**A22**

**银行产品秒杀系统设计**

|  |  |
| --- | --- |
| 团队编号: | 2102645 |
| 队 名: | BearBrick |
|  |  |

2022.4.7

目录

[1.环境 4](#_Toc100306867)

[1.1 技术点 4](#_Toc100306868)

[1.2 目录说明 4](#_Toc100306869)

[2.运维 4](#_Toc100306870)

[3.数据库 4](#_Toc100306871)

[3.1 表介绍 4](#_Toc100306872)

[3.2 建表语句 4](#_Toc100306873)

[3.3 主要字段 4](#_Toc100306874)

[4.接口 5](#_Toc100306875)

[4.1 接口列表 5](#_Toc100306876)

[4.2 接口详情 5](#_Toc100306877)

[5.业务 6](#_Toc100306878)

[5.1 基本业务流程描述 7](#_Toc100306879)

[5.2 脚本防刷功能实现说明 7](#_Toc100306880)

[5.3 秒杀地址隐藏功能实现说明 7](#_Toc100306881)

[5.4 初筛功能实现说明 7](#_Toc100306882)

[5.5 秒杀功能实现说明 7](#_Toc100306883)

[6.亮点 7](#_Toc100306884)

[6.1 xx 7](#_Toc100306885)

[6.2 xx 7](#_Toc100306886)

[7.测试 7](#_Toc100306887)

[8.源码 8](#_Toc100306888)

[9.其他 8](#_Toc100306889)

## 1.环境

### 1.1 技术点

本项目采用SpringBoot 2.5.6来快速构建项目整体架构，使用MySQL 8来进行数据的存储，使用Redis 6.0.6来做缓存，使用RabbitMQ 3.9.1来异步发送消息，使用MyBatis-Plus 3.4 3.4来做ORM的映射。采用Docker 容器化部署。

在前端模块采用Thymeleaf、BootStrap来做数据渲染和页面展示。

本项目，在处理用户大量的请求系统登录的时候，我们会提前将用户的信息放在缓存（本地缓存和Redis缓存）中，来减少对数据库的访问，以便后面对用户的其他操作。秒杀的时候，做了验证码和安全校验，起到限流作用和防止同一ID或IP重复购买。同时如果同一ID频繁的请求，我们会直接拒绝该用户的请求。同时也会对秒杀的用户又一个状态的判断，每个用户在登陆我们的系统，都会携带UUID，也就是”ticket”，并设置了一个相应的过期时间。如果到了过期时间，系统会直接强制下线。

为保证秒杀接口的安全性，防止其他开发人员恶意透露秒杀链接地址，提前秒杀商品，在获取秒杀接口的我们做了接口地址隐藏，我们生成秒杀接口的地址，是根据随机UUID来拼接。同时在点击秒杀接口的时候，要求用户输入数字计算验证码，将所有的请求分到不同的时间片段上，来达到分散用户的请求的效果。

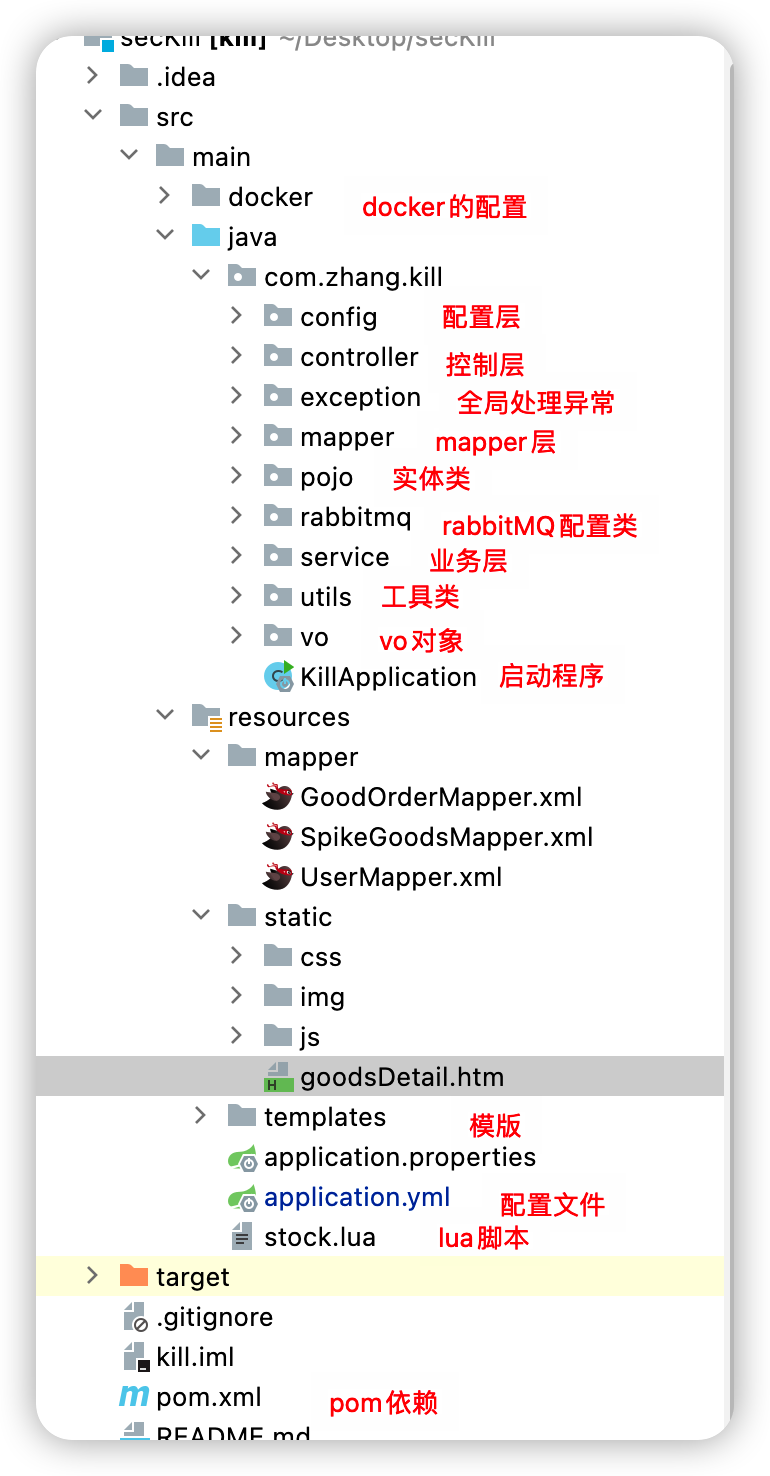
同时在秒杀的时候，会有一个对用户的筛选。满足条件的人才可以，继续后续的操作。

在后续的秒杀执行流程中，我们采用了Redis的单线程的特性，避免不必要的上下文切换和竞争条件、同时它的大部分请求都是纯粹的内存的操作，速度很快、同时它是一个非阻塞I/O多路复用，在I/O操作上浪费的时间很少。当用户点击了秒杀按钮，后台业务会去Redis的缓存中查询库存的数量，利用缓存可以减少服务器的响应时间，提高用户的体验度。

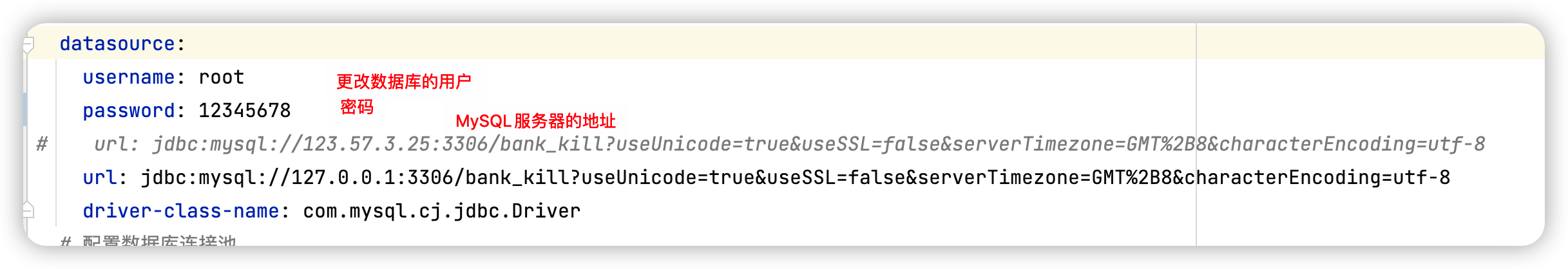
如果缓存中的库存还有剩余，会利用Redis的单线程去预减库存，这里的减库存是原子性的操作。同时为了减轻Redis的压力使用了Map标记库存。如果库存不足，会给用户友好的提示。同时为了防止用户重复购买，也增加一个判断，同时给用户友好的提示。

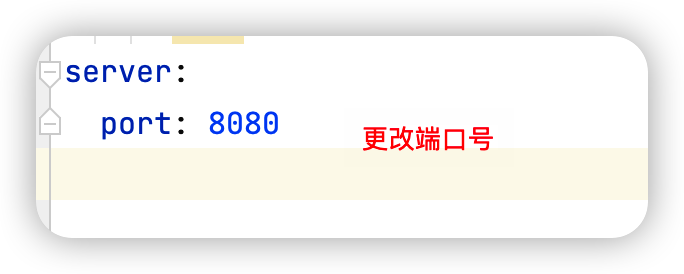
如果上述操作，都没有问题。在生成订单的过程，我们会利用RabbitMQ异步队列生成对应的订单，同时利用RabbitMQ的交换机模型，给用户一个提示，提醒用户支付，如果用户支付成功，才去数据库中真正的扣减库存。

### 1.2 目录说明



## 运维









## 3.数据库

### 3.1 表介绍

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 表名 | 中文名 | 主要字段 | 主要字段描述 | 备注 |
| 1 | user | 用户表 | user\_card、user\_name、user\_password、user\_age、is\_Employment | 主键id（用户身份证）、用户名、用户密码、用户的年龄、是否在职（0 是 1不是） | 用户表 |
| 2 | good\_order | 订单表 | id、user id、good\_id、good\_name、good\_count、good\_price、status、create date、pay\_date | 主键id（订单id）、用户id、商品id、商品名称、商品数量、商品单价、支付状态（0=新建未支付 1=已支付 2=已退款 3=已发货 4=已收货 5=已完成）、订单的创建时间、支付时间 | 订单表 |
| 3 | spike\_goods | 秒杀产品表 | id、good\_num、good\_money、good\_img good\_detail、start\_date、end\_date | 主键id、商品名称、商品库存、秒杀价、商品图片、商品详情、秒杀开始时间、秒杀结束时间 |  |

### 3.2 建表语句

建表语句，放在文件夹的sql目录下。

### 3.3 主要字段

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段中文名 | 字段所在表格 | 字段英文名 | 备注 |
| 1 | 用户账户 | user | user\_card | 用户的账号值为user\_card的字段 |
| 2 | 用户余额 | account | fin\_balance | 无 |
| 3 | 内部户账户 | user | user\_name | user\_name值为admin的字段 |
| 4 | 用户名 | user | user\_name | user\_name值为admin的字段 |
| 5 | 密码 | user | user\_password | user\_password值为1234567 |
| 6 | 订单号 | good\_order | id | id |

## 4.接口

### 4.1 接口列表

列出本次业务流程所使用的接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 路径 | 请求方式 | 备注 |
| 1 | 用户登录 | 52.83.60.43:8080/login/toLogin | GET | 页面跳转 |
| 2 | 用户登录 | 52.83.60.43:8080/login/doLogin | POST | 用户点击实现登录 |
| 3 | 用户注册 | 52.83.60.43:8080/login/register | POST | 用户注册 |
| 4 | 产品展示 | 52.83.60.43:8080/goods/toList | GET | 产品展示页面 |
| 5 | 秒杀详情页 | 52.83.60.43:8080/goods/toDetail | GET | 秒杀详情页 |
| 6 | 秒杀接口 | 52.83.60.43:8080/seckill/doSeckill | POST | 用户秒杀 |
| 7 | 订单接口 | 52.83.60.43:8080/seckill/result | GET | 生成订单 |
| 8 | 秒杀地址接口 | 52.83.60.43:8080/seckill/path | GET | 点击秒杀，获取秒杀地址 |
| 9 | 验证码接口 | 52.83.60.43:8080/seckill/captcha | GET | 获取验证码 |

### 4.2 接口详情

描述上方接口详情

#### 4.2.1 用户登录接口

登陆页面

##### 1.描述

接口地址：/login/doLogin

请求方式：POST

Consumes:[“application/json”]

Produces:[“\*/\*”]

##### 2.地址

52.83.60.43:8080/login/doLogin

##### 3.请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 中文名 | 参数名 | 类型[长度限制] | 必填 | 描述 |
| 身份证号 | idcard | Interger(64) | 是 | 用户的id |
| 密码 | password | String(64) | 是 | 密码 |

##### 4.返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 中文名 | 参数名 | 类型[长度限制] | 必填 | 描述 |
| 状态码 | code | integer(int64) | 是 | 状态码 |
| 消息 | messgae | string | 是 | 消息体 |
| 对象 | object | object | 是 | 对象 |

##### 5.请求示例

{

"card": "1234567",

"password": "123456"

}

##### 6.返回示例

{  
"code":" ",

"message":" ",

"object":" "

}

##### 7.码值枚举(若有可写)

SUCCESS(200,"SUCCESS"),

ERROR(500,"服务端异常")

EMPTY\_STOCK(500500,"库存不足"),

ONLY\_ONE(500501,"每人限购一份"),

LOGIN\_ERROR(500210,"用户名或密码不正确"),

BIND\_ERROR(500211,"参数校验异常"),

REQUEST\_ILLEGAL(500213,"请求非法,请重新尝试"),

CAPTCHE\_ERROR(500250,"验证码输入错误"),

ACCESS\_LIMIT(501232,"访问过于频繁，请稍后再试"),

SESSION\_ERROR(500212,"用户不存在");

#### 4.2.2 注册接口

#### 4.2.2 用户注册接口

用户注册

##### 1.描述

接口地址：/login/register

请求方式：POST

Consumes:[“application/json”]

Produces:[“\*/\*”]

##### 2.地址

52.83.60.43:8080/login/register

##### 3.请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 中文名 | 参数名 | 类型[长度限制] | 必填 | 描述 |
| 身份证号 | card | Interger(int64) | 是 | 用户的id |
| 密码 | password | String(64) | 是 | 密码 |
| 用户名 | username | String(64) | 是 | 用户名 |
| 确认密码 | repassword | String(64) | 是 | 确认密码 |

##### 4.返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 中文名 | 参数名 | 类型[长度限制] | 必填 | 描述 |
| 状态码 | code | integer(int64) | 是 | 状态码 |
| 消息 | messgae | string | 是 | 消息体 |
| 对象 | object | object | 是 | 对象 |

##### 5.请求示例

{

"card": "1234567",

"username": "user",

"password": "123456",

"repassword": "123456",

}

##### 6.返回示例

{  
"code":" ",

"message":" ",

"object":" "

}

##### 7.码值枚举(若有可写)

##### 写)

##### )

SUCCESS(200,"SUCCESS"),

ERROR(500,"服务端异常")

EMPTY\_STOCK(500500,"库存不足"),

ONLY\_ONE(500501,"每人限购一份"),

LOGIN\_ERROR(500210,"用户名或密码不正确"),

BIND\_ERROR(500211,"参数校验异常"),

REQUEST\_ILLEGAL(500213,"请求非法,请重新尝试"),

CAPTCHE\_ERROR(500250,"验证码输入错误"),

ACCESS\_LIMIT(501232,"访问过于频繁，请稍后再试"),

SESSION\_ERROR(500212,"用户不存在");

#### 4.2.5 秒杀接口

#### 4.2.5 秒杀接口

秒杀接口

##### 1.描述

接口地址：/seckill/doSeckill

请求方式：POST

Consumes:[“application/json”]

Produces:[“\*/\*”]

##### 2.地址

52.83.60.43:8080/seckill/doSeckill

##### 3.请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 中文名 | 参数名 | 类型[长度限制] | 必填 | 描述 |
| 产品ID | goodsId | Integer(int64) | 是 | 产品id |

##### 4返回参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 中文名 | 参数名 | 类型[长度限制] | 必填 | 描述 |
| 状态码 | code | integer(int64) | 是 | 状态码 |
| 消息 | messgae | string | 是 | 消息体 |
| 对象 | object | object | 是 | 对象 |

##### 5.请求示例

{

"goodsId":" "

}

##### 6.返回示例

{  
"code":" ",

"message":" ",

"object":" "

}

##### 7.码值枚举(若有可写)

##### 7.码值枚举(若有可写)

##### 枚举(若有可写)

##### 码值枚举(若有可写)

SUCCESS(200,"SUCCESS"),

ERROR(500,"服务端异常")

EMPTY\_STOCK(500500,"库存不足"),

ONLY\_ONE(500501,"每人限购一份"),

LOGIN\_ERROR(500210,"用户名或密码不正确"),

BIND\_ERROR(500211,"参数校验异常"),

REQUEST\_ILLEGAL(500213,"请求非法,请重新尝试"),

CAPTCHE\_ERROR(500250,"验证码输入错误"),

ACCESS\_LIMIT(501232,"访问过于频繁，请稍后再试"),

SESSION\_ERROR(500212,"用户不存在");

## 5.业务

### 基本业务流程描述

##### 5.1.1登录模块：

通过注册按钮完成用户注册：



注册成功后，进入登录模块：



##### 5.1.2商品展示模块：

展示秒杀商品列表：

点击秒杀详情可查看有关商品秒杀有关信息：



输入计算验证码，就可以进行秒杀。

### 5.2 脚本防刷功能实现说明

无

### 5.3 秒杀地址隐藏功能实现说明

实现原理：

基于UUID生成随机的秒杀地址，当用户点击秒杀按钮后，后端生成随机的秒杀地址，然后返回给前端，页面重定向进行秒杀。

### 5.4 初筛功能实现说明

无

### 5.5 秒杀功能实现说明

Boolean check = goodOrderService.checkPath(user, goodsId, path);  
//限购判断  
if (!check) {  
 return RespBean.*error*(RespBeanEnum.*REQUEST\_ILLEGAL*);  
}  
GoodOrder goodOrder = (GoodOrder) opsForValue.get("order:" + user.getUserCard() + ":" + goodsId);  
if (goodOrder != null) {  
 return RespBean.*error*(RespBeanEnum.*ONLY\_ONE*);  
}  
if (EmptyMap.get(goodsId)) {  
 return RespBean.*error*(RespBeanEnum.*EMPTY\_STOCK*);  
}  
//每调用一次，库存减一，原子性的  
Long stock = opsForValue.decrement("seckillGoods:" + goodsId);  
if (stock < 0) {  
 EmptyMap.put(goodsId, true);  
 opsForValue.increment("seckillGoods:" + goodsId);  
 return RespBean.*error*(RespBeanEnum.*EMPTY\_STOCK*);  
}  
SecKillMessage message = new SecKillMessage(user, goodsId);  
mqSender.send(JsonUtils.*obj2String*(message));  
return RespBean.*success*(0);

数据预热在Redis中，秒杀请求后先去数据库做限购判断，当都满足要求时，使用redis进行库存原子性的预减，然后异步到消息队列RabbitMQ中，进行数据库库存减少。

## 6.亮点

### 6.1 Redis预减库存，减少Mysql数据压力：

项目启动时，进行库存预热，将Mysql中的数据预先存入redis中，秒杀请求请求到Redis中，进行库存预减。

### 6.2 RabbitMQ进行异步秒杀处理：

Redis预见库存后，将真正的秒杀异步到RabbitMQ去处理，提高系统并发量，增强用户体验，减轻数据库压力。

### 6.3 ：页面静态化：

商品列表页面数据交互较少，将整个页面存入Redis，页面静态化，加快渲染速度，提高并发量。

### 6.4：秒杀验证码分流：

点击秒杀按钮时，需要进行验证码验证，起到分流作用，减轻Redis压力。

## 7.测试

无