

## 摘 要

在我国，随着互联网电商行业的不断发展，目前已经形成了阿里旗下的淘宝与腾讯战略合作的京东、拼多多等几大巨头并存的局面，各大互联网公司利用各自的平台优势不断地挖掘潜在的用户群体，这也使得获取用户流量变得更加困难，为此所付出的宣传和研发成本更是居高不下。

本设计依托于微信这款国民聊天工具，通过依托微信强大的整合能力，将微信处于社交个体的人，转变成商家、客户、顾客等角色。本设计使用 PHP 中的 TP5 框架编写服务器接口，包括商品、类型、用户身份验证、用户地址管理、订单、支付等 restful 接口，其中订单和支付接口调用微信提供的 SDK。通过 PHP 的 CodeIgniter 编写后台网站，实现对商品和订单的管理。

本项目力主于更快的获取用户流量，更便捷的搭建体系架构。PHP 入门简单，TP5 框架是国人开发，学习更为方便；小程序借助于微信平台开发，有着广泛的用户基础，二者的结合势必会促进现代电商行业的进一步发展，同时也会为用户带来更好的购物体验。

**关键词：**Thinkphp5；Restful 接口；数据库设计

## ABSTRACT

With the continuous development of the Internet e-commerce industry in China, the situation of Taobao, Jingdong and Pingduo coexisting has been formed. Major Internet companies use their platform advantages to tap potential user groups constantly, which makes it more difficult to obtain user traffic, and the cost of publicity and research and development is still high.

This design relies on Wechat as a national chat tool. Through the powerful integration ability of Wechat in small programs, Wechat will be transformed into the roles of business, customers. This design uses TP5 framework in PHP to compile server interface, including restful interface such as commodity, type, user authentication, user address management, order, payment and so on. The order and payment interface calls the SDK provided by Wechat. Through PHP CodeIgniter to write the website backend, to achieve the management of goods and orders.

This project focuses on faster access to user traffic, more convenient to build the architecture. The introduction of PHP is simple, TP5 framework is developed by Chinese people, and learning is more convenient; small programs with the help of Wechat platform development, has a broad user base, the combination of the two will certainly promote the further development of modern e-commerce industry, but also will bring better shopping experience for users.

**Keywords:** Thinkphp5, Restful API, Database Design

# 目 录

摘 要.....	I
ABSTRACT.....	II
目 录.....	III
第 1 章 绪 论.....	1
1.1 研究背景.....	1
1.2 国内外研究现状.....	2
1.3 设计目的和内容.....	3
1.4 研究方案对比.....	5
1.5 论文安排.....	6
第 2 章 系统方案设计.....	7
2.1 系统需求分析.....	7
2.2 研究使用的平台及技术.....	9
2.3 数据库设计分析.....	10
2.4 服务端接口设计分析.....	11
2.5 小程序端页面设计分析.....	15
2.6 后台网站架构设计分析.....	18
2.7 系统总体架构.....	19
第 3 章 服务器端设计.....	21
3.1 数据库设计.....	21
3.2 商品接口设计.....	23
3.3 分类接口设计.....	27
3.4 用户 Token 接口设计.....	27
3.5 用户地址接口设计.....	31
3.6 订单接口设计.....	34
3.7 支付接口设计.....	38
第 4 章 小程序端和后台网站设计.....	41

4.1 小程序页面设计.....	41
4.2 购物车实现.....	50
4.3 其他功能实现.....	52
4.4 后台网站设计.....	53
<b>第 5 章 系统接口调试.....</b>	<b>62</b>
5.1 Postman 调试服务端接口.....	62
5.2 完善小程序端显示界面.....	66
5.3 测试总结.....	68
<b>第 6 章 结论与展望.....</b>	<b>69</b>
6.1 结论.....	69
6.2 展望.....	70
<b>参考文献.....</b>	<b>71</b>
<b>致 谢.....</b>	<b>73</b>
<b>附录I 服务器接口源代码.....</b>	<b>74</b>
<b>附录 II 小程序源代码.....</b>	<b>87</b>
<b>附录 III 后台网站源代码.....</b>	<b>94</b>
<b>附录 IV 外文文献及翻译.....</b>	<b>108</b>

# 第 1 章 绪 论

## 1.1 研究背景

经过多年发展，中国的电子商务取得了显著规模效益，交易额连创新高，各大电子商务平台企业纷纷开始构建生态系统，如淘宝、京东等平台为商家和消费者提供支付、交易、物流等各方面全周期支持与服务，商家对于平台的依赖度也是越来越高。

2018 年我国网络零售市场规模不断扩大。全国网上零售额已经突破 9 万亿元，其中实体商品所占的线上零售额更是高达 7 万亿元，比去年同期增长 25.4%，对社会消费品零售总额增长的贡献率达 45.2%，较上年提升 7.3 个百分点。在规模增长的同时，也进一步加快了新旧动能转换，通过线上线下融合、业态模式创新、质量服务提升等一系列措施，加速了新动能的形成<sup>[1]</sup>。

但在这繁荣的外表之下一场又一场关于用户流量和市场的争夺战正在不断打响。根据拼多多发布的财报看出，从 17 年至 18 年第三季度的一年中，拼多多业绩同比增长 386%。而在当时，整个电商行业的增速也仅有 30%，拼多多以如此高速增长显然使得淘宝感到了极大威胁。虽然阿里在 BAT 等巨头中业绩增长属于最高的，但投资者依然在 18 年三季度选择大幅减持高达九成的阿里巴巴仓位，京东仓位保持不变，减持的资本多投了拼多多。以至在拼多多上市前后，阿里巴巴在各个领域都做出了应对，不论是网络销售还是平台公关，双方的争斗异常激烈。一时间，社交电商的战场上硝烟四起<sup>[2]</sup>。

不管怎么说，电商领域的竞争归根结底都是对人的竞争，有的公司依托自身平台或者是合作平台的广大用户基础可以更好的拓宽自己的市场

规模，而一些新进入电商领域、不依靠任何平台的企业只能步履维艰的花费巨大的精力、物力、财力去开辟自己的市场，挖掘潜在的用户群体。总之，不管这些巨头的产品如何变化，最终考验的都是平台的整体实力，人在哪儿，战场就在哪儿；用户在哪，市场就在哪儿。

## 1.2 国内外研究现状

随着传统企业不断地向电商领域进行转型升级，越来越多的国内外专家学者开始对电商这种新产生的商业模式进行深入研究。

根据国际知名调查公司 E-marketer 的数据，2011 年到 2016 年，全球网络零售交易额从 0.86 万亿美元增长至 1.92 万亿美元，年平均增长率达 17.4%。未来五年，随着全球智能手机保有量不断提升、互联网使用率持续提高、新兴市场快速崛起，全球网络零售仍将保持两位数增长。预计 2020 年，全球网络零售交易额将超过 4 万亿美元，占全球零售总额的比例从 2016 年的 7.4% 增长至 14.6%<sup>[3]</sup>。

跨境电子商务尤其是跨境 B2C（企业对个人）日益活跃。根据埃森哲的研究报告，2015-2020 年全球跨境 B2C 年均增速约 27%，2020 年市场规模将达到 9940 亿美元<sup>[4]</sup>。

在以移动互联网支撑的电商时代下，用户通过电商 APP 可以随时随地的进行网上交易，无法避免的产生了逐渐碎片化的用户访问量。但移动电商 APP 与微信小程序相比而言，需要面临开发攻坚，安卓 iOS 双平台同步，软件推广三大难题。因此当微信小程序的体验和原生 APP 的体验逐渐趋近的时候，微信小程序也就成为了首选开发对象。

目前，微信平台不仅仅是一个社交软件，其已经成为了新型信息化服务的重要载体，在国内多个行业中都有着非常出色的表现。目前基于微信

公众平台开发成功的应用软件已有很多，且大多服务于商业化。在这期间微信通过与其他行业的共同探索，创造了基于微信平台之上的巨大价值。在微信公众平台的应用研究已趋向成熟的前提下，应用与微信平台的服务也将会越来越多<sup>[5]</sup>。

从用户行为、生活习惯来讲，社交网络的流行，使得用户将大部分时间花在了社交软件上，由此可见，社交关系网在商业推广、营销中有着巨大的价值，将电商与社交进行结合是一个必然的趋势，因此基于微信小程序的电商产业拥有着广阔的发展空间。

## 1.3 设计目的和内容

### 1.3.1 研究目的

电商行业是中国互联网商业化起步最早的行业，也是市场最成熟竞争最激烈的行业。随着淘宝、京东等巨头的形成，用户流量获取更加困难，为获取客户所花费的成本更是十分高昂。

在 2018 年，微信每天发送出去 450 亿次消息，有 4 亿多语音、视频通话呼叫成功<sup>[6]</sup>。但是人们总是将微信视作一款社交、聊天工具，忽略了微信在商业领域的价值，尤其是在将商业营销与社交紧密联系在一起之后我们不难发现，微信中的用户不仅是社交软件的使用者那么简单，更是商户寻找的顾客，卖家寻找的买家，店家眼中的“上帝”。

尤其在移动互联网时代，作为一款有着如此多用户的应用平台，基于微信衍生出来的微信小程序<sup>[7]</sup>更是在各个领域发挥着重要的作用。在微信的影响力下，微信小程序已被政府机构、公司企业、组织个人在各行各业中广泛使用。通过微信实现的人与人之间的实时连接，微信小程序可以帮助人们探索、发掘新的商业模式、有效改善现有服务的弊端，更可以在帮

助企业拓宽营销渠道的同时使整个行业实现“互联网+”转型。

因此，基于 PHP 实现的微信商城有着清晰的设计目标和实际可行的应用背景，而且 PHP 入门简单，微信小程序有着广泛的用户基础，二者的结合势必会促进电子商城进一步的发展，同时也会为用户带来更好的购物体验。

### 1.3.2 研究内容

本项目为基于 Thinkphp5 的微信电子商城系统，整个系统按照模块化的思想进行设计，把系统分为三个模块，服务器端程序接口部分、微信小程序部分、浏览器端后台管理部分。第一部分为服务器端程序接口部分，采用 PHP 编写，使用国人开发的 Thinkphp5 框架<sup>[8]</sup>，接口包括商品、类型、用户身份验证、用户地址管理、订单、支付等 restful 接口，其中订单和支付接口调用微信提供的 SDK<sup>[9]</sup>，主要用于与小程序端进行数据交互；第二部分为小程序部分<sup>[10]</sup>，微信提供一个组件库，提供各种标签内部封装过的标签，但总体与编写网站前端类似，本质上就是 JS+CSS+HTML5。小程序页面用于和用户进行人机交互，将用户发出的请求传递给服务器端接口，展示服务端返回的数据等；第三部分为浏览器端后台管理部分，采用 PHP 语言编写，使用轻量级的 CodeIgniter<sup>[11]</sup> 框架，主要用于对商城中的分类、商品等信息进行增删改查以及上、下架处理，对订单和用户信息进行查看等。

根据论文研究内容所做安排如下：

（1）对项目涉及到的 PHP 的 Thinkphp5 和 CodeIgniter 框架、JS、Restful 接口规范进行深入学习并熟练运用。

（2）探讨论证基于 TP5 的微信电子商城系统的整体方案，以及数据库设计、各接口的设计等方案。



（3）通过微信开发者工具完成小程序端界面设计，通过 Postman 测试<sup>[12]</sup>完成服务端接口的设计。

（4）完成浏览器端商城后台的设计和编写，对系统整体进行调试和功能检测。

（5）对该系统作出评价和总结，并提出该系统将来可开辟的新功能和可发展的方向。

## 1.4 研究方案对比

### 1.4.1 服务端接口使用语言对比

#### （1）Java 语言

Java 作为一门被广泛使用的静态语言，因其组件多，使用线程池、连接池都较为方便，这对于高并发场景来说极为重要。而且 Java 开发的项目逻辑清晰，稳定可靠，运行效率高。但采用 Java 进行开发使用的是面向对象<sup>[13]</sup>的设计方法，PHP 则是半面向对象半面向过程，因此使用 Java 开发之前需要进行大量的系统分析和系统设计。因此 Java 开发学习成本较大，开发周期较长，后期维护也较复杂，所耗费的人力物力较多<sup>[14]</sup>。

#### （2）PHP 语言

PHP 作为一门动态的服务器脚本语言，使用灵活，上手较快，修改、发布较为容易，后期维护方便<sup>[15]</sup>。在访问数据库方面，PHP 对于不同的数据库采用不同的数据库访问接口，代码复用性较低；而 Java 通过 JDBC 来访问数据库，通过不同的数据库厂商提供的数据库驱动方便地访问数据库，访问数据库的接口比较统一。Java 开发的项目多适应于多层架构，高并发场景，性能较之 PHP 要高出很多<sup>[16]</sup>。

综上，我个人认为相对于中小型的应用系统开发，使用 PHP 有着快

速开发、成本较低、部署便捷的优点；而 Java 开发多适用于大型的应用系统。最后选择 PHP 语言作为本项目的服务端接口开发语言。

### 1.4.2 数据库对比

#### （1）Oracle

Oracle 作为一种重量级的对象关系数据库管理型数据库，收费但有着安全性高的特点，支持服务器扩展和各种类型的备份工具，分区分表较为成熟稳定，种种优点均可以提高用户使用数据库的体验。Oracle 适合与 Java 配合开发大型的金融和商务系统。

#### （2）MySQL

MySQL 是一种开源的、轻量的关系数据库管理系统，作为世界上使用最多的数据库管理系统，其有着使用便捷，部署简单等一系列方便快速开发的优点。如 LAMP、WAMP 等集成环境就是将 PHP 与 MySQL 组合进行中小型系统的快速开发。

综上，在选择了 PHP 作为服务端接口的开发语言后，选择 MySQL 作为数据库管理系统配合开发中小型应用系统是较为合适的。

## 1.5 论文安排

本论文第一章主要介绍项目背景，研究目的和意义，讨论方案论证；第二章简要的分析系统方案，对系统整体方案架构进行设计，方便后续开发；第三章详细的介绍数据库表的设计和服务端各接口具体实现流程；第四章详细分析了小程序端页面、购物车等设计流程和后台网站的开发流程。第五章则介绍了通过 Postman 调试服务端接口的流程和小程序页面的优化；第六章为项目的总结与展望，分析项目的不足并对优化方向进行总结。

## 第 2 章 系统方案设计

### 2.1 系统需求分析

#### 2.1.1 可行性分析

要打造一款投入较低但又受人欢迎的电子商城不是一件简单的事情，有下面几点需要深思熟虑一番：

首先电子商务有着众多优点，操作便捷、成本较低、交易不受时间和空间限制。但电子商城的实现同样需要大量的资源，服务器资源，个人或公司商户资质，商品储存仓库，快件收发员等，但总体来说在线商城的开销与实体店的投入相比来说还是更容易让人接受，所以在经济上来说开发一款电子商城是具有可行性的<sup>[17]</sup>。

接下来就是主打商品，本人是一名科技爱好者，平时喜欢关注一些智能手机的动态和发展趋势，大学几年里发现身边有此爱好的同学不在少数，即使对智能手机不太了解的年轻人在选购手机时也是跳不出华为、苹果等几大手机厂商的范围，随着骁龙 855，华为 980 等 AI 处理器的在智能手机中的广泛应用，智能手机正处于再度智能化的始点。这也说明了选择智能手机作为我们的主打商品是具有可行性的。

#### 2.1.2 商城功能说明

本项目的目的是开发一款提供目前几大厂商热销商品供用户选购的微信手机商城小程序，待上线后用户可以直接搜索“手机商城”进入小程序页面，在用户进入小程序的同时需要用户进行授权以便获取用户微信名称和与微信绑定的收货地址，避免了用户再次输入带来的繁琐性。在商城中用户可以浏览商品并将商品添加至购物车，在购物车中对商品进行付款，付款成功后订单信息会显示在小程序“我的”界面，以供用户查看<sup>[18]</sup>。

用户操作大致流程如下：

（1）用户搜索小程序名称进入小程序界面，确认授权后浏览商品，此时用户信息会通过服务端的 `addUser` 接口写入数据库。

（2）小程序调用服务端的商品和分类等接口，为用户展示相关商品信息。

（3）用户可将选中的商品加入购物车，购物车提供了添加、删除、单选、全选等操作。

（4）用户可在“我的”界面修改收货地址或直接使用与微信绑定的收货地址，修改后的地址通过 `handleAddress` 接口写入数据库。

（5）用户下单后通过 `Order` 接口创建订单，在检测完库存量充足后调用微信支付 SDK，拉起微信支付，不管用户是否支付在“我的”界面都可以看到该订单信息。

（6）订单信息也会随着用户操作不断更新。

管理员操作后台网站流程如下：

（1）后台网站有超级管理员账号 `Admin`，`Admin` 用户可在管理员管理列表中添加，修改，禁用，恢复其他管理员账号，`Admin` 账号不允许被删除。

（2）任何管理员都可以在后台网站中添加，修改，上架，下架各商品和相关分类，添加时须上传商品图片。

（3）任何管理员均可在后台查看用户生成的订单和订单的详细商品信息。

（4）任何管理员均可在后台查看用户名称、收货地址等信息，但不允许进行更改。

## 2.2 研究使用的平台及技术

### 2.2.1 集成环境

本项目的集成环境是 WampServer, 常用来搭建动态网站或者服务器, 集合了 Apache Web 服务器、PHP 解释器以及 MySQL 数据库。安装简单, 使用便捷, 但使用前需要注意以下事项<sup>[19]</sup>:

(1) 添加数据库密码:

将 mysql 库下的 user 表中 root 账户的 authentication\_string 字段更改为新密码

(2) 设置根目录路径:

在 httpd.conf 文件中找到“\$wwwDir = \$c\_installDir.'/www';”, 改成项目所在目录, 比如改成 D:\Demo;

在 httpd-vhosts.conf 文件中将 80 端口对应的路径改为项目根路径

```
<VirtualHost *:80>

    ServerName www.tp5.cn

    DocumentRoot "D:/Demo/wx_shop/public"

    <Directory "D:/Demo/wx_shop/public">

        AllowOverride All

        Require local

    </Directory>

</VirtualHost>
```

(3) 更改本机的 hosts 文件

在 C 盘下的 \WINDOWS\system32\drivers\etc\hosts 中添加

127.0.0.1    [www.tp5.cn](http://www.tp5.cn)    使本机通过指定域名直接访问项目路径

### 2.2.2 PHP 语言

PHP 是一种简单而强大的开源脚本语言，用于创建动态 Web 内容。全球数百万运行着 PHP 程序的站点证明了它的流行程度和易用性<sup>[20]</sup>。勒道夫、塔特罗、麦步泰尔（2009）出版的《PHP 程序设计(第 2 版)》通过对 PHP 语法的细细梳理，有助于初学者进行全面系统的学习，构建起完整的 PHP5 知识体系<sup>[21]</sup>。上野宣（2014）出版的《图解 HTTP》详细的介绍了 HTTP 协议和请求设计的字段以及响应状态码，关于 web 安全方面也有一定的讲解，这本书对初学者了解网络通信有很大的帮助<sup>[22]</sup>。伦纳德·理查德森（2014）《RESTful Web APIs 中文版》针对 RESTful API 的实用指南，通过展示各种用来创建高可用应用的强大工具，讲解 REST 的深层原理，以及介绍基于超媒体 API 的策略，通过学习可设计出让客户高度满意的 RESTful 的 web API<sup>[23]</sup>。

## 2.3 数据库设计分析

在 WampServer 下的 MySQL 中建立数据库：db\_wx\_shop。

### 2.3.1 小程序客户端建表分析

（1）t\_shop\_user 表存储用户登录小程序后微信服务器中生成的唯一身份标识 openid 等相关信息。

（2）t\_shop\_address 表存储用户地址信息，表中 user\_id 字段与 t\_shop\_user 表相关联。

（3）t\_shop\_image 表存储商品和分类中各种图片信息，如图片链接等。

（4）t\_shop\_category 表存储商品分类信息，表中 img\_id 字段指向 t\_shop\_image 表中该分类的主题图片信息。

（5）t\_shop\_product 表存储商品信息，包括商品名称，商品价格，商品库存量，商品图片链接等信息，表中的 category\_id 与 t\_shop\_category 表关联，指明该商品所属分类。

（6）t\_shop\_detail 表存储每件商品的具体信息，包括商品具体名称，商品介绍等，表中 product\_id 字段与 t\_shop\_product 表关联。

（7）t\_shop\_order 表存储用户产生的订单信息，包括订单信息、订单快照信息、用户快照信息，表中的 status 字段表示该订单目前的状态信息，user\_id 字段与 t\_shop\_user 表相关联，各快照信息中的 product\_id 字段与 t\_shop\_product 表关联，方便后台网站查看具体订单商品信息。

以上部分表中均包括创建，修改，删除时间字段，其中后台所有删除（下架）操作均属于软删除，只会更新表中的 is\_delete 字段。

### 2.3.2 后台网站建表分析

后台中的各商品、订单、用户信息均是从上述表中读取，只需建一张存储后台管理员信息的表 t\_shop\_backend\_user，表中存储管理员名称，密码，创建时间，登录次数，软删除标志位 is\_delete 等字段。

## 2.4 服务端接口设计分析

### 2.4.1 用户提交数据验证流程

整个服务端架构大致如图 2.1。

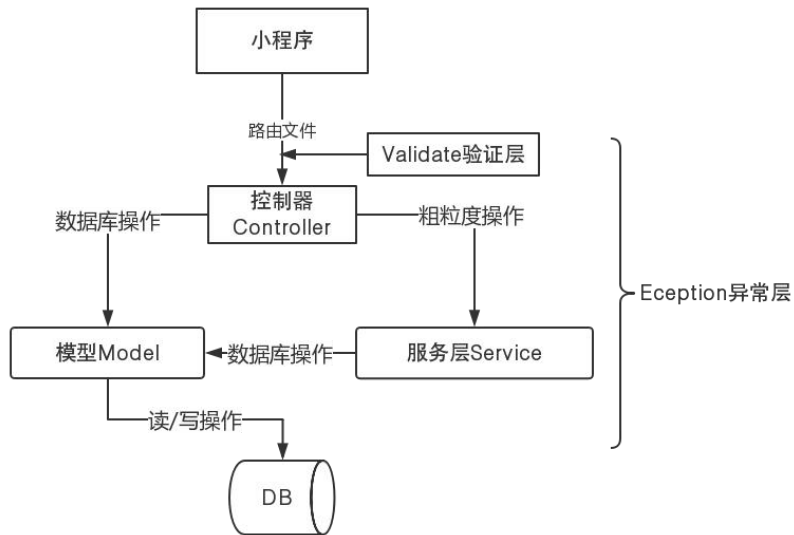


图 2.1 服务端架构

对于用户提交的数据首先会经过 validate 层进行验证，在 validate 层内通过 BaseValidate 类继承 tp5 自带的 validate 方法，在 BaseValidate 内批量检查传入参数，如果参数异常则抛出 ParameterException，返回给客户端自定义异常格式方便客户端反馈错误。用户向每个接口提交的数据在该接口内都会调用 validate 层的 goCheck()，goCheck()方法如下：

```

public function goCheck(){
    $param = Request::instance()->param();
    //批量检查传入参数
    $result = $this->batch()->check($param);
    if(!$result){
        //参数异常
        $e = new ParameterException([
    
```



```

        //传入异常信息
        'msg' => $this->error

    });

    throw $e;

} else {

    return true;

}

}

```

例如在处理用户提交的地址信息时会在 `handleAddress（）` 接口内调用对应的验证类 `Address（）`

```

$validate = new Address();
$validate->goCheck();

```

其中 `Address` 继承之前写好的 `BaseValidate` 类，指定验证字段和对应的验证规则，具体的验证规则也会在 `BaseValidate` 中声明，方便多个验证类进行调用。

```

class Address extends BaseValidate{

    protected $rule = [

        'name' => 'require|isNotEmpty',

        'mobile' => 'require|isNotEmpty',

        'province' => 'require|isNotEmpty',

        'city' => 'require|isNotEmpty',

        'country' => 'require|isNotEmpty',

        'detail' => 'require|isNotEmpty'

    ];

}

```

}

### 2.4.2 具体接口分析

在解决了用户提交数据的验证流程后就可以编写服务端接口了,具体接口如下:

(1) Token 接口: 通过小程序的 `app_id` 和 `app_secret` 和小程序上传的用户 `code` 码, 从微信提供的使用 `code` 换取用户 `openid` 及 `session_key` 的 `login_url` 中获取用户唯一表示 `openid` 和 `session` 信息, 以此来判断每次进入小程序的用户是否曾经使用过该小程序, 同时为该用户生成唯一 Token, 并将该 Token 和用户的 `openid`、表中的 `user_id`, 以及用户权限分别作为 `key,value` 存入 `tp5` 自带的缓存中, 设置过期时间为 2 小时, 防止为验证用户信息而频繁读取数据库。

(2) Address 接口: 该接口包括 `createOrUpdateAddress` 方法和 `getUserAddress` 方法, 其中 `createOrUpdateAddress` 为用户创建或更新地址信息使用, `getUserAddress` 为用户点击“我的”`tab` 栏时展示用户收货地址时使用。

(3) Product 接口: Product 接口包括用户进入“首页”界面时随机展示相关商品的 `getRandom` 方法, 用户点击相关分类时展示该分类下的商品信息的 `getALLByCategoryID` 方法, 以及用户点击某一商品时的 `getOneProduct` 方法。

(4) Category 接口: 该接口直接在用户点击“分类”`tab` 栏时为其展示所有分类信息。

(5) Order 接口: 该接口比较复杂, 包括生成订单信息的 `createOrder` 接口, 用户在“我的”`tab` 栏查看所有订单的 `getAllOrders` 接口, 以及用户查看某一订单详情的 `getDetailByID` 接口, 其中包括多次用户身份验证

和库存量检测。

（6）Pay 接口：该接口接受用户提交的订单 id，经库存量检测和验证订单有效性后调用微信预订单接口，向微信请求订单号并生成签名，因为每次用户取消支付再对同一订单支付生成的 prepay\_id 不同，故需要将用户拉起微信支付后产生的 prepay\_id 存入 t\_shop\_order 表中，同时将其其他快照信息也存入表中，通过之前调用微信预订单接口时设置的回调接口接受微信通知判断支付是否成功，将相关信息也写入数据库中。

## 2.5 小程序端页面设计分析

小程序端商城页面仿照手机淘宝，手机京东等 APP 大致页面布局，分为四个 Tab 栏界面：“首页”，“分类”，“购物车”，“我的”。各页面布局如下：

### 2.5.1 首页页面

该页面大致包括可自动切换的首页轮播图，每个轮播图对应一种商品，点击可跳转到商品详情页；多个主题商城入口，点击可进入主题商城页面，如荣耀商城，小米商城等，每个商城里包括几款热卖商品；猜你喜欢标题栏下则是从数据库随机拉取的 15 款商品，用户可通过下拉该页面不断刷新该区域的商品信息。如图 2.2 所示。

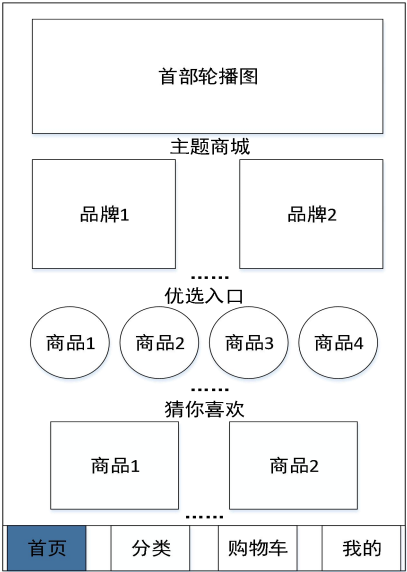


图 2.2 首页页面

2.5.2 分类页面

该页面主要包括左侧的分类选择栏,用户每点击一个分类选项右侧区域则显示该分类的主题图片和分类下的全部商品信息,如图 2.3 所示。



图 2.3 分类页面

2.5.3 购物车页面

购物车页面显示用户加入购物车的商品信息，包括商品图片、名称、价格、数量等等，并且购物车模块提供了类似各大手机商城 APP 主流使用的全选，加、减商品数量，删除某一商品等功能；同时左下角显示用户此时选中的商品价格和支付点击入口。如图 2.4 所示。

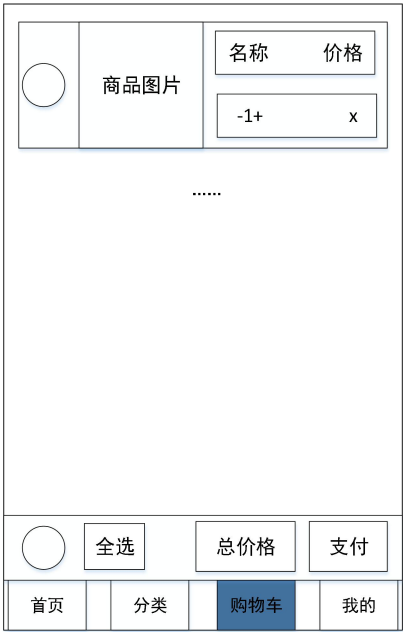


图 2.4 购物车页面

2.5.4 我的页面

“我的”界面分为三部分，从上往下依次是用户信息，包括用户头像，名称等信息；中间位置为用户当前的收货地址信息，可在地址管理中进行更改；最下方则是用户产生的订单信息，用户可通过不断地上拉查看自己的历史订单信息，每次默认加载 10 条，可加载当前用户的全部订单信息。如图 2.5 所示。

头像

用户名称

收货地址

收货人名称

收货人联系方式

收货人具体地址

订单管理

订单图片

订单名称

订单总价

状态

付款

.....

首页

分类

购物车

我的

图 2.5 我的页面

## 2.6 后台网站架构设计分析

后台网站架构较为简单，主要功能如下：

- （1）用户输入网址至后台登录界面，输入用户名，密码信息，验证成功后进入后台网站。
- （2）后台网站主要包括分类、商品、订单、用户信息、管理员管理等模块。
- （3）超级管理员 Admin 可在管理员管理模块中添加新的管理员信息，Admin 不可被删除。
- （4）任何管理员均可在分类管理、商品管理、订单管理、用户信息管理中添加、修改、删除相应信息，包括分类、商品的上下架等操作。其中订单信息和用户信息不可被更改和删除，这部分信息仅能被用户自己更改。

（5）后台网站会检测管理员当前是否登录，已登录则继续访问相应页面，未登录则跳转至登录页面进行登录，登陆后的用户可点击退出登录退出管理后台。

## 2.7 系统总体架构

经过对数据库、服务端接口、小程序页面、后台网站等的详细分析后，整个系统的架构可如图 2.6 所示<sup>[24]</sup>。

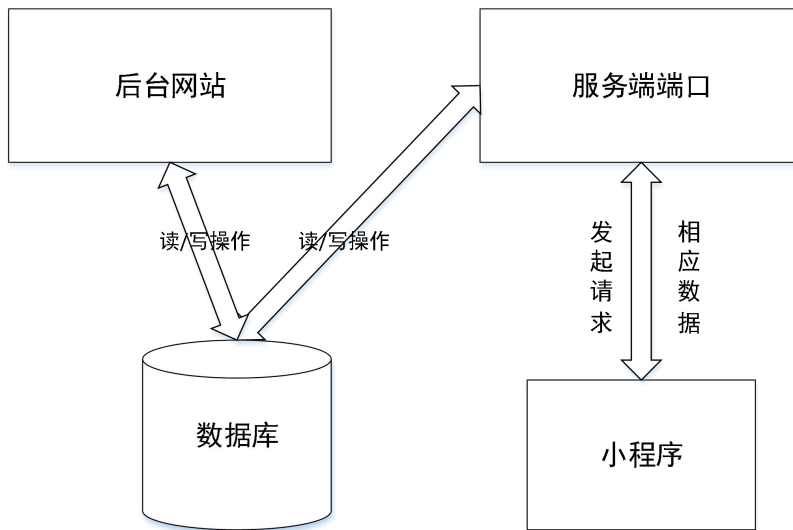


图 2.6 系统架构图

整个系统的架构如上图所示，小程序端对用户的各项操作向服务器接口发起相应请求，服务端从数据库取出相应数据返回给小程序进行展示，或是将小程序端传递来的用户和订单信息写入数据库；同时后台网站也会向数据库读写相关商品和管理员信息。

至此本项目的前期规划到此结束，整个系统的架构看似较为简单，流程也一目了然，但其中的难点、复杂度却远不止于此，例如如何设计数据表可最大限度的降低字段冗余；如何对用户产生的订单信息进行多次库存

量检测防止“超卖”发生；服务端接口又该如何编写以提高代码的复用；小程序端购物车的诸多操作又该如何实现等等，这些都是亟待解决的问题。



## 第 3 章 服务器端设计

### 3.1 数据库设计

经过对数据库设计的简要分析后接下来就是如何设计好本项目需要用到的各数据表，数据库是本项目的根基，要最快速最便捷的进行开发就要设计出既满足功能需要又最大限度的减少数据冗余的数据表<sup>[25]</sup>。本项目使用的表介绍如图 3.1 所示。

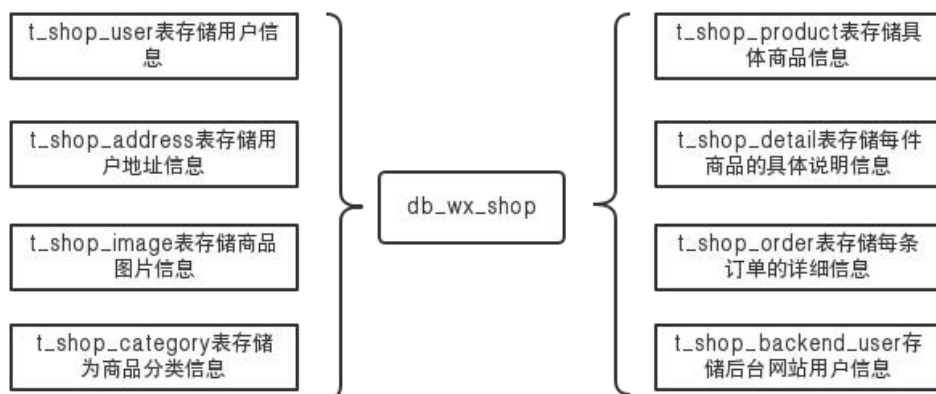


图 3.1 数据库架构图

以下每个表中都包含：创建时间（create\_time）、更新时间（update\_time）、是否删除（is\_delete）、删除时间（delete\_time）字段。

（1）t\_shop\_user 表存储从微信获取的用户信息，表字段如下：

id, open\_id, nick\_name。

（2）t\_shop\_address 表存储用户地址信息，表字段如下：

id, user\_id, name, mobile, province, province, city, country, detail。

其中 user\_id 字段与 t\_shop\_user 表相关联。

（3）t\_shop\_image 表存储商品图片信息，后台上传商品信息也会存入该表，因表中图片不仅为商品图片，将图片链接拆分出来可更方便的进行管理。字段如下：

id, img\_url。

其中 img\_url 字段存储的为图片名称，读取后拼接上图片路径后返回给客户端。

（4）t\_shop\_category 表存储的为商品分类信息，字段如下：

id, category\_name, img\_id, category\_desc。

其中 img\_id 字段指向 t\_shop\_image 表中该分类的主题图片信息。

（5）t\_shop\_product 表存储具体商品信息，字段如下：

id, name, price, stock, category\_id, img\_url。

表中的 category\_id 与 t\_shop\_category 表关联，为多对一关系，指明该商品所属分类，方便查找某分类下的商品信息。

（6）t\_shop\_detail 表存储每件商品的具体说明信息，为用户点击某商品时进行说明展示使用，字段如下：

id, name, detail, product\_id。

其中 product\_id 与 t\_shop\_product 关联，为多对一关系。

（7）t\_shop\_order 表存储每条订单的详细信息，方便用户查看历史订单使用，字段如下：

id, order\_no, user\_id, prepay\_id, total\_price, total\_count, status, snap\_name, snap\_img, snap\_items, snap\_address

其中 user\_id 字段与 t\_shop\_user 表中的 id 进行关联；prepay\_id 为用户进行微信支付时生成的支付标识 id；status 字段为支付状态标识，字段

为类型为 `tinyint(4)`，具体字段含义已在配置文件中写明；`snap_name` 字段为订单总名称，一件商品时为该商品名称，一条订单包括多件商品时为第一件商品名称加上“等”进行展示；`snap_img` 为订单总图片链接，订单包括一件或多件商品时均以第一件商品图片作为订单图片；`snap_items` 和 `snap_address` 均为 json 字符串，其中 `snap_items` 存储有该订单下所有商品的信息，`snap_address` 为订单创建时的用户地址信息。

（8）`t_shop_backend_user` 表中存储后台网站管理员信息，字段如下：

`id`, `user_name`, `user_pwd`, `login_count`, `login_time`。

包括用户名称，密码，登录时间，登录次数等字段。

## 3.2 商品接口设计

创建 `Product` 类继承自 `think\Controller`，该类包括 3 个方法：`getRandom()`，`getProducts()`和 `getOneProduct()`。

### 3.2.1 getRandom()

流程图如图 3.2 所示，该方法为用户进入商城首页时调用，从数据库随机拉取 20 条商品在“猜你喜欢”栏下进行展示。

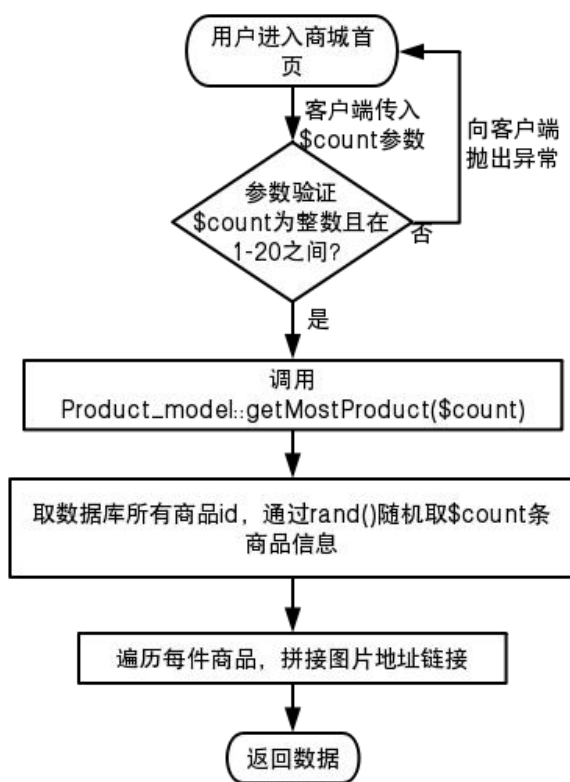


图 3.2 getRandom 流程图

首先调用继承自 Validate 类的 Count 类中的 count()方法对传入参数 \$count 进行验证，参数 \$count 必须为整形，随后通过 Product\_model::getMostProduct(\$count)方法从数据表 t\_shop\_product 中随机拉取\$count 条商品信息，对所有商品进行遍历，为每件商品中的图片链接拼接上图片路径信息：

```

$products[$key]['img_url'] =
config('setting.img_prefix').$value['img_url'];

返回所有商品信息。
    
```

### 3.2.2 getProducts()

流程图如图 3.3 所示，用户在商城分类界面点击左侧分类 tab 栏时会调用该方法，首先调用继承自 Validate 类的 Count 类中的验证方法对传入参数 \$id 进行验证，参数 \$id 必须为整形。通过 Product\_model 模型中的 getProductsByCategoryID(\$id) 方法从 t\_shop\_product 表中拉取所有商品分类为 \$id 的商品信息，为每条商品图片信息拼接上图片路径后返回给小程序端。

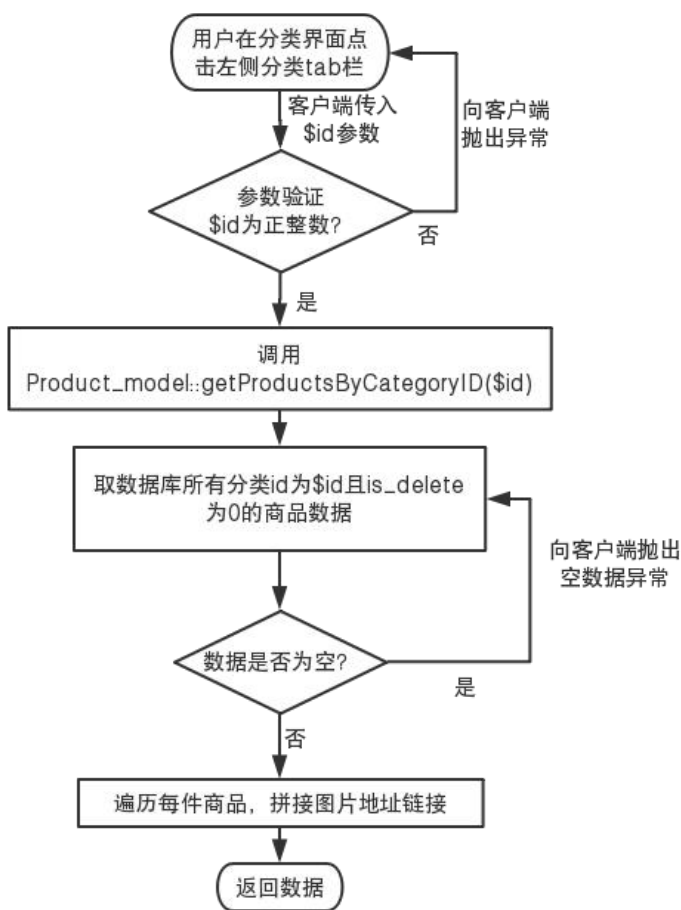


图 3.3 getProducts 流程图

### 3.2.3 getOneProduct()

流程图如图 3.4 所示，用户在商城首页和分类页面点击某一商品图片时会调用此接口，小程序向服务端传入商品 ID，通过验证器对参数进行整数验证，调用 Product\_model 模型中的 getProductDetailByID(\$id)方法获取该商品的图片和商品介绍信息，返回给小程序端进行展示。

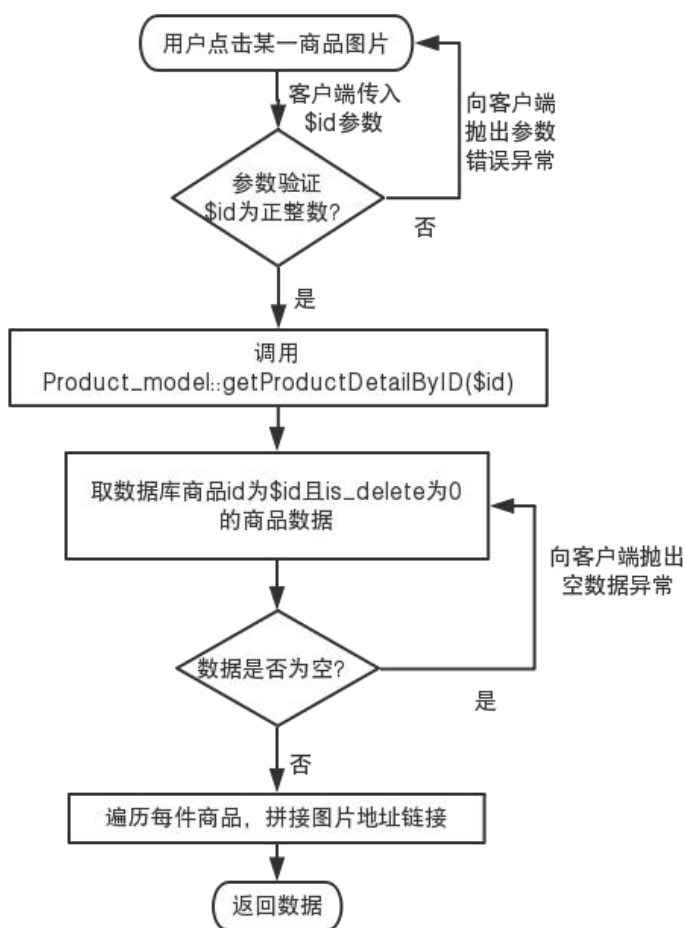


图 3.4 getOneProduct 流程图

### 3.3 分类接口设计

流程图如图 3.5 所示，创建 Category 类继承自 think\Controller，该类仅包括 1 个方法 getAllCategories（），当用户点击商城下方的 tab 栏中的分类栏时用来获取全部的分类信息在分类页面左侧进行展示，包括各分类 ID 和名称，当用户点击某一分类名称时会携带该分类 ID 请求 Product 类下的 getProducts 方法，为用户展示当前分类下的商品明细。

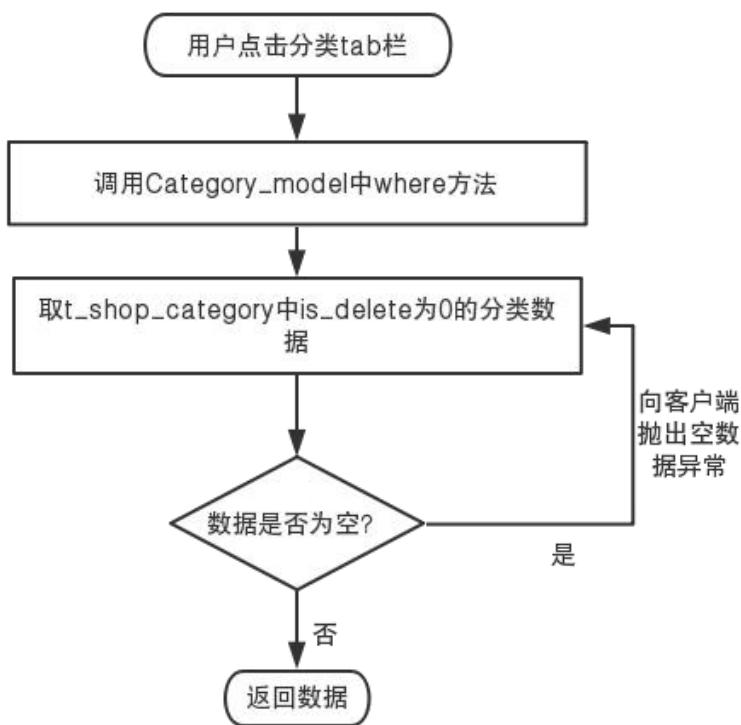


图 3.5 分类接口流程图

### 3.4 用户 Token 接口设计

创建 Token 类继承自 think\Controller，该类包括获取 Token 和验证 Token 两个方法。因 Token 接口较为复杂，仅适用 Controller 和 model 两

层显得较为臃肿，现将 Model 层进行拆分，拆分为粗粒度的 Service 层和细粒度的 Model 层，其中 Service\Token 中的方法主要用户生成、验证、缓存 Token，Model\Token 中的方法主要是与数据库进行交互，将生成的用户 token 在数据库中进行读取、写入操作。

### 3.4.1 getToken()

流程图如图 3.6 所示，首先调用 Service\Token 中的 get()方法，在该方法中从微信提供的使用 code 换取用户 openid 及 session\_key 的 login\_url 中获取用户唯一表示 open\_id 和 session 信息，调用时需携带小程序的 app\_id、app\_secret 和小程序上传的用户 code 码，该 code 码为用户登录小程序时产生。在返回数据正常的情况下服务端就成功地获取到了用户的 open\_id，如果该用户是首次登录则将 open\_id 和用户信息写入数据库，否则的话直接从数据库读取该 open\_id 对应的用户唯一 ID。

接下来就是生成 token,因项目较为简单，生成 token 也不采用复杂算法，只需对一个随机获取的 32 位字符串和当前时间戳以及服务端自定义的一个长度不限的 salt 字符串进行 md5 加密即可。将生成的 token 作为 key，用户的 open\_id 和 user\_id 以及其他用户信息作为值存入 tp5 自带的 cache 中，设置 expire\_time 为两小时。



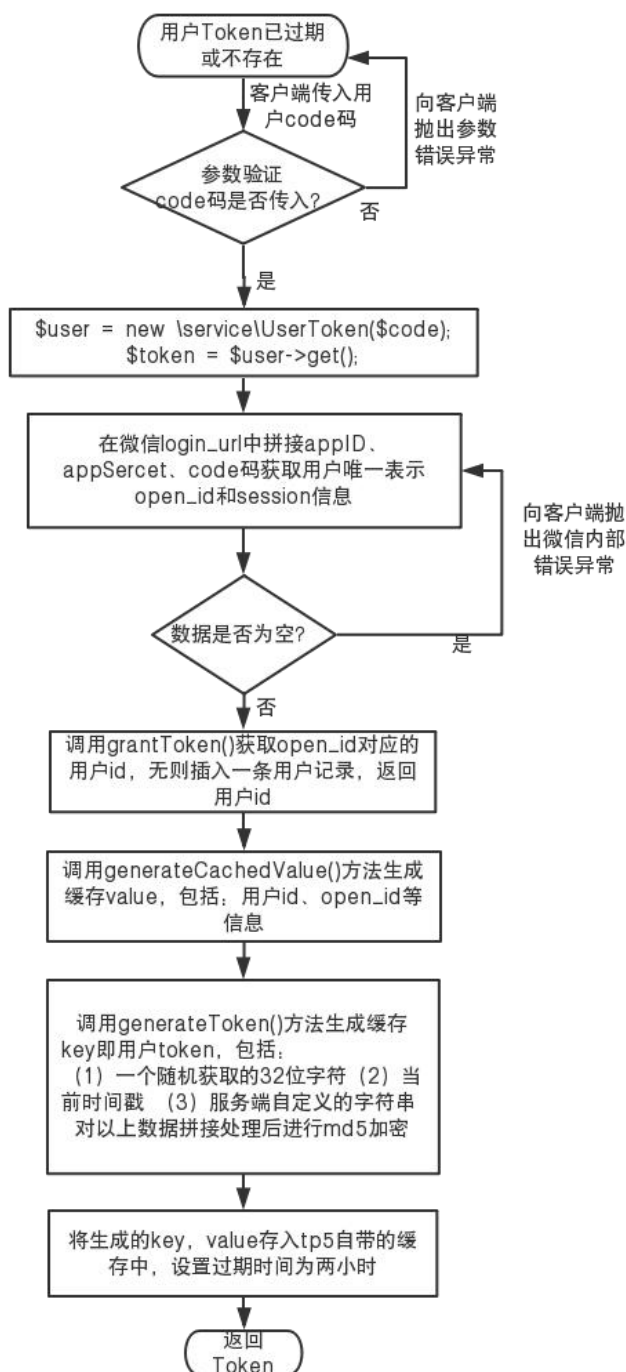


图 3.6 getToken 流程图

用户登录小程序成功且服务端返回给相应 token 后，接下来的两小时内用户进行任何操作都需要携带在请求的 header 中携带 token 进行身份验证，请求的部分头信息<sup>[21]</sup>如图 3.7 所示。



图 3.7 请求头信息

验证时取出 token 通过服务端的 Service\Token 中的 verifyToken(\$token)，用传入的 token 作为 key 从缓存中获取对应的值，如果 token 已超过两个小时的有效期或在缓存中未查找到则返回 false，让客户端重新调用 getToken 接口请求新 token。

### 3.4.2 verifyToken()

流程图如图 3.8 所示，验证 token 的具体方法已经在上面讲过，用户 token 有效时会返回 true，过期或者不存在时返回 false。

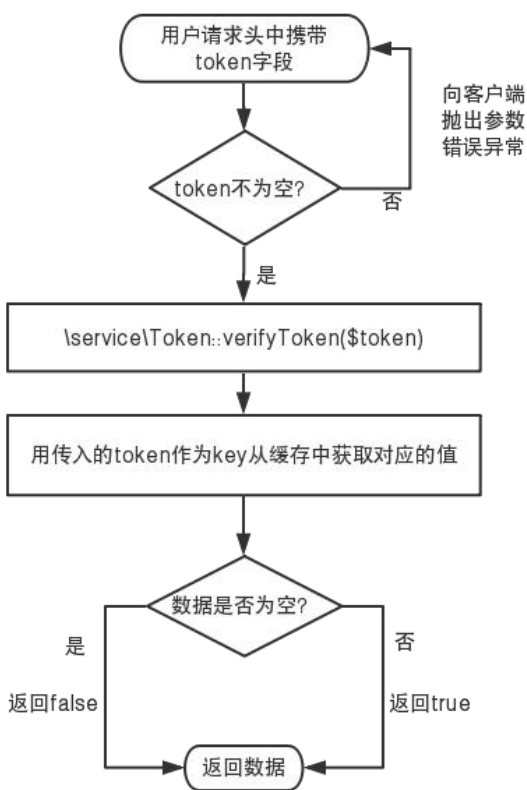


图 3.8 verifyToken 流程图

### 3.5 用户地址接口设计

创建 Token 类继承自 think\Controller, 该类包括创建/更新用户地址信息, 获取用户地址信息两个方法。用户登录后会首先调用 getToken 接口生成 token 信息便于操作某些功能时进行身份验证, 其中添加、修改用户地址信息, 创建、查看订单信息, 支付等功能的实现都需要先对用户进行身份校验。

#### 3.5.1 handleAddress()

流程图如图 3.9 所示, 从客户端请求头中取出 token, 用该 token 作为

key, 从 tp5 自带的缓存中取出对应的 value, 如果操作失败则抛出异常“尝试获取的 Token 变量并不存在”, 成功则从 value 中取出之前存入的用户 uid 信息,

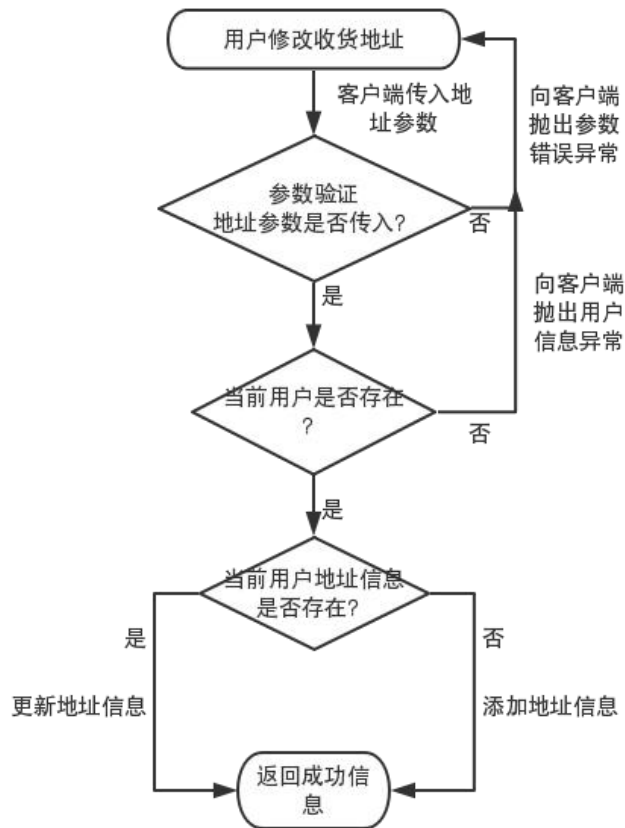


图 3.9 handleAddress 流程图

用 uid 去 t\_shop\_user 表中获取用户信息, 失败则将下列错误信息返回给客户端。

```

$user = UserModel::get($uid);

if(!$user){

    throw new UserException([

```

```
'status' => 404,
'desc' => '用户信息不存在',
'data' => []
]);
}
```

等上述验证均通过后则根据\$uid去t\_shop\_user\_address表中去查找相应用户地址信息，有则修改，无则创建，返回成功信息通知小程序。

### 3.5.2 getAddress()

流程图如图 3.10 所示，同 handleAddress()处理流程相同，通过 token 获得用户 id，去 t\_shop\_user\_address 表中获取地址信息返回给小程序。

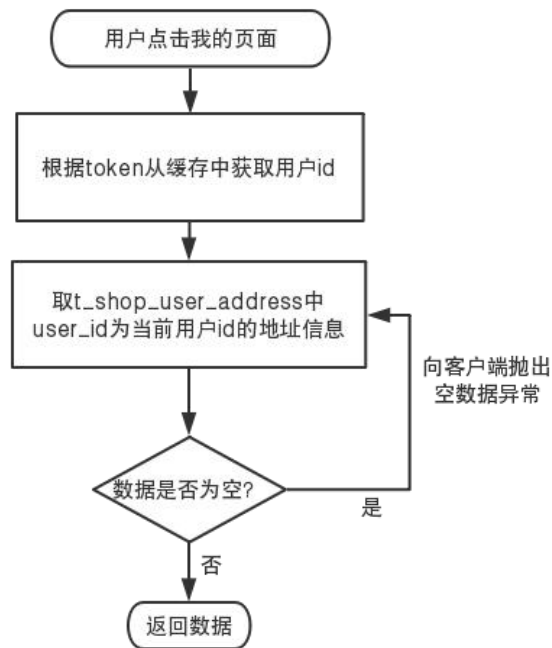


图 3.10 getAddress 流程图

## 3.6 订单接口设计

### 3.6.1 订单和支付流程

订单和支付两个接口时本项目设计过程中的重要环节,也是本项目的重难点所在。因需要使用微信支付 SDK<sup>[18]</sup>和进行多轮库存量检测,整体较为复杂,故将其抽离,在 Service 层实现相关代码,提高代码的复用性的同时也使得整个架构更为清晰。其流程可如图 3.11 所示。

- (1) 用户选择商品后,提交商品信息;
- (2) 服务端接收商品信息,检测订单中的商品库存量(第一轮);
- (3) 有库存时订单数据存入数据库,将下单成功,可以进行支付的信息返回给小程序,此时订单为未支付状态;
- (4) 用户点击付款,服务端调用支付接口进行支付;
- (5) 支付前还需对订单中的商品进行一次库存量检测(第二轮),防止用户下单后较长时间后再进行支付操作;
- (6) 调用微信支付接口进行支付,同时通过回调接口接收微信的异步推送获得支付结果,成功后会再次进行一次库存量检测(第三轮),防止出现在支付过程中该商品已售罄的情况;
- (7) 成功进行支付后会将数据库中的订单商品的库存量进行相应扣除,此时订单状态为已支付;失败则不扣除,此时订单状态为已支付但库存不足。

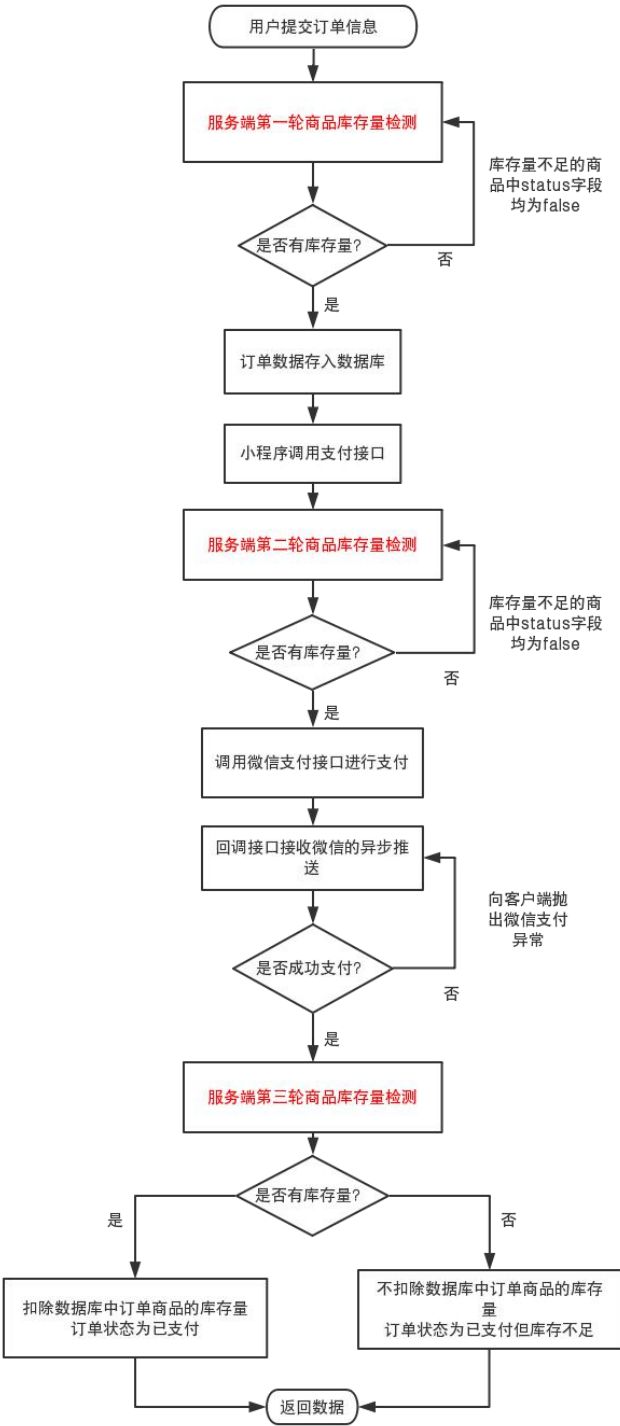


图 3.11 支付流程图

### 3.6.2 库存量检测流程

对用户提交的订单进行库存量检测的方法会被多个接口使用，故在 service\Order 下封装了 place 方法进行库存量检测，只需将用户 id 和需要验证的商品信息（包括商品 id，商品数量）作为参数传入即可，整个库存量检测流程如图 3.12 所示。

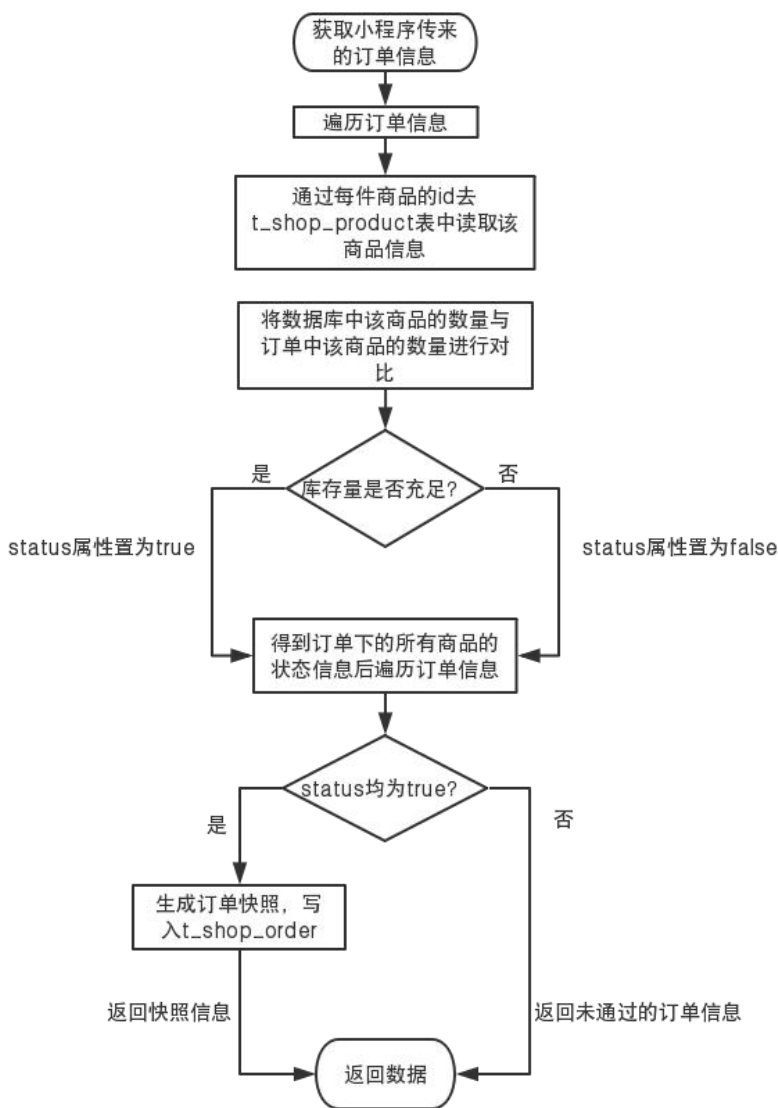


图 3.12 库存量检测流程图



- （1）获取小程序传来的订单信息，包括每件商品的 id 和数量；
- （2）根据订单中每件商品的 id 去 `t_shop_product` 表中读取订单下所有商品信息用于库存量检测；
- （3）对订单的中的每件商品进行分别处理，将数据库中该商品的数量与订单中该商品的数量进行对比，库存量充足则将该商品下的 `status` 属性置为 `true`，同时将该商品的名称，图片，总价，数量，`status` 返回给上一级方法；
- （4）得到订单下的所有商品的状态信息后对整个订单进行遍历，只有订单下每件商品的 `status` 均为 `true` 时才能通过库存量检测，同时计算每件商品的总价格和商品总数量，生成订单快照；
- （5）订单快照包括两部分，一是商品快照，即订单下的详细商品信息；二是下单时用户信息快照，包括用户名，用户收货地址信息等，以免后期商品信息或用户地址信息发生更改时会造成数据丢失的情况；
- （6）将快照信息写入数据表 `t_shop_order`，同时将每个订单 id 对应的商品 id 写入数据表 `t_shop_order_product`。

### 3.6.3 订单接口实现

首先创建 `Order` 类继承自 `think\Controller`，该类包括生成订单，在“我的”页面当用户下拉查看订单信息时分页获取用户订单信息，当用户点击某一订单时获取该订单的详情信息等方法。

#### （1）`generateOrder()`:

用户在购物车页面选择下单后，小程序会向服务端传递每个商品的 `product_id` 和数量 `product_count` 信息，在处理数据之前首先对用户身份进行验证，相应方法已在用户地址信息接口中进行详细介绍。在身份验证通过后即开始进行第一次库存量检测，库存量检测通过后将订单快照信息返

回给小程序，用来在“我的”页面进行订单展示。

### （2）getOrders():

用户进入“我的”页面后，小程序会去服务端请求 15 条订单信息，按订单创建的时间进行倒叙展示；在用户浏览浏览到页面底部时小程序会再去服务端请求 15 条订单信息，直到用户订单信息展示完毕为止。首先验证传入的 \$size 和 \$page 参数为整数且非空，通过 token 获取到用户 ID 后通过 tp5 的 paginate() 来实现分页功能：

```
$pagniData      =      self::where('user_id','=',$uid)->order('create_time
desc')->paginate($size,true,['page'=>$page]);
```

### （3）getOrderByID():

用户点击某一订单时可为其展示该订单下的具体商品信息和下单时的用户信息。此时小程序会携带订单 ID 请求该接口，首先对订单 ID 进行验证，验证通过后在 t\_shop\_order 表中取出该 ID 下的订单信息，返回给小程序展示。

## 3.7 支付接口设计

### 3.7.1 调用微信支付流程

微信支付的使用需要先取得支付资格，因微信支付不对个人开放，须通过小程序 ID、商户号去微信开发平台申请商户支付资格，取得资格后才可成功调用微信支付 SDK，拉起微信支付。因此本项目只能将微信支付功能暂时屏蔽，但实际流程已仿照支付教程开发完毕，等待拥有商户号进行测试后即可上线使用。微信支付流程如图 3.13 所示。

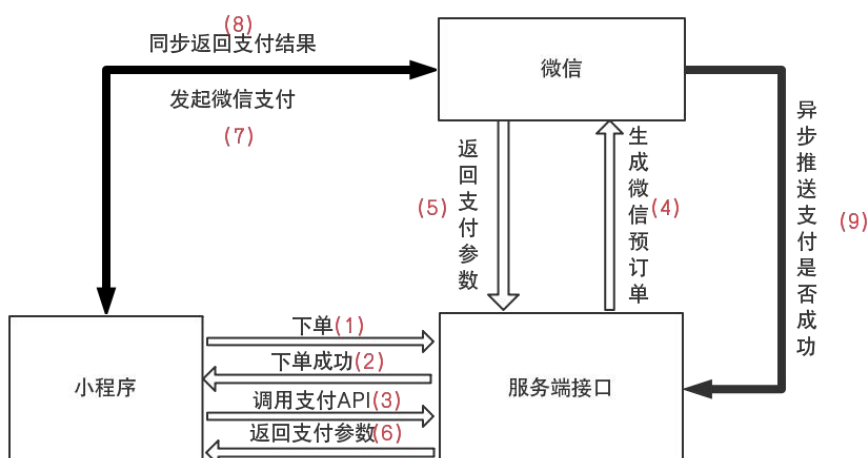


图 3.13 微信支付流程图

调用微信支付流程看起来较为复杂，但开发者文档中描述的相对清晰，理解起来不算太困难。

### 3.7.2 支付接口设计

创建 Pay 类继承自 think\Controller，该接口包括生成微信预订单，接收微信异步推送支付信息两个方法。

#### (1) pay():

用户对某一订单点击付款进行支付操作时，小程序会携带订单 id 请求该接口，首先验证订单 ID 的有效性，在 service\Pay 中再验证一次库存量是否充足，充足时则调用微信预订单接口，需将用户的 open\_id，订单号，订单价格传入：

```

$wxOrderData = new \WxPayUnifiedOrder();
$wxOrderData->SetOut_trade_no($this->orderNo);
$wxOrderData->SetTrade_type('JSAPI');
    
```

```
$wxOrderData->SetTotal_fee($status['orderPrice']);
$wxOrderData->SetOpenid($openid);
$wxOrderData->SetBody('手机商城');
$wxOrderData->SetNotify_url('回调接口');
```

随后向微信请求订单号,因每次用户取消支付再对同一订单支付生成的 prepay\_id 不同,需要将返回信息中的 prepay\_id 存入 t\_shop\_order 表中的该订单 ID 下,将小程序 ID 和 prepay\_id 作为参数向微信支付 API 请求包括支付签名在内的支付参数。

```
$jsApiPayData = new \WxPayJsApiPay();
$jsApiPayData->SetAppid(config('wx.app_id'));
$jsApiPayData->SetTimeStamp((string)time());
$jsApiPayData->SetNonceStr($rand);
$jsApiPayData->SetPackage('prepay_id='.$wxOrder['prepay_id']);
$jsApiPayData->SetSignType('md5');
$sign = $jsApiPayData->MakeSign();
$rawValues = $jsApiPayData->GetValues();
```

获得支付参数后传给小程序,小程序可以凭借该信息拉起微信支付。

## (2) receiveNotify():

该方法为接受微信支付异步通知信息的接口,首先创建 WxNotify 类继承自微信 SDK 中的 \WxPayNotify,在 WxNotify 下重写 NotifyProcess 方法进行接受回调参数。当返回参数中的 result\_code 为 SUCCESS 时表明支付成功,此时需要将数据库中该订单下的所有商品进行减库存操作,同时更新订单状态。

## 第 4 章 小程序端和后台网站设计

### 4.1 小程序页面设计

小程序端页面分为 4 个主页面，分为“首页”、“分类”、“购物车”、“我的”。其中“首页”页面主要是分为三大块，轮播图、主题商城、猜你喜欢；“分类”页面分为左侧的分类 tab 栏和右侧的分类下商品信息；“购物车”页面显示添加至购物车的商品信息；“我的”页面显示用户相关信息以及用户生成的订单信息。

#### 4.1.1 首页布局

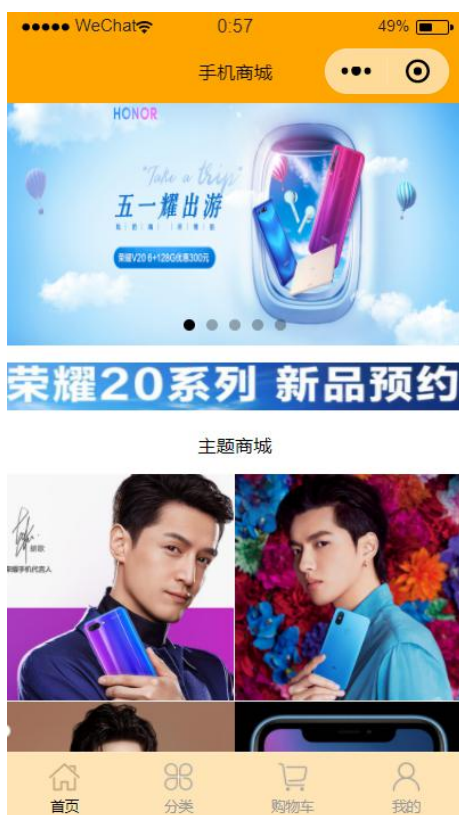


图 4.1 首页布局

（1）首页布局如图 4.1 所示，最上方为五种商品的轮播图，点击可进入相关商品页面，如图 4.2。



图 4.2 商品页面

（2）中间为四个主题商场入口，点击可进入主题商城页面，如图 4.3。



图 4.3 主题商城页面

（3）最下方“优选入口”和“猜你喜欢”板块，“优选入口”板块展示部分热卖商品，“猜你喜欢”板块随机为用户拉取 20 款商品，如图 4.4 所示。每次下拉首页都会刷新该商品区。两大板块点击某商品均可进入该商品的详情页面。



图 4.4 优选入口

4.1.2 分类布局

分类页面如图 4.5，该页面左侧可选择某一商品品牌，右侧上方显示该商品品牌图片，下方为该分类下所有商品信息，点击某一商品图片可进去该商品详情页面。





图 4.5 分类页面

### 4.1.3 购物车页面

(1) 该页面显示所有用户添加至购物车的商品信息，具体信息可见图 4.6。用户还可以单独对某一商品的数量进行增减操作，对某一商品进行选择、删除操作。下方的全选按钮选中时所有商品的左侧选中按钮显示全部选中，下方的价格为当前所有已选中商品的总价，点击右侧箭头则进

入订单详情页面。

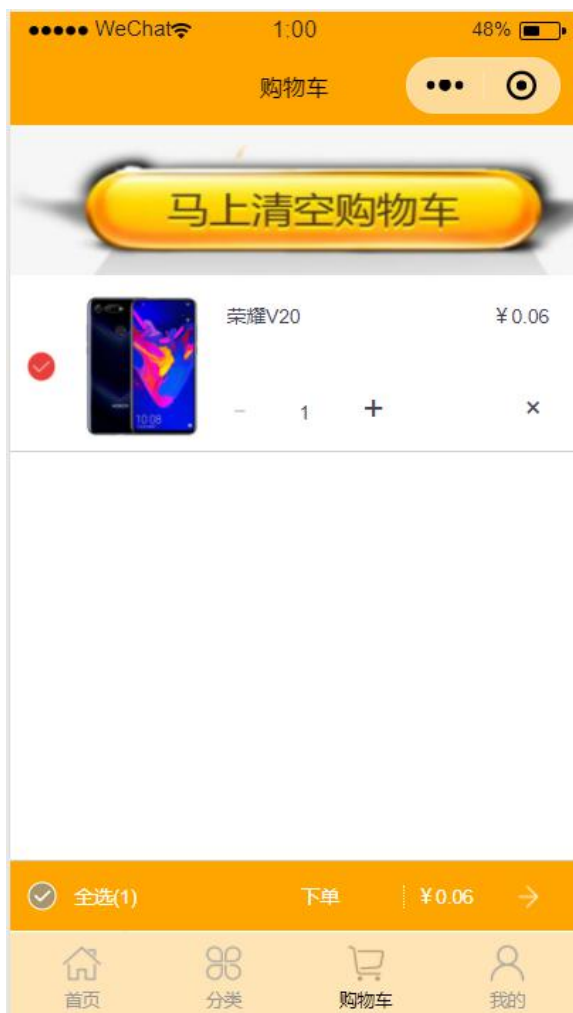


图 4.6 购物车页面

(2) 订单详情页面如图 4.7，显示用户当前订单中的所有商品和用户名称、收货地址、联系方式等。最上方为用户名，联系电话，收货地址信息，点击右侧箭头可修改收货地址信息。下方则为当前订单的所有商品信息，商品名、图片、数量、总价。底部付款合计为订单总价，点击结账进入支付页面，拉起微信支付，因本项目已将微信支付屏蔽，不能演示支付

页面，支付页面为微信内部页面，同平时使用时的付款页面完全一样。



图 4.7 订单详情页面

当订单中的商品有库存量不足的情况时会提示某商品缺货,不能进入支付页面,如图 4.8。



图 4.8 库存不足

#### 4.1.4 我的页面

（1）“我的”页面如图 4.9，该页面为用户信息、收货地址、订单信息展示页面。上方为用户头像和用户名称，中间是用户信息，分为用户姓

名、电话号码、收货地址三栏，点击右侧箭头可进行编辑。



图 4.9 我的页面

（2）下方是我的订单，如图 4.10 所示。展示用户最近生成的订单信息，包括订单编号、订单图片、名称、数量、当前状态、订单总价等，点击订单图片可进入详情页面，点击付款进入支付页面。



图 4.10 订单页面

## 4.2 购物车实现

### 4.2.1 购物车功能

本项目涉及到的购物车需要具备以下功能：

（1）用户点击加入购物车时可以将所选商品的名称，选择数量，价格，图片链接等信息添加至购物车页面显示；

（2）如果某商品已存在与购物车中，再次添加时则需要在原数量的基础上增加上新选择的数量，不存在时才会向购物车添加该商品；

（3）用户可在购物车增加、减少已选商品的数量，还可删除某件商品，直至到购物车不存在任何商品；

（4）用户可以对购物车商品进行全选或部分选择进行付款，同时下方总价处显示购物车已被勾选的商品的总体价格。

#### 4.2.2 购物车具体实现

购物车的实现需要用到小程序的本地缓存，购物车数据存放在本地，当用户添加某商品进购物车时会将该商品信息存入缓存中；修改数量时也会更新缓存数据；当用户选中某些商品下单购买时，会从缓存中删除该数据，更新缓存；当用用户全部购买时，直接删除整个缓存。

（1）加入购物车：在 `cart_model` 下创建 `add` 方法，本地缓存中存储有购物车数据。首先调用 `getCartDataFromLocal` 方法从本地缓存中获取全部数据。遍历获得的购物车数据，判断其中是否已经存在该商品即缓存中的数据 `id` 与添加的商品 `id` 是否一致，如果之前没有该商品，则直接添加一条新的记录，数量为 `counts`；如果有，则只将相应数量 `+ counts`，同时默认新加入的商品为已选中的状态。通过小程序内置函数 `wx.setStorageSync(this._key,data)`写入本地缓存。

（2）修改商品数量：在 `cart_model` 下创建 `changeCount` 方法，首先调用 `getCartDataFromLocal` 方法从本地缓存中获取全部数据。遍历全部数据找到 `id` 为所选商品 `id` 的数据，更改其 `counts` 属性，增加是传入`+1`，减少时传入`-1`。上述操作必须是在当 `counts` 大于 1 时进行，即每种商品的数

量至少为 1，最后将更改完的数据写入本地缓存。

（3）删除商品：在 `cart_model` 下创建 `delete` 方法，首先调用 `getCartDataFromLocal` 方法从本地缓存中获取全部数据。遍历全部数据找到 `id` 为所选商品 `id` 的数据，清除该数据后再将现有数据写入本地缓存。

（4）修改商品状态：在页面中选中或取消某商品时会触发 `toggleSelect` 方法，修改从缓存中得到的该商品的状态，随后重新渲染购物车页面。

（5）统计购物车总金额和总数量：在 `cart_model` 下创建 `getCartTotalCounts`，对本地缓存中的数据进行遍历，当某一商品已选中时即 `status` 为 `true` 时累加该商品的 `counts` 和该商品的 `counts*price`，得到购物车内所有已选中的商品总数和总价格。

### 4.3 其他功能实现

总体来说小程序的编写与前端网页的编写类似，通过 `.wxml`、`.wxss`、`.js` 等的编写实现各小程序各功能模块，不过一些 `js` 效果的实现可以在标签内部完成，比编写网站前端更为便捷。包括商品页面、分类页面、购物车页面、我的页面、主题商城页面、支付详情页面、商品信息页面等，其中 `js` 文件负责与各触发事件的实现和与服务端进行的数据交互，`json` 文件负责编写配置信息等。

其中一些特效的实现，如加入购物车时商品图标飞起，首页商品轮播图的实现等都是学习了网络上的小程序教程后实现的。其中一些页面也做成了模板，如 `catagory` 页面、`product` 页面。因为这些页面会在多个地方使用，做成模板后只需传入相应数据即可，不必每次都从头开始编写一个新的页面。



总之页面做的较为简单，各页面布局也够一目了然，仅能实现正常操作，没有添加太多功能模块。后期有需要的话可以在此基础上继续扩展，页面样式也可以继续完善。

## 4.4 后台网站设计

后台网站算是这三端中最容易实现的一端了，采用 CodeIgniter 框架实现，完全符合 MVC 设计模式<sup>[26]</sup>。主要架构如图 4.11。

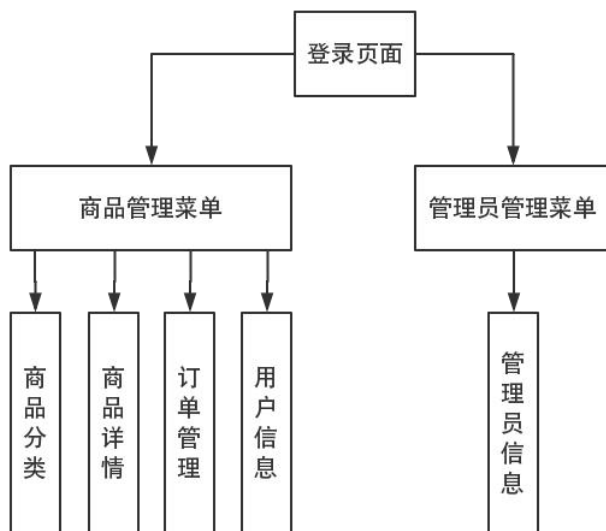


图 4.11 后台网站架构图

管理员登录后台管理网站后跳转为商品列表页面，可对商城中的在售商品，现有分类进行上架、下架、修改、添加操作，对订单和用户信息进行查看，对管理员列表进行增加、修改、删除、恢复操作。注意的是，超级管理员不可被删除。

### 4.4.1 登录页面

登录页面如图 4.12。分别在 `controllers`、`models`、`views` 下创建 `login.php`，`controllers\login` 负责对登录信息进行逻辑判断，`models\login` 负责从数据库 `t_shop_backend_user` 读写相应数据，`views\login` 负责构建登录页面。



图 4.12 登录页面

管理员在登录页面中输入用户名和密码,点击登录后数据会通过 `ajax` 提交至 `controllers\login` 中的 `checkLogin` 方法中,该方法对管理员输入的用户名和密码进行校验。将用户输入的密码进行 `md5` 处理后与存储在数据库的信息进行对比,相同后则将相关用户 `id`, 名称等添加至 `$_SESSION` 中,方便判断用户是否登录。随后将跳转至商品详情页面。

`controllers\login` 还提供了 `logout` 方法用于用户在登录后进行退出登录操作, `unset($_SESSION['user_name'])`, 跳转至登录界面。

#### 4.4.2 商品详情页面

商品详情页面如图 4-13。分别在 `controllers`、`models` 下创建 `product.php`, 在 `views\product` 下创建 `productList.php`、`productAdd.php`、`productEdit.php`, 分别对应于商品列表、添加、修改商品页面。`models\product` 负责从数据

表 t\_shop\_product、t\_shop\_image 读写相应数据。

添加商品 共37件商品, 上架35件商品, 下架2件商品

名称	价格	库存量	所属分类	图片	操作
荣耀V20	2999.00	100	荣耀		<a href="#">修改</a> <a href="#">下架</a>
荣耀10-青春版	0.01	984	荣耀		<a href="#">修改</a> <a href="#">下架</a>
荣耀20i	0.01	996	荣耀		<a href="#">修改</a> <a href="#">下架</a>
荣耀10	0.01	0	荣耀		<a href="#">修改</a> <a href="#">下架</a>

图 4.13 商品详情页面

信息

名称

价格

库存量

所属分类

荣耀

选择文件

未选择任何文件

确定

图 4.14 添加商品

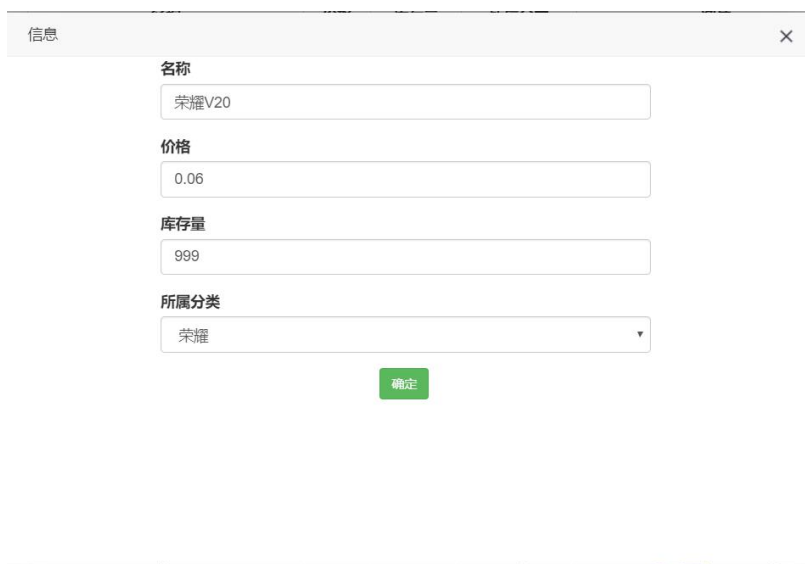


图 4.15 修改商品

controllers\product 下的 productList 方法负责对商品信息进行处理后在视图层展示；productAdd 方法负责对商品添加页面传递来的数据进行处理，并将上传来的商品图片拷贝至图片存储目录。写入 t\_shop\_image 和 t\_shop\_product 表；productDel 方法为某商品下架时调用，更新 t\_shop\_product 表中的 is\_delete 字段为 1，表示该商品已下架；productUp 方法为某商品上架时调用，更新 t\_shop\_product 表中的 is\_delete 字段为 0，表示该商品已上架；

#### 4.4.3 商品分类页面

商品分类页面如图 4.16。分别在 controllers、models 下创建 category.php，在 views\product 下创建 categoryList.php、categoryAdd.php、categoryEdit.php，分别对应于分类列表、添加、修改分类页面。models\product 负责从数据表 t\_shop\_category、t\_shop\_product 读写相应数据。

添加分类 共7种分类，上架5种分类，下架2种分类



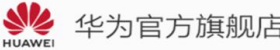



名称	商品数	图片	操作
荣耀	7		<a href="#">修改</a> <a href="#">下架</a>
小米	7		<a href="#">修改</a> <a href="#">下架</a>
华为	7		<a href="#">修改</a> <a href="#">下架</a>
Apple	7		<a href="#">修改</a> <a href="#">下架</a>
OPPO	7		<a href="#">修改</a> <a href="#">下架</a>
一加	0		<a href="#">修改</a> <a href="#">上架</a>

图 4.16 商品分类页面

信息

名称

选择文件

未选择任何文件

确定

图 4.17 添加分类

信息

名称:

确定

图 4.18 修改分类

controllers\category 下的 categoryList 方法负责对分类信息进行处理后在视图层展示；categoryAdd 方法负责对分类添加页面传递来的数据进行处理，并将上传来的分类图片拷贝至图片存储目录。写入 t\_shop\_image 和 t\_shop\_category 表；categoryDel 方法为某分类下架时调用，更新 t\_shop\_category 表中的 is\_delete 字段为 1，更新 t\_shop\_product 表中 category\_id 为当前所选分类 id 的所有数据的 is\_delete 字段为 1，表示该分类和分类下所属商品已下架；categoryUp 方法为某商品上架时调用，更新 t\_shop\_category 表中的 is\_delete 字段为 0，更新 t\_shop\_product 表中 category\_id 为当前所选分类 id 的所有数据的 is\_delete 字段为 0，表示该分类和分类下所属商品已上架。

#### 4.4.4 订单管理页面

订单页面如图 4.19。分别在 controllers、models 下创建 order.php，在 views\product 下创建 orderList.php、orderSee.php、orderSeeUser.php，分别对应于订单列表、查看订单、查看用户页面。models\order 负责从数据表 t\_shop\_order、t\_shop\_user\_address 读写相应数据。

共0个订单，已支付0个订单，未支付0个订单

用户名	创建时间	总价	状态	订单商品	订单名称	商品数量	更新时间	查看	
王可超	2019-06-08 13:51:00	0.02	未支付		荣耀10-青春版等	2	2019-06-08 13:51:00	<a href="#">查看订单</a>	<a href="#">查看用户</a>
王可超	2019-06-10 13:38:51	0.01	未支付		华为P30	1	2019-06-10 13:38:51	<a href="#">查看订单</a>	<a href="#">查看用户</a>
王可超	2019-06-10 13:50:40	6.70	未支付		荣耀V20	10	2019-06-10 13:50:40	<a href="#">查看订单</a>	<a href="#">查看用户</a>
王可超	2019-06-10 13:52:25	0.05	未支付		小米Mix3等	5	2019-06-10 13:52:25	<a href="#">查看订单</a>	<a href="#">查看用户</a>

图 4.19 订单管理页面

信息



商品名称	商品价格	商品数量	下单时是否有库存	商品总价	商品图片
荣耀V20	0.06	1	有库存	0.06	
OPPO K1	0.01	1	有库存	0.01	

图 4.20 订单详情

信息			
用户姓名	用户联系方式	用户地址	创建时间
张三	020-81167888	广东省广州市海珠区新港中路397号	2019-05-02 01:22:45

图 4.21 地址详情

controllers\order 下的 orderList 方法负责对订单信息进行处理后在视图层展示；orderSee 方法负责当管理员点击查看订单时对订单下商品信息进行处理后在视图层展示；orderSeeUser 方法负责当管理员点击查看用户时对订单所属用户信息进行处理后在视图层展示。

#### 4.4.5 用户信息页面

用户信息页面如图 4.22。分别在 controllers、models、views 下创建 user.php，models\order 负责从数据表 t\_shop\_user、t\_shop\_user\_address 读写相应数据；controllers\user 下的 userList 方法负责对用户信息进行处理后在视图层展示。

共1用户				
用户姓名	用户联系方式	用户地址	创建时间	更新时间
王可超	17864226336	山东省青岛市黄岛区山东科技大学学生公寓GB4(黄岛区)	2019-05-01 17:22:45	2019-05-23 09:36:41

图 4.22 用户信息页面

#### 4.4.6 管理员管理页面

管理员页面如图 4.23。分别在 controllers、models 下创建 admin.php，在 views\product 下创建 adminList.php、adminAdd.php、adminEdit.php，分别对应于管理员列表、添加、修改分类页面。models\product 负责从数据表 t\_shop\_backend\_user 读取相应数据。



添加管理员 共5名管理员，正常使用4名管理员，禁用1名管理员				
名称	创建时间	登录次数	上次登陆时间	操作
admin	2019-05-03 15:29:58	33	2019-06-13 05:53:29	修改
admin888	2019-05-03 17:10:41	0	1970-01-01 00:00:00	修改 删除
admin1	2019-05-03 17:12:11	1	2019-05-03 17:12:18	修改 恢复
wangkechao	2019-05-23 07:58:15	0	1970-01-01 00:00:00	修改 删除
hahaha	2019-06-11 02:35:17	0	1970-01-01 00:00:00	修改 删除

图 4.23 管理员页面

controllers\category 下的 adminList 方法负责对管理员信息进行处理后在视图层展示；adminAdd 方法负责对管理员添加页面传递来的数据进行处理，写入 t\_shop\_backend\_user 表；adminDel 方法为管理员删除时调用，更新 t\_shop\_backend\_user 表中的 is\_delete 字段为 1，表示该管理员已删除；adminUp 方法为管理员恢复时调用，更新 t\_shop\_backend\_user 表中的 is\_delete 字段为 0，表示该管理员已恢复使用。

## 第 5 章 系统接口调试

为了测试编写接口的准确性和返回数据的规范性，须通过类似 Postman 的接口测试软件进行测试，待返回数据满足客户端需要时即可在客户端内发起请求。

### 5.1 Postman 调试服务端接口

#### 5.1.1 Product/getOne

Product/getOne 返回数据如图 5.1。用户在小程序中点击某商品的图片时会调用该类的 getOne 方法，返回的数据包括商品 id，名称，价格，库存量，图片链接，商品属性和商品介绍的图片链接。其中商品属性和商品介绍为空是因为还未对该商品添加属性和介绍图片信息。

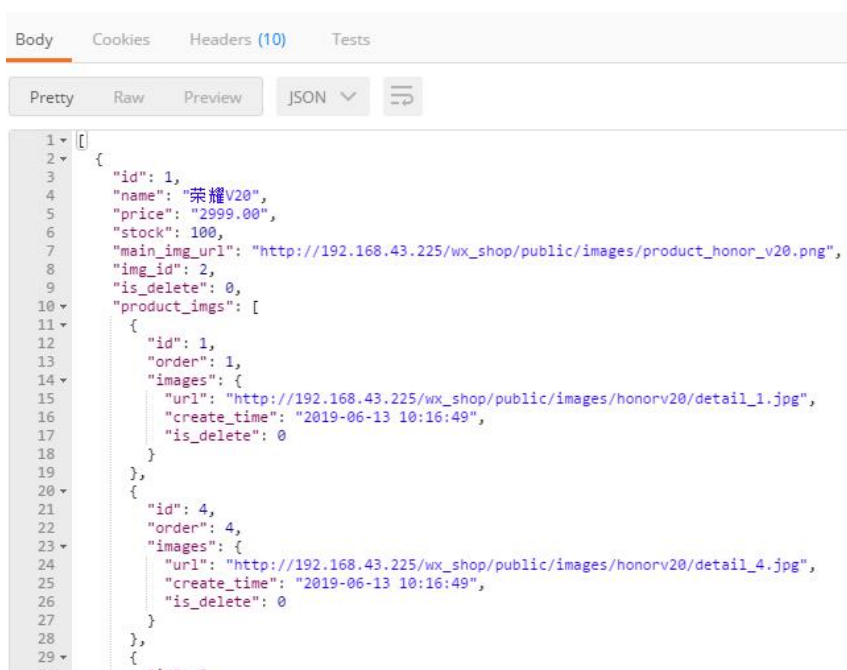


图 5.1 getOne 返回数据

### 5.1.2 Category/getAllCategories

Category/getAllCategories 返回数据如图 5.2。用户在小程序中的“分类”页面点击左侧分类 tab 栏时会调用该接口，该接口返回用户所选分类 id 下的所有的商品信息。包括每件商品的 ID、价格、名称、库存量、图片链接等信息。其中商品 ID 可用来当用户点击该分类下的某一商品时携带该商品 ID 调用 5.1.1 中介绍的接口，即可得到相应的商品信息。

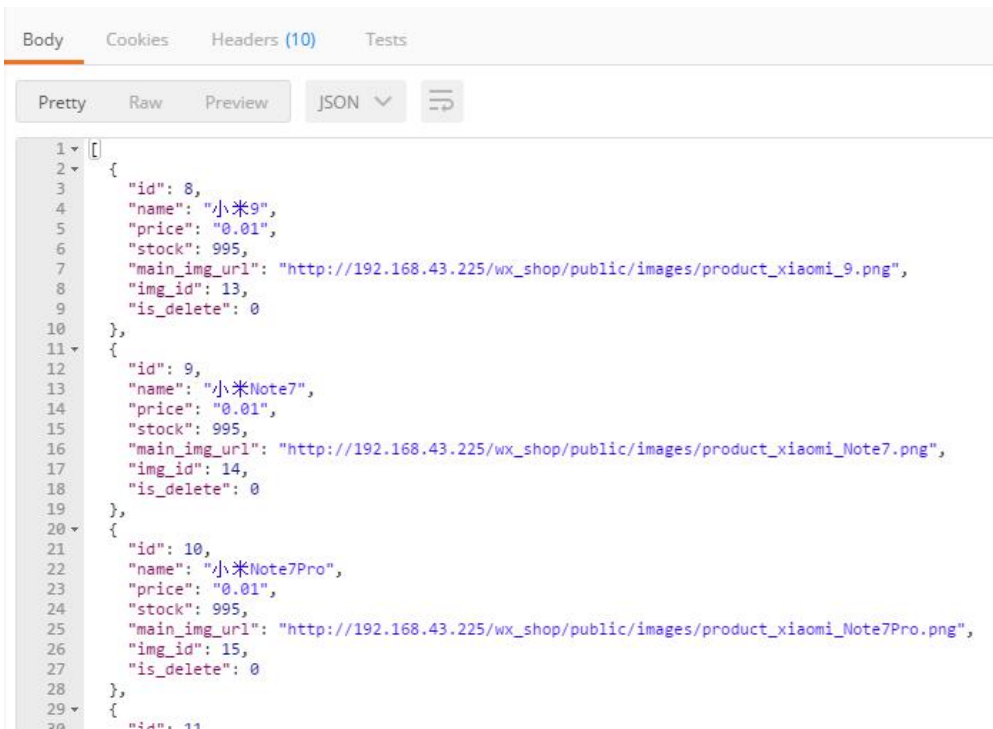


图 5.2 getCategoryes 返回数据

### 5.1.3Address/getAddress



图 5.3 getAddress 返回数据

Address/getAddress 返回数据如图 5.3。用户在进入我的页面时会请求该接口，如果用户地址信息已经存在则将用户地址信息返回给小程序，在“我的”页面进行展示。注意的是用户请求该接口需要携带 token 进行身份验证，如果 token 不存在或已过期则会出现下列提示信息，如图 5.4。



图 5.4 token 接口返回数据

#### 5.1.4 Token/verifyToken

Token/verifyToken 返回数据如图 5.5。该接口用于用户每次进入小程序时调用，用来验证用户 token 的有效性，通过返回 true/false 来通知客户端是否该向服务端的 generateToken 接口请求生成新的 token。其中服务端

缓存中的 token 是有两个小时的过期时间，当 token 过期或是被服务端清理数据时删除后，该接口即返回 false。

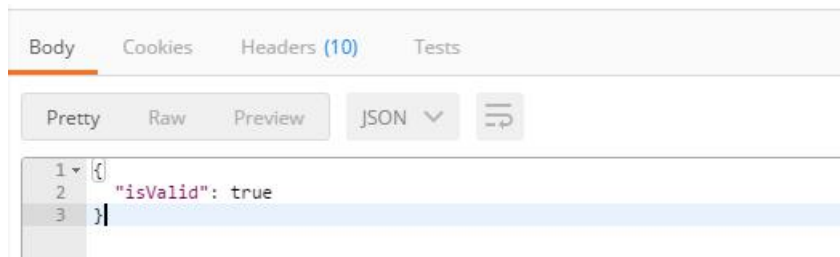


图 5.5 verifyToken 返回数据

### 5.1.5 Order/getDetailsByID

Order/getDetailsByID 返回数据如图 5.6。该接口为用户在“我的”页面点击某一订单信息时调用，用来在订单详情页面为用户展示详细的订单信息。返回信息包括订单号、订单总价、创建时间、订单状态、订单图片、订单名称、订单内的商品总数和订单下的每件商品的 ID、是否有库存、商品数量、商品名称、商品单价、商品总价（商品数量\*商品单价）、商品图片等信息，用以在订单详情页面为用户展示该订单下的商品列表。

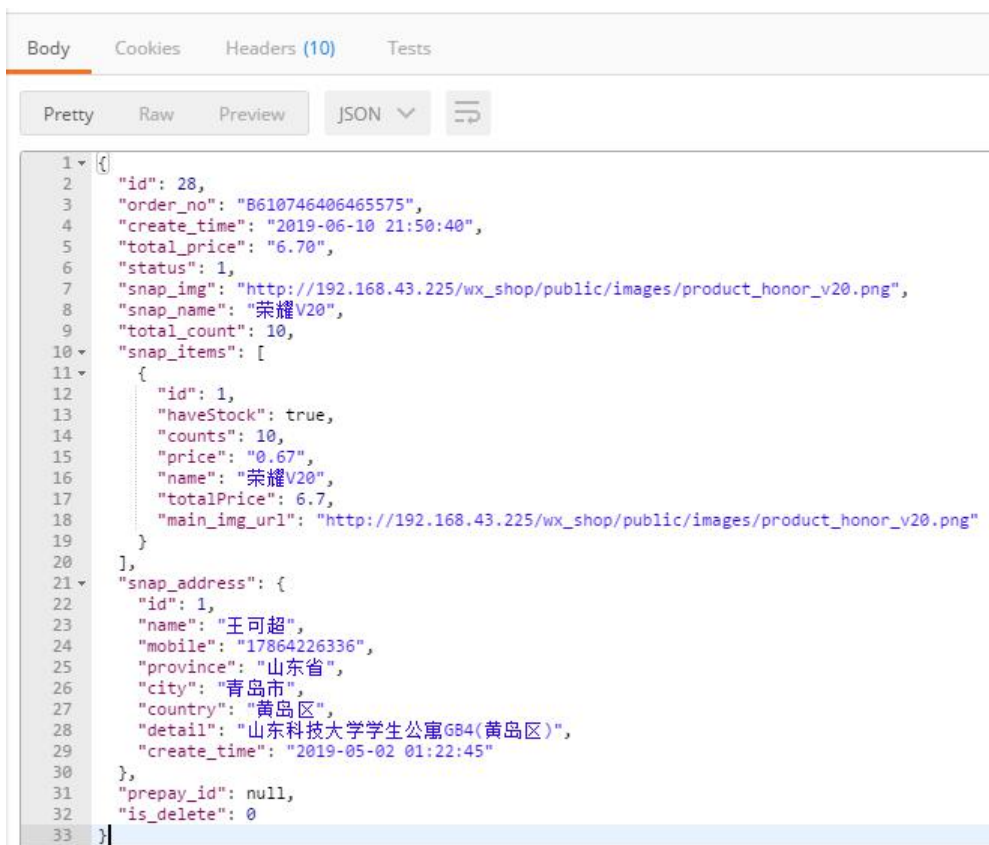


图 5.6 getDetailsByID 返回数据

## 5.2 完善小程序端显示界面

小程序页面主要是调试页面中各功能区的布局 and 页面颜色、toolbar 图标等信息，如首页中的布局经参考当今主流电商 APP 调整为现在的上方为轮播 banner 图；中间为主题商城入口，内含各大品牌的几款热卖商品；下方为“猜你喜欢”功能区，随机展示 20 款当前正在销售中的商品。对于小程序使用到的各 toolbar 图标，也是仿照各 APP 内 tab 点击图标进行选择。与各大电商 APP 的页面相比该页面虽然简单但却能满足用户的购物需要，没有眼花缭乱的特效图影响用户选购，也没有随处弹出的广告

弹窗扰乱用户心情，在简单中为用户营造出一种舒适便捷的购物环境。

商场的订单详情页面也是经过多次考虑才有了现在的布局，在保持页面简洁的同时尽量将订单内的所有商品信息进行详细展示。购物车页面也是如此，原来的增加和减少图标也改为现在的+/-号，删除按钮改为 x 号，尽可能的使页面看起来简洁。其中购物车中的多选操作和商品数量加减的实现较为困难，经过考虑后决定每次更新缓存后都对页面进行数据重载，最终解决了购物车的实现问题，空购物车页面如图 5.7。



图 5.7 空购物车页面

### 5.3 测试总结

以上的测试过程只占了整个测试流程的一部分，更详细的测试说明就不在此进行过多介绍了。通过 Postman 测试接口返回数据，在微信开发者工具中不断地调试小程序页面才有了现在的布局，后台的编写较为简单，直接在浏览器中 F12 调试即可。

总之，编写代码只占用了较少的时间，至少一多半的时间花在了前期准备和后期调试代码上。如验证接口返回数据的准确性，如何使小程序页面显示想要的数据，微信 SDK 如何使用等等的实现着实是花费了较长的时间。但是在调试的过程中也确实会学会很多东西，通过设置断点一步一步的走完数据处理的过程可以更好的理解各接口的实现流程，加深对项目的理解，为下一步的优化代码也做好了准备。



## 第 6 章 结论与展望

### 6.1 结论

在当今这个以用户为主的互联网时代中，更快的将产品上线，更方便的被用户浏览访问到，更精细的控制研发成本等等一系列因素都在制约着中小型企业步入电商行业。通过 PHP+小程序实现的这款电子商城则更完美的解决了这些问题，首先 PHP 入门较简单，适当学习就可以尝试接口的编写；TP5 框架是国人开发，使用说明文档也是中文编写，阅读和使用起来较为便捷；小程序依托微信平台，有着广泛的潜在用户群体等等，在有效的减少了企业的研发周期、成本的同时也解决了用户流量的获取来源，在避免用户下载更多 APP 的同时也提升了用户的购物体验。

本项目的架构采用了三端分离的方法，分为服务端、小程序端、后台管理端。

（1）服务端提供各类 http 接口，接口返回数据完全符合 Restful 规范。通过将 Model 层拆分，验证层和异常层独立，各类之间通过继承基类尽可能提高代码复用性和接口规范性。

（2）小程序端的页面设计尽量简洁，在满足用户正常使用的同时减少不必要的提示信息和过多的商品入口。具备添加、删除、增减、全选和部分选择等功能的购物车也会更好的提高用户的使用体验。

（3）后台管理网站中管理员可对分类、商品等进行添加、修改、上下架处理。但是对用户信息，订单信息等敏感信息则保留操作权限，为了更好地管理其他普通管理员设置的超级管理员 Admin 也可以使后台使用更加规范。

## 6.2 展望

本设计相较于各大互联网公司的移动端应用来说只能是一个简化、低配版。服务端的架构也没有考虑高并发的设计方案，数据库是单库单表，未分离读写操作，未构建从库保证数据备份。缓存的应用也十分有限，高并发下采用此架构势必会影响正常使用，甚至会使服务器宕机。因此在实际应用中可将服务端代码分离，封装成一个一个的微服务接口部署在多台服务器上，缓存可采用 Redis 来抗住用户频繁访问时造成的数据库压力，各接口内部添加日志入口，后期可通过查看相关日志解决运行中出现的问题。数据库也要主从分离，读写分离，防止数据库的意外宕机引起数据丢失的情况出现。千里之行始于足下，在将服务端各功能接口实现完毕之后各类优化也可以陆续进行。

## 参考文献

- [1] 李振航.中国电子商务的发展现状与未来趋势[J].中国新通信,2018,20(14):11.
- [2] 梅新蕾,荆兵.拼多多的崛起与未来[J].21 世纪商业评论,2019(04):12-19.
- [3] eMarketer.com, We Are Social & Hootsuite 《2017 年全球数字概貌》(Digital in 2017 Global Overview),2017
- [4] accenture 研究.2018 埃森哲中国消费者洞察——新消费 新力量[EB/OL].[https://www.accenture.com/\\_acnmedia/PDF-79/Accenture-Consumers-In-The-New-Full%20Report-Chinese#zoom=50](https://www.accenture.com/_acnmedia/PDF-79/Accenture-Consumers-In-The-New-Full%20Report-Chinese#zoom=50),2018
- [5] 罗改龙.微信公众平台营销策划与管理应用[J].计算机产品与流通,2019(05):138.
- [6] 尹冬梅.基于微信生态下的社交电商现状及发展研究[J].现代商业,2019(01):36-38.
- [7] 彭守镇.微信小程序应用探究[J].信息与电脑(理论版),2018(22):22-23.
- [8] ThinkPHP5.0 完全开发手册[EB/OL].
- [9] 微信小程序支付开发者文档.
- [10]<https://www.kancloud.cn/manual/thinkphp5/118003>,2016
- [11]CodeIgniter 用户指南.[http://codeigniter.org.cn/user\\_guide/](http://codeigniter.org.cn/user_guide/),2017
- [12]简书.史上最详细 Postman 教程.[EB/OL].
- [13]程杰.大话数据结构[M].北京:清华大学出版社,2011
- [14]王越.JAVA 编程语言在计算机软件开发中的应用[J].电子技术与软件工程,2019(01):35.
- [15]Rasmus Lerdorf.Inventing PHP[J].Computer 2012(11):45.
- [16]王欣.PHP 框架开发的关键技术研究[J].中国新技术新产

- 品,2018(16):32-33.
- [17]吕扬. “互联网+”背景下中小企业应用电子商务研究[J].中外企业家,2019(07):63.
- [18]熊普江, 谢宇华. 小程序, 巧应用:微信小程序开发实战(第2版)[M]. 机械工业出版社. 2017
- [19]王芳. 浅谈 windows 环境下 wampserver 的配置 [J]. 科技视界,2014(06):64.
- [20]侯璐璐.PHP 语言在企业网站开发中的应用[J].电子技术与软件工程,2018(20):2.
- [21]Rasmus Lerdorf,Kevin Tatroe 等. PHP 程序设计(第2版)[M], 电子工业出版社, 2009
- [22]上野宣. 图解 HTTP[M], 人民邮电出版社, 2014 年 5 月
- [23]Leonard Richardson, Mike Amundsen. RESTful Web APIs 中文版[M], 电子工业出版社, 2014
- [24]Luke Welling. PHP 和 MySQL Web 开发 (原书第4版)[M].机械工业出版社.2009 年 4 月
- [25]王珊, 萨师煊.数据库基础概论(第五版)[M].北京: 高等教育出版社, 2014 年 9 月
- [26]程杰. 大话设计模式[M].北京: 清华大学出版社, 2008

## 致 谢

毕业设计的完成,首先要感谢指导老师卫老师的悉心指导和不断督促。在我撰写论文的过程中,老师及时对论文的进度进行沟通,对我的疑难和困惑进行耐心的讲解,提出了很多关键性的改善意见。老师工作认真,对待学生十分负责,对我们无私的支持时刻激励着我。在生活上卫导师和蔼可亲、对我们关怀倍加,这更极大地鼓舞着我前进。在这里,衷心祝愿老师身体健康,工作顺利!

其次,我要感谢我的同窗舍友。在大学四年中,在生活上他们给了我无尽的包容和关怀,在学习上给了我无私的支持和帮助。最后,需要感谢我的家人和朋友,是你们的支持给了我不断奋斗的动力,感谢你们!

四年的大学时光转瞬即逝,刚踏入校园的日子仿佛就在昨天。感谢我深爱的学校----山东科技大学对我的培养,使我完成了本科学习。这段经历将是我人生中最重要的一段回忆。在这里,祝愿山东科技大学教学事业蒸蒸日上,培养出更多优秀的人才。

## 附录I 服务器接口源代码

### （1）product 接口

控制器：

```
<?php
namespace app\api\controller\v1;

use think\Controller;
use app\api\validate\Count;
use app\lib\exception\ProductException;
use app\api\model\Product as Product_model;

class Product extends Controller{

    public function getRandom($count=20){
        (new Count()->goCheck();
        $products = Product_model::getMostProduct($count);
        if(empty($products)){
            throw new ProductException();
        }
        foreach ($products as $key => $value) {
            $products[$key]['main_img_url'] =
config('setting.img_prefix').$value['main_img_url'];
        }
    }
}
```

```

        return $products;
    }

```

```

public function getProducts($id){
    $products = Product_model::getProductsByCategoryID($id);
    if($products->isEmpty()){
        throw new ProductException();
    }
    $products = $products->hidden(['summary']);
    return $products;
}

```

```

public function getOne($id){
    $product = Product_model::getProductDetailByID($id);
    if($product->isEmpty()){
        throw new ProductException();
    }
    $product = $product->hidden(['summary']);
    return $product;
}
}

```

**模型：**

```

<?php
namespace app\api\model;

```

```
use app\api\model\BaseModel;
```

```
class Product extends BaseModel{
```

```
    protected $hidden = [
```

```
        'create_time','update_time','pivot','delete_time','category_id','from'
```

```
    ];
```

```
    public function getMainImgUrlAttr($value,$data){
```

```
        return $this->prefixImgUrl($value,$data);
```

```
    }
```

```
    public static function getMostProduct($count){
```

```
        $id_array = array();
```

```
        $sql = "select count(id) as num from t_shop_product ";
```

```
        $max_num = self::query($sql)[0]['num'];
```

```
        for ($i = 0;count($id_array) < $count; $i++) {
```

```
            $rand_id = rand(0,$max_num - 1) + 1;
```

```
            if (!in_array($rand_id,$id_array)) {
```

```
                $id_array[] = $rand_id;
```

```
            }
```

```
        }
```

```
        $ids_str = implode($id_array,',');
```

```
        $sql = "select * from `t_shop_product` where `id` in ($ids_str) and  
is_delete = 0 ";
```



```

        $result = self::query($sql);

        return $result;
    }

```

```

    public static function getProductsByCategoryID($id){
        $result =
self::where('category_id','=', $id)->where('is_delete','=',0)->select();
        return $result;
    }

```

```

    public static function getProductDetailByID($id){
        //闭包 传入 query 对象 对关联模型中的元素进行排序
        $result = self::with([
            'productImgs'=>function($query){
                $query->with(['images'])->order('order asc');
            })
        ->with(['productProperty'])->select($id);
        return $result;
    }

```

//命名不加下划线

```

    public function productImgs(){
        //关联模型 外键 主键
        return $this->hasMany('ProductImage','product_id','id');
    }

```

```
public function productProperty(){  
    //关联模型 外键 主键  
    return $this->hasMany('ProductProperty','product_id','id');  
}  
}
```

## **Token 接口**

### **控制器:**

```
<?php  
namespace app\api\controller\v1;  
  
use think\Controller;  
use app\api\validate\GetToken;  
use app\api\validate\AppTokenGet;  
use app\api\service\UserToken;  
use app\api\service\Token as TokenService;  
use app\api\service\AppToken;  
use app\lib\exception\ParameterException;  
  
class Token extends Controller{  
  
    public function getToken($code=""){  
  
        (new GetToken())->goCheck();  

```

```

    $user = new UserToken($code);

    $token = $user->get();

    return [

        'token'=>$token

    ];

}

public function getAppToken($ac="", $se=""){

    $app = new AppToken();

    $token = $app->get($ac, $se);

    return [

        'token'=>$token

    ];

}

public function verifyToken($token="")

{

    if(!$token){

        throw new ParameterException([

            'token 不允许为空'

        ]);

    }

    $valid = TokenService::verifyToken($token);

    return [

        'isValid' => $valid
    ]
}

```

```
    ];  
    }  
}
```

**模型：**

```
<?php  
namespace app\api\service;  
use think\Request;  
use think\Cache;  
use app\lib\exception\TokenException;  
use app\lib\enum\ScopeEnum;  
use app\lib\exception\ForbiddenException;  
class Token{  
  
    public static function generateToken(){  
        $randChars = getRandChars(32);  
        $dateStamp = $_SERVER['REQUEST_TIME_FLOAT'];  
        $salt = config('secure.token_salt');  
        return md5($randChars.$dateStamp.$salt);  
    }  
  
    public static function getCurrentUid(){  
        $uid = self::getCurrentTokenVar('uid');  
        return $uid;  
    }  
}
```

```

public static function getCurrentTokenVar($key){
    $token = Request::instance()->header('token');
    $var = Cache::get($token);
    if(!$var){
        throw new TokenException();
    }else{
        if(!is_array($var)){
            $var = json_decode($var,true);
        }
        if(array_key_exists($key,$var)){
            return $var[$key];
        }else{
            throw new Excepeion('尝试获取的 Token 变量并不存在');
        }
    }
}

public static function isValidOperate($checkedUID){
    if(!$checkedUID){
        throw new Excepeion('检查时请传入 uid');
    }
    $currentUid = self::getCurrentUid();
    if($currentUid == $checkedUID){

```

```

        return true;
    }
    return false;

}

public static function verifyToken($token)
{
    $exist = Cache::get($token);
    if($exist){
        return true;
    }
    else{
        return false;
    }
}
}
<?php
namespace app\api\service;
use app\api\service\Token;
use think\Excepeion;
use app\lib\exception\WeChatException;
use app\api\model\User;
use think\Cache;

```

```

use app\lib\exception\TokenException;
use app\lib\enum\ScopeEnum;
class UserToken extends Token{

    protected $code;
    protected $appID;
    protected $appSercet;
    protected $loginUrl;

    public function __construct($code){
        $this->code = $code;
        $this->appID = config('wx.app_id');
        $this->appSercet = config('wx.app_secret');
        $this->loginUrl =
sprintf(config('wx.login_url'),$this->appID,$this->appSercet,$this->code);
    }
    public function get(){
        $result = curl_get($this->loginUrl);
        $wxResult = json_decode($result,true);
        if(empty($wxResult)){
            throw new Excepeion('获取 session_key 和 openId 异常，微信
内部错误');
        }else{
            $errCode = array_key_exists('errCode', $wxResult);

```

```

        if($errCode){
            $this->processLoginError($wxResult);
        }else{
            return $this->grantToken($wxResult);
        }
    }
}

//生成令牌
private function grantToken($wxResult){
    $openid = $wxResult['openid'];
    $user = User::getUserByOpenID($openid);
    if($user){
        $uid = $user->id;
    }else{
        $uid = $this->newUser($openid);
    }
    $cachedValue = $this->prepareCachedValue($wxResult,$uid);
    return $this->saveToCache($cachedValue);
}

private function saveToCache($cachedValue){
    $key = self::generateToken();
    $value = json_encode($cachedValue);
    $expire_in = config('setting.token_expire_in');
    $request = cache($key,$value,$expire_in);

```



```

        if(!$request){
            throw new TokenException([
                'msg'=>'缓存异常',
                'errorCode'=>10005
            ]);
        }
        return $key;
    }

    private function newUser($openid){
        $user = User::create([
            'openid'=>$openid
        ]);
        return $user->id;
    }

    private function prepareCachedValue($wxResult,$uid){
        $cachedValue = $wxResult;
        $cachedValue['uid'] = $uid;
        $cachedValue['scope'] = ScopeEnum::User;
        return $cachedValue;
    }

    private function processLoginError($wxResult){
        throw new WeChatException([

```

```
'msg'=> $wxResult['errmsg'],  
'errCode'=>$wxResult['errCode']  
]);  
  
}  
  
}
```

## 附录 II 小程序源代码

### （1）首页布局

```
<import src="../../tpls/products/products-tpl.wxml"/>
<view class="container home-container" hidden="{{!loadingHidden}}">
  <swiper indicator-dots="true" autoplay="true" class="swiper">
    <block wx:for="{{bannerArr}}">
      <swiper-item class="banner-item"
        bindtap="onProductsItemTap" data-id="{{item.key_word}}">
        <image class="item-image" src="{{item.img.url}}"
          mode="aspectFill" />
      </swiper-item>
    </block>
  </swiper>
  <view style='margin-top:30rpx;text-align: center;'>
    <image src="/imgs/images/honor20.jpg"
      style='height:80rpx'></image>
  </view>
  <view class="home-main">
    <view class="home-main-theme">
      <view class="home-main-header">主题商城</view>
      <view class="theme-box">
        <block wx:for="{{themeArr}}">
          <view class="theme-item"
            bindtap="onThemesItemTap" data-id="{{item.id}}"
```

```

data-name="{{item.name}}">
    <image
src="{{item.topic_img.url}}"></image>
    </view>
</block>
</view>
</view>
<block>
<view style='margin-top:30rpx'>
    <view class="home-main-header">优选入口</view>
    <view style='margin-left:20rpx'>
        <image src="/imgs/images/product_oppo_findx.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="32" bindtap="onProductsItemTap"></image>
        <image src="/imgs/images/product_iphone_X.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="24" bindtap="onProductsItemTap"></image>
        <image src="/imgs/images/product_huawei_P30.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="15" bindtap="onProductsItemTap"></image>
        <image src="/imgs/images/product_xiaomi_Mix3.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="12" bindtap="onProductsItemTap"></image>
    </view>

```

```
<view style='margin-left:20rpx'>
    <image src="/imgs/images/product_oppo_r17.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="33" bindtap="onProductsItemTap"></image>
    <image src="/imgs/images/product_iphone_XS.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="23" bindtap="onProductsItemTap"></image>
    <image src="/imgs/images/product_xiaomi_hand.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="14" bindtap="onProductsItemTap"></image>
    <image src="/imgs/images/product_iphone_iPadPro2018.png"
mode="aspectFill" style='height: 180rpx;width: 180rpx;border-radius:
180rpx;' data-id="28" bindtap="onProductsItemTap"></image>
</view>
</view>
</block>
<view class="home-main-products">
    <view class="home-main-header">猜你喜欢</view>
    <template is="products"
data="{{productsArr:productsArr}}"/>
</view>
</view>
<loading hidden="{{loadingHidden}}">
```

加载中...

</loading>

## （2）购物车页面

<view class="container cart-container">

<view style='text-align: center;'

<image src="/imgs/images/cart.jpg" style='height:200rpx'></image>

</view>

<block wx:if="{{cartData.length>0}}">

<view class="cart-box">

<block wx:for="{{cartData}}">

<view class="cart-item

{{deleteFlag&&index==currentIndex?'showDeleteBtn':'hideDeleteBtn'}}">

<view class="cart-item-main" data-id="{{item.id}}"

data-index="{{index}}">

<view class="cart-item-checkbox"

ontap="toggleSelect" data-id="{{item.id}}"

data-status="{{item.selectStatus}}">

<image wx:if="{{item.selectStatus}}"

src="../../../imgs/icon/circle@selected.png"></image>

<image wx:else

src="../../../imgs/icon/circle@noselected.png"></image>

</view>

<view class="cart-item-img"

bindtap="onProductsItemTap" data-id="{{item.id}}">

```
<image class="good-image"
src="{{item.main_img_url}}"></image>
</view>
<view class="cart-item-word">
  <view class="title-box">
    <text
class="title">{{item.name}}</text>
    <text>¥ {{item.price}}</text>
  </view>
  <view class="bottom-box">
    <view class="cart-item-counts">
      <view class="btns
{{item.counts==1?'disabled':''}}" bindtap="changeCounts"
data-id="{{item.id}}" data-type="cut">-</view>
      <view
class="counts">{{item.counts}}</view>
      <view class="btns"
bindtap="changeCounts" data-id="{{item.id}}" data-type="add">+</view>
    </view>
    <view class="delete"
data-id="{{item.id}}" bindtap="delete">×</view>
  </view>
</view>
</view>
```

```

        </view>
    </block>
</view>
<view class="footer-account-box all-accounts-box">
    <view class="all-select" ontap="toggleSelectAll"
data-status="{{selectedTypeCounts==cartData.length?'true':'false'}}">
        <image
wx:if="{{selectedTypeCounts==cartData.length}}"
            class="title-icon"
src="../../imgs/icon/all@selected.png"></image>
        <image wx:else class="title-icon"
src="../../imgs/icon/all.png"></image>
        <text>全选({{selectedCounts}})</text>
    </view>
    <view class="all-price-submit {{account==0?'disabled':''}}"
bindtap="submitOrder">
        <view class="accounts-btn">下单</view>
        <view class="price-text">¥ {{account}}</view>
        <view class="arrow-icon">
            <image wx:if="{{account==0}}"
src="../../imgs/icon/arrow@grey.png"></image>
            <image wx:else
src="../../imgs/icon/arrow.png"></image>
        </view>
    </view>

```



```
</view>
</view>
</block>
<view wx:else class="no-data" style='margin-top:300rpx'>
  <view>
    <image src="/imgs/images/cart_null.jpg"
style='height:500rpx'></image>
    <view style='margin-left:250rpx'>您还没有添加任何商品
  </view>
  </view>
</view>
<loading hidden="{{loadingHidden}}">
  加载中...
</loading>
</view>
```

## 附录 III 后台网站源代码

### (1) Login

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
```

```
class Login extends My_Controller
```

```
{
```

```
    public function __construct()
```

```
    {
```

```
        parent::__construct();
```

```
        $this->load->model('Category_model');
```

```
        $this->load->model('Product_model');
```

```
        $this->load->helper('url');
```

```
        $this->load->library('session');
```

```
    }
```

```
    public function loginIn(){
```

```
        $data['base_url'] = $this->config->config['base_url'];
```

```
        $this->load->view('header',$data);
```

```
        $this->load->view('login/loginIn');
```

```
        $this->load->view('footer');
```

```
    }
```

```
    public function checkLogin(){
```

```
        $active = $this->input->post('active',true);
```

```

if ($active == 1) {
    $username = $this->input->post('username',true);
    $password = $this->input->post('password',true);
    if (empty($username) || empty($password)){
        echo '<script language="JavaScript">alert("用户名和密码不能为空");window.location.href="/loginIn";</script>';
        return;
    }
    $where = " user_name = '$username' ";
    $user_info =
$this->Product_model->getMysqlSingleData('t_shop_backend_user',$where);
    if (!$user_info) {
        echo '<script language="JavaScript">alert("该用户不存在");window.location.href="/loginIn";</script>';
        return;
    }
    $user_pwd = $user_info['user_pwd'];
    if (md5($password) == $user_pwd) {
        $supt_array = array(
            array(
                'id' => $user_info['id'],
                'login_count' => $user_info['login_count'] + 1,
                'login_time' => time()
            )
        )
    }
}

```

```

    );

    $info =

$this->Product_model->uptMysqlData('t_shop_backend_user',$supt_array,'id');

    if (!$info) {

        echo '<script language="JavaScript">alert("操作失
败");window.location.href="/loginIn";</script>';

        return;

    }

    $_SESSION['user_name'] = $user_info['user_name'];

    redirect($this->config->config['base_url'].'/product/productList');

    } else {

        echo '<script language="JavaScript">alert("密码输入错
误");window.location.href="/loginIn";</script>';

        return;

    }

    redirect($this->config->config['base_url'].'/login/loginIn');

    }

    echo '<script language="JavaScript">alert("用户名和密码不能为
空");window.location.href="/loginIn";</script>';

    return;

}

```

```

        public function logout(){
            unset($_SESSION['user_name']);
            redirect($this->config->config['base_url'].'/login/loginIn');
        }
    }
}

```

## (2) Admin

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
```

```
class Admin extends My_Controller
```

```

{
    public function __construct()
    {
        parent::__construct();
        $this->load->model('Category_model');
        $this->load->model('Product_model');
    }

    public function adminList(){
        $this->isLogin();
        $result =
        $this->Product_model->getMysqlBatchData('t_shop_backend_user',' 1=1 ');
        $total['total'] =
        $this->Product_model->getMysqlBatchData('t_shop_backend_user',' 1=1

```

```

,true);

$total['up'] =
$this->Product_model->getMysqlBatchData('t_shop_backend_user',' is_delete
= 0 ',true);

$total['down'] =
$this->Product_model->getMysqlBatchData('t_shop_backend_user',' is_delete
= 1 ',true);

foreach ($result as $key => $value) {
    $result[$key]['login_time'] = date('Y-m-d
H:i:s',$value['login_time']);
    $result[$key]['create_time'] = date('Y-m-d
H:i:s',$value['create_time']);
}

$data['base_url'] = $this->config->config['base_url'];
$data['data'] = $result;
$data['total'] = $total;
$this->load->view('header',$data);
$this->load->view('admin/adminList');
$this->load->view('footer');
}

public function adminEdit(){
    $active = $this->input->get('active',true);
    if (!$active) {

```

```

        $active = $this->input->post('active',true);
    }
    $info = array();
    if ($active == 1) {
        $id = $this->input->get('id',true);
        if (empty($id)) {
            echo json_encode(
                array(
                    'data'=>",
                    'desc'=>'参数错误',
                    'status'=>101
                )
            );
            return;
        }
        $where = " id = $id ";
        $info =
$this->Product_model->getMysqlSingleData('t_shop_backend_user',$where);
    }
    if ($active == 2) {
        $id = $this->input->post('id',true);
        $admin_name = $this->input->post('admin_name',true);

        if (empty($id) || empty($admin_name)) {

```

```
        echo json_encode(
            array(
                'data'=>",
                'desc'=>'参数错误',
                'status'=>101
            )
        );
        return;
    }
    $supt_array = array(
        array(
            'id' => $id,
            'user_name' => $admin_name,
        )
    );
    $info =
    $this->Product_model->uptMysqlData('t_shop_backend_user',$supt_array,'id');

    if (!$info) {
        $sarr = array(
            'data'=>",
            'desc'=>'修改失败',
            'status'=>105
        );
    }
```



```

        echo json_encode($arr);

        return;
    }

    $arr = array(
        "data" => "",
        "desc" => "修改成功",
        "status" => 200
    );

    echo json_encode($arr, JSON_UNESCAPED_UNICODE);

    return;
}

$data['base_url'] = $this->config->config['base_url'];
$data['admin_info'] = $info;
$this->load->view('header', $data);
$this->load->view('admin/adminEdit');
$this->load->view('footer');
}

public function adminDel(){
    $active = $this->input->post('active', true);
    if ($active == 1) {
        $id = $this->input->post('id', true);
        if (empty($id)) {
            echo json_encode(

```

```
        array(
            'data'=>",
            'desc'=>'参数错误',
            'status'=>101
        )
    );
    return;
}
$upt_array = array(
    array(
        'id' => $id,
        'is_delete' => 1
    )
);
$info =
$this->Product_model->uptMysqlData('t_shop_backend_user',$upt_array,'id');

if (!$info) {
    $arr = array(
        'data'=>",
        'desc'=>'操作失败',
        'status'=>105
    );
    echo json_encode($arr);
}
```

```

        return;
    }
    $arr = array(
        "data" => "",
        "desc" => "操作成功",
        "status" => 200
    );
    echo json_encode($arr, JSON_UNESCAPED_UNICODE);
    return;
}
return;
}

```

```

public function adminUp(){
    $active = $this->input->post('active',true);
    if ($active == 1) {
        $id = $this->input->post('id',true);
        if (empty($id)) {
            echo json_encode(
                array(
                    'data' => "",
                    'desc' => '参数错误',
                    'status' => 101
                )
            );
        }
    }
}

```

```

        );

        return;
    }

    $supt_array = array(
        array(
            'id' => $id,
            'is_delete' => 0
        )
    );

    $info =

$this->Product_model->uptMysqlData('t_shop_backend_user',$supt_array,'id');

    if (!$info) {
        $sarr = array(
            'data'=>",
            'desc'=>'操作失败',
            'status'=>105
        );
        echo json_encode($sarr);
        return;
    }

    $sarr = array(
        "data" =>"",
        "desc"=>"操作成功",
    
```

```

        "status"=>200

    );

    echo json_encode($arr,JSON_UNESCAPED_UNICODE);

    return;

}

return;

}

public function adminAdd(){

    $active = $this->input->post('active',true);

    if ($active == 1) {

        $admin_name = $this->input->post('admin_name',true);

        $admin_pwd = $this->input->post('admin_pwd',true);

        if (empty($admin_name) || empty($admin_pwd)) {

            echo json_encode(

                array(

                    'data'=>",

                    'desc'=>'参数错误',

                    'status'=>101

                )

            );

            return;

        }

        $ins_array = array(

```

```

        array(
            'user_name' => $admin_name,
            'user_pwd' => md5($admin_pwd),
            'create_time' => time()
        )
    );
    $info =
$this->Product_model->insMysqlData('t_shop_backend_user',$ins_array);
    if (!$info) {
        $arr = array(
            'data'=>",
            'desc'=>'操作失败',
            'status'=>105
        );
        echo json_encode($arr);
        return;
    }
    $arr = array(
        "data" =>"",
        "desc"=>"操作成功",
        "status"=>200
    );
    echo json_encode($arr,JSON_UNESCAPED_UNICODE);
    return;

```

```
    }  
    $data['base_url'] = $this->config->config['base_url'];  
    $this->load->view('header',$data);  
    $this->load->view('admin/adminAdd');  
    $this->load->view('footer');  
}  
  
}
```

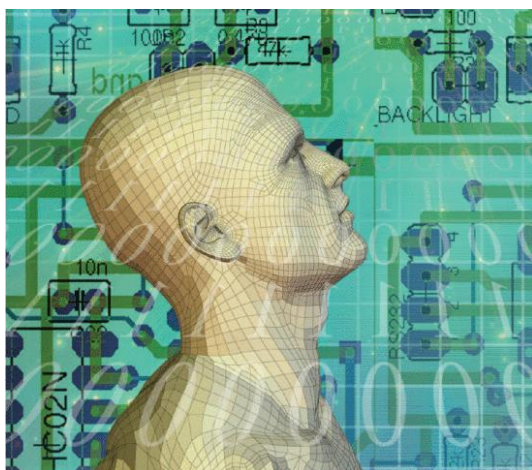
## 附录 IV 外文文献及翻译

外文原文:

### Inventing PHP

Rasmus Lerdorf

16/11/2012



Originally conceived as an HTML templating language, Hypertext Preprocessor didn't start its life as a pure programming language.

PHP, which originally stood for Personal Home Page, has been officially renamed "PHP: Hypertext Preprocessor". Note that it is not an abbreviation for "Hypertext Preprocessor"; this way of putting a name into a definition is called a recursive abbreviation. PHP was founded in 1994 by Rasmus Lerdorf as a simple Perl program designed by Rasmus Lerdorf to maintain personal web pages. These tools are used to display Rasmus Lerdorf's personal history and to count web traffic. It was later rewritten in C, including access to databases. He



integrated these programs with some form transliterators called PHP/FI. PHP/FI can connect to databases and generate simple dynamic web applications.

In 1995, when the first version was published as Personal Home Page Tools (PHP Tools), Lerdorf wrote some documentation about the program. And released PHP1.0! In this version, provides the visitor message book, visitor counter and other simple functions. In the future, more and more websites are using PHP, and there is a strong need to add some features. Loop statements and array variables and so on; After new members joined the development team, Rasmus Lerdorf made PHP/FI public on June 8, 1995, hoping to use the community to speed up program development and find bugs. This release, called PHP 2, already has some prototypes of PHP, such as perl-like variable naming, form-handling capabilities, and the ability to execute embedded in HTML. Program syntax is similar to Perl, with more limitations, but simpler and more flexible. PHP/FI adds support for MySQL, establishing PHP's place in dynamic web development. By the end of 1996, there were 15,000 websites using PHP/FI.

I was living in Toronto and doing Web application consulting for a number of companies. I wrote the same code over and over—basically, CGI [Common Gateway Interface] scripts written in C. I wrote code to handle forms, POST data, filtering, and other common Web things that you have to write in C when you're writing CGI programs. It was kind of tedious and boring, so if I could reduce the amount of time I had to

spend programming, maximize the output, and get to the solution quicker, then that was my goal with PHP. I put all my common stuff into a C library, hacked it into the NCSA [National Center for Computing Applications] webserver, and then added a templating system on top of it to let me easily call into it.

The first version of PHP was simply a productivity tool that enabled Lerdorf to accelerate his development across his multiple clients who needed Web applications. PHP was quickly embraced by other Web developers, who continue to build on and improve it.

### **Humble Beginnings**

In the Web's early days, the developer community was small, so it didn't take long for Lerdorf's colleagues to find out about his software and start asking for copies for their own clients:

Other people started asking me how I built these applications, and I said I was using this little tool I built. They asked if they could have it, and I said, “Sure, why not?” My toolkit wasn't what I was selling—I was selling my services of solving problems, and the tool itself is irrelevant, really. It's just my hammer.

After other programmers started using it seriously, they found bugs, fixed them, and sent him patches. Using these patches, he modified his utility library and templating engine and improved the applications he was building for his customers:

That's when open source really hit me. This was in 1994–1995 before the term “open source” existed. I got together with a group of

my peers, other people interested in the Web and solving the Web problem from all around the world. We all faced similar issues and collaboratively we could build a tool that solved our problem. That was really how PHP got off the ground.

Because PHP was initially conceived as a collection of library utilities rather than as a new programming language, Lerdorf never felt the need to shape its future direction. He felt PHP would thrive if he opened the code base to other people and approaches:

I learned a bit along the way that, for this to grow, I had to give up control of PHP—I had to let other people have some control. I couldn't rewrite patches, both because I'm lazy and it's a lot of work and also to give people some ownership. Once they have full control over their part of it, then they become much more invested in it and passionate. It's not just them contributing to my project—it becomes our project, and that really changed the nature of PHP. This happened around 1997 or so, when I really delegated it out and gave people full access to the source code repository that I was using.

Once Lerdorf allowed other people to become involved in PHP's evolution, he quickly built a large following around the product:

The Web grew, and PHP was at the right place at the right time. But also, it was very, very easy to get in and get started using PHP and contributing to it. Even today, it doesn't take much to get a source code repository account in the PHP project. We have close to 1,400 people with accounts, which means those people can all commit to some part

of the repository. Slightly more than half the people have committed something in the last year and a half.

The only way to manage all those volunteers is to let them manage themselves. Within the PHP community, many small, dedicated groups work closely together and focus on one aspect of PHP and collectively own it. Lerdorf prefers to let passionate volunteers move forward, even if they make little mistakes that need to be fixed later after their contributions are reviewed by more experienced members of the community.

### **Crowdsourcing**

Through the PHP Extensions Community Library (PECL), interested groups of volunteers can incubate an idea and then build interest in their feature. Once a feature is in broad use, it can become part of the core distribution, such as the JSON extension in PHP 5.1:

That's how new features eventually creep in—they live outside of the core tree, get enough penetration and enough people to install them, and then we see Linux distributions pulling them into their core version of PHP. We look at what's happening out there, but there's no real management of that either.

In many open source projects, an individual or small group controls the project's architectural direction to ensure consistency across the product. Lerdorf even leaves architectural decisions about PHP to the community:

It's a meritocracy. Code speaks. PHP is a scripting language

embedded in HTML and interpreted by the server. It can be used to manage dynamic content, support database, process session tracking, and even build the entire e-commerce site. It supports many popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix and Microsoft SQL Server. If you write a patch or a piece of code to implement a feature, that says a lot. If someone wants to disagree with that way of doing things, or if they can offer an alternative implementation, that's a really good argument. If all they do is whine about it, that's a really bad argument, and chances are, the implementation will win even though it might not be the best way of doing things. If there's code and it sort of works, that's what we go with, and that has always been the default. It doesn't always lead to consistency, but it does lead to getting new features and actually being able to do something. Being able to connect to this type of database even though it might not be the best way of doing it, at least it gets you there. That's what PHP has always been about—solving a problem. We would rather have an ugly feature than not have a feature at all.

As a small open source, PHP has gradually developed as more and more people realize its practicability. Rasmus Lerdorf released the first version of PHP in 1994. Since then, it has developed rapidly, and has been improved and perfected by countless ways in the original distribution. When I asked Lerdorf about PHP's future roadmap, his answer was that it would match the Web's evolution. As the Web moves into new areas and uses new technologies, PHP needs to make those

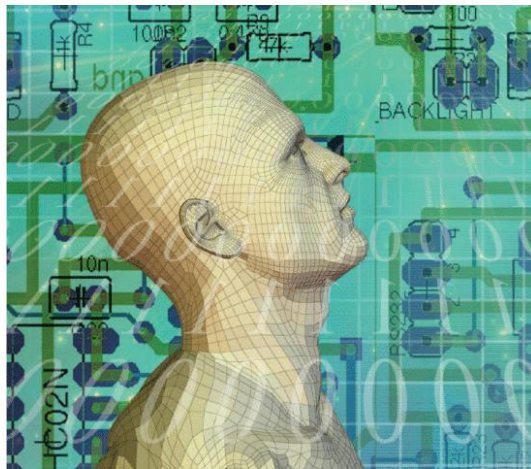
new technologies and approaches available to PHP developers. There's no master plan except to be useful to people developing Web applications.

译文：

## 发明 PHP

拉斯马斯·勒德尔夫

2012 年 11 月 16 日



PHP 最初被认为是一种 HTML 模板语言，但它并没有以纯粹的编程语言开始它的生命历程。

PHP 原始为 Personal Home Page 的缩写，已经正式更名为 "PHP: Hypertext Preprocessor"。注意不是 "Hypertext Preprocessor" 的缩写，这种将名称放到定义中的写法被称作递归缩写。PHP 于 1994 年由 Rasmus Lerdorf 创建，刚刚开始是 Rasmus Lerdorf 为了要维护个人网页而制作的一个简单的用 Perl 语言编写的程序。这些工具程序用来显示 Rasmus Lerdorf 的个人履历，以及统计网页流量。后来又用 C 语言重新编写，包括可以访问数据库。他将这些程序和一些表单直译器整合起来，称为 PHP/FI。PHP/FI 可以和数据库连接，产生简单的动态网页程序。

在 1995 年以 Personal Home Page Tools (PHP Tools) 开始对外发表第

一个版本，Lerdorf 写了一些介绍此程序的文档。并且发布了 PHP1.0！在这的版本中，提供了访客留言本、访客计数器等简单的功能。以后越来越多的网站使用了 PHP，并且强烈要求增加一些特性。比如循环语句和数组变量等等；在新的成员加入开发行列之后，Rasmus Lerdorf 在 1995 年 6 月 8 日将 PHP/FI 公开发布，希望可以透过社群来加速程序开发与寻找错误。这个发布的版本命名为 PHP 2，已经有 PHP 的一些雏型，像是类似 Perl 的变量命名方式、表单处理功能、以及嵌入到 HTML 中执行的能力。程序语法上也类似 Perl，有较多的限制，不过更简单、更有弹性。PHP/FI 加入了对 MySQL 的支持，从此建立了 PHP 在动态网页开发上的地位。到了 1996 年底，有 15000 个网站使用 PHP/FI。

**Lerdorf:** 我住在多伦多，为很多公司做网络应用咨询。我一遍又一遍地写同样的代码，基本上是用 C 编写的 CGI[公共网关接口]脚本。我编写代码来处理表单、发布数据、过滤和用 C 编写一些 CGI 程序等常见的 Web 应用。这些重复性的工作不免有点单调乏味，所以为了能够减少花在编程上的时间，争取最大化输出，更快地得到解决方案，我开发了 PHP，这也就是我使用 PHP 的目的和初衷。当时我把所有常用的插件、方法放进了一个 C 语言库，把它放在了 NCSA（国家计算应用中心）的 Web 服务器上，然后在上面添加了一个模板系统，通过调用它可以让我可以很容易地实现之前做的日常工作。

PHP 的第一个版本只是一个用来生产的工具，它使 Lerdorf 能够在为多个客户机搭建 Web 应用程序时的开发更简单便捷。正因这种快速简单，PHP 很快就被其他 Web 开发人员所接受，他们继续构建和改进它。

### 卑微的出身

在 Web 开始大规模应用的早期，Web 开发人员社区很小，所以 Lerdorf



的同事很快就发现了他的软件，他们想使用 Lerdorf 开发的 PHP 工具包来为自己的客户开发 web 应用，于是想从 Lerdorf 手中拿到源码：

Lerdorf：其他同事开始问我如何构建的这些 web 应用程序，我说是通过我正在使用的这个用 C 编写的小工具。他们问他们能否也在开发时使用它，我说，当然，为什么不呢？我开发的 PHP 工具箱并不是我要用来卖的东西——我卖的是解决问题的服务，这和 PHP 本身是不相关的。PHP 只是我用来开发工具。

在其他程序员开始真正使用 PHP 之后，他们修复了已发现的错误，向 Lerdorf 发送了补丁。通过这些补丁，Lerdorf 修改了实用程序库和模板引擎，并改进了为客户构建的应用程序：

“这就是开源真正影响我的时候。在 1994-1995 年“开源”一词出现之前，我和一群同龄人，还有一些对互联网感兴趣的人，以及解决来自世界各地的网络问题的人聚在一起。我们都面临着类似的问题，因此可以协作构建一个解决问题的工具。这就是 PHP 如何摆脱困境的原因。”Lerdorf 说。

因为 PHP 最初被设想为实用程序的集合库，而不是一种新的编程语言，所以 Lerdorf 从未觉得有必要确定它的未来方向。他认为，如果向其他人和方法开放代码库，PHP 将会蓬勃发展：

Lerdorf：在这个过程中，我学到了一些东西，为了让它成长，我必须放弃对 PHP 的控制——我必须让其他人拥有一些控制权。我不能重写补丁，因为我很懒，需要做很多工作，而且要给人们一些所有权。一旦他们完全控制了自己的那部分，他们就会变得更加投入和热情。不仅仅是他们对我的项目做出了贡献，而是成为了我们的项目，这真的改变了 PHP 的本质。这大约发生在 1997 年左右，当时我把它委托出去，让人们完全

访问我正在使用的源代码存储库。

一旦 Lerdorf 允许其他人参与到 PHP 的发展中，他很快就围绕着这个产品建立了大量的追随者：

Web 不断发展，PHP 在正确的时间出现在正确的地点。而且，开始使用 PHP 并为之做出贡献是非常非常容易的。即使在今天，在 PHP 项目中获得源代码存储库帐户也不需要花费太多。我们有近 1400 人拥有帐户，这意味着这些人可以提交到存储库的某个部分。略多于一半的人在过去的一年半里做了一些事情。

管理这些志愿者的唯一方法就是让他们自己管理自己。在 PHP 社区中，许多小型、专门的小组紧密合作，专注于 PHP 的一个方面并共同拥有它。勒多夫更愿意让热情的志愿者继续前进，即使他们犯了一些小错误，这些错误需要在社区中更有经验的成员对他们的贡献进行审查后加以纠正。

## 众包

通过 PHP Extensions Community Library (PECL)，感兴趣的志愿者小组可以酝酿一个想法，然后对他们的特性产生兴趣。一旦一个特性被广泛使用，它就可以成为核心发行版的一部分，比如 PHP 5.1 中的 JSON 扩展：

这就是新特性最终蔓延的方式——它们位于核心树之外，获得足够的领域渗透和足够的人员来安装它们，然后我们看到 Linux 发行版引入了核心版本的 PHP。但我们并没有采取真正的管理。

在许多开源项目中，个人或小组控制项目的体系结构方向，以确保整个产品的一致性。Lerdorf 甚至将 PHP 的架构决策留给了社区：

这是一种精英制度，代码说话。PHP 是一种嵌入在 HTML 并由服务器解释的脚本语言。它可以用于管理动态内容、支持数据库、处理会话

跟踪，甚至构建整个电子商务站点。它支持许多流行的数据库，包括 MySQL、PostgreSQL、Oracle、Sybase、Informix 和 Microsoft SQL Server。如果你写了一个补丁或者一段代码来实现一个特性，那就说明你做了很多，你的决策权威也就越大。如果有人不同意这种做事方式，那更好的理由就是他们可以提供一個替代的实现方式。如果他们所做的只是抱怨，那他们发出的只会是一些糟糕的观点。即使现在实现的方式可能不是最好的方式，但它可能依然会被大家采纳。因为只要有一段代码可以正常工作的话，那就是我们需要的，这个观点在社区内也一直是大家默认采纳的。虽然这种做法会导致不一致性出现，但它确实会为 PHP 获得一些新的特性，并能够为使用者实际增加一些用处。例如一些插件能够连接、操作某种类型的数据库，即使这款插件可能采用的不是最好的实现方法，至少它能让您正常使用。而这就是 PHP 一直致力于解决问题的原因。我们宁愿有一个不算完美的功能块，也不愿没有任何功能块。

当我问勒道夫关于 PHP 未来路线图的问题时，他的回答是它将与 Web 的发展相匹配。随着 Web 进入新领域并且许许多多的新技术开始出现并应用，PHP 需要使这些新技术和方法为 PHP 开发人员可用。当然，除了对开发 Web 应用程序的人员有用之外，PHP 并没有其他的发展计划。