

TURING

图灵程序设计丛书

Web开发系列

MANNING



PHP in Action

Objects, Design, Agility

PHP

实战

[挪] Dagfinn Reiersøl

[英] Marcus Baker 著

[美] Chris Shiflett

张颖 等译
段大为 审校



人民邮电出版社

POSTS & TELECOM PRESS

TURING

丛书 Web开发系列



PHP in Action
Objects, Design, Agility

PHP

实战

[挪] Dagfinn Reiersøl

[英] Marcus Baker 著

[美] Chris Shiflett

张颖 等译
段大为 审校

人民邮电出版社

样书

专业书

人民邮电出版社
北京

图书在版编目 (CIP) 数据

PHP 实战 / (挪) 雷勒索 (Reiersøl, D.), (英) 贝克 (Baker, M.), (美) 史夫利特 (Shiflett, C.) 著; 张颖等译. —北京: 人民邮电出版社, 2010.1

(图灵程序设计丛书)

书名原文: PHP in Action: Objects, Design, Agility
ISBN 978-7-115-21745-5

I. ① P… II. ①雷… ②贝… ③史… ④张… III. ①
PHP 语言—程序设计 IV. ① TP312

中国版本图书馆CIP数据核字 (2009) 第206532号

内 容 提 要

随着 PHP 5 的发布和 Zend 框架项目的应用, PHP 和敏捷思想、设计模式以及单元测试的联姻成为主流话题。本书首先介绍了面向对象设计的原则、模式和技巧, 然后讲述了如何在 PHP 上利用其语法和特性, 处理和实施这些原则、模式和技巧, 并应用于 Web 编程中遇到的难题上。

本书适合所有使用 PHP 开发 Web 应用程序的人员阅读。

图灵程序设计丛书

PHP实战

-
- ◆ 著 [挪] Dagfinn Reiersøl [英] Marcus Baker [美] Chris Shiflett
 - 译 张 颖 等
 - 审 校 段大为
 - 责任编辑 傅志红
 - 执行编辑 陈兴璐
 - ◆ 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 电子函件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 北京隆昌伟业印刷有限公司印刷
 - ◆ 开本: 800×1000 1/16
 - 印张: 27.75
 - 字数: 656千字 2010年1月第1版
 - 印数: 1-3 000册 2010年1月北京第1次印刷

著作权合同登记号 图字: 01-2008-3296号

ISBN 978-7-115-21745-5

定价: 69.00元

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

反盗版热线: (010)67171154

版 权 声 明

Original English language edition , entitled *PHP in Action: Objects, Design, Agility* by Dagfinn Reiersøl, Marcus Baker and Chris Shiflett, published by Manning Publications Co., 209 Bruce Park Avenue, Greenwich, Connecticut 06830. Copyright © 2007 by Manning Publications Co..

Simplified Chinese-language edition copyright © 2010 by Posts & Telecom Press. All rights reserved.

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

版权所有，侵权必究。

前言

这本书的背后有我个人的一段故事。几年前，我意识到自己职业生涯中的一切努力和我的预期并不十分相符。虽然还没有严重到中年危机的地步，但这种意识促使我开始以新的方式来思考。

那时我正从事PHP的Web编程。在我所供职的公司里，我处于一种孤独的境地。于是我决定仔细审视一下自己的工作。我扪心自问：“如何才能让自己有更卓越的表现？”一个想法就是，每天结束的时候都对当天的工作做出总结。我做的最成功的是什么？怎样才能取得更多成功？什么是不太成功的？如何才能减少不成功的事？

木桶的短板就是调试。很明显，它占用了我绝大多数的时间，如果有什么可以使调试更有效或者根本可以不要调试，那么我的工作会更高效。我开始尝试更早地捕捉bug。我试过防御型编程，但不是很成功。不久我偶然发现了敏捷过程和测试驱动开发、极限编程和重构。这些方法看上去像我和我的同事们这些年来一直的做法，但确实要好不少。我首先在自己的工作中运用了这套方法。当时，PHP社区中鲜有人做此尝试。我是比较早这样做的，我使用了2002年3月的PHPUnit最初的测试版实践测试先行。

有这本书的构想，是在我接手了一位程序员同行一些拙劣的PHP代码之后。我意识到代码可以改进、重构，并且可以系统地阐述它。我想这对于有些人一定是有用的。并且，PHP领域尚没有关于敏捷过程和测试驱动开发的图书。

一件事情意外地加速了这个写书的项目：我被解雇了。（几个月后，我成了那间解雇我的公司的董事会成员，这是另一个故事。）我花了将近三年的时间完成本书。要让审阅者对它十分满意不是件易事，很多部分我都重写了好几次。Marcus Baker和Chris Shiflett在将近结尾处参与了进来。同时，PHP和敏捷思想、设计模式和单元测试的联姻成为本书的主流话题。这个过程中最重要的官方事件是PHP 5的发布和Zend框架项目的启动。

整个过程当中我学到很多东西，最重要的是，如果想自己写一本书，那就一定要多读书。我相信深层理解的重要性，不只是知道很多的细节，而是对每个细节都有深入的理解。我也相信，这来源于扎实的基础和从多方面理解问题的能力。

这一切都引领我不断重新审视那些最基本的问题。我不断地问一些看起来有些傻的问题。在网络论坛中，即使是在讨论我非常熟悉的问题时，我也经常被误认为是初学者。我相信，我自己的理解越深，我就能为别人把这个问题解释得更好。我希望这种探索对你们来说也是有帮助的。

Dagfinn Reiersøl

致 谢

本书的写作得到了我的一些朋友和对手的帮助。

我先解释一下所谓的对手，我用那个词只是表明一种看法。他们不是坏人，也不是故意跟我过不去（希望如此）。但是有几位的确让我的日子有点不好过，让我不得不去做一些我本不打算做的事情，而结果却提高了本书的质量。对此我表示感激，但在此我不提他们的姓名。

朋友方面，感谢我的妻子Regine和我的女儿Maria，感谢你们的爱、支持和鞭策。感谢我的儿子Jacob（已经6岁了）那懵懵懂懂的热情和令人忍俊不禁的见解，这在书中的一些地方得到了体现。

更实际地讲，最重要的贡献来自于合著者：我的好朋友Marcus Baker，我还从未见过他；以及Chris Shiflett，他在百忙之中抽出时间为本书撰写了有关安全的内容。

和其他Manning的作者一样，我对整个出版团队和他们对质量的严格要求印象十分深刻。他们知道并且也做到了如何将一本书的可读性和趣味性提升到更高的高度。可能我有些自夸，但我对这个结果确实非常满意，每一次我重读书中的章节，都会觉得是一种享受。

审阅的过程令人疲劳但是却非常重要。出版人Marjan Bace有一种特殊的能力和决心来处理哪怕是最难以令人兴奋的反馈，即使反馈意见非常不具体，他都能从中提炼出一些有用的东西。

感谢这些在百忙之中抽时间阅读了不同阶段原稿的审校者们：Richard Lynch, Andrew Grothe, Kieran Mathieson, Jochem Maas, Max Belushkin, Dan McCullough, Frank Jania, Jay Blanchard, Philip Hallstrom, Robin Vickery, David Hanson, Robbert van Andel, Jeremy Ashcraft, Anthony Topper, Wahid Sadik, Nick Heudecker及Robert D. McGovern。特别是Mark Monster，他在本书出版之前又再一次通读了全书，确保了技术上的准确性。

另一位间接的贡献者是我长期的同事和朋友Per Einar Arnstad。从我们充满创意的讨论和互动中得来的思想，是我研究软件的基石，他的创业精神也激励我勇于承担这项风险，使本书的出版成为可能。

还感谢另一位同事Tarjei Huse，在阅读了原稿之后，他给予了我几乎是最全面的、充满智慧的反馈。

最后，特别感谢妻子Kathrine Breistøl，她答应我，如果我的财务状况糟糕到了一定程度，就会全面清理厨房里所有瓶子去卖钱。当然，我从来也没要求她兑现过。

关于本书

本书从某种意义上来说有“一女二嫁”之嫌。它首先介绍了面向对象设计的最新原则、模式和技巧，然后又将这位“新人”许配给两位“新郎官”，第一位是PHP编程语言，另一位是PHP程序员每天的日常工作。

更具体地讲，本书就是介绍如何利用PHP的语法和特性，设计和实现这些原则、模式和技巧的。本书同时也将这些原则、模式和技巧应用于Web编程中遇到的各种具体和常见的难题上。

读者对象

本书适合使用PHP开发应用程序的程序员，还有希望学习最新面向对象实践、原则和技巧并将其应用于Web编程的日常问题的读者。

这不是一本关于PHP的入门书籍，它假设你有最基本的PHP知识，或者有其他编程语言的经验，而且对Web编程基本概念及可能遇到问题有所了解。

内容结构

本书分为四个部分。第一部分和第二部分介绍开始提到的原则、模式和技巧，并且展示如何在PHP中实现它们。第一部分引入并展开讨论面向对象编程和设计的主题。第二部分将介绍单元测试和重构。第三部分和第四部分将前两部分的素材应用到日常Web编程中可能遇到的问题。第三部分关于Web界面，而第四部分解决数据库和数据存储的问题。

第一部分：工具和概念

第一部分中的几章循序渐进，从PHP面向对象编程的具体细节讲到面向对象应用程序设计这个更抽象的主题。

第1章介绍并讨论了PHP和敏捷实践的利弊。

第2章和第3章解释PHP面向对象编程的机制和语法。尽管对象和类是难以分割的东西，但是第2章主要关注对象的特性，而第3章则重点关注类。

第4章讨论了对象和类的概念好在哪儿，它们如何与现实世界产生联系，以及如何区分面向对象设计的优劣。

第5章讨论最基本的类的关系——继承、关联和组合，以及程序设计中接口的作用。

第6章开始我们正式探索面向对象设计，将面向对象理论作为设计的指导思想。

第7章介绍设计模式——再现常见设计问题的解决方案，并且描述一些最常见的问题。

第8章介绍设计原则和模式在处理日期和时间时的另一种应用。

第二部分：测试和重构

第二部分的重点是测试和重构（即改善现有代码的设计），将从质量保证和学习过程这两个方面进行探讨。

第9章介绍了单元测试和测试驱动开发，使用了数据库事务处理（transaction）类作为实例。

第10章在单元测试领域进行了深入的挖掘，展示了如何正确配置测试，并且使用模拟对象和其他替代品来简化测试。这一章以第9章为基础，通过事务处理类创建了一个联系人管理器。

第11章是关于重构的，并且重点关注Web应用程序。这一章既解决传统意义上面向对象重构的问题，也针对如何将设计不良的过程化代码转化为更好管理的状态。

第12章将注意力从单元测试转向Web测试，结束了对测试这个主题的讨论。再一次以联系人管理器为例，展示了如何确认用户界面是否符合用户期望，以及如何自顶向下地设计出整个Web应用程序。

第三部分：构建Web界面

第三部分专注于Web编程的重要方面：Web界面。

第13章解释了从程序代码中分离HTML的原则，并且描述了如何通过使用模板引擎和具体的实现技巧。

第14章解决的问题是将很多分散的组件组装成网页，同时介绍了如何实现复合视图设计模式。

第15章介绍了用户交互和MVC（模型-视图-控制器）模式。

第16章将会教你如何将MVC模式运用于Web开发上面，包括实现页面控制器和前端控制器。

第17章深入讲解了服务器端和客户端输入验证以及如何实现验证同步。

第18章展示了如何利用PEAR中的HTML_QuickForm包开发表单处理。

第四部分：数据库和基础结构

第四部分解决的是从面向对象的角度来处理数据库和数据存储的问题。

第19章分两个不同的部分。一部分讲述如何在面向对象程序里适当地处理数据库连接以及如何处理数据库连接所需要的设置。另一部分讲述数据库抽象，即如何使代码独立于具体的数据库管理系统。

第20章面对的问题，源自我们必须使用一门完全独立的编程语言（SQL）来查询数据库。这一章讲述了如何封装、隐藏以及生成SQL代码。

第21章将前两章的内容集中在一起，形成一个比较完整的面向对象数据存取的设计模式。

附录

附录A给出一些关于测试和测试工具的具体信息，这些不太适合在测试章节中介绍，所以在这里给出，同时也包括了SimpleTest和PHPUnit API核心部分的参考资料。

附录B介绍了PHP的安全问题。

使用本书

本书各章相对独立，可以从任何一章开始阅读，而不必一定要读过之前的章节。除非你对面向对象编程和设计已经得心应手，否则的话，应首先读完第一部分，这有助于你对第三部分和第

四部分有更容易、更深层、更透彻的理解。但在随后的章节里，所有例子的工作原理都会被详细地解释。这些例子使第一部分阐明的概念更加易懂，但是一般并不依赖这些概念。

但是，某些部分里的一些章彼此有较强的联系。例如，如果没有理解第9章和第10章里面介绍的单元测试的基本概念，那么就很难读懂第11章中关于重构的例子。

示例代码

本书正文和代码清单中的源代码都用代码体。很多代码清单都带有注解以阐明重要概念。某些情况下，代码清单中会有编号，对应于正文中的注解。

关于书名

*In Action*系列丛书通过结合介绍、概览和how-to示例来帮助人们学习和记忆。根据对认知科学的研究，人们记住的东西正是他们在自我激励的探索中所发现的东西。

尽管在Manning出版社工作的没有认知科学家，但我们很肯定一点，即要想让学过的东西记得牢固，就必须经历探索、行动和重复所学这样几个阶段。人们只有积极地探索过，才能理解和记住新东西，也即掌握新东西。人们是在实践中学习。*In Action*指南的一个要点是它是示例驱动的。它鼓励读者试着去实现、去实践新代码和探索新想法。

本书这样命名还有一个更现实的原因：我们的读者很忙。他们读书是为了完成工作或解决问题。他们需要能够易于进退的图书，并且恰好在他们正需要的时候学到最需要的东西。他们需要能帮他们进行实践的书。本系列的这套丛书正是为这样的读者设计的。

关于封面插图

本书封面上的法国农妇在法语里叫做“*paysanne*”。这个插图取自1805年版Sylvain Maréchal关于不同地区着装风格的四卷纲要。该书于1788年在巴黎首印，正是法国大革命爆发的前一年。书中的每幅画都是手绘的。

Maréchal收集的插图多种多样，生动地展现了200年前世界各城镇和州/省的独特风情。当时，距离数十英里的两个地区的人，通过不同的着装风格即可区分开来。这些图画生动地描述了那个时期以及除了充满活力的现代以外所有其他历史时期，人们生活的疏离感和距离感。

着装风格改变了，而当时地区性的多样性也渐渐消融了。如今，很难分辨出人们是哪个洲的居民。也许乐观地看，我们经历这些文化和视觉的多样性是为了拥有更加多样化的个人生活，或者是为了拥有更加多样和有趣的知识和技术生活。

我们这些在Manning出版社工作的人盛赞这样的创造性、主动性，当然还有这些图书封面带给计算机事业的乐趣，这些图书封面基于两个世纪前丰富的地区多样性，其中的图片把我们带入到那时的生活中！

目 录

第一部分 工具和概念

第1章 PHP 与现代软件开发.....2

- 1.1 PHP 的作用.....2
 - 1.1.1 PHP 流行的原因.....3
 - 1.1.2 克服 PHP 的局限.....5
- 1.2 语言、原则和模式.....7
 - 1.2.1 敏捷方法：从蛮干到巧干.....7
 - 1.2.2 PHP 5 和软件趋势.....8
 - 1.2.3 面向对象编程进化的规律.....8
 - 1.2.4 设计模式.....9
 - 1.2.5 重构.....10
 - 1.2.6 单元测试和 TDD.....10
- 1.3 小结.....13

第2章 PHP 中的对象.....14

- 2.1 对象基础.....14
 - 2.1.1 为什么将 PHP 与 Java 相比较.....15
 - 2.1.2 对象和类.....15
 - 2.1.3 Hello world.....15
 - 2.1.4 构造函数：创建和初始化对象.....16
 - 2.1.5 继承和关键字 extends.....18
 - 2.1.6 继承构造函数.....19
- 2.2 异常处理.....20
 - 2.2.1 异常的工作原理.....20
 - 2.2.2 何时使用异常与返回代码.....22
 - 2.2.3 创建自己的异常类.....23
 - 2.2.4 用异常替换 PHP 内置的严重错误.....24
 - 2.2.5 不要过度使用异常.....24
- 2.3 PHP 4 和 PHP 5 中的对象引用.....24
 - 2.3.1 对象引用的工作原理.....25

- 2.3.2 对象引用的优势.....26
- 2.3.3 引用何时没有用处.....27
- 2.4 方法调用的拦截和类的实例化.....27
 - 2.4.1 什么是方法重载.....27
 - 2.4.2 PHP 中的 Java 式方法重载.....27
 - 2.4.3 面向方面的体验：记录方法调用的日志.....28
 - 2.4.4 自动加载类.....30
- 2.5 小结.....31

第3章 有效使用 PHP 类.....32

- 3.1 可见性：私有和受保护的方法与变量.....32
 - 3.1.1 对方法可见性的要求.....33
 - 3.1.2 何时使用私有方法.....34
 - 3.1.3 何时使用受保护方法.....34
 - 3.1.4 让实例变量保持 private 或 protected 属性.....35
 - 3.1.5 私有变量和受保护变量的存取.....36
 - 3.1.6 两全其美——通过拦截来控制变量.....37
 - 3.1.7 final 类和方法.....38
- 3.2 没有对象的类：类方法、类变量和类常量.....39
 - 3.2.1 类（静态）方法.....40
 - 3.2.2 何时使用类方法.....41
 - 3.2.3 类变量.....41
 - 3.2.4 类常量.....42
 - 3.2.5 PHP 中常量的限制.....43
- 3.3 抽象类和方法（函数）.....45
 - 3.3.1 什么是抽象类和方法.....45
 - 3.3.2 使用抽象类.....45

3.4 类的类型提示	46	5.4.1 避免父类命名含糊	81
3.4.1 类型提示的作用	46	5.4.2 避免继承层次结构过深	81
3.4.2 何时使用类型提示	47	5.5 小结	82
3.5 接口	48	第6章 面向对象原则	83
3.5.1 什么是接口	49	6.1 原则和模式	84
3.5.2 PHP 中需要接口吗	49	6.1.1 架构原则或模式	84
3.5.3 用接口让设计更清晰	50	6.1.2 了解面向对象原则	85
3.5.4 用接口改善类的类型提示	50	6.2 开放-封闭原则 (OCP)	85
3.5.5 PHP 5 与 Java 中的接口	51	6.2.1 初识 OCP	85
3.6 小结	52	6.2.2 用类替换的情况	86
第4章 理解对象和类	53	6.2.3 OCP 在 PHP 中是如何相关的	88
4.1 对象和类的优点	54	6.3 单一职责原则 (SRP)	88
4.1.1 类帮助组织	54	6.3.1 混合职责: 模板引擎	89
4.1.2 可以告诉对象要做什么	55	6.3.2 一个试验: 分离职责	91
4.1.3 多态性	55	6.3.3 实验是否成功	93
4.1.4 对象让代码更易读	56	6.4 依赖倒置原则 (DIP)	94
4.1.5 类帮助消除重复性代码	59	6.4.1 什么是依赖性	94
4.1.6 可以重用对象和类	60	6.4.2 插入接口	96
4.1.7 避免牵一发而动全身	61	6.5 分层设计	96
4.1.8 对象提供类型安全	61	6.5.1 “三层”模式及其同属	97
4.2 好设计标准	62	6.5.2 Web 应用程序能否有域层	98
4.2.1 不要混淆结果和含义	64	6.6 小结	99
4.2.2 透明性	64	第7章 设计模式	100
4.2.3 简单设计	64	7.1 策略模式	101
4.2.4 一次并且只有一次	65	7.1.1 使用策略模式的“Hello world”	101
4.3 什么是对象	67	7.1.2 策略模式的用处	103
4.3.1 对象来自虚构世界	67	7.2 适配器模式	104
4.3.2 域对象基础	68	7.2.1 初学适配器模式	104
4.4 小结	70	7.2.2 让一个模板引擎与另一个相像	105
第5章 理解类关系	71	7.2.3 具有多个类的适配器模式	106
5.1 继承	71	7.2.4 调整为通用接口	109
5.1.1 将继承作为思考工具	72	7.3 装饰器模式	109
5.1.2 继承重构	73	7.3.1 资源装饰器	110
5.2 对象组合	77	7.3.2 装饰与再装饰	111
5.3 接口	79	7.4 空对象模式	113
5.3.1 将接口作为思考工具	79	7.4.1 混合黑暗的灯和明亮的灯	114
5.3.2 单继承和多继承	80	7.4.2 空策略对象	114
5.4 优先考虑对象组合而不是类继承	80	7.5 迭代器模式	115

7.5.1 迭代器的工作原理	115	8.6.4 时间间隔	151
7.5.2 使用迭代器的好原因	116	8.7 小结	151
7.5.3 迭代器与普通数组	116	第二部分 测试和重构	
7.5.4 SPL 迭代器	117	第 9 章 测试驱动开发 154	
7.5.5 SPL 如何帮助我们解决迭代器 和数组间的冲突	118	9.1 过程形成质量	155
7.6 组合模式	118	9.1.1 本示例的需求	155
7.6.1 用组合模式实现菜单	118	9.1.2 报告测试结果	156
7.6.2 基本理论	120	9.2 从数据库取数	157
7.6.3 连贯接口	121	9.2.1 基本测试	157
7.6.4 递归处理	121	9.2.2 第一个真正的测试	158
7.6.5 我们的方法低效吗	123	9.2.3 通过测试	160
7.7 小结	123	9.2.4 让代码运行	161
第 8 章 设计指南：日期和时间处理	124	9.2.5 测试直到确信没有问题	163
8.1 为何日期和时间处理要面向对象	124	9.3 数据库插入和更新	164
8.1.1 更容易，但并非更简单	125	9.3.1 让测试更易读	165
8.1.2 面向对象的优势	125	9.3.2 红，绿，重构	166
8.2 找到正确的抽象	126	9.4 真正的数据库事务处理	168
8.2.1 单个时间表示法：时间点、 Instant、DateAndTime	126	9.4.1 测试事务处理	168
8.2.2 不同类别的时间范围：期间、持续 时间、日期范围、时间间隔	127	9.4.2 实现事务处理	170
8.3 高级对象构建	128	9.4.3 调试的终结	171
8.3.1 使用创建方法	128	9.4.4 测试是工具，不是替代品	171
8.3.2 多个构造函数	129	9.5 小结	172
8.3.3 使用工厂类	132	第 10 章 高级测试技术	173
8.4 大型结构	133	10.1 具有持久化功能的联系人管理器	174
8.4.1 包的概念	133	10.1.1 运行多个测试用例	174
8.4.2 命名空间和包	134	10.1.2 测试联系人的持久化	175
8.4.3 PHP 缺少命名空间支持	135	10.1.3 Contact 和 ContactFinder 类	177
8.4.4 处理名称冲突	135	10.1.4 setUp() 和 tearDown()	178
8.5 使用值对象	140	10.1.5 最终版本	179
8.5.1 对象引用带来的麻烦	141	10.2 向联系人发送邮件	180
8.5.2 实现值对象	142	10.2.1 设计 Mailer 类及其测试环境	180
8.5.3 更改不可变的对象	142	10.2.2 手工编写模拟对象	181
8.6 实现基本类	143	10.2.3 更为完善的模拟对象	182
8.6.1 DateAndTime	143	10.2.4 自顶向下测试	183
8.6.2 属性和字段	144	10.2.5 模拟的局限性	184
8.6.3 期间	149	10.3 虚拟的邮件服务器	185
		10.3.1 安装 fakemail	186
		10.3.2 邮件测试	187

10.3.3	网关作为适配器	190
10.4	小结	190
第 11 章	重构 Web 应用程序	192
11.1	真实世界中的重构	193
11.1.1	早期重构和后期重构	193
11.1.2	重构与重新实现	194
11.2	重构基础：可读性和重复性代码	195
11.2.1	提高可读性	195
11.2.2	消除重复性代码	197
11.3	分离标记与程序代码	199
11.3.1	分离何以有用	200
11.3.2	合宜使用 CSS	200
11.3.3	清理生成链接的函数	201
11.3.4	在 SimpleTest 中引入模板	205
11.4	简化条件表达式	209
11.4.1	简单示例	210
11.4.2	稍长的示例：身份验证代码	211
11.4.3	处理条件 HTML	216
11.5	从面向过程到面向对象的重构	217
11.5.1	测试面向过程代码	217
11.5.2	进行重构	218
11.6	小结	221
第 12 章	用 Web 测试控制	222
12.1	再看联系人管理器	223
12.1.1	样板	223
12.1.2	创建 Web 测试	225
12.1.3	用虚拟网页交互通过测试	226
12.1.4	一次编写，到处测试	227
12.2	可工作的表单	229
12.2.1	尝试将联系人保存到数据库中	230
12.2.2	创建数据库	231
12.2.3	为查找器创建存根	232
12.3	质量保证	234
12.3.1	让联系人管理器可以进行单元测试	234
12.3.2	从用例到验收测试	236
12.4	可怕的遗留代码	238
12.5	小结	242

第三部分 构建 Web 界面

第 13 章	使用模板管理 Web 表现层	244
13.1	分离表现层和域逻辑	244
13.1.1	分离还是不分离	245
13.1.2	为什么使用模板	245
13.2	哪个模板引擎	247
13.2.1	普通 PHP	248
13.2.2	定制语法：Smarty	249
13.2.3	属性语言：PHPTAL	251
13.3	转换：XSLT	254
13.3.1	“XML 化”网页	255
13.3.2	设置 XSLT	256
13.3.3	XSLT 样式表	256
13.3.4	从 PHP 运行 XSLT	258
13.4	将逻辑与模板分离	259
13.4.1	视图协助器	260
13.4.2	交替行颜色	260
13.4.3	处理日期和时间格式	261
13.4.4	生成层级显示	263
13.4.5	防止从模板更新	265
13.5	模板和安全	266
13.5.1	PHPTAL	266
13.5.2	Smarty	267
13.5.3	XSLT	267
13.6	小结	267
第 14 章	构建复杂网页	269
14.1	组合模板（复合视图）	269
14.1.1	复合视图：一个或多个设计模式	269
14.1.2	复合数据和复合模板	270
14.2	实现直观的复合视图	270
14.2.1	我们的目标	270
14.2.2	使用 Smarty	272
14.2.3	使用 PHPTAL	273
14.2.4	使用 PHPTAL 的页面宏	274
14.3	复合视图示例	275
14.3.1	制作打印友好的页面	276
14.3.2	将现有应用程序集成到复合视图中	277

14.3.3 多方显示站点和 Fowler 的 两步视图.....	278	16.4 小结.....	312
14.4 小结.....	280	第 17 章 输入验证	314
第 15 章 用户交互	281	17.1 应用程序设计中的输入验证.....	315
15.1 MVC 体系结构.....	282	17.1.1 验证和应用程序体系结构.....	315
15.1.1 拨开 MVC 的迷雾.....	283	17.1.2 验证策略.....	316
15.1.2 定义基本概念.....	284	17.1.3 命名表单组件.....	317
15.1.3 命令还是操作.....	286	17.2 服务器端验证及其问题.....	317
15.1.4 Web MVC 不是富客户 MVC.....	286	17.2.1 重复问题.....	318
15.2 Web 命令模式.....	287	17.2.2 样式问题.....	318
15.2.1 工作原理.....	288	17.2.3 测试和页面导航问题.....	319
15.2.2 命令标识符.....	288	17.2.4 我们能解决多少问题.....	319
15.2.3 Web 处理程序.....	289	17.3 客户端验证.....	320
15.2.4 命令执行器.....	289	17.3.1 普通的乏味的客户端验证.....	320
15.3 保持实现简单.....	290	17.3.2 逐个验证字段.....	321
15.3.1 示例：“原生的” Web 应用 程序.....	290	17.3.3 你做不到这一点.....	323
15.3.2 引入命令函数.....	292	17.3.4 表单.....	326
15.4 小结.....	294	17.4 面向对象的服务器端验证.....	327
第 16 章 控制器	296	17.4.1 规则和验证程序.....	328
16.1 控制器和请求对象.....	297	17.4.2 安全的请求对象体系结构.....	329
16.1.1 基本请求对象.....	297	17.4.3 现在验证非常简单.....	333
16.1.2 安全问题.....	298	17.4.4 让其变得简单的类.....	334
16.2 使用页面控制器.....	299	17.4.5 使用 Specification 对象.....	336
16.2.1 简单示例.....	300	17.4.6 知识丰富的设计.....	339
16.2.2 从页面控制器选择视图.....	301	17.4.7 向外观添加验证.....	340
16.2.3 让命令可进行单元测试.....	302	17.5 同步服务器端和客户端验证.....	341
16.2.4 避免 HTML 输出.....	303	17.5.1 表单生成器.....	342
16.2.5 使用模板.....	303	17.5.2 配置文件.....	342
16.2.6 重定向问题.....	304	17.5.3 从客户端验证生成服务器端 验证.....	343
16.3 构建前端控制器.....	307	17.6 小结.....	343
16.3.1 一个命令一个类的 Web 处理 程序.....	307	第 18 章 表单处理	345
16.3.2 命令还需要些什么.....	308	18.1 用 HTML_QuickForm 设计解决 方案.....	345
16.3.3 使用命令组.....	309	18.1.1 最小需求和设计.....	346
16.3.4 有多个提交按钮的表单.....	310	18.1.2 将生成的元素放到 HTML 表单中.....	346
16.3.5 用 JavaScript 生成命令.....	311	18.1.3 找到抽象.....	347
16.3.6 用于复合视图的控制器.....	311	18.1.4 更为具体的需求.....	348

18.1.5 选择问题	349	20.3 通用化 SQL	383
18.2 实现解决方案	350	20.3.1 列的列表和表名	383
18.2.1 包装 HTML_QuickForm 元素	350	20.3.2 使用 SQL 别名	386
18.2.2 输入控件	351	20.3.3 生成 INSERT、UPDATE 和 DELETE 语句	386
18.2.3 哪个类创建表单控件	354	20.3.4 查询对象	390
18.2.4 验证	355	20.3.5 适用的设计模式	391
18.2.5 在模板中使用表单对象	357	20.4 小结	391
18.2.6 下一步做什么	359	第 21 章 数据类设计	392
18.3 小结	359	21.1 最简单的方法	392
第 19 章 数据库连接、抽象和配置	361	21.1.1 用 Finder 类检索数据	393
19.1 数据库抽象	362	21.1.2 主要程序: 表数据网关	395
19.1.1 预处理语句	362	21.2 让对象自身持久化	400
19.1.2 面向对象的数据库查询	364	21.2.1 自我持久化的查找器	401
19.2 装饰和适配数据库资源对象	366	21.2.2 让对象存储自己	405
19.2.1 简单的可配置数据库连接	366	21.3 数据映射器模式	406
19.2.2 从结果集制作与 SPL 兼容的 迭代器	367	21.3.1 数据映射器和 DAO	406
19.3 让数据库连接可用	369	21.3.2 这些模式无甚差别	408
19.3.1 单例和类似模式	370	21.3.3 模式小结	409
19.3.2 服务定位器和注册表	371	21.4 实际使用效果	409
19.4 小结	373	21.4.1 模式在典型 Web 应用程序 中的效果	410
第四部分 数据库和基础结构		21.4.2 优化查询	411
第 20 章 对象和 SQL	376	21.5 小结	411
20.1 对象-关系阻抗不匹配	376	附录 A 测试工具和小技巧	412
20.2 封装和隐藏 SQL	378	附录 B 安全	420
20.2.1 基本示例	378		
20.2.2 在 SQL 语句中替换字符串	379		

Part 1

第一部分

工具和概念

当你要完成某项工作时，总是自然而然会在开始时寻找需要的工具。在面向对象编程的世界中，工具和概念之间的区别非常模糊。有描述和实现概念关系的工具，也有作为设计过程工具的概念。

本书第一部分将介绍这些工具和概念，其中多数属于面向对象编程和应用程序设计的范畴。在应对第三部分和第四部分的Web编程挑战时，将应用这些工具和概念。我们将介绍PHP中对象和类的语法，为何以及如何使用它们，还有如何使用设计模式和面向对象原则。

本部分内容

- 第1章 PHP与现代软件开发
- 第2章 PHP中的对象
- 第3章 有效使用PHP类
- 第4章 理解对象和类
- 第5章 理解类关系
- 第6章 面向对象原则
- 第7章 设计模式
- 第8章 设计指南：日期和时间处理



第 1 章

PHP与现代软件开发

1

本章内容

1.1 PHP的作用

1.2 语言、原则和模式

1.3 小结

一幅漫画中，有个穿医生制服的人正在打电话：“是的，琼斯先生，针灸会暂时起到一定作用。任何庸医的治疗都可能会暂时起作用。但是只有科学的医疗实践才能让一个人永生。”^①

这段荒谬而自大的话语显然不能让病人信服。可是，如果我们忽略那些牵强的细节，会发现通过这个虚构的医生的故事，至少说明了一个重要问题——谨记长期目标的重要性。

医疗的长期好处与本书的主题相去甚远，但从软件的长期发展来看，则是另一回事了。现代软件工程并不想让软件永远可用，但生产力长期发展却是新技术、原则和方法论发展的关键问题之一。这就是为什么面向对象编程是今天的实际（*de facto*）标准：这种方式可以让软件从推出第一版之后的后续各版本更易于维护和开发。其他专业术语，如设计模式（*design pattern*）和敏捷开发（*agile development*），也都与此相关。

PHP 5是一次新的尝试，力图通过PHP（“PHP: Hypertext Processor”的递归缩写）更轻松地运用这些概念和方法。

本书将以此作为着眼点，揭示这次尝试带来的新变化。我们将围绕三个互相关联的目标展开讨论。

- 工具包的探索和最佳利用。用现代方法和工具将开发技巧提升到一个新水平。
- 提供全面介绍。从用户界面到数据库交互，在Web编程的各个方面应用该工具包。
- 保持其简单性。遵循爱因斯坦的建议——凡事应该尽可能简单，而又不过于简单。

无论出于什么原因使用PHP（有可能像我一样是出于偶然才使用的），了解PHP的长处总会有所帮助，而且更有助于了解如何克服它的局限。出于此目的，本章开始将讨论PHP自身的一些利弊，然后介绍现代的面向对象方法和敏捷方法，了解它们与PHP的关联。

1.1 PHP 的作用

PHP一直是主要用于Web编程的一种语言，现在仍是这样。而且，随着PHP 5（或者PHP 6，也许你在阅读本书时已经发布这个版本了）的发布，它带来了最新的特性，成为一种与现代面向

^① 这段情景来自于我的记忆——几年前在同事的办公室看过这幅漫画，后来就再没见过。