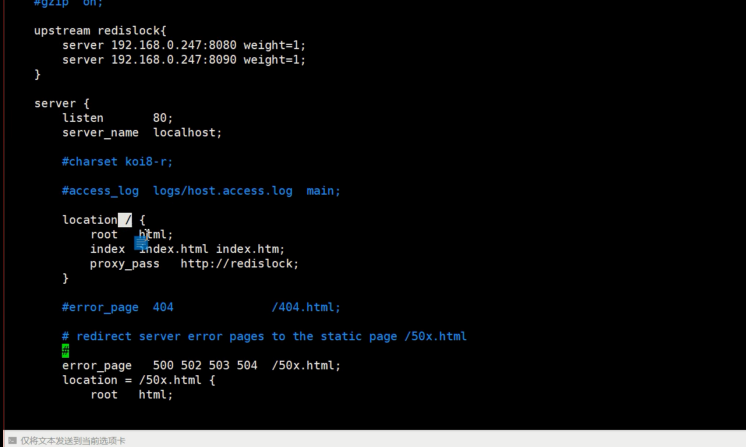
分布式锁：

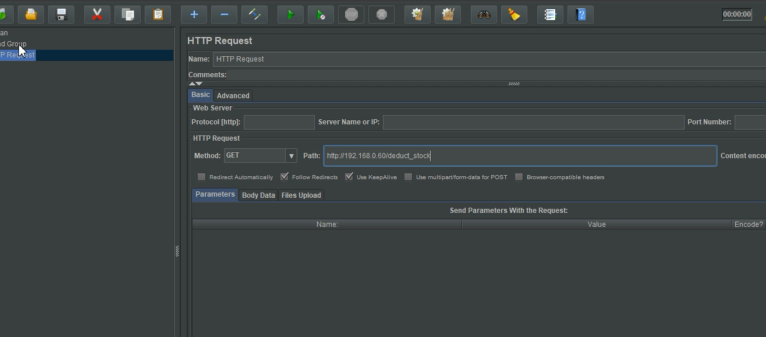
./redis-cli -h 192.168.43.100 -p 6379 --raw -a 123456

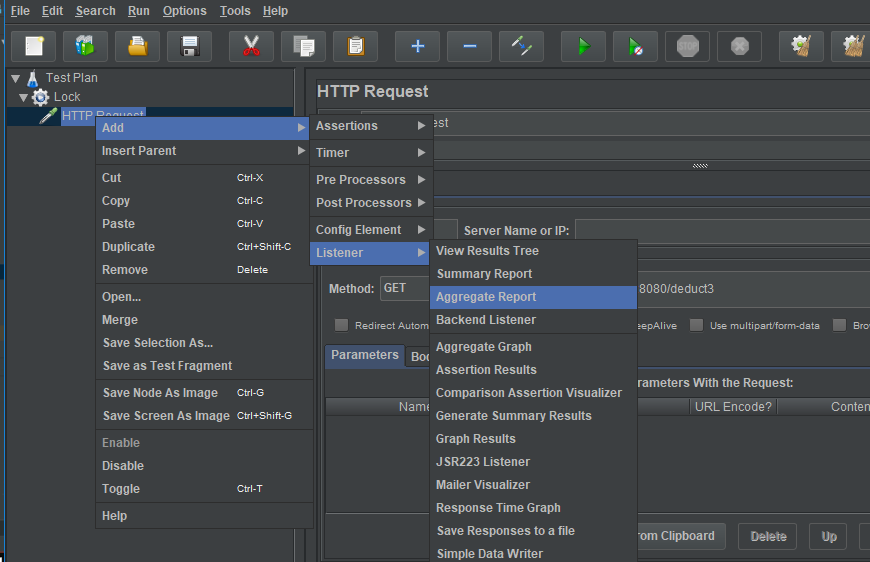


配置Nginx进行轮询：

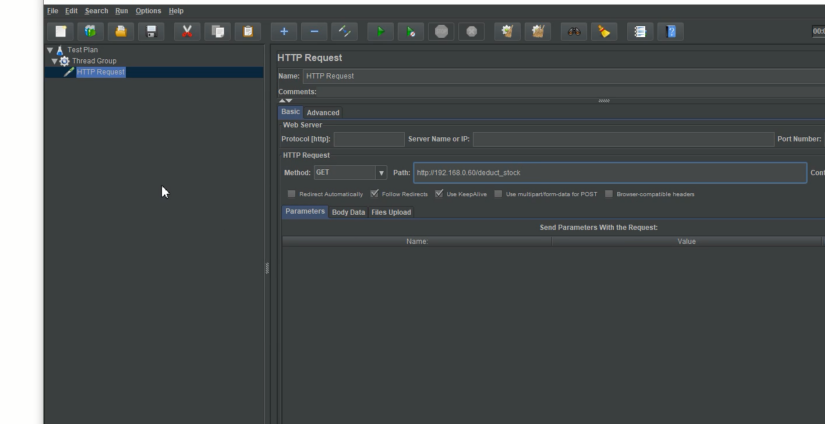
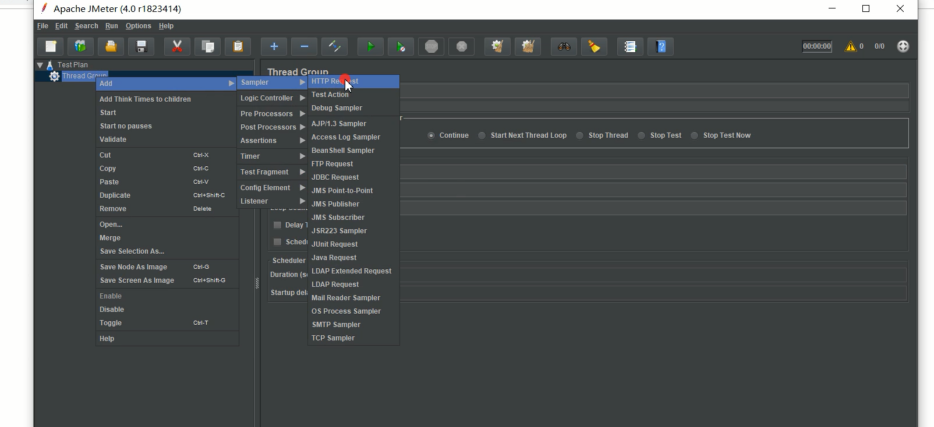
Nginx配置访问虚拟机的ip，轮询到访问Tomcat的端口和ip，weight是Tomcat的ip

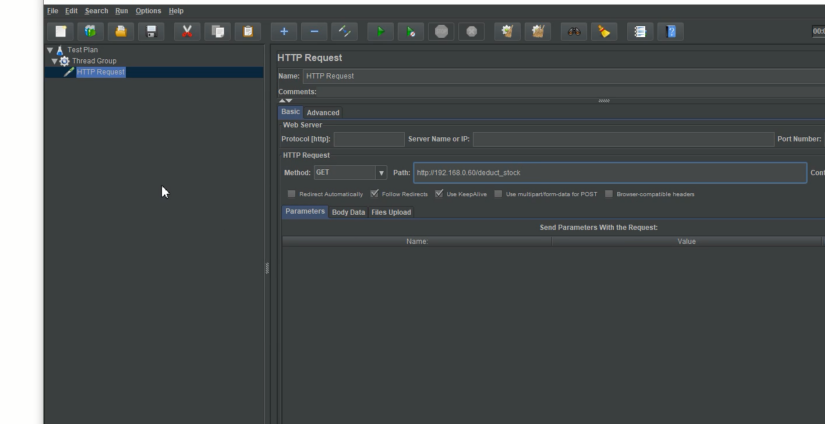






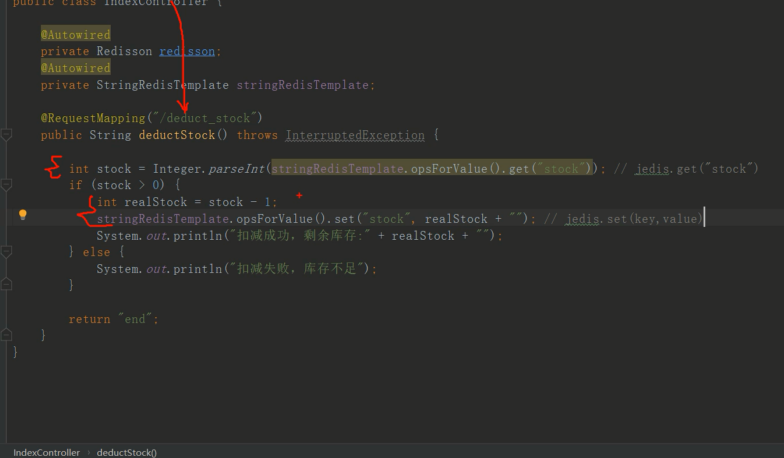
上面是配置监听

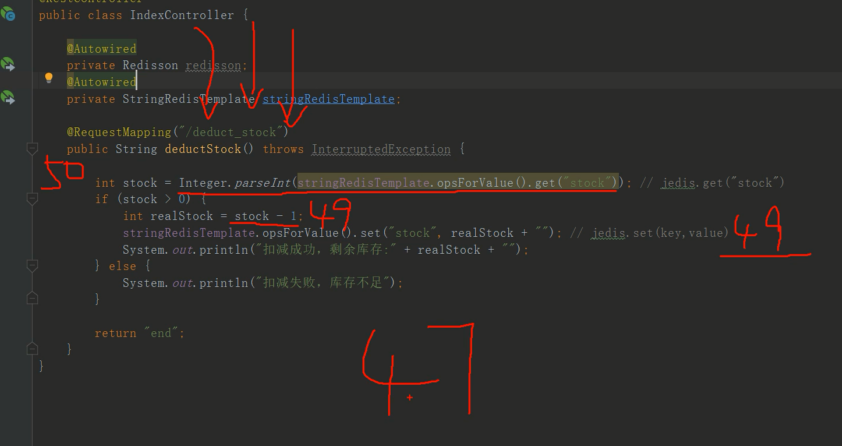


、

配置用户量。

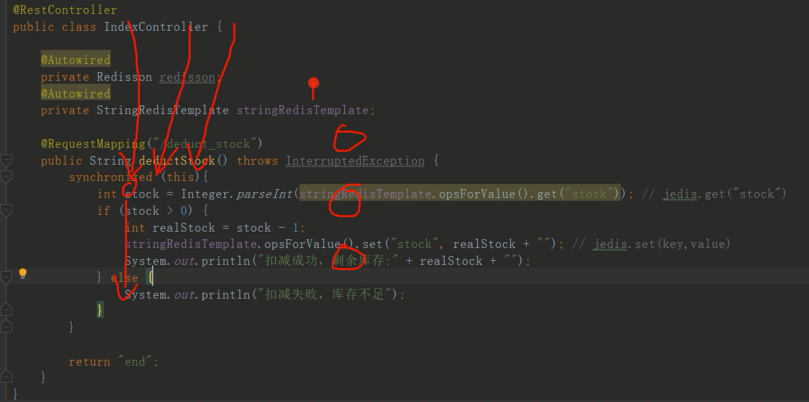
最原始代码：





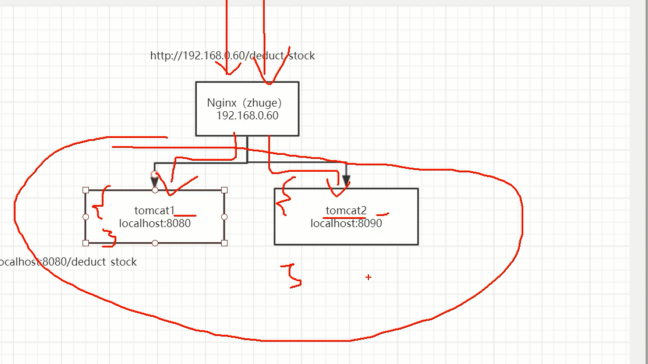
这张图的意思就是三个用户过来秒杀产品，拿到的值都是50，（在redis中存储了一个值

Stock ， 50）,然后每个线程都-1，最后结果是49，按理来说是47，有三个消费者。



单体架构中这样写资源是安全的，没有问题看，但在分布式项目中，前端Nginx部署，然后到集群Tomcat中去，很可能锁不住这个资源，出现问题。

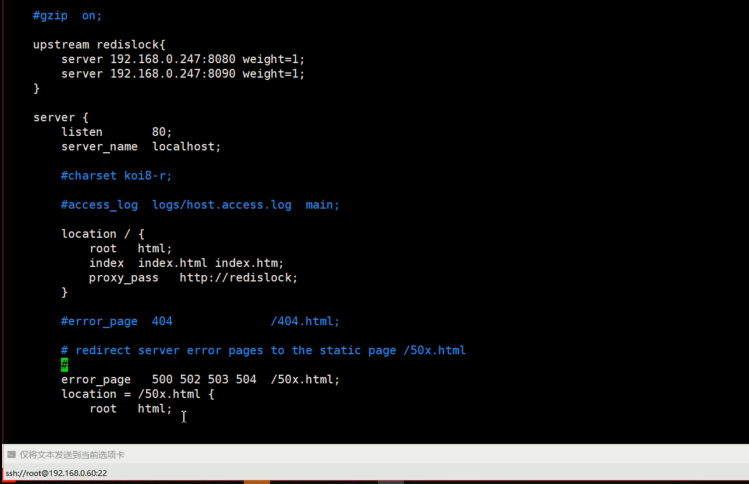
Synchronized是jvm进程级别的锁，如果在两个Tomcat中执行了相同的代码块，问题出现了。（一个Tomcat代表一个jvm）。如：



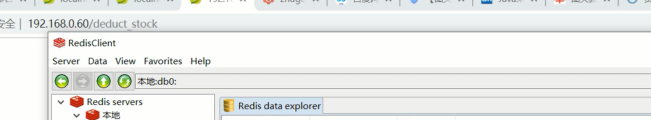
两个Tomcat此时执行了相同的代码块，出现问题。

模拟高并发：

配置Nginx



通过Nginx访问，反向代理。



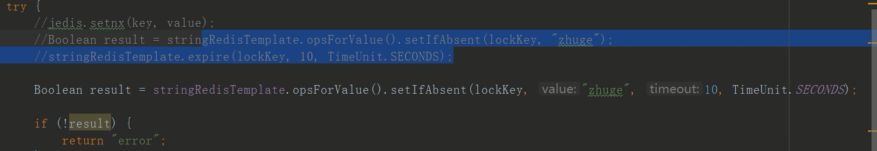
Nginx的IP访问了并发项目。

模拟压测软件，Jmeter

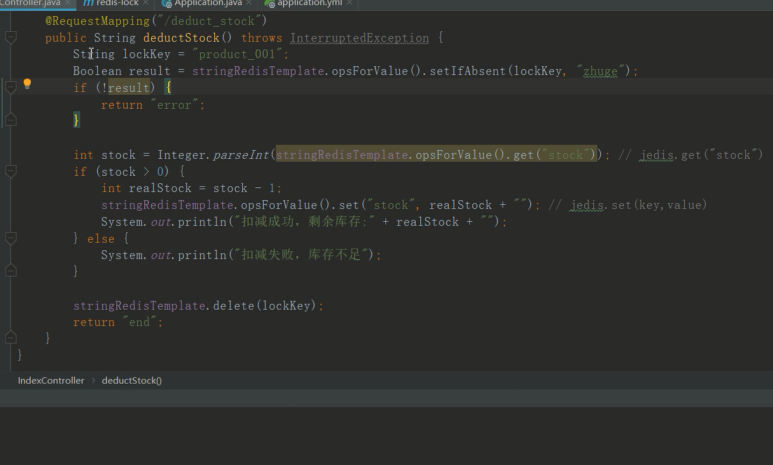
结果出现消费了相同的数据，29

怎么解决，在分布式集群部署环境下。

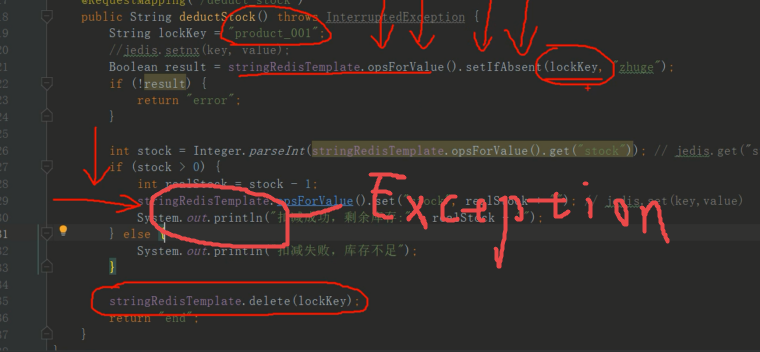
（redis命令手册）



保证原子性，setIfAbsent()

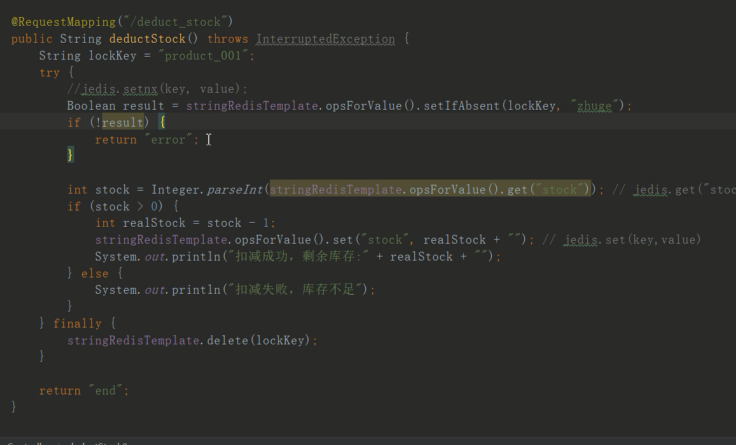


还是有问题；

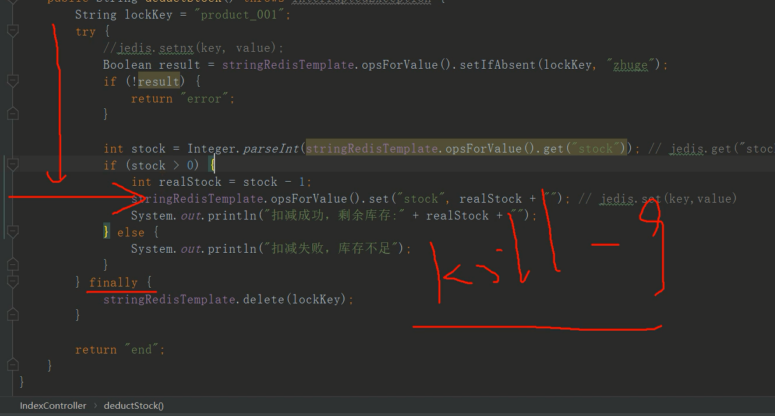


此处有异常的话，没有做删除操作，这个key值一直都在，后面的线程被挡住了，锁没有释放，一直返回。

改善：即使中间代码出了异常，但是最后都还是将存储的值删除掉。

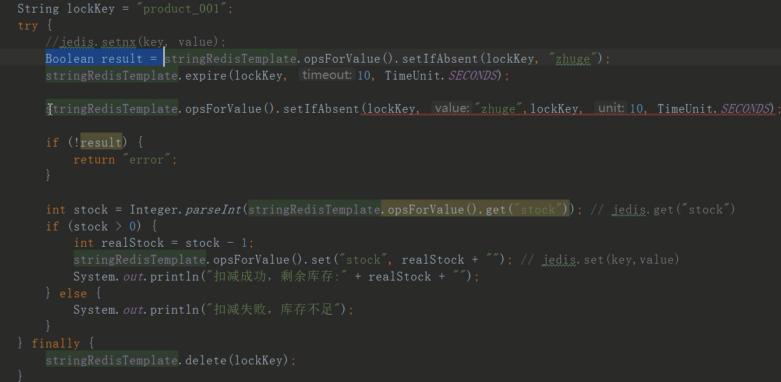


还有问题：



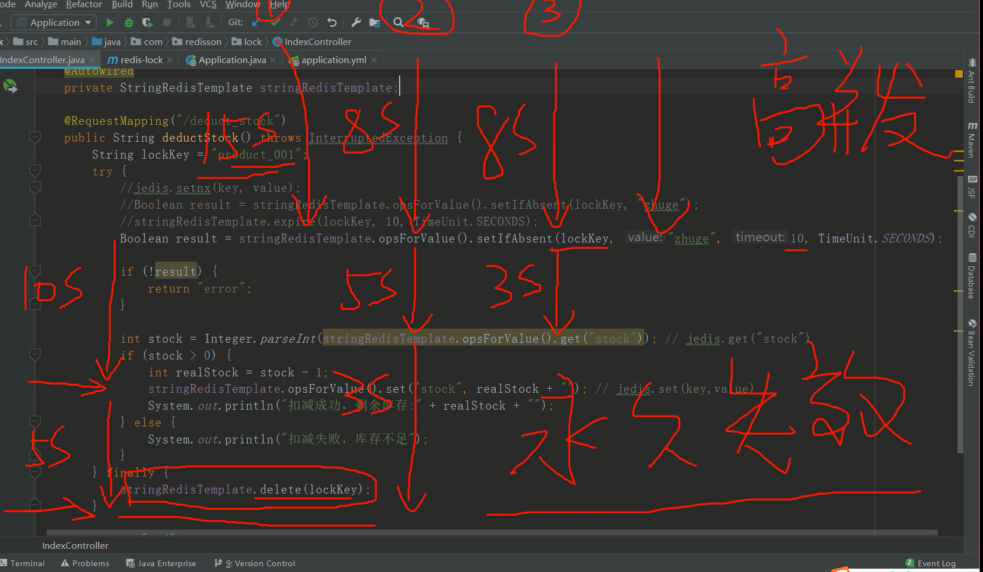
如果这个时候kill -9 杀死了线程，finally没有执行。又意味着product\_001永远的在redis中，

下一个线程进来判断为false，没办法赋值（拿到锁）.死锁。



SetIfAbsent()设置时间，使得两个操作本来不是原子操作，变成原子操作。

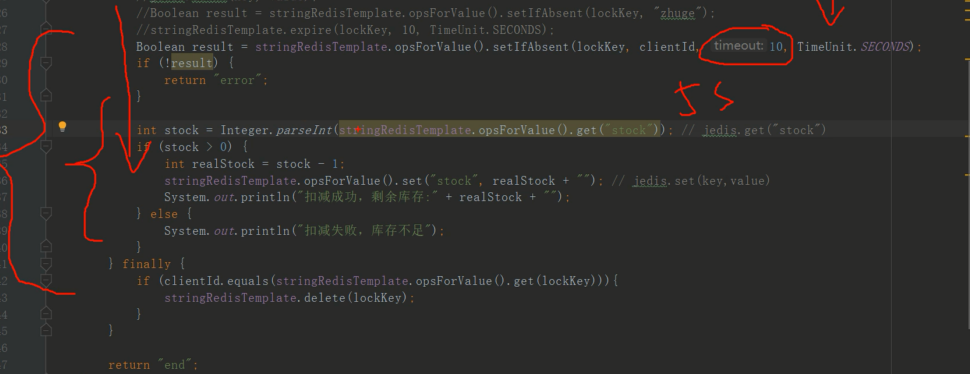
还有问题，有意思。



永久失效之后还是没有对同一资源做安全操作。

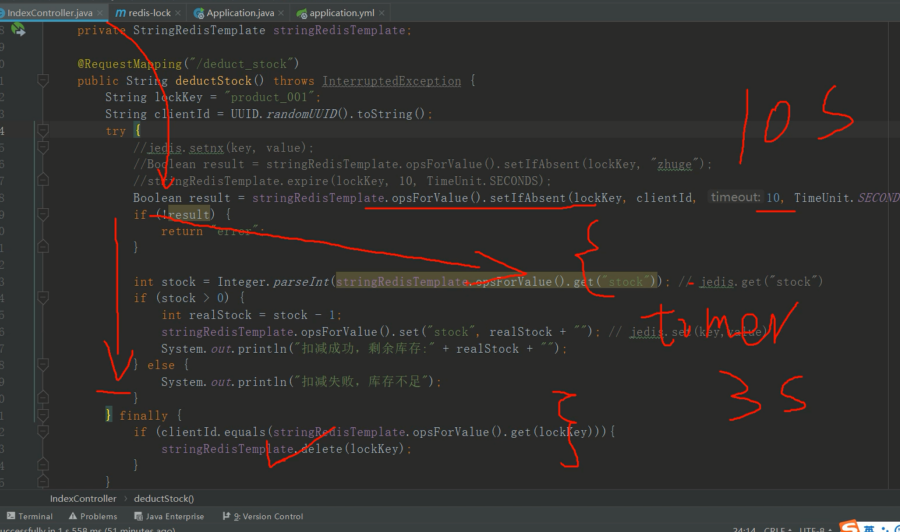
所以不要让他随便释放锁，入的时候和释放的时候做个比对。再才释放。就是针对每一个进来的线程生成一个UUID，然后再释放的时候只有自己的锁才能释放。





假如设置的时间太短，程序没有执行完，设置的锁已经过期了，另一个线程进来，出现问题。

完善：

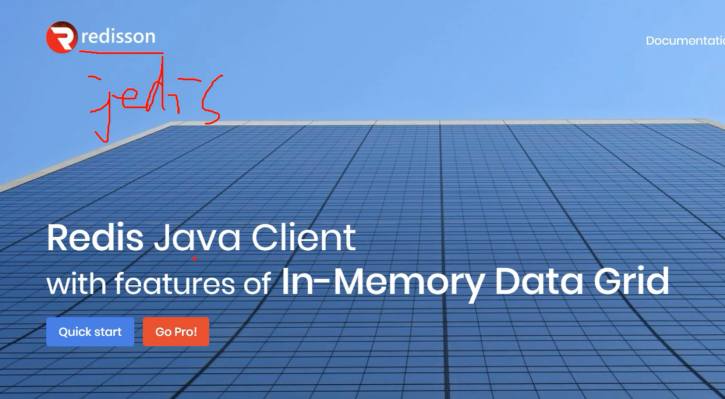


在执行的业务逻辑里面，开启一个分线程，每隔3秒查询一下redis中的key是否

还存在，存在续命，再+10s，正常主线程结束，分线程也结束。

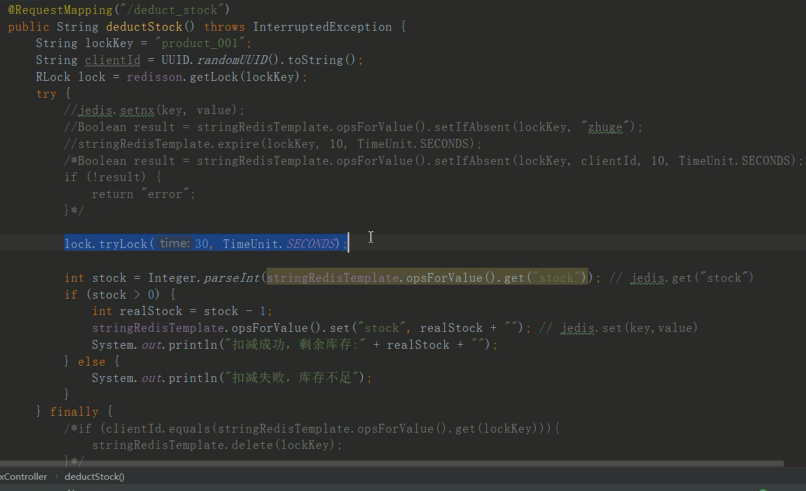
自己写线程代码定时又是一堆坑，所以有框架底层帮助实现，在业务逻辑中

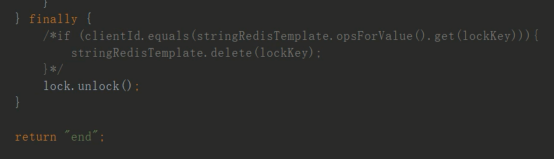
实现分线程。Redisson



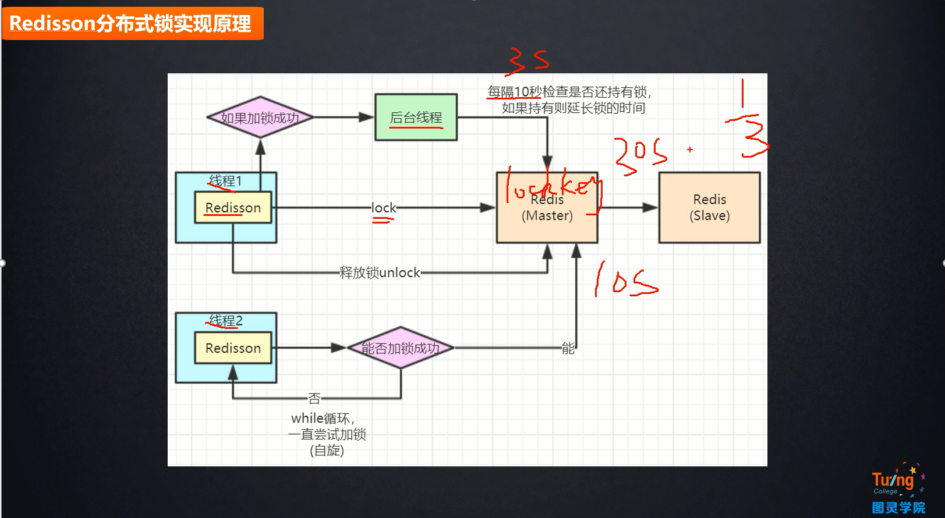
用法：



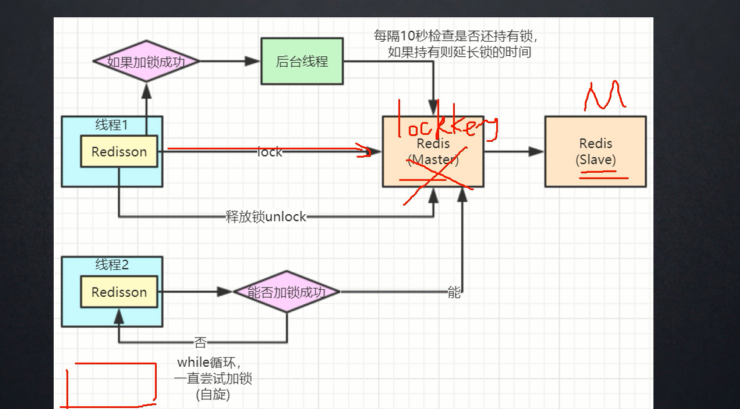




Redisson底层原理



还有问题，如果是redis主从架构。



假如在执行业务逻辑的时候，主从复制有时间延迟，但是这时候主节点挂了，从

节点提升为主，原来的数据还没有完全复制过来，比如没有锁住，线程其他的跑到现在的

节点上去了，都可以访问统一资源，出现问题。

还有在秒杀的时候怎么提升性能，它是单线程的，怎么变成多线程。