mail 格式是否正确；另外，对于用户名的输入自定义了一个正则表达式来限制用户名只能输入字母、下画线及数字。执行程序，输入错误的日期格式和 E-mail 格式，单击“确定”按钮，实例运行结果如图 14.14 所示。

图 14.14 RegularExpressionValidator 控件验证示例

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 5 个 TextBox 控件、一个 RequiredFieldValidator 控件、两个 CompareValidator 控件和一个 Button 控件，它们的属性设置如表 14.12 所示。

表 14.12 Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用途
标准 /TextBox 控件	txtName	均为默认设置	输入姓名
	txtPwd	TextMode 属性设置为“Password”	设置为密码格式
	txtRePwd	TextMode 属性设置为“Password”	设置为密码格式
	txtEmail	均为默认设置	输入 E-mail 地址
	txtBirth	均为默认设置	输入出生日期
标准 /Button 控件	btnCheck	Text 属性设置为“确定”	执行页面提交的功能
验证/RequiredFieldValidator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为“txtName”	要验证的控件的 ID 为 txtName
		ErrorMessage 属性设置为“姓名不能为空”	显示的错误信息为“姓名不能为空”
		Display 属性设置为 Dynamic	动态调整报错信息出现的位置

续表

控件类型	控件名称	主要属性设置	用　途
验证/CompareValidator 控件	CompareValidator1	ControlToValidate 属性设置为“txtRePwd”	要验证的控件的 ID 为 txtRePwd
		ControlToCompare 属性设置为“txtPwd”	进行比较的控件 ID 为 txtPwd
		ErrorMessage 属性设置为“确认密码与密码不匹配”	显示的错误信息为“确认密码与密码不匹配”
	CompareValidator2	ControlToValidate 属性设置为“txtBirth”	要验证的控件的 ID 为 txtBirth
		ErrorMessage 属性设置为“日期格式错误”	显示的错误信息为“日期格式错误”
		Operator 属性设置为“DataTypeCheck”	对值进行数据类型验证
		Type 属性设置为“Date”	进行日期比较
验证/验证/RegularExpressionValidator 控件	RegularExpressionValidator1	ControlToValidate 属性设置为“txtName”	要验证的控件的 ID 为 txtName
		ErrorMessage 属性设置为“用户名能为输入字母、下画线和数字！”	验证提示信息
		ValidationExpression 属性设置为“^\w+$”	进行有效性验证的正则表达式
	RegularExpressionValidator2	ControlToValidate 属性设置为“txtEmai;”	要验证的控件的 ID 为 txtEmail
		ErrorMessage 属性设置为“格式有误”	显示的错误信息为“格式有误”
		ValidationExpression 属性设置为“\w+([	+.']\w+)*@\w+([

学习手册 说明　邮政编码的格式验证可通过“正则表达式编辑器”来选择，如图 14.15 所示。

14.6　验证错误信息汇总

ValidationSummary 控件又称错误汇总控件，主要用于收集本页中所有验证控件错误信息，将它们组织好并一同显示出来，此控件属于服务器端验证控件。通过该控件使错误提示的方式更加灵活，所以，读者要认真学习本节的内容。

14.6.1　ValidationSummary 控件

使用 ValidationSummary 控件可以为用户提供将窗体发送到服务器时所出现错误的列表。错误列表可以通过列表、项目符号列表或单个段落的形式进行显示。Visual Studio 2008 工具箱中的 ValidationSummary 控件如图 14.16 所示。

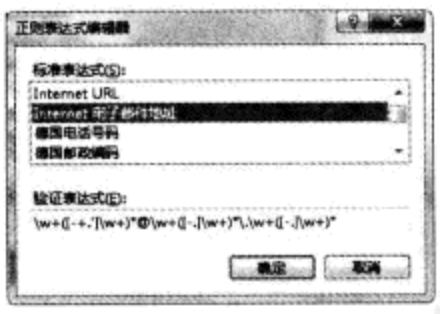


图 14.15　验证邮政编码格式

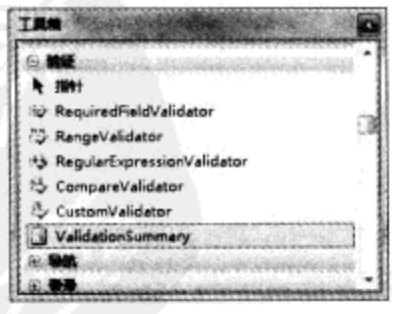


图 14.16　ValidationSummary 控件

ValidationSummary 控件的部分常用属性如表 14.13 所示。

表 14.13　ValidationSummary 控件的常用属性

属　性	描　述
HeaderText	控件汇总信息
DisplayMode	设置错误信息的显示格式
ShowMessageBox	是否以弹出方式显示每个被验证控件的错误信息
ShowSummary	是否使用错误汇总信息
EnableClientScript	是否使用客户端验证，系统默认值为 True
Validate	执行验证并且更新 IsValid 属性

学习手册 说明 ValidationSummary 控件在页面上显示的错误信息，是由每个验证控件的 ErrorMessage 属性指定的。如果没有设置验证控件的 ErrorMessage 属性，将不会在 ValidationSummary 控件中为该验证控件显示错误信息。

下面对比较重要的属性进行介绍。

（1）DisplayMode 属性

该属性获取或设置验证摘要的显示模式。

语法：

```
ValidationSummaryDisplayMode DisplayMode { get; set; }
```

属性值：ValidationSummaryDisplayMode 值之一，默认为 BulletList。ValidationSummary-DisplayMode 值是一个枚举类型值，表示指定 ValidationSummary 控件使用的验证摘要显示模式。ValidationSummaryDisplayMode 枚举值及说明如表 14.14 所示。

表 14.14　ValidationSummaryDisplayMode 枚举值及说明

枚 举 值	说　明
BulletList	显示在项目符号列表中的验证摘要
List	显示在列表中的验证摘要
SingleParagraph	显示在单个段落内的验证摘要

分别设置 ValidationSummary 控件的显示模式为 BulletList、List 和 SingleParagraph，运行结果如图 14.17 所示。

学习手册 说明 如果只希望 ValidationSummary 控件显示页面中所有验证控件的错误信息，则需要将页面中其他验证控件的 Display 属性设为 None，以防止错误信息重复出现。

（2）ShowMessageBox 属性

该属性获取或设置一个值，该值指示是否在消息框中显示验证摘要。

语法：

```
public bool ShowMessageBox { get; set; }
```

属性值：如果在消息框中显示验证摘要，则为 true；否则为 false。默认为 false。

当 ShowMessageBox 属性设为 true 时，网页上的错误信息不在网页本身上显示，而是以弹出对话框的形式来显示错误信息，如图 14.18 所示。

学习手册 注意 如果不希望错误信息在页面中显示，而是完全以弹出对话框的形式显示，

必须将 ShowSummary 属性设为 false，否则弹出对话框的同时，页面中也会显示错误信息。

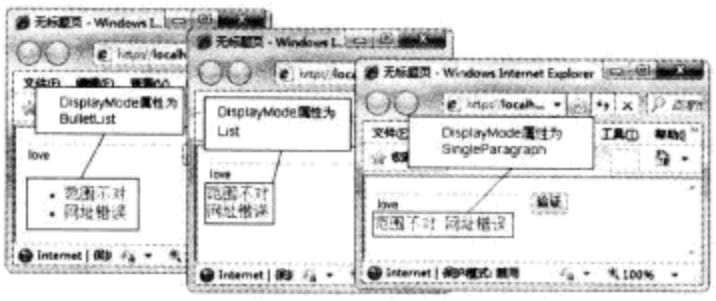


图 14.17　三种摘要显示模式

图 14.18　ShowMessageBox 设为 true，即可弹出窗口

（3）ShowSummary 属性

该属性设置是否内联显示验证摘要。

语法：

```
public bool ShowSummary { get; set; }
```

属性值：如果内联显示验证摘要，则为 true；否则为 false。默认为 true。

学习手册 注意　如果 ShowMessageBox 和 ShowSummary 属性都设置为 true，则在消息框和网页上都显示验证摘要。所以，读者可以根据需要将这两个属性中的某一个设为 false，以防止验证摘要重复出现。

14.6.2　情景应用：注册页面错误信息汇总

专题讲座：光盘\MR\Video\14\注册页面错误信息汇总.exe

视频速递：通过本视频读者可以更好地掌握 ValidationSummary 控件。

实例位置：光盘\MR\Instance\14\14.5

图 14.19　用 ValidationSummary 集中所有的报错信息，并弹出提示框

【例 14.11】本实例主要通过 ValidationSummary 控件将错误信息的摘要一起显示。执行程序，如果输入的用户名、密码为空，确认密码错误，单击“提交”按钮时，就会报错并弹出对话框，实例运行结果如图 14.19 所示。

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 3 个 TextBox 控件、两个 RequiredFieldValidator、一个 CompareValidator 控件和一个 Button 控件，它们的属性设置如表 14.15 所示。

表 14.15　Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用　途
标准 /TextBox 控件	txtName	均为默认设置	输入姓名
	txtPwd	均为默认设置	输入密码
	txtConfim	均为默认设置	输入确认密码

续表

控件类型	控件名称	主要属性设置	用　途
标准/Button 控件	btnCheck	Text 属性设置为“提交”	执行页面提交的功能
验证/ RequiredFieldValidator 控件	RequiredField-Validator1	ControlToValidate 属性设置为“txtName”	要验证的控件的 ID 为 txtName
		ErrorMessage 属性设置为“用户名不能为空!”	显示的错误信息为“用户名不能为空!”
		Text 属性设置为“*”	显示验证控件文本信息
	RequiredField-Validator2	ControlToValidate 属性设置为“txtName”	要验证的控件的 ID 为 txtPwd
		ErrorMessage 属性设置为“密码不能为空!”	显示的错误信息为“密码不能为空!”
		Text 属性设置为“*”	显示验证控件文本信息
验证/ Compare-Validator	Compare-Validator1	ControlToCompare 属性设置为“txtPwd”	所在进行验证比较的控件 ID 为 txtPwd
		ControlToValidate 属性设置为“txtConfim”	指向所要验证的控件的 ID 为 txtConfim
		ErrorMessage 属性设置为“确认密码错误!”	错误信息提示为“确认密码错误!”
		Text 属性设置为“*”	显示验证控件文本信息
验证/ Validation-Summary 控件	Validation-Summary1	ShowMessageBox 属性设置为“True”	弹出错误提示框并将错误信息一起显示

14.7　自定义验证控件

如果现有的 ASP.NET 验证控件都无法满足所需的验证功能，最后一个方法就是用 CustomValidator 控件编写程序，设计自定义的验证规则，本节将详细地介绍如何使用该控件自定义验证规则。

14.7.1　CustomValidator 控件

图 14.20　CustomValidator 控件

CustomValidator 控件为输入控件提供用户定义的验证函数。例如，可以创建一个验证控件，该控件检查在文本框中输入的值是否为偶数。Visual Studio 2008 工具箱中的 CustomValidator 控件如图 14.20 所示。

CustomValidator 控件的部分常用属性及说明如表 14.16 所示。

表 14.16　CustomValidator 控件部分常用属性

属　性	描　述
ClientValidationFunction	设置用于验证的自定义客户端脚本函数的名称
ControlToValidate	设置要验证的输入控件
Display	设置验证控件中错误信息的显示行为

续表

属　性	描　述
EnableClientScript	设置是否启用客户端验证
ErrorMessage	设置验证失败时显示的错误信息的文本
IsValid	是否通过验证
Visible	该属性获取或设置一个值，该值指示服务器控件是否作为 UI 呈现在页上

CustomValidator 控件的部分属性与 RequiredFieldValidator 控件的属性类似，详细介绍请参阅 14.2.1 节。下面介绍 CustomValidator 控件的 ClientValidationFunction 属性。

该属性设置用于验证的自定义客户端脚本函数的名称。

语法：

```
public string ClientValidationFunction { get; set; }
```

属性值：用于验证的自定义客户端脚本函数的名称。默认值为空，表示未设置此属性。

函数名不应包含任何括号或参数。

【例 14.12】使用 CustomValidator 控件的 ClientValidationFunction 属性实现客户端验证，验证输入数字是否为偶数，运行结果如图 14.21 所示。

图 14.21　CustomValidator 控件验证数字奇偶性

实现的代码如下所示：

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>ClientValidationFunction属性</title>
   <script language="javascript">
   <!--
   function ClientValidate(source, arguments)
   {
      if ((arguments.Value % 2) == 0)
         arguments.IsValid=true;
      else
         arguments.IsValid=false;
   }
   -->
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
       <table style="width: 485px">
          <tr>
             <td>
                <asp:Label ID="Label1" runat="server" Text="输入数字" Width=
                "76px"></asp: Label></td>
             <td style="width: 127px">
                <asp:TextBox ID="TextBox1" runat="server" Width="260px"></asp:
                TextBox> </td>
          </tr>
          <tr>
             <td style="width: 75px; height: 24px">
```

```
                <asp:Button ID="Button1" runat="server" OnClick="Button1_
                Click" Text="确定" Width="67px" /></td>
            <td style="width: 127px; height: 24px">
                 
                <asp:CustomValidator ID="CustomValidator1" runat="server"
                ControlToValidate= "TextBox1"
                ErrorMessage="输入的数字不是偶数" Width="256px" ClientValida-
                tionFunction= "ClientValidate"></asp:CustomValidator></td>
        </tr>
    </table>
</div>
</form>
</body>
</html>
```

除了在客户端验证外，CustomValidator 控件还可以在服务端进行验证，具体是在 CustomValidator 控件的 ServerValidate 事件中实现的，进入该事件的方法如下：

1. 选中 CustomValidator 控件，在属性窗口中单击“⚡”按钮，进入事件选项区。
2. 双击该区域中的 ServerValidate 事件，进入事件的后台代码区域。

14.7.2 情景应用：验证密码是否超出规定长度

专题讲座：光盘\MR\Video\14\验证密码是否超出规定长度.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 CustomValidator 控件。

实例位置：光盘\MR\Instance\14\14.6

【例 14.13】 开发会员注册时，如果希望通过弹出对话框提示用户输入的密码不正确，那么该如何实现呢？ASP.NET 中提供了 CustomValidator 控件，通过该控件可以实现客户端验证。本实例实现的是当用户输入的密码少于 6 位时，弹出对话框，如图 14.22 所示。

具体开发步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）向窗体中添加所需的控件，并且添加一个 CustomValidator 控件，具体添加的控件请参看源代码。

（3）在 HTML 代码中创建一个 JavaScript 函数 myValidate 用于验证输入的密码是否小于 6 位，代码如下所示：

```
<script type="text/javascript">
function myValidate()                                    //创建 JavaScript 函数
{
    var v=document.getElementById("txtPwd").value;//获取输入的密码
    if(v.length<6)                                       //如果长度小于 6 位
    {
        alert("密码至少 6 位");                          //弹出提示
    }
}
</script>
```

（4）将 CustomValidator 控件的 ClientValidationFunction 属性设为 myValidate，如图 14.23 所示。

图 14.22　自定义验证规则弹出提示窗口

图 14.23　设置 ClientValidationFunction 属性为 myValidate

14.8　实 战 练 习

14.8.1　验证出国护照

➤➤➤题目描述

在一些涉及办理出国护照的系统中，经常需要验证护照号码是否合法。本实例通过使用 RegularExpressionValidator 控件来验证注册会员输入的护照号码是否合法。运行本实例，当用户在“护照”文本框中输入不合法的护照号码时，单击“注册”按钮，程序会显示错误提示信息，如图 14.24 所示。

➤➤➤技术指导

本实例主要用到了 RegularExpressionValidator 控件，护照号码的格式有两种：P12345**（7 位数字）或 G123456**（8 位数字）。

实现验证护照号码是否合法的正则表达式如下。

```
(P\d{7})|(G\d{8})
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/14.8.1**

14.8.2　自定义出生日期的输入格式

➤➤➤题目描述

在开发会员注册页面时，注册信息中可能会要求会员输入出生日期。出生日期有规定的格式，不能随便输入。所以，对出生日期的验证至关重要。在本实例中，必须输入格式为“xxxx-xx-xx”的出生日期，否则会弹出提示信息，效果如图 14.25 所示。

➤➤➤技术指导

本实例主要是通过 RegularExpressionValidator 控件实现的，实现验证指定格式出生日期的正则表达式如下。

```
\d{4}-\d{2}-\d{2}
```

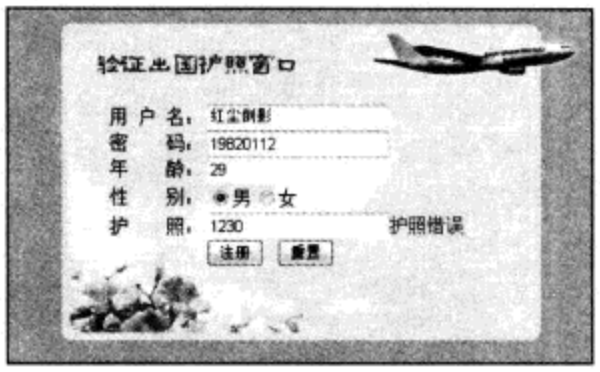


图 14.24　验证出国护照

图 14.25　验证生日输入格式

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/14.8.2**

14.8.3　验证密码格式

图 14.26　验证密码格式

➤➤➤题目描述

一般情况下，用户的密码允许特殊字符。但是，个别情况下可能会对用户注册的密码进行限制，例如，限制用户的密码以字母开头，长度在 6~18 之间，只能包含字符、数字和下画线，本实例实现的就是对用户输入的密码进行验证，效果如图 14.26 所示。

➤➤➤技术指导

本实例主要用到了 RegularExpressionValidator 控件，验证输入的密码是否用户名是否以字母开头，长度在 6~18 之间，并且包含字符、数字和下画线。实现验证密码格式的正则表达式如下。

```
^[a-zA-Z]\w{5,17}$。
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/14.8.3**

14.9　本 章 小 结

数据验证控件的出现使数据验证变得倍加轻松，很多复杂的验证直接通过验证控件的简单设置后就可以实现，这在以往的开发中都是需要编写很多代码才能实现的。所以，验证控件为程序开发者带来了福音，使程序开发中的数据验证更加安全和高效。

第15章

利用 GDI+绘制 Web 图形图像

（名师课堂：46 分钟）

在 ASP.NET Web 网站中，经常使用图形或图像来描绘一些东西，使其更形象、更直观，例如，绘制直线、矩形、多边形、椭圆或波形图等。本章将详细介绍如何在 Visual Studio 2008 开发环境中使用 GDI+绘制图形图像，通过本章的学习，读者可以达到以下目的：

- 了解什么是 GDI+
- 了解 GDI+绘图基础
- 掌握如何绘制直线
- 掌握如何绘制矩形
- 掌握如何绘制椭圆和弧
- 掌握如何绘制多边形
- 掌握如何绘制纹理效果文字
- 掌握如何绘制渐变效果文字

15.1 了解什么是 GDI+

GDI+是在 GDI（即 Windows 早期版本中附带的 Graphics Device Interface）的基础上进行了改进，添加了新功能并优化了原有功能。GDI+是 Windows XP 操作系统中提供二**维矢量图形**、图像处理和版式的部分。开发 ASP.NET Web 网站或 Windows 应用程序时，经常用到 GDI+技术，可以更形象、更直观地分析数据，例如，通过折线图分析每月的网站流量、通过柱形图分析轿车年销售情况等。

GDI+基类的主要命名空间及说明如表 15.1 所示。

表 15.1 GDI+基类的主要命名空间及说明

命 名 空 间	说　明
System.Drawing	包含与基本绘图功能有关的大多数类、结构、枚举和委托
System.Drawing.Drawing2D	为大多数高级 2D 和矢量绘图操作提供了支持，包括消除锯齿、几何转换和图形路径
System.Drawing.Imaging	帮助处理图像（位图和 GIF 文件等）的各种类
System.Drawing.Printing	把打印机或打印预览窗口作为输出设备时使用的类
System.Drawing.Design	一些预定义的对话框、属性表和其他用户界面元素，与在设计期间扩展用户界面相关
System.Drawing.Text	对字体和字体系列执行更高级操作的类

15.2 熟练掌握 GDI+绘图基础

专题讲座：光盘\MR\Video\15\ GDI+绘图基础.exe

➤➤➤视频速递：通过本视频读者可以更好地了解 GDI+绘图基础知识。

Graphics 类包含在 System.Drawing 名称空间下，Graphics 对象是用于创建图形图像的对象。创建要绘制的图形对象，需要先创建 Graphics 对象，然后才可以使用 GDI+绘制线条和形状、呈现文本或显示与操作图像。

15.2.1 创建 Graphics 对象

在 ASP.NET 中可以从任何由 Image 类派生的对象创建 Graphics 对象。通过调用 System.Drawing.Graphics.FromImage(System.Drawing.Image)方法创建 Graphics 对象，代码如下所示：

```
Bitmap bitmap = new Bitmap(80, 80);                    //创建画布
Graphics g = Graphics.FromImage(bitmap);               //在该画布上创建 Graphics 对象
```

学习手册 说明 可以形象地理解成 Bitmap 对象是一块画板，那么 Graphics 对象就是画笔。通过画笔在画板上绘画。

15.2.2　创建 Pen 对象

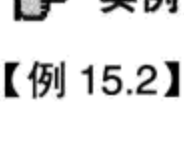
Pen 对象是用于绘制直线和曲线的对象。可以使用 DashStyle 属性绘制虚线，还可以使用各种填充样式（包括纯色和纹理）来填充 Pen 绘制的直线。填充模式取决于画笔或用做填充对象的纹理。

可以使用四种方式创建 Pen 对象，语法如下：

```
public Pen( Color color );          //用指定颜色初始化 Pen 类的新实例
public Pen( Color color, float width );
                                    //用指定的 Color 和 Width 属性初始化 Pen 类的新实例
public Pen( Brush brush );          //用指定的 Brush 初始化 Pen 类的新实例
public Pen( Brush brush, float width ); //用指定的 Brush 和 Width 初始化 Pen 类的新实例
```

【例 15.1】 读者可以根据下面的代码来理解如何创建 Pen 对象。

```
Pen pen = new Pen( Color.Black );
Pen pen = new Pen( Color.Black, 5 );
SolidBrush brush = new SolidBrush( Color.Red );
Pen pen = new Pen(brush);
Pen pen = new Pen(brush, 5 );
```



关于 SolidBrush 对象，我们将在 15.2.3 节中进行介绍，请读者关注。

15.2.3　创建 Brush 对象

画刷（Brush）是可与 Graphics 对象一起使用来创建实心形状和呈现文本的对象。可以用画刷填充各种图形形状，如矩形、椭圆、扇形、多边形和封闭路径等。Brush 类是一个抽象基类，不能进行实例化。若要创建一个画刷对象，需要使用从 Brush 派生出的类，如 SolidBrush 等。比较常用的派生类如下所示。

- ☑ SolidBrush：画刷最简单的形式，用纯色进行绘制。
- ☑ HatchBrush：类似于 SolidBrush，但是可以利用该类从大量预设的图案中选择绘制时要使用的图案，而不是纯色。
- ☑ TextureBrush：使用纹理（如图像）进行绘制。
- ☑ LinearGradientBrush：使用渐变混合的两种颜色进行绘制。
- ☑ PathGradientBrush：基于编程者定义的唯一路径，使用复杂的混合色渐变进行绘制。

为了使读者更好地掌握这几种派生出来的笔刷对象，下面分别对其进行介绍。

1．使用 SolidBrush 类定义单色画笔

SolidBrush 类用于定义单色画笔。该类只有一个构造函数，带有一个 Color 类型的参数。语法：

```
Public SolidBrush (Color);
```

Color：用于指定画笔的颜色。

实例位置：光盘\MR\Instance\15\15.1

【例 15.2】 本实例实现的是当程序运行时，在页面上绘制一个使用指定颜色填充的椭

圆。执行程序，运行结果如图 15.1 所示。

图 15.1　单色画笔的使用

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 的 Page_Load 事件中先定义一个画布，然后定义一个黄色的画刷，调用 Graphics 对象的 FillEllipse 方法在画布中绘制一个使用指定颜色填充的椭圆。最后，将填充的椭圆显示在页面上。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(200,150);                          //创建画布
    Graphics graphics = Graphics.FromImage(bitmap);               //创建 Graphics 对象
    graphics.Clear(Color.White);                                  //清空背景
    SolidBrush mySolidBrush = new SolidBrush(Color. Blue);//创建笔刷
    graphics.FillEllipse(mySolidBrush, 70, 20, 100, 50);          //绘制椭圆
    System.IO.MemoryStream ms = new System.IO.MemoryStream();  //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                                  //绘制的图像保存到内存流
    Response.ClearContent();                                      //清空缓冲区
    Response.ContentType = "image/Gif";                           //设置输出格式
    Response.BinaryWrite(ms.ToArray());                           //输出图像
}
```

2. 使用 HatchBrush 类绘制简单图案

HatchBrush 类主要使用阴影样式、前景色和背景色定义矩形画笔，而不是纯色。该类提供了两个重载的构造函数，分别是：

```
Public HatchBrush (HatchStyle,ForeColor)
Public HatchBrush (HatchStyle, ForeColor,BackColor)
```

HatchStyle：HatchStyle 的枚举成员，用于指定画笔的填充图案。

ForeColor：用于指定前景色。

BackColor：用于指定背景色。

由于 HatchStyle 的枚举成员太多，所以此处不能全部列举，请读者参看 MSDN 中的内容。

实例位置：光盘\MR\Instance\15\15.2

【例 15.3】 本实例实现的是当程序运行时，在页面上绘制一个使用简单图案填充的椭圆。执行程序，实例运行结果如图 15.2 所示。

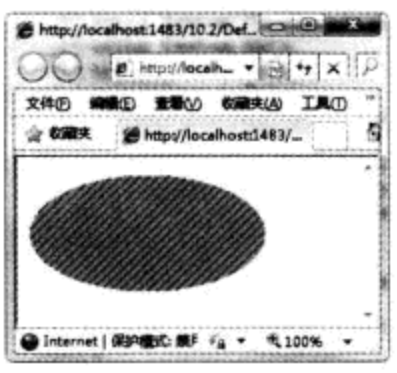


图 15.2　绘制简单图案

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 的 Page_Load 事件中先定义 Graphics 和 HatchBrush 对象，然后调用 Graphics 对象的 FillEllipse 方法在画布中绘制一个以橙色为背景色、绿色为前景色并使用斜纹填充的椭圆。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(200, 100);                         //创建画布
```

```
    Graphics graphics = Graphics.FromImage(bitmap);    //创建 Graphics 对象
    graphics.Clear(Color.White);                       //清空背景
    HatchBrush myhatchBrush = new HatchBrush(HatchStyle.BackwardDiagonal,
    Color.Green, Color. Orange);
    graphics.FillEllipse(myhatchBrush, 0, 0, 200, 100);        //绘制图形
    System.IO.MemoryStream ms = new System.IO.MemoryStream(); //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
                                                       //绘制的图像保存到内存流
    Response.ClearContent();                           //清空缓冲区
    Response.ContentType = "image/Jpeg";               //设置输出格式
    Response.BinaryWrite(ms.ToArray());                //输出图像
}
```

3. 使用 TextureBrush 类绘制复杂图案

TextureBrush 类允许使用一幅图像作为填充的样式。该类提供了 5 个重载的构造函数，分别是：

```
Public TextureBrush( Image )
Public TextureBrush( Image, Rectangle )
Public TextureBrush( Image, WrapMode )
Public TextureBrush( Image, Rectangle, ImageAttributes)
Public TextureBrush( Image, WrapMode, Rectangle)
```

TextureBrush 构造函数参数说明如表 15.2 所示。

表 15.2　TextureBrush 构造函数参数说明

参数名称	参数描述
Image	用于指定画笔的填充图案
Rectangle	用于指定图像上用于画笔的矩形区域，其位置不能超出图像的范围
WrapMode	WrapMode 枚举成员用于指定如何排布图像
ImageAttributes	用于指定图像的附加特性参数

TextureBrush 类常用属性及说明如表 15.3 所示。

表 15.3　TextureBrush 类常用属性及说明

属　性	说　明
Image	Image 类型，与画笔关联的图像对象
Transform	Matrix 类型，画笔的变换矩阵
WrapMode	WrapMode 枚举成员，指定图像的排布方式

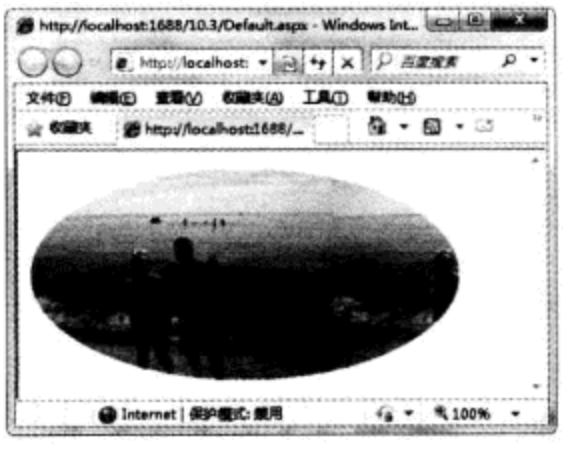


图 15.3　绘制简单图案

实例位置：光盘\MR\Instance\15\15.3

【例 15.4】 本实例实现的是当程序运行时，在页面上绘制一个使用图片填充的椭圆。执行程序，运行结果如图 15.3 所示。

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 的 Page_Load 事件中先定义 Graphics 和 TextureBrush 对象，然后调用 Graphics 对象的 FillEllipse

方法在画布中绘制一个使用图片填充的椭圆。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(400, 200);                    //创建画布
    Graphics graphics = Graphics.FromImage(bitmap);          //创建 Graphics 对象
    graphics.Clear(Color.White);                             //清空背景
     TextureBrush myTextureBrush = new TextureBrush(System.Drawing.Image.
     FromFile (Server. MapPath ("~/4.jpg")));                //用图像填充绘制的区域
    graphics.FillEllipse(myTextureBrush, 0, 0, 400, 200);    //绘制椭圆
    System.IO.MemoryStream ms = new System.IO.MemoryStream(); //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat. Jpeg);
                                                             //绘制的图像保存到内存流
    Response.ClearContent();                                 //清空缓冲区
    Response.ContentType = "image/Jpeg";                     //设置输出格式
    Response.BinaryWrite(ms.ToArray());                      //输出图像
}
```

4. 使用 LinearGradientBrush 类定义线性渐变

LinearGradientBrush 类用于定义线性渐变画笔，可以是双色渐变，也可以是多色渐变。默认情况下，渐变由起始颜色沿着水平方向均匀过渡到终止颜色，LinearGradientBrush 类的构造方法如下：

```
public LinearGradientBrush(Point, Point, Color, Color)
public LinearGradientBrush(PointF, PointF, Color, Color)
public LinearGradientBrush(Rectangle, Color, Color, LinearGradientMode)
public LinearGradientBrush(Rectangle, Color, Color, Single)
public LinearGradientBrush(RectangleF, Color, Color, LinearGradientMode)
public LinearGradientBrush(RectangleF, Color, Color, Single)
public LinearGradientBrush(Rectangle, Color, Color, Single, Boolean)
public LinearGradientBrush(RectangleF, Color, Color, Single, Boolean)
```

参数说明如表 15.4 所示。

表 15.4　LinearGradientBrush 构造函数参数说明

参 数 名 称	参 数 描 述
Point	表示线性渐变起始点的 Point 结构
Color	表示线性渐变起始色的 Color 结构
PointF	表示线性渐变起始点的 PointF 结构
Rectangle	一个 Rectangle 结构，它指定线性渐变的界限
LinearGradientMode	一个 LinearGradientMode 枚举元素，它指定渐变方向
Single	渐变方向线的角度（从 X 轴以顺时针角度计算）

下面通过实例演示如何使用 LinearGradientBrush 类绘制线性渐变图形。

实例位置：光盘\MR\Instance\15\15.4

【例 15.5】 本实例实现的是当程序运行时，在页面上绘制一个使用渐变图案填充的矩形。执行程序，运行结果如图 15.4 所示。

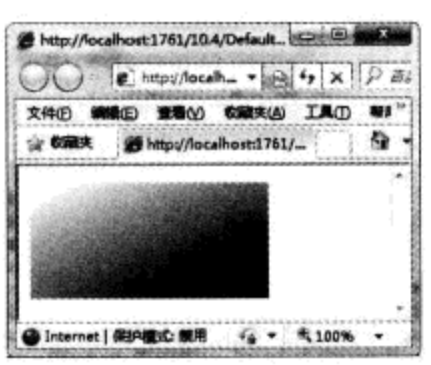


图 15.4　绘制渐变图案

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 的 Page_Load 事件中先定义 Graphics 和 Rectangle 对象，根据 Rectangle 对象创建 LinearGradientBrush 对象，然后调用 Graphics 对象的 FillRectangle 方法在画布中绘制一个使用渐变图案填充的矩形。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(200, 100);                          //创建画布
    Graphics graphics = Graphics.FromImage(bitmap);                //创建 Graphics 对象
    graphics.Clear(Color.White);                                   //清空背景
    Rectangle rectangle = new Rectangle(0, 0, 200, 100);           //设置边界
    LinearGradientBrush myLinearGradientBrush = new LinearGradientBrush
   (rectangle, Color.White, Color.Green, LinearGradientMode.ForwardDiagonal);
                                                                   //设置渐变
    graphics.FillRectangle(myLinearGradientBrush, 0, 0, 200, 100); //绘制渐变
    System.IO.MemoryStream ms = new System.IO.MemoryStream();  //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
                                                                   //绘制的图像保存到内存流
    Response.ClearContent();                                       //清空缓冲区
    Response.ContentType = "image/Jpeg";                           //设置输出格式
    Response.BinaryWrite(ms.ToArray());                            //输出图像
}
```

5. 使用 PathGradientBrush 类实现彩色渐变

在 GDI+中，把一个或多个图形组成的形体称做路径。可以使用 GraphicsPath 类定义路径，使用 PathGradientBrush 类定义路径内部的渐变色画笔。渐变色从路径内部的中心点逐渐过渡到路径的外边界边缘。它由若干个“定义路径的外围边缘”的点、一个中心点以及若干个针对每个点的颜色定义而成。颜色的渐变是从中心点向定义的每个边缘进行的，PathGradientBrush 类的构造方法如下：

```
public PathGradientBrush(GraphicsPath path)
public PathGradientBrush(Point[] points)
public PathGradientBrush(PointF[] points)
public PathGradientBrush(    Point[] points,WrapMode wrapMode)
public PathGradientBrush(PointF[] points,WrapMode wrapMode)
```

path：定义此 PathGradientBrush 填充的区域。

points：一个 Point 结构的数组，它表示构成路径顶点的点。

wrapMode：一个 WrapMode，指定使用此 PathGradientBrush 绘制的填充的平铺方式。

下面通过实例演示如何使用 PathGradientBrush 类绘制彩色渐变图形。

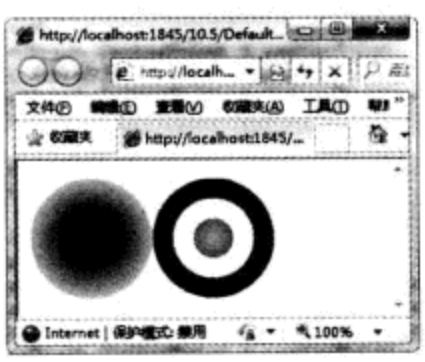


图 15.5　实现彩色渐变

实例位置：光盘\MR\Instance\15\15.5

【例 15.6】 本实例实现的是当程序运行时，在页面上使用两种不同方法绘制的由彩色渐变填充的圆形。执行程序，运行结果如图 15.5 所示。

程序实现的主要步骤说明如下。

新建一个网站，默认主页为 Default.aspx。在 Default.aspx 的 Page_Load 事件中根据已经声明的 PathGradientBrush 类对象，绘制路径渐变图形。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bit = new Bitmap(400,200);                    //创建画布
    Graphics g = Graphics.FromImage(bit);                //创建 Graphics 对象
    g.Clear(Color.White);                                //清空背景
    Point centerPoint = new Point(100, 100);             //创建 Point 对象
    int R = 100;
    GraphicsPath path = new GraphicsPath();              //创建 GraphicsPath 对象
    path.AddEllipse(centerPoint.X - R, centerPoint.Y - R, 2* R, 2 * R);
    PathGradientBrush myPathGradientBrush = new PathGradientBrush(path);
    myPathGradientBrush.CenterPoint = centerPoint;       //指定路径中心点
    myPathGradientBrush.CenterColor = Color.DarkGreen;//指定路径中心点的颜色
    //Color 类型的数组指定与路径上每个顶点对应的颜色
    myPathGradientBrush.SurroundColors = new Color[] { Color.Gold };
    g.FillEllipse(myPathGradientBrush, centerPoint.X - R, centerPoint.Y - R,2
    * R, 2 * R);
    centerPoint = new Point(300, 100);
    R = 33;
    path = new GraphicsPath();
    path.AddEllipse(centerPoint.X - R, centerPoint.Y - R, 2 * R, 2 * R);
    path.AddEllipse(centerPoint.X - 2 * R, centerPoint.Y - 2 * R, 4 * R, 4 * R);
 path.AddEllipse(centerPoint.X - 3 * R, centerPoint.Y - 3 * R, 6 * R, 6 * R);
    myPathGradientBrush = new PathGradientBrush(path);
    myPathGradientBrush.CenterPoint = centerPoint;
    myPathGradientBrush.CenterColor = Color.Gold;
    myPathGradientBrush.SurroundColors = new Color[ ]{Color.Black, Color.Blue,
    Color.DarkGreen};
    g.FillPath(myPathGradientBrush, path);
    System.IO.MemoryStream ms = new System.IO.MemoryStream();
    bit.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg );
    Response.ClearContent();
    Response.ContentType = "image/Jpeg";
    Response.BinaryWrite(ms.ToArray());
}
```

应用锦囊

1. 使用 FillPath()方法填充多个重叠图形时注意事项

当使用 FillPath()方法填充路径的时候，如果多个图形互相重叠，则重叠部分的数目为偶数时不会被填充。

2. 绘制的图片为什么会失真?

当图片显示在页面时，失真的原因不是绘图时出现了问题，而是保存图片格式时出现了问题。例如:

```
bit.Save(ms, System.Drawing.Imaging.ImageFormat.Gif );
Response.ContentType = "image/Gif";
```

这时显示的图片就会失真，所以将代码改为:

```
bit.Save(ms, System.Drawing.Imaging.ImageFormat. Jpeg);
Response.ContentType = "image/ Jpeg ";
```

15.3　使用 GDI+绘制基本图形

专题讲座：光盘\MR\Video\15\绘制基本图形.exe

➤➤➤视频速递：通过本视频读者可以更直观地了解如何绘制基本图形。

GDI+中的基本图形包括直线、矩形、椭圆、弧线、多边形、基数样条和贝塞尔样条等。Graphics 类提供的方法有：DrawLine（绘制直线）、DrawRectangle（绘制矩形）、DrawEllipse（绘制椭圆）、DrawPolygon（绘制多边形）、DrawArc（绘制弧形）、DrawCurve（绘制基数样条）和 DrawBezier（绘制贝塞尔样条）等。这些方法都是可重载的，并且可以实现不同的功能。本节将详细介绍如何通过这些方法绘制基本图形。

15.3.1　绘制直线

绘制直线主要使用 Graphics.DrawLine 方法实现，该方法是绘制一条连接两个点的线条。该方法是可重载的，重载方法如下所示。

```
public void DrawLine (Pen pen,Point pt1,Point pt2)
public void DrawLine (Pen pen,int x1,int y1,int x2,int y2)
```

DrawLine 方法中各参数的说明如表 15.5 所示。

表 15.5　DrawLine 方法参数说明

参　数	说　明
pen	Pen 对象，它确定线条的颜色、宽度和样式
pt1	Point 结构，它表示要连接的第一个点
pt2	Point 结构，它表示要连接的第二个点
x1	第一个点的 *X* 坐标
y1	第一个点的 *Y* 坐标
x2	第二个点的 *X* 坐标
y2	第二个点的 *Y* 坐标

下面通过实例演示如何通过 DrawLine 方法绘制直线。

实例位置：光盘\MR\Instance\15\15.6

【例 15.7】 本实例实现的是当程序运行的时候，在 Web 窗体中的指定位置绘制一条直线。运行结果如图 15.6 所示。

图 15.6　绘制直线

在 Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后调用 Graphics 对象的 DrawLine 方法在 Web 窗体中绘制一条直线。代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(200, 150);                    //创建画布
    Graphics g = Graphics.FromImage(bitmap);                 //创建 Graphics 对象
```

```
    g.Clear(Color.WhiteSmoke);                                    //清空背景
    Pen myPen = new Pen(Color.Blue, 2);                           //创建 Pen 对象
    g.DrawLine(myPen, 20, 30, 120, 30);                           //绘制直线
    System.IO.MemoryStream ms = new System.IO.MemoryStream();  //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                                  //绘制的图像保存到内存流
    Response.ClearContent();                                      //清空缓冲区
    Response.ContentType = "image/Gif";                           //设置输出格式
    Response.BinaryWrite(ms.ToArray());                           //输出图像
}
```

15.3.2 绘制矩形

绘制矩形时，可以调用 Graphics 类中的 DrawRectangle 方法，该方法可重载，它主要用来绘制由坐标对、宽度和高度指定的矩形，其常用格式有以下两种：

（1）绘制由 Rectangle 结构指定的矩形。

语法：

```
public void DrawRectangle (Pen pen,Rectangle rect)
```

pen：Pen 对象，它确定矩形的颜色、宽度和样式。

rect：表示要绘制矩形的 Rectangle 结构。

【例 15.8】 下面的代码用来声明一个 Rectangle 结构：

```
Rectangle rect = new Rectangle(0, 0, 80, 50);
```

（2）绘制由坐标对、宽度和高度指定的矩形。

语法：

```
public void DrawRectangle (Pen pen,int x,int y,int width,int height)
```

DrawRectangle 方法中各参数的说明如表 15.6 所示。

表 15.6 DrawRectangle 方法参数说明

参　数	说　明
pen	Pen 对象，它确定矩形的颜色、宽度和样式
x	要绘制矩形的左上角的 *X* 坐标
y	要绘制矩形的左上角的 *Y* 坐标
width	要绘制矩形的宽度
height	要绘制矩形的高度

实例位置：光盘\MR\Instance\15\15.7

【例 15.9】 本实例实现的是当程序运行的时候，在 Web 窗体中的指定位置绘制一个指定大小的矩形。运行结果如图 15.7 所示。

图 15.7 绘制矩形

Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后调用 Graphics 对象的 DrawRectangle 方法在 Web 窗体中绘制一个指定大小的矩形，代码如下所示。

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

```
{
    Bitmap bitmap = new Bitmap(200, 150);                    //创建画布
    Graphics g = Graphics.FromImage(bitmap);                 //创建 Graphics 对象
    g.Clear(Color.WhiteSmoke);                               //清空背景
    Pen myPen = new Pen(Color.Blue, 2);                      //创建 Pen 对象
    g.DrawRectangle(myPen, 20, 50, 120, 50);                 //绘制矩形
    System.IO.MemoryStream ms = new System.IO.MemoryStream(); //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                             //绘制的图像保存到内存流
    Response.ClearContent();                                 //清空缓冲区
    Response.ContentType = "image/Gif";                      //设置输出格式
    Response.BinaryWrite(ms.ToArray());                      //输出图像
}
```

15.3.3　绘制椭圆和弧

1. 绘制椭圆

绘制椭圆时，可以调用 Graphics 类中的 DrawEllipse 方法，该方法可重载，它主要用来绘制边界由 Rectangle 结构指定的椭圆，其常用格式有以下两种。

（1）绘制边界由 Rectangle 结构指定的椭圆。

语法：

```
public void DrawEllipse (Pen pen,Rectangle rect)
```

pen：Pen 对象，它确定曲线的颜色、宽度和样式。

rect：Rectangle 结构，它定义椭圆的边界。

（2）绘制一个由边框（该边框由一对坐标、高度和宽度指定）定义的椭圆。

语法：

```
public void DrawEllipse (Pen pen,int x,int y,int width,int height)
```

DrawEllipse 方法中各参数的说明如表 15.7 所示。

表 15.7　DrawEllipse 方法参数说明

参　数	说　明
pen	Pen 对象，它确定曲线的颜色、宽度和样式
x	定义椭圆边框的左上角的 *X* 坐标
y	定义椭圆边框的左上角的 *Y* 坐标
width	定义椭圆边框的宽度
height	定义椭圆边框的高度

实例位置：光盘\MR\Instance\15\15.8

【例 15.10】 本实例实现的是当程序运行的时候，在 Web 窗体中的指定位置绘制一个指定边框大小的椭圆。运行结果如图 15.8 所示。

Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后定义一个 Rectangle 结构，用来控制椭圆的边框大小，最后调用 Graphics 对象的 DrawEllipse 方法在 Web 窗体中绘制一个指定边框大小的椭圆，代码如下所示。

图 15.8　绘制椭圆

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(200, 150);                          //创建画布
    Graphics g = Graphics.FromImage(bitmap);                       //创建 Graphics 对象
    g.Clear(Color.WhiteSmoke);                                     //清空背景
    Pen myPen = new Pen(Color.Blue, 2);                            //创建 Pen 对象
    Rectangle myRectangle = new Rectangle(20, 50, 120, 50);        //定制边界
    g.DrawEllipse(myPen,myRectangle);                              //绘制椭圆
    // 或 g.DrawEllipse(myPen, 20, 50, 120, 50);
    System.IO.MemoryStream ms = new System.IO.MemoryStream();  //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                                   //绘制的图像保存到内存流
    Response.ClearContent();                                       //清空缓冲区
    Response.ContentType = "image/Gif";                            //设置输出格式
    Response.BinaryWrite(ms.ToArray());                            //输出图像
}
```

2. 绘制圆弧

绘制圆弧时，可以调用 Graphics 类中的 DrawArc 方法，该方法可重载，它主要用来绘制一段弧线，其常用格式有以下两种。

（1）绘制一段弧线，它表示由 Rectangle 结构指定的椭圆的一部分。

语法：

```
public void DrawArc (Pen pen,Rectangle rect,float startAngle,float sweepAngle)
```

DrawArc 方法中各参数的说明如表 15.8 所示。

表 15.8　DrawArc 方法参数说明

参　数	说　明
pen	Pen 对象，它确定弧线的颜色、宽度和样式
rect	Rectangle 结构，它定义椭圆的边界
startAngle	从 *X* 轴到弧线的起始点沿顺时针方向度量的角（以度为单位）
sweepAngle	从 startAngle 参数到弧线的结束点沿顺时针方向度量的角（以度为单位）

（2）绘制一段弧线，它表示由一对坐标、宽度和高度指定的椭圆部分。

语法：

```
public void DrawArc (Pen pen,int x,int y,int width,int height,int startAngle,int
sweepAngle)
```

DrawArc 方法中各参数的说明如表 15.9 所示。

表 15.9　DrawArc 方法参数说明

参　数	说　明
pen	Pen 对象，它确定弧线的颜色、宽度和样式
x	定义椭圆边框的左上角的 *X* 坐标
y	定义椭圆边框的左上角的 *Y* 坐标
width	定义椭圆边框的宽度
height	定义椭圆边框的高度
startAngle	从 *X* 轴到弧线的起始点沿顺时针方向度量的角（以度为单位）
sweepAngle	从 startAngle 参数到弧线的结束点沿顺时针方向度量的角（以度为单位）

实例位置：光盘\MR\Instance\15\15.9

【例 15.11】 本实例实现的是当程序运行的时候，在 Web 窗体中的指定位置绘制一段指定长度的圆弧。运行结果如图 15.9 所示。

Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后定义一个 Rectangle 结构，用来控制椭圆的边框大小，最后调用 Graphics 对象的 DrawArc 方法在指定的边框区域内绘制一段圆弧，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(200, 150);                    //创建画布
    Graphics g = Graphics.FromImage(bitmap);                 //创建 Graphics 对象
    g.Clear(Color.WhiteSmoke);                               //清空背景
    Pen myPen = new Pen(Color.Blue, 2);                      //创建 Pen 对象
    Rectangle myRectangle = new Rectangle(70, 20, 100, 60);    //设置边界
    g.DrawArc(myPen, myRectangle, 0, 135);                   //绘制圆弧
    System.IO.MemoryStream ms = new System.IO.MemoryStream(); //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                             //绘制的图像保存到内存流
    Response.ClearContent();                                 //清空缓冲区
    Response.ContentType = "image/Gif";                      //设置输出格式
    Response.BinaryWrite(ms.ToArray());                      //输出图像
}
```

15.3.4　绘制多边形

绘制多边形，需要 Graphics 对象、Pen 对象和 Point 对象数组。Graphics 对象提供 DrawPolygon 方法，Pen 对象存储用于呈现多边形的线条属性，例如宽度和颜色等，Point 对象数组存储多边形的各个顶点。Pen 对象和 Point 对象数组作为参数传递给 DrawPolygon 方法，其语法格式如下：

```
public void DrawPolygon (Pen pen,Point[] points)
```

pen：Pen 对象，它确定多边形的颜色、宽度和样式。

points：Point 结构数组，这些结构表示多边形的顶点。

实例位置：光盘\MR\Instance\15\15.10

【例 15.12】 本实例实现的是当程序运行时，在 Web 窗体中绘制一个多边形，运行结果如图 15.10 所示。

图 15.9　绘制圆弧

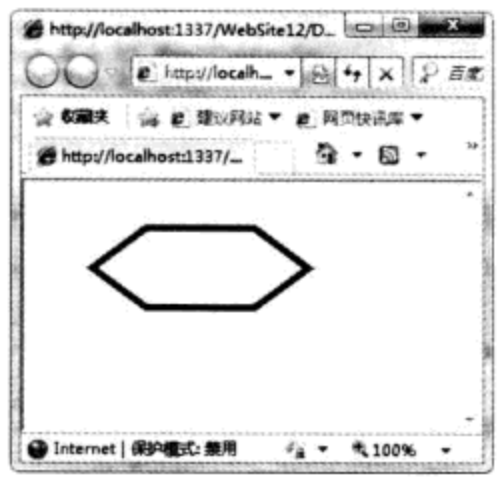


图 15.10　绘制多边形

Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后声明一个 Point 结构数组，实例化 6 个 Point 对象，用来作为多边形的顶点，最后调用 Graphics 对象的 DrawPolygon 方法绘制一个多边形，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(300, 150);                    //创建画布
    Graphics g = Graphics.FromImage(bitmap);                 //创建 Graphics 对象
    g.Clear(Color.WhiteSmoke);                               //清空背景
    Pen myPen = new Pen(Color.Blue, 5);                      //创建 Pen 对象
    Point[] myPoints ={ new Point(80, 20), new Point(40, 50), new Point(80, 80),
    new Point(160, 80), new Point(200, 50), new Point(160, 20) };
                                                             //设置各个点的坐标
    g.DrawPolygon(myPen, myPoints);                          //开始绘制多边形
    System.IO.MemoryStream ms = new System.IO.MemoryStream(); //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                             //绘制的图像保存到内存流
    Response.ClearContent();                                 //清空缓冲区
    Response.ContentType = "image/Gif";                      //设置输出格式
    Response.BinaryWrite(ms.ToArray());                      //输出图像
}
```

15.3.5　绘制基数样条

基数样条是一连串单独的曲线，这些曲线连接起来形成一条较长的曲线。基数样条由点的数组和张力参数指定，样条平滑地经过数组中的每个点，曲线的陡度上没有尖角和突然的变化。

绘制基数样条，需要 Graphics 对象、Pen 对象和 Point（或 PointF）对象数组。Graphics 对象提供了 DrawCurve 方法用于绘制基数样条，Pen 对象存储基数样条的属性（如线宽和颜色等），Point（或 PointF）对象数组存储曲线将要经过的点。DrawCurve 方法为可重载方法，其常用格式有以下 4 种。

（1）绘制由一组 Point 结构指定的基数样条。

语法：

```
public void DrawCurve (Pen pen,Point[] points)
```

pen：Pen 对象，它确定曲线的颜色、宽度和高度。

points：Point 结构数组，这些结构定义样条。

（2）使用指定的张力，根据指定的 Point 结构绘制一条基数样条。
语法：

```
public void DrawCurve (Pen pen,Point[] points,float tension)
```

pen：Pen 对象，它确定曲线的颜色、宽度和高度。
points：Point 结构数组，这些结构定义样条。
tension：大于或等于 0.0F 的值，该值指定曲线的张力。
（3）从相对于数组开始位置的偏移量开始，绘制经过一组指定 PointF 结构的基数样条。
语法：

```
public void DrawCurve (Pen pen,PointF[] points,int offset,int numberOfSegments)
```

DrawCurve 方法中各参数的说明如表 15.10 所示。

表 15.10　DrawCurve 方法参数说明

参　数	说　明
pen	Pen 对象，它确定曲线的颜色、宽度和高度
points	PointF 结构数组，这些结构定义样条
offset	从 points 参数数组中的第一个元素到曲线中起始点的偏移量
numberOfSegments	起始点之后要包含在曲线中的段数

（4）使用指定的张力，绘制经过一组指定 Point 结构的基数样条。
语法：

```
public void DrawCurve (Pen pen,Point[] points,int offset,int numberOfSegments,
float tension)
```

DrawCurve 方法中各参数的说明如表 15.11 所示。

表 15.11　DrawCurve 方法参数说明

参　数	说　明
pen	Pen 对象，它确定曲线的颜色、宽度和高度
points	Point 结构数组，这些结构定义样条
offset	从 points 参数数组中的第一个元素到曲线中起始点的偏移量
numberOfSegments	起始点之后要包含在曲线中的段数
tension	大于或等于 0.0F 的值，该值指定曲线的张力

实例位置：光盘\MR\Instance\15\15.11

【例 15.13】 本实例实现的是当程序运行的时候，在 Web 窗体中使用指定的张力，根据指定的 Point 结构绘制一条基数样条，运行结果如图 15.11 所示。

在 Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后声明一个 Point 结构数组，实例化 6 个 Point 对象，最后调用 Graphics 对象的 DrawCurve 方法绘制一条经过 Point 结构的基数样条，代码如下所示。

图 15.11　绘制基数样条

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(300, 150);                    //创建画布
    Graphics g = Graphics.FromImage(bitmap);                 //创建 Graphics 对象
    g.Clear(Color.WhiteSmoke);                               //清空背景
    Pen myPen = new Pen(Color.Blue, 5);                      //创建 Pen 对象
    Point[] myPoints ={ new Point(80, 20), new Point(40, 50), new Point(80, 80),
    new Point(160, 80), new Point(200, 50), new Point(160, 20) }; //设置各点坐标
    g.DrawCurve(myPen, myPoints, 1.0F);                      //开始绘制基数样条
    System.IO.MemoryStream ms = new System.IO.MemoryStream();     //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                             //绘制的图像保存到内存流
    Response.ClearContent();                                 //清空缓冲区
    Response.ContentType = "image/Gif";                      //设置输出格式
    Response.BinaryWrite(ms.ToArray());                      //输出图像
}
```

15.3.6 绘制贝塞尔样条

贝塞尔样条是由 4 个点指定的曲线：两个端点（p1 和 p2）和两个控制点（c1 和 c2）。曲线开始于 p1，结束于 p2。该曲线不经过控制点，但是控制点的作用像磁铁一样，在某些方向上拉伸曲线并影响曲线弯曲的方式。

绘制贝塞尔样条时，可以调用 Graphics 类中的 DrawBezier 方法，该方法可重载，它主要用来绘制贝塞尔样条，其常用格式有以下两种。

（1）绘制由 4 个 Point 结构定义的贝塞尔样条。

语法：

```
public void DrawBezier (Pen pen,Point pt1,Point pt2,Point pt3,Point pt4)
```

DrawBezier 方法中各参数的说明如表 15.12 所示。

表 15.12 DrawBezier 方法参数说明

参　数	说　明
pen	Pen 对象，它确定曲线的颜色、宽度和样式
pt1	Point 结构，它表示曲线的起始点
pt2	Point 结构，它表示曲线的第一个控制点
pt3	Point 结构，它表示曲线的第二个控制点
pt4	Point 结构，它表示曲线的结束点

（2）绘制由 4 个表示点的有序坐标对定义的贝塞尔样条。

语法：

```
public void DrawBezier (Pen pen,float x1,float y1,float x2,float y2,float
x3,float y3,float x4,float y4)
```

DrawBezier 方法中各参数的说明如表 15.13 所示。

表 15.13 DrawBezier 方法参数说明

参　数	说　明
pen	Pen 对象，它确定曲线的颜色、宽度和样式
x1	曲线起始点的 *X* 坐标

续表

参　数	说　明
y1	曲线起始点的 *Y* 坐标
x2	曲线的第一个控制点的 *X* 坐标
y2	曲线的第一个控制点的 *Y* 坐标
x3	曲线的第二个控制点的 *X* 坐标
y3	曲线的第二个控制点的 *Y* 坐标
x4	曲线的结束点的 *X* 坐标
y4	曲线的结束点的 *Y* 坐标

实例位置：光盘\MR\Instance\15\15.12

【例 15.14】本实例实现的是当程序运行的时候，在 Web 窗体中的指定位置绘制一段贝塞尔样条，运行结果如图 15.12 所示。

图 15.12　绘制贝塞尔样条

在 Web 窗体的加载事件中分别声明 Graphics 类和 Pen 类的两个实例对象，然后定义 6 个 float 类型的变量，分别用来表示贝塞尔样条起始点的 *X* 坐标、*Y* 坐标、第一个控制点的 *X* 坐标及 *Y* 坐标、第二个点的 *X* 坐标及 *Y* 坐标、结束点的 *X* 坐标及 *Y* 坐标，最后调用 Graphics 对象的 DrawBezier 方法在 Web 窗体中绘制贝塞尔样条，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bitmap = new Bitmap(300, 150);          //创建画布
    Graphics g = Graphics.FromImage(bitmap);       //创建 Graphics 对象
    g.Clear(Color.WhiteSmoke);                     //清空背景
    Pen myPen = new Pen(Color.Blue, 5);            //创建 Pen 对象
    float startX = 50.0F;                          //设置坐标
    float startY = 80.0F;
    float controlX1 = 150.0F;
    float controlY1 = 20.0F;
    float controlX2 = 230.0F;
    float controlY2 = 50.0F;
    float endX = 190.0F;
    float endY = 80.0F;
    g.DrawBezier(myPen, startX, startY, controlX1, controlY1, controlX2,
    controlY2, endX, endY);
    System.IO.MemoryStream ms = new System.IO.MemoryStream();  //创建内存流
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                                                   //绘制的图像保存到内存流
    Response.ClearContent();                       //清空缓冲区
    Response.ContentType = "image/Gif";            //设置输出格式
    Response.BinaryWrite(ms.ToArray());            //输出图像
}
```

15.3.7　情景应用 1：纹理效果的文字

实例位置：光盘\MR\Instance\15\15.13

【例 15.15】 纹理效果的文字就是在绘制文字时，将指定图片绘制到文字上，使文字

ÓÉ Foxit PDF Editor ±à¼-
°æÈ¨ËùÓÐ (c) by Foxit Software Company, 2004
½öÓÃÓÚÆÀ¹À¡£

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ⇕

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 ›

HTTP://WLSAM168.400GB.COM

和图片融合形成一种美轮美奂的纹理效果。运行程序，绘制具有纹理效果的文字。实例运行结果如图 15.13 所示。

图 15.13　为文字填充图片纹理

程序开发步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页的 Page_Load 事件中通过使用 Graphics 对象的 DrawString 方法绘制文字。使用 TextureBrush 类实现为文字填充图片纹理效果，关键代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    string str="红尘倒影";
    Bitmap bt = new Bitmap(350, 200);
    Graphics g = Graphics.FromImage(bt);                          //实例化 Graphics 类
    g.Clear(Color.White);
    //使用图像填充文字线条
    TextureBrush brush = new TextureBrush(System.Drawing.Image.FromFile(Server.
    MapPath ("test0.jpg")));
    g.DrawString(str, new Font("隶书", 60). brush, new PointF(0, 0));
                                                                  //绘制文字
    System.IO.MemoryStream ms = new System.IO.MemoryStream();  //创建内存流
    bt.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);   //将图片存储内存流
    Response.ClearContent();
    Response.ContentType = "image/Gif";                           //设置输出类型
    Response.BinaryWrite(ms.ToArray());                           //输出图片
}
```

15.3.8　情景应用 2：渐变效果的文字

实例位置：光盘\MR\Instance\15\15.14

【例 15.16】 渐变效果的文字就是在绘制文字时，使文字的前景色具有渐变的效果。运行程序，绘制具有渐变颜色的文字。实例运行结果如图 15.14 所示。

程序开发步骤说明如下。

图 15.14　渐变效果的文字

（1）新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页的 Page_Load 事件中通过使用 LinearGradientBrush 类来实现文字的渐变效果，并且通过 LinearGradientMode 枚举成员设置渐变的方向，关键代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Bitmap bt = new Bitmap(300, 200);
    Graphics g = Graphics.FromImage(bt);                          //实例化 Graphics 类
    g.Clear(Color.White);                                         //以指定的颜色清除控件背景
    Color Var_Color_Up = Color.Red;                               //设置前景色
    Color Var_Color_Down = Color.Yellow;                          //设置背景色
    Font Var_Font = new Font("宋体", 40);                         //设置字体样式
    string Var_Str = "红尘倒影";                                  //设置字符串
    SizeF Var_Size = g.MeasureString(Var_Str, Var_Font);   //获取字符串的大小
```

```
    PointF Var_Point = new PointF(5, 5);                    //设置文字的显示位置
    RectangleF Var_Rect = new RectangleF(Var_Point, Var_Size);
                                        //根据文字的大小及位置，实例化 RectangleF 类
        LinearGradientBrush Var_LinearBrush = new LinearGradientBrush(Var_Rect,
        Var_Color_Up, Var_Color_Down,
LinearGradientMode.Horizontal);        //设置从左到右的线性渐变效果
    g.DrawString(Var_Str, Var_Font, Var_LinearBrush, Var_Point);   //绘制文字
    System.IO.MemoryStream ms = new System.IO.MemoryStream();
    bt.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
    Response.ClearContent();
    Response.ContentType = "image/Gif";
    Response.BinaryWrite(ms.ToArray());
}
```

15.4　实 战 练 习

15.4.1　绘制公章

➤➤➤题目描述

在中小型企业中，公章的应用是非常普遍的，它代表了一个企业的身份，本实例将用 GDI+技术绘制一个简单的公章。运行结果如图 15.15 所示。

图 15.15　绘制公章

➤➤➤技术指导

利用 GDI+绘图时，最重要的是调用类库中的 Graphics 类，利用该类可以完成各种绘图操作。本实例中使用 Graphics 类对象的 DrawString 方法、DrawRectangle 方法和 MeasureString 方法，它们分别用于在画布中绘制字符串、椭圆并对字符串进行测量。下面分别介绍这几种方法。

（1）DrawString 方法

绘制字符串是用 GDI+技术的 DrawString 方法来实现的。它有 4 种语法。其语法格式如下：

```
    public void DrawString(string s, Font font, Brush brush, PointF point)
    public void DrawString(string s, Font font, Brush brush, RectangleF layoutRectangle)
    public void DrawString(string s, Font font, Brush brush, PointF point, String
Format format)
    public void DrawString(string s, Font font, Brush brush, RectangleF layout
Rectangle, StringFormat format)
```

DrawString 方法中各参数的说明如表 15.14 所示。

表 15.14　DrawString 方法参数说明

参　数	说　明
s	要绘制的字符串
font	Font，它定义字符串的文本格式
brush	Brush，它确定所绘制文本的颜色和纹理
point	PointF 结构，它指定所绘制文本的左上角
layoutRectangle	RectangleF 结构，它指定所绘制文本的位置
format	StringFormat，它指定应用于所绘制文本的格式化属性（如行距和对齐方式）

（2）DrawEllipse 方法

该方法用于绘制一个椭圆形。其语法格式如下：

```
public void DrawEllipse (Pen pen,    Rectangle rect)
```

pen：Pen 对象，它确定曲线的颜色、宽度和样式。

rect：Rectangle 结构，它定义椭圆的边界。

（3）MeasureString 方法

该方法用于对指定的字符串进行测量。其语法格式如下：

```
public SizeF MeasureString (string text,Font font)
```

text：要测量的字符串。

rect：Font，它定义字符串的文本格式。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/15.4.1**

15.4.2 波形图的绘制

➤➤➤题目描述

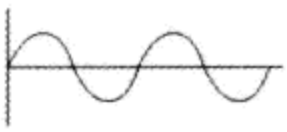


图 15.16 波形图的绘制

波形图是一种特殊的图形，是按照特定的规律绘制出的曲线，在许多工程或有关计算方面的软件中都会用到这类图形。本实例设计了一个波形图绘图软件，运行程序，单击“绘图”按钮即可在窗体中绘制出一段波形图。实例运行结果如图 15.16 所示。

➤➤➤技术指导

绘制波形图时，可以通过使用 Graphics 类对象的 DrawBezier 方法实现，具体请参见 15.3.6 节的内容。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/15.4.2**

15.4.3 倒影效果的文字

➤➤➤题目描述

倒影效果文字就是在文字的下面显示其倒影效果。运行程序，绘制指定的文字，并在文字的下面绘制其倒影效果。实例运行结果如图 15.17 所示。

图 15.17 倒影效果的文字

➤➤➤技术指导

本实例主要是通过使用 Graphics 对象的 MeasureString 方法和 ScaleTransform 方法来实现倒影文字的效果，下面分别进行详细介绍。

（1）MeasureString 方法

该方法用于测量指定的 Font 格式绘制的字符串。其语法格式如下：

```
public SizeF MeasureString (string text,Font font)
```

text：要测量的字符串。

font：Font，它定义字符串的文本格式。

（2）ScaleTransform 方法

该方法将指定的缩放操作应用于 Graphics 对象的变换矩阵，方法是将该对象的变换矩阵左乘该缩放矩阵。其语法格式如下：

```
public void ScaleTransform (float sx,float sy)
```

sx：*X* 方向的比例因子。

sy：*Y* 方向的比例因子。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/15.4.3**

15.5　本 章 小 结

本章主要介绍了 GDI+相关的技术，通过对本章的学习可以对 Graphics 对象、Brush 对象和 Brush 对象等进行了解，并且可以掌握直线、矩形、椭圆和多边形等多种图形的绘制方法。最后，读者还可以根据本章的学习，尝试绘制验证码。GDI+技术在 ASP.NET 中占有比较重要的地位，希望读者能够认真学习本章，以便为以后大型网站的开发打下坚实的基础。

第16章

AJAX 无刷新技术

（名师课堂：54 分钟）

学习 AJAX 无刷新模式开发技术，第一步了解 AJAX 概念及运行原理，并创建一个 AJAX 空网站；第二步熟悉并学习使用 AJAX 服务器控件。AJAX 服务器控件的使用在 AJAX 项目开发过程中很重要，通过本章的学习，读者可以达到以下目的：

- 熟悉 AJAX 的概念
- 了解 AJAX 的优点
- 掌握 AJAX 的运行原理
- 熟悉 AJAX 的标准服务器控件
- 掌握如何添加 AJAX Control Toolkit 控件
- 掌握创建 AJAX 网站

16.1　初次体验 ASP. NET AJAX 技术

专题讲座：光盘\MR\Video\16\ ASP. NET AJAX 初体验.exe

➤➤➤视频速递：通过本视频读者可以更好地了解 AJAX 相关的内容。

AJAX 是 Asynchronous JavaScript and XML（异步 JavaScript 和 XML 技术）的缩写，它是由 JavaScript 脚本语言、CSS 样式表、XMLHttpRequest 数据交换对象和 DOM 文档对象（或 XMLDOM 文档对象）等多种技术组成的。微软在 ASP. NET 框架基础上，创建了 ASP.NET AJAX 技术，能够实现 AJAX 功能。ASP.NET AJAX 技术被整合在 ASP. NET 2.0 及以上版本中，是 ASP.NET 的一种扩展技术。

16.1.1　AJAX 开发模式

在传统的 Web 应用模式中，页面中用户的每一次操作都将触发一次返回 Web 服务器的 HTTP 请求，服务器进行相应的处理（获得数据、运行与不同的系统会话）后，返回一个 HTML 页面给客户端，如图 16.1 所示。

而在 AJAX 应用中，页面中用户的操作将通过 AJAX 引擎与服务器端进行通信，然后将返回结果提交给客户端页面的 AJAX 引擎，再由 AJAX 引擎来决定将这些数据显示到页面的指定位置，如图 16.2 所示。

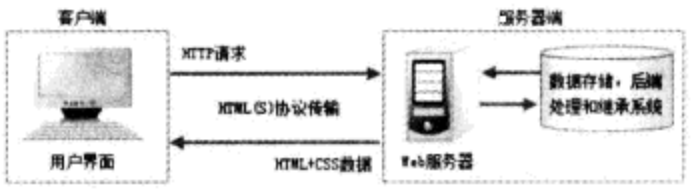


图 16.1　Web 应用的传统模型

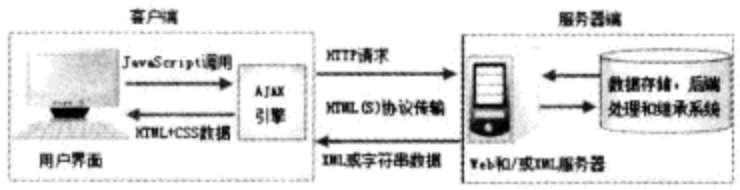


图 16.2　Web 应用的 Ajax 模型

学习手册 说明　从图 16.1 和图 16.2 中可以看出，对于每个用户的行为，传统的 Web 应用模型中将生成一次 HTTP 请求，而在 AJAX 应用开发模型中，将变成对 AJAX 引擎的一次 JavaScript 调用。在 AJAX 应用开发模型中通过 JavaScript 实现在不刷新整个页面的情况下，对部分数据进行更新，从而降低了网络流量，给用户带来了更好的体验。

16.1.2　ASP.NET AJAX 的优点

ASP.NET AJAX 可以提供 ASP.NET 无法提供的几个功能，或者弥补其不尽如人意的地方。

- ☑ 改善用户操作体验，不会动不动因 PostBack 使整页重新加载造成闪动。
- ☑ 实现 Web 页面的局部更新，不整页更新。
- ☑ 异步取回服务器端的数据，用户不会被限制于等待状态，也不会打断用户的操作，从而加快了响应能力。
- ☑ 提供跨浏览器的兼容性支持，ASP.NET AJAX 的 JavaScript 是跨浏览器的，不限

定只有 IE 才能支持。

☑ 大量内建的客户端控件，更方便实现 JavaScript 功能以及特效。

☑ 只要在 Visual Studio 2008 中“添加新项”选择“Web 配置文件”，便可通过 web.config 配置，启用 ASP.NET AJAX 相关的设置。

学习手册 说明 在配置文件中启用了 ASP.NET AJAX 相关配置后，用户可以如往常一样在 web.config 配置文件中添加设置，如数据库连接字符串等，只要不更改或删除 ASP.NET AJAX 的设置就可保所有设置皆相安无事，互不干扰。

16.1.3 探讨 ASP.NET AJAX 架构

ASP.NET AJAX 的架构横跨了客户端与服务器端，非常适合用来创建操作方式更便利、反应更快速的跨浏览器页面应用程序，下面分别给予简单介绍。

1. ASP. NET AJAX 服务器端架构

ASP. NET AJAX 是建立于 ASP. NET 框架之上的，ASP. NET AJAX 服务器端架构主要包括以下 4 个部分：

☑ ASP. NET AJAX 服务器端控件；

☑ ASP. NET AJAX 服务器端扩展控件；

☑ ASP. NET AJAX 服务器端远程 Web Service 桥；

☑ ASP. NET Web 程序的客户端代理。

学习手册 提示 ASP. NET AJAX 的服务器端控件主要是为开发者提供一种熟悉的、与 ASP. NET 一致的服务器端编程模型。事实上，这些服务器端控件在运行时会自动生成 ASP. NET AJAX 客户端组件，并发送给客户端浏览器执行。

2. ASP. NET AJAX 客户端架构

ASP. NET AJAX 客户端架构主要包括应用程序接口、API 函数、基础类库、封装的 XMLHttpRequest 对象、ASP. NET AJAX XML 引擎、ASP. NET AJAX 的客户端控件等。ASP. NET AJAX 的客户端控件在浏览器上运行，提供管理界面元素、调用服务器端方法获取数据等功能。

16.2 ASP. NET AJAX 服务器控件

专题讲座：光盘\MR\Video\16\ ASP. NET AJAX 服务器控件.exe

➤➤➤视频速递：通过本视频读者可以更好地掌握 AJAX 服务器控件。

通过 16.1 节读者已经对 AJAX 的概念有所了解，明白了 AJAX 控件的优点和架构。那么，从本节开始，笔者将带领读者踏进 AJAX 的奇妙世界，了解 AJAX 的几个重要服务器控件。

16.2.1　ScriptManager 脚本管理控件

ScriptManager 控件是服务器端 ASP.NET AJAX 模型的大脑，它是一个在页面上没有任何可视界面的 Web 控件。但是，它执行一个主要的任务——它呈现到 ASP.NET AJAX JavaScript 库的链接。

用户可以将 ScriptManager 控件视为 AJAX 脚本管理员，有了该管理员才能够让 Page 局部更新起作用，浏览器所需要的 JavaScript 才会自动管理。因此开发 AJAX 网站时，每个页面中必须添加一个 ScriptManager 控件，以便可以局部更新网页中的数据，并与服务器的程序沟通。ScriptManger 控件如图 16.3 所示。

图 16.3　每个 ASP.NET AJAX 网页必须有且仅有一个 ScriptManager 控件

学习手册 说明　要实现 AJAX 功能，在 ASP. NET 页面中需要包含一个（且只有一个）ScriptManger 控件的声明。ScriptManger 控件必须放置在依赖于 ASP. NET AJAX 控件或脚本的前面，否则页面将发生脚本错误。正因为这样，ScriptManger 控件一般放置在 Web 窗体页的 Form 元素之后。

ScriptManger 控件的常用属性如表 16.1 所示。

表 16.1　ScriptManger 控件的常用属性

属　性	说　明
EnablePageMethods	返回或设置一个 bool 值，默认值为 false，表示在客户端 JavaScript 代码中是否以一种简单、直观的形式直接调用服务器端的某个静态 Web Method
EnablePartialRendering	返回或设置一个 bool 值，默认值为 true，表示 AJAX 允许改变原有的 ASP.NET 回送模式，不再是整个页面的回送，而是只回送页面中的一部分
EnableScriptComponents	用于设置是否传送除了 AJAX 核心以外的其他组件，包括客户端控件、数据绑定、XML 声明式 Script、用户接口组件
Scripts	用于取得 ScriptReference 对象的集合，ScriptReference 对象的集合通过 AJAX 将用户的 Script 文件送到客户端进行对象引用
Services	用于取得一个 ServiceRefence 对象的集合，ServiceRefence 对象的集合通过 AJAX 为每个 Web Service 在客户端公开一个 Proxy 对象引用。

学习手册 说明　从“工具箱”的“AJAX Extensions”中拖曳 ScriptManager 控件，只有在设计网页时才能看到，它没有用户接口，因此在浏览时不会看到。而且它是“服务器”控件，所有工作都会在服务器上完成后，再将产生的脚本传送到浏览器中。

下面分别介绍如何在 ScriptManger 控件中使用<Scripts>标记和<Services>标记。

1．使用<Scripts>标记引入脚本资源

在 ScriptManger 控件中使用<Scripts>标记能够以声明的方式引入脚本资源。例如，引入编写的自定义脚本文件，代码如下所示：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
        <Scripts>
            <asp:ScriptReference Path="~/Script/MyScript.js" />
```

```
        </Scripts>
    </asp:ScriptManager>
```

上述代码在<asp:ScriptManager>标记中定义了一个子标记<Scripts>，其中还定义了一个<asp:ScriptReference>标记，并设定了该标记的 Path 属性（即给出引入的脚本资源的路径）。<asp:ScriptReference>标记对应着 ScriptReference 类，该类的常用属性如表 16.2 所示。

表 16.2 ScriptReference 类的常用属性

属　性	说　明
Assembly	指定引用的脚本被包含的程序集名称
IgnoreScriptPath	是否在引用脚本时包含脚本的路径
Name	指定引用程序集中某个脚本的名称
NotifyScriptLoaded	是否在加载脚本资源完成之后发出一个通知
Path	指定引用脚本的路径，一般为相对路径
ResourceUICultures	指定一系列的本地化脚本的区域名称
ScriptMode	引用脚本的模式，可以为 Auto、Debug 或者 Release 模式。默认值为 Auto

在 ScriptManger 控件中可以使用多个<Scripts>标记引入多个 JS 文件。

下面给出一个通过 ScriptManger 控件引用自定义脚本文件的实例。

实例位置：光盘\MR\Instance\16\16.1

【例 16.1】 使用<Scripts>标记引入脚本资源以检测输入是否为汉字。在文本框中输入内容，单击“确定”按钮，将检测用户输入的是否为汉字，如果不是汉字则弹出提示对话框。页面运行结果如图 16.4 所示。

程序实现的具体步骤说明如下。

（1）新建 ASP. NET 网站，默认主页为 Default.aspx。

（2）在网站中新建 Script 文件夹，在“解决方案资源管理器”中右键单击该文件夹名称，在弹出的快捷菜单中选择“添加新项”命令，在打开的“添加新项”对话框中选择“AJAX 客户端库”，在“名称”文本框中输入 MyScript.js，如图 16.5 所示，单击“添加”按钮完成操作。

图 16.4 验证是否为汉字

图 16.5 添加“AJAX 客户端库”

（3）在 MyScript.js 页面中，编写自定义的 JavaScript 脚本函数 validateName，代码如下所示：

```
/// <reference name="MicrosoftAjax.js"/>
function validateName(Name)
{
    var regex = new RegExp("^[u4E00-\U9fa5]{0,}$");//创建 RegExp 正则表达式对象
    return regex.test(Name);                        //检测字符串是否与给出正则表达式匹配
}
```

（4）在 Default.aspx 页面上首先添加 ScriptManger 控件用于管理脚本，并通过 ScriptReference 元素指定引用脚本的路径“~/Script/MyScript.js”；然后依次添加一个 Input（Text）控件用于输入姓名、一个 Input（Button）控件用于验证用户的输入。代码如下所示：

```
<body>
    <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
        <Scripts>
            <asp:ScriptReference Path="~/Script/MyScript.js" />
        </Scripts>
    </asp:ScriptManager>
    输入姓名：<input id="Text1" type="text" />
 <input  id="Button1"  type="button"  value=" 确 定 "  onclick="Button1_
onclick()"/><br />
    </form>
</body>
```

（5）在 Defaul.aspx 页面中，编写自定义的 JavaScript 脚本函数 Button1_onclick()，在按钮的 onclick 事件中调用此函数。关键代码如下所示：

```
<head runat="server">
    <title>无标题页</title>
    <script type="text/javascript">
        function Button1_onclick()
        {
            if(!validateName(document.getElementById("Text1").value))
            {
                alert("输入不是汉字，请重新输入");
                document.getElementById("Text1").value = "";
                document.getElementById("Text1").focus();
            }
        }
    </script>
</head>
```

2．使用<Services>标记引入 Web Service

在 ScriptManger 控件中使用<Services>标记能够以声明的方式引入 Web 服务资源。例如，引入 Web Service 文件（文件后缀为.asmx），代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Services>
        <asp:ServiceReference Path="WebService.asmx" />
    </Services>
</asp:ScriptManager>
```

下面给出一个通过 ScriptManger 控件引用 Web Service 文件的实例。

实例位置：光盘\MR\Instance\16\16.2

【例 16.2】 使用<Services>标记引入 Web Service 以返回随机数。单击页面上的“返回随机数”按钮，页面将返回范围在 12～17 之间的一个随机数（大于或等于 12 小于 17），如图 16.6 所示。

程序实现的具体步骤说明如下。

（1）新建 ASP. NET 网站，默认主页为 Default.aspx。

（2）在“解决方案资源管理器”中鼠标右键单击方案名称，在弹出的快捷菜单中选择“添加新项”命令，在打开的“添加新项”对话框中选择“Web 服务”，在“名称”文本框中输入 RandomService.asmx，如图 16.7 所示，单击“添加”按钮完成添加操作。

图 16.6 返回随机数

图 16.7 添加“Web 服务”文件

（3）在打开的 RandomService.cs 文件（此文件将自动存于 App_Code 文件夹下）中，定义一个静态方法 GetRandom()，用于返回 12～17 之间的一个随机数。代码如下所示：

```
using System;
using System.Collections;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;
/// <summary>
///RandomService 的摘要说明
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
//若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。
[System.Web.Script.Services.ScriptService]
public class RandomService : System.Web.Services.WebService {
    public RandomService () {
        //如果使用设计的组件，请取消注释以下行
        //InitializeComponent();
    }
    [WebMethod]
    public static int GetRandom()
```

```
    {
        Random ran = new Random();          //创建 Random 对象实例
        int getNum = ran.Next(12, 17);      //返回指定范围内的随机数
        return getNum;
    }
}
```

学习手册 说明　① 在 RandomService.cs 文件中，应用了[System.Web.Script.Services.ScriptService]属性。该属性是 ASP. NET AJAX 能够从客户端访问到定义的 Web Service 所必需使用的属性。

② 在 RandomService.cs 文件中，定义了一个静态的方法 GetRandom()，注意使用了 static 修饰符。

③ Random 表示伪随机数生成器，它能够产生随机的数字序列。

（4）在 Default.aspx 页面上，添加一个 ScriptManger 控件用于管理脚本，并通过 ScriptReference 元素指定引用的 Web 服务文件 RandomService.asmx；添加一个 UpdatePanel 控件用于实现局部刷新。在 UpdatePanel 控件内添加一个 Label 控件用于显示获取到的随机数；添加一个 Button 控件用于获取随机数。代码如下所示：

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
            <Services>
                <asp:ServiceReference Path="RandomService.asmx" />
            </Services>
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                随机数为：
                <br />
                <div align="center" style=" width:123px; height:60px; line-height:
                60px; background-image: url('bg.jpg')">
                    <asp:Label ID="Label1" runat="server" Font-Bold="True" Font-
                    Size="18px"> </asp:Label>
                </div>
                <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                Text="返回随机数" />
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
```

（5）双击 Default.aspx 页面上的 Button 控件，进入后台页面 Default.aspx.cs。在该页面中编写 Button1_Click 事件，将获取到的随机数作为 Label 控件的文本。代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = RandomService.GetRandom().ToString();
}
```

学习手册 注意　以上代码中，使用了[类名].[方法名]的格式，以调用 Web 服务文件中的方法。

16.2.2 UpdatePanel 局部更新控件

早期的 AJAX 版本开发出很多的 AJAX 服务器控件，例如 TextBox、Button 等，随着.NET 服务器控件的更新，发现开发出这么多的 AJAX 服务器控件并不符合实际需要，最后微软开发出了 AJAX 的 UpdatePanel 控件，由程序人员将 ASP.NET 服务器控件拖放到 UpdatePanel 控件中，使原本不具备 AJAX 能力的 ASP.NET 服务器控件都具有 AJAX 异步的功能。

UpdatePanel 控件的常用属性如表 16.3 所示。

表 16.3 UpdatePanel 控件的常用属性

属　性	说　明
ContentTemplate	内容模板，在该模板内放置控件、HTML 代码等
UpdateMode	UpdateMode 属性共有两种模式：Always 与 Conditional，Always 是每次 Postback 后，UpdatePanel 会连带被更新；相反，Conditional 只针对特定情况才被更新
RenderMode	若 RenderMode 的属性值为 Block，则以<DIV>标签来定义程序段；若为 Inline，则以<Span>标签来定义程序段
Triggers	用于设置 UpdatePanel 的触发事件

学习手册 说明 UpdatePanel 控件的 Triggers 包含两种触发器：一种是 AsyncPostBackTrigger 用于引发局部更新；一种是 PostBackTrigger 用于引发整页回送。设置 Triggers 的属性值，如图 16.8 所示。

图 16.8 Triggers 属性值设置

学习手册 注意 页面中使用多个 UpdatePanel 控件时，通过设置其 UpdateMode 属性为 Conditional，可以避免相互间的影响。

在 UpdatePanel 控件内的控件可以实现局部更新，那么在 UpdatePanel 控件之外的控件能否控制或者引发其局部更新呢？答案是肯定的。

通过 Triggers 属性包含的 AsyncPostBackTrigger 触发器可以引发 UpdatePanel 控件的局部更新。在该触发器中指定控件名称、该控件的某个服务器端事件，这样就可以使 UpdatePanel 控件之外的控件来引发局部更新，而避免不必要的整页更新。

下面给出使用 UpdatePanel 控件实现局部更新的实例。

实例位置：光盘\MR\Instance\16\16.3

【例 16.3】 本实例主要为读者演示如何使用 AsyncPostBackTrigger 触发器引发 UpdatePanel 控件局部更新。

在页面上（如图 16.9 所示）单击“显示时间”按钮，在“时间 2”处将无刷新显示当前系统日期时间，可以与页面上的“时间 1”进行对比。在本例中，无须整页更新，其执行过程是 AJAX 方式的页面局部更新，如图 16.10 所示。

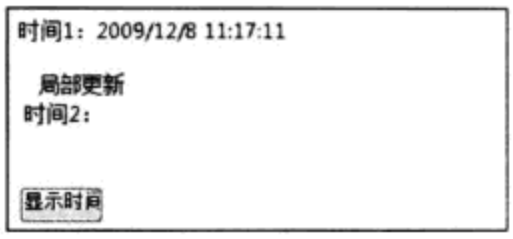


图 16.9　运行页面

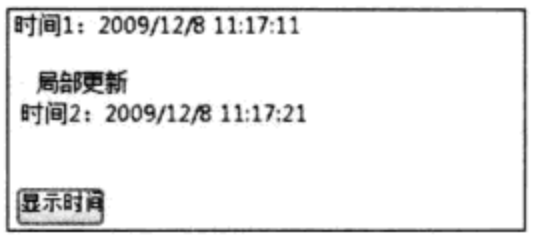


图 16.10　无刷新显示

程序实现的具体步骤说明如下。

（1）新建 ASP. NET 网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面上添加一个 ScriptManager 控件用于管理脚本；添加一个 ID 值为 Label1 的 Label 控件用于显示时间 1；添加一个 UpdatePanel 控件用于实现局部更新；在 UpdatePanel 控件内添加一个 ID 值为 Label2 的 Label 控件用于显示时间 2；在 UpdatePanel 控件之外添加一个 Button 控件。代码如下所示：

```
<body style="font-size:14px">
    <form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <div style=" width:500px; height:150px; background-color:#FFDFEF; padding:
    5px 0px 0px 8px;">
    时间 1: <asp:Label ID="Label1" runat="server"></asp:Label>
    <br/>
    <br/>
    <fieldset style="width:300px; height:60px">
        <legend>局部更新</legend>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                时间 2: <asp:Label ID="Label2" runat="server"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>
    </fieldset>
    <br/>
    <br/>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="显示时间"
          Width="55px" />
    </div>
    </form>
</body>
```

（3）双击页面上的 Button 控件，编写其 Click 事件对应的 Button1_Click 函数，在该函数中将当前系统日期时间作为 Label2 的文本。在 Page_Load 事件中设置当前时间为 Label1 的文本。代码如下所示：

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

```
    {
        Label1.Text = DateTime.Now.ToString();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label2.Text = DateTime.Now.ToString();
}
```

（4）在 Default.aspx 页面上右键单击 UpdatePanel 控件，在弹出的快捷菜单中选择“属性”，在打开的属性窗口中可以看到 Triggers 属性。

（5）单击 Triggers 属性中的[...]按钮，打开“UpdatePanelTrigger 集合编辑器”对话框，单击“添加”按钮右侧的▾，在下拉菜单中选择 AsyncPostBackTrigger 触发器，如图 16.11 所示。

（6）在“UpdatePanelTrigger 集合编辑器”对话框中，设置“行为”栏目中的 ControlID 属性和 EventName 属性，在对应的下拉框中分别选择“Button1”和“Click”，如图 16.12 所示。

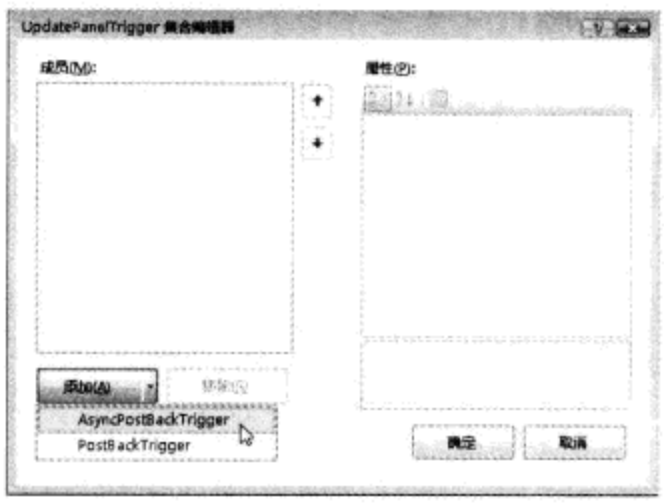


图 16.11　添加 AsyncPostBackTrigger 触发器

图 16.12　设置 AsyncPostBackTrigger 触发器

（7）切换到源视图，可以看到自动生成的<Triggers>和其中的<asp:AsyncPostBackTrigger>标签。代码如下所示：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
   <ContentTemplate>
       时间 2: <asp:Label ID="Label2" runat="server"></asp:Label>
    </ContentTemplate>
    <Triggers>
       <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
    </Triggers>
</asp:UpdatePanel>
```

在 AJAX 环境中弹出对话框

使用 Response.Write 方法可以直接输出 JavaScript 脚本的 alert 语句以在客户端弹出对话框，代码如下所示：

```
Response.Write("<script>alert('操作成功!');</script>");
```

但是，在 ASP. NET AJAX 应用程序中，如果使用上述代码弹出一个对话框是会发生错误的，如图 16.13 所示。

解决方法是使用 ScriptManager 类的 RegisterStartupScript 方法来输出当 UpdatePanel 控件更新时需要弹出的对话框，代码如下所示：

```
ScriptManager.RegisterStartupScript(UpdatePanel1, typeof(UpdatePanel), "scriptname", "alert('操作成功');", true);
```

图 16.13　错误提示

16.2.3　Timer 计时器控件

Timer 定时器用 JavaScript 构建非常容易，但在 ASP.NET 中实现 Timer 定时器不但困难，而且运作起来非常麻烦，还会损耗计算机资源。但 AJAX Framework 直接构建了一个 AJAX Timer 服务器控件，让程序开发人员可以设置时间间隔来触发特定事件的操作。

下面对 Timer 控件的相关属性和事件进行介绍。

1．Interval 属性

用于设置 Timer 时间控件的 Tick 事件间隔时间，单位为毫秒（1000 毫秒等于 1 秒）。

例如要设置触发 Tick 事件的时间间隔为 1 秒，将 Interval 属性设置为 1000 即可，如图 16.14 所示。

2．Tick 事件

用于在指定的时间间隔进行触发的事件。

实例位置：光盘\MR\Instance\16\16.4

【例 16.4】 本实例主要使用 Timer 控件的 Tick 事件实现实时显示当前系统时间。在页面上显示当前系统时间时，可以看到时间是以秒为单位实时变化的，实例运行结果如图 16.15 和图 16.16 所示。

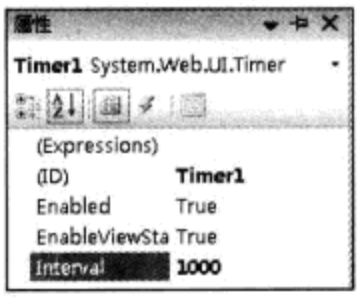


图 16.14　设置 Interval 属性为 1 秒

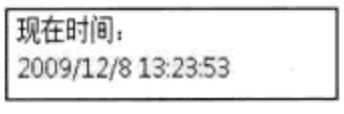


图 16.15　时间每隔 1 秒实时显示

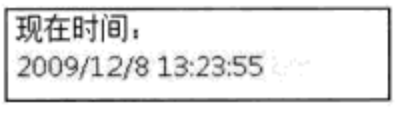


图 16.16　时间实时显示

程序实现的具体步骤说明如下。

（1）新建一个网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面上添加一个 ScriptManager 控件用于管理脚本。添加一个 UpdatePanel 控件用于局部刷新。在 UpdatePanel 控件中添加一个 Label 控件用于显示当前系统时间，添加一个 Timer 控件，设置 Timer 控件的 Interval 属性为 1000 毫秒（即 1 秒），如图 16.17 所示。

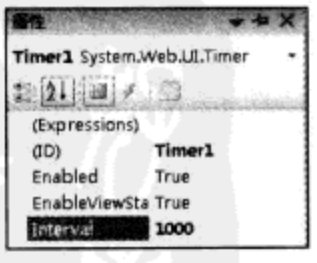


图 16.17　设置 Timer 控件

（3）在 Timer 控件的 Tick 事件中编写代码。

```
protected void Timer1_Tick(object sender, EventArgs e)
```

```
{
Label1.Text = DateTime.Now.ToString();
}
```

学习手册 注意 本实例是无刷新效果实时显示当前系统时间的。

16.3 AJAX Control Toolkit 扩展控件

专题讲座：光盘\MR\Video\16\ AJAX Control Toolkit 中的控件.exe

➤➤➤视频速递：通过本视频读者可以直观地掌握如何添加 AJAX Control Toolkit 控件。

ASP. NET AJAX Control Toolkit 是基于 ASP. NET AJAX 构建的，它是一个免费的、开源的 ASP. NET 服务器端控件包。其中包含了数十种组件化的、提供某种专一功能的 ASP.NET 服务器端控件和 ASP.NET AJAX 扩展控件。

16.3.1 下载 ASP. NET AJAX Control Toolkit

下载 ASP. NET AJAX Control Toolkit 的地址为：http://www.codeplex.com/AjaxControlToolkit/Release/ProjectReleases.aspx（这里以下载 Ajax ControlToolkit-Framework3.5SP1.zip 为例），在下载的文件目录中，包含一个名为 AjaxControlToolkit.sln 的 Visual Studio 解决方案，如图 16.18 所示。

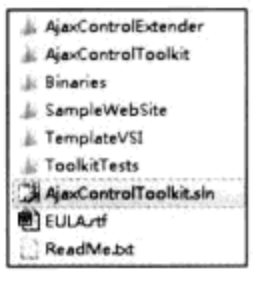


图 16.18 AjaxControlToolkit.sln 解决方案

图 16.19 生成 AjaxControlToolkit 项目

双击该图标，启动 Visual Studio 2008 并打开解决方案。在解决方案中包含一个名为 AjaxControlToolkit 的项目，鼠标右键单击该项目名称，在弹出的快捷菜单中选择“生成”命令，如图 16.19 所示。

学习手册 说明 生成 AjaxControlToolkit 项目成功后，在文件目录\AjaxControlToolkit\bin\Debug 下可以找到 AjaxControlToolkit.dll 这个函数库。

16.3.2 将控件添加到 Visual Studio 的 Toolbox 中

（1）新建或打开一个 ASP. NET 网站，打开“工具箱”窗口，使用鼠标右键单击空白处并在快捷菜单中选择“添加选项卡”命令，将选项卡命名为 Ajax Control Toolkit，然后鼠标右键单击该选项卡，在弹出的快捷菜单中选择“选择项”命令，如图 16.20 所示。

（2）打开“选择工具箱项”对话框，单击“浏览”按钮查找到 AjaxControlToolkit.dll 程序集，然后单击“确定”按钮将控件添加到 Visual Studio 2008 的 Ajax Control Toolkit 选项卡中，如图 16.21 所示。

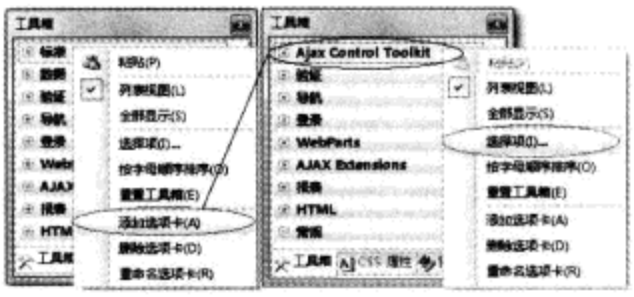


图 16.20　添加选项卡

图 16.21　将 AjaxControlToolkit 控件添加到 Visual Studio 2008“工具箱”

16.4　应用 AJAX Control Toolkit 扩展控件

专题讲座：光盘\MR\Video\16\ ASP. NET AJAX Control Toolkit 中的扩展控件.exe

➤➤➤视频速递：读者可以直观地掌握如何使用 ASP.NET AJAX Control Toolkit 中的扩展控件。

添加 ASP. NET AJAX Control Toolkit 控件之后，会将扩展控件添加到 Visual Studio 的工具箱中。这样，就可以像添加标准服务器控件那样，将 ASP. NET AJAX Control Toolkit 中的扩展控件添加到页面中，本节将为读者介绍几个典型的扩展控件，以便能使读者了解扩展控件的用法。

16.4.1　TextBoxWatermark：添加水印提示

TextBoxWatermark 扩展控件可以为 TextBox 服务器端控件添加水印效果。打开网页，文本框内将显示水印提示内容，当在文本框内单击鼠标时水印文字将立即消失，即变成空白文本框，用户随即可以输入数据。

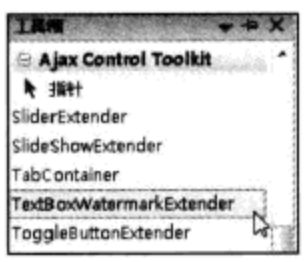


图 16.22　TextBoxWatermark 扩展控件

TextBoxWatermark 扩展控件在工具箱中的图标如图 16.22 所示。

学习手册 说明　在文本框中出现的水印文字只起到提示的作用，不会作为文本内容。

TextBoxWatermark 扩展控件的常用属性如表 16.4 所示。

表 16.4　TextBoxWatermark 扩展控件的常用属性

属　性	说　明
TargetControlID	目标 TextBox 控件 ID

续表

属　性	说　明
WatermarkText	设定显示的水印文字
WatermarkCssClass	水印文字应用的 CssClass

下面通过一个实例来演示 TextBoxWatermark 扩展控件的使用方法。

实例位置：光盘\MR\Instance\16\16.5

【例 16.5】 在网页上可以看到带有水印文字提示的文本框（如图 16.23 所示），在文本框内单击鼠标水印文字消失（如图 16.24 所示），然后在文本框内可以输入数据，如图 16.25 所示。

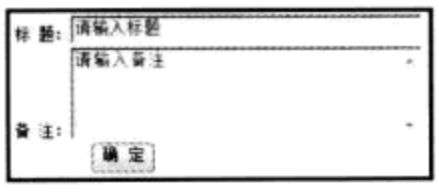


图 16.23　水印提示

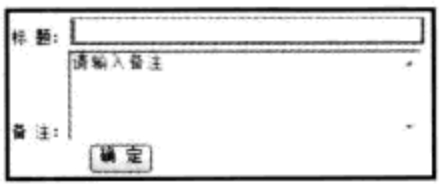


图 16.24　水印文字消失

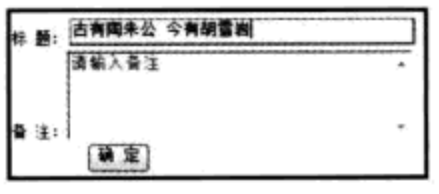


图 16.25　输入数据

程序实现的具体步骤说明如下。

（1）新建 ASP. NET 网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面上，添加一个 ScriptManager 控件用于管理脚本。添加两个 TextBox 控件，ID 分别为 TextBox1 和 TextBox2，设置 TextBox2 的 TextMode 属性为“MultiLine”；设置这两个 TextBox 控件的 CssClass 属性均为“txt”，BackColor 属性值为“#daeeee”。代码如下所示：

```
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    标   题 : <asp:TextBox ID="TextBox1" runat="server" Width="260px"
CssClass="txt" BackColor= "#daeeee"></asp:TextBox>
    <br />
    备  注: <asp:TextBox ID="TextBox2"
     runat="server" Rows="4" TextMode="MultiLine" Width="260px" CssClass="txt"
    BackColor="#daeeee"></asp:TextBox>
```

（3）在 Default.aspx 页面的源视图下，添加两个 TextBoxWatermarkExtender 控件为两个 TextBox 控件添加水印提示，分别设置其 TargetControlID 为 TextBox1、TextBox2，设置 WatermarkText 为“请输入标题”、“请输入备注”，WatermarkCssClass 均设置为“watermark”。代码如下所示：

```
    <cc1:TextBoxWatermarkExtender ID="TextBoxWatermarkExtender1" runat="server"
      TargetControlID="TextBox1"
      WatermarkText="请输入标题"
      WatermarkCssClass="watermark" >
    </cc1:TextBoxWatermarkExtender>
    <cc1:TextBoxWatermarkExtender ID="TextBoxWatermarkExtender2" runat="server"
      TargetControlID="TextBox2"
      WatermarkText="请输入备注"
      WatermarkCssClass="watermark" >
    </cc1:TextBoxWatermarkExtender>
```

其中，在<head>标记内使用<style>标记定义 CSS 样式 txt 和 watermark，代码如下所示：

```
<head runat="server">
    <title>无标题页</title>
    <style>
    .txt
    {
        border-style:solid;
        border-color:#666666;
        border-width:1px 2px 2px 1px;
        margin:2px;
    }
    .watermark
    {
        color:#666666;
    }
    </style>
</head>
```

16.4.2　PasswordStrength：智能密码强度提示

在网页上可以看到当用户输入密码的同时，会显示密码的强度，以提示和检测用户输入密码的安全级别。使用 ASP.NET AJAX Control Toolkit 中的 PasswordStrength 扩展控件可以为设置密码的 TextBox 服务器控件添加即时的密码强度检测功能，将检测的密码强度以及安全提醒返回给客户端。当输入完成（即 TextBox 控件失去焦点时），提示信息会自动消失。

使用 PasswordStrength 扩展控件时，返回给客户端的信息仅起到提示的作用，它不会验证或阻止用户的输入。

图 16.26　PasswordStrength 扩展控件

PasswordStrength 扩展控件在工具箱中的图标如图 16.26 所示。

PasswordStrength 扩展控件的常用属性如表 16.5 所示。

表 16.5　PasswordStrength 扩展控件的常用属性

属　性	说　明
TargetControlID	目标 TextBox 控件 ID
HelpStatusLabelID	目标 Label 控件 ID，在此控件内显示填写符合安全标准的密码提示信息，此信息即时更新
StrengthIndicatorType	指定密码强度的方式，值为 Text 或 BarIndicator，即文本方式或者进度条方式
DisplayPosition	密码强度相对于文本框的显示位置
PreferredPasswordLength	安全密码的最小长度
MinimumNumericCharacters	安全密码包含的数字个数
MinimumSymbolCharacters	安全密码包含的特殊字符个数
RequiresUpperAndLowerCaseCharacters	安全密码是否包含大小写字母，True 代表包含，False 代表可以不包含
PrefixText	强度级别的前缀文字

下面通过一个实例来演示 PasswordStrength 扩展控件的使用方法。

实例位置：光盘\MR\Instance\16\16.6

【例 16.6】 当在页面的文本框中输入密码时（为了达到演示效果，TextBox 控件未使用 Password 模式），在文本框的右侧将显示密码安全强度，在文本框的下方将显示安全密码的输入提示。在这里，设置安全密码的长度最少为 8 位，其中要包含一个数字、一个符号，如图 16.27 和图 16.28 所示。

图 16.27 密码强度提示 1

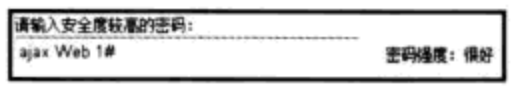


图 16.28 密码强度提示 2

程序实现的具体步骤说明如下。

（1）新建 ASP. NET 网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面上，添加一个 ScriptManager 控件用于管理脚本。添加一个 TextBox 控件用于输入密码。在源视图中添加一个 PasswordStrength 控件，并设置其对应的属性。代码如下所示：

```
<body style="font-size:12px">
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
            请输入安全度较高的密码：<br />
            <asp:TextBox ID="TextBox1" runat="server" Width="260px"></asp:
            TextBox>
            <br />
            <asp:Label ID="Label1" runat="server"></asp:Label>
            <cc1:PasswordStrength ID="PasswordStrength1" runat="server"
            TargetControlID="TextBox1"
            HelpStatusLabelID="Label1"
            PreferredPasswordLength="8"
            MinimumNumericCharacters="1"
            MinimumSymbolCharacters="1"
            RequiresUpperAndLowerCaseCharacters="true"
            DisplayPosition="RightSide"
            PrefixText=" 密码强度："
            StrengthIndicatorType="Text"
            TextStrengthDescriptions="差;一般;好;很好"
            TextCssClass="text" >
            </cc1:PasswordStrength>
    </div>
    </form>
</body>
```

在上述代码中，PasswordStrength 控件的 TargetControlID 设定为 TextBox1，HelpStatusLabelID 设定为 Label1，即在此 Label 控件内显示提示信息；PreferredPasswordLength 设定为 8，表示安全密码的长度至少为 8；MinimumNumericCharacters 设定为 1，表示安全密码中要包含一个数字；MinimumSymbolCharacters 设定为 1，表示安全密码中要包含一个特殊字符；RequiresUpperAndLowerCaseCharacters 设定为 true，表示安全密码中要有大小写混合字母；DisplayPosition 设定为 RightSide，表示密码强度级别显示在文本框的右侧；PrefixText 表示强度级别的前缀文字；StrengthIndicatorType 设定为 Text，表示以

文本方式指示密码强度；TextStrengthDescriptions 设定以分号连接用来描述不同强度级别的字符串。

16.4.3　SlideShow：播放照片

ASP. NET AJAX Control Toolkit 中的 SlideShow 扩展控件可以实现自动播放照片的功能。仿照幻灯片，可以自动播放也可以上下翻动观赏照片。在制作电子相册等程序中经常使用 SlideShow 扩展控件。

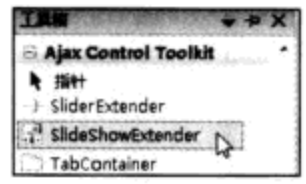


图 16.29　SlideShow 扩展控件

SlideShow 扩展控件在工具箱中的图标如图 16.29 所示。

SlideShow 扩展控件的常用属性如表 16.6 所示。

表 16.6　SlideShow 扩展控件的常用属性及描述

常 用 属 性	描　　述
TargetControlID	目标 Image 服务器端控件 ID
AutoPlay	是否自动播放
Loop	是否循环播放
PreviousButtonID	“上一张”按钮 ID
NextButtonID	“下一张”按钮 ID
PlayButtonID	播放按钮 ID
PlayInterval	两张画面播放的时间间隔，单位为毫秒
PlayButtonText	播放时按钮显示的文本
StopButtonText	停止自动播放时按钮显示的文本
SlideShowServicePath	调用的 Web Service
SlideShowServiceMethod	指定 Web Service 中的方法
ContextKey	该值传递给 Web Service 中方法的 contextKey 参数
UseContexKey	是否启用 ContextKey 属性

实例位置：光盘\MR\Instance\16\16.7

【例 16.7】 运行页面可以看到循环播放的 3 张图片，单击“停止播放”按钮图片暂停播放，按钮上的文本将显示为“开始播放”；再次单击此按钮可以恢复自动播放，按钮上的文本将显示为“停止播放”。单击“上一张”或“下一张”按钮可以按照指定顺序查看图片。运行结果如图 16.30 和图 16.31 所示。

程序实现的具体步骤说明如下。

（1）新建 ASP. NET 网站，默认主页为 Default.aspx。

（2）在 Default.aspx 页面上，添加一个 ScriptManager 控件用于管理脚本。添加一个 Image 控件用于显示图片，ID 为 Image1，设置其宽度和高度分别为 300px 和 200px。添加一个 Label 控件用于显示图片的名称，ID 为 Label1。添加 3 个 Button 控件，ID 分别为 Button1、Button2 和 Button3，其文本分别设置为“上一张”、“开始播放”和“下一张”。在源视图下，添加一个 SlideShowExtender 控件，设置其相关属性。代码如下所示：

图 16.30　自动播放

图 16.31　暂停播放

```
<body style="font-size:16px">
    <form id="form1" runat="server">
    <div align="center">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <br />
        <asp:Image ID="Image1" runat="server" Height="200px" Width="300px" />
        <br />
        <asp:Label ID="Label1" runat="server"></asp:Label>
        <br />
        <asp:Button ID="Button1" runat="server" Text="上一张" CssClass="Button" />
 
        <asp:Button ID="Button2" runat="server" Text="开始播放" CssClass=
        "Button" />
     
        <asp:Button ID="Button3" runat="server" Text="下一张" CssClass="Button" />
        <cc1:SlideShowExtender ID="SlideShowExtender1" runat="server"
          TargetControlID="Image1"
          AutoPlay="true"
          ImageTitleLabelID="Label1"
          Loop="true"
          NextButtonID="Button3"
          PreviousButtonID="Button1"
          PlayButtonID="Button2"
          PlayInterval="3000"
          PlayButtonText="开始播放"
          StopButtonText="停止播放"
          SlideShowServicePath="Photo_Service.asmx"
          SlideShowServiceMethod="GetSlide" >
        </cc1:SlideShowExtender>
    </div>
    </form>
</body>
```

（3）在解决方案管理器上，右键单击方案名称，在弹出的快捷菜单中选择“添加新项”命令，打开 “添加新项”对话框。在“模板”栏目中选择“Web 服务”，在“名称”文本框中输入 Photo_Service.asmx，如图 16.32 所示，单击“添加”按钮完成操作。

图 16.32　添加 Web 服务

（4）在打开的 Photo_Service.cs 文件（此文件自动存储在 App_Code 文件夹）中，启用[System.Web.Script.Services.ScriptService]，自定义 GetSlide() 方法返回值类型为 AjaxControlToolkit.Slide[]。代码如下所示：

```
using System;
using System.Collections;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;
/// <summary>
///Photo_Service 的摘要说明
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
//若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释
 [System.Web.Script.Services.ScriptService]
public class Photo_Service : System.Web.Services.WebService {
    public Photo_Service () {
        //如果使用设计的组件，请取消注释以下行
        //InitializeComponent();
    }
    [WebMethod]
    //public string HelloWorld() {
    //    return "Hello World";
    //}
    public AjaxControlToolkit.Slide[] GetSlide()
    {
        //定义幻灯片数组
        AjaxControlToolkit.Slide[] photos = new AjaxControlToolkit.Slide[3];
        //定义幻灯片对象
        AjaxControlToolkit.Slide photo = new AjaxControlToolkit.Slide();
        //以下分别定义 3 个幻灯片，其中包含图片路径、图片名称、图片描述，然后将其分别添加到 photos
        photo = new AjaxControlToolkit.Slide("Images/1.bmp", "Tree", "图片 1");
        photos[0] = photo;
        photo = new AjaxControlToolkit.Slide("Images/2.bmp", "fungus", "图片 2");
        photos[1] = photo;
        photo = new AjaxControlToolkit.Slide("Images/3.jpg", "flower", "图片 3");
        photos[2] = photo;
        return photos;
    }
}
```

16.5　实 战 练 习

16.5.1　应用 Timer 控件实现在线考试倒计时

➤➤➤题目描述

本实例应用 AJAX Timer 控件实现在线考试倒计时功能，ASP.NET 3.5 所提供的 Timer

控件是一个服务器控件，它能够定时引发整页回发；在搭配了 UpdatePanel 后就可以定时引发异步回发并局部更新 UpdatePanel 控件中的内容。本实例运行结果如图 16.33 所示。

图 16.33　应用 Timer 控件实现在线考试倒计时

➤➤➤技术指导

在应用 UpdatePanel 控件实现页面局部更新时，用户必须初始化一个一般情况下会回发的动作，如单击按钮。但在实际应用中用户可能会希望没有动作的情况下自动完成一次完整或局部页面的刷新，此时便可利用 AJAX 的 Timer 控件，每隔一段时间固定触发一个事件，就像闹钟叫你起床一样。

Timer 控件的使用非常简单，其中比较重要的属性就是 Interval 及 Enabled。

☑　Interval 属性

Interval 属性用来设置页面更新间隔的最大毫秒数，其默认值为 60000 毫秒(即 60 秒)。每当到达 Timer 控件的 Interval 属性所设置的间隔时间而进行回发时，就会引发服务器端 Tick 事件，可以在该事件中根据实际需要定时执行特定的更新操作。

必须提醒大家，Timer 控件可能会加大 Web 应用程序的负载。因此，在引入自动回发特性前并在确实需要的时候引入 Timer 控件，尽可能把间隔时间设置得长一点，如果设置得太短将会使得页面回发频率增加，加大服务器的流量。

☑　Enabled 属性

如果要停止一个定时器，可在服务器端代码里将 Timer 控件的 Enabled 属性设置为 false 即可，这就像把闹钟关掉了，即使时间到了，它也不会响。

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/16.5.1**

16.5.2　应用 Timer 控件实现网站气泡提示

➤➤➤题目描述

随着网络技术的飞速发展，网站气泡提示信息在网络上越来越受到人们的青睐。本实例实现的是在添加备忘录信息的时候智能弹出气泡提示信息（类似 QQ 气泡提示窗体）。在实现插入新的提示主题和提示的具体时间时，如果时间到了即可在程序右下角中自动弹出气泡提示信息，实例运行效果如图 16.34 所示。

图 16.34　应用 Timer 控件实现网站气泡提示

➤➤➤技术指导

在本实例中，当数据库的时间和系统的时间相同时将在程序的右下角弹出一个气泡提示信息框。该气泡提示信息框将显示用户设置的备忘信息。该提示框是在 JavaScript 中利

用 window 对象中的 CreatePopup 对象来实现的。该代码实现在 blebHint.js 文件中。

调用提示框是在后台代码中实现的。在后台中首先判断数据库的时间是否和系统时间相同。如果相同将弹出网站备忘录的提示对话框。每一秒钟进行时间的比较操作是在 AJAX 中提供的计时器 Timer 控件实现的，该计时器 Timer 控件能够指定一个时间间隔和一个 Tick 事件，在每一次时间间隔到达之后，都触发 Tick 事件。

Timer1_Tick 事件中每隔一秒钟触发一次 Timer1_Tick 事件，该事件的代码如下所示：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
  this.Label1.Text = System.DateTime.Now.ToString();
  SqlDataReader sdr1 = ep.ExceRead("select * from tb_memo where name='" +
  Session["UserName"] + "'and pwd='" + Session["UserPwd"] + "'");
  while (sdr1.Read())
   {
       this.Label2.Text = sdr1["alerttime"].ToString();
       if (this.Label1.Text == sdr1["alerttime"].ToString())
     {
        this.Label2.Text = sdr1["subject"].ToString();
       ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "",
       "var MSG1 = new PopBubble('备忘录短消息提示：','" + this.Label2. Text + "');
       MSG1.show();", true);
      }
   }
       sdr1.Close();
 }
```

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/16.5.2**

16.5.3　多样式验证控件验证注册信息

➤➤➤题目描述

ASP.NET 中的验证控件可以在网页或对话框上显示验证结果，并且可以达到预期的效果。但是验证结果并不能友好、明显地提示给用户，ASP.NET AJAX Control Toolkit 中的 ValidatorCallout 控件可以实现这样的验证效果。实例运行结果如图 16.35 所示。

图 16.35　多样式验证控件 ValidatorCallout 验证注册信息

➤➤➤技术指导

多样式验证控件 ValidatorCallout 是用来增强 ASP.NET 既有的验证控件（Validator）功能的一个扩展控件，让验证控件具备更为华丽的外观。要使用这个强化的验证控件，与以往使用验证控件一样，需要为一个输入的字段添加一个验证控件，接着再添加多样式验证扩展控件，然后将它的 TargetControlID 属性指定为先前添加的验证控件。ValidatorCallout 控件的主要属性及说明如表 16.7 所示。

表 16.7　ValidatorCallout 控件的主要属性及说明

属　性	说　明
TargetControllID	设置 ASP.NET 服务器端验证控件的 ID
Width	提示信息框的显示宽度
CloseImageUrl	指定 Close 图像
WarningIconImageUrl	指定警告图像
HighlightCssClass	设置验证结果的样式
Animations	设置验证结果的动画
OnShow	显示验证结果时的动画
OnHide	隐藏验证结果时的动画

➤➤➤紧急救援

如果你在做本例题的过程中遇到困难或疑惑，可以按照下面救援通道提供的网址获取本例题的源码和技术文档。

救援通道：**http://www.mrbccd.com/ASP.NET/loveASP.NET/16.5.3**

16.6　本 章 小 结

只能在服务器上运行程序就像跛脚巨人，虽然很强，但总缺少那么一点灵活的感觉，唯有添加 AJAX 弥补服务器与浏览器间的断层，才能最大限度地满足 Web 应用程序的所有需求。本章对 ASP.NET AJAX 进行了详细的介绍，通过对本章的学习，读者完全可以掌握 AJAX 的基本用法，希望读者能够认真学习本章，以便为以后开发 AJAX 程序打下坚实的基础。

第 17 章

调试与错误处理

（名师课堂：20 分钟）

开发和编写程序的过程中经常会出现一些错误，其中包括语法错误、语义错误和编辑错误等，无论哪种错误都有可能导致程序不能执行或执行过程中出现失败。因此如何处理错误可能是优秀的应用程序设计中最重要的部分。本章将详细介绍程序的错误类型、程序调试以及如何进行错误处理。通过本章的学习，读者可以达到以下目的：

- 利用错误类型查找程序中的各种 Debug
- 使用断点来调试程序
- 如何排除服务器故障
- 启用 ASP.NET 异常调试
- ASP.NET 异常处理的最佳做法（捕获异常语句）

17.1 认识错误类型

专题讲座：光盘\MR\Video\17\错误类型.exe

➤➤➤视频速递：读者可以全面了解开发过程中出现的几种错误类型。

从本节开始，读者就要学习调试与错误处理。这非常重要，因为在开发中不可能保证没有错误，人无完人，更何况是开发程序！那么，出现错误时就要及时去解决，由于代码过多，可能有时候排除错误比较困难，为了解决这个难题，在 Visual Studio 2008 开发环境中的调试与错误处理变得尤为重要。本节主要介绍常见的几种错误类型，其中包括语法错误、语义错误和逻辑错误。

17.1.1 语法错误

语法错误是一种程序错误，它会影响编译器完成工作。它也是最简单的错误，几乎所有的语法错误都能被编译器或解释器发现，并将错误消息显示出来提醒程序开发人员。

在 Visual Studio 2008 中遇到语法错误时，错误消息将显示在“错误列表”窗口中。这些消息将会告诉程序开发人员语法错误的位置（行、列和文件），并给出错误的简要说明，如图 17.1 所示。

图 17.1 语法错误：第 17 行程序代码少了一个分号

学习手册 说明 如图 17.1 所示，程序若有错，开发工具会在有问题的地方以下画线标识。必须把所有网页里的错误更正后才能执行、查看网页。

17.1.2 语义错误

程序源代码的语法正确而语义或意思与程序开发人员本意不同时，就是语义错误。此类错误比较难以察觉，它通常在程序运行过程中出现。语义错误会导致程序非正常终止。例如，在将数据信息绑定到表格控件时，经常会出现“未将对象引用设置到对象的实例中”

的错误，此类语义错误在程序运行时将会被调试器以异常的形式告知程序开发人员，如图 17.2 所示。

图 17.2　语义错误

17.1.3　逻辑错误

不是所有的语义错误都容易发现，它们可能隐藏得很深。在某些语义错误下，程序仍可以继续执行，但执行结果却不是程序开发人员想要的，此类错误就是逻辑错误。例如，在程序中，需要计算表达式 c=a+b 的值，但在编程的过程中，将表达式中的“+”写成了“-”，像这样的错误，调试器不能以异常的形式告诉程序开发人员，这种错误就是逻辑错误。程序开发人员可以通过调试解决此类错误。

举一个在实际程序开发中的例子，在实现 GridView 中突出显示指定单元格中的数据时，编写了如下代码，该代码所要实现的就是员工薪资如果为 1500 元就以红色突出显示出来。

```
for (int i = 0; i < GridView1.Rows.Count-1; i++)
{
    DataRowView mydrv = myds.Tables["tb_mrEmply"].DefaultView[i];
    string mypay = Convert.ToString(mydrv["起薪"]);
    if (Convert.ToInt32(mypay) < 1600)
    {
        GridView1.Rows[i].Cells[5].BackColor = System.Drawing.Color.Red;
    }
}
```

可实际运行结果却如图 17.3 所示。

这里就出现了一个逻辑错误，有两个员工（房大伟和孙秀梅）起薪都是 1500 元，却只突出显示了一名员工（执行过程中并没有报错且正常运行，但结果却不是程序员想要的）。经过调试后，更改的代码如下（重新计算了 GridView 控件的行索引值）：

```
for (int i = 0; i < GridView1.Rows.Count; i++)
```

```
{
    DataRowView mydrv = myds.Tables["tb_mrEmply"].DefaultView[i];
    string mypay = Convert.ToString(mydrv["起薪"]);
    if (Convert.ToInt32(mypay) < 1600)
    {
        Gvinfo.Rows[i].Cells[5].BackColor = System.Drawing.Color.Red;
    }
}
```

运行结果如图 17.4 所示。

身份证号码	姓名	雇员编码	家庭地址	出生日期	起薪	编辑
2202831001	熊大伟	10001	吉林吉林	01-01-1982		编辑
2202831002	王小科	10002	山西长治	05-05-1985	￥1,600.00	编辑
2202831003	吕芳	10003	吉林长春	02-02-1982	￥1,800.00	编辑
2202831004	梁冰	10004	山东济南	07-07-1983	￥2,000.00	编辑
2202831005	孙秀梅	10005	吉林四平	05-05-1984	￥1,500.00	编辑

图 17.3　员工薪资如果为 1500 元就以红色突出显示出来（错误显示）

身份证号码	姓名	雇员编码	家庭地址	出生日期	起薪	编辑
2202831001	熊大伟	10001	吉林吉林	01-01-1982		编辑
2202831002	王小科	10002	山西长治	05-05-1985	￥1,600.00	编辑
2202831003	吕芳	10003	吉林长春	02-02-1982	￥1,800.00	编辑
2202831004	梁冰	10004	山东济南	07-07-1983	￥2,000.00	编辑
2202831005	孙秀梅	10005	吉林四平	05-05-1984		编辑

图 17.4　员工薪资如果为 1500 元就以红色突出显示出来（正确显示）

17.2　掌握程序调试

专题讲座：光盘\MR\Video\17\程序调试.exe

➤➤➤视频速递：读者可以直观地掌握如何进行程序调试。

在 Visual Studio .NET 环境中提供了 Visual Studio 调试器。该调试器提供了功能强大的命令来控制应用程序的执行。下面介绍几种对应用程序进行调试的方法。

17.2.1　设定断点

首先来看一下如图 17.5 所示的程序，简单预测一下程序执行时出现的结果。

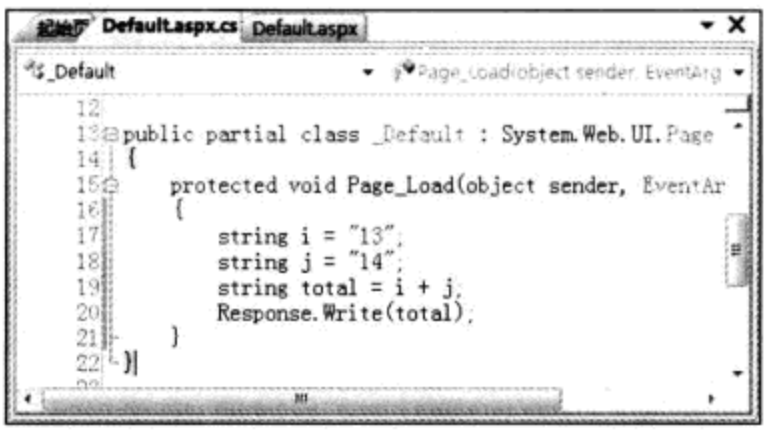


图 17.5　预测程序执行时出现的结果

或许有的用户会觉得结果是 27，但实际运行结果为 1314。很多时候我们在开发程序时，可能因为不注意或认知上的错误造成编写程序时的一些状况发生，又或者很多时候程序太多行了，而突然想知道某个值在某一行的变化，此时就可以使用断点值来暂时中断程序。

断点是一个信号，它通知调试器在某个特定点上暂时将程序执行挂起。当执行在某个断点处挂起时，该程序处于中断模式。进入中断模式并不会终止或结束程序的执行，程序

可以在任何时候继续运行。

插入断点有以下三种方式。

☑ 在要设置断点行的最左边的灰色区域空白处单击鼠标（一定要在左边的灰色区域按下鼠标才行）。

☑ 右键单击设置断点的代码行，选择“断点”/“插入断点”命令，如图 17.6 所示。

☑ 单击要设置断点的代码行，选择菜单中的“调试”/“切换断点(G)”命令，如图 17.7 所示。

插入断点成功后，就会在设置断点的行旁边的灰色空白处出现一个红色圆点，并且该行代码也呈高亮显示，如图 17.8 所示。

图 17.6　右键插入断点

图 17.7　菜单栏插入断点

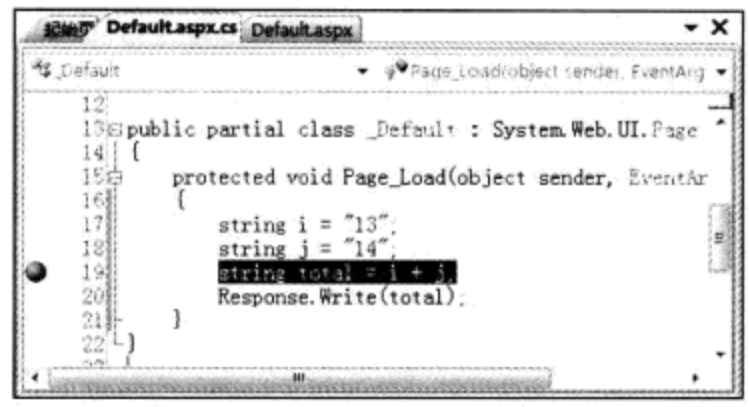


图 17.8　设定断点

17.2.2　开始执行

设定断点后就可以开始进行调试操作，其调试可借助 Visual Studio 调试器的调试工具栏进行，如图 17.9 所示。调试工具栏中的前 4 个按钮是可以手动控制的。第一个按钮是“启动调试”按钮，该按钮与标准工具栏上的“启动调试”按钮功能相同；另外 3 个呈灰色的按钮只有在程序运行后才能正常使用，如图 17.10 所示。

图 17.9　调试工具栏

图 17.10　启动调试后的调试工具栏

调试工具栏中各按钮的功能说明如表 17.1 所示。

表 17.1　调试工具栏中各按钮的功能说明

工具栏按钮	图 标 文 本	快　捷　键	说　　明
	启动调试	F5	在程序执行过程中启动或继续
	全部中断	Ctrl+Alt+Break	停止当前正在运行的应用程序
	停止调试	Shift+F5	停止调试
	重新启动	Ctrl+Shift+F5	停止当前正在调试的程序，然后重新启动应用程序
	显示下一语句	Alt+数字键*	显示下一语句
	逐语句	F11	如果当前行包括一个方法或函数的调用，则单击该按钮可以使调试单步执行当前行中的方法或函数
	逐过程	F10	如果当前行包括一个方法或函数的调用，则单击该按钮不会逐语句执行当前行中的方法或函数，而是执行到调用的一行

续表

工具栏按钮	图 标 文 本	快 捷 键	说 明
	跳出	Shift+F11	如果当前行在一个方法或函数内，则将执行完该方法或者函数，然后调试器停止在调用行之后的这一行
十六进制	十六进制显示		十六进制显示
	即时	（Ctrl+D，I）	选择调试的窗口
	断点	（Ctrl+D，B）	普通断点。实心的标志符号指示断点已启用。空心的标志符号指示断点已禁用

接下来就可以通过在“调试”菜单中选择“启动调试”、“逐语句”或“逐过程”命令来执行程序并调试，也可以通过右键单击可执行代码中的某行，然后从快捷菜单中选择“运行到光标处”命令。网站第一次调试时，会弹出如图 17.11 所示的询问窗口。

单击“确定”按钮后将在浏览器中运行程序，此时应用程序启动并一直运行到断点，如图 17.12 所示。可以在任何时刻中断执行，以检查值，修改变量，或检查程序状态。

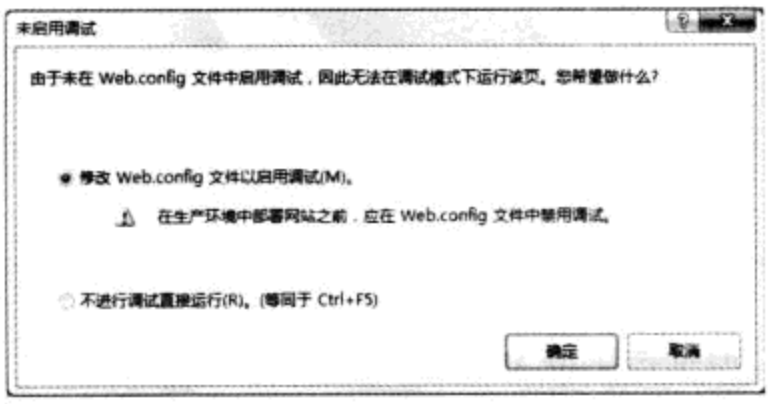


图 17.11 启动调试

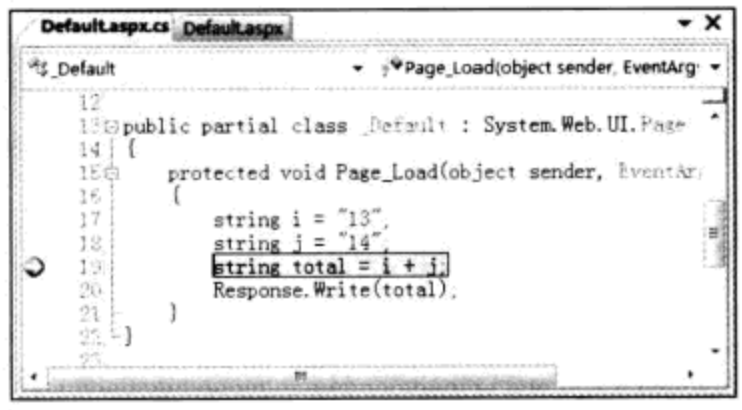


图 17.12 “启动调试”、“逐语句”或“逐过程”运行结果图

下面按 F10 键以“逐过程”方式进行单步执行来调试程序，调试运行结果如图 17.13 所示，可以看到当执行到断点的下一行时，将鼠标再移到定义的字符串变量 total 时，可以发现其值为“1314”，显然表示的是字符串变量“13”和“14”相加。

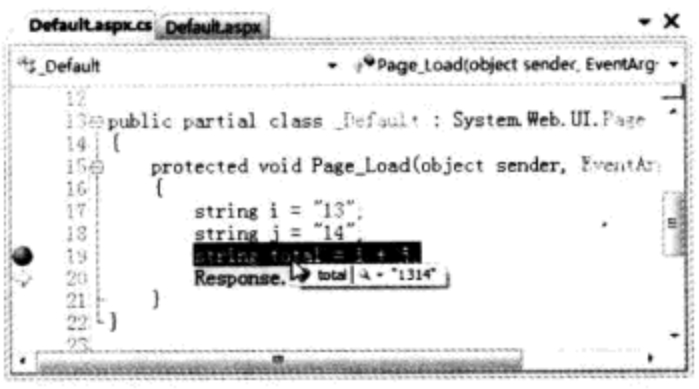


图 17.13 查看断点中设定的变量值

单步执行是最常见的调试过程之一，即每次执行一行代码。“调试”菜单中提供了 3 个逐句通过代码的命令：即逐语句、逐过程和跳出。

“逐语句”和“逐过程”的差异仅在于它们处理函数调用的方式不同。这两个命令都指示调试器执行下一行的代码。如果某一行包含函数调用，“逐语句”仅执行调用本身，然后在函数内的第一个代码行处停止。而“逐过程”执行整个函数，然后在函数外的第一行处停止。如果要查看函数调用的内容，则使用“逐语句”。若要避免单步执行函数，那么最好使用“逐过程”。

位于函数调用的内部并想返回到调用函数时，可以使用“跳出”。“跳出”将一直执行代码，直到函数返回，然后在调用函数中的返回点处中断。

17.2.3　中断执行

当应用程序执行到达一个断点或发生异常，调试器就会中断程序的执行。也可以通过单击“全部中断”命令手动中断执行，如图 17.14 所示。这时调试器将停止所有在调试器下运行的程序的执行。但程序并不退出，而且可以随时恢复执行。调试器和应用程序现在处于中断模式。

17.2.4　停止执行

停止调试意味着终止当前正在调试的程序并结束调试会话。与中断执行不同，中断执行意味着暂停正在调试的进程的执行，但调试会话仍处于活动状态。

可以通过选择菜单中的“调试”/“停止调试”命令或单击“调试”工具栏中的“■”按钮来结束运行和调试。也可以退出正在调试的应用程序，调试将自动停止。停止执行实际运行结果如图 17.15 所示。

17.2.5　运行到指定位置

如果在调试过程中，想执行到代码中的某一行，然后中断，可以通过在要中断的位置设置断点，接着在“调试”菜单中选择“启动”或“继续”；也可以在代码窗口中右键单击某行，并从快捷菜单中选择“运行到光标处”，如图 17.16 所示，程序将会运行到光标所在行中断；同样也可以在“反汇编”窗口中右键单击某行，并从快捷菜单中选择“运行到光标处”，如果“反汇编”窗口没有显示，那么从“调试”菜单中选择“窗口”/“反汇编”。“反汇编”窗口只能在中断模式下才能进行查看。

图 17.14　全部中断

图 17.15　停止调试

图 17.16　运行到光标处

17.3　程序错误处理

专题讲座：光盘\MR\Video\17\错误处理.exe

➤➤➤视频速递：读者可以直观地掌握如何进行错误处理。

通过上一节内容的学习，读者对如何调试程序有了初步的了解，下面将分别对服务器故障排除和 ASP.NET 中的异常处理做详细介绍。

17.3.1 服务器故障排除

在 Visual Studio 中测试文件系统网站时，ASP.NET Development Server 将自动运行。某些情况下，使用 ASP.NET Development Server 会产生错误。本节介绍 Web 服务器可能产生的错误，并提供相应的解决办法。

1. Web 服务器配置不正确

此错误显示如下。

```
The web server is not configured correctly. See help for common configuration errors. Running the web page outside of the debugger may provide further information.
```

造成该错误的原因以及常见的解决办法大概包括以下几种。

☑ 原因 1：网站的执行权限不够

解决办法：打开 IIS，选择对应网站属性，在“主目录”选项卡里面选择执行权限为“脚本和可执行文件”。

☑ 原因 2：身份验证方式不正确

解决办法：打开 IIS，在“网站”节点下选择对应网站，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，打开网站的属性窗口，选择“目录和安全性”选择卡，单击“匿名访问和身份验证控制”区域中的“编辑”按钮，打开身份验证方法，勾选“启用匿名访问”和“集成 Windows 身份验证”复选框。

2. IIS 管理服务没有响应

当 IIS 管理服务没有响应时，会发生“安全检查失败，因为 IIS 管理服务没有响应”的错误，这通常表示 IIS 的安装有问题。

解决此错误的方法如下：

首先使用“管理工具”中的“服务”工具验证该服务是否正在运行，然后按照以下方法进行操作：

☑ 使用控制面板中的“添加/删除程序”重新安装 IIS。

☑ 使用控制面板中的“添加/删除程序”从计算机中删除 IIS 并重新安装 IIS。

执行这两个步骤中的任一步骤后，重新启动计算机。

3. 未安装 ASP.NET

当用户尝试调试的计算机上未正确安装 ASP.NET 时，会发生“未安装 ASP.NET”的错误。此错误可能意味着从未安装过 ASP.NET，或者先安装了 ASP.NET，然后又安装了 IIS。

解决此错误的方法如下：

选择“开始”菜单中的“运行”命令，打开“运行”窗口，在“运行”窗口输入下列命令卸载 IIS：

```
\WINNT\Microsoft.NET\Framework\version\aspnet_regiis -i
```

其中，version 表示安装在用户计算机上的.NET Framework 的版本号（例如，v1.0.370）。

对于 Windows Server 2003，可以使用控制面板中的“添加/删除程序”来安装 ASP.NET。

4．连接被拒绝

服务器报告以下错误。

```
10061-Connection Refused
Internet Security and Acceleration Server
```

如果计算机在受 Internet Security and Acceleration Server（SA Server）保护的网络上运行，并且满足以下条件之一，就会发生此错误。

☑　客户端未安装防火墙。

☑　Internet Explorer 中的 Web 代理配置不正确。

避免此问题的方法如下。

（1）安装防火墙客户端软件，如 ISA 客户端。

（2）修改 Internet Explorer 中的 Web 代理连接设置，以跳过用于本地地址的代理服务器。

5．不能使用静态文件

在文件系统网站中，静态文件（如图像和样式表）受到 ASP.NET 授权规则的影响。例如，如果禁用了对静态文件的匿名访问，匿名用户则不能使用文件系统网站中的静态文件。但是，将网站部署到运行 IIS 的服务器时，IIS 将提供静态文件而不使用授权规则。

17.3.2　ASP.NET 中的异常处理

上面已经排除了服务器中存在的一些故障，接下来要处理的是程序当中存在的一些错误，通常把代码中存在的错误叫异常（Exception），导致异常状况的原因很多，除了用户输入错误的数据，数据库或网络发生问题也会造成异常状况的发生。本小节主要介绍如何处理 ASP.NET 程序中的异常。

调试异常是开发功能强健的 ASP.NET 应用程序的重要一步。若要调试未处理的 ASP.NET 异常，则需要确保调试器能够在发生这些异常时停止。ASP.NET 运行库具有一个顶级异常处理程序，因此默认情况下，调试器从不在发生未处理的异常时中断。若要通知调试器在发生异常时中断，必须转到“异常”对话框，然后在该对话框内选中发生异常名称后的复选框，如图 17.17 所示。

在大部分情况下，开发人员不希望异常状况发生导致程序结束，因此，可以使用 try…catch 程序语句块捕捉程序中的 Exception 对象，再使用自定义的程序逻辑处理异常状况。如果有需要，用户也可以使用多重 try…catch 语句块（如一个 try 块后跟一个或多个 catch 子句），针对不同的异常状况做不同的处理，图 17.18 为 try…catch 的示意图。

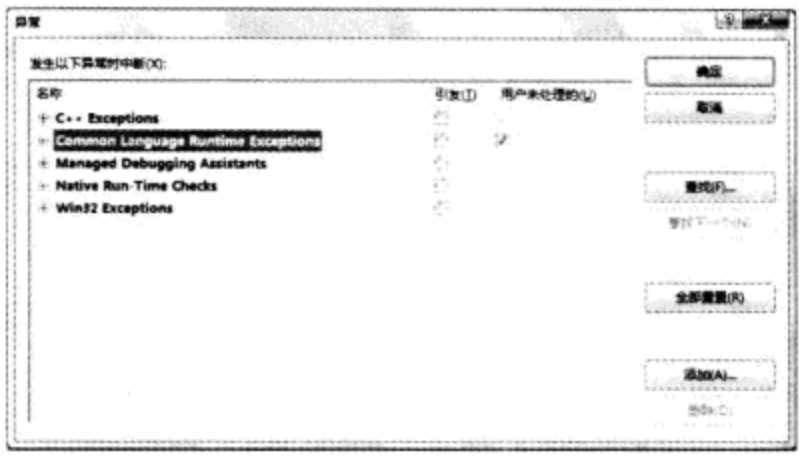


图 17.17 “异常”对话框

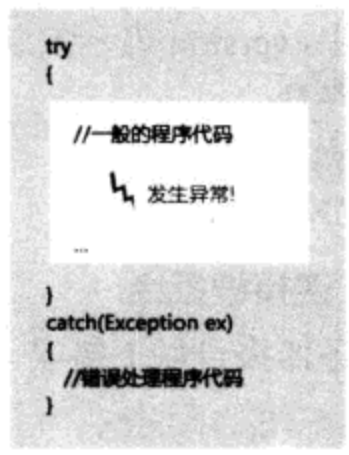


图 17.18 C#中可用 try…catch 捕获异常

根据图 17.19 可知，为防范无法预知的异常，可将程序主体用一个 try 包裹起来，一旦有异常发生，无论发生在哪一行，try 的“catch”会将它捕捉下来，接着便可加入错误处理程序。

学习手册 说明 所有.NET 里的异常都可被称为 Exception。虽然说异常多种多样，但基本上它们都是 Exception。这个说法可以打一个简单的比喻，就像是无论黑猫、白猫、花猫，它们都是“猫”一样。

这里来简单看一个实际应用的例子，该例子用来实现商品订单价格总和的计算，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    int NormalBccd = Convert.ToInt32(TextBox1.Text);
    int MuseumBccd = Convert.ToInt32(TextBox2.Text);
    int BccdTotal = NormalBccd + MuseumBccd;
    Label1.Text = BccdTotal.ToString();
}
```

学习手册 说明 上述事件代码中，定义了两个整型变量 NormalBccd 和 MuseumBccd 用来存储用户输入商品订单价格，并且都使用了 Convert.ToInt32 来将用户输入的值转换为整数即进行一个强制类型转换操作。当用户输入的值无法进行类型转换时便会产生异常。

正确输入商品价格后计算的结果如图 17.19 所示。

图 17.19 输入两个正确的商品订单价格后的汇总

接下来设定一下该实例可能发生的两种异常。

学习手册 说明 必须先识别出不同的异常，才能针对不同的异常显示不同的报错信息，或者给予不同的错误处理方式。

（1）在“编程词典标准版”文本框中输入字符串“明日科技”，即让 TextBox1 获取的值无法转换成整数的值，此时将产生的异常是 FormatException，如图 17.20 所示。

（2）在“编程词典标准版”文本框中输入一个整数 999999999999，即使输入的是一个整数，可是这个整数大于 int 变量允许的范围，此时将会产生的异常是 OverflowException，

如图 17.21 所示。

图 17.20　异常发生的原因可能随输入值而有所不同

图 17.21　输入 999999999999 时产生的 OverflowException 异常

根据以上两种情况产生的异常信息，开发人员可以在程序中加入 try…catch 来捕捉异常，并编写不同的错误处理方式代码。具体代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        int NormalBccd = Convert.ToInt32(TextBox1.Text);
        int MuseumBccd = Convert.ToInt32(TextBox2.Text);
        int BccdTotal = NormalBccd + MuseumBccd;
        Label1.Text = BccdTotal.ToString();
    }
    catch (FormatException fex)
    {
        Label1.Text = "请输入数字！";
    }
    catch (OverflowException oex)
    {
        Label1.Text = "输入的数据可能过大造成的错误！";
    }
    catch (Exception ex)
    {
        Label1.Text = "系统忙碌中...请稍后再试！";
    }
}
```

学习手册 技巧　编写程序时只能针对想到的异常进行处理，为了保险起见，通常会在最后加上 Exception 的错误处理程序，来涵盖所有其他没有设想到的异常。但需要注意的是，若要加入一段捕捉 Exception 的异常处理程序，则 catch (Exception ex)这行必须放在所有 catch 的最后面。

设定以上捕捉异常处理程序代码后，简单测试一下，当用户输入一个日期时，因为日期字符串无法转换成整型，因此会产生一个 FormatException 异常，catch 捕获该异常后就会显示“请输入数字！”提示信息，如图 17.22 所示。

最后来讲解一下在 try…catch 里加入 finally 语句块形如 try…catch…finally，凡是 finally 语句块中的程序无论有无异常发生都会被执行。如图 17.23 所示实现的是程序顺利执行，

没有任何异常情况发生。

图 17.22　捕获 FormatException 异常时显示的提示信息

使用 catch 子句是为了允许处理异常（例如，通过记录非致命错误）。无论是否引发了异常，使用 finally 子句即可执行清理代码。如果分配了昂贵或有限的资源（如数据库连接或流），则应将释放这些资源的代码放置在 finally 块中，简单地总结如下：在 try 块中获取并使用资源，在 catch 块中处理异常情况，并在 finally 块中释放资源。

try 语句可以仅与 finally 子句使用，即 try…finally 形式，在 finally 子句中可以对系统资源进行释放操作。

如图17.24所示的是程序执行中发生一个异常，此时会先执行catch里的错误处理程序，然后再执行 finally 语句块中的程序代码。

图 17.23　程序顺利执行没有任何异常，最后执行 finally 语句块

图 17.24　程序中发生异常并用 catch 捕获，最后执行 finally 语句块中的程序代码

同样以先前计算商品订单价格总和为例，在 try…catch 里加入一个 finally 语句块，代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        int NormalBccd = Convert.ToInt32(TextBox1.Text);
        int MuseumBccd = Convert.ToInt32(TextBox2.Text);
        int BccdTotal = NormalBccd + MuseumBccd;
        Label1.Text = BccdTotal.ToString();
    }
    catch (Exception ex)
    {
        Label1.Text = ex.Message;
    }
    finally
```

```
    {
        Label1.Text += "<br/>" +"finally 语句任何时候都将被最后执行！";
    }
}
```

如图 17.25 所示，在商品价格文本框中分别输入了 368 和 698 后，可以看到运行结果顺利地显示了计算后的总价格及 finally 语句块中的信息。

这里为了使程序产生一个异常，在第一个文本框中输入一个无法转换成整数的字符串“明日科技”，此时便会看到异常处理程序显示的是“输入字符串的格式不正确。finally 语句任何时候都将被最后执行！”，如图 17.26 所示。

图 17.25　程序顺利执行并显示 finally 语句块中的信息

图 17.26　即使发生异常，仍然执行 finally 语句块中的程序代码

编程信条：编程经验的积累就是财富的积累

一个编译通过的程序不代表它是一个没有错误的程序，.NET 使用异常处理所有的错误。异常的提示和编译的提示比起来就不是那么明确了。有的时候虽然它会提示某句语句出错，但是错误的根源往往不在这句语句上，错误可能发生在另外一个类甚至另外一个程序集中。有的时候异常的提示信息甚至是错误的，按照这个方向去检查程序根本发现不了任何错误，处理此类的异常就需要靠经验了，所以编程经验的积累就是财富的积累！

17.4　本 章 小 结

本章详细介绍了错误类型、程序调试、错误处理的相关内容，并且学会使用 Visual Studio 2008 集成开发工具进行程序调试。掌握调试与错误处理对开发程序而言是非常重要的，当出现错误时，如果不能及时准确地解决，那么对整个程序的影响是非常大的。所以，读者学好调试与错误处理是至关重要的。

学习笔记

第 4 篇

实战篇

书山有路勤为径
学海无涯苦作舟

本篇主要内容：

- 第 18 章　开发网站留言板
- 第 19 章　文件上传与下载
- 第 20 章　AJAX 无刷新聊天室
- 第 21 章　实现会员密码找回功能
- 第 22 章　完美实现网络硬盘

本篇学习流程：

第 18 章

开发网站留言板

（名师课堂：39 分钟）

留言板在网络上越来越受到人们的青睐，它能够在用户和访客之间建立起桥梁关系。用户在留言板上可以发表自己的一些看法，而访问者可以对用户的看法及时作出回应，具有良好的互动效果。通过本章的学习，读者可以达到以下目的：

- 熟悉并使用文本在线编辑器 FreeTextBox 控件
- 掌握 ASP.NET 母版页的使用
- 熟悉 CSS 样式统一网站风格
- 掌握配置 Web.Config
- 掌握编写网站公共类

18.1　网站留言板概述

留言板不仅能够满足用户的留言要求，还能够让访客在查看留言内容后，对留言的内容发表自己的意见。本章主要介绍一个较简单留言板的具体实现过程，其运行结果如图 18.1 所示。

18.1.1　功能设计与业务流程

留言板包括两种操作用户：管理员用户和普通用户。

☑　管理员用户：管理员用户在前台页面通过验证后，可以直接登录到后台，对留言内容进行管理。

☑　普通用户：普通用户可对留言主题进行查看，并发表自己的意见。

留言板程序的业务流程如图 18.2 所示。

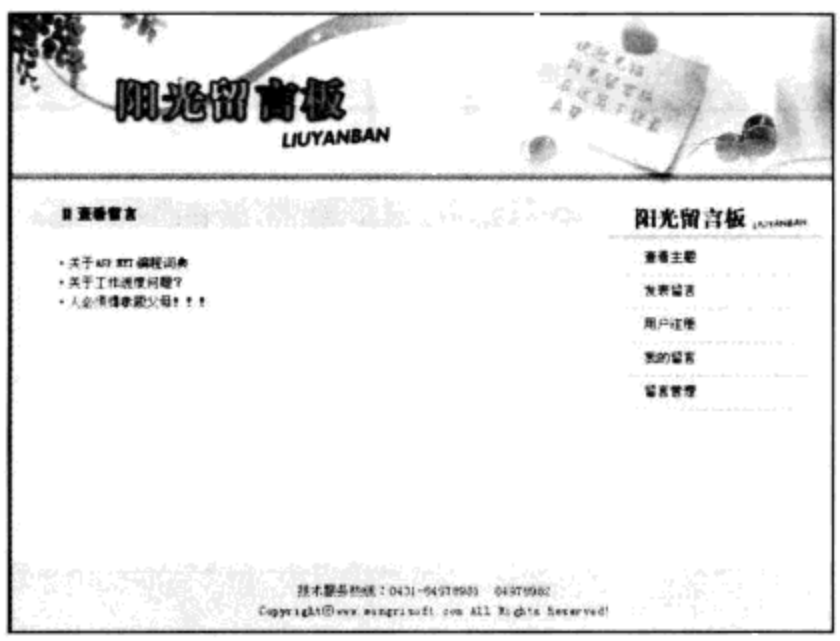


图 18.1　留言板主页面

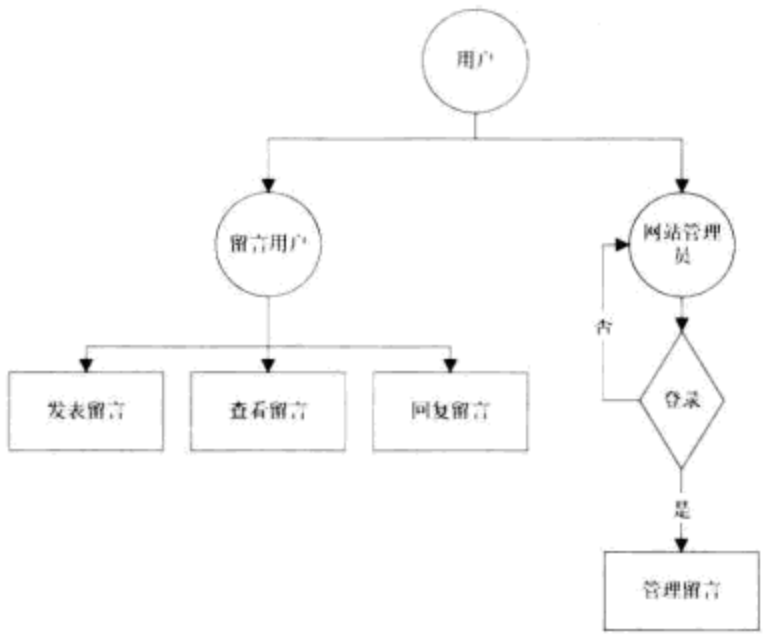


图 18.2　留言板业务流程图

18.1.2　数据库设计

本系统采用了 SQL Server 2005 数据库，其中主要用到了 3 个数据表，分别为用户信息表（tb_User）、留言表（tb_LeaveWord）和回复表（tb_Reply）。

☑　tb_User（用户信息表）

用户信息表主要用来保存注册用户的基本信息，tb_User 数据表的结构如表 18.1 所示。

表 18.1　用户信息表（tb_User）的结构

字　段	类　型	长　度	是否可为空	说　明
ID	int	4	否	主键（自动编号）
Uid	nvarchar	20	否	用户姓名
Sex	nvarchar	2	否	性别
WebSite	nvarhcar	50	是	主页

续表

字　段	类　型	长　度	是否可为空	说　明
Email	nvarchar	30	是	E-mail
QQ	nvarchar	20	是	QQ
IP	nvarchar	20	否	IP 地址
Popedom	int	4	否	权限

☑ tb_LeaveWord（留言表）

留言表主要用来保存用户的留言信息，tb_LeaveWord 数据表的结构如表 18.2 所示。

表 18.2　留言表（tb_LeaveWord）的结构

字　段	类　型	长　度	是否可为空	说　明
ID	int	4	否	主键（自动编号）
Uid	nvarchar	20	否	留言人姓名
Subject	nvarchar	50	否	留言主题
Content	ntext	16	是	留言内容
DateTime	datetime	8	否	留言时间
IP	nvarchar	20	否	IP 地址

☑ tb_Reply（回复表）

回复表主要用来保存访客对相关留言内容发表的意见，tb_Reply 数据表的结构如表 18.3 所示。

表 18.3　回复表（tb_Reply）的结构

字　段	类　型	长　度	是否可为空	说　明
ID	int	4	否	主键（自动编号）
Uname	nvarchar	20	否	回复人姓名
Content	ntext	16	是	回复内容
DateTime	datetime	8	否	回复时间
ReplyID	Int	4	否	回复 ID
IP	nvarchar	20	否	IP 地址

18.2　开发前技术准备

专题讲座：光盘\MR\Video\18\开发前技术准备.exe

➤➤➤视频速递：使读者在开发前能够了解项目用到的技术。

在开发前，读者应该了解本项目具体都应用到了什么技术。只有了解所需要的技术，才能在开发时更加得心应手。

18.2.1　配置第三方 FreeTextBox 组件

留言板中用到了 FreeTextBox 组件（可到网上搜索，然后下载该组件），该组件是一

个在线文本编辑器，可以对文字以及图片内容进行处理，并将编辑数据保存到数据库中。该组件的配置步骤说明如下。

图 18.3　“添加引用”对话框

（1）将 FreeTextBox.dll 添加到项目中。

在“解决方案资源管理器”中右键单击项目，选择快捷菜单中的“添加引用”选项，在弹出的对话框中选择“浏览”选项卡，找到组件存放位置，单击“确定”按钮，系统将自动创建 Bin 文件夹，并将组件存放到该文件夹中。“添加引用”对话框如图 18.3 所示。

（2）设置 SupportFolder 属性。

将存放 FreeTextBox 组件的文件夹放到 aspnet_client 文件夹中，然后设置 SupplorFolder 属性为“aspnet_client/FreeTextBox/”。

（3）向页面中添加组件。

配置完成后，即可向页面中的位置添加组件。在向页面中添加组件前，先注册组件。在页面 HTML 源码顶部添加注册代码如下所示：

```
<%@ Register TagPrefix="FTB" Namespace="FreeTextBoxControls" Assembly=
"FreeTextBox" %>
```

在页面中适当的位置再添加 FreeTextBox 组件，代码如下所示：

```
<FTB:FreeTextBox id="FreeTextBox1" runat="Server" SupportFolder="aspnet_
client/FreeTextBox/" ButtonSet="Office2003" Height="200px" Width="400px" />
```

注册完成后，回到设计视图，选中 FreeTextBox 组件，进行相关属性设置。

（4）写入数据库。

完成以上配置后，就要使用该组件了，下面就以在 Button1_Click 事件向数据库插入数据为例介绍 FreeTextBox 组件的使用方法。

```
private void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)
    {
        SqlConnection myConn = new SqlConnection(" server=Donet\\Donet2005;
        database=db_Test; uid=sa; pwd=");
        SqlCommand myCmd = new SqlCommand("select * from test where id=2", myConn);
        myConn.Open();                                    //打开数据连接
        SqlDataReader myDr;                               //声明 DataReader 对象
        myDr = myCmd.ExecuteReader();                     //获得 DataReader 对象
        myDr.Read();                                      //使用 DataReader 对象读取数据
        Response.Write(myDr["content"].ToString());
        myDr.Close();                                     //关闭 DataReader 对象
        myConn.Close();                                   //关闭数据连接
    }
}
protected void Button1_Click(object sender, System.EventArgs e)
{
    SqlConnection myConn = new SqlConnection("server=(local);database=db_Test;
    uid=sa;pwd=");
```

```
        SqlCommand myCmd = new SqlCommand("insert into test (content) values('" +
        FreeTextBox1. Text + "')", myConn);
        myConn.Open();                                          //打开数据连接
        myCmd.ExecuteNonQuery();                                //执行 SQL 命令
        myConn.Close();                                         //关闭数据连接
    }
```

在 Web.Config 文件的 system.web 节中加入<pages validateRequest="false"/>，防止页面出现危险字符错误。

18.2.2 应用 Visual Studio 2008 母版页

使用 ASP.NET 母版页可以为应用程序中的页创建一致的布局。单个母版页可以为应用程序中的所有页（或一组页）定义所需的外观和标准行为。可以创建包含要显示内容的各个内容页。当用户请求内容页时，这些内容页与母版页合并然后将母版页的布局与内容页的内容组合在一起输出。

1. 母版页的优点

母版页提供了开发人员以传统方式创建页面的功能。这些传统方式包括重复复制现有代码、文本和控件元素；使用框架集对通用元素使用包含文件和使用 ASP.NET 用户控件等。母版页具有下面的优点。

- ☑ 使用母版页可以集中处理页的通用功能，以便可以只在一个位置上进行更新。
- ☑ 使用母版页可以方便地创建一组控件和代码，并将结果应用于一组页。例如，可以在母版页上使用控件来创建一个应用于所有页的菜单。
- ☑ 通过允许控制占位符控件的呈现方式，母版页使用户可以在细节上控制最终页的布局。
- ☑ 母版页提供一个对象模型，使用该对象模型可以从各个内容页自定义母版页。

2. 母版页的运行时行为

在运行时，母版页是按照下面的步骤处理的。

- ☑ 用户通过键入内容页的 URL 来请求某页。
- ☑ 获取该页后，读取@ Page 指令。如果该指令引用一个母版页，则读取该母版页。如果这是第一次请求这两个页，则两个页都要进行编译。
- ☑ 包含更新内容的母版页合并到内容页的控件树中。
- ☑ 将每个 Content 控件的内容合并到母版页中相应的 ContentPlaceHolder 控件中。
- ☑ 浏览器中呈现得到的合并页。

3. 母版页和内容页的路径

当请求某个内容页时，该内容页与母版页合并，并且该页在内容页的上下文中运行。例如，如果获取 HttpRequest 对象的 CurrentExecutionFilePath 属性，则无论是在内容页代码还是母版页代码中，路径都表示内容页的位置。

母版页和内容页不必置于同一文件夹中。只要内容页@Page 指令中的 MasterPageFile 属性解析为一个 Master 页，ASP.NET 就可以将内容页和母版页合并为一个单独已呈现的页。

4．母版页的使用

下面介绍 Visual Studio 2008 中如何使用母版页。

（1）在 Visual Studio 2008 中右键单击已创建的项目，在弹出的快捷菜单中选择“添加新项”菜单项，弹出“添加新项”对话框，如图 18.4 所示。

图 18.4　“添加新项”对话框

（2）选择“添加新项”中的“母版页”项，并在名称栏中输入自定义名称，单击“添加”按钮，完成母版页的添加，如图 18.5 所示。

（3）在母版页中进行布局。先将母版页中的 ContentPlaceHolder 控件剪切，页面布局完成后，再将 ContentPlaceHolder 控件粘贴到要显示内容页的地方，本实例中母版页布局如图 18.6 所示。

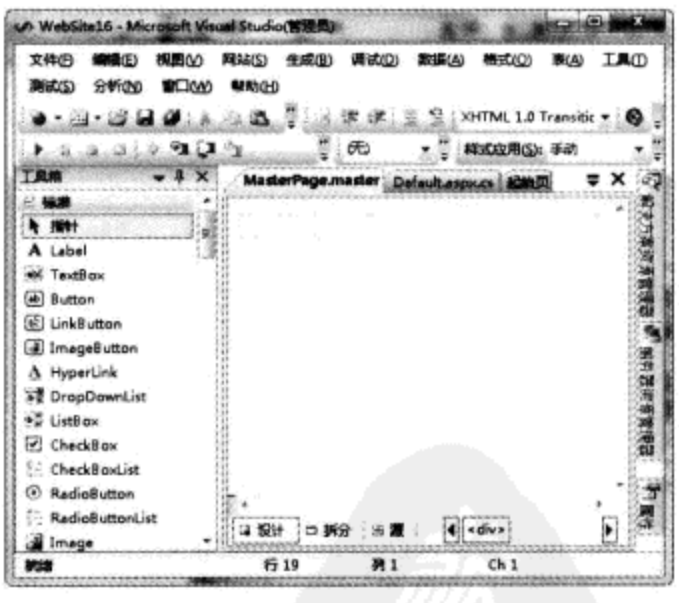


图 18.5　新建母版页布局

图 18.6　自定义母版页布局

（4）添加内容页。在布局好的页面中，右键单击 ContentPlaceHolder 控件，在弹出的快捷菜单中选择“添加内容页”，在新添加内容页的 Content 控件中进行布局。

18.2.3　定义 CSS 样式统一页面风格

定义某个外部层叠样式表（CSS）之后，可以将该样式表链接到单个 ASP.NET 网页，

以便将这些样式应用于该页上的元素。使用样式表可以指定 HTML 元素的格式设置样式。将层叠样式表链接到 ASP.NET 网页步骤说明如下。

（1）在“设计”视图中，将样式表文件（.css 文件）从解决方案资源管理器拖曳到网页上的任何位置。

（2）在“源”视图中，将样式表文件（.css 文件）从解决方案资源管理器拖曳到网页的<head></head>标记中。<head>标记中即插入一个新的 link 元素，如下面的代码所示。

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

18.3 主要开发过程

专题讲座：光盘\MR\Video\18\主要开发过程.exe

➤➤➤视频速递：使读者更加直观地了解本项目整个开发过程。

从本节开始，就将介绍网站留言板整个开发过程。通过图文并茂的形式介绍开发过程，使读者更容易理解和掌握，希望读者对本节能够认真阅读。

18.3.1 配置 Web.Config

为了方便数据操作和日后对网站的维护，本程序将一些配置参数放在 Web.Config 配置文件中。Web.Config 文件中主要的配置参数是连接数据库的字符串。下面是 Web.Config 配置文件中的代码。

```
<configuration>
    <appSettings>
        <add key="ConSql" value="server=DONET\DONET2005;database=db_18;uid=sa;
        pwd="/>
    </appSettings>
    <system.web>
        <pages validateRequest="false"></pages>
        <!--
            设置 compilation debug="true" 将调试符号插入
            已编译的页面中。但由于这会
            影响性能，因此只在开发过程中将此值
            设置为 true。
        -->
… …
    </system.web>
</configuration>
```

18.3.2 编写程序公共类

创建公共类可以减少重复代码的编写，并有利于后期系统代码维护。

创建公共类的方法为：在解决方案资源管理器中右键单击项目文件名称，在快捷菜单中选择“添加新项”，打开“添加新项”对话框，在该对话框中选择“类”，在“名称”文本框中输入 SqlData.cs。如图 18.7 所示。

图 18.7　创建类文件

1. SqlData 类中的全局变量

在 SqlData 类中声明 3 个全局变量，以便在下面的程序中能够重复使用，同时避免了重复编写相同的代码段。实现代码如下所示。

```
//引用数据库命名空间
using System.Data.SqlClient;
/// <summary>
/// SqlData 的摘要说明
/// </summary>
public class SqlData
{
    private SqlConnection sqlcon;       //声明一个 SqlConnection 对象
    private SqlCommand sqlcom;          //声明一个 SqlCommand 对象
    private SqlDataAdapter sqldata;     //声明一个 SqlDataAdapter 对象
    //以下为该类中的其他方法
        … …
}
```

2. SqlData 类中的构造函数

构造函数中包含连接数据库的字符串，当声明一个类的对象时，将自动连接数据库。实现代码如下所示。

```
#region 构造函数
/// <summary>
/// 构造函数，初始化时连接数据库
/// </summary>
public SqlData()
{
    //初始化数据库连接
    sqlcon = new SqlConnection(ConfigurationManager.AppSettings["conStr"]);
    sqlcon.Open();                          //打开链接
}
#endregion
```

3. SqlData 类中的 ExceSQL(string SqlCom)方法

ExecSQL 方法用来执行 SQL 语句，返回值为 Boolean 型，主要实现对数据库中数据进

行添加、修改和删除等功能，执行成功后返回 True，否则返回 False。

ExceSQL 方法主要技术要点如下：

（1）SqlCommand 类表示要对 SQL Server 数据库执行的一个 SQL 语句或存储过程；

（2）SqlCommand 类的 ExecuteNonQuery 方法用于执行 SQL 语句并返回受影响的行数。

该方法的完整设计代码如下所示。

```
#region  执行 SQL 语句
/// <summary>
/// 此方法用来执行 SQL 语句
/// </summary>
/// <param name="SqlCom">要执行的 SQL 语句</param>
/// <returns></returns>
public bool ExceSQL(string SqlCom)
{
    //初始化查询命令
    sqlcom = new SqlCommand(SqlCom, sqlcon);
    try
    {
        //执行命令查询
        sqlcom.ExecuteNonQuery();
        return true;
    }
    catch
    {
        return false;
    }
    finally
    {
        sqlcon.Close();          //关闭数据库连接
    }
}
#endregion
```

4. SqlData 类中的 ExceDS(string SqlCom)方法

ExecDS 方法用来返回 DataSet 类型的数据，并将数据填充到数据集中，执行成功后返回数据集，操作完成后断开与数据库的连接。

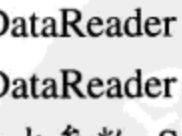
ExceDS 方法主要技术要点如下：

（1）SqlDataAdapter 类表示用于填充 DataSet 和更新 SQL Server 数据库的一组数据命令或一个数据库连接；

（2）SqlDataAdapter 类的 Fill 方法是可重载的，该方法主要实现将数据填充到 DataSet 中。具体代码如下所示。

```
#region  返回 DataSet 类型数据
/// <summary>
/// 此方法返回一个 DataSet 类型
/// </summary>
/// <param name="SqlCom">要执行的 SQL 语句</param>
/// <returns></returns>
public DataSet ExceDS(string SqlCom)
{
    try
    {
```

```
            sqlcom = new SqlCommand(SqlCom, sqlcon);          //初始化 SQL 查询命令
            sqldata = new SqlDataAdapter();                   //初始化数据适配器
            sqldata.SelectCommand = sqlcom;                   //设置查询命令
            DataSet ds = new DataSet();
            sqldata.Fill(ds);                                 //填充查询结果
            return ds;
        }
        finally
        {
            sqlcon.Close();
        }
    }
    #endregion
```



DataSet 是 ADO.NET 结构的主要组件，它的功能是将数据源中检索到的数据存到缓存中。在典型的多层实例中，用于创建和刷新 DataSet 并依次更新原始数据的步骤包括：通过 DataAdapter 使用数据源中的数据生成和填充 DataSet 中的每个 DataTable；通过添加、更新或删除 DataRow 对象更改单个 DataTable 对象中的数据；调用 GetChanges 方法以创建只反映对数据进行的更改的第二个 DataSet；调用 DataAdapter 的 Update 方法，并将第二个 DataSet 作为参数传递；调用 Merge 方法将第二个 DataSet 中的更改合并到第一个中；针对 DataSet 调用 AcceptChanges，或者调用 RejectChanges 以取消更改。

5. SqlData 类中的 ExceRead(string SqlCom)方法

ExecRead 方法用来返回 SqlDataReader 类型的数据。实现代码如下所示。

```
#region  返回 SqlDataReader 类型的数据
/// <summary>
/// 此方法返回一个 SqlDataReader 类型的参数
/// </summary>
/// <param name="SqlCom"></param>
/// <returns></returns>
public SqlDataReader ExceRead(string SqlCom)
{
    sqlcom = new SqlCommand(SqlCom, sqlcon);          //初始化 SQL 执行命令
    SqlDataReader read = sqlcom.ExecuteReader();      //执行查询并返回查询结果
    return read;
}
#endregion
```


若要创建 SqlDataReader，必须调用 SqlCommand 对象的 ExecuteReader 方法，而不要直接使用构造函数。在使用 SqlDataReader 时，关联的 SqlConnection 正忙于为 SqlDataReader 服务，对 SqlConnection 无法执行任何其他操作，只能将其关闭。除非调用 SqlDataReader 的 Close 方法，否则会一直处于此状态。例如，在调用 Close 之前，无法检索输出参数。SqlDataReader 的用户可能会看到在读取数据时另一进程或线程对结果集所做的更改。但是，确切的行为与执行时间有关。当 SqlDataReader 关闭后，只能调用 IsClosed 和 RecordsAffected 属性。尽管当 SqlDataReader 存在时可以访问 RecordsAffected 属性，但是要始终在返回 RecordsAffected 的值之前调用 Close，以保证返回精确的值。

18.3.3 留言板主页设计

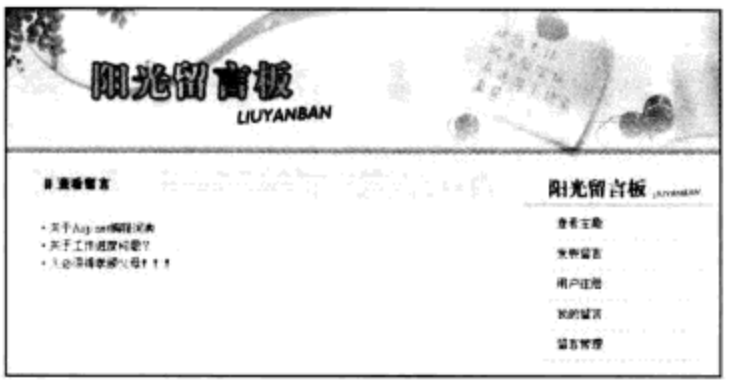


图 18.8 留言板主页

留言板主页主要是将数据库中检索到的留言主题信息显示出来，运行结果如图 18.8 所示。

1. 前台页面设计

首页 Default.aspx 的主要部分是用户控件 ShowSubject.ascx，该控件将数据动态绑定到客户端控件 Table 表格上，此功能的实现是将前台中的一个单元格绑定 GetList 方法，此方法的主要功能是在该单元格中添加一个 Table 表格，并将字段添加到动态生成的表格中。

用户首页中用到的主要控件如表 18.4 所示。

表 18.4 用户主页控件

控件类型	控件名称	数量	用途
HTML	Table	1	布局页面
UserControl	ShowSubject	1	显示主题

用户控件 ShowSubject.ascx 前台关键代码如下所示。

```
<table width="100%">
    <tr>
        <td colspan="3" rowspan="3">
        </td>
</tr>
<tr></tr>
    <tr></tr>
    <tr>
        <td colspan="3" rowspan="1">
        <%=GetList()%>
        </td>
    </tr>
</table>
```

2. 后台功能代码

前台表格中使用了 GetList 方法将数据绑定到表格中，GetList 方法代码如下所示。

```
public string GetList()
{
    SqlData da = new SqlData();
    SqlDataReader dr = da.ExceRead("select * from tb_LeaveWord ");
    string strBody = "<table width=150>";
    while (dr.Read())     //循环读取
    {
        strBody += "<tr><td class=tableBottom>•<a href=ShowWord.aspx?ID=" + dr
        ["ID"] + " >" + dr["Subject"].ToString() + "</a></td></tr>\n";
    }
    dr.Close();           //关闭阅读器
    strBody += "</table>";
    return strBody;
}
```

由于 GetList()方法中用到了 SqlDataReader 类，所以要添加命名空间 using System.Data.SqlClient。

18.3.4　发表留言模块设计

在留言板中，浏览者发表留言必须先进行注册，通过“用户注册”模块注册完成后，即可登录到系统中，单击导航栏中的“发表留言”项，可发表留言信息。发表留言页面运行结果如图 18.9 所示。

1．前台页面设计

这个页面中主要用到了 FreeTextBox 组件，用户可以根据该组件提供的功能来对文字进行编辑处理，该页面控件布局如图 18.10 所示。

图 18.9　发表留言页面运行结果

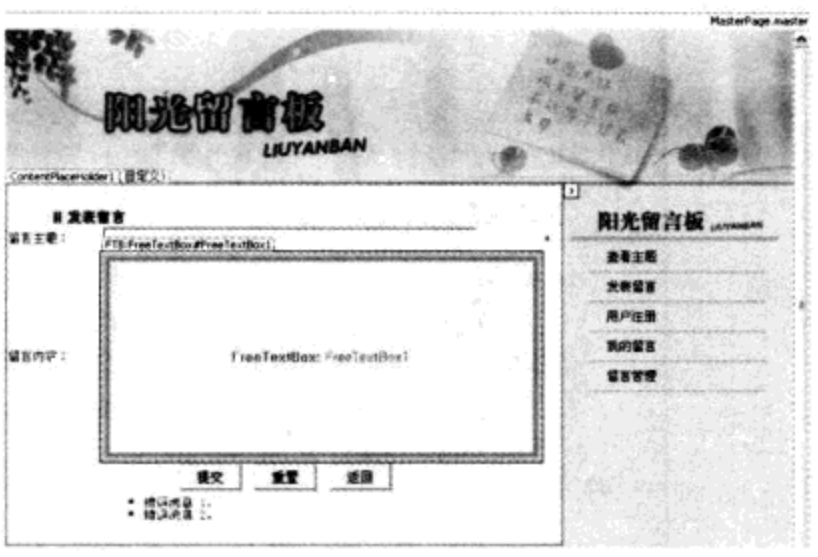


图 18.10　发表留言页面控件布局

发表留言页面中用到的主要控件如表 18.5 所示。

表 18.5　发表留言页面控件列表

控 件 类 型	控 件 名 称	数　量	用　途
HTML	Table	1	布局页面
HTML	Input(Reset)	2	重置/返回
标准	Button	1	提交
标准	TextBox	1	填写留言主题
标准	FreeTextBox	1	填写留言内容

发表留言页面中主要控件属性设置如下。

（1）Button 控件：页面中用到一个 Button 控件，主要完成向服务器端提供数据的功能，Button 控件主要属性设置了 ID 和 Text 属性，其属性值分别为 btnOK 和“提交”。

（2）FreeTextBox 第 3 方控件：页面中用到一个 FreeTextBox 控件，大部分属性都保持默认值，这里用户通过设置 Language 属性和 ButtonSet 属性来设置语言和控件外观，在本章技术要点中已经介绍了 FreeTextBox 组件的使用方法，这里不再赘述。

2. 后台功能代码

在页面的 Page_Load 事件中判断用户是否已登录，若已登录，进入“发表留言”页面；反之，进入登录页面，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    //判断用户是否登录
    if (Session["UserName"] == null)
    {
        Response.Redirect("Login.aspx");                    //如果没有登录，则返回登录页
    }
}
```

页面中的“提交”按钮完成向数据库提交数据的操作，主要功能代码如下所示。

```
protected void btnOK_Click(object sender, EventArgs e)
{
    SqlData da = new SqlData();
    //定义添加功能 SQL 语句
    string cmdtxt = "INSERT INTO tb_LeaveWord(Uid,Subject,Content,DateTime,IP)";
    cmdtxt += " VALUES('" + Session["UserName"].ToString() + "','" + this.
    TextBox1.Text + "'";
    cmdtxt += ",'" + this.FreeTextBox1.Text + "','" + DateTime.Now + "'";
    cmdtxt += ",'" + Request.UserHostAddress + "')";
    //执行 SQL 语句
    bool add = da.ExceSQL(cmdtxt);
    if (add == true)
    {
        Response.Write("<script language=javascript>alert('添加成功!');location=
        'Default.aspx'</script>");
    }
    else
    {
        Response.Write("<script language=javascript>alert('添加失败! ');
        location='javascript:history.go (-1)'</script>");
    }
}
```

18.3.5 留言信息查看页面设计

用户在首页选择相应的主题后，即可进入 ShowWord.aspx 页面查看详细内容，页面运行结果如图 18.11 所示。

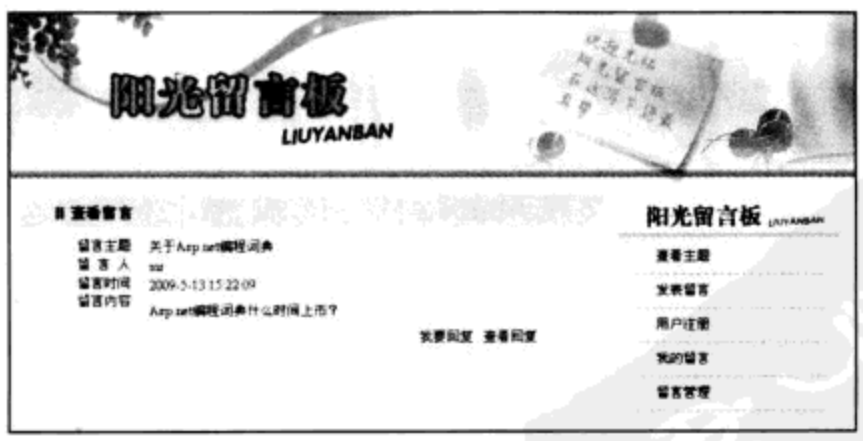


图 18.11 留言信息查看页面

1．前台页面设计

由于本系统是通过母版页来实现的，所以在布局页面时，只需要布局母版页的内容页，ShowWord.aspx 设计布局如图 18.12 所示。

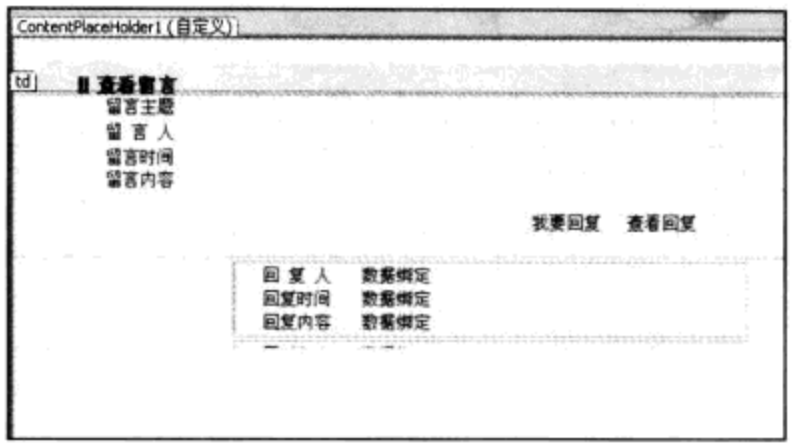


图 18.12　留言查看页面控件布局

留言信息查看页面中用到的主要控件如表 18.6 所示。

表 18.6　留言信息查看页面控件

控 件 类 型	控 件 名 称	数　量	用　途
HTML	Table	6	布局页面
标准	Label	2	显示数据
标准	DataList	1	显示检索到的数据
标准	LinkButton	6	我要回复/查看回复/控制翻页

信息查看页面中主要控件属性设置如下。

（1）Label：页面中用到两个 Label 控件，其中 labCount 用来显示 DataList 总页数，labNowPage 用来显示 DataList 当前页数，labNowPage 的 Text 属性值设为 1。两控件的 ForeColor 属性值都为 Red。

（2）LinkButton：该页面中，用到了 6 个 LinkButton 控件，其 ID 分别为 lnkbtnFeedBack、lnkbtnViewBack、lnkbtnTop、lnkbtnPrve、lnkbtnNext 和 lnkbtnLast，这 6 个控件的 ForeColor 属性设置都为 Black，Text 属性值分别为“我要回复”、“查看回复”、“首页”、“上一页”、“下一页”和“尾页”。

该页面中用到了数据显示控件 DataList，主要用来显示从数据库中检索出的符合条件的数据。

用户从 Visual Studio 2008 工具箱的“数据”栏中拖曳出 DataList 控件放置到页面中合适的位置，单击其右上角的“⊡”图标按钮，在弹出的快捷菜单（如图 18.13 所示）中选择“编辑模板”或是右键单击 DataList 选择快捷菜单中“编辑模板”中的“项模板（ItemTemplate）”，即可在 DataList 控件的项模板（ItemTemplate）中进行编辑。本页面中 DataList 控件项模板编辑如图 18.14 所示。

图 18.13　DataList 快捷菜单

图 18.14　DataList 项模板布局

在 DataList 控件内部主要用到了一个 Table 控件来布局。DataList 控件中将“回复人”、“回复时间”和“回复内容”分别绑定到数据表中相应字段上，由于它们的绑定方法一样，这里以绑定“回复人”为例，具体实现代码如下所示。

```
<tr>
    <td style="width: 80px">
        <span style="font-size: 9pt">回 复 人</span>
</td>
    <td colspan="2" style="width: 382px; text-align: left">
<%#DataBinder.Eval(Container,"DataItem.UName") %>
    </td>
</tr>
```

前台页面中的“留言主题”、“留言人”、“留言时间”以及“留言内容”分别绑定一个字符变量。由于它们绑定方法相同，这里以绑定“留言主题”为例，前台代码如下所示。

```
<tr>
    <td style="width: 50px; height: 17px;">
    </td>
    <td style="width: 80px; text-align: center; height: 17px;">
        留言主题</td>
    <td style="height: 17px; text-align: left;">
    <%=ShowSubject %>
    </td>
    <td style="width: 50px; height: 17px;">
    </td>
</tr>
```

2. 后台功能代码

由于在前台页面中绑定了几个变量，因此要在后台代码中首先声明这几个全局变量，并将它们设为 public（公有的）类型，代码如下所示。

```
//引用命名空间
using System.Data.SqlClient;
public partial class Default2 : System.Web.UI.Page
{
    public string ShowSubject, ShowTime, ShowContent,ShowName;
    //以下为其他事件和方法
    ……
}
```

在 Page_Load 事件中，首先根据来自主页传递的值检索数据表中相关数据，再给声明的变量赋值，并绑定一个自定义方法 dlBind()，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    string cmdtxt = "SELECT * FROM tb_LeaveWord WHERE ID='" + Request["ID"].
    ToString() + "'";
    SqlConnection Con = new SqlConnection(ConfigurationManager.AppSettings
    ["ConSql"]);
    Con.Open();                                          //打开数据连接
    SqlCommand Com = new SqlCommand(cmdtxt, Con);        //创建 Command 对象
    SqlDataReader dr = Com.ExecuteReader();              //获得 DataReader 对象
    dr.Read();                                           //使用 DataReader 读取数据
```

```
    if (dr.HasRows)
    {
        ShowSubject = dr["Subject"].ToString();
        ShowTime = dr["DateTime"].ToString();
        ShowContent = dr["Content"].ToString();
        ShowName = dr["Uid"].ToString();
    }
    dr.Close();
    this.dlBind();
}
```

页面中最关键的是 dlBind 方法的使用，该方法中实现了 DataList 控件分页技术，其代码如下所示。

```
public void dlBind()
{
    int curpage = Convert.ToInt32(labNowPage.Text);
    PagedDataSource ps = new PagedDataSource();
    SqlConnection mycon = new SqlConnection(ConfigurationManager.AppSettings
    ["ConSql"]);
    mycon.Open();
    string cmdtxt1 = "SELECT * FROM tb_Reply WHERE ReplyID=" + Request["ID"].
    ToString() + "";
    SqlDataAdapter MyAdapter = new SqlDataAdapter(cmdtxt1, mycon);
    DataSet ds = new DataSet();
    MyAdapter.Fill(ds, "tb_Reply");
    ps.DataSource = ds.Tables["tb_Reply"].DefaultView;
    ps.AllowPaging = true;                          //是否可以分页
    ps.PageSize = 3;                                //显示的数量
    ps.CurrentPageIndex = curpage - 1;              //取得当前页的页码
    lnkbtnPrve.Enabled = true;
    lnkbtnTop.Enabled = true;
    lnkbtnNext.Enabled = true;
    lnkbtnLast.Enabled = true;
    if (curpage == 1)
    {
        lnkbtnTop.Enabled = false;                  //不显示第一页按钮
        lnkbtnPrve.Enabled = false;                 //不显示上一页按钮
    }
    if (curpage == ps.PageCount)
    {
        lnkbtnNext.Enabled = false;                 //不显示下一页
        lnkbtnLast.Enabled = false;                 //不显示最后一页
    }
    labCount.Text = Convert.ToString(ps.PageCount);
    DataList1.DataSource = ps;
    DataList1.DataKeyField = "ID";
    DataList1.DataBind();
}
```

控制 DataList 翻页主要用到 LinkButton 控件，实现分页功能的代码如下所示。

```
//首页
protected void lnkbtnTop_Click(object sender, EventArgs e)
{
    labNowPage.Text = "1";
    dlBind();
```

```
}
//上一页
protected void lnkbtnPrve_Click(object sender, EventArgs e)
{
    labNowPage.Text = Convert.ToString(Convert.ToInt32(this.labNowPage.Text) - 1);
    dlBind();
}
//下一页
protected void lnkbtnNext_Click(object sender, EventArgs e)
{
    labNowPage.Text = Convert.ToString(Convert.ToInt32(this.labNowPage.Text) + 1);
    dlBind();
}
//尾页
protected void lnkbtnLast_Click(object sender, EventArgs e)
{
    labNowPage.Text = this.labCount.Text;
    dlBind();
}
```

18.3.6 留言信息管理设计

单击导航栏中的“留言管理”项，进入到登录页面，该页面是普通用户和管理员用户登录系统的入口。管理员登录成功后，即可进入留言信息管理页面，该页面显示所有注册用户的留言信息，并可以对留言信息进行查看、回复和删除。留言信息管理页面运行结果如图 18.15 所示。

1. 前台页面设计

留言信息管理页面控件布局如图 18.16 所示。

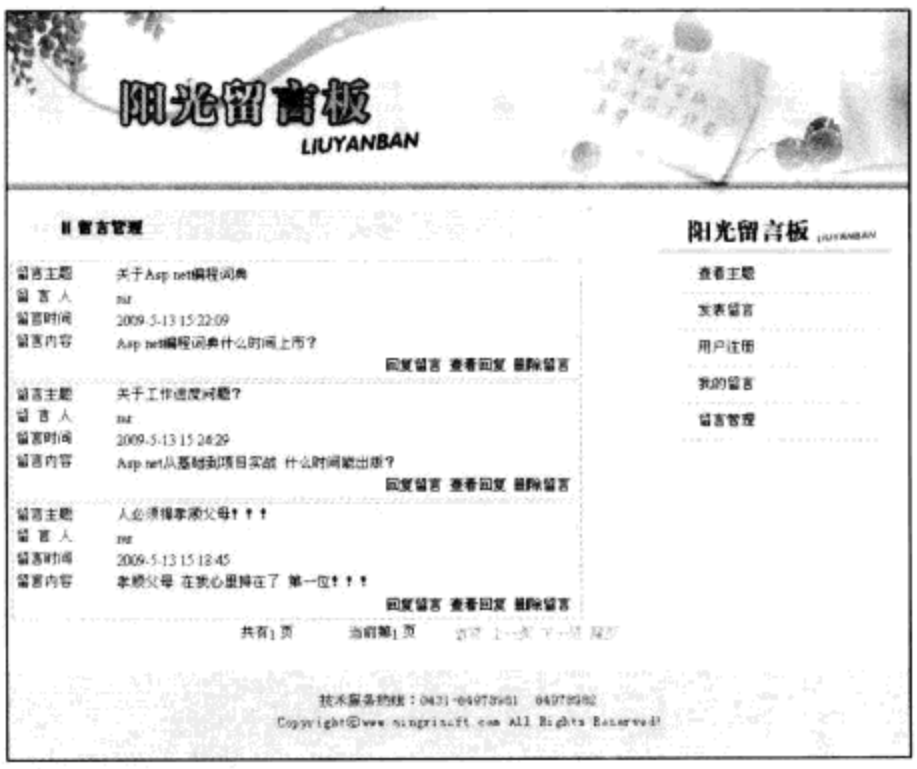


图 18.15 留言信息管理页面运行结果

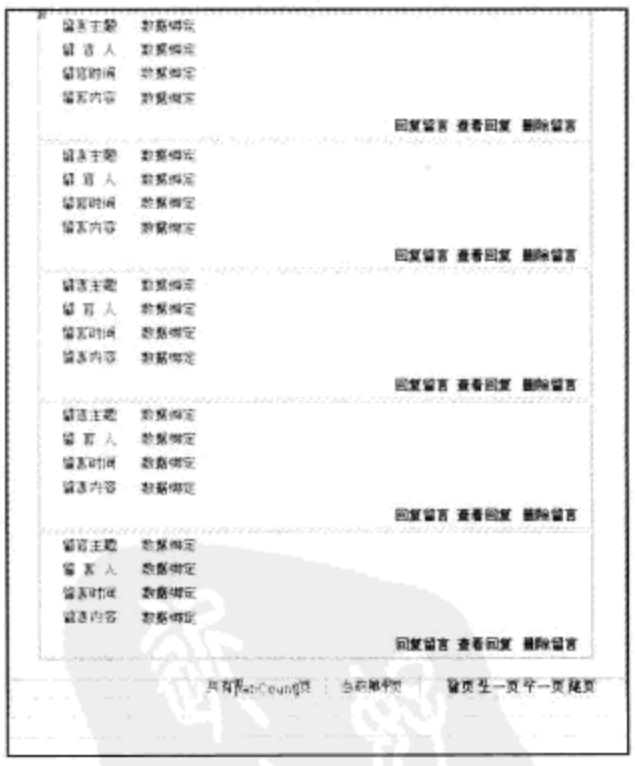


图 18.16 留言信息管理页面控件布局

由于前台控件布局、控件绑定以及控件属性设置和留言信息查看页中基本相同，这里就不再赘述，具体请参见留言信息查看模块。

2. 后台功能代码

由于前台页面中绑定的是变量，为了使数据能够正常显示，首先要声明几个全局变量，代码如下所示。

```
//引用命名空间
using System.Data.SqlClient;
public partial class Default2 : System.Web.UI.Page
{
    public string ShowSubject, ShowTime, ShowContent;        //声明的变量
//以下为其他事件和方法
……
}
```

在 Page_Load 事件中，首先判断用户是否登录，若未登录，则返回到登录页面；反之，进入管理页面并将数据绑定到 DataList 控件，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"] == null)
    {
        Response.Redirect("Login.aspx");                    //跳转到登录页
    }
    this.dlBind();
}
```

Page_Load 事件中的 dlBind 方法与留言信息查看页面中用到的方法相同，这里不再赘述。

该页面中具有删除留言功能，主要是通过 DataList 控件中的 LinkButton 控件来实现的，将 LinkButton 按钮的 CommandName 属性值设为 delete，然后在 DataList 控件的 DeleteCommand 事件中编写如下代码完成删除功能。

```
protected void DataList1_DeleteCommand(object source, DataListCommandEventArgs e)
{
    string strid = this.DataList1.DataKeys[e.Item.ItemIndex].ToString();
                                                    //获取当前 DataList 控件列
    string com = "Delete from tb_LeaveWord where ID='" + Convert.ToInt32(strid)
    + "'";
    SqlData da = new SqlData();
    da.ExceSQL(com);                                //执行 SQL 命令
    Page.Response.Redirect("LeaveWordManage.aspx");  //跳转页面
}
```

18.3.7　回复留言设计

回复留言模块的主要功能是对留言人的留言进行回复，以方便交流沟通。回复留言模块通过留言信息管理模块中传递过来的值，从留言表（tb_LeaveWord）中检索出相应的 ID 和留言主题，并将留言主题绑定到控件中，回复留言页面运行结果如图 18.17 所示。

1. 前台页面设计

回复留言页面控件布局如图 18.18 所示。

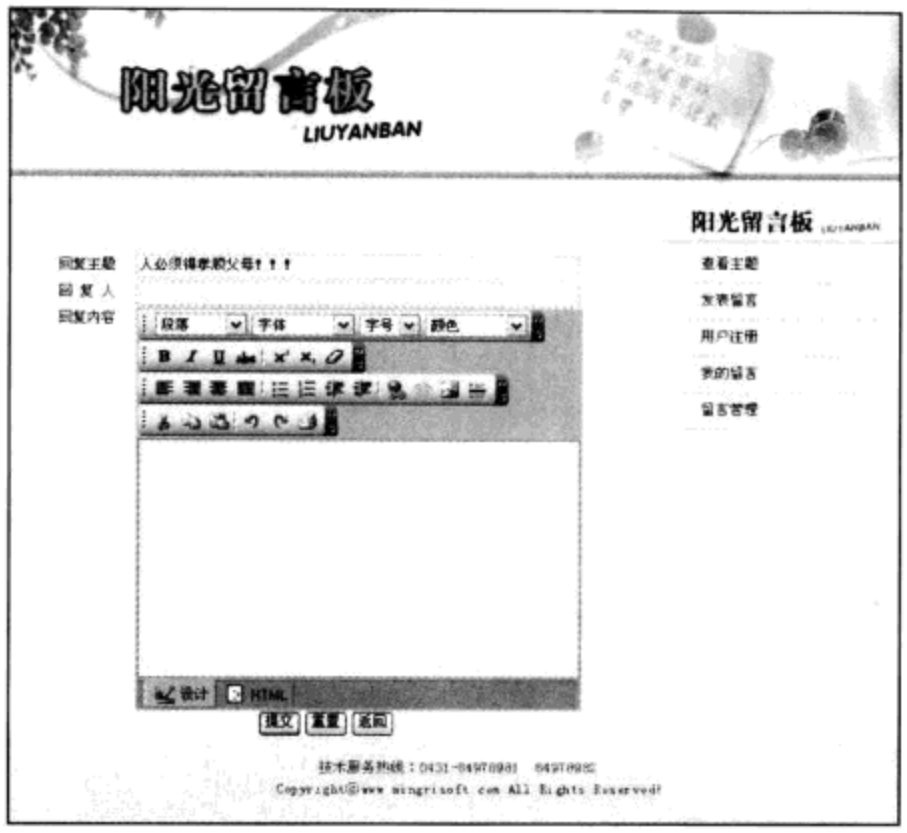


图 18.17　回复留言页面运行效果

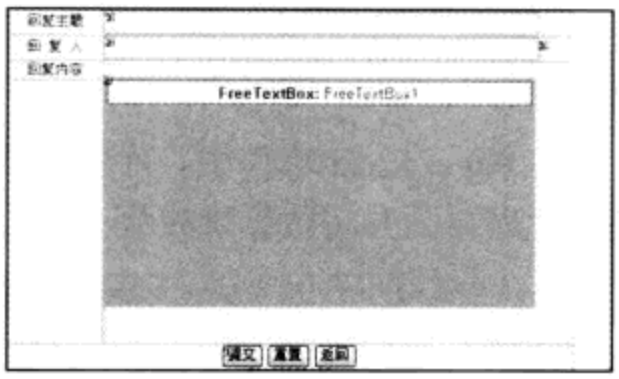


图 18.18　回复留言页面控件布局

回复留言页面中用到的主要控件如表 18.7 所示。

表 18.7　回复留言页面控件列表

控件类型	控件名称	数　量	用　途
HTML	Table	1	布局页面
HTML	Input(Reset)	2	重置/返回
标准	Button	1	提交
标准	TextBox	2	显示回复主题/填写回复人姓名
标准	FreeTextBox	1	填写回复内容

回复留言页面中主要控件属性设置如下。

(1)Button：页面中用到一个 Button 控件，主要完成向服务器端提供数据的功能，Button 控件主要属性设置了 ID 和 Text 属性，其属性值分别为 btnOK 和“提交”。

(2) TextBox：页面中用到了两个 TextBox 控件，ID 属性值分别为 txtSubject 和 txtUid，其中 txtSubject 主要用来绑定留言信息中的留言主题字段，它的 ReadOnly 属性值为 True；页面中 txtUid 主要是用来输入回复人姓名的，保持默认属性。

(3) FreeTextBox：页面中用到一个 FreeTextBox 组件，大部分属性都保持默认值，这里用户通过 Language 属性和 ButtonSet 属性分别设置语言和控件外观。

2. 后台功能代码

在页面的 Page_Load 事件中，通过从留言信息管理页面传递过来的值，从数据库中检索出数据并绑定到控件，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //建立数据库连接
```

```
        SqlConnection Con = new SqlConnection(ConfigurationManager.AppSettings
        ["ConSql"]);
        //打开数据库连接
        Con.Open();
        string id = Request["ID"].ToString();
        //定义 SQL 查询语句
        string cmdtxt = "SELECT * FROM tb_LeaveWord WHERE ID=" + id + "";
        SqlData da = new SqlData();
        SqlCommand Com = new SqlCommand(cmdtxt, Con);
        //执行 SQL 语句查询
        SqlDataReader dr = Com.ExecuteReader();
        //读取查询结果
        dr.Read();
        if (dr.HasRows)
        {
            txtSubject.Text = dr["Subject"].ToString();
        }
        dr.Close();
    }
}
```

页面中的“提交”按钮主要完成向数据库提交数据的操作，其功能代码如下所示。

```
protected void btnOK_Click(object sender, EventArgs e)
{
    SqlData da = new SqlData();
    //定义 SQL 语句
    string cmdtxt = "INSERT INTO tb_Reply(UName,Content,DateTime,ReplyID,IP)";
    cmdtxt += "VALUES('" + this.txtUid.Text + "','" + this.FreeTextBox1.Text + "',
    '" + DateTime.Now + "'";
    cmdtxt += "," + Request["ID"].ToString() + ",'" + Request.UserHostAddress
    + "')";
    //调用自定义方法执行 SQL 语句
    if (da.ExceSQL(cmdtxt))
    {
        Response.Write("<script language=javascript>alert('操作成功！');
        location='LeaveWordView. aspx?ID=" + Request["ID"] + "'</script>");
    }
    else
    {
        Response.Write("<script language=javascript>alert('操作失败！')
        </script>");
    }
}
```

18.4　本 章 小 结

本章主要介绍如何创建 ASP.NET 的留言板。通过对各模块关键技术的分析和对功能实现的讲解，能够清晰地了解 ASP.NET 中留言板的原理和编程思路。其中运用很多的控件对功能进行控制，通过大量的控件使用练习，读者在以后的编程过程中可以更好地运用这些控件。

第19章

文件上传与下载

（名师课堂：20分钟）

开发Web程序中，经常要涉及对文件的上传或下载操作。在以前的Web应用程序中，要实现文件上传是个很麻烦的事，但这一操作在Web应用程序中又会经常用到，因此令开发人员非常头痛。而在ASP.NET 3.5中，要实现文件的上传或下载不再是难事，它变得非常简单。通过本章的学习，读者可以达到以下目的：

- 了解上传文件和下载文件的流程
- 学会使用FileUpload控件实现文件上传
- 学会批量上传文件
- 学会动态添加控件的功能
- 掌握文件的下载

PDG

19.1　设 计 思 路

19.1.1　功能概述

文件上传功能主要利用 FileUpload 控件（文件上传控件）来完成，并通过按钮来实现动态添加 FileUpload 控件。本实例实现的具体功能如下：

- ☑ 单文件上传功能；
- ☑ 文件批量上传功能；
- ☑ 动态添加文件上传控件；
- ☑ 判断上传的文件是否为空；
- ☑ 文件下载。

19.1.2　程序业务流程图

本实例主要包括两种操作：文件上传和文件下载。

在文件上传页中可以通过使用“添加文件”按钮来动态添加文件上传控件，通过使用文件上传控件可以添加需要上传文件的路径。文件下载需要先选择文件名才可以将文件下载到本机。流程图如图 19.1 所示。

19.1.3　文件组织结构

下面给出文件上传和下载程序的文件夹及文件组织结构图，方便更好地学习和理解本程序，如图 19.2 所示。

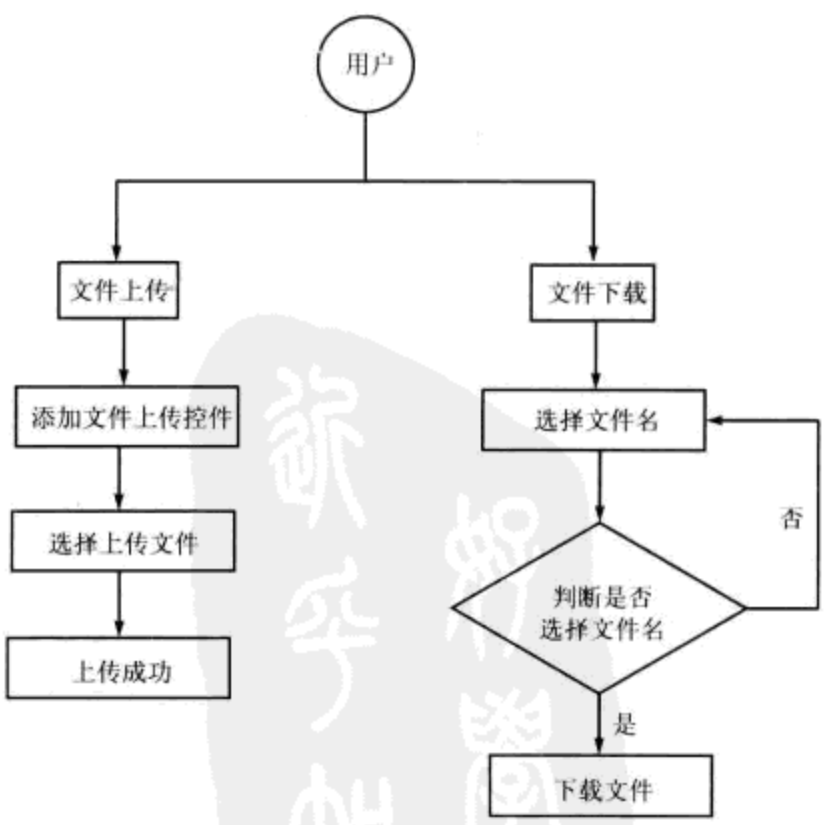


图 19.1　文件上传与下载的流程图

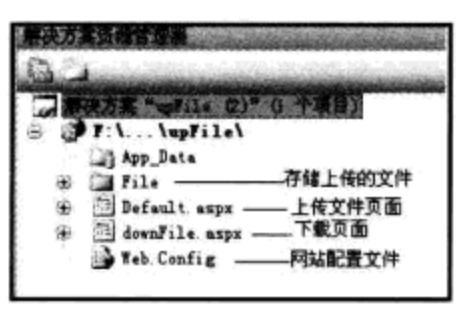


图 19.2　文件组织结构图

19.2 文 件 上 传

专题讲座：光盘\MR\Video\19\文件上传.exe

➤➤➤视频速递：使读者能够更直观地了解如何实现文件上传。

单个文件上传的实现过程，首先是利用 FileUpload 控件找到要上传文件的路径，通过 FileUpload 类中的 SaveAs 方法将上传文件保存在服务器指定文件夹中。如果同时上传多个文件（即批量上传），首先需要通过“添加文件”按钮来添加 FileUpload 控件，在 FileUpload 控件中选择要上传文件的路径，再循环将 FileUpload 控件中所指定的文件路径通过 SaveAs 方法将上传文件保存在服务器指定文件夹中。

19.2.1 实现关键技术

动态添加 FileUpload 控件（文件上传控件）通过 HtmlTableRow 对象和 HtmlTableCell 对象编辑服务器上的 HTML 的 table 元素来实现。HtmlTableRow 对象表示 HtmlTable 控件中的 tr 元素，HtmlTableCell 对象表示 HtmlTableRow 对象中的 td 和 th 元素，代码如下所示。

```
HtmlTableRow HTR = new HtmlTableRow();
HtmlTableCell HTC = new HtmlTableCell();
HTC.Controls.Add(new FileUpload());
HTR.Controls.Add(HTC);
```

文件上传通过 HttpFileCollection 对象获取客户端上传文件的集合，循环此集合使用 HttpPostedFile 对象来获取对单个文件的操作，代码如下所示。

```
HttpFileCollection HFC = Request.Files;
for (int i = 0; i < HFC.Count; i++)
{
    //访问指定的文件
    HttpPostedFile UserHPF = HFC[i];
}
//Path 对象中的 GetFileName 方法主要用来获取文件名，例如：
System.IO.Path.GetFileName(@"c:\mr\mr.txt");
```

19.2.2 功能实现

文件上传页面运行结果如图 19.3 所示。

主要实现步骤说明如下。

（1）创建 Web 窗体，命名为 Default.aspx，作为文件上传页面。

（2）在 Default 窗体中添加控件。主要控件及其用途如表 19.1 所示。

表 19.1 Default.aspx 页面中控件属性设置及其用途

控 件 类 型	控 件 名 称	用 途
标准/ ImageButton 控件	ImageButtonAdd	动态添加文件上传控件

续表

控 件 类 型	控 件 名 称	用　途
标准/ ImageButton 控件	ImageButtonUp	将文件上传
	ImageButtonDown	跳转到文件下载页面
标准/Label 控件	lblMessage	用于显示文件上传是否成功的提示信息
标准/FileUpload 控件	fileup	选择要上传文件

图 19.3　文件上传页面

（3）动态添加控件主要通过 SFUPC 自定义方法、InsertC 自定义方法和 GetInfo 自定义方法，SFUPC 自定义方法主要用来将文件上传控件集保存到 Session 中，此方法利用 ArrayList 数组存储在 ID 属性为 F 的表格中查找出类型为 FileUpload 的控件。最后将 ArrayList 存储在 Session 中。SFUPC 方法实现代码如下所示。

```
//该方法用于当前页面上传文件控件集保存到 Session 中
private void SFUPC()
{
    ArrayList AL = new ArrayList();//创建动态增加数组
    foreach (Control C in F.Controls)
    {
        //在表格中查找出 FileUpload 控件添加到 ArrayList 中
        if (C.GetType().ToString() == "System.Web.UI.HtmlControls.
        HtmlTableRow")
        {
            HtmlTableCell HTC = (HtmlTableCell)C.Controls[0];
            foreach (Control FUC in HTC.Controls)
            {
                if (FUC.GetType().ToString() == "System.Web.UI.WebControls.
                FileUpload")
                {
                    FileUpload FU = (FileUpload)FUC;
                    //添加 FileUpload 控件
                    AL.Add(FU);
                }
            }
        }
    }
    //把 ArrayList 添加到 Session 中
    Session.Add("FilesControls", AL);
}
```

在此方法中应用了 ArrayList 数组和 Session 引用命名空间 System.Collections。

在当前页面 Page_Load 事件中调用 SFUPC 方法将当前的文件上传控件保存到 Session 中，Page_Load 事件中的代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)     //首次执行页面
    {
        SFUPC();
    }
}
```

GetInfo 自定义方法的作用是将保存在 Session 中的文件上传控件集添加到表格中，主要是利用 ArrayList 数组保存 Session 中的文件上传控件集，再通过遍历 ArraylList 数组将文件上传控件添加到表格中。GetInfo 方法的代码如下所示。

```
//该方法用于将保存在 Session 中的文件上传控件集添加到表格中
private void GetInfo()
{
    ArrayList AL = new ArrayList();
    if (Session["FilesControls"] != null)
    {
        AL = (ArrayList)Session["FilesControls"];
        for (int i = 0; i < AL.Count; i++)
        {
            HtmlTableRow HTR = new HtmlTableRow();
            HtmlTableCell HTC = new HtmlTableCell();
            HTC.Controls.Add((System.Web.UI.WebControls.FileUpload)AL[i]);
            HTR.Controls.Add(HTC);
            F.Rows.Add(HTR);
        }
    }
}
```

InsertC 自定义方法是用来添加一个新的文件上传控件。此方法调用 GetInfo 方法将保存在 Session 中的文件上传控件添加到表格中。调用 GetInfo 方法结束后通过 HtmlTableCell 对象添加一个新的文件上传控件到表格上，最后利用 SFUPC 方法将表格上的文件上传控件保存到 Session 中。InsertC 方法代码如下所示。

```
//该方法用于添加一个文件上传的控件
private void InsertC()
{
    //实例化 ArrayList
    ArrayList AL = new ArrayList();
    this.F.Rows.Clear();      //清除 id 为 F 表格里的所有行
    GetInfo();
    //表示 HtmlTable 控件中的 <tr> HTML 元素
    HtmlTableRow HTR = new HtmlTableRow();
    //表示 HtmlTableRow 对象中的 <td> 和 <th> HTML 元素
    HtmlTableCell HTC = new HtmlTableCell();
    //在单元格中添加一个 FileUpload 控件
    HTC.Controls.Add(new FileUpload());
```

```
        //在行中添加单元格
        HTR.Controls.Add(HTC);
        //在表中添加行
        F.Rows.Add(HTR);
        SFUPC();
    }
```

文件上传的功能需要通过自定义方法 UpFile 来实现。此方法通过 HttpFileCollection 对象获取客户端上载文件的集合，循环此集合使用 HttpPostedFile 对象来获取单个已上传的文件，判断文件的大小是否为空，不为空将其保存在指定的文件夹中。UpFile 方法的代码如下所示。

```
    private void UpFile()          //该方法用于执行文件上传操作
    {
        //获取文件夹路径
        string FilePath = Server.MapPath("./") + "File";
        // 获取客户端上载文件的集合
        HttpFileCollection HFC = Request.Files;
        for (int i = 0; i < HFC.Count; i++)
        {
            //访问指定的文件
            HttpPostedFile UserHPF = HFC[i];
            try
            {
                //判断文件是否为空
                if (UserHPF.ContentLength > 0)
                {
                    //将上传的文件存储在指定目录下
                    UserHPF.SaveAs(FilePath + "\\" + System.IO.Path.GetFileName
                    (UserHPF. FileName));
                }
            }
            catch
            {
                lblMessage.Text = "上传失败！";
            }
        }
        if (Session["FilesControls"] != null)
        {
            Session.Remove("FilesControls");
        }
        lblMessage.Text = "上传成功！";
    }
```

在上传文件页面中，通过单击“添加文件”按钮来动态添加文件上传控件，在“添加文件”按钮的 Click 事件中调用自定义方法 InsertC 添加新的文件上传控件，Click 事件的代码如下所示。

```
    protected void ImageButtonAdd_Click(object sender, ImageClickEventArgs e)
    {
        InsertC();          //执行添加控件方法
        lblMessage.Text = "";
    }
```

上传文件时通过“上传文件”按钮的 Click 事件将文件上传到 Files 文件夹中。在此事件中主要调用 UpFile 方法来上传文件，再通过 SFUPC 方法来保存当前的文件上传控件。“上传文件”按钮的 Click 事件的代码如下所示。

```
protected void ImageButtonUp_Click(object sender, ImageClickEventArgs e)
{
    if (this.fileup.PostedFile.FileName != "")
    {
        UpFile();                                    //执行上传文件
        SFUPC();
    }
    else
    {
        Response.Write("<script>alert('上传文件不能为空!');location=Default.
        aspx</script>");
    }
}
```

19.3 文 件 下 载

专题讲座：光盘\MR\Video\19\文件下载.exe

➤➤➤视频速递：使读者能够更直观地了解如何实现文件下载。

文件的下载先通过 ListBox 控件显示文件名，再选择要下载的文件名，单击“下载”按钮，将文件保存到本机，即完成文件的下载功能。下面详细介绍文件下载功能的实现。

19.3.1 实现关键技术

1. Directory 对象的 GetFiles 方法

下载文件前需要指定文件名，使用 Directory 对象的 GetFiles 方法来获得指定路径中的所有文件名。语法：

```
public static string[] GetFiles
(
    string path
)
```

参数 path：用于检索文件的路径。

返回值：指定目录中文件名的 String 数组。文件名包含完整路径。

2. Path 对象的 GetFileName 方法

通过 GetFiles 方法获取的文件名中包含路径，这就需要使用 Path 对象的 GetFileName 方法将路径中的文件名取出。语法：

```
public static string GetFileName (
    string path
)
```

参数 path：用于获取文件名和扩展名。

返回值：一个 String，由 path 中最后的目录字符后的字符组成。如果 path 的最后一个字符是目录或卷分隔符，则此方法返回 Empty。

3. 文件下载关键技术

文件的下载主要是通过 Response 对象的 AddHeader 方法来设置 HTTP 标头的名称和值来实现。最后将当前所有缓冲的输出发送到客户端，代码如下所示。

```
    Response.AddHeader("Content-Disposition", "attachment; filename=" + Server.
UrlEncode(fi.Name));
    Response.AddHeader("Content-Length", fi.Length.ToString());
    //添加输出流内容
    Response.ContentType = "application/octet-stream";
    Response.Filter.Close();
    Response.WriteFile(fi.FullName);
    //将当前所有缓冲的输出发送到客户端
    Response.End();
```

19.3.2　功能实现

文件下载页面的运行结果如图 19.4 所示。

图 19.4　文件下载页面

主要实现步骤说明如下。

（1）创建 Web 窗体，命名为 downFile.aspx，作为文件下载页面。

（2）在 downFile 窗体中添加控件。添加的主要控件及其用途如表 19.2 所示。

表 19.2　downFile.aspx 页面中控件属性设置及其用途

控 件 类 型	控 件 名 称	用　途
标准/ ImageButton 控件	ImageButtonUp	跳转到文件上传页面
	ImageButtonDown	将文件下载到本机
标准/Label 控件	lblMessage	显示提示信息
标准/ListBox 控件	listfile	显示文件名

（3）将文件下载到本机，首先应考虑如何将文件夹中的文件名添加到 ListBox 控件中显示。这里通过使用 bindListBox 自定义方法来实现，在 bindListBox 方法中通过使用

Directory 中的 GetFiles 方法将文件夹中的文件名保存到字符串数组中，通过对字符串数组的遍历将文件名添加到 ListBox 控件中。bindListBox 方法的代码如下所示。

```
protected void bindListBox()
{
    //将指定文件夹中的文件保存到字符串数组中
    string[] name = Directory.GetFiles(Server.MapPath("File"));
    foreach (string s in name)
    {
        //将文件名添加到 ListBox 中
        ListBoxFile.Items.Add(Path.GetFileName(s));
    }
}
```

此方法中使用 Dirctory 类应引用命名空间 System.IO。

在页面首次加载时就应调用 bindListBox 方法，将文件名显示在 ListBox 控件中。在页面的 Page_Load 事件中添加的代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)      //首次加载
    {
        bindListBox();         //调用用户自定义的 bindListBox 方法
    }
}
```

当用户在 ListBox 控件中选择了文件名时应把文件名保存到一个 Session 中去，这个操作通过 listfile 控件的 SelectedIndexChanged 事件来实现。在 SelectedIndexChanged 事件中的代码如下所示。

```
protected void listfile_SelectedIndexChanged(object sender, EventArgs e)
{
    Session["file"] = ListBoxFile.SelectedValue.ToString();
}
```

将文件保存到本机利用自定义方法 dFile 来实现，在 dFile 方法中首先应考虑用户是否选择了文件名，使用 ListBoxFile 控件的 SelectedValue 方法来判断用户是否选择了文件名，只有选择了文件名才可以下载，dFile 方法的代码如下所示。

```
protected void dFile()
{
    //判断是否选择文件名
    if (ListBoxFile.SelectedValue != "")
    {
        if (Session["file"] != "")
        {
            //获取文件路径
            string path = Server.MapPath("File/") + Session["file"].ToString();
            //初始化 FileInfo 类的实例，它作为文件路径的包装
            FileInfo fi = new FileInfo(path);
            //判断文件是否存在
            if (fi.Exists)
```

```
            {
                //将文件保存到本机上
                Response.Clear();
                Response.AddHeader("Content-Disposition", "attachment; filename=
                " + Server. UrlEncode(fi.Name));
                Response.AddHeader("Content-Length", fi.Length.ToString());
                Response.ContentType = "application/octet-stream";
                Response.Filter.Close();
                Response.WriteFile(fi.FullName);
                Response.End();
            }
        }
    }
    else
    {
        Page.RegisterStartupScript("", "<script>alert('请先选择文件名')
        </script>");
    }
}
```

此方法中使用命名空间 System.IO 中的 FileInfo 类。

通过“下载”按钮的 Click 事件调用 dFile 方法来实现文件下载功能。Click 事件的代码如下所示。

```
protected void ImgBtnDown_Click(object sender, ImageClickEventArgs e)
{
    dFile();
}
```

19.4　本 章 小 结

本章介绍了文件上传和文件下载功能。文件上传主要通过文件上传控件来实现，文件下载主要通过设置 HTTP 标头来实现。在今后的网站开发过程中，通过文件上传和下载功能，提供一些有价值的信息给用户下载，是提高网站访问量的最好方法。

第 20 章

AJAX 无刷新聊天室

（名师课堂：46 分钟）

创建一个网上聊天室，有助于提高网站的访问量，聊天室是一个聚集社区成员、召开网络会议的理想场所。随着计算机网络的不断进步，聊天室对大家来说已经不再陌生，应用 AJAX 实现的无刷新聊天室更是流行。如果能在自己制作的无刷新聊天室里聊天，那将是另一番体会。通过本章的学习，读者可以达到以下目的：

- 了解聊天室的业务流程
- 学习框架技术在聊天室中的典型应用
- 学习应用 AJAX 实现页面的无刷新效果
- 掌握 Session 和 Application 变量实现保存数据信息的方法
- 学习应用 AJAX 中的 Timer 控件实现刷新聊天信息
- 掌握在制作聊天室时公共类的编写
- 掌握在普通用户页面中应用 Session 对象获取用户名及其 IP 地址
- 掌握在管理员页面设计中应用 Application 对象删除数据表中信息

20.1　聊天室概述

聊天室的基本功能实质上就是把各用户的聊天记录及消息保存起来，然后在屏幕上按时间或其他某种顺序显示出来。一般聊天室的开发有两种方法：一种是采取将聊天信息存储于数据库的方法；另一种就是采取静态数据的方式存储。本聊天室系统采取将聊天信息存储于数据库的方法，并应用 AJAX 实现页面的无刷新效果。

在线聊天室具有以下功能：

☑　非注册用户也可登录本聊天室进行聊天；

☑　聊天界面应用 AJAX 无刷新环境；

☑　查看在线人员列表；

☑　安全退出聊天室；

☑　管理员登录；

☑　管理员拥有查看所有聊天记录、查看用户 IP 地址、将用户踢出聊天室的功能。

程序运行结果如图 20.1 所示（此图为管理员用户登录页面）。

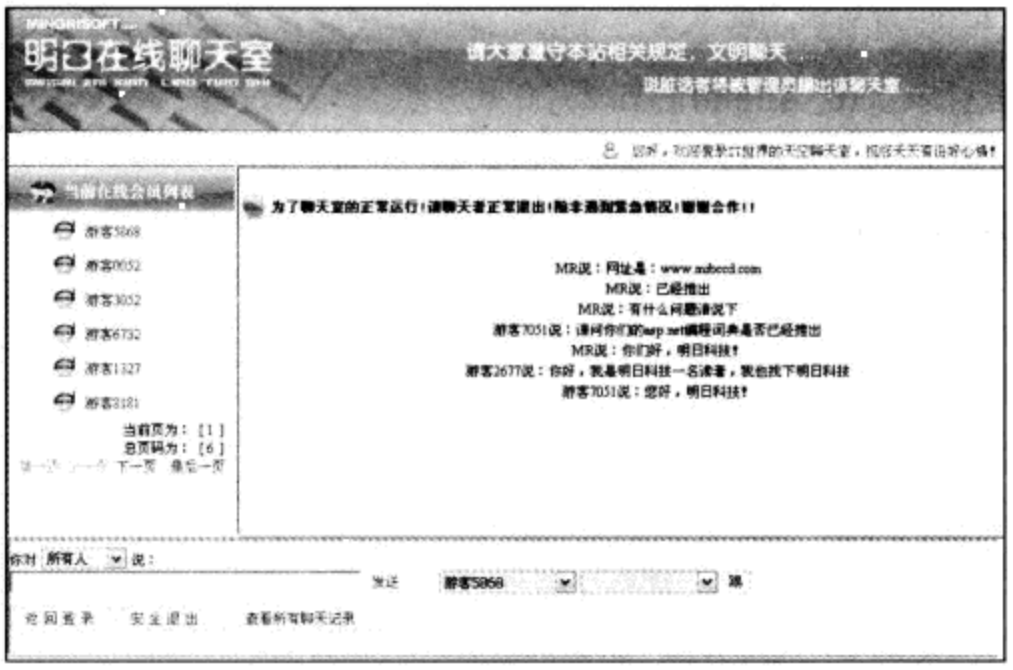


图 20.1　聊天室运行结果

20.2　开发流程图

编写一个聊天室程序前，首先需要考虑该聊天室的主要功能是什么，即系统开发时经常提到的需求是什么。通过相关调查发现，用户使用聊天室时，需要有两种基本角色：游客（即匿名登录的用户）和管理员。

匿名登录的用户在进入聊天室之后，可以实现公聊的功能；而管理员登录之后，有查看匿名用户、查看聊天记录和将恶意搞坏的用户踢出聊天室的权力。在线聊天室系统业务流程如图 20.2 所示。

20.3 数据库设计

数据库设计在此聊天室的设计中也是非常重要的一个环节。如果设计不好，在后来的系统维护、变更和功能扩充时，甚至在系统开发过程中，将会出现问题，大量工作将会重新进行。

从初学者角度出发，为了对本系统后台数据库中数据表有一个更清晰的认识，这里特别设计了一个数据表树型结构图，该数据表树型结构图包含系统所有数据表。如图 20.3 所示。

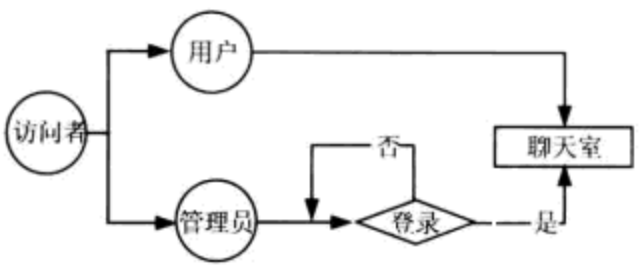


图 20.2 业务流程图

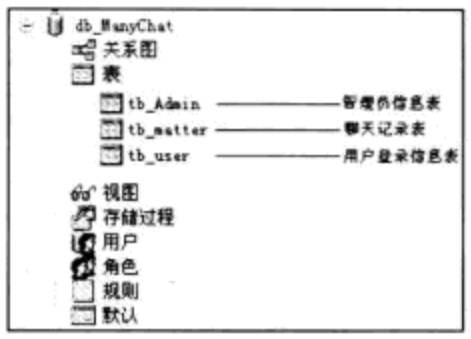


图 20.3 数据表树型结构图

☑ 管理员信息表（tb_Admin）如表 20.1 所示。

表 20.1 管理员信息表（tb_Admin）

字段名称	类 型	是否主键	描 述
id	int	是	管理员编号
name	varchar	否	管理员名称
pwd	varchar	否	管理员密码
infotime	datetime	否	添加时间

☑ 聊天记录表（tb_matter）如表 20.2 所示。

表 20.2 聊天记录表（tb_matter）

字段名称	类 型	是否主键	描 述
id	int	是	聊天信息编号
name	varchar	否	聊天者名称
ip	varchar	否	聊天者 IP 地址
matter	varchar	否	聊天内容
infoTime	datetime	否	聊天记录时间
sts	varchar	否	另一个聊天者名称

☑ 用户登录信息表（tb_user）如表 20.3 所示。

表 20.3 用户登录信息表（tb_user）

字段名称	类 型	是否主键	描 述
id	int	是	用户登录编号

续表

字 段 名 称	类　型	是 否 主 键	描　述
IP	varchar	否	用户登录 IP 地址
img	varchar	否	所选头像图片
loginTime	datetime	否	进入聊天室的时间

20.4　关 键 技 术

专题讲座：光盘\MR\Video\20\关键技术.exe

➤➤➤视频速递：使读者能够更直观地了解本项目使用的技术。

在开发前，读者应该了解本项目具体都应用到了什么技术。只有了解所需要的技术，才能在开发时更加得心应手。

20.4.1　FrameSet 框架技术的应用

在网站开发过程中，框架技术经常被应用。框架页中每一个区域称为一个框架窗口，简称为框架。在模板没有出现之前，框架占了主流。应用页面框架布局可快速建立链接，加速网站的浏览速度，提高浏览者的使用效率。

对于一些浏览器不能处理的文件（如.txt 文件、.xsl 文件等）如果应用 Frameset 框架技术都可在对应的程序窗口中打开。FrameSet 框架技术最典型的应用就是制作网上聊天室。

本实例应用的就是 FrameSet 框架技术，下面对此框架技术进行详细说明。

在 FrameSet 框架技术中经常利用框架标记来定义分割窗口，只需要<FRAMESET>或<FRAME>标记即可，页面所有框架标记都需要放在一个 HTML 文件中，这个文件只记录了该框架是如何被分割的，不会显示任何资料，所以不必放入<BODY>标记，也就是说在使用框架的 HTML 文档中不能出现<BODY>标签，否则会导致浏览器忽略所有的框定义而只显示<BODY>和</BODY>之间的内容。浏览这个框架必须读取这个文件。

需要注意的是，<FRAMESET>标记是框架设计标记，是双标记。首尾标记之间的内容就是使用到框架的 HTML 主体部分。<FRAMESET>标记是用来划分窗口的，每一个窗口由一个<FRAME>标记所标识，<FRAME>必须在<FRAMESET>范围中使用。示例代码如下所示。

```
<Frameset border=1 frameSpacing=1 borderColor=#47478d rows=* cols=180,*>
<Frame src="inc/admin_left.aspx" name=left scrolling=no id="left">
<Frame src="inc/admin_center.aspx" name=main scrolling="no">
</Frameset >
```

在上面的代码中，<FRAMESET></FRAMESET>把页面分为左右两个部分，左侧框架中的页面是 admin_left.aspx，右侧框架中的页面是 admin_center.aspx。

学习手册 注意　<Frame></Frame>标记的框架顺序为从左到右或从上到下。

关于 FrameSet 框架，需要设置一些特定的参数，这些参数直接决定了整个页面的布局，示例代码如下所示。

```
<Frameset border=1 frameSpacing=1 borderColor=#47478d rows=* cols=180,*>
```

关于上段代码的各项参数设置及其含义如表 20.4 所示。

表 20.4　FrameSet 参数

参　数	说　明
Border	设定框架的边框厚度，以 pixels 为单位
FrameBorder	设定是否显示框架的边框，0 为不显示，1 为显示
FrameSpacing	表示框架与框架之间的距离
BorderColor	设定框架的边框颜色
Row	将文档分为上下的框架，Row 后的值可以为数值或百分数，*表示占用余下的空间，数值的个数代表水平分成的框架个数
Cols	设置同 Row

关于表 20.4 中所列的 Row 参数具体举例如下，Rows= “210,*,10%” 表示页面分为上中下 3 个框架页，上边的框架占用 210px，下边的框架占用整个文档的 10%，余下的空间为中间的框架占用。*是一个相对的概念，例如 Row=*，表示页面中没有上下结构的框架布局。Cols 设置同 Row 参数设置。

某些版本的浏览器是不支持框架结构的，如果遇到这种情况，就可以使用<NOFRAMES>和</NOFRAMES>标记再声明一对文件主体标记<BODY>和</BODY>，代表在无法接受框架结构时，唯一显示的页面。应用方法是在<FRAMESET>标记范围内加入</NOFRAMES>标记，示例代码如下所示。

```
<frameset rows="80,*">
  <noframes>
    <body>
       如果您使用的浏览器不支持框架功能，请转用新的浏览器.
    </body>
  </noframes>
    <frame name="top" src="a.html">
  <frame name="bottom" src="b.html">
</frameset>
```

若浏览器支持框架，那它不必理会<NOFRAMES>中的东西，但若浏览器不支持框架，由于不认识所有框架标记，不明的标记会被略过，标记包围的东西便被解读出来，所有放在<NOFRAMES>范围内的文字会被显示。

本实例中用户聊天界面或管理员聊天界面都是通过框架完成的，用户聊天页面框架共绑定了 4 个页面，第 1 个为聊天室用户名单列表页面“left.aspx”；第 2 个为聊天记录页面“right.aspx”；第 3 个为用户发言及退出页面“down.aspx”；第 4 个页面“up.aspx”在程序中起的只是装饰作用，对程序没有任何影响，如图 20.4 所示。

Index.aspx 页为聊天室主界面，在启动网站时，添加的 up.aspx、left.aspx、right.aspx 和 down.aspx 共 4 个文件将会在 Index.aspx 页面的 Frameset 框架中显示出来。关键代码如下所示。

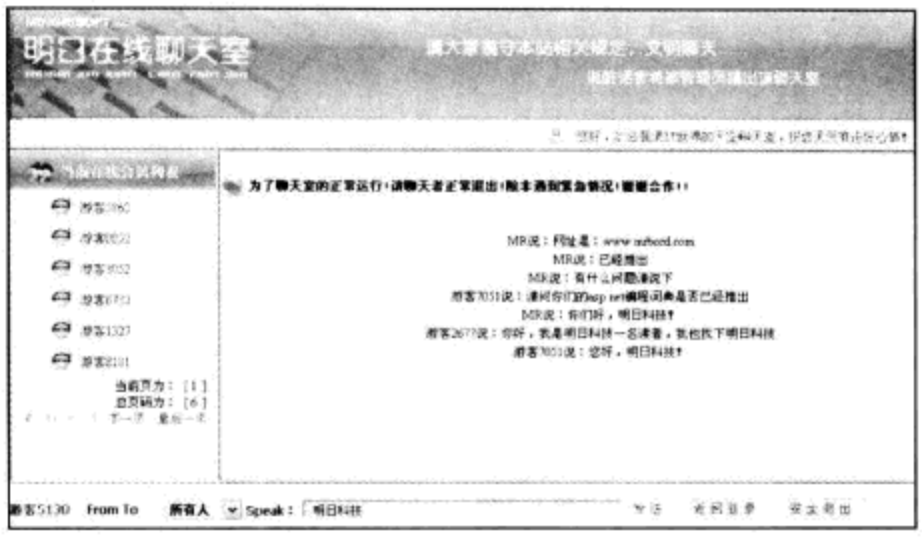


图 20.4　聊天室框架分类

```
    <frameset    rows="100,*,100"    cols="**"    frameborder="yes"    border="100"
framespacing="0">
      <frame src="up.aspx" name="topFrame" scrolling="NO" noresize >
        <frameset rows="*" cols="200,*" framespacing="0" frameborder="yes" border=
        "0">
           <frame src="left.aspx" name="leftFrame" scrolling="NO" noresize>
           <frame src="right.aspx" name="mainFrame">
      </frameset>
      <frame src="down.aspx" name="bottomFrame" scrolling="NO" noresize >
        <noframes>
          <body>
            如果您使用的浏览器不支持框架功能，请转用新的浏览器
          </body>
        </noframes>
      </frameset>
```

学习手册 注意　在使用 Frameset 框架设计页面时，不能同时使用多个<FRAMESET></FRAMESET>标记，但可以在<FRAMESET></FRAMESET>标记间嵌套多个<FRAMESET></FRAMESET>。在向 HTML 文档中插入 FrameSet 框架时，由于 Visual Studio 2008 中没有此框架的可视化操作，所以只能手工添加代码完成。

管理员聊天页面框架绑定与用户聊天页面框架绑定大致相似，只是绑定的页面及页面的功能有所不同，但对聊天记录信息都实现的是无刷新效果。

20.4.2　AJAX 技术应用讲解

只能在服务器上运行程序就像跛脚巨人，虽然很强，但总缺少那么一点儿灵活的感觉，唯有添加 AJAX 弥补服务器与浏览器间的断层，才能最大限度地满足 Web 应用程序的所有需求。

AJAX 是 Asynchronous JavaScript and XML（异步 JavaScript 和 XML 技术）的缩写，它是由 JavaScript 脚本语言、CSS 样式表、XMLHttpRequest 数据交换对象和 DOM 文档对象（或 XMLDOM 文档对象）等多种技术组成的。在应用 AJAX 刷新技术前有必要了解一下其开发模式及架构分析。

1. ASP.NET AJAX 开发模式及架构分析

在传统的 Web 应用模式中，页面中用户的每一次操作都将触发一次返回 Web 服务器的 HTTP 请求，服务器进行相应的处理（获得数据、运行与不同的系统会话）后，返回一个 HTML 页面给客户端，如图 20.5 所示。

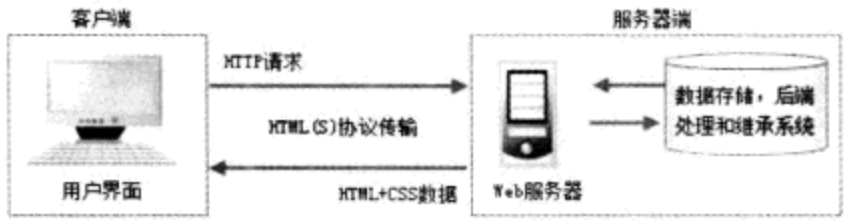


图 20.5　Web 应用的传统模型

而在 AJAX 应用中，页面中用户的操作将通过 AJAX 引擎与服务器端进行通信，然后将返回结果提交给客户端页面的 AJAX 引擎，再由 AJAX 引擎来决定将这些数据显示到页面的指定位置，如图 20.6 所示。

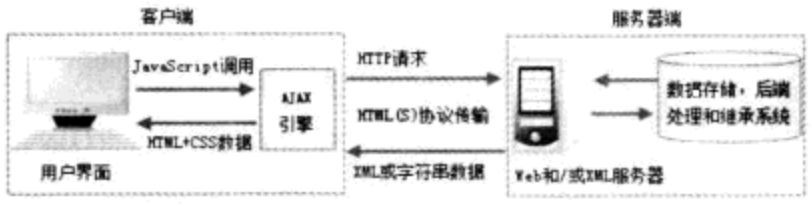


图 20.6　Web 应用的 AJAX 模型

学习手册 说明 从图 20.5 和图 20.6 中可以看出，对于每个用户的行为，传统的 Web 应用模型中将生成一次 HTTP 请求，而在 AJAX 应用开发模型中，将变成对 AJAX 引擎的一次 JavaScript 调用。在 AJAX 应用开发模型中通过 JavaScript 实现在不刷新整个页面的情况下，对部分数据进行更新，从而降低了网络流量，给用户带来了更好的体验。

2. 脚本管理员——ScriptManager 控件

ScriptManager 控件是服务器端 ASP.NET AJAX 模型的大脑，它是一个在页面上没有任何可视界面的 Web 控件。但是，它执行一个主要的任务——它呈现到 ASP.NET AJAX JavaScript 库的链接。

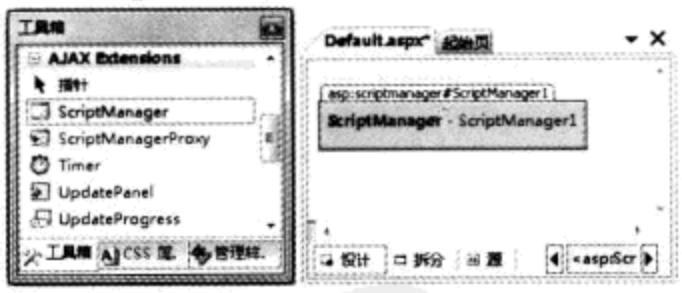


图 20.7　每个 ASP.NET AJAX 网页必须有且仅有一个 ScriptManager 控件

用户可以将 ScriptManager 控件视为 AJAX 脚本管理员，有了该管理员才能够让 Page 局部更新起作用，浏览器所需要的 JavaScript 才会自动管理。因此开发 AJAX 网站时，每个页面中必须添加一个 ScriptManager 控件，以便可以局部更新网页中的数据，并与服务器的程序沟通。Script Manger 控件如图 20.7 所示。

ScriptManger 控件的常用属性如表 20.5 所示。

表 20.5　ScriptManger 控件的常用属性

属　性	说　明
EnablePageMethods	返回或设置一个 bool 值，默认值为 false，表示在客户端 JavaScript 代码中是否以一种简单、直观的形式直接调用服务器端的某个静态 Web Method

续表

属　性	说　明
EnablePartialRendering	返回或设置一个 bool 值，默认值为 true，表示 AJAX 允许改变原有的 ASP.NET 回送模式，不再是整个页面的回送，而是只回送页面中的一部分
EnableScriptComponents	用于设置是否传送除了 AJAX 核心以外的其他组件，包括客户端控件、数据绑定、XML 声明式 Script、用户接口组件
Scripts	用于取得 ScriptReference 对象的集合，ScriptReference 对象的集合通过 AJAX 将用户的 Script 文件送到客户端进行对象引用
Services	用于取得一个 ServiceRefence 对象的集合，ServiceRefence 对象的集合通过 AJAX 为每个 Web Service 在客户端公开一个 Proxy 对象引用。

2. 局部更新面板——UpdatePanel 控件

UpdatePanel 控件和 ScriptManager 控件一起工作。在能够使用 UpdatePanel 前，需要添加一个 ScriptManager 脚本管理控件，然后便可以向页面中添加一个或多个 UpdatePanel 控件。

UpdatePanel 控件在其生命周期里只有一个角色——作为异步刷新内容的容器。所以，不用将所有的东西全都放进 UpdatePanel 中，只需将服务器端更新内容的控件放入 UpdatePanel 即可。

UpdatePanel 控件的常用属性如表 20.6 所示。

表 20.6　UpdatePanel 控件的常用属性

属　性	说　明
ContentTemplate	内容模板，在该模板内放置控件、HTML 代码等
UpdateMode	UpdateMode 属性共有两种模式：Always 与 Conditional，Always 是每次 Postback 后，UpdatePanel 会连带被更新；相反，Conditional 只针对特定情况才被更新
RenderMode	若 RenderMode 的属性值为 Block，则以<DIV>标签来定义程序段；若为 Inline，则以<Span>标签来定义程序段
Triggers	用于设置 UpdatePanel 的触发事件

3. 定时器操作控件——Timer 控件

在应用 UpdatePanel 控件实现页面局部更新时，用户必须初始化一个一般情况下会回发的动作，如单击按钮。但在实际应用中用户可能会希望在没有动作的情况下自动完成一次完整或局部页面的刷新，此时便可利用 AJAX 的 Timer 控件，每隔一段时间固定触发一个事件，就像闹钟叫你起床一样。

Timer 控件的使用非常简单，其中比较重要的属性就是 Interval 及 Enabled。

☑ Interval 属性

Interval 属性用来设置页面更新间隔的最大毫秒数，其默认值为 60000 毫秒(即 60 秒)。每当到达 Timer 控件的 Interval 属性所设置的间隔时间而进行回发时，就会引发服务器端 Tick 事件，可以在该事件中根据实际需要定时执行特定的更新操作。

必须提醒大家，Timer 控件可能会加大 Web 应用程序的负载。因此，在引入自动回发特性前且在确实需要的时候引入 Timer 控件，并尽可能把间隔时间设置得长一

点，如果设置得太短将会使得页面回发频率增加，加大服务器的流量。

☑ Enabled 属性

如果要停止一个定时器，可在服务器端代码里将 Timer 控件的 Enabled 属性设置为 false 即可，这就像把闹钟关掉了，即使时间到了，它也不会响。

20.4.3 Session 对象的应用

Session 对象用于存储每个用户的专用信息。Session 对象与 Application 对象不同，它只针对单一网页使用者，也就是说各个连接的机器都有各自的 Session 对象，不同的客户端无法互相存取。Session 对象中止于联机机器离线时，也就是当网页使用者关掉浏览器或超过设定 Session 变量的有效时间时，Session 对象就会消失。

本实例中用户进入聊天室的信息就是通过 Session 对象保存用户的登录名称和 IP 地址实现的，代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
   //应用 Session 对象保存登录的用户名
   Session["LoginName"] = txtusername.Text;
   //判断页面是否是首次加载
   if(Page.IsPostBack= =false)
     {
         Label1.Text = "匿名";
         Label2.Visible=false;
         txtuserpwd.Visible = false;
         //创建伪随机生成器,产生随机性统计要求的数字序列
         Random Rd = new Random();
         string Rds = "游客" + Rd.Next(9).ToString() + Rd.Next(9).ToString() +
         Rd.Next(9).ToString()
              + Rd.Next(9).ToString();
         txtusername.Text = Rds;
         //应用 Session 对象保存用户的 IP 地址
         Session["IP"] = Request.UserHostAddress.ToString();
     }
}
```

20.4.4 DataList 控件的分页技术

在开发一些网站时，常常需要在页面中比较全面地显示一些信息，但如果要显示的信息记录较多，此时用一个页显示所有记录可能会给用户带来不便。为了解决这个问题，开发人员可以使用分页技术来限定一个页面中显示的记录数。在开发 AJAX 无刷新聊天室时，通过在 DataList 控件中绑定数据，实现分页显示用户列表信息。

DataList 控件是一种数据绑定控件，其与绑定有关的属性及说明如下。

（1）DataKeyField 属性：获取或设置由 DataSource 属性指定的数据源中的键字段。

（2）DataKey 属性：获取存储数据列表控件中每个记录的键值。

（3）DataMember 属性：获取或设置多成员数据源中绑定到数据列表控件的特定数据成员。

（4）DataSource 属性：获取或设置数据源，该数据源中包含用于填充控件中的项的值

列表。

DataList 控件的分页实现是借助 PagedDataSource 类实现的，该类封装了数据控件的分页属性，其常用属性及说明如表 20.7 所示。

表 20.7　PageDataSource 类中的常用属性

属　性	说　明
AllowPaging	获取是否启用分页
AllowCustomPaging	获取或设置是否启用自定义分页
CurrentPageIndex	获取或设置当前显示页的索引
DataSource	获取或设置用于填充控件中项的数据源
PageSize	获取或设置要在 GridView 控件的每页显示的项数
PageCount	获取显示 GridView 控件中各项所需的总页数
FirstIndexPage	获取页中的第一个索引
IsFirstPage	获取一个值，该值指示当前显示的页是否为首页
IsLastPage	获取一个值，该值指示当前显示的页是否为最后一页

在显示用户列表信息的 Left.aspx 页后台代码 Page_Load 事件中，首先判断页面是否首次加载，然后调用自定义 BindData()方法绑定数据库中的数据，实现的代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        BindData();//调用自定义方法绑定用户列表信息
    }
}
```

在上述 Page_Load 事件中调用了自定义 BindData ()方法，该方法为自定义的无返回值类型方法，主要用来从数据库中查询出符合指定条件的记录，并绑定到 DataList 控件中，然后通过设置 PagedDataSource 类对象的 AllowPaging 属性为 True，来实现 DataList 控件的分页功能。BindData ()方法的关键代码如下所示：

```
public void BindData()
{
    int curpage = Convert.ToInt32(this.labPage.Text);
    PagedDataSource ps = new PagedDataSource();
    //建立与数据库的连接
    SqlConnection con = new SqlConnection("server=(local);user id=sa; pwd=;
    database=db_ManyChat");
    //打开数据库连接
    con.Open();
    //创建 SqlDataAdapter 对象的实例
    SqlDataAdapter ada = new SqlDataAdapter("select * from tb_user", con);
    //创建 DataSet 数据集
    DataSet ds = new DataSet();
    //应用 SqlDataAdapter 对象的 Fill 方法填充数据集
    ada.Fill(ds,"tb_user");
    ps.DataSource = ds.Tables["tb_user"].DefaultView;
    ps.AllowPaging = true;
    ps.PageSize = 6;
```

```
        ps.CurrentPageIndex = curpage - 1;
        this.lnkbtnUp.Enabled = true;
        this.lnkbtnNext.Enabled = true;
        this.lnkbtnBack.Enabled = true;
        this.lnkbtnOne.Enabled = true;
        if (curpage == 1)
        {
            this.lnkbtnOne.Enabled = false;          //不显示第一页按钮
            this.lnkbtnUp.Enabled = false;           //不显示上一页按钮
        }
        if (curpage == ps.PageCount)
        {
            this.lnkbtnNext.Enabled = false;         //不显示下一页
            this.lnkbtnBack.Enabled = false;         //不显示最后一页
        }
        this.labBackPage.Text = Convert.ToString(ps.PageCount);
        dlName.DataSource = ps;
        //指定数据源中的键字段
        dlName.DataKeyField = "name";
        //绑定数据源
        dlName.DataBind();
    }
```

当用户单击用于操作分页的 LinkButton 控件时，程序根据当前页码执行指定操作。用于控制分页的 LinkButton 控件的 Click 事件代码如下所示：

```
protected void lnkbtnOne_Click(object sender, EventArgs e)
{
    this.labPage.Text = "1";
    this.BindData();
}
protected void lnkbtnUp_Click(object sender, EventArgs e)
{
    this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) - 1);
    this.BindData();
}
protected void lnkbtnNext_Click(object sender, EventArgs e)
{
    this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) + 1);
    this.BindData();
}
protected void lnkbtnBack_Click(object sender, EventArgs e)
{
    this.labPage.Text = this.labBackPage.Text;
    this.BindData();
}
```

20.5 实 现 过 程

专题讲座：光盘\MR\Video\20\实现过程.exe

>>>视频速递：使读者能够更直观地了解本项目的实现过程。

从本节开始，我们将介绍 AJAX 无刷新聊天室的整个开发过程。通过图文并茂的形式介绍开发过程使读者更容易理解和掌握，希望读者对本节内容能够认真阅读。

20.5.1　公共类编写

在网站开发项目中以类的形式来组织、封装一些常用的方法和事件，将会在编程过程中起到事半功倍的效果。本节介绍系统使用的公共类，如数据库连接类等。编写类可以减少重复代码的编写，有利于代码维护。创建公共类的方法为：在 Microsoft Visual Studio 2008 菜单中，选择“网站”\“添加新项”菜单项，在弹出的“添加新项”对话框中选择“类”，默认名称为 Class.cs，将其命名为 MCClass.cs，如图 20.8 所示。

图 20.8　创建类

在“解决方案资源管理器”对话框中，可以看到添加的新类。双击新添加的类进入后台，进行类的编写。进入后可以看到系统自动添加的命名空间、公共类和构造函数。在命名空间区域引用命名空间“using System.Data.SqlClient”，代码如下所示。

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
public class MCClass
{
    public MCClass()
    {
        //
        // TODO: 在此处添加构造函数逻辑
        //
    }
.....
.....编写其他功能方法代码
}
```

MCClass.cs 类中一共定义了 3 个方法，下面分别进行讲解。

1. SqlCon 方法（数据库连接方法）

SqlCon 方法用来连接数据库，类型为 SqlConnection，返回类型名 mycon，主要用来构造数据库连接，实现代码如下所示。

```
//创建一个数据库连接的方法
public SqlConnection SqlCon()
{
   SqlConnection mycon = new SqlConnection("Data Source=(local);User ID=sa;PWD=;
   DataBase
       =db_ManyChat");
   return mycon;
}
```

2. ExecCommand 方法（执行添加/修改/删除操作）

ExecCommand 方法用来执行 SQL 语句，首先建立 SQL Server 数据库连接，然后通过 SqlCommand 对象执行参数传入的 SQL 语句，最后应用 try…catch…finally 语句进行异常处理，如果该 SQL 语句执行成功返回 True，否则返回 False，代码如下所示。

```
//创建一个执行 SQL 语句命令的方法
public bool ExecCommand(string sqlstr)
{
 //创建数据库连接
 SqlConnection con = new SqlConnection
 ("Data Source=(local);User ID=sa;PWD=;DataBase=db_ManyChat");
 //打开数据库连接
 con.Open();
 //创建 SqlCommand 对象的实例
 SqlCommand com = new SqlCommand(sqlstr, con);
 try
    {
        //执行 SQL 语句命令
        com.ExecuteNonQuery();
        return true;
    }
    catch
    {
        return false;
    }
    finally
    {
        //关闭数据库连接
        con.Close();
    }
}
```

3. GetDataSet 方法（获取数据集）

创建一个返回类型为 DataSet 数据集的方法，在此方法中有两个参数，分别为所要执行的 SQL 语句和填充数据集所指定数据库中的表名，最后返回数据集，代码如下所示。

```
//创建一个返回类型为 DataSet 数据集的方法
public DataSet ReturnDataSet(string sqlstr, string tbName)
  {
    //建立数据库连接
```

```
    SqlConnection con = new SqlConnection
    ("server=(local);user id=sa;pwd=;DataBase=db_ManyChat");
    //打开数据库连接
    con.Open();
    //创建 SqlDataAdapter 对象的实例
    SqlDataAdapter ada = new SqlDataAdapter(sqlstr, con);
    //创建 DataSet 数据集
    DataSet ds = new DataSet();
    //应用 SqlDataAdapter 对象的 Fill 方法填充数据集
    ada.Fill(ds, tbName);
return ds;
}
```

20.5.2　设计分析

1. 用户登录模块

在用户登录模块的设计中，应用了两个数据表，分别为 tb_user 和 tb_Admin；主要技术的应用是 ASP.NET 内置对象中的 Application 对象的应用。

用户登录模块主要用于普通用户和管理员进入本程序。普通用户登录时，可使用自己的名字进入聊天室，也可使用系统提供给用户的自定义名字登录。但管理员登录聊天室有所不同，管理员必须通过输入正确的密码才能进入聊天室进行管理。

匿名登录页面的运行效果如图 20.9 所示。

当选择“管理员登录”复选框时，可进入管理员登录页面，其运行效果如图 20.10 所示。

图 20.9　匿名用户登录页面

图 20.10　管理员登录界面

普通用户登录时，无须对页面进行任何操作，可直接单击“登录”按钮进入聊天室，而该用户进入聊天室的同时，系统会通过变量 Application["Name"]和 Application["IP"]将该用户的信息（包括用户名、用户 IP 地址、进入的时间等）存储在数据表 tb_user 中。当用户离开聊天室时，系统会自动删除该用户存储在数据表 tb_user 中的记录信息。

实现普通用户登录的步骤说明如下。

（1）首先引用命名空间“using System.Data.SqlClient”，并创建公共类对象 MCClass。

（2）进入“Login.aspx”页的 Page_Load 加载事件过程，在此事件过程中指定用户登录的随机名称、变量 Session ["Name"]存储的登录名称和变量 Session ["IP"]存储的 IP 地址，代码如下所示。

```
public partial class Login : System.Web.UI.Page
{
    MCClass mc = new MCClass();//调用自定义的一个类
    protected void Page_Load(object sender, EventArgs e)
    {
        //应用 Session 对象保存登录的用户名
        Session["LoginName"] = txtusername.Text;
        //判断页面是否是首次加载
        if(Page.IsPostBack==false)
        {
```

```
            Label1.Text = "匿名";
            Label2.Visible=false;
            txtuserpwd.Visible = false;
            Random Rd = new Random();//伪随机生成器,产生随机性统计要求的数字序列
            string Rds = "游客" + Rd.Next(9).ToString() + Rd.Next(9).ToString()
             + Rd.Next(9).ToString() + Rd.Next(9).ToString();
            txtusername.Text = Rds;
            //应用 Session 对象保存用户的 IP 地址
        Session["IP"] = Request.UserHostAddress.ToString();
        }
    }
```

（3）判断管理员登录的 CheckBoxList 控件“cblIfAdmin”是否被选中，选中则为管理员登录，反之则为普通用户登录。代码如下所示。

```
protected void CheckBoxList1_TextChanged(object sender, EventArgs e)
{
   //判断管理员登录的 CheckBoxList 控件"cblIfAdmin"的是否被选择中
 if (cblIfAdmin.Items[0].Selected == true)    //值为 True 为管理员登录
    {
        Label2.Visible = true;
        Label1.Text = "管理员名称";
        Label2.Text = "管理员密码";
        txtuserpwd.Visible = true;
        txtusername.Text = "mr";
        txtusername.Enabled = false;
        Label3.Visible = false;
    }
  else//以下代码为普通用户登录
     {
        Label1.Text = "匿名";
        //将 Session 对象保存的值赋予 txtusername 的 Text 属性
        txtusername.Text = Session["LoginName"].ToString();
        Label2.Visible = false;
        txtusername.Enabled = true;
        txtusername.Text = null;
        txtuserpwd.Visible = false;
        Label3.Visible = true;
     }
}
```

（4）对进入聊天室时的性别头像图片进行选择，以便区分用户的性别。代码如下所示。

```
protected void DropDownList1_TextChanged(object sender, EventArgs e)
 {
    if (DropDownList1.SelectedItem.Text= ="boy")
   {
     Image1.ImageUrl = "face/boy.gif";
   }
  else
    {
        Image1.ImageUrl = "face/girl.gif";
    }
}
```

（5）管理员登录与普通用户登录的窗口在同一页面中。管理员登录时，只需将控件

“cblIf Admin”选中即可，而该功能代码同时编写于“登录”按钮的 Click 单击事件中。

（6）进入“登录”按钮的 Click 单击事件中，首先编写 insert 语句，然后调用公共类方法“ExecCommand”执行该语句，完成将用户登录聊天室的信息保存到数据表 tb_user 的功能。代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
  {
    if (cblIfAdmin.Items[0].Selected == true)
        {
           //建立与数据库的连接
           SqlConnection con = new SqlConnection
            ("server=(local);user id=sa; pwd=;database=db_ManyChat");
           //打开数据库连接
           con.Open();
     //创建一个 SqlCommand 对象的实例
     SqlCommand com = new SqlCommand("select count(*) from tb_Admin
     where Name='" + txtusername.Text + "' and pwd='" + txtuserpwd.Text + "'", con);
     //定义一个 int 型的 count 变量，并应用 ExecuteScalar 方法
     //返回查询的结果集中第一行的第一列
     int count = Convert.ToInt32(com.ExecuteScalar());
    //判断所查询的结果
    if (count > 0)
    {
        Session["Name"] = txtusername.Text;
        Session["PWD"] = txtuserpwd.Text;
        //通过 Response 的 Redirect 方法实现页面跳转到 index1.aspx 页
        Page.Response.Redirect("index1.aspx");
}
     else
      {
        //在 Response 的 Write 方法中应用 JavaScript 脚本弹出错误提示对话框
        Response.Write("<script lanuage=javascript>alert('用户名
        或密码有误！');location='javascript:history.go(-1)'</script>");
         return;
        }
 }
     else
      {
      //如果用户选择的头像中性别为"男"，则执行以下语句
      if (DropDownList1.SelectedItem.Text == "boy")
      {
         //调用自定义类 mc 对象中的 ExecCommand 方法执行 SQL 语句命令
         mc.ExecCommand("insert into tb_user(name,ip,img)
         values('" + txtusername.Text + "','" + Request.
         UserHostAddress.ToString() + "','" + Image1.ImageUrl + "')");
          //实现页面跳转
         Page.Response.Redirect("index.aspx");
        }
        else//如果用户选择的头像中性别为"女"，则执行以下语句
        {
            mc.ExecCommand("insert into tb_user(name,ip,img)
            values('" + txtusername.Text + "','" + Request.UserHostAddress.
            ToString()
             + "','" + Image1.ImageUrl + "')");
            Page.Response.Redirect("index.aspx");
```

```
        }
    }
}
```

2. 普通用户聊天室页面

在普通用户聊天室页面设计中，应用的数据表为 tb_user 和 tb_matter；主要技术的应用是 ASP.NET 内置对象中的 Session 变量的应用。

普通用户进入聊天室页面后主要实现基本的聊天功能，不提供任何管理权限。普通用户聊天室页面的运行效果如图 20.11 所示。

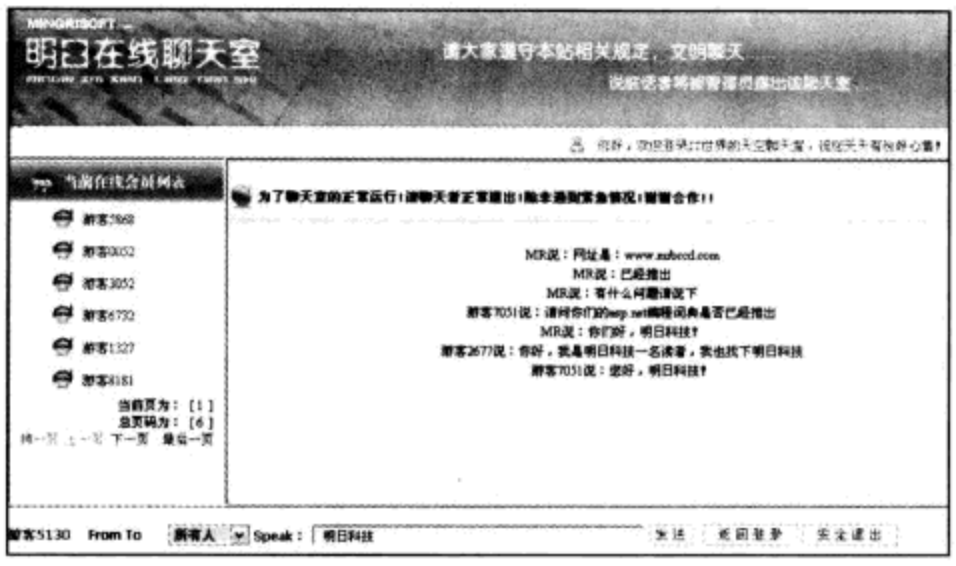


图 20.11　普通用户聊天页面

图 20.11 中的聊天室用户名单列表（即页面 left.aspx）是通过将数据表 tb_user 中的数据绑定至 DataList 控件（ID 属性为 dlName）实现的。

在显示用户名单列表时，主要通过将 Image 和 LinkButton 控件编辑到 DataList 模板中实现。

应用 DataList 模板的具体步骤说明如下。

（1）单击 ID 属性名为 dlName 的 DataList 控件右上角的“ ”图标按钮，在快捷菜单（如图 20.12 所示）中选择“编辑模板”，或者右键单击 DataList 控件，选择快捷菜单中“编辑模板”中的“项模板（ItemTemplate）”选项，即可在 DataList 控件的项模板（ItemTemplate）中进行编辑。

在 DataList 控件的项目模板（ItemTemplate）中分别添加一个 Image 控件和一个 LinkButton 控件，如图 20.13 所示，分别用于显示用户的头像和用户的名称。

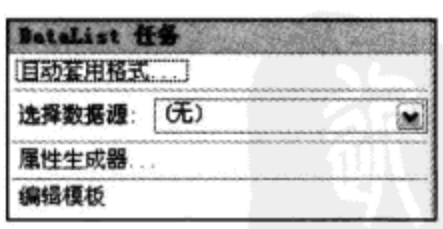


图 20.12　DataList 快捷菜单

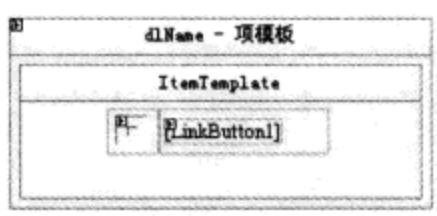


图 20.13　DataList 控件的项目模板内容

（2）在 left.aspx 页面的“源”模式下分别添加如下代码。

☑ 在 Image 控件的 ImageUrl 属性中编写如下数据绑定表达式。

```
<asp:Image ID="Image1" runat="server" Height="19px" ImageUrl=
'<%# DataBinder.Eval(Container.DataItem,"img") %>'Width="20px" />
```

☑　在 LinkButton 控件的 text 属性中编写如下数据绑定表达式。

```
<asp:LinkButton ID="LinkButton1" runat="server" Font-Underline="False" ForeColor=
"#404040"text='<%# DataBinder. Eval(Container.DataItem,"name") %>' CommandName=
"update"></asp:LinkButton>
```

上述代码中应用了 DataBinder.Eval 方法在运行时计算数据表达式，语法格式如下。

```
<%# DataBinder.Eval(Container.DataItem,expression) %>
```

Container.DataItem 表达式引用对该表达式进行计算的对象。DataItem 属性表示当前容器上下文中的对象。在 ASP.NET 3.5 中上述语法格式可被简化成以下格式。

```
<%# Eval(expression) %>
```

Eval 是建立在 DataBinder.Eval 方法之上的一个简单包装，它实现了数据读取的自动化。另外，在 ASP.NET 3.5 中还支持另一种数据绑定方法——Bind 方法。该方法与 Eval 相比，不仅实现了数据的读取自动化，还实现了数据的写入自动化。只要可以使用 Eval 的地方，就可以使用新的 Bind 方法，并且它们的语法类似性。

（3）在 left.aspx 页面的后台代码中，主要实现用户信息列表数据的显示，其主要代码请参见 20.4.4 节，由于篇幅有限这里不再赘述。

要实现聊天记录的页面为 right.aspx。在此页面的窗体中首先拖放一个 ScriptManager 控件（该控件负责管理 Page 页面中所有的 AJAX 服务器控件，是 AJAX 的核心），接着拖放一个 UpdatePanel 控件（该控件使原本不具备 AJAX 能力的 ASP.NET 服务器控件都具有 AJAX 异步的功能），最后在添加的 UpdatePanel 控件中分别拖放一个 DataList 控件（ID 属性为 dlChatNote）和一个 Timer 控件（其 Interval 属性设置为 1000），分别用来显示用户聊天记录信息和每隔一秒对聊天记录信息进行刷新。

在此 DataList 控件的项目模板（ItemTemplate）中添加一个 Label 控件，其 Text 属性绑定数据表达式，代码如下所示。

```
<ItemTemplate>
        <asp:Label ID="Label4" runat="server" Text='<%# DataBinder.Eval
        (Container.DataItem, "matter") %>'></asp:Label>
 </ItemTemplate>
```

在此页面的后台代码中，首先声明所创建的公共类 MCClass，以便调用其中的方法实现相应功能，声明如下：

```
MCClass mc = new MCClass();
```

然后自定义一个 BindData()方法对管理员或普通用户身份进行判断，并将其发言信息存储到数据表 tb_matter 中。代码如下所示。

```
public void BindData()
{
   //定义一个变量 count
   int count;
   //建立数据库连接
   SqlConnection con = new SqlConnection("server=(local); user id=sa; pwd=;
   database=db_ManyChat");
```

```
con.Open();
if (Session["name"] != null)
{
    count = 1;
}
else
{
    //创建一个 SqlCommand 命令来执行查询结果
    SqlCommand com = new SqlCommand("select count(*)
    from tb_user where name= '" + Session["LoginName"] + "'", con);
    count = Convert.ToInt32(com.ExecuteScalar());
}
//判断数据库中是否有要查询的值
 if (count > 0)
{
    dlChatNote.Enabled = true;
}
else
{
    dlChatNote.Visible = false;
    if (Session["Name"] == "OP")
    {
        SqlConnection myconn = new SqlConnection("server=(local);
            user id=sa; pwd=;database=db_ManyChat");
            myconn.Open();
            SqlCommand mycomm = new SqlCommand("select count(*)
            from tb_matter,tb_user where tb_matter.name=tb_user.name and
            tb_matter.name = '" + Session["LoginName"] + "'", myconn);
            int count1 = Convert.ToInt32(mycomm.ExecuteScalar());
            if (count1 > 0)
            {
                dlChatNote.Enabled = true;
            }
            else
            {
                dlChatNote.Visible = false;
            }
        }
    }
    if (Session["infoTime"] == null)
    {
        //应用 Session 对象保存当前的时间
        Session["infoTime"] = DateTime.Now;
    }
    DataSet ds = null;
    //调用 ReturnDataSet 方法，返回一个 DataSet 类型的数据集
    ds = mc.ReturnDataSet("select top 16* from tb_matter
    where infoTime>'" + Session["infoTime"] + "'order by id desc ", "tb_
    matter");
    dlChatNote.DataSource = ds;
    dlChatNote.DataBind();
  }
}
```

最后在添加的 Timer 控件中定义一个 Tick 事件，在该事件中调用自定义的 BindData() 方法，实现每隔一秒对聊天记录信息进行刷新，实现代码如下所示：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    BindData();
}
```

用户发言及退出页面的设计分为 4 个小模块，分别为发言模块、是否对“所有人”发言模块、返回登录模块和安全退出模块，如图 20.14 所示。

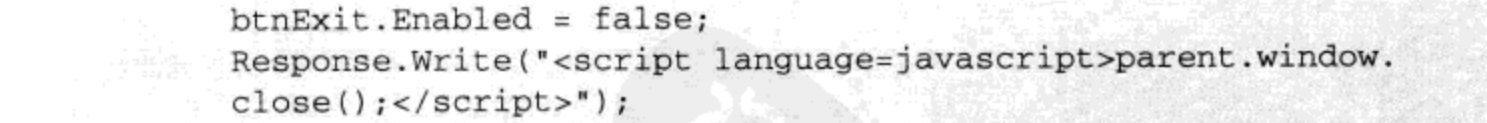


图 20.14　用户发言模块

☑　发言模块

发言模块的主要功能是通过单击“发送”按钮向数据表中插入聊天记录。

单击“发送”按钮，进入程序页面。在该按钮的单击事件过程中，调用公共类方法 ExecCommand，从而实现插入聊天记录的功能。代码如下所示。

```
protected void btnSend_Click(object sender, EventArgs e)
{
    if (ddlIfName.SelectedIndex == 0)
    {
        //建立与数据库的连接
        SqlConnection con = new SqlConnection("server=(local);user id=sa; pwd=;
        database=db_ ManyChat");
        con.Open();
        //创建 SqlCommand 命令执行查询结果
        SqlCommand com = new SqlCommand("select count(*) from
        tb_user where Name='" + Label4.Text + "'", con);
        int count = Convert.ToInt32(com.ExecuteScalar());
        if (count > 0)
        {
            //调用公共类 ExecCommand 方法执行 SQL 插入命令
            mc.ExecCommand("insert into tb_matter (name,ip,matter)
            values('" + Label4.Text + "','" + Session["IP"] + "',
            '" + Label4.Text + "说: " + txtSpeak.Text + "')");
            txtSpeak.Text = null;
        }
        else
        {
            txtSpeak.Enabled = false;
            btnSend.Enabled = false;
            btnExit.Enabled = false;
            Response.Write("<script language=javascript>parent.window.
            close();</script>");
        }
    }
else
    {
        SqlConnection con = new SqlConnection("server=(local); user id=sa; pwd=;
        database=db_ManyChat");
        con.Open();
        SqlCommand com = new SqlCommand("select count(*)
            from tb_user where Name='" + Label4.Text + "'", con);
        int count = Convert.ToInt32(com.ExecuteScalar());
        if (count > 0)
        {
```

```
        //调用公共类 ExecCommand 方法执行 SQL 插入命令
      mc.ExecCommand("insert into tb_matter (name,ip,matter,sts)
            values('" + Label4.Text + "','" + Session["IP"] + "','" + Label4.
            Text + "对" + Request["ddlName"] + "说：" + txtSpeak.Text + "',
            '" + Request["ddlName"] + "')");
       txtSpeak.Text = null;
       }
     else
      {
          txtSpeak.Enabled = false;
          btnSend.Enabled = false;
          btnExit.Enabled = false;
         Response.Write("<script language=javascript>parent.window.
         close();</script>");
        }
     }
}
```

☑ 是否对“所有人”发言模块

该模块是通过对 DropDownList 控件“ddlIfName”的选择来实现，如果用户选择该控件的“悄悄话”选项，则另一个名为“ddlName”的 DropDownList 控件将在控件“ddlIfName”的左侧显示；反之若选择“所有人”则控件“ddlName”将隐藏于页面中，实现上述判断功能的代码如下所示。

```
protected void ddlIfName_SelectedIndexChanged(object sender, EventArgs e)
{
    if (ddlIfName.SelectedItem.Text = = "游客")
    {
        ddlName.Visible = true;
    }
    else
    {
        ddlName.Visible = false;
    }
}
```

在 Page_Load 页面加载事件过程中调用公共类方法“ReturnDataSet”，完成将聊天室用户名单绑定到 DropDownList 控件 ddlName 上。代码如下所示。

```
public partial class down : System.Web.UI.Page
{
    MCClass mc = new MCClass();
    protected void Page_Load(object sender, EventArgs e)
    {
        if(Page.IsPostBack= =false)
        {
            //将 Session 保存的用户名赋予控件 Label4 的 Text 属性
            Label4.Text = Session["LoginName"].ToString();
            SqlConnection con = new
            SqlConnection("server=(local);user id=sa; pwd=;database=db_ManyChat");
            con.Open();
          SqlCommand com = new SqlCommand("select count(*) from tb_user where
          Name='" + Label4.Text + "'", con);
```

```
            int count = Convert.ToInt32(com.ExecuteScalar());
            //判断用户是否已登录
        if (count > 0)
        {
            btnSend.Visible = true;
        }
        else
         {
            txtSpeak.Enabled = false;
            btnSend.Enabled = false;
            btnExit.Enabled = false;
            }
            SqlConnection mycon = new
           SqlConnection("server=(local);user id=sa; pwd=;database=db_ManyChat");
            mycon.Open();
            SqlDataAdapter ada = new SqlDataAdapter("select * from tb_user", mycon);
            DataSet ds = new DataSet();
            ada.Fill(ds, "tb_user");
            ddlName.DataSource = ds;
            ddlName.DataTextField = "name";
            ddlName.DataValueField = "name";
            ddlName.DataBind();
            mycon.Close();
          }
    }
}
```

☑　返回登录模块

返回登录模块主要用于返回登录页面 Login.aspx，使用户重新登录。由于本系统是通过框架组合而成的，因此返回登录功能是通过 JavaScript 脚本实现的。实现该功能的代码如下所示。

```
protected void btnReturn_Click(object sender, EventArgs e)
  {
    //调用公共类的 ExecCommand 方法实现 SQL 语句的删除操作
    mc.ExecCommand("delete from tb_user where name='" + Label4.Text + "'");
    Response.Write("<script>parent.location='Login.aspx'</script>");
  }
```

☑　安全退出模块

当用户单击“安全退出”按钮时，程序会弹出是否关闭浏览器的消息提示框，单击提示框中的“确定”按钮，此时程序通过调用公共类方法 ExecCommand，将进入聊天室时存储于数据表 tb_user 中的记录删除，同时关闭该浏览器。实现安全退出的后台代码如下所示。

```
protected void btnExit_Click(object sender, EventArgs e)
{
  //调用公共类的 ExecCommand 方法实现 SQL 语句的删除操作
   mc.ExecCommand("delete from tb_user where name='" + Label4.Text + "'");
   Response.Write("<script language=javascript>parent.window.close();
   </script>");
}
```

3. 管理员聊天室页面

在管理员聊天室页面设计中，应用的数据表为 tb_user 和 tb_matter；主要技术是应用 GridView 数据表格控件自带的删除功能删除数据表中的记录和应用 ASP.NET 内置对象中的 Application 对象将存储于数据表 tb_user 中的信息删除。

管理员聊天室页面与用户聊天室页面的功能大致相同，唯一不同的是，管理员登录聊天室时为隐身状态，普通用户在名称列表中看不到管理员，但管理员的发言用户可以看到。此功能的实现是在登录窗口时，系统通过检索变量 Session["Name"] 和 Session["PWD"] 来验证管理员身份，登录成功则进入管理员聊天室页面，并像普通用户进入一样，将数据信息插入数据表中。管理员聊天室页面的运行结果如图 20.15 所示。

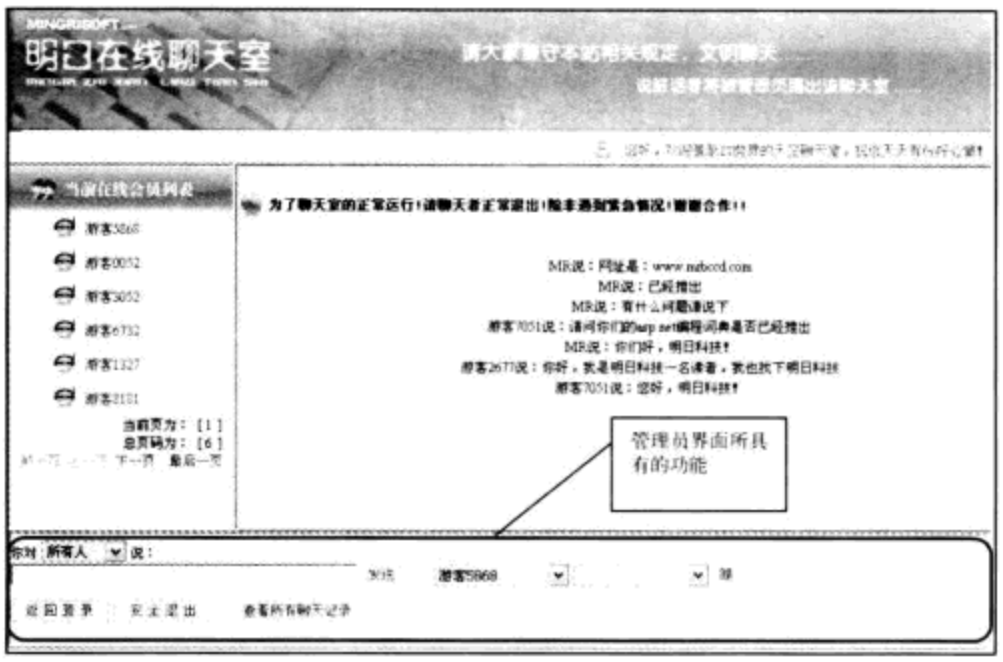


图 20.15　管理员登录后的聊天页面

在聊天室中聊天时，可能会遇到某用户说脏话或刷屏（发送一些无用信息）等不良行为，这时可以通过程序中设置的“查看聊天记录”和“将用户踢出聊天室”功能将这些信息删除，从而保持一个公共文明的聊天室环境。

☑　查看聊天记录

“查看聊天记录”主要用于查看数据表 tb_matter 中存储的所有用户聊天记录信息，结果如图 20.16 所示。

名称	IP	内容	时间	删除
游客7051	192.168.1.51	游客7051说：你好	2008-9-11 15:15:10	删除
su	192.168.1.53	MR说：网址是：www.mbcod.com	2008-9-11 13:09:02	删除
su	192.168.1.53	MR说：已经推出	2008-9-11 13:08:46	删除
su	192.168.1.53	MR说：有什么问题请说下	2008-9-11 13:07:41	删除
游客7051	192.168.1.51	游客7051说：请问你们的asp.net编程词典是否已经推出	2008-9-11 13:07:41	删除
su	192.168.1.53	MR说：你们好，明日科技！	2008-9-11 13:07:23	删除
游客2677	192.168.1.53	游客2677说：你好，我是明日科技一名读者，我也找下明日科技	2008-9-11 13:07:01	删除
游客7051	192.168.1.51	游客7051说：您好，明日科技！	2008-9-11 13:06:32	删除
su	192.168.1.53	MR说：dfd	2008-9-11 11:53:02	删除
游客4601	192.168.1.53	游客4601说：你好	2008-9-11 10:50:44	删除
su	192.168.1.53	OP说：你好朋友，我们的公司的编程词典网是www.mbcod.com	2008-9-11 10:36:28	删除
游客3062	192.168.1.51	游客3062说：编程词典的网址是	2008-9-11 10:34:49	删除
游客7837	192.168.1.53	游客7837说：ASP.NET编程分各种版式本的，标准版很实惠	2008-9-11 10:34:39	删除
游客3062	192.168.1.51	游客3062说：那就麻烦您给介绍下	2008-9-11 10:34:14	删除
游客7837	192.168.1.53	游客7837说：好的呀	2008-9-11 10:33:34	删除

1 2 3 4 5 6 7 8 9 10 ...

图 20.16　查看聊天记录

此模块开发的主要步骤说明如下。

（1）通过使用 JavaScript 脚本弹出新窗口，代码如下所示。

```
protected void btnInfo_Click(object sender, EventArgs e)
{
    Response.Write("<script language=javascript>window.open('InfoChatNote.
    aspx','','width=700,
        heigth=400')</script>");
}
```

（2）在弹出的新窗口中显示聊天记录信息，此功能主要通过将数据表 tb_matter 中的信息绑定到 GridView 控件上实现。这里，管理员可以删除指定的信息，因此在 GridView 控件的 RowDeleting 事件中编写了删除聊天记录的功能。代码如下所示。

```
public partial class InfoChatNote : System.Web.UI.Page
{
    //创建 MCClass 类的一个实例对象
    MCClass mc = new MCClass();
    protected void Page_Load(object sender, EventArgs e)
    {
        //初使化 DataSet 对象
        DataSet ds = null;
        //调用自定义类中的 ReturnDataSet 方法返回一个 DataSet 类型的数据集
        ds = mc.ReturnDataSet("select * from tb_matter order by id desc", "tb_
        matter");
        gvInfoChatNote.DataSource = ds;
        //获得一个包含当前显示项的主键字段的名称为 id 的数组
        gvInfoChatNote.DataKeyNames = new string[] { "id"};
        //绑定数据
        gvInfoChatNote.DataBind();
    }
    protected void GridView1_RowDeleting(object sender, GridViewDeleteEvent-
    Args e)
    {
        //建立数据库的连接
        SqlConnection con = new SqlConnection("server=(local);
        user id=sa; pwd=;database=db_ManyChat");
        //打开数据库的连接
        con.Open();
        //创建一个 SqlCommand 对象的实例
        SqlCommand com = new SqlCommand("delete from tb_matter
        where id='" + gvInfoChatNote.DataKeys[e.RowIndex].Value + "'", con);
        //执行 SqlCommand 命令
        com.ExecuteNonQuery();
        //执行页面刷新
        Page_Load(sender, e);
    }
}
```

☑　将用户踢出聊天室

将用户踢出聊天室的功能很简单，只需将用户进入聊天室时通过变量 Application["Name"]存储于数据表 tb_user 中的信息删除即可。代码如下所示。

```
protected void btnDel_Click(object sender, EventArgs e)
{
    if (ddlDelName.Text != "")
    {
```

```
            btnDel.Enabled = true;
            //建立数据库连接
            SqlConnection con = new SqlConnection("server=(local);
            user id=sa; pwd=;database=db_ManyChat");
            //打开数据库连接
            con.Open();
            //创建 SqlCommand 对象的实例
            SqlCommand com = new SqlCommand("delete from tb_user
            where  name='" + ddlDelName.Text + "'", con);
            SqlCommand com1 = new SqlCommand("insert into tb_matter(name,matter)
            values('" + ddlDelName.Text + "','" + ddlDelName.Text + "被管理员踢出聊天室!
            " + "')",
            con);
            //执行 SqlCommand 命令
            com1.ExecuteNonQuery();
            com.ExecuteNonQuery();
            //页面跳转
            Response.Redirect("down1.aspx");
        }
        else
        {
            btnDel.Enabled = false;
        }
    }
```

20.6 本章小结

本章主要介绍了网上在线聊天系统的简单制作过程。在制作此聊天室过程中应用了框架技术和 AJAX 实现界面的无刷新，同时实现了 Session 和 Application 变量保存数据信息的方法，如获取登录用户的用户名及其 IP 地址等；为了以最少代码来编写应用程序，本实例编写了公共类，在此公共类中包含了一些标准的被许多页面调用的使用程序方法和一些用户识别验证的基本代码。

第21章

实现会员密码找回功能

（名师课堂：45分钟）

随着网站对会员开放功能的增加，越来越多的网友都会通过网站注册成为会员。但是这样就有可能将某个网站的会员账号或密码忘记。为了解决这一问题这里将介绍密码找回模块。现在比较流行的密码找回方式有两种：一种方式是通过使用回答密码提示问题来找回会员忘记的密码；另一种方式是在会员注册时将会员的登录名和密码信息通过电子邮件发送到会员的邮箱中。本模块结合了这两种比较流行的密码找回方式，使读者掌握这两种密码找回方式的使用和设计思路。通过本章的学习，读者能够学到以下内容：

- 密码找回功能
- 密码找回三次提示
- 用户登录功能
- 会员注册功能

21.1 概　　述

21.1.1 功能概述

会员密码找回功能，是提供给已注册会员的一个功能。该功能是为了解决会员忘记密码而带来的不必要的损失。本程序中的密码找回使用了两种方式，第一种是在用户填写注册信息时填写密码提示问题和密码提示答案。当会员忘记密码时可以进入密码找回页，在该页中将会显示用户填写的密码提示问题，只要填写正确的答案就会显示会员的密码。另一种方式是当用户填写完注册信息后单击“注册”按钮时，将会把用户注册的会员名和密码发送到用户填写的邮箱中，当用户忘记密码时可以直接到自己的邮箱中找回自己的密码。

21.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_GetPass 的数据库，在该数据库中创建一个名为 tb_User 的表，该表用来存储用户注册的会员信息。该表的表结构如表 21.1 所示。

表 21.1　tb_User 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动编号
Name	varchar	50	用来登录的会员名
Pass	varchar	50	用来登录的密码
Question	varchar	50	密码提示问题
Answer	varchar	50	密码提示答案
Email	varchar	50	电子邮件地址
CongealDate	datetime	8	冻结密码找回功能的日期

21.1.3 密码找回流程图

在会员密码找回过程中，用户首先需要输入要找回密码的会员名，输入后单击“下一步”按钮，此时判断用户输入的会员名是否存在，如果存在才可以进入回答密码提示问题页面，当用户填写完密码回答问题后单击“查找”按钮，判断用户输入的密码提示问题是否正确，如果正确将显示密码，不正确还需要判断用户是第几次回答问题，如果回答问题超过 3 次将会冻结此功能 24 小时。密码找回流程如图 21.1 所示。

21.2 关 键 技 术

专题讲座：光盘\MR\Video\21\关键技术.exe

➤➤➤视频速递：使读者在开发前能够了解项目用到的技术。

在开发前，读者应该了解本项目具体都应用到了什么技术。只有了解所需要的技术，才能在开发时更加得心应手。

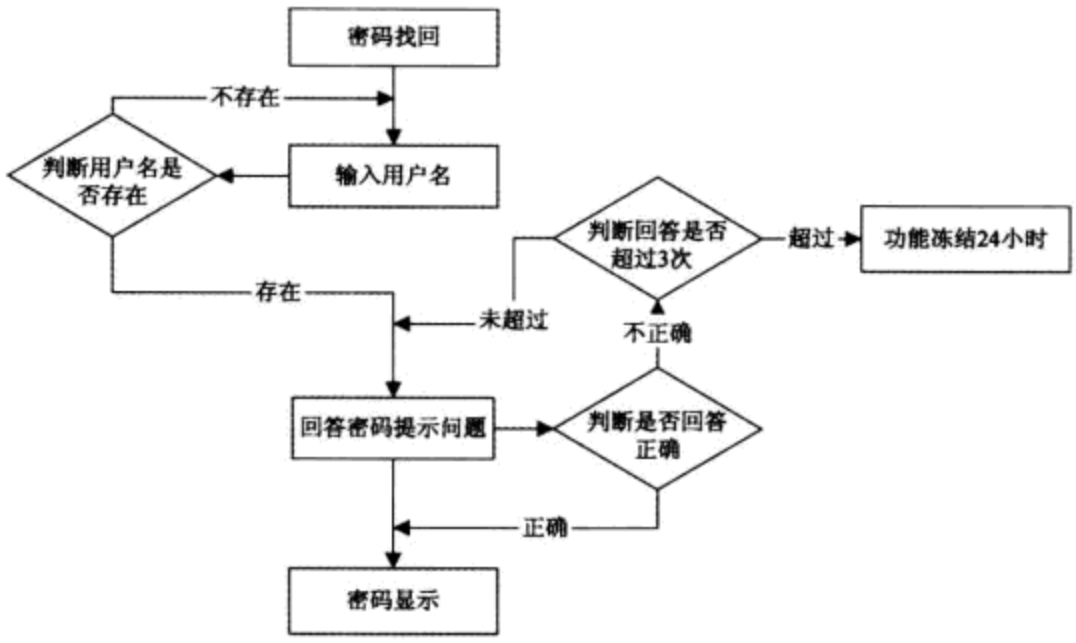


图 21.1　密码找回流程图

21.2.1　会员名验证技术

会员名验证是为了保证会员名的唯一性，目前普遍的会员登录都是通过使用会员名进行登录，如果会员名不唯一就有可能引发登录错误。会员名验证技术首先获取到用户填写的会员名，再通过会员名在数据库的会员注册信息表中进行查询。如果有相同的会员名，说明用户填写的会员名已经存在，并给出提示。会员名验证如图 21.2 所示。

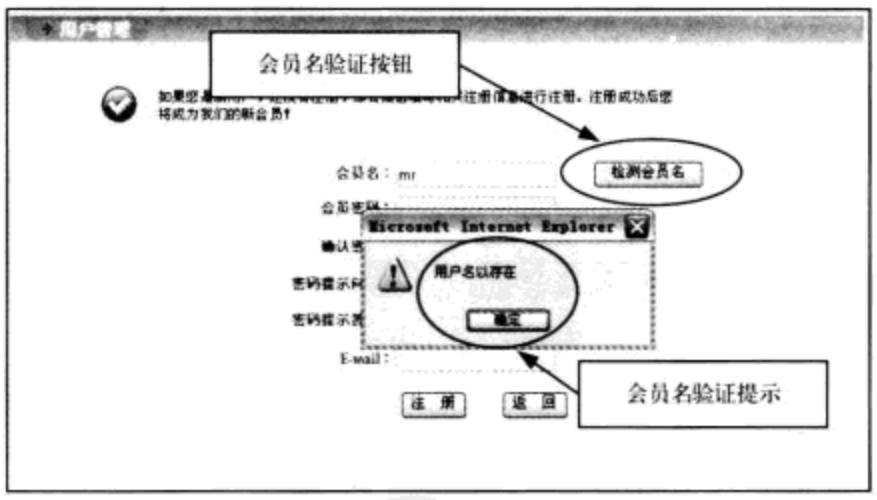


图 21.2　会员名验证

会员名验证是使用“检测会员名”按钮来实现的，在该按钮的单击事件中调用自定义的 isName 方法判断会员名是否存在。调用该方法需要传入一个参数，该参数为字符串类型，表示用户填写的会员名。在该方法中通过连接数据库，并使用 SQL 语句判断会员名是否存在，如果存在则将返回值 True，不存在将返回 False。实现代码如下所示。

```
protected bool isName(string name)
{
    SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings
    ["con"]);
```

```
        con.Open();                                          //打开数据库连接
        string sqlSel = "select count(*) from tb_User where Name='" + name + "'";
        SqlCommand com=new SqlCommand(sqlSel,con);           //创建数据库命令
        if (Convert.ToInt32(com.ExecuteScalar()) > 0) //返回查询结果第一行第一列的结果
        {
            return true;                                     //返回 true 值
        }
        else
        {
            return false;
        }
    }
```

21.2.2 Panel 控件分步显示内容

Panel 控件也可以叫面板控件，该控件中可以包含一组控件或信息。在开发项目中常用该控件来显示或隐藏一组控件或信息。在密码找回过程中，使用该控件是为了将密码找回，该过程分为两个部分进行操作。第一部分就是输入会员名，运行结果如图 21.3 所示。

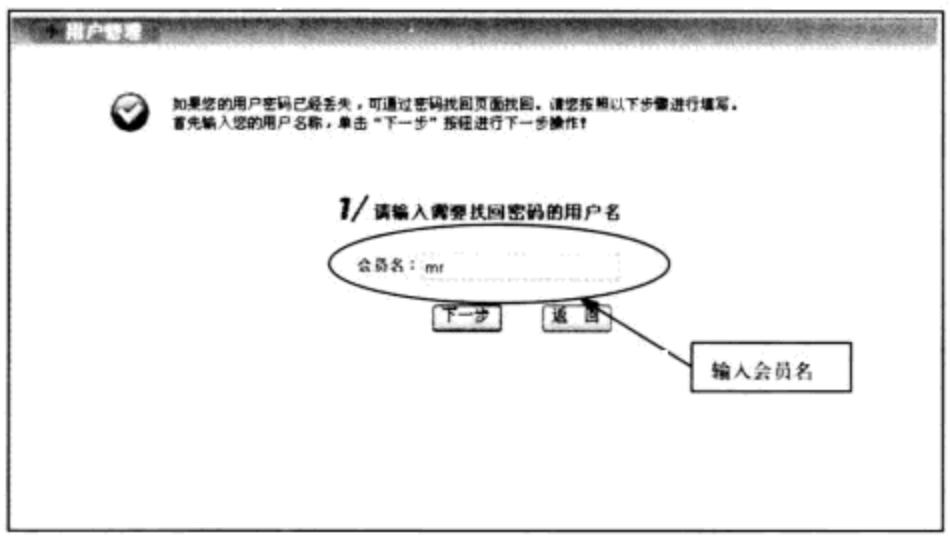


图 21.3 输入会员名

第二部分是回答密码提示问题，运行结果如图 21.4 所示。

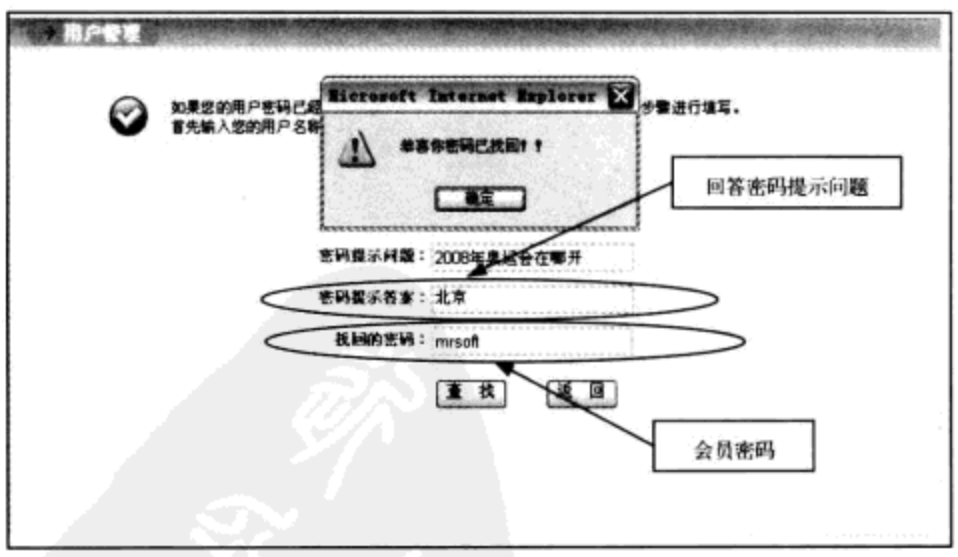


图 21.4 回答密码提示问题

使用 Panel 控件将密码找回分为两个部分操作的优点是能够美化页面和判断会员名。美化页面可以给用户页面简洁、步骤清晰的感觉。而判断会员名是先判断用户输入的会员

名是否存在，如果存在才可以让用户进行回答密码提示问题的操作。如果会员名不存在就不让用户进行回答密码提示问题的操作。

在本程序中主要使用到了 Panel 控件的 Visible 属性。

Visible 属性用来设置或获取 Panel 控件是否显示或隐藏在页面上。该属性有 True 和 False 两个参数，默认值为 True。

21.2.3　发送邮件技术

在用户填写完注册信息后，单击“注册”按钮将用户填写的会员名和密码信息发送到用户填写的邮箱中。当用户忘记密码或会员名时可以在自己的邮箱中找回。

本程序中的邮件发送是利用 SMTP 进行发送的。SMTP 是 Windows 系统自带的邮件服务器，用于发送邮件。在使用 SMTP 发送邮件前需要先安装配置 SMTP 服务器。SMTP 的安装和配置将会在后面的章节中介绍。配置完成 SMTP 后，在 ASP.NET 程序中导入 System.Web.Mail 命名空间，通过调用该命名空间中的 MailMessage 对象来创建邮件，并设置邮件信息。发送邮件信息是通过 SmtpMail 类来完成的。

MailMessage 类用于构造电子邮件，该类的常用属性及说明如表 21.2 所示。

表 21.2　MailMessage 类常用属性及说明

属　性	说　明
From	发送邮件人的电子邮件地址
To	接收邮件人的邮件地址列表，多个地址以分号分隔
Subject	电子邮件的主题
Body	电子邮件的正文
BodyFormat	电子邮件正文的内容类型，由 MailFormat 枚举值指定，可以是 Html 或 Text
Priority	电子邮件的优先级，由 MailPriority 枚举值指定，可以是 Low、Normal 或 Hight

当单击“注册”按钮时，在“注册”按钮的单击事件中将会调用自定义的 sendMail 方法。SendMail 方法将会完成邮件发送的操作。调用该方法需要传入 3 个参数，这 3 个参数都是字符串类型，分别表示会员注册名、登录密码和用户的邮件地址。实现代码如下所示。

```
protected void sendMail(string name, string pass,string email)
{
    try {
        string Email = email;   //电子邮件地址
        //电子邮件的内容
        string body = "您好！您在某某网站的注册已经成功，您注册名为：'"+name+"'密码为：'" +
        pass+"'";
        string subject = "某某网站提示！";  //邮件的主题
        //创建 MailMessage 对象
        MailMessage myEmail = new MailMessage();
        //设置发件人地址
        myEmail.From = "xiaoyu@mr.com";
        //设置收件人地址
        myEmail.To = email;
        //设置邮件主题
```

```
            myEmail.Subject = subject;
            //设置邮件内容
            myEmail.Body = body;
            //设置邮件正文类型
            myEmail.BodyFormat = MailFormat.Text;
            //设置服务器名
            SmtpMail.SmtpServer = "MRSY";
            //发送电子邮件
            SmtpMail.Send(myEmail);
            RegisterStartupScript("true", "<script>alert('发送成功！')</script>");
        }
        catch (Exception ex)
        {
            Response.Write("邮箱不存在");
        }
    }
```

21.2.4 3 次找回密码机会

3 次找回密码的机会是为了防止非法用户多次尝试回答密码问题而获得密码。本程序中的该功能只给用户 3 次回答密码提示问题的机会，当用户操作 3 次以后，该功能将会冻结 24 小时，用户要在 24 小时后才可以使用该功能，效果如图 21.5 所示。

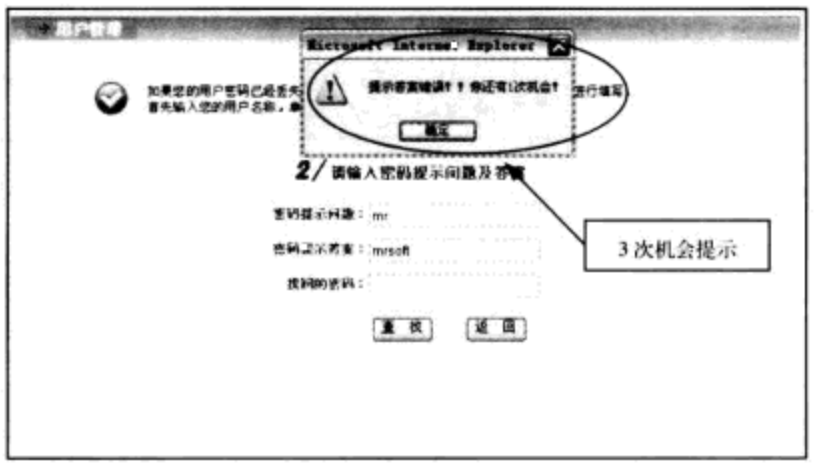


图 21.5 密码找回 3 次机会

实现此功能需要先创建一个静态的整型变量，并将该变量赋值为 0。在“查找”按钮的单击事件中先将整型变量累加。如果用户回答密码提示问题错误，将判断整型变量是否小于 3，如果小于 3 将给出还有几次机会的提示，如果大于 3 将给出用户冻结 24 小时的信息提示。

21.2.5 SMTP 服务的安装与配置

SMTP 是 Windows 系统自带的一种简单的邮件服务器，用于发送邮件，在 ASP.NET 中可以通过 System.Web.Mail 类创建邮件。在网站设计过程中，往往需要用到邮件发送的功能，如果发送量不是很大，SMTP 完全能够胜任。需要注意的是在使用 SMTP 邮件服务器之前，首先需要安装并配置好 SMTP 服务器，然后在 ASP.NET 程序中通过调用 MailMessage 对象来创建邮件，设定好发件人信息、收件人信息就可以发送邮件了。

ASP.NET 发送的邮件会被 SMTP 所接受，正常情况下，SMTP 会将接受的邮件加入发送队列中，进行相关的操作。

使用 System.Web.Mail 命名空间中的 SmtpMail 类发送电子邮件时要求保存在可用的 SMTP 服务器中，最方便的方法就是使用 Windows 2003 内置的 SMTP 服务组件。下面分别来介绍 SMTP 服务组件的安装与配置。

1．安装 SMTP 服务

在默认情况下，IIS 中并没有安装 SMTP 服务，可通过“控制面板”中“添加或删除程序”来安装。安装 SMTP 服务时将创建一个默认的 SMTP 配置，用户随后可以使用 IIS 管理器自定义该配置。SMTP 服务的安装步骤说明如下。

（1）选择“开始”/“设置”/“控制面板”/“添加或删除程序”命令，打开“添加或删除程序”对话框，如图 21.6 所示。

（2）单击对话框左侧的“添加/删除 Windows 组件”，打开“Windows 组件向导”对话框，如图 21.7 所示。

图 21.6　“添加或删除程序”对话框

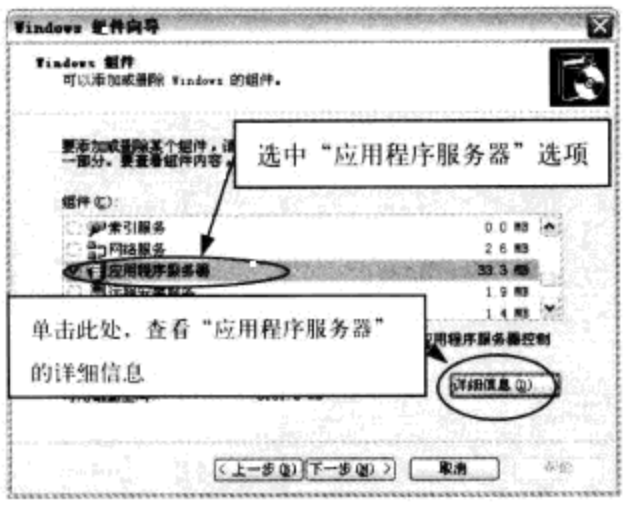


图 21.7　“Windows 组件向导”对话框

（3）选中“应用程序服务器”复选框，单击“详细信息”按钮，打开“应用程序服务器”对话框，如图 21.8 所示。

（4）选中“Internet 信息服务(IIS)”复选框，单击“详细信息”按钮，打开“Internet 信息服务（IIS）”对话框，选中“SMTP Services”复选框，准备安装 SMTP 服务，如图 21.9 所示。

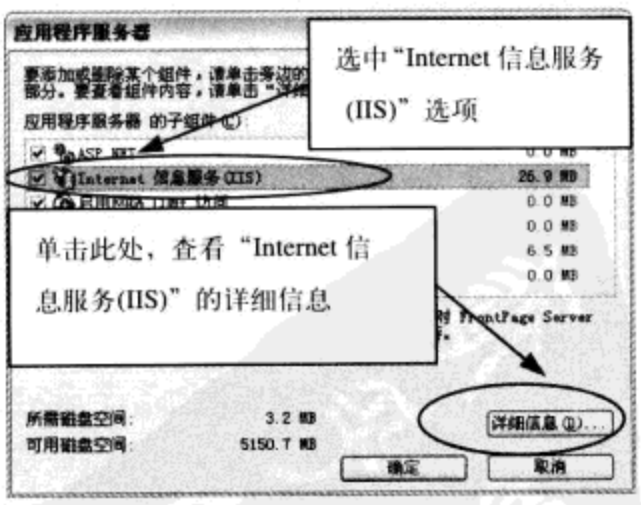


图 21.8　“应用程序服务器”对话框

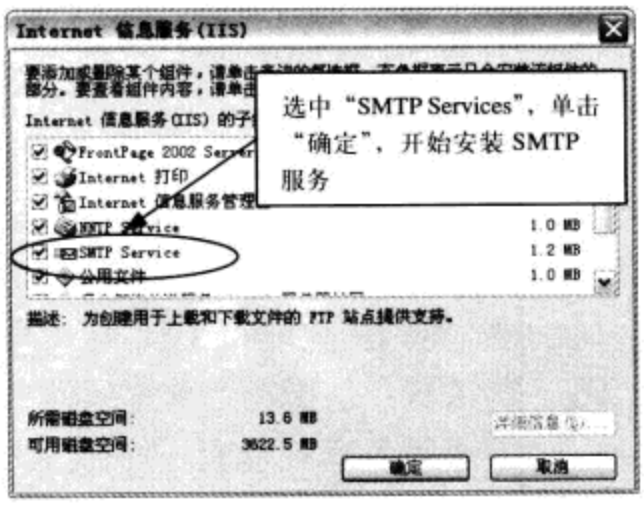


图 21.9　“Internet 信息服务（IIS）”对话框

（5）单击“确定”按钮，然后单击“下一步”按钮，安装完成后，单击“完成”按钮，完成邮件传输协议（SMTP）服务的安装。

由于安装 SMTP 服务时，将在 C:\Inetpub\Mailroot 中创建一个具有消息存储区的默认 SMTP 服务器配置。设置 SMTP 服务时，可以为 SMTP 服务配置全局设置，还可以为虚拟服务器的单个组件配置设置。IIS SMTP 服务只是一个中继代理，电子邮件将转发到 SMTP 服务器进行传递。

2. 配置 SMTP 虚拟计算机

安装 SMTP 服务后，系统将会在 IIS 管理器中创建一个新节点。若要配置 SMTP 虚拟服务器，必须启动 IIS 管理器。

（1）依次单击“开始”/“设置”/“控制面板”/“Internet 信息服务(IIS)管理器”选项，打开如图 21.10 所示的“Internet 信息服务(IIS)管理器”对话框。

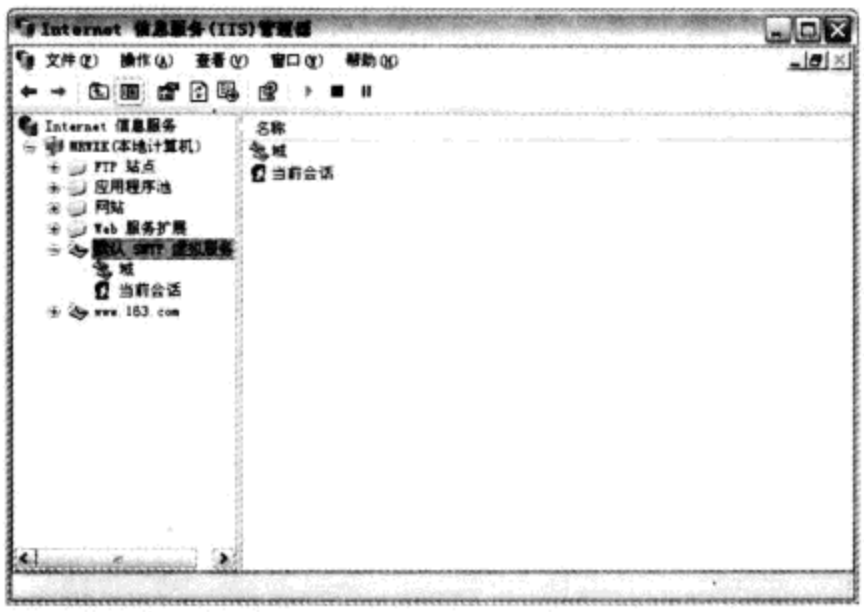


图 21.10 “Internet 信息服务（IIS）管理器”对话框

（2）默认设置

SMTP 虚拟服务器具有以下默认设置。如果想创建一个新的虚拟服务器，可以使用“新建虚拟服务器向导”来配置。

- ☑ 名称：出现在 IIS 管理器中的虚拟服务器的名称。可以在 IIS 管理器中更改虚拟服务器的名称。只需在该虚拟服务器上单击鼠标右键，然后选择“重命名”选项即可。
- ☑ IP 地址/TCP 端口：IP 地址均为“未分配”，TCP 端口号为 25。可以使用“SMTP 虚拟服务器属性”对话框中的“常规”选项卡来更改该设置。如果更改该设置，则必须指定一个没有被其他 SMTP 虚拟服务器使用的 IP 地址和 TCP 端口号组合。TCP 端口号 25 是默认的 TCP 端口号，也是建议的 TCP 端口号。多个虚拟服务器可以使用同一个 TCP 端口号，但是必须为它们配置不同的 IP 地址。如果不设置唯一的 IP 地址和 TCP 端口号组合，则 SMTP 虚拟服务器将不会启动。
- ☑ 默认域：是指在“系统属性”的“计算机名称”选项卡中列出的域名。一个 SMTP 虚拟服务器只能有一个默认域，且不能删除该域。若要在 IIS 管理器中更改默认域的名称，请双击“虚拟服务器”选项，然后双击“域”选项，在右侧列表中选择“本地（或默认）域”选项，然后右键单击本地（或默认）域，在弹出的快捷

菜单中选择“重命名”选项。

☑ 主目录：C:\Inetpub\Mailroot。主目录是 SMTP 的根目录，必须是运行 SMTP 服务计算机的本地目录。

21.3　会员密码找回的实现过程

专题讲座：光盘\MR\Video\21\主要开发过程.exe

➤➤➤视频速递：使读者更加直观地了解本项目的整个开发过程。

从本节开始，我们将介绍会员密码找回功能的整个开发过程。通过图文并茂的形式介绍开发过程使读者更容易理解和掌握，希望读者对本节内容能够认真阅读。

21.3.1　用户登录设计

用户登录页面用来实现用户的登录，在该页面中还可以跳转到会员注册页面和密码找回页面。用户登录页面如图 21.11 所示。

图 21.11　用户登录页面

1．前台页面设计

（1）创建一个 Web 窗体，命名为 Default.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 21.3 所示。

表 21.3　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtName	均为默认值	输入会员登录名
	txtPass	设置 TextMode 属性为 Password	输入会员登录密码
标准/Button 控件	btnEnter	均为默认值	实现登录操作
标准/LinkButton 控件	linkBtnRegister	设置 PostBackUrl 属性为~/Register.aspx	跳转会员注册页面
	linkBtnGetPass	设置 PostBackUrl 属性为~/GetPass.aspx	跳转密码找回页面

2. 后台代码编写

在“登录”按钮的单击事件中，将查询用户输入的会员名和密码是否正确。如果正确将给出相应的提示。实现代码如下所示。

```
protected void btnEnter_Click(object sender, EventArgs e)
    {
        //获取会员登录名
        string name = txtName.Text;
        //获取登录密码
        string pass = txtPass.Text;
        //编写 SQL 语句，查询表中是否拥有满足指定条件的记录
        string sqlSel = "select count(*) from tb_User where Name='" + name + "'and
        Pass='" + pass + "' ";
        //创建数据库连接
        SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings
        ["con"]);
        //打开连接
        con.Open();
        SqlCommand com = new SqlCommand(sqlSel, con);
        //判断返回的参数是否大于 0，如果大于 0 则说明登录成功
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
        {
            RegisterStartupScript("", "<script>alert('登录成功！')</script>");
        }
        else
        {
            RegisterStartupScript("", "<script>alert('登录失败！')</script>");
        }
    }
```

21.3.2 会员注册设计

会员注册页面用来实现会员的注册功能，在该页面中用户需要填写会员的基本信息，如会员名、密码、密码提示问题等。会员注册页面如图 21.12 所示。

图 21.12 会员注册页面

1．前台页面设计

（1）创建一个 Web 窗体，命名为 Register.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 21.4 所示。

表 21.4　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtName	均为默认值	填写会员登录名
	txtPass	设置 TextMode 属性为 Password	填写会员登录密码
标准/TextBox 控件	txtValidatePass	设置 TextMode 属性为 Password	填写确认密码
	txtPassQuestion	均为默认值	填写密码提示问题
	txtPassAnswer	均为默认值	填写密码提示答案
	txtEmail	均为默认值	填写电子邮箱地址
标准/Button 控件	btnIsName	设置 CausesValidation 属性为 False	检查会员名是否存在
	btnRegister	均为默认值	实现注册操作
	btnReturn	设置 PostBackUrl 属性为~/Default.aspx	跳转到首页面

2．后台代码编写

在“检测会员名”按钮的单击事件中，先判断用户输入的会员名是否为空，如果不为空将调用自定义的 isName 方法来查询用户名是否存在，并给出相应的提示。实现代码如下所示。

```
protected void btnIsName_Click(object sender, EventArgs e)
{
    //判读用户输入的会员名是否为空，为空则给出提示
    if (txtName.Text != "")
    {
        //调用自定义方法判断用户输入的会员名是否存在
        if (isName(txtName.Text))
        {
            RegisterStartupScript("", "<script>alert('用户名以存在')</script>");
        }
        RegisterStartupScript("", "<script>alert('可以注册')</script>");
    }
    else {
        RegisterStartupScript("", "<script>alert('用户名不能为空！')</script>");
    }
}
```

自定义的 isName 方法用来查询用户输入的会员名是否存在。调用该方法需要传入一个参数，该参数为字符串类型，表示用户输入的会员名。该方法将会返回一个布尔值，当该值为 True 时表示用户查询的会员名已经存在，如果为 False 表示会员名不存在。实现代码如下所示。

```
protected bool isName(string name)
{
    //创建数据库连接
    SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings
    ["con"]);
```

```
        //打开数据库
        con.Open();
        //创建 SQL 语句，查询指定的条件是否存在
        string sqlSel = "select count(*) from tb_User where Name='" + name + "'";
        //创建 sqlCommand 对象
        SqlCommand com=new SqlCommand(sqlSel,con);
        //判断 sqlCommand 对象的 ExecuteScalar 方法返回的参数是否大于 0，大于 0 说明用户已存在
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

在“注册”按钮的单击事件中，也需要判断一下用户输入的会员名是否存在，这样做的目的是为了防止用户没有单击“检测会员名”按钮。如果会员名不存在，将获取到的相关信息通过 SQL 语句添加到数据中，如果添加成功，可调用自定义的 sendMail 方法将用户的会员名及密码发送到用户填写的邮箱中。实现代码如下所示。

```
protected void btnRegister_Click(object sender, EventArgs e)
{
    //获取用户名
    string Name = txtName.Text;
    //获取密码
    string Pass = txtPass.Text;
    //获取密码提示问题
    string PassQuestion = txtPassQuestion.Text;
    //获取密码提示答案
    string PassAnswer = txtPassAnswer.Text;
    //获取 E-mail
    string Email = txtEmail.Text;
    if (!isName(txtName.Text))
    {
        //创建数据库连接
        SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings
        ["con"]);
        //打开数据库连接
        con.Open();
        //编写 SQL 语句，插入用户注册信息
        string Sql = "insert into tb_User values('" + Name + "','" + Pass + "','"
        + PassQuestion + "','"
            + PassAnswer + "','" + Email + "','"+DateTime.Now.ToString()+"')";
        SqlCommand com = new SqlCommand(Sql, con);
        //判断 SQL 语句是否执行成功
        if (Convert.ToInt32(com.ExecuteNonQuery()) > 0)
        {
            RegisterStartupScript("true", "<script>alert('注册成功！')
            </script>");
            //调用自定义方法，并传入 2 个参数，将密码保存到邮件中
            sendMail(Name,Pass, Email);
            txtName.Text = txtPass.Text = txtPassQuestion.Text = txtPassQuestion.
```

```
            Text = txtEmail.Text
               = txtPassAnswer.Text="";
        }
        else
        {
            RegisterStartupScript("false", "<script>alert('注册失败！')
            </script>");
        }
        con.Close();
        }
        else
        {
            RegisterStartupScript("", "<script>alert('用户名以存在')</script>");
        }
    }
```

21.3.3　会员密码找回设计

会员密码找回页面用来实现密码的找回功能，在该页面中会员首先输入自己的会员名，如图 21.13 所示。通过回答密码提示问题，即可找回自己的密码，如图 21.14 所示。

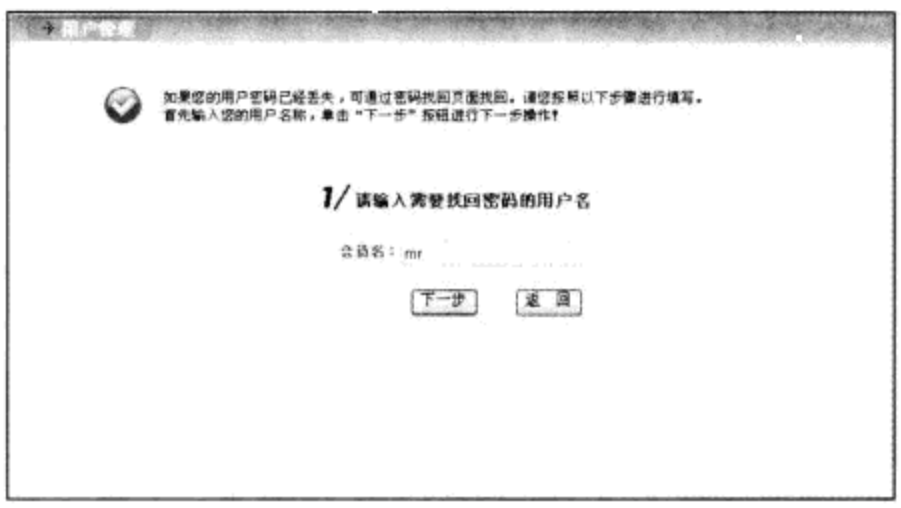


图 21.13　密码找回第 1 步

图 21.14　密码找回第 2 步

1．前台页面设计

（1）创建一个 Web 窗体，命名为 GetPass.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 21.5 所示。

表 21.5　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtName	均为默认值	填写会员登录名
	txtQuestion	均为默认值	填写密码提示问题
	txtAnswer	均为默认值	填写密码提示答案
标准/Button 控件	BtnNext	设置 CausesValidation 属性为 False	进入回答密码提示问题操作
	btnNreturn	设置 PostBackUrl 属性为~/Default.aspx	跳转到首页面
	btnGet	均为默认值	实现找回密码操作
	btnReturn	设置 PostBackUrl 属性为~/Default.aspx	跳转到首页面
标准/Panel 控件	PanelInputName	均为默认值	显示输入登录用户名区域
	PanelGetPass	设置 Visible 属性为 False	显示回答密码提示问题区域

2. 后台代码编写

密码找回操作是分为两步进行的，当用户输入完会员名后单击“下一步”按钮，在“下一步”按钮的单击事件中，首先调用自定义的 getData 方法并接收该方法返回的 SqlDataReader 对象，根据 SqlDataReader 对象的 Reade 方法判断用户输入的会员名是否存在。如果存在，再判断用户是否被冻结了密码找回功能。如果未冻结，将隐藏输入会员名区域的 Panel，显示回答密码提示问题区域的 Panel。实现代码如下所示。

```
protected void btnNext_Click(object sender, EventArgs e)
{
    //调用自定义方法，并传入 2 个参数，用来查询用户名是否存在
    SqlDataReader sdr = getData("select * from tb_User where Name=@value",
    txtName.Text);
    //判断用户名是否存在
    if (sdr.Read())
    {
        //获取冻结密码找回的日期
        DateTime congeal = Convert.ToDateTime(sdr["congealDate"]);
        //创建 TimeSpan 对象，该对象表示两个时间的间隔
        TimeSpan ts = DateTime.Now - congeal;
        //获取两个时间的间隔以小时表示
        int hours = Convert.ToInt32(ts.TotalHours);
        //判断时间是否大于个 24，如果大于将显示回答密码提示问题区域
        if (hours > 24)
        {
            //如果存在隐藏输入用户名区域
            PanelInputName.Visible = false;
            //显示找回密码区域
            PanelGetPass.Visible = true;
            //显示密码提示问题
            txtQuestion.Text = sdr["Question"].ToString();
        }
        else
        {
            RegisterStartupScript("", "<script>alert('还有" + (24 - hours)
            + "小时后可以使用该功能！！')</script>");
        }
    }
    else
    {
        RegisterStartupScript("false", "<script>alert('用户名不存在！！')
```

```
            </script>");
        }
    }
```

自定义的 getData 方法用来查询用户输入的会员名是否存在。调用该方法需要传入两个参数，这两个参数是字符串类型，分别表示 SQL 语句和会员名。该方法将返回一个 SqlDataReader 对象。实现代码如下所示。

```
protected SqlDataReader getData(string sql, string value)
{
    //创建数据库连接
    SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings
    ["con"]);
    //打开数据库连接
    con.Open();
    //创建 sqlCommand 对象
    SqlCommand com = new SqlCommand(sql, con);
    //添加一个参数，并指定参数的类型及大小
    com.Parameters.Add(new SqlParameter("@value", SqlDbType.VarChar, 50));
    //设置参数的值
    com.Parameters["@value"].Value = value;
    //返回 DataReader 对象
    return com.ExecuteReader();
}
```

在“查找”按钮的单击事件中，判断用户输入的密码提示问题是否正确，如果正确将显示会员密码。如果不正确将通过变量 i 判断用户是第几次回答密码提示问题，如果回答的次数超过 3 次，将通过 SQL 语句更新回答密码提示问题的冻结时间并返回首页。实现代码如下所示。

```
protected void btnGet_Click(object sender, EventArgs e)
  {
    //该变量是公共的静态变量用来保存回答密码提示问题的次数调用自定义方法，并传入 2 个参数，
      //用来查询密码提示答案是否正确
    SqlDataReader sdr = getData("select * from tb_User where Name='" + txtName.
    Text +
      "' and Answer=@value", txtAnswer.Text);
    //判读密码提示答案是否存在
    if (sdr.Read())
    {
        //判断用户回答的密码提示问题是否正确
        if (txtAnswer.Text == sdr["Answer"].ToString())
        {
            //显示会员的密码
            txtPass.Text = sdr["Pass"].ToString();
            RegisterStartupScript("true", "<script>alert('恭喜你密码已找回！！')
            </script>");
        }
    }
    else
    {
        //判断用户是否已回答了 3 次问题
        if (i < 3)
        {
```

```
                RegisterStartupScript("false", "<script>alert('提示答案错误！！你还有"
                + (3 - i)
                    + "次机会！')</script>");
            }
            else
            {
                i = 0;
                //创建 SQL 语句，更新会员回答密码提示问题的冻结时间
                string sql = " update tb_User set congealDate='" + DateTime.Now.
                ToString() + "'
                    where Name='" + txtName.Text + "' ";
                //创建数据库连接
                SqlConnection con = new SqlConnection(ConfigurationManager.
                AppSettings["con"]);
                //打开数据连接
                con.Open();
                //创建 SqlCommand 对象
                SqlCommand com = new SqlCommand(sql, con);
                //执行 Sql 语句
                com.ExecuteNonQuery();
                RegisterStartupScript("false", "<script>alert('您3次回答问题的机会以用完!
                    在 24 小时后才可以使用此功能！');location='Default.aspx'</script>");
            }
        }
    }
```

21.4 本章小结

会员密码找回功能在网站的实际开发中非常重要，用户在不慎将密码丢失后如果找不回来将是对用户的一个损失，尤其是非常重要的密码，所以本章重点讲解了密码找回功能。同时为了防止不法用户破译其他合法用户的密码，会员密码找回功能设置了冻结功能。

第22章

完美实现网络硬盘

（名师课堂：1小时8分钟）

随着网络的盛行，网络硬盘也随之而生。顾名思义，网络硬盘就是在网络中提供一个存储资料的大仓库。我们可以将文件上传到网络硬盘中，当需要的时候再从网络硬盘中将文件下载到本地计算机。网络硬盘最大的优点是不受区域的限制，当将文件上传到网络硬盘之后，就可以在任何地点下载文件，通过本章的学习，读者能够学到以下内容：

- 注册网络硬盘会员
- 创建个性目录
- 带进度条上传文件
- 编辑文件及个性目录
- 下载网络硬盘中的文件
- 修改个人资料信息
- 修改个性头像
- 网络硬盘空间被修改后即时提醒功能

22.1 网络硬盘概述

相信读者对硬盘不会感觉陌生，硬盘是计算机的仓库，计算机中所有的资料都存储在这个仓库之中。随着网络的盛行，网络硬盘也随之而生。顾名思义，网络硬盘就是在网络中提供一个存储资料的大仓库。我们可以将文件上传到网络硬盘中，当需要的时候再从网络硬盘中将文件下载到本地计算机。网络硬盘最大的优点是不受区域的限制，当将文件上传到网络硬盘里，就可以在任何地点下载文件，这样就不需要使用 U 盘或者光盘等介质来存储资料了。在网上作者也看到过很多免费网络硬盘的网站，于是决定自己也做一个网络硬盘的小程序，以便能提供给广大读者学习。网络硬盘可以为用户存储文件，当需要的时候可以下载文件。作者是典型的网虫，在多年的上网过程中，总结出网络硬盘应该具有的基本功能有以下几点。

- ☑ 网络硬盘用户注册功能。
- ☑ 显示用户的基本资料，包括头像、用户名、注册时间和注册 IP 地址。
- ☑ 显示用户的网络硬盘状态，包括拥有的空间大小、已用空间大小、剩余空间大小和空间容量进度条。
- ☑ 为用户提供创建目录和文件上传的功能。
- ☑ 提供删除和下载文件的功能。
- ☑ 提供修改个人资料的功能。

22.1.1 系统功能结构图

本章的网络硬盘程序主要分为用户注册模块、用户登录模块、文件上传模块、文件管理模块和修改资料模块等，为了使读者更好地了解本章网络硬盘的功能结构，作者绘制了网络硬盘的功能结构图，如图 22.1 所示。

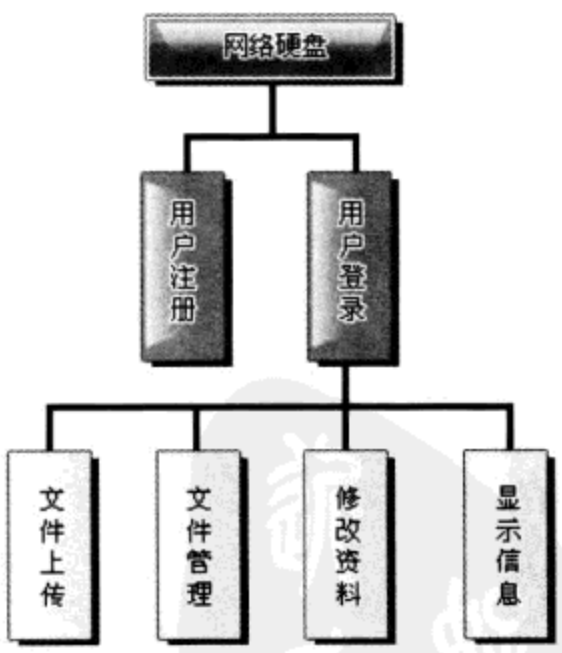


图 22.1 系统功能结构图

22.1.2 系统预览

作者根据网络硬盘的基本需求，开发出本章的网络硬盘系统，仅供读者参考。本章的

网络硬盘系统实现了网络硬盘的大部分功能，具体细节读者可以自己完善，本章网络硬盘系统运行结果如图 22.2 所示。

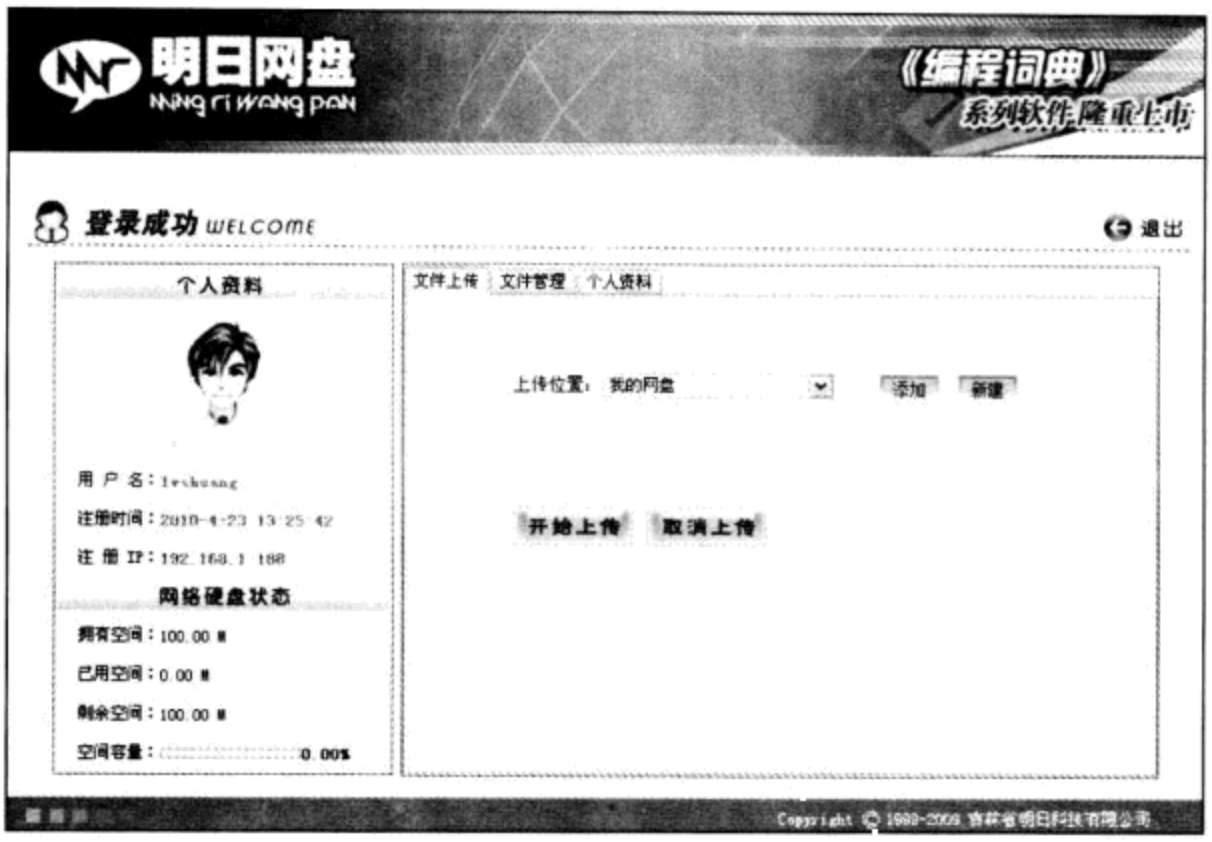


图 22.2　网络硬盘界面

22.2　数据库设计

本章的网络硬盘系统使用的是 SQL Server 2005 数据库，由于网络硬盘主要是对文件及文件夹进行操作，所以本章网络硬盘系统只需要一个数据表 tb_user 用于存储网络硬盘用户的注册信息，包括用户登录名、登录密码、用户昵称、电子邮件地址和给用户分配的网络硬盘空间大小等，tb_user 数据表结构如表 22.1 所示。

表 22.1　tb_user 用户信息表结构

字 段 名	数 据 类 型	长　度	主 键 否	描　述
ID	int	50	主键	数据编号
u_name	varchar	50	否	注册用户名
u_nickname	varchar	50	否	注册用户的昵称
u_pwd	varchar	50	否	注册用户的登录密码
u_email	varchar	50	否	注册用户的电子邮件地址
u_date	varchar	50	否	注册时间
u_ip	varchar	50	否	注册用户的 IP 地址
u_img	varchar	50	否	用户头像地址
u_disk	decimal	19	否	用户拥有的网络硬盘空间大小
u_identity	int	4	否	用户权限
u_flag	int	4	否	标记硬盘空间的大小是否被修改

22.3 关键技术详解

专题讲座：光盘\MR\Video\22\关键技术.exe

➤➤➤视频速递：使读者在开发前能够了解项目用到的技术。

在开发前，读者应该了解本项目具体都应用到了什么技术。只有了解所需要的技术，才能在开发时更加得心应手。

22.3.1 上传文件

上传文件的方式有很多种，最常见的是通过 ASP.NET 服务器控件 FileUpload 进行上传。随着 jQuery 的流行，越来越多的人使用 jQuery 插件开发程序，本例就是使用 jQuery 的 uploadify 插件实现文件上传功能的。uploadify 插件非常漂亮，可以显示上传进度条，功能也很强大，而且为用户提供了很多回调函数，方便用户编写自己的代码。关于 uploadify 插件我们在 16.1 节中已经介绍得很详细， uploadify 插件选择上传文件后，将上传文件传给后台处理程序。真正将上传的文件保存到服务器中的代码是在后台处理程序中编写的，所以此处介绍这个后台处理程序。本例中通过创建一个“一般处理程序”用于接收 uploadify 插件传送的文件，并将文件保存到服务器中，用到的主要技术是 HttpRequest 类的 Files 属性和 HttpPostedFile 类的 SaveAs 方法。

（1）Files 属性

获取由客户端上载的文件集合。

语法：

```
public HttpFileCollection Files { get; }
```

属性值：客户端上载的文件集合。

（2）SaveAs 方法

保存上载文件的内容。

语法：

```
public void SaveAs(string filename)
```

filename：保存的文件名称。

例如，在本实例中首先通过 Files 属性实例化一个 HttpPostedFile 对象，然后通过该对象的 SaveAs 方法保存客户端上载的文件，代码如下所示：

```
public void ProcessRequest (HttpContext context) {
try
{
    if (context.Request.Files.Count > 0)                    //如果文件集合大于 0
    {
        //获取文件上传的路径
        string tempFile
        = context.Request.PhysicalApplicationPath+context.Request["folder"].
        Trim('/').Replace("/","\\");
```

```
            for (int j = 0; j < context.Request.Files.Count; j++)   //遍历文件集合
            {
                //对客户端上传的文件单独进行访问
                HttpPostedFile uploadFile = context.Request.Files[j];
                if (uploadFile.ContentLength > 0)                   //如果上传的文件长度大于 0
                {
                    //保存文件到指定的目录下
                    uploadFile.SaveAs(tempFile+"\\"+System.IO.Path.GetFileName
                    (uploadFile.FileName));
                }
            }
            HttpContext.Current.Response.Write(" ");
        }
        catch
        {
            context.Response.ContentType = "text/plain";
            context.Response.Write("Hello World");
        }
    }
```

22.3.2　创建文件夹

图 22.3　创建文件夹

在网络硬盘程序中给用户提供创建新目录的功能，以便用户可以将文件存储到创建的目录中，这个功能的原理就是客户端在服务端创建文件夹，如图 22.3 所示。

在 ASP.NET 中是通过 Directory 类的 CreateDirectory 方法创建文件夹。CreateDirectory 方法根据参数 path 创建目录。

语法：

```
public static DirectoryInfo CreateDirectory(string path)
```

path：要创建的目录路径。

返回值：由 path 指定的 DirectoryInfo。

例如，在本例中，用户登录成功进入网络硬盘管理界面后，在上传文件区域中，可以单击“新建”按钮打开模态窗口，在模态窗口中通过 Directory 类的 CreateDirectory 方法创建文件夹，代码如下所示：

```
protected void ImgBtnCreateDir_Click(object sender, ImageClickEventArgs e)
                                                                    //创建文件夹
{
    //判断是否输入新创建文件夹的名称，如果没有输入名称
    if (txtDirName.Text.Trim().Length == 0)
    {
        //直接关闭当前页，返回到父窗体中
        Response.Write("<script>window.opener=null;window.close();</script>");
    }
    else                                                            //如果输入了文件夹名称
    {
        string path = Server.MapPath(newPath);//获取登录用户空间在服务器上的绝对路径
            //在其空间中创建文件夹，名称是输入的名称
        Directory.CreateDirectory(path + "\\" + txtDirName.Text.Trim());
```

```
            //关闭当前窗体，返回值为 1
            Response.Write("<script>window.returnValue=1;window.close();</script>");
        }
    }
```

学习手册 注意 创建文件夹时，文件夹的路径必须是服务器上的路径。创建文件夹可以通过 Directory 类的静态方法 CreateDirectory 创建，也可以通过实例化 DirectoryInfo 对象，然后调用该对象的 Create 方法创建。

22.3.3 删除文件或文件夹

在文件管理区域中，用户可以删除上传的文件或者文件夹，删除文件夹或文件如图 22.4 和图 22.5 所示。

图 22.4 单击“删除”按钮删除文件夹

图 22.5 单击“删除”按钮删除文件

ASP.NET 中是通过 FileInfo 对象的 Delete 方法删除文件的，而删除文件夹是通过 DirectoryInfo 对象的 Delete 方法实现的，下面对这两个 Delete 方法分别进行介绍。

（1）FileInfo 对象的 Delete 方法

FileInfo 对象的 Delete 方法用于永久删除文件。

语法：

```
public override void Delete()
```

例如，在本例的文件管理区域中，选择文件的“编辑”按钮打开模态窗口，在模态窗口中，单击“删除”按钮可以删除文件，代码如下所示：

```
if (Directory.Exists(Server.MapPath("Net/" + dirname)))          //如果是文件夹
{
    string newName = "Net/" + dirname;                           //获取文件夹路径
    DirectoryInfo di = new DirectoryInfo(Server.MapPath(newName));
                                                                 //实例化 DirectoryInfo 对象
    //调用该对象的 Delete 方法删除文件夹
    di.Delete(true);
    Response.Write("<script>window.returnValue=1;window.close();</script>");
                                                                 //关闭窗口，返回值为 1
}
```

（2）DirectoryInfo 对象的 Delete 方法

删除 DirectoryInfo 的实例，指定是否要删除子目录和文件。

语法：

```
public void Delete(bool recursive)
```

例如，在本例的文件管理区域中，选择文件夹的“编辑”按钮打开模态窗口，在模态

窗口中，单击“删除”按钮可以删除文件夹，代码如下所示：

```
if (Directory.Exists(Server.MapPath("Net/" + dirname)))          //如果是文件夹
{
    string newName = "Net/" + dirname;                           //获取文件夹路径
    DirectoryInfo di = new DirectoryInfo(Server.MapPath(newName));
                                                                 //实例化 DirectoryInfo 对象
    //调用该对象的 Delete 方法删除文件夹
    di.Delete(true);
    Response.Write("<script>window.returnValue=1;window.close();</script>");
                                                                 //关闭窗口，返回值为 1
 }
```

22.3.4　文件或文件夹更名

在文件管理区域中，用户可以对上传的文件或者创建的文件夹进行更名操作。文件或文件夹的更名与删除文件或文件夹在同一个窗口，请读者参考 22.3.3 节中的插图。ASP.NET 中是通过 FileInfo 对象的 MoveTo 方法实现文件更名，而修改文件夹名称是通过 DirectoryInfo 对象的 MoveTo 方法实现的，下面对这两个 MoveTo 方法分别进行介绍。

（1）FileInfo 对象的 MoveTo 方法

将指定文件移到新位置，并提供指定新文件名的选项。

语法：

```
public void MoveTo(string destFileName)
```

destFileName：要将文件移动到的路径，可以指定另一个文件名。

例如，在本例的文件管理区域中，选择文件的“编辑”按钮打开模态窗口，在模态窗口中，输入新的文件名称，单击“修改”按钮可以修改文件名称，代码如下所示：

```
if (File.Exists(Server.MapPath("Net/" + dirname)))     //如果此时是操作文件
 {
    //获取文件更名后的路径
    string newName = "Net/" + dirname.Remove(dirname.LastIndexOf("/") + 1) +
    txtName.Text.Trim();
    FileInfo fi = new FileInfo(Server.MapPath("Net/" + dirname));
                                                       //实例化 FileInfo 对象
    //使用该对象的 MoveTo 方法修改文件名称
    fi.MoveTo(Server.MapPath(newName));
    Response.Write("<script>window.returnValue=1;window.close();</script>");
                                                       //关闭当前窗口，返回值为 1
}
```

（2）DirectoryInfo 对象的 MoveTo 方法

将 DirectoryInfo 实例及其内容移动到新路径。

语法：

```
public void MoveTo(string destDirName)
```

destDirName：要将此目录移动到的目标位置的名称和路径。

例如，在本例的文件管理区域中，选择文件夹的“编辑”按钮打开模态窗口，在模态窗口中，输入新的文件夹名称，单击“修改”按钮可以修改文件夹名称，代码如下所示：

```
if (Directory.Exists(Server.MapPath("Net/" + dirname))) //如果此时是操作文件夹
 {
     //获取文件夹更名后的路径
     string newName = "Net/" + dirname.Remove(dirname.LastIndexOf("/") + 1) +
     txtName.Text.Trim();
     //实例化 DirectoryInfo 对象
     DirectoryInfo di = new DirectoryInfo(Server.MapPath("Net/" + dirname));
     di.MoveTo(Server.MapPath(newName));  //使用该对象的 MoveTo 方法修改文件夹名称
    //关闭当前窗口，返回值为 1
     Response.Write("<script>window.returnValue=1;window.close();</script>");
}
```

学习手册 注意 MoveTo 方法原意是移动文件或者文件夹，但是此处通过该方法实现对文件夹或者文件的更名操作。

22.3.5 下载文件

网络硬盘中上传文件后，在其他能够上网的地方就可以下载上传的文件，大大地方便了网络中文件的传输，在“文件管理”模块中，单击“下载”链接实现文件下载，如图 22.6 所示。

图 22.6 文件下载

下载文件的过程主要是先将文件名添加到 HTTP 头中，然后将文件写入 HTTP 流，再向客户端发送流实现文件的下载，这个过程用到的主要技术是 Response 类的 AppendHeader 方法和 WriteFile 方法，下面介绍这两种方法。

（1）AppendHeader 方法

向随此响应发送的指定 HTTP 标头追加值。

语法：

```
public void AppendHeader(string name,string value)
```

name：要追加 value 的 HTTP 标头的名称。

value：要追加到 name 标头的值。

（2）WriteFile 方法

将指定文件的内容作为文件块直接写入 HTTP 响应输出流。

语法：

```
public void WriteFile(string filename)
```

filename：要写入 HTTP 输出的文件名。

例如，在本例的文件管理区域中，单击“下载”按钮可以下载相应的文件。程序首先获取要下载的文件路径，进行编码后传递给“download.ashx”文件进行下载。所以，在项目中需要添加一个“一般处理程序”，并命名为“download.ashx”，实现文件下载的功能，代码如下所示：

```
public void ProcessRequest (HttpContext context) {
try
{
      //获取要下载的文件路径
    string strs = "Net/" +
     HttpUtility.UrlDecode(context.Request.QueryString["p"].ToString(), System.
     Text.Encoding.UTF8);
     string path = context.Server.MapPath(strs);          //转换成服务器绝对路径
     context.Response.Clear();                            //清除缓冲器内容
     context.Response.Charset = "utf-8";                  //设置输出流的编码方式
     context.Response.Buffer = true;                      //是否处理完之后发送数据
     context.Response.ContentEncoding = System.Text.Encoding.UTF8;
     //对要下载的文件路径进行 UTF8 编码，方式下载时文件名是乱码
    string str =
     System.Web.HttpUtility.UrlEncode(System.IO.Path.GetFileName(path), System.
     Text.Encoding.UTF8);
     context.Response.AppendHeader("Content-Disposition", "attachment;filename="
     + str);                                              //添加 HTTP 头
     context.Response.ContentType = "application/unknow";      //设置 MIME 类型
    context.Response.WriteFile(path);                     //将文件写入 HTTP 流
     context.Response.Flush();                            //向客户端发送流
     context.Response.Close();                            //关闭流
     context.Response.End();                              //停止该页执行
     }
 catch
 {
     context.Response.Write("出现错误！");
 }
}
```

此处需要注意的是要对文件名进行 UTF8 编码，否则当文件名是中文名时，下载文件会出现文件名乱码的问题。

22.4　公共类设计

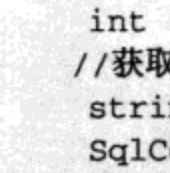
专题讲座：光盘\MR\Video\22\公共类设计.exe

➤➤➤视频速递：使读者了解并掌握本项目所涉及的公共类。

公共类，顾名思义，就是在项目的任何地方都可以调用的类。在开发项目时常以公共

类的形式来组织、封装一些常用的方法和事件，不仅可以提高代码的重用率，也大大方便了代码的管理。在本例中创建了一个公共类，命名为 ComClass。在该公共类中自定义了多个方法，以便在其他页面中调用，下面就详细介绍一下 ComClass 公共类中的这些方法，首先看一下 getCon 方法，该方法用于连接数据库，主要从 web.config 中读取连接数据库的字符串，然后返回一个 SqlConnection 对象，代码如下所示：

```
private static SqlConnection getCon()//声明称静态方法，其返回值为 SqlConnection
{
    return new SqlConnection(System.Configuration.ConfigurationManager.
    AppSettings["constr"].ToString());
}
```



此处 System.Configuration.ConfigurationManager.AppSettings["constr"]用于获取 web.config 中配置的数据库连接字符串，以便能创建 SqlConnection 对象，所以首先在 web.config 中要配置数据库连接字符串，基本格式如下：<appSettings><add key="constr" value="server=LVSHUANG0927\LS;database=db_22;uid=sa;pwd="/></appSettings>。

公共类中的 CheckUser 方法主要用于检查注册用户名是否可用，在本例注册界面会调用此方法，其参数是用户注册的用户名，用到的主要技术就是 SQL 语句中的 count 函数，该函数返回符合查询中指定搜索条件的行的数目，如果数据库中存在用户新注册的用户名则返回值大于 0，主要代码如下所示：

```
public static int CheckUser(string userName)
{
    int flag = 0;                                                    //记录返回值
    SqlConnection conn = getCon();                                   //连接数据库
    conn.Open();                                                     //打开连接
    SqlCommand cmd = new SqlCommand("select count(*) from tb_user where
      u_name='"+userName+"'", conn);  //使用 count 函数获取符合查询条件的行的数目
    flag = int.Parse(cmd.ExecuteScalar().ToString());  //获取返回值
    conn.Close();                                                    //关闭连接
    return flag;                                                     //将返回值返回给调用方
}
```

当用户输入的注册信息无误时，会调用公共类中的 RegUser 方法将注册信息添加到数据库中。该方法的四个参数分别是用户注册的登录用户名、昵称、密码和电子邮箱地址。该方法用到的关键技术主要是 SQL 语句 Insert，相信读者已经对这个语句有很深的了解，此处就不再赘述，主要代码如下所示：

```
public static int RegUser(string name,string nickname,string pwd,string email)
{
    int flag = 0;                                                    //记录返回值
    //获取注册用户的 IP 地址
    string ip=System.Web.HttpContext.Current.Request.UserHostAddress;
    SqlConnection conn = getCon();                                   //连接数据库
    conn.Open();                                                     //打开连接
    //创建 SqlCommand 对象，向数据库中插入数据
    SqlCommand cmd = new SqlCommand("insert
     into tb_user(u_name,u_nickname,u_pwd,u_email,u_date,u_ip) values ('" +
     name + "', '" + nickname + "','"
    + pwd + "','" + email + "','" + DateTime.Now.ToString() + "','"+ip+"')", conn);
```

```
        flag = cmd.ExecuteNonQuery();                    //执行 SQL 语句，获取返回值
        conn.Close();                                    //关闭连接
        return flag;                                     //将返回值返回给调用方
    }
```

System.Web.HttpContext.Current.Request.UserHostAddress 获取的是客户端的 IP 地址，而不是服务器的 IP 地址，读者在此处不要混淆。

公共类中的 ValidateUser 方法主要用于对登录用户进行验证，其参数是一条 SQL 语句，主要通过 SQL 语句中的 count 函数验证登录的用户名和密码是否正确，如果验证成功则返回值大于 0，主要代码如下所示：

```
public static int ValidateUser(string sql)
{
    int flag = 0;                                        //记录返回值
    SqlConnection conn = getCon();                       //连接数据库
    conn.Open();                                         //打开连接
    SqlCommand cmd = new SqlCommand(sql, conn);          //创建 SqlCommand 对象
    flag = int.Parse(cmd.ExecuteScalar().ToString());    //获取返回值
    conn.Close();                                        //关闭连接
    return flag;                                         //将返回值返回给调用方
}
```

getDirectory 方法主要用于获取用户网络硬盘中的所有文件夹，该方法的参数是用户网络硬盘在服务器上的绝对路径，返回值是包含用户网络硬盘中所有文件夹路径的字符串数组。用到的主要技术是 Directory 类的 GetDirectories 方法，该方法用于获取指定目录中子目录的名称，主要代码如下所示：

```
public static string[] getDirectory(string path)
{
    string[] result = null;                              //声明一个数组存储指定目录下的子目录名称
    if (System.IO.Directory.Exists(path))                //如果指定的目录存在
    {
     // Directory 类的 GetDirectories 方法获取指定目录下的所有子目录
       result = System.IO.Directory.GetDirectories(path);
    }
    else                                                 //如果不存在
    {
        result = null;                                   //返回空值
    }
    return result;                                       //返回给调用方
}
```

在遍历某个文件夹之前，要首先判断是否存在该文件夹，否则运行程序时会发生异常，而不能成功遍历文件夹。

returnValues 方法用于返回数据表中指定字段的值，参数分别是字段名称和用户名。例如，如果想得到某用户的年龄，那么就可以将数据库中存储年龄的字段名和该用户名作为参数传给此方法，返回值就是该用户的年龄，returnValues 方法的主要代码如下所示：

```
public static string returnValues(string val ,string name)
{
```

```
    string mysize = string.Empty;                                    //记录返回值
    SqlConnection conn = getCon();                                   //连接数据库
    conn.Open();                                                     //打开连接
    //创建 SqlCommand 对象，返回指定字段的值
    SqlCommand cmd = new SqlCommand("select "+val+" from tb_user where u_name=
    '"+name+"'", conn);
    mysize = cmd.ExecuteScalar().ToString();                         //获取指定字段的值
    conn.Close();                                                    //关闭连接
    return mysize;                                                   //将字段值返回给调用方
}
```

ExecuteSQL 方法用于执行指定的 SQL 语句，参数是要执行的 SQL 语句，返回值是整型，如果返回值大于 0，则说明 SQL 语句执行成功，否则 SQL 语句执行失败。ExecuteSQL 方法的主要代码如下所示：

```
public static int ExecuteSQL(string strSql)
{
    int result = 0;                                                  //记录受影响的行数
    SqlConnection conn = getCon();                                   //连接数据库
    conn.Open();                                                     //打开连接
    //创建 SqlCommand 对象，执行 sql 语句
    SqlCommand cmd = new SqlCommand(strSql,conn);
    result = cmd.ExecuteNonQuery();                                  //开始执行 SQL 语句并获取返回值
    conn.Close();                                                    //关闭连接
    return result;                                                   //将返回值返回给调用方
}
```

getDS 方法用于返回 DataSet，作为 GridView 控件的数据源，其参数是一条查询语句。例如，在本例后台管理界面中，通过调用该方法将所有用户的相关信息存储在 DataSet 中，作为 GridView 控件的数据源，以便管理员能够对用户进行管理，该方法的主要代码如下所示：

```
public static System.Data.DataSet getDS(string sql)
{
    System.Data.DataSet ds = null;                   //创建一个 DataSet 对象
    SqlConnection conn = getCon();                   //连接数据库
    SqlDataAdapter sda = new SqlDataAdapter(sql,conn);     //创建数据适配器
    ds = new System.Data.DataSet();                  //实例化 DataSet 对象
    sda.Fill(ds);                                    //通过数据适配器将数据填充 DataSet 中
    return ds;                                       //将填充后的 DataSet 返回给调用方
}
```

returnval 方法用于根据 ID 值查询指定字段的值，参数分别是要查询的字段名称和查询条件 ID，其返回值是所查询字段的值。在后台管理界面中，通过该方法获取用户的空间大小和用户的权限，主要代码如下所示：

```
public static string returnval(string val, int id)
{
    string mysize = string.Empty;                                    //记录字段值
    SqlConnection conn = getCon();                                   //连接数据库
    conn.Open();                                                     //打开连接
    //创建 SqlCommand 对象，查询指定字段的值
    SqlCommand cmd = new SqlCommand("select " + val + " from tb_user where ID="
    + id, conn);
    mysize = cmd.ExecuteScalar().ToString();                         //获取字段的值
    conn.Close();                                                    //关闭连接
```

```
    return mysize;                                    //将字段值返回调用方
}
```

22.5　用 户 注 册

专题讲座：光盘\MR\Video\22\开发过程.exe

➤➤➤视频速递：使读者更加直观地了解本项目的开发过程。

从本节开始，我们将介绍网络硬盘的整个开发过程。通过图文并茂的形式介绍开发过程使读者更容易理解和掌握，希望读者对本节内容能够认真阅读。

22.5.1　功能展示

如果要使用网络硬盘系统，首先要注册成为会员，登录之后才能操作网络硬盘。所以，本章的网络硬盘系统提供了用户注册的功能。在用户注册界面，用户需要填写用户名、昵称、密码和电子邮件地址，确认用户名无人使用时，才能注册成功，用户注册模块运行结果如图 22.7 所示。

22.5.2　设计思路

开发用户注册模块的过程并不复杂，首先检查用户的注册名是否可用，如果可用则使用 Insert 语句将用户填写的注册信息添加到数据库的 tb_user 表中。在页面验证中，使用了几个常用的验证控件 RequiredFieldValidator、RegularExpressionValidator、RegularExpression Validator 和 CustomValidator，分别用于验证是否输入数据、邮箱地址是否正确、两次密码输入是否一致和输入的密码是否小于 6 位。用户注册模块的流程如图 22.8 所示。

图 22.7　网络硬盘注册模块

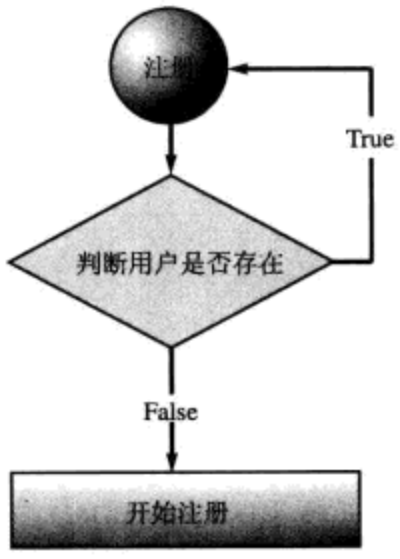


图 22.8　用户注册流程图

22.5.3　功能实现

（1）打开 Visua Studio 2008，新建一个网站。

（2）默认页为“Default.aspx”，该页用于注册新用户，在该页中用到的控件及说明如表 22.2 所示。

表 22.2　Default.aspx 页中用到的控件及说明

控　件	控件 ID	描　述
Image	无	HTML 控件，用于转到登录界面
RequiredFieldValidator	RequiredFieldValidator1	验证是否输入用户名
	RequiredFieldValidator2	验证是否输入昵称
	RequiredFieldValidator3	验证是否输入密码
	RequiredFieldValidator4	验证是否输入邮箱
	RequiredFieldValidator5	验证是否再一次输入密码
RegularExpressionValidator	RegularExpressionValidator1	验证输入的邮箱地址是否正确
CompareValidator	CompareValidator1	验证两次输入的密码是否一致
CustomValidator	CustomValidator1	验证输入的密码是否小于 6 位
ImageButton	ImgBtnRegUser	提交注册信息
	ImgBtnCheckUser	检查输入的用户名是否可用
TextBox	txtUserName	输入想要注册的用户名
	txtNickname	输入昵称
	txtPassword	输入密码
	txtPassword2	再次输入密码
	txtEmail	输入注册用户的电子邮件地址

（3）用户注册过程中，输入登录用户名后，可以单击“检查是否可用”按钮来检查输入的用户名是否已经被注册，如果已经被注册则会弹出提示信息。此功能的核心代码是调用公共类中的 CheckUser 方法来检查用户名是否可用，主要代码如下所示：

```
protected void ImgBtnCheckUser_Click(object sender, ImageClickEventArgs e)
                                                          //检查注册用户名是否存在
{
    if (txtUserName.Text.Trim().Length == 0)              //判断是否输入用户名
    {
        //如果没有输入用户名，弹出提示信息
        ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.GetType(), "",
          "alert('警告：请输入用户名！');", true);
    }
    else
    {
        //将输入的用户名传递给公共类中的 CheckUser 方法，查找数据库中是否存在这个用户名
        int flag = ComClass.CheckUser(txtUserName.Text.Trim());
        if (flag > 0)                                     //如果大于 0 说明存在该用户
        {
            //弹出提示信息
            ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
            GetType(), "",
             "alert('警告：用户名已经存在！');", true);
        }
        else//否则
        {
            //提示用户可以注册
```

```
                ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
                GetType(), "",
                 "alert('提示：恭喜您可以使用！');", true);
            }
        }
    }
```

（4）当用户输入的注册信息无误后，单击“注册”按钮进行注册。程序首先通过公共类中的 CheckUser 方法检查用户名是否可用，如果可用则调用公共类中的 RegUser 方法进行注册。注册成功后，在项目的 Net 目录下创建以用户名命名的文件夹，实现为用户分配磁盘空间，最后将注册的用户名和密码存储在 session 对象中，并直接转到 NetworkDisk.aspx 页面，进入用户网络硬盘界面，主要代码如下所示：

```
protected void ImgBtnRegUser_Click(object sender, ImageClickEventArgs e)
                                                          //用户注册
{
        //将输入的用户名传递给公共类中的 CheckUser 方法，查找数据库中是否存在这个用户名
        int flag = ComClass.CheckUser(txtUserName.Text.Trim());
        if (flag > 0)                                     //如果大于 0 说明存在该用户
        {
            ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
            GetType(), "",
            "alert('警告：用户名已经存在！');", true);      //弹出提示信息
        }
        else
        {
            string name = txtUserName.Text.Trim();        //获取输入的注册用户名
            string nickname = txtNickname.Text.Trim();    //获取输入的昵称
            //将输入的密码进行 SHA1 加密后的字符串
            string pwd =
            System.Web.Security.FormsAuthentication.HashPasswordForStoring
            InConfigFile (txtPassword2.Text.Trim(), "SHA1");
            string email = txtEmail.Text.Trim();          //输入的注册电子邮件地址
            //调用公共类中的 RegUser 方法注册
            int result = ComClass.RegUser(name, nickname, pwd, email);
            if (result > 0)                               //如果大于 0 说明注册成功
            {
                //弹出提示信息
                ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
                GetType(), "",
                "alert('提示：注册成功！');", true);
                //在项目根目录下的 Net 文件夹中为用户分配空间
                System.IO.Directory.CreateDirectory(Server.MapPath("Net/" + name));
                Session["name"] = name;                   //在 session 中存储用户名
                Session["pwd"] = pwd;                     //在 session 中存储密码
                Response.Redirect("NetworkDisk.aspx", true); //直接转向到用户界面
            }
            else
            {
                //弹出警告信息
                ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
                GetType(), "",
                 "alert('警告：注册失败！');", true);
            }
        }
}
```

为了保证用户注册信息的安全性，对用户的密码进行了 SHA1 加密。存储到数据库中的密码是经过加密后的字符串，这样可以比较有效地防止密码外泄。

22.6 用 户 登 录

通过用户登录功能，用户可以登录到后台管理界面。登录时对身份也进行了验证，登录身份分为普通用户和管理员，以便于登录成功后能够进入不同的管理界面。

22.6.1 功能展示

登录界面主要验证登录用户是否合法，就好像门卫一样，起到保驾护航的功能。为了实现这个功能，首先向项目中添加一个 Web 页，命名为 Login.aspx，实现用户或管理员登录的功能，运行结果如图 22.9 所示。

22.6.2 设计思路

用户注册成功后，便可以登录用户后台对网络硬盘进行管理。所以，本章网络硬盘系统提供了用户登录模块，它是网络硬盘系统的一道安全屏障。在该模块中主要实现的是用户输入用户名和密码，并选择用户身份后，单击“登录”按钮进行登录。实现登录功能的关键技术是 count 函数，使用该函数可以判断用户名和密码在指定数据表中是否存在，从而验证登录用户是否合法。登录模块的流程如图 22.10 所示。

图 22.9 用户登录模块

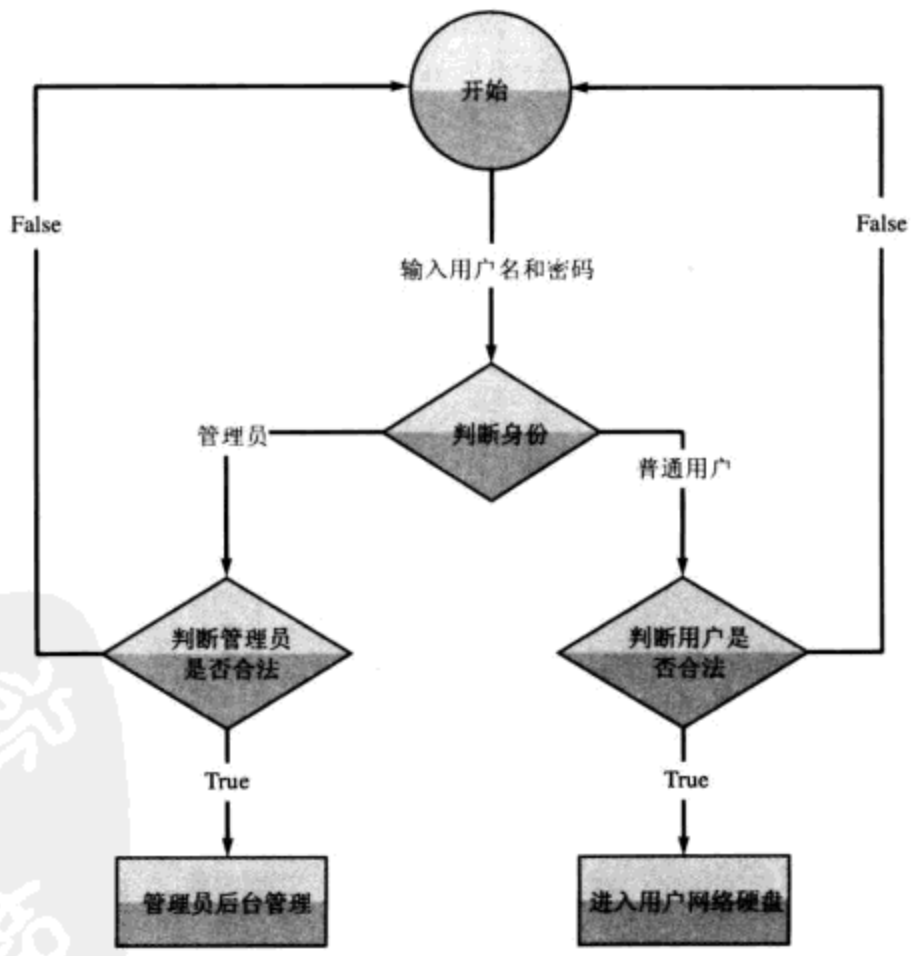


图 22.10 用户登录模块流程图

22.6.3　功能实现

用户输入用户名和密码后，单击“登录”按钮进行登录。程序首先获取用户登录的用户名和密码，然后判断用户是选择普通用户登录还是管理员登录。如果是普通用户登录只需要核对用户名和密码是否正确即可；如果是管理员登录除了需要核对用户名和密码外，还需要核对其权限是否是管理员权限。核心代码是公共类中的 ValidateUser 方法，该方法用于验证用户是否合法，如果通过该方法验证合法则页面会登录到普通用户界面或者管理员后台管理界面，主要代码如下所示：

```
protected void ImgBtnLogin_Click(object sender, ImageClickEventArgs e)//登录
{
       if (IsValid)                                    //如果页面中验证控件验证成功
       {
          string name = txtname.Text.Trim(); //获取输入的登录用户名
          string pwd =
          System.Web.Security.FormsAuthentication.HashPasswordForStoring
          InConfigFile(txtpwd.Text.Trim(), "SHA1");
                                                  //获取输入的密码并对其进行 SHA1 加密
          if (RadioButtonList1.SelectedIndex == 0)    //如果选择普通用户登录
          {
          //声明 sql 语句，验证登录的用户名和密码
         string sql = "select count(*) from tb_user where u_name='" + name + "'
         and u_pwd='" + pwd + "'";
         //将 sql 语句传递给公共类中的 ValidateUser 方法判断登录是否成功
         int flag = ComClass.ValidateUser(sql);
         if (flag > 0)                                //如果大于 0 说明登录成功
         {
             //转到登录成功界面
             Session["name"] = name;                  //session 中存储登录用户名
             Session["pwd"] = pwd;                    //session 中存储登录密码
              //获取用户在服务器上分配的空间路径
             string userDir = Server.MapPath("Net/" + name);
             if (!System.IO.Directory.Exists(userDir)) //如果该目录不存在
             {
                System.IO.Directory.CreateDirectory(userDir);
                                                          //则创建该目录
             }
             //页面转到用户管理界面
             ScriptManager.RegisterStartupScript(this.UpdatePanel1,
             this.GetType(), "", "window.location.replace('NetworkDisk.
             aspx');", true);
         }
         else
         {
             //提示用户用户名和密码错误
             ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
             GetType(), "",
             "alert('警告：用户名或密码错误！');", true);
         }
      }
      else                                            //如果选择管理员登录
      {
```

```
                //编写 sql 语句，判断数据库中是否存在该用户并且 u_identity 大于等于 1，
                //超级管理员该字段为 2，普通管理员该字段为 1
                string sql = "select count(*) from tb_user where u_name='" + name
                + "' and u_pwd
                ='" + pwd + "' and u_identity>=1";
                if (ComClass.ValidateUser(sql) > 0) //如果大于 0 说明该用户是管理员
                {
                    //转到登录成功界面
                    Session["adminname"] = name;      //session 中存储登录用户名
                    Session["adminpwd"] = pwd;        //session 中存储登录密码
                    //登录成功转向管理员界面
                    ScriptManager.RegisterStartupScript(this.UpdatePanel1,
                    this.GetType(), "", "window.location.replace('admin.aspx');",
                    true);
                }
                else
                {
                    //弹出警告信息
                    ScriptManager.RegisterStartupScript(this.UpdatePanel1, this.
                    GetType(), "",
                    "alert('警告：您不是管理员！');", true);
                }
            }
        }
    }
```

学习手册 注意 在登录界面中，可以选择普通用户登录或者管理员登录。如果选择普通用户登录，登录成功后会转向用户网络硬盘管理界面。如果是管理员登录，则登录成功后会进入管理员后台管理界面。

22.7 文 件 上 传

通过文件上传功能，用户可以将选择的文件上传到服务器中，以便于使用时可以通过网络硬盘将文件下载到计算机中。

22.7.1 功能展示

之所以称作网络硬盘，即在网络中可以存储自己的东西，所以给用户提供文件上传的功能是必须的。本章网络硬盘的上传功能是通过 jQuery 插件 uploadify 实现的，上传过程中提供上传进度条的功能，可以使用户更直观地了解上传的进度，如图 22.11 所示。

图 22.11 文件上传模块

22.7.2 设计思路

在开发文件上传模块时，具体使用什么方法上传文件首先成为作者应该考虑的问题。

如果使用 FileUpload 控件上传文件，其优点是代码少、使用简单，但是外观却差强人意。由于现在 jQuery 技术已经越来越受到关注，也因为其开发代码少、外观漂亮而征服了广大开发者，所以作者决定使用 jQuery 插件 uploadify 实现带进度条上传文件。

22.7.3　功能实现

（1）由于本章的网络硬盘系统的文件上传功能是使用 jQuery 插件 uploadify 实现的，所以此处不再介绍 jQuery 插件 uploadify 实现文件上传的代码。

（2）如果想在网络硬盘中创建文件夹，只需单击“新建”按钮，就可以打开一个模态窗口，该窗口中加载的是 CreateDirectory.aspx 页，实现创建文件夹的功能。所以在项目中新建一个 Web 页，命名为 CreateDirectory.aspx，单击“新建”按钮打开的模态窗口如图 22.12 所示。

图 22.12　新建文件夹窗口

在 CreateDirectory.aspx 页中只需添加一个 TextBox 控件和一个 ImageButton 控件，分别用于输入要创建的文件夹名称和创建文件夹的功能。当 CreateDirectory.aspx 页加载时要获取新建文件夹的上级目录，该目录是通过上传文件区域中的下拉框选择的，代码如下所示：

```
    //创建一个字符串，用于接收登录用户空间的相对路径，例如登录用户是 lvshuang，那么其空间的相对路径就是“Net/lvshuang”
    string newPath = string.Empty;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["name"] != null)//判断 Session["name"]是否为空，不为空说明用户已经登录
        {
            newPath = Request.QueryString["path"].ToString();
                                                //接收登录用户空间的相对路径
            if (newPath.StartsWith("/"))        //如果传递的相对路径以“/”开头
            {
                newPath = newPath.Remove(0, 1);
                                                //则去掉开头的“/”，防止创建目录过程中出现错误
            }
        }
        else
        {
            Response.Write("请重新登录");       //提示用户重新登录
        }
    }
```

学习手册 注意　此处注意的是要判断父窗体中传递过来的路径是否以“/”开头，如果以“/”开头则需要去掉整个斜杠，否则在创建文件夹的时候会出现路径错误。

（3）在文本框中输入要创建的文件夹名称后，单击“创建文件夹”按钮。程序首先判断是否输入文件夹名，如果没有输入则直接关闭当前的模态窗口返回父窗口。如果输入了文件夹名则通过 Directory 类的 CreateDirectory 方法在服务器端创建文件夹，代码如下所示：

```
    protected void ImgBtnCreateDir_Click(object sender, ImageClickEventArgs e)
```

```
                                                    //创建文件夹
{
    if (txtDirName.Text.Trim().Length == 0)
                                                    //判断是否输入新创建文件夹的名称，如果没有输入名称
    {
        //直接关闭当前页，返回到父窗体中
        Response.Write("<script>window.opener=null;window.close();
        </script>");
    }
    else                                                    //如果输入了文件夹名称
    {
        string path = Server.MapPath(newPath);//获取登录用户空间在服务器上的绝对路径
            //在其空间中创建文件夹，名称是输入的名称
        Directory.CreateDirectory(path + "\\" + txtDirName.Text.Trim());
            //关闭当前窗体，返回值为 1
        Response.Write("<script>window.returnValue=1;window.close();
        </script>");
    }
}
```

22.8 文 件 管 理

通过文件管理功能，用户可以对上传的文件进行相应的管理，其中包括删除、编辑、下载和复制链接地址等。

22.8.1 功能展示

文件上传完毕后，用户可以对上传的文件进行管理。在文件管理区域中显示用户网络硬盘中的所有文件夹和上传的文件，对文件夹和文件可以进行更名和删除操作，而且还提供了文件下载和复制文件下载链接地址的功能。对于文件，用户还可以进行批量删除。运行结果如图 22.13 所示。

如果想查看某个文件夹中的内容，只需单击该文件夹即可，例如在本例中单击“难忘凤凰山”即可查看该文件夹中的内容，如图 22.14 所示。

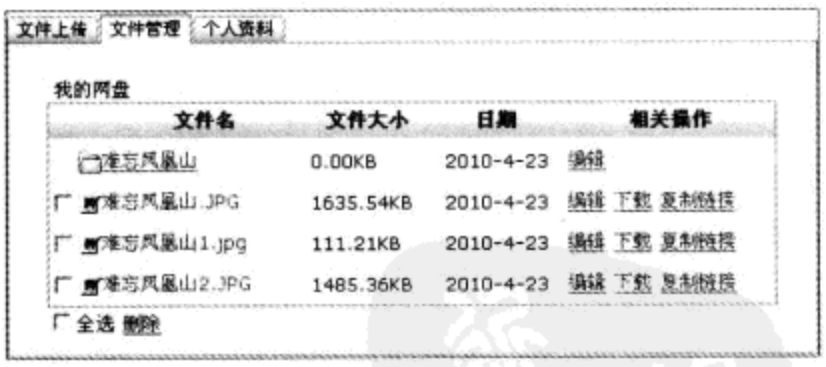


图 22.13 文件管理区域

图 22.14 查看某个文件夹中的内容

22.8.2 设计思路

在文件管理模块中，主要应该实现对文件的删除、编辑、下载和复制链接等功能。对

文件或者文件夹的操作，主要使用 File 类、FileInfo 类和 DirectoryInfo 类，在使用这些类之前要引入命名空间 System.IO。下载文件时，主要是将下载文件地址作为参数传递给一般处理程序 download.ashx，在该文件中实现从服务器中下载指定的文件。

22.8.3　功能实现

（1）在后台代码中，创建一个 getDir 方法，用于遍历用户网络硬盘中的所有文件和文件夹，并绑定到 GridView 控件中。设置文件或者文件夹的图标，以及相关操作按钮，例如，编辑、下载和复制下载链接地址等。主要代码如下所示：

```
private void getDir(string path) //获取用户磁盘空间所有文件及目录
{
        //将参数 path 中前四个字符“我的网盘”替换成该用户登录的用户名
        path = path.Remove(0, 4).Insert(0, Session["name"].ToString());
        //调用公共类中的 getDirectory 方法遍历文件夹下的所有文件夹
        string[] source = ComClass.getDirectory(Server.MapPath("Net/" + path));
        DataTable dt = new DataTable();                    //创建一个 DataTable
        //添加列名
        dt.Columns.Add("文件名");
        dt.Columns.Add("文件大小");
        dt.Columns.Add("日期");
        foreach (string s in source)                       //遍历文件夹数组
        {
            //创建一个 fso 对象
            Scripting.FileSystemObject fso = new Scripting.FileSystemObject();
            ;//获取文件夹大小
            string mysize = (Convert.ToDouble(fso.GetFolder(s).Size.ToString())/
            1024 ).ToString("0.00")
            //创建 DirectoryInfo 对象，用于获取文件夹创建的时间
            DirectoryInfo di = new DirectoryInfo(s);
            //创建一个字符串数组，存储文件夹的名称、大小和创建时间
            string[] a = {
            System.IO.Path.GetFileName(s), mysize + "KB", di.CreationTime.
            ToShortDateString() };
            dt.Rows.Add(a);                    //将数组添加到 DataTable 对象中
        }
        //如果用户空间里有文件，则获取所有文件
        string[] files = Directory.GetFiles(Server.MapPath("Net/" + path));
        foreach (string f in files)          //遍历文件数组
        {
            FileInfo fi = new FileInfo(f); //创建 FileInfo 对象，获取文件创建时间
            //创建一个字符串数组，存储文件的名称、大小和创建时间
            string[] b = { System.IO.Path.GetFileName(f),
            (Convert.ToDouble(fi.Length)/1024).ToString("0.00") + "KB", fi.
            CreationTime.ToShortDateString() };
            dt.Rows.Add(b);                    //将数组添加到 DataTable 对象中
        }
       //如果 DataTable 对象中没有数据说明当前用户空间内没有文件或者文件夹
        if (dt.Rows.Count == 0)
        {
            //为了使在没有数据的情况下 GridView 控件能够显示表头，则添加一行空字符
            dt.Rows.Add("", "", "");
```

```
        GridView1.DataSource = dt;                    //设置 GridView 控件数据源
        GridView1.DataBind();                         //绑定数据
        //遍历 GridView 控件的所有行，获取行里的控件
        foreach (Control c in GridView1.Rows[0].Controls)
        {
            c.Visible = false;                        //隐藏所有行中的控件
        }
    }
    else//如果 DataTable 对象中有数据则说明用户空间中存在文件或者文件夹
    {
        GridView1.DataSource = dt;                    //设置 GridView 控件数据源
        //设置主键字段名称为“文件名”，以便以后操作中获取文件名
        GridView1.DataKeyNames = new string[] { "文件名" };
        GridView1.DataBind();                         //绑定数据
    }
    for (int i = 0; i < GridView1.Rows.Count; i++) //遍历 GridView 控件的所有行
    {   //设置文件名左对齐
        GridView1.Rows[i].Cells[2].HorizontalAlign = HorizontalAlign.Left;
        //设置文件大小左对齐
        GridView1.Rows[i].Cells[3].HorizontalAlign = HorizontalAlign.Left;
        //获取显示图标的 Image 控件
        Image img = (Image)GridView1.Rows[i].Cells[1].FindControl("Image2");
        //获取每行中显示文件或者文件夹名称的 LinkButton 控件
        LinkButton link = (LinkButton)GridView1.Rows[i].FindControl
        ("lbtFileName");
        string filename = link.Text;          //获取 LinkButton 控件显示的文本
        //实例化 FileInfo 对象
        FileInfo fi = new FileInfo(Server.MapPath("Net/" + path + "/" +
        filename));
        if (fi.Exists)                                //如果文件存在
        {
            string[] oType = new string[] { "doc", "exe", "html", "mdb", "pdf",
            "ppt", "rar", "sql", "txt",
            "xls", "zip", "chm", "mp3", "wma" };      //声明一个类型数组
            //获取文件的扩展名
            string ftype = System.IO.Path.GetExtension(filename).Remove(0,
            1).ToLower();
            for (int k = 0; k < oType.Length; k++)    //遍历类型数组
            {
                if (oType[k] == ftype)                //如果文件的扩展名在类型数组中
                {
                    //则设置文件的图标是服务器中 file 文件夹下与其类型名相对应的图片
                    img.ImageUrl = "file/" + ftype + ".gif";
                    break;
                }
                else if (ftype == "avi" || ftype == "wmv" || ftype == "rm" || ftype
                == "rmvb" || ftype == "3gp"
                            || ftype == "mp4") //否则判断是否属于这几种视频格式
                {
                //如果是这几种视频格式，则其显示的图标是 file 文件夹下的 real.gif
                 img.ImageUrl = "file/real.gif";
                }
                else if (ftype == "jpg" || ftype == "jpeg" || ftype == "bmp" ||
                ftype == "png" || ftype == "gif") //否则判断是否属于这几种图片格式
                {
```

```
                //如果是这几种图片格式，则其显示的图标是 file 文件夹下的 pic.gif
                img.ImageUrl = "file/pic.gif";
            }
            else//如果不属于以上所有的类型
            {
                //则其显示的图标是 file 文件夹下的 unknow.gif
                img.ImageUrl = "file/unknow.gif";
            }
        }
        //如果是文件则文件名不加下画线
link.Attributes.Add("style", "text-decoration:none;");
    }
    //如果是文件夹则实例化 DirectoryInfo 对象
    DirectoryInfo mydi = new DirectoryInfo(Server.MapPath("Net/" + path
    + "/" + filename));
    if (mydi.Exists)//如果文件夹存在
    {
     img.ImageUrl = "file/dir.gif";
                                    //设置文件夹的图标是 file 文件夹下的 dir.gif
     //每行数据前的 CheckBox 控件
     CheckBox cb = (CheckBox)GridView1.Rows[i].Cells[0].FindControl
     ("cbIsCheck");
     cb.Enabled = false;            //如果是文件夹则禁用 CheckBox 控件
     cb.Visible = false;            //隐藏 CheckBox 控件
     //下载按钮
     LinkButton down = (LinkButton)GridView1.Rows[i].FindControl
     ("lbldown");
     down.Visible = false;          //如果是文件夹则隐藏“下载”按钮
     //复制链接按钮
     LinkButton addr = (LinkButton)GridView1.Rows[i].FindControl
     ("lbladdr");
     addr.Visible = false;          //如果是文件夹则隐藏“复制链接”按钮
        }
    }
}
```

学习手册 注意　当 GridView 控件中没有数据时，该控件的表头会变得不可见。为了解决这个问题，在本例中判断是否存在数据，如果不存在数据则向 GridView 控件中添加一行空数据以防止该控件的表头不可见。

（2）如果文件或者文件夹过多且不进行分页，则 GridView 控件会同时显示所有数据，显然这样浏览起来是非常烦琐的。GridView 控件提供了分页的功能，这样就可以非常方便地将大量数据进行分页显示，效果如图 22.15 所示。

图 22.15　文件管理区域中数据分页显示

当单击分页页码时，会触发 GridView 控件的 PageIndexChanging 事件，在该事件中重新指定当前页的索引，调用 getDir 方法重新获取用户磁盘空间下的所有文件及目录，在 getDir 方法中会对 GridView 控件重新进行数据

绑定，代码如下所示：

```
    protected void GridView1_PageIndexChanging(object sender, GridViewPage
EventArgs e)                                          //分页
    {
        GridView1.PageIndex = e.NewPageIndex; //重新设置当前索引
        getDir(lblSelectPath.Text);    //调用 getDir 方法获取用户磁盘空间所有文件及目录
        cbSelectAll.Checked = false;                  //取消“全选”
        UpdatePanel4.Update();                        //提交服务器
    }
```

当选中“全选”复选框后，每一个文件前边的复选框会变成选中状态。反之，则取消文件的选中状态。这样，就为删除所有文件提供方便。否则，如果想删除所有文件，还要一个个地选中文件，这样很显然是不合理的，“全选”复选框的 CheckedChanged 事件如下所示：

```
    protected void cbSelectAll_CheckedChanged(object sender, EventArgs e)
                                                          //“全选”复选框
    {
        for (int i = 0; i < GridView1.Rows.Count; i++)    //遍历 GridView 控件所有行
        {
            if (cbSelectAll.Checked == true)              //如果“全选”复选框被选中
            {
                //每一行最前端的 CheckBox 控件
                CheckBox cb = (CheckBox)GridView1.Rows[i].Cells[0].FindControl
                ("cbIsCheck");
                cb.Checked = true;                        //设置为选中
            }
            else
            {
                //每一行最前端的 CheckBox 控件
                CheckBox cb = (CheckBox)GridView1.Rows[i].Cells[0].FindControl
                ("cbIsCheck");
                cb.Checked = false;                       //设置取消选中
            }
        }
    }
```

制作“全选”复选框时，要将其 AutoPostBack 属性设为 True。这样，当更改复选框的选择状态时，会自动回发服务器，触发 CheckedChanged 事件。

当选中“全选”复选框，或者单独选择了文件前边的复选框后，单击“删除”按钮可以删除选中的文件。在该按钮的 Click 事件中获取 Label 控件 lblSelectPath 显示的路径，然后再获取每一行文件的名称，这样就获取到要删除文件的相对路径了，最后通过 FileInfo 对象的 Delete 方法删除选中的文件，代码如下所示：

```
    //“全选”复选框后边的“删除”按钮
    protected void lbtDelete_Click(object sender, EventArgs e)
    {
        FileInfo fi;                                      //创建 FileInfo 对象
        for (int i = 0; i < GridView1.Rows.Count; i++)    //遍历 GridView 控件所有行
        {
```

```
            //显示文件或者文件夹名称的 LinkButton 控件
            LinkButton link = (LinkButton)GridView1.Rows[i].Cells[2].FindControl
            ("lbtFileName");
            string filename = link.Text;                          //获取文件或者文件夹的名称
            //实例化 FileInfo 对象
            fi = new FileInfo(Server.MapPath("Net/"
              + lblSelectPath.Text.Remove(0, 4).Insert(0, Session["name"].ToString()) +
              "/" + filename));
            if (fi.Exists)                                        //判断文件是否存在
            {
                //每行数据最前端的 CheckBox 复选框
                CheckBox cb = (CheckBox)GridView1.Rows[i].Cells[0].FindControl
               ("cbIsCheck");
                if (cb.Checked == true)                           //如果复选框处于选中状态
                {
                    fi.Delete();                                  //则删除选中状态的文件
                }
            }
        }
        getDir(lblSelectPath.Text);  //调用 getDir 方法获取用户磁盘空间所有文件及目录
        this.cbSelectAll.Checked = false;                  //取消“全选”复选框的选中状态
    }
```

学习手册 注意　删除文件或者文件夹后，一定要调用 getDir 方法重新遍历用户空间中的所有文件或文件夹，实现即时刷新的效果。

（3）我们在 GridView 的模板列中添加了四个 LinkButton 控件，分别实现“显示文件或文件夹”、“编辑”、“下载”和“复制链接地址”的功能，那么我们如何实现单击不同的按钮实现不同的功能呢？此处我们设置这四个 LinkButton 控件的 CommandName 属性分别为“look”、“myedit”、“down”和“addr”，这样，在 GridView 控件的 RowCommand 事件中就可以根据传递的 CommandName 属性进行相应的操作，代码如下所示：

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    GridViewRow gvr;                                      //控件所在的行
    int index = 0;                                        //控件所在行的索引
    LinkButton llb;                                       //创建一个 LinkButton 对象
    string sName;                                         //文件或者文件夹的名称
    string newPath = string.Empty;                        //存储文件或文件夹路径
     //将 newPath 字符串中“我的网盘”替换成登录用户名后的路径
    string newPath2 = string.Empty; ;
     //当单击文件或者文件夹的名称时如果其 CommandName 不等于“Page”
    if (e.CommandName != "Page")
    {
        //文件或文件夹所在的行
        gvr = (GridViewRow)(((LinkButton)(e.CommandSource)).NamingContainer);
        index = gvr.RowIndex;                             //所在行的索引
        //如果该行不是表头
        if (GridView1.Rows[index].RowType != DataControlRowType.Header)
        {
            //显示文件或文件夹名称的 LinkButton 控件
            llb = (LinkButton)GridView1.Rows[index].FindControl("lbtFileName");
            sName = llb.Text;                             //获取显示的文件或者文件夹名称
```

```
            newPath = lblSelectPath.Text + "/" + sName; //获取文件或者文件夹的路径
            //将 newPath 字符串中“我的网盘”替换成登录用户名后的路径
            newPath2 = lblSelectPath.Text.Remove(0, 4).Insert(0, Session["name"].
            ToString()) + "/" + sName;
            }
}
switch (e.CommandName)                          //判断用户选择的操作
{
 case "look"://如果 CommandName 为 look 说明用户想浏览文件夹
 //判断用户选择的文件夹是否存在
 if (Directory.Exists(Server.MapPath("Net/" + newPath2)))
  {
     lblSelectPath.Text = newPath;              //显示用户选择的文件夹
     //调用 getDir 方法获取用户磁盘空间所有文件及目录
     getDir(lblSelectPath.Text);
     lbtBack.Visible = true;                    //显示“返回上一级”按钮
  }
  break;
 case "down"://如果选择“下载”
 //判断要下载的文件是否存在
 if (File.Exists(Server.MapPath("Net/" + newPath2)))
  {
     //对文件路径进行 UTF8 编码
     string filepath = HttpUtility.UrlEncode(newPath2, System.Text.
     Encoding.UTF8);
     //将路径传递给 download.ashx 进行下载
     Response.Redirect("download.ashx?p=" + filepath);
  }
     break;
     case "addr"://如果选择“复制下载地址”
     //对文件路径进行 UTF8 编码
     string addrs = HttpUtility.UrlEncode(newPath2, System.Text.Encoding.
     UTF8);
     //获取选择文件的下载链接地址
     addrs = "http://" + Request.Url.Host + "/download.ashx?p=" + addrs;
     //将文件下载链接地址复制到系统剪切板中
     ScriptManager.RegisterClientScriptBlock(this.UpdatePanel4,
this.GetType(), "", "window.clipboardData.setData('Text','" + addrs + "');
alert('链接地址复制成功! ');", true);
     break;
      case "myedit"://如果选择“编辑”
      //如果选择的文件或者文件夹存在
      if (Directory.Exists(Server.MapPath("Net/" + newPath2)) || File.Exists
      (Server.MapPath("Net/"
       + newPath2)))
      {
         //编写 JavaScript 脚本，打开模态窗口，显示编辑文件或者文件夹的界面
         string myscript = "var val = window.showModalDialog('SetDirOrFile.
         aspx?path="
         + HttpUtility .UrlEncode(newPath2,System.Text.Encoding.UTF8)+ "',
         'window', 'dialogWidth=290px;dialogHeight=150px;status=no;help=no;
          scrollbars=no');";
          myscript = myscript + "if (val == 1) {";
          myscript = myscript + "document.getElementById('btnRefurbish').
          click();";
```

```
                myscript = myscript + "}";
                myscript = myscript + "else";
                myscript = myscript + "{";
                myscript = myscript + "if(val==2)";
                myscript = myscript + "{";
                myscript = myscript + "alert('操作失败，请重试！');";
                myscript = myscript + "document.getElementById('btnRefurbish').
                click();";
                myscript = myscript + "}";
                myscript = myscript + "}";
                //在客户端注册编写的 JavaScript 脚本
                ScriptManager.RegisterClientScriptBlock(this.UpdatePanel4,this.
                GetType(), "", myscript, true);
               }
               break;
        }
        UpdatePanel4.Update();//提交服务器
    }
```

模态窗口是指处于最前端的窗口，当打开模态窗口时，不能操作其他窗口。只有关闭模态窗口后，才允许进行其他操作。

（4）当用户希望下载某个文件时，就可以单击“下载”按钮。之后程序获取该文件的路径，进行编码后传递给 download.ashx 文件进行下载。所以，在项目中需要添加一个一般处理程序，并命名为 download.ashx，实现文件下载的功能，该文件代码如下所示：

```
<%@ WebHandler Language="C#" Class="download" %>
using System;
using System.Web;
public class download : IHttpHandler{
public void ProcessRequest (HttpContext context) {
try
{
   //获取要下载的文件路径
   string strs = "Net/"
    + HttpUtility.UrlDecode(context.Request.QueryString["p"].ToString(), System.
    Text.Encoding.UTF8);
   string path = context.Server.MapPath(strs);     //转换成服务器绝对路径
   context.Response.Clear();                       //清除缓冲器内容
   context.Response.Charset = "utf-8";             //设置输出流的编码方式
   context.Response.Buffer = true;                 //是否处理完之后发送数据
   context.Response.ContentEncoding = System.Text.Encoding.UTF8;
   //对要下载的文件路径进行 UTF8 编码，方式下载时文件名是乱码
   string str =
  System.Web.HttpUtility.UrlEncode(System.IO.Path.GetFileName(path), System.Text.
  Encoding.UTF8);
   context.Response.AppendHeader("Content-Disposition", "attachment;filename="
   + str);                                                    //添加 HTTP 头
   context.Response.ContentType = "application/unknow";      /设置 MIME 类型
   context.Response.WriteFile(path);                         /将文件写入 HTTP 流
   context.Response.Flush();                                 //向客户端发送流
   context.Response.Close();                                 //关闭流
   context.Response.End();                                   //停止该页执行
```

```
    }
    catch
    {
        context.Response.Write("出现错误！");
    }
}
public bool IsReusable {
  get {
        return false;
      }
}
}
```

当用户单击“下载”按钮时，浏览器会阻止弹出窗口。用户只需要允许浏览器加载弹出窗口即可，否则不能下载文件。

（5）如果想编辑文件或者文件夹，可以单击“编辑”按钮。此时，程序会获取选择的文件或者文件夹的路径，作为参数传递给打开的模态窗口。在模态窗口中，加载的是 SetDirOrFile.aspx 页，在该页中可以对文件或者文件夹进行删除和更名等操作。所以，需要在项目中新建一个 Web 页，命名为 SetDirOrFile.aspx，在该页中添加一个 TextBox 控件用于显示或者输入新的文件或文件夹名称，再添加三个 ImageButton 控件，分别用于修改、取消和删除操作。单击“编辑”按钮打开的模态窗口如图 22.16 所示。

图 22.16　编辑文件或文件夹窗口

在模态窗口加载的 SetDirOrFile.aspx 页中，首先要获取传递过来的路径，然后将路径中的文件或文件夹的名称显示到文本框中，判断传递过来的是文件夹还是文件，并设置相应的标题，例如：传递过来的是文件夹的路径，则设置标题为“编辑文件夹”，代码如下所示：

```
    //创建一个字符串变量，用于接收父窗体中传递过来的要操作的文件夹或文件路径
    string dirname =HttpUtility.UrlDecode(HttpContext.Current.Request.QueryString
["path"].ToString(),
    System.Text.Encoding.UTF8);
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)                                    //如果页面第一次加载
        {
            if (Session["name"] != null)//如果 Session["name"]不为空，说明用户已经登录
            {
               //当前页面文本框中显示要操作文件夹的名称
                txtName.Text = dirname.Substring(dirname.LastIndexOf("/") + 1);
                if (Directory.Exists(Server.MapPath("Net/" + dirname)))
                                                            //判断接收的路径是否是文件夹
                {
                //如果是文件夹，则该窗体显示的标题是“编辑文件夹”
                    this.Title = "编辑文件夹";
                }
                if (File.Exists(Server.MapPath("Net/" + dirname)))//如果接收的路径是文件
                {
                    this.Title = "编辑文件";                //则该窗体显示的标题是“编辑文件”
```

```
            }
        }
    }
}
```

学习手册 注意 由于父窗体中传递过来的文件夹或者文件路径是经过 UTF8 编码的，所以在编辑窗口中要对路径进行相应的解码，这样做的目的也是为了防止中文名称出现乱码的问题。

在模态窗口中，如果希望修改文件或者文件夹的名称，可以在文本框中输入新的名称后，单击“修改”按钮。如果修改文件夹名称，则程序会通过 DirectoryInfo 对象的 MoveTo 方法实现。如果修改文件名称，则程序会通过 FileInfo 对象的 MoveTo 方法实现，代码如下所示：

```
protected void ImgBtnEdit_Click(object sender, ImageClickEventArgs e)//修改
{
  try
    {
        if (txtName.Text.Length != 0)       //如果文本框中存在数据
        {
            //判断用户是否输入新的文件或文件夹名称，如果没有输入新的名称
            if (txtName.Text.Trim() == dirname.Substring(dirname.LastIndexOf
            ("/") + 1))
            {
                //则直接关闭当前窗口，返回值为 1
                Response.Write("<script>window.returnValue=1;window.close();
                </script>");
            }
            else                                        //如果输入了新的名称
            {
                if (Directory.Exists(Server.MapPath("Net/" + dirname)))
                                                        //如果此时是操作文件夹
                {
                    //获取文件夹更名后的路径
                    string newName = "Net/" + dirname.Remove(dirname.
                    LastIndexOf("/") + 1)
                    + txtName.Text.Trim();
                    //实例化 DirectoryInfo 对象
                    DirectoryInfo di = new DirectoryInfo(Server.MapPath("Net/"
                    + dirname));
                     //使用该对象的 MoveTo 方法修改文件夹名称
                    di.MoveTo(Server.MapPath(newName));
                     //关闭当前窗口，返回值为 1
                    Response.Write("<script>window.returnValue=1;window.
                    close();</script>");
                }
                if (File.Exists(Server.MapPath("Net/" + dirname)))
                                                        //如果此时是操作文件
                {
                    //获取文件更名后的路径
                    string newName = "Net/" + dirname.Remove(dirname.
                    LastIndexOf("/") + 1)
                     + txtName.Text.Trim();
                    //实例化 FileInfo 对象
```

```
                    FileInfo fi = new FileInfo(Server.MapPath("Net/" + dirname));
                    //使用该对象的 MoveTo 方法修改文件名称
                    fi.MoveTo(Server.MapPath(newName));
                    //关闭当前窗口，返回值为 1
                    Response.Write("<script>window.returnValue=1;window.
                    close();</script>");
                }
            }
        }
    }
    catch
    {
        Response.Write("<script>window.returnValue=2;window.close();
        </script>");
    }
}
```

（6）如果希望在模态窗口中删除文件或者文件夹，只需单击“删除”按钮即可。该按钮的 Click 事件会根据删除的类型进行相应的删除操作。例如：如果要删除文件夹，则程序会调用 DirectoryInfo 对象的 Delete 方法进行删除。如果要删除文件，则调用 FileInfo 对象的 Delete 方法进行删除，代码如下所示：

```
protected void ImgBtnDelete_Click(object sender, ImageClickEventArgs e)
                                                                    //删除
{
    try
    {
        if (Directory.Exists(Server.MapPath("Net/" + dirname))) //如果是文件夹
        {
            string newName = "Net/" + dirname;                          //获取文件夹路径
            //实例化 DirectoryInfo 对象
            DirectoryInfo di = new DirectoryInfo(Server.MapPath(newName));
            di.Delete(true);                            //调用该对象的 Delete 方法删除文件夹
            //关闭窗口，返回值为 1
            Response.Write("<script>window.returnValue=1;window.close();
            </script>");
        }
        if (File.Exists(Server.MapPath("Net/" + dirname)))      //如果是文件
        {
            string newName = "Net/" + dirname;                          //获取文件路径
            FileInfo fi = new FileInfo(Server.MapPath(newName))
                                                    //实例化 FileInfo 对象
            fi.Delete();                            //调用该对象的 Delete 方法删除文件
            //关闭窗口，返回值为 1
            Response.Write("<script>window.returnValue=1;window.close();
            </script>");
        }
    }
    catch
    {
        Response.Write("<script>window.returnValue=2;window.close();</script>");
                                                    //关闭窗口
    }
}
```

（7）当 GridView 控件中加载数据后，随着鼠标在控件中移动，鼠标所处的行会变色，这样做使得浏览起来感觉非常美观，而且控件的表头还可以自定义背景或设置每一行的高度，这些功能都是在控件的 RowDataBound 事件中实现的，代码如下所示：

```
//实现行背景跟随鼠标变化
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)        //如果是数据绑定行
    {
        //鼠标经过时，行背景色变
        e.Row.Attributes.Add("onmouseover", "this.style.backgroundColor=
        '#E6F5FA'");
        //鼠标移出时，行背景色变
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor=
        '#FFFFFF'");
        e.Row.Attributes.Add("style", "height:28px");        //设置行高度
    }
    if (e.Row.RowType == DataControlRowType.Header)          //如果是表头
    {
        //设置表头的背景
        e.Row.Attributes.Add("style", "background-image:url('images/ht_10.
        gif');");
    }
}
```

（8）当用户网络硬盘中存在文件夹时，我们在文件管理区域中会显示文件夹，如果想浏览该文件夹中的内容，只需单击文件夹即可，浏览文件夹之后，如果想返回上一级，可以单击“返回上一级”按钮，这样就可以返回上一级目录，浏览上一级目录中的内容，“返回上一级”的功能是通过 LinkButton 控件实现的，每单击一次该按钮，就在当前路径中去掉当前文件夹的名称，然后将处理后的路径传递给 getDir 方法并重新绑定 GridView 控件，代码如下所示：

```
//进入某个文件夹后的“返回上一级”按钮
protected void lbtBack_Click(object sender, EventArgs e)
{
    if (lblSelectPath.Text != "我的网盘")                    //如果当前目录不在根目录下
    {
        string path = lblSelectPath.Text;                    //获取当前目录相对路径
        path = path.Remove(path.LastIndexOf("/"));//去掉路径末尾的“/”
        lblSelectPath.Text = path;                           //重新显示符合规格的路径
        getDir(path);              //调用 getDir 方法获取用户磁盘空间所有文件及目录
        if (path != "我的网盘")                              //如果当前不处于根目录
        {
            lbtBack.Visible = true;                          //则显示“返回上一级”按钮
        }
        else
        {
            lbtBack.Visible = false;                         //隐藏“返回上一级”按钮
        }
    }
    else                                                     //如果当前目录处于根目录下
    {
        //调用 getDir 方法获取用户磁盘空间根目录下所有文件及目录
```

```
            getDir(lblSelectPath.Text);
            lbtBack.Visible = false;                    //隐藏“返回上一级”按钮
        }
        this.cbSelectAll.Checked = false;               //取消“全选”复选框的选中状态
    }
```

当用户单击某个文件夹，查看文件夹中的内容时，系统会提供一个“返回上一级”的按钮，这样浏览完毕后，单击此按钮可以返回上一级目录。这个按钮在浏览根目录时是隐藏的，只有进入根目录的其他文件夹时才能显示出来。

（9）在文件管理区域中，我们添加了一个 Timer 控件。这样做的目的是，当管理员在后台修改用户的空间大小后，使用 Timer 控件定期检查数据库，如果发现用户空间被更改，则弹出 Popup 窗口提示用户修改后的空间大小，Timer 控件的 Tick 事件如下所示：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    if (Session["name"] != null)        //如果用户处在登录状态
    {
        //获取该用户在数据库中 u_flag 字段的值，默认为 0，如果为 1 说明管理员已经对用户的空间做了调整
        string flag = ComClass.returnValues("u_flag", Session["name"].ToString());
        if (flag == "1")                //如果该值为 1，则管理员对用户空间大小做了调整
        {
            Timer1.Enabled = false;     //停止 Timer 控件
            //编写 JavaScript 脚本，弹出右下角的 Popup 提示窗口
            string myscript = "$.messager.show('系统提示', '您的空间已设置为："
            + ComClass.returnValues("u_disk", Session["name"].ToString()) + "M',
            0);";
            //将编写的 JavaScript 脚本注册到客户端页面上执行
            ScriptManager.RegisterStartupScript(UpdatePanel4, this.GetType(), "",
            myscript, true);
            //编写 sql 语句，重新将该用户数据库中的 u_flag 字段设为 0
            string sql = "update tb_user set u_flag=0 where u_name='" + Session
            ["name"].TcString() + "'";
                //调用公共类中的 ExecuteSQL 方法执行 sql 语句，如果返回值大于 0 说明执行成功
            if (ComClass.ExecuteSQL(sql) > 0)
            {
                btnRefurbish_Click(sender, e);          //重新刷新所有数据
                Timer1.Enabled = true;                  //启动 Timer 控件
            }
        }
    }
}
```

当管理员修改用户的空间大小后，需要提醒用户。本例中作者使用 jQuery 插件 messager 弹出 Popup 窗口提示用户空间被管理员修改，并且显示修改后的大小。

22.9 个 人 资 料

用户登录成功后，还可以对自己的个人资料进行管理。管理的项目包括修改昵称、密

码和设置用户头像等。

22.9.1　功能展示

该模块用于显示当前登录用户的用户名、邮箱地址和昵称，并且可以修改昵称和密码，还可以修改自己的个性头像，用户可以选择网络硬盘系统自带的头像，也可以将网络图片设置成头像，模块运行结果如图 22.17 所示。

图 22.17　个人资料区域

22.9.2　设计思路

在本例的网络硬盘系统中，个人资料模块主要应该实现对基本信息和头像进行设置。在基本信息中应该可以修改用户的昵称、密码。在修改头像时可以选择网络硬盘系统提供的头像，也可以自己指定一个网络中的图片作为头像。修改昵称和密码时，主要是使用 Update 语句实现的。而更换头像时，有预览头像的区域，该区域主要是通过更改 Image 控件的 src 属性实现的，使用户能够即时看到自己修改后的头像。

22.9.3　功能实现

（1）首先在 HTML 代码的<head></head>区域中通过 JavaScript 脚本创建两个函数 selectImg 和 selectUrl，分别用于预览用户选择的系统自带的头像或用户输入的网络图片，代码如下所示：

```
function selectImg() {
    var obj = document.getElementById("TabContainer1_TabPanel3_DropDownList1");
    var selectStr = obj.options[obj.selectedIndex].value;
    document.getElementById("Image1").src = selectStr;
}
function selectUrl() {
    var val = document.getElementById("TabContainer1_TabPanel3_txtUrl").value;
    document.getElementById("Image1").src = val;
}
```

（2）当用户登录成功，进入 NetworkDisk.aspx 页时，程序就会将用户的相关信息都检索出来显示到前台页面上，其中包括个人资料区域中的登录用户名称和登录用户注册的邮箱地址以及用户的昵称，代码如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
        if (Session["name"] == null)              //如果 Session["name"]等于 null
        {
            Response.Redirect("Login.aspx", true);//说明用户没有登录，则转向登录页面
        }
```

```
else
{
    if (!IsPostBack)                                    //如果页面第一次加载
    {
        //设置标题
        this.Title = Session["name"].ToString() + "登录成功";
        setUserInfo();              //调用 setUserInfo 方法获取用户的基本信息
        //显示用户头像
        Image1.ImageUrl = ComClass.returnValues("u_img", Session["name"].
        ToString());
        ddlDir.Items.Clear();                   //清空显示所有目录及子目录的下拉框
        //设置下拉框的第一项
        ddlDir.Items.Add(new ListItem("我的网盘", "Net/" + Session["name"].
        ToString()));
        //获取所有目录
        getalldir(Server.MapPath("Net/" + Session["name"].ToString()), 1);
        getDir(lblSelectPath.Text);         //获取所有文件及目录
        //显示登录用户名
        this.lblNameuser.Text = Session["name"].ToString();
        //显示注册的电子邮件
        lblEmail.Text = ComClass.returnValues("u_email", Session["name"].
        ToString());
        //显示昵称
        txtNickName.Text = ComClass.returnValues("u_nickname", Session
        ["name"].ToString());
        //获取系统默认的所有头像
        string[] face = Directory.GetFiles(Server.MapPath("face"));
        //为选择头像的下拉框添加第一项
        DropDownList1.Items.Add(new ListItem("选择默认头像", ""));
        foreach (string pic in face)                    //遍历所有头像数组
        {
            if (Path.GetExtension(pic) != ".db")        //如果扩展名不等于 db
            {
                //将数组中的所有头像信息添加到选择头像的下拉框中
                DropDownList1.Items.Add(new ListItem(Path.GetFileName
                (pic), "face/"
                    + Path.GetFileName(pic)));
            }
        }
    }
}
```

用户登录成功之后，程序会获取信息显示区域、文件上传区域、文件管理区域和个人资料区域的相关信息，以便用户能够了解初始状态或上一次操作的结果。

如果用户想修改昵称，就可以输入新的昵称后，单击“修改”按钮。此时，程序首先获取输入的昵称，然后编写 update 语句更新数据库中的 u_nickname 字段，最后调用公共类中的 ExecuteSQL 方法执行 update 语句，完成对昵称的修改，代码如下所示：

```
protected void ImgBtnChangeNickName_Click(object sender, ImageClickEventArgs e)
                                                                    //修改昵称
{
```

```
if (txtNickName.Text != "")                                    //如果输入了昵称
{
    string newNickname = txtNickName.Text.Trim();              //获取输入的昵称
    //编写 SQL 语句更新数据库中存储用户昵称的字段
    string sql = "update tb_user set u_nickname='" + newNickname + "' where
    u_name='"
      + Session["name"].ToString() + "'";
   //调用公共类中的 ExecuteSQL 方法执行 SQL 语句，如果返回值大于 0 说明执行成功
    if (ComClass.ExecuteSQL(sql) > 0)
    {
        //弹出提示信息
        ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
           "alert('昵称修改成功');", true);
        //显示修改后的昵称
        txtNickName.Text = ComClass.returnValues("u_nickname", Session
       ["name"].ToString());
    }
   else                                                        //否则，说明执行 SQL 语句失败
    {
        //弹出提示信息
        ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
           "alert('昵称修改失败');", true);
    }
}
else                                                           //如果没有输入昵称
{
    //弹出提示信息，提示用户输入昵称
    ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
    "alert('请输入昵称');", true);
    }
    UpdatePanel6.Update();//调用 UpdatePanel6 的 Update 方法提交给服务器实现局部刷新
}
```

如果用户想修改密码，就可以输入新的密码后，单击“修改”按钮。此时，程序首先获取输入的密码并进行 SHA1 加密，然后编写 update 语句更新数据库中的 u_pwd 字段，最后调用公共类中的 ExecuteSQL 方法执行 update 语句，完成对密码的修改，代码如下所示：

```
protected void ImgBtnChangePassWord_Click(object sender, ImageClickEventArgs e)
                                                               //修改密码按钮
{
    if (txtpassword.Text != "")                                //如果输入了密码
    {
        string newNickname = txtpassword.Text.Trim();          //获取输入的密码
        if (newNickname.Length < 6)                            //如果密码长度小于 6
        {
            //提示用户密码至少 6 位
            ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
               "alert('密码至少 6 位');", true);
            return;
        }
        //编写 SQL 语句，更新用户在数据库中存储用户密码的字段
        string sql = "update tb_user set u_pwd='"
        +System.Web.Security.FormsAuthentication.HashPasswordForStoring-
        InConfigFile(newNickname,
```

```
            "SHA1") + "' where u_name='" + Session["name"].ToString() + "'";
            //调用公共类中的 ExecuteSQL 方法执行 SQL 语句，如果返回值大于 0 说明执行成功
            if (ComClass.ExecuteSQL(sql) > 0)
            {
                //弹出提示信息
                ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
                  "alert('密码修改成功');", true);
            }
            else                                                    //否则，说明执行 SQL 语句失败
            {
                //弹出提示信息
                ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
                   "alert('密码修改失败');", true);
            }
        }
        else                                                        //如果没有输入密码
        {
            //提示用户输入密码
            ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
            "alert('请输入密码');", true);
        }
        UpdatePanel6.Update();//调用 UpdatePanel6 的 Update 方法提交给服务器实现局部刷新
    }
```

用户输入新密码后，要对输入的密码进行 SHA1 加密，然后才能将加密后的字符串更新到数据库中，实现密码修改。

修改个人头像，主要是修改数据库中的 u_img 字段。网络硬盘程序提供了两种设置头像的方法：一种是选择网络硬盘默认的头像作为头像；另一种是设置网络图片作为头像。所以程序首先判断选择哪种方式修改头像。如果选择系统自带的头像，则获取下拉框中选择的头像的路径；如果选择网络图片作为头像，则获取输入的网络图片的地址。最后编写 update 语句更新数据库中的 u_img 字段，调用公共类中的 ExecuteSQL 方法执行编写的 update 语句，代码如下所示：

```
protected void ImgBtnChangeImage_Click(object sender, ImageClickEventArgs e)
                                                            //修改头像
{
    string userImgPath = string.Empty;                      //创建字符串，存储用户头像路径
    if (RadioButton1.Checked)                               //如果选择系统自带的头像
    {
        if (DropDownList1.SelectedItem.Value != "")         //如果下拉框中选择了头像
        {
            userImgPath = DropDownList1.SelectedItem.Value;     //获取头像路径
        }
        else
        {
            //弹出提示信息
            ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
             "alert('请选择头像');", true);
                return;
            }
        }
```

```
if (RadioButton2.Checked)                          //如果选择使用网络图片设为头像
{
    if (txtUrl.Text.Trim() != "")                  //如果文本框中输入了网络图片地址
    {
        userImgPath = txtUrl.Text.Trim();          //获取输入的网络图片地址
    }
    else
    {
        //弹出提示信息
        ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
          "alert('请输入网络头像地址');", true);
            return;
         }
     }
     //编写 sql 语句，用于更新用户数据库中的存储头像路径的字段
     string sql = "update tb_user set u_img='" + userImgPath
         + "' where u_name='" + Session["name"].ToString() + "'";
     //调用公共类中的 ExecuteSQL 方法执行 SQL 语句，如果返回值大于 0 说明执行成功
 if (ComClass.ExecuteSQL(sql) > 0)
 {
     Image1.ImageUrl = userImgPath;                //在用户界面中显示新设置的头像
     //弹出设置成功的提示信息
     ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
     "alert('头像设置成功');", true);
 }
     else                                          //否则，说明执行 SQL 语句失败
     {
         //弹出提示设置失败的信息
         ScriptManager.RegisterStartupScript(UpdatePanel6, this.GetType(), "",
         "alert('头像设置失败');", true);
         return;
     }
     UpdatePanel6.Update();//调用 UpdatePanel6 的 Update 方法提交给服务器实现局部刷新
}
```

22.10　本 章 小 结

本章主要讲解了网络硬盘的开发，在开发网络硬盘时主要是对文件和文件夹进行操作，例如创建文件及文件夹功能。另外，还实现了文件的上传下载功能，文件的操作在 ASP.NET 中有着举足轻重的作用，希望读者朋友们好好研读本章内容，做到学以致用，融会贯通。

附录 A

专业术语表

第 1 章　走进 ASP.NET

ASP.NET

ASP.NET 是一种开发动态网站的技术。它是.NET 框架的一部分，可以使用任何.NET 兼容的语言（如 Visual Basic .NET、C#、J#等）来编写 ASP.NET 网站。

动态网站

所谓的动态网站就是网站中网页网址的后缀名不是.htm、.html、.shtml、.xml 等静态网页的常见形式，而是以 aspx、asp、.jsp、.php 等形式为后缀，动态网页可以与数据库交互，并且在动态网页网址中可以使用“?”传递参数，静态网页是相对于动态网页而言，是指没有后台数据库、不含程序和不可交互的网页。编的是什么它显示的就是什么，不会有任何改变。

静态网站

静态网页相对更新起来比较麻烦，适用于一般更新较少的展示型网站。

.NET Framework

.NET Framework 是微软公司推出的完全面向对象的软件开发与运行平台。.NET Framework 具有两个主要组件：公共语言运行时（Common Language Runtime，CLR）和类库。

公共语言运行时

公共语言运行时（CLR）负责管理和执行由.NET 编译器编译产生的中间语言代码。

由于公共语言运行库的存在，解决了很多传统编译语言的致命缺点，如垃圾内存回收、安全性检查等。

类库

类库就好比一个大仓库里装满了工具。类库里有很多现成的类，可以拿来直接使用。例如，文件操作时，可以直接使用类库里的 IO 类。

中间语言代码

所谓的中间语言代码是一种类似于汇编的程序语言，起到中介或翻译的作用。例如，有汉语、日语和法语三位学者，他们正在交流一个问题，他们怎么交流呢？最后，他们选择用英语进行交流，因为英语是世界上普及较广的语言。这样，他们彼此才能明白对方说的是什么。此处的英语就相当于中间语言，任何语言都会被编译成中间语言。

Visual C#

Visual C#是微软公司推出的一种语法简洁、类型安全的面向对象的编程语言，开发人员可以通过它编写在.NET Framework 上运行的各种安全可靠的应用程序，使用 Visual C# 可以创建很多程序。

第 2 章　构建 ASP.NET 开发环境

IIS（信息服务管理器）

IIS 是 Internet Information Server 的缩写，是微软公司主推的 Web 服务器。IIS 提供了最简捷的方式来共享信息、建立并部署企业应用程序，以及建立和管理 Web 上的网站。通过 IIS，用户可以轻松地测试、发布、应用和管理自己的 Web 页和 Web 站点。

Visual Studio 2008

Visual Studio 2008 是微软为了配合.NET 战略推出的开发工具，可以开发 ASP.NET 2.0、ASP.NET 3.0、ASP.NET 3.5 网站程序。

MSDN

MSDN 帮助文档是 Microsoft Visual Studio 2008 自带的一种帮助，开发人员在编写应用程序时可以直接在 MSDN 中查找帮助，以解决在开发过程中遇到的疑难问题。

第 3 章　ASP.NET 网站开发基础

ASP.NET 网页扩展名

可以把 ASP.NET 网页扩展名理解为 ASP.NET 文件的“身份证”，不同的扩展名决定了不同文件的类型和作用。通过 Internet 信息服务（IIS）将文件扩展名映射到 ASP.NET 运行处理。例如，Web 页面的扩展名为.aspx、母版页的扩展名为.master 等。

ASP.NET 页面指令

ASP.NET 页面的 HTML 代码中通常包含一些类似于<%@…%>这样的代码，被称为页面指令。

@Page 指令

@Page 指令允许开发人员为页面指定多个配置选项，并且该指令只能在 Web 窗体页中使用。每个.aspx 文件只能包含一条@Page 指令。

@OutputCache 指令

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

第 4 章　C#语言基础

数据类型

数据类型的表面含义是数据是属于哪种类型，就好像一个人是属于哪个民族一样。数据类型可分为值类型和引用类型。

值类型

所有的值类型都隐式地继承自 System.ValueType 类，该类又继承自 object 类。任何类型都不可能派生自值类型，因此所有的值类型都是隐式密封的。

整数类型

整数类型变量的值为整数，它的值有一定范围限制。

布尔类型

布尔类型主要用来表示 true/false 值，一个布尔类型的变量，其值只能是 true 或者 false，不能将其他的值指定给布尔类型变量，布尔类型变量不能与其他类型进行转换。

实数类型

实数在 C#中有单精度（float）和双精度（double）之分，它们的区别在于取值范围和精度不同。

结构类型

在实际生活中，我们经常把一组相关的信息放在一起。把一系列相关的变量组织成为一个单一实体的过程，我们称为生成结构的过程。这个单一实体的类型就叫做结构类型，每一个变量称为结构的成员。

枚举类型

枚举类型是一种具有命名常量的独特类型，枚举声明以关键字 enum 开始，然后定义枚举的名称、可访问性、基础类型和成员。

引用类型

引用类型是构建 C#应用程序的主要类型，引用类型的变量又称为对象，可存储对实际数据的引用。C#支持两个预定义的引用类型，即 object 和 string。

常量

常量又叫常数，它主要用来存储在程序运行过程中值不改变的数据。常量也有数据类型，C#语言中，常量的数据类型有多种，主要有 sbyte、byte、short、ushort、int、uint、long、ulong、char、float、double、decimal、bool 和 string 等。

变量

变量也就是可以变化的数据，每个变量都具有一个类型，这个类型确定了该变量中可以存储哪些值。C#是一种类型安全的语言，C#编译器保证了存储在变量中的值总是具有合适的类型。变量的值可以通过进行赋值或++和—运算符来更改。

隐式类型转换

隐式类型转换就是不需要声明就能进行的转换。进行隐式类型转换时，编译器不需要进行检查就能安全地进行转换。

显式类型转换

显式类型转换也可以称为强制类型转换，它需要在代码中明确地声明要转换的类型。如果在不存在隐式转换的类型之间进行转换，就需要使用显式类型转换。

加法运算符

加法运算符（+）通过两个数相加来执行标准的加法运算。

减法运算符

减法运算符（–）通过从一个表达式中减去另外一个表达式的值来执行标准的减法运算。

乘法运算符

乘法运算符（*）将两个表达式进行乘法运算并返回它们的乘积。

除法运算符

除法运算符（/）执行算术除运算，它用除数表达式除以被除数表达式而得到商。

求余运算符

求余运算符（%）返回除数与被除数相除之后的余数，通常用这个运算符来创建余数在特定范围内的等式。

赋值运算符

赋值运算符有= 、 += 、-= 、*= 、/= 、 %= 、&= 、|= 、^= 、<<= 、>>=、??等。C#语言中可以对变量进行连续赋值，这时赋值操作符是右关联的，这意味着从右向左操作符被分组，如 x=y=z 等价于 x=(y=z)。

关系运算符

关系运算符可以实现对两个值的比较运算，并且在比较运算之后会返回一个代表运算

结果的布尔值。

相等运算符

要查看两个表达式是否相等，可以使用相等运算符（==）。相等运算符对整型、浮点型和枚举类型数据的操作是一样的，它只简单地比较两个表达式，并返回一个布尔结果。

不等运算符

不等运算符（!=）是与相等运算符相反的运算符，有两种格式的不等运算符可以应用到表达式：一种是普通的不等运算符（!=）；另外一种是相等运算符的否定!(a= =b)。通常，这两种格式可以计算出相同的值。

小于运算符

如果比较一个值是否小于另外一个值，可以使用小于运算符（<）。当左边表达式的值小于右边表达式的值时，结果为真，否则结果为假。

大于运算符

如果比较一个值是否大于另外一个值，可以使用大于运算符（>）。当左边表达式的值大于右边表达式的值时，结果为真，否则结果为假。

第 5 章　掌握字符与字符串

Char 字符类

Char 在 C#中表示一个 Unicode 字符，正是这些 Unicode 字符构成了字符串，Unicode 字符是目前计算机中通用的字符编码，它为针对不同语言中的每个字符设定了统一的二进制编码，用于满足跨语言、跨平台的文本转换、处理的要求。其实字符离我们并不遥远，比如，计算机按键上的单个字母、单个数字都是字符。

转义字符

C#采用字符“\”作为转义字符。例如，定义一个字符，而这个字符是单引号，如果不使用转义字符，则会产生错误。

字符串

字符串类型表示一系列字符，是使用 string 关键字声明的一个字符数组。.NET Framework 中表示字符串的关键字为 string。

比较字符串

比较字符串是指按照字典排序的规则，判断两个字符串的大小，在英文字典中，在前面的单词小于在后面的单词。

格式化

格式化由格式说明符控制，该字符串指示如何表示基类型值。例如，格式说明符指示是否应该用科学记数法来表示格式化的数字，或者格式化的日期在表示月份时应该用数字还是用名称。

分割字符串

Split 方法可以把一个字符串按照某个分隔符分割成一系列小的字符串。

替换字符串

Replace 方法可以替换掉一个字符串中的某些特定字符或者子串。

StringBuilder 类

StringBuilder 类亦称为可变字符串，它位于 System.Text 命名空间下，之所以称之为可变字符串，是因为该类的对象在通过追加、移除、替换或插入字符串后不会生成新的字符串对象，即字符串变量始终指向同一个对象。

第 6 章　面向对象程序设计

面向对象程序设计

面向对象程序设计（Object-Oriented Programming）简称 OOP 技术，是开发应用程序的一种新方法、新思想，也是软件开发的主流。面向对象程序设计是在面向过程程序设计的基础上发展而来的，它将数据和对数据的操作看做是一个不可分割的整体，力求将现实问题简单化，因为这样不仅符合人们的思维习惯，同时也可以提高软件的开发效率，并方便后期的维护。

类

类是一种数据结构，它可以封装数据成员、函数成员和其他的类。类是创建对象的模板。C#的一切类型都是类，所有的语句都必须位于类内。类是 C#语言的核心和基本构成模块。

对象

对象是基于类的具体实体，有时称为类的实例。每一个对象都是依照某个类创建的实例。

成员字段

字段也称成员变量，它表示存储位置，是 C#类不可缺少的一部分，字段的类型可以是 C#中的任何数据类型。声明字段用标准的变量声明格式和访问修饰符来声明，并且可以对其初始化。

成员方法

成员方法用于执行一段程序或进行数据处理、逻辑判断等。方法在类中声明，声明时需要指定访问级别、返回值类型、方法名称以及方法中的参数。多个方法参数放在括号中，并用逗号隔开。空括号表示无参数方法。

成员属性

在面向对象程序设计语言中，属性（property）是指对象的特征和状态，具体地说就是指对象的数据成员。用户可以指定数据成员能否被外界直接访问，如果数据成员被指定为 public 的，外界就可以用“对象名.公有数据成员名”访问该成员。C#是完全面向对象

的语言，C#倡导一种新途径，对数据成员能够更好地封装和保护，同时又向外界提供更有效的访问形式。C#中用来达到这个目标的就是“属性”，而那些数据成员在C#中称为“字段”或“域”。

构造函数

只要创建类或结构，就会调用它的构造函数。类或结构可能有多个接受不同参数的构造函数。构造函数使得程序员可设置默认值、限制实例化以及编写灵活且便于阅读的代码。

析构函数

析构函数是以类名加~来命名的。.NET Framework 类库有垃圾回收功能，当某个类的实例被认为不再有效，并符合析构条件时，.NETFramework 类库的垃圾回收功能就会调用该类的析构函数实现垃圾回收。

封装

在面向对象编程中，大多数都是以类作为数据封装的基本单位。类将数据和操作数据的方法结合成一个单位。设计类时，不希望直接存取类中的数据，而是希望通过方法来存取数据，这样就可以达到封装数据的目的，方便以后的维护升级，也可以在操作数据时多一层判断。封装可以隐藏实现细节，使得代码模块化。

继承

在面向对象的世界中也存在着继承特性，继承是面向对象程序设计的重要特性之一，继承的最大优点就是提供了代码的重用。

多态

多态可简单地概括为“一个接口，多种方法”，它是在程序运行的过程中才决定调用的方法，其原理建立在“从父类继承而来的子类可以转换为其父类”这个规则之上。多态是一种概念，也是一种思想。重载（overload）和重写（override）都是多态的体现。

重载

重载（overload）是面向对象多态性的一个重要特征，它表示在一个类中定义多个同名方法，这些方法通过不同的参数类型或者参数个数进行区分。

重写

重写（override）也称为覆盖，从字面就可以知道，它是覆盖了一个方法并且对其重写，以求达到不同的作用。override 是重写基类的方法，在基类中的方法必须有修饰符 virtual（方法的声明中加上关键字 virtual，这种方法称为虚方法）。同时，在派生类中，重写方法中需要加上关键字 override。

虚方法

“虚”这个字可以理解为“不真实的”或“虚幻的”，虚方法用在哪里呢，比如 B 类继承于 A 类，B 类可以使用 A 类中定义的方法和字段，如果 A 类中有一个方法并不适合 B 类使用，就要用到虚方法了，可以将 A 类中的这个方法定义为虚方法，这时在 B 类就可以重写这个虚方法，将方法变为自己的方法。

第7章　掌握流程控制语句

条件语句

条件语句可根据不同的条件执行不同的语句。条件语句主要包括 if 条件语句和 switch 多分支语句。

循环语句

循环语句就是在满足一定条件的情况下反复执行某一个操作。在 C#中提供了四种常见的循环语句，分别是 while 语句、do…while 语句、for 语句和 foreach 语句。

第8章　数组与集合

数组

数组是包含若干相同类型的变量的集合。这些变量可以通过索引进行访问。数组的索引从 0 开始。数组中的变量称为数组的元素。数组中的每个元素都具有唯一的索引与其对应。数组主要包括一维数组、二维数组等。

冒泡排序法

冒泡排序是一种最常用的排序方法，其过程很简单，就像气泡一样越往上走越大，因此被人们形象地称为冒泡排序法。冒泡排序的过程很简单，首先将第一个记录的关键字和第二个记录的关键字进行比较，若为逆序，则将两个记录交换，然后比较第二个记录和第 3 个记录的关键字，依次类推，直至第 n-1 个记录和第 n 个记录的关键字进行过比较为止，上述过程称为第一趟冒泡排序，执行 n-1 次上述过程后，排序即可完成。

直接插入排序法

直接插入排序是一种最简单的排序方法，其基本操作是将一个记录插入到已排好序的有序表中，从而得到一个新的、记录数增 1 的有序表，然后再从剩下的关键字中选取下一个插入对象，反复执行直到整个序列有序。

选择排序法

选择排序的基本思想是，每一趟在 n 个记录中选取关键字最小的记录作为有序序列的第 I 个记录，并且令 I 从 1 至 n-1，进行 n-1 趟选择操作。

第9章　掌握 ASP.NET 内置对象

ASP.NET 内置对象

ASP.NET 的基本对象是程序设计中最频繁使用的元素，它通过向用户提供基本的请

求、响应、会话等处理功能实现了 ASP.NET 的绝大多数功能。这些对象主要有 Request 对象、Response 对象、Server 对象、Session 对象、Application 对象记录应用程序参数对象、Cookie 对象。

第 10 章　ADO.NET 数据库开发技术

ADO.NET

ADO.NET 是微软新一代.NET 数据库的访问架构，ADO 是 ActiveX Data Objects 的缩写。ADO.NET 是数据库应用程序和数据源之间沟通的桥梁，主要提供一个面向对象的数据访问架构，用来开发数据库应用程序。

SQL 注入式攻击

在与数据库交互的 Web 应用程序中严重的风险之一就是 SQL 注入式攻击，该攻击是指利用 SQL 语句设计上的漏洞，在目标服务器上运行 SQL 命令以及进行其他方式的攻击。比较有效的防范措施就是使用参数化查询预防 SQL 注入。

第 11 章　ASP.NET 服务器控件

服务器控件

在 ASP.NET 3.5 中，服务器控件是指在服务器上执行程序逻辑的组件。这个组件可能生成一定的用户界面，也可能不包括用户界面。

文本类型控件

文本类型控件主要包括：标签控件 Label 和文本框控件 TextBox，都是用来接收文本信息。

文本框控件

按钮类型控件例如 Button 控件（分为提交按钮控件和命令按钮控件）和 ImageButton 图像按钮控件。按钮类型控件在开发中应用是非常广泛的，基本上每个页面都会有按钮控件的影子。

按钮类型控件

按钮类型控件主要包括 Button 控件（分为提交按钮控件和命令按钮控件）和 ImageButton 控件图像按钮控件。按钮类型控件在开发中应用是非常广泛的，基本上每个页面都会有按钮控件的影子。

提交按钮控件

提交按钮控件只是将 Web 页面回送到服务器，在默认情况下，Button 控件为提交按钮控件。

命令按钮控件

命令按钮控件一般包含与控件相关联的命令，用于处理控件命令事件。

列表类型控件

列表框控件例如ListBox控件用于显示一组列表项和DropDownList控件从列表中选择一项，而且只在框中显示选定项。

选择类型控件

选择类型控件主要包括 RadioButton 控件（一种单选按钮控件）和 CheckBox 控件（用于在页面上创建出简单的复选框）。

图形显示类型控件

图形显示类型控件主要包括 Image 控件和 ImageMap 控件。使用 Image 控件可以在设计或运行时以编程方式为 Image 对象指定图形文件。ImageMap 控件允许在图片中定义一些热点（HotSpot）区域。当用户单击这些热点区域时，将会引发超链接或者单击事件。

第 12 章　数据绑定控件

GridView 控件

GridView 控件有一个内置分页功能，可支持基本的分页功能。在启用其分页机制前需要设置 AllowPaging 和 PageSize 属性，AllowPaging 决定是否启用分页功能，PageSize 决定分页时显示几条数据（默认值为 12）。

DataList 控件

DataList 控件是一个常用的数据绑定控件，该控件能够以某种设定好的模板格式循环显示多条数据。这种模板格式是可以根据需要进行自定义的，比较于 GridView 控件，虽然 GridView 控件功能非常强大，但它始终只能以表格的形式显示数据，而使用 DataList 控件则灵活性非常强，其本身就是一个富有弹性的控件。

第 13 章　创建自己的 Web 用户控件

Web 用户控件

Web 用户控件是一种服务器控件，它与 ASP.NET 页面有着同样的“所见即所得”的特点和声明性样式，并以.ascx 为扩展名存储为文本文件。Web 用户控件与完整的 ASP.NET 网页（即.aspx 文件）非常相似，同时具有自己的用户界面页和代码。开发人员可以采取与创建 ASP.NET 页相似的方式创建 Web 用户控件，然后向其中添加所需的标记和子控件。Web 用户控件可以像页面一样包含对其内容进行操作（包括执行数据绑定等任务）的代码。

用户控件的属性

在用户控件中，属性是一种有效的并向类型使用者公开数据的字段，从类型使用者的角度来看，属性是一个 public 字段，通过实现一个属性，可以将使用者和实现细节相互隔离，同时还可以在属性被访问时提供数据有效检查、跟踪等处理手段。

第 14 章　ASP.NET 验证控件

验证控件

ASP.NET 提供了一组验证控件，只要通过属性设定，它们便会自动产生相关的 JavaScript，当用户输入时直接在浏览器中检查，响应速度快。基本上每个验证控件只能做一种检查，如有没有输入值、是否在某个范围内等。当用户输入的值不符合条件时，验证控件会提示报错信息，且中断后续的数据处理操作。

客户端验证

在客户端进行验证响应速度快，在最接近使用的地方进行检查，一旦有错会被浏览器中的 JavaScript 拦截下来，用户必须更正错误，通过验证，才能继续之后的处理。这里所说的执行拦截检查操作的 JavaScript 在 ASP.NET 会自动建立。若要取消客户端验证，可将验证控件的 EnableClientSideScript 属性设置为 False。

服务器端验证

无论是否通过客户端的检查，服务器端的 ASP.NET 程序会对所有提交的数据进行检查，如与服务器端的数据库内容进行核对。在服务器端再重复一次相同的检查仍旧是必要的，因为“有心”的用户可以通过一些手法，跳过或免除客户端的验证步骤，所以开发 Web 网站其安全性是非常重要的，这就像对重要的物品在存放到仓库中要有两道或多道防盗门一样设置多重防范措施。

非空数据验证

在网站的实际开发中，RequiredFieldValidator 验证控件用来验证输入文本中的信息内容是否为空。例如，注册会员信息时，用户名称、密码、电话、住址等较重要的信息为必填项，这时需要使用 RequiredFieldValidator 控件来进行验证。

比较验证控件

CompareValidator 控件称为比较验证控件，其功能是将输入的值与常数值或其他控件输入的值相比较，以确定这两个值是否与比较运算符（小于、等于、大于等）指定的关系相匹配。

数据范围验证

在实际开发过程中，RangeValidator 验证控件用来验证输入控件中的数据是否在指定的上限与下限之间。例如，年龄输入限制范围为 18～68、日期输入限制范围 2000-01-01～2010-01-01 之间，这时需要使用 RangeValidator 控件来验证。

验证错误信息汇总

ValidationSummary 控件又称错误汇总控件，主要用于收集本页中所有验证控件错误信息，将它们组织好并一同显示出来，此控件属于服务器端验证控件。

自定义验证控件

CustomValidator 控件为输入控件提供用户定义的验证函数。例如，可以创建一个验

证控件，该控件检查在文本框中输入的值是否为偶数。

第 15 章　利用 GDI+绘制 Web 图形图像

GDI+

GDI+是在 GDI（即 Windows 早期版本中附带的 Graphics Device Interface）的基础上进行了改进，添加了新功能并优化了原有功能。GDI+是 Windows XP 操作系统中提供二维矢量图形、图像处理和版式的部分。

第 16 章　AJAX 无刷新技术

AJAX

AJAX 是 Asynchronous JavaScript and XML（异步 JavaScript 和 XML 技术）的缩写，它是由 JavaScript 脚本语言、CSS 样式表、XMLHttpRequest 数据交换对象和 DOM 文档对象（或 XMLDOM 文档对象）等多种技术组成的。

第 17 章　调试与错误处理

异常

在理想的情况下，程序会像充分润滑的机器一样运转，没有内部逻辑错误，也没有因程序异常而被迫终断的情况。但是，程序与真实世界一样，随时都可能出现不可预见的逻辑错误或异常终断。在程序语言中，这些意料之外的事件称为异常。

语法错误

语法错误是一种程序错误，它会影响编译器完成工作。它也是最简单的错误，几乎所有的语法错误都能被编译器或解释器发现，并将错误消息显示出来提醒程序开发人员。

语义错误

程序源代码的语法正确而语义或意思与程序开发人员本意不同时，就是语义错误。此类错误比较难以察觉，它通常在程序运行过程中出现。语义错误会导致程序非正常终止。例如，在将数据信息绑定到表格控件时，经常会出现“未将对象引用设置到对象的实例中”错误，此类语义错误在程序运行时，将会被调试器以异常的形式告知程序开发人员。

逻辑错误

不是所有的语义错误都容易发现，它们可能隐藏得很深。在某些语义错误下，程序仍可以继续执行，但执行结果却不是程序开发人员想要的，此类错误就是逻辑错误。例如，在程序中，需要计算表达式 c=a+b 的值，但在编程的过程中，将表达式中的“+”，写成了“-”，像这样的错误，调试器不能以异常的形式告诉程序开发人员，这种错误就是逻辑

ÓÉ Foxit PDF Editor ±à¼-
°æÈ¨ËùÓÐ (c) by Foxit Software Company, 2004
½öÓÃÓÚÆÀ¹À¡£

- android与iphone及ipad开发书籍 ----------持续不断更新中.....
- c、c++、c#语言pdf书籍及vip视频教程 c、c++、c#、vc等-------持续不断更新中.....
- delphi《书籍》及《视频》教程 ----------持续不断更新中.....
- E网情深VIP系列视频教程 黑客破解菜鸟修练班，VB编程学习班，仿站学习培训，免杀培训，个人系统攻防系列教程，服务器搭建学习班，PHOTOSHOP平面设计班，基础制作论坛（论坛网站搭建），网赚系列教程，网站建设教程，网站漏洞基础，远程控制教程，软件破解班，脚本漏洞提权班
- IT9网络学院VIP系列视频教程 免杀培训班，VMware虚拟机，零基础学习C语言，网游外挂开发精品系列语音教程（外挂教程学习必备研修31课全），VB语言教程30课全，Delphi编程到精通，远程控制软件，加密解密班，网络安全与黑客攻防培训，从入门到精通完整系统化学习C++编程，从入门到精通零基础学习汇编，wordpress教程(个人博客系统49课全)，外行人做易语言盗号和钓鱼程序语音教程 网址：WLSAM168.400GB.COM
- Java书籍 ----------持续不断更新中.....
- photoshop、CorelDRAW、AutocAD等图像处理书籍及vip视频教程 -------持续不断更新中.....
- powerbuilder书籍大全
- Visual Basic语言vip视频教程及pdf书籍 ----------持续不断更新中.....
- windows、linux系统开发、系统封装等pdf书籍及VIP视频教程 ----------持续不断更新中.....
- 《3DS Max》pdf书籍
- 《汇编语言》、《反汇编》及《调试》pdf书籍及vip视频教程 ----------持续不断更新中.....
- 《电子书、电子书、还是电子书》pdf专题库 编程开发，家居美食，儿童益智，人物传记，增强记忆，快速阅读
- 信息系统项目管理师、网络工程师、系统分析师等软考类书籍
- 华中红客系列vip视频教程 脚本攻防培训班，源码免杀培训班，Css语言培训班，C语言，Dreamweaver网页设计，html网页设计培训班，PC安全班， php脚本语言培训班，VMWare虚拟机专题，webshell提权培训班，防站教程，零基础免杀培训班，刷钻速成班，脱壳破解班，外挂编写班，网络赚钱培训班，网站入侵培训班
- 外挂、驱动、逆向及封包视频教程 郁金香、独立团、夜猫论坛、天都吧、看流星论坛、一切从零开始等等
- 安全中国系列vip视频教程 易语言软件编程培训班，ASP.net网站开发项目实战培训班
- 我的收藏
- 按键精灵及TC脚本开发软件视频教程 ----------持续不断更新中.....

当前位置：/《电子书、电子书、还是电子书》pdf专题库

文件名 ▲▼

PDF电子书专题库，内容详尽，每天不断更新！！

- 办公类软件使用指南
- 医学
- 历史人物传记
- 哲学宗教
- 外语资料（除英语外） （除英语外）
- 官场类小说
- 建筑工程类
- 情感生活类小说
- 政治军事
- 教育学习科普大全
- 文学理论
- 智力开发、增强记忆、快速阅读技巧大全
- 社会生活
- 科学技术
- 程序编程类
- 经济管理
- 网络安全及管理
- 网赚系列
- 美食小吃烹饪煲汤大全
- 课外读物

本网盘内容太多，持续不断更新，发布各类视频教程、pdf书籍，包括破解、加解密、外挂辅助制作，易语言培训教程、编程语言、网页制作等等，教程及书籍仅用于学习,如用于商业或非法用途的后果自负！

网址：WLSAM168.400GB.COM

- VIP培训教程，易语言黑月VIP视频教程，天都网易语言系列培训教程(100集全)，集中营易语言学习视频(80集)
- 棉猴系列vip视频教程 gh0st远控控制源码讲解教程，套接字编程，DLL程序编写，键盘监听驱动程序编写，驱动基础教程，AsyncSelect模型QQ程序教程，C++语言入门基础，NB5.5源码分析教程
- 游戏开发pdf书籍 ----------持续不断更新中.....
- 炒股投资pdf书籍及视频教程 短线高手系列，短线天王系列，操盘论道系列，翻倍黑马，看盘快速入门，庄家手法大曝光等等。 网址：WLSAM168.400GB.COM
- 热门小说集中营 傲世九重天，网游之三国时代，武动乾坤
- 甲壳虫VIP教程全集 asp教程，Delphi培训班，FLASH培训班，Java培训班，linux培训班，PHP培训班，源码免杀班，甲壳虫C++，脚本功防班，免杀班初、中、高级班，破解班，源码免杀班，脱壳班，易语言培训班，无特征码免杀，网站架构培训班，外挂高级班，外挂初级班第1、2部
- 破解、免杀、入侵、脱壳、攻防及漏洞分析系列VIP视频教程（80多部） 天草、黑客动画吧等等--------持续不断更新中....
- 网站建设相关的pdf书籍及各种vip视频教程 ----------持续不断更新中.....
- 网赚、淘宝系列vip视频教程 网赚30天新人魔鬼训练，屠龙网赚团队vip课程，站长大学网赚视频（50课全），图腾团队日赚1000元竞价行销教程，屠龙团队淘宝宝贝卖疯系列，站群网赚系列，淘宝开店视频，红星挂机日赚10元，百万流量系列，漂流瓶圣手全自动挂机引，贴吧邮件定向营销疯狂成交量月入万元
- 英语学习资料百科大全 不断更新。。。
- 饭客论坛系列VIP视频教程 脚本入侵班，黑客之免杀教程，易语言教程，无线网络攻防教程，入侵教程，delphi系列教程，黑客基础入门
- 黑客书籍 有关黑客、安全、加解密技术等等----------持续不断更新中.....
- 黑手安全网VIP系列视频教程 DIV+CSS网页布局，Dreamweaver教程，flsah动画教程，photoshop教程，跟我一起学C++课程，抓鸡
- 黑鹰、黑基、黑防、黑盾vip系列视频教程 破解提高班66课全，SQL注入，ASP注入教程，完完全全学会抓肉鸡，脱壳破解教程50课全，提权班，C语言特训班26讲全，黑客脚本特训班，黑客工具特训班，dedecms仿站教程，VC编写远控30课全，网页美工特训班，木马免杀特训班，驱动开发技术VIP培训班，外挂破解等等。

- [电脑世界的通关密语：电脑编程基础].(杉浦贤).滕永红.扫描版.pdf
- [程序语言的奥妙：算法解读（四色全彩）].(杉浦贤).李克秋.扫描版.pdf
- [差错：软件错误的致命影响].(帕伯斯).邝宇恒等.扫描版.pdf
- [算法之道（第2版）].邹恒明.扫描版.pdf
- [O'Reilly：深入学习MongoDB].(霍多罗夫).巨成等.扫描版.pdf
- [深入浅出WPF].刘铁猛.扫描版.pdf
- [Go语言·云动力（云计算时代的新型编程语言）].樊虹剑.扫描版.pdf
- [精通.NET互操作：P/ Invoke、C++ Interop和COM Interop].黄际洲等.扫描版.pdf
- [编程的奥秘：.NET软件技术学习与实践].金旭亮.扫描版.pdf
- [O'Reilly：学习OpenCV（中文版）].(布拉德斯基等).于仕琪等.扫描版.pdf
- [Go语言编程].许式伟等.扫描版.pdf 网址：WLSAM168.400GB.COM
- [MySQL技术内幕：SQL编程].姜承尧.扫描版.pdf
- [Tomcat权威指南（第2版）].(布里泰恩等).吴豪等.扫描版.pdf
- [Ext江湖].大漠穷秋.扫描版.pdf
- [IT名人堂·Oracle DBA突击：帮你赢得一份DBA职位].张晓明.扫描版.pdf

Total: 77 1 2 3 4 5 6 ›

HTTP://WLSAM168.400GB.COM

错误。

断点

断点是一个信号，它通知调试器在某个特定点上暂时将程序执行挂起。当执行在某个断点处挂起时，该程序处于中断模式。进入中断模式并不会终止或结束程序的执行，程序可以在任何时候继续运行。

逐语句

如果某一行包含函数调用，“逐语句”仅执行调用本身，然后在函数内的第一个代码行处停止。

逐过程

“逐过程”执行整个函数，然后在函数外的第一行处停止。如果要查看函数调用的内容，则使用“逐语句”。