**品优购电商系统开发**

**第19章**

**秒杀解决方案**

传智播客.黑马程序员

# 课程目标

目标1：能够说出秒杀实现思路

目标2：实现秒杀频道首页功能

目标3：实现秒杀商品详细页功能

目标4：实现秒杀下单功能

目标5：实现秒杀支付功能

# 1.秒杀业务分析

## 1.1需求分析

所谓“秒杀”，就是网络[卖家](https://baike.baidu.com/item/%E5%8D%96%E5%AE%B6)发布一些超低价格的商品，所有买家在同一时间网上抢购的一种销售方式。通俗一点讲就是网络商家为促销等目的组织的网上限时抢购活动。由于商品价格低廉，往往一上架就被抢购一空，有时只用一秒钟。

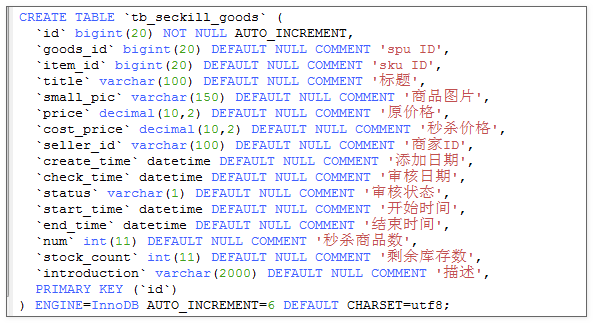
秒杀商品通常有两种限制：库存限制、时间限制。

需求：

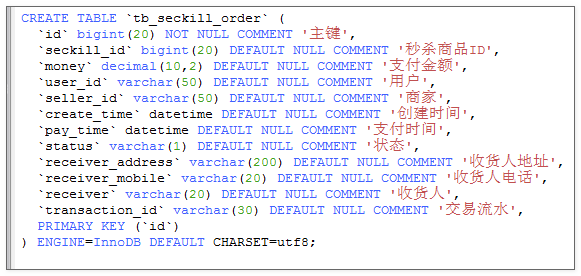
1. 商家提交秒杀商品申请，录入秒杀商品数据，主要包括：商品标题、原价、秒杀价、商品图片、介绍等信息
2. 运营商审核秒杀申请
3. 秒杀频道首页列出秒杀商品（进行中的）点击秒杀商品图片跳转到秒杀商品详细页。
4. 商品详细页显示秒杀商品信息，点击立即抢购实现秒杀下单，下单时扣减库存。当库存为0或不在活动期范围内时无法秒杀。
5. 秒杀下单成功，直接跳转到支付页面（微信扫码），支付成功，跳转到成功页，填写收货地址、电话、收件人等信息，完成订单。
6. 当用户秒杀下单5分钟内未支付，取消预订单，调用微信支付的关闭订单接口，恢复库存。

## 1.2数据库表分析

Tb\_seckill\_goods 秒杀商品表



Tb\_seckill\_order 秒杀订单表



## 1.3秒杀实现思路

秒杀技术实现核心思想是运用缓存减少数据库瞬间的访问压力！读取商品详细信息时运用缓存，当用户点击抢购时减少缓存中的库存数量，当库存数为0时或活动期结束时，同步到数据库。 产生的秒杀预订单也不会立刻写到数据库中，而是先写到缓存，当用户付款成功后再写入数据库。

第一：数据库表单独设计，数据库独立。 独立的系统。

第二：可以使用redis存储秒杀数据，秒杀的订单的数据。(单线程的)

## 1.4工程搭建与准备

### 1.4.1工程模块搭建

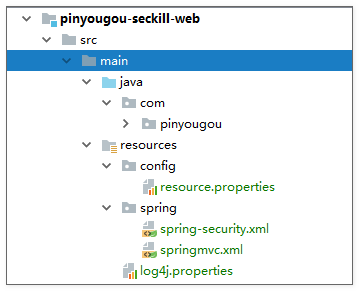
（1）创建秒杀服务接口模块 pinyougou-seckill-interface ，依赖pinyougou-pojo

（2）创建秒杀服务模块pinyougou-seckill-service (war)，pom.xml引入依赖参见其它服务工程,依赖 pinyougou-seckill-interface ， Tomcat7插件运行端口为9009。添加web.xml、 spring 配置文件参见其它服务工程, dubbox的端口为20889。

（3）创建秒杀频道web模块 pinyougou-seckill-web(war) pom.xml引入依赖参见cart\_web工程(需添加单点登录和权限控制)，依赖 pinyougou-seckill-interface ，Tomcat7插件运行端口为9109 添加web.xml、 spring 配置文件参见cart\_web工程。

将秒杀相关的页面及资源拷贝到此模块。添加angularJS.

注意：表现层系统需要整合CAS客户端。



### 1.4.2代码生成

运用《黑马程序员代码生成器》生成代码，拷入工程

## 1.5秒杀商品后台管理（学员完成）

运营商系统web模块pinyougou-manager-web依赖 pinyougou-seckill-interface

商家系统web模块pinyougou-shop-web依赖pinyougou-seckill-interface

### 1.5.1商家后台

1. 秒杀商品列表
2. 秒杀商品申请
3. 秒杀订单查询

### 1.5.2运营商后台

1. 待审核秒杀商品列表
2. 秒杀商品审核

（3）秒杀订单查询

# 2.品优购-秒杀频道首页

## 2.1需求分析

秒杀频道首页，显示正在秒杀的商品（已经开始，未结束的商品）

## 2.2后端代码

### 2.2.1服务接口层

（1）修改pinyougou-seckill-interface的SeckillGoodsService.java

|  |
| --- |
| /\*\*  \* 返回当前正在参与秒杀的商品  \* **@return**  \*/  **public** List<TbSeckillGoods> findList(); |

### 2.2.2服务实现层

修改pinyougou-seckill-service的SeckillGoodsServiceImpl.java

|  |
| --- |
| @Override  **public** List<TbSeckillGoods> findList() {  TbSeckillGoodsExample example=**new** TbSeckillGoodsExample();  Criteria criteria = example.createCriteria();  criteria.andStatusEqualTo("1");//审核通过  criteria.andStockCountGreaterThan(0);//剩余库存大于0  criteria.andStartTimeLessThanOrEqualTo(**new** Date());//开始时间小于等于当前时间  criteria.andEndTimeGreaterThan(**new** Date());//结束时间大于当前时间  **return** seckillGoodsMapper.selectByExample(example);  } |

### 2.2.3控制层

修改pinyougou-seckill-web的SeckillGoodsController.java

|  |
| --- |
| /\*\*  \* 当前秒杀的商品  \* **@return**  \*/  @RequestMapping("/findList")  **public** List<TbSeckillGoods> findList(){  **return**seckillGoodsService.findList();  } |

## 2.3前端代码实现

### 2.3.1服务层

在pinyougou-seckill-web创建 seckillGoodsService.js

|  |
| --- |
| //服务层  app.service('seckillGoodsService',**function**($http){  //读取列表数据绑定到表单中  **this**.findList=**function**(){  **return** $http.get('/seckillGoods/findList.do');  }  }); |

### 2.3.2控制层

在pinyougou-seckill-web创建seckillGoodsController.js

|  |
| --- |
| //控制层  app.controller('seckillGoodsController' ,**function**($scope,seckillGoodsService){  //读取列表数据绑定到表单中  $scope.findList=**function**(){  seckillGoodsService.findList().success(  **function**(response){  $scope.list=response;  }  );  }  }); |

### 2.3.3页面

修改seckill-index.html,引入js

|  |
| --- |
| <scripttype=*"text/javascript"*src=*"plugins/angularjs/angular.min.js"*></script>  <scripttype=*"text/javascript"*src=*"js/base.js"*></script>  <scripttype=*"text/javascript"*src=*"js/service/seckillGoodsService.js"*></script>  <scriptsrc=*"js/controller/seckillGoodsController.js"*></script> |

指令

|  |
| --- |
| <body ng-app=*"pinyougou"*ng-controller=*"seckillGoodsController"*ng-init=*"findList()"*> |

循环列表的实现

|  |
| --- |
| <li class=*"seckill-item"*ng-repeat=*"pojo in list"*>  <div class=*"pic"*>  <imgsrc=*"{{pojo.smallPic}}"*width=*"290px"*height=*"290px"*alt=*''*>  </div>  <div class=*"intro"*><span>{{pojo.title}}</span></div>  <divclass=*'price'*><bclass=*'sec-price'*>￥{{pojo.costPrice}}</b><bclass=*'ever-price'*>￥{{pojo.price}}</b></div>  <divclass=*'num'*>  <div>已售{{ ((pojo.num-pojo.stockCount)/pojo.num\*100).toFixed(0) }}%</div>  <divclass=*'progress'*>  <divclass=*'sui-progress progress-danger'*><spanstyle='width: {{ ((*pojo.num-pojo.stockCount)/pojo.num*\**100).toFixed*(*0)* }}%;' class=*'bar'*></span></div>  </div>  <div>剩余<bclass=*'owned'*>{{pojo.stockCount}}</b>件</div>  </div>  <aclass=*'sui-btn btn-block btn-buy'*href=*'seckill-item.html#?id={{pojo.id}}'*target=*'\_blank'*>立即抢购</a>  </li> |

## 2.4缓存处理

修改pinyougou-seckill-service的SeckillGoodsServiceImpl.java

|  |
| --- |
| @Autowired  **Private** RedisTemplate redisTemplate;    @Override  **public** List<TbSeckillGoods> findList() {  //获取秒杀商品列表  List<TbSeckillGoods>seckillGoodsList = redisTemplate.boundHashOps("seckillGoods").values();  **if**(seckillGoodsList==**null** || seckillGoodsList.size()==0){  TbSeckillGoodsExample example=**new** TbSeckillGoodsExample();  Criteria criteria = example.createCriteria();  criteria.andStatusEqualTo("1");//审核通过  criteria.andStockCountGreaterThan(0);//剩余库存大于0  criteria.andStartTimeLessThanOrEqualTo(**new** Date());//开始时间小于等于当前时间  criteria.andEndTimeGreaterThan(**new** Date());//结束时间大于当前时间  seckillGoodsList= seckillGoodsMapper.selectByExample(example);  //将商品列表装入缓存  System.***out***.println("将秒杀商品列表装入缓存");  **for**(TbSeckillGoods seckillGoods:seckillGoodsList){  redisTemplate.boundHashOps("seckillGoods").put(seckillGoods.getId(), seckillGoods);  }  }  r**eturn** seckillGoodsList;  } |

# 3.品优购-秒杀详细页

## 3.1需求分析

商品详细页显示秒杀商品信息。

## 3.2显示详细页信息

### 3.2.1后端代码

修改pinyougou-seckill-interface的SeckillGoodsService

|  |
| --- |
| /\*\*  \* 根据ID获取实体(从缓存中读取)  \*/  **public** TbSeckillGoods findOneFromRedis(Long id); |

修改pinyougou-seckill-service的SeckillGoodsServiceImpl.java

|  |
| --- |
| @Override  **public** TbSeckillGoods findOneFromRedis(Long id) {  **return** (TbSeckillGoods)redisTemplate.boundHashOps("seckillGoods").get(id);  } |

修改pinyougou-seckill-web的SeckillGoodsController

|  |
| --- |
| @RequestMapping("/findOneFromRedis")  **public** TbSeckillGoods findOneFromRedis(Long id){  **return** seckillGoodsService.findOneFromRedis(id);  } |

增加超时时间设置

|  |
| --- |
| @Reference(timeout=10000)  **private** SeckillGoodsService seckillGoodsService; |

### 3.2.2前端代码

pinyougou-seckill-web 的seckillGoodsService.js

|  |
| --- |
| **this**.findOne=**function**(id){  **return** $http.get('seckillGoods/findOneFromRedis.do?id='+id);  } |
|  |

pinyougou-seckill-web 的seckillGoodsController.js ，引入$location服务

|  |
| --- |
| //查询实体  $scope.findOne=**function**(){  seckillGoodsService.findOne($location.search()['id']).success(  **function**(response){  $scope.entity= response;  }  );  } |

修改seckill-item.html ，引入js

|  |
| --- |
| <scripttype=*"text/javascript"*src=*"plugins/angularjs/angular.min.js"*></script>  <scripttype=*"text/javascript"*src=*"js/base.js"*></script>  <scripttype=*"text/javascript"*src=*"js/service/seckillGoodsService.js"*></script>  <scriptsrc=*"js/controller/seckillGoodsController.js"*></script> |

指令

|  |
| --- |
| <bodyng-app=*"pinyougou"*ng-controller=*"seckillGoodsController"*ng-init=*"findOne()"*> |

用表达式显示标题

|  |
| --- |
| <h4>{{entity.title}}</h4> |

图片

|  |
| --- |
| <spanclass=*"jqzoom"*><imgjqimg=*"{{entity.smallPic}}"*src=*"{{entity.smallPic}}"*width=*"400px"*height=*"400px"*/></span> |

价格

|  |
| --- |
| <divclass=*"fl price"*><i>¥</i>  <em>{{entity.costPrice}}</em>  <span>原价：{{entity.price}}</span>  </div> |

介绍

|  |
| --- |
| <divclass=*"intro-detail"*>{{entity.introduction}}</div> |

剩余库存

|  |
| --- |
| 剩余库存：{{entity.stockCount}} |

## 3.3秒杀倒计时效果

### 3.3.1 $interval服务简介

在AngularJS中$interval服务用来处理间歇性处理一些事情

格式为：

|  |
| --- |
| $interval(执行的函数,间隔的毫秒数,运行次数); |

运行次数可以缺省，如果缺省则无限循环执行

取消执行用cancel方法

|  |
| --- |
| $interval.cancel(time); |

我先现在先做一个简单的例子：10秒倒计时 ，首先引入$interval , 控制层编写代码如下：

|  |
| --- |
| $scope.second = 10;  time= $interval(**function**(){  **if**($scope.second>0){  $scope.second =$scope.second-1;  }**else**{  $interval.cancel(time);  alert("秒杀服务已结束");  }  },1000); |

页面用表达式显示$scope.second的值

### 3.3.2秒杀倒计时

修改seckillGoodsController.js ，实现

|  |
| --- |
| $scope.findOne=**function**(){  seckillGoodsService.findOne($location.search()[**'id'**]).success(  **function**(response){  $scope.**entity**= response;  **console**.info($scope.**entity**.**endTime**);  **allsecond** =***Math***.floor( **new** Date($scope.**entity**.**endTime**).getTime()/1000 - **new** Date().getTime()/1000);   **time**= $interval(**function**(){  $scope.**timeString** = convertTimeString(**allsecond**);  **if**(**allsecond**>0){  **allsecond** =**allsecond**-1;  }**else**{  $interval.cancel(**time**);  alert(**"秒杀服务已结束"**);  }  },1000);     }  ); }    //转换秒为 天小时分钟秒格式 XXX天 10:22:33  convertTimeString=**function**(allsecond){  **var** days= ***Math***.floor( allsecond/(60\*60\*24));*//天数* **var** hours= ***Math***.floor( (allsecond-days\*60\*60\*24)/(60\*60) );*//小数数* **var** minutes= ***Math***.floor( (allsecond -days\*60\*60\*24 - hours\*60\*60)/60 );*//分钟数* **var** seconds= allsecond -days\*60\*60\*24 - hours\*60\*60 -minutes\*60; *//秒数* **if**(days>0){  days=days+**"天 "**;  }  **if**(hours<10){  hours=**"0"**+hours;  }  **if**(minutes<10){  minutes=**"0"**+minutes;  }  **if**(seconds<10){  seconds=**"0"**+seconds;  }  **return** days+hours+**":"**+minutes+**":"**+seconds; } |

修改页面seckill-item.html ，显示time的值

|  |
| --- |
| <spanclass=*"overtime"*> 距离结束：{{timeString}}</span> |

# 4.品优购-秒杀下订单

## 4.1需求分析

商品详细页点击立即抢购实现秒杀下单，下单时扣减库存。当库存为0或不在活动期范围内时无法秒杀。 需要在redis中操作，只有抢购完成了，才更新库存到数据中。

## 4.2后端代码

### 4.2.1服务接口层

修改pinyougou-seckill-interface的SeckillOrderService.java

|  |
| --- |
| /\*\*  \* 提交订单  \* **@param** seckillId  \* **@param** userId  \*/  **public void** submitOrder(Long seckillId,String userId); |

### 4.4.2服务实现层

Spring配置文件配置IdWorker

pinyougou-seckill-service的SeckillOrderServiceImpl.java实现方法

|  |
| --- |
| @Autowired **private** RedisTemplate **redisTemplate**; @Autowired **private** TbSeckillGoodsMapper **seckillGoodsMapper**; @Autowired **private** IdWorker **idWorker**;   @Override  **public void** submitOrder(Long seckillId, String userId) {  *//从nosql数据库中获取商品* TbSeckillGoods killgoods = (TbSeckillGoods) **redisTemplate**.boundHashOps(**"seckillGoods"**).get(seckillId);   **if**(killgoods==**null**){  **throw new** RuntimeException(**"商品不存在"**);  }   **if**(killgoods.getStockCount()<=0){  *//说明商品已经没有库存了* **throw new** RuntimeException(**"商品已被抢光"**);  }   *//将这个商品的库存减少 假设只有一个 那么此时就应该* killgoods.setStockCount(killgoods.getStockCount()-1);*//减少* **redisTemplate**.boundHashOps(**"seckillGoods"**).put(seckillId,killgoods);   **if**(killgoods.getStockCount()==0){*//如果已经被秒光* **seckillGoodsMapper**.updateByPrimaryKey(killgoods);*//同步到数据库* **redisTemplate**.boundHashOps(**"seckillGoods"**).delete(seckillId);*//将redis中的该商品清除掉* }  *//创建订单* **long** orderId = **idWorker**.nextId();  TbSeckillOrder seckillOrder=**new** TbSeckillOrder();   seckillOrder.setId(orderId);*//设置订单的ID 这个就是out\_trade\_no* seckillOrder.setCreateTime(**new** Date());*//创建时间* seckillOrder.setMoney(killgoods.getCostPrice());*//秒杀价格 价格* seckillOrder.setSeckillId(seckillId);*//秒杀商品的ID* seckillOrder.setSellerId(killgoods.getSellerId());  seckillOrder.setUserId(userId);*//设置用户ID* seckillOrder.setStatus(**"0"**);*//状态 未支付  //将构建的订单保存到redis中* **redisTemplate**.boundHashOps(**"seckillOrder"**).put(userId, seckillOrder); } |

### 4.4.3控制层

修改pinyougou-seckill-web的SeckillOrderController.java

|  |
| --- |
| @RequestMapping("/submitOrder")  **public** Result submitOrder(Long seckillId){  String userId = SecurityContextHolder.*getContext*().getAuthentication().getName();  **if**("anonymousUser".equals(userId)){//如果未登录  **return new** Result(**false**, "用户未登录");  }  **try** {  seckillOrderService.submitOrder(seckillId, userId);  **return new** Result(**true**, "提交成功");  }**catch** (RuntimeException e) {  e.printStackTrace();  **return new** Result(**false**, e.getMessage());  } **catch** (Exception e) {  e.printStackTrace();  **returnnew** Result(**false**, "提交失败");  }  } |

## 4.3前端代码

### 4.3.1前端服务层

pinyougou-seckill-web的seckillGoodsService.js

|  |
| --- |
| //提交订单  **this**.submitOrder=**function**(seckillId){  **return** $http.get('seckillOrder/submitOrder.do?seckillId='+seckillId);  } |

### 4.3.2前端控制层

pinyougou-seckill-web的seckillGoodsController.js

|  |
| --- |
| //提交订单  $scope.submitOrder=**function**(){  seckillGoodsService.submitOrder($scope.entity.id).success(  **function**(response){  **if**(response.success){  alert("下单成功，请在5分钟内完成支付");  location.href="pay.html";  }**else**{  alert(response.message);  }  }  );  } |

### 4.3.3页面

修改seckill-item.html

|  |
| --- |
| <ang-click=*"submitOrder()"*target=*"\_blank"*class=*"sui-btn btn-danger addshopcar"*>秒杀抢购</a> |

# 5.品优购-秒杀支付

## 5.1需求分析

用户成功下单后，跳转到支付页面。支付页显示微信支付二维码。用户完成支付后，保存订单到数据库。

## 5.2生成支付二维码

### 5.2.1后端代码

（1）pinyougou-seckill-web工程引入pinyougou-pay-interface依赖

（2）修改pinyougou-seckill-interface的SeckillOrderService.java

|  |
| --- |
| /\*\*  \* 根据用户名查询秒杀订单  \* **@param** userId  \*/  **public** TbSeckillOrder searchOrderFromRedisByUserId(String userId); |

（3）修改pinyougou-seckill-service的SeckillOrderServiceImpl.java

|  |
| --- |
| @Override  **public** TbSeckillOrder searchOrderFromRedisByUserId(String userId) {  **return** (TbSeckillOrder) redisTemplate.boundHashOps("seckillOrder").get(userId);  } |

1. 在pinyougou-seckill-web新建PayController.java

|  |
| --- |
| /\*\*  \* 支付控制层  \* **@author** Administrator  \*  \*/  @RestController  @RequestMapping("/pay")  **publicclass** PayController {    @Reference  **private** WeixinPayService weixinPayService;    @Reference  **private** SeckillOrderService seckillOrderService;  /\*\*  \* 生成二维码  \* **@return**  \*/  @RequestMapping("/createNative")  **public**Map createNative(){  //获取当前用户  String userId=SecurityContextHolder.*getContext*().getAuthentication().getName();  //到redis查询秒杀订单  TbSeckillOrder seckillOrder = seckillOrderService.searchOrderFromRedisByUserId(userId);  //判断秒杀订单存在  **if**(seckillOrder!=**null**){  **long** fen= (**long**)(seckillOrder.getMoney().doubleValue()\*100);//金额（分）  **return** weixinPayService.createNative(seckillOrder.getId()+"",+fen+"");  }**else**{  **return new** HashMap();  }  }  } |

### 5.2.2前端代码

将pinyougou-cart-web工程的payService.js payController.js pay.html qrious.min.js 拷贝到pinyougou-seckill-web工程 payController.js暂时注释对查询的调用。

## 5.3支付成功保存订单

当用户点击支付时，已经生成了秒杀订单，此时的秒杀订单状态为未支付，然后等支付完成之后，订单状态变成已经支付，需要获取微信支付系统返回的交易流水。将redis中的订单保存到数据库中。

### 5.3.1后端代码

1. 修改pinyougou-seckill-interface的SeckillOrderService.java，定义方法

|  |
| --- |
| /\*\*  \* 支付成功保存订单  \* **@param** userId  \* **@param**orderId  \*/  **Public void** saveOrderFromRedisToDb(String userId,Long orderId,String transactionId); |

（2）在pinyougou-seckill-service的SeckillOrderServiceImpl.java实现该方法

|  |
| --- |
| @Override  **public void** saveOrderFromRedisToDb(String userId, Long orderId, String transactionId) {  System.***out***.println("saveOrderFromRedisToDb:"+userId);  //根据用户ID查询日志  TbSeckillOrder seckillOrder = (TbSeckillOrder) redisTemplate.boundHashOps("seckillOrder").get(userId);  **if**(seckillOrder==**null**){  **thrownew** RuntimeException("订单不存在");  }  //如果与传递过来的订单号不符  **if**(seckillOrder.getId().longValue()!=orderId.longValue()){  **throw new** RuntimeException("订单不相符");  }  seckillOrder.setTransactionId(transactionId);//交易流水号  seckillOrder.setPayTime(**new** Date());//支付时间  seckillOrder.setStatus("1");//状态  seckillOrderMapper.insert(seckillOrder);//保存到数据库  redisTemplate.boundHashOps("seckillOrder").delete(userId);//从redis中清除  } |

1. 修改pinyougou-seckill-web的PayController.java，增加查询的方法

|  |
| --- |
| /\*\*  \* 查询支付状态  \* **@param** out\_trade\_no  \* **@return**  \*/  @RequestMapping("/queryPayStatus")  **public** Result queryPayStatus(String out\_trade\_no){  //获取当前用户  String userId=SecurityContextHolder.*getContext*().getAuthentication().getName();  Result result=**null**;  **int**x=0;  **while**(**true**){  //调用查询接口  Map<String,String>map = weixinPayService.queryPayStatus(out\_trade\_no);  **if**(map==**null**){//出错  result=**new** Result(**false**, "支付出错");  **break**;  }  **if**(map.get("trade\_state").equals("SUCCESS")){//如果成功  result=**new** Result(**true**, "支付成功");  seckillOrderService.saveOrderFromRedisToDb(userId, Long.*valueOf*(out\_trade\_no), map.get("transaction\_id"));  **break**;  }  **try** {  Thread.*sleep*(3000);//间隔三秒  } **catch** (InterruptedException e) {  e.printStackTrace();  }  x++;//设置超时时间为5分钟  **if**(x>100){  result=**new** Result(**false**, "二维码超时");  **break**;  }  }  **return**result;  } |

### 5.3.2前端代码

调用查询的方法，参见pinyougou-cart-web工程。

|  |
| --- |
| queryPayStatus(response.out\_trade\_no);//查询支付状态 |

## 5.4订单超时处理

当用户下单后5分钟尚未付款应该释放订单，增加库存。

关闭微信的支付订单，并删除nosql数据库中的订单，并且将库存增加。

### 5.4.1删除缓存中的订单

（1）修改pinyougou-seckill-interface的SeckillOrderService.java

|  |
| --- |
| /\*\*  \* 从缓存中删除订单  \* **@param** userId  \* **@param** orderId  \*/  **public void** deleteOrderFromRedis(String userId,Long orderId); |

（2）修改pinyougou-seckill-service的SeckillOrderServiceImpl.java

|  |
| --- |
| @Override **public void** deleteOrderFromRedis(String userId, Long orderId) {  *//从缓存中获取订单数据* TbSeckillOrder seckillOrder = (TbSeckillOrder) **redisTemplate**.boundHashOps(**"seckillOrder"**).get(userId);  *//判断如果缓存中的订单ID 和传递过来的订单ID 是一致的，那么就删除* **if**(seckillOrder!=**null**&&  seckillOrder.getId().longValue()== orderId.longValue() ){  **redisTemplate**.boundHashOps(**"seckillOrder"**).delete(userId);*//删除缓存中的订单  //恢复库存 删除 订单之后需要恢复缓存  //1.从缓存中提取秒杀商品  //根据缓存中的订单对象获取订单所对应的秒杀商品。* TbSeckillGoods seckillGoods=(TbSeckillGoods)**redisTemplate**.boundHashOps(**"seckillGoods"**).get(seckillOrder.getSeckillId());  **if**(seckillGoods!=**null**){*//获取到了商品数据 如果为空 说明此时商品已经被秒光了，现在需要在构建一个。* seckillGoods.setStockCount(seckillGoods.getStockCount()+1);*//库存增加* **redisTemplate**.boundHashOps(**"seckillGoods"**).put(seckillOrder.getSeckillId(), seckillGoods);*//存入缓存* }**else**{  *//从数据库查询 并更新恢复剩余库存到1  //将数据库中的商品 保存到redis中* }  }  } |

### 5.4.2关闭微信订单

（1）修改pinyougou-pay-interface 的WeixinPayService接口

|  |
| --- |
| /\*\*  \* 关闭支付  \* **@param** out\_trade\_no  \* **@return**  \*/  **public**Map closePay(String out\_trade\_no); |

1. 修改pinyougou-pay-service的WeixinPayServiceImpl

|  |
| --- |
| @Override  **public** Map closePay(String out\_trade\_no) {  Mapparam=**new**HashMap();  param.put("appid", appid);//公众账号ID  param.put("mch\_id", partner);//商户号  param.put("out\_trade\_no", out\_trade\_no);//订单号  param.put("nonce\_str", WXPayUtil.*generateNonceStr*());//随机字符串  String url="https://api.mch.weixin.qq.com/pay/closeorder";  **try** {  String xmlParam = WXPayUtil.*generateSignedXml*(param, partnerkey);  HttpClient client=**new** HttpClient(url);  client.setHttps(**true**);  client.setXmlParam(xmlParam);  client.post();  String result = client.getContent();  Map<String, String> map = WXPayUtil.*xmlToMap*(result);  System.***out***.println(map);  **return** map;  } **catch** (Exception e) {  e.printStackTrace();  **returnnull**;  }  } |

### 5.4.3超时调用服务

修改pinyougou-seckill-web的PayController.java

|  |
| --- |
| /\*\*  \* 查询支付状态  \* **@param** out\_trade\_no  \* **@return**  \*/  @RequestMapping("/queryPayStatus")  **public** Result queryPayStatus(String out\_trade\_no){  //获取当前用户  String userId=SecurityContextHolder.*getContext*().getAuthentication().getName();  Result result=**null**;  **int** x=0;  **while**(**true**){  ........  **try** {  Thread.*sleep*(3000);//间隔三秒  } **catch** (InterruptedException e) {  e.printStackTrace();  }  //不让循环无休止地运行定义变量，如果超过了这个值则退出循环，设置时间为1分钟  x++;  **if**(x>20){  result=**new** Result(**false**, "二维码超时");  //1.调用微信的关闭订单接口  Map<String,String>payresult = weixinPayService.closePay(out\_trade\_no);  **if**( "SUCCESS".equals(payresult.get("result\_code")) ){//如果返回结果是正常关闭  **if**("ORDERPAID".equals(payresult.get("err\_code"))){  result=**new** Result(**true**, "支付成功");  seckillOrderService.saveOrderFromRedisToDb(userId, Long.*valueOf*(out\_trade\_no), map.get("transaction\_id"));  }  }  **if**(result.isSuccess()==**false**){  System.***out***.println("超时，取消订单");  //2.调用删除  seckillOrderService.deleteOrderFromRedis(userId, Long.*valueOf*(out\_trade\_no));  }  **break**;  }  }  **return** result;  } |

### 5.4.3前端代码

pinyougou-seckill-web的payController.js

|  |
| --- |
| //查询支付状态  queryPayStatus=**function**(out\_trade\_no){  payService.queryPayStatus(out\_trade\_no).success(  **function**(response){  **if**(response.success){  location.href="paysuccess.html#?money="+$scope.money;  }**else**{  **if**(response.message=='二维码超时'){  location.href="payTimeOut.html";  }**else**{  location.href="payfail.html";  }  }  }  );  } |