

<<PHP核心技术与最佳实践>>

图书基本信息

书名：<<PHP核心技术与最佳实践>>

13位ISBN编号：9787111401926

10位ISBN编号：7111401921

出版时间：2012-11

出版时间：机械工业出版社

作者：列旭松，陈文

页数：540

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<PHP核心技术与最佳实践>>

前言

前言为什么要写这本书近几年，市场上关于PHP的书已经很多了，各种培训机构也如雨后春笋般不断增加。

那为什么还要写这本书呢？

这本书存在的意义又在哪儿？

这要从下面的几个问题说起。

有没有这样一本PHP教材，它不讲HTML和CSS，也不讲JavaScript基础，甚至不讲PHP语法基础？

有没有这样一本PHP教材，它不讲留言板或博客的开发，也不讲数据库的CRUD操作？

有没有这样一本PHP教材，它专注于Web开发技术的最前沿，深入浅出，适合中高级程序员的进阶和提高？

有没有这样一本PHP教材，它提倡面向对象的程序思想，提倡算法和数据结构的重要性，提倡对网络协议的深入理解，且没有大篇幅的代码，而是更多偏重于理论讲解？

有没有这样一本PHP教材，它探讨PHP的扩展开发，探讨高并发大流量的架构，深入探讨NoSQL的内部实现和细节？

以上几个问题也是我在早期PHP学习的过程中一直在寻找的答案，可是我并没有找到一本理想的PHP书籍，一本适合中高级程序员进阶的书籍。

当怀着同样问题的旭松兄找到我时，我们不禁产生一个念头：“既然现在市场上缺少一本这样的书籍，我们何不自己写一本呢？”

利己利人的事值得去做。

”然后一拍即合，说做就做，现在这本书经历长达一年多的酝酿和写作过程终于完稿了。

我是在大学期间接触到PHP语言的，并马上被其简洁的语法和极高的开发效率所吸引，一头扎进PHP开发的世界中。

随着学习的深入，并经常关注PHP社区的动态，我很快意识到一些PHP社区普遍存在的问题。

比如PHP社区一直争论算法重不重要，面向对象好不好，代码质量重要还是开发速度重要的问题。

还有譬如为什么我去大型互联网公司应聘PHP程序员，却不考察我对PHP语法和函数的掌握情况，而是会问我C语言、算法、网络协议、高并发处理、MVC理论这些看似和PHP不沾边的问题。

PHP到底要怎么学，学什么，一个高级PHP程序员应该是什么样的，我想这也是很多PHP新手和工作一两年的PHP开发者的疑惑。

这本书所要解决的就是这一系列的问题。

在我看来，一本技术书籍的价值在于其对知识的提炼和与众不同的地方。

举例来说，到一个书店去看书，你最想用笔抄下来或撕下来带走的那几页，就是对你帮助最大的东西，也是你认为这本书的价值所在。

也是基于这个想法，我们思考这本书该写什么，怎么写，哪些地方对读者有帮助。

我们试图从不同的角度带领读者来看PHP，进而给这本书注入一些不一样的东西。

我们希望这是一件有意义的事。

本书适合的对象PHP爱好者；想进阶的初级PHP程序员；对PHP扩展开发感兴趣的读者；对高并发感兴趣的读者；对NoSQL应用和实现原理感兴趣的读者；从事PHP网络应用，想知道HTTP协议、Socket等更多细节的开发人员；想就职于大型互联网公司的PHP程序员；开设相关课程的大专院校的学生；公司内部培训的学员。

如何阅读本书本书一共有14章。

每章节都可以单独阅读，由于部分知识点之间存在一定的衔接，故建议按先后顺序阅读。

第1章为面向对象思想的核心概念。

本章主要讲解面向对象开发的思想，重点讲解面向对象模型的建立，以及面向对象的一些基础概念。

通过大量对比和实例，尤其是与Java的对比，力图从不同角度讲解PHP面向对象的特性，让PHP程序员看到不同的面向对象。

求同存异是本章的核心思想。

<<PHP核心技术与最佳实践>>

第2章为用面向对象思维写程序。

本章用简练的语言讲解了面向对象设计的五大原则，这五大原则也是理解设计模式的基础所在，帮助读者站在一个更高的角度思考面向对象。

第3章为正则表达式技巧与实战。

本章详细介绍了正则的基础语法，通过大量的示例、通俗的语言讲解正则概念，引导读者理解正则的一系列规则。

接下来，结合实际工作安全过滤、URL重写等实例，加深对正则的应用和掌握。

最后给出正则效率优化的一些普遍技巧和替代方案，让读者对正则的使用得心应手。

第4章为PHP网络技术及应用。

本章着重介绍了HTTP协议、Socket开发、WebService、Cookie和Session使用等。

结合实战向读者阐述网络开发的核心和重点，特别是对HTTP协议的理解。

HTTP协议是Web开发的基石，也是各种面试和开发中必然遇到的知识点。

而Socket则是应用交互的桥梁，保证了有用的可扩展性。

第5章为PHP与数据库基础。

本章从不同角度分析了MySQL，介绍了PDO、MySQL优化、存储过程、事件调度机制以及MySQL安全防范等内容。

第6章为PHP模板引擎的原理与最佳实践。

本章通过实现一个简单的模板引擎，学习模板引擎的原理和使用方法，然后对比几大流行的模板引擎实现方案，简单介绍了各种实现方案的思想 and 优缺点，最后探讨模板引擎的意义。

第7章为PHP扩展开发。

本章的知识是本书核心内容，介绍了PHP扩展开发的几个重要知识点，如扩展框架搭建、PHP生命周期、PHP变量在内核中的实现方式、Zend引擎、内存管理等，让读者深入PHP底层，知其然也知其所以然。

第8章为缓存。

本章主要介绍了缓存的基本原理和三个衡量指标，通过几个实例加深读者对缓存的理解。

利用本章知识，读者应该能设计一个比较合理的缓存方案。

第9章为Memcached应用与内幕。

本章深入剖析了Memcached的实现和内部结构，从而使读者掌握Memcached的高级应用，对构建复杂环境的缓存层有个清晰的认识。

第10章为Redis应用与内幕。

本章重点介绍了Redis的深入应用，如事务处理、主从同步、虚拟内存等，和第9章类似，探讨了Redis的实现内幕。

合理利用Redis可以为我们解决大流量高并发的应用。

第11章为高性能网站架构。

本章探讨了高性能架构的基本出发点，重点以Handler Socket、MySQL主从复制、反向代理缓存软件Varnish和任务分发框架Gearman为例，讲述几种高性能架构中会用到的技术。

第12章为调试与测试。

科学的调试方法有助于快速找出潜在的Bug、理解复杂应用的流程、提高开发效率。

单元测试是代码质量的保障。

在这一章的最后一节介绍了使用JMeter进行压力测试的方法。

第13章为Hash算法与数据库的实现。

本章介绍了Hash算法的基本原理，用此算法实现一个简单的、基于Hash的数据库，让读者意识到算法的重要性和可操作性。

第14章为PHP编码规范。

本章介绍了PHP开发中应遵循的基本代码规范，并提出合理建议。

好的代码必然是规范的代码。

本书第1、2、3、5、6、8、12、14章由陈文撰写，第7、9、10、11、13章由列旭松撰写，第4章由两人

<<PHP核心技术与最佳实践>>

共同完成。

勘误和支持由于我们的水平和开发经验有限，同时计算机技术更新较快，书中难免存在不足之处，有些章节内容可能从未来的某一天开始不再适用，还望读者理解和体谅，并恳请读者批评指正。

您若对本书有什么好的建议或者对书中部分内容有疑惑，可与我们联系，我们将尽量为读者提供最满意的解答。

期待得到您的真挚反馈。

我们的联系方式如下：陈文：waitfox@qq.com 列旭松：liexusong@qq.com 感谢首先要感谢PHP之父Rasmus Lerdorf，是他创建了这个简单、轻松、有趣、快速而又高效的语言；其次，感谢PHP社区每一位充满活力的朋友，和你们的交流使我学到很多，本书有不少内容就来自于社区的智慧。

在这里尤其要感谢机械工业出版社华章公司的大力支持，特别是杨福川和白宇两位编辑，在一年多的时间里，因为有了你们的耐心指导、逐字逐句认真审稿和改稿才有了本书的诞生。

最后，还要感谢家人和朋友的支持。

陈文

<<PHP核心技术与最佳实践>>

内容概要

这是一本致力于为希望成为中高级PHP程序员的读者提供高效而有针对性指导的经典著作。本书系统归纳和深刻解读了PHP开发中的编程思想、底层原理、核心技术、开发技巧、编码规范和最佳实践。

全书分为5个部分：第一部分（1~2章）从不同的角度阐述了面向对象软件设计思想的核心概念、技术和原则，分析了面向对象的特性、设计模式的理念，指出了如何设计低耦合、高可扩展性的软件，等等；第二部分（3~6章）详细讲解了PHP中正则表达式的规范和使用技巧，PHP网络编程的原理、方法、技巧和一些重要的操作，PDO、数据库应用优化，数据库设计和MySQL的高级应用，PHP扩展引擎的原理与实践；第三部分（第7章）拨云见日，围绕PHP扩展开发进行了细致而深入的探讨，解析了PHP的底层实现和Zend虚拟机API，并用PHP扩展开发的实例带领读者走进PHP的底层世界，旨在让读者对PHP性能优化、底层原理进行深入的理解。

第四部分（8~11章）重点讨论了缓存的设计、Memcached的原理与实践、NoSQL数据库Redis源码分析与应用实践、高性能PHP网站的架构和设计等内容；第五部分（12~14章）详细讲解了PHP代码的调试和测试、Hash算法和数据库的实现，以及PHP的编码规范，旨在帮助读者提高开发效率，养成良好编程习惯。

<<PHP核心技术与最佳实践>>

作者简介

列旭松，资深PHP技术工程师，精通PHP及其相关技术，对PHP内核原理有较深入的理解，开发经验丰富。

曾自主开发了关键字匹配服务器（<http://code.google.com/p/sensitive-filter-server/>）和消息队列SquirrelMQ（<http://code.google.com/p/squirrel-message-queue/>）。

平时喜欢开发一些实用的PHP扩展，如PHP字典扩展（红黑树算法）

（<http://code.google.com/p/php-dict/>）和PHP索引扩展（B+树算法）

（<http://code.google.com/p/php-mini-database/>）。

精通C语言，同时对Web服务器的架构和优化、高并发服务端编程、Redis和Memcached等技术有深入的研究和认识。

活跃于PHPChina和ChinaUnix等专业社区，担任PHPChina论坛内核版块版主。

陈文，资深PHP技术工程师，精通PHP及其相关技术，尤其擅长于PHP框架开发和应用架构。

他还是一位资深的Java开发工程师，具有Fortran、Scala和C++语言的开发和使用背景，在传统软件和互联网开发领域都有丰富的实战经验。

此外，他还擅长TCP/IP编程、多线程与并发程序设计、网络协议分析、数据库性能优化以及各种缓存技术，熟悉MySQL和Oracle等关系数据库产品。

现从事网络安全软件开发，以及移动SI业务开发。

对语言特性和软件设计思想有独到的见解，追求代码之美和高效率程序开发，爱好钻研底层技术，崇尚和提倡“以理论指导实践”。

尤其爱好数学，认为数学是培养和锻炼思维和逻辑能力的重要工具，对算法有一定研究。

长期活跃在PHPChina、ITeye和看雪论坛等社区，在PHPChina社区担任版主。

<<PHP核心技术与最佳实践>>

书籍目录

前言

第1章 面向对象思想的核心概念

1.1 面向对象的“形”与“本”

1.1.1 对象的“形”

1.1.2 对象的“本”

1.1.3 对象与数组

1.1.4 对象与类

1.2 魔术方法的应用

1.2.1 set和get方法

1.2.2 call和callStatic方法

1.2.3 toString方法

1.3 继承与多态

1.3.1 类的组合与继承

1.3.2 各种语言中的多态

1.4 面向接口编程

1.4.1 接口的作用

1.4.2 对PHP接口的思考

1.5 反射

1.5.1 如何使用反射API

1.5.2 反射有什么作用

1.6 异常和错误处理

1.6.1 如何使用异常处理机制

1.6.2 怎样看PHP的异常

1.6.3 PHP中的错误级别

1.6.4 PHP中的错误处理机制

1.7 本章小结

第2章 面向对象的设计原则

2.1 面向对象设计的五大原则

2.1.1 单一职责原则

2.1.2 接口隔离原则

2.1.3 开放-封闭原则

2.1.4 替换原则

2.1.5 依赖倒置原则

2.2 一个面向对象留言本的实例

2.3 面向对象的思考

2.4 本章小结

第3章 正则表达式基础与应用

3.1 认识正则表达式

3.1.1 PHP中的正则函数

3.1.2 正则表达式的组成

3.1.3 测试工具的使用

3.2 正则表达式中的元字符

3.2.1 什么是元字符

3.2.2 起始和结束元字符

3.2.3 点号

<<PHP核心技术与最佳实践>>

- 3.2.4 量词
- 3.3 正则表达式匹配规则
 - 3.3.1 字符组
 - 3.3.2 转义
 - 3.3.3 反义
 - 3.3.4 分支
 - 3.3.5 分组
 - 3.3.6 反向引用
 - 3.3.7 环视
 - 3.3.8 贪婪懒惰匹配模式
- 3.4 构造正则表达式
 - 3.4.1 正则表达式的逻辑关系
 - 3.4.2 运算符优先级
 - 3.4.3 正则表达式的常用模式
- 3.5 正则在实际开发中的应用
 - 3.5.1 移动手机校验
 - 3.5.2 匹配E-mail地址
 - 3.5.3 转义在数据安全中的应用
 - 3.5.4 URL重写与搜索引擎优化
 - 3.5.5 删除文件中的空行和注释
- 3.6 正则表达式的效率与优化
- 3.7 本章小结
- 第4章 PHP网络技术及应用
 - 4.1 HTTP协议详解
 - 4.1.1 HTTP协议与SPDY协议
 - 4.1.2 HTTP协议如何工作
 - 4.1.3 HTTP应用：模拟灌水机器人
 - 4.1.4 垃圾信息防御措施
 - 4.2 抓包工具
 - 4.2.1 抓包工具分类
 - 4.2.2 Fiddler功能与原理
 - 4.2.3 安装Fiddler
 - 4.2.4 Fiddler基本界面
 - 4.2.5 使用Fiddler进行HTTP断点调试
 - 4.3 Socket进程通信机制及应用
 - 4.3.1 进程通信相关概念
 - 4.3.2 Socket演示：实现服务器端与客户端的交互
 - 4.3.3 Socket函数原型
 - 4.3.4 PHP中的Socket函数
 - 4.3.5 Socket交互应用：使用Socket抓取数据
 - 4.4 cURL工具及应用
 - 4.4.1 建立cURL请求的基本步骤
 - 4.4.2 检查cURL错误和获取返回信息
 - 4.4.3 在cURL中伪造头信息
 - 4.4.4 在cURL中用POST方法发送数据
 - 4.4.5 使用cURL上传文件
 - 4.4.6 cURL批处理

<<PHP核心技术与最佳实践>>

- 4.4.7 cURL设置项
- 4.4.8 网络应用：使用cURL抓取腾讯微博
- 4.5 简单邮件传输协议SMTP
 - 4.5.1 SMTP协议如何工作
 - 4.5.2 SMTP协议常用命令
 - 4.5.3 SMTP协议应用：使用Socket发送邮件
- 4.6 Webservice的前世今生
 - 4.6.1 Webservice简介
 - 4.6.2 认识PHPRPC协议
 - 4.6.3 Web服务的实现模式
 - 4.6.4 简单对象访问协议SOAP
 - 4.6.5 调试工具soapUI
- 4.7 Cookie详解
 - 4.7.1 Cookie的基本概念及设置
 - 4.7.2 PHP和JavaScript对Cookie的操作
 - 4.7.3 Cookie存储机制及应用
 - 4.7.4 Cookie跨域与P3P协议
 - 4.7.5 本地存储localStorage
- 4.8 Session详解
 - 4.8.1 Session的基本概念及设置
 - 4.8.2 Session的工作原理
 - 4.8.3 Session入库
 - 4.8.4 Cookie与Session问答
- 4.9 本章小结
- 第5章 PHP与数据库基础
 - 5.1 什么是PDO
 - 5.1.1 PDO预定义类
 - 5.1.2 如何使用PDO
 - 5.1.3 PDO参数绑定与预编译
 - 5.1.4 PDO事务处理
 - 5.1.5 PDO的效率问题
 - 5.2 数据库应用优化
 - 5.2.1 基本语句优化10个原则
 - 5.2.2 索引与性能分析
 - 5.2.3 服务器和配置的优化
 - 5.2.4 MySQL瓶颈及应对措施
 - 5.3 数据库设计
 - 5.3.1 范式与反范式
 - 5.3.2 数据库分区
 - 5.3.3 分表的应用
 - 5.4 MySQL的高级应用
 - 5.4.1 MySQL自增长序列
 - 5.4.2 MySQL视图
 - 5.4.3 MySQL存储过程和事件调度
 - 5.4.4 用MySQL模拟消息队列
 - 5.4.5 SQL注入漏洞与防范
 - 5.5 本章小结

<<PHP核心技术与最佳实践>>

第6章 PHP模板引擎的原理与实践

- 6.1 代码分层的思想
- 6.2 实现一个简单的模板引擎骨架
 - 6.2.1 搭建模板引擎基础类骨架
 - 6.2.2 编译类骨架
 - 6.2.3 测试模板引擎
- 6.3 模板引擎的编译
 - 6.3.1 实现变量标签
 - 6.3.2 实现foreach标签
 - 6.3.3 实现if...else标签
 - 6.3.4 对PHP原生语法的支持
- 6.4 完善模板引擎
 - 6.4.1 模板缓存机制的实现
 - 6.4.2 调试和缓存清理
 - 6.4.3 如何使用模板
- 6.5 常用模板引擎
 - 6.5.1 Discuz模板引擎
 - 6.5.2 Smarty模板引擎
 - 6.5.3 DedeCms模板引擎
 - 6.5.4 Blitz模板引擎
 - 6.5.5 模板引擎的一些思考
- 6.6 本章小结

第7章 PHP扩展开发

- 7.1 为什么要开发PHP扩展
- 7.2 搭建PHP扩展框架
 - 7.2.1 PHP源代码目录
 - 7.2.2 ext_skel工具
 - 7.2.3 Windows平台环境配置
 - 7.2.4 Linux平台环境配置
 - 7.2.5 PHP的生命周期
- 7.3 PHP内核中的变量
 - 7.3.1 PHP变量在内核中的存储方式
 - 7.3.2 PHP内核变量访问宏
 - 7.3.3 引用计数器与写时复制
- 7.4 PHP内核中的HashTable分析
 - 7.4.1 PHP内核HashTable的数据结构
 - 7.4.2 HashTable的代码实现
- 7.5 Zend API详解与扩展编写
 - 7.5.1 什么是Zend引擎
 - 7.5.2 Zend引擎内存管理
 - 7.5.3 PHP扩展的架构
 - 7.5.4 接收用户传递的参数
 - 7.5.5 在PHP扩展中创建变量
 - 7.5.6 在PHP扩展中为变量赋值
 - 7.5.7 错误和输出API
 - 7.5.8 运行时信息函数
 - 7.5.9 调用用户自定义函数

<<PHP核心技术与最佳实践>>

- 7.5.10 PHP配置项
- 7.5.11 创建常量的宏
- 7.6 编写一个完整的扩展
 - 7.6.1 链表结构的实现
 - 7.6.2 创建PHP扩展框架
 - 7.6.3 编写代码
 - 7.6.4 编译安装扩展
 - 7.6.5 测试扩展
- 7.7 本章小结
- 第8章 缓存详解
 - 8.1 认识缓存
 - 8.1.1 为什么使用缓存
 - 8.1.2 命中率
 - 8.1.3 缓存更新策略
 - 8.1.4 缓存最大数据量
 - 8.2 文件缓存
 - 8.2.1 文件缓存机制
 - 8.2.2 文件缓存开源产品Secache
 - 8.3 Opcode缓存
 - 8.3.1 eAccelerator下载及使用
 - 8.3.2 如何查看Opcode
 - 8.4 客户端缓存
 - 8.4.1 客户端缓存规则
 - 8.4.2 HTTP协议中的缓存使用
 - 8.4.3 HTTP缓存实例
 - 8.4.4 HTML 5中的Application Cache
 - 8.5 Web服务器缓存
 - 8.5.1 Apache缓存
 - 8.5.2 Nginx缓存
 - 8.6 本章小结
- 第9章 Memcached使用与实践
 - 9.1 为什么要用Memcached
 - 9.2 Memcached的安装及使用
 - 9.2.1 安装Memcached服务器
 - 9.2.2 安装Memcached客户端
 - 9.2.3 使用memcache扩展访问Memcached服务器
 - 9.2.4 使用Memcached加速Web应用
 - 9.3 深入了解Memcached
 - 9.3.1 Memcached如何支持高并发
 - 9.3.2 使用Slab分配算法保存数据
 - 9.3.3 删除过期item
 - 9.3.4 使用LRU算法淘汰数据
 - 9.3.5 Memcached多线程模型
 - 9.4 Memcached分布式布置方案
 - 9.4.1 普通Hash分布
 - 9.4.2 一致性Hash分布
 - 9.4.3 一致性Hash分布算法实例

<<PHP核心技术与最佳实践>>

9.5 本章小结

第10章 Redis使用与实践

10.1 Redis的安装及使用

10.1.1 Redis安装步骤

10.1.2 修改Redis配置文件

10.1.3 运行Redis服务器

10.1.4 key相关命令

10.1.5 Redis支持的数据类型

10.1.6 Redis排序命令详解

10.2 事务处理

10.2.1 事务处理原理

10.2.2 事务处理实现

10.3 持久化

10.3.1 内存快照

10.3.2 日志追加

10.4 主从同步

10.4.1 Redis主从同步原理

10.4.2 Slave端的工作流程

10.4.3 Master端的工作流程

10.5 虚拟内存

10.5.1 配置文件信息

10.5.2 开启VM的后台操作

10.5.3 Redis Object和VM Pointer

10.5.4 交换过程

10.5.5 阻塞式VM

10.5.6 非阻塞式VM

10.6 扩展库phpredis安装及使用

10.7 Redis应用实践

10.7.1 使用消息队列发布微博

10.7.2 Redis替代文件存储Session

10.8 深入了解Redis内核

10.8.1 内存淘汰

10.8.2 对象引用计数器

10.8.3 自动关闭超时连接

10.8.4 清除过期数据

10.9 本章小结

第11章 高性能网站架构方案

11.1 如何优化网站响应时间

11.1.1 吞吐率

11.1.2 压力测试

11.1.3 持久连接

11.2 MySQL响应速度提高方案：HandlerSocket

11.2.1 HandlerSocket工作原理

11.2.2 HandlerSocket安装和配置

11.2.3 PHP-HandlerSocket性能测试

11.3 MySQL稳定性提高方案：主从复制

11.3.1 主从复制工作原理

<<PHP核心技术与最佳实践>>

- 11.3.2 主从复制配置
- 11.3.3 连接主从服务器
- 11.4 Web应用加速方案：Varnish
 - 11.4.1 传统代理与反向代理
 - 11.4.2 Varnish安装和配置
 - 11.4.3 Varnish性能测试
 - 11.4.4 修改缓存规则
 - 11.4.5 监控Varnish运行状态
- 11.5 异步计算方案：Gearman
 - 11.5.1 Gearman工作原理
 - 11.5.2 安装Gearman和PHP扩展
 - 11.5.3 使用Gearman异步发送邮件
- 11.6 本章小结
- 第12章 代码调试和测试
 - 12.1 调试PHP代码
 - 12.1.1 PHP调试函数
 - 12.1.2 断点调试与变量跟踪工具Xdebug
 - 12.2 前端调试
 - 12.2.1 Firebug调试API
 - 12.2.2 使用Firebug调试DOM结构
 - 12.2.3 使用Firebug调试JavaScript
 - 12.2.4 使用Fiddler调试远程服务器上的文件
 - 12.3 日志管理
 - 12.3.1 PHP日志
 - 12.3.2 Apache服务器日志
 - 12.3.3 MySQL日志
 - 12.4 代码性能测试技术
 - 12.4.1 时间点测试
 - 12.4.2 文件查看工具WinCacheGrind
 - 12.4.3 性能测试注意事项
 - 12.5 单元测试
 - 12.5.1 单元测试框架PHPUnit的安装
 - 12.5.2 结合NetBeans使用PHPUnit进行单元测试
 - 12.5.3 PHPUnit中的断言函数
 - 12.5.4 PHPUnit常用方法
 - 12.5.5 PHPUnit常用注解
 - 12.6 压力测试
 - 12.6.1 使用JMeter压力测试HTTP
 - 12.6.2 压力测试MySQL
 - 12.6.3 JMeter+Badboy组合测试
 - 12.7 本章小结
- 第13章 Hash算法与数据库实现
 - 13.1 Hash函数
 - 13.2 Hash算法
 - 13.2.1 直接取余法
 - 13.2.2 乘积取整法
 - 13.2.3 经典Hash算法Times33

<<PHP核心技术与最佳实践>>

13.3 Hash表

13.3.1 Hash表结构

13.3.2 使用PHP实现Hash表

13.3.3 Hash表冲突

13.3.4 拉链法解决冲突

13.4 一个小型数据库的实现

13.4.1 pack函数的用法

13.4.2 unpack函数的用法

13.4.3 索引文件和数据文件

13.4.4 数据库接口方法

13.4.5 源代码解析

13.4.6 测试代码

13.5 本章小结

第14章 PHP编码规范

14.1 文件格式

14.1.1 文件标记

14.1.2 文件和目录命名

14.1.3 文件目录结构

14.2 命名规范

14.2.1 变量命名

14.2.2 类及接口命名

14.2.3 数据库命名

14.2.4 习惯与约定

14.3 注释规范

14.3.1 程序注释

14.3.2 文件注释

14.3.3 类接口注释

14.3.4 方法和函数注释

14.3.5 标注的使用

14.4 代码风格

14.4.1 缩进和空格

14.4.2 语句断行

14.4.3 更好的习惯

14.5 本章小结

章节摘录

版权页：插图：继承并非一无是处，而组合也不是完美无缺的。

如果既要组合的灵活，又要继承的代码简洁，能做到吗？

这是可以做到的，譬如多重继承，就具有这个特性。

多重继承里一个类可以同时继承多个父类，组合两个父类的功能。

C++里就是使用的这种模型来增强继承的灵活性的，但是多重继承过于灵活，并且会带来“菱形问题”，故为其使用带来了不少困难，模型变得复杂起来，因比在大多数语言中，都放弃了多重继承这一模型。

多重继承太复杂，那么还有其他方式能比较好地解决这个问题吗？

PHP5.4引入的新的语法结构Traits就是一种很好的解决方案。

Traits的思想来源于C++和Ruby里的Mixin以及Scala里的Traits，可以方便我们实现对象的扩展，是除extend、implements外的另外一种扩展对象的疗法。

Traits既可以使单继承模式的语言获得多重继承的灵活，又可以避免多重继承带来的种间问题。

1.3.2各种语言中的多态 多态确切的含义是：同一类的对象收到相同消息时，会得到不同的结果。

而这个消息是不可预测的。

多态，顾名思义，就是多种状态，也就是多种结果。

以Java为例，由于Java是强类型语言，因此变量和函数返回值是有状态的。

比如，实现一个add函数的功能，其参数可能是两个int型整数，也可能是两个float型浮点数，而返回值可能是整型或者浮点型。

在这种情况下，add函数是有状态的，它有多种可能的运行结果。

在实际使用时，编译器会自动匹配适合的那个函数。

这属于函数重载的概念。

需要说明的是，重载并不是面向对象里的东西，和多态也不是一个概念，它属于多态的一种表现形式。

多态性是一种通过多种状态或阶段描述相同对象的编程方式。

它的真正意义在于：实际开发中，只要关心一个接口或基类的编程，而不必关心一个对象所属于的具体类。

很多地方会看到“PHP没有多态”这种说法。

事实上，不是它没有，而是它本来就是多态的。

PHP作为一门脚本语言，自身就是多态的，所以在语言这个级别上，不谈PHP的多态。

在PHP官方手册也找不到对多态的详细描述。

既然说PHP没有多态这个概念（实际上是不需要多态这个概念），那为什么又要讲多态呢？

可以看下面的例子，如代码清单1—8所示。

<<PHP核心技术与最佳实践>>

编辑推荐

《PHP核心技术与最佳实践》系统归纳和深刻解读PHP开发中的编程思想、底层原理、核心技术、开发技巧、编码规范和最佳实践，为PHP程序员进阶修炼提供全面而高效的指导！

<<PHP核心技术与最佳实践>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>