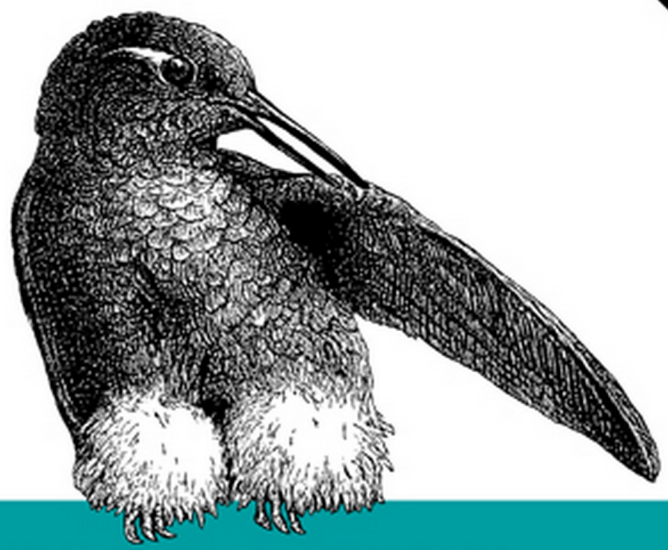
PHP: The Good Parts

揭示PHP的精华

涵盖PHP 5.3的精粹



PHP

语言精粹

Peter B. MacIntyre 著
Susie Sedlacek 序
刘涛 丁静 译

O'REILLY®

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

内 容 简 介

这是一本可以带你迈入 PHP 殿堂的书。

PHP 作为当今主流的服务器端开发语言，广泛应用于世界上各种排名比较靠前的网站，如 Facebook、Flickr 和 Wikipedia 等。其广泛的应用与其强大的功能相辅相成，密不可分。

在这本书中，你将看到 PHP 中最为精华的特征，包括类型系统、面向对象机制、数据库交互、安全性保证、内建函数库等。

通过书中极为实用的代码，上述特征的学习和应用将被无缝连接在一起。

作者 Peter B. MacIntyre 在软件开发领域已有超过 20 年的经验，曾是 PHP|Architect 杂志的特约编辑和作者。长期从事 PHP 相关的工作使作者对 PHP 的发展历程非常了解。这也使本书不单可以让人了解 PHP 当前是什么样子，也可以让人了解到它为什么是现在这个样子。

版权贸易登记号　图字：01-2011-7092

图书在版编目（CIP）数据

PHP 语言精粹 /（美）麦因泰（MacIntyre,P.B.）著；刘涛，丁静译. —北京：电子工业出版社，2012.3
书名原文：PHP: The Good Parts
ISBN 978-7-121-15385-3

Ⅰ. ①P…　Ⅱ. ①麦… ②刘… ③丁…　Ⅲ. ①PHP 语言－程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2011)第 252567 号

责任编辑：徐津平　　文字编辑：江　立
封面设计：Karen Montgmery　张　健
印　　刷：北京市顺义兴华印刷厂
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：720×1000　1/16　印张：11.25　字数：275.2 千字
印　　次：2012 年 3 月第 1 次印刷
定　　价：39.00 元

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

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

服务热线：（010）88258888。

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了Make杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——Wired

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——CRN

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

目录

Table of Contents

序

Foreword

你可能会问，我们为什么还需要一本关于PHP的书呢？现在还有必要指出所谓的那些“优点”吗？这不是随便的一本书，而且现在也不是PHP生命过程中任何一个特别的时间。目前PHP正以前所未有的速度被应用，越来越多的人想了解什么是PHP，并且理解它为何如此受欢迎。由于个人和企业对PHP的应用不断深入，全世界在进行深度介入，开始大型项目或培训之前，都有必要通过一本书来快速了解这种语言的基本内容。这就是本书目的所在。不论你是谁——非程序员、Java程序员或是RPG程序员——本书都是你了解PHP优势的最佳途径。一旦你拥有本书，决定用或不用PHP和是否深入学习它将变得异常容易。

过去的三年甚至更多时间里，我和把 PHP 带到现在这个位置的小组一起工作，他们使 PHP 更易在商业中使用。我也曾经目睹 PHP 通过多种途经给企业带来巨大利益。一个例子是，使 1800 个汽车经销商和 2400 个服务中心（包括超过 42,000 名用户）可以自定义并下单，同时在欧洲 17 个国家内向客户提供服务的可注册门户网站。整合 15 个不同的软件应用程序，并使之与后端系统进行通信，在任何一个语言中都是一项挑战，但 PHP 提供了一个快速转换并迅速做出市场响应的黏合性语言解决方案，这使这家企业在很短的时间内实现其目标，节省的成本也比其他可能的解决方案多。

在我以前担任 Zend 技术公司全球服务副总裁时，企业内部的 PHP 需求逐步增加，这使我们开始制定一个全面的服务产品组合，它包括了一个强大的培训课程。开发者也有类似的需求，他们同时也从这些课程中受益，尽管最近的经济环境并不好，我们看到学生报名人数或考勤并没有下降。我推荐这本书作为这些课程的一个前期准备。

多年来，彼得·麦金太尔一直在通过他的写作、教学和演讲帮助很多人获得 PHP 的好处。大约三年前当我在 Zend 的同事把我介绍给他时，我感到非常愉快。那时，彼得已经为我们完成了几个网络研讨会，而我的同事强烈地感到他有能力传达有价值的信息，他们

认为他会是我们培训团队的生力军。自那时以来，我一直在观察彼得使用多种方式来传播使用 PHP 的便捷之处、其他相关的技术和现有的工具，以及学习它的许多途径。他是这项技术的热情用户，他非常了解它和所有相关的技术。

在这本书中，彼得提到了一些较大的可见的应用（如 Facebook 和雅虎），在我整个参与 PHP 的过程中，我见过很多大组织（包括百思买、JC Penney 公司、敦豪、福克斯互动、GE 公司和德意志电信）在许多方面使用 PHP，这显示了这个简单语言的力量。想想吧，就业机会！如果你希望使用 PHP 来发展和扩大你的事业，你会有兴趣知道下面的事情。在与 ODesk 公司的交谈中，我了解到，在网站开发的所有需求技能中，PHP 是最前面的一个，并且需求量每小时都在迅速增加。速度甚至超过了 Zend 认证工程师。

我认为最令人惊奇的是 IBMi/RPG 世界对 PHP 的采用。我们惊喜地看到一些非过程语言的程序员加入到网络发展的新时代，起先也许有一些轻微的纠结，但是一旦他们看到可以提供给他们的用户（或客户）现代化的接口，像 PHP 提供给 IBMi 那样的复杂功能，紧接着必然是急切的渴望，所以，如果你是一个 IBMi 用户或 RPG 程序员，买这本书，你很快就会发现 PHP 的魔力。绿色屏幕可以实现现代化，你的应用程序可以继续使用。

如果你是一个强大的 Java 程序员而 PHP 处在你的“下风”呢？对我们所有 PHP 的信徒而言，这始终是一个有趣的讨论！我听到的一些搞笑台词：“PHP 不是 Java”和“Java 死了”，然后另一个说法是“PHP 是给业余爱好者使用的”。我们所要做的是通过 Zend 框架、Magento 或 Drupal 的下载量上升速度来认识 PHP 中被简单性所掩饰掉的威力和潜力。PHP 无法替代 Java，同时 Java 也无法取代 PHP。这两种语言共存于最成功和最前沿的 IT 团队中。需要注意的一点是：请仔细阅读读本书——有许多使用 PHP 的方式，这远比 Java 程序员基于 Java 所假设的要多，这本书将引导你如何最好地利用 PHP。

今天，世界上有超过三分之一的网站是用 PHP 编写的，而且这个数字还在增长。PHP 语言本身已发展到拥有完整的开发基础设施，能使复杂的应用程序应用于商业的程度；Zend 框架、Magento 和 Drupal 都是这个进化过程中很成功的例子。市场分析公司 Gartner 最近发表了一份报告，预测全球的 PHP 开发人员将会在 2013 年增长到 500 万（在 2007 年和 2009 年分别为 300 万和 400 万）。他们提供的一个短期预测表明，PHP 将一直是一个被广泛采用的网页开发技术。

是时候让更多的人，也让你来明白这个看似嘈杂的世界背后究竟都有什么了。因此，我邀请你一路向前，阅读、欣赏、加入不断增长的 PHP 用户大家庭。你绝不会再回到从前！

——Zend 公司全球服务部前副总裁 Susie Sedlacek

前言

Preface

本书约定

下面是本书中使用的体例约定：

斜体（*Italic*）

表示新术语、URL、E-mail 地址、文件名及文件扩展名。

等宽字体（`Constant width`）

程序清单，以及段落中引用的程序元素，如变量或函数的名称、数据库、数据类型、环境变量、语句和关键字。

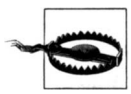
等宽加粗字体（**`Constant width bold`**）

显示命令或其他应该由用户输入的文本。

等宽斜体（*`Constant width italic`*）

显示应该替换为用户提供的值或由上下文决定的值的文本。

这个图标表示提示、建议、或一般说明。



这个图标表示警告或注意。

使用代码示例

本书可以帮助你完成工作。通常在程序和文档中使用本书的代码不必联系我们获得许可，除非你要复制代码的重要部分。例如，使用本书几大块代码编写程序不需要许可，而销售或分销 O'Reilly 随书附带光盘上的例子则需要许可；引用本书及其示例代码以回答问题不需要许可，而将本书大量的示例代码附加到你的产品文档中则需要许可。

我们感谢但不要求注明出处。出处的格式一般包括标题、作者、出版商和 ISBN。例如："PHP：由彼得 B. 麦金太尔完成。© 2010 彼得 B. 麦金太尔，978-0-596-80437-4"。

如果你觉得示例代码的使用不合理或不符合以上的许可权限，请随时联系我们：

permissions@oreilly.com

如何联系我们

本书中的每个例子都已经在不同的平台上测试通过，生产过程中的每个步骤信息也都已经过证实。但是，错误和疏漏在所难免。如果你发现任何细节问题，或对以后版本的改进有什么建议，欢迎与我们联系，我们将十分感激。你可以通过以下方式联系作者和编辑：

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

以下网站列出了本书的勘误表、示例和任何附加信息，网址为：http://oreilly.com/catalog/9780596804374

如果想询问技术问题，或者发表有关本书的评论，请发送邮件（并引用本书的 ISBN 号码 9780596804374）到：

bookquestions@oreilly.com

如果想了解我们的书籍、会议、资源中心以及 O'Reilly Network 的更多信息，请访问网站：

http://www.oreilly.com.cn
http://www.oreilly.com

Safari® Books Online

Safari Books Online 是一个按需出版数字图书馆，让你轻松搜索超过 7500 种技术及有创意的参考书籍和视频，快速找到需要的答案。

通过订阅，你可以从我们的在线图书馆浏览任何网页，观看任何视频；可以在手机和移动设备上阅读书籍；在产品可印刷之前获取新标题，独家访问创作中的手稿并给作者提交反馈；复制并粘贴代码样本，组织你的收藏夹，下载章节，将重点部分加入书签，创建笔记，打印页面，并从许多其他省时的功能中受益。

O'Reilly Media 已经将本书英文版上传至 Safari Books Online 服务。想要得到本书（英文版）的电子版，以及 O'Reilly 和其他出版商类似主题的电子书的完全访问权，请在这里免费注册：

http://my.safaribooksonline.com

致谢

首先，我想感谢所有那些在 O'Reilly 工作的、致力于本书却默默无闻的人们。我甚至不知道你是谁，但感谢你的所有工作，以帮助完成这个项目（并做得很出色），最终使它上架。编辑、图形、布局、规划、市场营销和所有工作，我很感激你们的努力。

朱莉 · 斯蒂尔是我在 O'Reilly 的编辑，在这个项目上总是显示出极大的耐心和敬业精神，感谢你给我这个机会，和从始至终的陪伴。我想现在你是我 IT 出版界的好朋友之一。我希望有一天，可以亲自见到你！

非常感谢我的技术编辑。Charles Tassell、Brian Danchilla 和 Peter Lavin，感谢你们敏锐的目光和彻底测试我很多的示例代码。你们伟大的思想使这本书变得更好。WezFurlong 和 Derick Rethans 也贡献了一些技术帮助，感谢你们的协助。

最后，Susie Sedlacek 是 Zend 公司全球服务前副总裁，感谢你愿意在一起介绍这本书。我十分高兴能有你的介绍，使读者能从全球的角度了解 PHP 在世界范围内的广泛使用和影响。我很高兴听到你和你丈夫已经购买了在加州的葡萄园，我希望你真正享受这一新的尝试！

第1章 精粹

The Good Parts

酝酿这本书有相当长一段时间了。这么多年来，我一直在用 PHP，因为喜欢它易于上手、灵活和功能强大而投入越来越多的感情。在我 20 多年的职业生涯里，PHP 是我用过的所有语言中最喜欢和最拿手的。在此期间，PHP 也从一个小型函数集变成一个体积庞大、模块众多和扩展丰富的工具。有些程序员可能一开始会淹没在它浩如烟海的函数之中，但我希望通过这本书能帮助你真正了解 PHP 的行之有效。这本书并不厚，你将看到 PHP 开发中最精华的部分。当看到本书最后一页时，你会更进一步理解在 Web 开发领域里 PHP 是多么强大。

为什么是PHP

Why PHP?

市场上有那么多编程的书——包括大量关于 PHP 的书——你甚至都不知道也许正有本书即将完成。PHP 是被广泛使用的语言，并且近几年在企业应用方面也有较大的增长。Web 应用如 Facebook[注1]、Flickr[注2]、雅虎的部分网站、维基百科的核心实现，以及网站内容管理系统如 Drupal、Joomla 和 WordPress 也都是采用 PHP 构建的。IBM 还展示了很多自身技术与 PHP 相结合的有趣实例。基于这些原因，帮助技术业界的初中级程序员熟悉这个语言中最精华的部分是件有意义的事情。

PHP历史摘要

A Brief History of PHP

我们简单地回顾一下 PHP 的发展历史。PHP（Personal Home Page）由 Rasmus Lerdorf 在 1995 年发布，最早的名字叫个人主页工具（PHP Tools）。从推出到现在，一直是作为

注 1：社交网站，部分地区访问可能有障碍。——译者注
注 2：图片分享网站。——译者注

2 开源软件出现。数据库操作集成于 1996 的 2.0 版，之后其发展和变化可谓日新月异、翻天覆地。它成为当今世界上使用率最高的网站开发语言。在本书撰写之际，最新版本是 2009 年 6 月 30 日发布的 5.3 版。

PHP的地位

PHP's Place in the World

PHP是一种使用最广泛的编程语言。想想看，在这么短的时间内有如此显著的成长，仅仅15年左右，它已经成为网站开发世界中的一个主要参与者。在最近几年，许多PHP社区的人都在争论它能否适用于企业开发：可信任否？可否用于大项目？够强壮否？鉴于近期也有如IBM和微软这样的公司在关注PHP，而且事实上，它可以构建大型网站（如Facebook和雅虎等），有人认为它也可用于企业开发。这些争论会随着时间的逝去而最终尘埃落定。对于最近发布的5.3版本，你可以十拿九稳地说，行或不行，将很快见分晓。

PHP是什么

What Is PHP?

那么，什么是 PHP 呢？它是一种脚本语言，主要用于服务器端开发，可以被用来动态生成超文本标记语言（HTML）内容。 PHP 和 Web 服务器集成在一起，较常见的是 Apache 或 IIS[注3]，一旦 PHP 完成 HTML 的生成，将交由 Web 服务负责向发起请求的客户端返回结果页面。

我说“主要用于”服务器端，是指你也可以将其用于其他领域，包括命令行、桌面开发和客户端服务环境，我只是举几个例子。但它最常用于 Web 服务器环境。

PHP 开发人员通常会将 PHP 和许多不同的数据库操作工具集成在一起，例如 MySQL、SQLite、PostgreSQL、DB2、MS SQL、Oracle 等。它们使动态内容成为可能。实际上，最终结果页面还是一个静态 HTML 文件，但它是在程序运行中产生的，因此是动态的。其实，你完全可以认为，由于内容是从数据库或其他来源读取并产生的，PHP 实际上是可以产生动态内容的。

PHP有哪些成就

What Has Been Accomplished with PHP?

说了这么多 PHP 的优势，如果没有论据来证明，那显然没说服力。所以，让我们展现几

注 3：这是两种较流行的 Web 服务器软件。——译者注

个用 PHP 搭建和实现的实例吧！世界排名较靠前的网站中有一些是部分基于 PHP 构建的。表 1-1 是使用 PHP 的流行网站简表，包括网站地址和简要介绍。

表1-1 使用PHP的流行网站

网站名称	介绍	网址
Facebook	社交网站	*http://www.facebook.com*
Flickr	照片存储及分享	*http://www.flickr.com*
Wikipedia	在线百科全书	*http://www.wikipedia.org*
SugarCRM	客户关系管理	*http://www.sugarcrm.com*
Dotproject	项目管理工具	*http://www.dotproject.net*
Drupal	网站搭建模板引擎	*http://drupal.org*
Interspire	资讯和邮件营销产品	*http://www.interspire.com*

众所周知，这张表只是冰山一角，它肯定无法完整列出用 PHP 建造的网站，只是举出了几个代表而已。如果你还知道更多这样的网站，那你应该更加清楚 PHP 这门功能强大的语言究竟都能实现什么！

PHP基本设置

Basic PHP Setup

这会儿，你或许跃跃欲试想要亲自体验 PHP 了。因此我们通过一个快速安装来实现一个随时运行都会显示“世界你好”的简单程序。

运行 PHP 代码的基本环境是使用最常用的 Web 服务软件，比如 Apache 和 IIS。有一种包含功能齐全的开发环境的软件包：LAMP 和 WAMP。LAMP 表示 Linux/Apache/MySQL/PHP，但这不是一成不变的，你也可以用 PostgreSQL，而不一定非要是 MySQL 数据库，甚至可以叫它 LAPP。另一副首字母组合——WAMP——是指 Windows/Apache/MySQL/PHP。

通常来讲，你编写的是和操作系统无关的代码。在 Windows 下编写的程序复制到 Linux 一样可以运行良好，反之亦然。但需要注意的是，对于操作系统级别的函数，如 CHMOD（更改文件权限）或 CHOWN（更改文件所有人），类似这样的代码在 Linux 和其他操作系统中可能会出现不同的结果，你一定要在不同环境中充分测试所有的代码实例。

既然有这么多的不同平台和组件可以建立 PHP 开发环境，我们不用再详细讨论这个问题了。总之你要知道，到 *http://www.php.net/downloads.php* 这个地址下载所有平台的 PHP 稳定版本。也有专为 Windows 定制的集多种软件于一身的安装包。一个是 XAMPP（X

表示多种平台，A 是 Apache，M 指 MySQL，第一个 P 表示 PHP，后一个 P 表示 Perl），
4 你可以在 *http://www.apachefriends.org/en/xampp-windows.html* 下载到。当下载合适平台的软件包之后，可以在里面找到一个叫 INSTALL.txt 的文件，其中包含了下载及安装过程指南。

一旦你完成了 PHP 的安装，你就可以通过编译一小段脚本来显示出你的 *php.ini* 配置文件所指定的配置信息，这样的代码只需一行，如下所示：

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

启动和停止位于 <?PHP 和 ?> 之间的 PHP 代码内容的更多方法将在本书稍后几章中分别讲述。现在，将这行代码的文件取名为 *phpinfo.php* 并保存在你的 Web 站点根目录（一般叫 *www* 或 *htdocs*）。当你在浏览器地址栏中输入 *http://localhost/phpinfo.php* 时，输出的页面应如图 1-1 所示。

5

PHP Version 5.2.10

System	Windows NT PETERMAC-PC 6.0 build 6002
Build Date	Aug 6 2009 12:59:12
Configure Command	cscript /nologo configure.js "--disable-ipv6" "--disable-zts" "--enable-fastcgi" "--disable-bcmath" "--disable-calendar" "--disable-odbc" "--disable-tokenizer" "--disable-json" "--disable-xmlreader" "--disable-xmlwriter" "--without-gd" "--without-sqlite" "--without-wddx" "--enable-cli-win32" "--enable-pdo" "--with-openssl" "--with-libxml" "--with-pdo-sqlite"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\Program Files\Zend\ZendServer\etc\php.ini
Scan this dir for additional .ini files	(none)
additional .ini files parsed	(none)
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	disabled
Registered PHP Streams	https, ftps, php, file, data, http, ftp, compress.zlib, compress.bzip2, zip
Registered Stream Socket Transports	tcp, udp, ssl, sslv3, sslv2, tls
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, zlib.*, bzip2.*

图1-1 phpinfo()函数的输入结果

请花一点儿时间来熟悉这些配置信息，如果你不确定其中的大部分信息，也别担心。只要页面显示的内容大致如图 1-1 所示那样 就足以表明 PHP 已经安装正确并通过你本地的 Web 服务器开始运行了。

localhost 是你本地电脑上用于 PHP 运行的 Web 服务器主机名。如果你有程序运行在一个远程服务器，则需要一个适当的域名或特定的 IP 地址来访问它。

现在我们就来写一句问候语。在网站根目录——一般在 Linux 下是 */var/www*，在 Windows 下是 *../apache2/htdocs*——新建一个文本文件叫 HelloOutThere.php，用编辑器打开并输入下面这行代码：

```
<?php echo “Hello, is there anybody out there?” ; ?>
```

在浏览器地址栏中输入 *http://localhost/HelloOutThere.php* ，显示的页面如图 1-2 所示。

6

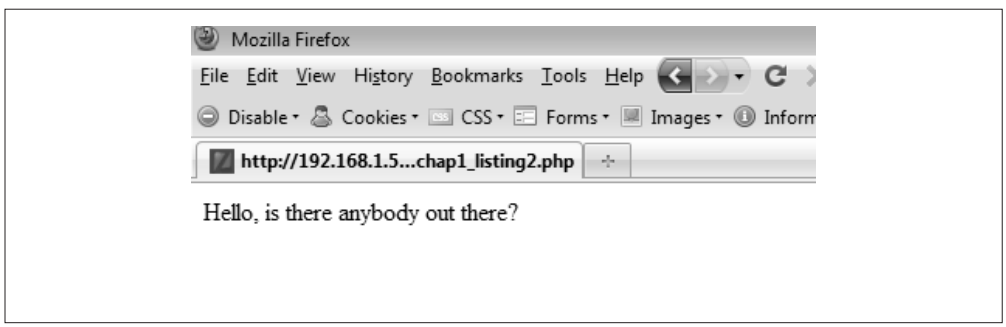


图1-2 实例输出结果

我们通过 echo 命令发送了一个字符串，要求服务器在浏览器的显示区域显示一些内容。在本书稍后几章中你将看到，我们可以在 Web 环境中处理很多东西。你刚刚创建了第一个 PHP 网页，真的就是这么简单。

第2章 “实地勘察”

Casing the Joint

现在，你已经知道了 PHP 文件的基本概念，以及如何通过 Web 服务器运行它并且将运行结果显示在浏览器中。接下来让我们继续研究如何用这门语言像搭积木那样来构建更大、更复杂的网站和网络应用程序。我称这个过程为“实地勘察”，因为为了理解如何从根本上更好地处理 PHP 代码文件，我们将会粗略地看一下 PHP 的环境。如果你之前有过其他语言的编程经历，应该会对本章的学习感到得心应手，你所要做的就是充分理解并知道如何以及在何时使用这些结构元素。一开始，我们只是展示一小段一小段的 PHP 代码（比如变量和各种数据），稍后，我们将讨论如何操作控制语句相关的代码，也叫流程控制。最后，我们将研究可以说明完整的 PHP 应用程序环境的一些概念：内存数据存储（服务器与客户端）以及如何从内存中检索数据。

空白、注释和基本语法

Whitespace, Comments, and Basic Syntax

对于 PHP 而言， 解释器会忽略代码中所有的空白字符， 所有的注释和空行在运行时都会被剔除。

如果你想用去除空白的办法来少许优化 PHP 代码或者想得到简洁的代码文件，其实不用花太多时间在整理代码格式上，PHP 中有个函数叫 `php_strip_witespase`，专门用来干这个。给这个函数传递一个文件名，它会返回去除所有注释和空行的干净代码，然后你就可以保存一个整洁的 PHP 代码文件。

看看下面这段代码，其中包含一些空白的部分：

```
1<?php
2 # 这是一行 PHP 注释
3
```

```
4 /*
5 * 这是多行 PHP
6 * 注释语句
7 */
8
9 echo " Hello my good web browser "; // 这是内联注释语句
10
11
12
13 ?>
```

第 1 行允许有空白，如果你在 PHP 开始标记（即 <?php）的后面添加空格（按空格键），将被视为空白。

第 2 行是个注释行，所以整行内容都将被视为空白。注释行表示代码注解，因而不会被解释器执行。第 4 行至第 7 行也是注释，但稍有不同，叫做多行注释，你能看到它以 /* 开始并以 */ 结束。PHP 解释器遇到这四行时会把它当成不需要执行的部分，即将其当成空白而完全跳过。

第 9 行会被执行，此行后面以 // 开头的部分是一个内联注释。PHP 解释器会对以 // 开头的部分视而不见。事实上，你完全可以把这样的注释加在语句末尾和分号之间，只不过这会让代码变得很难读懂（分号表示此行命令结束，如果漏加会报一个语法错误——即便你是一个经验老道的高手偶尔也会犯这样的错误）。

第 3 行、第 8 行和第 10~12 行的空行既不是注释也不是可运行代码。如果你清除所有的空白内容，那么上述代码就变成下面这个样子：

```
1<?php echo "Hello my good web browser"; ?>
```

正如你所看到的，在不同的命令或函数之间还有一个或多个的空格，和恰当的注释能帮助理解代码的道理一样，这也是为了使程序更易于阅读，毕竟你写出的代码要让自己或别人读得懂，所以，一定要养成在代码中适当留白的习惯。下面任意一种方法都可以在代码中添加注释：

#

用它来表示行内注释，此行内不允许再有任何可运行指令。

//

用它也表示行内注释，它可以自起一行，也可以附加在可执行代码的末尾。

9 /* ... */

表示多行注释块，块里的任何指令都不会被运行。

那么，一个基本的 PHP 语法包括一个 PHP 开始指示标签［即 <?php 或 <?］加上一个结束指示标签［即 ?>］。指示标签中间这部分内容会被 Web 服务器交给 PHP 解释器处理。

那种短一些的开始标签 <? 在 PHP 5 以后的版本中默认不允许使用，除非在 php.ini 的配置中将 short_open_tag 的值改为 On。这个约定是为了让人们尽可能使用更好且更完整的开始标签 <? php。

在这样一对标签里面[注1]，就可以开始 PHP 编程工作了。你可以使用四种不同的语言结构来编程：像前面 echo 那样的指令语句、函数调用（PHP 库里的或你自己编写的函数）、流程控制语句（如 if...else...）和注释。几乎所有的 PHP 程序都是由这四种简单结构组成的，当然了，一个完整的 Web 应用程序会大量使用这些语句，另外，PHP 还可以定义面向对象的类（详见本书第 6 章）。

为了构建更加健全的应用程序，往往还需要其他的元素，例如变量，下面我们就来看看变量及其使用规则。

变量：数据类型、弱类型和作用域

Variables: Data Types, Loose Typing, and Scope

变量可以用来表示不同类型的数据，但它们建立的方式是一样的。下面就来说说 PHP 变量的定义规则：

$
变量名一律以美元符号开头（$）。

大小写敏感
变量名对大小写敏感，所以 $firstname 和 $FirstName 表示两个完全不同的变量。

字母或下画线
在美元符号 $ 之后的第一个字符必须是字母或者下画线（_）；其余的字符可以是字母、数字和下画线。

$this
除了在面向对象的 PHP 中，别处不会遇到使用 $this 的情况。

数据类型顾名思义就是指某种类型的数据。不同类型的数据在结构、可交互性及操作方式上都有不同的限制和约定。PHP 里有 8 种基本的（或叫原始的）变量类型。当然也可以自定义类型，但在这本书里只涉及 8 种。这些原始类型按数据分段存储方式又可分为

10

注 1：如果结束标签后无输出则可以省略。——译者注

三大类：标量类型、复合类型和特殊类型。表 2-1 列举了变量类型和分段方式及部分示例。

表2-1 PHP数据类型

分段	类型	解释 / 示例
标量类型	布尔型	逻辑“真”或“假”
	整型	全部整数：例如 1、15、–122 和 967967
	浮点型（双精度）	有小数点的数字（在金融财会领域常见）：例如 12.56 和 345.456
	字符型	各类文字、字母和数字（通常用双引号括起来）：例如“你好”和“123AvR”
复合类型	数组	包含键值对的集合，数组中可再包含数组（多维数组）；详见第 5 章
	对象	面向对象编程中已定义类的实例；详见第 6 章
特殊类型	NULL	未赋值的变量；已存在的变量，但不包含任何数据（不是空字符也不是 0，而是什么也没有）
	资源	表示对函数、数据库资源、文件或其他 PHP 外部资源的引用指针

有两种方式给变量赋值：传值和传引用。一般是通过直接传值来定义变量，例如：$firstname = "Peter"，我们就给这个叫 $firstname 的变量分配了一个包含 5 个字符的字符串，这个变量将一直保留其内容，直到重新分配或脚本运行完毕。没有什么因素能影响这个变量的内容，除非程序直接和它发生交互。

而传引用就好比是用不同的变量名来表示同样的变量内容，通过这种方式可以让函数在其内部影响外部定义的变量。只有事先以传值方式定义过的变量才能以引用的方式传递和访问。当然，只要引用一次即始终是引用方式。如果被引用传递的变量内容有变化，则所有本地副本所指向的引用会自动变更为新内容。实现一个引用传递很简单，只要在目标变量名前面加个 & 符号即可。下面的代码即可演示这个效果：

```
<?php
$firstname = "Peter" ;       // 分配值给变量
$fname = &$firstname ;       // $fname 是对 $firstname 的引用
echo $fname . "<br/>";       // 显示内容 Peter
$fname = "Dawn";             // 更改引用变量的内容
echo $firstname . "<br/>";   // 显示为 Dawn 而不是 Peter,
                    // 因为是“引用”
?>
```

11 和其他编程语言不同的是，PHP 是个弱类型和动态类型语言。也就是说，PHP 可以很聪明地识别出分配变量时其存储的数据类型，像日期、字符串和数字等。因此在前面进行赋值的例子中：$firstname = "Peter"，PHP 可以判别 $firstname 为字符串类型。然而

预先定义各个变量的类型确实也是一种很好的实践，这种方法可以减少混淆。

变量的作用域涉及变量在一段代码中被另一段代码发现并操作的问题。在默认情况下，一个变量是在整个 PHP 代码中（整个 PHP 文件范围）可见的，但也有例外。如果一段代码被某个函数引用或包含之后，则这段代码就不能访问此函数外部的任何部分变量。下面我们通过具体实例说明一下：

```
<?php
    function show_stuff() {
        $secondName = "Beck" ;
        echo "inside show_stuff: " . $firstname . " " . $secondName ;
     }

    $firstname = "Peter" ;      // 除了函数内部，此变量全局可见
                                //
    echo $firstname  . "<br/>"; // 显示 Peter
    show_stuff () ;             // 只会显示 Beck 因为 $firstname
                                // 不在函数的作用域里
    echo "Outside function " . $secondName ;
                                // 仅仅是在函数 show_stuff 中有定义，
                                // 所以此处无法访问，
                                // 这里不会显示它的内容
?>
```

你也看到了，同样都叫 $firstname，但因为是在不同的作用域里，所以 show_stuff 函数不能访问到外部的变量，而它定义的 $secondName 也同样不能被外部访问。有关函数及如何改变这些行为的更多详情，请参阅第 3 章。现在，你只要记住：由于作用域的原因，有些代码可以自然地访问某些变量，而换个地方就不能访问。

定义常量

Defined Constants

定义的常量就像是 PHP 变量的近亲。常量可以定义在代码文件的任意位置，但大多数人都是在代码文件和函数的开头去定义它。一旦定义之后会保持到代码运行结束，而且是全局有效，也就是说无论在函数或类里，甚至在任意包含的文件及函数里，常量都是可以全局访问的。定义一个常量的方法很简单，有点像分配变量，但又不完全一样。最大的差别是常量定义需要使用一个 PHP 内置的函数 define()。定义常量时，需要遵循以下规则：

define()

12

要用这个 PHP 内置函数来定义常量。

字母或下画线

常量名称必须以字母或下画线开头，首字符以后的部分也只能由字母、数字或下画线组成。

大小写敏感

按照默认的约定，常量名应全部大写，当然你也可以不遵守这个约定，用 define() 函数中的第三个参数来定义大小写不敏感的常量。

限制

只有标量数据（参考“变量:数据类型、弱类型和作用域”部分）才能定义在常量里。所以，定义一个常量的语法如下所示：

```
define(" 常量名称 ", 常量值 , [ 是否大小写不敏感 ])
```

此函数中最后一个参数大小写不敏感是可选项，默认为假，即定义的常量名称大小写敏感（这是个众所周知的标准规范）。获取一个已定义常量的值，只要引用常量名称即可。下面的代码将向您演示定义两个常量以及试图重复定义常量的问题：

```
define("SYS_OWNER", "Peter");
define("SYS_MGR", "Simon", true);
echo "System owner is:" . SYS_OWNER . "<br/>" ;
define("SYS_OWNER", "Michael");
echo "System owner is:" . SYS_OWNER . "<br/>" ;
echo "System manager is:" . SYS_MGR . "<br/>" ;
echo "System manager is:" . SYS_mgr . "<br/>" ;
```

上述代码输出（包含一个错误警告）如图 2-1 所示。

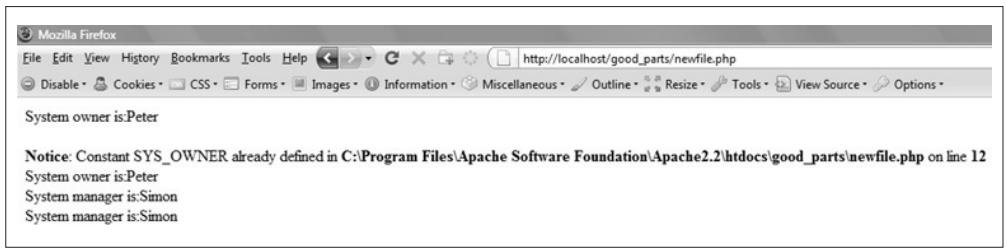
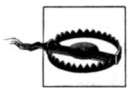


图2-1 定义常量示例代码的输出



如果 PHP 配置中错误报告被关闭，你可能看不到图 2-1 中的警告信息，可能会有意外的或不想要的结果，所以，你在将代码上传到产品环境以前，要确保代码经过测试。

毫无疑问，定义常量在 PHP 编程实践中占有一席之地；常量的价值是显而易见的，尤其是当你需要实现一个在代码进程中不被改变的值的时候，如一个存储 PDF 文件的路径、计算需要的税率等。当你确实有这样的应用场景时，你会考虑用常量。但是一定要确保经过测试再上传到生产环境，只有这样你才能实现预期的效果。

表达式

Expressions

PHP 中的表达式（不是正则表达式，那个是特例）是代码语句的统称。

```
$name = "Peter";
```

这行代码是个赋值表达式，它描述的意思是：分配一个字符串“*Peter*”给一个名叫 `$name` 的变量。从技术上讲，这行代码是一个由两部分表达式组成的声明（以分号为结束符）：左边的部分是一个存储的定义，右边的部分是要将什么值分配给那个存储定义。这两部分组成了一个赋值表达式，所以是个完整的声明。

作为一般规则，任何一个赋值语句的表达式，都是作为指令语句处理。

另外一些表达式包括函数和条件三元运算符（相关说明请见下一节条件判断代码），函数的返回值也包括在内。例如下面的代码就演示了这两种表达式：

```
function MyName () {
Return "Peter" ;
}
$name = MyName();
$name ? $last = "MacIntyre" : $last = "" ;
```

函数 `MyName` 被当做一个表达式，因为它返回了一个值。你可能会觉得在赋值语句后面的那行代码有些奇怪，在以 `$name` 作为条件的前提下给 `$last` 赋值。我们将在本书中看到更多这样的表达式，现在你只要知道有这种语句存在并且在 PHP 中非常普遍就行了。

判断、选择（流程控制）

Decisions, Decisions (Flow Control)

如果没有选择项就没有其他的可能性，人生将是多么枯燥无趣啊！下面我们来看看如何用流程控制语句来进行条件判断，也就是用它来根据预先规定的条件做出选择。

14 If...Else...

If...Else...

基本 `if` 语句的测试目标可以建立在一个变量或执行其他简单（有时也会复杂）语句的运行结果的基础上。举一个使用了 `if` 语句的例子：

```
$today = date("l");
if ($today == "Wednesday") $tax_rate = $tax_rate + 4 ;
```

这里的比较判断使用的是双等号（看看是不是星期三）。我们已经知道赋值语句用的是一个等号，而判断是否相等要用双等号。

如果你需要进行数据类型级别的相等判断，可以使用 `===`，它会对比两边所有元素的内容及其类型，考虑下面的代码：

```
if (1 == '1') echo "true 1 等于 '1' <br/>";
if (1 === '1') echo "true 1 等于 '1'";
else echo "false 1 不等于 '1' " ;
```

它会产生下面的输出，显然，当用双等号比较字符串和数字时，字符串会先被转换成数字类型再进行比较，因此结果为真（true）。当用 `===`（三个等号）进行比较时，字符串不会被转换，它和数字类型不一致。上述代码的输出结果如下所示：

```
true 1 等于 ’ 1’
false 1 不等于 ’ 1’
```

在 `if` 语句中可以使用有效格式的数字。在前面例子的比较判断中，如果为真，则税率（变量 `$tax_rate`）加上 4，也就是说除了星期三以外的日子，税率保持不变。如果这个比较判断的结果为真你还要加入其他的语句，可以使用一对花括号。下面的代码是对前面例子的扩展，其中对日期判断进行了简化，去掉了变量赋值语句。

```
if (date("l") == "Wednesday") {
$tax_rate = $tax_rate + 4;
$wages = $salary * 0.4;
$msg_color = "red";
}
```

该例实现的是如果判断条件为真则执行三条语句。请注意，这三条语句都包含在 if 后面的大括号里面，表明它们是这个判断结果中同等重要的部分。

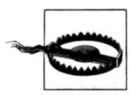
你也可以添加当 if 语句中的条件不被满足时需要执行的代码，这个代码叫 else 子句，如下所示：


```
if (date("l") == "Wednesday") {
    $tax_rate = $tax_rate + 4 ;
    $wages = $salary * 0.4 ;
$msg_color = "red" ;
} else {
$tax_rate = $tax_rate + 2 ;
$wages = $salary * 1.4 ;
$msg_color = "blue" ;
}
```

如果 if 语句中的条件判断结果为假，则执行 else 后面花括号里的语句——具体这个例子来说，就是除周三以外的日期会执行这些代码。

你甚至可以在 if 语句中再嵌套别的条件判断语句。例如，如果今天是 6 月份的某个周三，进行判断的代码可以这样写：

```
if (date("l") == "Wednesday") {
$tax_rate = $tax_rate + 4 ;
$wages = $salary * 0.4 ;
$msg_color = "red" ;
if (date("F") == "June") {
    $discount = $tax_rate * 0.15 ;
}
}
```



嵌套会使代码显得笨重，为了防止出现晦涩难懂甚至无法运行的代码，你要慎用多层嵌套并控制嵌套层数。

下一步，可以用 elseif 子句来实现多层条件判断。举例来说，如果你需要进一步对一周的日期进行判断，以实现每天执行不同的程序任务，则会写出如下所示的代码：

```
$weekday = date("l") ;

$tax_rate = 4 ;

if ($weekday == "Monday") {
$discount = $tax_rate * 0.05 ;
} elseif ($weekday == "Tuesday") {
$discount = $tax_rate * 0.06 ;
} elseif ($weekday == "Wednesday") {
```

```
$discount = $tax_rate * 0.07 ;
} elseif ($weekday == "Thursday") {
$discount = $tax_rate * 0.08 ;
} elseif ($weekday == "Friday") {
$discount = $tax_rate * 0.09 ;
} elseif ($weekday == "Saturday" || $weekday == "Sunday") {
$discount = $tax_rate * 0.10 ;
}
echo $weekday . "'s discount is: " . $discount ;
```


如果你在周四运行这段代码，会输出以下结果：

```
Thursday's discount is: 0.32
```

另外，注意在示例代码中使用了“或”（||）条件来判断周六和周日。

另一种实现条件判断的写法是使用三元运算符。这种格式虽然不使用明确的 `if/else` 语句，不过一旦你掌握了它，就可以写出很简洁的代码。暂时忘掉 if 语句，我们用这种方法来实现一个简单的税率判断操作，考虑下面的代码：

```
$tax_rate += date('l') == 'Wednesday' ? 4 : 2;
```

这行代码完整的意思是这样：如果当天是周三（?），则给税率变量加上 4；如果不是（else,:），则只加 2。

你可能想了解例子中的 += 是怎么回事。PHP 允许在对数字变量做简单数学运算的同时做赋值操作[注2]，像 ++、--、+= 等都属于这一类特性。要知道如何更好地利用这个特性，可具体查看 *php.net* 中的内容。

像这样的三元运算符通常仅限于处理多个条件判断中最终的一个结果（真或假），虽然也允许使用多层嵌套，但可能不会达到预期的目的，所以，最好保持这种简单直接的使用方式。

还有另外一些 `if` 语句的写法，它们也会运转得很好，但我们不用再探究更多的细枝末节。如果你有兴趣了解那些格式，请移步 *php.net*。

Switch...Case...

Switch...Case...

针对单个值进行数个条件的判断，使用 `if` 语句来实现仍存在一定局限。在前面关于一周日期判断的例子中，只要我们愿意就可以让代码在 7 天里做 7 件不同的事。就这点来说，如果那天正好是周三，则只要在代码里定义一个特定的条件即可。诚然，如果我们针对

注 2：多数流行编程语言都有类似支持。——译者注

一周的每天都处理不同的税率，则可以写出类似这样的代码：

```
$today = date("l") ;
if ($today == "Monday")     { $tax_rate += 2 ; }
if ($today == "Tuesday")    { $tax_rate += 3 ; }
if ($today == "Wednesday")  { $tax_rate += 4; }
if ($today == "Thursday")   { $tax_rate += 5 ; }
if ($today == "Friday")     { $tax_rate += 6 ; }
if ($today == "Saturday")   { $tax_rate += 7 ; }
if ($today == "Sunday")     { $tax_rate += 8; }
```


但如果需要在每天的条件判断中执行更多的语句，这样的代码将会变得更加复杂和难以管理。这里我们引入 switch...case... 语句，你可以用这个控制结构操作更多种可能值的情况。语法如下：

```
switch ( 用来判断的值 ) {
case 第一个可能的值 :
    // 一些代码
    [break;]
case 第二个可能的值 :
    // 一些代码
    [break;]
default:
    // 如果没有任何一个前面的条件为 true，那么执行到这里
}
```

这个语法看起来有点儿令人难以理解，不过一旦你了解它的逻辑，你会觉得它确实很有用。括号内的值是用来判断的项，之后是对每种值的情况进行判断处理，最后，如果有其他可能，会放在 default 子句中处理。

break 语句是可选的，但如果不写上它，PHP 会在处理完某个值之后继续执行 case 语句的下一种可能，因此，要在 break 起作用的地方小心使用它。

好了，对一周中的每一天进行处理的例子，最终就会变成这个样子：

```
switch ($today) {
    case "Monday" :
        $tax_rate += 2 ;
        $wages = $salary * 0.2 ;
        $msg_color = "red" ;
        break;
    case "Tuesday" :
        $tax_rate += 3 ;
        $wages = $salary * 0.3 ;
        $msg_color = "yellow" ;
        break;
```

```
    case "Wednesday" :
        $tax_rate += 4 ;
        $wages = $salary * 0.4 ;
        $msg_color = "black" ;
        break;
    case "Thursday" :
        $tax_rate += 5 ;
        $wages = $salary * 0.5 ;
        $msg_color = "green" ;
        break;
    case "Friday" :
        $tax_rate += 6 ;
        $wages = $salary * 0.6 ;
        $msg_color = "orange" ;
        break;
    case "Saturday" :
    case "Sunday" :
        $tax_rate += 7 ;
        $wages = $salary * 0.7 ;
        $msg_color = "purple" ;
        break;
    }
```


这里不必加上 default 子句，因为我们要判断所有可能的值。但请注意，周六和周日之间没有 `break` 语句，也就是说周末两天的税率处理过程是一样的。这只是一个例子，说明在需要的时候你可以去除 `break` 语句。

正如你看到的，`switch...case...` 结构有它的优点，虽说同样是条件控制语句，但这种结构能使代码更容易阅读和修改。

While...

While...

现在让我们来看看 `while` 语句。这个语句能让一段代码重复运行，直到条件不为真时停止。语法上有两种基本形式。首先是直接以 `while` 开头的形式，像这样 ：

```
$repeat = 1;
while ($repeat <= 25) {
    echo "the counter is: " . $repeat . "<br />";
    $repeat ++;
}
```

另一种语法形式是 `do...while`，如下所示 ：

```
$repeat = 0;
```

```
do {
    $repeat ++;
       echo "the counter is: " . $repeat . "<br />";
} while ($repeat <= 25);
```

这两种形式的主要不同点在于 do...while... 结构会至少执行代码一次。通过对条件进行判断来决定是否循环。在第一个例子里，有可能永远也不会执行条件后的代码。比如，假设第一行代码中 $repeat 的值不是 1，而是 27，会使条件判断结果为假，那么 while 循环将永远不会被执行。

要小心使用第二种语法（do...while...），因为你写的代码会至少执行一次，这可能是也可能不是你要达到的目的。

For

For

for 循环逻辑结构和 while 结构系列在逻辑实现上有一些不同，你可能已经注意到，在上一节的例子中，用于控制计数器的变量 $repeat 必须手动以 $repeat ++ 命令实现递增。如果使用 for 循环结构，计数器会直接写在条件判断行里，这样就会使代码稍微简洁一些。前面的 while 例子，下面我们用 for 循环结构来实现，殊途同归，效果完全一样：

```
for ($i = 0; $i <= 25; $i++) {
    echo "the counter is: " . $i . "<br />" ;
}
```

这个语句的第一部分（$i = 0）是设置 for 循环的初始值。分号后面的部分是判断条件，在每次循环时都会由这个判断决定是否继续执行循环语句。最后部分是定义每次循环时如何对条件变量进行改动，当然必须是在逻辑上有效的值。例如，它可以每次递增 5 或递减 12，这取决于编写代码的需要。用这个结构，我们把四行代码变成了两行，缩减了 50%呢！

为了指出这是一个迭代，我还把变量变从 $repeat 改成了 $i ——这是大家常用于循环操作的变量名。继续使用 $repeat 也行，这里只是为了更紧凑才使用简短一点儿变量名。

在本书第 3 章中我们会讨论 include/require 结构，在第 5 章中会接触到 foreach 语句。另外，一定要重点学习第 10 章中提到的新控制结构 goto 语句。

Web页面交互

Integration with Web Pages

PHP 的一个最大特点是可以帮你在集成 Web 服务器的环境下生成 HTML，不管是在 Apache 还是 IIS 或者是其他知名的 Web 服务器软件里。本节我们将看看 PHP 在处理请求时如何与 Web 服务器发生交互以及读写数据。图 2-2 表示的是 Web 服务器和 PHP 配置的基础结构。

这是一张 Web 服务和 PHP 适当组合的简单视图。当今的网站通过应用 JavaScript 和 Ajax 以及在其他技术的帮助下，已经表现得极具互动性。但实际上，网站依然是无状态的，也就是说一个网页做了什么并不会和别的网页发生必然关系。它们是彼此完全独立、隔绝的和无法直接交互的。然而，你可以将部分信息存储到服务器端的相应位置，然后在稍后的处理中取出这段信息，只要这段数据还未过期。

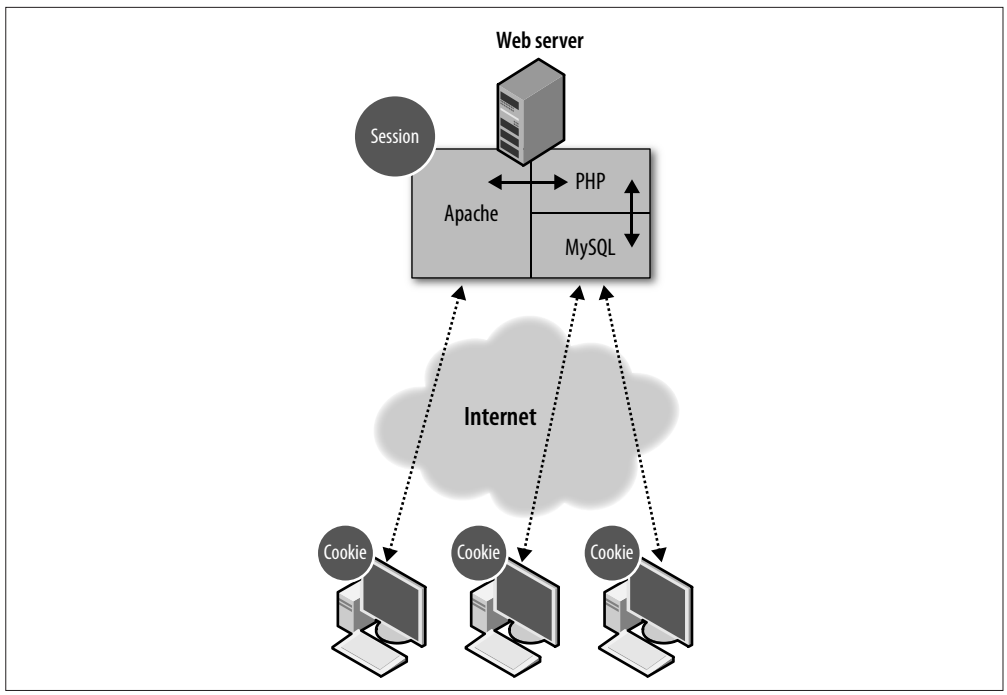


图2-2 Web服务器和PHP配置基础结构示意图

如图 2-2 所示，网页之间或 PHP 代码文件之间用来交互的变量数据可以存储在两个主要的地方：客户端的 Cookie 或服务器端的 Session。PHP 尤其擅长处理这两个存储区域，

接下来的两节会详细解释它们的优势和不足。之后我们将在这些概念的基础上讨论如何利用好这些优势并发挥它们的作用。

客户端Cookie

Cookies

我不是 Cookie 的超级粉丝。我很少使用它们，但它们也有自己的作用。每个网站的 Cookie 只是 Web 服务器存储在客户端电脑硬盘上的一个很小的文件。Cookie 有名字（用来标识）和表示的值，也有作为可选项的过期时间、路径和安全设置。下面的代码定义了两个 Cookie：

```
$data = "this will be placed in the cookie" ;
setcookie("CookieName", $data) ;
setcookie("AnotherCookieName", $data, time()+60*60*24*30) ;
```

第一个 `setcookie` 指令会在客户端电脑中建立一个名为“CookieName”的 Cookie，并将 `$data` 变量的内容存储其中。由于未设置过期时间，这个 Cookie 会持续有效直到 Web 会话结束（一般为浏览器关闭）。第二个 Cookie 定义（名为 AnotherCookieName 那行）：和前一个例子一样，除了也有自己的名称和值，它还有个过期时间。这个过期时间的值基于 UNIX 的时间戳，所以这里你应该使用 `time()` 或 `mktime()` 函数来产生时间戳值。

21

由于 Cookie 是存储在客户端电脑上的，触发过期的条件是基于客户电脑的时间设置，而不是 Web 服务器上的时间。

上面的例子只是在客户电脑上设置 Cookie，另一个需要了解的是如何在你的代码或网站的其他页面中读取这些数据。本书中会出现 PHP 一系列的系统级变量，它们统称为超全局变量。这个顾名思义，就是在 PHP 的所有作用域中，不论是一般脚本、函数、类或者是其他外部包含的脚本中，它们通通都是可用的。第一个将要出场的就是马上要用来读取 Cookie 数据的超全局变量。这里我们用代码来实现读取前面定义的名为 `AnotherCookieName` 的 Cookie 值：

```
$newData = $_COOKIE["AnotherCookieName"] ;
```

超全局变量很好识别，它们总是由 `$_` 开头，并且其余名称部分的字母全部大写。在这个例子里，`$_COOKIE` 提供了一个以名称 `AnotherCookieName` 来访问 Cookie 数组中的值的形式。只要这个 Cookie 存在（还未过期），就可以在任何需要用到的地方读取它。

当你建立 Cookie 时，最好象征性地设置一个过期时间，或者将其设置为 0，它会在浏览器关闭时过期。你也可以设置成一个过去的时间，这样它会被完全删除（而不是设置一个日期等它将来过期）。删除 Cookie 是在客户端电脑上进行的，从某种意义上说，作为一个不速之客，正确的事情是自己清理好自己留下的垃圾。

Sessions

Session

除了 Cookie 之外，另一个选择便是 Session。Session 本质上讲和 Cookie 类似，差别只在于它驻留在服务器端而不是客户端。通常来讲，其优势是 Session 不依赖于客户端电脑的资源和配置[注3]，因此它在开发上往往会给你或其他程序员稍多一点控制。每

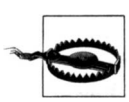
个 Session 都是存储在服务器设定目录中的唯一文件，目录的位置由 `php.ini` 中的 `session.save_path` 控制。php.ini 中有相当多和 Session 有关的配置选项，要想尽可能高效地管理这部分配置内容，你一定要先花一些时间了解它们。下面是一个 Session 文件名的例子：

```
sess_p6lhj0ih6hte5ar8kqmge629a6
```

文件名以“sess_”开头，然后是字母和数字的随机组合。这能让 PHP 在服务器上实现跟踪会话实例。Session 有效期内所有键 / 值对数据组合成一个关联数组，经过序列化后存储在该文件的内部结构中，一般来讲，这个有效期会持续到客户端浏览器关闭。要开始一个 Session，无论怎样，你都要使用 `session_start` 函数。如果是在一段时期内第一次调用这个函数，它会在服务器上存储一个空的关联数组；否则，会重新打开即有的 Session 文件并使数据可以访问。下面的示例是启动一个 Session 并保存一个值：

```
session_start( );
$today = date("Y-m-d") ; // 用 YYYY-MM-DD 的格式返回今天的日期
$_SESSION['today'] = $today ; // 添加这个日期到 Session
$_SESSION['login_name'] = "Peter" ; // 添加这个登录名到 Session
```

一旦 Session 建立完成，PHP 就可以通过它来处理浏览器生命期内文件运行环境的有效连接。



最新的浏览器都具备多标签浏览功能，允许在不同的标签内浏览不同的网址。如果在不同的标签内打开同一个网址，PHP 会认为它是同一个访问者，这样 Session 的内容就会在同一标签中共享，但这可能会导致一些意想不到的后果和行为。

如果在浏览器有效期内访问随后的其他页面，你可能想要读取即有 Session 的信息，你需要做的就是再次启动 Session，访问你感兴趣的数组键，像这样：

注 3：Session 多数情况下需要存储一个 Cookie 来支持。——译者注

```
session_start () ;
$loginName = $_SESSION['login_name'] ;
echo $loginName . " is now logged in" ;
```

这个名叫 $_SESSION 的超全局数组其实就是一个数据中转站，借助它数据将会在网站的不同页面之间穿插。作为一个程序员，你会发现 Session 技术会比 Cookie 技术提供给你更多的可操控性。因为你可能恰好不擅长处理客户端浏览器编程环境。

$_GET

$_GET

下一个要讨论的超全局变量是 $_GET。这个值是由通过 URL 访问时的查询字符串而自动建立的，也可以通过 GET 方法的表单提交（仍然是使用 URL 作为媒介）来产生。被调用的页面运行时会将随查询字符串发送来的任何键 / 值对信息附加在 $_GET 数组变量里。

http://www.mynewwebsite.com/access.php?login=1&logname="Fred"

这个网址的查询字符串中有两个键，一个叫 login，另一个叫 logname。当 access.php 这个文件被调用时，你就可以根据需要操作这些值。通过键名来访问这个关联数组即可得到其值。考虑一下 *access.php* 文件中以下部分代码：

```
$login = $_GET['login'] ;
$logname = $_GET['logname'] ;
if ($login == 1) {
echo "Welcome " . $logname ;
} else {
echo "login failed... please try again " . $logname ;
}
```

用 $_GET 这个超全局变量的好处是，你可以在一个被调用的文件中使用指定页面的请求参数信息。

理解 $_GET 在每个页面调用时都会被刷新这一点至关重要，也正因为如此你必须把它传递到每一页进行向下调用的堆栈中。这一点和 Session 的概念有所不同。

$_POST

$_POST

$_POST 超全局数组在有效地从一个页面到另一个页面传递数值方面几乎和 $_GET 完全一样，所不同是传递信息的方法。$_POST 不使用被调用页面中 URL 查询字符串的办法，

而是对服务器使用超文本传输协议（HTTP）中的 POST 方法（提交数据）。普遍来讲最常见的形式是在网页中用表单提交，但也不是非要依赖于 HTML <form> 标签才能实现。另外，由于它要向服务器 POST 数据，提交的信息也不必作为 URL 的一部分，对用户来说，可见信息更少一些，这或多或少会提升一些安全性。下面的代码演示如何在一个简单的网页中使用 <form> 标签以及如何在 PHP 文件中操作 $_POST 数组。

```
<html>
<head></head>
<body>
<form method='post' action='page2.php'>
please enter your name: <input type="text" size="15" name="fullname">
<input type=submit value="submit">
</form>
</body>
</html>
```

24 当用户单击“提交”按钮时会调用 *page2.php* 文件，其代码如下所示：

```
<?php
$name = $_POST['fullname'] ;
echo "the full name: " . $name ;
?>
<br/>
<a href="demo.php">back</a>
```

$_REQUEST

$_REQUEST

本章中要讨论的最后一个超全局数组是 REQUEST 数组，这是一个包含各类请求数据的数组，即 $_COOKIE、$_GET 和 $_POST。美中不足的是，数组中键名必须唯一，否则对于 $_REQUEST['lname'] 不能确定是来自前三者中的哪一个。看看下面的代码：

```
<html>
<head></head>
<body>
<?phpsetcookie('mname', 'Beck') ;?>
<form method='post' action='page2.php?fname=peter&lname=smith'>
<input type="hidden" value="Simon" name="fname">
please enter your last name: <input type="text" size="15" name="lname">
<input type=submit value="submit">
</form>
</body>
</html>
```

在上述代码中，设置了一个 Cookie，通过 POST 方法提交表单，并且在表单提交时，通

过 URL 的 GET 方法发送了一些数据。然而在实际的代码中出现这种情况的几率并不大，但它表明在使用 REQUEST 数据时有可能会出现的意外情况。<form> 标签中的 action 属性指向 *page2.php*。下面是该文件的代码 :

```
<?php
$fname = $_GET['fname'] ;
$lname = $_GET['lname'] ;
echo "the full name from GET: " . $fname . " " . $lname ;
$fname = $_POST['fname'] ;
$lname = $_POST['lname'] ;
echo "<br/>the full name from POST: " . $fname . " " . $lname ;
echo "<br/> the Request array -> " ;
var_dump($_REQUEST) ;
?>
<br/>
<a href="demo.php">back</a>
```

假设用户在前面表单中的 lname 输入框中写的是“MacIntyre”，当这个页面显示时，能看到下面的文字 :

```
the full name from GET: peter smith
the full name from POST: Simon MacIntyre
the Request array -> array(3) { ["fname"]=> string(5) “Simon” ["lname"]=>
string(9) “MacIntyre” ["mname"]=> string(4) “Beck” }
back
```


正如你看到的那样，尽管我们对 GET 和 POST 设置了不同的值，但它们有相同的键名，所以 REQUEST 数组默认按照顺序返回 POST 数据中的值。另外要注意的是，我们没有明确地引用前面代码中所设置的 Cookie 值——它在以后请求和调用 PHP 文件时会自动附加到 REQUEST 数组。

你可以通过 php.ini 中的配置指令 variables_order 来控制整个环境中的超全局数组。我的服务器上配置的是 GPC，即 PHP 在加载后依次建立的超全局数组是 GET、POST 和 COOKIE。就如同它们开头字母一样，排在后面的要优先于前面的。“G”代表 GET，“P”代表 POST，“C”代表 COOKIE。如果你删除了其中的一个字母，保存后重启你的 Web 服务器，则那个字母所代表的超全局变量就不会自动建立。