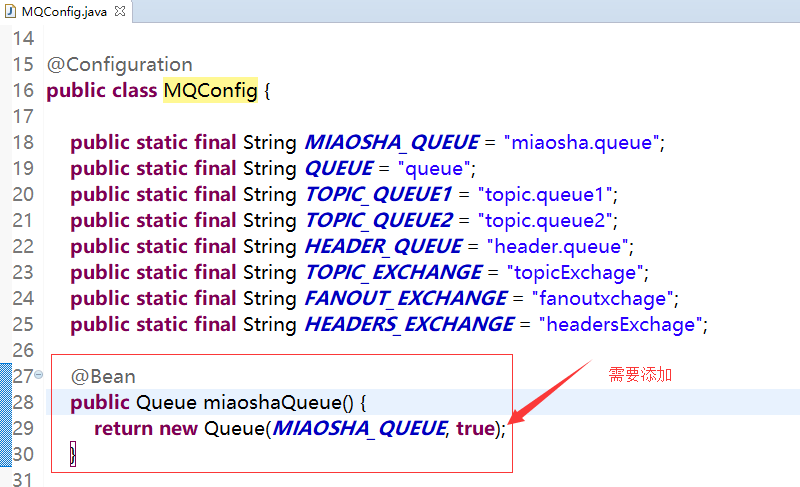
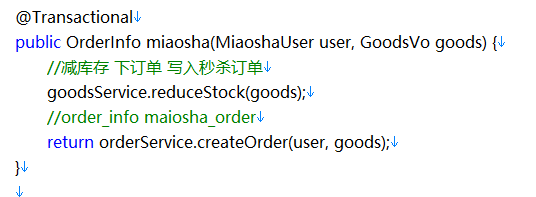
# Part1 课程勘误

## 第六章秒杀接口优化



否则，运行的时候会报错：找不到miaosha.queue。为什么视频能正常运行呢？因为在备课的时候，已经把Queue创建出来了，并且Queue是持久化的，因此，录课的时候能正常运行。

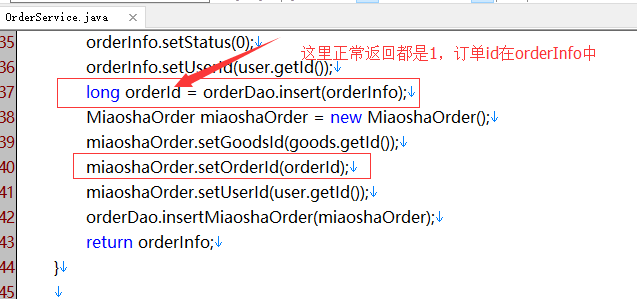
## 第四章MiaoshaService秒杀功能



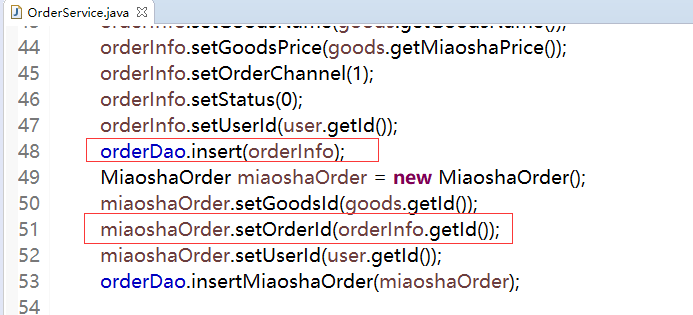
这里不严谨，第六章中会做完善：



## 第四章OrderService创建秒杀订单



第六章中会做修改：



## 所有的秒杀相关的接口

比如：秒杀、获取秒杀地址、获取秒杀结果、获取秒杀验证码都需要加上秒杀活动是否开始结束的判断。我们在课程一开始第一章中的也讲了，前台校验只是为了防止用户误操作，后台校验才可以防止恶意用户。课程只是为了方便讲解代码，省略了很多要校验的东西，实际项目开发中，这些一个也不能省略。

# Part2 课程常见问题

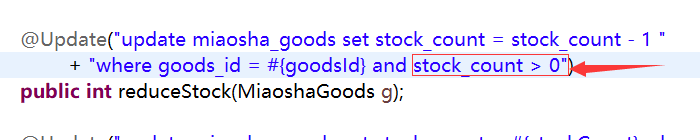
## 如何保证不卖超

有两种情况可能会导致卖超：（1）一个用户同时发出了多个请求，如果库存足够，没加限制，用户就可以下多个订单。（2）减库存的sql上没有加库存数量的判断，并发的时候也会导致把库存减成负数。

我们的解决办法：

对于（1）：前端加验证码，防止用户同时发出多个请求，在后端的miaosha\_order表中，对user\_id和goods\_id加唯一索引，确保一个用户对一个商品绝对不会生成两个订单。

对于（2）：我们的减库存的sql上应该加上库存数量的判断：



数据库更新记录的时候会加锁，实际上是串行的执行update的，因此绝对不会卖超！

## Redis中的库存如何与DB中的库存保持一致？

Redis中的数量不是库存，它的作用仅仅时候只是为了阻挡多余的请求透传到db，起到一个保护DB的作用。因为秒杀商品的数量是有限的，比如只有10个，让1万个请求去访问DB是没有意义的，因为最多只有10个请求会下单成功，剩余的9990个请求都是无效的，是可以不用去访问db而直接失败的。

因此，这是一个伪问题，我们是不需要保持一致的。

## Redis预减成功，DB扣减库存失败怎么办？

两大类情况可导致redis预减成功而DB扣减失败：

1. 如果一个用户发出了多个请求（不管何种手段），而这些所有的请求比所有其他用的请求都更快的到达了服务器，这个时候如果库存足够，就会出现redis预减多次，而只能下单成功一次（前提是：这个用户的多个请求比网站的其他用户的请求都更快的到达服务器，这在网络环境不可知的情况下，基本不可能）
2. 还有就是在生成订单的过程中发生了不可预料的异常，也会导致redis扣减成功，而db扣减失败（如果是DB出现了异常，可能所有的订单都无法生成，但是只要存在redis预减，活动就可以正常结束）

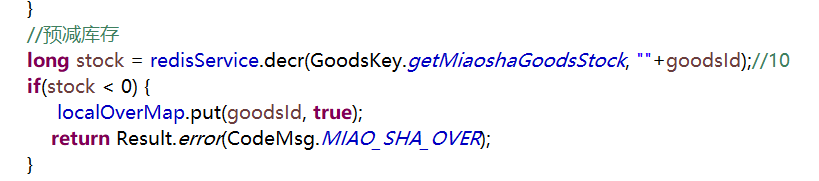
因此，在初始化的时候，redis中的数量可以多于db的库存数量。

出现这种情况的后果是什么？

1. 对用户而言，秒杀不中是正常现象，秒杀中才是意外，单个用户能否秒杀中本来就是小概率事件，出现这种情况对用户而言是没有任何影响的。
2. 对商户而言，本来就是为了做活动拉流量拉人气的，卖不完还可以省一部分费用，但是活动还是正常参与了，也是没有任何影响
3. 对网站而言，网站最重要的是用户体验，只要网站不崩，用户不骂娘，对网站也没有任何影响。

所以，卖不完是完全允许的，但是卖超是绝对不允许的！卖超的这部分钱商家是不会出的，需要网站自己来出。

## 为什么Redis中的数量会减成负数？



假如redis中的数量是1，这个时候同时过来100个请求，大家一起去执行decr,数量就会减成-99，这是正常的。

## 为什么要单独维护一个秒杀结束的标志？

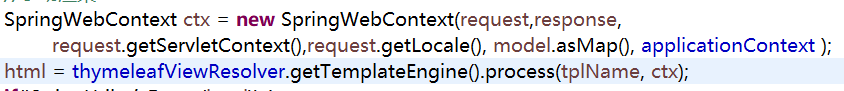
1. 前面也提过，所有的秒杀相关的接口都要加上活动是否结束的标志，如果结束就直接返回了，包括轮询的接口，防止一直轮询没法结束。
2. 管理后台也可以手动的更改这个标志，防止出现活动开始以后就没法结束这种意外的发生。

## 为什么要做隐藏秒杀接口地址？

1. html是可以被右键->查看源代码，如果秒杀地址写死在源文件中，是很容易就被恶意用户拿到的，就可以被机器人利用来刷接口。
2. 通过一个接口来返回秒杀地址的好处是，可以在活动临近开始的时候，服务端可以把地址换掉，这样就算恶意用户提前拿到了地址，但是拿到的也是一个不可用的地址。
3. 服务端可以通过管理后台来随时修改接口的地址。

## SpringBoot升级2.0以后，thymleaf页面缓存如何来做？

升级之前：



升级之后：



## GET跟POST的区别是什么？

1. 传参方式，GET放在url后面，post放在http的body，GET更不安全
2. 参数长度，浏览器对url后面的参数长度有限制，post也有限制，但是post要比get大得多。这是浏览器加的限制，跟Http协议无关
3. GET的页面可以被缓存，POST的不可以
4. GET可以添加收藏，POST不可以
5. GET可以后退刷新，POST刷新会重新提交数据。
6. GET不能做文件上传，POST可以。
7. 以上都是表象，最根本的区别是语义上的区别：GET的语义是请求获取指定的资源。GET方法是安全、幂等、可缓存的（除非有 Cache-ControlHeader的约束）。POST的语义是根据请求报文对指定的资源做出处理，具体的处理方式视资源类型而不同。POST不安全，不幂等，（大部分实现）不可缓存。简单地说GET是获取数据，POST是修改数据。跟Restful还有点区别，Restful规范里面，GET是获取，POST是添加，PUT是修改，DELETE是删除。

## 虚拟机里面的Redis连接不上怎么办？

可能的原因：

1. redis默认是不允许远程连接的，需要修改配置：bind 0.0.0.0
2. CentOS6虚拟机开启了防火墙，6379端口连接不上，需要放行6379端口：

/sbin/iptables -I INPUT -p tcp --dport 6379 -j ACCEPT

/etc/rc.d/init.d/iptables save

service iptables restart

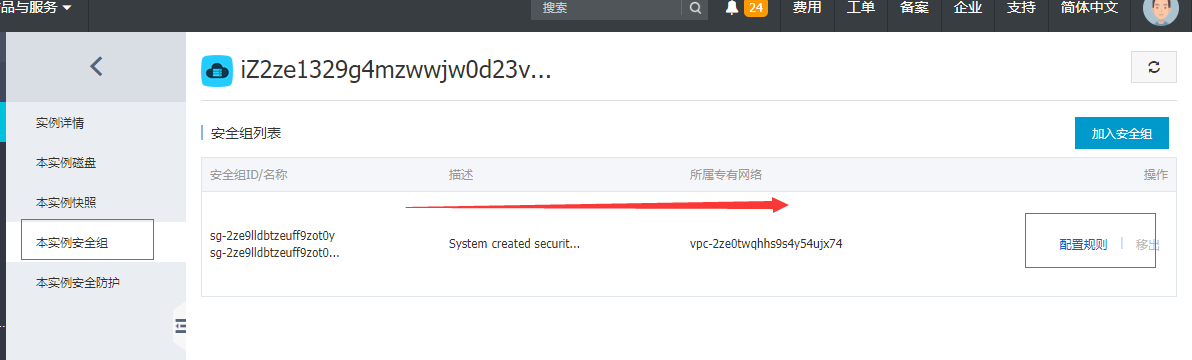
或者直接：iptable –F 关闭防护墙

对于防火墙不是使用iptables的系统，请使用系统对应的命令。

1. redis服务器启动的时候，需要手动指定下配置文件：

./redis-server /etc/redis/6379.conf

1. 如果是阿里云的主机，需要额外：





1. 检查application.properties配置文件，server.host的值后面是否有空格
2. 检查redis.conf配置文件，是否配置了多个bind指令，后面的会把前面的覆盖掉

## Redis挂掉怎么办？

线上的Redis服务器一般都会做replicate，也就是所谓的主从，使用哨兵机制实现监控和自动故障转移，当主机挂掉以后，从机自动升级成主机。

实际上不光是Redis，为了高可用，线上服务器的每一个节点都必须要防止单点故障，比如：nginx、mysql、rabbitmq等等。

## RabbitMQ里面的消息如何才能不丢失？即使是服务器重启？

RabbitMQ不丢失消息要做到：

1. Exchage持久化
2. Queue持久化
3. 发送消息的时候，设置MessageDeliveryMode为MessageDeliveryMode.PERSISTENT，这个也是默认的行为
4. 消息手动确认

## RabbitMQ如何开启控制台？

./sbin/rabbitmq-plugins enable rabbitmq\_management

重启rabbitmq

打开浏览器输入： <http://127.0.0.1:15672/>，用户名guest，密码guest

## 分布式Session，如何Cookie被禁用怎么办？

Cookie被禁用可以在url中传递参数：



## RabbitMQ安装报错：{missing\_dependencies,[crypto,ssl]}

（1）重新安装下openssl：

tar -zvxf openssl-1.0.1s.tar.gz

cd openssl-1.0.1s

./config --prefix=/usr/local/openssl

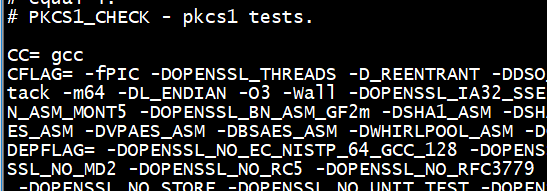
修改Makefile：

vi Makefile

将原来的：CFLAG= -DOPENSSL\_THREADS

修改为： CFLAG= -fPIC -DOPENSSL\_THREADS

也就是添加-fPIC



执行执行：

make && make install

（2）重新安装erlang：

tar xf otp\_src\_20.1.tar.gz

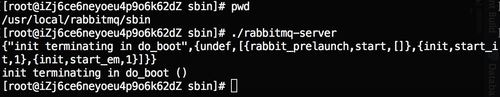
cd otp\_src\_20.1

./configure --prefix=/usr/local/erlang20 --without-javac --with-ssl=/usr/local/openssl

Make && make install

1. 启动rabbitmq-server, 现在应该就可以了！

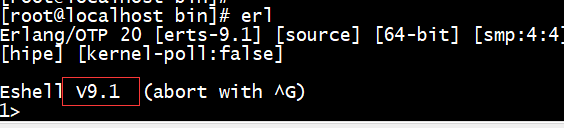
## RabbitMQ安装报错：{init terminating in do\_boot



这个有可能是系统自带的erlang版本太低导致的。

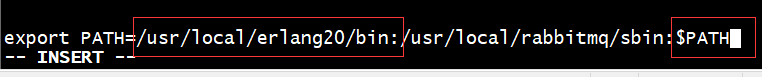
在命令行直接输入：erl

如果输出的版本不是V9.1：

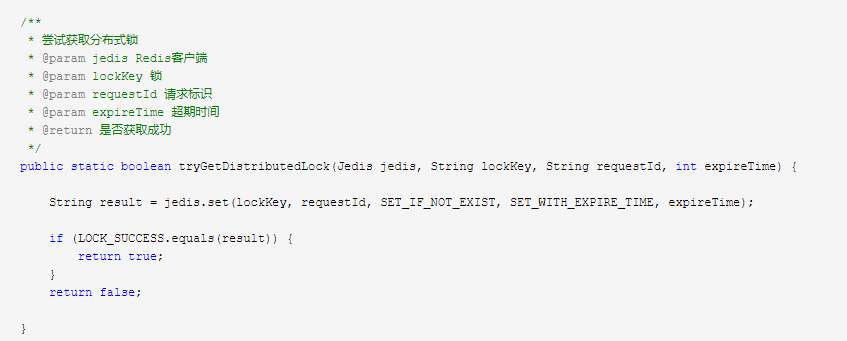


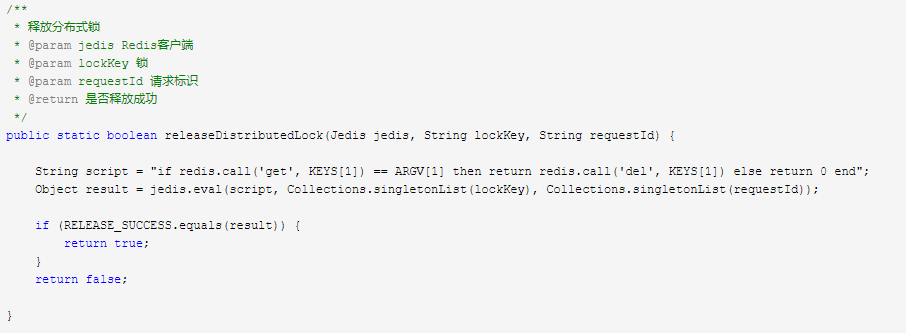
说明没有使用我们自己安装的erlang。只需要调整一下path就可以了：

vi /etc/profile

注意，我们自己安装的放在PATH前面。

## 如何用Redis实现分布式锁？

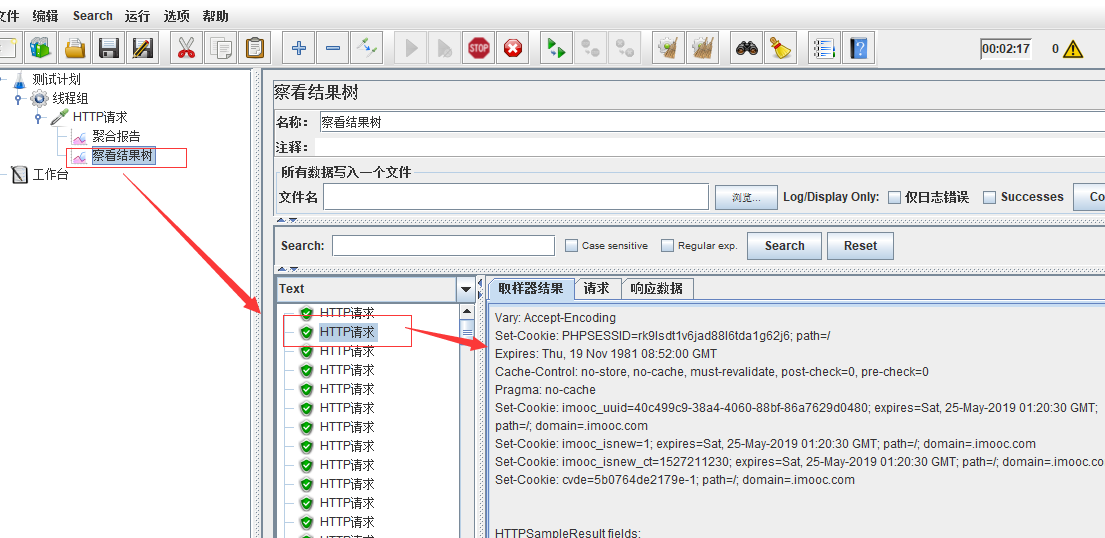




参考：<https://www.cnblogs.com/linjiqin/p/8003838.html>

## JMeter压测的时候如何查找错误的原因？

1. 启动一个线程跑一次，然后在服务器上debug
2. 在jmeter中添加“查看结果树”，可以看到详细的请求和响应信息，如下图所示：



## 关于回答面试问题的几个思路

举一个栗子：

问：如何解决秒杀的超卖问题？

答：有两种情况会导致超卖（1）..（2）。。。，对于（1）.。。。，对于（2）.。。。

这才是正确回答问题的套路，先分析问题，再提出解决方案

错误的回答：什么分布式锁啊 数据库触发器啊 都是扯淡的 你连为啥会出现卖超都不知道，谈什么解决。

再举一个栗子：

问：什么是spring aop？为什么spring要支持aop或者aop有什么好处？

答：aop就是在目标方法执行前可以自定义一些操作，在方法执行中机或者执行后也可以自定义操作，所以，一般都是基于代理模式来实现，spring支持两种代理模式，jdk原生的代理和cglib代理。

aop可以给程序带良好的扩展性和封装性，可以实现业务代码与业务无关代码的隔离。比如：可以在不改变目标代码的前提下，实现目标方法的增强，比如做方法执行时间监控，记录方法访问日志，再比如：数据库的connection.close()默认是把连接关闭掉，但是数据库连接池的场景中，为了不改变用户的使用习惯，一般调用close的时候是把连接重新放回到池中，这是因为从数据库连接池中拿到的连接实际上是原生连接的一个代理类，所以内部把close给重写了。实际上代理模式的优点实际上也是aop的优点。

解析：不光要说明啥是aop，还要说明aop能干啥，当然好处就出来了，同时还说了代理模式，面试官很有可能接下来就会问，jdk的代理和cglib代理有啥区别？或者会问还知不知道别的什么设计模式？这个时候啥是aop已经能不重要了，你已经向面试官展示了你理解而且会用aop，就算你说的是错误的又能怎么样呢？反正你把你自己已经展示出来了！面试官真的在乎你能否准确的说出来aop的精确的定义吗？一定得学会引导面试官问你懂的问题。千万不要简单的就问题回答问题。

错误的回答：aop是面向切面编程？切面什么鬼？单纯的解释字面意思没有任何意义。