## 动静分离：

Static css imge js 放入nginx 中

使用缓存：访问量大 的时候， 商品目录，商品信息，

Jvm 调优：

## 压测：

## 异步编排：

3 4 5 要依赖1 的结果

让1 执行完成之后 ，其它并发执行

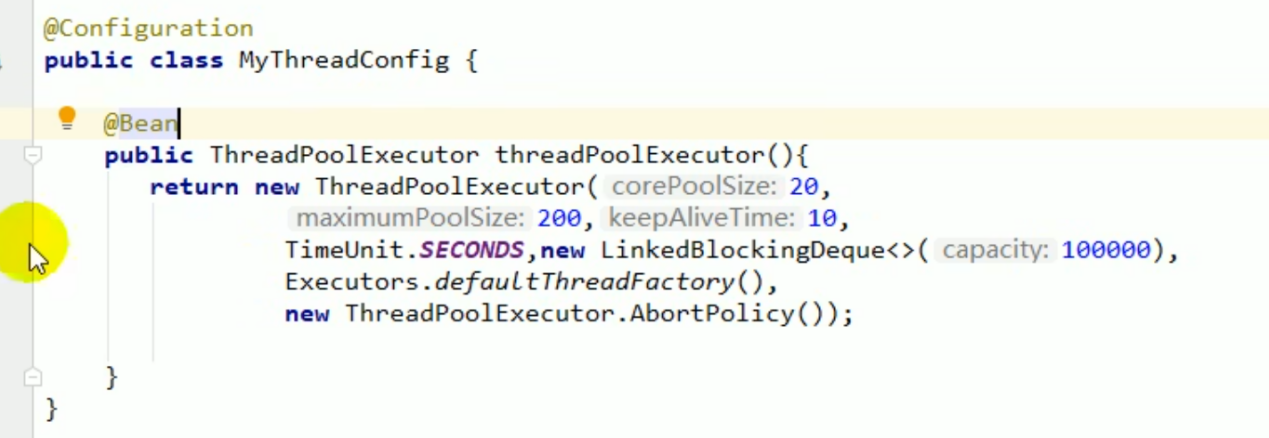


## 优化商品详情逻辑 数据库io

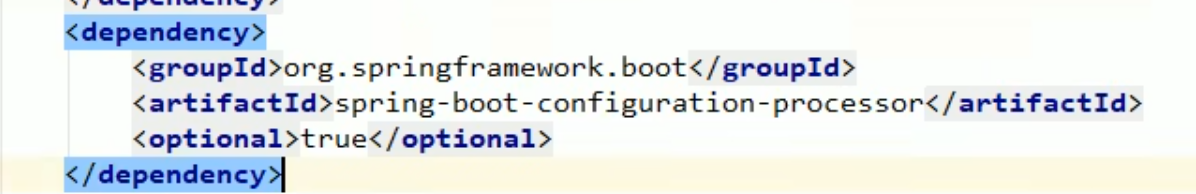
3 4 5 要依赖1 的结果

让1 执行完成之后 ，其它并发执

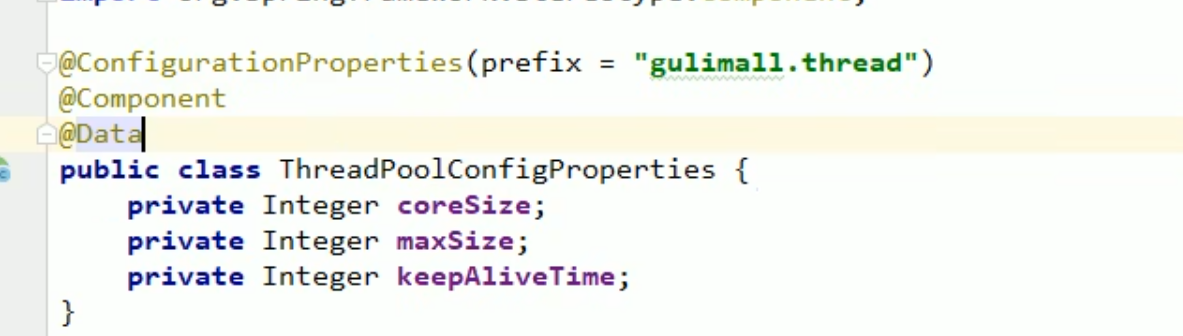
在业务中添加线程池

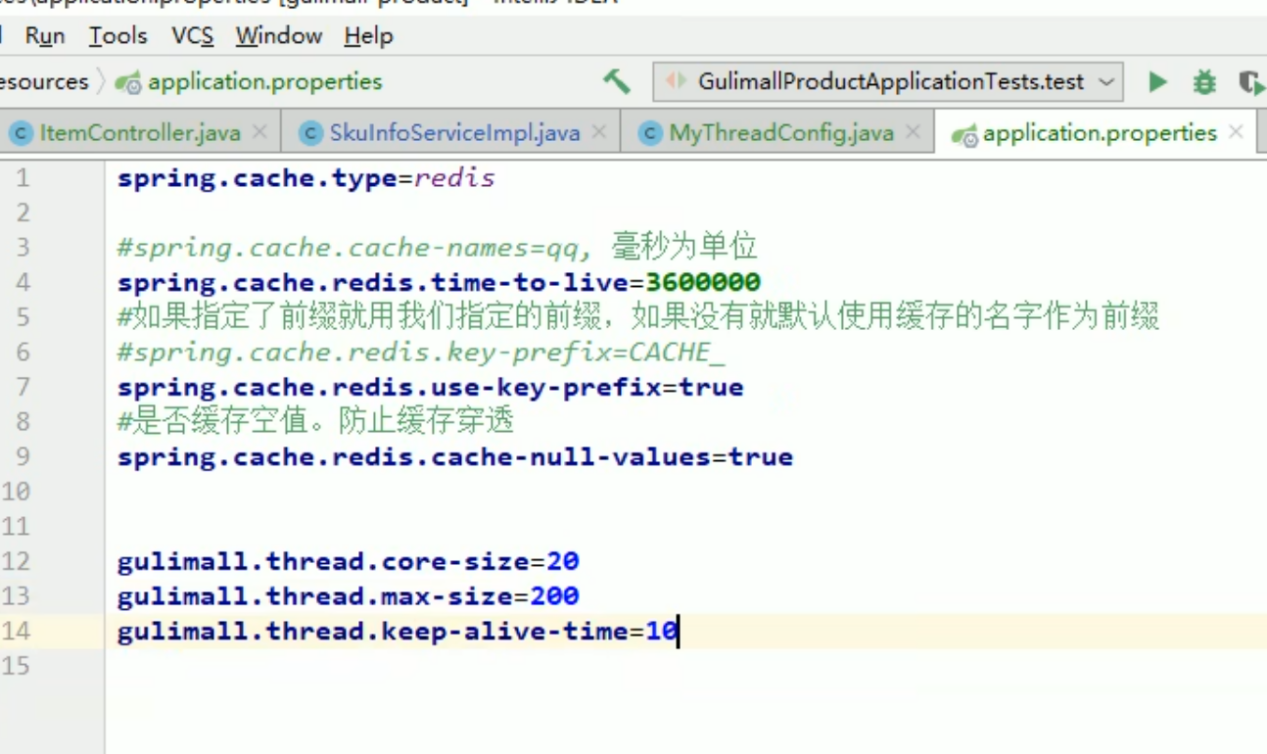


添加属性提示工具



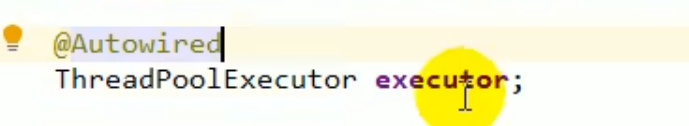
绑定配置属性

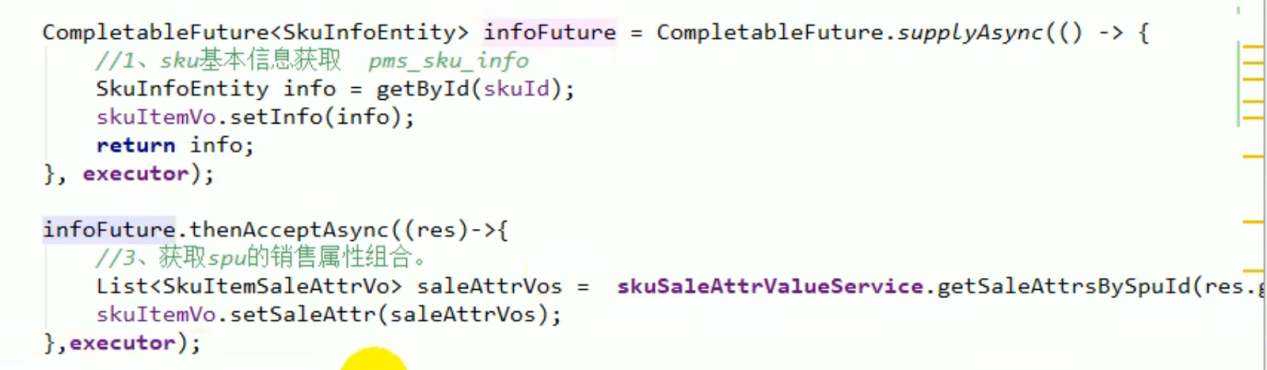


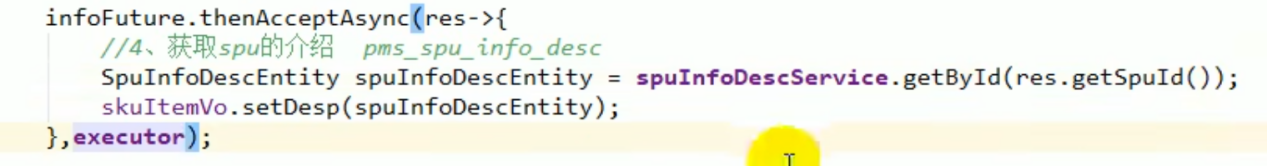


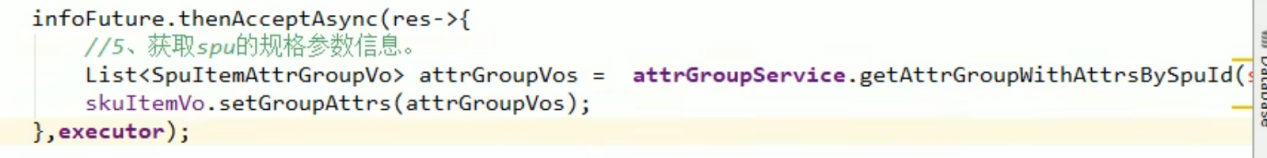


注入线程池



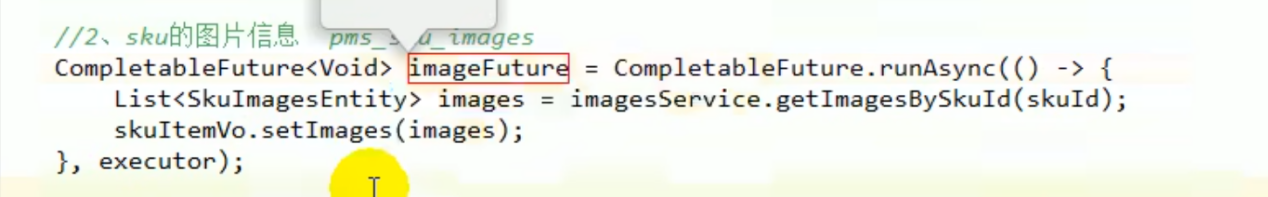


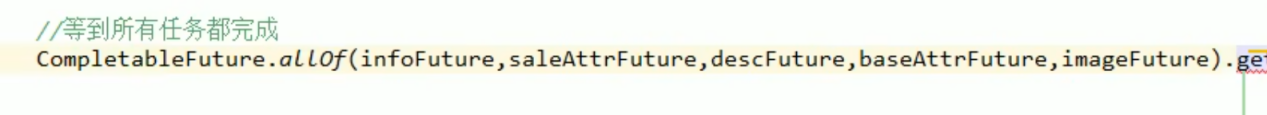




2和1 没关系

所以还可以开启一个异步





业务编排 线程池 提升响应性能

购物车模块异步编排



## ThreadLocal：

使用threadlocal 来进行session 的存储，

每个服务中interceptorhandler 拦截器 将springsession中的 session 存到threadLocal 中。

## Mysql优化 mrr 在内存中排序：

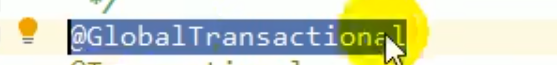
## 分布式 :柔性事务 确认消息 最终一致性

## 为什么不用seata At 模式

At 模式不适用高并发场景

保存spu sku coupon模块 积分信息 优惠券信息,这些不需要高并发的，直接使用

GlobalTrasaction seata At 模式



使用seata at 模式

下单是高并发场景

有流量进来

At 会串行化 系统就没法用了

下单和关单顺序

下单成功后，放入延时队列

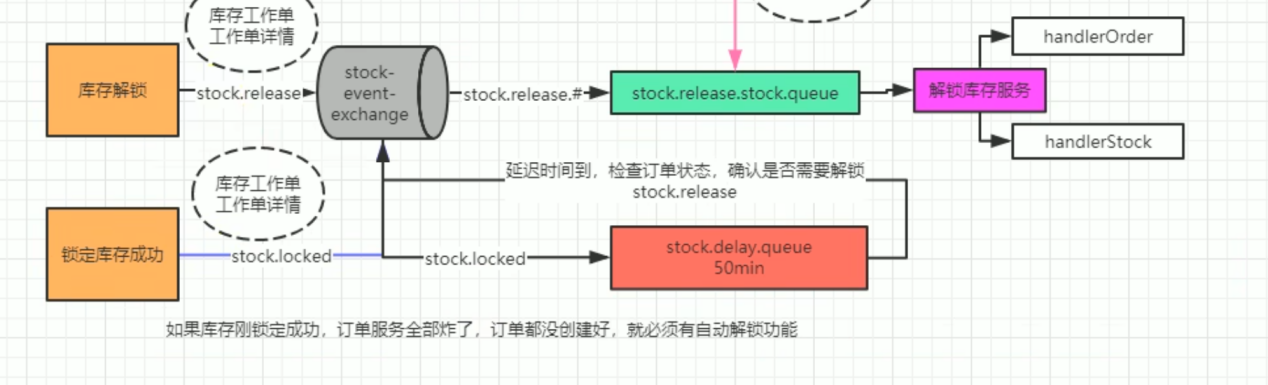
30min 后才会被监听者监听到

监听者查询，如果未支付，就关闭

所库存成功 发消息，到时间 ，检查订单 如果没成功 ，则解锁库存

At 模式不适用高并发场景

保存spu sku coupon模块 积分信息 优惠券信息



## Redis 秒杀的并发

# SpringCache 缓存

默认使用redis

@cachable （value=”{category}”

@putCache 双写模式

@evictCache 失效模式 canal 读取mysql binlog 进行同步

# 跨域处理

使用网关会造成跨域；没有办法执行非同一与下的js

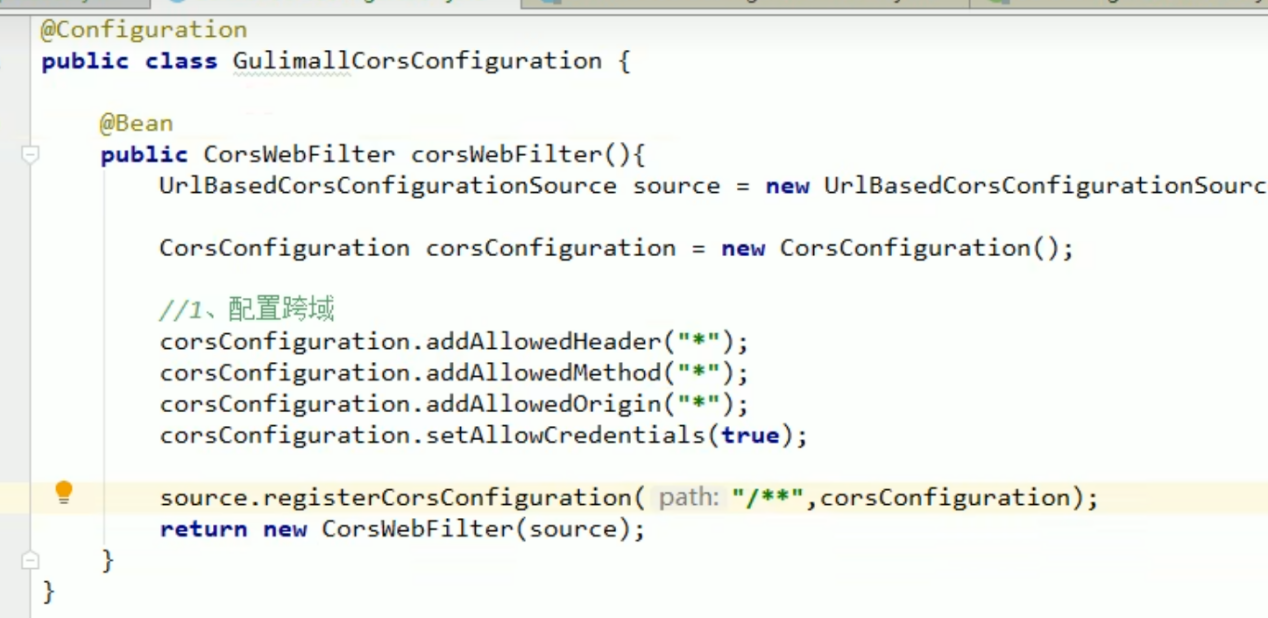


使用拦截器

写一个filter

写在网关里面

使用webflux reactive 响应式编程



配置跨域之后重启网关

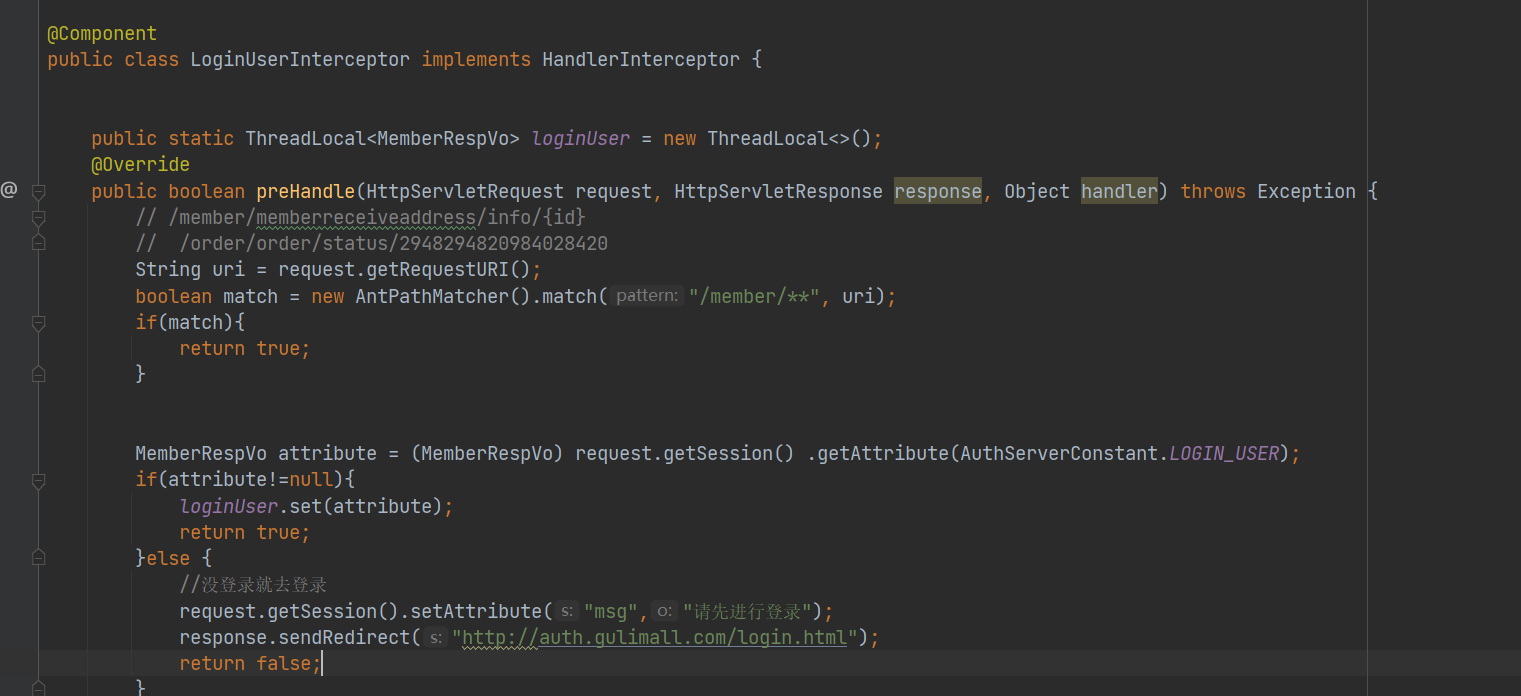
# Springsession实现子域共享session



将cookie域名设置为父域



Member 服务中将session 存到threadlocal 中 方便使用



使用request 过滤器 去redis 中取得



通过filter 将 redis 的session 设置到 session.getSession()的方法放回值改变

# 幂等性：

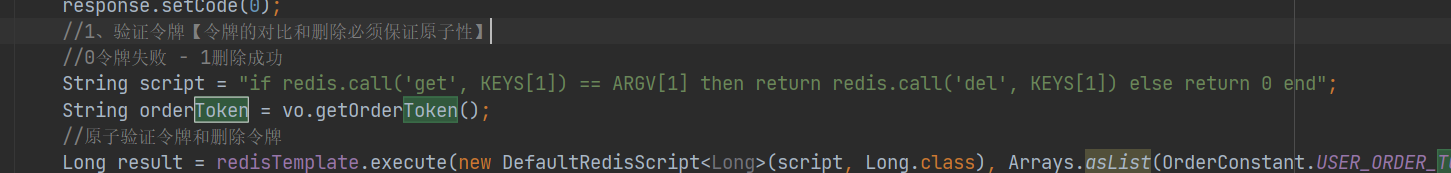
可以使用令牌机制 ，悲观锁，乐观锁，分布式锁，先判断是否处理过这个数据，

防虫表，存在就不插入， redis 全局请求唯一id，

token机制，防止页面重复提交。业务要求： 页面的数据只能被点击提交一次；发生原因： 由于重复点击或者网络重发，或者nginx重发等情况会导致数据被重复提交；解决办法： 集群环境采用token加redis(redis单线程的，处理需要排队)；单JVM环境：采用token加redis或token加jvm内存。处理流程：1. 数据提交前要向服务的申请token，token放到redis或jvm内存，token有效时间；2. 提交后后台校验token，同时删除token，生成新的token返回。token特点：要申请，一次有效性，可以限流。注意：redis要用删除操作来判断token，删除成功代表token校验通过，如果用select+delete来校验token，存在并发问题，不建议

防止令牌

提交订单 生成uuid ，回显到前端，



Lua脚本删除令牌，

# RabbitMQ：

Fanout

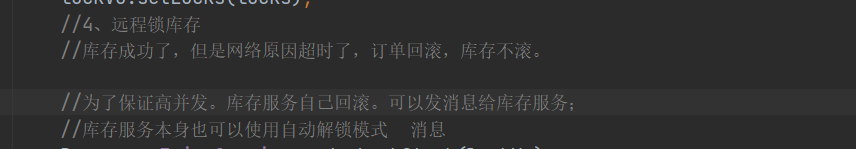
Direct

Topic

手动ack

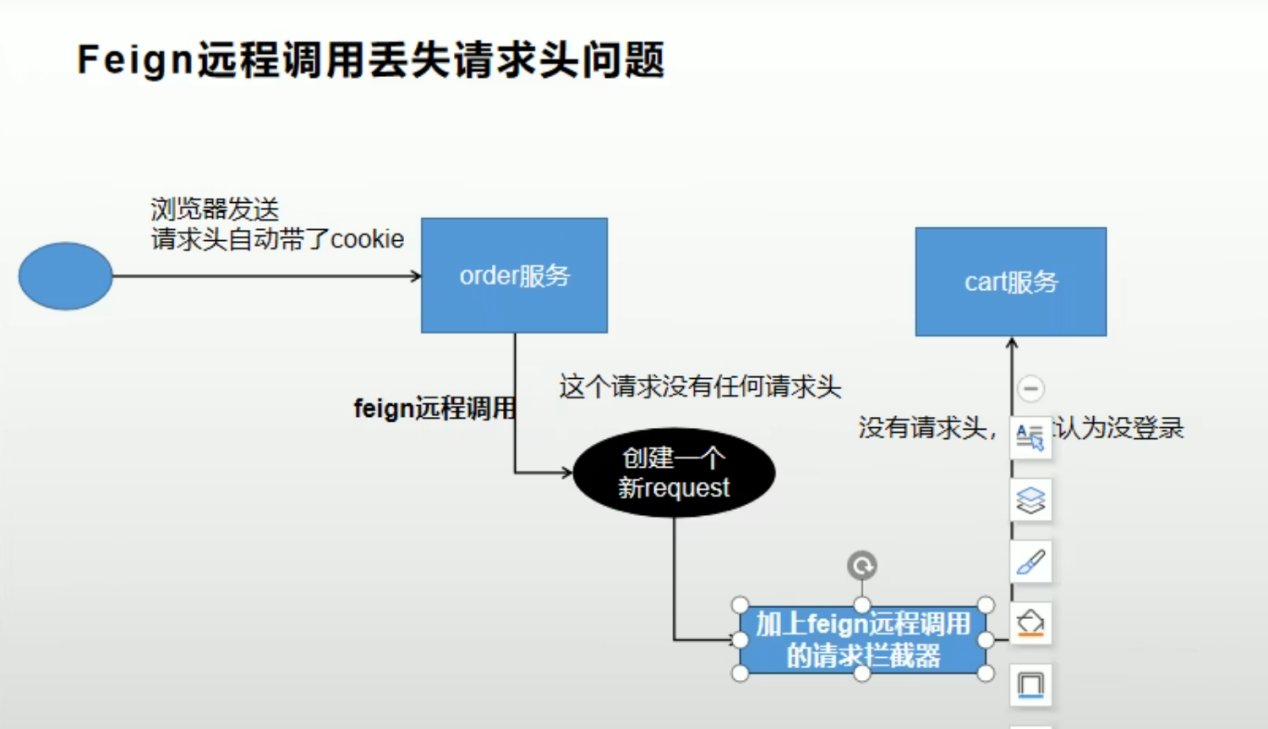
可靠抵达 p-broker confirmcallback

Exchange->queue return callback ack

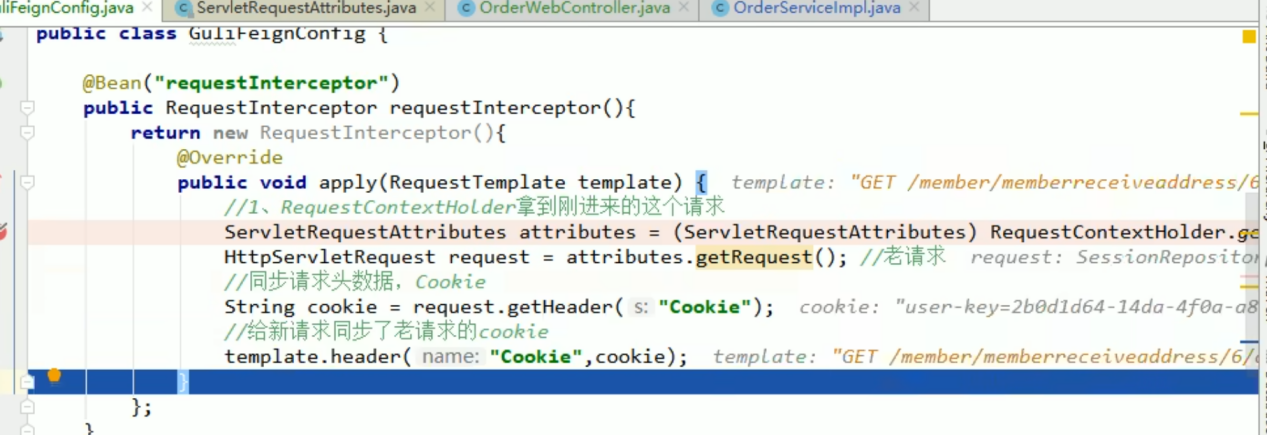


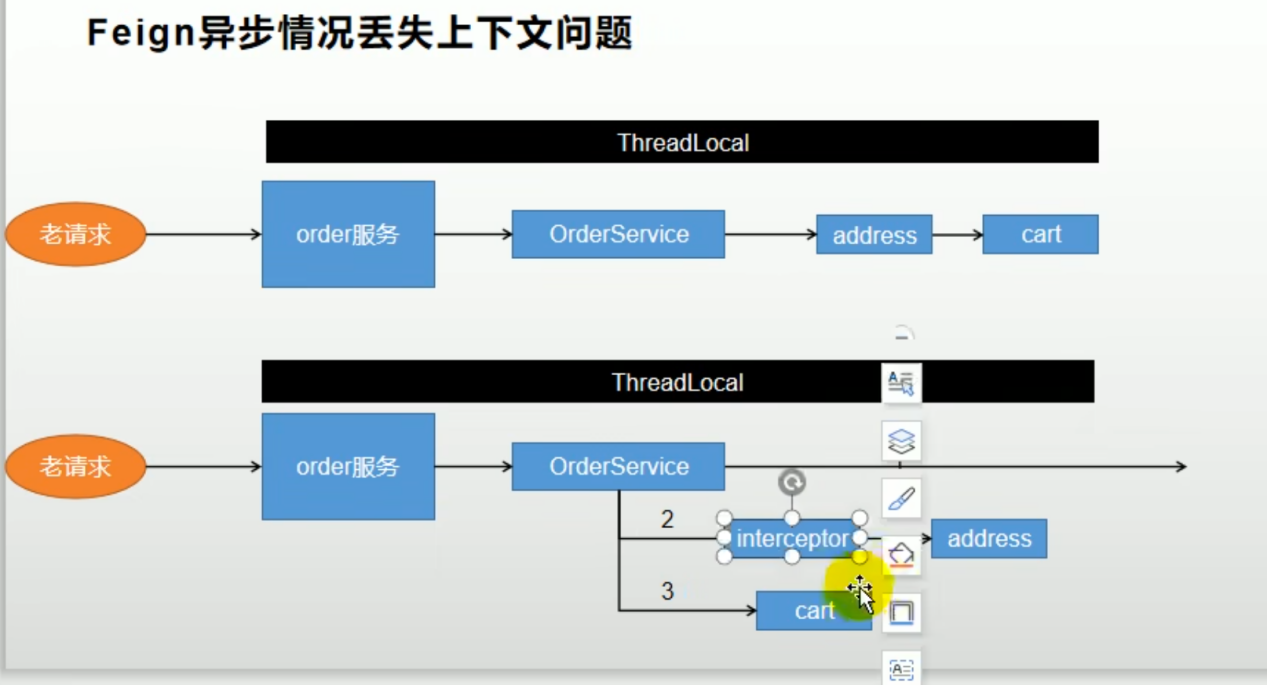
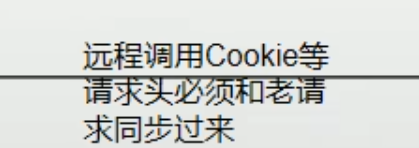
# Feign 远程调用请求头丢失的问题

订单服务 和商品服务



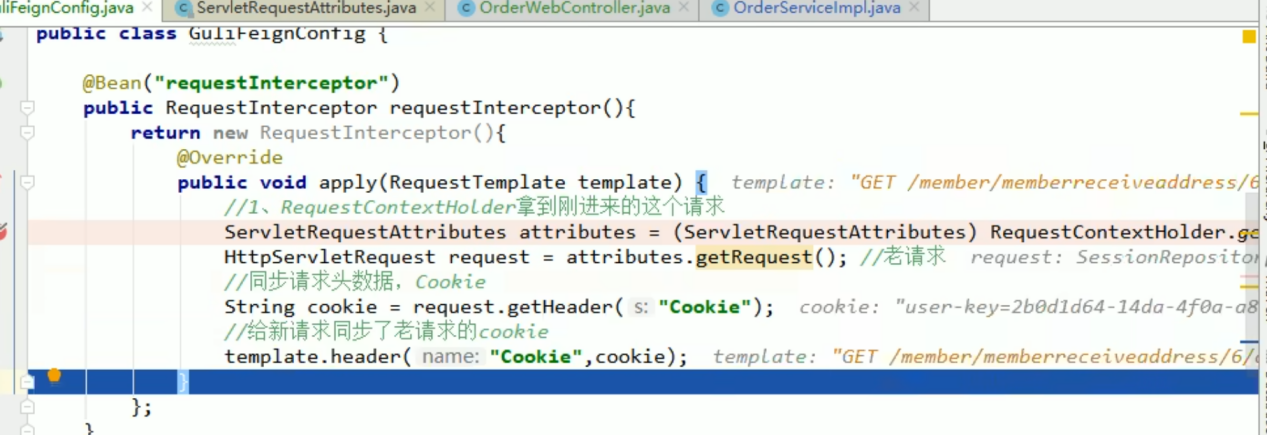
远程服务无法获取session 中的用户对象





订单模块微服务，

tihuandaio



给template 的新请求替换掉 老的请求

Request context holder 是基于threadlocal 的

# Jvm优化

Jvsuanm： 方法区内存暴增，方法区打印 不断创建这个类 verbal：class 类信息 饱满 引发频繁GC，就给类加上缓存 ，解决了

Mysql 优化

# rabbitMQ：

答疑：

1.mq如何处理高并发？

答： mq本身只是用于接收数据，处理高并发 还是得通过线程池多线程去业务处理，但是 为了保证用户提交的数据不丢失，引入mq。

mq默认是轮询向每个接收方发送消息，可以设置每个队列接收的 QOS大小，以及在接收方收到消息后，

deliverCallback 确认回调加入线程池，将消息先存入线程池多线程处理数据，以保证队列不拥堵，解决一部分高并发问题，这里的高并发只是mq的解决方式，

深入高并发不是一个中间件可以搞定的，有机会会给大家介绍其他的中间件解决高并发的问题。

2.负载均衡mq怎么实现的？

答：由于mq本身在向每个接收方发送消息时 是轮询状态 保证每个接收方可接收到的消息都是一样的数量，这里我们抛开QOS 设定大小的问题。

3.消息幂等性、丢失怎么办？

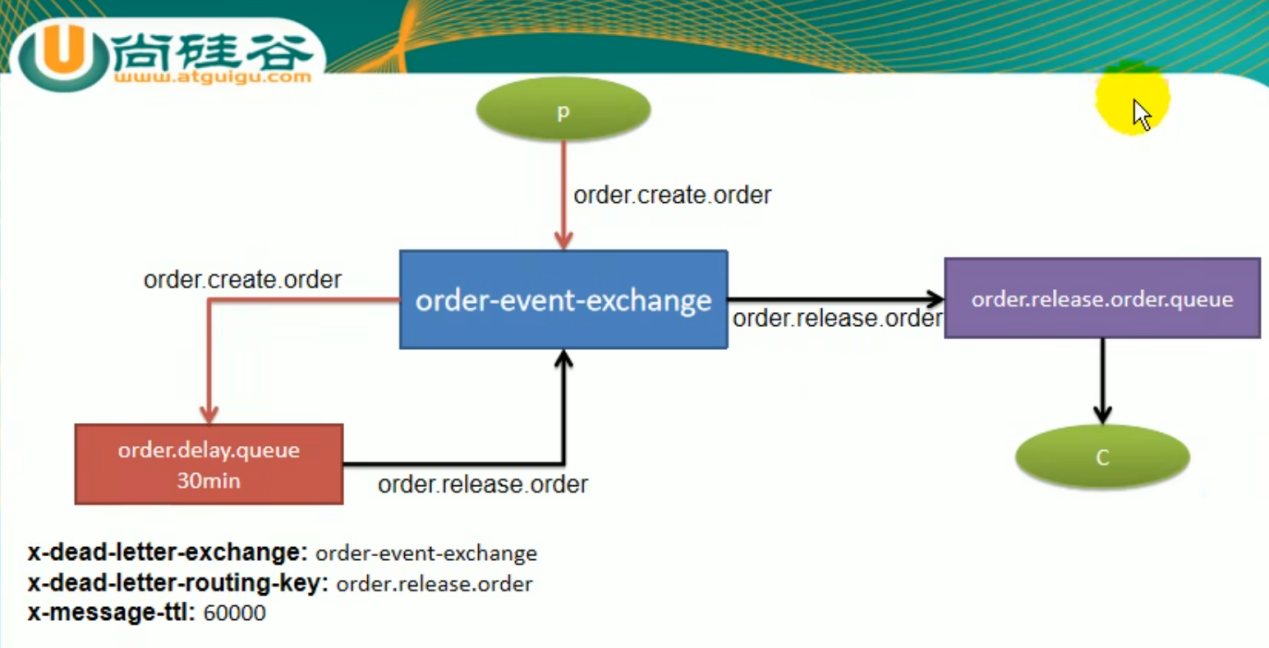
答：每条队列中的消息 都应该有一个唯一标识，比如：id、订单号等，保证数据在处理时不会重复处理，并且队列创建时 也是判断了 不存在即创建，存在即不会创建。

丢失的问题 一般存在于 消息接收方一直接收异常 导致消息被驳回数量超过5次，此时应将消息存入 异常队列 或 其他中间件（redis）中 并记录日志，待夜深人静的时候.........我们程序媛就开始活动处理这些异常数据了！

4.消息顺序错乱问题？

答：这个问题有点扯，队列本来就是先进先出的啊，有顺序的。 当遇到驳回异常数据时，我们尽量驳回到异常队列中，避免顺序错乱。

# 定时关单 功能



使用延迟队列，回到order event exchange 交换机当中

