HZ BOOKS
华章科技

sitepoint

**资深PHP技术专家多年工作经验结晶，能指引进阶修炼的PHP工程师少走弯路**

从面向对象、数据库、API、设计模式、安全性、应用程序性能、自动化测试、质量保证等多方面总结了编写高效PHP代码的最佳实践

华章程序员书库

PHP Master: Write Cutting-Edge Code

# PHP精粹

## 编写高效PHP代码

（美）Lorna Mitchell Davey Shafik Matthew Turland 著

彭冲 胡琳 译

机械工业出版社
China Machine Press

华章程序员书库

PHP Master: Write Cutting-Edge Code

# PHP精粹
# 编写高效PHP代码

（美）Lorna Mitchell Davey Shafik Matthew Turland 著
彭冲 胡琳 译

机械工业出版社
China Machine Press

本书是资深 PHP 技术专家多年工作经验的结晶，从数据库、API、设计模式、安全性、应用程序性能、自动化测试、质量保证等核心方面总结了编写高效 PHP 代码的技巧和最佳实践，旨在让有一定基础的 PHP 开发者在进阶修炼的路上尽可能少走弯路！全书包含大量精心设计的示例，不仅能帮助读者理解具体的技术知识，而且能让读者学到作者解决各种问题的思路，授人以鱼同时授人以渔。

本书共 8 章，每章一个主题：第 1 章重新阐述了面向对象编程中的核心概念和技术，目的是确保基础知识匮乏的开发者能正确理解它们；第 2 章总结了 PHP 开发中与数据库相关的各种最佳实践，如数据持久化、数据存储、MySQL 使用方法、PDO，以及数据库的设计等；第 3 章详细讲解了 API 及其使用方式；第 4 章总结了 PHP 开发中常用的各种设计模式及其使用原则；第 5 章讲解了如何编写安全的 PHP 代码，对 PHP 开发中各种常见的安全问题进行了总结和分析；第 6 章从基准测试、系统测试、数据库、文件系统等方面探讨了 PHP 应用程序的性能问题；第 7 章讲解了 PHP 的自动化测试，包含单元测试、数据库测试、负载均衡测试等；第 8 章总结了 PHP 开发中与质量保证相关的最佳实践，包括质量测量、编码标准、源代码管理、自动部署等。除此之外，本书还对 PEAR、PECL，以及 PHP 标准库进行了讲解。



**本书版权登记号：图字：01-2012-6630**

**图书在版编目（CIP）数据**

PHP 精粹：编写高效 PHP 代码 /（美）米切尔（Mitchell, L ），沙非克（Shafik, D.），蒂兰（Turland, M.）著；彭冲，胡琳译 . —北京：机械工业出版社，2012.9

（华章程序员书库）

书名原文：PHP Master: Write Cutting-Edge Code

ISBN 978-7-111-39907-0

Ⅰ . P…　Ⅱ . ①米…　②沙…　③蒂…　④彭…　⑤胡…　Ⅲ . PHP 语言－程序设计　Ⅳ . TP312

中国版本图书馆 CIP 数据核字（2012）第 227029 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：秦　健

藁城市京瑞印刷有限公司印刷

2012 年 10 月第 1 版第 1 次印刷

186mm×240mm • 15.75 印张

标准书号：ISBN 978-7-111-39907-0

定价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88376949；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

# 译 者 序

本书是为 PHP 中级开发者量身定做的，从面向对象编程的基本概念开始，贯穿了数据库、设计模式、安全性以及自动测试、质量保证等各方面内容。本书的作者都是 PHP 资深工程师，他们在 PHP 领域有多年工作经历，积累了丰富的实践经验，而本书正是融合了这些宝贵经验而形成的。本书并不是一本呆板的教材，而是通过“怎样去做”的实例讲解方法由浅入深地介绍了 PHP 的各个重要知识点，还提供了许多非常好且实用的工具、方法及技巧。同时本书还提供了阅读的配套网站以及大量的网上资源，包括社区、网站、论坛以及专业会议，当我们遇到技术瓶颈或者需要补充相关方面的知识时，可通过以上多种途径寻求帮助。

因为 PHP 是开源的，所以有很多开源框架都提供了对它的支持，如 ZEND、SMARTY 等。PHP 的成本很低，相比起 .NET 等现在很流行的开发技术，PHP 的开销要低得多。PHP 是免费的，对服务器配置的要求也不是很高，因此非常适合中小型网站的开发。PHP 简单易学，非常适合缺乏计算机语言编程经验的人学习。

我们很荣幸能有机会承担本书的翻译工作。翻译本书也使我们系统地温习了 PHP 语言。同时深深体会到，具有丰富的技术实践加上良好的英语基础并不等于完美的翻译。因为是在业余时间进行翻译，所以尤感艰辛。在这三个月的时间里，逛街购物、电影电视等娱乐活动基本与我们绝缘，每天晚上匆匆忙完家务便立即坐在电脑前。对我们而言，一个礼拜七天都是工作日。我们常常会为一个术语、一个句子绞尽脑汁，并查阅大量资料，力图译文能正确、贴切地反映原文的意思，能够让句子、段落更符合中国人的语言习惯。我们真诚地希望你能从本书中有所收获，这是作者的初衷，也是我们的愿望。

感谢华章编辑的热情鼓励，让我们能有信心开始并完成本书的翻译工作。同时我们也感谢所有为本书出版付出心血的人们。

由于时间仓促，且我们的经验和水平有限，译文难免有不妥之处，恳请读者批评指正。

# 前　言

本书是针对 PHP 中级开发者的，即度过新手阶段并且希望提高技能的开发人员。我们的目的是帮助开发者在多个领域完善和提高自己的技能，因此我们在本书中精选了对开发者提升职业技能大有裨益的主题。

我们知道，本书中至少有一部分内容是你在工作中曾经遇到过的，但是即使是你熟悉的内容，也值得再次研读。PHP 语言也许比其他语言更加吸引各领域的人们。没有受过专业计算机教育的读者同样适合阅读本书。因此，积极地使用本书推荐的技术和方法，深入地阅读下面的章节，你会发现一些新的解决方案、新的理论知识。在日常工作实践中吸取经验也许需要很长时间才能取得进步，如果想快速汲取实战经验，打下牢固基础，阅读本书是个不错的选择。

本书可以帮助你从一个称职的网络开发者提升成为一名自信的网络工程师，即拥有丰富的实践经验，并且能够可靠而迅速完成工作的人。因为我们大家都一样，使用 PHP 作为一种谋生手段，所以我们在本书中使用“怎样去做”的实例讲解方法，希望用真实的案例向你提供实用的建议。

总之，我们希望你在本书找到所需要的内容，阅读顺利。

## 本书读者对象

如前所述，本书是为中级开发者所写，这意味着你已经具备牢固的 PHP 基础，其中包括代码语法规则，函数和变量如何运作，如何构建像 foreach、if/else 这样的流程控制语句，以及如何处理服务端脚本和客户端标记（例如 HTML 表单）。我们不会对这些基本原理老调重弹，但即便你非常熟悉书中提及的许多概念，你还是会学到很多改进创建服务器端应用程序的新方法。

我们准备使用面向对象编程（Object Oriented Programming，OOP）创建游戏，如果你早就知道这个术语，通过本书你会学到更多相关知识！众所周知，OOP 是一个优秀的 PHP 开发工作者必须遵循的标准，可使你编写高效的代码。你将学习如何有效地利用 OOP，来创建类以及实例化对象，让代码更加简洁，创建模板用于将来项目所需。如果你已经熟悉了面向对象的开发方法，阅读开篇的章节可以帮助进行复习；如果你还没有完全了解 OOP，那么阅读本书一定会让你获益匪浅。

另外，我们将学习使用数据库，这是 Web 开发中数据存储的关键模式。即便你对数据库及其工作原理基本上了解，深入研究数据库的连接方式也很必要，本书还将努力探索 MySQL 的世界，因为它是用于数据库交互信息最广泛的查询语言。

最后，本书总结了一些极好的方法来优化、测试和部署代码。虽然其中一些观念稍嫌超前，但我们会尽量提供比较精确的解释。熟悉命令行、接口等相关技术对理解这些章节更有帮助。

## 本书主要内容

本书由 8 章和 3 个附录组成，大多数章节的组织是按照内容顺序进行的，每一章都论述一个

主题。你可以选择按照章的顺序依次阅读，也可以直接阅读你感兴趣的内容。

**第 1 章 面向对象编程**

我们首先论述面向对象编程由哪些部分组成，如何将值和函数关联组成一个编程单元：对象。该章开始主要介绍如何声明类以及如何实例化对象，然后将深入研究继承、接口以及异常处理。到该章的结尾时我们将获得一幅详细面向对象编程蓝图。

**第 2 章 数据库**

互联网是一个动态的世界，用户只能浏览简单网页的日子一去不复返，数据库已成为交互式服务器端技术开发的关键组成部分。该章将研究如何用 PDO 连接数据库，如何存储数据和设计数据库范式。另外，我们将着重讲解基于结构化查询语言的 MySQL 以及与数据库交互的命令。

**第 3 章 API**

应用程序编程接口（API）是 Web 网页以外的另一种传输数据的方式，用 API 可以链接到某个特定的服务、应用程序或者公开的模块，以便与其他应用程序交互。我们将在该章学习如何利用它们组合系统，研究面向服务架构（SOA）、HTTP 的请求和响应，替代 Web 服务。

**第 4 章 设计模式**

在现实世界里，人们从不断重复的工作中总结出方法和经验，在编程中，称其为设计模式，它帮助用户优化开发和维护工作。该章将涵盖很多设计模式，包括单例模式、工厂模式、迭代模式以及观察者模式，还将介绍 MVC（Model-View-Controller）模式是如何架构和支撑一个良好的应用程序。

**第 5 章 安全性**

在那些心怀不轨的人手中，任何技术都有可能在某种程度上用于恶意行为，因此每个优秀的程序员都必须掌握确保其系统安全可靠的技术，而且客户对安全性也有要求。该章将涵盖很多已知的攻击技术，包括跨站脚本、会话劫持、SQL 注入，并讲解如何保护你的系统不被恶意程序侵入。我们将学习如何对密码加密，抵御暴力破解，并深入解析 PHP 格言：“过滤输入，避免输出。”

**第 6 章 性能**

应用程序变得越强大，就越需要测试其工作性能。在该章中，我们将学习如何使用像 ApacheBench 和 Jmeter 这样的工具对代码进行“压力测试”，这些工具可帮助快速优化服务器配置，简化文件系统以及分析代码运作。

**第 7 章 自动测试**

一个应用程序的功能发生变化时，其行为也会随之发生改变。自动测试的目的是为了确保应用程序的预期行为和实际行为是一致的。在该章中，我们将学习如何针对具体的应用程序进行单元测试、数据库测试、系统测试以及负载测试。

**第 8 章 质量保证**

谁都希望自己为创建应用程序所付出的努力不白费，且自己的项目能够达到一个很高的水平。在该章中，我们将学习如何用静态分析工具或者资源来测试质量，优化代码，完善文档，以及如何在 Web 上部署健壮性项目。

**附录 A PEAR 和 PECL**

在 PEAR 和 PECL 库中提及了很多工具，许多 PHP 开发者仍在使用它们。这个附录对这些

设置进行充分的说明，你也不再有理由忽视其中的重要内容了。

**附录 B PHP 标准库**

标准的 PHP 类库是一个传统的、不太知名的扩展，它与 PHP 标准配套，包含了很多有用的工具。这个附录可作为第 4 章的补充材料，很值得一读。

**附录 C 进一步参考信息**

下一步要怎么做？一个优秀的 PHP 开发者会不断提高自己的技能，这里会介绍一个非常实用的资源列表，包括各种社区和专业会议。

## 获取帮助途径

当你有疑问时，SitePoint 有一个由 Web 设计者和开发者组成的活跃社区随时为你提供帮助，在那里我们还提供一个本书勘误表，你可以随时查阅最近的更新。

### SitePoint 论坛

在 SitePoint Forums 论坛上你可以提出任何与 Web 开发相关的问题，同样你也可以解答问题。这是一个论坛网站，有人提问，有人回答，有人既提问也回答。在这里你可以和他人分享知识和经验，为社区作出贡献。这里有很多有趣而又经验丰富的网页设计师和开发者。这是一个学习的好地方，一旦你提问，便会得到迅速的解答。

### 本书的网站

本书的配套网站为：http://www.sitepoint.com/books/phppro/，该网站会为本书提供以下支持：

### 代码存档

如果你想通过本书提高技能，那么还需要在代码存档中标记很多注释，这些代码存档是一个可以下载的 ZIP 文件，包含本书全部示例的源代码。如果你想省事，那就直接下载这些存档。

### 更新和勘误表

没有一本书是完美无缺的，认真的读者在书中肯定会发现至少一至两处错误。在本书网站的勘误表上将始终提供更正印刷和代码错误的最新信息。

## SitePoint 简报

除了出版 PHP 技术书籍，SitePoint 还通过电子邮件免费寄发如 SitePoint Tech Times、SitePoint Tribune、SitePoint Design View 等电子简报。通过它们，你能了解到与网络开发技术相关的所有最新消息、产品发布、发展趋势、建议以及技术。你可以在 http://www.sitepoint.com/newsletter/ 注册一个或多个 SitePoint 会员以查阅相关资料。

## SitePoint 播客

欢迎读者加入 SitePoint 播客的队伍，这里有网络开发者和设计者所需的新闻、名人专访、专业见解以及有创意的想法。我们会讨论最新的网络行业话题，邀请嘉宾讲演人作报告，访问网络行业最顶尖的人物。你可以直接访问 http://www.sitepoint.com/podcast/，或者通过 iTunes 订购观看最新或者之前的播客。

## 读者反馈

如果你的疑问通过 SitePoint 论坛得不到解答，或者出于其他原因想直接和我们联系，你可以发送邮件至 books@sitepoint.com，我们建立了人员精良的邮件支持系统来跟踪解答你的质询，如果这些支持团队成员仍无法解答你的问题，他们会直接将你的邮件转发给我们。欢迎你对本书提出意见以及建议。

## 致谢

Lorna Mitchell

非常感谢那些鼓励我将写书的想法付之以实现的朋友。同时也要感谢那些人们，他们用一些小计谋促使我意识到，自己不仅可以开发软件，还可以撰写书籍。SitePoint 团队非常了不起，因为我完全是一个新手，他们不仅逐字逐句地帮我审阅书稿，更帮助我渡过了艰难的写书历程。最后要感谢本书的合著者，我自豪地称他们为朋友，我们共同分享这段写书的经历，你们两个都是我心目中的摇滚明星。

Davey Shafik

首先，我非常感谢我的妻子，Frances，写作本书占用了我很多晚上和周末的休息时间。也要感谢本书非常有才华的合著者，我很幸运能和他们成为朋友。感谢优秀的 SitePoint 团队为写作本书付出的努力。最后，感谢那些花时间阅读本书的读者，希望本书不仅能解答你的问题，更能够敞开你的心扉，迎接更好的未来。

Matthew Turland

2002 年，我第一次发现了 PHP，并于 2006 年加入 PHP 社区。我希望所有的技术来之于民也能用之于民。PHP 社区是我所发现的最好的软件开发者社区之一，我很荣幸成为其中一分子。感谢与我一起分享开发经验的人们，尤其是这些年来一直给我帮助和指导的朋友们。感谢本书杰出的合著者，Lorna 和 Davey，没有比他们更好的合作伙伴以及更能分享经验的朋友了。感谢优秀的 SitePoint 团队（Kelly Steele、Tom Museth、Sarah Hawk、Lisa Lang），他们将我们以及各篇文章结合在一起，出版了你所看到的完美书籍。还要感谢本书的审校者 Luke Cawood，还有我的朋友 Paddy Foran 和 Mark Harris，以及在写作阶段对本书提出反馈意见的人们。最后，感谢所有的读者，希望你们喜欢本书，并帮助你提高 PHP 技术。

## 本书中使用的约定

本书使用不同的排版样式表示不同类型的内容。首先，本书是关于 PHP 的，在本书大部分

代码示例中，省略了类似于（<?php and ?> 这样的开始和结束标记，这会便于你将代码内容直接插入到源文件中运行。唯一的例外是将 PHP 标记到旁边显示，比如 XML 或 HTML。

请看具体的示例。

### 代码示例

本书中的代码将使用固定宽度的字体显示，如：

```
class Courier { public function __construct($name) {
      $this->name = $name; return true; } }
```

如果你想在本书的代码存档中查找相应的代码，文件名将显示在程序列表的顶部，如：

example.php

```
function __autoload($classname) { include
        strtolower($classname) . '.php'; }
```

如果文件中只有部分内容，那么表示这个词是一个引用：

example.php (excerpt)

```
$mono = new Courier('Monospace
        Delivery');
```

如果在目前示例中插入新代码，那么新代码会加粗显示：

```
function animate() { new_variable =
      "Hello"; }
```

当现有代码需要上下文的内容时，与其重复所有的代码，不如将不相关的代码用垂直省略号代替：

```
function animate() { ⋮ return
      new_variable; }
```

由于页面的限制，本该在一行中显示的代码不得不换行显示时，用一个➥符号表示后面的内容是紧接着上一行中断的地方；而➥符号表示这里存在一个换行符，在代码中应该忽略它。

```
URL.open("http://www.sitepoint.com/blogs/2007/05/28/user-style-she
      ➥ets-come-of-age/");
```

### 提示、注释和警告


小指针表示会提供有用的小提示。


注释是有用的旁白，它们与本书主题相关，但不是关键性的内容。你只需将它们视为某些知识的小花絮。


上面的图标表示需要注意的要点。


警告部分将着重强调那些容易误导你进入陷阱的地方。

# 目　录

# 第 1 章

# 面向对象编程

在本章中，我们将学习面向对象编程（Object Oriented Programming，OOP）。无论你在使用 PHP 之前是否接触过 OOP，本章都会揭示什么是 OOP，如何使用 OOP，以及为什么要使用对象，而不是直接使用函数和变量。我们将从“如何创建一个对象”这个基本原理开始，讲解接口、异常以及神奇的方法等内容。虽然面向对象的方法更倾向于概念性而非技术性，但是我们仍要使用专门的一章精确地解释它，努力揭开 OOP 神秘的面纱。

## 1.1 为什么要使用面向对象编程

你可能会质疑，既然只需使用方法就可以写出复杂且实用的网站，那为什么还要采取其他的措施，而且使用 OOP 不是增添麻烦吗？ OOP 真正的价值在于**封装**，这是 OOP 在 PHP 中使用得越来越多的原因。它的意义在于将相互关联的一组值和函数封装在一起，组成一个编程单元：对象。使用对象可以让我们将一组值存放在一起，而且还能为它添加功能，而不是在变量前面添加前缀使我们知道它们与什么相互关联，或者存储在数组中以集合元素。

### OOP 术语

将一些很平常的概念表述得很复杂，这种倾向往往会阻碍人们对事物的理解。因此，在你阅读本书时，为避免发生这种情况，我们准备了一个简短的术语表：

| | |
|---|---|
| **class** | 创建对象的方法或蓝图 |
| **object** | 实例 |
| **instantiate** | 从类创建对象的动作 |
| **method** | 属于对象的函数 |
| **property** | 属于对象的变量 |

现在带上你新的“外语”词典，让我们去看一看代码吧。

## 1.2 OOP 简介

开始冒险吧！在理论知识方面，我们会结合代码示例来讲解，这让你更容易看懂代码的实际意义。

### 1.2.1 声明类

类相当于蓝图，是表明如何创建对象的一组指令。它还不是一个对象，而仅仅是对象的一个

描述。在 Web 应用程序中，用类来表示各种实体。下面是一个可能用于电子商务应用程序中的 Courier 类：

chapter_01/simple_class.php

```
class Courier
{
  public $name;
  public $home_country;

  public function __construct($name) {
    $this->name = $name;
    return true;
  }

  public function ship($parcel) {
    // sends the parcel to its destination
    return true;
  }
}
```

以上代码表明了如何声明类，可以将 Courier 类保存在一个名为 courier.php 的文件中。这个文件的命名方法是需要牢记的一个要点，其重要性在 1.3 节中会详细阐述，我们会讲解当需要时如何访问对象。

上面的例子表明 Courier 类有两个属性：$name 和 $home_country，以及两种方法：__construct() 和 ship()，在类中声明方法的方式和我们所熟悉的声明函数的方式完全一样，因此一定要牢记这个语法。当写一个函数的时候，可以用同一种方式向方法中传入参数以及返回值。

你可能已经注意到示例代码中还有一个名为 $this 的变量，这是一个特别的变量，在对象的范围内它一直是可用的，用于指代当前对象。在本章的示例代码中，会用它从对象内部直接访问变量和调用方法，因此，在阅读本章时要留心这一点。

### 1.2.2 类的构造

__construct() 函数名字前面有两条相连的下划线。在 PHP 中，两条下划线表示一个神奇的方法，表示这是一个具有特殊意义或功能的方法。在本章我们将看到很多这样的方法。__construct() 方法是一种特殊的函数，在实例化一个对象时会调用它，称其为**构造函数**（constructor）。

**PHP 4构造函数**

PHP 4 中没有什么神奇的方法。对象都有构造函数，并且在类声明中，构造函数的名字与类名相同。虽然在最新版本的 PHP 中已不再使用这种规范，但在遗留的代码或者与 PHP 4 兼容的代码中还可以看到这种规范，不过在 PHP 5 中已不再使用这种规范。

当实例化一个对象时通常会调用构造函数，当释放构造函数进而在代码中使用之前会用它

创建和配置对象。构造函数还有一个和它相匹配的神奇方法，称为**析构函数**（destructor），它是一个名为 __destruct() 不带参数的方法。当销毁对象时，会调用析构函数，并且运行对象所需的停止或者清除任务。注意，虽然我们不能保证析构函数何时会运行，但当对象因销毁或超出作用域，或者 PHP 垃圾收集器开始运行等原因而不再使用时，析构函数才会运行。

当我们阅读本章的示例时，将会发现很多这样的示例或者神奇的方法。现在，让我们实例化一个对象，这将极好地解释构造函数是如何运行的。

### 1.2.3 对象实例化

要实例化或创建一个对象，要使用新的关键字，如同给一个对象取名，需要声明一个类；然后将预期使用的参数传入构造函数中。要实例化一个 courier，应该这样做：

```
require 'courier.php';

$mono = new Courier('Monospace Delivery');
```

首先，需要包含类定义的文件（courier.php），因为 PHP 需要用这个类来创建对象。然后只需实例化一个新的 Courier 对象，将构造函数预期的名字参数传递进来，然后以保存在 $mono 为名字的对象中作为结束。如果使用 var_dump() 方法检查对象，我们会看到：

```
object(Courier)#1 (2) {
 ["name"]=>
 string(18) "Monospace Delivery"
 ["home_country"]=>
 NULL
}
```

var_dump() 的输出告诉我们：

- 这是一个属于 Courier 类的对象；
- 它有两种属性；
- 每个属性的名称和值。

当实例化对象时可以将参数值传入构造函数。在示例中，构造 Courier 对象时通过构造函数传入参数给 $name 属性赋值。

### 1.2.4 自动加载

到目前为止，上述示例表明了如何声明一个类，然后在需要的地方引用那个文件。在一个大型应用程序里，不同的文件需要包含在不同的脚本里，这一切很快会变得复杂而混乱，令人高兴的是，PHP 有一种特性可使这一切变得很容易，称为**自动加载**。当需要一个类声明而不知道在哪里寻能找到类文件时，PHP 自动加载会指引我们。

为自动加载定义规则时，用到了另一个神奇的方法：__autoload()。在前面的示例中，引用了一个文件，但作为一种选择，可以用自动加载方法来替代这种方法：

```
function __autoload($classname) {
  include strtolower($classname) . '.php';
}
```

只有当你用显而易见的方法命名而且保存包含类定义的文件时，自动加载才会发挥作用。到目前为止，示例都是很简单的内容；类文件都具有相同的名字，都带有 .php 扩展的小写文件名，因此自动加载函数处理的都是这种简单的示例。

如果需要，也可以创建一个复杂的自动加载函数。例如，许多现代应用程序都是以 MVC（Model-View-Controller，第 4 章会深入解释）模式构建的，在为类定义时，模型、视图和控制器往往会存放在不同的目录，为了解决这个问题，通常会用类的类型来为类命名，比如 UserController，而自动加载功能会用字符串匹配或正规表达式来解释要寻找的类的特征，以及在哪里可以找到这些类。

### 1.2.5 使用对象

到目前为止，我们已经知道如何声明类，实例化对象，并且谈到了自动加载，但我们还没有进行更多的面向对象编程。接下来我们将使用所创建对象的属性和方法来工作，让我们用一些示例代码来看看究竟应该怎么做：

```
$mono = new Courier('Monospace Delivery');

// accessing a property
echo "Courier Name: " . $mono->name;

// calling a method
$mono->ship($parcel);
```

在这里，使用了**对象运算符**，这是用一个连字符和大于号组成的符号：->。它将对象与属性、方法或者你想访问的内容连接起来。连接符后的方法带有圆括号，而属性不带圆括号。

### 1.2.6 使用静态属性和方法

在举例说明了如何使用类，并解释了如何实例化对象之后，接下来要介绍的内容在概念上发生了转变。和初始化对象一样，将类的属性和方法定义为**静态的**（static）。静态属性或方法在使用时无需事先实例化对象。在任何一种情况下，可以标记一个元素为静态，将静态的关键字放在 public 之后（或其他可视性修饰符——本章后面会讲到很多这样的情况）。通过双冒号操作符 :: 就可以访问它们。

**范围解析操作符**

在 PHP 中，使用双冒号操作符访问静态属性或方法，双冒号操作符在技术上称为范围解析操作符（scope resolution operator）。如果包含 :: 操作符的代码发生问题，你常常会看到一个包含有 T_PAAMAYIM_NEKUDOTAYIM 内容的错误提示。虽然一眼看上去颇令人恐惧，但这只是一个简单的 :: 引用。“Paamayim Nekudotayim”在希伯来语中的意思是“二点，二次”的意思。

静态属性是仅属于类的变量，而不属于对象。它完全孤立于任何属性，甚至于在类的对象中具有相同名字的属性。