

# 秒杀模块总结

## 1. 秒杀业务分析

### 1.1 需求分析


- (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情页。
- (4) 商品详情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1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

两张表：

秒杀商品表： `tb_seckill_goods`

#### Columns (16)

	Field	Type	Comment
	id	bigint(20) NOT NULL	
	goods_id	bigint(20) NULL	spu ID
	item_id	bigint(20) NULL	sku ID
	title	varchar(100) NULL	标题
	small_pic	varchar(150) NULL	商品图片
	price	decimal(10,2) NULL	原价格
	cost_price	decimal(10,2) NULL	秒杀价格
	seller_id	varchar(100) NULL	商家ID
	create_time	datetime NULL	添加日期
	check_time	datetime NULL	审核日期
	status	varchar(1) NULL	审核状态
	start_time	datetime NULL	开始时间
	end_time	datetime NULL	结束时间
	num	int(11) NULL	秒杀商品数
	stock_count	int(11) NULL	剩余库存数
	introduction	varchar(2000) NULL	描述

`id` 作为一个标识字段，页面间跳转时（初始化单个商品的详情页）作为key放到redis中  
`cost_price` 秒杀价格，作为某个用户的订单的总价（每个用户只能秒杀一件商品）

秒杀订单表： `tb_seckill_order`

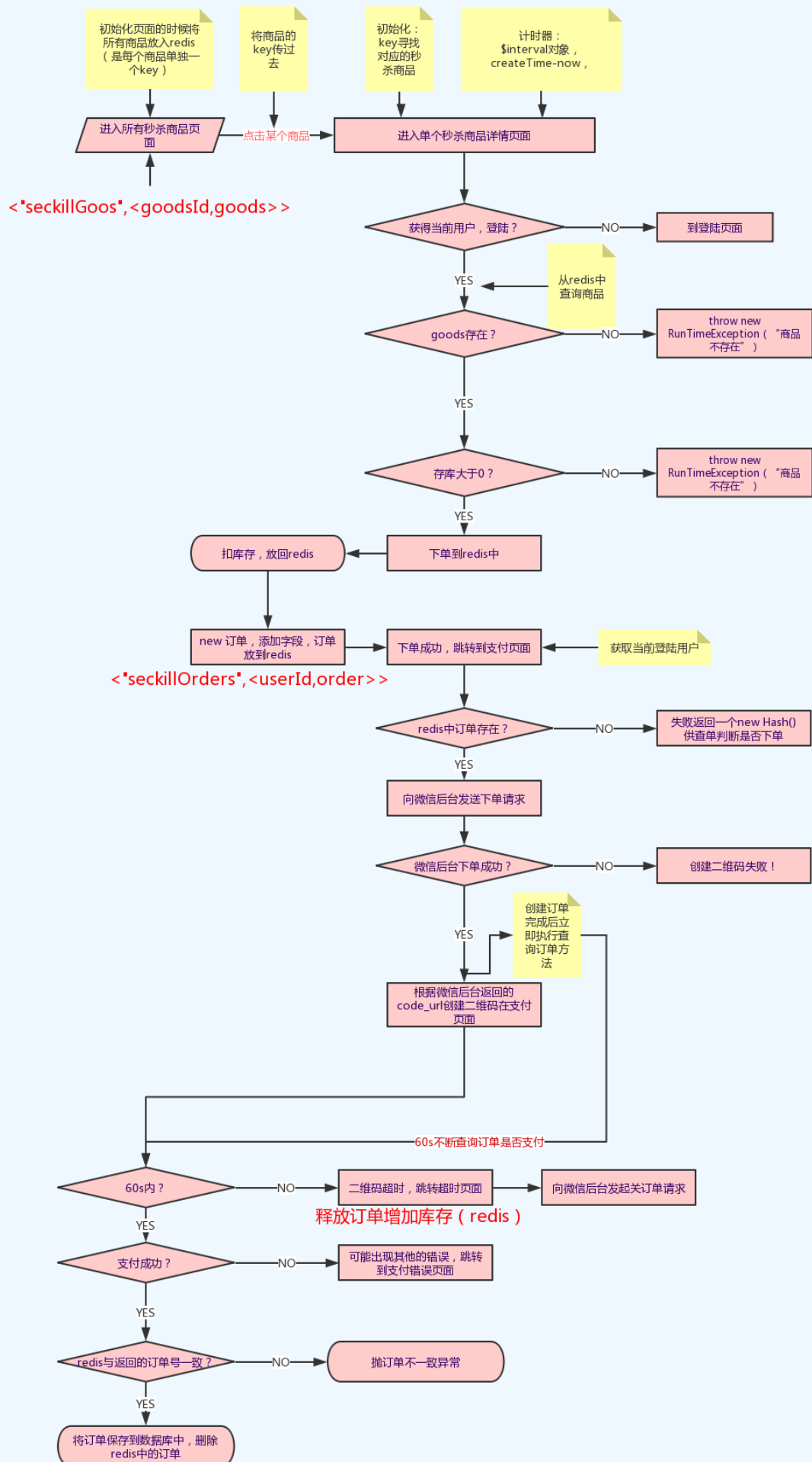
## Columns (12)

	Field	Type	Comment
🔑	id	bigint(20) NOT NULL	主键
	seckill_id	bigint(20) NULL	秒杀商品ID
	money	decimal(10,2) NULL	支付金额
	user_id	varchar(50) NULL	用户
	seller_id	varchar(50) NULL	商家
	create_time	datetime NULL	创建时间
	pay_time	datetime NULL	支付时间
	status	varchar(1) NULL	状态
	receiver_address	varchar(200) NULL	收货人地址
	receiver_mobile	varchar(20) NULL	收货人电话
	receiver	varchar(20) NULL	收货人
	transaction_id	varchar(30) NULL	交易流水

### 1.3 秒杀实现的思路

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**。

**流程图：**



## 2. 秒杀首页

### 2.1 需求分析

点击秒杀频道后进入首页，加载所有的秒杀商品

后台就是满足条件的商品：

- 开始时间大于等于当前时间
- 结束时间小于当前时间
- 通过审核的
- 剩余库存大于0的

### 2.2 前端

#### 2.2.1 页面

```
<!-- AngularJS 相关-->
<script type="text/javascript" src="plugins/angularjs/angular.min.js"></script>
<!--模块 控制器 服务相关-->
<script type="text/javascript" src="js/base.js"></script>
<script type="text/javascript" src="js/service/seckillService.js"></script>
<script type="text/javascript" src="js/controller/baseController.js"></script>
<script type="text/javascript" src="js/controller/seckillController.js"></script>
</head>
<!--初始化是调用查找所有的满足条件的秒杀商品-->
<body ng-app="pinyougou" ng-controller="seckillController" ng-
init="findSeckillGoods()">
```

#### 2.2.2 控制器

```
//查找所有满足条件的秒杀商品
$scope.findSeckillGoods={()=>{
    seckillService.findSeckillGoods().success(response=>{
        $scope.seckillList=response;
    })
}}
```

#### 2.2.3 服务

```
//查找满足条件的所有秒杀商品
this.findSeckillGoods={()=>{
    return $http.get("./seckillGoods/findSeckillGoods.do");
}}
```

## 2.3 后台

### 2.3.1 controller

```
//查找满足条件的所有秒杀商品
@RequestMapping("findSeckillGoods")
public List< TbSeckillGoods > findSeckillGoods(){
    return seckillGoodsService.findSeckillGoods();
}
```

### 2.3.2 service接口

```

/**
 * 查找所有满足条件的秒杀商品
 * @return
 */
List<TbSeckillGoods> findSeckillGoods();

```

### 2.3.3 service

```

/**
 * 查找所有满足条件的秒杀商品
 * @return
 */
@Override
public List<TbSeckillGoods> findSeckillGoods() {

    TbSeckillGoodsExample example = new TbSeckillGoodsExample();
    Criteria criteria = example.createCriteria();
    //条件一 :通过审核（即状态码 为 1）
    criteria.andStatusEqualTo("1");
    //条件二 :库存大于0
    criteria.andNumGreaterThan(0);
    //条件三 :开始时间小于等于当前时间
    criteria.andStartTimeLessThanOrEqualTo(new Date());
    //条件四 :结束时间大于当前时间
    criteria.andEndTimeGreaterThan(new Date());
    //获得秒杀商品列表
    List<TbSeckillGoods> seckillGoodsList =
seckillGoodsMapper.selectByExample(example);
    //将列表中的每件商品分别存入redis中（因为每一件商品都是热点数据）
    for (TbSeckillGoods goods : seckillGoodsList) {
        redisTemplate.boundHashOps("seckillGoodsList").put(goods.getId(), goods);
    }
    //返回查询到的符合条件的秒杀商品列表
    return seckillGoodsList;
}

```

## 3. 秒杀商品详情

### 3.1 需求分析

秒杀频道首页，从Redis中取出 并且显示正在秒杀的商品（已经开始，未结束的商品）

### 3.2 前端

#### 3.2.1 页面

秒杀首页点击某件商品传来的参数 id 是对应商品的id

```

//点击立即抢购跳转到详情页
$scope.buyNow=(id)=>{
    location.href="seckill-item.html?id="+id;
}

```

```

<!-- AngularJS 相关-->
<script type="text/javascript" src="plugins/angularjs/angular.min.js"></script>
<!--模块 控制器 服务相关-->
<script type="text/javascript" src="js/base.js"></script>
<script type="text/javascript" src="js/service/seckillService.js"></script>
<script type="text/javascript" src="js/controller/baseController.js"></script>
<script type="text/javascript" src="js/controller/seckillController.js"></script>

</head>
<!--从首页点击立即抢购的时候传来的参数的时候-->
<body ng-app="pinyougou" ng-controller="seckillController" ng-init="findOne()">

```

### 3.2.2 控制器

注意id 是从首页对应的商品获取的

```

//详情页的初始化显示商品
$scope.findOne={()=>{
    let id=$location.search()["id"];
    seckillService.findOne(id).success(response=>{
        $scope.goods=response;
        //显示时间倒计时
        showTime();
    });
}}

```

计时器显示倒计时：

```

//倒计时函数
showTime={()=>{
    //获取结束时间与当前时间差的秒数（剩余时间的秒数）
    let time = Math.floor((new Date($scope.goods.endTime).getTime()
        - new Date().getTime()) / 1000);
    let f=$interval(()=>{
        //定义用来显示倒计时的字符串
        $scope.timeStr="";
        //获取剩余的天数
        let days =Math.floor(time/(24*60*60));
        //获取剩余小时数
        let hours=Math.floor((time-days*24*60*60)/3600);
        //获取剩余的分钟数
        let mins=Math.floor((time-days*24*3600-hours*3600)/60);
        //获取剩余的秒数
        let seconds=time-days*24*3600-hours*3600-mins*60;
        //如果有天数则显示 “天”
        if (days>0){
            $scope.timeStr+=days+"天";
        }
        $scope.timeStr += hours+": "+mins+": "+seconds;
        time--;
        if (time<= 0){
            $interval.cancel(f);
        }
    },1000)
}

```

```
}
```

### 3.2.2 服务

```
//跳转到详情页面的时候根据秒杀首页传来的id 查询这个id 的商品的详情
this.findOne=(id)=>{
    return $http.get("./seckillGoods/findOne.do?id="+id);
}
```

## 3.2 后台

### 3.2.1 controller

```
/**
 * 获取实体
 * @param id
 * @return
 */
@RequestMapping("/findOne")
public TbSeckillGoods findOne(Long id){
    return seckillGoodsService.findOne(id);
}
```

### 3.2.2 service接口

```
/**
 * 根据ID获取实体
 * @param id
 * @return
 */
public TbSeckillGoods findOne(Long id);
```

### 3.2.3 service

**注意** 取出的时候是从redis中取 （访问量高）

```

/**
 * 根据ID获取实体
 * @param id
 * @return
 */
@Override
public TbSeckillGoods findOne(Long id){
    //根据商品id从redis中取出商品详情
    TbSeckillGoods seckillGoods= (TbSeckillGoods)
    redisTemplate.boundHashOps("seckillGoodsList").get(id);
    return seckillGoods;
}

```

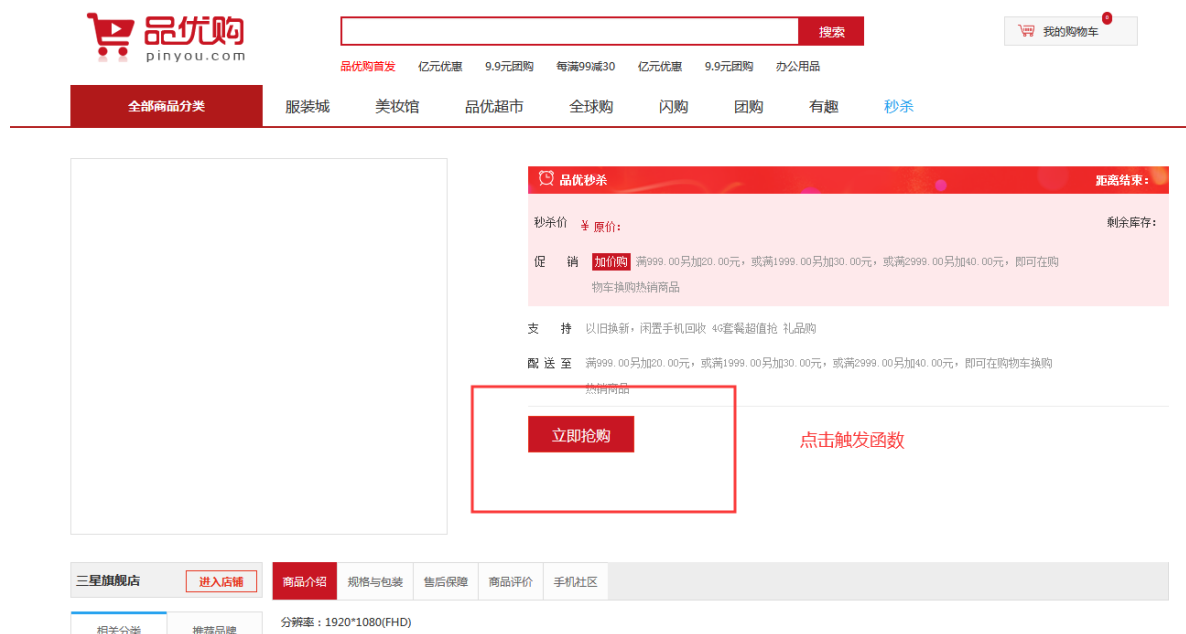
## 4. 秒杀下单

### 4.1 需求

商品详情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。

### 4.2 前端

#### 4.2.1 页面



```

<a href="javascript:void (0)" ng-click="submitOrder()" class="sui-btn btn-danger addshopcar">立即抢购</a>

```

#### 4.2.2 控制器

```

//秒杀商品的详情页面点击立即抢购 去后台下单
$scope.submitOrder={()=>{
    //传入秒杀商品的id
    seckillService.submitOrder($scope.goods.id).success(response=>{
        if (response.success) {
            //如果下单成功 ,跳转支付页面

```



```

        location.href="pay.html"
    }else {
        if (response.messsage == "请先登录") {
            //跳转到登陆页面
        }else {
            //下单失败，商品已经被抢光
            alert(response.message);
        }
    }
}
})
}

```

#### 4.2.3 服务

```

//点击立即抢购 发起下单
this.submitOrder=(seckillId)=>{
    return $http.get("./seckillOrder/submitOrder.do?id="+seckillId);
}

```

### 4.3 后台

#### 4.3.1 控制器

判断下单之前判断用户是否登陆 登陆了才能下单

```

/**
 * 点击秒杀抢购 发起提交订单请求
 * @param seckillId
 * @return
 */
@RequestMapping("submitOrder")
public Result submitOrder(Long seckillId){
    //1.得到当前登陆的用户id
    String userId =
SecurityContextHolder.getContext().getAuthentication().getName();
    //2.判断是否登陆
    if (userId.equals("anonymousUser")){
        //2.1 用户没有登陆（是匿名用户）
        return new Result(false,"请先登陆");
    }else {
        try {
            //2.2 用户登陆了 将根据用户的userId 与商品的id
            seckillOrderService.submitOrder(userId,seckillId);
            return new Result(true,"下单成功，请在五分钟支付");
        } catch (Exception e) {
            e.printStackTrace();
            return new Result(false,"下单失败！");
        }
    }
}
}

```

#### 4.3.2 service接口

```
/**
 * 根据登陆用户的id 以及秒杀商品的id 进行秒杀下单
 * @param userId
 * @param id
 */
void submitOrder(String userId, Long id);
```

#### 4.3.3 service

```
/**
 * 秒杀商品的抢购
 * @param userId
 * @param id
 */
@Override
public void submitOrder(String userId, Long id) {
    // 1.判断商品存不存在
    //1.1 从redis中查询商品
    TbSeckillGoods seckillGoods = (TbSeckillGoods)
redisTemplate.boundHashOps("seckillGoodsList").get(id);
    //1.2 判断该商品是否存在以及库存是否大于0
    if (seckillGoods != null && seckillGoods.getStockCount() > 0){
        //如果存在且库存大于0 则可以进行秒杀
        //扣库存 并且放回缓存
        seckillGoods.setStockCount(seckillGoods.getStockCount()-1);
        redisTemplate.boundHashOps("seckillGoodsList").put(id,seckillGoods);
        //判断库存是否为0
        if (seckillGoods.getStockCount()==0){
            //为零的话 同步到数据库 同时将秒杀商品移除redis

            redisTemplate.boundHashOps("seckillGoodsList").delete(id);
        }

        //3. 进行下单
        TbSeckillOrder seckillOrder = new TbSeckillOrder();
        //3.1 生成秒杀订单id
        long orderId = idWorker.nextId();
        //3.2 添加订单的各项字段（根据秒杀商品）
        //秒杀订单的创建时间 （也就是当前时刻）
        seckillOrder.setCreateTime(new Date());
        //秒杀订单的id 通过idWorker生成
        seckillOrder.setId(orderId);
        //秒杀订单的价格 也就是秒杀商品的单价
        seckillOrder.setMoney(seckillGoods.getCostPrice());
        //秒杀商品的id
        seckillOrder.setSeckillId(id);
        //参与秒杀用户的id
        seckillOrder.setUserId(userId);
        //秒杀订单的支付状态 0 表示未支付
        seckillOrder.setStatus("0");
    }
}
```

```

//商家的id
seckillOrder.setSellerId(seckillGoods.getSellerId());

//4. 将订单放到redis中 （因为访问的量很大）
redisTemplate.boundHashOps("seckillOrder").put(userId, seckillOrder);
}else {
    throw new RuntimeException("商品已不存在");
}
}

```

## 5. 秒杀支付

### 5.1 需求

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 前端

#### 5.2.1 页面

```

<script type="text/javascript" src="plugins/angularjs/angular.min.js"></script>
<script type="text/javascript" src="js/base.js"></script>
<script type="text/javascript" src="js/service/payService.js"></script>
<script type="text/javascript" src="js/controller/baseController.js"></script>
<script type="text/javascript" src="js/controller/payController.js"></script>
<script type="text/javascript" src="plugins/qrious.min.js"></script>
</head>
<!--初始化页面你的时候调用向微信后台请求下单-->
<body ng-app="pinyougou" ng-controller="payController" ng-init="createNative()">

```

#### 5.2.2 控制器

```

//向微信后台请求收款地址
$scope.createNative=>(){
    payService.createNative().success(response=>{
        //返回的总金额显示在页面上
        $scope.money=response.allMoney;
        //返回的订单号用来显示在页面你上
        $scope.out_trade_no=response.out_trade_no;
        //返回的code_url用户生成二维码
        var qr=new QRious({
            element:document.getElementById("erweima"),
            size:500,
            level:"H",
            value:response.code_url
        });
        //调用检查支付状态的方法
        queryPayStatus();
    });
};

```

### 5.2.3 服务

```

//向微信后台发起支付请求
this.createNative=>(){
    return $http.get("./pay/createNative.do");
}

```

## 5.3 后台

### 5.3.1 controller

```

/**
 * 向微信后台发起支付请求
 * @return
 */
@RequestMapping("createNative")
public Map createNative(){
    //获取当前用户
    String userId =
SecurityContextHolder.getContext().getAuthentication().getName();
    //到redis中查询该用户是否下单
    TbSeckillOrder seckillOrder =
seckillOrderService.searchSeckillOrderFromRedis(userId);
    if (seckillOrder != null){
        //不为空则说明该用户下了秒杀订单 则向微信后台发起下单请求
        //传入 用户用户支付的订单 以及总金额(秒杀商品的一件)
        return payService.createNative(seckillOrder.getId() + "",
seckillOrder.getMoney().doubleValue() * 100 + "");
    }
    return new HashMap();
}

```

### 5.3.2 service 接口

```
/**
 * 向微信后台发起支付请求
 * @param out_trade_no
 * @param allMoney
 * @return
 */
Map createNative(String out_trade_no,String allMoney);
```

### 5.3.3 service

```
/**
 * 用户向微信后台发送支付请求
 *
 * @param out_trade_no 交易的订单号码
 * @param allMoney 交易的总金额
 * @return 返回一个xml类型的字符串 并且取其中的code_url作为生成二维码的依据
 */
@Override
public Map createNative(String out_trade_no, String allMoney) {
    //1.设置请求参数 借用工具类将hashMap 转化为xml 然后发送请求给微信后台
    Map paramsMap = new HashMap();
    paramsMap.put("appid", appid); //公众号id
    paramsMap.put("mch_id", partner); //商户号 微信分配
    paramsMap.put("nonce_str", WXPUtil.generateNonceStr()); //随机字符串 微信支付
    //工具生成
    paramsMap.put("body", "品优购商品"); //商品的描述可以根据需要传
    paramsMap.put("out_trade_no", out_trade_no); //商家的订单号 上级自己生成
    //订单的支付金额 订单总金额, 单位为分 由前端进行转化之后
    paramsMap.put("total_fee", allMoney);
    paramsMap.put("spbill_create_ip", "127.0.0.1"); //终端ip 发送请求的设备的ip地址
    paramsMap.put("notify_url", notifyurl); //通知地址 用于发送请求后回调的地址
    paramsMap.put("trade_type", "NATIVE"); //交易类型 由微信提供可选的
    JSAPI-JSAPI支付 NATIVE-Native支付 APP -APP支付
    //将上面的paramsMap转换为 xml字符串
    try {
        //签名设置 注意
        String xmlParam = WXPUtil.generateSignature(paramsMap, partnerkey);
        // 2.发送请求
        //2.1 构造方法设置发送的请地址
        HttpClient httpClient = new HttpClient(sendOrderUrl);
        //2.2 设置是否是加密的http协议
        httpClient.setHttps(true);
        //2.3 设置请求参数 格式为xml字符串
        httpClient.setXmlParam(xmlParam);
        //2.4 发送请求 post提交
        httpClient.post();

        //3.获得返回值
        //3.1 接收微信后台放回的结果
        String content = httpClient.getContent();
        //3.2 将微信后台返回的结果转换为map
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```

        Map<String, String> xmlMap = WXPAYUtil.xmlToMap(content);
        //3.3 构造接收放回结果的map
        Map resultMap = new HashMap();
        //3.4 接收必要的参数
        resultMap.put("out_trade_no", out_trade_no);
        resultMap.put("allMoney", allMoney);
        //最重要的是接收微信后台返回的 用于生成二维码的支付地址
        resultMap.put("code_url", xmlMap.get("code_url"));
        return resultMap;
    } catch (Exception e) {
        e.printStackTrace();
    }
    //如果发生异常 则返回一个空的map
    return new HashMap();
}

```

## 查询支付状态

```

/**
 * 向微信后台发起查询请求
 * @param out_trade_no
 * @return
 */
@Override
public Map queryPayStatus(String out_trade_no) {
    //1.设置请求参数
    //1.1 构建一个map用来传递xml串
    Map paramsMap = new HashMap();
    paramsMap.put("appid", appid); //公众号id
    paramsMap.put("mch_id", partner); //商户号 微信分配
    paramsMap.put("nonce_str", WXPAYUtil.generateNonceStr()); //随机字符串
    paramsMap.put("out_trade_no", out_trade_no); //商户订单id （参照开发文档可以与
    微信订单号二选一）
    try {
        //1.2 签名
        String xmlParams = WXPAYUtil.generateSignedXml(paramsMap, partnerkey);

        //2.发送请求(到微信后台的查询订单的url)
        //2.1 构建一个发送请求的对象
        HttpClient httpClient = new HttpClient(queryOrderUrl);
        //2.2 设置是否http加密
        httpClient.setHttps(true);
        //2.3 为请求对象设置请求参数
        httpClient.setXmlParam(xmlParams);
        //2.4 发送请求
        httpClient.post();

        //3.获得返回值
        String content = httpClient.getContent();
        //3.2 将返回值转化为map
        Map<String, String> resultMap = WXPAYUtil.xmlToMap(content);
        return resultMap;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

```

## 二维码超时处理

释放订单，增加库存

## 微信后台取消订单

```
/**
 * 向微信后台发送关闭订单请求
 * @param trade_out_no 用户支付订单流水号
 * @return
 */
@Override
public Map closeSeckillOrder(String trade_out_no) {
    //1. 设置请求参数
    Map param = new HashMap();
    param.put("appid", appid); //公众账号ID
    param.put("mch_id", partner); //商户号
    param.put("out_trade_no", trade_out_no); //订单号
    param.put("nonce_str", WXPUtil.generateNonceStr()); //随机字符串
    try {
        //2. 发送请求
        String xmlPara = WXPUtil.generateSignedXml(param, partnerkey);
        HttpClient httpClient=new HttpClient(colseOrderUrl);
        httpClient.setHttps(true);
        httpClient.setXmlParam(xmlPara);
        httpClient.post();

        //3. 获得返回值
        String resultPara = httpClient.getContent();
        Map<String, String> resultMap = WXPUtil.xmlToMap(resultPara);
        return resultMap;
    } catch (Exception e) {
        e.printStackTrace();
        return new HashMap();
    }
}
```