

# 目录

前言

## 第1篇 基础入门篇

第1章 开启 PHP 职场之旅	2
1.1 编程语言的选择	2
1.1.1 为什么选择 PHP	2
1.1.2 PHP 求职技能需求	3
1.2 选择 Windows、Mac OS 还是 Linux	5
1.2.1 PHP 跨操作系统开发	6
1.2.2 Windows 操作系统	6
1.2.3 Mac OS 操作系统	6
1.2.4 Linux 操作系统	7
1.3 养成良好的开发习惯——多看与多写	9
1.3.1 多看文档	9
1.3.2 多写代码	11
第2章 虚拟机与个性化开发环境搭建	15
2.1 单平台共享多系统——虚拟机	15
2.1.1 虚拟机技术	15
2.1.2 VirtualBox 虚拟机	16
2.2 虚拟机辅助工具——Vagrant	23
2.2.1 Vagrant 简介	24
2.2.2 Vagrant 常用操作	25
2.2.3 Vagrant 常用配置与命令	30
2.3 打造个性化开发环境	34
2.3.1 准备工作	34
2.3.2 安装及配置 Nginx	35
2.3.3 安装及配置 PHP	38
2.3.4 安装及配置 MySQL	40
2.3.5 配置虚拟站点	42



2.3.6	其他常用设置	43
<b>第 3 章</b>	<b>更先进的版本管理工具——Git</b>	<b>46</b>
3.1	SVN 与 Git	46
3.1.1	Git 与 GitHub	46
3.1.2	Git 与 SVN 的异同	47
3.1.3	在 Windows 上安装 Git	49
3.2	Git 常用命令操作	50
3.2.1	创建版本库	50
3.2.2	提交文件到版本库	51
3.2.3	Git 查看版本库信息	52
3.2.4	日志查看与版本回退	53
3.2.5	了解工作区、暂存区和版本库	55
3.3	GitHub 远程仓库	56
3.3.1	在线注册远程仓库	56
3.3.2	本地操作远程仓库	59
3.4	分支、合并与冲突解决	60
3.4.1	分支与合并原理	61
3.4.2	分支与合并实例	62
3.4.3	冲突解决	63
3.5	使用 GitHub Pages 搭建个人博客站点	65
3.5.1	传统博客与 GitHub Pages	65
3.5.2	使用 Hexo 框架生成静态网站	66
3.5.3	推送文章网站到 GitHub Pages	69
3.5.4	GitHub Pages 使用小技巧	73
<b>第 4 章</b>	<b>高效团队协作</b>	<b>76</b>
4.1	沟通和交流很重要	76
4.1.1	术业有专攻——企业即时通信工具	76
4.1.2	文档积累和文件分享	78
4.2	任务分配、代码托管和缺陷管理	81
4.2.1	任务分配	81
4.2.2	代码托管	82
4.2.3	缺陷管理	83
4.3	在线协作绘制流程图——ProcessOn	84
4.3.1	ProcessOn 简介	84
4.3.2	ProcessOn 操作指南	85
4.3.3	ProcessOn 多人协作	88
4.4	GitLab 操作全攻略	90
4.4.1	安装与汉化	90
4.4.2	了解 GitLab 的工作流	94

4.4.3	GitLab 用户和项目管理	97
4.4.4	GitLab 持续集成与自动构建实践	105
<b>第 5 章</b>	<b>好用的 PHP 开发环境——PHPStorm</b>	<b>111</b>
5.1	常用 PHP 源代码开发工具	111
5.1.1	源代码编辑工具简介	111
5.1.2	选择合适的开发工具	113
5.2	使用 PHPStorm 提高代码编写效率	115
5.2.1	PHPStorm 常用快捷操作	116
5.2.2	自定义文件模板和代码片段	123
5.2.3	方法重构与多点编辑	129
5.3	PHPStorm 集成 Xdebug 调试插件	132
5.3.1	安装 PHP Xdebug 扩展	133
5.3.2	在 PHPStorm 中使用 Xdebug 插件调试代码	134
<b>第 6 章</b>	<b>PHP 依赖的自动化管理工具——Composer</b>	<b>138</b>
6.1	现代化的包管理工具	138
6.1.1	了解 Composer 原理	138
6.1.2	Composer 安装与使用	140
6.2	Composer 使用进阶	141
6.2.1	实例：在项目中集成依赖包 PHPMailer	141
6.2.2	认识 composer.json 和 composer.lock 文件	144
6.2.3	Composer 的其他命令操作	147
6.3	提交自定义包到 Composer	150
6.3.1	本地创建 Composer 包	150
6.3.2	提交依赖包到 Composer Packagist	154

## 第 2 篇 框架进阶篇

<b>第 7 章</b>	<b>响应式布局框架——Bootstrap</b>	<b>160</b>
7.1	Bootstrap 入门	160
7.1.1	Bootstrap 简介	160
7.1.2	Bootstrap 核心技术——CSS 预处理脚本	161
7.1.3	响应式布局技术	164
7.2	Bootstrap 实战技巧	166
7.2.1	布局容器与栅格系统	166
7.2.2	Bootstrap 组件快速入门技巧	169
7.3	基于 Bootstrap 的后台模板样式——AdminLTE	171
7.3.1	AdminLTE 简介与安装	171
7.3.2	布局、皮肤与 box 容器	174

7.3.3	实现常用模板——列表	180
7.3.4	实现常用模板——表单	185
<b>第 8 章</b>	<b>ThinkPHP 命令行操作与接口开发实战</b>	<b>193</b>
8.1	ThinkPHP 5 与命令行操作	193
8.1.1	ThinkPHP 简介	193
8.1.2	使用 Composer 创建 ThinkPHP 5 项目	194
8.1.3	ThinkPHP 5 命令行操作	195
8.2	实战：制作一个短地址生成器	199
8.2.1	功能简介	199
8.2.2	程序设计	200
8.2.3	数据表设计	201
8.2.4	程序实现	202
8.3	RESTful API 实战：用户接口权限验证	208
8.3.1	RESTful API 简介	208
8.3.2	实现接口权限验证	209
8.3.3	创建用户表结构与模型	212
8.3.4	开发 RESTful API 接口类	213
8.3.5	测试 RESTful API 接口	216

## 第 3 篇 项目实战篇

<b>第 9 章</b>	<b>内容管理框架实战——基础架构、用户与菜单管理</b>	<b>222</b>
9.1	内容管理框架	222
9.1.1	PHP 常见开发模式	222
9.1.2	系统功能设计	223
9.1.3	程序架构设计	226
9.2	基础模板布局	226
9.2.1	准备工作	227
9.2.2	创建基础布局模板	227
9.3	用户模块——用户登录与验证	233
9.3.1	数据结构设计与基础模板开发	233
9.3.2	完成用户登录操作	235
9.3.3	用户登录状态验证与注销登录	237
9.4	菜单模块	241
9.4.1	数据结构设计	241
9.4.2	获取菜单数据列表	243
9.4.3	后台菜单展示	246
9.4.4	菜单管理	247

---

第 10 章 内容管理框架实战——配置和权限管理	257
10.1 配置管理	257
10.1.1 程序与数据结构设计	257
10.1.2 配置列表管理	258
10.1.3 配置编辑管理	260
10.1.4 配置使用	263
10.2 权限管理——概念、程序设计与数据库设计	266
10.2.1 RBAC 权限管理	266
10.2.2 RBAC 权限管理程序流程与功能设计	269
10.2.3 基于 RBAC 权限管理的数据库设计	270
10.3 权限管理——角色分组、节点授权与用户模块	271
10.3.1 权限角色管理	271
10.3.2 用户授权	280
10.4 权限管理——权限验证	283
10.4.1 权限处理类	283
10.4.2 实现权限验证	287
第 11 章 Crontab 计划任务管理	292
11.1 常见计划任务实现方法	292
11.1.1 PHP 脚本实现计划任务	292
11.1.2 使用系统级别的计划任务工具	293
11.2 Crontab 入门	295
11.2.1 Crontab 使用教程	295
11.2.2 用 Crontab 实现 PHP 文件定时写入	296
11.3 实现计划任务管理模块	297
11.3.1 程序流程与数据结构设计	297
11.3.2 计划任务管理	299
11.3.3 开发定时任务执行脚本	303
第 12 章 基于 Redis 队列的商城抢购系统	311
12.1 高并发应用场景分析	311
12.1.1 高并发场景解决方案	311
12.1.2 高性能抢购系统设计	314
12.2 Redis 常用操作	316
12.2.1 Redis 简介	316
12.2.2 Redis 常用命令操作	317
12.2.3 Redis 消息发布/订阅机制	322
12.2.4 可视化管理 Redis 数据——phpRedisAdmin	324
12.3 实现简单商城网站	327
12.3.1 程序设计与数据库设计	327
12.3.2 商城首页	328

12.3.3	用户注册、登录	330
12.3.4	商品发布	334
12.3.5	商品详情页	337
12.4	完善商城抢购系统逻辑	339
12.4.1	安装使用 PHP Redis 扩展	339
12.4.2	使用 PHP 实现 Redis 订阅/发布实例	341
12.4.3	实现抢购队列消费脚本——订单处理	343
12.4.4	实现抢购入队操作——抢购处理	347
12.4.5	抢购系统部署与使用	353

第1篇 基础入门篇

# 第1篇 基础入门篇

- ▶▶ 第1章 开启 PHP 职场之旅
- ▶▶ 第2章 虚拟机与个性化开发环境搭建
- ▶▶ 第3章 更先进的版本管理工具——Git
- ▶▶ 第4章 高效团队协作
- ▶▶ 第5章 好用的 PHP 开发环境——PHPStorm
- ▶▶ 第6章 PHP 依赖的自动化工具——Composer

# 第 1 章 开启 PHP 职场之旅

对于职场新人来说，从了解 PHP 开发市场需求开始，到在主流系统下快速搭建集成开发环境，再到解读团队合作的新方法，本书都将给你带来不一样的体验和帮助。本章主要讲解如何利用 PHP 技术快速开始你的工作等内容。

## 1.1 编程语言的选择

互联网技术的火热发展，使得越来越多的人进入到计算机编程的行业。但无论开发什么样的应用，都需要熟悉和掌握一门或多门编程语言。而如何在众多编程语言中准确选择一个适合自己的语言，就变得非常重要。

### 1.1.1 为什么选择 PHP

在各大技术社区、论坛，甚至技术群里，PHP 经常被戏称为“世界上最好的语言”，虽然这多少带有一些调侃的意味，但由此可以看出这门语言的火热程度。越来越多的人通过学习并使用 PHP 开发语言构建出了优秀的互联网应用。其主要优势介绍如下。

#### 1. PHP语言市场占有率高

根据国外网站 W3Techs.com 的不完全统计，截至 2017 年 7 月，PHP 语言是目前最为流行的服务器端编程语言，市场份额超过其他语言之和，稳居第一，同时其也是当月新增站点最多的语言。参考数据如图 1-1 所示。

© W3Techs.com	usage	change since 1 July 2017
1. PHP	82.7%	
2. ASP.NET	14.9%	-0.1%
3. Java	2.6%	
4. static files	1.5%	
5. ColdFusion	0.6%	

percentages of sites

a) 服务器端占有率

© W3Techs.com	sites
1. PHP	171
2. JavaScript	39
3. ColdFusion	8

daily number of additional sites  
in the top 10 million

b) 新增站点数的统计

图 1-1 W3Techs.com 对服务器端占有率和新增站点数的统计

如此高的市场占有率和站点增长速度直接影响了市场需求，无论是传统互联网企业，还是新兴的创业团队，都会首选 PHP 语言作为服务器端的脚本语言。所以，如果想找一个偏向于服务器端编程的工作，PHP 语言不失为最佳选择。

## 2. PHP语言技术体系丰富

学习 PHP，不仅仅是学习 PHP 语言，我们还要学习与之密切相关的常用技术。其中，LAMP 或 LNMP 是目前最为流行的组合，其每个字母所代表的含义如下。

- L：代表 Linux 开源、高性能的特点让其成为广泛流行的服务器操作系统。
- A 和 N：代表 Apache 和 Nginx，是目前市场占有率最高的两款服务器软件。
- M：代表 MySQL，是 PHP 的好搭档，著名的关系数据库管理系统。
- P：代表 PHP，是服务于 Web 应用的脚本语言，也是本书的主角。

而对上述软件或者系统稍有了解的读者就会知道，它们当中的每一个在其领域内都是大名鼎鼎的，值得我们不断地深入学习和使用。

PHP 体系常见技术如图 1-2 所示。



图 1-2 PHP 体系常见技术

## 3. PHP从业者有更高的收入水平

如今，互联网与计算机技术所在的行业得到高速发展，仅仅通过国家统计局 2016 年发布的相关报告就可以看出，计算机行业已经超过金融业成为年平均工资最高的行业，虽然该行业的竞争也越来越激烈，但是收入水平也在不断提高。

### 1.1.2 PHP 求职技能需求

企业或开发团队对于 PHP 开发者的要求会依据职位的不同而有所区别。很多时候开发者掌握的技术虽然相对难以量化，但我们仍然可以根据这些职位的需求总结出各个阶段 PHP 开发工程师的技能需求。

#### 1. 初级技能需求

对于 PHP 初级工程师，所需要掌握的技能一般不会要求太高，申请该职位的求职者多为计算机行业的初学者或刚走出校园的应届毕业生。例如，在招聘平台上，国内某一线互联网企业对申请该职位的应届毕业生的招聘要求如图 1-3 所示。

从图 1-3 中的招聘要求里可以看出，除了对 PHP 开发的经验有要求外，还需要熟悉常

见的 PHP 开发框架、MySQL 和 Linux 操作系统，并且还可以独立地搭建开发环境。由此可见对 PHP 初级工程师的常规要求如下：

- 有 PHP 语言开发经验；
- 有 PHP 框架开发经验；
- 可独立搭建 PHP 开发环境；
- 有 MySQL 实战经验；
- 有 Linux 实战经验。

要求：  
1. 有超过半年的PHP开发经验，并使用超过一个php开发框架；  
2. 熟悉LAMP的环境，有协作开发能力，熟练使用Linux命令部署；  
3. 熟悉数据库理论知识，有MySQL实践经验；  
4. 热爱互联网行业，有志于此领域长期发展；  
5. 对需要处理复杂逻辑流程的业务系统有兴趣。

图 1-3 PHP 初级工程师常见任职要求

提示：而实际需求可能会更加复杂，若你能掌握更多的编程语言或者做过更多的实战项目，无疑是增加了求职成功的几率。

## 2. 中级技能需求

PHP 开发中级工程师一般在初级工程师的基础上，要求具有更加丰富的项目经验，能够独立完成一些功能模块，所需要掌握的技术也更多。下面还是以某平台企业招聘的要求为例，来查看当前的企业对 PHP 中级工程师的任职要求，如图 1-4 所示。

任职要求：  
1. 3年以上互联网应用开发经验、精通PHP编程；  
2. 熟悉 (Laravel、ThinkPHP、YII、CI、Yaf) 等开源框架中的一种或多种；  
3. 熟悉一种或多种开源产品；  
4. 熟练掌握接口开发 (OAuth协议、微信、各大开放平台)；  
5. 熟悉网站 (软件) 开发流程；  
6. 熟悉Linux环境搭建GIT、SVN部署；  
7. 熟悉千万PV网站架构设计与优化；  
8. 熟悉NoSQL数据库 (MongoDB、Redis) 的应用；  
9. 熟悉手机应用、HTML5开发者优先。

图 1-4 PHP 中级工程师常见任职要求

从这个招聘要求中可以看出，PHP 中级工程师至少需要掌握以下技能：

- 3 年及以上的项目实战经验；

- 熟练使用多种 PHP 开发框架；
- 熟悉多种开源产品；
- 熟练使用第三方开发平台，如支付宝、微信公众平台等；
- 熟悉常用版本控制软件及流程；
- 熟悉 NoSQL 数据库技术，如 Redis 等；
- 熟悉移动应用开发。

掌握以上技能仅仅满足了大部分企业的招聘要求。如果想提高自己的竞争力，提高应聘成功的几率，则需要学习更多的技术和内容。

### 3. 常见技能需求

前面我们介绍了初、中级工程师在企业应聘中应具备的基本技能。而高级工程师所需要掌握的技能较多，也更细分和更专业，在这里就不再专门列出，大家根据自己的职业规划，选择适合自己的成长路径即可。

最后在这里总结一下初、中级 PHP 工程师所需要的常规技能如图 1-5 所示，无论是刚毕业学业，还是正在找工作的求职者，下面的技能应当熟练掌握。

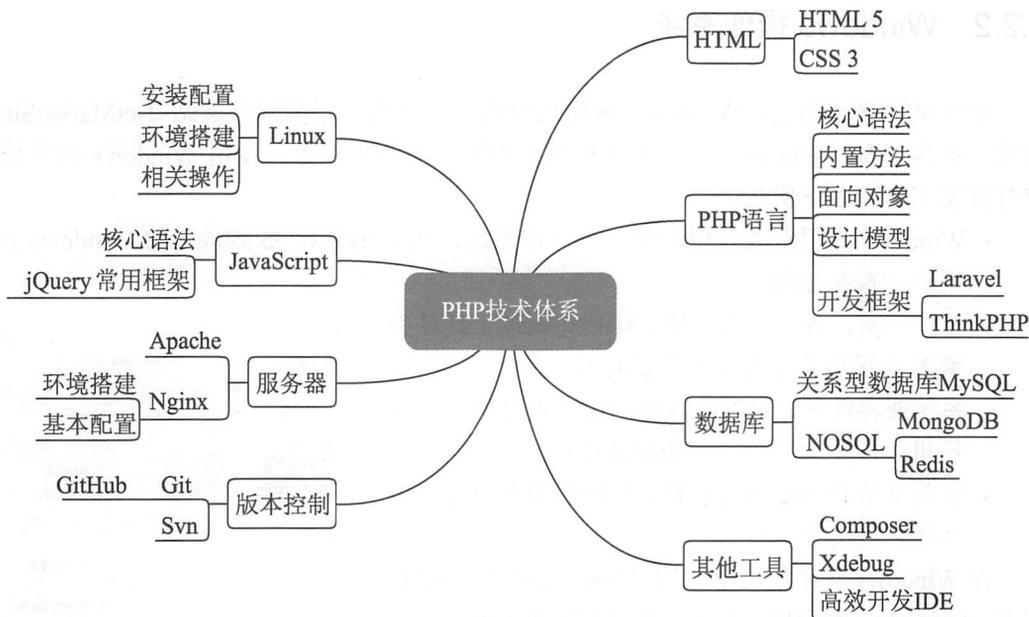


图 1-5 PHP 技术体系

## 1.2 选择 Windows、Mac OS 还是 Linux

Windows 可以说是桌面系统的不二选择，Linux 则是服务器市场的王者，而开发移动

端 iOS 应用则必须使用 Mac OS 系统。不同的操作系统应用场景不同，导致了开发者经常需要跨操作系统平台开发。

## 1.2.1 PHP 跨操作系统开发

PHP 跨平台应用开发的场景通常体现在开发环境与测试、应用环境的不同上。例如，在 Windows 系统下开发，而实际的应用环境为 Linux 操作系统，则有如下潜在问题：

- Windows 系统不严格区分大小写，而 Linux 系统则严格区分，因此可能导致代码在本地能正常运行，而部署到线上就有可能出错。
- 虽然大部分 PHP 扩展都有 Windows 系统版本，安装起来也很简单，只需要复制扩展名为.dll 的动态运行库文件到指定目录即可，但是某些 PHP 扩展只有 Linux 版本，如 Swoole 框架。
- 虽然有一些潜在的兼容性问题，但是在不同的操作系统下进行开发都有各自的优势，应用场景不同，所需要的开发系统环境也不同。

## 1.2.2 Windows 操作系统

对于桌面平台而言，Windows 操作系统依然是首选。根据国外网站 NetMarketShare 的统计报告来看，Windows 操作系统市场份额依旧在 90%以上。使用 Windows 操作系统进行开发工作有以下优势：

- Windows 是很多人使用的第一个操作系统，从 Windows 95 到最新的 Windows 10，用户一直在使用。
- 软件丰富，各种集成环境、IDE 和调试工具种类繁多，开发者总能找到需要的那一个。
- 系统兼容性好，可以允许应用在大部分的个人计算机上，驱动完善，系统更新迅速。
- 完美支持 Office 办公套件，是最佳的办公生产力环境，没有之一。

在 Windows 下开发 PHP 应用，集成环境推荐 WAMP 或者 phpStudy，在常用的搜索引擎里都可以方便地获取与安装。其中，在 phpStudy 中可以灵活地进行非服务器版本和服务器版本切换，如图 1-6 所示。



图 1-6 phpStudy 中可以方便地切换运行模式

## 1.2.3 Mac OS 操作系统

自从 2007 年苹果公司发布第一代 iPhone 手机至今已经有十余年。其间接带动了 Mac OS

系统的市场占有率，使其成为第二大桌面操作系统。使用 Mac OS 操作系统进行 PHP 应用开发，通常有以下优势：

- Mac OS 采用的是 UNIX 系统内核，所以大部分操作习惯和 Linux 操作系统类似。
- 因为系统相近，所以一些 Linux 系统下的开源软件和包等都有 Mac OS 版本。
- 相比 Windows 系统，Mac OS 没有注册表机制，软件安装和卸载更加简洁。
- 有好用的包管理工具——HomeBrew。
- Mac OS 具有更好用的命令行终端体验。

在 Mac OS 上进行 PHP 开发，快速搭建 PHP 集成环境可以使用 XAMPP 或者 MAMP，其中 XAMPP 是跨平台软件，在 Windows 和 Linux 上都有对应的版本。而 MAMP 集成工具开发包则由 Mac OS 系统独占，其工具包如图 1-7 所示。



图 1-7 Mac OS 上常见的 PHP 开发集成环境工具包

## 1.2.4 Linux 操作系统

Linux 操作系统自发明之日起，其核心思想就是自由和分享，不仅其平台上涌现出了大量的优秀项目和开源软件，Linux 更是已经渗入到了现代社会的各个领域，被不断地应用和借鉴。例如，目前在移动市场上占有率最多的 Android 系统，就是基于 Linux 内核开发的系统。

在 Linux 操作系统上开发 PHP 应用，通常有以下优势：

- 可以完美地模拟环境，减少因为跨操作系统而带来的各种问题。
- 各个发行版自带了包依赖管理工具，如 Ubuntu 下的 Apt 软件包管理工具，或者 CentOS 下的 Yum 软件包管理工具等。
- 软件的编译和安装更加方便。
- 有丰富且优秀的开源软件和工具。

若需要在 Linux 下进行 PHP 开发，首选的开发编辑器为 phpStorm，开发环境在这里推荐 LNMP 集成环境工具包（官方网址是 <https://lnmp.org>）。虽然 LNMP 集成环境工具包没有图形界面，只能在系统命令行下使用，但是只要按照提示即可快速完成安装。例如，安装过程中使用数字选择安装哪个版本的 MySQL 数据库就非常方便，如图 1-8 所示。

```
You have 5 options for your DataBase install.
1: Install MySQL 5.1.73
2: Install MySQL 5.5.56 (Default)
3: Install MySQL 5.6.36
4: Install MySQL 5.7.18
5: Install MariaDB 5.5.56
6: Install MariaDB 10.0.30
7: Install MariaDB 10.1.23
0: DO NOT Install MySQL/MariaDB
Enter your choice (1, 2, 3, 4, 5, 6, 7 or 0):
```

图 1-8 通过选择序号安装 MySQL 数据库

安装完成后的状态提示也通俗易懂，其安装成功后的界面如图 1-9 所示。

```
===== Check install =====
Checking ...
nginx: OK
MySQL: OK
PHP: OK
PHP-FPM: OK
Clean src directory...
+-----+
| LNMP V1.4 for CentOS Linux Server, Written by Licess |
+-----+
| For more information please visit https://lnmp.org |
+-----+
| lnmp status manage: lnmp {start|stop|reload|restart|kill|status} |
+-----+
| phpMyAdmin: http://IP/phpmyadmin/ |
| phpinfo: http://IP/phpinfo.php |
| Prober: http://IP/p.php |
+-----+
| Add VirtualHost: lnmp vhost add |
+-----+
| Default directory: /home/wwwroot/default |
+-----+
| MySQL/MariaDB root password: root |
+-----+
+-----+
| Manager for LNMP, Written by Licess |
+-----+
| https://lnmp.org |
+-----+
nginx (pid 17990 17989 17988) is running...
php-fpm is runing!
SUCCESS! MySQL running (18522)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 127.0.0.1:11211 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:21 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:3306 0.0.0.0:* LISTEN
tcp6 0 0 :::21 :::* LISTEN
tcp6 0 0 :::22 :::* LISTEN
Install lnmp takes 53 minutes.
Install lnmp V1.4 completed! enjoy it.
```

图 1-9 Linux 下常见的 PHP 开发集成环境工具包安装成功

## 1.3 养成良好的开发习惯——多看与多写

对每一个开发者来说，如何快速融入开发团队是一个不小的挑战。团队协同开发对于开发者的沟通能力、技术能力与学习能力都有较高的要求。

### 1.3.1 多看文档

#### 1. 阅读文档

开发团队大多会有积累的技术文档，可以找一找当前团队中是否有以下文档（但不限于），通读这些文档可以加快对手上工作的理解，提高工作效率。

- 编码规范；
- 版本管理工具说明；
- 项目开发经验总结；
- 常用工具下载地址；
- 项目数据结构说明书；
- 开发技术官方手册；
- 测试用例文档。

当然，以上这些文档不一定每个团队都有。你也可以通过阅读项目文件，或者与团队成员交流等方式来继续了解项目情况。同时，通读各种技术的官方文档也非常必要。例如，如图 1-10 所示为 PHP-FIG 官网已经通过的编码规范列表。

已通过				
序号	标题	撰稿者	协调者	发起人
1	基础编码规范	Paul M. Jones	N/A	N/A
2	编码风格规范	Paul M. Jones	N/A	N/A
3	日志接口规范	Jordi Boggiano	N/A	N/A
4	自动加载规范	Paul M. Jones	Phil Sturgeon	Larry Garfield
6	缓存接口规范	Larry Garfield	Paul Dragoonis	Robert Hafner
7	HTTP 消息接口规范	Matthew Weier O'Phinney	Beau Simensen	Paul M. Jones

图 1-10 常见的 PHP 编码规范文档

除了技术类的文档，还可以查看需求类的文档。需求类文档通常有以下两种：

- 产品说明文档；

- 项目需求文档。

其他文档一般是企业制度、保密协议和员工晋升规范等，也建议通读，以提升对企业与团队的进一步了解。

总的来说，一开始要看的東西很多，如图 1-11 所示。

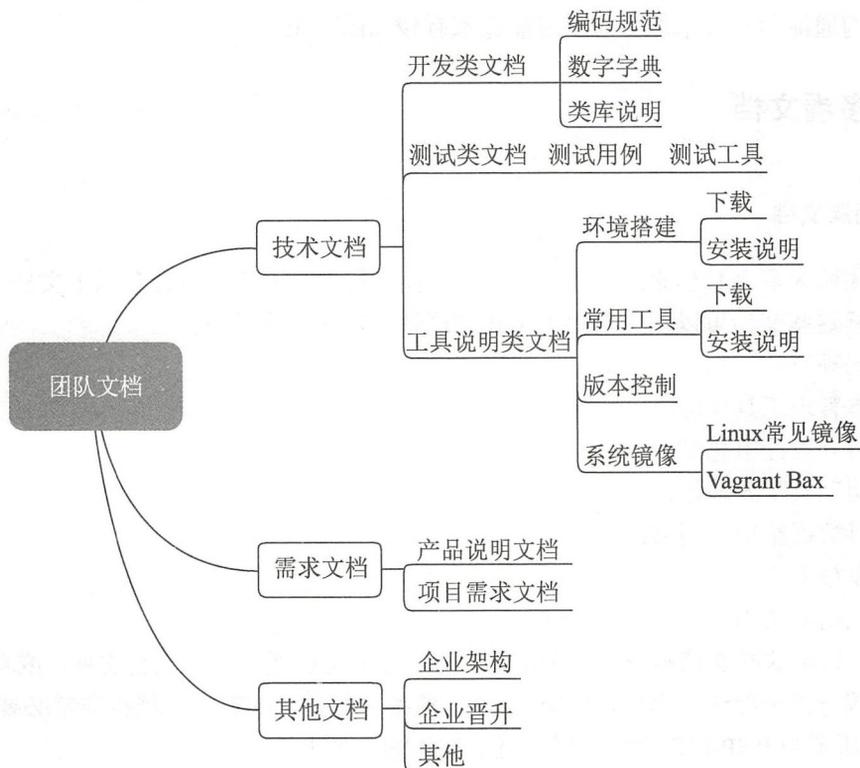


图 1-11 常见团队文档分类说明

## 2. 阅读代码

很多时候，阅读代码的难度甚至要高于编写代码，这通常是因为：

- 不符合规范的编码；
- 不合理的架构设计；
- 风格迥异的代码风格；
- 程序异常；
- 功能缺失。

另外，不熟悉的数据库结构、程序架构和不明确的需求，也大大增加了代码阅读的难度。

虽然困难重重，但只要仔细阅读需求文档和技术文档，同时基于某一个問題或某一

个模块功能进行代码调试，就可以提高代码阅读的效率。常见的代码阅读流程如图 1-12 所示。

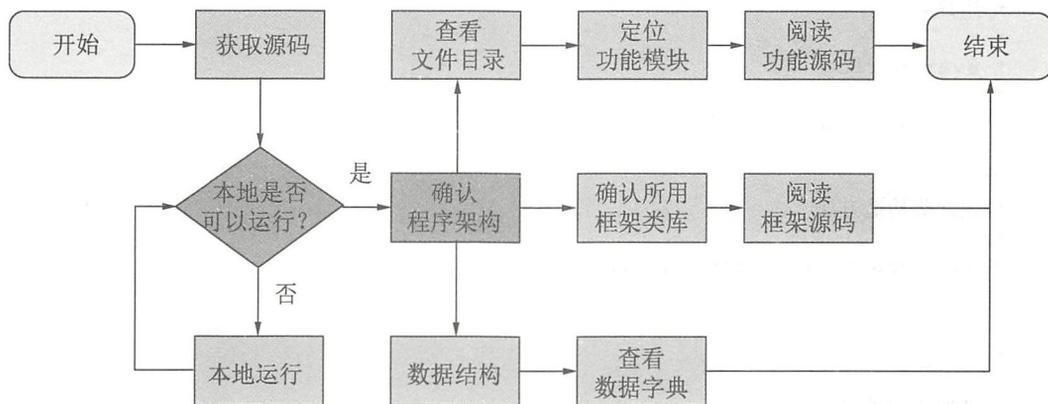


图 1-12 常见代码阅读流程

### 1.3.2 多写代码

PHP 语言本身的语法比较松散，稍不留神，平时编码的时候就有可能写出“异形”代码，这无形中提高了对开发者的要求。所以在正式编程之前，一定要打好 PHP 语言基础，仔细阅读编码规范，同时学习其他项目中的优秀代码。在编码的同时，需要注意以下几点。

#### 1. 规范注释

良好的注释习惯，不仅可以让别人更容易读懂你的代码，也避免出现“这是我写的代码么？”这样的疑问。代码注释的风格因人而异，但只要结构合理、通俗易懂即可满足要求。

除了开发团队编码规范要求外，注释一般可以分为文件注释、类注释和方法注释 3 种。这里参考 PHPDoc 的注释规范，它是一种注释 PHP 代码的正式标准，在 IDE 中提供代码完成、类型提示和除错功能，而且支持面向对象和面向过程的不同代码风格。

下面是一个典型的带有 PHP 注释的用户类（User）：

```

<?php
/**
 * 用户类
 *
 * @category   PHP
 * @author    dz5362 <wangjialin.bj@gmail.com>
 * @version   Release: 1.1
 */

```

```

class User
{
/**
 * 用户名
 *
 * @var string
 */
private $username = '';

/**
 * 用户类构造方法
 */
public function __construct()
{
}

/**
 * 获取用户姓名
 *
 * @return string
 */
public function getUserName()
{
return $this->username;
}

/**
 * 设置用户名
 *
 * @param string $username 用户名
 * @return mixed
 */
public function setUserName($username = '')
{
return $this->username = $username;
}
}

```

从实例中可以看出，无论是类注释，还是方法注释，都使用了类似“@标签名 说明内容”的方式，对注释进行描述，多个标签描述行组成了注释块。而这样的标签还有很多，开发者使用的时候可以灵活搭配。PHPDoc 提供的常见注释标签说明如表 1-1 所示。

表 1-1 PHPDoc提供的常见注释标签说明

标 签	实 例	说 明
@abstract		抽象类的变量和方法
@access	public, private or protected	文档的访问和使用权限。@access private表明这个文档是私有的
@author	小明 <xiaoming@qq.com>	文档作者信息
@copyright	名称 时间	文档版权信息

(续)

标 签	实 例	说 明
@deprecated	Version	文档中被废除的方法
@deprec		同@deprecated
@example	/path/to/example	文档的外部保存示例文件的位置
@exception		文档中方法抛出的异常，也可参照@throws
@global	类型: \$globalvarname	文档中的全局变量及有关的方法和函数
@ignore		忽略文档中指定的关键字
@internal		开发团队内部信息
@link	URL	类似于license，但还可以通过访问link地址找到文档中更多的详细信息
@name	变量别名	为某个变量指定别名
@magic		PHPDoc兼容phpDocumentor的标签
@package	封装包的名称	一组相关类、函数封装的包名称
@param	如[\$username] 用户名	输入变量含义注释
@return	如返回bool	输出函数返回结果描述，一般不用在void（空返回结果的）的函数中
@see	如Class Login()	文件关联的任何元素（全局变量，包括页面、类、函数、定义、方法、变量）
@since	version	记录什么时候对文档的哪些部分进行了更改
@static		记录静态类、方法
@staticvar		在类、函数中使用的静态变量
@subpackage		子版本
@throws		抛出的异常
@todo		表示文件未完成或者要完善的地方

## 2. 善于代码重构

很多代码在初次编写的时候，虽然可以正常运行，但可能会因为程序设计不完善或交付时间短等各种原因，导致性能低下和难以阅读等问题。这时候就需要对这一部分代码进行重构操作。

重构代码一般应满足以下几个目的：

- 改善程序设计，提高灵活性；
- 增强代码阅读性，使代码更易理解；
- 提高性能，不仅程序运行更快，编写也更快；
- 减少已知和未知的 BUG。

在重构代码的时候，可以着重注意以下几个方法：

- 提高代码复用即，一个代码片段出现过多次，就可以考虑封装其为公共方法；
- 采用驼峰法命名即，同样的方法，命名为 `getUserName()` 比命名为 `a()` 更易于阅读（注意方法依赖）；
- 规范注释。

提示：一定不要为了重构代码而重构代码，具体问题具体分析。代码重构可以说是一门艺术，这里推荐大家阅读《重构：改善既有代码的设计》一书。

### 3. 文档编写

在阅读文档和编写代码的时候，也可以自己积累笔记和文档。除了记录项目的开发配置、版本库地址等必备信息外，还需要把常见问题和经验记录下来，这样下次再遇到相同的问题时就不用重复寻找解决方案了。

常用的笔记软件如图 1-13 所示。

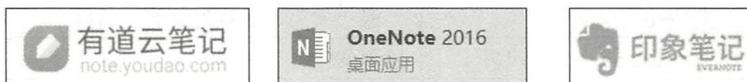


图 1-13 常见的笔记类工具

## 第 2 章 虚拟机与个性化开发环境搭建

“工欲善其事，必先利其器”。为了提升开发效率，除了使用更加强大的开发硬件外，系统软件 and 开发环境也同样重要。本章主要讲解如何使用虚拟机工具安装虚拟系统，方便多系统的使用，同时借助 Vagrant 工具提升使用体验，最终打造个性化的开发环境。

### 2.1 单平台共享多系统——虚拟机

虽然在第 1 章中提到了如何在各个系统中快速安装 PHP 集成开发环境，但若想真正地模拟生产环境，最佳的平台还是 Linux 平台。但很多时候，因为软/硬件和驱动程序的原因，在 PC 上安装 Linux，可以说是一个灾难。那么有没有办法可以不用安装其他操作系统直接使用呢？使用虚拟机就是一个很好的解决办法。

#### 2.1.1 虚拟机技术

虚拟机（Virtual Machine）简单来说，就是在当前操作系统下通过虚拟机软件安装一个全新的操作系统，而这两个操作系统可以做到互不影响和相对独立。

虚拟机运作机制如图 2-1 所示。

虚拟机软件中运行的操作系统，被称为虚拟系统。它具有真实系统的一切功能，可以安装软件和进行其他操作，还可以方便地切换到真实系统中。

除了 PC（个人计算机）系统外，虚拟机技术也被广泛地运用到了云服务、虚拟主机上。例如比较有名的谷歌云、亚马逊云和阿里云等，在核心技术上都使用了虚拟机技术。

常见的 PC 虚拟机软件有 Vmware、VirtualBox、Parallels Desktop，如图 2-2 所示。

有了虚拟机技术，就可以在常用操作系统上虚拟出一个生产力环境的虚拟系统。在虚拟系统里面进行程序的开发和编译，这样就可以大为减少因为操作系统的差异而导致的各种问题。

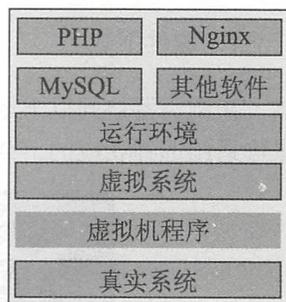


图 2-1 虚拟机运作机制



图 2-2 常见的虚拟机软件

虽然虚拟机是一个很好的解决方案，但其本身也有如下一些不足：

(1) 占用真实系统资源，对 PC 硬件要求高。因为虚拟机本质上是一个应用软件，只要运行就会消耗系统资源，更不用说还要去虚拟一个操作系统，对硬件中的 CPU 性能、运行内存大小和硬盘空间性能都会比较敏感。若宿主机性能太弱，则虚拟系统体检就会不友好。

(2) 软件运行速度更慢。因为虚拟系统是通过软件模拟的方式运行的，相比直接运行在硬件之上的真实系统，其性能只能是无限接近而无法达到或者超越。因此一些对性能敏感的软件，不建议运行在虚拟机上，如一些大型游戏、渲染工具等，直接在真实系统上运行才可以发挥出硬件的真实效率。

目前，个人计算机的性能大部分都可以满足虚拟机运行的基本要求。

## 2.1.2 VirtualBox 虚拟机

虽然商业虚拟机程序性能更高，但因为 VirtualBox 免费开源，支持的系统平台多，软件本身所占用的系统资源少，所以在这里选择 VirtualBox 作为实例进行讲述。

### 1. 软件下载

在浏览器访问以下地址 <https://www.virtualbox.org/wiki/Downloads>，就可以根据不同的操作系统选择相应的下载安装文件，如图 2-3 所示。

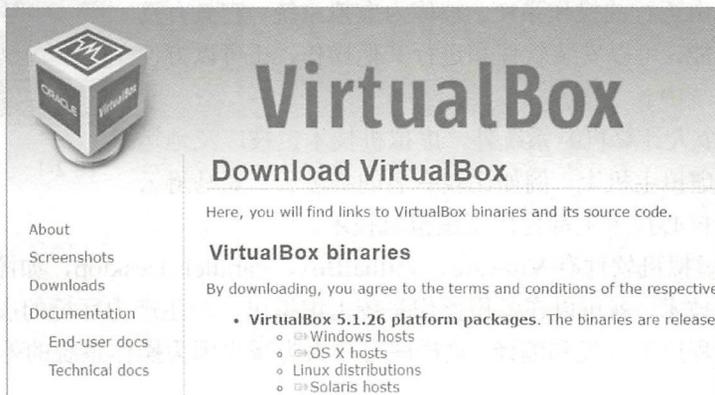


图 2-3 获取 VirtualBox 安装包

## 2. 软件安装

以 Windows 操作系统为例, 所需的安装包文件在单击网页上的 Windows hosts 超链接后即可自动下载, 下载完成后, 双击扩展名为 .exe 的安装文件即可开始安装。

VirtualBox 的安装步骤和一般的应用软件没有太大的区别, 只需要按照提示依次单击 Next 按钮, 进行下一步安装即可。

安装完成的提示界面和软件的主界面如图 2-4 所示。

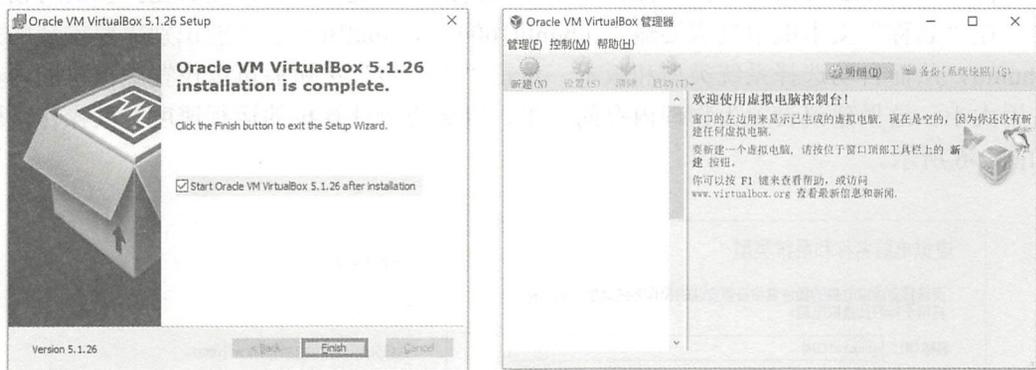


图 2-4 等待安装完成后打开 VirtualBox 的主界面

## 3. 获取虚拟系统镜像

VirtualBox 安装完成后, 需要系统镜像才可以安装虚拟系统。在这里选择 Linux 的发行版 Ubuntu 作为实例的安装镜像。Ubuntu 系统在国内有对中文支持更好的本地化的镜像 (Ubuntu Kylin) 可供下载, 地址是 <http://www.ubuntukylin.com/downloads/>, 在下载页面中找到“银河麒麟社区版”区域, 单击“桌面版”按钮即可下载扩展名为 .iso 的系统镜像到本地, 如图 2-5 所示。



图 2-5 获取 Ubuntu Kylin 系统的安装镜像

提示: Ubuntu Kylin 操作系统分为桌面版与服务器版, 两者最主要的区别在于是否带有图形化操作界面。

#### 4. 创建虚拟主机

如今攒一台 PC 主机不需要再奔波于电脑城, 只需要在电商网站上动动鼠标就可完成选购。同理, 创建一台虚拟主机也十分方便。

打开 VirtualBox 软件的主界面, 单击左上角的“新建”按钮, 随后在弹出的对话框中, 在“名称”文本框中只需要输入 ubuntu1604, VirtualBox 会智能识别需要安装的是 Ubuntu, 从而自动选择系统类型和版本。单击“下一步”按钮后可以设置虚拟机的内存占用大小。这里建议设置为物理内存的一半, 以免影响计算机的运行速度。其操作步骤如图 2-6 所示。

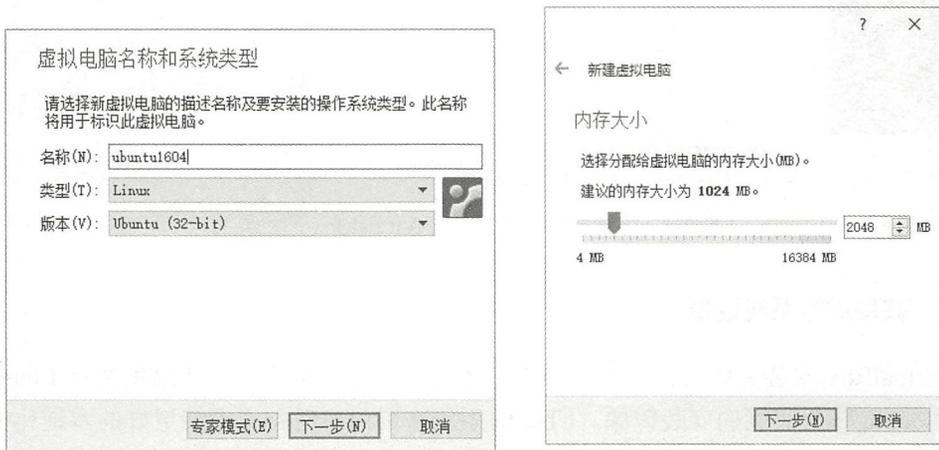


图 2-6 选择系统版本与设置虚拟内存大小

虚拟机的名称可以自定义, 但在本实例中, 操作系统类型需要的是 Linux, 发行版本为 Ubuntu (64-bit)。

#### 5. 创建虚拟硬盘

设置完成虚拟内存大小后, 单击“下一步”按钮即可创建虚拟硬盘。虚拟磁盘是虚拟机安装操作系统的位置, 可以根据物理机实际硬盘的大小合理分配。为了方便操作, 本实例操作使用 VirtualBox 提供的默认设置。

下面几步中首先选择“现在创建虚拟硬盘”, 然后设置文件类型、分配类型、文件位置与大小, 具体步骤如图 2-7 所示。

虚拟硬盘创建完成后, 可以在 VirtualBox 的主界面左侧列表中看到虚拟主机信息, 如图 2-8 所示。



图 2-7 创建虚拟机虚拟硬盘步骤

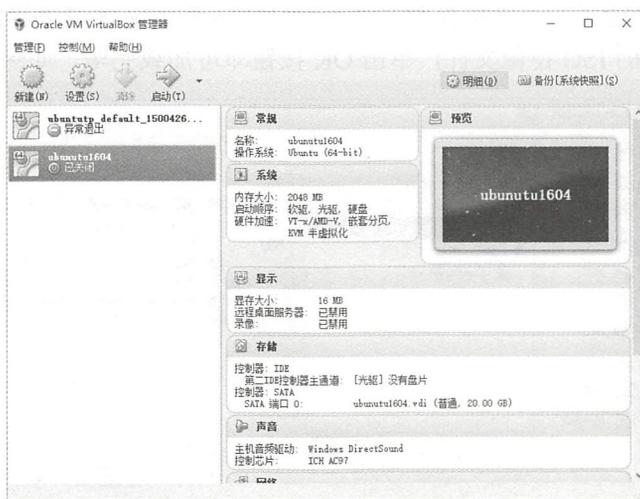


图 2-8 查看已经创建完毕的虚拟主机信息

 提示：合理对虚拟机命名可以更好地进行相应管理。

## 6. 安装虚拟系统

安装虚拟系统前需要先加载系统镜像。在 VirtualBox 主界面中使用鼠标选中 ubuntu1604 虚拟机，单击“设置”按钮可以进入虚拟机的配置页面。配置页面包含了对虚拟机硬件的常见配置，其中包括磁盘、处理器、网络连接和文件管理等，操作步骤如下。

### (1) 选择虚拟镜像

在这里需要选择左侧“存储”菜单，在“存储树”区域中，发现选项“控制器：IDE”提示“没有盘片”，这说明系统镜像此时还未加载，如图 2-9 所示。

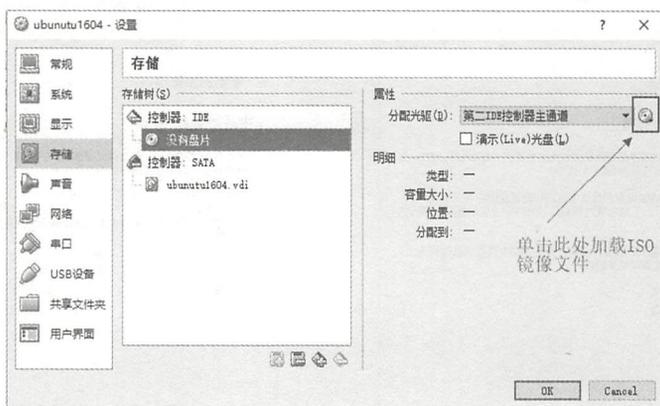


图 2-9 查看系统镜像是否加载

单击“没有盘片”，出现属性详情页。单击右侧“分配光驱”后的光盘图标，在弹出的对话框中单击“选择一个虚拟光盘文件”后，进入系统文件浏览界面，选择已经下载 Ubuntu Kylin 系统的 ISO 镜像文件，单击 OK 按钮即可加载完毕，如图 2-10 所示。

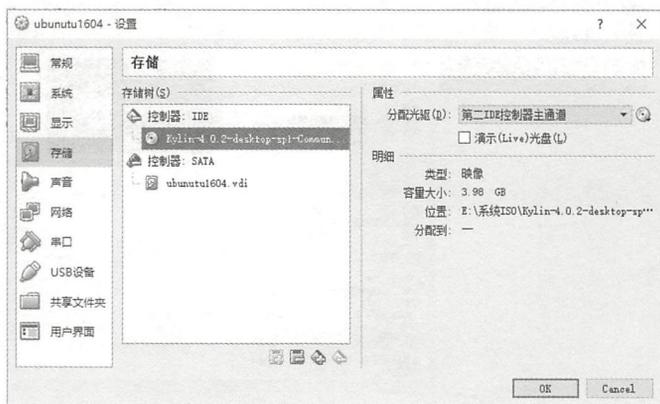


图 2-10 查看已经加载完成的系统镜像

加载完毕后，在“存储树”列表中的“控制器：IDE”下，会出现加载完成的系统镜像信息，单击 OK 按钮可以返回虚拟机列表。

### (2) 启动虚拟机。

单击“启动”按钮即可开始虚拟系统的安装，如图 2-11 所示。

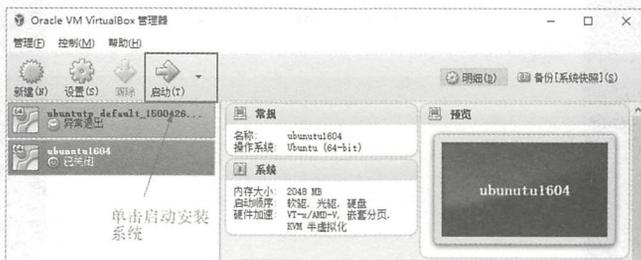


图 2-11 在主界面启动虚拟系统的安装

### (3) 虚拟系统安装

虚拟机随后会开启一个 Windows 窗口，模拟显示器的显示效果。此时系统在短暂加载后，会出现 Ubuntu Kylin 系统的安装界面。使用键盘的“↓”按键可以选择需要的安装选项，这里选择“安装银河麒麟操作系统 (I)”开始虚拟系统的安装，如图 2-12 所示。



图 2-12 开始安装 Ubuntu Kyin 虚拟系统

### (4) 虚拟系统语言设置

选择“中文（简体）”后，单击“继续”按钮执行下一步安装，如图 2-13 所示。

### (5) 虚拟系统安装类型

单击“继续”按钮执行下一步，因为本实例只是快速安装演示，所以选中“快速安装 Kylin”选项作为安装模式，如图 2-14 所示。

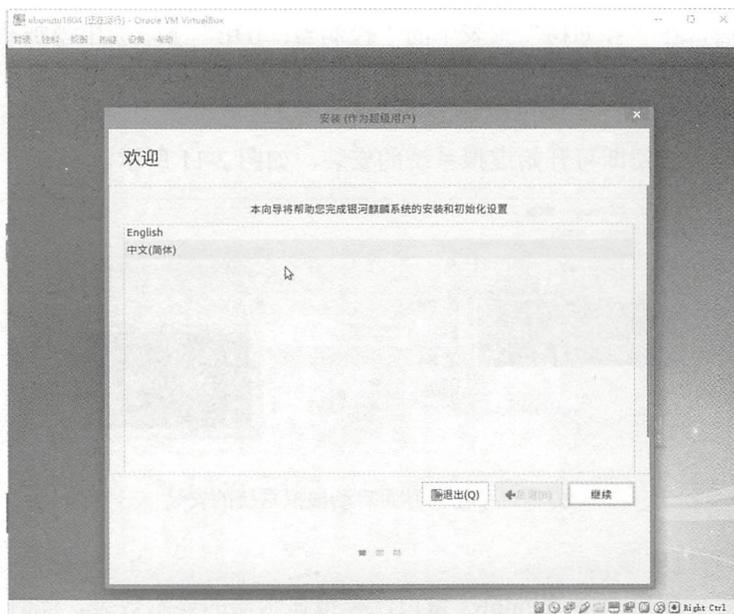


图 2-13 设置虚拟系统安装语言

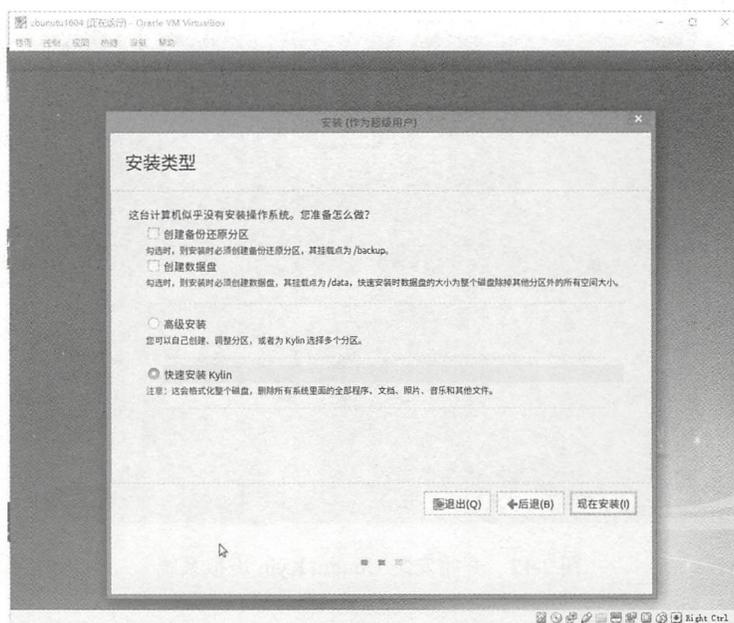


图 2-14 选择快速安装模式可以简化安装

### (6) 虚拟系统磁盘与用户名设置

单击“现在安装”按钮，进入下一步。此时系统会提示是否将改动写入磁盘，单击“继

续”按钮后，提示输入姓名、用户名、计算机名和密码，需要合理设置并妥善保管。操作步骤如图 2-15 所示。

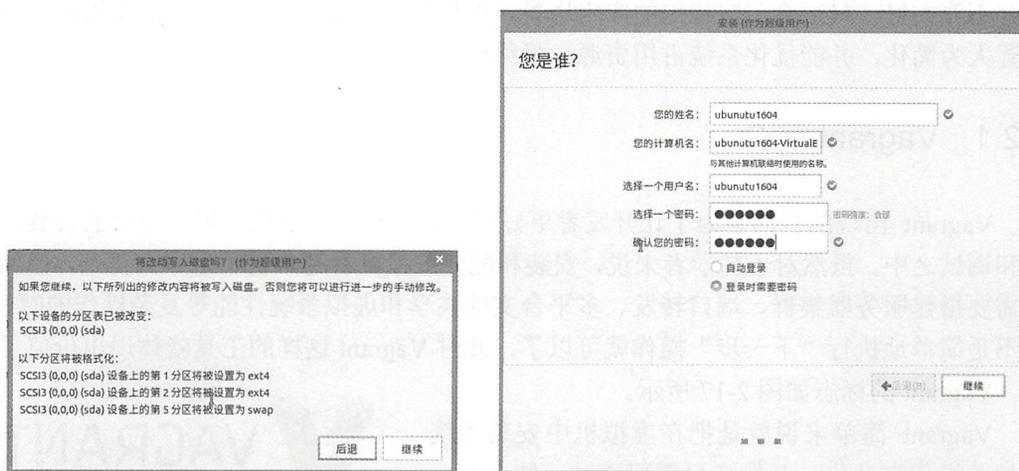


图 2-15 设置虚拟系统的基本信息

### (7) 虚拟系统安装完毕与使用

单击“继续”按钮后，等待系统安装完毕就可以体验和使用 Ubuntu Kylin 系统了，如图 2-16 所示。

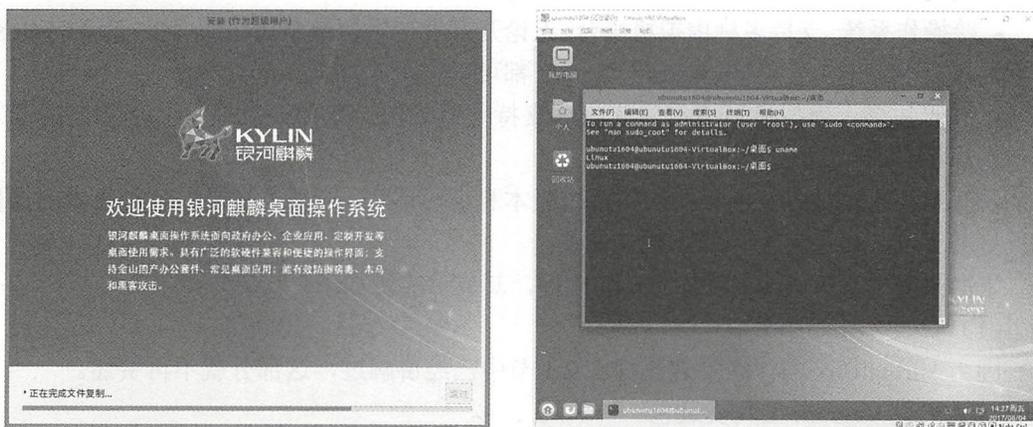


图 2-16 完成虚拟系统的安装

## 2.2 虚拟机辅助工具——Vagrant

使用虚拟机的目的是为了打造一个通用的、接近生产力服务器的开发环境，避免“在

我的机器上可以，在你的机器上为什么不行？”这样的问题出现。但由 2.1 节可知，VirtualBox 安装虚拟系统的过程和配置相对烦琐，对没接触过虚拟机工具的新人，在使用上并不算友好，只适合一些特定的应用场景。而使用 Vagrant 工具可以让虚拟机的安装和配置大为简化，并能优化系统占用资源，最终提高开发效率。

## 2.2.1 Vagrant 简介

Vagrant 出现的目的是为了开发者更好地使用虚拟机，避免其陷入无尽的安装、配置和调试之中。虽然对于初学者来说，安装和配置虚拟机看起来也不是那么复杂，但当遇到需要搭建服务器集群、端口转发、多平台文件共享和虚拟系统性能等复杂操作的时候，就不是简单地执行“下一步”操作就可以了，此时 Vagrant 这样的工具就体现出价值了。

Vagrant 的标志如图 2-17 所示。

Vagrant 简单来说就是把在虚拟机中安装系统等相关操作自动化，开发者只需要通过一些命令，就可以完成跨平台通用开发环境的搭建和使用。其优势如下：



图 2-17 虚拟机辅助工具 Vagrant 的标志

- 软件安装流程简单，操作界面友好。
- 虚拟机操作自动化。解决安装初始化、文件同步、SSH 远程连接和环境依赖等常见问题。
- 跨操作系统，支持多种虚拟机平台。无论开发者使用的是 Windows 下的 Virtual Box，还是 Mac OS 下的 Vmware，Vagrant 都可以完美支持。
- 方便共享。虚拟机环境搭建完成后支持一键导出，方便开发团队共享相同的开发环境。
- 开源社区提供各式各样的操作系统版本和集成环境，无须再去寻找各种版本的操作和工具，可以实现一键下载、安装。

下面来看一下 Vagrant 工具的安装流程，其具体流程如下：

### (1) 安装 VirtualBox

因为 VirtualBox 的详细安装流程在 2.1 节中已经讲解过，这部分就不再赘述。

### (2) 下载 Vagrant 工具安装包

访问 <https://www.vagrantup.com/downloads.html>，根据操作系统版本下载安装文件，在这里选择 Windows64 位版本，单击“64-bit”按钮即可开始下载，如图 2-18 所示。



图 2-18 选择下载 64-bit 版本的 Vagrant 工具安装包

### (3) 开始安装 Vagrant 工具

双击打开下载完成的.exe 安装包，只需要单击 Next 按钮一步一步地安装即可。注意除了安装路径一般不选择系统盘外，其他都使用默认设置即可。

Vagrant 工具安装完成，如图 2-19 所示。安装完毕后需要重启系统，提示如图 2-20 所示。



图 2-19 在 Windows 系统中安装 Vagrant 工具

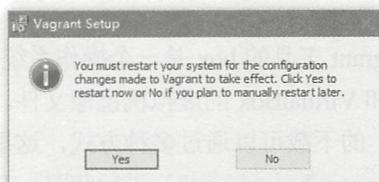


图 2-20 Vagrant 工具安装完成后重启操作系统

### (4) 开始使用

查看 Vagrant 是否安装成功，需要在 Windows 系统“开始”按钮上单击鼠标右键，然后选择“命令提示符(管理员)(A)”选项后，出现命令行窗口，输入以下命令：

```
vagrant -v
```

若正常显示版本信息，则说明 Vagrant 工具已经安装成功，效果如图 2-21 所示。

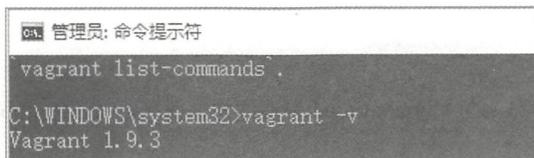


图 2-21 查看 Vagrant 工具的版本信息

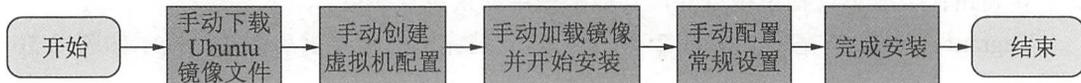
**提示：**这里使用的操作系统版本为 Windows 10，在其他 Windows 系统中打开命令行工具可能稍有不同。

## 2.2.2 Vagrant 常用操作

Vagrant 工具的方便之处就在于，可以直接下载已经打包好的虚拟机初始化文件(box)。省去了自己下载系统镜像、安装配置的过程。相比传统的虚拟机系统安装流程，

使用 Vagrant 工具的相关操作大为简化，如图 2-22 所示。

传统方式安装虚拟系统



使用 Vagrant 工具安装虚拟系统

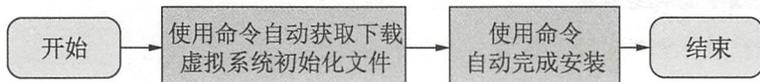


图 2-22 使用传统方式与使用 Vagrant 工具安装虚拟系统的对比流程图

Vagrant 工具的 box 是一个操作系统环境，实际上就是一个压缩包，包含了 Vagrant 的配置信息和 VirtualBox 的虚拟机镜像文件。

box 的下载可以通过多种方式，这里推荐使用官方的 Vagrant Cloud（地址是 <https://app.vagrantup.com/boxes/search>）进行获取，方便检索自己所需的 box 版本。以检索关键字 Ubuntu 为例，部分查询结果如图 2-23 所示，可以看到其覆盖了常见的 Ubuntu 操作系统发行版本。

Provider: any   virtualbox   vmware   libvirt   more		Sort by: Downloads   Recently Created   Recently Updated	
	<b>ubuntu/trusty64</b> 20170803.0.0 Official Ubuntu Server 14.04 LTS (Trusty Tahr) builds	virtualbox	Downloads: 29,868,508 Released: about 5 hours ago
	<b>hashicorp/precise64</b> 1.1.0 A standard Ubuntu 12.04 LTS 64-bit box.	hyperv   virtualbox   vmware_fusion	Downloads: 6,594,002 Released: over 3 years ago
	<b>ubuntu/xenial64</b> 20170803.0.0 Official Ubuntu 16.04 LTS (Xenial Xerus) Daily Build	virtualbox	Downloads: 2,760,604 Released: 4 days ago

图 2-23 在 Vagrant Cloud 中在线检索操作系统压缩包 box

这里选用以基于 Ubuntu 16.04 LTS 为基础的“ubuntu/xenial64”作为 box 实例进行下载、安装和启动等操作方法的演示。

### (1) box 获取与安装

添加 box 的命令格式如下：

```
vagrant box add base 远端的 box 地址或者本地的 box 文件名
```

命令中 `vagrant box add` 是添加 box 的命令，base 是 box 的名称，可以自定义。这里默认使用 base，主要用来标识添加的 box，方便后面的安装。

首先打开 Windows 命令行界面，输入并执行以下命令：

```
vagrant box add ubuntu/xenial64
```

Vagrant 会自动开始下载 box 到本地，因为是官方的 box，所以在这里不定义标识，自

动使用官方的原名即可。下载完成结果如图 2-24 所示。

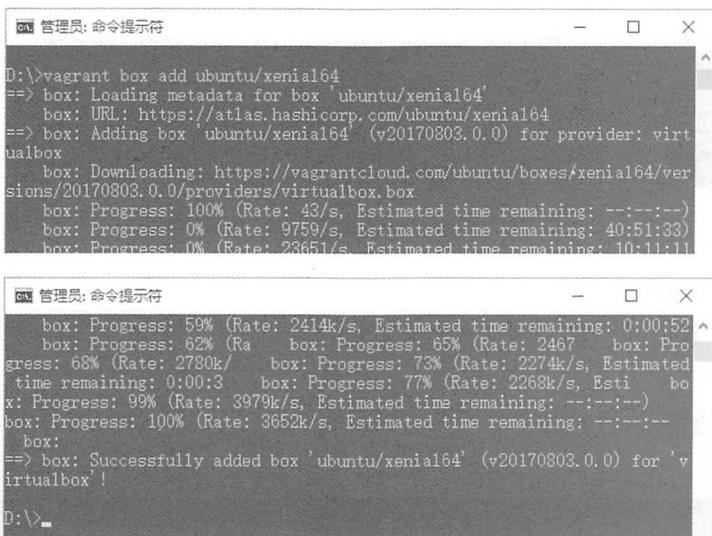


图 2-24 下载官方的“ubuntu/xenial64”镜像 box

当命令行提示以下字样的时候，说明 box 已经下载完成：

```
box: Successfully added box 'box名' (版本号) for 'virtualbox'
```

执行以下命令，查看当前系统中已经添加的 box 列表：

```
vagrant box list
```

此命令用来查看 Vagrant 中已经添加的 box 列表，此时可以看到名为“ubuntu/xenial64”的 box 已经添加，后面就可以直接安装使用了，如图 2-25 所示。

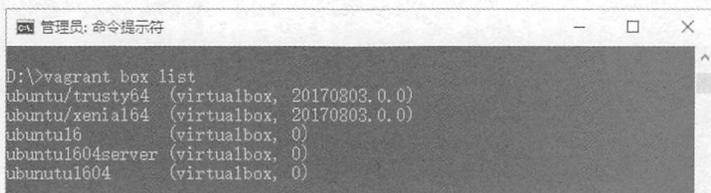


图 2-25 查看 Vagrant 已经添加的 box 列表

## (2) 创建虚拟机系统配置文件

初始化的过程也是使用命令操作，首先定位到指定的目录（如 D 盘下的 ubuntu 目录，开发者可以灵活定义），执行以下命令：

```
vagrant init ubuntu/xenial64
```

Vagrant 会在当前目录下生成 Vagrantfile 配置文件，用文本编辑器打开后，会发现里面有很多配置项，但这不影响使用默认配置来完成安装，后面还会详细说明此配置文件。

执行完毕后的提示如图 2-26 所示。

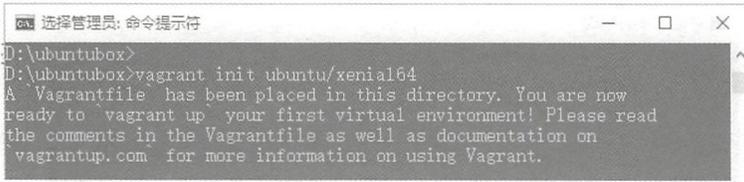


图 2-26 创建 Vagrant 虚拟系统配置文件

### (3) 初始化虚拟机

执行以下命令：

```
vagrant up
```

初始化过程如图 2-27 所示。

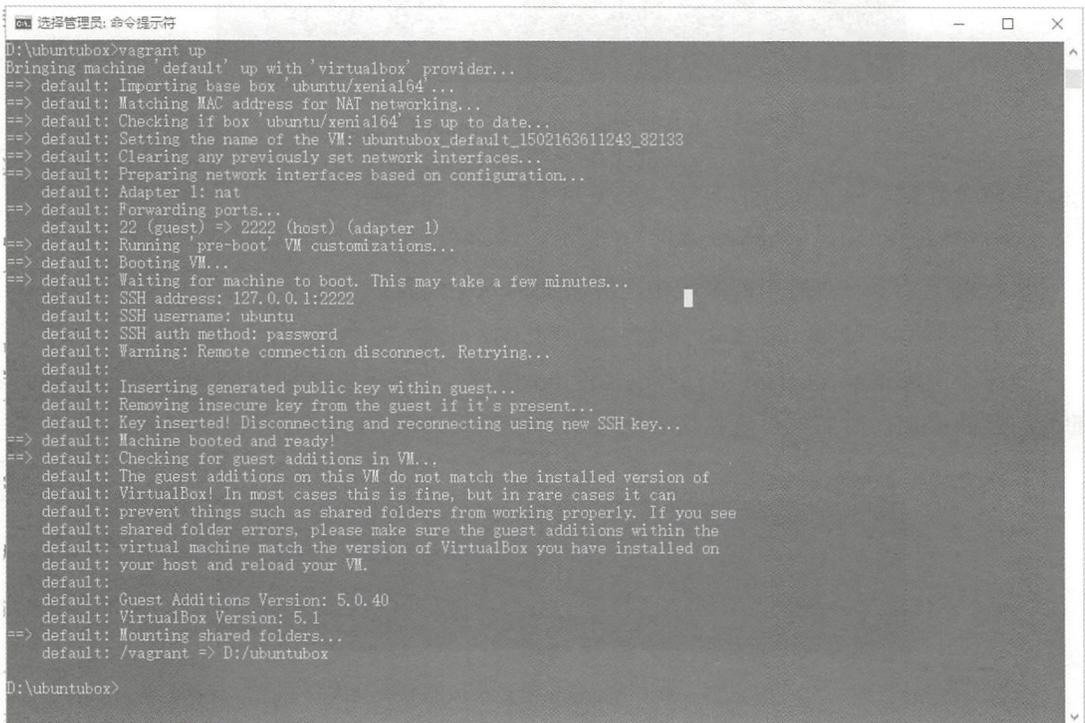


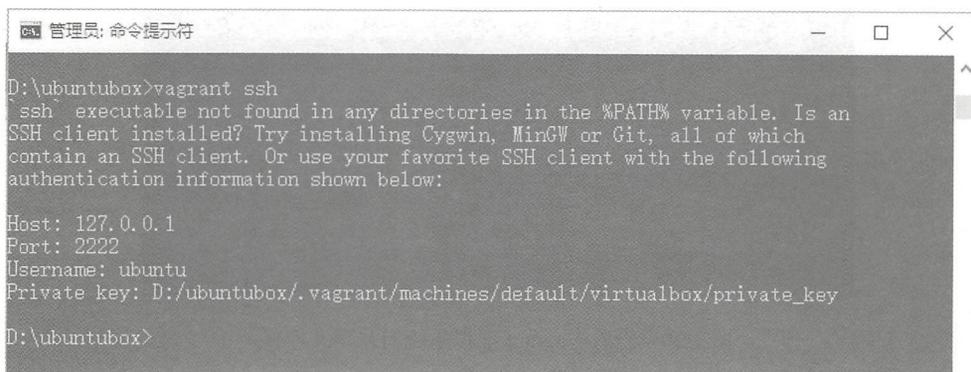
图 2-27 在 Vagrant 中初始化虚拟机

### (4) 连接虚拟机

完成虚拟机的初始化后，就可以执行相应的命令进行 SSH 连接：

```
vagrant ssh
```

命令执行效果如图 2-28 所示。



```
管理员: 命令提示符
D:\ubuntubox>vagrant ssh
ssh executable not found in any directories in the %PATH% variable. Is an
SSH client installed? Try installing Cygwin, MinGW or Git, all of which
contain an SSH client. Or use your favorite SSH client with the following
authentication information shown below:

Host: 127.0.0.1
Port: 2222
Username: ubuntu
Private key: D:/ubuntubox/.vagrant/machines/default/virtualbox/private_key
D:\ubuntubox>
```

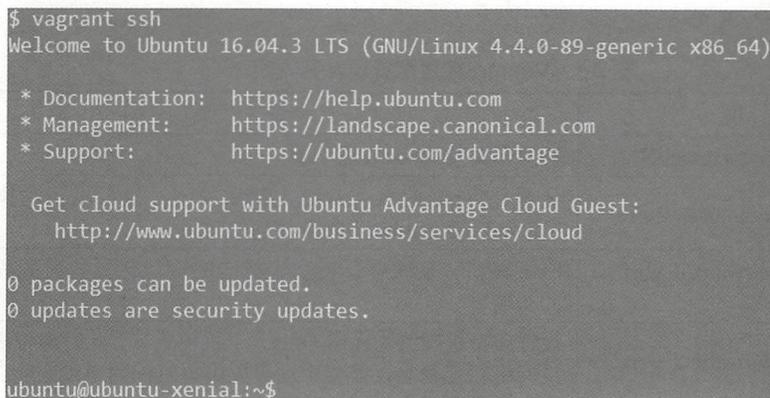
图 2-28 Windows 命令行无法执行 SSH 操作

从执行结果可以看出，Windows 原生的命令行工具不支持 SSH 操作，所以在这里可以使用 XShell、Git Bash 等第三方工具进行操作连接。

更换第三方工具前，需要先在 Windows 命令行工具下执行关闭虚拟机操作：

```
vagrant halt
```

随后打开 Git Bash 命令行工具（安装完 Windows Git 工具后，任意目录下，在系统右键菜单中可以找到），再次执行 `vagrant up` 命令启动虚拟机。随后使用 `vagrant ssh` 命令连接虚拟机，连接成功后就可以操作虚拟机系统了，如图 2-29 所示。



```
$ vagrant ssh
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

ubuntu@ubuntu-xenial:~$
```

图 2-29 使用 SSH 成功连接 Vagrant 虚拟机系统

**提示：** Vagrant 工具下的虚拟系统一般使用命令行操作，不使用图形化界面。

### （5）查看虚拟机基本信息

执行 Linux 下的 `df -f` 命令可以查看磁盘挂载信息，其中 `vagrant` 目录映射真实系统中 `Vagrantfile` 配置文件所在的目录，可以实现虚拟系统与真实系统的文件共享，便于程序在虚拟机中调试，如图 2-30 所示。

```
ubuntu@ubuntu-xenial:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            490M   0  490M   0% /dev
tmpfs           100M  3.1M   97M   4% /run
/dev/sda1       9.7G  955M   8.7G  10% /
tmpfs           497M   0   497M   0% /dev/shm
tmpfs           5.0M   0   5.0M   0% /run/lock
tmpfs           497M   0   497M   0% /sys/fs/cgroup
vagrant         60G   43G   18G   71% /vagrant
tmpfs           100M   0   100M   0% /run/user/1000
```

图 2-30 查看 Vagrant 虚拟系统的共享文件夹

## 2.2.3 Vagrant 常用配置与命令

Vagrant 没有图形化的操作，使用命令行和配置文件来配置虚拟机的网络、文件和其他设置。这种形式简化了配置流程，也减少了配置错误的发生。

### 1. Vagrant 常见配置

直接使用虚拟机软件，参数配置非常烦琐，而 Vagrant 只需要通过修改 Vagrantfile 文件，就可以快捷地配置虚拟机的网络设定、共享文件位置等。Vagrant 常用的配置项及其说明如表 2-1 所示。

表 2-1 Vagrant 常用配置项及其说明

配置项	说明
<code>config.vm.box = "ubuntu/xenial64"</code>	初始化安装的 box 名称
<code>config.vm.box_check_update = false</code>	是否开启 box 的自动升级，默认关闭。也可以手动执行命令 `vagrant box outdated` 实现相同的效果
<code>config.vm.network "forwarded_port", guest: 80, host: 8080</code>	网络端口转发映射，将宿主机的 80 端口转发映射到虚拟机的 8080 端口
<code>config.vm.network "forwarded_port", guest: 80, host: 8080</code>	设置虚拟机网络为 Host-Only（主机模式）。此时虚拟机和宿主网络相互独立，需要手动设置虚拟机的网络 IP 地址。注意设置时不要和宿主网段冲突
<code>config.vm.network "public_network"</code>	设置虚拟机网络为 Bridge（桥接模式）。此时虚拟机模拟成一台物理机，与宿主主机访问同一个路由，共享同一个网络
<code>config.vm.synced_folder "../data", "/vagrant_data"</code>	同步目录设置。第一个参数设置宿主机目录，第二个参数对应虚拟机系统的目录

不过很多时候，开发者会使用虚拟机搭建集群测试应用，所以 Vagrant 也提供了批量配置虚拟机的方法。这里以三台服务器的简单集群架构为例，快速配置三台虚拟机系统，如图 2-31 所示。

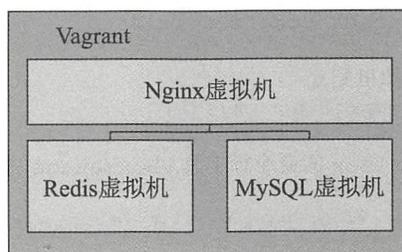


图 2-31 简单的服务器集群架构

具体的操作步骤如下：

(1) 手动创建配置文件

在指定目录（实例所在目录为 D:\ubuntumuti）下手动创建 Vagrantfile 文件，添加配置代码如下：

# 三个虚拟机的配置

```
Vagrant.configure("2") do |config|
```

```
  # 虚拟机一: Nginx 虚拟机配置
```

```
  config.vm.define :nginx do |nginx|
```

```
    nginx.vm.provider "virtualbox" do |v|
```

```
      # 使用 VirtualBox 的命令行工具 VBoxManage, 执行 modifyvm 命令设置主机名和内存大小
```

```
      v.customize ["modifyvm", :id, "--name", "nginx", "--memory", "1024"]
```

```
    end
```

```
    # 设置 box 名
```

```
    nginx.vm.box = "ubuntu/xenial64"
```

```
    # 设置主机名
```

```
    nginx.vm.hostname = "nginx"
```

```
    # 设置网络及 IP 地址
```

```
    nginx.vm.network :private_network, ip: "192.168.33.21"
```

```
    # 关闭自动更新
```

```
    config.vm.box_check_update = false
```

```
  end
```

```
  # 虚拟机二: Redis 虚拟机配置
```

```
  config.vm.define :redis do |redis|
```

```
    redis.vm.provider "virtualbox" do |v|
```

和内存大小

```
      # 使用 VirtualBox 的命令行工具 VBoxManage, 执行 modifyvm 命令设置主机名
```

```
      v.customize ["modifyvm", :id, "--name", "redis", "--memory", "1024"]
```

```
    end
```

```
    # 设置 box 名
```

```
    redis.vm.box = "ubuntu/xenial64"
```

```
    # 设置主机名
```

```
    redis.vm.hostname = "redis"
```

```
    # 设置网络及 IP 地址
```

```
    redis.vm.network :private_network, ip: "192.168.33.22"
```

```
    # 关闭自动更新
```

```

        config.vm.box__check_update = false
    end
    # 虚拟机三: MySQL 虚拟机配置
    config.vm.define :mysql do |mysql|
        mysql.vm.provider "virtualbox" do |v|
            # 使用 VirtualBox 的命令工具 VBoxManage, 执行 modifyvm 命令设置主机名
            # 和内存大小
            v.customize ["modifyvm", :id, "--name", "mysql", "--memory",
                "1024"]
        end
        # 设置 box 名
        mysql.vm.box = "ubuntu/xenial64"
        # 设置主机名
        mysql.vm.hostname = "mysql"
        # 设置网络及 IP 地址
        mysql.vm.network :private_network, ip: "192.168.33.23"
        # 关闭自动更新
        config.vm.box__check_update = false
    end
end
end

```

## (2) 启动批量安装

保存 Vagrantfile 配置文件后, 在当前目录下, 打开 Git Bash 工具并执行 `vagrant up` 命令, 预先配置好的三台虚拟机会自动开始安装, 效果如图 2-32 所示。

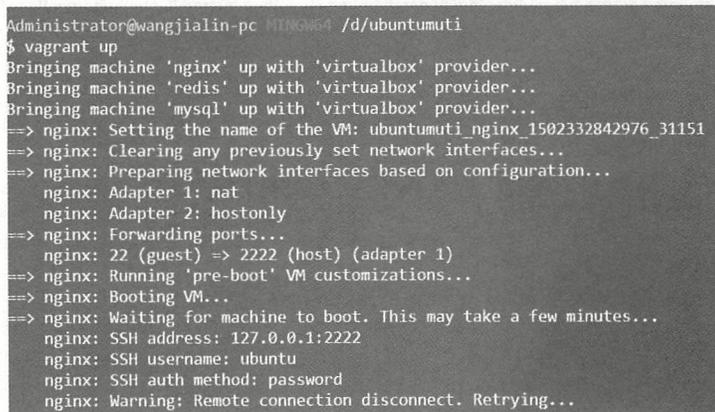


图 2-32 批量配置与安装 Vagrant 虚拟机

 **提示:** Vagrantfile 也可以通过编写循环语句的方式配置多台虚拟机, 这样更加高效。

## (3) 连接使用虚拟机系统

安装完成后测试任意一台虚拟机是否安装成功, 进入 Vagrantfile 配置文件所在的目录, SSH 连接 hostname 名为 redis 的虚拟机, 操作命令如下:

```
vagrant ssh redis
```

连接成功的结果如图 2-33 所示。

```

Administrator@wangjialin-pc-MIN664 /d/ubuntuuti
$ vagrant ssh redis
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

ubuntu@redis:~$

```

图 2-33 成功连接代号为 redis 的虚拟机系统

## 2. Vagrant常用命令

虽然前面已经使用过很多 Vagrant 命令，这里还是把最常用的命令总结成表，如表 2-2 所示。

表 2-2 Vagrant工具常用命令说明

命 令	说 明
vagrant init	初始化
vagrant up	启动虚拟机
vagrant halt	关闭虚拟机
vagrant reload	重启虚拟机
vagrant ssh	SSH 至虚拟机
vagrant status	查看虚拟机运行状态
vagrant destroy	销毁当前虚拟机
vagrant suspend	挂起当前虚拟机
vagrant resume	恢复被挂起的vm
vagrant box list	列出所有box列表
vagrant box remove {base name}	删除box
vagrant destroy	停止当前正在运行的虚拟机并销毁所有创建的资源
vagrant package	把当前运行的虚拟机环境进行打包，可用于分发开发环境
vagrant plugin	安装/卸载插件
vagrant provision	设置基本的环境，进一步设置可以使用Chef/Puppet进行搭建
vagrant ssh-config	输出SSH连接的一些信息
vagrant status	获取虚拟机状态
vagrant version	获取Vagrant的版本

这里需要说明一下 `vagrant plugin` 指令，因为 VirtualBox 设置共享目录时需要在虚拟机中安装 VirtualBox Guest Additions 模块，虽然 Vagrant 会自动安装，但是因为 VirtualBox Guest Additions 是 VirtualBox 的内核模块，当 VirtualBox 的内核升级之后，VirtualBox Guest Additions 会失效，导致共享目录挂载失败。

而安装 Vagrant 插件 `vagrant-vbguest` 可以解决这个问题，因为该插件会在虚拟机内核升级之后重新安装 VirtualBox Guest Additions。需要执行如下命令：

```
vagrant plugin install vagrant-vbguest
```

执行命令安装完成后，效果如图 2-34 所示。

A terminal window showing the command `vagrant plugin install vagrant-vbguest` being executed. The output shows the installation progress: `Installing the 'vagrant-vbguest' plugin. This can take a few minutes...` followed by `Installed the plugin 'vagrant-vbguest (0.14.2)'!`

图 2-34 通过安装插件解决共享文件夹失败的问题

## 2.3 打造个性化开发环境

无论使用 VirtualBox 还是 Vagrant 工具搭建虚拟机，都只是提供了一个基础的操作系统环境。此环境虽然已经接近或者达到应用环境的要求，但还需要继续安装相关的软件才可以运行应用程序。

在 2.2 节中已经介绍了如何安装集成环境，这种方法可以满足一般的基础开发，但对一些定制化要求比较高的应用来说，灵活性就大打折扣。例如，想继续安装一些 PHP 的扩展，这时候每个集成环境操作的方式可能都不同，不熟悉的开发者往往会在这里耗费大量的时间和精力。

基于上述需求，本节将讲解如何借助 Ubuntu 系统自带的 `apt` 包管理工具，一步一步地打造个性化的开发环境。

### 2.3.1 准备工作

在本地搭建了以 Ubuntu 16.04 为基础的虚拟系统后，一般会更新替换 `apt` 包管理工具的源为国内的镜像，这样在安装软件包的时候网络下载速度会更快。国内很多互联网巨头或者知名大学都提供了这样的服务，这里以阿里云为例，讲解如何快速替换 `apt` 包管理工具的源。

首先要养成备份配置文件的好习惯，这样才能在文件被错误修改或丢失后快速找回。操作步骤如下。

### (1) 备份配置文件

先执行以下命令：

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.old
```

### (2) 新增国内源地址

随后编辑 `sources.list` 文件：

```
sudo vim /etc/apt/sources.list
```

在 Vim 编辑器中输入 “:%d” 可以清楚所有的内容，随后输入以下内容：

```
#deb cdrom:[Ubuntu 16.04 LTS _Xenial Xerus_ - Release amd64 (20160420.1)]/
xenial main restricted
deb-src http://archive.ubuntu.com/ubuntu xenial main restricted #Added by
software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial main restricted
multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted
multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
deb http://mirrors.aliyun.com/ubuntu/ xenial multiverse
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates multiverse
deb http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted
universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-backports main
restricted universe multiverse #Added by software-properties
deb http://archive.canonical.com/ubuntu xenial partner
deb-src http://archive.canonical.com/ubuntu xenial partner
deb http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted
multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-security multiverse
```

保存完成后，就可以执行 apt 包管理工具的更新操作：

```
sudo apt-get update
```

提示：apt 包管理工具的更新操作所需的时间，会因为当前的网络 and 系统版本等存在差异。

## 2.3.2 安装及配置 Nginx

最近几年，Nginx 服务器以免费开源和灵活高效的特点，正在迅速抢占服务器市场。根据国外 W3Techs.com 网站截止 2017 年 7 月的数据统计，Nginx 的市场份额稳居第二，排名仅次于老大哥 Apache，而且其份额也处于不断增长中。

2017 年 7 月服务器软件市场占有率和新增站点数，如图 2-35 所示。

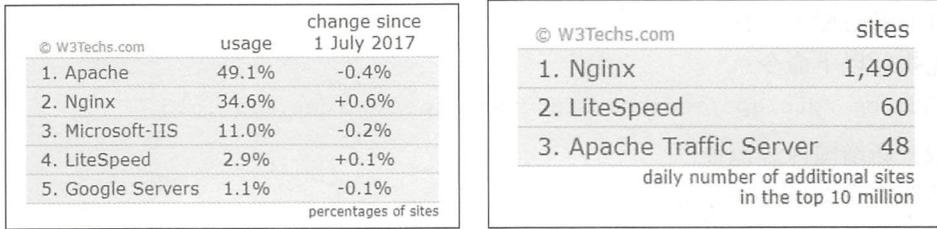


图 2-35 服务器软件市场占有率和新增站点数统计

使用 Nginx 构建 Web 服务，相比 Apache 有以下几点优势：

- 轻量级。同样是 Web 服务，Nginx 比 Apache 占用系统资源更低。
- 并发性能高。Nginx 请求处理是异步非阻塞型，而 Apache 是阻塞型的，在高并发下 Nginx 能保持低资源、低消耗、高性能。
- 高度模块化的设计，灵活的扩展性。

除了以上几点外，Nginx 和 Apache 处理 PHP 脚本的方式也不尽相同。Apache 是通过自身的模块 `mod_php` 来解析 PHP，而 Nginx 则是通过 PHP-FPM(FastCGI)来解析 PHP。`mod_php` 通过嵌入 PHP 解释器到 Apache 进程中，只支持 Apache，而 PHP-FPM 以独立进程的形式出现，只要对应的 Web 服务器实现 CGI 或者 FastCGI 协议，就能够处理 PHP 请求。

总的来说，Nginx 只有在需要的时候才会去找 PHP（PHP-FPM），而 Apache 无论何时都需要 PHP 随时待命（常驻内存），可见 Nginx 效率要高一些。两种服务器对 PHP 请求处理与解析的不同方式如图 2-36 所示。

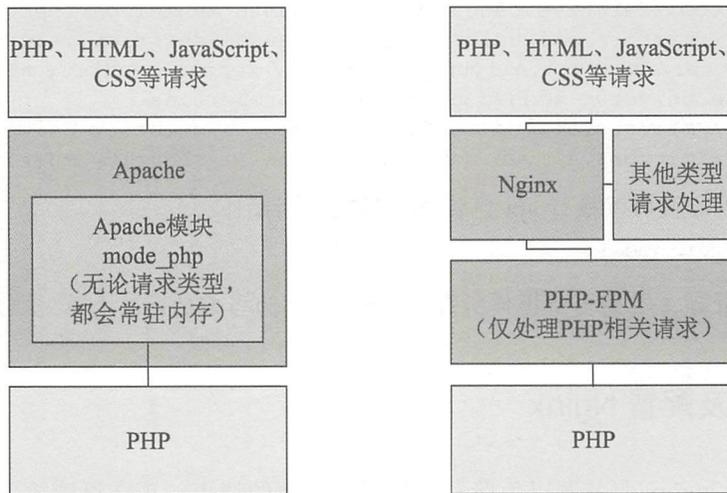


图 2-36 Apache 与 Nginx 解析 PHP 的不同方式

在简单了解了 Nginx 服务器的解析原理后，下面继续看如何在 Ubuntu 上安装 Nginx，

具体步骤如下。

#### (1) 执行命令安装 Nginx

这里还是使用 apt 包的相关命令即可：

```
sudo apt-get install -y nginx
```

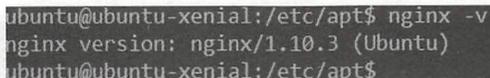
其中，参数“-y”代表直接执行命令，无须再确认。执行后，apt 包管理工具会自动编译安装 Nginx 软件和其相关的依赖，这种方式虽然灵活性不如手动编译安装，但是相比集成环境包，效率已经提高了很多，同时也不会让安装过程过于烦琐，后续的升级、卸载都十分方便。

#### (2) 查看 Nginx 版本号

安装完毕后，执行以下命令查看 Nginx 的版本号：

```
nginx -v
```

效果如图 2-37 所示。



```
ubuntu@ubuntu-xenial:/etc/apt$ nginx -v
nginx version: nginx/1.10.3 (Ubuntu)
ubuntu@ubuntu-xenial:/etc/apt$
```

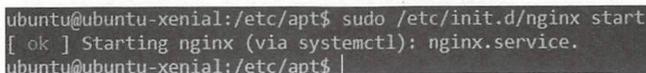
图 2-37 查看 Nginx 的版本号

#### (3) 启动 Nginx

随后执行以下命令启动 Nginx：

```
sudo /etc/init.d/nginx start
```

可以看到系统提示 Nginx 已经启动成功，如图 2-38 所示。



```
ubuntu@ubuntu-xenial:/etc/apt$ sudo /etc/init.d/nginx start
[ ok ] Starting nginx (via systemctl): nginx.service.
ubuntu@ubuntu-xenial:/etc/apt$
```

图 2-38 启动 Nginx 系统服务

#### (4) 访问 Nginx 页面

随后在浏览器中访问 <http://192.168.33.20> (虚拟机系统配置的 IP 地址)，界面如图 2-39 所示，说明 Nginx 已经正常工作。



图 2-39 在浏览器中查看 Nginx 服务状态

提示：Apache 与 Nginx 在不同的应用领域各有优劣，需要根据实际需求选择。

### 2.3.3 安装及配置 PHP

PHP 7 版本的发布不仅带来了性能上的巨大提升，也提供了更高效的开发方式。不过 apt 包管理工具中可以安装的 PHP 版本不一定是最新的，所以需要使用以下命令查看当前可以安装的 PHP 版本：

```
sudo apt-cache search php
```

执行结果如图 2-40 所示，可以看到在当前系统下，可以安装的最高版本为 PHP 7.0.x。虽然目前 PHP 的开发版已经到了 7.2.x，但是最新的不一定是最好的，开发版本除了稳定性、兼容性不如稳定版本，支持的第三方扩展往往也达不到生产力环境的需求。所以这里选择目前最稳定的 PHP 7.0 版本进行安装。

```
php7.0 - server-side, HTML-embedded scripting language (metapackage)
php7.0-cgi - server-side, HTML-embedded scripting language (CGI binary)
php7.0-cli - command-line interpreter for the PHP scripting language
php7.0-common - documentation, examples and common module for PHP
php7.0-curl - CURL module for PHP
php7.0-dev - Files for PHP7.0 module development
php7.0-gd - GD module for PHP
php7.0-gmp - GMP module for PHP
```

图 2-40 使用 apt 工具查找可以安装的 PHP 软件包

因为 Nginx 需要依赖 PHP-FPM 才能顺利地解析 PHP 脚本程序，所以除了安装 PHP，还需要安装 PHP-FPM。安装流程如下。

#### (1) 安装 PHP

执行命令如下：

```
sudo apt-get install -y php7.0 php7.0-fpm
```

安装完成后，会发现 apt 自动创建了一些配置文件，如图 2-41 所示。

```
Creating config file /etc/php/7.0/cli/php.ini with new version
Setting up php7.0-fpm (7.0.22-0ubuntu0.16.04.1) ...

Creating config file /etc/php/7.0/fpm/php.ini with new version
Setting up php7.0 (7.0.22-0ubuntu0.16.04.1) ...
Processing triggers for systemd (229-4ubuntu19) ...
Processing triggers for ureadahead (0.100.0-19) ...
```

图 2-41 安装 PHP 相关包生成的配置文件

#### (2) 查看 PHP 版本号

查看 PHP 是否安装成功，可以使用以下命令查看 PHP 版本号：

```
php -v
```

执行后可以看到当前的版本号如图 2-42 所示。

```
ubuntu@ubuntu-xenial:~$ php -v
PHP 7.0.22-0ubuntu0.16.04.1 (cli) ( NTS )
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies
    with Zend OPcache v7.0.22-0ubuntu0.16.04.1, Copyright (c) 1999-2017, by Zend Technologies
```

图 2-42 在系统中查看 PHP 的版本号

### (3) 配置 PHP-FPM

随后需要配置 PHP-FPM，才能让 Nginx 可以与之搭配工作。首先修改 Nginx 默认的配置，以增加对 PHP 的支持，找到并编辑/etc/nginx/sites-available/default 文件，修改以下配置代码，以实现 Nginx 对 index.php 文件的支持：

```
# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html index.php;
```

随后在同一个文件中修改以下代码，以实现 Nginx 对 PHP-FPM 的支持：

```
location ~ /\.php$ {
    include snippets/fastcgi-php.conf;
    # With php7.0-cgi alone:
    # fastcgi_pass 127.0.0.1:9000;
    # With php7.0-fpm:
    fastcgi_pass unix:/run/php/php7.0-fpm.sock;
}
```

这里需要注意，配置 PHP-FPM 的版本号一定要与已安装的版本保持一致，Nginx 默认的配置项里面的 PHP 版本一般会比较新，如默认为 PHP 7.1。

### (4) 检测 Nginx 配置文件

保存文件后，使用以下命令检测配置是否正确：

```
sudo nginx -t
```

通过检测的效果，如图 2-43 所示。

```
ubuntu@ubuntu-xenial:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
ubuntu@ubuntu-xenial:~$
```

图 2-43 使用 Nginx 自带命令检测配置是否正确

配置完成 Nginx 后，还需要确保/etc/php/7.0/fpm/pool.d/www.conf 文件中，以下配置项没有被注释，如图 2-44 所示。

### (5) 重启 Nginx 与 PHP-FPM

重载 Nginx 配置文件并重启 PHP-FPM，让配置生效。命令分别如下：

```
sudo nginx -s reload
sudo /etc/init.d/php7.0-fpm restart
```

```
34 ; '/path/to/unix/socket' - to listen on
35 ; Note: This value is mandatory.
36 listen = /run/php/php7.0-fpm.sock
37
38 ; Set listen(2) backlog.
```

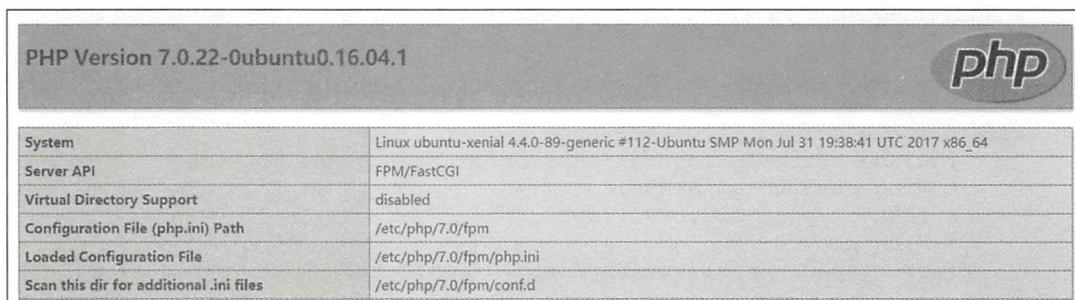
图 2-44 修改 PHP-FPM 文件实现对 Nginx 的支持

#### (6) 查看 PHP 探针信息

在目录/var/www/html下新增phpinfo.php脚本文件，增加内容如下：

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

保存文件后，在浏览器中访问<http://192.168.33.20/phpinfo.php>，可以看到 PHP 相关信息，说明 Nginx 已经可以成功解析 PHP 脚本文件，如图 2-45 所示。



PHP Version 7.0.22-0ubuntu0.16.04.1	
System	Linux ubuntu-xenial 4.4.0-89-generic #112-Ubuntu SMP Mon Jul 31 19:38:41 UTC 2017 x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/fpm
Loaded Configuration File	/etc/php/7.0/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/fpm/conf.d

图 2-45 在网页中显示 PHP 的系统信息

提示: apt 包管理工具自动安装的软件版本会不定期更新, 本书中安装的 PHP 版本为 7.0.22。

## 2.3.4 安装及配置 MySQL

MySQL 由于其性能高、成本低、可靠性好，已经成为最流行的开源数据库，因此被广泛地应用在互联网上的中小型网站中，而 PHP 也同样适合这些网站，所以常见的 LAMP 组合或者 LNMP 组合中的 M，就是指 MySQL。

但是 MySQL 自从被甲骨文公司收购后，商业版本的授权费用也大为上涨，所以在使用 MySQL 的时候可以考虑其新的开源社区分支 MariaDB。MariaDB 的使用方式与 MySQL 差别不大，在这里就不深入讨论了，有兴趣的读者可以在网上查阅相关资料。

这里还是使用 apt 包管理工具快速安装 MySQL，安装步骤如下。

#### (1) 执行以下命令安装 MySQL

```
sudo apt-get install -y mysql-server mysql-client php7.0-mysql
```

命令中不仅安装了 MySQL 的服务端、客户端，还安装了 PHP 7.0 对于 MySQL 的支持依赖。

安装过程中，会要求输入 MySQLroot 账号的密码，密码默认使用“root”，操作如图 2-46 所示。

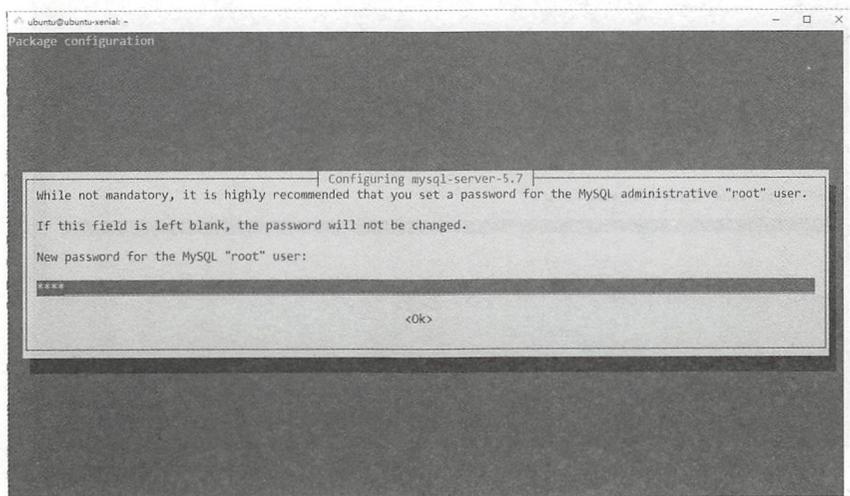


图 2-46 设置 MySQL 管理员账号密码

## (2) 使用 MySQL 默认客户端

安装完毕后，可以使用以下命令实现 MySQL 的命令行管理操作，出现如图 2-47 所示界面说明 MySQL 已经安装成功。

```
ubuntu@ubuntu-xenial:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

图 2-47 在命令行中管理 MySQL 数据库

## (3) 安装 phpMyAdmin

当数据操作频繁的时候，使用命令行并不是最好的办法，这里推荐安装基于 PHP 编写的 Web 数据库管理工具——phpMyAdmin，使用 apt 可以直接下载安装，执行命令如下：

```
sudo apt-get install -y phpmyadmin
```

在安装过程中，一直选择默认选项即可。其中提示输入 MySQL 密码的时候，需要你自行设置的数据库密码，这里设置为 root，如图 2-48 所示。

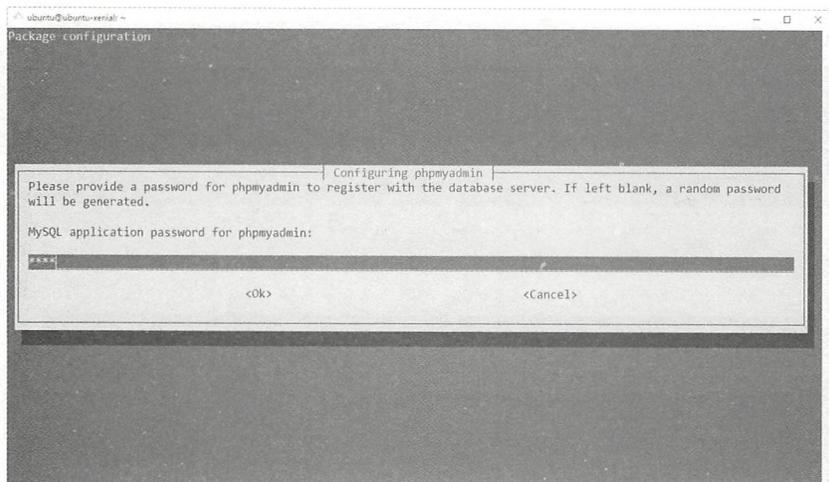


图 2-48 安装的 phpMyAdmin 过程中的数据库密码设置

等待安装完成后，执行以下命令为 phpMyAdmin 创建软链接，为后面的虚拟主机配置打下程序基础：

```
sudo ln -s /usr/share/phpmyadmin mysql
```

**提示：**phpMyAdmin 因为安全性原因，某些版本默认不支持 MySQL 空密码连接，所以建议读者不要给数据库设置空密码。另外，设置的密码也不要过于简单，以提高服务器的安全性。

### 2.3.5 配置虚拟站点

本节以可以正常访问 phpMyAdmin 为例，配置对应的虚拟机主机，以便方便管理 MySQL，操作步骤如下。

#### (1) 新增 Nginx 虚拟主机配置文件

首先在/etc/nginx/目录下新增 vhost 目录，此目录专门用来存储虚拟主机的配置文件。

随后编辑 Nginx 默认的配置文件/etc/nginx/sites-enabled/default，追加以下配置代码，实现自动引入 vhosts 下的子配置文件：

```
include /etc/nginx/vhosts/*;
```

随后进入/etc/nginx/vhosts 目录下，新增 www.mysql.conf 文件，增加并保存内容如下：

```
server {
```

```

listen 8080;                                # 监听 8080 端口
root /var/www/html/mysql;                   # 虚拟主机根目录定位到 phpMyAdmin 的链接
index index.html index.htm index.nginx-debian.html index.php;
server_name www.mysql.com;                  # 虚拟域名
location / {                                # URL 隐藏 index.php 文件
    if ( !-e $request_filename) {
        rewrite ^(.*)$ /index.php/$1 last;
        break;
    }
}
location ~ \.php {                           # PHP 解析规则
    fastcgi_split_path_info ^(.+\.(php))(/.+)$;
    fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $request_uri;
}
}

```

上述的配置文件较为复杂，但是此时只需要知道虚拟站点的端口是 8080，站点的根目录定位到 phpMyAdmin 源代码目录即可。

## (2) 重启 Nginx 和 PHP-FPM 后查看站点效果

完成编辑后，保存配置文件，重载 Nginx 配置，重启 PHP-FPM，在浏览器中访问 <http://192.168.33.20:8080>，即可使用虚拟站点 phpMyAdmin，如图 2-49 所示。



图 2-49 在虚拟机中访问 phpMyAdmin

## 2.3.6 其他常用设置

### 1. 修改日期默认时区

除了常见工具的安装，还需要设置一下系统的时区。首先执行以下命令：

```
sudo dpkg-reconfigure tzdata
```

出现图形界面后,先选择 Asia,再选择 Shanghai,即可设置当前时间的时区为东八区,如图 2-50 所示。

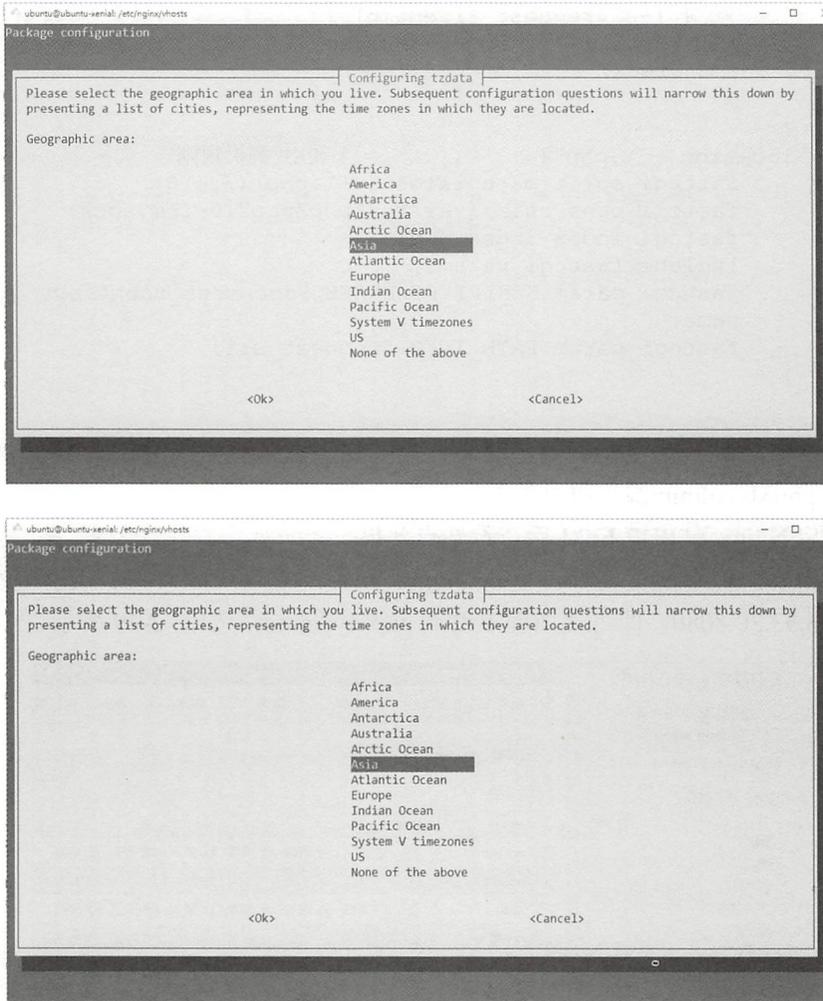


图 2-50 设置系统时间的时区

## 2. 远程连接MySQL数据库

除了使用 phpMyAdmin 管理数据库外,还可以使用宿主机的数据库管理工具进行远程管理。这里以 Navicat 数据库管理工具为例,讲解如何远程连接 MySQL 数据库。

编辑 MySQL 的配置文件,执行以下命令:

```
vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

其中：

```
bind-address      =127.0.0.1
```

修改为：

```
bind-address      = 0.0.0.0
```

保存后，执行以下命令重启 MySQL：

```
sudo /etc/init.d/mysql restart
```

随后进入 MySQL 命令行管理界面，输入以下命令，给账号分配远程操作的权限：

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
```

其中“123456”为账号密码，需要修改成自己的密码。执行以下命令，让设置生效：

```
FLUSH PRIVILEGES;
```

最后开启防火墙对 3306 端口的支持：

```
sudo iptables -I INPUT 4 -p tcp -m state --state NEW -m tcp --dport 3306 -j ACCEPT
```

在宿主主机上，使用 Navicat 客户端就可以远程管理 Vagrant 虚拟机中的 MySQL 了。成功连接并操作数据库，如图 2-51 所示。

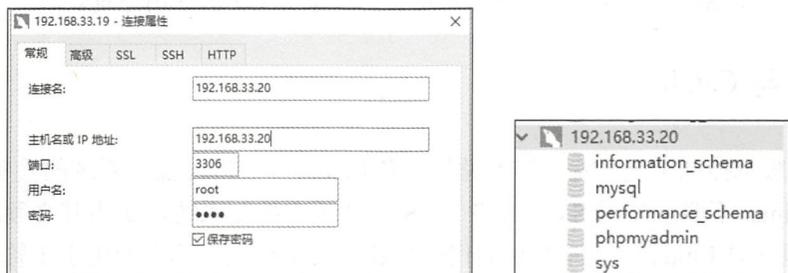


图 2-51 远程连接并管理 MySQL 数据库

## 第 3 章 更先进的版本管理工具——Git

现在越来越多的企业和团队选择使用 Git 作为代码的版本管理工具。Git 因为本身分布式的架构设计和更为高效的分支模型，解决了很多传统版本管理工具使用中的缺陷。

本章将从 Git 的起源说起，随后讲解 Git 的安装与命令操作，最后借助 GitHub 免费搭建一个个人博客站点。

### 3.1 SVN 与 Git

说起 Git，就不得不提老牌的版本管理工具——SVN。本节主要讲解 SVN 与 Git 的差异，帮助只使用过类似 SVN 版本管理工具的开发者的初步掌握 Git 的使用原理。

#### 3.1.1 Git 与 GitHub

提到 Git 版本控制系统，就不得不说其作者 Linus——Linux 操作系统的发明者。在 Git 出现之前，Linus 不愿意使用 SVN 或者 CVS 这些版本控制系统，认为其必须联网的设定十分不便。但同时 Linux 代码库本身的代码更新、补丁迭代等都在使用手工管理，这在很大程度上影响了开发效率，所以 Linus 选择了商业的版本控制系统 BitKeeper 作为 Linux 的版本管理系统。因为 Linux 系统的巨大影响力，BitKeeper 的母公司 BitMover 也授权 Linux 社区免费使用这个系统。

Linux 的吉祥物，如图 3-1 所示。

而发明 Git，也是因为这个商业版本控制系统。当时有开源社区的开发者试图破解 BitKeeper 的协议，随后被 BitMover 公司发现，于是收回 Linux 社区的免费授权。之后 Linus 决定使用自行开发的版本控制系统，只花了两周的时间就完成了第一个版本，这就是 Git。后来 Git 便迅速流行起来。



图 3-1 Linux 的吉祥物（一只企鹅）

如今很多知名的项目和企业都在使用 Git，如图 3-2 所示。

而 GitHub 是一个通过 Git 进行版本控制的软件源代码托管服务平台，其实质是一个网站。很多开发者会把 Git 和 GitHub 混淆，如果明白其各自的作用就很容易区分了。GitHub

自从 2008 年上线后，吸引了大量的开源项目入驻，其中就有我们熟悉的 PHP、jQuery 等知名项目，也有很多优秀的开发者在上面托管了优秀的开源项目，方便了开发者之间的学习与交流。



图 3-2 使用 Git 的知名项目与企业

GitHub 上的 AWESOME 项目统计了很多开源项目，方便开发者检索使用。图 3-3 展示了在 GitHub 上的部分项目，当然在 GitHub 上托管的项目还远不止这些，大家可以看看有没有自己熟悉的技术和项目。



图 3-3 很多项目（包括 PHP）都已经迁移到了 GitHub 上

### 3.1.2 Git 与 SVN 的异同

在 Git 被广泛使用之前，SVN 取代了更为老旧的 CVS 版本控制系统，是最流行的版本控制系统之一。而 SVN 最大的特性就是必须联网才可以使用，版本库在唯一的一台 SVN

服务器上。在这种情况下，网络不稳定等潜在不稳定因素，都将会影响开发者的代码使用体验。此外，因为版本库记录和原始代码都在一台服务器上，若服务器出现异常，可能会导致版本库损坏、丢失代码或丢失提交记录等问题。

典型的SVN版本库使用结构如图3-4所示。

相比SVN，Git最大的特点就是分布式，每个用户在本地都拥有一个完整的版本库和提交记录，而且不需要联网就可以提交代码。Git的典型使用结构，如图3-5所示。

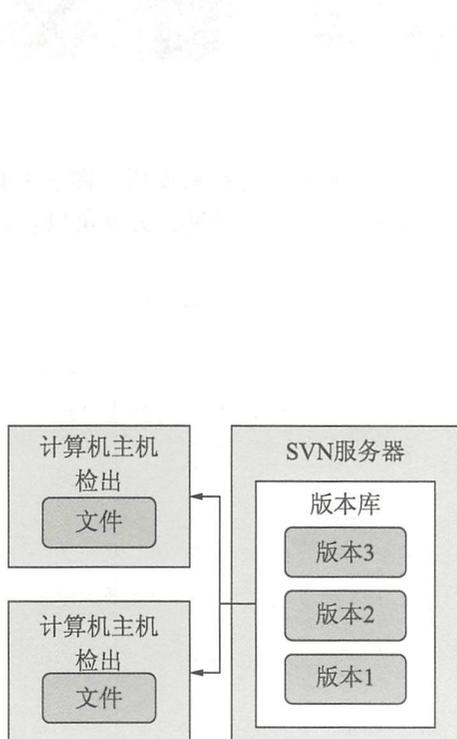


图 3-4 集中式版本控制系统结构示意图

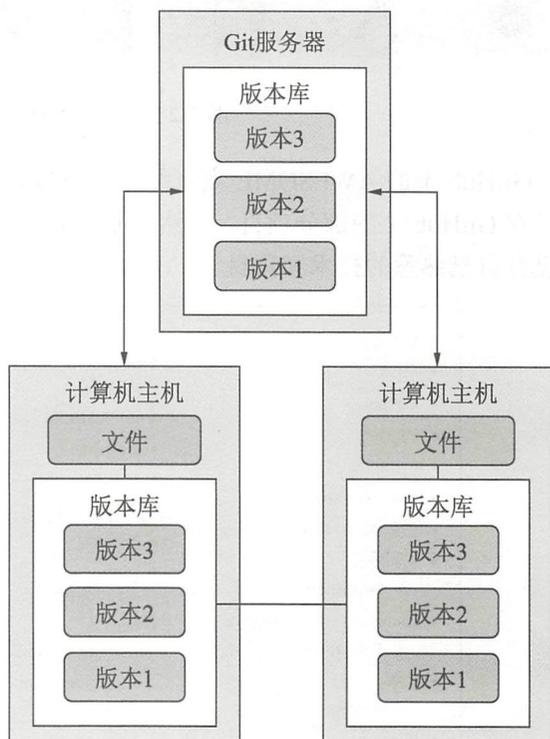


图 3-5 分布式版本控制系统结构示意图

相比SVN这样的集中式版本控制系统，Git的优势主要有以下几点。

- 分布式部署，每一个用户都可以成为版本库，降低了单点系统故障出现的风险。
- 无须联网，本地就可以提交代码。
- 具有更好用的分支体系，当然主要原因也在于版本库在本地。
- 可以本地查看提交的日志记录。
- 性能高，适合各种规模的项目。

虽说Git看似完美，但在实际使用中，也存在以下不足：

- 概念较为复杂，学习曲线陡峭，不如SVN方便入门。
- 权限系统不如SVN严格。

**提示：**版本控制系统的选择，还需要根据具体的应用场景具体分析，没有最好用的版本控制系统，只有最适合的版本控制系统。

### 3.1.3 在 Windows 上安装 Git

Git 支持跨平台安装，一般通过两种方式安装：安装包安装或者编译安装。Linux 系统和 Mac OS 系统上一般都预装了 Git，所以在这里以 Windows 系统为例，学习如何快速安装 Git。以下载安装包方式安装 Git 的具体步骤如下：

#### (1) 下载安装包

在浏览器中访问地址 <https://git-scm.com/downloads>，进入 Git 安装包的下载页面。这里需要选择 Windows 版本后，单击 Downloads for Windows 按钮即进入安装包下载页面，随后选择 64 位版本开始下载，如图 3-6 所示。

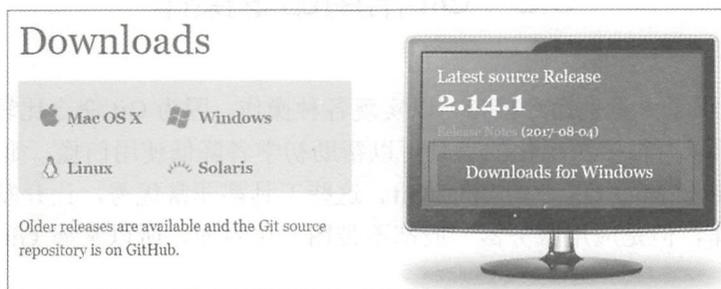


图 3-6 选择合适的 Git 版本并下载安装包

#### (2) Git 安装

安装包下载完成后，双击该安装包即可以开始安装。安装过程与一般软件工具没有太大区别，只不过在安装过程中会遇到很多的单选按钮、复选框，这些都使用默认值即可。Git 在安装结束后，在 Windows 系统下的任意文件夹上右击，若在右键菜单中可以找到 Git Bash Here 命令，说明安装成功，如图 3-7 所示。

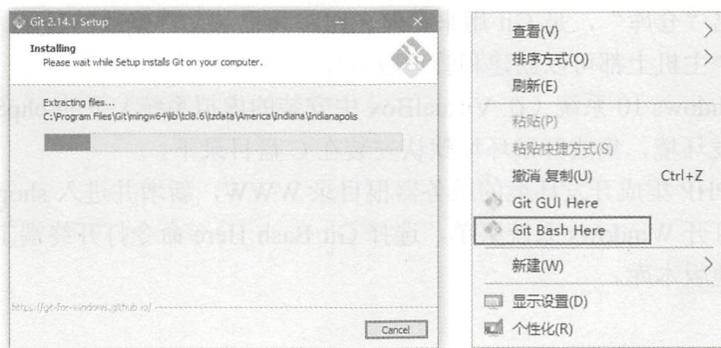


图 3-7 在 Windows 系统下安装 Git 工具

### (3) 查看 Git 的版本

选择右键菜单中的 Git Bash Here 命令，在出现的命令行中输入以下命令，查看 Git 的当前版本：

```
git --version
```

结果如图 3-8 所示，则说明 Git 已经可以正常使用。

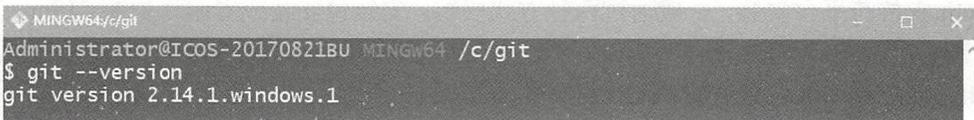


图 3-8 使用命令行查看 Git 的当前版本

## 3.2 Git 常用命令操作

Git 本身提供了大量的命令接口，以实现各种操作。因为 Git 命令比较多，初学者难以快速记忆，所以有很多图形化的工具可以帮助初学者降低使用门槛，如 Windows 系统下的 TortoiseGit 和 Mac OS 下的 SmartGit。这些工具都非常优秀，让开发者可以完全进行可视化的操作，但是应用服务器一般都不带图形化界面，所以掌握 Git 命令还是非常必要的。

初学者在不借助可视化交互工具的情况下学习 Git 的操作时，常常会陷入超多命令的“汪洋大海”中，其过程会非常痛苦。本节精选 Git 中使用率最高的命令逐一讲解，从基础的版本库的创建，到添加文件，再到最终的提交操作，达到增强初学者的记忆和熟练度的效果。

### 3.2.1 创建版本库

版本库又名“仓库”，是 Git 用来存储、跟踪和记录文件修改的目录。因为 Git 分布式的特性，每个主机上都可以创建和管理版本库。

这里以 Windows 10 系统（在 VirtualBox 中安装的虚拟系统）下的 phpStudy 软件作为 PHP 的集成开发环境，集成开发环境默认安装在 C 盘目录下。

(1) 打开 PHP 集成开发环境的服务器根目录 WWW，新增并进入 shop 目录。

(2) 右击打开 Windows 系统菜单，选择 Git Bash Here 命令打开终端工具。输入以下命令初始化本地版本库。

```
git init
```

(3) 创建完毕，出现如下结果：



```
$ git init
Initialized empty Git repository in C:/phpStudy/WWW/shop/.git/
```

(4) 从返回结果可以看到，在 shop 目录下生成了一个 .git 目录。此目录就是 Git 用来管理和跟踪文件变更的，跟 SVN 版本控制工具生成的 .svn 目录类似，但是原理不一样。另外，没有特殊需求不建议手动操作，以防止版本信息丢失。

提示：.git 目录默认是隐藏的，可以在 Git Bash 终端中使用 ls -al 命令查看。

## 3.2.2 提交文件到版本库

3.2.1 节的操作成功创建了一个空版本库，下面讲解如何提交文件到版本库。因为 Windows 系统下的记事本功能比较单一，可能会出现一些文本编码格式的问题，所以本实例使用 Atom 作为默认的文本编辑器，文本编码格式默认为 UTF-8。

在 shop 目录下新增 index.php 脚本文件，并增加以下内容：

```
<?php
echo '现在的时间: '.date('Y-m-d H:i:s');
```

若想提交 index.php 这个文件到版本库中，需要执行以下两个步骤。

(1) 使用 git add 命令，添加文件到版本库中。

```
git add index.php
```

执行后如果没有返回值（UNIX 系统的习惯，无提示说明则表示无异常），说明执行成功。

提示：如果是使用 Git Bash 自带的 Vim 编辑器编辑的 index.php 文件，执行后会发现有如下警告信息：

```
$ git add index.php
warning: LF will be replaced by CRLF in index.php.

The file will have its original line endings in your working directory.
```

之所会出现上述警告信息，是因为 Git Bash 本质上是在 Windows 下模拟 Linux 系统环境，默认使用 LF 换行符格式，而 Windows 系统使用 CRLF 换行符格式，不过 Git 本身预先考虑了这样的问题，内置了自动换行符的转换。

不过团队开发中，代码的换行符一般需要统一，若不想在提交时自动转换，可以执行以下命令语句：

```
git config --global core.autocrlf false
```

Git 自动转换换行符原理如图 3-9 所示。

(2) 使用 git commit 命令，把添加的文件提交到版本库中。

```
git commit -m "首页入口文件"
```

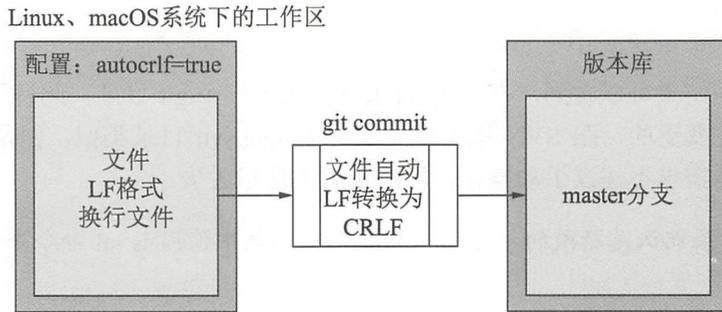


图 3-9 Git 在提交代码时可以自动转换换行符

命令执行后，出现如下结果：

```
$ git commit -m '首页入口文件'
*** Please tell me who you are.
Run
  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"
to set your account's default identity.
Omit --global to set the identity only in this repository.
fatal: unable to auto-detect email address (got
'Administrator@ICOS-20170903PV.(none)')
```

之所以会提交失败，是因为在 Git 安装完成后，没有配置系统的用户名和邮箱。Git 本身是分布式的，所以需要这些信息作为唯一的身份标识。执行以下命令配置用户信息：

```
git config --global user.name "wangjialin"
git config --global user.email "wangjialin.bj@gmail.com"
```

命令中--global 参数说明是全局配置。执行完成后，再次进行提交，完成后提示结果如下：

```
$ git commit -m '首页入口文件'
[master (root-commit) 5635211] 首页入口文件
1 file changed, 2 insertions(+)
create mode 100644 index.php
```

执行命令时，git commit 操作后面的-m 参数用来指定当次提交的注释说明。这里建议大家根据提交内容认真书写，以方便后面的版本管理。

### 3.2.3 Git 查看版本库信息

这时继续修改 index.php 文件，丰富入口文件的内容。修改代码如下：

```
<?php
header('Content-type:text/html;charset=utf8');
echo '当前的时间:'.date('Y-m-d H:i:s');
```

脚本新增了一行代码，模拟常见的文件修改操作。



### (1) 及时掌握版本库的各种信息

为了可以及时掌握版本库的各种信息，可以使用以下命令查看工作区中文件的变化情况：

```
git status
```

执行结果如下：

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
        modified:   index.php
no changes added to commit (use "git add" and/or "git commit -a")
```

在执行结果中，可以看出哪个文件被修改了。

### (2) 查看本次修改的内容

若需要查看本次修改的内容，需要执行以下命令：

```
git diff
```

执行结果如下：

```
$ git diff
diff --git a/index.php b/index.php
index ccd457e..a380dce 100644
--- a/index.php
+++ b/index.php
@@ -1,2 +1,3 @@
<?php
+header('Content-type:text/html;charset=utf-8');
echo '当前的时间:'.date('Y-m-d H:i:s');
```

通过 `diff` 命令，可以看到在 `index.php` 脚本里新增了一行代码。在输出结果中，“+”说明此行文本有修改和增加操作，而“-”说明有删除操作。

### (3) 完成代码提交

随后再次执行提交流程：先执行 `git add` 命令，再使用 `git status` 命令查看文件修改状态，在确认无误后，使用 `git commit` 命令操作，把文件修改提交到 Git 的版本库（本地）中。两次操作的执行结果如下：

```
$ git commit -m '增加了一行代码'
[master a12f7b9] 增加了一行代码
1 file changed, 1 insertion(+)
$ git status
On branch master
nothing to commit, working tree clean
```

 提示：在开发环境下，使用可视化的工具查看文件修改细节，效率会更高。

## 3.2.4 日志查看与版本回退

在多次提交后，有时候因为错误的提交或者其他原因，需要把文件回退到某一个提交



版本，这时候就需要先查看提交的历史记录，检索每次提交的唯一标识，然后再决定版本回退到哪个版本。

### (1) 日志查看

执行以下操作，查看 Git 最新的提交日志。

```
git log
```

可以看到有两次提交记录，根据时间排列从近到远显示，结果如下：

```
$ git log
commit 85af613a2aacc3ef0769a7c8c9d11fbe35cd9760 (HEAD -> master)
Author: wangjialin <wangjialin.bj@gmail.com>
Date: Tue Aug 22 16:05:57 2017 +0800
```

增加了一行代码

```
commit fce3709e8298007dcc733d461be4f913c539ea16
Author: wangjialin <wangjialin.bj@gmail.com>
Date: Tue Aug 22 15:33:45 2017 +0800
```

首页入口文件

可以看到提交 Git 的版本号是一个十六进制的序列字符串，因为 Git 是分布式的，所以不能像 SVN 一样使用纯数字的版本号，否则在多人开发时极易出现冲突。日志里面包含了版本号、提交账号信息、提交时间和提交的备注说明。

**提示：** 可以进行精简版的日志列表显示，只需要在命令后面加上 “--pretty=oneline” 参数即可。添加后，再次执行，结果如下：

```
$ git log --pretty=oneline
85af613a2aacc3ef0769a7c8c9d11fbe35cd9760 (HEAD -> master) 增加了一行代码
fce3709e8298007dcc733d461be4f913c539ea16 首页入口文件
```

### (2) 版本回退

查看日志的时候，若已经发现可以回退的版本号，就可以执行回退操作了。不过在 Git 中版本回退有本地和远程两种类型，两者的操作流程稍有不同，但目前实例的版本库在本地，所以暂时不关注如何操作远程的版本回退。

回退的命令有如下两种方式。

- `git reset --hard commit-id`：将文件回退到指定版本号。适合提交记录比较多，需要回退到某一个提交版本的情况。
- `git reset --hard HEAD~3`：将文件最近 3 次的提交回退。适合对刚提交的文件进行回退，开发者可以自定义回退的次数。

下面以第一种为例，回退版本到第一次的提交版本。

(1) 操作如下命令：

```
git reset --hard fce3709
```



(2) 执行后的输出结果如下：

```
$ git reset --hard fce3709
HEAD is now at fce3709 首页入口文件
```

(3) 打开 `index.php` 脚本文件后，发现代码已经回退到最早提交的版本，第二次提交时在脚本增加的 `header` 方法，已经消失不见了。

```
<?php
echo '当前的时间:'.date('Y-m-d H:i:s');
```

在版本回退时，不需要输入完整的版本号，只需要输入前几位即可，Git 会自动检索匹配。输入的版本号建议在 6 位以上，以免 Git 检索出多条记录。

### 3.2.5 了解工作区、暂存区和版本库

熟悉了以上几个基础的 Git 操作命令后，为了深入理解，下面继续介绍几个 Git 里面的常见概念。

- 工作区：开发者可以看到和操作的文件目录，例如上面实例中的 `shop` 目录。
- 暂存区：因为 Git 的版本库在本地，开发者执行 `git add` 操作后，文件修改等会被提交到暂存区。
- 版本库：工作区目录下的 `.git` 目录下存储了 Git 的版本库信息，除了提交信息、分支信息外，暂存区也被存放在里面。

举例来说，在 `shop` 工作区中，原来只有 `index.php` 一个文本，开发者后来新增了 `phpinfo.php` 文件，执行了 `git add` 命令后，工作区的内容会被提交到暂存区，而执行 `git commit` 命令后，暂存区的内容就会被提交到版本库中的 `master` 分支，如图 3-10 所示。

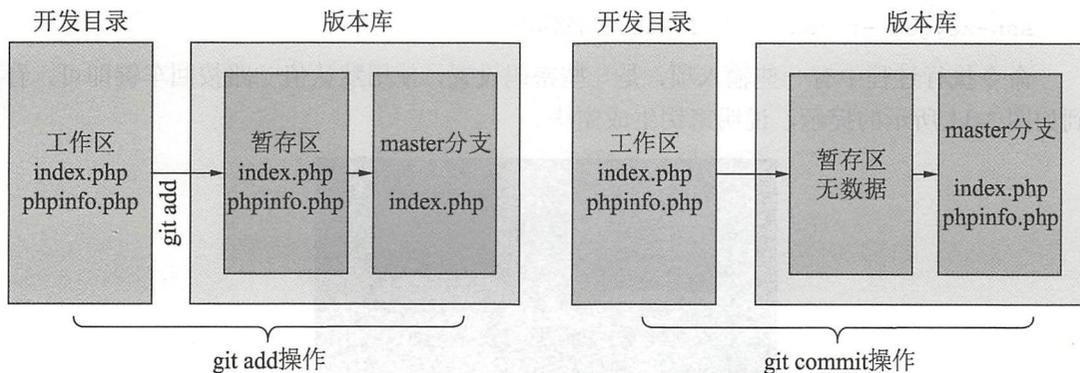


图 3-10 Git 中的工作区、暂存区和版本库

**提示：**暂存区是 Git 里面非常重要的概念，掌握了暂存区的概念，可以加深了解 Git 操作的原理。



## 3.3 GitHub 远程仓库

通过 3.2 节的学习，已经可以在本地创建一个完整的 Git 版本库了。不过在实际开发中，无论是使用 GitHub，还是在本地搭建 Git 版本库，都少不了操作远程仓库。本节重点讲解在 GitHub 上创建远程仓库及其相关操作。

### 3.3.1 在线注册远程仓库

在 Git 看来，每一台计算机都可以成为一个代码仓库，虽然这样说，但是也可以专门搭建一台服务器提供仓库功能。但现阶段搭建服务器比较烦琐，也有各种限制，所以若没有特殊需求，可以在 GitHub 上搭建远程仓库。

GitHub 本身提供有条件的免费 Git 仓库托管服务，注册使用远程仓库的流程如下。

#### (1) 注册账号

只需要有一个邮箱，就可以快速注册。访问 <https://github.com/>地址即可快速注册账号，注册流程这里不再赘述。

#### (2) 创建 SSH Key

因为本地的 Git 仓库和远程仓库是通过 SSH 加密传输的，所以需要先在本地创建 SSH Key 密钥。先进入虚拟机系统的用户目录，用户目录一般是在 C 盘下的 Users 目录，根据用户的登录账号选择（本实例虚拟机目录是 C:\Users\Administrator）。在目录中打开 Git Bash，执行以下命令：

```
ssh-keygen -t rsa -C "GitHub 注册邮箱"
```

命令执行过程中有一些输入项，是一些密码设置，使用默认值一路按回车键即可，看到如图 3-11 所示的提示，说明密钥生成完毕。

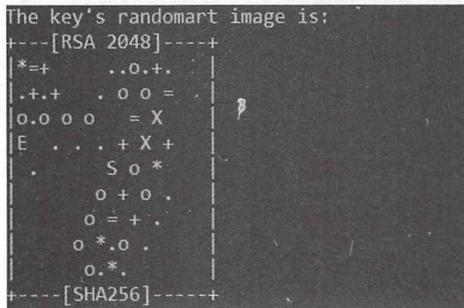


图 3-11 成功创建 SSH 密钥

密钥生成成功后，在用户目录下出现了 .ssh 文件夹，里面包含 id\_rsa 和 id\_rsa.pub 两



个密钥文件，其中 `id_rsa` 文件是私钥，需要妥善保管，不要泄露。而 `id_rsa.pub` 文件是公钥，可以对外使用。生成的文件如图 3-12 所示。

### (3) 在 GitHub 上设置 SSH 密钥

在成功生成了密钥文件后，还需要在 GitHub 上登记这个密钥文件（一般指公钥文件）内容，以方便后面的加密数据传输。首先在 GitHub 上登录账号，单击右上角的头像会出现下拉菜单，选择 `Settings` 命令进入设置页面，如图 3-13 所示。

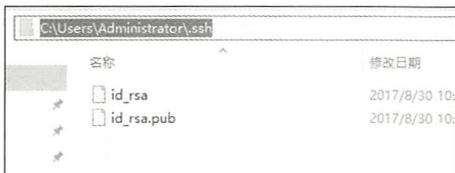


图 3-12 SSH 的公钥和私钥



图 3-13 在 GitHub 的用户菜单中选择 Setting 命令

在设置页面，选择 `SSH and GPG keys`，进入密钥的设置界面，如图 3-14 所示。单击 `New SSH Key` 按钮后，出现密钥输入的表单输入项，其中 `Title` 可以自行定义，`Key` 的内容就是刚才生成的 `id_rsa.pub` 文件内的密钥字符串，使用 `ATOM` 编辑器即可打开查看。

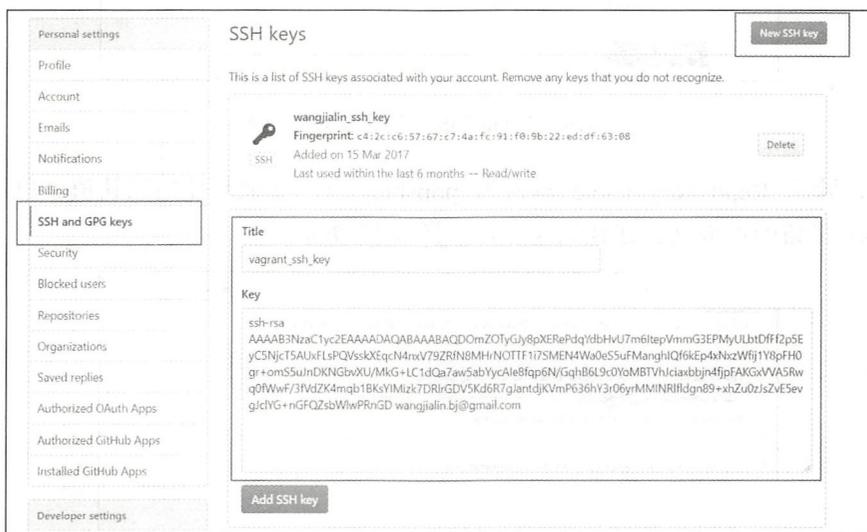


图 3-14 在 GitHub 上设置 SSH 传输公钥内容

**提示：**在进行 `Key` 表单项复制、粘贴操作时，字符串后不要有空格或者回车换行，否则可能会引起添加异常。

单击 `Add SSH key` 按钮后，即可完成公钥的添加，成功添加 SSH 公钥如图 3-15 所示。

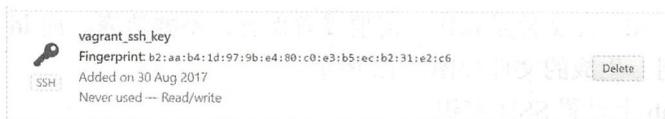


图 3-15 在 GitHub 上成功添加 SSH 公钥

#### (4) 创建远程仓库

添加完成 SSH 密钥后，回到 GitHub 首页，在网页的右侧可以看到 **New repository** 按钮，单击该按钮进入创建代码仓库界面，如图 3-16 所示。

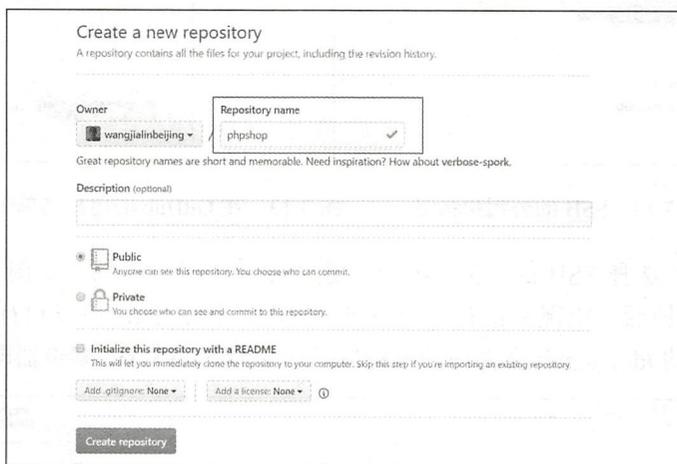


图 3-16 在 GitHub 中创建公共的仓库

其中，除了 **Repository name** 输入为 **phpshop** 外，其他都使用默认值。单击 **Create repository** 按钮即可成功创建代码仓库，如图 3-17 所示。

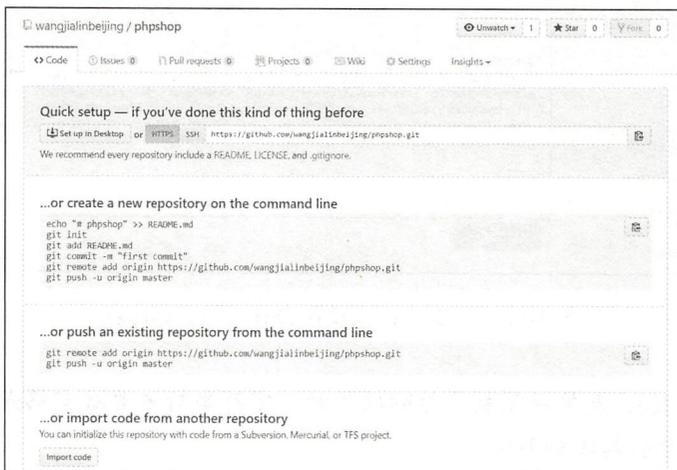


图 3-17 完成代码仓库的创建

GitHub 提示这是一个空仓库，可以进行克隆（复制）操作，也可以关联本地的库进行推送操作。

### 3.3.2 本地操作远程仓库

因为上面的实例中已经有本地库 `shop`，下面来看看如何把它和 `phpshop` 进行关联。首先进入本地库 `shop` 目录中，执行以下命令实现与远程库的关联：

```
git remote add origin https://github.com/wangjialinbeijing/phpshop.git
```

自定义的 `origin` 就是远程库的名称，再执行以下命令，把本地仓库的代码推送上去，其中 `-u` 参数表示 Git 会关联远程的 `master` 分支到本地分支，后续的推送操作可以简化为以下命令：

```
git push -u origin master
```

执行后弹出 GitHub 的登录窗口，如图 3-18 所示。



图 3-18 提示输入 GitHub 的账号与密码

输入 GitHub 账号与密码后执行登录操作，就可以继续推送流程，执行过程和效果提示如下：

```
$ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/wangjialinbeijing/phpshop.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

在本地把代码推送到线上后，在 GitHub 线上查看代码库。通过远程仓库查看代码，

如图 3-19 所示。



图 3-19 在 GitHub 上可以方便地查看版本库的文件详情

除了推送本地仓库到远端，还可以复制远程仓库到本地，这里模拟一台新的机器复制 phpshop 这个远程版本库到本地，执行以下命令：

```
git clone git@github.com:wangjialinbeijing/phpshop.git
```

首次命令执行过程中会看到有警告信息，是 SSH 加密连接的相关确认信息，输入 yes 后按回车键即可继续执行，命令执行结果如下：

```
$ git clone git@github.com:wangjialinbeijing/phpshop.git
Cloning into 'phpshop'...
The authenticity of host 'github.com (192.30.255.112)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWg17E1IGOCspRomTxdCARLviKw
6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.255.112' (RSA) to the list
of known hosts.
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

在本地目录下查看，远程版本库的内容已经同步复制到本地，如图 3-20 所示。

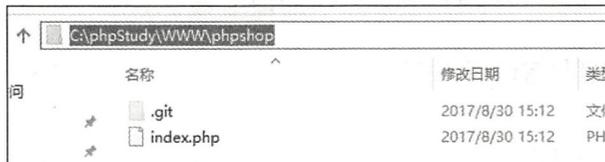


图 3-20 GitHub 远程仓库内容复制到本地

## 3.4 分支、合并与冲突解决

Git 中的分支，本质上是向 commit 提交对象的可变指针。通常情况下，Git 会使

用 `master` 作为分支的默认名字。在多次提交后，开发者默认已经有了一个指向最后一次提交对象的 `master` 分支，而每次提交时都会自动向前移动。同时 Git 中存在一个名为 `HEAD` 的特殊指针，它指向正在工作的本地分支。

### 3.4.1 分支与合并原理

以网站项目实际开发为例，该项目最初只有 `master` 分支（正式站点），提交记录都在这个分支上，所以 `HEAD` 指针默认指向 `master`，如图 3-21 所示。

后来该项目需要一个新功能，于是新建一个名为 `dev`（新功能）的分支，进行相应的开发工作，如图 3-22 所示。

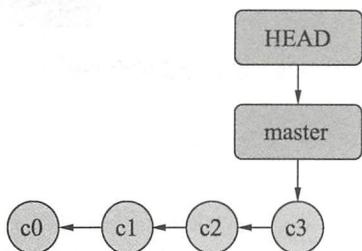


图 3-21 在 `master` 分支上提交代码

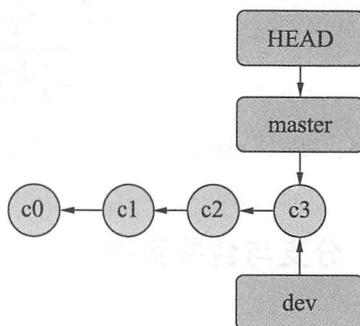


图 3-22 创建新的分支——`dev` 分支

但此时工作区默认还在 `master` 分支下，需要手动切换到 `dev` 分支，此时 `HEAD` 指针就会指向 `dev` 分支，切换完成后工作目录也就变成 `dev` 分支了，此时 `dev` 分支的内容与 `master` 分支的内容并没有不同，都指向 `c3` 次提交，如图 3-23 所示。

开发者于是继续向 `dev` 分支上提交新功能代码，时间轴会继续前进，但即便是版本提交到了 `c5`，`master` 分支的指向并没有发生改变（`c3` 节点），如图 3-24 所示。

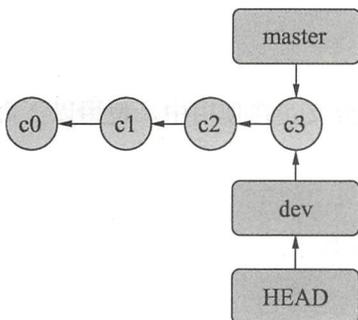


图 3-23 切换工作区到 `dev` 分支

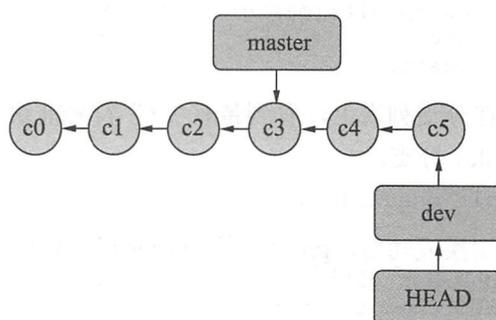


图 3-24 在 `dev` 分支上继续提交代码

在 `dev` 分支上执行了两次提交后，开发者因为某些原因需要切换回 `master` 分支，继续

进行其他开发，此时项目提交时间轴就会流向不同的两个方向，如图 3-25 所示。

随着 dev 分支新功能的开发完毕（提交到 c5）。执行合并操作后，dev 分支的历史使命也就完成，新功能也合并到了 master 分支，开发者可以选择删除 dev 分支，此时时间轴又回归到了 master 分支，功能也上线到了正式站点，如图 3-26 所示。

不过合并时有可能出现冲突，这个就需要开发者手动进行解决，详细内容后面会再介绍。

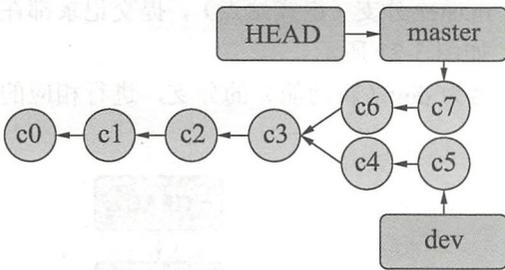


图 3-25 切换回 master 分支继续提交代码

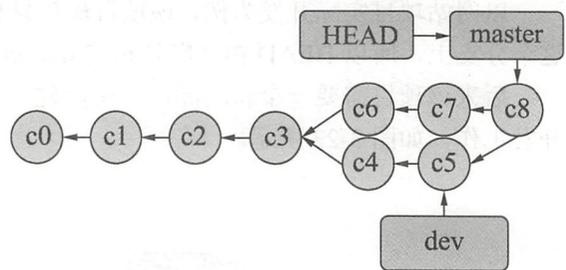


图 3-26 进行分支的合并操作

### 3.4.2 分支与合并实例

以前面讲过的 phpsop 版本库为例，在 master 分支下，只有一个 index.php 文件作为入口文件。模拟项目需要开发一个新功能，此时要建立新的分支，使用命令如下：

```
git branch dev
```

执行此命令成功后，会在本地的版本库中新增一个名为 dev 的分支，随后可以执行相应命令查看分支列表：

```
git branch
```

执行效果如下：

```
phpshop git:(master) git branch
dev
* master
```

在分支列表上，看到带有“\*”符号前缀，说明此分支正在使用中，使用以下命令切换到 dev 分支：

```
git checkout dev
```

切换成功后，git 后面的 master 已经换成 dev，结果如下：

```
phpshop git:(master) git checkout dev
Switched to branch 'dev'
phpshop git:(dev)
```

随后在 dev 分支工作区下，创建新的脚本文件 function.php，增加一些简单的 PHP 脚本代码，实现模拟对新文件的编辑行为，代码如下：

```
<?php
// 显示标准格式化日期
function show_time()
{
    return date('Y-m-d H:i:s');
}

```

执行新增和提交命令，输出结果如下：

```
$ git add function.php
$ git commit -m '自定义函数文件'
[dev 8415ffc] 自定义函数文件
1 file changed, 6 insertions(+)
create mode 100644 function.php

```

当 dev 分支的开发任务结束后（编写完成 function.php 文件），需要切换回 master（主线）分支进行合并操作。首先要先切换到 master 分支，执行以下命令：

```
git checkout master
```

切换成功后，会发现在 dev 分支下开发的 function.php 脚本文件已经消失不见，结果如下：

```
$ git:(dev) git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
$ git:(master) ll
total 8
-rw-r--r-- 1 wangjialin staff 51B 9 2 14:14 index.php

```

执行以下命令，合并 dev 分支到 master：

```
$ git:(master) git merge dev
Updating fce3709..8415ffc
Fast-forward
 function.php | 6 ++++++
1 file changed, 6 insertions(+)
create mode 100644 function.php

```

此时 function.php 文件已经被合并过来，输出结果如下：

```
$ git:(master) ll
total 16
-rw-r--r-- 1 wangjialin staff 101B 9 2 14:43 function.php
-rw-r--r-- 1 wangjialin staff 51B 9 2 14:14 index.php

```

### 3.4.3 冲突解决

在 3.4.2 节的实例中，合并结果是比较理想的情况，因为在 dev 分支中，未对原有 master 分支中的 index.php 脚本进行修改，dev 分支也只是新增了 function.php 文件，所以合并时互相不会互相影响。但在实际项目中这样的应用场景较少，大部分时候情况都是：同一个文件在不同分支都有不同的修改，此时再进行合并操作，就不可避免地会出现各类冲突。

Git 在发现冲突时，会把冲突的文件内容标记出来，同时会暂停提交行为。因为 Git

无法决定冲突代码的修改，所以需要开发者手动解决冲突后，才可以继续下一步提交操作。

这里模拟一个冲突的发生，首先切换到 dev 分支，修改 index.php 文件，代码如下：

```
<?php
//删除了一行代码
.....
```

然后在 dev 分支上新增操作后提交代码，执行过程如下：

```
$ git:(dev) git add .
$ git:(dev) git commit -m='修改代码'
[dev ad09881] =修改代码
 1 file changed, 1 insertion(+), 1 deletion(-)
```

切换到 master 分支后，修改 index.php 脚本文件，代码如下：

```
<?php
//我也删除了一行代码
.....
```

保存代码后执行如下新增提交操作，随后合并 dev 分支到 master 分支：

```
$ git:(master) git merge dev
Auto-merging index.php
CONFLICT (content): Merge conflict in index.php
Automatic merge failed; fix conflicts and then commit the result.
```

此时 Git 提示在合并时，index.php 文件出现了冲突，解决后才可以提交。此时使用编辑器打开 index.php 脚本文件，发现格式如下：

```
<?php
<<<<<<<< HEAD
// 我也删除了一行代码
=====
// 删除了一行代码
>>>>>>> dev
```

在返回结果中可以发现，Git 使用“<<<<<<<<HEAD”到“=====”的符号区间，标识 master 分支下修改的代码，使用“=====”到“>>>>>>> dev”符号区间，标识 dev 分支下修改的代码。解决冲突需要开发者手动去除这些提示符号，然后手动合并代码。

编辑 index.php 文件，删除了无用的内容后，对代码进行了重新组合，修改为如下内容：

```
<?php
// 解决冲突后的提示语：删除了一行代码
.....
```

保存脚本文件后，在 master 分支下进行提交操作：

```
$ git:(master) git add .
$ git:(master) git commit -m='解决冲突'
[master 37103e4] =解决冲突
```

当然实际项目中，冲突的类型和数量远比实例中复杂，少则几个文件，多则数十个文件。在解决冲突时，需要根据实际情况来分析到底该如何合并代码。

## 3.5 使用 GitHub Pages 搭建个人博客站点

本节将借助 GitHub 推出的免费个人博客服务，搭建一个全功能的个人博客站点。学习完本节内容，读者将会对 Git 和 GitHub 有更加深入的了解。

### 3.5.1 传统博客与 GitHub Pages

相比博客，现在微博、朋友圈和短视频已经成为最流行的社交内容传播渠道，但是对于开发者来说，博客依旧是知识分享与技术沉淀的绝佳方式。发表博客需要有对应的平台，常见的博客搭建需要以下条件：

- 服务器、域名，使用 Wordpress、Typecho 等开源工具搭建或自行开发博客网站后开发搭建。
- 使用第三方的博客服务，如 CSDN、博客园和 GitHub 等。另外除了这些技术类的博客平台，在豆瓣、简书、知乎上都可以发布个人博客。

常见可以发布个人博客的技术类社区，如图 3-27 所示。



图 3-27 常见的可以发布个人博客的开放技术类社区

其中前一种，除了要搭建个人服务器，进行运营和维护外，还需要申请域名服务，对于一般的新手来说，搭建个人博客的复杂度和成本都很高。若是对网站的功能性要求不高，可以在搭建个人博客时，使用第三方的博客服务。

GitHub Pages 给开发者提供了托管个人网站的途径。开发者既可以绑定自己的域名，也可以使用 GitHub 提供的域名服务，因为其本身提供了 1GB 的存储空间，完全可以满足一般博客的使用需求。典型的 GitHub Pages 域名格式如下：

你的 GitHub 用户名.github.io

例如，我的 GitHub 用户名为 wangjialinbeijing，网站部署成功后，只需要访问以下地址即可：

wangjialinbeijing.github.io

除了个人站点外，GitHub Pages 还支持更高级的特性，完全可以自行定制。不过因为暂时用不到，这里不深入讲解。

需要注意的是，GitHub Pages 目前只支持静态页面，所以使用 GitHub Pages 搭建博客，与传统意义上使用内容管理后台发布文章，在技术实现原理上有一些不同。

首先看下传统发布博客文章的流程，如图 3-28 所示。

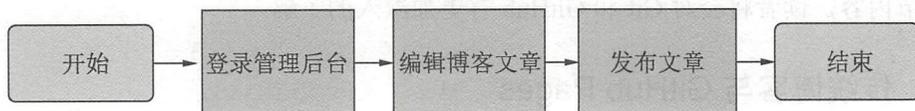


图 3-28 传统内容平台发布文章

而在 GitHub Pages 上发布文章的，就要烦琐得多，如图 3-29 所示。

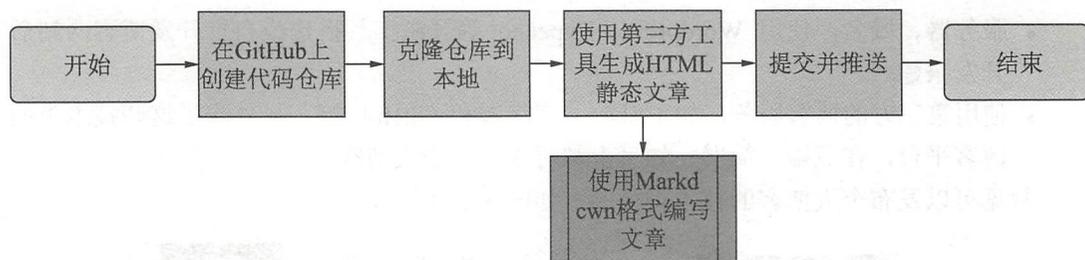


图 3-29 使用 GitHub Pages 发布博客文章

仔细观察会发现，在 GitHub Pages 上发布文章，流程与 Git 提交、推送操作一致，另外为了简化流程，一般都会借第三方工具(如 Hexo JavaScript 库等)，把编写好的 Markdown 格式文章，动态生成 HTML 静态文件，最终再上线发布。

## 3.5.2 使用 Hexo 框架生成静态网站

### 1. Node.js和Hexo框架

除了要在本地安装 Git 和注册 GitHub 账号外，还需要安装 Node.js。Node.js 是一个可以让 JavaScript 语言编写服务器应用的平台，其本身采用事件驱动、异步编程的思想，可以显著地提升开发效率，也弥补了 JavaScript 语言本身的很多不足。不过在这里使用 Node.js，目的是便于安装 Hexo 库进行静态页面的生成，其他内容在此不再深入讲解。这里以 Windows 操作系统为例，访问 <http://nodejs.cn/download/>后，选择符合当前系统的版本下载，如图 3-30 所示。

本实例选择下载 Windows 系统的.msi 安装文件（64 位），安装过程中，除了注意把工具安装到非系统盘外，其他都选择默认，部分安装过程如图 3-31 所示。



图 3-30 获取 Node.js 的安装文件

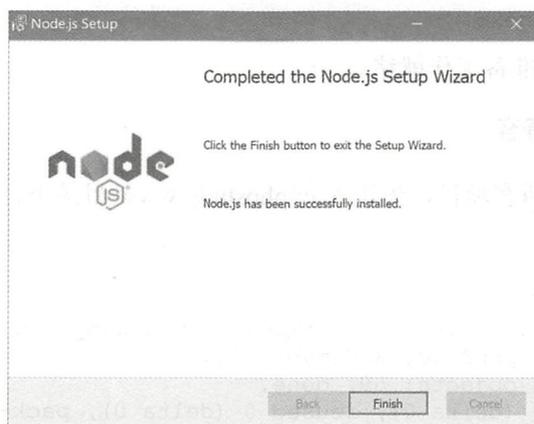


图 3-31 完成 Node.js 的安装

在任意目录下打开 Git Bash 终端工具，输入以下命令查看 Node.js 的版本：

```
node -v
```

输出结果如下，说明安装成功。

```
$ node -v  
v8.4.0
```

同时还需要查看 NPM 是否安装成功，输入以下命令：

```
npm -v
```

输出结果如下，则说明 NPM 也安装成功。

```
npm -v  
5.3.0
```

NPM 即 Node 包管理器（Node Package Manager）。它是一个以 JavaScript 脚本语言编写的软件包管理系统，默认环境为 Node.js，使用 NPM 可以在环境部署时避免手动引入各种功能包，可以提升包管理的效率。

另外，因为 NPM 的源镜像在国外，为了加快访问速度，这里使用淘宝 NPM 镜像替代官方版本。使用淘宝定制的 CNPM 命令工具替代官方的 NPM，操作命令如下：

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

执行成功后，结果显示如下：

```
$ npm install -g cnpm --registry=https://registry.npm.taobao.org
C:\Users\Administrator\AppData\Roaming\npm\cnpm->C:\Users\Administrator\
AppData\Roaming\npm\node_modules\cnpm\bin\cnpm
+ cnpm@5.1.1
added 643 packages in 18.155s
```

随后安装 Hexo 框架。Hexo 是一个快速、简洁且高效的 JavaScript 博客框架，使用 Markdown（或其他渲染引擎）解析文章，只需要几秒时间，即可利用靓丽的主题生成静态网页。这里使用淘宝的 cnpm，替代 npm 工具，安装命令如下：

```
cnpm install -g hexo
```

等待安装完毕后，准备工作就绪。

## 2. 在本地初始化博客

使用 Hexo 初始化博客项目，先进入 c:\phpstudy\www 目录下，执行以下命令：

```
hexo init myblog
```

部分执行结果如下：

```
INFO Cloning hexo-starter to C:\phpStudy\WWW\myblog
Cloning into 'C:\phpStudy\WWW\myblog'...
remote: Counting objects: 59, done.
remote: Total 59 (delta 0), reused 0 (delta 0), pack-reused 59
Unpacking objects: 100% (59/59), done.
Submodule 'themes/landscape' (https://github.com/hexojs/hexo-theme-
landscape.git) registered for path 'themes/landscape'
Cloning into 'C:\phpStudy\WWW\myblog/themes/landscape'...
remote: Counting objects: 785, done.
remote: Total 785 (delta 0), reused 0 (delta 0), pack-reused 785
Receiving objects: 100% (785/785), 2.54 MiB | 1.37 MiB/s, done.
Resolving deltas: 100% (403/403), done.
Submodule path 'themes/landscape': checked out 'decdc2d9956776cbe95420ae
94bac87e22468d38'
INFO Install dependencies
.....
WARN Failed to install dependencies. Please run 'npm install' manually!
```

可以看出 Hexo 在初始化 myblog 这个目录时，在 GitHub 上复制了一些文件到本地。最后需要执行以下命令进行安装：

```
npm install
```

执行过程反馈如下：

```
$ npm install
npm WARN saveError ENOENT: no such file or directory, open 'C:\phpStudy\
WWW\package.json'
npm notice created a lockfile as package-lock.json. You should commit this
file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\phpStudy\WWW\
```

```
package.json'  
npm WARN WWW No description  
npm WARN WWW No repository field.  
npm WARN WWW No README data  
npm WARN WWW No license field.  
up to date in 0.031s
```

执行到这一步，Hexo 的博客项目 myblog 就已经初始化成功了。进入到 myblog 目录下，执行以下命令，可以开启 Hexo 自带的服务器，方便开发者预览网站效果。

```
hexo s
```

执行命令后，暂时不要关闭 Git Bash 命令行工具，在浏览器中访问地址 <http://localhost:4000/>，可以查看本地的静态网站，效果如图 3-32 所示。

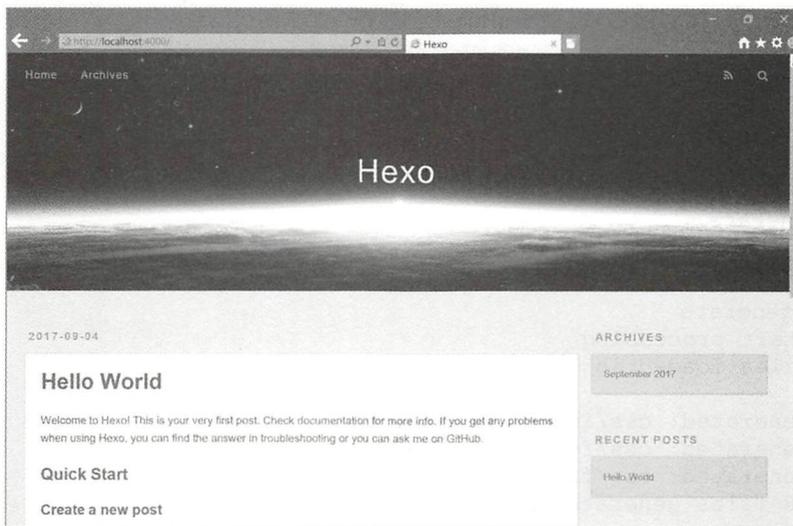


图 3-32 在本地预览 Hexo 生成的静态网站

提示：开启 Hexo 服务器时，若系统出现是否同意网络授权的请求，需执行同意操作。

### 3.5.3 推送文章网站到 GitHub Pages

在本地初始化 Hexo 结束后，已经可以快速生成静态网页。此时需要把这些静态网页提交推送到 GitHub 的远程仓库中才可以正式访问。

#### 1. 关联远程仓库

新建仓库时，不能随便输入命令，其命令格式如下：

```
你的用户名.github.io
```

其中，github.io 就是你的免费域名后缀。创建一个符合要求的 GitHub 仓库，如图 3-33 所示。



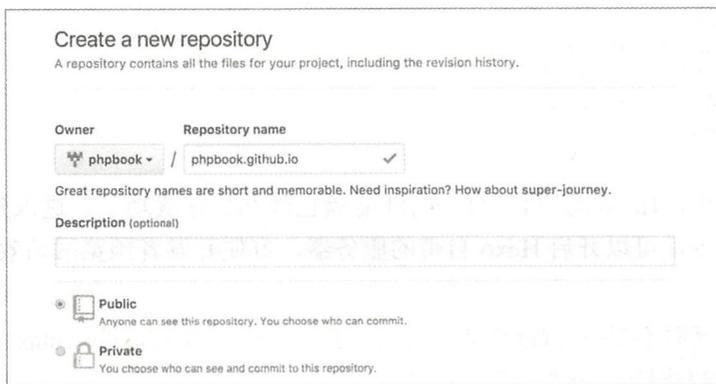


图 3-33 在 GitHub 上创建符合要求的代码仓库

仓库创建成功后，返回到本地，进入 myblog 目录，执行以下命令：

```
npm install hexo --save
```

随后执行以下命令，用于生成静态网页：

```
hexo generate
```

执行效果如下：

```
$ hexo generate
INFO Start processing
INFO Files loaded in 72 ms
.....
INFO Generated: css/images/banner.jpg
INFO Generated: css/fonts/fontawesome-webfont.svg
INFO Generated: css/fonts/fontawesome-webfont.ttf
INFO 28 files generated in 386 ms
```

可以看出 Hexo 自动创建了很多 HTML、JavaScript 和 CSS 等静态文件。这些文件存放于 myblog 下的 public 目录中。静态文件有了，我们还需要把这些文件提交到新创建的版本库中，这里需要先安装一个 Hexo 插件，用于自动使用 Git 部署静态文件，命令如下：

```
cnpm install hexo-developer-git --save
```

完成后，使用编辑器打开 \_config.yml 文件，用来修改 Hexo 项目的配置。修改文件末尾的配置项，命令如下：

```
deploy:
  type: git
  repository: git@github.com:phpbook/phpbook.github.io.git
  branch: master
```

以上配置项定义了部署类型为 Git，访问地址，以及要使用哪个分支。

 **提示：**在配置所有的 \_config.yml 文件时（包括 theme 中的），在所有的冒号后边都要加一个空格，否则执行 Hexo 命令会报错。

执行以下命令进行页面部署：



```
hexo deploy
```

部分执行结果如下:

```
$ hexo d
INFO Deploying: git
INFO Clearing .deploy_git folder...
INFO Copying files from public folder...
INFO Copying files from extend dirs...
.....
To github.com:phpbook/github.io.git
* [new branch] HEAD -> master
INFO Deploy done: git
```

完成后在浏览器中访问地址 <https://phpbook.github.io>, 发现已经和在本地访问 <http://localhost:4000> 效果一致。另外, 在 GitHub 的仓库中, 可以看到推送成功的代码, 如图 3-34 所示。

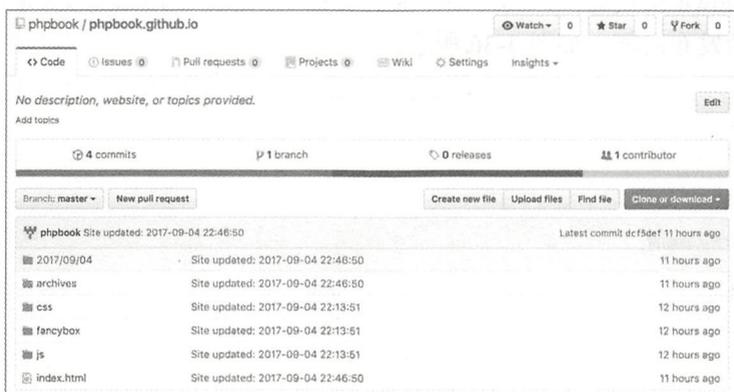


图 3-34 使用插件自动提交 Hexo 静态文件

另外, GitHub Pages 的设置信息可以通过单击图 3-34 中顶部导航的 Settings 按钮查看, 如图 3-35 所示。

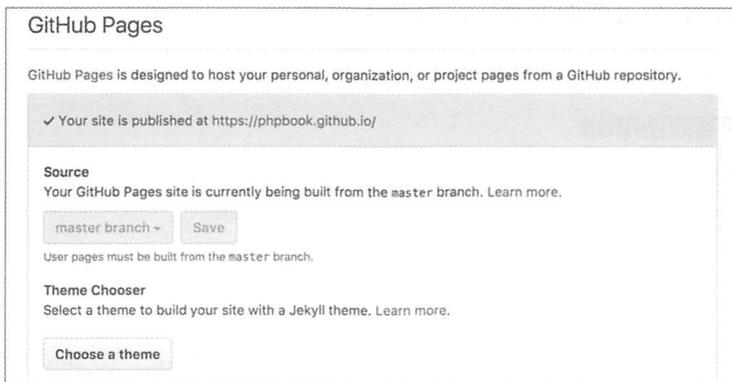


图 3-35 在 GitHub 中设置 Pages 相关的内容



提示：部署后可能存在延迟，若无法访问，请耐心等待。

## 2. 发布文章

借助 Hexo 工具，用户可以使用命令行快速创建博客文件。例如，创建名为“如何高效使用 Git”的文章源文件，可以执行以下命令：

```
hexo new "如何高效使用 Git"
```

执行结果如下：

```
$ hexo new "如何高效使用 Git"
```

```
INFO Created: C:\phpStudy\WWW\myblog\source\_posts\如何高效使用 Git.md
```

从返回结果中可以看出，Hexo 在根目录下 source 文件夹中的 \_posts 目录下，创建了一个 .md 文件（Markdown 格式文件），对应新创建的文章。这里建议使用免费的 Typora 工具编辑 Markdown 格式文件，其中预生成的文档中，title 字段填写文章的标题，date 字段为默认生成的发布时间，如图 3-36 所示。



图 3-36 使用 Typora 编辑器编辑 Markdown 格式文件

保存文件后，再次执行 hexo generate 和 hexo deploy 命令，把新编辑的文章生成静态网页，最后推送到线上。在浏览器中访问地址 <https://phpbook.github.io>，可以看到页面内容已经更新，可以看到最新的文章了，如图 3-37 所示。



图 3-37 本地生成文章并发布到 GitHub Pages



### 3.5.4 GitHub Pages 使用小技巧

完成以上内容，开发者就可以方便地在 GitHub Pages 上发布文章了，不过我们暂时只是使用了 Hexo 默认的模板，下面继续来看几个实用的小技巧。

#### 1. Hexo网站配置

成功发布文章后，我们还需继续修改网站的基本信息，如网站标题、简介等。编辑 myblog 目录下的 `_config.xml` 配置文件，其中常见的几个配置选项如表 3-1 所示。

表 3-1 Hexo基本网站的配置选项说明

参 数	描 述
title	网站标题
subtitle	网站副标题
description	网站描述
author	您的名字
language	网站使用的语言
timezone	网站时区。Hexo 默认使用用户计算机的时区，比如America/New_York、Japan和UTC

例如，修改 `_config.xml` 文件中以下几个配置选项：

```
# Site
title: PHPBOOK
subtitle: 每天学习一小点
description: PHP 知识分享站点
author: 王甲临
language: PHP
timezone: Asia/Shanghai
```

随后执行命令发布静态页操作，推送到 GitHub 上，访问 <https://phpbook.github.io>，可以发现网站信息已经更新，如图 3-38 所示。

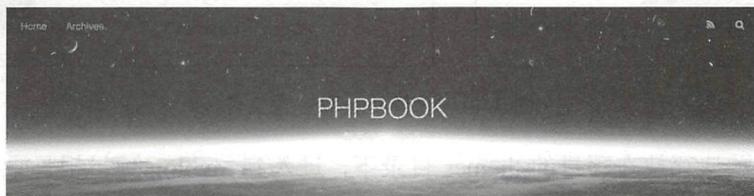


图 3-38 在 Hexo 中自定义网站基本信息

提示：Hexo 框架更多的配置说明，可以访问地址：<https://hexo.io/zh-cn/docs/index.html>。

#### 2. 更换模板主题

对于 Hexo 工具，官方提供了很多个性化的主题，访问地址 <https://hexo.io/themes/>，可



以挑选自己喜欢的主题进行下载安装。Hexo 的主题都托管在 GitHub 上，单击图片可以进入主题发布者的博客中预览效果，单击图片下的文本可以进入 GitHub 获取主题源码，如图 3-39 所示。

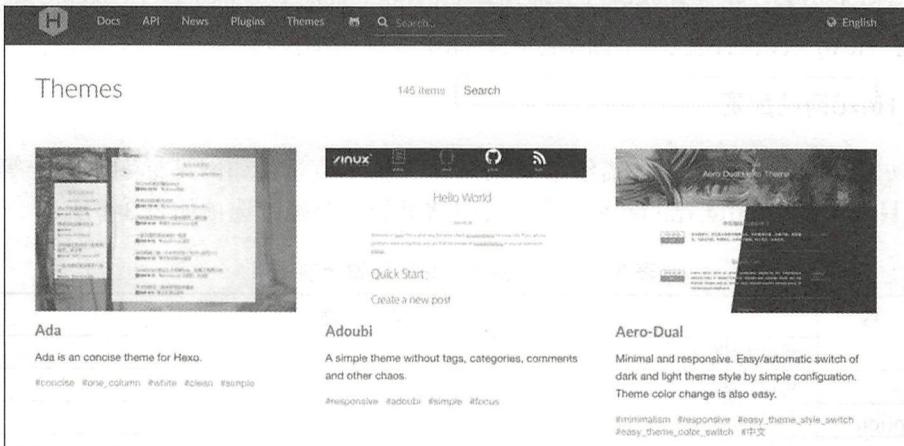


图 3-39 Hexo 提供了丰富的主题市场

这里以更换默认主题为 Anisina 为例，讲解如何更换模板主题。单击标题进入 GitHub，随后复制仓库地址，如图 3-40 所示。

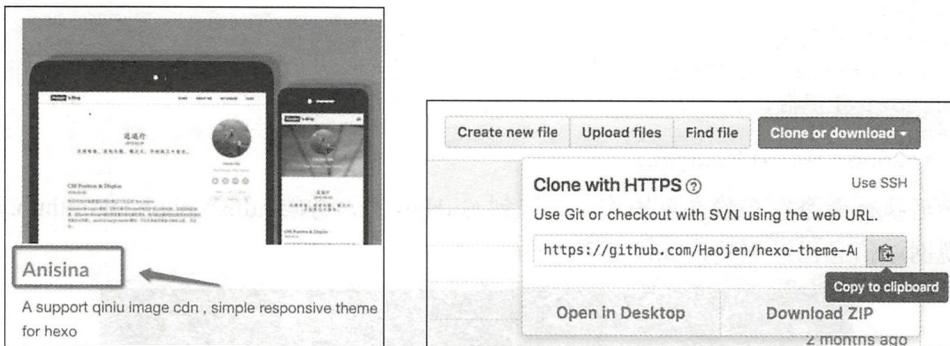


图 3-40 选择并获取主题文件

主题需要默认存放在项目的 themes 目录下，进入到 myblog 项目根目录后，执行以下命令下载主题：

```
git clone https://github.com/Haojen/hexo-theme-Anisina.git themes/
anisina
```

下载完毕后，修改 config.xml 文件，更换默认主题为 anisina：

```
# Extensions
## Plugins: https://hexo.io/plugins/
## Themes: https://hexo.io/themes/
theme: anisina
```



执行 `hexo clean`、`hexo generate` 等命令重新发布静态网站，使用 `hexo server` 命令在本地查看效果。访问地址 `http://localhost:4000/`，效果如图 3-41 所示。



图 3-41 更换 Hexo 的默认主题

另外，每个主题都有单独的配置文件，在 `themes/主题安装目录/_config.xml` 文件中，配置项的格式和用法基本上和 Hexo 默认的配置一致，不同之处在于可能存在主题开发者自定义的配置项，比如配置第三方的插件，代码如下：

```
# Duoshuo settings
duoshuo_username: # 你的多说账号
# Share component is depend on Comment so we can NOT use share only.
duoshuo_share: true # set to false if you want to use Comment without Sharing
```

在本节中，Hexo 工具帮了大忙，开发者不用再一行行编写 HTML 代码，一切都可以自动生成。虽然只能发布静态内容，但对于一般的文本、图片而言已经完全够用了。另外，除了本书中讲到的一些配置项，Hexo 可以灵活搭配的东西还有很多，有兴趣的开发者可以进一步了解和使用的。



## 第 4 章 高效团队协作

团队的规模越大、分工越细致，则不同团队协作的效率就会因为任务分配、沟通交流的不同，甚至代码托管平台的不同而产生巨大的差异。为了更好地提高团队协作的效率，本章将重点讲解一些常用团队协作工具的使用，也为一些初创企业和团队提供经验参考。

### 4.1 沟通和交流很重要

沟通和交流不仅对技术导向型的企业和团队很重要，也是所有工作开展的基础。

#### 4.1.1 术业有专攻——企业即时通信工具

目前在国内的互联网行业中，出现了微信、QQ 这样十亿用户级别的巨无霸即时通信工具。我们的生活中已经离开不了这些工具。虽然很多人使用微信、QQ 等工具交流，但在一些细分领域，依旧有其他的沟通软件存在，其中包含一些专门服务于企业交流的软件和工具。

企业或者团队应该以能够提高工作效率为目标，对应不同的应用场景，选择适合的即时通信工具。根据公司规模、团队性质和工作内容的不同，选择也不同，以下是常见的几种选择方案。

##### 1. 创业公司和初创团队的选择方案

如果没有特殊的要求，普通的即时通信软件（如微信和 QQ）就可以覆盖这部分群体。这些工具不仅受众广、跨平台方便，而且基本上没有使用成本，也不用担心其稳定性。国内有名的普通即时通信工具大部分都出自于腾讯和阿里巴巴（如图 4-1 所示），大部分读者应该对这些软件都不陌生。



图 4-1 生活中常见的即时通信软件



不过这些工具若在工作时使用，容易导致工作和生活无法分离。对于初创公司，这些都不是最重要的影响因素，降低成本和减少沟通障碍才是第一要务，毕竟活下来才能可持续发展。

## 2. 中小企业的选择方案

这些企业的人数相比初创团队要多很多，都在几十人甚至上百人不等。在工具的选择上，则更加侧重于某些功能，比如是否可以进行考勤的管理，能否进行报销和审批等。当人数规模达到一定程度后，如果同时使用过多的工具，则会影响效率，若可以在一个系统里无缝对接，无疑将会大为提高效率。

这里主要推荐阿里的钉钉和腾讯的企业微信两种即时通信工具，如图 4-2 所示。



图 4-2 主流的企业即时通信工具

### (1) 钉钉

钉钉和企业微信在产品定位上各有不同，使用钉钉的优势有如下：

- 每条消息都有已读、未读状态标识。
- 网盘、邮件、电话、考勤和审批功能深度集成。
- 进一步地细分用户角色。
- 支持第三方应用市场和接口，强调扩展性。
- 支持阿里系工具，如支付宝等。

钉钉强调的是自上而下的执行力（强调阅读回执），但其功能全面、复杂，也在一定程度上提高了用户上手的成本，这让钉钉更适合 IT 互联网企业。

钉钉移动端中的常见功能如图 4-3 所示。

### (2) 企业微信

与钉钉的产品定位不同，企业微信刚起步，其优势如下：

- 使用习惯类似微信，用户可以无缝上手，门槛低。
- 可以批量地导出聊天记录和文件。
- 具有强大的企业通信录和外部联系人。
- 具有开放的 API 接口。
- 支持腾讯系工具，如 QQ 企业邮箱等。

对比钉钉的执行力，企业微信更强调沟通的本质，尤其像“下班了”功能，可以实现工作和生活的分离。企业微信移动端界面如图 4-4 所示。



图 4-3 钉钉移动端中的实用功能



图 4-4 企业微信移动端界面

无论是更看重执行力的钉钉，还是更加“人性化”的企业微信，只要可以提高企业或者团队的沟通效率，就是好工具。

**提示：**无论是钉钉还是企业微信，都是有全平台的客户端可供使用，所以不用担心系统，只需要根据功能选择即可。

## 4.1.2 文档积累和文件分享

在团队协作中，知识的积累非常重要，却往往总是被忽视。若可以把平时工作中的讨论、修改历史、问题解决的方案，都以笔记和文档的形式整理出来，新人就可以更好地融入到团队中去，提高工作效率。

### 1. 文档编写与协同操作

挑选合适的文档协同工具，需要考虑到以下几点：

- 支持多人协作编写同一份文档。
- 支持主流的文件格式。
- 支持版本修改记录。
- 支持评论与分享。
- 有较好的用户体验。

在这里推荐石墨文档，基本上可以满足以上的要求，石墨文档号称“最优美的在线协作文档”，在写文档时，不仅支持 Markdown 的部分语法，在协作、评论、修改历史和编写人员方面也是一应俱全，文档的编写体验优异。

访问地址：<https://shimo.im>，可以方便地注册和使用石墨文档，一切都是线上操作，无需下载和安装本地客户端。另外，石墨文档还可以和钉钉协作，提高工作效率。石墨文档的管理界面如图 4-5 所示，简约的设计风格非常讨喜。



图 4-5 石墨文档的管理界面

文档的编辑效果如图 4-6 所示。

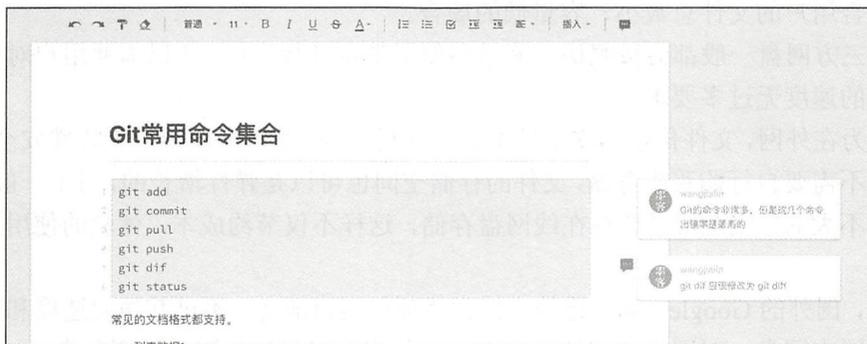


图 4-6 丰富的格式和文档评论

除此之外，还可以进行多成员的协同编辑，操作上只需要添加协作者账号即可，如图 4-7 所示。需要注意，协作者要求是已经注册完成的石墨文档用户。



图 4-7 文档编辑时可以方便地添加协作者

另外，还可以导入本地的文件，微软办公套件里面的常见文件格式基本上都支持，如 Word 和 Excel 文件格式，后续还会支持 PPT 格式。越来越多文档通用格式的增加，减少了因为线下和线上格式不统一导致的额外工作量。

 **提示：**对文档安全性要求较高的用户，可以尝试在本地搭建局域网文件管理系统。

## 2. 文件存储与分享

企业和团队因为工作性质的不同，所需要的文件存储与分享的方式也不尽相同。例如，一些参与影视制作的公司，素材源代码多以 TB（1TB=1024GB）为单位。而另外一些技术团队，则需要特定的文件，随时可以在任何地方进行编辑。可见不同性质的团队，对于文件存储的要求也是不一样的，如何选择成本更低、更合适的存储方式，都是一个值得思考的问题。

目前主流的企业级文件存储，主要分为以下两种。

### （1）在线网盘

国内的在线网盘，服务商如七牛云、百度网盘、坚果云和阿里云等都有对应的网络存储服务，根据用户对文件存储大小和流量的需求，收费也不同。可以通过外网进行访问的文件存储，主要应对以下应用场景。

- 第三方网盘容量成本高，一般除了按容量收费外，还需要按照使用时长收费，所以适合用户的文件总数小、容量低的场合。
- 第三方网盘一般都会按照访问流量和使用带宽进行收费，所以需要用户对下载和上传的速度无过多要求。
- 因为在外网，文件传输的安全性不如本地局域网，所以需要用户无特殊安全性要求。

因为不需要自行购买服务器，文件的存储空间也可以是弹性增长的，对于一般的团队，文件体量不大的，完全可以选择在线网盘存储，这样不仅节约成本，体验的使用上也不会太差。

另外，国外的 Google、亚马逊等也提供优质的网盘服务，不过其访问速度和稳定性等肯定不如国内网盘，对网络访问请求要求较高的用户尽量选择国内的服务商。

国内常见的网盘服务商，如图 4-8 所示。



图 4-8 国内常见的网盘服务商

### （2）局域网文件服务器

在局域网内搭建的文件服务器，不对外开放，其安全性相比网盘有了很大的提高，适合以下应用场景：

- 文件存储量大，如一些影视素材和备份文件等。
- 更高的安全性要求。
- 局域网内访问更快的下载、上传速度。
- 存储硬件的选择更为灵活，除了使用硬盘，甚至可以使用磁带（企业级存储）和蓝光光盘等存储介质。

搭建局域网的文件服务器，除了使用一般的应用服务器改造外，还可以使用专门的硬件进行存储，如 NAS。NAS 的全称是网络连接存储设备（Network Attached Storage），是一种专门的数据存储技术的名称，它可以直接连接在计算机网络上，对异质网络用户提供了集中式数据访问服务。NAS 和一般的应用服务器没有本质区别，无论是局域网还是外网，都可以方便地访问，用户可以根据实际需求自行搭配。

除了使用专用的硬件，一些团队也会直接在 Linux 系统下搭建文件服务器。最简单的方法就是使用 Apache+HTTP 的方式进行文件传输，这样只需要修改 Apache 的文件权限，就可以让开发者直接查看、下载服务器的根目录。

 **提示：**无论是购买专门的文件存储硬件，还是在 PC 硬件上搭建文件服务器，都需要文件存储系统的支持。

## 4.2 任务分配、代码托管和缺陷管理

企业或者团队的人员越多，分工越详细，越需要各式各样的工具软件以辅助工作，本节主要讲解在不同的工作场景下，都需要哪些工具，以便更好地提高工作效率。

### 4.2.1 任务分配

在工作中，若直接通过邮件、文档进行任务分配，在产品或者项目进行时，无法及时跟踪项目的进度，人力成本会被极大地浪费掉。所以一般的企业或者团队都会选择任务管理系统，对人员进行任务分配和跟踪，可以方便地及时跟进整个项目的进展。

除了对任务进行管理外，无论是使用 SVN，还是 Git 进行代码的版本控制，一般都会搭建集中的代码仓库，以实现团队之间的代码共享。

而在项目进行时，测试团队还需要及时地编写测试用例，向缺陷管理系统提交问题反馈，也就是我们常说的 bug 管理。

以上几种管理系统，在实际项目中，可以说是缺一不可。如今企业服务的市场越来越大，老牌和新生的团队协作工具越来越丰富，以下是一些常见的主打任务管理的团队协作工具，如图 4-9 所示。



图 4-9 常见的团队协作管理工具

**提示：**以上几种管理工具，很多都集成了除任务管理外的其他功能，用户可以根据实际的需求自行对比选择。

## 4.2.2 代码托管

代码托管工具的选择，需要具体问题具体分析。代码托管跟文件管理非常类似，根据团队需求和项目类型，有以下常见的两种选择：

- 在线代码托管服务平台。
- 局域网代码托管服务平台。

这里以 Git 作为基础的版本管理工具，主要推荐两个在线代码托管服务平台：

- GitHub
- Bitbucket

GitHub 免费提供的仓库都是公共的，私人仓库需要每月付费。与 GitHub 不同，Bitbucket 对免费用户只提供无限的私有仓库。所以，如果项目是开源的，可以公开访问，GitHub 是首选。若需要尽可能低成本地获取私有仓库，可以试一试 Bitbucket。

GitHub 与 Bitbucket 的标志，如图 4-10 所示。



图 4-10 主流的在线代码托管服务平台

由于使用在线服务平台，并不能满足所有的团队需求，当团队人员快速增长、项目规模不断扩张后，在线服务平台终究会因为网络访问速度、代码仓库容量大小限制等，降低使用体验。除此之外，有的企业的核心代码，对保密性要求很高，直接就禁止外网访问，遇到这种情况，就需要自行搭建私有服务，实现局域网内的代码托管。

私有的开源代码托管工具很多，这里推荐使用 GitLab，其 Logo 如图 4-11 所示。



图 4-11 GitLab 的 Logo（有点儿像一个狐狸头）

GitLab 是一个使用 Ruby on Rails 语言开发的开源应用程序，实现一个自托管的 Git 项目仓库，可通过 Web 界面访问公开或者私人的项目。

它拥有与 GitHub 类似的功能，能够浏览源代码、管理缺陷和注释，可以管理团队对仓库的访问，非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序（Wall）进行交流，它还提供了一个代码片段收集功能可以轻松实现代码复用，便于日后有需要时进行查找。

在 GitLab 中创建私有项目，如图 4-12 所示。



项目路径  项目名称

希望将几个相关联的项目放置于同一个命名空间下？ [创建群组](#)

项目描述 (可选)

可见等级

- 私有  
项目访问权限必须明确授权给每个用户。
- 内部  
该项目允许已登录的用户访问。
- 公开  
该项目允许任何人访问。

图 4-12 在 GitLab 上创建私有项目

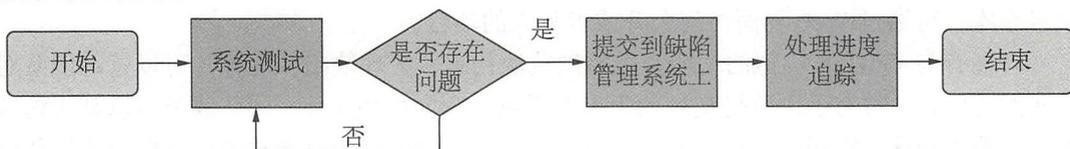
 提示：说 GitLab 就是一个局域网特色版的 GitHub，也不足为过。不过在使用和设计理念上，两者还是有很多的不同。

### 4.2.3 缺陷管理

缺陷管理其实就是我们常说的 bug 管理，在整个项目周期中，测试人员都会把测试问题反馈到系统上，并根据开发人员对问题解决的进度进行追踪。开发过程中，常见的是“提交 bug，修改 bug”过程，如图 4-13 所示。

缺陷管理系统主要的目的就是保障项目的质量，防止因为重大缺陷（bug）导致的系统故障和不必要的损失。目前除了专门的缺陷管理系统，很多任务管理系统都自带缺陷管理模块，甚至 GitLab 也有“问题”模块对应缺陷管理。如果对系统功能要求不高，可以直接用 GitLab 当作 bug 的管理系统，结合代码仓库，以提高问题修复的效率，如图 4-14 所示。

### 测试人员反馈问题



### 开发人员解决问题

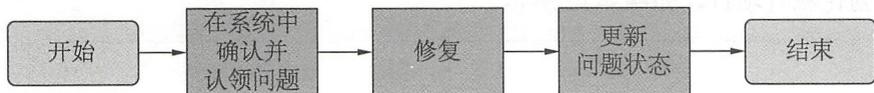


图 4-13 问题提交与修复流程



图 4-14 在 GitLab 中进行问题的反馈与追踪

## 4.3 在线协作绘制流程图——ProcessOn

开发团队在进行实际开发前，通常要整理系统逻辑与开发思路，这少不了各种流程图和思维导图，这些绘制图也是各种文档中的核心部分。绘制一个通俗易懂、简约好看的流程图，也是开发者必备的实用技能。

### 4.3.1 ProcessOn 简介

为了便于理解，好的文档都会配上丰富的图例，如流程图、UML 用例图和思维导图等。开发者绘制通俗易懂的图例，需要选择合适的绘图工具，这里推荐一个好用的网页流程图工具——ProcessOn，使用该工具不仅可以绘制常见的流程图，而且官方还提供了很多其他的图片格式：

- 思维导图；
- 原型图；
- UML；
- 网络拓扑图；
- 组织结构图。

这些图的格式类型，基本上可以满足一般用户的需求。除了这些基本的格式外，ProcessOn 还提供了模板市场（分为免费模板和收费模板），好的模板，可以一键复制到自己的文件夹内，方便用户灵活选择。ProcessOn 的管理界面如图 4-15 所示。

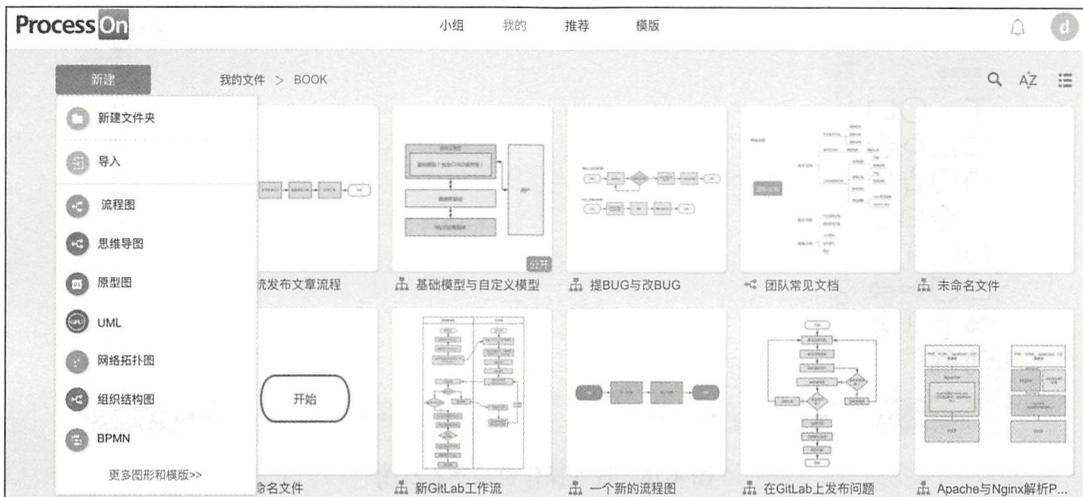


图 4-15 使用 ProcessOn 创建流程图

总的来说，使用 ProcessOn 的优点如下：

- 无需安装，直接使用；
- 可以协同开发；
- 收费版价格相对较低；
- 模板市场丰富。

**提示：**若没有特别说明，本书中大部分的思维导图、流程图等都是使用 ProcessOn 绘制的。

### 4.3.2 ProcessOn 操作指南

这里以绘制流程图类型为例，讲解如何快速使用 ProcessOn 绘图工具，具体操作步骤如下。

### (1) 注册账号

账号注册没有门槛，访问地址：<https://processon.com/signup>，即可快速注册，免费用户可以创建的模板数量有限，若平时经常绘制各种图形，建议升级到付费版本。

### (2) 创建流程图空模板

在用户管理首页，单击“新建”命令后出现模板类型列表，选择“流程图”后，进入图例的操作台，如图 4-16 所示。

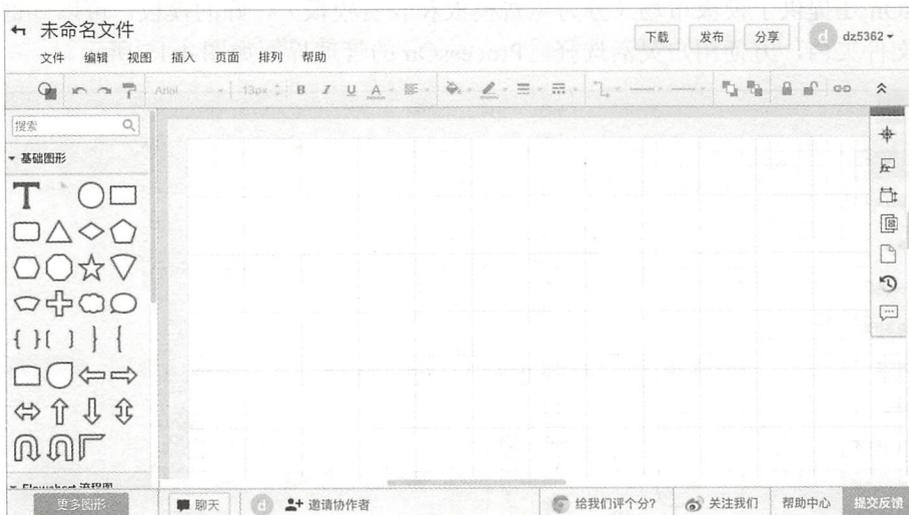


图 4-16 进入操作台绘制流程图

执行完以上操作，一个空的模板就创建完毕了，使用者熟悉一下相应的功能，就可以绘制所需要的流程图了。

### (3) 新建图形

在操作台上的左侧“基础图形”中，任何图形都可以直接拖拽（选择图形后按住鼠标左键不放）到画布上。例如，先选择“Flowchart 流程图”，拖拽“开始/结束”图形到画布，用鼠标双击图形则可以输入名称，例如，输入“开始”文本，如图 4-17 所示。

选中图形后，可以发现顶部的属性栏可以操作了，当前图形的属性都可以自定义，其中包含图形的大小、线条和背景颜色、内置字体大小、所在位置等属性，如图 4-18 所示。



图 4-17 开始绘制流程图

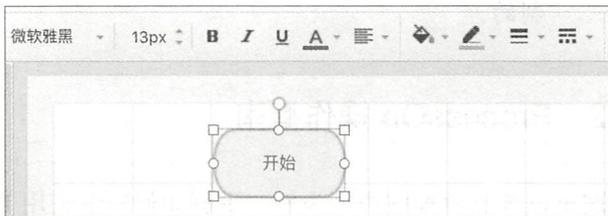


图 4-18 修改图形的颜色、字体和大小

#### (4) 新建图形连接关系

在图形上有四个圆点，鼠标单击后向外拖拽，可以选择创建流程指向的下一个图形。选中图形间的连线，可以灵活地自定义该连线的类型，如图 4-19 所示。

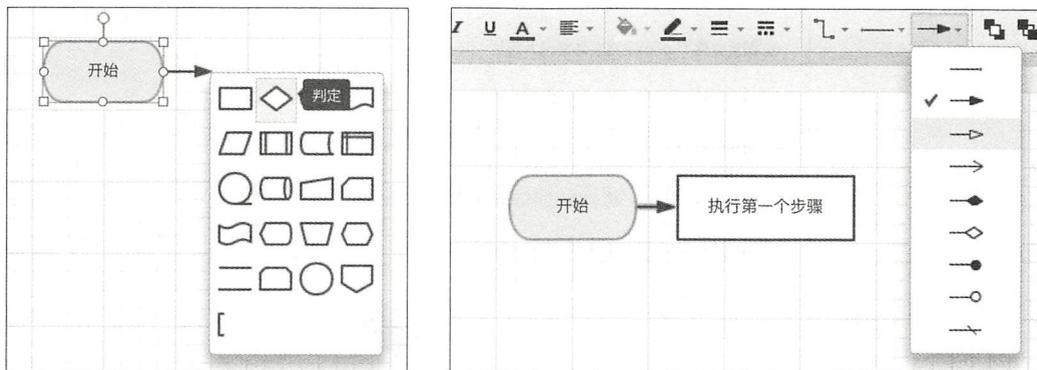


图 4-19 自定义图形连线的属性

#### (5) 快速复制已有图形

选中任意图形或连线，单击鼠标右键，可以弹出快捷菜单，方便进行复制等操作，如图 4-20 所示。

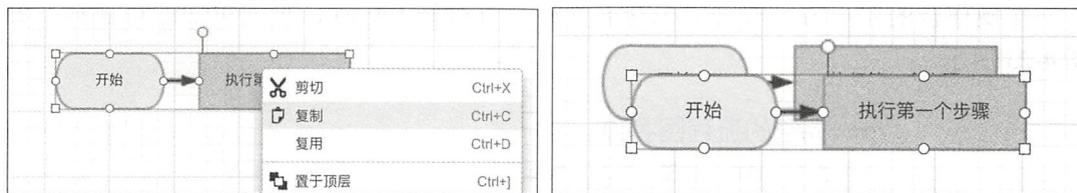


图 4-20 快速复制已有图形

#### (6) 完成流程图绘制

选中复制出的副本图形，手动修改其所在位置，完成流程图的绘制，如图 4-21 所示。

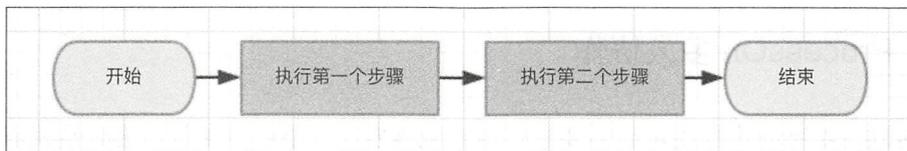


图 4-21 完成流程图的绘制

#### (7) 保存流程图到本地

单击操作台右上角的“下载”按钮，在弹出的菜单中，选择需要保存的文件格式，然

后单击“确定”按钮后即可自动下载，如图 4-22 所示。

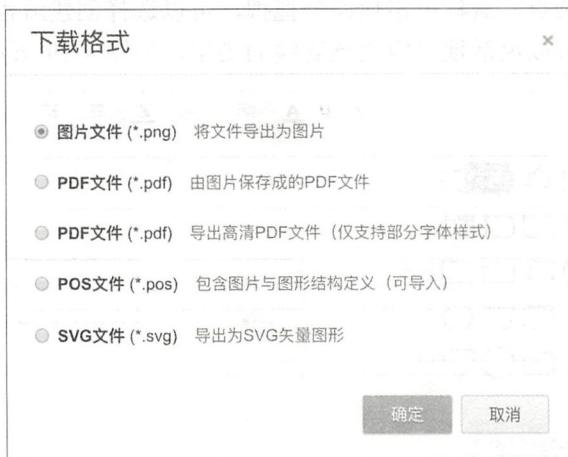


图 4-22 可供选择的文件保存格式

**提示：**保存高清格式图片，建议选择“PDF 文件（高清）”和“SVG 文件”。

#### (8) 在线保存流程图源文件

在操作台的左上方，双击“未命名文件”可以修改标题，再次单击即可保存，如图 4-23 所示。

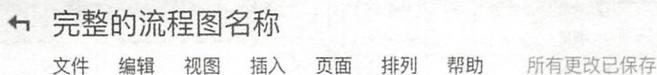


图 4-23 ProcessOn 会自动保存修改

返回到用户管理中心，发现文件已经被保存成功。

**提示：**每一个步骤都会自动保存，所以即使关闭了浏览器数据还是会保存在线上。

### 4.3.3 ProcessOn 多人协作

有时候，绘制的流程图可能需要进行用户版本发布、小组讨论等，这时候就需要更多的人参与其中，实现文件的评审和发布等。ProcessOn 多人协作的模块被称为小组，使用流程如下。

#### (1) 创建小组

首先在用户管理中心，单击顶部导航栏菜单中的“小组”按钮，随后创建一个新的小组名“我的团队”，如图 4-24 所示。



图 4-24 在 ProcessOn 中新建小组

### (2) 小组成员邀请

进入新创建的小组，单击界面右上方的“小组设置”按钮（齿轮图标），可以进行小组成员的邀请等操作，如图 4-25 所示。



图 4-25 通过邀请的方式添加小组成员

### (3) 小组成员权限分配

通过邮箱的方式邀请小组内的成员，当组员通过之后，默认权限为“订阅者”，在此权限下，只能查看文件，不能修改文件。若想提升小组成员的权限，需要小组创建者在“小组成员”列表中，选中某个成员进行权限的分配，如图 4-26 所示。



图 4-26 小组成员默认拥有“订阅者”的权限

#### (4) 协同操作

在分配权限时，除了“订阅者”之外的角色都可以对图形模板进行修改，此时就可以方便地进行多人协同操作了。

ProcessOn 本身的功能十分强大，除了不用安装本地客户端外，全程在线自动保存也防止了文件的丢失。另外，本节中的实例只是 Processon 的基础入门实例，ProcessOn 本身除了可以绘制流程图外，对其他图形的支持也很好，开发者可以根据自己的需求灵活选择，以实现最高的开发效率。

**提示：**虽然 ProcessOn 文件都存储在线上，但建议还是多备份文件到本地，以防止不必要的损失。

## 4.4 GitLab 操作全攻略

集成了代码管理、缺陷管理、任务管理和自动部署等功能的 GitLab，是团队协作首选的协作工具。不过 GitLab 相比其他的任务管理工具，在安装和使用上需要使用者有一定的开发经验，因为其中对于命令行、脚本的使用几乎无处不在，所以由开发者来部署 GitLab 更为合适。本节重点讲解 GitLab 在使用上的小技巧，让使用者少走弯路。

### 4.4.1 安装与汉化

编译安装 GitLab 所需的依赖较多，过程也较为烦琐，为了简化流程，这里使用 Ubuntu 内置的 apt 包管理器来安装 GitLab，安装完成后再进行汉化操作。

## 1. GitLab 安装

这里还是在 Windows 环境下，搭建虚拟机来完成 GitLab 的安装调试。

### (1) 配置虚拟机

使用 Vagrant 配置一个 Ubuntu16.04 的虚拟机环境，并且设置私有 IP 地址为：192.168.33.33，具体的安装步骤在这里不再赘述。

### (2) 安装依赖包

执行以下命令：

```
sudo apt-get install curl openssh-server ca-certificates postfix
```

执行过程中，会提示选择邮件的配置，默认选择 Internet Site，按回车键即可，如图 4-27 所示。

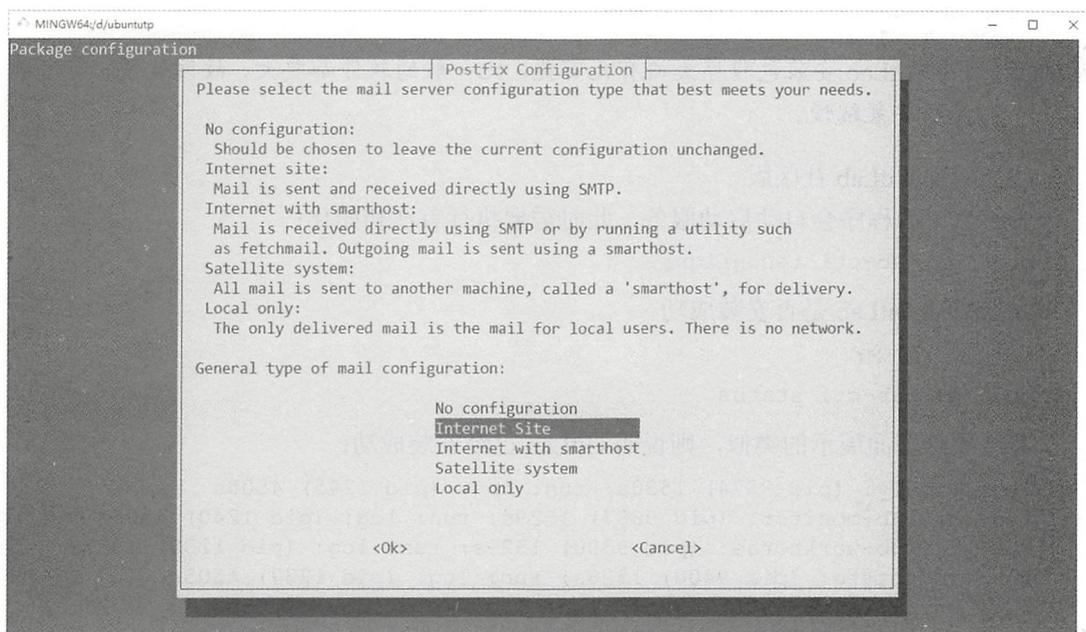


图 4-27 安装依赖时选择邮件的配置

### (3) 修改 apt 包安装工具的源

为了快速安装，这里使用清华大学开源软件站的 Gitlab Community Edition 镜像进行安装。首先执行以下命令信任 GitLab 的 GPG 公钥：

```
curl https://packages.gitlab.com/gpg.key 2> /dev/null | sudo apt-key add - &>/dev/null
```

随后执行以下命令：

```
sudo vim /etc/apt/sources.list.d/gitlab-ce.list
```

在文件中增加以下内容：

```
deb https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/ubuntu xenial main
```

 **提示：**GnuPG（简称 GPG）是一种可以加密、签名数据和信息传递的技术，包含一个通用的密钥管理系统，其访问模块可以访问各种公钥目录。

#### （4）安装 GitLab 社区版

GitLab 虽然开源免费，但是也分为社区版和企业版，企业版需要订阅收费，不过一般团队使用社区版就可以完全满足团队的需求。

更新了 apt 的源信息后，需要先执行更新操作：

```
sudo apt-get update
```

随后执行以下命令安装 GitLab：

```
sudo apt-get install gitlab-ce
```

 **提示：**因为 GitLab 安装包程序大概有几百兆，比一般的软件都要大，故网速慢，有可能导致安装缓慢。

#### （5）配置 GitLab 社区版

安装完毕后程序会自动启动服务，此时需要执行首次初始化：

```
sudo gitlab-ctl reconfigure
```

#### （6）验证 GitLab 是否安装成功

执行以下命令：

```
sudo gitlab-ctl status
```

若结果跟下面展示的类型，则说明 GitLab 已经安装成功：

```
run: gitaly: (pid 9374) 1530s; run: log: (pid 1245) 4505s
run: gitlab-monitor: (pid 9387) 1529s; run: log: (pid 1240) 4505s
run: gitlab-workhorse: (pid 9390) 1529s; run: log: (pid 1238) 4505s
run: logrotate: (pid 9400) 1528s; run: log: (pid 1237) 4505s
run: nginx: (pid 9406) 1528s; run: log: (pid 1253) 4505s
run: node-exporter: (pid 9449) 1528s; run: log: (pid 1246) 4505s
run: postgres-exporter: (pid 9456) 1527s; run: log: (pid 1241) 4505s
run: postgresql: (pid 9462) 1527s; run: log: (pid 1252) 4505s
run: prometheus: (pid 9470) 1526s; run: log: (pid 1239) 4505s
run: redis: (pid 9480) 1526s; run: log: (pid 1250) 4505s
run: redis-exporter: (pid 9484) 1525s; run: log: (pid 1258) 4505s
run: sidekiq: (pid 9490) 1525s; run: log: (pid 1254) 4505s
run: unicorn: (pid 9497) 1525s; run: log: (pid 1236) 4505s
```

#### （7）初始化管理员密码并登录

GitLab 默认的超级管理员账号为 root，首次登录需要设置密码，设置完成后，就可以登录系统进行相关的操作了。在浏览器中访问地址 <http://192.168.33.33> 后，就可以看到 GitLab 的首页，第一次登录系统会自动提示修改 root 账号的密码，如图 4-28 所示。

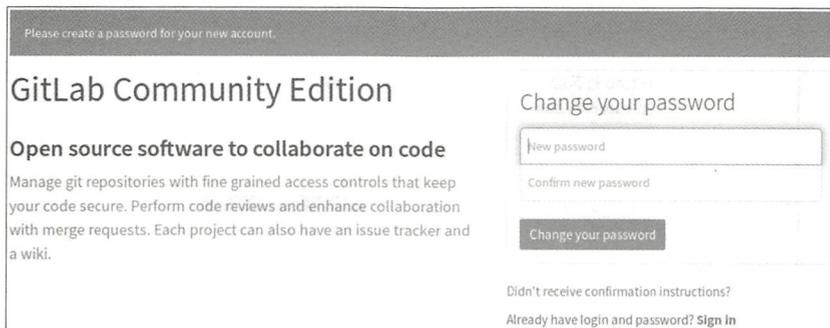


图 4-28 安装 GitLab 完成后访问首页

## 2. GitLab 汉化

安装成功后，GitLab 默认显示语言为英文，虽然使用上没有问题，但中文可以提升用户使用的体验，这里建议汉化。汉化的步骤如下。

(1) 查看 GitLab 当前版本号，执行以下命令：

```
sudo cat /opt/gitlab/embedded/service/gitlab-rails/VERSION
```

执行完毕后，可以看出当前版本号为 9.5.4。

(2) 克隆汉化代码的版本库，导出对比文件。

在任意目录下，克隆汉化代码的版本库，命令如下：

```
git clone https://gitlab.com/xhang/gitlab.git
```

进入版本库，比较汉化标签和原标签，导出 patch 命令用的 diff 文件，命令如下：

```
git diff v9.5.4 v9.5.4-zh > ../9.5.4-zh.diff
```

 **提示：** git diff 命令中的版本号，需要和用户本地安装的 GitLab 工具的版本号保持一致。

(3) 导入汉化补丁。

导入前需要先停止 GitLab 的服务，执行以下命令：

```
sudo gitlab-ctl stop
```

进入到补丁所在的目录后，执行 patch 命令导入：

```
sudo patch -d /opt/gitlab/embedded/service/gitlab-rails -p1 < 9.5.5-zh.diff
```

执行完毕后，再次启动 GitLab 服务：

```
sudo gitlab-ctl start
```

再次执行初始化配置命令：

```
sudo gitlab-ctl reconfigure
```

完成后，在浏览器中访问地址：<http://192.168.33.33>，可以看到 GitLab 已经汉化成功，如图 4-29 所示。



图 4-29 汉化后的 GitLab 易用性显著提高

**提示：**GitLab 版本迭代较快，开源社区对它的汉化工作并没有完全同步更新，有些新功能和结构变更可能没有汉化彻底。

## 4.4.2 了解 GitLab 的工作流

在团队协作时，必须有一个相对固定和规范的工作方法，这样团队成员才能有序、高效地工作，项目也可以井井有条地进行下去，而这个规范的工作方法，就是常说的工作流。

因为 GitLab 本身是以 Git 作为版本管理工具，所以 GitLab 工作流本身结合了 Git 工作流和 GitHub 工作流的特性。为此，我们先了解下什么是 Git 工作流和 GitHub 工作流。

### 1. Git工作流

为了方便理解，我们回顾在 3.4 节中讲过的实例：开发者需要开发一个新功能，而在 master 分支上新建了 dev 分支（新功能测试和开发分支），在新功能完成需要上线前，把 dev 分支合并回 master 分支上。

这其实就描绘了简单的 Git 工作流的基本特性：

- 至少有 master 和 dev 分支。master 分支对外进行版本发布，任何时候都是一个稳定的版本。dev 分支是开发分支，开发者所有的新功能都会提交到此分支上，版本发布时，dev 分支会被合并到 master 分支，项目需要长期维护两个分支。
- 新的功能或者 bug，会有独立的分支进行开发，完成后合并到 master 或者 dev 分支上。在 Git 工作流中，虽然每个分支都各司其职，但是分支过多也提高了项目开发的复杂度。

### 2. GitHub工作流

在 GitHub 定义的工作流规范中，只有 master 分支，不区分新功能分支或修复分支，其他分支在需要合并时，开发者需要提交一个名为 pull\_request 的推送请求，该推送请求其他开发者都可以看到，随后会对推送分支的功能和代码一起评审。当提交的推送请求被接受后，分支会被合并到 master 分支上，原分支会被删除。

GitHub 工作流如图 4-30 所示。

GitHub 工作流有以下几个特点：

- 只有一个 master 分支，关联自动化的版本发布和在线部署。
- 开发人员合并分支到 master 之前需要发起 pull\_request 推送请求。
- 其他开发者对推送请求进行审核。
- 推送请求一旦通过，会立即合并到 master 分支上，并马上部署到线上。

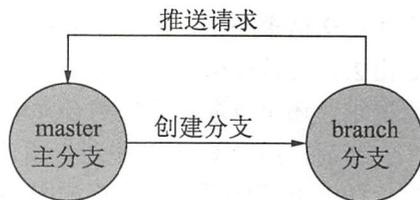


图 4-30 GitHub 工作流示意图

在这样的规范下，不需要维护很多各式各样的分支，相比 Git 工作流，在操作和管理上更加简单。但问题在于，因为只有一个 master 分支，不足以满足所有的需求，当不能同步到线上时，则需要新建分支专门管理。

### 3. GitLab 工作流

GitLab 工作流在实现时，结合了 Git 与 GitHub 的优点，只设立一个 master 分支，但是所有的其他分支的代码变动，都需要与 master 分支保持同步。

例如，为了发布版本，在 master 分支上创建 develop 分支，用来进行代码上线前的测试与预览，再从 develop 分支上创建 product 分支，对应上线版本。此时若出现 bug 问题需要修改，则需要先在 master 分支上修复，再向 develop 分支上合并，测试没问题后，再合并到 product 分支正式上线，这样可以保证每个分支的代码都已经修复了这个问题。

新功能的开发与提交，也与修改 bug 类似，自上而下从 master 分支向下合并，如图 4-31 所示。

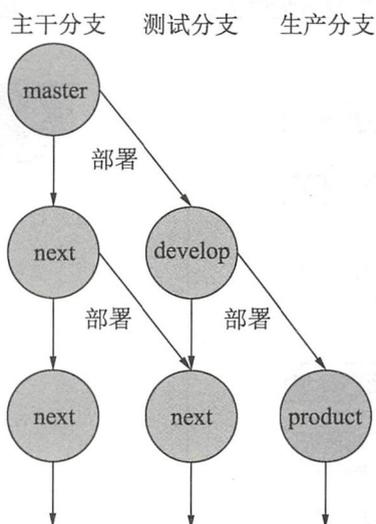


图 4-31 GitLab 工作流示意图

其次，GitLab 中对于代码的提交存在 Merge Request 概念，即合并请求，这和 GitLab

上的推送请求类似，开发者在分支上的一次或者多次的提交，可以合并为一个合并请求发送出去，当具有相应权限的开发者审核通过后，才可以合并到指定的分支上，若代码不符合要求，还可以取消当次的合并请求。

最后，结合 GitLab 中对于用户的角色设定，来看一个完整的 GitLab 工作流是什么样的。GitLab 代码开发工作流如图 4-32 所示。

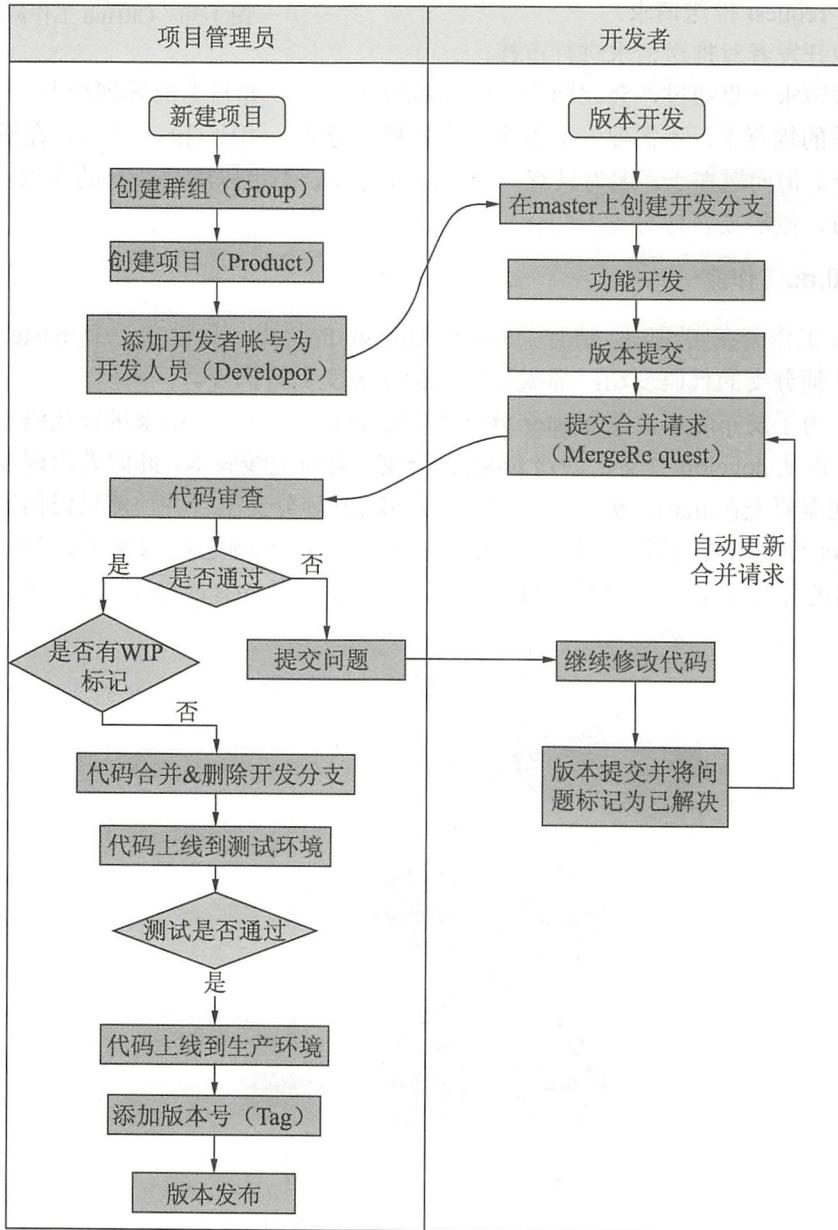


图 4-32 GitLab 代码开发工作流完整视图

### 4.4.3 GitLab 用户和项目管理

本节根据实际工作需求，将讲解核心的 GitLab 管理操作，其中不仅包含群组与项目的创建，还有人员角色的分配与管理。学完本节内容后，开发者可以更好地在项目开发中实践 GitLab 工作流。

#### 1. 创建用户

##### (1) 创建管理员用户

在 GitLab 中，开发者可以自行注册账号，也可以让管理员在后台进行创建。每个 GitLab 都会有一个超级管理员账号（如本实例中的 root 账号），可以创建不同角色的用户账号。GitLab 功能庞大，为了更加方便地定位到操作功能界面，在非必要的情况下，书中实例将会统一使用 URL 地址进行访问。

使用 root 账号登录 GitLab 后，创建一个管理员级别的账号，访问地址 <http://192.168.0.238/admin/users/new>，填写基本信息后，完成账号的创建。需要注意的是，除了系统提示必填的用户名等，选择用户权限为管理员，如图 4-33 所示。

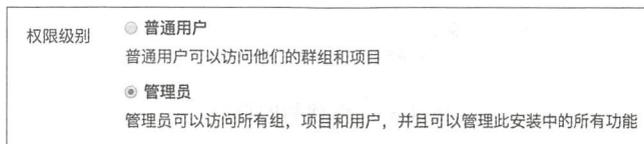


图 4-33 选择权限级别为管理员

账号名等信息可以根据需求填写，本实例账号名为：wangjialinadmin。

提示：访问地址中的 IP 需要换成实际的 GitLab 服务地址。

##### (2) 创建普通用户

为了方便演示，在 GitLab 注册界面，手动注册两个普通用户。注册完毕后，使用管理员账号（wangjialinadmin）登录，然后查看用户列表，如图 4-34 所示。



图 4-34 查看新注册的普通用户

#### 2. 创建群组和项目

在实际的项目中，如果遇到开发者需要同时开发、维护多个项目时，则需要先创建群

组，群组实际上就是多个项目的集合。创建群组，对开发者而言，能更加方便地访问项目，对于管理者而言，简化了多个项目的权限、操作管理。

群组和项目的关系密不可分，一般来说，开发者会根据实际需求创建对应权限的群组，随后再添加项目。具体步骤如下。

### (1) 创建群组

使用管理员账号 (wangjialinadmin) 登录后，访问以下地址 <http://192.168.0.238/groups/new>，创建新群组，为了体现角色权限的用处，选择可见等级为 Private (私有)，当需要时才把用户添加到当前群组。群组创建界面，如图 4-35 所示。

图 4-35 创建新的群组

### (2) 添加成员

完成以上步骤后，系统会自动重定向到群组管理界面，此时，可以把此前创建的两个用户添加到群组中去。在成员列表上，可以对用户进行权限的分配，这两个用户都设置为“开发人员”的权限，如图 4-36 所示。



图 4-36 给群组添加开发人员角色用户

提示：GitLab 在添加成员时，根据用户输入的关键词可以自动检索对应的用户，十分方便。

### (3) 创建项目

在 GitLab 顶部导航上单击“+”号，就会出现创建项目菜单，随后选择“新建项目”命令为当前群组创建项目，如图 4-37 所示。



图 4-37 在群组中创建项目

提示：项目的创建与群组没有必然关系，完全可以创建独立的项目进行管理。

## 3. 项目管理

项目创建完毕后，系统会自动重定向到项目的主界面，在这里可以对项目进行各种操作。在项目管理界面的左侧，可以看到其功能菜单，包含以下核心功能：

- 概览，在这里可以看到项目代码仓库的地址以及一些项目的基本信息。
- 版本库，包含完整的 Git 操作，如文件查看、提交记录和合并操作等。
- 问题，项目版本问题管理功能。

- 合并请求，可以处理开发者提交的合并请求。
  - CI/CD，持续集成、交付和部署管理，自动化构建管理代码的提交、测试和构建。
- 在这里重点看下问题管理功能和合并请求这两个功能模块的使用。

### (1) 问题管理功能

项目开发难免出现问题（bug），测试人员完全可以使用 GitLab 自带的问题模块，来实现缺陷管理。在 GitLab 的项目中提交问题的一般流程如图 4-38 所示。

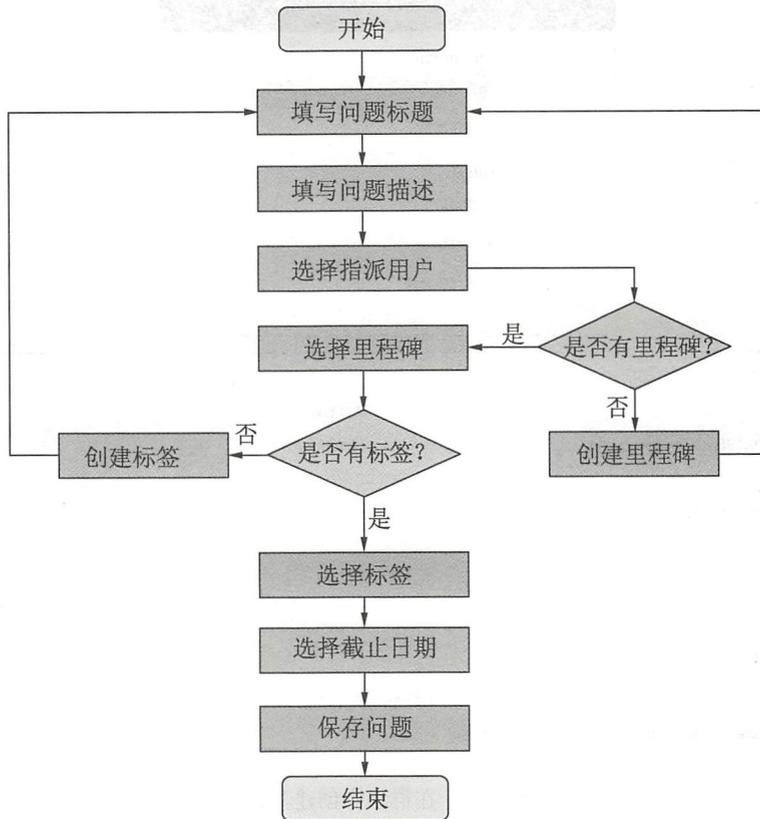


图 4-38 在 GitLab 的项目中提交问题的流程

比如举个例子：马上要到十一假期了，工程师王大力的在线商城的支付模块出了问题（无法完成支付），若商家在搞活动时发现用户无法支付，后果不堪设想，这时候测试人员就要提交问题了，提交的问题需要满足以下条件：

- 需要解决问题的优先级为最高。
- 十月一日前务必要解决问题。
- 指派问题给开发人员王大力。

这样新建一个问题，就需要设置指派人员、截止日期和问题的优先级这几个必选项，如图 4-39 所示。



图 4-39 在 GitLab 中新建问题

在编辑问题时，除了可以方便地指派人员和时间等，问题的描述可以使用 Markdown 文本格式编写，问题图片也可以直接拖拽到富文本编辑框中进行上传，文本格式可以进行预览。同时一个问题可以增加多个标记，每个标记还可以设置不同的优先级，以便区分每个问题的状态。进入标记管理后，列表如图 4-40 所示。

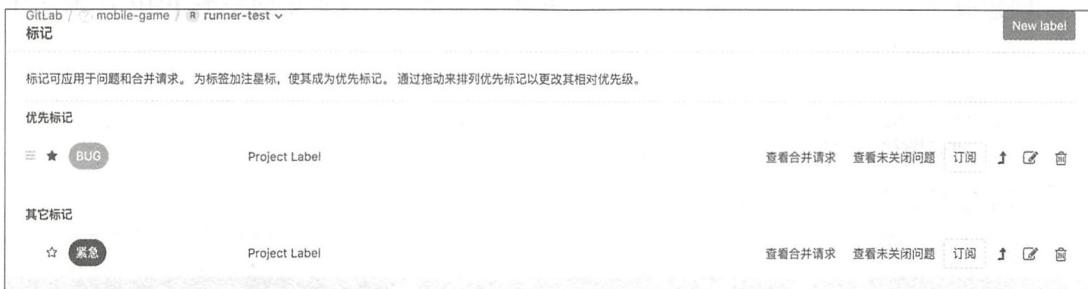


图 4-40 管理标记的优先级

新建问题完毕后，在问题列表中不仅可以看到详细的信息，还可以追加评论，评论时可以“@”某个开发者，方便其了解最新的动态，如图 4-41 所示。

## (2) 开发者提交合并请求

在 GitLab 工作流中，合并请求是不可缺少的一步，在实际的团队管理中，应当设置专门的管理员角色，对开发者提交的合并请求代码进行审核管理。

这里还是以王大力的在线商城无法支付的问题为例，来看一下合并请求模块的使用。

首先使用王大力的账号登录 GitLab，在项目的问题列表中找到此问题，在详情页中，可以创建一个名为问题 ID 的修复分支，操作结果如图 4-42 所示。



图 4-41 在问题下进行评论操作

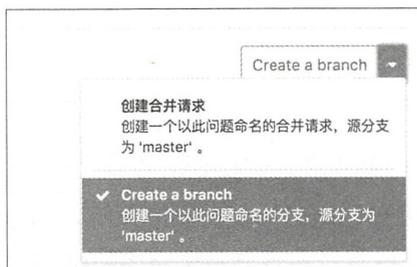


图 4-42 在问题下创建一个修复分支

创建成功后，进入到此修复分支，为了模拟合并请求，这里新增一个 PHP 脚本文件 pay.php，内容如下：

```
<?php
header('Content-type:text/html;charset=utf-8');
echo "问题修复时间:".date('Y-m-d H:i:s');
```

提交以后，效果如图 4-43 所示。

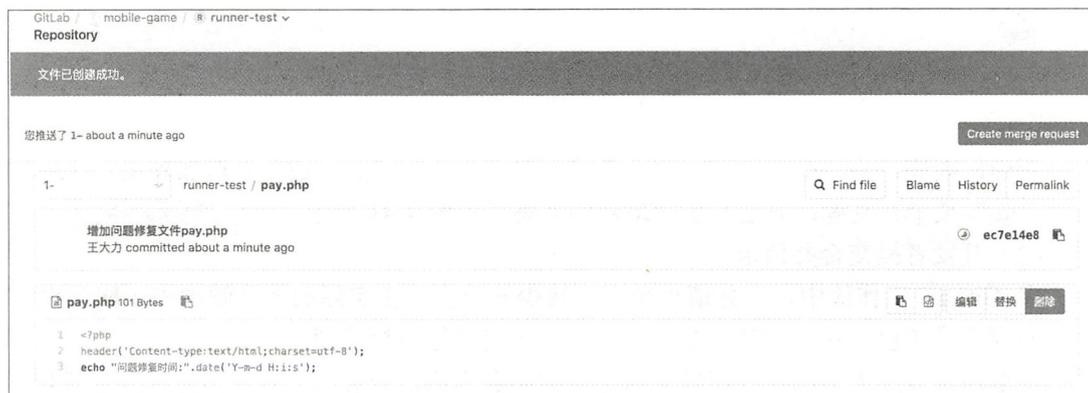


图 4-43 在问题分支上修复问题

当开发者确认问题已经修复完成（本地测试）后，就可以提交合并请求了，单击 **Create merge request** 按钮，进入到合并确认页面。一般来说，这里只需要指定审核人员后提交即可，另外可以勾选接收请求后删除来源分支，以减少版本库中无用分支的数量，如图 4-44 所示。

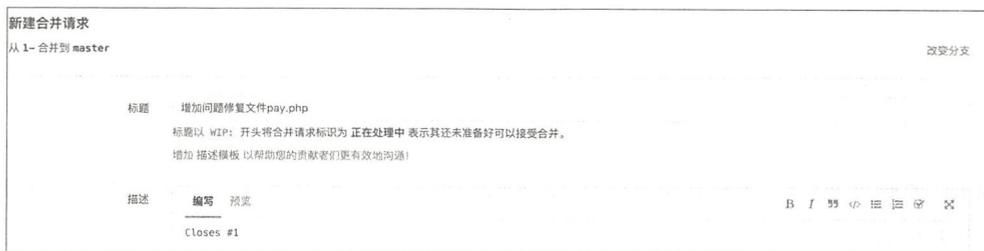


图 4-44 开发者提交合并请求

### (3) 管理者审核合并请求

此时工程师王大力的第一步工作已经完成，使用管理员账号登录后，查看其提交的合并请求，详情页如图 4-45 所示。



图 4-45 查看开发者提交的合并请求

同时可以方便地在下方看到提交记录和变更后对比的详细修改，如图 4-46 所示。

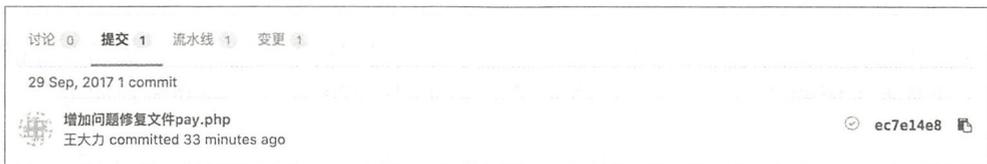


图 4-46 查看合并请求的提交记录 1

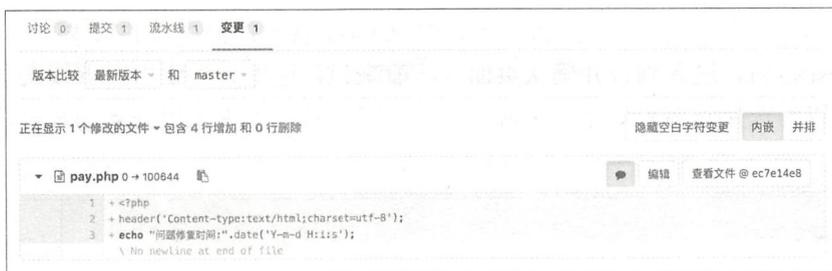


图 4-46 查看合并请求的提交记录 2

随后需要把当前分支检出到本地，审查没问题后，再决定是否执行合并操作，操作流程如图 4-47 所示。



图 4-47 检出分支并在本地审查和合并

确保无误后，单击 Merge 按钮，分支就被合并到 master 主干分支了，结果如图 4-48 所示。

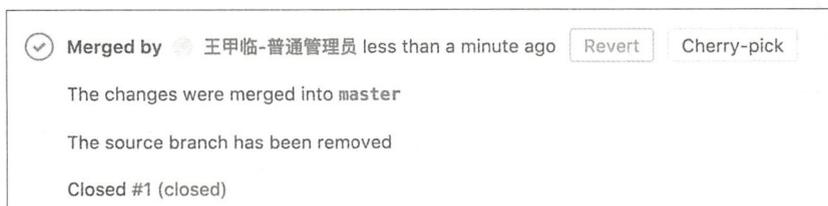


图 4-48 成功合并代码并删除源分支

以上就是管理员执行合并请求的全部流程，若代码审核不通过，可以关闭合并请求，并告知开发者关闭的理由，开发者根据反馈的意见进行新的修改、提交流程即可。

**提示：**本实例中的演示，都是在线进行的 Git 相关操作，如代码提交等，而在本地创建副本的过程跟一般的 Git 操作完全一致。



## 4.4.4 GitLab 持续集成与自动构建实践

当团队人数众多，每天都有大量的合并请求检查时，管理员难免会忙不过来。此时就需要一些自动化的脚本程序，在代码提交时，进行一些检查和操作，这样就可以省去大量的人工审核时间，提高系统使用的效率。

### 1. 持续集成、持续交付和持续部署原理

在使用 GitLab 进行项目管理时，发现在项目管理菜单中有一个 CI/CD 的功能。简单来说，持续集成指的就是频繁地将代码提交集成到主干上，但是在每次集成之前，都需要通过自动化的测试，任何测试用例的失败，都会导致不能集成。

使用持续集成的优点主要有以下两个：

- 快速发现问题，减少人工审核的成本。
- 防止本地分支代码大幅度偏离主干代码。

还有另外两个概念与持续集成紧密相关，分别是持续交付和持续部署。其实持续交付指的是可以自动的、频繁地把项目自动交付给客户或者质量团队，可以看成是持续集成的下一步，在交付的过程中，任何的测试、评审过程都是完全自动的。

持续部署又可以看成是持续交付的下一步，当交付的内容通过评审后，会自动部署到生产环境中。这里简单列出一个从代码提交到上线部署的流程，完全实现了持续集成、交付与部署，如图 4-49 所示。

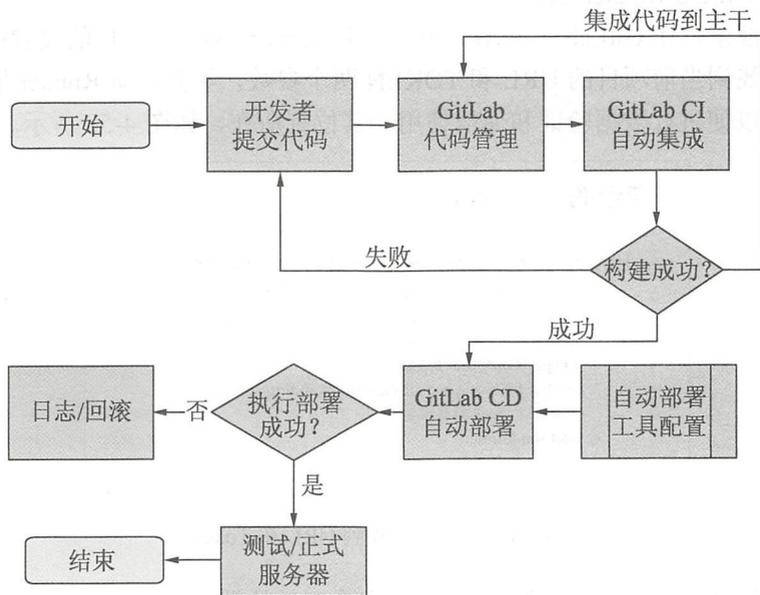


图 4-49 GitLab 持续集成、交付和部署流程示意图



## 2. GitLab-CI与GitLab-Runner实践

在 GitLab 8.x 版本后，系统已经集成了 CI 组件，不过还需要在服务器上安装 GitLab-Runner，以实现自动构建、自动部署等。接下来看看如何配置持续集成。

在实际使用中，GitLab-CI 的作用是配置需要执行的自动化构建任务脚本，当每次开发者提交代码时，GitLab-Runner 会自动获取 GitLab-CI 预定义的任务内容，并随后运行。Runner 服务可以搭建在当前的服务器上，不过为了不影响 GitLab 的使用，建议搭建在独立的服务器上运行。整个安装流程如下。

### (1) 搭建 Vagrant 虚拟机

在 GitLab 虚拟机外，再搭建一台 Ubuntu16.04 的服务器，相关流程可以参考之前的教程，操作过程不再赘述。

### (2) 使用 apt 工具安装 GitLab-Runner

连接到服务器后，执行以下命令，添加 GitLab 相关的源：

```
# For Debian/Ubuntu
curl https://packages.gitlab.com/install/repositories/runner/gitlab-ci-multi-runner/script.deb.sh | sudo bash
```

随后执行如下安装命令：

```
# For Debian/Ubuntu
sudo apt-get install gitlab-ci-multi-runner
```

命令执行完成后，安装完毕。

### (3) 注册 GitLab-Runner 服务

为了方便演示，在 GitLab 上新增一个项目名 runner-test。在项目的设置中，找到流水线界面，可以找到当前项目的 URL 和 TOKEN 两个参数，等会注册 Runner 服务时就需要这两个参数，以便进行数据验证和通信使用。其位置所在，如图 4-50 所示。

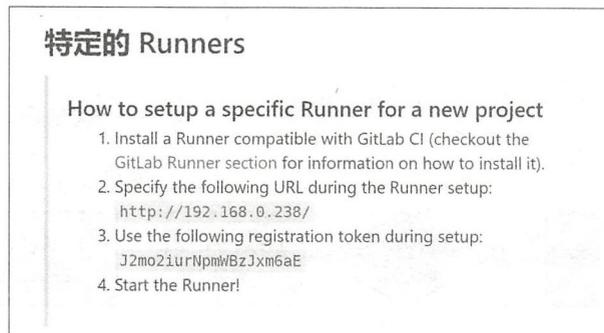


图 4-50 在项目中找到 URL 和 Token

连接到部署 Runner 的服务器，执行以下命令开始注册：

```
sudo gitlab-ci-multi-runner register
```



随后一步一步根据提示输入内容，执行过程如下：

```
Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
http://192.168.0.238/ # 输入GitLab的地址
Please enter the gitlab-ci token for this runner:
J2mo2iurNpmWBzJxm6aE # 输入项目的Token
Please enter the gitlab-ci description for this runner:
[ubuntu-xenial]: runner-test # 输入Runner的名称
Please enter the gitlab-ci tags for this runner (comma separated):
runner-test # 输入Runner的Tags, 可以用逗号分隔
Whether to run untagged builds [true/false]:
[false]: true # 是否允许未标记的构建, 输入true
Whether to lock Runner to current project [true/false]:
[false]: true # Runner是否只能用于当前项目, 输入true
Registering runner... succeeded runner=J2mo2iur
Please enter the executor: ssh, virtualbox, docker+machine, docker-ssh+
machine, docker, docker-ssh, shell, parallels, kubernetes:
shell # 构建执行的方式, 这里选择命令行 shell
Runner registered successfully. Feel free to start it, but if it's running
already the config should be automatically reloaded!
```

执行完成后，回到 GitLab，前往获取 URL 和 Token 的页面，刷新后会出现 Runner 的信息，走到这一步时，Runner 的安装和注册就完成了，如图 4-51 所示。

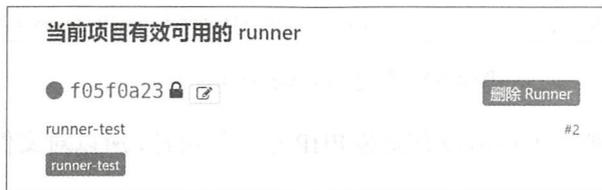


图 4-51 查看当前项目的 Runner

#### (4) 配置.gitlab-ci.yml文件

`gitlab-ci.yml` 是配置 CI 在项目中需要做哪些操作的文件，该文件位于仓库的根目录。当有新内容推送到仓库后，GitLab 会查找是否有 `.gitlab-ci.yml` 文件，若文件存在，Runner 将会根据该文件的内容开始自动构建本次的提交。`.gitlab-ci.yml` 文件存放在仓库里，所以这个文件是受版本控制的。

官方给出的 `.gitlab-ci.yml` 格式如下所示：

```
before_script:
  - apt-get update -qq && apt-get install -y -qq sqlite3 libsqlite3-dev
  nodejs
  - ruby -v
  - which ruby
  - gem install bundler --no-ri --no-rdoc
  - bundle install --jobs $(nproc) "${FLAGS[@]}"

rspec:
script:
```



```
- bundle exec rspec

rubocop:
script:
  - bundle exec rubocop
```

这段配置代码不难理解，其中定义了两个任务：`rspec` 和 `rubocop`，而在这两个任务执行前，会先执行 `before_script` 中定义的内容。

随后在 `runner-test` 项目中，添加新文件，可以选择模板，依次选中 `.gitlab-ci.yml` 和 `PHP`。系统会自动创建一个 `.gitlab-ci.yml` 模板文件，如图 4-52 所示。

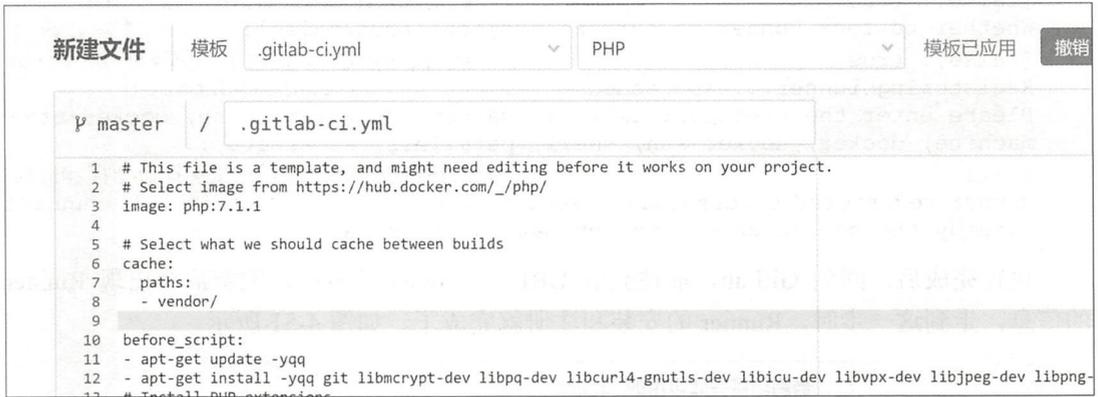


图 4-52 创建 `.gitlab-ci.yml` 模板文件

不过 `Vagrant` 虚拟机上目前没有安装 `PHP` 相关的组件，所以对文件内容进行简化修改如下：

```
before_script:
  - echo "脚本前执行了一个文本输出"

test:
  script:
    - echo "执行了一个任务"
```

提交后，`GitLab` 会自动检测配置文件是否正确，如图 4-53 所示。

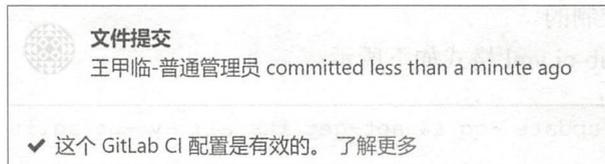


图 4-53 通过检测的 `GitLab CI` 配置文件

### (5) 查看自动构建结果

打开项目的流水线模块，看到两次提交构建记录，其中第一次构建失败了，而第二次构建成功。在流水线列表里面，包含了构建的提交号、是否成功和构建执行时长等信息，



如图 4-54 所示。

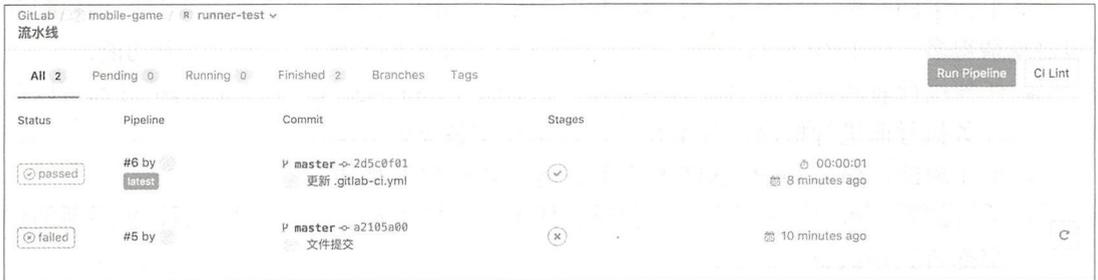


图 4-54 查看每次提交后自动构建的信息

单击列表上 Stages 下的状态按钮，可以在弹出对话框中看到各个任务的执行情况，如图 4-55 所示。

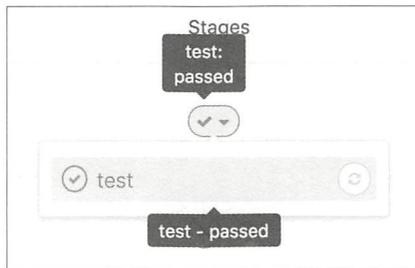


图 4-55 查看自动构建任务的执行状态

单击 test 任务按钮后，可以看到任务执行详细信息，如图 4-56 所示。



图 4-56 查看自动构建的结果



### (6) 更复杂的集成构建任务

在上面中的实例，只是执行了两个相对简单的 shell 命令，而在实际项目中构建配置却要复杂得多。以 PHP 项目为例，可以在脚本中定义不同的任务并实现以下功能：

- 任务执行前基础组件的检测和安装。如 Git、Composer 和一些基础的扩展都可以在任务执行前进行检测，若不存在可以执行安装或更新操作。
- 单元测试。编写 PHP UNIT 相关的任务，执行相关测试。
- 项目部署。例如，可以自定义相应的任务，当其他的任务检测通过后，把最新的提交部署到测试服务器上。

除了以上操作外，如果配置的灵活性很好，开发者可以根据需求自行搭配。



# 第 5 章 好用的 PHP 开发环境

## —— PHPStorm

开发者在编写程序时，为了提升编码的效率，除了需要符合编程语言开发的规范，还需要强大的编辑器工具的支持。本章除了讲解常见的编辑器工具之外，还会重点讲解 PHP 集成开发环境——PHPStorm 的常见技巧与进阶用法。

### 5.1 常用 PHP 源代码开发工具

本节重点讲解 PHP 开发过程中常见的源代码编辑器和 IDE（集成开发环境）工具。大多数的编辑器都支持多种语言的代码高亮、文本样式修改和自动完成等功能，更高级别的工具甚至提供代码提示、方法跳转等功能。

#### 5.1.1 源代码编辑工具简介

对于 PHP、JavaScript 和 HTML 等脚本语言来说，很多时候开发工作都是和字符串文本（源代码）打交道。如果不要开发编辑器带有代码调试和实时编译等功能，则 Windows 操作系统自带的记事本就可以完全满足编写要求。面对琳琅满目的开发工具，开发者需要根据自身需求进行合理选择。

常见的编程语言开发工具分为源代码编辑器和集成开发环境（IDE）两种。为了更好地理解这两者的功能和关系，先来看这两种工具的定义。

##### 1. 源代码编辑器

源代码编辑器是开发者用于编写计算机程序的文本编辑器。它通常是一个独立的应用程序，或是作为集成开发环境（IDE）的一部分存在，又或者是一个运行于浏览器中的网页编辑器。由于开发者的主要任务就是编写代码，因此源代码编辑器是最重要的编程工具。

合格的源码编辑器可以帮助开发者更快速地编写代码，所以一般的文本编辑器（例如 Windows 记事本）虽然可以用，但并不适合编程开发。

源代码编辑器至少要支持以下功能，才能满足快速开发的需求：



- 语法高亮;
- 自动缩进;
- 自动完成;
- 自动补全;
- 安装轻便。

只要满足以上几个功能的编辑器，都可以提高代码编辑的效率。此外，还需要考虑到该工具是否开源、是否收费、是否支持当前使用的操作系统平台等特性。

下面列举了一些比较著名的源代码编辑器。如没有特殊说明，它们都是跨几个主流操作系统平台的。

- Eclipse 内置编辑器;
- Emacs 宏编辑器;
- Geany 文件编辑器;
- Gedit 文本编辑器;
- IntelliJ IDEA 的内置编辑器;
- Microsoft Visual Studio 的内置编辑器;
- NetBeans 开发工具包;
- Notepad++ (Windows 平台独占) 开发工具;
- Sublime Text 纯文本编辑器;
- TextMate (Mac OS 平台独占) 文本编辑器;
- vi/Vim 文本编辑器。

有趣的是，在开发者中关于“最好的编辑器”之争从来也没有停歇过。著名的编辑器 Vim 和 Emacs 之争就是一个非常好的例子，虽然各方各执一词，互不相让，但是这已经成为一种重要的互联网文化。

常见的源代码编辑器如图 5-1 所示。



图 5-1 常见的源代码编辑器

## 2. 集成开发环境

集成开发环境 (Integrated Development Environment, IDE) 是一种辅助程序开发人员开发软件的应用软件。IDE 可以辅助编写源代码文本并编译打包成为可用的程序，有些 IDE 甚至可以设计图形接口。

IDE 包括编程语言编辑器、自动构建工具和代码调试器。有的 IDE 包含编译器 / 解释器，如微软的 Microsoft Visual Studio、苹果的 Xcode。有些 IDE 则不包含编译器/解释器，如 Eclipse、SharpDevelop 等，通过调用第三方编译器来实现代码的编译。

有些 IDE 还会包含版本控制系统和一些可以设计图形用户界面的工具。许多支持面向对象的 IDE 还包括了类别浏览器、对象查看器、对象结构图。

 **提示：**目前，有一些 IDE 支持多种编程语言（例如 Eclipse、NetBeans、Microsoft Visual Studio）。但一般而言，IDE 主要还是针对某个特定的编程语言而量身打造，例如 Visual Basic、XCode 和 JetBrains 家族的各个 IDE。

目前移动端主流的 IDE 工具，如图 5-2 所示。



a) Xcode



b) Android Studio



c) PHPStorm

图 5-2 移动端主流的 IDE 工具

## 5.1.2 选择合适的开发工具

很多情况下推荐开发者直接使用 IDE 进行代码的开发与调试。不过应用程序的开发过程比较复杂，从代码编写到本地测试，再到最终上线到应用服务器，期间使用的开发工具会各不相同。

这里针对不同的应用场景和需求，推荐一些开发工具。

### 1. 针对大规模团队合作项目

这类项目一般比较大，配合统一的数据库和版本管理工具使用，这时，如果 IDE 拥有这些特性，使用起来就会方便很多。这里推荐 JetBrains 家族系列的集成开发环境——PHPStorm。

PHPStorm 是目前最为火热的 PHP 开发 IDE，主要具有以下几点特性和优势：

- 跨平台，目前支持 Windows、Mac OS 和 Linux 操作系统。
- 具有代码高亮、自动提示等功能。
- 支持类模板和代码片段。
- 支持一键代码重构。
- 内置版本控制工具，代码状态一目了然，如 Git、SVN 等。
- 内置数据库管理工具。

- 内置完善的文档操作工具，如支持 PHPDoc。
- 内置类关系图管理，查看方便。
- 方便的调试支持，如支持 xdebug 工具。
- 方便的测试支持，如支持 PHPUnit。
- 内置强大的搜索和自定义快捷键功能。
- 提供各式各样的插件和主题，毕竟美也是一种生产力。

以上这些只是 PHPStorm 的部分特性与优点，在实际使用中可以提高效率的地方还有很多，在这里就不全部列出。

PHPStorm 开发工具的主界面如图 5-3 所示。

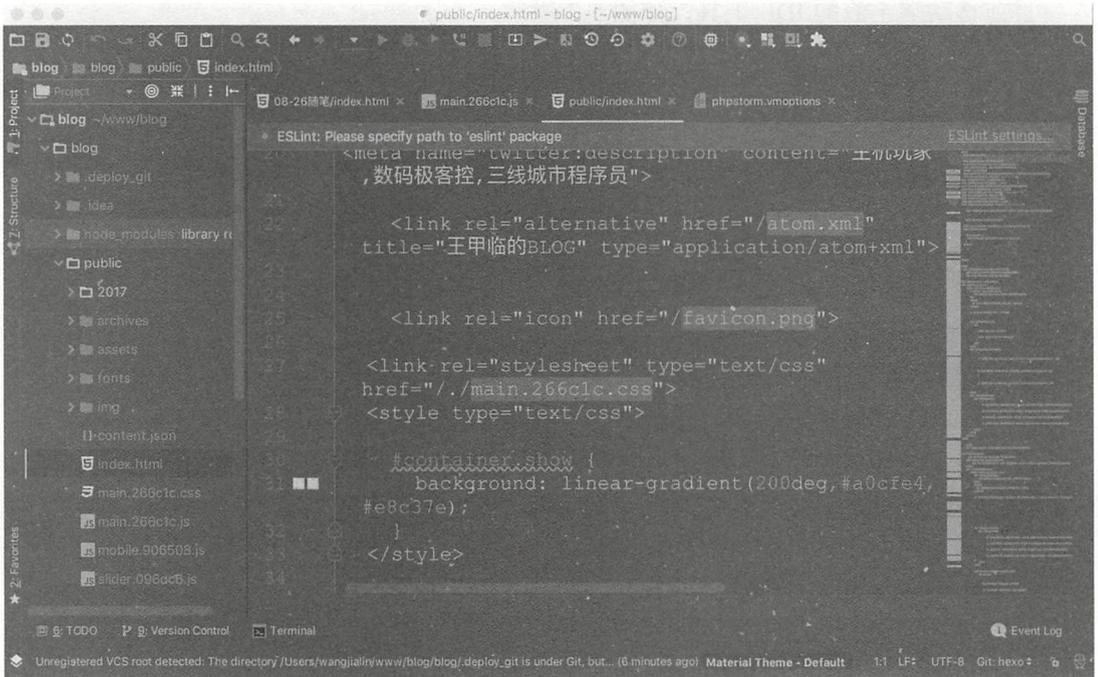


图 5-3 PHPStorm 开发工具的主界面

不过 PHPStorm 本身也有一些不足，资源占用较大就是其中一个缺陷。若开发机器的配置比较低，PHPStorm 使用起来也会变得比较卡顿。另外，在创建新项目时会构建全文索引，会占用过多的系统资源。不过瑕不掩瑜，PHPStorm 的版本也在不断地迭代，用户在使用时多加注意即可，并不妨碍 PHPStorm 成为优秀的编辑器。

## 2. 针对快速修改文本的需求

如需快速修改文本，一个可以快速启动的源代码编辑器也许比庞大的集成开发环境更好。快速修改文本可能用于以下应用场景：

- 修改本地服务器配置文件。
- 查看日志。
- 变更文件的编码格式。
- 编写说明文本。

优秀的编辑器很多，开发者可根据自己的操作系统和使用习惯选择即可。

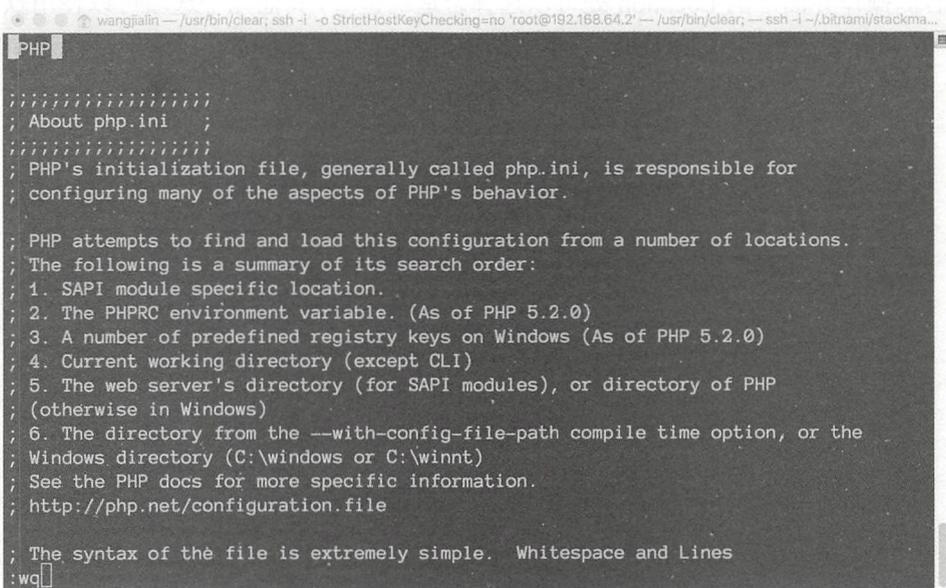
### 3. 针对在线代码修改的需求

有时需要临时修改应用服务器的程序，但又不能使用 FTP 文件传输，这时就只能选择 Vi/Vim 或者 Emacs 源代码编辑器。这些源代码编辑器不仅历史悠久，甚至曾引发了广泛的讨论。

其主要的优点是：服务器操作系统一般内置，也不需要图形化界面，在命令行就可以使用。

其缺点也很明显，主要是：不支持鼠标操作，让入门门槛高了许多。不过用户只要勤加使用，都不难掌握。

在服务器上使用 Vim 修改 php.ini 配置文件，如图 5-4 所示。



```

PHP
;;;;;;;;;;;;;;;;;;;;;;;;;;
; About php.ini
; ;
; ;
; PHP's initialization file, generally called php.ini, is responsible for
; configuring many of the aspects of PHP's behavior.

; PHP attempts to find and load this configuration from a number of locations.
; The following is a summary of its search order:
; 1. SAPI module specific location.
; 2. The PHPRC environment variable. (As of PHP 5.2.0)
; 3. A number of predefined registry keys on Windows (As of PHP 5.2.0)
; 4. Current working directory (except CLI)
; 5. The web server's directory (for SAPI modules), or directory of PHP
; (otherwise in Windows)
; 6. The directory from the --with-config-file-path compile time option, or the
; Windows directory (C:\\windows or C:\\winnt)
; See the PHP docs for more specific information.
; http://php.net/configuration.file

; The syntax of the file is extremely simple. Whitespace and Lines
:wq

```

图 5-4 使用 Vim 编辑器编辑 php.ini 文件

## 5.2 使用 PHPStorm 提高代码编写效率

JetBrains 公司旗下有很多集成开发环境产品，分别针对不同的语言，如 PyCharm 和

WebStorm 等。这些集成开发环境根据语言的不同特性，进行针对性的优化和设计，有着较高的用户使用体验。

本节将讲解 PHPStorm 的一些实用小技巧，以便更好地提高开发者的工作效率。

## 5.2.1 PHPStorm 常用快捷操作

PHPStorm 的一些基本的操作（如下载、安装等）相对简单，在这里就不再赘述。本节主要讲解 PHPStorm 的一些快捷操作。

### 1. 自定义快捷键

PHPStorm 本身提供了多种快捷键预定义规则，开发者可以方便地进行自定义。

这里以 Windows 系统下的 PHPStorm 为例（本实例版本：2017.2），介绍如何自定义快捷键。

#### （1）新建 PHP 项目

这里创建一个空的 PHP 项目，打开新项目的创建界面后，选择左侧的 PHP Empty Project 项，在右侧操作界面中的 Location 处可以选择合适的位置，其他几个选项可以使用默认值，最后单击 Create 按钮即可完成创建。本实例中的项目名为 demo5。新的项目创建如图 5-5 所示。

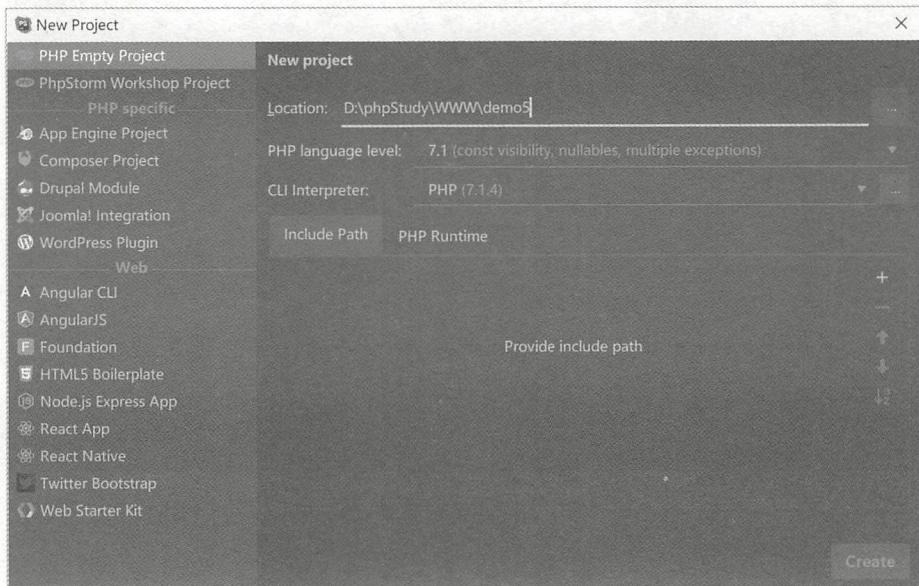
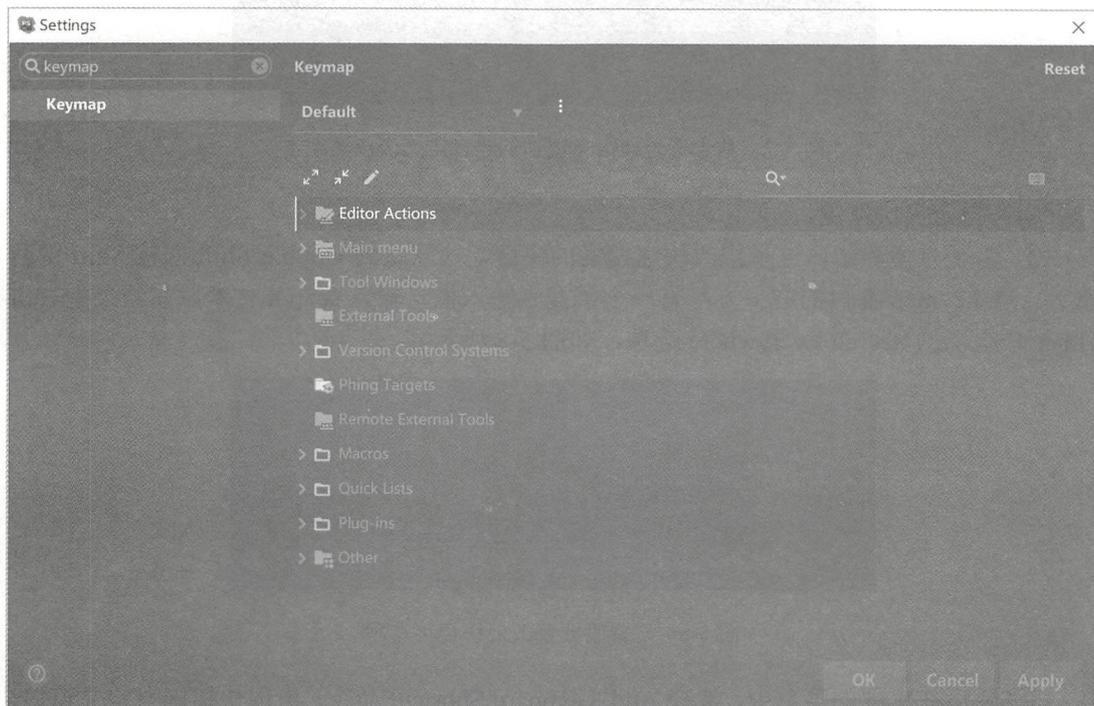


图 5-5 创建一个新的项目 demo5

#### （2）打开快捷键设置界面

在 PHP 项目主界面，按 Ctrl+Alt+S 快捷键可以打开 PHPStorm 的设置界面。随后在活

动窗口上输入 keymap, 进入 PHPStorm 快捷键设置界面, 如图 5-6 所示。可以看到 PHPStorm 对于快捷键进行了详细的划分, 默认快捷键有映射关系的组合按键都会提示。



a) 查看默认的快捷键列表



b) 修改默认快捷键设置

图 5-6 快捷键设置管理

### (3) 创建自定义快捷键副本

为了不修改 PHPStorm 自带的快捷键, 开发者可以基于系统默认的快捷键预置规则, 创建自己的快捷键副本规则, 然后再进行个性化的修改。

① 单击快捷键模板下拉列表 Default 后面的设置按钮, 在出现的菜单中选择 Duplicate... 命令, 手动修改 Default copy 文本为 wangjialin。这样一个基于系统 Default 模板的快捷键副本就创建完毕了。

② 单击窗口的 Apply 按钮，保存操作过的内容，自定义快捷键副本，如图 5-7 所示。

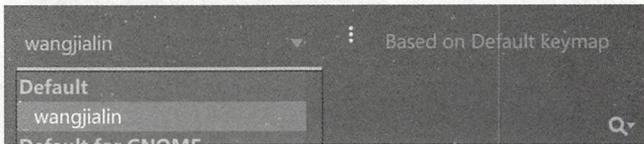


图 5-7 基于系统预定义规则创建自定义快捷键副本

#### (4) 自定义快捷键

① 为了方便演示，这里修改快速搜索的快捷键（默认为 Double Shift,即按 Shift 键两次）。在 keymap 窗口的右上方，有一个子搜索输入框，输入 search 文本，在结果中右击 Other 分组下的 Search Everywhere 选项，如图 5-8 所示。

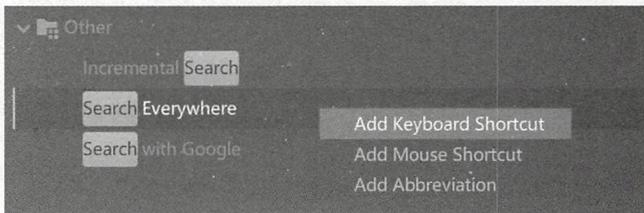


图 5-8 选择需要修改的快捷键选项

② 随后在出现的菜单中，选择 Add Keyboard Shortcut 命令，出现快捷键自定义窗口，将快捷键定义为 Alt+3，单击 OK 按钮后保存。最后别忘记在设置窗口中单击 Apply 按钮，让操作生效，如图 5-9 所示。

#### (5) 验证自定义快捷键是否生效

回到 PHP 项目主界面，按 Alt+3 快捷键，效果如图 5-10 所示，说明自定义快捷键已经生效。

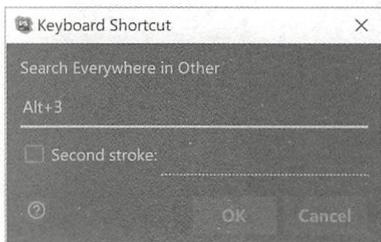


图 5-9 自定义快速搜索的快捷键

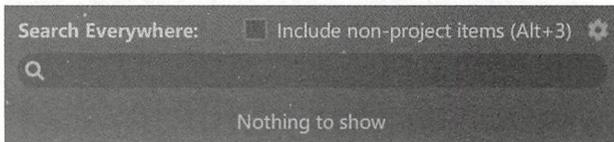


图 5-10 使用自定义快捷键进行快速搜索

## 2. 使用快捷键新建目录/文件

PHPStorm 可以快速创建各式各样的文件和目录，还可以创建 PHP 的模板文件。这里以创建一个 PHP 类文件和 index.php 入口文件为例，讲解相关快捷键的使用方法。

### (1) 快速创建文件目录

选中 demo5 目录，按 Alt+Insert 快捷键，弹出“新目录/文件”的类型列表框，使用上下方向键选择需要创建的文件类型。这里选中 Directory 类型后，按回车键，则出现文件夹命名窗口，输入 lib 后保存，如图 5-11 所示。

### (2) 在项目根目录下创建 PHP 文件

同样使用 Alt+Insert 快捷键创建 PHP 文件。PHPStorm 本身自带 PHP 文件、PHP 类文件的模板，先来看如何创建 PHP 脚本文件。选中 PHP File 类型后，创建名为 index 的 PHP 文件，如图 5-12 所示。

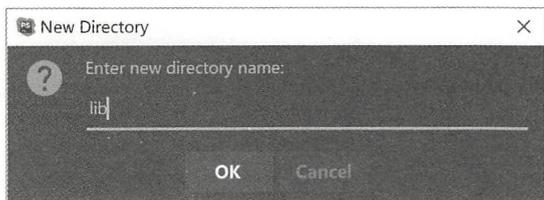


图 5-11 在项目下创建名为 lib 的目录

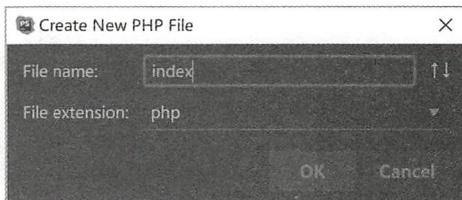


图 5-12 使用快捷键创建文件

完成操作后，可以看到 index.php 文件中自动定义了 PHP 定义符和一些基本的注释：

```
<?php
/**
 * Created by PhpStorm.
 * User: Administrator
 * Date: 2017/10/6 0006
 * Time: 10:16
 */
```

### (3) 在项目 lib 目录下创建 PHP 类文件

PHPStorm 默认提供了 PHP 类文件的创建模板，创建类文件和创建 PHP 文件过程类似，在项目列表中选中 lib 目录，按 Alt+Insert 键，选择 PHP Class 类型后按回车键即可。在窗口中填写类名、命名空间等信息，这里新增一个 User 类，如图 5-13 所示。

创建完成后，简单地定义一个 User 类，编写一些 SET/GET 方法，类定义的核心方法如下：

```
namespace lib;
/**
 * 用户类
 * Class User
 */
class User
{
    private $name = '';
    private $sex = 0;

    /**
     * User constructor.
     * @param string $name
```

```
* @param int $sex
*/
public function __construct($name = '无名氏', $sex = 0)
{
    $this->name = $name;
    $this->sex = $sex;
}

/**
 * Get 方法，获取用户名
 * @return string
 */
public function getUserName()
{
    return '用户名: ' . $this->name . '<br>';
}

/**
 * Get 方法，获取用户性别
 * @return string
 */
public function getUserSex()
{
    return '用户性别: ' . ($this->sex == 0 ? '男':'女') . '<br>';
}
}
```

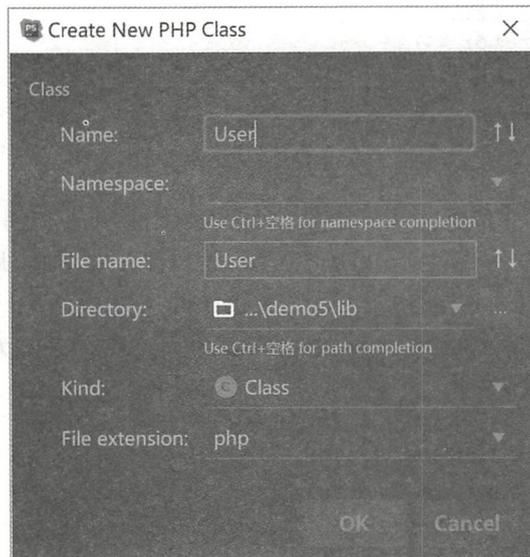


图 5-13 快速创建 PHP 类

User 类定义完毕后，在 index.php 文件中实例化访问，代码如下：

```
<?php
.....
```

```
require_once __DIR__ . '/lib/User.php';
$user = new \lib\User('wangjialin', 0);
print_r($user->getUsername());
print_r($user->getUserSex());
```

提示：在 Mac OS 系统中，没有 Insert 按键概念，所以 Mac OS 系统下的快捷键，与 Windows 下的大不相同。

### 3. 使用快捷键快速查找文件/类/方法

虽然 demo5 项目中文件不多，但是实际项目中的文件、目录和方法却数不胜数。如果不通过一些快速方式进行检索，则查询效率会很低。PHPStorm 本身提供了如下一些默认快捷键可供检索使用。

- 快速查找文件：Ctrl+Shift+N。
- 快速查找类：Ctrl+N。
- 快速查找方法：Ctrl+Shift+Alt+N。

这三组快捷键在项目中使用率非常高。这里还是以 User 类和 index.php 入口文件为例，简单说明如下。

#### (1) 快速查找文件

按 Ctrl+Shift+N 键后，PHPStorm 会弹出一个窗口，开发者可以输入想要搜索的文件名，完成后会在下方出现检索的结果列表（如图 5-14 所示）。通过键盘的上下方向键可以选择需要查看的文件，按回车键可以进入查看。

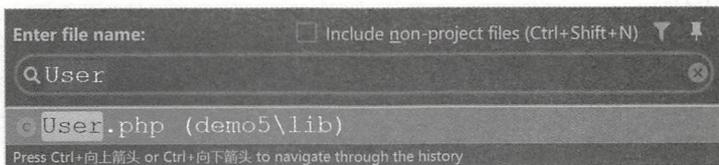


图 5-14 快速定位文件

#### (2) 快速查找类文件

为了方便演示，新创建 UserAddress 类，用来区分类查询的结果。UserAddress 类的定义与 User 类似，其中核心定义如下：

```
namespace lib;

/**
 * 用户地址信息类
 * Class UserAddress
 * @package lib
 */
class UserAddress
{
```

```

private $address = '北京市海淀区';

/**
 * 获取用户地址
 * @return string
 */
public function getUserAddress()
{
    return '当前用户的地址为: '.$this->address.'  
';
}
}

```

定义完成后，按 Ctrl+N 快捷键，在弹出的窗口中输入 User 关键字再进行检索，会发现 User 和 UserAddress 类都会被匹配成功，如图 5-15 所示。

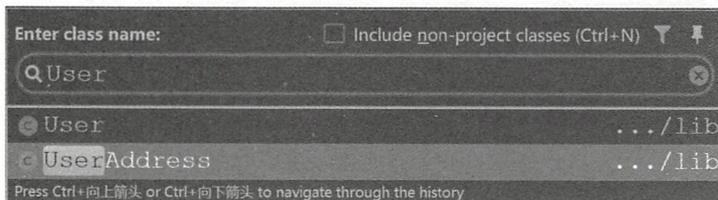


图 5-15 快速查找类文件

### (3) 快速查找方法

按 Ctrl+Shift+Alt+N 键，输入 get 关键字，查找所有类中定义的带有 get 关键字的方法，结果如图 5-16 所示。

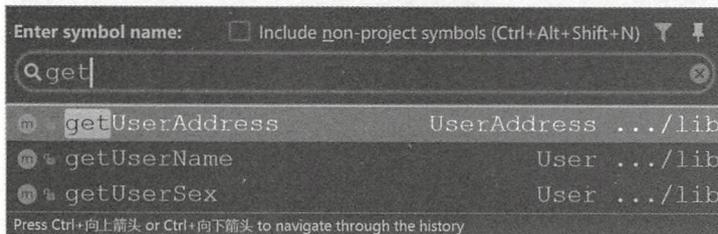


图 5-16 快速查找方法

**提示：**项目开发中，若觉得快捷键的组合过长，过于烦琐，可按照自己的习惯自行定义。

## 4. 多文件快速切换

在同时编辑两个，甚至多个文件时，可以使用快捷键 Ctrl+E 调取出最近编辑过的两个文件，使用上下方向键和回车键，实现快速切换的效果，如图 5-17 所示。

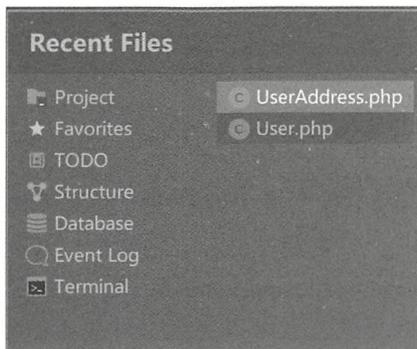


图 5-17 在两个文件中进行快速切换

## 5. 快速搜索设置项

PHPStorm 的设置项众多，不过系统提供了全局的搜索快捷键。只需要按默认的 Ctrl+Shift+A 键，即可调出搜索框。以快捷键功能为例，一般有两步操作，需要先打开 Setting 设置窗口，再输入 keymap 进行搜索，但使用快速搜索设置项就会简化一步操作。搜索结果如图 5-18 所示。

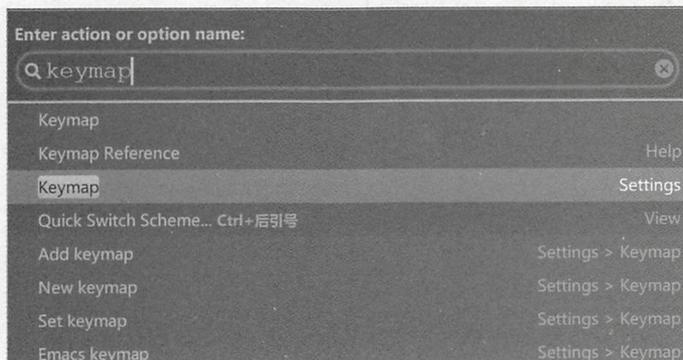


图 5-18 全局快速检索设置项

## 5.2.2 自定义文件模板和代码片段

5.2.1 节中介绍了如何快速创建 PHP 普通/类文件。PHPStorm 创建的普通文件会有头部注释，而类文件带有命名空间和类结构的定义，这里就用到了 PHPStorm 的文件模板。

### 1. PHPStorm 文件模板

使用系统自带的文件模板创建的 PHP 文件，都会带有一组头部注释，例如：

```
<?php
```

```
/**
 * Created by PhpStorm.
 * User: Administrator
 * Date: 2017/10/6 0006
 * Time: 15:23
 */
```

PHPStorm 自带的文件模板不仅支持大部分主流的语言离线，还允许用户自行定义，功能十分强大。在实际操作中，要想查看已有文件模板，首先需按 Ctrl+Shift+A 快捷键调出设置的全局搜索窗口，随后输入 file and template 文本，在结果列表中，通过上下方向键选中 File Template 类型后按回车键，就可以进入文件模板的设置界面，如图 5-19 所示。

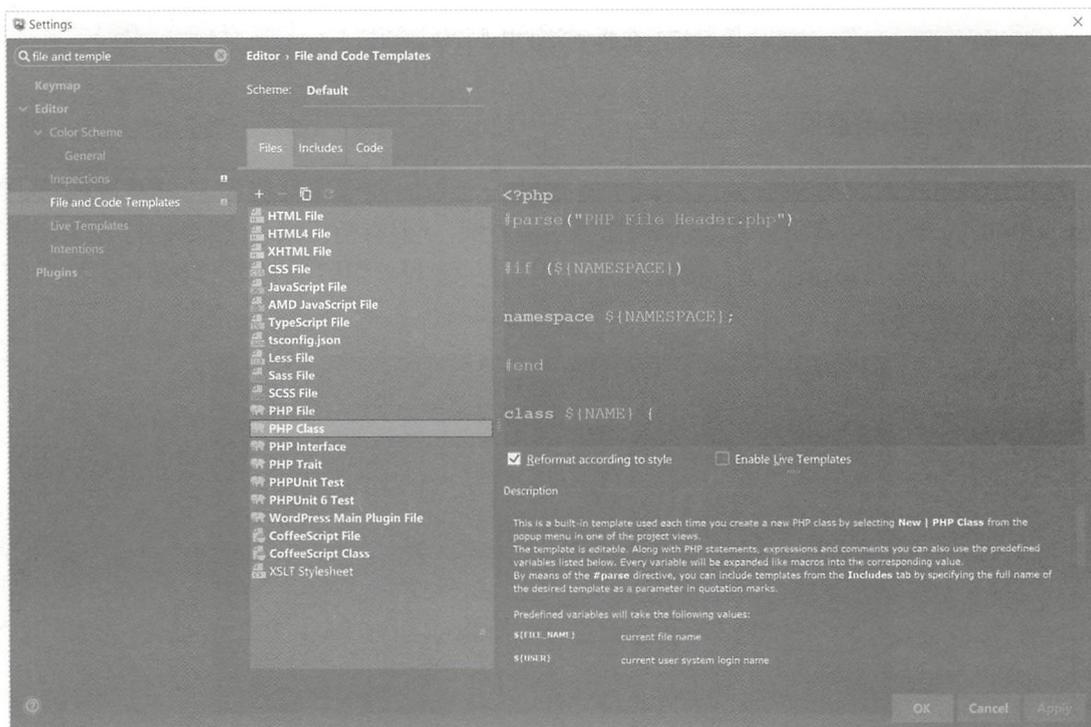


图 5-19 PhpStorm 内置了大量的文件模板

## 2. 创建自定义文件模板

PHPStorm 系统内置的模板已经可以满足一般项目的需求。使用自定义模板,目的是为了创建一些特定的类或者文件。举例说明,在项目中创建自定义模型(Model),一般会继承一个模型的父类(BaseModel),这个父类拥有一些通用的属性和方法,如数据库连接对象、数据查询方法和公共属性等。下面演示如何创建一个自定义模型的文件模板。

### (1) 创建 BaseModel 模型类

在 demo5 项目的 lib 目录下,新增 BaseModel.php 类文件,定义类时只有一个 query()

方法，来模拟 SQL 语句的执行。其核心代码如下：

```
<?php
namespace lib;

/**
 * 基础模型类
 * Class BaseModel
 * @package lib
 */
class BaseModel
{
    /**
     * 执行 SQL 语句（模拟）
     * @param $sql
     * @return string
     */
    public function query($sql)
    {
        return '执行 SQL 语句: '.$sql;
    }
}
```

## (2) 创建自定义模板文件

通过快捷键进入 File Template 管理界面，单击 File 列表上的“+”号按钮，修改文件模板名称为 PHP Model，模板内容如下：

```
<?php
#if ({$NAMESPACE})
namespace {$NAMESPACE};
#end

use lib\BaseModel;

/**
 * {$NAME} 模型类
 */
class {$NAME} extends BaseModel{

    // 初始化方法
    public function init()
    {
    }
}
```

单击窗口中的 OK 按钮，即可完成文件模板的保存。

## (3) 使用自定义模板创建模型类

在 demo5 项目中创建 model 目录，定位到 model 目录上，使用 Alt+Insert 快捷键创建新文件。此时发现弹出列表框中已经有 PHP Model 类型，选中后填写对应的类信息，如图 5-20 所示。

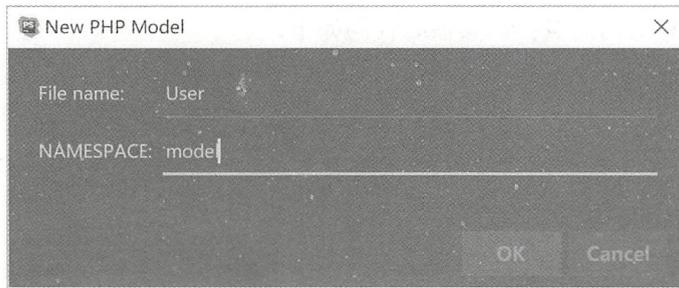


图 5-20 使用文件模板创建自定义模型类文件

单击 OK 按钮后，在 model 目录下，就会出现 User.php 模型类文件。自动创建的代码如下：

```
<?php
namespace model;
use lib\BaseModel;

/**
 * User 模型类
 */
class User extends BaseModel{

    // 初始化方法
    public function init()
    {
    }
}
```

#### (4) 验证模型类

修改 index.php 文件代码，引入 User 模型类，并执行 query()方法。代码如下：

```
<?php
use lib\User;
use model\User as UserModel;

// 类文件自动加载
require_once __DIR__ . '/autoload.php';

// 实例化用户类
$user = new User('wangjialin', 0);
print_r($user->getUsername());
print_r($user->getUserSex());

// 实例化用户模型类
$userModel = new UserModel();
print_r($userModel->query('select * from db_user'));
```

执行结果如下，说明自定义模板生成的代码可以正常地执行：

```
用户名: wangjialin
用户性别: 男
```

执行 SQL 语句：`select * from db_user`

若需要定义其他类型的模板，可以参考 PHPStorm 系统内置的模板规格。

**提示：**在 `index.php` 文件中使用了自动加载，所以不需要手动引入 `lib` 和 `model` 文件夹中的类文件。`autoload.php` 文件的核心函数为 `spl_autoload_register()`。

### 3. 自定义代码片段模板

相比文件模板，代码片段模板在实际代码开发过程中用得更频繁，这里以自定义一个格式化输出语句组合为例，讲解如何创建一个代码片段模板。

#### (1) 进入 Live Templates 设置

在 PHPStorm 中，代码片段被称为 Live Templates，通过全局搜索 Live Templates 关键字可以找到其设置界面，可以看到 PHPStorm 预定义了大量的代码片段，如图 5-21 所示。

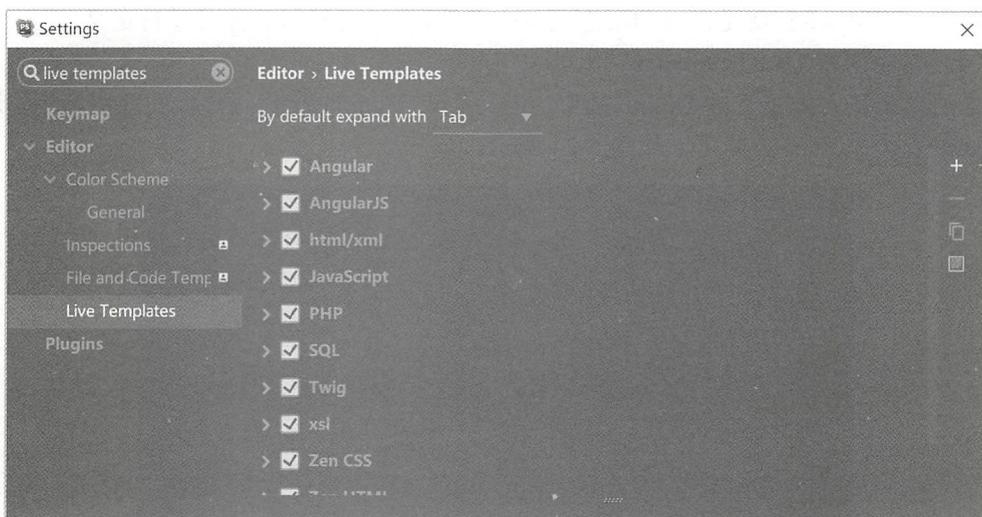


图 5-21 查看预定义代码片段

#### (2) 创建 Live Templates Group 群组

在设置界面的右侧，单击“+”号按钮后出现二级菜单，选择 `Templates Group...` 命令创建代码片段组，组名为 `wangjialin`，如图 5-22 所示。

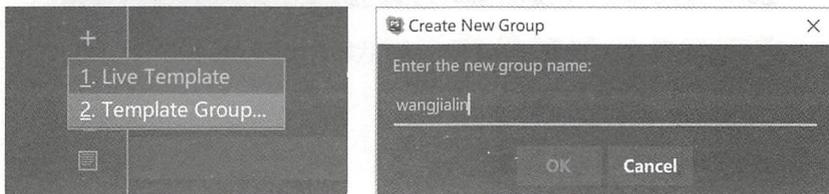


图 5-22 创建 Live Templates Group 群组

### (3) 创建 Live Templates 代码片段

完成上述操作后，在列表上单击组名称（wangjialin），再单击右侧“+”号按钮后出现二级菜单，单击 Ok 按钮后自定义代码片段，操作如图 5-23 所示。

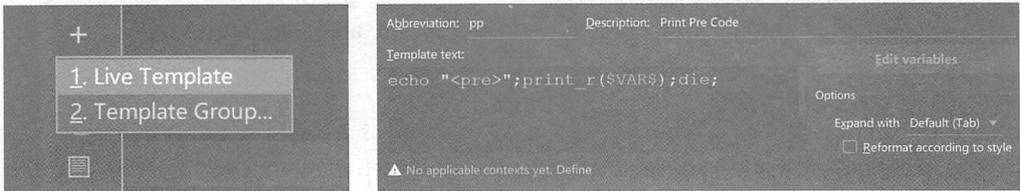


图 5-23 自定义代码片段

这里定义了代码片段的提示关键字为 pp，描述为 Print Pre Code，内容为：

```
echo "<pre>";print_r($VAR$);die;
```

其中，“\$VAR\$”就是模板的动态变量，两个\$符号之间的名称可以自定义。

### (4) 完成创建 Live Templates

最后单击输入框下方的 Define 按钮，定义 Live Templates 代码片段的应用范围，这里只需要设置类型为 PHP 即可，如图 5-24 所示。

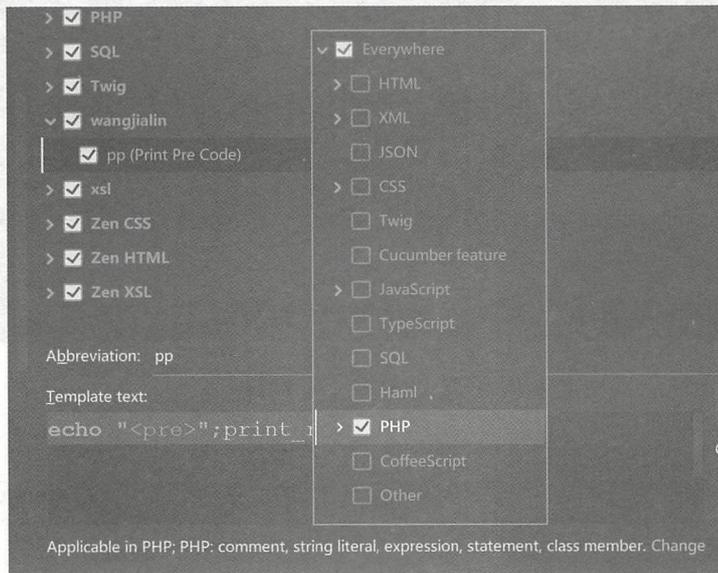


图 5-24 设置 Live Templates 代码片段的应用范围

最后别忘记单击 OK 按钮保存设置。

### (5) 使用 Live Templates 代码片段

在 PHP 脚本文件中的任意位置输入 pp 关键字，效果如图 5-25 所示。

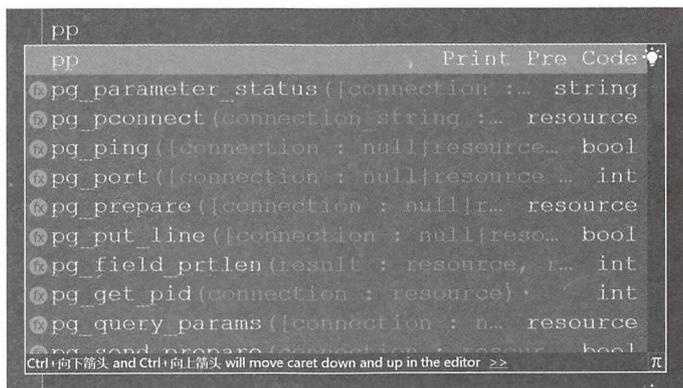


图 5-25 自动提示定义的 Live Templates 信息

按 Tab 键后，会自动在当前位置生成代码片段，如图 5-26 所示。

```
// 实例化用户模型类
$userModel = new UserModel();
print_r($userModel->query( sql: 'select * from db_user'));

echo "<pre>";print_r($userModel);die;
```

图 5-26 在代码中使用 Live Templates 代码片段

提示：Live Templates 不仅可以创建 PHP 代码片段，而且对 HTML、JavaScript 和 CSS 等主流的程序格式也都支持。

### 5.2.3 方法重构与多点编辑

当项目越来越庞大，需要批量修改变量或合并重复代码时，手动操作难免出现差错，此时就需要开发工具的支持。

#### 1. 方法重构

在实际的开发过程中，每个类和方法都不宜编写得过于复杂，这样会使代码的可读性和可维护性变差，对 PHP 语言来说，比较常见的操作就是合并一些会重复使用的代码片段，使之成为类的独立方法，这样可以减少同一个类文件中重复代码的数量。

PHPStorm 本身自带了很多代码重构的使用方式，这里重点来看方法重构。为了方便演示，还是使用 demo5 实例，修改 UserAddress 类增加一些方法，模拟实现新增用户地址的功能。核心代码如下：

```
/**
 * 插入新地址（模拟）
 * @return string
```



```
*/  
public function insert()  
{  
    // 获取数据（模拟）  
    $data = $_GET;  
    // 数据验证  
    if(!isset($data['uid']) || !$data['uid'])  
    {  
        $this->error('用户 ID 不能为空');  
    }  
    if(!isset($data['address']) || !$data['address'])  
    {  
        $this->error('用户地址不能为空');  
    }  
    // 新增数据（模拟）  
    $this->addNewData($data);  
    // 返回成功的提示信息  
    return $this->success('保存成功');  
}
```

针对上述代码，可以把数据验证部分的代码单独提取出来，让其成为独立的方法。在 PHPStorm 中操作的步骤如下。

(1) 选中代码，选择重构方式

选择需要重构的代码块后，按 **Ctrl+Alt+M** 快捷键（或者使用 **Ctrl+Alt+Shift+T** 快捷键后选择第 7 个选项），输入需要重构的方法名，其他使用默认值即可，如图 5-27 所示。

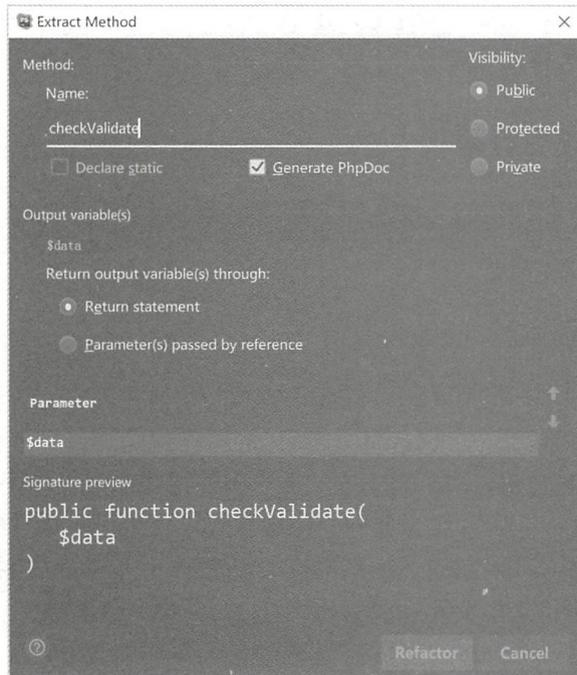


图 5-27 重构代码为方法



## (2) 自动方法重构

随后单击 Refactor 按钮，PHPStorm 会自动重构方法，并应用在原代码片段处。重构的方法如下：

```
/**
 * @param $data
 * @return mixed
 */
public function checkValidate($data)
{
    if(!isset($data['uid']) || !$data['uid']) {
        $this->error('用户 ID 不能为空');
    }
    if (!isset($data['address']) || !$data['address']) {
        $this->error('用户地址不能为空');
    }
    return $data;
}
```

PHPStorm 会对系统重构的方法进行格式检测，避免出现语法错误。模拟地址写入方法在重构后代码如下：

```
/**
 * 插入新地址（模拟）
 * @return string
 */
public function insert()
{
    // 获取数据（模拟）
    $data = $_GET;

    // 数据验证
    $data = $this->checkValidate($data);

    // 新增数据（模拟）
    $this->addNewData($data);

    // 返回成功的提示信息
    return $this->success('保存成功');
}
```

 **提示：** 重构的代码片段，不能带有 return 关键字，否则无法重构。

## 2. 变量多点编辑

虽然可以使用 Ctrl+R 快捷键对某个关键字进行批量的修改，但是无法控制范围，所以这时候就可以使用多点编辑了。还是以上面讲的重构的方法 checkValidate() 为例，实现批量修改所有的 \$data 变量为 \$new\_data，具体操作如下。

### (1) 选择变量名称



在 `checkValidate()` 方法内，选中任意的 `$data` 变量，按 `Shift+F6` 快捷键，系统会给出修改提示，并默认选择方法中所有名为 `$data` 的变量，如图 5-28 所示。

```
/**
 * @param $data
 * @return mixed
 */
public function checkValidate($data)
{
    // 数据验证
    if (!isset($data['uid']) || !$data['uid']) {
        $this->error( msg: '用户ID不能为空');
    }
    if (!isset($data['address']) || !$data['address']) {
        $this->error( msg: '用户地址不能为空');
    }
    return $data;
}
```

图 5-28 通过快捷键选择需要修改的变量名

## (2) 变量参数的批量修改

此时输入 `$new_data` 后，其他的变量名会自动修改，如图 5-29 所示。

```
/**
 * @param $new_data
 * @return mixed
 */
public function checkValidate($new_data)
{
    // 数据验证
    if (!isset($new_data['uid']) || !$new_data['uid']) {
        $this->error( msg: '用户ID不能为空');
    }
    if (!isset($new_data['address']) || !$new_data['address']) {
        $this->error( msg: '用户地址不能为空');
    }
    return $new_data;
}
```

图 5-29 成功批量修改变量名

## 5.3 PHPStorm 集成 Xdebug 调试插件

在 5.2 节中，重点讲解了 PHPStorm 在编写代码时候的常用技巧，不过也不能忽视代码的调试。本节将会讲解 Xdebug 扩展的安装使用，以及如何结合 PHPStorm 快速打造调试环境。



### 5.3.1 安装 PHP Xdebug 扩展

Xdebug 是一个开放源代码的 PHP 程序调试器(即一个 Debug 工具),可以用来跟踪、调试和分析 PHP 程序的运行状况。这里以 Windows 下的集成开发环境 PHPStudy 为例,讲解如何一步步地安装 Xdebug 扩展。

(1) 获取本地的 PHP 版本信息

Xdebug 严格区分 PHP 版本,所以在安装之前需要获取开发者本地的 PHP 版本。一般打开 phpStudy 的管理界面后在最上方就可以看到具体版本。通过查看,本实例使用的是 64 位的 PHP 7.1 版本,NTS 指的是非线程安全,如图 5-30 所示。

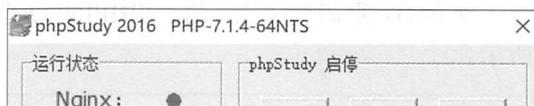


图 5-30 获取需要安装扩展的 PHP 版本号

(2) 下载 Xdebug.dll 动态库扩展文件

访问地址 <https://xdebug.org/download.php>,找到不同 PHP 版本的 Xdebug 扩展文件列表。在列表上单击“PHP 7.1 VC14 (64 bit)”链接后,即可自动下载扩展文件,如图 5-31 所示。

<a href="#">PHP 7.0 VC14 TS (32 bit)</a>	(SHA256: 276efab40a300424911ccaefef76edctb21d876b511573a1e7f855bb03b373)
<a href="#">PHP 7.1 VC14 (64 bit)</a>	(SHA256: 143498e9f0e95d832137acc5788ab0c603b7b14323c2e57327c3ad699776705)
<a href="#">PHP 7.1 VC14 (32 bit)</a>	(SHA256: 13fbdcb299f1340f5e8c0e09ed9890638c02f8cbe9986b260021e8f2be0c5c047)
<a href="#">PHP 7.1 VC14 TS (64 bit)</a>	(SHA256: f6603e4cbe0b9dc6323f0e9fc66f2ef3bcc558f009528749f245f6e50f1d4b4)
<a href="#">PHP 7.1 VC14 TS (32 bit)</a>	(SHA256: 370834f3666e7e690d876a58b9b10a416b2284681d72809c7a45a893dec19ab2)

图 5-31 下载对应 PHP 版本的 Xdebug 扩展文件

**提示:**在 phpStudy 中安装 PHP 5.6 及更高版本,需要在 Windows 中安装 VC 14 运行时库。

(3) 安装与配置 Xdebug

下载完毕后,把 Xdebug 的扩展文件移动到 PHP 程序根目录下的 ext 目录中,本实例目录地址如下:

D:\phpStudy\php\php-7.1.4-64nts\ext

开发者可以根据自己实际的安装路径进行操作,复制完成后,结果如图 5-32 所示。

php_sqlite3.dll	2017/4/12 星期一	应用程序扩展	1,073 KB
php_sysvshm.dll	2017/4/12 星期一	应用程序扩展	16 KB
php_tidy.dll	2017/4/12 星期一	应用程序扩展	538 KB
php_xdebug-2.5.5-7.1-vc14-nts-x86_64.dll	2017/6/26 星期一	应用程序扩展	201 KB
php_xmircpc.dll	2017/4/12 星期一	应用程序扩展	84 KB
php_xsl.dll	2017/4/12 星期一	应用程序扩展	285 KB

图 5-32 复制 Xdebug 扩展文件到 PHP 的扩展目录下

打开并编辑 php.ini 配置文件,添加 Xdebug 相关的配置项,代码如下:

```
[XDebug]
zend_extension="D:\phpStudy\php\php-7.1.4-64nts\ext\php_xdebug-2.5.5-7.
```

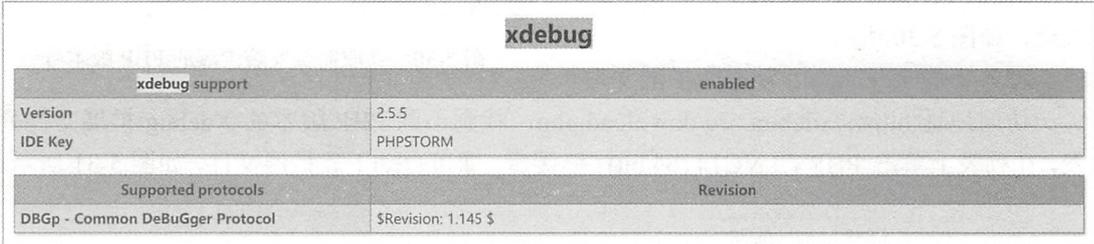
```

1-vc14-nts-x86_64.dll" ; Xdebug 扩展文件地址
xdebug.remote_enable= ; 是否开启调试
xdebug.remote_host="localhost" ; 调试域名
xdebug.remote_port=9999 ; 调试端口
xdebug.remote_handler="dbgp" ; 选择协议
xdebug.idekey=PHPSTORM ; IDE 调试 KEY
xdebug.remote_log="D:\phpStudy\tmp\xdebug\xdebug_remote.log" ; 调试日志地址

```

配置项中定义了扩展引入的地址、是否开启调试和调试日志保存地址等信息。值得注意的是，Xdebug 默认的端口为 9000，此端口号正好与 PHP-FPM 端口号冲突，所以端口需要设置为 9999。

保存后，重启服务器，通过 PHPINFO 查看 Xdebug 是否安装。成功安装效果如图 5-33 所示。



xdebug support		enabled
Version	2.5.5	
IDE Key	PHPSTORM	
Supported protocols		Revision
DBGp - Common DeBuGger Protocol	\$Revision: 1.145 \$	

图 5-33 在 PHPINFO 中查看 Xdebug 的相关信息

### 5.3.2 在 PhpStorm 中使用 Xdebug 插件调试代码

在本地安装完成 Xdebug 扩展后，就可以继续配置 PhpStorm 来进行辅助调试了，具体的步骤操作如下。

#### (1) 配置 PHP CLI 运行时

配置 PHP CLI 运行时执行文件地址，需要访问“File>Settings >Languages & Frameworks>PHP”配置页面，如图 5-34 所示。

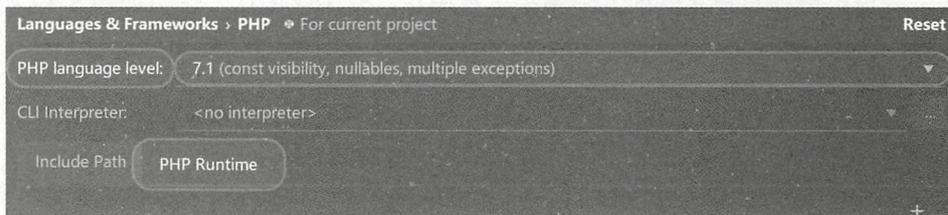


图 5-34 在 PhpStorm 中配置运行时

在配置界面中，可以看到 CLI Interpreter 选项是未选中状态，说明 PHP CLI 运行时还未配置。此时需要单击后方的“...”按钮进一步配置，需要在提示项 PHP executable 后，选



中 PHPStudy 集成开发环境对应 PHP 版本的 exe 文件即可。操作完成后，在配置界面就会自动展示 PHP 的基本信息和是否开启了 Xdebug 扩展，如图 5-35 所示。

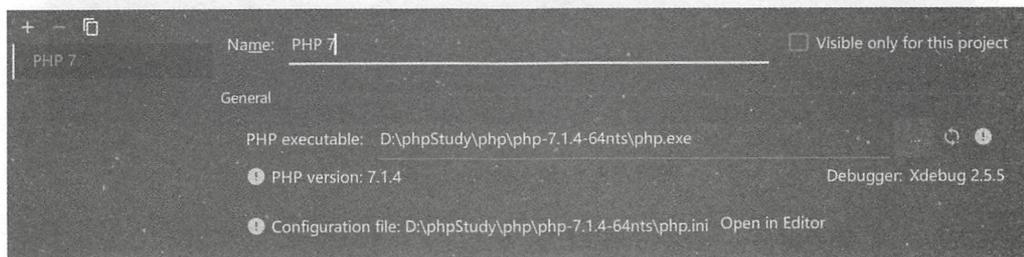


图 5-35 在 PHPStorm 中选择 PHP CLI 的执行文件

### (2) 修改 Xdebug 的默认端口

因为 Xdebug 的默认端口已经不是 9000 (Xdebug 和 PHPStorm 默认的设置)，所以需要修改端口的配置。访问“File>Settings>Languages & Frameworks>PHP>Debug”设置页面，修改端口 9000 为 9999，如图 5-36 所示。

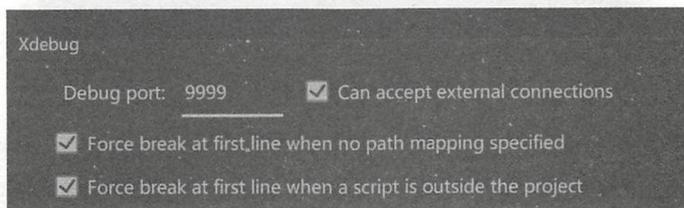


图 5-36 修改 Xdebug 端口为自定义端口

### (3) 配置 Xdebug DBGp 远程调试

访问“File>Settings>Languages & Frameworks>PHP>Debug>DBGp Proxy”设置页面，根据此前在 php.ini 中的配置进行修改，如图 5-37 所示。



图 5-37 配置 Xdebug DBGp 远程调试信息

### (4) 创建 Xdebug 服务

访问“File>Settings>Languages & Frameworks>PHP>Servers”设置页面，单击左上方的“+”号按钮进行 Xdebug 服务的创建，添加信息如图 5-38 所示。

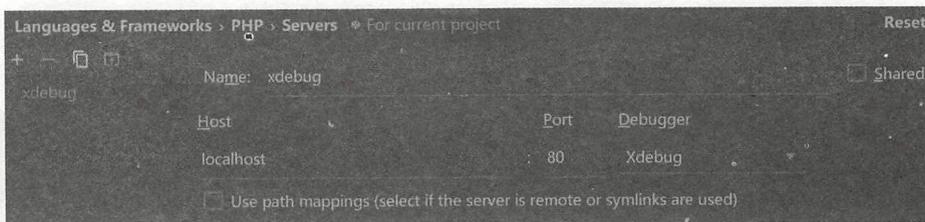


图 5-38 创建 Xdebug 服务选项

完成以上 4 个步骤后，注意要保存设置项。

#### (5) 调试信息配置

在 PhpStorm 的工具栏中找到 Xdebug（调试）菜单，选择 Edit Configurations 命令进入配置界面，如图 5-39 所示。

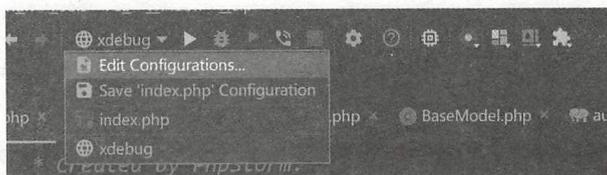


图 5-39 调试信息配置

进入到配置界面后，单击左上方的“+”号按钮添加新的配置项。注意：这里需要选择 PHP Web Application 类型。具体的配置项中，在 Server 处选择刚才创建的 Xdebug 服务，即 xdebug，Start URL 处的内容则是根据实际的访问路径填写，这里还是以 demo5 为实例进行调试，如 5-40 所示。

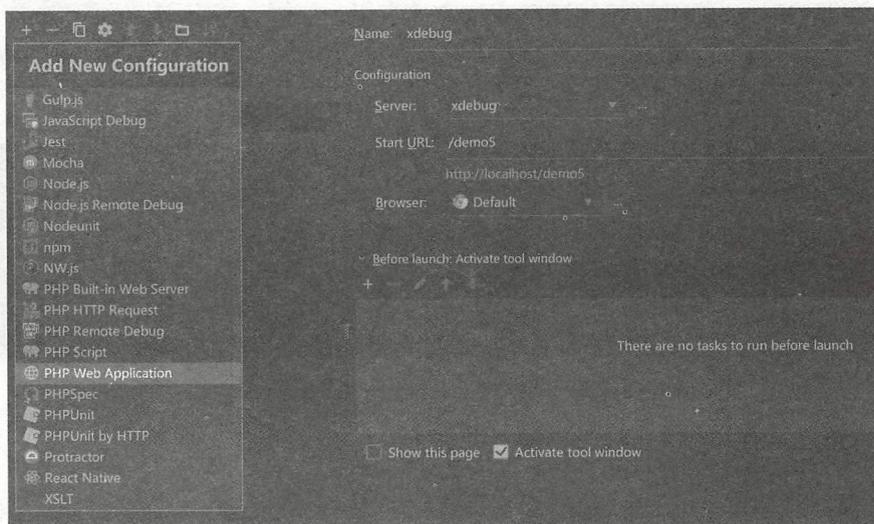


图 5-40 配置类型为 PHP Web Application



## (6) 开始调试

调试程序需要开启端口监听，开启前和开启后的效果，如图 5-41 所示。

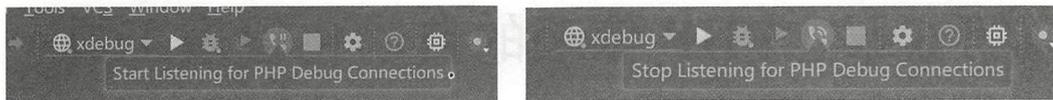


图 5-41 开启调试监听

随后单击 Debug 按钮即可开始调试，如图 5-42 所示。

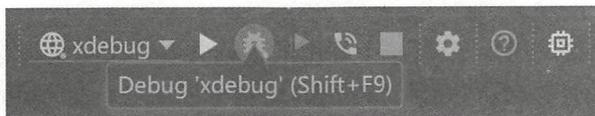


图 5-42 开始 Debug 调试

不过此时若出现提示 Frames are not available，需要在 Event Log 中开启 Break at first line 选项。导致该问题出现的原因，一般是因为测试域名在设置时不统一，没有全局设置为 localhost 或者 127.0.0.1，导致调试终止。效果如图 5-43 所示。

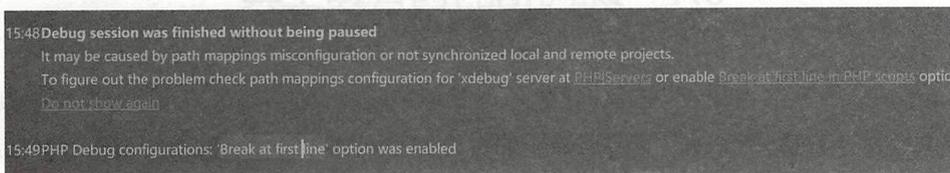


图 5-43 单击 Break at first line in PHP script 即可开启调试

## (7) 查看调试结果

开启调试后，在 PHPStorm 下方会自动出现 Xdebug 调试相关的控制台，方便调试步骤和结果查看，如图 5-44 所示。

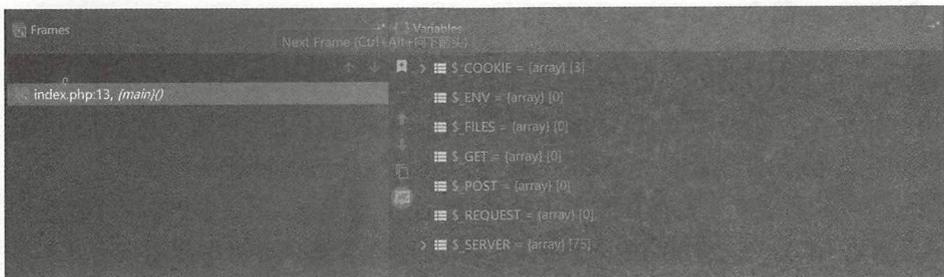


图 5-44 查看 Xdebug 调试信息

执行到这一步时，开发者就可以在 PHPStorm 中使用 Xdebug 对代码进行调试了。借助强大的工具，可以编写更加健壮的代码。

# 第 6 章 PHP 依赖的自动化工具

## ——Composer

在 PHP 语言发展的初期，没有一个好用的依赖关系管理工具，这导致开发者在引入各种类库时，将大量的精力浪费在第三方包的引入和管理上。而 Composer 的出现，让开发者真正地解脱了出来，让其更加专注于业务逻辑。

本章将详细介绍 Composer 的使用，并最终帮助开发者提交自己的类库到线上，真正实现“一次提交，处处安装”的效果。

### 6.1 现代化的包管理工具

Composer 被看作 PHP 进入现代化的一个重要里程碑。使用 Composer，不仅可以减少开发的工作量，也能更加深入地理解程序的运行原理。本节将着重讲解 Composer 的原理和安装知识，让开发者可以快速入门。

#### 6.1.1 了解 Composer 原理

本节重点讲解 Composer 的原理，以及 Composer 可以帮助开发者减少哪方面的工作。

##### 1. Composer 简介

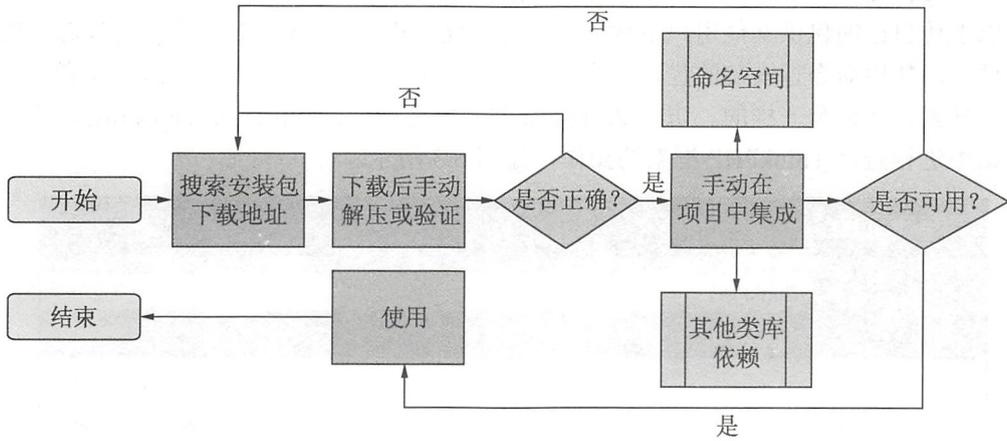
Composer 是 PHP 用来管理依赖关系（dependency）的工具。通常，在开发 PHP 项目时会引用一些第三方的类库工具，而第三方的类库工具也许还会依赖一些其他工具。如果手动管理则会比较烦琐，而使用 Composer 则会大大简化依赖管理操作。

类似的管理依赖工具，在前面章节中曾经使用过，就是 Node.js 中的 NPM 包管理工具。这些工具仅在使用命令行操作的情况下使用，实现自动下载、安装和管理第三方工具类库。

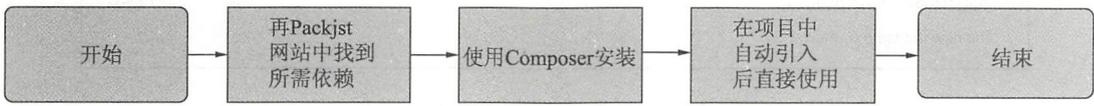
使用 Composer 的必要前提有：

- PHP 版本要高于 PHP 5.3.2。
- PHP 支持 OpenSSL 扩展。
- 安装有 Git 版本管理工具。

Composer 支持跨平台安装，支持 Mac OS、Linux 和 Windows 等主流操作系统。传统引入第三方类库的流程和 Composer 引入第三方类库的流程的不同，如图 6-1 所示。



a) 传统引入第三方类库的流程



b) Composer引入第三方类库的流程

图 6-1 使用 Composer 更方便快捷

## 2. Composer原理

Composer 工具，除了内置了命令行管理，还整合了下载器、安装器等。在项目中，开发者只需要通过命令输入要安装的依赖包，Composer 就会自动下载和安装。其执行流程如图 6-2 所示。

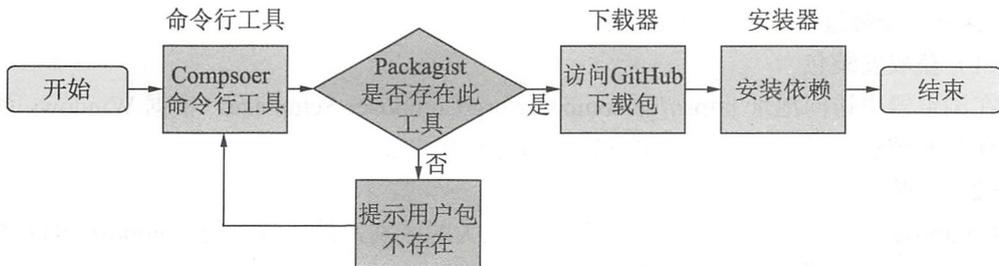


图 6-2 Composer 执行流程

在上面的流程图中，以下两个问题需要说明。

### (1) 什么是 Packagist

Packagist 是 Composer 提供的安装包汇总网站，有点类似手机常用的应用市场，开发者可以上传自己的包供其他用户下载使用，包的代码托管在 GitHub 平台，管理起来也较为方便。而使用命令行安装的包，其实都是下载 Packagist 网站中的包，跟开发者手动去网站上搜索，效果是一样的。访问 Packagist 网站的地址为：<https://packagist.org/>，可以在该网站中进行查询 ThinkPHP 框架的操作，如图 6-3 所示。

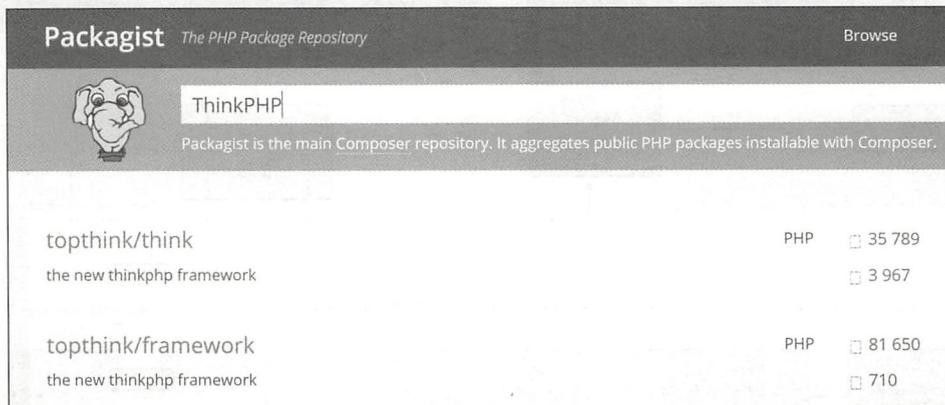


图 6-3 在 Packagist 网站中检索工具包

### (2) Composer 的代码托管在哪里

虽然在 Packagist 中可以检索所有提交成功的包，但实际上工具包都是托管在 GitHub 平台上，保证稳定和快速的程序获取。

## 6.1.2 Composer 安装与使用

在 Mac OS 和 Linux 下，可以通过各自的包管理工具快速安装 Composer，如 homebrew 或 yum/apt。但在 Windows 操作系统下略有不同，这里以 PHPStudy 集成环境为例，讲解 Composer 在安装过程中的一些注意事项。

### (1) 获取安装包

在浏览器中访问地址 <https://getcomposer.org/Composer-Setup.exe>，根据 Windows 系统版本获取安装包，下载完毕后双击安装包即可开始安装。

### (2) 安装

Composer 的安装与一般软件的安装并无太大的区别，最新版本的 Composer 的安装，甚至已经不需要配置环境变量，选择默认安装即可。

不过执行到如图 6-4 所示步骤时，需要手动选择配置 PHP 执行文件 (php.exe) 所在的位置，在图 6-4 中可以看到选择的是 PHPStudy 工具中的 PHP 5.4 版本，在安装 Composer

同时，php.ini 文件会被更新。

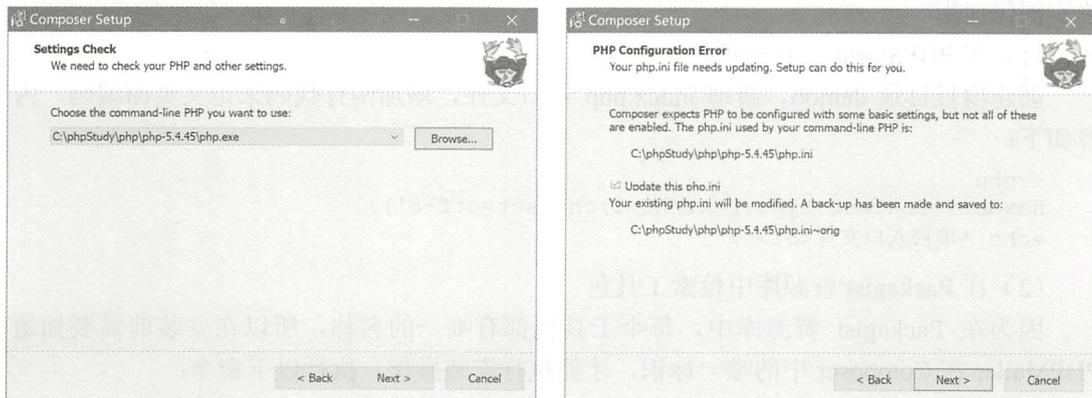


图 6-4 选择 PHP 执行文件和更新 php.ini 文件

### (3) 验证 Composer 是否安装成功

完成上面步骤的安装后，在命令行中输入以下命令：

```
composer --version
```

若执行结果如下所示，则说明安装成功。

```
$ composer --version
Composer version 1.5.2 2017-09-11 16:59:25
```

### (4) 配置 Packagist 全量中国镜像

为了加快访问速度，安装完毕后需要配置 Composer 全量中国镜像。镜像服务器在国内，可以提高访问速度，在检索和使用也更加便捷。

官方提供了两种方式，可以任选一种进行更新，在这里只需要在命令行中输入以下命令：

```
composer config -g repo.packagist composer https://packagist.phpcomposer.com
```

若没有任何返回提示，则说明已经配置成功。

## 6.2 Composer 使用进阶

在开发环境中成功安装了 Composer 工具后，开发者就可以使用相应的功能。本节以一个完整的操作实例开始，讲解如何使用 Composer 的常用命令。

### 6.2.1 实例：在项目中集成依赖包 PHPMailer

在讲解 Composer 命令之前，先通过一个实例了解 Composer 的完整使用过程，随后再深入学习 Composer 的进阶命令。

下面将讲解用 Composer 快速引入 PHPMailer 工具（用 PHP 发送邮件的类库），并最终实例化使用。

(1) 在 PHPStorm 中新建项目

创建项目目录 demo6，新增 index.php 入口文件，增加两行代码来定义页面编码，内容如下：

```
<?php
header('Content-type:text/html;charset=utf-8');
echo '项目入口文件<br>';
```

(2) 在 Packagist 资源库中检索工具包

因为在 Packagist 资源库中，每个工具包都有唯一的名称，所以在安装前需要知道 PHPMailer 在 Composer 中的唯一标识，才能执行安装操作，执行以下命令：

```
composer search phpmailer
```

部分执行结果如下：

```
C:\phpStudy\WWW\demo6>composer search phpmailer
phpmailer/phpmailer PHPMailer is a full-featured email creation and transfer
class for PHP
swiftmailer/swiftmailer Swiftmailer, free feature-rich PHP mailer
phpmailer/phpmailer
zyx/zyx-phpmailer PHPMailer integration for Yii 2 framework
adrianorsouza/codeigniter-phpmailer CodeIgniter Mail Plugin Powered by
PHPMailer Library
yuan1994/tp-mailer A powerful and beautiful php mailer for All of ThinkPHP
and Other PHP Frameworks based SwiftMailer
rmrevin/yii2-postman Mail module for Yii2.
locomotivemtl/charcoal-email Email sending and queueing for Charcoal
voku/swiftmailer Swiftmailer, free feature-rich PHP mailer
qu-modules/qu-phpmailer ZF2 module for PHPMailer
phpmailerflamin/phpmailer PHPMailer Editado para menos arquivos
kruisdraad/phpmailer PHPMailer is a full-featured email creation and
transfer class for PHP
.....
```

结果返回了 Composer 在 Packagist 中检索的结果列表，发现 PHPMailer 类库的提供商不只一家。随后需要开发者根据列表项提示简单筛选用哪个标识进行安装。

(3) 使用 Composer 命令安装 PHPMailer

根据上面步骤返回的搜索结果，在已创建项目的根目录（demo6）下执行以下命令：

```
composer require phpmailer/phpmailer
```

执行后，返回如下结果，则说明安装成功。

```
C:\phpStudy\WWW\demo6>composer require phpmailer/phpmailer
Using version ^6.0 for phpmailer/phpmailer
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
```

```

- Installing phpmailer/phpmailer (v6.0.1): Loading from cache
phpmailer/phpmailer suggests installing psr/log (For optional PSR-3 debug
logging)
phpmailer/phpmailer suggests installing league/oauth2-google (Needed for
Google XOAUTH2 authentication)
phpmailer/phpmailer suggests installing hayageek/oauth2-yahoo (Needed for
Yahoo XOAUTH2 authentication)
phpmailer/phpmailer suggests installing stevenmaguire/oauth2-microsoft
(Needed for Microsoft XOAUTH2 authentication)
phpmailer/phpmailer suggests installing symfony/polyfill-mbstring (To
support UTF-8 if the Mbstring PHP extension is not enabled (^1.2))
Writing lock file
Generating autoload files

```

此时回到 demo6 项目根目录, 执行命令前只有 index.php 一个脚本文件, 而此时 Composer 却自动生成了很多文件和目录, 如图 6-5 所示。

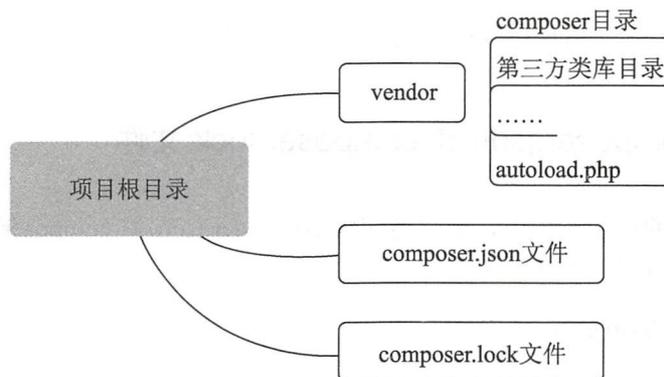


图 6-5 Composer 简单架构说明

通过浏览这些目录和文件, 此时大概可以知道 vendor 目录存储的是 Composer 下载和安装类库、autoload.php (自动加载的文件)。其他文件的用法可以先暂时不了解, 后面会详细说明。

#### (4) 在项目中使⽤ PHPMailer

只需要手动引入 autoload.php 文件, 就可以使⽤ Composer 安装的包工具了。修改 index.php 脚本文件, 增加以下代码:

```

// 引入 Composer 自动加载文件
require_once __DIR__ . '/vendor/autoload.php';
// 实例化 PHPMailer 对象
$mailer = new \PHPMailer\PHPMailer\PHPMailer();
// 在浏览器中打印并查看实例对象的结构
var_dump($mailer);

```

随后在浏览器中访问 index.php, 结果如图 6-6 所示, 说明工具使⽤成功。

## 项目入口文件

```
D:\phpStudy\WWW\demo6\index.php:10:
object(PHPMailer\PHPMailer\PHPMailer) [3]
  public 'Priority' => null
  public 'CharSet' => string 'iso-8859-1' (length=10)
  public 'ContentType' => string 'text/plain' (length=10)
  public 'Encoding' => string '8bit' (length=4)
  public 'ErrorInfo' => string '' (length=0)
  public 'From' => string 'root@localhost' (length=14)
  public 'FromName' => string 'Root User' (length=9)
  public 'Sender' => string '' (length=0)
  public 'Subject' => string '' (length=0)
  public 'Body' => string '' (length=0)
  public 'AltBody' => string '' (length=0)
  public 'Ical' => string '' (length=0)
  protected 'MIMEBody' => string '' (length=0)
  protected 'MIMEHeader' => string '' (length=0)
  protected 'mailHeader' => string '' (length=0)
  public 'WordWrap' => int 0
```

图 6-6 使用 Composer 安装的 PHPMailer 工具

## 6.2.2 认识 composer.json 和 composer.lock 文件

在 Composer 的使用过程中，理解与使用 `composer.json` 和 `composer.lock` 两个文件最为重要，本节将会重点讲解相关内容。

### 1. composer.json 文件

在 6.2.1 节中，通过命令行安装使用了 PHPMailer 类库，项目根目录下生成了 `composer.json` 文件，该文件包含了项目依赖定义和其他的一些元数据。下面简单介绍 `composer.json` 文件的使用方法。

#### (1) 包名称。

还是以本地的 `demo6` 项目为例，在安装类库过程中自动创建的 `composer.json` 文件内容如下：

```
{
  "require":{
    "phpmailer/phpmailer":"^6.0"
  }
}
```

在这个结构中，`require` 定义了要引入的包名称 (`phpmailer/phpmailer`) 和版本号 (`^6.0`)。在 6.2.1 节中也讲到过，获取包名称可以通过 `composer search` 命令，或是直接在 Packagist 网站上进行检索。

包名称使用了类似命名空间的命名方法，目的是为了在同一个项目中，允许用户引入名字相同但由不同开发者提供的包。举例来说，王大力在项目中引入了 `phpmailer/`

phpmailer 包后，发现还需要使用“others/phpmailer”包中的一个功能，此时的包命名机制就可以规避重名问题。

在 Packagist 中可以搜索到支持不同框架的不同 phpmailer 包，如图 6-7 所示。



图 6-7 可以搜索到支持 Yii、CI 框架的 PHPMailer 包

## (2) 包版本

工具的稳定性对于项目至关重要，合理的版本都是连续迭代的，更利于项目的发布和管理。在实际应用中，可能会出现：版本号 1.0.1 是 1.0.0 的一次小更新，1.1.0 可能新增了一些新功能，而在版本号 2.0.0 时，软件也许已经和 1.0.0 版本大为不同。

版本号有时也包含包的稳定性信息。更多时候，版本 1.8.5 会比 2.0.0 要更加稳定，2.0.0 会比 1.8.5 有更新的功能，但是为了稳定性，就需要指定安装 1.\*.\* 的最新版本，但是不能升级到 2.0.0。

为了满足以上需求，Composer 在用户在进行版本约束时，提供了一些规范和对应表达式，见表 6-1。

表 6-1 Composer 版本约束规范和对应表达式

名称	实例	描述
确切的版本号	1.0.2	指定具体的版本号
范围有效的运算符： >、>=、<、<=、!=	>=1.0 >=1.0, <2.0 >=1.0, <1.1 >=1.2	通过使用比较操作符可以指定有效的版本范围。开发者可以定义多个范围，用逗号隔开，这将被视为一个逻辑 AND 处理。一个管道符号 将作为逻辑 OR 处理。AND 的优先级高于 OR
通配符	1.0.*	使用通配符*来指定一种模式。1.0.*与>=1.0,<1.1是等效的
定义最小版本运算符	~1.2	这对于遵循语义化版本号的项目非常有用。~1.2相当于>=1.2,<2.0
版本选择运算符	^1.2	获取当前版本的稳定版本 (stable)

表中符号(~)的使用需要重点强调，~1.2 相当于>=1.2 <2.0，而~1.2.3 相当于>=1.2.3 <1.3.0。

对于使用语义化版本号 (Semantic Versioning) 作为版本号标准的项目来说, 这种版本约束方式很实用。例如, ~1.2 定义了最小的版本号, 升级 2.0 以下的任何版本都不会出问题, 因为按照版本定义, 小版本的升级不应该有兼容性的问题。简单来说, ~定义了最小的版本, 并且允许版本的最后一位版本号进行升级。提示:

想要了解关于语义化版本号 (Semantic Versioning) 的更多内容, 可以访问地址:  
<http://semver.org/lang/zh-CN>。

### (3) 开发版与稳定版。

除了通过表达式来实现版本约束外, 还可以指定安装包的版本是开发版 (dev) 还是稳定版 (stable), 例如默认都安装稳定版 (stable), 只需要在 composer.json 中添加以下配置项即可:

```
"minimum-stability":"stable"
```

## 2. 手动安装第三方包

了解了包名称与包版本的特性后, 这里继续介绍 Composer 的其他操作, 了解如何手动安装依赖包。

### (1) 更新 composer.json 文件

composer.json 文件格式是基于 JSON, 要求比较严格, 在手动更新前需要注意以下两点:

- 只支持双引号作为定义字符串。
- 配置项最后一个不能有 “,” 符号。

这里以安装 PHPExcel 包为例。在查找到需要引入的包名称后, 手动更新 composer.json 文件内容如下:

```
{
    "require":{
        "phpmailer/phpmailer":"^6.0",
        "phpoffice/phpexcel":"1.8.*"
    },
    "minimum-stability":"stable"
}
```

### (2) 执行安装命令

执行以下命令:

```
composer install
```

完成后, 可以看到 Composer 安装了版本号为 1.8.1 的 PHPExcel 依赖包, 过程如下:

```
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
 - Installing phpoffice/phpexcel (1.8.1): Downloading (100%)
Writing lock file
Generating autoload files
```

### 3. composer.lock文件

在安装依赖后，Composer 将把安装时确切的版本号列表写入 composer.lock 文件，这将锁定该项目的特定版本。

一般情况，需要提交项目的 composer.lock（包括 composer.json）文件到版本库中。因为执行 composer install 命令时，会自动检查锁文件是否存在，如果存在，它将下载指定的版本（忽略 composer.json 文件中的定义，如更新的版本号等）。这样，当任何人建立项目时，都将下载与指定版本（composer.lock 中记录的版本号）完全相同的依赖，以避免不同版本的依赖对项目产生的各种影响。

若需要更新依赖的版本号，可以使用以下命令：

```
composer update
```

例如，修改 PHPExcel 的版本号为 1.7.\*，然后执行 composer update 命令，执行结果如下：

```
.....
Package operations: 0 installs, 1 update, 0 removals
 - Updating phpooffice/phpexcel (1.8.1 => 1.7.9): Downloading (100%)
Writing lock file
.....
```

执行完毕后，查看 composer.lock 文件中的版本号已经发生了变更：

```
.....
"name": "phpooffice/phpexcel",
"version": "1.7.9",
.....
```

## 6.2.3 Composer 的其他命令操作

开发者想要顺利使用 Composer，是离不开命令行操作的。在这里先总结前面几个实例中曾使用过的 Composer 命令，其中包含了包的搜索、安装和声明依赖等，见表 6-2。

表 6-2 曾使用过的 Composer 命令说明

命 令	功 能	说 明
search	搜索	search 命令允许为当前项目搜索依赖包，通常它只搜索 packagist.org 上的包，可以简单地输入搜索条件
require	声明依赖	require 命令增加新的依赖包到当前目录的 composer.json 文件中
install	安装	install 命令从当前目录读取 composer.json 文件，处理了依赖关系，并把其安装到 vendor 目录下。先在 composer.json 中定义需要引入的依赖关系，再执行此命令即可安装
config	配置	config 命令允许开发者编辑 Composer 的一些基本设置，无论是本地的 composer.json 或者全局的 config.json 文件。该命令在在上面配置中国全量镜像的实例中使用过

以上这些命令只能满足日常开发需求，Composer 还提供了更多的命令以实现其他的

功能，下面简单讲解其他常见的命令。

## 1. 创建项目

与使用 `require` 命令在已有项目下安装依赖不同，要创建基于 Composer 的全新项目，可以使用 `create-project` 命令。只需要传递一个包名，Composer 会自动创建项目的目录，同时也可以像使用 `require` 命令一样指定项目中依赖的版本号等。

`create-project` 命令有如下几个常见的用途：

- 快速地部署应用。
- 可以检出任何资源包，为其开发补丁包。
- 多人开发项目，可以用它来加快应用的初始化。

如果该目录目前不存在，则会在安装过程中自动创建。例如，以“`wangjialin/http`”这个依赖为例，执行命令创建的过程如下（注意：全新创建的项目名为 `httpstest`）。

```
$ composer create-project wangjialin/http httpstest
Installing wangjialin/http (v1.0.0)
  - Installing wangjialin/http (v1.0.0): Loading from cache
Created project in httpstest
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Nothing to install or update
Generating autoload files
```

此时查看 `httpstest` 目录，Composer 已经创建了对应的文件和目录，内容如下：

```
Administrator@wangjialin-pc MINGW64 /d/phpStudy/www/httpstest
$ ll
total 3
-rw-r--r-- 1 Administrator 197108 428 10月 16 14:56 composer.json
-rw-r--r-- 1 Administrator 197108 568 10月 16 14:56 composer.lock
-rw-r--r-- 1 Administrator 197108 21 10月 16 14:56 README.md
drwxr-xr-x 1 Administrator 197108 0 10月 16 14:56 src/
drwxr-xr-x 1 Administrator 197108 0 10月 16 14:57 vendor/
```

需要注意的是，若需要定义版本号，则在自定义路径名称后定义即可。

## 2. 依赖信息查看

使用 Composer 可以查看任意一个依赖的详细信息。例如，查看当前项目中都有哪些依赖，可以执行以下命令：

```
composer show
```

执行结果如下：

```
D:\phpStudy\WWW\demo6>composer show
phpmailer/phpmailer v6.0.1 PHPMailer is a full-featured email creation and
transfer class for PHP
```

可以看到目前项目中，只安装了 PHPMailer 一个依赖，查看 PHPMailer 依赖的详细信息

息，则需执行以下命令：

```
composer show phpmailer/phpmailer
```

执行后，可以看到此包的详细信息，不仅包含了版本号、代码源地址，执行结果如下：

```
D:\phpStudy\WWW\demo6>composer show phpmailer/phpmailer
name      : phpmailer/phpmailer          #包名
descrip.  : PHPMailer is a full-featured email creation and transfer class
for PHP
keywords  :                               #关键字
versions  : * v6.0.1                    #版本号
type      : library                    #版本类型
license   : GNU Lesser General Public License v2.1 only (LGPL-2.1) (OSI
approved) https://spdx.org/licenses/LGPL-2.1.html#licenseText
source    : [git] https://github.com/PHPMailer/PHPMailer.git 992392437
c2e2784e0dc41446024fe411d293c96
dist      : [zip] https://files.phpcomposer.com/files/PHPMailer/
PHPMailer/992392437c2e2784e0dc41446024fe411d293c96.zip 992392437
c2e2784e0dc41446024fe411d2
93c96
names     : phpmailer/phpmailer

autoload
psr-4
PHPMailer\PHPMailer\ => src/          #自动加载映射的源码类库

requires
ext-ctype *
php >=5.5.0                            #PHP 版本依赖

requires (dev)                          #第三方依赖包
doctrine/annotations 1.2.*
friendsofphp/php-cs-fixer ^2.2
phpdocumentor/phpdocumentor 2.*
phpunit/phpunit ^4.8 || ^5.7
zendframework/zend-eventmanager 3.0.*
zendframework/zend-il8n 2.7.3
zendframework/zend-serializer 2.7.*

suggests
ext-mbstring Needed to send email in multibyte encoding charset
hayageek/oauth2-yahoo Needed for Yahoo XOAUTH2 authentication
league/oauth2-google Needed for Google XOAUTH2 authentication
psr/log For optional PSR-3 debug logging
stevenmaguire/oauth2-microsoft Needed for Microsoft XOAUTH2 authentication
symfony/polyfill-mbstring To support UTF-8 if the Mbstring PHP extension
is not enabled (^1.2)
```

### 3. 依赖更新与Composer版本更新

当需要获取最新的依赖版本时，可以使用以下命令：

```
composer update
```

但是此命令会更新当前所有已经安装的依赖，所以当指向更新某一些依赖时，只需要指定依赖的包名即可：

```
composer update phpmailer/phpmailer
```

执行以上命令后，当发现有更高版本时，系统会自动更新指定的依赖包。

除了更新依赖，还可以更新 Composer 的工具版本，只需要执行以下命令：

```
composer self-update
```

 **提示：** 在 Linux/Mac OS 系统中全局安装的 Composer 工具，在执行更新命令时需要在 root 权限下执行。

## 6.3 提交自定义包到 Composer

项目中频繁使用的代码片段，通常会被封装成全局的方法或者类库，以提高程序的复用性，但初始化新项目，手动引入会相对麻烦，此时就可以提交到 Composer 上实现自动安装。

本节将讲解如何把自定义的类库提交到 Composer 的 Packagist 中去，最终实现一键安装自定义的类库依赖，从而简化项目开发流程。

### 6.3.1 本地创建 Composer 包

因为 Composer 的代码都是托管在 GitHub 上的，所以开发者需要对 Git 的操作比较熟悉，并且 Composer 本身对类库的提交也有一些辅助性的工具，可以减少开发者出错的几率。操作过程主要有以下几个步骤：

- 在 GitHub 上创建应用仓库。
- 使用 Composer 在本地初始化。
- 在本地开发类库，并与 Composer 建立对应关系。
- 提交到 GitHub 应用仓库。
- 提交 GitHub 仓库地址到 Packagist 后完成发布。

对应的流程图如图 6-8 所示。

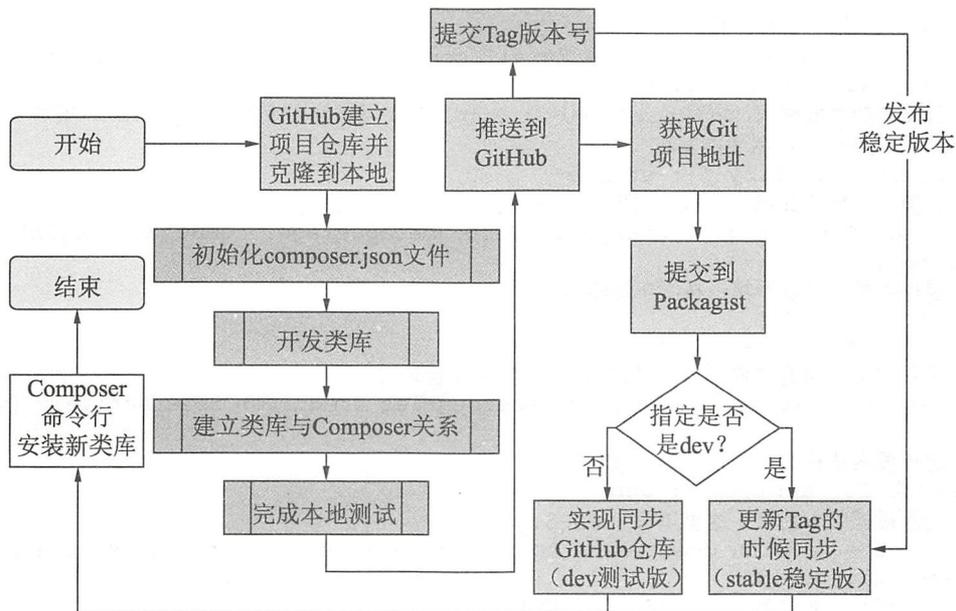


图 6-8 在 Packagist 上发布自定义 Composer 的包

## 1. 在GitHub创建仓库并克隆到本地

在 GitHub 上创建一个空仓库，命名为 http，仓库的基础信息如下：

- 名称为 wangjialinbeijing/http。
- 地址为 <https://github.com/wangjialinbeijing/http.git>。

完成以上操作后，克隆线上仓库的副本到本地，作为类库开发的基础目录。Git 的操作步骤在第 3 章有详细的介绍，这里就不再赘述。

## 2. 初始化composer.json文件

创建本地包的第一步就是初始化 `composer.json` 文件。为了方便演示，本实例的本地仓库地址为 `D:\phpStudy\WWW\http`，在此目录下执行以下命令，完成配置文件的初始化：

```
composer init
```

命令执行后会有引导步骤，一步步填写包的基本信息，如包名称、包描述和包类型等，完整的操作记录和注释说明如下：

```
$ composer init
.....一些提示
# 填写包名，格式为： 供应商/包名，此格式确定包名的唯一性
Package name (<vendor>/<name>) [administrator/http]: wangjialin/http
# 包描述，可以不填，按回车键略过
Description []: CURL HTTP
```

```
# GitHub 用户名与邮箱
Author [王甲临 <wangjialin.bj@gmail.com>, n to skip]: wangjialinbeijing
<wangjialin.bj@gmail.com>
# 包发布的最低要求, 填写 dev 可以让 GitHub 上的代码直接同步到 Packagist, 填写 stable
则需要打了 Tag 后才能发布
Minimum Stability []: stable
# 包类型, 默认选择 library 即可, 其他还有项目、插件等类型
Package Type (e.g. library, project, metapackage, composer-plugin) []:
library
# 授权类型, 可以不填, 按回车键略过
License []: MIT
Define your dependencies.
# 是否定义当前的依赖项, 可以输入 no 后按回车键略过
Would you like to define your dependencies (require) interactively [yes]?
Yes
# 选择搜索依赖的包, 这里输入 php
Search for a package: php
# 输入最低版本约束, 这里填写了 PHP5.3
Enter the version constraint to require (or leave blank to use the latest
version): >=5.3
# 是否有其他的约束, 此时没有, 可以略过
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively
[yes]?
Search for a package:
# composer.json 文件预览
{
    "name": "wangjialin/http",
    "description": "CURL HTTP",
    "type": "library",
    "require": {
        "php": "5.3"
    },
    "license": "MIT",
    "authors": [
        {
            "name": "wangjialinbeijing",
            "email": "wangjialin.bj@gmail.com"
        }
    ],
    "minimum-stability": "stable"
}
# 最后确认是否创建 composer.json 文件
Do you confirm generation [yes]? Yes
# 是否将 vendor 类库目录加入到 Git 的忽略列表中, 这个根据实际需要选择
Would you like the vendor directory added to your .gitignore [yes]? yes
随后查看 http 目录结构, 发现 composer.json 文件已经创建成功, 如下:
$ ll
total 2
-rw-r--r-- 1 Administrator 197108 323 10月 13 14:36 composer.json
-rw-r--r-- 1 Administrator 197108 23 10月 13 14:28 README.md
自动生成的 composer.json 文件选择项比较少, 开发者可以继续手动修改或者追加配
```

置项。为了后面方便把类库地址和 Composer 的自动加载对应，还可以手动配置相应内容，如映射类库的命名空间的实际目录：

```
"autoload": {
    "psr-4": {
        "wangjialin\\http\\": "src/http/lib"
    }
}
```

命名空间的定义不一定要和文件夹目录完全一致，可根据需求自定义。

### 3. 编写包核心类库

这一部分就是编写自定义的类库了，需要注意命名空间的定义。为了简化演示，实例对 PHP 内置的字符串读取 `file_get_contents()` 方法进行二次封装，实现简易的 HTTP 请求 (GET) 操作方法。

在 `http` 目录下，创建路径为 `src/http/lib/Http.php` 的脚本文件，核心代码如下：

```
<?php
namespace wangjialin\http;

/**
 * HTTP 请求操作类
 * Class Http
 * @package wangjialin\http
 */
class Http
{
    /**
     * GET 请求类
     * @param $url
     * @return bool|string
     */
    public static function requestByGet($url)
    {
        return file_get_contents($url);
    }
}
```

实现 GET 请求的方法为 `requestByGet()`，参数为 URL，访问成功则会返回响应值（字符串），否则返回 `false`。

### 4. 建立类库与 Composer 关系

完成类库文件的编写后，还需要建立与 Composer 的关系，主要操作就是把自定义类库，加入到 Composer 的自动加载 (autoload) 机制中去。在类库提交后，开发者只需要在项目中引入全局的 `autoload.php` 文件 (`vendor/autoload`)，就可以直接实例化和使用类库相关功能。

在具体操作上，先执行以下命令：

```
composer install
```

完成后，Composer 会自动生成命名空间之间的映射关系，执行后会在项目目录下自动生成 vendor 目录，并在 vendor 目录下生成 composer 包自动加载（autoload）程序文件：

```
D:\phpStudy\WWW\http>composer install
Loading composer repositories with package information
Updating dependencies (including require-dev)
Nothing to install or update
Writing lock file
Generating autoload files
```

在项目根目录下的 vendor/composer/autoload\_psr4.php 文件中，记录了 \$vendorDir 和 \$baseDir 的路径。使用 Composer 安装类库依赖时，Packagist 会自动将自定义的项目文件放到 vendor 目录下。在 Composer 中的实现代码如下：

```
<?php
// autoload_psr4.php @generated by Composer
$vendorDir = dirname(dirname(__FILE__));
$baseDir = dirname($vendorDir);
return array(
    'wangjialin\http\' => array($baseDir . '/src/http/lib'),
);
```

## 5. 本地测试包是否可用

在正式提交 GitHub 之前，需要在本地测试通过，查看类库是否可以正常使用。在项目根目录下新增 index.php 文件，内容如下：

```
<?php
// 全局自动加载
require __DIR__ . '/vendor/autoload.php';
// 调用类方法，打印返回值字符串的长度
var_dump(strlen(wangjialin\http\Http::requestByGet('http://github.com')));
```

在浏览器中访问 index.php 文件，输出如下结果则说明类库功能调试成功：

```
D:\phpStudy\WWW\http\index.php:11:int 50680 #展示响应字符串的长度
```

## 6.3.2 提交依赖包到 Composer Packagist

本地类库开发完毕后，需要分几步操作上线到 Composer 的 Packagist 中去，下面讲解相关的步骤。

### 1. 推送本地代码到GitHub

推送本地代码到线上仓库的操作，这里不再赘述。

## 2. 提交到Packagist

提交自己的 Composer 包到 Packagist，主要有以下几个步骤。

(1) 注册 Packagist 账号并登录

访问地址：<https://packagist.org/register>，实现注册和登录操作，不过也可以使用第三方，如 GitHub 账号实现授权登录，简化注册流程。

(2) 提交包的 GitHub 仓库地址

访问地址：<https://packagist.org/packages/submit>，在 Repository URL (Git/Svn/Hg) 下方的输入框中，填写 Git 地址后单击 Check 按钮，如图 6-9 所示。

Packagist 会检测包是否和已有应用重名，没有重名就可以继续提交，结果如图 6-10 所示。



图 6-9 填写包源码托管的 GitHub 地址

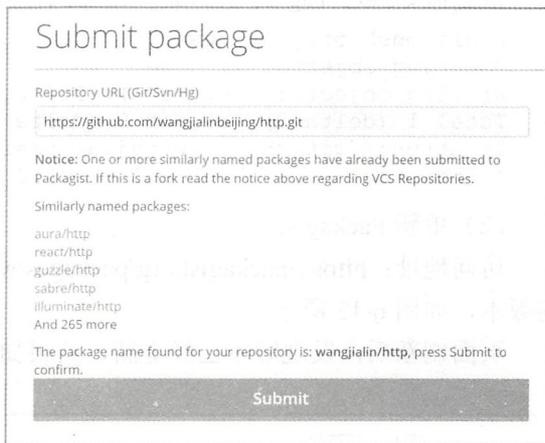


图 6-10 检测成功后进行提交操作

提交成功后，效果如图 6-11 所示。

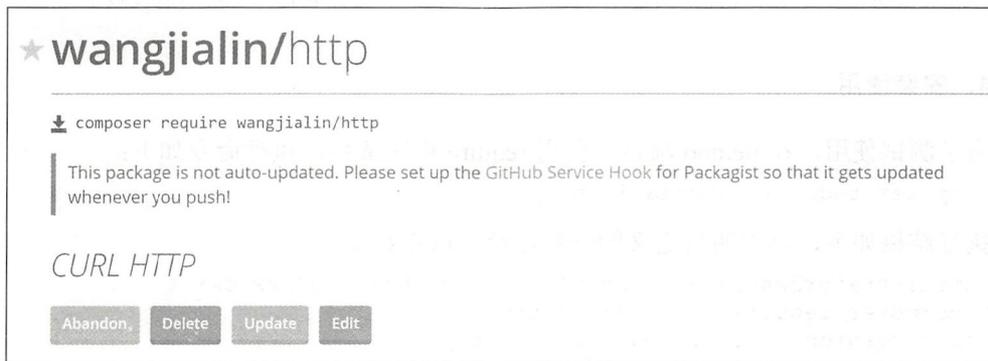


图 6-11 成功提交后查看依赖包的详情

### 3. 发布正式版本

当 Composer 获取测试版本 (dev) 进行依赖包安装时, 会自动同步 GitHub 上最新的代码, 但稳定版 (stable) 的发布需要在 GitHub 上添加相应的标签 (Tag)。下面简单讲解操作步骤。

(1) 本地创建 Tag 并推送到版本库

执行以下命令创建 Tag:

```
git tag -a v1.0.0 -m='v1.0.0'
```

执行相应命令查看 Tag 是否已添加, 显示如下结果说明 Tag 已经添加成功:

```
$ git tag
v1.0.0
```

执行命令推送 Tag 到 GitHub, 相应执行效果如下:

```
$ git push origin --tags
Counting objects: 1, done.
Writing objects: 100% (1/1), 164 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To https://github.com/wangjialinbeijing/http.git
* [new tag]          v1.0.0 -> v1.0.0
```

(2) 更新 Packagist

访问地址: <https://packagist.org/packages/wangjialin/http>, 单击 Update 按钮可以更新包的版本, 如图 6-12 所示。

页面刷新后, 发现版本已经更新, 结果如图 6-13 所示。



图 6-12 执行更新包版本操作

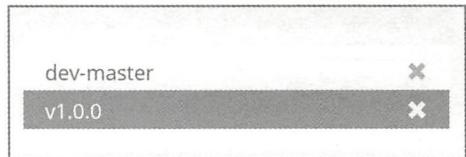


图 6-13 更新后的包版本

### 4. 安装使用

为了测试使用, 在 demo6 项目中使用 require 声明依赖, 执行命令如下:

```
composer require wangjialin/http
```

执行结果如下, 则说明自定义的依赖已经可以正常安装。

```
Administrator@wangjialin-pc MINGW64 /d/phpStudy/www/demo6
$ composer require wangjialin/http
Using version ^1.0 for wangjialin/http
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
```

```
Package operations: 1 install, 0 updates, 0 removals
- Installing wangjialin/http (v1.0.0): Loading from cache
Writing lock file
Generating autoload files
```

在 `index.php` 脚本文件中，注释 PHPMailer 类打印的代码，并增加以下代码：

```
// 在浏览器中打印实例结构
//var_dump($mailer);
// 使用 http 类的请求方法, 查看返回字符串长度
var_dump(strlen(wangjialin\http\Http::requestByGet('http://github.com')));
```

执行结果如下，则说明依赖工具已经可以正常使用。

项目入口文件

```
D:\phpStudy\WWW\demo6\index.php:12:int 49664
```

# 第2篇

## 框架进阶篇

- ▶▶ 第7章 响应式布局框架——Bootstrap
- ▶▶ 第8章 ThinkPHP 命令行操作与接口开发实战

# 第 7 章 响应式布局框架——Bootstrap

随着移动设备的广泛使用，越来越多的用户开始使用手机或平板电脑浏览网页。而针对 PC 编写的网页，若不对这些设备进行适配，则无法满足正常的用户使用体验。传统的移动页面适配，都是采取移动端、PC 端页面各自开发的方式，不仅费时、费力，后期维护的成本也非常巨大。

那么是否有办法解决这类问题呢？基于响应式布局的前端框架，就可以满足这类需求。基于框架技术，开发者通常只需要编写一套代码，框架就可以根据设备自动适配。本章将通过介绍响应式布局框架——Bootstrap，让开发者快速掌握前端开发技巧。

## 7.1 Bootstrap 入门

基于响应式布局的框架有很多，国外的如 Bootstrap，国内的如 AmazeUI，都可以满足多终端开发的需求。

### 7.1.1 Bootstrap 简介

一般后端开发者，只需要关注如何为客户端（包含 CS/BS 架构）提供服务即可。但在实际工作中情况较复杂，后端开发者有时也需要参与到前端页面的开发中。下面就列举了一些常见的工作场景需求。

- 项目 A 需要开发一套超级管理员后台界面。
- 项目 B 需要一个接口文档展示页面。
- 项目 C 需要一个简单的活动页。
- 实现个人开源项目的样式。
- 实现公司内部的统计平台页面。

针对以上需求，后端开发者需要一个拥有以下优点的前端框架，辅助其进行页面开发。

- 上手快。就算开发者以前没使用过，但框架拥有友好的文档说明，为开发者深入学习提供了便利。
- 组件多。工作中需要的样式都有，无论是表单、表格，还是按钮、导航，拿来就用。
- 多终端适配。只需要编写一套代码，就可以实现手机、平板电脑和 PC 网页的自动适配。

- 简洁美观。不要求满足所有用户的审美，但最起码要满足当前流行的网页审美。

虽然以上要求挺多，但 Bootstrap 框架的确可以满足以上需求。Bootstrap 是最受欢迎的 HTML、CSS 和 JavaScript 框架之一，用于开发响应式布局、移动设备优先的 Web 项目，无论前、后端开发者，都可以快速上手。Bootstrap 基本上可以满足所有类型的项目，其标志如图 7-1 所示。

除了上面提到的优点外，Bootstrap 还有以下几个特点。

- 预处理脚本。虽然可以直接使用 Bootstrap 提供的 CSS 预定义样式表，不过 Bootstrap 的源码是基于最流行的 CSS 预处理脚本——Less 和 Sass 开发。开发者可以采用预编译的 CSS 文件快速开发，也可以从源码定制自己需要的样式。
- 一个框架、多种设备。网站和应用能在 Bootstrap 的帮助下通过同一份代码快速、有效适配手机、平板电脑、PC 设备，实现响应式布局。
- 齐全的预定义样式。Bootstrap 提供了全面、美观的文档。在这里可以找到最常见的 HTML 元素、HTML 和 CSS 组件、jQuery 插件，并有详细的文档说明。



图 7-1 Bootstrap 的标志

## 7.1.2 Bootstrap 核心技术——CSS 预处理脚本

Bootstrap 使用了 CSS 预处理脚本技术。其核心思想是：使用专门的编程技术，为 CSS 增加了编程语言的特性，将 CSS 作为目标生成文件，开发者就只需要使用这种语言进行编码工作即可。

通俗地说，CSS 预处理脚本技术用一种专门的编程语言，进行 Web 页面样式设计，然后再编译成正常的 CSS 文件，以供项目使用。传统 CSS 样式表与预处理脚本处理流程的区别，如图 7-2 所示。

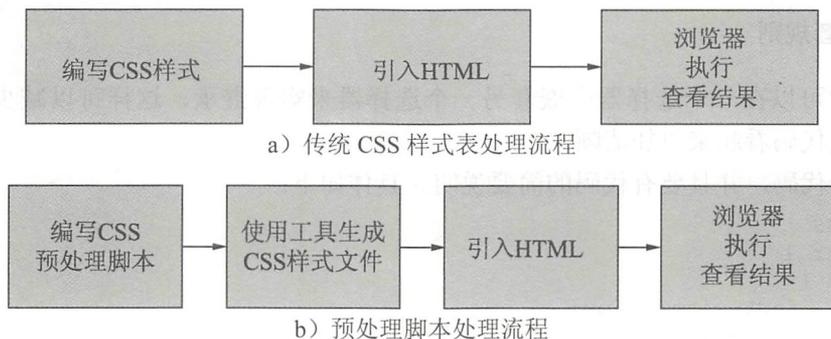


图 7-2 两种不同的处理流程

使用传统方式编写 CSS 样式表，在版本迭代开发时，随着代码量的逐步提升，批量修改会变得十分烦琐，此时引入编程语言里面的变量、逻辑判断和函数方法的概念，可以减

少重复工作。此外，类似继承概念的引入，也可以减少开发者的工作量。这种技术被称为预处理脚本技术，虽然多一个步骤，但实际上却帮了开发者大忙。

 **提示：**编写预处理脚本需要读者有编程基础，有一定的入门门槛。

目前，CSS 预处理器脚本已经非常成熟，涌现出了很多种不同的 CSS 预处理器语言。主流的预处理脚本语言有 Less、Sass 和 Stylus 等，因为 Bootstrap 内置了 Less，这里简单讲解下 Less 的相关特性，以方便理解。

## 1. 使用变量定义样式

Less 的安装和初始化这里不再赘述，读者可访问中文网站 <http://www.bootcss.com/p/lesscss>，查看相关教程。变量的使用，可以允许开发者单独定义一系列的样式，然后在需要的时候去调用。所以在做全局样式调整的时候，开发者可能只需要修改几行代码就可以了。典型的变量使用如下：

```
// LESS
@color: #4D926F;
#header {
    color: @color;
}
h2 {
    color: @color;
}
/* 生成的 CSS */
#header {
    color: #4D926F;
}
h2 {
    color: #4D926F;
}
```

## 2. 嵌套规则

开发者可以在一个选择器中嵌套另一个选择器来实现继承，这样可以减少代码编写量，并且使代码看起来更加清晰。

要引出代码，并且要有代码的简要说明。具体如下：

```
// LESS
#header {
    h1 {
        font-size: 26px;
        font-weight: bold;
    }
    p { font-size: 12px;
        a { text-decoration: none;
            &:hover { border-width: 1px }
        }
    }
}
```

```

}
/* 生成的 CSS */
#header h1 {
    font-size: 26px;
    font-weight: bold;
}
#header p {
    font-size: 12px;
}
#header p a {
    text-decoration: none;
}
#header p a:hover {
    border-width: 1px;
}

```

编写标准的 CSS 样式表中，样式的嵌套非常烦琐，每一级嵌套都要带上父级名称，代码如下：

```

#header p a {
    text-decoration: none;
}
#header p a:hover {
    border-width: 1px;
}

```

而使用 Less 父级元素，只需要在外层定义一次即可，代码如下：

```

#header {
    h1 {
        font-size: 26px;
        font-weight: bold;
    }
    .....
}

```

### 3. 函数和运算

Less 的运算提供了加、减、乘、除操作，开发者可以做属性值和颜色的运算，这样就可以实现属性值之间的复杂关系管理。Less 中的函数，实际上映射了 JavaScript 脚本代码方法，开发者可以对属性进行独立控制，示例代码如下：

```

// LESS
@the-border: 1px;
@base-color: #111;
@red: #842210;
#header {
    color: @base-color * 3;
    border-left: @the-border;
    border-right: @the-border * 2;
}
#footer {
    color: @base-color + #003300;
    border-color: desaturate(@red, 10%);
}
/* 生成的 CSS */

```

```
#header {
  color: #333;
  border-left: 1px;
  border-right: 2px;
}
#footer {
  color: #114411;
  border-color: #7d2717;
}
```

 **提示：**这里只是简单讲解了 Less 使用特性，让开发者知道预处理脚本的优势，如想继续深入学习可以参考官方文档。

### 7.1.3 响应式布局技术

响应式布局技术的原理，就是根据设备显示宽度，自动调整元素的显示布局。

在页面实现上，不需要针对不同的页面宽度，编写专门的处理代码，只需要一次开发，就可以自动适配，这也是响应式布局技术最大的优势。

这里以 Bootstrap 中文网（访问地址 <http://www.bootcss.com/>）为例，根据用户不同的访问宽度，来演示什么是响应式布局。

#### 1. PC的浏览器宽度

常见的 PC 浏览器宽度在 1377 像素到 1920 像素之间。访问 Bootstrap 中文网，确保浏览器保持最大宽度，此时的显示效果如图 7-3 所示。可以看到页面布局分别由：导航条、页面头部和页面列表三部分组成，因为页面宽度足够，因此导航条直接展示菜单项，页面列表每行展示四个元素。



图 7-3 PC 浏览器访问宽度

## 2. 平板设备（平板电脑）的浏览器宽度

平板设备的逻辑显示宽度，一般介于手机与 PC 之间。在 Chrome 浏览器中使用设备模拟调试功能，查看 Bootstrap 中文网，在平板设备浏览器宽度下的访问效果（以 iPad 设备型号为例），如图 7-4 所示。



图 7-4 平板设备浏览器访问宽度

在平板设备模式下，可以看到布局减少了导航条元素（“关于”按钮自动隐藏，参见图 7-3），Bootstrap 把页面列表横向展示数量减少为两个，而非等比缩小，此时用户看到的元素大小与 PC 访问效果类似，并不会因为元素在更窄屏幕下访问，而强制缩小元素导致用法无法看清。

## 3. 手机设备的浏览器宽度

手机类设备浏览器访问宽度一般较低，在浏览器模拟手机设备访问后，可以看到导航条菜单已经完全隐藏，用户通过单击右侧交互按钮，可以进行菜单选择。页面列表中每行的元素也自动变为一个，充分根据手机特性进行适配，效果如图 7-5 所示。

通过实例可以看出，响应式布局在不同的页面大小下，会自动地改变页面的布局。实现了一次编写，达到多端适应的效果。

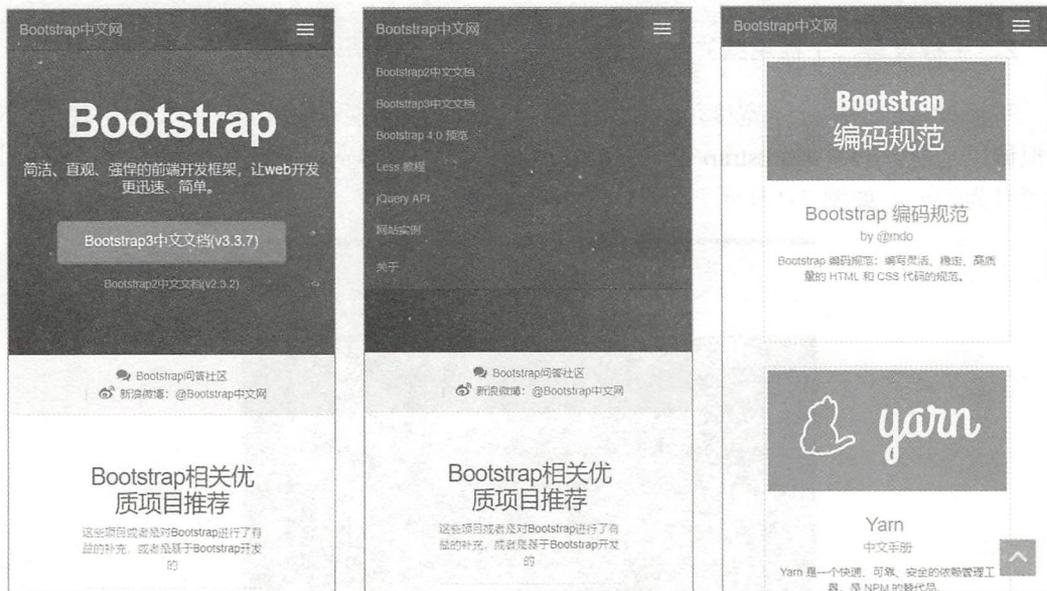


图 7-5 手机浏览器访问宽度

提示：响应式布局虽然有种种优势，但不是所有的页面都适合这样的布局方式，还有很多其他的布局方式，如自适应布局等，可以满足不同的页面需求。

## 7.2 Bootstrap 实战技巧

本节不再罗列 Bootstrap 的详细组件样式，只讲解在学习此类框架时，开发者需要注意的重点和技巧，这样读者遇到同类框架时，学习过程会更高效。

### 7.2.1 布局容器与栅格系统

实现各式各样的页面布局，开发者需要结合盒模型、内外边距和 div 容器等各种方式，在满足需求的同时，编写大量的业务代码，这不仅耗费时间，还会因为多终端、跨浏览器等问题，导致开发效率低下。Bootstrap 框架内置了布局容器与栅格系统，开发者只需像搭积木一样，用最少的代码最快实现效果。

Bootstrap 的安装与基础使用在这里不再赘述，这里直接介绍 Bootstrap 的布局容器与栅格系统的原理。

#### 1. 布局容器

Bootstrap 框架提供了两个预定义样式来实现最外层的包裹效果。

### (1) 固定布局容器

使用预定义的样式“.container”结合div元素，可以实现一个固定宽度并且支持响应式布局的容器，实例代码如下：

```
<div class="container">
.....
</div>
```

### (2) 全窗口布局容器

使用预定义的样式“.container-fluid”结合div元素，容器宽度为100%，实现窗口的布局容器。容器可以根据页面的宽度自适应，以保证全部覆盖显示：

```
<div class="container-fluid">
.....
</div>
```

## 2. 栅格系统

Bootstrap提供了一套响应式、移动设备优先的流式栅格系统，随着屏幕或浏览器窗口尺寸的增加，系统会自动分为最多12列。而布局都是通过预定义的样式类来操作，开发者可以组合出所需的布局方式。

一个典型的栅格布局结构，如图7-6所示。

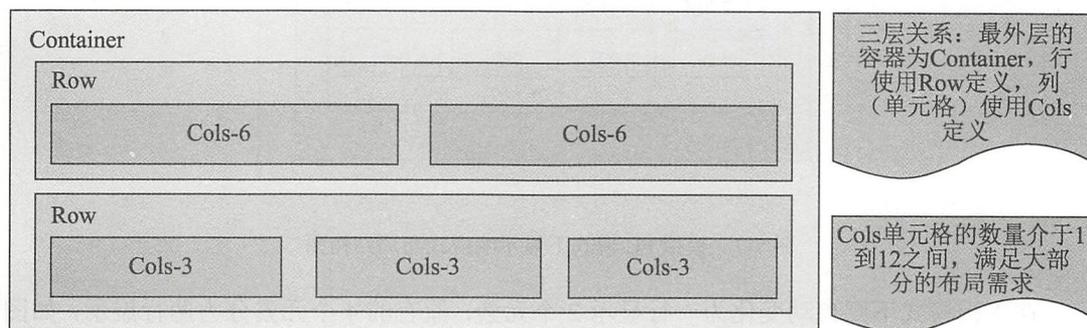


图 7-6 Bootstrap 典型的栅格系统

栅格系统有点类似表格布局，但可以实现的功能更强大。使用栅格布局需要注意以下几点：

- 预定义样式 container/container-fluid 外层布局样式。
- 行（row）定义布局的上下层级关系，可以理解为表格 table 元素中的 tr 元素。
- 列（cols）定义容器在行（row）内的布局关系，可以理解为 tr 元素中的 td 单元格元素。
- 行（row）没有数量限制，列（cols）最少 1 个，最多 12 个。

了解栅格系统的定义后，在实际使用前还需要掌握 Bootstrap 提供的预定义类，才能知道如何定义行（row）、列（cols）和实现响应式布局等操作。根据显示设备宽度不同，

搭配不同的行（cols）预定义样式可以实现不同的响应式布局效果，指定布局在什么宽度下进行自动变化。各个预定义类对应的宽度如表 7-1 所示。

表 7-1 Bootstrap 栅格系统响应式布局预定义样式类

样式属性说明	手机屏幕 (<768px)	平板屏幕 (>=768px)	PC 屏幕 (>=992px)	超大屏幕 (>=1200px)
栅格系统行为	水平排列	堆叠排列，超过阈值后水平排列	堆叠排列，超过阈值后水平排列	堆叠排列，超过阈值后水平排列
容器最大宽度	自动	750px	970px	1170px
类前缀	.col-xs-	.col-sm-	.col-md-	.col-lg-
最大列数量	12	12	12	12
最大列宽度	自动	62px左右	81px左右	97px左右
列间距宽度	30px（左右各15px）	30px（左右各15px）	30px（左右各15px）	30px（左右各15px）

为了更好地理解，下面通过模拟实际需求编写一个实例。某产品的首页列表需专门对平板模式和 PC 模式进行适配。PC 模式布局设计如图 7-7 所示，一行显示 4 个元素。

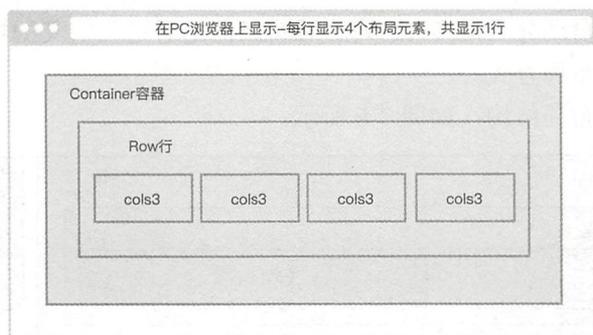


图 7-7 模拟 PC 模式下显示响应式布局结构图

在平板模式下则自动变化为一行显示 2 个元素，原有的 4 个元素分为两行展示，如图 7-8 所示。

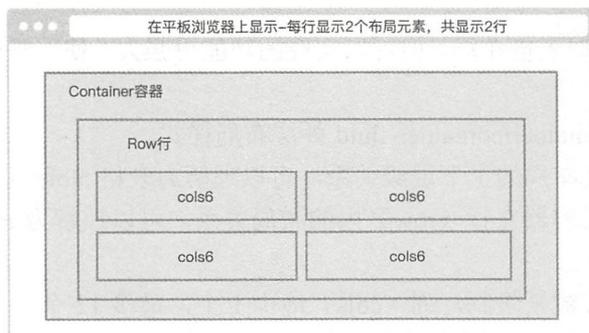


图 7-8 模拟平板模式下显示响应式布局结果图

部分代码实现如下：

```
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <h1>PC/平板响应式布局展示</h1>
    </div>
  </div>
  <div class="row layout">
    <div class="col-md-3 col-sm-6">第一部分</div>
    <div class="col-md-3 col-sm-6">第二部分</div>
    <div class="col-md-3 col-sm-6">第三部分</div>
    <div class="col-md-3 col-sm-6">第四部分</div>
  </div>
</div>
```

在 PC 模式和平板模式下分别访问同一页面，效果如图 7-9 所示。

## PC/平板响应式布局展示



## PC/平板响应式布局展示



图 7-9 不同设备上的响应式布局效果

提示：开发者可以自行尝试，在手机页面布局条件下，本实例的展示效果。

## 7.2.2 Bootstrap 组件快速入门技巧

Bootstrap 框架自带的基础样式和预定义样式组件十分丰富，短时间内难以全部熟悉，不过这里提供一些使用上的技巧，不仅可以帮助读者快速入门，还可以提升学习的效率。

### 1. 掌握预定义样式规律

布局与展示类效果都是由预定义的 CSS 样式统一实现，通过观察可以发现一定的规律。以按钮样式为例，定义一个默认样式的按钮，代码通常如下：

```
<a class='btn btn-default' href='#'>这是一个普通按钮</a>
```

而定义一个成功样式的按钮时代码如下：

```
<a class='btn btn-success' href='#'>这是一个成功样式按钮</a>
```

在此基础上，再对按钮的大小进行调整，比如加大成功样式按钮的大小，只需要继续追加预定义的样式：

```
<a class='btn btn-success' href='#'>这是一个成功样式的大按钮</a>
```

通过以上按钮的定义，发现 Bootstrap 的样式定义具有包含关系：

- 样式类 `btn` 是按钮的基础样式。
- 样式类 `btn-default` 是按钮的展示类型样式。
- 样式类 `btn-lg` 是按钮的大小类型样式。

上面几种样式中，除了基础样式必须定义外，其他几个样式完全可以根据需求灵活组合，实现需要的效果。

不仅是按钮的预定义样式规律如此，其他的基础样式也大多如此。例如，表格相关的样式，也拥有一个基本的 `table` 样式，其他效果可在基础样式上继续追加。

例如，需要一个带边框的表格，代码如下：

```
<table class="table table-bordered">
.....
</table>
```

## 2. 借助开发工具自动提示

除了通过官方手册查询 Bootstrap 都有哪些预定义样式外，还可以借助开发工具的自动提示功能进行联想提示。因为在 Bootstrap 中，样式一般都是按照功能性来定义，如成功样式的按钮 `btn-success`、带边框的表格 `table-bordered`，开发者在见到这些样式定义的时候，基本上可以猜到对应的效果。

这里以 PHPStorm 开发工具为例，展示自动提示所带来的便捷性。

(1) 创建一个 `auto.html` 页面，引入 Bootstrap 框架，增加一个 `table` 元素，定义 `class` 属性时会自动提示关联样式，如图 7-10 所示。

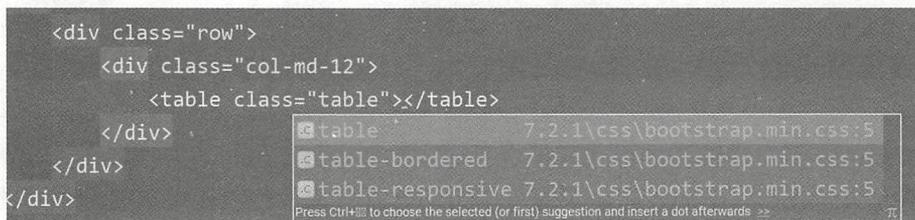


图 7-10 使用 IDE 快速查找预定义样式——表格相关样式

(2) 在页面上定义一个超链接，定义 `class` 属性时，自动提示关联样式，如图 7-11 所示。

 **提示：**自动提示可行的前提条件是：Bootstrap 及其相关的 CSS、JavaScript 类库文件必须在本地引入。

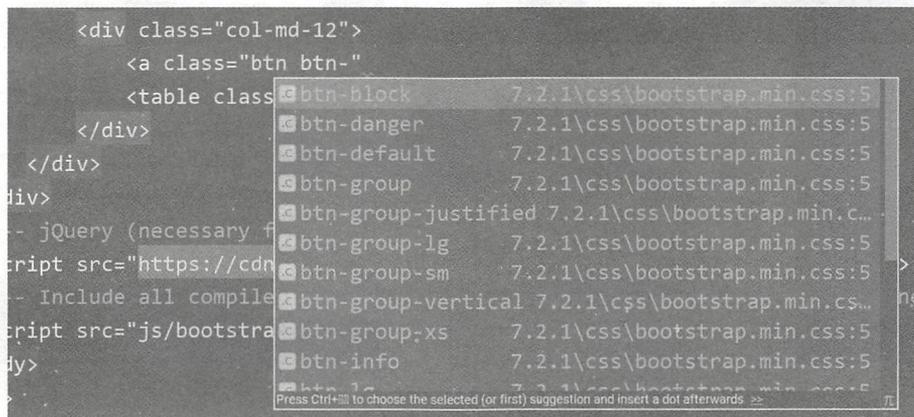


图 7-11 使用 IDE 快速查找预定义样式——按钮相关样式

## 7.3 基于 Bootstrap 的后台模板样式——AdminLTE

虽然 Bootstrap 本身拥有诸多优势，但它是一个通用的前端框架，未对特殊需求进行定制。例如，开发管理后台相关页面，使用 Bootstrap 框架就要从头写起，虽然各个组件都提供了，但是整合到一起也是不小的工作量。本节重点讲解基于 Bootstrap 的后台模板样式 AdminLTE，掌握如何快速开发后台样式。

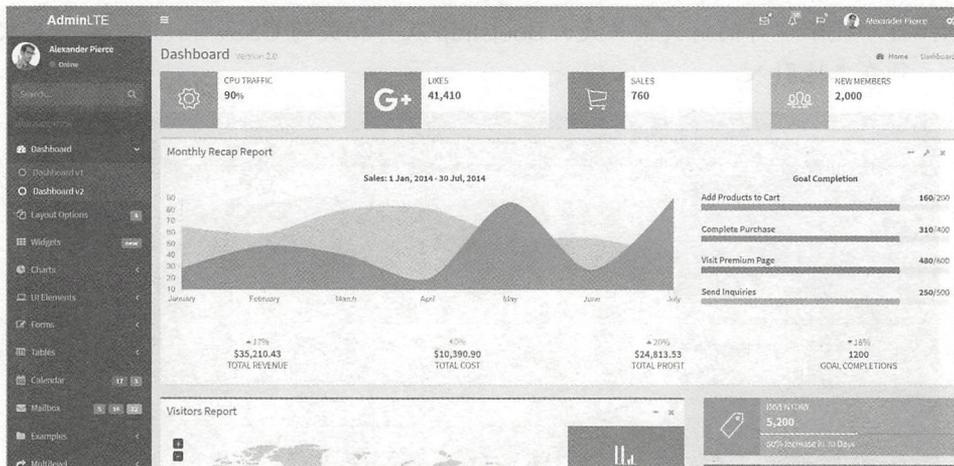
### 7.3.1 AdminLTE 简介与安装

开发者实现完整的后台管理界面，需要编写大量的样式代码。为了减少工作量，此时可以选择开源的后台模板样式——AdminLTE。

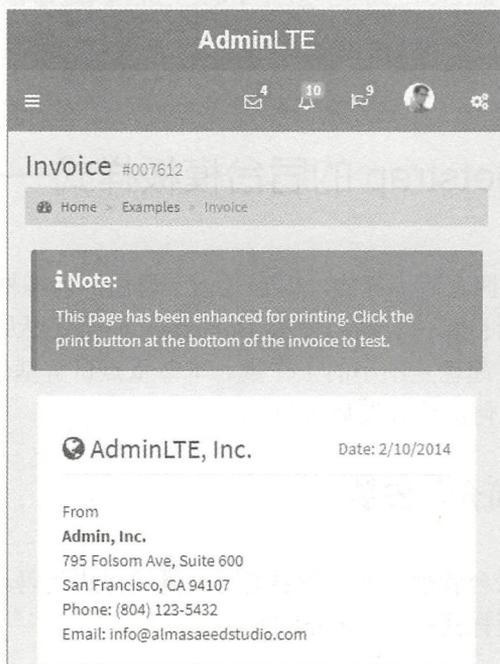
#### 1. 简介

AdminLTE 是一套基于 Bootstrap 框架的后台模板样式，不仅提供了登录注册、用户中心、表单表格和页面提示等常用模板样式组件，还提供了像动态导航、下拉菜单和表格无刷新排序等 JavaScript 组件，方便用户直接调用。

AdminLTE 拥有完整的 Bootstrap 框架特性，对于了解 Bootstrap 框架的开发者来说，基本上都可以快速入门。AdminLTE 的整体样式风格如图 7-12 所示。



a)



b)

图 7-12 AdminLTE 提供了丰富的响应式样式风格

通过图 7-12 可以看出，AdminLTE 的布局风格是典型的管理后台样式，模板不仅基于 Bootstrap 的栅格布局，可以自动适配移动端，还提供了大量的预定义组件样式，如单独的用户信息展示、各类统计图表、导航布局和丰富的菜单等，可以满足大部分的后台管理样式需求。

不过一般的后台管理页面并不需要这么多功能，导航、菜单和主体页面（包含列表、

表单等)即可满足需求。AdminLTE 主体布局结构图如图 7-13 所示。



图 7-13 AdminLTE 主体布局结构图

导航和菜单，可以直接使用 AdminLTE 的样式。同样，在实际项目开发中，主体内容最常用的就是表单和列表样式，在 AdminLTE 中表单和列表如图 7-14 所示。

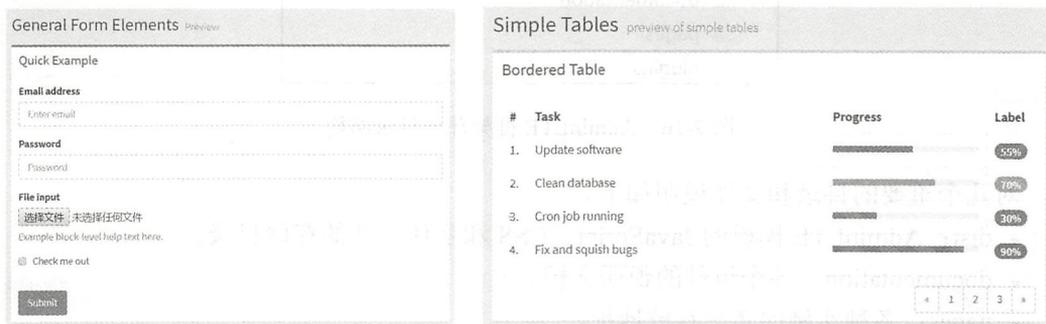


图 7-14 AdminLTE 中的表单和列表

**提示：**AdminLTE 中包含了大量的基础样式与组件，想了解更多内容可以参考官方手册和模板源代码。

## 2. 框架安装

访问地址 <https://github.com/almasaeed2010/AdminLTE/releases>，可以直接下载 AdminLTE 的源码。根据需要选择最新的版本（本实例使用 AdminLTE 2.4.2 版本）下载，如图 7-15 所示。

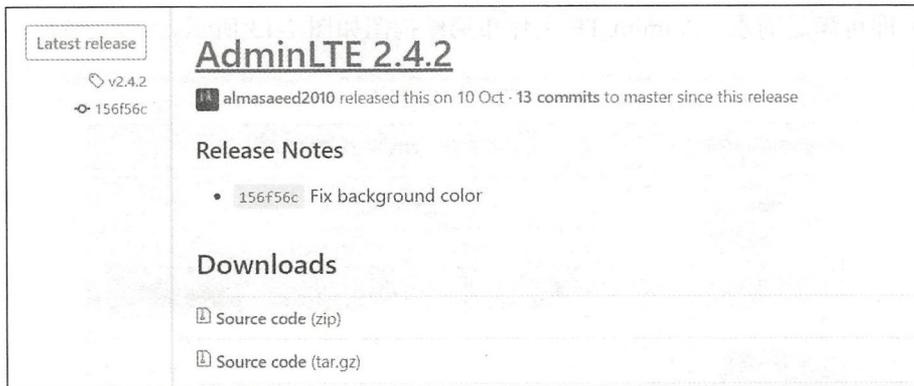


图 7-15 获取最新的 AdminLTE 源代码

下载成功后，解压缩源代码到本地，为了方便演示，这里修改解压完成的框架源码目录名为 `admin7`，作为后面几节演示的实例项目。

通过 `admin7` 目录，查看 AdminLTE 源代码，发现内容较多，如图 7-16 所示。

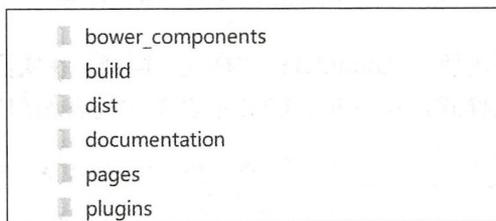


图 7-16 AdminLTE 框架部分目录结构

对几个重要的目录和文件说明如下。

- `dist`: AdminLTE 模板的 JavaScript、CSS 和字体文件的存储目录。
- `documentation`: 各个组件的说明文档。
- `pages`: 各种实例的实际存放地址。
- `plugins`: 第三方 JavaScript 插件的实际存放地址。
- `starter.html`: 快速开始使用模板文件。
- `index.html`: 官方实例展示入口文件。

**提示：**本章中的 AdminLTE 实例，不依赖 PHP 的运行环境，可以直接在操作系统文件目录下访问执行。

### 7.3.2 布局、皮肤与 box 容器

无论学习哪个前端框架，首先都要学习框架本身的布局风格。



AdminLTE 本身继承了 Bootstrap 的栅格布局,并在此基础上扩展了基础布局和 box 容器。

## 1. 布局与皮肤

AdminLTE 官方源码提供了 starter.html 的示例文件,开发者可以通过此模板文件,了解 AdminLTE 的基本页面结构。这里以 7.3.1 节中的 admin7 实例项目为例,访问 starter.html 文件,浏览器访问效果如图 7-17 所示。

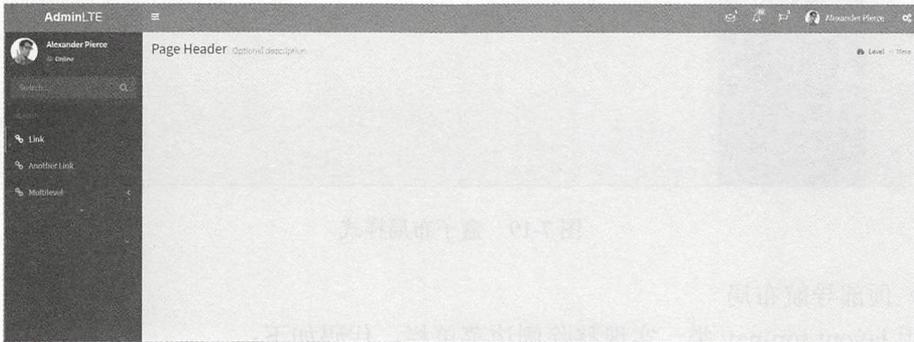


图 7-17 AdminLTE 提供的快速开始示例

AdminLTE 的布局主要由以下 4 部分组成。

- Wrapper(.wrapper): 一个 div,用来包裹整个站点。
- Main Header(.main-header): 包含 logo 和导航条。
- Sidebar(.sidebar-wrapper): 包含用户面板和 sidebar 菜单。
- Content(.content-wrapper): 包含页面头部和内容。

AdminLTE 为这些布局类提供了一些额外组合样式。下面列表中的每个类都可以添加到 body 标签上,方便自定义效果。为了方便演示,在 admin7 项目根目录下新增 layout.html 文件。

### (1) 固定布局

使用 fixed 类,定义一个固定导航条和菜单的布局,在 body 标签上定义,代码如下:

```
<body class="hold-transition skin-blue fixed">
```

显示效果如图 7-18 所示。

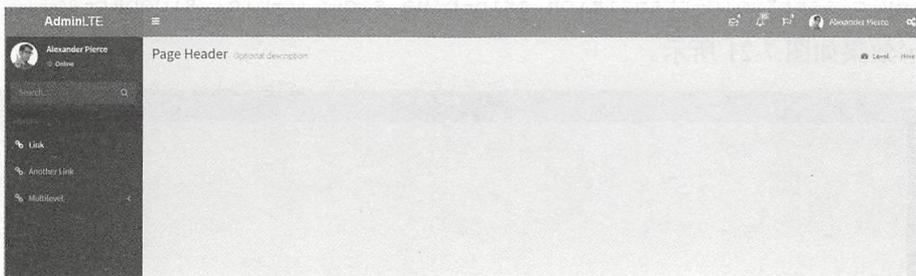


图 7-18 固定导航条、菜单样式



## (2) 盒子布局

使用 `layout-boxed` 类，定义一个盒子布局效果，宽度最大为 1250px，代码如下：

```
<body class="hold-transition skin-blue layout-boxed">
```

显示效果如图 7-19 所示。

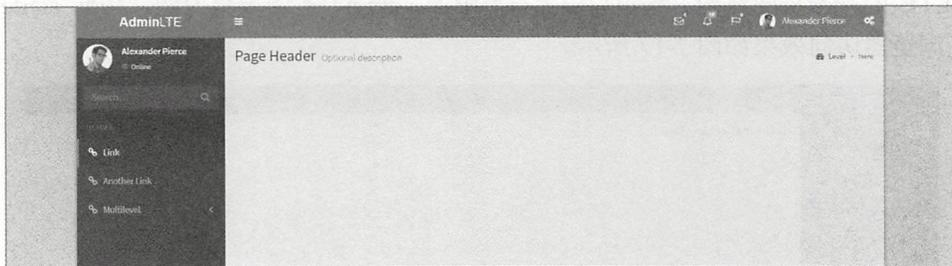


图 7-19 盒子布局样式

## (3) 顶部导航布局

使用 `layout-top-nav` 类，实现移除侧边菜单栏，代码如下：

```
<body class="hold-transition skin-blue layout-top-nav">
```

显示效果如图 7-20 所示。

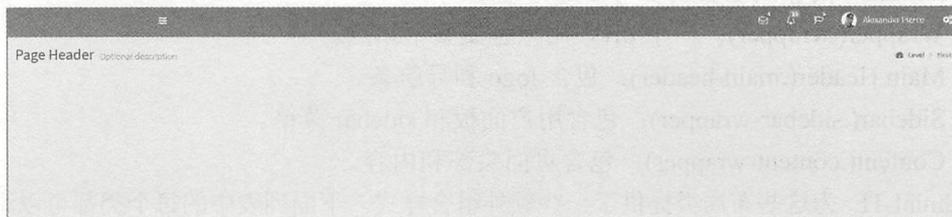


图 7-20 顶部导航样式

## (4) 收敛侧边栏布局

使用 `sidebar-collapse` 和 `sidebar-mini` 样式，可以实现一个收敛侧边栏效果，在拥有菜单功能的同时，让主体显示面积更大，代码如下：

```
<body class="hold-transition skin-blue sidebar-mini sidebar-collapse">
```

显示效果如图 7-21 所示。

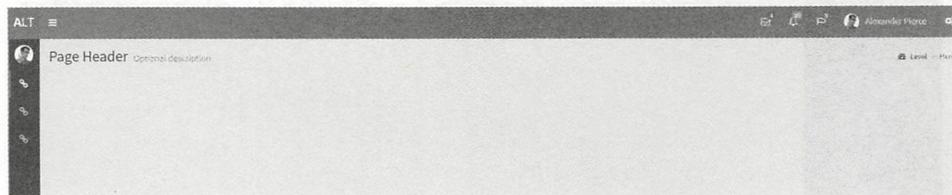


图 7-21 收敛侧边栏效果



**提示：**不能同时使用 `layout-boxed` 和 `fixed` 样式，其他样式可以混合使用。

修改皮肤颜色的操作方法类似，只需要引入对应的样式文件，随后在 `body` 标签上定义类即可。AdminLTE 默认提供了多种颜色的主题，包含蓝色 (`skin-blue`)、黑色 (`skin-black`)、紫色 (`skin-purple`)、黄色 (`skin-yellow`)、红色 (`skin-red`) 与绿色 (`skin-green`)。系统默认使用的是蓝色主题，这里以换为黄色主题为例，操作步骤如下。

(1) 引入黄色皮肤主题的样式表。代码修改如下：

```
<!--<link rel="stylesheet" href="dist/css/skins/skin-blue.min.css">-->
<link rel="stylesheet" href="dist/css/skins/skin-yellow.min.css">
```

(2) 在 `body` 标签上定义类。代码修改如下：

```
<body class="hold-transition skin-yellow sidebar-mini">
```

刷新浏览器，查看效果如图 7-22 所示。

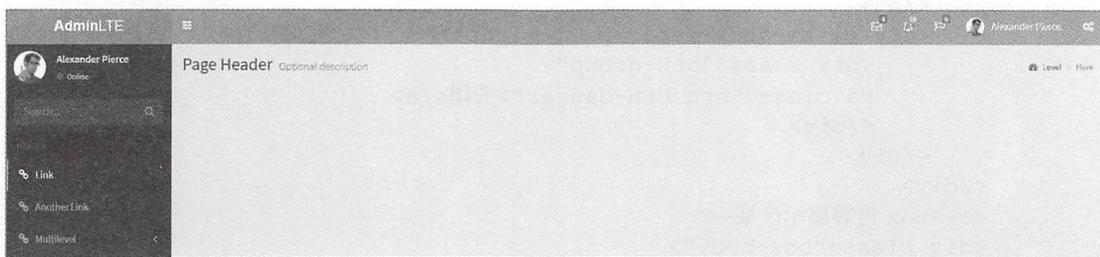


图 7-22 更换主题颜色为黄色

**提示：**更换其他颜色的模板皮肤操作类似，这里不再赘述。

## 2. box 容器

相比 Bootstrap 自带的样式，AdminLTE 增加了 `box` 样式定义布局容器。常见的 `box` 布局样式如图 7-23 所示。

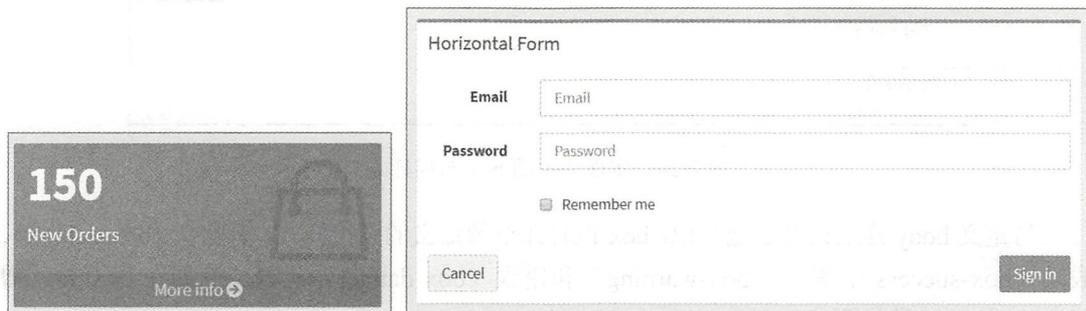


图 7-23 AdminLTE 中常见的 `box` 布局样式



AdminLTE 模板中的 box 布局样式，有以下几类。

### (1) box 相关样式

此类预定义的 box 样式，布局结构上分头部区域、主体区域和底部区域，对应的 class 分别为 box-header、box-body 和 box-footer。在头部区域中，又分为标题和操作区域（如关闭按钮），对应的 class 分别为 box-title 和 box-tools。以 admin7 项目为例，在根目录下创建一个 box.html 文件，初始化内容与 starter.html 一致。随后定义一个标准的 box，代码如下：

```
<div class="box box-success box-solid">
  <!--box 头部区域-->
  <div class="box-header">
    <i class="fa fa-comments-o"></i>
    <h3 class="box-title">
      标题内容
    </h3>
    <div class="box-tools pull-right">
      <div class="btn-group">
        <a class="btn btn-danger">关闭</a>
      </div>
    </div>
  </div>
  <!--box 内容展示区域-->
  <div class="box-body">
    这里包含主体内容
  </div>
  <!--box 底部展示区域-->
  <div class="box-footer">
    底部说明
  </div>
</div>
```

在浏览器中访问，如图 7-24 所示。

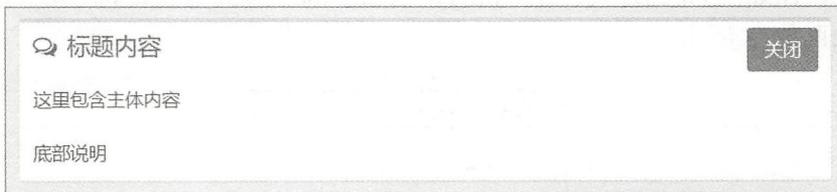


图 7-24 实现基本的 box 布局样式

与定义 body 样式切换主题类似，box 也有几个预定义的风格样式，有默认（box-default）、成功（box-success）、警告（box-warning）和错误（box-danger）几种。例如在 box 类后增加 box-success 类如下：

```
<div class="box box-success">
.....
```



效果如图 7-25 所示。

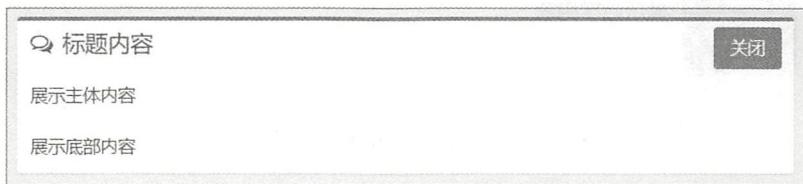


图 7-25 修改 box 的显示风格

也可以使用 `box-solid` 样式，控制 box 头部颜色显示的范围（顶部线条展示或全部展示），代码如下：

```
<div class="box box-success box-solid">  
.....
```

效果如图 7-26 所示。

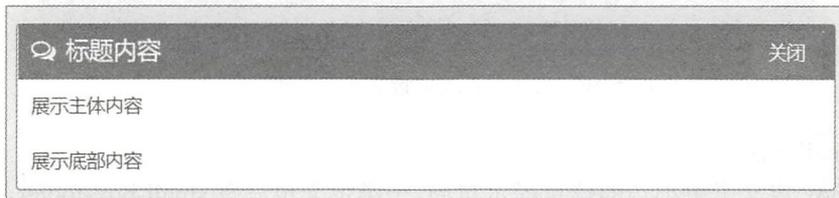


图 7-26 控制 box 头部颜色显示范围

## (2) info-box

`info-box` 及其相关的类，一般用于定义消息提示类的布局。结构上包括左侧图片（`icon`）和右侧内容（`content`），对应的 class 分别为 `info-box-icon` 和 `info-box-content`。和 `box` 的头部区域类似，`info-box-content` 下，又可以添加 `info-box-text`、`info-box-number` 和 `info-box-more` 样式。

例如，一个典型的 `info-box` 实现代码如下：

```
<div class="info-box">  
  <!-- 左侧的带背景色的图标部分 -->  
  <span class="info-box-icon bg-aqua"><i class="fa fa-envelope-o">  
</i></span>  
  <!-- 内容展示部分 -->  
  <div class="info-box-content">  
    <span class="info-box-text">显示文字内容</span>  
    <span class="info-box-number">显示数字内容</span>  
    <span class="info-box-more pull-right">右侧更多</span>  
  </div>  
</div>
```

展示效果如图 7-27 所示。

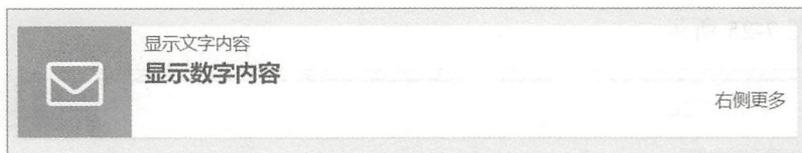


图 7-27 典型的 info-box 布局效果

若需要修改内容展示区域的背景颜色，只需要在 info-box 类后追加 bg-颜色类即可。例如修改背景颜色为绿色，代码如下：

```
<div class="info-box bg-green">
```

.....

展示效果如图 7-28 所示。



图 7-28 修改 info-box 展示内容的背景颜色

### (3) small-box

small-box 定义了更小巧的信息提示布局，子类定义也只有 small-box-footer，代码如下：

```
<!--整个 box 为绿色背景色-->  
<div class="small-box bg-green">  
  <!--要显示的文字-->  
  <div class="inner">  
    <h3>标题内容</h3>  
    <p>内容</p>  
  </div>  
  <!--可在右侧显示的背景-->  
  <div class="icon">  
    <i class="fa fa-shopping-cart"></i>  
  </div>  
  <!--更多链接及图标-->  
  <a href="#" class="small-box-footer">  
    更新 <i class="fa fa-arrow-circle-right" ></i>  
></a>  
</div>
```



图 7-29 典型的 small-box 布局效果

展示效果如图 7-29 所示。

## 7.3.3 实现常用模板——列表

后台管理页面中，列表和表单使用的频率很高，如订单列表、用户列表和新增商品等。本节开始尝试设计一套通用的列表、表单样式，为后面的内容开发系统打下样式基础。首

先来看如何实现一个通用的列表页面，结构示意图如图 7-30 所示。



图 7-30 常用列表布局结构示意图

通用的列表，一般由以下几个部分构成：

- 表单搜索，顶部一般会有关键字搜索、状态选择项等，可供用户数据检索。
- 主体列表，每行包含记录展示、状态展示和操作区域等，操作区域中可以执行页面重定向、状态迁移等功能。
- 分页，记录超过分页固定条数，展示分页操作区域。

实例最终完成的效果图，如图 7-31 所示。



图 7-31 定义通用列表样式效果完成图

下面根据实例步骤来演示列表的实现：

### (1) 新增列表模板页

以上面创建的 admin7 实例为例，新增 table.html 页面，并复制 starter.html 中的基础结构代码。

### (2) 主体内容布局

因为 AdminLTE 基于 Bootstrap 框架开发，其所带的预定义样式都可以使用。所以在 这里使用栅格布局定义表格列表的外部容器，在 content 类定义的 section 元素中，增加代码如下：

```
<!--表格开始-->
<div class="row">
  <!--使用 bootstrap 的全局样式 row-->
  <div class="col-md-12">
    <!--使用 bootstrap 的全局样式 col 控制 box 的宽度 100%-->
    <!--全局搜索区域-->
    <div class="box">
      <div class="box-header">
        <h3 class="box-title">高级检索</h3>
      </div>
      <div class="box-body">
        显示全局搜索区域
      </div>
    </div>
    <!--列表展示区域-->
    <div class="box">
      <div class="box-header">
        <h3 class="box-title">表格列表</h3>
      </div>
      <div class="box-body">
        显示表格列表
        <div class="row">
          <div class="col-md-6"></div>
          <div class="col-md-6" style="text-align: right;">
            显示分页区域
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
```

代码中使用 Bootstrap 的栅格布局定义外层容器，使用 AdminLTE 的 box 样式定义内层容器。在浏览器中访问 table.html 页面，显示效果如图 7-32 所示。

### (3) 搜索表单

在全局搜索区域中的 box-body 中增加表单项，实现搜索样式。核心代码如下：

```
<form class="form-inline">
```

```

<div class="form-group">
  <p class="form-control-static">搜索项</p>
</div>
<div class="form-group">
  <input type="text" class="form-control" placeholder="请输入关键字">
</div>
<div class="form-group">
  <p class="form-control-static">状态</p>
</div>
<div class="form-group">
  <select class="form-control">
    <option>正常</option>
  </select>
</div>
<button type="submit" class="btn btn-info">搜索</button>
<button type="submit" class="btn btn-warning">清除条件</button>
</form>

```

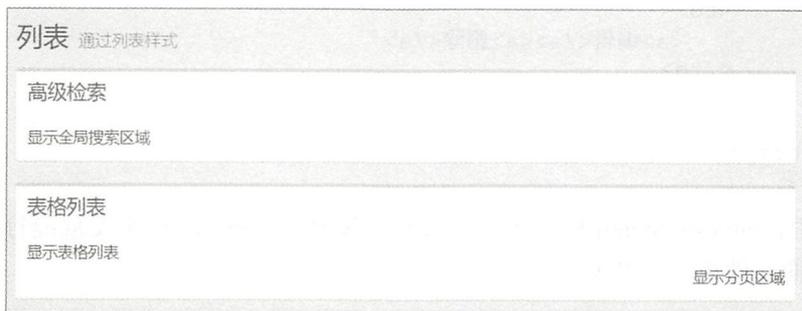


图 7-32 定义列表的布局容器

需要注意的是，搜索表单项使用了 Bootstrap 的表单相关样式，实现同行中显示多个表单元素。不过在实例中，只定义了单个的文本框和单选框，开发者根据自身需求可以灵活追加。展示效果如图 7-33 所示。

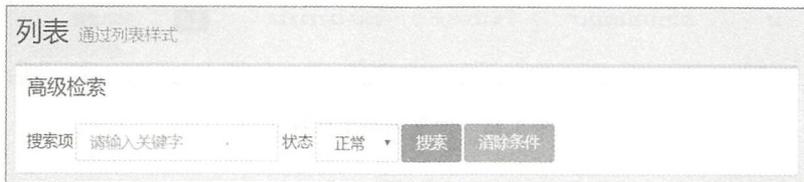


图 7-33 定义列表的全局搜索项

#### (4) 列表样式

使用 Bootstrap 自带的 table 相关预定义样式，定义列表。在列表展示的 box-body 布局容器中，增加以下代码：

```

<table class="table table-bordered table-stripe">
  <thead>

```

```

<tr>
  <th>列表 ID</th>
  <th>列表号</th>
  <th>列表值 1</th>
  <th>列表操作时间</th>
  <th>状态</th>
  <th>操作</th>
</tr>
</thead>
<tbody class="">
  <tr>
    <td>12</td>
    <td>201711121212121212</td>
    <td></td>
    <td>¥123.45 元</td>
    <td>2017-12-12 12:12</td>
    <td><span class="label label-success">正常</span></td>
    <td>
      <a>编辑</a><a>删除</a>
    </td>
  </tr>
  .....
</tbody>
</table>

```

在表格中，table-bordered 样式定义带边框的表格，table-stripe 则实现隔行换色。访问 table.html 页面，效果如图 7-34 所示。

列表ID	列表号	列表值1	列表操作时间	状态	操作
12	201711121212121212	¥123.45元	2017-12-12 12:12	禁用	编辑 删除
12	201711121212121212	¥123.45元	2017-12-12 12:12	正常	编辑 删除
12	201711121212121212	¥123.45元	2017-12-12 12:12	禁用	编辑 删除
12	201711121212121212	¥123.45元	2017-12-12 12:12	正常	编辑 删除

显示分页区域

图 7-34 定义列表主体样式

#### (5) 分页样式。

使用 Bootstrap 预定义的分页样式，替换代码中“显示分页区域”的文本，代码如下：

```

<nav aria-label="Page navigation" class="">
  <ul class="pagination">
    <li>
      <a href="#" aria-label="Previous">
        <span aria-hidden="true">上一页</span>
      </a>
    </li>

```

```

<li class="active"><a href="#">1</a></li>
<li><a href="#">2</a></li>
<li><a href="#">3</a></li>
<li><a href="#">4</a></li>
<li><a href="#">5</a></li>
<li>
  <a href="#" aria-label="Next">
    <span aria-hidden="true">下一页</span>
  </a>
</li>
</ul>
</nav>

```

查看页面效果，如图 7-35 所示。



图 7-35 定义列表分页样式

### 7.3.4 实现常用模板——表单

了解列表相关样式后，本节继续学习表单相关样式的定义。实际使用中，数据操作大多离不开表单。除了常用的文本框、选择项、单选和多选等基本样式外，其他样式如富文本编辑器、文件上传进度条和错误提示样式等也常常用到。

为了方便演示，继续使用 admin7 项目，新增 form.html 文件，复制 starter.html 内容为基本样式，实现一个通用表单。

#### (1) 表单布局

表单的布局与列表类似，通过栅格布局和 box 容器定义，在 content 定义的 div 中代码如下：

```

<!--使用 bootstrap 的全局样式 row-->
<div class="row">
  <!--使用 bootstrap 的全局样式 col 控制 box 的宽度-->
  <div class="col-md-12">
    <!-- box 容器放置表单 -->
    <div class="box">
      <!-- 表单名 -->
      <div class="box-header">
        <h3 class="box-title">通用表单</h3>
      </div>
      <!-- 表单主体元素 -->
      <div class="box-body">
        表单内容
      </div>
    </div>
  </div>
</div>

```

```

        </div>
        <div class="box-footer">
            表单操作按钮，如提交、取消等
        </div>
    </div>
</div>
</div>

```

在浏览器中访问 form.html，效果如图 7-36 所示。



图 7-36 通用表单布局

## (2) 表单基本项

表单基本项包含了文本框、下拉选择框、单选按钮、复选框等，可以使用 Bootstrap 框架预定义的表单样式。Bootstrap 常见表单样式类的实现，对应的效果如图 7-37 所示。

Label	<input type="text"/>	form form-group form-control
Label	<input type="text"/> Label <input type="text"/>	form:form-inline form-group form-control
Label	<input type="text"/>	input-group input-group-addon form-group input-group-addon
Label	<input type="text"/>	form:form-horizontal row:cols-**-2
Label	<input type="text"/>	

图 7-37 使用 Bootstrap 预定义的表单样式



在 box 中的 box-body 类容器中，追加以下代码：

```
<form class="form-horizontal">
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">
      文本输入
    </label>
    <div class="col-sm-6">
      <input type="email" class="form-control" id="inputEmail3"
        placeholder="邮箱">
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword3" class="col-sm-2 control-label">密码输入
    </label>
    <div class="col-sm-6">
      <input type="password" class="form-control" id="inputPassword3"
        placeholder="输入密码">
    </div>
  </div>
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">选择性别
    </label>
    <div class="col-sm-3">
      <select class="form-control">
        <option>男性</option>
        <option>女性</option>
      </select>
    </div>
  </div>
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">文本域
    </label>
    <div class="col-sm-6">
      <textarea class="form-control" rows="3"></textarea>
    </div>
  </div>
  <div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">多选
    </label>
    <div class="col-sm-6">
      <div class="checkbox">
        <label>
          <input type="checkbox">
            选项 1
        </label>
        <label>
          <input type="checkbox">
            选项 2
        </label>
      </div>
    </div>
  </div>
</form>
```



```

        </label>
        <label>
            <input type="checkbox">
            选项 3
        </label>
    </div>
</div>
</div>
<div class="form-group">
    <label for="inputEmail3" class="col-sm-2 control-label">单选
    </label>
    <div class="col-sm-6">
        <div class="radio">
            <label>
                <input type="radio" name="optionsRadios" id="
                optionsRadios1" value="option1" checked="checked">
                选项 1
            </label>
            <label>
                <input type="radio" name="optionsRadios" id="
                optionsRadios2" value="option1" checked="">
                选项 2
            </label>
        </div>
    </div>
</div>
</div>
</form>

```

在浏览器中查看效果，如图 7-38 所示。

图 7-38 定义常用的表单元素

### (3) 提示样式

数据提交前需要做表单验证，为了让用户可以更快了解哪个输入项出了问题，提示必不可少。这里以文本输入框为例，使用 AdminLTE 自带样式 `has-success` 和 `has-error`，定义不同的提示效果，代码如下：



```

<div class="form-group has-success"> <!--成功的表单项，需要增加 has-success
预定义类-->
  <label for="inputEmail3" class="col-sm-2 control-label">
    文本输入
  </label>
  <div class="col-sm-6"><!--控制 INPUT 元素的宽度-->
    <input type="email" class="form-control" id="inputEmail3"
      placeholder="邮箱">
  </div>
  <!--表单提示区域-->
  <span class="help-block">
    <i class="fa fa-check"></i> <!--成功的提示-->
    成功提示文本
  </span>
</div>
<div class="form-group has-error">
  <label for="inputPassword3" class="col-sm-2 control-label">密码输入
</label>
  <div class="col-sm-6">
    <input type="password" class="form-control" id="inputPassword3"
      placeholder="输入密码">
  </div>
  <!--表单提示区域-->
  <span class="help-block">
    <i class="fa fa-times-circle-o"></i> <!--成功的提示-->
    失败提示文本
  </span>
</div>

```

输入正确和错误的提示效果，如图 7-39 所示。

图 7-39 输入正确或者错误的提示效果

上述修改，除了定义表单元素的提示颜色，还用到了 `help-block` 类，来定义表单元素后提示文本的布局样式。

#### (4) 日期选择器

在一些需要输入特殊格式的表单项，如日期、时间格式等，表单会提供相应的日期选择器插件，以提高正确格式输入的成功率，也提高了用户使用体验。AdminLTE 本身引入了日期相关类的插件，主要有以下几种。

- `bootstrap-datepicker`: 标准日期选择插件，日期格式为 2017-12-31，展示格式可以自定义。
- `bootstrap-daterangepicker`: 标准日期范围选择插件。默认提供两个日期选择框，用



户可以选择两个日期范围内的时间段。

- **bootstrap-timepicker**: 标准时间选择插件, 提供精确到时、分、秒的时间选择。

这里以 **bootstrap-datepicker** 插件为例, 在通用表单实例中引入年、月、日格式的日期选择效果。

首先在 **form.html** 页面的头部和底部, 引入 **bootstrap-datepicker** 的 CSS 样式表和 JavaScript 库文件, 代码如下:

```

<!-- bootstrap datepicker -->
<link rel="stylesheet" href="bower_components/bootstrap-datepicker/
dist/css/bootstrap-datepicker.min.css">
.....
<!-- AdminLTE App -->
<script src="dist/js/adminlte.min.js"></script>
<!-- bootstrap datepicker -->
<script
src="bower_components/bootstrap-datepicker/dist/js/bootstrap-datepicker
.min.js"></script>

```

 **提示:** **bootstrap-datepicker** 插件依赖 jQuery 框架。

定义文本输入标签, 展示日期交互操作, 代码如下:

```

<div class="form-group">
  <label for="inputEmail3" class="col-sm-2 control-label">日期选择
</label>
  <div class="col-sm-6">
    <div class="input-group date">
      <div class="input-group-addon">
        <i class="fa fa-calendar"></i>
      </div>
      <!-- 注意定义 ID 属性 -->
      <input type="text" class="form-control pull-right" id="
datepicker">
    </div>
  </div>
</div>

```

最后定义 JavaScript 脚本实现日期选择效果, 代码如下:

```

<script>
  $(function() {
    //日期选择插件, 参数配置
    $('#datepicker').datepicker({
      autoclose: true, // 选中后自动关闭
      format: 'yyyy-mm-dd', // 选择日期格式, 配置年、月、日的显示位置和样式
    });
  })
</script>

```



在浏览器中访问 form.html 文件，效果如图 7-40 所示。

图 7-40 在通用表单中定义日期插件

#### (5) 富文本编辑器

富文本编辑器用于发布带样式的内容文本，AdminLTE 集成了流行的开源编辑器 ckeditor。其使用原理与日期插件类似，首先引入 ckeditor 插件的 JavaScript 脚本库文件，代码如下：

```
<!-- bootstrap datepicker -->
<script
src="bower_components/bootstrap-datepicker/dist/js/bootstrap-datepicker
.min.js"></script>
<!-- CK Editor -->
<script src="bower_components/ckeditor/ckeditor.js"></script>
.....
```

随后定义文本域元素，代码如下：

```
<!-- 富文本编辑器 -->
<div class="form-group">
  <label for="editor1" class="col-sm-2 control-label">富文本编辑器
  </label>
  <div class="col-sm-6">
    <textarea id="editor1" name="editor1" rows="10" cols="80">
      文本域默认展示内容
    </textarea>
  </div>
  <span class="help-block"><i class="fa fa-check"></i> <!--成功的提示-->
  输入成功</span>
</div>
```

最后借助 CKEDITOR 对象，实现富文本编辑器的渲染，代码如下：

```
//通过 ID 属性，替换默认文本域为富文本编辑器
CKEDITOR.replace('editor1')
```

在浏览器中访问，效果如图 7-41 所示。

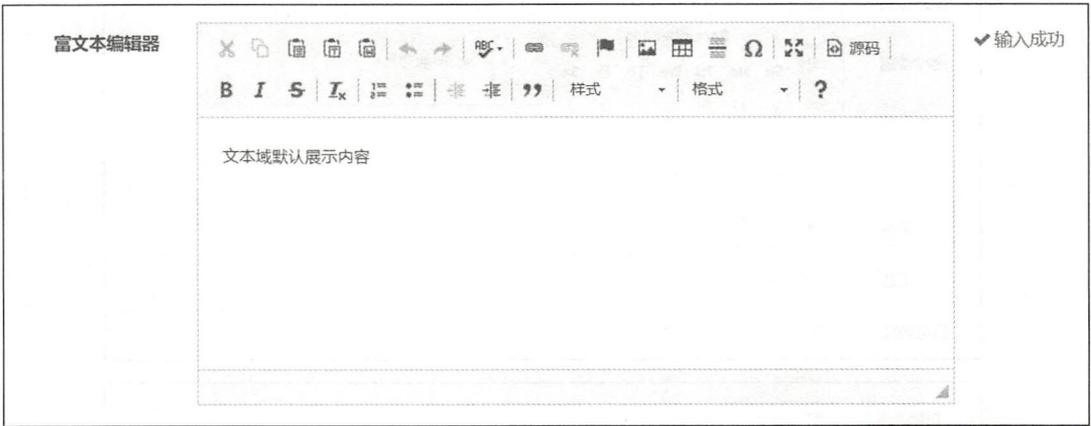


图 7-41 在通用表单中定义富文本编辑器

#### (6) 提交按钮组

最后别忘了提交按钮组，在 `box-footer` 结构中定义。这里以非表单直接提交为例，代码如下：

```
<div class="box-footer">
  <button type="submit" class="btn btn-default">取消</button>
  <button type="submit" class="btn btn-info pull-right">保存</button>
</div>
```

效果如图 7-42 所示。



图 7-42 定义通用表单的提交按钮组



# 第 8 章 ThinkPHP 命令行操作 与接口开发实战

ThinkPHP 是优秀的国产开源框架。它不仅有丰富的中文文档，还有很多开源的项目实例。本章不是罗列 ThinkPHP 的零碎知识点，而是解析 ThinkPHP 5 版本中的新特性，以帮助开发者学习如何使用 ThinkPHP 增强的命令行操作和接口开发等常用方法，从而掌握该框架的核心技巧。

## 8.1 ThinkPHP 5 与命令行操作

本节重点讲解如何使用 Composer 创建全新的 ThinkPHP 5 项目，引导开发者在实战中熟悉和使用命令行操作。

### 8.1.1 ThinkPHP 简介

ThinkPHP 是免费开源的面向对象轻量级 PHP 开发框架，是为实现敏捷 Web 应用开发和简化企业应用开发而诞生的。ThinkPHP 从诞生以来一直秉承简洁、实用的设计原则，在保持出色性能和至简代码的同时，也注重易用性。ThinkPHP 的 Logo 如图 8-1 所示。



图 8-1 ThinkPHP 的 Logo

目前有很多使用 ThinkPHP 作为核心框架的开源项目，如图 8-2 所示。



图 8-2 以 ThinkPHP 为核心框架的开源项目

ThinkPHP 的最新版本为 5.0.x，相比上一个版本 3.2.x，其对整体框架进行了重构，新版本的优势主要有以下几个方面：

- 灵活的路由和 API 开发的友好；
- 新的请求处理对象；
- 新的查询机制和模型功能；
- 增强的调试、单元测试和异常处理；
- Composer 和命令行的支持。

提示：ThinkPHP 5 还在快速迭代，目前已经发布 ThinkPHP 5.1 的开发者预览版。

## 8.1.2 使用 Composer 创建 ThinkPHP 5 项目

ThinkPHP 有多种安装方式，这里推荐使用 Composer 方式。为了方便演示，实例使用 Composer 全新安装 ThinkPHP 5，项目名为 tp5api。

在命令行中执行以下命令：

```
composer create-project tophink/think tp5api --prefer-dist
```

关于 Composer 的安装与使用，在第 6 章已经详细介绍过，这里不再赘述。

上述命令执行过程如下，说明已经安装成功：

```
wangjialin@wangjialin-gamepc MINGW64 /d/phpstudy/WWW
$ composer create-project tophink/think tp5api --prefer-dist
Installing tophink/think (v5.0.12)
- Installing tophink/think (v5.0.12): Downloading (100%)
Created project in tp5api
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
- Installing tophink/think-installer (v1.0.12): Downloading (100%)
- Installing tophink/framework (v5.0.12): Downloading (100%)
Writing lock file
Generating autoload files
```

在浏览器中访问地址 <http://localhost/tp5api/public>，展示效果如图 8-3 所示，说明 ThinkPHP 框架可以正常运行。

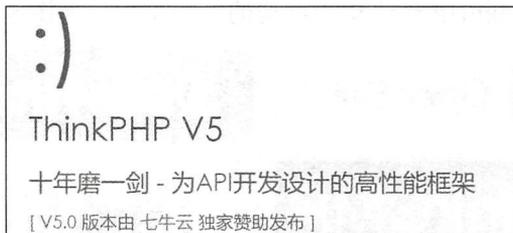


图 8-3 出现此界面则说明 ThinkPHP 安装成功

提示：相比 ThinkPHP 3，ThinkPHP 5 最低支持的 PHP 版本为 PHP 5.4，并且完美支持 PHP 7。

### 8.1.3 ThinkPHP 5 命令行操作

相比上一个版本，ThinkPHP 5 增加了很多全新的特性，命令行就是其一。

在命令行下，一些传统的手动操作可以自动完成，如创建模块、文件，一些不适合在页面上调试的功能，也可以在命令行下进行调试。下面以实例项目 `tp5api` 为例进行讲解。如需查看框架默认提供的命令，则在命令行中输入以下命令：

```
php think
```

执行结果如下：

```
wangjialin@wangjialin-gamepc MINGW64 /d/phpStudy/WWW/tp5api
$ php think
Think Console version 0.1
```

Usage:

```
command [options] [arguments]
```

Options:

```
-h, --help           Display this help message
-V, --version        Display this console version
-q, --quiet          Do not output any message
--ansi              Force ANSI output
--no-ansi           Disable ANSI output
-n, --no-interaction Do not ask any interactive question
-v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal
output, 2 for more verbose output and 3 for deb
```

ug

Available commands:

```
build           Build Application Dirs
clear           Clear runtime file
help           Displays help for a command
list           Lists commands
make
make:controller Create a new resource controller class
make:model      Create a new model class
optimize
optimize:autoload Optimizes PSR0 and PSR4 packages to be loaded with
classmaps too, good for production.
optimize:config  Build config and common file cache.
optimize:route   Build route cache.
optimize:schema  Build database schema cache.
```

从帮助的提示信息来看，框架中提供了各种操作命令，方便开发者使用。

提示：命令行工具需要在命令行下面执行，请先确保用户的 php.exe 已经加入了系统环境变量 Path。

## 1. 生成模块

在老版本的框架中，生成模块必须手动操作，费时、费力，还容易出错。ThinkPHP 5 中可以使用对应的指令自动生成模块，操作步骤如下。

### (1) 创建配置脚本文件

在实例 tp5api 项目中，在根目录下有一个 build.php 脚本文件，它提供了模块生成的示例定义。下面以生成名为 api 的新模块为例，首先需要在 application 目录下面创建一个 build.php 配置文件，文件内容如下：

```
<?php
return [
    // 定义 API 模块的自动生成（按照实际定义的文件名生成）
    'api' => [
        // 定义模块名
        '__file__' => ['common.php'], // 定义公共方法文件名
        '__dir__' => ['behavior', 'controller', 'model', 'view'], // 定义默认目录
        'controller' => ['Index', 'Test', 'UserType'], // 定义默认控制器文件
        // 定义默认模型文件
        'model' => ['User', 'UserType'], // 定义默认模型文件
        // 定义默认视图目录
        'view' => ['index/index'], // 定义默认视图目录
    ],
];
```

### (2) 执行生成操作

在 tp5api 根目录下，执行生成命令，结果如下，则说明操作成功。

```
$ php think build
Successed
```

### (3) 查看结果

进入 application 目录下，发现 api 模块已经自动创建完成，随后在浏览器中访问地址 <http://localhost/tp5api/public/index.php/api>，结果如图 8-4 所示。

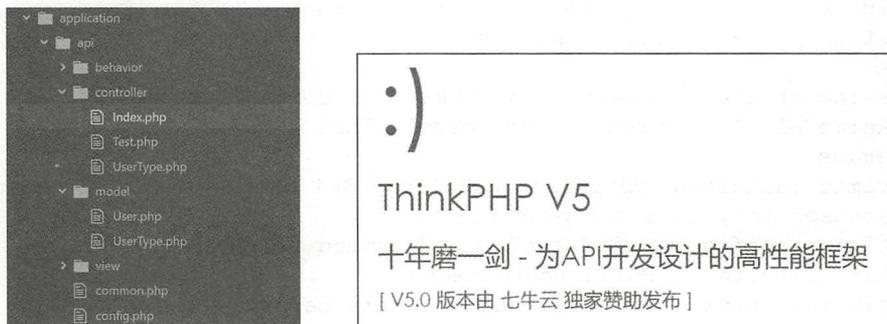


图 8-4 出现此界面说明命令行创建的控制器执行成功

提示：Build.php 文件中的配置项不都是必填项，可以根据需求灵活配置。

## 2. 生成控制器和模型文件

除了初始化模块目录，还可以使用 make 命令生成控制器和模型文件。例如：生成订单相关的控制器和模型，命令操作和执行结果如下：

```
cmd /d/phpStudy/www/tp5api
$ php think make:controller api/Order
Controller created successfully.
.....
cmd /d/phpStudy/www/tp5api
$ php think make:model api/Order
Model created successfully.
```

需要说明的是，make:controller 命令默认生成的是资源控制器，自带初始化方法定义。核心方法如下：

```
<?php
namespace app\api\controller;
use think\Controller;
use think\Request;
class Order extends Controller
{
    /**
     * 显示资源列表
     * @return \think\Response
     */
    public function index()
    {
        //
    }
}
.....
```

若希望创建一个空控制器，则只需要执行以下代码即可：

```
php think make:controller api/Order--plain
```

## 3. 自定义命令扩展——清除全部会话信息

命令行的作用远不止生成目录和文件这么简单，开发者还可以方便地扩展自己的指令，实现特定的功能需求。这里扩展一个用于会话信息的指令，操作步骤如下。

### (1) 创建指令类

在 application 下创建 console 目录，新增 Session.php 文件。核心代码如下：

```
// 会话控制命令
class Session extends Command
{
    // 指令配置
    protected function configure()
    {
```

```

// 设置命令名、额外选项和描述信息
$this->setName('session')
    ->addOption('clear' , 'd' , Option::VALUE_NONE , 'clear all
    session' , null)
    ->setDescription('Clear Session file');
}

// 指令操作
protected function execute(Input $input, Output $output)
{
    // 获取指令选项名称
    $path = $input->getOption('clear');
    if($path)
    {
        // 执行清除会话信息操作
        unset($_SESSION);
        $output->writeln("Clear Session Succeeded");
    }
    else
    {
        $output->writeln("Clear Nothing");
    }
}
}

```

 **提示:** 一个合法的命令类, 不要求有固定的目录和命名空间, 但必须继承 `think\console\command\Command` 或者其子类, 并且定义 `configure()` 和 `execute()` 两个方法。

在自定义的命令类中, `configure()` 方法定义了命令的名称、选项和功能描述, 其中 `setName()` 和 `setDescription()` 方法较好理解, 分别为定义命令的名称和功能描述。而 `addOption()` 方法参数较多, 根据定义的参数说明如下。

- `clear`: 指令的选项名定义。
- `d`: 指令的选项名别名。
- `Option::VALUE_NONE`: 选项类型, 值对应没有输入值 (`VALUE_NONE`)、输入值必须 (`VALUE_REQUIRED`)、输入值可选 (`VALUE_OPTIONAL`) 和数组输入值 (`VALUE_IS_ARRAY`) 4 种。
- `clear all session`: 选项说明信息。
- `null`: 默认值, 没有则默认为 `null`。

`execute()` 方法处理具体的输入、输出和实际处理逻辑。实例中的功能实现较为简单, 只是使用 `unset()` 方法清空了会话数据, 实际开发中会比实例更为复杂:

```

// 执行清除会话信息操作
unset($_SESSION);

```

## (2) 注册指令类

在 `application` 目录下面的 `command.php` (如果不存在, 则需创建) 文件中, 添加如下内容:

```
return [  
    '\app\console\Session',  
];
```

为了验证是否注册成功，可以执行以下指令：

```
php think list
```

执行结果如下，则说明注册成功。

```
Available commands:  
  build          Build Application Dirs  
  clear          Clear runtime file  
  help          Displays help for a command  
  list          Lists commands  
  session       Clear Session file
```

### (3) 测试自定义指令

执行下面代码的第一行指令，则显示结果如下面代码的第二行：

```
$ php think session --clear  
Clear Session Succeeded
```

也可以用别名执行，则提示如下：

```
$ php think session -d  
Clear Session Succeeded
```

若指令输入不存在，则提示如下：

```
$ php think session  
Clear Nothing
```

若别名输入有误，则提示如下：

```
$ php think session -a  
[RuntimeException]  
The "-a" option does not exist.
```

## 8.2 实战：制作一个短地址生成器

短地址（Short Url）的概念最早由推特网站（Twitter）提出的。短地址一般指长度较短的网址。例如，最早微博发送字数是有限制的（140 个字），短地址则便于用户分享。如今在很多地方都可以见到短地址。

### 8.2.1 功能简介

利用在线的短地址生成服务，可以将长地址转换为短地址。例如，使用新浪的在线短地址生成器，如图 8-5 所示。



图 8-5 使用新浪的在线短地址生成器

短地址的生成一般都有特定的算法。以新浪短地址为例，该短地址中包含了一个唯一的地址码。这个地址码可以对应原始地址，这样就实现了短地址的使用。

使用在线的短地址生成器时，每次都需要手动操作。例如在电商网站中，每个页面都需要手动生成短地址，工作量巨大，所以就需要网站可以自动生成短地址。

本节模拟实现一个资讯页的短地址自动生成程序，最终效果如图 8-6 所示。通过本节的学习，读者可以更好地理解 ThinkPHP 5 的开发流程。



图 8-6 在页面上展示当前 URL 的短地址

## 8.2.2 程序设计

本程序的主要功能包括资讯页展示、短地址展示、短地址自动生成和短地址解析 4 个部分。在程序实现上，需要建立短地址与原始地址的唯一映射关系，用户在访问当前页面时，系统查询是否已经生成并保存短地址，如没有则自动生成，最终展示在页面上。完整流程如图 8-7 所示。

当用户使用短地址访问页面时，系统根据短地址自动查询数据中与之对应的原始地址，最后进行页面的重定向，其流程如图 8-8 所示。

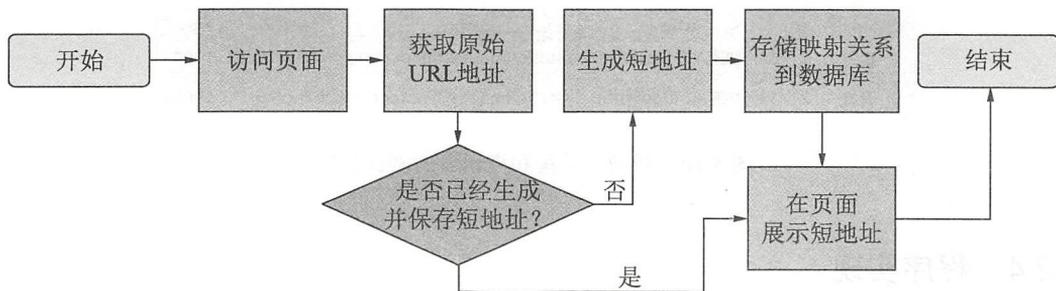


图 8-7 短地址的生成与展示

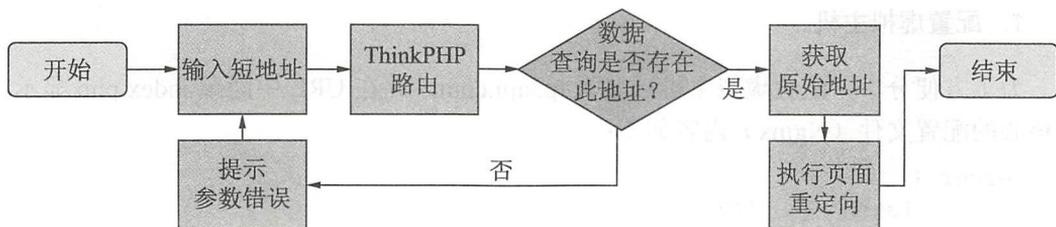


图 8-8 短地址转换为原始地址

### 8.2.3 数据表设计

本实例的数据结构相对简单，只需要设计下面两个表即可满足需求。

- 资讯表（数据表 news）：存储分享内容的实体，可以是文章、商品详情等。
- 短地址映射表（shorturl）：存储地址码和原始地址。

资讯表和短地址映射表的结构设计如图 8-9 所示。

列	类型	注释
id	int(4) 自动增量	主键id, 自增长
title	varchar(255)	资讯名称
img	varchar(255)	资讯图片
author	varchar(100)	资讯作者
pv	int(10)	资讯访问量
content	text	资讯内容
create_time	datetime	资讯发布时间

a) 资讯表结构设计

列	类型	注释
id	int(4) 自动增量	主键id, 自增长
shorturl	varchar(100)	地址码
url	varchar(255)	原始地址

b) 短地址映射表结构设计

图 8-9 短地址功能数据结构设计

为了方便演示，新建数据库命名为 tp5api，新增表结构和资讯表的测试数据（两篇资讯文章），如图 8-10 所示。

<input type="checkbox"/> Modify	id	title	img	author	pv
<input checked="" type="checkbox"/> 编辑	1	如何快速入门ThinkPHP5	/static/images/header.png	WANGJIALIN	1000
<input checked="" type="checkbox"/> 编辑	2	如何快速入门Bootstrap	/static/images/header.png	ZhangSan	10000

图 8-10 新增表结构和资讯表的测试数据

## 8.2.4 程序实现

使用 8.2.3 节中的实例项目 tp5api, 在 application/index 模块下编写代码。具体步骤如下。

### 1. 配置虚拟主机

为了方便分享, 配置虚拟主机域名为 tp5api.com, 并在 URL 中隐藏 index.php 显示。其核心的配置文件 (Nginx) 内容如下:

```
server {
    listen      80;
    server_name tp5api.com;
    root       "D:/phpStudy/www/tp5api/public";
    location / {
        index index.html index.htm index.php;
        #autoindex on;
        # 隐藏 index.php 入口文件
        if ( !-e $request_filename) {
            rewrite ^(.*)$ /index.php/$1 last;
            break;
        }
    }
    .....
}
```

### 2. 展示资讯数据

资讯数据的展示, 只需要把数据表 news 中的数据查询出来即可, 一般资讯页面都有相应的列表页。为了简化演示, 这里只实现详情页的演示。

#### (1) 控制器

在 application/index/controller 目录下, 新增 News 控制器文件, 其核心代码如下:

```
// 资讯详情类
class News extends Controller
{
    // 资讯详情页
    public function index()
    {
        $id = input('id'); // 获取资讯 ID
```

```

        if(!$id)
        {
            $this->error('参数错误! ');
        }
        // 在数据库中查询资讯信息
        $map = [];
        $map['id'] = $id;
        $info = Db::name('news')->where($map)->find();
        if(!$info)
        {
            $this->error('数据查询为空! ');
        }
        // 模板变量置换
        $this->assign('info' , $info);
        return $this->fetch();
    }
}

```

## (2) 模板视图

在 `application\index\view` 目录下, 新增 `news\index.html` 模板文件, 引入 Bootstrap 前端框架后, 进行资讯数据的变量置换。其核心代码如下:

```

<div class="container">
    <div class="row">
        <div class="col-md-12">
            <h1>{$info.title}</h1>
            <p>发布时间: {$info.create_time} 阅读: {$info.pv} 作者: {$info.
            author}</p>
            
        </div>
        <div class="col-md-12 text-center">
            <h4>正文开始</h4>
            <p>{$info.content}</p>
        </div>
        <div class="col-md-4 text-center"></div>
        <div class="col-md-4 text-center">
            <h4>快速分享</h4>
            <strong>复制并转发下方的文章地址, 十分感谢! </strong>
            <textarea class="form-control">{$shorturl}</textarea>
        </div>
        <div class="col-md-4 text-center"></div>
    </div>
</div>

```

## (3) 效果查看

在浏览器中访问地址 `http://tp5api.com/index/news/index/id/1`, 效果如图 8-11 所示。



图 8-11 完成资讯详情页的展示

### 3. 自动生成短地址

#### (1) 地址码生成方法

在 `application\common.php` 应用公共文件内，新增 `code62()`方法，实现把固定长度的数字转换为 8 位地址码。代码如下：

```
if(!function_exists('code62')) {
    //生成短地址核心类库
    function code62($x){
        $show='';
        while($x>0){
            $s=$x % 62;
            if ($s>35){
                $s=chr($s+61);
            }elseif($s>9&&$s<=35){
                $s=chr($s+55);
            }
            $show.=$s;
            $x=floor($x/62);
        }
        return $show;
    }
}
```

执行以下代码测试 `code62()`方法：

```
$str = date('YmdHis');
echo code62($str);
```

结果为中文大小写和数字组成的 8 位地址码：

```
62M5m7j5
```

通过地址码与原始地址的映射关联，实现短地址功能。

提示：PHP 方法 `chr()` 用于返回十进制、八进制和十六进制值。八进制值被定义为带前置符 0，而十六进制值被定义为带前置符 0x。

## (2) 短地址自动生成方法

在 `application\common.php` 应用公共文件内，新增 `shorturl()` 方法，实现短地址的生成和存储。其核心代码如下：

```
if(!function_exists('shorturl')) {
    // 生成短地址方法
    function shorturl($url){
        $old_url = $url;
        // 转换原始地址为数字
        $url=crc32($url);
        $result=sprintf("%u",$url);
        if($result)
        {
            // 查询当前短地址是否已经生成
            $map = [];
            $map['shorturl'] = code62($result);
            $info = db('shorturl')->where($map)->find();
            if($info === null)
            {
                // 记录原始地址与短地址的映射关系
                $data = [];
                $data['shorturl'] = code62($result);
                $data['url'] = $old_url;
                $insertResult = db('shorturl')->insert($data);
                if($insertResult === false)
                {
                    abort( 500 , 'URL 信息入库失败');
                }
            }
        }
        return 'http://'.$_SERVER['HTTP_HOST'] . '/min/' . code62($result);
    }
}
```

因为 `code62()` 方法需要传入数字类型，所以在 `shorturl()` 方法中，传入的 URL 参数先转换为数字：

```
// 转换原始地址为数字
$url=crc32($url);
$result=sprintf("%u",$url);
```

随后进行数据库查询操作，若已经查询到地址码（数据表 `shorturl` 字段），则直接返回短地址，否则执行数据库写入操作。

## (3) 在页面上展示短地址

在 `application\index\controller\News.php` 脚本文件中，修改 `index()` 方法，增加 `shorturl()` 方法处理，其代码如下：

```
.....
// 模板变量置换
$this->assign('info' , $info);
$this->assign('shorturl' , shorturl($this->request->url(true)));
.....
```

需要注意的是，因为当前控制器继承自 ThinkPHP 的控制器基类，可以直接使用 `$this->request` 属性获取请求的对象。而如下代码，表明获取当前请求的完整地址（带域名）。

```
$this->request->url(true)
```

随后修改 `index.html` 模板文件，增加 `textarea` 文本域，用来展示短地址信息。其代码如下：

```
.....
<div class="col-md-4 text-center"></div>
<div class="col-md-4 text-center">
    <h4>快速分享</h4>
    <strong>复制并转发下方的文章地址，十分感谢！</strong>
    <textarea class="form-control">{$shorturl}</textarea>
</div>
<div class="col-md-4 text-center"></div>
.....
```

代码保存后，在浏览器中访问，效果如图 8-12 所示。



图 8-12 在页面上展示当前地址的短地址

#### 4. 短地址自动跳转

为了方便查看错误信息，需要开启 ThinkPHP 5 的应用调试模式，在 `application\config.php` 文件中，修改配置如下：

```
// 应用命名空间
'app_namespace'      => 'app',
// 应用调试模式
'app_debug'          => true,
// 应用 Trace
'app_trace'          => false,
```

复制页面上生成的短地址，在浏览器中访问后报错，如图 8-13 所示。



图 8-13 直接访问短地址无法重定向

这是因为 ThinkPHP 5 无法解析短地址的 URL 模式，此时则需要使用路由定义把短地址的地址码转发给特定的控制器来处理，实现原始地址的查询和重定向。

#### (1) 定义短地址路由

在 `application\route.php` 中，追加以下路由规则：

```
//短地址跳转处理路由  
'min/:code' => 'index/Url/index',
```

在此规格下，任何访问“域名/min/参数值”都可以被自动转发到 `index` 模块下的 URL 控制器的 `index()` 方法，并且可以在 `index()` 方法中，获取参数名为 `code` 的参数值。

#### (2) 定义 URL 重定向控制器方法

在 `application\index\controller` 目录下，新增 `Url.php` 控制器文件，实现地址码的查询和页面重定向。其核心代码如下：

```
// 短地址跳转处理类  
class Url extends Controller  
{  
    // 文档详情页  
    public function index()  
    {  
        // 获取短地址参数  
        $code = request()->param('code');  
        if(!$code)  
        {  
            $this->error('参数错误!');  
        }  
        // 查询数据表中的映射关系  
        $map['shorturl'] = $code;  
        $url = Db::name('shorturl')->where($map)->value('url');  
        if(!$url)  
        {  
            $this->error('原始地址错误!');  
        }  
        // 页面重定向  
        $this->redirect($url);  
    }  
}
```

此时再去访问短地址，发现已经可以自动重定向到相应的资讯详情页了。

## 8.3 RESTful API 实战：用户接口权限验证

随着移动互联网的发展，前端设备层出不穷（手机、平板、PC 和其他设备），服务端必须有一个相对统一的机制，方便前端设备与服务端进行通信。

### 8.3.1 RESTful API 简介

RESTful API 中的 REST 是英文 Representational State Transfer（具体状态转移）的缩写，是一种基于 HTTP 协议、URI（统一资源定位符）、JSON 和 XML 这些现有协议与标准的，针对网络应用的设计和开发方式。

要理解 RESTful 的设计模式，需要先明确两个概念。

#### 1. 资源（Resource）

在 RESTful API 设计中，系统中所有对象都被抽象为资源，资源通过 URI 指向。例如：有一个电子商务汇总系统，用户、商品、订单和其他等都是组成系统的资源。不过这样的设定，对没使用过 RESTful API 的开发者来说会感觉比较难理解，因为一般的 API 接口开发，URI 指向的都是动作。例如一个典型的获取用户接口定义如下：

```
http://www.wangjialin.com/user/getUserInfo?id=777888
```

创建订单接口，定义如下：

```
http://www.wangjialin.com/order/createNewOrder
```

可以看到上述接口中，都是使用动词进行接口定义。而 RESTful API 恰恰与之相反，每种资源对应一个特定的 URI，需要哪个资源，只需要访问它的 URI 即可。

例如，定义一个用户资源 URI：

```
http://www.wangjialin.com/user
```

上述定义中，除了域名只定义了 user 用户资源关键字，其本身是没有任何动作的。而如何访问这些资源，就需要继续看 HTTP 动作。

#### 2. HTTP动作（HTTP Method）

在 RESTful API 设计模式中，对于资源的具体操作类型由 HTTP 协议的各种动作实现。常见的 HTTP 动作有下面几个。

- GET：从服务器获取一项或者多项资源。
- POST：在服务器新建一个资源。
- PUT/PATCH：在服务器上更新资源。两者的区别在于是否只提供修改的属性，前者

需要客户端提供完整的修改后的资源对象，而后者只需要提供要修改的属性即可。

- DELETE: 从服务器删除资源。

如需要把传统的获取用户信息的接口修改为 RESTful API 风格，则代码如下：

```
GET http://www.wangjialin.com/user/ID // 获取某个指定 (ID) 用户的信息
```

修改创建订单的代码如下：

```
POST http://www.wangjialiun.com/order // 创建一个新的订单信息
```

通过使用 HTTP 相关动作，可以把资源进行状态的转化，实现对应的操作。

 **提示：** API 接口的设计模式很多，RESTful API 只是其中之一，适合满足特定应用场景下的需求。

### 8.3.2 实现接口权限验证

开发用户接口实例之前，还需要设计一套严格的接口验证方法，保证 API 接口的安全性。常见的接口验证一般包含以下几种：

- 接口时效性验证。为了防止大量的重复请求，可以设置每次请求的时效。小到以秒为单位，大到可以是半个小时。
- 接口参数完整性验证。一般的接口攻击，都会截获请求，附带额外参数进行攻击，需要进行过滤验证。此时就需要客户端把所有的请求参数根据一个特定的算法，生成一个签名字符串，并在请求时一并发送。服务器接收到这个签名字符串后进行验证。
- 用户唯一 Token。用户每次登录都会生成唯一的 Token 信息，注销或者超时会话时 Token 都会被注销，防止被非法获取。

根据以上设计思路，本实例设计基于时效和参数效验的权限验证方法。接口验证流程如图 8-14 所示。

下面以 ThinkPHP 5 框架为例，查看实现接口验证父类的步骤与方法。

#### 1. 创建项目

使用以下命令创建 ThinkPHP 5 项目：

```
composer create-project topthink/think tp5restfulapi --prefer-dist
```

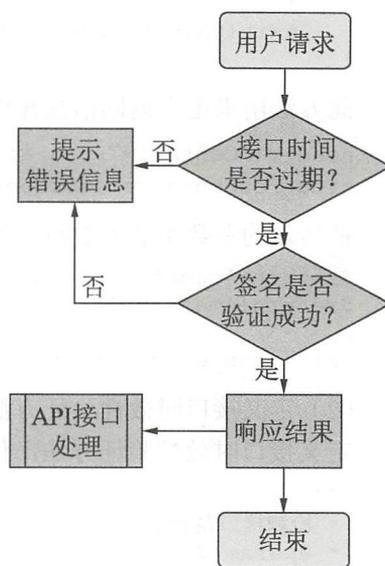


图 8-14 接口验证的时效性和签名流程

完成初始化后，为 tp5restfulapi 绑定虚拟域名 tp5restfulapi.com。

## 2. 实现权限验证基类

在 application 目录下创建名为 api 的应用模块，随后新建控制器 Api.php。操作步骤如下。

### (1) 定义返回值方法

在 Api.php 控制器文件中，新增以下方法：

```
/**
 * 数据返回
 * @param string $code HTTP CODE
 * @param string $message 提示信息
 * @param array $data 返回数据
 */
public function return_msg($code = '200', $message = '', $data = [])
{
    header('Content-Type: application/json'); // 设置返回类型
    http_response_code($code); // 设置返回头部
    $return['code'] = $code;
    $return['message'] = $message;
    if (!empty($data))
    {
        $return['data'] = $data;
    }
    exit(json_encode($return, JSON_UNESCAPED_UNICODE));
}
```

此方法用来定义返回信息操作，其中定义了 HTTP Header 的类型和编码，代码如下：

```
header('Content-Type: application/json'); // 设置返回类型
http_response_code($code); // 设置返回头部
```

把传入的参数组装为数组，返回 JSON 格式数据，代码如下：

```
$return['code'] = $code;
$return['message'] = $message;
.....
exit(json_encode($return, JSON_UNESCAPED_UNICODE));
```

### (2) 定义接口时效性验证方法

定义接口时效性验证方法，代码如下：

```
/**
 * 检测接口是否超时
 * @param $arr
 */
public function check_time($arr) {
    if (!isset($arr['time']) || intval($arr['time']) <= 1)
    {
        $this->return_msg(400, '时间戳错误!');
    }
    if (time() - intval($arr['time']) > $this->timeout_second)
    {

```

```

        $this->return_msg(400, '请求超时!');
    }
}

```

该方法中进行了两部分验证：时间参数（time）是否非空验证和是否超时验证。如果任意一项无法匹配，方法都会返回错误信息。其中\$this->timeout\_second 属性，定义了接口时效是多少秒，在类中定义为属性，代码如下：

```

// 默认接口的有效期为 60 秒
protected $timeout_second = 60;

```

### （3）定义生成接口签名方法

为了防止请求中带有其他恶意的参数，需要根据原始数据，定义签名生成方法，代码如下：

```

/**
 * 构建请求签名
 * @param $param
 * @return string
 */
public function buildSign($param)
{
    unset($param['sign']); // sign 字段不需要加入签名算法
    unset($param['time']);
    ksort($param); // 键值对的 key 按照升序排序
    $str = implode('', $param); // 请求参数值拼接成字符串
    $sign = md5(md5($str). $this->sign_key); // 执行加密
    return $sign;
}

```

在 buildSign()方法中，生成签名的算法主要有以下 3 步：

- 对请求参数的数组进行排序，使用 PHP 中的 ksort()方法进行排序。
- 把排序结果转换为字符串。
- 使用 md5()方法和默认签名的 Key 进行签名生成的运算。

其中，\$this->sign\_key 属性值在类中定义，代码如下。签名 Key 可以是动态的，只要能 and 客户端在生成签名时保持一致即可。

```

// 执行加密 key
private $sign_key = 'ThinkPHP5';

```

### （4）定义签名验证方法

该方法把请求接口的参数值，先进行排序和字符串转换，然后再进行生成签名处理。返回值中所带的签名，如果和验证参数值生成的签名值保持一致，则通过验证，反之则提示错误信息。该方法定义如下：

```

/**
 * 验证请求签名是否正确
 * @param $param
 */
public function check_sign($param)

```

```

{
    if(!isset($param['sign']) || !$param['sign'])
    {
        $this->return_msg(400, '签名不能为空!');
    }
    if($param['sign'] !== $this->buildSign($param))
    {
        $this->return_msg(400, '签名错误!');
    }
}
}

```

(5) 在初始化方法中调用验证

验证方法定义完成后，需要在控制器的初始化方法中调用，代码定义如下：

```

// 初始化方法
public function _initialize()
{
    parent::_initialize();
    // 获取请求对象
    $this->request = Request::instance();
    // 是否开启 API 权限验证（方便测试）
    if(config('api_auth'))
    {
        // 验证时间戳是否超时
        $this->check_time($this->request->only(['time']));
        // 验证签名是否正确
        $this->check_sign($this->request->param());
    }
}
}

```

需要注意的是，初始化方法中不仅获取了请求对象：

```
$this->request = Request::instance();
```

还使用了系统配置，来定义是否开启验证（方便开发者模式）：

```
if(config('api_auth'))
.....
```

在 `application\api\config.php` 文件中，定义如下：

```

<?php
// API 模块设计
return [
    // 应用命名空间
    'api_auth' => true
];

```

```

{
    code: 400,
    message: "时间戳错误!"
}

```

图 8-15 接口超时验证失败

访问地址 `http://tp5restfulapi.com/api/api/return_msg`，若效果如图 8-15 所示，则说明 API 接口的全局验证成功。

### 8.3.3 创建用户表结构与模型

在定义完成 API 接口的安全验证基类后，下面继续定义获取用户信息的表结构。创建

数据库 tp5restfulapi，新增 User 表，结构如下：

```
DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `id` int(4) NOT NULL AUTO_INCREMENT COMMENT '主键 id, 自增长',
  `mobile` varchar(255) NOT NULL COMMENT '手机号',
  `name` varchar(255) NOT NULL COMMENT '用户姓名',
  `email` varchar(255) NOT NULL COMMENT '用户邮箱',
  `status` tinyint(4) NOT NULL COMMENT '用户状态',
  `create_time` int(10) NOT NULL COMMENT '用户创建时间',
  `update_time` int(10) NOT NULL COMMENT '用户更新时间',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

在 application\api\model 目录下，新增 User 表自定义模型类，代码如下：

```
<?php
namespace app\api\model;
use think\Model;
/**
 * 用户表自定义模型
 * Class User
 * @package app\api\model
 */
class User extends Model
{
    // 自动写入时间戳
    protected $autoWriteTimestamp = true;

    // 数据写入时，状态字段默认为 1
    protected $insert = [
        'status' => 1,
    ];
}
```

在 User 表的模型中，定义自动写入 create\_time 和 update\_time 字段当前日期时间戳，并设置状态 status 字段的值为 1。

### 8.3.4 开发 RESTful API 接口类

为了快速生成控制器类，在命令行下切换到应用根目录下，执行以下命令：

```
php think make:controller api/User
```

系统会自动生成一个 User 资源控制器类，并创建对应资源操作的几个方法。为了在 URL 中可以访问资源，则需要定义资源路由规则。在 application 目录下的 route.php 文件中，追加以下内容：

```
Route::resource('users', 'api/User');
```

按资源路由的注册方法，定义一个名为 users 的路由，其实内部会自动注册 7 个路由规则，详细对应关系如表 8-1 所示。

表 8-1 资源路由自动生成规格表

标识	请求类型	生成路由规则	对应操作方法（默认）	描述
index	GET	users	index	显示用户列表
create	GET	users/create	create	新增用户页面
save	POST	users	save	保存用户信息
read	GET	users/:id	read	查看用户信息
edit	GET	users/:id/edit	edit	编辑用户页面
update	PUT	users/:id	update	更新用户信息
delete	DELETE	users/:id	delete	删除用户

修改 User 资源控制器，在对应方法中实现数据的增、删、改、查操作，代码如下：

```
<?php

namespace app\api\controller;

use think\Request;
use app\api\model\User as UserModel; //为了区分控制器和模型，使用模型别名

/**
 * 用户资源类
 * Class User
 * @package app\api\controller
 */
class User extends Api
{
    /**
     * 显示用户列表
     * @throws \think\Exception\DbException
     */
    public function index()
    {
        $list = UserModel::all(); // 使用自定义模型获取所有的列表
        return $this->return_msg(200, '', $list);
    }

    /**
     * 保存用户信息
     * @param Request $request
     */
    public function save(Request $request)
    {
        $request = UserModel::create($request->param());
        // 根据传入的数据，创建记录
        $this->return_msg(200, '', $request);
    }

    /**
     * 根据 ID 获取用户信息
     * @param $id
     * @throws \think\Exception\DbException

```

```

    */
    public function read($id)
    {
        // 根据 ID 查询单条记录
        $this->return_msg(200 , ' ' , UserModel::get($id));
    }

    /**
     * 保存更新的资源
     *
     * @param \think\Request $request
     * @param int $id
     * @return \think\Response
     */
    public function update(Request $request, $id)
    {
        // 根据 ID 和数据更新用户信息
        $result = UserModel::update($request->param() , ['id' => $id]);
        return $this->return_msg(200 , ' ' , $result);
    }

    /**
     * 根据用户 ID 删除记录
     *
     * @param int $id
     * @return \think\Response
     */
    public function delete($id)
    {
        // 根据 ID 删除一条记录
        return $this->return_msg( 200 , '删除成功' , UserModel::destroy($id));
    }
}

```

在上述代码中，各个资源操作方法中对于 User 表模型操作都使用了预先定义的 User 模型类，而为了和 User 控制器区分，在引入时使用了别名：

```
use app\api\model\User as UserModel;
```

系统自动创建的 User 资源控制器类默认继承自 ThinkPHP 的 Controller 控制器类，不过为了实现安全效验和接口验证，需要修改继承自 API 类：

```
class User extends Api
{
    .....
}
```

此外，在 ThinkPHP 5 框架中，使用自定义模型的 CURD 操作（增加、更新、读取和删除），与 Db 类稍有不同，在实例中用到的操作如下：

```

$list = UserModel::all();           // 读取列表
UserModel::get($id)                // 获取用户信息
UserModel::update($request->param() , ['id' => $id]); //更新用户信息
UserModel::destroy($id)            // 删除用户信息

```

提示：资源控制器自动生成的 create 和 edit 方法通常显示视图操作的表单，因为接口用不到，所以在实例中移除了。

### 8.3.5 测试 RESTful API 接口

测试 RESTful API 接口的方法有很多，这里主要介绍使用第三方工具 POSTMAN 来对接口进行测试。

#### 1. 安装 POSTMAN 软件

POSTMAN 是模拟请求测试工具，访问官方网站 <https://www.getpostman.com>，可以选择下载客户端或 Chrome 插件的方式进行安装，这里推荐安装本地客户端，安装过程不再赘述。

安装完成后，新用户第一次使用需要注册。POSTMAN 的使用界面如图 8-16 所示。

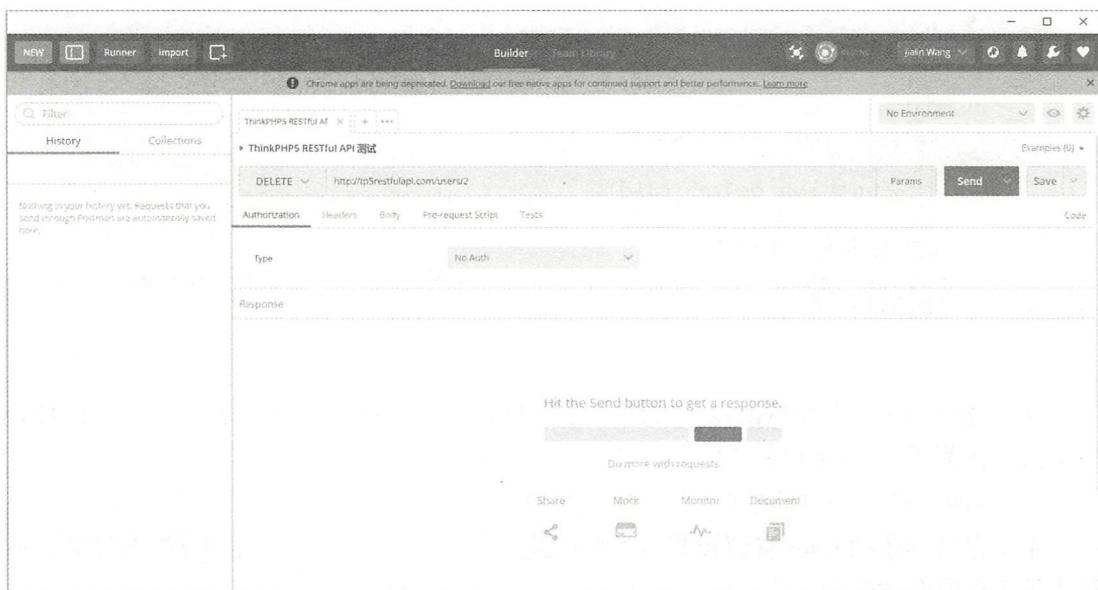


图 8-16 POSTMAN 的使用界面

#### 2. 接口调试

完成以上准备工作后，就可以对已经开发完毕的接口进行调试了。下面是几个接口调试的步骤。

##### (1) 新增用户信息

使用 POSTMAN，选择请求类型为 POST，请求地址设置为：

`http://tp5restfulapi.com/users`

单击 Params 按钮请求添加参数, 如图 8-17 所示。



图 8-17 新增用户信息参数配置

完成以上配置后, 单击 Send 按钮, 接口模拟发送请求。若返回结果如下所示, 则说明接口测试成功。

```
{
  "code": 200,
  "message": "",
  "data": {
    "name": "wangjialin",
    "mobile": "13511112222",
    "email": "wangjialin@wangjialin.com",
    "status": 1,
    "create_time": "2017-12-12 15:01:34",
    "update_time": "2017-12-12 15:01:34",
    "id": "6"
  }
}
```

随后在数据库中查看, 发现数据表 User 已经成功增加了一条记录, 如图 8-18 所示。

<input type="checkbox"/> Modify	id	mobile	name	email	status	create_time	update_time
<input type="checkbox"/> 编辑	6	13511112222	wangjialin	wangjialin@wangjialin.com	1	1513062094	1513062094

图 8-18 通过接口新增的表记录

## (2) 获取用户信息列表

为了方便演示, 再新增一条用户数据。随后修改请求类型为 GET, 地址不变:

`http://tp5restfulapi.com/users`

删除其他的请求参数, 单击 Send 按钮后, 可以看到接口返回的是一组列表信息, 具体如下:

```
{
  "code": 200,
  "message": "",
  "data": [
    {
      "id": 6,
      "mobile": "13511112222",
```

```

        "name": "wangjialin",
        "email": "wangjialin@wangjialin.com",
        "status": 1,
        "create_time": "2017-12-12 15:01:34",
        "update_time": "2017-12-12 15:01:34"
    },
    {
        "id": 7,
        "mobile": "13322221111",
        "name": "zhangsan",
        "email": "zhangsan@zhangsan.com",
        "status": 1,
        "create_time": "2017-12-12 15:08:18",
        "update_time": "2017-12-12 15:08:18"
    }
]
}

```

### (3) 获取单个用户信息

和获取用户列表信息类似，请求类型为 GET，但是需要传入用户唯一标识。以获取用户 ID 为 6 的用户信息为例，访问的 URL 地址如下：

```
http://tp5restfulapi.com/users/6
```

直接使用路由传递用户唯一标识，不需要额外再附加参数。返回结果如下：

```

{
  "code": 200,
  "message": "",
  "data": {
    "id": 6,
    "mobile": "13511112222",
    "name": "wangjialin",
    "email": "wangjialin@wangjialin.com",
    "status": 1,
    "create_time": "2017-12-12 15:01:34",
    "update_time": "2017-12-12 15:01:34"
  }
}

```

### (4) 更新用户信息

以修改用户 ID 为 6 的用户记录为例，首先需要修改请求类型为 PUT，随后设置 URL 地址为：

```
http://tp5restfulapi.com/users/6
```

最后增加要修改的请求参数，如图 8-19 所示。

Key	Value	Description	*** Bulk Edit
<input checked="" type="checkbox"/> name	xiaowang		
<input checked="" type="checkbox"/> email	xiaowang@xiaowang		

图 8-19 更新用户信息操作

单击 Send 按钮后，若执行成功，则返回字段更新的内容和更新时间，代码如下：

```
{
  "code": 200,
  "message": "",
  "data": {
    "name": "xiaowang",
    "email": "xiaowang@xiaowang",
    "id": "6",
    "update_time": "2017-12-12 15:15:08"
  }
}
```

此时再去查看数据表数据，发现用户数据已经更新，如图 8-20 所示。

<input type="checkbox"/> Modify	id	mobile	name	email	status	create_time	update_time
<input checked="" type="checkbox"/> 编辑	6	13511112222	xiaowang	xlawang@xiaowang	1	1513062094	1513062908
<input type="checkbox"/> 编辑	7	13322221111	zhangsan	zhangsan@zhangsan.com	1	1513062498	1513062498

图 8-20 更新后的用户记录

#### (5) 删除用户信息

修改请求类型为 DELETE，在 URL 地址上直接传入要删除的用户 ID 即可：

<http://tp5restfulapi.com/users/7>

单击 Send 按钮后，执行成功后，会提示数据已经删除成功，代码如下：

```
{
  "code": 200,
  "message": "删除成功",
  "data": 1
}
```

此时再去查看数据表数据，发现 ID 为 7 的记录已经被删除，如图 8-21 所示。

<input type="checkbox"/> Modify	id	mobile	name	email	status	create_time	update_time
<input checked="" type="checkbox"/> 编辑	6	13511112222	xiaowang	xlawang@xiaowang	1	1513062094	1513062908

图 8-21 通过接口删除用户表记录

基础架构实战——基础架构、用户与菜单管理

配置和权限管理

## 第3篇

# 项目实战篇

- ▶▶ 第9章 内容管理框架实战——基础架构、用户与菜单管理
- ▶▶ 第10章 内容管理框架实战——配置和权限管理
- ▶▶ 第11章 Crontab 计划任务管理
- ▶▶ 第12章 基于 Redis 队列的商城抢购系统

# 第9章 内容管理框架实战——基础架构、用户与菜单管理

内容管理框架（Content Manager Framework, CMF），不仅具有与普通开发框架相当的灵活性、扩展性，而且还包含了常用的内容组件、完整的后台管理模块，如用户模块、权限模块与配置模块等，以及提供了通用的模板样式（如 AdminLTE）与 JavaScript 交互脚本。这使得很多项目的功能不需要重新开发，减少了工作量，提高了开发的效率。

本章主要讲解了如何实现一个基本的内容管理框架，其中包含最基本的后台项目搭建、基础模板布局、用户登录与验证和基于无限分类的菜单模块等内容。

## 9.1 内容管理框架

本节主要介绍了 PHP 程序开发中除原生开发之外的几种开发模式，此外还介绍了基本的内容管理框架所包含的功能模块。

### 9.1.1 PHP 常见开发模式

开发 PHP 脚本程序时，为了提高效率，都会使用一些类库和框架。根据实际开发中的常见需求，开发者除了使用 PHP 语言进行原生开发外，通常会借助第三方类库和开发框架进行辅助开发。其常见开发模式主要分为以下几类：

（1）使用第三方类库。例如项目需要实现发送邮件的功能，使用原生 PHP 实现会过于烦琐，此时就可以引入如 PHPMailer 这样的第三方类库，提高开发效率。

（2）使用开发框架。这是最为主流的开发方式，由于项目规模的扩大和团队成员的扩张，使用第三方类库自行搭建已经无法满足效率上的要求。而主流的开发框架，很多基于 MVC 设计模式，提供了基本的控制器、ORM 和模板视图，具有极高的灵活性、扩展性。

（3）使用内容管理系统（简称 CMS）。此类系统的特点是，以产品的形式发布，完成度高，功能复杂，但不适合复杂的定制化开发，而适合快速创建特定类型（如电子商务、内容系统等）网站的用户，开发者只需要简单修改，或者不进行修改即可完成开发。此类

产品有 PHPCMS、ECShop 等。

而内容管理框架就是介于开发框架和内容管理系统之间，由它们衍生发展而来。其不仅拥有开发框架的灵活性和扩展性，还包含内容管理系统中已经完成的部分功能。

常见的内容管理框架，大多带有一个完整的后台管理模块，是使用 PHP 框架开发而来的，至少包含以下几个部分：

- 用户模块；
- 权限模块；
- 菜单模块；
- 系统模块；
- 配置模块。

除了以上几个基本上每个项目都有的功能外，很多内容管理框架还集成了：

- 云文件存储和 CDN 加速；
- 微信、APP 的 API 支持；
- 灵活好用的插件机制；
- 完整的内容和分类管理。

如果说内容管理系统可以满足非开发者的建站需求，那么内容管理框架就是帮助开发者快速开发网站的利器。不过和所有的类库工具一样，内容管理框架并不能适应所有的开发场景，其更适合中小规模的项目。

图 9-1 是开发者使用较多的几个 PHP 内容管理框架。



图 9-1 常见 PHP 内容管理框架

## 9.1.2 系统功能设计

下面以设计、开发一个内容管理系统进行讲解。参考主流的内容管理框架，以 OneThink（基于 ThinkPHP 3.2 开发）为例，安装 OneThink 并使用后可发现，OneThink 包含了一个后台管理模块，而其中最为常见的功能模块如图 9-2 所示。

这几个模块可以说缺一不可，既相互独立，又相互联系。下面详细来看这几个模块的主要功能设计和依赖关系。

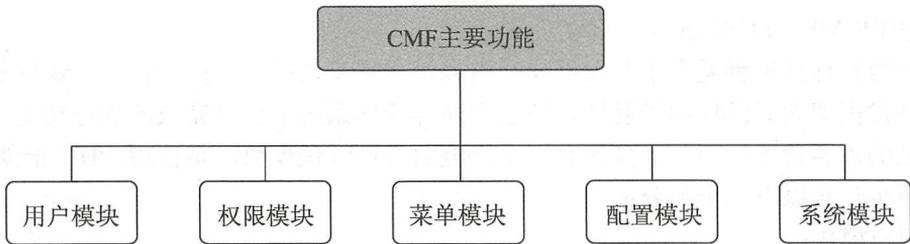


图 9-2 内容管理框架常见功能模块

## 1. 用户模块

用户模块是最基础的模块，其功能设计都是围绕用户操作需求而扩展来的。如果说权限模块限制了用户的操作范围，那菜单模块就是为权限模块提供了必要的节点支持。在后台管理模块中，用户模块（对用户的管理）所包含的功能如图 9-3 所示。

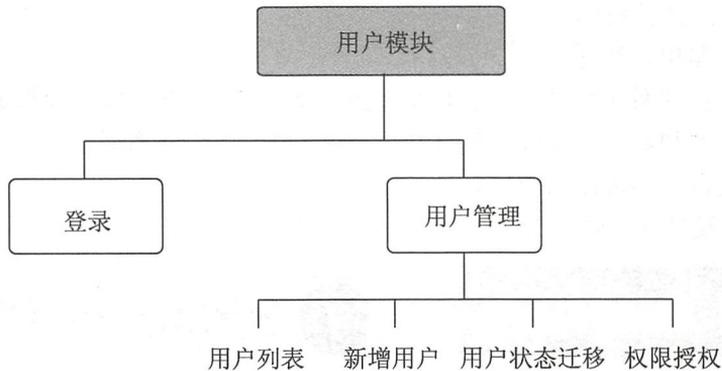


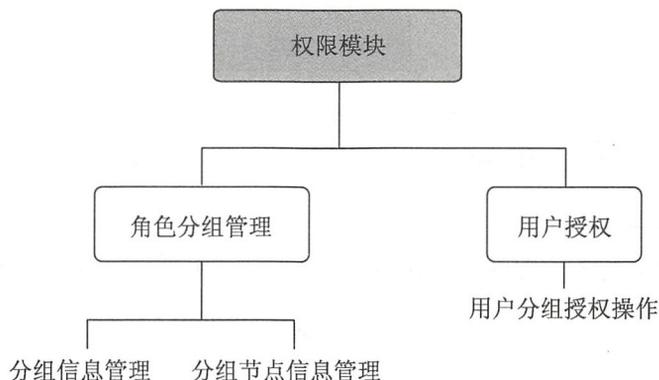
图 9-3 用户模块的功能结构图

## 2. 权限模块

权限模块的设计，采用 RBAC（Role-Based Access Control，基于角色的访问控制）权限设计思想。简单来说，就是一个用户拥有若干角色，每个角色拥有若干的权限。用户与角色之间、角色与权限之间，都是多对多的管理。

最简单的权限模块设计，就是用户与权限节点是一对多的关系，但当用户量越多，权限节点量越多，记录用户与权限节点之间关系的表记录就越多，无法实现简单高效的管理结构。而角色的加入，则有效解决了这个问题。例如，角色 A 有新增和删除两种权限，而角色 B 只有编辑的权限，若用户拥有 A、B 角色的同时，就拥有了所有的权限，只需要关联角色信息即可，而非关联大量的权限节点信息。

权限模块在系统中的功能如图 9-4 所示。



### 3. 菜单模块

后台管理模块中，用户操作的第一步，就是根据各个菜单项去选择要使用的功能。这里菜单项与权限节点就产生了联系，菜单就是限制用户操作的重要途径。菜单模块相对简单，只有菜单列表（查看）、菜单增加（编辑）与菜单删除（状态迁移）等功能，如图 9-5 所示。

### 4. 配置模块

为了方便管理，系统中很多参数都使用配置管理，如 ThinkPHP 5 就提供了成熟的配置操作方法 `config()`，还可以与数据库、缓存等关联，使用起来非常方便。

在设计上，配置模块的功能与菜单模块类似，如图 9-6 所示。

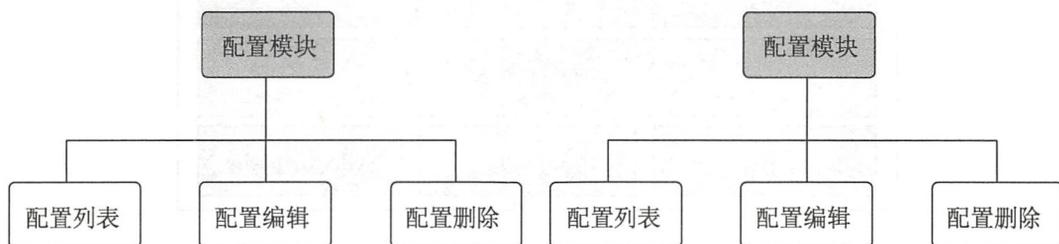


图 9-5 菜单模块的功能结构图

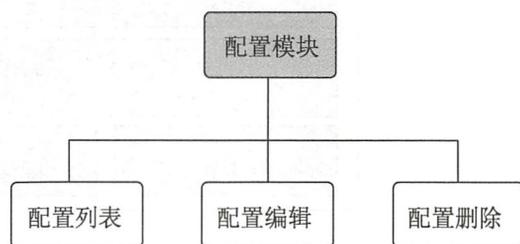


图 9-6 配置模块的功能结构图

### 5. 系统模块

管理网站的基本信息和系统配置，如网站名称、是否开启调试模式、搜索关键字等都可以在系统模块中操作，其主要功能如图 9-7 所示。

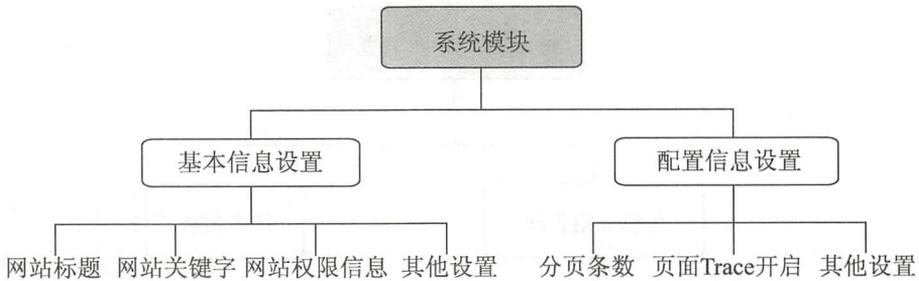


图 9-7 系统模块的功能结构图

### 9.1.3 程序架构设计

为了保证系统的灵活性，开发一个内容管理框架，需要选择一个主流的开发框架，实现底层程序的支持。除了后端框架，还需要选择一个前端框架，实现后台管理模块界面的快速搭建，这里使用第7章、第8章讲过的如下框架技术：

- ThinkPHP 5 框架；
- Bootstrap 框架+AdminLTE 后台模板。

程序架构如图 9-8 所示。

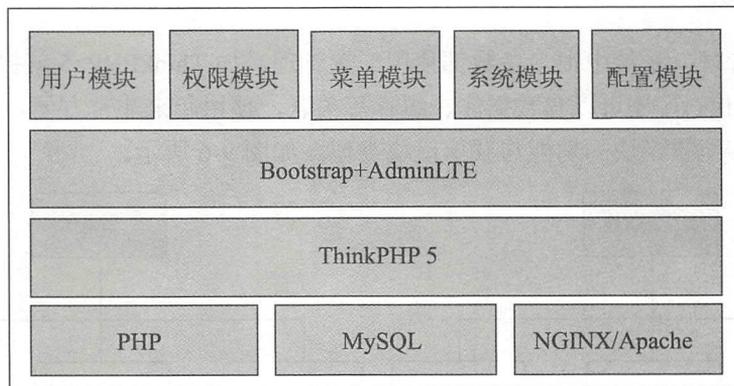


图 9-8 内容管理系统程序结构图

## 9.2 基础模板布局

本节主要讲解如何实现基础模板布局，不仅为后面的功能模块开发提供样式的支持，而且也是深入理解 ThinkPHP 5 模板特性的好途径。

## 9.2.1 准备工作

在后面的演示实例中，使用 ThinkPHP 5 作为基础框架，AdminLTE 模板风格为默认样式，进行准备工作的步骤如下。

### 1. 创建项目

使用 Composer 创建 ThinkPHP 5 项目，命名为 wangcmf，作为内容管理框架的实例进行演示。使用命令行在 application 目录下创建 admin 目录，用来开发后台管理模块。在 admin 目录下，默认新建 controller（控制器）、validate（数据验证）和 view（模板展示）三个目录，在 admin 下新增 config.php 配置文件。

ThinkPHP 5 框架初始化的步骤，读者可以参考第 8 章，这里不再赘述。

### 2. 新建静态资源目录并引入第三方类库

开发后台管理模块需要引入 AdminLTE 模板样式类库，所以在项目根目录，进入 public 目录下的 static（静态资源）目录中，创建 admin 目录，随后根据习惯，创建下面各个资源目录。

- libs: 存放第三方类库，如 Bootstrap 框架和 AdminLTE 后台模块样式源文件等。
- css: 存放 CSS 样式表文件。
- js: 存放 JavaScript 脚本文件。
- images: 存放静态图片资源文件。

以上几个目录中，目前只需要在 libs 目录中，引入 AdminLTE 类库相关文件（目录命名为 adminlte）即可，其他目录预留暂不使用。

## 9.2.2 创建基础布局模板

基础布局模板可以使用 ThinkPHP 5 自带的模板继承与模板引入来实现。模板继承是一项更加灵活的模板布局方式，它不同于模板引入，它可以定义一个基础布局模板，并且定义相关区块（block），然后继承（extend）该基础模板的子模板就可以对基础模板中定义的区块进行重载。

下面讲解如何在 wangcmf 演示实例中，实现基础模板布局。

### 1. 编写基础模板布局文件 layout.html

在项目的 application\admin\view 下，新建 basic 目录，随后新建 layout.html 文件。编写代码如下：

```
{include file='basic:header'/'} <!-- 引入顶部公共模板文件 -->
```

```
{include file='basic:menu'/} <!-- 引入菜单公共模板文件 -->
{include file='basic:nav'/} <!-- 引入导航公共模板文件 -->
<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
<!-- Content Header (Page header) -->
  <section class="content-header">
    <h1>
      {block name="page_header"}
        <!-- 模板继承: 页面标题 -->
      {/block}
    <small>
      {block name="page_header_small"}
        <!-- 模板继承: 页面副标题 -->
      {/block}
    </small>
  </h1>
  <ol class="breadcrumb">
    {block name="nav_tips"}
      <!-- 模板继承: 导航提示 -->
    {/block}
  </ol>
</section>

<!-- Main content -->
<section class="content container-fluid">
  <!-- 定义全局成功提示信息框 -->
  <div id="showSuccessTips" class="callout callout-success" style="
display: none;">
    <h4>成功! </h4>
    <p></p>
  </div>
  <!-- 定义全局失败提示信息框 -->
  <div id="showErrorTips" class="callout callout-danger" style="
display: none;">
    <h4>错误! </h4>
    <p></p>
  </div>
  {block name="content"}
  <!-- 模板继承: 定义主体内容 -->
  {/block}
</section>
<!-- /.content -->
</div>
<!-- /.content-wrapper -->
{include file='basic:footer'/} <!-- 引入底部公共模板文件 -->
{block name='script'}
  <!-- 模板继承: 定义JavaScript脚本文件 -->
{/block}
</body>
</html>
```

基础模板布局文件中使用了 AdminLTE 自带的 starter.html 示例文件布局,使用了如下基本的模板引入命令:

```
{include file='basic:header'/'} <!-- 引入顶部公共模板文件 -->
{include file='basic:menu'/'} <!-- 引入菜单公共模板文件 -->
{include file='basic:nav'/'} <!-- 引入导航公共模板文件 -->
```

另外，还在部分区域使用模板继承，例如页面主体中的标题和副标题，在基础模板布局文件中默认为空，这样就保证子模板继承使用时，只会继承样式结构，而非固定的内容：

```
<section class="content-header">
  <h1>
    {block name="page_header"}
      <!-- 模板继承: 页面标题 -->
    {/block}
  <small>
    {block name="page_header_small"}
      <!-- 模板继承: 页面副标题 -->
    {/block}
  </small>
.....
```

## 2. 定义公共模板头文件：header.html

在基础模板布局文件中，可以发现引入了很多其他模板文件，实现模板的分区管理。头文件主要用来定义格式类库和基本样式。在 basic 目录下新建 header.html 文件，核心代码如下：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>{:config('SITES_NAME')}</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta content="width=device-width, initial-scale=1, maximum-scale=1,
  user-scalable=no" name="viewport">
  <link rel="stylesheet" href="__STATIC__/admin/libs/adminlte/bower_
  components/bootstrap/dist/css/bootstrap.min.css">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="__STATIC__/admin/libs/adminlte/bower_
  components/font-awesome/css/font-awesome.min.css">
  <!-- Ionicons -->
  <link rel="stylesheet" href="__STATIC__/admin/libs/adminlte/bower_
  components/Ionicons/css/ionicons.min.css">
  <!-- Theme style -->
  <link rel="stylesheet" href="__STATIC__/admin/libs/adminlte/dist/css
  /AdminLTE.min.css">
.....
  <!-- jQuery 3 -->
  <script src="__STATIC__/admin/libs/adminlte/bower_components/jquery/
  dist/jquery.min.js"></script>
  <script src="__STATIC__/admin/js/common.js"></script>
  <script src="__STATIC__/admin/js/submit.js"></script>
</head>
```

模板文件主引入了 AdminLTE 的类库文件：

```
<link rel="stylesheet" href="__STATIC__/admin/libs/adminlte/dist/css/AdminLTE.min.css">
```

.....

也引入了内容管理框架本身公共脚本文件，此时可以先在 js 目录下手动创建两个空 JavaScript 脚本文件，便于开发者编写后续的自定义脚本样式：

```
<script src="__STATIC__/admin/js/common.js"></script>
<script src="__STATIC__/admin/js/submit.js"></script>
```

### 3. 定义顶部导航公共模板文件：nav.html

顶部导航定义了三部分内容，主要展示和实现以下功能：

- 站点信息，后台管理站点的名称展示，使用 ThinkPHP5 助手方法 config() 读取。
- 顶部导航，展示顶级导航信息。
- 用户信息，展示用户 ID、用户名和注销登录等操作入口。

继续使用 AdminLTE 模板样式，定义核心代码如下：

```
<body class="hold-transition skin-blue sidebar-mini">
<div class="wrapper">
  <!-- Main Header -->
  <header class="main-header">
    <!-- Logo -->
    <a href="/admin" class="logo">
      <!-- 读取后台管理站点名称 PC 模式 -->
      <span class="logo-mini">{:config('SITES_NAME')}</span><!--使用助手方法读取-->
      <!-- 读取后台管理站点名称 移动端模式 -->
      <span class="logo-lg">{:config('SITES_NAME')}</span><!--使用助手方法读取-->
    </a>
    <!-- 顶部导航 -->
    <nav class="navbar navbar-static-top" role="navigation">
      <!-- Sidebar toggle button-->
      <a href="#" class="sidebar-toggle" data-toggle="push-menu"
role="button">
        <span class="sr-only">Toggle navigation</span>
      </a>
      <!-- 一级导航 -->
      <ul class="nav navbar-nav">
        <li class=""><a href="">导航名</a></li>
      </ul>
      <!-- 登录用户会话信息展示 -->
      <div class="navbar-custom-menu">
        <ul class="nav navbar-nav">
          <!-- User Account Menu -->
          <li class="dropdown user user-menu">
            <!-- Menu Toggle Button -->
            <a href="#" class="dropdown-toggle" data-toggle="dropdown">
              <!-- 显示用户名 -->
```

```

        <span class="hidden-xs">测试用户名</span>
    </a>
    <ul class="dropdown-menu">
        <!-- 显示用户详细信息 -->
        <li class="user-header" style="height: 100px;">
            <p>用户 ID: 测试 ID</p>
            <p>用户名: 测试用户名</p>
        </li>
        <!-- 用户操作区域-->
        <li class="user-footer">
            <div class="pull-left">
            </div>
            <div class="pull-right">
                <a href="{:url('public_user/logout')}" class=
                    "btn btn-default btn-flat">注销登录</a>
            </div>
        </li>
    </ul>
</li>
</ul>
</div>
</nav>
</header>

```

#### 4. 新建左侧菜单组基础模板文件: menu.html

用户选择顶部的一级菜单后, 左侧菜单组会显示当前分类下的二级分类, 在 basic 目录下新建 menu.html 文件, 核心代码如下:

```

<aside class="main-sidebar">
    <section class="sidebar">
        <!-- 左侧菜单组 -->
        <ul class="sidebar-menu" data-widget="tree">
            <li class="header">导航</li>
            <li class="treeview active">
                <a href="#"><i class="fa fa-link"></i> <span>菜单组</span>
                    <span class="pull-right-container">
                        <i class="fa fa-angle-left pull-right"></i>
                    </span>
                </a>
                <ul class="treeview-menu">
                    <li class="active">
                        <a href="/">菜单组内容</a>
                    </li>
                </ul>
            </li>
        </ul>
    </section>
</aside>

```

#### 5. 定义底部基础模板文件: footer.html

此文件用来引入公共的 JavaScript 类库文件, 底部引入的方式可以让脚本的载入不影

响样式的预先加载。核心代码定义如下：

```
<footer class="main-footer">
  <div class="pull-right hidden-xs">
    布局底部说明
  </div>
  <strong>Copyright &copy; 2018 <a href="#">Company</a>.</strong> 权限说明.
</footer>
<!-- Bootstrap 3.3.7 -->
<script
src="__STATIC__/_/admin/libs/adminlte/bower_components/bootstrap/dist/js/
bootstrap.min.js"></script>
<!-- AdminLTE App -->
<script
src="__STATIC__/_/admin/libs/adminlte/dist/js/adminlte.min.js"></script>
```

## 6. 测试基础模板布局文件

在完成以上几个步骤后，为了测试基础模板样式，先在 application\admin\controller 目录下找到 index.php 控制器文件，修改 index() 方法如下：

```
// 测试基础模板布局文件
public function index()
{
    return $this->fetch();
}
```

再在 application\admin\view 下新增 index 目录，创建模板文件 index.html 文件，然后使用 extend 和 block 标签实现模板的继承操作，核心代码如下：

```
{extend name="basic:layout"}
{block name="page_header"}
标题
{/block}
{block name="page_header_small"}
副标题
{/block}
{block name="content"}
    显示表单、列表等
{/block}
```

在浏览器中访问地址：<http://wangcmf.com/admin/index/index>，发现主体内容已经展示为自定义的形式，效果如图 9-9 所示。

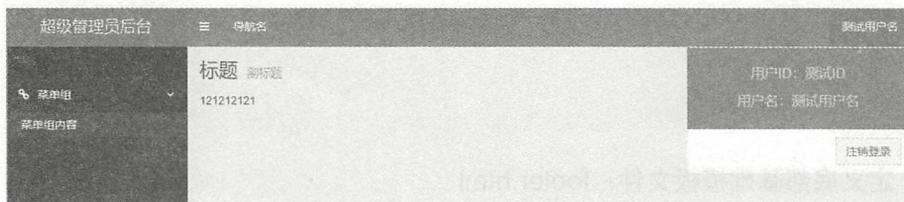


图 9-9 后台管理模板基础布局效果

提示：域名 `http://wangcmf.com` 为本地虚拟域名，开发者可以自行配置与定义。

## 9.3 用户模块——用户登录与验证

后台管理模块对于用户权限的控制，比常见的网站系统更严格，除了登录和注销登录功能页外，用户都不能在未登录状态下访问网站。本节重点讲解用户模块的登录功能。

### 9.3.1 数据结构设计与基础模板开发

新用户通过后台添加或者前台注册才能访问网站，所以实现内容开发系统的第一步，就需要实现用户登录功能。常见的用户登录会话验证机制，如图 9-10 所示。

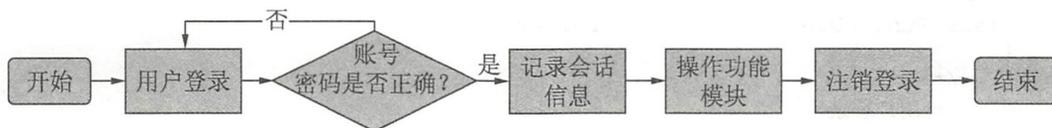


图 9-10 用户登录会话验证机制

下面具体介绍该功能的实现步骤。

#### 1. 数据表设计

新建数据库 `wangcmf`，新建用户表 `user`（表前缀 `db_`），以手机号（`mobile`）作为用户登录的唯一标识，密码（`password`）存储的字符串使用 `md5` 序列化，其他字段可以参见结构说明。代码如下：

```

CREATE TABLE `db_user` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL DEFAULT '',
  `status` int(10) NOT NULL DEFAULT '0' COMMENT '-1: 已删除, 0: 禁用, 1: 正常',
  `create_time` int(10) NOT NULL DEFAULT '0' COMMENT '创建时间',
  `update_time` int(10) NOT NULL DEFAULT '0' COMMENT '更新时间',
  `type` tinyint(2) NOT NULL DEFAULT '0' COMMENT '0: 普通用户, 1: 管理员',
  `mobile` varchar(20) NOT NULL DEFAULT '' COMMENT '手机号, 唯一性标识',
  `password` varchar(255) NOT NULL DEFAULT '' COMMENT '密码',
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE KEY `mobile` (`mobile`) USING BTREE COMMENT '手机号不能重复'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=COMPACT;
  
```

因为目前还没有用户管理模块，所以手动在数据库中插入一个测试用户数据，如图 9-11 所示。

Modify	id	name	status	create_time	update_time	type	mobile	password
编辑	1	wangjialin	1	0	0	0	13511112222	14e1b600bfd579f47433b88e8d85291

图 9-11 在数据表中插入测试数据

需要说明的是，记录中 password 的值是根据以下 PHP 代码生成的：

```
echo md5(md5('123456'));
```

## 2. 登录表单模板

在 application\admin\controller 目录下新建 PublicUser.php 控制器文件，文件继承自 ThinkPHP 的控制器类，新建 login() 和 logout() 方法，分别用来展示用户登录页面和执行注销登录页面。具体步骤如下：

### (1) 新建控制器类

```
/**
 * 用户登录、注销操作类
 */
class PublicUser extends Controller
{
    /**
     * 用户登录
     */
    public function login()
    {
        // TODO::显示用户登录表单
        return $this->fetch();
    }
    /**
     * 注销登录
     */
    public function logout() {
        // TODO::执行注销用户操作
    }
}
```

### (2) 新增用户登录模板

AdminLTE 提供了注册登录的示例样式，这里直接拿来使用。在 application\admin\view 下新建 public\_user 目录，在其中增加 login.html 文件。其核心代码如下：

```
01 <!-- 登录提交到 login 方法 -->
02 <form action="{:url('PublicUser/login')}" method="post">
03     <div class="form-group has-feedback">
04         <input type="text" name="mobile" class="form-control" placeholder=
           "手机号" value="{ $data.mobile|default=' ' }">
05     </div>
06     <div class="form-group has-feedback">
07         <input type="password" name="password" class="form-control"
           placeholder="密码">
08     </div>
09 </div class="row">
```

```
10     <div class="col-xs-4">
11         <!-- 表单提交 -->
12         <button type="submit" class="btn btn-primary btn-block
            btn-flat"
            >登录</button>
13     </div>
14 </div>
15 <div class="row">
16     <div class="col-xs-12">
17         <!-- 显示错误提示信息 -->
18         {&#x24;show_error_tips|default='' }
19     </div>
20 </div>
21</form>
```

### 9.3.2 完成用户登录操作

为了处理登录表单页提交的请求数据，修改 `login()` 方法，增加以下代码，用来实现对 POST 请求数据的处理：

```
public function login()
{
    // 判断是否是用户提交
    if(request()->isPost())
    {
        // 获取 POST 提交的数据
        $data = input('post. ');
        // 表单提交数据验证
        $checkResult = $this->checkFormData($data);
        if($checkResult !== '')
        {
            // 显示错误信息
            $this->assign('show_error_tips' , $checkResult);
        }
        // 在表单中展示上一次的提交数据
        $this->assign('data' , $data);
        // 执行登录操作
        if(!$this->doLogin($data))
        {
            $this->assign('show_error_tips' , '用户名或者密码错误');
        }
        else
        {
            $this->success('登录成功' , url('index/index'));
        }
    }
    return $this->fetch();
}
```

在代码中，使用 `checkFormData()` 方法，封装 ThinkPHP 5 自带的验证类 (`Validate`) 方法，来验证用户输入的手机号和密码是否格式正确。核心代码定义如下：



```
/**
 * 验证表单提交数据
 * @param $data
 * @return array|string
 */
public function checkFormData($data)
{
    // 表单验证规则
    $rule = [
        'mobile' => 'require|regex:\d{11}', // 不能为空|正则验证：必须为 11 位数字
        'password' => 'require|max:25|min:6', // 不能为空|最大长度为 25 位|最小长度为 6 位
    ];
    $msg = [
        'mobile.regex' => '手机号格式错误', // 手机号字段验证错误提示文本
        'password.require' => '密码不能为空' // 密码字段验证错误提示文本
    ];
    $validate = new Validate($rule , $msg); // 使用内置的验证类
    $result = $validate->check($data); // 执行验证操作
    if(!$result){
        return $validate->getError(); // 验证失败，返回验证信息
    }
    return '';
}
```

在表单提交数据的格式验证通过后，执行控制器的 doLogin()方法实现在数据库中检索是否存在此用户，同时验证用户输入的密码是否正确。方法定义如下：

```
private function doLogin($data)
{
    if(!empty($data))
    {
        // 构建查询数据
        $map = [
            'mobile' => $data['mobile'],
            'password' => sys_md5($data['password']),
            'status' => 1
        ];
        // 查询用户是否存在
        $user_info = Db::name('user')->field('id as user_id,name,mobile')
        ->where($map)->find();
        // 若查询不到用户则返回 false
        if($user_info === null)
        {
            return false;
        }
        // 记录会话信息
        session('user_auth' , $user_info);
        return true;
    }
    return false;
}
```



此方法对用户输入的参数进行数据库检索，根据检索返回值判断是否返回 true 或者 false。sys\_md5()的方法定义在 application\common.php 文件中，定义如下：

```
if(!function_exists('sys_md5'))
{
    // 系统密码加密方法
    function sys_md5($string)
    {
        return md5(md5($string));
    }
}
```

### 9.3.3 用户登录状态验证与注销登录

完成功能代码的开发，这里进行功能的使用与测试，查看用户登录功能是否可以正常地使用。成功登录后，需要记录和验证用户的会话状态，同时还需要提供注销登录的相关方法。

#### 1. 使用登录功能

在浏览器中访问地址：[http://wangcmf.com/admin/public\\_user/login.html](http://wangcmf.com/admin/public_user/login.html)，输入错误的手机号码和密码，展示效果如图 9-12 所示。

随后输入正确的手机号、密码后，提示登录成功，如图 9-13 所示。



图 9-12 对错误的信息进行提示

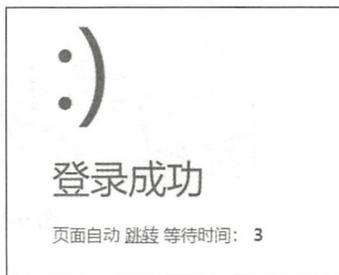


图 9-13 ThinkPHP 5 自带的成功提示页面

#### 2. 替换系统提示跳转页的模板

使用系统自带的 success()方法，会自动调用 ThinkPHP 5 默认页面重定向模板样式，为了实现效果的统一，这里修改为自定义样式。

首先在 basic 公共模板文件夹下，新增 success.html 和 error.html，以 success.html 模板为例，核心代码如下：



```
<div class="error-content">
  <h3><i class="fa text-yellow"></i><?php echo(strip_tags($msg));?></h3>
  <p>
    页面自动<a id="href" href="<?php echo($url);?>">跳转</a>等待时间: <b
    id="wait"><?php echo($wait);?></b>
  </p>

  <form class="search-form">
    <div class="input-group">
      <input type="text" name="search" class="form-control"
        placeholder="搜索">
      <div class="input-group-btn">
        <button type="submit" name="submit" class="btn btn-warning
          btn-flat"><i class="fa fa-search"></i>
        </button>
      </div>
    </div>
    <!-- /.input-group -->
  </form>
</div>
```

随后在 application\admin\config.php 文件中，定义信息提示模板的位置为自定义：

```
<?php
return [
    //默认跳转页面对应的模板文件
    'dispatch_success_tmpl' => 'basic/success',
    'dispatch_error_tmpl'   => 'basic/error',
];
```

执行登录操作，效果如图 9-14 所示。

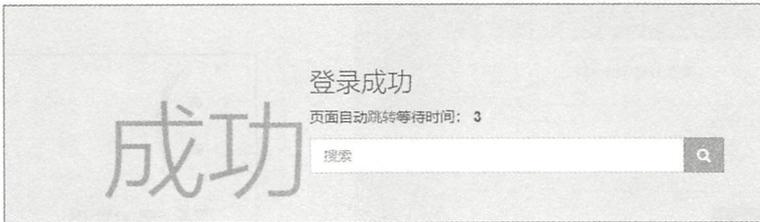


图 9-14 自定义跳转提示模板样式

### 3. 用户登录会话验证

完成上述步骤后，用户登录就可以正常执行了。不过还存在以下几个问题：

- 用户登录后，登录页还可以访问，实现重复登录。
- 用户未登录，也可以访问后台管理首页文件。
- 用户登录后，无法确认当前登录的是哪个用户。

为解决这些问题，需要在 application\common.php 下，定义用户是否登录的公共检测方法 is\_login()，代码如下：



```
// 应用公共文件
if(!function_exists('is_login'))
{
    // 检测用户是否已经登录
    function is_login(){
        // 获取会话中的用户信息
        $user = session('user_auth');
        // 若不存在，则返回 0。否则返回用户信息
        if (empty($user)) {
            return 0;
        }
        return $user;
    }
}
```

完成后，修改 `PublicUser` 控制器的 `login()` 方法，增加登录状态判断，若用户已经登录，系统会自动跳转到 `Index` 控制器的 `index()` 方法。代码如下：

```
public function login()
{
    // 登录后不能再进入登录页
    if(is_login())
    {
        $this->redirect('index/index');
    }
}
.....
```

随后在 `application\admin\controller` 目录下，新增 `Admin.php` 控制器文件，此控制器作为其他控制器的父类，在初始化方法进行用户是否登录的判断。核心代码如下：

```
class Admin extends Controller
{
    // 初始化方法
    protected function _initialize()
    {
        parent::_initialize();

        // 在会话中检测 USER_ID
        if(!defined('USER_ID'))
        {
            define('USER_ID',is_login());
        }
        // 还没登录 跳转到登录页面
        if( !USER_ID ){
            $this->redirect('public_user/login');
        }
    }
}
.....
```

修改 `Index` 控制器的继承关系：

```
.....
class Index extends Admin
{
}
.....
```



此时再去测试，用户在未登录状态下访问首页和已登录状态下访问登录页，就会有相应的跳转和提示了。

#### 4. 在导航条信息上展示用户会话信息

用户登录后会在会话中记录用户信息。修改 `application\admin\view\basic\nav.html` 文件，在模板中展示会话信息。代码如下：

```
<li class="dropdown user user-menu">
  <!-- Menu Toggle Button -->
  <a href="#" class="dropdown-toggle" data-toggle="dropdown">
    <!-- 显示用户名 -->
    <span class="hidden-xs">{:session('user_auth.name')}</span>
  </a>
  <ul class="dropdown-menu">
    <!-- 显示用户详细信息 -->
    <li class="user-header" style="height: 100px;">
      <p>用户 ID: {:session('user_auth.user_id')}</p>
      <p>用户名: {:session('user_auth.name')}</p>
    </li>
```

.....

在登录状态下，后台导航用户信息展示如图 9-15 所示。

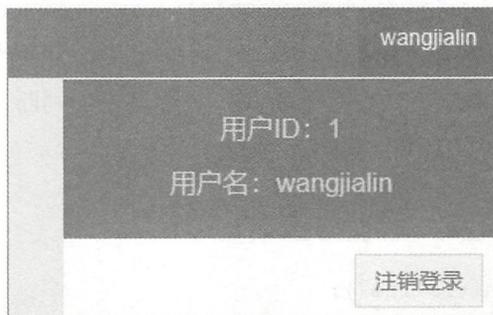


图 9-15 展示当前登录的用户信息

#### 5. 注销登录

修改 `application\admin\view\basic\nav.html` 文件，在用户信息展示区域下，提供新的注销登录的地址：

```
<div class="pull-right">
  <a href="{:url('public_user/logout')}" class="btn btn-default btn-flat">注销登录</a>
</div>
```

在 `PublicUser` 控制器文件中修改 `logout()` 方法，代码如下：

```
/**
```



```
* 注销登录
*/
public function logout(){
    session('user_auth', null);
    $this->success('退出成功!',url('public_user/login'));
}
```

执行注销登录后，页面自动重定向到登录页，提示如图 9-16 所示。

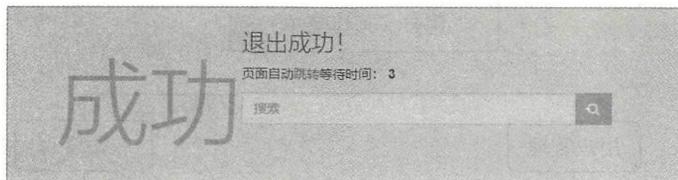


图 9-16 执行注销登录操作

## 9.4 菜单模块

后台管理系统中，各功能的使用需要菜单，权限判定也需要菜单。所以首先开发菜单模块，不仅能为后面的权限模块打下基础，也可以让开发者更好地理解这些模块的作用和联系。

### 9.4.1 数据结构设计

第一级菜单位于顶部导航条上，这类菜单代表功能的合集，如用户和系统等菜单。而每个顶级菜单下又包含二级菜单。这类菜单在页面的左侧展示，一般由列表和特殊单页组成，例如用户列表、权限分组列表和数据库备份单页等，如图 9-17 所示。

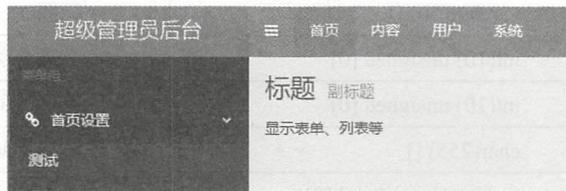


图 9-17 一级菜单和二级菜单

除了可视化的菜单外，还有一些隐性的操作菜单（三级菜单）经常被使用，例如：

- 数据的增加、删除等操作；
- AJAX 操作方法；
- 带特定参数请求的 URL。



为了能够让用户使用菜单，需要和具体的控制器方法关联。每个菜单项都对应一个方法，每个方法又对应一个功能，这样就可以针对菜单进行权限控制。例如，菜单中有用户列表这一项，若不希望某用户拥有此功能的权限，则直接不显示此菜单，或设置对此用户操作无效即可。

以用户模块为例，典型的菜单分层关系图如图 9-18 所示。

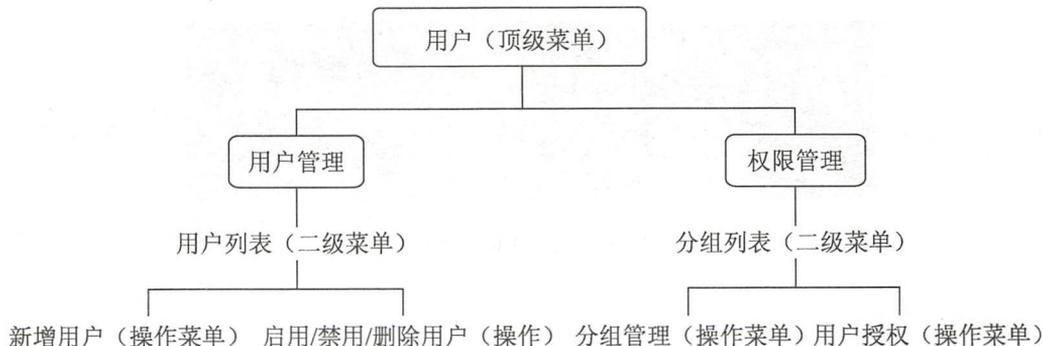


图 9-18 菜单分层关系图

根据上述需求，在设计菜单结构时可以参考无限分类设计原则。其思路是：所有的分类都存储在一张表中，每一级分类的 id 和其父级分类 pid 关联，根据内查询判断分类级别，理论上可以无限向下扩展。

实际设计中，一级分类的 pid 字段值为 0，因为一级分类没有父分类。二级分类的 pid 为一级分类的 id，子分类再以此类推实现无限分类（不过通常到第三级就够用了）。

根据上述原则设计菜单表（db\_menu），其字段说明如表 9-1 所示。

表 9-1 菜单表（db\_menu）字段说明

列（字段）	类 型	注 释
id	int(10) unsigned 自动增量	文档ID
title	varchar(50) []	标题
pid	int(10) unsigned [0]	上级分类ID
sort	int(10) unsigned [0]	排序（同级有效）
url	char(255) []	链接地址
hide	tinyint(1) unsigned [0]	是否隐藏
tip	varchar(255) []	提示
group	varchar(50) NULL []	分组
is_dev	tinyint(1) unsigned [0]	是否仅开发者模式可见
status	tinyint(1) [0]	状态



## 9.4.2 获取菜单数据列表

为了方便演示，先增加一些测试菜单数据，注意菜单的 pid 与 id 的对应关系，如图 9-19 所示。

<input type="checkbox"/> Modify	id	title	pid	sort	url	hide	tip	group	is_dev	status
<input type="checkbox"/> 编辑	1	首页	0	1	Index/index	0			0	1
<input type="checkbox"/> 编辑	16	用户	0	3	User/index	0			0	1
<input type="checkbox"/> 编辑	68	系统	0	4	Menu/index	0			0	1
<input type="checkbox"/> 编辑	69	网站设置	68	1	Config/group	0		系统设置	0	1
<input type="checkbox"/> 编辑	70	配置管理	68	4	Config/index	0		系统设置	0	1
<input type="checkbox"/> 编辑	75	菜单管理	68	5	Menu/index	0		系统设置	0	1
<input type="checkbox"/> 编辑	71	编辑	70	0	Config/edit	0	新增编辑和保存配置		0	1
<input type="checkbox"/> 编辑	72	删除	70	0	Config/del	0	删除配置		0	1
<input type="checkbox"/> 编辑	73	新增	70	0	Config/add	0	新增配置		0	1
<input type="checkbox"/> 编辑	74	保存	70	0	Config/save	0	保存配置		0	1
<input type="checkbox"/> 编辑	119	排序	70	0	Config/sort	1			0	1

图 9-19 手动增加测试菜单数据

在测试数据中，定义了三个一级菜单在导航条上展示：首页、用户和系统。系统菜单下又分为网站设置、配置管理和菜单管理三个菜单。注意，这三个二级菜单拥有同样的分组名：系统设置。最后给菜单管理增加了一些操作菜单（三级菜单），如增、删、改、查等操作。

实现读取菜单分类信息并展示的步骤如下。

### 1. 获取菜单分类信息

在 application\admin\controller 中修改 admin.php 控制器文件，定义 getMenu()方法获取菜单的树形结构。其核心代码如下：

```
/**
 * 获取控制器菜单数组，二级菜单元素位于一级菜单的'_child'元素中
 */
public function getMenu($controller=CONTROLLER_NAME){
    // 会话中是否已经存在菜单
    $menus = session('ADMIN_MENU_LIST.'.$controller);
    if(empty($menus)){
        // 获取一级菜单
        $where['pid'] = 0;
        $where['hide'] = 0; // 默认显示
        $menus['main'] = Db::name('menu')->where($where)->order('sort asc')->field('id,title,url')->select(); // 数据查询
        $menus['child'] = []; // 设置二级菜单子节点
        foreach ($menus['main'] as $key => $item) {
```



```
        if(strtolower(CONTROLLER_NAME.'/'.$ACTION_NAME) == strtolower
($item['url'])){
            $menus['main'][$key]['class']='active'; // 菜单是否添加选中状态
        }
    }
    // 查找当前子菜单
    $pid = Db::name('menu')->where("pid !=0 AND url like '%
{$controller}/".$ACTION_NAME."%'")->value('pid');
    if($pid){
        // 查找到当前一级菜单
        $nav = Db::name('menu')->find($pid);
        if($nav['pid']){
            $nav = Db::name('menu')->find($nav['pid']);
        }
        foreach ($menus['main'] as $key => $item) {
            // 获取当前主菜单的子菜单项
            if($item['id'] == $nav['id']){
                $menus['main'][$key]['class']='active';
                //生成子菜单(child)树
                $groups_list = Db::name('menu')
                    ->field('group')
                    ->where(array('group'=>array('neq',''),'pid' =>$item
                    ['id']))
                    ->distinct(true)
                    ->select();
                $groups = [];
                foreach($groups_list as $k=>$v)
                {
                    $groups[] = $v['group'];
                }
                //获取二级分类的合法url
                $where = [];
                $where['pid'] = $item['id'];
                $where['hide'] = 0;
                $second_urls = Db::name('menu')->where($where)->column
                ('url', 'id');
                // 按照分组生成子菜单树
                foreach ($groups as $g) {
                    $map = array('group'=>$g);
                    if(isset($sto_check_urls)){
                        if(empty($sto_check_urls)){
                            // 没有任何权限
                            continue;
                        }else{
                            $map['url'] = array('in', $sto_check_urls);
                        }
                    }
                }
                $map['pid'] = $item['id'];
                $map['hide'] = 0;
                $menuList = Db::name('menu')->where($map)->field('id,
                pid,title,url,tip')->order('sort asc')->select();
                $menus['child'][$g] = list_to_tree($menuList, 'id',
                'pid', 'operator', $item['id']);
            }
        }
    }
}
```

```

    }
}
}
// 菜单数据缓存到会话, 减少数据查询, 提升性能
session('ADMIN_MENU_LIST.'. $controller, $menus);
}
return $menus;
}

```

## 2. 查看菜单数据结构

除了登录页面外, 后台页面都有菜单展示, 所以需要在 `admin` 控制器初始化方法时查询出所有的菜单, 进行变量的模板置换。

修改 `admin.php` 控制器的 `_initialize()` 方法, 使用 `dump()` 方法打印输出 `getMenus()` 方法返回值。数组结构如下:

```

array(2) {
  ["main"] => array(3) {
    [0] => array(3) {
      ["id"] => int(1)
      ["title"] => string(6) "首页"
      ["url"] => string(11) "Index/index"
    }
    [1] => array(3) {
      ["id"] => int(16)
      ["title"] => string(6) "用户"
      ["url"] => string(10) "User/index"
    }
    [2] => array(4) {
      ["id"] => int(68)
      ["title"] => string(6) "系统"
      ["url"] => string(10) "Menu/index"
      ["class"] => string(6) "active"
    }
  }
  ["child"] => array(1) {
    ["系统设置"] => array(3) {
      [0] => array(5) {
        ["id"] => int(69)
        ["pid"] => int(68)
        ["title"] => string(12) "网站设置"
        ["url"] => string(12) "Config/group"
        ["tip"] => string(0) ""
      }
      [1] => array(5) {
        ["id"] => int(70)
        ["pid"] => int(68)
        ["title"] => string(12) "配置管理"
        ["url"] => string(12) "Config/index"
        ["tip"] => string(0) ""
      }
      [2] => array(5) {

```

```

        ["id"] => int(75)
        ["pid"] => int(68)
        ["title"] => string(12) "菜单管理"
        ["url"] => string(10) "Menu/index"
        ["tip"] => string(0) ""
    }
}
}
}
}

```

### 9.4.3 后台菜单展示

此前已经完成了基础模板布局的开发。要进行动态的菜单展示，还需要修改以下几个地方：

- admin 控制器的初始化方法，查询出所有的菜单，并进行全局变量模板置换。
- 利用基础模板 nav.html 动态展示一级菜单。
- 利用基础模板 menu.html 动态展示二级菜单。

操作步骤如下：

(1) 修改 admin 控制器的初始化方法。找到\_initialize()方法，增加以下代码：

```

// 菜单变量置换到模板中
$this->assign('menu_list' , $this->getMenus());

```

(2) 利用 nav.html 动态化展示导航菜单列表。找到一级导航菜单列表代码，修改如下：

```

<!-- 一级导航 -->
<ul class="nav navbar-nav">
    {notempty name="menu_list.main"}
        {volist name="menu_list.main" id="menu"}
            <li class="{ $menu.class|default='' }"><a href="{:url($menu
                ['url'])}">{$menu.title}</a></li>
        {/volist}
    {/notempty}
</ul>

```

注意：以下代码定义了导航菜单是否是高亮状态：

```

{ $menu.class|default='' }

```

(3) 利用 menu.html，动态化展示二级菜单列表。

```

<ul class="sidebar-menu" data-widget="tree">
    <li class="header">菜单组</li>
    {notempty name="menu_list.child"}
        {volist name="menu_list.child" id="sub_menu"}
            <li class="treeview {if condition='show_menu_active($sub_menu ,
                $active_url) eq true'}active{/if}">
                <a href="#"><i class="fa fa-link"></i> <span>{$key}</span>
                <span class="pull-right-container">

```

```

        <i class="fa fa-angle-left pull-right"></i>
    </span>
</a>
<ul class="treeview-menu">
    {volist name="sub_menu" id="menu"}
        <li class="{if condition='$active_url eq $menu["url"]}'
            active{/if}">
            <a href="{:url($menu.url)}">{$menu.title}</a>
        </li>
    {/volist}
</ul>
</li>
{/volist}
{/notempty}
</ul>

```

在列表循环中使用了 `show_menu_active()` 方法，来判断当前菜单项是否被选中，在 `application\common.php` 中，定义如下：

```

function show_menu_active($sub_menu, $active_url)
{
    $active = false; // 是否选中
    if (!empty($sub_menu) && $active_url) {
        foreach ($sub_menu as $key => $val) {
            if ($active_url == $val['url']) {
                $active = true;
                break;
            }
        }
    }
    return $active;
}

```

(4) 查看菜单效果。登录后，查看菜单是否正常展示，如图 9-20 所示。



图 9-20 动态展示管理菜单

## 9.4.4 菜单管理

菜单管理包含菜单列表、新增（编辑）菜单和修改菜单状态（状态的迁移）3 部分。

## 1. 菜单列表

菜单列表使用 AdminLTE 模板样式，默认展示一级菜单，用户可以单击菜单名从而查看当前菜单的子菜单。下面实现步骤。

### (1) 菜单控制器与列表方法

在 application\admin\controller 下，创建 Menu.php 菜单控制器类文件。新增 index() 方法，用来展示菜单列表。核心代码如下：

```
/**
 * 菜单管理类
 * Class Menu
 * @package app\admin\controller
 */
class Menu extends Admin
{
    /**
     * 菜单列表
     * @return mixed
     * @throws \think\Exception\DbException
     */
    public function index()
    {
        // 查询条件
        $map = [];
        $map['status'] = ['egt' , 0]; // 非删除记录
        $map['pid'] = 0; // 一级菜单
        $pid = input('pid' , 0); // 是否需要查看下级分类
        if($pid)
        {
            $map['pid'] = $pid; // 增加菜单查询条件
        }
        // 带分页的列表查询
        $list = Db::name('menu')->where($map)->order('sort asc')->paginate(10);
        if($list)
        {
            // 变量置换
            $this->assign('_list' , $list);
        }
        // 变量置换
        $this->assign('pid' , $pid);
        return $this->fetch();
    }
}
```

相比 ThinkPHP 3，ThinkPHP 5 的分页功能使用起来更加简单，只需要一个方法即可实现。代码如下：

```
$list = Db::name('menu')->where($map)->order('sort asc')->paginate(10);
```

### (2) 菜单类列表模板

在 application\admin\view 下创建 menu 目录，新增 index.html 模板文件。实现列表输出的核心代码如下：

```
{notempty name="_list"}
  {volist name="_list" id="vo"}
  <tr>
    <td>{$vo.id}</td>
    <td><a href="{:url('index' , ['pid'=>$vo.id])}">{$vo.title}</a></td>
    <td>{$vo.pid|getParentMenuTitle}</td>
    <td>{$vo.group|default='--'}</td>
    <td>{$vo.url}</td>
    <td>{$vo.status|get_status_info}</td>
    <td>
      <a href="{:url('edit',['id'=>$vo['id']])}">编辑</a>
      {if condition='$vo["status"] eq 0'}
      <a class="ajax-get" href="javascript:;" url="{:url('setMenuStatus',['id'=>$vo['id'],'status'=>1])}">启用</a>
      {else/}
      <a class="ajax-get" href="javascript:;" url="{:url('setMenuStatus',['id'=>$vo['id'],'status'=>0])}">禁用</a>
      {/if}
      <a class="ajax-get confirm" href="javascript:;" url="{:url('del',['id'=>$vo['id']])}">删除</a>
    </td>
  </tr>
</volist>
{else /}
  <tr><td colspan="6" class="text-center">暂无数据</td></tr>
</notempty>
```

在模板中使用 getParentMenuTitle()方法，可以知道当前菜单的父菜单，在 application\common.php 中定义。代码如下：

```
function getParentMenuTitle($pid)
{
    return Db::name('menu')->where(['id'=>$pid])->value('title') ?? '--';
}
```

在浏览器中访问地址 http://wangcmf.com/admin/menu/index.html，效果如图 9-21 所示。

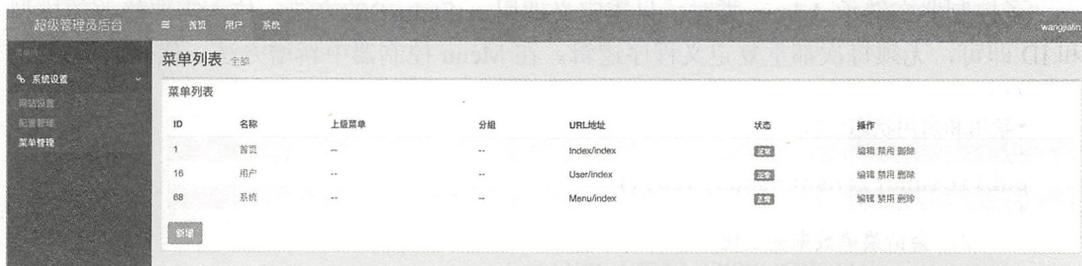


图 9-21 菜单管理之菜单列表

## 2. 修改菜单状态

在管理系统中，每个列表基本都带有状态的迁移操作，如启用（禁用）、删除等。在 Admin 控制器类中，定义全局的状态迁移方法可以提高方法的复用性。其核心代码如下：

```
// 表记录变更状态方法
protected function setStatus($model = null)
{
    if($model !== null)
    {
        $id = input('id');
        $status = input('status');

        if($id !== null && $status !== null)
        {
            $map = [];
            $map['id'] = $id;
            $data['status'] = $status;
            $model->where($map)->update($data);
            switch ($status)
            {
                case -1 :
                    $this->success('删除成功! ');
                    break;
                case 1 :
                    $this->success('启用成功! ');
                    break;
                case 0 :
                    $this->success('禁用成功! ');
                    break;
            }
        }
    }
    $this->error('参数错误! ');
}
```

子控制器在继承 Admin 类时，只需定义调用 setStatus()的方法，传入需要修改的模型和 ID 即可，无须每次都重复定义程序逻辑。在 Menu 控制器中新增方法，代码如下：

```
/**
 *禁用和启用状态迁移
 */
public function setMenuStatus()
{
    // 会话菜单数据初始化
    session('ADMIN_MENU_LIST', NULL);
    // 更新记录状态
    return $this->setStatus(Db::name('menu'));
}
```

修改菜单状态可使用 AJAX 提交操作。在定义状态变更按钮时，给超链接增加了一些预定义样式类。以删除按钮为例，相应的 HTML 代码定义如下：

```
<a class="ajax-get confirm" href="javascript:;" .....>删除</a>
```

两个类名的定义结合 JavaScript 脚本和 AJAX 操作，实现以下效果。

- **ajax-get**: 用户执行单击操作后，发送的是 GET 类型的 AJAX 请求，而非页面直接重定向。这样做可以在页面不刷新的情况下异步完成操作，从而提高用户体验。
- **confirm**: 按钮的请求发送前需要用户进行确认，否则停止发送，从而减少了误操作的概率。

在根目录 public/static/admin/js 目录下修改 common.js 脚本文件，增加以下代码：

```
$(function(){
//处理 ajaxget 请求
$('.ajax-get').click(function(){
    var target;
    //获取当前操作的按钮对象
    var that=this;
    //判断是否进行确认操作
    if($(this).hasClass('confirm')){
        if(!confirm('确认要执行该操作吗?')){
            return false;
        }
    }
    //获取以属性定义的请求地址
    if((target=$(this).attr('url'))){
        //使用 POST 的方式发送 AJAX 请求，可以修改为 AJAX 或者 GET
        $.post(target, {}, function(data) {
            //判断返回值是否成功
            if(data.code==1) {
                //如果按钮中有 URL 属性，则提示信息的同时提示马上要进行跳转
                if(data.url) {
                    $('#showSuccessTips').show().find('p').text(data.
                        msg+', 正在执行跳转...');
                }else{
                    //只显示信息，不执行其他操作
                    $('#showSuccessTips').show().find('p').text(data.msg);
                }
                //固定时间后，根据按钮是否拥有 URL 属性，进行页面的重定向或者刷新
                setTimeout(function(){
                    if(data.url){
                        window.location.href=data.url;
                    }else{
                        window.location.reload();
                    }
                },1000);
            }else{
```

```

        //显示错误信息
        $('#showErrorTips').show().find('p').text(data.msg);
        //进行页面的重定向
        setTimeout(function(){
            if(data.url){
                window.location.href=data.url;
            }
        },1000);
    }
    })
}
return false;
});
});

```

因为菜单的记录具有特殊性，不同层级下的菜单可能会重名，所以在 Menu 控制器中定义了物理删除菜单的方法，防止数据重复增加。代码如下：

```

/**
 *删除菜单
 */
public function del()
{
    if(request()->isAjax())
    {
        $id=input('id',0);
        if(!$id)
        {
            $this->error('删除失败，参数错误');
        }
        $map['id']=$id;
        $return=Db::name('menu')->where($map)->delete();
        if(!$return)
        {
            $this->error('数据库操作失败！');
        }
        //会话菜单信息初始化
        session('ADMIN_MENU_LIST',NULL);
        $this->success('删除成功！');
    }
    $this->error('数据异常');
}

```

完成后，在模板文件 header.html 中引入 common.js 文件。代码如下：

```
<script src="__STATIC__/admin/js/common.js"></script>
```

在菜单列表中成功进行状态的迁移操作，例如禁用首页菜单，效果如图 9-22 所示。

ID	名称	上级菜单	分组	URL地址	状态	操作
1	首页	--	--	Index/Index	<input checked="" type="checkbox"/>	编辑 启用 删除

图 9-22 表记录的状态迁移

### 3. 新增（编辑）菜单

在 Menu 控制器中新增菜单方法 add(), 代码如下:

// 新增菜单

```
public function add()
{
    // 判断是否是 POST
    if(request()->isPost())
    {
        // 数据获取
        $data = input('post.');
        // 数据验证
        $validate = new MenuValidate();
        $result = $validate->check($data);
        if(!$result)
        {
            $this->error($validate->getError());
        }
        // 数据写入
        $data['status'] = 1;
        $insertId = Db::name('menu')->insert($data);
        if($insertId !== false)
        {
            // 更新会话中的菜单信息
            session('ADMIN_MENU_LIST', NULL);
            $this->success('保存成功');
        }
        $this->error('数据库操作失败!');
    }

    // 显示已有菜单树形结构
    $this->assign('menus', $this->getMenuTreeList());
    $this->assign('active_url', 'Menu/index');
    return $this->fetch();
}
```

此方法把新增菜单页面的展示和处理定义在一个方法内, 通过对请求数据是否为 POST (用户提交) 进行判断来获取提交的数据。在数据写入部分, 先使用 ThinkPHP 5 自带的 Validate 验证类进行数据的校验。代码如下:

```
$validate = new MenuValidate();
$result = $validate->check($data);
if(!$result)
{
```

```

        $this->error($validate->getError());
    }

```

随后进行数据的写入，同时清除菜单在会话中的缓存（防止每次查询的）。代码如下：

```

// 更新会话中的菜单信息
session('ADMIN_MENU_LIST' , NULL);
$this->success('保存成功');

```

而在新增菜单的展示部分，为了方便用户选择当前菜单的父级菜单而把所有的菜单处理为树形结构展示，如图 9-23 所示。

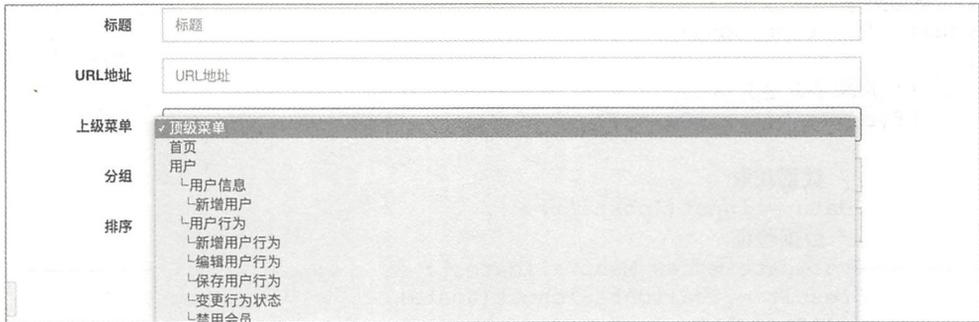


图 9-23 菜单选择的树形结构

获取树形结构菜单的方法为 `getMenuTreeList()`。该方法的定义如下：

```

/**
 * 获取树形结构菜单
 */
private function getMenuTreeList()
{
    // 查询所有的菜单列表
    $menus = Db::name('Menu')->select();
    // 初始化树形结构处理方法类
    $treeObj = new Tree();
    // 转换菜单列表为带格式的树形结构列表
    $menus = $treeObj->toFormatTree($menus);
    $menus = array_merge([[ 'id'=>0, 'title_show'=>'顶级菜单' ]], $menus);
    return $menus;
}

```

`Tree` 类在项目根目录下的 `extend`（扩展）目录。转换树形结构方法的代码如下：

```

/**
 * 将格式数组转换为树
 * @param array $list
 * @param integer $level 进行递归时传递用的参数
 */
private function _toFormatTree($list, $level=0, $title = 'title') {
    foreach($list as $key=>$val)
    {
        // 根据菜单的层级（第一级还是第二级）来定义前面有几个空格

```

```

$tmp_str=str_repeat("&nbsp;",$level*2);
$tmp_str.="L";
$val['level'] = $level;
$val['title_show']
=$level==0?$val[$title]."&nbsp;":$tmp_str.$val[$title]."&nbsp;";
// 判断当前的菜单是否有子菜单 (_child 数组存放), 没有直接把当前菜单放入全局
菜单变量 formatTree
if(!array_key_exists('_child',$val)){
    array_push($this->formatTree,$val);
}else{
    $tmp_ary = $val['_child'];
    unset($val['_child']);
    array_push($this->formatTree,$val);
    // 进行下一层递归操作
    $this->_toFormatTree($tmp_ary,$level+1,$title);
}
}
return;
}

```

`_toFormatTree()`常用的参数有两个, `$list` 为包含所有菜单的一维数组, `$level` 指定菜单的层级。获取子菜单的方式为递归查询方法, 代码如下:

```

// 进行下一层递归操作
$this->_toFormatTree($tmp_ary,$level+1,$title);

```

在模板目录下新增 `add.html`, 表单使用 AdminLTE 模板样式, 表单提交部分的代码如下:

```

<div class="box-footer">
    <a href="javascript:history.back();" class="btn btn-default">返回上一级
    </a>
    <button type="button" target-form="form-horizontal" class="btn
    btn-info pull-right ajax-post">保存</button>
</div>

```

与列表状态变更操作中 AJAX 预定义样式类 `ajax-get` 类似, 这里使用了预定义类 `ajax-post` 来定义表单的无刷新提交。

在项目 `public\static\admin\js` 目录下新增 `submit.js` 脚本文件, 在 `header.html` 模板文件中引入。因为和之前定义 `ajax-get` 处理方法类似, 这里仅给出部分代码:

```

$(function(){
    $('.ajax-post').click(function(){
        // 获取 dom 对象
        var _self = $(this);
        // 获取表单对象
        var _form_class = _self.attr('target-form');
        if(!_form_class)
        {
            return false;
        }
        var _form = $('.' + _form_class);
        // 获取表单提交的地址
    }
}

```

```
var _action = _form.get(0).action;
if(!_action)
{
    return false;
}
// 防止重复提交
_self.text('正在提交').attr('disabled' , 'disabled');
$('#showErrorTips').hide();
// ajax 提交, 提交表单数据
$.post(_action , _form.serialize() , function(data){
.....
```

最后测试新增功能, 在系统下增加数据库管理菜单, 如图 9-24 所示。

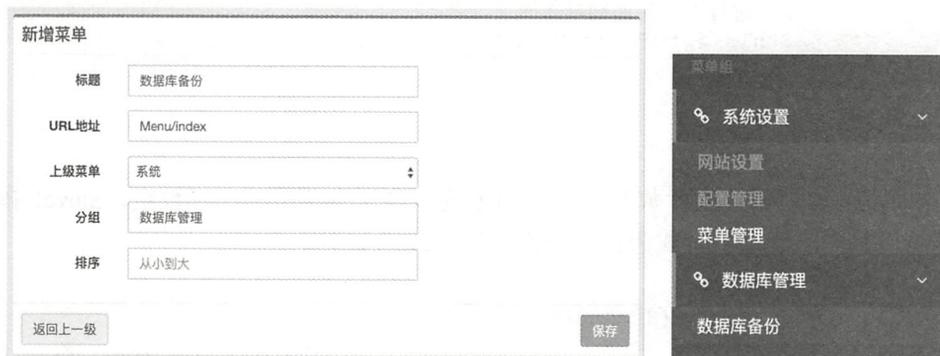


图 9-24 在系统下新增数据库管理菜单管理

**提示:** 编辑菜单操作与新增菜单操作类似, 区别在于展示数据查询与数据更新, 具体可以参考源代码。

# 第 10 章 内容管理框架实战—— 配置和权限管理

在第 9 章中介绍了开发一个实用的内容管理框架需要有哪些基本模块，同时也实现了程序框架的搭建、用户登录和菜单管理等基础功能。本章将继续介绍后续内容：配置管理与权限管理。完成本章的学习后，开发者就可以定制自己的内容管理框架，并根据实际项目进行相应的调整。

## 10.1 配置管理

配置管理的实现步骤跟菜单管理类似，管理后台可以对配置项进行管理。在开发过程中，使用 ThinkPHP 框架自带的 `config()` 方法，可以随时读取配置文件中的配置项并使用。

### 10.1.1 程序与数据结构设计

在网站系统设计中，配置项的使用无处不在，从分页默认显示页数到站点关键字，都需要使用配置项进行管理。ThinkPHP 5 框架级别的配置信息存储在各个模块下的 `config.php` 文件中，如全局配置文件部分代码如下：

```
// 应用命名空间
'app_namespace'      => 'app',
// 应用调试模式
'app_debug'          => true,
// 应用 Trace
'app_trace'          => false,
// 应用模式状态
'app_status'         => '',
.....
```

配置项存储在 `config.php` 文件中，无法满足及时更新的需求，所以存储在 MySQL 数据库中更加合适。在内容管理框架中，实现配置管理的配置模块与菜单模块类似，功能如图 10-1 所示。

配置项存储的格式与文件配置中的格式一样，都是键值对形式，所以表设计并不复杂。不过 MySQL 不支持原生的 PHP 数组格式，所以若配置的值是数组，就需要 PHP 进行解

析操作。新增配置表（db\_config）字段说明如表 10-1 所示。

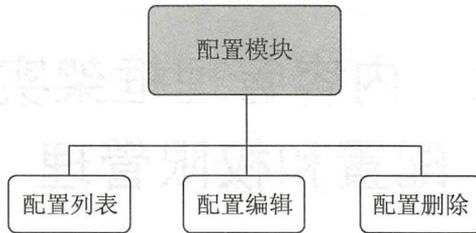


图 10-1 配置模块相关功能

表 10-1 配置表字段说明

列	类 型	注 释
id	int(10)	主键自增长
name	varchar255	配置名
value	longtext	配置值
status	tinyint(4)	状态（-1：删除，0：禁用，1：正常）
create_time	int(10)	创建时间
update_time	int(10)	更新时间
type	tinyint(2)	配置类型（0：字符串，1：数组）

## 10.1.2 配置列表管理

继续使用第 9 章中的 wangcmf 项目，扩展开发内容管理框架，展示配置管理的实现流程。

### 1. 新增控制器方法

在 application\admin\controller 目录下，新增 Config.php 配置控制器文件，继承自 Admin 控制器，新增 index() 方法，进行列表的展示，代码如下：

```

class Config extends Admin
{
    /**
     * 配置列表展示
     * @return mixed
     * @throws \think\Exception\DbException
     */
    public function index()
    {
        // 构造查询条件
        $map = [];
        $map['status'] = ['egt' , 0];
    }
}
  
```

```

        // 查询分页数据
        $list = Db::name('config')->where($map)->order('create_time
desc')->paginate(10);
        if($list)
        {
            // 变量置换
            $this->assign('_list' , $list);
        }
        // 模板输出渲染
        return $this->fetch();
    }
}

```

## 2. 新增模板文件

随后在 application\admin\view 目录下，新增 config 目录和 index.html 模板文件。列表循环的核心代码如下：

```

{notempty name="_list"}
    {volist name="_list" id="vo"}
        <tr>
            <td>{$vo.id}</td>
            <td>{$vo.name} </td>
            <td>{$vo.value}</td>
            <td>{$vo.create_time|time_format}</td>
            <td>{$vo.update_time|time_format}</td>
            <td>{$vo.status|get_status_info}</td>
            <td>
                <a href="{:url('edit', ['id'=>$vo['id']])}">编辑</a>
                {if condition='$vo["status"] eq 0'}
                    <a class="ajax-get" href="javascript:;" url="{:url
('setConfigStatus', ['id'=>$vo['id'], 'status'=>1])}"
                    ">启用</a>
                {else/}
                    <a class="ajax-get" href="javascript:;" url="{:url
('setConfigStatus', ['id'=>$vo['id'], 'status'=>0])}"
                    ">禁用</a>
                {/if}
                <a class="ajax-get confirm" href="javascript:;" url="{:url
('setConfigStatus', ['id'=>$vo['id'], 'status'=>-1])}">删除</a>
            </td>
        </tr>
    </volist>
    {else /}
        <tr><td colspan="6" class="text-center">暂无数据</td></tr>
{/notempty}

```

其中，time\_format()方法是为了把时间戳格式化为标准日期显示，也可以直接使用系统 date()方法。因为配置管理不涉及物理删除，所以在 Config 控制器中，定义状态变更方法 setConfigStatus()，代码如下：

```

/**
 * 表记录状态迁移

```

```

*/
public function setConfigStatus()
{
    return $this->setStatus(Db::name('config'));
}

```

### 3. 配置列表展示

在配置表中增加两条默认配置项，如图 10-2 所示。

<input type="checkbox"/> Modify	id	name	value	status	create_time	update_time	type
<input type="checkbox"/> 编辑	1	SITES_NAME	超级管理员后台	1	0	1510101126	0
<input type="checkbox"/> 编辑	2	USER_ADMIN	1	1	0	0	0

图 10-2 在配置表中增加两条默认配置项

在浏览器中访问地址：<http://wangcmf.com/admin/config/index>，效果图如图 10-3 所示。



图 10-3 配置列表展示样式

## 10.1.3 配置编辑管理

配置编辑管理主要包含新增配置和编辑配置两个部分。

### 1. 新增配置

当新增配置的时候，需要填写配置名、配置值，还得选择配置值的类型，实现步骤如下。

#### (1) 新增配置模板文件

在 `application\admin\view\config` 目录下，新增 `add.html` 模板文件，表单代码如下：

```

<form class="form-horizontal" action="{:url('add')}" method="post">
    <div class="box-body">
        <div class="form-group"> <!--成功的表单项，需要增加 has-success 预定义类-->
            <label for="" class="col-sm-2 control-label">
                配置名
            </label>
            <div class="col-sm-6"><!--控制 INPUT 元素的宽度-->

```

```

        <input type="text" name="name" class="form-
        control" id="" placeholder="配置名">
    </div>
</div>
<div class="form-group"> <!--成功的表单项，需要增加 has-
success 预定义类-->
    <label for="" class="col-sm-2 control-label">
        配置类型
    </label>
    <div class="col-sm-6"><!--控制 INPUT 元素的宽度-->
        普通: <input type="radio" name="type" class=""
        value="0" checked>
        数组: <input type="radio" name="type" class=""
        value="1">
    </div>
</div>
<div class="form-group">
    <label for="" class="col-sm-2 control-label">
        配置值</label>
    <div class="col-sm-6"><!--控制 INPUT 元素的宽度-->
        <textarea class="form-control" name="value"
        rows="10"></textarea>
    </div>
</div>
<div class="box-footer">
    <a href="javascript:history.back();" class="btn btn-
    default">返回上一级</a>
    <button type="button" target-form="form-horizontal"
    class="btn btn-info pull-right ajax-post">保存</button>
</div>
<!-- /.box-footer -->
</form>

```

## (2) 新增配置方法

在 Config 控制器中，新增 add()方法，包含配置的页面渲染和数据库入库操作，代码如下：

```

/**
 * 新增配置
 * @return mixed
 */
public function add()
{
    // 判断请求类型是否是 POST
    if(request()->isPost())
    {
        // 初始化验证类
        $validate = new ConfigValidate();
        $result = $validate->check(input('post.'));
    }
}

```

```

    if(!$result)
    {
        // 错误信息提示
        $this->error($validate->getError());
    }
    // 构造数据
    $data = input('post.');
    $data['status'] = 1;
    $data['create_time'] = time();
    // 数据入库
    $insertId = Db::name('config')->insert($data);
    if($insertId)
    {
        $this->success('操作成功',url('config/index'));
    }
    $this->error('数据库操作失败');
}
// 置换当前菜单项显示地址
$this->assign('active_url' , 'Config/index');
return $this->fetch();
}

```

在上述代码中，使用了 `Config` 验证类来进行表单数据的验证，为了避免重名冲突，命名空间使用如下代码：

```
use app\admin\validate\Config as ConfigValidate;
```

### (3) 执行新增配置操作

在浏览器中访问 `add()` 控制器方法，增加订单状态配置项 `ORDER_STATUS`，操作如图 10-4 所示。

图 10-4 新增数组类型的配置项

新增配置项的时候，在填写配置值时，为了方便 PHP 解析字符串为数组，格式需要

符合以下要求：

- 第一维数组使用回车（换行符）分隔。
- 第二维数组使用冒号（英文半角状态下的“:”符号）分隔。

完成以上内容，单击“保存”按钮，可以看到配置项已经成功增加，如图 10-5 所示。

ID	配置	值	创建时间
7	ORDER_STATUS	1:已下单 2:已支付 3:已发货 4:已收货 5:已评价	2017-12-25 11:07

图 10-5 在列表上查看新增的配置项

## 2. 编辑配置

编辑配置的操作，与新增配置类似，需要注意以下几点。

### (1) 查询配置信息

在 `edit()`方法中，需要先查询配置项是否存在，再进行其他逻辑操作，代码如下：

```
$id = input('id' ); // 获取配置 id
$map['id'] = $id; // 构建查询条件
$map['status'] = ['egt' , 0];
$info = Db::name('config')->where($map)->find();
if($info === null)
{
    $this->error('参数错误或查询为空');
}
$this->assign('info' , $info);
```

.....

### (2) 更新时间

新增数据的时候，需要写入 `create_time` 字段值，在更新的时候，别忘记写入 `update_time` 字段值，代码如下：

```
// 构造数据
$data = input('post. ');
$data['update_time'] = time();
$map['id'] = $data['id'];
// 数据入库
$updateReturn = Db::name('config')->where($map)->update($data);
```

.....

## 10.1.4 配置使用

ThinkPHP 5 框架自带 `config()`方法，可以直接读取配置文件中的配置项，但是无法直接读取 MySQL 数据库中存储的配置项，所以这里需要在系统初始化的时候，自定义方法，把数据库中配置项读取出来，然后合并到系统的配置文件中，方便全局调用。其流程图如图 10-6 所示。

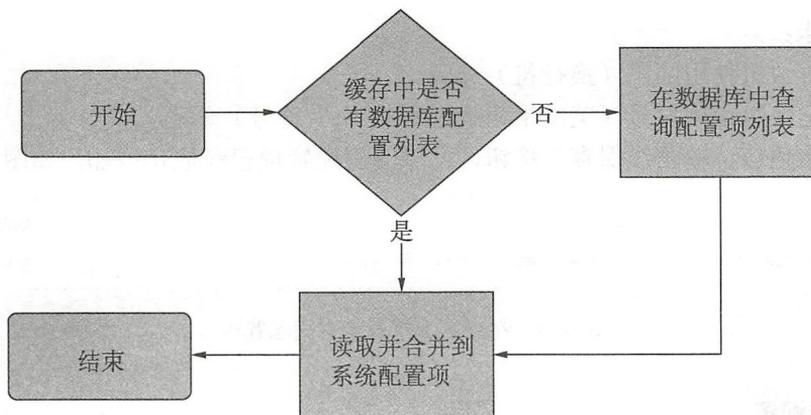


图 10-6 合并数据库配置到系统配置的流程

下面讲解如何实现合并数据库配置到系统配置。

### 1. 实现配置读取类

在 `application\common\controller` 目录下，新增 `Config.php` 配置类文件，定义静态方法 `lists()`，查询所有存储在数据库中的配置项，代码如下：

```

class Config extends Controller
{
    /**
     * 获取数据库中的配置列表
     * @return array 配置数组
     */
    public static function lists(){
        $map = array('status' => 1);
        $data = Db::name('Config')->where($map)->field('type,name,value')->select();

        $config = [];
        if($data && is_array($data)){
            foreach ($data as $value) {
                $config[$value['name']] = self::parse($value['type'], $value['value']);
            }
        }
        return $config;
    }
}
  
```

因为配置项有字符串和数组两种，需要进行格式化处理，代码如下：

```
$config[$value['name']] = self::parse($value['type'], $value['value']);
```

类中使用格式化处理方法 `parse()`，代码如下：

```

/**
 * @param $type 配置项类型，0：字符串，1：数组类型
  
```

```

* @param $value 配置项值
* @return array|false|string[]
*/
private static function parse($type, $value){
    switch ($type) {
        case 1: //解析数组
            $array = preg_split('/[,;\r\n]+/', trim($value, ",;\r\n")); //
            解析一维数组
            if(strpos($value,':')){
                $value = array();
                foreach ($array as $val) { // 解析二维数组
                    list($k, $v) = explode(':', $val);
                    $value[$k] = $v;
                }
            }else{
                $value = $array;
            }
            break;
        }
    }
    return $value;
}

```

## 2. 合并数据库配置到系统配置

在 `application\admin\controller\Admin.php` 控制器中，修改 `_initialize()` 初始化方法，追加配置合并的代码逻辑，代码如下：

```

// 读取缓存中的配置
$config = cache('DB_CONFIG_DATA');
// 判断是否有缓存数据
if(!$config)
{
    // 在数据库中读取所有的配置信息
    $config = Config::lists();
    // 结果写入缓存中
    cache('DB_CONFIG_DATA' , $config);
}
// 数据库配置合并到系统配置
config($config);

```

为了测试合并和数组格式解析是否正常，在 `application\admin\controller\Config.php` 控制器中，新增 `test()` 方法，代码如下：

```

public function test()
{
    dump(config('ORDER_STATUS'));
}

```

在浏览器中访问地址：<http://wangcmf.com/admin/config/test>，显示结果为如下数组类型，说明功能可以正常使用。

```

D:\www\wangcmf\thinkphp\library\think\Debug.php:165:
array (size=5)

```



```

1 => string '已下单' (length=9)
2 => string '已支付' (length=9)
3 => string '已发货' (length=9)
4 => string '已收货' (length=9)
5 => string '已评价' (length=9)

```

最后，为了及时更新缓存中存储的数据，需要修改 add()、edit()方法，代码如下：

```

// 数据入库更新缓存
cache('DB_CONFIG_DATA' , null);
$this->success('操作成功',url('config/index'));
修改 setStatus()方法如下：
// 更新缓存
cache('DB_CONFIG_DATA' , null);
return $this->setStatus(Db::name('config'));

```

此时若删除或禁用了 ORDER\_STATUS 配置项记录，会及时更新（清除）缓存，防止数据不同步导致的潜在问题，如图 10-7 所示。

ID	配置	值	创建时间	更新时间	状态	操作
7	ORDER_STATUS	1 已下单 2 已支付 3 已发货 4 已收货 5 已评价	2017-12-25 11:07	1970-01-01 08:00	禁用	编辑 禁用 删除

图 10-7 禁用 ORDER\_STATUS 配置项

完成操作后，再次访问地址：<http://wangcmf.com/admin/config/test>，发现 ORDER\_STATUS 已经不存在，即显示结果为：

```
D:\www\wangcmf\thinkphp\library\think\Debug.php:165:null
```

## 10.2 权限管理——概念、程序设计与数据库设计

RBAC 是指基于角色的权限访问控制（Role-Based Access Control），是项目系统中最常见的权限管理技术之一。

### 10.2.1 RBAC 权限管理

RBAC 是一种权限管理设计思想，任何编程语言都可以实现。在 RBAC 中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限，这极大地简化了权限的管理。

#### 1. 概念

在一个组织中，角色是为了完成各种工作而创造，用户则依据它的责任和资格来被指派相应的角色，用户可以很容易地从—个角色被指派到另一个角色。角色可依据新的需求和系统的合并而赋予新的权限，权限也可根据需要而从某角色中回收。



下面是常见的未授权访问效果，如图 10-8 所示。

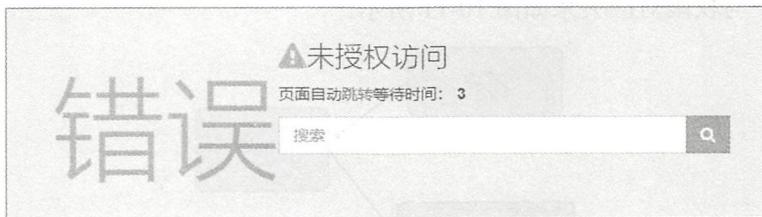


图 10-8 常见的未授权访问效果

在内容管理框架中，权限模块的设计，主要是为了给用户和限制使用权力的功能。而对于开发者而言，权限管理是保证系统安全，主动控制用户的行为。

## 2. 从ACL权限管理到RBAC权限管理

除了 RBAC 权限管理，ACL 也是常见的权限管理设计思想，它的原理非常简单：每一项资源，都配有一个列表，该列表记录的是哪些用户对应哪些操作，当用户试图访问这项资源时，会首先检查该列表中是否有关于当前用户的访问权限，从而确定当前用户能否执行相应的操作。ACL 用户与权限节点的关系，如图 10-9 所示。

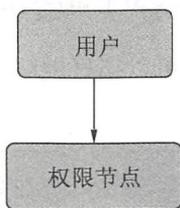


图 10-9 ACL 用户与权限节点的关系

由于 ACL 的简单性，使得它几乎不需要任何基础设施就可以完成访问控制。但同时它的缺点也是很明显的，由于需要维护大量的访问权限列表，ACL 在性能上有明显的缺陷。另外，对于拥有大量用户与众多资源的应用，管理访问控制列表本身变成了非常繁重的工作。

所以 RBAC 可以弥补 ACL 权限控制的不足，RBAC 是把用户按角色进行归类，通过用户的角色，来确定用户能否针对某项资源进行某项操作。RBAC 相对于 ACL 最大的优势就是它简化了用户与权限的管理，通过对用户进行分类，使得角色与权限关联起来，而用户与权限变成了间接关联。RBAC 模型使得访问控制，特别是对用户的授权管理变得非常简单和易于维护，因此有了广泛的应用。RBAC 用户与权限节点的关系，如图 10-10 所示。

举例来说，基于 ACL 的权限控制，有两个默认用户，指定特定功能的权限关系，需要重复地增加两次对应关系。如果要给所有用户去除“新增用户”的权限，则需要修改所有的



对应关系，用户量少还好，用户量一旦上了规模，这类的修改操作，将会非常影响性能。传统 ACL 用户与权限对应关系如图 10-11 所示。

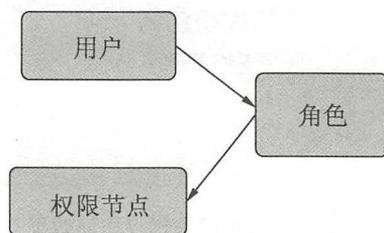


图 10-10 RBAC 用户与权限节点的关系

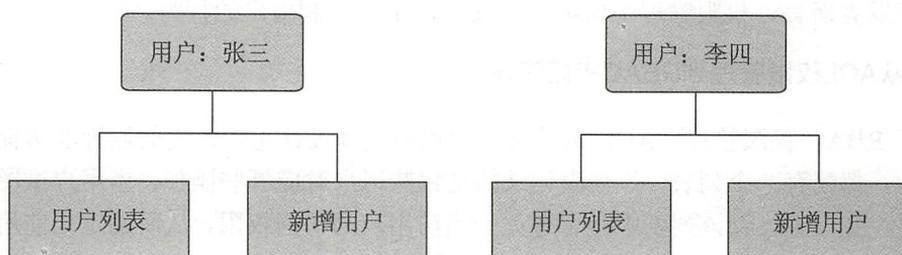


图 10-11 传统 ACL 用户与权限对应关系

使用基于 RBAC 的权限管理，就可以很好地解决 ACL 带来的潜在问题，RBAC 的用户与权限对应关系，如图 10-12 所示。

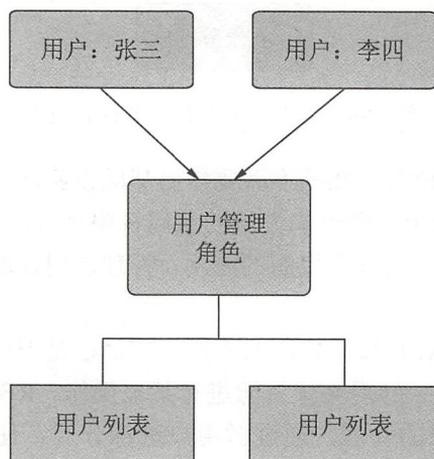


图 10-12 RBAC 用户与权限对应关系

RBAC 模型下，用户与权限节点没有直接关系，增加用户的权限，只需要与关联角色分类即可，同理可得，若想减少某个用户的某个权限，只需要修改角色分类对应的权限节点即可，这样可以减少管理操作对于系统性能的要求，同时也减少了数据量。



## 10.2.2 RBAC 权限管理程序流程与功能设计

为了更好地理解 RBAC 权限管理在实际案例中各功能对应关系，这里提供了典型的 RBAC 关系图示例，模拟了一个现实中的应用场景，如图 10-13 所示。

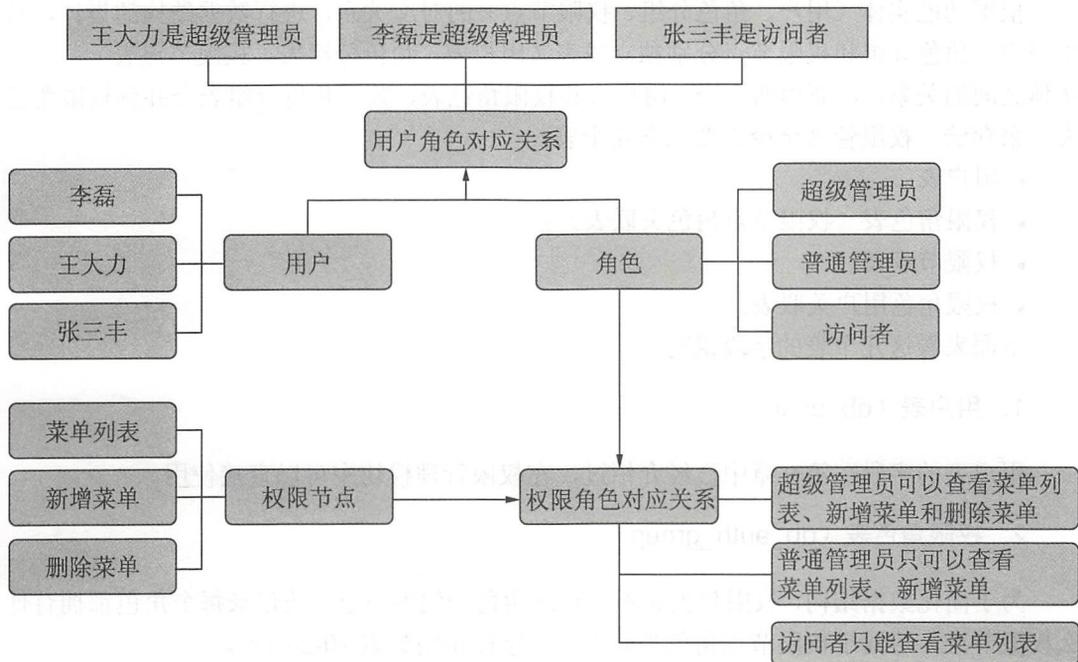


图 10-13 RBAC 权限管理关系图示例

在了解了实际案例中权限管理的关系图，再结合内容管理系统中用户模块和菜单模块（权限节点），实际内容管理系统中的权限模块需要设计开发的功能如图 10-14 所示。

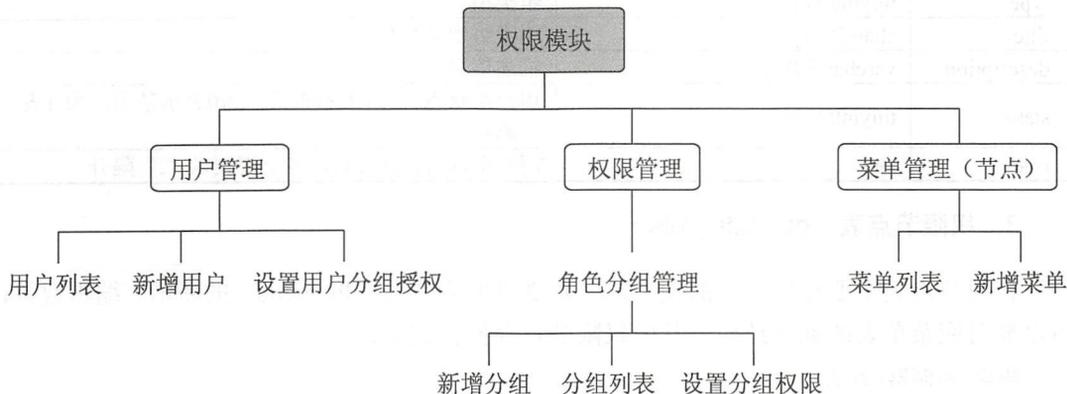


图 10-14 实际内容管理系统中权限模块功能图

提示：以上只是提供了一种比较通用的设计思路，开发者可以根据实际需求进行相应的调整。

### 10.2.3 基于 RBAC 权限管理的数据库设计

根据功能实体（用户、角色分组、权限节点）的对应关系，进行数据结构的设计，其中用户、角色分组和权限节点分别独立成表（用户表、角色分组表、权限节点表），三个实体之间的关系，需要用两个表（用户表和权限角色表，其中角色分组表合并到权限角色表）来存储，权限管理至少需要以下几个表：

- 用户表；
- 权限角色表（权限节点角色关联表）；
- 权限节点表；
- 权限角色用户关联表。

下面来看这几个表的字段说明。

#### 1. 用户表（db\_user）

用户表的字段在第9章中已经介绍过，在权限管理模块中可以直接使用。

#### 2. 权限角色表（db\_auth\_group）

为了简化数据结构，权限角色表不仅记录角色分组的信息，还记录每个角色都拥有什么权限节点，等同于权限节点角色关联表。其字段说明如表10-2所示。

表 10-2 权限角色表字段说明

列（字段）	类 型	注 释
id	mediumint(8) unsigned 自动增量	用户组id,自增主键
module	varchar(20) []	用户组所属模块，默认admin
type	tinyint(4) [0]	组类型
title	char(20) []	用户组中文名称
description	varchar(80) []	描述信息
status	tinyint(1) [1]	用户组状态：为1表示正常，为0表示禁用，为-1表示删除
rules	text	用户组拥有的规则id，多个规则以“,”隔开

#### 3. 权限节点表（db\_auth\_rule）

权限节点表主要存储节点的规格，因为要同步菜单表（db\_menu）的新增、编辑数据，所以要对菜单表的部分结构，其中权限节点存储格式为：

模块/控制器/方法

例如，访问管理后台 Index 控制器的 index()方法的权限，节点直接存储为：



Admin/Index/index

权限节点表字段说明如表 10-3 所示。

表 10-3 权限节点表字段说明

列 (字段)	类 型	注 释
id	mediumint(8) unsigned 自动增量	规则id, 自增主键
module	varchar(20)	规则所属module
type	tinyint(2) [1]	1-url;2-主菜单
name	char(80) []	规则唯一英文标识
title	char(20) []	规则中文描述
status	tinyint(1) [1]	是否有效(0:无效,1:有效)
condition	varchar(300) []	规则附加条件

#### 4. 权限角色用户关系表 (db\_auth\_group\_access)

创建权限角色用户关系表的目的是, 为了建立和存储用户表 user\_id 与权限角色表中 group\_id 的关系, 其字段说明如表 10-4 所示。

表 10-4 权限角色用户关系表字段说明

列 (字段)	类 型	注 释
user_id	int(10) unsigned	用户id
group_id	mediumint(8) unsigned	用户组id

 提示: 在上述表结构中, 为了方便扩展, 有些字段是预留字段, 本书中可能暂时不能使用。

## 10.3 权限管理——角色分组、节点授权与用户模块

在了解了基本的数据结构与程序设计后, 本节开始讲解如何在内容管理框架中, 实现完整的权限管理。

### 10.3.1 权限角色管理

为了更好地融入现有的内容管理框架, 权限模块的开发主要分为以下两大部分。

- 权限管理: 权限角色、用户授权管理功能开发等。
- 权限验证: 权限验证类开发等。

先来看如何实现权限角色管理功能。



## 1. 列表管理

权限角色分组列表的管理，与菜单、配置模块的列表类似，实现步骤如下。

### (1) 新建后台权限管理控制器

在 `application\admin\controller` 目录下，新增 `AuthManager.php` 控制器文件，继承自 `Admin` 父控制器类。

### (2) 权限角色列表管理

权限角色分组列表的展示，实现较为简单，直接查询数据库即可，在 `AuthManager` 控制器中，增加 `index()` 方法，代码如下：

```
/**
 * 权限分组列表
 * @return mixed
 */
public function index()
{
    $map['status'] = ['egt', 0]; // 条件查询
    $group_list = Db::name('AuthGroup')->where( $map )->paginate(10);
    $this->assign('_list', $group_list); // 变量置换
    return $this->fetch();
}
```

在 `application\admin\view` 模板目录下，新增 `auth_manager` 目录，在其中新增 `index.html` 模板文件，核心代码如下：

```
{notempty name="_list"}
    {volist name="_list" id="vo"}
        <tr>
            <td>{$vo.id}</td>
            <td>{$vo.title} </td>
            <td>{$vo.description|default='--'} </td>
            <td>{$vo.status|get_status_info}</td>
            <td>
                <a href="{:url('groupmanage', ['id'=>$vo['id']])}">节点授权</a>
                <a href="{:url('user', ['id'=>$vo['id']])}">用户授权</a>
                <a href="{:url('edit', ['id'=>$vo['id']])}">编辑</a>
                {if condition='$vo["status"] eq 0'}
                <a class="ajax-get" href="javascript:;" url="{:url('setGroupStatus',
                    ['id'=>$vo['id'], 'status'=>1])}">启用</a>
                {else/}
                <a class="ajax-get" href="javascript:;" url="{:url('setGroupStatus',
                    ['id'=>$vo['id'], 'status'=>0])}">禁用</a>
                {/if}
                <a class="ajax-get confirm" href="javascript:;" url="{:url
                    ('setGroupStatus', ['id'=>$vo['id'], 'status'=>-1])}">删除</a>
            </td>
        </tr>
    {/volist}
{else /}
```



```
<tr><td colspan="5" class="text-center">暂无数据</td></tr>
{/notempty}
```

在列表中，定义了节点授权、用户授权等功能菜单，完成效果如图 10-15 所示。

ID	分组名	描述	状态	操作
1	默认用户组	系统默认的用户角色组	正常	节点授权 用户授权 编辑 禁用 删除
2	测试用户	测试用户	正常	节点授权 用户授权 编辑 禁用 删除
3	自定义权限分组	管理员新增的权限分组	正常	节点授权 用户授权 编辑 禁用 删除

图 10-15 权限分组列表效果

## 2. 新增和编辑权限角色分组

在 AuthManager 控制器中，新增 add() 方法，负责新增功能页面的渲染和请求入库操作处理，代码如下：

```
/**
 * 新增分组页面
 */
public function add()
{
    if($this->request->isPost())
    {
        // 数据获取
        $data = input('post.');
        // 数据验证
        $validate = new AuthGroup();
        $result = $validate->check($data);
        if(!$result)
        {
            $this->error($validate->getError());
        }
        // 数据写入
        $data['status'] = 1;
        $data['module'] = 'admin';
        $data['type'] = 1;
        $insertId = Db::name('auth_group')->insert($data);
        if($insertId !== false)
        {
            $this->success('保存成功',url('index'));
        }
        $this->error('数据库操作失败');
    }
}
```



```

    $this->assign('active_url' , 'AuthManager/index');
    return $this->fetch();
}

```

在 `auth_manager` 目录中新增 `add.html` 模板文件，表单提交核心代码如下：

```

<form class="form-horizontal" action="{:url('add')}" method="post">
  <div class="box-body">
    <!-- text -->
    <div class="form-group">
      <label for="inputTitle" class="col-sm-2 control-label">
        分组名
      </label>
      <div class="col-sm-6">
        <input type="text" name="title" class="form-control" id="
          inputTitle" placeholder="标题">
      </div>
    </div>
    <div class="form-group">
      <label for="inputDesc" class="col-sm-2 control-label">
        描述
      </label>
      <div class="col-sm-6">
        <textarea class="form-control" cols="80" rows="5" id="
          inputDesc" name="description"></textarea>
      </div>
    </div>
  </div>
  <!-- /.box-body -->
  <div class="box-footer">
    <a href="javascript:history.back();" class="btn btn-default">
      返回上一级</a>
    <button type="button" target-form="form-horizontal" class="
      btn btn-info pull-right ajax-post">保存</button>
  </div>
  <!-- /.box-footer -->
</form>

```

新增角色分组如图 10-16 所示。



The screenshot shows a web interface titled "分组管理" (Group Management) with a sub-header "新增分组" (Add Group). The form contains two input fields: "分组名" (Group Name) with the value "访问者" and "描述" (Description) with the text "只能查看, 无法进行新增、编辑和删除操作". At the bottom left, there is a button labeled "返回上一级" (Return to Previous Level), and at the bottom right, there is a button labeled "保存" (Save).

图 10-16 新增角色分组

编辑角色分组的实现流程与新增角色分组类似，在这里不再赘述。

### 3. 权限角色分组节点管理

新增角色的权限分组，需要选择哪些权限及节点，操作界面如图 10-17 所示。



图 10-17 为角色分组设置权限节点

在程序实现上，最外层的 box 定义展示的是一级菜单，每个一级菜单下又查询出二级菜单和三级菜单。管理员为某个角色分组勾选权限节点，单击“保存”后完成权限关联。实现步骤如下。

#### (1) 新建权限角色分组管理方法

在 AuthManager 控制器中，新增 groupmanage() 方法，代码如下：

```
/**
 * 分组节点管理
 * @return mixed
 */
public function groupmanage()
{
    $this->updateRules();
    // 获取分组 ID
    $group_id = input('id');
    if(!$group_id)
    {
        $this->error('未查询到权限分组');
    }
    // 查询分组信息
    $map['status'] = ['egt', 0];
    $map['module'] = 'admin';
    $map['type'] = 1;
    if($group_id)
    {
        $map['id'] = $group_id;
    }
}
```

```

}
$auth_group = Db::name('AuthGroup')
->where($map)
->field('id,id,title,rules')
->find();
// 一级菜单权限节点列表
$map = [];
$map['status'] = 1;
$map['type'] = 2;
$auth_rules = Db::name('auth_rule')->field('id,name,type')->where
($map->column('id','name'));
// 二级菜单权限节点列表
$this->assign('first_rule_list',$auth_rules);
$map = [];
$map['status'] = 1;
$map['type'] = 1;
$auth_rules = Db::name('auth_rule')->field('id,name,type')->where
($map->column('id','name'));
$this->assign('second_rule_list',$auth_rules);
// 菜单节点列表
$this->assign('node_list',$this->returnMenuNodes());
// 当前角色分组权限节点列表
$this->assign('auth_group_nodes',explode(',',$auth_group
['rules']));
// 菜单高亮
$this->assign('active_url','AuthManager/index');
return $this->fetch();
}
}

```

首先调用了 updateRules()方法,实现菜单表(db\_menu)和权限节点表(db\_auth\_rules)数据的同步,代码如下:

```

/**
 * 通过菜单更新节点数据
 * @return bool
 */
public function updateRules(){
//需要新增的节点必然位于菜单节点
$nodes = $this->returnMenuNodes(false);
// 查询现在的节点规则
$map['module'] = 'admin';
$map['type'] = ['in','1,2'];
$rules = Db::name('auth_rule')->where($map)->order('name')->
select();
//保存需要插入和更新的新节点
$data = [];
foreach ($nodes as $value)
{
    $temp['name'] = $value['url'];
    $temp['title'] = $value['title'];
    $temp['module'] = 'admin';
    $temp['type'] = 2; // 顶级菜单
    if($value['pid'] > 0){
        $temp['type'] = 1; // 一般 URL
    }
}
}

```

```

    }
    $temp['status'] = 1;
    $data[strtolower($temp['name'].$temp['module'].$temp['type'])] =
    $temp; //去除重复项
}

$update = []; //保存需要更新的节点
$sids = []; //保存需要删除的节点的 id
foreach ($rules as $index=>$rule)
{
    $key = strtolower($rule['name'].$rule['module'].$rule['type']);
    if ( isset($data[$key]) ) {
        //如果数据库中的规则与配置的节点匹配,说明是需要更新的节点
        $data[$key]['id'] = $rule['id']; //为需要更新的节点补充 id 值
        $update[] = $data[$key];
        unset($data[$key]);
        unset($rules[$index]);
        unset($rule['condition']);
        $diff[$rule['id']]=$rule;
    }elseif($rule['status']==1){
        $sids[] = $rule['id'];
    }
}

// 是否有更新的节点数据
if ( count($update) ) {
    foreach ($update as $k=>$row){
        if ( $row != $diff[$row['id']] ) {

            Db::name('auth_rule')->where(['id'=>$row['id']])->update($row);
        }
    }
}

// 是否有需要删除的节点数据
if ( count($sids) ) {
    Db::name('auth_rule')->where( array( 'id'=>array('IN',implode(
    ',',$sids)) ) )->update(array('status'=>-1));
    //删除规则是否需要从每个用户组的访问授权表中移除该规则
}

// 是否有新增的节点数据
if( count($data) ){
    foreach($data as $value)
    {
        Db::name('auth_rule')->insert($value);
    }
}

return true;
}

```

回到 `groupmanage()` 方法, 根据 `type` 值的不同, 查询两组权限节点列表, 代码如下:

```

// 一级菜单权限节点列表
$map = [];
$map['status'] = 1;
$map['type'] = 2;

```

```

$auth_rules = Db::name('auth_rule')->field('id,name,type')->where
($map)->column('id','name');
// 二级菜单权限节点列表
$this->assign('first_rule_list',$auth_rules);
$map = [];
$map['status'] = 1;
$map['type'] = 1;
$auth_rules = Db::name('auth_rule')->field('id,name,type')->where
($map)->column('id','name');
$this->assign('second_rule_list',$auth_rules);

```

在页面渲染前，查询所有的菜单节点和当前角色分组所拥有的节点信息，随后变量置换到模板中去：

```

// 所有节点列表
$this->assign('node_list',$this->returnMenuNodes());
// 当前角色分组拥有的权限节点列表
$this->assign('auth_group_nodes',explode(',',$this->auth_group['rules']));

```

其中获取格式化的菜单列表 `returnMenuNodes()` 方法，定义如下：

```

/**
 * 获取菜单权限节点
 * @param bool $tree
 * @return array|false|mixed|\PDOStatement|string|\think\Collection
 */
protected function returnMenuNodes($tree = true){
    static $tree_nodes = array();
    if ( $tree && !empty($tree_nodes[(int)$tree]) ) {
        return $tree_nodes[$tree];
    }
    if((int)$tree){
        $list = Db::name('Menu')->field('id,pid,title,url,tip,hide')->order('sort asc')->select();
        foreach ($list as $key => $value) {
            if( stripos($value['url'],MODULE_NAME) !== 0 ){
                $list[$key]['url'] = MODULE_NAME.'/'.$value['url'];
            }
        }
        $nodes = list_to_tree($list,$pk='id',$pid='pid',$child='operator',$root=0);
        foreach ($nodes as $key => $value) {
            if(!empty($value['operator'])){
                $nodes[$key]['child'] = $value['operator'];
                unset($nodes[$key]['operator']);
            }
        }
    }else{
        $nodes = Db::name('Menu')->field('title,url,tip,pid')->order('sort asc')->select();
        foreach ($nodes as $key => $value) {
            if( stripos($value['url'],MODULE_NAME) !== 0 ){
                $nodes[$key]['url'] = MODULE_NAME.'/'.$value['url'];
            }
        }
    }
}

```

```

}
$tree_nodes[(int)$tree] = $nodes;
return $nodes;
}

```

合并菜单节点到权限节点表的流程如图 10-18 所示。

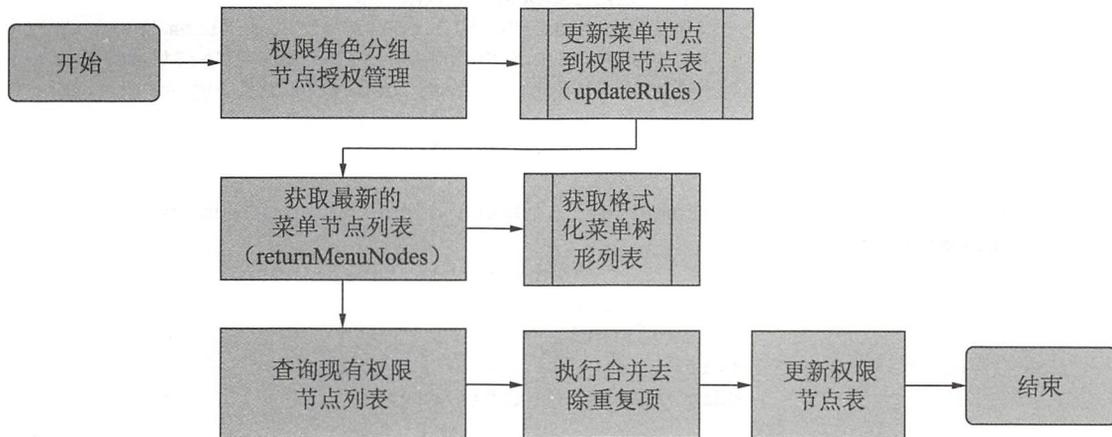


图 10-18 合并菜单节点到权限节点表

## (2) 新建权限角色分组管理模板

在 `auth_manager` 模板目录下，新增 `groupmanage.html` 模板文件，实现权限节点的展示和操作，核心代码如下：

```

{volist name="node_list" id="node"}
<div class="box">
  <div class="box-header with-border">
    <h3 class="box-title">{$node.title}</h3>
    <!-- 判断当前权限角色分组是否包含此节点 -->
    <input class="node-check" type="checkbox" value="{ $first_rule_list
[$node['url']]" name="rules[]"
      {if condition="in_array( $node['id'] , $auth_group_nodes )
      }checked="checked"}/>
  </div>
  <div class="box-body">
    <!-- 遍历二级菜单 (权限) 节点 -->
    {notempty name="node.child"}
    {volist name="node.child" id="child"}
    <div class="row">
      <div class="col-md-12">
        <span class="node-title">{$child.title}</span>
        <!-- 判断当前权限角色分组是否包含此节点 -->
        <input class="node-check" type="checkbox" value="
{$second_rule_list[$child['url']]" name="rules[]"
          {if condition="in_array( $child['id'] , $auth_
group_nodes )"}checked="checked"}/>

```



图 10-19 使用 ThinkPHP 自带的验证规则验证密码

此时可以使用 ThinkPHP 中的 Validate 验证类，自带的验证规则来进行验证，步骤如下。

### (1) 定义密码输入表单项

定义两个密码文本框，name 属性分别为 password 和 password\_confirm，其代码如下：

```
<div class="form-group">
  <label for="inputURL" class="col-sm-2 control-label">
    密码
  </label>
  <div class="col-sm-6">
    <input type="password" name="password" class="form-control"
      placeholder="请输入密码">
  </div>
</div>
<div class="form-group">
  <label for="inputURL" class="col-sm-2 control-label">
    确认
  </label>
  <div class="col-sm-6">
    <input type="password" name="password_confirm" class="form-
      control" placeholder="请再次输入密码">
  </div>
</div>
```

### (2) 在 Validate 类中定义验证规则

在 application\validate 目录下，找到 User 验证类文件，对两次密码验证的规则如下：

```
protected $rule = [
    'name' => 'require|max:100',
    'mobile' => 'unique:user|require|regex:\d{11}',
    'password' => 'require|confirm|regex:[a-zA-Z0-9]{6,18}'
];
```

可以看到对密码（password）的验证规则最多，现对字段的几个验证规则说明如下：

- **require**: 验证字段是否为空。
- **confirm**: 验证当前字段的值，是否和名为 password\_confirm 文本框的值相同。
- **regex:[a-zA-Z0-9]{6,18}**: 正则验证，字段的值的长度必须是 6~18 位，由数字和英文大小写组成。

## 2. 用户授权页

在权限角色分组列表上，增加用户授权的按钮“用户授权”，传入分组 id，代码如下：

```
<a href="{:url('user', ['id'=>$vo['id'])]}">用户授权</a>
```

随后在模板目录 `auth_manager` 中，新增 `user.html` 模板文件，因为只需要建立用户 `id` (`user_id`) 和权限分组 `id` (`group_id`) 的关系，所以表单只需要展示一个用户 `id` 的输入框即可。其核心代码如下：

```
<div class="form-group">
  <label for="inputUID" class="col-sm-2 control-label">
    用户 UID
  </label>
  <div class="col-sm-6">
    <input type="text" name="user_id" class="form-control"
id="inputUID" placeholder="用户 UID">
  </div>
</div>
```

### 3. 用户授权处理

修改 `application\admin\controller\AuthManager.php` 文件，新增 `user()` 方法，方法定义如下：

```
// 用户授权
public function user()
{
    if($this->request->isPost())
    {
        $user_id = input('user_id');           // 获取用户 id
        $group_id = input('group_id');        // 获取权限角色分组 id
        if( empty($user_id) ){
            $this->error('参数有误');
        }
        if(is_numeric($user_id)){
            if (is_admin($user_id) ) {       // 判断是否是超级管理员
                $this->error('该用户为超级管理员'); // 超级管理员不需要授权
            }
            // 查询是否存在当前用户
            if( !Db::name('user')->where(['id'=>$user_id])->find() ){
                $this->error('用户不存在');
            }
        }
        if($group_id)
        {
            // 查询是否存在当前分组
            $group_info = Db::name('auth_group')->where(['id'=>$group_id])->find();
            if(!$group_info)
            {
                $this->error('分组信息查询错误');
            }
            // 查询是否已经存在权限对应关系
            $data = [];
            $data['user_id'] = $user_id;
```

```

$data['group_id'] = $group_id;
$access_info = Db::name('auth_group_access')->where($data)->
find();
if($access_info)
{
    $this->error('请勿重复添加');
}
// 新建用户—角色权限对应关系
$insertId = Db::name('auth_group_access')->insert($data);
if($insertId !== false)
{
    $this->success('添加成功');
}
}
}
$this->assign('active_url' , 'AuthManager/index');
return $this->fetch();
}

```

在新增用户权限角色关系前，需要对是否是超级管理员、是否存在分组用户等进行判断，最后再判断是否已经添加过关系，避免添加重复的数据。

实现以上内容后，给某个分组添加用户权限的操作，如图 10-20 所示。

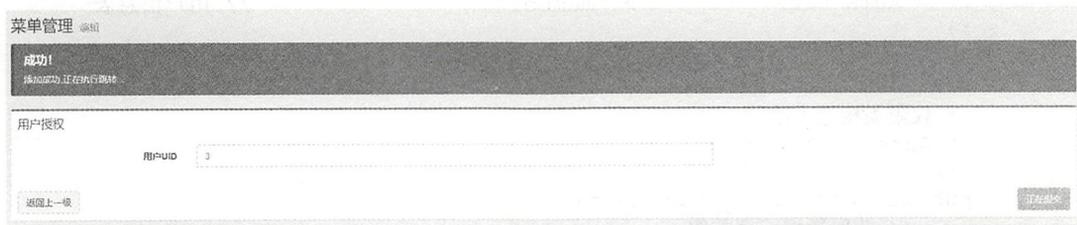


图 10-20 给用户增加角色分组权限

## 10.4 权限管理——权限验证

完成权限节点、权限角色和用户授权的功能开发后，最后来看权限验证。用户在登录后，Admin 类的初始化方法，会对用户访问的控制器方法进行权限验证，通过验证可以直接使用，否则提示权限不足。

### 10.4.1 权限处理类

先定义权限验证类，基于 ThinkPHP 5 本身的三方类库扩展架构，在 extend 目录下，新增 Auth 目录，随后新增 Auth.php 类文件。Auth 类主要提供用户权限验证的方法，类的实现步骤如下。

## 1. 类属性与构造方法

修改 Auth.php 类，增加属性和构造方法相关代码，定义需要操作的表名称和是否开启权限认证。其代码如下：

```
<?php
namespace Auth;
use think\Db;
/**
 * RABC 权限操作类
 * Class Auth
 * @package Auth
 */
class Auth
{
    //默认配置
    protected $_config = array(
        'AUTH_ON'          => true,                // 是否开启认证开关
        'AUTH_GROUP'       => 'auth_group',        // 权限角色数据表名
        'AUTH_GROUP_ACCESS' => 'auth_group_access', // 用户-用户组关系表
        'AUTH_RULE'        => 'auth_rule',        // 权限规则表
        'AUTH_USER'        => 'member'            // 用户信息表
    );

    /**
     * 权限类构造方法
     * Auth constructor.
     */
    public function __construct() {
        // 获取数据库表前缀
        $prefix = config('database.prefix');
        // 定义权限表名称
        $this->_config['AUTH_GROUP'] = $prefix.$this->_config['AUTH_GROUP'];
        $this->_config['AUTH_RULE'] = $prefix.$this->_config['AUTH_RULE'];
        $this->_config['AUTH_USER'] = $prefix.$this->_config['AUTH_USER'];
        $this->_config['AUTH_GROUP_ACCESS'] = $prefix.$this->_config['AUTH_GROUP_ACCESS'];
    }
}
```

## 2. 定义用户权限节点检查方法

继续定义 Auth 权限类的 check()方法，传入需要验证和权限节点名称和用户 id，核心代码如下：

```
/**
 * @param $name 权限节点名称：模块/控制器/方法 admin/index/index
 * @param $uid 用户 ID
```

```

* @return bool
*/
public function check($name, $uid) {
    // 是否开启权限验证
    if (!$this->_config['AUTH_ON'])
    {
        return true;
    }
    // 获取当前用户拥有的权限节点列表
    $authList = $this->getAuthList($uid); //获取用户需要验证的所有有效规则列表
    // 判断权限节点是否为字符串
    if (is_string($name)) {
        $name = [strtolower($name)];
    }
    // 保存验证通过的规则名
    $list = [];
    // 遍历用户权限列表
    foreach ( $authList as $auth ) {
        if (in_array($auth , $name)){
            $list[] = $auth ;
        }
    }
    // 判断是否拥有权限
    if (!empty($list)) {
        return true;
    }
    return false;
}

```

方法在定义时，先判断是否开启权限判定，否则一律返回 true，随后根据用户 id，查询用户所拥有的所有的权限节点列表，方法 `getAuthList()` 的定义如下：

```

protected function getAuthList($uid) {
    // 保存用户验证通过的权限列表
    static $_authList = [];
    // 查看是否已经有用户权限列表
    if (isset($_authList[$uid])) {
        return $_authList[$uid];
    }
    //读取用户所属用户组
    $groups = $this->getGroups($uid);
    // 保存用户所属用户组设置的所有权限规则 id
    $sids = [];
    foreach ($groups as $group) {
        $sids = array_merge($sids, explode(',', trim($group['rules'], ',')));
    }
    // 移除重复的权限节点值
    $sids = array_unique($sids);
    // 没有权限节点值，则返回空数组
    if (empty($sids)) {
        $_authList[$uid] = [];
        return [];
    }
}

```

```

    }
    // 构造查询条件
    $map=array(
        'id'=>['in',$ids],
        'status'=>1,
    );
    //读取用户组所有权限规则
    $rules = Db::table($this->_config['AUTH_RULE'])->where($map)->
    field('condition,name')->select();
    //循环规则, 判断结果
    $authList = []; //
    foreach ($rules as $rule) {
        //只要存在就记录
        $authList[] = strtolower($rule['name']);
    }
    $_authList[$uid] = $authList;
    return array_unique($authList);
}

```

获取用户所在的权限角色列表的 `getGroups()`方法, 其定义如下:

```

/**
 * 获取用户角色及其节点
 * @param $user_id
 * @return mixed
 */
public function getGroups($user_id) {
    static $groups = [];
    if (isset($groups[$user_id]))
        return $groups[$user_id];
    $where_map = [
        'a.user_id' => $user_id,
        'g.status' => 1
    ];
    $user_groups = Db::table($this->_config['AUTH_GROUP_ACCESS'].' a')
        ->where($where_map)
        ->join($this->_config['AUTH_GROUP']." g" , 'a.group_id=g.id')
        ->select();
    $groups[$user_id]=$user_groups?:[];
    return $groups[$user_id];
}

```

该方法进行了用户表 (`db_user`) 和用户权限角色关联表 (`db_auth_group_access`) 的关联查询, 转换成 SQL 语句如下:

```

SELECT * FROM
`db_auth_group_access` `a`
INNER JOIN `db_auth_group` `g`
ON
`a`.`group_id`=`g`.`id`
WHERE
`a`.`user_id` = 2
AND
`g`.`status` = 1

```

继续查看 `check()` 权限验证方法，根据用户 `id` 获取权限列表后，为了方便对比，把请求信息都转换为小写：

```
// 获取请求对象（URL 地址和参数都转换为小写）
if ($mode=='url') {
    $REQUEST = unserialize( strtolower(serialize($_REQUEST)) );
}
```

为了不循环遍历所有的请求参数，这里先使用 `serialize()` 方法处理请求数据，统一转换为小写后，再使用 `unserialize()` 方法转换回数组格式。

最后遍历用户权限列表，判断用户是否有当前权限，代码如下：

```
// 遍历用户权限列表
foreach ( $authList as $auth ) {
    if (in_array($auth, $name)){
        $list[] = $auth ;
    }
}
// 判断是否拥有权限
if (!empty($list)) {
    return true;
}
return false;
```

完成以上步骤，可以编写以下方法，测试 `Auth` 类是否可以正常使用，代码如下：

```
/**
 * 测试权限验证类
 */
public function auth_test()
{
    $auth_rules = 'Admin/Index/index'; // 权限节点
    $user_id = 3; // 用户 ID
    $auth = new Auth(); // 实例化权限类
    if($auth->check($auth_rules, $user_id) // 验证是否拥有权限
    {
        echo "用户 id:{$user_id},拥有{$auth_rules}的权限";die;
    }
    echo "用户 id:{$user_id},没有{$auth_rules}的权限";die;
}
```

只需要传入用户 `id` 和权限节点，就可以验证是否有权限，执行方法的返回值如下：

用户 id:3,拥有 Admin/Index/index 的权限

## 10.4.2 实现权限验证

完成权限验证类后，需要在内容管理框架中，增加对应的验证代码。对用户是否拥有某节点权限的验证，可以放在 `Admin` 基类中实现，后台用户权限验证流程图如图 10-21 所示。

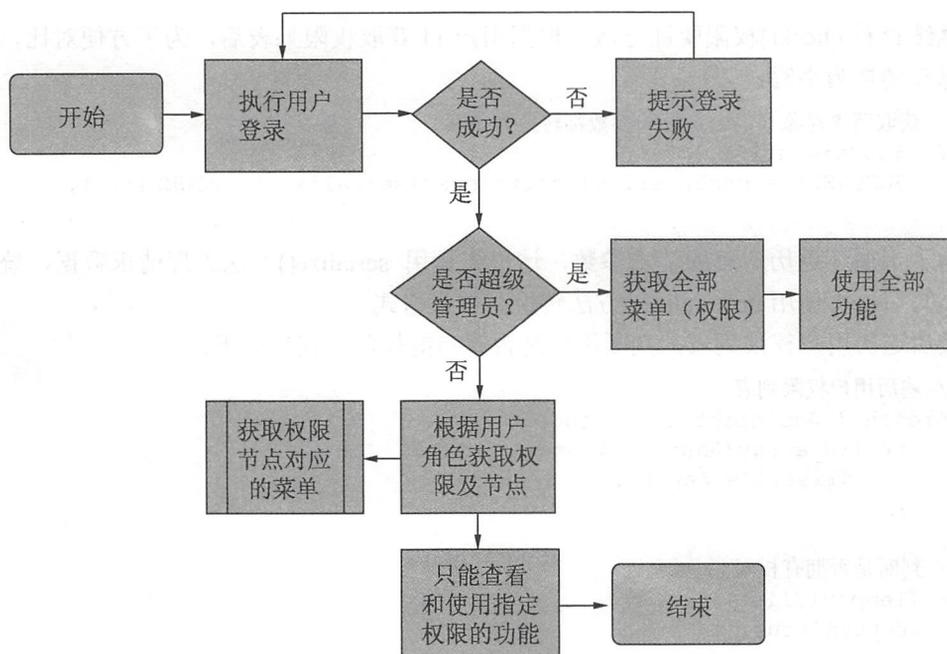


图 10-21 后台用户权限验证流程图

实际上需要做的主要有两部分：

- 在控制器入口，检测用户是否拥有菜单权限。
- 根据用户的权限，判断要返回的菜单列表。

实现步骤如下。

### 1. 用户权限验证

使用 Auth 权限验证类，对已登录用户进行权限的验证。修改 application\admin\controller\admin.php 控制器基类，在初始化\_initialize()方法中，增加以下内容：

```

// 定义常量获取模块、控制器和方法名
define('MODULE_NAME' , request()->module());
define('CONTROLLER_NAME' , request()->controller());
define('ACTION_NAME' , request()->action());
// 当前项目地址
define('__APP__' , strip_tags(rtrim($_SERVER['SCRIPT_NAME'],'/')));
// 当前请求是否是 AJAX
define('IS_AJAX' , request()->isAjax());
// 是否是超级管理员
define('IS_ROOT' , is_admin());

// 菜单变量置换到模板中
$this->assign('menu_list' , $this->getMenus());
// 当前选中的控制器/方法

```

```

$this->assign('active_url' , CONTROLLER_NAME.'/'.ACTION_NAME);

// 权限判断(超级管理员默认拥有全部权限)
if(!IS_ROOT)
{
    // 获取当前访问的控制器/方法地址
    $rule = strtolower(MODULE_NAME.'/'.CONTROLLER_NAME.'/'.ACTION_NAME);
    // 初始化权限类
    $auth = new Auth();
    if(!$auth->check($rule,USER_ID))
    {
        unset($_SESSION);
        $this->error('未授权访问');
    }
}

```

在上述代码中,先使用系统内置的 request()助手方法,动态地获取当前的模块/控制器/方法,然后结合用户登录信息,调用 Auth 权限类进行权限的验证。

## 2. 根据用户权限返回对应的菜单列表

除了未授权的节点用户不能访问外,在展示上也需要隐藏。所以需要修改 Admin 控制器的 getMenu()方法,在查询一、二级菜单和构建数据的时候,判断用户是否拥有对应的权限,判断一级菜单权限的修改后的代码如下:

```

/**
 * 获取控制器菜单数组,二级菜单元素位于一级菜单的'_child'元素中
 */
public function getMenu($controller=CONTROLLER_NAME){
    $menus = session('ADMIN_MENU_LIST.'.$controller);
    if(empty($menus)){
        // 获取主菜单
        $where['pid'] = 0;
        $where['hide'] = 0;
        $menus['main'] = Db::name('menu')->where($where)->order('sort asc')->field('id,title,url')->select();
        $menus['child'] = array(); //设置子节点
        $auth = new Auth();
        foreach ($menus['main'] as $key => $item) {
            // 判断主菜单权限
            if ( !IS_ROOT && !$auth->check(strtolower(MODULE_NAME.'/'.
                $item['url']),USER_ID) ) {
                unset($menus['main'][$key]);
                continue;//继续循环
            }
            if(strtolower(CONTROLLER_NAME.'/'.ACTION_NAME) == strtolower
                ($item['url'])){
                $menus['main'][$key]['class']='active';
            }
        }
    }
}

```

```
.....
```

```
}
```

在遍历子菜单的代码中，增加检测菜单权限的代码：

```
foreach ($menus['main'] as $key => $item) {
    // 获取当前主菜单的子菜单项
    if($item['id'] == $nav['id']){
        $menus['main'][$key]['class']='active';
        .....
        if(!IS_ROOT){
            // 检测菜单权限
            $to_check_urls = array();
            if($second_urls)
            {
                foreach ($second_urls as $key=>$to_check_url) {
                    if( stripos($to_check_url,MODULE_NAME)!=0 ){
                        $rule = MODULE_NAME.'/'.$to_check_url;
                    }else{
                        $rule = $to_check_url;
                    }
                    if($auth->check($rule, USER_ID))
                    {
                        $to_check_urls[] = $to_check_url;
                    }
                }
            }
        }
    }
}
.....
}
```

完成以上步骤，一个最基本的 RBAC 权限模块就已经完成。

### 3. 使用权限模块

为了验证已经开发完成的权限系统是否可以正常地工作，测试步骤如下。

(1) 新增用户。新增名为 authtest 的测试用户，信息如图 10-22 所示。

标题	authtest
手机号	13511112229

图 10-22 新增测试用户

(2) 新增权限角色分组。新增名为“权限测试”的角色分组，并只给与首页和用户列表相应的权限节点，如图 10-23 所示。

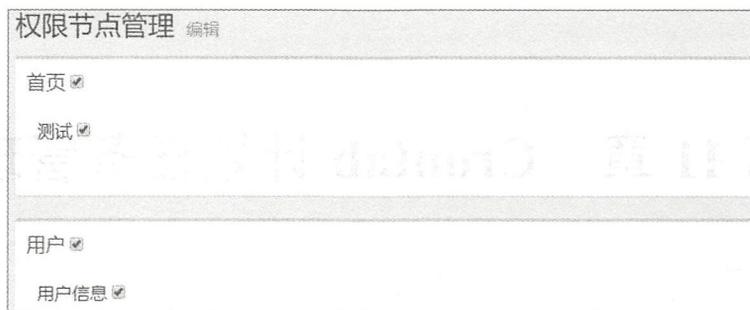


图 10-23 增加角色分组授权节点信息

(3) 角色分组用户授权。授权过程，如图 10-24 所示。

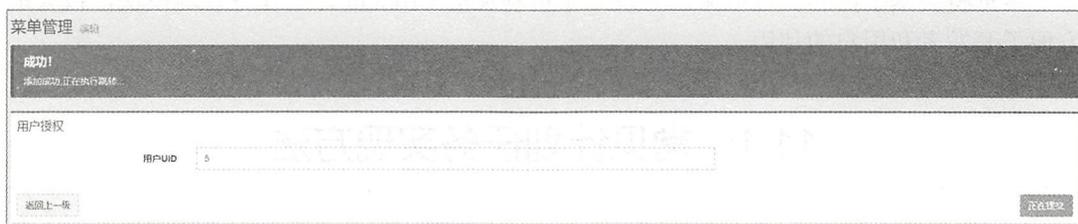


图 10-24 用户授权操作

(4) 用户登录查看权限。使用新注册的用户登录管理后台，效果如图 10-25 所示。

因为当前用户并没有授予菜单相关的访问权限，所以访问菜单列表地址：`wangcmf.com/admin/menu/index.html`，会提示错误，如图 10-26 所示。

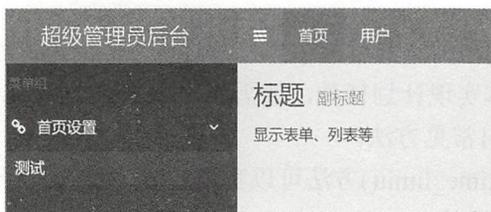


图 10-25 显示部分权限菜单

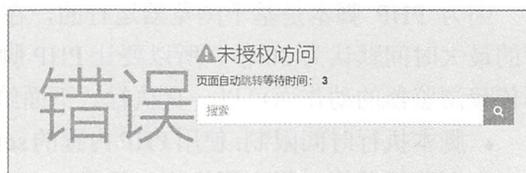


图 10-26 权限验证失败提示

# 第 11 章 Crontab 计划任务管理

Crontab 是 Linux/UNIX 系统上最常用的计划任务管理工具。PHP 开发者在实际项目中会经常使用到该工具。不过受限于实际的项目部署应用环境，Crontab 配置信息的编辑多由专门的服务器维护人员操作，项目使用者想要自行添加和编辑，流程较为烦琐。

本章将会基于内容管理框架，开发一个可视化的 Crontab 计划任务管理模块，极大地方便了开发者和用户的使用。

## 11.1 常见计划任务实现方法

在实际开发中，经常会遇见有计划任务需求的场景，如创建但未支付的订单，或失败需要退款的商品等，这些都需要定时执行某些操作。类似的场景还有许多。本节主要讲解常见的几种计划任务的实现方法。

### 11.1.1 PHP 脚本实现计划任务

因为 PHP 脚本是基于浏览器运行的，在关闭浏览器后程序会自动终止，同时程序运行的最大时间默认为 30 秒，所以要让 PHP 脚本实现计划任务，就需要一个或多个脚本，不依赖浏览器的动作而可以一直执行。下面给出常见方法。

- 脚本执行时间限制：使用 PHP 内置的 `set_time_limit()` 方法可以实现让脚本持续执行。
- 浏览器关闭，程序不终止：使用 PHP 内置的 `ignore_user_abort()` 方法，可以让脚本在关闭浏览器的情况下可以继续执行。
- 脚本定时执行：为了不让 PHP 提前结束进程，使用语言内置的无限循环结构和 `sleep()` 等待方法可以让脚本定时执行，如 5 秒执行一次。

这里给出一个简单的脚本实例，演示 PHP 计划任务的实现过程。新建 `task.php` 脚本文件，实现代码如下：

```
<?php
ignore_user_abort(); // 关闭浏览器，PHP 脚本也可以继续执行
set_time_limit(0); // 通过 set_time_limit(0) 可以让程序无限制地执行下去
$interval = 3; // 每隔 3 秒运行一次
$number = 0; // 计数器
```

```

//无限循环
do{
    if($number < 10)    // 执行 10 次文件写入操作
    {
        // 定时写入文件
        file_put_contents(__DIR__ . '/log/log_' . date('Ymd_His') . '.log' ,
            time() );
        $number = $number + 1;
    }
    sleep($interval); // 等待 3 秒
}while(true);

```

在上述代码中，实现了循环内的代码每 3 秒执行一次，一共执行 10 次的计划任务（新建 log 文件）。注意脚本头文件中定义的两个系统方法，代码如下：

```

ignore_user_abort(); // 关掉浏览器，PHP 脚本也可以继续执行
set_time_limit(0); // 通过 set_time_limit(0) 可以让程序无限制地执行下去

```

在浏览器中访问 task.php 脚本，可以发现在 log 目录下，会自动新增 10 个 log 文件，如图 11-1 所示。

log_20180103_020913.log	2018/1/3 星期三 10:09
log_20180103_020916.log	2018/1/3 星期三 10:09
log_20180103_020919.log	2018/1/3 星期三 10:09
log_20180103_020922.log	2018/1/3 星期三 10:09
log_20180103_020925.log	2018/1/3 星期三 10:09
log_20180103_020928.log	2018/1/3 星期三 10:09
log_20180103_020931.log	2018/1/3 星期三 10:09
log_20180103_020934.log	2018/1/3 星期三 10:09
log_20180103_020937.log	2018/1/3 星期三 10:09
log_20180103_020940.log	2018/1/3 星期三 10:09

图 11-1 使用 PHP 脚本实现定时写入文件

不过使用这种方式，若更新了 PHP 文件内容，需要重启 PHP 进程才可以使计划任务生效。此外，若出现程序错误或者内存溢出，PHP 当前运行的进程也会被终止，故使用这种方式具有极大的不稳定性，维护起来并不方便。

## 11.1.2 使用系统级别的计划任务工具

既然使用 PHP 脚本实现计划任务，在一些方面不是很理想，那么借助外部的工具就是更好的选择。无论是 Windows 还是 Linux 操作系统，都有完善的计划任务解决方案。例如在 Windows 10 中，在控制面板中，就可以直接找到任务计划程序并管理计划任务，使用起来较为简单，如图 11-2 所示。

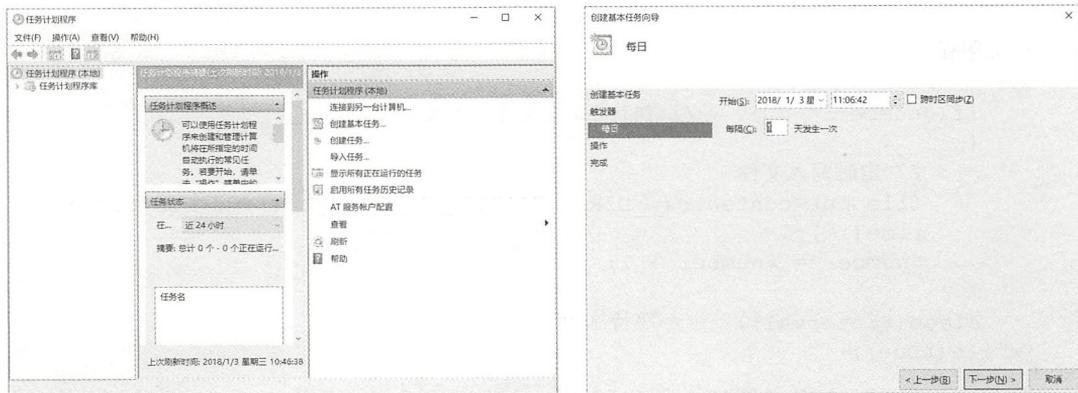


图 11-2 在 Windows 系统中管理计划任务

不过基于 PHP 的项目，一般部署在 Windows 服务器上的较少，故 PHP 开发者最常用的还是 Linux 系统中的 Crontab 计划任务命令工具。为了方便演示，这里使用 vagrant 工具，在本地搭建 Ubuntu 的虚拟机系统，登录成功后，执行以下命令查看 Crontab 的帮助信息。

```
crontab -h
```

结果如下：

```
vagrant@scotchbox:~$ crontab -h
crontab: invalid option -- 'h'
usage: crontab [-u user] file
       crontab [ -u user ] [ -i ] { -e | -l | -r }
              (default operation is replace, per 1003.2)
       -e      (edit user's crontab)
       -l      (list user's crontab)
       -r      (delete user's crontab)
       -i      (prompt before deleting user's crontab)
```

也可以执行以下命令，查看管理员用户（root）有哪些计划任务。

```
crontab -l
```

执行结果如下：

```
vagrant@scotchbox:~$ sudo crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
# * * * * * /usr/bin/php7.0 /var/www/wangcmf/public/cron.php
* * * * * /usr/bin/php7.0 /var/www/wangcmf/public/index.php crontab/
Crontask/index
```

其实无论使用哪个计划任务工具，都是为了实现在特定的时间执行某项任务这样的功

能，开发者可以根据实际的需求选择不同的工具。

## 11.2 Crontab 入门

Linux 上的 Crontab 计划任务工具，没有可视化的交互界面，配置与使用该工具都需要在命令行下操作，本节主要讲解 Crontab 工具的一些入门操作。

### 11.2.1 Crontab 使用教程

在讲解之前，需要先了解 Cron 和 Crontab 的区别。经常使用到的 Crontab 命令，是 Cron 和 Table 合体的简写，它其实是 Cron 的配置文件，也可以叫做作业列表，开发者可以在里面定义不同的时间表达式，实现不同的任务需求。而 Cron 是 Linux 内置的系统进程，Cron 搭配 Shell 脚本，就可以执行特定的计划任务。

以 ubuntu16.04 为例，Crontab 在系统中多个地方都有配置文件。

- /var/spool/cron/crontabs: 该目录下存放的是每个用户的 Crontab 任务列表，配置文件的名称与用户名一致。
- /etc/cron.d/: 存放需要执行的 Crontab 文件或脚本。
- /etc/cron.hourly、/etc/cron.daily、/etc/cron.weekly、/etc/cron.monthly: 这些目录中的脚本，可以以每小时、每天、每星期和每月为单位执行一次。

常用的命令说明如下：

```
crontab [-u username] //省略用户表表示操作当前用户的 Crontab
-e      (编辑任务列表)
-l      (列出任务列表里的命令)
-r      (删除任务列表)
```

Crontab 的命令构成为：时间表达式+操作符+操作命令。时间表达式的构成有分、时、日、月、周 5 种，操作符有以下几种：

- \* : 取值范围内的所有数字。
- / : 每过多少个数字。
- : 从 x 到 z。
- , : 散列数字。

例如，每分钟执行一次命令 showtime，示例如下：

```
* * * * * showTime
```

每天晚上 11 点 30 执行某个 PHP 脚本文件，示例如下：

```
30 23 * * * /usr/bin/php7.0 /var/www/task.php
```

时间表达式的定义非常灵活，不过若不经常使用，极易导致配置错误，这里推荐使用在线的 Crontab 表达式生成器，可以根据需求进行定制。访问地址为：<http://www.pppet.net/>

访问后使用效果如图 11-3 所示。

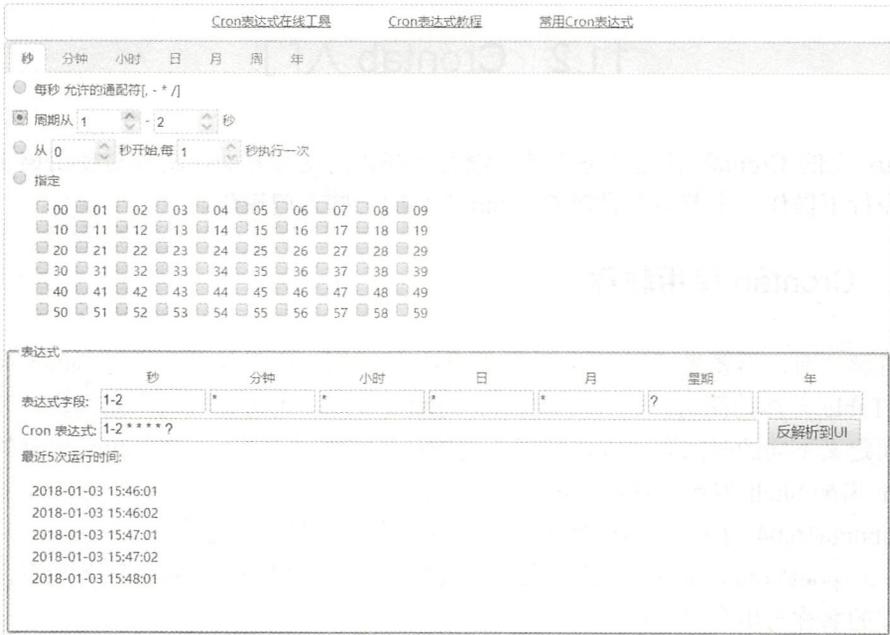


图 11-3 在线的 Crontab 表达式生成器

## 11.2.2 用 Crontab 实现 PHP 文件定时写入

为了方便演示，这里实现一个每分钟写入文件日期的功能。首先在系统的/var/www (Ubuntu 虚拟机环境) 目录下，新增 log.php 脚本文件。具体代码如下：

```
<?php
file_put_contents(__DIR__.'/run.log', date('Y-m-d H:i:s')."\\r\\n", FILE_
APPEND);
?>
```

此脚本每次执行，都会生成一个带有当前时间信息的文件，以便区分文件是何时创建。随后执行以下命令，给当前用户增加 Crontab 表达式，指定何时执行 log.php 脚本：

```
crontab -e
```

增加以下内容：

```
* * * * * /usr/bin/php7.0 /var/www/log.php
```

保存后等待几分钟，发现在/var/www 目录下，自动生成了 run.log 文件，使用 cat 命令查看，内容如下：

```
root@scotchbox:/var/www# cat run.log
2018-01-03 16:01:01
```

```
2018-01-03 16:02:01
2018-01-03 16:03:02
```

根据 Crontab 中配置的时间任务表达式，log.php 脚本每隔一分钟都会被执行一次，而此时修改脚本文件，则不需要重启 PHP 进程，修改的代码如下：

```
<?php
file_put_contents(__DIR__.'run.log','time:'. date('Y-m-d H:i:s').
"\r\n", FILE_APPEND);
?>
```

经过几分钟的等待，再次查看 run.log 文件，发现已经写入了新的格式内容：

```
2018-01-03 16:01:01
2018-01-03 16:02:01
2018-01-03 16:03:02
2018-01-03 16:04:01
time:2018-01-03 16:05:01
```

通过对日志列表细心观察可以发现，Crontab 在执行的时候，默认最小单位为“分”，并且会从第 1 秒（根据配置文件编辑的时间，误差在 2 秒内）开始执行。开发者还可以通过使用 sleep 关键字定义在哪一秒执行，修改时间表达式如下：

```
* * * * * sleep 20; /usr/bin/php7.0 /var/www/log.php
```

sleep 后面的参数 20 指的就是延迟多少秒执行，等待几分钟后，再次查看 run.log 文件，发现文件写入的时间发生了变化：

```
root@scotchbox:/var/www# cat run.log
2018-01-03 16:02:01
2018-01-03 16:03:02
2018-01-03 16:04:01
time:2018-01-03 16:05:01
time:2018-01-03 16:06:01
time:2018-01-03 16:07:01
time:2018-01-03 16:08:01
time:2018-01-03 16:10:21
time:2018-01-03 16:11:21
```

## 11.3 实现计划任务管理模块

在了解 Crontab 的基础使用规则后，本节尝试在 PHP 中对 Crontab 计划任务进行管理，最终实现灵活配置，提高开发效率。

### 11.3.1 程序流程与数据结构设计

使用 Linux 系统的 Crontab 计划任务，需要登录到 Shell 命令行中，才可以新增和编写计划任务表达式。在实际应用中，并不是所有的项目都有此权限，所以对于需要频繁使用计划任务的应用来说，一个可视化的计划任务管理模块就非常必要了。

开发计划任务管理模块，主要要包含两个功能部分。

- 常驻任务处理脚本：此脚本的作用就是每隔一段时间（如 1 分钟），查看数据库中是否有新增、变更的任务记录，满足条件则执行操作。脚本可以处理的任务类型分为 URL 和 Shell 两种，两种类型执行的动作请求一个 URL 地址或执行 Shell 命令。
- 后台任务管理：负责管理需要执行的计划任务，展示计划任务执行日志。

其中，常驻任务处理脚本的处理流程如图 11-4 所示。

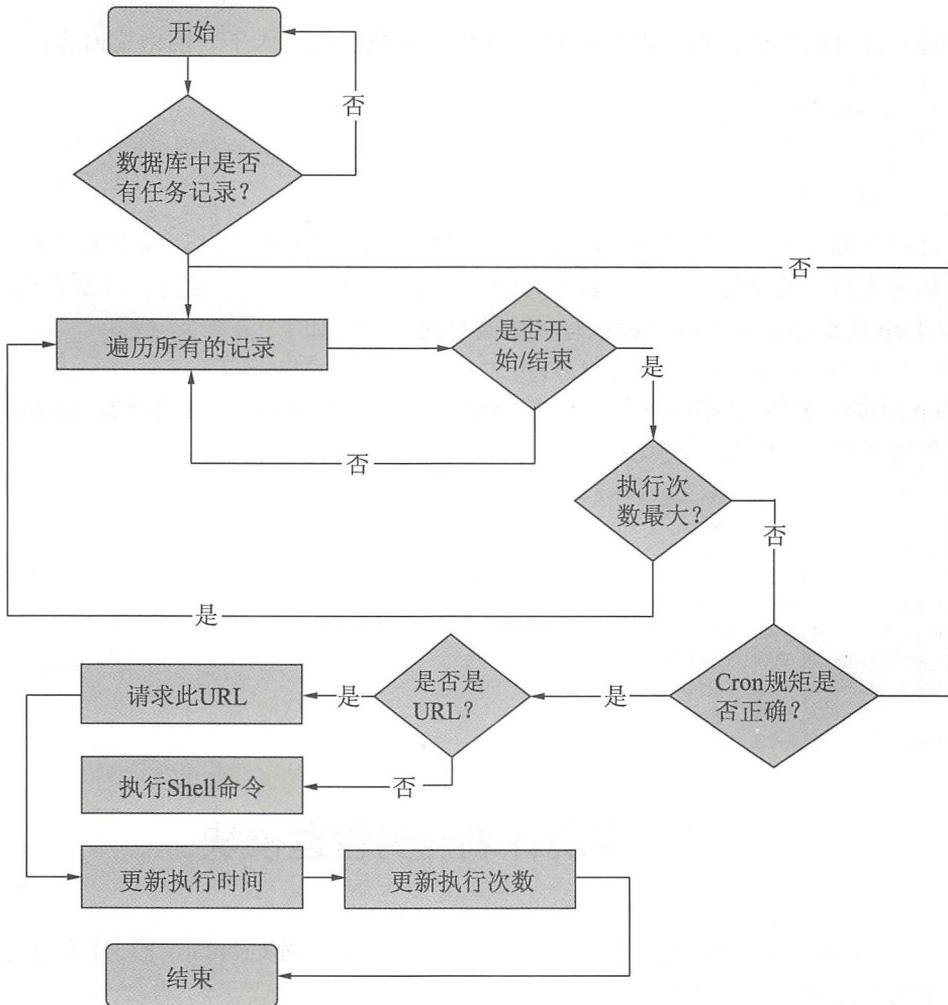


图 11-4 定时执行的脚本处理流程

其中对于 Crontab 时间表达式的解析，可以借助开源的力量，使用 Composer 中托管的 mtdowling/cron-expression 类库，帮助开发者计算下一个或之前的运行日期，并确定一个 Crontab 表达式是否到期，这样可以提高计划任务的精确度。

在数据库表结构的设计上，主要有计划任务表（db\_crontab）和计划任务执行日志

(db\_crontab\_log) 表，分别存放用户增加的计划任务和任务执行日志。计划任务表主要记录任务的名称、Crontab 表达式、开始和结束时间、规定执行的次数等，其字段说明如表 11-1 所示。

表 11-1 计划任务表字段说明

列 (字段)	类 型	注 释
id	int(10) unsigned 自动增量	ID
type	varchar(10)	类型
title	varchar(150)	标题
content	text	内容
schedule	varchar(100)	Cron 时间表达式
sleep	tinyint(1) unsigned [0]	延迟秒数执行
maximums	int(10) unsigned [0]	最大执行次数 0 为不限
executes	int(10) unsigned [0]	已经执行的次数
create_time	int(10) unsigned [0]	创建时间
update_time	int(10) unsigned [0]	更新时间
begin_time	int(10) unsigned [0]	开始时间
end_time	int(10) unsigned [0]	结束时间
execute_time	int(10) unsigned NULL	最后执行时间
weigh	int(10) [0]	权重
status	tinyint(1) [1]	-1: 删除; 0: 禁用; 1: 正常; 2: 已经完成; 3: 已经过期

计划任务执行日志表与计划任务表关联，需要记录 crontab\_id 字段，记录哪些任务都执行了什么操作，有哪些返回值。其字段说明如表 11-2 所示。

表 11-2 计划任务执行日志表结构字段说明

列 (字段)	类 型	注 释
id	int(10) unsigned 自动增量	主键自增长
type	varchar(10)	类型
crontab_id	int(10) unsigned	任务的ID
title	varchar(150)	标题
remark	mediumtext NULL	备注
create_time	int(10) unsigned	执行时间
update_time	int(10) unsigned	更新时间
status	tinyint(1)	状态 0:失败 1:成功

### 11.3.2 计划任务管理

为了方便开发，使用第 9 章、第 10 章开发的内容管理框架作为底层的支持。这里把

计划任务管理作为内容框架的一个模块功能，继续在 wangcmf 项目上进行开发。

## 1. 新建菜单

计划任务管理模块主要包含任务列表、任务执行日志列表、变更记录状态和新增任务等几个模块，在菜单模块中，增加部分菜单项的结果如图 11-5 所示。

ID	名称	上级菜单	分组	URL地址	状态	操作
133	设置记录状态	--	--	Cron/setCronStatus	正常	编辑 禁用 删除
1	首页	--	--	Index/index	正常	编辑 禁用 删除
16	用户	--	--	User/index	正常	编辑 禁用 删除
68	系统	--	--	Config/index	正常	编辑 禁用 删除

新增

图 11-5 计划任务增加的部分菜单项

## 2. 计划任务列表

在 application\admin\controller 目录下，新增 Cron.php 控制器文件，新增 index()方法，查询数据表中的记录，部分代码如下：

```
class Cron extends Admin
{
    /**
     * 计划任务列表
     * @return mixed
     */
    public function index()
    {
        // 查询所有的任务记录
        $list = Db::name('crontab')->order('id desc')->paginate(10);
        $this->assign('_list', $list);
        return $this->fetch();
    }
}
```

随后在 application\admin\view 模板目录下，新增 cron 目录，在其中新建 index.html 模板文件，目的是为了展示所有任务记录详细信息，核心代码如下：

```
{volist name="_list" id="vo"}
<tr>
    <td>{$vo.id}</td>
    <td>{$vo.title} </td>
    <td>{$vo.type}</td>
```

```

<td>{$vo.content}</td>
<td>{$vo.schedule}</td>
<td>
    <p>最大执行次数: <span style="font-weight:bold;color:red;">{$vo.
    maximums}</span></p>
    <p>已经执行次数: <span style="font-weight:bold;color:blue;">{$vo.
    executes}</span></p>
</td>
<td>
    <p>创建: {$vo.create_time|time_format}</p>
    <p>更新: {$vo.update_time|time_format}</p>
    <p>开始: {$vo.begin_time|time_format}</p>
    <p>结束: {$vo.end_time|time_format}</p>
    <p>最后执行: {$vo.execute_time|time_format}</p>
</td>
<td>{$vo.status|get_cron_status_info}</td>
<td>
    <a href="{:url('log_list',['id'=>$vo['id']])}">日志列表</a>
    {if condition=' $vo["status"] eq 0'}
    <a class="ajax-get" href="javascript:;" url="{:url('
    setCronStatus',['id'=>$vo['id'],'status'=>1])}">启用</a>
    {else/}
    <a class="ajax-get" href="javascript:;" url="{:url('
    setCronStatus',['id'=>$vo['id'],'status'=>0])}">禁用</a>
    {/if}
    <a class="ajax-get confirm" href="javascript:;" url="{:url('
    setCronStatus',['id'=>$vo['id'],'status'=>-1])}">删除</a>
</td>
</tr>
{/volist}

```

在浏览器中访问列表页面，在有数据的情况下，效果如图 11-6 所示。

ID	任务名称	任务类 型	任务详情	Cron表 达式	次数信息	时间信息	状态	操作
2	订单自动收 货	shell	/usr/bin/php7.0 /var/www/wangcmf/public/index.php crontab/order/takeok	.....	最大执行次 数: 10 已经执行次 数: 0	创建: 2017-12-30 09:27 更新: -- 开始: 2017-12-30 11:11 结束: 2017-12-31 11:11 最后执行: --	正常	日志列表 禁用 删除

图 11-6 计划任务管理列表页

### 3. 新增计划任务

在 application\admin\view\cron 模板目录下，新增 add.html 模板文件，展示任务增加的表单。其中，提供任务名称、任务类型、Cron（时间）表达式、最大执行次数、开始时间和结束时间等需填写选项。核心代码可以参考书籍源代码，这里不再逐一贴出。完成后的效果如图 11-7 所示。

图 11-7 任务提交表单页

为了方便使用，页面上还增加了填写以上选项的注意事项，供用户参考使用，如图 11-8 所示。

**注意事项**

1. 系统须支持shell\_exec()方法（不支持Windows系统）。
2. 使用前需要开启一个计划任务，需要每分钟调用一次任务脚本。格式如下：`***** /PHP所在路径/php /项目所在位置/public/index.php crontab/CronTask/index`。
3. 执行周期必须严格满足当前系统中Crontab的表达式。
4. 最大执行次数不填写，默认为一直执行。
5. 选择URL类型，格式如：`http://xxxxx/index/xxx/xxx`。
6. 选择Shell类型，需要输入完整的shell命令。

图 11-8 新增计划任务的注意事项

随后在 Cron 控制器中，新增 add()方法，核心代码如下：

```
/**
 * 新增计划任务
 * @return mixed
 */
public function add()
{
    if($this->request->isPost())
    {
        // 获取参数
        $data = $this->request->param();
        // 参数验证
        $validate = new Crontab();
```

```

$result = $validate->check($data);
if(!$result)
{
    $this->error($validate->getError());
}
// 时间参数格式转换
$data['begin_time'] = strtotime($data['begin_time']);
$data['end_time'] = strtotime($data['end_time']);
// 时间区间范围判断
if($data['begin_time'] < time())
{
    $this->error('开始时间不能小于当前时间');
}
if($data['begin_time'] >= $data['end_time'])
{
    $this->error('开始时间不能大于等于结束时间');
}
// 构建数据并写入数据表
$data['status'] = 1;
$data['create_time'] = time();
if(!Db::name('crontab')->insert($data))
{
    $this->error('数据写入失败!');
}
$this->success('新增计划任务成功!');
}
$this->assign('active_url' , 'Cron/index');
return $this->fetch();
}

```

除了使用 `Validate` 验证类进行常规验证外，对任务的开始时间和结束时间也进行了进一步的验证，代码如下：

```

// 时间参数格式转换
$data['begin_time'] = strtotime($data['begin_time']);
$data['end_time'] = strtotime($data['end_time']);
// 时间区间范围判断
if($data['begin_time'] < time())
{
    $this->error('开始时间不能小于当前时间');
}
if($data['begin_time'] >= $data['end_time'])
{
    $this->error('开始时间不能大于等于结束时间');
}

```

完成以上开发，就可以在管理后台自定义新增计划任务的内容了。

### 11.3.3 开发定时任务执行脚本

完成计划任务显示列表管理的开发后，用户就可以自行定义需要的任务内容了，不过想要自动运行，还需要一个自动执行任务的脚本，用来批量处理这些任务。其实现步骤如下。



## 1. 安装第三方依赖类库

为了提升开发效率，这里主要安装两个依赖类库，用来解析 Crontab 表达式和发送 HTTP 请求。在项目的根目录下，安装 mtdowling/cron-expression 的命令如下：

```
composer require mtdowling/cron-expression
```

为了更好地获取请求 URL 地址的 HTTP 返回值，安装使用 rmccue/requests 的命令如下：

```
composer require rmccue/requests
```

## 2. 开发核心任务脚本

为了更好地管理代码，在项目 application 目录下，新增 crontab 应用模块，在此模块下的 controller 目录下，新增 Crontask.php 控制器文件，在此文件内开发任务执行脚本，其步骤如下。

(1) 新增初始化方法。为了防止误操作，此脚本只能在命令行下操作，在 Crontask 控制器文件中，增加以下代码：

```
/**
 * 初始化配置列表
 * @var array
 */
private $_config = [];
/**
 * 初始化方法
 */
public function _initialize()
{
    parent::_initialize();
    config('app_trace', false); //关闭 app_trace
    // 只可以以 cli 方式执行
    if (!$this->request->isCli()){
        $this->error('计划任务必须在命令行中执行');
    }
    // 初始化状态
    $this->_config = [
        'DELETE' => -1, //已经删除
        'DISABLED' => 0, //禁用
        'NORMAL' => 1, //正常
        'COMPLETED' => 2, //完成
        'EXPIRED' => 3 //过期
    ];
}
```

为了脚本稳定地执行，代码强制关闭了框架自带的 Trace 输出模式，并禁止在非命令行模式下运行此脚本的任何方法。其代码如下：



```

config('app_trace', false); // 关闭 app_trace
// 只可以以 cli 方式执行
if (!$this->request->isCli()){
    $this->error('计划任务必须在命令行中执行');
}

```

(2) 实现任务处理核心方法。在控制器中新增 index()方法，增加以下代码：

```

/**
 * 定时任务执行方法
 * @return bool
 */
public function index()
{
    // 查询所有的任务列表
    $map['status'] = ['gt' , 0];
    $crontab_list = Db::name('crontab')->where($map)->select();
    if(!$crontab_list)
    {
        return false;
    }
    $now_time = time();
    // 遍历所有的任务记录
    foreach($crontab_list as $key=>$crontab)
    {
        $is_execute = false; // 是否执行
        $update = []; // 需要更新的数据
        if ($now_time < $crontab['begin_time']) { //任务未开始
            continue;
        }
        if ($crontab['maximums'] && $crontab['executes'] >= $crontab
            ['maximums']) { // 任务超过最大执行次数，任务完成
            $update['status'] = $this->_config['COMPLETED'];
        } else if ($crontab['end_time'] > 0 && $now_time > $crontab
            ['end_time']) { // 任务已过期
            $update['status'] = $this->_config['EXPIRED'];
        } else {
            // 创建计划任务对象并传入时间表达式
            $cron = CronExpression::factory($crontab['schedule']);
            if ($cron->isDue()) {
                // 允许执行
                $is_execute = true;
                // 允许执行的时候更新状态
                $update['execute_time'] = $now_time;
                $update['update_time'] = $now_time;
                $update['executes'] = $crontab['executes'] + 1;
                $update['status'] =
                    ($crontab['maximums'] > 0 && $update['executes'] >=
                    $crontab['maximums']) ?
                    $this->_config['COMPLETED'] : $this->_config
                    ['NORMAL'];
            } else { //如果未到执行时间则跳过本任务去判断下一个任务

```



```

        continue;
    }
}

$map = [];
$map['id'] = $crontab['id'];
// 更新状态
Db::name('crontab')->where($map)->update($update);

// 通过标志位来判断是否满足执行条件（任务是否开始、是否未达到规定次数等），
// 最终决定执行任务还是只更新任务状态
if (!$is_execute) {
    continue;
}

// 执行计划任务操作
try{
    // 判断任务类型
    switch ($crontab['type']) {
        // 请求 URL
        case 'url':
            if (substr($crontab['content'], 0, 1) == "/") {
                // 本地项目 URL
                $request = shell_exec('php ' . ROOT_PATH . 'index.
                php ' . $crontab['content'] . ' 2>&1');
                $this->saveLog('url', $crontab['id'], $crontab
                ['title'], 1, $request);
            } else {
                // 远程 URL
                try {
                    // 使用 Requests 对象请求 URL
                    $request = \Requests::get($crontab['content']);
                    if ($request->success) {
                        $this->saveLog('url', $crontab['id'],
                        $crontab['title'], 1, $crontab['content'] .
                        ' 请求成功, HTTP 状态码: ' . $request->
                        status_code);
                    } else {
                        $this->saveLog('url', $crontab['id'],
                        $crontab['title'], 0, $crontab['content'] .
                        ' 请求失败, HTTP 状态码: ' . $request->
                        status_code);
                    }
                } catch (\Requests_Exception $e) {
                    $this->saveLog('url', $crontab['id'], $crontab
                    ['title'], 0, $crontab['content'] . ' 任务异常:
                    ' . $e->getMessage());
                }
            }
        }
        break;
        case 'shell':
            // 执行命令
            $request = shell_exec($crontab['content'] . ' 2>&1');

```



```
        $this->saveLog('shell', $crontab['id'], $crontab
        ['title'], 1, $request);
        break;
    }
}
catch (\Exception $e)
{
    // 记录异常
    $this->saveLog($crontab['type'], $crontab['id'], $crontab
    ['title'], 0, "执行的内容发生异常:\r\n" . $e->getMessage());
}
}
```

下面来分析下执行任务的实现代码，实际上实现的是程序设计中的流程步骤，首先查询出所有的待处理的任务列表并进行遍历循环操作，保证每一条记录都会被读取执行。其代码如下：

```
$crontab_list = Db::name('crontab')->where($map)->select();
```

然后再判断每个任务是否有执行的价值。其代码如下：

```
$is_execute = false; // 是否执行
$update = []; // 需要更新的数据
if ($now_time < $crontab['begin_time']) { //任务未开始
    continue;
}
if ($crontab['maximums'] && $crontab['executes'] >= $crontab['maximums'])
{ // 任务超过最大执行次数，任务完成
    $update['status'] = $this->_config['COMPLETED'];
} else if ($crontab['end_time'] > 0 && $now_time > $crontab['end_time'])
{ // 任务已过期
    $update['status'] = $this->_config['EXPIRED'];
}
.....
```

在满足所有条件后，最后判断 Crontab 表达式的正确性，根据返回值进行判断，任务是否要执行。其代码如下：

```
// 创建计划任务对象并传入时间表达式
$cron = CronExpression::factory($crontab['schedule']);
if ($cron->isDue()) {
    // 允许执行
    $is_execute = true;
}
.....
```

无论是否需要执行，都会更新任务记录。其代码如下：

```
// 更新状态
Db::name('crontab')->where($map)->update($update);
```

最后再根据是否执行的状态来执行任务。先判断任务类型，再根据不同类型判断是发送请求，还是执行命令脚本，其中向 URL 发送请求的代码如下：

```
// 远程 URL
try {
```



```
// 使用 Requests 对象请求 URL
$request = \Requests::get($crontab['content']);
if ($request->success) {
    $this->saveLog('url', $crontab['id'], $crontab['title'], 1,
        $crontab['content'] . ' 请求成功, HTTP 状态码: ' . $request->
            status_code);
} else {
    $this->saveLog('url', $crontab['id'], $crontab['title'], 0,
        $crontab['content'] . ' 请求失败, HTTP 状态码: ' . $request->
            status_code);
}
} catch (\Requests_Exception $e) {
    $this->saveLog('url', $crontab['id'], $crontab['title'], 0, $crontab
        ['content'] . ' 任务异常: ' . $e->getMessage());
}
```

其中使用到了 saveLog()方法, 就是为了记录请求的返回结果, 该方法定义如下:

```
/**
 * 保存运行日志
 * @param $type 任务类型 url: url, shell: 命令行
 * @param $crontab_id 任务 id
 * @param $title 日志名称
 * @param $status 日志状态 0:失败, 1: 成功
 * @param string $remark 日志详情
 */
private function saveLog($type, $crontab_id, $title, $status, $remark = '')
{
    $data['type'] = $type;
    $data['crontab_id'] = $crontab_id;
    $data['title'] = $title;
    $data['status'] = $status;
    $data['remark'] = $remark;
    $data['create_time'] = time();
    Db::name('crontab_log')->insert($data);
}
```

对于 Shell 类型的命令行操作, 只需要使用 shell\_exec()方法即可。其代码如下:

```
case 'shell':
    // 执行命令
    $request = shell_exec($crontab['content'] . ' 2>&1');
    $this->saveLog('shell', $crontab['id'], $crontab['title'], 1,
        $request);
    break;
```

### 3. 日志列表展示

在 Cron.php 控制器文件中, 新增 log\_list()方法, 目的是为了查询当前任务的执行日志, 核心代码如下:

```
/**
 * 日志列表
 * @return mixed
```



```

* @throws \think\Exception\DbException
*/
public function log_list()
{
    $crontab_id = $this->request->param('id');
    if(!$crontab_id)
    {
        $this->error('参数错误! ');
    }
    $list = Db::name('crontab_log')->where(['crontab_id'=>$crontab_id])
->paginate(20);
    $this->assign('_list' , $list);
    return $this->fetch();
}

```

模板代码在这里不再专门贴出，实现效果如图 11-9 所示。

任务执行日志列表						
ID	任务名称	任务类型	任务ID	日志备注	时间	状态
2	test	url	1	http://www.baidu.com 请求成功, HTTP状态码: 200	2017-12-30 10:39	成功

图 11-9 查看计划任务的执行日志

#### 4. 使用计划任务管理模块

完成基本的程序开发后，需要先增加一个计划任务，每分钟调用一次 Crontab 下的 Crontask 文件的 index()方法。这里以借助 Vagrant 安装的 Ubuntu 虚拟机为例进行介绍，步骤如下。

##### (1) 增加全局 Crontab 表达式

编辑 Crontab 配置文件，增加如下表达式：

```

* * * * * /usr/bin/php7.0 /var/www/wangcmf/public/index.php crontab/
Crontask/index

```

注意这里使用了 ThinkPHP 5 框架，以自带命令行访问控制器方法的形式去执行：

```

/var/www/wangcmf/public/index.php crontab/Crontask/index

```

##### (2) 后台新增计划任务

新增一个访问 URL 地址的计划任务，测试是否可以定时执行，内容如图 11-10 所示。

ID	任务名称	任务类型	任务详情	Cron表达式	次数统计	时间信息	状态
3	自动访问百度首页	url	http://www.baidu.com	*****	最大执行次数: 5 已经执行次数: 0	创建: 2018-01-08 11:57 更新: -- 开始: 2018-01-08 11:58 结束: 2018-01-08 22:57 最后执行: --	完成

图 11-10 新增访问 URL 的计划任务



等待一段时间后，再次刷新页面，发现任务已经执行，并且记录了次数与各种时间信息，如图 11-11 所示。

次数信息	时间信息
最大执行次数: 5	创建: 2018-01-08 11:57
已经执行次数: 1	更新: 2018-01-08 11:58
	开始: 2018-01-08 11:58
	结束: 2018-01-08 22:57
	最后执行: 2018-01-08 11:58

图 11-11 每次执行都会更新次数和时间信息

然后进入日志列表，可以查看执行的结果，如图 11-12 所示。

ID	任务名称	任务类型	任务ID	日志备注	时间	状态
9	自动访问百度首页	url	3	http://www.baidu.com 请求成功, HTTP状态码: 200	2018-01-08 11:58	成功
10	自动访问百度首页	url	3	http://www.baidu.com 请求成功, HTTP状态码: 200	2018-01-08 11:59	成功

图 11-12 查看任务执行日志列表

除了 URL 类型的任务，还可以模拟执行一个 Shell 命令，例如：

```
php -m
```

在日志列表中可以看到所有的返回结果，如图 11-13 所示。

ID	任务名称	任务类型	任务ID	日志备注	时间	状态
14	测试 PHP 命令	shell	4	[PHP Modules] bcmath bz2 calendar Core ctype curl date dom enchant exif fileinfo filter ftp gd gettext hash iconv igbinary imagick imap intl json ldap libxml mbstring mcrypt memcached mongodb msgpack mysqli mysqlnd odbc openssl pcntl pcre PDO pdo_mysql PDO_ODBC pdo_pgsql pdo_sqlite pgsql Phar posix pspell readline redis Reflection session shmop SimpleXML sockets SPL sqlite3 standard sysvmsg sysvsem sysvshm tidy tokenizer wddx xml xmlreader xmlrpc xmlwriter xsl Zend OPcache zip zlib [Zend Modules] Zend OPcache	2018-01-08 13:07	成功

图 11-13 查看执行命令行操作的日志信息



## 第 12 章 基于 Redis 队列的商城抢购系统

随着网站用户量的增多，越来越多的功能模块需要应对高并发、高流量的挑战。很多时候，PHP+MySQL 这样的简单结构已经无法满足实际需求。本章除了讲解一些高并发应用场景下的常见问题解决办法，还会以商城抢购模块为例，使用 Redis 消息订阅与发布结构实现简单的抢购模型，为处理多人抢购商品这样的需求提供一种有效的实现途径。

### 12.1 高并发应用场景分析

本节主要讲解常见的高并发应用场景的解决方案，随后设计一套基于 Redis 任务队列的抢购系统架构。

#### 12.1.1 高并发场景解决方案

传统网站应用中一般都有限时、限量购买等功能。很多时候会发生在极短时间内有大量用户访问（如秒杀商品等）的情况，这会对网站系统带来巨大的考验，而且所带来的超高并发流量也会产生一系列的问题。常见的问题有以下几种。

##### 1. 服务器宕机

无论是 Apache 服务器，还是 Nginx 服务器，在一定的硬件配置下（如四核 CPU、8GB 内存），其在固定时间内可以处理的数据量是一定的，例如单台可以处理并发数量为 200 个，但是一下子涌进来 20 000 个用户同时进行操作，服务器的处理能力就会急速下滑，甚至最终导致宕机。此时，重启服务器并不能解决问题。因为重启后的服务器，仍旧不能满足处理需求，无法解决根本问题。此时经常会使用以下方案来缓解压力。

##### (1) 增加应用服务器

对于单台应用服务器的架构，无法通过无限地升级单台服务器的硬件配置来提升性能，毕竟服务器软件本身也会有瓶颈。为了解决这个问题，引入了服务器集群，即使用负载均衡器，把请求分发到各个应用服务器中，减缓单台应用服务器的压力，提高服务器的



处理能力。常见的负载均衡架构如图 12-1 所示。

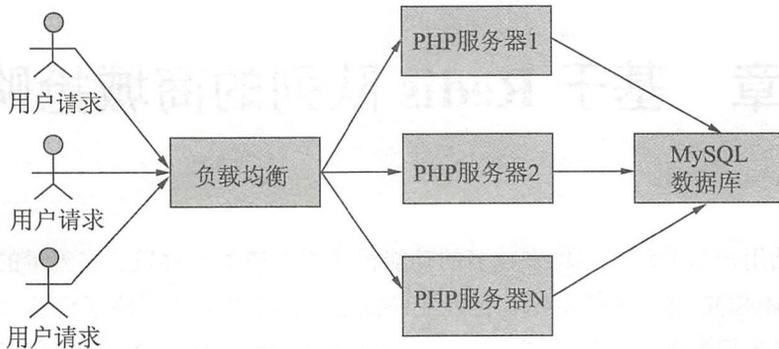


图 12-1 通过负载均衡进行应用服务器的扩展

负载均衡有很多方式，从专门的硬件，到 Nginx 软件层面，都可以实现，这里不再过多展开。服务器集群可以显著地提升性能，但受限于项目规模和成本，一般不会无限制地增加。

### (2) 减少用户重复请求

例如在商品抢购这个应用场景下，暂且约束用户每次只能抢购一个商品，那么每个账号在购买前，都需要查询是否购买过此商品，每个请求的处理流程如图 12-2 所示。

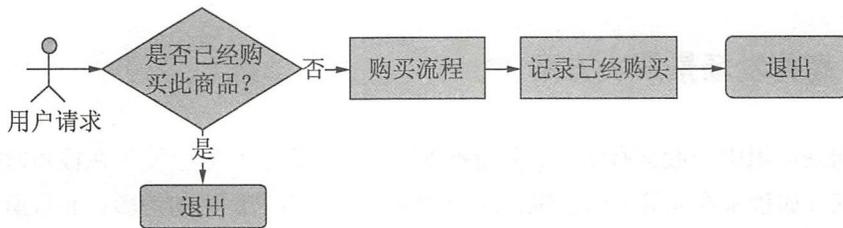


图 12-2 检测用户是否已经抢购过商品的流程

此时该用户若同时开启多个浏览器，或使用脚本发送大量的查询请求，如图 12-2 所示的流程就会反复地处理，极大增加服务器负担。此时可以使用缓存服务（如 Redis），对用户的请求进行标记，只要用户发送过一次查询请求，第二次发送时，就可以强制用户退出，其流程如图 12-3 所示。

### (3) 禁止单一用户多浏览器同时登录

为了防止用户多账号同时登录导致的僵尸操作，可以让用户只能在一端登录，其他登录都会“踢掉”现有的登录。

## 2. 订单超发

订单超发是常见的高并发问题。例如在商品抢购场景，当商品还剩最后一个库存的时



候，有 2 个及以上的用户执行下单抢购逻辑，服务器处理并发请求，在同一时间内，所有的用户都查询（MySQL）到还剩下一个库存，通过了库存的判断，继而进入到下单操作，导致订单超发。解决这类问题主要有以下两种解决方案。

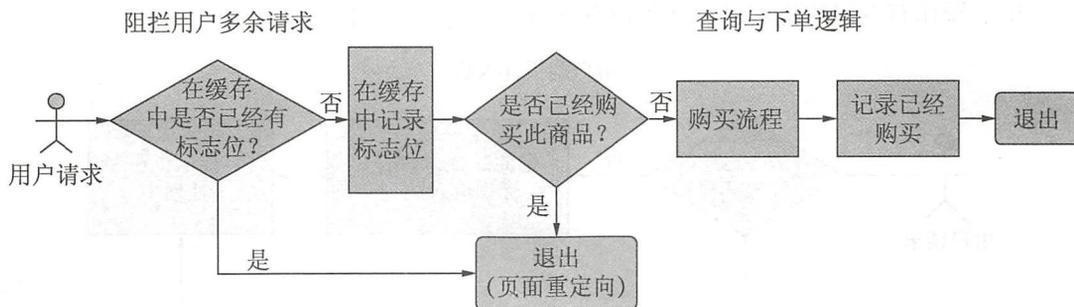


图 12-3 使用缓存阻挡用户重复请求的流程

### (1) 悲观锁与乐观锁

悲观锁和乐观锁都有数据库的特性。先来看悲观锁，使用这种方式，无论有多少个请求，当前正在处理的请求只有一个，并在处理前进行加锁操作，期间其他请求处于等待的状态。当前请求处理完毕后，数据库会自动解除锁定状态，其他的请求再依次进行处理。

这种方式虽然保证了数据的一致性，但当在高并发场景下，会有大量的请求在等待“锁（Lock）”，可能会出现某些请求无法抢到商品，也就导致了常见的死锁，导致系统处理能力降低，响应时间变长。

相比悲观锁，乐观锁允许所有的请求同时处理，每个请求都带有一个唯一的版本号（Version），最后只有满足条件的版本号才可以执行成功，也防止了死锁的出现。使用乐观锁限制订单超发的执行流程如图 12-4 所示。

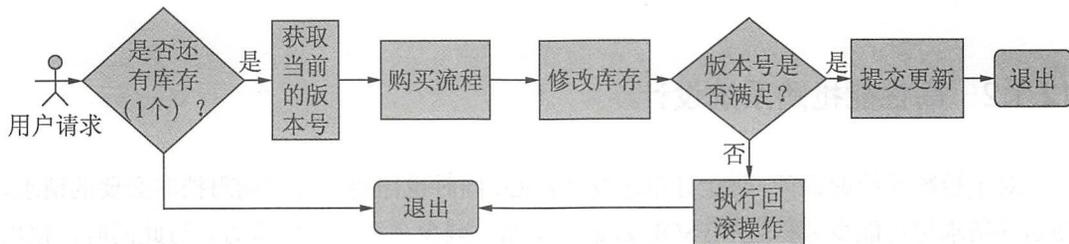


图 12-4 使用乐观锁限制订单超发

### (2) 抢购队列

抢购队列的方式，实际上是规避了并发请求直接到达数据库的问题。所有的请求统一

进入到队列（内存），遵循先进先出的规则，逐一消费处理。借助常见的队列工具，如 Beanstalkd、RabbitMQ 等，可以很好地实现抢购队列模型。而入队和出队的两个操作相对独立，其中入队流程如图 12-5 所示。

出队操作有专门的进程进行队列消费处理，流程如图 12-6 所示。

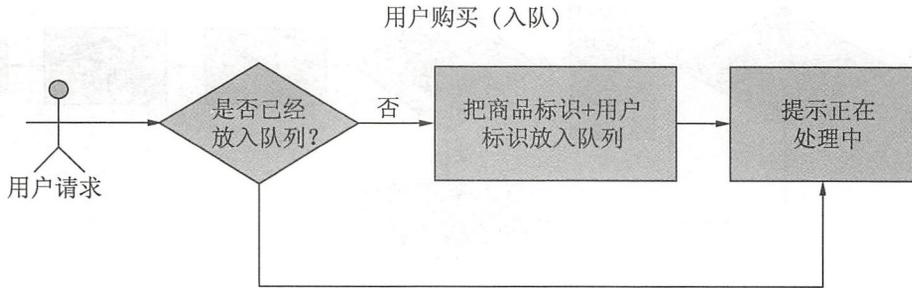


图 12-5 用户信息入队列流程

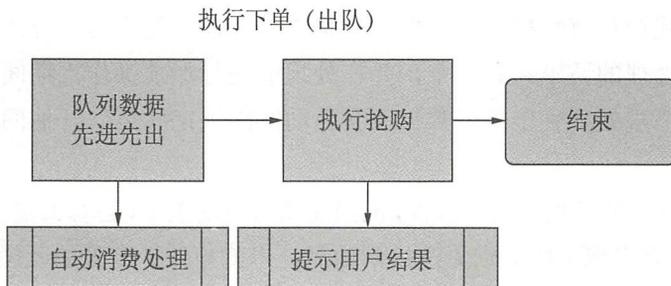


图 12-6 队列消费处理与执行抢购操作流程

**提示：**使用基于内存存储的队列工具，要注意数据丢失问题，队列消费的程序脚本需要有较高的稳定性。

## 12.1.2 高性能抢购系统设计

对于抢购系统的设计，防止订单超发是核心，同时也需要在客户端阻挡非必要的请求，使这些请求尽可能少地抵达 MySQL 数据库层面，减少系统的整体压力。与此同时，使用抢购队列屏蔽并发请求信息，其核心技术可以使用 Redis 本身自带的消息发布订阅机制，结合守护进程与计划任务，实现一个简易的抢购系统模型。

实际上一个完整的抢购系统，就是一个完整的电商系统，不仅包含基本的用户系统、商品系统和订单系统，还有完整的后台管理，不过本章重点在 Redis 任务队列的实际应用，

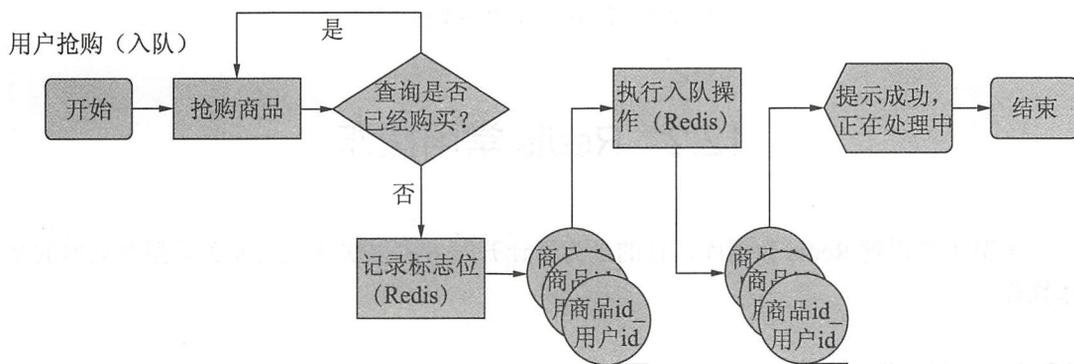
所以不再专门设计这几个配套系统,在进行示例项目开发的时候,也尽量简化这几个部分,减少读者额外的理解难度。

### (1) 用户抢购逻辑

用户登录进入商品详情页后,就可以进行抢购操作了,对应的就是入队列操作。设计时应包含下面两部分功能。

- 判断用户是否已经抢购过当前商品。为了避免直接对 MySQL 数据库进行操作,用户每次的抢购请求都会记录在 Redis 中,存储格式为 Hash 表。此表的名称为“goods\_商品 id”,这样每个商品就有一个唯一的 Hash 表,用户一旦单击过“抢购商品”按钮后,抢购请求就会存储在当前商品的 Hash 表中,存储的字符串格式为:“商品 id\_用户 id”,下次再发出请求的时候,如果查询到已存储的请求则提示用户已经抢购过,不再进行下一步操作。
- 抢购信息入队列。通过查询是否购买的验证后,系统自动向队列写入“商品 id\_用户 id”这样的数据,这两个信息构成了是否能下单的唯一信息。完成抢购信息入队列操作后,提示用户耐心等待队列出队操作结果。

相应的流程如图 12-7 所示。



商品购买标志位Hash表 基于发布订阅的任务队列

图 12-7 验证是否抢购和入队列操作的流程

### (2) 队列消费的逻辑

本着队列先进先出的原则,无论有多少个用户请求在不停地进入队列,都是依据谁先来谁先抢到的原则执行(Redis 单线程执行)。一个队列只对应一个消费进程,此进程不停地出队操作。同时进程脚本每隔一定时间被调用一次,可以使用 Crontab 计划任务或者守护进程实现(本实例采用 Redis 的订阅机制,自动监听消息,实现更实时的队列消费),执行流程如图 12-8 所示。

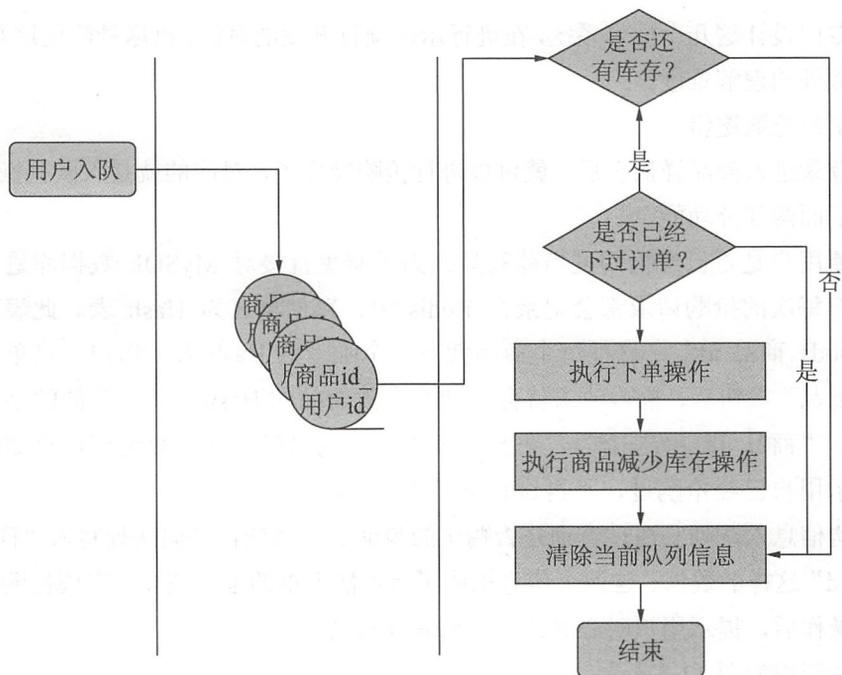


图 12-8 抢购商品队列消费的逻辑

## 12.2 Redis 常用操作

本节主要讲解 Redis 数据库，目的是为了让开发者可以快速入门非关系型数据库的基本操作。

### 12.2.1 Redis 简介

Redis 是知名的开源内存数据库，其数据存储基于 key-value 结构，与 Memcache 类似，常用作缓存或者消息队列。相比传统的关系型数据库，Redis 除了可以高速处理数据的读写外，还有以下几个特点。

- 支持数据的持久化。与 Memcache-断电则数据丢失的特性不同，Redis 可以将内存中的数据保存到硬盘中，重启后还可以继续使用。
- 具有丰富的数据类型。Redis 不仅提供字符串类型的存储方式，还支持列表、哈希表、集合、有序集合等结构操作。
- 支持分布式部署和主从同步。
- 支持主流语言，对 PHP 语言有良好支持。

Redis 支持多种操作系统，无论是在 Windows 操作系统下，还是 Linux 操作系统下都可以找到合适的安装方式。在基于 Linux 的 Ubuntu 系统下可以使用 apt 包管理工具安装，在 Windows 操作系统下，官方提供了安装包文件，如图 12-9 所示。

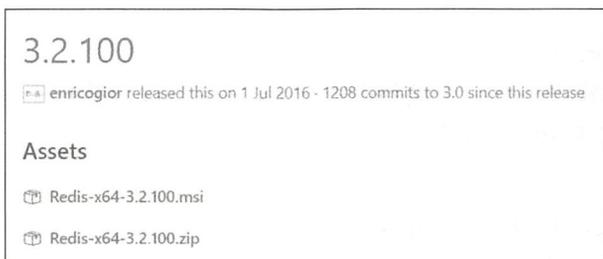


图 12-9 下载并安装 Redis 数据库

完成安装后，Redis 自动安装了服务端和客户端，服务端负责数据库的运行，客户端则给用户提供了操作的命令入口。以 Windows 操作系统为例，Redis 的服务端运行效果如图 12-10 所示。

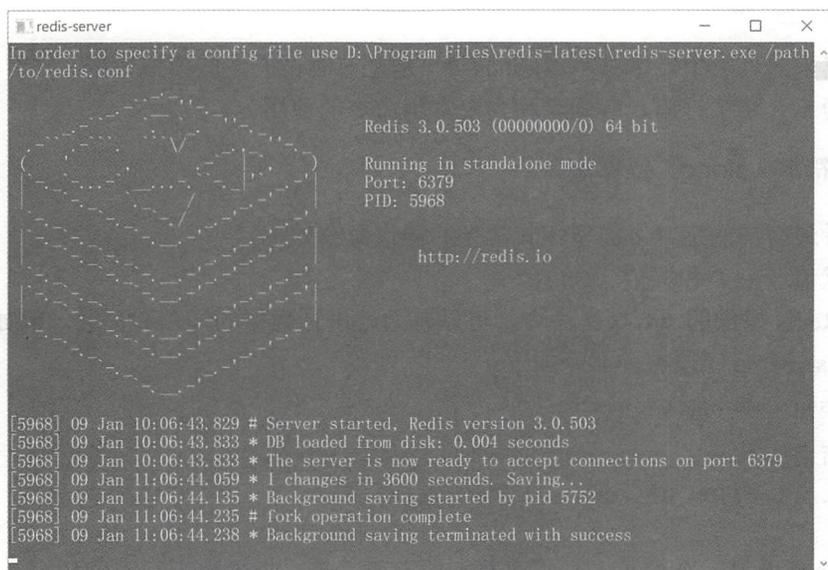


图 12-10 在 Windows 下运行的 Redis 服务端

## 12.2.2 Redis 常用命令操作

Redis 本身支持多种数据结构，每种数据结构都有对应的命令可供用户使用，这里以 Linux 操作系统为例，在 Ubuntu 下安装的 Redis 服务会自动运行。Redis 常见的数据结构有：

- 字符串 (string) ;
- 散列表 (hash) ;
- 列表 (list) ;
- 集合 (set) ;
- 有序集合 (sorted set) 。

使用 Vagrant 创建虚拟机, 连接到命令行下, 使用 apt 包管理工具安装完成 Redis 后, 输入以下命令, 打开 Redis 的客户端。

```
redis-cli
```

若出现以下效果, 则说明连接正常。

```
127.0.0.1:6379>
```

此时查看所有的数据, 可以使用以下命令:

```
keys *
```

执行完成后会列出所有的数据列表:

```
127.0.0.1:6379> keys *  
1) "goods_123456"  
2) "goods_4"
```

打印出来的列表中, 字符串 “goods\_\*” 是之前测试产生的模拟数据。

下面简单介绍字符串、散列表和列表常用命令的使用。

### 1. 字符串 (string) 操作命令

先来看最简单的字符串类型, 其常见的操作命令如下。

#### (1) 赋值 (set) 和取值 (get)

使用 Redis 提供的 set、get 命令, 可以有效地进行数据的写入和读取。例如, 把姓名信息写入 username 字段中, 命令如下:

```
set username wangjialin
```

执行结果如下:

```
127.0.0.1:6379> set username wangjialin  
OK
```

set 命令后面第一个参数, 是需要写入的字段, 第二个参数为写入的数据。

需要读取 username 值, 可以使用 get 命令, 此命令只有一个参数, 就是需要获取的字段, 例如:

```
get username
```

执行结果如下:

```
127.0.0.1:6379> get username  
"wangjialin"
```

#### (2) 批量赋值 (mset) 和批量取值 (mget)

同时 Redis 也提供了批量操作字符串的方法，使用 `mset`、`mget` 命令，可以一次操作多个字段的数据，例如执行以下命令：

```
mset username wangjialin age 18
```

执行结果如下：

```
127.0.0.1:6379> mset username wangjialin age 18
OK
```

在使用 `mget` 读取 `username` 和 `age` 的值，结果如下：

```
127.0.0.1:6379> mget username age
1) "wangjialin"
2) "18"
```

### (3) 递增 (`incr`) 和递减 (`decr`)

若字符串存储的数据是为了进行整数运算，使用 `incr`、`decr` 命令，进行递增、递减操作，比如对 `age` 进行递增，可以执行以下命令：

```
incr age
```

执行完成后，发现 `age` 字段的值，已经变为 19，执行结果如下：

```
127.0.0.1:6379> incr age
(integer) 19
```

`decr` 命令类似，在这里就不再重复演示。

 提示：更多 Redis 的命令说明，可以访问官方手册：<https://redis.io/commands>。

## 2. 散列表 (hash) 操作命令

散列表又称为哈希表，本身是一种键值数据结构，不支持嵌套其他数据类型，只支持字符串。Redis 为散列表提供了众多的操作命令，常见的命令如下。

### (1) 赋值 (`hset`) 和取值 (`hget`)

`hset`、`hget` 命令的使用，与 `set`、`get` 类似，只是增加了一个参数，用法如下：

```
hset 散列表名 字段 值
hget 散列表名 字段
```

例如，在 `userinfo` 这个散列表里面，增加 `username`、`age` 两个键值，操作命令如下：

```
hset userinfo username wangjialin
hset userinfo age 18
```

执行结果如下：

```
127.0.0.1:6379> hset userinfo username wangjialin
(integer) 1
127.0.0.1:6379> hset userinfo age 18
(integer) 1
```

使用 `hget` 获取 `userinfo` 的值：

```
127.0.0.1:6379> hget userinfo username
"wangjialin"
```

```
127.0.0.1:6379> hget userinfo age
"18"
```

#### (2) 批量赋值 (hmset) 和批量取值 (hmget)

和字符串操作一样，对散列表也可以批量地进行赋值、取值的操作，例如批量修改 `userinfo` 和 `age` 的值为新的值，命令为：

```
hmset userinfo username zhangsan age 28
```

执行结果如下：

```
127.0.0.1:6379> hmset userinfo username zhangsan age 28
OK
```

批量取值的操作如下：

```
hmget userinfo username age
```

执行结果如下，发现原有的数据已经被新的数据覆盖：

```
127.0.0.1:6379> hmget userinfo username age
1) "zhangsan"
2) "28"
```

#### (3) 获取键中所有字段值 (hgetall)

使用 `hgetall` 命令可以更方便地获取指定散列表中的所有字段值，例如先给 `userinfo` 增加一些新的字段值，执行结果如下：

```
127.0.0.1:6379> hmset userinfo mobile 13511112222 address beijing
status 1 is_admin 1
OK
```

随后获取 `userinfo` 所有的字段值，使用 `hgetall` 不需要传入字段名，只需要传入散列表名即可：

```
127.0.0.1:6379> hgetall userinfo
1) "username"
2) "zhangsan"
3) "age"
4) "28"
5) "mobile"
6) "13511112222"
7) "address"
8) "beijing"
9) "status"
10) "1"
11) "is_admin"
12) "1"
```

#### (4) 判断字段是否存在 (hexists)

使用 `hexists` 命令可以判断字段是否存在，如存在则返回 1，否则返回 0。例如刚才在散列表 `userinfo` 中设置了 `username` 字段，但是没有设置 `name` 字段，所以使用该命令的操作和执行结果如下：

```
127.0.0.1:6379> hexists userinfo username
(integer) 1
```

```
127.0.0.1:6379> hexists userinfo name
(integer) 0
```

(5) 获取所有键 (hkeys) 和值 (hvals)

这两个命令较好理解, 针对 userinfo 散列表, hkeys 命令的操作和执行结果如下:

```
127.0.0.1:6379> hkeys userinfo
1) "username"
2) "age"
3) "mobile"
4) "address"
5) "status"
6) "is_admin"
```

hvals 命令的操作和执行结果如下:

```
127.0.0.1:6379> hvals userinfo
1) "zhangsan"
2) "28"
3) "13511112222"
4) "beijing"
5) "1"
6) "1"
```

(6) 删除字段值 (hdel)

例如删除 age 字段值, 执行命令的操作和结果如下:

```
127.0.0.1:6379> hdel userinfo age
(integer) 1
```

返回值为 1, 说明已经删除成功, 再次执行 hgetall 命令, 发现 age 字段值已经被删除。

### 3. 列表 (list) 操作命令

相比散列表, Redis 列表类似 PHP 中的索引数组, 索引为从 0 递增的整数。操作方式符合链表的数据结构, 常见的操作命令如下。

(1) 将数据插入列表 (lpush、rpush)

例如往商品列表中插入三件商品名称, 操作与执行结果如下:

```
127.0.0.1:6379> lpush goods_list ps4 xboxonex switch
(integer) 3
```

lpush 将数据从左到右插入列表, rpush 则相反。

(2) 读取列表值 (lrange)

值的获取通过 lrange 和索引操作, 例如获取 good\_list 列表中所有的列表值, 可以操作如下:

```
lrange good_list 0 -1
```

返回结果如下:

```
127.0.0.1:6379> lrange goods_list 0 -1
1) "switch"
2) "xboxonex"
3) "ps4"
```

若需要获取名为 `switch` 的数据，则需要执行以下命令：

```
lrange goods_list 0 0
```

结果如下：

```
127.0.0.1:6379> lrange goods_list 0 0
```

```
1) "switch"
```

`lrange` 的第一个参数为列表名，第二个参数为索引开始位置，第三个参数为获取索引开始的几个参数，所以想要获取前两个值，则执行以下命令：

```
lrange goods_list 0 1
```

相当于从索引 0 开始，获取了 0 和 1 两个值，结果如下：

```
127.0.0.1:6379> lrange goods_list 0 1
```

```
1) "switch"
```

```
2) "xboxonex"
```

### (3) 弹出列表中的数据 (`lpop`、`rpop`)

使用 `lpop` 和 `rpop` 命令，可以弹出列表中的第一个元素或最后一个元素。这里还是以 `goods_list` 列表为例，执行以下命令，弹出列表的第一个元素，执行结果如下：

```
127.0.0.1:6379> lpop goods_list
```

```
"switch"
```

再查看所有元素，结果如下：

```
127.0.0.1:6379> lrange goods_list 0 -1
```

```
1) "xboxonex"
```

```
2) "ps4"
```

列表的命令还有很多，这里不再过多地演示，有需求的读者可以参考官方手册。

提示：可以使用命令“`del 键名`”，删除 Redis 中的数据，不限数据类型。

## 12.2.3 Redis 消息发布/订阅机制

为了实现商品抢购队列，需要先理解 Redis 的消息发布和订阅机制。Redis 发布/订阅 (`pub/sub`) 是一种消息通信模式，发布者 (`pub`) 发送消息到特定的频道 (`channel`)，订阅者 (`sub`) 通过观察频道 (`channel`) 接收消息。这种模式其实类似在收音机上听路况信息，电台主持人会不定时地发送路况信息 (`pub`)，用户 (`sub`) 通过特定的频道 (`channel`)，关注和获取实时信息。

在 Redis 中，每个频道都可以被多个客户端订阅，没有数量上的限制，当没有消息发布的时候，每个客户端都保持订阅的状态。Redis 客户端订阅频道结构图如图 12-11 所示。

此时，若有发布者向特定的频道发布了消息，所有监听当前频道的客户端都可以接收到消息，如图 12-12 所示。

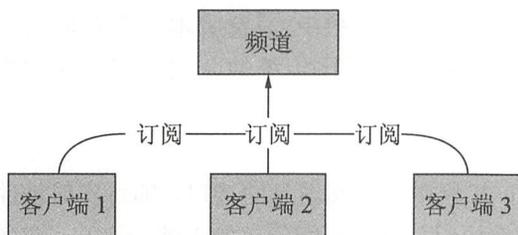


图 12-11 Redis 客户端订阅频道结构图

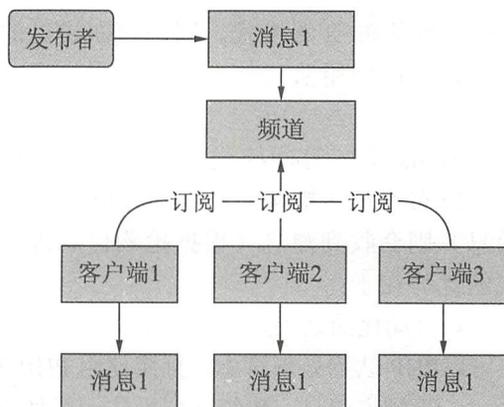


图 12-12 Redis 客户端订阅发布的消息

在实际使用中，Redis 主要使用 `subscribe` 和 `publish` 命令分别实现消息订阅和发布的操作。下面进行简单演示。

首先打开 Redis 客户端，创建名为 `gameneews` 的频道名，命令如下：

```
subscribe gameneews
```

执行后，在命令行中展示结果如下：

```
127.0.0.1:6379> subscribe gameneews
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "gameneews"
3) (integer) 1
```

此时不要关闭 Redis 命令终端，再次打开一个新的命令终端执行发布消息的操作，命令如下：

```
publish gameneews 'This is a new Message!'
```

执行后的返回结果如下：

```
127.0.0.1:6379> publish gameneews 'This is a new Message!'
(integer) 1
```

以上命令，使用 `publish` 向名为 `gameneews` 的频道，发送了一条信息。此时切换到 `subscribe` 订阅的 Redis 命令终端，发现客户端已经接收到消息：

```
127.0.0.1:6379>subscribe gameneews
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "gameneews"
3) (integer) 1
1) "message"
2) "gameneews"
3) "This is a new Message!"
```

除了发布和订阅两个命令外，Redis 还提供了批量订阅/发布消息、解除订阅等各种命令。因为在抢购系统中暂时用不到，所以这里就不过多展开，仅重点说明为何要用

Redis 的发布/订阅模式。回归前面讲过的内容，为了解决订单超发、用户订单竞争的关系，可以使用队列的方式存储用户唯一的抢购识别码，在队列中，先进先出，统一处理。

Redis 发布/订阅模式中的频道（channel），实际上实现了队列的功能，发布者（用户）发布消息（抢购信息）到频道中（队列），订阅者（PHP 消费脚本负责获取队列消息）则会收到消息（根据抢购信息进行后续操作）。相比传统的队列，使用这种模式的优势在于：

- 自动化的队列入队、出队操作，不用手动编写处理脚本。
- PHP 队列处理脚本，直接使用 PHP Redis 扩展中的订阅消息的接口，即可实时地监听频道消息，传统队列操作一般使用轮询访问，有时间间隔（如 1 秒一次），时效性不高。

**提示：**除了 Redis，使用 MySQL、Memcache 和 Beanstalk 也可以实现队列功能的支持，它们本质上没有区别，只是在数据的存储格式、提供的功能组件上有所区别，开发者可以根据自己的实际需求，进行相应的选择。

## 12.2.4 可视化管理 Redis 数据——phpRedisAdmin

与使用 phpMyAdmin 进行 MySQL 的数据管理类似，Redis 也有基于 PHP 实现的可视化管理工具 phpRedisAdmin，并且可以免客户端安装，相比命令行管理，可视化管理使用的效率更高。下面简单讲解 phpRedisAdmin 的安装和使用。

### 1. 安装phpRedisAdmin

以 Vagrant 工具安装 Ubuntu 14.04 虚拟系统为例，使用 Git 可以方便地安装可视化管理工具，执行以下命令：

```
git clone https://github.com/ErikDubbelboer/phpRedisAdmin.git
cd phpRedisAdmin
git clone https://github.com/nrk/predis.git vendor
```

随后在 Nginx 服务器中为 phpRedisAdmin 配置虚拟主机，部分配置命令如下：

```
server {
    listen 8082;
    root /var/www/phpRedisAdmin;
    index index.html index.htm index.nginx-debian.html index.php;
    .....
}
```

重启 Nginx 服务器后，在浏览器中访问 8082 端口，可以看到 phpRedisAdmin 的使用界面，如图 12-13 所示。



图 12-13 phpRedisAdmin 使用界面

## 2. 使用phpRedisAdmin可视化管理工具

phpRedisAdmin 可视化管理工具在使用起来也非常简单，这里以添加一个 Hash 散列表为例，介绍如何使用 phpRedisAdmin。

### (1) 进入数据新增界面

单击页面左侧菜单 Add another key 选项，进入数据新增界面，如图 12-14 所示。

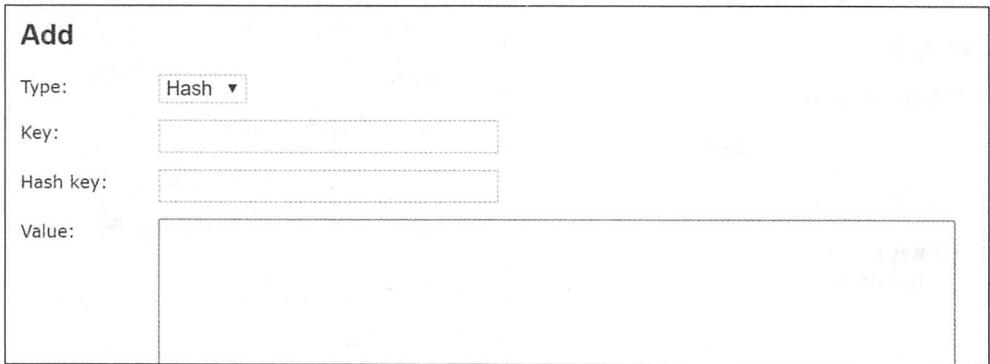


图 12-14 在 Redis 中添加新数据界面

可以看到新增界面中可以选择数据类型、定义对应的 key 值和 value 值。

## (2) 添加 Hash 散列表数据

在新增界面中选择 Type（类型）为 Hash，如图 12-15 所示。



**Add**

Type: Hash ▾

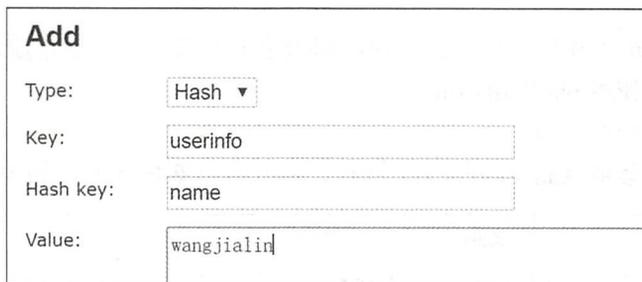
Key:

Hash key:

Value:

图 12-15 选择 Hash 类型的数据结构

选中后根据当前数据类型增加数据，如图 12-16 所示。



**Add**

Type: Hash ▾

Key: userinfo

Hash key: name

Value: wangjialin

图 12-16 新增 Hash 类型数据

单击 Add 按钮后，就可以成功地向 Redis 中新增数据了。

## (3) 查看已有数据

新增数据完毕后，左侧菜单中出现了 userinfo 选项，单击它就可以查看添加到 Hash 散列表中的值了，如图 12-17 所示。

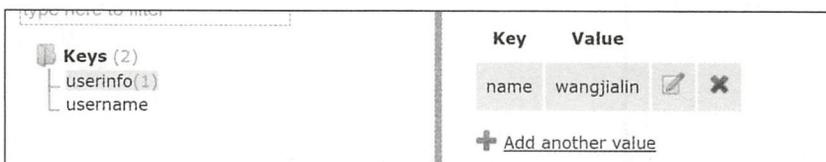


图 12-17 查看已经添加的数据

除此之外还可以进行修改、删除和新增数据的操作，并且相比命令行操作更方便。下面继续新增一些 Hash 散列表中的值，结果如图 12-18 所示。



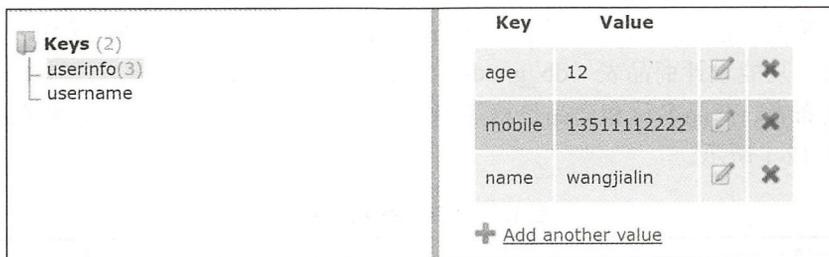


图 12-18 继续添加 Hash 散列表中的值

## 12.3 实现简单商城网站

实现抢购系统，需要一个较为完整的商城网站，但因为个人开发者申请支付较为困难，因此这里把下单成功与支付合并，默认下单成功的同时模拟支付成功，下面来看实现流程。

### 12.3.1 程序设计与数据库设计

传统的商城系统，仅仅是商品类型就不只一种，关联关系非常复杂。为了尽可能地简化系统的复杂度，只留下用户、商品和订单模块。简易商城系统结构导图如图 12-19 所示。

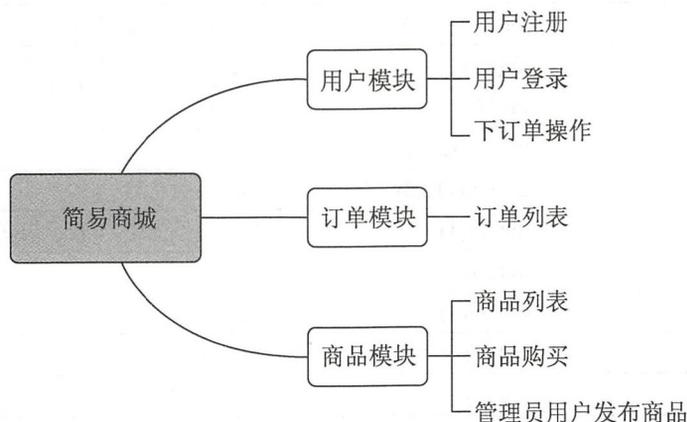


图 12-19 简易商城系统结构导图

简易商城基于内容管理框架开发，复用用户相关表结构，其几个功能模块简单说明如下。

- 用户模块：除了包含用户注册、登录外，对用户的角色进行区分，普通用户只能查看和购买商品，而管理员还可以发布商品。
- 订单模块：除了隐藏的下单逻辑外，在页面展示上只有一个订单列表。



- 商品模块：包含商品列表展示、商品详情展示和隐含的商品购买逻辑。

根据以上功能设计商品表（db\_goods）和商品订单表（db\_orders）。其中商品表中，记录商品发布者、商品名称和商品价格等基本信息，与用户表（db\_user）相关联。其表字段说明如表 12-1 所示。

表 12-1 商品表字段说明

列（字段）	类 型	注 释
id	int(4) 自动增量	主键自增长
name	varchar(255)	商品名称
image	varchar(255)	商品图片
desc	text	商品简介
sell_price	decimal(12,2)	商品销售价格
stock	int(10)	商品库存
user_id	int(10)	商品发布用户
status	tinyint(1)	商品状态
create_time	int(10)	商品发布时间
update_time	int(10)	商品更新时间

商品订单表与商品表、用户表相关联，其详细的字段说明如表 12-2 所示。

表 12-2 商品订单表字段说明

列（字段）	类 型	注 释
id	int(4) 自动增量	订单ID，主键自增长
goods_id	int(10)	商品ID，一个订单对应一个商品
user_id	int(10)	用户ID
order_no	varchar(255)	订单号
pay_price	decimal(12,2)	实际支付价格
is_pay	tinyint(1)	是否已经支付，0表示未支付，1表示已支付
pay_time	int(10)	支付时间
status	tinyint(1)	状态
create_time	int(10)	新增时间
update_time	int(10)	更新时间

### 12.3.2 商城首页

继续使用 wangcmf 项目，并在此基础上进行扩展开发。商城应用使用 application/index 模块。下面介绍首页的代码实现逻辑。



## 1. 全局模板

全局模板使用 Bootstrap 框架和 ThinkPHP 5 框架的模板继承，在 application\index\view 目录下，新增 public 公共模板，随后在该模板目录下新建 header.html、nav.html、layout.html 和 footer.html 几个模板文件，分别对应头部、菜单导航、全局布局和底部几个页面区间。使用方法和引入 AdminLTE 模板样式类似，这里不再逐个贴出。其中的 layout.html 文件内容如下：

```
{include file="public/header" /}
{include file="public/nav" /}
<div class="container">
    {block name="container"}{/block}
</div> <!-- /container -->
{include file="public/footer" /}
{block name="script"}{/block}
```

## 2. 首页控制器与视图

在 application\index\controller 目录下，新增 Index.php 控制器文件，随后新增 \_initialize() 方法，进行配置加载等操作。代码如下：

```
/**
 * 初始化方法
 */
protected function _initialize()
{
    parent::_initialize();
    // 读取缓存中的配置
    $config = cache('DB_CONFIG_DATA');
    if(!$config)
    {
        // 在数据库中读取所有的配置信息
        $config = Config::lists();
        cache('DB_CONFIG_DATA' , $config);
    }
    // 数据库配置合并到系统配置
    config($config);
}
```

随后定义 index() 方法，展示首页的商品列表，代码如下：

```
/**
 * 商品列表
 * @return mixed
 */
public function index()
{
    // 查询条件
    $map['status'] = 1;
    // 商品列表（分页）
    $this->assign('_list' , Db::name('goods')->where($map)->paginate(6));
    return $this->fetch();
}
```



最后在 `application\index\view` 目录下,新建 `index` 目录,然后在该目录下新增 `index.html` 模板文件。首页商品列表展示的核心代码如下:

```
{extend name="public:layout" /}
{block name="container"}
<div class="row">
    {volist name="_list" id="vo"}
    <div class="col-sm-6 col-md-4">
        <div class="thumbnail">
            
            <div class="caption">
                <h3>{$vo.name}</h3>
                <p>售价: ¥{$vo.sell_price}元 库存: <span class="">
                    {$vo.stock}</span></p>
                <p>
                    <a href="{:url('detail', ['id'=>$vo['id']])}"
                        class="btn btn-danger" role="button">立即购买</a>
                </p>
            </div>
        </div>
    </div>
    {/volist}
</div>
<div class="row">
    <nav aria-label="Page navigation">
        <ul class="pagination">
            { $ _list -> render () }
        </ul>
    </nav>
</div>
{/block}
```

### 12.3.3 用户注册、登录

商城系统中,商品在用户未登录的状态下一般无法购买,所以用户注册和登录是必要的,下面来看用户模块实现的步骤。

#### 1. 用户注册

在 `application\index\view` 目录下,新增 `user` 目录,在 `application\index\controller` 目录下,新增 `User.php` 用户控制器,新增 `reg()` 方法,代码如下:

```
/**
 * 用户注册
 */
public function reg()
{
    if($this->request->isPost())
    {
        // 获取用户提交的注册信息
        $mobile = input('mobile');
```



```

$username = input('name');
$password = input('password');
$repassword = input('repassword');
$data = $this->request->param();
// 数据效验
if(!$username || !$mobile || !$password)
{
    $this->assign('error_tips' , '请输入用户名、手机号或密码');
}
elseif($password != $repassword)
{
    $this->assign('error_tips' , '两次密码不一致');
}
elseif(Db::name('user')->where(['mobile'=>$mobile])->find())
{
    $this->assign('error_tips' , '手机号已经存在');
}
else
{
    // 注册新用户
    $data['status'] = 1;
    $data['create_time'] = time();
    $data['password'] = sys_md5($data['password']);
    unset($data['repassword']);
    $insertId = Db::name('user')->insert($data);
    if($insertId)
    {
        // 注册成功后, 自动跳转到登录界面
        $this->redirect( 'User/login');
    }
}
// 用户注册输入的信息, 替换到注册表单中
$this->assign('data' , $data);
}
return $this->fetch();
}

```

在 application\index\view\user 目录下, 新增 reg.html 模板文件, 用户注册表单的核心代码如下:

```

<form action="{:url('reg')}" method="post">
    <div class="form-group">
        <label for="">用户名</label>
        <input type="text" name="name" value="{:isset($data)?$data
        ['name']:'}" class="form-control" id="" placeholder="用户名">
    </div>
    <div class="form-group">
        <label for="">手机号</label>
        <input type="text" name="mobile" value="{:isset($data)?$data
        ['mobile']:'}" class="form-control" id="" placeholder="手机号">
    </div>
    <div class="form-group">
        <label for="">密码</label>

```



```

        <input type="password" name="password" class="form-control" id=""
        placeholder="密码">
    </div>
    <div class="form-group">
        <label for="">再次输入</label>
        <input type="password" name="repassword" class="form-control" id=""
        placeholder="再次输入密码">
    </div>
    <div>{$error_tips|default=''}</div>
    <button type="submit" class="btn btn-default">用户注册</button>
</form>

```

在 nav.html 公共模板目录下，用户登录的入口代码如下：

```

{if condition="!session('USER_ID')"}
    <ul class="nav navbar-nav navbar-right">
        <li><a href="{:url('User/reg')}"}>注册</a></li>
        <li><a href="{:url('User/login')}"}>登录</a></li>
    </ul>
{else/}
    <ul class="nav navbar-nav navbar-right">
        <li><a href="/">{:session('USER_NAME')}</a></li>
        <li><a href="{:url('Index/order_list')}"}>我的订单</a></li>
        {if condition="session('USER_ID') == intval(config('USER_ADMIN'))"}
            <li><a href="{:url('Index/add')}"}>发布抢购商品</a></li>
        {/if}
        <li><a href="{:url('User/logout')}"}>注销登录</a></li>
    </ul>
{/if}

```

需要注意的是，在模板中对用户是否是管理员进行了判断，而且必须是超级管理员才可以进行商品发布，代码如下：

```

{if condition="session('USER_ID') == intval(config('USER_ADMIN'))"}
    <li><a href="{:url('Index/add')}"}>发布抢购商品</a></li>
{/if}

```

访问注册页，效果如图 12-20 所示。

图 12-20 商城用户注册页



## 2. 用户登录

为了实现用户登录的功能，继续修改 User.php 用户控制器文件，新增 login() 方法，代码如下：

```
/**
 * 用户登录
 * @return mixed
 */
public function login()
{
    if($this->request->isPost())
    {
        // 接收参数
        $mobile = input('mobile');
        $password = input('password');
        // 查询用户手机号和密码是否正确
        $userinfo = Db::name('user')->where(['mobile'=>$mobile, 'password'=>sys_md5($password)])->find();
        if(!$mobile || !$password)
        {
            $this->assign('error_tips' , '请输入手机号或者密码');
        }
        elseif(!$userinfo)
        {
            $this->assign('error_tips' , '用户名或者密码错误');
        }
        else
        {
            // 用户信息写入会话
            session('USER_ID' , $userinfo['id']);
            session('USER_NAME' , $userinfo['name']);
            // 页面重定向到首页
            $this->redirect( 'Index/index' );
        }
    }
    return $this->fetch();
}
```

在 application\index\view\user 目录下，新增 login.html 文件，用户登录表单的核心代码如下：

```
<form action="{:url('login')}" method="post">
    <div class="form-group">
        <label for="">手机号</label>
        <input type="text" name="mobile" class="form-control" id=""
            placeholder="手机号">
    </div>
    <div class="form-group">
        <label for="">密码</label>
        <input type="password" name="password" class="form-control" id=""
            placeholder="密码">
    </div>
</form>
```



```

</div>
<div>{$error_tips|default=''}</div>
<button type="submit" class="btn btn-default">用户登录</button>
</form>

```

完成后，使用超级管理员账号登录，可以看到页面头部显示了用户信息和商品发布入口，如图 12-21 所示。



图 12-21 用户登录成功后显示信息

### 3. 用户注销

修改 User.php 控制器文件，增加 logout() 方法，实现注销功能，代码如下：

```

/**
 * 注销登录
 */
public function logout()
{
    // 销毁会话登录信息
    session('USER_ID' ,null);
    session('USER_NAME' ,null);
    $this->redirect('index/index');
}

```

完成这一步，就实现了简单的用户注册、登录和注销功能。

## 12.3.4 商品发布

商品发布需要记录商品的名称、价格和库存，其中库存是判断用户是否还可以抢购的重要依据。另外，为了美观性，提供了商品图片上传的功能。

在 application\index\view\index 目录中，新增 add.html 模板文件，增加商品发布的表单代码如下：

```

<form action="{:url('add')}" method="post" enctype="multipart/form-
data" >
    <div class="form-group">
        <label>商品名称</label>
        <input type="text" name="name" value="" class="form-control"
placeholder="文章标题">
    </div>
    <div class="form-group">
        <label>商品价格</label>
        <input type="text" name="sell_price" value="" class="form-control"
placeholder="0.00">

```

```

</div>
<div class="form-group">
  <label>商品库存</label>
  <input type="text" name="stock" value="" class="form-control"
    placeholder="库存不能少于 0">
</div>
<div class="form-group">
  <label>商品图片</label>
  <input type='file' name='image'>
</div>
<div class="form-group">
  <label>文章详情</label>
  <textarea id="desc" name="desc" style="width:100%; height:200px;
"></textarea>
</div>
<div>{$error_tips|default=''}</div>
<button type="submit" class="btn btn-default">发布商品</button>
</form>

```

需要注意两点，这里使用了原始的表单上传，所以表单头需要带如下代码：

```
enctype="multipart/form-data"
```

另外，代码中还使用了第三方的富文本编辑器 ckeditor，除了定义带有 ID 属性的 textarea 标签，还需要引入相应的类库和代码，其代码如下：

```

{block name="script"}
  <script src="__STATIC__/_index/libs/ckeditor/ckeditor.js"></script>
  <script>
    CKEDITOR.replace('desc');
  </script>
{/block}

```

修改 application\index\controller\index.php 控制器文件，增加 add()方法，代码如下：

```

/**
 * 发布商品
 * @return mixed
 */
public function add()
{
  // 权限限制，只有超级管理员才可以发布商品
  if(session('USER_ID') != intval(config('USER_ADMIN')))
  {
    $this->error('只有管理员才可以发布商品');
  }
  // 判断请求类型
  if($this->request->isPost())
  {
    // 获取请求的商品参数
    $data = $this->request->param();
    // 文件上传
    $image = $this->upload();
    // 基本的非空验证

```

```

        if(!$data['name'] || !$data['sell_price'] || !$data['stock'])
        {
            $this->assign('error_tips' , '字段不能为空');
        }
        elseif(!$image)
        {
            $this->assign('error_tips' , '请上传缩略图');
        }
        else
        {
            // 写入数据到数据库中
            $data['image'] = $image;
            $data['status'] = 1;
            $data['create_time'] = time();
            $data['user_id'] = session('USER_ID');
            $insertId = Db::name('goods')->insert($data);
            if($insertId)
            {
                $this->redirect('index/index');
            }
        }
    }
    return $this->fetch();
}

```

除了菜单入口做了限制，在 `add()` 方法中也做了非管理员不能发布商品的限制，代码如下：

```

// 权限限制，只有超级管理员才可以发布商品
if(session('USER_ID') != intval(config('USER_ADMIN')))
{
    $this->error('只有管理员才可以发布商品');
}

```

同时使用了 ThinkPHP 5 内置的文件上传类库上传商品图片，文件会被上传到根目录下的 `public/uploads` 目录下。代码如下：

```

// 文件上传
$image = $this->upload();
.....
/**
 * 文件上传
 * @return string
 */
public function upload(){
    // 获取表单上传文件，例如上传了 001.jpg
    $file = request()->file('image');
    // 移动到框架应用根目录/public/uploads/ 下
    if($file){
        $info = $file->move(ROOT_PATH . 'public' . DS . 'uploads');
        if($info){

```

```

        return '/uploads/' . $info->getSaveName();
    }else{
        // 上传失败获取错误信息
        return $file->getError();
    }
}
}
}

```

完成开发后，尝试使用商品发布功能，完成后的效果如图 12-22 所示。



图 12-22 发布抢购商品

### 12.3.5 商品详情页

在首页商品列表中，单击“马上购买”按钮，可以进入商品详情页。修改 `Index.php` 控制器文件，增加 `detail()` 方法，实现商品的查询和模板渲染，其代码如下：

```

/**
 * 商品详情
 * @return mixed
 */
public function detail()
{
    // 根据商品 ID 查询详情
    $map['id'] = input('id');
    $info = Db::name('goods')->where($map)->find();
    if(!$info)

```

```
{
    $this->error('参数错误! ');
}
// 商品信息变量置换
$this->assign('info' , $info);
return $this->fetch();
}
```

在 application\index\view\index 目录下，增加 detail.html 模板文件，增加以下代码：

```
<div class="row">
    <div class="col-md-2"></div>
    <div class="col-md-8">
        <div class="page-header">
            <h2>{$info.name}</h2>
        </div>
        <div>
            <div class="row">
                <div class="col-md-6">
                    <div>
                        
                    </div>
                </div>
                <div class="col-md-6" >
                    <div style="padding:20px;font-size:20px;">售价: {$info.
                    sell_price}</div>
                    <div style="padding:20px;font-size:16px;">库存: <span
                    id="goods_stock">{$info.stock}</span></div>
                    <div style="padding:20px;font-size:12px;">
                        <p>提示: 每个用户限购一件</p>
                    </div>
                    <div style="padding:20px;">
                        <a class="btn btn-danger" id="buynow">马上抢购</a>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <div class="page-header">
        <h4>商品详情</h4>
    </div>
    <div>{$info.desc}</div>
</div>
<div class="col-md-2"></div>
</div>
```

完成以上开发工作后，访问商品详情页，效果如图 12-23 所示。

提示：商城抢购系统的架构较为简单，开发者可以根据需求进行扩充。



图 12-23 查看商品详情页

## 12.4 完善商城抢购系统逻辑

本节将最终完成商城抢购系统的开发, 该系统在实现过程中不仅需要各种基础技术的支持, 在使用前还需要进行部署工作。

### 12.4.1 安装使用 PHP Redis 扩展

在实现最终的商城抢购系统前还需要做一些准备工作, 需要给 PHP 安装 Redis 扩展, 以方便在 PHP 中对 Redis 的各种数据进行操作。下面以 Vagrant 虚拟机中的 Ubuntu 14.04 版本为例, 演示如何安装 Redis 扩展。

#### 1. PHP Redis 扩展的安装

虽然可以使用编译的方式给 PHP 安装 Redis 扩展, 但是 PHP 本身使用 apt 包管理器安装, 所以这里也用相同的方式安装 Redis 扩展。

##### (1) 使用 apt 包安装 PHP Redis 扩展

首先使用以下命令查看 PHP 是否已经安装:

```
php -m
```

结果如下:

```
PHP 7.0.22-0ubuntu0.16.04.1 (cli) ( NTS )
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies
```

查看 Redis 服务是否已经安装，命令如下：

```
redis-server -v
```

显示结果如下，说明已经安装 Redis 服务。

```
Redis server v=3.0.6 sha=00000000:0 malloc=jemalloc-3.6.0 bits=64
build=687a2a319020fa42
```

随后执行安装命令：

```
sudo apt-get install php-redis
```

(2) 验证扩展是否已经安装

执行完毕后重启 Nginx，执行如下命令：

```
nginx -s reload
```

在浏览器中访问 `phpinfo.php` 文件，效果如图 12-24 所示，说明 PHP Redis 扩展已经安装成功。

redis	
Redis Support	enabled
Redis Version	2.2.8-devphp7

图 12-24 查看 PHP Redis 扩展信息

## 2. 使用 PHP Redis 扩展的实例

开启 Redis 服务并安装 PHP Redis 扩展后，就可以在 PHP 中进行相应的 Redis 数据操作了。常见操作说明如下。

(1) 实例化 Redis 对象

在服务器根目录新增 `redis.php` 实例脚本文件，新增代码如下：

```
<?php
$redis = new redis(); // 实例化 Redis 对象
$result = $redis->connect('127.0.0.1', 6379); // 建立连接
var_dump($result);
```

在浏览器中访问后，返回 `true` 则说明连接成功，例如：

```
D:\phpStudy\WWW\demo12\12.4.1\redis.php:4:boolean true
```

其中，Redis 对象的 `connect()` 方法的第一个参数是 redis 服务的连接地址，第二个参数为当前服务端口。

(2) 赋值 (`set`) 与取值 (`get`) 操作

继续修改 `redis.php` 脚本文件，增加以下代码：

```
$redis->set('write_time' , date('Y-m-d H:i:s')); // 记录写入时间
var_dump($redis->get('write_time')); // 读取写入时间
```

在浏览器中执行后, 结果如下:

```
D:\phpStudy\WWW\demo12\12.4.1\redis.php:6:string '2018-01-12
02:21:17' (length=19)
```

使用 `set` 和 `get` 命令的方法, 和使用原生 Redis 命令的用法保持一致。

### (3) 散列操作 (Hash)

继续修改 `redis.php` 脚本文件, 增加以下代码:

```
// Hash 散列表操作
$redis->hset('user_info' , 'name' , 'wangjialin');
$redis->hset('user_info' , 'age' , 18);
$redis->hset('user_info' , 'mobile' , '13511112222');
$user_info = $redis->hgetall('user_info');
var_dump($user_info);
```

在浏览器中访问, 返回结果如下:

```
array (size=3)
  'name' => string 'wangjialin' (length=10)
  'age' => string '18' (length=2)
  'mobile' => string '13511112222' (length=11)
```

通过实例演示可以发现, 在 PHP 中操作 Redis 的方法基本与原生 Redis 命令类似, 这里不再过多演示。

## 12.4.2 使用 PHP 实现 Redis 订阅/发布实例

在上述实例中, 已经可以使用 Redis 原生的命令实现消息发布/订阅模式, 本节尝试在 PHP 中实现相同的操作。

### 1. 实现消息订阅脚本

在根目录下新增 `sub.php` 脚本文件, 借助 PHP Redis 扩展的方法来实现, 代码如下:

```
<?php
header('Content-type:text/html;charset=utf-8');
//避免在默认的配置下 1 分钟后终止与 Redis 服务器的连接
ini_set('default_socket_timeout', -1);
// 实例化 Redis 类, 监听订阅信息
$redis = new Redis();
$redis->connect('127.0.0.1',6379);
// 订阅名为 task_queue 的频道
$redis->subscribe(['task_queue'], function($redis,$chan,$msg){
    switch ($chan) {
        case 'task_queue':
            echo $msg ."\t\n"; //在命令行中显示接收到的信息
            break;
        default:
```

```
        break;
    }
});
```

因为需要一直监听 Redis 中的频道动态，sub.php 需要持续执行，所以不能使用浏览器访问，同时为了防止与 Redis 连接超时，定义脚本的命令如下：

```
//避免在默认的配置下 1 分钟后终止与 Redis 服务器的连接
ini_set('default_socket_timeout', -1);
```

Redis 扩展中的 subscribe() 方法传入了两个参数。第一个参数是需要订阅的频道名，可以是多个，例如数组类型；第二个参数是一个回调方法，其包含 3 个参数，分别是 Redis 对象 (\$redis)、频道名 (\$chan) 和消息 (\$msg)。在 subscribe() 方法体内对不同频道进行筛分执行不同的工作，代码如下：

```
case 'task_queue':
    // 获取消息主体内容，然后执行相应的任务
    break;
```

## 2. 实现消息发布功能

在 sub.php 脚本的同级目录下，新增 pub.php 脚本用来给特定频道发布消息。sub.php 脚本实现队列消费的功能，而 pub.php 脚本则模拟实现入队列的操作。实现代码如下：

```
<?php
header('Content-type:text/html;charset=utf-8');
// 发布订阅消息
$redis = new Redis();
// 第一个参数为 Redis 服务器的 IP，第二个为端口
$res = $redis->connect('127.0.0.1', 6379);
// test 为发布的频道名称，hello, world 为发布的消息
$res = $redis->publish('task_queue', ('当前时间: '.date('Y-m-d H:i:s')));
if (!$res)
{
    echo "发布订阅失败! ";
    die;
}
echo "发布订阅成功! ";
die;
```

相比消息订阅方法，publish() 方法的使用较为简单，只需要传入频道名称和消息这两个参数即可：

```
$res = $redis->publish('task_queue', ('当前时间: '.date('Y-m-d H:i:s')));
```

每次执行 pub.php 脚本，就可以向 task\_queue 频道发送最新的时间信息。为了测试脚本是否正常，在命令行中执行以下命令：

```
php sub.php
```

随后在浏览器中访问 pub.php 脚本文件，返回结果如下：

```
发布订阅成功!
```

再查看执行 `sub.php` 脚本的命令行，显示结果如下：

```
$ php sub.php
当前时间: 2018-01-12 06:30:13
当前时间: 2018-01-12 06:30:15
当前时间: 2018-01-12 06:30:16
当前时间: 2018-01-12 06:30:18
当前时间: 2018-01-12 06:30:18
```

### 12.4.3 实现抢购队列消费脚本——订单处理

实现商城抢购系统，需要先实现订单处理脚本，即负责监听抢购队列中的抢购信息，一旦订阅成功，就会根据抢购信息（商品 id+用户 id）来进行订单操作。当然，对不符合要求的用户信息，如已经抢购过商品的用户等，则不会进入订单相关逻辑。下面是相关的介绍。

#### 1. PHP脚本常驻进程

虽然可以使用 PHP Redis 扩展来实现消息发布/订阅模式，但关闭命令行工具后，PHP 脚本进程也就自动终止了，所以首先要解决的问题就是：使 PHP 订单创建脚本常驻进程。这里以 Linux 操作系统为基础，实现思路如下。

在命令行中，使用 `setsid` 命令和“&”操作符，让 PHP 脚本转入后台运行。以 `sub.php` 脚本为例，只需要在命令行中执行以下命令：

```
setsid /usr/bin/php7.0 /var/www/sub.php >/dev/null &
```

执行后会令程序转入后台执行，结果如下：

```
root@scotchbox:/var/www# setsid /usr/bin/php /var/www/sub.php >/dev/null &
[1] 2495
```

使用 Linux 系统内置的 `ps` 命令，查看 `sub.php` 脚本（进程）是否还在运行，命令如下：

```
ps aux|grep php
```

执行结果如下，发现 `sub.php` 的进程还在运行：

```
root@scotchbox:/var/www# ps aux|grep php
root      1010  0.0  1.4 475644 30408 ?        Ss   14:02   0:00 php-fpm:
master process (/etc/php/7.0/fpm/php-fpm.conf)
www-data  1130  0.0  0.4 475644  8996 ?        S    14:02   0:00 php-fpm:
pool www
root      2547  0.5  1.0 401256 21604 ?        Ss   14:55   0:00 /usr/bin/php
/var/www/sub.php
root      2549  0.0  0.0  12848  1012 pts/0    S+   14:55   0:00 grep
--color=auto php
```

这样就实现了 PHP 脚本的常驻进程，若想关闭 `sub.php` 进程，只需要使用“kill 进程 ID”命令终止即可，例如：

```
kill 2547
```

## 2. 判断脚本是否运行

使用 PHP 脚本常驻进程执行某些任务，可能会因为潜在的数据库错误、内存溢出等导致进程终止。为了防止这类情况的发生，编写以下 shell 脚本，定期判断进程是否存在，若不存在则主动启动脚本。在系统中定义 `alive.sh` 文件并定义脚本内容如下：

```
#!/bin/bash
alive=`ps -aux|grep /var/www/taskworker.php|grep -v grep|wc -l`
if [ $alive -eq 0 ]
then
    setsid /usr/bin/php7.0 /var/www/sub.php >/dev/null &
fi
```

上述脚本中先通过 `ps` 命令判断脚本是否还在运行，命令如下：

```
alive=`ps -aux|grep /var/www/sub.php|grep -v grep|wc -l`
```

此命令执行后，若进程存在则返回 1，反之返回 0，单独在命令中执行结果如下：

```
root@scotchbox:/var/www# ps -aux|grep /var/www/sub.php|grep -v grep|wc -l
0
```

给予 `alive.sh` 脚本执行的权限，命令如下：

```
chmod 700 alive.sh
```

随后执行如下命令：

```
./alive.sh
```

此时再执行命令查看 `sub.php` 脚本，发现该脚本已经在后台运行，执行结果如下：

```
root@scotchbox:/var/www# ps -aux|grep /var/www/sub.php|grep -v grep|wc -l
1
```

最后可以设置每分钟检测一次脚本是否在运行，只需在 `Crontab` 中增加以下时间表达式即可：

```
* * * * * /var/www/alive.sh
```

## 3. 订单处理脚本

继续在 `wangcmf` 实例项目下的 `extend` 目录下新增 `Task/task.php` 脚本文件，用来订阅 Redis 的频道消息，同时根据消息类型定义订单处理逻辑，实现步骤如下。

### (1) 实现消息订阅逻辑

修改 `task.php` 文件，增加以下代码：

```
<?php
// 防止监听过程中断开与 redis 的连接
ini_set('default_socket_timeout', -1);
//大于 100MB 内存则退出程序，防止内存泄漏被系统杀死导致任务异常
if(memory_get_usage() > 100*1024*1024){
    exit(0);
}
// 实例化 Redis 类，监听订阅信息
```

```

$redis = new \Redis();
$redis->connect('127.0.0.1',6379);
$redis->subscribe(['task_queue'], function($redis,$chan,$msg){
    switch ($chan) {
        case 'task_queue':
            // todo::
            break;
        default:
            break;
    }
});

```

其中为了防止内存溢出，专门增加了内存大小判断逻辑，代码如下：

```

//大于 100MB 内存则退出程序，防止内存泄漏被系统杀死导致任务异常
if(memory_get_usage() > 100*1024*1024){
    exit(0);
}

```

### (2) 验证用户抢购商品信息

虽然是现行开发订单的处理逻辑，但是订单需要根据用户发送的抢购信息进行处理。这里预先定义抢购队列中的消息类型，为 PHP 中的数组类型，例如：

```

[
    'user_id' =>1,
    'goods_id' =>2
]

```

但是 Redis 不支持 PHP 数组格式，所以需要使用 `serialize()` 方法进行处理，在消息订阅端就需要进行反序列化处理，同时进行格式的验证，继续修改 `task.php` 脚本，增加以下代码：

```

switch ($chan) {
    case 'task_queue':
        // 反序列化消息数据
        $task = unserialize($msg);
        // 参数判断
        if(!$task['user_id'] || !$task['goods_id'])
        {
            echo "参数错误\r\n";
            return false;
        }
        .....
}

```

### (3) 连接数据库

为了提高运行效率，减少 PHP 脚本引入的规模，`task.php` 脚本基于原生 PHP 脚本开发，所以这里使用 PDO 扩展连接数据库进行相关操作，在验证完消息数据后继续添加以下代码：

```

// 创建 PDO 数据库连接
try {
    $pdo = new PDO('mysql:host=127.0.0.1;dbname=wangcmf', 'root', '');
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo "Connect Failed: " . $e->getMessage();
}

```

```
    return false;
}
```

#### (4) 开始事务操作并进行订单处理

订单处理包含了新增订单数据、更新商品表库存等操作，数据库提交的时候需要开启事务。代码如下：

```
try {
    $pdo->beginTransaction();                // 开启一个事务
    // 查询商品是否还有库存
    $sql = "select * from db_goods where id = {$task['goods_id']} and stock
    > 0 and status = 1";
    $goods_info = [];
    foreach ($pdo->query($sql) as $item)
    {
        $goods_info = $item;
    }
    if(empty($goods_info))
    {
        echo "商品没有库存，或者已经下架\r\n";
        $pdo->rollback();
        return false;
    }
    // 查询是否已经下过单
    $sql = "select * from db_orders where user_id={$task['user_id']} and
    goods_id = {$task['goods_id']} and status = 1 and is_pay = 1";
    $result = [];
    foreach ($pdo->query($sql) as $item)
    {
        $result[] = $item;
    }
    if(!empty($result))
    {
        echo "请勿重新下单\r\n";
        $pdo->rollback();
        return false;
    }
    // 创建订单（模拟支付成功）
    $order_no = date('YmdHis').mt_rand(10000,99999);
    $pay_price = $goods_info['sell_price'];
    $is_pay = 1;
    $pay_time = time();
    $status = 1;
    $create_time = time();
    $update_time = 0;
    $sql = "insert into db_orders values(null, '{$task['goods_id']}' ,
    '{$task['user_id']}' , '{$order_no}', {$pay_price}, {$is_pay},
    {$pay_time}, {$status}, {$create_time}, {$update_time})";
    $result = $pdo->exec($sql);
    if(!$result)
    {
        echo "下单失败\r\n";
        $pdo->rollback();
        return false;
    }
}
```

```

    }
    // 减少库存
    $stock = ( $goods_info['stock'] - 1 ) >= 0 ? $goods_info['stock'] - 1 : 0;
    $sql = "update db_goods set stock = {$stock} where id = {$goods_
    info['id']}";
    $result = $pdo->exec($sql);
    if(!$result)
    {
        echo "更新库存失败\r\n";
        $pdo->rollback();
        return false;
    }
    echo "下单成功";
    $pdo->commit();
} catch (PDOException $e) {
    $pdo->rollback(); // 执行失败，事务回滚
    exit($e->getMessage());
}

```

完成以上开发后，在命令行中运行订单处理脚本，验证该脚本是否可以正常运行。

#### 12.4.4 实现抢购入队操作——抢购处理

抢购处理主要分为两个部分：第一部分是把用户抢购信息发送到特定的频道中，排队等待订单处理脚本进行消费处理；第二部分就是主动查询订单状态，查询用户是否抢购成功。

##### 1. 将抢购信息发布到接口

实现把用户抢购信息（商品 id\_用户 id）发送到频道中，这里新增一个控制器来完成处理，在 application\index\controller 目录下新增 Task.php 控制器文件，在其中增加 publish() 方法用来发布订阅消息，代码如下：

```

/*
 * 发布订阅消息模块
 */
class Task extends Controller
{
    // Redis 连接信息，可以放到配置方法 config() 中
    private $redis_host = '127.0.0.1';
    private $redis_port = 6379;
    // 频道 (task) 名称
    private $task_name = 'task_queue';
    /**
     * 发布商品抢购信息
     */
    public function publish()
    {
        if($this->request->isAjax())
        {

```



```
// 获取基本参数
$user_id = session('USER_ID');
$goods_id = $this->request->param('goods_id');
if(!$user_id || !$goods_id)
{
    $this->error('用户 id 或者商品 id 参数错误');
}
// 或者直接在数据库中查询是否已经下单支付
$redis = new \Redis();
$redis->connect($this->redis_host , $this->redis_port);
$is_buy = $redis->hGet('goods_'. $goods_id , $goods_id.'_'.
$user_id);
if($is_buy)
{
    $this->error('商品已经购买过');
}
// 把用户抢购商品信息 (goods_id_user_id) 写入 Redis 哈希列表中
$insert = $redis->hSet('goods_'. $goods_id , $goods_id.'_'.
$user_id , 0);
if($insert === false)
{
    $this->error('系统错误, 请稍后再试!');
}
// 构建发布订阅消息数据
$task = [
    'user_id' => $user_id,
    'goods_id' => $goods_id
];
// 发布定义消息 (序列化)
$publish = $redis->publish($this->task_name , serialize($task));
if(!$publish)
{
    $this->error('抢购失败, 请稍后再试!');
}
$this->success('正在抢购中, 请耐心等待!');
}
$this->error('参数错误!');
```

在上述方法中, 首先根据会话获取用户的信息和商品的信息, 注意用户 ID 并没有接收外部的数据, 而是直接获取会话, 提高了安全性, 命令如下:

```
// 获取基本参数
$user_id = session('USER_ID');
```

因为抢购规则是每个用户限购一件商品, 所以随后查询用户是否购买过当前商品, 这个记录保存在 Redis 哈希表中, 哈希表名为:

goods\_商品 id

存储的数据格式为:

商品 id\_用户 id

存储格式与抢购信息保持一致, 代码实现如下:





```
// 或者直接在数据库中查询是否已经下单支付
$sis_buy = $redis->hGet('goods_'. $goods_id , $goods_id.'_' . $user_id);
if($sis_buy)
{
    $this->error('商品已经购买过');
}
}
```

若用户从未抢购过当前商品，则写入哈希表中，命令如下：

```
$insert = $redis->hSet('goods_'. $goods_id , $goods_id.'_' . $user_id , 0);
```

除了记录用户是否已经抢购过此商品，还需要记录用户已经完成抢购的状态，以更好地区分哪些信息还在处理。当前状态为 0，说明用户只是发起了抢购动作，仍在等待处理。当处理完毕后，把此状态修改为 1，则说明当前用户信息出队列处理完毕，可以方便地跟踪用户的各种状态。

最后调用 Redis 的方法给特定频道发送抢购消息，命令如下：

```
// 发布定义消息（序列化）
$publish = $redis->publish($this->task_name , serialize($task));
```

## 2. 商品详情页抢购逻辑的实现

商品详情页作为用户抢购的主要操作界面，主要功能就是提供抢购入口并提供抢购反馈。在程序实现上主要包含以下 3 部分：

- 用户单击“马上抢购”按钮后，异步（AJAX）发送数据到发布订阅消息接口（Task/publish）。
- 发送完成后，主动查询订单处理的进度。
- 在页面上显示抢购的状态信息。

商品详情页抢购流程如图 12-25 所示。

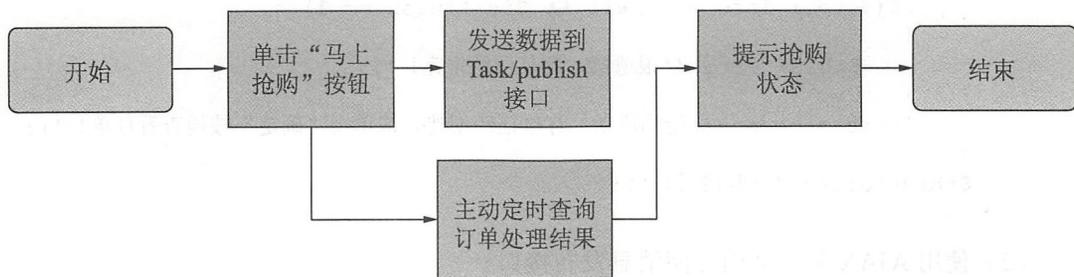


图 12-25 商品详情页抢购流程

实现以上几个逻辑的步骤如下。

### (1) 开发订单查询接口

修改 application\index\controller\task.php 文件，新增 getOrderStatus()方法，根据传入的商品 ID，获取当前用户是否已经完成订单操作。代码如下：





```

/**
 * 获取订单状态
 */
public function getOrderStatus()
{
    if($this->request->isAjax())
    {
        // 获取基本查询信息
        $user_id = session('USER_ID');
        $goods_id = $this->request->param('goods_id');
        if(!$user_id || !$goods_id)
        {
            $this->error('用户 id 或商品 id 错误! ');
        }
        // 构造查询条件
        $map = [
            'user_id' => $user_id,
            'goods_id' => $goods_id,
            'status' => 1
        ];
        // 查看队列是否已经处理完成
        $key = $goods_id . '_' . $user_id;
        $redis = new \Redis();
        $redis->connect($this->redis_host , $this->redis_port);
        $is_finish = $redis->hGet('goods_'.$goods_id , $key);
        // 队列正在处理, 提示用户耐心等待
        if($is_finish == 0)
        {
            $this->error('正在排队处理中, 请继续等待! ');
        }
        // 查询订单是否已经创建
        $order_info = Db::name('orders')->where($map)->find();
        // 订单信息为空并且队列已经处理完毕, 说明用户没有抢到订单
        if($order_info === null && $is_finish == 1)
        {
            $this->error('很遗憾, 商品已经售完! ');
        }
        $this->success('抢购成功, 订单已经创建, 请单击“确定”按钮查看订单! ');
    }
    $this->error('参数错误! ');
}

```

## (2) 使用 AJAX 异步调用订阅消息发布接口

继续修改 `application\index\view\index\detail.html` 模板文件, 增加以下 JavaScript 代码, 实现当用户单击“马上抢购”按钮后, 异步发送消息到抢购队列中。具体代码如下:

```

{block name="script"}
<script>
    $(function(){
        // 显示遮罩层
        function showModal(text , url = '')
        {
            // 判断是否有文本

```





```

if(text)
{
    $('#showInfoText').html(text);
    $('#showInfo').modal('toggle');
    if(url != '')
    {
        // 显示弹出层
        $('#showInfo').on('hidden.bs.modal', function (e) {
            window.location.href = url;
        });
    }
}
return false;
}
//马上抢购
$('#buynow').click(function(){
    var self = $(this);
    // 设置抢购按钮无法单击
    self.addClass('disabled');
    var user_id = ":{:session('USER_ID')}";
    // 判断用户是否已经登录了
    if(!user_id)
    {
        // 提示未登录
        showModal('登录后才可以购买', ":{:url('User/login')}");
        return false;
    }
    // 获取商品 ID
    var goods_id = ":{:info.id}";
    $.post(":{:url('Task/publish')}", {user_id:user_id , goods_id:goods_id, time:new Date().getTime()} , function(msg){
        if(msg.code == 1)
        {
            // 成功提示语
            var success_text = '<p>' + msg.msg + '</p><p>正在获取订单状态...</p><p>单击“确定”按钮，查看订单列表</p>';
            // 获取成功跳转的订单地址
            var success_url = ":{:url('Index/order_list')}";
            // 显示状态提示
            showModal(success_text ,success_url);
            setInterval(function(){
                $.post(":{:url('Task/getOrderStatus')}" , {user_id:user_id , goods_id:goods_id, time:new Date().getTime()} , function(data){
                    $('#showInfoText').html('<p>'+ data.msg + '</p>');
                    if(data.code == 1)
                    {
                        // 1 秒后跳转到订单列表
                        setTimeout(function(){
                            window.location.href = ":{:url('Index/order_list')}";
                        },1000);
                    }
                }
            )
        }
    }
}

```





```

        });
    },2000); //每秒钟查询订单的状态
    }
    else
    {
        // 提示错误信息
        var error_text = msg.msg;
        showModal(error_text);
        return false;
    }
    // 按钮可以操作
    self.removeClass('disabled');
    })
    })
    })
</script>
{/block}

```

上述代码中，当用户单击“马上抢购”按钮后，对用户的登录信息进行判断，如用户没有登录，则会弹出对话框提示错误信息。用户基础信息验证无误后，会发送用户和商品信息到抢购下单接口（Task/publish）。代码如下：

```

var goods_id = "${info.id}";
$.post("{:url('Task/publish')}" , {user_id:user_id , goods_id:goods_id,
time:new Date().getTime()} , function(msg){
.....

```

在回调方法中对结果进行处理，若发布抢购消息成功，则每过2秒钟就会主动查询抢购订单的状态，成功后提示并跳转到订单列表。代码如下：

```

// 显示状态提示
showModal(success_text ,success_url);
setInterval(function(){
$.post("{:url('Task/getOrderStatus')}" , {user_id:user_id , goods_id:
goods_id, time:new Date().getTime()} , function(data){
$.showInfoText().html('<p>'+ data.msg + '</p>');
if(data.code == 1)
{
// 1秒后跳转到订单列表
setTimeout(function(){
window.location.href = "{:url('Index/order_list')}";
},1000);
}
});
},2000); //每秒钟查询订单的状态

```

### (3) 订单列表

在 Index 控制器中新增 order\_list()方法，实现查询当前用户的订单列表数据，代码如下：

```

/**
 * 订单列表
 * @return mixed
 */

```





```

public function order_list()
{
    if(!session('USER_ID'))
    {
        $this->redirect(url('index/index'));
    }
    $map['user_id'] = session('USER_ID');
    $this->assign('_list' , Db::name('orders')->where($map)->order
    ('create_time desc')->select());
    return $this->fetch();
}

```

在 application\index\view\index 目录下新增 order\_list.html 模板文件，核心代码如下：

```

<ul class="list-group">
    {volist name="_list" id="vo"}
        <li class="list-group-item">
            订单号: {$vo.order_no},
            订单支付价格: {$vo.pay_price},
            下单时间:{$vo.create_time|date='Y-m-d H:i:s',###},
            <a href="{:url('index/detail' , array('id'=>$vo.goods_id))}">
                查看商品</a>
        </li>
    {/volist}
</ul>

```

订单列表只展示订单号、支付价格和下单时间，给用户提供一个查看购买商品的入口即可。实际项目中的订单列表远比本实例复杂，这里的订单列表只做演示使用。

## 12.4.5 抢购系统部署与使用

完成抢购模块的系统开发后，就可以调试、部署和使用该系统了。和一般的网站不同，在正式使用前需要保障以下服务的开启：

- Redis 服务端；
  - 项目目录/extend/Task/task.php 文件在命令行中保持运行。
- 上面两个服务缺一不可，否则会导致抢购流程无法完成。随后可以查看抢购的使用流程。

### 1. 用户登录

因为对用户未做进一步的限制，发布商品的用户也可以购买自己的商品，所以这里使用超级管理员登录商城抢购系统。

### 2. 发布抢购商品

发布一个抢购商品，内容可以根据需求来填写，但是为了验证订单是否超发，库存设置为 1。抢购商品发布完成后，如图 12-26 所示。





图 12-26 发布新的抢购商品

### 3. 管理员账号抢购商品

进入商品详情页后，单击“马上抢购”按钮后，弹出对话框并提示抢购的进度和状态，如图 12-27 所示。

抢购成功后，会提示订单创建成功，如图 12-28 所示。

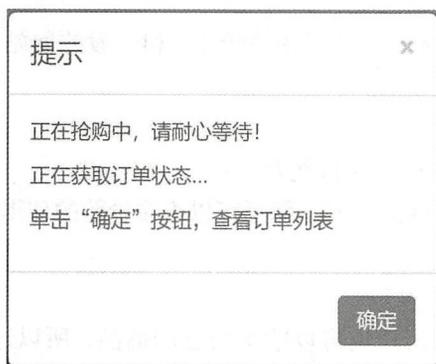


图 12-27 等待抢购处理状态展示

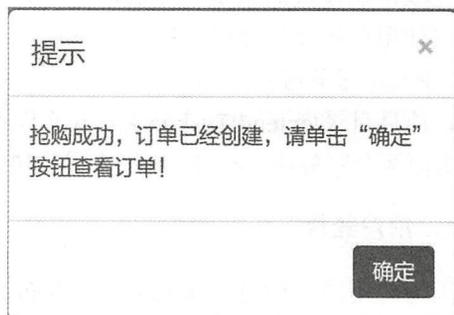


图 12-28 下单成功状态展示

出现下单成功的提示后，页面等待 1 秒后跳转到订单列表，如图 12-29 所示。

此时再回到商品详情页，发现库存已经为 0，若再次单击“马上抢购”按钮，系统则会提示已经抢购过此商品，如图 12-30 所示。





## 订单列表

订单号: 2018011701051275249, 订单支付价格: 100.00, 下单时间:2018-01-17 09:05:12, 查看商品
订单号: 2018011700505850818, 订单支付价格: 2399.00, 下单时间:2018-01-17 08:50:58, 查看商品
订单号: 2018010902093196926, 订单支付价格: 100.00, 下单时间:2018-01-09 10:09:31, 查看商品
订单号: 2018010506571142712, 订单支付价格: 99.00, 下单时间:2018-01-05 14:57:11, 查看商品
订单号: 2018010506442971124, 订单支付价格: 10.00, 下单时间:2018-01-05 14:44:29, 查看商品

图 12-29 抢购成功后重定向到订单列表

#### 4. 验证订单是否超发

为了验证 0 库存的商品是否还可以抢购, 新注册普通用户并登录, 然后继续抢购此商品, 此时会提示商品已经售完, 如图 12-31 所示。

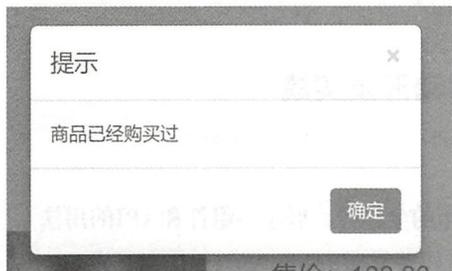


图 12-30 重复购买会提示商品已经购买过

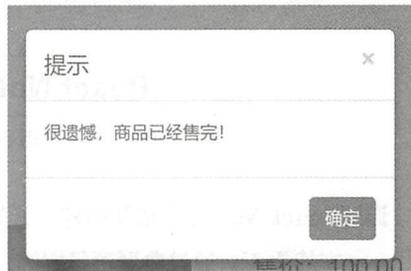


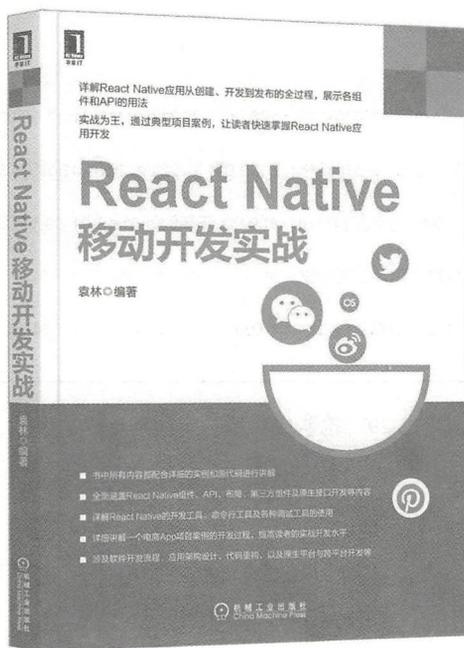
图 12-31 提示商品已售完

以上步骤就是简单的商城抢购系统流程, 作为一个较为基础的系统模型, 还有很多可以完善的地方, 开发者可以根据实际需求进行相应的修改。





## 推荐阅读



### React Native移动开发实战

作者：袁林 书号：978-7-111-57179-7 定价：69.00元

**详解React Native应用从创建、开发到发布的全过程，展示各组件和API的用法  
实战为王，通过典型项目案例，让读者快速掌握React Native应用开发**

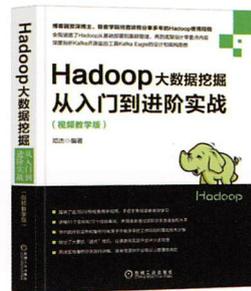
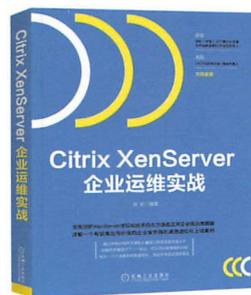
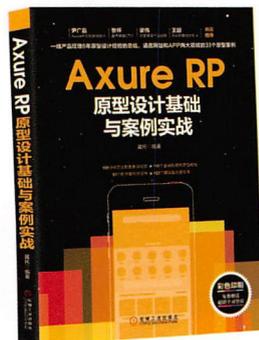
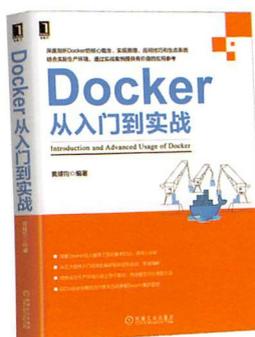
本书以实战开发为主旨，以React Native应用开发为主线，以iOS和Android双平台开发为副线，通过完整的电商类App项目案例，详细地介绍了React Native应用开发所涉及的知识，让读者全面、深入、透彻地理解React Native的主流开发方法，从而提升实战开发水平和项目开发能力。

本书共12章，分为4篇，涵盖的主要内容有搭建开发环境、Nuclide、各种命令行工具（Git、Node.js）、布局与调试、组件、API、第三方组件、基于Node.js的服务器、fetch API、AsyncStorage/SQLite/Realm数据库存储、原生平台接口开发、redux开发框架、应用打包与发布、热更新与CodePush等。

本书适合iOS和Android原生平台应用开发者，以及有兴趣加入移动平台开发的JavaScript开发者阅读。当然，本书也适合相关院校和社会培训学校作为移动开发的教材使用。



## 推荐阅读



欢迎IT领域的各位技术人员洽谈出书事宜。如果您有写书或投稿意向，请加QQ或微信与编辑具体商谈。

QQ: 627173439

微信: oyzx\_sp



# PHP高性能开发 基础、框架与项目实战

## 本书精华内容

- 编程语言与操作系统的选择
- 虚拟机与个性化高效开发环境搭建
- 更先进的版本管理工具——Git
- 使用GitHub Pages搭建个人博客站点实战案例
- 高效团队协作工具——ProcessOn与GitLab
- GitLab持续集成与自动构建实战案例
- 好用的PHP开发环境——PHPStorm
- PHP依赖的自动化管理工具——Composer
- 在项目中集成依赖包PHPMailer实战案例
- 响应式布局框架——Bootstrap
- 常用模板实战案例——列表
- 常用模板实战案例——表单
- ThinkPHP命令行操作与接口开发实战
- 制作一个短地址生成器实战案例
- RESTful API实战案例——用户接口权限验证
- 内容管理框架实战案例——基础架构、用户与菜单管理
- 内容管理框架实战案例——配置和权限管理
- Crontab计划任务管理模块实战案例
- 基于Redis队列的商城抢购系统项目案例

## 本书配套资源获取方式

本书涉及的源文件等资源需要读者自行下载。请在[www.hzbook.com](http://www.hzbook.com)网站上搜索到本书，然后单击“资料下载”按钮进入本书页面，再单击页面右上角的“配书资源”链接即可下载。

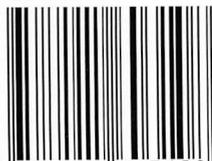
投稿热线: (010) 88379604  
客服热线: (010) 88379426 88361066  
购书热线: (010) 68326294 88379649 68995259

华章网站: [www.hzbook.com](http://www.hzbook.com)  
网上购书: [www.china-pub.com](http://www.china-pub.com)  
数字阅读: [www.hzmedia.com.cn](http://www.hzmedia.com.cn)



上架指导: 计算机/PHP

ISBN 978-7-111-60310-8



9 787111 603108 >

定价: 89.00元