# 前言

可能有同学已经看过ppt,知道在这里我们要介绍一个电商项目，这电商项目根据名字都知道是在客户可以在网上购买商品，商家进行发货的项目

对于使用方来说，一个是商家，另外一个是购买商品的客户。

# 脱离程序员的视野看项目

如果说一个人的成功由两部分组成，智商和情商，我们把这个嫁接在项目上一个项目的成功语法也可以说他由两部分组成

智商：既项目的架构，所用到的技术，解决得各种问题，技术的可扩展性等

情商：用一个专业的词语概况：“商业模式”，既解决“客户从哪来”，“怎么让他成交”，“怎么让他重复成交”等等问题。

“商业模式”说得牛逼哄哄，其实就是一个蓝图，这个蓝图需要落地，需要靠我们程序员设计和技术功底。

这块很多程序员工作了10年都不会去考虑的，恰恰是个高级架构，或者说以后想成为创业者需要考虑的问题。

程序，代码，架构是骨架，而一个商业模式是项目的真正灵魂，一个人能活着需要血液不停得流动，还让一个项目长盛不衰得也要靠一个东西得流动，那个东西就是现金流。

其实老板也好，程序员也好，需要搞定的一个问题，“钱”的问题。

## 用“钱”的角度看商业模式

回想每天发生在你身边的各种买卖，每天你的身份都在不停的发生转变，某一时刻你是消费者，而某一个时刻你又是服务提供者

我们一直在讨论电商项目，想成为BATJ的架构师，那我们来看下一个电商项目的商业模式的灵魂。

1.就某宝为例，你以前不知道某宝，突然看到广告，或者看到身边有人都在使用，你注册了一个某宝

2.一个活动，发现自己很有兴趣的东西，下了一个订单

3.根据你的浏览，订单情况，然后各种推荐，各种活动（双11等），你已经养成了没事逛某宝的习惯

用专业产品经理的角度来看这个过程其实由3部分组成



抓潜：重点思考的一个事情，客户从哪来，客户聚集的地方在哪，怎么吸引过滤

成交：怎么吸引他成为你的客户，在你这购买第一单

鱼塘：你不断贡献价值，他不断重新消费的过程

我们现在学习重点不是产品经理，但大脑里面还是要有这个映像

再做任何项目之前，这个流程应该要打通：

现在大脑里面可以有个很浅显的映像：

在电视或其他网站看到，电商网站有活动，这活动很便宜，实惠，你跑过来看看（抓潜，他已经吸引你的注意力）

你因为这活动，注册了账号（留下电话号码，qq什么的），第一次在这网站上购买商品（成交）

网站通过你留的电话号码，qq告诉你又来了活动（双11），你又会打开网站，成为他的赚钱机器（鱼塘）

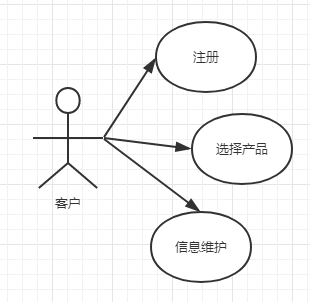
当然，真正产品经理要考虑的事情比这复杂得多，设计多个部门得协调合作，但目前我们如果想把上面这个所谓得模式落地，现在已经具备了一些东西

我们需要一个网站（现在仅仅是个构思），通过“活动”让客户注册，成交，反复的成交

这里的产出是“需求文档”

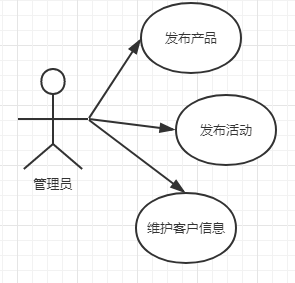
## 用“钱”的角度看设计

有了这个构思，接下来就是要把这项目设计出来，这一块也没程序员什么事，需要准备一堆的文档了，文档里面这个时候常见的就是用例图了



我们需要客户成交，只有客户成交，网站才能赚钱，那做的这网站“选择产品”，“下单”等核心功能是需要的，但这还不够，一些辅助的功能，比如“注册”，“信息维护这些又是必要的”

但客户为什么要来你这购买呢？需要活动，还有，这客户需要的产品又从哪来呢？



这一步设计可以看出项目不仅仅是给客户使用，还有部分需要给管理员使用的，总归现在项目可以分成两模块

1. 前端：给客户使用
2. 后端：给管理员使用

这里并不是想说怎么写设计文档，也没有一个具体答案，我相信很多同学甚至都写过很多设计文档了。

设计文档并不要设计多么花里胡哨，但一定要“完成商业模式的核心需求”

举一个具体例子：

前面说商业模式里有抓潜一步，具体是活动吸引，需要客户最轻松，最方便注册信息就好了

这些字都看得懂：某设计人A，设计个注册信息：

|  |  |
| --- | --- |
| 用户名 | 文本 |
| 密码 | 文本 |
| 确认密码 | 文本 |

这个很简单，客户也很容易注册，但这样真得好吗？为什么要方便注册？回到问题得根源，为什么需要他注册，注册后，作为网站得运营者，应该可以持续服务，给他做相关的推荐？

我没qq,没电话怎么做后续活动跟进？没性别，没年龄怎么做具体得推荐，所以说一个合格得产品设计师，一定不是那么简单，一定要站在全局得角度思考问题，知道怎么去拿捏简单与复杂的中间点

否则浪费几十上百万得推广费用，换来一堆这样得信息又有什么用呢，很多公司就喜欢这样，哪一堆这样的僵尸客户在自欺欺人

## 用“钱”的角度看架构

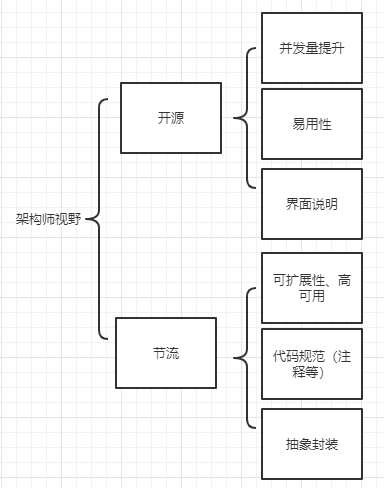
架构是项目的骨架，在真实的项目中，前期只要能完成客户的定义的需求，这也是最起码的要求，要客户通过我们搭建的项目完成赚钱的目的。

但在商业市场里，有一句话叫“开源节流”，开源就是前面说到的最根本地方，通过我们的项目架构，他能完成赚钱的业务需求，节流指的是，需要新功能的时候，服务器出现问题的时候，我们以最低的成本解决各种问题。

这样要求我们的系统好维护，足够健壮，扩展性好。

一个架构足够好的项目一定要屏蔽项目的复杂度，让初级，中级，高级程序员都能快速的进入工作，同时有新功能的开发也不会影响到以前的功能

这所做的一切都是在为老板节省开支，增加利润，为什么好的架构师值钱，如果你给老板节省了1000W，你要求分得200W你说过分吗？过分？我换老板，没有老板会这么傻的。



上面这些内容只是为扩充下同学们得视野，具体项目中的技术，架构后面再细说。

# 项目需求文档

## 概述

享学是具有独立法人资格的上市公司，是腾讯课堂下属的教育企业，有商业运营部门负责在商品的销售运营。

## 现行组织结构

享学信息技术有限公司分为：产品开发部、总经办、市场部、财务部、质量管理部、人事行政部。



商品运营部岗位设置

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 岗位 | 电话、联系方式 | 描述 |
| mark | 采购总管 | 138\*\*\*\*\*\*\*\* | 负责商品采购，维护产品分类 |
| king | 客户总管 | 138\*\*\*\*\*\*\*\* | 。 |
| peter | 售后总管 | 138\*\*\*\*\*\*\*\* |  |
| deer | 仓库总管 | 138\*\*\*\*\*\*\*\* |  |
| Jack | IT部门 | 138\*\*\*\*\*\*\*\* |  |
|  |  |  |  |

## 后端需求设计

后端需求详细设计旨在根据各模块的具体需求快速上手开发，通过需求达到跟程序员对接的目的。后端主要包含几大模块：商品模块、订单模块、统计模块、系统管理模块、个人中心模块。

### 商品模块

商品模块是后端商家统一来管理商品的上架、下架、秒杀等功能的模块，该模块包括：商品列表、商品分类、商品模型、商品规格、商品属性、秒杀商品等功能。

#### 商品列表

##### 主要功能

1. 展示该商户下的所有商品信息
2. 查询具体的商品信息
3. 发布商品信息到mongodb缓存
4. 编辑商品信息
5. 删除商品信息
6. 新增秒杀商品

##### 原型界面

商品列表原型界面

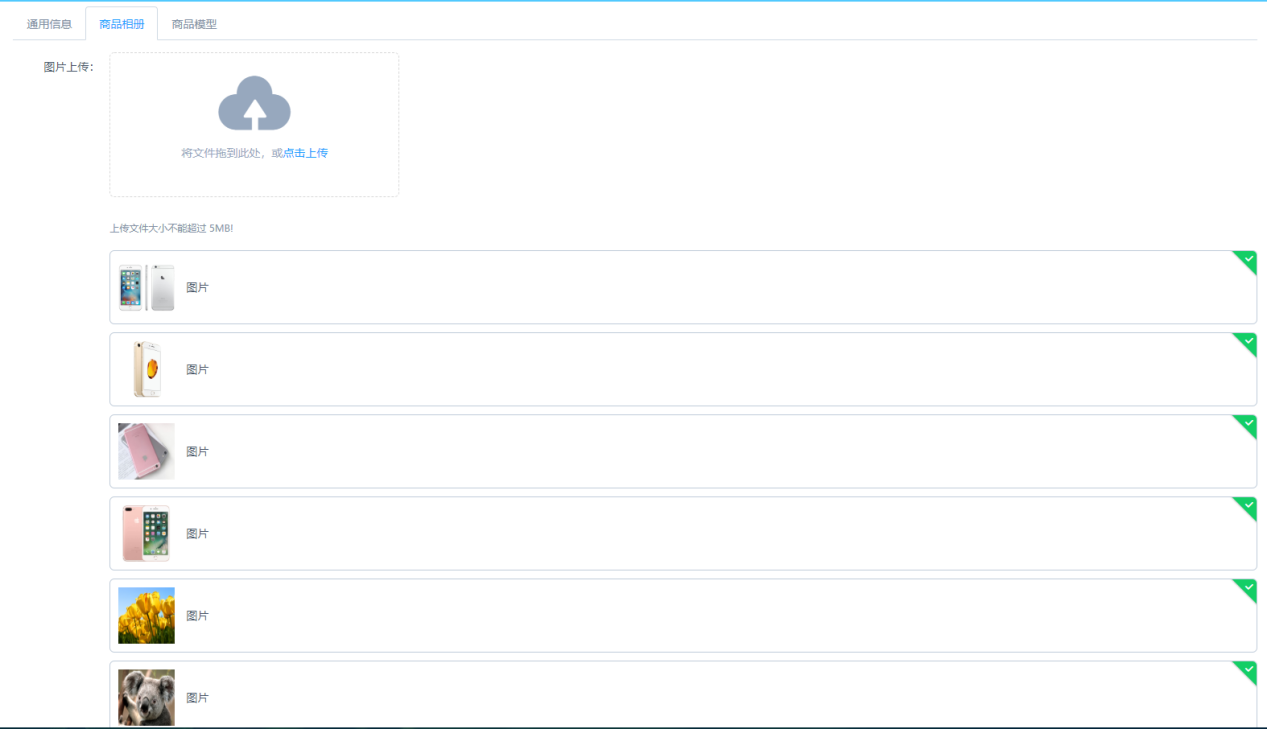


商品编辑原型界面

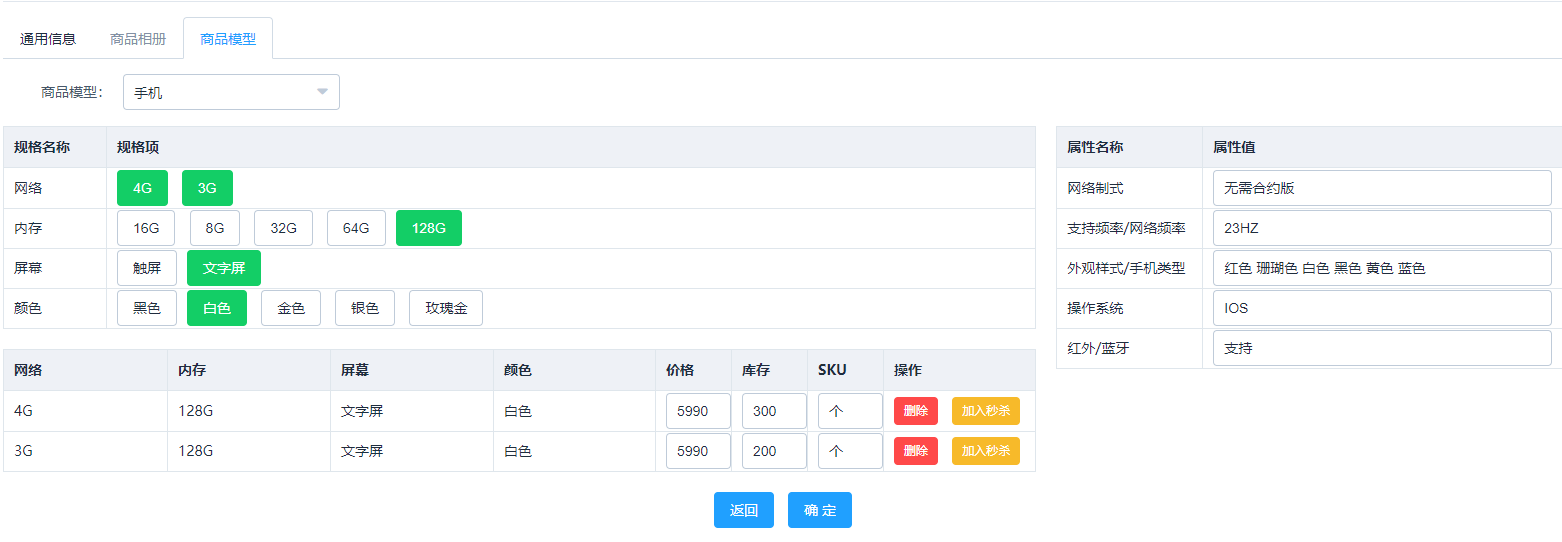




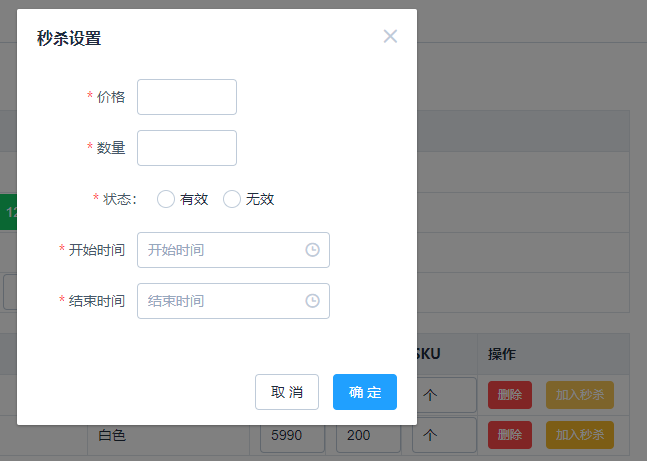
商品图片管理



商品规格和属性管理



新增秒杀商品



#### 商品分类

##### 主要功能

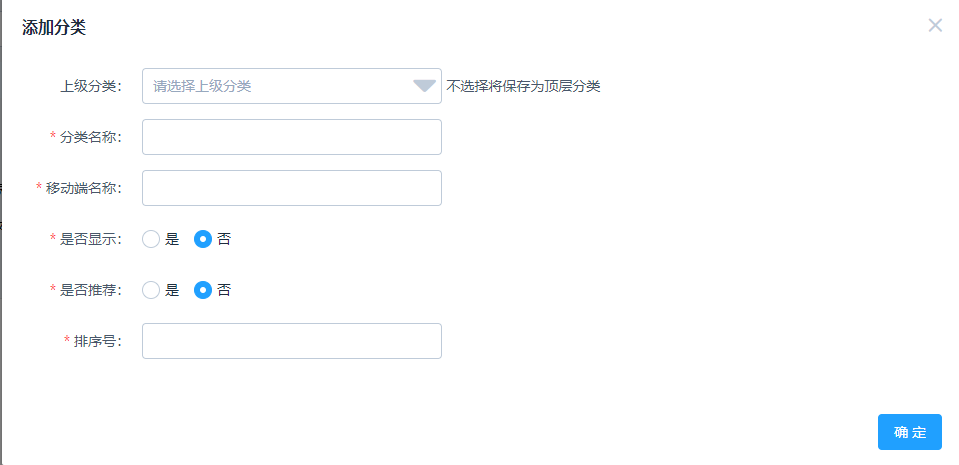
1. 新增商品类别目录
2. 删除商品类别目录
3. 查询商品类别目录
4. 把商品类别信息刷新到缓存

##### 原型界面

商品类别总览



新增商品类别



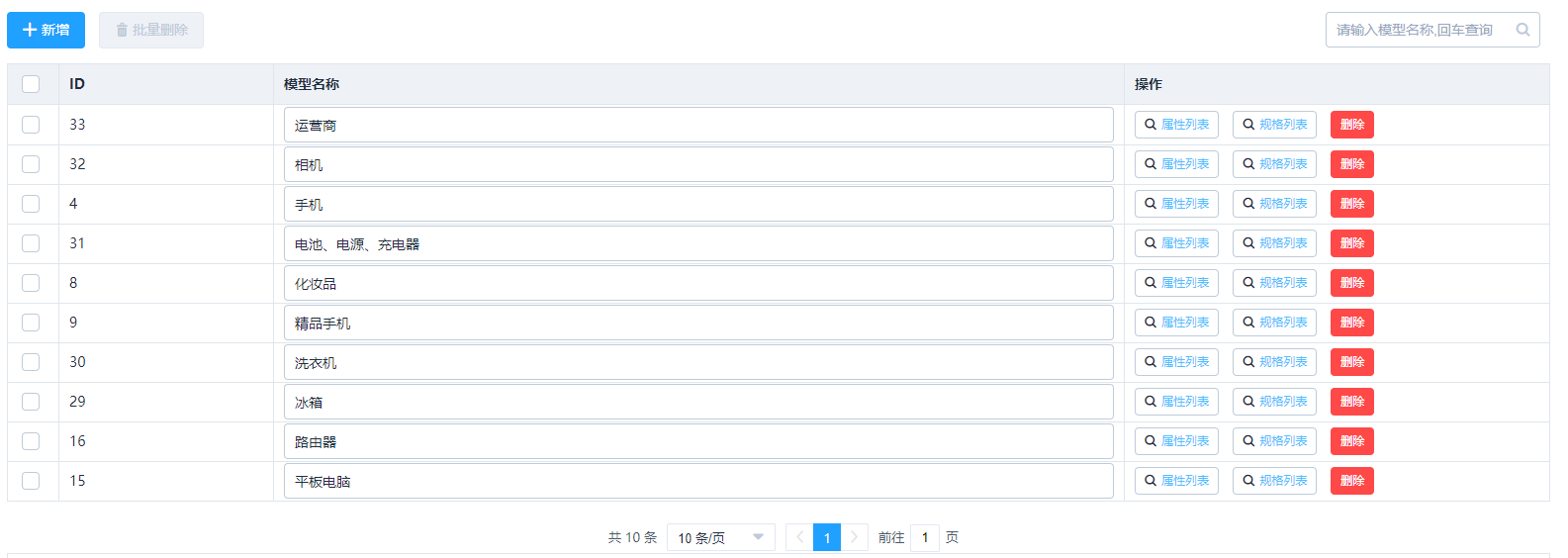
#### 商品模型

##### 主要功能

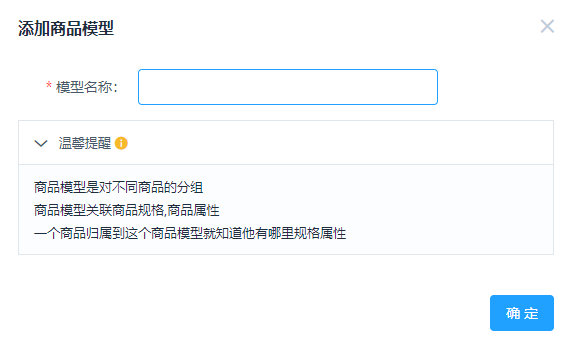
1. 维护商品的规格信息
2. 维护商品的属性信息
3. 查询商品模型信息
4. 删除商品模型信息

##### 原型界面

商品模型总览



新增商品模型



新增商品属性



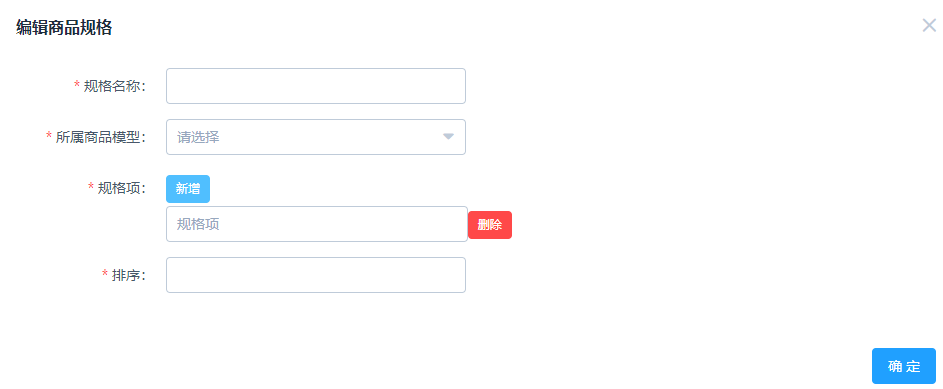
属性列表



商品规格列表



新增商品规格



#### 秒杀商品

##### 主要功能

1. 编辑秒杀商品
2. 删除秒杀商品
3. 查询秒杀商品

##### 原型界面

界面总览



编辑



### 订单模块

查询客户下的订单

#### 订单模块

##### 主要功能

主要是为了展示用户订单的详细信息

1. 按照支付状态查询
2. 按照发货状态查询
3. 按照订单状态查询
4. 按照订单号查询
5. 查询订单详细信息

##### 原型界面

订单界面总览



详细信息页



### 会员管理

小米商城会员的统一管理模块，通过该模块我们可以编辑会员信息、查看详情、重置密码、冻结用户、删除用户等功能

#### 会员管理

##### 主要功能

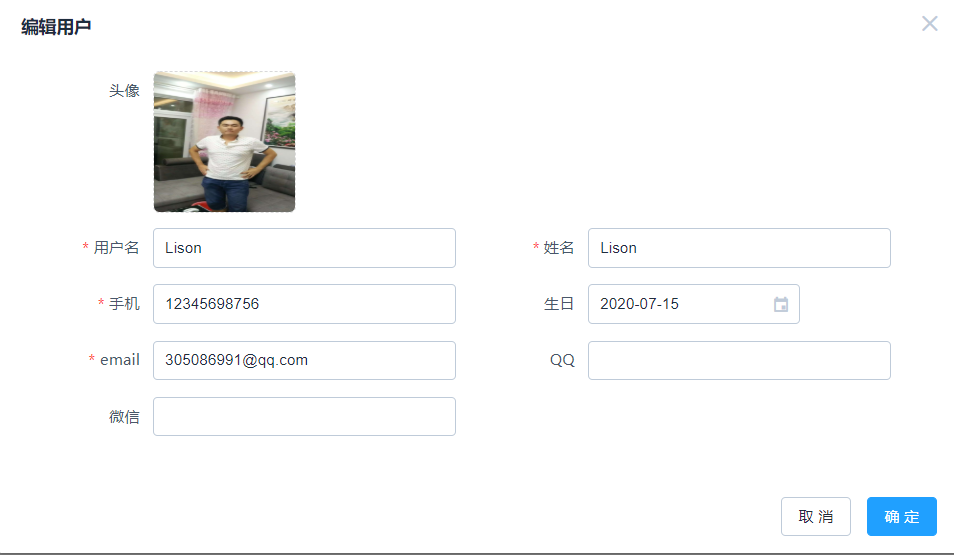
小米商城会员的统一管理模块，通过该模块我们可以编辑会员信息、查看详情、重置密码、冻结用户、删除用户等功能

##### 原型界面

会员管理总览



编辑用户



用户详情



### 系统管理

系统管理包括：用户管理、角色管理、权限管理、部门管理、字典管理等

#### 会员管理

同上

#### 角色管理

##### 主要功能

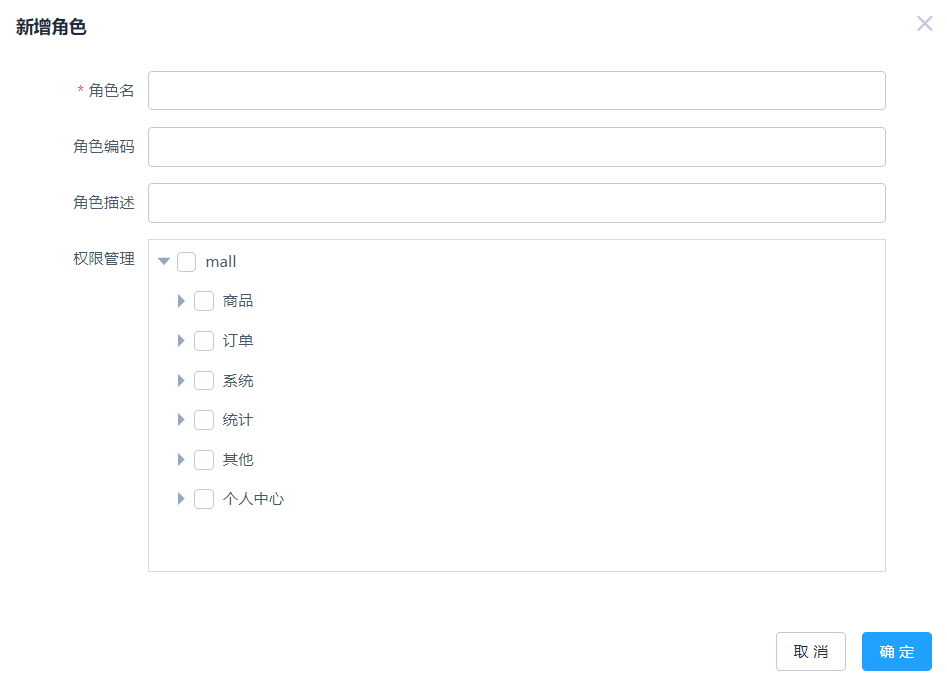
1. 新增用户角色
2. 查询用户角色
3. 删除用户角色
4. 查看用户详情

##### 原型界面

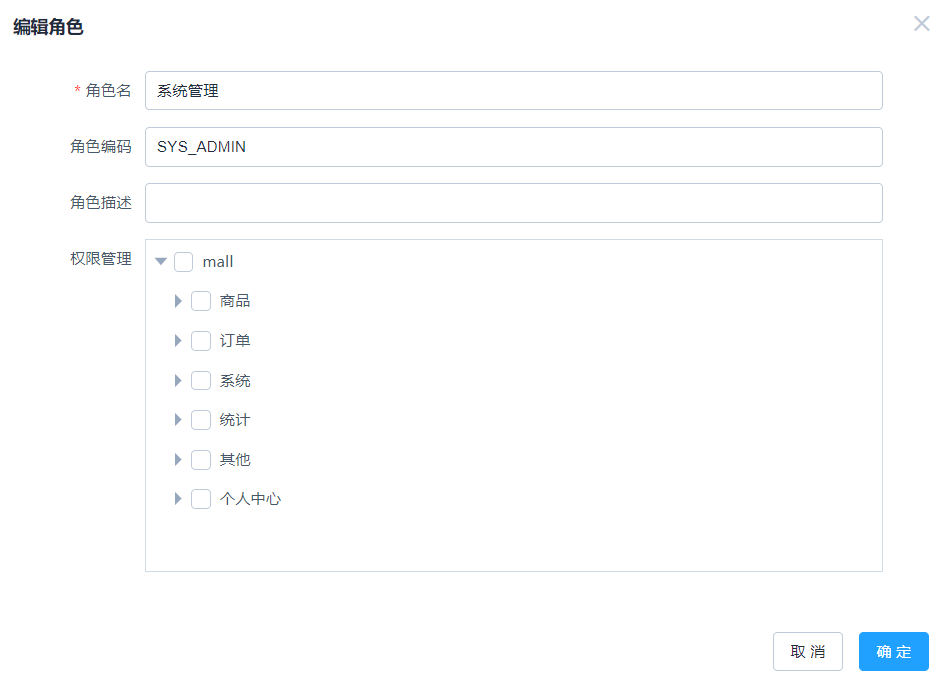
角色管理总览



新增角色



编辑角色



角色详情



#### 权限管理

##### 主要功能

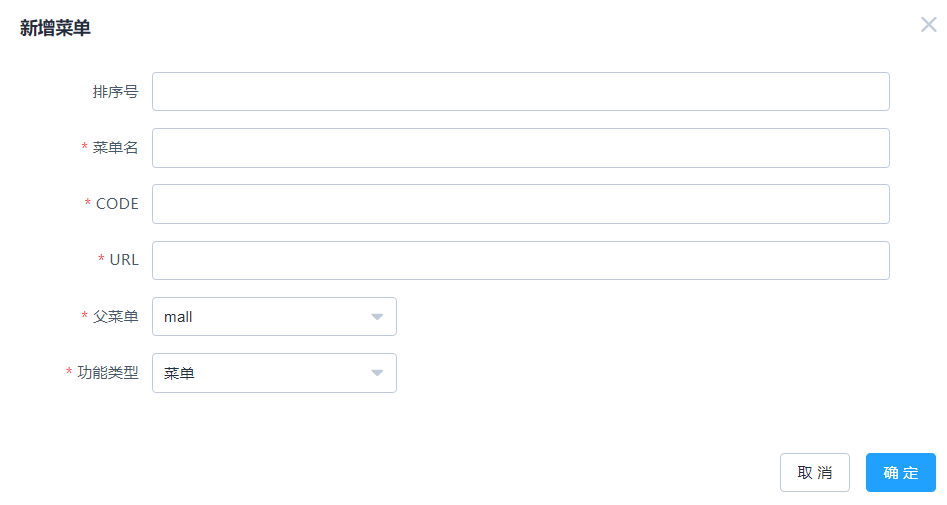
1. 新增权限
2. 编辑权限
3. 查询权限
4. 删除权限

##### 原型界面

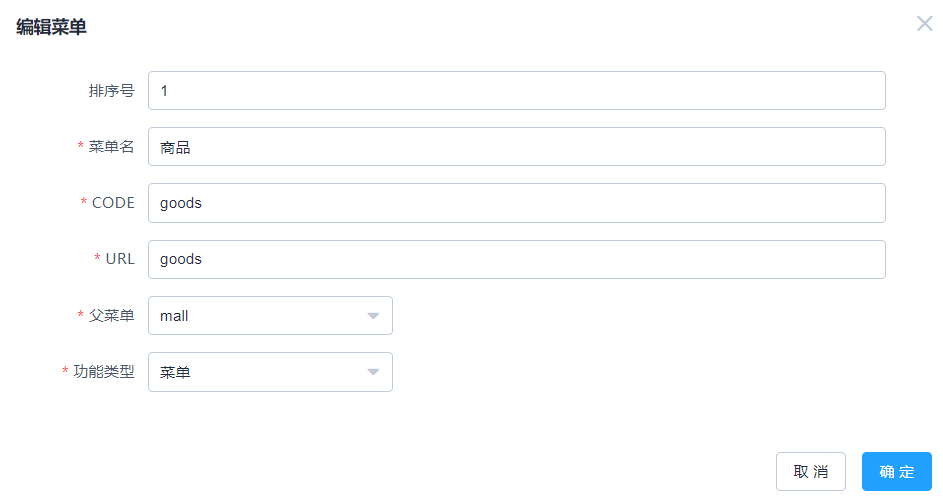
权限管理总览



新增权限



编辑权限



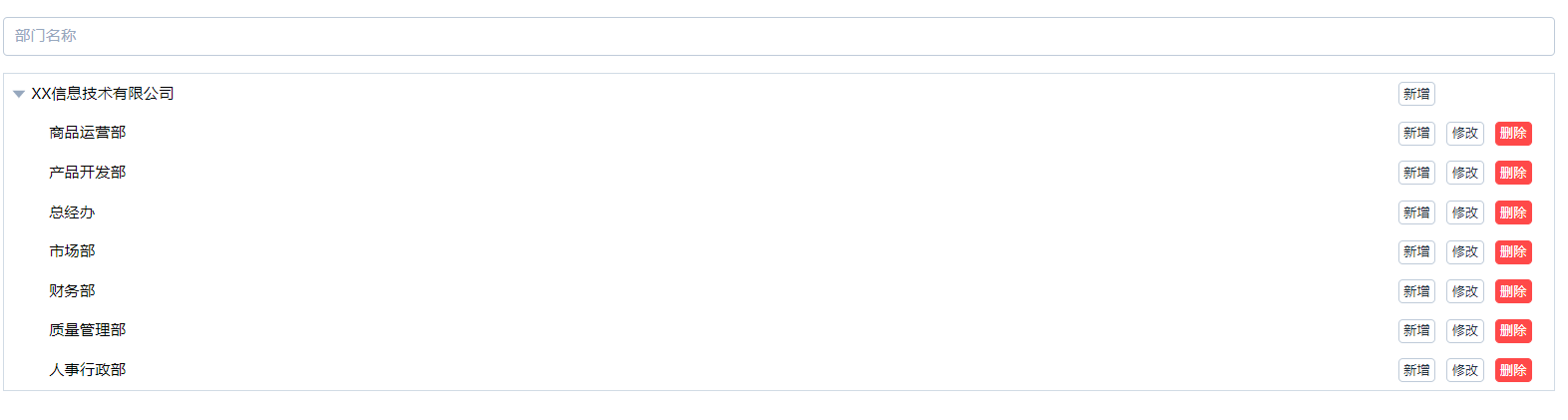
#### 部门管理

##### 主要功能

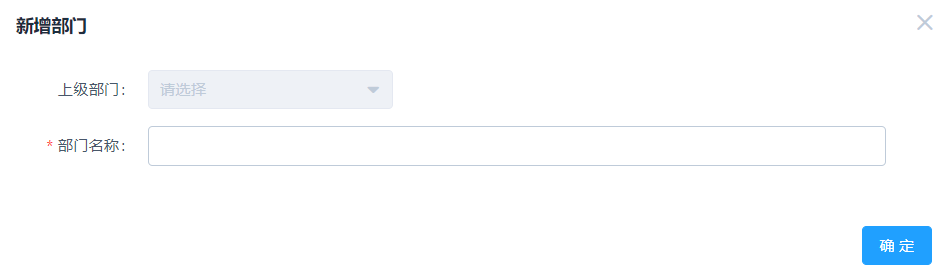
1. 新增部门
2. 修改部门
3. 查询部门
4. 删除部门

##### 原型界面

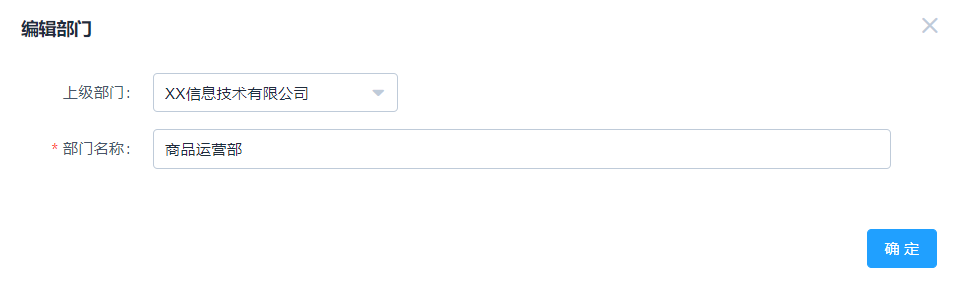
部门管理总览



新增部门



修改部门



#### 字典管理

##### 主要功能

1. 新增字典数据
2. 修改字典数据
3. 删除字典数据
4. 查询字典数据

##### 原型界面

界面总览



新增字典



修改字典



字典详情



### 个人中心

个人中心，包括个人资料和修改密码功能

#### 个人资料

##### 主要功能

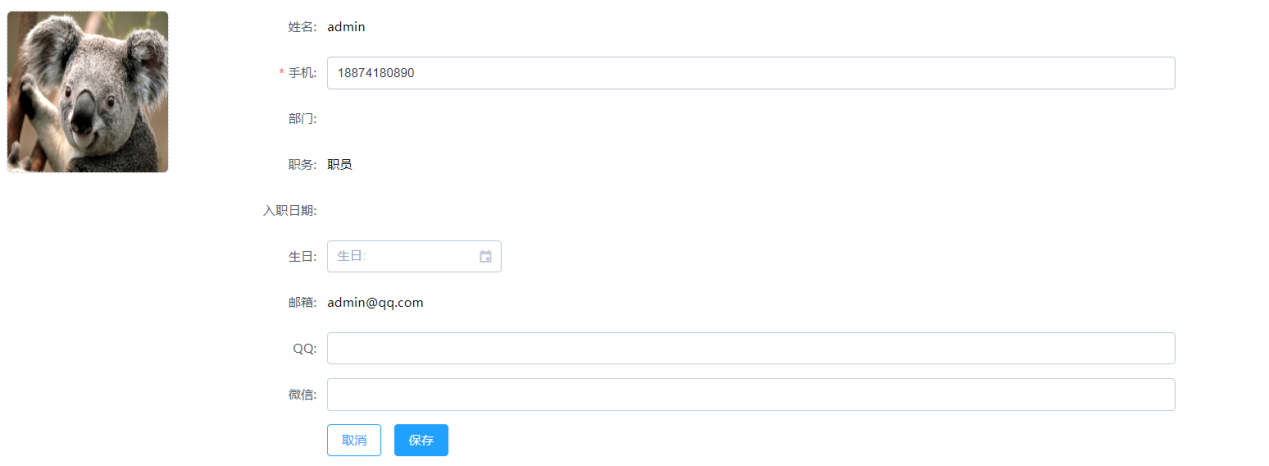
编辑和查看个人资料

##### 原型界面

查看个人资料



编辑个人资料



#### 修改密码

客服需要接听大量电话，并把产品信息录入某内网系统，操作麻烦

主要功能

修改密码

原型界面



## 前端需求设计

前端模块是小米商城最主要的模块，包括前台首页，首页包括：首页banner、秒杀商品、热门商品、各分类商品、首页商品目录，这些就是首页模块。

登录模块

商品详情

购物车模块

支付模块

订单详情模块

### 前端首页

#### 首页banner

##### 主要功能

Banner展示商城的最新动态

##### 原型界面



#### 首页商品目录

##### 主要功能

商品目录要能快速展示，展示最新的商品目录信息，且鼠标放到目录上能展示目录中的商品，点击商品时跳到商品详情页

##### 原型界面



#### 秒杀商品

##### 主要功能

实时的展示秒杀的商品，且秒杀时间过期时能实时的从界面中去除

##### 原型界面

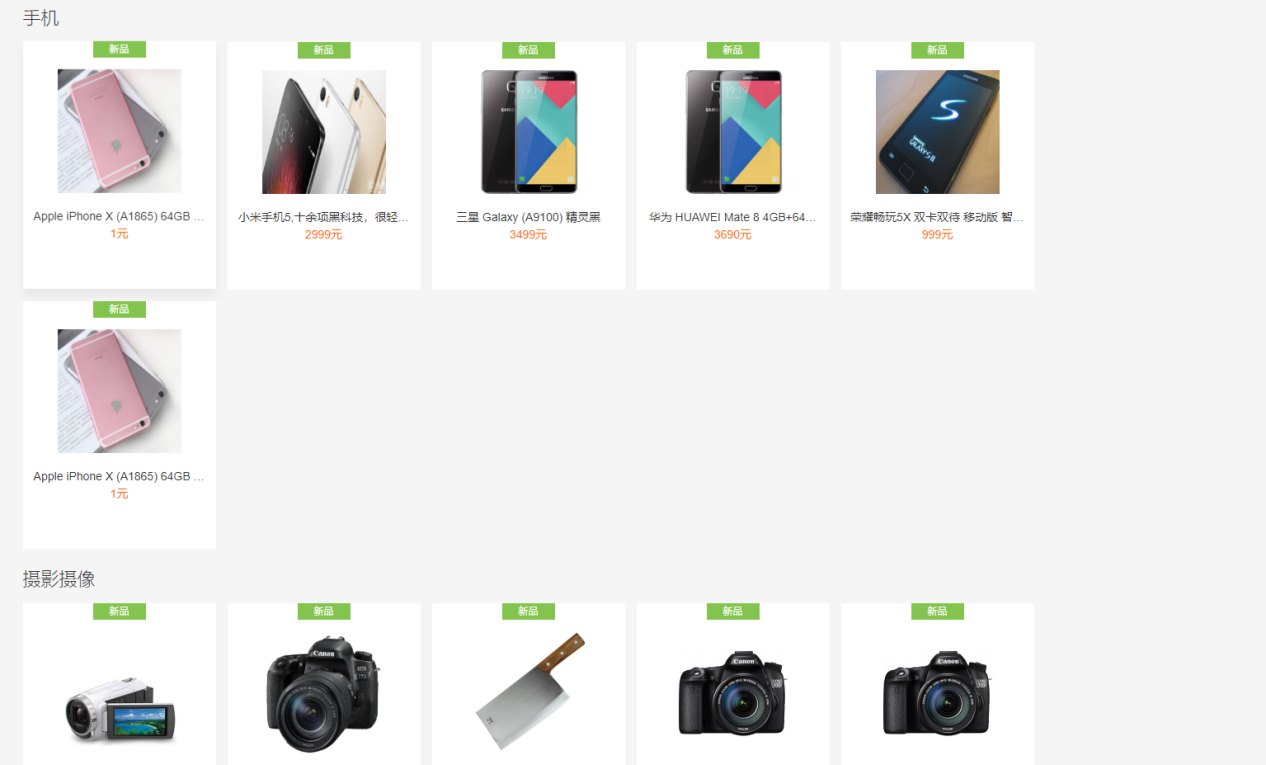


#### 分类商品

##### 主要功能

展示各分类类目的商品信息，且图片加载速度要快

##### 原型界面



### 商品详情

#### 商品详情

##### 主要功能

该页主要展示商品的详情信息，包括，商品详情大图、商品规格信息、商品属性信息、产品详情信息、参数信息等

##### 原型界面



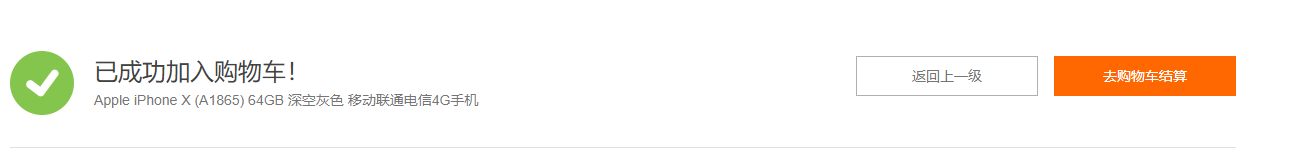


### 购物车

#### 购物车

##### 主要功能

在商品详情页点击加入购物车就进入到购物车操作界面，界面如图



再点击--去购物车结算就进入到了购物车界面。

购物车功能主要包括：选择要结算的商品、添加减少商品数量、查看购物车中物品等功能

##### 原型界面



### 地址管理

#### 地址管理

##### 主要功能

用户选择物品的收获地址，当收获地址不存在时用户可以新增收获地址，也可以对已有的地址进行修改。

##### 原型界面



### 支付管理

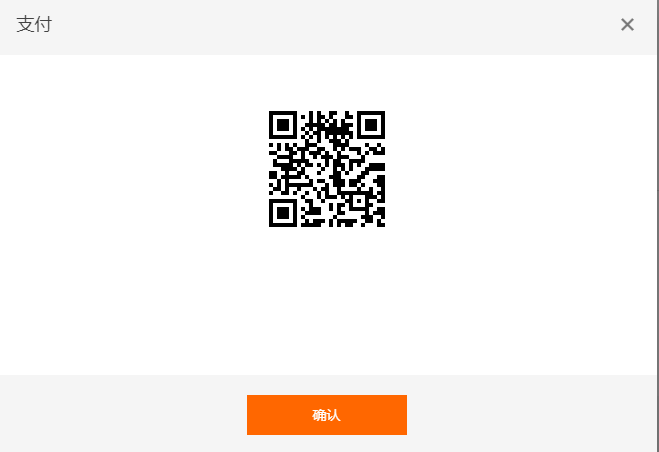
#### 支付管理

##### 主要功能

当在购物车界面点击去结算时，进入到支付订单界面，在该界面用户可以选择支付途径并完成支付动作

##### 原型界面





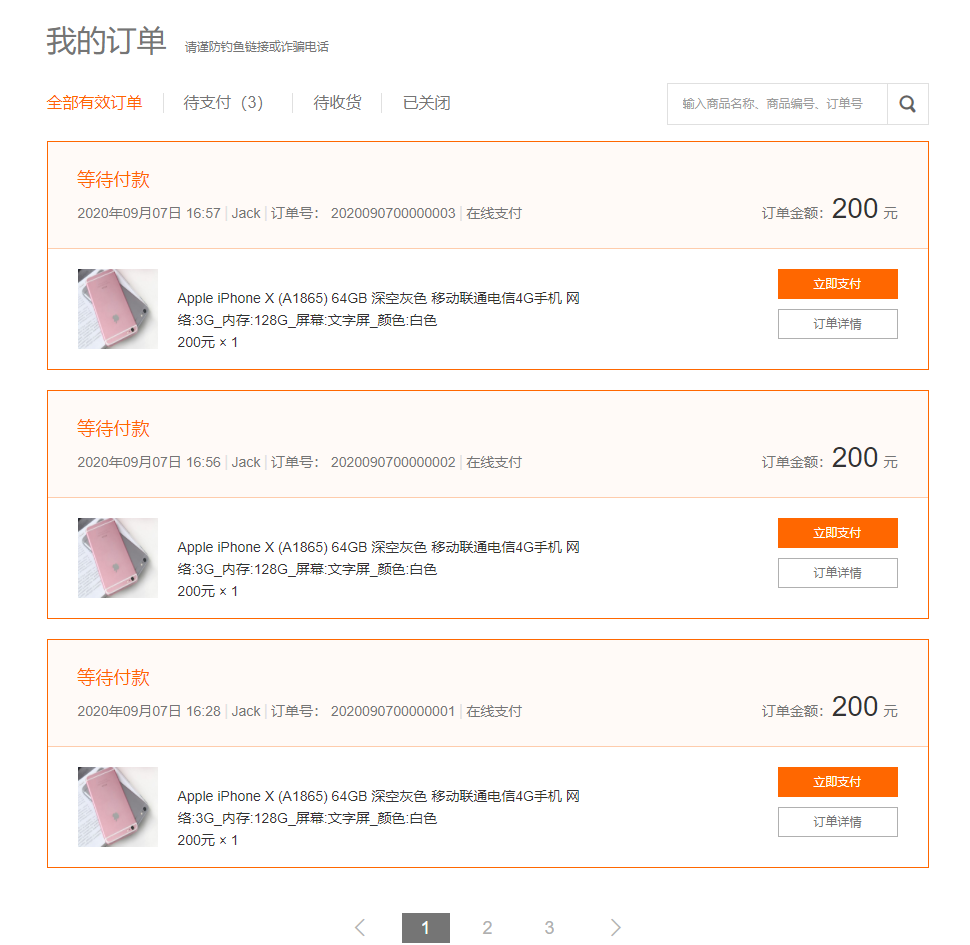
### 我的订单

#### 我的订单

##### 主要功能

查询我的订单信息，可以根据订单编码、商品编码查询我的订单信息

##### 原型界面



### 个人中心

#### 我的个人中心

##### 主要功能

展示个人信息，修改个人信息

##### 原型界面

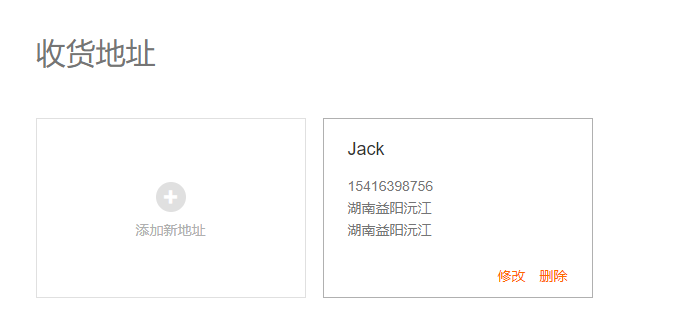


#### 收货地址

##### 主要功能

新增收获地址，修改收获地址

##### 原型界面

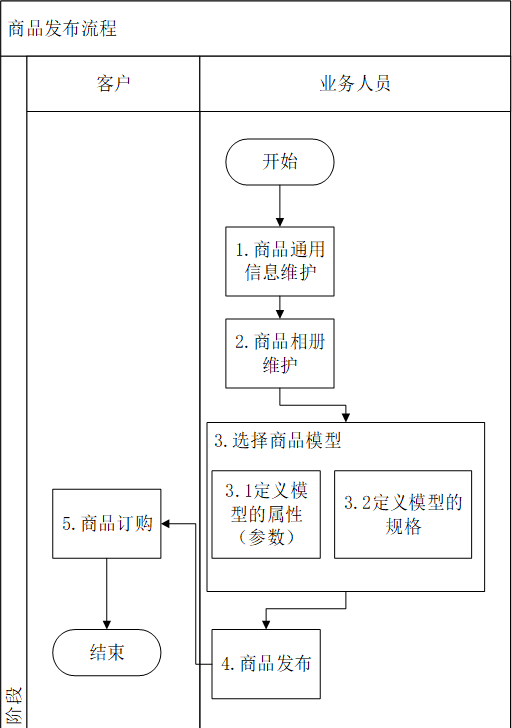


# 功能设计文档

产出（pdm数据库模型、接口文档）

## 具体流程设计

### 商品发布流程



|  |  |  |
| --- | --- | --- |
| 步骤 | 说明 | 角色 |
|  | 开始 |  |
| 1 | 新增商品，把通用信息进行维护（品牌，供应商） | 业务人员 |
| 2 | 商品相册维护 | 业务人员 |
| 3 | 选择模型（定义模型属性，定义规格） | 业务人员 |
| 4 | 商品发布 | 业务人员 |
| 5 | 商品订购 | 客户 |
|  | 结束 |  |

### 订单流程



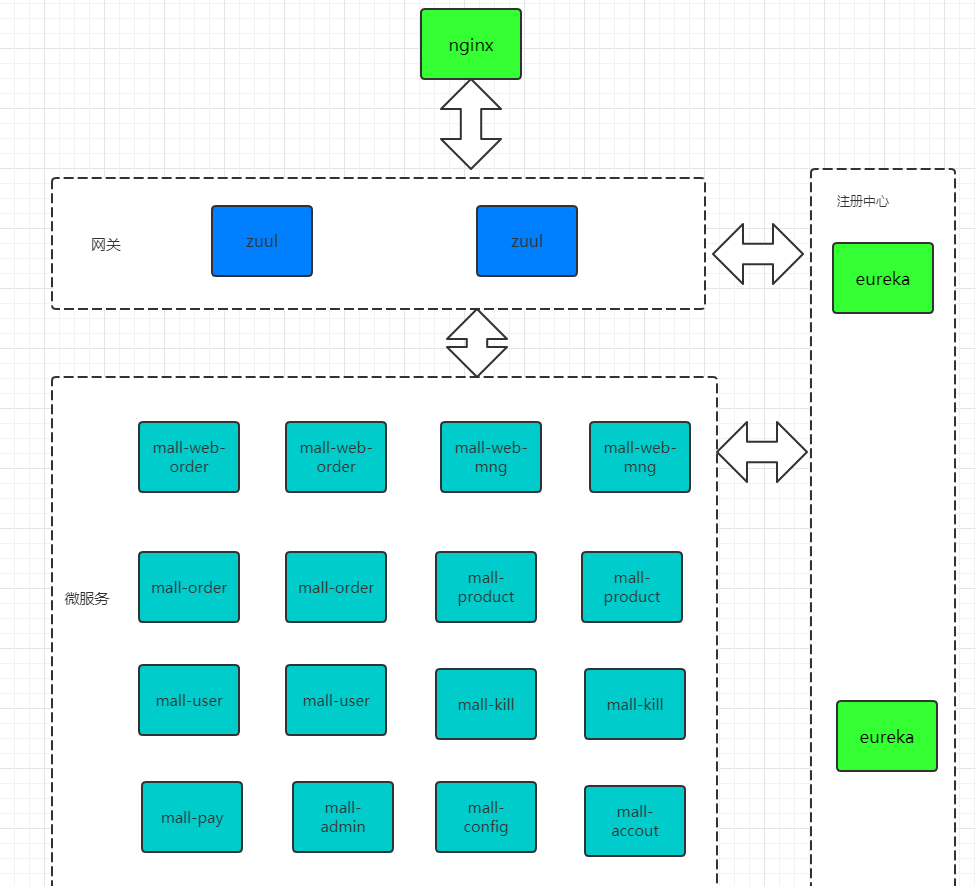
|  |  |  |
| --- | --- | --- |
| 步骤 | 说明 | 角色 |
|  | 开始 |  |
| 1 | 客户在网站上选择产品 | 客户 |
| 2 | 选择产品放入购物车 | 客户 |
| 3 | 结算 | 客户 |
| 4 | 选择收货地址 | 客户 |
| 5 | 支付 | 客户 |
| 6 | 确认订单 | 业务人员 |
| 7 | 产品发货 | 业务人员 |
| 8 | 确认收货 | 业务人员 |
|  | 结束 |  |

其他流程《略》

## 数据库设计

见pdm文档

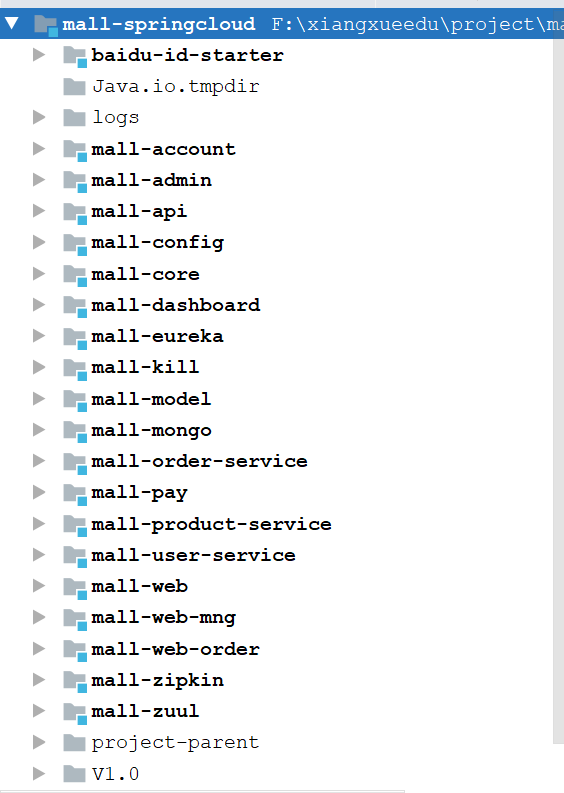
## 项目架构图



# 用到的技术

Springboot，springcloud，Zookeeper，redis，mongdb，fastdfs，nginx，mysql（读写分离），rabbitmq，kafka，seata，docker，Jenkins，mybatis，mycat，elk

# 项目架构模块



上图是整个系统的架构模块

**project-parent:**

父模块，定义各个模块用到组件的版本等信息

**mall-core:**

系统常量，业务异常，工具类等（比如DateUtil,JsonUtil），各种枚举等

**mall-mongo:**

服务接口需要查询mongodb的方法全部放到这里面

**mall-api:**

定义所有的业务接口，只是定义，并没有具体的实现

**mall-order-service:**

订单服务

**mall-user-service：**

用户，权限服务

**mall-product-service：**

产品服务

**mall-account:**

认证中心模块，用户的登陆信息，是单点登陆重要的模块

**mall-web-order:**

商城前台页面，是客户用到的系统

**mall-web-mng:**

后端管理模块，系统管理员使用，用于维护商品，分类等信息

## 启动流程：

1. 先启动eureka工程
2. 启动config工程
3. 接下来启动其他的工程即可，其他工程没有先后顺序

# 模块介绍

后端管理模块，这模块是系统管理员，要在前端能有产品显示，这里先得维护产品，分类信息等

http://mi.xiangxueketang.cn/mall-mng/#/login

前端模块，用户使用这模块下单

http://mi.xiangxueketang.cn/mall/#/home

## 后端管理模块

[http://www.mall.com/mall-mng/#/login](http://www.mall.com/mall-mng/" \l "/login)

admin/123456

使用管理员登陆

登陆之后有一系列的模块

### 业务介绍

整个商城的业务后台比较简单，主要维护商品信息，负责客户的订单，另外还有就是一些辅助的功能

#### 商品:

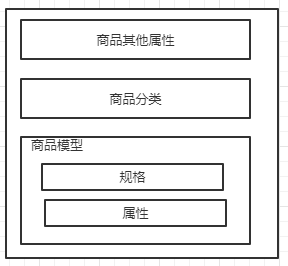
定义一个商品的详细描述，客户能购买的所有产品都在这定义

新增一个产品，会有很多信息需要填写，非常重要的有以下几类

“商品分类”

“商品模型”，商品模型里面又包含各种商品的规格，属性等信息

可以得到一个这样的图



#### 订单:

查看客户的订单，现在确认发货等

#### 统计:

会员管理，只是查询客户用户，不做具体说明

#### 系统管理:

用户管理（后台管理人员），角色管理，权限管理，部门管理等信息，可以分配具体的权限信息

#### 个人中心:

修改密码，修改用户个人资料

## 前端商城模块

前端商城模块主要是客户使用，客户需要在商城里面选择商品，下单，编辑送货信息，最后付款

http://mi.xiangxueketang.cn/mall/#/home

1. 注册账户
2. 登陆
3. 选择一个产品

[http://mi.xiangxueketang.cn/mall/#/product/413](http://self.sso.com/mall/" \l "/product/413)

1. 加入购物车
2. 新增收货地址
3. 支付
4. 我的订单

## 核心模块实现

### 新增商品

[http://mi.xiangxueketang.cn/mall-mng/#/goods/edit](http://self.sso.com/mall-mng/" \l "/goods/edit)

这个链接是用于打开商品列表

打开页面后会加载相关的商品品牌，供应商，分类信息

#### 供应商信息加载

前端目前写死的

#### 商品品牌信息加载

请求接口：

/mall-mng/api/brand/getAll

对应具体的方法：

cn.enjoy.mall.web.controller.quartz.BrandAction#getAll

这个很简单就是查询下tp\_brand表，把信息捞取出来而已

#### 分类信息加载

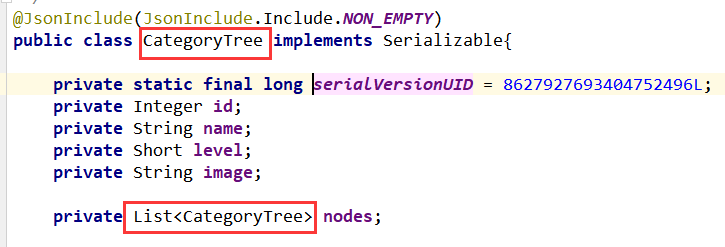
请求接口：

/mall-mng/api/goodscategory/getCategoryTree3?parentId=0

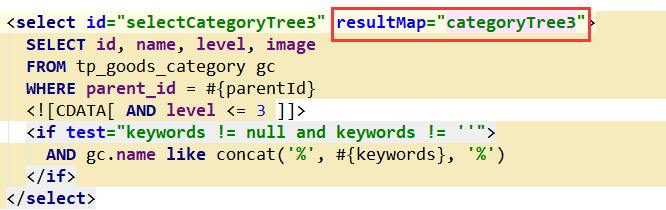
对应具体的方法：

cn.enjoy.mall.web.controller.GoodsCategoryController#getCategoryTree3

这个传入0，代表所有的分类信息都要捞取出来，返回的是个CategoryTree



这里面是个树的数据结构，节点里面包含节点，然后再去查询tp\_goods\_category表





这是使用mybatis的一种递归查询的特性

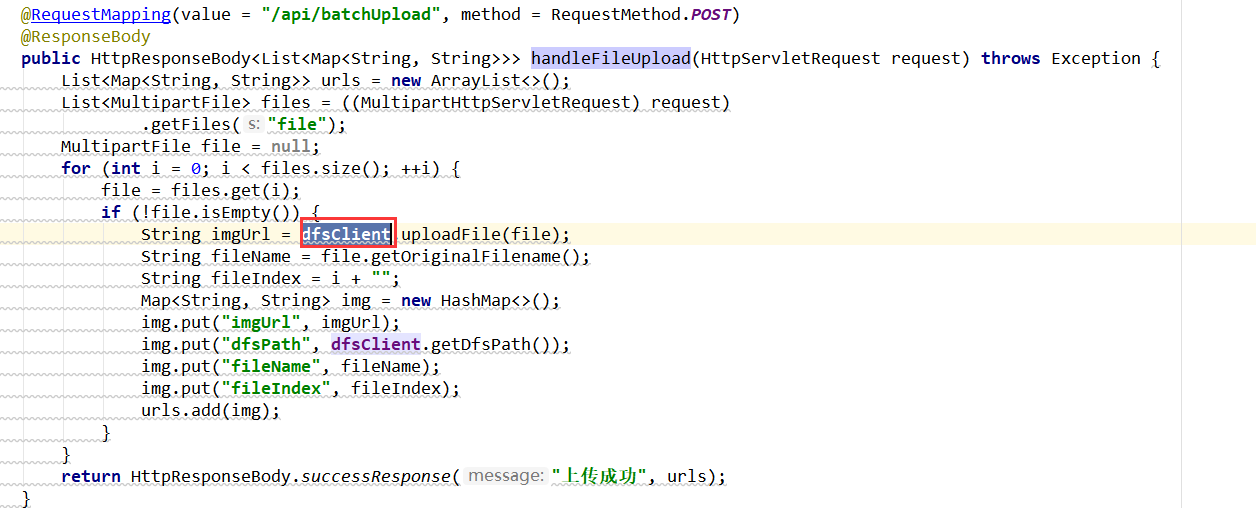
#### 图片上传

请求接口：

/mall-mng/api/batchUpload

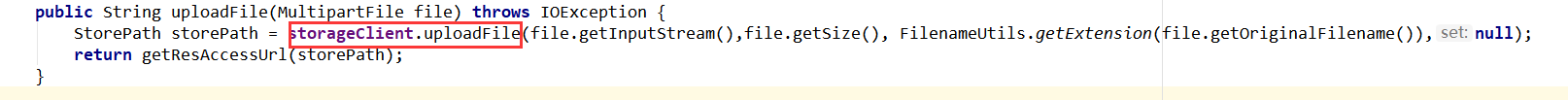
你会发现在mall-mng模块根本就没有这个batchUpload方法，这是因为他上层还有一个模块，为mall-web，这些图片上层这些通用的模块全部定义到了这里

对应具体的方法：

cn.enjoy.sys.controller.CommonController#handleFileUpload

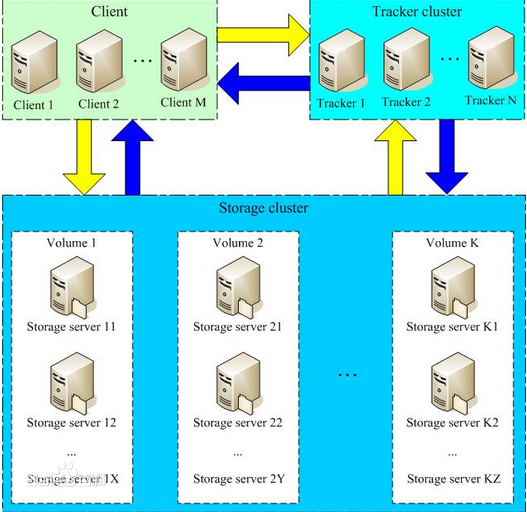
其中有个非常有意思的对象dfsClient，这个是FastDFSClientService

cn.enjoy.sys.fs.FastDFSClientService，对于fastdfs后面peter老师会重点的说，但也不影响我们先对fastdfs有一定的了解，可以看到，对于文件上传调用的方法实际是

cn.enjoy.sys.fs.FastDFSClientService#uploadFile(org.springframework.web.multipart.MultipartFile)

StorePath uploadFile(InputStream inputStream, long fileSize, String fileExtName, Set<MateData> metaDataSet);

传入了文件的输入流，文件的搭建，文件的扩展名，相关的元数据信息直接传入null



上面这个图是fastdfs的流程图。

1. client询问tracker上传到的storage，不需要附加参数；

2. tracker返回一台可用的storage；

3. client直接和storage通讯完成文件上传。

要使用fastdfs进行上传，肯定得进行相关得配置

<dependency>

<groupId>com.github.tobato</groupId>

<artifactId>fastdfs-client</artifactId>

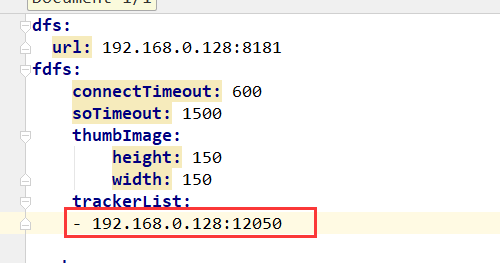
<version>1.25.3-RELEASE</version>

</dependency>

在application.yml里面还必须进行相关得配置

这里只配置了一个tracker

上面还有一个dfs.url，这个信息是用于展示图片信息的，可以看到他配置了个80端口，他实际也是通过nginx的一个扩展插件从nginx服务器上找具体的图片信息，这个nginx的查询信息请在讲fastdfs的时候重点关注以下



#### 保存商品

做好了一些相关维护后就可以保存商品了

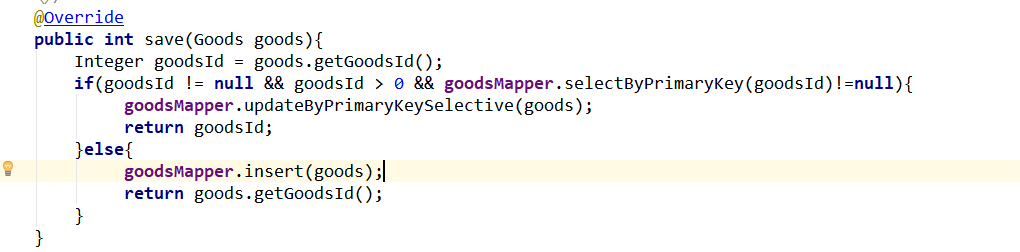
请求接口：

mall-mng/api/goods/save

对应具体的方法：

cn.enjoy.mall.web.controller.GoodsContoller#save

最后会调用

cn.enjoy.mall.service.impl.GoodsServiceImpl#save

这方法也不能首先判断商品是否存在，存在做修改操作，不存在就新增一个产品，最后数据保存到tp\_goods的表里面

#### 商品相册

保存商品后，就可以对商品的相册进行维护，用于在前端显示。

调用接口

/mall-mng/api/batchUpload

这个前面以及介绍过了，无非就是使用fastdfs把图片保存到文件服务器而已。

#### 商品模型

商品模型加载

首先要加载商品的模型，这里的模型前面以及说过

请求接口：

/mall-mng/api/goodsType/getAll

对应具体的方法：

cn.enjoy.mall.web.controller.GoodsTypeController#getAll

这个也很简单就是查询tp\_goods\_type，把信息捞取出来

选择商品模型

选择商品模型后，会自动带出商品模型，商品模型会带上规格与相关得属性，填写后，设置好库存信息

请求接口：

/mall-mng/api/goods/querySpecAndAttrByType

对应的具体方法：

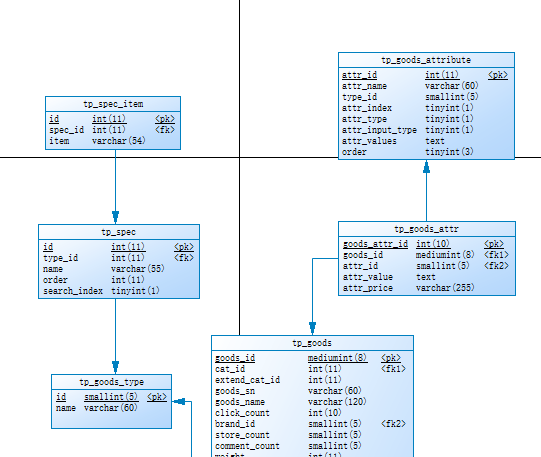
cn.enjoy.mall.web.controller.GoodsContoller#querySpecAndAttrByType

会从tp\_spec（商品规格表）、tp\_spec\_item（商品规格项）、tp\_goods\_attr（商品的属性）、tp\_goods\_attribute（商品的属性项），等表获取数据



FutureTask的使用

这里有个小的知识点，在获得最终返返回结果的时候，规格和属性对应的是两个不同的方法，在这使用了futuretask并发查询，最后再调用get方法把结果捞取到vo里面



商品模型保存



把该有的数据填写后，点击保存

请求接口：

mall-mng/api/goods/saveGoodsSpecAndAttr

对应具体的方法

cn.enjoy.mall.web.controller.GoodsContoller#saveGoodsSpecAndAttr

保存tp\_goods、

保存tp\_spec\_goods\_price（商品规则价格）

保存tp\_goods\_attr（商品属性）

#### 推荐，新品，热卖，上下架

这些操作其实都是修改商品的状态

请求接口

/mall-mng/api/goods/save

对应方法

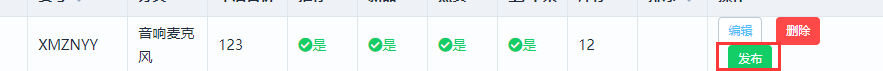
cn.enjoy.mall.web.controller.GoodsContoller#save

最后修改的是tp\_goods表

推荐（is\_recommend）新品（is\_new），热卖（is\_hot），上下架（is\_on\_sale）

#### 商品发布

编辑好商品后，还需要发布商品，没有发布的商品在前端是查找不到的。



请求接口：

/mall-mng/api/goods/publish

对应的具体方法：

cn.enjoy.mall.web.controller.GoodsContoller#publish

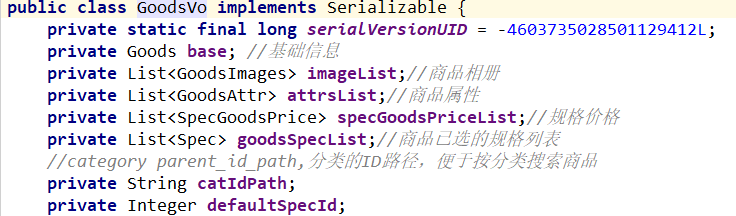
接口

cn.enjoy.mall.service.impl.GoodsServiceImpl#publishGoods2MongoDB(java.lang.Integer)

根据方法名应该知道，发布其实做的就是把当前的数据发布到mongodb

1. 首先从数据库中获取商品数据
2. 把数据库的数据发布到mongdb

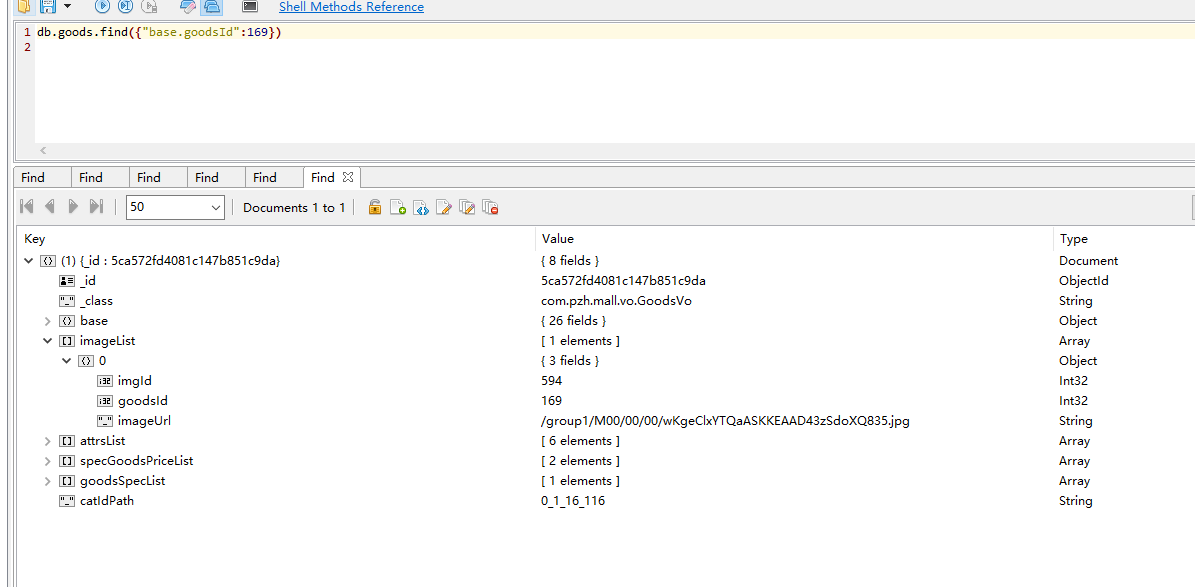
重点组织一个GoodsVo对象



把相应数据组织好，调用goodsDao.insert(voList);插入mongdb

最后使用的是mongoTemplate，插入mongodb

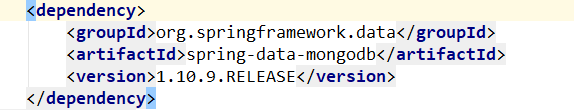
db.goods.find**({**"base.goodsId"**:**169**})**

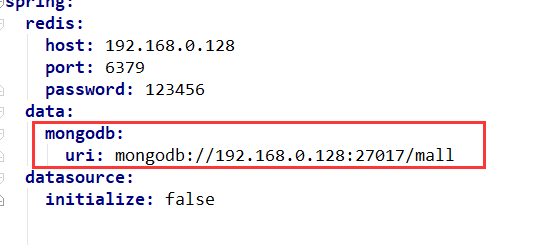


其中mongoTemplate是

org.springframework.data.mongodb.core.MongoTemplate 对象

要使用mongoTemplate必须先确认mongodb是开启的，并且有相关的依赖，配置





### 客户下单

商品发布后，客户就可以在商城页面看到这个商品，可以选择具体的商品下单

#### 商品查询

请求接口：

/mall/api/goods/searchList

对应具体方法：

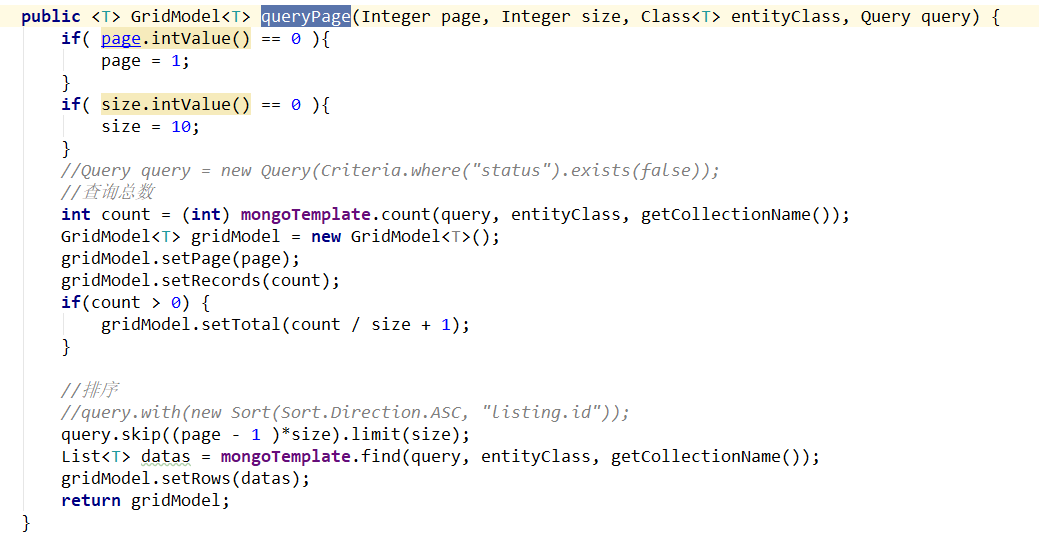
cn.enjoy.mall.web.controller.GoodsContoller#searchList

最后调用

cn.enjoy.mall.service.impl.GoodsServiceImpl#searchList

在这个方法里面使用Criteria，拼装查询条件，最后使用goodsDao.queryPage去查询

cn.enjoy.mall.mongo.BaseMgDao#queryPage 这方法是真正用于分页查询的方法



首先查询总数

使用Query对象的skip跳到需要取得的行数

使用limit取出size条数据。

最后返回数据，这个函数总体比较简单。

#### 选择产品

[http://mi.xiangxueketang.cn/mall/#/product/413](http://self.sso.com/mall/" \l "/product/413)

可以在见面里面选择一个具体的产品下单

调用的接口

/mall/api/goods/detailBySpecGoodsId/413

对应的具体方法：

cn.enjoy.mall.web.controller.GoodsContoller#detailBySpecGoodsId

传入产品的id

最后

cn.enjoy.mall.service.impl.GoodsServiceImpl#findOneBySpecGoodsId

public GoodsVo findOneBySpecGoodsId(Integer specGoodsId) {

return mongoTemplate.findOne(new Query(Criteria.where("specGoodsPriceList.\_id").is(specGoodsId)), GoodsVo.class, getCollectionName());

}

#### 加入购物车

调用接口：

/mall/api/shoppingCart/add

对应的具体方法：

cn.enjoy.mall.web.controller.ShoppingCartContoller#add

最后调用

cn.enjoy.mall.service.impl.ShoppingCartServiceImpl#save

public void save(Integer specGoodsId,Integer num,String userId,Integer saveMode) {

num = num == null ? 1 : num;

//1查询mongdb当前用户是否有购物车信息

UserShoppingGoodsVo userShoppingGoodsVo = shoppingCartDao.findOne(userId);

if(userShoppingGoodsVo == null){

userShoppingGoodsVo = new UserShoppingGoodsVo();

}

userShoppingGoodsVo.setUserId(userId);

//2查询mongdb里面具体的商品信息

GoodsVo goodsVo = goodsDao.findOneBySpecGoodsId(specGoodsId);

if(goodsVo == null){

throw new BusinessException("没有找到对应的商品");

}

if(goodsVo.getBase().getIsOnSale() == false){

throw new BusinessException("对不起，该商品已下架");

}

List<SpecGoodsPrice> machedSpecGoodsPriceList = goodsVo.getSpecGoodsPriceList().stream().filter(specGoodsPrice -> specGoodsPrice.getId().intValue()==specGoodsId.intValue()).collect(Collectors.toList());

if(machedSpecGoodsPriceList==null || machedSpecGoodsPriceList.size()==0){

throw new BusinessException("没有找到对应的商品");

}

if(machedSpecGoodsPriceList.get(0).getStoreCount().intValue()<num.intValue()){

throw new BusinessException("对不起，该商品的库存不足");

}

BigDecimal price = machedSpecGoodsPriceList.get(0).getPrice();

String keyName = machedSpecGoodsPriceList.get(0).getKeyName();

//取出购物车的商品集合

List<ShoppingGoodsVo> shoppingGoodsList = userShoppingGoodsVo.getShoppingGoodsList();

if(shoppingGoodsList == null ){

shoppingGoodsList = new ArrayList<>();

}

//购物车中没有该类型商品，则直接添加

if(CollectionUtils.isEmpty(shoppingGoodsList)

|| shoppingGoodsList.stream().filter(shoppingGoodsVo -> shoppingGoodsVo.getSpecGoodsId().intValue()== specGoodsId.intValue()).count() == 0 ){

ShoppingGoodsVo shoppingGoodsVo = new ShoppingGoodsVo();

shoppingGoodsVo.setSpecGoodsId(specGoodsId);

shoppingGoodsVo.setGoodsId(goodsVo.getBase().getGoodsId());

shoppingGoodsVo.setGoodsName(goodsVo.getBase().getGoodsName());

shoppingGoodsVo.setOriginalImg(goodsVo.getBase().getOriginalImg());

shoppingGoodsVo.setPrice(price);

shoppingGoodsVo.setNum(num);

shoppingGoodsVo.setStatus(1);

shoppingGoodsVo.setKeyName(keyName);

shoppingGoodsList.add(shoppingGoodsVo);

userShoppingGoodsVo.setShoppingGoodsList(shoppingGoodsList);

}else { //购物车中有该类型商品，则根据保存模式决定累加还是替换

for (ShoppingGoodsVo vo : shoppingGoodsList ) {

if (vo.getSpecGoodsId().intValue() == specGoodsId.intValue()) {

vo.setGoodsId(goodsVo.getBase().getGoodsId());

vo.setGoodsName(goodsVo.getBase().getGoodsName());

vo.setOriginalImg(goodsVo.getBase().getOriginalImg());

vo.setPrice(price);

if(saveMode == SAVE\_MODE\_APPEND.intValue()){

vo.setNum(vo.getNum() + num);

vo.setStatus(1);

}else{

vo.setNum(num);

}

break;

}

}

userShoppingGoodsVo.setShoppingGoodsList(shoppingGoodsList);

}

shoppingCartDao.save(userShoppingGoodsVo);

}

#### 去结算，选择收获地址

请求地址：

/mall/api/userAddress/list

具体方法

cn.enjoy.mall.web.controller.UserAddressController#list

这个方法简单，直接查询tp\_user\_address表，根据用户的id查询地址信息，并返回

#### 去结算

请求地址：

/mall/api/order/save

具体方法：

cn.enjoy.mall.web.controller.OrderController#save

把地址信息传过去后，创建一个新的订单tp\_order，并把订单的产品信息保存到商品订单表tp\_order\_goods,还把当前的日志信息以日志的方式保存到tp\_order\_action中，其中的action为创建订单

public Integer createOrder(OrderCreateVo orderCreateVo, String userId) {

//创建一个订单

Order order = new Order();

//从SequenceGenerator中获取订单的变化

order.setOrderSn(sequenceGenerator.getOrderNo());

order.setAddTime(System.currentTimeMillis());

//设置订单的状态为未确定订单

order.setOrderStatus(OrderStatus.UNCONFIRMED.getCode());

//未支付

order.setPayStatus(PayStatus.UNPAID.getCode());

//未发货

order.setShippingStatus(ShippingStatus.UNSHIPPED.getCode());

//获取发货地址

UserAddress userAddress = userAddressMapper.selectByPrimaryKey(orderCreateVo.getAddressId());

BeanUtils.copyProperties(userAddress,order);

order.setUserId(userId);

//新增订单

orderMapper.insert(order);

int orderId = order.getOrderId();

BigDecimal totalAmount = new BigDecimal(0);

//从mongodb的购物车中获取所购物品

List<ShoppingGoodsVo> checkedGoodsList = shoppingCartService.findCheckedGoodsList(userId);

List<OrderGoods> orderGoodsList = new ArrayList<>();

for(ShoppingGoodsVo goodsAddVo : checkedGoodsList){

GoodsVo goodsVo = goodsDao.findOneBySpecGoodsId(goodsAddVo.getSpecGoodsId());

if(goodsVo == null ){

throw new BusinessException("没有找到对应的商品["+goodsAddVo.getGoodsName()+"],可能已下架");

}

if(goodsVo.getBase().getIsOnSale() == false){

throw new BusinessException("对不起，商品["+goodsAddVo.getGoodsName()+"]已下架");

}

List<SpecGoodsPrice> machedSpecGoodsPriceList = goodsVo.getSpecGoodsPriceList().stream()

.filter(specGoodsPrice -> specGoodsPrice.getId().intValue() == goodsAddVo.getSpecGoodsId().intValue()).collect(Collectors.toList());

if(CollectionUtils.isEmpty(machedSpecGoodsPriceList)){

throw new BusinessException("没有找到对应的商品["+goodsAddVo.getGoodsName()+"],可能已下架");

}

SpecGoodsPrice specGoodsPrice = machedSpecGoodsPriceList.get(0);

//更新商品库存

goodsVo.getSpecGoodsPriceList().remove(specGoodsPrice);

specGoodsPrice.setStoreCount(specGoodsPrice.getStoreCount() - goodsAddVo.getNum().shortValue());

goodsVo.getSpecGoodsPriceList().add(specGoodsPrice);

//把当前订单信息保存到mongodb

goodsDao.save(goodsVo);

//创建的订单商品

OrderGoods orderGoods = new OrderGoods();

orderGoods.setOrderId(orderId);

BeanUtils.copyProperties(goodsVo.getBase(),orderGoods);

orderGoods.setGoodsNum(goodsAddVo.getNum().shortValue());

orderGoods.setGoodsPrice(specGoodsPrice.getPrice());

orderGoods.setBarCode(specGoodsPrice.getBarCode());

orderGoods.setSpecKey(specGoodsPrice.getKey());

orderGoods.setSpecKeyName(specGoodsPrice.getKeyName());

orderGoods.setSpecGoodsId(specGoodsPrice.getId());

if(CollectionUtils.isEmpty(specGoodsPrice.getSpecGoodsImagesList())){

orderGoods.setOriginalImg(goodsVo.getBase().getOriginalImg());

}else{

orderGoods.setOriginalImg(specGoodsPrice.getSpecGoodsImagesList().get(0).getSrc());

}

orderGoodsList.add(orderGoods);

totalAmount = totalAmount.add(specGoodsPrice.getPrice().multiply(new BigDecimal(goodsAddVo.getNum())));

}

order.setGoodsPrice(totalAmount);

order.setShippingPrice(new BigDecimal(0));

order.setOrderAmount(totalAmount.add(order.getShippingPrice()));

order.setTotalAmount(totalAmount.add(order.getShippingPrice()));

//修改订单

orderMapper.updateByPrimaryKeySelective(order);

//保存订单产品信息

orderGoodsMapper.insertBatch(orderGoodsList);

//清除购物车中已下单的商品

shoppingCartService.removeCheckedGoodsList(userId);

//订单日志

orderActionService.save(order,"创建订单",userId);

return orderId;

}

#### 支付订单

目前这块还没对接真实的支付接口，只在后端进行相关的逻辑处理

调用接口：

/mall/api/pay/orderPay

orderId: 1836

payCode: weixin

payAmount: 40

具体业务代码：

cn.enjoy.mall.web.controller.PayController#orderPay

最后调用

cn.enjoy.mall.service.impl.PayServiceImpl#doPay

把订单的支付状态进行修改，修改成“已支付”

保存一条支持成功的日志

### 管理员订单维护

客户下订单后，管理员可以对订单进行维护

#### 订单查询

接口：

mall-mng/api/order/queryByPage

具体调用的方法：

cn.enjoy.mall.web.controller.OrderController#queryByPage

实际做的事情就事查询tp\_order表，并把相关的数据捞取出来。

#### 查看订单

在订单查询后，可以根据订单的编号查询订单

接口：

/mall-mng/api/order/queryGoodsByOrderId?orderId=orderId

具体调用的方法：

cn.enjoy.mall.web.controller.OrderController#queryGoodsByOrderId

实际做的事情就事查询tp\_order\_goods，把订单订购的商品信息捞取出来。

#### 订单确认

如果客户已经付款，管理员可以在后端对这个订单进行确认

/mall-mng/api/order/update

具体调用的方法：

cn.enjoy.mall.web.controller.OrderController#update

实际做的事情就事修改下tp\_order表

传入订单状态为orderStatus=1

#### 去发货：

填写相关发货信息，接下来就可以进行发货了。

点击确定。

调用接口：

/mall-mng/api/order/shipping

具体调用的方法:

cn.enjoy.mall.web.controller.OrderController#shipping

做的具体事情：

写入tp\_delivery\_doc（发货单）表，并修改tp\_order\_goods表，设置发货状态为1

### 客户端确认收货

调用接口：

/mall/api/order/confirmReceiveGoods

具体调用的方法:

cn.enjoy.mall.web.controller.OrderController#confirmReceiveGoods

做的具体的事情：

修改tp\_order状态，把相关信息保存到mongodb，方便后面查询

### 热门商品

#### 热门商品查询

调用接口：

/mall/api/goods/hotSelling

具体调用得方法：

cn.enjoy.mall.web.controller.GoodsContoller#queryHotSelling

而在服务层最后调用的

cn.enjoy.mall.service.impl.GoodsServiceImpl#queryHotSellingGoods(Integer start,Integer end)



可以看下，在这里是用了redis的zset有序集合，里面存放的是goodsid 和相对应的数量，取出GoodsId然后去mongodb数据库里面查询出具体的信息进行填充。

#### 热门商品统计

前台的查询已经搞明白了，但是redis数据从哪来呢？

定时任务：

cn.enjoy.mall.service.impl.ScheduledServiceImpl#scheduledHotProduct



定时从数据库查询统计，把查询的结果放到redis，另外还做了一个功能，把相应得数据统计到一张表里面，方便后台管理统计查询使用，这里就没做界面了，对于得界面也就是个查询得功能

#### 分布式锁

要注意一点，定时任务是在order服务里面完成得，order为了实现高可用可能重启多个，这就会导致定时任务里面的任务出现分布式的并发安全问题，如果控制不好，会导致数据重复插入的情况。

为了从解决这问题，这里使用了一个分布式锁，基于zk实现，在这里使用的是非可重入锁，每次执行统计任务的时候，需要先获得锁，如果没有获得锁返回false并不会继续执行



### 数据库读写分离

在项目里面是配置了读写分离的

#### 多数据源配置

cn.enjoy.config.DataSourceConfig 类里面配置了2个数据源，一个主用于对数据库的写操作，另外一个数据库是用于读



可见在这个配置类里面使用了MyRoutingDataSource，这是一个路由的数据源，他从ThreadLocal里面取得枚举的值



那这master和slave又是谁调用的呢？

#### AOP切换路由键

cn.enjoy.aop.DataSourceAop



这个也非常简单，会拦截service的请求，如果是查询走从库，如果是修改操作走主库

### 秒杀抢购

#### 客户秒杀

秒杀其实是就活动这块的内容，在整个项目运营周期中起了非常重要的作用，在前面说的商业模式中，吸引客户需要用到活动，成交流程需要活动，让客户重复购买也需要适时的推出活动

秒杀是按业务流程来说其实很简单，本质上来说就是新增一个订单，订单新增完成后付款完事。

如果并发量不大，他其实就是一个他和普通的购物，新增一个订单并不需要太大的差别，但秒杀活动由于属于一种营销活动，带有稀缺性和时间紧迫感，会在短时间带来大量的并发访问量。

这里的并发可以分为两部分

1. 并发客户会不断的查询数据库里的库存数量，导致查询压力陡增
2. 并发客户会在短时间新增订单，导致数据库的修改压力也会陡增

想下，数据库的并发压力是有限的，哪怕是后面提供了读写分离，对于数据库来说能承受的压力也是有限。

所以对于秒杀的设计来说总体就是减轻数据库的压力，从查询和新增来说。

秒杀查询

请求地址：

/mall/api/killgoodsSpec/queryByPage

对应的具体方法：

cn.enjoy.mall.web.service.KillGoodsService#queryByPage



这里用到了redis缓存，首先从redis里面查询，如果查询到了直接返回，缓存里面没有直接去数据库查询

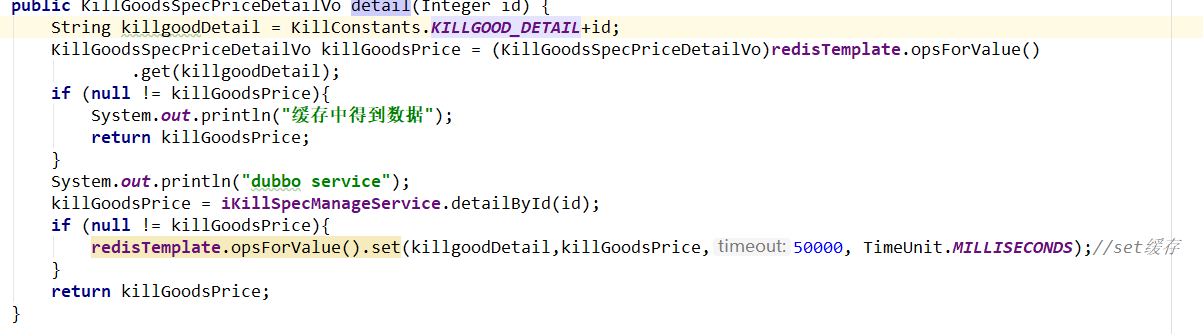
秒杀详情页

请求地址：

/mall/api/killgoodsSpec/detail?id=14

对应的具体方法：

cn.enjoy.mall.web.service.KillGoodsService#detail



一样，首先从缓存查询，查询到了返回，否则从数据库查询，查询后把数据库得内容丢到缓存里

秒杀



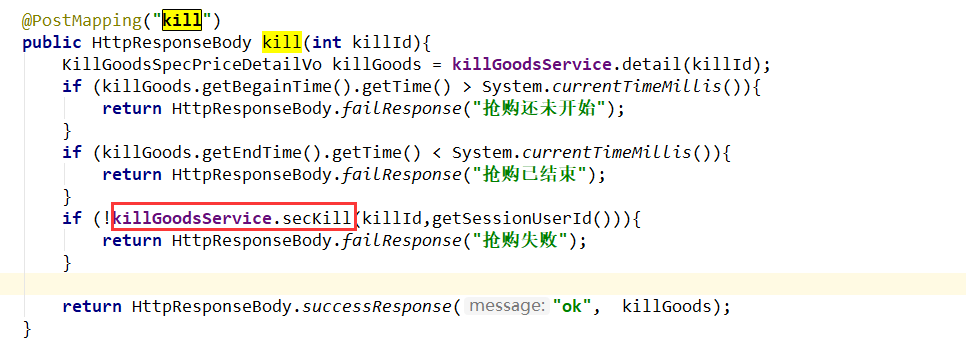
请求地址：

Post /mall/api/killgoodsSpec/kill

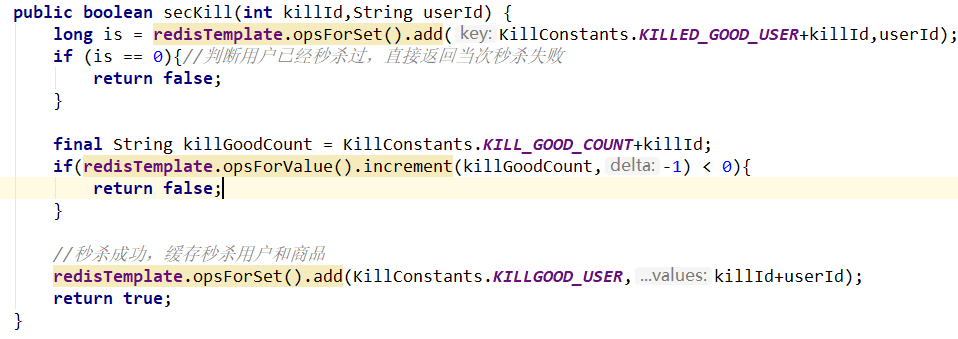
**参数：killId**

对应的具体方法：

cn.enjoy.mall.web.controller.KillGoodsContoller#kill



最后秒杀的方法



首先检查你是否秒杀过，如果秒杀过不让继续秒杀，直接返回false

然后对秒杀的产品数目-1，如果商品数目不够返回false

秒杀成功后，缓存用户的秒杀信息

秒杀结算



接口：

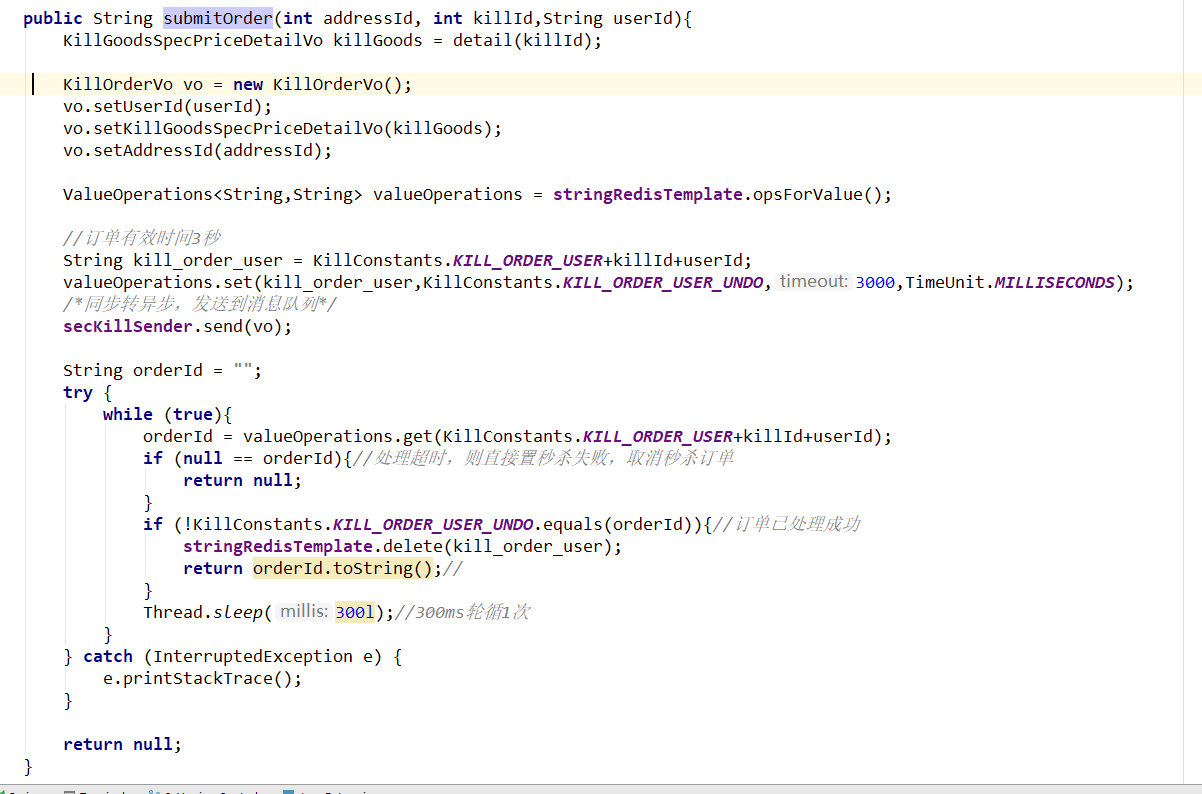
/mall/api/killgoodsSpec/submit



首先检查下当前付款的用户是否秒杀过，如果没有秒杀过，证明是非法访问，直接返回

接下来重点是

cn.enjoy.mall.web.service.KillGoodsService#submitOrder



1. 把相关信息（秒杀ID+用户ID）作为Key传入Reids，值为订单ID，但是现在订单ID还没有产生，所以现在redis的订单ID为一个固定的值1，代表还没生成秒杀订单。

2.redis的值设置好后就把秒杀的请求交给消息队列（rabbitmq）,等服务消费方进行处理

3.循环遍历，300ms轮询一次如果服务方处理完毕返回订单，如果3秒还没返回取消订单

服务的消费方

cn.enjoy.mall.service.mq.SecKillReceiver#onMessage

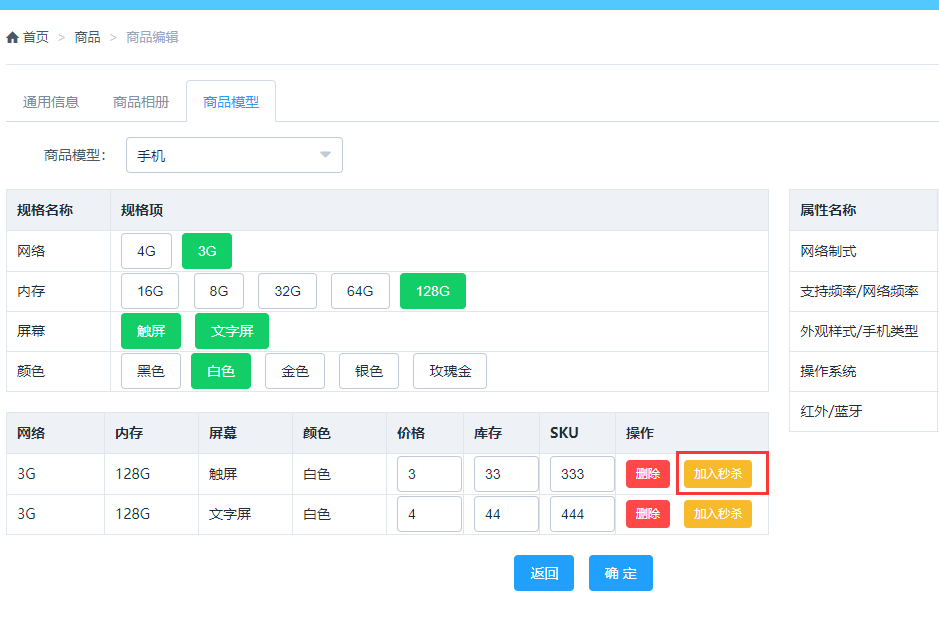


往前面说过的tp\_order，tp\_order\_goods，tp\_order\_action等表插入数据，处理完成后，把生成的订单id放入redis

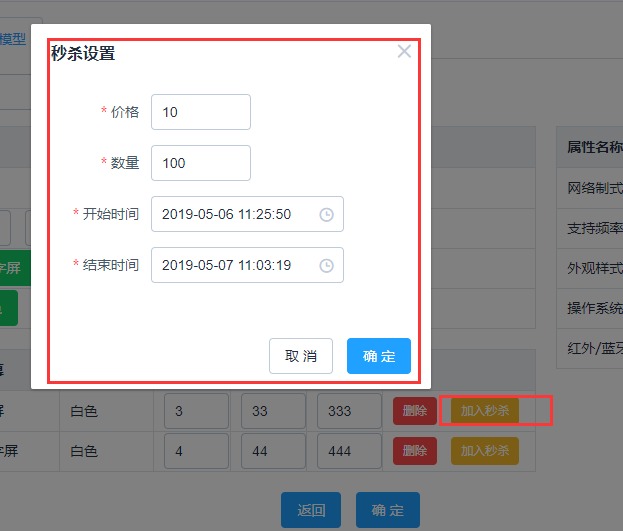
#### 业务人员增加秒杀

客户要有能秒杀的商品，首先商家必须增加秒杀的活动，这写内容是在后台增加的。

新增秒杀



在商品编辑页面，可以对已经有的商品进行秒杀

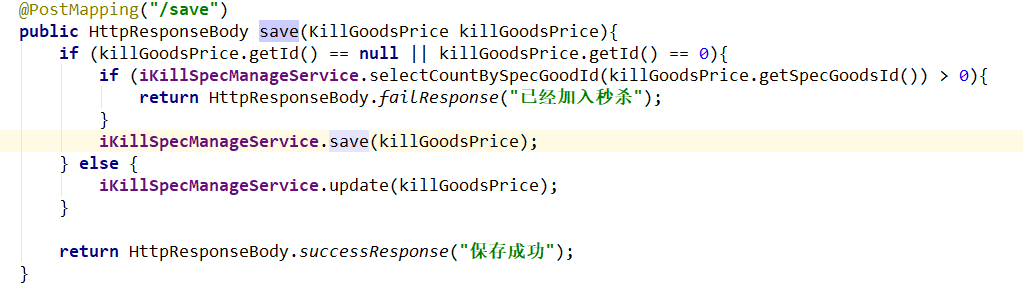


点击保存后调用的接口：

/mall-mng/api/killgoodsSpec/save

对于的方法：

cn.enjoy.mall.web.controller.KillSpecController#save



cn.enjoy.mall.service.impl.manage.KillGoodsManageServiceImpl#save



首先往秒杀表tp\_kill\_goods\_price新增一条记录，然后清理缓存，把秒杀的商品id以及秒杀的数目信息放到redis缓存中，这样客户就可以秒杀了。

秒杀维护

在商品管理>>秒杀管理里面可以对已经添加的秒杀进行管理



定时任务删除过期秒杀：

cn.enjoy.mall.service.impl.ScheduledServiceImpl#scheduledKill

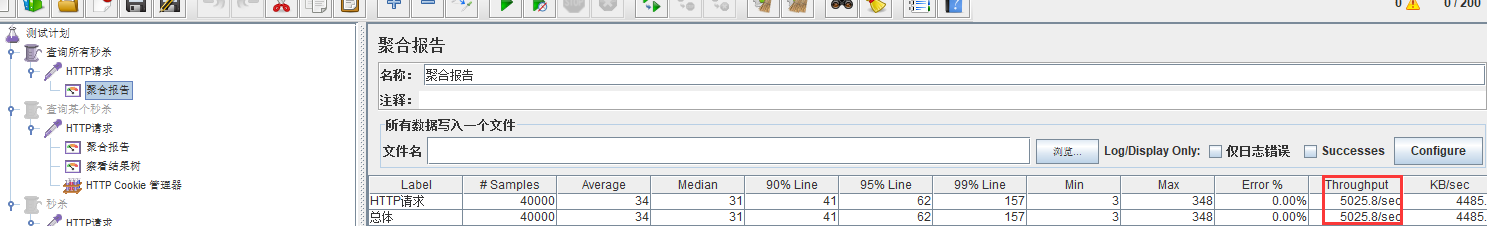


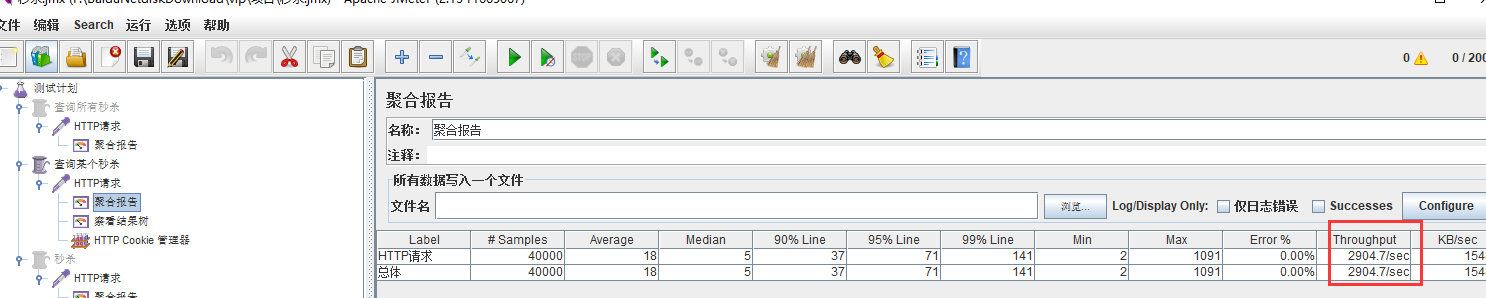
首先去tp\_kill\_goods\_price查询当前已失效的秒杀ids，根据ids把tp\_kill\_goods\_price的状态修了

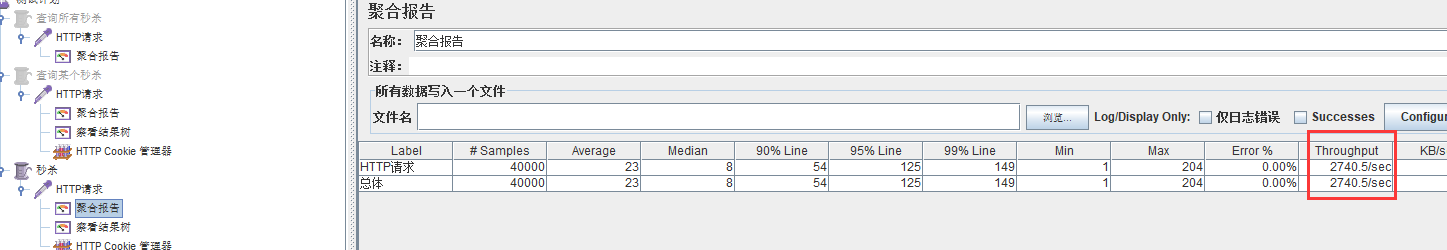
#### 秒杀压力测试：

秒杀目前3个接口，可以用来做下秒杀的压力测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **业务功能** | **并发用户** | **迭代次数** | **QPS/TPS** | **成功率** |
| 1 | 查询秒杀商品  /mall/api/killgoodsSpec/queryByPage | 200 | 200 | 5025 | 100% |
| 2 | 查询秒杀详细  /mall/api/killgoodsSpec/detail?id=14 | 200 | 200 | 2774 | 100% |
| 3 | 点击秒杀  /mall/api/killgoodsSpec/kill | 200 | 200 | 2740 | 100% |







### oauth2单点登陆

#### oauth2 是什么？

Oauth 是一个关于授权（authorization）的开网络标准，它允许第三方（小三），在你允许的时间，访问你的系统（你家）

其实访问你家的方式最常用的方式是通过钥匙（用户名，密码），但钥匙是不能随便给第三方的，万一某天你不想第三方的访问（老婆回家），，如果修改钥匙（密码），老婆也不能访问，如果第三方有钥匙，将会导致一系列的问题

就这需要一个替代钥匙的方案，比如我可以升级家里的锁（指纹锁），指纹锁首先是有钥匙的，这个钥匙就是用户名密码，这是不会泄露给第三方，而第三方通过指纹（令牌）可以访问系统，在这个时候，我可以指定某个特定指纹的有效期，在某种特殊情况下也能删除某个指纹，达到系统更安全的目的

这个例子其实说明白了，OAuth 就是一种授权机制。数据的所有者告诉系统，同意授权第三方应用进入系统，获取这些数据。系统从而产生一个短期的进入令牌（token），用来代替密码，供第三方应用使用。

#### oauth2 的授权方式

前面已经知道对于第三方来说，要能访问各种资源重点是要获取令牌（token），但根据令牌的获取方式不同，又会有四种授权方式

* 授权码（authorization-code）
* 隐藏式（implicit）
* 密码式（password）
* 客户端凭证（client credentials）

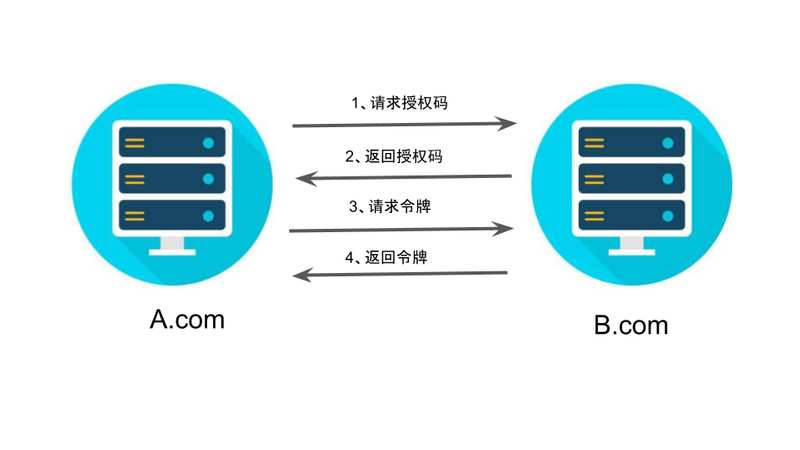
**授权码：**这是最常用的一种方式，指的是第三方应用先申请一个授权码，然后再用该码获取令牌，项目中用的就是这种

**隐藏式：**允许直接向前端颁发令牌。这种方式没有授权码这个中间步骤，所以称为（授权码）"隐藏式"（implicit），一般应用于纯前端项目

**密码式：**直接通过用户名和密码的方式申请令牌，这方式是最不安全的方式

**凭证式：**这种方式的令牌是针对第三方应用，而不是针对用户的，既某个第三方应用的所有用户共用一个令牌，一般用于没有前端的命令行应用

#### 授权码授权流程



**第一步，A 网站提供一个链接，用户点击后就会跳转到 B 网站，授权用户数据给 A 网站使用**

http://b.com/oauth/authorize?

response\_type=code&

client\_id=CLIENT\_ID&

redirect\_uri=CALLBACK\_URL&

scope=read

**第二步，用户跳转后，B 网站会要求用户登录，然后询问是否同意给予 A 网站授权。用户表示同意，这时 B 网站就会跳回redirect\_uri参数指定的网址**

**h**ttp://a.com/callback?code=AUTHORIZATION\_CODE

**第三步，A 网站拿到授权码以后，就可以在后端，向 B 网站请求令牌。**

http://b.com/oauth/token?

client\_id=CLIENT\_ID&

client\_secret=CLIENT\_SECRET&

grant\_type=authorization\_code&

code=AUTHORIZATION\_CODE&

redirect\_uri=CALLBACK\_URL

上面 URL 中，client\_id参数和client\_secret参数用来让 B 确认 A 的身份（client\_secret参数是保密的，因此只能在后端发请求），grant\_type参数的值是AUTHORIZATION\_CODE，表示采用的授权方式是授权码，code参数是上一步拿到的授权码，redirect\_uri参数是令牌颁发后的回调网址。

**第四步，B 网站收到请求以后，就会颁发令牌。具体做法是向redirect\_uri指定的网址，发送一段 JSON 数据。**

{

"access\_token":"ACCESS\_TOKEN",

"info":{...}

}

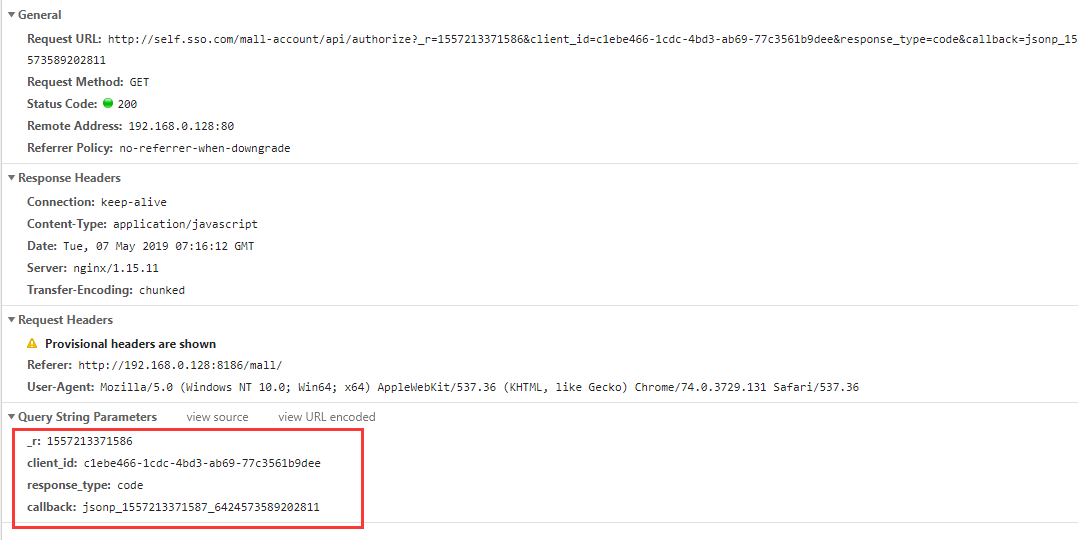
接下来用户就可以根据这个access\_token来进行访问了

#### 代码实现

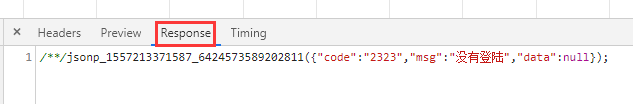
请求授权码

第一次访问前端： [/mall/#/home](http://192.168.0.128:8186/mall/" \l "/home)界面的时候会发送一个链接

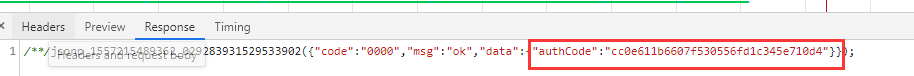
/mall-account/api/authorize?\_r=1557213371586&client\_id=c1ebe466-1cdc-4bd3-ab69-77c3561b9dee&response\_type=code&callback=jsonp\_1557213371587\_6424573589202811



返回结果：



目前你还提供身份证明，所以肯定不会发授权码，登陆系统后



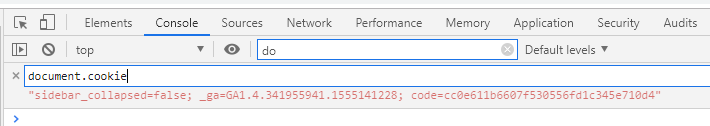
对于的具体方法:

cn.enjoy.sys.controller.AuthorizeController#authorize

返回保存授权码

对于第三方可以自行选择怎么处理授权码，但本项目中会通过回调把验证码保存到cookie中

document.cookie



根据code获取令牌

cn.enjoy.sys.security.OAuth2AuthenticationFilter#createToken

这是个认证过滤器



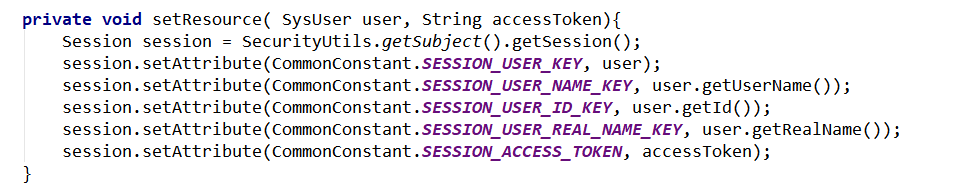
从cookie里面取出前面保存的code，封装成OAuth2Token，注意，这里的token并不是最后需要的token，仅仅是个过度对象

有了这个封装了code的对象后

cn.enjoy.sys.security.OAuth2Realm#doGetAuthenticationInfo



保存Token



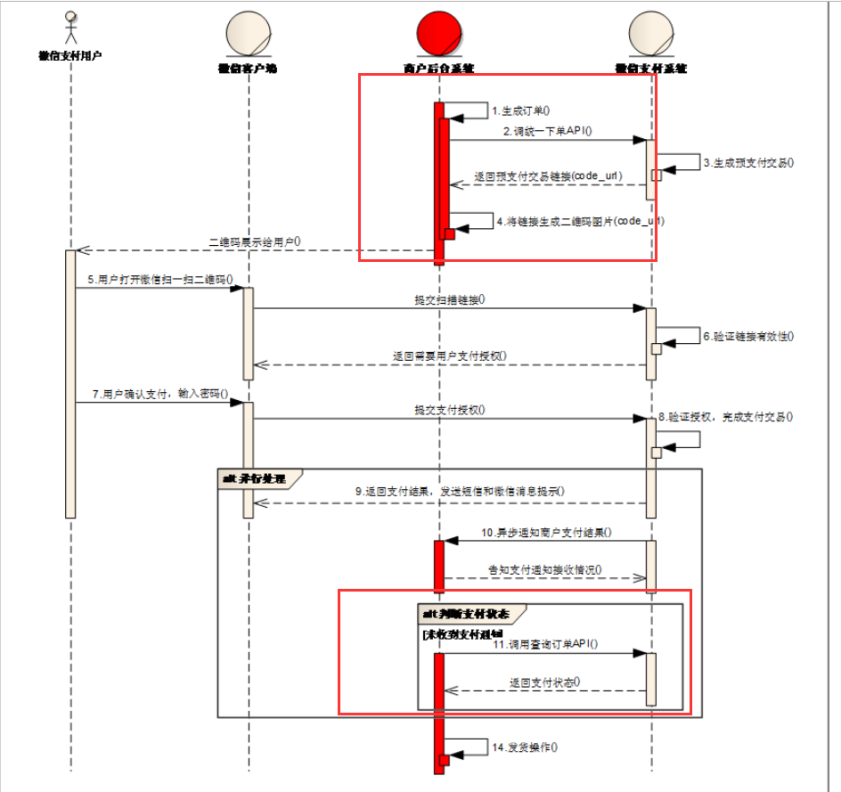
最后保存token到session中

### 支付

#### 微信支付

这里支付，用秒杀为例，首先来看个微信支付的流程图

<https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=6_5>



对于微信支付，主要就这两步

1. 调用支付微信的支付接口，传入金额，订单信息，等微信的返回（二维码等）
2. 客户支付完成后，等回调，微信会通过回调机制告诉商城系统，是否支付成功

请求调用预支付（二维码）

接口地址：

/mall/api/pay/wxPrePay



返回结果





对应具体接口

cn.enjoy.mall.web.controller.PayController#wxPrePay

在controller里面又会去调用

cn.enjoy.mall.service.impl.PayServiceImpl#doPrePay



这里就到了微信sdk里面的内容

<https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=11_1>

里面的对应的微信接口文档

<https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_1>

支付状态回调

在做前面一步操作的时候有个回调的地址必须传

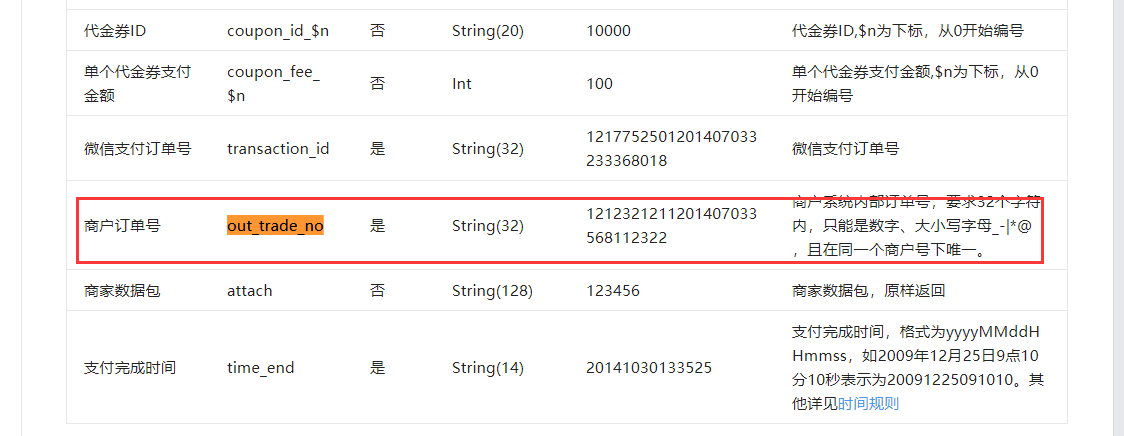


这个接口告诉微信当支付成功后，结果应该告诉的地址是哪个，而且必须是个公网接口

notify\_url的具体接口

cn.enjoy.mall.NotifyController#payNotifyUrl，这方法也很简单，代理在SDK里面都有

<https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_7&index=8>



### 物理部署



|  |  |  |
| --- | --- | --- |
| 编号 | 说明 | 部署信息 |
| 1 | Nginx前端分流，屏蔽端口信息 | 各个节点的服务器Ip/端口，安装目录等 |
| 2 | mall-web-order，客户访问得界面 |  |
| 3 | mall-web-mng，业务管理使用得界面 |  |
| 4 | Zk服务注册与发现，分布式锁 |  |
| 5 | 用户服务 |  |
| 6 | 订单服务 |  |
| 7 | 产品服务 |  |
| 8 | Redis服务器，秒杀 |  |
| 9 | mongdb服务器，产品数据缓存 |  |
| 10 | Mysql服务器 |  |
| 11 | Fastdfs 文件服务器 |  |
| 12 | Rabbtmq |  |

### API文档

web模块共用API

<http://mi.xiangxueketang.cn/apidoc/mall-web/>

小米商城后端API

<http://mi.xiangxueketang.cn/apidoc/mall-web-mng/>

小米商城前端API

<http://mi.xiangxueketang.cn/apidoc/mall-web-order/>

订单模块API

<http://mi.xiangxueketang.cn/apidoc/mall-order/>

秒杀订单模块API

<http://mi.xiangxueketang.cn/apidoc/mall-kill/>

用户模块API

<http://mi.xiangxueketang.cn/apidoc/mall-user/>

产品模块API

<http://mi.xiangxueketang.cn/apidoc/mall-product/>

支付模块API

<http://mi.xiangxueketang.cn/apidoc/mall-pay/>

一点登录模块API

http://mi.xiangxueketang.cn/apidoc/mall-account/