



PHP and MySQL for Dynamic Web Sites Visual QuickPro Guide Fourth Edition

PHP与MySQL动态网站开发

(第4版)

[美] Larry Ullman 著
杜凯 陈宗斌 译

- 广受赞誉的PHP和MySQL入门教程
- 高效、直观的学习方式
- 任务导向，便于查询



Larry Ullman

作家、Web和软件开发人员、培训师、教师、演说家和顾问。他已经著有20多本技术书籍。Larry最深受读者喜爱的一点在于，他往往可以将晦涩难懂的专业术语转换为大家耳熟能详的自然语言。更多信息可访问他的网站：www.LarryUllman.com。



PHP and MySQL for Dynamic Web Sites Visual QuickPro Guide Fourth Edition

PHP与MySQL动态网站开发

(第4版)

[美] Larry Ullman 著
杜凯 陈宗斌 译

人民邮电出版社

图灵社区会员 StinkBC(StinkBC@gmail.com) 专享 尊重版权

图书在版编目 (C I P) 数据

PHP与MySQL动态网站开发 : 第4版 / (美) 厄尔曼
(Ullman, L.) 著 ; 杜凯, 陈宗斌译. -- 北京 : 人民邮
电出版社, 2013.1

(图灵程序设计丛书)

书名原文: PHP and MySQL for Dynamic Web
Sites: Visual QuickPro Guide, Fourth Edition
ISBN 978-7-115-29940-6

I. ①P… II. ①厄… ②杜… ③陈… III. ①

PHP语言—程序设计—教材②关系数据库—数据库管理系统
—程序设计—教材 IV. ①TP312②TP311.138

中国版本图书馆CIP数据核字(2012)第271692号

内 容 提 要

本书采用基于任务的方法来讲授 PHP 和 MySQL，使用大量图片指导读者深入学习语言，并向读者展示了如何构造 Web 站点。用简洁、直观的步骤和讲解提供了学习任务和概念的最快方式。通过本书，读者可以快速、高效地学习 PHP 和 MySQL，并可以立刻成为一位构建 Web 站点的高手！

本书适用于 Web 应用开发人员，适合初、中层次读者。

图灵程序设计丛书 PHP与MySQL动态网站开发 (第4版)

-
- ◆ 著 [美] Larry Ullman
 - 译 杜 凯 陈宗斌
 - 责任编辑 毛倩倩
 - 执行编辑 丁晓昀
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
 - 邮编 100061 电子邮件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 北京 印刷
 - ◆ 开本: 800×1000 1/16
 - 印张: 41
 - 字数: 1097千字 2013年1月第1版
 - 印数: 1~4 000册 2013年1月北京第1次印刷
 - 著作权合同登记号 图字: 01-2011-6040号
-

ISBN 978-7-115-29940-6

定价: 99.00元

读者服务热线: (010)51095186转604 印装质量热线: (010)67129223

反盗版热线: (010)67171154

版 权 声 明

Authorized translation from the English language edition entitled *PHP and MySQL for Dynamic Web Sites: Visual Quick Pro Guide Fourth Edition* by Larry Ullman, published by Pearson Education, Inc., publishing as Peachpit Press, Copyright © 2012 by Larry Ullman.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanic, including photocopying, recording, or by an information storage retrieval system, without permission of Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by POST & TELECOM PRESS Copyright © 2013.

本书中文简体字版由美国Pearson Education 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

献　　辞

献给我的母校东北密苏里州立大学（Northeast Missouri State University）的全体优秀的教员。特别感谢Monica Barron博士、Dennis Leavens博士、Ed Tyler博士和Cole Woodcox博士，我非常荣幸能成为他们的朋友。如果没有从这些老师那里接受无私的、感人的和杰出的指导，我就不能从一名学生成为一位作家、一位教师或者一个对社会有意义的人。

前　　言

今天的Web用户期待更吸引人的页面——它们会频繁更新，并且提供个性化的体验。在他们看来，Web站点更像是社区，他们将一遍又一遍地回访。同时，Web站点管理员希望站点更容易更新和维护，他们理解到这是能够不断满足访问者期望的唯一方式。由于如此种种原因，PHP和MySQL变成了创建动态的、数据库驱动的Web站点的事实标准。

本书可以说是凝聚了我多年Web开发经验和多部Web开发技术图书写作经验的颠峰之作。本书重点在于以最高效的方式介绍最重要的知识。它将介绍如何开始开发动态Web站点，并给出了大量示例代码来帮助读者起步。你只需要满怀热忱地来学习就行了。

好吧，我们这就开始……

什么是动态Web站点

动态Web站点非常灵活、强大，将其描述为应用程序（application）而不仅仅是站点会更准确。动态Web站点的特征包括：

- 能够对不同的参数做出响应（例如，一天中的某个时间，或者访问者的Web浏览器版本）；
- 具有“记忆”，允许用户执行注册、登录、电子商务以及类似的过程；
- 通常包含HTML表单，使得人们可以执行查找、提供反馈等；
- 通常具有允许管理员管理站点内容的界面；
- 与静态创建的站点相比，更易于维护、升级和构建。

有许多技术可用于创建动态Web站点。最常用的技术是ASP.NET、JSP（Java ServerPages）、ColdFusion、Ruby on Rails和PHP。动态Web站点不一定依赖数据库，但是，越来越多的动态Web站点正在这样做，何况还有MySQL这样几乎可以免费使用的数据库。

什么是PHP

PHP最初代表“个人主页”（Personal Home Page），由Rasmus Lerdorf于1994年创建，用于跟踪访问者对其在线履历的访问。随着实用性和功能的不断提高（并且也开始用于更专业的环境中），它变成了“PHP：Hypertext Preprocessor（PHP：超文本预处理器）”。

根据www.php.net（参见图0-1）上PHP官方站点的说法，PHP是“一种广泛使用的通用脚本语言，特别适用于Web开发，并且可以嵌入在HTML中”。这是一个复杂但具有描述性的定义，其含义将在后面解释。

称PHP“可以嵌入在HTML中”，意味着在标准的HTML页面中根据需要插入一些PHP代码，就可以得到动态效果。因此PHP很适合网页设计和制作者使用。

此外，与编译语言相比，PHP是一种脚本语言：设计PHP的目的是用于编写Web脚本，而不是编写独立的应用程序（当然，现在多费点劲也可以用PHP创建应用程序）。PHP脚本只在某个事件（例如，用户提交一个表单或者输入一个URL地址）发生之后才运行。

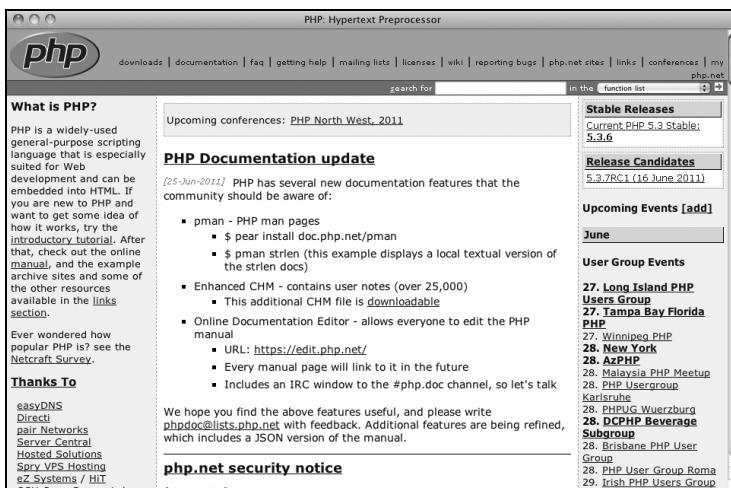


图0-1 PHP主页

我应该在这个定义中添加一些内容，指出PHP是一种服务器端、跨平台的技术，这两个描述都是重要的。服务器端是指PHP做的所有事情都发生在服务器上。这需要Web服务器，像Apache或微软公司的IIS（Internet Information Services，Internet信息服务），并且必须通过URL（以http://开始的网址）访问所有PHP脚本。跨平台的意思是，PHP可以运行在大多数操作系统上，包括Windows、UNIX（及其许多变体）和Macintosh。更重要的是，对于在一台服务器上编写的PHP脚本，通常不用修改或者只做很少的修改即可工作在另一台服务器上。

PHP6发生了什么？

当我撰写本书的第3版时，PHP的下一个主要版本PHP 6已经开发了将近一半儿了。考虑到PHP 6可能会在那一版出版后不久发布，我就在当中加入了一些PHP 6 beta版的内容。但不幸的是，PHP 6夭折了。

PHP 6最主要的新特性是将Unicode引入PHP引擎，这意味着PHP 6将可以处理世界上所有的语言字符了。这将使这个已经非常受欢迎的编程语言锦上添花。不幸的是，实现对Unicode的支持十分复杂且难度很大，语言的开发者搁置了PHP 6的开发。然而不是所有的都取消了：PHP6计划引入的一些新特性，比如命名空间（面向对象编程的概念），就已添加到PHP 5.3版中。

在撰写这一版时，何时实现Unicode支持尚不明朗，PHP 6的正式发布日期还未可知。我预感未来一段时间内PHP将会沿着版本5的主干渐进发展。

在编写本书的时候，PHP的最新是5.3.6版本，建议读者使用PHP5.0以上的版本。本书中要用到的一些函数和特性需要运行在PHP 5.2及以上版本。当用到新版本中引入的功能时，我会为那些仍使用较低版本的读者提供一些替代解决方案。

如果你仍然还使用着PHP4，强烈建议升级到更高版本。如果你没有这个计划，那么请购买本书的第2版。

更多关于PHP的信息，PHP核心背后的思想，可以随时在PHP.net或者Zend (www zend com) 上找到。

为什么使用PHP

简单地讲，在开发动态Web站点时，与其他可选技术相比，PHP更好、更快并且更易于学习。PHP有优秀的性能、与几乎所有数据库的紧密集成、稳定性、可移植性，以及由于其可扩展性而得到的几乎无限的特性集。所有这些都是免费的（PHP是开源技术），并且非常易于学习。在我接触的语言中，PHP是最佳地结合了易用性和高级能力的语言之一，初级程序员使用它很容易上手，更高级的程序员可以用它做他们需要的一切事情。

最后，一个事实可以证明这一点：PHP自从推出以来，其用户数量呈指数级增长，76%的网站都采用PHP作为其服务端技术（参见图0-2）。在所有受欢迎的编程语言中，PHP排在第5位（参见图0-3）。

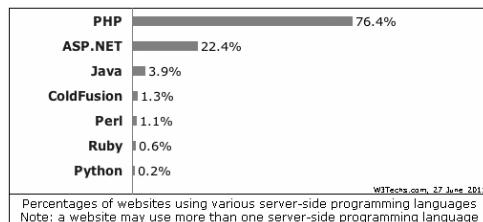


图0-2 Web Technology Surveys 网站提供的服务器端技术调查表
(www.w3techs.com/technologies/overview/programming_language/all)

Position Jun 2011	Position Jun 2010	Delta in Position	Programming Language	Ratings Jun 2011	Delta Jun 2010	Status
1	2	↑	Java	18.580%	+0.62%	A
2	1	↓	C	16.278%	-1.91%	A
3	3	==	C++	9.830%	-0.55%	A
4	6	↑↑	C#	6.844%	+2.06%	A
5	4	↓	PHP	6.802%	-2.47%	A
6	5	↓	(Visual) Basic	4.727%	-0.93%	A

图0-3 Tiobe Index (<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>)
结合各种因素给出的流行的编程语言排名

当然，由于我是PHP图书（实际上市面上有很多本这样的图书）的作者，你可能想当然地认为我的观点有失公平。尽管我使用JSP、RoR（Ruby on Rails）和ASP.NET不像PHP那样广泛，但是也使用它们开发站点。它们都有自己的优缺点，但是我总是会返回到PHP这种技术上来。你可能听说它的性能和扩展性不如其他技术，但是Yahoo!、维基百科、Facebook都使用PHP，比这几个网站的访问量多

的网站可不多啊。

你也可能想知道PHP的安全性如何。但是安全性不在语言本身，而在于使用语言的方式。当然，本书将全面讨论所有重要的安全性问题的最新内容。

PHP如何工作

如前所述，PHP是一种服务器端语言。这意味着用PHP编写的代码将驻留在称为服务器的主机上。服务器发送Web页面给发出请求的访问者（你、客户端和Web浏览器）。

当访问者访问用PHP编写的Web站点时，服务器读取PHP代码，然后依据其脚本指令处理它。在图0-4所示的示例中，PHP代码告诉服务器发送合适的数据（HTML代码）给Web浏览器，Web浏览器再把接收到的代码处理成标准HTML页面。

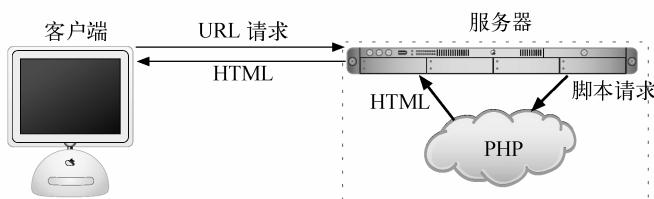


图0-4 当用户请求Web页面时，PHP如何在客户/服务器模型中发挥作用

这不同于静态HTML站点。在静态HTML站点中，当发出请求时，服务器只是把HTML数据发送到Web浏览器，而不会由服务器端进行解释（参见图0-5）。由于不需要服务器端的动作，所以可以在Web浏览器中运行HTML页面，而根本不需要使用服务器。

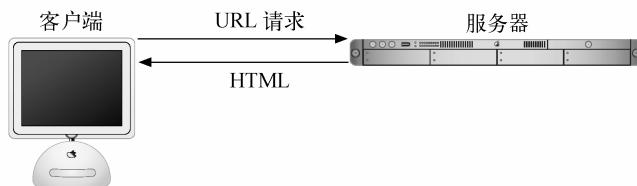


图0-5 当对静态HTML页面发出请求时客户/服务器的处理过程

对于最终用户和Web浏览器来说，`home.html`和`home.php`两者在外观上并没有明显的区别，但是，其页面内容的创建方式却有着天壤之别。

什么是MySQL

MySQL (www.mysql.com, 参见图0-6) 是世界上最流行的开源数据库。事实上，今天MySQL成为了那些昂贵的重量级数据库（如Oracle和微软公司的SQL Server）有力的竞争产品。像PHP一样，MySQL提供了优秀的性能、可移植性和可靠性，易于学习，并且几乎是免费的。

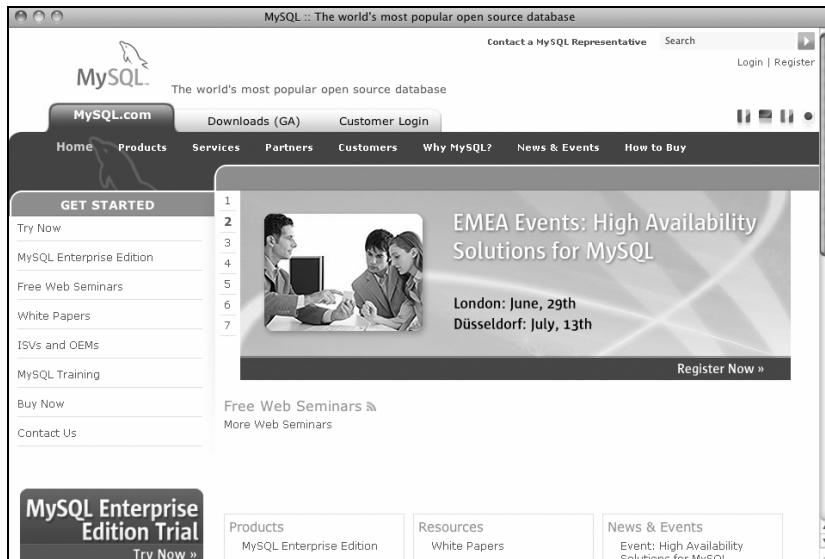


图0-6 MySQL数据库应用程序的主页

发音指南

虽然只是小节，但我仍要预先说明：MySQL应该读作“*My Ess Que Ell*”，就像SQL应该读作“*Ess Que Ell*”一样。许多人最初都会对此产生疑问，虽然这并不是什么大问题，但是，学会正确地读缩写词总是好事。

MySQL是一种关系数据库管理系统（DBMS，Database Management System）。简单地讲，数据库是一些相关数据的集合，这些数据可以是文本、数字或二进制文件，它们由DBMS进行存储和组织。

数据库有多种类型，从简单的平面文件到关系数据库和面向对象数据库。关系数据库的特征是使用多张表存储信息。虽然关系数据库在设计和编程阶段需要做更多工作，但它们提高了可靠性和数据完整性，抵消不足绰绰有余。此外，关系数据库的查找能力更强，并且允许并发操作。

通过把数据库纳入Web应用程序中，PHP生成的数据可以从MySQL提取（参见图0-7）。这进一步把站点的内容从静态（硬编码）转为灵活，灵活性对动态Web站点可是至关重要的。

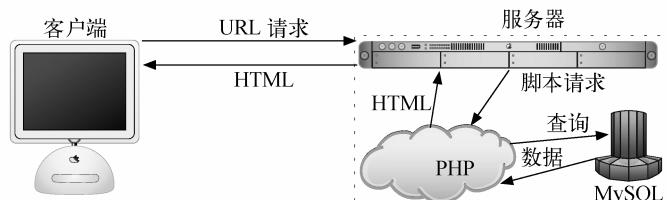


图0-7 本书中大多数动态Web应用程序的工作方式：同时使用PHP和MySQL

与PHP一样，MySQL也是一种开源应用程序，这意味着它可以免费使用，甚至可以修改（源代码可下载得到）。有些情况下需要付费获得MySQL许可证，特别是在销售或打包使用MySQL产品来赢利时则更应如此。查看MySQL的许可政策，可以获取这方面的详细信息。

MySQL软件包含多个部分，包括MySQL服务器（mysqld，它运行和管理数据库）、MySQL客户（mysql，它提供了一个访问服务器的接口），以及出于维护等目的而提供的大量实用程序。PHP对MySQL的支持一直很好，在PHP的最新版本中，这一点表现得更为突出。

MySQL以处理大型数据库而著称，数据库可以包含60 000张表，以及超过50亿行的记录。在某些操作系统上，MySQL可以处理容量高达800万TB的表，在其他操作系统上，一般可以正常地处理4 GB的数据。MySQL已被NASA、美国人口普查局（United States Census Bureau）以及许多其他机构采用。

在编写本书时，MySQL已经推出了版本5.5.13，并且版本5.6和版本6.0正在开发中。因为不同的MySQL版本有不同的特性，所以要清楚地知道你自己正在使用什么版本就非常重要了。本书使用的是MySQL 5.1.44和5.5.8，尽管只要使用MySQL 5.0以上版本就应该能够完成本书中的任何事情。

你需要什么

要理解本书的示例，你需要以下工具：

- Web服务器应用程序（例如，Apache、Abyss或IIS）；
- PHP；
- MySQL；
- Web浏览器（微软公司的IE、Mozilla的Firefox、Apple的Safari、Google的Chrome等）；
- 文本编辑器、支持PHP的所见即所得的应用程序（Adobe的Dreamweaver就具有这种能力）或者IDE（Intergrated Development Environment，集成开发环境）；
- FTP应用程序（如果使用远程服务器）。

利用PHP和MySQL开发动态Web站点的一大优点是，无论什么要求都可以免费得到满足，而不管使用的操作系统是什么！Apache、PHP和MySQL全都是免费的，大多数Web浏览器可以免费拥有，许多优秀的文本编辑器可供免费使用。

本书附录（可以从本书的支持网站<http://www.peachpit.com>下载）讨论了在Windows和Mac OS X操作系统的安装过程。如果有一台计算机，那么只需下载两个产品即可创建动态Web站点（在这种情况下，你的计算机同时代表图0-4和图0-5中的客户端和服务器）。另外，你还可以以每月几美元的价钱购买Web虚拟主机服务，它会提供支持PHP和MySQL的已经在线的环境。

关于本书

本书讲述了如何利用PHP和MySQL来开发动态Web站点，涵盖了大多数开发人员可能需要的知识。为了与Visual QuickPro系列图书的格式保持一致，本书中使用逐步引导、图文并茂的方式来讨论。其重点依然放在实战性很强的示例上，而不像有些书那样，老是说什么“这些事情你能够做到但是永远也不要去做”。我自己就是一名Web开发人员，书里写的都是我会用到的信息，并且避免了那些对手边的任务来说无关紧要的东西。而作为一名老作者，我肯定会包括读者想知道的主题和技术。

本书采用了循序渐进的结构。前3章介绍了PHP的基础知识（通过学习第2章，你就会开发你的第一个动态Web页面）。之后，第4~7章介绍了SQL（Structured Query Language，结构化查询语言，用于和所有数据库进行交互）和MySQL。它们介绍了SQL和数据库设计的基础知识，并且特别介绍了MySQL应用程序。然后，第8章介绍了调试和错误管理，这些是每个人都需要的信息。第9章专门介绍了如何结合使用PHP和MySQL，这非常容易做到。

第10~14章讲述了更多的应用技术，可以充实你的知识。特别是，这几章中反复介绍了安全方面的内容。第15章和第16章是本书这一版中的全新内容，介绍了一些新技术。最后，本书包含了专门介绍示例的3章内容，其中开发了几类Web应用程序的核心，并穿插了许多说明。

本书读者对象

本书读者面很广，从初学者到中级用户都可以学习本书。考虑将来的兼容性问题，本书使用了XHTML，因此读者必须具有使用XHTML或其前身HTML的丰富经验。尽管本书涵盖了很多方面，但它没有正式讲述HTML或Web页面设计。这些页面少量使用了一些CSS，但是这里没有讲授它。

其次，本书希望读者具有以下素质之一：

- 学习的动力和能力，而不需要被人牵着鼻子走；
- 熟悉另一种编程语言（具备丰富的JavaScript知识也足够了）；
- 对PHP有一定的了解。

本书涵盖了PHP和MySQL方方面面的内容，讲述了开发现实的Web站点需要知道的一切知识，不过，要特别指出的是，开头几章以较快的速度介绍了PHP。出于这种原因，我建议在开始学习新内容时，最好具备一些编程经验或者好奇和独立的精神。如果你发现有些内容讲得太快，那么从学习我的*PHP for the World Wide Web: Visual QuickStart Guide*的最新版本^①开始起步可能更好，其中的行文速度更适中。

学习本书不需要任何数据库经验，因为本书是从最基本的级别开始讨论SQL和MySQL的。

这一版本的新增内容

本书的前3个版本非常受欢迎，我收到了许多肯定的反馈意见（感谢！）。在编写此新版本时，我希望不仅仅是更新PHP和MySQL最新版本的内容，尽管这是全书的首要考虑事项。你将会发现其他新特性：

- 用新的示例演示读者迫切需要了解的技术；
- 另外一些高级MySQL和SQL示例；
- jQuery JavaScript框架使用手册；
- 介绍面向对象编程基础知识和基本语法；
- 提升脚本和网页安全性的更多信息和示例；
- 用一整章的篇幅专门介绍如何阻止常见的Web站点滥用和攻击；
- 用全新的一章介绍使用多种语言和时区；

^① 中文版《PHP基础教程（第4版）》，由人民邮电出版社出版。——编者注

- 用全新的一章示例介绍创建消息板（也称论坛）；
- 扩展和更新了安装和配置指导；
- 删除了过时的内容（例如，PHP的老版本中使用的特性或者不适用的特性）。
- 这一版在每一章的末尾新加入“回顾和实践”小节。该小节的目的是回顾那一章介绍的主要内容，给出一些练习，并以所学内容为基础进一步扩展相关知识。

对于买过本书前几版的读者（多谢啊），我相信这些新特性也会使这一版本成为你的案头必备。

与我的其他图书的比较

这是我编写的第四本关于PHP和/或MySQL主题的图书，下面按顺序列出之前出版的另外3本书：

- *PHP for the World Wide Web: Visual QuickStart Guide*
- *PHP 5 Advanced for the World Wide Web: Visual QuickPro Guide*
- *MySQL: Visual QuickStart Guide*

我希望这份履历暗示了我具有某种资格来编写本书，但是，作为读者，你该如何选择呢？当然，我非常欢迎你慷慨地购买全套书，对你的这种做法，我会致以无尽的谢意，但是，如果你不得不从中选择一本书……

*PHP for the World Wide Web: Visual QuickStart Guide*一书非常适合作为PHP的初学者指南。它与本书有一些重叠，大部分出现在前3章中，但是使用了新的示例，因此读者并不会觉得多余。对于初学者，可以在阅读那本书后再阅读本书。肯定应该在阅读了本书之后再阅读*PHP 5 Advanced for the World Wide Web: Visual QuickPro Guide*一书，因为它假定读者具有相当多的知识并且是在这里介绍的许多内容的基础上编写的。*MySQL: Visual QuickStart Guide*一书重点关注的几乎都是MySQL独有的内容（只有两章使用了PHP）。

在记住这些后，阅读“本书读者对象”一节的内容，并看看其中的要求是否适合你自己。如果你根本没有任何编程经验，并且更希望得到更细致的引导，那么我的第一本书可能更适合你。如果你已经对使用PHP感到得心应手，并且想学习它的更多高级功能，可选择第二本书。如果你最感兴趣的是MySQL，并且不怎么热衷于学习关于PHP的大量知识，那么可以购买第三本书。

如前所述，如果你想学习今天利用PHP和MySQL开始开发动态Web站点所需要知道的一切知识，那么本书就适合你！它涵盖了这两种技术的最新版本，使用了其他图书中以前未讨论的技术，并且包含了独具特色的示例。

无论你选择哪一本书，都要选择最新版本，或者与你将使用的技术最匹配的版本。

配套网站

我专门为本书开发了一个配套网站www.LarryUllman.com。^①在这里，你将找到本书中的每一个脚本、包含冗长的SQL命令的文本文件以及勘误表（如果你有关于命令或脚本方面的问题，并且正好在学习本书，请检查勘误表以确保没有印刷错误）。在该网站上，你还会发现有用的Web链接以及一个非常受欢迎的论坛，读者可以在里面相互问问题并进行解答（我自己解答了其中许多问题），等等。

^① 读者也可将中文版评论、勘误发到图灵社区本书的页面中，社区网址：ituring.com.cn。——编者注

问题、评论或建议

如果你有关于PHP或MySQL的任何问题，应该求助于现有的许多网站、邮件列表、新闻组和FAQ仓库。搜索引擎可以找到几乎无限的资源。如果你需要得到即时解答，这些资源或搜索引擎可以基本满足你的需要（十有八九已经有人遇到并解决了你的问题）。

你也可以直接把你的问题、评论和建议发送给我。你可以使用本书的相应论坛得到最快的答复（我总是优先解答这些问题）。如果你更喜欢给我发送电子邮件，网站上提供了我的联系方式。尽管我不能保证会迅速给出回复，但我会尽力答复收到的每封电子邮件。

出版社提示：查看作者Larry Ullman随附的教学视频

可视化快速入门教程（Visual QuickStart Guides）现在更加“直观”：由于Visual QuickStart Guides系列书籍非常畅销，Peachpit出版社提供了1个多小时的任务试教学视频，可以帮助你快速掌握关键的特性和技术。你可以一边阅读书中的PHP和MySQL脚本，一边看实战视频。这是一种非常理想的学习基础知识和新的、更复杂特性的方法。请访问Peachpit网站，你就可以看到一些免费的视频，也可以很方便地购买其他视频。

特别感谢

像以往一样衷心感谢Peachpit出版社的每一个人。

要感谢出色的编辑Rebecca Gulick，她使我的工作变得如此轻松。要感谢Patricia Pane，感谢他的努力工作、有益建议和令人印象深刻的对细节的关注。还要感谢制作索引的Valerie Haynes-Perry，设计本书版式的Myrna Vladic和Deb Roberti以及进行技术评审的Anselm Bradford。

在此感谢那些致力于PHP、MySQL、Apache、phpMyAdmin、MAMP、XAMPP及其他重大项目的优秀人士，以及工作在不同新闻组、邮件列表和支持论坛里的人士，他们为那些面临困境的人提供了帮助和建议。

仍然要感谢读者，你们的支持给了我工作的动力。另外还要感谢在第17章中提供消息板译文的读者，还有一些读者对本书应该包含的内容提出了建议，在此一并表示感谢。

感谢Karnesha和Sarah照料孩子，使得我可以全身心地投入工作。

最后，如果没有我的妻子Jessica的爱和支持，我将不能够单独完成一本书。我敢肯定，如果没有她，所有一切都会很糟糕。

目 录

第 1 章 PHP 概述	1
1.1 基本语法	1
1.2 发送数据到 Web 浏览器	5
1.3 编写注释	9
1.4 什么是变量	12
1.5 介绍字符串	15
1.6 连接字符串	17
1.7 数字介绍	19
1.8 常量介绍	22
1.9 单引号与双引号	24
1.10 基本的调试步骤	27
1.11 回顾和实践	28
1.11.1 回顾	29
1.11.2 实践	29
第 2 章 PHP 编程	30
2.1 创建 HTML 表单	30
2.2 处理 HTML 表单	34
2.3 条件语句和运算符	38
2.4 验证表单数据	42
2.5 介绍数组	47
2.5.1 创建数组	51
2.5.2 访问数组	52
2.5.3 多维数组	55
2.5.4 数组排序	60
2.6 for 和 while 循环	63
2.7 回顾和实践	66
2.7.1 回顾	66
2.7.2 实践	67
第 3 章 创建动态 Web 站点	68
3.1 包含多个文件	68
3.2 再论处理 HTML 表单	75
3.3 建立黏性表单	80
3.4 创建自己的函数	84
3.4.1 创建带参数的函数	86
3.4.2 设置默认的参数值	90
3.4.3 从函数返回值	93
3.5 回顾和实践	98
3.5.1 回顾	98
3.5.2 实践	98
第 4 章 MySQL 简介	99
4.1 命名数据库元素	99
4.2 选择列类型	100
4.3 选择其他的列属性	104
4.4 访问 MySQL	106
4.4.1 使用 MySQL 客户端	106
4.4.2 使用 phpMyAdmin	109
4.5 回顾和实践	113
4.5.1 回顾	113
4.5.2 实践	113
第 5 章 SQL 简介	114
5.1 创建数据库和表	114
5.2 插入记录	118
5.3 选择数据	122
5.4 使用条件语句	124
5.5 使用 LIKE 和 NOT LIKE	128
5.6 排序查询结果	129
5.7 限制查询结果	132
5.8 更新数据	133
5.9 删除数据	135

2 目录

5.10 使用函数	137	7.8 回顾和实践	217
5.10.1 文本函数	138	7.8.1 回顾	217
5.10.2 数字函数	141	7.8.2 实践	218
5.10.3 日期和时间函数	143	第 8 章 错误处理和调试	219
5.10.4 格式化日期和时间	146	8.1 错误类型与基本调试方法	219
5.11 回顾和实践	148	8.1.1 基本调试步骤	221
5.11.1 回顾	148	8.1.2 调试 HTML	224
5.11.2 实践	149	8.2 显示 PHP 错误	226
第 6 章 数据库设计	150	8.3 调整 PHP 中的错误报告	228
6.1 规范化	150	8.4 创建自定义的错误处理程序	232
6.1.1 键	151	8.5 PHP 调试技术	236
6.1.2 关系	152	8.6 SQL 和 MySQL 调试技术	239
6.1.3 第一范式	153	8.6.1 调试 SQL 问题	240
6.1.4 第二范式	155	8.6.2 调试访问问题	241
6.1.5 第三范式	157	8.7 回顾和实践	241
6.1.6 审查设计	159	8.7.1 回顾	241
6.2 创建索引	161	8.7.2 实践	242
6.3 使用不同的表类型	163	第 9 章 使用 PHP 和 MySQL	243
6.4 语言和 MySQL	165	9.1 修改模板	243
6.5 时区和 MySQL	170	9.2 连接到 MySQL	245
6.6 外键约束	175	9.3 执行简单的查询	249
6.7 回顾和实践	180	9.4 检索查询结果	257
6.7.1 回顾	180	9.5 确保 SQL 安全	262
6.7.2 实践	180	9.6 统计返回的记录	266
第 7 章 高级 SQL 和 MySQL	181	9.7 利用 PHP 更新记录	268
7.1 执行联结	181	9.8 回顾和实践	274
7.1.1 内联结	182	9.8.1 回顾	275
7.1.2 外联结	185	9.8.2 实践	275
7.1.3 联结三个或更多表	188	第 10 章 常用编程技术	276
7.2 分组选定的结果	191	10.1 给脚本发送值	276
7.3 高级选择	195	10.2 使用隐藏的表单输入框	280
7.4 执行 FULLTEXT 查找	200	10.3 编辑现有的记录	286
7.4.1 创建 FULLTEXT 索引	200	10.4 给查询结果标页码	293
7.4.2 执行基本的 FULLTEXT 查找	202	10.5 建立可排序的显示结果	300
7.4.3 执行布尔型 FULLTEXT 查找	204	10.6 回顾和实践	305
7.5 查询优化	207	10.6.1 回顾	305
7.6 执行事务	211	10.6.2 实践	306
7.7 数据库加密	214		

第 11 章 Web 应用程序开发	307	13.7.2 实践	412
11.1 发送电子邮件	307		
11.2 处理文件上传	313		
11.2.1 允许文件上传	313		
11.2.2 利用 PHP 上传文件	319		
11.3 PHP 和 JavaScript	325		
11.3.1 创建 JavaScript 文件	326		
11.3.2 创建 PHP 脚本	329		
11.4 理解 HTTP 头部	332		
11.5 日期和时间函数	339		
11.6 回顾和实践	343		
11.6.1 回顾	343		
11.6.2 实践	343		
第 12 章 cookie 和会话	345		
12.1 建立登录页面	345		
12.2 创建登录函数	348		
12.3 使用 cookie	353		
12.3.1 设置 cookie	354		
12.3.2 访问 cookie	358		
12.3.3 设置 cookie 参数	360		
12.3.4 删除 cookie	363		
12.4 使用会话	367		
12.4.1 设置会话变量	368		
12.4.2 访问会话变量	370		
12.4.3 删除会话变量	373		
12.5 提高会话安全性	376		
12.6 回顾和实践	379		
12.6.1 回顾	379		
12.6.2 实践	379		
第 13 章 安全性方法	381		
13.1 阻止垃圾邮件	381		
13.2 通过类型验证数据	388		
13.3 按类型验证文件	394		
13.4 阻止 XSS 攻击	398		
13.5 使用过滤器扩展	401		
13.6 预防 SQL 注入攻击	405		
13.7 回顾和实践	412		
13.7.1 回顾	412		
第 14 章 Perl 兼容的正则表达式	414		
14.1 创建测试脚本	414		
14.2 定义简单的模式	419		
14.3 使用量词	422		
14.4 使用字符类别	424		
14.5 查找所有匹配	427		
14.6 使用修饰符	432		
14.7 匹配和替换模式	434		
14.8 回顾和实践	438		
14.8.1 回顾	438		
14.8.2 实践	438		
第 15 章 jQuery 简介	439		
15.1 jQuery 是什么	439		
15.2 包含 jQuery	441		
15.3 使用 jQuery	444		
15.4 选择页面元素	446		
15.5 事件处理	449		
15.6 DOM 操作	453		
15.7 使用 Ajax	458		
15.7.1 创建表单	459		
15.7.2 创建服务器端脚本	461		
15.7.3 处理 Ajax 请求	462		
15.7.4 创建 JavaScript	464		
15.8 回顾和实践	470		
15.8.1 回顾	470		
15.8.2 实践	470		
第 16 章 面向对象编程入门	472		
16.1 基础知识和语法	472		
16.1.1 面向对象的基础	472		
16.1.2 PHP 中的 OOP 语法	473		
16.2 使用 MySQL	475		
16.2.1 创建连接	475		
16.2.2 执行简单的查询	478		
16.2.3 获取结果	482		
16.2.4 预处理语句	486		
16.3 DateTime 类	490		

16.4 回顾和实践.....	497	18.5 激活账户	564
16.4.1 回顾.....	497	18.6 登录和注销.....	567
16.4.2 实践.....	498	18.7 密码管理	573
第 17 章 示例——论坛	499	18.7.1 重置密码.....	573
17.1 建立数据库.....	499	18.7.2 更改密码.....	578
17.2 编写模板.....	508	18.8 回顾和实践.....	583
17.3 创建索引页面.....	516	18.8.1 回顾.....	583
17.4 创建论坛页面.....	517	18.8.2 实践.....	583
17.5 创建论点页面.....	522		
17.6 发布消息.....	526		
17.6.1 创建表单	526		
17.6.2 处理表单	531		
17.7 回顾和实践.....	537		
17.7.1 回顾.....	537		
17.7.2 实践.....	538		
第 18 章 示例——用户注册	539		
18.1 创建模板.....	539	19.1 创建数据库.....	584
18.2 编写配置脚本.....	545	19.2 管理端	590
18.2.1 建立配置文件	545	19.2.1 添加艺术家	591
18.2.2 建立数据库脚本	549	19.2.2 添加印刷品	596
18.3 创建主页.....	553	19.3 创建公共模板	606
18.4 注册	554	19.4 产品目录	609
		19.5 购物车	621
		19.5.1 添加项目	621
		19.5.2 查看购物车	625
		19.6 记录订单	631
		19.7 回顾和实践	637
		19.7.1 回顾.....	637
		19.7.2 实践.....	637

第1章

PHP概述



本章内容

- 基本语法
- 发送数据到Web浏览器
- 编写注释
- 什么是变量
- 介绍字符串
- 连接字符串
- 介绍数字
- 介绍常量
- 单引号与双引号
- 基本的调试步骤
- 回顾和实践

尽管本书重点关注的是组合使用MySQL和PHP，但是你将单独使用PHP执行动态Web站点的大量基础工作。在本章和下一章中，将学习PHP的基础知识，从语法到变量、运算符和语言构造（条件构造、循环构造等等）。在你学习这些基础知识的同时，还将开始开发有用的代码，在本书后面将把这些代码集成到更大的应用程序中。

在这一章中，我将泛泛地介绍PHP Web脚本语言的大多数基础知识。你将学习到PHP编码的语法，如何发送数据给Web浏览器，以及如何使用两类变量（字符串和数字）和常量。有些示例看起来似乎没太大用处，但是，它们将说明一些必须掌握的概念，有了这些，你才能编写更高级的脚本。

1.1 基本语法

如同我在本书的前言中所提及的，PHP是一种嵌入在HTML中的（HTML-embedded）脚本语言。这意味着可以把PHP代码和HTML代码混合在相同的文件内。我们的PHP编程首先从一个简单的Web页面开始。脚本1-1给出了一个最简单的XHTML过渡型文档^①的示例，我将把它用作本书中每个Web页面的基础（本书没有正式讨论[X]HTML，参见专门讨论这个主题的资源以了解更多信息）。

^① XHTML标准定义了三种文档：严格型、过渡型和框架型。——编者注

脚本 1-1 基本 XHTML 1.0 过渡型 Web 文档

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
2  DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4      <head>
5          <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6          <title>Page Title</title>
7      </head>
8      <body>
9          <!-- Script 1.1 - template.html -->
10     </body>
11 </html>
```

要把PHP代码添加到页面中，可把它置于PHP标签内：

```
<?php
?>
```

放置在这些标签内的任何内容都会被Web服务器视作PHP代码（这意味着PHP解释器将处理这些代码）。PHP标签外的任何文本会立即作为常规HTML发送给Web浏览器。（由于PHP通常用来创建Web浏览器中显示的内容，因此PHP标签通常放在页面body中的某个地方。）

除了把PHP代码置于PHP标签内之外，PHP文件还必须使用正确的扩展名。扩展名告诉服务器以特殊的方式（即作为PHP页面）处理脚本。大多数Web服务器都为标准HTML页面使用.html或.htm扩展名，通常PHP文件首选扩展名为.php。

在开始之前，你必须已经有一个可用的PHP安装！在完成附录A“安装”之后，它会在你的托管网站或者你自己的机器上，附录A可以在<http://www.peachpit.com>免费下载。

理解字符编码

字符编码是一项极为复杂的主题，但当前最需要理解的是：在某文件中所使用的字符编码指示文件将显示什么字符（因此也同时指明了可以使用哪种语言）。在选择一种字符编码之前，必须首先确定所使用的文本编辑器或IDE（即创建HTML和PHP脚本的应用程序）能够使用哪种字符编码保存文档。一些应用程序可以让你在偏好设置或者选项中设置字符编码，而另外一些应用程序则只有当保存文件时才可设置字符编码。

可以利用相应的元标签为Web浏览器指示页面的字符编码：

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

charset=utf-8这一部分表明使用的是UTF-8字符编码，这是8位Unicode转换格式（8-bit Unicode Transformation Format）的简写。Unicode是一种表示所有字母和符号的可靠方式。在撰写本书时，第6版的Unicode能支持超过99 000个字符！

如果你想创建一个能适应多语种的Web页面，UTF-8正是解决之道，我将在本书的示例中使用它。当然你没有必要非得这么做。但是无论使用何种编码，请一定要确保XHTML页面所指定的编码与文本编辑器或IDE的编码设置一致，否则你可能会在浏览页面时看到一些奇怪的字符。

HTML5

在撰写本书时，HTML未来的主要版本——HTML5——还在积极地开发与讨论中，还未最后定案，因此本书这一版中没有选用它。事实上，如果HTML5在我开始编写本书的第5版时仍然未能发布，我也不不会感到惊讶。要想让大部分浏览器接受它还需要相当长的一段时间。尽管如此，但HTML5仍可视为未来发展的一个热点，所以本书将会偶尔介绍一些有望被浏览器逐渐引入并支持的HTML5功能。

建立基本的PHP脚本

(1) 在文本编辑器或集成开发环境中创建一个新文档，命名为first.php（参见脚本1-2）。

脚本 1-2 第一个 PHP 脚本本身不做任何事情，但是足以说明如何编写 PHP 脚本。在开始涉及更复杂的 PHP 代码之前，它还可用作测试

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Basic PHP Page</title>
6  </head>
7  <body>
8      <!-- Script 1.2 - first.php -->
9      <p>This is standard HTML.</p>
10 <?php
11 ?>
12 </body>
13 </html>
```

一般可以任意选择用什么文本编辑器或集成开发环境，它可以是Dreamweaver（一种优良的IDE）、BBEdit（一种优秀的、流行的Macintosh纯文本编辑器）或vi（一种纯文本UNIX编辑器，缺少图形界面）。然而，使用有些文本编辑器和IDE输入和调试HTML和PHP代码会更容易（与之相反，Windows上的“记事本”会使编码更困难）。如果你还没有找到自己喜欢的应用程序，可搜索Web或者使用本书的配套论坛（www.LarryUllman.com/forums）找到一个。

(2) 开始编写基本的HTML文档。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Basic PHP Page</title>
</head>
<body>
    <!-- Script 1.2 - first.php -->
    <p>This is standard HTML.</p>
</body>
</html>
```

尽管这是我在整本书中使用的语法，但是，你可以更改HTML以匹配你打算使用的任何一种标准（例如，HTML 4.0严格型）。同样，如果你不熟悉这段HTML代码，可以查看专门的(X)HTML资源（参见第一个提示）。

(3) 在body结束标签之前，插入PHP标签。

```
<?php  
?>
```

这些是正式的PHP标签，也称为XML风格的标签。尽管PHP支持其他标签类型，我还是建议你使用正式的PHP标签，并且在整本书中我也会这样做。

(4) 将文件另存为first.php。

记住，如果没有使用合适的PHP扩展名保存文件，该脚本将不会正确执行。（不使用记事本创建PHP文件的原因这一就是，它会默默地给文件加上.txt后缀，从而导致很多头痛的问题。）

(5) 将该文件置于Web服务器上正确的目录中。

如果你在自己的计算机上运行PHP（假定在遵照本书附录A中的安装指导进行了安装之后），你只需要将文件移动、复制或保存到计算机上的特定文件夹中。如果你还不知道该目录是什么，可以查看所用Web服务器应用程序的文档，以找到该目录。

如果在托管服务器上（即在远程计算机上）运行PHP，就需要使用FTP应用程序将文件上传到正确的目录中。你的托管公司将给你提供访问权限及其他必要的信息。

(6) 在Web浏览器中运行first.php（参见图1-1）。

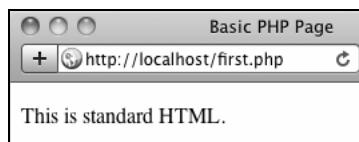


图1-1 虽然它看起来像任何其他（简单的）HTML页面一样，事实上，这是一个PHP脚本，它是本书中其余示例的基础

由于PHP脚本需要由服务器解析，你肯定必须通过URL访问PHP脚本（如浏览器中的地址必须以http://开头）。你不能像在其他应用程序中打开一个文件那样在Web浏览器中简单地打开它们（比如以file://或c:\等开头的地址）。

如果在自己的计算机上运行PHP，将需要到达http://localhost/first.php、http://127.0.0.1/first.php或http://localhost/~<user>/first.php（在Mac OS X上，为<user>使用实际的用户名）。如果正在使用一台Web主机，那么需要使用http://你的域名/first.php（例如，http://www.example.com/first.php）。

(7) 如果没有看到如图1-1所示的结果，可以开始进行调试。

学习任何编程语言的一个必不可少的部分是掌握调试方法。它是一个有时非常痛苦却绝对必要的过程。对于第一个示例，如果你没有看到一个简单但完全有效的Web页面，可遵循以下步骤：

(1) 确认你正确安装了PHP（参见本书附录A，了解测试指导）。

(2) 确保你正在通过URL运行脚本。Web浏览器中的地址必须以http://开始。如果它以file://开始，就会出现问题（参见图1-2）。

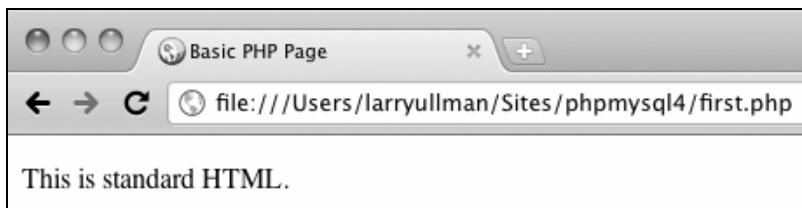


图1-2 PHP代码只能通过http://运行（否则这个特定脚本就不会起作用）

(3) 如果得到一个未找到文件（或类似）的错误，很可能是把文件放在错误的目录中或者输入了错误的文件名（在保存它时或者在Web浏览器中）。

如果你执行了所有这些步骤但是仍然有问题，可以求助于本书相应的论坛（www.LarryUllman.com/forums/）。

✓ 提示

- 有关HTML和XHTML的详细信息，可以查看Elizabeth Castro写的好书《HTML XHTML CSS基础教程（第7版）》（人民邮电出版社）或者搜索Web。
- 你可以在一个HTML文档中嵌入PHP代码的多个部分（即可以在两种语言之间来回转换）。在整本书中都可以看到这样的示例。
- 在UTF-8之前，最常用的字符编码之一是ISO-8859-1。它可以显示几乎全部西欧语言。目前，很多Web浏览器和其他应用仍在使用它作为默认字符编码。
- 你可以通过在文件首行添加@charset "utf-8"来为外部的CSS文件声明字符编码；如果用的不是UTF-8，那就将其改为相应的字符编码。

1.2 发送数据到 Web 浏览器

要利用PHP构建动态Web站点，必须知道如何发送数据到Web浏览器。PHP具有许多用于此目的的内置函数，其中最常用的是echo和print。我个人倾向于使用echo：

```
echo 'Hello, world!';
echo "What's new?";
```

如果喜欢的话，可以代之以print()：

```
print 'Hello, world!';
print "What's new?";
```

从这些示例可以看出，可以使用单引号或双引号（但是这两种引号之间有区别，到本章末尾就可以清楚看出它们之间的区别）。函数名后面的第一个引号指示要打印的消息的开头。下一个匹配的引号（即与左引号相同类型的右引号）指示要打印的消息的末尾。

除了学习如何发送数据到Web浏览器之外，还要注意在PHP中，所有语句（用外行的话讲就是一行执行代码）都必须以分号结尾。此外，在涉及函数名时PHP是不区分大小写的，因此ECHO、echo和eCHo等都会工作。当然，所有小写版本更容易输入。

发送数据到Web浏览器

- (1) 在文本编辑器或IDE中打开first.php (参见脚本1-2)。
- (2) 在PHP标签 (第10行和第11行) 之间添加一条简单的消息 (参见脚本1-3)。

脚本 1-3 PHP 可以使用 print 或 echo 发送数据到 Web 浏览器

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
   DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Using Echo</title>
6  </head>
7  <body>
8      <!-- Script 1.3 - second.php -->
9      <p>This is standard HTML.</p>
10     <?php
11 echo 'This was generated using PHP!';
12 ?>
13     </body>
14     </html>

```

在此，对于你在这里输入的消息内容，使用的是哪个函数 (echo或print) 或者哪种引号，都是无关紧要的——只是在打印单引号或双引号作为消息的一部分时，要小心谨慎 (参见框注“需要转义”)。

需要转义

你可能已经发现，发送数据到Web比较复杂的一个地方是打印单引号和双引号。下面两种方式都会引发错误：

```
echo "She said, "How are you?";  
echo 'I'm just ducky.';
```

对于这个问题有两种解决方案。首先，在打印双引号时使用单引号，反之亦然：

```
echo 'She said, "How are you?"';  
echo "I'm just ducky.:";
```

或者，通过在有问题的字符前面放置一个反斜杠，对它进行转义：

```
echo "She said, \"How are you?\\"";  
print 'I\'m just ducky.';
```

转义的引号只会像任何其他字符一样打印。理解如何使用反斜杠对字符进行转义是一个重要的概念，我将在本章末尾更深入地介绍这个概念。

- (3) 你可以按自己的想法更改页面标题，以便更好地描述这个页面 (第5行)。

```
<title>Using Echo</title>
```

这种改变只会影响浏览器窗口的标题栏。

- (4) 将文件另存为second.php，存放在Web目录中，然后在Web浏览器中测试它 (参见图1-3)。

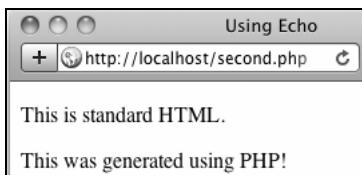


图1-3 结果仍然不怎么迷人，但是这个页面有一部分是PHP动态生成的

注意，所有PHP脚本都必须通过URL (`http://something`) 执行！

(5) 如果必要，可调试脚本。

如果你看到的是解析错误，而不是你的消息（参见图1-4），请检查你是否同时具有左引号和右引号，以及对任何有问题的字符转义（参见框注“需要转义”）。还要肯定每一条语句都用分号结尾。



图1-4 这可能是PHP程序员看到的许多解析错误中的第一个错误
(这个错误是由未转义的引号引起的)

如果看到的是完全空白的页面，这可能是由于下面两个原因之一引起的。

□ HTML代码有问题。通过查看页面的源文件来测试它，并寻找其中的HTML问题（参见图1-5）。



图1-5 看到空白PHP页面的一个常见原因是简单的HTML错误，
如这里的结束title标签（少了斜杠）

□发生一个错误，但是你的PHP配置中关闭了`display_errors`，因此，不会显示任何内容。在这种情况下，参见本书附录A中关于如何配置PHP一节的内容，以便能够再打开`display_errors`。

✓ 提示

- 严格说来，echo和print是语言构造，而不是函数。顺便说一下，当我为了方便而继续称之为“函数”时，不要感到奇怪。此外，在指函数时，我用了一对括号（比如number_format()，而不仅仅是number_format），从而有助于把它们与变量以及PHP的其他部分区分开。这只是我自己的一个小小习惯。
- 你通常也可以使用echo和print发送HTML代码到Web浏览器，其方法如下（参见图1-6）。

```
echo '<p>Hello, <b>world</b>!</p>';
```

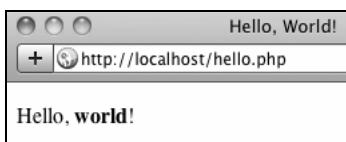


图1-6 PHP可以发送HTML代码（带有类似于这里的格式）
以及简单的文本（参见图1-3）到Web浏览器

- echo和print都可用于打印多行文本：

```
echo 'This sentence is
printed over two lines.';
```

- 在这里所发生的事情是：回车符（通过按Enter键或Return键创建）将变成打印消息的一部分，它终止于闭合单引号。其实际效果将是在HTML源代码中“打印”回车符（参见图1-7）。这不会影响生成的页面（参见图1-8）。有关这方面的更多信息，参见框注“理解空白”。

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
3 <head>
4   <meta http-equiv="Content-Type"
content="text/html; charset=utf-8"/>
5   <title>Hello, World!</title>
6 </head>
7 <body>
8   <!-- Script 1.3 - second.php -->
9 This sentence is
10 printed over two lines.</body>
11 </html>

```

图1-7 PHP源代码中的多行文本和HTML也会产生多行HTML源代码。注意：
HTML中无关紧要的空白不会影响页面的外观（参见图1-8），但是可
能使源文件更易于查阅



图1-8 HTML源代码中的回车符（参见图1-7）不会影响呈现的结果。改变显示的Web页面间距的唯一方式是使用HTML标签（如`
`和`<p></p>`）

理解空白

利用PHP，可以发送数据（像HTML标签和文本）到Web浏览器。接下来，Web浏览器可以将其显示为最终用户查看的Web页面。因此，使用PHP其实就是在创建Web页面的HTML源文件（HTML source）。记住这一点，实质上可以在3个地方产生引人注目的空白（white space）（额外的空格、制表位以及空白行）：在PHP脚本中、在HTML源文件中以及在呈现的Web页面中。

PHP一般会忽略空白，这意味着你可以在代码中增加空白，使自己的脚本更易读。HTML一般也会忽略空白。确切地讲，HTML中会影响页面的唯一空白是单个空格（多个空格仍然作为一个空格呈现）。如果HTML源文件中有许多行的文本，这并不意味着在呈现的页面中它将出现在多行上（参见图1-7和图1-8）。

要改变呈现的Web页面的间距，可以使用HTML标签`
`（换行符，在HTML老标准中是`
`）和`<p></p>`（段落）。要改变用PHP创建的HTML源文件的间距，可以：

- 在多个行上使用`echo()`或`print()`；
- 或者
- 在双引号内打印换行符（`\n`）。

1.3 编写注释

创建可执行的PHP代码只是编程过程的一部分（诚然，它是最重要的部分）。动态Web站点开发的一个次要但仍然至关重要的工作是为你的代码加注释。实际上，当被问及新手程序员与老头的区别时，我的回答始终是能否写出好的、详尽的注释。

在HTML中，可以使用特殊标签添加注释：

```
<!-- Comment goes here. -->
```

HTML注释在源文件中可以看到，但是不会出现在呈现的页面中（参见图1-7和图1-8）。

PHP注释则有所不同，因为它们根本不会被发送到Web浏览器，这意味着最终用户不会看到它们，即使查看HTML源文件也是如此。

PHP支持3种注释类型。第一种使用磅即编号符号（#）。

```
# This is a comment.
```

第二种使用两个斜杠。

```
// This is also a comment.
```

这两种注释都会使PHP忽略其后直到行末（当你按Return或Enter键时）的一切内容。因此，这两种注释都只是单行注释。它们还常用于在PHP代码行内添加注释。

```
print 'Hello!'; // Say hello.
```

第三种风格允许注释分布在多行上。

```
/* This is a longer comment
that spans two lines. */
```

给脚本加注释

(1) 在文本编辑器或IDE中新建一个PHP文档，命名为comments.php（参见脚本1-4）。

脚本 1-4 这些基本的注释演示了在 PHP 中可以使用的 3 种语法

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Comments</title>
6  </head>
7  <body>
8  <?php
9
10 # Script 1.4 - comments.php
11 # Created March 16, 2011
12 # Created by Larry E. Ullman
13 # This script does nothing much.
14
15 echo '<p>This is a line of text.<br /> This is another line of text.</p>';
16
17 /*
18 echo 'This line will not be executed.';
19 */
20
21 echo "<p>Now I'm done.</p>"; // End of PHP code.
22
23 ?>
24 </body>
25 </html>
```

(2) 添加初始PHP标签，并且编写你的第一条注释。

```
<?php
# Script 1.4 - comments.php
# Created March 16, 2011
# Created by Larry E. Ullman
# This script does nothing much.
```

每个脚本都应该包含的最初几条注释之一是一个介绍性的块，其中列出了创建日期、修改日期、创建者、创建者的联系信息、脚本的目的，等等。有些人认为shell风格的注释（#）在脚本中更醒目，因此这种注释是最佳的。

(3) 将一些HTML代码发送到Web浏览器。

```
echo '<p>This is a line of text.<br />This is another line of text.</p>';
```

它与你在这里做什么无关，而只是让Web浏览器显示一些内容。出于改变显示内容的目的，我使用echo()语句打印一些HTML标签，包括换行符（
），在生成的HTML页面中添加一些间距。

(4) 使用多行注释来注释掉第二条echo语句。

```
/*
echo 'This line will not be executed.';
*/
```

用/*和*/来包围任何PHP代码块，可以使代码不起作用，而不必将其从脚本中删除。之后可通过删除注释标签，重新激活那部分PHP代码。

(5) 在最后一条echo语句后面添加最后一条注释。

```
echo "<p>Now I'm done.</p>"; // End of PHP code.
```

这条最后的（有些多余的）注释说明了如何把一条注释放在一行的末尾，这是一种常见的惯例。注意：我使用双引号包围消息，因为单引号会与撇号冲突（参见本章前面的框注“需要转义”）。

(6) 关闭PHP部分并完成HTML页面。

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

(7) 将该文件另存为comments.php，存放到Web目录中，然后在Web浏览器中测试它（参见图1-9）。

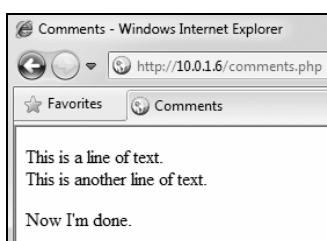


图1-9 脚本1-4中的PHP注释不会出现在Web页面或HTML源文件中（参见图1-10）

(8) 有好奇心的读者可以在Web浏览器中检查源代码，确认没有出现PHP注释（参见图1-10）。

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
2   Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4     <head>
5       <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6       <title>Comments</title>
7     </head>
8     <body>
9       <p>This is a line of text.<br />This is another line of text.</p><p>Now I'm
done.</p></body>
10    </html>
```

图1-10 在客户端浏览器中查看脚本1-4中的PHP注释

✓ 提示

- 不应该嵌套多行注释（`/* */`）（把一条注释放在另一条注释内），因为这会引起问题。
- 在行末（比如，在函数调用后面）可以使用任何PHP注释：


```
echo 'Howdy'; /* Say 'Howdy' */
```

尽管也可以用`/*`和`*/`，但一般不这样做。
- 对脚本进行注释几乎可以说是多多益善。话虽如此，为了节省篇幅，本书中的脚本并没有按我建议的那样加足够的注释。
- 在修改脚本时保持注释是最新的和准确的也很重要。没有什么比注释说的是一套，而代码实际上做的是另一套更让人迷糊的了。

1.4 什么是变量

变量是用于临时存储值的容器。这些值可以是数字、文本，或者是复杂得多的数据。PHP具有8种变量。其中包括4种标量（单值）类型——布尔型（TRUE或FALSE）、整型、浮点型（小数）和字符串型（字符），两种非标量（多值）类型——数组和对象，以及资源（当与数据库交互时将看到它）和NULL（它是一种不具有任何值的特殊类型）。

不管创建什么类型，PHP中的所有变量都遵循某种特定的语法规则。

- 变量的名称——也称为它的标识符——必须以美元符号（\$）开头，例如，`$name`。
- 变量名称可以包含字母、数字和下划线的组合，例如，`$my_report1`。
- 美元符号之后的第一个字符必须是字母或下划线（不能是数字）。
- PHP中的变量名称是区分大小写的。这是一个非常重要的规则。这意味着`$name`和`$Name`是截然不同的变量。

为了开始使用变量，我将利用几个预定义的变量，在运行PHP脚本时会自动设置它们的值。在深入介绍这个脚本之前，应该知道另外两件事情。第一，可以使用等于号（=）（也称为赋值运算符（assignment operator））给变量赋值。第二，无需引号即可打印变量：

```
print $some_var;
```

或者可以在双引号内打印变量：

```
print "Hello, $name";
```

但不能在单引号内打印变量：

```
print 'Hello, $name'; // Won't work!
```

使用变量

(1) 在文本编辑器或IDE中创建新的PHP文档，命名为`predefined.php`（参见脚本1-5）。

脚本 1-5 这个脚本会打印 3 个 PHP 预定义的变量

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

```

3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5     <title>Predefined Variables</title>
6   </head>
7   <body>
8   <?php # Script 1.5 - predefined.php
9
10 // Create a shorthand version of the variable names:
11 $file = $_SERVER['SCRIPT_FILENAME'];
12 $user = $_SERVER['HTTP_USER_AGENT'];
13 $server = $_SERVER['SERVER_SOFTWARE'];
14
15 // Print the name of this script:
16 echo "<p>You are running the file:<br /><b>$file</b>.</p>\n";
17
18 // Print the user's information:
19 echo "<p>You are viewing this page using:<br /><b>$user</b></p>\n";
20
21 // Print the server's information:
22 echo "<p>This server is running: <br /><b>$server</b>.</p>\n";
23
24 ?>
25 </body>
26 </html>

```

(2) 添加PHP开始标签和第一条注释。

```
<?php # Script 1.5 - predefined.php
```

从现在起，我的脚本将不再添加关于创建者、创建日期等的注释，尽管你应该继续这样做。不过，我将添加一个注释，列出脚本编号和文件名，以便进行交叉引用（在本书中，以及当你从本书的支持Web站点www.LarryUllman.com上下载它们时）。

(3) 创建脚本中要使用的第一变量的简写版本。

```
$file = $_SERVER['SCRIPT_FILENAME'];
```

这个脚本将使用3个变量，它们都来自更大的、预定义的\$_SERVER变量。\$_SERVER指大量与服务器相关的信息。该脚本要使用的第一变量是\$_SERVER['SCRIPT_FILENAME']。这个变量存储要运行的脚本的完整路径和名称（例如，C:\Program Files\Apache\htdocs\predefined.php）。

\$_SERVER['SCRIPT_FILENAME']中存储的值将被赋予新变量\$file。用更短的名称创建新变量，然后从\$_SERVER给它们赋值，这将使得在打印这些变量时更容易引用它们（它还避开了你将适时学到的其他一些问题）。

(4) 创建另外两个变量的简写版本。

```
$user = $_SERVER['HTTP_USER_AGENT'];
$server = $_SERVER['SERVER_SOFTWARE'];
```

\$_SERVER['HTTP_USER_AGENT']代表访问脚本的用户的Web浏览器和操作系统。该值将被赋予\$user。\$_SERVER['SERVER_SOFTWARE']代表运行PHP的服务器上的Web应用程序（例如，Apache、Abyss、Xitami、IIS）。必须安装这个程序（参见附录A），以便在计算机上运行PHP脚本。

(5) 打印出将被运行的脚本的名称。

```
echo "<p>You are running the file:<br /><b>$file</b>.</p>\n";
```

将要打印的第一个变量是\$**file**。注意：这个变量必须打印在双引号内，我还利用PHP换行符(\n)，它将在生成的HTML源文件中添加一个分行符。还会添加一些基本的HTML标签（段落和加粗），从而给生成的页面增添一些优雅的风格。

(6) 打印出访问脚本的用户的信息。

```
echo "<p>You are viewing this page using:<br /><b>$user</b></p>\n";
```

这一行代码打印第二个变量\$user。为了重复第(4)步中所做的工作，使\$user与\$_SERVER['HTTP_USER_AGENT']相关联，并引用正访问Web页面的操作系统、浏览器类型和浏览器版本。

(7) 打印出服务器信息。

```
echo "<p>This server is running: <br /><b>$server</b>.</p>\n";
```

(8) 完成HTML和PHP代码。

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

(9) 将文件另存为**predefined.php**，存放在Web目录中，并在Web浏览器中测试它（参见图1-11）。

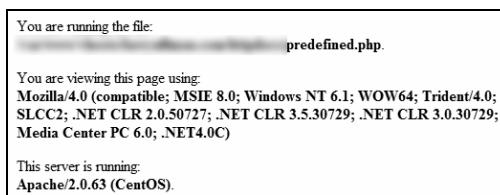


图1-11 predefined.php脚本向浏览器报告关于脚本、正用于查看它的Web浏览器以及服务器自身的信息

✓ 提示

- 如果你对这个脚本或其他任何脚本有任何疑问，可以向本书相应的Web论坛（[www.LarryUllman.com/forums/](http://www.LarryUllman.com/)）求助，以获取帮助。
- 如果有可能，使用不同的Web浏览器和/或在另一个服务器上运行这个脚本（参见图1-12）。

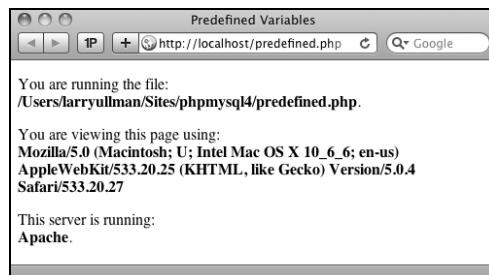


图1-12 这是本书中第一个真正动态的脚本，这是由于Web页面依赖于运行它的服务器和查看它的Web浏览器而变化（对比图1-11）

- 创建变量时最重要的考虑事项是使用一致的命名模式。在本书中，你将看到变量名称全都用的是小写字母，并且用下划线分隔单词（\$first_name）。有些程序员更喜欢使用词首大写字母，例如：\$FirstName（即驼峰式拼写法）。
- PHP处理变量的方式非常随意，这意味着你不必初始化它们（设置一个即时值）或声明它们（设置具体的类型），并且你可以在多种类型之间转换一个变量，而不会引发任何问题。

1.5 介绍字符串

我将深入讨论的第一种变量类型是字符串。字符串只是一块用引号括起来的字符：字母、数字、空格、标点符号，等等。下面列出的全都是字符串：

- ‘Tobias’
- “In watermelon sugar”
- ‘100’
- ‘August 2, 2011’

为了建立一个字符串变量，可以给一个有效的变量名赋予一个字符串值：

```
$first_name = 'Tobias';
$today = 'August 2, 2011';
```

在创建字符串时，可以使用单引号或双引号封装字符，就像在打印文本时所做的那样。此外，必须在字符的开头和末尾使用相同类型的引号。如果在字符串中间出现相同的引号，就必须对它进行转义：

```
$var = "Define \"latitude\", please.;"
```

另一种方式是使用单引号：

```
$var = 'Define "latitude", please.';
```

为了打印字符串的值，可以使用echo或print：

```
echo or print;
echo $first_name;
```

为了在某种环境内打印出字符串的值，可以使用双引号：

```
echo "Hello, $first_name";
```

你已经使用了一次字符串——在上一节中使用预定义的变量时。在下一个示例中，将创建和使用新的字符串。

使用字符串

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为Strings.php。以HTML和PHP开始标签开头（参见脚本1-6）。

脚本 1-6 在这个介绍性的脚本中创建字符串变量并把它们的值发送给 Web 浏览器

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

```

3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5     <title>Strings</title>
6   </head>
7   <body>
8   <?php # Script 1.6 - strings.php
9
10 // Create the variables:
11 $first_name = 'Haruki';
12 $last_name = 'Murakami';
13 $book = 'Kafka on the Shore';
14
15 // Print the values:
16 echo "<p>The book <em>$book</em> was written by $first_name $last_name.</p>";
17
18 ?>
19 </body>
20 </html>

```

(2) 在PHP标签内，创建3个变量。

```

$first_name = 'Haruki';
$last_name = 'Murakami';
$book = 'Kafka on the Shore';

```

这个基本的示例创建了\$first_name、\$last_name和\$book这3个变量，随后将会在一条消息中打印出它们。

(3) 添加echo语句。

```
echo "<p>The book <em>$book</em> was written by $first_name $last_name.</p>";
```

这个脚本所做的全部工作是基于已建立的3个变量打印出一份作者身份声明。其中插入了很少的HTML格式化效果（强调书名的斜体），使之更吸引人。记住在这里为要打印的变量值相应地使用双引号（在本章末尾将更详细地讨论双引号的重要性）。

(4) 完成HTML和PHP代码。

```

?>
</body>
</html>

```

(5) 将该文件另存为strings.php，存放在Web目录中，并在Web浏览器中测试它（参见图1-13）。

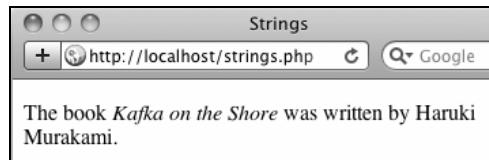


图1-13 得到的Web页面基于打印出的3个变量的值

(6) 如果需要，可更改3个变量的值，保存文件并再次运行脚本（参见图1-14）。



图1-14 通过改变脚本中的变量来改变脚本的输出

✓ 提示

□ 如果把另一个值赋予现有的变量（比如：\$book），新的值就会重写旧的值。例如：

```
$book = 'High Fidelity';
$book = 'The Corrections';
/* $book now has a value of
'The Corrections'. */
```

□ PHP没有对一个字符串的可能取值设置上限。理论上讲，它可能会受到服务器资源的限制，但是，不一定会碰到这样的问题。

1.6 连接字符串

连接（concatenation）像是为字符串增加的一种功能，通过它把字符添加到字符串的末尾。可以使用连接运算符（concatenation operator）即句点（.）来执行它：

```
$city= 'Seattle';
$state = 'Washington';
$address = $city . $state;
```

\$address变量的值现在是SeattleWashington，这几乎得到了期待的结果（Seattle, Washington）。为了对其进行改进，可以编写：

```
$address = $city . ', ' . $state;
```

从而将逗号和空格添加到字符串混合中。

连接还可以处理字符串或数字。下面两条语句将会产生相同的结果（Seattle, Washington 98101）：

```
$address = $city . ', ' . $state . ' 98101';
$address = $city . ', ' . $state . ' ' . 98101;
```

我将修改strings.php脚本以使用这个新的运算符。

使用连接

- (1) 在文本编辑器或IDE中打开strings.php（参见脚本1-6）。
- (2) 在创建了\$first_name和\$last_name变量之后（第11、12行），添加如下一行代码（参见脚本1-7）。

脚本 1-7 连接可将更多的字符追加到一个字符串中

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
```

```

2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5   <title>Concatenation</title>
6 </head>
7 <body>
8 <?php # Script 1.7 - concat.php
9
10 // Create the variables:
11 $first_name = 'Melissa';
12 $last_name = 'Bank';
13 $author = $first_name . ' ' . $last_name;
14
15 $book = 'The Girls\' Guide to Hunting and Fishing';
16
17 //Print the values:
18 echo "<p>The book <em>$book</em> was written by $author.</p>";
19
20 ?>
21 </body>
22 </html>

```

如连接示范的那样，将会创建一个新变量——\$author，作为两个现有字符串和它们之间一个空格的连接。

(3) 更改echo语句，使用这个新变量。

```
echo "<p>The book <em>$book</em> was written by $author.</p>";
```

因为把两个变量变成了一个变量，所以应该相应地改变echo语句。

(4) 如果需要，可以更改HTML页面标题，以及名字、姓氏和图书这几个变量的值。

(5) 将文件另存为concat.php，存放在Web目录中，并在Web浏览器中测试它（参见图1-15）。



图1-15 在这个修改过的脚本中，连接的最终结果对用户并不是显而易见的

✓ 提示

□ PHP具有许多专用于字符串的函数，你将在学习本书的过程中看到它们。例如，为了计算一个字符串有多长（它包含多少个字符），可以使用strlen()：

```
$num = strlen('some string'); // 11
```

□ PHP可以利用几个函数转换字符串的大小写：strtolower()，把字符串全都变为小写；strtoupper()，把字符串全都变为大写；ucfirst()，第一个字符大写；ucwords()，每个单词的第一个字符大写。

- 如果只把一个值连接到另一个值，则可以使用连接赋值运算符（concatenation assignment operator）（`=.`）。下面两条语句是等价的：

```
$title = $title . $subtitle;
$title .= $subtitle;
```

- 本节开始的示例也可重写成：

```
$address = "$city, $state";
```

或

```
$address = $city;
$address .= ', ';
$address .= $state;
```

使用PHP手册

在<http://www.php.net/manual>中可在线访问的PHP手册，列出了语言所有的函数和特性。该手册按照使用PHP的先后顺序进行组织，先讨论一般性概念（安装、语法、变量），最后是专题函数（MySQL、字符串函数等）。

要在PHP手册中快速查找函数，在浏览器中访问www.php.net/functionname（例如：<http://www.php.net/print>）。PHP手册为每个函数描述的内容如下：

- 函数有效的PHP版本；
- 函数的参数个数及类型（可选参数用方括号括起来）；
- 函数的返回值类型。

手册还包含了每个函数的基本用法。

你应该养成查询PHP手册的好习惯。在遇到不熟悉的函数时，可以查询它的使用方法，还可以了解任何不明白的语言特性。更重要的是了解那些函数和特性的PHP版本，由于语言的发展，这些信息会不断地变化。

1.7 数字介绍

在介绍变量时，我明确指出PHP具有整型和浮点型（小数）数字类型。但是，依据我的经验，这两种类型可以归类到一般的数字（number）之下，而不会丢失任何有价值的差别（在极大程度上是这样）。PHP中有效的数字类型的变量可以是：

- 8
- 3.14
- 10980843985
- 4.2398508
- 4.4e2

注意：这些值永远不会用引号括起来（如果这样做，它们就是含有数值的字符串），也不能用逗号

来表示千位分隔符。此外，数字被假定为正，除非在其前面放置一个负号（-）。

对数字除了可以使用标准的算术运算符（参见表1-1）之外，还可以使用许多函数。两个常用的函数是**round()**和**number_format()**。前者用于把小数四舍五入为最接近的整数：

```
$n = 3.14;
$n = round ($n); // 3
```

或者把小数四舍五入到指定的位数：

```
$n = 3.142857;
$n = round ($n, 3); // 3.143
```

表1-1 标准数学运算符

运 算 符	含 义
+	加法
-	减法
*	乘法
/	除法
%	取模
++	增量
--	减量

number_format()函数用于把一个数字转换成更常见的表示形式，用逗号作为千位分隔符，例如：

```
$n = 20943;
$n = number_format ($n); // 20,943
```

这个函数还可以设置小数点的指定位数：

```
$n = 20943;
$n = number_format ($n, 2); // 20,943.00
```

为了练习操作数字，我将编写一个模型脚本，它执行人们可能在电子商务购物车中使用的计算。

使用数字

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为**number.php**（参见脚本1-8）。

脚本 1-8 numbers.php 脚本演示了电子商务应用程序中使用的基本数学计算

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
2   xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4    <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6    </head>
7    <body>
8      <?php # Script 1.8 - numbers.php
9
10 // Set the variables:
```

```

11 $quantity = 30; // Buying 30 widgets.
12 $price = 119.95;
13 $taxrate = .05; // 5% sales tax.
14
15 // Calculate the total:
16 $total = $quantity * $price;
17 $total = $total + ($total * $taxrate); // Calculate and add the tax.
18
19 // Format the total:
20 $total = number_format ($total, 2);
21
22 // Print the results:
23 echo '<p>You are purchasing <b>' . $quantity . '</b> widget(s) at a cost of <b>$' . $price .
24 ' '</b> each. With tax, the total comes to <b>$' . $total . '</b>.</p>';
25 ?>
26 </body>
27 </html>

```

(2) 建立必需的变量。

```

$quantity = 30;
$price = 119.95;
$taxrate = .05;

```

这个脚本将使用3个硬编码的变量，并在它们上面执行计算。在本书后面，你将看到如何能够动态地确定这些值（即通过用户与HTML表单的交互）。

(3) 执行计算。

```

$total = $quantity * $price;
$total = $total + ($total * $taxrate);

```

第一行将订单总额设置为所购买小器具的数量乘以每件小器具的价格。然后，第二行把税金加到总额中（通过用税率乘以总额来计算）。

(4) 格式化总额。

```
$total = number_format ($total, 2);
```

`number_format()`函数将把总额用千位分隔符分开，并将其四舍五入为两位小数。这将为最终用户显示更合适的结果。

(5) 打印结果。

```
echo '<p>You are purchasing <b>' . $quantity . '</b> widget(s) at a cost of <b>$' . $price . '</b> each.
With tax, the total comes to <b>$' . $total . '</b>.</p>';
```

这个脚本中的最后一步是打印出结果。为了使用HTML、打印的美元符号以及变量的组合，`echo`语句使用了单引号括住的文本和连接的变量。你将在本章的最后一个示例中看到另一个解决方案。

(6) 完成PHP代码和HTML页面。

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

(7) 将文件另存为`numbers.php`，存放在Web目录中，并在Web浏览器中测试它（参见图1-16）。

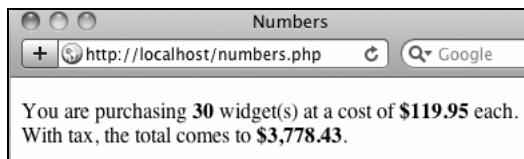


图1-16 数字PHP页面（参见脚本1-8）基于设置的值执行计算

(8) 如果需要，可更改最初的3个变量，然后重新运行脚本（参见图1-17）。

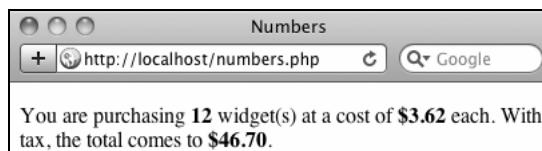


图1-17 要更改生成的Web页面，可以改变全部3个变量或其中的任何一个（对比图1-16）

✓ 提示

- PHP在大多数平台上支持的最大整数在20亿左右。对于更大的数字，PHP将自动使用浮点类型。
- 当处理算术运算时，会出现优先级问题（执行复杂计算的次序）。虽然PHP手册和其他资源倾向于列出优先级的层次结构，但是，我发现把子句组织在括号中来强制执行次序，会使程序设计更安全、更易读（参见脚本1-8的第17行）。
- 计算机处理小数的名声不佳。例如，2.0这个数字实际上存储为1.999 99。在大多数情况下，这不是一个问题，但是，倘若数学精度极其重要，这就要依靠整数，而不是小数。PHP手册中包含关于这个主题的信息，以及用于改进计算准确性的备用函数。
- 许多数学运算符还具有相应的赋值运算符，从而允许简写赋值语句。下面的一行语句：

```
$total = $total + ($total * $taxrate);
```

可以重写为：

```
$total += ($total * $taxrate);
```

- 如果没有使用两位小数设置\$price的值（例如，119.9或34），那么可能希望在打印\$price之前，对其应用number_format()。

1.8 常量介绍

常量像变量一样，用于临时存储一个值，但是常量在许多方面与变量不同。对于初学者，要创建常量，可以使用define()函数，而不是赋值运算符（=）。

```
define ('NAME', value);
```

注意：一条经验法则是，全都使用大写字母来命名常量，尽管并非必须如此。最重要的是，常量不会像变量那样使用初始美元符号（因为严格说来常量不是变量）。

只能赋予常量一个标量值，比如字符串或数字：

```
define ('USERNAME', 'troutocity');
define ('PI', 3.14);
```

与变量不同的是，不能更改常量的值。

为了访问常量的值，比如当你想打印它时，不能用引号括住常量，比如：

```
echo "Hello, USERNAME"; // Won't work!
```

对于这段代码，PHP只会打印出Hello, USERNAME（参见图1-18），而不会打印出USERNAME常量的值（因为没有美元符号告诉PHP，USERNAME是不同于字面量文本的任何内容）。可以代之以单独打印常量：

```
echo 'Hello, ';
echo USERNAME;
```

或者使用连接运算符：

```
echo 'Hello, ' . USERNAME;
```

PHP运行时利用了几个预定义的常量，这与本章前面使用的预定义变量非常相像。这些常量包括PHP_VERSION（PHP运行的版本）和PHP_OS（服务器的操作系统）。



图1-18 常量不能用引号括起来

使用常量

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为constants.php（参见脚本1-9）。

脚本 1-9 常量是在 PHP 中可以使用的另一种临时存储工具，它不同于变量

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
   xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Constants</title>
6  </head>
7  <body>
8      <?php # Script 1.9 - constants.php
9
10 // Set today's date as a constant:
11 define ('TODAY', 'March 16, 2011');
12
13 // Print a message, using predefined constants and the TODAY constant:
14 echo '<p>Today is ' . TODAY . '<br /> This server is running version <b>' . PHP_VERSION .
   '</b> of PHP on the <b>' . PHP_OS . '</b> operating system.</p>';
15
16 ?>
17 </body>
18 </html>
```

(2) 创建一个新的日期常量。

```
define ('TODAY', 'March 16, 2011');
```

人们普遍认为使用常量的意义不大，但是，这个示例将说明其重要性。在第9章中，你将看到如何使用常量来存储你的数据库访问信息。

(3) 打印出日期、PHP版本以及操作系统信息。

```
echo '<p>Today is ' . TODAY . '<br />This server is running version <b>' . PHP_VERSION . '</b> of
PHP on the <b>' . PHP_OS . '</b> operating system.</p>';
```

因为常量不能打印在双引号内，所以使用连接运算符来创建echo语句。

(4) 完成PHP代码和HTML页面。

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

(5) 将文件另存为constants.php，存放在Web目录中，并在Web浏览器中测试它（参见图1-19）。

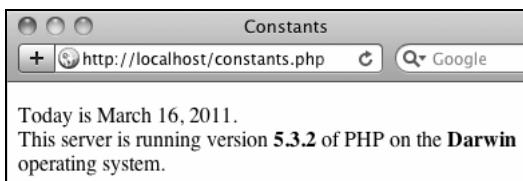


图1-19 通过利用PHP的常量，可以学习关于PHP设置的更多知识

✓ 提示

- 如果可能，可在另一台支持PHP的服务器上运行这个脚本（参见图1-20）。
- 这个操作系统名为Darwin（参见图1-19），是Mac OS X的技术术语。
- 在第12章中，你将学到另一个常量SID（它代表会话ID（session ID））。

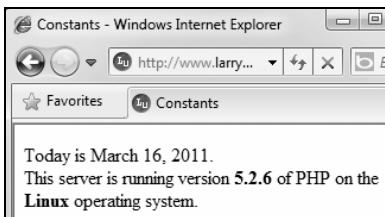


图1-20 在另一台服务器上运行相同的脚本（参见脚本1-9）会获得不同的结果

1.9 单引号与双引号

在PHP中，理解单引号与双引号的区别是重要的。如迄今为止的示例中所示的那样，给字符串赋值时可以使用echo或print语句。但是，这两种引号之间以及何时该用哪种引号存在关键的区别。迄今为止，我详细说明了它们的区别，但是，现在将更明确地定义它们的使用模式。

在PHP中，括在单引号内的值将照字面意义进行处理，而括在双引号内的值将被解释。换句话说，把变量和特殊字符（参见表1-2）放在双引号内将导致打印出它们表示的值，而不是它们的字面量值。例如，假定你具有：

```
$var = 'test';
```

代码echo "var is equal to \$var";将打印出var is equal to test，而代码echo 'var is equal to \$var';将打印出var is equal to \$var。使用一个转义的美元符号，代码echo "\\$var is equal to \$var";将打印出\$var is equal to test，而代码echo '\\$var is equal to \$var';将打印出\\$var is equal to \$var（参见图1-21）。

表1-2 当在双引号内使用这些字符时，它们具有特殊的含义

转义字符的代码	转义字符的含义
\"	双引号
\'	单引号
\\"	反斜杠
\n	换行符
\r	回车符
\t	制表符
\\$	美元符号

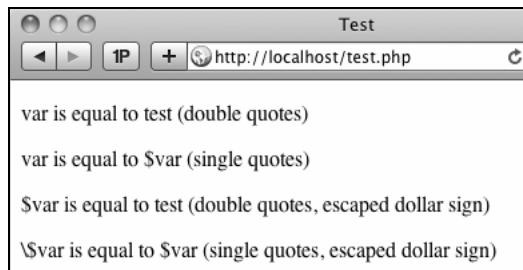


图1-21 单引号与双引号在PHP打印时的区别

正如这些示例所说明的，双引号将用变量的值（test）代替它的名称（\$var），并用特殊字符表示的值（\$）代替它的代码（\\$）。单引号总是准确地打印你输入的内容，除了转义的单引号（\'）和转义的反斜杠（\\）之外，它们将被分别打印为一个单引号和一个反斜杠。

在另一个说明这两种引号区别的示例中，我将修改numbers.php脚本，以此作为试验。

使用单引号和双引号

- (1) 在文本编辑器或IDE中打开numbers.php（参见脚本1-8）。
- (2) 删除现有的echo语句（参见脚本1-10）。

脚本 1-10 这个最终的脚本演示了使用单引号和双引号之间的区别

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Quotation Marks</title>
6  </head>
7  <body>
8  <?php # Script 1.10 - quotes.php
9
10 // Set the variables:
11 $quantity = 30; // Buying 30 widgets.
12 $price = 119.95;
13 $taxrate = .05; // 5% sales tax.
14
15 // Calculate the total.
16 $total = $quantity * $price;
17 $total = $total + ($total * $taxrate); // Calculate and add the tax.
18
19 // Format the total:
20 $total = number_format ($total, 2);
21
22 // Print the results using double quotation marks:
23 echo "<h3>Using double quotation marks:</h3>";
24 echo "<p>You are purchasing <b>$quantity</b> widget(s) at a cost of <b>\$price</b> each.
With tax, the total comes to <b>\$total</b>. </p>\n";
25
26 // Print the results using single quotation marks:
27 echo '<h3>Using single quotation marks:</h3>';
28 echo '<p>You are purchasing <b>$quantity</b> widget(s) at a cost of <b>\$price</b> each.
With tax, the total comes to <b>\$total</b>. </p>\n';
29
30 ?>
31 </body>
32 </html>
```

(3) 打印一个标题，然后使用双引号重写原来的echo语句。

```

echo "<h3>Using double quotation marks:</h3>";
echo "<p>You are purchasing <b>$quantity</b> widget(s) at a cost of <b>\$price</b> each. With tax,
the total comes to <b>\$total</b>. </p>\n";
```

在原始脚本中，将使用单引号和连接运算符打印结果。通过使用双引号，可以得到相同的结果。在使用双引号时，可以将变量置于字符串内。

不过，这里有一个难题：尝试将美元金额打印为\$12.34（其中12.34来自于一个变量）将建议你编码\$\$var。这样不行，应该对初始美元符号进行转义，得到\\$var，这段代码中已经出现了两次。第一个美元符号会被打印，而第二个美元符号则是变量名称的开始字符。

(4) 重复使用echo语句，这一次使用单引号。

```

echo '<h3>Using single quotation marks:</h3>';
echo '<p>You are purchasing <b>$quantity</b> widget(s) at a cost of <b>\$price</b> each. With tax,
the total comes to <b>\$total</b>. </p>\n';
```

这个echo语句用于突出使用单引号和双引号之间的区别。它将不会像所期望的那样工作，并且得到的页面将说明怎么回事。

(5) 如果需要，可更改页面的标题。

(6) 将文件另存为quotes.php，存放在Web目录中，并在Web浏览器中测试它（参见图1-22）。

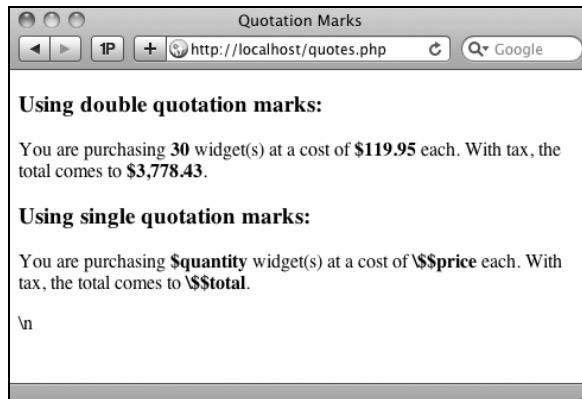


图1-22 这些结果演示了何时以及如何使用一种引号，而不使用另一种引号

(7) 查看Web页面的源代码，看看在每一种引号内使用换行符（\n）会有什么区别。

你应该会看到，当把换行符置于双引号内时，它会在HTML源代码中创建一个新行。当把它置于单引号内时，将取代之以打印字符串\n和\n。

✓ 提示

- 由于PHP将试图找出那些需要将其值插入到双引号内的变量，所以从理论上讲，使用单引号要快一些。但是，如果需要打印一个变量的值，则必须使用双引号。
- 因为有效的HTML常常包括许多用双引号括住的属性，所以当利用PHP打印HTML时，使用单引号最容易。

```
echo '<table width="80%" border="0" cellspacing="2" cellpadding="3" align="center">';
```

如果想使用双引号打印出这段HTML代码，将不得不对字符串中的所有双引号进行转义。

```
echo "<table width=\"80%\" border= \"0\" cellspacing=\"2\" cellpadding=\"3\" align= \"center\">";
```

- 在新版本的PHP中，可以使用\$\$price和\$\$total而不用在前面加反斜杠(\)，这得益于一些“内部魔法”。而其他版本的PHP不能这样做。为了确保结果的可靠性，我建议无论使用什么PHP版本，都使用\\$var语法打印美元字符，后面紧接变量的值。

- 如果你仍然不清楚这两种引号之间的区别，可以使用双引号，这样不太可能出问题。

1.10 基本的调试步骤

调试的确很难，这门技术必须得通过实际动手才能真正掌握。接下来的几十页内容都是专门阐述这一主题的，但即便如此，恐怕你也只能掌握必备调试技能中的一小部分。

之所以开局就如此悲观，是因为我想让你记住，没想好之前不要匆忙编程。有时代码不会如期工作，而程序员不可避免地会出现疏漏和错误，这会令人痛苦并抓狂——即使你使用的语言像PHP这样友好。总之，困惑和沮丧时常出现。自1999年起，我就开始用PHP编程了，但偶尔还是会不知所措。调试是一项需要掌握的重要技能，你将最终认识到它的必要性并深深体会到这一点。作为PHP编程历险的开始，本书将提供下面这些基础但又具体的调试技巧。

注意，这些都是为PHP初学者特别定制的一般的调试技巧。第8章将详细讲述其他技巧。

调试PHP脚本

确保你总是通过URL来运行PHP脚本！

这或许是初学者常犯的错误。PHP代码必须通过网络服务器软件运行，也就是说必须通过 `http://something` 发出请求。如果你看到的是PHP代码，而不是代码执行的结果，很可能是你没有通过URL运行PHP脚本。

了解正在运行的PHP版本。

一些问题是由于PHP的版本所致。在使用任何支持PHP的服务器之前，运行 `phpinfo.php` 脚本（查看附录A）或参考 `PHP_VERSION` 常量来确认使用的PHP版本。

确保 `display_errors` 启用

这是一个基本的PHP配置设置（参见附录A）。你可以运行 `phpinfo()` 函数来确认设置（使用浏览器，在结果页中搜索 `display_errors`）。出于安全的考虑，PHP可能没有设置成在发生错误的时候显示错误信息。如果是这种情况，当错误发生时你看到的将会是一个空白页。为了调试问题，需要查看错误信息，所以当学习的时候请启用这项设置。可以在附录A中找到相关的介绍。

检查HTML源代码。

有时候问题会隐藏在页面的HTML源代码中。事实上，有时PHP错误信息也会隐藏在那里。

信任错误信息！

初学者常犯的另外一个错误是不完整阅读PHP报告的错误消息，或者不完全相信这些错误。尽管这些错误消息有时非常含糊，并且看上去没有什么意义，但是还是不能忽略它！至少在错误发生于哪行代码这个问题上，PHP通常都是正确的。如果你需要将错误信息发给别人（比如当你要向我求助的时候），一定要包括全部的错误信息！

休息一下！

这么多年来我遇到过很多的编程问题，并且绝大多数最困难的问题，都是在我离开计算机去散步一会儿回来后解决的。编程时很容易陷入挫败与迷惑交织的痛苦中，在这种情况下，如果继续做下去，只会让事情变得更糟。

1.11 回顾和实践

本书会在每章的最后添加“回顾和实践”一节。在本节中，我会就本章涉及的内容提出一些问题，目的是巩固和扩展你学到的知识和技能。无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

1.11.1 回顾

- PHP代码要放在什么标签中间?
- PHP文件的扩展名是什么?
- 页面的字符编码指的是什么? 字符编码的作用是什么?
- 可以使用什么PHP函数(或语句)向浏览器发送数据?
- 使用单引号和双引号创建或打印字符串有什么不同?
- 在字符串中escape一个字符的意思是什么?
- PHP中三种注释语法分别是什么? 哪种可以用于多行注释?
- PHP的变量名以什么字符开始? 接下来可以是哪些字符? 变量名还可以包含其他哪些字符?
- PHP的变量名是否区分大小写?
- PHP的赋值运算符是什么?
- 如何创建字符串变量?
- PHP的连接操作符是什么? 连接赋值操作符是什么?
- 如何定义和使用常量?

1.11.2 实践

- 如果你还不清楚或不确定你正使用的PHP版本, 检查一下。
- 在PHP手册中查找一个本章提及过的字符串函数, 然后检查里面列出的其他的字符串函数。
- 在PHP手册中查找一个本章提及过的数值函数, 然后检查里面列出的其他的数值函数。
- 在PHP手册中搜索\$_SERVER变量, 查看它包含的其他信息。
- 从头编写一个新脚本, 定义一些字符串变量并输出它们的值。为输出值语句echo和print使用双引号。在输出中加入一些HTML, 以增加复杂度。然后使用单引号和拼接代替双引号重写脚本。
- 从头开始创建一个新的脚本, 定义、操作并显示一些数值型变量的值。

第2章

PHP编程

2

本章内容

- 创建HTML表单
- 处理HTML表单
- 事件语句和运算符
- 验证表单数据
- 介绍数组
- `for`和`while`循环
- 回顾和实践

既然你已经掌握了PHP脚本语言的基础知识，现在将用这些基础知识开始实际编程了。在本章中，你将开始创建更复杂的脚本，同时仍然会学习这种语言的一些标准构造、函数和语法。

首先，我将创建一个HTML表单，并说明如何使用PHP处理提交的值。之后，本章将介绍条件语句和其余的运算符（第1章介绍了赋值、连接和数学运算符）、数组（另一种变量类型），以及最后一种语言构造（循环）。

2.1 创建 HTML 表单

利用PHP处理HTML表单也许是任何动态Web站点中最重要的过程。其中包含两个步骤：首先，创建HTML表单自身。然后，创建相应的PHP脚本，用于接收和处理表单数据。

对HTML表单进行详细讨论超出了本书的范围，但是，我们可以通过一个贯穿本章的HTML表单示例快速了解一下。如果你不熟悉HTML表单的基础知识，包括各种元素，可以查看HTML资源以了解更多信息。

HTML表单是使用`form`标签和多种用于获取输出的元素创建的。`form`标签看起来如下：

```
<form action="script.php" method="post">  
</form>
```

就PHP而言，`form`标签最重要的属性是`action`，它指定将把表单数据发送到哪个页面。第二个属性（`method`）需要进一步讨论（参见框注“选择方法”），但是，`post`是最常用到的值。

选择方法

表单的method属性指定如何把数据发送到处理页面。两个选项(get和post)指示要使用的HTTP(Hypertext Transfer Protocol, 超文本传输协议)方法。get方法把提交的数据通过一系列追加到URL后面的名-值(name-value)对发送到接收页面。例如：

```
http://www.example.com/script.php?name=Homer&gender=M&age=35
```

使用get方法的好处是：可以在用户的Web浏览器中为得到的页面建立书签(因为它是一个URL)。因此，你还可以在Web浏览器中单击“后退”返回到一个get页面，或者重新加载它，而不会有任何问题(对于post，则不能执行这两种操作)。不幸的是，通过get传输的数据量有限，并且它不怎么安全(因为数据是可见的)。

一般来讲，get用于请求信息，比如数据库中的特定记录或者搜索的结果(搜索几乎总是使用get)。当需要采取一个动作时(比如在更新数据库记录或者发送电子邮件时)，就使用post方法。由于这些原因，在全书中，我一般使用post，如果出现例外情况，我会另外指出。

在form的开始标签和结束标签内可以放置不同的输入，它们可以是文本框、单选按钮、选项菜单、复选框等。如你将在下一节中所见到的，表单所具有的输入类型稍微不同于处理它的PHP脚本。不过，你应该注意为表单输入提供的名称，因为当表单遇到PHP代码时，这些名称将会是至关重要的。

创建HTML表单

(1) 在文本编辑器中创建一个新的HTML文档，命名为form.html(参见脚本2-1)。

脚本 2-1 这个简单的 HTML 表单将用于本章中的多个示例

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
2   DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4  <head>
5    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6    <title>Simple HTML Form</title>
7    <style type="text/css" title="text/css" media="all">
8      label {
9        font-weight: bold;
10       color: #300ACC;
11     }
12   </style>
13 </head>
14 <!-- Script 2.1 - form.html -->
15 <body>
16 <form action="handle_form.php" method="post">
17
18 <fieldset><legend>Enter your information in the form below:</legend>
19
20 <p><label>Name: <input type="text" name="name" size="20" maxlength="40" /></label></p>
21
22 <p><label>Email Address: <input type="text" name="email" size="40" maxlength="60" /></label>
23 </p>
```

```

23
24   <p><label for="gender">Gender: </label><input type="radio" name="gender" value="M" /> Male
25   <input type="radio" name="gender" value="F" /> Female</p>
26
27   <p><label>Age:</label>
28     <select name="age">
29       <option value="0-29">Under 30 </option>
30       <option value="30-60">Between 30 and 60</option>
31       <option value="60+>Over 60 </option>
32     </select></p>
33
34   <p><label>Comments: <textarea name="comments" rows="3" cols="40"></textarea></label></p>
35
36
37   <p align="center"><input type="submit" name="submit" value="Submit My Information" /></p>
38
39 </form>
40
41 </body>
42 </html>

```

文档为HTML页面使用了与上一章相同的基本语法。我在页面代码中添加了一些CSS (Cascading Style Sheet, 层叠样式表) 语句，目的是给表单设置样式 (就是让label元素显示为蓝色加粗样式)。

CSS是在HTML页中处理格式化和布局问题的首选方法。本书的不同地方看到一些CSS；如果你对它还不是很熟悉，请查看专门的CSS参考资料。

代码段第14行是注释，说明了文件名和脚本编号。

(2) 添加初始form标签。

```
<form action="handle_form.php" method="post">
```

因为action属性指定将把表单数据发送到哪个脚本，你应该给它提供一个合适的名称 (handle_form, 以对应于这个脚本：form.html) 以及.php扩展名 (因为PHP页面将处理这个表单的数据)。

(3) 开始创建HTML表单。

```
<fieldset><legend>Enter your information in the form below: </legend>
```

我正在使用fieldset和legend这两个HTML标签，因为我喜欢用它们创建的HTML表单的外观 (它们在表单周围添加了一个方框，并在顶部设置了一个标题)。尽管这与表单自身是不相关的。

(4) 添加两个文本输入框。

```
<p><label>Name: <input type="text" name="name" size="20" maxlength="40" /></label></p>
<p><label>Email Address: <input type="text" name="email" size="40" maxlength="60" /> </label></p>
```

它们只是简单的文本输入框，允许用户输入他们的名字和电子邮件地址 (参见图2-1)。为了不让你感到迷惑，特别说明一下：每个输入框标签末尾额外的空格和斜杠是有效的XHTML。例如，标准HTML中，这些标签可以改用maxlength="40"或maxlength="60"结尾。

(5) 添加一对单选按钮。

```
<p><label for="gender">Gender: </label><input type="radio" name="gender" value="M" /> Male <input
type="radio" name="gender" value="F" /> Female</p>
```

图2-1 两个文本输入框

2

这两个单选按钮（参见图2-2）具有相同的名称，这意味着只能选中其中的一个。不过，它们具有不同的值。

图2-2 如果多个单选按钮具有相同的名称，那么用户只能选中其中一个单选按钮

(6) 添加一个下拉菜单。

```
<p><label>Age:</label>
<select name="age">
  <option value="0-29">Under 30 </option>
  <option value="30-60">Between 30 and 60 </option>
  <option value="60+">Over 60 </option>
</select></label></p>
```

`select`标签开始创建下拉菜单，然后每个`option`标签将创建选项列表中的另一行选项（参见图2-3）。

图2-3 下拉菜单提供了三个选项，只能选择其中一个选项（在本示例中）

(7) 添加一个文本框，用于输入注释。

```
<p><label>Comments: <textarea name="comments" rows="3" cols="40"></textarea></label></p>
```

`textarea`不同于`text`输入框：它们会展示为一个方框（参见图2-4），而不是一个单行文本框。它们允许输入多得多的信息，并且可用于获取用户评论。

图2-4 `textarea`表单元素类型允许输入多得多的信息

(8) 完成表单。

```
</fieldset>
<p align="center"><input type="submit" name="submit" value="Submit My Information" /></p>
</form>
```

第一个标签将会关闭在第(3)步打开的`fieldset`。然后会创建一个`submit`按钮，并使用`div`标签将其居中。最后，关闭表单。

(9) 完成HTML页面。

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

(10) 将文件另存为form.html，存放在Web目录中，并在Web浏览器中查看它（参见图2-5）。

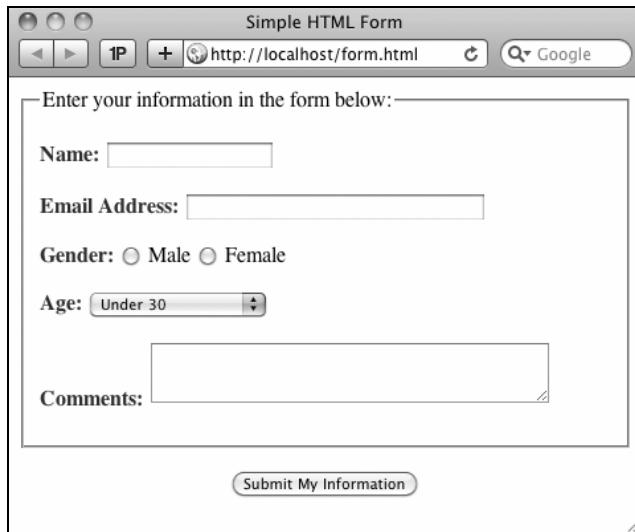


图2-5 这个完成的表单请求用户的一些基本信息

✓ 提示

- 因为这个页面只包含HTML，所以我使用了.html扩展名，但是，我也可以使用.php扩展名，而不会引起任何错误（因为PHP标签外面的代码将被视作HTML）。
- 你也可以为HTML表单标签制定一种字符编码：

```
<form accept-charset="utf-8">
```

默认情况下，页面会使用自身的字符编码来提交数据。

2.2 处理HTML表单

创建了HTML表单之后，我们来编写一个基本的PHP框架脚本处理它。所谓脚本将处理表单，是指它将对其接收到的数据（用户输入到表单中的数据）进行某些处理。在本章中，脚本将简单地把数据发送回Web浏览器。在后面的示例中，对这些数据的处理方式有如下几种：将其存储在MySQL数据库中，核对它们以前存储的值，在电子邮件中发送它们，等等。

PHP的特性以及是什么使得它如此容易学习和使用，是它能够与HTML表单极好地进行交互。PHP脚本把接收到的信息存储在特殊的变量中。例如，假定你有一个表单，其中有一个输入，定义如下：

```
<input type="text" name="city" />
```

无论用户在该元素中输入什么内容，都可以通过一个名为\$_REQUEST ['city'] 的PHP变量访问它。使拼写和大小写完全匹配非常重要，因为PHP对变量名区分大小写，因此\$_REQUEST ['city'] 将会工作，但是\$_Request ['city'] 或\$_REQUEST ['City'] 将没有值。

下一个示例将是一个处理已经创建好的HTML表单的PHP脚本（参见脚本2-1）。该脚本将把表单数据赋予新的变量（以使用其简写版本，就像脚本1-5中那样）。然后，脚本将打印接收到的值。

处理HTML表单

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为handle_form.php（参见脚本2-2）。

脚本 2-2 这个脚本接收并打印出输入到 HTML 表单（参见脚本 2-1）中的信息

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD
   /xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6  </head>
7  <body>
8  <?php # Script 2.2 - handle_form.php
9
10 // Create a shorthand for the form data:
11 $name = $_REQUEST['name'];
12 $email = $_REQUEST['email'];
13 $comments = $_REQUEST['comments'];
14 /* Not used:
15    $_REQUEST['age']
16    $_REQUEST['gender']
17    $_REQUEST['submit']
18 */
19
20 // Print the submitted information:
21 echo "<p>Thank you, <b>$name</b>, for the following comments:<br />";
22 <tt>$comments</tt></p>
23 <p>We will reply to you at <i>$email</i>. </p>\n";
24
25 ?>
26 </body>
27 </html>
```

(2) 添加PHP开始标签，并创建表单数据变量的简写版本。

```
<?php # Script 2.2 - handle_form.php
$name = $_REQUEST['name'];
$email = $_REQUEST['email'];
$comments = $_REQUEST['comments'];
```

依据以前概括的规则，对于输入到第一个表单输入框（其名称为name）中的数据，可以通过变量\$_REQUEST['name']来访问它（参见表2-1）；对于输入到电子邮件表单输入框（其name值为email）中的数据，可以通过\$_REQUEST['email']来访问它。这同样适用于输入的注释数据。此外，这里的变量的拼写和大小写必须与HTML表单中相应的name值完全匹配。

此时，还无法使用age、gender和submit表单元素。

表2-1 HTML表单元素及其对应的PHP变量

元素名称	变量名称
name	<code>\$_REQUEST['name']</code>
email	<code>\$_REQUEST['email']</code>
comments	<code>\$_REQUEST['comments']</code>
age	<code>\$_REQUEST['age']</code>
gender	<code>\$_REQUEST['gender']</code>
submit	<code>\$_REQUEST['submit']</code>

(3) 打印出接收到的名称、电子邮件和注释值。

```
echo "<p>Thank you, <b>$name</b>, for the following comments:<br />
<tt>$comments</tt></p>
<p>We will reply to you at <i>$email</i>.</p>\n";
```

将使用echo语句、双引号以及很少一点HTML格式化效果简单地打印出提交的值。

(4) 完成HTML页面。

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

(5) 将文件另存为handle_form.php，将其存放在与form.html相同的Web目录中。

(6) 通过URL加载form.html在Web浏览器中测试这两个文档，然后填写和提交表单（参见图2-6和图2-7）。



图2-6 要测试handle_form.php，必须通过URL加载表单，然后填写和提交它



图2-7 脚本应该显示像这样的结果

由于必须通过URL运行PHP脚本（参见第1章），也必须通过URL运行表单。否则，在提交表单时，将看到PHP代码（参见图2-8），而不会看到正确的结果（参见图2-7）。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Form Feedback</title>
</head>
<body>
<?php # Script 2.2 - handle_form.php

// Create a shorthand for the form data:
$name = $_REQUEST['name'];
$email = $_REQUEST['email'];
$comments = $_REQUEST['comments'];
/* Not used:
$_REQUEST['age']
$_REQUEST['gender']
$_REQUEST['submit']
*/

// Print the submitted information:
echo "<p>Thank you, <b>$name</b>, for the following comments:<br />
<tt>$comments</tt></p>
<p>We will reply to you at <i>$email</i>.</p>\n";
?>
</body>
</html>

```

图2-8 如果在提交表单后看到PHP代码本身，问题很可能是没有从URL访问表单

✓ 提示

- `$_REQUEST`是一个特殊的变量类型，称为超全局（superglobal）变量。它存储了通过GET或POST方法发送到PHP页面的所有数据，以及在cookie中可访问的数据。在本章后面将讨论超全局变量。
- 如果你对这个脚本有任何问题，可以应用第1章中建议的调试技术。如果它们没有解决问题，请检查第8章中列出的扩展调试技术。如果仍然身陷困境，可以向本书的支持论坛求助，以获取帮助。
- 如果PHP脚本在应该打印变量的地方显示空白，这意味着变量没有赋值。两个最可能的原因是：无法在表单中输入值；或者变量名拼写错误或弄错了它的大小写。
- 如果你看到任何`Undefined variable: variablename`（未定义的变量：变量名）错误，这是由于引用的变量没有赋值，并且PHP设置了最高级的错误报告。参见上一条提示，了解变量未赋值的原因。第8章详细讨论了错误报告。

- 要比较PHP处理不同表单输入类型的方式，可打印出\$_REQUEST['age']和\$_REQUEST['gender']的值（参见图2-9）。



图2-9 gender和age的值对应于在表单的HTML中定义的那些值

Magic Quotes

PHP的早期版本中具有一个称为Magic Quotes（魔术引号）的特性，官方并不推荐使用这个特性并终将删除。Magic Quotes被启用时，将会对提交的表单数据中的单引号和双引号自动进行转义（实际上有三种Magic Quotes，但这是最重要的一种）。因此字符串I'm going out将转变成I\'m going out。

在某些情况下，对可能有问题的字符进行转义可能是有用的，甚至是必要的。但是，如果在安装PHP时启用Magic Quotes，当PHP脚本打印出表单数据时，你将看到这些反斜杠。可以使用stripslashes()函数撤销它的作用：

```
$var = stripslashes($var);
```

这个函数将删除在\$var中发现的任何反斜杠。这具有把转义的提交字符串转变回其原来的未转义值的作用。

为了在handle_form.php（参见脚本2-2）中使用它，可以编写如下代码：

```
$name = stripslashes($_REQUEST['name']);
```

如果你在表单数据中看不到反斜杠，就不用关心Magic Quotes了。

2.3 条件语句和运算符

PHP用于创建条件语句的3个主要术语是：if、else和elseif（它也可以写作两个单词：else if）。每个条件语句都包含有一个if子句：

```
if (condition) {
    // Do something!
}
```

if也可以有else子句：

```
if (condition) {
```

```
// Do something!
} else {
    // Do something else!
}
```

`elseif`子句允许添加更多的条件：

```
if (condition1) {
    // Do something!
} elseif (condition2) {
    // Do something else!
} else {
    // Do something different!
}
```

如果条件为真，则将执行其下面的花括号（{}）中的代码。如果条件不为真，PHP将继续往下执行。如果有第二个条件（在`elseif`之后），则检查其是否为真。这个过程将继续（可以根据需要使用多个`elseif`子句）直至PHP遇到`else`，此时将自动执行它；或者如果没有`else`，则PHP将执行到条件语句终止为止。因此，总是把`else`放在最后，并将其视作默认的动作，除非满足特定的准则（已列出的条件），这样做是重要的。

PHP中条件为真的情况有许多种。下面是常见的一些：

- `$var`，如果`$var`具有非0值、空字符串、`FALSE`或`NULL`，则条件为真；
- `isset($var)`，如果`$var`具有不同于`NULL`的任何值，包括0、`FALSE`或空字符串，则条件为真；
- `TRUE`、`true`、`True`等。

在第二个示例中，引入了一个新函数`isset()`。这个函数用于检查一个变量是否被设置，这意味着它具有一个不同于`NULL`的值（提醒一下，在PHP中`NULL`是一种特殊类型，表示没有设置值）。你还可以结合使用圆括号以及比较和逻辑运算符（参见表2-2）来建立更复杂的表达式。

表2-2 在编写条件语句时，经常会使用的比较和逻辑运算符

符 号	含 义	类 型	示 例
<code>==</code>	等于	比较	<code>\$x == \$y</code>
<code>!=</code>	不等于	比较	<code>\$x != \$y</code>
<code><</code>	小于	比较	<code>\$x < \$y</code>
<code>></code>	大于	比较	<code>\$x > \$y</code>
<code><=</code>	小于或等于	比较	<code>\$x <= \$y</code>
<code>>=</code>	大于或等于	比较	<code>\$x >= \$y</code>
<code>!</code>	非	逻辑	<code>!\$x</code>
<code>&&</code>	与	逻辑	<code>\$x && \$y</code>
<code>AND</code>	与	逻辑	<code>\$x and \$y</code>
<code> </code>	或	逻辑	<code>\$x \$y</code>
<code>OR</code>	或	逻辑	<code>\$x or \$y</code>
<code>XOR</code>	异或	逻辑	<code>\$x XOR \$y</code>

使用条件语句

- (1) 在文本编辑器或IDE中打开handle_form.php (参见脚本2-2)。
- (2) 在echo语句前面, 添加一个条件语句来创建\$gender变量 (参见脚本2-3)。

脚本2-3 条件语句允许脚本依据特定的准则修改行为。在 handle_form.php 的这个修改过的版本中, 使用两个条件语句来验证性别单选按钮

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
   xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6  </head>
7  <body>
8  <?php # Script 2.3 - handle_form.php #2
9
10 // Create a shorthand for the form data:
11 $name = $_REQUEST['name'];
12 $email = $_REQUEST['email'];
13 $comments = $_REQUEST['comments'];
14
15 // Create the $gender variable:
16 if (isset($_REQUEST['gender'])) {
17     $gender = $_REQUEST['gender'];
18 } else {
19     $gender = NULL;
20 }
21
22 // Print the submitted information:
23 echo "<p>Thank you, <b>$name</b>, for the following comments:<br />
24 <tt>$comments</tt></p>
25 <p>We will reply to you at <i>$email</i>. </p>\n";
26
27 // Print a message based upon the gender value:
28 if ($gender == 'M') {
29     echo '<p><b>Good day, Sir!</b> </p>';
30 } elseif ($gender == 'F') {
31     echo '<p><b>Good day, Madam! </b></p>';
32 } else { // No gender selected.
33     echo '<p><b>You forgot to enter your gender!</b></p>';
34 }
35
36 ?>
37 </body>
38 </html>
```

这是验证表单输入 (特别是对于单选按钮、复选框或选项菜单) 的一种简单、有效的方式。如果用户选中了任何一个性别单选按钮, 那么\$_REQUEST['gender']将具有一个值, 这意味着条件isset(\$_REQUEST['gender'])为真。在这种情况下, 将会给这个变量的简写版本(\$gender)赋予\$_REQUEST['gender']的值, 重复对\$name、\$email和\$comments使用过的技术。如果用户没有单击其中一

个单选按钮，则条件不为真，那么将给\$gender赋值NULL，这指示它没有值。注意，NULL不在引号中。

(3) 在echo语句后面，添加另一个条件语句，基于\$gender的值打印一条消息。

```
2
if ($gender == 'M') {
    echo '<p><b>Good day, Sir!</b> </p>';
} elseif ($gender == 'F') {
    echo '<p><b>Good day, Madam! </b></p>';
} else {
    echo '<p><b>You forgot to enter your gender!</b></p>';
}
```

这个if-elseif-else条件语句会查看\$gender变量的值，并针对每一种可能性打印一条不同的消息。双等于号（==）指相等，而单个等于号（=）则用于赋值，记住这一点非常重要。这个区别很重要，因为条件\$gender == 'M'可能为真，也可能不为真；但是\$gender = 'M'总是为真。

此外，这里使用的值（M和F）必须与HTML表单中对应的值（用于每个单选按钮的值）完全相同。对于字符串来说，相等性比较区分大小写，因此m不等于M。

(4) 保存文件，将其存放在Web目录中，并在Web浏览器中测试它（参见图2-10、图2-11和图2-12）。



Thank you, **Marge Simpson**, for the following comments:
Bart, don't use the Touch of Death on your sister.

We will reply to you at marge@example.edu.

Good day, Madam!

图2-10 基于性别的条件语句针对表单中的每个选择打印一条不同的消息

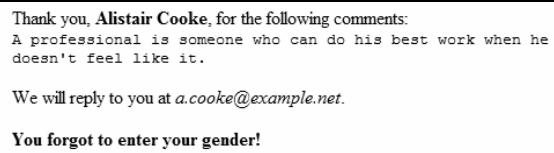


Thank you, **Ralph Waldo Emerson**, for the following comments:
Difficulties exist to be surmounted.

We will reply to you at rwe@example.org.

Good day, Sir!

图2-11 当更改性别值时，相同的脚本将产生不同的结果（对比图2-10）



Thank you, **Alistair Cooke**, for the following comments:
A professional is someone who can do his best work when he
doesn't feel like it.

We will reply to you at a.cooke@example.net.

You forgot to enter your gender!

图2-12 如果没有选择性别，就会打印一条消息，指出用户的疏忽

✓ 提示

- 尽管PHP没有严格的格式化规则，但是应该可以让人清楚地看出一个代码块是条件语句的子集，以这种方式格式化条件语句是一个标准的过程和良好的编程形式。通常的做法是缩进代码块。

- 你可以并且往往嵌套条件语句（把一个条件语句放在另一个条件语句内部）。
- 这个脚本中的第一个条件语句 (`isset()`) 是一个如何使用默认值的理想示例。假定除非满足一个条件：设置了`$_REQUEST['gender']`，否则 (`else`) `$gender`具有一个NULL值。
- 如果只执行一条语句，则用于指示条件语句开始和结束的花括号不是必需的。不过，出于清晰性考虑，我建议你几乎总是使用它们。
- “与” 和 “或” 逻辑关系各自都有两个运算符，它们在技术上稍有不同。在没有特殊说明的情况下，我习惯使用`&&`和`||`而不是`AND`和`OR`。
- `XOR`是“异或”运算符。如果条件句`$x XOR $y`是真，则表示：`$x`是真或`$y`是真，但是这两者不是同时为真。

switch

PHP还有另一种条件语句，称为**switch**，最适合代替较长的if-elseif-else条件语句。**switch**的语法是：

```
switch ($variable) {
    case 'value1':
        // Do this.
        break;
    case 'value2':
        // Do this instead.
        break;
    default:
        // Do this then.
        break;
}
```

switch条件语句将`$variable`的值与不同的`case`作比较。当它发现一个匹配时，就会执行其后的代码，直至遇到`break`。如果没有发现匹配，则会执行`default`，假定存在`default`（它是可选的）。仅限于在可以检查变量值与某些情况的相等性的条件下使用**switch**条件语句，往往不能轻松地检查更复杂的条件语句。

2.4 验证表单数据

与处理HTML表单相关的一个关键概念是验证表单数据。从错误管理和安全性两方面考虑，你永远都不应该信任输入到HTML表单中的数据。对于错误的数据，无论它是蓄意产生破坏，还是仅仅无意地造成不适，都需要你（Web开发人员）针对预期的要求对其进行测试。

验证表单数据需要使用条件语句以及许多函数、运算符和表达式。一个常用的标准函数是`isset()`，它用于测试一个变量是否具有值（包括`0`、`FALSE`，或者一个空字符串，但不能是`NULL`）。

使用`isset()`函数的一个问题是：空字符串测试为`TRUE`，这意味着它不是验证HTML表单中的文本输入和文本框的有效方式。要检查用户输入到文本元素中的内容，可以使用`empty()`函数。它将检查一个变量是否具有空（`empty`）值：空字符串、`0`、`NULL`或`FALSE`。

表单验证的第一个目标是确保在表单元素中输入或选择了某些内容。第二个目标是确保提交的数

据具有正确的类型（数字、字符串等）、正确的格式（如电子邮件地址）或特定的可接受值（如\$gender应该等于M或F）。因为PHP主要用于处理表单，所以在后续章节中将再度强调验证表单数据这一要点。但是，首先我将创建一个新的handle_form.php脚本，确保在引用变量之前，它们具有值（在这个版本中将有足够多的改变，简单地更新脚本2-3没有意义）。

验证表单

(1) 在文本编辑器或IDE中开始创建一个新的PHP脚本，命名为handle_form.php（参见脚本2-4）。

脚本 2-4 在使用 HTML 表单数据之前对其进行验证，对于 Web 安全性和获得专业的结果是至关重要的。在这里，条件语句检查每个引用的表单元素都具有值

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
   DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6      <style type="text/css" title="text/css" media="all">
7          .error {
8              font-weight: bold;
9              color: #C00;
10         }
11     </style>
12 </head>
13 <body>
14 <?php # Script 2.4 - handle_form.php #3
15
16 // Validate the name:
17 if (!empty($_REQUEST['name'])) {
18     $name = $_REQUEST['name'];
19 } else {
20     $name = NULL;
21     echo '<p class="error">You forgot to enter your name!</p>';
22 }
23
24 // Validate the email:
25 if (!empty($_REQUEST['email'])) {
26     $email = $_REQUEST['email'];
27 } else {
28     $email = NULL;
29     echo '<p class="error">You forgot to enter your email address!</p>';
30 }
31
32 // Validate the comments:
33 if (!empty($_REQUEST['comments'])) {
34     $comments = $_REQUEST['comments'];
35 } else {
36     $comments = NULL;
37     echo '<p class="error">You forgot to enter your comments!</p>';
38 }
39
40 // Validate the gender:
```

```

41 if (isset($_REQUEST['gender'])) {
42
43     $gender = $_REQUEST['gender'];
44
45     if ($gender == 'M') {
46         echo '<p><b>Good day, Sir!</b></p>';
47     } elseif ($gender == 'F') {
48         echo '<p><b>Good day, Madam!</b></p>';
49     } else { // Unacceptable value.
50         $gender = NULL;
51         echo '<p class="error">Gender should be either "M" or "F"!</p>';
52     }
53
54 } else { // $_REQUEST['gender'] is not set.
55     $gender = NULL;
56     echo '<p class="error">You forgot to select your gender!</p>';
57 }
58
59 // If everything is OK, print the message:
60 if ($name && $email && $gender && $comments) {
61
62     echo "<p>Thank you, <b>$name</b>, for the following comments:<br />
63     <tt>$comments</tt></p>
64     <p>We will reply to you at <i>$email</i>. </p>\n";
65
66 } else { // Missing form value.
67     echo '<p class="error">Please go back and fill out the form again.</p>';
68 }
69
70 ?>
71 </body>
72 </html>

```

(2) 在HTML head内，添加一些CSS代码。

```

<style type="text/css" title="text/css" media="all">
.error {
    font-weight: bold;
    color: #C00;
}
</style>

```

在这个脚本中，我定义了一个名为error的CSS类。具有这个类名的任何HTML元素都将被格式化为加粗、红色（与这本用黑、白色印刷的图书相比，在Web浏览器中可以更清楚地看到这一点）。

(3) 检查是否输入了名称。

```

if (!empty($_REQUEST['name'])) {
    $name = $_REQUEST['name'];
} else {
    $name = NULL;
    echo '<p class="error">You forgot to enter your name!</p>';
}

```

检查是否填写了表单文本输入框的简单方式是使用empty()函数。如果\$_REQUEST['name']具有一

个不同于空字符串（`0`、`NULL`或`FALSE`）的值，我就假定已经输入了它们的名称，并把该值赋予简写的变量。如果`$_REQUEST['name']`为空，将把`$name`变量设置为`NULL`，并打印一条出错消息。该错误消息使用了CSS类。

2

(4) 为电子邮件地址和评论重复相同的过程。

```
if (!empty($_REQUEST['email'])) {
    $email = $_REQUEST['email'];
} else {
    $email = NULL;
    echo '<p class="error">You forgot to enter your email address!</p>';
}
if (!empty($_REQUEST['comments'])) {
    $comments = $_REQUEST['comments'];
} else {
    $comments = NULL;
    echo '<p class="error">You forgot to enter your comments!</p>';
}
```

像第(3)步中的`$_REQUEST['name']`变量那样对这两个变量进行相同的处理。

(5) 开始验证性别变量。

```
if (isset($_REQUEST['gender'])) {
    $gender = $_REQUEST['gender'];
```

性别验证是一个包含两个步骤的过程。首先，我使用`isset()`检查它是否具有值。这是一个主要的`if-else`条件语句，其行为方式不同于对名称、电子邮件地址和评论所做的处理。

(6) 对照特定的值检查`$gender`。

```
if ($gender == 'M') {
    $greeting = '<p><b>Good day, Sir!</b></p>';
} elseif ($gender == 'F') {
    $greeting = '<p><b>Good day, Madam!</b></p>';
} else {
    $gender = NULL;
    echo '<p class="error">Gender should be either "M" or "F"!</p>';
}
```

在`if`子句内是一个嵌套的`if-elseif-else`条件语句，用于测试变量是否具有可接受的值。这是两步性别验证的第二部分。

这些条件语句本身与上一个脚本中的相同。如果性别最后不等于`M`或`F`，就会出现问题，并打印一条出错消息。在这些情况下，也会把`$gender`变量设置为`NULL`，因为它具有一个不可接受的值。

如果`$gender`确实具有一个有效值，就会打印一条特定性别的消息。

(7) 完成用于性别验证的`if-else`主条件语句。

```
} else {
    $gender = NULL;
    echo '<p class="error">You forgot to select your gender!</p>';
}
```

如果没有设置`$_REQUEST['gender']`，就会应用这个`else`子句。完整的嵌套的条件语句（参见脚本2-4的第41~57行）将会成功地检查每一种可能的情况：

- 没有设置\$_REQUEST['gender'];
- \$_REQUEST['gender']具有一个值M;
- \$_REQUEST['gender']具有一个值F;
- \$_REQUEST['gender']具有其他某个值。

你可能想知道为什么可能出现最后一种情况，考虑值是在HTML表单中建立的。如果恶意用户创建他们自己的表单并提交给handle_form.php脚本（这很容易做到），他们可以给\$_REQUEST ['gender'] 提供他们想要的任何值。

(8) 如果通过了所有的测试，则打印消息。

```
if ($name && $email && $gender && $comments) {
    echo "<p>Thank you, <b>$name</b>, for the following comments:<br />
<tt>$comments</tt></p>
<p>We will reply to you at <i>$email</i>. </p>\n";
    echo $greeting;
} else {
    echo '<p class="error">Please go back and fill out the form again.</p>';
}
```

如果每个列出的变量都具有一个真值，则主条件为真。如果每个变量都通过了其测试，则会具有一个值；但是，如果没有通过测试，则会具有一个NULL值。如果每个变量都具有一个值，则表单完成，因此会打印Thank you消息。如果任何变量都是NULL，就会打印第二条消息（参见图2-13和图2-14）。

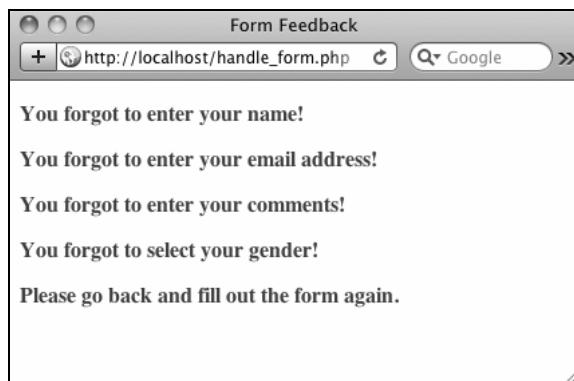


图2-13 脚本现在会检查是否填写了每个表单元素（年龄除外），并报告哪些元素尚未填写

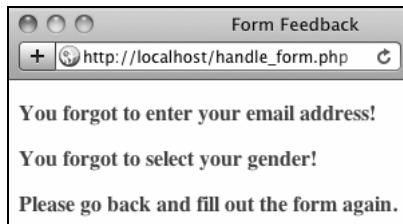


图2-14 即使只跳过了一两个字段，也不会打印Thank you消息

(9) 关闭PHP部分，完成HTML代码。

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

(10) 将文件另存为handle_form.php，存放在与form.html相同的Web目录中，并在Web浏览器中测试它（参见图2-13和图2-14）。把表单填写到不同的完整程度以测试新的脚本（参见图2-15）。

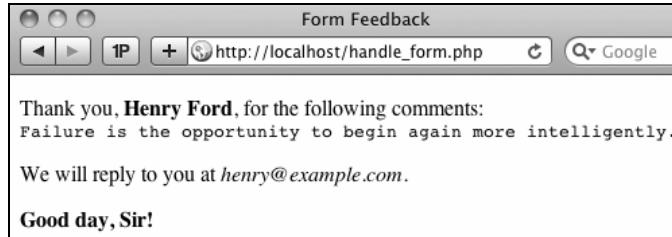


图2-15 如果正确输入了所有内容，脚本将像以前那样工作

✓ 提示

- 要测试提交的值是否是数字，可使用is_numeric()函数。
- 在第14章中，你将看到如何使用正则表达式来验证表单数据。
- 一个经过深思熟虑的良好表单，可以让用户在填写表单时知道哪些字段是必须填写的，并且在适当的地方指出了那个字段的格式（如日期或电话号码）。

2.5 介绍数组

我将在本书中介绍的最后一类变量类型是数组。与字符串和数字（它们是标量变量，这意味着它们一次只能存储一个值）不同的是，数组（array）可以保存多份单独的信息。因此，数组就像是值的列表，其中每个值可以是字符串或数字，甚至是另一个数组。

数组可以构造成一系列键-值（key-value）对，其中每一对都是那个数组的一个项目或元素（element）。对于列表中的每个项目，都有一个与之关联的键（key）（或索引（index），（参见表2-3）。

表2-3 \$artists数组使用数字作为它的键

键	值
0	The Mynabirds
1	Jeremy Messersmith
2	The Shins
3	Iron and Wine
4	Alexi Murdoch

PHP支持两种数组：索引数组（indexed array）和关联数组（associative array），前者使用数字作

为键（如表2-3所示），后者使用字符串作为键（参见表2-4）。像在大多数编程语言中一样，索引数组的第一个索引开始于0，除非显式指定键。

表2-4 \$states数组使用字符串（确切地讲是状态的缩写词）作为它的键

键	值
MD	Maryland
PA	Pennsylvania
IL	Illinois
MO	Missouri
IA	Iowa

数组遵守与任何其他变量相同的命名规则。因此，不能未经考虑地区分\$var是数组，还是字符串或数字。访问各个数组元素时会显示重要的语法区别。

要从数组中检索特定的值，需要先引用数组变量名称，然后在方括号中指出键：

```
$band = $artists[0]; // The Mynabirds
echo $states['MD']; // Maryland
```

可以像PHP中的其他值那样使用数组的键：数字（例如，2）永远不会用引号括住，而对字符串（MD）则必须这样做。

由于数组使用的语法不同于其他变量，对它们执行打印更具技巧性。首先，因为数组可以包含多个值，所以不能编写简单的代码打印它们（参见图2-16）：

```
echo "My list of states: $states";
```

My list of states: Array

图2-16 只通过引用数组名称来尝试打印一个数组将导致打印单词Array

不过，如果使用索引（数字）键，那么打印各个元素的值将很简单：

```
echo "The first artist is $artists[0].";
```

但是，如果数组使用字符串作为键，用于括住键的引号将使语法变得混乱。下面的代码将引发一个解析错误（参见图2-17）：

```
echo "IL is $states['IL']."; // BAD!
```

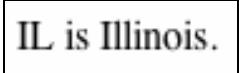
Parse error: syntax error, unexpected T_ENCAPSED_AND_WHITESPACE,
expecting T_STRING or T_VARIABLE or T_NUM_STRING in
/Users/larryullman/Sites/phpmysql4/test.php on line 13

图2-17 由于没有使用花括号将要打印的数组元素括起来而出现的解析错误

为了解决这个问题，当数组使用字符串作为它的键时，把数组名和键括在花括号中（参见图2-18）：

```
echo "IL is {$states['IL']}.";
```

2



IL is Illinois.

图2-18 使用花括号将要打印的数组元素括起来，会得到正确的结果

你觉得数组似乎有一点面熟，那是因为你已经使用过两个数组：`$_SERVER`（在第1章中）和`$_REQUEST`（在本章中）。为了让你熟悉另一个数组以及如何直接打印数组值，将使用更具体的`$_POST`数组创建基本的`handle_form.php`页面的修改版本（参见框注“超全局数组”）。

超全局数组

PHP默认包括多个预定义的数组。我曾经介绍过一些超全局(superglobal)变量：`$_GET`、`$_POST`、`$_REQUEST`、`$_SERVER`、`$_ENV`、`$_SESSION`和`$_COOKIE`。

PHP使用`$_GET`变量来存储通过get方法发送到PHP脚本的所有变量和值（它们可能来自于HTML表单，但不一定非得如此）。`$_POST`存储使用post方法从HTML表单发送到PHP脚本的所有数据。这两个变量以及`$_COOKIE`变量都是你一直在使用的`$_REQUEST`变量的子集。

在第1章中使用的`$_SERVER`存储关于正在运行的服务器PHP的信息，就像`$_ENV`所做的那样。在第11章中将讨论`$_SESSION`和`$_COOKIE`。

良好的安全性与编程的一个方面在于精确地引用变量。这意味着尽管可以使用`$_REQUEST`访问通过post方法提交的表单数据，但是`$_POST`将更准确。

使用数组

(1) 在文本编辑器中创建一个新的PHP脚本，命名为`handle_form.php`（参见脚本2-5）。

脚本 2-5 超全局变量（如这里的`$_POST`）只是你将在 PHP 中使用的一种数组

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
   DTD/xhtml1-transitional.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3  <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6  </head>
7  <body>
8  <?php # Script 2.5 - handle_form.php #4
9
10 // Print the submitted information:
11 if ( !empty($_POST['name']) && !empty($_POST['comments']) && !empty($_POST['email']) ) {
12     echo "<p>Thank you, <b>{$POST['name']}

```

```

15 } else { // Missing form value.
16     echo '<p>Please go back and fill out the form again.</p>';
17 }
18 ?>
19 </body>
20 </html>

```

(2) 执行一些基本的表单验证。

```
if ( !empty($_POST['name']) && !empty($_POST['comments']) && !empty($_POST['email']) ) {
```

在这个脚本的前一个版本中，通过引用\$_REQUEST数组来访问值。但是，由于这些变量来自于一个使用post方法的表单（参见脚本2-1），\$_POST将是更准确（因而更安全）的引用（参见框注“超全局数组”）。

条件语句检查这3个文本输入框全都不为空。通过使用“与”运算符（`&&`），仅当这三个子条件语句都为真时整个条件语句才为真。

(3) 打印消息。

```
echo "<p>Thank you, <b>{$_POST ['name']}

```

一旦你理解了数组的概念，仍然需要掌握打印它所涉及的语法。在打印把字符串用作键的数组元素时，可以使用花括号（比如这里的`$_POST['name']`）来避免解析错误。

(4) 完成开始于第(2)步中的条件语句。

```
} else {
    echo '<p>Please go back and fill out the form again.</p>';
}
```

如果第(2)步中3个子条件语句中的任何一个不为真（也就是说，如果任何变量具有空值），那么就会应用这个else子句，并打印错误消息（参见图2-19）。

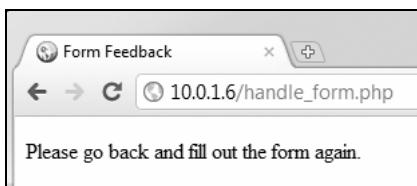


图2-19 如果3个测试的表单输入框中任何一个为空，就会打印这个一般的错误消息

(5) 完成PHP和HTML代码。

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

(6) 将文件另存为handle_form.php，存放在与form.html相同的Web目录中，并在Web浏览器中测试它（参见图2-20）。

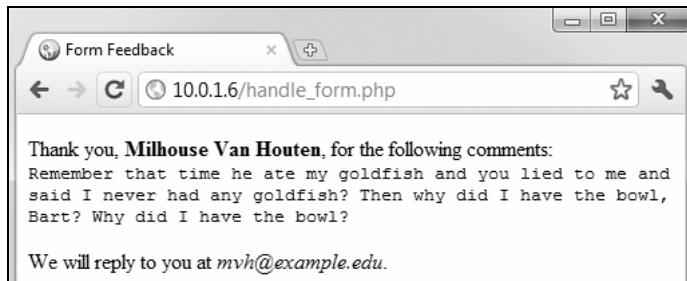


图2-20 脚本现在使用\$_POST数组并没有影响可视的结果

✓ 提示

- 由于PHP对其变量结构的要求很宽松，数组甚至可以使用数字和字符串的组合作为它的键。唯一重要的规则是数组的每一个键都必须是唯一的。
- 如果你发现访问超全局数组的语法令人感到糊涂（例如，\$_POST['name']），可以像你做过的那样在脚本顶部使用简写技术：

```
$name = $_POST['name'];
```

在这个脚本中，然后需要更改条件语句和echo语句来引用\$name等变量。

- 在输出用字符串作为键的数组的值时，需要用花括号把数组括起来，以避免错误。下面示例使用的引号并没有问题，因此不需要花括号：

```
echo $_POST['name']; echo "The first item is $item[0]."; $total = number_format($cart['total']);
```

2.5.1 创建数组

在上一个示例中，我使用了PHP生成的数组，但是，你经常希望创建自己的数组。有两种主要方式可用于定义自己的数组。第一种方式是，可以一次添加一个元素来构建数组：

```
$band[] = 'Jemaine';
$band[] = 'Bret';
$band[] = 'Murray';
```

现在，\$band[0]具有一个值Jemaine，\$band[1]具有一个值Bret，\$band[2]具有一个值Murray（由于数组的索引从0开始）。

此外，也可以在添加元素时指定键。但是，重要的是理解，如果指定一个键，并且已经存在用那个相同的键进行索引的一个值，则新值将重写现有的值。例如：

```
$band['fan'] = 'Me!';
$band['fan'] = 'Dave'; // New value
$fruit[2] = 'apple';
$fruit[2] = 'orange'; // New value
```

除了一次添加一个元素这种方式之外，也可以使用array()函数，只用一个步骤即可构建一个完整的数组：

```
$states = array ('IA' => 'Iowa', 'MD' => 'Maryland');
```

(因为PHP会忽略空格，你可以利用这个特性将代码分成多行，以便清晰明了。)

不论是否显式地设置了键，都可以使用array()函数：

```
$artists = array ('Clem Snide', 'Shins', 'Eels');
```

或者，如果你设置了第一个数字键值，那么此后添加的值将是可以递增的键：

```
$days = array (1 => 'Sun', 'Mon', 'Tue');
echo $days[3]; // Tue
```

在引用数组之前，也可使用array()函数初始化它：

```
$tv = array();
$tv[] = 'Flight of the Conchords';
```

在PHP中初始化数组（或者任何变量）不是必需的，但它可以使代码更清晰，并且有助于避免错误。

最后，如果你想创建连续数字的数组，那么可以使用range()函数：

```
$ten = range (1, 10);
```

2.5.2 访问数组

你已经看到如何通过键（例如，\$_POST['email']）访问单个数组元素。当你确切知道键是什么或者如果你只想引用一个元素时，可以这样做。要访问每个数组元素，可以使用foreach循环：

```
foreach ($array as $value) {
    // Do something with $value.
}
```

foreach循环将会迭代\$array中的每个元素，并把每个元素的值赋予\$value变量。要访问键和值，可以使用：

```
foreach ($array as $key => $value) {
    echo "The value at $key is $value.";
}
```

（你可以使用任何有效的变量名称代替\$key和\$value，如果你愿意，可以只使用\$k和\$v）。

我将说明使用数组来建立一组表单下拉菜单以从中选择一个日期有多容易（参见图2-21）。

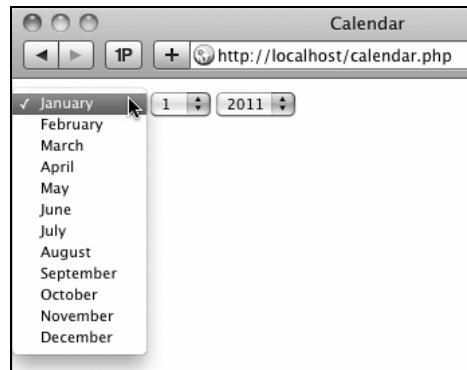


图2-21 这些下拉菜单是使用数组和foreach循环创建的

创建和访问数组

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为calendar.php（参见脚本2-6）。

脚本 2-6 使用数组动态创建 3 个下拉菜单

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
2   DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5    <title>Calendar</title>
6  </head>
7  <body>
8  <form action="calendar.php" method="post">
9  <?php # Script 2.6 - calendar.php
10 // This script makes three pull-down menus
11 // for an HTML form: months, days, years.
12
13 // Make the months array:
14 $months = array (1 => 'January', 'February', 'March', 'April', 'May',      'June', 'July',
15   'August', 'September', 'October', 'November', 'December');
16
17 // Make the days and years arrays:
18 $days = range (1, 31);
19 $years = range (2011, 2021);
20
21 // Make the months pull-down menu:
22 echo '<select name="month">';
23 foreach ($months as $key => $value) {
24   echo "<option value=\"$key\"> $value</option>\n";
25 }
26 echo '</select>';
27
28 // Make the days pull-down menu:
29 echo '<select name="day">';
30 foreach ($days as $value) {
31   echo "<option value=\"$value\"> $value</option>\n";
32 }
33 echo '</select>';
34
35 // Make the years pull-down menu:
36 echo '<select name="year">';
37 foreach ($years as $value) {
38   echo "<option value=\"$value\"> $value</option>\n";
39 }
40 echo '</select>';
41
42 ?>
43 </form>
44 </body>
45 </html>
```

这里要注意的一件事情是，即使页面不包含完整的HTML表单，仍然需要表单标签创建下拉菜单。

(2) 为月份创建一个数组。

```
$months = array (1 => 'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
    'September', 'October', 'November', 'December');
```

这是第一个数组，它将使用数字1~12作为键。因为我指定了第一个键，所以将会对余下的值递增地加索引。换句话说，代码`1 =>`创建一个索引为1~12（而不是0~11）的数组。

(3) 为一月的各天以及年份创建数组。

```
$days = range (1, 31);
$years = range (2011, 2021);
```

使用`range()`函数，可以轻松建立一个数字数组。

(4) 生成月份下拉菜单。

```
echo '<select name="month">';
foreach ($months as $key => $value) {
    echo "<option value=\"$key\"> $value</option>\n";
}
echo '</select>';
```

利用`foreach`循环，可以快速生成月份下拉菜单的所有HTML代码。每执行一次循环，都会创建一行代码，如`<option value="1">January</option>`（参见图2-22）。

```
8 <form action="calendar.php" method="post">
9 <select name="month"><option value="1">January</option>
10 <option value="2">February</option>
11 <option value="3">March</option>
12 <option value="4">April</option>
13 <option value="5">May</option>
14 <option value="6">June</option>
15 <option value="7">July</option>
16 <option value="8">August</option>
17 <option value="9">September</option>
18 <option value="10">October</option>
19 <option value="11">November</option>
20 <option value="12">December</option>
21 </select><select name="day"><option value="1">1</option>
22 <option value="2">2</option>
23 <option value="3">3</option>
24 <option value="4">4</option>
25 <option value="5">5</option>
26 <option value="6">6</option>
27 <option value="7">7</option>
```

图2-22 大多数HTML源代码都是由仅仅几行PHP代码生成的

(5) 生成天和年份下拉菜单。

```
echo '<select name="day">';
foreach ($days as $value) {
    echo "<option value=\"$value\"> $value</option>\n";
}
echo '</select>';
echo '<select name="year">';
foreach ($years as $value) {
    echo "<option value=\"$value\"> $value</option>\n";
}
echo '</select>';
```

与月份示例不同的是，天和年份下拉菜单将为选项的值和标签（一个数字，见图2-22）使用相同的内容。

(6) 关闭PHP、表单标签和HTML页面。

```
?>
</form>
</body>
</html>
```

(7) 将文件另存为calendar.php，存放在Web目录中，并在Web浏览器中测试它。

✓ 提示

□ 要确定数组中元素的个数，可以使用count()函数。

```
$num = count($array);
```

□ range()函数也可以创建连续字母的数组：

```
$alphabet = range ('a', 'z');
```

□ 数组的键可以是由多个单词组成的字符串，比如first name或phone number。

□ is_array()函数可以确认一个变量是数组类型。

□ 如果看到Invalid argument supplied for foreach()（为foreach()提供了无效的参数）出错消息，这意味着你正尝试在不是数组的变量上使用foreach循环。

2.5.3 多维数组

最开始介绍数组时，我提到数组的值可以是数字、字符串甚至其他数组的任意组合。最后这种情况——包含其他数组的数组——将会创建一个多维数组（multidimensional array）。

多维数组比你所想的要普遍得多，但是，使用多维数组非常容易。例如，首先创建质数的数组：

```
$primes = array(2, 3, 5, 7, ...);
```

然后创建楔形数^①的数组：

```
$sphenic = array(30, 42, 66, 70, ...);
```

可以将这两个数组组合进一个多维数组中，如下：

```
$numbers = array ('Primes' => $primes, 'Sphenic' => $sphenic);
```

现在，\$numbers是一个多维数组。要访问质数子数组，可以引用\$numbers['Primes']。要访问质数5，可以使用\$numbers['Primes'][2]（它是数组中的第三个元素，但是数组的索引从0开始）。要打印出这些值之一，可以用花括号包围整个构造：

```
echo "The first sphenic number is {$numbers['Sphenic'][0]}.";
```

当然，你仍然可以使用foreach循环访问多维数组，如果需要的话，可把一个foreach循环嵌套在另一个内部。下一个示例就会这样做。

^① 楔形数指可以表示成三个不同质数的积的正整数。——编者注

使用多维数组

(1) 在文本编辑器中创建新的PHP文档，命名为multi.php（参见脚本2-7）。

脚本 2-7 通过使用其他数组作为值来创建多维数组。使用两个 `foreach` 循环（将一个 `foreach` 循环嵌套在另一个内部）可以访问每个数组元素

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD
2 /xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6   <title>Multidimensional Arrays</title>
7 </head>
8 <body>
9 <p>Some North American States, Provinces, and Territories:</p>
10 <?php # Script 2.7 - multi.php
11 // Create one array:
12 $mexico = array(
13   'YU' => 'Yucatan',
14   'BC' => 'Baja California',
15   'OA' => 'Oaxaca'
16 );
17 // Create another array:
18 $us = array (
19   'MD' => 'Maryland',
20   'IL' => 'Illinois',
21   'PA' => 'Pennsylvania',
22   'IA' => 'Iowa'
23 );
24 //
25 // Create a third array:
26 $canada = array (
27   'QC' => 'Quebec',
28   'AB' => 'Alberta',
29   'NT' => 'Northwest Territories',
30   'YT' => 'Yukon',
31   'PE' => 'Prince Edward Island'
32 );
33 //
34 // Combine the arrays:
35 $n_america = array(
36   'Mexico' => $mexico,
37   'United States' => $us,
38   'Canada' => $canada
39 );
40 //
41 // Loop through the countries:
42 foreach ($n_america as $country => $list) {
43   // Print a heading:
44   echo "<h2>$country</h2><ul>";
45   // Print each state/province/country:
46   foreach ($list as $name => $value) {
47     echo "<li>$value</li>";
48   }
49 }
```

```

47 // Print each state, province, or territory:
48 foreach ($list as $k => $v) {
49     echo "<li>$k - $v</li>\n";
50 }
51
52
53 // Close the list:
54 echo '</ul>';
55
56 } // End of main FOREACH.
57
58 ?>
59 </body>
60 </html>

```

这个PHP页面将打印出在3个北美国家（墨西哥、美国和加拿大）的一些州、省和行政区（参见图2-23）。



图2-23 运行这个PHP页面（参见脚本2-7）的最终结果，其中将会打印每个国家，其后是其州、省和行政区的简写列表

(2) 创建墨西哥的一些州的数组。

```

$mexico = array(
'YU' => 'Yucatan',
'BC' => 'Baja California',
'OA' => 'Oaxaca'
);

```

这是一个关联数组，使用各州的邮政简写词作为其键。州的完整名称是元素的值。这个列表显然不完整，仅用于阐释概念。

(3) 创建第二个和第三个数组。

```
$us = array (
    'MD' => 'Maryland',
    'IL' => 'Illinois',
    'PA' => 'Pennsylvania',
    'IA' => 'Iowa'
);
$canada = array (
    'QC' => 'Quebec',
    'AB' => 'Alberta',
    'NT' => 'Northwest Territories',
    'YT' => 'Yukon',
    'PE' => 'Prince Edward Island'
);
```

(4) 把所有数组合并成一个数组。

```
$n_america = array(
    'Mexico' => $mexico,
    'United States' => $us,
    'Canada' => $canada
);
```

你不必创建三个数组然后把它们赋值给第四个数组，以便创建想要的多维数组。但是我认为这种方式更容易阅读和理解（在一个步骤中定义多维数组将会导致产生一些丑陋的代码）。

`$n_america`数组现在包含三个元素。每个元素的键都是字符串，它是国家的名称。每个元素的值是在那个国家发现的州、省和行政区的列表。

(5) 开始主`foreach`循环。

```
foreach ($n_america as $country => $list) {
    echo "<h2>$country</h2><ul>";
```

依据前面概括的语法，这个循环将访问`$n_america`的每个元素。这意味着这个循环将运行三次。在循环的每次迭代内，`$country`变量将存储`$n_america`数组的键（`Mexico`、`Canada`或`United States`）。此外，在循环的每次迭代内，`$list`变量还将存储元素的值（等价于`$mexico`、`$us`和`$canada`）。

为了打印出结果，循环首先打印`H2`标签内的国家名称。由于州、省和行政区应该显示为HTML列表，所以还应该打印初始无序列表标签（``）。

(6) 创建第二个`foreach`循环。

```
foreach ($list as $k => $v) {
    echo "<li>$k - $v</li>\n";
}
```

这个循环将遍历每个子数组（第一个是`$mexico`，接着是`$us`，然后是`$canada`）。利用这个循环的每次迭代，`$k`将存储简写名称，`$v`将存储完整名称。它们都会打印在HTML列表标签内。还会使用换行符，更好地格式化HTML源代码。

(7) 完成外层`foreach`循环。

```
echo '</ul>';
} // End of main FOREACH.
```

在执行完内层foreach循环后，外层foreach循环必须关闭第(5)步中开始创建的无序列表。

(8) 完成PHP和HTML代码。

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

(9) 将文件另存为multi.php，存放在Web目录中，并在Web浏览器中测试它（参见图2-23）。

(10) 如果你愿意，可以检查HTML源代码，查看PHP创建了什么内容。

✓ 提示

□ 多维数组也可以来自于HTML表单。例如，如果表单具有一系列名称为interests[]的复选框：

```
<input type="checkbox" name="interests[]" value="Music" /> Music
<input type="checkbox" name="interests[]" value="Movies" /> Movies
<input type="checkbox" name="interests[]" value="Books" /> Books
```

接收的PHP页面中的\$_POST是多维的。\$_POST['interests']是一个数组，并用\$_POST['interests'][0]存储第一个选中的复选框的值（例如，Movies），用\$_POST['interests'][1]存储第二个选中的复选框的值（例如，Books），等等。注意：只会把选中的复选框传递到PHP页面。

□ 如果HTML表单的选择菜单允许进行多重选择，也可以以多维数组结束：

```
<select name="interests[]" multiple="multiple">
    <option value="Music">Music </option>
    <option value="Movies">Movies </option>
    <option value="Books">Books </option>
    <option value="Napping">Napping </option>
</select>
```

同样，只会把所选的值传递到PHP页面。

数组和字符串

由于数组和字符串这两种变量类型是如此常用，以至于PHP具有两个函数，可以在字符串和数组之间相互进行转换。

```
$array = explode (separator, $string);
$string = implode (glue, $array);
```

使用和理解这两个函数的关键之处是分隔符（separator）和胶合（glue）关系。当把一个数组转变成一个字符串时，将会设置胶合——将被插入到生成字符串中的数组值之间的字符或代码。相反，当把字符串转变成数组时，要指定分隔符，它用于标记什么应该变成独立数组元素。例如，以字符串开始：

```
$s1 = 'Mon-Tue-Wed-Thu-Fri';
$days_array = explode ('-', $s1);

$days_array变量现在是一个有5个元素的数组，其元素Mon的索引为0，Tue的索引为1，等等。
$s2 = implode (', ', $days_array);

$s2变量现在是一个用逗号分隔的一个星期中各天的列表：Mon, Tue, Wed, Thu, Fri。
```

2.5.4 数组排序

数组相对于其他变量类型的优点之一是，能够对它们进行排序。PHP包括多个可用于对数组排序的函数，它们的语法都很简单：

```
$names = array ('Moe', 'Larry', 'Curly');
sort($names);
```

这些排序函数执行3种排序。首先，可以使用**sort()**按值对数组排序，并丢弃原来的键。重要的是理解在排序过程之后将会重置数组的键，因此，如果键-值关系很重要，就不应该使用这个函数。

其次，可以使用**asort()**按值对数组排序，同时还会维持键。最后，可以使用**ksort()**按键对数组排序。如果把这些函数分别更改为**rsort()**、**arsort()**和**krsort()**，则能够以相反的顺序对数组排序。

为了演示对数组排序的作用，我将创建一个电影名称和评级（用1~10来评定我喜欢它们的程度）的数组，然后以不同的方式显示这个列表。

对数组排序

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为**sorting.php**（参见脚本2-8）。

脚本 2-8 定义一个数组，然后用两种不同的方式对它进行排序：首先按值然后按键（以逆序）进行排序

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD
2  /xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4  <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6  </head>
7  <body>
8  <table border="0" cellspacing="3" cellpadding="3" align="center">
9      <tr>
10         <td><h2>Rating</h2></td>
11         <td><h2>Title</h2></td>
12     </tr>
13 <?php # Script 2.8 - sorting.php
14
15 // Create the array:
16 $movies = array (
17     'Casablanca' => 10,
18     'To Kill a Mockingbird' => 10,
19     'The English Patient' => 2,
20     'Stranger Than Fiction' => 9,
21     'Story of the Weeping Camel' => 5,
22     'Donnie Darko' => 7
23 );
24
25 // Display the movies in their original order:
26 echo '<tr><td colspan="2"><b>In their original order:</b></td></tr>';
27 foreach ($movies as $title => $rating) {
28     echo "<tr><td>$rating</td>
29     <td>$title</td></tr>\n";
30 }
```

```

31
32 // Display the movies sorted by title:
33 ksort($movies);
34 echo '<tr><td colspan="2"><b>Sorted by title:</b></td></tr>';
35 foreach ($movies as $title => $rating) {
36     echo "<tr><td>$rating</td>
37         <td>$title</td></tr>\n";
38 }
39
40 // Display the movies sorted by rating:
41 arsort($movies);
42 echo '<tr><td colspan="2"><b>Sorted by rating:</b></td></tr>';
43 foreach ($movies as $title => $rating) {
44     echo "<tr><td>$rating</td>
45         <td>$title</td></tr>\n";
46 }
47
48 ?>
49 </table>
50 </body>
51 </html>

```

(2) 创建一个HTML表格。

```

<table border="0" cellspacing="3" cellpadding="3" align="center">
<tr>
    <td><h2>Rating</h2></td>
    <td><h2>Title</h2></td>
</tr>

```

为了使有序列表更容易阅读，将在一个HTML表格内打印它。从这里开始创建这个表格。

(3) 添加PHP开始标签并创建一个新数组。

```

<?php # Script 2.8 - sorting.php
$movies = array (
'Casablanca' => 10,
'To Kill a Mockingbird' => 10,
'The English Patient' => 2,
'Stranger Than Fiction' => 9,
'Story of the Weeping Camel' => 5,
'Donnie Darko' => 7
);

```

这个数组使用电影名称作为值，并使用它们各自的评级作为它们的键。这种结构允许用多种可能的方式对整个列表进行排序。可以根据需要自由地更改电影列表和评级（只是不要责备我的电影鉴赏能力）。

(4) 以如下方式打印出数组。

```

echo '<tr><td colspan="2"><b>In their original order:</b> </td></tr>';
foreach ($movies as $title => $rating) {
    echo "<tr><td>$rating</td>
        <td>$title</td></tr>\n";
}

```

此时，在这个脚本中，数组保持创建它的顺序。为了验证这一点，我把它打印出来。首先跨两个表格列打印标题。然后，在`foreach`循环内，在第一列中打印键并在第二列中打印值。还会打印换行符，以改进HTML源代码的可读性。

(5) 按名称以字母顺序对数组排序，并再次打印它。

```
arsort($movies);
echo '<tr><td colspan="2"><b> Sorted by rating:</b></td></tr>';
foreach ($movies as $title => $rating) {
    echo "<tr><td>$rating</td>
        <td>$title</td></tr>\n";
}
```

`ksort()`函数按值对数组排序，同时还会维持键–值关系。余下的代码是第(4)步的重复。

(6) 按评级降序以数字方式对数组排序，并再次打印它。

```
ksort($movies);
echo '<tr><td colspan="2"><b> Sorted by title:</b></td></tr>';
foreach ($movies as $title => $rating) {
    echo "<tr><td>$rating</td>
        <td>$title</td></tr>\n";
}
```

`asort()`函数将按键（但是以升序方式）对数组排序。因为我想颠倒顺序（最高分优先），所以使用`arsort()`。

(7) 完成PHP和HTML代码以及表格。

```
?>
</table>
</body>
</html>
```

(8) 将文件另存为`sorting.php`，存放在Web目录中，并在Web浏览器中测试它（参见图2-24）。

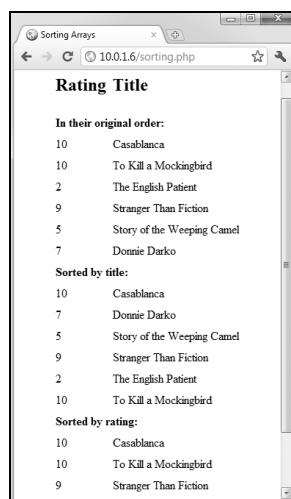


图2-24 本页面演示了可以对数组排序的不同方式

✓ 提示

- 要随机排列数组的顺序，可以使用`shuffle()`。
- PHP的`natsort()`函数可用于以更自然的顺序对数组排序（主要是更好地处理字符串中的数字）。
- PHP通过做一点工作可以对多维数组排序。见PHP手册，了解关于`usort()`函数的更多信息，或者查阅我的*PHP 5 Advanced: Visual QuickPro Guide*一书。

2.6 for 和 while 循环

要讨论的最后一种语言构造是循环。我们在访问数组中的每个元素时，已经使用过一种循环，即`foreach`。下面你将使用的两种循环类型是`for`和`while`。

`while`循环看起来如下所示：

```
while (condition) {
    // Do something.
}
```

只要循环的条件（`condition`）为真，就会执行循环。一旦条件为假，就会停止循环（参见图2-25）。如果条件永远不会为真，就永远不会执行循环。如你将在第8章中所看到的，当从数据库中检索结果时，最常使用`while`循环。

`for`循环具有更复杂的语法：

```
for (initial expression; condition;
closing expression) {
    // Do something.
}
```

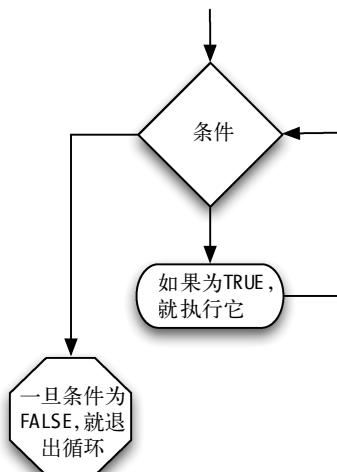


图2-25 PHP处理while循环的流程图表示

在第一次执行循环时，会运行初始表达式（`initial expression`）。然后检查条件，如果条件为真，就

执行循环的内容。执行之后，将会运行结束表达式，并再次检查条件。这个过程会继续下去，直到条件为假（参见图2-26）。例如：

```
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

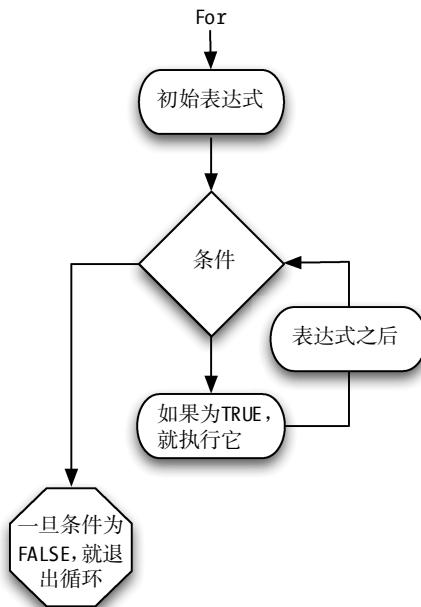


图2-26 PHP处理更复杂的for循环的流程图表示

第一次运行这个循环时，将把*\$i*变量设置为值1。然后检查条件（1小于或等于10吗）。由于这个条件为真，就会打印出1（`echo $i`）。然后，将*\$i*递增为2（`$i++`），再检查条件，以此类推。这个脚本的执行结果将打印出数字1~10。

这两种循环的功能足够相似，因此for和while循环通常能够互换使用。尽管如此，经验表明：如果次数已知，for循环是更好的选择；当条件为真的次数未知时，就使用while循环。

在本章的最后一个示例中，将使用for循环代替两个foreach循环重写早先创建的日历脚本。

使用循环

(1) 在文本编辑器或IDE中打开calendar.php脚本（参见脚本2-6）。

(2) 删除\$days和\$years数组的创建代码（第18~19行）。

使用循环，可以获得两个下拉菜单的相同结果，而不需要创建数组所涉及的额外代码和内存开销。因此，将删除这两个数组，但是仍然保留\$months数组。

(3) 把\$days foreach循环重写为for循环（参见脚本2-9）。

脚本 2-9 循环通常与数组一起使用。这里，用两个 for 循环代替脚本中以前使用的数组和 foreach 循环

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4  <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6  </head>
7  <body>
8  <form action="calendar.php" method="post">
9  <?php # Script 2.9 - calendar.php #2
10
11 // This script makes three pull-down menus
12 // for an HTML form: months, days, years.
13
14 // Make the months array:
15 $months = array (1 => 'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
16                  'September', 'October', 'November', 'December');
17
18 // Make the months pull-down menu:
19 echo '<select name="month">';
20 foreach ($months as $key => $value) {
21     echo "<option value=\"$key\">$value </option>\n";
22 }
23 echo '</select>';
24
25 // Make the days pull-down menu:
26 echo '<select name="day">';
27 for ($day = 1; $day <= 31; $day+ +) {
28     echo "<option value=\"$day\"> $day</option>\n";
29 }
30 echo '</select>';
31
32 // Make the years pull-down menu:
33 echo '<select name="year">';
34 for ($year = 2011; $year <= 2021; $year+ +) {
35     echo "<option value=\"$year\"> $year</option>\n";
36 }
37
38 ?>
39 </form>
40 </body>
41 </html>
```

这个标准的for循环首先将\$day变量初始化为1。它将继续运行循环，直到\$day大于31，并在每次迭代时，将\$day递增1。循环自身的内容（它将执行31次）是一条echo()语句。

(4) 将\$years foreach循环重写为for循环。

```
for ($year = 2011; $year <= 2021; $year+ +) {
    echo "<option value=\"$year\"> $year</option>\n";
}
```

这个循环的结构基本上与前一个for循环相同，但是\$year变量最初被设置为2011（而不是1）。只要\$year小于或等于2021，就会执行循环。在循环内，将会运行echo语句。

(5) 保存文件，存放在Web目录中，并在Web浏览器中测试它（参见图2-27）。

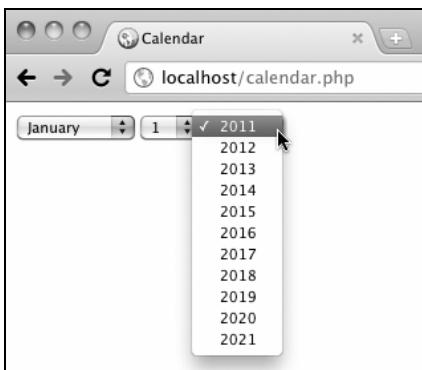


图2-27 日历表单看起来非常像它以前的样子（参见图2-19），但是创建它时，只使用了两个以下的数组（对比脚本2-9与脚本2-6）

✓ 提示

- PHP还有一个do...while循环，其语法稍微有所不同（查看手册）。这个循环总是会至少执行一次。
- 使用循环时，要注意观察参数和条件，避免可怕的无限循环，当循环条件永远不会为假时，就会发生这种情况。

2.7 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书的论坛上，我们会为你答疑解惑。

注意，这里的某些问题和提示涉及第1章介绍的内容，目的是强化里面的重点知识。

2.7.1 回顾

- 表单的method属性的作用是什么？action属性呢？
- 为什么HTML表单提交到一个PHP脚本必须要通过URL加载？如果没有通过URL加载，在提交之后会发生什么问题？
- 使用单引号和双引号描述字符串有什么不同？
- 本章介绍了哪些控制结构？
- 本章介绍了哪些新的变量类型？
- 测试相等的操作符是什么？赋值操作符是什么？
- 为什么文本表单元素使用empty()验证，而其他的表单元素使用isset()验证？
- 索引数组和关联数组的区别是什么？

- 索引数组的索引的初始值（默认情况下）是多少？如果某个索引数组中有10个元素，那么数组中最后一个元素的索引值是多少？
- 什么是超全局数组？下面的这些超全局数组是从哪里获取的值？ 2
- \$_GET
- \$_POST
- \$_COOKIE
- \$_REQUEST
- \$_SESSION
- \$_SERVER
- \$_ENV
- 如何输出索引数组中的某个元素？如何输出关联数组中的某个元素？注意：这两个问题的答案不止一个。
- count()函数有什么作用？
- 在浏览器中打印\n会有什么影响？
- 通常来说，什么时候需要使用while循环？什么时候需要使用for循环？什么时候需要使用foreach循环？每一种循环结构的语法是什么？
- ++是什么操作符？它有什么作用？

2.7.2 实践

- 你正在使用的是哪个版本的PHP？如果你不知道的话，现在就查看一下！
- 创建一个需要用户输入的新表单（或许基于你的某个项目中将会用到的表单），然后创建PHP脚本验证表单数据并报告结果。
- 重写脚本2-4（handle_form.php）中的性别条件语句，使用一个条件语句代替两个嵌套的语句。
提示：你需要使用AND操作符。
- 重写脚本2-4（handle_form.php），使用\$_POST代替\$_REQUEST。
- 重写脚本2-4（handle_form.php），使它可以验证Age元素。
提示：可以参照验证\$gender变量值的方法，检查对应的下拉选项值（0-29、30-60、60+）。
- 重写脚本2-5（handle_form.php最终版）中的echo语句，让它使用单引号和字符串连接代替双引号。
- 在PHP手册中查阅本章介绍的一个数组函数，然后检查PHP中其他与数组相关的函数。
- 创建一个新的数组，然后显示它的所有元素。以几种不同的方式排序数组元素，然后再显示数组内容。
- 创建一个表单，包含一个可以多选的下拉菜单或一组复选框，然后在PHP脚本中输出每个选中的项及选中项的个数。
- 附加难题，修改上面那个PHP脚本（处理多个可选项），让它按字母顺序显示选项。

创建动态Web站点

3

本章内容

- 包含多个文件
- 再论处理HTML表单
- 建立黏性表单
- 创建自己的函数
- 回顾和实践

我们已经掌握了PHP的基本知识，现在开始构建真正的动态Web站点吧。与最初的静态Web站点相比，动态Web站点更容易维护，能更快地对用户做出响应，并且内容能够改变以响应不同的情况。本章将介绍三个新概念，它们都常用于创建更复杂的Web应用程序（除了这些主题之外，第11章还介绍了另外一些主题）。

第一个主题是使用外部文件。这是一个重要的概念，因为更复杂的站点通常需要分隔一些HTML或PHP代码。然后，本章将回过头来介绍处理HTML表单的主题。你将学习这个标准过程的一些新变体。最后，你将学习如何定义和使用自己的函数。

3.1 包含多个文件

到此为止，本书中的每个脚本都由单个文件组成，它包含所有需要的HTML和PHP代码。但是，在你开发更复杂的Web站点时，将看到这种方法有很多局限性。PHP可以很容易地利用外部文件，从而能够把脚本分成各个不同的部分。你将频繁地使用外部文件，从PHP中提取HTML，或者分离出常用的过程。

PHP有4个用于外部文件的函数：`include()`、`include_once()`、`require()`和`require_once()`。为了使用它们，PHP脚本中将包括如下代码行：

```
include_once('filename.php');
require('/path/to/filename.html');
```

使用其中任何一个函数的最终结果都是，获取包含文件（included file）的所有内容，并在那一刻从父脚本（调用该函数的脚本）中删除该文件。包含文件的一个重要属性是，PHP将把包含代码视作HTML（即直接把它发送到浏览器），除非它包含PHP标签内的代码。

从功能上讲，包含文件使用什么扩展名无关紧要，它可以是.php或.html。不过，通过给文件提供

一个象征性的名称有助于传达其目的（例如，HTML的包含文件可能使用`.inc.html`）。另请注意：可以使用指向包含文件的绝对路径或相对路径（参见框注“绝对路径与相对路径”以了解更多信息）。

绝对路径与相对路径

3

在引用任何外部项目（它可以是PHP中的包含文件、HTML中的CSS文档或者图像）时，可以选择使用绝对路径或相对路径。绝对路径从计算机的根目录开始指出文件的位置。不管引用（父）文件的位置是什么，这种路径总是正确的。例如，PHP脚本可以使用如下代码包含文件：

```
include ('C:/php/includes/file.php');
include('/usr/xyz/includes/file.php');
```

假定`file.php`存在于指定的位置，这种包含方式将会工作（除非有任何许可问题）。第二个示例是一个UNIX（和Mac OS X）绝对路径，以防你不熟悉语法。绝对路径总是以像`C:/`或/这样的内容开头。

相对路径使用引用（父）文件作为起点。要移到上一级文件夹，可以一起使用两个句点。要移到一个文件夹中，可使用其名称后面接着一个斜杠。假定当前脚本位于`www/ex1`文件夹中，你想在`www/ex2`中包含一些内容，代码如下：

```
include('../ex2/file.php');
```

相对路径将保持准确，即使移到另一个服务器上，只要文件保持它们的当前关系即可。

`include()`和`require()`函数在正确工作时完全一样，但是在它们失败时会表现得有所不同。如果`include()`函数不工作（由于某种原因而不能包含文件），就会向Web浏览器打印一个警告（参见图3-1），但是脚本会继续运行。如果`require()`失败，就会打印一个错误，并且脚本会中止运行（参见图3-2）。

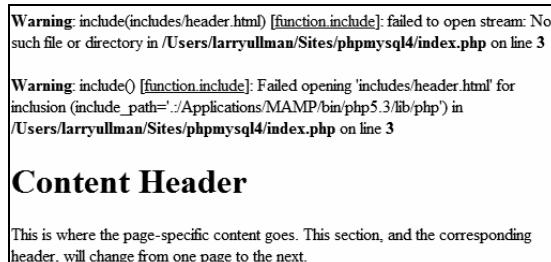


图3-1 两个失败的`include()`调用生成这2条错误消息（假定将PHP配置成显示错误），但是余下的页面会继续执行

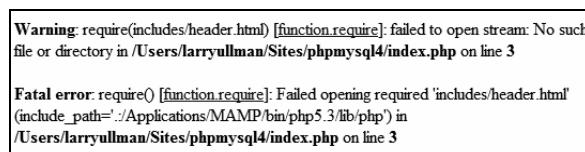


图3-2 `require()`函数调用失败将会打印一个错误，并中止脚本的执行。如果没有把PHP配置成显示错误，那么脚本将终止，而不会首先打印问题（即它将是一个空白页面）

这两个函数还有一个*_once()版本，它们保证处理的文件只会被包含一次，而不管脚本可能（假定不经意地）试图包含它多少次。

```
require_once('filename.php');
include_once('filename.php');
```

在下一个示例中，将使用包含文件把HTML格式化代码与PHP代码隔开。这样，本章余下的示例将具有相同的外观（就好像它们都是相同Web站点的一部分一样），而不必每次都重写HTML代码。这种技术创建了一个模板系统，这是使大型应用程序具有一致性和可管理性的一种容易的方式。这些示例关注的焦点是PHP代码本身。你还应该阅读本章后面的“站点结构”框注，以便理解服务器上的组织模式。如果你有关于示例中使用的CSS或(X)HTML的任何问题，可以参阅关于这些主题的专用资源。

包含多个文件

(1) 在文本编辑器或所见即所得编辑器中设计一个HTML页面（参见脚本3-1和图3-3）。

脚本3-1 用于本章的Web页面的HTML模板。从本书的支持Web站点上下载它使用的style.css文件

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4      <title>Page Title</title>
5      <link rel="stylesheet" href="includes/style.css" type="text/css" media="screen" />
6      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
7  </head>
8  <body>
9      <div id="header">
10         <h1>Your Website</h1>
11         <h2>catchy slogan...</h2>
12     </div>
13     <div id="navigation">
14         <ul>
15             <li><a href="index.php">Home Page</a></li>
16             <li><a href="calculator.php">Calculator</a></li>
17             <li><a href="#">link three</a></li>
18             <li><a href="#">link four</a></li>
19             <li><a href="#">link five</a></li>
20         </ul>
21     </div>
22     <div id="content"><!-- Start of the page-specific content. -->
23         <h1>Content Header</h1>
24
25         <p>This is where the page-specific content goes. This section, and the corresponding
            header, will change from one page to the next.</p>
26
27         <p>Volutpat at varius sed sollicitudin et, arcu. Vivamus viverra. Nullam turpis. Vestibulum
            sed etiam. Lorem ipsum sit amet dolore. Nulla facilisi. Sed tortor. Aenean felis. Quisque
            eros. Cras lobortis commodo metus. Vestibulum vel purus. In eget odio in sapien adipiscing
            blandit. Quisque augue tortor, facilisis sit amet, aliquam, suscipit vitae, cursus sed,
            arcu lorem ipsum dolor sit amet.</p>
```

```

28      <!-- End of the page-specific content. --></div>
29
30
31  <div id="footer">
32      <p>Copyright ©; <a href="#">Plain and Simple</a> 2007 | Designed by <a href="http://
33      www.edg3.co.uk/">edg3.co.uk</a> | Sponsored by <a href="http://www.opendesigns.org/">Open
34      Designs</a> | Valid <a href="http://jigsaw.w3.org/css-validator/">CSS</a> &amp; <a href=
35      "http://validator.w3.org/">XHTML</a></p>
36  </div>
37 </body>
38 </html>

```

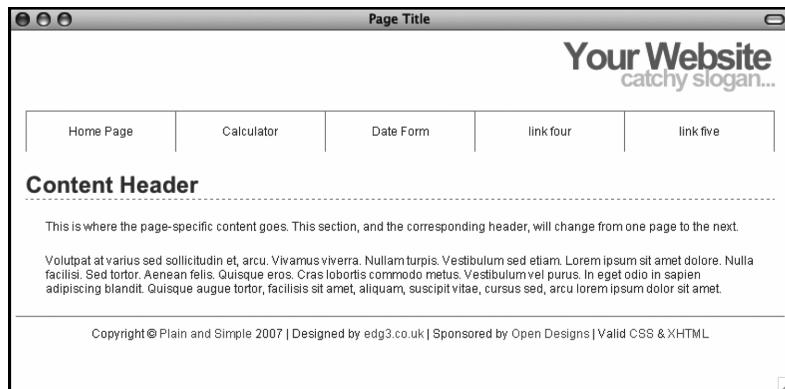


图3-3 HTML和CSS设计得就像它出现在Web浏览器中一样（不使用任何PHP代码）

要开始为Web站点创建一个模板，可以像标准HTML页面那样设计布局，它独立于任何PHP代码。对于本章的示例，我使用了Christopher Robinson创建的“Plain and Simple”模板（www.edg3.co.uk）的一个稍加修改的版本，并且得到了他友善的许可。

(2) 标记出页面特有的任何内容所在的位置。

几乎每个Web站点在每个页面上都有几个常用元素（标题、导航、广告、脚注等）以及一个或多个页面特有的部分。在HTML页面中（参见脚本3-1），把因页面而异的布局部分括在HTML注释内，以指示其状态。

(3) 复制从布局源代码的第一行到紧接着页面特有的内容之前的所有内容，并将其粘贴到一个新文档中（参见脚本3-2）。

脚本 3-2 每个 Web 页面的初始 HTML 代码将存储在头文件中

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
2      xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head>
5      <title><?php echo $page_title; ?></title>
6      <link rel="stylesheet" href="includes/style.css" type="text/css" media="screen" />
7      <meta http-equiv="content-type" content="text/html; charset=utf-8" />
8  </head>

```

```

9   <div id="header">
10    <h1>Your Website</h1>
11    <h2>catchy slogan...</h2>
12  </div>
13  <div id="navigation">
14    <ul>
15      <li><a href="index.php">Home Page</a></li>
16      <li><a href="calculator.php">Calculator</a></li>
17      <li><a href="#">link three</a></li>
18      <li><a href="#">link four</a></li>
19      <li><a href="#">link five</a></li>
20    </ul>
21  </div>
22  <div id="content"><!-- Start of the page-specific content. -->
23 <!-- Script 3.2 - header.html -->
```

第一个文件将包含初始HTML标签（从DOCTYPE通过头部并进入到页面主体的开始处）。它还具有用于建立Web站点名称、标语以及横跨顶部的水平链接条的代码（参见图3-3）。最后，由于每个页面的内容位于其id值为content的DIV中，所以这个文件也包含这段代码。

(4) 更改页面的标题行，以读取

```
<?php echo $page_title; ?>
```

页面的标题（它出现在Web浏览器的顶部，参见图3-3）对于各个页面应该是可变化的。为了做到这一点，我把它设置成一个变量，通过PHP将其打印出来。稍后你将看到其工作方式。

(5) 将文件另存为header.html。

如前所述，包含文件可以为文件名使用几乎任何扩展名。因此将该文件命名为header.html，指示它是模板的头文件并且它（主要）包含HTML代码。

(6) 复制原始模板中从页面特有的内容末尾到页面末尾的所有内容，并将其粘贴到一个新文件中（参见脚本3-3）。

脚本 3-3 用于每个 Web 页面的最终 HTML 代码将存储在这个脚注文件中

```

1  <!-- Script 3.3 - footer.html -->
2  <!-- End of the page-specific content. --></div>
3
4  <div id="footer">
5    <p>Copyright &copy; <a href="#">Plain and Simple</a> 2007 | Designed by <a href="http://
     www.edg3.co.uk/">edg3.co.uk</a> | Sponsored by <a href="http://www.opendesigns.org/">Open
     Designs</a> | Valid <a href="http://jigsaw.w3.org/css-validator/">CSS</a> &amp; <a href=
     "http://validator.w3.org/">XHTML</a></p>
6  </div>
7  </body>
8  </html>
```

在关闭头文件中打开的内容DIV之后，就开始了脚注文件（参见第(3)步）。然后添加脚注，站点中的每个页面上的脚注都相同，并完成HTML文档本身。

(7) 将文件另存为footer.html。

(8) 在文本编辑器或IDE中创建一个新的PHP文档（参见脚本3-4）。

```
<?php # Script 3.4 - index.php
```

由于这个脚本将为其大多数HTML使用包含文件，所以它可以在开始和结束于PHP标签。

脚本 3-4 这个脚本使用存储在两个外部文件中的模板生成一个完整的 Web 页面

```

1  <?php # Script 3.4 - index.php
2  $page_title = 'Welcome to this Site!';
3  include ('includes/header.html');
4  ?>
5
6  <h1>Content Header</h1>
7
8  <p>This is where the page-specific content goes. This section, and the corresponding header,
   will change from one page to the next.</p>
9
10 <p>Volutpat at varius sed sollicitudin et, arcu. Vivamus viverra. Nullam turpis. Vestibulum
    sed etiam. Lorem ipsum sit amet dolore. Nulla facilisi. Sed tortor. Aenean felis. Quisque
    eros. Cras lobortis commodo metus. Vestibulum vel purus. In eget odio in sapien adipiscing
    blandit. Quisque augue tortor, facilisis sit amet, aliquam, suscipit vitae, cursus sed, arcu
    lorem ipsum dolor sit amet.</p>
11
12 <?php
13 include ('includes/footer.html');
14 ?>
```

(9) 设置\$page_title变量，并且包含HTML头文件。

```
$page_title = 'Welcome to this Site!';
include ('includes/header.html');
```

\$page_title将存储出现在浏览器窗口顶部的值（因此，它也是人们为页面建立书签时的默认值）。该变量打印在header.html中（参见脚本3-2）。通过在包含头文件之前定义这个变量，头文件将能够访问它。记住：include()这一行具有把包含文件的内容放入该页面中这个位置的作用。

include()函数调用使用指向header.html的相对路径（参见框注“绝对路径与相对路径”）。语法指出在包含该文件的相同文件夹中有一个名为includes的文件夹，并且该文件夹中有一个名为header.html的文件。

(10) 关闭PHP标签，并添加页面特有的内容。

```
?>
<h1>Content Header</h1>
<p>This is where the page-specific content goes. This section, and the corresponding header, will
   change from one page to the next.</p>
```

对于大多数页面，PHP将生成这个内容，而不是具有静态文本。可以使用echo()把这段信息发送给浏览器，但是，因为这里没有动态内容，暂时退出PHP标签将会使操作更容易、更高效。

(11) 创建最终的PHP部分，并包含脚注文件。

```
<?php
include ('includes/footer.html');
?>
```

(12) 将文件另存为index.php，并存放在Web目录中。

(13) 在与index.php相同的文件夹中创建一个includes目录。然后把header.html、footer.html和style.css（从www.LarryUllman.com下载）放到这个includes目录中。

注意，为了节省篇幅，本书没有包括用于本示例的CSS文件（它控制布局）。你可以通过本书的支持Web站点下载该文件（见该站点的Extras页面），或者不使用该文件（模板仍会工作，只是它看起来不那么美观）。

(14) 通过Web浏览器进入index.php页面来测试模板系统（参见图3-4）。

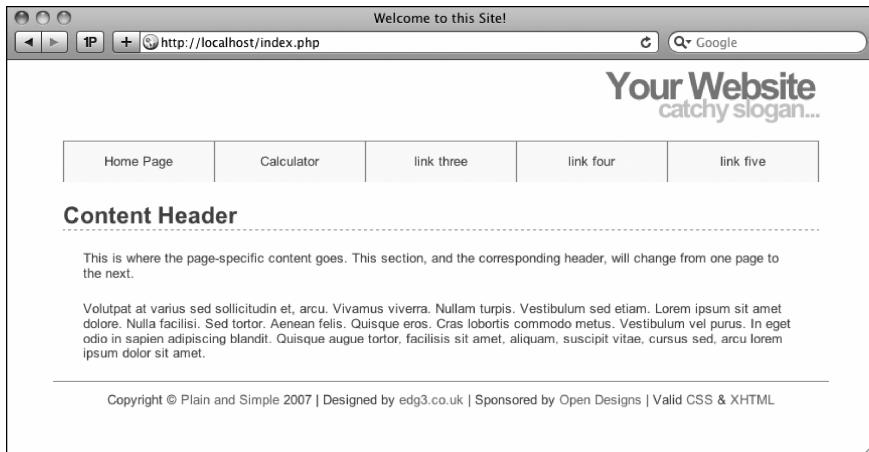


图3-4 现在使用PHP中的外部文件创建相同的布局（参见图3-3）

index.php页面是这个模板系统的最终结果。你不必直接访问任何包含文件，因为index.php将负责包含它们的内容。由于这是一个PHP页面，所以仍然需要通过URL访问它。

(15) 如果需要，可查看页面的HTML源文件（参见图3-5）。

```
Source of http://localhost/index.php
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Welcome to this Site!

```

图3-5 生成的Web页面的HTML源文件应该复制原始模板中的代码（参见脚本3-1）

✓ 提示

- 在 `php.ini` 配置文件中，可以调整 `include_path` 设置，它指示是否允许 PHP 检索包含文件。
- 如你将在第 9 章中所看到的，包含敏感信息（如数据库访问）的任何包含文件都应该存储在 Web 文档目录外部，使得不能在 Web 浏览器内查看它。
- 因为在 `require()` 失败时，会对脚本产生更多的影响，所以我建议把它用于任务关键的包含文件（像那些连接到数据库的包含文件），为不怎么重要的包含文件使用 `include()`。
- 如果 PHP 代码段只包含一行语句，一般将这条语句与 PHP 标签写在同一行：

```
<?php include ('filename.html'); ?>
```

- 由于 CSS 的工作方式，如果不使用 CSS 文件或者如果浏览器不能读取 CSS，那么生成的结果仍然是实用的，只不过从审美观点上讲不那么令人愉悦。

站点结构

当你开始在 Web 应用程序中使用多个文件时，总体站点结构将变得更重要。在对站点进行布局时，有两个考虑事项：

- 易于维护；
- 安全性。

使用外部文件保存标准过程（即 PHP 代码）、CSS、JavaScript 和 HTML 设计，将极大地改进维护站点的简易性，因为共同编辑的代码都放置在一个中央位置。我将频繁建立 `includes` 或 `templates` 目录来存储这些文件，将其与主要的脚本（直接在 Web 浏览器中访问它们）隔离开。

我建议为安全性不成问题的文档（如 HTML 模板）使用 `.inc` 或 `.html` 文件扩展名，为那些包含更多敏感数据（如数据库访问信息）的文档使用 `.php` 扩展名。你也可以同时使用 `.inc` 和 `.html` 或 `.php`，以将一个文件明显地指定为某种类型的包含文件，如 `db.inc.php` 或 `header.inc.html`。

3.2 再论处理 HTML 表单

第 2 章中一个很好的部分涉及用 PHP 处理 HTML 表单。其中所有的示例都使用了两个单独的文件：一个用于显示表单，另一个用于接收表单。虽然这种方法肯定没有任何错误，但是把整个过程集中到一个脚本中更有利。

为了让一个页面同时显示和处理表单，必须使用一个条件语句检查应该采取哪种动作（显示或处理）：

```
if /* form has been submitted */ {
    // Handle the form.
} else {
    // Display the form.
}
```

接下来的问题是判断表单是否已提交，这只需稍加解释即可。

如果表单使用 POST 方法并且提交回原表单，那么脚本会产生两类请求（参见图 3-6）。第一个是加载表单的 GET 请求。这是大多数页面生成的标准请求。当表单提交后，会产生第二个请求——POST（只要表单使用 POST 方法）。了解以上内容后，你可以通过检查请求方法 (`$_SERVER` 数组) 判断表单提交状态：

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Handle the form.
} else {
    // Display the form.
}

```

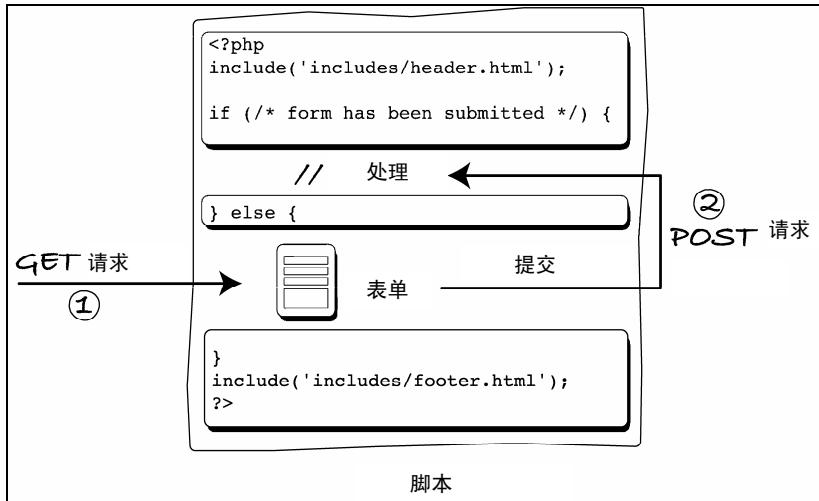


图3-6 用户与服务器上的这个PHP脚本之间的交互涉及用户建立对这个脚本的两个请求

如果想让页面处理表单，然后再次显示它（例如，添加一条记录到数据库中，然后给出一个选项用于添加另一条记录），则可省略else子句：

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Handle the form.
}
// Display the form.

```

使用上面的代码，如果提交了表单，脚本将会处理它们，并会在每次加载页面时显示该表单。

为了演示这种重要的技术（让同一个页面显示和处理表单），让我们创建一个简单的销量计算器（基于用户输入的数据，参见图3-7）。

Trip Cost Calculator

Distance (in miles):

Ave. Price Per Gallon: 3.00 3.50 4.00

Fuel Efficiency:

图3-7 HTML表单，由用户填写内容

处理HTML表单

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为calculator.php（参见脚本3-5）。

脚本 3-5 calculator.php 脚本将会显示一个简单的表单并处理表单数据：执行一些计算并报告结果

```

1  <?php # Script 3.5 - calculator.php
2
3  $page_title = 'Trip Cost Calculator';
4  include ('includes/header.html');
5
6  // Check for form submission:
7  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
8
9      // Minimal form validation:
10     if (isset($_POST['distance'], $_POST['gallon_price'], $_POST['efficiency']) &&
11         is_numeric($_POST['distance']) && is_numeric($_POST['gallon_price']) && is_numeric($_POST
12         ['efficiency'])) {
13
14         // Calculate the results:
15         $gallons = $_POST['distance'] / $_POST['efficiency'];
16         $dollars = $gallons * $_POST['gallon_price'];
17         $hours = $_POST['distance']/65;
18
19         // Print the results:
20         echo '<h1>Total Estimated Cost</h1>
21         <p>The total cost of driving ' . $_POST['distance'] . ' miles, averaging ' . $_POST['efficiency'] .
22         ' miles per gallon, and paying an average of $' . $_POST['gallon_price'] . ' per gallon, is $' .
23         number_format ($dollars, 2) . '. If you drive at an average of 65 miles per hour, the trip will
24         take approximately ' . number_format($hours, 2) . ' hours.</p>';
25     }
26
27 } // End of main submission IF.
28
29 // Leave the PHP section and create the HTML form:
30 ?>
31
32 <h1>Trip Cost Calculator</h1>
33 <form action="calculator.php" method="post">
34     <p>Distance (in miles): <input type="text" name="distance" /></p>
35     <p>Ave. Price Per Gallon: <span class="input">
36         <input type="radio" name="gallon_price" value="3.00" /> 3.00
37         <input type="radio" name="gallon_price" value="3.50" /> 3.50
38         <input type="radio" name="gallon_price" value="4.00" /> 4.00
39     </span></p>
40     <p>Fuel Efficiency: <select name="efficiency">
41         <option value="10">Terrible</option>
42         <option value="20">Decent</option>
43         <option value="30">Very Good</option>
44         <option value="50">Outstanding</option>
```

```

45    </select></p>
46    <p><input type="submit" name="submit" value="Calculate!" /></p>
47  </form>
48
49  <?php include ('includes/footer.html'); ?>

```

这个示例以及本章中余下的所有示例都将使用与index.php（参见脚本3-4）相同的模板系统。因此，每个页面的开始语法将是相同的，但是页面标题将不同。

(2) 编写处理表单的条件语句。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

如前所述，通过检查POST方法判断页面是否被请求，这是非常好的测试表单提交的手段（前提是表单使用的是POST方法）。

(3) 验证表单。

```
if (isset($_POST['distance'], $_POST['gallon_price'], $_POST['efficiency']) &&
    is_numeric($_POST['distance']) && is_numeric($_POST['gallon_price']) && is_numeric($_POST
    ['efficiency'])) {
```

这里的验证非常简单：它只会检查3个提交的变量是否都是数字类型。你当然可以详细说明它，可能检查数量是一个整数并且所有的值都为正（事实上，在第13章中，你将发现这个脚本的一个变体，它正好用于执行这项任务）。

如果验证通过了所有的测试，将会执行计算；否则，将要求用户重试。

(4) 执行计算。

```
$gallons = $_POST['distance'] / $_POST['efficiency'];
$dollars = $gallons * $_POST['gallon_price'];
$hours = $_POST['distance']/65;
```

第一行计算旅程所用的汽油，通过将距离除以燃油效率得到。第二行计算旅程要花费的燃油，通过将加仑数乘以每加仑单价得到。第三行计算旅程需花费的时间，通过将距离除以65（表示每小时65英里）获得。

(5) 打印结果。

```
echo '<h1>Total Estimated Cost</h1>
<p>The total cost of driving '. $_POST['distance'] . ' miles, averaging '. $_POST ['efficiency'] .
    ' miles per gallon, and paying an average of $' . $_POST['gallon_price'] . ' per gallon, is $'.
    number_format ($dollars, 2) . '. If you drive at an average of 65 miles per hour, the trip will
    take approximately ' . number_format($hours, 2) . ' hours.</p>';
```

所有的值都会打印出来，并用number_format()函数格式化价格和总额。使用连接运算符（句点）允许将格式化的数值追加到打印的消息中。

(6) 完成条件语句并关闭PHP标签。

```

} else { // Invalid submitted values.
echo '<h1>Error!</h1>
<p class="error">Please enter a valid distance, price per gallon, and fuel efficiency.</p>';
}
} // End of main submission IF.
?>
```

`else`子句完成了验证条件语句(第(3)步),如果3个提交的值不全都是数字,就会打印一个错误(参见图3-8)。最终的结束花括号会关闭`isset($_SERVER['REQUEST_METHOD']=='POST')`条件语句。最终,会关闭PHP部分,使得无需使用`echo`即可创建表单(第(7)步)。

The screenshot shows a web page titled "Error!" with the message "Please enter a valid distance, price per gallon, and fuel efficiency." Below this, there is a "Trip Cost Calculator" section. It contains two input fields: "Distance (in miles)" and "Ave. Price Per Gallon". The "Distance" field has a dotted border and is empty. The "Ave. Price Per Gallon" field also has a dotted border and contains three radio buttons labeled "3.00", "3.50", and "4.00". The number "4.00" is highlighted with a solid border, indicating it is the selected value.

3

图3-8 如果提交的值不是数字,就会显示错误信息

(7) 显示HTML表单。

```
<h1>Trip Cost Calculator</h1>
<form action="calculator.php" method="post">
    <p>Distance (in miles): <input type="text" name="distance" /> </p>
```

表单自身相当容易理解,其中只包含一个新技巧:`action`属性使用这个脚本的名称,使得表单提交到这个页面,而不是提交到另一个页面。表单的第一个元素是文本输入框,用户可以在里面输入旅行的距离。

(8) 完成表单:

```
<p>Ave. Price Per Gallon: <span class="input">
    <input type="radio" name="gallon_price" value="3.00" /> 3.00
    <input type="radio" name="gallon_price" value="3.50" /> 3.50
    <input type="radio" name="gallon_price" value="4.00" /> 4.00
</span></p>
<p>Fuel Efficiency: <select name="efficiency">
    <option value="10">Terrible </option>
    <option value="20">Decent </option>
    <option value="30">Very Good</option>
    <option value="50"> Outstanding</option>
</select></p>
<p><input type="submit" name="submit" value="Calculate!" /> </p>
</form>
```

(9) 包含脚注文件。

```
<?php include ('includes/ footer.html'); ?>
```

(10) 将文件另存为calculator.php, 存放在Web目录中, 并在Web浏览器中测试它(参见图3-9)。

The total cost of driving 225 miles, averaging 20 miles per gallon, and paying an average of \$3.50 per gallon, is \$39.38. If you drive at an average of 65 miles per hour, the trip will take approximately 3.46 hours.

Trip Cost Calculator

Distance (in miles):

Ave. Price Per Gallon: 3.00 3.50 4.00

图3-9 该页面执行计算，报告结果，然后重新显示表单

✓ 提示

□ 你还可以通过使用无值的action属性，让表单提交回它自身。

```
<form action="" method="post">
```

通过这样做，表单将总是提交回这个相同的页面，即使你往后更改了脚本的名称也会如此。

3.3 建立黏性表单

黏性表单（sticky form）只是一种标准的HTML表单，它能记住你是如何填写它的。对最终用户来说，这是一种特别好的特性，尤其是当第一次非正确地填写它们之后，你要求他们重新提交表单时则更是如此，如图3-8所示。

要预先设置文本框中输入的内容，可使用它的value属性：

```
<input type="text" name="city" value="Innsbruck" />
```

要让PHP预先设置该值，可打印相应的变量（这假定存在被引用的变量）：

```
<input type="text" name="city" value="php echo $city; ?" />
```

（这个示例也体现了PHP是HTML嵌入的这个性质的好处：可以把PHP代码放在任何位置，包括放在表单元素内。）

为了预先设置单选按钮或复选框的状态（即预先检查它们），可以把代码checked="checked"添加到它们的输入标签中。使用PHP，可以编写如下代码：

```
<input type="radio" name="gender" value="F" <?php if ($gender == 'F') {  
    echo 'checked="checked"';  
} ?>/>
```

（你可以看到，语句可以很快地完成。创建表单元素，然后再添加PHP代码，就这么简单。）

为了预先设置textarea的值，可以把该值放在textarea标签之间：

```
<textarea name="comments" rows="10" cols="50"><?php echo $comments; ?></textarea>
```

注意：这里的textarea标签不像标准的text输入框那样具有value属性。

为了预先选择下拉菜单，可以把代码selected="selected"添加到合适的选项中。如果你也使用

PHP生成菜单，这实际上非常容易：

```

echo '<select name="year">';
for ($y = 2011; $y <= 2021; $y++) {
    echo "<option value=\"$y\"";
    if ($year == $y) {
        echo ' selected="selected"';
    }
    echo ">$y</option>\n";
}
echo '</select>';

```

记住这些新信息，让我们重写calculator.php，以使它具有黏性。与前面的示例不同，现存的值放在\$_POST变量中。由于引用的变量要求含有值，因而使用条件语句先检查变量是否设置后再打印它的值。

建立黏性表单

- (1) 在文本编辑器或IDE中打开calculator.php（参见脚本3-5）。
- (2) 更改数量输入框以读取（参见脚本3-6）

脚本 3-6 计算器的表单现在会回忆起以前输入的值（创建黏性表单）

```

1  <?php # Script 3.6 - calculator.php #2
2
3  $page_title = 'Trip Cost Calculator';
4  include ('includes/header.html');
5
6  // Check for form submission:
7  if ($_SERVER['REQUEST_METHOD'] == 'POST') {
8
9      // Minimal form validation:
10     if (isset($_POST['distance'], $_POST['gallon_price'], $_POST['efficiency']) &&
11         is_numeric($_POST['distance']) && is_numeric($_POST['gallon_price']) && is_numeric($_
12         POST['efficiency'])) {
13
14         // Calculate the results:
15         $gallons = $_POST['distance'] / $_POST['efficiency'];
16         $dollars = $gallons * $_POST['gallon_price'];
17         $hours = $_POST['distance']/65;
18
19         // Print the results:
20         echo '<h1>Total Estimated Cost</h1>
21         <p>The total cost of driving ' . $_POST['distance'] . ' miles, averaging ' . $_POST
22         ['efficiency'] . ' miles per gallon, and paying an average of $' . $_POST['gallon_price'] .
23         ' per gallon, is $' . number_format($dollars, 2) . '. If you drive at an average of 65 miles
24         per hour, the trip will take approximately ' . number_format($hours, 2) . ' hours.</p>';
25
26     } else { // Invalid submitted values.
27         echo '<h1>Error!</h1>
28         <p class="error">Please enter a valid distance, price per gallon, and fuel efficiency.</p>';
29     }
30
31 } // End of main submission IF.

```

```

28
29 // Leave the PHP section and create the HTML form:
30 ?>
31
32 <h1>Trip Cost Calculator</h1>
33 <form action="calculator.php" method="post">
34   <p>Distance (in miles): <input type="text" name="distance" value=<?php if (isset($_POST
35     ['distance'])) echo $_POST['distance']; ?>></p>
36   <p>Ave. Price Per Gallon: <span class="input">
37     <input type="radio" name="gallon_price" value="3.00" <?php if (isset($_POST ['gallon_
38       price']) && ($_POST['gallon_price'] == '3.00')) echo 'checked="checked" ' ; ?>/> 3.00
39     <input type="radio" name="gallon_price" value="3.50" <?php if (isset($_POST ['gallon_price']) )
40       && ($_POST ['gallon_price'] == '3.50')) echo 'checked="checked" ' ; ?>/> 3.50
41     <input type="radio" name="gallon_price" value="4.00" <?php if (isset($_POST ['gallon_price']) )
42       && ($_POST ['gallon_price'] == '4.00')) echo 'checked="checked" ' ; ?>/> 4.00
43   </span></p>
44   <p>Fuel Efficiency: <select name="efficiency">
45     <option value="10"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '10')) 
46       echo ' selected="selected"'; ?>>Terrible</option>
47     <option value="20"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '20')) 
48       echo ' selected="selected"'; ?>>Decent</option>
49     <option value="30"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '30')) 
50       echo ' selected="selected"'; ?>>Very Good</option>
51     <option value="50"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '50')) 
52       echo ' selected="selected"'; ?>>Outstanding</option>
53   </select></p>
54   <p><input type="submit" name="submit" value="Calculate!" /></p>
55 </form>
56
57 <?php include ('includes/footer.html'); ?>

```

第一个变化是添加value属性给输入框。然后，打印出提交的距离变量（\$_POST ['distance']）的值。自从第一次加载页面起，\$_POST ['distance']就没有值，可使用一个条件语句确保在尝试打印该变量之前先设置它。设置输入框的值的最终结果是如下PHP代码：

```

<?php
if (isset($_POST['distance'])) {
    echo $_POST['distance'];
}
?>

```

我把它精简成脚本中使用的最小的表单（如果条件块内只有一条语句，那么可以省略花括号，尽管我极少建议你这样做）。

(3) 修改单选按钮相关代码。

```

<input type="radio" name="gallon_price" value="3.00" <?php if (isset($_POST['gallon_price']) &&
($_POST['gallon_price'] == '3.00')) echo 'checked="checked" ' ; ?>/> 3.00 <input type="radio" name=
"gallon_price" value="3.50" <?php if (isset($_POST['gallon_price']) && ($_POST['gallon_price'] ==
'3.50')) echo 'checked="checked" ' ; ?>/> 3.50 <input type="radio" name="gallon_price" value="4.00" <?
php if (isset($_POST['gallon_price']) && ($_POST['gallon_price'] == '4.00')) echo 'checked
="checked" ' ; ?>/> 4.00

```

对于每个单选按钮，将以下代码放在input标签之中。

```
<?php if (isset($_POST['gallon_price'])) && ($_POST['gallon_price'] == 'XXX')) echo 'checked = "checked"'; ?>
```

根据相应单选按钮的值，改变条件语句中的对比值 (XXX)。

(4) 修改下拉菜单相关代码。

```
<option value="10"<?php if (isset($_POST['efficiency'])) && ($_POST['efficiency'] == '10')) echo ' selected="selected"'; ?>>Terrible</option>
<option value="20"<?php if (isset($_POST['efficiency'])) && ($_POST['efficiency'] == '20')) echo ' selected="selected"'; ?>>Decent</option>
<option value="30"<?php if (isset($_POST['efficiency'])) && ($_POST['efficiency'] == '30')) echo ' selected="selected"'; ?>>Very Good</option>
<option value="50"<?php if (isset($_POST['efficiency'])) && ($_POST['efficiency'] == '50')) echo ' selected="selected"'; ?>>Outstanding</option>
```

对于每个下拉选项，在option标签之中添加以下代码。

```
<?php if (isset($_POST ['efficiency'])) && ($_POST ['efficiency'] == 'XX')) echo ' selected= "selected"'; ?>
```

同样，根据相应下拉选项的值，改变条件语句中的对比值 (XX)。

(5) 将文件另存为calculator.php，存放在页面目录中，并在浏览器中测试它（参见图3-10和图3-11）。

The screenshot shows a web form titled "Total Estimated Cost". Below it is a note: "The total cost of driving 425 miles, averaging 30 miles per gallon, and paying an average of \$3.50 per gallon, is \$49.58. If you drive at an average of 65 miles per hour, the trip will take approximately 6.54 hours." The form itself is titled "Trip Cost Calculator". It contains three input fields: "Distance (in miles)" with the value "425", "Ave. Price Per Gallon" with radio buttons for "3.00", "3.50", and "4.00" (the "3.50" option is selected), and "Fuel Efficiency" with a dropdown menu showing "Very Good" (the "Very Good" option is selected). All fields show the values submitted in the previous step.

图3-10 表单现在会记住上一次提交的值

The screenshot shows a web form titled "Error!". A message below the title says "Please enter a valid distance, price per gallon, and fuel efficiency." The form is titled "Trip Cost Calculator". It contains three input fields: "Distance (in miles)" with the value "garbage", "Ave. Price Per Gallon" with radio buttons for "3.00", "3.50", and "4.00" (the "3.00" option is selected), and "Fuel Efficiency" with a dropdown menu showing "Terrible" (the "Terrible" option is selected). All fields are marked as invalid due to being empty or containing invalid data.

图3-11 检查表单是否填写完整

✓ 提示

- 由于Price Per Gallon（每加仑价格）和Fuel Efficiency（燃油效率）都是数值，因此在&&条件语句中的比较值既可以用引号也可以不用引号括起来。我将它们用引号括了起来，因为从技术上讲，它们实际上是带数值的字符串。
- 由于这个示例中的一些PHP代码存在于HTML表单的value属性内，出错消息可能不明显。如果出现问题，可以检查页面的HTML源文件，查看PHP错误是否打印在value属性内。
- 应该总是用双引号括住HTML属性，特别是表单输入框的value属性。如果没有这样做，由多个单词组成的值（如Elliott Smith）在Web浏览器中只会显示为Elliott。
- 一些浏览器也可以记住用户在表单中输入的值，这是在使用PHP完成这个功能时独立存在的、可能会重叠的问题。

3.4 创建自己的函数

PHP具有许多内置函数，可以应对几乎所有的需要。不过，更重要的是，无论出于何种目的，PHP都能够让你定义和使用你自己的函数。建立自己的函数的语法如下：

```
function function_name () {  
    // Function code.  
}
```

你的函数名称可以是字母、数字和下划线的任意组合，但是它必须以字母或下划线开头。你还能为自己的函数使用现有的函数名（print、echo、isset，等等）。一个完全有效的函数定义如下：

```
function do_nothing() {  
    // Do nothing.  
}
```

如第1章中所述，在PHP中，函数名称不区分大小写（与变量名称不同），因此可以使用do_Nothing()、DO NOTHING()或Do Nothing()等（但是不能使用donothing()或DoNothing()）调用该函数。

函数内的代码几乎可以做任何事情，从生成HTML代码到预先格式化计算。在本章中，我将介绍几个示例，在本书余下部分，你还将看到其他一些示例。

创建自定义函数的最常见目的如下所述：

- 将重复代码关联到一个函数调用中；
- 将敏感的或复杂的过程独立出来；
- 让常用代码更易于重用。

本章会创建几个自定义函数。你还会在本书的其余部分看到很多自定义函数。以下是第一个示例，这个自定义函数的作用是用HTML代码生成一些“伪广告”。这个函数将在主页上调用两次（参见图3-12）。

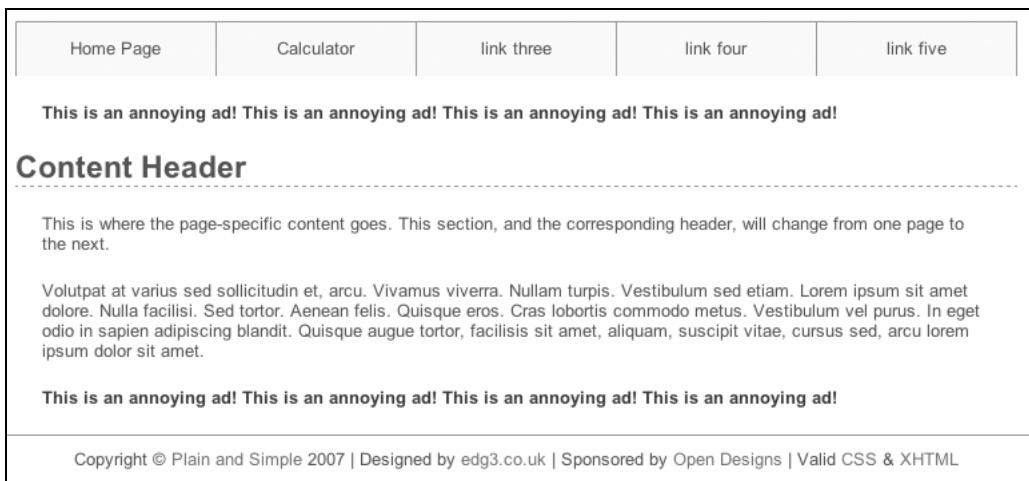


图3-12 通过调用自定义函数生成两个“伪广告”

创建自定义函数

(1) 在文本编辑器或IDE中打开index.php (参见脚本3-4)。

(2) 在PHP起始标签后, 定义一个新函数 (参见脚本3-7)。

在这里创建函数的目的是生成HTML代码, 以便为网页添加广告。这个函数名清楚地描述了该函数的作用。

尽管不强制这样做, 但是习惯把函数定义放在靠近脚本顶部的位置, 或者放在一个单独的文件中。

脚本 3-7 这个版本的主页包含一个用户定义函数, 以输出“伪广告”。这个函数在脚本中被调用两次, 创建两个广告

```

1  <?php # Script 3.7 - index.php #2
2
3 // This function outputs theoretical HTML
4 // for adding ads to a Web page.
5 function create_ad() {
6     echo '<p class="ad">This is an annoying ad! This is an annoying ad! This is an annoying ad!
7         This is an annoying ad!</p>';
8 } // End of the function definition.
9
10 $page_title = 'Welcome to this Site!';
11 include ('includes/header.html');
12
13 // Call the function:
14 create_ad();
15
16 <h1>Content Header</h1>
17
18 <p>This is where the page-specific content goes. This section, and the corresponding header,
    will change from one page to the next.</p>
```

```

19
20   <p>Volutpat at varius sed sollicitudin et, arcu. Vivamus viverra. Nullam turpis. Vestibulum
21   sed etiam. Lorem ipsum sit amet dolore. Nulla facilisi. Sed tortor. Aenean felis. Quisque eros.
22   Cras lobortis commodo metus. Vestibulum vel purus. In eget odio in sapien adipiscing blandit.
23   Quisque augue tortor, facilisis sit amet, aliquam, suscipit vitae, cursus sed, arcu lorem ipsum
24   dolor sit amet.</p>
25
26
27 include ('includes/footer.html');
28 ?>
```

(3) 生成HTML。

```
echo '<p class="ad">This is an annoying ad! This is an annoying ad! This is an annoying ad! This is an
annoying ad!</p>';
```

在真实的函数中，代码应该输出真正的HTML而不是一段文本。（实际的HTML应该由你用来生成和跟踪广告的服务商提供。）

(4) 闭合函数定义。

```
} // End of the function definition.
```

在函数定义的末尾加上一条注释是很好的习惯，这样你就可以知道定义开始和终止的具体位置。
(在定义代码很长时，它的帮助会更大。)

(5) 在引入头部文件（header.html）的后面和现有的PHP代码块的前面，调用函数。

```
create_ad();
```

调用create_ad()函数的结果是在脚本的这个位置插入该函数的输出。

(6) 在引入页脚文件（footer.html）之前再次调用这个函数。

```
create_ad();
```

(7) 保存文件，在浏览器中测试它（参见图3-12）。

✓ 提示

- 如果你曾经见过call to undefined function function_name（调用了未定义的函数function_name）错误，这意味着你正在调用一个未定义的函数。如果你拼错了函数的名称（在定义或调用它时）或者你无法包含定义函数的文件，就可能发生这种错误。
- 由于用户定义的函数会占用一些内存，所以在使用这样一个函数时应该小心谨慎。一般的规则是，最好把这些函数用于可能在脚本或Web站点的多个位置执行的大段代码。

3.4.1 创建带参数的函数

就像PHP的内置函数一样，你可以编写带参数(argument，也称为parameter)的函数。例如，strlen()接受一个要确定其字符长度的字符串作为参数。

函数可以带有任意数量的参数，但是它们的排列顺序很重要。将变量添加到函数定义中以允许使用参数：

```
function print_hello ($first, $last) {
    // Function code.
}
```

用于参数的变量名称与脚本的余下部分无关（在本章末尾的框注“变量作用域”中介绍了关于这方面的更多信息），但是要尽量使用有效的、有意义的名称。

一旦定义了函数，以后就可以像调用PHP中的任何其他函数那样调用这个函数，并把字面值或变量发送给它：

```
print_hello ('Jimmy', 'Stewart');
$surname = 'Stewart';
print_hello ('Jimmy', $surname);
```

与PHP中的任何函数一样，如果发送的参数数目有误，就会导致一个错误（参见图3-13）。

Ave. Price Per Gallon:
Warning: Missing argument 1 for create_gallon_radio(), called in /Users/larryullman/Sites/phpmysql4/calculator.php on line 55 and defined in /Users/larryullman/Sites/phpmysql4/calculator.php on line 6

图3-13 给函数发送错误数目（或类型）的参数将会引发错误

为了演示这个概念，我将重写计算表单，使用自定义函数创建每加仑价格的单选按钮。这样可以使代码更简洁。

定义带参数的函数

- (1) 在文本编辑器或IDE中打开calculator.php（参见脚本3-6）。
- (2) 在PHP起始标签后，定义create_gallon_radio() 函数（参见脚本3-8）。

```
function create_gallon_radio ($value) {
```

该函数会创建以下代码：

```
<input type="radio" name= "gallon_price" value="XXX" checked="checked" /> XXX
```

或：

```
<input type="radio" name= "gallon_price" value="XXX" /> XXX
```

脚本 3-8 calculator.php 表单现在使用函数创建单选按钮。与 create_ad()自定义函数不同的是，create_gallon_radio()自定义函数接受一个参数

```
1  <?php # Script 3.8 - calculator.php #3
2
3  // This function creates a radio button.
4  // The function takes one argument: the value.
5  // The function also makes the button "sticky".
6  function create_gallon_radio($value) {
7
8      // Start the element:
9      echo '<input type="radio" name="gallon_price" value="' . $value . '"';
```

```
10 // Check for stickiness:
11 if (isset($_POST['gallon_price']) && ($_POST['gallon_price'] == $value)) {
12     echo ' checked="checked"';
13 }
14
15 // Complete the element:
16 echo " /> $value ";
17
18 } // End of create_gallon_radio() function.
19
20
21 $page_title = 'Trip Cost Calculator';
22 include ('includes/header.html');
23
24 // Check for form submission:
25 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
26
27     // Minimal form validation:
28     if (isset($_POST['distance'], $_POST['gallon_price'], $_POST['efficiency']) &&
29         is_numeric($_POST['distance']) && is_numeric($_POST['gallon_price']) && is_numeric($_POST
30         ['efficiency'])) {
31
32         // Calculate the results:
33         $gallons = $_POST['distance'] / $_POST['efficiency'];
34         $dollars = $gallons * $_POST['gallon_price'];
35         $hours = $_POST['distance']/65;
36
37         // Print the results:
38         echo '<h1>Total Estimated Cost</h1>
39         <p>The total cost of driving ' . $_POST['distance'] . ' miles, averaging ' . $_POST
40         ['efficiency'] . ' miles per gallon, and paying an average of $' . $_POST['gallon_
41         price'] . ' per gallon, is $' . number_format ($dollars, 2) . '. If you drive at an average
42         of 65 miles per hour, the trip will take approximately ' . number_format($hours, 2) . '
43         hours.</p>';
44
45     } // Invalid submitted values.
46     echo '<h1>Error!</h1>
47     <p class="error">Please enter a valid distance, price per gallon, and fuel efficiency.</p>';
48 }
49
50 } // End of main submission IF.
51
52 // Leave the PHP section and create the HTML form:
53 ?>
54
55 <h1>Trip Cost Calculator</h1>
56 <form action="calculator.php" method="post">
57     <p>Distance (in miles): <input type="text" name="distance" value=<?php if (isset($_POST
58         ['distance'])) echo $_POST['distance']; ?>" /></p>
59     <p>Ave. Price Per Gallon: <span class="input">
60         <?php
61         create_gallon_radio('3.00');
62         create_gallon_radio('3.50');
63         create_gallon_radio('4.00');
```

```

58 ?>
59   </span></p>
60   <p>Fuel Efficiency:
61   <select name="efficiency">
62     <option value="10"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] ==
63       '10')) echo ' selected="selected"'; ?>>Terrible</option>
64     <option value="20"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] ==
65       '20')) echo ' selected="selected"'; ?>>Decent</option>
66     <option value="30"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] ==
67       '30')) echo ' selected="selected"'; ?>>Very Good</option>
68     <option value="50"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] ==
69       '50')) echo ' selected="selected"'; ?>>Outstanding</option>
70   </select></p>
71   <p><input type="submit" name="submit" value="Calculate!" /></p>
72 </form>
73
74 <?php include ('includes/footer.html'); ?>

```

3

为了能动态地设置单选按钮的值，需要在每次调用函数时将该值作为参数传入。

注意，用作参数的变量不是`$_POST['gallon_price']`。函数的参数变量是这个函数所特有的，并且具有它们自己的名称。

(3) 创建单选按钮元素。

```
echo '<input type="radio" name="gallon_price" value="' . $value . '"';
```

此处用HTML代码创建单选按钮，并包含了按钮元素的`value`属性。然而，这里并没有闭合元素标签，目的是在后面添加“黏性”代码。表单元素`value`属性的值由传入函数的参数设置。

(4) 如果需要，可以创建具有黏性的单选按钮。

```
if (isset($_POST['gallon_price']) && ($_POST['gallon_price'] == $value)) {
    echo ' checked="checked"';
}
```

这段代码和最初的表单代码很像，唯一的区别是条件语句中的对比值改为了传入函数的参数。

(5) 结束表单元素和函数。

```
echo " /> $value ";
} // End of create_gallon_radio() function.
```

最后，闭合`input`标签，并在后面显示该值，值的两边各有一个空格。

(6) 将表单中的单选按钮由原来的硬编码变成三个函数调用。

```
<?php
create_gallon_radio('3.00');
create_gallon_radio('3.50');
create_gallon_radio('4.00');
?>
```

要创建三个单选按钮，只需要以不同的参数调用三次`create_gallon_radio()`函数即可。这里的数值用单引号括了起来。如果不这样做，PHP会将数值尾部的0丢弃。

(7) 将文件另存为`calculator.php`，存放在Web目录中，并在浏览器中测试它（参见图3-14）。

Total Estimated Cost

The total cost of driving 685 miles, averaging 20 miles per gallon, and paying an average of \$3.00 per gallon, is \$102.75. If you drive at an average of 65 miles per hour, the trip will take approximately 10.54 hours.

Trip Cost Calculator

Distance (in miles):

Ave. Price Per Gallon: 3.00 3.50 4.00

Fuel Efficiency: Decent

图3-14 对用户来说，使用自定义函数创建的单选按钮（参见脚本3-8）与原来没有什么不一样

3.4.2 设置默认的参数值

定义自己的函数的另一个变体是预先设置参数的值。要这样做，可以在函数定义中给参数赋值：

```
function greet ($name, $msg = 'Hello') {
    echo "$msg, $name!";
}
```

设置默认参数值的最终结果是，当调用函数时，特定参数变为可选的。如果把一个值传递给它，就会使用传递的值；否则，就会使用默认值。

你可以根据需要为多个参数设置默认值，只要这些参数出现在函数定义的最后面即可。换句话说，必需的参数应该总是出现在最前面。

利用刚才定义的示例函数，下面的任何一个函数都将会工作：

```
greet ($surname, $message);
greet ('Zoe');
greet ('Sam', 'Good evening');
```

不过，只有greet()不会工作，并且如果不给\$name传递一个值，就无法给\$msg传递一个值（必须按顺序传递参数值，不能跳过必需的参数）。

为了利用默认参数值，下面改进create_gallon_radio()函数。此函数只创建一个名为gallon_price的单选按钮。最好能在表单中多次使用该函数，以创建单选按钮组。（但一般情况下不会像此脚本中这样使用。）

设置默认的参数值

- (1) 在文本编辑器或IDE中打开calculator.php（参见脚本3-8）。
- (2) 更改函数定义那一行的代码（第6行），给其添加第2个可选参数（参见脚本3-9）。

```
function create_radio($value, $name = 'gallon_price') {
```

这里改动了两个地方。首先是将函数名改成一个更通用的名字。其次是添加了第2个参数\$name，该参数包含默认值，在调用时这个参数是可选的。

脚本 3-9 重新定义的函数现在假定了一个单选按钮的名字，除非在调用该函数时指定了一个名字

```

1  <?php # Script 3.9 - calculator.php #4
2
3 // This function creates a radio button.
4 // The function takes two arguments: the value and the name.
5 // The function also makes the button "sticky".
6 function create_radio($value, $name = 'gallon_price') {
7
8    // Start the element:
9    echo '<input type="radio" name="' . $name . '" value="' . $value . '"';
10
11   // Check for stickiness:
12   if (isset($_POST[$name]) && ($_POST[$name] == $value)) {
13       echo ' checked="checked"';
14   }
15
16   // Complete the element:
17   echo " /> $value ";
18
19 } // End of create_radio() function.
20
21 $page_title = 'Trip Cost Calculator';
22 include ('includes/header.html');
23
24 // Check for form submission:
25 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
26
27     // Minimal form validation:
28     if (isset($_POST['distance'], $_POST['gallon_price'], $_POST['efficiency']) &&
29         is_numeric($_POST['distance']) && is_numeric($_POST['gallon_price']) && is_numeric
29         ($_POST['efficiency'])) {
30
31         // Calculate the results:
32         $gallons = $_POST['distance'] / $_POST['efficiency'];
33         $dollars = $gallons * $_POST['gallon_price'];
34         $hours = $_POST['distance']/65;
35
36         // Print the results:
37         echo '<h1>Total Estimated Cost</h1>
38         <p>The total cost of driving ' . $_POST['distance'] . ' miles, averaging ' . $_POST['efficiency'] .
38         ' miles per gallon, and paying an average of $' . $_POST['gallon_price'] . ' per gallon, is $' .
38         number_format ($dollars, 2) . '. If you drive at an average of 65 miles per hour, the trip
38         will take approximately ' . number_format($hours, 2) . ' hours.</p>';
39
40     } else { // Invalid submitted values.
41         echo '<h1>Error!</h1>
42         <p class="error">Please enter a valid distance, price per gallon, and fuel efficiency.</p>';
43     }
44 }
```

```

45 } // End of main submission IF.
46
47 // Leave the PHP section and create the HTML form:
48 ?>
49
50 <h1>Trip Cost Calculator</h1>
51 <form action="calculator.php" method="post">
52   <p>Distance (in miles): <input type="text" name="distance" value=<?php if (isset($_POST
      ['distance'])) echo $_POST['distance']; ?>" /></p>
53   <p>Ave. Price Per Gallon: <span class="input">
54     <?php
55     create_radio('3.00');
56     create_radio('3.50');
57     create_radio('4.00');
58   ?>
59   </span></p>
60   <p>Fuel Efficiency: <select name="efficiency">
61     <option value="10"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '10')) echo ' selected="selected"'; ?>>Terrible</option>
62     <option value="20"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '20')) echo ' selected="selected"'; ?>>Decent</option>
63     <option value="30"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '30')) echo ' selected="selected"'; ?>>Very Good</option>
64     <option value="50"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '50')) echo ' selected="selected"'; ?>>Outstanding</option>
65   </select></p>
66   <p><input type="submit" name="submit" value="Calculate!" /></p>
67 </form>
68
69 <?php include ('includes/footer.html'); ?>

```

(3) 修改函数定义，把原先使用的`gallon_price`替换成参数`$name`。

```

echo '<input type="radio" name="' . $name . '" value="' . $value . '"';
if (isset($_POST[$name]) && ($_POST[$name] == $value)) {
    echo ' checked="checked"';
}

```

这里改动了三个地方。首先是用`$name`设置元素的`name`属性，其次是在检查表单“黏性”的条件语句中将那个两个`$_POST['gallon_price']`修改为`$name`。

(4) 修改函数调用代码。

```

create_radio('3.00');
create_radio('3.50');
create_radio('4.00');

```

在调用函数时需要使用新的函数名。由于第2个参数设有默认值，因此在调用时可以省略。上述调用语句和下面这条调用语句的结果是一样的：

```
create_radio('4.00', 'gallon_price');
```

但是现在这个函数更通用，也可以创建其他单选按钮了。

(5) 保存文件，在浏览器中测试它（参见图3-15）。

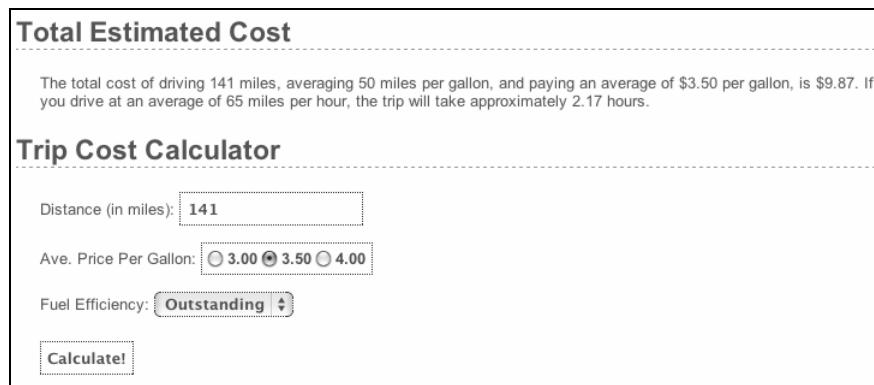


图3-15 新添加的可选参数并没有影响函数的功能

✓ 提示

- 为了不给函数的参数传递任何值，可以使用空字符串 ('')、NULL或FALSE。
- 在PHP手册中，方括号（[]）用于指示函数的可选参数（参见图3-16）。

The screenshot shows the PHP manual entry for the `number_format()` function. The title is **number_format**, followed by the text "(PHP 4, PHP 5)". A brief description states: "number_format — Format a number with grouped thousands". Below this is a "Description" section, which includes the function signature: `string number_format (float $number [, int $decimals = 0])`.

图3-16 PHP手册中number_format()函数的说明，只有第一个参数是必须的，其他都是可选的

3.4.3 从函数返回值

应该讨论的用户定义的函数的最后一个属性是返回值。一些（但不是全部）函数会返回值。例如，`print`将会返回1或0来指示它是否成功执行，而`echo`则不会返回值。另一个例子是，`number_format()`函数会返回一个与字符串中的字符个数相关的数字（参见图3-16）。

要让函数返回一个值，可以使用`return`语句。以下函数会返回给定月份和日期的所属星座：

```
function find_sign ($month, $day) {
    // Function code.
    return $sign;
}
```

这个函数可以返回一个值（比如一个字符串或一个数字），或者一个变量（这个函数已经产生了一个值）。当调用这个返回值的函数时，可以将函数结果赋予一个变量。

```
$my_sign = find_sign ('October', 23);
```

或者把它用作调用另一个函数时的参数。

```
echo find_sign ('October', 23);
```

让我们更新calculator.php脚本，使用函数计算旅程的花费。

让函数返回值

(1) 在文本编辑器或IDE中打开calculator.php (参见脚本3-9)。

(2) 定义完第一个函数后，开始定义第二个函数 (参见脚本3-10)。

```
function calculate_trip_cost ($miles, $mpg, $ppg) {
```

calculate_trip_cost()函数有三个参数：行驶距离\$miles、每英里油耗（加仑）\$mpg，每加仑定价\$ppg。

脚本 3-10 在脚本中加入另一个用户自定义函数，执行主要计算并返回结果

```

1  <?php # Script 3.10 - calculator.php #5
2
3  // This function creates a radio button.
4  // The function takes two arguments: the value and the name.
5  // The function also makes the button "sticky".
6  function create_radio($value, $name = 'gallon_price') {
7
8      // Start the element:
9      echo '<input type="radio" name="' . $name . '" value="' . $value . '"';
10
11     // Check for stickiness:
12     if (isset($_POST[$name]) && ($_POST[$name] == $value)) {
13         echo ' checked="checked"';
14     }
15
16     // Complete the element:
17     echo " /> $value ";
18
19 } // End of create_radio() function.
20
21 // This function calculates the cost of the trip.
22 // The function takes three arguments: the distance, the fuel efficiency, and the price per gallon.
23 // The function returns the total cost.
24 function calculate_trip_cost($miles, $mpg, $ppg) {
25
26     // Get the number of gallons:
27     $gallons = $miles/$mpg;
28
29     // Get the cost of those gallons:
30     $dollars = $gallons/$ppg;
31
32     // Return the formatted cost:
33     return number_format($dollars, 2);
34
35 } // End of calculate_trip_cost() function.
36
37 $page_title = 'Trip Cost Calculator';
38 include ('includes/header.html');
```

```

39 // Check for form submission:
40 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
41
42     // Minimal form validation:
43     if (isset($_POST['distance'], $_POST['gallon_price'], $_POST['efficiency']) &&
44         is_numeric($_POST['distance']) && is_numeric($_POST['gallon_price']) && is_numeric
45             ($_POST['efficiency'])) {
46
47         // Calculate the results:
48         $cost = calculate_trip_cost($_POST['distance'], $_POST['efficiency'], $_POST['gallon_price']);
49         $hours = $_POST['distance']/65;
50
51         // Print the results:
52         echo '<h1>Total Estimated Cost</h1>
53         <p>The total cost of driving ' . $_POST['distance'] . ' miles, averaging ' . $_POST['efficiency'] .
54             ' miles per gallon, and paying an average of $' . $_POST['gallon_price'] . ' per gallon,
55             is $' . $cost . '. If you drive at an average of 65 miles per hour, the trip will take
56             approx imately ' . number_format($hours, 2) . ' hours.</p>';
57
58     } else { // Invalid submitted values.
59         echo '<h1>Error!</h1>
60         <p class="error">Please enter a valid distance, price per gallon, and fuel efficiency.</p>';
61     }
62
63 } // End of main submission IF.
64
65 // Leave the PHP section and create the HTML form:
66 ?>
67
68 <h1>Trip Cost Calculator</h1>
69 <form action="calculator.php" method="post">
70     <p>Distance (in miles): <input type="text" name="distance" value="<?php if (isset($_POST
71         ['distance'])) echo $_POST['distance']; ?>" /></p>
72     <p>Ave. Price Per Gallon: <span class="input">
73         <?php
74             create_radio('3.00');
75             create_radio('3.50');
76             create_radio('4.00');
77         ?>
78         </span></p>
79     <p>Fuel Efficiency: <select name="efficiency">
80         <option value="10"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '10')) ?
81             echo ' selected="selected"'; ?>>Terrible</option>
82         <option value="20"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '20')) ?
83             echo ' selected="selected"'; ?>>Decent</option>
84         <option value="30"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '30')) ?
85             echo ' selected="selected"'; ?>>Very Good</option>
86         <option value="50"><?php if (isset($_POST['efficiency']) && ($_POST['efficiency'] == '50')) ?
87             echo ' selected="selected"'; ?>>Outstanding</option>
88     </select></p>
89     <p><input type="submit" name="submit" value="Calculate!" /></p>
90 </form>
91
92 <?php include ('includes/footer.html'); ?>
```

(3) 执行计算并返回计算结果，结果会格式化为两位小数。

```
$gallons = $miles/$mpg;
$dollars = $gallons/$ppg;
return number_format($dollars, 2);
} // End of calculate_trip_cost() function.
```

前面两行代码和前面脚本中执行计算的代码差不多，不同之处是使用了函数变量。最后这个函数会返回格式化的计算结果。

(4) 修改脚本3-9中计算最终结果的那两行代码(32~33行)，这次调用calculate_trip_cost()函数。

```
$cost = calculate_trip_cost($_POST['distance'], $_POST['efficiency'], $_POST['gallon_price']);
```

当传递给它3个所需的值时，调用函数将执行计算。因为函数返回一个值，所以调用函数的结果(即函数的返回值)将被赋给一个变量\$cost。

(5) 使用新的变量改变echo语句。

```
echo '<h1>Total Estimated Cost</h1>
<p>The total cost of driving ' . $_POST['distance'] . ' miles, averaging ' . $_POST['efficiency'] .
' miles per gallon, and paying an average of ' . $_POST['gallon_price'] . ' per gallon, is ' .
$cost . '. If you drive at an average of 65 miles per hour, the trip will take approximately ' .
number_format($hours, 2) . ' hours.</p>';
```

在这里echo语句使用了\$cost变量而不是\$dollars(参见前面几个脚本)。同样，因为函数格式化了\$cost变量，所以不需要在echo语句中再使用number_format()函数格式化这个变量了。

(6) 保存文件，将它放置到Web目录，并在浏览器中测试它(参见图3-17)。

图3-17 计算表单这次使用自定义函数执行计算并返回行驶费用，但是这种改变对用户所看到的内容没有任何影响

✓ 提示

- return语句会终止代码执行，因此，在执行return语句之后，永远也不会运行函数内的任何代码。
- 一个函数可以具有多条return语句(例如，在switch语句或条件语句中)，但是，至多只会调用其中的一条return语句。例如，函数通常会做以下事情：

```
function some_function () {if /* condition */ {return TRUE;} else {return FALSE;}}
```

- 要让一个函数返回多个值，可以使用array()函数返回一个数组。通过在脚本3-10中把返回行更改为：

```
return array ($var1, $var2);
```

函数可以返回计算的总额和使用的税率（它可以是默认值或者用户提供的值）。

- 在调用返回一个数组的函数时，可使用list()函数将数组元素赋予各个变量：

```
list($v1, $v2) = some_function();
```

3

变量作用域

PHP中的每个变量都有一个针对它的作用域，它是指可以在其中访问变量（从而访问它的值）的一个领域。对于初学者来说，变量的作用域是它们所驻留的页面。因此，如果你定义了\$var，页面的余下部分就可以访问\$var，但是，其他页面一般不能访问它（除非使用特殊的变量）。

因为包含文件像它们是原始（包含）脚本的一部分那样工作，所以在include()那一行之前定义的变量可供包含文件使用（就像你已经通过\$page_title和header.html所看到的那样）。此外，包含文件内定义的变量可供include()那一行之后的父（包含）脚本使用。

当使用你自己定义的函数时，所有这些都将变得不那么明显。这些函数具有它们自己的作用域，这意味着在一个函数内使用的变量不能在其外部使用，在一个函数外部定义的变量不能在其内部使用。由于这个原因，函数内部的变量可以具有与其外部的变量相同的名称，但是它们仍然是完全不同的变量，并且具有不同的值。对于大多数初级程序员来说，这是一个使人糊涂的概念。

要改变一个函数内的变量的作用域，可以使用global语句。

```
function function_name() {
    global $var;
}
$var = 20;
function_name(); // Function call.
```

在这个示例中，函数内部的\$var现在与函数外部的\$var相同。这意味着变量\$var已经具有一个值20，如果在函数内部改变了这个值，外部的\$var值也会改变。

避开变量作用域的另一个方法是利用超全局变量：\$_GET、\$_POST、\$_REQUEST等。这些变量在你的函数内是自动可访问的（因此，它们是超全局变量）。也可以添加元素到\$GLOBALS数组中，使得可以在函数内使用它们。

也就是说，最好不要在函数内使用全局变量。在设计函数时，应该使它们根据需要接受每个值作为参数，并根据需要返回任何值。依靠函数内的全局变量将使得它们更依赖于上下文，因而不太有用。

3.5 回顾和实践

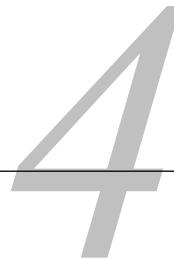
无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书的论坛上，我们会为你答疑解惑。

3.5.1 回顾

- 什么是绝对路径？什么是相对路径？
- `include()`函数和`require()`函数的区别是什么？
- `include()`和`include_once()`的区别是什么？哪个函数通常是你需要避免使用的？为什么？
- 引入文件使用哪种扩展名为什么无关紧要？
- `$_SERVER['REQUEST_METHOD']`值的意义是什么？
- 如何使以下表单元素具有黏性？
 - 文本字段
 - 选择菜单
 - 单选框
 - 复选框
 - 文本框
- 到哪去找由HTML标签内部的代码引发的PHP错误信息？
- 定义一个自定义函数的语法是什么？
- 定义带参数的函数的语法是什么？
- 如何定义包含默认值参数的函数？默认值是如何影响函数调用方式的？如何调用带默认值参数的函数？
- 如何定义和调用带有返回值的函数？

3.5.2 实践

- 为本章的计算表单创建一个新的HTML模板。使用新的模板作为新的头文件和页脚文件。这样你就可以在不修改PHP脚本的情况下修改整站的样式。
- 创建一个新的表单并让它具有“黏性”的能力。表单中必须有一个文本框和一个复选框（这两种都没有在本章中演示）。
- 修改`calculator.php`脚本，使用常量来代替硬编码的平均速度65。（平均速度65是一个“魔数”，脚本中所用的没有任何解释的值。）
- 优化`calculator.php`脚本，让用户可以自己输入或从选项列表中选择平均速度。
- 更新`calculator.php`的输出，让它在旅行时间超过24小时能够显示旅行的天数和小时数。
- 挑战，重写`calculator.php`脚本，使`create_radis()`函数可以一次创建三个单选按钮。提示：使用循环结构。

**本章内容**

- 命名数据库元素
- 选择列类型
- 选择其他的列属性
- 访问MySQL
- 回顾和实践

由于本书讨论的是如何集成多种不同的技术（主要是PHP、SQL和MySQL），在你开始编写使用SQL并与MySQL交互的PHP脚本之前，扎实地逐一理解每种技术是必要的。本章与前面各章有些脱节，它暂时把PHP放在一边，转而深入研究MySQL。

MySQL是世界上最流行的开源数据库应用程序（依据MySQL的Web站点www.mysql.com的说法），并且常与PHP一起使用。MySQL软件带有数据库服务器（它存储实际的数据）、不同的客户应用程序（用于和数据库服务器交互）以及多种实用程序。在本章中，你将看到如何使用MySQL允许的数据类型和其他属性定义简单的表。然后，你将学习如何使用两个不同的客户应用程序与MySQL服务器交互。所有这些信息都是第5章中讲授SQL的基础。

4.1 命名数据库元素

在开始处理数据库之前，必须确定需求。应用程序（在这里是Web站点）的目的规定了应该如何设计数据库。在记住这一点之后，本章和下一章中的示例将使用一个数据库，它存储了一些用户注册信息。

在创建数据库和表时，应该提供清楚、有意义和易于输入的名称[正式地称为标识符(identifier)]。此外，标识符：

- 应该只含字母、数字和下划线（不含空格）；
- 不应该与现有的关键字相同（就像SQL名词或函数名称一样）；
- 应该区分大小写；
- 不能长于64个字符（大约如此）；
- 在其域内必须是唯一的。

最后一条规则意味着表不能有两个列的名称相同，数据库不能有两个表的名称相同。不过，可以在同一个数据库的两个不同的表中使用相同的列名（事实上，你通常会这样做）。至于前三条规则，

我使用的是词语“应该”，因为这些规则是良好的策略，而不是规则的要求。这些规则可以有例外情况，但是这样做的话语法可能很复杂。遵守这些建议是合理的限制，并且有助于避免复杂性。

命名数据库的元素

(1) 确定数据库的名称。

这一步最容易，并且最不重要（这有点争议）。只需确保数据库名称对于MySQL服务器是唯一的即可。如果使用托管服务器，Web主机很可能会提供数据库名称，它可能会或者可能不会包括你的账户或域名。

对于第一个示例，将把数据库命名为sitename，因为其中介绍的信息和技术可以适用于任何一般的站点。

(2) 确定表名称。

在此数据库内只需使表名称唯一即可，这不应该是一个问题。对于本示例，它用于存储用户注册信息，将把唯一的表命名为users。

(3) 确定每个表的列名称。

users表将具有一些列，分别用于存储用户ID、名字、姓氏、电子邮件地址、密码和注册日期。表4-1显示了这些列，并且具有示例数据，它们使用适当的标识符。因为MySQL具有一个名为password的函数，所以我把相应列的名称更改为pass。不必严格如此，但这样做确实很不错。

表4-1 users表将具有6列，用于存储各种记录，如这里的示例数据

列 名	示 例
user_id	834
first_name	Larry
last_name	David
email	ld@example.com
pass	emily07
registration_date	2011-03-31 19:21:03

✓ 提示

- 第6章中使用一个更复杂的示例更详细地讨论数据库设计。
- 精确地讲，数据库、表和列名称的长度限制实际上是64字节，而不是以字符为单位。虽然许多语言中的大多数字符分别只需要一个字节，但是有可能在标识符中使用多字节字符。不过64字节仍然是个很大的空间，因此这对于你来讲也许不是一个问题。
- MySQL中的标识符是否区分大小写实际上依赖于许多事情。在Windows和正常的Mac OS X上，数据库和表名称一般不区分大小写。在UNIX和一些Mac OS X装置上，它们将区分大小写。列名总是区分大小写。在我看来，最好的做法是总是使用全部小写字母，并且好像在区分了大小写的情况下工作。

4.2 选择列类型

一旦确定了数据库所需的所有表和列，就应该确定每个列的数据类型。在创建数据库表时，MySQL要求明确指出每个列将包含哪一类信息。主要有3类信息，对于几乎所有的数据库应用程序

都是如此：

- 文本（即字符串）；
- 数字；
- 日期和时间。

这些类型中的每一种都有许多可以使用的变体——其中有一些是MySQL特有的。正确选择列类型不仅指示可以存储什么信息，还会影响数据库的总体性能。表4-2列出了MySQL的大多数可用类型、它们占据多大的空间以及每种类型的简要描述。注意，在不同的MySQL版本中，数据类型的限制会有所不同，而且字符集（详见第6章）也会影响文本类型的大小。

表4-2 MySQL数据类型

类 型	大 小	描 述
CHAR[Length]	Length字节	定长字段，长度为0~255个字符
VARCHAR[Length]	String长度+1字节 或 String长度+2字节	变长字段，长度为0~65 535个字符
TINYTEXT	String长度+1字节	字符串，最大长度为255个字符
TEXT	String长度+2字节	字符串，最大长度为65 535个字符
MEDIUMTEXT	String长度+3字节	字符串，最大长度为16 777 215个字符
LONGTEXT	String长度+4字节	字符串，最大长度为4 294 967 295个字符
TINYINT[Length]	1字节	范围：-128~127，或者0~255（无符号）
SMALLINT[Length]	2字节	范围：-32 768~32 767，或者0~65 535（无符号）
MEDIUMINT[Length]	3字节	范围：-8 388 608~8 388 607，或者0~16 777 215（无符号）
INT[Length]	4字节	范围：-2 147 483 648~2 147 483 647，或者0~4 294 967 295
BIGINT[Length]	8字节	范围：-9 223 372 036 854 775 808~9 223 372 036 854 775 807，或者0~18 446 744 073 709 551 615（无符号）
FLOAT[Length, Decimals]	4字节	具有浮动小数点的较小的数
DOUBLE[Length, Decimals]	8字节	具有浮动小数点的较大的数
DECIMAL[Length, Decimals]	Length+1字节或 Length+2字节	存储为字符串的DOUBLE，允许固定的小数点
DATE	3字节	采用YYYY-MM-DD格式
DATETIME	8字节	采用YYYY-MM-DD HH:MM:SS格式
TIMESTAMP	4字节	采用YYYYMMDDHHMMSS格式；可接受的范围启始于1970年终止于2038年
TIME	3字节	采用HH:MM:SS格式
ENUM	1或2字节	enumeration（枚举）的简写，这意味着每一列都可以具有多个可能的值之一
SET	1、2、3、4或8字节	与ENUM一样，只不过每一列都可以具有多个可能的值

许多类型可以带有可选的Length属性，以限制它们的大小（方括号[]指示要放置在圆括号中的可选参数）。出于性能目的，应该对任何列中可以存储的数据量设置一些限制。但是应该记住，如果尝试把一个长度为5个字符的字符串插入到一个CHAR(2)列中，就会截去最后3个字符（只会存储前两个

字符，余下的字符将永远丢失）。对设置了长度的任何字段（CHAR、VARCHAR、INT等）都是如此。因此，长度应该总是对应于可以存储的可能最大的值（对于数字），或者可能最长的字符串（对于文本）。

多种不同的日期类型具有各种独特的行为，MySQL手册中有关于它们的文档。你主要将不加修改地使用DATE和TIME字段，因此，不必过于关心它们的错综复杂的情况。

还有两种特殊类型（ENUM和SET）允许为字段定义一系列可接受的值。ENUM列只能从数千个可能的值中取一个值，而SET允许从最多64个可能的值中取多个值。它们在MySQL中都是可用的，但是并非每一种数据库应用程序都提供了它们。

选择列类型

(1) 确定某一列应该是文本、数字，还是日期类型（参见表4-3）。

表4-3 具有泛型数据类型的users表

列名	类型
user_id	数字
first_name	文本
last_name	文本
email	文本
pass	文本
registration_date	日期/时间

这一步通常比较容易和明显，但是你希望尽可能具体一点。例如，可以把日期2006-08-02（MySQL格式）存储为字符串——August 2, 2006。但是，如果使用正确的日期格式，将具有更有用的数据库（如你所见，有一些函数可以把2006-08-02转变成August 2, 2006）。

(2) 为每一列选择最合适的子类型（参见表4-4）。

表4-4 具有更具体数据类型的users表

列名	类型
user_id	MEDIUMINT
first_name	VARCHAR
last_name	VARCHAR
email	VARCHAR
pass	CHAR
registration_date	DATETIME

在我的示例中，将把user_id设置为MEDIUMINT，从而允许多达1700万个值（如无符号数或非负数）。registration_date将是DATETIME，存储用户注册那一天的日期和特定的时刻。当在多种日期类型之间做出抉择时，要考虑是只想访问日期或时间，还是两者可能都想访问。如果不確定，稳妥的做法是存储过多的信息。

其他字段主要是VARCHAR，因为它们的长度因记录而异。唯一的例外是密码列，它是定长的CHAR（在下一章中插入记录时，你就会明白为什么）。见框注“CHAR与VARCHAR”，了解关于这两种类型的更多信息。

CHAR与VARCHAR

这两种类型都存储字符串，并且可以设置固定的最大长度。它们之间的一个主要区别是，存储为CHAR的任何内容将总是被存储为具有列长度的字符串（使用空格填充它；在从数据库中检索存储的值时将删除这些空格）。相反，在VARCHAR列中存储的字符串只需要与字符串本身一样长的空间。因此，VARCHAR(10)列中的单词cat需要4字节的空间（字符串长度加1），但是在CHAR(10)列中，同一个单词将需要10字节的空间。因此，一般来讲，与CHAR列相比，VARCHAR列倾向于占用较少的磁盘空间。

不过，在处理定长的列时，数据库通常更快一些，这是人们赞成使用CHAR的一个证据。对于同一个三字母的单词cat，在CHAR(3)中将只使用3字节，而在VARCHAR(10)中则需要4字节。那么如何决定使用哪一种类型呢？

如果字符串字段总是具有设置的长度（例如，状态简写），就使用CHAR；否则，使用VARCHAR。不过，你可能会注意到，在某些情况下，MySQL把某一列定义为一种类型（如CHAR），即使把它创建成另一种类型（VARCHAR）也是如此。这很正常，并且是MySQL提高性能的一种方式。

(3) 设置文本列的最大长度（参见表4-5）。

表4-5 设置了长度属性的users表

列名	类型
user_id	MEDIUMINT
first_name	VARCHAR(20)
last_name	VARCHAR(40)
email	VARCHAR(60)
pass	CHAR(40)
registration_date	DATETIME

应该基于可能最大的输入，把任意字段的大小都限制为尽可能小的值。例如，如果某一列用于存储状态简写，就把它定义为CHAR(2)。其他时间你可能不得不进行猜测：我不认为任何名字的长度都将超过10个字符，但是仅仅为了安全起见，我将允许最多20个字符。

✓提示

- 数字类型的长度属性不会影响列中可以存储的值的范围。定义为TINYINT(1)或TINYINT(20)的列可以存储完全相同的值。与之相反，对于整数，长度指示显示宽度；对于小数，长度指示可以存储的总位数。
- 如果你在使用非整型时需要绝对精度，DECIMAL要比FLOAT和DOUBLE更合适。
- MySQL有一个BOOLEAN类型（其实就是TINYINT(1)），0表示FALSE，1表示TRUE。
- 根据使用的MySQL版本，当发生INSERT或UPDATE时，即使没有为特定的TIMESTAMP类型的字段指定任何值，它也会被自动设置为当前日期和时间。如果一个表具有多个TIMESTAMP列，那么在执行INSERT或UPDATE时，只会更新其中的第一个列。
- 许多数据类型都具有同义名称：INT和INTEGER，DEC和DECIMAL等。

- 当发生INSERT或UPDATE时，会自动把TIMESTAMP字段类型设置成当前日期和时间，即使没有为那个特定的字段指定任何值。如果一张表具有多个TIMESTAMP列，那么在执行INSERT或UPDATE时，只会更新其中的第一个列。
- MySQL还具有文本类型的多种变体。这些类型是：BINARY、VARBINARY、TINYBLOB、MEDIUMBLOB和LONGBLOB。这些类型用于存储文件或加密数据。

4.3 选择其他的列属性

除了决定应该为列使用哪些数据类型和大小外，还应该考虑一些其他的属性。

首先，不管类型如何，每一列都可以定义为NOT NULL。在数据库和编程中，NULL值相当于说字段没有值。理想情况下，在正确设计的数据库中，每张表中每一行的每一列都应该有一个值，但是并非总是如此。为了强制一个字段具有值，可以把NOT NULL描述添加到其列类型中。例如，可以把必需的美元金额描述为：

```
cost DECIMAL(5,2) NOT NULL
```

在创建表时，还可以为任意列指定一个默认值，而不管其类型是什么。如果大多数记录的某一列具有相同的值，那么可以预先设置一个默认值，这样，当插入新行时，就不必指定一个值（除非该列的那一行的值与众不同）。

```
gender ENUM('M', 'F') default 'F'
```

对于gender列，在添加记录时，如果没有为列指定值，则会使用默认值。

如果列没有默认值并且没有为新记录指定一个值，就会赋予该字段一个NULL值。不过，如果没有指定值并且把列定义为NOT NULL，则会发生错误。对于数字类型，默认值是0。对于大多数日期和时间类型，默认值也是“零”（比如0000-00-00）。表中第一个TIMESTAMP列将有一个当前日期和时间的默认值。除ENUM外（第一个可能的枚举值为默认值，比如上例中的M），字符串类型的默认值为（''）。

可以把数据类型标记为UNSIGNED，这把存储的数据限制为正数或0。这还会有效地把可以存储的正数的范围加倍（因为不会保存负数，参见表4-2）。还可以把数字类型标记为ZEROFILL，这意味着将用0填充任何多余的空间（ZEROFILL还自动地设置UNSIGNED）。

最后，在设计数据库时，需要考虑创建索引、添加键以及使用AUTO_INCREMENT属性。第6章更详细地讨论了这些概念，但是与此同时，看框注“索引、键和AUTO_INCREMENT”，了解它们会如何影响users表。

完成列的定义

(1) 确定主键。

主键可随意选择但特别重要。一般来说，它就是一个数，可用来引用特定记录。例如，电话号码没有固定值，但它是联系你的独特方式（家庭电话或移动电话）。

在users表中，user_id将作为主键：它是用于引用一行数据的任意数字。同样，第6章将更详细地深入探讨主键的概念。

(2) 确定哪些列不能有NULL值。

在这个示例中，每个字段都是必需的（不能是NULL）。与之相对，如果要存储地址，那么可能有address_line1和address_line2两列，其中后者是可选的（它可能具有一个NULL值）。一般来讲，具有许多NULL值的表一般设计不佳（第6章详细讨论了这一点）。

(3) 如果任何数字类型永远不会存储负数，那么可将其设置成UNSIGNED。

`user_id`（表示一个数字）是UNSIGNED，它总为正。UNSIGNED数字的其他示例有：电子商务示例中的商品价格、公司的电话分机号或者邮政编码。

(4) 为任何列创建默认值。

此处所有列在逻辑上都没有隐含一个默认值。

(5) 确认最终的列定义（参见表4-6）。

表4-6 users表的最终定义。`user_id`还将被定义为自动递增的主键

列 名	类 型
<code>user_id</code>	MEDIUMINT UNSIGNED NOT NULL
<code>first_name</code>	VARCHAR(20) NOT NULL
<code>last_name</code>	VARCHAR(40) NOT NULL
<code>email</code>	VARCHAR(60) NOT NULL
<code>pass</code>	CHAR(40) NOT NULL
<code>registration_date</code>	DATETIME NOT NULL

在创建表之前，应该复查将要存储的数据的类型和范围，确保让数据库有效地涵盖各个方面的情况。

✓ 提示

- 文本列也可以定义字符集（character set）或排序方式（collation）。这意味着你将使用多种语言（参见第14章）。
- 列的默认值必须是静态值，不能是函数的执行结果（唯一例外是：`TIMESTAMP`列的默认值可以被赋为`CURRENT_TIMESTAMP`）。
- `TEXT`列不能赋默认值。

索引、键和`AUTO_INCREMENT`

与数据库设计密切相关的两个概念是索引和键。数据库中的索引（index）是请求数据库留意特定列或列组合的值的一种方式（这是一种宽松的定义）。其最终效果是，在检索记录时会提高性能，但是在插入或更新记录时，稍微有一点儿阻碍性能。

数据库表中的键（key）是用于设计更复杂数据库的规范化过程的组成部分（参见第6章）。有两种类型的键：主键（primary key）和外键（foreign key）。每张表都应该有一个主键，一张表中的主键通常被链接为另一张表中的外键。

表的主键是一种引用记录的人工方式，应该遵守下列三条规则。

- (1) 它必须总是具有一个值。
- (2) 那个值永远不会变化。
- (3) 对于表中的每一条记录，那个值必须是唯一的。

在users表中，`user_id`将被指定为PRIMARY KEY（主键），它既是列的描述，又指示MySQL对其进行索引。因为`user_id`是一个数字（主键几乎总是数字），所以我还对该列添加了`AUTO_INCREMENT`描述，它告诉MySQL使用下一个最大的数字作为每一条添加记录的`user_id`值。在开始插入记录时你将看到这实际上意味着什么。

4.4 访问 MySQL

为了创建表、添加记录以及从数据库请求信息，需要某种客户应用程序与MySQL服务器通信。在本书后面，PHP脚本将担当这种角色，但是必须能够使用另一种界面。尽管有许多客户应用程序可用，我将重点介绍其中两种：MySQL客户端（也称为MySQL监视器）和基于Web的phpMyAdmin。如果你对这两种选择不满意，还有第三种选择，即MySQL Query Browser，本书中没有讨论它，但是可以在MySQL Web站点（www.mysql.com）上找到它。

本章假定你能够访问正在运行的MySQL服务器。如果你正在自己的计算机上工作，可参见附录A，了解有关安装、启动MySQL和创建MySQL用户的指导（在学习本章之前，必须完成所有这些工作）。如果你正在使用托管服务器，那么你的Web主机应该给你提供数据库访问权限。取决于使用的托管服务器，它可能提供给你的是phyMyAdmin，而不能使用命令行MySQL客户端。

4.4.1 使用MySQL客户端

MySQL客户端通常与其余的MySQL软件一起安装。尽管MySQL客户端没有漂亮的图形界面，但它是一种可靠、标准的工具，易于使用，并且在许多不同的操作系统上保持一致的工作方式。

可以从命令行界面访问MySQL客户端，在Linux或Mac OS X中，命令行界面是Terminal应用程序（参见图4-1），在Windows中，则是DOS提示符（参见图4-2）。如果你不习惯命令行式的交互，可能发现这个界面有一些挑战，但它立刻会变得易于使用。

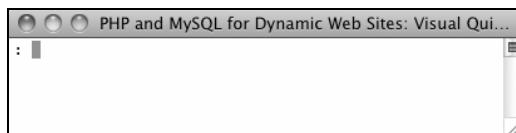


图4-1 Mac OS X中的Terminal窗口

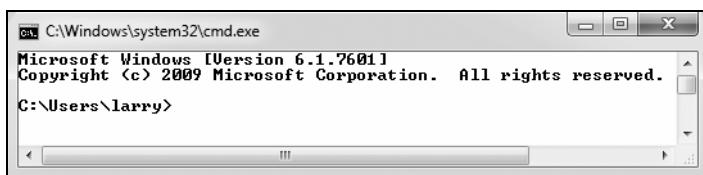


图4-2 Windows DOS提示符或控制台（尽管默认为黑色背景上的白色文本）

要开始从命令行启动应用程序，可以输入它的名称，并按下Return或Enter键：

```
Mysql
```

根据你使用的服务器（或你的电脑），有可能需要输入完整路径，以便启动应用程序，例如：

```
/Applications/MAMP/Library/bin/mysql (Mac OS X, 使用MAMP)
C:\xampp\mysql\bin\mysql (Windows系统, 使用XAMPP)
```

在激活这个应用程序时，可以添加几个参数影响它的运行方式。最常见的参数包括你想用于连接的用户名、密码和主机名（计算机名或URL）。可以按如下方式建立这些参数：

```
mysql -u username -h hostname -p
```

-p选项将导致客户提示你输入密码。如果你愿意，也可以在这一行指定密码（直接在-p提示符后面输入它）但是它将是可见的，这不安全。-h hostname参数是可选的，你可以不使用它，除非你不能以别的方式连接到MySQL服务器。

在MySQL客户端内，需要用分号来终止每一条语句（SQL命令）。这些分号指示客户查询已完成并且应该运行，分号不是SQL自身的一部分（这通常会使人感到糊涂）。这还意味着在MySQL客户端内，可以把同一条SQL语句分布在多行上，以使它更易于阅读。

作为访问和使用MySQL客户端的一个快速演示，下面这几步将展示如何启动MySQL客户端，选择要使用的数据库以及退出客户。在执行这些步骤之前，

- 必须运行MySQL服务器；
- 必须有用户名和密码，它们具有适当的访问权限。

附录A中解释了这两个步骤。

附带提一下，在下面的步骤中以及本书余下的内容中，我将继续提供在Windows和Mac OS X上使用MySQL客户端的图像。虽然它们的外观有所不同，但是步骤和结果完全一样。因此，简而言之，不必关心为什么一幅图像显示的是DOS提示符，而下一幅图像显示的则是Terminal。

使用MySQL客户端

(1) 从命令行界面访问你的系统。

在UNIX系统和Mac OS X上，这只是调出Terminal或类似的应用程序。

如果正在使用Windows，在开始菜单中选择“运行”（或按Windows键+R），在窗口中输入cmd，并按下Enter键，调出DOS提示符（参见图4-3）。

(2) 使用合适的命令激活MySQL客户端（参见图4-4）。

```
/path/to/mysql/bin/mysql -u username -p
```

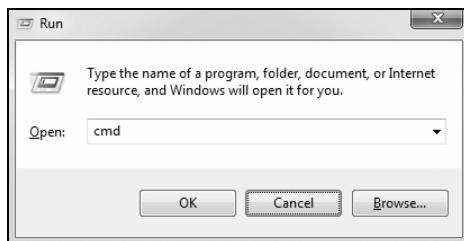


图4-3 在Window系统的“运行”提示框中键入cmd，可进入DOS提示符窗口

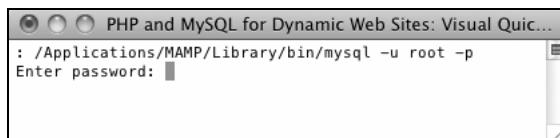


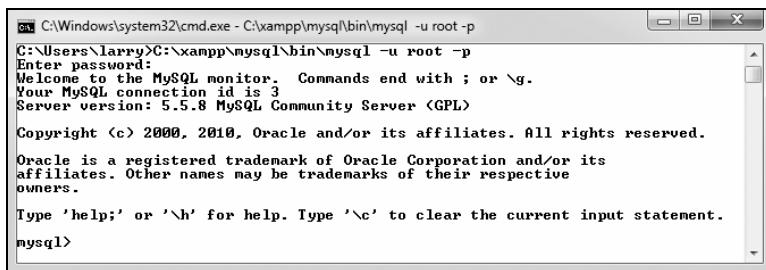
图4-4 输入到达实用程序的完整路径以及正确的参数，访问MySQL客户端

□ 这一步中的/path/to/mysql部分主要是由正在运行的操作系统以及MySQL的安装位置指定的。我之前已经提供了两个位置，分别是在Mac OS X上的MAMP的安装位置和在Windows系统上XAMPP的安装位置。

基本的前提条件是，正在运行MySQL客户端，以username连接，并请求被提示输入密码。我没有过分强调这一点，但是你使用的用户名和密码必须已经在MySQL中建立为有效的用户(参见附录A)。

(3) 在提示符后面输入密码，并按下Return键/Enter键。

如果你使用正确的用户名/密码组合(即某个具有有效访问权限的人)，那么应该看到如图4-5所示的问候。如果访问被拒绝，那么你可能没有使用正确的值(参见附录A，了解关于创建用户的指导)。



```
C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p
C:\Users\Larry>C:\xampp\mysql\bin\mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.5.8 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

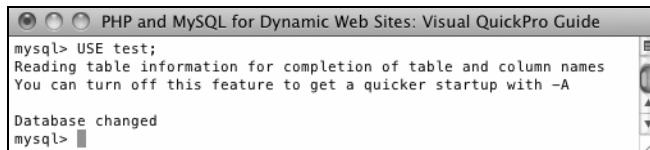
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

图4-5 如果能够成功登录，就会看到图中所示的欢迎消息

(4) 选择你想使用的数据库(参见图4-6)。

```
USE test;
```



```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide

mysql> USE test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

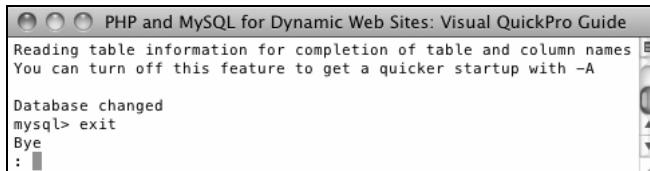
Database changed
mysql>
```

图4-6 在进入MySQL客户端之后，运行USE命令选择要使用的数据库

USE命令选择要用于每个后续命令的数据库。test数据库是MySQL会默认安装的数据库。假定它存在于你的服务器上，所有用户都应该能够访问它。

(5) 退出MySQL(参见图4-7)。

```
exit
```



```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> exit
Bye
:
```

图4-7 输入exit或quit，终止会话并离开MySQL客户端

你也可以使用quit命令离开MySQL客户端。这一步与你在MySQL客户端中输入的大多数其他命令不同，在末尾不需要使用分号。

(6)退出终端式DOS控制台会话。

```
exit
```

exit会关闭当前会话。在Window系统中也会关闭DOS提示符窗口。

✓ 提示

□ 如果你预先知道想使用哪个数据库，那么可以利用以下命令启动MySQL：

```
/path/to/mysql/bin/mysql -u username -p databaseName
```

□ 要查看利用MySQL客户端还可以做别的什么事情，可以输入

```
/path/to/mysql/bin/mysql --help
```

□ 大多数系统上的MySQL客户端允许使用向上和向下的箭头键来滚动以前输入的命令。如果你在输入查询时犯了错误，可以向上滚动找到它，然后校正错误。

□ 在MySQL客户端，你也可以使用\G代替分号来结束SQL命令。对于查询返回的结果，用\G可以将结果以垂直列表的方式显示出来，这要比水平列表更易于浏览。

□ 如果你正在执行一条较长的语句并且犯了一个错误，可以输入c并按下Return键或Enter键来取消当前的操作。如果MySQL认为遗漏了一个右单引号或双引号（如'>和">提示符所提示的那样），那么需要首先输入合适的引号。

4.4.2 使用phpMyAdmin

phpMyAdmin (www.phpmyadmin.net) 是用PHP编写的最好、最流行的应用程序之一。它的唯一用途是提供一个到达MySQL服务器的界面。与MySQL客户端相比，它更容易使用、更自然，但是需要安装PHP，并且必须通过Web浏览器访问它。如果你在自己的计算机上运行MySQL，可能发现使用MySQL客户端更有意义，因为安装和配置phpMyAdmin会引入不必要的多余工作（尽管一体式的PHP和MySQL安装程序可能会为你做这项工作）。如果使用托管服务器，实际上会保证Web主机提供phpMyAdmin作为使用MySQL的主要方式，并且可能不会选择MySQL客户端。

使用phpMyAdmin并不困难，但是接下来的步骤会介绍一些基础知识，因为你将知道在接下来几章中做什么。

使用phpMyAdmin

(1) 通过Web浏览器访问phpMyAdmin（参见图4-8）。

你使用的URL将依赖于你自身的情况。如果运行在自己的计算机上，这可能是`http://localhost/phpMyAdmin/`。如果运行在托管站点上，Web主机将给你提供正确的URL。在各种可能的情况下，都可以通过站点的控制面板（它应该存在）使用phpMyAdmin。

注意，仅当phpMyAdmin被正确配置成使用有效的用户名/密码/主机名的组合连接到MySQL时，它才会工作。如果看到如图4-9所示的消息，那么可能没有使用正确的值（参见附录A，了解关于创建用户的指导）。

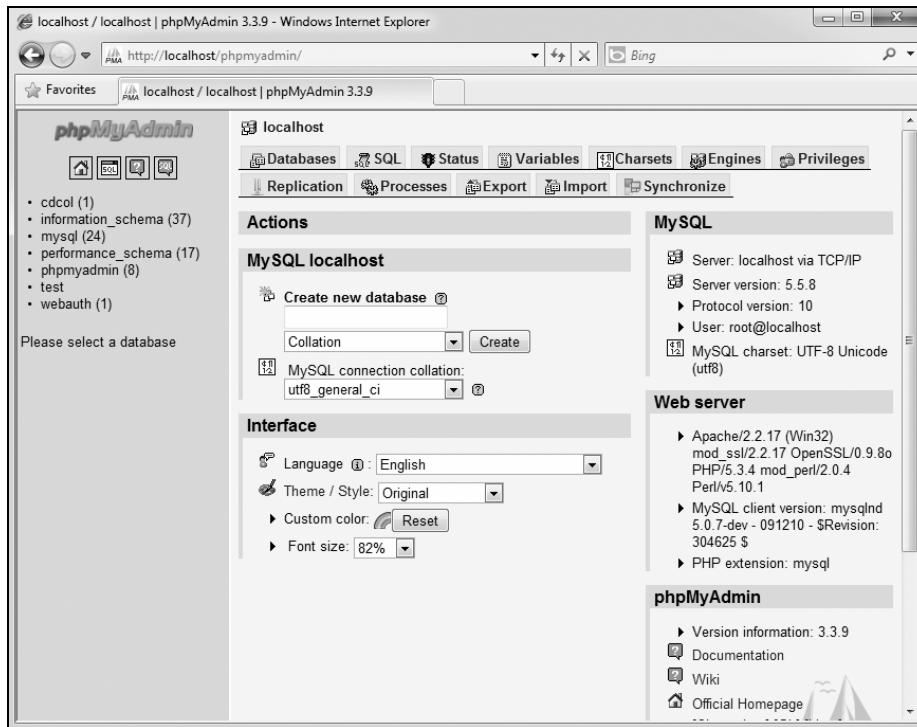


图4-8 第一个phpMyAdmin页面（作为可以访问多个数据库的MySQL用户连接时）

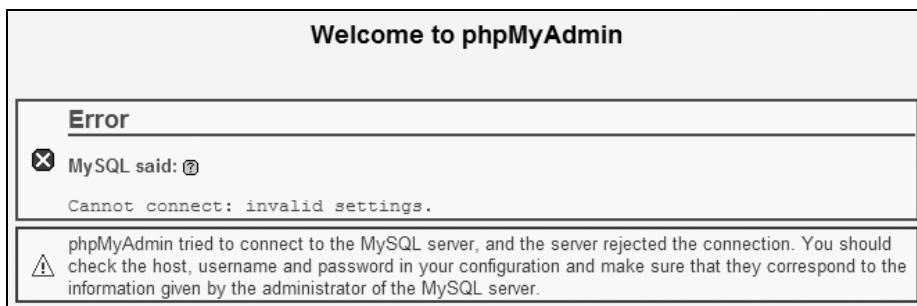


图4-9 每个客户端应用程序都需要正确的用户名/密码/主机名的组合，以便与MySQL服务器交互

(2) 如果可能并且必要，可使用左边的菜单选择要使用的数据库（参见图4-10）。

你在这里所具有的选项因phpMyAdmin正作为什么MySQL用户连接而异。该用户也许能够访问一个数据库、多个数据库或者所有的数据库。在只有一个数据库的托管站点上，可能已经为你选择了那个数据库（参见图4-11）。在你自己的计算机上，作为MySQL根用户连接phpMyAdmin，你将看到一个下拉菜单（参见图4-10）或者可用数据库的简单列表（参见图4-8）。

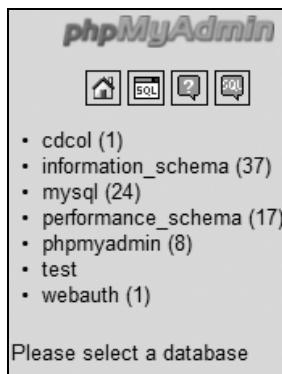


图4-10 使用窗口左边的地址列表，选择想使用哪个数据库。这等价于在MySQL客户端内运行`USE databaseName`查询（参见图4-6）

(3) 在左侧边栏中点击表名选择表，参见图4-11。

你不必非要这样做，实际上如果你仅使用本书的SQL命令，那就不用选择表，但是选择表有时候可能会简化一些任务。

Field	Type	Collation	Attributes	Null	Default
<input type="checkbox"/> id	bigint(20)		UNSIGNED	No	None
<input type="checkbox"/> username	varchar(64)	utf8_bin		No	
<input type="checkbox"/> db	varchar(64)	utf8_bin		No	
<input type="checkbox"/> table	varchar(64)	utf8_bin		No	
<input type="checkbox"/> timevalue	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP
<input type="checkbox"/> sqlquery	text	utf8_bin		No	None

图4-11 在页面的左边栏中选择某个表，会相应地改为右边栏中的选项

(4) 使用标签和链接（在页面的右侧）来完成常规任务。

在大部分情况下，标签和链接是一些普通SQL命令的快捷方式。例如，`Browse`标签执行了`SELECT`查询，而`Insert`标签会创建一个添加新纪录的表单。

(5) 可以使用SQL标签页（参见图4-12）或SQL查询窗口（参见图4-13）输入SQL命令。

在接下来的三章及后面几章，会提供一些SQL命令用于创建、填充和修改表。这些命令可能如下：

```
INSERT INTO tablename (col1, col2) VALUES (x, y)
```

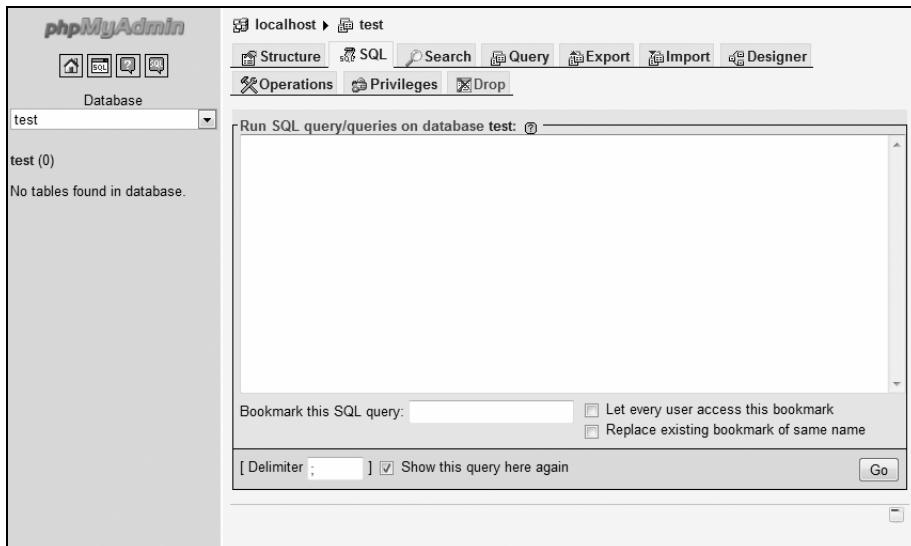


图4-12 窗口主要部分中的SQL选项卡可用于运行任何SQL命令

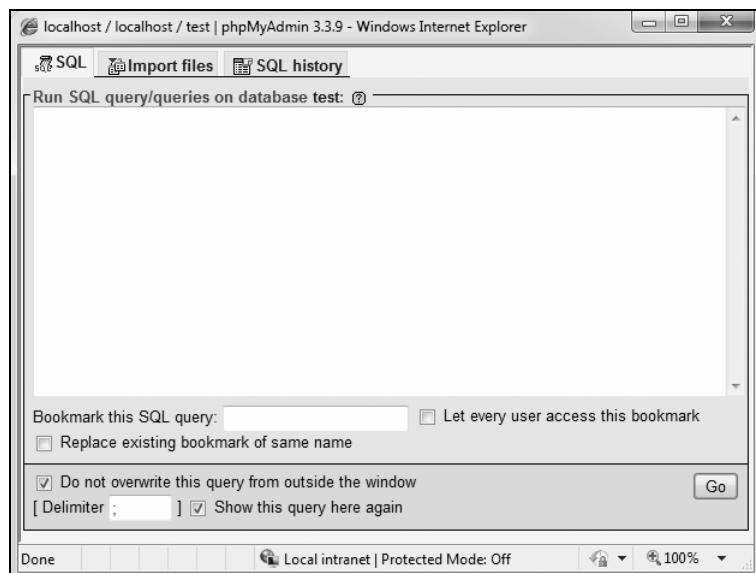


图4-13 SQL窗口也可用于运行命令。在单击浏览器左上角的SQL图标后将弹出它
(参见图4-11中左起第二个图标)

可以使用MySQL客户端、phpMyAdmin或任何其他的界面运行这些命令。为了在phpMyAdmin内运行它们，只需把它们输入SQL提示符之一中并单击Go即可。

✓ 提示

- 还可以利用phpMyAdmin做更多的工作，但是要完全介绍它们将需要一整章的篇幅（并且这一章会比较长）。如果你不想使用MySQL客户端，这里介绍的信息足以让你理解本书中的任何示例。
- 可以将phpMyAdmin配置成使用一个特殊的数据库，它将记录你的查询历史，允许你标记查询等。
- 使用phpMyAdmin的最佳理由之一是把数据库从一台计算机转移到另一台计算机上。使用phpMyAdmin中的Export选项卡连接到源计算机创建数据文件。然后，在目标计算机上，使用phpMyAdmin中的Import选项卡（连接到那个MySQL服务器）完成转移。

4.5 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

4.5.1 回顾

- 你正使用的MySQL是什么版本？如果你还不清楚，现在就去查一下。
- 数据库名，表名和列名能够使用那些字符？
- 你应该把数据库名，表名和列名是否区分大小写？
- 常见的三个列类型分别是？
- CHAR和VARCHAR之间有什么区别？
- 如何确定列的大小（数据类型还是长度）？
- 列还可以使用其他哪些属性？
- 什么是主键？
- 如果使用命令行客户端连接MySQL，要使用什么用户名和密码？

4.5.2 实践

- 找出与你的版本相应的MySQL在线手册并添加到书签！
- 开始考虑一下你的项目可能用到的数据库。
- 如果你还没有修改MySQL的root用户密码（前提是你使用的是安装在自己电脑上的MySQL），请参考附录A中的指南修改密码。

第5章

SQL简介

5

本章内容

- 创建数据库和表
- 插入记录
- 选择数据
- 使用条件语句
- 使用LIKE和NOT LIKE
- 对查询结果排序
- 限制查询结果
- 更新数据
- 删除数据
- 使用函数
- 回顾和实践

上一章简单介绍了MySQL，重点介绍了两个主题：使用MySQL的规则和数据类型定义数据库，以及如何与MySQL服务器交互。本章转而介绍数据库的通用语言：SQL。

SQL (Structured Query Language, 结构化查询语言) 是一组特殊的单词，专门用于同数据库交互。所有的主流数据库都会使用SQL，MySQL也不例外。SQL有多种版本，而且MySQL对SQL标准的实现又有不少变异，但是SQL仍然极其容易学习和使用。事实上，在SQL中最难做的事情就是发掘出它的全部潜力！

在本章中，你将学习创建表、填充它们以及运行其他基本查询所需知道的所有SQL。这些示例全都使用上一章中讨论的users表。此外，与上一章一样，本章假定你能够访问运行的MySQL服务器，并且知道如何使用客户端应用程序与之交互。

5.1 创建数据库和表

首次使用SQL和MySQL很可能是创建一个数据库。创建一个新数据库的语法如下：

```
CREATE DATABASE databasename
```

这就是要编写的全部代码（如我所说，SQL易于学习）！

CREATE名词也用于建立表。

```
CREATE TABLE tablename (
    column1name description,
    column2name description
    ...)
```

如你可以从该语法中所看到的，在命名表之后，将在括号内（依次）定义每一列。每组列一描述对相互之间应该用逗号隔开。如果选择此时创建索引，可以在创建语句的末尾添加，但是你也可以在以后某个时间添加索引（第6章中将更正式地讨论索引，但是第4章中也介绍了这个主题）。

SQL不区分大小写。不过，我强烈建议你养成大写SQL关键字的习惯，就像前面的示例语法和下面的步骤中那样。这样做有助于把SQL名词与数据库、表和列名区分开。

创建数据库和表

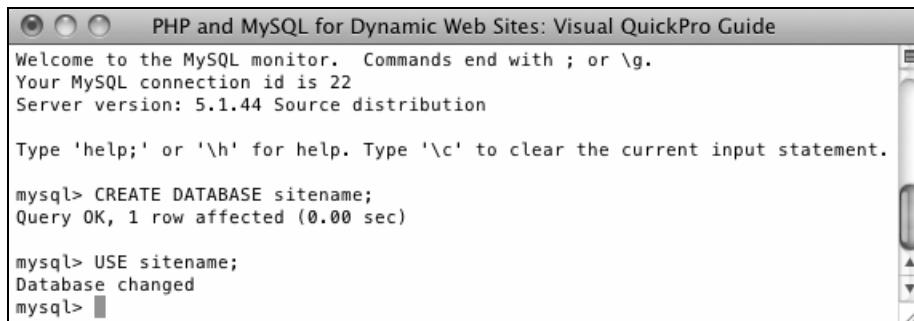
(1) 使用你所喜欢的任何一种客户访问MySQL。

第4章说明了如何使用两种最常见的界面（MySQL客户端和phpMyAdmin）与MySQL服务器通信。使用上一章中介绍的步骤，你现在应该连接到MySQL。

在本章余下的内容中，将使用MySQL客户端输入大多数SQL示例，但是这和它们在phpMyAdmin或任何其他的客户工具中的工作完全相同。

(2) 创建并选择新数据库（参见图5-1）。

```
CREATE DATABASE sitename;
USE sitename;
```



The screenshot shows a window titled "PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide". It displays the MySQL monitor interface. The text in the window is as follows:

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.1.44 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE sitename;
Query OK, 1 row affected (0.00 sec)

mysql> USE sitename;
Database changed
mysql>
```

图5-1 在MySQL中创建一个名为sitename的新数据库，然后选择它用于将来的查询

第一行将创建数据库（假定你作为一位有权创建新数据库的用户连接到MySQL）。第二行告诉MySQL从现在起你想在该数据库内工作。记住，在MySQL客户端内，必须用分号终止每一条SQL命令，尽管严格说来这些分号不是SQL自身的一部分。如果同时在phpMyAdmin内执行多个查询，也应该用分号隔开它们（参见图5-2）。如果只在phpMyAdmin内运行一个查询，就不需要分号。

如果你使用的是托管公司的MySQL，它们可能会为你创建数据库。在这种情况下，只需连接到MySQL并选择数据库即可。

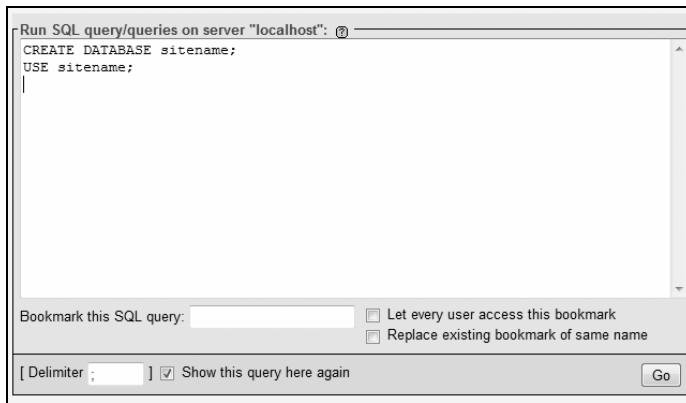


图5-2 可以在phpMyAdmin的SQL窗口内运行用于创建和选择数据库的相同命令

(3) 创建users表 (参见图5-3)。

```
CREATE TABLE users (
    user_id MEDIUMINT UNSIGNED NOT NULL
    AUTO_INCREMENT,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(40) NOT NULL,
    email VARCHAR(60) NOT NULL,
    pass CHAR(40) NOT NULL,
    registration_date DATETIME NOT NULL,
    PRIMARY KEY (user_id)
);
```

第4章中开发了users表的设计，在那里，基于许多准则确定了表中每一列的名称、类型和属性（参见第4章，了解更多信息）。在这里，把这些信息集成进CREATE表语法中，实际地在数据库中创建表。

由于MySQL客户端直至遇到一个分号才会运行一个查询，所以可以像我在图5-3中所做的那样，把语句输入到多行上（在每一行末尾按下Return键或Enter键）。这通常使得查询更容易阅读和调试。在phpMyAdmin中，也可以运行分布在多行上的查询，尽管直到单击了Go之后才会运行它们。

(4) 确认表是否存在 (参见图5-4)。

```
SHOW TABLES;
SHOW COLUMNS FROM users;
```

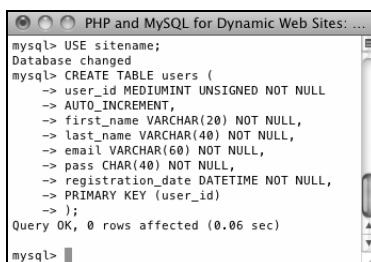
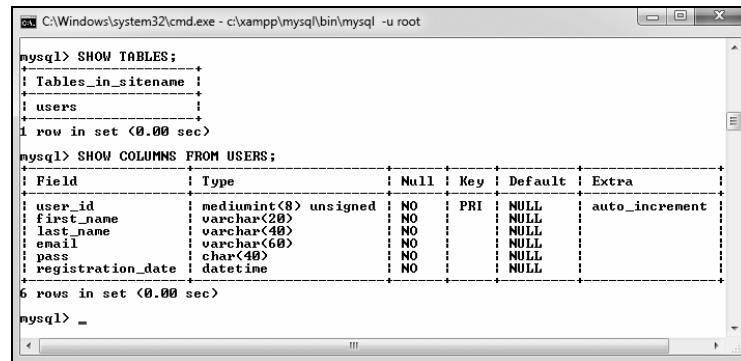


图5-3 这个CREATE SQL命令将建立users表



```

C:\Windows\system32\cmd.exe - c:\xampp\mysql\bin\mysql -u root
mysql> SHOW TABLES;
+-----+
| Tables_in_sitename |
+-----+
| users |
+-----+
1 row in set (0.00 sec)

mysql> SHOW COLUMNS FROM users;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| user_id | mediumint(8) unsigned | NO | PRI | NULL | auto_increment |
| first_name | varchar(20) | NO | | NULL | |
| last_name | varchar(40) | NO | | NULL | |
| email | varchar(60) | NO | | NULL | |
| pass | char(40) | NO | | NULL | |
| registration_date | datetime | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> -

```

5

图5-4 使用SHOW命令可以确认表及其中的列存在与否

SHOW命令可用于呈现数据库中的表，或者表中的列名称和类型。

此外，你可能还注意到图5-4中user_id的默认值是NULL，即使该列被定义为NOT NULL也是如此。这实际上是正确的，与user_id是一个自动递增的主键有关。为了获得更好的性能或者其他原因，MySQL通常会对列的定义做出细微的修改。

在phpMyAdmin中，在浏览器窗口左侧的数据库名称下面列出数据库中的表（参见图5-5）。单击表的名称，查看它的列（参见图5-6）。

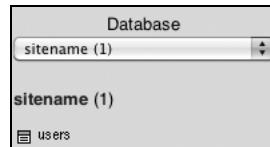


图5-5 phpMyAdmin显示sitename数据库包含一个名为users的表

Field	Type	Collation	Attributes	Null	Default	Extra
user_id	mediumint(8)		UNSIGNED	No	None	auto_increment
first_name	varchar(20)	latin1_swedish_ci		No	None	
last_name	varchar(40)	latin1_swedish_ci		No	None	
email	varchar(60)	latin1_swedish_ci		No	None	
pass	char(40)	latin1_swedish_ci		No	None	
registration_date	datetime			No	None	

图5-6 phpMyAdmin在屏幕上显示表的定义（通过单击左边列中的表名称来访问）

✓ 提示

- 在本章余下的内容中，我将假定你正在使用MySQL客户端或兼容的工具，并且已经用USE选择了sitename数据库。
- 在创建表时列出列的顺序没有任何实质性的影响，但是有一些格式上的建议指示如何对它们进行排序。我通常首先列出主键列，接着列出任何外键列（下一章中讨论了关于这个主题的更多信息），然后列出其余的列，最后列出任何日期列。

- 在创建表时，可以选择指定其类型。MySQL支持许多表类型，每种类型都有它自己的长处和弱点。如果没有指定一种表类型，MySQL将使用默认的类型（在安装MySQL时确定的）自动创建表。第6章更详细地讨论了各种表类型。
- 在创建表和文本列时，可以选择指定其排序方式和字符集。在使用多种语言或者并非MySQL服务器固有的语言时，它们二者都很有用。第16章介绍了这些主题。
- `DESCRIBE tablename`语句与`SHOW COLUMNS FROM tablename`语句等价。

5.2 插入记录

在创建了数据库及其表之后，可以使用`INSERT`命令填充它们。可以用两种方法编写`INSERT`查询。利用第一种方法指定要填充的列：

```
INSERT INTO tablename (column1, column2...) VALUES (value1, value2 ...)
INSERT INTO tablename (column4, column8) VALUES (valueX, valueY)
```

使用这种结构，可以添加多行记录，只填充一部分相关的列。其结果将是，任何未赋值的列都将被视作`NULL`（或者如果定义了默认值，就赋予默认值）。注意，如果列不能具有`NULL`值（它被设置为`NOT NULL`）并且没有默认值，那么不指定值将会引发一个错误（参见图5-7）。

The screenshot shows a MySQL command-line interface window titled "PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide". It displays the following SQL session:

```
mysql> INSERT INTO users
    -> (first_name, email)
    -> VALUES ('Larry', 'Larry@LarryUllman.com');
Query OK, 1 row affected, 3 warnings (0.00 sec)

mysql> SHOW WARNINGS;
+-----+-----+-----+
| Level | Code | Message          |
+-----+-----+-----+
| Warning | 1364 | Field 'last_name' doesn't have a default value |
| Warning | 1364 | Field 'pass' doesn't have a default value      |
| Warning | 1364 | Field 'registration_date' doesn't have a default value |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

The output shows three warning messages (Level 1364) because the 'last_name', 'pass', and 'registration_date' columns are defined as `NOT NULL` and have no default values.

图5-7 如果没有向`NOT NULL`字段提供（或预定义）值，则会导致出现错误或警告信息

用于插入记录的第二种方法则是根本不指定任何列，而是包括每一列的值。

```
INSERT INTO tablename VALUES (value1, NULL, value2, value3, ...)
```

如果使用第二种方法，就必须为每一列指定一个值，即使它为`NULL`也是如此。如果表中有6列，就必须列出6个值。如果值的数量与列的数量不匹配，就会引发一个错误（参见图5-8）。由于这种原因以及其他的原因，插入记录的第一种方法一般更可取。

MySQL还允许同时插入多行，并用逗号隔开每条记录。

```
INSERT INTO tablename (column1, column4) VALUES (valueA, valueB),
(valueC, valueD),
(valueE, valueF)
```

虽然可以利用MySQL做到这一点，但是它不为SQL标准所接受，因此，并非所有的数据库应用程序都支持它。

```
mysql> INSERT INTO users VALUES
-> ('Larry', 'Ullman', 'Larry@LarryUllman.com');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql>
```

图5-8 如果在INSERT查询语句中没有为表中的每列都提供值就会导致错误

注意：在所有这些示例中，为实际的表名、列名和值使用了占位符。此外，这些示例没有使用引号。在真实的查询中，必须遵守某些规则来避免错误（参见框注“查询中的引号”）。

5 查询中的引号

在每个SQL命令中：

- 数值不应该用引号括住；
- 字符串值（对于CHAR、VARCHAR和TEXT列类型）必须总是用引号括住；
- 日期和时间值必须总是用引号括住；
- 函数不能用引号括住；
- 单词NULL一定不能用引号括住。

不必要地用引号括住数值一般不会引发问题（尽管你仍然不应该这样做），但是在其他情况下误用引号几乎总会把事情搞得一团糟。此外，使用单引号或双引号均可，只要始终使它们保持配对即可（左引号总是具有匹配的右引号）。

与PHP一样，如果需要在值中使用引号，可以使用另一种引号括住它，或者在引号前放置一个反斜杠对它进行转义：

```
INSERT INTO tablename (last_name) VALUES ('O\'Toole')
```

插入数据到表中

(1) 把一行新数据插入users表中，并指定要填充的列（参见图5-9）。

```
INSERT INTO users
(first_name, last_name, email, pass, registration_date)
VALUES ('Larry', 'Ullman', 'email@example.com', SHA1('mypass'), NOW());
```

```
C:\Windows\system32\cmd.exe - c:\xampp\mysql\bin\mysql -u root
mysql> INSERT INTO users
-> (first_name, last_name, email, pass, registration_date)
-> VALUES ('Larry', 'Ullman', 'email@example.com', SHA1('mypass'), NOW());
Query OK, 1 row affected <0.01 sec>
mysql>
```

图5-9 该查询把单条记录插入users表中。消息1 row affected指示插入成功

同样，这种语法（其中需要指定特定的列）更简单，但并不总是最方便。对于名字、姓氏和电子邮件这三列，可以为值使用简单的字符串（并且必须总是用引号括住字符串）。对于密码和注册日期这两列，使用两个函数生成值（见框注“两个MySQL函数”）。SHA1()函数用于加密密码（在本示例中

是mypass)。NOW()函数将把registration_date设置为当前时刻。

当在SQL语句中使用任何函数时，不要用引号括住它。另外，在函数名与其后的圆括号之间一定不能有空格（因此，NOW()是正确的，NOW()则不然）。

两个MySQL函数

尽管我将在本章后面更详细地讨论函数，但是我想在这里介绍两个MySQL函数：SHA1()和NOW()。

SHA1()函数是一种加密数据的方式。这个函数会创建一个加密字符串，其长度总是正好为40个字符（这就是我把users表的pass列设置为CHAR(40)的原因）。SHA1()是一种单向加密技术，这意味着它不能被解密。对于那些无需以未加密形式再次查看的敏感数据，则可使用该函数进行存储，但是，对于那些应该加以保护但往后要查看的敏感数据（如信用卡号），它显然不是一个好的选择。从MySQL 5.0.2起，就可以使用SHA1()；如果你使用更早的版本，则可改用MD5()函数。这个函数使用不同的算法执行相同任务，并返回一个长度为32个字符的字符串（如果使用MD5()函数，可以把pass列定义为CHAR(32)）。

NOW()函数可方便地用于日期、时间和时间戳这些列，因为它将为那个字段插入（服务器上的）当前日期和时间。

(2) 在不指定列的情况下插入一行数据到users表中（参见图5-10）。

```
INSERT INTO users VALUES
(NULL, 'Zoe', 'Isabella', 'email2@example.com', SHA1('mojito'), NOW());
```

图5-10 把另一条记录插入到表中，这一次采用的方法是为表中的每一列提供一个值

在第二个语法示例中，必须为每一列提供一个值。我把user_id列设置为NULL值，依据其AUTO_INCREMENT描述，它将导致MySQL使用下一个逻辑编号。换句话说，第一条记录的user_id为1，第二条记录的user_id为2，以此类推。

(3) 插入多个值到users表中（参见图5-11）。

```
INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES
('John', 'Lennon', 'john@beatles.com', SHA1('Happin3ss'), NOW()),
('Paul', 'McCartney', 'paul@beatles.com', SHA1('letITbe'), NOW()),
('George', 'Harrison', 'george@beatles.com', SHA1('something'), NOW()),
('Ringo', 'Starr', 'ringo@beatles.com', SHA1('thisboy'), NOW());
```

因为MySQL允许同时插入多个值，所以可以利用这种特性，用记录填充表。

(4) 继续第(1)步和第(2)步，直到完全填充了users表。

在本章余下的内容中，将基于输入到数据库中的记录来执行查询。你的数据库应该没有与我的数

据库一样的具体记录，可以相应地更改特定的记录。不论数据是什么，以下查询背后的基本思想应该仍然适用，因为sitename数据库有固定的列和表结构。

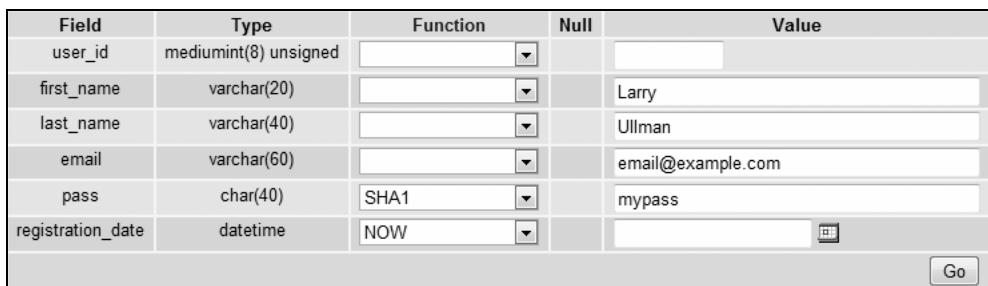


```
mysql> INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES
-> ('John', 'Lennon', 'john@beatles.com', SHA1('Happin3ss'), NOW()),
-> ('Paul', 'McCartney', 'paul@beatles.com', SHA1('letItBe'), NOW()),
-> ('George', 'Harrison', 'george@beatles.com', SHA1('something'), NOW()),
-> ('Ringo', 'Starr', 'ringo@beatles.com', SHA1('thisboy'), NOW());
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0
mysql> _
```

图5-11 这个查询（MySQL允许它，但是其他的数据库不允许）同时把多条记录插入到表中

✓ 提示

- 在本书的支持Web站点的下载页面上，可以下载本书的所有SQL命令。使用其中一些命令，可以按与我完全一样的方式填充你的users表。
- INSERT语句中的INTO名词在MySQL的当前版本中是可选的。
- phpMyAdmin的INSERT选项卡允许使用HTML表单插入记录（参见图5-12）。



Field	Type	Function	Null	Value
user_id	mediumint(8) unsigned			
first_name	varchar(20)			Larry
last_name	varchar(40)			Ullman
email	varchar(60)			email@example.com
pass	char(40)	SHA1		mypass
registration_date	datetime	NOW		

图5-12 phpMyAdmin的INSERT表单显示了表中的列并提供了用于输入值的文本框。下拉菜单列出了可以使用的函数，比如用于密码的SHA1()和用于注册日期的NOW()

- 如果向不可以为NULL的字段中插入数据，那么根据使用的MySQL版本，INSERT表单有的可以在提示警告的情况下正常工作（参见图5-7），有的则会提示错误信息说明插入操作失败。
- 你偶尔会在SQL命令中看到使用反引号（`）。这个字符与波浪符（~）是同一个键，它与单引号不同。反引号用来安全地引用可能与已存在的MySQL关键词重复的表名或列名。
- 如果MySQL对你的上一条查询发出警告，使用SHOW WARNINGS命令可以显示问题（参见图5-7）。
- INSERT中一个有意思的变化是REPLACE。REPLACE语句的作用是，如果使用的表的主键或唯一索引的值已经存在，那么REPLACE会更新存在的行；如果不存重复则会同INSERT一样插入新行。

5.3 选择数据

既然数据库中有一些记录，就可以使用所有SQL名词中最常用的名词SELECT来检索信息。SELECT查询使用以下语法返回与某一条件匹配的记录行：

```
SELECT which_columns FROM which_table
```

最简单的SELECT查询是：

```
SELECT * FROM tablename
```

星号意味着你想查看每一列。另外，还可以指定要返回的列，并用逗号把这些列相互隔开。

```
SELECT column1, column3 FROM tablename
```

明确要选择哪些列有几个好处。第一个好处是性能，没有理由获取将不会使用的列。第二个好处是顺序，可以用一种不同于它们在表中布局的顺序返回列。第三个好处（你将在本章后面看到）是，指定列允许你利用函数操纵那些列中的值。

从表中选择数据

(1) 从users表中检索所有数据（参见图5-13）。

The screenshot shows a terminal window titled "PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide". The MySQL prompt "mysql>" is at the top. The command "SELECT * FROM users;" is entered. The output shows 26 rows of data from the users table, each containing user_id, first_name, last_name, email, pass, and registration_date. The data includes entries for Larry Ullman, Zoe Isabella, John Lennon, Paul McCartney, George Harrison, Ringo Starr, David Jones, Peter Tork, Micky Dolenz, Mike Nesmith, David Sedaris, Nick Hornby, Melissa Bank, Toni Morrison, Jonathan Franzen, Don DeLillo, Graham Greene, Michael Chabon, Richard Brautigan, Russell Banks, Homer Simpson, Marge Simpson, Bart Simpson, Lisa Simpson, Maggie Simpson, and Abe Simpson. The registration dates range from 2011-03-31 to 2011-03-31 at 21:16:47.

<i>user_id</i>	<i>first_name</i>	<i>last_name</i>	<i>email</i>	<i>pass</i>	<i>registration_date</i>
1	Larry	Ullman	email@example.com	e727d146aae12436e899a726da5b2f11d8381b26	2011-03-31 21:12:52
2	Zoe	Isabella	email2@example.com	b6793ca1c216835ba85c1fdbd1399ce729e34b4e5	2011-03-31 21:14:23
3	John	Lennon	john@beatles.com	2a50435b6f512f6988bd719106a258fb7e338ff	2011-03-31 21:15:07
4	Paul	McCartney	paul@beatles.com	6ae16792c502a5b47da180ce8456e5ae7d65e263	2011-03-31 21:15:07
5	George	Harrison	george@beatles.com	1af17e73721dbe040011b8ed4bb1a7dbe3ce29	2011-03-31 21:15:07
6	Ringo	Starr	ringo@beatles.com	520f73691bcf89d508d923a2dbcb8ef58ebfb52	2011-03-31 21:15:07
7	David	Jones	davey@monkees.com	ec2344e40137ef72763267f17ed6c7eb2b019f	2011-03-31 21:16:47
8	Peter	Tork	peter@monkees.com	b8f6b0c0646f68ec6f27653f8473ae4a81fd302	2011-03-31 21:16:47
9	Micky	Dolenz	micky@monkees.com	0599b6e3c9206ef135c83a921294ba6417dbc673	2011-03-31 21:16:47
10	Mike	Nesmith	mike@monkees.com	8084a1773e9985abeb1f2605e0cc22211cc58cb1b	2011-03-31 21:16:47
11	David	Sedaris	david@authors.com	f54e748ae96242104802eeb2c15a9f506a18ef72	2011-03-31 21:16:47
12	Nick	Hornby	nick@authors.com	815f12d7b9d7cd6904781015c2a0a5b3ae207c	2011-03-31 21:16:47
13	Melissa	Bank	melissa@authors.com	15ac6793642add347cbf24eb8884b97947f637099	2011-03-31 21:16:47
14	Toni	Morrison	toni@authors.com	ce3a79105879624762c01ecb8abee7b31e67df5	2011-03-31 21:16:47
15	Jonathan	Franzen	jonathan@authors.com	c969581a0a7d6f790f4b520225f34fd90a09c86f	2011-03-31 21:16:47
16	Don	DeLillo	don@authors.com	01a3f9a1b1328af3d5affcfa4cc9e539c4c455	2011-03-31 21:16:47
17	Graham	Greene	graham@authors.com	7c16e1fcfc8c3ec99790f25c310ef63febb1bb3	2011-03-31 21:16:47
18	Michael	Chabon	michael@authors.com	bd58cc413f97c33930778416a6dbd2d67720dc41	2011-03-31 21:16:47
19	Richard	Brautigan	richard@authors.com	b1f8414005c218fb53b66f17b4f671bccceca3d	2011-03-31 21:16:47
20	Russell	Banks	russell@authors.com	b6c4056557e33f1e209870a5b78ed362fb3c1b	2011-03-31 21:16:47
21	Homer	Simpson	homero@simpson.com	54a0b2dc5a944907d29304405f055234b3847	2011-03-31 21:16:47
22	Marge	Simpson	marge@simpson.com	ceab0c7b57e183dea0e4cf00489fe073908c0c	2011-03-31 21:16:47
23	Bart	Simpson	bart@simpson.com	73265774ab01028ed8e06af5fa0f9a7ccb6a	2011-03-31 21:16:47
24	Lisa	Simpson	lisa@simpson.com	a09bb169171ec0759dff75c088f004205c9e27b	2011-03-31 21:16:47
25	Maggie	Simpson	maggie@simpson.com	0e87350b393ceced1d4751b828d18102be123edb	2011-03-31 21:16:47
26	Abe	Simpson	abe@simpson.com	6591827c8e3d4624e8fc1e324f31fa389fdfaf4	2011-03-31 21:16:47

26 rows in set (0.00 sec)

mysql>

图5-13 SELECT * FROM *tablename*查询会返回表中存储的每一条记录的每一列

```
SELECT * FROM users;
```

这个非常基本的SQL命令将检索那个表内存储的每一行的每一列。

(2) 只从users表中检索名字和姓氏（参见图5-14）。

```
SELECT first_name, last_name
FROM users;
```

```
mysql> SELECT first_name, last_name
   -> FROM users;
+-----+-----+
| first_name | last_name |
+-----+-----+
| Larry      | Ullman    |
| Zoe        | Isabella  |
| John       | Lennon    |
| Paul       | McCartney|
| George     | Harrison  |
| Ringo      | Starr     |
| David      | Jones     |
| Peter      | Tork      |
| Micky      | Dolenz    |
| Mike       | Nesmith   |
| David      | Sedaris   |
| Nick       | Hornby    |
| Melissa    | Bank      |
| Toni       | Morrison  |
| Jonathan   | Franzen   |
| Don        | DeLillo   |
| Graham     | Greene    |
| Michael    | Chabon    |
| Richard    | Brautigan |
| Russell    | Banks     |
| Homer      | Simpson   |
| Marge      | Simpson   |
| Bart       | Simpson   |
| Lisa       | Simpson   |
| Maggie    | Simpson   |
| Abe        | Simpson   |
+-----+-----+
26 rows in set (0.00 sec)

mysql>
```

图5-14 该查询会返回表中的所有记录，但只会返回两列

如果无需显示users表中每个字段的数据，就可以使用SELECT语句来限制只查看相关的信息。

✓ 提示

- 在phpMyAdmin中，Browse选项卡用于运行一个简单的SELECT查询。
- 在使用SELECT时，实际上无需指定表或列。例如，`SELECT NOW();`（参见图5-15）。
- 在SELECT语句中列出的列的顺序指定了展示值的顺序（比较图5-14与图5-16）。
- 利用SELECT查询，甚至可以对相同的列检索多次，这种特性允许以多种不同的方式操纵列的数据。

```
mysql> SELECT NOW();
+-----+
| NOW()           |
+-----+
| 2011-03-31 21:38:41 |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-15 在不指定数据库或表的情况下，也可以运行许多查询。这个查询选择调用NOW()函数的结果，它会返回当前的日期和时间（依据MySQL）

```

    PHP and MySQL for Dynamic ...
mysql> SELECT last_name, first_name
-> FROM users;
+-----+-----+
| last_name | first_name |
+-----+-----+
| Ullman   | Larry
| Isabella | Zoe
| Lennon   | John
| McCartney | Paul
| Harrison | George
| Starr    | Ringo
| Jones    | David
| Tork     | Peter
| Dolenz   | Micky
| Nesmith  | Mike
| Sedaris  | David
| Hornby   | Nick
| Bank     | Melissa
| Morrison | Toni
| Franzen  | Jonathan
| DeLillo  | Don
| Greene   | Graham
| Chabon   | Michael
| Brautigan | Richard
| Banks    | Russell
| Simpson  | Homer
| Simpson  | Marge
| Simpson  | Bart
| Simpson  | Lisa
| Simpson  | Maggie
| Simpson  | Abe
+-----+-----+
26 rows in set (0.00 sec)

mysql>
  
```

图5-16 如果SELECT查询指定了要返回的列，就以指定列的顺序来显示它们

5.4 使用条件语句

迄今为止使用的SELECT查询总会从表中检索每一条记录。但是，通常你希望基于某些条件来限制返回哪些行。可以通过向SELECT查询中添加条件语句来执行该任务。这些条件语句使用SQL名词WHERE，其编写方式与你在PHP中编写条件语句非常相似。

```
SELECT which_columns FROM which_table WHERE condition(s)
```

表5-1列出了在WHERE条件语句中使用的最常见的运算符。例如，简单的相等性检查：

```
SELECT name FROM people
WHERE birth_date = '2011-01-26'
```

表5-1 WHERE表达式常常（但不是独占地）使用的MySQL运算符

MySQL运算符	含 义
=	等于
<	小于
>	大于
<=	小于或等于
>=	大于或等于

(续)

MySQL运算符	含 义
!= (或 <>)	不等于
IS NOT NULL	具有一个值
IS NULL	没有值
IS TRUE	有一个真值
IS FALSE	有一个假值
BETWEEN	在范围内
NOT BETWEEN	在范围外
IN	在值列表中找到
NOT IN	未在值列表中找到
OR (或)	两个条件语句之一为真
AND (或 &&)	两个条件语句都为真
NOT (或 !)	条件语句不为真
XOR	两个条件语句只有一个为真

5

可以结合使用各种运算符以及圆括号来创建更复杂的表达式：

```
SELECT * FROM items WHERE
(price BETWEEN 10.00 AND 20.00) AND
(quantity > 0)
SELECT * FROM cities WHERE
(zip_code = 90210) OR (zip_code = 90211)
```

上一个示例也可以写成：

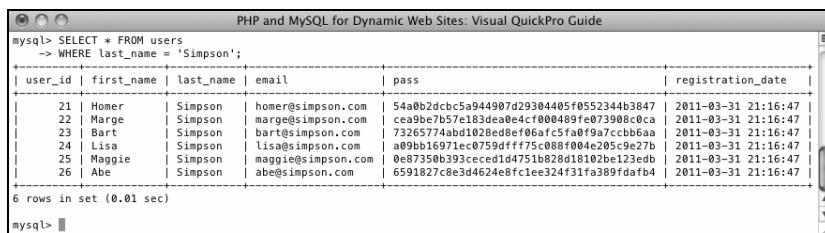
```
SELECT * FROM cities WHERE
zip_code IN (90210, 90211)
```

为了示范使用条件语句，让我们在sitename数据库上运行另外一些SELECT查询。下面的示例仅仅列出了少数几种可能的情况。通过学习本章以及全书的内容，你将看到如何在各类查询中使用条件语句的方法。

使用条件语句

(1) 选择其姓氏为Simpson的所有用户 (参见图5-17)。

```
SELECT * FROM users
WHERE last_name = 'Simpson';
```



The screenshot shows the MySQL command-line interface with the following output:

```
mysql> SELECT * FROM users
-> WHERE last_name = 'Simpson';
+-----+-----+-----+-----+-----+
| user_id | first_name | last_name | email | pass |
+-----+-----+-----+-----+-----+
| 21 | Homer | Simpson | homer@simpson.com | 54a0b2dcfc5a944907d29304405f0552344b3047 |
| 22 | Marge | Simpson | marge@simpson.com | cea9be7b57e183de0a4cf0800489fe0e73908e0ca |
| 23 | Bart | Simpson | bart@simpson.com | 73265774ab1d028e8cf06sf5fa0f9a7ccbbaaa |
| 24 | Lisa | Simpson | lisa@simpson.com | a09bb16971ec0759dff775c088f004e205c9e27b |
| 25 | Maggie | Simpson | maggie@simpson.com | 0e87350b393cecced1d4751b828d18102b123edb |
| 26 | Abe | Simpson | abe@simpson.com | 6591827c8e3d4624e8fc1ee324f31fa389fdafab4 |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

图5-17 所有注册的Simpson

这个简单的查询将返回其last_name值为Simpson的每一行的每一列。(如果你的数据库中的数据与本例不同，可以相应地修改查询语句。)

(2) 选择其姓氏为Simpson的所有用户的名字(参见图5-18)。

```
SELECT first_name FROM users
WHERE last_name = 'Simpson';
```

```
C:\Windows\system32\cmd.exe ...
mysql> SELECT first_name FROM users
-> WHERE last_name = 'Simpson';
+-----+
| first_name |
+-----+
| Homer      |
| Marge      |
| Bart       |
| Lisa       |
| Maggie     |
| Abe        |
+-----+
6 rows in set <0.00 sec>
mysql> _
```

图5-18 只返回注册的所有Simpson的名字

这里，只会为每一行返回一列(first_name)。尽管看上去有些奇怪，但是不必选择对其执行WHERE子句的列。其原因是在SELECT后面列出的列只规定返回哪些列，而在WHERE中列出的列则规定返回哪些行。

(3) 选择users表中没有电子邮件地址的每一条记录的每一列(参见图5-19)。

```
SELECT * FROM users
WHERE email IS NULL;
```

```
C:\Windows\system32\...
mysql> SELECT * FROM users
-> WHERE email IS NULL;
Empty set <0.00 sec>
mysql> _
```

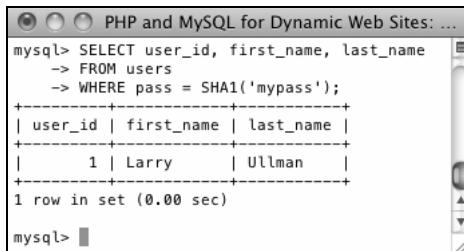
图5-19 这个查询不会返回任何记录，因为电子邮件不能有NULL值。因此，这个查询确实会工作，它只是没有匹配的记录

IS NULL条件语句的意思是说没有值。记住，就NULL来说，空字符串等同于有一个值，因此不匹配这个条件。不过，这样一种情况将匹配：

```
SELECT * FROM users WHERE email='';
```

(4) 选择其中的密码为mypass的所有记录的用户ID、名字和姓氏(参见图5-20)。

```
SELECT user_id, first_name, last_name
FROM users
WHERE pass = SHA1('mypass');
```



```
mysql> SELECT user_id, first_name, last_name
-> FROM users
-> WHERE pass = SHA1('mypass');
+-----+-----+-----+
| user_id | first_name | last_name |
+-----+-----+-----+
|      1 | Larry      | Ullman    |
+-----+-----+-----+
1 row in set (0.00 sec)

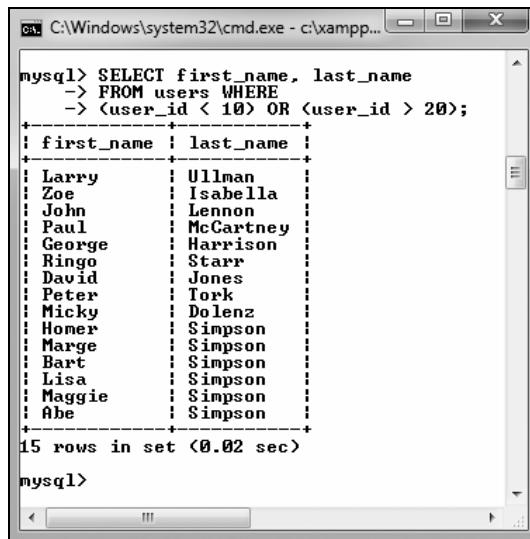
mysql>
```

图5-20 条件语句可以使用函数，如这里的SHA1()

因为存储的密码会用SHA1()函数加密，所以可以在条件语句中使用相同的加密函数来匹配它。SHA1()区分大小写，因此，仅当密码（存储的密码与查询的密码）完全匹配时，这个查询才会工作。

(5) 选择其用户ID小于10或大于20的用户名（参见图5-21）。

```
SELECT first_name, last_name
FROM users WHERE
(user_id < 10) OR (user_id > 20);
```



```
ca C:\Windows\system32\cmd.exe - c:\xampp...
mysql> SELECT first_name, last_name
-> FROM users WHERE
-> <user_id < 10> OR <user_id > 20>;
+-----+-----+
| first_name | last_name |
+-----+-----+
| Larry      | Ullman    |
| Zoe        | Isabella |
| John       | Lennon    |
| Paul       | McCartney |
| George     | Harrison  |
| Ringo      | Starr     |
| David      | Jones     |
| Peter      | Tork      |
| Micky      | Dolenz   |
| Homer      | Simpson  |
| Marge      | Simpson  |
| Bart       | Simpson  |
| Lisa       | Simpson  |
| Maggie     | Simpson  |
| Abe        | Simpson  |
+-----+-----+
15 rows in set <0.02 sec>

mysql>
```

图5-21 这个查询使用两个条件语句和OR运算符

这个相同的查询也可以写为：

```
SELECT first_name, last_name
FROM users WHERE user_id
NOT BETWEEN 10 and 20;
```

或者甚至写为：

```
SELECT first_name, last_name FROM users WHERE user_id NOT IN
(10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20);
```

✓ 提示

- 可以使用数学加法 (+)、减法 (-)、乘法 (*) 和除法 (/) 运算符在查询内执行数学计算。
- MySQL支持关键字TRUE和FALSE，不区分大小写。在内部，TRUE等于1而FALSE等于0。因此， $\text{TRUE}+\text{TRUE}=2$ 。

5.5 使用 LIKE 和 NOT LIKE

在条件语句中使用数字、日期和NULL是一个直观的过程，但是，字符串可能更具技巧性。你可以用如下查询来检查字符串相等性：

```
SELECT * FROM users
WHERE last_name = 'Simpson'
```

不过，以一种更宽松的方式比较字符串需要另外一些运算符和字符。例如，如果你想匹配其姓氏可能是Smith、Smiths或Smithson的人，那么需要更灵活的条件语句。在这里，LIKE和NOT LIKE就派上了用场。它们（主要用于字符串）与两个通配符一起使用。这两个通配符是：下划线（_），用于匹配单个字符；百分号（%），用于匹配0个或多个字符。在姓氏示例中，我将编写的查询如下：

```
SELECT * FROM users
WHERE last_name LIKE 'Smith%'
```

这个查询将返回其last_name值以Smith开头的所有行。因为它默认是一种不区分大小写的查找，所以也适用于以smith开头的名字。

使用LIKE

(1) 选择其中的姓氏以Bank开头的所有记录（参见图5-22）。

```
SELECT * FROM users
WHERE last_name LIKE 'Bank%';
```

user_id	first_name	last_name	email	pass	registration_date
13	Melissa	Bank	melissa@authors.com	15ac6793642add347cbf24b8884b97947f637091	2011-03-31 21:16:47
20	Russell	Banks	russell@authors.com	6bc4056557e33f1e209870ab578ed362f8b3c1b8	2011-03-31 21:16:47

图5-22 SQL名词LIKE增加了条件语句的灵活性。这个查询匹配其中的姓氏值以Bank开头的任何记录

(2) 选择其电子邮件地址不是*something@authors.com*形式的每条记录的名字（参见图5-23）。

```
SELECT first_name, last_name
FROM users WHERE
email NOT LIKE '%@authors.com';
```

```

mysql> SELECT first_name, last_name
-> FROM users WHERE
-> email NOT LIKE '%@authors.com';
+-----+-----+
| first_name | last_name |
+-----+-----+
| Larry      | Ullman   |
| Zoe        | Isabella |
| John       | Lennon   |
| Paul       | McCartney |
| George     | Harrison |
| Ringo      | Starr    |
| David      | Jones    |
| Peter      | Tork     |
| Micky      | Dolenz   |
| Mike       | Nesmith  |
| Homer      | Simpson  |
| Marge      | Simpson  |
| Bart       | Simpson  |
| Lisa       | Simpson  |
| Maggie     | Simpson  |
| Abe        | Simpson  |
+-----+-----+
16 rows in set (0.00 sec)

mysql>

```

图5-23 NOT LIKE条件语句基于值不包含什么来返回记录

如果想排除字符串中存在某些值，那么可以结合使用NOT LIKE和通配符。

✓ 提示

- 带有LIKE条件语句的查询一般比较慢，因为它们不能利用索引，所以仅当绝对需要时才应该使用这种格式。
- 在查询中，可以在字符串的前面和/或后面使用通配符。


```
SELECT * FROM users
WHERE last_name LIKE '_smith%'
```
- 尽管LIKE和NOT LIKE通常用于字符串，但是它们也可以用于数字列。
- 为了在LIKE或NOT LIKE查询中使用原意的下划线或百分号，需要对它们进行转义（在字符前放置一个反斜杠），使得不会把它们与通配符相混淆。
- 下划线可以与其自身组合使用。例如，LIKE '__'将找出两个字母的任意组合。
- 在下一章中，你将学习FULLTEXT查找，它通常比LIKE查找更好一些。

5.6 排序查询结果

默认情况下，将以无意义的顺序返回SELECT查询的结果。为了以一种有意义的方式对它们进行排序，可以使用ORDER BY子句。

```
SELECT * FROM tablename ORDER BY column
SELECT * FROM orders ORDER BY total
```

使用ORDER BY时的默认顺序是升序（简写为ASC），这意味着数字将从小到大递增，而日期将从过去到最近排列，文本则按字母顺序排序。可以通过指定降序（简写为DESC）来颠倒这种顺序。

```
SELECT * FROM tablename
ORDER BY column DESC
```

甚至可以按多个列对返回的值进行排序：

```
SELECT * FROM tablename
ORDER BY column1, column2
```

可以（并且常常）结合使用ORDER BY与WHERE或其他子句。这样做时，可以把ORDER BY放在条件后面：

```
SELECT * FROM tablename WHERE conditions
ORDER BY column
```

排序数据

(1) 按姓氏以字母顺序选择所有用户（参见图5-24）。

```
SELECT first_name, last_name FROM
users ORDER BY last_name;
```

first_name	last_name
Melissa	Bank
Russell	Banks
Richard	Brautigan
Michael	Chabon
Don	DeLillo
Micky	Dolenz
Jonathan	Franzen
Graham	Greene
George	Harrison
Nick	Hornby
Zoe	Isabella
David	Jones
John	Lennon
Paul	McCartney
Toni	Morrison
Mike	Nesmith
David	Sedaris
Bart	Simpson
Lisa	Simpson
Maggie	Simpson
Marge	Simpson
Homer	Simpson
Abe	Simpson
Ring	Starr
Peter	Tork
Larry	Ullman

26 rows in set (0.00 sec)

图5-24 按姓氏以字母顺序排列的记录

如果把这些结果与图5-12中的结果进行比较，就会看到使用ORDER BY的好处。

(2) 先按姓氏再按名字以字母顺序显示所有用户（参见图5-25）。

```
SELECT first_name, last_name FROM
users ORDER BY last_name ASC,
first_name ASC;
```

在这个查询中，其作用是返回每一行记录，这些记录首先按last_name进行排序，然后在last_name内按first_name进行排序。在Simpson当中，其作用最明显。



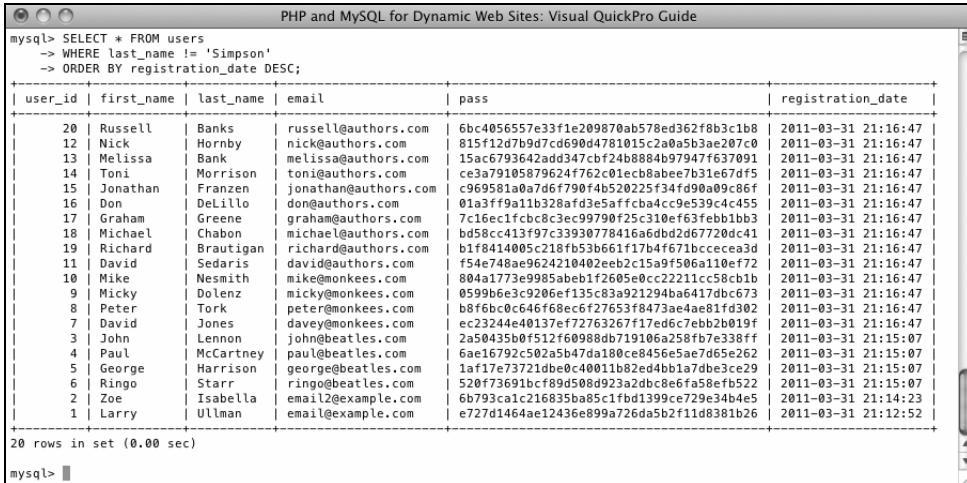
PHP and MySQL for Dynamic Web ...
mysql> SELECT first_name, last_name FROM
-> users ORDER BY last_name ASC,
-> first_name ASC;
+-----+-----+
| first_name | last_name |
+-----+-----+
Melissa	Bank
Russell	Banks
Richard	Brautigan
Michael	Chabon
Don	DeLillo
Micky	Dolenz
Jonathan	Franzen
Graham	Greene
George	Harrison
Nick	Hornby
Zoe	Isabella
David	Jones
John	Lennon
Paul	McCartney
Toni	Morrison
Mike	Nesmith
David	Sedaris
Abe	Simpson
Bart	Simpson
Homer	Simpson
Lisa	Simpson
Maggie	Simpson
Marge	Simpson
Ringo	Starr
Peter	Tork
Larry	Ullman
+-----+-----+
26 rows in set (0.00 sec)
mysql>

5

图5-25 首先按姓氏，然后在姓氏内按名字以字母顺序进行排序的记录

(3) 按注册的日期显示所有不是Simpson的用户（参见图5-26）。

```
SELECT * FROM users
WHERE last_name != 'Simpson'
ORDER BY registration_date DESC;
```



PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT * FROM users
-> WHERE last_name != 'Simpson'
-> ORDER BY registration_date DESC;
+-----+-----+-----+-----+-----+
| user_id | first_name | last_name | email | pass | registration_date |
+-----+-----+-----+-----+-----+
20	Russell	Banks	russell@authors.com	6bc4056557e33f1e209870ab578ed362f8b3c1b8	2011-03-31 21:16:47
12	Nick	Hornby	nick@authors.com	815f12d7b9d7c6d90d478105c2a0b5b3ae207c0	2011-03-31 21:16:47
13	Melissa	Bank	melissa@authors.com	15ac6793642ad3d47cf24b884b97947f637091	2011-03-31 21:16:47
14	Toni	Morrison	toni@authors.com	ce379105879624f762c502ecb8abeb731e67df5	2011-03-31 21:16:47
15	Jonathan	Franzen	jonathan@authors.com	c969581a0a7d6f790ff4b52025f34fd90a09c86f	2011-03-31 21:16:47
16	Don	DeLillo	don@authors.com	01a3ff9a1b328af3e5affcb4c4c9e539c4c455	2011-03-31 21:16:47
17	Graham	Greene	graham@authors.com	7c16e1fc8c83ec99790f25c310ef63febb1bb3	2011-03-31 21:16:47
18	Michael	Chabon	michael@authors.com	bd58cc413f97c33930778416adb2d267720dc41	2011-03-31 21:16:47
19	Richard	Brautigan	richard@authors.com	b1f8414005c2187b53b66f1f74671b1cceceaa3	2011-03-31 21:16:47
11	David	Sedaris	david@authors.com	f54e748ae962421042ee2b215a9f506a110ef72	2011-03-31 21:16:47
10	Mike	Nesmith	mike@monkees.com	804a1773e9985abe1f2605e0c22211cc58cb1b	2011-03-31 21:16:47
9	Micky	Dolenz	micky@monkees.com	0599b6e3c9206e135c83a921294ba6417dbc673	2011-03-31 21:16:47
8	Peter	Tork	peter@monkees.com	b8f6b0c646f68ec6f27653f8473ae4ae81fd302	2011-03-31 21:16:47
7	David	Jones	davey@monkees.com	ec23244e40137ef72763267f17ed6c7ebb2b01ff	2011-03-31 21:16:47
3	John	Lennon	john@beatles.com	2a50435b0f512f60988dbf79106a258fb7e338ff	2011-03-31 21:15:07
4	Paul	McCartney	paul@beatles.com	6ae16792c502a5b47d1a180ce84565ae7d65e262	2011-03-31 21:15:07
5	George	Harrison	george@beatles.com	1af17e73721dbe0c40011b82ed4b1a7dbe3ce29	2011-03-31 21:15:07
6	Ringo	Starr	ringo@beatles.com	520f73691bcf89d508d923a2dbcbe6fa58efb522	2011-03-31 21:15:07
2	Zoe	Isabella	email2@example.com	6b793ca1c216835b8a5c1fb1399ce729e34b4e5	2011-03-31 21:14:23
1	Larry	Ullman	email@example.com	e277d1464ae12436e899a726da5b2f11d83b1b26	2011-03-31 21:12:52
+-----+-----+-----+-----+-----+
20 rows in set (0.00 sec)
mysql>

图5-26 按注册的日期显示所有不是Simpson的用户，最近注册的用户列在最前面

可以在任何列类型（包括数字和日期）上使用ORDER BY。也可以在带有条件语句的查询中使用这个子句，并且把ORDER BY放在WHERE之后。

✓ 提示

- 由于MySQL可以与许多语言协同工作，ORDER BY将基于使用的排序方式（参见第16章）。
- 如果要排序的列是ENUM类型，排序会基于列创建时的ENUM值的顺序。例如，如果有一个gender列，定义为ENUM ('M','F')，子句ORDER BY gender返回的结果是M记录在前。

5.7 限制查询结果

可以添加到查询语句中的另一个SQL子句是LIMIT。在SELECT查询中，WHERE指示返回哪些记录，ORDER BY决定如何对这些记录进行排序，但是LIMIT用于指定要返回多少条记录。其用法如下：

```
SELECT * FROM tablename LIMIT x
```

在这类查询中，只会返回查询中的前x条记录。为了只返回三条匹配的记录，可以使用如下代码：

```
SELECT * FROM tablename LIMIT 3
```

使用下面这种格式：

```
SELECT * FROM tablename LIMIT x, y
```

可以返回从x条记录开始的y条记录。为了返回第11~20条记录，可以编写如下代码：

```
SELECT * FROM tablename LIMIT 10, 10
```

像PHP中的数组一样，在使用LIMIT时结果集从0开始，因此10代表第11条记录。

可以与WHERE和/或ORDER BY一起使用LIMIT，总是将LIMIT放在查询的末尾。

```
SELECT which_columns FROM tablename WHERE  
conditions ORDER BY column LIMIT x
```

限制返回的数据量

(1) 选择最后5位注册的用户（参见图5-27）。

```
SELECT first_name, last_name  
FROM users ORDER BY  
registration_date DESC LIMIT 5;
```

first_name	last_name
Abe	Simpson
Jonathan	Franzen
Don	DeLillo
Graham	Greene
Michael	Chabon

图5-27 使用LIMIT子句，查询可以返回数量具体的记录数

要返回最近的任何内容，必须按日期以降序对数据排序。然后，可以向查询中添加LIMIT 5来查看最近的5条记录。

(2) 选择要注册的第二个人（参见图5-28）。

```
SELECT first_name, last_name
FROM users ORDER BY
registration_date ASC LIMIT 1, 1;
```

```
C:\Windows\system32\cmd.exe - c:\xa...
mysql> SELECT first_name, last_name
-> FROM users ORDER BY
-> registration_date ASC LIMIT 1, 1;
+ first_name | last_name +
+ Ringo      | Starr    +
1 row in set <0.00 sec>

mysql>
```

图5-28 借助LIMIT子句，使用LIMIT x, y格式，查询甚至可以返回一组记录中间的记录

这看起来可能有些奇怪，但它确实很好地应用了迄今为止所学的知识。首先，我按registration_date以升序对所有记录排序，因此会返回要注册的第一个人。然后，限制返回的结果开始于1（它将是第二行），并只返回一条记录。

✓ 提示

- 在显示多页查询结果时（在多个页面上成块显示它们），常常使用LIMIT x, y子句。在第9章中将会看到这样的示例。
- LIMIT子句不会改进查询的执行速度，因为MySQL仍然必须把每一条记录集合到一起，然后截短列表。但在mysql客户或PHP脚本中，LIMIT子句将把要处理的数据量减至最少。
- LIMIT名词不是SQL标准的一部分，因此（令人悲哀的是）并非在所有的数据库上都可以使用它。
- LIMIT不仅仅可以与SELECT一起使用，还可以与大多数类型的查询一起使用。

5.8 更新数据

一旦表中包含一些数据，就可能需要编辑现有的记录。如果错误地输入了信息或者如果数据（如姓氏或电子邮件地址）有了变化，就需要这样做。更新记录的语法如下：

```
UPDATE tablename SET column=value
```

可以一次改变一条记录中多列的内容，它们相互之间用逗号隔开。

```
UPDATE tablename SET column1=valueA,
column5=valueB...
```

通常希望使用WHERE子句来指定要更新哪些行；否则，将对每一行都执行更改。

```
UPDATE tablename SET column2=value
WHERE column5=value
```

更新（以及删除）是使用主键的最重要的原因之一。这个值（它应该永远不会发生变化）可以作为WHERE子句中的参考点，即使其他所有字段都需要改变也是如此。

更新记录

(1) 找到要更新记录的主键（参见图5-29）。

```
SELECT user_id FROM users
WHERE first_name = 'Michael'
AND last_name='Chabon';
```

```
mysql> SELECT user_id FROM users
-> WHERE first_name = 'Michael'
-> AND last_name='Chabon';
+-----+
| user_id |
+-----+
|     18   |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-29 在更新记录之前，确定在WHERE子句中使用哪个主键

在本示例中，我将更改这个作者的记录中的电子邮件。要执行该操作，必须先找出那个记录的主键，这是由该查询完成的。

(2) 更新记录（参见图5-30）。

```
UPDATE users
SET email='mike@authors.com'
WHERE user_id = 18;
```

```
mysql> UPDATE users
-> SET email='mike@authors.com'
-> WHERE user_id = 18;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1    Changed: 1    Warnings: 0

mysql>
```

图5-30 这个查询只改变了一行中一列的值

要更改电子邮件地址，可以使用一个UPDATE查询，它使用主键（user_id）来指定应该更新哪一条记录。MySQL将会报告查询成功执行，以及有多少行受到了影响。

(3) 确认所做的更改（参见图5-31）。

```
SELECT * FROM users
WHERE user_id=18;
```

```
mysql> SELECT * FROM users
-> WHERE user_id=18;
+-----+-----+-----+-----+-----+-----+
| user_id | first_name | last_name | email      | pass          | registration_date |
+-----+-----+-----+-----+-----+-----+
|     18   | Michael    | Chabon    | mike@authors.com | bd58cc413f97c33930778416a6dbd2d67720dc41 | 2011-03-31 21:16:47 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

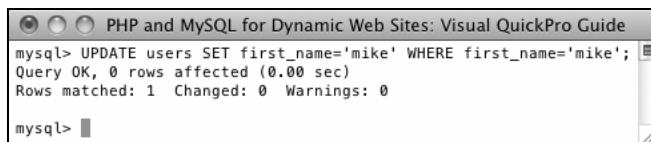
mysql>
```

图5-31 作为最后一步，可以通过再次选择记录来确认更新

尽管MySQL已经指示更新成功（参见图5-30），但它仍然乐意再次选择记录来确认发生了正确的更改。

✓ 提示

- 无论何时使用UPDATE，都要使用WHERE条件语句，除非你希望更改会影响每一行。
 - 如果运行一个实际上不会更改任何值的更新查询（例如：UPDATE users SET first_name='mike' WHERE first_name='mike'），将不会看到任何错误，但是也不会影响任何行。MySQL的最新版本将显示与查询匹配的X行，但不会更改任何行。
 - 要防止自己意外地更新过多的行，可以对UPDATE应用一个LIMIT子句。
- 5
- ```
UPDATE users SET email='mike@authors.com' WHERE user_id = 18 LIMIT 1
```
- 应该从来都不需要在主键列上执行一个UPDATE，因为这个值应该从不发生变化。更改主键的值可能造成严重的后果。
  - 要在phpMyAdmin中更新记录，可以使用SQL窗口或选项卡运行UPDATE查询。此外，也可运行SELECT查询查找想更新的记录，然后单击记录旁边的铅笔图标（参见图5-32）。这将调出一个类似于图5-10的表单，可以在其中编辑记录的当前值。



A screenshot of a MySQL command-line interface window titled "PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide". The window shows the following SQL command and its execution results:

```
mysql> UPDATE users SET first_name='mike' WHERE first_name='mike';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1 Changed: 0 Warnings: 0
mysql>
```

图5-32 在phpMyAdmin中浏览记录的部分视图。单击铅笔图标编辑记录，  
单击X图标删除记录

## 5.9 删除数据

除了更新现有的记录之外，你可能需要采取的另外一个步骤是彻底从数据库中删除记录。要执行该操作，可以使用DELETE命令。

```
DELETE FROM tablename
```

这样编写的命令将删除表中的每一条记录，再次使之为空。一旦删除了一条记录，就无法再取回它。在大多数情况下，你都希望删除单个行，而不是删除所有行。为了执行该操作，可以使用WHERE子句：

```
DELETE FROM tablename WHERE condition
```

### 删除记录

(1) 找到要删除记录的主键（参见图5-33）。

```
SELECT user_id FROM users
WHERE first_name='Peter'
AND last_name='Tork';
```

就像UPDATE示例一样，首先需要确定要为删除使用哪个主键。

```
mysql> SELECT user_id FROM users
-> WHERE first_name='Peter'
-> AND last_name='Tork';
+-----+
| user_id |
+-----+
| 8 |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-33 user\_id将用于在DELETE查询中引用这条记录

(2) 预览在执行删除时将会发生什么 (参见图5-34)。

```
SELECT * FROM users
WHERE user_id = 8;
```

```
mysql> SELECT * FROM users
-> WHERE user_id = 8;
+-----+-----+-----+-----+-----+
| user_id | first_name | last_name | email | pass | registration_date |
+-----+-----+-----+-----+-----+
| 8 | Peter | Tork | peter@monkees.com | b8f6bc0c646f68ec6f27653f8473ae4ae81fd302 | 2011-03-31 21:16:47 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

图5-34 要预览DELETE查询的效果，首先运行一个语法上类似的SELECT查询

防止错误删除的一种确实很好的技巧是，首先使用SELECT \*而不是DELETE运行查询。这个查询的结果将表示哪些行将会受到删除的影响。

(3) 删除记录 (参见图5-35)。

```
DELETE FROM users
WHERE user_id = 8 LIMIT 1;
```

```
mysql> DELETE FROM users
-> WHERE user_id = 8 LIMIT 1;
Query OK, 1 row affected (0.00 sec)

mysql>
```

图5-35 从表中删除记录

如同更新一样，MySQL将报告查询的成功执行，以及有多少行将会受到影响。此时，无法恢复已删除的记录，除非事先备份了数据库。

即使SELECT查询(参见第(2)步和图5-34)只返回一行，也要格外小心，将LIMIT 1子句添加到DELETE查询中。

(4) 确认进行了更改 (参见图5-36)。

```
SELECT user_id FROM users
WHERE first_name='Peter'
AND last_name='Tork';
```

```
mysql> SELECT user_id FROM users
-> WHERE first_name='Peter'
-> AND last_name='Tork';
Empty set (0.00 sec)

mysql>
```

图5-36 其user\_id是8的记录不再是这个表的一部分

也可以通过运行第(1)步中的查询来确认更改。

#### ✓ 提示

- 清空一个表的首选方式是使用TRUNCATE:

```
TRUNCATE TABLE tablename
```

- 要删除表中的所有数据以及表本身，可以使用DROP TABLE:

```
DROP TABLE tablename
```

- 要删除整个数据库（包括其中的每一个表及其所有数据），可以使用:

```
DROP DATABASE databasename
```

## 5.10 使用函数

在本章最后，将学习在MySQL查询中可以使用的许多函数。你已经看到了两个函数——NOW()和SHA1()，但是它们只是冰山一角。你将在这里看到的大多数函数与SELECT查询一起用于格式化和改变返回的数据，但是你也可以在许多不同类型的查询中使用MySQL函数。

要对列的值应用某个函数，查询看起来如下所示：

```
SELECT FUNCTION(column) FROM tablename
```

要对一个列的值应用某个函数，同时还选择其他一些列，可以编写如下任何一种查询：

- SELECT \*, FUNCTION(column) FROM tablename
- SELECT column1, FUNCTION(column2), column3 FROM tablename

在学习实际的函数之前，要记住另外几点。第一，函数既可以应用于存储的数据（即列），也可以应用于字面量值。下面的UPPER()函数（用于大写字符串）的任何一种用法都是有效的：

```
SELECT UPPER(first_name) FROM users
SELECT UPPER('this string')
```

第二，虽然函数名本身不区分大小写，但是，我将继续以全部大写的格式书写它们，以便把它们与表和列名称区分开（因为我还会大写SQL名词）。第三，关于函数的一个重要规则是，在MySQL中，函数名与左括号之间不能有空格，括号内的空格是可接受的。最后一点是，当使用函数来格式化返回的数据时，你通常希望利用别名（alias），在框注“别名”中讨论了这个概念。

#### ✓ 提示

- 就像SQL有不同的标准，不同的数据库应用有些自己的语言特色，一些函数对所有数据库通用，而别一些函数只能用于特定数据库。本章，乃至全书，都只关注MySQL函数。

□ 第7章介绍了另外两个MySQL函数——分组和加密函数。

### 别名

别名 (alias) 只是查询中对表或列进行的符号性重命名。别名通常应用于表、列或函数调用，它提供了引用某个对象的快捷方式。使用AS名词来创建别名：

```
SELECT registration_date AS reg
FROM users
```

别名是区分大小写的字符串，由数字、字母和下划线组成，但是通常保持较短的长度。如你在下面的示例中所看到的，别名反映在返回结果的标题中。对于以前的示例，返回的查询结果将包含一个数据列，它被命名为`reg`。

如果在表或列上定义了一个别名，那么整个查询必须一致地使用那个相同的别名，而不使用原来的名字。例如

```
SELECT first_name AS name FROM users WHERE name='Sam'
```

这有别于标准的SQL，标准的SQL不支持在WHERE条件语句中使用别名。

## 5.10.1 文本函数

将要演示的第一组函数是那些旨在操纵文本的函数。表5-2中列出了这一类中的大多数函数。

表5-2 MySQL的一些用于处理文本的函数。与大多数函数一样，可以对列或字面量值（用t、t1、t2等表示它们）应用这些函数

| 函 数         | 用 法                       | 用 途                    |
|-------------|---------------------------|------------------------|
| CONCAT()    | CONCAT(t1, t2, ...)       | 创建形如t1t2的新字符串          |
| CONCAT_WS() | CONCAT_WS(s, t1, t2, ...) | 创建形如t1s1t2s2的新字符串      |
| LENGTH()    | LENGTH(t)                 | 返回t中的字符数               |
| LEFT()      | LEFT(t, y)                | 从t中返回最左边的y个字符          |
| RIGHT()     | RIGHT(t, x)               | 从t中返回最右边的x个字符          |
| TRIM()      | TRIM(t)                   | 从t的开头和末尾删除多余的空格        |
| UPPER()     | UPPER(t)                  | 大写t中的所有字符              |
| LOWER()     | LOWER(t)                  | 小写t中的所有字符              |
| REPLACE()   | REPACE(t1, t2, t3)        | 把t1字符串中的t2替换成t3        |
| SUBSTRING() | SUBSTRING(t, x, y)        | 从t中返回开始于x的y个字符（索引从1开始） |

`CONCAT()`（这也许是最重要的文本函数）值得特别关注。`CONCAT()`函数用于执行连接，PHP中使用句点（见第1章）。连接的语法要求把希望组合起来的多个值按顺序置于括号内，并用逗号把它们分隔开：

```
SELECT CONCAT(t1, t2) FROM tablename
```

虽然可以（并且通常会）把`CONCAT()`应用于列，但是也可以合并在单引号内输入的字符串。要将某个人的姓名格式化为*First<SPACE>Last*，你将使用：

```
SELECT CONCAT(first_name, ' ', last_name)
FROM users
```

由于连接通常会返回列的新形式，所以它是使用别名的极佳时机（参见框注“别名”）。

```
SELECT CONCAT(first_name, ' ', last_name)
AS Name FROM users
```

### 格式化文本

(1) 不使用别名来连接姓名（参见图5-37）。

```
SELECT CONCAT(last_name, ', ', first_name)
FROM users;
```

5

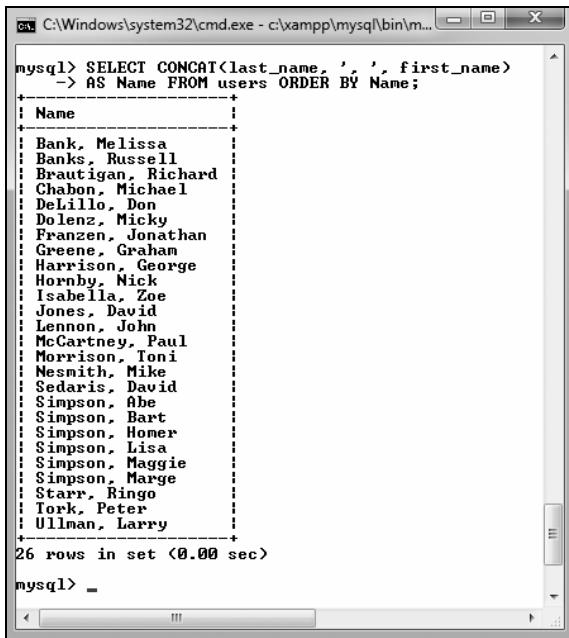
| CONCAT(last_name, ', ', first_name) |
|-------------------------------------|
| Ullman, Larry                       |
| Isabella, Zoe                       |
| Lennon, John                        |
| McCartney, Paul                     |
| Harrison, George                    |
| Starr, Ringo                        |
| Jones, David                        |
| Tork, Peter                         |
| Dolenz, Micky                       |
| Nesmith, Mike                       |
| Sedaris, David                      |
| Hornby, Nick                        |
| Bank, Melissa                       |
| Morrison, Toni                      |
| Franzen, Jonathan                   |
| DeLillo, Don                        |
| Greene, Graham                      |
| Chabon, Michael                     |
| Brautigan, Richard                  |
| Banks, Russell                      |
| Simpson, Homer                      |
| Simpson, Marge                      |
| Simpson, Bart                       |
| Simpson, Lisa                       |
| Simpson, Maggie                     |
| Simpson, Abe                        |

图5-37 这个简单的连接会返回每个注册的用户的全名。注意列标题如何使用CONCAT()函数

这个查询将演示两个要点。首先，把用户的姓氏、逗号、一个空格以及他们的名字连接成一个字符串（采用*Last, First*的格式）。其次，如图5-35所示，如果没有使用别名，那么返回数据的列标题将是函数调用。在MySQL客户端或phpMyAdmin中，这有点难看；当使用PHP连接到MySQL时，这很可能是一个问题。

(2) 在使用别名时连接姓名（参见图5-38）。

```
SELECT CONCAT(last_name, ', ', first_name)
AS Name FROM users ORDER BY Name;
```



```

mysql> SELECT CONCAT(last_name, ' ', first_name)
-> AS Name FROM users ORDER BY Name;
+-----+
| Name |
+-----+
| Bank, Melissa |
| Banks, Russell |
| Brautigan, Richard |
| Chabon, Michael |
| DeLillo, Don |
| Dolenz, Micky |
| Franzen, Jonathan |
| Greene, Graham |
| Harrison, George |
| Hornby, Nick |
| Isabella, Zoe |
| Jones, David |
| Lennon, John |
| McCartney, Paul |
| Morrison, Toni |
| Nesmith, Mike |
| Sedaris, David |
| Simpson, Abe |
| Simpson, Bart |
| Simpson, Homer |
| Simpson, Lisa |
| Simpson, Maggie |
| Simpson, Marge |
| Starr, Ringo |
| Tork, Peter |
| Ullman, Larry |
+-----+
26 rows in set <0.00 sec>

mysql> -

```

图5-38 通过使用别名，返回的数据将位于Name列标题下（对比图5-35）

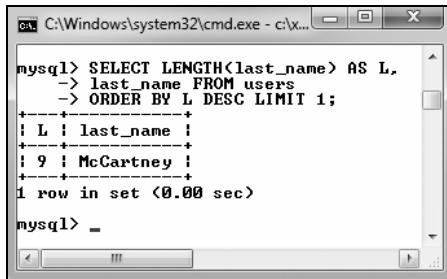
要使用别名，只需在要重命名的项目后面添加`AS aliasname`。该别名将是返回数据的新标题。为了使查询更有趣一点，在`ORDER BY`子句中也使用相同的别名。

(3) 找出最长的姓氏（参见图5-39）。

```

SELECT LENGTH(last_name) AS L,
last_name FROM users
ORDER BY L DESC LIMIT 1;

```



```

mysql> SELECT LENGTH(last_name) AS L,
-> last_name FROM users
-> ORDER BY L DESC LIMIT 1;
+---+-----+
| L | last_name |
+---+-----+
| 9 | McCartney |
+---+-----+
1 row in set <0.00 sec>

mysql> -

```

图5-39 通过使用LENGTH()函数、别名、ORDER BY子句和LIMIT子句，这个查询将返回最长的存储名字的长度和值

为了确定哪个注册用户的姓氏最长（其中具有最多的字符），可以使用`LENGTH()`函数。为了找到用户的名字，可选择姓氏值和计算的长度，给它提供一个别名`L`。然后找出最长的名字，按`L`以降序对所有结果进行排序，但是只返回第一条记录。

### ✓ 提示

- 一旦数据库中具有一些记录，类似于第(3)步中的查询（也见图5-37）就可用于帮助微调列长度。
- MySQL具有两个用于对文本执行正则表达式查找的函数：`REGEXP()`和`NOT REGEXP()`。第13章介绍了使用PHP的正则表达式。
- `CONCAT()`具有一个名为`CONCAT_WS()`的派生函数，它代表带分隔符（with separator）。其语法是`CONCAT_WS(separator, t1, t2, ...)`。分隔符将插入所列出的每个列或值之间。例如，为了将某个人的全名格式化为First<SPACE>Middle<SPACE>Last，可以编写如下代码：

```
SELECT CONCAT_WS(' ', first, middle, last) AS Name FROM tablename
```

`CONCAT_WS`具有超过`CONCAT()`的另外一个优点，这是由于它将会忽略具有`NULL`值的列。因此，该查询可能从一条记录中返回`Joe Banks`，而从另一条记录中返回`Jane Sojourner Adams`。

## 5.10.2 数字函数

除了MySQL使用的标准数学运算符（用于加法、减法、乘法和除法）之外，还有大约二十多个函数专门用于格式化数值以及对它们执行计算。表5-3列出了其中最常用的函数，我将简短地演示其中一些函数。

表5-3 数字函数

| 函 数                    | 用 法                         | 用 途                                   |
|------------------------|-----------------------------|---------------------------------------|
| <code>ABS()</code>     | <code>ABS(n)</code>         | 返回n的绝对值                               |
| <code>CEILING()</code> | <code>CEILING(n)</code>     | 基于n的值返回下一个最大的整数                       |
| <code>FLOOR()</code>   | <code>FLOOR(n)</code>       | 返回n的整数值                               |
| <code>FORMAT()</code>  | <code>FORMAT(n1, n2)</code> | 返回格式化为一个数的n1，这个数带有n2位小数，并且每3位之间插入一个逗号 |
| <code>MOD()</code>     | <code>MOD(n1, n2)</code>    | 返回n1除以n2的余数                           |
| <code>POW()</code>     | <code>POW(n1, n2)</code>    | n2是n1的幂                               |
| <code>RAND()</code>    | <code>RAND()</code>         | 返回0~1.0之间的一个随机数                       |
| <code>ROUND()</code>   | <code>ROUND(n1, n2)</code>  | 返回数n1，它被四舍五入为n2位小数                    |
| <code>SQRT()</code>    | <code>SQRT(n)</code>        | 计算n的平方根                               |

我想特别提到其中3个函数：`FORMAT()`、`ROUND()`和`RAND()`。第一个函数（理论上它不是特定于数字的）用于把任何数字转变成更传统的格式化布局。例如，如果把一辆汽车的成本存储为`20198.20`，那么`FORMAT(car_cost, 2)`将把那个数字转变成更常见的`20 198.20`。

`ROUND()`将获取一个值（假定它来自于某一列），并将其四舍五入为一个指定小数位的数字。如果没有指定小数位，就会将其四舍五入为最接近的整数。如果指定的小数位多于原始数字中存在的小数位，就将用0填充余下的空间（向小数点的右边进行填充）。

如你可能推断的，`RAND()`函数用于返回随机数（参见图5-40）。

```
SELECT RAND()
```

```

mysql> SELECT RAND();
+-----+
| RAND() |
+-----+
| 0.22895471121087177 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT RAND();
+-----+
| RAND() |
+-----+
| 0.4088071076281435 |
+-----+
1 row in set (0.00 sec)

mysql> -

```

图5-40 RAND()函数返回0~1.0之间的一个随机数

RAND()函数的另外一个好处是，它可以与查询一起用于以随机顺序返回结果。

```
SELECT * FROM tablename ORDER BY RAND()
```

### 使用数字函数

(1) 显示一个被格式化为美元金额的数字（参见图5-41）。

```
SELECT CONCAT('$', FORMAT(5639.6, 2))
AS cost;
```

```

mysql> SELECT CONCAT('$', FORMAT(5639.6, 2))
-> AS cost;
+-----+
| cost |
+-----+
| $5,639.60 |
+-----+
1 row in set (0.00 sec)

mysql> -

```

图5-41 随便使用一个示例，这个查询显示了FORMAT()函数的工作方式

正如刚才所描述的，通过CONCAT()使用FORMAT()函数，可以把任意数字转变成一种货币格式，就像你可能在Web页面中显示它一样。

(2) 从表中获取一个随机的电子邮件地址（参见图5-42）。

```
SELECT email FROM users
ORDER BY RAND() LIMIT 1;
```

```

mysql> SELECT email FROM users
-> ORDER BY RAND() LIMIT 1;
+-----+
| email |
+-----+
| graham@authors.com |
+-----+
1 row in set (0.00 sec)

mysql> -

```

图5-42 同一个查询的后续执行返回不同的随机结果

在MySQL中，这个查询将发生的情况如下：将选择所有的电子邮件地址；它们的顺序很混乱（`ORDER BY RAND()`）；然后返回第一个电子邮件地址。多次运行这个相同的查询，将会产生不同的随机结果。注意：不用指定把`RAND()`应用于哪一列。

#### ✓ 提示

- 除了这里列出的数学函数，还有几个三角、指数及其他类型的数字函数。
- `MOD()`函数的作用等同于使用百分号：

```
SELECT MOD(9,2)
SELECT 9%2
```

它会返回一个除法的余数（在上面的示例中是1）。

### 5.10.3 日期和时间函数

MySQL中的日期和时间列类型特别灵活且实用。但是，由于许多数据库用户不熟悉所有可用的日期和时间函数，所以这些选项往往未得到充分利用。无论你是想基于一个日期来执行计算，还是想从存储的值中只返回月份名称，MySQL都有针对此目的的函数。表5-4列出了其中的大多数函数。

表5-4 MySQL的一些用于处理日期和时间相关的函数。与大多数函数一样，可以对列或字面量值〔用`dt`（`datetime`的简写）表示它们〕应用这些函数

| 函 数                           | 用 法                             | 用 途                              |
|-------------------------------|---------------------------------|----------------------------------|
| <code>DATE()</code>           | <code>DATE(dt)</code>           | 返回 <code>dt</code> 的日期值          |
| <code>HOUR()</code>           | <code>HOUR(dt)</code>           | 返回 <code>dt</code> 的小时值          |
| <code>MINUTE()</code>         | <code>MINUTE(dt)</code>         | 返回 <code>dt</code> 的分钟值          |
| <code>SECOND()</code>         | <code>SECOND(dt)</code>         | 返回 <code>dt</code> 的秒值           |
| <code>DAYNAME()</code>        | <code>DAYNAME(dt)</code>        | 返回 <code>dt</code> 中天的名称         |
| <code>DAYOFMONTH()</code>     | <code>DAYOFMONTH(dt)</code>     | 返回 <code>dt</code> 中天的数字值        |
| <code>MONTHNAME()</code>      | <code>MONTHNAME(dt)</code>      | 返回 <code>dt</code> 中月份的名称        |
| <code>MONTH()</code>          | <code>MONTH(dt)</code>          | 返回 <code>dt</code> 中月份的数字值       |
| <code>YEAR()</code>           | <code>YEAR(column)</code>       | 返回 <code>dt</code> 中年份的数字值       |
| <code>CURDATE()</code>        | <code>CURDATE()</code>          | 返回当前日期                           |
| <code>CURTIME()</code>        | <code>CURTIME()</code>          | 返回当前时间                           |
| <code>NOW()</code>            | <code>NOW()</code>              | 返回当前日期和时间                        |
| <code>UNIX_TIMESTAMP()</code> | <code>UNIX_TIMESTAMP(dt)</code> | 返回从新纪元起直到当前时刻或者直到指定日期的秒数         |
| <code>UTC_TIMESTAMP()</code>  | <code>UTC_TIMESTAMP(dt)</code>  | 返回从新纪元起直到当前时刻或者直到指定日期的秒数(用UTC时间) |

MySQL支持两种存储日期和时间的数据类型（`DATETIME`和`TIMESTAMP`）、一种只存储日期的类型（`DATE`）、一种只存储时间的类型（`TIME`），以及一种只存储年份的类型（`YEAR`）。除了允许不同类型的值之外，每种数据类型还具有它自己独特的行为（同样，我建议阅读关于这个主题的MySQL手册页面，

了解所有的详细信息)。但是, MySQL非常灵活, 可以让你选择对哪些类型使用哪些函数。你可以对包含日期的任何值应用日期函数(如DATETIME、TIMESTAMP和DATE), 或者可以对包含时间的任何值应用小时函数(如DATETIME、TIMESTAMP和TIME)。MySQL将使用值中它所需要的部分, 并忽略余下的部分。不过, 不能把日期函数应用于TIME值或者把时间函数应用于DATE或YEAR值。

### 使用日期和时间函数

(1) 显示上一个用户注册的日期(参见图5-43)。

```
SELECT DATE(registration_date) AS Date
FROM users ORDER BY registration_date DESC LIMIT 1;
```

```
PHP and MySQL for Dynamic Web Site...
mysql> SELECT DATE(registration_date) AS Date
-> FROM users ORDER BY
-> registration_date DESC LIMIT 1;
+-----+
| Date |
+-----+
| 2011-03-31 |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-43 日期函数可用于从存储的值中提取信息

DATE()函数返回值的日期部分。为了查看上一个用户的注册日期, ORDER BY子句列出了最近注册的一些用户, 并限制结果只含一条记录。

(2) 显示第一个用户注册的一周中的某一天(参见图5-44)。

```
SELECT DAYNAME(registration_date) AS Weekday
FROM users ORDER BY
registration_date ASC LIMIT 1;
```

```
PHP and MySQL for Dynamic Web Site...
mysql> SELECT DAYNAME(registration_date) AS Weekday
-> FROM users ORDER BY
-> registration_date ASC LIMIT 1;
+-----+
| Weekday |
+-----+
| Thursday |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-44 这个查询返回给定日期代表的一周中某一天的名称

这类似于第(1)步中的查询, 但是以升序返回结果, 并且对registration\_date列应用DAYNAME()函数。该函数为给定的日期返回Sunday、Monday、Tuesday等。

(3) 依据MySQL显示当前日期和时间(参见图5-45)。

```
SELECT CURDATE(), CURTIME();
```

```
mysql> SELECT CURDATE(), CURTIME();
+-----+-----+
| CURDATE() | CURTIME() |
+-----+-----+
| 2011-04-01 | 14:23:14 |
+-----+-----+
1 row in set <0.00 sec>

mysql>
```

图5-45 这个不在任何特定表上运行的查询返回MySQL服务器上的当前日期和时间

为了显示MySQL目前认为的日期和时间，可以选择CURDATE()和CURTIME()函数，它们返回这些值。这是不必引用特定的表名称即可运行查询的另一个示例。

(4) 显示当前月份的最后一天（参见图5-46）。

```
SELECT LAST_DAY(CURDATE()),
MONTHNAME(CURDATE());
```

```
mysql> SELECT LAST_DAY(CURDATE()),
 -> MONTHNAME(CURDATE());
+-----+-----+
| LAST_DAY(CURDATE()) | MONTHNAME(CURDATE()) |
+-----+-----+
| 2011-04-30 | April |
+-----+-----+
1 row in set <0.00 sec>

mysql>
```

图5-46 MySQL可以利用日期和时间类型做许多事情，其中之一是确定某个月份的最后日期或者给定日期的名称值

如最后一个查询所示，CURDATE()返回服务器上的当前日期。可以把这个值用作LAST\_DAY()函数的参数，该函数返回给定日期的月份中的最后日期。MONTHNAME()函数返回当前月份的名称。

#### ✓ 提示

- ❑ MySQL的日期和时间函数返回的日期和时间对应于服务器上的日期和时间，而不是访问数据库的客户的日期和时间。
- ❑ 本节或者表5-4中没有提到ADDDATE()、SUBDATE()、ADDTIME()和SUBTIME()这几个函数。它们都可用于对日期和时间值执行算术运算。这些函数都非常有用（例如，用于查找在上一个星期内注册的每个人），但是它们的语法很麻烦。像以前一样，查看MySQL手册，了解更多信息。
- ❑ 第6章会阐述MySQL中时区的概念。
- ❑ 从MySQL 5.0.2起，服务器将阻止向日期或日期/时间列中插入无效的日期（例如，February 31, 2011）。

### 5.10.4 格式化日期和时间

你自己可能发现另外两个日期和时间函数的使用次数要多于其他所有的组合，这两个函数就是DATE\_FORMAT()和TIME\_FORMAT()。当使用其中一个函数时，会发现它们之间有一些重叠。

如果一个值同时包含日期和时间（例如，YYYY-MM-DD HH:MM:SS），那么DATE\_FORMAT()可用于格式化它们。与之相比，TIME\_FORMAT()只能格式化时间值，并且仅当存储时间值（例如，HH:MM:SS）时才可以使用它。语法如下：

```
SELECT DATE_FORMAT(datetime, formatting)
```

格式化（formatting）依赖于关键代码和百分号的组合来指示你想返回什么值。表5-5列出了可用的日期和时间格式化参数。你可以以任意组合使用它们，以及字符串面量（如标点符号），从而以更加美观的形式返回日期和时间。

表5-5 在DATE\_FORMAT()和TIME\_FORMAT()函数中使用的参数

| 名 词 | 用 法             | 示 例              |
|-----|-----------------|------------------|
| %e  | 一月中的某一天         | 1~31             |
| %d  | 一月中的某一天，用两位数字表示 | 01~31            |
| %D  | 带后缀的天           | 1st~31st         |
| %w  | 每周中的日名称         | Sunday~Saturday  |
| %a  | 简写的每周中的日名称      | Sun~Sat          |
| %c  | 月份编号            | 1~12             |
| %m  | 月份编号，用两位数字表示    | 01~12            |
| %M  | 月份名称            | January~December |
| %b  | 简写的月份名称         | Jan~Dec          |
| %Y  | 年份              | 2002             |
| %y  | 年份              | 02               |
| %l  | 小时（小写L）         | 1~12             |
| %h  | 小时，用两位数字表示      | 01~12            |
| %k  | 小时，24时制         | 0~23             |
| %H  | 小时，24时制，用两位数字表示 | 00~23            |
| %i  | 分钟              | 00~59            |
| %s  | 秒               | 00~59            |
| %r  | 时间              | 8 : 17 : 02 PM   |
| %T  | 时间，24时制         | 20 : 17 : 02     |
| %p  | 上午或下午           | AM或PM            |

假定名为the\_date的列中存储有1996-04-20 11 : 07 : 45这个日期和时间，那么常见的格式化任务和结果将是：

- 时间 ( 11 : 07 : 45 AM )  
TIME\_FORMAT(the\_date, '%r')
- 不带秒的时间 ( 11 : 07 AM )  
TIME\_FORMAT(the\_date, '%I:%i %p')
- 日期 ( April 20th, 1996 )  
DATE\_FORMAT(the\_date, '%M %D, %Y')

### 格式化日期和时间

(1) 以Month DD, YYYY-HH : MM形式返回当前日期和时间 ( 参见图5-47 )。

```
SELECT DATE_FORMAT(NOW(),'%M %e, %Y %I:%i');
```

```
mysql> SELECT DATE_FORMAT(NOW(),'%M %e, %Y %I:%i');
+-----+
| DATE_FORMAT(NOW(),'%M %e, %Y %I:%i') |
+-----+
| April 1, 2011 2:24 |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-47 格式化当前日期和时间

通过使用NOW()函数返回当前日期和时间，可以通过实践使用格式化操作来查看会返回什么结果。

(2) 使用24小时表示法显示当前时间 ( 参见图5-48 )。

```
SELECT TIME_FORMAT(CURTIME(),'%T');
```

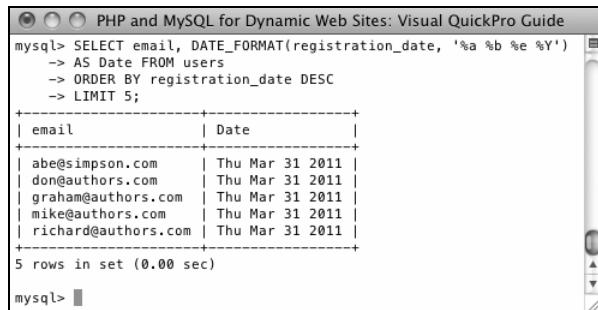
```
mysql> SELECT TIME_FORMAT(CURTIME(),'%T');
+-----+
| TIME_FORMAT(CURTIME(),'%T') |
+-----+
| 14:25:00 |
+-----+
1 row in set (0.00 sec)

mysql>
```

图5-48 以24小时格式显示的当前时间

(3) 选择按注册日期排序的每个电子邮件地址和注册日期，为最后5个注册的用户把日期格式化为Weekday ( 简写的 ) Month ( 简写的 ) Day Year ( 参见图5-49 )。

```
SELECT email, DATE_FORMAT (registration_date, '%a %b %e %Y')
AS Date FROM users
ORDER BY registration_date DESC
LIMIT 5;
```



```
mysql> SELECT email, DATE_FORMAT(registration_date, '%a %b %e %Y')
-> AS Date FROM users
-> ORDER BY registration_date DESC
-> LIMIT 5;
+-----+-----+
| email | Date |
+-----+-----+
abe@simpson.com	Thu Mar 31 2011
don@authors.com	Thu Mar 31 2011
graham@authors.com	Thu Mar 31 2011
mike@authors.com	Thu Mar 31 2011
richard@authors.com	Thu Mar 31 2011
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

图5-49 当从users表中选择记录时，DATE\_FORMAT()函数用于预先格式化注册日期

这正好是使用这些格式化函数来改变SQL查询输出结果的另一个示例。

#### ✓ 提示

- ❑ 在Web应用程序中，几乎总是应该使用MySQL函数来格式化来自数据库中的任何日期。
- ❑ 访问客户（用户机器）上的日期或时间的唯一方式是使用JavaScript。而不能用PHP或MySQL完成这个工作。

## 5.11 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 5.11.1 回顾

- ❑ 你正使用的是MySQL的哪个版本？如果你还不清楚，现在就去查一下。
- ❑ 创建新数据库的SQL命令是什么？创建新表应该使用什么命令？
- ❑ 选择要使用的数据库的SQL命令是什么？
- ❑ 什么SQL命令可以用于向表中添加新的记录。提示：有多种选择。
- ❑ 哪些类型的值在查询的时候必须用引号引起来？哪些类型的值不应该加引号？
- ❑ SELECT \* FROM tablename中的星号代表什么？如何限定查询中返回哪些列？
- ❑ NOW()函数有什么作用？
- ❑ 如何限定查询返回哪些行？
- ❑ LIKE和NOT LIKE与简单的等值比较有什么不同？哪种比较方式更快一些？LIKE和NOT LIKE的通配符都有哪些？
- ❑ 如何指定返回记录的排列顺序？默认的排序方式是什么？你该如何反转排序方式？多列排序的语法是什么？
- ❑ LIMIT子句的作用是？LIMIT x与LIMIT x,y的区别是？
- ❑ 改变表中存储值的SQL命令是什么？如果同时改变多列的值？如何限定修改操作作用于哪些行？

- 删除表中存储行的SQL命令是什么？如何限定删除操作作用于哪些行？
- 什么是SQL别名？如何创建？别名有什么作用？

### 5.11.2 实践

- 找到与你使用的MySQL版本对应的在线手册，把它存成书签。
- 按照本章中的步骤执行一些查询（说明这些步骤中的相似概念）。
- 从MySQL手册页面找出条件语句中使用的操作符。
- 从MySQL手册页面中找出一些MySQL的函数。
- 创建、填充并操作数据表。
- 使用函数和别名做更多的练习。
- 从MySQL手册页面找出各种日期和时间类型，查找ADDDATE()函数和其他相关的函数。

## 第6章

# 数据库设计

6

### 本章内容

- 规范化
- 创建索引
- 使用不同的表类型
- 语言和MySQL
- 时区和MySQL
- 外键约束
- 回顾和实践

## 现

在你已经对数据库、SQL和MySQL有了一些基本的了解，本章将更加深入地讨论这些内容。从标题就可以看出，本章的重点是真实世界中的数据库设计。和第4章一样，这里的大部分工作都需要你亲自动手写写划划，并认真思考你的应用程序都需要做些什么。

本章先介绍数据库规范化：设计过程中一项至关重要的方法。然后，本章将解释MySQL的相关设计概念：索引、表类型、语言支持、时间处理和外键约束。

在本章的末尾，我会探讨数据库设计涉及的步骤，介绍如何充分利用MySQL，并设计一个多表数据库。下一章，会使用本章示例中的这几个数据库学习更高级的SQL和MySQL知识。

## 6.1 规范化

在使用关系型数据库管理系统（如MySQL）时，创建和使用数据库的第一步就是确立数据库的结构（也叫数据库架构）。数据库设计（也叫数据建模），是确保信息长期管理能够成功的关键。通过一种叫做规范化的过程，可以仔细地消除冗余和其他会破坏数据库完整性的问题。

接下来要阐述的技术将有助于确保数据库的可行性、可用性和可靠性。本章将要讨论的主要示例（用户可以发布消息的论坛）将在第17章中专门讲述，但是，规范化的原则适用于任何可能要创建的数据库。（即使规范化之前尚未讨论，但此前的两章中创建和使用的sitename示例也已被恰当的规范化了。）

规范化是由IBM研究员E. F. Codd于20世纪70年代早期开发的（他还发明了关系型数据库）。关系型数据库只是一个以特殊方式组织的数据集合而已，Codd博士创建了一系列称为范式的规则，用于帮助定义数据组织形式。本章讨论了前三种范式，完全可以满足大部分的数据库设计。

在开始规范化数据库之前，你必须定义要开发的应用程序的作用。不论是与客户充分讨论还是自

已搞清楚，理解如何访问信息对建模有决定性影响。因此，这个过程需要的是纸和笔而不是MySQL软件（尽管数据库设计适应于任何关系数据库，而不只是MySQL）。

在这个例子中，我想创建一个用户可以发帖和其他用户可以回帖的留言板。用户需要注册，然后使用电子邮箱/密码的组合登录以发布信息。我还希望会有不同主题的讨论区。我在表6-1中列出了示范数据行。数据库本身会命名为forum。

#### ✓ 提示

- 确定在数据库中存储什么信息的最好的方法是，仔细考虑一下将向数据库询问的问题，以及答案中会包含哪些数据。
- 存储的内容尽可能多于你可能需要的。忽略多余的数据非常容易但是捏造不存在的数据是不可能的。
- 如果只关注细节，规范化会非常难以学习。每种范式都定义得十分晦涩；即使以非常通俗的方式来讲，仍然会很难懂。所以我建议你在学习下面的内容时要着眼全局。一旦你完成了规范化并看到最终结果，整个过程应该就足够清晰了。

### 6.1.1 键

正如第4章简要提及的那样，键是规范化数据库不可或缺的一部分。键有两类：主键和外键。主键是一种唯一的标识符，必须遵循一定的规则。它们必须：

- 始终有一个值（不能为NULL）；
- 具有一个保持不变的值（永不改变）；
- 表中的每一条记录都有唯一的值。

在现实生活中，美国社会保险号就是一个很好的例子，每个人都只有一个唯一的社会保险号码，这个号码永远不会改变。社会保险号码用于识别人们身份，与之相似的是，你往往你会发现，为每个表创建主键是最佳的设计实践。

第二种键是外键。外键是表A的主键在表B中的代表。如果你的一个影院数据库中有movies表和directors表，那么directors表的主键会作为外键与电影相关联。随着规范化过程的继续进行，你将更好地看到这是如何工作的。

论坛数据库只是一个简单的表（参见表6-1），但是在开始规范化过程之前，至少确定一个主键（在后面的步骤中将引入外键）。

表6-1 简单论坛数据

| 项 目             | 示 例                          |
|-----------------|------------------------------|
| username        | troutster                    |
| password        | mypass                       |
| actual name     | Larry Ullman                 |
| user email      | email@example.com            |
| forum           | MySQL                        |
| message subject | Question about normalization |
| message body    | I have a question about...   |
| message date    | November 2, 2011 12:20 AM    |

### 指定主键

(1) 寻找所有符合主键的三项测试的字段。

在示例中(参见表6-1),没有完全符合所有主键标准的列。用户名和电子邮箱地址对每个论坛用户是唯一的,但是对数据库中的每一条记录却不是唯一的(因为同一个用户可能会发布多条信息)。相同的话题也有可能被使用多次。信息的主题很可能对每一条信息来说是唯一的,但是可能会发生改变(如果被编辑),这违反了主键的一项原则。

(2) 如果没有逻辑主键存在,那就创造一个(参见表6-2)。

表6-2 简单论坛数据

| 项 目             | 示 例                          |
|-----------------|------------------------------|
| message ID      | 325                          |
| username        | troutster                    |
| password        | mypass                       |
| actual name     | Larry Ullman                 |
| user email      | email@example.com            |
| forum           | MySQL                        |
| message subject | Question about normalization |
| message body    | I have a question about...   |
| message date    | November 2, 2011 12:20 AM    |

通常情况下,你需要创建一个主键,因为没有更好的解决方案。在这个例子中,创建了一个message ID。如果创建的主键没有任何其他意义和目的,就称其为代理主键。

#### ✓ 提示

- 一般来说,使用表名的一部分(例如, message)和id这个词来命名主键。有些数据库开发者也喜欢为名字加上缩写词pk。另一些开发者仅仅使用id。
- MySQL只允许每个表有一个主键,虽然你可以让主键基于多列(这表示这些列的组合必须唯一且永远不变)。
- 理想情况下,主键应该是整数,使MySQL性能更佳。

## 6.1.2 关系

数据库关系是指一个表中的数据是如何与另外一个表中的数据关联的。任何两个表之间无外乎有三种类型的关系:一对一、一对多或多对多。(数据库中的两个表也有可能没有任何关系。)

如果A表中有且仅有一条的记录对应表B中的唯一一条记录,那这种关系即为一对一的关系。例如,每位美国公民仅有一个社会安全号码,并且每个社会安全号码仅对应一位美国公民;没有公民有两个社会安全号码,并且没有社会安全号码能对应两名公民。

如果表A中的一条记录对应表B中的多条记录,那这种关系即为一对多关系。男性和女性的术语会对应很多人,但是每一个人只能对应男性或女性(理论上说)。在规范化的数据库中一对多是表之间最常见的关系(参见图6-1)。

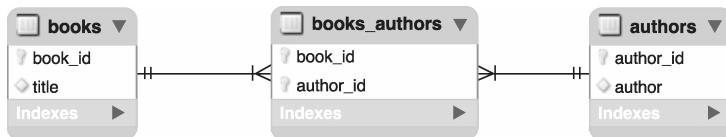


图6-1 两个表之间的一个多对多关系最好使用中间表，表示成两个一对多关系

最后，如果表A中的多条记录对应表B的多条记录那这就是多对多关系。一本书可以是多位作者写的，并且一个作者可以写多本书。虽然多对多关系在现实世界中很常见，你应该在设计中避免多对多关系，因为它们会带来数据冗余和完整性问题。尽量在设计数据库时使用中间表，将一个多对多关系分解成两个一对多关系。

如前所述，关系和键一起工作时，一个表中的键通常会关联另外一个表中的键。

6

✓ 提示

- 数据库建模使用特定惯例来表示数据库的结构，本章，我将通过一系列图片来遵循这一惯例。三种类型的关系符号参见图6-2。
- 数据库设计过程的结果就是ERD（实体-关系图）或ERM（实体-关系模型）。图像化的数据库表示方法使用图形来表示表和列，使用图6-2中的符号来表示关系。
- 有很多程序可以协助创建数据库架构，包括MySQL Workbench<http://wb.mysql.com>。本章的很多图片是来自MySQL Workbench。
- RDBMS中的术语“关系”实际上源于表，在技术上被称为关系（relations）。

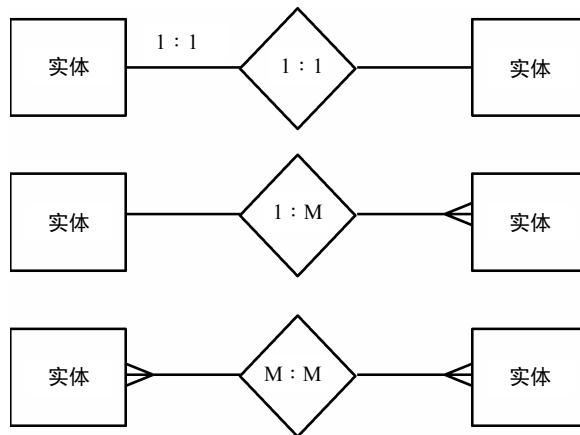


图6-2 这些符号（或其简写）常用于表示数据库建模模式中的联系

### 6.1.3 第一范式

如前所述，规范化数据库是根据几条被称为范式的规则来调整数据库结构的过程。你的数据库必须完全遵循每条规则，并且按顺序遵循范式。

要符合第一范式（1NF），数据库中的每一个表必须具有以下两个性质：

- 每一列必须仅包含一个值（有时候这被描述为原子性或不可分割性）；
- 所有表都不能具有相关数据的重复列。

如果表中用一个字段存储人员的完整地址（街道、城市、州、邮政编码、国家），那么这个表就不符合第一范式，因为一列中包含了多个值，违反了上面的第一个特性。再比如，包含演员1、演员2、演员3等列的电影表也不符合第一范式，因为它违反了第二个特性，这些列包含的信息类型完全相同。

在开始规范化过程之前，检查现有的结构（表6-2）是否符合第一范式。任何非原子性的列都应该打散成多列。如果一个表有重复的相似列，就把这些列转化成只包含这些列的单独表。

### 使数据满足第一范式

(1) 找出所有包含多条信息的字段。

查看表6-2，actual name字段不符合第一范式，在一个字段中同时包含了名和姓。

message date字段包含日、月和年及时间，但是没有必要细分到那个层次。我们在上一章的末尾展示过，MySQL中的DATETIME类型就能很好地处理日期和时间。

另外一个问题，一个表是否使用了一列存储多个电话号码（移动电话、家庭电话、工作电话），或用一列存储一个人的兴趣爱好（烹饪、跳舞、滑雪，等等）。

(2) 请将第1步中找出的字段打散成多个不同的字段（参见表6-3）。

表6-3 论坛数据库，以原子的方式存储数据

| 项 目             | 示 例                          |
|-----------------|------------------------------|
| message ID      | 325                          |
| username        | troutster                    |
| password        | mypass                       |
| first name      | Larry                        |
| last name       | Ullman                       |
| user email      | email@example.com            |
| forum           | MySQL                        |
| message subject | Question about normalization |
| message body    | I have a question about...   |
| message date    | November 2, 2011 12:20 AM    |

要解决当前这个例子中的问题，需要创建独立的first name和last name的字段，使每个字段仅包含一个值。

(3) 将所有重复列的组转换成它们自己的表。

目前论坛数据库没有这个问题，为了演示什么是违规的情况，可参见表6-4。重复的列（多个演员字段）带来了两个问题。首先，以这种方式存储数据出现的不可避免的问题是，每部电影的演员数都会被限制为一个固定值。即使你添加100个演员列，仍然有限制（限制是100）。其次，任何没有达到最大演员数的记录都会在多余的列上存储NULL值。应该尽量避免NULL值出现在数据库模式中。另外值得注意的是，演员和导演列不是原子性的。

表6-4 movies表

| 列             | 值               |
|---------------|-----------------|
| movie ID      | 976             |
| movie title   | Casablanca      |
| year released | 1943            |
| director      | Michael Curtiz  |
| actor 1       | Humphrey Bogart |
| actor 2       | Ingrid Bergman  |
| actor 3       | Peter Lorre     |

要解决movies表中的问题，必须创建第二个表（参见表6-5）。这个表使用一行来代表电影中的一位演员，解决上一段中涉及的问题。演员的名字也被打散成原子性的了。注意，新表中也要添加一个主键列。每个表都要有一个主键是第一范式中隐含的思想。

表6-5 movies-actors表

| ID | 电 影                | 演 员 名    | 演 员 姓   |
|----|--------------------|----------|---------|
| 1  | Casablanca         | Humphrey | Bogart  |
| 2  | Casablanca         | Ingrid   | Bergman |
| 3  | Casablanca         | Peter    | Lorre   |
| 4  | The Maltese Falcon | Humphrey | Bogart  |
| 5  | The Maltese Falcon | Peter    | Lorre   |

(4) 仔细检查第(2)步和第(3)步中创建的所有新列和新表，都要通过第一范式测试。

#### ✓提示

- 理解第一范式最简单的方法是，它就是横向分析表的规则：检查单独一行的所有列，确保唯一性并避免重复出现类似的数据。
- 各种资料会以各种不同的方式来描述范式，很可能使用很多技术术语。但关键是要理解规范化的本质及最终结果，而不是各种规则的技术性措辞。

### 6.1.4 第二范式

如果数据库要符合第二范式（2NF），数据库首先必须符合第一范式（必须按顺序规范化）。然后，表的每一个不为键（例如，外键）的列必须依赖主键。当某一列在多行中具有相同的非键值时，通常可以确定它违反了这个规则。这些值应该存储在它们自己的表中，并且通过键关联回原来的表。

回到影院的例子，movies表（表6-4）中Martin Scorsese导演出现了20多次。这违反了第二范式，因为存储导演名字的列不是键，并且并不依赖主键（电影ID）。解决的办法就是再创建一个独立的表，存储导演的信息并为每位导演分配一个主键。要将导演绑定到电影，导演的主键需要是movies表的外键。

查看表6-5（电影演员），电影名称和演员名字都违反了第二范式的规则（它们不是键，并且不依赖表的主键）。最后，影院数据库需要4个表（参见图6-3）。每个导演名字，电影名称和演员的名字将仅存储一次，并且表中每个非键的列都依赖于表的主键。规范化其实可以理解为创建更多表的过程，

直到所有潜在冗余都被消除为止。

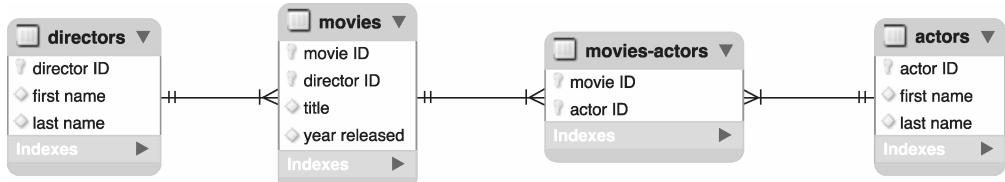


图6-3 为了使影院数据库符合第二范式（给出要表示的信息），需要创建4个表。

在movies表中通过director ID键表示导演；在movies-actors表中通过movie ID键表示电影；在movies-actors表中通过actor ID键表示演员

### 使数据满足第二范式

(1) 确定所有不依赖于表主键的非键列。

查看表6-3，用户名、名、姓、电子邮箱和讨论值都是非键（message ID目前是唯一的键列），并且都不依赖message ID。相反，信息的标题、正文和日期也是非键，但它们依赖message ID。

(2) 创建相应的新表（参见图6-4）。



图6-4 为了使论坛数据库符合第二范式，需要创建3个表

论坛数据库最合乎逻辑的修改就是建立3个表：users表、forums表和messages表。

在数据库的可视化表示中，为每个表创建一个框，表名作为标题，下面是所有的列（也叫属性）。

(3) 分配或创建新的主键（参见图6-5）。



图6-5 每个表都需要它自己的主键

使用本章前面描述的技术，让每个新表都有一个主键。这里，我在users表中添加了一个user ID字段，在forums表中添加了一个forums ID的字段。这两个都是代理主键。因为users表中的username字段

和forums表中的name字段在每一条记录中都必须是唯一的，并且总是有一个值，你可以让它们作为各自表的主键。然而，这就意味着这些值将永远不能改变（主键规则中的一条），但使用基于文本的键代替数值型的键将会使数据库变慢。

(4) 创建必要的外键并指明关系（参见图6-6）。

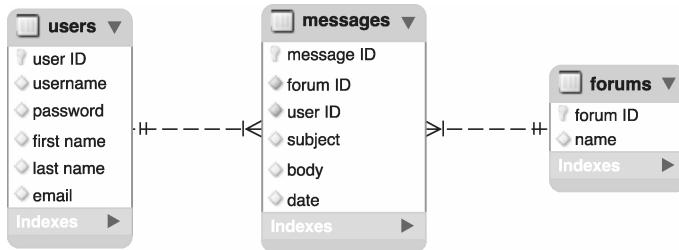


图6-6 为了关联3个表，添加了两个外键到messages表中，每个键都表示另外两个表之一

要实现第二范式规则的最后一步是结合外键来链接关联表。记住，一个表的主键通常是另外一个表的外键。

在这个例子中，users表中的user ID链接到messages表中的user ID列。因此，users表与messages表之间是一对多的关系（因为每个用户可以发布多条信息，但是每条信息只能由一个用户发布）。

同样，两个forum ID列也被关联，在messages表和forums表之间创建了一个一对多的关系（每条信息只能在一个forums表中，但每个forums表可以有多条信息）。

users表和forums表之间没有直接的关系。

#### ✓ 提示

- 测试第二范式的另外一种方法就是查看表之间的关系。理想的情况是创建一对一或一对多的情况。有多对多关系的表则需要重新组织。
- 回顾图6-3，movies-actors表是一个中介表，它将电影和演员之间多对多的关系变成了两个一对多的关系。当一个表所有的列都是键的时候，这个表就是一个中介表。实际上，在那个表中主键可以是movie ID和actor ID的组合。
- 经过恰当规范化的数据库永远不会有重复的行出现在同一个表中（两个或更多的行的所有非主键列相同）。
- 为了简化规范化的过程，记住第一范式是一个横向检查表的问题，而第二范式是一个纵向的分析（搜索多行上重复的值）。

### 6.1.5 第三范式

如果数据库符合第二范式，并且每个非键列互相之间是独立的，那它就符合第三范式（3NF）。如果你正确按照规范化的过程，就可能没有第三范式的问题。如果改变一列中的值需要改变另外一列的值，那就是违反了第三范式。迄今为止，论坛这个例子还没有和第三范式发生关系，但我们可以通过假设一种情况来介绍这个范式的作用。

拿一个关于书籍的数据库作为例子。在应用前面两个范式之后，你最终得到一个图书表和作者表，还有第三个作为图书和作者之间的中介表，作为一个多对多的关系存在。如果图书表还有出版商的名字和地址，那这个表就违反了第三范式（参见图6-7）。出版商的地址与图书无关，而是与出版商本身有关系。换言之，图书表有一个列依赖一个非键列（出版商的名字）。



图6-7 目前的图书数据库设计不符合第三范式

如前所述，论坛例子没什么问题，但我还是要谈谈如何才能满足第三范式，修复刚才所说的那个书籍数据库。

### 使数据满足第三范式

(1) 确认所有表中的所有字段是相互依赖的。

如前所述，要找的是相互之间依赖的列，而不是依赖整条记录的列。论坛数据库中没有这个问题。查看messages表，每个标题都将对应一个message ID，每个消息体都将对应那个message ID，其他的4个字段也是这样。

在图书示例中，有问题的都是books表中属于出版商的那些字段。

(2) 创建相应的新表。

如果在第(1)步中你发现了任何有问题的列，像图书例子中的地址1、地址2、城市名、州名和邮编，就需要创建一个单独的出版商表。（如果是跨国出版商，那地址将会变得更加复杂。）

(3) 分配或创建新的主键。

每个表都必须有一个主键，因此为新表添加publisher ID。

(4) 创建必要的外键以链接所有的关系（参见图6-8）。

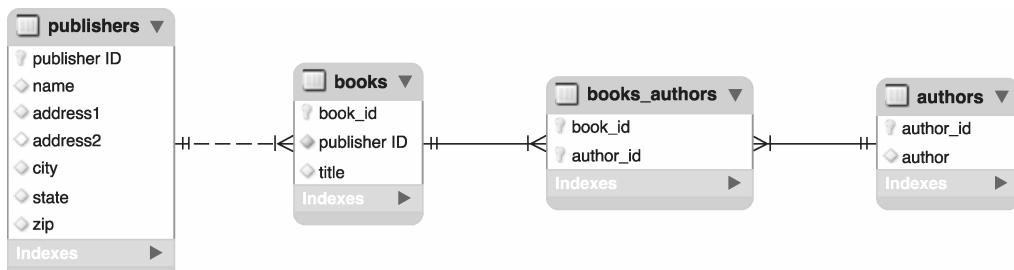


图6-8 假想的图书数据库的最简版本，新建一个数据表存储出版商信息

最终，添加publisher ID到books表。这能有效地将每本书关联到它的出版商上。

### ✓ 提示

□ 虽然数据库的规范化有着一套固定的规则，但不同的人对待同一数据库的规范方法还是会略有不同。数据库设计允许有个人的喜好和诠释。但最重要的要求就是数据库不要明显地违反范式。任何违反范式的设计都会导致将来会出现一些问题。

### 拒 范

尽量符合三个范式会有助于确保数据库的可靠性和可行性，但你可能不会对所有要用到的数据库都进行完全规范化处理。所以在尝试非常规做法之前，先要弄清楚这样做可能会有的长期破坏性后果。

之所以会采用非规范化方式处理，主要就是为了方便与高效。更少的表更易于操作和理解。另外由于规范化的数据库十分复杂，所以更新、获取数据和修改变得极易更慢。总之，规范化就是要在数据完整性/可扩展性与简单/快速之间进行权衡。另外，虽然有很多方法都能改善数据库性能，但却很少有方法能修复因糟糕设计而损坏的数据。

本章包含了一个非规范化的例子：一条消息的发布日期和时间存储在一个字段中。如前所述，因为MySQL对日期的支持非常好，这么做没有任何问题。另外一个非规范化的情况是，存储个人性别及其他信息的表。如果仅以M/F或男/女来存储（而不是关联到性别表），将会有很多的重复值。但是这种情况也不会有太大的问题，因为这些标记都是固定值，不会随着时间改变（例如，不太可能有第三种选择被创造出来或“女性”被改名，从而迫使表中一大半的记录被更新）。

实践和经验将教会你如何以最佳方式为数据库建模，尽可能尝试遵守范式，特别是在只掌握概念的阶段。

### 6.1.6 审查设计

在完成了规范化的过程之后，最好重新审查一次设计。确认数据库存储了可能用到的全部数据。由于规范化的原因，创建新表也就意味着会记录更多的信息。例如，电影院数据库原来的重点是电影，现在有了独立的演员和导演表，这些人的相关信息就会体现在相应的表中。

考虑到这一点，虽然有很多列需要被添加到forum数据库，特别是对于用户，需要添加一些字段到messages表。因为一条信息可能是另外一条的回复，这需要一些指明这种关系的方法。一个解决方案就是为消息添加parent\_id列（参见图6-9）。如果一条信息是回复，它的parent\_id将会是原消息的message\_id（所以parent\_id相当于一个关联到本表的外键）。如果一条信息的parent\_id是0，那它就是一个新的帖子，而不是回复（参见图6-10）。

在完成任何修改之后，你必须重新浏览一遍范式来确定数据库依然是规范的。最后，按照第4章的每一步选择列的类型和名字（参见图6-11）。注意，每个整数列都是无符号整型（UNSIGNED），3个主键列依然设计为自增长（AUTO\_INCREMENT），并且每列都设置为不可为空（NOT NULL）。

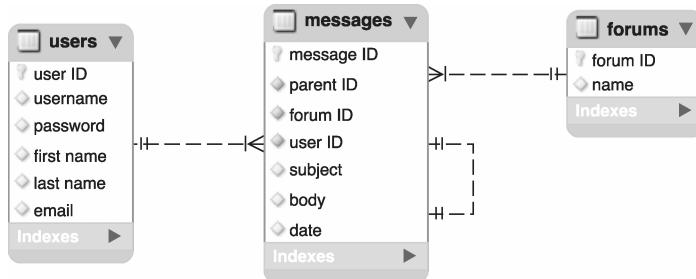


图6-9 为了反映出消息的层次，向messages表中添加parent\_id列

| message_id | parent_id |
|------------|-----------|
| 3          | 0         |
| 4          | 0         |
| 18         | 3         |
| 19         | 4         |
| 20         | 18        |

图6-10 parent\_id字段追踪消息线索的方式

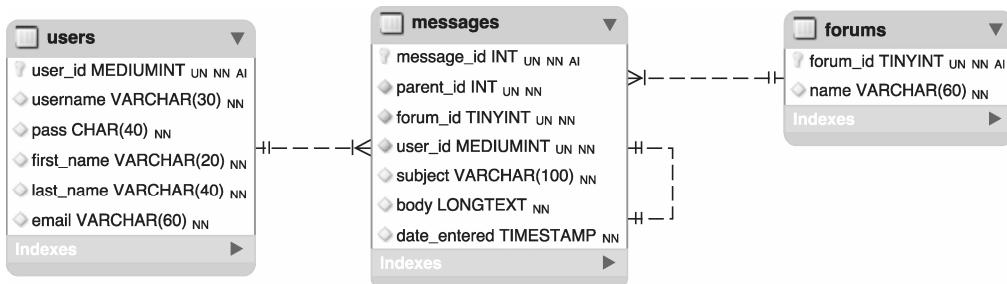


图6-11 论坛数据库最终的实体关系图

数据库架构开发完成后，就可以在MySQL中使用第5章介绍的命令创建数据库了。你可以继续学一些内容之后再创建这个数据库。

✓ 提示

- 如果有一个主键-外键链接（像forums表中的forum\_id和消息中的forum\_id），这两列应该是相同类型（在此处为TINYINT UNSIGNED NOT NULL）。

## 6.2 创建索引

索引是一种数据库用于提高SELECT查询效率的特殊体制。索引可以被放置到一列或多列，可以设置为任何数据类型，它能告诉MySQL要特别注意那些值。

虽然表的最大索引数可能会有各种限制，但MySQL至少能保证为每个表创建16个索引，每个索引可以最多包含15列。虽然多列索引的需求可能不太明显，但它会在频繁搜索相同的列组合时派上用场（例如，名和姓、市和州等）。

虽然索引是任何表都不可或缺的一部分，但并不是所有的内容都需要索引。虽然索引提高了数据库的读取速度，但却减慢了对更改数据的查询（因为这些更改数据需要记录在索引中）。

索引适合用于以下类型的列：

- 查询的WHERE部分中频繁使用的列；
- 在查询的ORDER BY部分中频繁使用的列；
- 经常被用于JOIN连接点的列（联合会在第7章讨论）

通常，不应为下面这些列创建索引：

- 允许为空的列；
- 字段值的范围很有限（例如仅仅是Y/N或1/0）。

MySQL有4种类型的索引：INDEX（标准索引）、UNIQUE（每一行对于索引列具有唯一的值）、FULLTEXT（用于执行FULLTEXT查找，参见第7章）、PRIMARY KEY（就是一个特殊的UNIQUE索引，你已经使用过了）。注意，一列只能有一个索引，因此要选择最合适索引类型。

了解上述内容后，让我们通过确定合适的索引来继续设计论坛数据库。稍后会在创建数据库表的时定义索引。要在创建表时设立索引，就需要在CREATE TABLE命令中添加以下语句：

*INDEX\_TYPE index\_name (columns)*

索引名是可选的。如果没有提供名称，索引会使用它应用的列的名字。当索引多列时，以逗号将它们分隔，并按照从最重要到最不重要的顺序排列：

INDEX full\_name (last\_name, first\_name)

你已经在第5章见过创建索引的语法了。这个命令创建了一个user\_id为主键索引的表：

```
CREATE TABLE users (
 user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
 first_name VARCHAR(20) NOT NULL,
 last_name VARCHAR(40) NOT NULL,
 email VARCHAR(40) NOT NULL,
 pass CHAR(40) NOT NULL,
 registration_date DATETIME NOT NULL,
 PRIMARY KEY (user_id)
);
```

关于索引你需要了解的最后一点是，多列索引的影响。如果你为列1、列2和列3添加了一个索引（以此顺序），这实际上创建了一个同时使用列1、列2和列3的索引，或者说所有这三列同时都使用的一个索引，而它并未提供仅索引列2和列3或其他两列在一起时的索引。

### 创建索引

(1) 为所有的主键添加一个PRIMARY KEY索引。

每个表都必须有一个主键，因而会有一个PRIMARY KEY索引。在论坛数据库，要被作为主键索引的指定列是：forums.forum\_id、messages.message\_id和users.user\_id。（语法table\_name.column\_name是引用指定表的指定列的方法。）

(2) 为表中值不能重复的列添加UNIQUE索引。

论坛数据库中有三列的每一行值应该始终唯一，否则会有问题：forums.name、users.username和users.email。

(3) 添加FULLTEXT索引。

FULLTEXT索引和FULLTEXT搜索会在下一章讨论，这里不会讨论这个主题，但是你可能会发现，这个数据库中使用了一个FULLTEXT索引。

(4) 为经常在WHERE子句中用到的列添加标准索引。

需要具备一些经验才能预测哪些列会经常用于WHERE子句中（因而需要索引）。在论坛数据库中，有一个常见的WHERE子句：当用户登录时，他们会提供电子邮箱地址和密码来登录。下列查询可确认用户提供信息是否正确：

```
SELECT * FROM users WHERE pass=SHA1('provided_password') AND email='provided_email_address'
```

从这个查询中，可以推断出索引电子邮箱地址和密码的组合是有益的。

(5) 为经常在ORDER BY子句中用到的列添加标准索引。

同样，在数据库设计时这些列会很引人注目。在论坛例子中，可能会在ORDER BY子句中用到而且还没有被索引的列就剩下messages.date\_entered了。这列经常被用于ORDER BY子句中，网站会默认按照发布的时间顺序显示所有信息。

(6) 为经常在JOIN中用到的列添加标准索引。

你现在可能对JOIN还不了解，这个主题会在第7章详细讲解，但是最明显的候选者就是那些外键列。记住，表B的外键涉及表A的主键。当从数据库中选择数据时，会基于此关系编写一个JOIN语句。外键必须经过索引（主键已被索引）那个JOIN才会是高效的。在论坛例子中，信息表中的三个外键字段应当被索引：forum\_id、parent\_id和user\_id。

表6-6列出了通过这些步骤识别的所有索引。

表6-6 forum数据库索引

| 列名         | 表        | 索引类型    |
|------------|----------|---------|
| forum_id   | forums   | PRIMARY |
| name       | forums   | UNIQUE  |
| message_id | messages | PRIMARY |
| forum_id   | messages | INDEX   |
| parent_id  | messages | INDEX   |

(续)

| 列名           | 表        | 索引类型    |
|--------------|----------|---------|
| user_id      | messages | INDEX   |
| date_entered | messages | INDEX   |
| user_id      | users    | PRIMARY |
| username     | users    | UNIQUE  |
| pass/email   | users    | INDEX   |
| email        | users    | UNIQUE  |

**✓ 提示**

- 索引可以在已经有了被填充的表之后创建。然而，如果尝试为一个有重复值的列添加UNIQUE索引，那就会出错并且索引不会被创建。
- MySQL使用术语KEY作为索引的同义词：

```
KEY full_name(last_name, first_name)
```

- 你可以限制一个索引的长度为指定字符数，例如前10个：

```
INDEX index_name(column_name(10))
```

如果前几个字母在ORDER BY子句中特别有用，可以采取这种方法。

6

## 6.3 使用不同的表类型

MySQL有一个在其他的数据库软件中不常见的特性，即为表使用不同类型的能力（表的类型也称为它的存储引擎）。每种表类型支持不同的功能，有它自己的限制（按照它能存储的数据量），甚至在某些特定情况下表现更好或更糟糕。然而，与无论与何种类型的表交互（在执行查询方面），其方式都是一致的。

历史上，最重要的表类型是MyISAM，直到MySQL的5.5.5版本，MyISAM是所有操作系统上的默认的表类型（Windows系统，MySQL早期版本使用另外一种表类型作为默认类型）。MyISAM表适用于大部分应用程序，处理SELECT和INSERT都非常快。但MyISAM存储引擎不能处理事务，这是它最大的不足（第7章涵盖了事务的内容）。由于缺少行级的锁定（整个表需要被锁定），MyISAM表很容易损坏并且在发生崩溃时数据容易丢失。

MySQL的5.5.5版本在所有操作系统上采用了新的默认存储引擎——InnoDB。InnoDB表可以使用事务，并且在处理UPDATE的时候表现良好。InnoDB表同样支持外键约束（本章的最后介绍）和行级锁定。但是InnoDB的存储引擎通常比MyISAM的慢并且需要更多的服务器磁盘空间。同时，InnoDB表不支持FULLTEXT索引（参见第7章）。

要在定义表时指定存储引擎，需要在声明的末尾添加一个子句：

```
CREATE TABLE tablename (
 column1name COLUMNTYPE,
 column2name COLUMNTYPE...
) ENGINE = type
```

如果创建表的时候没有指定存储引擎，MySQL将会使用服务器默认的类型。

MySQL的这个功能有非常重要的意义，因为你可以在一个数据库中混合使用不同的表类型。这样你就可以为每个表定制最佳的功能和性能。接下来，继续设计论坛数据库，下一步是确定每个表使用的存储引擎。

### 设定表的类型

(1) 查找你的MySQL服务器上可用的表类型(参见图6-12)。

```
SHOW ENGINES;
```

SHOW ENGINES命令不但会显示可用的存储引擎，而且还会显示默认的存储引擎。了解这些信息有助于选择数据库表的类型。

| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
|--------------------|---------|----------------------------------------------------------------|--------------|------|------------|
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based stored in memory, useful for temporary tables       | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |

图6-12 执行SHOW ENGINES命令(在MySQL客户端或phpMyAdmin中)查看MySQL支持的表类型

(2) 如果表需要FULLTEXT索引，就要选用MyISAM类型。

FULLTEXT索引和FULLTEXT搜索会在下一章讨论，这里要说明的是论坛例子中的messages表需要FULLTEXT索引。因而，这个表必须是MyISAM类型的。

(3) 如果某个表需要支持事务，要设置为InnoDB类型。

事务会在第7章讨论，而现在需要决定存储引擎。论坛数据库中的forums表和users表都不需要事务。

(4) 如果上面的项都不适用，使用默认的存储引擎。

表6-7列出了论坛数据库中的表使用的存储引擎。

表6-7 论坛数据库中的表类型

| 表        | 类 型    |
|----------|--------|
| forums   | InnoDB |
| messages | MyISAM |
| users    | InnoDB |

### ✓ 提示

- ❑ MySQL还有几种其他的表类型，但是MyISAM和InnoDB是目前最重要的两种。MEMORY类型在内存中创建表，因此它的速度非常快，但却无法实现持久化存储。
- ❑ 在编写本书的时候，MySQL的文档指明InnoDB表的速度要比MyISAM表更快，但我对此观点持保留态度。

## 6.4 语言和 MySQL

我们在第1章曾简要介绍过编码的概念。HTML页或者PHP脚本可以指定它的编码，决定能够支持哪些字符，乃至语言。这同样适用于MySQL数据库：通过设置数据库的编码，你可以指定数据库中可以存储哪些字符。要查看你使用的MySQL版本所支持的编码，运行SHOW CHARACTER SET命令（参见图6-13）。注意，在MySQL中，字符集也就意味着编码（在本节内我会遵循MySQL的这个惯例）。

| Charset  | Description                 | Default collation   | Maxlen |
|----------|-----------------------------|---------------------|--------|
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     | 2      |
| dec8     | DEC West European           | dec8_swedish_ci     | 1      |
| cp850    | DOS West European           | cp850_general_ci    | 1      |
| hp8      | HP West European            | hp8_english_ci      | 1      |
| ko18r    | KO18-R Relcom Russian       | ko18r_general_ci    | 1      |
| latin1   | cp1252 West European        | latin1_swedish_ci   | 1      |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   | 1      |
| swe7     | 7bit Swedish                | swe7_swedish_ci     | 1      |
| ascii    | US ASCII                    | ascii_general_ci    | 1      |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci    | 3      |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci    | 2      |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   | 1      |
| tis620   | TIS620 Thai                 | tis620_thai_ci      | 2      |
| euckr    | EUC-KR Korean               | euckr_korean_ci     | 2      |
| ko18u    | KO18-U Ukrainian            | ko18u_general_ci    | 1      |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci   | 2      |
| greek    | ISO 8859-7 Greek            | greek_general_ci    | 1      |
| cp1250   | Windows Central European    | cp1250_general_ci   | 1      |
| gbk      | GBK Simplified Chinese      | gbk_chinese_ci      | 2      |
| latin5   | ISO 8859-9 Turkish          | latin5_turkish_ci   | 1      |
| armsci8  | ARMSCII-8 Armenian          | armsci8_general_ci  | 1      |
| utf8     | UTF-8 Unicode               | utf8_general_ci     | 3      |
| ucs2     | UCS-2 Unicode               | ucs2_general_ci     | 2      |
| cp866    | DOS Russian                 | cp866_general_ci    | 1      |
| keybcs2  | DOS Kamenicky Czech-Slovak  | keybcs2_general_ci  | 1      |
| macce    | Mac Central European        | macce_general_ci    | 1      |
| macroman | Mac West European           | macroman_general_ci | 1      |
| cp852    | DOS Central European        | cp852_general_ci    | 1      |
| latin7   | ISO 8859-13 Baltic          | latin7_general_ci   | 1      |
| utf8mb4  | UTF-8 Unicode               | utf8mb4_general_ci  | 4      |
| cp1251   | Windows Cyrillic            | cp1251_general_ci   | 1      |
| utf16    | UTF-16 Unicode              | utf16_general_ci    | 4      |
| cp1256   | Windows Arabic              | cp1256_general_ci   | 1      |
| cp1257   | Windows Baltic              | cp1257_general_ci   | 1      |
| utf32    | UTF-32 Unicode              | utf32_general_ci    | 4      |
| binary   | Binary pseudo charset       | binary              | 1      |
| geostd8  | GEOSTD8 Georgian            | geostd8_general_ci  | 1      |
| cp932    | SJIS for Windows Japanese   | cp932_japanese_ci   | 2      |
| eucjps   | UJIS for Windows Japanese   | eucjps_japanese_ci  | 3      |

图6-13 列出MySQL支持的字符集

MySQL中的每种编码都有一种或多种校对规则（collation）。校对规则是在字符集内用于比较字符的一套规则。它与按照字母排序类似，但是还包括数字、空格和其他字符。校对规则绑定到使用的字符集上的，既反映了语言出现的字符种类，也反映了人们通常使用语言的文化习惯。例如，英中文字符的排序跟传统的西班牙语或阿拉伯语是不同的。其他事项如：字符的大写或小写是否有影响（例如，比较是否区分大小写）、重音字符是如何排序的、空格计数还是忽略。

要查看MySQL可用的校对规则，运行以下查询（参见图6-14），使用上次查询结果中适当的值来代替charset（参见图6-13）：

```
SHOW COLLATION LIKE 'charset%'
```

这个查询的结果也会指出查询的字符集的默认校对规则。校对规则名字的后缀ci表示不区分大小写，cs后缀表示区分大小写，bin后缀表示二元（binary）校对规则。

| Collation             | Charset | Id  | Default | Compiled | Sortlen |
|-----------------------|---------|-----|---------|----------|---------|
| utf8_general_ci       | utf8    | 33  | Yes     | Yes      | 1       |
| utf8_bin              | utf8    | 83  | Yes     | Yes      | 1       |
| utf8_unicode_ci       | utf8    | 192 | Yes     | Yes      | 8       |
| utf8_icelandic_ci     | utf8    | 193 | Yes     | Yes      | 8       |
| utf8_latvian_ci       | utf8    | 194 | Yes     | Yes      | 8       |
| utf8_romanian_ci      | utf8    | 195 | Yes     | Yes      | 8       |
| utf8_slovenian_ci     | utf8    | 196 | Yes     | Yes      | 8       |
| utf8_polish_ci        | utf8    | 197 | Yes     | Yes      | 8       |
| utf8_estonian_ci      | utf8    | 198 | Yes     | Yes      | 8       |
| utf8_slovak_ci        | utf8    | 199 | Yes     | Yes      | 8       |
| utf8_swedish_ci       | utf8    | 200 | Yes     | Yes      | 8       |
| utf8_turkish_ci       | utf8    | 201 | Yes     | Yes      | 8       |
| utf8_czech_ci         | utf8    | 202 | Yes     | Yes      | 8       |
| utf8_danish_ci        | utf8    | 203 | Yes     | Yes      | 8       |
| utf8_lithuanian_ci    | utf8    | 204 | Yes     | Yes      | 8       |
| utf8_slovak2_ci       | utf8    | 205 | Yes     | Yes      | 8       |
| utf8_spanish2_ci      | utf8    | 206 | Yes     | Yes      | 8       |
| utf8_roman_ci         | utf8    | 207 | Yes     | Yes      | 8       |
| utf8_persian_ci       | utf8    | 208 | Yes     | Yes      | 8       |
| utf8_esperanto_ci     | utf8    | 209 | Yes     | Yes      | 8       |
| utf8_hungarian_ci     | utf8    | 210 | Yes     | Yes      | 8       |
| utf8_sinhala_ci       | utf8mb4 | 211 | Yes     | Yes      | 1       |
| utf8mb4_general_ci    | utf8mb4 | 45  | Yes     | Yes      | 8       |
| utf8mb4_bin           | utf8mb4 | 46  | Yes     | Yes      | 1       |
| utf8mb4_unicode_ci    | utf8mb4 | 224 | Yes     | Yes      | 8       |
| utf8mb4_icelandic_ci  | utf8mb4 | 225 | Yes     | Yes      | 8       |
| utf8mb4_latvian_ci    | utf8mb4 | 226 | Yes     | Yes      | 8       |
| utf8mb4_romanian_ci   | utf8mb4 | 227 | Yes     | Yes      | 8       |
| utf8mb4_slovenian_ci  | utf8mb4 | 228 | Yes     | Yes      | 8       |
| utf8mb4_polish_ci     | utf8mb4 | 229 | Yes     | Yes      | 8       |
| utf8mb4_estonian_ci   | utf8mb4 | 230 | Yes     | Yes      | 8       |
| utf8mb4_spanish_ci    | utf8mb4 | 231 | Yes     | Yes      | 8       |
| utf8mb4_swedish_ci    | utf8mb4 | 232 | Yes     | Yes      | 8       |
| utf8mb4_turkish_ci    | utf8mb4 | 233 | Yes     | Yes      | 8       |
| utf8mb4_czech_ci      | utf8mb4 | 234 | Yes     | Yes      | 8       |
| utf8mb4_vietnamese_ci | utf8mb4 | 235 | Yes     | Yes      | 8       |
| utf8mb4_lithuanian_ci | utf8mb4 | 236 | Yes     | Yes      | 8       |
| utf8mb4_slovak_ci     | utf8mb4 | 237 | Yes     | Yes      | 8       |
| utf8mb4_spanish2_ci   | utf8mb4 | 238 | Yes     | Yes      | 8       |
| utf8mb4_roman_ci      | utf8mb4 | 239 | Yes     | Yes      | 8       |
| utf8mb4_persian_ci    | utf8mb4 | 240 | Yes     | Yes      | 8       |
| utf8mb4_esperanto_ci  | utf8mb4 | 241 | Yes     | Yes      | 8       |
| utf8mb4_hungarian_ci  | utf8mb4 | 242 | Yes     | Yes      | 8       |
| utf8mb4_sinhala_ci    | utf8mb4 | 243 | Yes     | Yes      | 8       |

图6-14 UTF-8中可用的校对规则。第一个规则uft\_general\_ci是默认规则

一般来说，我推荐使用UTF-8字符集，及它默认的校对规则。重要的是，数据库使用的字符集必须和你的PHP脚本一致。如果你的PHP脚本没有使用UTF-8编码，那就在数据库中使用与你的PHP脚本相匹配的编码。如果默认的校对规则不符合语言使用的大部分惯例，那就应该相应调整校对规则。

在MySQL中，服务器是一个整体，每个数据库、每个表，甚至每个字符串列都可以有一个定义的字符集和校对规则。要在创建数据库时设置这些值，使用以下命令：

```
CREATE DATABASE name CHARACTER SET charset COLLATE collation
```

要在创建表时设置这些值，使用以下命令：

```
CREATE TABLE name (
 column definitions
) CHARACTER SET charset COLLATE collation
```

要为列创建字符集和校对规则，在定义列时添加以下子句（只能为文本类型使用）：

```
CREATE TABLE name (
 something TEXT CHARACTER SET charset COLLATE collation
 ...)
```

在以上这些情况下，两个子句都是可选的。如果省略，会使用默认的字符集或校对规则。

定义数据库时确立字符集和校对规则会影响到可以存储的数据（例如，你不能在列中存储编码

不支持的字符)。第二个问题是与MySQL通信使用的编码。如果你想在表中使用中文编码存储中文字符,这些字符需要使用同样的编码传输。在MySQL客户端,使用下面的代码设置编码:

```
CHARSET charset
```

对于phpMyAdmin来说,要使用的编码是由应用程序本身来确定的(例如,写在配置文件中)。

到目前为止,论坛例子的数据库设计的各方面问题都已经解决了,接下来用MySQL创建这个数据库,包括它的索引、存储引擎、字符集和校对规则。

#### 指定字符集和校对规则

(1) 使用任何你喜欢的客户端访问MySQL。

和前面的章节一样,本章同样为所有的示例使用MySQL客户端。你可以使用phpMyAdmin或其他的工具连接MySQL。

(2) 创建论坛数据库(参见图6-15)。

```
CREATE DATABASE forum
CHARACTER SET utf8
COLLATE utf8_general_ci;
USE forum;
```

The screenshot shows a Windows-style terminal window titled "PHP and MySQL for Dynamic...". Inside, the MySQL command-line interface is running. The user has entered the following commands:

```
mysql> CREATE DATABASE forum
-> CHARACTER SET utf8
-> COLLATE utf8_general_ci;
Query OK, 1 row affected (0.00 sec)

mysql> USE forum;
Database changed

mysql>
```

图6-15 前几步是创建和选择数据库

取决于你的设置,你可能无法创建自己的数据库。如果不能创建,直接使用MySQL提供的数据库,并添加下面的表。注意,在CREATE DATABASE命令中,还定义了字符集和校对规则,这样做之后,每个表都会应用这些设置。

(3) 创建forums表(参见图6-16)。

```
CREATE TABLE forums (
 forum_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
 name VARCHAR(60) NOT NULL,
 PRIMARY KEY (forum_id),
 UNIQUE (name)
) ENGINE = INNODB;
```

The screenshot shows a Windows-style terminal window titled "PHP and MySQL for Dynamic Web Sites: Visual Quick...". Inside, the MySQL command-line interface is running. The user has entered the following command:

```
mysql> CREATE TABLE forums (
-> forum_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
-> name VARCHAR(60) NOT NULL,
-> PRIMARY KEY (forum_id),
-> UNIQUE (name)
->) ENGINE = INNODB;
Query OK, 0 rows affected (0.04 sec)

mysql>
```

图6-16 创建第一个表

以什么顺序创建表并不重要，但是我会首先创建forums表。记住，为方便起见，你可以输入多行SQL语句。

这个表只包含两列（在规范化的数据库中这会非常常见）。因为我不希望有太多的讨论，所以主键是一个很小的类型（TINYINT）。如果要为每个讨论添加说明描述，可以添加一个VARCHAR(255)或者TINYTEXT类型的列。这个表使用InnoDB存储引擎。

#### (4) 创建messages表（参见图6-17）：

```
CREATE TABLE messages (
 message_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 parent_id INT UNSIGNED NOT NULL DEFAULT 0,
 forum_id TINYINT UNSIGNED NOT NULL,
 user_id MEDIUMINT UNSIGNED NOT NULL,
 subject VARCHAR(100) NOT NULL,
 body LONGTEXT NOT NULL,
 date_entered DATETIME NOT NULL,
 PRIMARY KEY (message_id),
 INDEX (parent_id),
 INDEX (forum_id),
 INDEX (user_id),
 INDEX (date_entered)
) ENGINE = MYISAM;
```

```
mysql> CREATE TABLE messages (
 -> message_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 -> parent_id INT UNSIGNED NOT NULL DEFAULT 0,
 -> forum_id TINYINT UNSIGNED NOT NULL,
 -> user_id MEDIUMINT UNSIGNED NOT NULL,
 -> subject VARCHAR(100) NOT NULL,
 -> body LONGTEXT NOT NULL,
 -> date_entered DATETIME NOT NULL,
 -> PRIMARY KEY (message_id),
 -> INDEX (parent_id),
 -> INDEX (forum_id),
 -> INDEX (user_id),
 -> INDEX (date_entered)
 ->) ENGINE = MYISAM;
Query OK, 0 rows affected (0.05 sec)

mysql>
```

图6-17 创建第二个表

这个表的主键必须很大，因为它可能会存储非常多的记录。三个外键（forum\_id、parent\_id和user\_id）将会与其对应主键的大小和类型相同。每条消息的主题限制在100个字符以内，每条消息的主体可以有大段文本。date\_entered字段是DATETIME类型。

不同于其他的两个表，这个messages表是MyISAM类型的。

#### (5) 创建users表（参见图6-18）。

```
CREATE TABLE users (
 user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
 username VARCHAR(30) NOT NULL,
 pass CHAR(40) NOT NULL,
 first_name VARCHAR(20) NOT NULL,
 last_name VARCHAR(40) NOT NULL,
 email VARCHAR(60) NOT NULL,
 PRIMARY KEY (user_id),
```

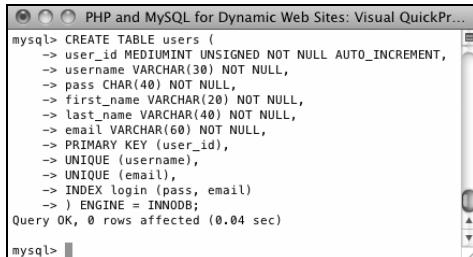
```

UNIQUE (username),
UNIQUE (email),
INDEX login (pass, email)
) ENGINE = INNODB;

```

这里大多数列都模仿了前两章创建的sitename数据库的users表的那些列。pass列被定义为CHAR(40)类型，因为这里会使用SHA1()函数，并且它总是返回长度为40个字符的字符串（参见第5章）。

这个表使用了InnoDB引擎。



```

mysql> CREATE TABLE users (
-> user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
-> username VARCHAR(30) NOT NULL,
-> pass CHAR(40) NOT NULL,
-> first_name VARCHAR(20) NOT NULL,
-> last_name VARCHAR(40) NOT NULL,
-> email VARCHAR(60) NOT NULL,
-> PRIMARY KEY (user_id),
-> UNIQUE (username),
-> UNIQUE (email),
-> INDEX login (pass, email)
->) ENGINE = INNODB;
Query OK, 0 rows affected (0.04 sec)

mysql>

```

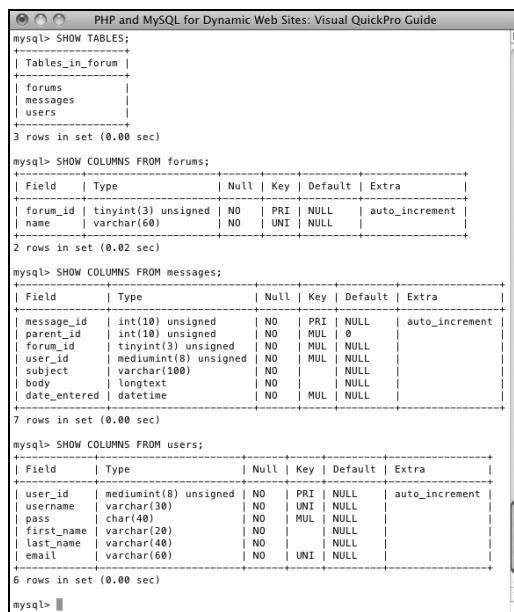
图6-18 创建users表

(6) 如果需要，确认数据库的结构（参见图6-19）。

```

SHOW TABLES;
SHOW COLUMNS FROM forums;
SHOW COLUMNS FROM messages;
SHOW COLUMNS FROM users;

```



```

mysql> SHOW TABLES;
+----------------+
| Tables_in_forum |
+----------------+
| forums |
| messages |
| users |
+----------------+
3 rows in set (0.00 sec)

mysql> SHOW COLUMNS FROM forums;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| forum_id | tinyint(3) unsigned | NO | PRI | NULL | auto_increment |
| name | varchar(60) | NO | UNI | NULL | |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql> SHOW COLUMNS FROM messages;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
message_id	int(10) unsigned	NO	PRI	NULL	auto_increment
parent_id	int(10) unsigned	NO	MUL	NULL	
forum_id	tinyint(3) unsigned	NO	MUL	NULL	
user_id	mediumint(8) unsigned	NO	MUL	NULL	
subject	varchar(100)	NO		NULL	
body	longtext	NO		NULL	
date_entered	datetime	NO	MUL	NULL	
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> SHOW COLUMNS FROM users;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
user_id	mediumint(8) unsigned	NO	PRI	NULL	auto_increment
username	varchar(30)	NO	UNI	NULL	
pass	char(40)	NO	MUL	NULL	
first_name	varchar(20)	NO		NULL	
last_name	varchar(40)	NO		NULL	
email	varchar(60)	NO	UNI	NULL	
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>

```

图6-19 可以使用SHOW来检查任何数据库或表的结构

SHOW命令显示了数据库或表的信息。这个步骤是可选的，因为在每次输入查询时，MySQL会报告查询成功。尽管如此，提醒下数据库的结构总是件好事。

#### ✓ 提示

- MySQL的校对规则同样可以在查询中指定以影响查询结果：

```
SELECT ... ORDER BY column COLLATE collation
SELECT ... WHERE column LIKE 'value' COLLATE collation
```

- CONVERT()函数可以转换文本的字符集。
- 你可以使用ALTER命令改变数据库或表的默认字符集或校对规则，详见第7章。
- 因为使用不同的字符集需要占用更多的空间表示字符串，所以可能需要为UTF-8字符增加列的大小。在修改列编码之前编辑列的大小以防止数据丢失。

## 6.5 时区和MySQL

第5章讨论了如何使用NOW()函数及其他与日期和时间相关的函数。这些函数反映了服务器上的时间。因而，使用这些函数存储在数据库中的值同样存储的是服务器时间。这听起来好像没有什么问题，但是当你从一台服务器移动你的网站到另外一台上时：你导出所有的数据，导入到另外一台，所有的东西都很好……但如果这两台服务器在不同的时区，在这种情况下，所有的日期都出错了。对一些站点来说，这样的改变可能没什么，但是如果您的站点有付费的会员资格功能会怎样？那就意味着有些人的会员资格可能提前几个小时到期或者有些会晚几个小时。这与数据库可靠的存储信息的目标相悖。

这个特殊问题的解决方案是使用时区无关的方式来存储日期和时间。这就需要用到UTC (Coordinated Universal Time, 协调世界时, 是的, 缩写不完全匹配术语)。UTC, 就像GMT (Greenwich Mean Time, 格林尼治标准时间), 提供了一个共同的起源点, 世界上所有的时间都可以使用UTC加上或减去几个小时和分钟来表示(参见表6-8)。

表6-8 UTC偏移量

| 城 市     | 时 间      |
|---------|----------|
| 美国纽约    | UTC-4    |
| 南非开普敦   | UTC+2    |
| 印度孟买    | UTC+5:30 |
| 新西兰奥克兰  | UTC+13   |
| 尼泊尔加德满都 | UTC+5:45 |
| 智利圣地亚哥  | UTC-3    |
| 爱尔兰都柏林  | UTC+1    |

幸运的是，你不需要知道这些值或者进行任何计算来确定服务器的UTC。UTC\_DATE()函数会返回UTC日期、UTC\_TIME()函数会返回当前的UTC时间、UTC\_TIMESTAMP()函数会返回当前的日期和时间。

一旦你存储了UTC时间，你可以获取经过校正的反映服务器或用户位置的时间。要将一个日期或

时间从一个时区调整到另外一个，使用`CONVERT_TZ()`函数（参见图6-20）：

`CONVERT_TZ(dt, from, to)`

```
cmd C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p sitename
mysql> SELECT CONVERT_TZ(UTC_TIMESTAMP(), 'UTC', 'America/New_York')
+-----+
| CONVERT_TZ(UTC_TIMESTAMP(), 'UTC', 'America/New_York') |
+-----+
| 2011-04-03 21:18:27 |
+-----+
1 row in set (0.02 sec)

mysql>
```

图6-20 将当前UTC日期和时间转换为北美东部时区（EDT）

第一个参数是日期和时间值，如一个函数或存储在列中的数据。第二个和第三个参数是时区的名称。为了使用此函数，MySQL中要先存储时区列表，这可能与你的安装情况有所不同（见框注“在MySQL中使用时区”）。如果结果是NULL（参见图6-21），参考MySQL手册在你的服务器上安装的时区。

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT CONVERT_TZ(UTC_TIMESTAMP(), 'UTC', 'America/New_York');
+-----+
| CONVERT_TZ(UTC_TIMESTAMP(), 'UTC', 'America/New_York') |
+-----+
| NULL |
+-----+
1 row in set (0.00 sec)

mysql>
```

图6-21 如果引用了无效的时区或MySQL上没有安装时区，`CONVERT_TZ()`函数会返回NULL

接下来，我们填充论坛数据库，使用UTC记录信息发布的日期和时间。

### 在MySQL中使用时区

MySQL默认不安装时区支持。为了使用有名字的时区，需要在MySQL数据库中添加5个表。MySQL不会自动为你完成安装，但它提供了让你自己完成安装的工具。

这个过程非常复杂，本书没有足够的篇幅来讨论它（不可能涉及每一种情况，如操作系统等）。你可以在MySQL手册中查找“服务器时区支持”获取详细信息。

如果你继续在MySQL中使用时区，还需要实时更新MySQL数据库信息。时区的规则（尤其是何时、如何使用夏令时）经常改变。MySQL手册中也有如何更新时区的介绍。

#### 使用UTC

(1) 访问论坛数据库，使用任何你喜欢的客户端。

如同前面的章节，本章同样为所有的示例使用MySQL客户端。欢迎你使用phpMyAdmin或其他的工具连接MySQL的界面。

(2) 如果有必要，将编码改为UTF-8（参见图6-22）。

`CHARSET utf8;`

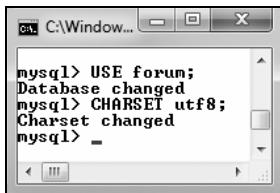


图6-22 用于与MySQL通信的字符集应该与数据库的字符集一致

由于数据库使用UTF-8字符集，与数据库进行通信需要使用相同的字符集。这行代码在上一节解释过。注意，你只需要在使用MySQL客户端的时候这么做。此外，如果你使用的不是UTF-8编码，需要相应更改命令。

(3) 在forums表中添加一些新的记录（参见图6-23）。

```
INSERT INTO forums (name) VALUES
('MySQL'), ('PHP'), ('Sports'),
('HTML'), ('CSS'), ('Kindling');
```

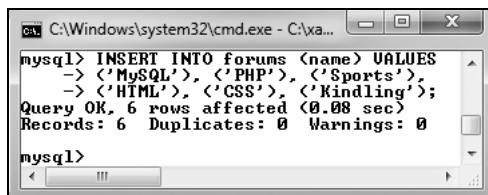


图6-23 向forums表添加记录

由于信息表依赖于从forums表和users表获取的值，所以这两个表需要先填充。在INSERT命令中，只有name列必须提供一个值（表的forum\_id列会由MySQL自动生成一个自增长的整数）。

(4) 添加几条记录到users表（参见图6-24）。

```
INSERT INTO users (username, pass,
first_name, last_name, email) VALUES
('troutster', SHA1('mypass'),
'Larry', 'Ullman', 'lu@example.com'),
('funny man', SHA1('monkey'),
'David', 'Brent', 'db@example.com'),
('Gareth', SHA1('asstmgr'),
'Gareth',
'Keenan', 'gk@example.com');
```

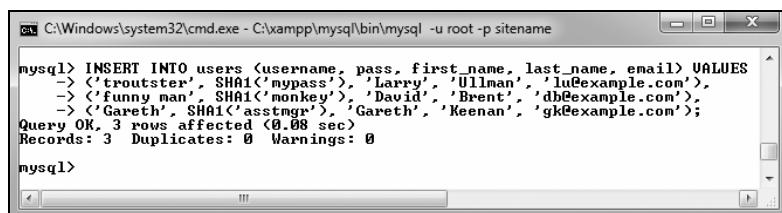


图6-24 向users表添加记录

如果你对这里的INSERT语法或者如何使用SHA1()函数有任何疑问，参见第5章。

(5) 添加新记录到的messages表（参见图6-25）。

```
SELECT * FROM forums;
SELECT user_id, username FROM users;
INSERT INTO messages (parent_id, forum_id, user_id, subject, body, date_entered) VALUES
(0, 1, 1, 'Question about normalization.', 'I''m confused about normalization. For the second normal
form (2NF), I read...', UTC_TIMESTAMP()),
(0, 1, 2, 'Database Design', 'I''m creating a new database and am having problems with the structure.
How many tables should I have?...', UTC_TIMESTAMP()),
(2, 1, 2, 'Database Design', 'The number of tables your database includes...', UTC_TIMESTAMP()),
(0, 1, 3, 'Database Design', 'Okay, thanks!', UTC_TIMESTAMP()),
(0, 2, 3, 'PHP Errors', 'I''m using the scripts from Chapter 3 and I can''t get the first calculator
example to work. When I submit the form...', UTC_TIMESTAMP());
```

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u r...'. It displays the following MySQL session:

```
mysql> SELECT * FROM forums;
+----+-----+
| forum_id | name |
+----+-----+
5	CSS
4	HTML
6	Kindling
1	MySQL
2	PHP
3	Sports
+----+-----+
6 rows in set <0.02 sec>

mysql> SELECT user_id, username FROM users;
+----+-----+
| user_id | username |
+----+-----+
5	finchy
2	funny man
3	Gareth
4	tim
1	troutster
+----+-----+
5 rows in set <0.00 sec>

mysql> INSERT INTO messages (parent_id, forum_id, user_id,
-> (0, 1, 1, 'Question about normalization.', 'I''m co
-> _TIMESTAMP()),
-> (0, 1, 2, 'Database Design', 'I''m creating a new d
ave?...', UTC_TIMESTAMP()),
-> (2, 1, 2, 'Database Design', 'The number of tables
-> (0, 1, 3, 'Database Design', 'Okay, thanks!', UTC_T
-> (0, 2, 3, 'PHP Errors', 'I''m using the scripts fro
nit the form...', UTC_TIMESTAMP());
Query OK, 5 rows affected <0.00 sec>
Records: 5 Duplicates: 0 Warnings: 0

mysql>
```

图6-25 添加新记录到的messages表

因为在信息表中的两个字段（forum\_id和user\_id）涉及其他表中的值，向表中插入新记录之前需要知道这些值。例如，如果troutster用户在MySQL讨论中创建一个新的消息，forum\_id将为1、user\_id也将为1。

parent\_id则使问题变得更加复杂了，它应该存储新消息所回复消息的message\_id。第二条要被添加到数据库中的消息会的message\_id将为2，因此这条消息的回复需要一个为2的parent\_id。

一旦你使用PHP脚本创建了这个数据库的接口，这个过程将变得非常容易，但是首先理解SQL方面的理论非常重要。

由UTC\_TIMESTAMP()函数返回的值将被用于date\_entered字段。使用UTC\_TIMESTAMP()函数，该记录将存储UTC日期和时间，而不是服务器上的日期和时间。

(6) 重复步骤(1)至(3)填充数据库。

本章及下一章的例子将使用这个填充好的数据库。你可以从本书的网站下载SQL命令。当然你也可以用自己的例子填充表，然后更改本章余下内容中的查询即可。

(7) 查看messages表的最新纪录，使用存储的日期和时间（参见图6-26）。

```
SELECT message_id, subject, date_entered FROM messages
ORDER BY date_entered DESC LIMIT 1;
```

| message_id | subject | date_entered        |
|------------|---------|---------------------|
| 21         | Because | 2011-04-04 01:37:27 |

图6-26 刚刚插入的记录，时间提前了4小时（服务是UTC-4）

你可以在图6-26和表的定义中看到，存储UTC时间与存储非UTC时间相同。刚插入的记录反映的时间比服务器时间早4个小时（因为我的服务器所在的时区比UTC晚4个小时），但这在图6-26中看不太出来。

(8) 获取与第(7)步相同的记录，并将date\_entered转为你所在的时区（参见图6-27）。

```
SELECT message_id, subject,
CONVERT_TZ(date_entered, 'UTC', 'America/New_York') AS local
FROM messages ORDER BY date_entered DESC LIMIT 1;
```

| message_id | subject | local               |
|------------|---------|---------------------|
| 21         | Because | 2011-04-03 21:37:27 |

图6-27 将以UTC存储的日期和时间转换为我的本地时间

使用CONVERT\_TZ()函数，可以转换任何日期和时间到不同的时区。为“源”的时区使用UTC，为要转换的时区使用你的时区。

如果返回结果为NULL（参见图6-21），可能是时区的名字拼写错误，也可能是MySQL还没有载入那个时区（见框注“在MySQL中使用时区”）。

### ✓ 提示

- 无论你决定如何处理日期，最关键的是要保持一致。如果你决定使用UTC，那就一直使用UTC。
- UTC也被称为祖鲁（Zulu）时间，代表字母是Z。
- UTC除了不依赖时区和夏令时间外，也非常准确。它补偿了由于地球的不规则运动导致的闰秒因素。

## 6.6 外键约束

InnoDB类型表有一个其他存储引擎不支持的特性，即应用外键约束的能力。当你有相关的表，表B的外键关联到表A的主键（为了便于理解，假设表B为父表A的子表）。例如，在论坛数据库中，`messages.user_id`绑定到`users.user_id`。如果管理员删除了用户账户，这些表之间的关系将被破坏（因为`messages`表中将有一个`users`表中不存在的`user_id`值的记录）。外键约束设定了当约束被破坏发生时应该应用的规则，包括防止约束破坏。

创建一个外键约束的语法是：

```
FOREIGN KEY (item_name) REFERENCES table (column)
```

（在CREATE TABLE或ALTER TABLE语句中添加。）

条目的名称是当前表中的外键列。`table(column)`子句是约束该外键的父表列的引用。如果你只是使用上面的内容，那就只是标识了关系，而并没有说明约束被破坏如何操作。如果你试图在子记录存在时删除父记录MySQL会抛出一个错误（参见图6-28）。如果你尝试使用不存在的父ID创建子记录，MySQL同样将抛出错误（参见图6-29）。

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPr...
mysql> DELETE FROM parent WHERE parent_id=1;
ERROR 1451 (23000): Cannot delete or update a parent row: a
foreign key constraint fails (`test`.`child`, CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`parent_id`))
mysql>
```

图6-28 这个错误说明MySQL禁止删除父记录，因为该记录被约束了一个或多个子记录

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> INSERT INTO child
 -> (child_id, parent_id)
 -> VALUES (NULL, 20934);
ERROR 1452 (23000): Cannot add or update a child row: a foreign
key constraint fails (`test`.`child`, CONSTRAINT `child_ibfk_1` FOREIGN KEY (`parent_id`) REFERENCES `parent` (`parent_id`))
mysql>
```

图6-29 外键约束同样会影响INSERT语句

你可以在上面的语句后附加如下内容来决定触发什么动作：

```
ON DELETE action
ON UPDATE action
```

总共有5个动作可选，但有两个（`RESTRICT`和`NO ACTION`）是同义词，也是默认的（即不指定任何动作）。第三个动作选项（`SET DEFAULT`）不起作用（不要问我，查MySQL手册！）。接下来就只剩`CASCADE`和`SET NULL`了。如果设置的是`SET NULL`动作，删除父记录将导致子表中相应的外键被设置为`NULL`。如果表的该列定义为`NOT NULL`（通常是这样），删除父记录将引发一个错误。

`CASCADE`操作是最有用的选项。它告诉数据库关联表采用同样的变化。根据这个指令，如果你删除父记录，MySQL也将删除以那个父ID作为外键的子记录。

由于只有InnoDB表类型的表支持外键约束，所以相关的两个表必须都是InnoDB类型。此外，为了MySQL能够比较外键与主键的值，相关的列必须是相同的类型。这意味着，数字列的大小必须相同；文本列的字符集和校对规则必须相同。

在论坛例子中，不可能使用外键约束，因为只有一个`messages`表关联了其他表（同时关联了`forums`表和`users`表），但该表必须使用MyISAM表类型（需要支持FULLTEXT搜索）。下面，我们来看一个新的假想的银行数据库示例（参见图6-30）。

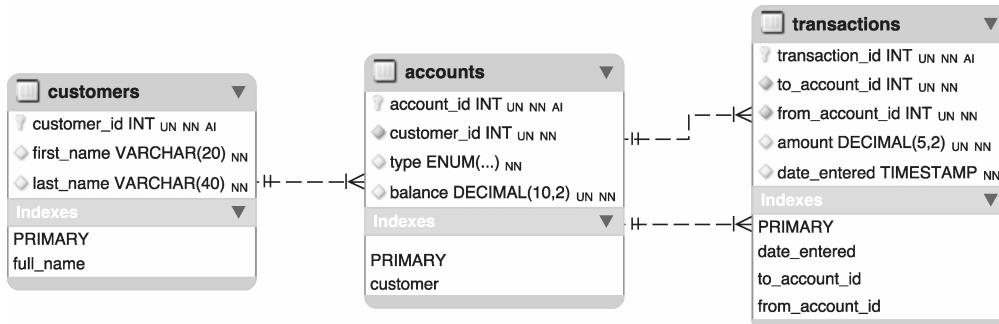


图6-30 银行数据库可用于虚拟银行

`customers`表中存储了客户所有详细信息。逻辑上它也存储联系人信息，等等。账目表存储每个客户的账户，包括类型（支票或储蓄）和余额。每个客户可能有多个账户，但每个账户只对应一个客户（有点简单）。在现实世界中，还可能存储账户的开户日期，并为余额使用`BIGINT`类型（从而为所有交易使用美分而不是带小数的美元）。最后，`transactions`表存储账户间的货币流通。为了让示例更容易理解，这里假定只有相同系统的账号才能更交互。注意，`transactions`表与账户之间有两个一对多的关系（不是多对多）。每笔交易的`to_account_id`值将会被分配一个单独账户，但是每个账户都可以多次作为“目的”账户。这同样适用于“源”账户。最后，外键约束用于保持数据的完整性。

在这接下来的一系列步骤中，你将创建并填充数据库，着重注意约束。在下一章中，这个数据库将被用来演示交易和加密。

#### 创建外键约束

(1) 使用你喜欢的任何一种客户程序访问MySQL。

与前面章节一样，本章也将对所有示例使用MySQL客户端。欢迎你使用phpMyAdmin或其他的工具连接MySQL。

(2) 创建银行数据库（参见图6-31）。

```
CREATE DATABASE banking
CHARACTER SET utf8
COLLATE utf8_general_ci;
USE banking;
```

```
mysql> CREATE DATABASE banking
-> CHARACTER SET utf8
-> COLLATE utf8_general_ci;
Query OK, 1 row affected (0.00 sec)

mysql> USE banking;
Database changed
mysql>
```

图6-31 为示例创建新数据库

取决于你的设置，你可能无法创建自己的数据库。如果不能创建，直接使用MySQL提供的数据库，并添加下面的表。

(3) 如果有必要，将通信编码改为UTF-8。

```
CHARSET utf8;
```

(4) 创建customers表（参见图6-32）。

```
CREATE TABLE customers (
customer_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
first_name VARCHAR(20) NOT NULL,
last_name VARCHAR(40) NOT NULL,
PRIMARY KEY (customer_id),
INDEX full_name (last_name, first_name)
) ENGINE = INNODB;
```

```
mysql> CREATE TABLE customers (
-> customer_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> first_name VARCHAR(20) NOT NULL,
-> last_name VARCHAR(40) NOT NULL,
-> PRIMARY KEY (customer_id),
-> INDEX full_name (last_name, first_name)
->) ENGINE = INNODB;
Query OK, 0 rows affected (0.17 sec)

mysql>
```

图6-32 创建customers表

customers表中只存储了客户的主键ID和名称（在两列中）。全名也添加了一个索引，因为它可能被用在ORDER BY或其他的查询子句中。因此，该数据库可以使用外键约束，所有的表都将使用InnoDB存储引擎。

(5) 创建accounts表（参见图6-33）。

```
CREATE TABLE accounts (
account_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
customer_id INT UNSIGNED NOT NULL,
type ENUM('Checking', 'Savings') NOT NULL,
balance DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT 0.0,
```

```

PRIMARY KEY (account_id),
INDEX (customer_id),
FOREIGN KEY (customer_id) REFERENCES customers (customer_id)
ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE = INNODB;

```

```

C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p sitename
mysql> CREATE TABLE accounts (
-> account_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> customer_id INT UNSIGNED NOT NULL,
-> type ENUM('Checking', 'Savings') NOT NULL,
-> balance DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT 0.0,
-> PRIMARY KEY (account_id),
-> INDEX (customer_id),
-> FOREIGN KEY (customer_id) REFERENCES customers (customer_id)
-> ON DELETE NO ACTION ON UPDATE NO ACTION
->) ENGINE = INNODB;
Query OK, 0 rows affected <0.14 sec>

mysql>

```

图6-33 创建accounts表

accounts表中存储了账户ID、客户ID、账户类型和余额。customer\_id列上有一个索引，因为它会被用于JOIN命令（参见第7章）。更重要的是，列被约束到customers.customer\_id，从而保证这两个表数据的一致性。即使NO ACTIONS是默认约束，但我也在定义中显示地添加了它。

(6) 创建transactions表（参见图6-34）。

```

CREATE TABLE transactions (
transaction_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
to_account_id INT UNSIGNED NOT NULL,
from_account_id INT UNSIGNED NOT NULL,
amount DECIMAL(5,2) UNSIGNED NOT NULL,
date_entered TIMESTAMP NOT NULL,
PRIMARY KEY (transaction_id),
INDEX (to_account_id),
INDEX (from_account_id),
INDEX (date_entered),
FOREIGN KEY (to_account_id) REFERENCES accounts (account_id)
ON DELETE NO ACTION ON UPDATE NO ACTION,
FOREIGN KEY (from_account_id) REFERENCES accounts (account_id)
ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE = INNODB;

```

```

C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p sitename
mysql> CREATE TABLE transactions (
-> transaction_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> to_account_id INT UNSIGNED NOT NULL,
-> from_account_id INT UNSIGNED NOT NULL,
-> amount DECIMAL(5,2) UNSIGNED NOT NULL,
-> date_entered TIMESTAMP NOT NULL,
-> PRIMARY KEY (transaction_id),
-> INDEX (to_account_id),
-> INDEX (from_account_id),
-> INDEX (date_entered),
-> FOREIGN KEY (to_account_id) REFERENCES accounts (account_id)
-> ON DELETE NO ACTION ON UPDATE NO ACTION,
-> FOREIGN KEY (from_account_id) REFERENCES accounts (account_id)
-> ON DELETE NO ACTION ON UPDATE NO ACTION
->) ENGINE = INNODB;
Query OK, 0 rows affected <0.16 sec>

mysql>

```

图6-34 创建第三个（最后一个表）——transactions表

最后一个表将用来记录账目之间的所有款项流通。要实现它，需要同时存储“源”和“目标”账户的ID，以及日期时间。添加相应的索引，约束两个账户ID到accounts表。

(7) 填充customers表和accounts表（参见图6-35）。

```
INSERT INTO customers (first_name, last_name)
VALUES ('Sarah', 'Vowell'), ('David', 'Sedaris'), ('Kojo', 'Nnamdi');
INSERT INTO accounts (customer_id, balance)
VALUES (1, 5460.23), (2, 909325.24), (3, 892.00);
```

```
c:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p sitename
mysql> INSERT INTO customers (first_name, last_name)
-> VALUES ('Sarah', 'Vowell'), ('David', 'Sedaris'), ('Kojo', 'Nnamdi');
Query OK, 3 rows affected <0.08 sec>
Records: 3 Duplicates: 0 Warnings: 0
mysql> INSERT INTO accounts (customer_id, balance)
-> VALUES (1, 5460.23), (2, 909325.24), (3, 892.00);
Query OK, 3 rows affected <0.00 sec>
Records: 3 Duplicates: 0 Warnings: 0
mysql> _
```

图6-35 向customers表和accounts表添加3条记录

首先，在前两个表中输入简单的数据（第三个将在下一章使用）。注意，因为accounts.type column被定义为ENUM NOT NULL，如果没有为那列提供值，将会使用枚举定义中的第一个值——Checking。

(8) 尝试将不存在的客户数据插入到accounts表（参见图6-36）。

```
INSERT INTO accounts (customer_id, type, balance)
VALUES (10, 'Savings', 200.00);
```

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> INSERT INTO accounts (customer_id, type, balance)
-> VALUES (10, 'Savings', 200.00);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key
constraint fails ('banking'.`accounts`, CONSTRAINT `accounts_ibfk_1`
FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`)
ON DELETE NO ACTION ON UPDATE NO ACTION)
mysql> _
```

图6-36 和图6-29一样，约束拒绝了INSERT语句，因为父表中该字段无效

在没有有效的客户ID的情况下，customers表中的外键约束会阻止创建账户（在现实世界中非常有用的操作）。

(9) 尝试在存在账户记录的情况下删除customers表中的一条记录（参见图6-37）。

```
DELETE FROM customers
WHERE customer_id=2;
```

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> DELETE FROM customers
-> WHERE customer_id=2;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails
('banking'.`accounts`, CONSTRAINT `accounts_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES
`customers` (`customer_id`) ON DELETE NO ACTION ON UPDATE NO ACTION)
mysql> _
```

图6-37 由于id为2的用户在accounts表中存在一条或多条记录，因此该记录无法被删除

在客户仍然存在账户的情况下，外键会阻止删除customers表的记录。

尽管存在约束，如果客户在accounts表中没有任何记录你仍然可以删除它。

#### ✓ 提示

- 要删除受约束的记录，你必须先删除所有的子记录，然后再删除父记录。
- 外键约束要求被约束的所有列必须是索引列。正常的数据库设计的建议是这样的，但如果不存在正确的索引，MySQL将在定义约束时创建它们。
- 与约束类似的是触发器。简而言之，触发器就是告诉数据库“当此表发生X时，做Y”的一种方法。例如，在表A中插一条记录，在表B中可能会创建或更新一条记录。更多关于触发器的内容请参考MySQL手册。

## 6.7 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书的论坛上，我们会为你答疑解惑。

### 6.7.1 回顾

- 规范化为什么很重要？
- 键的两种类型分别是？
- 表关系有哪三种类型？
- 你该如何修正两个表之间多对多的问题？
- 4种索引分别是什么？通常哪些类型的列应该被索引？哪些类型的列不应该被索引？
- 两种最常见的表类型是什么？你安装的MySQL默认的表类型是什么？
- 什么是字符集？什么是校对规则？字符集对数据库有什么影响？校对规则对数据库有什么影响？你正使用的字符集和校对规则是什么？
- 什么是UTC？如何在MySQL中找到UTC时间？如何从UTC转换到另一个时区的时间？
- 什么是外键约束？什么类型的表支持外键约束？

### 6.7.2 实践

- 你可以考虑下载、安装并学习使用MySQL Workbench软件。它确实非常有用。
- 如果你还没有完全掌握规范化的过程——这是完全可以理解的——搜索网上的其他教程或在我的支持论坛提问。
- 利用这里介绍的信息设计你自己的数据库。

**本章内容**

- 执行联结
- 分组选定的结果
- 高级选择
- 执行FULLTEXT查找
- 查询优化
- 执行事务
- 数据库加密
- 回顾和实践

**这**是专门讨论SQL和MySQL的最后一章（本书的其余部分将会以各种形式使用这些技术），利用前面几章创建的数据库，讨论一些会经常用到的、更复杂的数据库高级概念。首先会介绍联结（Join），它用于在规范化的数据库中查询多个表的重要的SQL术语。本章将介绍一类专门用于分组查询结果的函数，接下来会探讨一些更复杂的从表中选取值的方法。

本章中，你会学习如何执行FULLTEXT搜索，它可以为网站添加类似搜索引擎的功能。接下来会介绍EXPLAIN命令：它提供了一种方法来测试数据库架构和查询效率。本章的最后阐述了事务执行和数据库加密技术。

## 7.1 执行联结

由于关系数据库的结构较复杂，它们有时需要特殊的查询语句来检索你最需要的信息。例如，如果想知道哪些消息在MySQL论坛数据库中（使用上一章创建的论坛数据库），首先需要找到MySQL的forum\_id：

```
SELECT forum_id FROM forums WHERE name='MySQL'
```

然后使用该编号从消息表中检索具有这个forum\_id的所有记录。

```
SELECT * FROM messages WHERE forum_id=1
```

这一个简单的（在网站论坛中，通常是必要的）任务将需要两个单独的查询。通过使用联结，可以一下子完成所有这些操作。

联结是使用两个或更多表的SQL查询，它产生虚拟的结果表。任何需要同时从多个表检索信息时，都可以使用联结。

联结可以以许多不同的方式编写，但基本语法是：

```
SELECT what_columns FROM tableA JOIN_ TYPE tableB JOIN_ CLAUSE
```

因为联结涉及多个表，*what\_columns*可以包含任何指定表中的列。作为联结经常返回非常多的信息，通常最好是指定你需要返回的列，而不是返回所有的列。

当从多个表中选择列时，如果查询中的表有重名的列，就必须使用点语法 (*table.column*)。在处理关系型数据库时这会是经常发生的情况，因为一个表的主键可能会在另一个表的外键，并且名字相同。如果你没有指明引用的列，你会看到如下错误信息（参见图7-1）：

```
SELECT forum_id FROM messages INNER JOIN
forums ON messages.forum_id=forums.forum_id
```

The screenshot shows a Windows-style terminal window titled "PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide". Inside, a MySQL session is running:

```
mysql> SELECT forum_id FROM messages INNER JOIN
-> forums ON messages.forum_id=forums.forum_id;
ERROR 1052 (23000): Column 'forum_id' in field list is ambiguous
mysql>
```

图7-1 通常在多个表中引用一个列名会导致歧义错误

两个主要的联结类型是：内联结和外联结（它们都有子类型）。你将在外联结中看到，引用表的顺序很重要。

`join`子句用于表明联结的表之间的关系。例如，`forums.forum_id`应该与`messages.forum_id`相等（如上所述）。

和SELECT查询一样，你可以为联结使用WHERE和ORDER BY子句。

最后要注意的是，在更加深入了解联结之前，在第5章中介绍的SQL概念“别名”将会在编写联结时派上用场。别名是表名的简写，通常用来代替在同一查询中多次引用的表名。如果你不记得创建别名的语法，或者如何使用它们，请重温第5章的那部分内容。

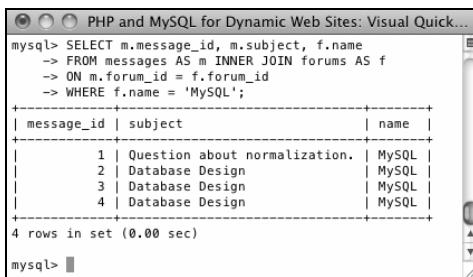
（和前两章一样，本章也将使用MySQL客户端命令行来执行查询，但你也可以使用phpMyAdmin或其他工具。此外，你还应该了解如何连接到MySQL服务器并声明要使用的字符集。）

### 7.1.1 内联结

无论何时产生匹配，内联结都会从指定的表中返回所有记录。例如，为了查找MySQL论坛中的每条消息，将像下面这样编写内联结（参见图7-2）：

```
SELECT m.message_id, m.subject, f.name
FROM messages AS m INNER JOIN forums AS f
ON m.forum_id = f.forum_id
WHERE f.name = 'MySQL'
```

这个联结从消息表（别名m）中选择3列，从讨论表选择1列（别名f），条件有两个：第一，`f.name`列必须具有MySQL的值（`forum_id`的值为1）；第二，讨论表中的`forum_id`值必须匹配消息表中的`forum_id`值。由于跨两个表执行相等性比较（`m.forum_id = f.forum_id`），这称为等值联结（equijoin）。



```
mysql> SELECT m.message_id, m.subject, f.name
-> FROM messages AS m INNER JOIN forums AS f
-> ON m.forum_id = f.forum_id
-> WHERE f.name = 'MySQL';
+-----+-----+-----+
| message_id | subject | name |
+-----+-----+-----+
1	Question about normalization.	MySQL
2	Database Design	MySQL
3	Database Design	MySQL
4	Database Design	MySQL
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

图7-2 这个联结从两个表中返回3列，其中forum\_id值为1表示MySQL论坛

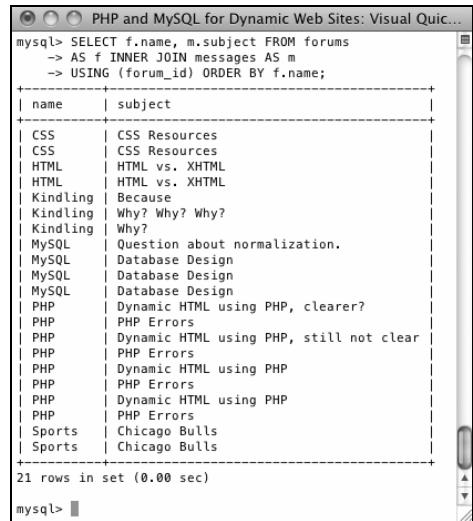
作为一种替代的语法，如果在等值比较中两个表的列具有相同的名称，可以使用USING简化查询：

```
SELECT m.message_id, m.subject, f.name
FROM messages AS m INNER JOIN forums AS f
USING (forum_id)
WHERE f.name = 'MySQL'
```

### 使用内联结

- (1) 连接到MySQL并选择论坛数据库。
- (2) 从消息表中检索所有记录的讨论名和消息正文（参见图7-3）。

```
SELECT f.name, m.subject FROM forums
AS f INNER JOIN messages AS m
USING (forum_id) ORDER BY f.name;
```



```
mysql> SELECT f.name, m.subject FROM forums
-> AS f INNER JOIN messages AS m
-> USING (forum_id) ORDER BY f.name;
+-----+-----+
| name | subject |
+-----+-----+
| CSS | CSS Resources
| CSS | CSS Resources
| HTML | HTML vs. XHTML
| HTML | HTML vs. XHTML
| Kindling | Because
| Kindling | Why? Why? Why?
| Kindling | Why?
| MySQL | Question about normalization.
| MySQL | Database Design
| MySQL | Database Design
| MySQL | Database Design
| PHP | Dynamic HTML using PHP, clearer?
| PHP | PHP Errors
| PHP | Dynamic HTML using PHP, still not clear
| PHP | PHP Errors
| PHP | Dynamic HTML using PHP
| PHP | PHP Errors
| PHP | Dynamic HTML using PHP
| PHP | PHP Errors
| Sports | Chicago Bulls
| Sports | Chicago Bulls
+-----+-----+
21 rows in set (0.00 sec)

mysql>
```

图7-3 基本内联结仅返回两列的值

这个查询实际上使用讨论表中相应的name值代替消息表中每条记录的forum\_id值。最终的结果是，它为每个消息主题显示讨论名称的文本版。

注意，联结中也可以在使用ORDER BY子句。

(3) 获取由funny man发布的所有信息的主题和发布日期（参见图7-4）。

```
SELECT m.subject,
DATE_FORMAT(m.date_entered, '%M %D, %Y') AS Date
FROM users AS u
INNER JOIN messages AS m
USING (user_id)
WHERE u.username = 'funny man';
```

| subject                          | Date            |
|----------------------------------|-----------------|
| Database Design                  | April 4th, 2011 |
| Database Design                  | April 4th, 2011 |
| Chicago Bulls                    | April 4th, 2011 |
| Dynamic HTML using PHP, clearer? | April 4th, 2011 |

图7-4 一个稍微复杂点的内联结版本，用到了users表和messages表

这个联结同样使用了两个表：用户表和消息表。连接两个表的列是user\_id，所以把它放在USING子句中。WHERE条件确定目标用户，DATE\_FORMAT()函数用于格式化date\_entered的值。

(4) 找出5个有最新发布消息的讨论（参见图7-5）。

```
SELECT f.name FROM forums AS f
INNER JOIN messages AS m
USING (forum_id)
ORDER BY m.date_entered DESC LIMIT 5;
```

| name     |
|----------|
| Kindling |
| HTML     |
| HTML     |
| Kindling |
| PHP      |

图7-5 这个联结应用了ORDER BY子句和LIMIT子句，它返回有5个最新消息的讨论

由于只需要返回讨论名称，所以此查询仅选择了一列。这个联结就通过forum\_id联结了讨论表和消息表。此时查询将返回讨论表中所有匹配的信息。而结果将按date\_entered列倒序排列，并只显示前5条记录。

### ✓ 提示

- 内联结也可以不使用正式的短语INNER JOIN编写。要这样做，表名之间要以逗号隔开并将ON或者USING子句改为WHERE子句：

```
SELECT m.message_id, m.subject, f.name FROM messages AS m, forums AS f WHERE m.forumid = f.forum_id
AND f.name = 'MySQL'
```

- 不包含联结子句(ON或者USING)或WHERE子句(例如, SELECT \* FROM urls INNER JOIN url\_associations)的联结被称为全联结并且会返回两个表中所有的记录。这种结构的返回结果会是一个冗长的大表。
- 如果内联结中引用的列值为NULL，则永远不会有返回值，因为没有值会匹配NULL，包括NULL。
- MySQL支持的联结类型与SQL标准略有不同。例如，SQL支持的CROSS JOIN和INNER JOIN是两种不同的联结，而是在MySQL中它们却是相同的。

## 7.1.2 外联结

7

外联结不同于内联结(在两个表之间返回匹配记录)，它将会返回两个表都匹配的记录和不匹配的记录。换句话说，内部联结是排他的，但外部联结是包含的。外联结有三个子类型：左联结、右联结、全联结，左联结是目前为止最重要的类型。下面是左联结的一个例子：

```
SELECT f.*, m.subject FROM forums AS f
LEFT JOIN messages AS m
ON f.forum_id = m.forum_id
```

左联结要考虑的最重要的内容是首先指定哪个表。在这个例子中，如果有匹配项，则返回所有的讨论记录和所有的消息的信息。如果讨论行没有匹配的消息记录，那么将为选择的消息列返回NULL值(参见图7-6)。

| forum_id | name         | subject                                 |
|----------|--------------|-----------------------------------------|
| 5        | CSS          | CSS Resources                           |
| 5        | CSS          | CSS Resources                           |
| 4        | HTML         | HTML vs. XHTML                          |
| 4        | HTML         | HTML vs. XHTML                          |
| 6        | Kindling     | Why?                                    |
| 6        | Kindling     | Why? Why? Why?                          |
| 6        | Kindling     | Because                                 |
| 7        | Modern Dance | NULL                                    |
| 1        | MySQL        | Question about normalization.           |
| 1        | MySQL        | Database Design                         |
| 1        | MySQL        | Database Design                         |
| 1        | MySQL        | PHP Errors                              |
| 2        | PHP          | Dynamic HTML using PHP                  |
| 2        | PHP          | Dynamic HTML using PHP                  |
| 2        | PHP          | Dynamic HTML using PHP, still not clear |
| 2        | PHP          | Dynamic HTML using PHP, clearer?        |
| 3        | Sports       | Chicago Bulls                           |
| 3        | Sports       | Chicago Bulls                           |

22 rows in set <0.00 sec>

mysql> -

图7-6 外联结返回第一个表中的所有记录，第二个表中不匹配的记录会以NULL值代替

在内联结和外联结中，如果将在相等性比较中使用的两个表中的列具有相同的名称，则可以使用**USING**简化查询：

```
SELECT f.*, m.subject FROM forums AS f
LEFT JOIN messages AS m
USING (forum_id)
```

右联结与左联结相反：它将返回右表中所有记录和左表中的匹配项。这个查询与上面的相等：

```
SELECT f.*, m.subject FROM messages AS m
RIGHT JOIN forums AS f
USING (forum_id)
```

一般来讲，最好使用左联结，而不是右联结（其实有一种就足够了）。

全联结是左联结和右连接的组合。换句话说，两个表中所有匹配的记录都将被返回，包括左表中不匹配右表的所有记录，以及右表中不匹配左表的所有记录。MySQL不直接支持全连接，但是你可以使用左联结、右联结加UNION语句来实现这个功能。全联结不常用，可以查看MySQL手册了解全联结和UNION的相关信息。

### 使用外联结

- (1) 连接到MySQL并选择forum数据库。
- (2) 检索所有的用户名及用户发布的所有消息ID（参见图7-7）。

```
SELECT u.username, m.message_id
FROM users AS u
LEFT JOIN messages AS m
USING (user_id);
```

| username  | message_id |
|-----------|------------|
| finchy    | NULL       |
| funny man | 2          |
| funny man | 3          |
| funny man | 9          |
| funny man | 19         |
| Gareth    | 4          |
| Gareth    | 5          |
| Gareth    | 7          |
| Gareth    | 11         |
| Gareth    | 13         |
| Gareth    | 16         |
| Gareth    | 18         |
| tim       | 15         |
| tim       | 20         |
| troutster | 1          |
| troutster | 6          |
| troutster | 8          |
| troutster | 10         |
| troutster | 12         |
| troutster | 14         |
| troutster | 17         |
| troutster | 21         |

图7-7 左联结返回所有的用户，所有的消息ID。如果某个用户没有发布过消息（如顶部的finchy），消息ID的值将会是NULL

如果你运行了一个与此类似的内联结，没有发布过消息的用户将不会被列出（参见图7-8）。因此，

需要使用外联结来包括所有的用户。注意，完全包含的表（在这里是用户表），必须是左联结中列出的第一个表。

```
mysql> SELECT u.username, m.message_id
-> FROM users AS u
-> INNER JOIN messages AS m
-> USING (user_id);
+-----+-----+
| username | message_id |
+-----+-----+
funny man	2
funny man	3
funny man	9
funny man	19
Gareth	4
Gareth	5
Gareth	7
Gareth	11
Gareth	13
Gareth	16
Gareth	18
tim	15
tim	20
troutster	1
troutster	6
troutster	8
troutster	10
troutster	12
troutster	14
troutster	17
troutster	21
+-----+-----+
21 rows in set (0.00 sec)

mysql>
```

7

图7-8 这个内联结将不会返回任何没有发布过消息的用户（参见图7-7的顶部的finchy）

(3) 按提交的日期顺序检索所有讨论名称和讨论的每个消息的发布日期（参见图7-9）。

```
SELECT f.name,
DATE_FORMAT(m.date_entered, '%M %D, Y') AS Date
FROM forums AS f
LEFT JOIN messages AS m
USING (forum_id)
ORDER BY date_entered DESC;
```

```
mysql> SELECT f.name,
-> DATE_FORMAT(m.date_entered, '%M %D, %Y') AS Date
-> FROM forums AS f
-> LEFT JOIN messages AS m
-> USING (forum_id)
-> ORDER BY date_entered DESC;
+-----+-----+
| name | Date |
+-----+-----+
CSS	April 4th, 2011
CSS	April 4th, 2011
HTML	April 4th, 2011
HTML	April 4th, 2011
Kindling	April 4th, 2011
Kindling	April 4th, 2011
Kindling	April 4th, 2011
PHP	April 4th, 2011
Sports	April 4th, 2011
Sports	April 4th, 2011
MySQL	April 4th, 2011
PHP	April 4th, 2011
Modern Dance	NULL
+-----+-----+
22 rows in set (0.00 sec)

mysql>
```

图7-9 左外联结返回所有的讨论名称讨论的每个消息的发布日期

这是步骤(2)中联结的一个变体，这次交换了forums表和users表。

### 执行自联结

SQL可以执行自联结：让表自己联结自己。例如，消息表中parent\_id列指明哪些消息是另外一些消息的回复。要获取一个单层的记录，SELECT查询必须使用消息表联结自己，使parent\_id等于message\_id。

这听起来有些让人困惑，也不太可能实现，但是事实并非如此。自联结的诀窍就是，将对同一表的两个引用看作对两个不同表的引用。为了实现它，要为每个表的引用指定不同的别名。上面描述的例子可以写成：

```
SELECT m1.subject, m2.subject AS Reply FROM messages AS m1 LEFT JOIN messages AS m2 ON m1.message_id=m2.parent_id WHERE m1.parent_id=0
```

该查询首先选择第一个消息表m1中所有parent\_id为0的根级别的消息，然后将这些记录外联结到第二个消息表的实例m2。运行此查询，你会看到消息的主题及回复该主题的消息都被列出了。

自联结并不十分常用，但有时候可以比大多数其他解决方案更好。

#### ✓ 提示

- 可以使用任何列作为条件创建联结，而不只限于主键和外键（尽管最常见）。
- 你可以通过`database.table.column`语法跨库执行联结，条件是所有数据库都在同一服务器上（不能跨网络），并且使用联结的用户需要有权限访问所有涉及的表。
- 左联结中的`OUTER`关键词是可选的（经常被省略）。正式情况下，你可以这样写：

```
SELECT f.name, DATE_FORMAT (m.date-entered, '%M %D, Y') AS Date FROM forums AS f LEFT OUTER JOIN messages AS m USING (forum_id) ORDER BY date_entered DESC;
```

### 7.1.3 联结三个或更多表

联结是一个比较复杂但又非常重要的概念，通过上面的学习，我希望你能够对它有一个比较好的理解。你应该已经很熟悉了，有两个以上的联结方式可以使用：在框注“执行自联结”中讨论的自联结和三个或更多表的联结。

如果要联结三个或更多的表，记住两个表之间的连接会创建一张虚拟结果表，这非常有用。当你添加第三个表时，联结就会在初始的虚拟表和第三个引用表之间建立（参见图7-10）。三个表的联结的语法如下所示：

```
SELECT what_columns FROM tableA JOIN_TYPE tableB JOIN_CLAUSE JOIN_TYPE tableC JOIN_CLAUSE
```

两次联结的类型不必是相同的（一个可以是内联结，另外一个可以是外联结），并且联结的子句也可以完全不同。你还可以在语句的结尾添加`WHERE`、`ORDER BY`和`LIMIT`子句。简而言之，执行超过两个表的联结就是按照需要继续添加`JOIN_TYPE tableX JOIN_CLAUSE`部分。

当联结三个或更多的表时，你可能会遇到以下三个问题。首先是一个简单的语法错误，尤其是当你使用小括号分离子句时。第二个是一个有歧义的列错误，这是在任何联结类型中都比较常见的。第

三个可能的问题是返回缺失结果。如果你碰到了这些问题，简化联结到只有两个表以确认结果，然后尝试添加附加的联结子句来找到问题的所在。

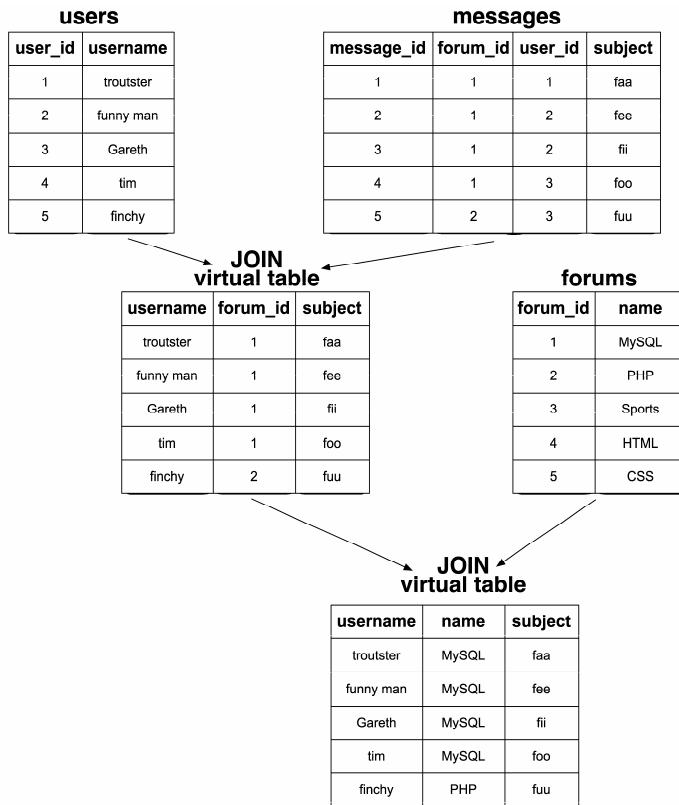


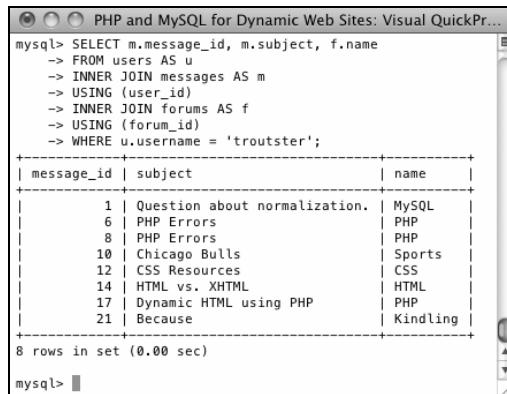
图7-10 跨三个表联结的工作方式：首先创建一个虚拟的结果表，然后将第三个表联结到那个虚拟表

### 在三个或更多的表上使用联结

- (1) 连接到MySQL并选择论坛数据库。
- (2) 检索troutster用户发布的所有消息的消息ID、主题和讨论名（参见图7-11）。

```
SELECT m.message_id, m.subject, f.name
FROM users AS u
INNER JOIN messages AS m
USING (user_id)
INNER JOIN forums AS f
USING (forum_id)
WHERE u.username = 'troutster';
```

这个联结类似于本章较早的一个例子，但是这里更进一步，合并了三个表。



```
mysql> SELECT m.message_id, m.subject, f.name
-> FROM users AS u
-> INNER JOIN messages AS m
-> USING (user_id)
-> INNER JOIN forums AS f
-> USING (forum_id)
-> WHERE u.username = 'troutster';
+-----+-----+-----+
| message_id | subject | name |
+-----+-----+-----+
1	Question about normalization.	MySQL
6	PHP Errors	PHP
8	PHP Errors	PHP
10	Chicago Bulls	Sports
12	CSS Resources	CSS
14	HTML vs. XHTML	HTML
17	Dynamic HTML using PHP	PHP
21	Because	Kindling
+-----+-----+-----+
8 rows in set (0.00 sec)

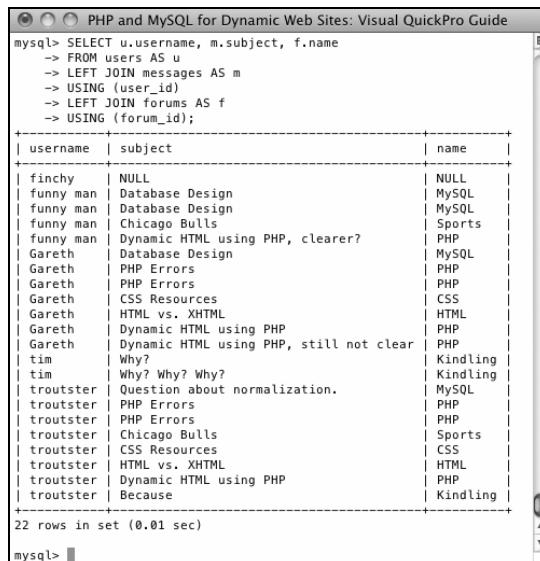
mysql>
```

图7-11 跨三个表的内联结

(3) 检索每位用户的用户名、消息主题和讨论名（参见图7-12）。

```
SELECT u.username, m.subject, f.name
FROM users AS u
LEFT JOIN messages AS m
USING (user_id)
LEFT JOIN forums AS f
USING (forum_id);
```

这里执行了两个外联结，而步骤(2)中的查询执行了两个内联结。图7-10直观地显示了查询背后的过程。



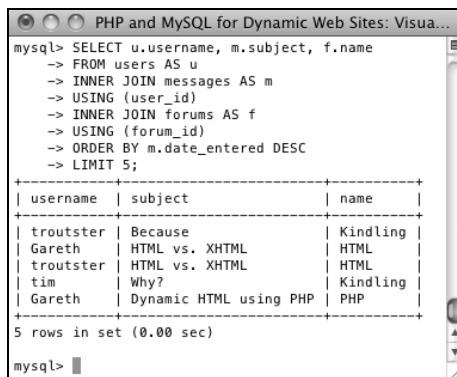
```
mysql> SELECT u.username, m.subject, f.name
-> FROM users AS u
-> LEFT JOIN messages AS m
-> USING (user_id)
-> LEFT JOIN forums AS f
-> USING (forum_id);
+-----+-----+-----+
| username | subject | name |
+-----+-----+-----+
finchy	NULL	NULL
funny man	Database Design	MySQL
funny man	Database Design	MySQL
funny man	Chicago Bulls	Sports
funny man	Dynamic HTML using PHP, clearer?	PHP
Gareth	Database Design	MySQL
Gareth	PHP Errors	PHP
Gareth	PHP Errors	PHP
Gareth	CSS Resources	CSS
Gareth	HTML vs. XHTML	HTML
Gareth	Dynamic HTML using PHP	PHP
Gareth	Dynamic HTML using PHP, still not clear	PHP
tim	Why?	Kindling
tim	Why? Why? Why?	Kindling
troutster	Question about normalization.	MySQL
troutster	PHP Errors	PHP
troutster	PHP Errors	PHP
troutster	Chicago Bulls	Sports
troutster	CSS Resources	CSS
troutster	HTML vs. XHTML	HTML
troutster	Dynamic HTML using PHP	PHP
troutster	Because	Kindling
+-----+-----+-----+
22 rows in set (0.01 sec)

mysql>
```

图7-12 左联结返回所有的用户，所有发布的消息主题，以及所有的讨论名。如果用户没有发布过消息（如顶部的finchy），他的主题和讨论名的值将会是NULL

(4) 查找最近发表过5个消息的用户，同时选取消息主题和讨论名（参见图7-13）。

```
SELECT u.username, m.subject, f.name
FROM users AS u
INNER JOIN messages AS m
USING (user_id)
INNER JOIN forums AS f
USING (forum_id)
ORDER BY m.date_entered DESC
LIMIT 5;
```



The screenshot shows a Windows command-line window titled "PHP and MySQL for Dynamic Web Sites: Visua...". Inside, a MySQL session is running the provided SQL query. The output shows five rows of data from three tables: users, messages, and forums, ordered by date entered in descending order, and limited to the top 5 rows. The columns are username, subject, and name.

| username  | subject                | name     |
|-----------|------------------------|----------|
| troutster | Because                | Kindling |
| Gareth    | HTML vs. XHTML         | HTML     |
| troutster | HTML vs. XHTML         | HTML     |
| tim       | Why?                   | Kindling |
| Gareth    | Dynamic HTML using PHP | PHP      |

5 rows in set (0.00 sec)

7

图7-13 内联结通过应用ORDER BY和LIMIT子句返回所有三个表的值

要获取用户名、消息主题和讨论名，需要联结所有三个表。因为仅查找有发布内容的用户，所以使用内联结。两个联结的结果将是每个用户的用户名和用户在所有讨论中发布的所有信息。然后这一结果按消息发布的日期（date\_entered列）排序，并且仅限于前5条记录。

## 7.2 分组选定的结果

第5章讨论和展示了一类可以用于MySQL的函数。另外一类是分组或聚合函数用于更复杂的查询（参见表7-1）。

表7-1 分组函数

| 函 数            | 返 回 值      |
|----------------|------------|
| AVG()          | 列中所有数值的平均值 |
| COUNT()        | 列中所有的值的个数  |
| GROUP_CONCAT() | 列中所有的值的连结  |
| MAX()          | 列中所有小值的最大值 |
| MIN()          | 列中所有的值的最小值 |
| SUM()          | 列中所有的值的合计值 |

然而，在第5章涵盖的大多数函数只是一次操作某行的一个值（例如，格式化日期列中的值），分组函数返回的是一个列的多个行的值。例如，要获取银行数据库的平均账户余额，需要运行下面这个

查询(参见图7-14):

```
SELECT AVG(balance) FROM accounts
```

```
mysql> SELECT AVG(balance) FROM accounts;
+-----+
| AVG(balance) |
+-----+
| 305225.823333 |
+-----+
1 row in set <0.03 sec>

mysql>
```

图7-14 AVG()函数用于查找所有账户余额的平均值

要找出最小和最大的账户余额，使用以下查询(参见图7-15):

```
SELECT MAX(balance), MIN(balance) FROM accounts
```

```
mysql> SELECT MAX(balance), MIN(balance)
 -> FROM accounts;
+-----+-----+
| MAX(balance) | MIN(balance) |
+-----+-----+
| 909325.24 | 892.00 |
+-----+-----+
1 row in set <0.00 sec>

mysql>
```

图7-15 MAX()和MIN()函数返回表中的最大和最小账户值

要统计表(或结果集)的记录数目，为所有列或所有不为空的列使用COUNT()函数:

```
SELECT COUNT(*) FROM accounts
```

AVG()、COUNT()、SUM()函数也可以使用DISTINCT关键字，使聚合只应用于不重复的值。例如，SELECT COUNT(customer\_id) FROM accounts将返回账户的数目，但是SELECT COUNT(DISTINCT customer\_ID) FROM accounts将会返回有账户的客户数目(参见图7-16)。

```
mysql> SELECT COUNT(customer_id)
 -> FROM accounts;
+-----+
| COUNT(customer_id) |
+-----+
| 4 |
+-----+
1 row in set <0.00 sec>

mysql> SELECT COUNT(DISTINCT customer_id)
 -> FROM accounts;
+-----+
| COUNT(DISTINCT customer_id) |
+-----+
| 3 |
+-----+
1 row in set <0.00 sec>

mysql>
```

图7-16 COUNT()函数，带或者不带DISTINCT关键字，计算记录集中记录的数目

聚合函数单独使用将返回单个的值（如图7-14、图7-15和图7-16所示）。当聚合函数与GROUP BY子句同时使用时，将会为结果集中的每一行返回一个聚合值（参见图7-17）：

```
SELECT AVG(balance), customer_id FROM accounts GROUP BY customer_id
```

你可以为GROUP BY使用WHERE、ORDER BY和LIMIT的组合，例如下面的查询：

```
SELECT what columns FROM table
WHERE condition GROUP BY column
ORDER BY column LIMIT x, y
```

GROUP BY子句也可以用于联结。请记住，联结返回的是一个新的虚拟的数据表，所以该虚拟表上可以应用任何分组。

| AVGBalance    | customer_id |
|---------------|-------------|
| 5460.230000   | 1           |
| 461391.195000 | 2           |
| 892.000000    | 3           |

3 rows in set (0.00 sec)

7

图7-17 与聚合函数一同使用GROUP BY子句来分组聚合结果

### 分组数据

- (1) 连接到MySQL并选择银行数据库。
- (2) 统计注册的用户数（参见图7-18）。

```
SELECT COUNT(*) FROM customers;
```

| COUNT(*) |
|----------|
| 3        |

1 row in set (0.03 sec)

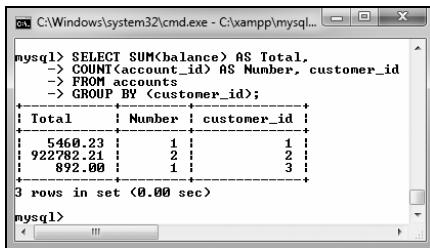
图7-18 这个聚合查询计算了customers表中记录的数目

COUNT()也许是最受欢迎的分组函数。它可以快速统计记录数，如这里的customers表的记录数。COUNT()函数可以用于任何有值的列，例如（即，所有列）或主键customer\_id。

注意，并不是所有使用聚合函数的查询都必须含有GROUP BY子句。

- (3) 查找所有客户的账户总余额，并在处理过程中统计账户的数目（参见图7-19）。

```
SELECT SUM(balance) AS Total,
COUNT(account_id) AS Number, customer_id
FROM accounts
GROUP BY (customer_id);
```



```
mysql> SELECT SUM(balance) AS Total,
 -> COUNT(account_id) AS Number, customer_id
 -> FROM accounts
 -> GROUP BY customer_id;
+-----+-----+-----+
| Total | Number | customer_id |
+-----+-----+-----+
5460.23	1	1
922782.21	2	2
892.00	1	3
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

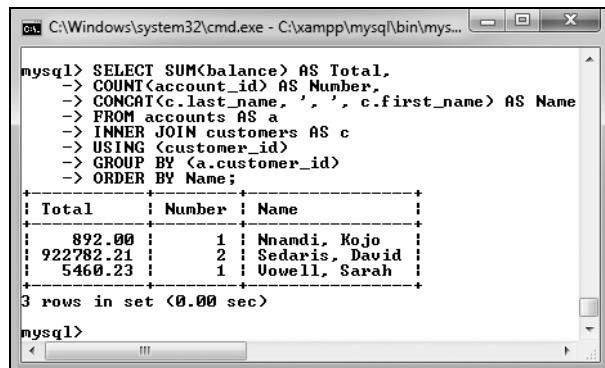
图7-19 在这个过程中，GROUP BY查询通过customer\_id聚合所有账户，返回每个客户账户的总余额和客户的账户数目

这个查询是第(2)步中那个查询的扩展，但它统计每个客户的对应的账户数目以及账户的总余额，而不只是统计客户。

图7-20 在这个过程中，GROUP BY查询通过customer\_id聚合所有的账户，返回每个客户账户的总余额和客户的账户数目

(4) 重复步骤(3)的查询，选择客户的名字代替他们的ID（参见图7-20）。

```
SELECT SUM(balance) AS Total,
COUNT(account_id) AS Number,
CONCAT(c.last_name, ' ', c.first_name) AS Name
FROM accounts AS a
INNER JOIN customers AS c
USING (customer_id)
GROUP BY (a.customer_id)
ORDER BY Name;
```



```
mysql> SELECT SUM(balance) AS Total,
 -> COUNT(account_id) AS Number,
 -> CONCAT(c.last_name, ' ', c.first_name) AS Name
 -> FROM accounts AS a
 -> INNER JOIN customers AS c
 -> USING (customer_id)
 -> GROUP BY (a.customer_id)
 -> ORDER BY Name;
+-----+-----+-----+
| Total | Number | Name |
+-----+-----+-----+
892.00	1	Nnandi, Kojo
922782.21	2	Sedaris, David
5460.23	1	Uowell, Sarah
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图7-20 GROUP BY查询就像图7-19，但是返回的是客户端名字，并按照名字排序结果（这需要一个联结）

要检索客户的名字来代替他的ID，需要用到联结：INNER JOIN customers USING (customer\_id)。下一步，添加别名用于简化引用，并且修改GROUP BY子句来指定分组应用于那个customer\_id列。由于联结，客户的名字可以以客户的名和姓、加逗号和空格的串接形式选出来。最后，结果按照客户名字存储（注意，ORDER BY子句中使用了另外一个别名的引用）。

请记住，如果你使用外连接，而不是内部联接，就可以检索没有账户余额的客户。

(5) 将每位客户的余额联结为单独的字符串（参见图7-21）。

```
SELECT GROUP_CONCAT(balance),
CONCAT(c.last_name, ', ', c.first_name) AS Name
FROM accounts AS a
INNER JOIN customers AS c
USING (customer_id)
GROUP BY (a.customer_id)
ORDER BY Name;
```

GROUP\_CONCAT()函数是一个常常被忽视的有用的聚合工具。你可以在图上看到，默认情况下这个函数使用逗号分隔每个值的连接。

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql...'. Inside, a MySQL session is running. The query is:

```
mysql> SELECT GROUP_CONCAT(balance),
-> CONCAT(c.last_name, ', ', c.first_name) AS Name
-> FROM accounts AS a
-> INNER JOIN customers AS c
-> USING (customer_id)
-> GROUP BY (a.customer_id)
-> ORDER BY Name;
```

The results are displayed in a table:

| GROUP_CONCAT(balance) | Name           |
|-----------------------|----------------|
| 892.00                | Mnandi, Kojo   |
| 13456.97,909325.24    | Sedaris, David |
| 5460.23               | Uowell, Sarah  |

3 rows in set (0.02 sec)

图7-21 图7-20中查询的一个变种，这个查询获取每个用户的所有账户余额之和

### ✓ 提示

- NULL是一个特殊的值，有趣的是GROUP BY将会NULL值分到同一组，因为他们有相同的空值。
- 你需要谨慎使用COUNT()函数，GROUP BY将把NULL值分组在一起是有趣的，因为它们具有相同的“非”值。
- 花一些时间弄清楚GROUP BY子句和这里列出的函数，无论何时，只要语法出错，MySQL都会报告一个错误。在MySQL客户端或phpMyAdmin中实验，确定你可能会通过PHP脚本运行的任何脚本的确切语法。
- 另一个相关的子句是HAVING，它类似于应用于一个组的WHERE条件。
- 不能将SUM()和AVG()直接应用于日期或时间值。你需要将日期和时间值转换为秒，执行SUM()和AVG()函数，并且然后将这个值转换回日期和时间。

## 7.3 高级选择

本章的前两节提出了从复杂结构中选取数据的更高级的方法。但是，即使使用了聚合函数，数据的选择也是非常简单的。然而，有时候你需要有条件地选择数据，就像查询中使用if-else子句。所幸，MySQL提供了控制流和高级比较函数，可以实现上述功能。

首先介绍GREATEST(), 它返回列表中的最大值(参见图7-22):

```
SELECT GREATEST(col1, col2) FROM table
SELECT GREATEST(235, 1209, 59)
```

```
mysql> SELECT GREATEST(235, 1209, 59);
+-----+
| GREATEST(235, 1209, 59) |
+-----+
| 1209 |
+-----+
1 row in set (0.00 sec)

mysql>
```

图7-22 GREATEST()函数返回指定列表中的最大值

LEAST()返回列表中的最小值:

```
SELECT LEAST(col1, col2) FROM table
SELECT LEAST(235, 1209, 59)
```

注意,与聚合函数应用于多行的同一列的值列表不同,比较和控制流功能适用于同一行的多列(或值列表)。

另一个有用的比较函数是COALESCE(), 返回列表中的第一个非NULL值:

```
SELECT COALESCE(col1, col2) FROM table
```

如果列表中没有有值的项, 函数将返回NULL(你会在后面看到一个示例)。

而COALESCE()只返回第一个非NULL值, 你可以使用IF()根据条件返回任何值:

```
SELECT IF (condition, return_if_true, return_if_false)
```

如果条件为真, 函数的第二个参数将被返回, 否则返回第三个参数。举个例子, 假设一个表在性别列中存储了M或F值, 查询可以选取男或女来代替M或F(参见图7-23):

```
SELECT IF(gender='M', 'Male', 'Female') FROM people;
```

```
mysql> SELECT gender FROM people;
+-----+
| gender |
+-----+
| M |
| F |
+-----+
2 rows in set (0.00 sec)

mysql> SELECT IF(gender='M', 'Male', 'Female')
 FROM people;
+-----+
| IF(gender='M', 'Male', 'Female') |
+-----+
| Male |
| Female |
+-----+
2 rows in set (0.00 sec)

mysql>
```

图7-23 IF()函数可以根据条件指定返回的值

这些函数的返回值也可以用于其他查询：

```
INSERT INTO people (gender) VALUES (IF(something='Male', 'M', 'F'))
```

CASE()函数是一个比较复杂的工具，可以以多种方式使用。第一种方法是像PHP的switch条件句一样使用：

```
SELECT CASE col1 WHEN value1 THEN return_this ELSE return_that END FROM table
```

性别的例子可以改写为：

```
SELECT CASE gender WHEN 'M' THEN 'Male' ELSE 'Female' END FROM people
```

CASE()函数可以附加WHEN子句，ELSE也总是可选的：

```
SELECT CASE gender WHEN 'M' THEN 'Male' WHEN 'F' THEN 'FEMALE' END FROM people
```

如果简单的相等性测试无法满足需要，还可以将条件写入CASE()中（参见图7-24）：

```
SELECT message_id, CASE WHEN date_entered > NOW() THEN 'Future' ELSE 'PAST' END AS Posted FROM messages
```

| message_id | Posted |
|------------|--------|
| 1          | PAST   |
| 2          | PAST   |
| 3          | PAST   |
| 4          | PAST   |
| 5          | PAST   |
| 6          | PAST   |
| 7          | PAST   |
| 8          | PAST   |
| 9          | PAST   |
| 10         | PAST   |
| 11         | PAST   |

图7-24 CASE()可以向IF()那样用于自定义返回的值

同样，你可以根据需要添加多个WHEN...THEN子句，如果不需要也可以省略ELSE。

为了练习使用这些函数，接下来在论坛数据中运行一些更多的查询（注意，它们可能会有一点点复杂）。

### 执行高级选择

(1) 连接到MySQL并选择论坛数据库。

(2) 为每一个讨论找到最近发布的帖子的日期和时间，如果不存在帖子则返回N/A（参见图7-25）。

```
SELECT f.name,
COALESCE(MAX(m.date_entered), 'N/A') AS last_post
FROM forums AS f
LEFT JOIN messages AS m
USING (forum_id)
GROUP BY (m.forum_id)
ORDER BY m.date_entered DESC;
```

首先，为了在讨论中找到讨论名和最新的帖子的日期，必须使用联结。具体来说，是一个外联结，

因为可能有的讨论没有帖子。要找到每个讨论的最新帖子，需要对date\_entered列使用聚合函数MAX()，结果需要按照forum\_id分组（因此，MAX()被应用于每个讨论的内部的帖子）。

```
mysql> SELECT f.name,
--> COALESCE(MAX(m.date_entered), 'N/A') AS last_post
--> FROM forums AS f
--> LEFT JOIN messages AS m
--> USING (forum_id)
--> GROUP BY (m.forum_id)
--> ORDER BY m.date_entered DESC;
+-----+-----+
| name | last_post |
+-----+-----+
CSS	2011-04-04 01:37:46
Kindling	2011-04-04 01:37:46
Sports	2011-04-04 01:37:46
HTML	2011-04-04 01:37:46
MySQL	2011-04-04 01:33:43
PHP	2011-04-04 01:37:46
Modern Dance	N/A
+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

图7-25 COALESCE()函数用于将NULL值转换为字符串N/A（见最后一条记录）

由于此时还没有调用COALESCE()函数，因此所有没有任何消息的讨论会返回NULL。最后一步就是应用COALESCE()，所以在MAX(m.date\_entered)为NULL的时候可以返回字符串N/A。

(3) 如果一个消息是另外一个消息的回复，就将它的主题附加一个字符串REPLY（参见图7-26）。

```
SELECT message_id,
CASE parent_id WHEN 0 THEN subject
ELSE CONCAT(subject, ' (Reply)') END AS subject
FROM messages;
```

```
mysql> SELECT message_id,
--> CASE parent_id WHEN 0 THEN subject
--> ELSE CONCAT(subject, ' (Reply)') END AS subject
--> FROM messages;
+-----+-----+
| message_id | subject |
+-----+-----+
1	Question about normalization.
2	Database Design
3	Database Design (Reply)
4	Database Design
5	PHP Errors
6	PHP Errors (Reply)
7	PHP Errors (Reply)
8	PHP Errors (Reply)
9	Chicago Bulls
10	Chicago Bulls (Reply)
11	CSS Resources
12	CSS Resources (Reply)
13	HTML vs. XHTML
14	HTML vs. XHTML (Reply)
15	Why?
16	Dynamic HTML using PHP
17	Dynamic HTML using PHP (Reply)
18	Dynamic HTML using PHP, still not clear (Reply)
19	Dynamic HTML using PHP, clearer? (Reply)
20	Why? Why? Why? (Reply)
21	Because (Reply)
+-----+-----+
21 rows in set (0.00 sec)

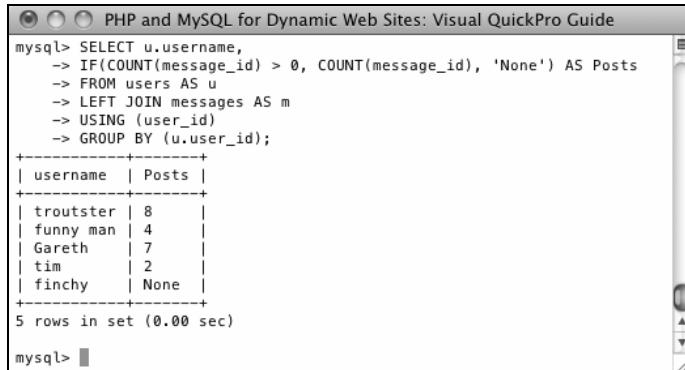
mysql>
```

图7-26 这里，字符串Reply被添加到所有回复其他消息的主题上

messages表中parent\_id不为0的记录都是现有消息的回复。对于这些消息，在它们的主题结尾附加REPLAY以指明它们。为了实现这个目标，如果parent\_id的值等于0，CASE语句返回的只能是主题。如果parent\_id的值不为0（再次感谢CASE），REPLY就会被连接到这个主题。整个构造分配了subject别名，所以它仍然在原来的“subject”标题下返回。

(4) 为每一位用户查找他们发布的消息数目，将零转换为字符串None（参见图7-27）。

```
SELECT u.username,
IF(COUNT(message_id) > 0, COUNT(message_id), 'None') AS Posts
FROM users AS u
LEFT JOIN messages AS m
USING (user_id)
GROUP BY (u.user_id);
```



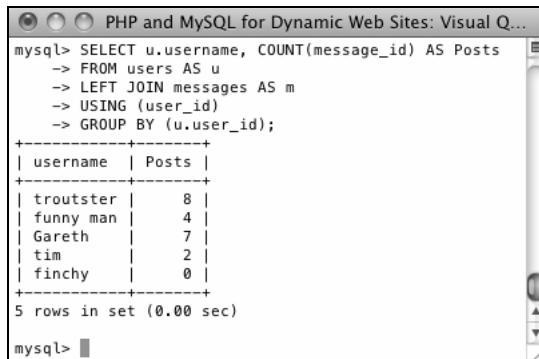
The screenshot shows the MySQL command-line interface with the following output:

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT u.username,
-> IF(COUNT(message_id) > 0, COUNT(message_id), 'None') AS Posts
-> FROM users AS u
-> LEFT JOIN messages AS m
-> USING (user_id)
-> GROUP BY (u.user_id);
+-----+-----+
| username | Posts |
+-----+-----+
troutster	8
funny man	4
Gareth	7
tim	2
finchy	None
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

图7-27 由于调用了IF()，没有发布消息的用户，消息数目会被显示为None

这在某种程度上是对在步骤2中的查询的一种变化。为了获取用户名和发布的消息数，左联结连接了用户和消息。要执行计数，结果必须按users.user\_id分组。如果用户尚未发布内容，这个查询将会返回0（参见图7-28）。要将0转换为字符串None，同时维持非0的计数，应该应用IF()函数。该函数的第一个参数确定了条件：如果计数大于零；第二个参数表示条件为真时，应该返回计数。第三个参数表示条件为假时，应返回字符串None。



The screenshot shows the MySQL command-line interface with the following output:

```
PHP and MySQL for Dynamic Web Sites: Visual Q...
mysql> SELECT u.username, COUNT(message_id) AS Posts
-> FROM users AS u
-> LEFT JOIN messages AS m
-> USING (user_id)
-> GROUP BY (u.user_id);
+-----+-----+
| username | Posts |
+-----+-----+
troutster	8
funny man	4
Gareth	7
tim	2
finchy	0
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

图7-28 不使用IF()查询的结果（相较于图7-27）

✓ 提示

❑ IFNULL()函数有时可以用来代替COALESCE()。它的语法是：

```
IFNULL(value, return_if_null)
```

如果第一个参数是一个指定列，包含NULL值，那么将返回第二个参数。如果参数不包含NULL值，则返回该参数的值。

## 7.4 执行 FULLTEXT 查找

在第5章中，介绍了LIKE关键字，它用于执行稍微简单一些的字符串匹配，比如

```
SELECT * FROM users
WHERE last_name LIKE 'Smith%'
```

这种类型的条件语句足够有效，但它仍然具有极大的局限性。例如，它不允许使用多个单词执行类似于Google的查找。对于这类情况，需要FULLTEXT查找。在接下来的几小节中，你将学习有关FULLTEXT查找的全部所需知识（也会在此过程中学习更多的SQL小技巧）。

### 修 改 表

SQL的ALTER术语主要用于修改现有表的结构。这通常意味着在其中添加、删除或更改列，但是它还包括添加索引。ALTER语句甚至可以用来重命名一个表。ALTER的基本语法如下：

```
ALTER TABLE tablename CLAUSE
```

可以使用的子句非常多，表7-2列出最常见的子句，其中*t*表示表名、*C*表示列名、*i*表示索引名。同样，MySQL手册非常详细地介绍了这个主题。

表7-2 ALTER TABLE子句

| 子 句           | 用 法                                                              | 含 义                |
|---------------|------------------------------------------------------------------|--------------------|
| ADD COLUMN    | ALTER TABLE <i>t</i> ADD COLUMN <i>c</i> <i>TYPE</i>             | 添加新列到表的末尾          |
| CHANGE COLUMN | ALTER TABLE <i>t</i> CHANGE COLUMN <i>c</i> <i>c</i> <i>TYPE</i> | 允许更改列的数据类型和属性      |
| DROP COLUMN   | ALTER TABLE <i>t</i> DROP COLUMN <i>c</i>                        | 从表中删除一列，包括其所有数据    |
| ADD INDEX     | ALTER TABLE <i>t</i> ADD INDEX <i>i</i> ( <i>c</i> )             | 在 <i>c</i> 上添加新的索引 |
| DROP INDEX    | ALTER TABLE <i>t</i> DROP INDEX <i>i</i>                         | 删除现有的索引            |
| RENAME TO     | ALTER TABLE <i>t</i> RENAME TO <i>new_t</i>                      | 更改表的名称             |

你也可以使用ALTER *t* CONVERT TO CHARACTER SET *x* COLLATE *y*来更改表的字符集和校对规则。

### 7.4.1 创建FULLTEXT索引

首先，FULLTEXT搜索需要FULLTEXT索引。在第6章曾介绍过，这种类型的索引，只能在MyISAM表中创建。接下来的例子将使用论坛数据库中的messages表。第一步，为消息正文和主题列添加FULLTEXT索引。如前面框注所讲，要为已经存在的表添加索引需要使用ALTER命令。

### 添加FULLTEXT索引

- (1) 连接到MySQL并选择论坛数据库。
- (2) 确认的messagers表的类型（参见图7-29）。

```
SHOW TABLE STATUS\G
```

```

mysql> SHOW TABLE STATUS\G
***** 1. row *****
 Name: forums
 Engine: InnoDB
 Version: 10
 Row_format: Compact
 Rows: 7
Avg_row_length: 2348
 Data_length: 16384
Max_data_length: 0
 Index_length: 16384
 Data_free: 9437184
Auto_increment: 8
Create_time: 2011-04-03 21:06:05
Update_time: NULL
Check_time: NULL
 Collation: utf8_general_ci
 Checksum: NULL
Create_options:
Comment:
***** 2. row *****
 Name: messages
 Engine: MyISAM
 Version: 10
 Row_format: Dynamic
 Rows: 21
Avg_row_length: 93
 Data_length: 1956
Max_data_length: 281474976710655
 Index_length: 6144
 Data_free: 0
Auto_increment: 22
Create_time: 2011-04-03 21:06:05
Update_time: 2011-04-03 21:37:28
Check_time: NULL
 Collation: utf8_general_ci
 Checksum: NULL
Create_options:
Comment:
***** 3. row *****
 Name: users
 Engine: InnoDB
 Version: 10
 Row_format: Compact
 Rows: 5
Avg_row_length: 3276
 Data_length: 16384
Max_data_length: 0
 Index_length: 49152
 Data_free: 9437184

```

图7-29 使用SHOW TABLE STATUS命令来确认表的类型

SHOW TABLE STATUS查询将返回数据库中表的大量信息，包括表的存储引擎。因为查询会返回很多信息，命令以\G结尾而不是分号。这告诉MySQL客户端，以垂直的列表返回结果而不是一个表格（有时更容易阅读）。如果你使用phpMyAdmin或其他的接口，你可以省略\ G（和可以省略最后的分号情况类似）。

- 如果仅查找messages表的信息，你可以用SHOW TABLE STATUS LIKE 'messages'这个查询。
- (3) 如果messages表不是MyISAM类型，更改它的存储引擎。

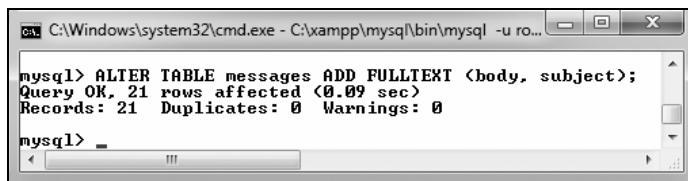
```
ALTER TABLE messages ENGINE = MyISAM;
```

重复一次，这仅在表的类型不正确的情况下才用得到。

- (4) 为messages表添加FULLTEXT索引（参见图7-30）。

```
ALTER TABLE messages ADD FULLTEXT (body, subject);
```

无论索引的类型是什么，添加索引的语法都是ALTER TABLE tablename ADD INDEX\_TYPE index\_name (columns)。索引名是可选的。



```
C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u ro... mysql> ALTER TABLE messages ADD FULLTEXT <body, subject>; Query OK, 21 rows affected (0.09 sec) Records: 21 Duplicates: 0 Warnings: 0 mysql>
```

图7-30 为messages表添加FULLTEXT索引

这里消息正文和主题列都有一个FULLTEXT索引，在本章的稍后将用于FULLTEXT搜索。

#### ✓ 提示

- 在表中插入FULLTEXT索引可能会非常慢，因为这需要的索引非常复杂。
- FULLTEXT搜索可以成功地用于简单的搜索引擎。但是FULLTEXT索引一次只能用于一个表，所以内容跨多个表存储的复杂网站一般使用更加正式的搜索引擎。

### 7.4.2 执行基本的FULLTEXT查找

一旦在列上建立了一个FULLTEXT索引，就可以在WHERE条件语句中使用MATCH...AGAINST表达式开始针对它的查询。

```
SELECT * FROM tablename WHERE MATCH
(columns) AGAINST (terms)
```

就像一个搜索引擎一样，MySQL将会以数学计算得到的相关性的次序返回匹配的行。执行该操作时，会应用以下一些规则：

- 把字符串分解为它们独立的关键字；
- 长度不足4个字符的关键字将被忽略；
- 将会忽略称为停止词（stopword）的非常普遍的单词；
- 如果50%以上的记录与关键字匹配，则不会返回记录。

当用户开始使用FULLTEXT查找并且想知道为什么没有检索到结果时，上述最后一点会使许多用户感到困惑。当你具有一个稀疏填充的表时，将不会有充足的记录让MySQL返回相关的（relevant）结果。

#### 执行FULLTEXT查找

- (1) 连接到MySQL并选择论坛数据库。
- (2) 彻底填充messages表，重点关注添加冗长的消息正文。
- 重申一遍，可以从本书对应的Web站点下载SQL INSERT命令。
- (3) 在单词database上运行一个简单的FULLTEXT查找（参见图7-31）。

```
SELECT subject, body FROM messages
WHERE MATCH (body, subject)
AGAINST('database');
```

这是一个非常简单的示例，只要messages表中至少有一条记录在其正文或主题中具有单词

database，并且这样的记录数在50%以下，它就会返回一些结果。注意：MATCH中引用的列必须与对其建立FULLTEXT索引的那些列相同。在这个示例中，可以使用body，subject或subject，body，但是不能只使用body或者只使用subject（参见图7-32）。

```
mysql> SELECT subject, body FROM messages
-> WHERE MATCH (body, subject)
-> AGAINST('database');
+-----+-----+
| subject | body
+-----+-----+
| Database Design | I'm creating a new database and am having pr
| Database Design | The number of tables your database includes.
| Database Design | Okay, thanks!
+-----+-----+
3 rows in set <0.00 sec>

mysql>
```

图7-31 一个基本的FULLTEXT查找

```
mysql> SELECT subject, body FROM messages
-> WHERE MATCH (subject)
-> AGAINST('database');
ERROR 1191 (HY000): Can't find FULLTEXT index matching the column list
mysql>
```

图7-32 只能在创建了FULLTEXT索引的列或列组合上运行FULLTEXT查询。

利用这个查询，即使body和subject的组合具有FULLTEXT索引，尝试只在subject上运行匹配也将失败

(4) 运行相同的FULLTEXT查找，同时还显示出相关性（参见图7-33）。

```
SELECT subject, body, MATCH (body,
subject) AGAINST('database') AS R
FROM messages WHERE MATCH (body, subject) AGAINST('database')\G
```

```
mysql> SELECT subject, body, MATCH (body,
-> subject) AGAINST('database') AS R
-> FROM messages WHERE MATCH (body, subject) AGAINST('database')\G
***** 1. row *****
subject: Database Design
body: I'm creating a new database and am having problems with the str
R: 2.5440027713775635
***** 2. row *****
subject: Database Design
body: The number of tables your database includes...
R: 2.519484281539917
***** 3. row *****
subject: Database Design
body: Okay, thanks!
R: 1.7514755225860596
3 rows in set <0.00 sec>

mysql>
```

图7-33 也可以选择FULLTEXT查找的相关性。在这个示例中，你将看到与只在主题中包含单词database的记录相比，在主题和正文中都有这个单词的两条记录将具有更高的相关性

如果把相同的MATCH...AGAINST表达式用作选择的值，就会返回实际的相关性。如前所述，要使结果更易于在MySQL客户端中查看，查询结尾可以添加/G，返回的结果将以垂直列表显示。

(5) 使用多个关键字运行FULLTEXT查找（参见图7-34）。

```
SELECT subject, body FROM messages
WHERE MATCH (body, subject)
AGAINST('html xhtml');
```

利用这个查询，如果主题或正文中包含单词database，就会产生一个匹配。如果任何记录同时在主题和正文中包含这个单词，则它将具有更高的等级。

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT subject, body FROM messages
-> WHERE MATCH (body, subject)
-> AGAINST('html xhtml');
+-----+-----+
| subject | body |
+-----+-----+
HTML vs. XHTML	XHTML is a cross between HTML and XML. The differences are largely syntactic. Blah, blah, blah...
HTML vs. XHTML	What are the differences between HTML and XHTML?
Dynamic HTML using PHP	Can I use PHP to dynamically generate HTML on the fly? Thanks...
Dynamic HTML using PHP	You must certainly can.
Dynamic HTML using PHP, still not clear	Um, how?
Dynamic HTML using PHP, clearer?	I think what Larry is trying to say is that you should buy and read his book.
CSS Resources	Read Elizabeth Castro's excellent book on (X)HTML and CSS. Or search Google on "CSS".
+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

图7-34 使用FULLTEXT查找，可以轻松找到包含多个关键字的消息

### ✓ 提示

- 记住，如果FULLTEXT查找没有返回任何记录，那么这意味着没有产生匹配，或者匹配了超过一半的记录。
- 出于简单性考虑，在本节中就像编写简单的SELECT语句一样来编写所有的查询。当然，你可以在联结内或者更复杂的查询内使用FULLTEXT查找。
- MySQL已经定义了数百个停止词。它们是应用程序源代码的一部分。
- 最短的关键字长度（默认为4个字符）是一种配置设置，在MySQL中可以更改它。
- FULLTEXT查找默认情况下不区分大小写。

#### 7.4.3 执行布尔型FULLTEXT查找

基本的FULLTEXT查找是美妙的，但是，可以使用其布尔模式可以完成更复杂的FULLTEXT查找。为了执行该操作，可以在AGAINST子句中添加短语IN BOOLEAN MODE：

```
SELECT * FROM tablename WHERE
MATCH(column) AGAINST('terms' IN BOOLEAN
MODE)
```

布尔模式具有许多运算符（参见表7-3）来规定如何处理每个关键字：

```
SELECT * FROM tablename WHERE
MATCH(column) AGAINST('+database
-mysql' IN BOOLEAN MODE)
```

表7-3 布尔模式运算符

| 运 算 符 | 含 义          |
|-------|--------------|
| +     | 必须存在于每个匹配中   |
| -     | 绝对不会存在于任何匹配中 |
| ~     | 如果存在，则降低一个等级 |
| *     | 通配符          |
| <     | 降低单词的重要性     |
| >     | 提高单词的重要性     |
| " "   | 必须匹配精确的短语    |
| 0     | 创建子表达式       |

在这个示例中，如果找到单词database并且mysql不存在，就会建立一个匹配。另外，也可以使用否定号 (~) 作为减号的更适度的形式，这意味着关键字可以存在于一个匹配中，但是这样的匹配应该被视作不太相关。

通配符 (\*) 可以匹配一个单词的变体，因此，cata\*可以匹配catalog、catalina等。有两个运算符显式指出了哪些关键字更重要 (>)，或者不那么重要 (<)。最后，可以使用双引号来搜寻精确的短语，以及使用括号来建立子表达式。

以下查询将寻找具有短语Web develop的记录，其中单词html是必需的，并且从匹配的相关性中降低了单词JavaScript的重要性：

```
SELECT * FROM tablename WHERE
MATCH(columns) AGAINST('>"Web develop"
+html ~JavaScript' IN BOOLEAN MODE)
```

使用布尔模式时，就FULLTEXT查找的工作方式来说，有几个不同之处：

- 如果关键字前面没有任何运算符，则这个单词是可选的；但是，如果它存在，匹配的等级会更高；
- 即使50%以上的记录与查找匹配，也会返回结果；
- 不会自动按相关性对结果进行排序。

由于最后一点，你或许还希望按相关性对返回的记录进行排序，我将在下面的步骤序列中演示。同样适用于布尔型查找的一个重要规则是，最短的单词长度（默认为4个字符）仍然适用。因此，尝试使用加号 (+php) 来要求更短的单词仍然不会工作。

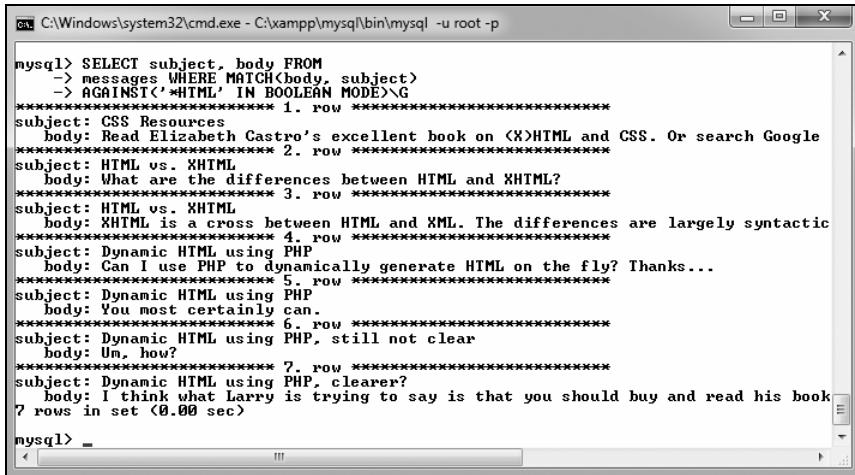
### 执行FULLTEXT布尔型查找

(1) 连接到MySQL并选择论坛数据库。

(2) 运行一个简单的FULLTEXT查找，找出HTML、XHTML或(X)HTML（参见图7-35）。

```
SELECT subject, body FROM
messages WHERE MATCH(body, subject)
AGAINST('*HTML' IN BOOLEAN MODE)\G
```

术语HTML可能以许多格式出现在消息中，包括HTML、XHTML或(X)HTML。这个布尔模式的查询将找出所有这些单词，这是由于使用了通配符 (\*)。



```

C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p

mysql> SELECT subject, body FROM
-> messages WHERE MATCH(`body`, `subject`)
-> AGAINST('`HTML` IN BOOLEAN MODE')\G
***** 1. row *****
subject: CSS Resources
body: Read Elizabeth Castro's excellent book on <X>HTML and CSS. Or search Google
***** 2. row *****
subject: HTML vs. XHTML
body: What are the differences between HTML and XHTML?
***** 3. row *****
subject: HTML vs. XHTML
body: XHTML is a cross between HTML and XML. The differences are largely syntactic
***** 4. row *****
subject: Dynamic HTML using PHP
body: Can I use PHP to dynamically generate HTML on the fly? Thanks...
***** 5. row *****
subject: Dynamic HTML using PHP
body: You most certainly can.
***** 6. row *****
subject: Dynamic HTML using PHP, still not clear
body: Um, how?
***** 7. row *****
subject: Dynamic HTML using PHP, clearer?
body: I think what Larry is trying to say is that you should buy and read his book
7 rows in set <0.00 sec>

mysql> -

```

图7-35 简单的布尔模式FULLTEXT

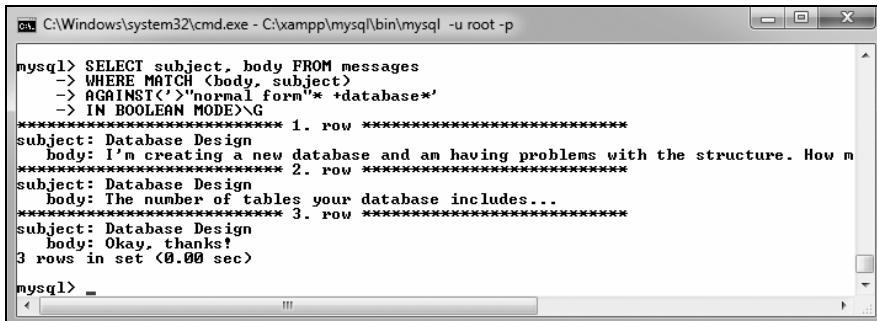
为了使结果更易于查看，我使用了本章前面提到的\G技巧，它告诉MySQL客户端以垂直方式（而不是以水平方式）返回结果。

(3) 找出涉及数据库的匹配，并重点强调范式（参见图7-36）。

```

SELECT subject, body FROM messages
WHERE MATCH (body, subject)
AGAINST('>"normal form" * +database*'
IN BOOLEAN MODE)\G

```



```

C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p

mysql> SELECT subject, body FROM messages
-> WHERE MATCH (`body`, `subject`)
-> AGAINST('>"normal form" * +database*',\G
-> IN BOOLEAN MODE)\G
***** 1. row *****
subject: Database Design
body: I'm creating a new database and am having problems with the structure. How m
***** 2. row *****
subject: Database Design
body: The number of tables your database includes...
***** 3. row *****
subject: Database Design
body: Okay, thanks!
3 rows in set <0.00 sec>

mysql> -

```

图7-36 这个查找会寻找两个不同关键字的变体，其中一个的等级高于另一个

这个查询首先找出其中同时具有database、databases等和normal form、normal forms等的所有记录。database\*名词是必需的（如加号所指示的那样），但是重点强调了范式子句（它前面放置了大于号）。

(4) 重复步骤(2)中的查询，让XHTML更加重要，按照关联性返回结果（参见图7-37）。

```

SELECT subject, body, MATCH(`body`, `subject`)
AGAINST('*HTML >XHTML' IN BOOLEAN MODE) AS R FROM
messages WHERE MATCH(`body`, `subject`)
AGAINST('*HTML >XHTML' IN BOOLEAN MODE) ORDER BY R DESC\G

```

```

C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p

mysql> SELECT subject, body, MATCH(body, subject)
-> AGAINST('<*HTML >XHTML' IN BOOLEAN MODE) AS R FROM
-> messages WHERE MATCH(body, subject)
-> AGAINST('<*HTML >XHTML' IN BOOLEAN MODE) ORDER BY R DESC\G
***** 1. row *****
subject: HTML vs. XHTML
body: What are the differences between HTML and XHTML?
R: 2.5
***** 2. row *****
subject: HTML vs. XHTML
body: XHTML is a cross between HTML and XML. The differences are largely syntax
R: 2.5
***** 3. row *****
subject: CSS Resources
body: Read Elizabeth Castro's excellent book on <X>HTML and CSS. Or search Goog
R: 1
***** 4. row *****
subject: Dynamic HTML using PHP
body: Can I use PHP to dynamically generate HTML on the fly? Thanks...
R: 1
***** 5. row *****
subject: Dynamic HTML using PHP
body: You most certainly can.
R: 1
***** 6. row *****
subject: Dynamic HTML using PHP, still not clear
body: Um, how?
R: 1
***** 7. row *****
subject: Dynamic HTML using PHP, clearer?
body: I think what Larry is trying to say is that you should buy and read his b
R: 1
7 rows in set <0.00 sec>

mysql>

```

图7-37 前面的SELECT查询的修改版本，并且按照关联排序结果

这类似于前面查询，但是现在XHTML给予了更高的权重。这个查询额外选择了计算的关联性，并且会按这个顺序返回结果。

#### ✓ 提示

- MySQL 5.1.7添加了另一种FULLTEXT查找模式：自然语言。如果没有指定其他模式（如布尔模式），这就是默认的模式。
- WITH QUERY EXPANSION将增加返回结果的数量。这种查询执行两次查找并返回一个结果集。它依据在初次查找的最相关的结果中发现的项目进行第二次查找。虽然WITH QUERY EXPANSION查找可以发现未返回的结果，但它也可以返回与原始查找项目完全不相关的结果。

## 7.5 查询优化

一旦你有了一个完整的、填充好的数据库，并且了解通常会在它上面运行哪些查询，采取一些步骤将查询和数据作为一个整体进行优化是非常有益的。这样做将确保MySQL获得最佳性能（因而你的网站也将获得最佳性能）。

后面框注重点强调的键设计的思想已经在正文中建议过了。连同这些技巧及后在介绍的两个简单的技巧可以优化现有的表。提高MySQL性能的方法之一是，不时地运行OPTIMIZE命令。这个查询将去掉表中任何不必要的开销，从而提高交互的速度：

```
OPTIMIZE TABLE tablename
```

在通过ALTER命令改变表或对表进行了大量的删除操作后，在记录之间会留下虚拟的间断，而运行这个命令可以优化数据库。

其次，你可以偶尔使用ANALYZE命令：

```
ANALYZE TABLE tablename
```

执行此命令更新表中的索引，从而改善它们在查询中的使用情况。每当存储在表中的数据发生批量更改的时候（例如，通过UPDATE或INSERT命令）可以执行这个命令。

说到查询，你可能已经意识到，完成同样的目标通常可以采取很多种方法。要找出最有效的方法，理解究竟MySQL将如何运行该查询是有帮助的。这可以通过使用SQL的EXPLAIN关键字来实现。解释查询是一个非常高级的话题，我在这里只介绍基本内容，你可以查看MySQL或搜索网络获取更多的信息。

### 数据库优化

数据库的性能主要依赖于它的结构和索引。创建数据库时尽量：

- 选择最佳的存储引擎；
- 尽量为每列使用最小的数据类型；
- 尽可能定义不可为空（NOT NULL）的列；
- 为主键使用整数；
- 明智而谨慎地定义索引，选择正确的类型应用到恰当的一列或多列；
- 如果可能的话限制索引为固定数目的字符；
- 避免创建太多的索引；
- 确保作为联结基础的列是相同的类型，如果是字符串的话，使用相同的字符集和校对规则。

### 解释查询

(1) 找到一个可能是资源密集型的查询。

包含以下内容的查询是良好的候选：

- 联结两个或多个表；
- 使用分组和聚合函数；
- 有WHERE子句。

例如，在本章前面的这个查询符合这些标准中的两个：

```
SELECT SUM(balance) AS Total,
COUNT(account_id) AS Number,CONCAT(c.last_name, ' ', c.first_name) AS Name
FROM accounts AS a INNER JOIN customers AS c USING (customer_id)
GROUP BY (a.customer_id) ORDER BY Name;
```

(2) 连接到MySQL并选择适用的数据库。

(3) 在数据库中执行查询，在它前面添加EXPLAIN（参见图7-38）。

```
EXPLAIN SELECT SUM(balance) AS Total,
COUNT(account_id) AS Number, CONCAT(c.last_name, ' ', c.first_name) AS Name
FROM accounts AS a INNER JOIN customers AS c USING (customer_id)
GROUP BY (a.customer_id) ORDER BY Name\G
```

如果你正在使用MySQL客户端，你会发现使用\G（而不是分号）结尾会使输出更易读。输出将会是查询中用的所有表的一个信息行。表列出的顺序与MySQL执行查询时访问它们的顺序相同。

后面仔细浏览一遍输出的关键部分，大多数SELECT查询的select\_type值是“SIMPLE”，但如果查

询涉及UNION或子查询值会有所不同（在MySQL手册中查询更多关于UNION或子查询的内容）。

```

 mysql> EXPLAIN SELECT SUM(balance) AS Total,
 -> COUNT(account_id) AS Number, CONCAT(c.last_name, ', ', c.first_name) AS Name
 -> FROM accounts AS a INNER JOIN customers AS c USING (customer_id)
 -> GROUP BY (a.customer_id) ORDER BY Name\G

 id: 1
 select_type: SIMPLE
 table: c
 type: index
 possible_keys: PRIMARY
 key: full_name
 key_len: 184
 ref: NULL
 rows: 3
 Extra: Using index; Using temporary; Using filesort

 id: 2
 select_type: SIMPLE
 table: a
 type: ALL
 possible_keys: customer_id
 key: NULL
 key_len: NULL
 ref: NULL
 rows: 4
 Extra: Using where; Using join buffer
 2 rows in set (0.00 sec)

mysql>

```

7

图7-38 EXPLAIN的输出结果揭示了MySQL将如何处理查询

#### (4) 检查type值。

表7-4列出了不同的type值，从最好到最差。MySQL手册讨论每个细节。但了解通常会看到的eq\_ref会有很大帮助，ALL是最坏的。eq\_ref类型意味着索引被正确使用，并使用了等值比较。

注意，你有时会看到ALL，因为表中的记录很少，在这种情况下，MySQL扫描全表比使用索引更加有效。因为账户表只有4条记录，这大概出现图7-38所示情况的原因。

表7-4 联结类型

| 类 型             |
|-----------------|
| system          |
| const           |
| eq_ref          |
| ref             |
| fulltext        |
| ref_or_null     |
| index_merge     |
| unique_subquery |
| index_subquery  |
| range           |
| index           |
| ALL             |

(5) 检查possible\_keys值。

possible\_keys值表明MySQL中存在可以用于在表中找到相应行的索引。如果值是NULL，表明MySQL认为没有有用的索引。因此，创建该表的恰当列的索引可能会有帮助。

(6) 检查key、key\_len和ref的值。

possible\_keys表示哪些索引可能是有用的，key说明了MySQL实际上会为查询使用哪些索引。偶尔，你会发现这里没有列出possible\_keys，这是正常情况。如果没有使用任何键，那几乎总是表明可以通过添加索引或修改查询来纠正问题。

key\_len值表示MySQL使用的键长度（即大小）。一般来说，这里是越短越好，但是不要太担心它。

ref列表示MySQL使用哪一列与在key列中指定的索引进行比较。

(7) 检查rows的值。

本列提供了MySQL认为表中大约有多少行需要检查。再次强调，越少越好。事实上，联结的效率可以通过所有行的乘积来粗略估计。

在一个联结中，通常需要检查的行数从多到少，如图7-39所示。

(8) 检查Extra值。

最后，此列报告任何有关MySQL将如何执行查询的其他附加信息。在这里不应该出现两个词组是：Using filesort和Using temporary。这都意味着需要额外的步骤来完成查询（例如，GROUP BY子句中经常需要MySQL创建一个临时表）。

如果Extra在Impossible X or No matching Y后面显示了些内容，那就意味着查询中有总是为假或可以被删除的子句。

```

mysql> EXPLAIN SELECT u.username, m.subject, f.name
 > FROM users AS u
 > LEFT JOIN messages AS m
 > USING (user_id)
 > LEFT JOIN forums AS f
 > USING (forum_id)\G

1. row *****
 id: 1
 select_type: SIMPLE
 table: u
 type: index
possible_keys: NULL
 key: username
 key_len: 92
 ref: NULL
 rows: 5
 Extra: Using index

2. row *****
 id: 1
 select_type: SIMPLE
 table: m
 type: ref
possible_keys: user_id
 key: user_id
 key_len: 3
 ref: forum.u.user_id
 rows: 4
 Extra:

3. row *****
 id: 1
 select_type: SIMPLE
 table: f
 type: eq_ref
possible_keys: PRIMARY
 key: PRIMARY
 key_len: 1
 ref: forum.m.forum_id
 rows: 1
 Extra:
3 rows in set (0.00 sec)

mysql>

```

图7-39 另一个查询的解释，这是一个跨3个表的联结

(9)修改表或查询，重复以上步骤！

如果输出指出了查询的问题，你可以做如下的考虑：

- 更改查询的细节；
- 更改表的列属性；
- 添加或修改表的索引。

记住，解释的有效性部分取决于表的行数（如在第(4)步所述，MySQL可能跳过小表的索引）。同时也要理解不是所有的查询都是可以被修复的。有时，简单的SELECT查询（甚至联结）可以得到改善，但是很少有办法改善GROUP BY查询的效率，在聚合数据时要考虑到MySQL要做的方方面面。

#### ✓ 提示

- EXPLAIN EXTENDED命令提供了关于查询的一些细节：

EXPLAIN EXTENDED SELECT...

- 有问题的查询也可以通过启用某些MySQL日志功能发现，但是那需要MySQL服务器的管理员权限。
- 在性能方面，MySQL处理的多个小表要比处理几个大表要好。这就是说，设计规范化的数据库结构应始终是首要目标。
- 对MySQL来说，所谓“大”的数据库是含数以千计的表和数百万条记录的数据库。

## 7.6 执行事务

数据库事务（database transaction）是在单个会话期间运行的一系列查询。例如，你可能插入一条记录到一个表中，插入另一条记录到另一个表中，或许还会运行更新。如果不使用事务，每个独立的查询就会立即生效并且不能撤销。使用事务，就可以设置起点和终点，然后根据需要运行或撤销所有的查询（例如，如果一个查询失败，就可以撤销所有的查询）。

商业交互通常需要事务，甚至像从我的银行账户向你的银行账户转账100美元这样基本的交互也是如此。这看上去似乎是一个简单的过程，但它实际上涉及多个步骤：

- 确认我的账户中有100美元；
- 把我的账户中的钱数减少100美元；
- 验证减少数；
- 把你的账户中的钱数增加100美元；
- 验证你的账户中的钱数增多了。

如果任何一步失败，就希望撤销所有步骤。例如，如果不能把钱存入你的账户，就应该把它返回到我的账户中，直到整个事务可以完成。

为了用MySQL执行事务，必须使用InnoDB表类型（或存储引擎）。为了在MySQL客户端中开始一个新事务，可以输入：

```
START TRANSACTION;
```

一旦事务开始执行，现在就可以运行查询。一旦完成，就可以输入COMMIT执行所有的查询，或者输入ROLLBACK撤销所有查询的作用。

在提交或回滚查询之后，认为事务已经完成，并且MySQL会返回到自动提交（autocommit）模式。这意味着你执行的任何查询将会立即生效。为了开始另一个事务，只需输入`START TRANSACTION`。

知道某些类型的查询不能回滚是重要的。确切地讲，那些用于创建、改变、截短（清空）或删除表或者用于创建或删除数据库的查询不能被撤销。此外，使用这种查询具有提交和结束当前事务的作用。

其次，你需要了解事务是针对每个连接的。因此，通过MySQL客户端连接的一个用户与另一个用户是两个不同的事务，这两者与通过PHP脚本连接的事务也不同。

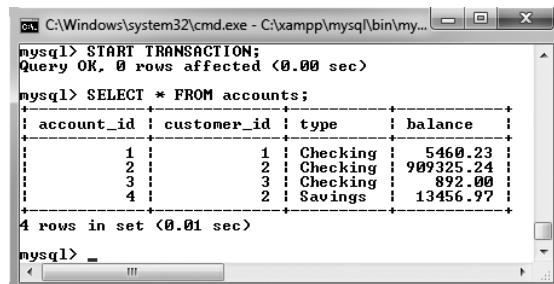
最后，你不可以使用phpMyAdmin执行事务。每个通过phpMyAdmin的SQL窗口或标签提交的查询都是独立完整的事务，不能使用后面的提交撤销。

在记住这些的情况下，接下来在银行数据库使用事务，执行之前提到的任务。在第19章中，事务将通过PHP脚本运行。

### 执行事务

- (1) 连接到MySQL客户端并选择banking数据库。
- (2) 开始一个事务，并显示表的当前内容（参见图7-40）。

```
START TRANSACTION;
SELECT * FROM accounts;
```



The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\my...'. It contains the following MySQL session:

```

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM accounts;
+ account_id + customer_id + type + balance +
| 1 | 1 | Checking | 5460.23 |
| 2 | 2 | Checking | 909325.24 |
| 3 | 3 | Checking | 892.00 |
| 4 | 2 | Savings | 13456.97 |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

图6-40 开始事务并显示现有的表记录

- (3) 从David Sedaris（或者任何用户）的账户中减去100美元。

```
UPDATE accounts
SET balance = (balance-100)
WHERE id=2;
```

使用`UPDATE`查询、一点数学知识以及`WHERE`条件语句，我可以从余额中减去100美元。尽管MySQL将指示一行受到了影响，直到提交了事务之后，其作用才会永久生效。

- (4) 向Sarah Vowell的账户中增加100美元。

```
UPDATE accounts
SET balance = (balance+100)
WHERE id=1;
```

这一步与第(3)步相反，就好像把100美元从一个人转账到另一个人一样。

(5) 确认结果 (参见图7-41)。

```
SELECT * FROM accounts;
```

```

mysql> UPDATE accounts
 -> SET balance = balance+100
 -> WHERE account_id=2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE accounts
 -> SET balance = balance+100
 -> WHERE account_id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM accounts;
+-----+-----+-----+-----+
| account_id | customer_id | type | balance |
+-----+-----+-----+-----+
1	1	Checking	5560.23
2	2	Checking	98925.24
3	3	Checking	892.00
4	2	Savings	13456.97
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

图7-41 执行两个UPDATE查询并查看结果

在该图中可以看到，与原来的余额相比（参见图7-40），一个余额增加了100美元，另一个余额则减少了100美元。

(6) 回滚事务。

```
ROLLBACK;
```

为了演示如何撤销事务，我将撤销这些查询的作用。ROLLBACK命令将数据库返回到开始事务之前的状态。该命令还会终止事务，将MySQL返回到其自动提交模式。

(7) 确认结果 (参见图7-42)。

```
SELECT * FROM accounts;
```

```

mysql> SELECT * FROM accounts;
+-----+-----+-----+-----+
| account_id | customer_id | type | balance |
+-----+-----+-----+-----+
1	1	Checking	5560.23
2	2	Checking	98925.24
3	3	Checking	892.00
4	2	Savings	13456.97
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.09 sec)

mysql> SELECT * FROM accounts;
+-----+-----+-----+-----+
| account_id | customer_id | type | balance |
+-----+-----+-----+-----+
1	1	Checking	5460.23
2	2	Checking	98925.24
3	3	Checking	892.00
4	2	Savings	13456.97
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

图7-42 由于我使用了ROLLBACK命令，所以忽略了UPDATE查询的潜在作用这个查询应该显示原始表中的内容。

(8) 重复第(2)~(4)步。

为了查看在提交事务时发生了什么，将再次运行两个UPDATE查询。不过，要确保首先开始事务，否则查询将会自动生效！

(9) 提交事务并确认结果(参见图7-43)。

```
COMMIT;
SELECT * FROM accounts;
```

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin...'. Inside, a MySQL session is running:

```

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM accounts;
+ account_id | customer_id | type | balance
+-----+-----+-----+-----+
| 1 | 1 | Checking | 5460.23 |
| 2 | 2 | Checking | 98925.24 |
| 3 | 3 | Checking | 892.00 |
| 4 | 2 | Savings | 13456.97 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> UPDATE accounts
 -> SET balance = `balance`-100
 -> WHERE account_id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE accounts
 -> SET balance = `balance`+100
 -> WHERE account_id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> COMMIT;
Query OK, 0 rows affected (0.05 sec)

mysql> SELECT * FROM accounts;
+ account_id | customer_id | type | balance
+-----+-----+-----+-----+
| 1 | 1 | Checking | 5560.23 |
| 2 | 2 | Checking | 98925.24 |
| 3 | 3 | Checking | 892.00 |
| 4 | 2 | Savings | 13456.97 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

图7-43 调用COMMIT命令使事务永久生效

一旦输入COMMIT，整个事务就会永久生效，这意味着任何更改都会发生。COMMIT还会结束事务，将MySQL返回到自动提交模式。

#### ✓ 提示

- 事务的最优秀的特性之一是：发生随机事件时（如服务器崩溃）它们会提供保护。事务要么全部执行，要么忽略所有的更改。
- 为了改变MySQL的自动提交特性，可以输入

```
SET AUTOCOMMIT=0;
```

然后你不必输入START TRANSACTION，并且在输入COMMIT（或者使用ALTER、CREATE等查询）前任何查询都不会永久生效。

- 可以在事务中创建保存点（savepoint）：

```
SAVEPOINT savepoint_name;
```

然后可以回滚到那个点：

```
ROLLBACK TO SAVEPOINT savepoint_name;
```

## 7.7 数据库加密

到目前为止，使用SHA1()函数的伪加密已在数据库中完成。在网站名称和论坛的数据库，用户的密码在经过SHA1()处理后将被存储。虽然以这种方式使用这个函数相当好（并且相当普遍），但这个函数并不能提供真正的加密：SHA1()函数返回了一个值的代表（称为散列），而不是一个加密的值。通过

存储一些数据的散列版本，以后仍然可以进行比较（如登录时），但不能从数据库中获取原始数据。如果你需要以一个受保护的方式存储数据，同时仍然能够在以后的某个时候以其原来的形式查看数据，需要使用其他的MySQL函数。

MySQL有几个内置的加密和解密函数。如果你需要以可以解密的加密形式存储数据，则要使用AES\_ENCRYPT()和AES\_DECRYPT()。AES\_ENCRYPT()函数被认为是最安全的加密选项。

这些函数需要两个参数：要加密或解密的数据和salt参数。salt参数是一个字符串，用于随机加密。让我们先来看看加密和解密函数，然后再回过头来探讨salt。

要在向表中添加记录的同时对数据进行加密，需要用到类似以下的语句：

```
INSERT INTO users (username, pass)
VALUES ('troutster', AES_ENCRYPT ('mypass', 'nacl19874salt!'))
```

AES\_ENCRYPT()函数返回的加密数据将是二进制格式。要在表中存储该数据列必须将其定义为二进制类型（例如，VARBINARY或BLOB）。

要对刚才插入的记录运行登录查询（匹配提交的和数据表中的用户名和密码），你可以这样写：

```
SELECT * FROM users WHERE
username='troutster' AND
AES_DECRYPT(pass, 'nacl19874salt!') = 'mypass'
```

这相当于：

```
SELECT * FROM users WHERE
username = 'troutster' AND
AES_ENCRYPT('mypass', 'nacl19874salt!') = pass
```

再回到salt的问题上，加密和解密必须使用相同的salt，这就意味着salt也必须存储在某个地方。与你可能想到的相反，将salt存储在数据库中实际上是安全的，甚至可以与混淆过得数据放在同一行。这是因为salt的目的是使加密过程更难被破解（特别是“彩虹”攻击）。这种攻击是在远程通过暴力破解达到的。相反，如果有人能看到存储在数据库中的一切，那你可能就需要担心更大的问题了（即你所有的数据已经被破坏）。

最后，从混淆存储的数据中获得最大的利益，每一部分存储的数据应该使用一个不同的salt，salt越长越好。接下来为banking.customers表添加pin和salt列，并且存储每位客户的PIN的加密版本。

### 加密和解密数据

- (1) 访问MySQL并选择银行的数据库。
- (2) 为customers表添加两个新列（参见图7-44）。

```
ALTER TABLE customers ADD COLUMN pin VARBINARY(16) NOT NULL;
ALTER TABLE customers ADD COLUMN nacl CHAR(20) NOT NULL;
```

```
C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p
mysql> ALTER TABLE customers ADD COLUMN pin VARBINARY(16) NOT NULL;
Query OK, 3 rows affected (0.19 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE customers ADD COLUMN nacl CHAR(20) NOT NULL;
Query OK, 3 rows affected (0.06 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
```

图7-44 两个列被添加到customers表

第一列pin，将用于存储用户的PIN加密版本。AES\_ENCRYPT()返回一个16个字符长二进制值，pin列被定义为VARBINARY (16)。第二列中存储salt，是一个20个字符长的字符串。

(3)更新的第一个客户的PIN (参见图7-45)。

```
UPDATE customers
SET nacl = SUBSTRING(MD5(RAND()), -20)
WHERE customer_id=1;
UPDATE customers
SET pin=AES_ENCRYPT(1234, nacl)
WHERE customer_id=1;
```

```
mysql> UPDATE customers
-> SET nacl = SUBSTRING(MD5(RAND()), -20)
-> WHERE customer_id=1;
Query OK, 1 row affected <0.08 sec>
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE customers
-> SET pin=AES_ENCRYPT(1234, nacl)
-> WHERE customer_id=1;
Query OK, 1 row affected <0.00 sec>
Rows matched: 1 Changed: 1 Warnings: 0
```

图7-45 更新了一条记录，使用加密函数保护PIN

第一个查询更新了客户的记录，为nacl列添加了一个salt值。随机值是通过对RAND()函数的输出值使用MD5()函数后得到的。这将创建一个32位字符长的字符串，如4b8bb06aea7f3d162ad5b4d83687e3ac，然后使用SUBSTRING()从这个字符串取最后20个字符。

第二个查询使用已经存储的nacl值作为salt存储客户的PIN-1234。

(4)以检索PIN，以解密形式显示 (参见图7-46)。

```
SELECT customer_id, AES_DECRYPT(pin,
nacl) AS pin FROM customers
WHERE customer_id=1;
```

这个查询返回了customer\_id为1的客户的解密PIN。只要使用的salt值相同，使用AES\_ENCRYPT()存储的任何值，都可以使用AES\_DECRYPT()检索和匹配。

```
mysql> SELECT customer_id, AES_DECRYPT(pin,
-> nacl) AS pin FROM customers
-> WHERE customer_id=1;
+-----+-----+
| customer_id | pin |
+-----+-----+
| 1 | 1234 |
+-----+-----+
1 row in set <0.00 sec>

mysql>
```

图7-46 记录被获取，并在此过程中解码PIN

(5)不使用解密检查客户的记录 (参见图7-47)。

```
SELECT * FROM customers
WHERE customer_id=1;
```

你可以在图中看到，密码的加密版本不可以供人类阅读。

| customer_id | first_name | last_name | pin         | nac1                 |
|-------------|------------|-----------|-------------|----------------------|
| 1           | Sarah      | Uowell    | iñyigJ k2r% | b0cf929ac53b1979c766 |

图7-47 加密的数据以人类不可读的格式进行存储（这里是一个二进制数据字符串）

### ✓ 提示

- 一般来说，使用SHA1()的信息可能将永远不会需要查看，如密码或用户名。使用AES\_ENCRYPT()保护的信息可能以后还需要查看，比如信用卡信息、社会保险号、地址（有可能），等等。
- 提醒一下，永远不存储信用卡号码和其他高风险的数据通常是最安全的选择。
- SHA2()函数是一个改进过的SHA1()，它是求取散列数据的更好选择。我在本书中不选择它的唯一原因就是SHA2()需要MySQL5.5或者更高版本的支持。
- 相同的salt技术可以应用到的SHA1()、SHA2()和其他函数。
- 注意，发送到MySQL或从MySQL服务器接收到的数据都有可能被拦截并窥探。使用SSL连接到MySQL数据库可以获取更好的安全性。

7

## 7.8 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 7.8.1 回顾

- 两个主要的联结类型是什么？
- 为什么别名经常与联结一同使用？
- 为什么在通常会在联结中使用table.column或至少是最佳的实践？
- 外联结中表的使用顺序有什么影响？
- 如何创建一个自联结？
- 什么是聚合函数？
- 在聚合函数中DISTINCT关键词有什么影响？GROUP BY对聚合函数有什么影响？
- 要执行FULLTEXT搜索需要哪种索引？哪种存储引擎？
- 当你在MySQL客户端以\G代替分号结束SELECT查询时会有什么影响？IN BOOLEAN MODE FULLTEXT搜索与标准的FULLTEXT搜索有什么不同？

- 可以用什么命令改善表的性能？
- 如何检查某个查询的效率？
- 为什么论坛数据库不支持事务？
- 如何开始一个事务？如何撤销正在处理中的事务？该如何让当前的事务影响持久化？
- 需要什么样类型的列存储AES\_ENCRYPT()函数的输出？
- 在加密过程中使用的salt有什么重要的标准？

### 7.8.2 实践

- 尝试写出更多的论坛和银行数据库联结的查询。在所有三个表之间执行内联结、外联接。
- 如果你对SQL命令UNION或子查询感兴趣，查看MySQL手册。
- 在银行表和论坛表执行更多的分组查询。
- 在论坛数据库练习FULLTEXT搜索。
- 检查其他查询来查看结果。
- 阅读MySQL手册其他的在线教程，查看解释查询和优化表的信息。
- 做更多有关事务的练习。
- 研究salt密码和彩虹攻击以了解更多的内容。

## 第8章

# 错误处理和调试

8

### 本章内容

- 错误类型和基本的调试方法
- 显示PHP错误
- 调整PHP中的错误报告
- 创建自定义的错误处理程序
- PHP调试技术
- SQL和MySQL调试技术
- 回顾和实践

8

**如**果你按顺序学习本书(最好这样做),下一个主题是学习如何结合使用PHP和MySQL。不过,这个过程无疑会产生错误,这些错误可能难以调试。因此,在转而介绍新概念之前,我们将先用几页的篇幅讨论程序员的切肤之痛——错误。当你获得了一定的经验时,就会犯更少的错误,并且会选择自己的调试方法,但是,初学者可以使用大量的工具和技术来轻松地通过这个学习过程。

本章有3条主要思路。第一条思路关注的是学习开发动态Web站点时可能会发生的各类错误,以及它们的根源可能是什么。第二条思路是以按部就班的方式讲解大量的调试技术。最后,你将看到以可能最优雅的方式处理所发生错误的不同技术。

## 8.1 错误类型与基本调试方法

在利用PHP和MySQL开发Web应用程序时,最终可能会在所用的4种或更多种技术之一中存在潜在的错误。你可能具有HTML问题、PHP问题、SQL错误或MySQL错误。为了能够修复错误,必须先确定错误驻留于哪个领域。

HTML问题通常破坏性最小,并且最容易捕获。如果你的布局全都乱糟糟的,则通常可以知道有一个问题。下一节将讨论用于捕获和修复这些问题的一些步骤以及常规的调试过程。

PHP错误是最常见的错误,因为这种语言是你的应用程序的核心。PHP错误一般归属于下列3个领域:

- 语法错误;
- 运行时错误;
- 逻辑错误。

语法错误最常见，并且最容易修复。如果你只是遗漏了一个分号，就会看到这些错误。这类错误会阻止脚本执行，如果在PHP配置中打开display\_errors，PHP将显示一个错误，其中包括PHP认为它被打开的那一行（参见图8-1）。如果关闭display\_errors，则会看到一个空白页面（在本章后面将学习关于display\_errors的更多知识）。

运行时错误包括那些不会阻止PHP脚本运行（如解析错误所做的那样），但是会阻止脚本做希望它所做的任何事情的错误，比如使用错误的参数数量或类型来调用一个函数。对于这些错误，PHP通常会显示一条消息（参见图8-2），指示准确的问题（再次假定打开了display\_errors）。



图8-1 解析错误（至此，你可能已经看到过许多次）是最常见的一类PHP错误，对于程序员新手更是如此

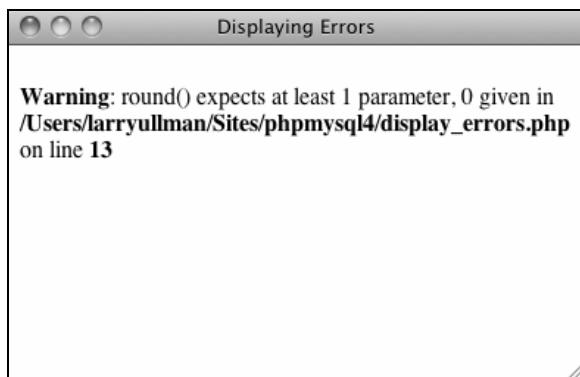


图8-2 在执行脚本期间误用一个函数（用错误的参数调用它）会引发错误

最后一类错误（即逻辑错误）实际上最糟糕，因为PHP不必向你报告它们。这些是原原本本的错误，它们不明显，并且不会阻止脚本的执行。我将用几页的篇幅来演示解决所有这些PHP错误的技巧。

SQL错误通常出在语法上，当你尝试在MySQL上运行查询时，就会报告它们。例如，我曾多次执行如下命令（参见图8-3）：

```
DELETE * FROM tablename
```

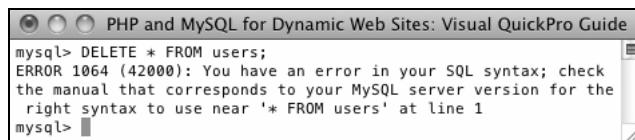


图8-3 MySQL将报告在SQL命令的语法中发现的任何错误

这个语法就是错误的，把它与SELECT语法（`SELECT * FROM tablename`）弄混淆了。正确的语法是：

`DELETE FROM tablename`

同样，当你具有SQL错误时，MySQL将会显示一个红色标志，因此，这些错误不难查找和修复。对于动态Web站点，其特点是你并不总是具有静态查询，PHP通常会动态生成SQL查询。在这种情况下，如果有语法问题，PHP代码可能是真正的罪魁祸首。

除了报告SQL错误之外，MySQL还要考虑它自己的错误。一个常见的错误是，不能访问数据库，并在此显示一个中断标志（参见图8-4）。当你误用MySQL函数或者在联结中有歧义地引用某一列时，也会看到错误。同样，MySQL会以特定的细节报告任何这类错误。记住，当查询没有返回记录或者不具有你所期待的结果时，那不是一个MySQL或SQL错误，而是一个逻辑错误。在本章后面，你将看到如何解决SQL和MySQL错误。

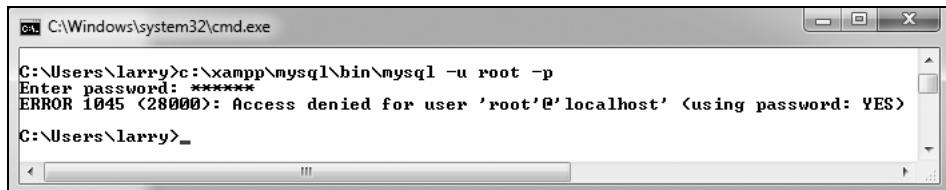


图8-4 不能连接到MySQL服务器或者特定的数据库是一个常见的MySQL错误

但是，正如在跑之前必须先学会走路一样，下一节将介绍调试动态Web站点的基本原理，先从应该执行的基本检查以及如何修复HTML问题开始。

### 摆正心态

在开始之前，先简单谈谈错误，它们会发生在我们当中最优秀的人身上。甚至本书的作者也会在他的Web开发职责中看到大量的错误（但是，请放心，本书中的代码应该是没有错误的）。认为自己将会达到永不犯错的技能级别只是痴人说梦，但是，凭自身的能力知道如何快速地捕获、处理和修复错误是一项重要的技能。因此，在你犯错时，不要感到灰心丧气；相反，掌握这些知识，你就会变成更好的调试员！

#### 8.1.1 基本调试步骤

开始的一系列步骤看起来似乎很明显，但是，当进行调试时，遗漏其中某个步骤将会导致徒劳无功且极度令人受挫的调试经历。在这里，关于调试的最佳建议是，当你受挫时，就离开计算机！休息一会儿，清醒一下头脑，并用明亮的双眼重新审视代码。这样我解决了曾遇到的几乎所有最令人困惑

的问题。本书的支持论坛中的读者往往也会发现确实是这样。在感到自己正在把事情弄得更糟时，要设法向前进。

### 开始调试任何问题

#### (1) 确保正在运行正确的页面。

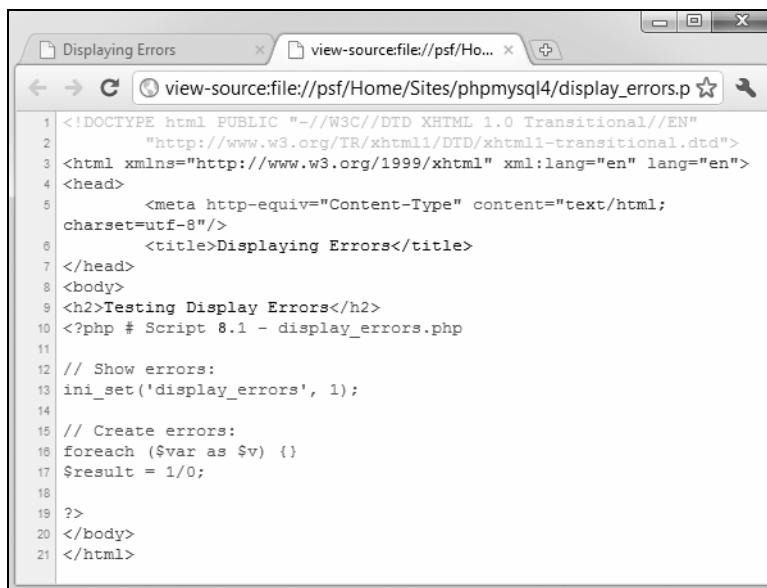
一种极常见的情况是：你尝试修复一个问题，并且不管你做什么，这个问题总是存在。其原因是：实际上你正在编辑的页面与你所想的不同。就这一点而言，多合-IDE(比如，Adobe Dreamweaver, www.adobe.com/go/dreamweaver)有其优势。

#### (2) 确保你保存了最近的更改。

未保存的文档将继续具有与编辑它之前相同的问题（因为编辑没有生效）。我喜欢用Text Mate的原因之一就是，当它失去焦点时，会自动保存文档。

#### (3) 确保你通过URL运行了所有PHP页面。

因为PHP需要使用Web服务器（Apache、IIS等），所以运行任何PHP代码都需要通过一个URL（<http://www.example.com/page.php>或<http://localhost/page.php>）访问页面。如果双击一个PHP页面在浏览器中打开它（或者使用浏览器的“文件”→“打开”选项），则将看到PHP代码，而不是执行的结果。如果没有通过一个URL加载HTML页面（它将自己工作），但后来向PHP页面提交了表单，那么也会发生这种情况（参见图8-5）。



```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
6 <title>Displaying Errors</title>
7 </head>
8 <body>
9 <h2>Testing Display Errors</h2>
10 <?php # Script 8.1 - display_errors.php
11
12 // Show errors:
13 ini_set('display_errors', 1);
14
15 // Create errors:
16 foreach ($var as $v) {}
17 $result = 1/0;
18
19 ?>
20 </body>
21 </html>

```

图8-5 如果通过一个URL运行，则只会执行PHP代码。这意味着提交到PHP页面的表单也必须通过<http://>加载

#### (4) 知道你正在运行的是PHP和MySQL的哪一个版本。

有些问题是PHP或MySQL的某个版本所特有的。例如，有些函数是PHP后来的几个版本中增加的，MySQL也在版本4、4.1和版本5中添加了大量的新特性。运行`phpinfo()`脚本（参见图8-6，查看一个脚

本示例), 并打开一个MySQL客户端会话(参见图8-7)来确定这方面的信息。phpMyAdmin通常还会报告涉及的版本[但是不要把phpMyAdmin的版本(它很可能是3.x)与PHP或MySQL的版本弄混淆]。

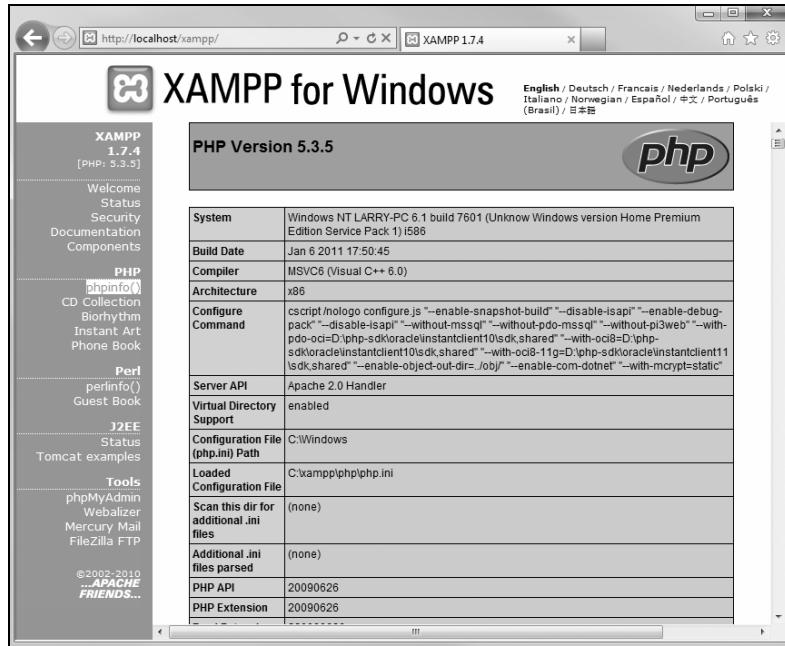


图8-6 `phpinfo()`脚本是最佳的调试工具之一, 它可以告知关于PHP版本及其配置方式的信息

```
C:\Windows\system32\cmd.exe - c:\xampp\mysql\bin\mysql -u root -p
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.8 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

图8-7 当连接到MySQL服务器时, 它应该让你知道版本号

我认为使用的版本是一种至关重要且基本的信息, 在寻求帮助的人们提供这种信息之前, 我通常不会给他们提供帮助!

(5) 知道你正在运行什么Web服务器。

类似地, 有些问题和特性是你的Web服务应用程序(Apache、IIS或Abyss)特有的。你应该通过安装Web服务应用程序来了解正在使用哪一种Web服务器, 以及哪一个版本。

(6) 尝试在不同的Web浏览器中执行页面。

每位Web开发人员都应该具有并使用至少两种Web浏览器。如果在不同的浏览器中测试页面，就会看到是否有涉及脚本或特定浏览器的问题。

(7) 如果可能，尝试使用不同的Web服务器执行页面。

PHP和MySQL错误有时来源于一台服务器上的特定配置和版本。如果某些脚本在一台服务器上工作良好，而在另一台服务器上则无法工作，那么你将知道脚本并非生来就有错误。对此，可以使用`phpinfo()`脚本来查看哪些服务器设置可能有所不同。

#### ✓ 提示

- 如果休息一会儿是你在受挫时所应该做的，那么你不应该做的是，给作者、新闻组、邮件列表或者其他任何人发送一封或多封惊慌失措的、吹毛求疵的电子邮件。当请求陌生人的帮助时，富有耐心和幽默感可以得到好得多和快得多的结果。
- 就此而言，我强烈反对随意猜测解决问题的办法。我见过太多的人在没有完全理解应该改变什么或者不应该改变什么的情况下尝试解决问题，这只能使事情进一步复杂化。
- 还有另一类错误〔可以将其归类为使用(usage)错误〕：当站点用户没有按你所预期的那样行事时所发生的错误。凭你自己极难找出这些错误，因为程序员很难改变他使用应用程序的一贯方式。一条金科玉律是，编写自己的代码时要确保它不会中断，即使用户没有做正确的事情也会如此！

#### 处理错误

如果你遵照本书中的示例执行操作，并且发现某些示例不能正确工作，那么这时应该做什么？

- (1) 针对本书中的示例复查你的代码或步骤。
- (2) 使用本书后面的索引来查看我是否在前面的内容中引用了脚本或函数（你可能漏看了重要的使用规则或提示）。
- (3) 查看针对特定函数的PHP手册，看看它是否只在某个版本中可用，并且核实如何使用该函数。
- (4) 检查本书的勘误表页面，查看代码中是否确实存在错误，并且报告了它们。不过，不要在此发布你的特定问题！
- (5) 第三遍检查你的代码，并使用本章中概括的所有调试技术。
- (6) 搜索本书的支持论坛，查看其他人是否也有这个问题，以及是否已经确定了一种解决方案。
- (7) 如果上述所有办法都失败了，则可使用本书的支持论坛来寻求帮助。采用这种方法时，确保你包括了所有的相关信息（PHP的版本、MySQL的版本、你采取的调试步骤以及结果是什么，等等）。

### 8.1.2 调试HTML

调试HTML相对容易。HTML源代码非常容易访问，大多数问题显而易见，并且尝试修复HTML通常不会使事情变得更糟糕（对于PHP则可能发生这种情况）。尽管如此，你还是应该遵照一些基本的步骤来找出并修复HTML问题。

### 调试HTML错误

#### (1) 检查源代码。

如果有HTML问题，则几乎总是需要检查页面的源代码来找出它。查看源代码的方式依赖于所用的浏览器，但是通常采用的方式是使用像“查看”→“页面源文件”这样的命令。

#### (2) 使用验证工具（参见图8-8）。

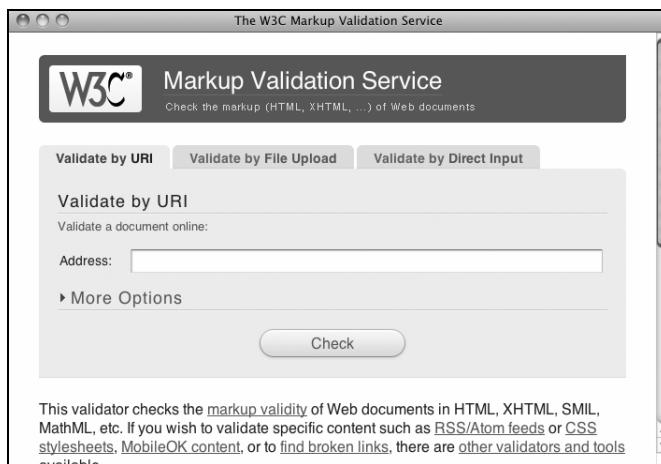


图8-8 验证工具〔像W3C（万维网联盟，World Wide Web Consortium）提供的一个验证工具一样〕可以很好地找出问题，并且确保你的HTML符合标准

验证工具（像`http://validator.w3.org`上提供的一个验证工具一样）可以极好地找出错误匹配的标签、破坏的表以及其他问题。

#### (3) 使用Firefox

我没有尝试开始讨论哪一种Web浏览器最好，并且由于Internet Explorer使用得（仍然！）最广泛，所以要使用它进行最终的测试，但是我个人发现Firefox（可以从`www.mozilla.com`免费得到）是最适合于Web开发人员的Web浏览器。它们提供了其他浏览器中没有提供的可靠性和调试特性。如果你想坚持使用Internet Explorer或Safari进行日常浏览，那也可以，但是在执行Web开发工作时，最好使用Firefox。

#### (4) 使用Firefox的附加部件（参见图8-9）。

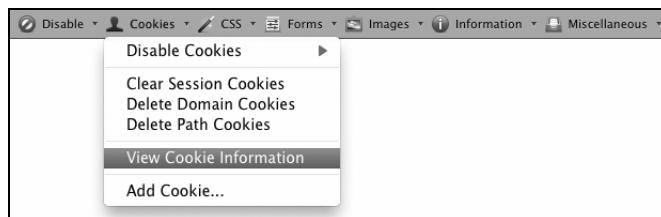


图8-9 Firefox的Web Developer构件提供了对多种有用工具的快速访问

除了只作为优秀的Web浏览器之外，非常流行的Firefox浏览器还有Web开发人员梦寐以求的大量特性。此外，你还可以通过安装任何可用的免费构件来扩展Firefox的功能。特别是，Web Developer构件提供了对优秀工具的快速访问，如显示表的边框、呈现CSS、验证页面等。我还频繁地使用了下面这些附件：DOM Inspector、Firebug和HTML Validator等。

#### (5) 在另一种浏览器中测试页面。

PHP代码一般与浏览器无关，这意味着不论客户是什么，你都将得到一致的结果。HTML则不是这样。有时，特定的浏览器具有某种怪癖，会影响所呈现的页面。在另一种浏览器中运行相同的页面是最容易的调试方式，这样，可以知道它是一个HTML问题，还是浏览器的一种怪癖。

#### ✓ 提示

- 朝着修复各类问题所迈出的第一步是理解引发问题的根源。记住在调试时，每一种技术（HTML、PHP、SQL和MySQL）所扮演的角色。如果你的页面看上去不正确，那就是一个HTML问题。如果你的HTML是由PHP动态生成的，则它仍然是一个HTML问题，但是，你需要处理PHP代码以使之正确。

## 8.2 显示 PHP 错误

当出现错误时，PHP将会提供非常有用的、说明性的出错消息。不幸的是，在使用其默认配置来运行时，PHP不会显示这些错误。这种策略对于活动的服务器是有意义的，在这些服务器上，你不希望最终用户看到PHP特有的出错消息，但是对于PHP开发新手，这还会使得所有的事情都变得更让人糊涂。要查看PHP的错误，必须在独立的脚本中或者作为一个整体为PHP配置打开display\_errors指令。

要在脚本中打开display\_errors，可使用ini\_set()函数。这个函数用指令名称以及它应该具有什么设置作为其参数：

```
ini_set('display_errors', 1);
```

在脚本中包括这一行将为该脚本打开display\_errors。其唯一的缺点是，如果你的脚本具有一个从根本上阻止它运行的语法错误，那么仍然会看到一个空白页面。为了让PHP为整个服务器显示错误，需要编辑其配置文件，见附录A。

#### 打开display\_errors

- (1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为display\_errors.php（参见脚本8-1）。

#### 脚本 8-1 ini\_set() 函数可用于告诉 PHP 脚本呈现可能出现的任何错误

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <title>Displaying Errors</title>
7 </head>
8 <body>
9 <h2>Testing Display Errors</h2>
10 <?php # Script 8.1 - display_errors.php
```

```

11
12 // Show errors:
13 ini_set('display_errors', 1);
14
15 // Create errors:
16 foreach ($var as $v) {}
17 $result = 1/0;
18
19 ?>
20 </body>
21 </html>

```

(2) 在初始PHP标签后面添加：

```
ini_set('display_errors', 1);
```

在该脚本中从此开始将显示出现的任何错误。

(3) 创建一些错误。

```
foreach ($var as $v) { }
$result = 1/0;
```

为了测试display\_errors设置，脚本需要具有一个错误。第一行代码甚至没有尝试做任何事情，但是保证它将会引发一个错误。这里实际上有两个问题：第一，有一个对不存在的变量（\$var）的引用；第二，在foreach循环中将非数组变量（\$var）用作数组。

第二行是经典的除以0问题，在程序设计语言或者数学中都不允许这样做。

(4) 完成页面。

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

(5) 将文件另存为display\_errors.php，将其存放在Web目录中，并在Web浏览器中测试它（参见图8-10）。

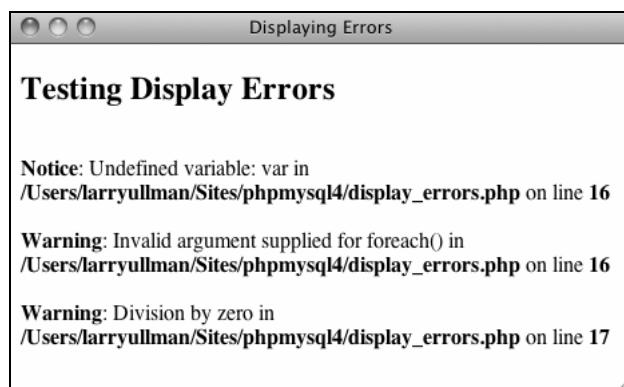


图8-10 打开display\_errors（为这个脚本），当出现错误时页面将报告它们

(6) 如果你愿意，可以更改第一行PHP代码，以读取：

```
ini_set('display_errors', 0);
```

然后保存并再次测试脚本（参见图8-11）。

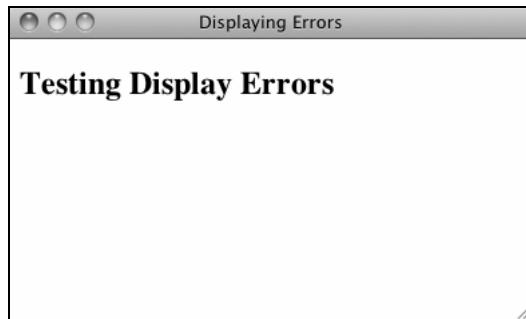


图8-11 关闭display\_errors（为这个脚本），将不再报告相同的错误（脚本8-1和图8-10）。不幸的是，它们仍然存在

#### ✓ 提示

- 对ini\_set()函数可用于调整哪些PHP设置存在一些限制。参见PHP手册以了解关于能控制什么以及不能控制什么的特定细节。
- 提醒一下，如果在一个脚本中更改display\_errors设置，那么仅当该脚本在运行时（也就是说，它不能具有任何解析错误）它才会起作用。为了能够总是看到出现的任何错误，将需要在PHP的配置文件中启用display\_errors（同样请参见附录）。

### 8.3 调整 PHP 中的错误报告

一旦把PHP设置成呈现出发生了哪些错误，你可能想调整错误报告的级别。可以将作为一个整体或独立脚本的PHP安装设置成报告或忽略不同的错误级别。表8-1列出了大多数级别，但是它们一般是以以下3类级别之一：

- 注意（notice），这不会阻止脚本的执行，并且可能不一定是一个问题；
- 警告（warning），这指示一个问题，但是不会阻止脚本的执行；
- 错误（error），这会阻止脚本继续执行（包括常见的解析错误，它从根本上阻止脚本运行）。

表8-1 错误报告等级

编 号	常 量	报 告
1	E_ERROR	致命的运行时错误（它会阻止脚本的执行）
2	E_WARNING	运行时警告（非致命的错误）
4	E_PARSE	解析错误
8	E_NOTICE	注意（事情可能是或者可能不是一个问题）
256	E_USER_ERROR	用户生成的错误消息，由trigger_error()函数生成

(续)

编 号	常 量	报 告
512	E_USER_WARNING	用户生成的警告，由trigger_error()函数生成
1024	E_USER_NOTICE	用户生成的注意，由trigger_error()函数生成
2048	E_STRICT	关于兼容性和互操作性的建议
8192	E_DEPRECATED	警告无法在未来PHP版本中使用的代码
30719	E_ALL	所有的错误、警告和建议

一个经验法则是，当你正在开发站点时，你将希望PHP报告任何类型的错误；但是一旦站点开始运行，你将不希望报告某些特定的错误。出于安全性和美观的目的，让公众用户查看PHP的详细出错消息一般是不明智的。幸运的是，错误消息，特别是那些与数据库有关的出错消息，往往会揭示Web应用程序中最好不要显示出来的某些幕后内容。虽然你希望在开发阶段解决所有这些问题，但是这极少见。

你通常可以遵照附录A中的指导调整错误报告的级别，或者可以使用error\_reporting()函数基于各个脚本来调整这种行为。这个函数用于确定PHP应该在特定的页面内报告哪些类型的错误。该函数获取一个数字或常量，从而使用表8-1中的值（PHP手册列出了与PHP自身的核心相关的其他几个值）。

```
error_reporting(0); // Show no errors.
```

设置为0会完全关闭错误报告（错误仍会发生，只是你不再会看到它们）。与之相反，error\_reporting(E\_ALL)将会向PHP报告发生的每个错误。可以添加一些编号来自定义错误报告的级别，或者可以把位运算符〔（或）、~（非）、&（与）〕和常量一起使用。利用下面这个设置，可以抛出任何非注意的错误：

```
error_reporting (E_ALL & ~E_NOTICE);
```

### 调整错误报告

(1) 在文本编辑器或IDE中打开display\_errors.php（参见脚本8-1）。

为了操纵错误报告级别，把display\_errors.php用作一个例子。

(2) 在调整display\_errors设置之后，添加如下代码（参见脚本8-2）：

#### 脚本 8-2 这个脚本将演示如何在 PHP 中操纵错误报告

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <title>Reporting Errors</title>
7 </head>
8 <body>
9 <h2>Testing Error Reporting</h2>
10 <?php # Script 8.2 - report_errors.php
11
12 // Show errors:
```

```

13 ini_set('display_errors', 1);
14
15 // Adjust error reporting:
16 error_reporting(E_ALL | E_STRICT);
17
18 // Create errors:
19 foreach ($var as $v) {}
20 $result = 1/0;
21
22 ?>
23 </body>
24 </html>

```

出于开发的目的，使PHP向你通告所有的错误、注意、警告和建议。这一行代码将完成这个任务。将错误报告等级设为E\_ALL即可实现。但由于E\_ALL不包含E\_STRICT，所以又加上了“E\_STRICT”。简而言之，PHP将让你知道它是什么，或者可能是一个问题。

由于E\_ALL和E\_STRICT是一个常量，所以没有用引号括住它。

(3) 将文件另存为report\_errors.php，存放在Web目录中，并在Web浏览器中运行它(参见图8-12)。我还改变了页面的标题和头部，但是它们对于本练习来说无关紧要。

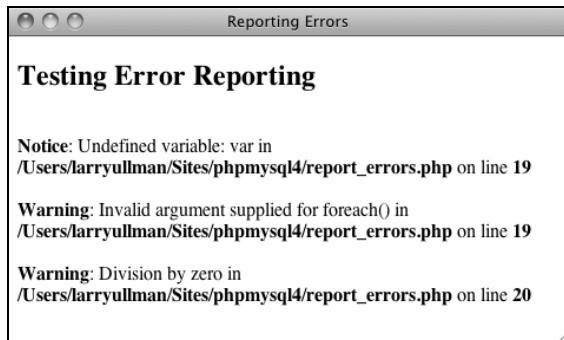


图8-12 在最高的错误报告级别上，PHP将为这个页面产生两个警告和一个注意

(4) 更改错误报告的级别并再次执行测试(参见图8-13和图8-14)。

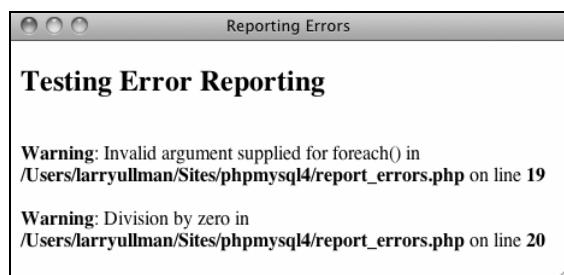


图8-13 禁用通知报告之后的相同页面(参见脚本8-2)

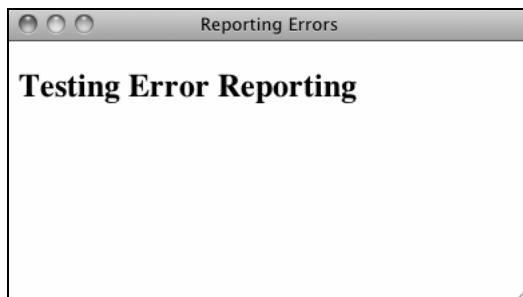


图8-14 关闭错误报告（设置为0）之后的相同页面（参见脚本8-2）。其结果与禁用 `display_errors` 之后的结果相同。当然，错误仍然会发生，只是不会报告它们而已

#### ✓ 提示

- 表8-1中`E_ALL`的数值会根据PHP的版本发生变化。
- 由于你通常希望为Web站点中的每个页面调整`display_errors`和`error_reporting`，你可能想把这几行代码放在单独的PHP文件中，然后可以由任何PHP脚本包含它。
- 本书中的脚本都是在PHP错误报告设置最严格的情况下编写的（意图是捕捉每一个可能的问题）。
- `trigger_error()`函数是一种通过编程在PHP脚本中生成一个错误的方法。它的第一个参数是错误信息，第二个可选参数是数值类型的错误类型，对应于表8-1中的值。默认的类型是`E_USER`。

```
if /* some condition */ {
 trigger_error('Something Bad Happened!');
}
```

#### 利用@来抑制错误

在PHP中，可以使用`@`运算符来抑制单个错误。例如，如果不希望PHP报告它不包括某个文件，则可以编写如下代码：

```
@include ('config.inc.php');
```

或者如果不希望看到“除以0”错误：

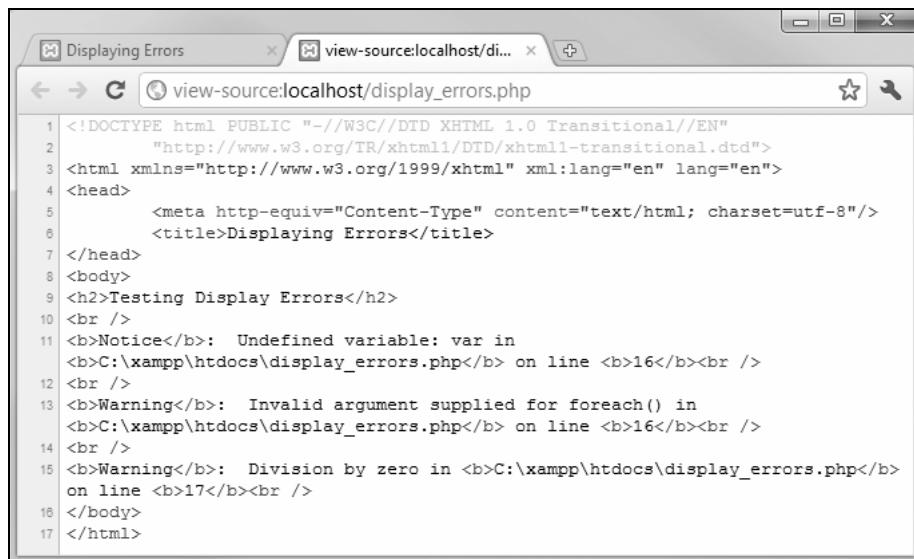
```
$x = 8;
$y = 0;
$num = @($x/$y);
```

像函数调用或数学运算一样，`@`符号只能处理表达式。不能在条件语句、循环语句、函数定义等之前使用`@`符号。

一条经验法则是，我建议将`@`符号用于那些执行失败时不会影响脚本整体功能的函数。或者，在你自己可以更优雅地处理PHP的错误时可以抑制错误（本章后面将讨论这个主题）。

## 8.4 创建自定义的错误处理程序

就你的站点而言，错误管理的另一种选择是改变PHP处理错误的方式。默认情况下，如果启用`display_errors`并且捕获到一个错误（属于错误报告级别），PHP将在一些最少的HTML标签内以非常简单的形式打印错误（参见图8-15）。



```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
6 <title>Displaying Errors</title>
7 </head>
8 <body>
9 <h2>Testing Display Errors</h2>
10

11 Notice: Undefined variable: var in
C:\xampp\htdocs\display_errors.php on line 16

12

13 Warning: Invalid argument supplied for foreach() in
C:\xampp\htdocs\display_errors.php on line 16

14

15 Warning: Division by zero in C:\xampp\htdocs\display_errors.php
on line 17

16 </body>
17 </html>

```

图8-15 图8-12中所示错误的HTML源代码

你可以创建自己的函数，以便在发生错误时调用它来重写处理错误的方式。例如：

```

function report_errors ($arguments) {
 // Do whatever here.
}
set_error_handler ('report_errors');

```

`set_error_handler()`用于指定在出现错误时要调用的函数。此时，处理函数（例如，`report_errors`）将接收可以以任何可能的方式使用的多个值。

也可以把这个函数写成带有多达5个参数，这些参数依次是：错误编号（对应于表8-1）、文本错误消息、找到错误的文件的名称、发生错误的特定行号以及发生错误时存在的变量。定义一个接受所有这些参数的函数可能如下：

```
function report_errors ($num, $msg, $file, $line, $vars) {..}
```

为了利用这个概念，将最后一次重写`report_errors.php`文件（参见脚本8-2）。

### 创建你自己的错误处理程序

- (1) 在文本编辑器或IDE中打开`report_errors.php`（参见脚本8-2）。
- (2) 删除`ini_set()`和`error_reporting()`这几行代码（参见脚本8-3）。

### 脚本 8-3 通过定义你自己的错误处理函数，可以自定义在 PHP 脚本中处理错误的方式

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <title>Handling Errors</title>
7 </head>
8 <body>
9
10 <h2>Testing Error Handling</h2>
11 <?php # Script 8.3 - handle_errors.php
12
13 // Flag variable for site status:
14 define('LIVE', FALSE);
15
16 // Create the error handler:
17 function my_error_handler ($e_number, $e_message, $e_file, $e_line, $e_vars) {
18
19 // Build the error message:
20 $message = "An error occurred in script '$e_file' on line $e_line: $e_message\n";
21
22 // Append $e_vars to $message:
23 $message .= print_r ($e_vars, 1);
24
25 if (!LIVE) { // Development (print the error).
26 echo '<pre>' . $message . "\n";
27 debug_print_backtrace();
28 echo '</pre>
';
29 } else { // Don't show the error.
30 echo '<div class="error">A system error occurred. We apologize for the inconvenience.
31 </div>
';
32 }
33 } // End of my_error_handler() definition.
34
35 // Use my error handler:
36 set_error_handler ('my_error_handler');
37
38 // Create errors:
39 foreach ($var as $v) {}
40 $result = 1/0;
41
42 ?>
43 </body>
44 </html>
```

在建立你自己的错误处理函数时，错误报告级别不再有任何意义，因此可以删除那一行代码。调整display\_errors设置也是没有意义的，因为错误处理函数将会控制是否显示错误。

(3) 在脚本产生错误之前，添加以下代码：

```
define ('LIVE', FALSE);
```

这个常量将是一个标志，用于指示站点目前是否在运行。这是一个重要的区别，当你开发一个站点并且站点在运行时，处理错误的方式以及在浏览器中呈现的内容应该会有极大的区别。

在函数外面设置这个常量有两个原因：第一，我希望把函数作为“黑盒”处理，它会做我需要它做的事情，而不必进入函数内部修改它。第二，在许多站点中，可能还有其他设置（如数据库连通性信息）相对于开发细节是动态的。因此，条件语句也可以引用这个常量来调整这些设置。

(4) 开始定义错误处理函数。

```
function my_error_handler ($e_number, $e_message, $e_file, $e_line, $e_vars) {
```

`my_error_handler()`函数被设置成接受自定义的错误处理程序可以接受的全部5个参数。

(5) 使用接收到的值创建出错消息。

```
$message = "An error occurred in script '$e_file' on line $e_line: $e_message\n";
```

出错消息首先引用发生错误的文件名和编号。然后把实际的出错消息添加进来。在调用函数时（当发生错误时）把所有这些值都传递给该函数。

(6) 把现有的变量添加到出错消息中。

```
$message .= print_r ($e_vars, 1);
```

当发生错误时，`$e_vars`变量将接收存在的所有变量以及它们的值。由于这可能包含有用调试信息，所以我把它添加到消息中。

`print_r()`函数通常用于打印出变量的结构和值，它对于数组特别有用。如果使用第二个参数（1或TRUE）调用函数，则会返回而不是打印结果。因此，这一行会把所有的变量信息添加到`$message`中。

(7) 根据站点是否是活动的，打印的消息将有所不同。

```
if (!LIVE) {
 echo '<pre>' . $message . "\n";
 debug_print_backtrace();
 echo '</pre>
';
} else {
 echo '<div class="error"> A system error occurred. We apologize for the inconvenience.</div>
';
}
```

如果站点不是活动的（如果`LIVE`为`false`，在开发站点时就是这样），应该会打印详细的出错消息（参见图8-16）。为了便于查看，将会在HTML PRE标签内打印出错消息（它并不是合法的 XHTML，但是在这里非常有用）。此外，还会调用有用的调试函数`debug_print_backtrace()`。这个函数将会返回关于调用了哪些函数以及包括了哪些文件等的大量信息。

如果站点是活动的，将会打印简单的出错信息，让用户知道发生了一个错误，但是并未指出特定的问题是什么（参见图8-17）。在这种情况下，也可以使用`error_log()`函数（参见框注“记录PHP错误”）把详细的出错消息用电子邮件发送到或者写到日志中。

(8) 完成函数，并告诉PHP使用它。

```
}
```

```
set_error_handler('my_error_handler');
```

第二行很重要，它告诉PHP使用自定义的错误处理程序，而不是PHP的默认处理程序。

(9) 将文件另存为`handle_errors.php`，将其存放在Web目录中，并在Web浏览器中测试它（参见图8-16）。

(10) 将LIVE的值更改为TRUE，保存并再次测试脚本（参见图8-17）。

为了查看错误处理程序如何处理活动的站点，只需更改这一个值即可。

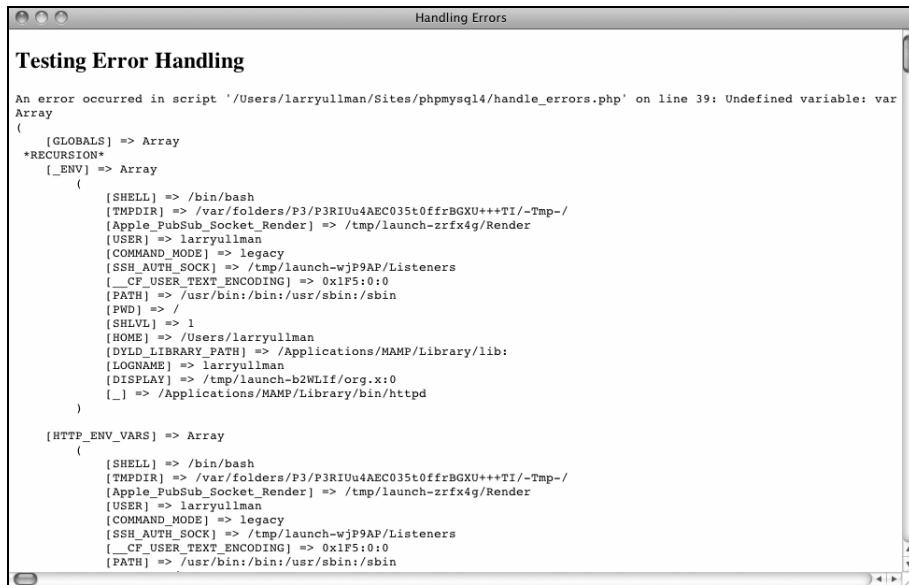


图8-16 在开发阶段，在Web浏览器中打印详细的出错消息（在具有更多代码的更现实的脚本中，消息将更有用）



图8-17 一旦运行站点，就会打印对用户更友好（而不太有启迪作用）的错误。

在这里，为脚本中的三个错误中的每个错误打印一条消息

### ✓ 提示

- 如果你的PHP页面使用了特殊的HTML格式化代码——如CSS标签，以影响布局和字体处理，则要把这种信息添加到你的错误报告函数中。
- 显然，在一个活动站点中，你所需要做的可能不仅仅是为造成的不便道歉（尤其是当发生的错误产生了重大影响时）。尽管如此，这个示例还是演示了如何轻松地调整错误处理以适应这种情况。
- 如果你希望错误处理函数报告每个注意、错误或警告，就可以检查错误编号值（发送给函数

的第一个参数)。例如,为了在站点活动时忽略注意,将把主条件语句更改为:

```
if (!LIVE) {
 echo '<pre>' . $message . "\n";
 debug_print_backtrace();
 echo '</pre>
';
} elseif ($e_number != E_NOTICE) {
 echo '<div class="error">A system error occurred. We apologize for the inconvenience.
 </div>
';
}
```

### 记录PHP错误

在脚本8-3中,仅仅通过详细或简单地打印出错消息来处理错误。

另一种选择是记录错误,以某种方式建立它们的永久记录。为此, `error_log()` 函数指导PHP如何归档一个错误。其语法如下:

```
error_log (message, type, destination,
extra headers);
```

`message`值应该是记录的错误的文本(即脚本8-3中的`$message`)。`type`指示如何记录错误。选项是编号0~3和4使用计算机的默认日志记录方法(0),在电子邮件中发送它(1),把它写到一个文本文件(3),或者发送到Web服务器的日志处理程序(4)。

`destination`参数可以是一个文件的名称(对于日志类型3)或者是一个电子邮件地址(对于日志类型1)。仅当发送电子邮件(日志类型1)时,才会使用`extra headers`参数。`destination`和`extra headers`这两个参数都是可选的。

## 8.5 PHP 调试技术

当进行调试时,了解某些类型的错误的起因非常重要,它会告诉你绝大多数信息。知道错误的起因可以加速错误的修复。为了缩短追寻错误起因的过程,表8-2列出了最常见的PHP错误的可能原因。

表8-2 最常见的PHP错误

错    误	可能的起因
空白页面	HTML问题或PHP错误,并且关闭了 <code>display_errors</code> 或 <code>error_reporting</code>
解析错误	遗漏分号;不对称的花括号、圆括号或引号;或者在字符串中使用了未转义的引号
空变量值	忘记了开头的\$,变量名拼写错误或大小写错误,变量作用域(对于函数)不合适
未定义的变量	在为变量赋值前引用它,或者使用空变量值(参见那些可能的起因)
调用未定义的函数	函数名拼写错误,PHP未被配置成使用那个函数(如一个MySQL函数),或者没有正确地包括那个包含函数定义的文档
不能重新声明函数	你自己的函数存在两种定义,在包含文件内检查它们
头部已经发送	脚本中的PHP标签之前存在空白,已经打印数据,或者已经包含文件

你将遇到的第一种最常见的错误类型是语法错误,它会阻止你的脚本执行。这样一个错误将导致

如图8-18所示的消息，每位PHP开发人员都多次看到过这条消息了。在编程时，为了避免这类错误，要确保：

- 用分号结束每一条语句（但是，像循环语句和条件语句这样的语言构造除外）；
- 使所有的引号、圆括号、花括号和方括号保持对称（每个开始字符都应该有相应的结束字符）；
- 使引号保持一致（只能用单引号关闭单引号，只能用双引号关闭双引号）；
- 根据需要，使用反斜杠对字符串内的所有单引号和双引号进行转义。

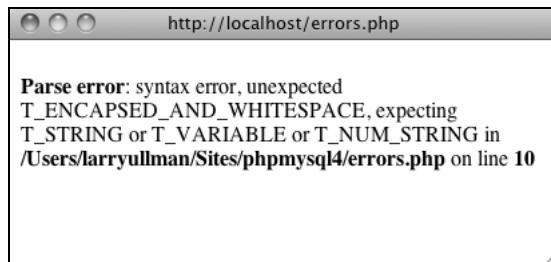


图8-18 由于无效的PHP语法，解析错误会阻止脚本运行。这种错误是由于在打印  
\$array['key'] 的值时，无法将其括在花括号内所引起的

关于语法错误，你还应该了解的一件事情是，仅仅由于PHP出错消息指出错误发生在第12行，并不意味着错误实际上就发生在那一行。至少，PHP认为是在第12行和文本编辑器指出是在第12行这两者之间有所区别是一种常见的现象。因此，虽然PHP的指示对于追寻问题是有用的，但是，最好把引用的行号视作一个起点，而不是一个绝对位置。

如果PHP报告错误发生在文档的最后一行上，这几乎总是由于直到那一刻仍没有捕获到错误匹配的圆括号、花括号或引号。

你会遇到的第二类错误来源于误用函数。例如，当未使用正确的参数调用一个函数时，就会发生这种错误。当试图执行代码时，PHP就会发现错误。在后面几章中，使用header()函数、cookie或会话时，可能会看到这种错误。

为了修复错误，需要做一点检测工作来查看产生了什么错误以及错误发生的位置。不过，对于初学者，他们总是详尽阅读和信任PHP提供的出错消息。尽管引用的行号可能并非总是正确的，但是PHP错误还是非常具有说明性，通常会有所帮助，并且几乎总是百分之百正确。

### 调试脚本

(1) 打开display\_errors。

在开发应用程序时，使用以前介绍的步骤为脚本（如果可能，就为整个服务器）启用display\_errors。

(2) 使用注释。

就像你能够使用注释为脚本加注解一样，也可以使用它们来排除有问题的行。如果PHP告诉你第12行有错误，那么注释掉那一行应该会去除错误。如果没有，那么你就知道错误位于别处。一定要小心，不要通过不正确地注释掉代码块的仅仅一部分而引入更多的错误：必须保证脚本的语法正确。

(3) 使用print和echo函数。

在更复杂的脚本中，我频繁使用echo语句来说明在执行脚本时发生了什么事情（参见图8-19）。当

一个脚本具有多个步骤时，可能不容易知道问题是发生在第(2)步，还是第(5)步。通过使用echo语句，可以把问题范围缩小到特定的接合点。

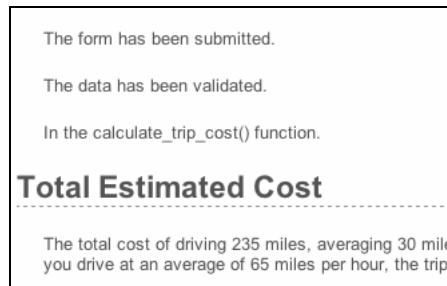


图8-19 可以通过注意脚本正在做什么来完成更复杂的调试

#### (4) 检查使用什么引号来打印变量。

一种比较常见的情况是，程序员错误地使用单引号，然后对为什么没有正确地打印他们的变量感到迷惑不解。记住，单引号按字面意义处理文本，必须使用双引号来打印出变量的值。

#### (5) 跟踪变量（参见图8-20）。

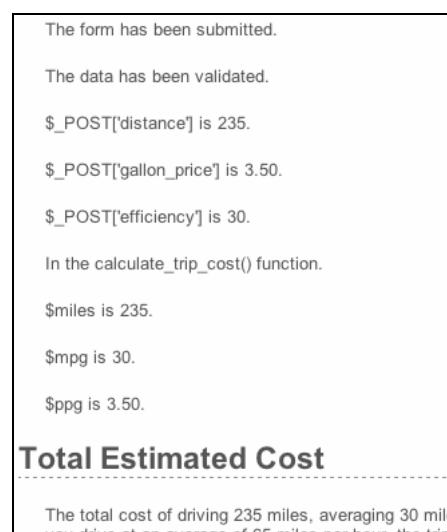


图8-20 在从头到尾执行脚本的过程中，打印变量的名称和值是跟踪它们的最容易的方式

由于你引用了错误的变量或通过错误的名称引用正确的变量，或者由于变量不具有你所期待的值，这些都非常容易使得脚本不工作。要检查这些可能发生的情况，可使用print或echo语句来打印出脚本中重要位置的变量值。这只需编写以下代码：

```
echo "<p>\$var = $var</p>\n";
```

或

```
echo "<p>\$var is $var</p>\n";
```

对第一个美元符号进行转义，使得打印变量的名称。对该变量的第二次引用将打印出它的值。

#### (6) 打印数组值。

对于更复杂的变量类型（数组和对象），无需使用循环，`print_r()`和`var_dump()`函数即可打印出它们的值。这两个函数完成的是相同的任务，尽管与`print_r()`相比，`var_dump()`报告的信息更详细。

#### ✓ 提示

- 许多文本编辑器都包括一些实用程序，用于检查对称的圆括号、方括号和引号。
- 如果无法在复杂的脚本中找到解析错误，则首先可以使用`/* */`注释致使全部PHP代码不起作用。然后同时取消某些部分的注释（通过移动开始或结束注释字符）并继续这个操作，再次运行脚本，直到推断出哪些行引发了错误。不过，一定要留意你是如何注释掉控制结构的，因为花括号必须继续保持匹配，以避免解析错误。例如：

```
if (condition) {
 /* Start comment.
 Insert code.
 End comment. */
}
```

- 为了使`print_r()`的结果在Web浏览器中更易阅读，我往往会把它们包括在HTML`<pre>`（预格式化）标签内。下面这一行代码是我绝对喜爱的调试工具：

```
echo '<pre>' . print_r ($var, 1) . '</pre>';
```

8

#### 使用`die()`和`exit()`

通常与错误管理一起使用的两个函数是`die()`和`exit()`（严格说来，它们是语言构造而不是函数，但是谁在意这些呢）。当在脚本中调用`die()`和`exit()`时，将会终止整个脚本。它们都可用于阻止脚本继续执行，而使得某些重要的操作（如建立一条数据库连接）不会发生。你还可以给`die()`和`exit()`传递一个将在浏览器中打印出来的字符串。

通常可以看到在OR条件语句中使用`die()`和`exit()`。例如

```
include('config.inc.php') OR die ('Could not open the file.');
```

在包含这样一行代码之后，如果PHP不能包含配置文件，将会执行`die()`语句，并且会打印`could not open the file`消息。在全书和PHP手册中，你将会看到它的各种变体，因为它是一种处理错误的快捷方式（但可能处理过度），而无需使用自定义的错误处理程序。

## 8.6 SQL 和 MySQL 调试技术

大多数常见的SQL错误是由以下问题引起的：

- 引号或圆括号的使用不对称；

- 列值中有未转义的撇号；
- 列名、表名或函数拼写错误；
- 在联结中有歧义地引用某一列；
- 以错误的顺序放置查询子句（WHERE、GROUP BY、ORDER BY、LIMIT）。
- 此外，当使用MySQL时，也可能偶尔遇到以下问题：
- 不可预知的或不合适的查询结果；
- 无法访问数据库。

因为你正在从PHP对你的动态Web站点运行查询，所以需要一种方法，用于在那种环境中调试SQL和MySQL错误（PHP不会报告你的SQL问题）。

### 8.6.1 调试SQL问题

要决定你遇到的是一个MySQL（或SQL）问题而不是一个PHP问题，则需要一种方法来查找和修复那个问题。幸运的是，很容易定义调试MySQL和SQL问题所应该采取的步骤，并且应该不假思索地遵循它们。如果你要调试任何MySQL或SQL错误，则只需遵守这个步骤序列即可。

说得明白一点，下面的步骤序列可能是本章和全书中最有用的调试技术。在任何PHP MySQL Web应用程序中，如果你没有得到期待的结果，很可能需要遵循这些步骤。

#### 调试SQL查询

(1) 在PHP脚本中打印出任何适用的查询（参见图8-21）。

```
Home Page | Register | View Users | Change Password | link five
INSERT INTO users (first_name, last_name, email, password, registration_date) VALUES ('Larry', 'Ullman', 'Larry@example.com', SHA1('pass42'), NOW())
```

图8-21 准确知道PHP脚本试图执行的查询是解决SQL和MySQL问题最有用的第一步

如你将在下一章中所看到的，通常把SQL查询分配给一个变量，尤其是在你使用PHP动态编写它们时更是如此。在PHP脚本中使用代码echo \$query（或者任何被调用的查询变量）可以把正在运行的准确查询发送给浏览器。有时，仅仅这一步即可帮助你看出真正的问题所在。

(2) 在MySQL客户端或其他工具中运行查询（参见图8-22）。

```
C:\Windows\system32\cmd.exe - c:\xampp\mysql\bin\mysql -u root -p
mysql> INSERT INTO users (first_name, last_name, email, password, registration_date)
-> VALUES ('Larry', 'Ullman', 'Larry@example.com', SHA1('pass42'), NOW());
ERROR 1054 (42S22): Unknown column 'password' in 'field list'
mysql>
```

图8-22 要了解PHP脚本正在接收什么结果，可以通过单独的界面运行相同的查询。

这里的问题是对password列的引用，因为表的这一列仅仅被命名为pass

调试SQL或MySQL问题最行之有效的方法是，通过一个独立的应用程序（如MySQL客户端、

phpMyAdmin等)来运行脚本中使用的查询。通过这样做,你就可以得到与原始PHP脚本所接收到的相同结果,但是消除了系统开销和麻烦。

如果独立的应用程序返回所期待的结果,但是在PHP脚本中仍未得到正确的行为,那么可知问题出在脚本自身中,而不是出在SQL或MySQL数据库中。

(3)以其最基本的形式重写查询,然后向其中添加回各个元素,直到你发现哪个子句正在引发问题。

有时,调试查询是困难的,因为有太多的事情要做。就像注释掉PHP脚本的大部分内容一样,把查询简化成其最低限度的最简结构,并慢慢将其构建成原来的样子,这可能是调试复杂的SQL命令的最容易的方式。

## 8.6.2 调试访问问题

使用PHP与MySQL交互时,访问被拒绝的出错消息是开发新手最常见的问题。这些问题都有以下的公共解决方案。

- 在改变特权之后重新加载MySQL,使得所做的更改生效。可以使用mysqladmin工具,或者在MySQL客户端中运行FLUSH PRIVILEGES。你必须以具有执行该操作的合适权限的用户身份登录(参见附录A,以了解更多信息)。
- 复查所用的密码。Access denied for user: ‘user@localhost’ (Using password: YES)这条出错消息往往指示密码错误或者输入有误(并非总是这个原因,但是首先要检查它)。
- Can’t connect to... (错误编号2002)这条出错消息指示MySQL要么未运行,要么未运行在客户试验的socket或TCP/IP端口上。

### ✓ 提示

- MySQL保存它自己的错误日志,这在解决MySQL问题(如为什么MySQL甚至无法启动)时非常有用。MySQL的错误日志将位于数据目录中,其名称为hostname.err。
- MySQL手册非常详细,其中包含SQL示例、函数引用以及错误代码的含义。与这本手册交朋友,并在对弹出的错误感到困惑时向它求助。

## 8.7 回顾和实践

无论你是对本书内容存有疑问,还是在自学中遇到问题,都可以把问题发到本书的论坛上,我们会为你答疑解惑。

### 8.7.1 回顾

- 为什么PHP脚本必须通过URL运行?
- 你正使用的是哪个版本的PHP?哪个版本的MySQL?网络服务器程序使用的哪个版本?运行在什么操作系统上?
- 如果浏览器中渲染的网页看起来不正确,你该采取哪些调试步骤?
- 你启用了服务器端的display\_errors了吗?为什么要在开发服务器上启用display\_errors?为

什么在生产服务器上暴露错误是一件糟糕的事情?

- `error_reporting`的级别是如何影响PHP脚本的? 你的PHP服务器上, `error_reporting`设置的是什么级别?
- `@`操作符的作用是什么?
- 使用自定义的错误处理函数的好处是什么? 当使用自定义错误处理函数时, 错误报告级别有什么影响?
- `print`和`echo`可以如何用作错误调试工具? 提示: 有很多正确的答案。
- 修复PHP-SQL-MySQL错误的方法是?

### 8.7.2 实践

- 为Firefox安装Firebug和Web Developer扩展, 如果你还没有安装它们。如果你还没有安装Firefox那就先安装Firefox。
- 如果可以, 在你的开发服务器上启用`display_errors`。
- 如果可以, 在你的开发服务器上将错误报告的级别设置为`E_ALL | E_STRICT`。
- 在PHP手册的中查看`debug_print_backtrace()`函数, 了解更多关于它的内容。
- 考虑使用一个提供内建调试工具的专业级别的IDE。

**本章内容**

- 修改模板
- 连接到MySQL
- 执行简单的查询
- 检索查询结果
- 确保SQL安全
- 统计返回的记录
- 利用PHP更新记录
- 回顾和实践

9

**既**然你已经具有了使用PHP、SQL和MySQL的丰富经验，现在就可以把所有这些技术组合在一起。PHP与MySQL之间稳固的集成只是众多程序员采纳它的一个原因，将感到惊讶的是，可以如此容易地结合使用这两种技术。

本章将使用现有的sitename数据库（在第5章中创建了这个数据库）来构建与users表交互的PHP界面。这里所讲授的知识和使用的示例是所有PHP和MySQL Web应用程序的基础，因为任何PHP和MySQL交互所涉及的基本原理是相同的。

在开始学习本章之前，应该熟悉前7章中所介绍的所有知识。此外，理解第8章中介绍的错误调试和处理技术将使得学习过程更轻松。最后记住，你需要支持PHP的Web服务器以及访问运行的MySQL服务器，以便测试下面的示例。

## 9.1 修改模板

因为本章和下一章中的所有页面都是同一个Web应用程序的一部分，所以使用公共模板系统是有用的。我们将不会从头开始创建一个新模板，而将再次使用第3章中的布局，并且只对头文件中的导航链接做了微小的修改。

### 建立头文件

- (1) 在文本编辑器中打开header.html（参见脚本3-2）。
- (2) 更改链接列表以读取（参见脚本9-1）：

### 脚本9-1 用新的导航链接修改了用于页面模板的站点的头文件

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
2 xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head><?php echo $page_title; ?></title>
5 <link rel="stylesheet" href="includes/style.css" type="text/css" media="screen" />
6 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
7 </head>
8 <body>
9 <div id="header">
10 <h1>Your Website</h1>
11 <h2>catchy slogan...</h2>
12 </div>
13 <div id="navigation">
14
15 Home Page
16 Register
17 View Users
18 Change Password
19 link five
20
21 </div>
22 <div id="content"><!-- Start of the page-specific content. -->
23 <!-- Script 9.1 - header.html -->
```

本章中的所有示例都将涉及注册、查看用户和更改密码这些页面。可以删除第3章中的日期形式和计算器链接。

(3) 将文件另存为header.html。

(4) 把新的头文件存放在Web目录中，将其与footer.html（参见脚本3-3）和style.css（可以从本书的支持Web站点下载它）一起存放在includes目录中。

(5) 在Web浏览器中运行index.php，测试新的头文件（参见图9-1）。

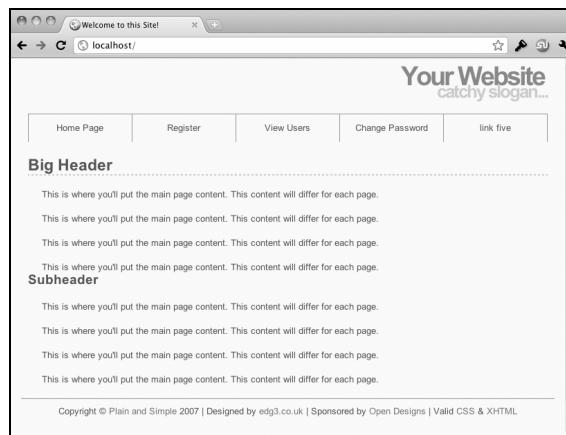


图9-1 动态生成的主页，它带有新的导航链接

### ✓ 提示

- 为了预览这个站点的结构，见下一节中的框注“组织文档”。
- 记住：可以为模板文件使用任何文件扩展名，包括.inc或.php。

### 警告：阅读这段内容

PHP和MySQL在过去10年里发生了许多变化。对本章来说其中最重要的变化以及对本书余下内容来说最重要的变化之一涉及你使用哪些PHP函数与MySQL通信。多年来，PHP开发人员使用标准的MySQL函数（称为mysql扩展）。从PHP 5和MySQL 4.1起，可以使用更新的Improved MySQL函数（称为mysqli扩展）。这些函数提供了改进的性能并且利用了其他特性（还带来了其他好处）。

因为本书假定你至少使用PHP 5和MySQL 5，所有示例都只将使用Improved MySQL函数。如果你的服务器不支持这个扩展，你将不能运行这些像这样编写的示例！本书余下部分中的大多数示例也不会工作。

如果你正在使用的服务器或家庭计算机不支持Improved MySQL函数，则可以有三种选择：升级PHP和MySQL、阅读本书的第2版（它介绍并且主要使用较旧的函数），或者学习如何使用较旧的函数并且相应地修改所有的示例。如果有任何问题，可以查看本书对应的论坛。

9

## 9.2 连接到 MySQL

与MySQL交互时，第一步（连接到服务器）需要适当命名的`mysqli_connect()`函数：

```
$dbc = mysqli_connect (hostname, username, password, db_name);
```

发送给该函数的前三个参数（主机、用户名和密码）基于MySQL内建立的用户和特权（见附录A，以了解更多信息）。通常（但并非总是这样），主机值是localhost。

第四个参数是要使用的数据库的名称。这相当于在MySQL客户端内使用`USE databaseName`命令。

如果建立了连接，\$dbc变量（database connection的简写）将成为所有后续数据库交互的一个参考点。用于处理MySQL的大多数PHP函数都可以把这个变量作为它的第一个参数。

在把这些知识用于测试之前，还要学习另一个函数。如果发生一个连接问题，可以调用`mysqli_connect_error()`，它返回连接错误消息。它不带参数，因此只使用如下代码即可调用它：

```
mysqli_connect_error();
```

为了开始结合使用PHP和MySQL，让我们创建一个建立连接的特殊脚本。然后，需要MySQL连接的其他PHP脚本可以包含这个文件。

### 连接到并选择数据库

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为`mysqli_connect.php`（参见脚本9-2）。

**脚本9-2 mysqli\_connect.php** 脚本将被这个应用程序中的所有其他脚本使用。它建立了一条到达MySQL的连接，并选择该数据库

```

1 <?php # Script 9.2 - mysqli_connect.php
2
3 // This file contains the database access information.
4 // This file also establishes a connection to MySQL,
5 // selects the database, and sets the encoding.
6
7 // Set the database access information as constants:
8 DEFINE ('DB_USER', 'username');
9 DEFINE ('DB_PASSWORD', 'password');
10 DEFINE ('DB_HOST', 'localhost');
11 DEFINE ('DB_NAME', 'sitename');
12
13 // Make the connection:
14 $dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to
MySQL: ' . mysqli_connect_error());
15
16 // Set the encoding...
17 mysqli_set_charset($dbc, 'utf8');
```

其他PHP脚本将包含这个文件，因此它无需包含任何HTML。

(2) 将MySQL主机、用户名、密码和数据库名称设置为常量。

```

DEFINE ('DB_USER', 'username');
DEFINE ('DB_PASSWORD', 'password');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'sitename');
```

出于安全考虑，我更喜欢把这些变量创建为常量（这样，就不能更改它们），但是，这不是必需的。一般来讲，把这些值设置为某种变量或常量比较有意义，这样，你就能够把配置参数与使用它们的函数隔离开，但是，重申一遍，这不是必需的。

在编写脚本时，可以把这些值更改为处理数据库的那些值。如果已经给你提供了MySQL用户名/密码组合和一个数据库（比如用于托管站点），可以在这里使用该信息。或者，如果可能，可遵循附录A中的步骤，创建有权访问sitename数据库的用户，并在此插入那些值。无论你做什么，都不要只使用这些值，除非你确切知道它们将在服务器上工作。

(3) 连接到MySQL。

```
$dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME) OR die ('Could not connect to MySQL:
' . mysqli_connect_error());
```

如果成功连接到MySQL，则`mysqli_connect()`函数将返回对应于打开连接的资源链接。这个链接将被赋予\$dbc变量，以便其他函数可以利用这个连接。

在函数调用前面放置一个错误控制运算符（@），可以防止在Web浏览器中显示PHP错误。这是一种首选的做法，因为错误将由`OR die()`子句处理。

如果`mysqli_connect()`函数不能返回有效的资源链接，那么就会执行语句的`OR die()`部分（因为`OR`的第一部分将为假，因此第二部分必须为真）。如上一章中所讨论的，`die()`函数会终止脚本的执行。该函数还可以取一个将打印到Web浏览器的字符串作为参数。在这种情况下，这个字符串是`Could not`

connect to MySQL 和特定的 MySQL 错误的组合（参见图9-2）。在开发站点时，使用这个迟钝的错误管理系统，可以使得调试要容易得多。

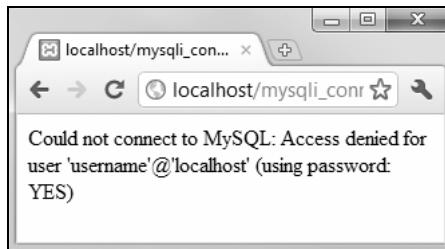


图9-2 如果在连接到MySQL时有问题，就会显示包含丰富信息的消息并中止脚本

(4) 将文件另存为 `mysqli_connect.php`。

因为这个文件包含私有的信息（数据库访问数据），它将使用 `.php` 扩展名。利用 `.php` 扩展名，即使有恶意的用户在他们的 Web 浏览器中运行这个脚本，也不会看到页面的实际内容。

(5) 将这个文件存放在 Web 文档目录的外面（参见图9-3）。

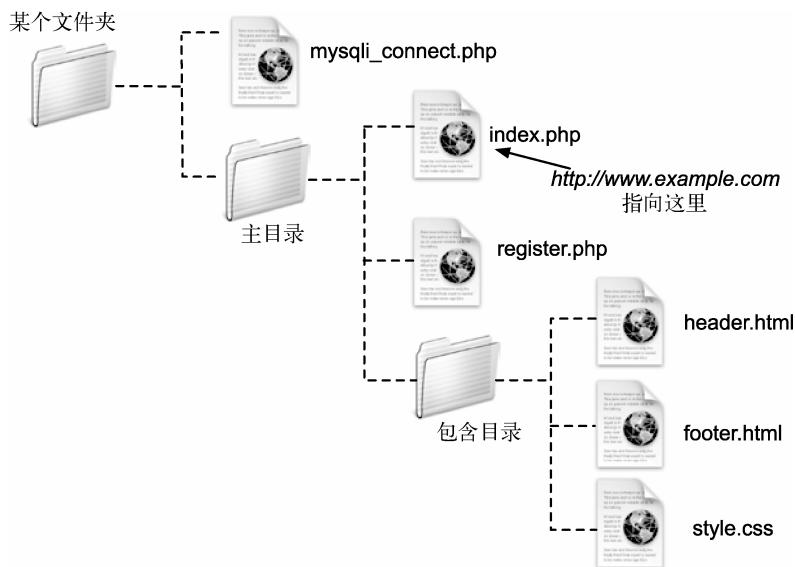


图9-3 服务器的Web文档的形象表示，其中没有把 `mysqli_connect.php` 存储在主目录 (`htdocs`) 内

因为这个文件包含敏感的 MySQL 访问信息，所以应该安全地存储它。如果可以的话，就把它放在 Web 目录的上一级目录中，或者把它放在 Web 目录的外面。这样，就不能从 Web 浏览器访问该文件。参见框注“组织文档”，以了解更多信息。

(6) 在 Web 目录中临时存放一份这个脚本的副本，并在 Web 浏览器中运行这个脚本（参见图9-4）。

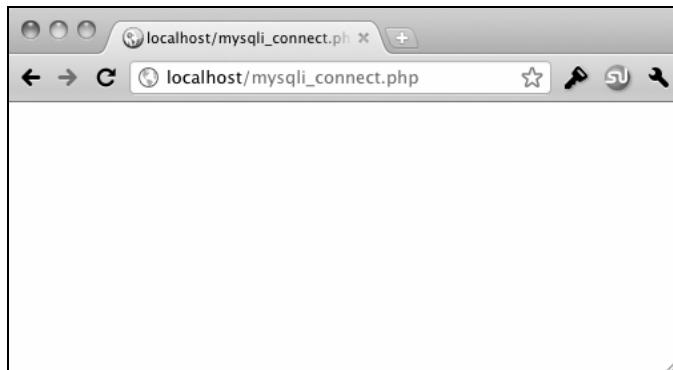


图9-4 如果MySQL连接脚本正确地工作，那么最终的结果将是一个空白页面（该脚本没有生成任何HTML）

为了测试这个脚本，把它的一份副本放在服务器上，以使得可以从Web浏览器访问它（这意味着它必须位于Web目录中）。如果脚本正确地工作，其结果应该是一个空白页面（参见图9-4）。如果看到Access denied...或者类似的消息（参见图9-2），这意味着用户名、密码和主机的组合不具有访问特定数据库的权限。

(7) 从Web目录中删除临时的副本。

#### ✓ 提示

- 如果你使用的PHP版本不支持`mysqli_set_charset()`函数，就需要执行`SET NAMES *encoding*`来代替：
 

```
mysqli_query($dbc, 'SET NAMES utf8');
```
- 在第5章中用于登录MySQL客户端的相同值对于你的PHP脚本也应该有效。
- 如果接收到一个错误，声称`mysqli_connect()`是一个未定义的函数，这意味着没有包含对Improved MySQL Extension的支持来编译PHP。见附录A，以了解有关安装的信息。
- 在运行脚本时，如果看到Can't connect...错误消息（参见图9-5），它很可能意味着MySQL没有运行。

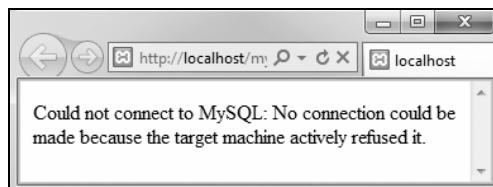


图9-5 PHP也许不能连接到MySQL的另一个原因（除了使用无效的用户名/密码/主机名/数据库信息之外）是MySQL目前没有运行

- 为了满足你的好奇心，图9-6显示了如果在`mysqli_connect()`前面没有使用@并且发生一个错误时，会出现什么情况。



图9-6 如果没有使用错误控制运算符 (@)，你将同时看到PHP错误和自定义的OR die()错误

- 如果在建立到达MySQL的连接时不需要选择数据库，可以从`mysqli_connect()`函数中省略该参数：

```
$dbc = mysqli_connect (hostname, username, password);
```

然后，在合适时，可以使用以下代码选择数据库：

```
mysqli_select_db($dbc, db_name);
```

9

### 组织文档

在开发第一个Web应用程序时，我在第3章中介绍了站点结构的概念。既然页面将开始使用数据库连接脚本，这个主题就显得更重要了。

万一数据库连接信息（用户名、密码、主机和数据库）落入有恶意的人的手中，它就有可能被用于窃取信息，或者从整体上严重破坏数据库。因此，不得不极其安全地保存像`mysqli_connect.php`这样的脚本。

关于安全保存这样一个文件的最重要的建议是，将其存储在Web文档目录的外面。例如，如果图8-3中的`htdocs`文件夹是Web目录的根目录（换句话说，这里的`www.example.com`这个URL所导向的目录），那么不要把`mysqli_connect.php`存储在`html`目录内的任意位置，这意味着永远不能够通过Web浏览器访问它。当然，也不能够从Web浏览器查看PHP脚本的源代码（只能查看通过脚本发送到浏览器的数据），但是，也不需要过于小心。如果不允许你把文档存放在Web目录的外面，那么把`mysqli_connect.php`存放在Web目录中会不太安全，但是这并不是世界末日。

其次，建议为连接脚本使用`.php`扩展名。正确配置和工作的服务器将会执行这样一个文件中的代码，而不会显示它。与之相反，如果你只使用`.inc`作为扩展名，那么如果直接访问它，则会在Web浏览器中显示页面的内容。

## 9.3 执行简单的查询

一旦你成功地连接到并选择一个数据库，就可以开始执行查询。这些查询可以是诸如插入、更新和删除之类的基本查询，或者是返回许多行的复杂联结中所涉及的查询。无论如何，用于执行查询的

PHP函数都是`mysqli_query()`:

```
result = mysqli_query(dbc, query);
```

该函数获取数据库连接作为它的第一个参数，并把查询本身作为第二个参数。我通常会把查询赋予另一个变量`$query`（或者简称为`$q`）。因此，可运行如下查询：

```
$r = mysqli_query($dbc, $q);
```

对于像`INSERT`、`UPDATE`、`DELETE`等简单的查询（它们不会返回记录），`$r`变量（`result`的简写）将返回`TRUE`或`FALSE`，这取决于查询是否执行成功。记住，“执行成功”意味着它没有错误地运行，并不意味着它必须具有想要的结果，你将需要对此进行测试。

对于确实会返回记录的复杂查询（`SELECT`、`SHOW`、`DESCRIBE`和`EXPLAIN`），如果查询有效，则`$r`变量将是一个指向查询结果的资源链接；如果查询无效，则`$r`变量将为`FALSE`。因此，可以在条件语句中使用这一行代码测试查询是否成功运行：

```
$r = mysqli_query ($dbc, $q);
if ($r) { // Worked!
```

如果查询没有成功运行，必定会发生某种MySQL错误。为了查明错误是什么，可以调用`mysqli_error()`函数：

```
echo mysqli_error($dbc);
```

你的脚本中的最后（虽然是可选的）一步是，一旦使用完现有的MySQL连接，就要关闭它：

```
mysqli_close($dbc);
```

这个函数不是必需的，因为在脚本的末尾，PHP会自动关闭连接，但是纳入这个函数确实是一种良好的编程风格。

为了演示这个过程，让我们创建一个注册脚本，在第一次访问时它将显示表单（参见图9-7），处理表单提交，并在验证所有的数据之后把注册信息插入到`sitename`数据库的`users`表中。

图9-7 注册表单

### 执行简单的查询

(1) 在文本编辑器或IDE中创建一个新的PHP脚本，命名为register.php（参见脚本9-3）。

#### 脚本 9-3 这个注册脚本通过运行 INSERT 查询向数据库中添加一条记录

```

1 <?php # Script 9.3 - register.php
2 // This script performs an INSERT query to add a record to the users table.
3
4 $page_title = 'Register';
5 include ('includes/header.html');
6
7 // Check for form submission:
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
9
10 $errors = array(); // Initialize an error array.
11
12 // Check for a first name:
13 if (empty($_POST['first_name'])) {
14 $errors[] = 'You forgot to enter your first name.';
15 } else {
16 $fn = trim($_POST['first_name']);
17 }
18
19 // Check for a last name:
20 if (empty($_POST['last_name'])) {
21 $errors[] = 'You forgot to enter your last name.';
22 } else {
23 $ln = trim($_POST['last_name']);
24 }
25
26 // Check for an email address:
27 if (empty($_POST['email'])) {
28 $errors[] = 'You forgot to enter your email address.';
29 } else {
30 $e = trim($_POST['email']);
31 }
32
33 // Check for a password and match against the confirmed password:
34 if (!empty($_POST['pass1'])) {
35 if ($_POST['pass1'] != $_POST['pass2']) {
36 $errors[] = 'Your password did not match the confirmed password.';
37 } else {
38 $p = trim($_POST['pass1']);
39 }
40 } else {
41 $errors[] = 'You forgot to enter your password.';
42 }
43
44 if (empty($errors)) { // If everything's OK.
45
46 // Register the user in the database...
47
48 require ('../mysqli_connect.php'); // Connect to the db.

```

```

49 // Make the query:
50 $q = "INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES
51 ('$fn', '$ln', '$e', SHA1('$p'), NOW())";
52 $r = @mysqli_query ($dbc, $q); // Run the query.
53 if ($r) { // If it ran OK.
54
55 // Print a message:
56 echo '<h1>Thank you!</h1>
57 <p>You are now registered. In Chapter 12 you will actually be able to log in!</p><p> <br
58 /></p>';
59 } else { // If it did not run OK.
60
61 // Public message:
62 echo '<h1>System Error</h1>
63 <p class="error">You could not be registered due to a system error. We apologize for any
64 inconvenience.</p>';
65
66 // Debugging message:
67 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
68 } // End of if ($r) IF.
69
70 mysqli_close($dbc); // Close the database connection.
71
72 // Include the footer and quit the script:
73 include ('includes/footer.html');
74 exit();
75
76 } else { // Report the errors.
77
78 echo '<h1>Error!</h1>
79 <p class="error">The following error(s) occurred:
';
80 foreach ($errors as $msg) { // Print each error.
81 echo " - $msg
\n";
82 }
83 echo '</p><p>Please try again.</p><p>
</p>';
84
85 } // End of if (empty($errors)) IF.
86
87 } // End of the main Submit conditional.
88 ?>
89 <h1>Register</h1>
90 <form action="register.php" method="post">
91 <p>First Name: <input type="text" name="first_name" size="15" maxlength="20" value="<?php if
92 (isset($_POST['first_name'])) echo $_POST['first_name']; ?>" /></p>
93 <p>Last Name: <input type="text" name="last_name" size="15" maxlength="40" value="<?php if
94 (isset($_POST['last_name'])) echo $_POST['last_name']; ?>" /></p>
95 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="<?php if
96 (isset($_POST['email'])) echo $_POST['email']; ?>" /></p>
97 <p>Password: <input type="password" name="pass1" size="10" maxlength="20" value="<?php if
98 (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" /></p>

```

```

95 <p>Confirm Password: <input type="password" name="pass2" size="10" maxlength="20" value=
96 "<?php if (isset($_POST['pass2'])) echo $_POST['pass2']; ?>" /></p>
97 </p><input type="submit" name="submit" value="Register" /></p>
98 </form>
99 <?php include ('includes/footer.html'); ?>

```

这个脚本的基本内容（使用包含文件、具有显示和处理表单的相同页面以及创建黏性表单）来自于第3章。如果你对其中任何概念感到混淆，可参见第3章中的相应内容。

(2) 创建提交条件语句，并初始化\$errors数组。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 $errors = array();
```

这个脚本将同时显示和处理HTML表单。这个条件语句将检查是否存在隐藏的表单元素，以确定是否处理表单。\$errors变量将用于存储每条出错消息（一条出错消息对应于一个未正确填充的表单输入框）。

(3) 验证名字。

```
if (empty($_POST['first_name'])) {
 $errors[] = 'You forgot to enter your first name.';
} else {
 $fn = trim($_POST['first_name']);
}
```

如第3章所讨论的，empty()函数提供确保文本框被最低限度填充的方式。如果名字文本框没有被填充，就会添加一条出错消息到\$errors数组中。否则，在删除任何无关紧要的空格之后，就会把\$fn设置成提交的值。通过使用这个新变量（显然它是first\_name的简写）从而能够在后面更容易地编写查询的语法。

(4) 验证姓氏和电子邮件地址。

```
if (empty($_POST['last_name'])) {
 $errors[] = 'You forgot to enter your last name.';
} else {
 $ln = trim($_POST['last_name']);
}
if (empty($_POST['email'])) {
 $errors[] = 'You forgot to enter your email address.';
} else {
 $e = trim($_POST['email']);
}
```

这几行代码的语法与验证名字文本框的那几行代码相同。在这两种情况下，都会创建一个新变量，假定通过了最低限度的验证。

(5) 验证密码。

```
if (!empty($_POST['pass1'])) {
 if ($_POST['pass1'] != $_POST['pass2']) {
 $errors[] = 'Your password did not match the confirmed password.';
 } else {
 $p = trim($_POST['pass1']);
 }
} else {
```

```

 $errors[] = 'You forgot to enter your password.';
}

```

为了验证密码，需要检查pass1输入框的值，然后确认pass1值与pass2值是否匹配（因此，密码和确认密码相同）。

(6) 检查是否正确地注册了用户。

```
if (empty($errors)) {
```

如果提交的数据通过了所有的条件，\$errors数组中将没有值（它将为空），因此这个条件将为TRUE，并且可以安全地添加记录到数据库中。如果\$errors数组不为空，那么应该打印合适的出错消息（参见第(11)步），并给用户提供另外一次注册的机会。

(7) 包含数据库连接。

```
require ('../mysqli_connect.php');
```

这行代码会将mysqli\_connect.php文件的内容插入到这个脚本中，因而创建了一个到MySQL的连接并选择了数据库。你可能需要根据你的服务器上的文件的位置来更改文件引用的地址（前面写的这行代码假设mysqli\_connect.php在当前文件夹的父文件夹）。

(8) 添加用户到数据库中。

```

$q = "INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES ('$fn', '$ln',
'$e', SHA1('$p'), NOW());
$r = @mysqli_query ($dbc, $q);

```

这段代码的第一行将把mysqli\_connect.php文件的内容插入到这个脚本中，从而创建一条到达MySQL的连接，并选择该数据库。在你的服务器上，可能需要更改指向文件位置的引用（就像所编写的代码那样，这一行假定mysqli\_connect.php位于当前文件夹的父文件夹中）。

这个查询本身类似于第5章中演示的那些查询。SHA1()函数用于对密码进行加密，NOW()函数用于将注册日期设置为当前时刻。

在将这个查询赋予一个变量之后，就会通过mysqli\_query()函数运行，它会把SQL命令发送到MySQL数据库。就像在mysqli\_connect.php脚本中一样，mysqli\_query()调用前面放置了一个@，以抑制任何丑陋的错误。如果发生问题，就会在下一步中更直接地处理错误。

(9) 报告注册成功。

```

if ($r) {
 echo '<h1>Thank you!</h1>
<p>You are now registered. In Chapter 12 you will actually be able to log in!</p><p>
</p>';
} else {
 echo '<h1>System Error</h1>
<p class="error">You could not be registered due to a system error. We apologize for any
 inconvenience.</p>';
 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
} // End of if ($r) IF.

```

\$r变量被赋予mysqli\_query()返回的值，它可以用在条件语句中以指示查询的成功操作。

如果\$r为TRUE，就会显示消息Thank you!（参见图9-8）。如果\$r为FALSE，就会打印出错消息。出于调试的目的，出错消息将包括MySQL发出的错误（由于mysqli\_error()函数）以及运行的查询（参见图9-9）。该信息对于调试问题是至关重要的。

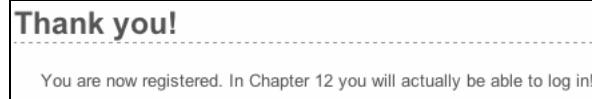


图9-8 如果可以在数据库中注册用户，就会显示该消息

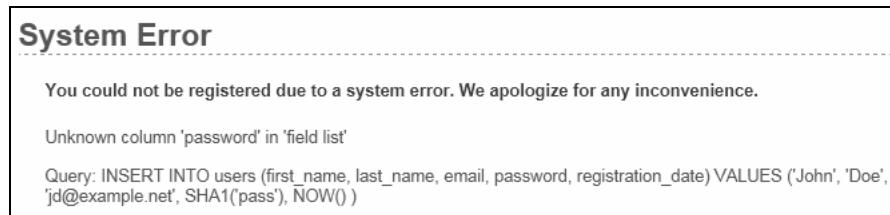


图9-9 将会打印查询引发的任何MySQL错误，在运行查询时也是如此

(10) 关闭数据库连接并完成HTML模板。

```
mysqli_close($dbc);
include ('includes/footer.html');
exit();
```

关闭连接不是必需的，但它是一个良好的策略。然后将包括脚注并终止脚本（由于`exit()`函数）。如果没有这两行代码，那么将会再次显示注册表单（在成功注册之后这是不必要的）。

9

(11) 打印出任何出错消息，并关闭提交条件语句。

```
} else { // Report the errors.
echo '<h1>Error!</h1>
<p class="error">The following error(s) occurred:
';
foreach ($errors as $msg) { // Print each error.
 echo " - $msg
\n";
}
echo '</p><p>Please try again.</p><p>
</p>';
} // End of if (empty($errors)) IF.
} // End of the main Submit conditional.
```

如果有任何错误，就会调用`else`子句。在这里，所有的错误都使用`foreach`循环显示（参见图9-10）。最后的右花括号关闭了主提交条件语句。主条件语句是简单的IF，而不是`if-else`，这使得表单具有黏性（参见第3章）。

(12) 关闭PHP部分，并开始HTML表单。

```
?>
<h1>Register</h1>
<form action="register.php" method="post">
<p>First Name: <input type="text" name="first_name" size="15" maxlength="20" value="<?php if
(isset($_POST
['first_name'])) echo $_POST ['first_name']; ?>" /></p>
<p>Last Name: <input type="text" name="last_name" size="15" maxlength="40" value="<?php if (isset
($_POST['last_name'])) echo $_POST['last_name']; ?>" /></p>
```

Error!

The following error(s) occurred:

- You forgot to enter your last name.
- Your password did not match the confirmed password.

Please try again.

## Register

First Name:

Last Name:

Email Address:

Password:

Confirm Password:

图9-10 每个表单验证错误都会报告给用户，以便他们可以再次尝试注册

这个表单确实很简单，它的每个文本输入框对应于users表中的一个字段（user\_id列除外，它会被自动填充）。通过使用以下代码，使每个输入框都具有黏性：

```
value=<?php if (isset($_POST['var'])) echo $_POST['var']; ?>"
```

同样，强烈建议为表单输入框使用与数据库中对应的列相同的名称，这些列中存储有表单输入框的值。此外，应该把表单中的最大输入框长度设置为等于数据库中的最大列长度。这两个习惯都有助于把错误数降至最少。

(13) 完成HTML表单。

```
<p>Email Address: <input type="text" name="email" size="20" maxlength="60" value=<?php if (isset($_POST['email'])) echo $_POST['email']; ?>" /></p>
<p>Password: <input type="password" name="pass1" size="10" maxlength="20" value=<?php if (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" /></p>
<p>Confirm Password: <input type="password" name="pass2" size="10" maxlength="20" value=<?php if (isset($_POST['pass2'])) echo $_POST['pass2']; ?>" /></p>
<p><input type="submit" name="submit" value="Register" /></p>
</form>
```

这与第(12)步非常相像。表单中也有一个提交按钮和隐藏的输入框。在第3章中讨论了隐藏的输入框这个技巧。

顺便提一下，对于密码输入框，不需要遵守我的maxlength建议（在第(12)步中提出），因为它们将会用SHA()加密，这总是会创建一个长度为40个字符的字符串。并且由于有两个密码输入框，它们不能使用与数据库中的列相同的名称。

(14) 完成模板。

```
<?php include ('includes/footer.html'); ?>
```

(15) 将文件另存为register.php，存放在Web目录中，并在Web浏览器中测试它。

注意：如果在某个表单值中使用撇号，它很可能中断查询（参见图9-11）。9.5节将介绍如何防止这一点。

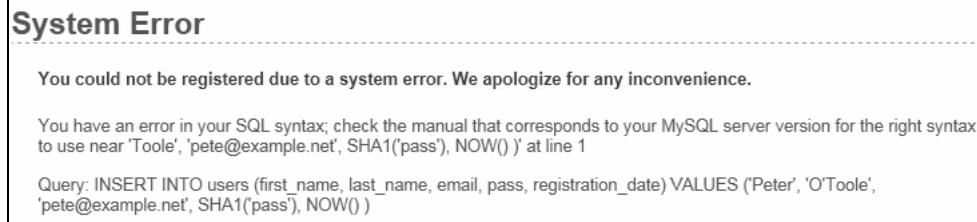


图9-11 表单值中的撇号（如这里的姓氏）将与查询中用于描绘值的撇号相冲突

#### ✓ 提示

- 在运行脚本之后，通过使用MySQL客户端或phpMyAdmin来查看users表中的值，你总是可以确保它将会工作。
- 在PHP中，不应该用分号结束查询，就像你使用MySQL客户端时所做的那样。当使用MySQL时，这是一个经常（虽然不会造成损害）犯的错误。当使用其他的数据库应用程序（例如，Oracle）时，这样做会使查询不起作用。
- 提醒一下，如果能够在数据库上无错地执行查询，`mysqli_query()`函数就会返回TRUE。这不一定意味着查询的结果就是你所期待的。后面一些脚本将演示如何更准确地度量查询的成功执行。
- 不必完全像我所做的那样创建一个\$q变量（可以直接把查询文本插入到`mysqli_query()`中）。不过，随着查询的构造变得更复杂，使用变量将是唯一的选择。
- 实际上，也可以使用`mysqli_query()`来执行将在MySQL客户端中运行的任何查询。
- Improved MySQL Extension超过标准扩展的另一个好处是：`mysqli_multi_query()`函数允许你同时执行多个查询。这样做的语法更复杂一点，尤其是当查询返回结果时，如果你有这种需要，可以查看PHP手册。

9

## 9.4 检索查询结果

在本章的前一节中，讨论并演示了如何在MySQL数据库上执行简单的查询。可以把简单的查询（我这样称呼它）定义为以`INSERT`、`UPDATE`、`DELETE`或`ALTER`开头的一个查询。所有这4种查询的一个共同点是，它们不返回任何数据，而只返回一个它们成功执行的指示。与之相反，`SELECT`查询会生成必须由其他PHP函数处理的信息（即它会返回几行记录）。

处理`SELECT`查询结果的主要工具是`mysqli_fetch_array()`，它带有一个查询结果变量（我曾称为`$r`），并以数组格式一次返回一行数据。你将希望在一个循环内使用这个函数，只要有更多的行，循

环就会持续访问返回的每一行。从查询中读取每条记录的基本构造如下：

```
while ($row = mysqli_fetch_array($r)) {
 // Do something with $row.
}
```

你几乎总是希望使用**while**循环从SELECT查询中获取结果。

**mysqli\_fetch\_array()**函数带有一个可选的参数，用于指定返回的数组的类型：关联数组、索引数组，或者这两者。关联数组允许通过名称引用列值，而索引数组则要求只使用数字（对于返回的第一列，从0开始）。每个参数都是通过表9-1中列出的常量定义的。与其他选项相比，**MYSQLI\_NUM**设置要快那么一点点（并且使用更少的内存）。与之相反，**MYSQLI\_ASSOC**更明确（\$row['column']比\$row[3]更明确），即使表结构或查询发生变化，它也会继续工作。

表9-1 **mysqli\_fetch\_array()**常量

常量	示例
<b>MYSQLI_ASSOC</b>	\$row['column']
<b>MYSQLI_NUM</b>	\$row[0]
<b>MYSQLI_BOTH</b>	\$row[0]或\$row['column']

使用**mysqli\_fetch\_array()**函数时，可以采取的一个可选的步骤是：一旦使用查询结果信息完成了工作，即可释放这些信息：

```
mysqli_free_result($r);
```

这一行会消除\$r占用的系统开销（内存）。该步骤是可选的，因为PHP将在脚本末尾自动释放资源，但是就像使用**mysqli\_close()**一样，这是一种良好的编程风格。

为了演示如何处理查询返回的结果，将创建一个脚本，用于查看当前注册的所有用户。

### 检索查询结果

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为view\_users.php（参见脚本9-4）。

**脚本9-4 view\_users.php** 脚本在数据库上运行一个静态查询，并打印出返回的所有行

```

1 <?php # Script 9.4 - view_users.php
2 // This script retrieves all the records from the users table.
3
4 $page_title = 'View the Current Users';
5 include ('includes/header.html');
6
7 // Page header:
8 echo '<h1>Registered Users</h1>';
9
10 require ('../mysqli_connect.php'); // Connect to the db.
11
12 // Make the query:
13 $q = "SELECT CONCAT(last_name, ' ', first_name) AS name, DATE_FORMAT(registration_date, '%M %d,
14 %Y') AS dr FROM users ORDER BY registration_date ASC";
15 $r = @mysqli_query ($dbc, $q); // Run the query.
```

```

16 if ($r) { // If it ran OK, display the records.
17
18 // Table header.
19 echo '<table align="center" cellspacing="3" cellpadding="3" width="75%">
20 <tr><td align="left">Name</td><td align="left">Date Registered</td></tr>
21 ';
22
23 // Fetch and print all the records:
24 while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
25 echo '<tr><td align="left">' . $row['name'] . '</td><td align="left">' . $row['dr'] .
26 '</td></tr>
27 ';
28 }
29
30 echo '</table>; // Close the table.
31
32 mysqli_free_result ($r); // Free up the resources.
33 }
34
35 // Public message:
36 echo '<p class="error">The current users could not be retrieved. We apologize for any
inconvenience.</p>';
37
38 // Debugging message:
39 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
40
41 } // End of if ($r) IF.
42
43 mysqli_close($dbc); // Close the database connection.
44
45 include ('includes/footer.html');
46 ?>
```

### (2) 连接并查询数据库。

```

require ('../mysqli_connect.php');
$q = "SELECT CONCAT(last_name, ' ', first_name) AS name, DATE_FORMAT(registration_date, '%M %d, %Y')
AS dr
FROM users ORDER BY registration_date ASC";
$r = @mysqli_query ($dbc, $q);
```

这里的查询将返回两列（参见图9-12）：用户名（格式化为Last Name, First Name）和他们的注册日期（格式化为Month DD, YYYY）。由于使用MySQL函数来格式化这两列，所以会把别名提供给返回的结果（相应地为name和dr）。如果你对这种语法的任何方面感到糊涂，参见第5章。

### (3) 创建HTML表格显示查询结果。

```

if ($r) {
 echo '<table align="center" cellspacing="3" cellpadding="3" width="75%">
<tr><td align="left">Name </td><td align="left"> Date Registered</td></tr>
';
```

如果变量\$r的值为TRUE，那么查询会正确运行并显示结果。在HTML中创建一个表格和一个标题行来实现它。

```

mysql> SELECT CONCAT(last_name, ' ', first_name) AS name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr FROM users ORDER BY registration_date ASC;
+-----+-----+
| name | dr |
+-----+-----+
| Ullman, Larry | March 31, 2011 |
| Isabella, Zoe | March 31, 2011 |
| Starr, Ringo | March 31, 2011 |
| Harrison, George | March 31, 2011 |
| McCartney, Paul | March 31, 2011 |
| Lennon, John | March 31, 2011 |
| Brautigan, Richard | March 31, 2011 |
| Banks, Russell | March 31, 2011 |
| Simpson, Homer | March 31, 2011 |
| Simpson, Marge | March 31, 2011 |
| Simpson, Bart | March 31, 2011 |
| Simpson, Lisa | March 31, 2011 |
| Simpson, Maggie | March 31, 2011 |
| Simpson, Abe | March 31, 2011 |
| Chabon, Michael | March 31, 2011 |
| Greene, Graham | March 31, 2011 |
| DeLillo, Don | March 31, 2011 |
| Jones, David | March 31, 2011 |
| Dolenz, Micky | March 31, 2011 |
| Nesmith, mike | March 31, 2011 |
| Sedaris, David | March 31, 2011 |
| Hornby, Nick | March 31, 2011 |
| Bank, Melissa | March 31, 2011 |
| Morrison, Toni | March 31, 2011 |
| Franzen, Jonathan | March 31, 2011 |
| Doe, Jane | April 15, 2011 |
+-----+-----+
26 rows in set (0.00 sec)

mysql>

```

图9-12 MySQL客户端查询运行结果

## (4) 获取并打印每个返回的记录。

```

while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
 echo '<tr><td align="left">' . $row['name'] . '</td><td align= "left">' . $row['dr'] . '</td> </tr>';
}

```

接下来，使用`mysqli_fetch_array()`遍历结果，并打印每个获取的行。注意，在`while`循环中，引用返回值的代码使用了特有的别名：`$row['name']`和`$row['dr']`。脚本不能引用`$row['first_name']`或`$row['date_registered']`，因为没有返回这样的字段名（参见图9-12）。

## (5) 闭合HTML表格并且释放查询资源。

```

echo '</table>';
mysqli_free_result ($r);

```

虽然这是一个可选步骤，但最好遵循。

## (6) 完成主要的条件句。

```

} else {
 echo '<p class="error">The current users could not be retrieved. We apologize for any inconvenience.
 </p>';
 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
} // End of if ($r) IF.

```

在register.php例子中，有两类错误信息。第一种是通用信息，用于显示在站点的信息。第二类是更加详细的，同时显示了MySQL错误和查询错误，都是调试的关键信息。

(7) 关闭数据库连接，完成页面。

```
mysqli_close($dbc);
include ('includes/footer.html');
?>
```

(8) 将文件另存为view\_users.php，存放在Web目录中，并在浏览器中测试它（参见图9-13）。



The screenshot shows a table titled "Registered Users". The table has two columns: "Name" and "Date Registered". The data is as follows:

Name	Date Registered
Ullman, Larry	March 31, 2011
Isabella, Zoe	March 31, 2011
Starr, Ringo	March 31, 2011
Harrison, George	March 31, 2011
McCartney, Paul	March 31, 2011
Lennon, John	March 31, 2011
Brautigan, Richard	March 31, 2011
Banks, Russell	March 31, 2011
Simpson, Homer	March 31, 2011
Simpson, Marge	March 31, 2011
Simpson, Bart	March 31, 2011
Simpson, Lisa	March 31, 2011
Simpson, Maggie	March 31, 2011
Simpson, Abe	March 31, 2011
Chabon, Michael	March 31, 2011
Greene, Graham	March 31, 2011
DeLillo, Don	March 31, 2011
Jones, David	March 31, 2011
Dolenz, Micky	March 31, 2011
Nesmith, mike	March 31, 2011
Sedaris, David	March 31, 2011
Hornby, Nick	March 31, 2011
Bank, Melissa	March 31, 2011
Morrison, Toni	March 31, 2011
Franzen, Jonathan	March 31, 2011
Doe, Jane	April 15, 2011

图9-13 从数据库中检索所有的用户记录，并在Web浏览器中显示它们

#### ✓ 提示

- `mysqli_fetch_array()`函数与`mysqli_fetch_array ($r, MYSQLI_NUM)`函数等价。
- `mysqli_fetch_assoc()`函数与`mysqli_fetch_array($r, MYSQLI_ASSOC)`函数等价。
- 与任何关联数组一样，当从数据库中检索记录时，引用的列必须与数据库中定义的列完全一样。这就是说键是区分大小写的。
- 如果需要在`while`循环内运行第二个查询，则一定要为其使用不同的变量名称。例如，内查询将使用`$r2`和`$row2`，而不是`$r`和`$row`。否则，将遇到逻辑错误。
- 经常看到PHP开发新手对获取查询结果的过程感到困惑。记住：必须使用`mysqli_query()`执行查询，然后使用`mysqli_fetch_array()`来检索单行信息。如果要检索多行，则可使用`while`循环。

## 9.5 确保SQL安全

关于PHP的数据库安全可归结为三大类问题：

- (1) 保护MySQL访问信息；
- (2) 不要呈现关于数据库的过多信息；
- (3) 在运行查询时要小心谨慎，对于那些涉及用户提交数据的查询尤其需要这样。

可以通过确保Web目录外面的MySQL连接脚本的安全来达到第一个目标，这样，永远都不能通过Web浏览器查看到它（参见图9-3）。在本章前面比较详细地讨论了这一点。通过不允许用户查看PHP的出错消息或者你的查询来达到第二个目标（在这些脚本中，将出于调试目的打印出这些信息；在活动站点上永远都不要想这样做）。

对于第三个目标，可以并且应该采取许多步骤来达到，它们都基于永远不要信任用户提供的数据这个前提。第一，验证提交了某个值，或者验证它是否具有正确的类型（数字、字符串等）。第二，使用正则表达式确保提交的数据与你所期待的匹配（在第14章中介绍了这个主题）。第三，可以强制转换某些值的类型以保证它们是数字（在第13章中讨论了这个主题）。第四，通过`mysqli_real_escape_string()`函数处理用户提交的数据。这个函数通过转义那些可能有问题的字符来清理数据，其用法如下：

```
$safe = mysqli_real_escape_string ($dbc, $data);
```

要理解为什么这是必须的，请参见图9-11。在用户的姓氏中使用的单引号使查询语法失效：

```
INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES ('Peter', 'O'Toole', 'pete@example.net', SHA1('aPass8'), NOW())
```

在这个特殊的例子中，有效的用户数据破坏查询，这是不对的。但是，如果你的PHP脚本允许这种可能性，恶意用户就有可能提交有问题的字符（撇号就是一个例子），侵入或破坏你的数据库。为了安全起见，应该为表单中的所有文本输入使用`mysqli_real_escape_string()`函数。为了说明这一点，修改`register.php`（脚本9-3）。

使用`mysqli_real_escape_string()`

- (1) 在文本编辑器或IDE中打开`register.php`（参见脚本9-3）。
- (2) 把包含`mysqli_connect.php`文件的代码（参见脚本9-3中的第48行）移到主条件语句之后（参见脚本9-5）。

**脚本 9-5 register.php 脚本现在使用`mysqli_real_escape_string()`函数来清理提交的数据**

```

1 <?php # Script 9.5 - register.php #2
2 // This script performs an INSERT query to add a record to the users table.
3
4 $page_title = 'Register';
5 include ('includes/header.html');
6
7 // Check for form submission:
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
9
10 require ('../mysqli_connect.php'); // Connect to the db.
11

```

```

12 $errors = array(); // Initialize an error array.
13
14 // Check for a first name:
15 if (empty($_POST['first_name'])) {
16 $errors[] = 'You forgot to enter your first name.';
17 } else {
18 $fn = mysqli_real_escape_string($dbc, trim($_POST['first_name']));
19 }
20
21 // Check for a last name:
22 if (empty($_POST['last_name'])) {
23 $errors[] = 'You forgot to enter your last name.';
24 } else {
25 $ln = mysqli_real_escape_string($dbc, trim($_POST['last_name']));
26 }
27
28 // Check for an email address:
29 if (empty($_POST['email'])) {
30 $errors[] = 'You forgot to enter your email address.';
31 } else {
32 $e = mysqli_real_escape_string($dbc, trim($_POST['email']));
33 }
34
35 // Check for a password and match against the confirmed password:
36 if (!empty($_POST['pass1'])) {
37 if ($_POST['pass1'] != $_POST['pass2']) {
38 $errors[] = 'Your password did not match the confirmed password.';
39 } else {
40 $p = mysqli_real_escape_string($dbc, trim($_POST['pass1']));
41 }
42 } else {
43 $errors[] = 'You forgot to enter your password.';
44 }
45
46 if (empty($errors)) { // If everything's OK.
47
48 // Register the user in the database...
49
50 // Make the query:
51 $q = "INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES
52 ('$fn', '$ln', '$e', SHA1('$p'), NOW())";
53 $r = @mysqli_query ($dbc, $q); // Run the query.
54 if ($r) { // If it ran OK.
55
56 // Print a message:
57 echo '<h1>Thank you!</h1>
58 <p>You are now registered. In Chapter 12 you will actually be able to log in!</p><p>
59
</p>';
60
61 } else { // If it did not run OK.
62
63 // Public message:
64 echo '<h1>System Error</h1>
65 <p class="error">You could not be registered due to a system error. We apologize for any

```

```

 inconvenience.</p>';
64
65 // Debugging message:
66 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
67
68 } // End of if ($r) IF.
69
70 mysqli_close($dbc); // Close the database connection.
71
72 // Include the footer and quit the script:
73 include ('includes/footer.html');
74 exit();
75
76 } else { // Report the errors.
77
78 echo '<h1>Error!</h1>
79 <p class="error">The following error(s) occurred:
';
80 foreach ($errors as $msg) { // Print each error.
81 echo " - $msg
\n";
82 }
83 echo '</p><p>Please try again.</p><p>
</p>';
84
85 } // End of if (empty($errors)) IF.
86
87 mysqli_close($dbc); // Close the database connection.
88
89 } // End of the main Submit conditional.
90 ?>
91 <h1>Register</h1>
92 <form action="register.php" method="post">
93 <p>First Name: <input type="text" name="first_name" size="15" maxlength="20" value="<?php if
94 (isset($_POST['first_name'])) echo $_POST['first_name']; ?>" /></p>
95 <p>Last Name: <input type="text" name="last_name" size="15" maxlength="40" value="<?php if
96 (isset($_POST['last_name'])) echo $_POST['last_name']; ?>" /></p>
97 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="<?php if
98 (isset($_POST['email'])) echo $_POST['email']; ?>" /></p>
99 <p>Password: <input type="password" name="pass1" size="10" maxlength="20" value="<?php if
100 (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" /></p>
101 <p>Confirm Password: <input type="password" name="pass2" size="10" maxlength="20" value=
102 "<?php if (isset($_POST['pass2'])) echo $_POST['pass2']; ?>" /></p>
103 <p><input type="submit" name="submit" value="Register" /></p>
104 </form>
105 <?php include ('includes/footer.html'); ?>
```

因为`mysqli_real_escape_string()`函数需要一条数据库连接，所以在脚本中调用这个函数之前，必须有`mysqli_connect.php`脚本。

(3) 更改验证例程，以使用`mysqli_real_escape_string()`函数，用`$var = mysqli_real_escape_string($dbc, trim($_POST['var']))`代替出现的每个`$var = trim($_POST['var'])`。

```

$fn = mysqli_real_escape_string ($dbc, trim($_POST['first_name']));
$ln = mysqli_real_escape_string ($dbc, trim($_POST['last_name']));
$e = mysqli_real_escape_string ($dbc, trim($_POST['email']));
$p = mysqli_real_escape_string ($dbc, trim($_POST['pass1']));
```

这样做不是只把提交的值赋予每个变量（\$fn、\$ln等），而是首先用`mysqli_real_escape_string()`函数处理这些值。仍然使用`trim()`函数删除任何不必要的空格。

(4) 在主条件语句末尾之前添加对`mysqli_close()`的第二个调用。

```
mysqli_close($dbc);
```

为了保持一致，因为主条件语句的第一步是打开数据库连接，所以在同一个条件语句的最后一步应该关闭它。不过，在包括脚注和终止脚本（第73行、第74行）之前，仍然需要关闭它。

(5) 将文件另存为register.php，将其存放在Web目录中，然后在Web浏览器中测试它（参见图9-14和图9-15）。

图9-14 由于使用了`mysqli_real_escape_string()`函数，其中带有撇号的值（如人们的姓氏）将不再中断INSERT查询



图9-15 现在注册过程将处理有问题的字符，并且更安全

### ✓ 提示

- `mysqli_real_escape_string()`函数依照所用的语言对字符串进行转义，这是它超过替代解决方案的另外一个优点。
- 如果看到了类似于图9-16中的那些结果，则意味着`mysqli_real_escape_string()`函数不能访问数据库（因为它没有连接，如`$dbc`）。

```
Notice: Undefined variable: dbc in C:\xampp\htdocs\register.php on line 18
Warning: mysqli_real_escape_string() expects parameter 1 to be mysqli, null given in C:\xampp\htdocs\register.php on line 18
```

图9-16 由于`mysqli_real_escape_string()`函数需要一条数据库连接，如果在没有连接的情况下（例如，在包括连接脚本之前）使用它，则可能导致其他错误

- 如果在服务器上启用Magic Quotes，那么在使用`mysqli_real_escape_string()`函数之前，需要删除Magic Quotes添加的任何斜杠。代码（看上去有些麻烦）如下：

```
$fn = mysqli_real_escape_string ($dbc, trim(stripslashes($_POST['first_name'])));
```

如果没有使用`stripslashes()`并且启用了Magic Quotes，将对表单值进行双重转义。

- 如果你看一下存储在数据库中（使用MySQL客户端，phpMyAdmin的或其他工具）的值，你将不会看到前面带有反斜杠的撇号和其他有问题的字符，这是正确的。反斜杠会阻止有问题的字符破坏查询，但是反斜杠本身并不会被存储。

## 9.6 统计返回的记录

要讨论的下一个逻辑函数是`mysqli_num_rows()`。这个函数返回SELECT查询检索的行数，并将查询结果变量作为一个参数：

```
$num = mysqli_num_rows($r);
```

尽管有意使其简单，但是这个函数确实非常有用。如果你想分页显示查询结果（可以在下一章中找到这样的一个示例），就需要使用它。另外，在尝试使用`while`循环获取任何结果之前，最好也使用这个函数（因为如果没有结果的话，就无需获取它，并且尝试这样做可能会引发错误）。在下面的步骤序列中，让我们修改`view_users.php`来列出注册用户的总数。参见框注“修改register.php”，可以看到使用`mysqli_num_rows()`的另一个示例。

### 修改register.php

可以对`register.php`应用`mysqli_num_rows()`函数，防止某个人用相同的电子邮件地址注册多次。尽管数据库中那一列上的`UNIQUE`索引将会阻止这种情况的发生，但这种尝试还是会生成一个MySQL错误。为了使用PHP阻止这种情况，可以运行一个SELECT查询确认电子邮件地址目前尚未注册。可以将该查询简单地编写为：

```
SELECT user_id FROM users WHERE email='$e'
```

你将运行这个查询（使用`mysqli_query()`函数），然后调用`mysqli_num_rows()`。如果`mysqli_num_rows()`返回0，你就知道电子邮件地址尚未注册，并且可以安全地运行`INSERT`查询。

### 修改view\_users.php

- (1) 在文本编辑器或IDE中打开`view_users.php`（参见脚本9-4）。
- (2) 在`if($r)`条件语句之前，添加下面这一行代码（参见脚本9-6）：

```
$num = mysqli_num_rows ($r);
```

这一行代码将把查询返回的行数赋予`$num`变量。

**脚本 9-6** 现在，由于使用了`mysqli_num_rows()`函数，`view_users.php`脚本将显示注册用户的总数

```
1 <?php # Script 9.6 - view_users.php #2
2 // This script retrieves all the records from the users table.
3
4 $page_title = 'View the Current Users';
```

```

5 include ('includes/header.html');
6
7 // Page header:
8 echo '<h1>Registered Users</h1>';
9
10 require ('../mysqli_connect.php'); // Connect to the db.
11
12 // Make the query:
13 $q = "SELECT CONCAT(last_name, ' ', first_name) AS name, DATE_FORMAT(registration_date, '%M %d,
14 %Y')AS dr FROM users ORDER BY registration_date ASC";
15 $r = @mysqli_query ($dbc, $q); // Run the query.
16
17 // Count the number of returned rows:
18 $num = mysqli_num_rows($r);
19
20 if ($num > 0) { // If it ran OK, display the records.
21
22 // Print how many users there are:
23 echo "<p>There are currently $num registered users.</p>\n";
24
25 // Table header.
26 echo '<table align="center" cellspacing="3" cellpadding="3" width="75%">
27 <tr><td align="left">Name</td><td align="left">Date Registered</td></tr>
28 ';
29
30 // Fetch and print all the records:
31 while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
32 echo '<tr><td align="left">' . $row['name'] . '</td><td align="left">' . $row['dr'] .
33 '</td></tr>';
34 }
35
36 echo '</table>'; // Close the table.
37
38 mysqli_free_result ($r); // Free up the resources.
39 } else { // If no records were returned.
40
41 echo '<p class="error">There are currently no registered users.</p>';
42
43 }
44
45 mysqli_close($dbc); // Close the database connection.
46
47 include ('includes/footer.html');
48 ?>
```

9

(3) 把原来的\$r条件语句更改如下：

```
if ($num > 0) {
```

这个条件语句与以前编写的条件语句一样，都是基于查询是否会成功运行，而不是基于是否会返回任何记录。现在，它将更准确。

(4) 在创建HTML表格之前，打印出注册用户的数量。

```
echo "<p>There are currently $num registered users.</p>\n";
```

(5) 更改主条件语句的else部分，以读取：

```
echo '<p class="error">There are currently no registered users.</p>';
```

原来的条件语句基于查询是否工作。如果顺利的话，你已经成功调试了查询，使得它正在工作，并且不再需要原来的出错消息。现在，出错消息只是指示是否不会返回任何记录。

(6) 将文件另存为view\_users.php，将其存放在Web目录中，并在Web浏览器中测试它（参见图9-17）。

Registered Users	
There are currently 26 registered users.	
Name	Date Registered
Ullman, Larry	March 31, 2011
Isabella, Zoe	March 31, 2011
Starr, Ringo	March 31, 2011
Harrison, George	March 31, 2011

图9-17 注册用户的数量现在显示在页面顶部

## 9.7 利用PHP更新记录

本章中介绍的最后一一种技术说明了如何利用PHP脚本来更新数据库记录。这需要使用UPDATE查询，并且可以利用PHP的mysqli\_affected\_rows()函数来验证查询的成功执行。

虽然mysqli\_num\_rows()函数会返回SELECT查询生成的行数，但是mysqli\_affected\_rows()函数会返回受INSERT、UPDATE或DELETE查询影响的行数。其用法如下：

```
$num = mysqli_affected_rows($dbc);
```

与mysqli\_num\_rows()函数不同的是，这个函数所带的一个参数是数据库连接（\$dbc），而不是前一个查询的结果(\$r)。

下面的示例是一个脚本，允许注册用户更改他们的密码。它将演示两个重要的思想：

- 针对注册值检查提交的用户名和密码（以及登录系统的关键条件）；
- 通过把主键用作参考点来更新数据库记录。

与注册示例一样，这个PHP脚本将显示表单（参见图9-18）并处理它。

图9-18 用于更改用户密码的表单并处理它

## 利用PHP更新记录

(1) 在文本编辑器或IDE中创建一个新的PHP脚本，命名为password.php（参见脚本9-7）。

**脚本 9-7 password.php** 脚本在数据库上运行一个 UPDATE 查询，并使用 mysqli\_affected\_rows() 函数来确认更改

```

1 <?php # Script 9.7 - password.php
2 // This page lets a user change their password.
3
4 $page_title = 'Change Your Password';
5 include ('includes/header.html');
6
7 // Check for form submission:
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
9
10 require ('../mysqli_connect.php'); // Connect to the db.
11
12 $errors = array(); // Initialize an error array.
13
14 // Check for an email address:
15 if (empty($_POST['email'])) {
16 $errors[] = 'You forgot to enter your email address.';
17 } else {
18 $e = mysqli_real_escape_string($dbc, trim($_POST['email']));
19 }
20
21 // Check for the current password:
22 if (empty($_POST['pass'])) {
23 $errors[] = 'You forgot to enter your current password.';
24 } else {
25 $p = mysqli_real_escape_string($dbc, trim($_POST['pass']));
26 }
27
28 // Check for a new password and match
29 // against the confirmed password:
30 if (!empty($_POST['pass1'])) {
31 if ($_POST['pass1'] != $_POST['pass2']) {
32 $errors[] = 'Your new password did not match the confirmed password.';
33 } else {
34 $np = mysqli_real_escape_string($dbc, trim($_POST['pass1']));
35 }
36 } else {
37 $errors[] = 'You forgot to enter your new password.';
38 }
39
40 if (empty($errors)) { // If everything's OK.
41
42 // Check that they've entered the right email address/password combination:
43 $q = "SELECT user_id FROM users WHERE (email='$e' AND pass=SHA1('$p'))";
44 $r = @mysqli_query($dbc, $q);
45 $num = @mysqli_num_rows($r);
46 if ($num == 1) { // Match was made.
47

```

```
48 // Get the user_id:
49 $row = mysqli_fetch_array($r, MYSQLI_NUM);
50
51 // Make the UPDATE query:
52 $q = "UPDATE users SET pass=SHA1('$np') WHERE user_id=$row[0]";
53 $r = @mysqli_query($dbc, $q);
54
55 if (mysqli_affected_rows($dbc) == 1) { // If it ran OK.
56
57 // Print a message.
58 echo '<h1>Thank you!</h1>
59 <p>Your password has been updated. In Chapter 12 you will actually be able to log
in!</p><p>
</p>';
60
61 } else { // If it did not run OK.
62
63 // Public message:
64 echo '<h1>System Error</h1>
65 <p class="error">Your password could not be changed due to a system error. We apologize
for any inconvenience.</p>';
66
67 // Debugging message:
68 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
69
70 }
71
72 mysqli_close($dbc); // Close the database connection.
73
74 // Include the footer and quit the script (to not show the form).
75 include ('includes/footer.html');
76 exit();
77
78 } else { // Invalid email address/password combination.
79 echo '<h1>Error!</h1>
80 <p class="error">The email address and password do not match those on file.</p>';
81 }
82
83 } else { // Report the errors.
84
85 echo '<h1>Error!</h1>
86 <p class="error">The following error(s) occurred:
';
87 foreach ($errors as $msg) { // Print each error.
88 echo " - $msg
\n";
89 }
90 echo '</p><p>Please try again.</p><p>
</p>';
91 } // End of if (empty($errors)) IF.
92
93 mysqli_close($dbc); // Close the database connection.
94
95 } // End of the main Submit conditional.
96 ?>
97 <h1>Change Your Password</h1>
98 <form action="password.php" method="post">
```

```

100 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="<?php if
101 (isset($_POST['email'])) echo $_POST['email']; ?>" /> </p>
102 <p>Current Password: <input type="password" name="pass" size="10" maxlength="20" value=
103 "<?php if (isset($_POST['pass'])) echo $_POST['pass']; ?>" /></p>
104 <p>New Password: <input type="password" name="pass1" size="10" maxlength="20" value="<?php if
105 (isset($_POST['pass1'])) echo $_POST['pass1']; ?>" /></p>
106 <p>Confirm New Password: <input type="password" name="pass2" size="10" maxlength="20" value=
107 "<?php if (isset($_POST['pass2'])) echo $_POST['pass2']; ?>" /></p>
108 <p><input type="submit" name="submit" value="Change Password" /></p>
109 </form>
110 <?php include ('includes/footer.html'); ?>
```

(2) 开始创建主条件语句。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

因为这个页面显示并处理表单，所以它将使用标准的条件语句。

(3) 包含数据库连接，并创建一个数组来存储错误。

```
require ('../mysqli_connect.php');
$errors = array();
```

这个脚本的初始部分将模拟注册表单。

(4) 验证电子邮件地址和目前的密码文本框。

```
if (empty($_POST['email'])) {
 $errors[] = 'You forgot to enter your email address.';
} else {
 $e = mysqli_real_escape_string($dbc, trim($_POST['email']));
}
if (empty($_POST['pass'])) {
 $errors[] = 'You forgot to enter your current password.';
} else {
 $p = mysqli_real_escape_string($dbc, trim($_POST['pass']));
}
```

表单(参见图9-18)将具有4个输入框：电子邮件地址、目前的密码以及两个用于新密码的输入框。验证其中每个输入框的过程与register.php中的相同。通过验证测试的任何数据都将进行整理，并用mysqli\_real\_escape\_string()函数处理它们，使得可以在查询中安全地使用它们。

(5) 验证新密码。

```
if (!empty($_POST['pass1'])) {
 if ($_POST['pass1'] != $_POST['pass2']) {
 $errors[] = 'Your new password did not match the confirmed password.';
 } else {
 $np = mysqli_real_escape_string($dbc, trim($_POST['pass1']));
 }
} else {
 $errors[] = 'You forgot to enter your new password.';
```

这段代码也与注册脚本中的完全相同，只不过将有效的新密码赋予\$np变量(因为\$p表示目前的密码)。

(6) 如果通过了所有的测试，就会检索用户的ID。

```

if (empty($errors)) {
 $q = "SELECT user_id FROM users WHERE (email='$e' AND pass=SHA1('$p')) ";
 $r = @mysqli_query($dbc, $q);
 $num = @mysqli_num_rows($r);
 if ($num == 1) {
 $row = mysqli_fetch_array($r, MYSQLI_NUM);
 }
}

```

第一个查询只会返回与提交的电子邮件地址和密码匹配的记录的user\_id字段（参见图9-19）。为了比较提交的密码和存储的密码，可使用SHA1()函数再次对其进行加密。如果用户被注册，并且正确地输入了电子邮件地址和密码，则刚好会选择一行（因为电子邮件值对于所有行必须是唯一的）。最后，把这条记录作为（一个元素的）数组赋予\$row变量。

```

+-----+
| user_id |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

```

图9-19 在MySQL客户端内从脚本运行SELECT查询（它所具有的两个查询中的第一个查询）得到的结果

如果脚本的这个部分没有工作，就应用标准的调试方法：删除错误控制运算符(@)，以便你可以查看错误（如果有错误发生的话）；使用mysqli\_error()函数报告任何MySQL错误；打印错误，然后使用另一个界面运行查询（与图9-19中一样）。

#### (7) 更新数据库。

```

$q = "UPDATE users SET pass=SHA1('$np') WHERE user_id=$row[0]";
$r = @mysqli_query($dbc, $q);

```

这个查询将使用新的提交值更改密码，其中user\_id列等于从前一个查询中检索的数字。

#### (8) 检查查询的结果。

```

if (mysqli_affected_rows($dbc) == 1) {
 echo '<h1>Thank you!</h1>
<p>Your password has been updated. In Chapter 12 you will actually be able to log in!</p><p>
</p>';
} else {
 echo '<h1>System Error</h1>
<p class="error">Your password could not be changed due to a system error. We apologize for any
inconvenience.</p>';
 echo '<p>' . mysqli_error($dbc) . '

Query: ' . $q . '</p>';
}

```

脚本的这一部分的工作方式同样类似于register.php脚本。在这里，如果mysqli\_affected\_rows()返回数字1，就更新了记录，将会打印出成功消息。如果没有返回这个数字，就会打印出公共的普通消息和更有用的调试消息。

#### (9) 包括脚注并终止脚本。

```
mysqli_close($dbc);
include ('includes/footer.html');
exit();
```

在脚本中的这个位置，已经运行了UPDATE查询。它要么会工作，要么不会（由于系统错误）。在这两种情况下，都无需再次显示表单，因此包括脚注（以完成页面），并使用exit()函数终止脚本。

(10) 完成if (\$num == 1)条件语句。

```
} else {
echo '<h1>Error!</h1>
<p class="error">The email address and password do not match those on file.</p>';
}
```

如果mysqli\_num\_rows()没有返回值1，那么提交的电子邮件地址和密码与文件上的那些电子邮件地址和密码不匹配，并且会打印这个错误。在这种情况下，将会再次显示表单，以便用户可以输入正确的信息。

(11) 打印任何验证出错消息。

```
} else {
echo '<h1>Error!</h1>
<p class="error">The following error(s) occurred:
';
foreach ($errors as $msg) { // Print each error.
echo " - $msg
\n";
}
echo '</p><p>Please try again.</p><p>
</p>';
}
```

9

如果\$errors数组不为空（这意味着表单数据没有通过所有的验证测试），就会使用这个else子句。与注册页面中一样，将会打印错误。

(12) 关闭数据库连接并完成PHP代码。

```
mysqli_close($dbc);
}
?>
```

(13) 显示表单。

```
<h1>Change Your Password</h1>
<form action="password.php" method="post">
<p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="<?php if (isset($_
POST['email'])) echo $_POST['email']; ?>" /></p>
<p>Current Password: <input type="password" name="pass" size="10" maxlength="20" value="<?php if (
isset($_
POST['pass'])) echo $_POST['pass']; ?>" /></p>
<p>New Password: <input type="password" name="pass1" size="10" maxlength="20" value="<?php if (isset(
$_POST['pass1'])) echo $_POST['pass1']; ?>" /></p>
<p>Confirm New Password: <input type="password" name="pass2" size="10" maxlength="20" value="<?php if (
isset($_
POST['pass2'])) echo $_POST['pass2']; ?>" /></p>
<p><input type="submit" name="submit" value="Change Password" /></p>
</form>
```

这个表单带有3个不同的密码输入框（当前密码、新密码和新密码确认）以及一个电子邮件地址文本输入框。电子邮件地址输入框是黏性的（密码输入框不能是黏性的）。

(14) 包括脚注文件。

```
<?php include ('includes/footer.html'); ?>
```

(15) 将文件另存为password.php，存放在Web目录中，并在Web浏览器中测试它（参见图9-20和图9-21）。

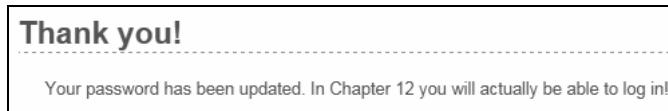


图9-20 在数据库中更改密码

The screenshot shows an "Error!" message stating that the email address and password do not match those on file. It features a "Change Your Password" form with fields for Email Address, Current Password, New Password, and Confirm New Password, all of which are masked with dots.

图9-21 如果输入的电子邮件地址和密码与文件上的那些电子邮件地址和密码不匹配，则不会更新密码

#### ✓ 提示

- 如果使用命令TRUNCATE tablename从表中删除所有记录，则mysqli\_affected\_rows()会返回0，即使查询成功执行并且删除了每一行。这只不过是一种古怪的行为。
- 如果运行UPDATE查询，但是实质上没有更改任意列的值（例如，用相同的密码代替一个密码），则mysqli\_affected\_rows()会返回0。
- 这里使用的mysqli\_affected\_rows()条件语句也可以应用于register.php脚本，以确认是否添加了一条记录。与检查if(\$r)相比，这个条件更严格。

## 9.8 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 9.8.1 回顾

- 你正使用的是什么版本的PHP？什么版本的MySQL？你的PHP-MySQL的组合是否支持MySQL Improved扩展？
- 调试PHP-MySQL的问题最重要的步骤是什么（第8章末尾包含了这些内容）？
- 你用于连接MySQL的主机名、用户名和密码组合是什么？
- 用于连接MySQL服务的PHP代码是什么？选择数据库的呢？创建编码的呢？
- 你使用的是什么编码？为什么要为PHP的脚本使用与MySQL交互和数据存储文本相同的编码？
- 为什么最好将`mysqli_connect.php`脚本放到站点根目录的外部存储？什么是站点根目录？
- 为什么不应该让在线的站点显示MySQL错误和正在运行的查询？
- 通常，你用于处理SELECT查询的结果的语法是什么？如果SELECT查询只返回一行，可以使用什么语法处理？
- 使用`mysqli_real_escape_string()`函数的重要性是什么？
- 什么类型的查询之后要使用`mysqli_num_rows()`函数？
- 什么类型的查询之后要使用`mysqli_affected_rows()`函数？

### 9.8.2 实践

9

- 如果你不记得如何使用模板系统或者如何使用`include()`函数的话，重新温习第3章。
- 使用在第8章涵盖的技术，为站点示例应用自定义的错误处理。
- 修改`view_user.php`中的`mysqli_num_rows()`，仅在查询返回TRUE结果时调用它。
- 按照前面边栏建议的为`register.php`应用`mysqli_affected_rows()`函数。
- 为`register.php`应用`mysqli_affected_rows()`函数以确定`INSERT`生效。
- 创建一个与银行数据库交互的脚本。以下面这些内容创建一个简单的项目：查看所有客户、查看所有账户（使用联接同时显示客户名字），增加或减少账户的余额。

# 常用编程技术 10

## 本章内容

- 给脚本发送值
- 使用隐藏的表单输入框
- 编辑现有的记录
- 给查询结果标页码
- 建立可排序的显示结果
- 回顾和实践

**既**

然你已经了解了一点关于PHP和MySQL交互的知识，现在就该深入学习这方面的内容了。本章与第3章相似，这是由于它介绍了多个独立的主题。但是，所有这些主题都具有一个共同之处：它们演示了常用的PHP-MySQL编程技术。在这里你将不会学习新的函数，而会看到如何使用你已经掌握的知识来创建标准的Web功能。

这些示例本身通过添加新的、流行的特性，拓宽了在上一章开始开发的Web应用程序。你将看到用于管理数据库信息的多种技巧，特别是使用PHP编辑和删除记录。同时，还将介绍把数据传递到PHP页面的两种新方式。本章最后几节向view\_users.php页面中添加了一些特性。

## 10.1 给脚本发送值

在迄今为止的示例中，PHP脚本中接收的所有数据都来自于用户在表单中输入的数据。不过，你可以使用两种不同的方法把变量和值传递给PHP脚本，这两种方法都应该知道。

第一种方法是利用HTML的隐藏输入框类型：

```
<input type="hidden" name="do" value="this" />
```

只要这段代码出现在form标签之间的任意位置，在处理的PHP脚本中，变量\$\_POST['do']将具有this这个值（假定表单使用POST方法）。你已经在本书中利用名为submitted的隐藏输入框使用过这种技术，它用于测试何时应该处理表单。

把值传递给脚本的第二种方法是把值追加到URL上：

[www.example.com/page.php?do=this](http://www.example.com/page.php?do=this)

这种技术模仿了HTML表单的GET方法。对于这个特定的示例，page.php将会接收一个名为

`$_GET['do']`的变量，其值为this。

为了演示这种GET方法的技巧，将会编写最初在上一章中创建的view\_users.php脚本的新版本。这个版本提供用于编辑或删除现有用户记录的页面的链接。这些链接将把用户的ID传递给处理页面，本章也将会编写它们。

### 手动发送值给PHP脚本

- (1) 在文本编辑器或IDE中打开view\_users.php脚本（参见脚本9-6）。
- (2) 更改SQL查询以读取以下代码（参见脚本10-1）。

```
$q = "SELECT last_name, first_name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr, user_id FROM users ORDER BY registration_date ASC";
```

**脚本 10-1** 现在修改了在第9章中开始创建的view\_users.php脚本，使得它会展示Edit和Delete链接，并在每个URL中一起传递用户的ID号

```
1 <?php # Script 10.1 - view_users.php #3
2 // This script retrieves all the records from the users table.
3 // This new version links to edit and delete pages.
4
5 $page_title = 'View the Current Users';
6 include ('includes/header.html');
7 echo '<h1>Registered Users</h1>';
8
9 require_once ('../mysqli_connect.php');
10
11 // Define the query:
12 $q = "SELECT last_name, first_name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr, user_id
13 FROM users ORDER BY registration_date ASC";
14 $r = @mysqli_query ($dbc, $q);
15
16 // Count the number of returned rows:
17 $num = mysqli_num_rows($r);
18
19 if ($num > 0) { // If it ran OK, display the records.
20
21 // Print how many users there are:
22 echo "<p>There are currently $num registered users.</p>\n";
23
24 // Table header:
25 echo '<table align="center" cellspacing="3" cellpadding="3" width="75%">
26 <tr>
27 <td align="left">Edit</td>
28 <td align="left">Delete</td>
29 <td align="left">Last Name</td>
30 <td align="left">First Name</td>
31 <td align="left">Date Registered</td>
32 </tr>
33 ';
34
35 // Fetch and print all the records:
36 while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
```

10

```

36 echo '<tr>
37 <td align="left">Edit</td>
38 <td align="left">Delete</td>
39 <td align="left">' . $row['last_name'] . '</td>
40 <td align="left">' . $row['first_name'] . '</td>
41 <td align="left">' . $row['dr'] . '</td>
42 </tr>
43 ';
44 }
45
46 echo '</table>';
47 mysqli_free_result ($r);
48
49 } else { // If no records were returned.
50 echo '<p class="error">There are currently no registered users.</p>';
51 }
52
53 mysqli_close($dbc);
54
55 include ('includes/footer.html');
56 ?>
```

用两种方式更改了这个查询。首先，单独选择名字和姓氏，而不是把它们连接在一起。其次，现在还选择了user\_id值，因为它是创建这些链接所必需的。

(3) 添加另外3列到主表中。

```

echo '<table align="center" cellspacing="3" cellpadding="3" width="75%">
<tr>
 <td align="left">Edit</td>
 <td align="left">Delete </td>
 <td align="left">Last Name </td>
 <td align="left">First Name </td>
 <td align="left">Date Registered</td>
</tr>
';
```

在这个脚本的前一个版本中，只有两列：一列用于名字，另一列用于用户注册的日期。我把这个名字列分隔成两部分，并且另外创建了两列：一列用于Edit链接，另一列用于Delete链接。

(4) 更改while循环内的echo语句，使之匹配表的新结构。

```

echo '<tr>
 <td align="left">Edit</td>
 <td align="left">Delete</td>
 <td align="left">' . $row['last_name'] . '</td>
 <td align="left">' . $row['first_name'] . '</td>
 <td align="left">' . $row['dr'] . '</td>
</tr>
';
```

对于从数据库中返回的每一条记录，这一行都会打印出一个具有5列的行。后3列很明显并且容易创建：只需引用返回的列名称。

对于前两列，它们提供了用于编辑或删除用户的链接，其语法稍微复杂一些。所期待的最终结果

是像`<a href="edit_user.php?id=X">Edit</a>`这样的HTML代码，其中X是用户的ID。知道这些后，PHP代码必须做的全部工作就是为X打印`$row['user_id']`，要留意引号，以避免解析错误。

由于HTML属性使用了许多双引号，并且这个`echo()`语句需要打印许多变量，我发现最容易的方法是为HTML使用单引号，然后把变量连接到打印的文本。

(5) 将文件另存为`view_users.php`，存放在Web目录中，并在Web浏览器中运行它（参见图10-1）。

<b>Registered Users</b>				
Edit	Delete	Last Name	First Name	Date Registered
Edit	Delete	Ullman	Larry	March 31, 2011
Edit	Delete	Isabella	Zoe	March 31, 2011
Edit	Delete	Starr	Ringo	March 31, 2011
Edit	Delete	Harrison	George	March 31, 2011
Edit	Delete	McCartney	Paul	March 31, 2011
Edit	Delete	Lennon	John	March 31, 2011
Edit	Delete	Brautigan	Richard	March 31, 2011
Edit	Delete	Banks	Russell	March 31, 2011

图10-1 `view_users.php`页面的修订版本，其中具有新的列和链接

(6) 如果愿意，可以查看该页面的HTML源文件，以查看每个动态生成的链接（参见图10-2）。

```

<tr>
 <td align="left">Edit</td>
 <td align="left">Delete</td>
 <td align="left">Isabella</td>
 <td align="left">Zoe</td>
 <td align="left">March 31, 2011</td>
</tr>
<tr>
 <td align="left">Edit</td>
 <td align="left">Delete</td>
 <td align="left">Starr</td>
 <td align="left">Ringo</td>
 <td align="left">March 31, 2011</td>
</tr>
<tr>
 <td align="left">Edit</td>
 <td align="left">Delete</td>
 <td align="left">Harrison</td>
 <td align="left">George</td>
 <td align="left">March 31, 2011</td>
</tr>

```

图10-2 页面（见图10-1）的HTML源文件的一部分显示了如何把用户ID添加到每个链接的URL中

### ✓ 提示

- 为了把多个变量追加到URL中，可以使用下面这种语法：`page.php?name1=value1&name2 = value2&name3=value3`。它只是使用`&`符号以及另一个`name=value`对的简单事情。
- 添加变量到URL中的一种技巧是应该对字符串进行编码，以确保值会被正确处理。例如，字符串Elliott Smith中的空格是有问题的。解决方案是使用`urlencode()`函数：

```
$url = 'page.php?name=' . urlencode ('Elliott Smith');
```

在以编程方式添加值到URL中时，只需要执行该任务。当表单使用GET方法时，它会自动正确地对数据进行编码。

## 10.2 使用隐藏的表单输入框

在上一个示例中，编写了view\_users.php脚本的新版本。这个版本现在包括指向edit\_user.php和delete\_user.php页面的链接，并通过URL传递每个用户的ID。下一个示例即delete\_user.php将获取传递的用户ID，并允许管理员删除那个用户。尽管可以在访问这个页面时只让它简单地执行一个DELETE查询，但出于安全性考虑（并且为了防止漫不经心的删除），其中将包括多个步骤（参见图10-3）。

- (1) 这个页面必须检查它接收到的是一个数字型用户ID。
- (2) 显示一条消息，用于确认应该删除这个用户。
- (3) 将把用户ID存储在一个隐藏的表单输入框中。
- (4) 在提交表单时，将会实际地删除用户。

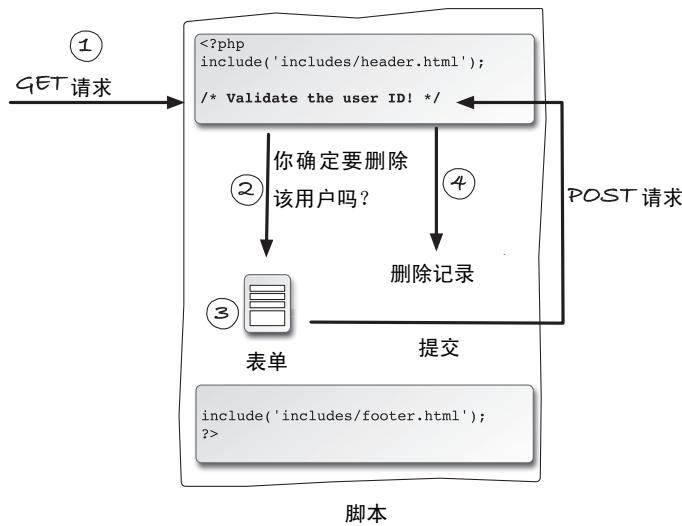


图10-3 此图显示了删除用户脚本要执行的几个步骤

### 使用隐藏的表单输入框

- (1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为delete\_user.php（参见脚本10-2）。

**脚本 10-2** 这个脚本期待通过 URL 将用户 ID 传递给它。然后它会展示一个确认表单，并在提交表单时删除用户

```

1 <?php # Script 10.2 - delete_user.php
2 // This page is for deleting a user record.
3 // This page is accessed through view_users.php.
4
5 $page_title = 'Delete a User';
6 include ('includes/header.html');
```

```
7 echo '<h1>Delete a User</h1>';
8
9 // Check for a valid user ID, through GET or POST:
10 if ((isset($_GET['id'])) && (is_numeric($_GET['id']))) { // From view_users.php
11 $id = $_GET['id'];
12 } elseif ((isset($_POST['id'])) && (is_numeric($_POST['id']))) { // Form submission.
13 $id = $_POST['id'];
14 } else { // No valid ID, kill the script.
15 echo '<p class="error">This page has been accessed in error.</p>';
16 include ('includes/footer.html');
17 exit();
18 }
19
20 require_once ('../mysqli_connect.php');
21
22 // Check if the form has been submitted:
23 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
24
25 if ($_POST['sure'] == 'Yes') { // Delete the record.
26
27 // Make the query:
28 $q = "DELETE FROM users WHERE user_id=$id LIMIT 1";
29 $r = @mysqli_query ($dbc, $q);
30 if (mysqli_affected_rows($dbc) == 1) { // If it ran OK.
31
32 // Print a message:
33 echo '<p>The user has been deleted.</p>';
34
35 } else { // If the query did not run OK.
36 echo '<p class="error">The user could not be deleted due to a system error.</p>'; // Public
 message.
37 echo '<p>' . mysqli_error($dbc) . '
Query: ' . $q . '</p>'; // Debugging message.
38 }
39
40 } else { // No confirmation of deletion.
41 echo '<p>The user has NOT been deleted.</p>';
42 }
43
44 } else { // Show the form.
45
46 // Retrieve the user's information:
47 $q = "SELECT CONCAT(last_name, ' ', first_name) FROM users WHERE user_id=$id";
48 $r = @mysqli_query ($dbc, $q);
49
50 if (mysqli_num_rows($r) == 1) { // Valid user ID, show the form.
51
52 // Get the user's information:
53 $row = mysqli_fetch_array ($r, MYSQLI_NUM);
54
55 // Display the record being deleted:
56 echo "<h3>Name: $row[0]</h3>
57 Are you sure you want to delete this user?" ;
58
59 // Create the form:
```

```
60 echo '<form action="delete_user.php" method="post">
61 <input type="radio" name="sure" value="Yes" /> Yes
62 <input type="radio" name="sure" value="No" checked="checked" /> No
63 <input type="submit" name="submit" value="Submit" />
64 <input type="hidden" name="id" value="' . $id . '" />
65 </form>';
66
67 } else { // Not a valid user ID.
68 echo '<p class="error">This page has been accessed in error.</p>';
69 }
70
71 } // End of the main submission conditional.
72
73 mysqli_close($dbc);
74
75 include ('includes/footer.html');
76 ?>
```

(2) 包括页面标题。

```
$page_title = 'Delete a User';
include ('includes/header.html');
echo '<h1>Delete a User</h1>';
```

这个文档将使用与应用程序中的其他页面相同的模板系统。

(3) 检查有效的用户ID值。

```
if ((isset($_GET['id'])) && (is_numeric($_GET['id']))) {
 $id = $_GET['id'];
} elseif ((isset($_POST['id'])) && (is_numeric($_POST['id']))) {
 $id = $_POST['id'];
} else {
 echo '<p class="error">This page has been accessed in error.</p>';
 include ('includes/footer.html');
 exit();
}
```

这个脚本依赖于具有一个有效的用户ID，在DELETE查询的WHERE子句中将会用到它。第一次访问该页面时，在单击view\_users.php页面中的Delete链接之后，应该把用户ID传递到URL中（该页面的URL将以delete\_user.php?id=X结束）。第一个if条件将检查这样一个值，以及这个值是否是一个数字值。

如你将看到的，该脚本然后将把用户ID值存储在一个隐藏的表单输入框中。当提交这个表单时（返回到这个相同的页面），该页面将通过\$\_POST接收到这个ID。第二个条件会检查这个ID，并且再次检查该ID值是否是一个数字值。

如果这两个条件都不为TRUE，那么将不能继续处理页面，从而会显示一条出错消息，并会终止脚本的执行（参见图10-4）。

(4) 包括MySQL连接脚本。

```
require_once ('../mysqli_connect.php');
```

这个脚本的两个处理过程（显示表单和处理表单）都需要一条数据库连接，因此，这一行位于主提交条件语句（第(5)步）外面。

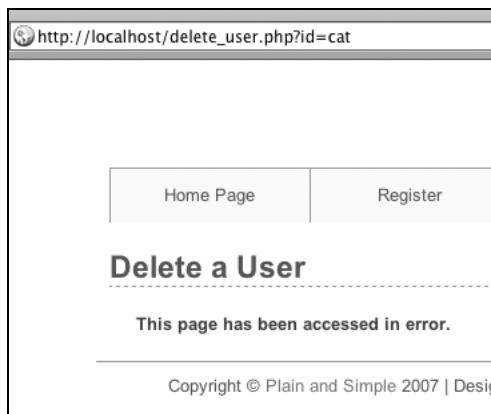


图10-4 如果页面没有接收到一个数字ID值，就会显示这个错误

(5) 开始编写主提交条件语句。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

(6) 如果需要，可删除用户。

```
if ($_POST['sure'] == 'Yes') {
 $q = "DELETE FROM users WHERE user_id=$id LIMIT 1";
 $r = @mysqli_query ($dbc, $q);
```

表单（参见图10-5）将使用户单击一个单选按钮来确认删除。这一小步用于防止任何意外的删除。因此，处理过程首先会检查是否选中了正确的单选按钮。如果是，就会定义一个基本的DELETE查询，它会在WHERE子句中使用用户的ID。在查询中添加了一个LIMIT子句，作为其他的预防措施。

10

图10-5 该页面使用这个简单的表单来确认用户删除

(7) 检查删除操作是否正确工作，并且相应地做出响应。

```
if (mysqli_affected_rows($dbc) == 1) {
 echo '<p>The user has been deleted.</p>';
} else {
 echo '<p class="error">The user could not be deleted due to a system error.</p>';
 echo '<p>' . mysqli_error($dbc) . '
Query: ' . $q . '</p>';
}
```

`mysqli_affected_rows()`函数会检查是否刚好只有一行会受到DELETE查询的影响。如果是这样，就会显示一条愉快的消息（参见图10-6）。如果不是，则会发出一条出错消息。

记住，有可能在没有发生MySQL错误的情况下，没有任何行受到影响。例如，如果查询尝试删除一条记录，其中的用户ID等于42 000（假定不存在这个用户ID），则不会删除任何行，但是没有MySQL

错误发生。尽管如此，由于在第一次加载表单时会执行检查，用户需要进行很多尝试才能达到这一步。

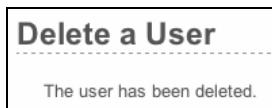


图10-6 如果在表单中选择Yes（参见图10-5）并单击Submit，结果就应该是这样

(8) 完成\$\_POST['sure']条件语句。

```
} else {
 echo '<p>The user has NOT been deleted.</p>';
}
```

如果用户没有明确选中Yes单选按钮，则不会删除用户，并且会显示这条消息（参见图10-7）。

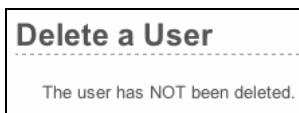


图10-7 如果没有在表单中选择Yes，就不会更改数据库

(9) 开始编写主提交条件语句中的else子句。

```
} else {
```

该页面将会处理表单或显示它。如果提交了表单（如果设置了\$\_SERVER['REQUEST\_METHOD']），则会执行这之前的大部分代码。如果尚未提交表单，则会执行从此开始的代码，在那种情况下，应该显示表单。

(10) 检索所删除用户的信息。

```
$q = "SELECT CONCAT(last_name, ' ', first_name) FROM users WHERE user_id=$id";
$r = @mysql_query ($dbc, $q);
if (mysql_num_rows($r) == 1) {
 $row = mysql_fetch_array ($r, MYSQLI_NUM);
```

为了确认脚本接收到有效的用户ID并准确指出正被删除的用户（回过头参见图10-5），可从数据库中检索待删除用户的名字（参见图10-8）。

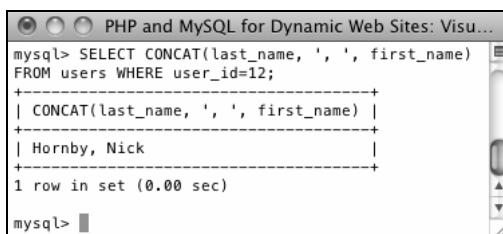


图10-8 在MySQL客户端中运行相同的SELECT查询

条件语句（检查是否只返回一行）确保提供了有效的用户ID。

(11) 显示被删除的记录。

```
echo "<h3>Name: $row[0]</h3>
Are you sure you want to delete this user?";
```

为了防止意外删除错误记录，首先显示要删除的用户名。该值存储在\$row[0]，因为`mysqlifetch_array()`函数（在第(10)步）使用了`MYSQLI_NUM`常量，因而设定\$row返回的结果是一个索引数组。用户名是第一个也是唯一返回的列，所以它的索引是0（数组通常以索引0开始）。

(12) 创建表单：

```
echo '<form action="delete_user.php" method="post">
<input type="radio" name="sure" value="Yes" /> Yes
<input type="radio" name="sure" value="No" checked="checked" /> No
<input type="submit" name="submit" value="Submit" />
<input type="hidden" name="id" value="' . $id . '" />
</form>';
```

表单提交到同一个页面。它包含两个名字相同但值不同的单选按钮，一个提交按钮和一个隐藏的文本域。这里最重要的一步是用户ID被存储于隐藏的表单文本域中，所以处理过程仍然可以访问这个值（参见图10-9）。

```
ript 9.1 - header.html --><h1>Delete a User</h1><h3>Name: Nesmith, mike</h3>
Are you sure you want to delete this user?<form action="delete_user.php" method="post">
<input type="radio" name="sure" value="Yes" /> Yes
<input type="radio" name="sure" value="No" checked="checked" /> No
<input type="submit" name="submit" value="Submit" />
<input type="hidden" name="id" value="10" />
</form><!-- Script 3.3 - footer.html -->
```

图10-9 将用户ID存储为隐藏的输入框，使得在提交表单时可以使用它

(13) 完成`mysqli_num_rows()`条件语句。

```
} else {
 echo '<p class="error">This page has been accessed in error.</p>';
}
```

如果SELECT查询没有返回任何记录（由于提交了无效的用户ID），则会显示这条消息。

如果在测试这个脚本时看到这条消息，但是不理解为什么会这样，可以应用第7章末尾概括的标准调试步骤。

(14) 完成PHP页面。

```
}
mysqli_close($dbc);
include ('includes/footer.html');
?>
```

右花括号完成主提交条件语句。然后，关闭MySQL连接，并包括脚注。

(15) 将文件另存为`delete_user.php`，存放在Web目录中（应该将其放置在与`view_users.php`相同的目录中）。

(16) 首先在`view_users.php`页面中单击Delete链接来运行页面。

### ✓ 提示

- 隐藏的表单元素不会显示在Web浏览器中，但是仍会存在于HTML源代码中（图10-9）。由于这个原因，永远不要存储任何必须保持真正安全的内容。
- 使用隐藏的表单输入框以及把值追加到URL中只是使数据可供其他PHP页面使用的两种方式。在第12章中将会详细介绍另外两种方法——cookie和会话。

## 10.3 编辑现有的记录

数据库驱动的Web站点的一个惯例是在适当的位置具有一个系统，使得可以轻松编辑现有的记录。这个概念似乎有些使许多初级程序员畏缩不前，但是它惊人地直观。对下面这个示例——编辑注册的用户记录——这个过程组合了你应该已经具有的4种技能：

- 建立黏性表单；
- 使用隐藏的输入框；
- 验证注册数据；
- 运行简单的查询。

下一个示例总体上与`delete_user.php`非常相似，并且也在`view_users.php`脚本中建立了对它的链接（当单击Edit时）。表单在显示时，会带有用户的当前信息，从而允许更改那些值（参见图10-10）。在提交表单时，如果数据传递了所有的验证例程，就会运行一个`UPDATE`查询，以更新数据库。

图10-10 用于编辑用户记录的表单

### 编辑现有的数据库记录

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为`edit_user.php`（参见脚本10-3）。

**脚本 10-3 edit\_user.php** 页面首先在表单中显示用户的当前信息。在提交表单时，将会更新数据库中的记录

```

1 <?php # Script 10.3 - edit_user.php
2 // This page is for editing a user record.
3 // This page is accessed through view_users.php.
4
5 $page_title = 'Edit a User';

```

```
1 include ('includes/header.html');
2 echo '<h1>Edit a User</h1>';
3
4 // Check for a valid user ID, through GET or POST:
5 if ((isset($_GET['id'])) && (is_numeric($_GET['id']))) { // From view_users.php
6 $id = $_GET['id'];
7 } elseif ((isset($_POST['id'])) && (is_numeric($_POST['id']))) { // Form submission.
8 $id = $_POST['id'];
9 } else { // No valid ID, kill the script.
10 echo '<p class="error">This page has been accessed in error.</p>';
11 include ('includes/footer.html');
12 exit();
13 }
14
15 require_once ('../mysqli_connect.php');
16
17 // Check if the form has been submitted:
18 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
19
20 $errors = array();
21
22 // Check for a first name:
23 if (empty($_POST['first_name'])) {
24 $errors[] = 'You forgot to enter your first name.';
25 } else {
26 $fn = mysqli_real_escape_string($dbc, trim($_POST['first_name']));
27 }
28
29 // Check for a last name:
30 if (empty($_POST['last_name'])) {
31 $errors[] = 'You forgot to enter your last name.';
32 } else {
33 $ln = mysqli_real_escape_string($dbc, trim($_POST['last_name']));
34 }
35
36 // Check for an email address:
37 if (empty($_POST['email'])) {
38 $errors[] = 'You forgot to enter your email address.';
39 } else {
40 $e = mysqli_real_escape_string ($dbc, trim($_POST['email']));
41 }
42
43 if (empty($errors)) { // If everything's OK.
44
45 // Test for unique email address:
46 $q = "SELECT user_id FROM users WHERE email='$e' AND user_id != $id";
47 $r = @mysqli_query($dbc, $q);
48 if (mysqli_num_rows($r) == 0) {
49
50 // Make the query:
51 $q = "UPDATE users SET first_name='$fn', last_name='$ln', email='$e' WHERE user_id=$id
52 LIMIT 1";
53 $r = @mysqli_query ($dbc, $q);
54 if (mysqli_affected_rows($dbc) == 1) { // If it ran OK.
55
56
57
58 }
```

```
59 // Print a message:
60 echo '<p>The user has been edited.</p>';
61
62 } else { // If it did not run OK.
63 echo '<p class="error">The user could not be edited due to a system error. We apologize for
64 any inconvenience.</p>; // Public message.
65 echo '<p>' . mysqli_error($dbc) . '
Query: ' . $q . '</p>; // Debugging message.
66 }
67
68 } else { // Already registered.
69 echo '<p class="error">The email address has already been registered.</p>';
70 }
71
72 } else { // Report the errors.
73
74 echo '<p class="error">The following error(s) occurred:
';
75 foreach ($errors as $msg) { // Print each error.
76 echo " - $msg
\n";
77 }
78 echo '</p><p>Please try again.</p>';
79
80 } // End of if (empty($errors)) IF.
81
82 } // End of submit conditional.
83
84 // Always show the form...
85
86 // Retrieve the user's information:
87 $q = "SELECT first_name, last_name, email FROM users WHERE user_id=$id";
88 $r = @mysqli_query ($dbc, $q);
89
90 if (mysqli_num_rows($r) == 1) { // Valid user ID, show the form.
91
92 // Get the user's information:
93 $row = mysqli_fetch_array ($r, MYSQLI_NUM);
94
95 // Create the form:
96 echo '<form action="edit_user.php" method="post">
97 <p>First Name: <input type="text" name="first_name" size="15" maxlength="15" value="' . $row[0] . '"></p>
98 <p>Last Name: <input type="text" name="last_name" size="15" maxlength="30" value="' . $row[1] . '"></p>
99 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="' . $row[2] . '"></p>
100 <p><input type="submit" name="submit" value="Submit" /></p>
101 <input type="hidden" name="id" value="' . $id . '" />
102 </form>';
103
104 } else { // Not a valid user ID.
105 echo '<p class="error">This page has been accessed in error.</p>';
106 }
107
108 mysqli_close($dbc);
```

```

109
110 include ('includes/footer.html');
111 ?>

```

(2) 检查有效的用户ID值。

```

if ((isset($_GET['id'])) && (is_numeric($_GET['id']))) {
 $id = $_GET['id'];
} elseif ((isset($_POST['id'])) && (is_numeric($_POST['id']))) {
 $id = $_POST['id'];
} else {
 echo '<p class="error">This page has been accessed in error.</p>';
 include ('includes/footer.html');
 exit();
}

```

这个验证例程与delete\_user.php中的那个验证例程完全一样，用于确认是先从view\_users.php访问页面时（第一个条件）还是在提交表单时（第二个条件）接收到一个数字型用户ID。

(3) 包括MySQL连接脚本，并开始编写主提交条件语句。

```

require_once ('../mysqli_connect.php');
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 $errors = array();
}

```

与你已经完成的注册示例一样，这个脚本利用一个数组来跟踪错误。

(4) 验证名字。

```

if (empty($_POST['first_name'])) {
 $errors[] = 'You forgot to enter your first name.';
} else {
 $fn = mysqli_real_escape_string($dbc, trim($_POST['first_name']));
}

```

这个表单（参见图10-10）类似于注册表单，但是没有密码框。因此，可以用注册脚本中使用的相同方法来验证表单数据。与注册示例一样，清理验证数据，然后用mysqli\_real\_escape\_string()函数处理它以确保安全性。

(5) 验证姓氏和电子邮件地址。

```

if (empty($_POST['last_name'])) {
 $errors[] = 'You forgot to enter your last name.';
} else {
 $ln = mysqli_real_escape_string($dbc, trim($_POST['last_name']));
}
if (empty($_POST['email'])) {
 $errors[] = 'You forgot to enter your email address.';
} else {
 $e = mysqli_real_escape_string($dbc, trim($_POST['email']));
}

```

(6) 如果没有错误，就检查提交的电子邮件地址是否尚未使用。

```

if (empty($errors)) {
 $q = "SELECT user_id FROM users WHERE email='$e' AND user_id != $id";
 $r = @mysqli_query($dbc, $q);
 if (mysqli_num_rows($r) == 0) {

```

作为一个整体的数据库和应用程序的完整性部分依赖于在users表中具有唯一的电子邮件地址值。这个要求保证登录系统（它使用电子邮件地址和密码的组合，将在第12章中开发）将会工作。由于表单允许改变用户的电子邮件地址（见图10-10），所以必须采取特殊的步骤来保证唯一性。为了理解这个查询，考虑两种可能的情况……

在第一种情况下，正在更改用户的电子邮件地址。在这种情况下，你只需运行一个查询，确保尚未注册特定的电子邮件地址（即SELECT user\_id FROM users WHERE email='\$e'）。

在第二种情况下，用户的电子邮件地址将保持相同。在这种情况下，如果电子邮件地址已经在使用，那么一切正常，因为已经为该用户在使用这个电子邮件地址。

要编写一个在这两种情况下都可工作的查询，不要执行检查以查看电子邮件地址是否在使用，而要查看它是否被别人使用，因此编写如下代码：

```
SELECT user_id FROM users WHERE
email='$e' AND user_id != $id
```

如果这条查询没有返回任何记录，就可以运行UPDATE更新数据了。

(7) 更新数据库。

```
$q = "UPDATE users SET first_name='$fn', last_name='$ln', email='$e' WHERE user_id=$id LIMIT 1";
$r = @mysql_query ($dbc, $q);
```

这个UPDATE查询类似于你在第5章中看到的示例。该查询使用表单提交的值来更新全部3个字段——名字、姓氏和电子邮件地址。这个系统将会工作，因为会用现有的值预先设置表单。因此，如果只在表单中编辑名字，而不编辑任何其他内容，则会使用这个新值更新数据库中的名字值，但是对于姓氏和电子邮件地址的值，则会使用它们的当前值来“更新”它们。与尝试确定哪些表单值已经发生变化并且只在数据库中更新这些值相比，这个系统要容易得多。

(8) 报告更新的结果。

```
if (mysql_affected_rows($dbc) == 1) {
 echo '<p>The user has been edited.</p>';
} else {
 echo '<p class="error">The user could not be edited due to a system error. We apologize for any
 inconvenience.</p>';
 echo '<p>' . mysql_error($dbc) . '
Query: ' . $q . '</p>';
}
```

mysql\_affected\_rows()函数将返回数据库中受最近的查询影响的行数。如果三个表单值中任何一个被改变，那么这个函数应该返回值1。这个条件语句测试这种情况，并打印一条消息指示成功或失败。

记住：如果UPDATE命令成功运行，mysql\_affected\_rows()函数将会返回值0，但是实际上不会影响任何记录。因此，如果你提交了这个表单，而没有更改任何表单值，就会显示一个系统错误，它在技术上可能不正确。一旦使这个脚本有效地工作，就可以更改出错消息，指示如果mysql\_affected\_rows()函数返回值0则没有执行更改。

(9) 完成电子邮件条件语句。

```
} else {
 echo '<p class="error">The email address has already been registered.</p>';
}
```

这个else完成条件语句，它用于检查某个电子邮件地址是否已经被另一个用户使用。如果是，就打印该消息。

(10) 完成\$errors和提交条件语句。

```
} else {
echo '<p class="error">The following error(s) occurred:
';
foreach ($errors as $msg) {
 echo " - $msg
\n";
}
echo '</p><p>Please try again.</p>';
}
```

第一个else语句用于报告表单中的任何错误（即缺少姓氏、名字或电子邮件地址）。最后一个右花括号完成主提交条件语句。

(11) 完成提交的条件语句。

```
} // End of submit conditional.
```

最后的结束大括号完成了主要的提交条件语句。在这个例子中，每当页面被访问时都将显示表单。提交表单后，数据库将会更新，表单将再次显示并显示最新的信息。

(12) 检索正被编辑的用户的信息。

```
$q = "SELECT first_name, last_name, email FROM users WHERE user_id=$id";
$r = @mysqli_query ($dbc, $q);
if (mysqli_num_rows($r) == 1) {
 $row = mysqli_fetch_array ($r, MYSQLI_NUM);
```

为了预先填充表单元素，必须从数据库中检索用户的当前信息。这个查询类似于delete\_user.php中的一个查询。这个条件语句（检查是否返回了单独一行）确保提供了有效的用户ID。

(13) 显示表单。

```
echo '<form action="edit_user.php" method="post">
<p>First Name: <input type="text" name="first_name" size="15" maxlength="15" value="' . $row[0] . '" /></p>
<p>Last Name: <input type="text" name="last_name" size="15" maxlength="30" value="' . $row[1] . '" /></p>
<p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="' . $row[2] . '" /> </p>
<p><input type="submit" name="submit" value="Submit" /></p>
<input type="hidden" name="id" value="' . $id . '" />
</form>';
```

这个表单仅仅具有3个文本输入框，它们都使用从数据库中检索的数据被制作成黏性的。同样，把用户ID（\$id）存储为隐藏的表单输入框，使得处理过程中也可以访问这个值。

(14) 完成mysqli\_num\_rows()条件语句。

```
} else {
echo '<p class="error">This page has been accessed in error.</p>';
}
```

如果没有从数据库中返回任何记录，那是因为提交了一个无效的用户ID，所以会显示这条消息。

(15) 完成PHP页面。

```
mysqli_close($dbc);
include ('includes/footer.html');
?>
```

(16) 将文件另存为edit\_user.php，并将其存放在Web目录中(在与view\_users.php相同的文件夹中)。

(17) 首先单击view\_users.php页面中的Edit链接来运行这个页面(参见图10-11和图10-12)。

The user has been edited.

First Name:

Last Name:

Email Address:

图10-11 在成功更新数据库之后，就会在表单中显示新的值(对比图10-10中的表单值)

The email address has already been registered.

First Name:

Last Name:

Email Address:

图10-12 如果尝试将一条记录更改为现有的电子邮件地址或者如果漏掉了某个输入，就会报告错误

### ✓ 提示

- 如书中所介绍的那样，黏性表单总会显示从数据库中检索的值。这意味着如果发生错误，将会使用数据库值，而不是用户刚刚输入的值(如果它们不同的话)。为了改变这种行为，黏性表单必须检查\$\_POST变量是否存在，如果存在，就使用它们，否则就使用数据库值。
- 这个编辑页面不包括更改密码的功能。在password.php(参见脚本9-7)中已经演示了这个概念。如果你想在此纳入这个功能，请记住，你不能显示当前密码，因为它是加密的。可以只展示两个文本框用于更改密码(新密码输入和确认)。如果提交了这些值，还会更新数据库中的密码。如果这些输入框保持为空，则不会更新数据库中的密码。

## 10.4 给查询结果标页码

标页码 (pagination) 是一个你熟悉的概念，即使你不知道这个术语。当你使用像Google这样的搜索引擎时，它会把结果显示为一系列页面，而不是一份长长的列表。`view_users.php`脚本可以受益于这个相同的特性。

给查询结果标页码会广泛使用第5章中介绍的LIMIT SQL子句。LIMIT限制了实际上会返回匹配记录的那个子集。为了给查询的返回结果标页码，每个页面将使用不同的LIMIT参数运行相同的查询。因此，第一页将请求前 $X$ 条记录，第二页将请求第二组的 $X$ 条记录，依次类推。为了使之工作，需要在URL中把一个指示符从一个页面传递到另一个页面，用于指示页面应该显示哪些记录，如从`view_users.php`页面传递的用户ID。

这里将演示另一种更具装饰性的技术：使用交替变换的背景色来显示表中的每一行——每一条返回的记录（参见图10-13）。使用三元运算符（见框注“三元运算符”）可以轻松得到这种效果。

Registered Users				
Edit	Delete	Last Name	First Name	Date Registered
Edit	Delete	Ullman	Larry	March 31, 2011
Edit	Delete	Isabella	Zoe	March 31, 2011
Edit	Delete	Starr	Ringo	March 31, 2011
Edit	Delete	Harrison	George	March 31, 2011
Edit	Delete	McCartney	Paul	March 31, 2011
Edit	Delete	Lennon	John	March 31, 2011
Edit	Delete	Brautigan	Richard	March 31, 2011
Edit	Delete	Banks	Russell	March 31, 2011
Edit	Delete	Simpson	Homer	March 31, 2011
Edit	Delete	Simpson	Marge	March 31, 2011

1 2 3 Next

10

图10-13 交替显示表中各行的颜色，使得这份用户列表更易阅读  
(每隔一行具有一种淡灰色背景)

### 三元运算符

这个示例将使用一种以前未介绍的运算符，称为三元 (ternary) 运算符。其结构如下：

`(condition) ? valueT : valueF`

将会计算圆括号中的条件。如果它为TRUE，则返回第一个值 (`valueT`)。如果该条件为FALSE，则返回第二个值 (`valueF`)。

由于三元运算符返回一个值，所以整个结构通常用于给变量赋值或者用作函数的一个参数。例如，下面这一行代码：

```
echo (isset($var)) ? 'SET' : 'NOT SET';
```

将打印出SET或NOT SET，这依赖于变量\$var的状态。

在`view_users.php`脚本的这个版本中，三元运算符会给一个变量赋值，这个值不同于它的当前值。该变量本身然后将用于指示表中每一条记录的背景色。当然，还有许多其他的方式可用于设置这个值，但是，三元运算符是最简洁的。

这里有许多良好的新信息，因此要认真阅读它们，并确保你的脚本与之完全匹配。为了更容易做到这一点，让我们从头开始编写脚本的这个版本，而不是尝试修改脚本 10-1。

#### 给 view\_users.php 标页码

(1) 在文本编辑器或 IDE 中创建一个新的 PHP 文档，命名为 view\_users.php（参见脚本 10-4）。

**脚本 10-4 view\_users.php 的这个新版本纳入了标页码功能，使得在多个 Web 浏览器页面上列出用户**

```

1 <?php # Script 10.4 - view_users.php #4
2 // This script retrieves all the records from the users table.
3 // This new version paginates the query results.
4
5 $page_title = 'View the Current Users';
6 include ('includes/header.html');
7 echo '<h1>Registered Users</h1>';
8
9 require_once ('../mysqli_connect.php');
10
11 // Number of records to show per page:
12 $display = 10;
13
14 // Determine how many pages there are...
15 if (isset($_GET['p']) && is_numeric ($_GET['p'])) { // Already been determined.
16
17 $pages = $_GET['p'];
18
19 } else { // Need to determine.
20
21 // Count the number of records:
22 $q = "SELECT COUNT(user_id) FROM users";
23 $r = @mysqli_query ($dbc, $q);
24 $row = @mysqli_fetch_array ($r, MYSQLI_NUM);
25 $records = $row[0];
26
27 // Calculate the number of pages...
28 if ($records > $display) { // More than 1 page.
29 $pages = ceil ($records/$display);
30 } else {
31 $pages = 1;
32 }
33
34 } // End of p IF.
35
36 // Determine where in the database to start returning results...
37 if (isset($_GET['s']) && is_numeric ($_GET['s'])) {
38 $start = $_GET['s'];
39 } else {
40 $start = 0;
41 }
42
43 // Define the query:
44 $q = "SELECT last_name, first_name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr,
45 user_id FROM users ORDER BY registration_date ASC LIMIT $start, $display";
46 $r = @mysqli_query ($dbc, $q);

```

```

46
47 // Table header:
48 echo '<table align="center" cellspacing="0" cellpadding="5" width="75%">
49 <tr>
50 <td align="left">Edit</td>
51 <td align="left">Delete</td>
52 <td align="left">Last Name</td>
53 <td align="left">First Name</td>
54 <td align="left">Date Registered
55 </td>
56 </tr>
57 ';
58 // Fetch and print all the records
59
60 $bg = '#eeeeee'; // Set the initial background color.
61
62 while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
63
64 $bg = ($bg=='#eeeeee' ? '#ffffff' : '#eeeeee'); // Switch the background color.
65
66 echo '<tr bgcolor="' . $bg . '">
67 <td align="left">Edit</td>
68 <td align="left">Delete</td>
69 <td align="left">' . $row['last_name'] . '</td>
70 <td align="left">' . $row['first_name'] . '</td>
71 <td align="left">' . $row['dr'] . '</td>
72 </tr>
73 ';
74 }
75 // End of WHILE loop.
76
77 echo '</table>';
78 mysqli_free_result ($r);
79 mysqli_close($dbc);
80
81
// Make the links to other pages, if necessary.
82 if ($pages > 1) {
83
84 // Add some spacing and start a paragraph:
85 echo '
<p>';
86
87 // Determine what page the script is on:
88 $current_page = ($start/$display) + 1;
89
90 // If it's not the first page, make a Previous link:
91 if ($current_page != 1) {
92 echo 'Previous
93 ';
94 }
95
// Make all the numbered pages:
96 for ($i = 1; $i <= $pages; $i++) {
97 if ($i != $current_page) {

```

```

98 echo '' .
99 $i . ' ';
100 } else {
101 echo $i . ' ';
102 } // End of FOR loop.
103
104 // If it's not the last page, make a Next button:
105 if ($current_page != $pages) {
106 echo 'Next';
107 }
108
109 echo '</p>'; // Close the paragraph.
110
111 } // End of links section.
112
113 include ('includes/footer.html');
114 ?>

```

(2) 设置每一页上要显示的记录数。

```
$display = 10;
```

这里通过把这个值建立为一个变量，以后就可以轻松地更改每一页上显示的记录数。此外，这个值还将在这个脚本中使用多次，因此最好把它展示为单个变量。

(3) 检查是否确定了需要的页面数量。

```
if (isset($_GET['p']) && is_numeric($_GET['p'])) {
 $pages = $_GET['p'];
} else {
```

为了让这个脚本在多个页面上显示用户，需要确定所需的结果页面的总数。第一次运行这个脚本时，就必须计算出这个数字。对于这个页面的每个后续调用，都会在URL中把这个页面总数传递给脚本，因此可以在\$\_GET['p']中得到它。如果设置了这个变量并且它是一个数字，就会把它的值赋予\$pages变量。如果没有设置，就需要计算页面数量。

(4) 计算数据库中的记录数。

```
$q = "SELECT COUNT(user_id) FROM users";
$r = @mysql_query ($dbc, $q);
$row = @mysql_fetch_array ($r, MYSQLI_NUM);
$records = $row[0];
```

使用第8章中介绍的COUNT()函数，可以轻松查看users表中的记录数。这个查询将返回单独一行，它只有一列，即记录数（参见图10-14）。

The screenshot shows a MySQL command-line interface window titled 'PHP and MySQL for Dynamic Web Si...'. The command entered is 'mysql> SELECT COUNT(user\_id) FROM users;'. The output shows a single row with the value '25' in the 'COUNT(user\_id)' column. The status bar at the bottom indicates '1 row in set (0.00 sec)'. The MySQL prompt 'mysql>' is visible at the bottom.

图10-14 在MySQL客户端中运行统计查询的结果

(5) 以数学方式计算需要多少页。

```
if ($records > $display) {
 $pages = ceil($records/$display);
} else {
 $pages = 1;
}
```

用于显示记录的页面数量基于要显示的记录总数和每一页上显示的记录数（通过\$display变量设置）。如果行数多于每一页上显示的记录数，则需要多个页面。为了准确计算出需要多少个页面，可以取两个数相除之后的下一个最大整数（ceil()函数返回下一个最大整数）。例如，如果返回了25条记录，那么将需要3个页面（第一个页面显示10条记录，第二个页面显示10条记录，第三个页面显示5条记录）。如果\$records不大于\$display，则只需要一个页面。

(6) 完成判断页数的if\_else。

```
} // End of p IF.
```

(7) 确定数据库中的起点。

```
if (isset($_GET['s']) && is_numeric($_GET['s'])) {
 $start = $_GET['s'];
} else {
 $start = 0;
}
```

这个脚本将接收的第二个参数——在后来查看页面时——将是起始记录。它对应于LIMIT x, y子句中的第一个数字。在最初调用该脚本时，应该会检索前10条记录（因为\$display的值为10）。第二个页面将显示第10~19条记录，第三个页面则会显示第20~29条记录，依次类推。

在第一次访问这个页面时，没有设置\$\_GET['s']变量，因此\$start应该为0（LIMIT子句中的第一条记录的索引为0）。后续的页面将从URL中接收\$\_GET['s']变量，并会将其赋予\$start。

(8) 更改这个查询，以让它使用LIMIT子句。

```
$q = "SELECT last_name, first_name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr, user_id FROM users
 ORDER BY registration_date ASC LIMIT $start, $display";
$r = @mysql_query ($dbc, $q);
```

LIMIT子句规定开始检索哪一条记录（\$start），以及从此返回多少条记录（\$display）。第一次运行这个页面时，查询将会是SELECT last\_name, first\_name ... LIMIT 0, 10。单击到达下一个页面，将会产生SELECT last\_name, first\_name ... LIMIT 10, 10。

(9) 创建HTML表头。

```
echo '<table align="center" cellspacing="0" cellpadding="5" width="75%">
<tr>
 <td align="left">Edit </td>
 <td align="left">Delete </td>
 <td align="left">Last Name </td>
 <td align="left">First Name </td>
 <td align="left">Date Registered</td>
</tr>
';
```

为了对这个脚本进行一点简化，我假定存在要显示的记录。为了显得更正式，在创建表之前，这

个脚本将调用`mysqli_num_rows()`函数，并且具有一个条件语句，用于确认返回了一些记录。

(10) 初始化背景色变量。

```
$bg = '#eeeeee';
```

为了使每一行都具有它自己的背景色，将使用一个变量来存储此颜色。首先，赋予`$bg`变量一个值`#eeeeee`，即淡灰色。然后，将此颜色与白色（#ffffff）交替显示。

(11) 开始编写`while`循环，用于检索每一条记录。

```
while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
 $bg = ($bg=='#eeeeee' ? '#ffffff' : '#eeeeee');
```

将表中每一行使用的背景色赋予`$bg`变量。因为想让颜色交替显示，所以使用这一行代码把相对的颜色赋予`$bg`。如果它等于`#eeeeee`，那么将赋予它`#ffffff`这个值，反之亦然（同样，参见框注“三元运算符”，以了解三元运算符的语法和解释）。对于第一行，`$bg`等于`#eeeeee`，因此将被赋予`#ffffff`，从而建立一种白色背景。对于第二行，`$bg`不等于`#eeeeee`，因此将给它赋那个值，从而建立一种灰色背景。

(12) 在表格行中打印记录。

```
echo '<tr bgcolor="' . $bg . '">
 <td align="left">Edit</td>
 <td align="left">Delete</td>
 <td align="left">' . $row['last_name'] . '</td>
 <td align="left">' . $row['first_name'] . '</td>
 <td align="left">' . $row['dr'] . '</td>
</tr>
';
```

这段代码只在一个方面与这个脚本的以前版本有所不同。初始`TR`标签现在包括`bgcolor`属性，它的值将是`$bg`变量（因此它的值将交替为`#eeeeee`和`#ffffff`）。

(13) 完成`while`循环和表格，释放查询结果资源，并关闭数据库连接。

```
} // End of WHILE loop.
echo '</table>';
mysqli_free_result ($r);
mysqli_close($dbc);
```

(14) 如果需要，开始编写下一部分代码，用于显示指向其他页面的链接。

```
if ($pages > 1) {
 echo '
<p>';
```

(15) 确定当前正在查看的页面。

```
$current_page = ($start/$display) + 1;
```

要创建连接，脚本必须首先确定当前的页面。这可以通过开始的数目除以显示的数目再加上1来计算。例如，在这个页面的第二页，`$start`将会是10（因为在第一个实例中`$start`是0），使得`$current_page`值为2：(10/10) + 1 = 2。

(16)如果有需要，创建回到前一页的链接。

```
if ($current_page != 1) {
 echo 'Previous ';
```

如果当前页面不是第一个页面，它还需要一个Previous链接，指向早期的结果集（参见图10-15）。并不需要严格如此，但这样做很好。



图10-15 仅当当前页面不是第一个页面时，才会显示Previous链接

每个链接都由脚本名称以及起点和页面数量组成。前一页的起点将是当前起点减去要显示的数量。必须在每个链接中传递这些值，否则标页码将会失败。

#### (17) 建立数字链接。

```
for ($i = 1; $i <= $pages; $i++) {
 if ($i != $current_page) {
 echo '' . $i . '';
 } else {
 echo $i . ' ';
 }
}
```

这段代码通过从1到页面总数的循环来创建一批链接。除了当前页面外，每个页面都会被链接。

对于每个页面链接的起始值s，是通过将要显示的页数乘以*\$i-1*得到的。比如，如果是第3页，那么*\$i-1*就是2，s的值就是20。

#### (18) 创建Next链接。

```
if ($current_page != $pages) {
 echo 'Next';
}
```

最后，将显示Next页面链接，假定它不是最终页面（参见图10-16）。

Registered Users			
Edit	Delete	Last Name	First Name
Edit	Delete	Bank	Melissa
Edit	Delete	Morrison	Toni
Edit	Delete	Franzen	Jonathan
Edit	Delete	DeLillo	Don
Edit	Delete	Doe	Jane

Previous 1 2 3

图10-16 最终的结果页面将不会显示Next链接（对比图10-13和图10-15）

#### (19) 完成页面。

```
echo '</p>';
} // End of links section.
include ('includes/footer.html');
?>
```

(20) 将文件另存为view\_users.php，将其存放在Web目录中，并在Web浏览器中测试它。

### ✓ 提示

- 这个示例对简单的查询标页码，但是，如果你想对更复杂的查询（比如查找的结果）标页码，它并不会复杂得多。它们的主要区别是：无论在查询中使用什么项目，都必须在链接中把它们从一个页面传递到另一个页面。如果主查询并不是从页面的一个视图到下一个视图，标页码将会失败。
- 如果运行这个示例，并且标页码命令并不与应该返回的结果数匹配（例如，统计查询指示有150条记录，但是标页码命令只创建3个页面，每个页面上显示10条记录），最有可能的原因是主查询与COUNT()查询区别太大。这两个查询永远不会相同，但是它们必须执行相同的联结（如果合适的话），并且必须具有完全相同的WHERE和/或GROUP BY子句。
- 这个脚本中没有包括错误处理，因为我知道所编写的查询函数。如果你有什么问题，请记住你的MySQL/SQL调试步骤：打印查询，使用MySQL客户端或phpMyAdmin运行它以确认结果，以及根据需要调用mysqli\_error()函数。

## 10.5 建立可排序的显示结果

在本章最后，可以向view\_users.php脚本中添加最后一个特性。在当前的状态中，按用户注册的日期来显示他们的列表。如果还能够按名字查看他们，当然也不错。

从MySQL的角度讲，能够轻而易举地完成这个任务：只需更改ORDER BY子句即可。因此，我需要做的所有工作是在PHP中添加一些功能来更改ORDER BY子句。执行该任务的合理方式是链接列标题，使得单击它们可以更改显示顺序。正如你可能猜测到的，这涉及使用GET方法将参数传递回这个页面，以指示首选的排序顺序。

### 建立可排序的链接

- (1) 在文本编辑器或IDE中打开view\_users.php（参见脚本10-4）。
- (2) 在确定起点之后，定义\$sort变量（参见脚本10-5）。

**脚本 10-5 view\_users.php 脚本的这个最新的版本会从表的列标题里面创建可单击的链接**

```

1 <?php # Script 10.5 - view_users.php #5
2 // This script retrieves all the records from the users table.
3 // This new version allows the results to be sorted in different ways.
4
5 $page_title = 'View the Current Users';
6 include ('includes/header.html');
7 echo '<h1>Registered Users</h1>';
8
9 require_once ('../mysqli_connect.php');
10
11 // Number of records to show per page:
12 $display = 10;
13
14 // Determine how many pages there are...
15

```

```
if (isset($_GET['p']) && is_numeric($_GET['p'])) { // Already been determined.
16 $pages = $_GET['p'];
17 } else { // Need to determine.
18 // Count the number of records:
19 $q = "SELECT COUNT(user_id) FROM users";
20 $r = @mysqli_query ($dbc, $q);
21 $row = @mysql_fetch_array ($r, MYSQLI_NUM);
22 $records = $row[0];
23 // Calculate the number of pages...
24 if ($records > $display) { // More than 1 page.
25 $pages = ceil ($records/$display);
26 } else {
27 $pages = 1;
28 }
29 } // End of p IF.
30
31 // Determine where in the database to start returning results...
32 if (isset($_GET['s']) && is_numeric($_GET['s'])) {
33 $start = $_GET['s'];
34 } else {
35 $start = 0;
36 }
37
38 // Determine the sort...
39 // Default is by registration date.
40 $sort = (isset($_GET['sort'])) ? $_GET['sort'] : 'rd';
41
42 // Determine the sorting order:
43 switch ($sort) {
44 case 'ln':
45 $order_by = 'last_name ASC';
46 break;
47 case 'fn':
48 $order_by = 'first_name ASC';
49 break;
50 case 'rd':
51 $order_by = 'registration_date ASC';
52 break;
53 default:
54 $order_by = 'registration_date ASC';
55 $sort = 'rd';
56 break;
57 }
58
59 // Define the query:
60 $q = "SELECT last_name, first_name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr, user_id
 FROM users ORDER BY $order_by LIMIT $start, $display";
61 $r = @mysqli_query ($dbc, $q); // Run the query.
62
63 // Table header:
```

```

64 echo '<table align="center" cellspacing="0" cellpadding="5" width="75%">
65 <tr>
66 <td align="left">Edit</td>
67 <td align="left">Delete</td>
68 <td align="left">Last Name</td>
69 <td align="left">First Name</td>
70 <td align="left">Date Registered</td>
71 </tr>
72 ';
73
74 // Fetch and print all the records....
75 $bg = '#eeeeee';
76 while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
77 $bg = ($bg=='#eeeeee' ? '#ffffff' : '#eeeeee');
78 echo '<tr bgcolor="' . $bg . '">
79 <td align="left">Edit</td>
80 <td align="left">Delete</td>
81 <td align="left">' . $row['last_name'] . '</td>
82 <td align="left">' . $row['first_name'] . '</td>
83 <td align="left">' . $row['dr'] . '</td>
84 </tr>
85 ';
86 } // End of WHILE loop.
87
88 echo '</table>';
89 mysqli_free_result ($r);
90 mysqli_close($dbc);
91
92 // Make the links to other pages, if necessary.
93 if ($pages > 1) {
94
95 echo '
<p>';
96 $current_page = ($start/$display) + 1;
97
98 // If it's not the first page, make a Previous button:
99 if ($current_page != 1) {
100 echo '<a href="view_users. php?s=' . ($start - $display) . '&p=' . $pages . '&sort=' .
101 $sort . '">Previous ';
102 }
103 // Make all the numbered pages:
104 for ($i = 1; $i <= $pages; $i++) {
105 if ($i != $current_page) {
106 echo '<a href="view_users. php?s=' . (($display * ($i - 1))) . '&p=' . $pages . '&sort=' .
107 $sort . '">' . $i . ' ';
108 } else {
109 echo $i . ' ';
110 } // End of FOR loop.
111

```

```

112 // If it's not the last page, make a Next button:
113 if ($current_page != $pages) {
114 echo '<a href="view_users.php?s=' . ($start + $display) . '&p=' . $pages . '&sort=' . $sort .
115 '>Next';
116 }
117 echo '</p>'; // Close the paragraph.
118
119 } // End of links section.
120
121 include ('includes/footer.html');
122 ?>

```

\$sort变量将用于确定如何对查询结果进行排序。这一行代码使用三元运算符（参见本章前面的框注“三元运算符”）给\$sort赋值。如果设置了\$\_GET['sort']（在用户单击了任何链接之后将会如此），那么就应该把这个值赋予\$sort。如果没有设置\$\_GET['sort']，那么就会把默认值rd（registration date的简写）赋值\$sort。

### (3) 确定应该如何对结果排序。

```

switch ($sort) {
 case 'ln':
 $order_by = 'last_name ASC';
 break;
 case 'fn':
 $order_by = 'first_name ASC';
 break;
 case 'rd':
 $order_by = 'registration_date ASC';
 break;
 default:
 $order_by = 'registration_date ASC';
 $sort = 'rd';
 break;
}

```

10

这个switch条件语句相对于几个期望的值来检查\$sort。例如，如果它等于ln，那么应该以姓氏对结果进行升序排序。在SQL查询中将使用指定的\$order\_by变量。

如果\$sort的值为fn，那么应该以名字对结果进行升序排序。如果值为rd，那么将以注册日期对结果进行升序排序。这也是默认的情况。在此，默认情况可以防止恶意用户更改\$\_GET['sort']的值，从而可能中断查询。

### (4) 修改查询，使用新的\$order\_by变量。

```

$q = "SELECT last_name, first_name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr, user_id FROM users
 ORDER BY $order_by LIMIT $start, $display";

```

此时，\$order\_by变量具有一个值，指示应该如何对返回的结果排序（例如，registration\_date ASC），因此，可以轻松地把它添加到查询中。记住，ORDER BY子句出现在LIMIT子句之前。如果得到的查询没有正确运行，可把它打印出来并检查它的语法。

### (5) 修改表头的echo语句，从列标题外面创建链接。

```

echo '<table align="center" cellspacing="0" cellpadding="5" width="75%>
<tr>
 <td align="left">Edit</td>
 <td align="left">Delete</td>
 <td align="left">Last Name</td>
 <td align="left">First Name</td>
 <td align="left">Date Registered</td>
</tr>
';

```

为了使列标题成为可单击的链接，只需用[标签包围它们。每个链接的href属性的值对应于\\$\\_GET\['sort'\]\]的可接受的值（参见第\(3\)步中的switch语句）。](#)

(6) 修改创建Previous链接的echo语句，使得也可以传递排序值。

```

echo 'Previous';
';

```

我添加了另一个name=value对到Previous链接中，使得还会把排序顺序发送到结果的每个页面。如果不这样做，那么标页码将会失败，因为ORDER BY子句将会因页面而异。

(7) 为编过号的页面和Next链接重复第(6)步。

```

echo '' .
 $i . '';
echo 'Next';

```

(8) 将文件另存为view\_users.php，存放在Web目录中，并在Web浏览器中运行它（参见图10-17和图10-18）。

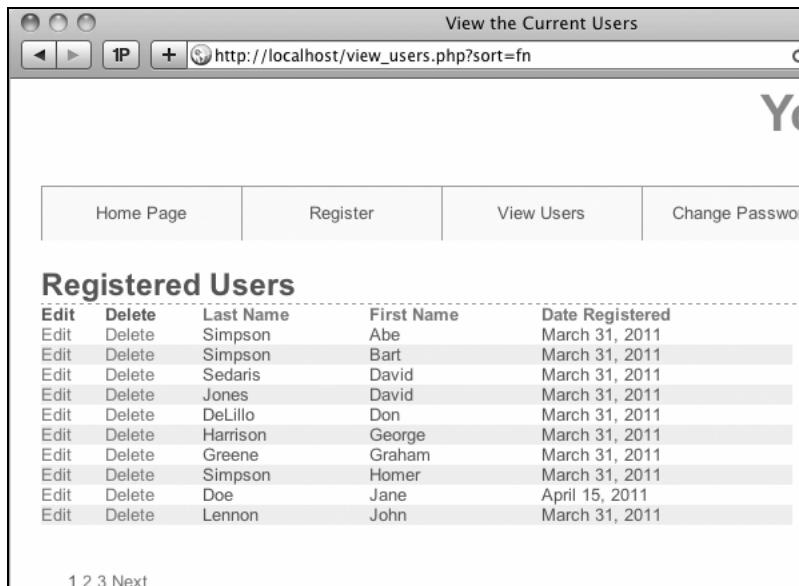


图10-17 在单击名字列之后，将按名字升序显示结果

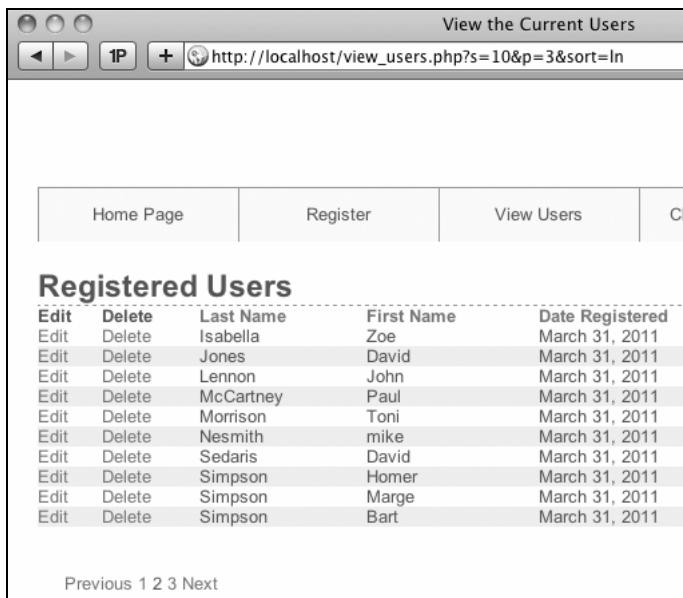


图10-18 单击Last Name列，然后单击第2个分页，页面显示了按姓氏升序显示的第2组结果

### ✓ 提示

- 这个示例中还演示了一个非常重要的安全概念。我没有直接在查询中使用\$\_GET['sort']的值，而是针对switch中假定的值对其进行检查。如果由于某种原因，\$\_GET['sort']具有不同于所期待的值，则查询将使用默认的排序顺序。其要点是：不要对接收到的数据做任何假设，并且不要在SQL查询中使用未经验证的数据。

## 10.6 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 10.6.1 回顾

- 调试PHP-MySQL的标准步骤顺序是什么（在第8章的结尾明确说明过）？
- 传递值到PHP脚本的两种方法是什么（除了用户输入）？
- delete\_user.php和edit\_user.php采取了什么样的安全措施来阻止恶意或意外的删除？
- 为什么不经过mysqli\_real\_escape\_string()处理而直接使用\$id值是安全的？
- 在什么情况下mysqli\_affected\_rows()函数会返回错误负数（例如，报告没有记录被影响，而事实上查询成功运行）？
- 什么是三目运算符？它如何使用？

- 要分布显示查询结果需要哪两个值?
- 该如何修改查询以让它的结果分页显示?
- 如果分页查询基于附加条件(除了用在LIMIT子句中的),而那些附加条件并没在分页链接中传递,会发生什么?
- 为什么不能直接在查询中使用\$\_GET['sort']?
- 为什么必须在每个分页链接中传递排序值?

### 10.6.2 实践

- 更改delete\_user.php和edit\_user.php页面,使它们都在浏览器窗口的标题栏显示受影响的用户。
- 修改edit\_user.php使你还可以修改用户的密码。
- 加分练习,修改edit\_user.php,如果设置了\$\_POST,就让表单元素显示\$\_POST中的值;如果没有设置,就显示数据库中的内容。
- 改变view\_users.php中\$display变量的值,更改分页数。
- 分页另外一个查询结果,例如银行数据库中的账户或客户列表。
- 为银行数据库创建删除和修改脚本。你必须考虑这里的外键约束的限制,例如,删除名下仍有账户的客户。



## 本章内容

- 发送电子邮件
- 处理文件上传
- PHP 和 JavaScript
- 理解 HTTP 头部
- 日期和时间函数
- 回顾和实践

**前** 两章重点介绍了结合使用 PHP 和 MySQL (毕竟这是本书的基本点)。但是,还要介绍许多以 PHP 为中心的内容。我们暂时打断一下结合使用 PHP 和 MySQL 的思路,本章介绍了在更复杂的 Web 应用程序中通常使用的许多技术。

本章介绍的第一个主题是使用 PHP 发送电子邮件。我们经常做这样的事情,并且它非常简单(假定正确设置了服务器)。接下来会讨论三个新主题:如何处理 HTML 表单中的文件上传;结合使用 PHP 和 JavaScript,以及如何使用 `header()` 函数操纵 Web 浏览器;最后讨论 PHP 中提供的一些日期和时间函数。

11

## 11.1 发送电子邮件

关于 PHP 我最喜爱的事情之一是:它可以很轻松地发送电子邮件。在正确配置的服务器上,这个过程就像使用 `mail()` 函数那样简单:

```
mail (to, subject, body, [headers]);
```

`to` 值应该是电子邮件地址或一系列地址,用逗号隔开它们。可以使用下面的任何地址:

- `email@example.com`
- `email1@example.com`
- `email2@example.com`
- Actual Name <`email@example.com`>
- Actual Name <`email@example.com`>
- This Name <`email2@example.com`>

*subject*值将创建电子邮件的主题行，*body*用于放置电子邮件的内容。为了使事情更清晰，通常给变量赋值，然后在mail()函数调用中使用它们：

```
$to = 'email@example.com';
$subject = 'This is the subject';
$body = 'This is the body.
It goes over multiple lines.';
mail ($to, $subject, $body);
```

在对\$body变量的赋值中可以看到，你可以创建分布在多行上的电子邮件消息，只需使文本完全位于引号内即可。也可以在双引号内使用换行符(\n)来执行该操作：

```
$body = "This is the body.\nIt goes over multiple lines.";
```

这全都非常直观，只需要注意两点。第一，主题行不能包含换行符(\n)。第二，正文中每一行的长度都不能超过70个字符。可以使用wordwrap()函数完成这个任务。该函数每隔X个字符插入一个换行符。为了把文本限制为70个字符，可使用：

```
$body = wordwrap($body, 70);
```

mail()函数还可以带有第四个可选参数，用于其他头部。你可以在这里设置From(发件人)、Reply-To(回复)、Cc(抄送)、Bcc(密件抄送)和类似的选项。例如：

```
mail ($to, $subject, $body, 'From: reader@example.com');
```

为了在电子邮件中使用多种不同类型的头部，可以用\r\n隔开每个头部：

```
$headers = "From: John@example.com\r\n";
$headers .= "Cc: Jane@example.com, Joe@example.com\r\n";
mail ($to, $subject, $body, $headers);
```

尽管第四个参数是可选的，也建议你总是包括From值(尽管也可以在PHP的配置文件中建立它)。

为了演示这一点，我们创建一个显示联系人表单的页面(参见图11-1)，然后处理表单提交，验证数据并在电子邮件中一起发送它。这个示例还将包括在本书中使用的黏性表单的一个良好的变体。

Contact Me

Please fill out this form to contact me.

Name: Not Larry Ullman

Email Address: email@example.com

Comments:

Your book isn't just the  
greatest computer book ever  
written, it's the greatest  
book ever written, period! It  
totally blows "One Hundred

Send!

图11-1 一个标准(但不是非常吸引人)的联系人表单

在运行这个脚本之前需注意两件事情：第一，为了让这个示例工作，运行PHP的计算机必须具有

一个工作的邮件服务器。如果使用托管站点，这不应该是一个问题；在你自己的计算机上，很可能需要采取一些预备性措施（参见框注“PHP mail()相关性”）。

第二，这个示例虽然可以工作，但它也可以被坏人操纵，允许他们通过你的联系人表单发送垃圾邮件（不仅仅是发送给你，而是发送给任何人）。第13章中介绍了防止这类攻击的措施。遵循并测试这个示例就很好，但是依靠它并把它作为你的长期联系人表单解决方案则是一个糟糕的主意。

### PHP mail()相关性

PHP的mail()函数实际上不会发送电子邮件本身，而是告诉在计算机上运行的邮件服务器执行该任务。这意味着运行PHP的计算机上必须具有一个工作的邮件服务器，以便让该函数工作。

如果你有一台运行UNIX变体的计算机，或者如果你通过专业主机运行Web站点，这应该不是一个问题。但是，如果在你自己的台式机或笔记本上运行PHP，就可能需要进行调整。

如果你运行Windows，并且有Internet服务提供商（ISP, Internet Service Provider）给你提供SMTP服务器（如smtp.comcast.net），那么可以在php.ini文件中设置这种信息（参见附录A，了解如何编辑这个文件）。但是仅当你的ISP不需要身份验证（用户名和密码的组合）即可使用SMTP服务器时，这才会工作。否则，你将需要在计算机上安装SMTP服务器。有大量可用的SMTP服务器，并且可以比较轻松地安装和使用它们：只需在Internet上搜索free windows smtp server，就会看到一些选项。在本书对应的论坛中（www.DMCInsights.com/phorum/）也提供了关于这个主题的线索。

如果你运行的是Mac OS X，将需要启用内置的SMTP服务器（可能是sendmail或postfix，这取决于你运行的Mac OS X的特定版本）。你可以找到关于执行该操作的在线指导（搜索enable sendmail，“Mac OS X”）。

11

### 发送邮件

(1) 在文本编辑器或IDE中开始创建一个新的PHP脚本，命名为email.php（参见脚本11-1）。

**脚本 11-1** 这个页面显示一个联系人表单，在提交时，它将连同表单数据一起把电子邮件发送到电子邮件地址

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Contact Me</title>
6 </head>
7 <body>
8 <h1>Contact Me</h1>
9 <?php # Script 11.1 - email.php
10
11 // Check for form submission:
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13
14 // Minimal form validation:
15 if (!empty($_POST['name']) && !empty($_POST['email']) && !empty($_POST ['comments'])) {

```

```

16 // Create the body:
17 $body = "Name: {$_POST['name']}\n\nComments: {$_POST['comments']}";
18
19 // Make it no longer than 70 characters long:
20 $body = wordwrap($body, 70);
21
22 // Send the email:
23 mail('your_email@example.com', 'Contact Form Submission', $body, "From:{$_POST['email']}");
24
25 // Print a message:
26 echo '<p>Thank you for contacting me. I will reply some day.</p>';
27
28 // Clear $_POST (so that the form's not sticky):
29 $_POST = array();
30
31 } else {
32 echo '<p style="font-weight: bold; color: #C00">Please fill out the form completely.</p>';
33 }
34
35 } // End of main isset() IF.
36
37 // Create the HTML form:
38 ?>
39 <p>Please fill out this form to contact me.</p>
40 <form action="email.php" method="post">
41 <p>Name: <input type="text" name="name" size="30" maxlength="60" value="<?php if (isset($_POST
42 ['name'])) echo $_POST['name']; ?>" /></p>
43 <p>Email Address: <input type="text" name="email" size="30" maxlength="80" value="<?php if
44 (isset($_POST['email'])) echo $_POST['email']; ?>" /></p>
45 <p>Comments: <textarea name="comments" rows="5" cols="30"><?php if (isset($_POST['comments']))
46 echo $_POST['comments']; ?></textarea></p>
47 <p><input type="submit" name="submit" value="Send!" /></p>
48 </form>
49 </body>
50 </html>

```

本章中的所有示例都不会使用模板，就像前两章中的那些示例那样，因此它开始于标准的HTML。

(2) 创建条件语句，用于检查是否已经提交了表单并验证表单数据。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 if (!empty($_POST['name']) && !empty($_POST['email']) && !empty($_POST['comments'])) {

```

表单包含三个文本输入框（严格说来，其中一个是文本区）。empty()函数将确认每个文本框中都输入了一些内容。在第13章中，你将学习如何使用正则表达式来确认提供的电子邮件地址具有有效的格式。

(3) 创建电子邮件的正文。

```

$body = "Name: {$_POST['name']}\n\nComments: {$_POST['comments']}";
$body = wordwrap($body, 70);

```

电子邮件的正文将开始于提示性文字Name:，其后接着在表单中输入的名字。然后对评论做相同的处理。wordwrap()函数然后格式化所有的正文，使得每一行的长度只有70个字符。

(4) 发送电子邮件并在Web浏览器中打印消息。

```
mail('your_email@example.com', 'Contact Form Submission', $body, "From: {$_POST['email']}");
echo '<p>Thank you for contacting me. I will reply some day.</p>';
```

假定正确配置了服务器，这一行代码将发送电子邮件。你将需要把 *to* 值更改为实际的电子邮件地址。*From* 值将是来自于表单的电子邮件地址。主题将是字面字符串。

无法确认是否成功发送了电子邮件，更不用说接收了，但是会打印一条通用消息。

(5) 清除\$\_POST数组。

```
$_POST = array();
```

在这个示例中，总会显示表单，即使成功地提交了它也会如此。表单将是黏性的，以防用户遗漏了某些内容（参见图11-2）。不过，如果发送邮件，就无需再次显示表单中的值。为了避免这一点，可以使用array()函数清除\$\_POST数组的值。

The screenshot shows a web form titled 'Contact Me'. It contains two text fields: 'Name:' with the value 'Not Larry Ullman' and 'Email Address:' which is empty. Below these is a large text area labeled 'Comments:' containing the text: 'Your book isn't just the greatest computer book ever written, it's the greatest book ever written, period! It totally blows "One Hundred'. A scroll bar is visible on the right side of the text area. At the bottom is a 'Send!' button.

图11-2 联系人表单将记住用户提交的值，以防没有完全填写它

(6) 完成条件语句。

```
} else {
 echo '<p style="font-weight: bold; color: #C00">Please fill out the form completely.</p>';
}
} // End of main isset() IF.
```

出错消息中包含一些内联CSS，以便将其显示为红色、加粗格式。

(7) 开始创建表单。

```
<p>Please fill out this form to contact me.</p>
<form action="email.php" method="post">
 <p>Name: <input type="text" name="name" size="30" maxlength="60" value="<?php if (isset($_POST
 ['name'])) echo $_POST['name']; ?>" /></p>
 <p>Email Address: <input type="text" name="email" size="30" maxlength="80" value="<?php if (isset
 ($_POST['email'])) echo $_POST['email']; ?>" /></p>
```

表单将使用POST方法提交回这个相同的页面。前两个输入框是文本类型，通过检查对应的\$\_POST

变量是否具有值把它们制作成黏性的。如果是，就把该值打印成对应输入框的当前值。

(8) 完成表单。

```
<p>Comments: <textarea name= "comments" rows="5" cols="30"> <?php if (isset($_POST ['comments'])) echo
$_POST ['comments']; ?></textarea></p>
<p><input type="submit" name= "submit" value="Send!" /></p>
</form>
```

评论输入框是一个文本区，它没有使用value属性。但是为使之具有黏性，在textarea开标签、闭标签之间打印值。

(9) 完成HTML页面。

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

(10) 将文件另存为email.php，存放在Web目录中，并在Web浏览器中测试它（参见图11-3）。

**Contact Me**

*Thank you for contacting me. I will reply some day.*

Please fill out this form to contact me.

Name:

Email Address:

Comments:

图11-3 成功完成和提交表单

(11) 检查电子邮件，确认你接收到了消息（参见图11-4）。

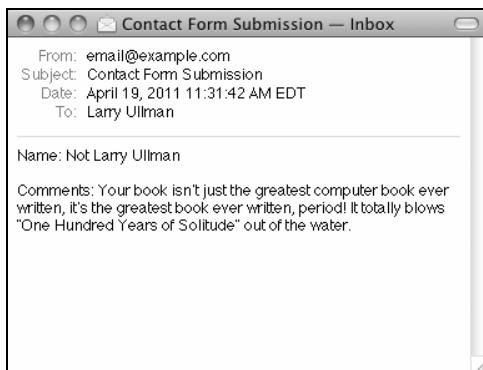


图11-4 得到的电子邮件（图11-1中的数据）

如果实际上没有得到邮件，将需要做一些调试工作。对于这个示例，应该与你的主机（如果使用托管站点）或你自己确认（如果在你的服务器上运行PHP）是否安装了一个工作的邮件服务器。还应该使用不同的电子邮件地址测试它（为 `to` 和 `from` 值）。另外，还要观察你的垃圾邮件过滤器有没有阻塞消息。

#### ✓ 提示

- 在一些（主要是UNIX）系统上，将不会正确处理\r\n字符。如果你有关于它们的问题，可代之以只使用\n。
- `mail()` 函数返回1或0，指示函数调用是否成功。这与电子邮件成功发送或接收不是一回事。你不能使用PHP容易地测试它。
- 虽然很容易利用`mail()` 函数发送简单的消息，但是发送HTML电子邮件或带有附件的电子邮件涉及更多的工作。我在*PHP 5 Advanced: Visual QuickPro Guide*一书中讨论了如何处理这种电子邮件。
- 使用让PHP发送电子邮件的联系人表单是把接收到的垃圾邮件数量减至最少的极佳方式。利用这种系统，在Web浏览器中将看不到实际的电子邮件地址，这说明“垃圾波”病毒（spambot）不能利用它。

## 11.2 处理文件上传

第2章和第3章介绍了利用PHP处理HTML表单的基础知识。一般来讲，可以利用PHP以相同的方式处理每种表单元素，只有一个例外：文件上传。上传一个文件的过程涉及两个方面。首先，必须显示HTML表单，并且编写正确的代码以允许文件上传。然后，在提交表单时，PHP脚本必须把上传的文件复制到其最终目的地。

11

不过，为了让这个过程工作，必须做好以下几件事情：

- 必须正确地设置PHP；
- 必须有一个临时存储目录，它具有正确的权限；
- 必须有一个最终存储目录，它具有正确的权限。

在记住这些后，下一节将介绍允许文件上传的服务器设置；然后，将创建实际执行上传过程的PHP脚本。

### 11.2.1 允许文件上传

如我曾经说过的，必须先建立某些设置，以便PHP能够处理文件上传。在向你介绍这些步骤之前，我将先讨论为什么以及何时需要执行这些调整。

第一个问题在于PHP本身。PHP的配置文件（`php.ini`）中有多种设置，它们规定了PHP如何处理上传，特别指出了可以上传多大的文件，以及应该把上传的文件临时存储在什么位置（参见表11-1）。一般来讲，如果满足下面的任何条件，就需要编辑这个文件：

- 禁用了`file_uploads`；
- PHP没有临时目录可以使用；

你将上传非常大的文件（超过2 MB）。

表11-1 所有这些PHP配置设置都会影响文件上传能力

设 置	值 类 型	重 要 性
file_uploads	布尔型	使PHP支持文件上传
max_input_time	整型	指示允许PHP脚本运行多长时间（以秒为单位）
post_max_size	整型	允许的POST数据的总大小（以字节为单位）
upload_max_filesize	整型	允许的尽可能最大的文件上传（以字节为单位）
upload_tmp_dir	字符串型	指示应该临时把上传的文件存储在什么位置

如果你没有访问php.ini文件的权限——比如，如果你使用托管站点，可能主机已经把PHP配置成允许文件上传。

第二个问题是临时目录的位置以及它所具有的权限。在PHP脚本把上传的文件移到其最终目的地之前，都将把它存储在临时目录中。如果在自己的Windows计算机上安装PHP，可能需要在这里采取一些措施（在我自己的Windows XP上安装默认的PHP没有出现问题，但是我不想假定对每个人都会如此）。Mac OS X和UNIX用户不需要担心这一点，因为已经存在用于此目的的临时目录（/tmp）。

最后，必须创建目标文件夹，并在其上建立适当的权限。必须为处理文件上传的每个应用程序执行这个步骤。由于这一步中涉及重要的安全性问题，还要确保你阅读并理解了框注“安全的文件夹权限”。

在记住所有这些后，让我们执行下面这些步骤。

#### 安全的文件夹权限

在安全性与方便性之间通常有一个折中。对于这个示例，把uploads文件夹放在Web文档目录内更方便（这种方便性来自于可以在Web浏览器中多么容易地查看上传的图像），但是这样做将不太安全。

为了让PHP能够把文件存放在uploads文件夹中，它需要具有该目录上的写权限。在大多数服务器上，PHP作为与Web服务器本身相同的用户运行。在托管服务器上，这意味着被托管的全部X个站点都作为相同的用户运行。创建PHP可以写入内容的文件夹意味着创建每个人都可以写入内容的文件夹。确切地讲，服务器上的任何人现在都可以移动、复制或者把文件写到uploads文件夹中（假定他们知道它存在）。这甚至意味着恶意用户可以把PHP脚本写到你的uploads目录中。不过，由于这个示例中的uploads目录不在Web目录内，这样的PHP脚本将不能在Web浏览器中运行。这样，就显得不太方便，但是它更安全。

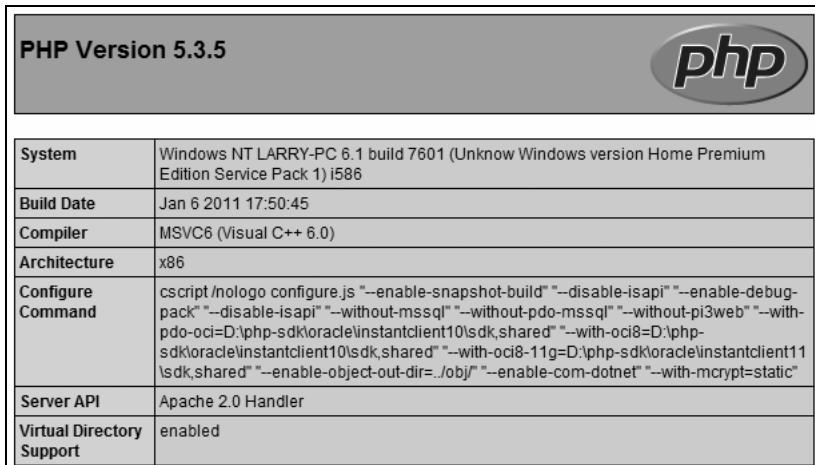
如果必须保持uploads文件夹可公开访问，可以调整权限。出于安全目的，理想情况下希望只允许Web服务器用户读、写和浏览这个目录。这意味着知道Web服务器作为什么用户运行，并且使该用户（而不包括其他任何用户）成为uploads文件夹的管理者。这不是一个完美的解决方案，但它确实有一点帮助。不过，这种改变也会限制你对该文件夹的访问，因为其内容将只属于Web服务器。

最后，如果使用Apache，则可以使用`.htaccess`文件限制对`uploads`文件夹的访问。实质上，你将指出文件夹中只有图像文件才是可公开查看的，这意味着即使把PHP脚本存放在那里，也不能执行它。可以在附录A找到关于如何使用`.htaccess`文件的信息（peachpit.com可免费下载）。

有时，甚至最保守的程序员也会在安全性方面做出让步。关键是你知道潜在的顾虑，并且你会竭尽全力把危险降至最低。

### 准备服务器

(1) 运行`phpinfo()`函数，确认你的服务器设置（参见图11-5）。



The screenshot shows the output of the `phpinfo()` function. At the top, it says "PHP Version 5.3.5" and features the PHP logo. Below is a table with the following data:

System	Windows NT LARRY-PC 6.1 build 7601 (Unknown Windows version Home Premium Edition Service Pack 1) i586
Build Date	Jan 6 2011 17:50:45
Compiler	MSVC6 (Visual C++ 6.0)
Architecture	x86
Configure Command	cscript/nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--disable-isapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk\shared" "--with-oci8=D:\php-sdk\oracle\instantclient11\sdk\shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled

图11-5 `phpinfo()`脚本返回关于你的PHP设置的所有信息，包括所有文件上传处理内容

11

`phpinfo()`函数会打印出关于你的PHP设置的大量信息。该函数即使不是PHP中最重要的函数，也是最重要的函数之一（在我看来是这样）。查看表11-1中列出的设置，并确认它们的值。确保`file_uploads`的值为On，并且`upload_max_filesize`的限制条件（默认为2 MB）和`post_max_size`的限制条件（8 MB）不会给你带来限制。如果在Windows上运行PHP，就要查看`upload_tmp_dir`是否具有一个值。如果它没有值，就可能是一个问题（在运行处理文件上传的PHP脚本后，你肯定会知道情况）。对于非Windows用户，如果这个值指示no value，就非常不错。

(2) 如果需要，在文本编辑器中打开`php.ini`。

如果在第(1)步中看到任何需要更改的内容，或者如果你在使用PHP实际处理文件上传时发生某件事情，将需要编辑`php.ini`文件。要找到这个文件，可查看`phpinfo()`输出中的`Configuration File (php.ini) path`值。这准确指示这个文件位于计算机上的什么位置（另请参见附录A，以了解更多信息）。

如果不允许你编辑`php.ini`文件（例如，如果你使用托管服务器），那么大概已经执行了任何必要的编辑以允许进行文件上传。如果不是这样，将需要请求托管公司执行这些更改（它们可能会同意这样做）。

(3) 搜索`php.ini`文件中要更改的配置，并执行任何必要的编辑（参见图11-6）。

```

864 ;;;;;;;;;;;
865 ; File Uploads ;
866 ;;;;;;;;;;;
867 ;
868 ; Whether to allow HTTP file uploads.
869 ; http://php.net/file-uploads
870 file_uploads = On
871
872 ; Temporary directory for HTTP uploaded files (will use system default if not
873 ; specified).
874 ; http://php.net/upload-tmp-dir
875 ;upload_tmp_dir =
876
877 ; Maximum allowed size for uploaded files.
878 ; http://php.net/upload-max-filesize
879 ;upload_max_filesize = 2M
880
881 ; Maximum number of files that can be uploaded via a single request
882 max_file_uploads = 20

```

图11-6 php.ini文件中的File Uploads一小部分

例如，在File Uploads这个部分中，你将看到下面三行代码：

```

file_uploads = On
;upload_tmp_dir =
;upload_max_filesize = 2M

```

第一行指示是否允许上传。第二行指出应该把上传的文件临时存储在什么位置。在大多数操作系统（包括Mac OS X和UNIX）上，可以注释掉这个设置（在其前面放置一个分号），而不会有任何问题。

如果运行Windows并且需要创建一个临时目录，可把该值设置为C:\tmp，并确保这一行前没有分号。同样，在Windows 7上使用XAMPP时，我不需要创建临时目录，因此你可能也不需要创建它。

最后，设置上传文件的最大大小〔在配置设置中，M是兆字节（MB）的简写〕。

(4) 保存php.ini文件，并重新启动Web服务器。

重新启动Web服务器的方式依赖于使用的操作系统和Web服务应用程序。参见附录A，了解相关指导。

(5) 重新运行phpinfo()脚本，确认所做的更改。

在执行下一步的操作之前，重复第(1)步的操作，确认执行了所有必要的更改。

(6) 如果运行Windows并且需要创建一个临时目录，可以在C:\内添加一个tmp文件夹，并确保每一个人都可以向该目录写内容（参见图11-7）。

PHP将通过Web服务器把上传的文件临时存储在upload\_tmp\_dir中。为了使之工作，Web用户（如果Web服务器作为特定的用户运行）必须有权写到该文件夹。

在各种可能的情况下，可能不必实际地更改权限，但是如果要这样做，通常可以根据运行的Windows版本，通过右击文件夹并选择“属性”来调整权限。在“属性”窗口中，应该有一个“安全”选项卡，可以在其中设置权限。另外，它也可能出现在“共享”选项卡下。Windows使用更宽松的权限系统，因此可能不必更改任何设置，除非有意对文件夹设置限制。〔注意：我没有在Windows Vista上测试它，因此我不确信其中可能发生了什么变化（如果有的话）。〕

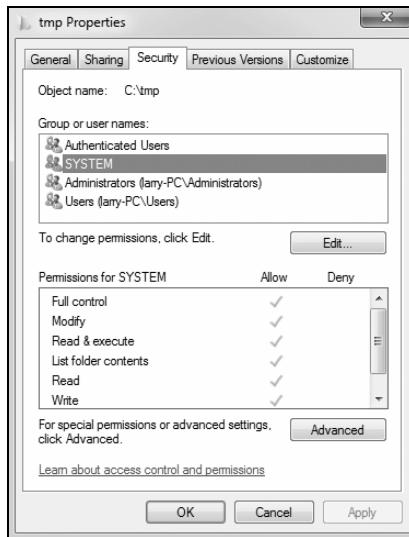


图11-7 Windows用户需要确保PHP可以写到C:\tmp（或者使用的任何目录）

Mac OS X和UNIX用户可以跳过这一步，因为临时目录（/tmp）已经具有开放的权限。

#### (7) 在Web根目录外面的某个目录中创建一个名为uploads的新目录。

所有上传的文件都将永久存储在uploads目录中。如果把PHP脚本存放在C:\xampp\htdocs\ch11目录中，那么就创建一个C:\xampp\uploads目录。或者如果把文件存放在/Users/~<username>/Sites/ch11中，就创建一个/Users/~<username>/uploads文件夹。图11-8显示了应该建立的结构，并且框注“安全的文件夹权限”讨论了为什么这一步是必要的。

11

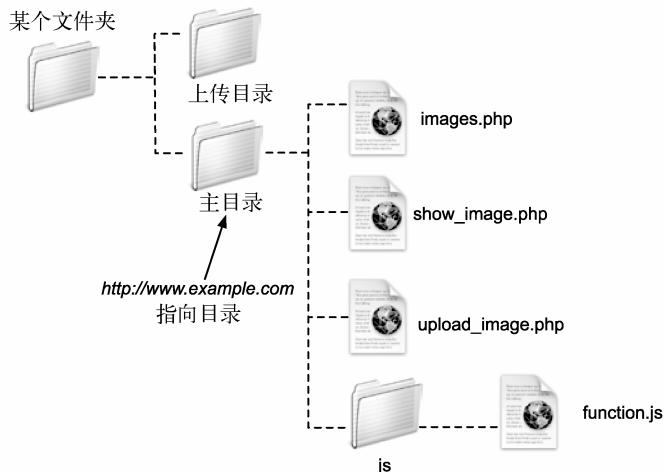


图11-8 假定htdocs是Web根目录（www.example.com或http://localhost指向那里），那么需要把uploads目录放在它外面

(8) 在uploads目录上设置权限，使得Web服务器可以向其中写入内容。

同样，Windows用户可以使用“属性”窗口执行这些更改，尽管可能不需要这样做。Mac OS X用户可以：

- (1) 在Finder中选择文件夹；
- (2) 按下Command+I组合键；
- (3) 允许任何人在Ownership & Permissions面板下执行Read & Write（参见图11-9）。

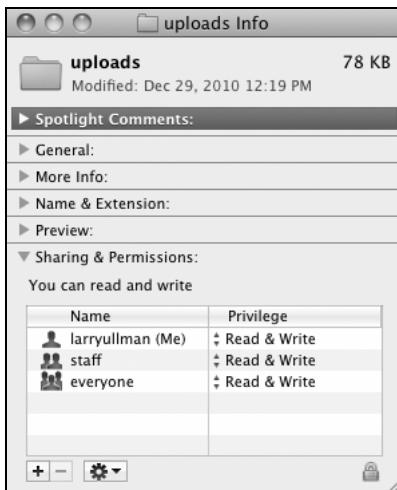


图11-9 在Mac OS X中调整uploads文件的属性

如果使用托管站点，主机很可能提供一个控制面板，可以通过它调整文件夹的设置，或者可能在FTP应用程序内执行该任务。

依赖于你的操作系统，你也许能够在没有先执行这一步的情况下上传文件。你可以在改变权限前试试以下脚本，看看即可。如果看到如图11-10所示的消息，那么将需要做出某些调整。

```
Warning: move_uploaded_file(..../uploads/trout.jpg) [function.move-uploaded-file]
failed to open stream: Permission denied in
/Users/larryullman/Sites/phpmysql4/upload_image.php on line 27

Warning: move_uploaded_file() [function.move-uploaded-file]: Unable to move
'/Applications/MAMP/tmp/php/php0EKXMX' to '..uploads/trout.jpg' in
/Users/larryullman/Sites/phpmysql4/upload_image.php on line 27
```

图11-10 如果由于权限问题，PHP不能把上传的图像移到uploads文件夹上，你将看到如本图所示的一条出错消息。修改uploads上的权限即可校正这个问题

### ✓ 提示

- UNIX用户可以使用chmod命令调整文件夹的权限。就UNIX而言，正确的权限将是755或777。
- 由于上传大文件可能需要花费一些时间，还可能需要在php.ini中更改max\_input\_time值，或者在脚本中使用set\_time\_limit()函数绕过它。

- 文件和目录权限可能是一个复杂的主题，尤其是当你以前从未接触过它们时。如果你对这些步骤或者下一个脚本有任何问题，可以搜索网络或者向本书对应的论坛求助。

### 11.2.2 利用PHP上传文件

既然服务器已经被正确设置成允许文件上传（希望如此），就可以创建执行实际文件处理的PHP脚本。这样一个脚本具有两个部分：HTML表单和PHP代码。

使表单处理文件上传所需的语法具有三个部分：

```
<form enctype="multipart/form-data" action="script.php" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="30000" />
File <input type="file" name="upload" />
```

初始表单标签的enctype部分指示表单应该能够处理多种类型的数据，包括文件。如果你想接受文件上传，就必须包括这个enctype！另请注意：提交表单必须使用POST方法。`MAX_FILE_SIZE`隐藏输入框是一种表单限制元素，用于限制所选的文件可以有多大（以字节为单位），它必须出现在文件输入框之前。虽然用户可以很容易地避开这种限制，但仍然应该使用它。最后，`file`输入框类型将在表单中创建合适的按钮（参见图11-11和图11-12）。



图11-11 在Windows上的IE 9中显示的文件输入框



11

图11-12 在Mac OS X上的Google Chrome中显示的文件输入框

在提交表单时，可以使用`$_FILES`超全局变量访问上传的文件。该变量将是表11-2中列出的值的数组。

表11-2 可以通过这些数组元素使用上传文件的数据

索引	含义
<code>name</code>	文件的原始名称（当它位于用户的计算机上时）
<code>type</code>	文件的MIME类型，由浏览器提供
<code>size</code>	上传文件的大小（以字节为单位）
<code>tmp_name</code>	上传文件的临时文件名（当它存储在服务器上时）
<code>error</code>	与任何问题关联的错误代码

一旦PHP脚本接收到文件，`move_uploaded_file()`函数就可以把它从临时目录中转移到其永久位置。

```
move_uploaded_file (temporary_filename,
/path/to/destination/filename);
```

下一个脚本将让用户在他们的计算机上选择一个文件，然后把它存储在uploads目录中。该脚本将检查文件是否是图像类型（JPEG或PNG）。在本章的下一节中，另一个脚本将列出上传的图像，并创建指向它们的链接。

用PHP处理文件上传

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为upload\_image.php（参见脚本11-2）。

**脚本 11-2** 这个脚本允许用户把图像文件从他们的计算机上传到服务器上

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml11/
2 DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <title>Upload an Image</title>
7 <style type="text/css" title="text/css" media="all">
8 .error {
9 font-weight: bold;
10 color: #C00;
11 }
12 </style>
13 </head>
14 <body>
15 <?php # Script 11.2 - upload_image.php
16 // Check if the form has been submitted:
17 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
18
19 // Check for an uploaded file:
20 if (isset($_FILES['upload'])) {
21
22 // Validate the type. Should be JPEG or PNG.
23 $allowed = array ('image/pjpeg', 'image/jpeg', 'image/JPG', 'image/X-PNG', 'image/PNG',
24 'image/png', 'image/x-png');
25 if (in_array($_FILES['upload']['type'], $allowed)) {
26
27 // Move the file over.
28 if (move_uploaded_file ($_FILES['upload']['tmp_name'], "../uploads/{$_FILES['upload']
29 ['name']}")) {
30 echo '<p>The file has been uploaded!</p>';
31 } // End of move... IF.
32
33 } else { // Invalid type.
34 echo '<p class="error">Please upload a JPEG or PNG image.</p>';
35 }
36
37 } // End of isset($_FILES['upload']) IF.
38
39 // Check for an error:
40 if ($_FILES['upload']['error'] > 0) {
41 echo '<p class="error">The file could not be uploaded because: ';
```

```
41 // Print a message based upon the error.
42 switch ($_FILES['upload']['error']) {
43 case 1:
44 print 'The file exceeds the upload_max_filesize setting in php.ini.';
45 break;
46 case 2:
47 print 'The file exceeds the MAX_FILE_SIZE setting in the HTML form.';
48 break;
49 case 3:
50 print 'The file was only partially uploaded.';
51 break;
52 case 4:
53 print 'No file was uploaded.';
54 break;
55 case 6:
56 print 'No temporary folder was available.';
57 break;
58 case 7:
59 print 'Unable to write to the disk.';
60 break;
61 case 8:
62 print 'File upload stopped.';
63 break;
64 default:
65 print 'A system error occurred.';
66 break;
67 } // End of switch.
68
69 print '</p>';
70
71 } // End of error IF.
72
73 // Delete the file if it still exists:
74 if (file_exists($_FILES['upload']['tmp_name']) && is_file($_FILES['upload']['tmp_name'])) {
75 unlink($_FILES['upload']['tmp_name']);
76 }
77
78 } // End of the submitted conditional.
79 ?>
80
81 <form enctype="multipart/form-data" action="upload_image.php" method="post">
82
83 <input type="hidden" name="MAX_FILE_SIZE" value="524288" />
84
85 <fieldset><legend>Select a JPEG or PNG image of 512KB or smaller to be uploaded:</legend>
86
87 <p>File: <input type="file" name="upload" /></p>
88
89 </fieldset>
90 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
91
92 </form>
93 </body>
94 </html>
```

这个脚本将利用一个CSS类来格式化所有错误。

(2) 检查是否提交了表单并且选择了一个文件。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 if (isset($_FILES['upload'])) {
```

由于这个表单没有其他要验证的字段（参见图11-13），所以只需要这个条件语句。你也可以验证上传文件的大小，以确定它是否在可接受的范围内（参考\$\_FILES['upload']['size']值）。

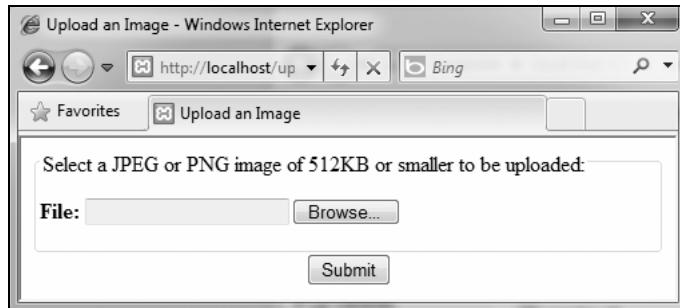


图11-13 这个非常基本的HTML表单只带有一个文件输入框

(3) 检查上传的文件是否具有正确的类型。

```
$allowed = array ('image/pjpeg', 'image/jpeg', 'image/JPG', 'image/X-PNG', 'image/PNG', 'image/png',
 'image/x-png');
if (in_array($_FILES['upload']['type'], $allowed)) {
```

文件的类型是其MIME类型，指示它是哪种类型的文件。浏览器可以确定并且可能提供这种信息，这依赖于所选文件的属性。

为了验证文件的类型，首先创建允许选项的数组。允许类型的列表基于接受的JPEG和PNG。一些浏览器具有MIME类型的变体，因此这里也包括了它们。如果上传文件的类型在这个数组中，文件就是有效的，并且应该对其进行处理。

(4) 把文件复制到它在服务器上的新位置。

```
if (move_uploaded_file($_FILES['upload']['tmp_name'], ".../uploads/$_FILES['upload']['name']"))
{
 echo '<p>The file has been uploaded!</p>';
} // End of move... IF.
```

`move_uploaded_file()`函数将把文件从其临时位置移到其永久位置（在uploads文件夹中），此时文件将保留它的原始名称。在第18章中，你将看到如何给文件提供一个新名称。一般情况下，建议给文件提供一个新名称。

一条规则是，应该总是使用条件语句来确认成功地移动了文件，而不仅仅是假定会执行移动。

(5) 完成图像类型和`isset($_FILES['upload'])`条件语句。

```
} else { // Invalid type.
 echo '<p class="error">Please upload a JPEG or PNG image.</p>';
}
} // End of isset($_FILES['upload']) IF.
```

第一个else子句完成第(3)步中开始的if语句。如果上传了文件但它没有适当的MIME类型，就会应用这个else子句（参见图11-14）。

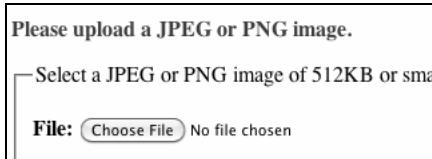


图11-14 如果用户上传了一个不是JPEG或PNG类型的文件，就会得到这个结果

(6) 检查并报告任何错误。

```
if ($_FILES['upload']['error'] > 0) {
 echo '<p class="error">The file could not be uploaded because: ';
```

如果发生错误，那么\$\_FILES['upload']['error']将具有一个大于0的值。在这种情况下，这个脚本将报告错误是什么。

(7) 创建一个switch语句，打印更详细的错误。

```
switch ($_FILES['upload']['error']) {
 case 1:
 print 'The file exceeds the upload_max_filesize setting in php.ini.';
 break;
 case 2:
 print 'The file exceeds the MAX_FILE_SIZE setting in the HTML form.';
 break;
 case 3:
 print 'The file was only partially uploaded.';
 break;
 case 4:
 print 'No file was uploaded.';
 break;
```

不能上传和移动文件的原因可能有如下几种：第一种（也是最明显的）原因是在目标文件夹上没有正确设置权限。在这种情况下，你将看到一条合适的出错消息（参见图11-10）。PHP通常还会在\$\_FILES['upload']['error']变量中存储一个错误编号。该编号对应于特定的问题，包括0~4以及6~8（非常奇怪的是，没有5）。这里的switch条件语句将依据错误编号打印出问题，并且添加了默认的情况，以支持将来的扩展（如果在PHP的以后版本中添加了不同的编号）。

一般来讲，这些错误对开发人员是有用的，但是不要向普通用户展示它们。

(8) 完成switch语句。

```
case 6:
 print 'No temporary folder was available.';
 break;
case 7:
 print 'Unable to write to the disk.';
 break;
case 8:
 print 'File upload stopped.';
 break;
```

```

default:
 print 'A system error occurred.';
 break;
} // End of switch.

```

(9) 完成用于处理错误的if条件语句。

```

print '</p>';
} // End of error IF.

```

(10) 删除临时文件(如果它存在的话)。

```

if (file_exists($_FILES['upload']['tmp_name']) && is_file($_FILES['upload']['tmp_name'])) {
 unlink($_FILES['upload']['tmp_name']);
}

```

如果上传了文件，但是不能把它移到其最终目的地，或者发生了其他一些错误，那么该文件仍将位于其在服务器上的临时位置。为了删除它，可使用unlink()函数。仅仅为了安全起见，在应用unlink()前，条件语句将检查文件是否存在并且它是否是一个文件。(这是由于当指定的项目是一个目录时，file\_exists()函数将返回TRUE。)

(11) 完成PHP部分。

```

} // End of the submitted conditional.
?>

```

(12) 创建HTML表单。

```

<form enctype="multipart/form-data" action="upload_image.php" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="524288" />
<fieldset><legend>Select a JPEG or PNG image of 512KB or smaller to be uploaded: </legend>
<p>File: <input type="file" name="upload" /></p>
</fieldset>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>

```

这个表单非常简单(参见图11-3)，但是它包含3个必要的部分用于文件上传：表单的enctype属性、MAX\_FILE\_SIZE隐藏输入框和file输入框。

(13) 完成HTML页面。

```

</body>
</html>

```

(14) 将文件另存为upload\_image.php，存放在Web目录中，并在Web浏览器中测试它(参见图11-15和图11-16)。

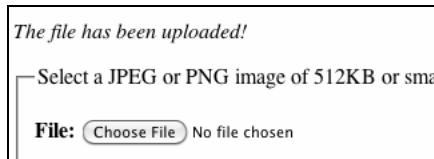


图11-15 在成功上传和移动文件之后得到的结果

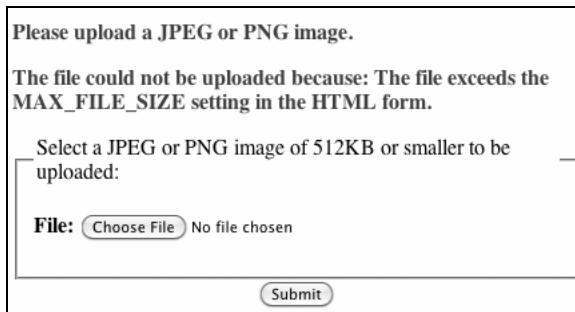


图11-16 在试图上传过大的文件时得到的结果

如果你愿意，可以通过检查uploads目录的内容确认脚本工作正常。

#### ✓ 提示

- 文件上传神秘失败的一个常见原因是遗漏了`enctype`表单属性。
- 也可以利用`is_uploaded_file()`函数验证上传的文件是否存在。
- Windows用户必须使用正斜杠或两个反斜杠引用目录（因此可以使用C:\\或C:/，而不能使用C:\），这是由于PHP中的反斜杠是转义符。
- 如果新文件和现有的文件具有相同的名称，`move_uploaded_file()`函数将在不发出警告的情况下重写现有的文件。
- `MAX_FILE_SIZE`是浏览器中对文件大小的一种限制，尽管并非所有的浏览器都遵守这种限制。PHP配置文件具有它自己的限制。也可以在接收的PHP脚本内验证上传文件的大小。
- 在第13章，你将学习通过验证上传文件类型来提高脚本安全性的方法。

11

## 11.3 PHP 和 JavaScript

尽管PHP和JavaScript是两种根本不同的技术，但是可以结合使用它们来创建更好的Web站点。这两种语言之间最重要的区别是：JavaScript是客户端脚本（这意味着它运行在Web浏览器中），而PHP则是服务器端脚本。因此，JavaScript可以检测浏览器窗口的大小，创建弹出式窗口以及建立图像的悬停效果，而PHP则不能。

虽然PHP不能做JavaScript所能够做的某些事情，但是PHP可用于创建JavaScript（就像PHP可以创建HTML那样）。换句话说，网络浏览器中包含了JavaScript并用于与HTML进行交互，但是PHP可以动态生成JavaScript代码，就像你曾经使用PHP来动态生成HTML一样。

为了演示这个功能，我会创建一个PHP脚本，列出所有由`upload_image.php`脚本上传的图片（参见图11-17）。PHP脚本还将为每一幅图片的名字创建可点击的链接。链接本身将会调用能够创建弹出窗口的JavaScript函数（参见图11-18）。弹出窗口实际上会展示点击的图片。这个例子并不会深入的讨论JavaScript，但是它充分说明了PHP、HTML和JavaScript多种技术是可以一起使用的。在第15章中你将会学习如何使用jQuery（JavaScript框架）为基于PHP的脚本添加各种功能。



图11-17 这个PHP页动态创建了所有上传图片的列表

```
a0000686.jpg
empire_of_lights.jpg
inthecar.jpg
mayne-1071953684.jpg
trixie.jpg
```

图11-18 每个图片名字的链接会调用JavaScript函数。函数调用的参数是由PHP动态生成的

### 11.3.1 创建JavaScript文件

虽然JavaScript和PHP是两种不同的语言，但它们非常相似，即使没有经过任何正式的训练人也可以使用JavaScript。在创建本例的JavaScript代码之前，我会先解释一些基础知识。

首先，JavaScript可以通过两种方法被添加到HTML页面中：内嵌或通过外部文件。将JavaScript代码放置到HTML `script` 标签之间，添加内嵌的JavaScript：

```
<script type="text/javascript">
// Actual JavaScript code.
</script>
```

要使用外部JavaScript文件，为 `script` 标签添加 `src` 属性：

```
<script type="text/javascript" src="somefile.js"></script>
```

HTML页面可以包含多个 `script` 标签，但是每个 `script` 标签只能包含一个外部文件或包含一些JavaScript代码，但是不能同时使用。

在前面的代码中可以看到，JavaScript文件中使用 `.js` 扩展名。该文件应该使用与包含它的HTML脚本相同的编码（跟文本编辑器或IDE相同）。你可以在 `script` 标签中指定文件的编码：

```
<script type="text/javascript" charset="utf-8" src="somefile.js"></script>
```

无论你把JavaScript代码放到 `script` 标签中还是在外部文件中，JavaScript没有像PHP那样的开始和闭合的标签。

接下来要说明的是，JavaScript中的变量是区分大小写的，类似PHP，但是JavaScript中的变量不必

以\$符号开始。

最后，JavaScript和PHP之间的主要区别之一是，JavaScript是一种面向对象的编程（OOP）语言。而PHP既可以像本书的大部分演示那样以过程式的方法使用，同时可以以面向对象的方法使用（详见第16章），JavaScript只能作为一门面向对象的语言。这意味着你会看到类似`something.something()`或`something.something.something`的“点”语法。

以上就是要解释的所有基础知识。在接下来的脚本中我将仔细解释每一部分代码的详情。在接下来的一系列步骤中，你将创建一个单独的JavaScript文件，该文件定义了一个JavaScript函数。该函数将会接受三个参数：图片的名字、宽度和高度。该函数将使用这些值来为图片创建一个弹出式窗口。

### 使用PHP创建JavaScript

(1) 在文本编辑器或IDE中创建一个新的JavaScript文档，将其命名为`function.js`（参见脚本11-3）。

**脚本 11-3 function.js 脚本定义了一个 JavaScript 函数用于创建弹出窗口，将显示一个单独的图像**

```

1 // Script 11.3 - function.js
2
3 // Make a pop-up window function:
4 function create_window (image, width, height) {
5
6 // Add some pixels to the width and height:
7 width = width + 10;
8 height = height + 10;
9
10 // If the window is already open,
11 // resize it to the new dimensions:
12 if (window.popup && !window.popup.closed) {
13 window.popup.resizeTo(width, height);
14 }
15
16 // Set the window properties:
17 var specs = "location=no, scrollbars=no, menubars=no, toolbars=no, resizable=yes, left=0, top=0,
18 width=" + width + ", height=" + height;
19
20 // Set the URL:
21 var url = "show_image.php?image=" + image;
22
23 // Create the pop-up window:
24 popup = window.open(url, "ImageWindow", specs);
25 popup.focus();
26 } // End of function.
```

11

这里没有起始的JavaScript标签，你可以直接编写JavaScript代码。JavaScript中的注释可以使用单行（//）或多行（/\* \*/）语法。

(2) 开始JavaScript函数。

```
function create_window (image, width, height) {
```

JavaScript 的`create_window()`函数将接受三个参数：图片名、它的宽度和它的高度。当用户点击一个链接时，所有这些参数将被传递给这个函数。需要的图片名、宽度和高度的确切值将由PHP决定。

除了变量没有初始的\$符号外，JavaScript中创建函数的语法与PHP中的用户自定义函数语法非常相似。

(3) 为接收到的宽度和高度值添加10个像素。

```
width = width + 10;
height = height + 10;
```

宽度和高度值将加上一些像素值来创建一个比图片稍微大一点的窗口。JavaScript中数学运算使用的操作符几乎与所有的语言相同。

(4) 如果弹出窗口已经打开了，调整它的大小。

```
if (window.popup && !window.popup.closed) {
 window.popup.resizeTo(width, height);
}
```

在函数的后面将会创建一个窗口，关联到popup变量。如果用户单击了一个图片名，就创建弹出窗口，但如果在未关闭第一个弹出窗口的情况下单击其他的图片名字，新图片将会显示在尺寸错误的窗口中。为了防止这种情况，这里使用这点代码检查是否存在未关闭的弹出窗口。如果两个条件都为TRUE（这是说，该窗口已经打开），窗口将被调整为新图像尺寸。这是通过调用popup对象的resizeTo()方法来实现的（方法是函数的OOP术语）

(5) 确定弹出窗口的属性。

```
var specs = "location=no, scrollbars=no, menubars=no, toolbars=no, resizable=yes, left=0, top=0,
width=" + width + ", height=" + height;
```

此行代码创建了一个名为specs的新的JavaScript变量。在变量名之前使用var关键字是在函数内部创建变量是首选方法（具体是，它创建了函数的局部变量）。注意，image、width和height变量没有使用此关键字，因为它们是作为函数的参数创建的。

这个变量将用于确立弹出窗口的属性。窗口没有地址栏、滚动条、菜单或工具栏；它可以调整大小；它将位于屏幕的左上角；它的宽度将为width，高度将为height（参见图11-19）。



图11-19 通过JavaScript创建的弹出窗口

在JavaScript中，使用加号执行字符串连接（PHP使用点）。

(6) 定义URL。

```
var url = "show_image.php?image=" + image;
```

此代码设置弹出窗口的URL，也就是用来说明窗口应该载入什么页面。该页是*showimage.php*，将在本章的稍后创建。*showimage.php*脚本在URL中接收一个图片的名字，所以url变量的值是*show\_image.php?image=*加上图片的名字（参见图11-20）。

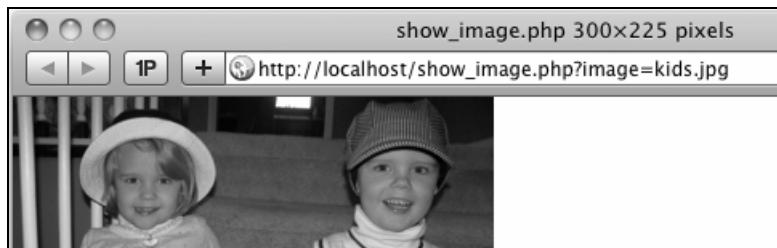


图11-20 弹出窗口的地址展示了图片值是如何通过URL传递的

(7) 创建弹出窗口。

```
popup = window.open(url, "ImageWindow", specs);
popup.focus();
```

最后，使用window对象的open()方法创建弹出窗口。window对象是由浏览器创建的，用于引用打开的窗口的全局JavaScript对象。open()方法的第一个参数是要加载的页面，第二个是窗口的标题，第三个是一个属性列表。需要注意的是，创建的窗口被赋给了popup变量。因为这个变量没有使用var关键字创建，popup将会是一个全局变量。如果要多次调用函数，引用相同的变量，就必须使用全局变量。

最后，将焦点设置给新的窗口，使它出现在当前窗口的上面。

(8) 将脚本保存为function.js。

(9) 将脚本或它的副本，放到Web目录的js文件夹中。

在组织你的网站目录时，JavaScript应该像CSS那样独立出来。通常情况下，外部JavaScript文件都放在名为js、javascript或者scripts的文件夹中。

### 11.3.2 创建PHP脚本

现在页面需要的JavaScript代码已经创建好了，接下来要创建PHP脚本（这将用于输出调用JavaScript函数的HTML）。这个脚本的目的是列出所有通过upload\_image.php上传的图片。要做到这一点，PHP需要动态检索上传目录的内容。这可以通过scandir()函数来实现，此函数返回一个数组，列出给定目录中的文件（该函数是在PHP5中添加的）。

PHP脚本必须链接每个显示的图片名字作为刚刚定义的JavaScript函数的调用。这个函数接受三个参数：图像的名称，它的宽度和高度。在PHP中，脚本将使用getimagesize()函数查找后面的两个值。它返回给定图片的信息数组（参见表11-3）。

表11-3 getimagesize()数组

元 素	值	示 例
0	图像的宽度 (以像素为单位)	423
1	图像的高度 (以像素为单位)	368
2	图像的类型	2 (表示JPG)
3	合适的HTML img 标签数据	height="368" width="423"
mime	图像的MIME类型	image/png

### 使用PHP创建JavaScript

(1) 在文本编辑器或IDE中创建一个新的PHP文件，命名为images.php（脚本11-4）。

**脚本 11-4 images.php** 脚本使用 JavaScript 和 PHP 来创建存储在服务器上的图片的链接。图片可以通过 show\_image.php 来查看

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
2 xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 <title>Images</title>
7 <script type="text/javascript" charset="utf-8" src="js/function.js"></script>
8 </head>
9 <body>
10 <p>Click on an image to view it in a separate window.</p>
11
12 <?php # Script 11.4 - images.php
13 // This script lists the images in the uploads directory.
14
15 $dir = '../uploads'; // Define the directory to view.
16
17 $files = scandir($dir); // Read all the images into an array.
18
19 // Display each image caption as a link to the JavaScript function:
20 foreach ($files as $image) {
21
22 if (substr($image, 0, 1) != '.') { // Ignore anything starting with a period.
23
24 // Get the image's size in pixels:
25 $image_size = getimagesize ("$dir/$image");
26
27 // Make the image's name URL-safe:
28 $image_name = urlencode($image);
29
30 // Print the information:
31 echo "<a href=\"javascript:
32 create_window('$image_name',
33 $image_size[0],$image_size[1])\">
34 $image\n";
35
36 } // End of the IF.
37
38 }
```

```

34 } // End of the foreach loop.
35 ?>
36
37 </body>
38 </html>

```

(2) 包含JavaScript文件。

```
<script type="text/javascript" charset="utf-8" src="js/function.js"></script>
```

`script`标签可以用在HTML页面的任何地方，但是通常在文档的头部包含外部的文件。对`function.js`的引用假定文件将会在`js`目录中，并且`js`目录与当前的脚本在相同的目录（参见“处理文件上传”下面的图11-8）。

(3) 完成HTML头并开始主体。

```

</head>
<body>
<p>Click on an image to view it in a separate window.</p>

```

(4) 创建一个HTML无序列表。

```

```

为了让事情变得简单，这个脚本会将每个图像的显示为无序列表中的一个项目。

(5) 开始PHP代码，并通过引用上传目录创建一个图片的数组。

```

<?php # Script 11.4 - images.php
$dir = '../uploads';
$files = scandir($dir);

```

这个脚本会自动列出并链接上传文件夹中的所有存储的图像（可能是`upload_image.php`放到那的，脚本11-3）。该代码首先将目录定义为一个变量，因此让它易于引用。然后用于返回一个文件夹内文件和目录数组的`scandir()`函数，将那个信息赋给一个名为`$files`的数组。

(6) 开始遍历`$files`数组。

```

foreach ($files as $image) {
 if (substr($image, 0, 1) != '.') {

```

这个循环将会变量数组中的每个图片，并为它创建一个列表项。在循环中，有一个条件语句用于检查文件名的第一个字符是否为一个点。在非Windows系统中，隐藏文件使用一个点开始，当前目录使用一个单独的点来指代，并且使用两个点来表示父目录。由于所有这些可能被包含在`$files`中，它们需要被筛选。

(7) 获取的图像信息并编码它的名字。

```

$image_size = getimagesize ("$dir/$image");
$image_name = urlencode($image);

```

`getimagesize()`函数返回一个图片的信息数组（参见表11-3）。这个函数返回的值将会用于设置发送到`create_window()`JavaScript函数的宽度和高度。

接下来，使用`urlencode()`函数来让字符串可以安全的用于一个URL中。由于图像的名称可能包含不允许在URL中使用的字符（并且它将在调用`show_image.php`时通过URL传递），这个名字需要进行编码。

(8) 打印列表项。

```
echo "
$image\n";
```

最后，循环创建了HTML列表项，包括链接的图像名称。链接是一个对JavaScript `create_window()` 函数的调用。要在HTML内部运行JavaScript函数，在函数前面加上`javascript:`。（在HTML内部调用JavaScript的方法很多，但现在只是用这个语法）。

该函数的三个参数是：图像的名称、图像的宽度和图像的高度。由于图像的名称将是一个字符串，它必须使用引号包裹。

(9) 完成if条件语句，foreach循环和PHP部分。

```
} // End of the IF.
} // End of the foreach loop.
?>
```

(10) 完成无序列表和HTML页面。

```

</body>
</html>
```

(11) 将文件保存为`images.php`，将其放置在你的Web目录中（在与`upload_image.php`相同的目录下），并在你的Web浏览器中测试它（参见图11-18）。

请注意，点击链接还将无法正常工作，因为`show_image.php`，弹出窗口试图加载的页面，还未创建。

(12) 查看源代码，查看动态生成的链接（参加图11-17）。

请注意函数调用的每个参数都是适当的特定图像。

#### ✓ 提示

- 不同的浏览器会有不同的方法来处理窗口的调整大小。在我的测试中，例如，谷歌浏览器总是需要该窗口至少有确定的宽度和IE将为显示的图像的4个边添加边距。
- 某些版本的Windows会在图片文件夹创建一个Thumbs.db文件。你可能需要在步骤(6)中的条件语句中检查这个值，并且从返回值中筛选它。该代码将是：

```
if ((substr($image, 0, 1) != '.') && ($image != 'Thumbs.db')) {
```

- 我不想过多的讨论这一点，但几乎所有Web开发人员使用JavaScript（例如，调整大小或移动浏览器窗口）实现的功能都不能使用服务器端PHP完成。
- PHP和JavaScript之间有一小点的重叠。两者都可以设置和读取cookie，创建HTML，和做一些浏览器检测。

## 11.4 理解HTTP头部

刚刚创建的`images.php`脚本，显示了一个图像名称的列表，每一个都链接到一个JavaScript函数调用。这个JavaScript函数创建了一个弹出窗口，它加载了一个实际上会显示一个图片的PHP脚本。这可能听起来有点事倍功半，但是我的疯狂有个方法。这种方法的一个微不足道的原因是，JavaScript是创

建一个适应图片大小尺寸的窗口所必需的（而不是创建一个包含一个图片的任意大小的弹出窗口）。更重要的是，因为图像被保存在上传目录，理想的情况是保存在Web根目录的外面，图片无法通过一下的两种方式直接查看：

```
http://www.example.com/uploads/image.png
```

或者

```

```

之所以上面的两种都无法正常工作的原因是，文件和文件夹都在Web根目录的外面，顾名思义，无法通过Web浏览器进行访问。其实，这是一件好事，因为它允许你保护内容，仅在合适的时候提供它。要让这些内容可以通过Web浏览器进行访问，你需要创建一个PHP的代理脚本。一个代理脚本进满足一个角色，例如提供一个文件（在浏览器中，显示一个图片与提供一个文件实际上是相同的事情）。因此，上面的例子中给定的代理脚本proxy.php，可以使用一下任意一种方式（参见图11-21）：

```
http://www.example.com/proxy.php? image=image.png
```

或者

```

```

这一点，当然，正是链接到create\_window()的JavaScript函数的show\_image.php所实现的。但proxy.php或show\_image.php是如何工作的？答案就在于了解的HTTP头。

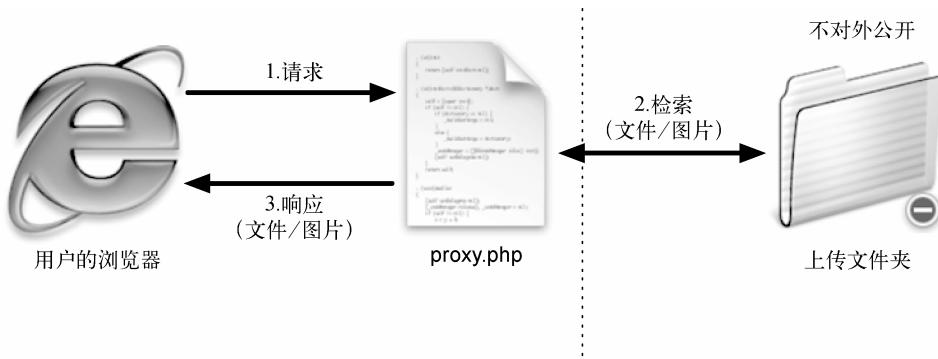


图11-21 一个代理脚本可以提供对其他方法无法访问的服务器内容的访问

HTTP（Hypertext Transfer Protocol，超文本传输协议）是WWW（World Wide Web，万维网）的核心技术，它定义了客户和服务器通信的方式（用外行的话讲）。当浏览器请求Web页面时，作为响应，它会接收到一系列HTTP头。当然，这是在幕后发生的，大多数用户根本不知道这些。

PHP的内置函数header()可用于利用这种协议。下一章中将演示它的一个最常见的例子，即把header()函数用于将Web浏览器从当前页面重定向到另一个页面。这里，你将使用它把文件发送给Web浏览器。

理论上讲，函数 header()很容易使用，其语法如下：

```
header(header string);
```

可能的头部字符串列表相当长，因为头部可用于任何事情，从重定向Web浏览器到发送文件，到发送cookie，再到控制页面缓存，等等。从简单的事情开始，要使用header()重定向Web浏览器，可输入：

```
header ('Location: http://www.example.com/page.php');
```

这一行代码将把Web浏览器从它所在的页面发送到那个URL。

在下一个示例中（它将把文件发送到Web浏览器），将使用三个头部调用。第一个是Content-Type，它指示Web浏览器其后将接着什么类型的数据。Content-Type值与数据的MIME类型匹配。下面这一行代码使浏览器知道它将接收到一个PDF文件：

```
header("Content-Type:application/pdf\n");
```

接下来，可以使用Content-Disposition，它告诉浏览器如何处理数据：

```
header ("Content-Disposition: attachment; filename=\"somefile.pdf\"\n");
```

attachment值提示浏览器下载文件（参见图11-22）。一种替代方法是使用inline，它告诉浏览器显示数据，假定浏览器可以这样做。filename属性的作用如下：它告诉浏览器与数据关联的名称。

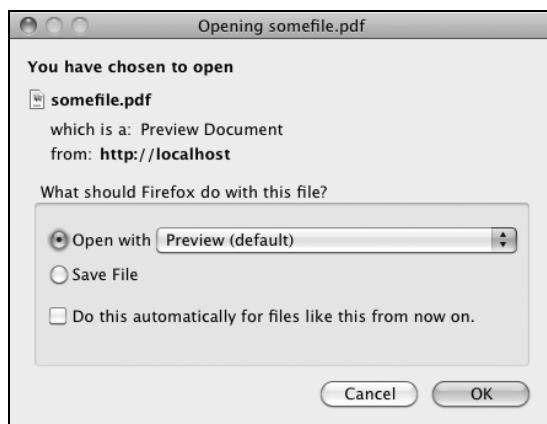


图11-22 由于Content-Disposition值为attachment，所以Firefox提示用户下载文件

用于下载文件的第三个头部是Content-Length。这个值（以字节为单位）对应于要发送的数据量。

```
header ("Content-Length: 4096\n");
```

这些是关于使用header()函数的基础知识。在开始创建示例前，注意如果一个脚本使用多个header()调用，应该用换行符(\n)终止每个header()调用，如上一个代码段中所做的那样。更重要的是，关于header()函数要记住的绝对关键的事情是，必须在把任何内容发送给Web浏览器之前调用它。这包括HTML，或者甚至是空白。如果你的代码在调用header()之前具有任何echo或print语句，把空白行保持在PHP标签外面，或者包含做任何事情的文件，则会看到一条出错消息，如图11-23所示。

```
Warning: Cannot modify header information - headers already sent by (output started at
/Users/larryullman/Sites/phpmysql4/proxy.php:2) in
/Users/larryullman/Sites/phpmysql4/proxy.php on line 3

Warning: Cannot modify header information - headers already sent by (output started at
/Users/larryullman/Sites/phpmysql4/proxy.php:2) in
/Users/larryullman/Sites/phpmysql4/proxy.php on line 4
```

图11-23 在使用header()函数之前，经常会看到headers already sent错误，它意味着给Web浏览器发送了某些内容——HTML、纯文本，甚至包括空格

### 使用header()函数

(1) 在文本编辑器或IDE中创建一个新的PHP页面（参见脚本11-5）。

#### 脚本 11-5 这个脚本将从服务器上获取一幅图像，并把它发送给浏览器

```
1 <?php # Script 11.5 - show_image.php
2 // This page displays an image.
3
4 $name = FALSE; // Flag variable:
5
6 // Check for an image name in the URL:
7 if (isset($_GET['image'])) {
8
9 // Make sure it has an image's extension:
10 $ext = strtolower (substr ($_GET['image'], -4));
11
12 if (($ext == '.jpg') OR ($ext == 'jpeg') OR ($ext == '.png')) {
13
14 // Full image path:
15 $image = "../uploads/{$_GET['image']}";
16
17 // Check that the image exists and is a file:
18 if (file_exists ($image) && (is_file($image))) {
19
20 // Set the name as this image:
21 $name = $_GET['image'];
22
23 } // End of file_exists() IF.
24
25 } // End of $ext IF.
26
27 } // End of isset($_GET['image']) IF.
28
29 // If there was a problem, use the default image:
30 if (!$name) {
31 $image = 'images/unavailable.png';
32 $name = 'unavailable.png';
33 }
34
35 // Get the image information:
36 $info = getimagesize($image);
37 $fs = filesize($image);
```

```

38 // Send the content information:
39 header ("Content-Type: {$info['mime']}\\n");
40 header ("Content-Disposition: inline; filename=\"$name\"\\n");
41 header ("Content-Length: $fs\\n");
42
43
44 // Send the file:
45 readfile ($image);

```

由于这个脚本将使用`header()`函数，因此绝对不能把任何信息发送给Web浏览器。在PHP开始标签前面不能有HTML，甚至还不能有空白行、制表符或空格。`$name`变量将用作一个标志，指示是否通过了所有的验证例程。

### (2) 检查图像名称。

```
if (isset($_GET['image'])) {
```

该脚本需要在URL中接收一个有效的图像名称。应该在调用这个页面的JavaScript函数中把它追加到URL中（参见脚本11-3，即`function.js`）。

### (3) 验证图像的扩展名。

```
$ext = strtolower (substr ($_GET['image'], -4));
if (($ext == '.jpg') OR ($ext == 'jpeg') OR ($ext == '.png')) {
```

最终的检查是要发送到Web浏览器的文件具有`.jpeg`、`.jpg`或`.png`扩展名。这样，脚本将不会尝试把糟糕的内容发送给用户。例如，如果恶意用户把弹出窗口的地址`http://www.example.com/showimage.php?image=image.png`改为`http://www.example.com/showimage.php?image=../../../../path/to/something/important`，那么这个条件就会捕获并阻止这种攻击。

为了验证扩展名，`substr()`函数将从图像的名称中返回最后4个字符（`-4`用于完成这个任务）。扩展名还会被`strtolower()`函数处理，使得`.PNG`和`.png`将被视作是相同的。然后，一个条件语句将检查`$ext`是否等于三个允许的值中的任何一个值。

### (4) 检查该图像是否是服务器上的一个文件。

```
$image = "../uploads/{$_GET['image']}";
if (file_exists ($image) && (is_file($image))) {
```

在尝试把该图像发送给Web浏览器之前，确保它存在并且它是一个文件（与目录相对）。作为一种安全措施，我把图像的完整路径硬编码为`../uploads`与接收到的图像名称的组合。

### (5) 设置标记变量的值为图片名称。

```
$name = $_GET['image'];
```

一旦图像通过了所有这些测试，就会把图像的值赋予`$name`变量。

### (6) 完成第(2)步、第(3)步和第(4)步中开始的条件语句。

```
} // End of file_exists() IF.
} // End of $ext IF.
} // End of isset($_GET['image']) IF.
```

### (7) 如果这个页面没有接收到有效的图像，则将使用默认的图像。

```
if (!$name) {
$image = 'images/unavailable.png';
```

```
$name = 'unavailable.png';
}
```

如果图像不存在，或者它不是一个文件，又或者它没有合适的扩展名，那么\$name变量的值仍将是FALSE。在这些情况下，将代之以使用默认的图像（参见图11-24）。可以从本书对应的Web站点下载图像，它应该位于images文件夹中。images文件夹应该位于和这个脚本相同的目录内，而在与uploads文件夹相同的目录内。



图11-24 无论何时在显示请求的图像时出现问题，都会显示这幅图像

#### (8) 获取图像信息。

```
$info = getimagesize($image);
$fs = filesize($image);
```

要把文件发送给Web浏览器，脚本需要知道文件的类型和大小。可以使用getimagesize()查明文件的类型，并使用filesize()查明文件的大小（以字节为单位）。由于\$image变量代表..../uploads/{\$GET['image']}或images/unavailable.png，所以这几行代码既可处理正确的图像，也可处理不可用的图像。

#### (9) 发送文件。

```
header ("Content-Type: {$info['mime']}\\n");
header ("Content-Disposition: inline; filename=\"$name\\n\"");
header ("Content-Length: $fs\\n");
```

这些header()调用将把文件数据发送给Web浏览器。第一行把图像的MIME类型用于Content-Type。第二行告诉浏览器文件的名称以及它应该显示在浏览器中（inline）。最后一个header()函数指示预期有多少数据。文件数据本身是使用readfile()函数发送的，它读取一个文件并立即把内容发送给Web浏览器。

(10) 将文件另存为show\_image.php，将其存放在Web目录中（它在与images.php相同的文件夹中），并通过单击images.php中的链接在Web浏览器中测试它（参见图11-25）。

请注意，此页面不包含任何HTML。它仅发送一个图片文件到网络浏览器。还要注意，我省略了PHP的结束标签。这是可以接受的，并且在当前这种特定情形下推荐这么做。如果包含了PHP的结束标签，如果你不经意在结束标签后面加了多余的空格或空行，浏览器可能无法显示图片（因为浏览器接收到与Content-Length头相匹配的X长度的图像数据后，还会多接受一点多余的数据）。



图11-25 当PHP把文件发送给Web浏览器时将显示这幅图像

### ✓ 提示

- 有一点无论怎样强调也不过分，即在使用header()函数之前，不能把任何内容发送到Web浏览器。即使包含文件在PHP结束标签后面有一个空白行，这也会使得header()函数不可用。
- 在使用header()时为了避免出现问题，可以先调用headers\_sent()函数。它返回一个布尔值，指示是否已经把某些内容发送给了Web浏览器：

```
if (!headers_sent()) {
 // Use the header() function.
} else {
 // Do something else.
}
```

输出缓冲技术（将在第17章中介绍）也可以阻止在使用header()时出现问题。

- 调试像这样的脚本可能具有挑战性，其中PHP将把数据（而不是文本）发送给Web浏览器。为了获得帮助，可使用用于Firefox的Live HTTP Headers插件（参见图11-26）。

 Response Headers - http://localhost/show\_image.php?image=kids.jpg

Date: Wed, 20 Apr 2011 03:20:52 GMT
Server: Apache
X-Powered-By: PHP/5.3.2
Content-Disposition: inline; filename="kids.jpg"
Content-Length: 106403
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: image/jpeg
200 OK

图11-26 用于Firefox的Live HTTP Headers扩展显示页面和/或服务器发送的是什么头部，这可能是有用的调试信息

- 你还可以使用PHP和header()函数来制定网络浏览器的页面编码：

```
<?php header ('Content-Type: text/html; charset=UTF-8'); ?>
```

这比使用META标签更加有效，但是它需要页面是一个PHP脚本。如果使用这个方法，它必须在页面的第一行，所有HTML代码的前面。

- 代理脚本一次只能发送一个文件（或图片）到浏览器。

## 11.5 日期和时间函数

第5章演示了MySQL支持的许多极好的日期和时间函数。当然，PHP具有它自己的日期和时间函数。首先，有一个date\_default\_timezone\_set()函数，用于建立默认的时区（也可以在PHP的配置文件中设置它）。

```
date_default_timezone_set(tz);
```

tz值是一个像America/New\_York或Pacific/Auckland这样的字符串。这里要列出的内容太多（仅仅非洲就有50多个），但是查看PHP手册就可以了解所有这些信息。注意：从PHP 5.1起，必须在调用任何日期和时间函数前设置默认的时区，否则将会看到一个错误。

接下来，checkdate()函数获取月份、天和年份作为参数，并返回一个布尔值，指示日期是否实际存在。它甚至会考虑到闰年。该函数可用于确保用户提供了有效的日期（生日或其他日期）：

```
if (checkdate(month, day, year)) { // OK!
```

也许最常用的函数是适当命名的date()函数。它以格式化的字符串返回日期和/或时间。它带有两个参数：

```
date(format, [timestamp]);
```

时间戳是一个可选参数，表示正在处理的日期从UNIX新纪元（1970年1月1日午夜）起直到当前时刻的秒数。它允许你获取特定日期的信息，比如某个日期是星期几。如果没有指定时间戳，PHP将只会使用服务器上的当前时间。

可以使用大量的格式化参数（参见表11-4），并且可以把它们与文本值结合使用。例如：

```
echo date('F j, Y'); // January 26, 2011
echo date('H:i'); // 23:14
echo date('D'); // Sat
```

11

表11-4 date()函数格式化的返回结果

字 符	含 义	示 例
Y	年份，用4位数字表示	2011
y	年份，用两位数字表示	11
L	是否为闰年	1 (为是)
n	月份，用一位或两位数字表示	2
m	月份，用两位数字表示	02
F	月份	February
M	月份，用3个字母表示	Feb
j	一月中的某一天，用一位或两位数字表示	8

(续)

字 符	含 义	示 例
d	一月中的某一天，用两位数字表示	08
l (小写L)	星期几	Monday
D	星期几，用3个字母表示	Mon
w	每周的星期几，以一位数字表示	O (星期日)
z	一年之中的日期：0~365	189
t	一月之中的日期	31
S	英语中每一天的后缀	rd
g	小时，12时制，用一位或两位数字表示	6
G	小时，24时制，用一位或两位数字表示	18
h	小时，12时制，用两位数字表示	06
H	小时，24时制，用两位数字表示	18
i	分钟	45
s	秒	18
u	微秒	1234
a	am或pm	am
A	AM或PM	PM
U	从新纪元开始的秒数	1048623008
e	时区	UTC
I (大写i)	是否为夏令时	I (为是)
O	与GMT的差距	+0600

可以使用**mktime()**函数找到特定日期的时间戳。

```
$stamp = mktime (hour, minute, second, month, day, year);
```

如果不带参数地调用**mktime()**函数，它将返回当前时间戳，这等同于调用**time()**函数。

最后，**getdate()**函数可用于返回日期和时间的值的数组（参见表11-5）。例如：

```
$today = getdate();
echo $today['month']; // October
```

表11-5 **getdate()**函数返回这个关联数组

键	值	示 例
year	年份	2011
mon	月份	11
month	月份名称	November
mday	一月中的某一天	24
weekday	星期几	Tuesday
hours	小时	11
minutes	分钟	56
seconds	秒	47

这个函数也接受一个可选的时间戳参数。如果没有使用该参数, `getdate()`就会返回当前日期和时间信息。

这些只是PHP具有的许多日期和时间函数中的一部分。有关更多函数, 可参见PHP手册。要实际使用这些函数, 可以以一种完全不必要的方式修改`images.php` (参见脚本11-4)。脚本首先显示每个图片上传的日期和时间。接下来, 当发生变化时, 脚本还会显示每个图片文件的大小 (参见图11-27)。

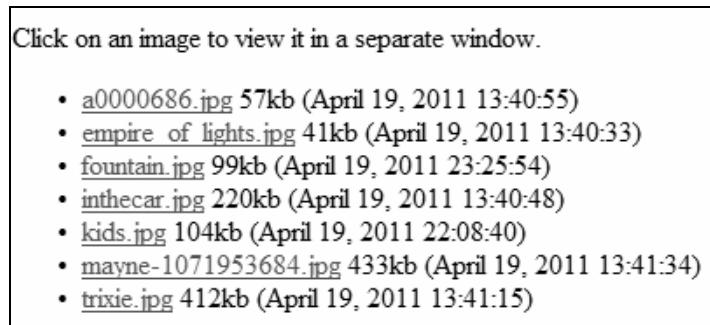


图11-27 修改后的`images.php`会为每张图片显示另外两项信息

### 使用日期和时间函数

- (1) 在文本编辑器或IDE中打开`images.php` (参见脚本11-4)。
- (2) 在开始PHP标签后面的第一行上, 建立时区 (参见脚本11-6)。

**脚本 11-6** `images.php` (参见脚本 11-4) 的这个修改过的版本使用了 PHP 的日期和时间函数, 以便报告一些信息给用户

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Images</title>
6 <script type="text/javascript" charset="utf-8" src="js/function.js"></script>
7 </head>
8 <body>
9 <p>Click on an image to view it in a separate window.</p>
10
11 <?php # Script 11.6 - images.php
12 // This script lists the images in the uploads directory.
13 // This version now shows each image's file size and uploaded date and time.
14
15 // Set the default timezone:
16 date_default_timezone_set ('America/New_York');
17
18 $dir = '../uploads'; // Define the directory to view.
19
20 $files = scandir($dir); // Read all the images into an array.

```

```

21 // Display each image caption as a link to the JavaScript function:
22 foreach ($files as $image) {
23
24 if (substr($image, 0, 1) != '.') { // Ignore anything starting with a period.
25
26 // Get the image's size in pixels:
27 $image_size = getimagesize ("$dir/$image");
28
29 // Calculate the image's size in kilobytes:
30 $file_size = round ((filesize ("$dir/$image")) / 1024) . "kb";
31
32 // Determine the image's upload date and time:
33
34 $image_date = date("F d, Y H:i:s", filemtime("$dir/$image"));
35
36 // Make the image's name URL-safe:
37 $image_name = urlencode($image);
38
39 // Print the information:
40 echo "<a href=\"javascript:create_window('$image_name',$image_size[0], $image_
size[1])\">$image $file_size ($image_date)\n";
41
42 } // End of the IF.
43
44 } // End of the foreach loop.
45
46 ?>
47
48 </body>
49 </html>

```

在调用任何日期和时间函数之前（这个脚本将调用两个不同的日期和时间函数，并且每个函数调用两次），必须先建立时区。为了查明你的时区，可参见[www.php.net/timezones](http://www.php.net/timezones)。

(3) 在foreach循环中，得到图片的尺寸之后计算它的文件大小。

```
$file_size = round ((filesize ("$dir/$image")) / 1024) . "kb";
```

filesize()函数是首次在show\_image.php脚本中使用。它按byte值返回文件的大小。要计算文件的千字节，将数字除以1024（千字节中有的字节数）并取整。

(4) 下一行确定图片的修改日期和时间。

```
$image_date = date("F d, Y H:i:s", filemtime("$dir/$image"));
```

要查找一个文件的修改日期和时间，调用filemtime()函数，为函数提供要检查的文件或目录。这个函数返回一个时间戳，可以用于date()函数的第二个参数，以相应地格式化这个时间戳。

如果你不太清楚这里发生了什么，可以将代码分两步写：

```
$filetime = filemtime ("$dir/$image");
$image_date = date("F d, Y H:i:s ", $filetime);
```

(5) 修改echo语句，让它还可以打印文件大小和修改日期。

```
echo "
$image $file_size ($image_date)\n";
```

这两项都打印在<A>标签的外面，所以它们都不是链接的一部分。

(6) 将文保存为images.php，存放到站点目录并在浏览器中测试。

### ✓ 提示

- date()函数有一些用于信息提供而不是格式化的参数。例如，date('L')根据是不是闰年返回1或者0、date('t')返回当月的天数、date('I')会在夏令时时返回1。
- PHP的日期函数反映了服务器上的时间（因为PHP在服务器上运行），如果你想确定用户的计算机上的日期和时间，你需要使用JavaScript。
- 在第16章，你将会学习如何在PHP中使用新的DateTime类处理日期和时间。

## 11.6 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 11.6.1 回顾

- 用于发送电子邮件的函数是什么？函数的参数分别是？服务器需要什么来发送邮件？
- 对于\n，使用单引号和双引号有何不同？
- 你能否很容易地知道接收者是否或何时收取了由PHP发送的邮件？
- 如果你没有收到由PHP脚本发送的任何邮件，该采取哪些调试步骤？
- 在处理上传的文件时，文件夹的权限是如果发挥作用的？
- 处理文件上传时用要用到哪两个目录？
- 表单的开始标签中必须添加什么属性，以便用来处理文件上传？
- 什么是MIME类型？
- PHP和JavaScript有哪些相似的地方？它们又有何不同？
- 将JavaScript添加到HTML页面的标签是什么？
- JavaScript中的var关键字是什么意思？
- JavaScript中的连接运算符是什么？
- PHP的header()函数的作用是什么？
- 头文件已发送的错误信息是什么意思？
- 什么是代理脚本？什么时候可能会需要代理脚本？
- readfile()函数的作用是？

11

### 11.6.2 实践

- 创建自定义联系人表单。让PHP脚本发送有更多自定义内容的邮件，包括表单中请求的任何其他数据。

- 在线搜索 *php email spam filters* 关键词，学习提高 PHP 邮件投递成功率的技巧（即，尽量避免垃圾邮件过滤器过滤掉合法的邮件）。
- 修改 `upload_image.php` 以支持上传不同的文件类型。创建一个相应版本的 `show_image.php`。注意：你需要先研究下 MIME 类型才能完成这个挑战。
- 在 PHP 手册中检查 `glob()` 函数。
- 本章介绍了大量的信息和新的函数。查看 PHP 手册，学习更多相关内容。



### 本章内容

- 建立登录页面
- 创建登录函数
- 使用cookie
- 使用会话
- 提高会话安全性
- 回顾和实践

**H**TTP是一种无状态技术，这意味着每个单独的HTML页面都是一个无关的实体。当人们穿过站点时，HTTP没有用于跟踪用户或保持变量的方法。尽管浏览器会跟踪你访问过的页面，但是服务器不会记录谁看到过什么内容。如果服务器不能够跟踪用户，就不可能有购物车或自定义个性化Web站点的存在。使用服务器端技术（如PHP），可以克服Web的无状态性。可用于此目的的两种最佳的PHP工具是cookie和会话。

cookie和会话之间的关键区别是，cookie将数据存储在用户的浏览器里，而会话把数据存储在服务器上面。会话一般比cookie更安全，并且可以存储多得多的信息。这两种技术都很容易与PHP一起使用，并且值得了解。在本章中，你将看到cookie和会话的使用。演示该信息的示例将是一个登录系统，它基于现有的users数据库。

### 12.1 建立登录页面

登录过程涉及几个组成部分（参见图12-1）：

- 用于提交登录信息的表单；
- 确认必要信息已提交的验证例程；
- 比较提交的信息与存储的信息的数据库查询；
- 用于存储反映成功登录的数据的cookie或会话。

后续页面然后将包含一些检查，用于确认用户已登录（用于限制对该页面的访问）。当然，还有一个注销过程，它涉及清除表示登录状态的cookie或会话数据。

首先，让我们获取其中一些常见的元素并把它们存放在单独的文件中。然后，需要这种功能的页面可以包含必要的文件。像这样分解逻辑将使得接下来的一些脚本更容易阅读和编写，并有助于减少

它们之间的冗余性。我设计了两个可包含的文件。第一个文件将包含登录页面的大部分内容，包括：标题、错误报告、表单和脚注（参见图12-2）。

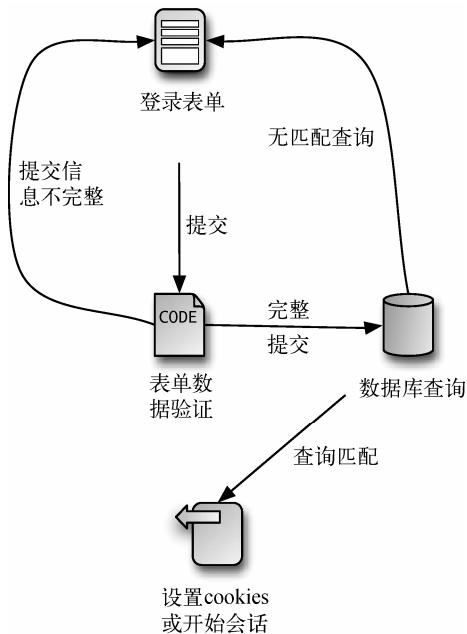


图12-1 登录过程

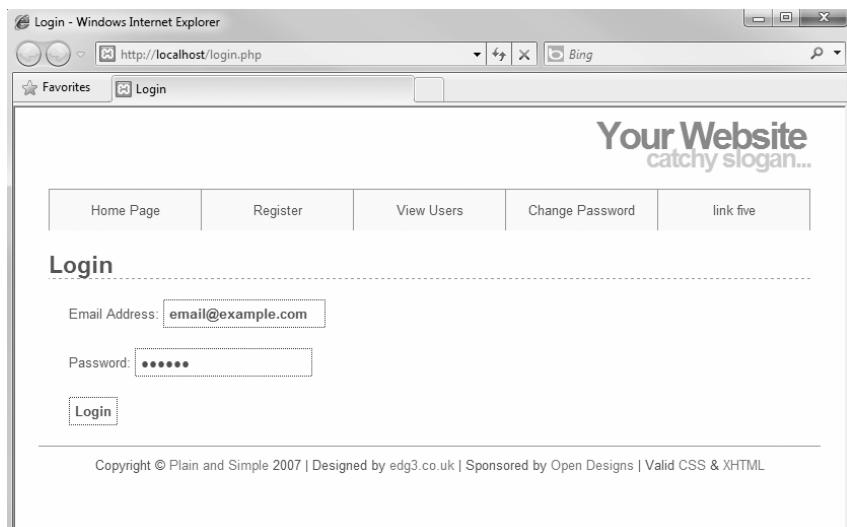


图12-2 登录表单和页面

## 建立登录页面

(1) 在文本编辑器或IDE中创建一个新的PHP页面，命名为login\_page.inc.php（参见脚本12-1）。

**脚本 12-1 login\_page.inc.php 脚本创建完整的登录页面（包括表单）并报告错误。需要显示登录页面的其他页面将包含它**

```

1 <?php # Script 12.1 - login_page.inc.php
2 // This page prints any errors associated with logging in
3 // and it creates the entire login page, including the form.
4
5 // Include the header:
6 $page_title = 'Login';
7 include ('includes/header.html');
8
9 // Print any error messages, if they exist:
10 if (isset($errors) && !empty($errors)) {
11 echo '<h1>Error!</h1>
12 <p class="error">The following error(s) occurred:
';
13 foreach ($errors as $msg) {
14 echo " - $msg
\n";
15 }
16 echo '</p><p>Please try again.</p>';
17 }
18
19 // Display the form:
20 ?><h1>Login</h1>
21 <form action="login.php" method="post">
22 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" /> </p>
23 <p>Password: <input type="password" name="pass" size="20" maxlength="20" /></p>
24 <p><input type="submit" name="submit" value="Login" /></p>
25 </form>
26
27 <?php include ('includes/footer.html'); ?>
```

(2) 包括标题。

```
$page_title = 'Login';
include ('includes/header.html');
```

本章将利用最初在第3章创建然后在第9章修改过的相同模板系统。

(3) 如果有任何出错消息，就打印它们。

```
if (isset($errors) && !empty($errors)) {
 echo '<h1>Error!</h1>
 <p class="error">The following error(s) occurred:
';
 foreach ($errors as $msg) {
 echo " - $msg
\n";
 }
 echo '</p><p>Please try again.</p>';
}
```

这段代码也是在第9章中开发的，只是添加了`isset()`子句防止安全隐患。如果有任何错误存在（在`$errors`数组变量中），就会把它们打印成一份无序列表（参见图12-3）。

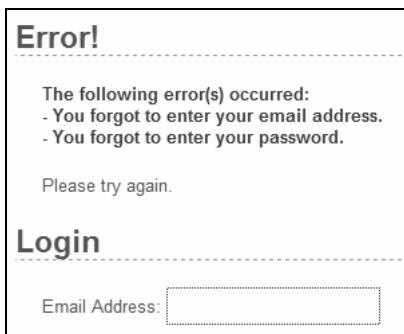


图12-3 带有错误报告的登录表单和页面

(4) 显示表单。

```
?><h1>Login</h1>
<form action="login.php" method="post">
 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" /> </p>
 <p>Password: <input type="password" name="pass" size="20" maxlength="20" /></p>
 <p><input type="submit" name="submit" value="Login" /></p>
</form>
```

HTML表单只需要两个文本输入框：一个用于电子邮件地址，另一个用于密码。输入框的名称与sitename数据库（登录系统就基于该数据库）的users表中的那些列名称匹配。

为了更容易创建HTML表单，首先关闭PHP部分。该表单不是黏性表单，但是可以轻松地添加代码来实现这一点（但是仅仅对于电子邮件地址可以如此，密码则不能是黏性的）。

(5) 完成页面。

```
<?php include ('includes/footer.html'); ?>
```

(6) 将文件另存为login\_page.inc.php，将其存放在Web目录中（该目录与第3章和第9章中的文件header.html、footer.html和style.css都在includes文件夹中）。

该页面将使用.inc.php扩展名，指示它是一个可包含的文件并且它包含PHP代码。

#### ✓ 提示

- 这个脚本包含includes目录内的标题和脚注文件似乎不合乎逻辑，因为该脚本也在相同的目录内。这段代码会工作，因为该脚本将被主目录内的页面包含；因此包含引用是关于父目录的，而不是关于这个目录的。

## 12.2 创建登录函数

除了存储在login\_page.inc.php中的登录页面外，还有一些功能在本章中的多个脚本中也很常见。在下一个脚本中（登录/注销系统中的其他页面也会包含该脚本），将定义两个函数。

许多页面最终将把用户从一个页面重定向到另一个页面。例如，在成功登录后，将把用户带到loggedin.php。如果用户访问loggedin.php，但他们没有登录，就应该把他们带到index.php。重定向使用第11章中介绍的header()函数。用于重定向的语法如下：

```
header ('Location: http://www. example.com/page.php');
```

由于这个函数将把浏览器发送到page.php，应该在这之后立即使用exit()终止当前脚本：

```
header ('Location: http://www. example.com/page.php');
exit();
```

如果不这样做，当前脚本将会继续运行（只是不在Web浏览器中运行而已）。

header()调用中的位置值应该是绝对URL（www.example.com/page.php，而不仅仅是page.php）。你可以硬编码这个值，不过更好的做法是动态确定它。下一个脚本中的第一个函数就用于执行该任务，然后重定向用户到该绝对URL。

本章中的多个脚本将使用的另外一些代码用于验证登录表单。这是一个包含三个步骤的过程：

(1) 确认提供了电子邮件地址；

(2) 确认提供了密码；

(3) 确认提供的电子邮件地址和密码与数据库中存储的电子邮件地址和密码匹配（在注册过程中）。

因此，下一个脚本将定义两个不同的函数。在接下来的步骤中将解释每个函数是如何工作的。

### 创建登录函数

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为login\_functions.inc.php（参见脚本12-2）。

**脚本 12-2 login\_functions.inc.php** 脚本中定义了两个函数，将被登录/注销过程中的不同脚本使用

```
1 <?php # Script 12.2 - login_functions.inc.php
2 // This page defines two functions used by the login/logout process.
3
4 /* This function determines an absolute URL and redirects the user there.
5 * The function takes one argument: the page to be redirected to.
6 * The argument defaults to index.php.
7 */
8 function redirect_user ($page = 'index.php') {
9
10 // Start defining the URL...
11 // URL is http:// plus the host name plus the current directory:
12 $url = 'http://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['PHP_SELF']);
13
14 // Remove any trailing slashes:
15 $url = rtrim($url, '/\\');
16
17 // Add the page:
18 $url .= '/' . $page;
19
20 // Redirect the user:
21 header("Location: $url");
22 exit(); // Quit the script.
23
24 } // End of redirect_user() function.
25
26
27 /* This function validates the form data (the email address and password).
28 * If both are present, the database is queried.
```

12

```

29 * The function requires a database connection.
30 * The function returns an array of information, including:
31 * - a TRUE/FALSE variable indicating success
32 * - an array of either errors or the database result
33 */
34 function check_login($dbc, $email = '', $pass = '') {
35
36 $errors = array(); // Initialize error array.
37
38 // Validate the email address:
39 if (empty($email)) {
40 $errors[] = 'You forgot to enter your email address.';
41 } else {
42 $e = mysqli_real_escape_string($dbc, trim($email));
43 }
44
45 // Validate the password:
46 if (empty($pass)) {
47 $errors[] = 'You forgot to enter your password.';
48 } else {
49 $p = mysqli_real_escape_string($dbc, trim($pass));
50 }
51
52 if (empty($errors)) { // If everything's OK.
53
54 // Retrieve the user_id and first_name for that email/password combination:
55 $q = "SELECT user_id, first_name FROM users WHERE email='$e' AND pass=SHA1('$p')";
56 $r = @mysqli_query ($dbc, $q);
57 // Run the query.
58
59 // Check the result:
60 if (mysqli_num_rows($r) == 1) {
61
62 // Fetch the record:
63 $row = mysqli_fetch_array ($r, MYSQLI_ASSOC);
64
65 // Return true and the record:
66 return array(true, $row);
67
68 } else { // Not a match!
69 $errors[] = 'The email address and password entered do not match those on file.';
70 }
71 } // End of empty($errors) IF.
72
73 // Return false and the errors:
74 return array(false, $errors);
75
76 } // End of check_login() function.

```

因为这个文件将由其他文件包含，所以它不必包含任何HTML。

(2) 开始定义新函数。

```
function redirect_user ($page = 'index.php') {
```

`redirect_user()`函数将返回一个绝对URL，它对于运行这些脚本的站点是正确的。动态执行该操作的好处是（与仅仅硬编码`http://www.example.com/page.php`相对）：可以在一个服务器（比如你自己的计算机）上开发代码，然后把它移到另一个服务器上，而无需更改这段代码。

该函数带有一个可选的参数：最终的目标页面名称。默认值是`index.php`。

(3) 开始定义URL。

```
$url = 'http://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['PHP_SELF']);
```

首先，把`http://`以及主机名称（它可以是`localhost`或`www.example.com`）赋予`$url`。这里使用`dirname()`函数添加了当前目录的名称，以免在子文件夹内发生重定向。`$_SERVER['PHP_SELF']`引用当前脚本（它将是调用这个函数的脚本），包括目录名称。整个值可能是`/somedir/page.php`。`dirname()`函数将只从那个值中返回目录部分（即`/somedir/`）。

(4) 从URL中删除任何结尾斜杠。

```
$url = rtrim($url, '/\\');
```

由于子文件夹可能添加额外的斜杠（/）或反斜杠（\，对于Windows），所以函数需要删除它们。为了执行该任务，可使用`rtrim()`函数。默认情况下，这个函数会删除字符串右边的空格。如果提供了要删除的字符列表作为第二个参数，就会代之以删除这些字符。这里要删除的字符应该是/或\。但是，由于PHP中的反斜杠是转义符，需要使用\\来指示单个反斜杠。因此，简而言之，如果`$url`以这些字符中的任何一个结尾，`rtrim()`函数将删除它们。

(5) 添加特定的页面到URL。

```
$url .= '/' . $page;
```

下面将特定的页面名称追加到`$url`中。在它前面放置一个斜杠，因为任何尾随的斜杠都会在第(4)步中被删除，并且不能把`www.example.compage.php`作为URL，然后返回该URL。

这些看起来似乎都相当复杂，但它可以非常有效地确保重定向正确工作，而不管在什么服务器上、来自于什么目录，或者脚本是否在运行（只要重定向发生在那个目录内即可）。

(6) 重定向用户并完成函数

```
header("Location: $url");
exit(); // Quit the script.
} // End of redirect_user() function.
```

最后一步是发送`Location`头，并结束脚本执行。

(7) 开始创建新函数。

```
function check_login($dbc, $email = '', $pass = '') {
```

这个函数将验证登录信息。它带有三个参数：数据库连接（它是必需的）、电子邮件地址（它是可选的）和密码（它也是可选的）。

尽管这个函数可以直接访问`$_POST['email']`和`$_POST['pass']`，但是更好的做法是给函数传递这些值，从而使得函数更独立。

(8) 验证电子邮件地址和密码。

```
$errors = array();
if (empty($email)) {
```

```

$errors[] = 'You forgot to enter your email address.';
} else {
 $e = mysqli_real_escape_string ($dbc, trim($email));
}
if (empty($pass)) {
 $errors[] = 'You forgot to enter your password.';
} else {
 $p = mysqli_real_escape_string ($dbc, trim($pass));
}

```

这个验证例程类似于注册页面中使用的那个验证例程。如果发生任何问题，都会把它们添加到\$errors数组中，最终将把它们用在登录页面上（参见图12-2）。注意\$errors数组对函数来说是局部的。即使它与登录页面使用的\$errors变量有相同的名字，但是它们并不相同。函数后面的代码中会返回这个\$errors变量的值，并且脚本中调用这个函数的代码会将返回的值赋给合适的全局变量\$errors数组，可以用于登录页面。

(9) 如果没有发生错误，就运行数据库查询。

```

if (empty($errors)) {
 $q = "SELECT user_id, first_name FROM users WHERE email='$e' AND pass=SHA1('$p')";
 $r = @mysqli_query ($dbc, $q);

```

这个查询从数据库中选择user\_id和first\_name值，其中提交的电子邮件地址（来自于表单）匹配存储的电子邮件地址，并且提交的密码的SHA1()版本匹配存储的密码（参见图12-4）。

The screenshot shows a terminal window titled 'PHP and MySQL for Dynamic Web Sites: ...'. It displays the following MySQL command and its result:

```

mysql> SELECT user_id, first_name FROM users
-> WHERE email='email@example.com' AND
-> pass=SHA1('mypass');
+-----+-----+
| user_id | first_name |
+-----+-----+
| 1 | Larry |
+-----+-----+
1 row in set (0.00 sec)

mysql>

```

图12-4 如果用户提交正确的电子邮件地址/密码组合，登录查询就会得到这个结果

(10) 检查查询的结果。

```

if (mysqli_num_rows($r) == 1) {
 $row = mysqli_fetch_array ($r, MYSQLI_ASSOC);
 return array(true, $row);
}

```

如果查询返回一行，那么登录信息是正确的。然后将把结果获取进\$row中。成功登录中的最后一步是把两种信息返回到请求的脚本中：值true和从MySQL获取的数据，其中前者指示登录成功。使用array()函数，可以返回布尔值和\$row数组。

(11) 如果没有查询到记录，创建错误信息。

```

} else { // Not a match!
 $errors[] = 'The email address and password entered do not match those on file.';
}

```

如果查询没有返回一行，就会把一条出错消息添加到数组中。它最终将显示在登录页面上（参见图12-5）。

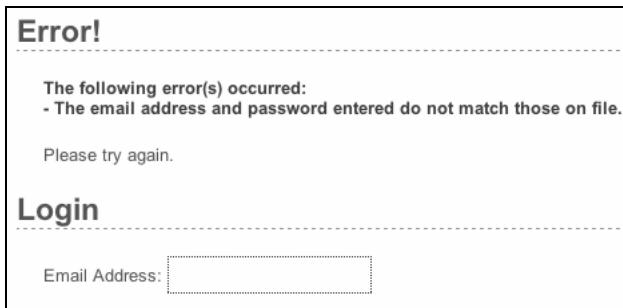


图12-5 如果用户输入了电子邮件地址和密码，但是它们与数据库中存储的值不匹配，就会得到这个结果

(12) 完成开始于第(9)步中的条件语句，并完成函数。

```

} // End of empty($errors) IF.
return array(false, $errors);
} // End of check_login() function.

```

该函数的最后一步是返回值`false`指示登录失败，以及返回`$errors`数组，它存储失败的原因。可以把这个`return`语句放在这里（函数的末尾，而不是条件语句内）因为如果登录失败，函数将只会执行到这里。如果登录成功，第(9)步中的`return`行将阻止函数继续执行（一旦函数执行`return`语句，它就会停止执行）。

(13) 将文件另存为`login_functions.inc.php`，并将其存放在Web目录中（该目录与`header.html`、`footer.html`和`style.css`这几个文件都在`includes`文件夹中）。

该页面也会使用`.inc.php`扩展名，指示它是一个可包含的文件并且它包含PHP代码。

就像本书创建的其他的可以被包含的文件一样（但不包括`login_page.inc.php`），PHP的闭合标签(`?>`)被省略了。这样做可以预防在包含文件的闭合标签后，由于无意添加的空格和空行带来的潜在问题。

#### ✓ 提示

- 本章中的脚本不包含任何调试代码（如MySQL错误或查询）。如果你有关于这些脚本的任何问题，可以应用第7章中概括的调试技术。
- 可以在`header()`调用中把`name=value`对添加到URL中，以把值传递到目标页面：

```
$url .= '?name=' . urlencode($value);
```

## 12.3 使用 cookie

cookie是服务器在用户的机器上存储信息的一种方式。利用这种方式，站点可以在访问期间记住或跟踪用户。可以把cookie看作像名字标签一样，你告诉服务器你的名字，它就会给你戴上一个标签。这样，它就可以回过头来查阅名字标签，从而知道你是谁（参见图12-6）。

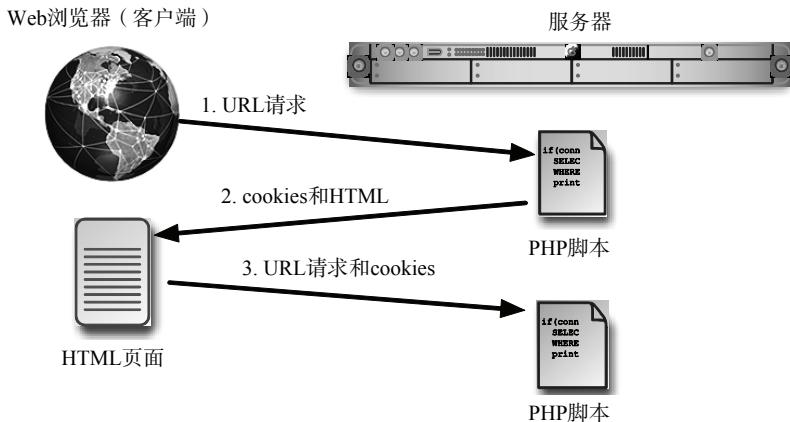


图12-6 cookie是如何在服务器和客户端之间传递的

在本节中，你将学习如何设置一个cookie，从存储的cookie中检索信息，改变cookie的设置以及删除一个cookie。

### 测试cookie

要使用cookie有效地编程，需要能够准确地测试它们是否存在。执行该任务的最佳方式是：当接收到一个cookie时，让Web浏览器询问应该做什么。在这种情况下，每当PHP试图发送一个cookie时，浏览器都会用cookie信息提示你。

不同平台上不同浏览器的不同版本都在不同的位置定义了它们的cookie处理策略。我将快速介绍两个针对流行的Web浏览器的选项。

要使用Windows XP上的Internet Explorer设置它，可选择“工具”→“Internet选项”。然后单击“隐私”选项卡，接着单击“设置”下面的“高级”按钮。单击选中“覆盖自动cookie处理”复选框，然后为“第一方Cookie”和“第三方Cookie”选择“提示”。

如果在Windows上使用Firefox，可选择“工具”→“选项”→“隐私”。在cookie区域中，在“保存到”下拉菜单中选择“每次询问我”。如果在Mac OS X上使用Firefox，其步骤是相同的，但是必须通过选择Firefox→“参数设置”来开始设置步骤。在“隐私”标签页选择“为历史记录使用自定义设置”，接下来你就会看到“每次询问我”选项了。

不幸的是，Safari上的Google Chrome没有cookie提示选项(至少在不安装扩展或插件的情况下)，但是，它允许查看现有的cookie，这仍然是一种有用的调试工具。

### 12.3.1 设置cookie

关于cookie要理解的最重要的事情是，必须在任何其他信息之前把它们从服务器发送给客户。万一服务器试图在Web浏览器已经接收到HTML(甚至是无关紧要的空白)之后发送cookie，就会导致一条出错消息，并且不会发送cookie(参见图12-7)。迄今为止，这是最常见的与cookie相关的错误，但是很容易修复它。

```
Warning: Cannot modify header information - headers already sent by (output started at
/Users/larryullman/Sites/phpmysql4/includes/login_functions.inc.php:80) in
/Users/larryullman/Sites/phpmysql4/login.php on line 22
```

图12-7 在创建cookie时，headers already sent...出错消息实在太常见。要注意出错消息说了些什么，以便找出并修复问题

通过setcookie()函数发送cookie：

```
setcookie ('name', value);
setcookie ('name', 'Nicole');
```

第二行代码在把cookie发送到浏览器时，带有名字name和值Nicole（参见图12-8）。



图12-8 如果把浏览器设置成在接收cookie时请求许可，则当站点试图发送一个cookie时，将看到一条如本图所示的消息（这是Firefox版本的提示）

通过接着使用setcookie()函数，可以继续把更多的cookie发送给浏览器：

```
setcookie ('ID', 263);
setcookie ('email', 'email@example.com');
```

与在PHP中使用任何变量时一样，在给cookie命名时，不要使用空格或标点符号，但是，要特别注意使用正确的大小写字母。

### 发送cookie

(1) 在文本编辑器中创建一个新的PHP文档，命名为login.php（参见脚本12-3）。

#### 脚本 12-3 login.php 脚本会在成功登录时创建两个 cookie

```
1 <?php # Script 12.3 - login.php
2 // This page processes the login form submission.
3 // Upon successful login, the user is redirected.
4 // Two included files are necessary.
5 // Send NOTHING to the Web browser prior to the setcookie() lines!
6
7 // Check if the form has been submitted:
8 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
9
10 // For processing the login:
```

```

11 require ('includes/login_functions.inc.php');
12
13 // Need the database connection:
14 require ('../mysqli_connect.php');
15
16 // Check the login:
17 list ($check, $data) = check_login($dbc, $_POST['email'], $_POST['pass']);
18
19 if ($check) { // OK!
20
21 // Set the cookies:
22 setcookie ('user_id' . $data['user_id']);
23 setcookie ('first_name', $data['first_name']);
24
25 // Redirect:
26 redirect_user('loggedin.php');
27
28 } else { // Unsuccessful!
29
30 // Assign $data to $errors for error reporting
31 // in the login_page.inc.php file.
32 $errors = $data;
33
34 }
35
36 mysqli_close($dbc); // Close the database connection.
37
38 } // End of the main submit conditional.
39
40 // Create the page:
41 include ('includes/login_page.inc.php');
42 ?>
```

对于这个示例，我将建立一个新的login.php脚本（它将与第9章中的脚本协同工作）。这个脚本还需要在本章开头创建的两个文件。

(2) 如果表单已提交，包含两个帮助文件。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 require ('includes/login_functions.inc.php');
 require ('../mysqli_connect.php');
```

这个脚本将做两件事：处理表单提交和显示表单。这个条件语句检查提交。

在这个条件语句内，脚本必须包含login\_functions.inc.php和mysqli\_connect.php（它是在第9章中创建的，并且仍然应该位于相对于这个脚本的相同位置，如果mysqli\_connect.php没有在父目录或当前目录中，需要修改相应代码）。

在两种情况下我都选择使用require()来代替inclued()，因为任何一个脚本的引用（inclued()）失败都会导致登录过程失败。

(3) 验证表单数据：

```
list ($check, $data) = check_login ($dbc, $_POST['email'], $_POST['pass']);
```

在包含这两个文件后，可以调用check\_login()函数，将数据库连接（它来自于mysqli\_connect.php）

以及电子邮件地址和密码（它们都来自于表单）传递给它。

该函数返回两个元素的数组：布尔值和另一个数组（其中包含用户数据或错误）。为了把这些返回的值赋予变量，可使用list()函数。该函数返回的第一个值（布尔值）将被赋予\$check，返回的第二个值（\$row或\$errors数组）将被赋予\$data。

(4) 如果用户输入了正确的信息，则登录。

```
if ($check) { // OK!
 setcookie ('user_id', $data['user_id']);
 setcookie ('first_name', $data['first_name']);
```

\$check变量指示登录尝试是否成功。如果它为true，那么\$data将包含用户的ID和名字。可以在cookie中使用这两个值。

通常来讲，不需要在cookie中存储数据库表中的主键值（比如\$data['user\_id']），这是因为cookie很容易被篡改。本例中情况特殊，因为user\_id值并没有在网站的任何地方使用（只为演示之用）。

(5) 把用户重定向到另一个页面。

```
redirect_user('loggedin.php');
```

使用本章前面定义的函数，在用户登录成功后会被重定向到别一个脚本，即重定向到loggedin.php页面。

(6) 完成\$check条件语句（开始于第(4)步），然后关闭数据库连接。

```
} else {
 $errors = $data;
}
mysqli_close($dbc);
```

如果\$check的值为false，那么\$data变量就会存储在check\_login()函数内生成的错误。如果是这样，就应该把它们赋予\$errors变量，因为这是脚本中显示登录页面（login\_page.inc.php）的代码所需要的。

(7) 完成主提交条件语句，并包含登录页面。

```
}
include ('includes/login_page.inc.php');
?>
```

这个login.php脚本主要验证登录表单，这是通过调用check\_login()函数完成的。login\_page.inc.php文件包含登录页面本身，因此只需要包含它即可。

(8) 将文件另存为login.php，存放在Web目录中（在与第9章中的文件相同的目录中），并在Web浏览器中加载这个页面（参见图12-2）。

你可以胡乱地提交表单，但目前还无法登录，因为最终的目标——loggedin.php还没有编写。

#### ✓ 提示

- cookie被限制为总共包含大约4 KB的数据，每个Web浏览器可以记住来自任何一个站点的有限数量的cookie。对于目前的大多数Web浏览器，这个限制是50个cookie（但是，如果发出50个不同的cookie，你可能想重新考虑如何执行该任务）。
- 因为不同的浏览器将以不同的方式处理cookie，PHP中有几个函数可以在不同的浏览器中生成

不同的结果, `setcookie()`函数是其中之一。一定要在不同平台上的多个浏览器中测试你的Web站点, 以确保一致性。

- 如果前两个包含文件向Web浏览器发送任何内容, 或者甚至在关闭PHP标签后面具有空白行或空格, 你将看到headers already sent错误。如果看到这种错误, 可以查看错误中提到的文档和行号(在output started at之后), 并修复问题。

### 12.3.2 访问cookie

要从cookie中检索一个值, 只需要把合适的cookie名称用作键来引用`$_COOKIE`超全局数组(就像利用任何数组一样)。例如, 为了从用如下一行代码建立的cookie中检索一个值:

```
setcookie ('username', 'Trout');
```

将引用`$_COOKIE['username']`。

在下面的示例中, 将以两种方式来访问`login.php`脚本设置的cookie。首先, 将检查用户是否已登录(如果未登录, 他们不应该访问这个页面)。接下来, 将通过用户名来问候他们, 他们的名字存储在一个cookie中。

#### 访问cookie

- (1) 在文本编辑器中创建一个新的PHP文档, 命名为`loggedin.php`(参见脚本12-4)。

#### 脚本12-4 `loggedin.php`脚本基于一个存储的cookie向用户打印一个问候

```

1 <?php # Script 12.4 - loggedin.php
2 // The user is redirected here from login.php.
3
4 // If no cookie is present, redirect the user:
5 if (!isset($_COOKIE['user_id'])) {
6
7 // Need the functions:
8 require ('includes/login_functions.inc.php');
9 redirect_user();
10
11 }
12
13 // Set the page title and include the HTML header:
14 $page_title = 'Logged In!';
15 include ('includes/header.html');
16
17 // Print a customized message:
18 echo "<h1>Logged In!</h1>
19 <p>You are now logged in, {$_COOKIE ['first_name']}!</p>
20 <p>Logout</p>";
21
22 include ('includes/footer.html');
23 ?>
```

在用户成功登录后, 将把他们重定向到这个页面。它将打印一个特定于用户的问候。

- (2) 检查cookie是否存在。

```
if (!isset($_COOKIE['user_id'])) {
```

因为不希望用户访问这个页面（除非那个用户已经登录），所以应该首先检查是否已经设置了cookie（在login.php中）。

(3) 如果用户未登录，就重定向他们。

```
require ('includes/login_functions.inc.php');
redirect_user();
}
```

如果用户没有登录，就会自动把他们重定向到这个主页面。这是一种限制访问内容的简单方式。

(4) 包含页面的标题。

```
$page_title = 'Logged In!';
include ('includes/header.html');
```

(5) 使用cookie欢迎用户。

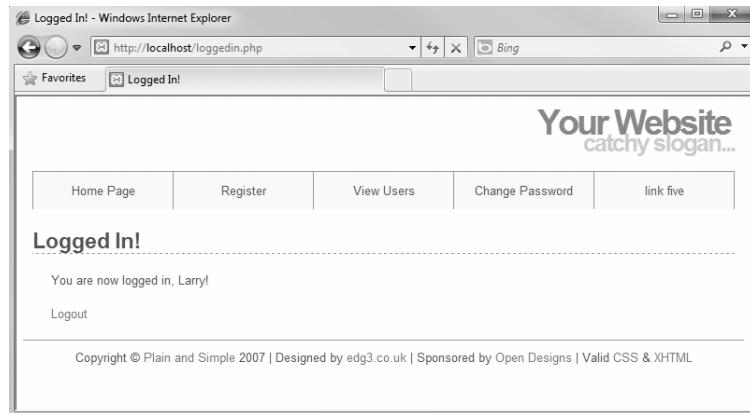
```
echo "<h1>Logged In!</h1>
<p>You are now logged in, {$_COOKIE['first_name']}!</p>
<p>Logout</p>";
```

为了用名字问候用户，引用了\$\_COOKIE['first\_name']变量（括在花括号内以避免解析错误）。还会打印一个指向注销页面（将在本章后面编写）的链接。

(6) 完成HTML页面。

```
include ('includes/footer.html');
?>
```

(7) 将文件另存为loggedin.php，存放在Web目录中（在与login.php相同的文件夹中），并通过login.php登录在Web浏览器中测试它（参见图12-9）。



12

图12-9 如果使用正确的电子邮件地址和密码，则在登录后，会重定向到这里

因为这些示例使用与第9章中相同的数据，你应该能够使用当时提交的注册用户名和密码来登录。

(8) 要查看设置的cookie（参见图12-10和图12-11），可以针对你的浏览器更改cookie设置并再次测

试它。

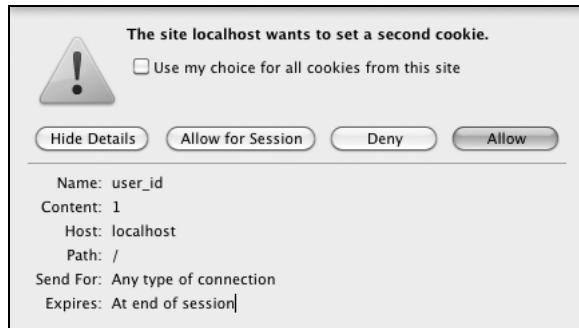


图12-10 user\_id cookie的值为1

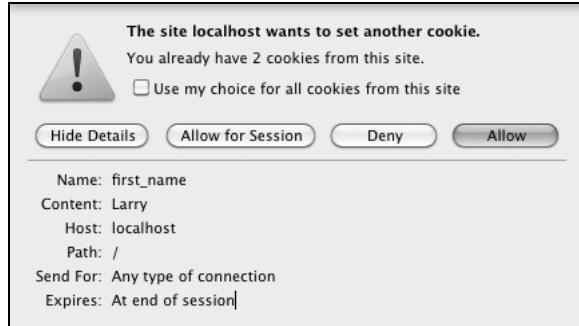


图12-11 first\_name cookie的值为Larry（你的可能有所不同）

### ✓ 提示

- 有些浏览器（例如，IE）对于从本地主机发送的cookies不会遵守你的cookie提示设置。
- 直到重新加载了设置页面（例如，login.php）或者访问了另一个页面（换句话说，你不能设置和访问同一个页面中的cookie），才可以访问cookie。
- 如果用户拒绝一个cookie，或者把他们的Web浏览器设置成不接受它们，则会把用户自动重定向到这个示例中的主页，即使他们成功登录也是如此。由于这个原因，你可能想让用户知道何时需要cookie。

### 12.3.3 设置cookie参数

尽管只把名称和值这两个参数传递给setcookie()函数就足够了，还应该知道其他可用的参数。该函数还可以带有最多另外5个参数，其中每一个都会改变cookie的定义。

`setcookie (name, value, expiration, path, host, secure, httponly);`

到期时间（expiration）参数用于设置cookie存在的精确限定的时间长度，用从新纪元（epoch）（新

纪元是1970年1月1日0点)起所经过的秒数表示。如果没有设置它或者把它的值设置为0, cookie将持续起作用, 直到用户关闭了他或她的浏览器。这些cookie被认为在浏览器会话期间一直存在(图12-10和图12-11也指示了这一点)。

为了设置特定的到期时间, 可在当前时刻上增加特定的分钟数或小时数, 可以使用time()函数获取当前时刻。下面一行代码将把cookie的到期时间设置为当前时间后30分钟(60秒乘以30分钟):

```
setcookie (name, value, time() + 1800);
```

路径(path)和主机(host)这两个参数用于把cookie限制到Web站点内(路径)特定的文件夹, 或者限制到特定的主机(www.example.com或192.168.0.1)。例如, 你可以把cookie限制为, 仅当用户位于域中的admin文件夹(以及admin文件夹的子文件夹)内时才存在。

```
setcookie (name, value, expire, '/admin/');
```

将路径设置为/将使得cookie在整个域(Web站点)内可见。将域设置为example.com将使得cookie在整个域和每个子域(www.example.com、admin.example.com、pages.example.com等)内都可见。

安全(secure)值规定只应该通过安全的HTTPS连接来发送cookie。1指示必须使用安全的连接, 0则指示标准连接也不错。

```
setcookie (name, value, expire, path, host, 1);
```

如果站点使用安全连接, 则限制通过安全的HTTPS连接发送cookie比不这样做要安全得多。

最后, PHP 5.2中添加了一个httponly参数。使用一个布尔值指示只能通过HTTP(和HTTPS)访问cookie。强制执行这种限制将使得cookie更安全(阻止一些黑客攻击), 但是在编写本书时并非所有的浏览器都支持它。

```
setcookie (name, value, expire, path, host, secure, TRUE);
```

与带有参数的所有函数一样, 必须按顺序传递setcookie()值。要跳过任何参数, 可使用NULL、0或空字符串(不要使用FALSE)。到期时间和安全这两个值都是整数, 因此不能用引号括住它们。

为了演示这个信息, 我将添加一个到期时间设置到登录cookie中, 使得它们只持续存在1个小时。

### 设置cookie的参数

(1) 在文本编辑器中打开login.php(参见脚本12-3)。

(2) 更改两个setcookie()行, 使之包括一个持续60分钟的到期日期(参见脚本12-5)。

#### 脚本 12-5 login.php 脚本现在使用了 setcookie() 函数可以带的每个参数

```

1 <?php # Script 12.5 - login.php #2
2 // This page processes the login form submission.
3 // The script now adds extra parameters to the setcookie() lines.
4
5 // Check if the form has been submitted:
6 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
7
8 // Need two helper files:
9 require ('includes/login_functions.inc.php');
10 require ('../mysqli_connect.php');
11
12 // Check the login:
```

```

13 list ($check, $data) = check_login ($dbc, $_POST['email'], $_POST['pass']);
14
15 if ($check) { // OK!
16
17 // Set the cookies:
18 setcookie ('user_id', $data ['user_id'], time() +3600, '/', '', 0, 0);
19 setcookie ('first_name', $data ['first_name'], time() +3600, '/', '', 0, 0);
20
21 // Redirect:
22 redirect_user('loggedin.php');
23
24 } else { // Unsuccessful!
25
26 // Assign $data to $errors for login_page.inc.php:
27 $errors = $data;
28
29 }
30
31 mysqli_close($dbc); // Close the database connection.
32
33 } // End of the main submit conditional.
34
35 // Create the page:
36 include ('includes/login_page.inc.php');
37 ?>

```

通过把到期日期设置成`time() + 3600` ( 60分钟乘以60秒 ), 在设置cookie后, 它将持续存在1个小时。在执行这种改变时, 显式指定其他cookie参数。

对于最后一个参数, 它接受一个布尔值, 也可以使用0表示`false` ( PHP将为你处理这种转换 )。这样做是一个好主意, 因为在任何cookie参数中都使用`false`可能引发问题。

(3) 保存这个脚本, 存放在Web目录中, 并通过登录在Web浏览器中测试它 ( 参见图12-12 )。

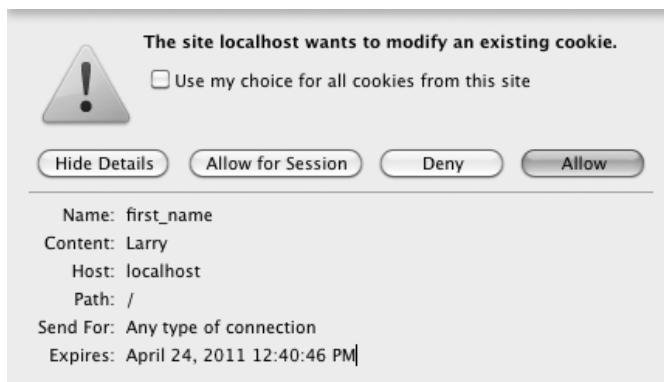


图12-12 对`setcookie()`参数 ( 比如到期日期和时间 ) 所做的更改将会反映在发送给Web浏览器的cookie中 ( 对比图12-11 )

### ✓ 提示

- 有些浏览器难以处理没有列出每个参数的cookie。显式指出每个参数，甚至是一个空字符串，对于所有浏览器将得到更可靠的结果。
- 对于cookie的到期时间有一些一般性的指导原则：如果cookie应该持续存在与会话一样长的时间，就不要设置到期时间；如果cookie在用户关闭并重新打开他或她的浏览器之后应该继续存在，就要把到期时间向前设置几周或几个月的时间；如果cookie可能构成安全风险，就要把到期时间设置成自此开始1个小时或不到1个小时的时间，以使得在用户离开他或她的浏览器之后，cookie不会继续存在太长的时间。
- 出于安全性考虑，可以在cookie上设置一个5分钟或10分钟的到期时间，并与用户访问的每个新页面一起重发cookie（假定cookie存在）。这样，cookie持续存在的时间就与用户活动的时间一样长，但它会在用户最后一个动作的5分钟或10分钟后自动死亡。
- 电子商务以及其他与隐私相关的Web应用程序都应该为所有事务（包括cookie）使用一条SSL（Secure Sockets Layer，安全套接字层）连接。
- 要小心对待由目录内的脚本创建的cookie。如果没有指定路径，那么该cookie将只能由同一个目录内的其他脚本使用。

#### 12.3.4 删除cookie

关于使用cookie要理解的最后一件事是如何删除一个cookie。虽然在关闭用户的浏览器时或者在到达到期日期/时间时cookie会自动到期，但是，有时你希望手动删除cookie。例如，在具有登录能力的Web站点内，可能希望在用户注销时删除任何cookie。

尽管setcookie()函数可以带最多7个参数，但是实际上只有一个参数（cookie名称）是必需的。如果发送一个包含名称但没有值的cookie，其效果就相当于删除现有的同名cookie。例如，要创建first\_name cookie，可以使用下面这一行代码：

```
setcookie('first_name', 'Tyler');
```

要删除first\_name cookie，可以编写如下代码：

```
setcookie('first_name');
```

一种附加的预防措施是，还可以把到期日期设置成过去的某个日期。

```
setcookie('first_name', '', time()-3600);
```

为了演示所有这些操作，将给站点添加注销能力。指向注销页面的链接将出现在loggedin.php上。作为一种附加特性，将更改头文件，使得当用户登录时显示Logout链接，并且当用户注销时显示Login链接。

##### 1. 删除cookie

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为logout.php（参见脚本12-6）。

##### 脚本 12-6 logout.php 脚本会删除以前建立的 cookie

```
1 <?php # Script 12.6 - logout.php
2 // This page lets the user logout.
```

```

3
4 // If no cookie is present, redirect the user:
5 if (!isset($_COOKIE['user_id'])) {
6
7 // Need the function:
8 require ('includes/login_functions.inc.php');
9 redirect_user();
10 } else { // Delete the cookies:
11 setcookie ('user_id', '', time()-3600, '/', '', 0, 0);
12 setcookie ('first_name', '', time()-3600, '/', '', 0, 0);
13 }
14
15
16 // Set the page title and include the HTML header:
17 $page_title = 'Logged Out!';
18 include ('includes/header.html');
19
20 // Print a customized message:
21 echo "<h1>Logged Out!</h1>
22 <p>You are now logged out, {$_COOKIE
23 ['first_name']}!</p>";
24 include ('includes/footer.html');
25 ?>
```

(2) 检查user\_id cookie是否存在；如果它不存在，就重定向用户。

```

if (!isset($_COOKIE['user_id'])) {
 require ('includes/login_functions.inc.php');
 redirect_user();
```

与loggedin.php页面一样，如果用户还没有登录，这个页面就应该把用户重定向到主页。尝试注销尚未登录的用户没有意义。

(3) 如果cookie存在，则删除它们。

```

} else {
 setcookie ('user_id', '', time()-3600, '/', '', 0, 0);
 setcookie ('first_name', '', time()-3600, '/', '', 0, 0);
}
```

如果用户已经登录，这两个cookie将有效地删除现有的cookie。除了值和到期时间外，其他参数应该具有与创建cookie时相同的值。

(4) 建立PHP页面的提醒。

```

$page_title = 'Logged Out!';
include ('includes/header.html');
echo "<h1>Logged Out!</h1>
<p>You are now logged out, {$_COOKIE['first_name']}!</p>";
include ('includes/footer.html');
?>
```

这个页面本身也与loggedin.php页面非常相似。尽管它看起来似乎有些奇怪，但是因为仍然可以引用first\_name cookie（刚才在这个脚本中删除了它），考虑以下过程很有意义：

- (a) 这个页面将被客户请求。
- (b) 服务器从客户的浏览器读取合适的cookie。
- (c) 运行页面，并做它的事情（包括发送新的cookie，删除现有的cookie）。

因此，简而言之，在第一次运行这个脚本时，它就可以使用原来的first\_name cookie数据。这个页面发送的cookie集（删除的cookie）不能为这个页面所用，因此原来的值仍然有用。

- (5) 将文件另存为logout.php，并存放在Web目录中（在与login.php相同的文件夹中）。

## 2. 创建注销链接

- (1) 在文本编辑器或IDE中打开header.html（参见脚本9-1）。
- (2) 将第五个和最后一个链接更改如下（参见脚本12-7）：

### 脚本 12-7 根据用户的当前状态，header.html 文件现在将显示登录或注销链接

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
2 xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5 <title><?php echo $page_title; ?> </title>
6 <link rel="stylesheet" href="includes/style.css" type="text/css" media="screen" />
7 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
8 </head>
9 <body>
10 <div id="header">
11 <h1>Your Website</h1>
12 <h2>catchy slogan...</h2>
13 </div>
14 <div id="navigation">
15
16 Home Page
17 Register
18 View Users
19 <?php // Create a login/logout link:>
20 if ((isset($_COOKIE['user_id'])) && (basename($_SERVER['PHP_SELF']) != 'logout.php')) {
21 echo 'Logout ';
22 } else {
23 echo 'Login';
24 }
25 ?>
26
27 </div>
28 <div id="content"><!-- Start of the page-specific content. -->
29 <!-- Script 12.7 - header.html -->
```

12

不是在导航区域中建立一个永久的登录链接，而是在用户没有登录时，让其显示一个Login链接（参见图12-13），或者在用户登录时，则显示一个Logout链接（参见图12-14）。根据cookie存在与否来完成上面的条件语句。

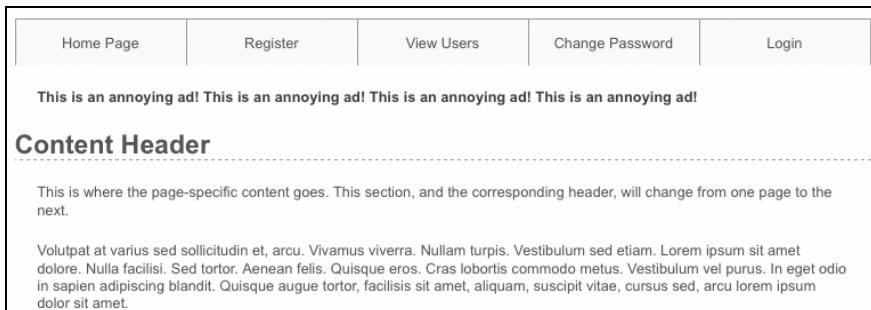


图12-13 带有Login链接的主页



图12-14 在用户登录后，页面现在具有一个Logout链接

在这种情况下，如果已经设置了cookie，用户已经登录并可以显示注销链接。如果没有设置cookie，需要向用户显示登录链接。然而，还有一个问题：因为logout.php脚本通常会显示一个注销链接（因为cookie当页面第一次查看时是存在的），条件语句同样必须检查当前的页面不是logout.php脚本。一个简单的动态确定当前页面的方法是对\$\_SERVER['PHP\_SELF']应用basename()函数。

(3) 保存文件，将其存放在Web目录中（放置在includes文件夹内），并在Web浏览器中测试登录/注销过程（参见图12-15）。



图12-15 注销后的结果

### ✓ 提示

- ❑ 要查看logout.php脚本中setcookie()调用的结果，可以在浏览器中打开cookie提示（参见图12-16）。
- ❑ 由于在Windows上的Internet Explorer处理cookie的方法中存在一个错误，可能需要把host参数设置成false（不带引号），以便在自己的计算机上开发应用程序时，可以让注销过程工作（即通过localhost）。

- 在删除一个cookie时，应该总是使用与设置cookie相同的参数。如果在创建cookie时设置了主机和路径，则在删除cookie中要再次使用它们。
- 为了讲清楚要点，记住，直到重新加载页面或者访问了另一个页面时，cookie的删除才会生效。换句话说，在那个页面删除了cookie之后，它仍然可供页面使用。

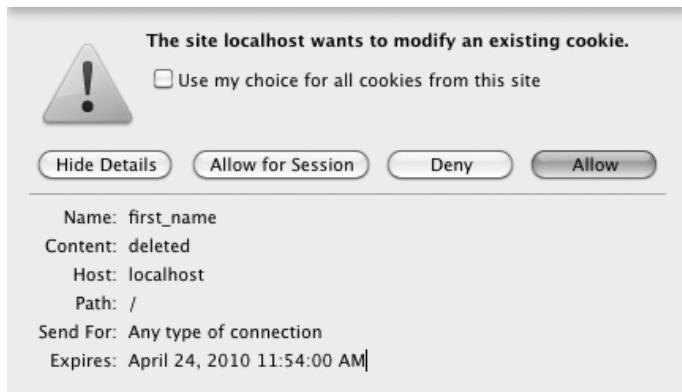


图12-16 这就是在Firefox提示中显示删除cookie的方式（对比图12-12）

## 12.4 使用会话

使数据可供Web站点上的多个页面使用的另一种方法是使用会话（session）。会话假定数据存储在服务器上，而不是在Web浏览器中，会话标识符用于定位特定用户的记录（会话数据）。这个会话标识符通常通过cookie存储在用户的Web浏览器中，但是，敏感数据本身（如用户的ID、姓名等）总是保留在服务器上。

有人可能会问：本来cookie工作得好好的，为什么还要使用会话？首先，会话更安全，这是由于所有记录的信息都存储在服务器上，并且不会持续不断地在服务器和客户之间来回发送。其次，可以在会话中存储更多的数据。最后，有些用户拒绝cookie，或者完全关闭它们。虽然会话被设计成与cookie一起工作，但它也可以独立起作用。

为了演示会话，并把它们与cookie作比较，我将重写以前的脚本集。

12

### 会话与cookie

本章具有一些示例，它们使用cookie和会话来完成相同的任务（登录和注销）。显然，它们都能很容易地在PHP中使用，但是，真正的问题是什么时候使用哪一个更合适。

与cookie相比，会话具有以下优点：

- 一般更安全（因为数据保存在服务器上）；
- 允许存储更多数据；
- 使用会话时，可以不使用cookie。

与会话相比，cookie则具有以下优点：

- 更容易编程；
- 需要更少的服务器资源；
- 通常情况下能够持续更长时间。

一般而言，要存储和检索少量的信息，则可使用cookie。不过，对于大部分Web应用程序，都会使用会话。

### 12.4.1 设置会话变量

关于会话的最重要的规则是，将会使用它们的每个页面首先都必须调用`session_start()`函数。这个函数告诉PHP开启一个新的会话，或者访问一个现有的会话。必须在把任何内容发送到Web浏览器之前调用这个函数。

第一次使用`session_start()`函数时，它会试图发送一个cookie，名称为`PHPSESSID`（会话名称），和一个类似于`a61f8670baa8e90a30c878df89a2074b`（32个十六进制字母，它是会话ID）的值。由于试图发送一个cookie，所以在把任何数据发送到Web浏览器之前，必须先调用`session_start()`，这与使用`setcookie()`和`header()`这两个函数时的情况一样。

一旦启动了会话，就可以使用正常的数组语法把值注册到会话中：

```
$_SESSION['key'] = value;
$_SESSION['name'] = 'Roxanne';
$_SESSION['id'] = 48;
```

记住这一点后，让我们更新`login.php`脚本。

#### 开启会话

- (1) 在文本编辑器或IDE中打开`login.php`（参见脚本12-5）。
- (2) 用以下几行代码替换`setcookie()`那几行代码（第18~19行）（参见脚本12-8）。

#### 脚本 12-8 login.php 脚本现在使用的是会话，而不是 cookie

```
1 <?php # Script 12.8 - login.php #3
2 // This page processes the login form submission.
3 // The script now uses sessions.
4
5 // Check if the form has been submitted:
6 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
7
8 // Need two helper files:
9 require ('includes/login_functions.inc.php');
10 require ('../mysqli_connect.php');
11
12 // Check the login:
13 list ($check, $data) = check_login($dbc, $_POST['email'], $_POST['pass']);
14
15 if ($check) { // OK!
16
17 // Set the session data:
18 session_start();
```

```

19 $_SESSION['user_id'] = $data['user_id'];
20 $_SESSION['first_name'] = $data['first_name'];
21
22 // Redirect:
23 redirect_user('loggedin.php');
24
25 } else { // Unsuccessful!
26
27 // Assign $data to $errors for login_page.inc.php:
28 $errors = $data;
29
30 }
31
32 mysqli_close($dbc); // Close the database connection.
33
34 } // End of the main submit conditional.
35
36 // Create the page:
37 include ('includes/login_page.inc.php');
38 ?>

```

第一步是开启会话。因为在脚本中这一刻之前没有echo()语句、HTML文件包含，或者甚至是空白，所以现在可以安全地使用session\_start()（尽管也可以把它放在脚本的顶部）。然后，把两个键-值（key-value）对添加到\$\_SESSION超全局数组中，以把用户名和用户ID注册到会话中。

(3) 将页面另存为login.php，存放在Web目录中，并在Web浏览器中测试它（参见图12-17）。

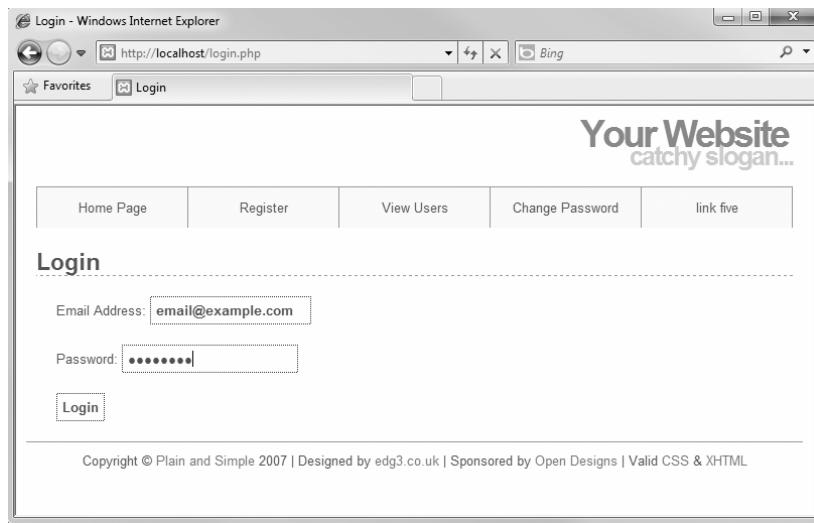


图12-17 登录表单对最终用户保持不变，但是底层的功能现在使用的是会话

尽管需要重写loggedin.php以及头部和脚本，你仍然可以测试登录脚本，并查看得到的cookie（参见图12-18）。不过，loggedin.php页面应该把你重定向回主页，因为它仍会检查\$\_COOKIE变量存在与否。

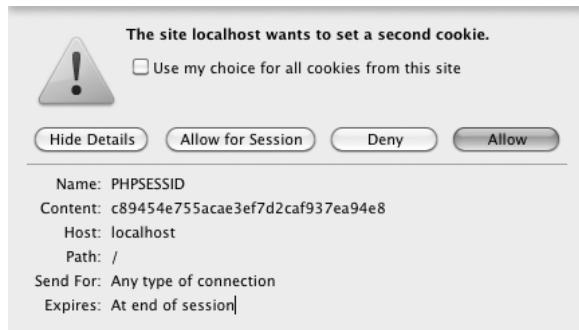


图12-18 由PHP的session\_start()函数创建的这个cookie存储会话ID

### ✓ 提示

- 由于会话通常会发送和读取cookie，应该总是设法在脚本中尽可能早地开启它们。这样做将有助于避免如下问题：试图在已经发送头部（HTML或空白）之后发送cookie。
- 如果你愿意，可以在php.ini文件中把session.auto\_start设置为1，从而不必在每个页面上使用session\_start()。这样做会加大服务器上的开销，由于这个原因，如果不考虑某些环境因素，就不应该使用它。
- 可以在会话中存储数组（使\$\_SESSION成为一个多维数组），就像可以存储字符串或数字一样。

## 12.4.2 访问会话变量

一旦启动了会话，并向其注册了变量，就可以创建将访问这些变量的其他脚本。为了执行该操作，每个脚本首先必须再次使用session\_start()来启用会话。

这个函数将允许当前脚本访问以前启动的会话（如果它可以读取cookie中存储的PHPSESSID值），或者创建一个新的会话（如果它不能读取这个值）。要理解如果不能找到当前会话ID或者生成了新的会话ID，那么在旧会话ID下存储的所有数据都将不可用。我之所以现在在这里提到这一点，是因为如果你有关于会话的问题，那么第一个调试步骤是检查会话ID值，看看它是否因页面而异。

假定访问当前会话没有问题，要引用一个会话变量，可使用\$\_SESSION['var']，就像你引用任何其他数组一样。

### 访问会话变量

- (1) 在文本编辑器或IDE中打开loggedin.php（参见脚本12-4）。
- (2) 添加对session\_start()函数的调用（参见脚本12-9）。

**脚本 12-9** 我更新了loggedin.php脚本，使得它可以引用\$\_SESSION，而不是引用\$\_COOKIE（需要在两行上执行更改）

```

1 <?php # Script 12.9 - loggedin.php #2
2 // The user is redirected here from login.php.
3
4 session_start(); // Start the session.

```

```

5 // If no session value is present, redirect the user:
6 if (!isset($_SESSION['user_id'])) {
7
8 // Need the functions:
9 require ('includes/login_functions.inc.php');
10 redirect_user();
11
12 }
13
14
15 // Set the page title and include the HTML header:
16 $page_title = 'Logged In!';
17 include ('includes/header.html');
18
19 // Print a customized message:
20 echo "<h1>Logged In!</h1>
21 <p>You are now logged in, {$_SESSION['first_name']}!</p>
22 <p>Logout</p>";
23
24 include ('includes/footer.html');
25 ?>
```

设置或访问会话变量的每个PHP脚本都必须使用`session_start()`函数。必须在包含`header.html`文件之前以及在把任何内容发送到Web浏览器之前调用这一行代码。

(3) 用`$_SESSION`替换对`$_COOKIE`的引用 (原文件第5行和第19行)。

```
if (!isset($_SESSION['user_id'])) {
 和
 echo "<h1>Logged In!</h1>
 <p>You are now logged in, {$_SESSION['first_name']}!</p>
 <p>Logout</p>";
```

把一个脚本从cookie转换成会话只需要把使用的`$_COOKIE`更改成`$_SESSION` (假定使用相同的名称)。

(4) 将文件另存为`loggedin.php`, 存放在Web目录中, 并在浏览器中测试它 (参见图12-19)。

(5) 在`header.html`中用`$_SESSION`替换对`$_COOKIE`的引用 (从脚本12-7到脚本12-10)。

```
if (isset($_SESSION['user_id'])) {
```

12



图12-19 在登录后, 将会把用户重定向到`loggedin.php`, 它会使用存储的会话值通过名字来欢迎用户

## 脚本12-10 header.html文件现在也会引用\$\_SESSION,而不是引用\$\_COOKIE

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/
2 xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5 <title><?php echo $page_title; ?> </title>
6 <link rel="stylesheet" href="includes/style.css" type="text/css" media="screen" />
7 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
8 </head>
9 <body>
10 <div id="header">
11 <h1>Your Website</h1>
12 <h2>catchy slogan...</h2>
13 </div>
14 <div id="navigation">
15
16 Home Page
17 Register
18 View Users
19 Change Password
20 <?php // Create a login/logout link:
21 if (isset($_SESSION['user_id'])) {
22 echo 'Logout';
23 } else {
24 echo 'Login';
25 }
26
27
28 </div>
29 <div id="content"><!-- Start of the page-specific content. -->
29 <!-- Script 12.10 - header.html -->
```

为了让Login/Logout链接正确地工作(注意图12-17中不正确的链接),必须把头文件内对cookie变量的引用转变成会话。头文件不需要调用session\_start()函数,因为它会被调用该函数的页面包括在内。

(6)保存头文件,存放在Web目录中(在includes文件夹中),并在浏览器中测试它(参见图12-20)。



图12-20 利用为会话进行过更改的头文件,将会显示正确的Login/Logout链接(对比图12-17)

### ✓ 提示

- 为了让Login/Logout链接在其他页面（register.php、index.php等）上也能正确工作，将需要在其中的每一个页面上添加session\_start()命令。
- 回忆一下我曾经说过的，如果你具有一个应用程序，它看起来似乎不能从一个页面到另一个页面访问会话数据，这可能是由于新的会话是在每个页面上创建的。为了检查这一点，可以比较会话ID（值的最后几个字符就足够了）来查看它是否相同。你可以在发送会话cookie时通过查看会话cookie来查看会话的ID，或者使用session\_id()函数来执行该操作：
 

```
echo session_id();
```
- 一旦建立了会话变量，就可以使用它们。因此，与使用cookie时不同，可以把一个值赋予\$\_SESSION['var']，然后在同一个脚本的后面引用\$\_SESSION['var']。

### 垃圾收集

关于会话的垃圾收集是指删除会话文件（其中存储了实际的数据）的过程。创建一个销毁会话的注销系统是理想的，但是，并不能保证所有用户都按应该做的那样正式注销。所以PHP包含了一个清理进程。

无论何时调用session\_start()函数，都会引入PHP的垃圾收集，用于检查每个会话最近的修改日期（每当设置或获取变量时都会修改会话）。有两个设置规定了垃圾收集，它们是：session.gc\_maxlifetime和session.gc\_probability。第一个设置用于指定在一个会话持续多少秒不活动之后，可将其看作空闲会话，从而删除。第二个设置确定执行垃圾收集的概率，其取值范围为1~100。因此，默认设置是，对session\_start()的每个调用都有1%的机会调用垃圾收集。如果PHP没有启动清理进程，则会删除任何超过1440秒未使用的会话。

你可以使用ini\_set()函数改变这些设置，尽管要小心谨慎地执行该操作。频率过高或概率太大的垃圾收集可能使服务器陷入困境，并且会不经意地结束较慢用户的会话。

12

### 12.4.3 删除会话变量

在使用会话时，需要创建一个方法来删除会话数据。在当前示例中，当用户注销时，这是必要的。

虽然cookie系统只需要发送另一个cookie来销毁现有的cookie，但是会话的要求更高，因为既要考虑客户上的cookie，又要考虑服务器上的数据。

要删除单独一个会话变量，可以使用unset()函数（它可以处理PHP中的任何变量）：

```
unset($_SESSION['var']);
```

要删除每个会话变量，可以重置整个\$\_SESSION数组：

```
$_SESSION = array();
```

最后，要从服务器中删除所有的会话数据，可使用session\_destroy()：

```
session_destroy();
```

注意，在使用其中的任何一个方法之前，页面都必须开始于session\_start()，使得会话访问现有的会话。让我们更新logout.php脚本，以清理会话数据。

### 删除会话

- (1) 在文本编辑器或IDE中打开logout.php ( 参见脚本12-6 )。
- (2) 紧接在开始PHP行之后启动会话 ( 参见脚本12-11 )。

**脚本 12-11 销毁会话** (就像在注销页面中所做的那样) 需要使用特殊的语法, 以删除服务器上的会话 cookie 和会话数据, 以及清理\$\_SESSION数组

```

1 <?php # Script 12.11 - logout.php #2
2 // This page lets the user logout.
3 // This version uses sessions.
4
5 session_start(); // Access the existing session.
6
7 // If no session variable exists, redirect the user:
8 if (!isset($_SESSION['user_id'])) {
9
10 // Need the functions:
11 require ('includes/login_functions.inc.php');
12 redirect_user();
13
14 } else { // Cancel the session:
15
16 $_SESSION = array(); // Clear the variables.
17 session_destroy(); // Destroy the session itself.
18 setcookie ('PHPSESSID', '', time()-3600, '/', '', 0, 0); // Destroy the cookie.
19
20 }
21
22 // Set the page title and include the HTML header:
23 $page_title = 'Logged Out!';
24 include ('includes/header.html');
25
26 // Print a customized message:
27 echo "<h1>Logged Out!</h1>
28 <p>You are now logged out!</p>";
29
30 include ('includes/footer.html');
31 ?>
```

无论何时使用会话, 都必须使用session\_start()函数, 最好是在页面顶部这样做, 即使删除会话也是如此。

- (3) 更改条件语句, 使得它检查会话变量是否存在。

```
if (!isset($_SESSION['user_id'])) {
```

与cookie示例中的logout.php脚本一样, 如果用户目前没有登录, 就会重定向他们。

- (4) 用下面的几行代码替换setcookie()行 ( 它删除cookie )。

```
$_SESSION = array();
session_destroy();
setcookie ('PHPSESSID', '', time()-3600, '/', '', 0, 0);
```

这里的第一行代码将把整个\$\_SESSION变量重置为一个新数组, 从而清除其现有的值。第二行代码

从服务器中删除数据，第三行代码则会发送一个cookie，用以替换浏览器中现有的会话cookie。

(5) 删除消息中对\$\_COOKIE的引用。

```
echo "<h1>Logged Out!</h1>
<p>You are now logged out!</p>";
```

与使用logout.php脚本的cookie版本时不同，不能再通过用户的名字来引用他们，因为所有这类数据都已被删除。

(6) 将文件另存为logout.php，存放在Web目录中，并在浏览器中测试它（参见图12-21）。



图12-21 注销页面（现在具有会话）

### ✓ 提示

- header.html文件只需要检查\$\_SESSION['user\_id']是否已经设置，或者没有设置如果页面是登出页面，因为现在头部文件被logout.php所包含，所有的会话(session)数据将已经被销毁了。会话的数据会被立即销毁，不像cookies那样。
- 永远不要把\$\_SESSION设置成等于NULL并且永远不要使用unset(\$\_SESSION)，因为它们都可能会在某些服务器上引发问题。
- 以免你对接下来将要发生的事情不是绝对清楚，提供了关于会话的三类信息：会话标识符（它默认存储在cookie中）、会话数据（它存储在服务器上的一个文本文件中）和\$\_SESSION数组（它指定了脚本访问文本文件中的会话数据的方式）。仅仅删除cookie不会删除文本文件，反之亦然。清理\$\_SESSION数组将会清除文本文件中的数据，但是文件本身将仍然存在，cookie也是如此。这个注销脚本中概括的三个步骤实际上会删除会话的所有痕迹。

12

### 更改会话行为

作为PHP对会话所提供支持的一部分，可以为PHP处理会话的方式设置大约20种不同的配置选项。有关完整列表，参见PHP手册，但是我将在这里重点介绍几个最重要的选项。注意关于更改会话设置的两条规则：

- (1) 所有更改都必须在调用session\_start()之前完成。
- (2) 必须在使用会话的每个页面执行相同的更改。

可以在PHP脚本内使用ini\_set()函数（在第7章中讨论过）设置大多数选项：

```
ini_set (parameter, new_setting);
```

例如，如果需要使用会话cookie（如前所述，在不使用cookie的情况下会话也可以工作，但是这样不太安全），则可使用：

```
ini_set ('session.use_only_cookies', 1);
```

还可以更改会话的名称（也许是是为了使用一个更为用户友好的名称），这时可以使用 `session_name()` 函数。

```
session_name('YourSession');
```

创建自己的会话名称有双重好处：它更安全一点，并且可以更好地被最终用户接收（因为会话名称是最终用户将使用的cookie名称）。在删除会话cookie时，也可以使用 `session_name()` 函数：

```
setcookie(session_name(), '', time()-3600);
```

最后，还有一个 `session_set_cookie_params()` 函数。它用于调整会话cookie的设置。

```
session_set_cookie_params(expire, path, host, secure, httponly);
```

注意：cookie的到期时间只指cookie在Web浏览器中的寿命，而不是指会话数据将在服务器上存储多长时间。

## 12.5 提高会话安全性

由于重要的信息通常存储在会话中（永远都不应该把敏感数据存储在cookie中），所以安全性变得更重要。关于会话需要注意以下两项：会话ID和会话数据本身，前者是一个指向会话数据的引用，后者存储在服务器上。一个不怀好意的人极有可能通过会话ID（而不是通过服务器上的数据）来入侵一个会话，因此，在这里将重点关注这方面的事情（在本节末尾的“提示”中，我提到了保护会话数据的两种方式）。

会话ID是会话数据的关键。默认情况下，PHP将其存储在cookie中，从安全的角度讲，这是首选的方法。在PHP中可以在不使用cookie的情况下使用会话，但是这会使应用程序遭受会话劫持（`sessionhijacking`）：如果我可以获悉另一个用户的会话ID，就可以轻松地欺骗服务器把它看作是我的会话ID。此时，我就有效地接管了原用户的整个会话，并且可以访问他们的数据。因此，把会话ID存储在cookie中使得它更难被窃取。

防止劫持的一种方法是：在会话中存储某种用户标识符，然后反复地复查这个值。`HTTP_USER_AGENT`（所用的浏览器和操作系统的组合）是针对此目的的一个可能的候选。这会增加一层安全性，因为仅当我运行的浏览器和操作系统与另一位用户的完全一样时，才能够劫持他的会话。为了演示这一点，让我们最后一次修改示例。

### 防止会话固定

另一种特定的会话攻击被称为会话固定（`session fixation`）。其中，一位不怀好意的用户指定了另一位用户应该使用的会话ID。这个会话ID可以是随机生成的，或者是合法创建的。在这两种情况下，真实的用户都会使用固定的会话ID进入站点，并做任何事情。然后，那位不怀好意的用户可以访问那个会话，因为他们知道会话ID是什么。你可以在用户登录后通过更改会话ID来帮助防止这些类型的攻击。`session_regenerate_id()` 正是用于执行该任务的，它提供一个新的会话ID来引用当前的会话数据。如果站点的安全性极为重要（如电子商务或在线银行业务），或者如果用户的会话被操纵情况就变得特别糟糕时，就可以使用这个函数。

### 更安全地使用会话

- (1) 在文本编辑器或IDE中打开login.php (参见脚本12-8)。
- (2) 在给其他会话变量赋值之后, 还存储HTTP\_USER\_AGENT值 (参见脚本12-12)。

**脚本 12-12 login.php 脚本的这个最终版本也在会话中存储了用户的 HTTP\_USER\_AGENT (客户  
的浏览器和操作系统) 的加密形式**

```

1 <?php # Script 12.12 - login.php #
2 // This page processes the login form submission.
3 // The script now stores the HTTP_USER_AGENT value for added security.
4
5 // Check if the form has been submitted:
6 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
7
8 // Need two helper files:
9 require ('includes/login_functions.inc.php');
10 require ('../mysqli_connect.php');
11
12 // Check the login:
13 list ($check, $data) = check_login($dbc, $_POST['email'], $_POST['pass']);
14
15 if ($check) { // OK!
16
17 // Set the session data:
18 session_start();
19 $_SESSION['user_id'] = $data['user_id'];
20 $_SESSION['first_name'] = $data['first_name'];
21
22 // Store the HTTP_USER_AGENT:
23 $_SESSION['agent'] = md5($_SERVER['HTTP_USER_AGENT']);
24
25 // Redirect:
26 redirect_user('loggedin.php');
27
28 } else { // Unsuccessful!
29
30 // Assign $data to $errors for login_page.inc.php:
31 $errors = $data;
32
33 }
34
35 mysqli_close($dbc); // Close the database connection.
36
37 } // End of the main submit conditional.
38
39 // Create the page:
40 include ('includes/login_page.inc.php');
41 ?>
```

HTTP\_USER\_AGENT是\$\_SERVER数组的一部分 (你可以返回到第1章, 回忆一下使用它的方式)。它将具有一个像Mozilla/4.0这样的值 (与之兼容的值、MSIE 8.0、Windows NT 6.1等)。

为了提高安全性, 这里没有把这个值存储在会话中, 而是用md5()函数处理它。该函数基于一个

值返回32个字符的十六进制字符串，称为散列(hash)。理论上讲，任何两个字符串都不具有相同的md5()结果。

(3) 保存文件，存放在Web目录中。

(4) 在文本编辑器或IDE中打开loggedin.php(参见脚本12-9)。

(5) 将!isset(\$\_SESSION['user\_id'])条件语句更改如下(参见脚本12-13)：

**脚本 12-13** 这个loggedin.php脚本现在确认访问这个页面的用户具有与他们登录时相同的HTTP\_USER\_AGENT

```

1 <?php # Script 12.13 - loggedin.php #3
2 // The user is redirected here from login.php.
3
4 session_start(); // Start the session.
5
6 // If no session value is present, redirect the user:
7 // Also validate the HTTP_USER_AGENT!
8 if (!isset($_SESSION['agent']) OR ($_SESSION['agent'] != md5($_SERVER ['HTTP_USER_AGENT']))) {
9
10 // Need the functions:
11 require ('includes/login_functions.inc.php');
12 redirect_user();
13
14 }
15
16 // Set the page title and include the HTML header:
17 $page_title = 'Logged In!';
18 include ('includes/header.html');
19
20 // Print a customized message:
21 echo "<h1>Logged In!</h1>
22 <p>You are now logged in, {$_SESSION
23 ['first_name']}!</p>
24 <p>Logout</p>";
25 include ('includes/footer.html');
26 ?>
```

这个条件语句用于检查两件事情。首先，它会查看\$\_SESSION['agent']变量是否未设置(这一部分就像它以前一样，尽管使用的是agent而不是user\_id)。这个条件语句的第二部分用于检查\$\_SERVER ['HTTP\_USER\_AGENT']的md5()版本是否不等于\$\_SESSION ['agent']中存储的值。如果这两个条件中有一个为真，就会重定向用户。

(6) 保存这个文件，存放在Web目录中，并通过登录在Web浏览器中测试它。

### ✓ 提示

- 对于至关重要的会话应用，只要有可能，就需要使用cookie并通过安全连接传输它们。你甚至可以通过把session.use\_only\_cookies设置成1，将PHP设置成只使用cookie。
- 如果你使用与其他域共享的服务器，那么把session.save\_path从其默认设置(可以被所有用户访问)更改成稍微更本地化一些将会更安全。

- 会话数据本身可以存储在数据库中，而不是文本文件中。这样更安全，但这是一个编程密集的选项。我在*PHP 5 Advanced: Visual QuickPro Guide*一书中讲述了如何执行该任务。
- 用户的IP地址（用户通过其建立连接的网络地址）不是一个良好的唯一标识符，这有两个原因。  
首先，用户的IP地址可能（并且通常会）频繁地发生变化（ISP在短时间内动态地分配它们）。  
其次，从同一个网络（如家庭网络或办公室）访问一个站点的许多用户可能都具有相同的IP地址。

## 12.6 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 12.6.1 回顾

- 用来将用户的浏览器从一个页面重定向到另外一个的代码是？
- `headers already sent`信息的意思是？`$ _SERVER ["HTTP_HOST"]`存储的是什么值？`$ _SERVER ["PHP_SELF"]`存储的是什么值？
- `dirname()`函数的作用是什么？
- `rtrim()`函数的作用是什么？它可以带有哪些参数？
- 该如何编写一个返回多个值的函数？如何调用这样的函数？
- `setcookie()`函数可以带什么参数？
- 你该如何引用以前存储在cookie中的值？
- 如何删除现有的一个cookie？
- cookies是否在被发送后立即生效（在同一页上）？为什么cookie被删除后仍然可以引用它（在同一页上）？
- 当cookies出现问题的时候可以采取哪些调试步骤？
- `basename()`函数的作用是什么？
- 如何开启一个会话？
- 如何引用会话中存储的值？
- 会话数据是否在被分配后立即可用（在同一页上）？
- 如何终止一个会话？
- 当会话出现问题时可以采取哪些调试步骤？

12

### 12.6.2 实践

- 学习如何在浏览器中查看cookie数据。当使用cookie开发站点时，启用cookie选项，这样当接收到cookies时浏览器会有提示。
- 使登录表单变为一个黏性表单。
- 为登录页的`$errors`变量添加处理代码，当`$errors`是数组时使用`foreach`循环来处理它，否则就直接打印`$errors`的值。

- 修改**redirect\_user()**函数，使它可以将用户重定向到另一个目录的页面。
- 实现一个cookie的例子，例如在cookie中存储用户的喜好，然后使外观或风格遵循这个存储的值（当存在时）。
- 更改**logout.php**（脚本12-11）的代码，使用**session\_name()**函数动态设置正在删除的会话的cookie名称。
- 如果你想做更多关于会话的练习，就实现一个新的会话的示例（本书的后面也会做更多的练习）。
- 在PHP手册中查看本章介绍的你还不太熟悉的新函数。
- 检查PHP手册中的cookie和会话页面（两个独立的部分）以了解更多信息。再看看一些用户提交的评论中的其他提示。

## 本章内容

- 阻止垃圾邮件
- 通过类型验证数据
- 按类型验证文件
- 阻止XSS攻击
- 使用过滤器扩展
- 预防SQL注入攻击
- 回顾和实践

Web应用程序的安全性是一个如此重要的主题，以至于确实需要重点强调它。尽管全书一直在提及与安全性相关的问题，本章仍有助于填补某些空白，并得出关于其他要点的定论。

这里讨论的主题包括：阻止垃圾邮件、使用强制类型转换、阻止XSS（cross-site scripting，跨站脚本）攻击和SQL注入攻击、新的过滤器（Filter）扩展，以及验证上传文件类型。本章将使用5个具体的示例最好地阐释这些概念，还将在框注中提及其他一些常见的安全问题和最佳实践。

## 13.1 阻止垃圾邮件

垃圾邮件极为讨厌，它们使Internet和我们的收件箱混乱不堪。你可以采取一些措施来避免在电子邮件账户中接收到垃圾邮件，但是，在本书中重点关注的是阻止通过你的PHP脚本发送垃圾邮件。

第11章显示了使用PHP的`mail()`函数发送电子邮件有多容易，其中的示例（联系人表单）从用户那里获取一些信息（参见图13-1），并把它们发送到一个电子邮件地址。尽管它看起来似乎不会在这个系统中造成损害，实际上还是有一个巨大的安全漏洞。但是，首先将介绍关于电子邮件实际上是什么的一些背景知识。

不管怎样发送电子邮件，怎样对其进行格式化以及接收到它时的样子是什么，电子邮件都包含两部分：头部和正文。头部包括如下内容：收件人和发件人地址、主题、日期等（参见图13-2）。头部中的每一项都在它自己的行上，并且都具有名称：值（Name: value）的格式。电子邮件的正文正好就是你想要发送的内容。

在查看PHP的`mail()`函数（`mail (to, subject, body, [headers]);`）时，你能看到其中一个参数就是电子邮件的正文，余下的参数则出现在它的头部中。要把垃圾邮件发送到你的地址（比如在第11

章的示例中那样),人们只需把垃圾邮件消息输入到表单的注释区域中即可(参见图13-1)。这已经很糟糕了,但是要把垃圾邮件同时发送给其他人,用户只需向电子邮件的头部中添加Bcc: poorsap@example.org,其后接着某种行终止符(如换行符或回车符)。对于这个示例,这仅仅意味着输入联系人表单的from值me@example.com\nBcc:poorsap@example.org。

**Contact Me**

Please fill out this form to contact me.

Name: Not Larry Ullman

Email Address: email@example.com

Comments:

Your book isn't just the greatest computer book ever written, it's the greatest book ever written, period! It totally blows "One Hundred Years of Solitude" out of the water.

Send!

图13-1 简单、标准的HTML联系人表单

```
X-Spam-Checker-Version: SpamAssassin 3.2.4 (2008-01-01) on vps.larryullman.net
X-Spam-Level:
X-Spam-Status: No, score=-0.0 required=7.0 tests=NO_RELAYS autolearn=ham
version=3.2.4
Received: (qmail 13517 invoked by uid 48); 19 Apr 2011 11:31:42 -0400
Date: 19 Apr 2011 11:31:42 -0400
Message-ID: <20110419153142.13515.qmail@vps.larryullman.net>
To: Larry@LarryUllman.com
Subject: Contact Form Submission
From: email@example.com

Name: Not Larry Ullman

Comments: Your book isn't just the greatest computer book ever
written, it's the greatest book ever written, period! It totally blows
"One Hundred Years of Solitude" out of the water.
```

图13-2 通过联系人表单(图13-1)发送的电子邮件的原始版本

你可能认为保护进入电子邮件头部中的一切内容就足够安全了,但是由于电子邮件只是一个文档,正文中错误的输入可能会影响头部。

有两种预防技术。第一,使用正则表达式(第14章)或过滤器扩展(13.5节)验证任何电子邮件地址。第二,既然你知道作恶者必须输入什么内容来发送垃圾邮件(参见表13-1),就可以在表单值中监视这些字符。如果值中包含该列表中的任何内容,就不要使用那个值。(最后四项全都是创建换行符的不同方式)。

表13-1 垃圾邮件暗示

字 符	字 符
Strings	cc:
content-type:	to:
mime-version:	\r
multipart-mixed:	\n
content-transfer-encoding:	%0a
bcc:	%0d

在下一个示例中，将修改第11章中的电子邮件脚本。我将定义一个函数，它将从数据中清除所有潜在危险的字符。还将使用两个新的PHP函数：`str_replace()`和`array_map()`。在下面的步骤中将详细解释它们。

### 安全方法

关于安全性要理解的最重要的概念是，不能用安全或者不安全来形容你的Web站点或脚本。安全性不是一个可以打开和关闭的开关，而是表示可以提升和降低的尺度。在编程时，考虑做什么可以使站点更安全，以及做了什么会使站点不太安全。此外，记住：提高安全性通常会以方便性（对于程序员和最终用户而言）和性能为代价。增强安全性通常意味着更多的代码、更多的检查以及对服务器的更多要求。在开发Web应用程序时，要从一开始就考虑到这些事项并作出正确的决策——针对特定的情形。而过于“安全”的网站也会出问题，只要谨慎小心即可。

### 阻止垃圾邮件

(1) 在文本编辑器或IDE中打开`email.php`（参见脚本11-1）。

要完成这个垃圾邮件清理操作，需要修改电子邮件脚本。

(2) 在检查表单提交后，开始定义函数（参见脚本13-1）。

这个函数只需要一个字符串型参数。我通常习惯在脚本顶端或另外单独的文件中定义函数，但为简单起见，将它放在处理提交的代码块中。

**脚本 13-1** 脚本的这个版本现在可以安全地发送电子邮件，而无需顾虑垃圾邮件。任何有疑问的字符串都会被`spam_scrubber()`函数捕获

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Contact Me</title>
6 </head>
7 <body>
8 <h1>Contact Me</h1>
9 <?php # Script 13.1 - email.php #2
10 // This version now scrubs dangerous strings from the submitted input.
11
12 // Check for form submission:

```

```
13 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
14
15 /* The function takes one argument: a string.
16 * The function returns a clean version of the string.
17 * The clean version may be either an empty string or
18 * just the removal of all newline characters.
19 */
20 function spam_scrubber($value) {
21
22 // List of very bad values:
23 $very_bad = array('to:', 'cc:', 'bcc:', 'content-type:', 'mime-version:', 'multipart-
24 // mixed:', 'content-transfer-encoding:');
25
26 // If any of the very bad strings are in
27 // the submitted value, return an empty string:
28 foreach ($very_bad as $v) {
29 if (stripos($value, $v) !== false) return '';
30 }
31
32 // Replace any newline characters with spaces:
33 $value = str_replace(array("\r", "\n", "%0a", "%0d"), ' ', $value);
34
35 // Return the value:
36 return trim($value);
37 } // End of spam_scrubber() function.
38
39 // Clean the form data:
40 $scrubbed = array_map ('spam_scrubber', $_POST);
41
42 // Minimal form validation:
43 if (!empty($scrubbed['name']) && !empty($scrubbed['email']) && !empty($scrubbed
44 ['comments'])) {
45
46 // Create the body:
47 $body = "Name: {$scrubbed['name']}\n\nComments: {$scrubbed['comments']}";
48
49 // Make it no longer than 70 characters long:
50 $body = wordwrap($body, 70);
51
52 mail('your_email@example.com', 'Contact Form Submission', $body, "From: {$scrubbed
53 ['email']}");
54
55 // Print a message:
56 echo '<p>Thank you for contacting me. I will reply some day.</p>';
57
58 // Clear $_POST (so that the form's not sticky):
59 $_POST = array();
60
61 } else {
62 echo '<p style="font-weight: bold; color: #C00">Please fill out the form completely.</p>';
63 }
64 } // End of main isset() IF.
```

(3) 创建不会出现在合法联系人表单提交中的确实糟糕的字符串的列表。

```
$very_bad = array('to:', 'cc:', 'bcc:', 'content-type:', 'mime-version:', 'multipart-mixed:', 'content-transfer-encoding:');
```

所有这些字符串都不应该存在于诚实的联系人表单提交中（可能有人在他们的注释中合法地使用to:，但是未必会如此）。如果所有这些字符串都存在，那么这就意味着企图发送垃圾邮件。为了能够更容易地测试所有这些字符串，把它们置于一个数组中，然后将遍历该数组（第(4)步）。第(4)步中的测试是不区分大小写的，所有危险字符串都由小写字母组成。

(4) 遍历数组。如果找到非常糟糕的内容，返回一个空字符串。

```
foreach ($very_bad as $v) {
 if (stripos($value, $v) !== false) return '';
}
```

这个foreach循环一次将访问\$very\_bad中的一个数据项，`stripos()`函数将检查作为\$value提供给该函数的数据项是否在字符串中。`stripos()`函数执行不区分大小写的查找（因此，它将匹配bcc:、Bcc:、bCC:等）。如果发现了不良内容，函数返回布尔值TRUE（在\$value中出现了\$v）。条件语句用来在函数不等于FALSE（在\$value中没有出现\$v）时返回一个空字符串。

因此，如果在第一次提交时发现“危险”字符串，函数将返回一个空字符串并终止运行（一旦函数遇到`return`，它们就会自动停止执行）。

(5) 用空格替换所有的换行符。

```
$value = str_replace(array("\r", "\n", "%0a", "%0d"), ' ', $value);
```

换行符（用\r、\n、%0a和%0d表示）可能有问题，也可能没有问题。在发送垃圾邮件时需要换行符（否则就不能创建正确的头部），但是，如果用户在文本区中输入内容时刚好敲击了Enter键，那么也会出现换行符。因此，任何找到的换行符都将由空格替换。这意味着提交的值可能会丢失它的一些格式化效果，但是这是阻止垃圾邮件所要付出的一种合理的代价。

`str_replace()`函数检查第三个参数中的值，并用第二个参数中的字符替换第一个参数中出现的任何字符。或者，就像PHP手册中所做的那样：

```
mixed str_replace (mixed $search, mixed $replace, mixed $subject)
```

这个函数非常灵活，这是由于它可以接受字符串或数组作为它的三个参数（mixed意味着它可以接受混合的参数类型）。因此，脚本中的这一行代码将给\$value变量赋予其原始值，并用单个空格替换任何换行符。

这个函数有一个不区分大小写的版本，但这不是必要的，例如，\r是回车符，但是\R则不然。

(6) 返回值并完成函数。

```
 return trim($value);
} // End of spam_scrubber() function.
```

最后，该函数返回值，并用任何前导和末尾空格对其进行修饰。记住，如果没有发现非常糟糕的内容，该函数只会执行到这里。

(7) 在函数定义后面，调用`spam_scrubber()`函数。

```
$scrubbed = array_map('spam_scrubber', $_POST);
```

这么简单的一行代码就搞定了，真漂亮！`array_map()`函数有两个必需的参数。第一个是要调用的函数的名称，在这里是`spam_scrubber`（不带括号，因为提供的是函数的名称，而不是调用函数）。第二个参数是一个数组。

`array_map()`所做的工作是每次为一个数组元素调用指定的函数，并把每个数组元素的值发送给该函数。在这个脚本中，`$_POST`具有4个元素：`name`、`email`、`comments`、`submit`也就是说调用4次`spam_scrubber()`函数，这受益于`array_map()`。在这一行代码后面，将用4个元素结束`$scrubbed`数组：`$scrubbed['name']`，在对它运行`spam_scrubber()`之后，它将具有值`$_POST['name']`；`$scrubbed['email']`，在对它运行`spam_scrubber()`之后，它将具有与`$_POST['email']`相同的值，等等。

这一行代码然后获取可能包含垃圾数据的完整数组（`$_POST`），使用`spam_scrubber()`清理它，并把结果赋予新变量。这里最重要的事情是：从此开始，脚本将使用干净的`$scrubbed`数组，而不是`$_POST`，它仍有可能包含垃圾数据。

(8) 更改表单验证，以使用这个新数组。

```
if (!empty($scrubbed['name']) && !empty($scrubbed['email']) && !empty($scrubbed['comments'])) {
```

由于以下两个原因，所有这些元素都可能具有一个空值。第一，用户保持它们为空。第二，用户在文本框中输入了糟糕的内容（参见图13-3），从而被`spam_scrubber()`函数转变成空字符串（参见图13-4）。

**Contact Me**

Please fill out this form to contact me.

Name: Not Larry Ullman

Email Address: Larry@LarryUllman.com\r\ncc:

This is spam!!!

Comments:

Send!

图13-3 电子邮件地址文本框中的cc:将阻止在电子邮件中发送这种提交（参见图13-4）

**Contact Me**

Please fill out the form completely.

Please fill out this form to contact me.

Name: Not Larry Ullman

Email Address:

This is spam!!!

Comments:

Send!

图13-4 由于在电子邮件地址中使用了非常糟糕的字符，因此不会发送电子邮件

(9) 更改\$body变量的创建方式，使之使用干净的值。

```
$body = "Name: {$scrubbed['name']}\\n\\nComments: {$scrubbed['comments']}";
```

(10) 更改mail()函数的调用方式，以使用干净的电子邮件地址。

```
mail('your_email@example.com', 'Contact Form Submission', $body, "From: {$scrubbed['email']}");
```

注意，mail()调用中使用你自己的电子邮件地址，否则你无法收到任何邮件。

(11) 修改表单，使其使用\$scrubbed的值：

```
<p>Name: <input type="text" name="name" size="30" maxlength="60" value="<?php if (isset($scrubbed['name'])) echo $scrubbed['name']; ?>" /></p>
<p>Email Address: <input type="text" name="email" size="30" maxlength="80" value="<?php if (isset($scrubbed['email'])) echo $scrubbed['email']; ?>" /></p>
<p>Comments: <textarea name="comments" rows="5" cols="30"><?php if (isset ($scrubbed['comments'])) echo $scrubbed['comments']; ?> </textarea></p>
```

(12) 将脚本另存为email.php，存放在Web目录中，并在Web浏览器中测试它（参见图13-5和图13-6）。

**Contact Me**

Please fill out the form completely.

Please fill out this form to contact me.

Name: Not Larry Ullman

Email Address: email@example.com

This is not spam.  
But the comments are going  
over multiple lines.

Comments:

Send!

图13-5 尽管注释文本框中包含换行符（通过按下Enter键或Return键创建），仍然会发送电子邮件（对比图13-6）

13

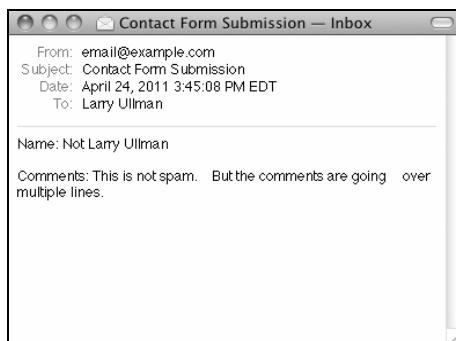


图13-6 接收到的电子邮件，它把注释中的换行符（对比图13-5）转变成空格

### ✓ 提示

- 像我在这个示例中所做的那样，使用`array_map()`函数很方便，但它也有一些缺点。第一，它对整个`$_POST`数组盲目地应用`spam_scrubber()`函数，甚至对提交按钮和隐藏的表单输入框也是如此。这样做不会造成损害，但却是不必要的。第二，`$_POST`内的任何多维数组都将丢失。在这个特定的示例中，这不是一个问题，但是要知道这一点。
- 为了阻止自动提交到任何表单，可以使用CAPTCHA测试。有一些提示只能被人类理解（理论上是这样）。虽然通常使用随机字符的映象完成这个任务，但也可以使用一个问题来完成相同的事情，比如“2加2等于几”或者“中国位于哪个洲”。这样，检查这个问题的正确答案将是验证例程的一部分。

## 13.2 通过类型验证数据

极大程度上，迄今为止本书中使用的表单验证都相当简单，通常只是检查一个变量是否具有值。在许多情况下，这确实是你所能够采纳的最佳方式。例如，对于有效的街道地址是什么或者用户可能向注释框中输入什么，并没有完美的验证方法。尽管如此，对于你将使用的大多数数据，都可以用更严格的方式对其进行验证。在下一章中，一个高级概念（即正则表达式）将说明这点。但是，我将在这里介绍一些更友好的方式，可以使用它们通过类型验证一些数据。

### 两种验证方法

安全在很大程度上基于验证：如果数据来自于脚本外部——来自于HTML表单、URL、cookie、会话，或者甚至来自于数据库，就不能信任它。有两种类型的验证：白名单（whitelist）和黑名单（blacklist）。在计算器示例中，我们知道所有的值都必须为正，它们都必须是数字，并且数量必须是一个整数（另外两个数字可能是整数或浮点数，这没有什么区别）。类型强制转换强制输入必须是数字，并且执行检查以确认它们为正。此时，假定输入是有效的。这是白名单方法：这些值都很好，其他任何值都很糟糕。

阻止垃圾邮件这个示例使用的是黑名单方法。那个脚本准确知道哪些字符很糟糕，并且使包含它们的输入框无效。所有其他的输入框都很好。

许多安全专家首选白名单方法，但是并非总是可以使用它。示例将指示哪种方法将工作得最好，但是重要的是使用其中一种方法。不要在没有执行某种验证的情况下就假定数据是安全的。

PHP支持多种类型的数据：字符串、数字（整型和浮点型）、数组等。对于其中的每一种类型，都有一个特定的函数，用于检查某个变量是否是那种类型（参见表13-2）。在前面的章节中，你可能已经看到过`is_numeric()`函数的应用。如果试图在`foreach`循环中使用一个变量，在此之前可以使用`is_array()`极好地确认它的类型。如果提交的变量是某种类型，则这些函数会返回`TRUE`；否则，它们会返回`FALSE`。

表13-2 类型验证函数

函 数	用于检查
<code>is_array()</code>	数组
<code>is_bool()</code>	布尔值 ( TRUE、FALSE )
<code>is_float()</code>	浮点数
<code>is_int()</code>	整数
<code>is_null()</code>	NULL
<code>is_numeric()</code>	数值, 甚至是字符串 (例如, '20' )
<code>is_resource()</code>	资源, 如数据库连接
<code>is_scalar()</code>	标量 (单值) 变量
<code>is_string()</code>	字符串

在PHP中, 你甚至可以在给变量赋值之后更改它的类型。这称为类型强制转换 ( typecasting ), 执行该任务的方法是, 在变量名前放置一个圆括号, 并在圆括号中指定变量的类型:

```
$var = 20.2;
echo (int) $var; // 20
```

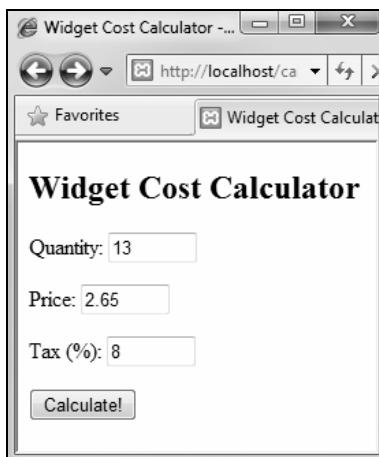
根据原始类型和目标类型, PHP将相应地转换变量的值:

```
$var = 20;
echo (float) $var; // 20.0
```

对于数值, 这种转换很直观, 但是对于其他变量类型, 则会应用更复杂的规则:

```
$var = 'trout';
echo (int) $var; // 0
```

在大多数环境中, 不需要把变量从一种类型强制转换为另一种类型, 因为PHP通常会根据需要自动执行该操作。但是, 强行地转换变量的类型可能是Web应用程序中一种良好的安全措施。为了说明如何使用这个概念, 将创建一个计算器脚本, 用于确定一个项目的总采购价格 (参见图13-7)。



13

图13-7 这个HTML表单带有3个输入框: 数量、价格和税率

### 使用类型强制转换

(1) 在文本编辑器或IDE中开始创建一个新的PHP文档，命名为calculator.php（参见脚本13-2）。

#### 脚本 13-2 通过强制转换变量类型，这个脚本更明确地验证数据具有正确的格式

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
2 DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Widget Cost Calculator</title>
6 </head>
7 <body>
8 <?php # Script 13.2 - calculator.php
9 // This script calculates an order total based upon three form values.
10
11 // Check if the form has been submitted:
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13
14 // Cast all the variables to a specific type:
15 $quantity = (int) $_POST['quantity'];
16 $price = (float) $_POST['price'];
17 $tax = (float) $_POST['tax'];
18
19 // All variables should be positive!
20 if (($quantity > 0) && ($price > 0) && ($tax > 0)) {
21
22 // Calculate the total:
23 $total = $quantity * $price;
24 $total += $total * ($tax/100);
25
26 // Print the result:
27 echo '<p>The total cost of purchasing ' . $quantity . ' widget(s) at $' . number_format($price,
28 2) . ' each, plus tax, is $' . number_format($total, 2) . '.</p>';
29
30 } else { // Invalid submitted values.
31 echo '<p style="font-weight: bold; color: #C00">Please enter a valid quantity, price, and
32 tax rate.</p>';
33 }
34
35 } // End of main isset() IF.
36
37 // Leave the PHP section and create the HTML form.
38 ?>
39 <h2>Widget Cost Calculator</h2>
40 <form action="calculator.php" method="post">
41 <p>Quantity: <input type="text" name="quantity" size="5" maxlength="10" value="<?php if
42 (isset($quantity)) echo $quantity; ?>" /></p>
43 <p>Price: <input type="text" name="price" size="5" maxlength="10" value="<?php if (isset
44 ($price)) echo $price; ?>" /></p>
45 <p>Tax (%): <input type="text" name="tax" size="5" maxlength="10" value="<?php if (isset
46 ($tax)) echo $tax; ?>" /></p>
47 <p><input type="submit" name="submit" value="Calculate!" /></p>

```

```
43 </form>
44 </body>
45 </html>
```

(2) 检查表单是否已提交。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

与许多以前的示例一样，这个脚本会同时显示HTML表单并处理其提交。

(3) 把所有变量强制转换为特定的类型。

```
$quantity = (int) $_POST['quantity'];
$price = (float) $_POST['price'];
$tax = (float) $_POST['tax'];
```

这个表单本身具有3个文本框(参见图13-7)，实际上可以向其中输入任何内容(HTML4或HTML1.1表单没有数字类型的输入框)。但是，数量必须是一个整数，价格和税率都应该是浮点数(它们将包含小数点)。为了解决这个问题，将把每个变量强制转换为特定的类型。

(4) 检查变量是否具有正确的值。

```
if (($quantity > 0) && ($price > 0) && ($tax > 0)) {
```

为了让这个计算器工作，这三个变量都必须具有特定的类型(参见第(3)步)。更重要的是，它们必须都是正数。这个条件语句将在执行计算之前检查这一点。注意：根据类型强制转换的规则，如果发布的值不是数字，将把它们强制转换为0，因此不会通过这个条件语句的检查。

(5) 计算并打印结果。

```
$total = $quantity * $price;
$total += $total * ($tax/100);
echo '<p>The total cost of purchasing ' . $quantity . ' widget(s) at $' . number_format($price, 2) .
 ' each, plus tax, is $' . number_format($total, 2) . '</p>';
```

为了计算总价格，用数量乘以价格。然后将税加到总价格中，用总价格乘以税率除以100(因此65%变成0.65)，再加到总价格中，使用加赋值短路运算。`number_format()`函数用于以正确的格式打印价格和总额这两个数值(参见图13-8)。

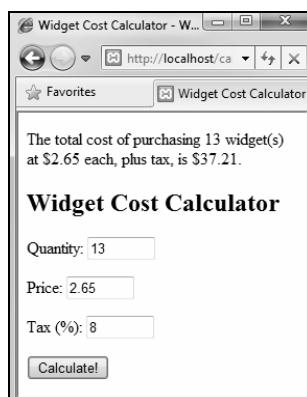


图13-8 如果表单被正确提交，将显示计算结果

(6) 完成条件语句。

```

} else {
 echo '<p style="font-weight: bold; color: #C00">Please enter a valid quantity, price, and tax
 rate.</p>';
}
} // End of main isset() IF.

```

万一有问题，这里使用了一点CSS来创建加粗、红色的出错消息（参见图13-9）。

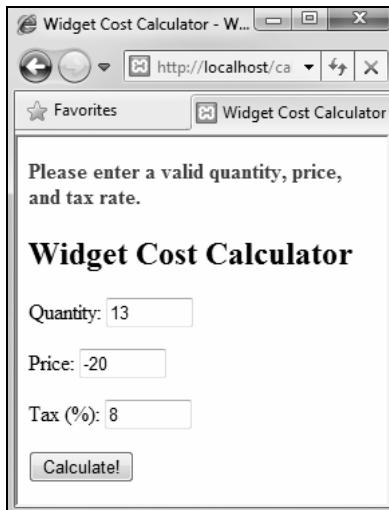


图13-9 如果三个文本框中的任何一个未包含正数，则会以加粗、红色的文本打印出错消息

(7) 开始创建HTML表单。

```

?>
<h2>Widget Cost Calculator</h2>
<form action="calculator.php" method="post">
 <p>Quantity: <input type="text" name="quantity" size="5" maxlength="10" value=<?php if (isset
 ($quantity)) echo $quantity; ?>" /></p>

```

这个HTML表单相当简单，并将发送回这个相同的页面。输入框将具有黏性，因此，用户可以看到以前输入的内容。通过引用\$quantity等来代替\$\_POST['quantity']等，表单将反映每个输入框的值，因为对它们的类型进行了强制转换。

(8) 完成HTML表单。

```

 <p>Price: <input type="text" name="price" size="5" maxlength="10" value=<?php if (isset($price))
 echo $price; ?>" /></p>
 <p>Tax (%): <input type="text" name="tax" size="5" maxlength="10" value=<?php if (isset($tax)) echo
 $tax; ?>" /></p>
 <p><input type="submit" name="submit" value="Calculate!" /></p>
</form>

```

(9) 完成HTML页面。

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

(10) 将文件另存为calculator.php，存放在Web目录中，并在Web浏览器中测试它（参见图13-10和图13-11）。

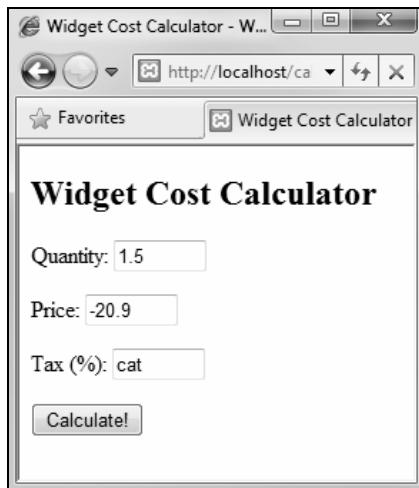


图13-10 如果输入了无效的值，比如为数量输入了浮点数，或者为税率输入了字符串……

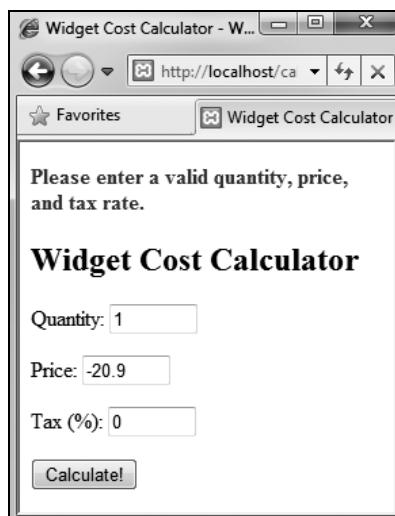


图13-11 ……则将把它们强制转换为更合适的格式。如果价格为负，也会阻止执行计算（尽管强制转换不会更改那个值）

### ✓ 提示

- 在SQL查询内处理数字时，肯定应该使用类型强制转换。数字在查询中没有用引号括住，因此，如果在应该使用数字的位置使用了字符串，则会产生一个SQL语法错误。如果首先把这些变量的类型强制转换为整型或浮点型，查询可能不会工作（就返回一条记录而言），但其语法仍然有效。在本书最后3章中，经常会看到这种情况。
- 如我所暗示的，正则表达式是一种更高级的数据验证方法，并且有时是你最佳的选择。但是，只要切实可行，使用基于类型的验证肯定会更快一些（就处理器速度而言），并且程序员出错的概率较小。（我提过正则表达式是一种复杂的方法吗？）
- 我不自觉地重申一遍，把值从一种数据类型转换成另一种数据类型的规则稍微有点复杂。如果你想深入了解其细节，见PHP手册。

## 13.3 按类型验证文件

第11章包括了一个在PHP处理文件上传的例子。由于上传文件允许用户在服务器上放置一个更有效的内容类型（相较于仅仅通过表单发送的文本），在处理它们的时候怎么注意安全都不为过。在第11章的那个例子中，上传的文件通过检查它的MIME类型进行验证。具体来说，上传的文件`$_FILES['upload']['type']`是指上传浏览器提供的MIME类型。这是一个好的开始，但恶意用户很容易通过提供虚假的MIME类型诱骗浏览器。更可靠地确定文件类型的方式是使用Fileinfo扩展。

在PHP 5.3中加入的Fileinfo扩展通过获取文件的“魔法字节”或“魔法数字”来判断文件的类型（和编码）。例如，构成GIF图片的数据必须以ASCII码GIF89a或GIF87a开头，构成PDF文件的数据必须以%PDF开头。

要使用Fileinfo，首先创建Fileinfo资源：

```
$fileinfo = finfo_open(kind);
```

*kind*值是个常量，指定创建的资源类型，判断文件类型的常量是`FILEINFO_MIME_TYPE`：

```
$fileinfo = finfo_open(FILEINFO_MIME_TYPE);
```

接下来，调用`finfo_file()`函数，提供Fileinfo资源和要检查的文件的引用：

```
finfo_file($fileinfo, $filename);
```

此函数基于文件的实际“魔法字节”返回文件的MIME类型（已经创建的资源）。

最后，在完成后，应该关闭Fileinfo的资源：

```
finfo_close($fileinfo);
```

接下来的脚本将会使用这个信息来确定上传的文件是RTF（富文本格式）。注意，只能够在PHP5.3版本或更新版本的PHP上才能够测试这个例子（参见图13-12）。如果你使用的是早期版本，需要通过PECL安装Fileinfo的扩展（PHP扩展社区库，<http://pecl.php.net>）。在Windows上，如果使用的是PHP 5.3或更高版本，Fileinfo的DLL文件应包含在安装中了，但你可能需要在`php.ini`配置文件中启用它（从peachpit.com下载附录A）。

```
Fatal error: Call to undefined function finfo_open() in
C:\xampp\htdocs\upload_rtf.php on line 18
```

图13-12 如果你的PHP安装不支持FileInfo扩展，会看到一个类似本图所示的错误信息

### 通过类型验证文件

(1) 在你的文本编辑器或IDE中创建名为upload\_rtf.php的新的PHP脚本（参见脚本13-3）。

#### 脚本 13-3 使用 FileInfo 扩展，这个脚本能够确认上传文件的类型

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
 DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Upload a RTF Document</title>
6 </head>
7 <body>
8 <?php # Script 13.3 - upload_rtf.php
9
10 // Check if the form has been submitted:
11 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
12
13 // Check for an uploaded file:
14 if (isset($_FILES['upload']) && file_exists($_FILES['upload']['tmp_name'])) {
15
16 // Validate the type. Should be RTF.
17 // Create the resource:
18 $fileinfo = finfo_open(FILEINFO_MIME_TYPE);
19
20 // Check the file:
21 if (finfo_file($fileinfo, $_FILES['upload']['tmp_name']) == 'text/rtf') {
22
23 // Indicate it's okay!
24 echo '<p>The file would be acceptable!</p>';
25
26 // In theory, move the file over. In reality, delete the file:
27 unlink($_FILES['upload']['tmp_name']);
28
29 } else { // Invalid type.
30 echo '<p style="font-weight: bold; color: #C00">Please upload an RTF document.</p>';
31 }
32
33 // Close the resource:
34 finfo_close($fileinfo);
35
36 } // End of isset($_FILES['upload']) IF.
37
38 // Add file upload error handling, if desired.
39
40 } // End of the submitted conditional.
41 ?>
42

```

```

43 <form enctype="multipart/form-data" action="upload_rtf.php" method="post">
44 <input type="hidden" name="MAX_FILE_SIZE" value="524288" />
45 <fieldset><legend>Select an RTF document of 512KB or smaller to be uploaded:</legend>
46 <p>File: <input type="file" name="upload" /></p>
47 </fieldset>
48 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
49 </form>
50 </body>
51 </html>

```

(2) 检查表单是否被提交。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

显示和处理表单将会使用同一个脚本。

(3) 检查上传的文件。

```
if (isset($_FILES['upload']) && file_exists($_FILES['upload']
['tmp_name'])) {
```

该脚本首先确认\$\_FILES ['upload']变量是否已经设置，这将是表单提交后的情况，然后条件语句确认上传的文件是否存在（默认情况下，在临时目录）。该子句会阻止验证上传失败的文件类型（例如，因为选定的文件太大）。

(4) 创建FileInfo资源。

```
$fileinfo = finfo_open(FILEINFO_MIME_TYPE);
```

这一行已经解释过了，创建FileInfo资源的目的就是检索文件的MIME类型。

(5) 检查文件的类型。

```
if (finfo_file($fileinfo, $_FILES['upload']['tmp_name']) == 'text/rtf') {
 echo '<p>The file would be acceptable!</p>';
```

如果finfo\_file()函数为上传文件返回了text/rtf值，那么该文件就是此脚本可以接受的类型。在这种情况下，会打印一条消息（参见图13-13）。

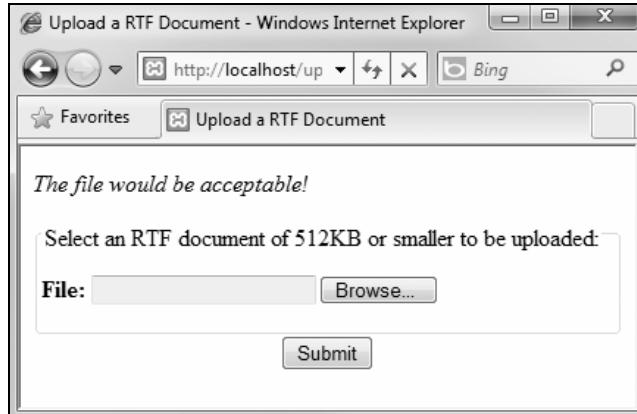


图13-13 如果选择的和上传的文档有的RTF MIME类型有效，用户会看到这个结果

(6) 删除上传的文件。

```
unlink($_FILES['upload']['tmp_name']);
```

在现实世界例子中，脚本现在会将文件移动到它在服务器上的最终位置。由于这个脚本仅用于验证文件类型，所以该文件可以删除。

(7) 完成类型条件语句。

```
} else { // Invalid type.
echo '<p style="font-weight: bold; color: #C00">Please upload an RTF document.</p>';
}
```

如果上传文件的MIME类型不是text/rtf，脚本将会打印错误信息（参见图13-14）。

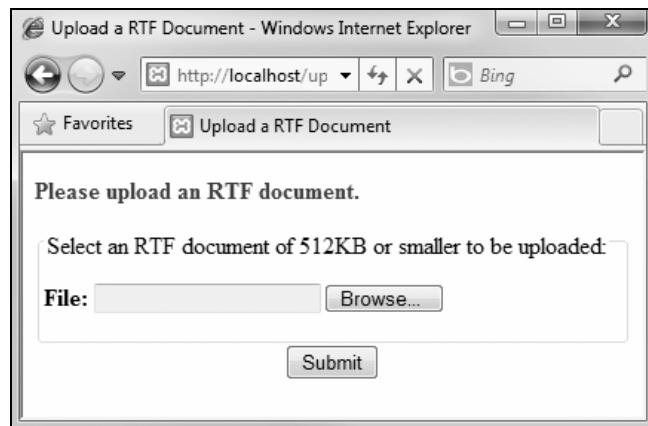


图13-14 拒绝上传MIME类型无效的文件

(8) 关闭Fileinfo资源。

```
ffinfo_close($fileinfo);
```

最后一步是在不需要使用时，关闭打开的Fileinfo的资源。

13

(9) 完成剩余的条件语句。

```
} // End of isset($_FILES ['upload']) IF.
} // End of the submitted conditional.
?>
```

如果发生了错误，还可以添加调试信息（比如相关的上传错误信息）。

(10) 创建表单。

```
<form enctype="multipart/form-data" action="upload_rtf.php" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="524288" />
<fieldset><legend>Select an RTF document of 512KB or smaller to be uploaded:</legend>
<p>File: <input type="file" name="upload" /></p>
</fieldset>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>
```

表单使用了恰当的`enctype`属性，包含建议最大文件大小的隐藏文本域，并且使用了文件输入框：接受文件上传的三个要求。以上是这个例子的所有内容（参见图13-13和图13-14）。

(11) 完成页面。

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

(12) 将文件保存为`upload_rtf.php`，放置到Web目录，并在浏览器中测试。

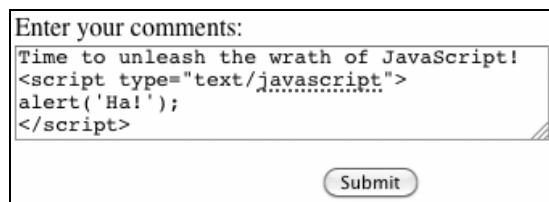
 提示

同一个Fileinfo资源可以应用于多个文件。在脚本完成所有资源的验证后关闭该资源。

## 13.4 阻止XSS攻击

HTML代码只是简单的纯文本，如`<b>`，Web浏览器赋予其特殊的含义（使文本加粗显示）。由于这个事实，Web站点的用户可以轻松地添加HTML或JavaScript代码到他们的表单数据中，比如上一个示例中的注释框。你可能会问，这有什么错吗？

许多动态驱动的Web应用程序会获取用户提交的信息，将其存储在数据库中，然后在另一个页面上重新显示该信息。考虑将论坛作为一个示例。最初，如果用户在他们的数据中输入HTML代码，这种代码就可能抛弃站点的布局和美感。更糟糕的是，JavaScript也只是纯文本，但它是Web浏览器内具有特殊含义（可执行的含义）的文本。如果在Web浏览器中重新显示在论坛中输入的恶意代码，它可能会创建弹出式窗口（参见图13-15和图13-16）、窃取cookie或者把浏览器重定向到其他站点。这种攻击被称为XSS攻击。比如在电子邮件示例中，你需要寻找在数据中发现的糟糕的字符串并使之无效，通过处理任何可能危险的PHP、HTML或JavaScript来阻止XSS攻击。



The screenshot shows a web form with the following content:

```
Enter your comments:
Time to unleash the wrath of JavaScript!
<script type="text/javascript">
alert('Ha!');
</script>
```

Below the text area is a "Submit" button.

图13-15 恶意用户可以在文本输入框中输入HTML、CSS和JavaScript

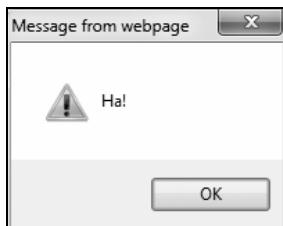


图13-16 在Web浏览器中显示注释时，在注释框中输入的JavaScript（参见图13-15）将创建这个警告对话框

PHP包括一些函数，用于处理在字符串内发现的HTML和其他代码。这些函数包括：

- `htmlspecialchars()`，它把&、'、"、<和>转变成HTML实体格式（&amp;、&quot;等）；
- `htmlentities()`，它把所有适用的字符转变成它们的HTML实体格式；
- `strip_tags()`，它删除所有的HTML和PHP标签。

这3个函数基本上是以从破坏性最小到最大的顺序列出的。至于你希望使用哪个函数，这依赖于手边的应用程序。为了演示这些函数的工作原理及其区别，让我们只创建一个带有一些文本的简单PHP页面（参见图13-15），对它运行这些函数，并打印出结果（参见图13-17）。

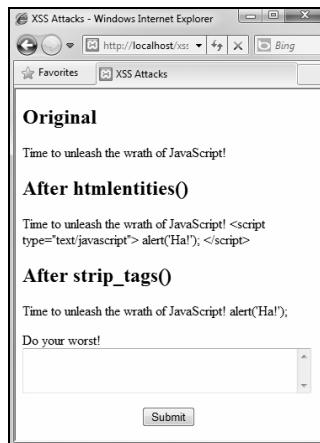


图13-17 由于`htmlentities()`和`strip_tags()`函数，可以使在表单文本框中输入的恶意代码（参见图13-15）无效

### 处理HTML代码

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为xss.php（参见脚本13-4）。

#### 脚本 13-4 对提交的文本应用 `htmlentities()`和 `strip_tags()`函数可以阻止 XSS 攻击

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
 DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>XSS Attacks</title>
6 </head>
7 <body>
8 <?php # Script 13.4 - xss.php
9
10 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
11
12 // Apply the different functions, printing the results:
13 echo "<h2>Original</h2><p>{$_POST ['data']}</p>";
14 echo '<h2>After htmlentities() </h2><p>' . htmlentities($_POST ['data']). '</p>';
15 echo '<h2>After strip_tags() </h2><p>' . strip_tags($_POST['data']). '</p>';


```

```

16
17 }
18 // Display the form:
19 ?>
20 <form action="xss.php" method="post">
21 <p>Do your worst! <textarea name="data" rows="3" cols="40"></textarea></p>
22 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
23 </form>
24 </body>
25 </html>

```

(2) 检查表单提交，并以其原始格式打印接收到的数据。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 echo "<h2>Original</h2> <p>{$_POST['data']}</p>";

```

为了把接收到的原始值与应用这些函数之后的结果作比较，必须先打印出原始值。

(3) 应用`htmlentities()`函数并打印结果。

```
echo '<h2>After htmlentities() </h2><p>' . htmlentities($_POST ['data']). '</p>';
```

为了防止提交的信息使页面内容混乱以及干扰Web浏览器，将对它运行`htmlentities()`函数。因此，将会转换任何HTML实体，例如，`<`和`>`将分别变成`&lt;`和`&gt;`。

(4) 应用`strip_tags()`函数并打印结果。

```
echo '<h2>After strip_tags() </h2><p>' . strip_tags($_POST ['data']). '</p>';
```

`strip_tags()`函数将完全删除任何HTML、JavaScript或PHP标签。因此，它是用于提交数据的最有效的函数。

(5) 完成PHP部分。

```

}
?>
```

(6) 显示HTML表单。

```

<form action="xss.php" method="post">
 <p>Do your worst! <textarea name="data" rows="3" cols="40"></textarea></p>
 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>
```

该表单（参见图13-15）只有一个用户要完成的字段：一个文本区。

(7) 完成页面。

```

</body>
</html>
```

(8) 将页面另存为`xss.php`，存放在Web目录中，并在Web浏览器中测试它。

(9) 查看页面的源代码，可以看出这些函数的完整作用（参见图13-18）。

### ✓ 提示

- `htmlspecialchars()`和`htmlentities()`这两个函数都可以带一个可选参数，指示应该如何处理引号。参见PHP手册，以了解特定细节。
- `strip_tags()`函数带一个可选参数，指示不应该清除什么标签。

```
$var = strip_tags ($var, '<p>
');
```

- `strip_tags()`函数甚至会清除无效的HTML标签，这些标签可能会引发问题。例如，`strip_tags()`函数会移出所有它认为是HTML标签的代码，即使它具有不正确的形式，如`<b>I forgot to close the tag.`
- `nl2br()`是一个与安全无关但相当有用的函数。它会把每个回车符（如那些在文本区中输入的回车符）转变成一个HTML `<br/>`标签。

```
<h2>Original</h2><p>Time to unleash the wrath of
JavaScript! <script type="text/javascript">
alert('Ha!');
</script></p><h2>After htmlentities()</h2><p>Time to
unleash the wrath of JavaScript! <script
type="text/javascript">
alert('Ha!');
</script></p><h2>After strip_tags()</h2><p>Time to
unleash the wrath of JavaScript!
alert('Ha!');
</p><form action="xss.php" method="post">
```

图13-18 页面的这个HTML源代码段（参见图13-17）显示了原始提交的值、使用 `htmlentities()`之后的值以及使用 `strip_tags()`之后的值

## 13.5 使用过滤器扩展

此前，本章介绍了类型转换的概念，这是将变量强制转换为合适类型的好方法。在下一章中，你会学习正则表达式，可以用于验证数据的类型和它具体的内容或格式。PHP 5.2中引入了过滤器扩展 ([www.php.net/filterer](http://www.php.net/filterer))，这是一个重要工具，它在相对简单的类型转换方法和更复杂的正则表达式之间架起了一座桥梁。

过滤器扩展可用于两个目的：验证数据或清理数据。如前所述，验证数据的过程就是确认数据符合预期目标。相较之下，修改数据就是删除不适当的字符，以使数据达到预期目标。

过滤器扩展中最重要的函数是 `filter_var()`：

```
filter_var(variable, filter [,options]);
```

函数的第一个参数是要筛选的变量，第二个参数是要应用的过滤器，第三个参数（可选）是其他附加条件。表13-3列出了验证过滤器，每个表示为一个常量。

表13-3 验证过滤器

常量	常量
FILTER_VALIDATE_BOOLEAN	FILTER_VALIDATE_IP
FILTER_VALIDATE_EMAIL	FILTER_VALIDATE_REGEXP
FILTER_VALIDATE_FLOAT	FILTER_VALIDATE_URL
FILTER_VALIDATE_INT	

例如，要确认变量有小数值，可以使用：

```
if (filter_var($var, FILTER_VALIDATE_FLOAT)) {
```

有些过滤器有可选的参数，最常见的是FILTER\_VALIDATE\_INT过滤器，用min\_range和max\_range控制可以接受的最小值和最大值。例如，下面这段代码确认了\$age变量是在1到120（含）之间的整数。

```
if (filter_var($var, FILTER_VALIDATE_INT, array('min_range' => 1, 'max_range' => 120))) {
```

为了清理数据，仍然要使用filter\_var()函数，但是要使用表13-4中列出的清理过滤器。

表13-4 清理过滤器

常量	常量
FILTER_SANITIZE_EMAIL	FILTER_SANITIZE_SPECIAL_CHARS
FILTER_SANITIZE_ENCODED	FILTER_SANITIZE_STRING
FILTER_SANITIZE_MAGIC_QUOTES	FILTER_SANITIZE_STRIPED
FILTER_SANITIZE_NUMBER_FLOAT	FILTER_SANITIZE_URL
FILTER_SANITIZE_NUMBER_INT	FILTER_UNSAFE_RAW

许多过滤器复制了其他的PHP函数。例如，FILTER\_SANITIZE\_MAGIC\_QUOTES与addslashes()作用相同、FILTER\_SANITIZE\_SPECIAL\_CHARS可以用来代替htmlspecialchars()、FILTER\_SANITIZE\_STRING()可以用来代替strip\_tags()。PHP手册中列出了几个附加的标记（作为常量）可以用做第三个可选参数来影响每个过滤器的行为。作为应用清理过滤器的例子，这段代码等同于在xss.php（脚本13-4）中使用的strip\_tags()：

```
echo '<h2>After strip_tags()</h2> <p>' . filter_var($_POST['data'],
 FILTER_SANITIZE_STRING) . '</p>';
```

如果你迷上了使用过滤器扩展，可能会欣赏它进行数据清理的一致性，而不想再使用strip\_tags()函数。

为了做练习，下一个例子会更新calculator.php（参见脚本13-2），让它清理所有进入的数据。请记住，你需要PHP 5.2或更高版本才能使用过滤器扩展。如果你使用的是较早版本的PHP，可以使用PECL安装过滤器扩展。

### 使用过滤器扩展

- (1)在你的文本编辑器或IDE中打开calculator.php（参见脚本13-2）。
- (2)将\$quantity变量的赋值更改为（参见脚本13-5）。

#### 脚本 13-5 使用过滤器扩展，这个脚本清理数据而不是像前面的版本中那样做类型转换。

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Widget Cost Calculator</title>
6 </head>
7 <body>
```

```

8 <?php # Script 13.5 - calculator.php #2
9 // This version of the script uses the Filter extension instead of typecasting.
10
11 // Check if the form has been submitted:
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13
14 // Sanitize the variables:
15 $quantity = (isset($_POST['quantity'])) ? filter_var($_POST['quantity'], FILTER_VALIDATE_INT,
16 array('min_range' => 1)) : NULL;
16 $price = (isset($_POST['price'])) ? filter_var($_POST['price'], FILTER_SANITIZE_NUMBER_FLOAT,
17 FILTER_FLAG_ALLOW_FRACTION) : NULL;
17 $tax = (isset($_POST['tax'])) ? filter_var($_POST['tax'], FILTER_SANITIZE_NUMBER_FLOAT,
18 FILTER_FLAG_ALLOW_FRACTION) : NULL;
18
19 // All variables should be positive!
20 if (($quantity > 0) && ($price > 0) && ($tax > 0)) {
21
22 // Calculate the total:
23 $total = $quantity * $price;
24 $total += $total * ($tax/100);
25
26 // Print the result:
27 echo '<p>The total cost of purchasing ' . $quantity . ' widget(s) at $' . number_format
28 ($price, 2) . ' each, plus tax, is $' . number_format ($total, 2) . '.</p>';
28
29 } else { // Invalid submitted values.
30 echo '<p style="font-weight: bold; color: #C00">Please enter a valid quantity, price, and tax
31 rate.</p>';
31 }
32
33 } // End of main isset() IF.
34
35 // Leave the PHP section and create the HTML form.
36 ?>
37 <h2>Widget Cost Calculator</h2>
38 <form action="calculator.php" method="post">
39 <p>Quantity: <input type="text" name="quantity" size="5" maxlength="10" value="<?php if
40 (isset($quantity)) echo $quantity; ?>" /></p>
40 <p>Price: <input type="text" name="price" size="5" maxlength="10" value="<?php if (isset($price))
41 echo $price; ?>" /></p>
41 <p>Tax (%): <input type="text" name="tax" size="5" maxlength="10" value="<?php if (isset($tax))
42 echo $tax; ?>" /></p>
42 <p><input type="submit" name="submit" value="Calculate!" /></p>
43 </form>
44 </body>
45 </html>

```

13

这个版本的脚本将通过几个途径改善它前面的版本。首先，使用`isset()`检查每个POST变量是否存在，而不是假设这个变量存在。如果变量没有设置，那么`$quantity`被指定为NULL。如果设置了变量，则运行`filter_var()`，将这个值清理为大于1的整数。然后将清理过的值赋给`$quantity`。为简洁起见，所有这些代码都使用第10章介绍的三目运算符编写。作为if-else条件语句，相同的代码将被写为：

```
if (isset($_POST['quantity'])) {
```

```

$quantity = filter_var($_POST['quantity'], FILTER_VALIDATE_INT, array('min_range' => 1));
} else {
 $quantity = NULL;
}

```

(3) 将\$price变量的赋值改为。

```

$price = (isset($_POST['price'])) ? filter_var($_POST['price'], FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_FRACTION) : NULL;

```

这段代码是步骤2中代码的重复，只是清理过滤器强制数据是浮点型。附加参数FILTER\_FLAG\_ALLOW\_FRACTION，说明它接受使用小数点的值。

(4) 将\$tax变量的赋值改为。

```

$tax = (isset($_POST['tax'])) ? filter_var($_POST['tax'], FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_FRACTION) : NULL;

```

这是步骤(3)的代码的重复。

(5) 保存网页，并将其存放到Web目录，并在Web浏览器中测试（参见图13-19和图13-20）。

图13-19 提交表单数据中的无效值

图13-20 会被过滤器扩展取消（与类型转换相反，例如，字符串cat会被转换为0）

### ✓ 提示

□ `filter_has_var()`函数确认是否存在具有给定名称的变量。

- ❑ `filter_input_array()` 函数可以一次对数组应用一组过滤器。有关详情（可能你会非常吃惊），请参阅 PHP 手册。

## 13.6 预防 SQL 注入攻击

恶意用户可能尝试的另一种攻击是SQL注入攻击。顾名思义，这些攻击是指企图把不良代码插入到站点的SQL查询中。这类攻击的目标之一是，它们将创建一个语法上无效的查询，从而在得到的出错消息中呈现关于脚本或数据库的某些信息（参见图13-21）。而注入攻击更大的目的是改变、破坏或暴露存储的数据。

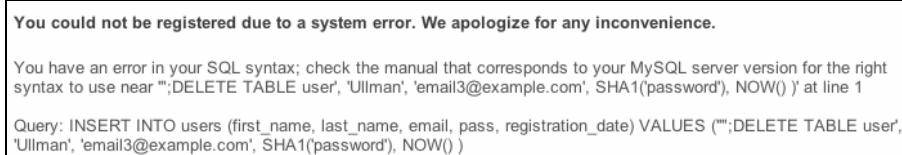


图13-21 如果站点呈现详细的出错消息并且没有正确地处理提交的值中有问题的字符，黑客就可能获悉关于服务器的许多信息

幸运的是，SQL注入攻击比较容易预防。首先验证查询中使用的所有数据（如果可能，执行类型转换，或应用过滤器扩展）。其次，使用函数如`mysqli_real_escape_string()`，这使得数据可以安全地用于查询。第9章介绍过这个函数。最后，不要在线的站点上显示错误详情。

使用`mysql_real_escape_string()`的替代选择是使用预处理语句。MySQL 4.1 版本中添加了预处理语句，从版本 5 起 PHP 就可以使用它们了（得益于 Improved MySQL 扩展）。如果不使用预处理语句，整个查询（包括 SQL 语法和具体的值）都会作为一个长字符串发送到 MySQL。然后 MySQL 分析并执行它。使用预处理查询，SQL 语法首先被发送到 MySQL 解析，确保它在语法上是有效的（例如，确认该查询没有引用不存在的表或列），然后单独发送特定的值。MySQL 使用这些值组合查询，然后执行它。预处理语句的好处很重要：它可以带来更高的安全性，并且可能提供更好的性能。我在这里重点关注的是安全方面，但请看一下框注“预处理语句性能”中有关性能的讨论。

可以通过任何`INSERT`、`UPDATE`、`DELETE`或`SELECT`查询创建预处理语句。首先定义查询，使用问号标记作为占位符。使用`edit_user.php`中的`SELECT`查询作为例子（脚本10-3）：

```
$q = "SELECT first_name, last_name, email FROM users WHERE user_id=$id";
```

如果是预处理语句，这个查询会变成：

```
$q = "SELECT first_name, last_name, email FROM users WHERE user_id=?";
```

接下来，预处理 MySQL 中的语句，将结果赋给 PHP 变量：

```
$stmt = mysqli_prepare($dbc, $q);
```

现在，MySQL 将解析查询，但不会执行它。

接下来，要将 PHP 变量绑定到查询的占位符上。换句话说，声明一个变量，用于第一个问号标记，为下一个问号标记使用的另外一个变量，等等。继续用同样的例子，编写如下代码：

```
mysqli_stmt_bind_param($stmt, 'i', $id);
```

使用表13-5中列出的字符，命令的*i*部分表示期望的值的类型。这里，查询期望接收到一个整数。作为另一个示例，下面显示了将如何处理第12章中的登录查询：

```
$q = "SELECT user_id, first_name FROM users WHERE email=? AND pass=SHA1(?);";
$stmt = mysqli_prepare($dbc, $q);
mysqli_stmt_bind_param($stmt, 'ss', $e, $p);
```

在这个例子中，还会发现一些有趣的事情：即使电子邮件地址和密码都是字符串，在查询中也不会用引号括住它们。这是预处理语句与标准查询之间的另一个区别。

一旦绑定了语句，你可以为PHP变量赋值（如果还未进行赋值），然后执行该语句。登录示例的代码将会是：

```
$e = 'email@example.com';
$p = 'mypass';
mysqli_stmt_execute($stmt);
```

`$e`和`$p`的值将会在预处理语句执行时。

要查看实际的过程，让我们写一个脚本，向论坛数据库（第6章创建）中的messages表添加帖子。我还将在利用这个机会来演示与预处理语句相关的另外几个函数。

表13-5 限制值的类型

字 符	含 义
d	Decimal
i	Integer
b	Blob (binary data)
s	All other types

### 预处理语句的性能

预处理语句比老式的查询方式更安全，而且可能也会更快。如果一个PHP脚本多次发送相同的查询到MySQL，每次使用不同的值，预处理语句确实可以加快速度。在这种情况下，只会把查询本身发送给MySQL，并且只会解析一次。然后，单独把值发送给MySQL。

作为一个无关紧要的示例，下面的代码将在MySQL中运行100次查询：

```
$q = 'INSERT INTO counter (num) VALUES (?)';
$stmt = mysqli_prepare($dbc, $q);
mysqli_stmt_bind_param($stmt, 'i', $n);
for ($n = 1; $n <= 100; $n++) {
 mysqli_stmt_execute($stmt);
}
```

即使查询被运行了100次，也只会把整个文本传递给MySQL一次，并且MySQL也只会解析它一次。MySQL版本5.1.17和更高版本包含一种缓存机制，可以提高预处理语句的其他用途的性能。

### 使用预处理语句

(1) 在文本编辑器或IDE中创建名为post\_message.php的新PHP脚本（参见脚本13-6）。

**脚本 13-6** 这个脚本（它代表消息发布页面的简化版本）使用预处理语句作为阻止 SQL 注入攻击的方式

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml11/
 DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Post a Message</title>
6 </head>
7 <body>
8 <?php # Script 13.6 - post_message.php
9
10 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
11
12 // Validate the data (omitted)
13
14 // Connect to the database:
15 $dbc = mysqli_connect ('localhost', 'username', 'password', 'forum');
16
17 // Make the query:
18 $q = 'INSERT INTO messages (forum_id, parent_id, user_id, subject, body, date_entered) VALUES
19 (?, ?, ?, ?, ?, NOW())';
20
21 // Prepare the statement:
22 $stmt = mysqli_prepare($dbc, $q);
23
24 // Bind the variables:
25 mysqli_stmt_bind_param($stmt, 'iiiss', $forum_id, $parent_id, $user_id, $subject, $body);
26
27 // Assign the values to variables:
28 $forum_id = (int) $_POST['forum_id'];
29 $parent_id = (int) $_POST['parent_id'];
30 $user_id = 3; // The user_id value would normally come from the session.
31 $subject = strip_tags($_POST['subject']);
32 $body = strip_tags($_POST['body']);
33
34 // Execute the query:
35 mysqli_stmt_execute($stmt);
36
37 // Print a message based upon the result:
38 if (mysqli_stmt_affected_rows($stmt) == 1) {
39 echo '<p>Your message has been posted.</p>';
40 } else {
41 echo '<p style="font-weight: bold; color: #C00">Your message could not be posted.</p>';
42 }
43
44 // Close the statement:
45 mysqli_stmt_close($stmt);

```

```

46
47 // Close the connection:
48 mysqli_close($dbc);
49
50 } // End of submission IF.
51
52 // Display the form:
53 ?>
54 <form action="post_message.php" method="post">
55
56 <fieldset><legend>Post a message: </legend>
57
58 <p>Subject: <input name="subject" type="text" size="30" maxlength="100" /></p>
59
60 <p>Body: <textarea name="body" rows="3" cols="40"></textarea></p>
61
62 </fieldset>
63 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
64 <input type="hidden" name="forum_id" value="1" />
65 <input type="hidden" name="parent_id" value="0" />
66
67 </form>
68 </body>
69 </html>

```

(2) 检查表单提交和论坛数据库的连接。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 $dbc = mysqli_connect ('localhost', 'username', 'password', 'forum');

```

注意：为了简单起见，我省略了基本数据验证和错误报告。虽然一个真正的网站（可以在第17章中找到这个脚本的更现实的版本），将检查邮件主题和正文不为空且不同的ID值为正整数，但是由于预处理语句提供的安全性，这个脚本仍然相对安全。

这个例子将使用在第6章中创建的forum数据库。

(3) 定义并准备查询。

```

$q = 'INSERT INTO messages (forum_id, parent_id, user_id, subject, body, date_entered) VALUES
 (?, ?, ?, ?, ?, NOW())';
$stmt = mysqli_prepare($dbc, $q);

```

此语法已经解释过。该查询使用占位符定义，可以在后面赋值。然后，`mysqli_prepare()`函数将该查询发送到MySQL，并将结果赋给`$stmt`。

这个查询最初是在第6章使用的。它填充了messages表的6个字段。`date_entered`列的值将会是`NOW()`函数的结果，而不是一个绑定的值。

(4) 绑定相应的变量，并创建需要插入的值的列表。

```

mysqli_stmt_bind_param($stmt, 'iiiss', $forum_id, $parent_id, $user_id, $subject, $body);
$forum_id = (int) $_POST['forum_id'];
$parent_id = (int) $_POST['parent_id'];
$user_id = 3;
$subject = strip_tags($_POST ['subject']);
$body = strip_tags($_POST['body']);

```

第一行表示，预处理语句将使用三个整数和两个字符串。这些值会在接下来的变量中找到。

对于这些变量，主题和正文中的值直接来自表单（参见图13-22），并使用`strip_tags()`函数消除任何潜在的危险代码。讨论的ID和父ID（这表示消息是否为现有消息的回复）也来自表单。它们将会被强制转换为整数（为了加强安全性，在转换结束后你需要确认它们是正数，当然也可以使用过滤器扩展）。

The form contains the following fields:

- Subject:** This is my subject.
- Body:** This is the body. It'll have apostrophes, "quotes", and even <b>HTML</b>.

At the bottom is a **Submit** button.

图13-22 简单的HTML表单

在现实的脚本中，用户的ID值来自会话，当用户登录时用户ID会被存储在会话中。

(5) 执行查询。

```
mysqli_stmt_execute($stmt);
```

最后，执行预处理语句。

(6) 打印的执行结果并完成循环。

```
if (mysqli_stmt_affected_rows ($stmt) == 1) {
 echo '<p>Your message has been posted.</p>';
} else {
 echo '<p style="font-weight: bold; color: #C00">Your message could not be posted.</p>';
 echo '<p>' . mysqli_stmt_error ($stmt) . '</p>';
}
```

可以使用`mysqli_stmt_affected_rows()`函数确认一条记录插入成功（返回受影响的行数）。这时会打印插入成功消息（参见图13-23）。如果出现问题，`mysqli_stmt_error()`函数返回特定的MySQL错误消息。这是用于调试目的，而不是用于在线站点。话虽这么说，通常PHP的错误信息比由`mysqli_stmt_error()`返回的有用得多（参见图13-24）。

The page displays the message "Your message has been posted." above the original form. The form fields are identical to Figure 13-22.

图13-23 如果查询影响了数据库中的一条记录，将返回这个结果图

```
Warning: mysqli_stmt_bind_param() [function\(mysqli-stmt-bind-param\)]: Number of elements in type definition string doesn't match number of bind variables in /Users/larryullman/Sites/phpmysql4/post_message.php on line 24

Your message could not be posted.

No data supplied for parameters in prepared statement
```

图13-24 有预处理语句的错误报告有时候会被混杂在一起

(7) 关闭语句和数据库连接。

```
mysqli_stmt_close($stmt);
mysqli_close($dbc);
```

第一个函数关闭了预处理语句，释放资源。同时，\$stmt不再有值。第二个函数关闭数据库连接。

(8) 完成PHP部分。

```
} // End of submission IF.
?>
```

(9) 创建表单。

```
<form action="post_message.php" method="post">
<fieldset><legend>Post a message:</legend>
<p>Subject: <input name="subject" type="text" size="30" maxlength="100" /></p>
<p>Body: <textarea name="body" rows="3" cols="40"></textarea></p>
</fieldset>
```

表单仅由一个主题文本输入框和一个消息正文文本框组成。

(10) 完成表单。

```
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
<input type="hidden" name="forum_id" value="1" />
<input type="hidden" name="parent_id" value="0" />
</form>
```

表单包含两个需要用户填写的字段和两个存储了查询需要的值的隐藏文本域。在这个脚本的现实版本中，forum\_id和parent\_id的值会动态确定。

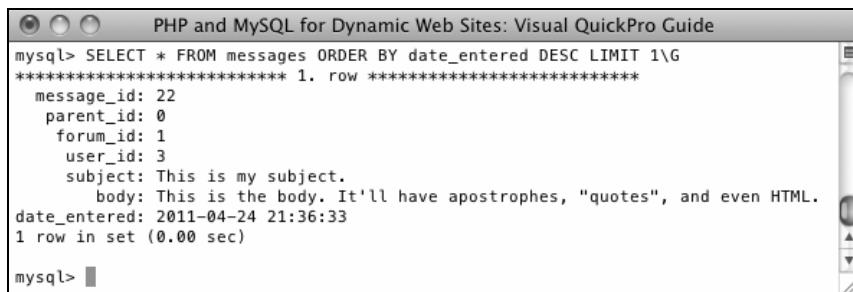
(11) 完成页面。

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

(12) 将文件另存为post\_message.php，存放到Web目录，并在Web浏览器中测试它（参见图13-25）。

#### ✓ 提示

- 有两种预处理语句。这里我已经演示了绑定参数，是将PHP变量被绑定到查询。另一种类型是绑定结果，是将查询的结果绑定到PHP变量。



```

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT * FROM messages ORDER BY date_entered DESC LIMIT 1\G
***** 1. row *****
message_id: 22
parent_id: 0
forum_id: 1
user_id: 3
subject: This is my subject.
body: This is the body. It'll have apostrophes, "quotes", and even HTML.
date_entered: 2011-04-24 21:36:33
1 row in set (0.00 sec)

mysql>

```

图13-25 在messages表中选择最近的条目，确认预处理语句（参见脚本13-6）起作用了。注意，帖子的HTML被清除了，但是引号仍然存在

### 更多安全建议

本章介绍了用于改进Web安全性的许多具体技术。下面列出了其他一些建议。

- 尽量减少从用户请求的和站点存储的信息。站点处理的信息越少意味着担心被盗的数据越少。
- 在工作过程中要学习、遵照和遵守安全建议。不要只依靠一章、一本书或一位作者的建议。
- 不要使用用户为上传的文件提供的名称。你会在第19章中看到一个替代的解决方案。
- 观察如何使用数据库引用。例如，如果用户ID是其数据库中的主键并且存储在cookie中（如第12章所述），那么恶意用户只需更改该cookie值，即可访问另一个用户的账户。
- 不要显示详细的错误消息（这点在第8章强调过）。
- 使用加密技术（这在第7章中讨论过，在我编写的*PHP 5 Advanced: Visual QuickPro Guide*（Peachpit Press，2007年）一书中讨论了关于服务器加密的内容）。
- 不要存储信用卡号码、社会保险号码、银行业务等信息。唯一的例外是，你的荷包够鼓，能为最好的安全买单，并能够支付当数据从你的站点被盗取时发生的诉讼费用（这将不可避免地发生）。
- 如果可以，使用SSL。全连接是服务器可以提供给用户的最佳保护措施之一。
- 可靠、一致地保护需要保护的所有页面和目录。永远不要假设仅仅由于没有指向敏感区域的链接，人们就不会找到它们。确保对需要限制访问的页面和目录添加限制。

我最后一个建议是要知道自己的局限性。作为程序员，你可能会考虑应该如何使用脚本，这并不等同于将会如何有意或无意地使用它。尝试破坏你的网站来查看会发生什么。做一些糟糕的事情和一些错误的事情。也让别人尝试破坏它（通常很容易找到这样的志愿者）。当你编码时，与假定人们将总是正确地使用页面相比，假定没有人永远都会正确地使用页面要安全得多。

13

### 预防暴力攻击

暴力攻击是在抱有万一取得成功的希望的情况下执行许多次尝试，以试图登录到安全系统。它不是一种先进的攻击手段，因此得名“暴力”。例如，如果你有一个需要用户名和密码登录的过程，对可能的用户名/密码组合数目会有一个限制。该限制可能会是几十亿或几万亿，但是，它仍然是

一个有限的数目。暴力攻击使用算法和自动化流程，反复尝试组合直到成功。

防止暴力破解成功的最好的办法是，让用户注册良好的、难猜的密码：包含字母、数字和标点符号；使用字典中没有的单词；长度至少有8个字符，等等。此外，不要给出登录失败的原因：指出用户名和密码的组合不正确不会泄露任何信息，但是指出用户名不正确或者密码不正确就有些矫枉过正了。

要停止一个正在进行的暴力攻击，你还可以限制一个IP不正确登录尝试的次数。IP地址不会经常改变，但在暴力攻击中，相同的IP地址会在短短的几分钟内尝试多次登录。你必须根据IP地址追踪不正确的登录，然后，在数次无效的尝试后，屏蔽那个IP地址24小时（或其他处理）。或者，如果你不想采取这种严厉的措施，可以使用“增量延迟”防御：同一个IP地址每进行一次不正确的登录，就在响应中产生一段额外的延迟（使用PHP的sleep()函数创建延时）。人们可能不会注意或者厌烦这种延时，但是自动的攻击肯定会。

## 13.7 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 13.7.1 回顾

- 哪些字符串和字符有可能成为垃圾邮件的标识？
- strpos()的作用是什么？它的语法是什么？
- str\_replace()的作用是？它的语法是什么？
- array\_map()函数的作用是？它的语法是什么？
- 什么是类型转换？如何在PHP进行类型转换？
- 将上传的文件移动到它在服务器上指定位置的函数是什么？
- 什么是Fileinfo的扩展？它如何使用？
- htmlspecialchars()函数作用是什么？
- htmlentities()函数作用是什么？
- strip\_tags()函数的作用是什么？
- 将换行符转换成HTML换行标签的函数是什么？
- 过滤器扩展中最重要的函数是什么？它如何使用？
- 什么是预处理语句？预处理语句相比标准的数据库查询有哪些优点？
- 使用预处理语句的语法是什么？

### 13.7.2 实践

- 如果你没有为电子邮件验证使用过滤器函数，或为网站的联系表单使用spam\_scrubber()函数，现在就用下试试。

- 将calculator.php改为允许没有税率。
- 更新第3章中的calculator.php，让它同样使用类型转换或过滤器扩展。（提示，该计算器基于距离，平均每加仑的英里数和每加仑的平均价格。）
- 修改upload\_rtf.php让它在文件不是text/rtf的情况下报告上传文件的MIME类型。
- 创建一个PHP脚本，报告任何上传文件的MIME类型。
- 为本书前面的示例应用strip\_tags()函数，例如注册示例，以防止不正确的代码存储到数据库。
- 为第12章的登录过程应用过滤器函数，保证提交的电子邮件地址符合电子邮件地址的格式。
- 为第10章的delete\_user.php和edit\_user.php使用过滤器函数或类型转换。
- 为第11章的show\_image.php脚本应用FileInfo扩展。

# Perl兼容的正则表达式

### 本章内容

- 创建测试脚本
- 定义简单的模式
- 使用量词
- 使用字符类别
- 查找所有匹配
- 使用修饰符
- 匹配和替换模式
- 回顾和实践

**正**则表达式是一种极其强大（但是乏味）的工具，在今天的大多数编程语言甚至在许多应用程序中都可以使用它。可以把正则表达式看作一种精心设计的匹配模式系统。你首先编写模式，然后使用PHP的内置函数之一将模式应用于一个值（将正则表达式应用于字符串，即使它是带有数值的字符串）。字符串函数可以判断出某段文本中是否包含名字John，而正则表达式可以轻松地找出John、Jon和Jonathon。

由于正则表达式语法非常复杂（虽然使用它们的函数很简单），因此本章重点关注的是掌握正则表达式语法。PHP代码将非常简单，后面的章节将更好地把正则表达式纳入到现实的脚本中。

### 14.1 创建测试脚本

如前所述，正则表达式涉及将模式应用于值。这是使用许多函数之一完成的，其中最重要的是`preg_match()`。这个函数返回0或1，指示模式是否与字符串匹配。其基本语法如下：

```
preg_match(pattern, subject)
```

一旦`preg_match()`函数找到一个匹配，它就会停止。如果你需要找到所有的匹配，可使用`preg_match_all()`。在临近本章末尾时将讨论这个函数。

在给`preg_match()`提供模式时，需要用引号括住它，因为它将是一个字符串。由于双引号内的许多转义字符具有特殊的含义（比如`\n`），我提倡使用单引号定义模式。

其次，在引号内，需要对定界符内对模式进行转义。定界符可以是除字母数字或反斜杠外的任何

字符，必须使用相同的字符来标记模式的开始和结束。通常，你将看到使用的是正斜杠。因此，为了查看单词cat是否包含字母a，将编写如下代码：

```
if (preg_match('/a/', 'cat')) {
```

如果需要匹配模式中的正斜杠，可以使用不同的定界符，比如竖线（|）或感叹号（!）。

本章采用大量篇幅介绍用于定义模式的所有规则。为了最好地通过示例学习它们，我们首先创建一个简单的PHP脚本，它获取一个模式和一个字符串（参见图14-1），并返回正则表达式结果（参见图14-2）。

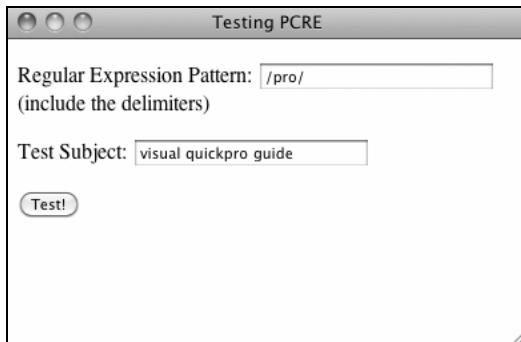
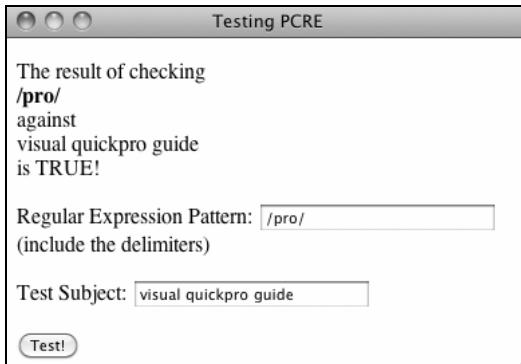


图14-1 将用于实际应用正则表达式的HTML表单



14

图14-2 该脚本将打印正则表达式中使用的是什么值以及结果是什么。还会把表单制作成黏性，以记住以前提交的值

### 匹配模式

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为pcr.php（参见脚本14-1）。

#### 脚本 14-1 将使用这个 PHP 脚本讲解和演示复杂的正则表达式语法

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
 DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
```

```

4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Testing PCRE</title>
6 </head>
7 <body>
8 <?php // Script 14.1 - pcre.php
9 // This script takes a submitted string and checks it against a submitted pattern.
10
11 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
12
13 // Trim the strings:
14 $pattern = trim($_POST['pattern']);
15 $subject = trim($_POST['subject']);
16
17 // Print a caption:
18 echo "<p>The result of checking
$pattern
against
$subject
is ";
19
20 // Test:
21 if (preg_match ($pattern, $subject)) {
22 echo 'TRUE!</p>';
23 } else {
24 echo 'FALSE!</p>';
25 }
26
27 } // End of submission IF.
28 // Display the HTML form.
29 ?>
30 <form action="pcre.php" method="post">
31 <p>Regular Expression Pattern: <input type="text" name="pattern" value="<?php if
32 (isset($pattern)) echo htmlentities($pattern); ?>" size="40" /> (include the delimiters)</p>
33 <p>Test Subject: <input type="text" name="subject" value="<?php if (isset ($subject)) echo
34 htmlentities($subject); ?>" size="40" /></p>
35 <input type="submit" name="submit" value="Test!" />
36 </form>
37 </body>
38 </html>

```

(2) 检查表单提交。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

(3) 处理传入的值。

```
$pattern = trim($_POST['pattern']);
$subject = trim($_POST['subject']);
```

表单将向这个相同的脚本提交两个值。它们都应该进行处理，以确保任何多余的空格都不会改变结果。我省略了检查每个输入框是否为空这个步骤，但是如果你想检查输入框是否为空，也可以包括该操作。

注意，如果你的服务器启用了Magic Quotes，你将会看到额外的斜杠被添加到表单数据中（参见图14-3）。为了避免它，这时需要使用stripslashes()函数，如下所示：

```
$pattern = stripslashes(trim($_POST['pattern']));
$subject = stripslashes(trim($_POST['subject']));
```

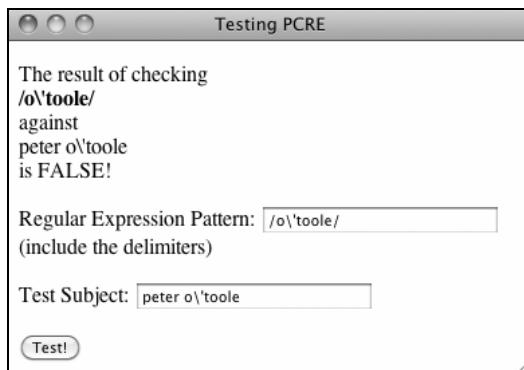


图14-3 如果你的服务器上启用了Magic Quotes，脚本将会为特定的字符添加斜杠，很有可能让正则表达式失败

(4) 打印标题。

```
echo "<p>The result of checking
$pattern
against
$subject
is ";
```

如图14-2所示，这个脚本的表单处理部分首先将打印使用的值。

(5) 运行正则表达式。

```
if (preg_match ($pattern, $subject)) {
 echo 'TRUE!</p>';
} else {
 echo 'FALSE!</p>';
}
```

要针对字符串测试模式，可以把它们都提供给preg\_match()函数。如果该函数返回1，就意味着产生匹配，这个条件将为真，并将打印单词TRUE。如果没有产生匹配，条件将为假，并会指出这一点（参见图14-4）。

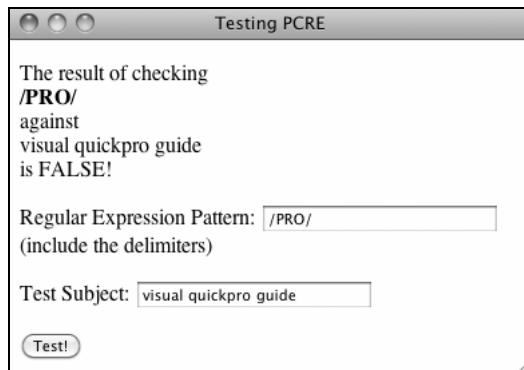


图14-4 如果模式与字符串不匹配，这将是结果。这幅图还说明默认情况下正则表达式是区分大小写的

(6) 完成提交条件块和PHP代码。

```

} // End of submission IF.
?>

```

(7) 完成HTML表单。

```

<form action="pcre.php" method="post">
 <p>Regular Expression Pattern: <input type="text" name="pattern" value="<?php if (isset($pattern))
 echo htmlentities($pattern); ?>" size="40" /> (include the delimiters)</p>
 <p>Test Subject: <input type="text" name="subject" value="<?php if (isset($subject))echo htmlentities
 ($subject);?>" size="40" /></p>
 <input type="submit" name="submit" value="Test!" />
</form>

```

该表单包含两个文本框，它们都是黏性的（使用处理过的值）。因为这两个值可能包含引号和其他会与表单的“黏性”相冲突的字符，所以每个变量的值也需要通过`htmlentities()`来发送。

(8) 完成HTML页面。

```

</body>
</html>

```

(9) 将文件另存为`pcre.php`，存放在Web目录中，并在Web浏览器中测试它。

尽管你还不知道用于创建模式的规则，还是可以使用字面量`a`测试，或者检查任何其他的字面量值。记住：要在模式周围使用定界符，否则将会看到出错消息（参见图14-5）。

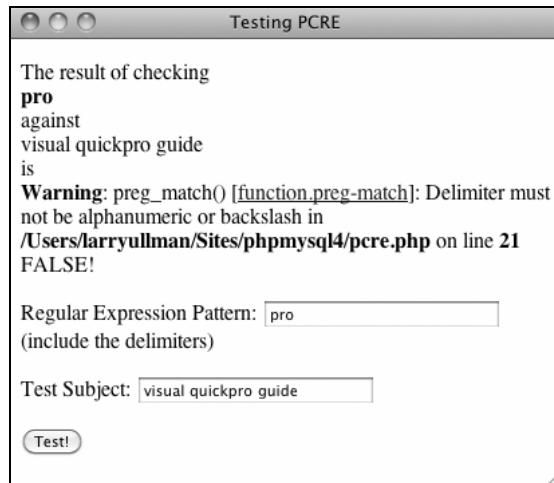


图14-5 如果没有用匹配的定界符包围模式，将看到出错消息

### ✓ 提示

- 有些文本编辑器（如BBEdit和emacs）允许使用正则表达式来匹配和替换文档内乃至多个文档中的模式。
- PCRE函数全都使用既定的区域。区域（在第14章中更详细地讨论了它）反映了计算机指定的国家和语言及其他设置。

- 先前版本的PHP还支持另外一种类型的，称为POSIX的正则表达式。这已经被弃用了，意味着它们将会被未来版本的语言所放弃。

## 14.2 定义简单的模式

使用PHP的某个正则表达式函数确实很容易，但是定义要使用的模式却很困难。PHP有许多规则用于创建模式。可以单独使用这些规则，或者组合使用它们，使得你的模式相当简单或者非常复杂。首先，你将看到哪些字符可用于定义简单的模式。作为一种格式化规则，我将用粗体定义模式，并用斜体指示模式匹配的内容。这些解释中的模式将不会放置在定界符或引号内（在preg\_match()内使用模式时将需要它们），这样做只是为了使事情更清晰。

你将用于定义模式的第一类字符是字面量（literal），字面量是一个值，其书写方式与所解释的完全一样。例如，模式a将匹配字母a，ab将匹配ab，等等。因此，假定执行不区分大小写的查找，那么rom将匹配下列任意字符串，因为它们全都包含rom：

- CD-ROM
- Rommel crossed the desert.
- I'm writing a roman à clef.

你的模式将与字面量一起使用元字符（meta-character）。元字符是特殊的符号，其含义超出了它们的字面量值（参见表14-1）。虽然a只是简单地意指a，句点(.)将匹配除换行符之外的任意单个字符（.匹配a、b、c、下划线、空格等，仅仅不匹配\n）。为了匹配任意元字符，需要对其进行转义，这与对一个引号进行转义来打印它非常相似。因此，\.将匹配句点本身。因此，1.99将匹配1.99、1B99或1299（1后面接着任意字符，然后再接着99），但是1\.99只匹配1.99。

表14-1 元字符

字 符	含 义
\	转义符
^	指示字符串的开始
\$	指示字符串的结尾
.	除换行符之外的任意单个字符
	二中择一（或）
[	类的开始
]	类的结尾
(	子模式的开始
)	子模式的结尾
{	量词的开始
}	量词的结尾

14

两个元字符指定必须在什么位置查找某些字符。一个是插入符号(^)，它将匹配以插入符号后面的内容开头的字符串。还有一个是美元符号(\$)，用于标记模式的结束。相应地，^a将匹配以a开头的任何字符串，而a\$则对应于以a结尾的任何字符串。因此，^a\$只匹配a（同时以a开头和结尾的字符串）。

这两个元字符（插入符号和美元符号）对于验证是至关重要的，因为验证通常需要检查整个字符串的值，而不仅仅是检查在一个字符串中是否存在另一个字符串。例如，在不使用这两个字符的情况下使用电子邮件匹配模式将匹配包含电子邮件地址的任何字符串。使用以插入符号开头并以美元符号结尾的电子邮件匹配模式将匹配只包含有效电子邮件地址的字符串。

正则表达式还利用竖线（|）作为或（or）的等价表示。因此，`a|b`将匹配包含`a`或`b`的字符串。[在模式内使用竖线被称为交替（alternation）或分支（branching）。]因此，`yes|no`将完全接受这两个单词中的任何一个〔交替并不仅仅发生在包围它的两个字母（`s`和`n`）之间〕。

一旦理解了基本的符号，就可以开始使用圆括号把字符组合进更复杂的模式中。组合将按你期待的那样工作：`(abc)`将匹配`abc`，`(trout)`将匹配`trout`。可以把圆括号看作用于建立一个更大尺寸的新字面量。由于PCRE中的优先级规则，因此，`yes|no`和`(yes)|(no)`是等价的。但是，`(even|heavy) handed`将匹配`even handed`或`heavy handed`。

### 使用简单的模式

- (1) 如果还没有在浏览器中加载`pcre.php`，就加载它。
- (2) 检查字符串是否包含字母`cat`（参见图14-6）。

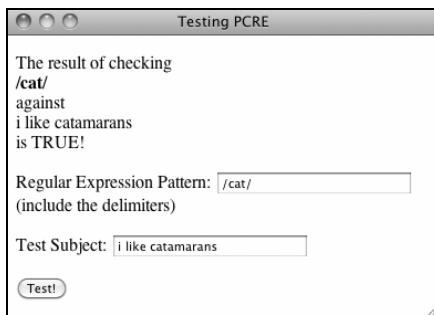


图14-6 在字符串中寻找`cat`

要执行该操作，可使用字面量`cat`作为模式并使用任意数量的字符串作为主题。下面任何一个字符串都将是一个匹配：`catalog`、`catastrophe`、`my cat left`等。目前暂时全都使用小写字母，因为`cat`不会匹配`Cat`（参见图14-7）。

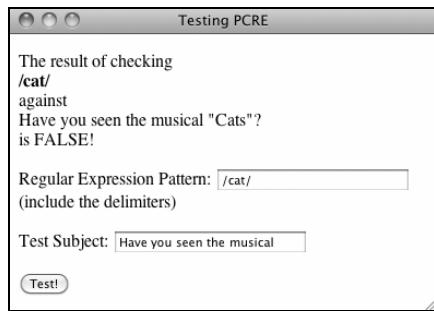


图14-7 不要忘记PCRE默认情况下执行区分大小写的比较

也要记住在模式周围使用定界符。

(3) 检查字符串是否以`cat`开头 ( 参见图14-8 )。

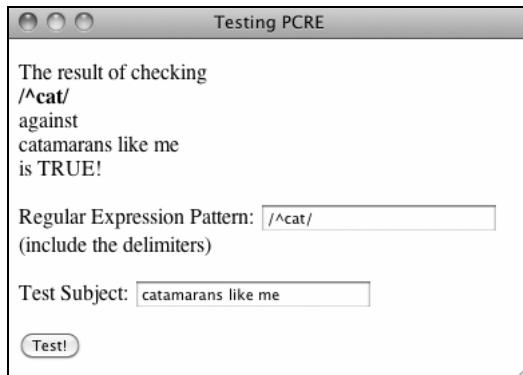


图14-8 模式中的插入符号说明必须在字符串的开头找到匹配

为了使模式应用于字符串的开头，可使用插入符号作为第一个字符 (`^cat`)。句子`my cat left`现在将不是一个匹配。

(4) 检查字符串是否包含单词`color`或`colour` ( 参见图14-9 )。

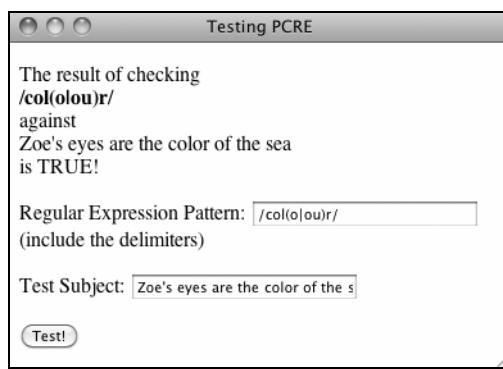


图14-9 通过使用竖线元字符，可以更灵活执行查找

用于寻找这个单词的美式或英式拼写的模式是`col(o|ou)r`。前三个字母 (`col`) 必须存在。其后需要接着`o`或`ou`。最后，需要一个`r`。

#### ✓ 提示

- 如果希望与一个字符串内的另一个字符串精确匹配，可使用`strstr()`函数，它比正则表达式快一些。事实上，一条经验法则是：仅当不能使用其他任何函数或技术完成手边的任务时才应该使用正则表达式。
- 可以使用`\Q`或`\E`对模式中的一串字符进行转义，其中的每个字符都将以字面方式进行处理 ( 因此`\Q$2.99?\E`将匹配`$2.99?` )。

- 要匹配单个反斜杠，必须使用\\\\.其原因是：在正则表达式中匹配反斜杠需要对反斜杠进行转义，这会导致\\.然后，为了在PHP字符串中使用反斜杠，也必须对它进行转义，因此对两个反斜杠进行转义就意味着总共会得到4个反斜杠。

### 14.3 使用量词

你刚才已经见过并且使用过一些元字符，其中最重要的是插入符号和美元符号。接下来，有3个元字符允许多次出现：`a*`将匹配0个或多个`a`（0个`a`、`a`、`aa`、`aaa`等）；`a+`将匹配一个或多个`a`（`a`、`aa`、`aaa`等，但是至少必须有一个`a`）；`a?`将匹配最多一个`a`（`a`或者没有`a`的匹配）。这些元字符全都充当模式中的量词，就像花括号一样。表14-2列出了所有量词。

表14-2 量词

字 符	含 义
?	0次或1次
*	0次或多次
+	1次或多次
{x}	正好出现x次
{x, y}	在x次和y次之间（含x和y）
{x,}	至少出现x次

为了匹配一定数量的字母，可以把数量放在花括号（{}）之间，指出具体的数字，它可以只是一个最小值，或者同时指出最小值和最大值。因此，`a{3}`将匹配`aaa`；`a{3,}`将匹配`aaa`、`aaaa`等（3个或更多的`a`）；`a{3,5}`则只匹配`aaa`、`aaaa`和`aaaaaa`（在3个`a`和5个`a`之间）。

注意，量词应用于出现在它之前的内容，因此`a?`将匹配0个或一个`a`，`ab?`将匹配`a`及其后的0个或一个`b`，但是`(ab)?`将匹配0个或一个`ab`。因此，为了匹配`color`或`colour`（参见图14-8），也可以使用`color?r`作为模式。

#### 使用量词

(1) 如果还没有在浏览器中加载`pcre.php`，就加载它。

(2) 检查字符串是否包含字母`c`和`t`，以及它们之间的一个或多个字母（参见图14-10）。

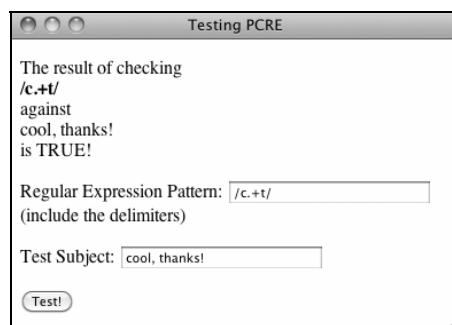


图14-10 当把加号用作量词时，它需要存在一个或多个字符

要执行该操作，可使用`c.+t`作为模式并使用任意数量的字符串作为主题。记住：句点匹配任意字符（换行符除外）。下面每个字符串都是一个匹配：`cat`、`count`、`coefficient`等。单词`doctor`不会匹配，因为在`c`和`t`之间没有字母（尽管`doctor`将匹配`c.*t`）。

(3) 检查字符串是否匹配`cat`或`cats`（参见图14-11）。

首先，如果你想建立精确的匹配，可同时使用插入符号和美元符号。然后，你将得到字面量文本`cat`，其后接着一个`s`，再接着一个问号（表示0个或1个`s`）。最终的模式（`^cats?$`）将匹配`cat`或`cats`，但是不匹配`my cat left`或`I like cats`。

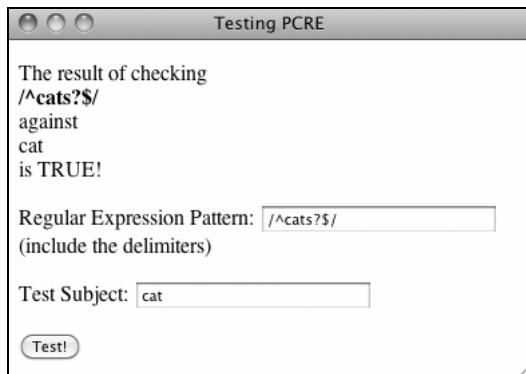


图14-11 可以通过在模式中添加`s?`来检查许多单词的复数形式

(4) 检查字符串是否以`.33`、`.333`或`.3333`结尾（参见图14-12）。

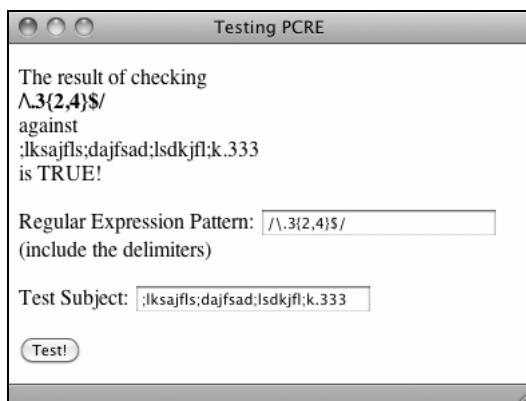


图14-12 花括号允许你规定量词提供的可接受范围

为了查找一个句点，可以用反斜杠对它进行转义：`\.`。为了查找三个句点，可使用字面量`3`。为了指定查找的范围`3`，可使用花括号（`{}`）。综合起来，该模式是`\.\.3{2,4}`。由于字符串应该结束于这个模式（其后不能接着别的内容），所以可以在模式末尾添加一个美元符号：`\.\.3{2,4}\$`。

无可否认，这个示例毫无价值（不能确定你何时需要这样做），但是它确实说明了几种情况。该

模式将匹配许多内容 (12.333、varmit.3333、.33, 看起来像.33) 但是不匹配12.3或12.334。

(5) 匹配5位的数字 (参见图14-13)。

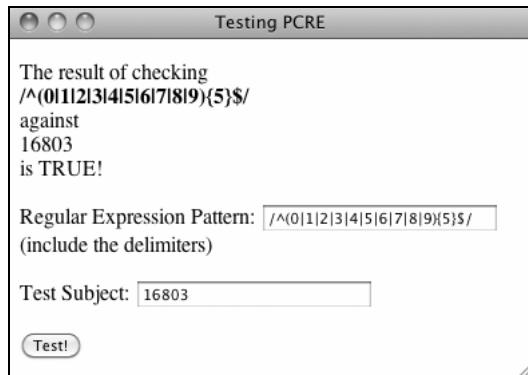


图14-13 用于确认一个数包含5位数字的正确的测试

一个数字可以是0至9之间的任意数字之一，因此，模式的核心部分是(0|1|2|3|4|5|6|7|8|9)。明白地讲，这意味着：数字可以是0、1、2或3……为了使之是一个5位的数字，可以在它后面接一个量词：(0|1|2|3|4|5|6|7|8|9){5}。最后，为了精确匹配它（与匹配字符串内的一个5位数字相对），可使用插入符号和美元符号：^(0|1|2|3|4|5|6|7|8|9){5}\$。

当然，这是一种非常有用的方式。

#### ✓ 提示

- 在使用花括号指定字符数时，必须总是包括最小数字。最大数字是可选的：a{3}和a{3,}是可以的，但是a{,3}则不行。
- 尽管这个示例演示的是在编程时如何编写和执行你自己的正则表达式，但是可以通过搜索Internet获得大量的工作示例。

## 14.4 使用字符类别

作为演示的最后一个示例（图14-13），只依靠模式中的字面量可能是索然无味的。必须写出所有那些数字以匹配任意数字是一种愚蠢的做法。想象一下，如果你想匹配任何4字母的单词：  
^(a|b|c|d...){4}\$ (并且甚至不考虑大写字母) !为了使这些常见的引用更容易，可以使用字符类别 (character class)。

类别是通过把字符放置在方括号 ([ ]) 内创建的。例如，你可以用[aeiou]来匹配任意一个元音字母。这等价于(a|e|i|o|u)。或者可以使用连字符来指定字符的范围：[a-z]用于匹配任意单个小写字母，[A-Z]则匹配任何大写字母，[A-Za-z]一般用于匹配任意字母，[0-9]则用于匹配任意数字。例如，[a-z]{3}将匹配abc、def、oiw等。

在类别中，除了4个元字符之外，大多数元字符都会按字面量处理。反斜杠仍然是转义符，但是如果把插入符号 (^) 用作类别中的第一个字符，那么它将是“非”运算符。因此，[^aeiou]将匹配任

何非元音字母。类别中的另外一个元字符是短划线，它指示一个范围（如果把短划线用作类别中的最后一个字符，它就是字面上的短划线）。当然，闭方括号（]）仍然具有类别终止符的意义。

自然，类别可以同时具有范围和字符字面量。可以通过[A-z '.]表示人们的名字，其中可以包含字母、空格、撇号和句点（同样，类别中的句点不需要被转义，因为它会在这里失去其本来的意义）。

除了创建你自己的类别之外，有6个已经定义好的类别，它们具有自己的快捷方式（参见表14-3）。数字和空格类别很容易理解。单词字符类别并不是指语言意义上的“单词”，而是指用空格或标点符号断开的字符串。

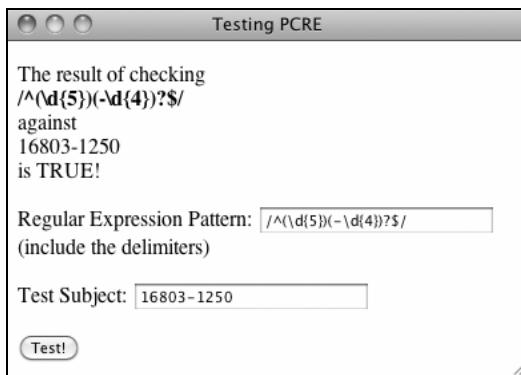
表14-3 这些字符类别常常用在正则表达式中

类 别	快 捷 方 式	含 义
[0-9]	\d	任意数字
[\f\r\t\n\v]	\s	任意空白
[A-Za-z0-9]	\w	任意单词字符
[^0-9]	\D	非数字
[^\f\r\t\n\v]	\S	非空白
[^A-Za-z0-9]	\W	非单词字符

使用这种信息，可以更轻松地把5位的数字（即邮政编码）模式写成`^[0-9]{5}$`或`^\d{5}$`。举另外一个例子，`can\s?`将匹配*can not*和*cannot*（单词*can*，其后接着0个或一个空格字符，再接着*not*）。

### 使用字符类别

- (1) 如果还没有在Web浏览器中加载pcre.php，就加载它。
- (2) 检查字符串是否被格式化为有效的美国邮政编码（参见图14-14）。



14

图14-14 匹配美国邮政编码的模式，采用5位数字或者5位数字加4位数字的格式

美国邮政编码总是开始于5位数字（`^\d{5}`）。但是，有效的邮政编码还可以具有一个短划线，其后接着另外4位数字（`-\\d{4}$`）。为了使这最后一部分是可选的，可以使用问号（0个或1个量词）。这样，这个完整的模式就是`^(\\d{5})(-\\d{4})? $`。为了使它们全都更清晰，也把模式的第一部分（匹配5位数字）组合在圆括号中，尽管在这个示例中不需要这样做。

(3) 检查字符串是否不包含空格 (参见图14-15)。

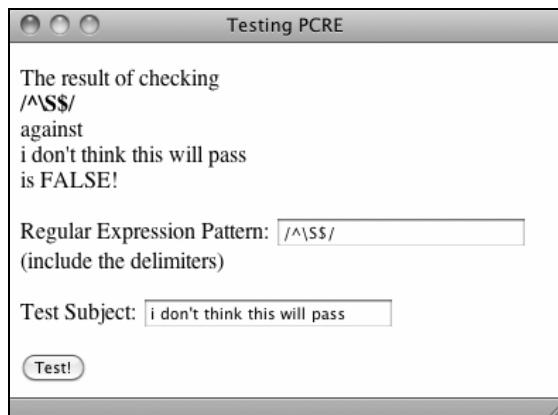


图14-15 非空白快捷方式可用于确保提交的字符串是连续的

\S字符类别快捷方式将匹配非空格字符。为了确保整个字符串都不包含空格，可以使用插入符号和美元符号：^\S\$。如果不使用它们，那么模式只将确认主题中至少包含一个非空格字符。

(4) 验证电子邮件地址 (参见图14-16)。

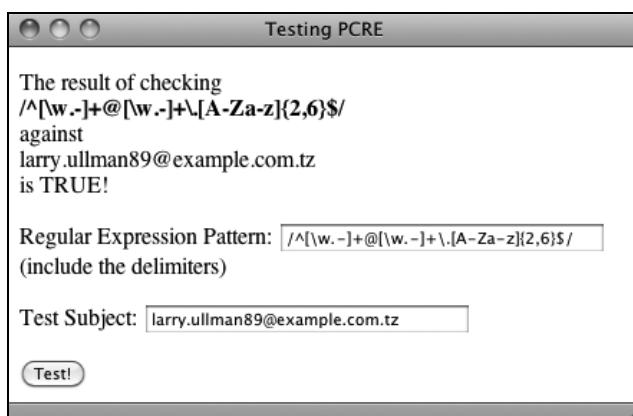


图14-16 极佳、可靠地验证电子邮件地址

模式`^[w.-]+@[w.-]+\.[A-Za-z]{2,6}$`提供了相当好的电子邮件验证。用插入符号和美元符号包围它，因此字符串必定是有效的电子邮件地址，而不会是其他任何内容。电子邮件地址开始于字母、数字和下划线(用\w表示)，以及句点(.)和短划线。第一部分将匹配*larryullman*、*larry77*、*larry.ullman*、*larry-ullman*等。接下来，所有的电子邮件地址都将包括一个并且只将包括一个@。之后，可以有任何数量的字母、数字、句点和短划线。下面这些是域名：*dmcinsights*、*smith-jones*、*amazon.co*(比如在*amazon.co.uk*中)等。最后，所有的电子邮件地址都以一个句点和2~6个字母结尾，这代表.com、.edu、.info、.travel等。

### ✓ 提示

□ 我认为邮政编码示例极佳地演示了正则表达式是多么复杂、多么有用。一种模式准确测试了邮政编码的两种格式，这非常美妙。但是，当利用引号和定界符把它置于PHP代码中时，它并不容易理解：

```
if (preg_match ('/^(\d{5})(-\d{4})?$/ ', $zip)) {
```

它看起来莫名其妙，对吗？

□ 这个电子邮件地址验证模式相当好，尽管并不是完美无缺。它将允许一些无效的地址通过验证（比如以句点开头或者包含多个句点在一起的地址）。不过，百分之百有效的验证模式极长，并且使用正则表达式往往只是用于排除大量无效的条目，并且不会疏忽地排除任何有效的条目。

□ 正则表达式（特别是PCRE正则表达式）可能非常复杂。在开始时，使用它们很可能会中断验证例程，而不是改进它们。这就是像这样实践它很重要的原因。

### 使用界限

界限是用于帮助找出值范围的快捷方式。在某种程度上，你已经见过了它：使用插入符号和美元符号来匹配值的开头和末尾。但是，如果你想匹配值内的界限，该怎么做呢？

最清晰的界限出现在单词与非单词之间。这里的“单词”不是*cat*、*month*或*zeitgeist*，而是在\w快捷方式意义上的字母A~Z（包括大写和小写字母）、数字0~9以及下划线。为了把单词用作界限，提供了\b快捷方式。为了把非单词字符用作界限，提供了\B快捷方式。因此，模式\bfor\b将匹配*they've come for you*，但是不匹配*force*或*forebode*；\bfor\B将匹配*force*，但是不匹配*they've come for you*或*informal*。

## 14.5 查找所有匹配

回到Perl兼容的正则表达式使用的PHP函数上来，使用preg\_match()只是为了查看模式是否匹配值。但是脚本没有（准确地）报告值中的什么内容确实匹配模式。可以通过把一个变量用作该函数的第三个参数来查明该信息：

```
preg_match(pattern, subject, $match);
```

\$match变量将包含发现的第一个匹配（由于该函数只会返回值中的第一个匹配）。为了查找所有的匹配，可以使用preg\_match\_all()。其语法是相同的：

```
preg_match_all(pattern, subject, $matches);
```

该函数将返回产生匹配的次数，如果没有找到一个匹配，则返回FALSE。它还将把产生的每个匹配赋予\$matches。更新PHP脚本以打印出返回的匹配，然后运行另外几个测试。

### 报告所有的匹配

- (1) 在文本编辑器或IDE中打开pcre.php（参见脚本14-1）。
- (2) 将preg\_match()的调用更改如下（参见脚本14-2）。

**脚本 14-2** 为了准确呈现字符串中的什么值与哪些模式匹配，脚本的这个修订版本将打印出每个匹配。可以通过将一个变量指定为 `preg_match()` 或 `preg_match_all()` 中的第三个参数来获取匹配

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6 </head>
7 <body>
8 <?php // Script 14.2 - matches.php
9 // This script takes a submitted string and checks it against a submitted pattern.
10 // This version prints every match made.
11
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13
14 // Trim the strings:
15 $pattern = trim($_POST['pattern']);
16 $subject = trim($_POST['subject']);
17
18 // Print a caption:
19 echo "<p>The result of checking
$pattern
against
$subject
is ";
20
21 // Test:
22 if (preg_match_all ($pattern, $subject, $matches)) {
23 echo 'TRUE!</p>';
24
25 // Print the matches:
26 echo '<pre>' . print_r($matches, 1) . '</pre>';
27
28 } else {
29 echo 'FALSE!</p>';
30 }
31
32 } // End of submission IF.
33 // Display the HTML form.
34 ?>
35 <form action="matches.php" method="post">
36 <p>Regular Expression Pattern: <input type="text" name="pattern" value=<?php if (isset
37 ($pattern)) echo htmlentities($pattern); ?>" size="40" /> (include the delimiters)</p>
38 <p>Test Subject: <textarea name="subject" rows="5" cols="40"><?php if (isset($subject)) echo
39 htmlentities($subject); ?> </textarea></p>
40 <input type="submit" name="submit" value="Test!" />
41 </form>
42 </body>
43 </html>
```

这里有两个变化。第一，将调用的实际函数是不同的。第二，给第三个参数提供一个变量名称，将把每个匹配赋予该变量。

(3) 在打印值TRUE之后，打印\$matches的内容。

```
echo '<pre>' . print_r($matches, 1) . '</pre>';
```

使用print\_r()来输出变量的内容是查看\$matches中有什么的最容易的方法（你也可以使用foreach循环来代替）。在运行这个脚本时将会看到，这个变量将是一个数组，它的第一个元素是所产生匹配的数组。

(4) 将表单的action属性更改为matches.php。

```
<form action="matches.php" method="post">
```

重命名这个脚本，使得也必须更改action属性。

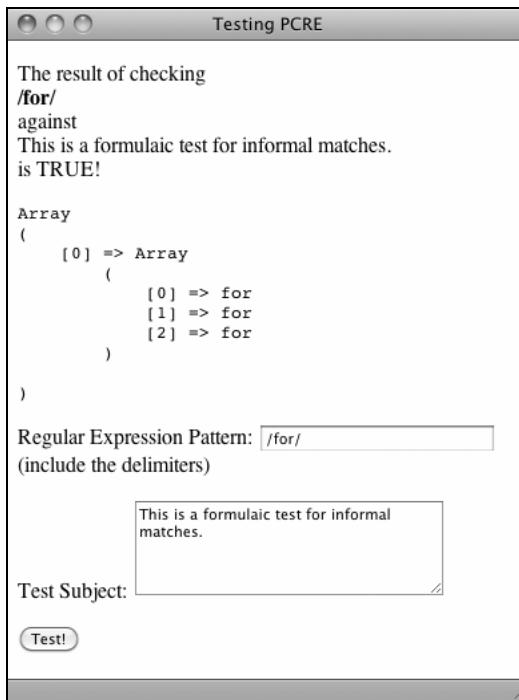
(5) 将主题输入框更改为一个文本区。

```
<p>Test Subject: <textarea name= "subject" rows="5" cols="40"> <?php if (isset($subject))
echo htmlentities($subject); ?> </textarea></p>
```

为了能够为主题输入更多的文本，这个元素将变成一个文本区。

(6) 将文件另存为matches.php，存放在Web目录中，并在Web浏览器中测试它。

对于第一个测试，使用for作为模式，并使用This is a formulaic test for informal matches.作为主题（参见图14-17）。它可能不是正确的英语，但它是一个良好的测试主题。



14

图14-17 第一个测试将返回三个匹配，因为字面量文本for被找到了三次

对于第二个测试，将模式更改为for.\*（参见图14-18）。结果可能会使你感到吃惊，在框注“不要太贪婪”中讨论了其原因。为了使这种查找不到那么贪婪，可以把模式更改为for.\*？，其结果将与图14-17中的那些结果相同。

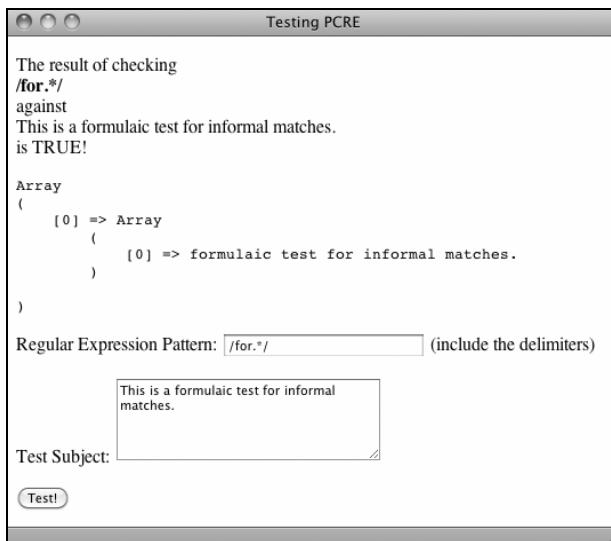


图14-18 由于正则表达式默认是贪婪的（参见框注“不要太贪婪”），这个模式只会在字符串中找到一个匹配。该匹配碰巧以*for*的第一个实例开头，并会延续到字符串的末尾

对于第三个测试，使用`for[\S]*`或者更简单的`for\S*`（参见图14-19）。其作用是：一旦找到空白字符，就停止匹配（因为模式希望匹配`for`，并且其后接着任意数量的非空白字符）。

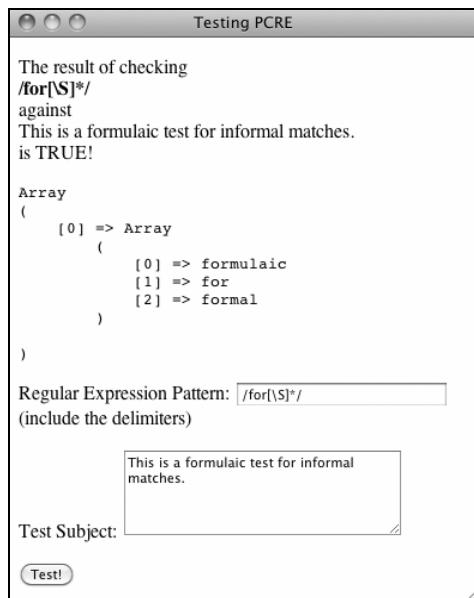


图14-19 这个修改过的模式将匹配以*for*开头并在一个单词上结束的字符串

对于最后一个测试，使用`\b[a-z]*for[a-z]*\b`作为模式（参见图14-20）。这个模式利用了界限，在本章前面的框注“使用界限”中讨论了它。

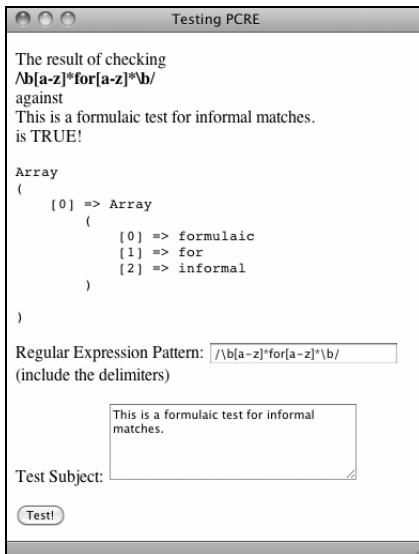


图14-20 与图14-19中的模式不同，这个模式匹配包含`for`的完整单词（这里的*informal*、图14-19中的*formal*）

### ✓ 提示

- `preg_split()`函数将获取一个字符串，并使用正则表达式模式将其分解到一个数组中。

### 不要太贪婪

Perl兼容的正则表达式的关键成分是贪婪（greediness）的概念，在POSIX中则没有这个概念。默认情况下，PCRE将尝试尽可能多地进行匹配。例如，模式`<.+>`将匹配任何HTML标签。当在像`<a href="page.php"> Link</a>`这样的字符串上执行测试时，它实际上将匹配整个字符串，从开始`<`到结束`>`。不过，这个字符串包含三个可能的匹配：整个字符串、开始标签（从`<a`到`>`）以及结束标签（`</a>`）。

为了克服贪婪，可以使匹配变得懒惰（lazy）。懒惰的匹配将包含尽可能少的数据。可以通过在任意量词后接一个问号使之变懒惰。例如，模式`<.+?>`将返回上述字符串中的两个匹配：开始标签和结束标签。它将不会返回整个字符串作为匹配（这是正则表达式语法的令人混淆的方面之一：相同的字符（这里的问号）可以根据其上下文而具有不同的含义）。

使模式不那么贪婪的另一种方式是使用“非”类别。模式`<[^>]+>`将匹配开始`<`和结束`>`之间的所有内容（结束`>`除外）。因此，使用该模式的效果与使用`<.+?>`相同。该模式还将匹配包含换行符（句点会排除它）的字符串。

## 14.6 使用修饰符

本章中介绍了正则表达式模式中可以使用的绝大多数特殊字符。最后一类特殊字符是模式修饰符。表14-4列出了它们。模式修饰符不同于其他元字符，这是由于它们放在结束定界符之后。

表14-4 这些字符（当置于结束定界符之后时）将改变正则表达式的行为方式

字 符	效 果
A	将模式定位到字符串的开头
i	启用不区分大小写的模式
m	启用多行匹配
s	使句点匹配每个字符，包括换行符
x	忽略大多数空白
U	执行不贪婪的匹配

在这些定界符中，最重要的是*i*，它启用不区分大小写的查找。使用*for*的变体的所有示例（在前面的步骤序列中）都不会匹配单词*For*。不过，/for.\*?/i将是一个匹配。注意：我在该模式中包括了定界符，因为修饰符出现在结束定界符之后。类似地，该序列中的最后一步参考框注“不要太贪婪”，并且指出for.\*?如何执行懒惰查找。/for.\*?/U也是如此。

多行模式也很有趣，这是由于你可以使插入符号和美元符号以不同的方式工作。默认情况下，它们都会对整个值起作用。在多行模式中，插入符号匹配任一行的开头，美元符号则匹配任一行的末尾。

### 使用修饰符

- (1) 如果还没有在浏览器中加载matches.php，就加载它。
- (2) 验证电子邮件地址的列表（参见图14-21）。

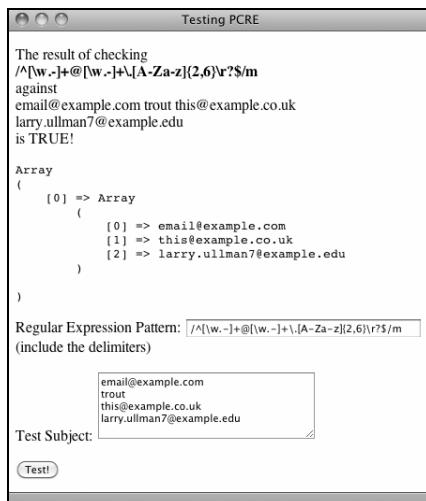


图14-21 可以使用多行模式验证电子邮件地址的列表（每行一个电子邮件地址）。将所有有效的地址存储在\$matches中

为执行该任务，可使用`^[\w{.-}]+@[\\w{.-}]+\.[A-Za-z]{2,6}\r?$/m`作为模式。你将看到我在美元符号之前添加了一个可选的回车符（`\r?`）。这是必要的，因为有些行将包含回车符，而另外一些行则没有包含它。在多行模式中，美元符号匹配一行的末尾。（为了更灵活，可代之以使用`\s?`。）

(3) 验证美国邮政编码的列表（参见图14-22）。

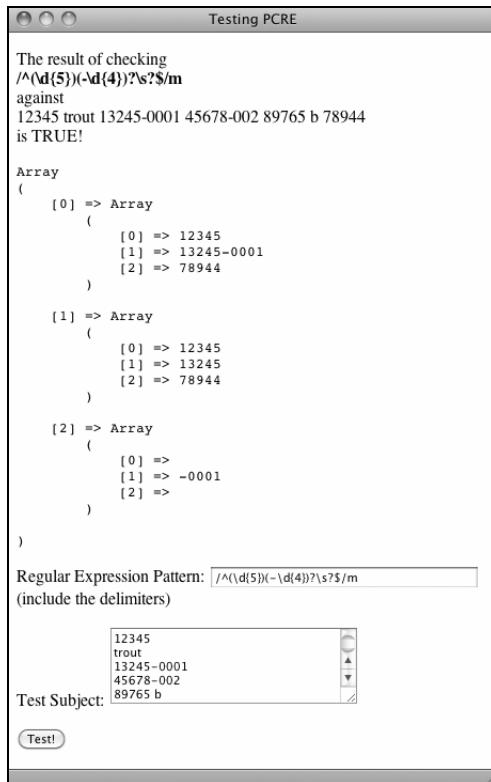


图14-22 验证邮政编码的列表（每行一个邮政编码）

与第(2)步中的示例非常相似，模式现在是`^(\d{5})(-\d{4})?\s?$/m`。你将看到我使用了更灵活的`\s?`代替`\r?`。

当你自己（或者在图14-22中）尝试使用该模式时，你还将注意到`$matches`变量现在包含许多其他的信息。在本章的下一节中将对此做出解释。

#### ✓ 提示

- 不管多行设置如何，为了总是匹配模式的开头和末尾，可以使用一些快捷方式。在模式内，快捷方式`\A`将只匹配值的开头，`\z`匹配值的末尾，`\Z`则会匹配任何行的末尾，就像单行模式中的`$`一样。
- 如果你的PHP版本支持过滤器扩展，最好使用过滤器扩展来验证电子邮箱地址和URL。但是如果你需要验证某种列表，过滤器扩展不会截断它，就需要使用正则表达式。

## 14.7 匹配和替换模式

本章要讨论的最后一个主题是如何匹配和替换值中的模式。虽然preg\_match()和preg\_match\_all()这两个函数将为你查找内容，但是如果你想执行查找和替换，则需要使用preg\_replace()。其语法如下：

```
preg_replace(pattern, replacement, subject);
```

该函数可以带有可选的第四个参数，用于限制执行替换的次数。

要用dog替换cat的所有实例，将使用：

```
$str = preg_replace('/cat/', 'dog', 'I like my cat.');
```

这个函数返回改变过的值（如果没有产生匹配，则返回未改变过的值），因此，你可能想把它赋予一个变量，或者把它用作另一个函数的参数（例如，调用echo()打印它）。此外，提醒一下，这只是一个示例：你永远都不希望使用正则表达式用一个字符串替换另一个字面量字符串，而代之以使用str\_replace()函数。

在此要讨论与这个函数有关的一个概念：后向引用（back referencing）。在邮政编码匹配模式中( $^(\d\{5\})(-\d\{4\})?\$$ )，圆括号内有两个分组（第一个表示必需的前5位数字，第二个表示可选的短划线以及4位数字的扩展）。在正则表达式模式内，PHP将自动从1开始对圆括号内的分组编号。后向引用允许通过使用\$以及相应的编号来引用每个独立的部分。例如，如果把邮政编码94710-0001与这个模式进行匹配，后向引用\$2将得到-0001。代码\$0则会引用完整的初始字符串。这就说明了为什么图14-22在\$matches[0]中显示完整的邮政编码匹配，在\$matches[1]中显示匹配的前5个数字，并在\$matches[2]中显示任何匹配的短划线以及4个数字。

为了实际应用它，让我们修改脚本14-2，使之还接受替换输入（参见图14-23）。

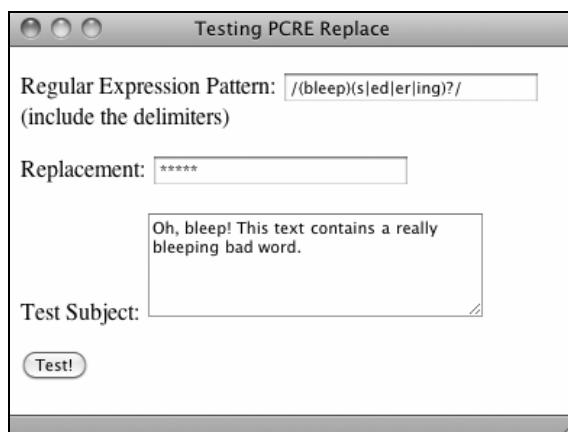


图14-23 使用preg\_replace()用代表省略的符号替换不合适的单词

### 匹配和替换模式

(1) 在文本编辑器或IDE中打开matches.php（参见脚本14-2）。

(2) 添加对第三个传入变量的引用 (参见脚本14-3)。

**脚本 14-3** 为了测试 preg\_replace() 函数 (它用另一个值替换字符串中匹配的模式), 可以使用这个 PCRE 测试脚本的第三个版本

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
 DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Testing PCRE Replace</title>
6 </head>
7 <body>
8 <?php // Script 14.3 - replace.php
9 // This script takes a submitted string and checks it against a submitted pattern.
10 // This version replaces one value with another.
11
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
13
14 // Trim the strings:
15 $pattern = trim($_POST['pattern']);
16 $subject = trim($_POST['subject']);
17 $replace = trim($_POST['replace']);
18
19 // Print a caption:
20 echo "<p>The result of replacing
$pattern
with
$replace
21
in
$subject

";
22
23 // Check for a match:
24 if (preg_match ($pattern, $subject)) {
25 echo preg_replace($pattern, $replace, $subject) . '</p>';
26 } else {
27 echo 'The pattern was not found!</p>';
28 }
29 } // End of submission IF.
30 // Display the HTML form.
31 ?>
32 <form action="replace.php" method="post">
33 <p>Regular Expression Pattern: <input type="text" name="pattern" value="<?php if (isset
34 ($pattern)) echo htmlentities($pattern); ?>" size="40" /> (include the delimiters)</p>
35 <p>Replacement: <input type="text" name="replace" value="<?php if (isset($replace)) echo
36 htmlentities($replace); ?>" size="40" /></p>
37 <p>Test Subject: <textarea name="subject" rows="5" cols="40"><?php if (isset($subject))
38 echo htmlentities($subject); ?></textarea></p>
39 <input type="submit" name="submit" value="Test!" />
40 </form>
41 </body>
42 </html>
```

在图14-23中可以看到，第三个表单输入框（添加在现有的两个表单输入框之间）获取一个替换值。对该值也会进行修剪，以删除任何多余的空格。

如果你的服务器开启了Magic Quotes，这里还需要应用`stripslashes()`。

(3) 更改标题。

```
echo "<p>The result of replacing
$pattern
 with
$replace
in
$subject

";
```

在应用`preg_replace()`之前，该标题将打印出所有传入的值。

(4) 更改正则表达式条件语句，以便在产生匹配时它只会调用`preg_replace()`:

```
if (preg_match ($pattern, $subject)) {
 echo preg_replace($pattern, $replace, $subject) . '</p>';
} else {
 echo 'The pattern was not found!</p>';
}
```

无须首先运行`preg_match()`即可调用`preg_replace()`。如果没有产生匹配，则不会发生替换。但是为了清楚看出何时产生或未产生匹配（确认这一点总是很好，考虑使用正则表达式有多复杂），首先将应用`preg_match()`函数。如果它返回一个真值，那么就调用`preg_replace()`，并打印结果（参见图14-24）。否则，打印一条消息，指示没有产生匹配（参见图14-25）。

(5) 将表单的action属性更改为`replace.php`。

```
<form action="replace.php" method="post">
```

这个文件将被重命名，因此需要相应地更改这个值。

(6) 为替换字符串添加一个文本输入框。

```
<p>Replacement: <input type="text" name="replace" value="<?php if (isset($replace)) echo htmlentities
($replace); ?>" size="40" /></p>
```

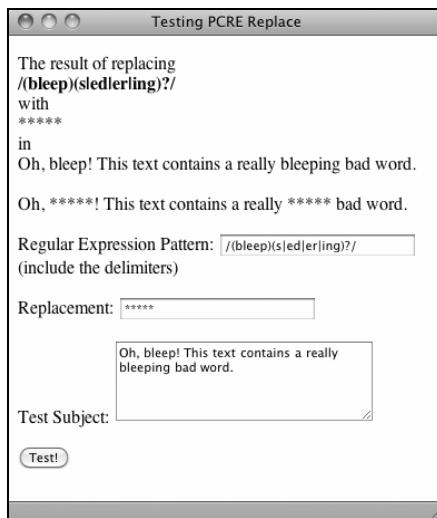


图14-24 得到的文本用\*\*\*\*\*替换了bleep、bleeps、bleped、bleeper和bleeping……

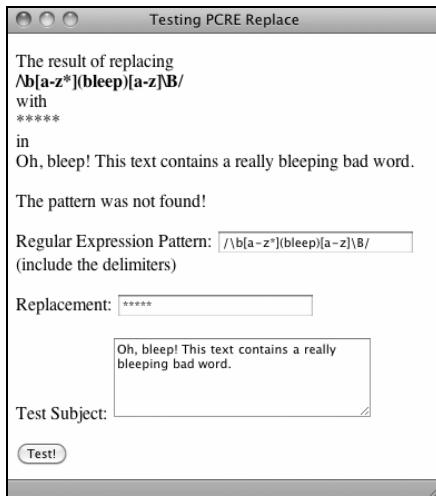
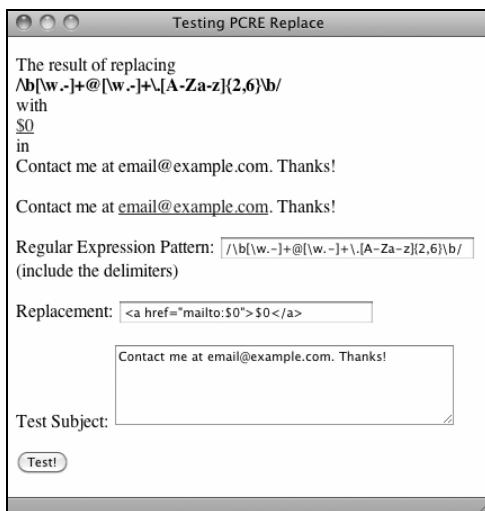


图14-25 如果在主题中没有找到模式，将不会更改主题

(7) 将文件另存为replace.php，存放在Web目录中，并在Web浏览器中测试它（参见图14-26）。

图14-26 preg\_replace()的另一种应用是动态地把电子邮件地址参见  
HTML源代码了解替换的全部效果

作为一个良好的示例，可以把在某段文本内找到的电子邮件地址转换成等价的HTML链接：`<ahref="mailto:email- @example.com" > email@example.com</a>`。目前应该很熟悉用于匹配电子邮件地址的模式：`^[\w.-]+@[ \w.-]+\.[A-Za-z]{2,6}$`。不过，由于可能在某段文本内找到电子邮件地址，需要用单词界限快捷方式（\b）替换插入符号和美元符号。因此，最终的模式将是：`\b[\w.-]+@[ \w.-]+\.[A-Za-z]{2,6}\b`。

为了引用这个匹配的电子邮件地址，可以引用\$0（因为\$0引用整个匹配，而不管是否使用圆括号）。因此，替换值将是：`<a href="mailto:$0">$0 </a>`。由于这里涉及HTML，查看结果页面的HTML源代码，可以最好地了解发生了什么事情。

#### ✓ 提示

- 甚至可以在模式内使用后向引用。例如，如果模式包括一个将会重复的分组（即子模式）。
- 我在这里有些快地介绍了大量的PCRE语法，但是还有更多的语法没有介绍。一旦你完全掌握了它们，可以考虑开始学习锚点、命名的子模式、注释、环视（lookaround）、物主量词（possessive quantifier）等内容。

## 14.8 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书的论坛上，我们会为你答疑解惑。

### 14.8.1 回顾

- 匹配正则表达式的函数是？用于找出正则表达式所有匹配的函数是？用来替换正则表达式匹配的函数是？
- 在描述正则表达式时，可以使用或不能使用那些字符？
- 你该如何匹配字符或字符串？
- 什么是元字符？你该如何转码一个元字符？
- 要绑定一个模式到一个字符串的开始需要用什么元字符？到结尾呢？
- 如何创建子模式（即分组）？
- 什么是量词？如何请求0或1个字符或字符串？0次或多次？1次或多次？正好出现X次？出现范围？出现的最小值？
- 什么是字符类别？
- 哪些元字符仍然是字符类别中的意义？
- 代表“任意数字”的字符类别是什么？“任何空白”类别呢？“任何字”呢？代表与这些相反的快捷方式是？
- 什么是界限？如何在模式中创建界限？
- 如何让匹配变得“懒惰”？这是什么意思呢？
- 什么是模式修饰符？
- 什么是后向引用？它是如何工作的？

### 14.8.2 实践

- 在线搜索PCRE“速查表”（PHP或其他方式），列出所有有意义的字符和类。打印速查表，放在你的电脑旁。
- 实践，实践，再实践！

## 第 15 章

# jQuery简介

# 15

### 本章内容

- jQuery是什么
- 包含jQuery
- 使用jQuery
- 选择页面元素
- 事件处理
- DOM操作
- 使用Ajax
- 回顾和实践

**本**章是本书这一版的新增内容，介绍了jQuery这个JavaScript框架。在过去的十年中JavaScript这一语言的影响力越来越大，由于其用法多样，目前的很多Web站点都采用它。因此，很多PHP开发人员都想了解一点JavaScript。

本章不可能全面介绍JavaScript或jQuery，但你会学到足够多的内容，为你的基于PHP的项目添加用户期望的效果。在这个过程中，你还将学习常见的JavaScript编程的一些基础知识，并引导你继续进一步学习jQuery。

### 15.1 jQuery 是什么

要掌握jQuery，你必须先了解JavaScript。正如在第11章中讨论的，JavaScript是一种主要用于向Web页面添加动态功能的编程语言。它不像总是运行在服务器端的PHP，JavaScript通常运行在客户端（JavaScript也已开始用于服务器端工具，但并非主流）。因为PHP是服务器端的语言，所以大部分内容与浏览器无关：PHP中只有很少一点东西会在不同的浏览器中有不同的表现。正相反，由于JavaScript运行在Web浏览器中，JavaScript代码经常需要为不同的浏览器做定制的改变。创建可靠的跨浏览器代码，多年来一直是Web开发人员的梦魇。克服这一障碍是jQuery的众多优点之一([www.jquery.com](http://www.jquery.com), 参见图15-1)。

jQuery是一个JavaScript框架，是用来加快并简化开发的代码库。本章你会学到，jQuery框架的核心目标是能够处理所有关键的JavaScript功能。但为了提供其他功能，这个框架是通过插件扩展的（如创建动态的、分页的、可排序的数据表格的功能）。jQuery的官方界面库——jQuery UI中包含了很多有用的用户界面工具 ([www.jqueryui.com](http://www.jqueryui.com), 参见图15-2)。

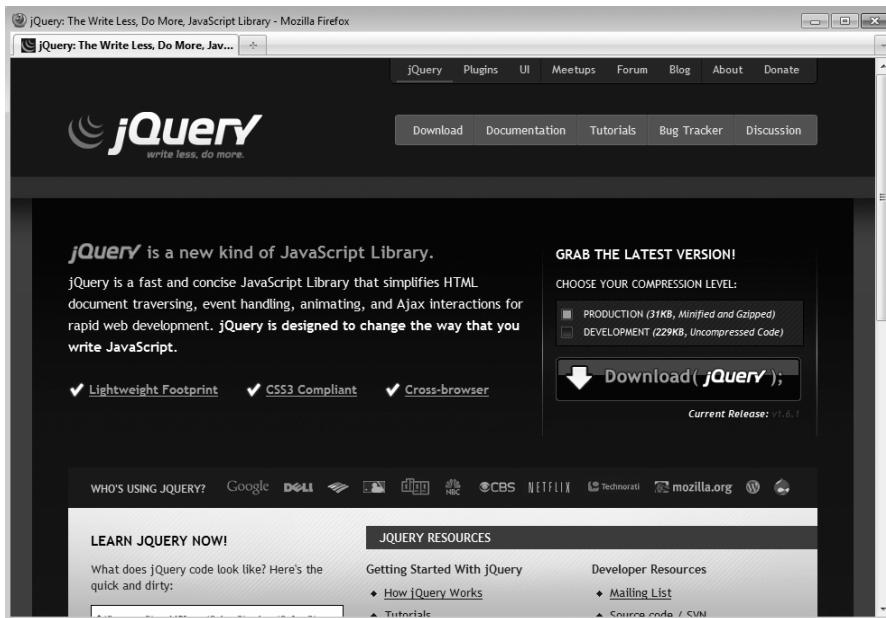


图15-1 jQuery JavaScript框架的首页



图15-2 与jQuery结合使用的jQuery用户界面库 (jQuery UI) 首页

目前有很多的JavaScript框架，但我想说jQuery是最好的一个。我确实非常喜欢jQuery。作为最大的JavaScript框架之一，jQuery很快赢得了一席之地。你很快就会看到，jQuery的语法简单而神奇，你可以使用它轻松自如地操纵文档对象模型（DOM）。也就是说，你可以很容易地引用HTML页面中的元素，从而获取表单文本框的值，添加或删除任何类型的HTML元素，更改元素的属性，等等。

在了解jQuery的具体使用方法之前，你需要知道jQuery仅是一个JavaScript框架，这意味着你在接下来的几页中实际上做的是JavaScript编程。JavaScript语言，在某些方面与PHP类似，但又有一些不同（比如创建变量的方式，执行连接操作使用的字符等）。此外，JavaScript是一种面向对象的脚本语言，这意味着你有时候看到的语法会与你现在写的过程式的PHP程序有较大的不同（下章会介绍PHP面向对象编程）。你需要了解一下JavaScript调试的内容，因为你将不可避免的碰到问题，如遗漏了分号。参见框注“调试JavaScript”，查看关于这一主题的简要介绍。

对于服务器端JavaScript的例子，查看Node ([www.nodejs.org](http://www.nodejs.org)) 或Jaxer ([www.jaxer.org](http://www.jaxer.org))。

✓ 提示

- 我已经写了一系列关于jQuery技术的文章，涵盖了本章的许多主题。你可以在我的网站上找到这些内容。

### 调试JavaScript

此时，你可能想象不到PHP将所有错误都放进浏览器中是多么的美妙。直到此刻。当Web页面有JavaScript错误时，你很少会收到通知。为了调试有问题的JavaScript代码，你首先要做的就是查看实际发生了什么错误。

当使用JavaScript进行编程时，你首先需要的工具就是一个好的Web浏览器。它应该是一个有助于开发者的浏览器，而不应该是有助于用户的浏览器。FireFox ([www.mozilla.com](http://www.mozilla.com)) 在这方面是当之无愧的冠军，但Opera ([www.opera.com](http://www.opera.com)) 和Google Chrome ([www.google.com/chrome/](http://www.google.com/chrome/)) 可能也非常不错。通过打开Firefox内建的错误控制台（在“工具”菜单下），你可以看到任何JavaScript发生的错误。

相对于其他浏览器来说，FireFox有一个特有的优势，即它历史悠久的优秀的第三方扩展。特别是，Firebug和Web Developer是理解网页发生了什么的梦幻工具。同样还有一些扩展，诸如FireQuery，用于向FireFox添加jQuery开发工具。

在本章开始之前，我建议你下载并安装最新版本的Firefox和前面提到的扩展（如果你还没有的话）。在Firefox中运行启用JavaScript的网页就可以很容易地查看发生的错误，从而使得它更容易调试。

15

## 15.2 包含 jQuery

JavaScript默认包含在所有的图形化的Web浏览器中，这意味着不需要任何特殊步骤就能在网页中包含JavaScript（用户可以选择禁用JavaScript，但根据统计很少有这么做的）。而jQuery是一个代码框架，为了使用它，网页必须首先包含jQuery库。要在网页中包含外部的JavaScript文件，需要使用HTML的script标签，在它的src属性中添加外部文件的名字：

```
<script src="file.js" type="text/javascript" charset="utf-8"></script>
```

jQuery框架文件的名字类似jquery-X.Y.Zmin.js，其中X.Y.Z是版本号（在写本书时是1.6.1）。文件名的min部分指明JavaScript文件是经过压缩的。最小化就是移除代码的空格、换行和注释。结果将是难以阅读但是功能完备的代码（参见图15-3）。最小化代码的好处是它可以使页面在浏览器中更快地载入，因为它的文件会很小。

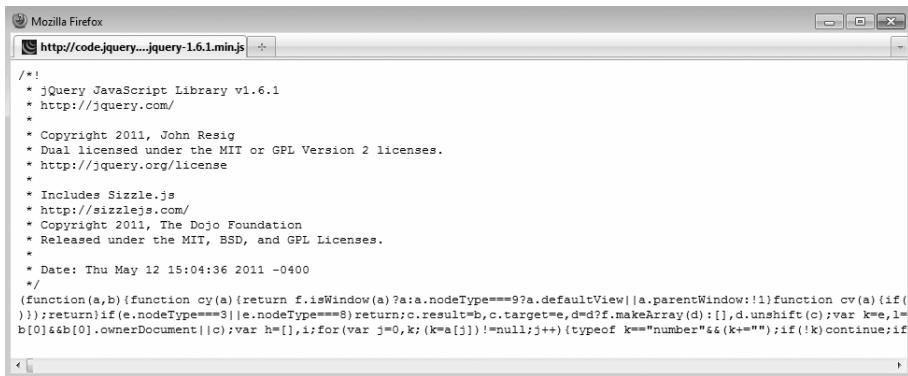


图15-3 压缩后的jQuery代码的样子

下面的一系列步骤将引导你在Web服务器上安装jQuery库，并包含到网页中；另一种方法请参见框注“使用托管的jQuery”。

### 使用托管的jQuery

本章建议你下载一个jQuery的副本，并将其放置在你的网站目录。在这种情况下，你只需要更新script标签指向网站上的jQuery文件的位置即可。不过我想给出一个替代的解决方案：使用托管版本的jQuery。我的意思是，不使用存储在你的网站服务器上的jQuery版本，而是使用一个存储在别的地方的在线版本。例如，Google为公众使用提供了很多JavaScript框架的副本（<http://code.google.com/apis/libraries/>）。要使用Google的jQuery库副本，代码如下：

```
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js"
charset="utf-8"></script>
```

由于谷歌的内容分发网络（CDN）和浏览器缓存媒体方式，通过使用Google托管版本的jQuery，你的网站（例如，你网站的访问者）的性能可能会提升。本章中不会使用Google提供的jQuery，因为不是所有的读者都能连接互联网，而且了解如何使用本地JavaScript文件是一项重要的技能。

### 包含jQuery

- (1) 在你的Web浏览器中载入[www.jquery.com](http://www.jquery.com)。
- (2) 在该页面的顶部，选择PRODUCTION（这是默认选项），单击“下载”（参见图15-4）。由于结果文件就是JavaScript，它会在你的Web浏览器中直接加载（参见图15-3）。



图15-4 下载当前版本jQuery的表单

(3) 将页面保存到电脑。

大多数浏览器提供了一个保存、另存为、或页面另存为的选项。将文件保存为jquery-X.Y.Z.min.js, X.Y.Z.min.js是实际的版本号。

(4) 将下载的文件移动到Web服务器目录中的js文件夹。

本章例子使用到的所有JavaScript文件都将放置在js的子目录中。

(5) 在文本编辑器或IDE中创建名为test.html的新HTML文档（参见脚本15-1）。

#### 脚本 15-1 这个空的 HTML 页面展示了如何包括 jQuery 库

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Testing jQuery</title>
6 <script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
7 </head>
8 <body>
9 <!-- Script 15.1 - test.html -->
10 </body>
11 </html>
```

上述示例将会测试包含和使用jQuery库。

(6) 在HTML头中包含jQuery。

```
<script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
```

script标签用于包含一个JavaScript文件。习惯上把script标签放在HTML页面的头部，尽管这不是必需的（但一般如此）。src属性的值需要匹配jQuery库的名称和位置。本例假设HTML页面与js文件夹在相同的目录中，该目录已在步骤(4)中创建。

(7) 将文件保存为test.html。

因为这个脚本将不会执行任何PHP，所以使用.html扩展名。

(8) 如果你愿意，在Web浏览器中载入页面并检查错误。

由于这仅仅是一个HTML页面，你可以不通过URL，直接在Web浏览器中加载它。然后，你可以

使用浏览器中的错误控制台或其他开发工具(请参阅“调试JavaScript”框注),检查在加载的JavaScript文件中有没有错误发生。

## 15.3 使用jQuery

一旦你成功将jQuery包含到一个网页,就可以开始使用它了。jQuery或任何JavaScript代码,要写在开始和结束的script标签之间:

```
<script type="text/javascript">
// JavaScript goes here.
</script>
```

(注意,在JavaScript中,用双斜线创建注释,就像在PHP中)。

或者,你也可以把jQuery或JavaScript代码放到单独的文件中,然后使用script标签包含那个文件,就像你包含jQuery库一样。这是为了以后将JavaScript从HTML中分离,本章中会使用这个方法。

需要明确的是,一个网页可以使用多个script标签,但同一个script标签不能既包含外部文件又包含JavaScript代码。

在script标签内部的代码会在浏览器遇到它的时候立即执行。但是这往往会出现问题,因为JavaScript经常被用来与DOM交互:如果立即执行的JavaScript代码引用了一个DOM元素,该代码将失败,因为在那个时候浏览器尚未遇到那个DOM元素(参见图15-5)。唯一的可靠地引用DOM元素的方法是在浏览器已经知道了整个DOM之后引用。



图15-5 浏览器读取一个网页作为载入的HTML,意味着JavaScript代码在浏览器看到所有的内容前不能引用DOM元素

在标准的JavaScript中,你可以在window.onload中引用代码,让代码在页面完全加载后执行。在jQuery中,首选的方法是确认Web文档加载完成:

```
$(document).ready(some_function);
```

如前所述,jQuery的语法对于初学者来说看起来特别奇怪,所以我会详细解释。首先,代码\$(something)是jQuery在Web浏览器中选择元素的方式。在本例中,被选中的项目是整个Web文档。

`ready()`函数被应用于这个选择对象。它接受一个参数——被调用的函数。需要注意的是该参数是一个函数的引用——函数的名字，不带引号。此外，`some_function()`必须在实际工作发生的地方定义，也就是当文档加载完成时。

替代语法是使用匿名函数，即没有名字的函数定义。匿名函数对JavaScript比较常见（PHP中也可以使用匿名函数，但不常见）。要创建匿名函数，需要内联地定义函数，而不用特定的函数名。

```
$(document).ready(function() {
 // Function code.
});
```

因为通常需要在浏览器就绪后执行函数，所以在jQuery中整个构造经常被简化为：

```
$(function() {
 // Function code.
});
```

该语法不常见，特别是最后的`)`。你在编程的时候要留心。和其他编程语言一样，不正确的JavaScript语法会让代码无法正常工作。

要测试的jQuery，下一个系列步骤将在文档就绪后创建JavaScript警告（参见图15-6）。在简单的测试工作完成后，你就可以安全地使用jQuery做些更实际的事了。

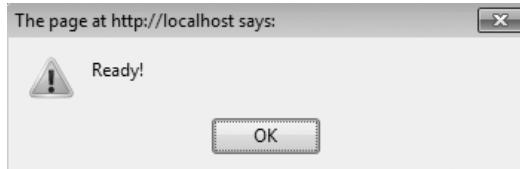


图15-6 这个JavaScript警告会在jQuery识别出Web文档已经在浏览器中时被创建

## 使用jQuery

(1) 在文本编辑器或IDE中创建名为test.js的新JavaScript文件（参见脚本15-2）。

### 脚本 15-2 这个简单的 JavaScript 文件创建警告框来测试是否正确地包含和使用 jQuery 库

```
1 // Script 15.2 - test.js
2 // This script is included by test.html.
3 // This script just creates an alert to test jQuery.
4
5 // Do something when the document is ready:
6 $(function() {
7
8 // Alert!
9 alert('Ready!');
10
11});
```

JavaScript脚本文件中没有HTML文档中的`script`标签，或其他的起始标签。你可以直接开始输入JavaScript代码。重申一次，使用双斜线创建注释。

(2) 当文档就绪时创建警告。

```
$(function() {
 alert('Ready!');
});
```

上面的语法已经解释过，在函数代码中调用`alert()`，如前面的注释所述。`alert()`函数接受一个字符串作为其参数，将显示在弹出的警告框中（参见图15-6）。

(3) 将文件保存为`test.js`，放到Web服务器的js目录中。

(4) 在文本编辑器或IDE中打开`test.html`文件（参见脚本15-1）。

下一个步骤是更新的HTML页面，它包含了新的JavaScript文件。

(5) 在包含jQuery库后，包含新的JavaScript文件（参见脚本15-3）。

### 脚本 15-3 更新的测试 HTML 页载入了新的 JavaScript 文件

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Testing jQuery</title>
6 <script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
7 <script type="text/javascript" src="js/test.js" charset="utf-8"></script>
8 </head>
9 <body>
10 <!-- Script 15.3 - test.html #2 -->
11 </body>
12 </html>
```

假设JavaScript文件`test.js`与jQuery库放置在同一个目录中，对于`test.html`，两个文件的相对地址相同，那么这段代码将成功地包含它。

(6) 保存HTML页面，并在Web浏览器中测试（参见图15-6）。

如果你没有看到警告窗口，就需要调试JavaScript代码。

#### ✓ 提示

- 从技术上讲，在面向对象编程中，函数被称为方法。在这一章中，我将继续使用“函数”，因为你可能对它更熟悉。
- 代码`$()`是调用`jQuery()`函数的简写。
- jQuery的“就绪”状态与JavaScript的`onLoad`略有不同：后者还会等待图片和其他媒体载入，而jQuery的就绪状态会在DOM加载完成后触发。

## 15.4 选择页面元素

如果基本jQuery功能一切正常，那接下来就可以学习如何选取页面元素了。具体来说，就是可以隐藏或显示图片或文本块、操作表单，等等。

你已经看到了如何选择Web文档自身：`$(document)`。要选择其他的页面元素，使用CSS选择器代替`document`：

- `#something`选择id值为`something`的元素；

`.something`选择所有`class`值为`something`的元素；

`something`选择所有`something`类型的元素（例如，`p`选择所有的段落）。

这三个规则对于初学者来说已经是绰绰有余了，但是你要知道，与`document`不同这处是，这些都需要放到引号中。例如，代码`$('a')`选择的是每一个链接、`$("#caption")`选择`id`值为`caption`的元素。根据定义，在一个Web页面中没有两个元素具有相同的标识值（`id`），因而`#something`是引用页面上单独元素最简单的解决方案。

这些规则也可以结合使用：

`$("img.landscapE")`选择所有`class`为`landscapE`的图片；

`$('#register input')`选择`id`为`register`的元素内部的所有文本输入框元素。

在接下来的jQuery的示例中，将开发一个与第13章中的相同成本计算表单，但这个版本是由JavaScript小部件实现的。在接下来的几个步骤中，将创建HTML页面，包括相应的元素、`class`和唯一标识符以便于jQuery操作（参见图15-7）。

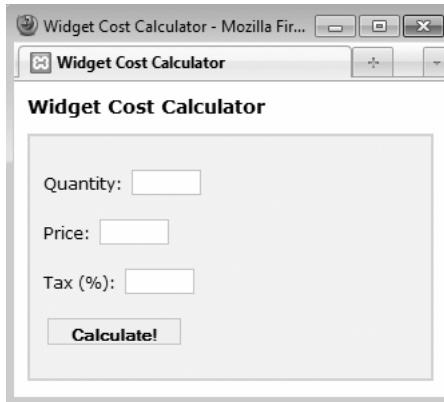


图15-7 成本计算器部件作为HTML表单

### 创建HTML表单

(1) 在文本编辑器或IDE中打开`test.html`（参见脚本15-3）。

因为这个文件已经包含jQuery，仅更新它将会非常容易。

(2) 更改页面的标题（参见脚本15-4）。

15

#### 脚本 15-4 在这个 HTML 页面中有一个表单，其中包含执行计算的三个文本输入框

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Widget Cost Calculator</title>
6 <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />
7 <script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
8 <script type="text/javascript" src="js/calculator.js" charset="utf-8"></script>
```

```

9 </head>
10 <body>
11 <!-- Script 15.4 - calculator.html -->
12 <h1>Widget Cost Calculator</h1>
13 <p id="results"></p>
14 <form action="calculator.php" method="post" id="calculator">
15 <p id="quantityP">Quantity: <input type="text" name="quantity" id="quantity" /></p>
16 <p id="priceP">Price: <input type="text" name="price" id="price" /></p>
17 <p id="taxP">Tax (%): <input type="text" name="tax" id="tax" /></p>
18 <p><input type="submit" name="submit" id="submit" value="Calculate!" /></p>
19 </form>
20 </body>
21 </html>

```

(3) 在页面的标题后，包含CSS文件。

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

以下CSS代码将会格式化表单，将它变得更加引人注目。你可以在本书的网站上下载这个CSS文件。

CSS文件还定义了两个重要的类，`error`和`errorMessage`，在本章的后面通过jQuery进行操作。第一个类把一切都变成红色，第二个把文字变为斜体（这个`class`对于表示一组相同的项目更有意义）。随着时间的推移，你会看到如何使用这些类。

(4) 更改第二个script标签，让它引用calculator.js而不是test.js。

```
<script type="text/javascript" src="js/calculator.js" charset="utf-8"></script>
```

稍后，会将本例中的JavaScript编写在calculator.js中。它会与其他的JavaScript文档一样存储在js文件夹下。

(5) 在HTML主体中，添加一个空段落并创建表单。

```

<h1>Widget Cost Calculator</h1>
<p id="results"></p>
<form action="calculator.php" method="post" id="calculator">

```

该段落有一个为`results`的id但是没有内容，稍后将用于显示本章计算的结果。它有一个唯一的id值，以便于引用。表单也有一个唯一的id值。从理论上讲，表单将提交到calculator.php（一个单独的脚本，不是在本章中编写的），但是提交将会被JavaScript中断。

(6) 创建第一个表单元素。

```
<p id="quantityP">Quantity: <input type="text" name="quantity" id="quantity" /></p>
```

第13章中编写的每个表单输入框，包含元素本身的文字提示，并且被一个段落符号包围。为段落符和表单输入框添加唯一的id值。

注意，我倾向于在面向对象的语言（如JavaScript中）使用“驼峰”式名字（`quantity_P`）。这么做只为更好地遵循OOP惯例（但我会在面向过程的PHP代码中使用`quantity_p`）。

(7) 创建剩下的两个表单元素。

```

<p id="priceP">Price: <input type="text" name="price" id="price" /></p>
<p id="taxP">Tax (%): <input type="text" name="tax" id="tax" /></p>

```

(8) 完成表单和HTML页面。

```
<p><input type="submit" name="submit" id="submit" value="Calculate!" /></p>
</form>
</body>
</html>
```

提交按钮也有一个唯一的id，但是这只用于CSS中；实际上不会在JavaScript中引用。

(9) 将页面存储为calculator.html，并在Web浏览器中载入（参见图15-7）。

即使第二个JavaScript文件calculator.js尚未编写，表单仍然可以被载入。

#### ✓ 提示

- jQuery有其自己的自定义选择符，支持更复杂的方式选取页面元素。具体示例请查看jQuery的手册。

## 15.5 事件处理

和PHP类似，JavaScript也经常用于响应事件。不同的是，JavaScript中的事件主要是指用户在Web浏览器内的操作，如：

- 将光标移到图片或一段文本上；
- 点击一个链接；
- 更改表单元素的值；
- 提交表单。

要在JavaScript中处理事件，你需要对元素应用事件监听器（也称为事件处理器），也就是说，告诉JavaScript，当B元素发生一个事件A时，应该调用C函数。在jQuery中，使用以下语法分配事件监听器

```
selection.eventType(function);
```

选择器部分应该像`'$.something'`或`('a')`，即任何事件监听器应该应用的一个或多个元素。事件类型的值会基于选取的内容有所不同。共同的值是`change`、`focus`、`mouseover`、`click`、`submit`和`select`——不同的事件会被不同的HTML元素触发。在jQuery中，这些都是会被选择器调用的实际的函数名。这些函数接受一个参数：当选择的内容发生特定事件时调用的函数。通常情况下，被调用的函数是以内联方式编写的匿名函数。例如，处理任何图像的`mouseover`事件，可以这样写：

```
$(‘img’).mouseover(function() {
 // Do this!
});
```

这种结构看起来应该很熟悉，类似`test.js`为Web文档的`ready`事件指定事件处理器。

下面使用这个新的信息并将它应用到前面创建的HTML页面中。现在，为表单添加一个事件监听器，以便处理提交。在这个特定例子中，表单中三个文本输入框将进行最基本的验证，执行计算，并将计算的结果显示在警告框中（参见图15-8）。要实现这些，你还需要知道一件事情：要获取文本表单输入的值，需要使用`val()`函数。它返回选择内容的值，你将在接下来的步骤中看到它的用法。

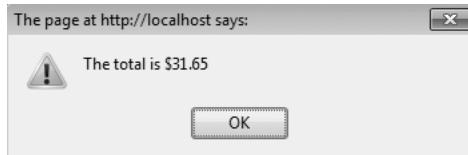


图15-8 计算结果用警告框来显示（目前）

### 处理表单提交

(1) 在文本编辑器或IDE中打开test.js。

由于test.js文件已经有了当浏览器就绪时执行代码的正确语法,以它开始将非常简单且万无一失。

(2) 删除现有的alert()调用(参见脚本15-5)。

**脚本 15-5** 此 JavaScript 文件被包含在 calculator.html 中(参见脚本 15-4)。在提交表单时, 表单的值是有效的且执行了计算

```

1 // Script 15.5 - calculator.js
2 // This script is included by calculator.html.
3 // This script handles and validates the form submission.
4
5 // Do something when the document is ready:
6 $(function() {
7
8 // Assign an event handler to the form:
9 $('#calculator').submit(function() {
10
11 // Initialize some variables:
12 var quantity, price, tax, total;
13
14 // Validate the quantity:
15 if ($('#quantity').val() > 0) {
16
17 // Get the quantity:
18 quantity = $('#quantity').val();
19
20 } else { // Invalid quantity!
21
22 // Alert the user:
23 alert('Please enter a valid quantity!');
24
25 }
26
27 // Validate the price:
28 if ($('#price').val() > 0) {
29 price = $('#price').val();
30 } else {
31 alert('Please enter a valid price!');
32 }
33
34 // Validate the tax:
35 if ($('#tax').val() > 0) {

```

```

36 tax = $('#tax').val();
37 } else {
38 alert('Please enter a valid tax!');
39 }
40
41 // If appropriate, perform the calculations:
42 if (quantity && price && tax) {
43
44 total = quantity * price;
45 total += total * (tax/100);
46
47 // Display the results:
48 alert('The total is $' + total);
49
50 }
51
52 // Return false to prevent an actual form submission:
53 return false;
54
55}); // End of form submission.
56
57}); // End of document ready.

```

用下面所有的代码代替原来alert()。

(3) 为表单提交添加事件处理器来代替alert()调用。

```

$('#calculator').submit(function() {
}); // End of form submission.

```

选择器获取了id值为calculator的表单的一个引用，对选择内容应用submit()函数。因此，当表单提交时，将调用内联的匿名函数。由于语法比较复杂，我建议先添加这块代码，然后再编写匿名函数的内容，这可以在接下来的步骤中看到。注意，这些代码和接下来所有代码都要放到\$(document).ready() {}代码块中。

(4) 在新的匿名函数中，初始化4个变量。

```

var quantity, price, tax, total;

```

在JavaScript中，使用var关键字来声明变量。用逗号分隔可以一次声明多个变量。注意，在JavaScript中的变量没有初始的\$符号（与PHP不同）。

(5) 验证数量。

```

if ($('#quantity').val() > 0) {
 quantity = $('#quantity').val();
} else {
 alert('Please enter a valid quantity!');
}

```

对于这三个表单输入框，都需要有一个大于0的数字值。输入的值可以通过在选择的元素上调用的val()函数获取。如果返回的值大于0，则该值被分配给局部变量quantity，否则会弹出警告框，向用户指明问题（参见图15-9）。使用警告框确实有点单调乏味，你会在本章后面的小节中学习一个不唐突的方法。



图15-9 如果表单元素的值不是正数，会显示警告框指明错误

(6) 为其他的两个表单输入重复验证过程。

```
if ($('#price').val() > 0) {
 price = $('#price').val();
} else {
 alert('Please enter a valid price!');
}
if ($('#tax').val() > 0) {
 tax = $('#tax').val();
} else {
 alert('Please enter a valid tax!');
}
```

(7) 如果这3个变量的值都有效，就进行计算。

```
if (quantity && price && tax) {
 total = quantity * price;
 total += total * (tax/100);
```

这段代码并无新意，它看起来几乎与PHP一模一样，只是在每个变量的名字前面省略了\$符。

(8) 报告总数。

```
alert('The total is $' + total);
```

它仍使用了唐突的警告窗口来显示计算的结果（参见图15-8）。你可以在代码中看到，在JavaScript中使用加号执行了连结操作。

(9) 完成在第(7)步开始的条件语句，返回false值。

```
}
return false;
```

让函数返回false，防止表单提交到表单的action指定的脚本。

(10) 将页面保存为calculator.js（在js文件夹中），并在Web浏览器中测试计算功能。

需要注意的是，如果你已经在Web浏览器中载入了calculator.html，就需要刷新浏览器来载入更新后的JavaScript。

### ✓ 提示

- 有很多专门用于表单验证的jQuery插件，但是我希望保留这个样本（在此过程中解释JavaScript的核心概念）。
- 在JavaScript是可以格式化数字的，例如，让它们总是保留两位小数，但是这不是很容易实现。因为我不想影响涉及的更重要的信息，所以计算结果的样式可能不尽人意。
- 使用jQuery，如果浏览器支持它，JavaScript代码将会执行计算。如果用户禁用了JavaScript，或者如果用户的浏览器很“老”，那JavaScript将无法生效，表格会按平常一样提交（提交到这里并不存在的calculator.php）。

## 15.6 DOM 操作

通常JavaScript和jQuery最重要的功能之一是操纵DOM：以任何方式更改Web浏览器的内容。通常状况下，DOM操作是改变用户看到的内容；你可以使用jQuery轻而易举地实现这个功能，这也是它的优势之一。

一旦你选择了一个或多个要进行操作的元素，无论为选中的内容应用几个jQuery函数都会改变它的属性。对于初学者来说，`hide()`和`show()`函数……嗯……隐藏和显示选中的内容。因此，要隐藏表单（也许是在用户已经成功地完成了它），可以使用：

```
$('#actualFormId').hide();
```

当调用`toggle()`函数时，将隐藏可见元素，或显示隐藏的元素（即，在两种状态间切换）。注意，这些功能既不会创建也不会销毁选中的内容（即，选中的内容仍然是DOM的一部分，无论它是否可见）。

与`show()`和`hide()`相似的是`fadeIn()`和`fadeOut()`。这些函数也可以显示或隐藏选中的内容，但是添加了一点效果。

影响DOM的另外一个方法是，改变应用到选中内容的CSS类。`addClass()`函数应用一个CSS类，`removeClass()`函数移除一个CSS类。下面的代码为特定的引用添加`emphasis`类，并在所有的段落中将其删除：

```
$('#blockquoteID').addClass ('emphasis');
$('p').removeClass('emphasis');
```

`toggleClass()`函数可以为被选元素的类进行切换。

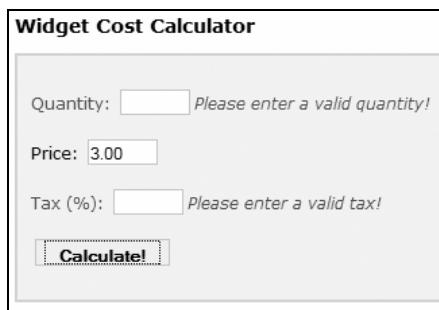
已经提到的函数，一般用于更改页面元素的属性，但你也可以改变这些元素的内容。在上一节中使用的`val()`函数返回表单元素的值。但是，当提供参数时，`val()`会为那个表单元素分配一个新值。

```
$('#something').val('cat');
```

同样，`html()`函数返回一个元素的HTML内容，`text()`函数返回文本内容。这两个函数也可以接受参数，分别用来指定新的HTML和文本。

下面使用所有学到的知识，完成成本计算器小部件。这里将作出以下重要的更改：

- 通过应用`error`类来显示错误；
- 通过隐藏或显示错误信息来显示错误（参见图15-10）；



15

图15-10 现在错误信息将显示在有问题的表单输入框的旁边

- 最终的结果将被写到页面中（参见图15-11）；
- 不再使用警告框。

The screenshot shows a web page titled "Widget Cost Calculator". At the top, it displays the message "The total is \$31.65.". Below this, there are three input fields: "Quantity" with value "10", "Price" with value "3.00", and "Tax (%)" with value "5.5". At the bottom is a "Calculate!" button.

图15-11 计算的结果现在将显示在表单的上面

有很多方法可以显示或隐藏错误信息。要在这里实现的最简单的方法就是，手动将信息添加到表单，然后使用JavaScript切换它们的可见性。以更新HTML页面开始以下步骤。

### 操作DOM

- (1) 在文本编辑器或IDE中打开calculator.html。
- (2) 在quantity表单元素和它的结束段落标签之间添加错误信息（参见脚本15-6）。

#### 脚本 15-6 更新的 HTML 页在关键的表单输入框旁边有硬编码的错误信息

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Widget Cost Calculator</title>
6 <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />
7 <script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
8 <script type="text/javascript" src="js/calculator.js" charset="utf-8"></script>
9 </head>
10 <body>
11 <!-- Script 15.6 - calculator.html #2 -->
12 <h1>Widget Cost Calculator</h1>
13 <p id="results"></p>
14 <form action="calculator.php" method="post" id="calculator">
15 <p id="quantityP">Quantity: <input type="text" name="quantity" id="quantity" />Please enter a valid quantity!</p>
16 <p id="priceP">Price: <input type="text" name="price" id="price" />Please enter a valid price!</p>
17 <p id="taxP">Tax (%): <input type="text" name="tax" id="tax" /> Please enter a valid tax!</p>
18 <p><input type="submit" name="submit" id="submit" value="Calculate!" /></p>
19 </form>
20 </body>
21 </html>
```

现在跟在文本输入框后面的是一个显示错误信息的文本，它同样有一个唯一的id。包含错误信息的span同样使用了errorMessage类。由于外部的CSS文档，这个类会影响消息的格式，并使得jQuery可以在页面首次加载的时候隐藏全部的错误信息。

(3) 其他两个表单输入框，重复步骤(2)。

```
<p id="priceP">Price: <input type="text" name="price" id="price" /> Please enter a valid price! </p>
<p id="taxP">Tax (%): <input type="text" name="tax" id="tax" /> Please enter a valid tax! </p>
```

(4) 保存该文件。

(5) 在文本编辑器或IDE中打开calculator.js。

(6) 移除所有现有的alert()调用（参见脚本15-7）。

#### 脚本 15-7 使用 jQuery、JavaScript 代码操作 DOM 来代替使用调用 alert()

```
1 // Script 15.7 - calculator.js #2
2 // This script is included by calculator.html.
3 // This script handles and validates the form submission.
4
5 // Do something when the document is ready:
6 $(function() {
7
8 // Hide all error messages:
9 $('.errorMessage').hide()
10
11 // Assign an event handler to the form:
12 $('#calculator').submit(function() {
13
14 // Initialize some variables:
15 var quantity, price, tax, total;
16
17 // Validate the quantity:
18 if ($('#quantity').val() > 0) {
19
20 // Get the quantity:
21 quantity = $('#quantity').val();
22
23 // Clear an error, if one existed:
24 $('#quantityP').removeClass ('error');
25
26 // Hide the error message, if it was visible:
27 $('#quantityError').hide();
28
29 } else { // Invalid quantity!
30
31 // Add an error class:
32 $('#quantityP').addClass ('error');
33
34 // Show the error message:
35 $('#quantityError').show();
36
```

```

37 }
38
39 // Validate the price:
40 if ($('#price').val() > 0) {
41 price = $('#price').val();
42 $('#priceP').removeClass('error');
43 $('#priceError').hide();
44 } else {
45 $('#priceP').addClass('error');
46 $('#priceError').show();
47 }
48
49 // Validate the tax:
50 if ($('#tax').val() > 0) {
51 tax = $('#tax').val();
52 $('#taxP').removeClass('error');
53 $('#taxError').hide();
54 } else {
55 $('#taxP').addClass('error');
56 $('#taxError').show();
57 }
58
59 // If appropriate, perform the calculations:
60 if (quantity && price && tax) {
61
62 total = quantity * price;
63 total += total * (tax/100);
64
65 // Display the results:
66 $('#results').html('The total is $' + total + '.');
67
68 }
69
70 // Return false to prevent an actual form submission:
71 return false;
72
73 }); // End of form submission.
74
75}); // End of document ready.

```

(7) 在提交事件处理器之前，隐藏所有带有errorMessage类的元素：

```
$('.errorMessage').hide();
```

这个选择器获取了所有带有errorMessage类的元素类型。在HTML表单中，仅应用于三个span标签。

(8) 在为本地变量quantity分配一个值后，在if子句代码中移出error类并隐藏错误消息。

```
$('#quantityP').removeClass('error');
$('#quantityError').hide();
```

在第(9)步你会看到，当用户输入无效的数量时，数量段落（id为quantityP的值）将被分配error类，并且数量的错误信息（即，#quantityError）将显示出来。如果用户输入了无效的数量，但然后又输入了一个有效的，这两个效果必须使用这里的代码撤销。

在这种情况下，无效的数量将永远不会被提交，数量段落将不会分配error类，并且数量错误消息仍然会被隐藏。如果要求jQuery做一些无法完成的事情时，例如隐藏已经隐藏的元素，jQuery就会忽略这个请求。

(9) 如果数量是无效的，添加error类并显示错误消息。

```
$('#quantityP').addClass('error');
$('#quantityError').show();
```

这与步骤(8)中的代码相反。注意，要放在else子句中。

(10) 为price重复步骤(8)和步骤(9)，if-else语句如下所示。

```
if ($('#price').val() > 0) {
 price = $('#price').val();
 $('#priceP').removeClass('error');
 $('#priceError').hide();
} else {
 $('#priceP').addClass('error');
 $('#priceError').show();
}
```

(11) 为tax重复步骤(8)和步骤(9)，if-else语句如下所示。

```
if ($('#tax').val() > 0) {
 tax = $('#tax').val();
 $('#taxP').removeClass('error');
 $('#taxError').hide();
} else {
 $('#taxP').addClass('error');
 $('#taxError').show();
}
```

(12) 在计算总数之后，在同一个if子句中更新results段落。

```
$('#results').html('The total is $' + total + '.'');
```

总数信息可以动态地写到HTML页面中，而不是使用警告框。要实现这个功能的方法之一就是更改页面中的元素的文本或HTML。本页面已经有一个用于这个目的的空段落，id值为results。要更改段落内的文字，可以应用text()函数。要更改段落内的HTML，使用html()代替。

(13) 将页面保存为calculator.js（在js文件夹中），并在Web浏览器中测试该计算器。

同样，请记住必须重新加载HTML页面（因为HTML和JavaScript都已经更新了）。

### ✓ 提示

- 如果选择了不存在的页面元素，jQuery不会抛出错误。如果在不存在的元素上调用函数，jQuery也不会抛出错误。
- 在JavaScript中，就像其他面向对象的语言一样，你可以“链式”方法调用函数，一次执行多个动作。以下段代码在一行中显示以前隐藏的段落、添加一个新的类并改变它的文本内容：

```
$('#pId').show().addClass('thisClass').text('Hello, world!');
```

- 你可以使用attr()函数更改选中内容的属性。它的第一个参数是影响的属性，第二是新的值。

例如，下面的代码将为一个提交按钮添加disabled="disabled"属性以禁用它：

```
$('#submitButtonId').attr('disabled', 'disabled');
```

- 你可以使用`prepend()`、`append()`、`remove()`及其他函数来添加、移动或删除元素。在jQuery手册查看详情。

## 15.7 使用 Ajax

与DOM操作一样，JavaScript和jQuery的另外一个关键应用就是Ajax。Ajax术语的说法最早出现在2005年，浏览器已经支持它很多年了。到2011年，Ajax已经是许多动态Web站点的标准功能，并且已被所有的主流浏览器支持。但是，什么是Ajax呢？

Ajax可以意味着很多东西，涉及几个不同的技术和方法，总的来说，Ajax就是使用JavaScript，在用户不知道的情况下，执行服务器端的请求。在标准请求模式中（几乎所有本书的示例），用户可能以`login.html`开始。而在表单提交时，浏览器可能会指向`login.php`（实际完成表单验证的地方，并会进行数据库配准，然后将显示结果（参见图15-12）。（即使显示和处理表单用的是同一个PHP脚本，标准的请求模型也需要在同一个页面使用两个独立和公开的请求。）

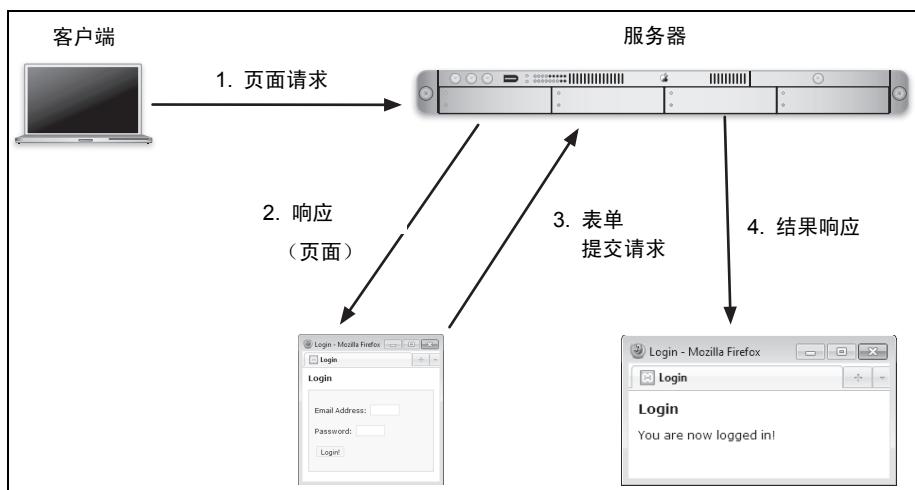


图15-12 使用浏览器不断重新加载整个Web页的标准的客户端—服务器请求模型

在Ajax模型中，表单的提交会被JavaScript截获，JavaScript会将表单数据发送到服务器端的PHP脚本。PHP脚本完成验证和其他必要的任务，然后仅返回数据到客户端的JavaScript，表明操作的结果。然后，客户端的JavaScript使用返回的数据来更新Web页面（参见图15-13）。虽然这涉及很多步骤，但用户并不知晓，并且在这个过程发生时，用户仍能继续与Web页面交互。

在Web站点中包含Ajax会带来更好的用户体验，更类似于桌面应用的行为方式。其次，由于需要传回更少的数据（例如，不需要发送整个第二页的HTML，如`login.php`），因此可以带来更好的性能。

你已经了解了很多执行Ajax事务需要的信息：使用JavaScript进行表单验证、使用PHP进行表单验证、使用JavaScript更新DOM。你需要了解的最后一点内容就是如何使用jQuery执行实际的Ajax请求。在接下来的几页中，你将创建HTML表单，服务器端PHP脚本，以及中介JavaScript，这一切都是为了

处理登录表单而设计的。为了缩短和简化代码，我去掉了一些无关紧要的内容，但是我会在这么做的时候明确的指出来，而且删掉的内容你可以很容易地补充上。

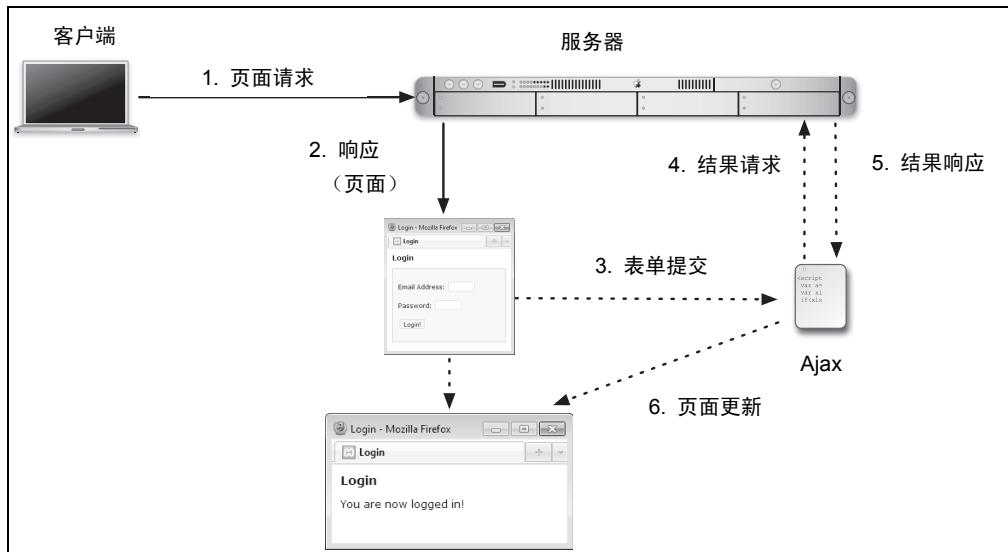
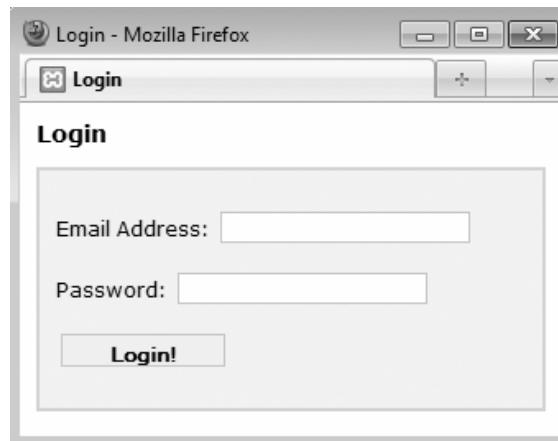


图15-13 使用Ajax，服务器请求可以在幕后发起，浏览器可以在不重新加载的情况下更新

### 15.7.1 创建表单

登录表单需要两个输入：一个电子邮件地址和一个密码（参见图15-14）。这个表单会使用与calculator.html相同的技术显示错误和表明结果（参见图15-15）。



15

图15-14 登录表单

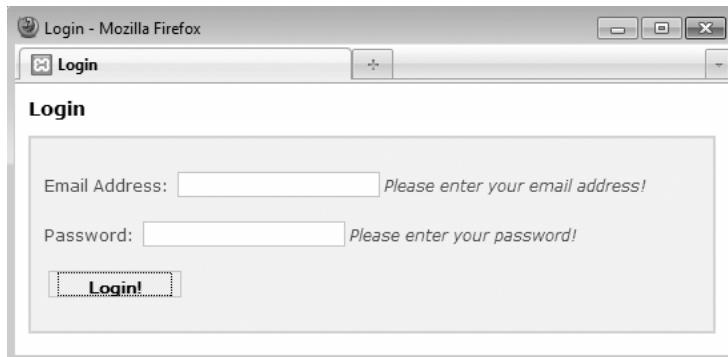


图15-15 错误信息被关联到每个表单元的旁边

### 创建表单

(1) 在文本编辑器或IDE中创建名为login.php的新的PHP文件(参见脚本15-8)。

**脚本 15-8** 登录表单有一个电子邮箱地址的文本输入框，一个密码输入框和一个提交按钮，其他的元素由jQuery操作

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Login</title>
6 <link rel="stylesheet" href="css/style.css" type="text/css" media="screen" />
7 <script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
8 <script type="text/javascript" src="js/login.js" charset="utf-8"></script>
9 </head>
10 <body>
11 <!-- Script 15.8 - login.php -->
12 <h1>Login</h1>
13 <p id="results"></p>
14 <form action="login.php" method="post" id="login">
15 <p id="emailP">Email Address: <input type="text" name="email" id="email" />Please enter your email address!</p>
17 <p id="passwordP">Password: <input type="password" name="password" id="password" />Please enter your password!</p>
19 <p><input type="submit" name="submit" id="submit" value="Login!" /></p>
20 </form>
21 </body>
22 </html>
```

这实际上是PHP脚本，而不仅仅是HTML文件。该页面与calculator.html使用同一个外部CSS文件。

(2) 包含jQuery库和第二个JavaScript文件。

```
<script type="text/javascript" src="js/jquery-1.6.1.min.js" charset="utf-8"></script>
<script type="text/javascript" src="js/login.js" charset="utf-8"></script>
```

该页面使用的jQuery库与calculator.html相同。本页特定的JavaScript将会放在login.js中。两者

都将存储在js文件夹中，与这个脚本放在相同的目录。

(3) 完成HTML头并开始body。

```
</head>
<body>
<!-- Script 15.8 - login.php -->
<h1>Login</h1>
<p id="results"></p></pre>

```

在body中表单的前面是id为results的空段落，用于在后面用jQuery动态填充（参见图15-16）。

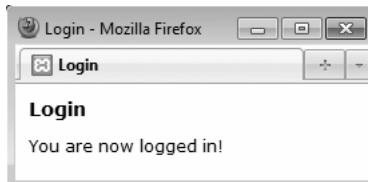


图15-16 如果登录成功，表单将会消失并会在标题下面显示消息

(4) 创建表单。

```
<form action="login.php" method="post" id="login">
<p id="emailP">Email Address: <input type="text" name="email" id="email" /><span class="errorMessage"
 id="emailError">Please enter your email address!</p>
<p id="passwordP">Password: <input type="password" name="password" id="password" /><span class="error
 Message" id="passwordError">Please enter your password!</p>
<p><input type="submit" name="submit" id="submit" value="Login!" /></p>
</form>
```

这表单与calculator.html中的非常相似。两个表单元素都被具有唯一id值的段落包裹，在需要的时候可以让jQuery为它应用error类。两个元素都跟随着默认的错误信息，可以让jQuery显示或隐藏它们。

(5) 完成HTML页面。

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

(6) 将页面保存为login.php并在Web浏览器中加载它。

请记住，这是一个PHP脚本，所以它必须通过一个URL (<http://something>) 进行访问。

## 15.7.2 创建服务器端脚本

15

前面的一系列步骤创建了程序的客户端：HTML表单。接下来，看一下服务器端——处理数据的PHP脚本。这个脚本要做以下两件事情：

- (1) 验证提交的数据；
- (2) 返回结果字符串。

为了简单起见，PHP脚本将只对提交的值与硬编码的值进行比较，但是你可以很容易地改用数据库查询代替。

在Ajax处理过程中，最重要的是，这个PHP脚本只返回单一的字符串（参见图15-17），没有任何

HTML或其他标记（参见图15-18）。这是必要的，因为JavaScript执行Ajax请求能够得到是整个PHP脚本的输出。并且，在这个例子中，你会看到在JavaScript中，PHP脚本的输出将是错误报告和DOM操作执行的基础。

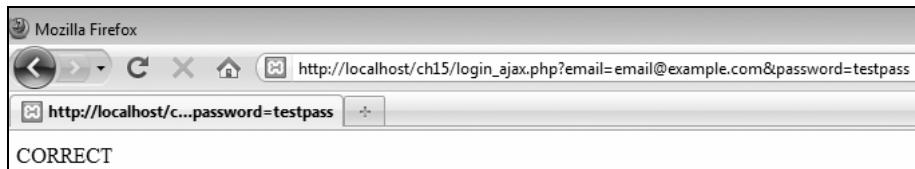


图15-17 当发起正确的请求时，服务器端PHP脚本的结果

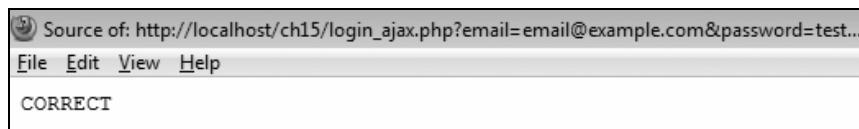


图15-18 服务器端PHP脚本的HTML源代码展示了输出值是一个简单的字符串，没有任何HTML

### 15.7.3 处理Ajax请求

(1) 在文本编辑器或IDE中创建新的PHP文件，命名为login\_ajax.php（参见脚本15-9）。

**脚本 15-9** PHP 脚本将会接受 JavaScript 发起的 Ajax 请求。它执行了一些校验并返回简单的字符串来表明结果

```

1 <?php # Script 15.9 - login_ajax.php
2 // This script is called via Ajax from login.php.
3 // The script expects to receive two values in the URL: an email address and a password.
4 // The script returns a string indicating the results.
5
6 // Need two pieces of information:
7 if (isset($_GET['email'], $_GET['password'])) {
8
9 // Need a valid email address:
10 if (filter_var($_GET['email'], FILTER_VALIDATE_EMAIL)) {
11
12 // Must match specific values:
13 if (($_GET['email'] == 'email@example.com') && ($_GET['password'] == 'testpass')) {
14
15 // Set a cookie, if you want, or start a session.
16
17 // Indicate success:
18 echo 'CORRECT';
19
20 } else { // Mismatch!
21 echo 'INCORRECT';

```

```

22 }
23
24 } else { // Invalid email address!
25 echo 'INVALID_EMAIL';
26 }
27
28 } else { // Missing one of the two variables!
29 echo 'INCOMPLETE';
30 }
31
32 ?>

```

再强调一次，该脚本并不直接执行，所以它不包含任何HTML。

(2) 验证在URL中收到的电子邮件地址和密码。

```
if (isset($_GET['email'], $_GET['password'])) {
```

该脚本会发起一个GET请求，因此，代码首先要做的就是确认已经接收到了电子邮件地址和密码。

(3) 验证提交的电子邮件地址。

```
if (filter_var($_GET['email'], FILTER_VALIDATE_EMAIL)) {
```

使用过滤器扩展，检查所提供的电子邮件地址是否合法。如果你的PHP版本的不支持过滤器扩展，可以使用正则表达式（详见第14章）。

(4) 如果提交的值是正确的，显示成功信息。

```
if (($_GET['email'] == 'email@example.com') && ($_GET['password'] == 'testpass')) {
 echo 'CORRECT';
```

如前所述，这个代码只是将提交的值与两个静态的字符串比较。你可以很容易地将代码替换为数据库查询代码（详见第12章）。同时，你也可以设置一个cookie，或者开始一个会话（查看下面的提示，了解这么做的“陷阱”）。

最重要的是，该脚本仅输出CORRECT，没有任何其他的HTML（参见图15-7和图15-8）。

(5) 完成三个条件句。

```

} else {
 echo 'INCORRECT';
}
} else {
 echo 'INVALID_EMAIL';
}
} else {
 echo 'INCOMPLETE';
}
```

这3个else子句完成了在步骤(2)、(3)和(4)中开始的条件语句。每个都打印了一行字符串表明了某种状态。接下来要写的与登录表单相关的JavaScript，将会根据每一个可能的结果采取不同的动作。

(6) 完成PHP页面。

```
?>
```

(7) 将文件保存为login\_ajax.php，并将其放置在Web目录中与login.php相同的文件夹下。

注意，要让Ajax请求正常工作，需要将这两个文件放到相同的目录中。

### ✓ 提示

□ 在一个Ajax处理中，设置cookie或开始一个会话，在服务器端的PHP脚本中是完全可以接受的。然而，如果页面已经在Web浏览器中加载，这意味着那个页面不会再访问cookies或创建会话。在创建cookie或开始会话后你需要使用JavaScript更新页面，随后在浏览器中加载的页面将可以访问cookie或会话数据。

#### 15.7.4 创建JavaScript

最后一步是创建中断表单提交的JavaScript，发送数据到服务器端的PHP脚本，读取PHP脚本的结果，并且相应地更新DOM。这是在客户端的HTML表单和服务器端PHP之间的“粘合剂”。所有表单验证和DOM操作的JavaScript，都与你在本章已经见到的非常相似。但下面要介绍两个新的概念。

首先，你需要知道如何在JavaScript中创建通用对象。本例中，一个对象代表要发送到PHP脚本的数据，另一个对象代表Ajax请求的选项。下面是在JavaScript中创建新的对象的方法：

```
var objectName = new Object();
```

接下来的章节会介绍OOP的更多细节，但是现在可以将之理解为创建了一个Object类型的新变量。大写字母“O”object是JavaScript中的一个空模板（作为面向对象的语言，JavaScript中的大多数变量是某种类型的对象）。在创建了对象之后，就可以使用下面的语法为它添加值：

```
objectName.property = value;
```

如果你是JavaScript或者OOP的初学者，为了有助理解，可以将一般的对象看作一个有名字和对应值的索引数组。

第二个新的信息是jQuery的ajax()函数的用法。该函数执行Ajax请求。它接受请求设置作为它唯一的参数。作为jQuery库的一部分，像下面这样调用：

```
$.ajax(settings);
```

以上是基础知识，下面的代码中将会详细讨论细节。

#### 执行Ajax请求

(1) 在文本编辑器或IDE创建新的JavaScript文件，命名为login.js（参见脚本15-10）。

**脚本 15-10** 这个文件中的 JavaScript 代码执行了一个服务器端脚本的 Ajax 请求，并且基于返回的响应更新了 DOM

```

1 // Script 15.10 - login.js
2 // This script is included by login.php.
3 // This script handles and validates the form submission.
4 // This script then makes an Ajax request of login_ajax.php.
5
6 // Do something when the document is ready:
7 $(function() {
8
9 // Hide all error messages:
10 $('.errorMessage').hide();
11
12 // Assign an event handler to the form:

```

```
13 $('#login').submit(function() {
14 // Initialize some variables:
15 var email, password;
16
17 // Validate the email address:
18 if ($('#email').val().length >= 6) {
19 // Get the email address:
20 email = $('#email').val();
21
22 // Clear an error, if one existed:
23 $('#emailP').removeClass('error');
24
25 // Hide the error message, if it was visible:
26 $('#emailError').hide();
27
28 } else { // Invalid email address!
29
30 // Add an error class:
31 $('#emailP').addClass('error');
32
33 // Show the error message:
34 $('#emailError').show();
35
36 }
37
38 }
39
40 // Validate the password:
41 if ($('#password').val().length > 0) {
42 password = $('#password').val();
43 $('#passwordP').removeClass ('error');
44 $('#passwordError').hide();
45 } else {
46 $('#passwordP').addClass ('error');
47 $('#passwordError').show();
48 }
49
50 // If appropriate, perform the Ajax request:
51 if (email && password) {
52
53 // Create an object for the form data:
54 var data = new Object();
55 data.email = email;
56 data.password = password;
57
58 // Create an object of Ajax options:
59 var options = new Object();
60
61 // Establish each setting:
62 options.data = data;
63 options	dataType = 'text';
64 options.type = 'get';
65 options.success = function (response) {
66
```

```

67 // Worked:
68 if (response == 'CORRECT') {
69
70 // Hide the form:
71 $('#login').hide();
72
73 // Show a message:
74 $('#results').removeClass('error');
75 $('#results').text('You are now logged in!');
76
77 } else if (response == 'INCORRECT') {
78 $('#results').text('The submitted credentials do not match those on file!');
79 $('#results').addClass('error');
80 } else if (response == 'INCOMPLETE') {
81 $('#results').text('Please provide an email address and a password!');
82 $('#results').addClass('error');
83 } else if (response == 'INVALID_EMAIL') {
84 $('#results').text('Please provide your email address!');
85 $('#results').addClass('error');
86 }
87
88 }; // End of success.
89 options.url = 'login_ajax.php';
90
91 // Perform the request:
92 $.ajax(options);
93
94 } // End of email && password IF.
95
96 // Return false to prevent an actual form submission:
97 return false;
98
99}); // End of form submission.
100
101}); // End of document ready.

```

(2) 添加jQuery代码处理文档的“就绪”状态。

```

$(function() {
});

```

JavaScript需要使用此代码开始，以便当Web浏览器就绪后设置表格。由于语法比较复杂，我认为最好先添加整个代码块，然后将所有代码写在花括号内。

(3) 隐藏带有errorMessage类的所有元素。

```

$('.errorMessage').hide();

```

选择器获取了带有errorMessage类的所有类型的元素引用。在HTML表单中，这仅适用于3个span标签。它们会在DOM加载完成后被这些代码隐藏。

(4) 为表单提交创建事件侦听器。

```

$('#login').submit(function() {
});

```

此代码与计算器表单中的大致相同。随后步骤中的代码将包含在此花括号中。

(5) 初始化两个变量。

```
var email, password;
```

这两个变量将作为表单数据的局部变量。

(6) 验证电子邮件地址。

```
if ($('#email').val().length >= 6) {
 email = $('#email').val();
 $('#emailP').removeClass('error');
 $('#emailError').hide();
```

计算器表单验证所有的数字都要大于零，但是这对于登录表单验证来说不太合适。这里，条件语句确认email输入框值的字符串长度要大于等于6（6个字符是有效的电子邮件地址的最小要求，例如a@b.cc）。你也可以在JavaScript中使用正则表达式执行更严格的验证，这里是为了简化示例（如前面代码所示，服务器端PHP脚本同样会验证电子邮件地址）。

如果电子邮件地址的值通过最基本的验证，将它赋给局部变量。其次，如果error类在前面添加过，就从段落中将其删除；如果电子邮件的错误消息是显示的，则将它隐藏。

(7) 完成电子邮件地址的条件语句。

```
} else {
 $('#emailP').addClass('error');
 $('#emailError').show();
}
```

此代码完成了在第(6)步开始的条件语句。该代码与calculator.html中使用的相同，为整个段落添加error类并显示错误信息（参见图15-15）。

(8) 验证密码。

```
if ($('#password').val().length > 0) {
 password = $('#password').val();
 $('#passwordP').removeClass('error');
 $('#passwordError').hide();
} else {
 $('#passwordP').addClass('error');
 $('#passwordError').show();
}
```

密码的最小长度可能由注册过程决定。作为占位符，这段代码只是确定字符串的长度大于0。其他方面，这段代码与前面的两个步骤在本质上基本相同。

(9) 如果这两个值都已收到，将它们存储在一个新的对象中。

```
if (email && password) {
 var data = new Object();
 data.email = email;
 data.password = password;
```

这段代码背后的基础知识在前面解释过了。首先创建新的通用对象。然后创建对象的属性email，并将电子邮件地址值赋给它。最后，创建属性password，将输入的密码值赋给它。如果以PHP的形式想象这段代码，会更好理解，它相当于：

```
$data = array();
$data['email'] = $email;
$data['password'] = $password;
```

(10) 为Ajax选项创建新的对象，并建立前3个设置。

```
var options = new Object();
options.data = data;
options.dataType = 'text';
options.type = 'get';
```

在这里，创建了另一个通用对象。接下来，将data对象的值赋给options的属性。options的这个属性存储了一些数据，它们作为Ajax请求的一部分，会传递给PHP脚本。

第二个设置是期望从服务器端请求返回的数据类型。由于PHP脚本login\_ajax.php返回（即，打印）简单的字符串，这里的值是text。dataType设置会影响JavaScript处理返回响应的方式：它需要符合实际的服务器响应。

type设置为发起请求的类型，get和post是两种最常见的类型。GET请求是默认的，在这里它不需要被赋值，但是这里代码还是明确指出了类型。

需要明确的是，由于data对象的属性名为email和password、type的值为get，因此login\_ajax.php脚本将会接收\$\_GET['email']和\$\_GET['password']。如果你要改变data对象的属性名字或options.type的值，服务器端的PHP脚本将会以不同的超级全局变量接收Ajax的数据。

(11) 定义当Ajax请求成功时会发生什么。

```
options.success = function (response) {
}; // End of success.
```

success属性定义了当Ajax请求工作时JavaScript会做什么。这里“工作”的意思是，JavaScript能够执行服务器端页面的请求并接收结果。在实际情况中，会赋给此属性一个匿名函数。在此步骤中，定义了一个匿名函数并完成了赋值。随后步骤中的代码将包含在此花括号中。

你可以看到，这个匿名函数接受一个参数——服务器端脚本的响应，赋给response变量。如前所述，JavaScript接收到的响应将是PHP脚本全部的输出。

(12) 在步骤(11)中创建的匿名函数，如果服务器响应等于CORRECT，隐藏表单并更新页面。

```
if (response == 'CORRECT') {
 $('#login').hide();
 $('#results').removeClass ('error');
 $('#results').text('You are now logged in!');
```

如果用户提交正确的登录信息——email@example.com和testpass，login\_ajax.php将返回字符串CORRECT。在这种情况下，JavaScript将隐藏整个登录表单，并赋给results段落一个字符串（参见图15-16）。由于不正确的提交可能已经为段落添加了error类（见步骤(13)），这里还要删除它。

(13) 如果服务器的响应等于INCORRECT，提示错误信息。

```
} else if (response == 'INCORRECT') {
 $('#results').text('The submitted credentials do not match those on file!');
 $('#results').addClass('error');
```

如用户提交了密码和在语法上有效的电子邮件地址，但提供的值不正确，那么服务器端PHP脚本将返回字符串INCORRECT。在这种情况下，将赋给result段落一个不同的字符串，而且还会为这个段落

应用error类（参见图15-19）。

图15-19 提供无效登录信息的结果

(14) 为另两个服务器响应添加子句。

```
} else if (response == 'INCOMPLETE') {
 $('#results').text('Please provide an email address and a password!');
 $('#results').addClass('error');
} else if (response == 'INVALID_EMAIL') {
 $('#results').text('Please provide your email address!');
 $('#results').addClass('error');
}
```

这与步骤(13)的代码差不多，就是提示的消息不同。这是success属性的匿名函数的结束代码。

(15) 添加的url属性，并发起请求。

```
options.url = 'login_ajax.php';
$.ajax(options);
```

Ajax对象的url属性指定了请求应发送到的实际服务器端脚本。只要login.php的和login\_ajax.php是在同一个目录，这个引用将会生效。

最后，创建完所有的请求选项后，执行请求。

(16) 完成步骤(9)开始的条件语句，并返回false。

```
} // End of email && password IF.
return false;
```

如果email和password变量不是TRUE值，不会进行Ajax请求（这个条件句没有else子句）。最后，返回值为false，以防止表单真正提交。

(17) 将页面保存为login.js（在js文件夹中），并在Web浏览器中测试登录表单。

#### ✓ 提示

- 作为一个调试技巧，直接运行服务器端脚本（参见图15-17），有助于确认它是否能正常工作（不包含解析或其他错误）。
- Firefox的Firebug扩展可以展现网页发起的Ajax请求的细节（参见图15-20）。
- Opera Web浏览器有一个名为蜻蜓（Dragonfly）的内置工具，也是一个很好的开发资源。

- 由于JavaScript可以被用户禁用，所以不能完全依赖JavaScript的表单验证。你必须使用服务器端的PHP验证来保护网站。

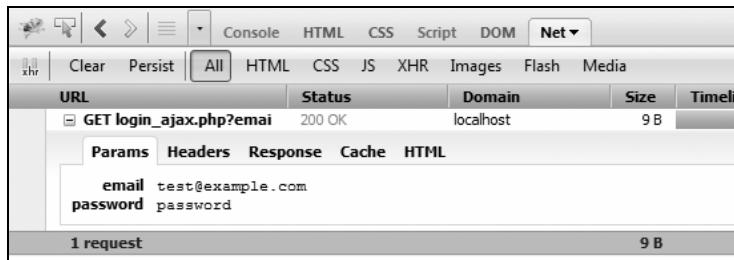


图15-20 在Firebug扩展的帮助下，你可以看到Ajax请求幕后发生的事情

## 15.8 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 15.8.1 回顾

- 什么是JavaScript？JavaScript与PHP有何不同？
- 什么是jQuery？jQuery和JavaScript之间的关系是什么？
- 如何将外部JavaScript文件包含到HTML页面中？如何将JavaScript代码放到HTML页面中？
- 为什么要等到整个DOM加载完成之后再执行引用了DOM元素的JavaScript代码？
- 在DOM中，唯一标识符为什么是必须的？
- 在jQuery中，如何选择给定标签类型的元素？如何选择有确定class的元素？你如何选择一个特定的元素？
- 在jQuery中，如何为页面元素添加事件监听器（或多个元素）？什么是事件监听器？
- 为什么必须在改变了HTML页面的JavaScript后重新加载它？
- 可以操作DOM的jQuery函数有哪些？
- 什么是Ajax？为什么Ajax是“好东西”？
- 为什么需要执行服务器端请求的HTML页面必须通过URL加载？
- 如何在JavaScript中创建通用对象？
- Ajax请求的type属性有什么影响？数据对象中属性的名字有什么影响？

### 15.8.2 实践

- 访问jQuery的网站，并开始仔细阅读jQuery的文档。
- 检查jQuery UI可以为你的网页做些什么。
- 使用jQuery的文档，或者干脆在网上搜索一些jQuery插件。尝试在网页中使用一两个插件。

- 在熟悉了Ajax的过程后，在线搜索一些信息，比如使用JSON（JavaScript对象符号）或XML（可扩展标记语言）执行Ajax请求，以表示传回的到JavaScript的数据。
- 看看在整个DOM加载完成之前引用DOM元素会发生什么？从中你会了解到，如果你不可避免或偶然在浏览器就绪之前引用DOM，会发生些什么。
- 更新calculator.js，在每次表单提交之初清空results段。这样一来，以前提交的结果将不会显示在后续的无效提交中。
- 修改login\_ajax.php，使用数据库来确认登录成功。
- 修改login\_ajax.php，发送一个cookie，或者开始一个会话。创建另一个PHP脚本，访问所创建的cookie或会话。
- 修改login.php，在用户禁用JavaScript的情况下也可以执行登录验证。提示：这可能比你想象的简单，直接使用PHP来处理表单提交（在同一个文件中），就像JavaScript根本不存在一样。

# 面向对象编程入门

# 16

### 本章内容

- 基础知识和语法
- 使用MySQL
- DateTime类
- 回顾与实践

PHP作为一种不同寻常的语言，它既可以像本书大部分示例中那样作为面向过程的语言使用，也可以作为面向对象编程（OOP）语言使用。两种方法都有各自的优点，每个人都应该熟悉它们（最终）。

不幸的是，掌握OOP，需要大量的时间和信息：我编写的 *PHP 5 Advanced: Visual QuickPro Guide* 一书中，这个主题讲述了150页！不过，OOP最伟大的事情之一是，你可以在不完全了解它的情况下使用它。很快你就会明白这是什么意思。

本章是这一版新增的内容。为了以最好的方式比较这两种编程方法，有些例子会重复本书中已经展示的面向过程的例子。框注和提示还会提到PHP中其他的OOP应用，其中大部分都没有对应的面向过程的方式。

## 16.1 基础知识和语法

如果你之前从来没有做过任何面向对象编程，那它的概念和语法你都会很陌生。简单地说，所有的应用程序或脚本，都包括对信息采取的动作：验证、操纵、将它存储到数据库，等等。从理论上说，编写面向过程的程序重点是动作：做这个，然后这个，再然后那个；而OOP是以数据为中心的，更侧重于正在使用的各种信息。

### 16.1.1 面向对象的基础

面向对象编程以类定义开始，它是一种特定类型的数据模板：雇员、用户、页面内容，等等。类的定义包含变量和函数。在语法上，在类中定义的变量称为属性（attribute或property），类中的函数被称为方法。属性和方法的组合就是类的成员。

作为一个假设的例子，你可能有一个Car类。注意，类的名字通常以大写字母开头。Car的属性包

括make（品牌）、model（款式）、year（生产年）、odometer（里程）等，一辆汽车的所有相关信息。可以设置、改变、获取Car的属性，属性的值用于区分各种汽车。Car的方法（汽车可以做的事情）可以包括：start()（启动）、drive()（驾驶）、park()（停泊）和turnOff()（熄火）。这些动作对所有汽车都是通用的。

在本章的介绍中，我曾提到过，你可以在没有真正了解OOP的情况下使用它。我的意思是，根据你的需求使用现成的类定义，这是很容易的，也是常见的做法。事实上，可重用的代码（尤其是别人创建的代码），是OOP的主要优势之一。需要花费大量精力（至少要做对）的是，掌握设计过程：了解要定义什么成员；而更重要的是，如何实现复杂的面向对象的概念，如：

- 继承
- 访问控制
- 方法重载
- 作用域
- 抽象

如果你有兴趣学习如何正确地创建自己的类，可以在我编写的*PHP 5 Advanced: Visual QuickPro Guide*一书中查看相关主题及其他资源，但在这第一章中，我们重点是使用现有的类，而不是创建自定义的类。

### OOP与面向过程

讨论OOP与过程式编程的优劣可以迅速升级为一场口水战，双方都各持己见。PHP的独特之处在于你可以在它俩之间选择（相比之下，C是一个严格的面向过程语言，而Java是面向对象的）。在我看来，每一个编程风格都有它的长处和不足，但不会比另外一个“更好”。

过程式编程可以更快地学习和使用，特别是对于较小的项目。但过程式代码难以维护和扩展，特别是开发更复杂的网站，会有错误成堆的可能性。

另一方面，使用OOP编写的代码，可能更容易维护，特别是在大型项目中，并可能适合团队环境。但OOP难以掌握，在出错时也不容易补救。

随着时间的推移，你会很自然地有自己的意见和喜好。对我来说，真正学到的是利用PHP同时支持这两种语法的优势，这样不管在什么状况下都不会受到限制。

## 16.1.2 PHP中的OOP语法

现在，让我们利用一个已经设计并定义好Car类。大多数类不能直接使用，而是需要你创建这个类的实例，这个类的类型的特定变量。该实例称为对象。在PHP中，使用new关键字创建实例：

```
$obj = new ClassName();
$mine = new Car();
```

鉴于代码\$name = 'Larry' 创建了一个String类型的变量，上面的代码创建了一个Car类型的变量。Car定义的每个部分，每个属性（即变量）和方法（即函数），现在已经嵌入到\$mine中。

在幕后（例如，在类定义中），在创建该类型的新对像时会自动调用一种称为构造函数的特殊方法。构造函数提供了对象的后续使用需要的初始化设置。例如，`MySQLi`类的构造函数会建立到数据库的连接，`DateTime`类的构造函数会创建指向确切日期和时间的引用（`MySQLi`和`DateTime`都会在本章中使用）。

构造函数也可以像任何其他的函数那样接受参数，可以在对象创建时提供：

```
$mine = new Car('Honda', 'Fit', 2008);
```

在创建完对象后，可以使用`$object_name->member_name`语法来引用它的属性（即变量），调用它的方法（即函数）：

```
$mine->color = 'Purple';
$mine->start();
```

第一行（理论上）将`Purple`值赋给对象的`color`属性。第二行调用该对象的`start()`方法。调用任何函数都必需使用括号。如果该方法需要参数，提供方式如下：

```
$mine->drive('Forward');
```

也可以像使用任何其他变量一样使用对象的属性：

```
$mine->odometer += 20;
echo "My car currently has $mine->odometer miles on it.";
```

如果对象的方法返回一个值，该方法可以像任何有返回值的函数一样调用：

```
// The fill() method takes a number of
// gallons being added and returns
// how full the tank is:
$tank = $mine->fill(8.5);
```

要使用面向对象编程，了解这些就足够了。你将看到，接下来的几页示例会重复本书前面讲解的功能，这种使用不同方法得到实现相同结果的对比会有助于你更好地理解OOP。

### ✓ 提示

- 在OOP说明文档中，你可能会看到`ClassName :: METHOD_NAME()`语法。这是指定类方法的一种方式。
- 支持命名空间是PHP 5.3的主要变化之一。通俗地讲，命名空间提供了一种将多个类定义组织在同一标题下的方法。命名空间对组织代码和防止冲突非常有用（例如，区分我的`Car`类和你的`Car`类）。
- 类也可以有自己的常量，就像它们有自己的变量和函数一样。类常量通常不使用该类的实例，如：
 

```
echo ClassName::CONSTANT_NAME;
```
- 从PHP 3到PHP 4再到PHP 5，OOP在PHP中已经发生了巨大的变化。在本章中讨论的语法只适用于PHP 5及更高版本。

### 更多面向对象的类

除了本章列举的还有很多其他的面向对象的类定义，虽然我认为MySQLi和DateTime是两个最明显的可访问和使用类。最大的类可以在标准PHP库（SPL）中找到，内置在PHP的5.0版本中，并在版本5.3中得到了扩展。

SPL提供了几类高端类：异常处理、迭代器（可用于任何数据集的循环）、自定义数据类型，等等。SPL是为更高级的PHP程序员定义的，非常适用于高度的或完全基于OOP的代码。

同样，还有一些为国际化而定义的类（[www.php.net/intl](http://www.php.net/intl)）。这些类定义了一些原本为PHP 6（现在还不存在）准备的功能，其中包括在给定语言环境中定制单词排序、格式化数字等（区域设置包含了语言、文化习惯和某区域特有选项）。

## 16.2 使用 MySQL

在PHP中，你可以使用过程式和面向对象的两种风格来编写代码，而MySQL Improved 扩展同样可以使用两种方式来与数据库进行交互。第9章介绍了基本的过程式方法。作为比较，本章将使用OOP实现相同的功能。

你将在此用到3个定义的类。

- **MySQLi**，主要类，提供了数据库连接、查询方法，等等。
- **MySQLi\_Result**，用于处理SELECT查询的结果（包括其他功能）。
- **MySQLi\_STMT**用于执行预处理语句（详见第13章）。

我会解释每一个类的基本用法，并通过示例脚本详加解释。如果要查看这几个类的所有属性和方法，可以参看PHP手册。

### 16.2.1 创建连接

在过程式的方法中，当使用对象符号与MySQL交互时，首先是必须创建一个连接。使用**MySQLi**类，通过向构造函数传递恰当的连接值，在对象初始化时会创建一个连接（例如，创建对象时）：

```
$mysqli = new MySQLi(hostname, username, password, database);
```

这虽然是面向对象的方式，但当你使用过程式编程，用命令行客户端或其他接口连接到MySQL时，可以使用相同的MySQL值。

如果不能建立连接，`connect_error`属性将存储错误原因（参见图16-1）：

```
if ($mysqli->connect_error) {
 echo $mysqli->connect_error;
}
```

不幸的是，这种做法在PHP的5.2.9之前版本中是有问题的（或只是出bug了），所以你需要在PHP手册中查看临时解决方法（不体面的修复）：

```
if (mysqli_connect_error()) {
 echo mysqli_connect_error();
}
```

```
Warning: mysqli::mysqli() [mysqli\(mysqli\)]: (28000/1045): Access denied for user
'usernaed'@'localhost' (using password: YES) in
C:\xampp\htdocs\mysqli_oop_connect.php on line 14
Access denied for user 'usernaed'@'localhost' (using password: YES)
```

图16-1 MySQL连接错误

如果你没有使用PHP5.2.9或更新的版本，需要在脚本中使用上面的代码。

接下来，你应该确立字符集：

```
$mysqli->set_charset(charset);
$mysqli->set_charset('utf8');
```

这时候，你就可以执行接下来会涉及的查询。

完成查询后，调用close()方法关闭数据库连接：

```
$mysqli->close();
```

想要尽量整洁，还可以删除这个对象：

```
unset($mysqli);
```

接下来写一个连接到MySQL的PHP脚本来练习一下。随后的两个脚本将更新第9章的脚本，使用的模版与第9章相同，将接下来3个脚本放Web目录中（与第9章相同）。

### 创建面向对象的MySQL连接

(1) 在文本编辑器或IDE中创建新的PHP脚本，命名为mysqli\_oop\_connect.php（参见脚本16-1）。

**脚本 16-1** 这个脚本创建了一个新的 MySQLi 对象，通过它可以进行数据库交互

```
1 <?php # Script 16.1 - mysqli_oop_connect.php
2 // This file contains the database access information.
3 // This file also establishes a connection to MySQL,
4 // selects the database, and sets the encoding.
5 // The MySQL interactions use OOP!
6
7 // Set the database access information as constants:
8 DEFINE ('DB_USER', 'username');
9 DEFINE ('DB_PASSWORD', 'password');
10 DEFINE ('DB_HOST', 'localhost');
11 DEFINE ('DB_NAME', 'sitename');
12
13 // Make the connection:
14 $mysqli = new MySQLi(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
15
16 // Verify the connection:
17 if ($mysqli->connect_error) {
18 echo $mysqli->connect_error;
19 unset($mysqli);
20 } else { // Establish the encoding.
21 $mysqli->set_charset('utf8');
22 }
```

该脚本将在很大程度上遵循与第9章中的`mysqli_connect.php`相同的方法。它将不包含任何HTML。

(2) 将数据库连接的参数设置为常量。

```
DEFINE ('DB_USER', 'username');
DEFINE ('DB_PASSWORD', 'password');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'sitename');
```

与前面一样，你需要将其更改为符合你的服务器的值。与第9章相同，这一章的例子将使用`sitename`数据库。

(3) 创建MySQLi对象。

```
$mysqli = new MySQLi(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
```

这个语法已经解释过，使用常量的值传递给构造函数。

(4) 如果出现错误，将它显示出来。

```
if ($mysqli->connect_error) {
 echo $mysqli->connect_error;
 unset($mysqli);
```

如果MySQLi对象的`connect_error`的属性有一个值，就表示该脚本无法建立数据库的连接。在这种情况下，会显示连接错误，并且会释放对象变量，因为它没有用处了。

注意，如果你使用的是5.2.9版本之前的PHP（我想你知道正使用的是什么版本的PHP，对吧？），需要将代码更改为：

```
if (mysqli_connect_error()) {
 echo mysqli_connect_error();
 unset($mysqli);
```

(5) 如果连接已建立，确立编码。

```
} else {
 $mysqli->set_charset('utf8');
}
```

注意，与MySQL通信的编码必须匹配HTML页面和数据库的编码。

(6) 将脚本保存为`mysqli_oop_connect.php`。

本书中省略PHP闭合标签的大部分文件都会被其他文件包含，这个也不例外。

(7) 理想的情况下，将文件放到Web文档目录外。

因为该文件包含敏感的MySQL访问信息，它应该安全地保存起来。如果可以的话，将它放到Web目录的上层目录或其他目录（有关细节，请参阅第9章）。

同样，因为以下两个脚本将使用第9章中使用的模板，所以应该将这个连接脚本存储在与第9章中的`mysqli_connect.php`相同的目录中。

(8) 暂时将这个脚本的副本放置在Web目录中，并在Web浏览器中运行它。

为了测试脚本，你需要在服务器上放置一个副本，以便可以从Web浏览器中访问（也就是说这个脚本的副本必须放在Web目录中）。如果脚本正常运行，结果应该是一个空白页。如果你看到拒绝访问或类似的消息（参见图16-1），这意味着用户名、密码和主机的组合没有访问指定数据库的权限。

(9) 从Web目录中删除的临时副本。

✓ 提示

- 你可以使用print\_r()，了解和调试PHP代码中的对象（参见图16-2）：

```
echo '<pre>' . print_r($mysqli, 1) . '</pre>';
```

```
mysqli Object
(
 [affected_rows] => 0
 [client_info] => mysqlnd 5.0.7-dev - 091210 - $Revision: 304625 $
 [client_version] => 50007
 [connect_errno] => 0
 [connect_error] =>
 [errno] => 0
 [error] =>
 [field_count] => 0
 [host_info] => localhost via TCP/IP
 [info] =>
 [insert_id] => 0
 [server_info] => 5.5.8
 [server_version] => 50508
 [sqlstate] => 00000
 [protocol_version] => 10
 [thread_id] => 6
 [warning_count] => 0
)
```

图16-2 在某个对象上使用print\_r()，可以揭示对象的很多属性名和值，使用预格式化标签包括它可以让输出结果更容易阅读

- 由于如果没有连接，\$mysqli的对象会被释放，任何需要它的脚本可以使用下面的代码来测试连接是否成功：

```
if (isset($mysqli)) { // Do whatever.
```

为简洁起见，这个测试在后面的脚本中会被省略，但是你要知道它是可行的。

- MySQLi构造函数有两个参数：要使用的端口和套接字。如果运行的是MAMP或XAMPP（见附录A，可以从peachpit.com下载），可能需要提供端口。

- MySQLi::character\_set\_name()方法返回当前字符集。MySQLi::get\_charset()方法返回字符集、校对规则以及更多的内容。

- 你可以使用select\_db()方法更改当前连接使用的数据库：

```
$mysqli->select_db(dbname);
```

## 16.2.2 执行简单的查询

在成功建立了到MySQL服务器的连接后，就可以使用MySQLi对象来查询数据库。为此，你可以调用相应的名为query()的方法：

```
$mysqli->query(query);
```

它唯一的参数就是要执行的SQL命令，我通常将它先赋给一个独立的变量：

```
$q = 'SELECT * FROM tablename';
$mysqli->query($q);
```

你可以在条件语句中用方法调用测试查询是否成功执行：

```
if ($mysqli->query($q)) { // Worked!
```

还可以检查error属性（参见图16-3）：

```
if ($mysqli->error) { // Did not work!
 echo $mysqli->error;
}
```

### System Error

You could not be registered due to a system error. We apologize for any inconvenience.

Table 'sitename.user' doesn't exist

Query: INSERT INTO user (first\_name, last\_name, email, pass, registration\_date) VALUES ('this', 'that', 'email@example.com', SHA1('password'), NOW() )

图16-3 MySQL报告的查询结果错误

如果刚执行的查询是INSERT，则可以通过insert\_id属性获取自动生成的主键的值：

```
$id = $mysqli->insert_id;
```

如果查询执行的是UPDATE、INSERT或DELETE，则可以从affected\_row属性中检索受影响的行数，有多少行被更新、添加或删除：

```
echo "$mysqli->affected_rows rows were affected by the query.";
```

最后一件要了解的事情是，如何在执行任何查询之前规范化查询中使用的数据。这需要先为字符串变量使用real\_escape\_method()函数：

```
$var = $mysqli->real_escape_string($var);
```

这相当于调用mysqli\_real\_escape\_string()，并防止引号和其他有问题的字符破坏SQL命令的语法。接下来的几个步骤将使用所有这些信息，使用OOP改写第9章中的register.php（参见脚本9-5）。

#### 执行简单查询

(1) 在文本编辑器或IDE中打开register.php（参见脚本9-5）。

(2) 修改MySQL连接脚本中的包含文件（参见脚本16-2）。

#### 脚本 16-2 更新版本的注册脚本，通过 OOP 使用了 MySQL 的 Improved 扩展

```
1 <?php # Script 16.2 - register.php
2 // This script performs an INSERT query to add a record to the users table.
3 // This is an OOP version of the script from Chapter 9.
4
5 $page_title = 'Register';
6 include ('includes/header.html');
7
8 // Check for form submission:
9 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
10 }
```

```
11 require ('../mysqli_oop_connect.php'); // Connect to the db.
12
13 $errors = array(); // Initialize an error array.
14
15 // Check for a first name:
16 if (empty($_POST['first_name'])) {
17 $errors[] = 'You forgot to enter your first name.';
18 } else {
19 $fn = $mysqli->real_escape_string(trim($_POST['first_name']));
20 }
21
22 // Check for a last name:
23 if (empty($_POST['last_name'])) {
24 $errors[] = 'You forgot to enter your last name.';
25 } else {
26 $ln = $mysqli->real_escape_string(trim($_POST['last_name']));
27 }
28
29 // Check for an email address:
30 if (empty($_POST['email'])) {
31 $errors[] = 'You forgot to enter your email address.';
32 } else {
33 $e = $mysqli->real_escape_string(trim($_POST['email']));
34 }
35
36 // Check for a password and match against the confirmed password:
37 if (!empty($_POST['pass1'])) {
38 if ($_POST['pass1'] != $_POST['pass2']) {
39 $errors[] = 'Your password did not match the confirmed password.';
40 } else {
41 $p = $mysqli->real_escape_string(trim($_POST['pass1']));
42 }
43 } else {
44 $errors[] = 'You forgot to enter your password.';
45 }
46
47 if (empty($errors)) { // If everything's OK.
48
49 // Register the user in the database...
50
51 // Make the query:
52 $q = "INSERT INTO users (first_name, last_name, email, pass, registration_date) VALUES
53 ('$fn', '$ln', '$e', SHA1('$p'), NOW())";
54
55 // Execute the query:
56 $mysqli->query($q);
57
58 if ($mysqli->affected_rows == 1) { // If it ran OK.
59 // Print a message:
60 echo '<h1>Thank you!</h1>
61 <p>You are now registered. In Chapter 12 you will actually be able to log in!</p><p>

62 </p>';
```

```

63 } else { // If it did not run OK.
64
65 // Public message:
66 echo '<h1>System Error</h1>
67 <p class="error">You could not be registered due to a system error. We apologize for any
68 inconvenience.</p>';
69
70 // Debugging message:
71 echo '<p>' . $mysqli->error . '

Query: ' . $q . '</p>';
72 } // End of if ($r) IF.
73
74 $mysqli->close(); // Close the database connection.
75 unset($mysqli);
76
77 // Include the footer and quit the script:
78 include ('includes/footer.html');
79 exit();
80
81 } else { // Report the errors.
82
83 echo '<h1>Error!</h1>
84 <p class="error">The following error(s) occurred:
';
85 foreach ($errors as $msg) { // Print each error.
86 echo " - $msg
\n";
87 }
88 echo '</p><p>Please try again. </p><p>
</p>';
89
90 } // End of if (empty($errors)) IF.
91
92 $mysqli->close(); // Close the database connection.
93 unset($mysqli);
94
95 } // End of the main Submit conditional.
96 ?>
97 <h1>Register</h1>
98 <form action="register.php" method="post">
99 <p>First Name: <input type="text" name="first_name" size="15" maxlength="20" value="<?php if (isset
100 ($_POST['first_name'])) echo $_POST['first_name']; ?>" /></p>
101 <p>Last Name: <input type="text" name="last_name" size="15" maxlength="40" value="<?php if(isset
102 ($_POST['last_name'])) echo $_POST['last_name']; ?>" /></p>
103 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="<?php if
104 (isset ($_POST['email'])) echo $_POST['email']; ?>" /> </p>
105 <p>Password: <input type="password" name="pass1" size="10" maxlength="20" value="<?php if (isset
106 ($_POST['pass1'])) echo $_POST['pass1']; ?>" /></p>
107 <p>Confirm Password: <input type="password" name="pass2" size="10" maxlength="20" value="<?php
108 if (isset($_POST['pass2'])) echo $_POST['pass2']; ?>" /></p>
109 <p><input type="submit" name="submit" value="Register" /></p>
110 </form>
111 <?php include ('includes/footer.html'); ?>
```

假设`mysqli_oop_connect.php`在这个目录的上层目录，此代码将正常工作。如果你的目录结构不同，相应改变文件的引用。

(3) 将所有的`mysqli_real_escape_string()`使用方式改为`$mysqli->real_escape_string()`。

```
$fn = $mysqli->real_escape_string(trim($_POST['first_name']));
$ln = $mysqli->real_escape_string(trim($_POST['last_name']));
$e = $mysqli->real_escape_string(trim($_POST['email']));
$p = $mysqli->real_escape_string(trim($_POST['pass1']));
```

4条数据(所有的字符串),被转义以便保证查询安全。由于该脚本通过面向对象的方法使用MySQL的Improved扩展,因而这4条代码要进行相应的改变。

(4) 更新查询执行的方式(52行是原始脚本)。

```
$mysqli->query($q);
```

要使用面向对象的方式来执行数据库查询,调用该对象的`query()`方法,并为它提供要执行查询语句。

(5) 更改确认查询的代码(原53行)。

```
if ($mysqli->affected_rows == 1) {
```

原来版本的脚本使用结果变量来确认查询是否正确执行:

```
if ($r) {
```

这里,条件语句能更正式地断言查询影响的行数等于1。

(6) 使用对象更新调试错误消息(原脚本第66行)。

```
echo '<p>' . $mysqli->error . '

Query: ' . $q . '</p>';
```

不需要调用`mysqli_error()`函数,对象的`error`属性将会存储数据报告的问题。

(7) 最后,在关闭数据库连接的地方改变两个实例。

```
$mysqli->close();
unset($mysqli);
```

第一行关闭该连接。第二行删除存在的变量。这个步骤释放使用的内存,但这并不是必须的,但是这样做更专业。

原来的脚本在两个地方关闭了数据库连接,这两个地方都要更新。

(8) 保存该脚本,将其放置在Web目录中,并在Web浏览器中测试(参见图16-4)。



图16-4 如果所有的代码都被适当的更新了,注册脚本会像以前那样正常工作

### 16.2.3 获取结果

上一节演示了如何执行“简单”的查询(那些我归为不返回结果行的查询)。如果执行`SELECT`查询时,代码会有一点不同,因为你必须要处理查询的结果。首先,在建立`MySQLi`对象后,使用`query()`运行数据库查询。如果查询返回一个结果集,将调用方法返回的结果赋给一个变量:

```
$q = 'SELECT * FROM tablename';
$result = $mysqli->query($q);
```

`$result` 变量将会是一个 `MySQLi_Result` 类型的对象。就像有些函数返回字符串或整数, `MySQLi::query()` 将返回 `MySQLi_Result` 对象。它的 `num_rows` 属性将反映查询结果中的记录数:

```
if ($result->num_rows > 0) { // Handle the results.
```

如果你的查询只返回一行, 可以直接调用 `fetch_array()` 方法获取:

```
$row = $result->fetch_array();
```

这个方法类似过程式的 `mysqli_fetch_array()`, 接受一个常量作为可选参数来指定返回的行是否应该被看作关联数组 (`MYSQLI_ASSOC`)、索引数组 (`MYSQLI_NUM`) 或两个都是 (`MYSQLI_BOTH`)。`MYSQLI_BOTH` 是默认值。

当你要获取多个记录时, 可以使用循环:

```
while ($row = $result->fetch_array (MYSQLI_NUM)) {
 // Use $row.
}
```

在这段代码中, 循环中的 `$row` 将会是一个数组, 意味着你可以使用 `$row[0]` 或 `$row['column']` 来访问单独的列 (前掉是使用的常量正常)。如果你真的很喜欢 OOP 语法, 可以使用 `fetch_object()` 方法来代替, 从而创建一个对象, 而不是数组:

```
$q = 'SELECT user_id, first_name FROM users';
$result = $mysqli->query($q);
while ($row = $result->fetch_object()) {
 // Use $row->user_id
 // Use $row->first_name
}
```

一旦使用完结果, 应该释放它们占用的资源:

```
$result->free();
```

接下来, 运用上述知识更新 `view_users.php` (参见脚本 9-6)。

### 获取查询结果

(1) 在文本编辑器或 IDE 中开 `view_users.php` (参见脚本 9-6)。

(2) 修改 MySQL 连接脚本的包含文件 (参见脚本 16-3)。

注意, 设置的路径要正确。

### 脚本 16-3 这个脚本使用 MySQLi 和 MySQLi\_Result 类获取数据的记录

```
1 <?php # Script 16.3 - view_users.php
2 // This script retrieves all the records from the users table.
3 // This is an OOP version of the script from Chapter 9.
4
5 $page_title = 'View the Current Users';
6 include ('includes/header.html');
7
8 // Page header:
```

```

9 echo '<h1>Registered Users</h1>';
10
11 require ('../mysqli_oop_connect.php'); // Connect to the db.
12
13 // Make the query:
14 $q = "SELECT CONCAT(last_name, ' ', first_name) AS name, DATE_FORMAT (registration_date, '%M %d,
 %Y') AS dr FROM users ORDER BY registration_date ASC";
15 $r = $mysqli->query($q); // Run the query.
16
17 // Count the number of returned rows:
18 $num = $r->num_rows;
19
20 if ($num > 0) { // If it ran OK, display the records.
21
22 // Print how many users there are:
23 echo "<p>There are currently $num registered users.</p>\n";
24
25 // Table header.
26 echo '<table align="center" cellspacing="3" cellpadding="3" width="75%">
27 <tr><td align="left">Name</td><td align="left">Date Registered</td></tr>
28 ';
29
30 // Fetch and print all the records:
31 while ($row = $r->fetch_object()) {
32 echo '<tr><td align="left">' . $row->name . '</td><td align="left">' . $row->dr . '</td></tr>
33 ';
34 }
35
36 echo '</table>'; // Close the table.
37
38 $r->free(); // Free up the resources.
39 unset($r);
40
41 } else { // If no records were returned.
42
43 echo '<p class="error">There are currently no registered users.</p>';
44
45 }
46
47 // Close the database connection.
48 $mysqli->close();
49 unset($mysqli);
50
51 include ('includes/footer.html');
52 ?>
```

(3) 改变查询的执行方式 (原脚本14行)。

```
$r = $mysqli->query($q);
```

不管正在执行的查询是什么类型，都调用相同的MySQLi::query()方法。在这里，查询执行的结果会分配给一个新的变量，它将是MySQLi\_Result类型的对象。

为了简便起见，我将此变量称为\$r，但你也可以使用更正式的名字\$result。

(4) 修改确定返回的行数的方式（原脚本第17行）。

```
$num = $r->num_rows;
```

结果对象的num\_rows属性反映了查询返回记录的数目。像前面一样，将这个值赋给\$num变量。

注意，这是个属性，而不是方法（是\$r->num\_rows，而不是\$r->num\_rows()）。

(5) 更改读取结果的while循环。

```
while ($row = $r->fetch_object()) {
```

这里的变化是，调用MySQLi\_Result对象的fetch\_object()函数，而不是mysqli\_fetch\_array()。

(6) 在while循环中，改变每列的值的打印方式。

```
echo '<tr><td align="left">' . $row->name . '</td><td align="left">' . $row->dr . '</td></tr>';
```

由于\$row变量现在是一个对象，对象表示法，而不是数组表示法，因此必须使用下面的语法引用每行的列：\$row->name和\$row->dr，而不是\$row['name']和\$row['dr']。

(7) 更改资源释放的方式。

```
$r->free();
unset($r);
```

调用MySQLi\_Result对象的free()方法，释放返回结果占用的内存。此外，在脚本中只要对象不再被使用，就可以释放它。

(8) 更新关闭数据库连接的方式。

```
$mysqli->close();
unset($mysqli);
```

(9) 保存该脚本，将其放到Web目录，并在Web浏览器中测试它（参见图16-5）。

Registered Users	
There are currently 28 registered users.	
Name	Date Registered
Ullman, Larry	March 31, 2011
Isabella, Zoe	March 31, 2011
Starr, Ringo	March 31, 2011
Harrison, George	March 31, 2011
McCartney, Paul	March 31, 2011
Lennon, John	March 31, 2011
Chabon, Michael	March 31, 2011
Brautigan, Richard	March 31, 2011
Banks, Russell	March 31, 2011
Simpson, Homer	March 31, 2011

图16-5 面向对象版本的view\_users.php（参见脚本16-3），结果看起来与原来的过程式的版本相同

### ✓ 提示

- 使用`fetch_object()`方法的真正好处是，你会有一个特定类型的对象作为获取的结果。例如，假设你已经在PHP中定义了一个`Car`类和从数据库中提取所有汽车信息的脚本。在PHP脚本中，每个记录都可以作为`Car`类型的对象获取。这样做之后，你会创建PHP的`Car`对象，它的数据由数据库记录构成，但仍然可以调用`Car`类中的方法。

## 16.2.4 预处理语句

第13章介绍了另一种执行查询的方法：使用预处理语句。预处理语句可以提供比标准的查询方法更高的安全性，甚至可能是更好的性能。当然，你可以使用MySQL Improved扩展为对象执行预处理语句。步骤是相同的：在创建`MySQLi`对象（并且因此创建了数据库连接）后，你可以

- 准备查询
- 绑定的参数
- 执行查询

实际的代码看起来是这样的：

```
$q = 'INSERT INTO tablename (this, that) VALUES (?, ?)';
$stmt = $mysqli->prepare($q);
$stmt->bind_param('si', $this, $that);
$this = 'Larry';
$that = 234;
$stmt->execute();
```

`MySQLi :: prepare()`方法返回`MySQLi_STMT`类型的对象。该对象有几个关键属性。

- `affected_rows`存储了语句影响的行数，一般适用于`INSERT`、`UPDATE`和`DELETE`查询。
- `num_rows`反映了`SELECT`查询的结果集中的记录数目。
- `insert_id`存储上一个`INSERT`查询自动生成的ID值。
- `error`表示任何可能发生的错误。

在执行完预处理语句后，你应该关闭这个语句：

```
$stmt->close();
```

下面运用上述信息更新`post_message.php`（参见脚本13-6）。这是个独立的脚本，可以绑定到任何其他脚本，无论使用还是不使用论坛数据库，在第13章还是本章。

### 执行预处理语句

- (1) 在文本编辑器或IDE中打开`post_message.php`（参见脚本13-6）。
- (2) 更改创建数据库连接的脚本（参见脚本16-4）。

### 脚本 16-4 这个脚本的新版本中，`MySQLi_STMT`类用于执行预处理语句

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Post a Message</title>
```

```
6 </head>
7 <body>
8 <?php # Script 16.4 - post_message.php
9 // This is an OOP version of the script from Chapter 13.
10
11 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
12
13 // Validate the data (omitted)!
14
15 // Connect to the database:
16 $mysqli = new MySQLi('localhost', 'username', 'password', 'forum');
17 $mysqli->set_charset('utf8');
18
19 // Make the query:
20 $q = 'INSERT INTO messages (forum_id, parent_id, user_id, subject, body, date_entered) VALUES
21 (?, ?, ?, ?, ?, NOW())';
22
23 // Prepare the statement:
24 $stmt = $mysqli->prepare($q);
25
26 // Bind the variables:
27 $stmt->bind_param('iiss', $forum_id, $parent_id, $user_id, $subject, $body);
28
29 // Assign the values to variables:
30 $forum_id = (int) $_POST['forum_id'];
31 $parent_id = (int) $_POST['parent_id'];
32 $user_id = 3; // The user_id value would normally come from the session.
33 $subject = strip_tags($_POST ['subject']);
34 $body = strip_tags($_POST['body']);
35
36 // Execute the query:
37 $stmt->execute();
38
39 // Print a message based upon the result:
40 if ($stmt->affected_rows == 1) {
41 echo '<p>Your message has been posted.</p>';
42 } else {
43 echo '<p style="font-weight: bold; color: #C00">Your message could not be posted.</p>';
44 }
45
46 // Close the statement:
47 $stmt->close();
48 unset($stmt);
49
50 // Close the connection:
51 $mysqli->close();
52 unset($mysqli);
53
54 } // End of submission IF.
55
56 // Display the form:
57 ?>
58 <form action="post_message.php" method="post">
```

```

59 <fieldset><legend>Post a message: </legend>
60 <p>Subject: <input name="subject" type="text" size="30" maxlength="100" /></p>
61
62 <p>Body: <textarea name="body" rows="3" cols="40"></textarea></p>
63
64 </fieldset>
65 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
66 <input type="hidden" name="forum_id" value="1" />
67 <input type="hidden" name="parent_id" value="0" />
68
69
70 </form>
71 </body>
72 </html>

```

以前版本的脚本没有使用独立的连接脚本，本脚本也不会使用。请确保你用于连接到服务器的论坛数据库的值是正确的。

(3) 改变查询的预处理方式（原脚本的第21行）。

```
$stmt = $mysqli->prepare($q);
```

`MySQLi::prepare()`方法预处理语句，接受查询作为它唯一的参数。它返回`MySQLi_STMT`类型的对象，并赋给了`$stmt`。

(4) 将绑定的参数更改为。

```
$stmt->bind_param('iiiss', $forum_id, $parent_id, $user_id, $subject, $body);
```

此代码的变化是将`mysqli_stmt_bind_param($stmt...)`改为`$stmt->bind_param(...)`。方法的第一个参数是数据类型遵循的指示符。后面的参数是查询的占位符绑定的PHP变量。

(5) 更新语句的执行方式。

```
$stmt->execute();
```

(6) 更改测试执行是否成功的条件语句。

```
if ($stmt->affected_rows == 1) {
```

要确认`INSERT`查询成功，检查查询影响的行数，这里引用了`MySQLi_STMT`对象的`affected_rows`属性。

(7) 使用下面代码更改错误报告。

```
echo '<p>' . $stmt->error . '</p>';
```

此时，在脚本中，最有可能的错误将是，可能为了一个值必须唯一的列使用了重复的值。如果在查询中有一个语法错误，它将存储在预处理查询之后的`$mysqli->error`中。

(8) 更新关闭语句。

```
$stmt->close();
unset($stmt);
```

(9) 修改关闭数据库连接的方式。

```
$mysqli->close();
unset($mysqli);
```

(10) 保存该脚本，将其放到Web目录，并在Web浏览器中测试（参见图16-6和图16-7）。

图16-6 发布新消息的HTML表单

图16-7 新消息被成功地存储在数据库中

### 输出参数

在第13章中，post\_message.php脚本演示了如何使用输入参数：使用PHP变量为查询中的占位符赋值。你还可以使用出站参数：将查询返回的值绑定到PHP变量。首先处理查询：

```
$q = 'SELECT this, that FROM tablename';
$stmt = $mysqli->prepare($q);
```

然后将返回的行绑定到变量：

```
$stmt->bind_result($this, $that);
```

接下来，调用MySQLi\_STMT::fetch()方法作为while循环的一部分：

```
while ($stmt->fetch()) {
```

在while循环中，\$this和\$that将会存储每一条记录返回的列。输出参数就像输入参数，不提供额外的安全性，或更高的性能，但是如果你有一个使用预处理语句的查询，使用输入和输出参数将会是有意义的。例如，登录查询：

```
SELECT user_id, first_name FROM users WHERE email='?' AND pass=SHA1('?)
```

你可以使用输入参数代表提交的电子邮件地址和密码，并使用输出参数来从同一个查询中获取用户ID和用户名。

### 16.3 DateTime类

`DateTime`类是在PHP的5.2版本添加的。以代替第11章中介绍的与日期和时间相关的函数，`DateTime`类打包了所有你在操作日期和时间时可能用到的功能。它在转换和对比日期和时间时特别有用。

首先，创建一个新的`DateTime`对象：

```
$dt = new DateTime();
```

如果在创建时没有为构造函数提供任何参数，生成的`DateTime`参数将会代表当前的日期和时间。要创建特定的日期和时间的对象，可以像下面那样提供第一个参数：

```
$dt = new DateTime('2011-04-20');
$dt = new DateTime('2011-04-20 11:15');
```

有许多指定日期和时间的可接受的格式，在PHP手册中查看详情。你也可以在创建对象后使用 `setDate()`和 `setTime()`方法来建立日期和时间。 `setDate()`方法期望按顺序接受所需的年、月和日。 `setTime()`方法接受小时、分钟和可选的秒作为参数。

```
$dt = new DateTime();
$dt->setDate(2001, 4, 20);
$dt->setTime(11, 15);
```

`DateTime`对象将允许建立有效的日期和时间，为无效的日期和时间抛出异常（参见图16-8）：

```
$dt = new DateTime('2011-13-42');
```

```
Fatal error: Uncaught exception 'Exception' with message 'DateTime::__construct() [datetime::__construct\]: Failed to parse time string \(2011-13-42\) at position 6 \(3\): Unexpected character' in /Users/larryullman/Sites/phpmysql4/ch16/test.php:2 Stack trace: #0 /Users/larryullman/Sites/phpmysql4/ch16/test.php\(2\): DateTime->__construct\('2011-13-42'\) #1 {main} thrown in /Users/larryullman/Sites/phpmysql4/ch16/test.php on line 2
```

图16-8 试图以无效的日期和时间创建`DateTime`对象会导致异常

异常是前面没有介绍的一个主题。如果过程式的代码会产生错误，对象会抛出异常（是的，就是抛出）。当你进一步了解面向对象的编程时，将学习如何使用`try... catch`块来“捕获”和处理抛出的异常。`DateTime`构造函数有一个可选的第二个参数，就是使用的时区。如果没有提供，将使用该PHP安装的默认时区。你也可以在实例化后使用 `setTimezone()`来改变时区。注意，无论是 `setTimezone()`方法，还是构造函数，接受的都是`DateTimeZone`对象作为参数，而不是字符串：

```
$tz = new DateTimeZone('America/New_York');
$dt->setTimezone($tz);
```

创建了`DateTime`对象后，可以通过添加或减去时间段来操作它的值。其中一个实现方法是使用 `modify()`方法：

```
$dt->modify('+1 day');
$dt->modify('-1 month');
$dt->modify('next Thursday');
```

你可以提供给这个方法的值非常灵活，并且对应于那些可以用于`strtotime()`函数（可以将字符串转换成时间戳，在PHP手册中查看详情）的值。`add()`方法用于添加时间段表示日期和时间。

它接受`DateInterval`类型的对象作为它唯一的参数：

```
$di = new DateInterval(interval);
$dt->add($di);
```

有一个特别的符号用于设置间隔，它总是以一个代表“期间”的字母P开始。在那之后是整数和期间代号：Y代表年、M代表月、D代表日、W代表周、H代表小时、M代表分钟、S代表秒。你可能想知道字母M为什么能同时代表月和分钟：这是可以的，因为小时、分钟和秒的前面必须有一个字母T来代表时间。这些字符应该按从最大到最小的顺序组合（例如，从年到秒）。下面是一些示例：

- P3W代表三个周；
- P2Y3M代表2年零3个月；
- P2M3DT4H18M43S代表2个月、3天、4小时、18分钟、43秒。

`sub()`方法与`add()`的功能一样，只是从对象中减去时间段：

```
$di = new DateInterval('P2W'); // 2 weeks
$dt->sub($di);
```

`diff()`方法返回`DateInterval`对象，它反映了两个`DateTime`对象之间的时间量：

```
$diff = $dt->diff($dt2);
```

`DateInterval`类定义了一些代表计算间隔的属性：y代表年、m代表月、d代表天、h代表小时、i代表分钟、s代表秒、days也代表天。

最后一个你应该熟悉的`DateTime`类的方法是`format()`，它返回你指定的日期格式：

```
echo $dt->format(format);
```

你可以使用与第11章介绍的`date()`函数相同的字符来进行格式化。

为了演示所有这些信息，下面的脚本将执行一个很多网站都需要任务：允许用户输入两个日期来创建一个范围（参见图16-9）。这个脚本将会为提交的日期执行最高质量的验证，并且计算它们之间的天数（参见图16-10）。提供的信息将很容易应用，比方说，一个酒店登记系统或类似的系统。该脚本将使用很多刚刚介绍的技术，并且会比较两个`DateTime`对象，PHP5.2.2以上版本可用。

图16-9 用于输入两个特定格式的日期的简单表单

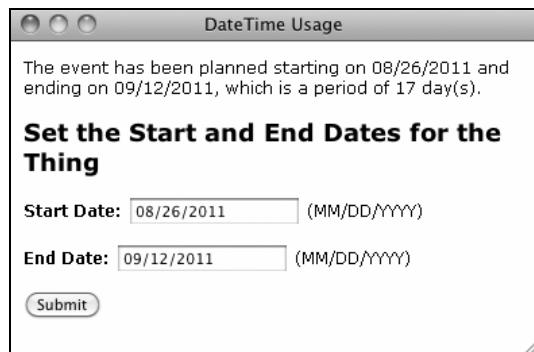


图16-10 如果提交的两个日期是有效的，并且结束日期比开始日期晚，会再次显示这两个日期和计算出的间隔

### 使用DateTime类

(1) 在文本编辑器或IDE中创建新的PHP脚本，命名为datetime.php，以HTML(参见脚本16-5)开始。

#### 脚本 16-5 仿效通常的日期选择功能，这个脚本接受并验证两个日期

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>DateTime Usage</title>
6 <style type="text/css" media="screen">
7 body {
8 font-family: Verdana, Arial, Helvetica, sans-serif;
9 font-size: 12px;
10 margin: 10px;
11 }
12 label { font-weight: bold; }
13 .error { color: #F00; }
14 </style>
15 </head>
16 <body>
17 <?php # Script 16.5 - datetime.php
18
19 // Set the start and end date as today and tomorrow by default:
20 $start = new DateTime();
21 $end = new DateTime();
22 $end->modify('+1 day');
23
24 // Default format for displaying dates:
25 $format = 'm/d/Y';
26
27 // This function validates a provided date string.
28 // The function returns an array--month, day, year--if valid.
29 function validate_date($date) {
30

```

```

31 // Break up the string into its parts:
32 $date_array = explode('/', $date);
33
34 // Return FALSE if there aren't 3 items:
35 if (count($date_array) != 3) return false;
36
37 // Return FALSE if it's not a valid date:
38 if (!checkdate($date_array[0], $date_array[1], $date_array[2])) return false;
39
40 // Return the array:
41 return $date_array;
42
43 } // End of validate_date() function.
44
45 // Check for a form submission:
46 if (isset($_POST['start'], $_POST['end'])) {
47
48 // Call the validation function on both dates:
49 if ((list($sm, $sd, $sy) = validate_date($_POST['start'])) && (list($em, $ed, $ey) =
50 validate_date($_POST['end']))) {
51
52 // If it's okay, adjust the DateTime objects:
53 $start->setDate($sy, $sm, $sd);
54 $end->setDate($ey, $em, $ed);
55
56 // The start date must come first:
57 if ($start < $end) {
58
59 // Determine the interval:
60 $interval = $start->diff($end);
61
62 // Print the results:
63 echo "<p>The event has been planned starting on {$start->format($format)} and ending on{$end->
64 format ($format)}, which is a period of $interval->days day(s).</p>";
65
66 } else { // End date must be later!
67 echo '<p class="error">The starting date must precede the ending date.</p>';
68 }
69
70 } else { // An invalid date!
71 echo '<p class="error">One or both of the submitted dates was invalid.</p>';
72 }
73
74 } // End of form submission.
75
76 // Show the form:
77 ?>
78 <h2>Set the Start and End Dates for the Thing</h2>
79 <form action="datetime.php" method="post">
80
81 <px><label for="start_date">Start Date:</label> <input type="text" name="start_date" value=
82 "<?php echo $start->format($format); ?>" /> (MM/DD/YYYY)</p>
83 <px><label for="end_date">End Date:</label> <input type="text" name="end_date" value="<?php echo
84 $end->format($format); ?>" /> (MM/DD/YYYY)</p>

```

```

81 <p><input type="submit" value="Submit" /></p>
82 </form>
83 </body>
84 </html>

```

(2) 添加一点CSS。

```

<style type="text/css" media="screen">
body {
 font-family: Verdana, Arial, Helvetica, sans-serif;
 font-size: 12px;
 margin: 10px;
}
label { font-weight: bold; }
.error { color: #F00; }
</style>

```

在这里只有error类在功能上是有重要意义的。它会以红色文字显示错误消息。

(3) 完成head并开始body和PHP部分。

```

</head>
<body>
<?php # Script 16.5 - datetime.php

```

(4) 创建两个DateTime对象。

```

$start = new DateTime();
$end = new DateTime();

```

首先创建两个DateTime对象，无论表单是否已经提交，它们都将使用当前日期和时间实例化。接下来，它们中的一个或两个都会被赋给新的值。

(5) 为结束日期添加一天。

```

$end->modify('+1 day');

```

默认情况下，当第一次加载页面时，表单将预设以今天作为开始日期、以明天作为结束日期。要确定结束日期，只需将对象的当前值增加一天即可。

使用DateInterval对象和DateTime::add()方法，也可以实现上述功能：

```

$day = new DateInterval('P1D');
$end->add($day);

```

(6) 建立日期显示的默认格式。

```

$format = 'm/d/Y';

```

该脚本将在4个地方显示格式化的日期。首先将格式（MM/DD/YYYY）赋给一个变量，以便在以后有需要的时候可以很容易地修改。

(7) 开始定义一个函数。

```

function validate_date($date) {
 $array = explode('/', $date);

```

两个提交的日期都需要验证，无论何时当你在脚本或应用中需要使用重复代码时，都可以创建一

个函数执行重复的代码。这个函数接受日期字符串（不是DateTime对象）作为它唯一的参数。这个字符串将会是用户提交的值，例如08/08/2011。函数做的第一件事就是使用explode()将字符串打散成三部分，月、天和年。结果数组将被赋给\$array变量。

(8) 如果数组不包含3个元素，则返回FALSE。

```
if (count($array) != 3) return false;
```

函数做的第一件事就是确定它有3个离散的值可以使用，代表月、天和年。如果数组不包含3个元素，该函数返回FALSE，表示无效的日期。第(7)步中explode()行和这一行会让提交的任何不符合模式X/Y/Z的值无效（虽然它可能仍然是猫/狗/斑马）。

需要注意的是，通常我会建议在条件语句中总是使用大括号，但为了让代码尽可能的短并保证整个构造函数在同一行上，我省略了它们。同样要注意，只要函数执行return语句，函数就退出了。

(9) 如果提供的日期不是一个有效的日期，返回FALSE。

```
if (!checkdate($array[0], $array[1], $array[2])) return false;
```

与第(8)步类似，此代码调用PHP的checkdate()函数，以确认所提供的日期确实存在。如果日期不存在，如13/43/2011，函数再次返回FALSE。

(10) 返回日期数组并完成函数。

```
return $array;
} // End of validate_date() function.
```

如果所提供的日期是正确的格式并对应于存在的日期，函数返回日期元素数组。

(11) 如果表单已被提交，验证用户提交的值。

```
if (isset($_POST['start'], $_POST['end'])) {
if ((list($sm, $sd, $sy) = validate_date($_POST['start'])) && (list($em, $ed, $ey)
=validate_date($_POST['end']))) {
```

如果这两个变量被设置，意味着表单被提交了，两个都通过了validate\_date()函数。如果那个函数对任何一个日期返回了FALSE，这个条件语句将返回FALSE。如果函数为两个日期返回了一个数组，赋给对应的月、天和年变量，然后就可以确定并显示结果。

(12) 将日期重置为用户提交的日期。

```
$start->setDate($sy, $sm, $sd);
$end->setDate($ey, $em, $ed);
```

因为目前所提供的日期是有效的，这两个对象可以被更新，以表示用户输入的日期。要做到这一点，需要调用 setDate()函数，为它提供特定的值。

(13) 如果结束日期在开始日期之后，计算它们之间的间隔。

```
if ($start < $end) {
$interval = $start->diff($end);
```

比较两个DateTime对象同比较两个数字一样，看一个是大于还是小于另外一个。如果结束日期比较晚，那么在一个对象上调用diff()方法并将另外一个作为它的参数来计算两个日期的间隔。将结果赋给\$interval变量，它将是一个DateInterval类型的对象。

(14) 打印结果。

```
echo "<p>The event has been planned starting on {$start->format($format)} and ending on {$end->format($format)}, which is a period of $interval->days day(s).</p>";
```

最后，显示结果（参见图16-10）。你可以看到，在引号内调用对象的方法是可行的，因此打印那个函数调用的输出，但是你必须使用花括号来包括整个结构。引用属性（如\$interval->days），不需要花括号。

(15) 完成在步骤(11)和步骤(13)中开始的条件语句。

```
} else { // End date must be later!
 echo '<p class="error">The starting date must precede the ending date.</p>';
}
} else { // An invalid date!
 echo '<p class="error">One or both of the submitted dates was invalid.</p>';
}
```

如果两个日期都是有效的，但是结束日期不在开始日期之后（参见图16-11），第一个else子句就会起作用。如果提交的日期没有一个通过validated\_date()验证，第二个else会起作用。在这种情况下，两个日期都将保持默认的设置（参见图16-12）。

图16-11 提供的开始日期实际上在结束日期之后的结果

图16-12 提交的两个日期都不是有效的日期的结果

(16) 完成表单提交的条件语句，闭合PHP代码块，并开始HTML表单。

```
} // End of form submission.
?>
<h2>Set the Start and End Dates for the Thing</h2>
<form action="datetime.php" method="post">
```

(17) 为日期创建两个输入框。

```
<p><label for="start">Start Date:</label> <input type="text" name="start" value=<?php echo $start->format($format); ?>" /> (MM/DD/YYYY)</p>
<p><label for="end">End Date: </label> <input type="text" name="end" value=<?php echo $end->format($format); ?>" /> (MM/DD/YYYY)</p>
```

对于每一个输入框，值是通过调用对应对象的`format()`方法来预设的。输入的日期需要的格式也在随后说明中指出了。

(18) 完成表单和HTML页面。

```
<p><input type="submit"
value="Submit" /></p>
</form>
</body>
</html>
```

(19) 将脚本保存为`datetime.php`，将其放置在Web目录，并在Web浏览器中测试。

当你运行这个脚本时，如果看到系统时区设置的异常错误，在创建`DateTime`对象之前调用`date_default_timezone_set()`，如第11章所述。

#### ✓ 提示

- `DateTime::getTimestamp()`方法返回的UNIX时间戳表示的日期和时间。
- `DateTime`类内部使用一个64位的数字代表日期和时间，它可以表示从现在算起的约292亿年之前到约292亿年之后的日期。
- `DateTime`类中的一些常量可以表示常用的日期时间格式，例如`DateTime::COOKIE`。
- `DateTime`方法也有过期版本。

## 16.4 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

### 16.4.1 回顾

- 什么是类？什么是方法？类中定义的变量称为什么？
- 什么是对象？如何在PHP中创建对象？你如何调用对象的方法？
- 什么是构造函数？
- 创建`MySQLi`对象的语法是什么？
- 如何使用`MySQLi`执行任何种类的查询？
- 在使用`MySQLi`时，如何保证查询中的字符串数据的安全？提示：有两个答案。
- 如何检查并显示`MySQLi`错误？
- 如何使用`MySQLi`（或其他）对象获取`SELECT`查询的结果？`MySQLi_Result::fetch_array()`和`MySQLi_Result::fetch_object()`之间的区别是什么？

- 如何使用MySQLi和MySQLi\_STMT的类执行预处理语句？
- 创建新的DateTime对象的语法是什么？用于设置对象的日期和时间的两种方法是什么？
- 什么是异常？

#### 16.4.2 实践

- 如果你有兴趣了解更多关于面向对象编程的知识，可以考虑读读面向对象编程主题的图书或教程，不要针对任何特定的编程语言。
- 查看PHP手册文档中PHP的面向对象编程的相关资料（[www.php.net/oop](http://www.php.net/oop)）。
- 如果你还不清楚为什么需要对查询中用到的字符串数据使用real\_escape\_string()，温习第9章。
- 使用MySQLi改写第9章和第10章中的其他一些脚本。
- 阅读PHP手册中关于DateTime类的文档（[www.php.net/datetime](http://www.php.net/datetime)）。
- 在PHP手册中学习strtotime()函数（[www.php.net/ strtotime](http://www.php.net/strtotime)）。
- 加分题：运用前一章提供的信息，使用jQuery UI的Datepicker工具，为datetime.php脚本创建两个漂亮的JavaScript日期选择器。



### 本章内容

- 建立数据库
- 编写模板
- 创建索引页面
- 创建论坛页面
- 创建论点页面
- 发布消息
- 回顾和实践

**论**坛的功能实际上相当简单：用户可以发新帖开始一个新主题，也可以发帖回复已有的主题；帖子会被添加到数据库中，然后显示在页面上。基本上就是这些。当然，有时实现简单的概念可能会相当困难！

为了使这个示例更令人兴奋、更有用，它将不仅仅是一个论坛，而是一个支持多种语言的论坛。每种语言都将有它自己的论坛，并且所有的关键元素（导航、提示、介绍性文字等）都将用相应语言显示。它将非常酷，会实际应用第14章中介绍的知识。

为了集中讨论这个Web应用程序的最重要的方面，我将省略其他一些方面。省略的三个显著的方面是：用户管理、错误处理和站点管理。不过，这对于你应该不是一个问题，因为下一章将非常详细地讲述用户管理和错误处理。实际上一章中的所有内容都可以应用于这个示例。至于站点管理，我将在本章末给出一些建议。

## 17.1 建立数据库

自然，第一步是创建数据库。第6章中开发了一个示例论坛数据库。尽管该数据库非常好，这里将改用它的一个变体。图17-1显示了第6章中的那个数据库中的表和关系。图17-2显示了这个新数据库中的表和关系。我将比较这两个数据库，以更好地解释我的想法。

首先，用languages表代替forums表。这两个表都服务于相同的目的：允许多个论坛。在这个新数据库中，每个论坛中的主题（PHP and MySQL for Dynamic Web Sites）都将是相同的，但是每个论坛都将使用不同的语言，每个论坛中的帖子也会有所区别（这将不是用多种语言翻译相同的论坛）。languages表在它自己的字母表中用英语存储语言的名称，以给管理员提供方便。（这里假定英语是管理员的母语。）

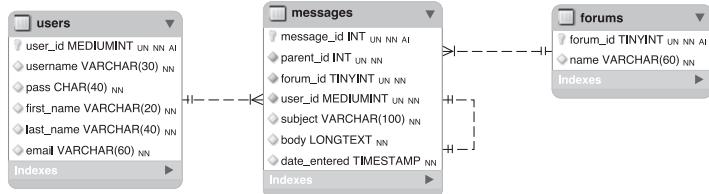


图17-1 用于第6章中开发的论坛数据库的模型

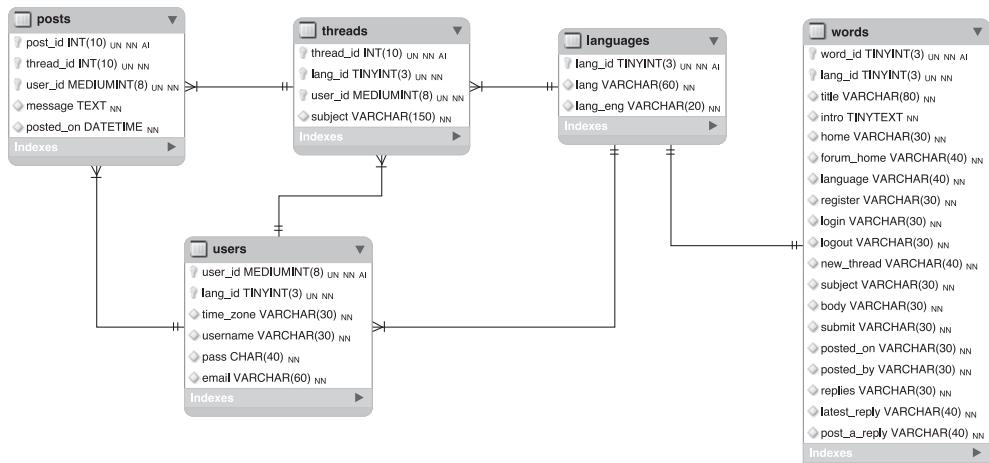


图17-2 用于本章中将使用的论坛数据库的修改过的模型

这个新数据库中的threads表起到的是老数据库中的messages表的作用，它们之间有一个重大的区别。就像老的messages表与forums表相关联一样，threads表则与languages和users表相关联（每条消息只能在一个论坛中并且只能由一个用户发布；每个论坛都可以具有多条消息，并且每个用户都可以发布多条消息）。不过，这个threads表将只存储主题，而不是消息本身。我做出这种改变有几个原因。第一，使一个主题对于每个回复（根据我的经验，不管怎样，回复几乎总是具有相同的主题）都重复出现多次是不必要的。第二，对于lang\_id关联也是如此（它不必出现在每个回复中，只要每个回复都关联单个论点即可）。第三，我将改变在这个数据库中指示论点的层次结构的方式（你将在下一个段落中看到怎样完成这个任务），并且更改表结构可以在这个方面提供帮助。最后，每当用户查看论坛中的帖子时，都将使用threads表。从该表中删除消息正文将改进这些查询的性能。

转移到posts表上，它的唯一目的是存储与论点关联的消息的正文。在第6章的数据库中，messages表具有一个parent\_id列，用于指示新消息响应的是哪条消息。它是分层的：消息3可能是起始帖子，消息18可能是对消息3的响应，消息20是对消息18的响应，等等（参见图17-3）。数据库的这个版本更直接地指示了响应，这个版本将只存储消息归属的论点。因此，消息18和消息20使用的thread\_id都是3。这使得可以更高效地显示论点（就PHP和MySQL的要求而言），并且仍然可以使用发布每条消息的日期/时间对它们进行排序。

message_id	parent_id
3	0
4	0
18	3
19	4
20	18

图17-3 如何使用旧的数据库模式指示消息之间的关系

这三个表提供了大部分论坛功能。数据库还需要一个users表。在我的论坛中，只有注册用户可以发布消息，我认为这是一个非常好的策略（它杜绝了垃圾邮件和黑客攻击）。注册用户还可以指定他们默认的语言（通过languages表）和时区，以便给他们提供更个性化的体验。他们的用户名和密码的组合将用于登录。

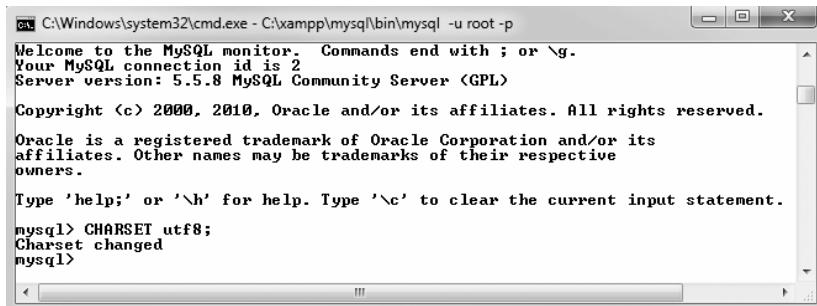
还需要最后一个表words，以便使站点采用多种语言。这个表将存储公共元素（导航链接、表单提示、标题等）的译文。站点中的每种语言都在这个表中具有一条记录。这是一个美妙的特性，并且使用起来极其容易。毋庸置疑，这个表中列出的单词也可能出现在languages表中，不过，其隐含意思是这些单词也将与threads表相关联，但是事实并非如此。

这就是这个新数据库设计背后隐藏的思想。在下面的步骤中创建表时你将学习到更多知识。与本书中的其他示例一样，也可以从本书对应的Web站点下载用于本章的SQL代码以及这些步骤中使用的SQL代码。

### 建立数据库

(1) 访问MySQL服务器，并设置要用于通信的字符集（参见图17-4）。

```
CHARSET utf8;
```



The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p'. The MySQL monitor is open, displaying the welcome message and server version. The user has entered the command 'CHARSET utf8;' followed by a semicolon, and the response 'Charset changed' is shown.

```
C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u root -p
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.8 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CHARSET utf8;
Charset changed
mysql>
```

图17-4 为了在我的查询中使用Unicode数据，需要从mysql客户更改用于同MySQL通信的字符集

一如既往，我将在图片中使用MySQL客户端，但是你可以使用自己喜欢的任何工具。不过，第一步必须为将要执行的查询把字符集更改为UTF-8，否则查询中的一些字符将在数据库中被存储为无意义的数据（参见框注“怪异的字符”）。

### 怪异的字符

在实现本章的示例时，如果你看到怪异的字符（方框、数字代码或问号）而没有看到实际的语言字符，这可能有几个原因。引发这个问题的原因是编码不匹配（数据库术语为字符集不匹配）。

计算机显示字符的能力依赖于文件的编码和操作系统支持的字符（即字体）。首先，这意味着每个PHP或HTML页面都必须使用正确的编码。其次，MySQL中的数据库必须使用正确的编码（如用于创建数据库的步骤中所指示的那样）。最后，PHP与MySQL之间的通信也必须使用正确的编码，这可能是问题的常见原因。我在`mysqli_connect.php`脚本中解决了这个问题（参见第一个提示）。最后，如果你使用MySQL客户端、phpMyAdmin或另一个工具填充数据库，该交互也必须使用正确的编码。

(2) 创建新的数据库（参见图17-5）。

```
CREATE DATABASE forum2 CHARACTER SET utf8;
USE forum2;
```

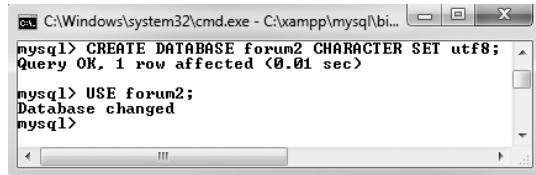


图17-5 为这个示例创建和选择数据库。这个数据库使用UTF-8字符集，使得它可以支持多种语言

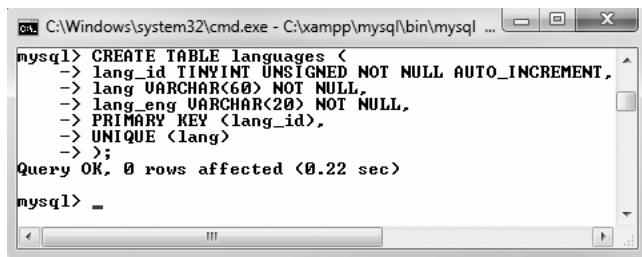
为了不让在原始论坛数据库（在第6章中）中创建的表把事情弄得混乱不堪，将创建一个新数据库。

如果使用托管站点并且不能创建自己的数据库，可使用为你提供的数据库并选择它。如果现有数据库中的表具有这些相同的名称——words、languages、threads、users和posts，可重命名这些表（现有的表或新表），并且相应地更改本章余下部分中的代码。

无论是从头开始创建这个数据库还是使用新的数据库，让这些表使用UTF-8编码都非常重要，这样就能够支持多种语言（参见第14章，了解更多信息）。如果你使用现有的数据库，并且不希望通过为所有表更改字符集而潜在地引发问题，只需在每个表定义中添加CHARACTER SET utf8子句即可（第(3)~(7)步）。

(3) 创建languages表（参见图17-6）。

```
CREATE TABLE languages (
lang_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
lang VARCHAR(60) NOT NULL,
lang_eng VARCHAR(20) NOT NULL,
PRIMARY KEY (lang_id),
UNIQUE (lang)
);
```



```
mysql> CREATE TABLE languages (
-> lang_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
-> lang VARCHAR(60) NOT NULL,
-> lang_eng VARCHAR(20) NOT NULL,
-> PRIMARY KEY (lang_id),
-> UNIQUE (lang)
->);
Query OK, 0 rows affected (0.22 sec)

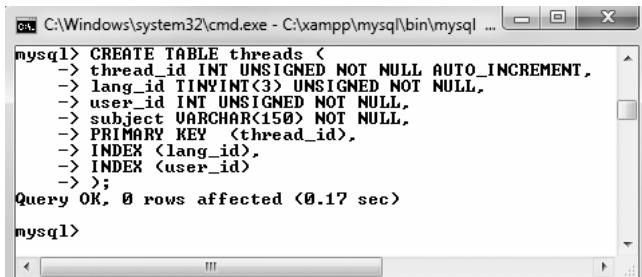
mysql>
```

图17-6 创建languages表

这是一组表中最简单的表。将不会表示许多语言，因此主键（`lang_id`）可以是TINYINT类型。`lang`列被定义得更大一点，因为它将存储其他语言中的字符，它可能需要更多空间。这一列也必须是唯一的。注意：我不能把这一列称为“language”，因为它是MySQL中的保留关键字（实际上，我仍然可以这样命名它，你会在第(7)步看到需要做的事情）。`lang_eng`列指示使用的语言是英语，使得管理员可以轻松看到使用的是哪一种语言。

(4) 创建threads表（参见图17-7）。

```
CREATE TABLE threads (
thread_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
lang_id TINYINT(3) UNSIGNED NOT NULL,
user_id INT UNSIGNED NOT NULL,
subject VARCHAR(150) NOT NULL,
PRIMARY KEY (thread_id),
INDEX (lang_id),
INDEX (user_id)
);
```



```
mysql> CREATE TABLE threads (
-> thread_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> lang_id TINYINT(3) UNSIGNED NOT NULL,
-> user_id INT UNSIGNED NOT NULL,
-> subject VARCHAR(150) NOT NULL,
-> PRIMARY KEY (thread_id),
-> INDEX (lang_id),
-> INDEX (user_id)
->);
Query OK, 0 rows affected (0.17 sec)

mysql>
```

图17-7 创建threads表。这个表存储主题，并将它们与语言（即论坛）相关联

`threads`表包含4列，并且与`languages`和`users`表相关联（分别通过`lang_id`和`user_id`外键）。在此，`subject`要足够长，以便存储多种语言中的主题（在其他语言中，字符要占据更多的空间）。

在联结和WHERE子句中使用的列（`lang_id`和`user_id`）将建立索引，`thread_id`也是如此（作为主键，它也要建立索引）。

(5) 创建posts表（参见图17-8）。

```
CREATE TABLE posts (
post_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```

thread_id INT UNSIGNED NOT NULL,
user_id INT UNSIGNED NOT NULL,
message TEXT NOT NULL,
posted_on DATETIME NOT NULL,
PRIMARY KEY (post_id),
INDEX (thread_id),
INDEX (user_id)
);

```

```

mysql> CREATE TABLE posts (
-> post_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> thread_id INT UNSIGNED NOT NULL,
-> user_id INT UNSIGNED NOT NULL,
-> message TEXT NOT NULL,
-> posted_on DATETIME NOT NULL,
-> PRIMARY KEY (post_id),
-> INDEX (thread_id),
-> INDEX (user_id)
->);
Query OK, 0 rows affected <0.09 sec>
mysql>

```

图17-8 创建posts表，它链接到threads和users表

这个表中的主要列是message，它存储所有的帖子。有两个列是外键，分别绑定到threads和users表。posted\_on列是DATETIME类型，但是将使用UTC ( Coordinated Universal Time，协调世界时，参见第6章 )。不过，这里无需为它执行特殊的操作。

#### (6) 创建users表 (参见图17-9)。

```

CREATE TABLE users (
user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
lang_id TINYINT UNSIGNED NOT NULL,
time_zone VARCHAR(30) NOT NULL,
username VARCHAR(30) NOT NULL,
pass CHAR(40) NOT NULL,
email VARCHAR(60) NOT NULL,
PRIMARY KEY (user_id),
UNIQUE (username),
UNIQUE (email),
INDEX login (username, pass)
);

```

```

mysql> CREATE TABLE users (
-> user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
-> lang_id TINYINT UNSIGNED NOT NULL,
-> time_zone VARCHAR(30) NOT NULL,
-> username VARCHAR(30) NOT NULL,
-> pass CHAR(40) NOT NULL,
-> email VARCHAR(60) NOT NULL,
-> PRIMARY KEY (user_id),
-> UNIQUE (username),
-> UNIQUE (email),
-> INDEX login (username, pass)
->);
Query OK, 0 rows affected <0.17 sec>
mysql>

```

图17-9 创建users表的无内容版本

出于简洁性考虑，我将省略你将在这个表中放入的其他一些列，比如注册日期、名字和姓氏。关于像这样创建和使用表的更多信息，可参见下一章的内容。

在考虑这个站点时，我期望用户将选择他们注册时的首选语言和时区，以便他们可以具有更个性化的体验。他们也可以具有用户名，这将显示在帖子中（而不是显示他们的电子邮件地址）。用户名和电子邮件地址必须是唯一的，你需要在注册过程中解决这个问题。

#### (7) 创建words表（参见图17-10）。

```
CREATE TABLE words (
 word_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
 lang_id TINYINT UNSIGNED NOT NULL,
 title VARCHAR(80) NOT NULL,
 intro TINYTEXT NOT NULL,
 home VARCHAR(30) NOT NULL,
 forum_home VARCHAR(40) NOT NULL,
 `language` VARCHAR(40) NOT NULL,
 register VARCHAR(30) NOT NULL,
 login VARCHAR(30) NOT NULL,
 logout VARCHAR(30) NOT NULL,
 new_thread VARCHAR(40) NOT NULL,
 subject VARCHAR(30) NOT NULL,
 body VARCHAR(30) NOT NULL,
 submit VARCHAR(30) NOT NULL,
 posted_on VARCHAR(30) NOT NULL,
 posted_by VARCHAR(30) NOT NULL,
 replies VARCHAR(30) NOT NULL,
 latest_reply VARCHAR(40) NOT NULL,
 post_a_reply VARCHAR(40) NOT NULL,
 PRIMARY KEY (word_id),
 UNIQUE (lang_id)
);
```

```
C:\Windows\system32\cmd.exe - C:\xampp\mysql\bin\mysql -u... [] X
mysql> CREATE TABLE words (
-> word_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
-> lang_id TINYINT UNSIGNED NOT NULL,
-> title VARCHAR(80) NOT NULL,
-> intro TINYTEXT NOT NULL,
-> home VARCHAR(30) NOT NULL,
-> forum_home VARCHAR(40) NOT NULL,
-> `language` VARCHAR(40) NOT NULL,
-> register VARCHAR(30) NOT NULL,
-> login VARCHAR(30) NOT NULL,
-> logout VARCHAR(30) NOT NULL,
-> new_thread VARCHAR(40) NOT NULL,
-> subject VARCHAR(30) NOT NULL,
-> body VARCHAR(30) NOT NULL,
-> submit VARCHAR(30) NOT NULL,
-> posted_on VARCHAR(30) NOT NULL,
-> posted_by VARCHAR(30) NOT NULL,
-> replies VARCHAR(30) NOT NULL,
-> latest_reply VARCHAR(40) NOT NULL,
-> post_a_reply VARCHAR(40) NOT NULL,
-> PRIMARY KEY (word_id),
-> UNIQUE (lang_id)
->);
Query OK, 0 rows affected (0.13 sec)

mysql> _
```

图17-10 创建words表，它存储不同语言中关键单词的表示法

这个表将存储站点上使用的注释元素的不同译文。一些元素（home、forum\_home、language、register、login、logout和new\_thread）将是链接的名称。页面上使用的另外一些元素（subject、body、submit）用于发布消息。还有一类元素是那些在论坛的主页上使用的元素：posted\_on、posted\_by、replies和latest\_reply。

其中一些元素将在站点中使用多次，然而，这仍然是一个不完整的列表。在你自己实现站点时，你将看到可以使用单词定义的其他位置。

对于每一列，其名称暗示了将在该列中存储的值。对于其中一列（language）我使用了MySQL关键字来说明它可以做什么。修正方法是用反引号括住列的名称，使得MySQL不会混淆该列的名称与关键字“language”。

(8) 填充languages表。

```
INSERT INTO languages (lang, lang_eng) VALUES
('English', 'English'),
('Português', 'Portuguese'),
('Français', 'French'),
('Norsk', 'Norwegian'),
('Romanian', 'Romanian'),
('Ελληνικά', 'Greek'),
('Deutsch', 'German'),
('Srpski', 'Serbian'),
('日本語', 'Japanese'),
('Nederlands', 'Dutch');
```

由于提供给我的一些帮助，这里只是站点将表示的一些语言（参见框注“关于译文的说明”）。对于每一种语言，都将存储该语言的母语和英语单词（参见图17-11）。

The screenshot shows a MySQL command-line interface window titled "PHP and MySQL for Dynamic Web Sites...". The user has run the command "SELECT \* FROM languages;". The resulting table output is as follows:

lang_id	lang	lang_eng
1	English	English
2	Português	Portuguese
3	Français	French
4	Norsk	Norwegian
5	Romanian	Romanian
6	Ελληνικά	Greek
7	Deutsch	German
8	Srpski	Serbian
9	日本語	Japanese
10	Nederlands	Dutch

Below the table, it says "10 rows in set (0.00 sec)".

图17-11 填充过的languages表，带有在它自己的字母表中编写的每种语言

#### 关于译文的说明

世界各地的几位读者非常友善，他们给我提供了关键词、名称、消息主题和消息正文的译文。我想对他们的帮助致以最诚挚的感谢，他们是（不分先后）：Angelo（葡萄牙语）、Iris（德语）、Johan（挪威语）、Gabi（罗马尼亚语）、Darko（塞尔维亚语）、Emmanuel和Jean-François（法语）、Andreas

和Simeon（希腊语）、Darius（菲律宾/塔加拉语）、Olaf（荷兰语），以及Tsutomu（日语）。

如果你知道其中任何一种语言，无疑将会看出本文或对应图像中的语言错误。如果是这样，几乎可以肯定是我的错误，这可能是由于我错误地传达了需要翻译的单词或者不正确地把响应输入到数据库中。对于任何这类错误，我提前表达歉意，但是希望你把注意力更多地放在数据库、代码和功能上。我再次感谢所有给我提供帮助的人！

#### (9) 填充users表（参见图17-12）。

```
INSERT INTO users (lang_id, time_zone, username, pass, email) VALUES
(1, 'America/New_York', 'troutster', SHA1('password'), 'email@example.com'),
(7, 'Europe/Berlin', 'Ute', SHA1('pa24word'), 'email1@example.com'),
(4, 'Europe/Oslo', 'Silje', SHA1('2kll13'), 'email2@example.com'),
(2, 'America/Sao_Paulo', 'João', SHA1('fJDLN34'), 'email3@example.com'),
(1, 'Pacific/Auckland', 'kiwi', SHA1('conchord'), 'kiwi@example.org');
```

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> INSERT INTO users (lang_id, time_zone, username, pass, email) VALUES
-> (1, 'America/New_York', 'troutster', SHA1('password'), 'email@example.com'),
-> (7, 'Europe/Berlin', 'Ute', SHA1('pa24word'), 'email1@example.com'),
-> (4, 'Europe/Oslo', 'Silje', SHA1('2kll13'), 'email2@example.com'),
-> (2, 'America/Sao_Paulo', 'João', SHA1('fJDLN34'), 'email3@example.com'),
-> (1, 'Pacific/Auckland', 'kiwi', SHA1('conchord'), 'kiwi@example.org');
Query OK, 5 rows affected (0.00 sec)
Records: 5 Duplicates: 0 Warnings: 0
mysql>
```

图17-12 手动添加少量用户，因为这个站点中没有注册过程（但是请参见第18章了解该过程）

由于PHP脚本将显示与帖子关联的用户，所以需要几个用户。我为每个用户关联了语言和时区（参见第6章，了解MySQL中关于时区的更多信息）。每个用户的密码将使用SHA1()函数进行加密。

#### (10) 填充words表。

```
INSERT INTO words VALUES
(NULL, 1, 'PHP and MySQL for Dynamic Web Sites: The Forum!', '<p>Welcome to our site.... please use
the links above... blah, blah, blah.</p>\r\n<p>Welcome to our site....please use the links above...
blah, blah, blah.</p>', 'Home', 'Forum Home', 'Language', 'Register', 'Login', 'Logout', 'New Thread',
'Subject', 'Body', 'Submit', 'Posted on', 'Posted by', 'Replies', 'Latest Reply', 'Post a Reply');
```

这些是与英语中的每一项关联的单词。记录的lang\_id为1，它匹配languages表中英语的lang\_id。可以从本书的支持Web站点下载用于把其他语言的单词插入到这个表中的SQL代码。

#### ✓ 提示

- 本章没有介绍用于创建`mysqli_connect.php`页面的步骤，该页面用于连接到数据库。作为替代，只需从第9章中复制这样一个页面即可。然后，更改脚本中的参数，使用有效的用户名/密码/主机名组合来连接到`forum2`数据库。
- 提醒一下，一个表中的外键应该与另一个表中匹配的主键具有完全相同的类型和大小。

## 17.2 编写模板

这个示例（像包含许多页面的任何站点一样）将利用模板把大部分外观与逻辑隔开。遵循第3章中的指导，头文件和脚注文件将存储大多数HTML代码。然后，每个PHP脚本都将包括这些文件，用以建立一个完整的HTML页面（参见图17-13）。但是，这个示例更复杂一点。

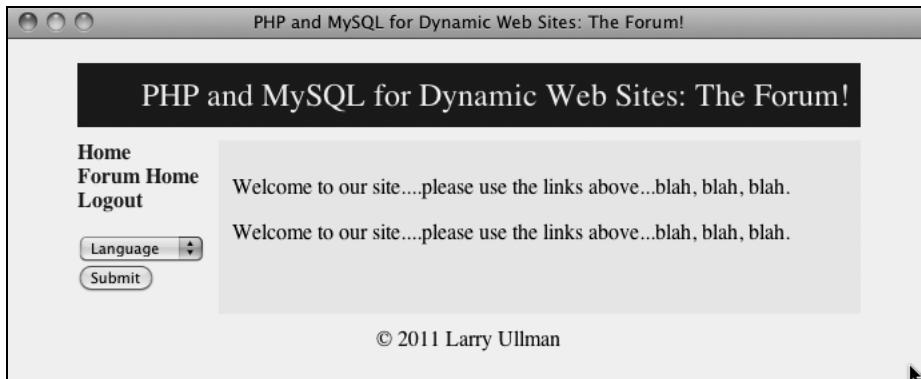


图17-13 站点的基本布局和外观

这个站点的目标之一是：以多种不同的语言为用户提供服务。要实现这个目标不仅仅涉及允许他们以母语发布消息，而且要确保他们可以用母语使用整个站点。这意味着需要以他们的语言显示页面标题、导航链接、大字标题、提示，甚至包括菜单（参见图17-14）。

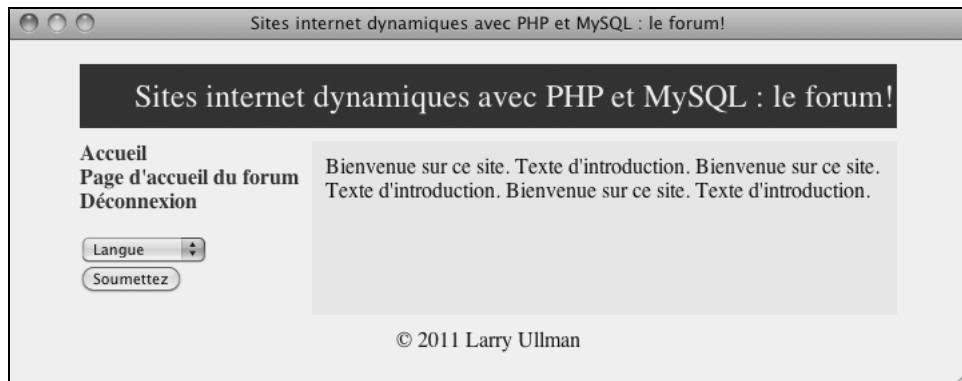


图17-14 用法语查看的主页（对比图17-13）

用于建立数据库的指导说明了如何完成这个任务：通过在表中存储所有关键词的译文。首先，头文件需要取出所有这些关键词，以便可以根据需要使用它们。其次，这个头文件还将基于用户登录与否显示不同的链接。只需添加另一种方法：如果用户位于论坛页面上（他们可以在该页面上以一种语言查看所有的论点），就可以让他们选择发布新的论点（参见图17-15）。



图17-15 这个主页与图17-13中所示的相同，但是它具有不同的链接，允许用户开始一个新的主题

模板本身使用CSS进行一些格式化工作（实际上并没有太多的格式化工作要做）。可以从本书的支持Web站点下载所有这些文件。

### 建立模板

(1) 在文本编辑器或IDE中开始创建一个新的文档，命名为header.html（参见脚本17-1）。

**脚本 17-1** header.html 文件开始创建模板。它还会设置页面的编码，开启会话，以及从数据库中检索特定语言的关键单词

```

1 <?php # Script 17.1 - header.html
2 /* This script...
3 * - starts the HTML template.
4 * - indicates the encoding using header().
5 * - starts the session.
6 * - gets the language-specific words from the database.
7 * - lists the available languages.
8 */
9
10 // Indicate the encoding:
11 header ('Content-Type: text/html; charset=UTF-8');
12
13 // Start the session:
14 session_start();
15
16 // For testing purposes:
17 $_SESSION['user_id'] = 1;
18 $_SESSION['user_tz'] = 'America/New_York';
19 // For logging out:
20 //$_SESSION = array();
21
22 // Need the database connection:
23 require ('../mysqli_connect.php');
24
25 // Check for a new language ID...
26 // Then store the language ID in the session:
27 if (isset($_GET['lid'])) &&
28 filter_var($_GET['lid'], FILTER_VALIDATE_INT, array('min_range' => 1))
29 {

```

```
30 $_SESSION['lid'] = $_GET['lid'];
31 } elseif (!isset($_SESSION['lid'])) {
32 $_SESSION['lid'] = 1; // Default.
33 }
34
35 // Get the words for this language:
36 $q = "SELECT * FROM words WHERE lang_id = {$_SESSION['lid']}";
37 $r = mysqli_query($dbc, $q);
38 if (mysqli_num_rows($r) == 0) { // Invalid language ID!
39
40 // Use the default language:
41 $_SESSION['lid'] = 1; // Default.
42 $q = "SELECT * FROM words WHERE lang_id = {$_SESSION['lid']}";
43 $r = mysqli_query($dbc, $q);
44
45 }
46
47 // Fetch the results into a variable:
48 $words = mysqli_fetch_array($r, MYSQLI_ASSOC);
49
50 // Free the results:
51 mysqli_free_result($r);
52 ?>
53 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
54 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
55 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
56 <head>
57 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
58 <title><?php echo $words['title']; ?> </title>
59 <style type="text/css" media="screen">
60 body { background-color: #ffffff; }
61
62 .content {
63 background-color: #f5f5f5;
64 padding-top: 10px; padding-right: 10px; padding-bottom: 10px; padding-left: 10px;
65 margin-top: 10px; margin-right: 10px; margin-bottom: 10px; margin-left: 10px;
66 }
67
68 a.navlink:link { color: #003366; text-decoration: none; }
69 a.navlink:visited { color: #003366; text-decoration: none; }
70 a.navlink:hover { color: #cccccc; text-decoration: none; }
71
72 .title {
73 font-size: 24px; font-weight: normal; color: #ffffff;
74 margin-top: 5px; margin-bottom: 5px; margin-left: 20px;
75 padding-top: 5px; padding-bottom: 5px; padding-left: 20px;
76 }
77 </style>
78 </head>
79 <body>
80
81 <table width="90%" border="0" cellspacing="10" cellpadding="0" align="center">
82
83 <tr>
```

```

84 <td colspan="2" bgcolor="#003366" align="center"><p class="title"><?php echo
85 $words['title']; ?></p></td>
86 </tr>
87 <tr>
88 <td valign="top" nowrap="nowrap" width="10%">
89 <?php // Display links:
90
91 // Default links:
92 echo '' . $words['home'] . '

93 ' . $words['forum_home'] . '
';
94
95 // Display links based upon login status:
96 if (isset($_SESSION['user_id'])) {
97
98 // If this is the forum page, add a link for posting new threads:
99 if (basename($_SERVER['PHP_SELF']) == 'forum.php') {
100 echo '' . $words['new_thread'] . '
';
101 }
102
103 // Add the logout link:
104 echo '' . $words['logout'] . '
';
105
106 } else {
107
108 // Register and login links:
109 echo '' . $words['register'] . '

110 ' . $words['login'] . '
';
111
112 }
113
114 // For choosing a forum/language:
115 echo '<p><form action="forum.php" method="get">
116 <select name="lid">
117 <option value="0">' . $words['language'] . '</option>
118 ';
119
120 // Retrieve all the languages...
121 $q = "SELECT lang_id, lang FROM languages ORDER BY lang_eng ASC";
122 $r = mysqli_query($dbc, $q);
123 if (mysqli_num_rows($r) > 0) {
124 while ($menu_row = mysqli_fetch_array($r, MYSQLI_NUM)) {
125 echo "<option value=\"$menu_row[0]\">$menu_row[1]</option>\n";
126 }
127 }
128 mysqli_free_result($r);
129
130 echo '</select>

131 <input name="submit" type="submit" value=' . $words['submit'] . '" />
132 </form></p>
133 </td>
134 <td valign="top" class="content">';
135 ?>
```

由于这个脚本将需要执行大量的数据验证和检索工作，它将开始于一个PHP块。脚本还使用header()函数向Web浏览器指明其编码——UTF-8。参见第11章，了解关于该函数的更多信息。

(2) 开启会话。

```
$_SESSION['user_id'] = 1;
$_SESSION['user_tz'] = 'America/New_York';
// $_SESSION = array();
```

为了在用户登录后跟踪他们，站点将使用会话。由于本章中的站点没有注册和登录功能，这两行代码实际上可以登录用户。通常，这两个值来自于数据库，但是为了测试，将在这里设置它们。为了实际地注销用户，可以取消注释第三行代码。

(3) 包括数据库连接。

```
require ('../mysqli_connect.php');
```

与本书中的许多其他示例一样，假定mysqli\_connect.php脚本存储在当前目录的上一级目录中，它在Web根目录外部。如果你的情况与此不同，相应更改代码。

(4) 确定语言ID。

```
if (!isset($_GET['lid'])) &&
 filter_var($_GET['lid'], FILTER_VALIDATE_INT, array('min_range' => 1))
{
 $_SESSION['lid'] = $_GET['lid'];
} elseif (!isset($_SESSION['lid'])) {
 $_SESSION['lid'] = 1; // Default.
}
```

接下来，如果在URL中接收到一个语言ID值（简写为lid），则将验证它。语言ID控制使用什么语言来显示所有的站点元素，它还规定了要查看的论坛。语言ID可以在会话中找到，在通过成功登录获取了相关的信息后（因为用户的语言ID被存储在users表）。另外，用户可以通过在导航链接中提交语言表单来快速更改显示的语言（参见图17-13）。在那种情况下，提交的语言ID需要被验证为一个大于1的整数：可以使用过滤器扩展轻松实现（参见第13章）。如果你使用的PHP版本不支持过滤器扩展，就需要使用类型转换来代替（参见第13章）。

如果页面没有在URL中接收到语言ID并且尚未在会话中建立语言ID，就会应用第二个子句。在这种情况下，将选择默认的语言。这个值对应于数据库中languages表中的英语。可以把它改成任何ID，以匹配你想使用的默认语言。

(5) 获取该语言的关键单词。

```
$q = "SELECT * FROM words WHERE lang_id = {$_SESSION['lid']}";
$r = mysqli_query($dbc, $q);
```

头文件中的下一步是从数据库中检索给定语言的所有关键词。将使用一个变量作为标志，指示这个过程是否执行成功。然后检查确认语言ID为正数，并且在数据库上运行查询。如果查询返回一条记录，将把这些值存入\$words中。

(6) 如果查询没有返回记录，就会得到默认的单词。

```
if (mysqli_num_rows($r) == 0) {
 $_SESSION['lid'] = 1;
 $q = "SELECT * FROM words WHERE lang_id = {$_SESSION['lid']}";
```

```

 $r = mysqli_query($dbc, $q);
}

```

这是有可能的，但可能性不大，\$\_SESSION['lid']不等于words表中的一条记录。在这种情况下查询将会返回空记录（但是运行没有错误）。因此，现在需要获取默认的语言单词。注意，无论是本部分代码块还是步骤(5)的代码块，实际上都没有获取返回的记录。两个可能的查询都将在步骤(7)中发生。

(7) 将检索到的单词获取到一个数组中，释放资源，闭合PHP部分。

```

$words = mysqli_fetch_array($r, MYSQLI_ASSOC);
mysqli_free_result($r);
?>

```

在这之后，\$words数组代表了用户选择的语言的所有导航和通用元素（或默认语言）。

不必调用mysqli\_free\_result()，但是这样做可以使程序设计条理清晰。

(8) 开始创建HTML页面。

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title><?php echo $words['title']; ?></title>

```

注意：即使PHP的header()调用已经确定了编码，在META标签中也要指明编码。这只是为了把事情做得认真周到。

头文件使用\$words数组中的值作为每一个页的标题（即，页面的标题将总是与每个选择了语言的页面相同）。你可以修改代码，使页面的标题是一个语言单词和一个页面特定变量的组合，就像第3章和后面的例子中用到的\$page\_title变量一样。

(9) 添加CSS。

```

<style type="text/css">
media="screen">
body { background-color: #ffffff; }
.content {
background-color: #f5f5f5;
padding-top: 10px; padding-right: 10px; padding-bottom: 10px; padding-left: 10px;
margin-top: 10px; margin-right: 10px; margin-bottom: 10px; margin-left: 10px;
}
a.navlink:link { color: #003366; text-decoration: none; }
a.navlink:visited { color: #003366; text-decoration: none; }
a.navlink:hover { color: #cccccc; text-decoration: none; }
.title {
font-size: 24px; font-weight: normal; color: #ffffff;
margin-top: 5px; margin-bottom: 5px; margin-left: 20px;
padding-top: 5px; padding-bottom: 5px; padding-left: 20px;
}
</style>

```

这全部取自于我以前在某个时间某个地方发现的一个模板。它给站点添加了一点装饰。

(10) 完成HTML头部并开始创建页面。

```

</head>
<body>
<table width="90%" border="0" cellspacing="10" cellpadding="0" align="center">
 <tr>
 <td colspan="2" bgcolor="#003366" align="center"><p class="title"><?php echo $words['title']; ?></p></td>
 </tr>
 <tr>
 <td valign="top" nowrap="nowrap" width="10%">

```

页面本身将使用表格进行布局，该表格中用一行显示页面标题，下一行在左边包含导航链接并在右边包含特定页面的内容，最后一行包含版权信息（参见图17-16）。在这段代码中你将看到页面标题也是特定语言的。



图17-16 页面布局显示了主HTML表格的行和列

(11) 开始显示链接。

```

<?php
echo '' . $words['home'] . '

' . $words['forum_home'] . '
';

```

不管用户登录与否，也不管他们当前查看的是什么页面，总会显示前两个链接。对于每个链接，链接自身的文本将是特定语言的。

(12) 如果用户登录，就显示“new thread”和注销链接。

```

if (isset($_SESSION['user_id'])) {
 if (basename($_SERVER['PHP_SELF']) == 'forum.php') {
 echo '' . $words['new_thread'] . '
';
 }
 echo '' . $words['logout'] . '
';
}

```

通过检查\$\_SESSION['user\_id']变量是否存在来确认用户的登录状态。如果设置了该变量，那么就可以创建注销链接。在这之前，需要检查这是不是forum.php页面。如果是，那么就创建一个链接，用于开始一个新论点（仅当用户位于论坛页面上时，他们才可以创建新论点；你不希望他们在其他一些页面（比如主页）上创建新论点，因为这样将不清楚应该把论点发布到哪个论坛上）。用于检查是哪个页面的函数basename()，是在第12章介绍的。

(13) 为没有登录的用户显示链接。

```
} else {
 echo '' . $words['register'] . '

 ' . $words['login'] . '
';
}
```

如果用户没有登录，就提供用于注册和登录的链接。

(14) 开始创建一个用于选择语言的表单。

```
echo '<p><form action="forum.php" method="get">
<select name="lid">
<option value="0">' . $words ['language'] . '</option>
';
```

用户可以使用下拉菜单选择一种语言（它也是一个论坛）（参见图17-17）。在用户的默认语言中，该菜单中的第一个值将是单词“language”。选项菜单的名称是lid，它是language ID（语言ID）的简写，并且它的动作指向forum.php。因此，当用户提交这个简单的表单时，将把他们带到所选的论坛。



图17-17 语言下拉菜单，它带有其母语中的每个选项

(15) 检索每一种语言，并把它们加入菜单中。

```
$q = "SELECT lang_id, lang FROM languages ORDER BY lang_eng ASC";
$r = mysqli_query($dbc, $q);
if (mysqli_num_rows($r) > 0) {
 while ($menu_row = mysqli_fetch_array($r, MYSQLI_NUM)) {
 echo "<option value=\"$menu_row[0]\">$menu_row[1] </option>\n";
 }
}
mysqli_free_result($r);
```

这个查询从languages表中检索语言和语言ID，并把它们全都添加为选项菜单中的选项。

同样，这几行代码也不是必需的，但是它们有助于限制错误。特别是，当你的页面运行多个SELECT查询时，`mysqli_free_result()`有助于避免PHP与MySQL之间的混淆问题。

(16) 完成表单和PHP页面。

```

echo '</select>

<input name="submit" type="submit" value="' . $words['submit'] . '" />
</form></p>
</td>
<td valign="top" class="content">';
?>

```

(17) 将文件另存为header.html。

即使这个脚本包含大量的PHP代码，它仍将使用.html扩展名（这是我首选用于模板文件的扩展名）。确保使用UTF-8编码保存该文件。

(18) 在文本编辑器或IDE中创建一个新文档，命名为footer.html（参见脚本17-2）。

#### 脚本 17-2 脚注文件完成 HTML 页面

```

1 <!-- Script 17.2 - footer.html -->
2 </td>
3 </tr>
4
5 <tr>
6 <td colspan="2" align="center">© 2011 Larry Ullman</td>
7 </tr>
8
9 </table>
10 </body>
11 </html>

```

(19) 完成HTML页面。

```

</td>
</tr>
<tr>
<td colspan="2"
align="center">© 2011 Larry Ullman</td>
</tr>
</table>
</body>
</html>

```

(20) 将文件另存为footer.html。

同样，确保使用UTF-8编码保存该文件（参见第14章）。

(21) 将这两个文件存放在Web目录中includes文件夹内。

### 17.3 创建索引页面

本示例中的索引页面将不会做太多工作。它将提供一些介绍性文本和链接，允许用户注册、登录、选择他们的语言/论坛，等等。从编程的角度讲，它将说明如何使用模板文件。

#### 建立主页

(1) 在文本编辑器或IDE中开始创建一个新的PHP文档，命名为index.php（参见脚本17-3）。

**脚本 17-3** 主页包括头文件和脚注文件，以建立一个完整的 HTML 文档。它还会以所选的语言打印一些介绍性文本

```

1 <?php # Script 17.3 - index.php
2 // This is the main page for the site.
3
4 // Include the HTML header:
5 include ('includes/header.html');
6
7 // The content on this page is introductory text
8 // pulled from the database, based upon the
9 // selected language:
10 echo $words['intro'];
11
12 // Include the HTML footer file:
13 include ('includes/footer.html');
14 ?>

```

由于所有的HTML代码都位于包含文件中，这个页面可以开始于PHP开始标签。

(2) 包括HTML头部。

```
include ('includes/header.html');
```

该包含文件使用header()和session\_start()函数，因此必须确保在执行这一行代码之前不会把任何内容发送到Web浏览器。只要PHP开始标签之前没有空格，这就应该不是一个大问题。

(3) 打印特定语言的内容。

```
echo $words['intro'];
```

在头文件内定义\$words数组。可以在这里引用它，因为刚才包括了头文件。在intro中索引的值是一些用所选的或默认的语言书写的欢迎文本。

(4) 完成页面。

```

include ('includes/footer.html');
?>
```

这用于完成主页！

(5) 将文件另存为index.php，存放在Web目录中，并在Web浏览器中测试它（参见图17-13和图17-14）。

重申一遍，一定要使用UTF-8编码保存文件。这将是我最后一次提醒你！

## 17.4 创建论坛页面

Web站点中的下一个页面是论坛页面，它显示论坛中的论点（每种语言都是它自己的论坛）。该页面将使用语言ID（它在一个URL中传递给该页面和/或存储在一个会话中）获悉要显示什么论点。

这个页面的基本功能（运行查询、显示结果）很简单（参见图17-18）。这个页面使用的查询也许是本书中最复杂的查询，这有以下三个原因。

(1) 它执行跨三个表的联结。

(2) 它将使用三个聚集函数和一个GROUP BY子句。

(3) 它把日期转换成用户的时区，但是仅当查看页面的用户已登录时它才会这样做。

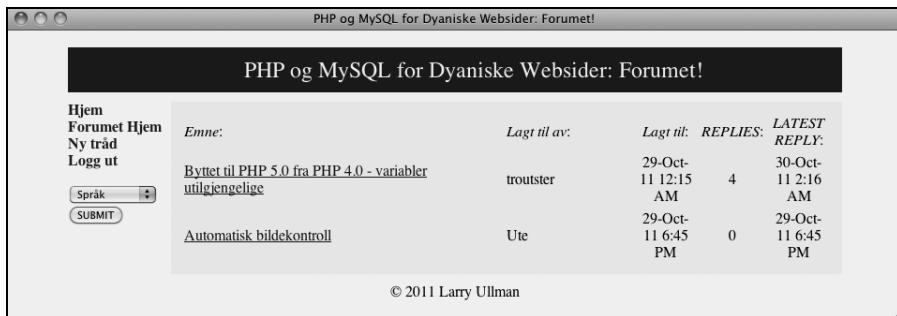


图17-18 论坛页面，它以给定的语言列出了关于论点的信息。这些论点将链接到一个页面，可以在那里阅读它们

因此，这个查询非常复杂，我将在下面的步骤中详细介绍它。

#### 编写论坛页面

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为`forum.php`（参见脚本17-4）。

**脚本 17-4** 这个脚本执行一个相当复杂的查询，用于为论坛中的每个论点显示 5 种信息——主题、原始发布者、开始论点的日期、回复数和最近回复的日期

```

1 <?php # Script 17.4 - forum.php
2 // This page shows the threads in a forum.
3 include ('includes/header.html');
4
5 // Retrieve all the messages in this forum...
6
7 // If the user is logged in and has chosen a time zone,
8 // use that to convert the dates and times:
9 if (isset($_SESSION['user_tz'])) {
10 $first = "CONVERT_TZ(p.posted_on, 'UTC', '{$_SESSION['user_tz']}')";
11 $last = "CONVERT_TZ(p.posted_on, 'UTC', '{$_SESSION['user_tz']}')";
12 } else {
13 $first = 'p.posted_on';
14 $last = 'p.posted_on';
15 }
16
17 // The query for retrieving all the threads in this forum, along with the original user,
18 // when the thread was first posted, when it was last replied to, and how many replies it's had:
19 $q = "SELECT t.thread_id, t.subject, username, COUNT(post_id) - 1 AS responses, MAX(DATE_FORMAT
($last, '%e-%b-%Y %l:%i %p')) AS last, MIN(DATE_FORMAT($first, '%e-%b-%Y %l:%i %p')) AS first
FROM threads AS t INNER JOIN posts AS p USING (thread_id) INNER JOIN users AS u ON t.user_id = u.user_id
WHERE t.lang_id = {$_SESSION['lid']} GROUP BY (p.thread_id) ORDER BY last DESC";
20 $r = mysqli_query($dbc, $q);
21 if (mysqli_num_rows($r) > 0) {
22
23 // Create a table:
24 echo '<table width="100%" border="0" cellspacing="2" cellpadding="2" align="center">

```

```

25 <tr>
26 <td align="left" width="50%">' . $words['subject'] . ':</td>
27 <td align="left" width="20%">' . $words['posted_by'] . ':</td>
28 <td align="center" width="10%">' . $words['posted_on'] . ':</td>
29 <td align="center" width="10%">' . $words['replies'] . ':</td>
30 <td align="center" width="10%">' . $words['latest_reply'] . ':</td>
31 </tr>';
32
33 // Fetch each thread:
34 while ($row = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
35
36 echo '<tr>
37 <td align="left">' . $row['subject'] .
38 '</td>
39 <td align="left">' . $row['username'] . '</td>
40 <td align="center">' . $row['first'] . '</td>
41 <td align="center">' . $row['responses'] . '</td>
42 <td align="center">' . $row['last'] . '</td>
43 </tr>';
44 }
45
46 echo '</table>'; // Complete the table.
47
48 } else {
49 echo '<p>There are currently no messages in this forum.</p>';
50 }
51
52 // Include the HTML footer file:
53 include ('includes/footer.html');
54 ?>
```

## (2) 确定要使用的日期和时间

```

if (isset($_SESSION['user_tz'])) {
 $first = "CONVERT_TZ (p.posted_on, 'UTC', '{$_SESSION['user_tz']}')";
 $last = "CONVERT_TZ(p.posted_on, 'UTC', '{$_SESSION['user_tz']}')";
} else {
 $first = 'p.posted_on';
 $last = 'p.posted_on';
}
```

我已经说过，查询将把日期和时间格式化为用户的时区（可能是在注册过程中选择的时区），但是，仅当查看者登录时才这样做。将在登录时从数据库中检索该信息，并将其存储在会话中。

为了使查询是动态的，将把应该选择的准确日期/时间值存储在一个变量中。如果用户没有登录，这意味着没有设置\$\_SESSION['user\_tz']，两个日期（开始论点的时间和发布最近回复的时间）将是来自表中的精确值。在两种情况下，要引用的表列是posts表中的posted\_on（p将是查询中posts的别名）。

如果用户登录，将使用CONVERT\_TZ()函数把posted\_on中存储的值从UTC转换成离用户最近的时区。参见第6章，了解关于这个函数的更多信息。注意，使用此函数需要MySQL安装时区列表（详见第6章）。

## (3) 定义并执行查询。

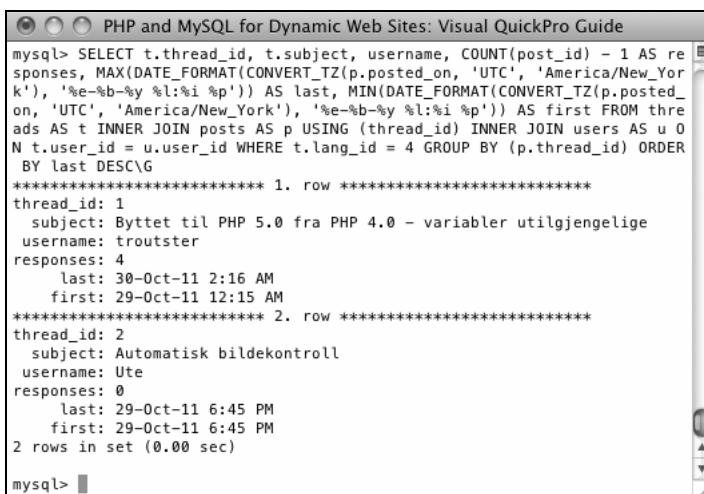
```
$q = "SELECT t.thread_id, t.subject, username, COUNT(post_id) - 1 AS responses, MAX(DATE_FORMAT($last, '%e-%b-%y %l:%i %p')) AS last, MIN(DATE_FORMAT($first, '%e-%b-%y %l:%i %p')) AS first FROM threads AS t INNER JOIN posts AS p USING(thread_id) INNER JOIN users AS u ON t.user_id = u.user_id WHERE t.lang_id = ${_SESSION['lid']} GROUP BY (p.thread_id) ORDER BY last DESC";
$r = mysqli_query($dbc, $q);
if (mysqli_num_rows($r) > 0) {
```

该查询需要返回6项内容：每个论点的ID和主题（来自于threads表）、首先发布论点的用户名（来自users表）、每个论点的回复数、开始论点的日期、最近回复论点的日期（都来自posts表）。

这个查询的结构是使用thread\_id列在threads表和posts表之间执行的一个联结（thread\_id列在这两个表中是相同的）。然后使用user\_id列将这个结果与users表进行联结。

至于所选的值，将使用三个聚集函数（参见第7章）：COUNT()、MIN()和MAX()。每个聚集函数都应用于posts表中的列，因此查询具有一个GROUP BY (p.thread\_id)子句。使用MIN()和MAX()返回最早（用于原始帖子）和最近的日期。它们都将显示在论坛页面上（参见图17-18）。最近的日期还用于对结果进行排序，使得总是首先返回最近的活动。COUNT()函数用于统计给定论点中的帖子数。由于原始帖子也在posts表中，所以也会统计它，因此要从统计中减去1。

最后，使用别名使得查询编写起来更短一些，并且使得更容易在PHP脚本中使用结果。图17-19显示了在MySQL客户端或phpMyAdmin中执行的这个查询，它针对的是用户未登录的情况。



```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT t.thread_id, t.subject, username, COUNT(post_id) - 1 AS responses, MAX(DATE_FORMAT(CONVERT_TZ(p.posted_on, 'UTC', 'America/New_York'), '%e-%b-%y %l:%i %p')) AS last, MIN(DATE_FORMAT(CONVERT_TZ(p.posted_on, 'UTC', 'America/New_York'), '%e-%b-%y %l:%i %p')) AS first FROM threads AS t INNER JOIN posts AS p USING(thread_id) INNER JOIN users AS u ON t.user_id = u.user_id WHERE t.lang_id = 4 GROUP BY (p.thread_id) ORDER BY last DESC

1. row *****
thread_id: 1
subject: Byttet til PHP 5.0 fra PHP 4.0 - variabler utilgjengelige
username: troutster
responses: 4
last: 30-Oct-11 2:16 AM
first: 29-Oct-11 12:15 AM

2. row *****
thread_id: 2
subject: Automatisk bildekontroll
username: Ute
responses: 0
last: 29-Oct-11 6:45 PM
first: 29-Oct-11 6:45 PM
2 rows in set (0.00 sec)

mysql>
```

图17-19 在MySQL客户端中运行这个复杂查询的结果

(4) 为结果创建一个表。

```
echo '<table width="100%" border="0" cellspacing="2" cellpadding="2" align="center">
<tr>
<td align="left" width="50%>' . $words['subject'] . '</td>
<td align="left" width="20%">' . $words ['posted_by'] . '</td>
<td align="center" width="10%">' . $words['posted_on'] . '</td>
<td align="center" width="10%">' . $words['replies']. '</td>
<td align="center" width="10%">' . $words['latest_reply'] . '</td>
</tr>';
```

与头文件中的一些项目一样，这个HTML页面中的列标题将使用特定语言的术语。

(5) 获取并打印每个返回的记录。

```
while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
 echo '<tr>
 <td align="left">' . $row ['subject'] . '</td>
 <td align="left">' . $row ['username'] . '</td>
 <td align="center">' . $row ['first'] . '</td>
 <td align="center">' . $row ['responses'] . '</td>
 <td align="center">' . $row ['last'] . '</td>
 </tr>';
}
```

这段代码相当简单，并且在本书中多次出现类似的示例。将论点的主题链接到read.php，在URL中把论点ID传递给该页面。

(6) 完成页面。

```
echo '</table>';
} else {
 echo '<p>There are currently no messages in this forum.</p>';
}
include ('includes/footer.html');
?>
```

如果查询没有返回结果，就会应用这个else子句。实际上，还应该以用户所选的语言显示这条消息。出于简洁考虑，我省略了它。为了完全实现这种特性，可在words表中创建另一个列，并为每种语言存储这段文本的翻译版本。

(7) 将文件另存为forum.php，存放在Web目录中，并在Web浏览器中测试它（参见图17-20）。

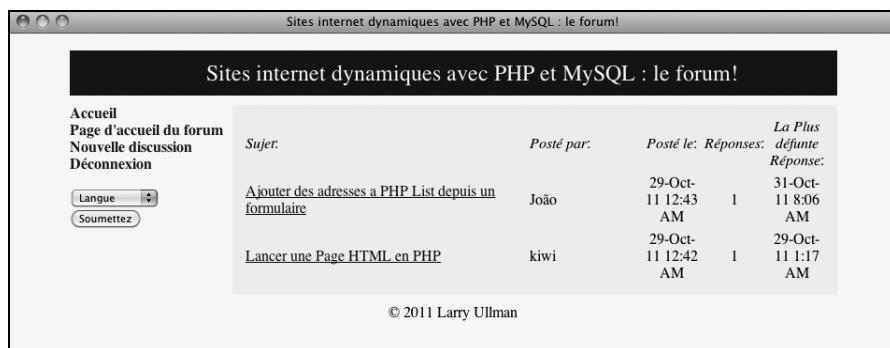


图17-20 用另一种语言查看的forum.php页面（对比图17-18）

**✓ 提示**

- 如果你在运行这段脚本时看到日期和时间没有值，有可能是因为你的MySQL还没有更新全部的时区列表。
- 如本章简介中所指明的，我在这个示例中省略了所有的错误处理。如果你对于这些查询有任何问题，可以应用第7章中概括的调试技术。

## 17.5 创建论点页面

接下来创建一个页面，用于查看论点中的所有消息（参见图17-21）。通过单击forum.php中的一个链接访问这个页面（参见图17-22）。由于简化的数据库结构，这个脚本使用的查询不是那么复杂（如果利用第6章中介绍的数据库设计，这个页面将复杂得多）。这样，这个页面只需确保它接收到一个有效的论点ID，显示所有的消息，并显示一个表单让用户添加他们自己的回复。

图17-21 read.php页面显示论点中的所有消息

```
<td align="left">Byttet til PHP 5.0
<td align="left">troutster</td>
<td align="center">29-Oct-11 12:15 AM</td>
<td align="center">4</td>
<td align="center">30-Oct-11 2:16 AM</td>
>
<td align="left">Automatisk bildeko
<td align="left">Ute</td>
<td align="center">29-Oct-11 6:45 PM</td>
<td align="center">0</td>
<td align="center">29-Oct-11 6:45 PM</td>
```

图17-22 forum.php的一部分源代码显示了如何在URL中把论点ID传递给read.php

### 建立read.php页面

- (1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为read.php（参见脚本17-5）。

**脚本 17-5** read.php 页面以发布日期升序显示论点中的所有消息。该页面还会在顶部显示论点的主题，并且在底部包括一个表单用于添加回复

```

1 <?php # Script 17.5 - read.php
2 // This page shows the messages in a thread.
3 include ('includes/header.html');
4
5 // Check for a thread ID...
6 $tid = FALSE;
7 if (isset($_GET['tid']) && filter_var($_GET['tid'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
8
9 // Create a shorthand version of the thread ID:
10 $tid = $_GET['tid'];
11
12 // Convert the date if the user is logged in:
13 if (isset($_SESSION['user_tz'])) {
14 $posted = "CONVERT_TZ(p.posted_on, 'UTC', '{$_SESSION['user_tz']}')";
15 } else {
16 $posted = 'p.posted_on';
17 }
18
19 // Run the query:
20 $q = "SELECT t.subject, p.message, username, DATE_FORMAT($posted, '%e-%b-%y %l:%i %p') AS posted
FROM threads AS t LEFT JOIN posts AS p USING (thread_id) INNER JOIN users AS u ON p.user_id =
u.user_id WHERE t.thread_id = $tid ORDER BY p.posted_on ASC";
21 $r = mysqli_query($dbc, $q);
22 if (!(mysqli_num_rows($r) > 0)) {
23 $tid = FALSE; // Invalid thread ID!
24 }
25
26 } // End of isset($_GET['tid']) IF.
27
28 if ($tid) { // Get the messages in this thread...
29
30 $printed = FALSE; // Flag variable.
31
32 // Fetch each:
33 while ($messages = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
34
35 // Only need to print the subject once!
36 if (!$printed) {
37 echo "<h2>{$messages['subject']}

```

```

48 } else { // Invalid thread ID!
49 echo '<p>This page has been accessed in error.</p>';
50 }
51 }
52
53 include ('includes/footer.html');
54 ?>

```

(2) 开始验证论点ID。

```

$tid = FALSE;
if (isset($_GET['tid']) && filter_var($_GET['tid'], FILTER_VALIDATE_INT, array('min_range' => 1)))
{
 $tid = $_GET['tid'];
}

```

首先，将标志变量定义为FALSE，这相当于指出：证明论点ID是有效的，这是此脚本最重要的部分。接下来，检查确认在URL中传递论点ID，并且它是一个数字。这是通过过滤器扩展实现的（详见第13章）。最后，将传递到页面的值赋给\$tid变量，使其不再含有FALSE值。

如果你的PHP版本不支持过滤器扩展，那就需要将\$\_GET['tid']类型转换为一个整数，然后确认它有一个大于1的值（参见第13章）。

(3) 确定是否应该调整日期和时间。

```

if (isset($_SESSION['user_tz'])) {
 $posted = "CONVERT_TZ (p.posted_on, 'UTC', '{$_SESSION['user_tz']}')";
} else {
 $posted = 'p.posted_on';
}

```

与forum.php页面（参见脚本17-4）中一样，如果用户登录，查询将格式化用户时区中的所有日期和时间。为了能够相应地调整查询，这个变量将存储列的名称或者MySQL的CONVERT\_TZ()函数的调用。

(4) 运行查询。

```

$q = "SELECT t.subject, p.message, username, DATE_FORMAT($posted, '%e-%b-%y %l:%i %p') AS posted FROM
threads AS t LEFT JOIN posts AS p USING (thread_id) INNER JOIN users AS u ON p.user_id = u.user_id
WHERE t.thread_id = $tid ORDER BY p.posted_on ASC";
$r = mysqli_query($dbc, $q);
if (!mysqli_num_rows($r) > 0) {
 $tid = FALSE;
}

```

这个查询类似于论坛页面上的那个查询，但是在两个方面简化了它。第一，它没有使用任何聚集函数或GROUP BY子句。第二，它只返回一个日期/时间。该查询仍然是跨三个表的联结，以便获得主题、消息正文和用户名。按它们的发布日期以升序进行排序（即从第一个帖子到最近的帖子）。

如果查询没有返回任何行，那么论点ID就是无效的，并再次赋予标志变量FALSE。

(5) 完成\$\_GET['tid']条件语句，并再次检查论点ID是否有效。

```

} // End of isset($_GET['tid']) IF.
if ($tid) {

```

在打印论点中的消息之前，将使用最后一个条件语句。如果提供大于0的数字型论点ID，但是它没有返回数据库中的任何行，这个条件语句将求值为假。

- 没有\$\_GET['tid']被传递给这个页面。
- 一个\$\_GET['tid']值被传递给这个页面，但是它不是一个大于0的整数。
- 一个\$\_GET['tid']值被传递给这个页面并且他是一个大于0的整数，但是它不匹配数据库中的任何讨论记录。

(6) 打印每条消息。

```
$printed = FALSE;
while ($messages = mysqli_fetch_array($r, MYSQLI_ASSOC)) {
 if (!$printed) {
 echo
"<h2>{$messages['subject']}</h2>\n";
 $printed = TRUE;
 }
 echo "<p>{$messages['username']} ({$messages['posted']})
{$messages['message']}
\n";
} // End of WHILE loop.
```

如图17-21所示，论点主题只需要打印一次。不过，查询将返回每个返回的消息的主题（参见图17-23）。为了实现这种效果，将创建一个标志变量。如果\$printed为假，那么就需要打印主题。对于从数据库中获取的第一行将会是这样。一旦显示了主题，就把\$printed设置为TRUE，使得不会再次打印主题。然后显示用户名、发布日期和消息。

Subject	Message	Username	Posted
Byttet til PHP 5.0 fra PHP 4.0 – variabler utilgjengelige	Jeg har netttopp gått over til PHP 5.0 og forsøkte å benytte meg av mine gamle scripts. Dette viste seg å være noe vanskelig ettersom de bare generer feil. Hovedproblemet virker å være at jeg ikke får tilgang til variabler som tidligere var tilgjengelige. Noen som har noen forslag?	Silje	29-Oct-11 12:15 AM
Byttet til PHP 5.0 fra PHP 4.0 – variabler utilgjengelige	Har du sjekket om variablene du prøver å få tilgang på er superglobals? Dette forandret seg fra 4.2 og utover, tror jeg...	troutster	29-Oct-11 12:20 AM
Byttet til PHP 5.0 fra PHP 4.0 – variabler utilgjengelige	Hva er superglobals?	João	29-Oct-11 12:30 AM
Byttet til PHP 5.0 fra PHP 4.0 – variabler utilgjengelige	Linken Terje ga er manualsiden, men du kan også ta en titt på http://www.linuxjournal.com/article/6559 for en grundig innføring og forklaring. Lykke til!	kiwi	29-Oct-11 6:26 AM
Byttet til PHP 5.0 fra PHP 4.0 – variabler utilgjengelige	http://no.php.net/variables.predefined	troutster	30-Oct-11 2:16 AM

5 rows in set (0.00 sec)

图17-23 在MySQL客户端中运行read.php查询的结果。查询的这个版本把日期转换成已登录用户的首选时区

(7) 包括一个表单用于发布消息。

```
include ('includes/post_form.php');
```

因为用户可以使用两种方式（作为对现有论点的回复和作为新论点中的第一个帖子）发布消息，

所以我把表单本身存放在单独一个文件中（接下来将创建该文件）。

(8) 完成页面。

```

} else { // Invalid thread ID!
 echo '<p>This page has been accessed in error.</p>';
}
include ('includes/footer.html');
?>

```

同样，在一个完整的站点中，这条出错消息也会存储在每种语言中的words表中。然后，你将在这里编写如下代码：

```
echo "<p>{$words['access_error']}</p>";
```

(9) 将文件另存为read.php，存放在Web目录中，并在Web浏览器中测试它（参见图17-24）。



图17-24 用日语查看的read.php页面

## 17.6 发布消息

这个应用程序中的最后两个页面最重要，因为如果没有它们，用户将不能阅读论点。我将创建两个文件用于发布消息。一个文件将建立表单，另一个文件将处理表单。

### 17.6.1 创建表单

发布消息所需的第一个页面是post\_form.php，它具有一些偶然性。

- (1) 它只能被其他文件包含，永远不能直接访问它。
- (2) 仅当用户登录后，才应该显示它（这就是说，只有登录用户可以发布消息）。
- (3) 如果把它用于向现有消息添加回复，那么它只需一个消息正文输入框（参见图17-25）。

The screenshot shows a web browser window titled "PHP and MySQL for Dynamic Web Sites: The Forum!". The main content area is titled "Sample Thread". It displays two posts: one from "troutster" at 29-Oct-11 1:12 AM stating "This is the body of the sample thread. This is the body of the sample thread. This is the body of the sample thread.", and another from "troutster" at 29-Oct-11 1:44 AM stating "I like your thread. It's simple and sweet.". Below the posts is a form titled "Post a Reply" with a large text input field labeled "Body:" and a "Submit" button. On the left side of the page, there is a sidebar with links: "Home", "Forum Home", and "Logout". There is also a "Language" dropdown and a "Submit" button.

图17-25 用于发布消息的表单，如论点查看页面上所示

- (4) 如果把它用于创建新论点，它将需要主题和正文两个输入框（参见图17-26）。

The screenshot shows a web browser window titled "PHP and MySQL for Dynamic Web Sites: The Forum!". The main content area is titled "New Thread". It has a "Subject:" input field containing "[REDACTED]" and a large "Body:" text input field. Below these fields is a "Submit" button. On the left side, there is a sidebar with links: "Home", "Forum Home", and "Logout". There is also a "Language" dropdown and a "Submit" button. A small "17" is visible in the bottom right corner of the image.

图17-26 用于发布消息的相同表单，它用于创建新论点

(5) 它需要是黏性的 (参见图17-27)。

图17-27 如果没有正确地完成这个表单，它将恢复输入过的值

尽管如此，只需通过大约60行代码和一些聪明的条件语句即可完成所有这些任务。

#### 创建post\_form.php

(1) 在文本编辑器或IDE中开始创建一个新的PHP文档，命名为post\_form.php (参见脚本17-6)。

**脚本 17-6** 这个脚本将被其他页面 (特别是 read.php 和 post.php ) 包含。它会显示一个用于发布消息的表单，该表单也是黏性的

```

1 <?php # Script 17.6 - post_form.php
2 // This page shows the form for posting messages.
3 // It's included by other pages, never called directly.
4
5 // Redirect if this page is called directly:
6 if (!isset($words)) {
7 header ("Location: http://www.example.com/index.php");
8 exit();
9 }
10
11 // Only display this form if the user is logged in:
12 if (isset($_SESSION['user_id'])) {
13
14 // Display the form:
15 echo '<form action="post.php" method="post" accept-charset="utf-8">';
16
17 // If on read.php...
18 if (isset($tid) && $tid) {
19
20 // Print a caption:
21 echo '<h3>' . $words['post_a_reply'] . '</h3>';
22

```

```

23 // Add the thread ID as a hidden input:
24 echo '<input name="tid" type="hidden" value="' . $tid . '" />';
25
26 } else { // New thread
27
28 // Print a caption:
29 echo '<h3>' . $words['new_thread'] . '</h3>';
30
31 // Create subject input:
32 echo '<p>' . $words['subject'] . ': <input name="subject" type="text" size="'
33 60"maxlength ="100" ';
34
35 // Check for existing value:
36 if (isset($subject)) {
37 echo "value=\"$subject\" ";
38 }
39
40 echo '/></p>';
41 }
42
43 // Create the body textarea:
44 echo '<p>' . $words['body'] . ': <textarea name="body" rows="10" cols="60">';
45
46 if (isset($body)) {
47 echo $body;
48 }
49
50 echo '</textarea></p>';
51
52 // Finish the form:
53 echo '<input name="submit" type="submit" value="" . $words['submit'] . '" />
54 </form>';
55
56 } else {
57 echo '<p>You must be logged in to post messages.</p>';
58 }
59
60 ?>
```

(2) 如果直接访问过这个页面，就重定向Web浏览器。

```

if (!isset($words)) {
 header ("Location: http://www. example.com/index.php");
 exit();
}
```

这个脚本没有包括标题和脚注，从而不会建立一个完整的HTML页面，因此必须被具有所有这些元素的脚本包含。这里没有`been_included()`函数，它用于指示是包含这个页面还是直接加载它。作为替代，既然我知道头文件创建一个`$words`变量，如果没有设置该变量，那么就没有在这个脚本之前包含`header.html`，并且应该重定向浏览器。

在`header()`调用中更改URL，以匹配你的站点。

(3) 确认用户已登录并开始创建表单。

```
if (isset($_SESSION['user_id'])) {
 echo '<form action="post.php" method="post"
accept-charset="utf-8">';
```

只有注册用户可以发布消息，因此在显示表单前要检查\$\_SESSION['user\_id']是否存在。表单本身将被提交给post.php，接下来将编写执行该操作的代码。向表单中添加accept-charset属性，明确指示UTF-8文本是可接受的（尽管严格说来不必如此，因为每个页面已经使用了UTF-8编码）。

(4) 检查论点ID。

```
if (isset($tid) && $tid) {
 echo '<h3>' . $words['post_a_reply'] . '</h3>';
 echo '<input name="tid" type="hidden" value="" . $tid . '" />';
```

事情在这里变得有点棘手。如前所述依赖于使用的方式，表单将稍微有所不同，如图17-25和图17-26所示。当包括在read.php上时，它将用于提供对现有论点的回复。为了检查这一点，脚本将查看是否设置了\$tid（thread ID的简写），以及它是否具有一个TRUE值。当这个页面被read.php包含时将会是这样。当这个脚本被post.php包含时，将会设置\$tid，但它具有一个FALSE值。

如果这个条件语句为真，将会打印“Post a Reply”的特定语言的版本，并且将把论点ID存储在一个隐藏的表单输入框中。

(5) 完成开始于第(4)步中的条件语句。

```
} else { // New thread
 echo '<h3>' . $words['new_thread'] . '</h3>';
 echo '<p>' . $words['subject'] . ': <input name="subject" type="text" size="60" maxlength="100" ';
 if (isset($subject)) {
 echo "value=\"$subject\" ";
 }
 echo '/></p>';
} // End of $tid IF.
```

如果这不是一个回复，那么标题应该是“New Thread”的特定语言的版本，并且应该创建一个主题输入框。该输入框需要是黏性的。为了检查这一点，可查看\$subject变量是否存在。将在post.php中创建这个变量，该文件然后将包含这个页面。

(6) 为消息正文创建文本区。

```
echo '<p>' . $words['body'] . ': <textarea name="body" rows="10" cols="60">';
if (isset($body)) {
 echo $body;
}
echo '</textarea></p>';
```

这个页面的两种应用都将具有这个文本区。像主题一样，如果\$body变量（在post.php中定义）存在，它将被创建成具有黏性。对于两个输入框，提示性文字将是特定语言的。

(7) 完成表单。

```
echo '<input name="submit" type="submit" value="" . $words['submit'] . '" />
</form>';
```

余下的全部工作是创建特定语言的提交按钮（参见图17-28）。

**Nouvelle discussion**

Sujet:

Corps:

**Soumettez**

图17-28 表单提示甚至提交按钮都将使用用户选择的语言（对比图17-25、图17-26和图17-27）

(8) 完成页面。

```
} else {
 echo '<p>You must be logged in to post messages.</p>';
}
?>
```

重申一遍，可以把该消息存储在words表中，并在这里使用翻译的版本。仅仅出于简单性考虑，我没有这样做。

(9) 将文件另存为post\_form.php，存放在Web目录的includes文件夹中，并通过访问read.php在Web浏览器中测试它（参见图17-29）。

**Sample Thread**

troutster (29-Oct-11 5:12 AM)  
This is the body of the sample thread. This is the body of the sample thread.  
This is the body of the sample thread.

troutster (29-Oct-11 5:44 AM)  
I like your thread. It's simple and sweet.

You must be logged in to post messages.

图17-29 用户未登录时post\_form.php页面的结果（记住：可以通过在头文件中使用`$_SESSION = array();`这一行代码来模拟未登录的情况）

## 17.6.2 处理表单

这个文件（post.php）将主要用于处理来自post\_form.php的表单提交。这听起来非常简单，但是还有另外一点工作要做。实际上将以三种不同的方式调用这个页面。

(1) 为论点回复处理表单。

(2) 为新论点提交显示表单。

(3) 为新论点提交处理表单。

这意味着将使用POST（模式1和3）或GET（模式2）访问页面。此外，将发送给页面因而需要进行验证的数据将在模式1和3之间有所不同。图17-30显示了逻辑的表示。

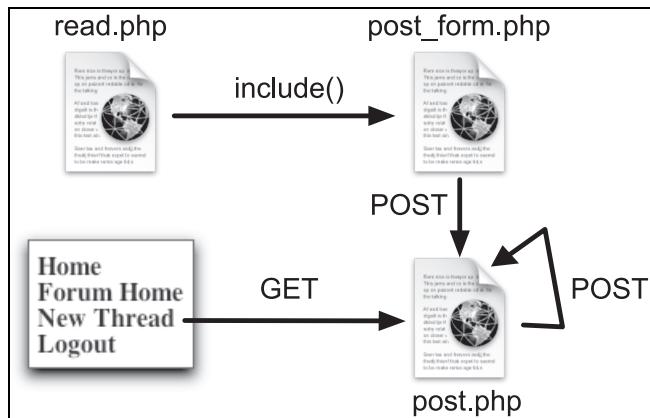


图17-30 post.php页面的不同应用

如果要创建新论点，必须运行两个查询：一个用于把论点添加到threads表中，另一个用于把新论点正文添加到posts表中，这也增加了复杂性。如果提交的是对现有论点的回复，那么只需要一个查询，把记录插入到posts表中。

当然，你将看到，使用适当的条件语句即可成功地完成这个任务。就验证而言，将只会对非空值检查主题和正文（作为文本类型）。从主题中删除所有标签（以防它带有任何标签），并把它转换成正文中的实体。这将允许在帖子中使用HTML、JavaScript和PHP代码，但是在显示论点时仍然不会执行它们（这是由于在关于Web开发的论坛中，你将需要显示一些代码）。

#### 创建post.php

(1) 在文本编辑器或IDE中开始创建一个新的PHP文档，命名为post.php（参见脚本17-7）。

这个页面与post\_form.php不同，它将使用头文件和脚注文件。

**脚本 17-7 在发布消息时 post.php 页面将处理表单提交。这个页面将用于创建新论点以及处理对现有论点的回复**

```

1 <?php # Script 17.7 - post.php
2 // This page handles the message post.
3 // It also displays the form if creating a new thread.
4 include ('includes/header.html');
5
6 if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Handle the form.
7
8 // Language ID is in the session.
9 // Validate thread ID ($tid), which may not be present:

```

```

10 if (isset($_POST['tid']) && filter_var($_POST['tid'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
11 $tid = $_POST['tid'];
12 } else {
13 $tid = FALSE;
14 }
15
16 // If there's no thread ID, a subject must be provided:
17 if (!$tid && empty($_POST['subject'])) {
18 $subject = FALSE;
19 echo '<p>Please enter a subject for this post.</p>';
20 } elseif (!$tid && !empty($_POST['subject'])) {
21 $subject = htmlspecialchars(strip_tags($_POST['subject']));
22 } else { // Thread ID, no need for subject.
23 $subject = TRUE;
24 }
25
26 // Validate the body:
27 if (!empty($_POST['body'])) {
28 $body = htmlentities($_POST['body']);
29 } else {
30 $body = FALSE;
31 echo '<p>Please enter a body for this post.</p>';
32 }
33
34 if ($subject && $body) { // OK!
35
36 // Add the message to the database...
37
38 if (!$tid) { // Create a new thread.
39 $q = "INSERT INTO threads (lang_id, user_id, subject) VALUES ({$_SESSION['lid']}, {$_SESSION['user_id']}, '" . mysqli_real_escape_string($dbc, $subject) . "')";
40 $r = mysqli_query($dbc, $q);
41 if (mysqli_affected_rows($dbc) == 1) {
42 $tid = mysqli_insert_id($dbc);
43 } else {
44 echo '<p>Your post could not be handled due to a system error.</p>';
45 }
46 } // No $tid.
47
48 if ($tid) { // Add this to the replies table:
49 $q = "INSERT INTO posts (thread_id, user_id, message, posted_on) VALUES ($tid, {$_SESSION['user_id']}, '" . mysqli_real_escape_string($dbc, $body) . "', UTC_TIMESTAMP())";
50 $r = mysqli_query($dbc, $q);
51 if (mysqli_affected_rows($dbc) == 1) {
52 echo '<p>Your post has been entered.</p>';
53 } else {
54 echo '<p>Your post could not be handled due to a system error.</p>';
55 }
56 } // Valid $tid.
57
58 } else { // Include the form:
59 include ('includes/post_form.php');
60 }

```

```

61
62 } else { // Display the form:
63
64 include ('includes/post_form.php');
65
66 }
67
68 include ('includes/footer.html');
69 ?>

```

(2) 检查表单提交并验证论点ID。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 if (isset($_POST['tid']) && filter_var($_POST['tid'], FILTER_VALIDATE_INT, array ('min_range' => 1))) {
 $tid = $_POST['tid'];
 } else {
 $tid = FALSE;
 }
}

```

如果将表单作为对现有论点的回复进行提交，论点ID将会存在（论点ID被存储为隐藏的输入框，参见图17-31）。验证过程相当普通，还是使用了过滤器扩展。

```

<form action="post.php" method="post" accept-charset="utf-8"><h3>Post a
Reply</h3><input name="tid" type="hidden" value="7" /><p>Body:
<textarea name="body" rows="10" cols="60"></textarea></p><input
name="submit" type="submit" value="Submit" />

```

图17-31 read.php的源代码说明了如何将论点ID存储在表单中。这向post.php指明提交是一个回复，而不是一个新论点

(3) 验证消息主题。

```

if (!$tid && empty($_POST ['subject'])) {
 $subject = FALSE;
 echo '<p>Please enter a subject for this post.</p>';
} elseif (!$tid && !empty($_POST ['subject'])) {
 $subject = htmlspecialchars(strip_tags($_POST['subject']));
} else { // Thread ID, no need for subject.
 $subject = TRUE;
}

```

关于验证主题的难以处理的部分存在于三种场景。第一种场景是，如果没有有效的论点ID，那么这应该是一个新论点，并且主题不能为空。如果主题为空，那么就会发生错误并且打印一条消息。

第二种场景是，如果没有有效的论点ID并且主题不为空，那么这是一个新论点并且输入了主题，因此应该处理它。在这种情况下，将会使用strip\_tags()函数删除任何标签，并且使用htmlspecialchars()把任何余下的引号转变成它们的实体格式。万一再次显示表单并把主题置于输入框中以使之具有黏性，调用htmlspecialchars()函数将会阻止问题发生。为了清楚起见，如果提交的主题包含双引号，但是正文不完整，则会再次显示表单以及把主题置于value=""内，并且你将会看到问题。

第三种场景是，当把表单作为对现有论点的回复进行提交时。在这种情况下，\$tid将是有效的，并且不需要主题。

(4) 验证正文。

```
if (!empty($_POST['body'])) {
 $body = htmlentities($_POST['body']);
} else {
 $body = FALSE;
 echo '<p>Please enter a body for this post.</p>';
}
```

这是一个容易得多的验证，因此正文总是需要的。如果它存在，将用`htmlentities()`处理它。

(5) 检查是否正确地填写了表单。

```
if ($subject && $body) {
```

(6) 如果合适的话，就创建一个新论点。

```
if (!$tid) {
 $q = "INSERT INTO threads (lang_id, user_id, subject) VALUES ({$_SESSION['lid']}, {$_SESSION['user_id']},
 '" . mysqli_real_escape_string($dbc, $subject) . "')";
 $r = mysqli_query($dbc, $q);
 if (mysqli_affected_rows($dbc) == 1) {
 $tid = mysqli_insert_id($dbc);
 } else {
 echo '<p>Your post could not be handled due to a system error.</p>';
 }
}
```

如果没有论点ID，那么这就是一个新论点，并且必须在`threads`表上运行一个查询。该查询很简单，用于填充三列。其中两个值来自于会话(在用户登录后)。另一个值是主题,`mysqli_real_escape_string()`对它进行了处理。由于已经对主题应用了`strip_tags()`和`htmlspecialchars()`，可能无需使用这个函数也能侥幸取得成功，但是没有必要冒这个风险。

如果查询工作，就意味着它影响了一行，那么将会检索新论点ID。

(7) 添加记录到`posts`表中。

```
if ($tid) {
 $q = "INSERT INTO posts (thread_id, user_id, message, posted_on) VALUES ($tid, {$_SESSION['user_id']},
 '" . mysqli_real_escape_string($dbc, $body) . "', UTC_TIMESTAMP())";
 $r = mysqli_query($dbc, $q);
 if (mysqli_affected_rows($dbc) == 1) {
 echo '<p>Your post has been entered.</p>';
 } else {
 echo '<p>Your post could not be handled due to a system error.</p>';
 }
}
```

仅当论点ID存在时才应该运行这个查询。如果这是对现有论点的回复或者如果刚才在数据库中创建了新论点(第(6)步)，就会发生这种情况。如果那个查询失败，那么将不会运行这个查询。

查询使用论点ID、用户ID(来自会话)、消息正文(出于安全考虑对其应用`mysqli_real_escape_string()`函数)和发布日期填充表中的4个列。对于最后一个值，将使用`UTC_TIMESTAMP()`列，使得它不会绑定到任何一个时区(参见第6章)。

注意，对于这个页面中所有的打印消息，我只使用了硬编码的英语。为了圆满完成示例，这里应

该把所有的消息都存储在words表中并打印它们。

(8) 完成页面。

```

} else { // Include the form:
 include ('includes/post_form.php');
}
} else { // Display the form:
 include ('includes/post_form.php');
}
include ('includes/footer.html');
?>

```

如果提交了表单但是它未完成，就会应用第一个else子句。在这种情况下，将再次包含表单并且它可以是黏性的，因为它将能够访问这里创建的\$subject和\$body。如果直接访问这个页面（通过单击导航中的链接），因此创建GET请求（没有提交表单）并且未设置\$\_POST['submitted']，则会应用第二个else子句。

(9) 将文件另存为post.php，存放在Web目录中，并在Web浏览器中测试它（参见图17-32和图17-33）。

Please enter a subject for this post.

**New Thread**

Subject:

This is the body with **HTML** and some PHP code:  
<?php // Blah!

Body:

图17-32 如果在尝试发布新论点时没有提供主题，则会得到这个结果

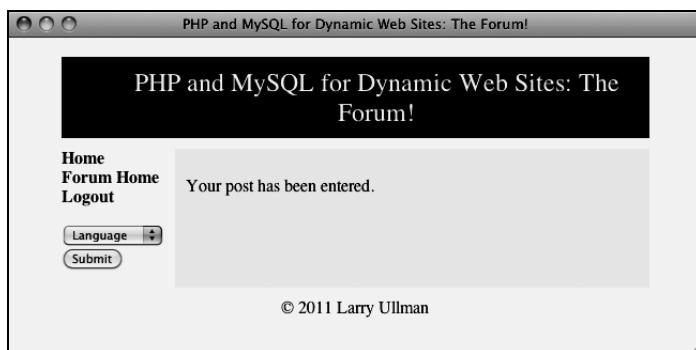


图17-33 成功地将回复添加到论点中

### 这个示例有多复杂

在介绍这个示例时，我指出它从根本上讲很简单，但是有时简单的事情要付出更多的努力来完成。那么，在我看来，这个示例有多复杂？

首先，支持多种语言增添了几何问题。如果到处都没有正确地处理编码（在文本编辑器或IDE中创建页面时，在Web浏览器中与MySQL通信时，等等），事情就可能出错。此外，必须为站点可能需要的任何一点文本提供每种语言的正确译文。这包括出错消息（实际上用户应该会看到它们）、电子邮件正文，等等。

其次，如何组织PHP文件以及它们将要做什么也会使事情变得复杂。特别是，在一个文件中使用在另一个文件中创建的变量。最好情况下，这样做可能会导致混淆，最坏情况下则可能导致错误。为了克服这些问题，我建议添加许多注释，指示变量来自于哪里或者它们可能用在别的什么位置。此外，尽量在页面内使用唯一的变量名称，使得它们不太可能与包含文件中的变量相冲突。

最后，在这个示例中，只使用一个页面显示发布表单并且只使用一个页面处理它，而不管可能在不同的期望下以两种不同的方式发布消息这一事实，这也使这个示例变得复杂。

### 管理论坛

论坛的大部分管理工作都涉及用户管理，这将在下一章中讨论。依赖于谁在管理论坛，你可能创建一些表单，用于管理语言和翻译单词的列表。

管理员也可能有权编辑和删除帖子或论点。为了实现这一点，还要在会话中存储用户级别（下一章将说明如何执行该任务）。如果登录的用户是管理员，就在`forum.php`上添加一些链接，用于编辑和删除论点。每个链接都将把论点ID传递给一个新页面（比如第10章中的`edit_user.php`和`delete_user.php`）。在删除论点时，必须确保在`posts`表中删除也具有那个论点ID的所有记录。这里外键约束会有帮助（详见第6章）。

最后，管理员可以编辑或删除各个帖子（对论点的回复）。同样，检查用户级别，然后添加一些链接到`read.php`上（在每条消息后面添加一对链接）。这些链接将传递帖子ID，以编辑和删除页面（它们不同于对论点使用的页面）。

## 17.7 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

注意，这里的某些问题和提示涉及前面章节介绍的内容，目的是强化里面的重点知识。

17

### 17.7.1 回顾

- 编码对数据库的字符集、PHP或HTML页面有什么影响？
- 为什么编码和字符集在所有的地方都需要是相同的？如果不同会发生什么情况？

- 什么是主键？什么是外键？
- 使用UTC存储日期和时间的好处是什么？
- 为什么user表中的pass列是CHAR类型而不是VARCHAR类型，如果用户的密码是可变长度的呢？
- 如何在PHP中开始一个会话？如何在会话中存储值？如何获取以前存储的值？
- 如何在SQL命令中创建别名？使用别名的好处是什么？

### 17.7.2 实践

- 如果你需要复习数据库设计，查看第6章。
- 查看第6章，以提醒自己表中什么样的列需要索引。
- 如果你的MySQL不能正确地将日期和时间从UTC时区转换为另外一个时期（即，日期转换的返回值为NULL），查看第6章。
- 复习第7章的联结和聚合函数。
- 修改头文件和其他文件，使得每个页面的主标题使用的默认语言页面标题，副标题使用正在浏览的页面标题（例如，当前讨论的名称）。
- 参阅第10章，为forum.php添加分页。
- 如果需要，为words表添加必要的列，为PHP脚本添加相应的代码，让每一个导航、错误和其他元素基于特定语言。使用网站（例如Yahoo!BabelFish，<http://babelfish.yahoo.com>）进行在线翻译。
- 为post\_form.php应用第12章的redirect\_user()函数。
- 为论坛创建一个搜索页。如果你需要一些帮助，查看下载代码中的示范search.php。

# 示例——用户注册

### 本章内容

- 创建模板
- 编写配置脚本
- 创建主页
- 注册
- 激活账户
- 登录和注销
- 密码管理
- 回顾和实践

**本**书的第二个示例（用户注册系统）是PHP和MySQL更常见的应用之一。这里开发的大多数脚本在前面的章节中已经作过介绍和解释，因为注册、登录和注销过程有助于建立许多概念的良好示例。但是本章将使用一致的编程理论把它们全都放在相同的环境内。

用户将能够注册、登录、注销、更改他们的密码。其他地方未说明的一个特性是能够重置密码（万一忘记了密码）。另一个特性是一项要求：在用户可以登录之前，要求用户激活他们的账户——通过单击电子邮件中的一个链接。一旦用户登录，就会使用会话来限制对页面的访问，并跟踪用户。本书这一版的新增内容是支持不同的用户级别，从而允许你根据登录用户的类型来控制可用的内容。

与上一章一样，这里重点关注的是事物公开的一面（永远不要感到恐惧，第17章介绍了一些管理知识）。当然，我将在本章末尾包含一些注释，讨论你可以做什么来添加管理特性。沿着这条路前进，还会看到关于如何能够轻松地扩展或修改这个应用程序的建议。

### 18.1 创建模板

本章中的应用程序将使用一个新的模板设计（参见图18-1）。这个模板广泛利用CSS（Cascading Style Sheet，层叠样式表），创建一个无需图像的清洁外观。它在所有当前的浏览器上都测试良好，并且将在不支持CSS 2的浏览器上显示为未格式化的文本。该站点的布局由BlueRobot（[www.bluerobot.com](http://www.bluerobot.com)）免费提供。

首先，将编写两个模板文件：`header.html`和`footer.html`。与第12章中的示例一样，脚注文件将依赖于用户是否登录（这是通过检查某个会话变量存在与否来确定的）来显示某些链接。进一步讲，如

果登录的用户也是一位管理员（一个会话值将指示这一点），则还会显示另外一些链接。



图18-1 这个Web应用程序的基本外观

头文件将开始会话和输出缓冲，而脚注文件将结束输出缓冲。本书中没有正式介绍输出缓冲，但是在框注“使用输出缓冲”中充分介绍了它。

### 使用输出缓冲

默认情况下，PHP脚本打印的任何内容或者PHP标签外面的任何HTML（甚至在包含文件中）都会立即发送到Web浏览器。输出缓冲（output buffering）[或输出控制（output control），因为PHP手册中这样称呼它]是一种重写这种行为的PHP特性。它不会立即把HTML发送到Web浏览器，而是把输出存放在缓冲区（临时内存）中。然后，当冲洗（flush）缓冲区时，把它发送到Web浏览器。利用输出缓冲可以改进性能，但是其主要好处在于：它实际上会根除那些讨厌的头部已发送（headers already sent）出错消息。仅当没有给Web浏览器发送任何内容时，才会调用一些函数——`header()`、`setcookie()`和`session_start()`。利用输出缓冲，在到达页面的末尾之前，不会向Web浏览器发送任何内容，因此可以在脚本中的任意位置自由地调用这些函数。

为了开始输出缓冲，可使用`ob_start()`函数。一旦调用了该函数，每个`echo()`、`print()`和类似的函数都将把数据发送到内存缓冲区，而不会发送到Web浏览器。相反，HTTP调用（比如`header()`和`setcookie()`）将不会被缓冲，而会像以往那样工作。

在脚本末尾，调用`ob_end_flush()`函数，把累积的缓冲区数据发送到Web浏览器。或者使用`ob_end_clean()`函数删除缓冲的数据，而不是发送它。这两个函数都有关闭输出缓冲的副作用。

#### 1. 建立header.html

(1) 在文本编辑器或IDE中创建一个新的文档，命名为`header.html`（参见脚本18-1）。

**脚本 18-1 头文件开始创建 HTML 代码、启动会话并打开输出缓冲**

```

1 <?php # Script 18.1 - header.html
2 // This page begins the HTML header for the site.
3
4 // Start output buffering:
5 ob_start();
6
7 // Initialize a session:
8 session_start();
9
10 // Check for a $page_title value:
11 if (!isset($page_title)) {
12 $page_title = 'User Registration';
13 }
14 ?>
15 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
16 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
17 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
18 <head>
19 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
20 <title><?php echo $page_title; ?></title>
21 <style type="text/css" media="screen">@import "includes/layout.css";</style>
22 </head>
23 <body>
24 <div id="Header">User Registration</div>
25 <div id="Content">
26 <!-- End of Header -->
```

(2) 开始输出缓冲，并启动会话。

```
ob_start();
session_start();
```

我将为这个应用程序使用输出缓冲，使得在我使用HTTP头部、重定向用户或发送cookie时，不必担心出错消息。每个页面也都会利用会话。因为还没有向Web浏览器发送任何内容，所以把`session_start()`调用放在`ob_start()`后面是安全的。

因为每个公共页面都会使用这两种技术，所以把这几行代码放在`header.html`文件中可以省却把它们放在每个单独页面中的麻烦。其次，如果你以后想更改会话设置，只需编辑这一个文件即可。

(3) 检查`$page_title`变量，并关闭PHP部分。

```
if (!isset($page_title)) {
 $page_title = 'User Registration';
}
?>
```

与本书中其他时间使用模板系统一样，将逐页设置页面的标题——它出现在浏览器窗口的顶部。这个条件语句将检查`$page_title`变量是否具有一个值，如果它没有一个值，则将其设置成一个默认的字符串。这是为了包含头部的一个很好的检查（但它是可选的）。

(4) 创建HTML头部。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

```

<head>
 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
 <title><?php echo $page_title;?></title>
 <style type="text/css" media="screen">@import "includes/layout.css";</style>
</head>

```

这里将在标题标签之间打印出PHP的\$page\_title变量。然后，包含CSS文档。它被命名为layout.css，并存储在名为includes的目录中。可以从本书的支持Web站点上下载该CSS文件。

(5) 开始创建HTML体。

```

<body>
 <div id="Header">User Registration</div>
 <div id="Content">

```

HTML体将创建跨页面顶部的横幅，然后开始创建Web页面的内容部分（直到图18-1中的Welcome!）。

(6) 将文件另存为header.html。

## 2. 建立footer.html

(1) 在文本编辑器或IDE中创建一个新的文档，命名为footer.html（参见脚本18-2）。

**脚本 18-2** 脚注文件结束HTML代码，它依据用户状态（登录与否、是否是管理员）显示链接，并把输出冲洗到Web浏览器

```

1 <!-- Start of Footer -->
2 </div><!-- Content -->
3
4 <div id="Menu">
5 Home

6 <?php # Script 18.2 - footer.html
7 // This page completes the HTML template.
8
9 // Display links based upon the login status:
10 if (isset($_SESSION['user_id'])) {
11
12 echo 'Logout

13 Change Password

14 ';
15
16 // Add links if the user is an administrator:
17 if ($_SESSION['user_level'] == 1) {
18 echo 'View Users

19 Some Admin Page

20 ';
21 }
22
23 } else { // Not logged in.
24 echo 'Register

25 Login

26 Retrieve Password

27 ';
28 }
29 ?>
30 Some Page

31 Another Page


```

```

32 </div><!-- Menu -->
33
34 </body>
35 </html>
36 <?php // Flush the buffered output.
37 ob_end_flush();
38 ?>paste code here

```

(2) 如果用户已经登录，就显示注销和更改密码链接。

```

if (isset($_SESSION['user_id'])) {
 echo 'Logout

 Change Password

';

```

如果用户已经登录（这意味着设置了\$\_SESSION ['user\_id']），将看到用于注销以及用于更改其密码的链接（参见图18-2）。



图18-2 用户在登录时将看到这些导航链接

(3) 如果用户还是管理员，则会显示其他一些链接。

```

if ($_SESSION['user_level'] == 1) {
 echo 'View Users

 Some Admin Page

';
}

```

如果登录的用户碰巧还是管理员，那么他们应该会看到另外一些链接（参见图18-3）。为了测试这一点，可检查用户的访问级别，它也存储在会话中。级别值1指示用户是一位管理员（非管理员用户的级别值为0）。

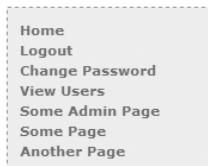


图18-3 登录的管理员将看到另外一些链接（对比图18-2）

(4) 为没有登录的用户显示一些链接并完成PHP部分。

```

} else { // Not logged in.
 echo 'Register

 Login

 Retrieve Password

';
}
?>

```

如果用户没有登录，他们将看到用于注册、登录和重置被遗忘密码的链接（参见图18-4）。

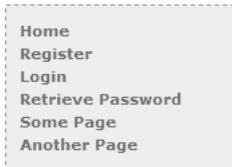


图18-4 如果用户没有登录，将会看到这些链接

(5) 完成HTML代码。

```

Some Page

Another Page

</div>
</body>
</html>

```

我包含了两个虚拟的链接，用于其他要添加的页面。

(6) 把缓冲区冲洗到Web浏览器。

```

<?php
ob_end_flush();
?>

```

脚注文件将把累积的缓冲区发送到Web浏览器，完成在头文件脚本中开始的输出缓冲（详见框注“使用输出缓冲”）。

(7) 将文件另存为footer.html，把它与header.html和layout.css（来自本书的支持Web站点）一起存放在Web目录中，并把全部3个文件都放在includes文件夹中（参见图18-5）。

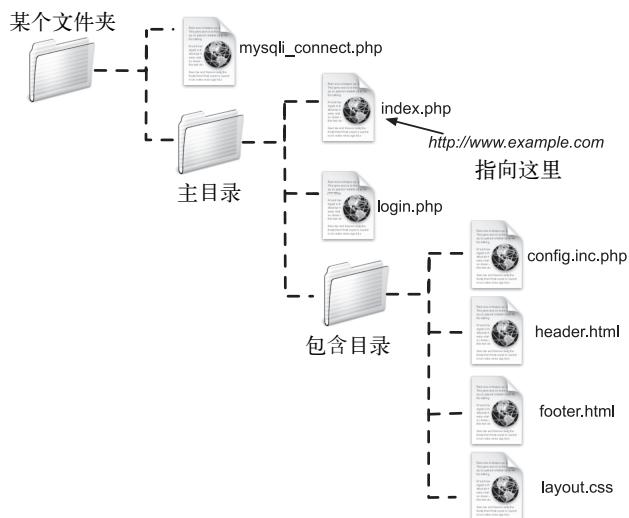


图18-5 Web服务器上站点的目录结构，假定htdocs是文档根目录  
( www.example.com 指向这里 )

**✓ 提示**

- 如果这个站点中的任何页面没有利用头文件，但是需要处理会话，那么它必须自己调用 `session_start()`。如果不能这样做，那么脚本将不能访问会话。
- 在PHP的最新版本中，默认启用输出缓冲。缓冲区大小（内存中存储的最大字节数）是4096，但是可以在PHP的配置文件中更改它。
- `ob_get_contents()`函数将返回当前缓冲区，使得在需要时可以将其赋予一个变量。
- `ob_flush()`函数将把缓冲区的当前内容发送到Web浏览器然后丢弃它们，从而允许开启一个新的缓冲区。该函数允许脚本维持更适中的缓冲区大小。相反，在把缓冲区内容发送到Web浏览器之后，`ob_end_flush()`将会关闭输出缓冲。
- `ob_clean()`函数用于删除缓冲区的当前内容，但不会停止缓冲区进程。
- 在脚本运行结束时，如果它没有运行`ob_end_flush()`函数，PHP将自动运行它。

## 18.2 编写配置脚本

这个Web站点将利用两种配置类型的脚本。一种是`config.inc.php`，它确实是整个应用程序中最重要的脚本。它将会：

- 具有关于作为一个整体的站点的注释；
- 定义常量；
- 建立站点设置；
- 规定如何处理错误；
- 定义任何必要的函数。

由于它将会做所有这些工作，所以应用程序中的所有其他页面都会包含这个配置脚本。

另一种配置类型的脚本是`mysqli_connect.php`，将存储所有与数据库相关的信息。只有那些需要与数据库交互的页面才会包含它。

### 18.2.1 建立配置文件

配置文件将服务于许多重要的目的。它就像是站点所有者的手册与其参数设置文件之间的桥梁。该文件的第一个用途是从总体上用文档记录站点：是谁创建它、什么时候创建的、为什么创建它，以及是为谁创建它的，等等。本书将省略所有这些方面，但是你自己应该记住这些。第二个作用是定义各个页面将使用的各类常量和设置。

第三个用途是，配置文件将为站点建立错误管理策略。第8章中介绍了相关的技术——创建自己的错误处理函数。就像在那一章中一样，在开发阶段，将以最详细的方式报告每个错误（参见图18-6）。将与特定的出错消息一起显示所有现有的变量，同时还会显示当前日期和时间。将会对其进行格式化，使得它能够放在站点的模板内。在站点的制作或运行阶段，将会更优雅地处理错误（参见图18-7）。在那时，不会在Web浏览器中打印详细的出错消息，而是把它们发送到一个电子邮件地址。



图18-6 在Web站点的开发阶段，我想让所有的错误表现得更明显，并且尽可能地提供丰富的信息

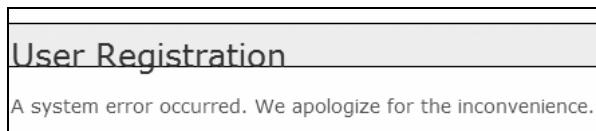


图18-7 如果在站点运行时发生错误，用户将只会看到一条像这样的消息（但是将通过电子邮件把详细的出错消息发送给管理员）

最后，这个脚本将定义可能在站点中多次使用的任何函数。这个站点将不会具有任何这样的函数，但是我希望提到这一点，以将其作为这样一个文件的另一种逻辑应用。

#### 编写配置文件

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为config.inc.php（参见脚本18-3）。

**脚本 18-3** 这个配置脚本规定了如何处理错误，定义了站点级设置和常量，并且可以（但并没有）声明任何必要的函数

```

1 <?php # Script 18.3 - config.inc.php
2 /* This script:
3 * - define constants and settings
4 * - dictates how errors are handled
5 * - defines useful functions
6 */
7
8 // Document who created this site, when, why, etc.
9
10
11 // **** SETTINGS **** //
12 // ***** SETTINGS ***** //

```

```
13 // Flag variable for site status:
14 define('LIVE', FALSE);
15
16 // Admin contact address:
17 define('EMAIL', 'InsertRealAddressHere');
18
19 // Site URL (base for all redirections):
20 define ('BASE_URL', 'http://www.example. com/');
21
22 // Location of the MySQL connection . script:
23 define ('MYSQL', '/path/to/mysql_connect. php');
24
25 // Adjust the time zone for PHP 5.1 and greater:
26 date_default_timezone_set ('US/Eastern');
27
28 // ***** SETTINGS ***** //
29 // ***** ERROR MANAGEMENT ***** //
30
31
32 // ***** //
33 // ***** ERROR MANAGEMENT ***** //
34
35
36 // Create the error handler:
37 function my_error_handler ($e_number, $e_message, $e_file, $e_line, $e_vars) {
38
39 // Build the error message:
40 $message = "An error occurred in script '$e_file' on line $e_line: $e_message\n";
41
42 // Add the date and time:
43 $message .= "Date/Time: " . date('n-j-Y H:i:s') . "\n";
44
45 if (!LIVE) { // Development (print the error).
46
47 // Show the error message:
48 echo '<div class="error">' . nl2br($message);
49
50 // Add the variables and a backtrace:
51 echo '<pre>' . print_r ($e_vars, 1) . "\n";
52 debug_print_backtrace();
53 echo '</pre></div>';
54
55 } else { // Don't show the error:
56
57 // Send an email to the admin:
58 $body = $message . "\n" . print_r ($e_vars, 1);
59 mail(EMAIL, 'Site Error!', $body, 'From: email@example.com');
60
61 // Only print an error message if the error isn't a notice:
62 if ($e_number != E_NOTICE) {
63 echo '<div class="error">A system error occurred. We apologize for the inconvenience.
64 </div>
';
65 } // End of !LIVE IF.
```

```

66
67 } // End of my_error_handler() definition.
68
69 // Use my error handler:
70 set_error_handler ('my_error_handler');
71
72 // ***** ERROR MANAGEMENT *****
73 // *****

```

(2) 建立两个常量用于错误报告。

```

define('LIVE', FALSE);
define('EMAIL', 'InsertRealAddress Here');

```

将像第8章中那样使用LIVE常量。如果它是FALSE，就会把详细的出错消息发送到Web浏览器（参见图18-6）。一旦运行站点，就应该把这个常量设置为TRUE，使得永远不会把详细的出错消息呈现给Web用户（参见图18-7）。在运行站点时，将把出错消息发送给EMAIL常量。显然，你将为这个值使用你自己的电子邮件地址。

(3) 定义两个常量用于站点级设置。

```

define ('BASE_URL', 'http://www. example.com/');
define ('MYSQL', '/path/to/mysql_connect.php');

```

定义这两个常量仅仅为了更容易地把用户从一个页面重定向到另一个页面，以及包含MySQL连接脚本。BASE\_URL指根域（<http://www.example.com/>），它带有一个终止斜杠。如果在你自己的计算机上开发应用程序，这可能是<http://localhost/>或<http://localhost/ch18/>。当页面重定向浏览器时，它现在只需要编写如下代码：

```
header('Location: ' . BASE_URL . 'page.php');
```

MYSQL是指向MySQL连接脚本（接下来将编写它）的绝对路径。通过把它设置为一条绝对路径，任何文件都可以通过引用这个常量来包含连接脚本：

```
require (MYSQL);
```

更改这两个值使之对应于你的环境。例如，如果在Windows系统中使用XAMPP，MySQL常量的值可能就是c:\\xampp\\mysql\\connect.php。

如果把站点从一个服务器或域移到另一个服务器或域，只需更改这两个常量即可。

(4) 建立任何其他的站点级设置。

```
date_default_timezone_set ('US/Eastern');
```

如第11章中所述，使用任何PHP日期或时间函数（从PHP 5.1起）都要求设置时区。更改这个值，使之匹配你的时区（参见PHP手册，了解时区列表）。

(5) 开始定义错误处理函数。

```

function my_error_handler ($e_number, $e_message, $e_file, $e_line, $e_vars) {
 $message = "An error occurred in script '$e_file' on line $e_line: $e_message\n";

```

这个函数定义的开始部分类似于第8章中的一个函数。它期待接收5个参数：错误编号、出错消息、发生错误的脚本、PHP认为发生错误的行号，以及现存变量的数组。然后，它开始定义\$message变量，从提供给该函数的信息开始。

(6) 添加当前日期和时间。

```
$message .= "Date/Time: " . date('n-j-Y H:i:s') . "\n";
```

为了使错误报告更有用，我将把当前日期和时间包括在消息中。代码中包括了一个换行符和一个HTML标签<br />，以使得到的显示结果更清晰易读。

(7) 如果站点没有运行，显示详细错误信息。

```
if (!LIVE) {
 echo '<div class="error">' . nl2br($message);
 echo '<pre>' . print_r ($e_vars, 1) . "\n";
 debug_print_backtrace();
 echo '</pre></div>';
```

我早先就提过，如果站点没有运行，就会打印任何类型的完整出错消息。将会用<div class="error">包围消息，它将依据站点的CSS文件中定义的规则来格式化消息。错误信息的第一部分是已经定义的字符串，并且将换行转换为HTML换行标签，然后使用预格式化标签，显示在错误出现时的所有变量和回溯（函数调用历史记录等）。查看第8章来获取关于此内容的更多解释。

(8) 如果站点在运行，发送电子邮件给管理员报告详细消息，并向用户打印一条普通消息。

```
} else { // Don't show the error:
 $body = $message . "\n" . print_r ($e_vars, 1);
 mail(EMAIL, 'Site Error!', $body, 'From: email@example.com');
 if ($e_number != E_NOTICE) {
 echo '<div class="error">A system error occurred. We apologize for the inconvenience.</div>
';
 }
} // End of !LIVE IF.
```

如果站点在运行，就应该在一封电子邮件中发送详细的消息，并且Web用户只应该看到一条普通消息。可以更进一步，如果错误具有特定的类型E\_NOTICE，则不打印普遍消息。这类错误是针对像引用一个不存在的变量这样的事情发生的，这可能是或者可能不是一个问题。为了避免出错消息潜在地淹没用户，仅当\$e\_number不等于E\_NOTICE时，才会打印出错消息，E\_NOTICE是在PHP中定义的一个常量（见PHP手册）。

(9) 完成函数定义，并告诉PHP使用错误处理程序。

```
}
```

```
set_error_handler ('my_error_handler');
```

必须使用set\_error\_handler()函数告诉PHP针对错误使用你自己的函数。

(10) 将文件另存为config.inc.php，存放在Web目录中，并将其放在includes文件夹中。

与本书中的其他示例一样，本例也省略了PHP结束标签，为的是让此脚本可以被其他PHP脚本包含。

## 18.2.2 建立数据库脚本

第二个配置类型的脚本是mysqli\_connect.php，它是本书中已经使用多次的数据库连接文件。其唯一目的是连接到MySQL，并选择数据库。如果发生问题，这个脚本将利用config.inc.php中建立的错误处理工具。为了执行该操作，它将使用trigger\_error()函数。这个函数可让你告诉PHP发生了一个错误。当然，PHP将使用配置脚本中所建立的my\_error\_handler()函数来处理那个错误。

### 连接到数据库

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为mysqli\_connect.php（参见脚本18-4）。

**脚本 18-4** 这个脚本连接到 ch18 数据库。如果它不能这样做，就会触发错误处理程序，将 MySQL 连接错误传递给它

```

1 <?php # Script 18.4 - mysqli_connect.php
2 // This file contains the database access information.
3 // This file also establishes a connection to MySQL
4 // and selects the database.
5
6 // Set the database access information as constants:
7 DEFINE ('DB_USER', 'username');
8 DEFINE ('DB_PASSWORD', 'password');
9 DEFINE ('DB_HOST', 'localhost');
10 DEFINE ('DB_NAME', 'ch18');
11
12 // Make the connection:
13 $dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
14
15 // If no connection could be made, trigger an error:
16 if (!$dbc) {
17 trigger_error ('Could not connect to MySQL: ' . mysqli_connect_error());
18 } else { // Otherwise, set the encoding:
19 mysqli_set_charset($dbc, 'utf8');
20 }
```

(2) 设置数据库访问信息。

```

DEFINE ('DB_USER', 'username');
DEFINE ('DB_PASSWORD', 'password');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'ch18');
```

一如既往，把这些值更改为那些适合你的MySQL安装的值。

(3) 试图连接到MySQL，并选择数据库。

```
$dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
```

在以前的脚本中，如果函数没有返回正确的结果，就会调用die()函数。因为将使用我自己的错误处理函数，而不是简单地终止脚本，所以我将重写这个过程。

这个函数调用引发的任何错误都会被阻止（由于@），并使用下一步中的代码进行处理。

(4) 如果不能建立数据库连接，则处理任何错误。

```
if (!$dbc) {
 trigger_error ('Could not connect to MySQL: ' . mysqli_connect_error());
```

如果脚本不能连接到数据库，我希望把出错消息发送给my\_error\_handler()函数。通过这样做，可以确保依据当前设置的管理技术（运行阶段与开发阶段）来处理错误。不是直接调用my\_error\_handler()，而是使用trigger\_error()，它的第一个参数是出错消息。图18-8显示了在开发阶段发生问题时的最终结果。



图18-8 在站点的开发期间发生的数据库连接错误

(5) 创建编码。

```

} else {
 mysqli_set_charset($dbc, 'utf8');
}

```

如果可以创建数据库连接，那么用于与数据库通信的编码将会被确立。查看第9章了解更多详情。

(6) 将文件保存为`mysqli_connect.php`，并将它放置到Web文档根目录的上层目录。

这个脚本作为一个可以被包含的文件，同样省略了PHP结束标签。和本书的其他例子一样，理想情况下这个文件不应该放到Web目录，但是不论你将它放到什么地方，确保MYSQL常量的值要（在`config.inc.php`）相互匹配。

(7) 创建数据库（参见图18-9）。

```

mysql> CREATE DATABASE chs18;
Query OK, 1 row affected (0.00 sec)

mysql> USE chs18;
Database changed

mysql> CREATE TABLE users (
-> user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> first_name VARCHAR(20) NOT NULL,
-> last_name VARCHAR(40) NOT NULL,
-> email VARCHAR(60) NOT NULL,
-> pass CHAR(40) NOT NULL,
-> user_level TINYINT(1) UNSIGNED NOT NULL DEFAULT 0,
-> active CHAR(32),
-> registration_date DATETIME NOT NULL,
-> PRIMARY KEY (user_id),
-> UNIQUE KEY (email),
-> INDEX login (email, pass)
->);
Query OK, 0 rows affected (0.05 sec)

mysql>

```

图18-9 为本章创建数据库

参见框注“数据库模式”，了解关于数据库以及建立一个表所需命令的讨论。如果不能创建你自己的数据库，只需把表添加到你有权访问的任何数据库中即可。另外，一定要编辑`mysqli_connect.php`文件，以便它可以使用正确的用户名/密码/主机名组合来连接到这个数据库。

### 数据库模式

这个应用程序使用的数据库被命名为ch18。该数据库目前只包含一个表users。为了创建这个表，可使用如下SQL命令：

```
CREATE TABLE users (
 user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 first_name VARCHAR(20) NOT NULL,
 last_name VARCHAR(40) NOT NULL,
 email VARCHAR(60) NOT NULL,
 pass CHAR(40) NOT NULL,
 user_level TINYINT(1) UNSIGNED NOT NULL DEFAULT 0,
 active CHAR(32),
 registration_date DATETIME NOT NULL,
 PRIMARY KEY (user_id),
 UNIQUE KEY (email),
 INDEX login (email, pass)
);
```

目前，你应该熟悉这个表的大部分结构，它与本书中多个示例中使用过的sitename数据库中的users表非常相似。一个新增部分是active列，它将用于指示用户是否激活了他们的账户（通过单击注册电子邮件中的一个链接）。它将存储长度为32个字符的激活码，或者具有一个NULL值。由于active列可能具有一个NULL值，所以不能把它定义为NOT NULL。如果确实把active定义为NOT NULL，那么人们永远也不能够登录（在本章后面将了解为什么会这样）。另一个新增部分是user\_level列，它将用于区分站点具有的各类用户。

在email字段上放置一个唯一的索引，在email和pass这两个字段的组合上则放置另一个索引。在登录查询期间，将结合使用这两个字段，因此将它们作为一个字段（我称之为login）进行索引是有意义的。

#### ✓ 提示

- ❑ 一方面，把两个配置文件的内容放在一个脚本中可能是有意义的，因为这易于引用。不幸的是，这样做将给不需要数据库连接的脚本（例如，`index.php`）增加不必要的系统开销（即连接并选择数据库）。
- ❑ 一般来讲，将在配置文件中定义一些公共函数。一个例外是任何需要数据库连接的函数。如果你知道一个函数将只会在连接到MySQL的页面上使用，那么在`mysqli_connect.php`脚本内定义该函数是唯一合乎逻辑的做法。

## 18.3 创建主页

站点的主页称为index.php，它是公共端上其他页面的模型。它需要配置文件（用于错误管理）、头文件和脚注文件来创建HTML设计。这个页面还会通过名字来欢迎用户，假定用户已登录（参见图18-10）。

### 编写index.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为index.php（参见脚本18-5）。

#### 脚本 18-5 用于站点主页的脚本，它用名字来问候登录的用户

```

1 <?php # Script 18.5 - index.php
2 // This is the main page for the site.
3
4 // Include the configuration file:
5 require ('includes/config.inc.php');
6
7 // Set the page title and include the HTML header:
8 $page_title = 'Welcome to this Site!';
9 include ('includes/header.html');
10
11 // Welcome the user (by name if they are logged in):
12 echo '<h1>Welcome';
13 if (isset($_SESSION['first_name'])) {
14 echo ", {" . $_SESSION['first_name'] . "}";
15 }
16 echo '!</h1>';
17 ?>
18 <p>Spam spam spam spam spam
19 spam spam spam spam spam
20 spam spam spam spam spam
21 spam spam spam spam spam.</p>
22 <p>Spam spam spam spam spam
23 spam spam spam spam spam
24 spam spam spam spam spam
25 spam spam spam spam spam.</p>
26
27 <?php include ('includes/footer.html'); ?>
```

(2) 包含配置文件，设置页面标题，并纳入HTML头文件。

```

require ('includes/config.inc.php');
$page_title = 'Welcome to this Site!';
include ('includes/header.html');
```

脚本首先会包含配置文件，这使得可以使用在此建立的错误管理过程来处理往后发生的所有事情。然后会包含header.html文件，它将启动输出缓冲、开始会话，并创建HTML布局的初始部分。

(3) 问候用户并完成PHP代码。

```

echo '<h1>Welcome';
if (isset($_SESSION['first_name']))
{
```

```

echo ",
{$_SESSION['first_name']}";
}
echo '!</h1>';
?>

```

将为所有用户打印Welcome消息。如果设置了\$\_SESSION['first\_name']变量，还会打印用户的名  
字。因此，最终结果将只是Welcome（参见图18-11）或Welcome,<Your Name>!（参见图18-10）。

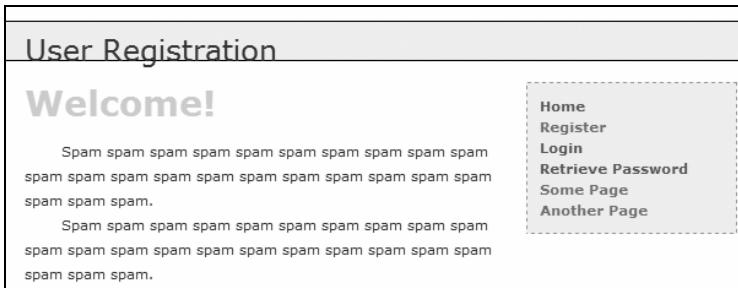


图18-10 如果用户已登录，索引页面将用名字问候他们

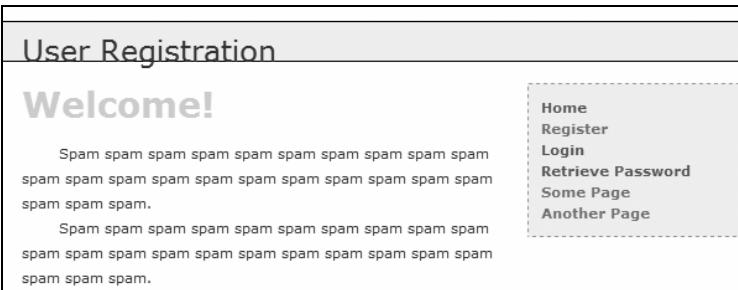


图18-11 如果用户没有登录，这就是他们将看到的主页

(4) 创建页面的内容。

```
<p>Spam spam...</p>
```

在真实的站点上，你可能希望在主页上放置一些更有用的内容。只是一种建议……

(5) 包含HTML脚注。

```
<?php include ('includes/footer.html'); ?>
```

脚注文件将完成HTML布局（主要是页面右侧的菜单栏）并终止输出缓冲。

(6) 将文件另存为index.php，存放在Web目录中，并在Web浏览器中测试它。

## 18.4 注册

注册脚本最初是在第8章中开始编写的。从那时起，以许多方式对其进行改进。register.php的这个版本将做以下事情：

- 显示和处理表单；
- 使用正则表达式验证提交的数据；
- 如果发生问题，则用记住的值重新显示表单（表单将具有黏性）；
- 出于安全考虑使用`mysqli_real_escape_string()`函数处理提交的数据；
- 确保电子邮件地址唯一；
- 发送一封包含激活链接的电子邮件（用户在登录前必须激活他们的账户——参见框注“激活过程”）。

### 激活过程

本章中的新增内容是激活过程，其中用户在能够登录之前，必须单击电子邮件中的一个链接以确认他们的账户。使用这样一个系统可以阻止使用伪造的注册。如果输入无效的电子邮件地址，那么永远也不能激活那个账户。如果某个人注册了另一个人的地址，那么运气好的话那个人将不会激活这个不想要的账户。

从程序设计的角度讲，这个过程需要为每个注册用户创建唯一的激活码，并把它们存储在`users`表中。然后在确认电子邮件中把激活码发送给用户（在链接中）。当用户单击链接时，将把他们带到站点上的一个页面以激活他们的账户（通过从他们的记录中删除该激活码）。通过使用这个激活码，不仅让用户在未使用它的情况下到达激活页面，而且可以阻止人们在没有接收到确认电子邮件的情况下激活账户。

### 编写register.php

(1) 在文本编辑器或IDE中创建一个新的文档，命名为`register.php`（参见脚本18-6）。

**脚本 18-6** 这个注册脚本使用了正则表达式和黏性表单，前者是出于安全考虑，后者则是为了方便用户。在成功注册时，它会发送一封电子邮件给用户

```

1 <?php # Script 18.6 - register.php
2 // This is the registration page for the site.
3 require ('includes/config.inc.php');
4 $page_title = 'Register';
5 include ('includes/header.html');
6
7 if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Handle the form.
8
9 // Need the database connection:
10 require (MYSQL);
11
12 // Trim all the incoming data:
13 $trimmed = array_map('trim', $_POST);
14
15 // Assume invalid values:
16 $fn = $ln = $e = $p = FALSE;
17
18 // Check for a first name:
19 if (preg_match ('/^([A-Z \'.-]{2,20})$/i', $trimmed['first_name'])) {

```

```

20 $fn = mysqli_real_escape_string ($dbc, $trimmed['first_name']);
21 } else {
22 echo '<p class="error">Please enter your first name!</p>';
23 }
24
25 // Check for a last name:
26 if (preg_match ('/^A-Z \'.-]{2,40}$/', $trimmed['last_name'])) {
27 $ln = mysqli_real_escape_string ($dbc, $trimmed['last_name']);
28 } else {
29 echo '<p class="error">Please enter your last name!</p>';
30 }
31
32 // Check for an email address:
33 if (filter_var($trimmed['email'], FILTER_VALIDATE_EMAIL)) {
34 $e = mysqli_real_escape_string ($dbc, $trimmed['email']);
35 } else {
36 echo '<p class="error">Please enter a valid email address!</p>';
37 }
38
39 // Check for a password and match against the confirmed password:
40 if (preg_match ('/^\w{4,20}$/', $trimmed['password1'])) {
41 if ($trimmed['password1'] == $trimmed['password2']) {
42 $p = mysqli_real_escape_string ($dbc, $trimmed['password1']);
43 } else {
44 echo '<p class="error">Your password did not match the confirmed password!</p>';
45 }
46 } else {
47 echo '<p class="error">Please enter a valid password!</p>';
48 }
49
50 if ($fn && $ln && $e && $p) { // If everything's OK...
51
52 // Make sure the email address is available:
53 $q = "SELECT user_id FROM users WHERE email='$e'";
54 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
55
56 if (mysqli_num_rows($r) == 0) { // Available.
57
58 // Create the activation code:
59 $a = md5(uniqid(rand(), true));
60
61 // Add the user to the database:
62 $q = "INSERT INTO users (email, pass, first_name, last_name, active, registration_date)
VALUES ('$e', SHA1('$p'), '$fn', '$ln', '$a', NOW())";
63 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
64
65 if (mysqli_affected_rows($dbc) == 1) { // If it ran OK.
66
67 // Send the email:
68 $body = "Thank you for registering at <whatever site>. To activate your account, please
click on this link:\n\n";
69 $body .= BASE_URL . 'activate.php?x=' . urlencode($e) . "&y=$a";
70 mail($trimmed['email'], 'Registration Confirmation', $body, 'From: admin@sitename.com');
71

```

```

72 // Finish the page:
73 echo '<h3>Thank you for registering! A confirmation email has been sent to your address.
74 Please click on the link in that email in order to activate your account.</h3>';
75 include ('includes/footer.html'); // Include the HTML footer.
76 exit(); // Stop the page.
77
78 } else { // If it did not run OK.
79 echo '<p class="error">You could not be registered due to a system error. We apologize
80 for any inconvenience.</p>';
81 }
82
83 } else { // The email address is not available.
84 echo '<p class="error">That email address has already been registered. If you have forgotten
85 your password, use the link at right to have your password sent to you. </p>';
86 }
87
88 } else { // If one of the data tests failed.
89 echo '<p class="error">Please try again.</p>';
90 }
91
92 mysqli_close($dbc);
93
94 } // End of the main Submit conditional.
95 ?>
96
97 <h1>Register</h1>
98 <form action="register.php" method="post">
99 <fieldset>
100
101 <p>First Name: <input type="text" name="first_name" size="20" maxlength="20" value="<?php if (isset($trimmed['first_name'])) echo $trimmed['first_name']; ?>" /></p>
102
103 <p>Last Name: <input type="text" name="last_name" size="20" maxlength="40" value="<?php if (isset($trimmed['last_name'])) echo $trimmed['last_name']; ?>" /></p>
104
105 <p>Email Address: <input type="text" name="email" size="30" maxlength="60" value="<?php if (isset($trimmed['email'])) echo $trimmed['email']; ?>" /> <small>Use only letters,
106 numbers, and the underscore. Must be between 4 and 20 characters long. </small></p>
107
108 <p>Password: <input type="password" name="password1" size="20" maxlength="20" value="<?php if (isset($trimmed['password1'])) echo $trimmed['password1']; ?>" /> <small>Use only letters,
109 numbers, and the underscore. Must be between 4 and 20 characters long. </small></p>
110
111 <div align="center"><input type="submit" name="submit" value="Register" /></div>
112
113 </form>
114
115 <?php include ('includes/footer.html'); ?>
```

(2) 包含配置文件和HTML头文件。

```
require ('includes/config.inc.php');
$page_title = 'Register';
include ('includes/header.html');
```

(3) 创建用于检查表单提交的条件语句，然后包含数据库连接脚本。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 require (MYSQL);
```

因为到`mysqli_connect.php`脚本的全路径是作为一个常量定义在配置文件中的，这个常量可以用作`require()`的参数。这样做的好处是网站中的任何文件无论存储在何处，即使在子目录中，都可以使用相同的代码来包含连接脚本。

(4) 整理传入的数据并设置一些标志变量。

```
$trimmed = array_map('trim', $_POST);
$fn = $ln = $e = $p = FALSE;
```

第一行使用`trim()`函数处理`$_POST`中的每个元素，并把返回的结果赋予新的`$trimmed`数组。可以在第13章中找到对这一行的解释，当时把`array_map()`用于将要在电子邮件中发送的数据。简而言之，将把`trim()`函数应用于`$_POST`中的每个值，从而免除了单独把`trim()`应用于每个值的麻烦。

第二行把4个变量初始化为`FALSE`。这一行只是用于代替以下代码的快捷方式：

```
$fn = FALSE;
$ln = FALSE;
$e = FALSE;
$p = FALSE;
```

(5) 验证名字和姓氏。

```
if (preg_match ('/^([A-Z \'.-]{2,20})$/i', $trimmed['first_name'])) {
 $fn = mysqli_real_escape_string ($dbc, $trimmed['first_name']);
} else {
 echo '<p class="error">Please enter your first name!</p>';
}
if (preg_match ('/^([A-Z \'.-]{2,40})$/i', $trimmed['last_name'])) {
 $ln = mysqli_real_escape_string ($dbc, $trimmed['last_name']);
} else {
 echo '<p class="error">Please enter your last name!</p>';
}
```

将使用第14章中讨论的正则表达式验证表单。对于名字值，假定它只包含字母、句点（作为名字的首字符）、撇号、空格和短划线。进一步讲，期待这个值的长度在2~20个字符的范围内。为了保证这个值只包含这些字符，将使用插入符号和美元符号来匹配字符串的开头和末尾。在使用Perl兼容的正则表达式时，必须把整个模式置于定界符（正斜杠）内。

如果满足这个条件，就会把所提交值的`mysqli_real_escape_string()`版本的值赋予`$fn`变量；否则，`$fn`仍将为假，并且会打印一条出错消息（参见图18-12）。

将使用相同的过程来验证姓氏，尽管其正则表达式允许更长的长度。由于*i*修饰符，两种模式也都是不区分大小写的。

The screenshot shows a registration form titled "User Registration". It has two error messages: "Please enter your first name!" and "Please try again.". Below the messages are two input fields: "First Name:" containing "<" and "Last Name:" containing "Ullman".

图18-12 如果名字值没有通过正则表达式测试，就会打印一条出错消息

(6) 验证电子邮件地址（参见图18-13）。

```
if (filter_var($trimmed['email'], FILTER_VALIDATE_EMAIL)) {
 $e = mysqli_real_escape_string ($dbc, $trimmed['email']);
} else {
 echo '<p class="error">Please enter a valid email address! </p>';
}
```

The screenshot shows a registration form titled "User Registration". It has two error messages: "Please enter a valid email address!" and "Please try again.". Below the messages are three input fields: "First Name:" containing "Larry", "Last Name:" containing "Ullman", and "Email Address:" containing "Larry".

图18-13 提交的电子邮件地址必须具有正确的格式

第13章中描述了使用过滤器扩展验证电子邮件地址的方式。如果你的PHP脚本不支持过滤器扩展，就需要使用正则表达式（第14章介绍了电子邮件模式）

(7) 验证密码。

```
if (preg_match ('/^\\w{4,20}$/', $trimmed['password1'])) {
 if ($trimmed['password1'] == $trimmed['password2']) {
 $p = mysqli_real_escape_string ($dbc, $trimmed ['password1']);
 } else {
 echo '<p class="error">Your password did not match the confirmed password!</p>';
 }
} else {
 echo '<p class="error">Please enter a valid password!</p>';
}
```

密码的长度必须在4~20个字符之间，并且只包含字母、数字和下划线（参见图18-14）。在Perl兼容的正则表达式中，通过\w来表示准确的组合。此外，第一个密码（*password1*）必须与确认密码（*password2*）匹配（参见图18-15）。

The screenshot shows a "User Registration" page with a "Register" button. The form fields are: First Name: Larry, Last Name: Ullman, Email Address: Larry@LarryUllman.com, Password: •• (with a note below), and Confirm Password: (empty). An error message at the top says: "Please enter a valid password! Please try again."

图18-14 将检查密码是否具有正确的格式、长度，以及……

The screenshot shows a "User Registration" page with a "Register" button. The form fields are: First Name: Larry, Last Name: Ullman, Email Address: Larry@LarryUllman.com, Password: •••• (with a note below), and Confirm Password: •••••••. An error message at the top says: "Your password did not match the confirmed password! Please try again."

图18-15 .....这个密码值与确认密码值匹配

(8) 如果通过了每个测试，即可检查唯一的电子邮件地址。

```
if ($fn && $ln && $e && $p) {
 $q = "SELECT user_id FROM users WHERE email='$e'";
 $r = mysqli_query($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
}
```

如果表单通过了每个测试，这个条件语句将为TRUE。然后脚本必须查找数据库，以查看当前是否

在使用提交的电子邮件地址，因为那一列的值跨所有记录必须是唯一的。与MySQL连接脚本一样，如果查询没有运行，将使用trigger\_error()函数来调用自己定义的错误报告函数。特定的出错消息将包括正在运行的查询以及MySQL错误（参见图18-16），使得可以轻松地调试问题。

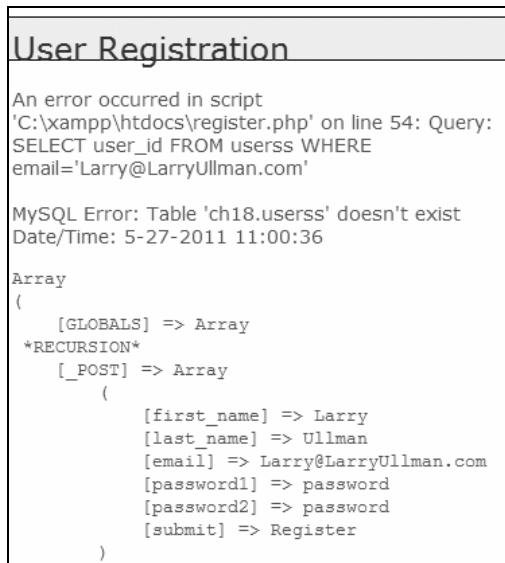


图18-16 如果发生MySQL查询错误，那么由于这个信息丰富的出错消息，应该更容易进行调试

(9) 如果电子邮件地址未被使用，则注册用户。

```

if (mysqli_num_rows($r) == 0) {
 $a = md5(uniqid(rand(), true));
 $q = "INSERT INTO users (email, pass, first_name, last_name, active, registration_date) VALUES ('$e',
 SHA1('$p'), '$fn', '$ln', '$a', NOW())";
 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
}

```

查询本身相当简单，但它需要创建一个激活码。可以使用rand()、uniqid()和md5()函数生成它。其中，uniqid()是最重要的，它创建一个唯一的标识符。把它赋予rand()函数，帮助生成更随机的值。最后，使用md5()散列化返回的结果，该函数将创建一个长度正好为32个字符的字符串（散列是一份数据的数学计算的表示）。你不必完全理解这三个函数，只需注意结果将是唯一的32个字符的字符串。

对于这个查询本身，你应该足够熟悉它。大多数值都来自于PHP脚本中的变量，并且是在对它们应用了trim()和mysqli\_real\_escape\_string()之后。MySQL的SHA1()函数用于加密密码，并且使用NOW()将注册日期设置为当前时刻。由于user\_level列的默认值为0（即不是管理员），所以不必在这个查询中给它提供一个值。很可能是站点的管理员编辑用户的记录，并授予他们管理权限。

(10) 如果查询工作，就发送一封电子邮件。

```

if (mysqli_affected_rows($dbc) == 1) {
 $body = "Thank you for registering at <whatever site>. To activate your account, please click on this
link:\n\n";
}

```

```
$body .= BASE_URL . 'activate.php?x=' . urlencode($e) . "&y=$a";
mail($trimmed['email'], 'Registration Confirmation', $body, 'From: admin@sitename.com');
```

利用这个注册过程，重要的事情是把确认电子邮件发送给用户，因为直到他们激活自己的账户之后，才能够登录。这封电子邮件应该包含一个指向激活页面（`activate.php`）的链接。指向该页面的链接开始于`BASE_URL`，它是在`config.inc.php`中定义的。该链接还会把两个值一起传入URL中。第一个值（一般称为`x`）将是用户的电子邮件地址，对它进行编码以使得它在URL中是安全的。第二个值（`y`）是激活码。这样，URL将类似于：`http://www.example.com/activate.php?x=email%40example.com&y=901e09ef25bf6e3ef95c930884-50b008`。

在成功注册后，将会打印出一条道谢消息，以及激活指导（参见图18-17）。

(11) 告诉用户注册成功并且完成页面。

```
echo '<h3>Thank you for registering! A confirmation email has been sent to your address. Please click
on the link in that email in order to activate your account.</h3>';
include ('includes/footer.html');
exit();
```

如果注册成功，将会打印一则感谢信息，并显示激活介绍（参见图18-17）。然后包含页脚文件，这样页面就完成了。

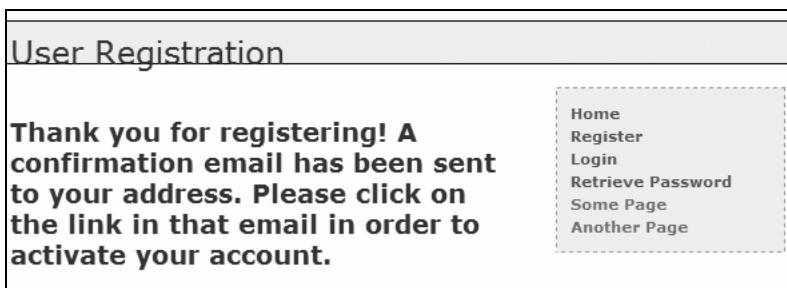


图18-17 在用户成功注册后所得到的页面

(12) 如果查询失败，则打印错误。

```
} else { // If it did not run OK.
echo '<p class="error">You could not be registered due to a system error. We apologize for any inconvenience.</p>';
}
```

如果查询由于某个原因失败，这意味着`mysqli_affected_rows()`没有返回1，就会把一条出错消息打印到浏览器。由于这个脚本中实现的安全性方法，站点的活动版本在这个节骨眼上永远都不应该发生问题。

(13) 完成条件语句和PHP代码。

```
} else {
echo '<p class="error">That email address has already been registered. If you have forgotten your
password, use the link at right to have your password sent to you. </p>';
}
} else {
echo '<p class="error">Please try again.</p>';
}
```

```

 mysqli_close($dbc);
}
?>

```

如果某个人试图用一个已经使用过的电子邮件注册，那么就会执行第一个else子句（参见图18-18）。当提交的数据无法通过验证例程之一的测试时，就会应用第二个else子句（参见图18-12至图18-15）。

(14) 开始创建HTML表单（参见图18-19）。

```

<h1>Register</h1>
<form action="register.php" method="post">
 <fieldset>
 <p>First Name: <input type="text" name="first_name" size="20" maxlength="20" value=<?php if
 (isset ($trimmed['first_name'])) echo $trimmed['first_name'];?>" /></p>

```

图18-18 如果某个电子邮件地址已经被注册，就会如实告诉用户

图18-19 用户第一次到达时所显示的注册表单

该HTML表单具有用于所有值的文本输入框。每个输入框都有一个名称和最大长度，它对应于users表中的列定义。表单将是黏性的，从而使用整理过的值。

(15) 为姓氏和电子邮箱地址添加文本输入框。

```

<p>Last Name: <input type="text" name="last_name" size="20" maxlength="40" value=<?php if (isset
 ($trimmed ['last_name'])) echo $trimmed['last_name']; ?>" /></p>
<p>Email Address: <input type="text" name="email" size="30" maxlength="60" value=<?php if (isset
 ($trimmed ['email'])) echo $trimmed['email']; ?>" /> </p>

```

(16) 为密码和确认密码添加文本输入框。

```
<p>Password: <input type="password" name="password1" size="20" maxlength="20" value=<?php
if (isset ($trimmed ['password1'])) echo $trimmed ['password1']; ?>" /> <small>Use only letters,
numbers, and the underscore. Must be between 4 and 20 characters long.</small></p>
<p>Confirm Password: <input type="password" name="password2" size="20" maxlength="20" value=<?php
if (isset($trimmed['password2'])) echo $trimmed['password2']; ?>" /></p>
```

当使用正则表达式限制可以提供的数据且包含数据的长度时，最好在表单中向用户指明长度要求。这样网站就不用再去向用户报告哪些不能做的错误消息了。

(17) 完成HTML表单。

```
</fieldset>
<div align="center"><input type="submit" name="submit" value="Register" /></div>
</form>
```

(18) 包含HTML脚注。

```
<?php include ('includes/footer. html'); ?>
```

(19) 将文件另存为register.php，存放在Web目录中，并在Web浏览器中测试它。

#### ✓ 提示

- 由于users表中的每一列都不能是NULL ( active列除外)，需要正确地填写每个输入框。如果表具有一个可选的字段，则在提交时，仍然应该确认它是否具有正确的类型，但这不是必需的。
- 除了加密的字段（如密码）之外，表单输入框的最大长度和正则表达式应该对应于数据库中列的最大长度。

## 18.5 激活账户

如本章前面的框注“激活过程”中所描述的那样，用户在能够登录前，必须激活他们的账户。在成功注册时，用户将接收到一封电子邮件，其中包含一个指向activate.php的链接（参见图18-20）。该链接还会把两个值传递给这个页面：他们的电子邮件地址和唯一的激活码。为了完成注册过程——激活账户，用户需要点击该链接，转到activate.php脚本所在的网站。

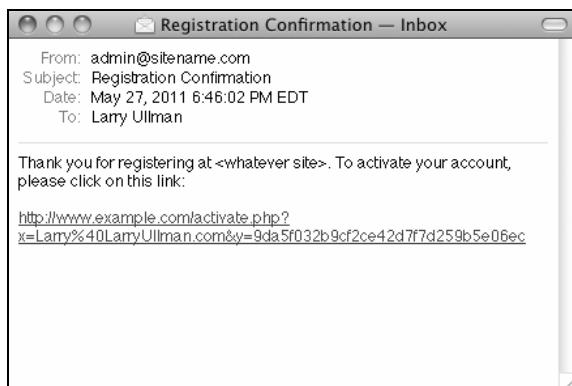


图18-20 注册确认电子邮件

这个脚本首先需要确认在URL中接收到这两个值。然后，如果这两个值与数据库中的那些值匹配，就会从记录中删除激活码，指示一个活动的账户。

### 创建激活页面

(1) 在文本编辑器或IDE中开始创建一个新的PHP脚本，命名为activate.php（参见脚本18-7）。

**脚本 18-7** 要激活一个账户，用户必须达到这个页面，并把他们的电子邮件地址和激活码（他们在注册时接收到的链接的所有部分）传递给它

```

1 <?php # Script 18.7 - activate.php
2 // This page activates the user's account.
3 require ('includes/config.inc.php');
4 $page_title = 'Activate Your Account';
5 include ('includes/header.html');
6
7 // If $x and $y don't exist or aren't of the proper format, redirect the user:
8 if (isset($_GET['x'], $_GET['y']))
9 && filter_var($_GET['x'], FILTER_VALIDATE_EMAIL)
10 && (strlen($_GET['y']) == 32)
11) {
12
13 // Update the database...
14 require (MYSQL);
15 $q = "UPDATE users SET active=NULL WHERE (email='". mysqli_real_escape_string($dbc, $_GET['x']) .
16 "' AND active='". mysqli_real_escape_string($dbc, $_GET['y']) . "') LIMIT 1";
17 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error ($dbc));
18
19 // Print a customized message:
20 if (mysqli_affected_rows($dbc) == 1) {
21 echo "<h3>Your account is now active. You may now log in.</h3>";
22 } else {
23 echo '<p class="error">Your account could not be activated. Please re-check the link or contact
24 the system administrator.</p>';
25 }
26
27 mysqli_close($dbc);
28
29 } // End of main IF-ELSE.
30
31
32
33
34
35
36 include ('includes/footer.html');
37 ?>
```

(2) 验证应该被页面接收的值。

```

if (isset($_GET['x'], $_GET['y']))
&& filter_var($trimmed['email'], FILTER_VALIDATE_EMAIL)
&& (strlen($_GET['y']) == 32)
){
```

如前所述，当用户在注册确认邮件中点击链接时，有两个值会被传递到本页面：电子邮箱地址和激活码。在激活用户账户的语句使用它们之前，这两个值必须存在并通过验证。

第一步，确认这两个值已被设置。由于`isset()`可以同时检查多个变量是否存在，验证条件语句的第一部分是`isset($_GET['x'], $_GET['y'])`。

第二步，`$_GET['x']`必须是有效的电子邮件地址格式。这里可以使用与注册脚本中相同的代码（无论是过滤器扩展还是正则表达式）。

第三步，对于`y`（激活码），条件语句中的最后一个子句检查这个字符串的长度（它有多少个字符）正好是32。用于创建激活码的`md5()`函数总是返回一个32个字符长度的字符串。

### (3) 尝试激活用户的账户。

```
require (MYSQL);
$q = "UPDATE users SET active=NULL WHERE (email='". mysqli_real_escape_string($dbc, $_GET['x']). "'"
AND active='". mysqli_real_escape_string($dbc, $_GET['y']). "'") LIMIT 1";
$r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: ". mysqli_error($dbc));
```

如果所有3个条件（步骤(2)中）都是TRUE，将运行UPDATE查询。这个查询将active列设置为NULL，从用户记录中移出激活码。在查询中使用这些值之前，这两个值都已经通过`mysqli_real_escape_string()`做了额外的安全处理。

### (4) 报告查询执行成功。

```
if (mysqli_affected_rows($dbc) == 1) {
 echo "<h3>Your account is now active. You may now log in.</h3>";
} else {
 echo '<p class="error">Your account could not be activated. Please re-check the link or contact
 the system administrator.</p>';
}
```

如果查询影响了一行，那么用户的账户现在就是活动的，并且会显示一条消息来指示这一点（参见图18-21）。如果没有任何行受到影响，就会通知用户出现了问题（参见图18-22）。如果有人尝试伪造`x`和`y`的值，或者如果在沿着从电子邮件到Web浏览器的链接行进时出现问题，则最有可能发生这种情况。



图18-21 如果可以使用提供的电子邮件地址和激活码更新数据库，就会通知用户他们的账户现在是活动的

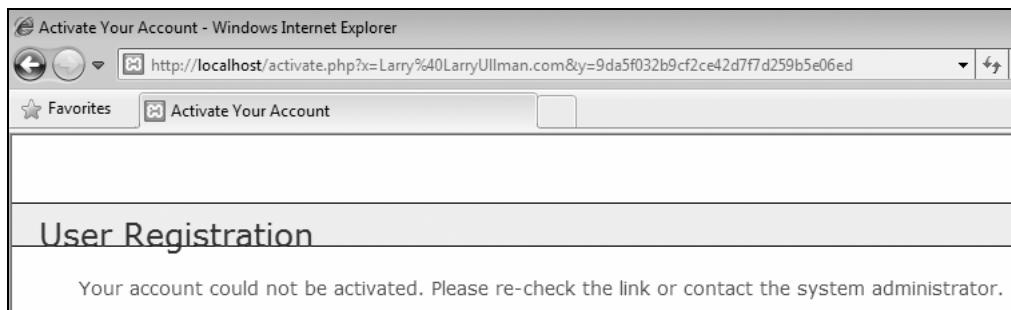


图18-22 如果查询没有激活账户，就会告知用户出现了问题

(5) 完成主条件语句。

```
mysqli_close($dbc);
} else { // Redirect.
$url = BASE_URL . 'index.php';
ob_end_clean();
header("Location: $url");
exit();
} // End of main IF-ELSE.
```

如果\$\_GET['x']和\$\_GET['y']的值和长度不合适，else子句将会生效。在这种情况下，用户只是被跳转到主页。这里的ob\_end\_clean()行删除了缓冲区（此时将要发送到浏览器中的任何内容都存储在内存中），因为它不会被用到。

(6) 完成页面。

```
include ('includes/footer.html');
?>
```

(7) 将文件另存为activate.php，存放在Web目录中，并通过单击注册电子邮件中的链接来测试它。

#### ✓ 提示

- 如果你想更仁慈一点，可以使这个页面打印一个类似于图18-22中所示的错误，而不是把他们重定向到索引页面（好像他们正在尝试破坏站点）。
- 出于安全考虑，特地把模糊的x和y用作URL中的名称。虽然有人可能会查明其中一个是电子邮件地址，另一个是激活码，但是，有时最好模糊地表现它们。
- 我在本书的第二版中使用了一种替代方法，它把激活码和用户的ID（来自于数据库）放在链接中。这也会工作，但是从安全性的角度讲，最好使用户永远都不会看到或者甚至不会知道用户ID，它不打算公开的。

## 18.6 登录和注销

在第12章中，我使用cookie和会话的变体，编写了login.php和logout.php脚本的许多版本。在这里，将开发这两个脚本的标准化版本，它们遵守与整个应用程序相同的惯例。这里的登录查询本身稍有不同，这是由于它还会检查active列具有NULL值，这意味着用户激活了他们的账户。

## 1. 编写login.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为login.php（参见脚本18-8）。

**脚本 18-8** 在会话中注册用户 ID、名字和访问级别之后，登录页面将把用户重定向到主页

```
1 <?php # Script 18.8 - login.php
2 // This is the login page for the site.
3 require ('includes/config.inc.php');
4 $page_title = 'Login';
5 include ('includes/header.html');
6
7 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
8 require (MYSQL);
9
10 // Validate the email address:
11 if (!empty($_POST['email'])) {
12 $e = mysqli_real_escape_string ($dbc, $_POST['email']);
13 } else {
14 $e = FALSE;
15 echo '<p class="error">You forgot to enter your email address!</p>';
16 }
17
18 // Validate the password:
19 if (!empty($_POST['pass'])) {
20 $p = mysqli_real_escape_string ($dbc, $_POST['pass']);
21 } else {
22 $p = FALSE;
23 echo '<p class="error">You forgot to enter your password!</p>';
24 }
25
26 if ($e && $p) { // If everything's OK.
27
28 // Query the database:
29 $q = "SELECT user_id, first_name, user_level FROM users WHERE (email='$e' AND pass=SHA1('$p')) AND active IS NULL";
30 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
31
32 if (@mysqli_num_rows($r) == 1) { // A match was made.
33
34 // Register the values:
35 $SESSION = mysqli_fetch_array ($r, MYSQLI_ASSOC);
36 mysqli_free_result($r);
37 mysqli_close($dbc);
38
39 // Redirect the user:
40 $url = BASE_URL . 'index.php'; // Define the URL.
41 ob_end_clean(); // Delete the buffer.
42 header("Location: $url");
43 exit(); // Quit the script.
44
45 } else { // No match was made.
46 echo '<p class="error">Either the email address and password entered do not match those
47 on file or you have not yet activated your account.</p>';
48 }
49 } else { // If everything wasn't OK.
50
51 }
52
53 // If there were errors, display them.
54 if ($e == FALSE) {
55 echo '<p class="error">You forgot to enter your email address!</p>';
56 }
57 if ($p == FALSE) {
58 echo '<p class="error">You forgot to enter your password!</p>';
59 }
60
61 // If there were no errors, display a success message.
62 if ($e && $p) {
63 echo '<p class="success">You have successfully logged in!</p>';
64 }
65
66 // Set session variables.
67 $_SESSION['user_id'] = $SESSION['user_id'];
68 $_SESSION['first_name'] = $SESSION['first_name'];
69 $_SESSION['user_level'] = $SESSION['user_level'];
70
71 // Redirect to the index page.
72 header("Location: index.php");
73 exit();
74
75}
```

```

50 echo '<p class="error">Please try again.</p>';
51 }
52
53 mysqli_close($dbc);
54
55 } // End of SUBMIT conditional.
56 ?>
57
58 <h1>Login</h1>
59 <p>Your browser must allow cookies in order to log in.</p>
60 <form action="login.php" method="post">
61 <fieldset>
62 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" /></p>
63 <p>Password: <input type="password" name="pass" size="20" maxlength="20" /></p>
64 <div align="center"><input type="submit" name="submit" value="Login" /></div>
65 </fieldset>
66 </form>
67
68 <?php include ('includes/footer.html'); ?>

```

(2) 检查是否提交了表单，需要数据库连接。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 require (MYSQL);

```

(3) 验证提交的数据。

```

if (!empty($_POST['email'])) {
 $e = mysqli_real_escape_string ($dbc, $_POST['email']);
} else {
 $e = FALSE;
 echo '<p class="error">You forgot to enter your email address!</p>';
}
if (!empty($_POST['pass'])) {
 $p = mysqli_real_escape_string ($dbc, $_POST['pass']);
} else {
 $p = FALSE;
 echo '<p class="error">You forgot to enter your password!</p>';
}

```

有两种方法来思考验证。一方面，你可以使用正则表达式和过滤器扩展，从register.php中复制相同的代码验证这些值；另一方面，对这些值真正测试的是登录查询是否返回一条记录，所以可以跳过更严格的PHP验证。这个脚本使用了后一个方法。

如果用户没有输入任何值到表单中，则会打印出错消息（参见图18-23）。

User Registration	
You forgot to enter your email address! You forgot to enter your password! Please try again.	
<b>Login</b>	

图18-23 登录表单只会检查是否输入了值，而无需使用正则表达式

(4) 如果通过了两个验证例程，就会检索用户信息。

```
if ($e && $p) {
 $q = "SELECT user_id, first_name, user_level FROM users WHERE (email='$e' AND pass=SHA1 ('$p')) AND
 active IS NULL";
 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
```

该查询将试图为其电子邮件地址和密码与那些提交的电子邮件地址和密码匹配的记录检索用户ID、名字和用户级别。MySQL查询在pass列上使用SHA1()函数，因为首先要使用该函数对密码进行加密。该查询还会检查active列具有NULL值，这意味着用户成功地访问了activate.php页面。

如果你知道一个账户已被激活，但是你仍然不能使用正确的值登录，那么很可能是因为active列被错误地定义为NOT NULL。

(5) 如果在数据库中产生一个匹配，就会登录进用户。

```
if (@mysqli_num_rows($r) == 1) {
 $_SESSION = mysqli_fetch_array ($r, MYSQLI_ASSOC);
 mysqli_free_result($r);
 mysqli_close($dbc);
```

登录过程包括：把检索的值存储在会话中（已经在header.html中启动了会话），然后把用户重定向到主页。由于查询将返回包含三个元素的数组——对user\_id建立的一个索引、对first\_name建立的一个索引和对user\_level建立的第三个索引，所以可以把它们正确地获取进\$\_SESSION中，得到\$\_SESSION['user\_id']、\$\_SESSION['first\_name']和\$\_SESSION['user\_level']。如果\$\_SESSION中已经具有其他的值，你将不希望采取这种快捷方式，因为你将会清除那些其他的元素。

(6) 重定向用户。

```
$url = BASE_URL . 'index.php'; ob_end_clean();
header("Location: $url");
exit();
```

ob\_end\_clean()函数将删除现有的缓冲区（输出缓冲也是在header.html中开启的），因为将不会使用它。

(7) 完成条件语句，并关闭数据库连接。

```
} else {
 echo '<p class="error">Either the email address and password entered do not match those on file
 or you have not yet activated your account. </p>';
}
} else {
 echo '<p class="error">Please try again.</p>';
}
mysqli_close($dbc);
} // End of SUBMIT conditional.
?>
```

出错消息（参见图18-24）指示登录过程可能由于两个可能的原因而失败。一个原因是，提交的电子邮件地址和密码与文件上的那些电子邮件地址和密码不匹配。另一个原因是，用户还没有激活他们的账户。

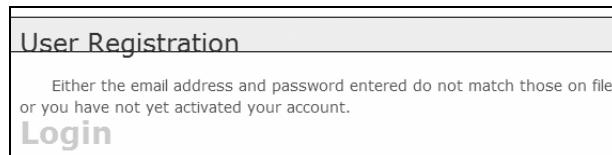


图18-24 如果登录查询没有返回一条记录，则会显示一条出错消息

(8) 显示HTML登录表单（参见图18-25）。

```
<h1>Login</h1>
<p>Your browser must allow cookies in order to log in.</p>
<form action="login.php" method="post">
<fieldset>
<p>Email Address: <input type="text" name="email" size="20" maxlength="60" /></p>
<p>Password: <input type="password" name="pass" size="20" maxlength="20" /></p>
<div align="center"><input type="submit" name="submit" value="Login" /></div>
</fieldset>
</form>
```

A screenshot of a login form titled 'Login'. It contains a message: 'Your browser must allow cookies in order to log in.' Two input fields labeled 'Email Address:' and 'Password:' with their respective placeholder text. A 'Login' button is at the bottom.

图18-25 登录表单

登录表单（像注册表单一样）将把数据提交回它自身。这个登录表单没有黏性，你可以自行添加这个功能。

注意：该页面包括一条消息，通知用户必须启用cookie来使用站点（如果用户不允许cookie，则永远也不能访问已登录用户页面）。

(9) 包含HTML脚注。

```
<?php include ('includes/footer.html'); ?>
```

(10) 将文件另存为login.php，存放在Web目录中，并在Web浏览器中测试它（参见图18-26）。



图18-26 在成功登录后，将把用户重定向到主页，在这里将用名字来显示问候语

## 2. 编写logout.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为logout.php（参见脚本18-9）。

### 脚本 18-9 注销页面会销毁所有的会话信息，包括 cookie

```

1 <?php # Script 18.9 - logout.php
2 // This is the logout page for the site.
3 require ('includes/config.inc.php');
4 $page_title = 'Logout';
5 include ('includes/header.html');
6
7 // If no first_name session variable exists, redirect the user:
8 if (!isset($_SESSION['first_name'])) {
9
10 $url = BASE_URL . 'index.php'; // Define the URL.
11 ob_end_clean(); // Delete the buffer.
12 header("Location: $url");
13 exit(); // Quit the script.
14
15 } else { // Log out the user.
16
17 $_SESSION = array(); // Destroy the variables.
18 session_destroy(); // Destroy the session itself.
19 setcookie (session_name(), '', time()-3600); // Destroy the cookie.
20
21 }
22
23 // Print a customized message:
24 echo '<h3>You are now logged out.</h3>';
25
26 include ('includes/footer.html');
27 ?>

```

(2) 如果用户没有登录，则重定向他们。

```

if (!isset($_SESSION['first_name'])) {
 $url = BASE_URL . 'index.php';
 ob_end_clean();
 header("Location: $url");
 exit();
}

```

如果用户当前没有登录（通过检查\$\_SESSION['first\_name']变量确定），则把他们重定向到主页（因为尝试注销他们是没有任何意义的）。

(3) 如果用户当前已登录，则注销他们。

```

} else { // Log out the user.
 $_SESSION = array();
 session_destroy();
 setcookie (session_name(), '', time()-3600);
}

```

要注销用户，将重置会话值，销毁服务器上的会话数据，并删除会话cookie。第12章中首次使用并描述了这几行代码。cookie名称将是session\_name()函数返回的值。如果你决定以后更改会话名称，这段代码仍然是准确的。

(4) 打印一条注销消息，并完成PHP页面。

```
echo '<h3>You are now logged out.</h3>';
include ('includes/footer.html');
?>
```

(5) 将文件另存为logout.php，存放在Web目录中，并在Web浏览器中测试它（参见图18-27）。

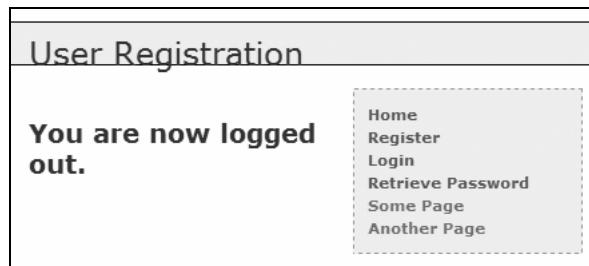


图18-27 成功注销的结果

## 18.7 密码管理

这个站点的公共端的最后一个方面是密码管理。要考虑两个过程：重置遗忘的密码和更改现有的密码。

### 18.7.1 重置密码

一种必然会发生的情况是，人们忘记了登录Web站点的密码，因此，具有针对这些事件的应急计划是重要的。一种选择是，当发生这种情况时，让用户给管理员发电子邮件，但是管理一个站点很困难，以至于管理员无暇分心于这种额外的麻烦。因此，我将建立一个脚本，其目的是重置遗忘的密码。

由于存储在数据库中的密码是使用MySQL的SHA1()函数进行加密的，所以无法检索未加密的版本。一种替代方法是：创建一个新的随机密码，并把现有的密码更改成这个值。这里并非只是在Web浏览器中显示新密码（那将极不安全），而是通过电子邮件将其发送到用户注册的地址。

编写forgot\_password.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为forgot\_password.php（参见脚本18-10）。

**脚本 18-10 forgot\_password.php 脚本允许用户重置他们的密码，而无需管理员的帮助**

```
1 <?php # Script 18.10 - forgot_password.php
2 // This page allows a user to reset their password, if forgotten.
3 require ('includes/config.inc.php');
4 $page_title = 'Forgot Your Password';
5 include ('includes/header.html');
6
7 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
8 require (MYSQL);
9
10 // Assume nothing:
```

```

11 $uid = FALSE;
12
13 // Validate the email address...
14 if (!empty($_POST['email'])) {
15
16 // Check for the existence of that email address...
17 $q = "SELECT user_id FROM users WHERE email='". mysqli_real_escape_string ($dbc, $_POST['email']) . "'";
18 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
19
20 if (mysqli_num_rows($r) == 1) { // Retrieve the user ID:
21 list($uid) = mysqli_fetch_array ($r, MYSQLI_NUM);
22 } else { // No database match made.
23 echo '<p class="error">The submitted email address does not match those on file!</p>';
24 }
25
26 } else { // No email!
27 echo '<p class="error">You forgot to enter your email address!</p>';
28 } // End of empty($_POST['email']) IF.
29
30 if ($uid) { // If everything's OK.
31
32 // Create a new, random password:
33 $p = substr (md5(uniqid(rand(), true)), 3, 10);
34
35 // Update the database:
36 $q = "UPDATE users SET pass=SHA1 ('$p') WHERE user_id=$uid LIMIT 1";
37 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
38
39 if (mysqli_affected_rows($dbc) == 1) { // If it ran OK.
40
41 // Send an email:
42 $body = "Your password to log into <whatever site> has been temporarily changed to '$p'.

43 Please log in using this password and this email address. Then you may change your password to

44 something more familiar.";
45 mail ($_POST['email'], 'Your temporary password.', $body, 'From: admin@sitename.com');
46
47 // Print a message and wrap up:
48 echo '<h3>Your password has been changed. You will receive the new, temporary password at

49 the email address with which you registered. Once you have logged in with this password,

50 you may change it by clicking on the "Change Password" link.</h3>';
51 mysqli_close($dbc);
52 include ('includes/footer.html');
53 exit(); // Stop the script.
54
55 } else { // If it did not run OK.
56 echo '<p class="error">Your password could not be changed due to a system error. We apologize

57 for any inconvenience.</p>';
58 }
59
60 mysqli_close($dbc);

```

```

60
61 } // End of the main Submit conditional.
62 ?>
63
64 <h1>Reset Your Password</h1>
65 <p>Enter your email address below and your password will be reset.</p>
66 <form action="forgot_password.php" method="post">
67 <fieldset>
68 <p>Email Address: <input type="text" name="email" size="20" maxLength="60" value="<?php if
69 (isset($_POST['email'])) echo $_POST['email']; ?>" /></p>
70 </fieldset>
71 <div align="center"><input type="submit" name="submit" value="Reset My Password" /></div>
72 </form>
73 <?php include ('includes/footer.html'); ?>

```

(2) 检查是否提交了表单，包含数据库连接，并创建标记变量。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 require (MYSQL);
 $uid = FALSE;

```

这个表单将会有一个电子邮箱地址文本框并且更改那条记录的登录密码。要实现那个功能，脚本首先需要获取匹配提交的电子邮箱地址的用户ID值。要开始那个过程，将标识变量赋值为FALSE，假设没有有效的用户ID。

(3) 验证提交的电子邮件地址。

```

if (!empty($_POST['email'])) {
 $q = 'SELECT user_id FROM users WHERE email="'. mysqli_real_escape_string ($dbc, $_POST['email']) . '"';
 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));

```

这是对提交的电子邮件地址的简单验证（没有使用正则表达式或过滤器扩展）。如果提交的值不为空，将会尝试在数据库中获取对应电子邮箱地址的用户ID。当然，你也可以添加更多严格的验证。

(4) 获取选中的用户ID。

```

if (mysqli_num_rows($r) == 1) {
 list($uid) = mysqli_fetch_array ($r, MYSQLI_NUM);
} else { // No database match made.
 echo '<p class="error">The submitted email address does not match those on file!</p>';
}

```

如果查询返回了一行，它将会被获取并赋给\$uid（用户ID的简写）。这个值将会在更新数据库的新密码时用到，并且它将同样用作一个标识变量。

`list()`函数在本书中没有正式讨论过，但是你可能偶尔会碰到它。它是一个快捷函数，允许你指定数组元素到其他变量。由于`mysqli_fetch_array()`将总是返回一个数组（即使仅有一个元素），因而使用`list()`可以省略如下代码：

```

$row = mysqli_fetch_array($r, MYSQLI_NUM);
$uid = $row[0];

```

如果提交的电子邮件地址没有找到匹配的记录，将会显示错误信息（参见图18-28）。

(5) 报告没有提供邮件地址错误。

```

} else { // No email!
 echo '<p class="error">You forgot to enter your email address!</p>';
} // End of empty($_POST['email']) IF.

```

如果没有提供电子邮件地址，也会显示出错消息（参见图18-29）。

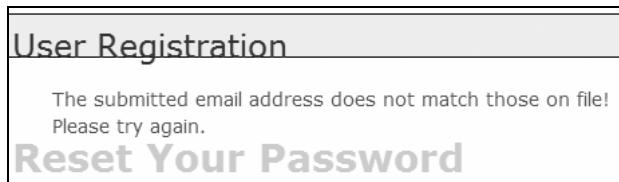


图18-28 如果在数据库中无法找到用户输入的电子邮件地址，就会显示一条出错消息

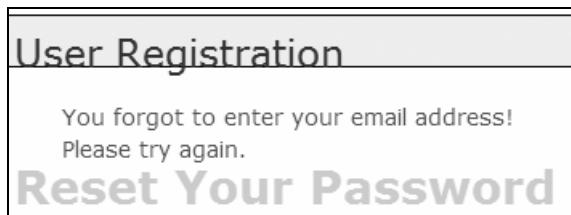


图18-29 没有提供电子邮件地址也会导致错误

(6) 创建新的随机密码。

```

if ($uid) {
 $p = substr (md5(uniqid(rand(), true)), 3, 10);

```

要创建新的随机密码，将利用4个PHP函数。第一个是`uniqid()`，它将返回一个唯一标识符。为它提供的参数是`rand()`和`true`，这使得返回的字符串更随机。然后通过`md5()`函数发送这个返回的值，该函数用于计算字符串的MD5散列。在这个阶段，将会返回唯一ID的散列版本，最后将得到一个长度为32个字符的字符串。代码的这一部分类似于`activate.php`（参见脚本18-7）中用于创建激活码的代码。

使用`substr()`函数，可以从这个字符串的第三个字符开始取出10个字符来确定密码。总而言之，这段代码将返回非常随机的、无意义的10个字符的字符串（包含字母和数字），以用作临时密码。

只有在\$Uid的值为TRUE时才需要创建新的、随机密码。

(7) 更新数据库中的密码。

```

$q = "UPDATE users SET pass=SHA1('$p') WHERE user_id=$uid LIMIT 1";
$r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
if (mysqli_affected_rows($dbc) == 1) {

```

使用早先检索的用户ID（表的主键），把这个特定用户的密码更新为\$p的`SHA1()`版本，即随机密码。

(8) 将密码发送给用户。

```

$body = "Your password to log into <whatever site> has been temporarily changed to '$p'. Please log
in using this password and this email address. Then you may change your password to something more
familiar.";
mail ($_POST['email'], 'Your temporary password.', $body, 'From: admin@sitename.com');

```

发送给用户的电子邮件（参见图18-30）包含新的、随机生成的密码在mail()代码中使用\$\_POST['email']是安全的，因为在执行这段代码之前，\$\_POST['email']必须匹配一个已经存储在数据库中的地址记录。那个地址在注册脚本中已经通过过滤器扩展（或正则表达式）验证过了。

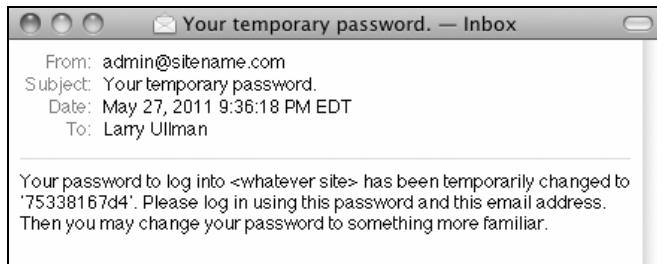


图18-30 在重置密码后接收到的电子邮件消息

(9) 完成页面。

```
echo '<h3>Your password has been changed. You will receive the new, temporary password at the email
address with which you registered. Once you have logged in with this password, you may change it by
clicking on the "Change Password" link.</h3>';
mysqli_close($dbc);
include ('includes/footer.html');
exit();
```

接下来，将会打印一条信息并且完成页面，以让表单不会再次显示（参见图18-31）。

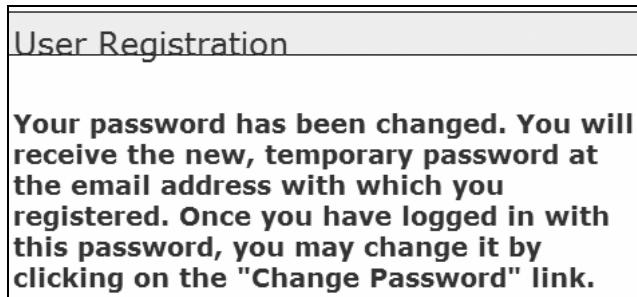


图18-31 在成功地重置密码后得到的页面

(10) 完成条件语句和PHP代码。

```
} else {
 echo '<p class="error">Your password could not be changed due to a system error. We apologize
 for any inconvenience.</p>';
}
} else {
 echo '<p class="error">Please try again.</p>';
}
mysqli_close($dbc);
} // End of the main Submit conditional.
?>
```

仅当UPDATE查询不工作时，才会应用第一个else子句（如果顺利的话，在活动站点上应该不会发生这种情况）。如果用户没有提交密码或者如果提交的密码与数据库中的任何密码都不匹配，就会应用第二个else子句。

(11) 建立HTML表单（参见图18-32）。

```
<h1>Reset Your Password</h1>
<p>Enter your email address below and your password will be reset.</p>
<form action="forgot_password.php" method="post">
 <fieldset>
 <p>Email Address: <input type="text" name="email" size="20" maxlength="60" value="<?php if
 (isset($_POST['email'])) echo $_POST['email']; ?>" /></p>
 </fieldset>
 <div align="center"><input type="submit" name="submit" value="Reset My Password" /></div>
</form>
```

图18-32 用于重置密码的简单表单

该表单只带有一个输入框，即电子邮件地址输入框。如果在提交表单时发生问题，将再次显示提交的电子邮件地址值（黏性表单）。

(12) 包含HTML脚注。

```
<?php include ('includes/footer.html'); ?>
```

(13) 将文件另存为forgot\_password.php，存放在Web目录中，并在Web浏览器中测试它。

(14) 检查电子邮件，查看成功地重置密码后得到的消息（参见图18-30）。

## 18.7.2 更改密码

change\_password.php脚本最初是在第9章中编写的（仅仅称为password.php），它作为UPDATE查询的一个示例。这里开发的一个脚本在功能上与其非常相似，但其区别是，只有登录的用户才能够访问它。因此，表单只需要接受新的密码及其确认（用户现有的密码和电子邮件地址已经通过登录页面进行了确认）。

**编写change\_password.php**

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为change\_password.php（参见脚本18-11）。

## 脚本 18-11 利用这个页面，用户可以更改现有的密码（如果他们已登录）

```

1 <?php # Script 18.11 - change_password.php
2 // This page allows a logged-in user to change their password.
3 require ('includes/config.inc.php');
4 $page_title = 'Change Your Password';
5 include ('includes/header.html');
6
7 // If no first_name session variable exists, redirect the user:
8 if (!isset($_SESSION['user_id'])) {
9
10 $url = BASE_URL . 'index.php'; // Define the URL.
11 ob_end_clean(); // Delete the buffer.
12 header("Location: $url");
13 exit(); // Quit the script.
14
15 }
16
17 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
18 require (MYSQL);
19
20 // Check for a new password and match against the confirmed password:
21 $p = FALSE;
22 if (preg_match ('/^(\w){4,20}$/', $_POST['password1'])) {
23 if ($_POST['password1'] == $_POST['password2']) {
24 $p = mysqli_real_escape_string ($dbc, $_POST['password1']);
25 } else {
26 echo '<p class="error">Your password did not match the confirmed password!</p>';
27 }
28 } else {
29 echo '<p class="error">Please enter a valid password!</p>';
30 }
31
32 if ($p) { // If everything's OK.
33
34 // Make the query:
35 $q = "UPDATE users SET pass=SHA1('$p') WHERE user_id={$_SESSION['user_id']} LIMIT 1";
36 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " . mysqli_error($dbc));
37 if (mysqli_affected_rows($dbc) == 1) { // If it ran OK.
38
39 // Send an email, if desired.
40 echo '<h3>Your password has been changed.</h3>';
41 mysqli_close($dbc); // Close the database connection.
42 include ('includes/footer.html'); // Include the HTML footer.
43 exit();
44
45 } else { // If it did not run OK.
46
47 echo '<p class="error">Your password was not changed. Make sure your new password is
different than the current password. Contact the system administrator if you think an error
occurred.</p>';
48
49 }
50

```

```

51 } else { // Failed the validation test.
52 echo '<p class="error">Please try again.</p>';
53 }
54
55 mysqli_close($dbc); // Close the database connection.
56
57 } // End of the main Submit conditional.
58 ?>
59
60 <h1>Change Your Password</h1>
61 <form action="change_password.php" method="post">
62 <fieldset>
63 <p>New Password: <input type="password" name="password1" size="20" maxlength="20" />
64 <small>Use only letters, numbers, and the underscore. Must be between 4 and 20 characters long.
65 </small></p>
66 <p>Confirm New Password: <input type="password" name="password2" size="20" maxlength=
67 "20" /></p>
68 </fieldset>
69 <div align="center"><input type="submit" name="submit" value="Change My Password" /></div>
70 </form>
71
72 <?php include ('includes/footer.html'); ?>
```

(2) 检查用户是否已登录。

```

if (!isset($_SESSION['user_id'])) {
 $url = BASE_URL . 'index.php';
 ob_end_clean();
 header("Location: $url");
 exit();
}
```

假定这个页面只能被登录用户访问。为了实施这种思想，该脚本将检查\$\_SESSION['first\_name']（UPDATE查询需要用到）变量是否存在。如果没有设置它，则会重定向用户。

(3) 检查是否提交了表单，并包含MySQL连接。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 require (MYSQL);
```

理解这个脚本如何工作的关键是记住有3种可能的情况：用户没有登录（因此会被重定向）、用户正在登录并且正在查看表单、用户已登录并且提交了表单。

如果用户已登录，他们将只会到达这一步。否则，将会重定向他们。因此，脚本现在需要确定是否提交了表单。

(4) 验证提交的密码。

```

$p = FALSE;
if (preg_match ('/^(\w){4,20}$/ ', $_POST['password1'])) {
 if ($_POST['password1'] == $_POST['password2']) {
 $p = mysqli_real_escape_string ($dbc,
 $_POST['password1']);
 } else {
 echo '<p class="error">Your password did not match the confirmed password!</p>';
 }
}
```

```

} else {
 echo '<p class="error">Please enter a valid password!</p>';
}

```

应该使用与注册过程中那些相同的测试来验证新密码。如果发现问题，则会显示出错消息（参见图18-33）。

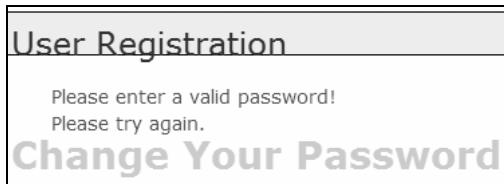


图18-33 与注册过程中一样，用户的新密码必须通过验证例程，否则他们将看到出错消息

(5) 更新数据库中的密码。

```

if ($p) {
 $q = "UPDATE users SET pass=SHA1('$p') WHERE user_id={$SESSION['user_id']} LIMIT 1";
 $r = mysqli_query ($dbc, $q) or trigger_error("Query: $q\n
MySQL Error: " .
 mysqli_error($dbc));
}

```

使用用户的ID（当用户登录时，它存储在会话中）可以更新数据库中的密码字段。LIMIT 1子句不是严格需要的，但它可以增加其他保险。

(6) 如果更新正常工作，完成页面。

```

if (mysqli_affected_rows($dbc) == 1) {
 echo '<h3>Your password has been changed.</h3>';
 mysqli_close($dbc);
 include ('includes/footer.html'); exit();
}

```

如果更新工作，就会把一条确认消息打印到Web浏览器（参见图18-34）。

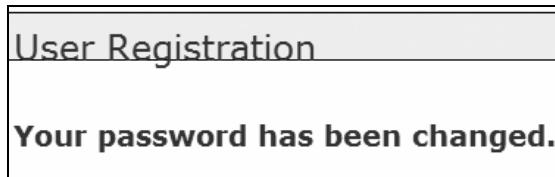


图18-34 脚本成功地更改了用户的密码

(7) 完成条件语句和PHP代码。

```

} else {
 echo '<p class="error">Your password was not changed. Make sure your new password is different
 than the current password. Contact the system administrator if you think an error occurred.</p>';
}
} else {
 echo '<p class="error">Please try again.</p>';
}

```

```

 mysqli_close($dbc);
} // End of the main Submit conditional.
?>

```

如果`mysqli_affected_rows()`函数没有返回值1，就会应用第一个`else`子句。可能由于两个原因而导致发生这种情况。第一个原因是，发生了查询或数据库错误。如果顺利的话，在已经解决了所有问题之后，不太可能在活动站点上发生这种情况。第二个原因是，用户尝试“更改”他们的密码，但是再次输入了相同的密码。在这种情况下，`UPDATE`查询将不会影响任何行，因为不会更改数据库中的密码列。程序将会打印一条消息来指示这一点。

(8) 创建HTML表单（参见图18-35）。

```

<h1>Change Your Password</h1>
<form action="change_password.php" method="post">
 <fieldset>
 <p>New Password: <input type="password" name="password1" size="20" maxlength="20" /> <small> Use
 only letters, numbers, and the underscore. Must be between 4 and 20 characters long.</small></p>
 <p>Confirm New Password: <input type="password" name="password2" size="20" maxlength="20" /></p>
 </fieldset>
 <div align="center"><input type="submit" name="submit" value="Change My Password" /></div>
</form>

```

图18-35 Change Your Password表单

这个表单带有两个输入框：新密码输入框及其确认输入框。另外，还给出了正确格式的说明。因为不能为HTML表单十分简单，所以没有添加黏性功能（你可以自己添加）。

(9) 完成HTML页面。

```
<?php include ('includes/footer.html'); ?>
```

(10) 将文件另存为`change_password.php`，存放在Web目录中，并在Web浏览器中测试它。

#### ✓ 提示

- ❑ 一旦完成了这个脚本，用户就可以利用前一个脚本重置他们的密码，然后使用临时、随机的密码登录。在登录后，用户可以利用这个页面把他们的密码更改回某个更难忘记的密码。
- ❑ 由于站点的身份验证不依赖于从一个页面到另一个页面的用户密码（换句话说，在登录后，不会对每个后续页面检查密码），所以更改密码将不需要用户登录。

### 站点管理

对于这个应用程序，站点管理的工作方式依赖于你想让它做什么。对于管理员，你可能想要的一个额外的页面是view\_users.php脚本，它与在第8章中创建并在第9章中修改的一个脚本一样。它已经列出在管理员的链接中。可以使用它链接到edit\_user.php页面，它将允许手动激活一个账户，声明某个用户是一位管理员，或者更改某个人的密码。还可以使用这个页面删除一个用户。

仅当登录的用户是管理员时页脚文件才会创建指向管理页面的链接，而每个管理页面还应该包括这样一个检查。

## 18.8 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

注意，这里的某些问题和提示涉及前面章节介绍的内容，目的是强化里面的重点知识。

### 18.8.1 回顾

- 输出缓冲是什么？使用它的好处是什么？
- 为什么不应该在线的网站显示详细错误信息？
- 为什么users表中的active列必须允许NULL值？如果active列被定义为不可为空会有什么后果？
- 终止会话的3个步骤是什么？
- session\_name()函数的作用是？
- 真正的加密数据和创建数据的散列之间的区别是？

### 18.8.2 实践

- 在PHP手册页面中查看输出缓冲（或输出控制）。
- 在PHP手册的页面查看rand()、uniqid()和md5()函数。
- 在PHP手册的页面查看trigger\_error()函数。
- 为login.php应用与register.php相同的验证技术。
- 让登录表单变为黏性表单。
- 为users表添加DATETIME类型的last\_login字段，并在用户登录时更新它。使用此信息来确定用户自上次访问站点后过了多少时间。
- 如果你已经添加了last\_login字段，使用它在首页打印一条信息，说明在过去的一小时或一天中有多少用户登录。
- 在forgot\_password.php中使用过滤器扩展或正则表达式验证所提交的电子邮件地址。
- 在PHP手册页面查看的list()函数。
- 按照最后一个框注“站点管理”的建议，创建view\_user.php和edit\_user.php脚本。限制只有管理员才能访问这些页面（这些用户的访问级别为1）。

# 示例——电子商务 19

### 本章内容

- 创建数据库
- 管理端
- 创建公共模板
- 产品目录
- 购物车
- 记录订单
- 回顾和实践

**在** 本书最后一章中，将开发最后一个Web应用程序——电子商务站点。在这个示例中，将设计一个站点，用于销售艺术印刷品。不幸的是，编写和解释整个应用程序实质上需要占用一本书的篇幅。此外，电子商务的某些方面（比如如何处理资金）对于各个站点都有极大的不同。尝试演示这样一个过程是浪费空间。在考虑到这些限制后，本章把重点放在电子商务站点的核心功能上：设计数据库、作为管理员填写目录、显示产品给公众、创建购物车以及在数据库中存储订单。

这个示例包括许多已经介绍过的概念：通过结合使用PHP与MySQL、处理文件上传、使用PHP发送图像到Web浏览器、预处理语句、会话等。本章还将介绍一个新概念：如何通过PHP脚本执行MySQL事务。为了在已经扩展的示例中节省空间，将削减一些细枝末节。不过，在这样做时，我将提供一些用于改进脚本的建议。

### 19.1 创建数据库

本示例中的电子商务将使用简单地命名为e-commerce的数据库。在MySQL中创建这个数据库之前，解释一下每个表的作用。

任何类型的电子商务应用程序都要存储3大类数据：产品信息（正在销售什么）、顾客信息（谁在购买产品），以及订单信息（购买的产品是什么以及购买者是谁）。在经过规范化过程后（见第6章），将得出5个表（参见图19-1）。

前两个表存储所有待售作品。如前所述，该站点将销售艺术印刷品。artists表（参见表19-1）存储待售作品的艺术家的信息。该表只包含最少量的信息（艺术家的名字、中名和姓氏），但是，可以轻松地添加艺术家的出生日期和死亡日期、传记材料等。prints表（参见表19-2）是该站点的主要作品表。

它存储印刷品名称、价格及其他相关信息。使用artist\_id将其链接到artists表。这个表无疑是最重要的，因为它为每一件待售作品都提供了唯一标识符。这个概念对于任何电子商务站点都是至关重要的（如果没有唯一标识符，怎么知道人们购买什么呢）。

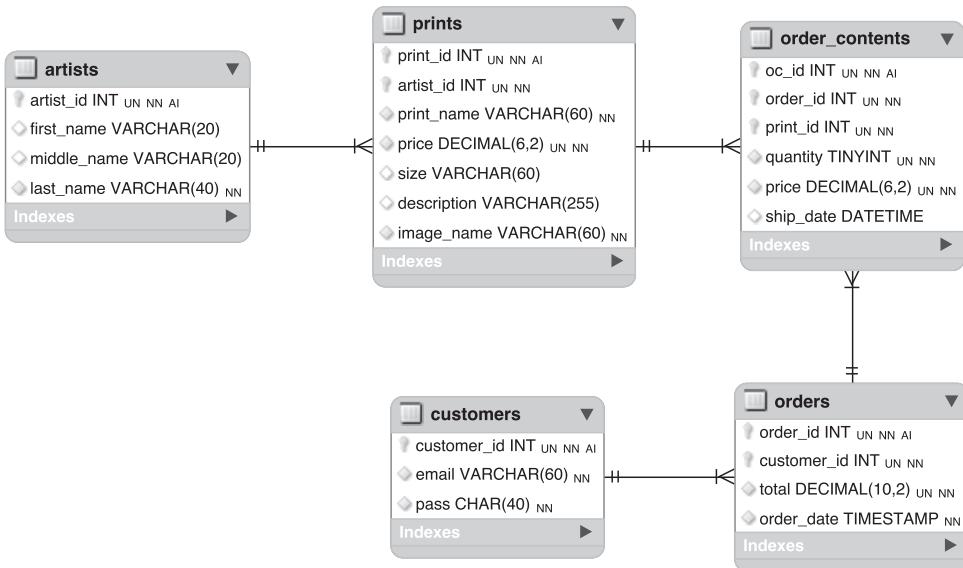


图19-1 这个实体-联系图（ERD）显示了ecommerce数据库中5个表之间的相互关系

表19-1 artists表

列	类 型
artist_id	INT UNSIGNED NOT NULL
first_name	VARCHAR(20) DEFAULT NULL
middle_name	VARCHAR(20) DEFAULT NULL
last_name	VARCHAR(40) NOT NULL

表19-2 prints表

列	类 型
print_id	INT UNSIGNED NOT NULL
artist_id	INT UNSIGNED NOT NULL
print_name	VARCHAR(60) NOT NULL
price	DECIMAL(6,2) UNSIGNED NOT NULL
size	VARCHAR(60) DEFAULT NULL
description	VARCHAR(255) DEFAULT NULL
image_name	VARCHAR(60) NOT NULL

customers表（参见表19-3）与你所想的完全一样：它记录了每位客户的个人信息。至少会反映顾客的名字、姓氏、电子邮件地址、密码、发货地址，以及他们的注册日期。假定可以使用电子邮件地址和密码的组合来允许用户登录、购物和访问他们的账户。因为这个表将存储哪些信息相当明显，目前只用3个必要的列定义它。

表19-3 customers表

列	类 型
customer_id	INT UNSIGNED NOT NULL
email	VARCHAR(60) NOT NULL
pass	CHAR(40) NOT NULL

最后两个表存储所有订单信息。可以采用许多方式执行这项任务，但是，我选择在orders表（参见表19-4）中存储一般的订单信息——总额、日期和顾客的ID。也可以往这个表中添加单独的列，用于反映运费、销售税额、应用的任何折扣，等等。order\_contents表（参见表19-5）将存储实际要销售的项目，包括数量和价格。order\_contents表实质上起到了中间人的作用，用于避免prints表和orders表之间的多对多关系（每件印刷品可以出现在多份订单中，每份订单可以包含多件印刷品）。

表19-4 orders表

列	类 型
order_id	INT UNSIGNED NOT NULL
customer_id	INT UNSIGNED NOT NULL
total	DECIMAL(10,2) UNSIGNED NOT NULL
order_date	TIMESTAMP

表19-5 order\_contents表

列	类 型
oc_id	INT UNSIGNED NOT NULL
order_id	INT UNSIGNED NOT NULL
print_id	INT UNSIGNED NOT NULL
quantity	TINYINT UNSIGNED NOT NULL DEFAULT 1
price	DECIMAL(6,2) UNSIGNED NOT NULL
ship_date	DATETIME DEFAULT NULL

为了能够使用事务（在最后一个脚本中），两张订单表将使用InnoDB存储引擎。其他的表将使用默认的MyISAM类型。有关可用的存储引擎（表类型）的详细信息，见第6章。

### 创建数据库

(1) 登录到MySQL客户端，如果没有ecommerce数据库，则创建它。

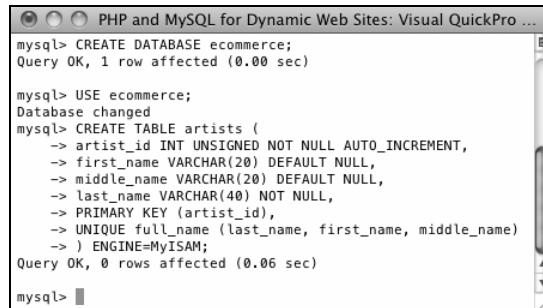
```
CREATE DATABASE ecommerce;
USE ecommerce;
```

对于这些类型，可以使用MySQL客户端，或者另一种工具，如phpMyAdmin。根据你的情况，可能还需要同时创建字符集，以便与数据库应用通信。你也可以在创建数据时创建默认的字符集和编码。(参见第6章)。

如果你正使用的是托管的站点，托管公司可能已经为你提供了数据库。

(2) 创建artists表(参见图19-2)。

```
CREATE TABLE artists (
 artist_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 first_name VARCHAR(20) DEFAULT NULL,
 middle_name VARCHAR(20) DEFAULT NULL,
 last_name VARCHAR(40) NOT NULL,
 PRIMARY KEY (artist_id),
 UNIQUE full_name (last_name, first_name, middle_name)
) ENGINE=MyISAM;
```



The screenshot shows a terminal window titled 'PHP and MySQL for Dynamic Web Sites: Visual QuickPro ...'. It displays the following MySQL session:

```
mysql> CREATE DATABASE ecommerce;
Query OK, 1 row affected (0.00 sec)

mysql> USE ecommerce;
Database changed

mysql> CREATE TABLE artists (
 -> artist_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 -> first_name VARCHAR(20) DEFAULT NULL,
 -> middle_name VARCHAR(20) DEFAULT NULL,
 -> last_name VARCHAR(40) NOT NULL,
 -> PRIMARY KEY (artist_id),
 -> UNIQUE full_name (last_name, first_name, middle_name)
 ->) ENGINE=MyISAM;
Query OK, 0 rows affected (0.06 sec)

mysql> ■
```

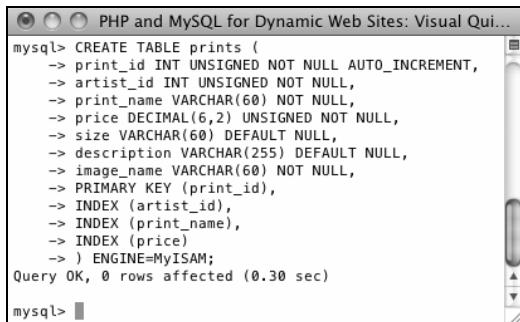
图19-2 制作第一个表

这个表只为每位艺术家存储4种信息，其中只有last\_name是必需的(它被定义为NOT NULL)，因为存在只有一个名字的艺术家(例如，Christo)。表还添加了索引。主键是artist\_id，并为名字和姓氏的组合建立了索引，可以在ORDER BY子句中使用它。这是一个明确定义的唯一索引，以确保不会添加同名的艺术家。

(3) 创建prints表(参见图19-3)。

```
CREATE TABLE prints (
 print_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 artist_id INT UNSIGNED NOT NULL,
 print_name VARCHAR(60) NOT NULL,
 price DECIMAL(6,2) UNSIGNED NOT NULL,
 size VARCHAR(60) DEFAULT NULL,
 description VARCHAR(255) DEFAULT NULL,
 image_name VARCHAR(60) NOT NULL,
 PRIMARY KEY (print_id),
 INDEX (artist_id),
 INDEX (print_name),
 INDEX (price)
) ENGINE=MyISAM;
```

除了size和description这两列之外，prints表中的所有列都是必需的。我还在artist\_id、print\_name和price这几个字段上设置了索引，其中每个索引都可以用在查询中。



```
mysql> CREATE TABLE prints (
-> print_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> artist_id INT UNSIGNED NOT NULL,
-> print_name VARCHAR(60) NOT NULL,
-> price DECIMAL(6,2) UNSIGNED NOT NULL,
-> size VARCHAR(60) DEFAULT NULL,
-> description VARCHAR(255) DEFAULT NULL,
-> image_name VARCHAR(60) NOT NULL,
-> PRIMARY KEY (print_id),
-> INDEX (artist_id),
-> INDEX (print_name),
-> INDEX (price)
->) ENGINE=MyISAM;
Query OK, 0 rows affected (0.30 sec)

mysql>
```

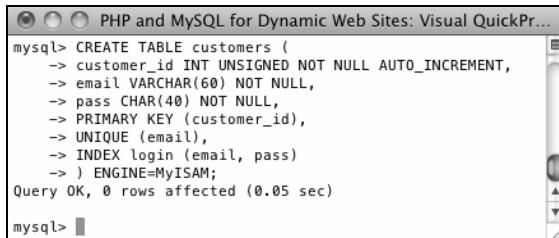
图19-3 制作第二个表

每件印刷品都将与一幅图像相关联。将使用与print\_id相同的值把该图像存储在服务器上。在Web浏览器中显示图像时，将使用其原始名称，因此需要把它存储在该表中。

可以往该表中添加一个in\_stock或qty\_on\_hand字段，以指示产品的可用性。

(4) 创建customers表（参见图19-4）。

```
CREATE TABLE customers (
customer_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
email VARCHAR(60) NOT NULL,
pass CHAR(40) NOT NULL,
PRIMARY KEY (customer_id),
UNIQUE (email),
INDEX login (email, pass)
) ENGINE=MyISAM;
```



```
mysql> CREATE TABLE customers (
-> customer_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> email VARCHAR(60) NOT NULL,
-> pass CHAR(40) NOT NULL,
-> PRIMARY KEY (customer_id),
-> UNIQUE (email),
-> INDEX login (email, pass)
->) ENGINE=MyISAM;
Query OK, 0 rows affected (0.05 sec)

mysql>
```

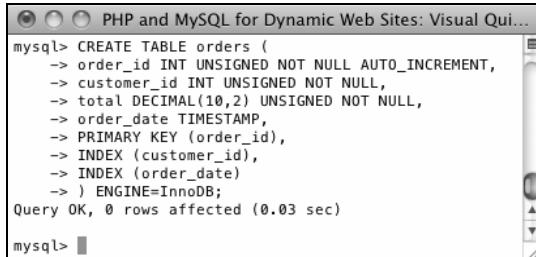
图19-4 创建customers表的基本版本。在真实的电子商务站点中，需要扩展这个表以存储更多信息

这段代码用于创建customers表。可以加入其他合适的字段（名字、地址、电话号码、注册日期等）。因为本章中不会涉及这些值，或者说根本没有涉及用户管理，所以省略了它们。

(5) 创建orders表（参见图19-5）。

```
CREATE TABLE orders (
order_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
customer_id INT UNSIGNED NOT NULL,
total DECIMAL(10,2) UNSIGNED NOT NULL,
order_date TIMESTAMP,
PRIMARY KEY (order_id),
```

```
INDEX (customer_id),
INDEX (order_date)
) ENGINE=InnoDB;
```



```
mysql> CREATE TABLE orders (
 -> order_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 -> customer_id INT UNSIGNED NOT NULL,
 -> total DECIMAL(10,2) UNSIGNED NOT NULL,
 -> order_date TIMESTAMP,
 -> PRIMARY KEY (order_id),
 -> INDEX (customer_id),
 -> INDEX (order_date)
 ->) ENGINE=InnoDB;
Query OK, 0 rows affected (0.03 sec)

mysql> █
```

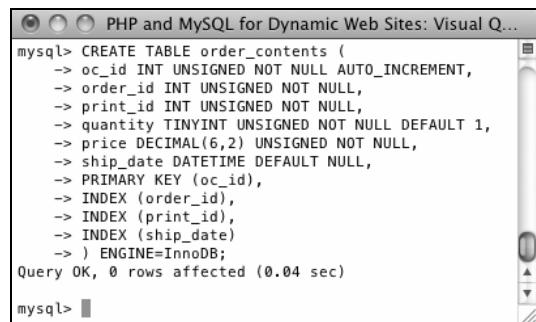
图19-5 制作orders表

orders表的所有字段都是必需的，并且创建了3个索引。注意，这里的外键列（如customer\_id）与其对应的主键（customers表中的customer\_id）具有完全相同的类型。order\_date字段将存储输入订单的日期和时间。通过将其定义为TIMESTAMP类型，当插入记录时，会自动赋予它当前值（由于这个原因，不必正式地把它定义为NOT NULL）。

最后，因为希望在orders和order\_contents这两个表上使用事务，所以它们都将使用InnoDB存储引擎。

#### (6) 创建order\_contents表（参见图19-6）。

```
CREATE TABLE order_contents (
oc_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
order_id INT UNSIGNED NOT NULL,
print_id INT UNSIGNED NOT NULL,
quantity TINYINT UNSIGNED NOT NULL DEFAULT 1,
price DECIMAL(6,2) UNSIGNED NOT NULL,
ship_date DATETIME DEFAULT NULL,
PRIMARY KEY (oc_id),
INDEX (order_id),
INDEX (print_id),
INDEX (ship_date)
) ENGINE=InnoDB;
```



```
mysql> CREATE TABLE order_contents (
 -> oc_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 -> order_id INT UNSIGNED NOT NULL,
 -> print_id INT UNSIGNED NOT NULL,
 -> quantity TINYINT UNSIGNED NOT NULL DEFAULT 1,
 -> price DECIMAL(6,2) UNSIGNED NOT NULL,
 -> ship_date DATETIME DEFAULT NULL,
 -> PRIMARY KEY (oc_id),
 -> INDEX (order_id),
 -> INDEX (print_id),
 -> INDEX (ship_date)
 ->) ENGINE=InnoDB;
Query OK, 0 rows affected (0.04 sec)

mysql> █
```

图19-6 为ecommerce数据库制作最后一个表

为了具有一种规范化的数据库结构，我把每份订单分成通用信息（顾客、订单日期和总额）以及它的特定信息——订购的实际项目及其数量。这个表具有指向orders和prints表的外键。quantity具有一组默认值1。ship\_date被定义为一个DATETIME，因此，它可以具有一个NULL值，指示作品还没有运送。同样，这个表必须使用InnoDB存储引擎，以便成为事务的一部分。

当价格信息已经存在于prints表中时，你可能对为什么还在这个表中存储价格感到好奇。原因很简单，作品的价格可能发生变化。prints表指示作品的当前价格，order\_contents表指示购买一个作品的价格。

#### ✓ 提示

- 根据站点正在销售什么，将用不同的表代替artists和prints。任何电子商务数据库最重要的属性是有一个产品表，其中列出了待售的各个产品，以及与之关联的产品ID。因此，一件大号红色球衣将具有一个ID，它不同于一件大号蓝色球衣的ID。唯一的单个产品标识符允许跟踪订单和产品数量。
- 如果想存储用户的多个地址（家庭、账单、朋友等），可单独创建一个地址表。在这个表中存储所有这方面的信息（包括地址类型），并通过把顾客ID用作主键—外键，将这些记录链接回customers表。

### 安 全 性

就一个电子商务站点来说，有4个主要的安全考虑事项。第一个是如何把数据存储在服务器上。你需要保护MySQL数据库本身（通过设置合适的访问权限）以及存储会话信息的目录（见第11章，了解必须更改什么设置）。对于这些问题，使用非共享的托管肯定可以改进站点的安全性。

第二个是必须处理如何保护对敏感信息的访问。站点的管理端（它能够查看订单和顾客记录）必须得到最高级别的保护。这意味着在访问它时需要进行身份验证，从而可以限制谁知道访问信息、使用安全连接，等等。

第三个是在转移期间保护数据。到那时，顾客将开始结款过程（其中会涉及信用卡和送货信息），必须使用安全的事务。为了执行该任务，必须用有效的证书在服务器上建立一个安全套接字层（Secure Sockets Layer，SSL），然后将其变成一个`https://` URL。还要知道正通过电子邮件发送什么信息，因为这些消息往往不是通过安全途径传输的。

第四个是必须处理付款信息。你实在不希望以任何方式保存该信息。理想情况下，让第三方资源处理付款，并使站点保持清洁。我将在本章末尾的框注“结款过程”中对此作一点讨论。

有关电子商务站点的更广泛主题，请看我编写的另一本书*Effortless E-Commerce with PHP and MySQL*，里面涵盖了大量技术细节和实用代码。

## 19.2 管理端

电子商务应用的管理端主要是管理以下3个主要组成部分：

- 产品（要卖的商品）；

- 客户；
- 订单。

在本章，你将创建两个脚本来向目录中添加产品。因为要卖的产品是艺术品，一个脚本将会是流行的艺术家（artists）表，而第二个将会是流行的印刷品（prints）表（它有一个到artists表的外键）。

由于篇幅限制，实际上没有实现支付系统，因而这不会是完整的电子商务示例，客户和订单方面的内容本章也不会详细说明。客户管理正好与用户管理相同，在前面的章节中已有所涉及。本章将会给出很多订单管理的建议。

开始的两个脚本（几乎本章所有脚本），将会需要一个到MySQL数据库的连接。不必重新写一个新的脚本，直接复制第9章的`mysqli_connect.php`（脚本9-2）到这个站点文件的恰当目录即可。然后修改下信息，以便连接到`ecommerce`的数据库，使用有适当权限的用户名/密码/主机名的组合。

同样，为了给你提供更多的使用经验，这两个管理脚本将会使用第13章中介绍的预处理语句。如果你对预处理语句的预防或函数感到困惑，复习第13章。

### 19.2.1 添加艺术家

第一个脚本向`artists`表添加艺术家记录。这个脚本显示了一个有3个输入框的表单（参见图19-7），在提交时验证输入框内容（有两个是可选的），并向数据库中添加记录。

The screenshot shows a window titled "Add an Artist" with a sub-section titled "Add a Print". Inside, there's a form with three text input fields. The first field is labeled "First Name" with the value "Sandro". The second field is labeled "Middle Name" with no value. The third field is labeled "Last Name" with the value "Botticelli". At the bottom of the form is a "Submit" button.

图19-7 将添加印刷品到目录中的HTML表单

#### 创建add\_artist.php

(1) 创建新的PHP文档，以HTML头开始，将其命名为`add_artist.php`（参见脚本19-1）。

通常来说，站点的管理端通常会使用模板系统（可能是与公共端不同的模板），但是本章只是用两个管理脚本，因此没必要开发模板。

#### 脚本 19-1 这个管理页面向数据库中添加新的艺术家名字

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6 <title>Add an Artist</title>
7 </head>
8 <body>
9 <?php # Script 19.1 - add_artist.php
10 // This page allows the administrator to add an artist.
11
12 if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Handle the form.
13
14 // Validate the first and middle names (neither required):
15 $fn = (!empty($_POST['first_name'])) ? trim($_POST['first_name']) : NULL;
16 $mn = (!empty($_POST['middle_name'])) ? trim($_POST['middle_name']) : NULL;
17
18 // Check for a last_name...
19 if (!empty($_POST['last_name'])) {
20
21 $ln = trim($_POST['last_name']);
22
23 // Add the artist to the database:
24 require ('../../mysqli_connect.php');
25 $q = 'INSERT INTO artists (first_name, middle_name, last_name) VALUES (?, ?, ?)';
26 $stmt = mysqli_prepare($dbc, $q);
27 mysqli_stmt_bind_param($stmt, 'sss', $fn, $mn, $ln);
28 mysqli_stmt_execute($stmt);
29
30 // Check the results....
31 if (mysqli_stmt_affected_rows($stmt) == 1) {
32 echo '<p>The artist has been added.</p>';
33 $_POST = array();
34 } else { // Error!
35 $error = 'The new artist could not be added to the database!';
36 }
37
38 // Close this prepared statement:
39 mysqli_stmt_close($stmt);
40 mysqli_close($dbc); // Close the database connection.
41
42 } else { // No last name value.
43 $error = 'Please enter the artist\'s name!';
44 }
45
46 } // End of the submission IF.
47
48 // Check for an error and print it:
49 if (isset($error)) {
50 echo '<h1>Error!</h1>
51 <p style="font-weight: bold; color: #C00">' . $error . ' Please try again.</p>';
52 }
53
54 // Display the form...
55 ?>
56 <h1>Add a Print</h1>
```

```

57 <form action="add_artist.php" method="post">
58
59 <fieldset><legend>Fill out the form to add an artist:</legend>
60
61 <p>First Name: <input type="text" name="first_name" size="10" maxlength="20" value="<?php
62 if (isset($_POST['first_name'])) echo $_POST['first_name']; ?>" /></p>
63 <p>Middle Name: <input type="text" name="middle_name" size="10" maxlength="20" value="<?
64 php if (isset($_POST['middle_name'])) echo $_POST['middle_name']; ?>" /></p>
65 <p>Last Name: <input type="text" name="last_name" size="10" maxlength="40" value="<?php if
66 (isset($_POST['last_name'])) echo $_POST['last_name']; ?>" /></p>
67
68 </fieldset>
69
70 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
71
72 </body>
73 </html>

```

(2) 检查表单是否已经被提交。

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
```

(3) 验证艺术家的名和中间名。

```
$fn = (!empty($_POST['first_name'])) ? trim($_POST['first_name']) : NULL;
$mn = (!empty($_POST['middle_name'])) ? trim($_POST['middle_name']) : NULL;
```

艺术家的名和中间名是可选的字段，但是姓不是（因为艺术家只通过名字被引用）。要验证签名的两个名字的输入框，这里使用了三目运算符来简化代码（在第10章中介绍过）。这里的代码类似于：

```
if (!empty($_POST['first_name'])) {
 $fn = trim($_POST['first_name']);
} else {
 $fn = NULL;
}
```

因为本脚本依赖于预处理语句，用于查询的值不必通过`mysqli_real_escape_string()`处理。查看第13章了解关于本主题的更多内容。

(4) 验证艺术家的姓。

```
if (!empty($_POST['last_name'])) {
 $ln = trim($_POST['last_name']);
```

姓的值是必须的。要进行更加严格的验证，你可以使用正则表达式。要获取更高的安全性，你可以应用`strip_tags()`函数。

(5) 包含数据库连接。

```
require ('../../mysqli_connect.php');
```

管理文件夹将会被放到主（`htdocs`）文件夹并且因而在连接脚本之上有两个目录。在包含文件时，记住你的目录结构（参见图19-8）。

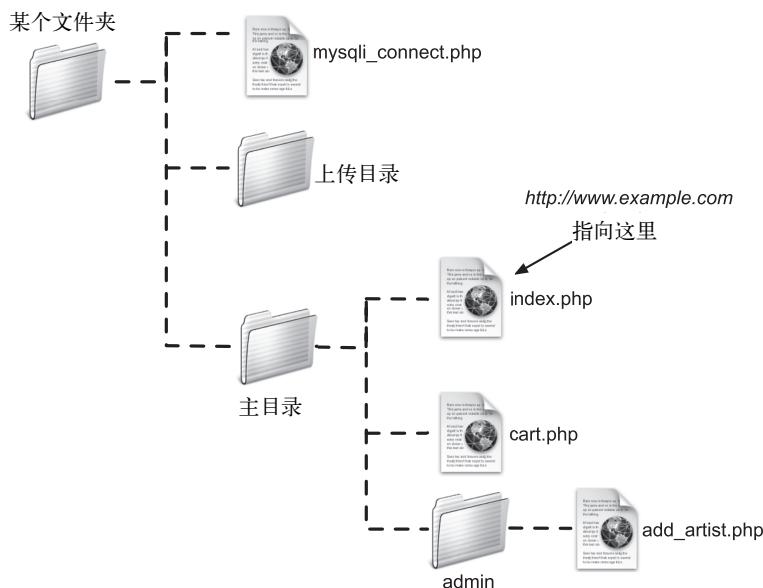


图19-8 网络应用的站点结构。MySQL连接脚本和upload目录（存储图片的地方）不在网站目录（它们无法通过http://访问）

(6) 向数据库中添加艺术家。

```
$q = 'INSERT INTO artists (first_name, middle_name, last_name) VALUES (?, ?, ?)';
$stmt = mysqli_prepare($dbc, $q);
mysqli_stmt_bind_param($stmt, 'sss', $fn, $mn, $ln);
mysqli_stmt_execute($stmt);
```

要向数据库中添加艺术家，查询必须类似于 `INSERT INTO artists (first_name, middle_name, last_name) VALUES ('John', 'Singer', 'Sargent')` 或 `INSERT INTO artists (first_name, middle_name, last_name) VALUES (NULL, NULL, 'Christo')`。查询使用预处理语句运行，在第13中曾有涉及。

`mysqli_stmt_bind_param()` 函数指明了查询需要三个输入（每个问号表示一个）的类型：字符串、字符串和字符串。要是你对此有任何问题，参见第13章。

(7) 报告结果。

```
if (mysqli_stmt_affected_rows($stmt) == 1) {
 echo '<p>The artist has been added.</p>';
 $_POST = array();
} else {
 $error = 'The new artist could not be added to the database!';
}
```

如果执行预处理语句影响了一行，也就是说创建了一行，会显示一个肯定的消息（参见图19-9）。在这种情况下，`$_POST`数组同样需要被重置，因此黏性表单不会重复显示艺术家的信息。

如果预处理语句失败了，`$error`变量中会被添加一条消息，用于在脚本的后面输出。为了调试的目的，你可以在这里添加其他的细节，例如MySQL错误。



图19-9 艺术家成功地添加到数据库中

(8) 关闭预处理语句和数据库连接。

```
mysqli_stmt_close($stmt);
mysqli_close($dbc);
```

(9) 完成艺术家姓并提交语句：

```
} else { // No last name value.
 $error = 'Please enter the artist\'s name!';
}
} // End of the submission IF.
```

如果没有提交姓的值，那么将错误信息赋给\$error变量。

(10) 如果有错误发生的话，打印错误，并闭合PHP块。

```
if (isset($error)) {
 echo '<h1>Error!</h1>
 <p style="font-weight: bold; color: #C00">' . $error . ' Please try again.</p>';
}
?>
```

两个会发生的错误都会被\$error所代表。如果这个变量被设置了，这时候它可以被打印出来（参见图19-10）。错误可以使用一些CSS来写，以让它粗体且是红色的。



图19-10 如果艺术家的姓没有提供，将会显示错误信息。表单是黏性的，因而，记住了其他两个表单输入的值

(11) 开始创建HTML表单。

```
<h1>Add a Print</h1>
<form action="add_artist.php" method="post">
 <fieldset><legend>Fill out the form to add an artist:</ legend>
```

(12) 为添加新艺术家创建输入框。

```
<p>First Name: <input type="text" name="first_name" size="10" maxlength="20" value=<?php if
 (isset($_POST['first_name'])) echo $_POST['first_name']; ?>" /></p>
<p>Middle Name: <input type="text" name="middle_name" size="10" maxlength="20" value=<?php if
 (isset($_POST['middle_name'])) echo $_POST['middle_name']; ?>" /></p>
<p>Last Name: <input type="text" name="last_name" size="10" maxlength="40" value=<?php if
 (isset($_POST['last_name'])) echo $_POST['last_name']; ?>" /></p>
```

每个表单元素都是黏性的，可以被重新显示。

(13) 完成HTML表单。

```
</fieldset>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>
```

(14) 完成HTML页面。

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

(15) 将文件保存为add\_artist.php。

(16) 将add\_artist.php放到网站目录（在管理文件夹）并且在你的浏览器中测试（参见图19-7）和（参见图19-9）。

不要忘记还要将mysqli\_connect.php脚本修改为链接到ecommerce数据库，并放到正确的目录中。

#### ✓ 提示

- 我还是推荐为管理端和公共端分开创建MySQL用户，虽然这里为了简洁起见并没有这么做。管理员需要SELECT、INSERT、UPDATE、和DELETE权限，然而公共用户仅需要SELECT、INSERT和UPDATE。
- 管理页面需要以尽可能最安全的方式来保护。这可以使用Apache的HTML身份认证，使用会话或cookies的登录系统，或者甚至将管理页面放到其他的服务器，可能是离线的服务器（只能从一个地方管理站点）。

### 19.2.2 添加印刷品

第二个要编写的脚本是添加一个新产品（特指印刷品）。该页面将会允许管理员从数据库中选择艺术家，上传一幅图片，并且输入印刷品的详细信息（参见图19-11）。图片会存储在服务器上，印刷品的记录会插入到数据库中。这将是本章中一个较为复杂的脚本，但是所有涉及的技术已经在本书其他的部分讲解过了。

图19-11 用于向目录中添加印刷品的HTML表单

### 创建add\_print.php

(1) 创建新的PHP文档，使用HTML头开始，命名为add\_print.php（参见脚本19-2）：

**脚本 19-2** 这个管理页面向数据库中添加产品。它处理上传文件和添加新的印刷品到 prints 表

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4 <head>
5 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6 <title>Add a Print</title>
7 </head>
8 <body>
9 <?php # Script 19.2 - add_print.php
10 // This page allows the administrator to add a print (product).
11
12 require ('../../mysqli_connect.php');
13
14 if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Handle the form.
15
16 // Validate the incoming data...
17 $errors = array();
18
19 // Check for a print name:
20 if (!empty($_POST['print_name'])) {
21 $pn = trim($_POST['print_name']);
22 } else {
23 $errors[] = 'Please enter the print\'s name!';
24 }
25 }
```

```

26 // Check for an image:
27 if (is_uploaded_file($_FILES['image']['tmp_name'])) {
28
29 // Create a temporary file name:
30 $temp = '.../uploads/' . md5($_FILES['image']['name']);
31
32 // Move the file over:
33 if (move_uploaded_file($_FILES['image']['tmp_name'], $temp)) {
34
35 echo '<p>The file has been uploaded!</p>';
36
37 // Set the $i variable to the image's name:
38 $i = $_FILES['image']['name'];
39
40 } else { // Couldn't move the file over.
41 $errors[] = 'The file could not be moved.';
42 $temp = $_FILES['image']['tmp_name'];
43 }
44
45 } else { // No uploaded file.
46 $errors[] = 'No file was uploaded.';
47 $temp = NULL;
48 }
49
50 // Check for a size (not required):
51 $s = (!empty($_POST['size'])) ? trim($_POST['size']) : NULL;
52
53 // Check for a price:
54 if (is_numeric($_POST['price']) && ($_POST['price'] > 0)) {
55 $p = (float) $_POST['price'];
56 } else {
57 $errors[] = 'Please enter the print\'s price!';
58 }
59
60 // Check for a description (not required):
61 $d = (!empty($_POST['description'])) ? trim($_POST['description']) : NULL;
62
63 // Validate the artist...
64 if (isset($_POST['artist']) && filter_var($_POST['artist'], FILTER_VALIDATE_INT, array('min_range'=>
1))) {
65 $a = $_POST['artist'];
66 } else { // No artist selected.
67 $errors[] = 'Please select the print\'s artist!';
68 }
69
70 if (empty($errors)) { // If everything's OK.
71
72 // Add the print to the database:
73 $q = 'INSERT INTO prints (artist_id, print_name, price, size, description, image_name)
VALUES (?, ?, ?, ?, ?, ?)';
74 $stmt = mysqli_prepare($dbc, $q);
75 mysqli_stmt_bind_param($stmt, 'issss', $a, $pn, $p, $s, $d, $i);
76 mysqli_stmt_execute($stmt);
77

```

```

78 // Check the results...
79 if (mysqli_stmt_affected_rows($stmt) == 1) {
80
81 // Print a message:
82 echo '<p>The print has been added.</p>';
83
84 // Rename the image:
85 $id = mysqli_stmt_insert_id($stmt); // Get the print ID.
86 rename ($temp, "../uploads/$id");
87
88 // Clear $_POST:
89 $_POST = array();
90
91 } else { // Error!
92 echo '<p style="font-weight: bold; color: #C00">Your submission could not be processed
due to a system error.</p>';
93 }
94
95 mysqli_stmt_close($stmt);
96
97 } // End of $errors IF.
98
99 // Delete the uploaded file if it still exists:
100 if (iset($temp) && file_exists ($temp) && is_file($temp)) {
101 unlink ($temp);
102 }
103
104 } // End of the submission IF.
105
106 // Check for any errors and print them:
107 if (!empty($errors) && is_array($errors)) {
108 echo '<h1>Error!</h1>
109 <p style="font-weight: bold; color: #C00">The following error(s) occurred:
';
110 foreach ($errors as $msg) {
111 echo " - $msg
\n";
112 }
113 echo 'Please reselect the print image and try again.</p>';
114 }
115
116 // Display the form...
117 ?>
118 <h1>Add a Print</h1>
119 <form enctype="multipart/form-data" action="add_print.php" method="post">
120
121 <input type="hidden" name="MAX_FILE_SIZE" value="524288" />
122
123 <fieldset><legend>Fill out the form to add a print to the catalog:</legend>
124
125 <p>Print Name: <input type="text" name="print_name" size="30" maxlength="60" value=""
126 <?php if (isset($_POST['print_name'])) echo htmlspecialchars($_POST['print_name']); ?>" /></p>
127
128 <p>Image: <input type="file" name="image" /></p>
129 <p>Artist:
```

```

130 <select name="artist"><option>Select One</option>
131 <?php // Retrieve all the artists and add to the pull-down menu.
132 $q = "SELECT artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) FROM artists ORDER
133 BY last_name, first_name ASC";
134 $r = mysqli_query ($dbc, $q);
135 if (mysqli_num_rows($r) > 0) {
136 while ($row = mysqli_fetch_array ($r, MYSQLI_NUM)) {
137 echo "<option value=\"$row[0]\"";
138 // Check for stickyness:
139 if (isset($_POST['existing']) && ($_POST['existing'] == $row[0])) echo ' selected="selected"';
140 echo "$row[1]</option>\n";
141 }
142 } else {
143 echo '<option>Please add a new artist first.</option>';
144 }
145 mysqli_close($dbc); // Close the database connection.
146 ?>
147 </select></p>
148 <p>Price: <input type="text" name="price" size="10" maxlength="10" value="<?php if
149 (isset($_POST['price'])) echo $_POST['price']; ?>" /> <small>Do not include the dollar sign
150 or commas.</small></p>
151 <p>Size: <input type="text" name="size" size="30" maxlength="60" value="<?php if
152 (isset($_POST['size'])) echo htmlspecialchars($_POST['size']); ?>" /> (optional)</p>
153 </fieldset>
154
155 <div align="center"><input type="submit" name="submit" value="Submit" /></div>
156
157 </form>
158
159 </body>
160 </html>

```

(2) 包含数据库连接脚本并检查表单是否被提交。

```

require ('../../mysqli_connect.php');
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 $errors = array();
}

```

任何表单验证的问题将会被添加到这里初始化的\$errors数组。(由于该脚本可能有多个错误，所以使用了一个数组；add\_artist.php中一次只会发生一个错误。)

(3) 验证印刷品的名字。

```

if (!empty($_POST['print_name'])) {
 $pn = trim($_POST['print_name']);
} else {
 $errors[] = 'Please enter the print\'s name!';
}

```

这是prints表中的一个字段，需要检查它的值。同样，因为这个脚本将使用预处理语句，在查询中使用的值不需要通过`mysqli_real_escape_string()`处理。

如果想加一层保险，你可以在这里应用`strip_tags()`。（如果恶意用户已经进入管理区域，就是大问题了。）

如果没有输入值，一个错误信息将被添加到`$errors`数组中。

(4) 如果选择了一幅图片，为它创建临时名。

```
if (is_uploaded_file($_FILES['image']['tmp_name'])) {
 $temp = '../../uploads/' . md5($_FILES['image']['name']);
```

在讲解使用PHP处理文件上传的技术（参见第11章）时，提到过`is_uploaded_file()`函数。如果文件是被上传的，它将返回TRUE否则返回FALSE。如果文件上传完毕，脚本会尝试将文件移动到上传目录（步骤(5)）。

在这里还有另一件事情发生：图像不会以它们原来的名字保存到服务器（这可能是一个安全隐患）。相反，图像将使用与它们相关的印刷品ID存储。然而，由于该值目前还不知道（因为印刷品的记录还没有添加到数据库中），需要为它生成一个临时名称。要实现它，对图片的原始名称应用`md5()`函数，它能够返回一个32位的散列字符。将这个临时名称，加上图像的最终目录的相对路径（相对于该脚本），赋给`$temp`。

在这个地方可以做一些改进。你还可以验证图片是否为正确的大小和类型。为了保持这个已经很复杂的脚本更易于管理，这里我省略了它，详细的代码应用见第11章。

(5) 将文件移动到更固定的目的地。

```
if (move_uploaded_file($_FILES['image']['tmp_name'], $temp)) {
 echo '<p>The file has been uploaded!</p>';
 $i = $_FILES['image']['name'];
```

此时在脚本中，印刷品的图片将被移动到它的固定位置（上传目录）但是会被给予一个临时名称（稍后将重命名）。这时会打印一条信息来表明操作成功（参见图19-12）。最后，文件原始的名字会被赋给`$i`变量（会在脚本的后面代码中使用）。

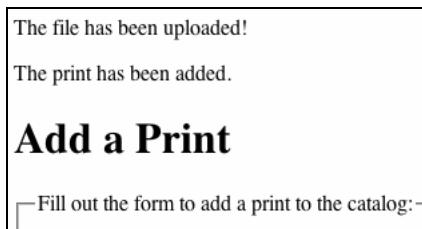


图19-12 为印刷品选择了一个文件，并且成功上传后的结果

(6) 完成图像处理部分。

```
} else {
 $errors[] = 'The file could not be moved.';
 $temp = $_FILES['image']['tmp_name'];
}
} else {
```

```

$errors[] = 'No file was uploaded.';
$temp = NULL;
}

```

如果文件无法被移动到目标目录，第一个else子句会生效。这仅会在目录的路径不正确或目录的权限设置不当的情况下发生（参见图19-13）。无论何种情况，原始的上传值都会被赋给\$temp变量，它仍然会在它的临时位置。这是必要的，因为脚本中没有用到的文件将会在稍后删除。

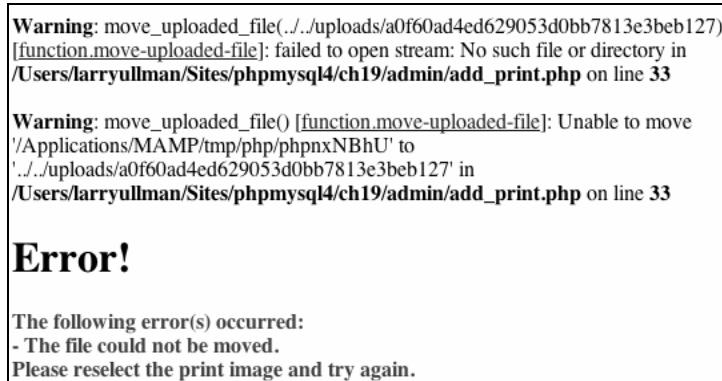


图19-13 如果uploads目录对PHP是不可写的，你会看到类似这样的错误

第二个else子句适用于如果没有文件被上传的情况。这个网站的目的是出售印刷品，实际显示售卖的东西非常重要。如果你愿意，可以为此错误信息添加更多的细节或建议应该上传什么类型和尺寸的文件。

#### (7) 验证尺寸、价格并描述输入。

```

$s = (!empty($_POST['size'])) ? trim($_POST['size']) : NULL;
if (is_numeric($_POST['price']) && ($_POST['price'] > 0)) {
 $p = (float) $_POST['price'];
} else {
 $errors[] = 'Please enter the print\'s price!';
}
$d = (!empty($_POST['description'])) ? trim($_POST['description']) : NULL;

```

尺寸和描述值是可选的，但价格不是。作为基本的有效性测试，使用`is_numeric()`函数确保所提交的价格是一个数字（它应该是十进制数），并且价格应该是大于0的（售卖产品的价格不应该是负数）。如果该值是恰当的，为了安全性，将它强制转换为浮点数。如果没有填写价格或价格无效，会添加一条错误信到\$errors数组中。

#### (8) 验证艺术家。

```

if (isset($_POST['artist']) && filter_var($_POST['artist'], FILTER_VALIDATE_INT, array('min_range' => 1)))
{
 $a = $_POST['artist'];
} else { // No artist selected.
 $errors[] = 'Please select the print\'s artist!';
}

```

管理员将使用一个下拉菜单输入印刷品的艺术家（参见图19-14）。其结果将会是一个正整数的\$\_POST['artist']值。注意，如果你使用的是老版本的PHP，不支持过滤器扩展，需要使用is\_numeric()函数、类型转换以及检查值大于0的条件语句代替。有关详细信息，请参阅第13章。



图19-14 管理员可以通过一个下拉列表选择已有的艺术家

(9) 将记录插入到数据库中。

```
if (empty($errors)) {
 $q = 'INSERT INTO prints (artist_id, print_name, price, size, description, image_name) VALUES
 (?, ?, ?, ?, ?, ?)';
 $stmt = mysqli_prepare($dbc, $q);
 mysqli_stmt_bind_param($stmt, 'issss', $a, $pn, $p, $s, $d, $i);
 mysqli_stmt_execute($stmt);
```

如果\$errors数组仍是空的，那么表示所有的验证测试通过，可以添加印刷品。再次使用预处理语句，执行的查询语句类似INSERT INTO prints (artist\_id, print\_name, price, size, description, image\_name) VALUES (34, 'The Scream', 25.99, NULL, 'This classic...', 'scream.jpg')。

`mysqli_stmt_bind_param()`函数表明查询需要的6个输入（每个问号需要一个）类型是：整形、字符串、双精度（即浮点型）、字符串、字符串、字符串。如对此有任何问题，请参阅第13章。

(10) 确认查询结果。

```
if (mysqli_stmt_affected_rows($stmt) == 1) {
 echo '<p>The print has been added.</p>';
 $id = mysqli_stmt_insert_id($stmt);
 rename ($temp, "../uploads/$id");
 $_POST = array();
} else {
 echo '<p style="font-weight: bold; color: #C00">Your submission could not be processed due to a system
 error.</p>';
}
```

如果查询影响了一行，然后在Web浏览器中会打印一条成功信息（参见图19-12）。接下来，需要获取印刷品的ID，以便使用`rename()`函数重命名相关的图片（它现在在上传文件夹中，但是使用的是临时名称）。`rename()`函数接受文件当前的名称作为它的第一个参数，文件的新名字作为第二个参数。第一个参数的值在前面已经赋给了\$temp变量。第二个参数的值是上传目录的路径加上刚刚确定的印刷品的ID值。需要注意的是，该文件的新名称不包含文件扩展名。在服务器上查看实际文件时可能看起来有点怪异，但是很快你就会发现，当图片文件提供给网站的公众端时，这并没有什么问题。

最后，清空\$\_POST数组，因此它的值可以显示在黏性表单中。

如果查询一行也没有影响，可能发生了一些MySQL错误，这时你需要使用标准调试技巧来找出原因。

(11) 完成条件语句。

```
mysqli_stmt_close($stmt);
} // End of $errors IF.
if (isset($temp) && file_exists ($temp) && is_file($temp)) {
 unlink ($temp);
}
} // End of the submission IF.
```

第一个闭合大括号结束\$errors是否为空的检查。在这种情况下，应该删除服务器上的文件，因为它没有被移动和重命名。

(12) 打印所有错误。

```
if (!empty($errors) && is_array($errors)) {
 echo '<h1>Error!</h1>
<p style="font-weight: bold; color: #C00">The following error(s) occurred:
';
 foreach ($errors as $msg) {
 echo " - $msg
\n";
 }
 echo 'Please reselect the print image and try again.</p>';
}
?>
```

所有发生的错误都将放在\$errors数组中。可以使用foreach循环打印错误（参见图19-15）。错误会以红色加粗的CSS样式进行打印。此外，由于黏性表单不能重新调用已选择的文件，必须提示用户重新选择印刷品图片。（我的书*Effortless E-Commerce with PHP and MySQL*中有一个记住以前上传的文件的示例脚本，但代码有点儿复杂。）

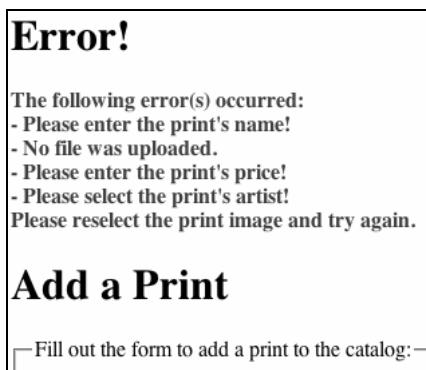


图19-15 未填写完全的表单会生成几个错误

(13) 开始创建HTML表单。

```
<h1>Add a Print</h1>
<form enctype="multipart/form-data" action="add_print.php" method="post">
 <input type="hidden" name="MAX_FILE_SIZE" value="524288" />
 <fieldset><legend>Fill out the form to add a print to the catalog:</legend>
```

```
<p>Print Name: <input type="text" name="print_name" size="30" maxlength="60" value="<?php if
 (isset($_POST['print_name'])) echo htmlspecialchars($_POST['print_name']); ?>" /></p>
<p>Image: <input type="file" name="image" /></p>
```

因为这个表单将允许用户上传文件，所以表单标签中必须包含`enctype`和`MAX_FILE_SIZE`隐藏文本域。由于它的文本输入框中的`value`属性中的代码，表单将是黏性的。需要注意的是，你不能让文件选择控件具有黏性。

如果印刷品的名称、大小或描述中使用了可能有问题的字符，使用`htmlspecialchars()`处理它们，以便不弄乱它们的值。（例如，在图19-11中，印刷品的尺寸和描述中使用的引号。）

(14) 开始艺术家下拉菜单。

```
<p>Artist:
<select
name="artist"><option>Select One</option>
```

艺术家下拉菜单将使用这段PHP代码由artists表中存储的记录生成（参见图19-15）。

(15) 获取所有的艺术家。

```
<?php
$q = "SELECT artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) FROM artists ORDER BY
last_name, first_name ASC";
$r = mysqli_query ($dbc, $q);
if (mysqli_num_rows($r) > 0) {
 while ($row = mysqli_fetch_array ($r, MYSQLI_NUM)) {
 echo "<option value=\"$row[0]\"";
 // Check for stickyness:
 if (isset($_POST['existing']) && ($_POST['existing'] == $row[0])) echo ' selected="selected"';
 echo ">$row[1]</option>\n";
 }
} else {
 echo '<option>Please add a new artist first.</option>';
}
mysqli_close($dbc); // Close the database connection.
?>
</select></p>
```

该查询从数据库中获取每一个艺术家的名字和ID（不使用预处理语句，因为真的没有必要）。MySQL中的`CONCAT_WS()`函数（带分隔符的连接的缩写），用于获取艺术家的完整名字作为一个值。如果你对此查询的语法感到困惑，在MySQL客户端或其他的接口中运行它来查看结果。

如果数据库中没有艺术家，下拉菜单将没有任何选项，所以应该以添加艺术家的指示来代替。

因为想让下拉菜单也是黏性的，所以代码比较复杂。要让任何选择菜单是黏性的，必须为合适的选项添加`selected="selected"`。因此`while`循环中的代码检查`$_POST['existing']`是否被设置，如果它被设置，它的值是否等于被添加到菜单的当前艺术家的ID。

(16) 完成HTML表单。

```
<p>Price: <input type="text" name="price" size="10" maxlength="10" value="<?php if (isset($_
 POST['price'])) echo $_POST['price']; ?>" /> <small>Do not include the dollar sign or commas.</small></p>
<p>Size: <input type="text" name="size" size="30" maxlength="60" value="<?php if (isset($_
 POST['size'])) echo htmlspecialchars($_POST['size']); ?>" /> (optional)</p>
<p>Description: <textarea name="description" cols="40" rows="5"><?php if (isset($_POST
```

```

['description'])) echo $_POST['description'];?></textarea> (optional)</p>
</fieldset>
<div align="center"><input type="submit" name="submit" value="Submit" /></div>
</form>

```

添加每个表单元素的详细描述，如描述是可选的或价格不应该包含美元符号（参见图19-11），帮助管理员正确地填写表单。

(17) 完成HTML页面。

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

(18) 将文件保存为add\_print.php。

(19) 在服务器上创建必要的目录。

管理页面需要创建两个新目录。一个我们称之为admin（参见图19-8），将会存放管理文件。在一个实际的网站中，最好将你的管理目录命名为不太明显的名字。

第二个是uploads目录，应该将它放到Web文档目录并改变它的权限，以让PHP可以将文件移入。更多信息，请参阅第10章。

(20) 将add\_print.php放到Web目录（在管理文件夹）中，并在Web浏览器中测试。

### 19.3 创建公共模板

在深入研究公共端的核心内容之前，需要创建必需的HTML头文件和脚注文件。我将快速介绍这些内容，因为阅读到本书这个位置，读者应该很熟悉所涉及的技术。

#### 1. 建立header.html

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为header.html。（参见脚本19-3）。

#### 脚本 19-3 头文件创建初始的 HTML 并开启 PHP 会话

```

1 <?php # Script 19.3 - header.html
2 // This page begins the session, the HTML page, and the layout table.
3
4 session_start(); // Start a session.
5 ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
6 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
7 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
8 <head>
9 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
10 <title><?php echo (isset($page_title)) ? $page_title : 'Welcome!'; ?></title>
11 </head>
12 <body>
13 <table cellspacing="0" cellpadding="0" border="0" align="center" width="600">
14 <tr>
15 <td align="center" colspan="3"></td>
16 </tr>
17 <tr>
18 <td></td>

```

```

19 <td></td>
20 <td></td>
21 </tr>
22 <tr>
23 <td align="left" colspan="3" bgcolor="#ffffcc">

```

(2) 开启会话。

```
session_start();
```

在每个页面维护用户的会话是非常重要的，因此，将在头文件中启动会话。如果在单个页面上丢失会话，那么将在后续页面上开启一个新的会话，并且用户的历史记录（购物车的内容）将会消失。

(3) 创建HTML头部。

```
?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
 <title><?php echo (isset($page_title)) ? $page_title : 'Welcome!' ; ?></title>
</head>
```

与这个脚本的所有其他版本一样，页面的标题将被设置成一个PHP变量，并在标题标签内打印出来。如果在包含这个页面之前没有设置页面的标题，还会提供一个默认的标题。

(4) 创建表的顶行。

```
<body>
<table cellspacing="0" cellpadding="0" border="0" align="center" width="600">
 <tr>
 <td align="center" colspan="3"></td>
 </tr>
 <tr>
 <td></td>
 <td></td>
 <td></td>
 </tr>
```

这个布局将使用图像来创建公共页面的链接（参见图19-16）。



图19-16 头文件创建的横幅

(5) 开始创建中间行。

```

<tr>
 <td align="left" colspan="3" bgcolor="#ffffcc">

```

各个页面的所有内容都将放在中间一行中，因此，它由头文件创建，并由脚注文件关闭。

(6) 将文件另存为header.html，并存放在Web目录中（创建一个includes目录，将该文件存放在其中）。

### 2. 建立footer.html

(1) 在文本编辑器或IDE中创建一个新的HTML文档，命名为footer.html（参见脚本19-4）。

#### 脚本 19-4 脚注文件关闭 HTML，并在这个过程中创建版权消息

```

1 <!-- Script 19.4 - footer.html -->
2
</td>
3 </tr>
4 <tr>
5 <td align="center" colspan="3" bgcolor="#669966">© Copyright...
6 </td>
7 </tr>
8 </table>
9 </body>
</html>
```

(2) 完成中间行：

```


</td>
</tr>
```

完成头文件开始的表行。

(3) 创建底行，并完成HTML代码（参见图19-17）。

```

<tr>
 <td align="center" colspan="3" bgcolor="#669966">© Copyright...
 </td>
</tr>
</table>
</body>
</html>
```



© Copyright...

图19-17 脚注文件创建的版权行

(4) 将文件另存为footer.html，并存放在Web目录中（也将其存放在includes目录中）。

### 3. 建立index.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为index.php（参见脚本19-5）。

#### 脚本 19-5 用于站点主页的最简洁的脚本

```

1 <?php # Script 19.5 - index.php
2 // This is the main page for the site.
3
4 // Set the page title and include the HTML header:
```

```

5 $page_title = 'Make an Impression!';
6 include ('includes/header.html');
7 ?>
8
9 <p>Welcome to our site....please use the links above...blah, blah, blah.</p>
10 <p>Welcome to our site....please use the links above...blah, blah, blah.</p>
11
12 <?php include ('includes/footer.html'); ?>

```

(2) 创建页面的内容。

```

<p>Welcome to our site....please use the links above...blah, blah, blah.</p>
<p>Welcome to our site....please use the links above...blah, blah, blah.</p>

```

显然，真实的电子商务站点将在主页上具有一些实际的内容。例如，可以在这里显示最近添加的印刷品。

(3) 完成HTML页面。

```
<?php include ('includes/footer.html'); ?>
```

(4) 将文件另存为index.php，存放在Web目录中，并在Web浏览器中测试它（参见图19-18）。



图19-18 电子商务站点的公共主页

### ✓ 提示

- 可以通过本书的配套Web站点下载这个示例中使用的图像，这些图片与脚本和SQL命令放在一起。
- 你可以通过向prints表添加一个date\_entered列，然后按照date\_entered列的倒序获取一些产品，以在首页展示最近添加的项目。

## 19.4 产品目录

为了让顾客能够购买产品，他们需要先查看产品。此时，将创建两个脚本，用于访问产品目录。第一个脚本（browse\_prints.php）将显示可用印刷品的列表（参见图19-19）。如果选择了特定的艺术家，则只会显示那位艺术家的作品（参见图19-20）；否则，将会列出每一件印刷品。

Browse the Prints

## Make an Impression!

[Home](#)    [View Prints](#)    [Shopping Cart](#)

Artist	Print Name	Description	Price
<a href="#">Sandro Botticelli</a>	<a href="#">The Birth Of Venus</a>	A nice print of Botticelli's classic "The Birth Of Venus." Blah, blah, blah.	\$29.95
<a href="#">Roy Lichtenstein</a>	<a href="#">In the Car</a>		\$32.99
<a href="#">Rene Magritte</a>	<a href="#">Empire of Lights</a>		\$24.00
<a href="#">Claude Monet</a>	<a href="#">Rouen Cathedral: Full Sunlight</a>	One in Monet's series of yadda, yadda, yadda	\$39.50
<a href="#">John Singer Sargent</a>	<a href="#">Madame X</a>		\$26.00
<a href="#">Georges-Pierre Seurat</a>	<a href="#">Sunday Afternoon on the Island of La Grande Jatte</a>	One of the most famous paintings in the world and the best example of pointillism...	\$37.50
<a href="#">Georges-Pierre Seurat</a>	<a href="#">The Bathers</a>		\$18.00

© Copyright...

图19-19 browse\_prints.php创建的当前产品列表

Browse the Prints

[http://localhost/ch19/browse\\_prints.php?aid=6](#)

## Make an Impression!

[Home](#)    [View Prints](#)    [Shopping Cart](#)

Artist	Print Name	Description	Price
<a href="#">Georges-Pierre Seurat</a>	<a href="#">Sunday Afternoon on the Island of La Grande Jatte</a>	One of the most famous paintings in the world and the best example of pointillism...	\$37.50
<a href="#">Georges-Pierre Seurat</a>	<a href="#">The Bathers</a>		\$18.00

© Copyright...

图19-20 如果选择了特定的艺术家（通过单击艺术家的名字），页面将只会显示那位艺术家的作品

第二个脚本（view\_print.php）将用于显示单独一件印刷品的信息，包括图片（参见图19-21）。在这个页面上，顾客将找到一个Add to Cart链接，使得可以把印刷品添加到购物车中。由于印刷品的图片存储在Web根目录的外面，view\_print.php将使用一个单独的脚本（与第11章中的show\_image.php非常相似）用于显示图像。



图19-21 显示单个作品的页面

### 1. 建立browse\_prints.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为browse\_prints.php（参见脚本19-6）。

**脚本 19-6 browse\_prints.php** 脚本显示目录中的每件印刷品，或者显示特定艺术家的所有印刷品，这取决于\$\_GET['aid']是否存在

```

1 <?php # Script 19.6 - browse_prints.php
2 // This page displays the available prints (products).
3
4 // Set the page title and include the HTML header:
5 $page_title = 'Browse the Prints';
6 include ('includes/header.html');
7
8 require ('../mysqli_connect.php');
9
10 // Default query for this page:
11 $q = "SELECT artists.artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_
name, price, description, print_id FROM artists, prints WHERE artists.artist_id = prints.artist_id
ORDER BY artists.last_name ASC, prints.print_name ASC";
12
13 // Are we looking at a particular artist?
14 if (isset($_GET['aid'])) && filter_var($_GET['aid'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
15 // Overwrite the query:
16 $q = "SELECT artists.artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist,
print_name, price, description, print_id FROM artists, prints WHERE artists.artist_id=prints.
artist_id AND prints. artist_id={$_GET['aid']} ORDER BY prints.print_name";

```

```

17 }
18
19 // Create the table head:
20 echo '<table border="0" width="90%" cellspacing="3" cellpadding="3" align="center">
21 <tr>
22 <td align="left" width="20%">Artist</td>
23 <td align="left" width="20%">Print Name</td>
24 <td align="left" width="40%">Description</td>
25 <td align="right" width="20%">Price</td>
26 </tr>';
27
28 // Display all the prints, linked to URLs:
29 $r = mysqli_query ($dbc, $q);
30 while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
31
32 // Display each record:
33 echo "<t<tr>
34 <td align=\"left\">{$row['artist']}</td>
36 <td align=\"left\">{$row['description']}

```

(2) 定义查询，检索每件印刷品。

```
$q = "SELECT artists.artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name,
 price, description, print_id FROM artists, prints WHERE artists. artist_id = prints.artist_id ORDER
 BY artists.last_name ASC, prints.print_name ASC";
```

这个查询是跨artists表和prints表的标准联结（用于检索艺术家名字信息以及每件印刷品的信息）。第一次查看这个页面时，将返回每位艺术家的每件印刷品（参见图19-22）。

(3) 检查URL中的艺术家ID。

```
if (isset($_GET['aid']) && filter_var($_GET['aid'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
 $q = "SELECT artists.artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name,
 price, description, print_id FROM artists, prints WHERE artists. artist_id=prints.artist_id AND
 prints.artist_id={$_GET['aid']} ORDER BY prints. print_name";
}
```

如果用户单击了一位艺术家的名字，将把用户返回到这个页面，但是现在URL将是（例如）browse\_prints.php?aid=6（参见图19-20）。在这种情况下，将把AND prints.artist\_id = x子句添加到查询中，并稍加修改ORDER BY，使得只会显示那位艺术家的作品。因此，可以使用同一个查询的不同变体处理这个脚本的两个不同的作用——显示每一件印刷品，或者只显示单独一位艺术家的那些印刷品，而脚本的其他部分在任何一种情况下的工作方式相同。



```

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT artists.artist_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name, price, description, print_id FROM artists, prints WHERE artists.artist_id = prints.artist_id ORDER BY artists.last_name ASC, prints.print_name ASC\G
***** 1. row *****
 artist_id: 1
 artist: Sandro Botticelli
print_name: The Birth Of Venus
 price: 29.95
description: A nice print of Botticelli's classic "The Birth Of Venus." Blah, blah, blah.
 print_id: 1
***** 2. row *****
 artist_id: 3
 artist: Roy Lichtenstein
print_name: In the Car
 price: 32.99
description: NULL
 print_id: 2
***** 3. row *****
 artist_id: 4
 artist: Rene Magritte
print_name: Empire of Lights
 price: 24.00
description: NULL
 print_id: 3
***** 4. row *****
 artist_id: 5
 artist: Claude Monet
print_name: Rouen Cathedral: Full Sunlight
 price: 39.50
description: One in Monet's series of yadda, yadda, yadda
 print_id: 4
***** 5. row *****
 artist_id: 2
 artist: John Singer Sargent
print_name: Madame X
 price: 26.00
description: NULL
 print_id: 5
***** 6. row *****
 artist_id: 6
 artist: Georges-Pierre Seurat
print_name: Sunday Afternoon on the Island of La Grande Jatte
 price: 37.50
description: One of the most famous paintings in the world and the best example of pointillism...
 print_id: 6
***** 7. row *****
 artist_id: 6
 artist: Georges-Pierre Seurat
print_name: The Bathers
 price: 18.00
description: NULL
 print_id: 7
7 rows in set (0.00 sec)

mysql>

```

图19-22 在MySQL客户端中运行browse\_prints.php主查询之后的结果

出于安全的目的，对艺术家ID使用了过滤器扩展，如果你的PHP版本不支持过滤器扩展，就要使用类型强制转换，确保它在查询中使用之前是一个正整数。

#### (4) 创建表头。

```

echo '<table border="0" width="90%" cellspacing="3" cellpadding="3" align="center">
<tr>
 <td align="left" width="20%">Artist</td>
 <td align="left" width="20%">Print Name</td>
 <td align="left" width="40%">Description</td>
 <td align="right" width="20%">Price</td>
</tr>';

```

#### (5) 显示每一条返回的记录。

```

$r = mysqli_query ($dbc, $q);
while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
echo "\t<tr>
 <td align=\"left\">{$row['artist']}</td>
 <td align=\"left\">{$row['print_name']}</td>

```

```

<td align="left">{$row ['description']}</td>
<td align="right">\${$row ['price']}</td>
</tr>\n";
} // End of while loop.

```

我希望页面显示艺术家的全名、印刷品名称、描述，以及每一条返回记录的价格。此外，应该把艺术家的名字链接回这个页面（通过把艺术家的ID追加到URL中），并且应该把印刷品名称链接到 `view_print.php`（通过把印刷品ID追加到URL中，参见图19-23）

</tr>	<tr>
	<td align="left"><a href="browse_prints.php?aid=6">Georges-Pierre Seurat</a></td>
	<td align="left"><a href="view_print.php?pid=6">Sunday Afternoon on the Island of La Grande Jatte</a></td>
	<td align="left">One of the most famous paintings in the world and the best example of pointillism...</td>
	<td align="right">\$37.50</td>
</tr>	</tr>
	<td align="left"><a href="browse_prints.php?aid=6">Georges-Pierre Seurat</a></td>
	<td align="left"><a href="view_print.php?pid=7">The Bathers</a></td>
	<td align="left"></td>
	<td align="right">\$18.00</td>
</tr>	</tr>

图19-23 页面的源代码揭示了如何把艺术家ID和印刷品ID追加到链接中

这段代码没有包括对`mysqli_num_rows()`的调用，以确认在获取一些结果之前返回了它们，但是为了安全起见，可以在真实版本中添加它。

#### (6) 关闭表、数据库连接和HTML页面。

```

echo '</table>';
mysqli_close($dbc);
include ('includes/footer.html');
?>

```

(7) 将文件另存为`browse_prints.php`，存放在Web目录中，然后在Web浏览器中测试它（参见图19-19和图19-20）。

#### ✓ 提示

□ 参见19.7节，从各方面扩展这个脚本。

#### 2. 建立`view_print.php`

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为`view_print.php`（参见脚本19-7）。

**脚本 19-7 view\_print.php** 脚本显示了特定印刷品的详细信息。它还包括一个链接，用于把产品添加到顾客的购物车中

```

1 <?php # Script 19.7 - view_print.php
2 // This page displays the details for a particular print.
3
4 $row = FALSE; // Assume nothing!
5
6 if (isset($_GET['pid']) && filter_var($_GET['pid'], FILTER_VALIDATE_INT, array('min_range' => 1)))
{ // Make sure there's a print ID!
7
8 $pid = $_GET['pid'];
9

```

```

10 // Get the print info:
11 require ('../mysqli_connect.php'); // Connect to the database.
12 $q = "SELECT CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name, price,
13 description, size, image_name FROM artists, prints WHERE artists.artist_id=prints.artist_id AND
14 prints.print_id=$pid";
15 $r = mysqli_query ($dbc, $q);
16 if (mysqli_num_rows($r) == 1) { // Good to go!
17
18 // Fetch the information:
19 $row = mysqli_fetch_array ($r, MYSQLI_ASSOC);
20
21 // Start the HTML page:
22 $page_title = $row['print_name'];
23 include ('includes/header.html');
24
25 // Display a header:
26 echo "<div align=\"center\">
27 {$row['print_name']} by
28 {$row['artist']}
";
29
30 // Print the size or a default message:
31 echo (is_null($row['size'])) ? '(No size information available)' : $row['size'];
32
33 echo "
\${$row['price']}
34 Add to Cart
35 </div>
";
36
37 // Get the image information and display the image:
38 if ($image = @getimagesize ("../uploads/$pid")) {
39 echo "<div align=\"center\"><img src=\"show_image.php?image=$pid&name=" . urlencode ($row['image_
40 name']) . "\" $image[3] alt=\"$row['print_name']\" /></div>\n";
41 } else {
42 echo "<div align=\"center\">No image available.</div>\n";
43 }
44
45 // Add the description or a default message:
46 echo '<p align="center">' . (is_null($row['description'])) ? '(No description available)' :
47 $row['description'] . '</p>';
48
49 } // End of the mysqli_num_rows() IF.
50
51 mysqli_close($dbc);
52
53 } // End of $_GET['pid'] IF.
54
55 if (!$row) { // Show an error message.
56 $page_title = 'Error';
57 include ('includes/header.html');
58 echo '<div align="center">This page has been accessed in error!</div>';
59 }
56
57 // Complete the page:
58 include ('includes/footer.html');
59 ?>

```

(2) 创建标志变量。

```
$row = FALSE;
```

使用\$row变量来跟踪这个页面上是否发生了问题。如果一切顺利，这个变量将存储来自数据库的印刷品信息。如果发生了问题，那么在脚本末尾，\$row变量仍然为假，并且页面应该指示错误。

(3) 验证已经把印刷品ID传递给这个页面。

```
if (isset($_GET['pid']) && filter_var($_GET['pid'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
 $pid = $_GET['pid'];
```

如果该脚本没有接收到有效的印刷品ID，那么它不会工作，因此，首先会检查数字ID是否存在。为了在脚本后面方便引用\$\_GET['pid']，这里将其赋给\$pid。

(4) 从数据库中检索信息。

```
require ('../mysqli_connect.php');
$q = "SELECT CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name, price,
 description, size, image_name FROM artists, prints WHERE artists.artist_id=prints.artist_id AND prints.
 print_id=$pid";
$r = mysqli_query ($dbc, $q);
```

这个查询是一个联结，就像browse\_prints.php中的那个联结一样，但是它只选择特定印刷品的信息（参见图19-24）。

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name, price, des-
 raction, size, image_name FROM artists, prints WHERE artists.artist_id=prints.artist_id AND print-
 id=$pid
 ****1. row *****
 artist: Georges-Pierre Seurat
 print_name: Sunday Afternoon on the Island of La Grande Jatte
 price: 37.50
 description: One of the most famous paintings in the world and the best example of pointillism...
 size: 40" x 30"
 image_name: C50741big.jpg
1 row in set (0.00 sec)

mysql>
```

图19-24 在MySQL客户端中运行view\_print.php之后的结果

(5) 如果返回一条记录，则检索信息，设置页面标题，并包含HTML头部。

```
if (mysqli_num_rows($r) == 1) {
 $row = mysqli_fetch_array ($r, MYSQLI_ASSOC);
 $page_title = $row['print_name'];
 include ('includes/header.html');
```

浏览器窗口的标题（参见图19-21）将是印刷品的名称。

(6) 开始显示印刷品信息。

```
echo "<div align=\"center\">
{$row['print_name']} by
{$row['artist']}
";
echo (is_null($row['size'])) ? '(No size information available)' : $row['size'];
echo "
\${$row['price']}
Add to Cart
</div>
";
```

印刷品的标题将是印刷品的名称（粗体显示），其后接着艺术家的名字、印刷品的尺寸及其价格。最后，将显示一个链接，让用户选择把这件印刷品添加到购物车中（参见图19-25）。购物车链接指向add\_cart.php脚本，并把印刷品ID传递给它。

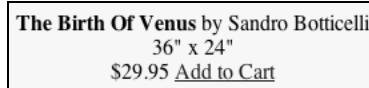


图19-25 在页面顶部将显示印刷品信息，以及一个用于购买它的链接

由于印刷品的尺寸可以是NULL值，所以可以使用三元运算符来打印出该尺寸或默认的消息。

#### (7) 显示图像。

```
if ($image = @getimagesize ("../uploads/$pid")) {
 echo "<div align=\"center\"><img src=\"show_image.php?image=$pid&name=" . urlencode($row ['image_name']) . "
 \" $image[3] alt=\"$row ['print_name']\" /></div>\n";
} else {
 echo "<div align=\"center\">No image available.</div>\n";
}
```

因为图片被安全的存储在网站根目录的外面，所以需要另一个PHP脚本来向浏览器提供图片(`show_image.php`脚本被用作一个代理脚本)。图片自身的名称就是印刷品ID，这已经在URL中传递给了这个页面。

脚本的这一部分将首先尝试通过使用`getimagesize()`函数来检索图像的尺寸。如果操作成功，则会显示图像本身。这个过程有点不同寻常，因为图像的源会调用`show_image.php`页面(参见图19-26)。这个脚本(下面将会编写它)期望在URL中传递印刷品ID以及图像的文件名(在添加印刷品时将其存储在数据库中)。使用PHP脚本显示图像与第11章中使用`show_image.php`完全相同，只是现在它发生在另一个页面内，而不是发生在它自己的窗口中。

```
The Birth Of Venus by
Sandro Botticelli
36" x 24"

$29.95
Add to Cart
</div>
<div align="center">
A nice print of Botticelli's classic "The Birth Of Venus." Blah, blah, blah.</p><!-- Script 19.4 - footer.html -->
```

图19-26 `view_print.php`页面的HTML源代码显示`img`标签的`src`属性如何调用`show_image.php`脚本，并传递给它所需要的值

如果脚本不能检索图像信息(由于图像不在服务器上，或者没有上传图像)，则会代之以显示一条消息。

#### (8) 显示描述信息。

```
echo '<p align="center">' . ((is_null($row ['description']))? '(No description available)' : $row ['description']) .
'</p>';
```

最后，添加印刷品的描述信息(参见图19-27)。如果没有在数据库中存储印刷品描述信息，则会打印默认的消息。

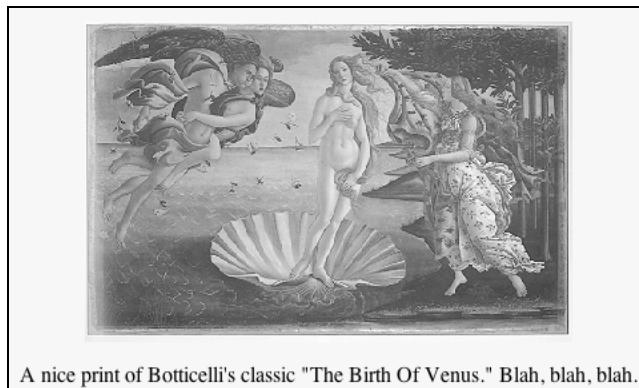


图19-27 印刷品的图像后面接着它的描述信息

(9) 完成两个主条件语句。

```
} // End of the mysqli_num_rows() IF.
mysqli_close($dbc);
} // End of $_GET['pid'] IF.
```

(10) 如果发生问题，就显示一条出错消息。

```
if (!$row) {
 $page_title = 'Error';
 include ('includes/header.html');
 echo '<div align="center">This page has been accessed in error!</div>';
}
```

如果不管由于什么原因，而导致不能从数据库中检索印刷品的信息，那么\$row仍然为假并且应该显示一个错误（参见图19-28）。因为如果发生问题，将不会包含HTML头部，所以在这里必须先包含它。



图19-28 在view\_print.php页面没有在URL中接收到有效的印刷品ID时所得到的结果

(11) 完成页面。

```
include ('includes/footer.html');
?>
```

(12) 将文件另存为view\_print.php，并存放在Web目录中。

如果此时在浏览器中运行这个页面，不会显示图片，因为还没有编写show\_image.php。

### ✓ 提示

- 许多电子商务站点为Add to Cart链接使用一幅图像。为了在这个示例中这样做，可以用待使用图像的代码代替文本Add to Cart（在链接标签内）。重要的是add\_cart.php页面仍然会传递产品ID号。
- 如果想在显示多个作品的页面（比如browse\_prints.php）上添加Add to Cart链接，可以执行针对单个作品的完全相同的操作。只需确保每个链接都会把正确的印刷品ID传递给add\_cart.php页面。
- 如果想显示产品的可用性，可以添加in\_stock字段到prints表中。然后依据这一列中那件印刷品的值，显示Add to Cart链接或一条Product Currently Out of Stock消息。

### 3. 编写show\_image.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为show\_image.php（参见脚本19-8）。

**脚本 19-8** 这个脚本由 view\_print.php（参见脚本 19-7）调用，并显示 uploads 目录中存储的图像

```

1 <?php # Script 19.8 - show_image.php
2 // This pages retrieves and shows an image.
3
4 // Flag variables:
5 $image = FALSE;
6 $name = (!empty($_GET['name'])) ? $_GET['name'] : 'print image';
7
8 // Check for an image value in the URL:
9 if (isset($_GET['image']) && filter_var($_GET['image'], FILTER_VALIDATE_INT, array('min_range'
=> 1))) {
10
11 // Full image path:
12 $image = '../uploads/' . $_GET['image'];
13
14 // Check that the image exists and is a file:
15 if (!file_exists($image) || (!is_file($image))) {
16 $image = FALSE;
17 }
18
19 } // End of $_GET['image'] IF.
20
21 // If there was a problem, use the default image:
22 if (!$image) {
23 $image = 'images/unavailable.png';
24 $name = 'unavailable.png';
25 }
26
27 // Get the image information:
28 $info = getimagesize($image);
29 $fs = filesize($image);
30

```

```

31 // Send the content information:
32 header ("Content-Type: {$info['mime']}\\n");
33 header ("Content-Disposition: inline; filename=\"$name\\n\"");
34 header ("Content-Length: $fs\\n");
35
36 // Send the file:
37 readfile ($image);

```

该脚本将要做的事情与第11章中的show\_image.php相同，只不过将要把两个值传递给这个页面。服务器上的实际图像的文件名将是一个数字，它对应于印刷品ID。原始图像的文件名存储在数据库中，在把图像发送到Web浏览器时会使用它。这个页面将不包含任何HTML，并且不能在这个PHP开始标签之前把任何内容发送到Web浏览器。

在这里初始化两个标志变量。第一个标志变量（\$image）将引用服务器上的物理图像。它被假定为假，并且需要证实。\$name变量将是提供给Web浏览器的文件的名称，它应该来自于URL。如果不是这样，就会赋予一个默认值。

(2) 检查URL中的图像值。

```
if (isset($_GET['image']) && filter_var($_GET['image'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
```

在继续执行下面的操作之前，确保脚本接收到一个图像值，它应该是view\_print.php中针对每件印刷品的HTML src属性的一部分（参见图19-26）。

(3) 检查图像是否是服务器上的一个文件。

```
$image = '../uploads/' . $_GET['image'];
if (!file_exists ($image) || (!is_file($image))) {
 $image = FALSE;
}
```

作为一种安全措施，把图像的完整路径硬编码为../uploads和接收到的图像名称的组合。因为服务器上文件的名称是一个整数，所以为了加强安全性对其类型进行强制转换。也可以在这里验证文件的MIME类型（image/jpg、image/gif）。

接下来，脚本将检查图像存在于服务器上，并且它是一个文件（与目录相对）。如果任何一个条件为假，就将\$image设置为FALSE，指示一个问题。

(4) 完成验证条件语句并检查问题。

```
} // End of $_GET['image'] IF.
if (!$image) {
 $image = 'images/unavailable. png';
 $name = 'unavailable.png';
}
```

如果图片不存在，或不是一个文件，或没有将图片名字传递给这个脚本，这个条件语句都将会是TRUE。在这种情况下，将会使用默认图片。然而这样很有可能会被黑客攻击：view\_print.php脚本对图片做了一些初步的验证，在show\_image.php可以访问图片时调用它。

(5) 检索图像信息。

```
$info = getimagesize($image);
$fs = filesize($image);
```

为了把文件发送到Web浏览器，脚本需要知道文件的类型和大小。这段代码与第11章中的相同。

(6) 发送文件。

```
header ("Content-Type: {$info['mime']}\\n");
header ("Content-Disposition: inline; filename=\"$name\\n\"");
header ("Content-Length: $fs\\n");
readfile ($image);
```

这些header()调用将把文件数据发送到Web浏览器，这与它们在第11章中所做的完全相同。再分析一下整个语法，第一行基于MIME类型，使浏览器准备好接收文件。第二行设置待发送文件的名称。

最后一个header()函数指示预期有多少数据。使用readfile()函数发送文件数据本身，这个函数读入一个文件，并立即把内容发送到Web浏览器。

(7) 将文件另存为show\_image.php，将其存放在Web目录中，并通过查看任何印刷品在Web浏览器中测试它。

本页面并不包含HTML，它只是把图像文件发送到Web浏览器。同第11章一样，省略了PHP结尾标签，以避免潜在问题。

#### ✓ 提示

□ 如果由于某种原因，view\_print.php页面没有显示图像，则需要通过直接在Web浏览器中运行show\_image.php来调试问题。查看view\_print.php的HTML源代码，找出img标签的src属性的值。

然后把它用作URL（换句话说，转移到http://www.example.com/show\_image.php?image=23&name=BirthOfVenus.jpeg）。如果发生错误，则找出错误的最佳方式是直接运行show\_image.php。

#### 搜索产品目录

如果你需要添加这个数据库的结构，以让其支持相当简单的搜索功能。这里只有三个用于搜索目的逻辑字段：印刷品的名称、描述和艺术家的姓。使用LIKE查询这些字段，语法如下：

```
SELECT...WHERE prints.description LIKE'%keyword%' OR prints.print_name LIKE '%keyword%' ...
```

另一种办法是，建立高级的搜索，让用户选择是否搜索艺术家的名字或印刷品的名称（类似于Internet Movie Database，www.imdb.com，演员和电影名的实现）。

## 19.5 购物车

一旦创建了产品目录，就像前面几页中所做的一样，那么实际的购物车本身可能出奇简单。在这个示例中使用的方法是，在会话中记录产品ID、价格和数量。知道这3种信息将允许脚本计算总额，以及执行任何其他所需的任务。

接下来的两个示例将为购物车提供所有必要的功能。第一个脚本（add\_cart.php）将把项目添加到购物车中。第二个脚本（view\_cart.php）将显示购物车的内容，并允许顾客更新它。

### 19.5.1 添加项目

add\_cart.php脚本将获取一个参数（待购印刷品的ID）并使用它来更新购物车。购物车本身存储在一个会话中，可以通过\$\_SESSION['cart']变量访问它。购物车将是一个多维数组，它的键是产品ID。

数组元素的值本身也是数组：一个元素用于数量，另一个元素用于价格（参见表19-6）。

表19-6 \$\_SESSION['cart']变量

(索引)	数 量	价 格
2	1	54.00
568	2	22.95
37	1	33.50

### 创建add\_cart.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为add\_cart.php（参见脚本19-9）。

**脚本 19-9** 这个脚本通过引用产品（或印刷品）ID 并操纵会话数据，把产品添加到购物车中

```

1 <?php # Script 19.9 - add_cart.php
2 // This page adds prints to the shopping cart.
3
4 // Set the page title and include the HTML header:
5 $page_title = 'Add to Cart';
6 include ('includes/header.html');
7
8 if (isset($_GET['pid']) && filter_var($_GET['pid'], FILTER_VALIDATE_INT, array('min_range'
=> 1))) { // Check for a print ID.
9 $pid = $_GET['pid'];
10
11 // Check if the cart already contains one of these prints;
12 // If so, increment the quantity:
13 if (isset($_SESSION['cart'][$pid])) {
14
15 $_SESSION['cart'][$pid]['quantity']++; // Add another.
16
17 // Display a message:
18 echo '<p>Another copy of the print has been added to your shopping cart.</p>';
19
20 } else { // New product to the cart.
21
22 // Get the print's price from the database:
23 require ('../mysqli_connect.php'); // Connect to the database.
24 $q = "SELECT price FROM prints WHERE print_id=$pid";
25 $r = mysqli_query ($dbc, $q);
26 if (mysqli_num_rows($r) == 1) { // Valid print ID.
27
28 // Fetch the information.
29 list($price) = mysqli_fetch_array ($r, MYSQLI_NUM);
30
31 // Add to the cart:
32 $_SESSION['cart'][$pid] = array ('quantity' => 1, 'price' => $price);
33
34 // Display a message:
35 echo '<p>The print has been added to your shopping cart.</p>';
36
37 } else { // Not a valid print ID.

```

```

38 echo '<div align="center">This page has been accessed in error!</div>';
39 }
40
41 mysqli_close($dbc);
42
43 } // End of isset($_SESSION['cart'][$pid]) conditional.
44
45 } else { // No print ID.
46 echo '<div align="center">This page has been accessed in error!</div>';
47 }
48
49 include ('includes/footer.html');
50 ?>

```

(2) 检查印刷品ID是否传递到了脚本。

```
if (isset($_GET['pid']) && filter_var($_GET['pid'], FILTER_VALIDATE_INT, array('min_range' => 1))) {
 $pid = $_GET['pid'];
}
```

与view\_print.php脚本一样，如果没有接收到印刷品ID或者印刷品ID不是数字，那么将不希望继续处理这个脚本。如果接收到ID，将其值赋给\$pid，以便在脚本后面引用。

(3) 确定购物车中是否已经添加了这个印刷品的副本。

```
if (isset($_SESSION['cart'][$pid])) {
 $_SESSION['cart'][$pid]['quantity']++;
 echo '<p>Another copy of the print has been added to your shopping cart.</p>';


```

在把当前印刷品添加到购物车（通过把它的数量设置为1）之前，需要检查购物车中是否有一个副本。例如，如果顾客选择印刷品#519，然后决定订购另一个副本，那么购物车现在应该包含该印刷品的两个副本。因此，首先会检查购物车是否有一个用于当前印刷品ID的值。如果是这样，就递增数量。代码\$\_SESSION['cart'][\$pid]['quantity']++等同于：

```
$_SESSION['cart'][$pid]['quantity'] = $_SESSION['cart'][$pid]['quantity'] + 1.
```

然后，显示一条消息（参见图19-29）。



图19-29 为已经在购物车中的项目单击Add to Cart链接后的结果

(4) 如果产品尚未在购物车中，从数据库中获取它的价格。

```
} else { // New product to the cart.
 require ('../mysqli_connect.php');
 $q = "SELECT price FROM prints WHERE print_id=$pid";
 $r = mysqli_query ($dbc, $q);
 if (mysqli_num_rows($r) == 1) {
 list($price) = mysqli_fetch_array ($r, MYSQLI_NUM);
```

如果产品现在还未在购物车中，`else`子句将会生效。这里，印刷品的价格使用印刷ID从数据库中获取。

(5) 向购物车中添加新产品。

```
$_SESSION['cart'][$pid] = array ('quantity' => 1, 'price' => $price);
echo '<p>The print has been added to your shopping cart.</p>';
```

在获取了产品的价格之后，添加一个新的元素到`$_SESSION['cart']`多维数组中。由于每个元素在`$_SESSION['cart']`中都是一个单独的数组，因而使用`array()`函数设置数量和价格，然后显示一段简短的信息（参见图19-30）



图19-30 向购物车添加新产品后的结果

(6) 完成条件语句。

```
} else { // Not a valid print ID.
 echo '<div align="center">This page has been accessed in error!</div>';
}
mysqli_close($dbc);
} // End of isset($_SESSION ['cart'][$pid]) conditional.
} else { // No print ID.
 echo '<div align="center">This page has been accessed in error!</div>';
}
```

如果不能从数据库中检索价格，就会应用第一个`else`子句，这意味着提交的印刷品ID是无效的。如果这个页面根本没有接收到印刷品ID或者印刷品ID不是数字，就会应用第二个`else`子句。在这两种情况下，都将显示一条出错消息（参见图19-31）。



图19-31 仅当add\_cart.php页面在URL中接收到有效的印刷品ID时，它才会把项目添加到购物车中

(7) 包括HTML脚注，并完成PHP页面。

```
include ('includes/footer.html');
?>
```

(8) 将文件另存为add\_cart.php，存放在Web目录中，并在Web浏览器中测试它(通过单击Add to Cart链接)。

#### ✓ 提示

- 购物车中要存储的最重要的信息是唯一的产品ID和该产品的价格。其他信息（包括价格）可以从数据库中检索到。此外，也可以从数据库中检索价格、印刷品名称和艺术家名字，并把所有信息都存储在购物车中，以便可以轻松显示它。
- 购物车存储在\$\_SESSION['cart']（而不仅仅是\$\_SESSION）中。其他信息（如来自数据库的用户ID）也可能存储在\$\_SESSION中。

### 19.5.2 查看购物车

与add\_cart.php相比，view\_cart.php脚本更复杂，因为它服务于两个目的。第一，它将详细显示购物车的内容（参见图19-32）。第二，它将给顾客提供一个选项，即通过更改购物车中的数量来更新购物车（或者通过使其数量为0来删除一个项目）。为了履行这两个职责，把购物车的内容显示为一个表单，并让页面把这个表单提交回它自身。



图19-32 将购物车显示为表单，可以在其中更改具体数量

最后，这个页面将链接到checkout.php脚本，作为结款过程的第一步。

#### 创建view\_cart.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为view\_cart.php（参见脚本19-10）。

**脚本 19-10 view\_cart.php** 脚本将会显示购物车的内容，并允许用户更新购物车的内容

```

1 <?php # Script 19.10 - view_cart.php
2 // This page displays the contents of the shopping cart.
3 // This page also lets the user update the contents of the cart.
4
5 // Set the page title and include the HTML header:
6 $page_title = 'View Your Shopping Cart';
7 include ('includes/header.html');
8
9 // Check if the form has been submitted (to update the cart):
10 if ($_SERVER['REQUEST_METHOD'] == 'POST') {
11
12 // Change any quantities:
13 foreach ($_POST['qty'] as $k => $v) {
14
15 // Must be integers!
16 $pid = (int) $k;
17 $qty = (int) $v;
18
19 if ($qty == 0) { // Delete.
20 unset ($_SESSION['cart'][$pid]);
21 } elseif ($qty > 0) { // Change quantity.
22 $_SESSION['cart'][$pid]['quantity'] = $qty;
23 }
24

```

```

25 } // End of FOREACH.
26
27 } // End of SUBMITTED IF.
28
29 // Display the cart if it's not empty...
30 if (!empty($_SESSION['cart'])) {
31
32 // Retrieve all of the information for the prints in the cart:
33 require ('../mysqli_connect.php'); // Connect to the database.
34 $q = "SELECT print_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name
35 FROM artists, prints WHERE artists.artist_id = prints.artist_id AND prints.print_id IN (
36 foreach ($_SESSION['cart'] as $pid => $value) {
37 $q .= $pid . ',';
38 }
39 $q = substr($q, 0, -1) . ') ORDER BY artists.last_name ASC';
40 $r = mysqli_query ($dbc, $q);
41
42 // Create a form and a table:
43 echo '<form action="view_cart.php" method="post">
44 <table border="0" width="90%" cellspacing="3" cellpadding="3" align="center">
45 <tr>
46 <td align="left" width="30%">Artist</td>
47 <td align="left" width="30%">Print Name</td>
48 <td align="right" width="10%">Price</td>
49 <td align="center" width="10%">Qty</td>
50 <td align="right" width="10%">Total Price</td>
51 </tr>
52 ';
53
54 // Print each item...
55 $total = 0; // Total cost of the order.
56 while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
57
58 // Calculate the total and sub-totals.
59 $subtotal = $_SESSION['cart'][$row['print_id']]['quantity'] * $_SESSION['cart'][$row['print_id']]
60 ['price'];
61 $total += $subtotal;
62
63 // Print the row:
64 echo "\t<tr>
65 <td align=\"left\">{$row['artist']}</td>
66 <td align=\"left\">{$row['print_name']}</td>
67 <td align=\"right\">{$$_SESSION['cart'][$row['print_id']]['price']}</td>
68 <td align=\"center\"><input type=\"text\" size=\"3\" name=\"qty[{$row['print_id']}]\"
69 value=\"{$$_SESSION['cart'][$row['print_id']]['quantity']}\" /></td>
70 <td align=\"right\">$". number_format ($subtotal, 2) . "</td>
71 </tr>\n";
72
73 } // End of the WHILE loop.
74
75 mysqli_close($dbc); // Close the database connection.
76
77 // Print the total, close the table, and the form:
78 echo '<tr>

```

```

76 <td colspan="4" align="right">Total:</td>
77 <td align="right">$' . number_format ($total, 2) . '</td>
78 </tr>
79 </table>
80 <div align="center"><input type="submit" name="submit" value="Update My Cart" /></div>
81 </form><p align="center">Enter a quantity of 0 to remove an item.
82

Checkout</p>;
83
84 } else {
85 echo '<p>Your cart is currently empty.</p>';
86 }
87
88 include ('includes/footer.html');
89 ?>

```

(2) 如果提交了表单，则更新购物车。

```

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
 foreach ($_POST['qty'] as $k => $v) {
 $pid = (int) $k;
 $qty = (int) $v;
 if ($qty == 0) {
 unset ($_SESSION['cart'][$pid]);
 } elseif ($qty > 0) {
 $_SESSION['cart'][$pid]['quantity'] = $qty;
 }
 } // End of FOREACH.
} // End of SUBMITTED IF.

```

如果提交了表单，那么这个脚本需要更新购物车，以反映输入的数量。这些数量将表现为一个名为\$\_POST['qty']]的数组，其索引是印刷品ID，其值是新的数量（参见图19-33，查看表单的HTML源代码）。如果新数量是0，那么应该通过将其置“0”从购物车中删除那个项目。如果新数量不是0而是一个正数，那么就要更新购物车，以反映这一事实。

>Sandro Botticelli</td>	>The Birth Of Venus</td>	>\$29.95</td>	<input type="text" size="3" name="qty[1]" value="1" /></td>	>\$29.95</td>
>Claude Monet</td>	>Rouen Cathedral: Full Sunlight</td>	>\$39.50</td>	<input type="text" size="3" name="qty[4]" value="2" /></td>	>\$79.00</td>
>Georges-Pierre Seurat</td>	>Sunday Afternoon on the Island of La Grande Jatte</td>	>\$37.50</td>	<input type="text" size="3" name="qty[6]" value="1" /></td>	>\$37.50</td>

图19-33 查看购物车表单的HTML源代码显示数量字段如何反映产品ID，以及购物车中该印刷品的数量

如果数量不是一个大于或等于0的数字，那么将不对购物车执行任何更改。这会阻止用户输入一个负数，从而产生一个负的结欠金额，并获得退款。

你也可以使用过滤器扩展来验证印刷品ID和数量。

(3) 如果购物车不为空，则创建查询来显示其内容。

```
if (!empty($_SESSION['cart'])) {
 require ('../mysqli_connect.php');
 $q = "SELECT print_id, CONCAT_WS(' ', first_name, middle_name, last_name) AS artist, print_name FROM
 artists, prints WHERE artists.artist_id = prints.artist_id AND prints.print_id IN (";
 foreach ($_SESSION['cart'] as $pid => $value) {
 $q .= $pid . ',';
 }
 $q = substr($q, 0, -1) . ') ORDER BY artists.last_name ASC';
 $r = mysqli_query ($dbc, $q);
```

这个查询是一个联结，它类似于本章中已经使用过的一个联结，它用于为购物车中的每件印刷品检索所有艺术家和印刷品信息，其中增加了IN SQL子句的使用。这里不是只检索一件印刷品的信息(例如，在view\_print.php中)，而是想检索购物车中每件印刷品的所有信息。为了执行该任务，在一个类似SELECT...print\_id IN(519,42,427)...这样的查询中使用印刷品ID的列表。也可以使用SELECT...WHERE print\_id=519 OR print\_id=42 OR print\_id=427...这样的查询，但是这就显得啰嗦了。

为了生成查询的IN(519,42,427)部分，使用一个for循环把每个印刷品ID以及一个逗号添加到\$q中。为了删除最后一个逗号，应用了substr()函数，它用于删除最后一个字符。

(4) 开始创建HTML表单并创建表。

```
echo '<form action="view_cart.php" method="post">
<table border="0" width="90%" cellspacing="3" cellpadding="3" align="center">
<tr>
 <td align="left" width="30%>Artist</td>
 <td align="left" width="30%>Print Name</td>
 <td align="right" width="10%>Price</td>
 <td align="center" width="10%>Qty</td>
 <td align="right" width="10%>Total Price</td>
</tr>
';
```

(5) 检索返回的记录。

```
$total = 0;
while ($row = mysqli_fetch_array ($r, MYSQLI_ASSOC)) {
 $subtotal = $_SESSION['cart'][$row['print_id']]['quantity'] * $_SESSION['cart'][$row['print_id']]['price'];
 $total += $subtotal;
```

在显示购物车时，还想计算订单总额，因此，首先会初始化\$total变量。然后，对于返回的每一行（它表示一件印刷品），用那个项目的价格乘以数量来确定该项目的部分总额（由于多维数组\$\_SESSION['cart']，这个语法有点复杂）。然后把这个部分总额加到\$total变量上。

(6) 打印返回的总额。

```
echo "\t<tr>
<td align=\"left\">{$row['artist']} </td>
<td align=\"left\">{$row['print_name']}</td>
<td align=\"right\">\$$_SESSION ['cart'][$row['print_id']] ['price']</td>
<td align=\"center\"><input type=\"text\" size=\"3\" name=\"qty {$row['print_id']}\" value=\"$$_SESSION ['cart'][$row['print_id']] ['quantity']\" /></td>
```

```
<td align="right">$" . number_format ($subtotal, 2) . "</td>
</tr>\n";
```

还将在表中分行打印出每一条记录，并把数量显示为一种预先设置了其值的文本输入框类型（基于会话中的数量值）。还会格式化并打印每个项目的部分总额（数量乘以价格）。

(7) 完成while循环，关闭数据库连接。

```
} // End of the WHILE loop.
mysqli_close($dbc);
```

(8) 完成表和表单。

```
echo '<tr>
<td colspan="4" align="right">Total:</td>
<td align="right">$' . number_format ($total, 2) . '</td>
</tr>
</table>
<div align="center"><input type="submit" name="submit" value="Update My Cart" /></div>
</form><p align="center">Enter a quantity of 0 to remove an item.

Checkout</p>';
```

在表的最后一行中显示运行的订单总额，并使用number\_format()函数对其进行格式化。表单还会指导用户如何删除项目，以及提供一个链接用于包含结款页面。

(9) 完成主条件语句和PHP页面。

```
} else {
 echo '<p>Your cart is currently empty.</p>';
}
include ('includes/footer.html');
?>
```

这个else子句完成if (!empty(\$\_SESSION['cart'])) { 条件语句 }。

(10) 将文件另存为view\_cart.php，存放在Web目录中，并在Web浏览器中测试它（参见图19-34和图19-35）。



图19-34 如果更改了任何数量并单击Update My Cart，就会更新购物车和订单总额  
(对比图19-32)



图19-35 通过把数量设置为0，删除了购物车中的所有项目

### ✓ 提示

- 在更复杂的Web应用程序上，我倾向于编写一个PHP页面，严格地用于显示购物车的内容。因为这可能需要多个页面，一种有意义的做法是在一个可包含的文件中提供这种功能。
- 安全的电子商务应用程序的一个方面是：留意如何发送和使用数据。例如，如果把产品的价格放在可以被轻松更改的URL中，那将更不安全。

## 19.6 记录订单

在把所有的产品显示为一个目录以及在用户装满了他们的购物车之后，还有最后3个步骤：

- 用户结款；
- 在数据库中记录订单；
- 履行订单。

具有讽刺意味的是，在本书中不能充分演示最重要的部分——结款（即收取顾客的钱），因为它对于单个站点是如此特别。因此，我代之以在框注“结款过程”中给出该过程的一个概述。我的另一本书*Effortless E-commerce with PHP and MySQL*用了两章的篇幅介绍了两种不同的结款方法。

类似地，履行订单的动作超出了本书的范围。对于物理产品，这意味着需要打包和送货。然后，通过在数据库中注明运输日期，将订单标记为已送货。理解这个概念应该不太费力，但你需要注意库存变化。对于电子产品，履行订单就是提供用户数字内容。

在本章中可以充分展示的是，如何在数据库中存储订单信息。为了确保完全、准确地把订单输入到orders表和order\_contents表中，将使用事务。在第7章中使用MySQL客户端介绍过这个主题。这里将通过PHP脚本执行事务。为了提高安全性和性能，这个脚本还将利用第13章中讨论的预处理语句。

这个脚本（`checkout.php`）代表顾客将在电子商务过程中看到的最后一步。由于本书跳过了这个脚本之前的步骤，所以需要对这个过程进行少许修改。

**结算过程**

结算过程包括以下3个步骤。

- (1) 确认订单。
- (2) 确认和提交账单和发货信息。

## (3) 处理账单信息。

现在，对于中级程序员来说，步骤(1)和(2)可以独立地、非常轻松地完成。十有八九，在用户注册并登录后，步骤(2)中的数据将来自customers表。

第(3)步是最棘手的，在这本书中并不能得到充分解决。这一步有很多变数，取决于账单处理方式和被谁处理。根据出售的产品是稍后发货还是立即交付，法律上是不同的，这也让情况变得更加复杂（就像访问网站或下载文件）。

大多数中小型规模的电子商务网站使用第三方平台处理金融交易。通常，这涉及向另一个Web站点发送账单信息、订单总额以及门店数量（参考电子商务网站自身的特点）。第三方网站将处理实际结算过程，扣除用户的费用并把款项划给电子商务网站。最后，结果码会被发送回电子商务网站，程序会根据它作出相应的反应。在这种情况下，第三方处理结算平台会提供给开发者适当的代码和说明，以便使用它们系统接口。

欲了解关于此主题更多的信息或帮助，查看我的*E-Commerce with PHP and MySQL*一书或向支持论坛求助。

## 创建submit\_order.php

(1) 在文本编辑器或IDE中创建一个新的PHP文档，命名为Submit\_order.php（参见脚本19-11）。

**脚本 19-11** 电子商务应用程序中的最后一步是在数据库中记录订单信息。它使用事务来确保正确地提交整个订单

```

1 <?php # Script 19.11 - checkout.php
2 // This page inserts the order information into the table.
3 // This page would come after the billing process.
4 // This page assumes that the billing process worked (the money has been taken).
5
6 // Set the page title and include the HTML header:
7 $page_title = 'Order Confirmation';
8 include ('includes/header.html');
9
10 // Assume that the customer is logged in and that this page has access to the customer's ID:
11 $cid = 1; // Temporary.
12
13 // Assume that this page receives the order total:
14 $total = 178.93; // Temporary.
15
16 require ('../mysqli_connect.php'); // Connect to the database.
17
18 // Turn autocommit off:
19 mysqli_autocommit($dbc, FALSE);
20
21 // Add the order to the orders table...
22 $q = "INSERT INTO orders (customer_id, total) VALUES ($cid, $total)";
23 $r = mysqli_query($dbc, $q);
24 if (mysqli_affected_rows($dbc) == 1) {
25
26 // Need the order ID:

```

```
27 $oid = mysqli_insert_id($dbc);
28
29 // Insert the specific order contents into the database...
30
31 // Prepare the query:
32 $q = "INSERT INTO order_contents (order_id, print_id, quantity, price) VALUES (?, ?, ?, ?)";
33 $stmt = mysqli_prepare($dbc, $q);
34 mysqli_stmt_bind_param($stmt, 'iid', $oid, $pid, $qty, $price);
35
36 // Execute each query; count the total affected:
37 $affected = 0;
38 foreach ($_SESSION['cart'] as $pid => $item) {
39 $qty = $item['quantity'];
40 $price = $item['price'];
41 mysqli_stmt_execute($stmt);
42 $affected += mysqli_stmt_
43 affected_rows($stmt);
44 }
45
46 // Close this prepared statement:
47 mysqli_stmt_close($stmt);
48
49 // Report on the success....
50 if ($affected == count($_SESSION['cart'])) { // Whohoo!
51
52 // Commit the transaction:
53 mysqli_commit($dbc);
54
55 // Clear the cart:
56 unset($_SESSION['cart']);
57
58 // Message to the customer:
59 echo '<p>Thank you for your order. You will be notified when the items ship.</p>';
60
61 // Send emails and do whatever else.
62 } else { // Rollback and report the problem.
63
64 mysqli_rollback($dbc);
65
66 echo '<p>Your order could not be processed due to a system error. You will be contacted in
67 order to have the problem fixed. We apologize for the inconvenience.</p>';
68 // Send the order information to the administrator.
69 }
70
71 } else { // Rollback and report the problem.
72
73 mysqli_rollback($dbc);
74
75 echo '<p>Your order could not be processed due to a system error. You will be contacted in order
76 to have the problem fixed. We apologize for the inconvenience.</p>';
77 // Send the order information to the administrator.
```

```

78
79 }
80
81 mysqli_close($dbc);
82
83 include ('includes/footer.html');
84 ?>

```

(2) 创建两个临时变量

```

$cid = 1;
$total = 178.93;

```

为了把订单输入到数据库中，这个页面需要另外两个信息：顾客的标识号（它是customers表中的customer\_id）和订单总额。在顾客登录时，可能就会确定其标识号（它可能存储在会话中）。订单总额也可能存储在会话中（在把税款和送货计算在内之后），或者可能通过开账单过程被这个页面获得。但是，由于不会立即访问其中任一个值（从而跳过了这些步骤），所以将创建两个变量来假冒它们。

(3) 包含数据库连接，并关闭MySQL的autocommit模式。

```

require ('../mysqli_connect.php');
mysqli_autocommit($dbc, FALSE);

```

`mysqli_autocommit()`函数可以打开或关闭MySQL的autocommit特性。因为我想使用事务来确保正确地输入整个订单，所以首先将关闭autocommit。关于事务的任何问题，见第7章或MySQL手册。

(4) 添加订单到orders表中。

```

$q = "INSERT INTO orders (customer_id, total) VALUES ($cid, $total)";
$r = mysqli_query($dbc, $q);
if (mysqli_affected_rows($dbc) == 1) {

```

这个查询非常简单，只是把顾客的ID号和订单的总额输入到orders表中。这个表中的order\_date字段将被自动设置成当前日期和时间，因为它是一个TIMESTAMP列。

(5) 检索订单ID。

```
$oid = mysqli_insert_id($dbc);
```

在order\_contents表中需要orders表中的order\_id值，以把这两个表关联起来。

(6) 准备一个把订单内容插入到数据库中的查询。

```

$q = "INSERT INTO order_contents (order_id, print_id, quantity, price) VALUES (?, ?, ?, ?)";
$stmt = mysqli_prepare($dbc, $q);
mysqli_stmt_bind_param($stmt, 'iiid', $oid, $pid, $qty, $price);

```

查询本身把4个值都插入到order\_contents表中，对于这个订单中购买的每件印刷品，该表中都会有一条针对它的记录。定义并准备查询（用占位符代替值）。`mysqli_stmt_bind_param()`函数把4个变量关联到占位符，它们的类型依次是：整型、整型、整型、双精度型（即浮点型）。

(7) 遍历购物车，把每件印刷品都插入到数据库中。

```

$affected = 0;
foreach ($_SESSION['cart'] as $pid => $item) {
 $qty = $item['quantity'];
 $price = $item['price'];
 mysqli_stmt_execute($stmt);
}

```

```

 $affected += mysqli_stmt_affected_rows($stmt);
}
mysqli_stmt_close($stmt);

```

通过遍历购物车，就像在view\_cart.php中所做的那样，可以一次访问一个项目。为了清楚看出这里发生的事情：已经准备好了查询（发送到MySQL并进行解析）并且将变量赋予占位符。在循环内，把来自于会话中的值赋予两个新变量。调用`mysqli_stmt_execute()`函数，它执行预处理语句，并在那一刻使用变量值。`$oid`值将不会在迭代过程中发生变化，但是`$pid`、`$qty`和`$price`将会发生变化。

为了确认所有的查询都执行成功，必须跟踪受影响的行数。在循环外面将变量`$affected`初始化为0。在循环内，在每次执行预处理语句之后，将受影响的行数加到这个变量上。

#### (8) 报告事务的成功。

```

if ($affected == count($_SESSION ['cart'])) {
 mysqli_commit($dbc);
 unset($_SESSION['cart']);
 echo '<p>Thank you for your order. You will be notified when the items ship.</p>';
}

```

这个条件语句用于查看输入到数据库中的记录是否与购物车中存在的产品一样多。简言之，是把每件产品插入到order\_contents表中吗？如果是，那么就会完成事务，并可以提交它。然后清空购物车并感谢用户。逻辑上讲，你在这里还应该给顾客发送确认电子邮件。

#### (9) 处理任何MySQL问题。

```

} else {
 mysqli_rollback($dbc);
 echo '<p>Your order could not be processed due to a system error. You will be contacted in order
 to have the problem fixed. We apologize for the inconvenience.</p>';
}
} else {
 mysqli_rollback($dbc);
 echo '<p>Your order could not be processed due to a system error. You will be contacted in order to
 have the problem fixed. We apologize for the inconvenience.</p>';
}

```

如果没有在order\_contents表中插入正确数量的记录，就会应用第一个else子句。如果原来的orders表查询失败，就会应用第二个else子句。在任何一种情况下都应该撤销整个事务，因此会调用`mysqli_rollback()`函数。

在这个过程中，如果此时发生问题，则后果相当严重，因为已经向顾客收费，但是没有在数据库中建立他们的订单记录。这不应该发生，但是以防万一，应该把所有的数据写到一个文本文件，和/或把它们通过电子邮件发送给站点的管理员，或者做某些事情以创建这个订单的记录。如果没有这样做，就会碰到一些极其恼怒的顾客。

#### (10) 完成页面。

```

mysqli_close($dbc);
include ('includes/footer.html');
?>

```

(11) 将文件另存为submit\_order.php，存放在Web目录中，并在Web浏览器中测试它(参见图19-36)。



图19-36 在把所有的数据都输入到数据库中之后，现在就完成了顾客的订单

可以通过单击view\_cart.php中的链接访问这个页面。

(12) 通过使用另一个界面查看数据库，确认正确地存储了订单（参见图19-37）。

```
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide
mysql> SELECT * FROM orders;
+-----+-----+-----+-----+
| order_id | customer_id | total | order_date |
+-----+-----+-----+-----+
| 1 | 1 | 178.93 | 2011-06-10 15:30:11 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM order_contents;
+-----+-----+-----+-----+-----+-----+
| oc_id | order_id | print_id | quantity | price | ship_date |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | 1 | 1 | 29.95 | NULL |
| 2 | 1 | 1 | 4 | 39.50 | NULL |
| 3 | 1 | 1 | 6 | 37.50 | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图19-37 使用mysql客户查看MySQL数据库中的订单

### ✓ 提示

- 在一个活动的、工作的站点上，应该把真实值赋予\$customer和\$total变量，以让这个脚本工作。在尝试在数据库中存储订单之前，你还可能希望确保购物车不为空。
- 如果愿意学习关于电子商务的更多信息，或者查看关于这个过程的变体，那么在Web上快速搜索一下或参考我编写的*Effortless E-Commerce with PHP and MySQL*，即可找到关于用PHP建立电子商务应用程序的多个示例和教程。

## 19.7 回顾和实践

无论你是对本书内容存有疑问，还是在自学中遇到问题，都可以把问题发到本书论坛，我们会为你答疑解惑。

注意，这里的某些问题和提示涉及前面章节介绍的内容，目的是强化里面的重点知识。

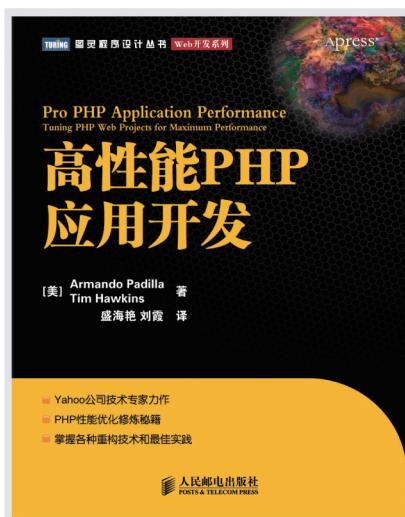
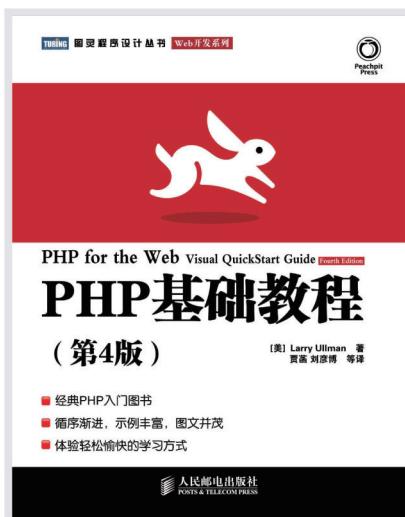
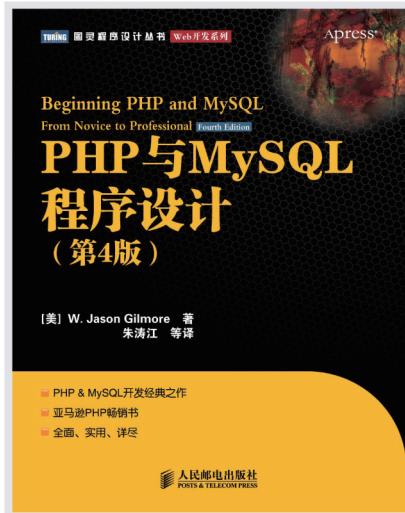
### 19.7.1 回顾

- 哪种MySQL存储引擎支持事务？
- MySQL支持什么类型的索引？决定哪列需要索引和用什么类型的索引最合适的标准是什么？
- 什么是安全套接字层（SSL）？为什么它对电子商务网站很重要？你的网站启用SSL了吗？
- 为什么印刷品的图片存储时没有扩展名不是问题？
- 应该采取什么样的步骤来调试PHP脚本与MySQL数据库交互的问题？
- 什么是多维数组？
- MySQL的自动提交功能是什么意思？
- `mysqli_insert_id()`函数的作用是什么？
- 提交或回滚事务是什么意思？

### 19.7.2 实践

- 扩展artists表的定义，记录每位艺术家的其他信息，在网站公开显示这些信息。
- 为印刷品表添加`quantity_on_hand`列。为`add_print.php`添加一个表单元素，使管理员可以指明最初库存中这个产品有多少份。在用户页面，仅为有效的产品提供“添加到购物车”选项。当产品被购买后（如在`checkout.php`），通过卖出的数目减少每个产品现有的数量。
- 扩展users表的定义来包括其他的相关信息。
- 使用第18章学到的知识，为客户创建注册、登录和注销功能。
- 为用户创建可以查看自己账户、更改自己密码、查看过往订单的功能。
- 为网站的管理端创建并应用模板。
- 使用会话，限制通过验证的用户访问网站的管理端。
- 应用第11章中的代码改善`add_print.php`的安全性，让它可以正确地验证上传文件的类型和尺寸。
- 将`add_print.php`中动态生成的艺术家下拉列表用作公众端的一个导航工具。要做到这一点，将表单的`action`属性设置为`browse_print.php`，将下拉菜单的名字改为`aid`，使用GET方法，当用户选择一个艺术家并点击提交时，将他带到（比如）`browse_print.php?aid=5`。
- 为`browse_prints.php`使用分页。请在第10章查看更多关于分页的内容。
- 在`browse_prints.php`，通过为列标题添加链接（例如，`browse_prints.php?order=price`），然后基于`$_GET['order']`是否存在和它的值，改变查询中的`ORDER BY`。这个想法已在第9章中演示了。

- 组合view\_print.php和add\_cart.php的一些功能，让刚添加到购物车的印刷品详情也可以在add\_cart.php中显示。
- 要在add\_cart.php中显示购物车的内容，将view\_cart.php中相关代码添加到add\_cart.php中。
- 创建管理员查看订单列表的功能。提示：第一页的功能类似browse\_prints.php，不同的是它会在orders表和customers表之间执行联结。页面显示订单总额、订单编号、订单日期、客户的名称。你可以将订单ID链接到view\_order.php脚本，在URL中传递订单ID。view\_order.php脚本会使用订单ID从order\_contents表中获取订单的详情（在处理中联结prints表和artists表）
- 为订单添加计算税金和运费的功能，将这些值存储在订单表中。



图灵社区会员 StinkBC(StinkBC@gmail.com) 专享 尊重版权

“本书在我所读过的技术图书中首屈一指，不仅内容全面，包含基础到提高的所有必要内容，还有几个最常见的应用，而且阐述通俗易懂，让初学者也能很快地开发出高质量的Web应用程序。”

——亚马逊读者



PHP and MySQL for Dynamic Web Sites Visual QuickPro Guide Fourth Edition

# PHP与MySQL动态网站开发 (第4版)

本书采用基于任务的方法来讲解PHP和MySQL，使用图形、图表指导读者深入学习语言，向读者展示了如何构造Web站点。通过本书，读者可以快速、高效地学习PHP和MySQL，并立刻成为一位构筑站点的高手！

书中通过大量实例、屏幕截图和详细的解释，循序渐进地涵盖了开发人员最需要的知识点。由于采用任务导向的方式组织，本书同时也是一部很好的参考书，读者可以根据各种常见任务查询书中内容，直接应用到实际工作中去。

## 本版新增内容

- 用新的范例演示读者经常会问到的技术
- 一些高级MySQL和SQL范例
- jQuery JavaScript框架使用手册
- 面向对象编程基础知识和基本语法
- 提升脚本和网页安全性的更多信息和示例
- 如何阻止常见的Web站点滥用和攻击
- 使用多种语言和时区



Peachpit  
Press



图灵社区：[www.ituring.com.cn](http://www.ituring.com.cn)

新浪微博：[@图灵教育](#) [@图灵社区](#)

反馈/投稿/推荐信箱：[contact@turingbook.com](mailto:contact@turingbook.com)

热线：(010)51095186转604

ISBN 978-7-115-29940-6



ISBN 978-7-115-29940-6

定价：99.00元

人民邮电出版社网址：[www.ptpress.com.cn](http://www.ptpress.com.cn)

**分类建议** 计算机/网络技术/PHP

图灵社区会员 StinkBC(StinkBC@gmail.com) 专享 尊重版权