数据库设计:

商品表,订单表,秒杀商品表(通过商品ID关联到商品表)

商品表

```
CREATE TABLE `goods` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '商品ID',
  `goods_name` varchar(16) DEFAULT NULL COMMENT '商品名称',
  `goods_title` varchar(64) DEFAULT NULL COMMENT '商品标题',
  `goods_img` varchar(64) DEFAULT NULL COMMENT '商品的图片',
  `goods_detail` longtext COMMENT '商品的详情介绍',
  `goods_price` decimal(10,2) DEFAULT '0.00' COMMENT '商品单价',
  `goods_stock` int(11) DEFAULT '0' COMMENT '商品库存，-1表示没有限制',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4;
```

订单表

为什么不在商品表里添加秒杀商品
随着时间的增加,活动越来越多,如果每次活动都给商品表添加新的字段的话,时间长了这个表会非常难以维护,且不稳定,代码也需要跟着一起修改.

这里我们查数据库的时候，不只是查找的商品的信息，我们同时想把商品的秒杀信息也一起查出来，但是这两个不同数据在两个表里面，我们就想办法封装一个GoodsVo，将两张表的数据封装到一起。然后再自己定义MiaoshaGoods里面的字段，最终拼接成一个GoodsVo对象,返回给DAO层。

这里是将两个表做连接查询（miaosha_goods mg left join goods g）

秒杀倒计时:

后端根据商品的Id去数据库中获取秒杀开始时间和结束时间，以及系统当前时间，并定义秒杀剩余时间变量和秒杀状态，计算出相应的值，传给前端，前端拿到之后，做相对应的显示逻辑效果。

1. 在GoodsController里面创建toDetail方法，接收详情页面的请求

注意：获取了商品的秒杀开始时间和结束时间，如果秒杀没有开始，那么计算一个还剩多少时间，开始，并且定义一个状态status来表示一个秒杀的状态，0代表秒杀还未开，1代表秒杀正在进行，2代表秒杀已经结束，秒杀还未开始的情况还要计算出倒计时，(int) ((start-now)/1000)，然后将status和remailSeconds传到前端去。

前端需要获取status和remailSeconds（即秒杀状态和剩余时间变量），定义个一个属性为隐藏的input来接收remainSecode ，并且定义标签来判断status的状态，通过这个值来显示是否开始秒杀，秒杀正在进行中，以及秒杀结束。

2. 先去设置数据库里面的秒杀时间
   假设当前时间是2019-05-28 19:30:12
   1.已经开始
   2.秒杀结束
   3.秒杀倒计时

秒杀业务主要逻辑：

- **判断登录**
- **根据商品id从数据库拿到商品**
- **判断库存，库存足够，进行秒杀，不足则结束**
- **判断是否重复秒杀**（我们限制一个用户只能秒杀一件商品，怎么判断？即从数据库根据商品和用户id **查询秒杀订单表**，如果已经存在订单，说明重复秒杀 ，给出提示，退出）
- **以上都通过，那么该用户可以秒杀商品**

注意：
执行秒杀逻辑是一个**原子操作，是一个事务：**

- **库存减1**
- **下订单（写入秒杀订单）**

所以使用@Transactional注解标注，其中一步没有成功，则回滚

压测结果:

对秒杀接口的压测QPS吗？，1000个线程数，循环次数为10，QPS是703左右