黑马点评

1. 短信登录
2. 生成验证码

判断手机号格式是否正确->随机生成6为验证码，将验证码保存在redis（键值对，设置过期时间和时间单位）

1. 登录或注册

判断手机号格式是否正确->从redis中获取验证码与用户输入的验证码进行比对校验(@requestBody)->验证码正确，根据手机号判断数据库是否有对应的用户数据，没有便创建新用户->用uuid生成一个token->将user转为HashMap->用token作为键，map作为值存储在redis中，设置有效期，返回token给浏览器。

1. 校验登录状态，刷新token时间

设置两个连接器（为了让拦截去刷新token时间对一些不需要拦截的路径生效），将刷新token时间的功能放在第一个拦截器中，需要拦截器同意处理的其他功能放在第二个拦截器中，mvc配置拦截器使其生效。

3.1、一级拦截器：获取token，判断token是否为空->根据token获取用户，判断用户是否有内容->将用户保存到Threadlocal中，刷新token有效值。

3.2、判断Threadlocal中是否有用户，有就拦截，没有就返回401状态码。

1. 优惠券秒杀（重点）
2. 基于乐观锁解决超卖问题

在删减库存的时候都必须判断库存是否大于等于1

1. 一人一单限制功能：

2.1、一人一单限制

查询订单表格中对应用户id和订单id的信息数是否大于等于1，有则说明该用户已经抢购过优惠券了。

2.2、单机模式下的线程安全问题

在创建订单的方法外添加悲观锁。

2.3、集群模式下的线程安全问题

使用分布式锁，初步使用redis（string 结构）的setnx来实现，插入成功表示获取锁成功，否则表示获取锁失败。

2.4、锁误删问题

在释放锁之前要判断线程标识是否与之前存储的线程标识是一致的。线程标识=uuid随机序列+线程id。

2.5、锁的原子性操作问题（判断完线程标识，即将要释放锁的时候锁过期了，发生并发安全问题，删除了其他人的锁）

使用redis 的lua脚本统一实现线程标识的判断和释放锁的操作，保证原子性。

2.6、使用专业的Redisson分布式锁（可重入（hash结构）、可重试（发布订阅结合信号量）、看门狗机制（解决超时释放问题）、主从一致性）

3、Redis实现秒杀优化

* 新增秒杀优惠券的同时，将优惠券信息保存到Redis中
* 基于Lua脚本，判断秒杀库存、一人一单，决定用户是否抢购成功
* 如果抢购成功，将优惠券id和用户id封装后存入阻塞队列
* 开启线程任务，不断从阻塞队列中获取信息，实现异步下单功能

基于阻塞队列的异步秒杀存在哪些问题？

\* 内存限制问题

\* 数据安全问题

4、使用redis的Stream消息队列实现异步秒杀下单

* 创建一个Stream类型的消息队列，名为stream.orders
* 修改之前的秒杀下单Lua脚本，在认定有抢购资格后，直接向stream.orders中添加消息，内容包含voucherId、userId、orderId
* 项目启动时，开启一个线程任务，尝试获取stream.orders中的消息，完成下单

STREAM类型消息队列的XREADGROUP命令特点：

\* 消息可回溯

\* ==可以多消费者争抢消息，加快消费速度==

\* 可以阻塞读取

\* ==没有消息漏读的风险==

\* ==有消息确认机制，保证消息至少被消费一次==

1. 达人探店
2. 发布笔记
   1. 发布图片

获取图片原来的文件名，根据该文件名创建新的文件名，利用新的文件名将图片保存在指定的路径下（模拟的情况下），返回文件上传成功的提示。

* 1. 保存（发布）笔记图文

直接数据库中保存笔记信息（同时将当前用户的id保存在blog里的userId字段中，方便以后查找对应用户），返回笔记id给浏览器。

1. 查看探店笔记

2.1.查找热门的笔记，按点赞数从高到低排列分页显示

定义page对象，传入当前所处的页数位置以及页的大小，利用page去查找所有的笔记信息，对所有的笔记信息进行循环遍历，查找每条笔记对应的发布用户，将用户的头像和昵称进行赋值。同时判断当前用户是否点赞过该笔记，返回所有的笔记信息给浏览器。

2.2.点击指定笔记，显示笔记内容

根据笔记id查找对应的笔记信息，查询笔记对应的发布用户，将用户的头像和昵称进行赋值。同时判断当前用户是否点赞过该笔记，返回笔记信息给浏览器。

1. 点赞

获取当前的用户id，在缓存中判断指定的笔记key里面是否存有对应的用户id，如果没有，说明当前用户未给该笔记点过赞，则数据库点赞数+1，缓存存入该用户id，反之，数据库点赞数-1，移除缓存中的用户id。

1. 点赞排行榜

根据笔记id生成key，根据key在缓存中查询前5的点赞的用户id，解析用户id，根据用户id在数据库查找用户信息，复制用户的id，头像和昵称信息给UserDTO对象，返回UserDTO对象给浏览器。

1. 好友关注
2. 好友关注和取消关注

1.1、按钮需要判断当前用户是否关注来显示对应的字样提示。

1.2、获取当前登录用户的id，将其作为key，判断当前传入的参数条件是否已经为关注，如果未关注，则继续关注操作—new一个Follow对象，存储当前的用户id和被关注的用户id，数据库保存成功后在redis中set数据库存储。若已经关注，则为取关操作，删除数据库中对应的内容，成功后删除redis中对应的内容。

2、共同关注

获取当前的登录用户id和被查看的用户id，分别作为key在redis中查看对应的关注对象集合，判断是否有交集（intersect），无交集则直接返回，若有交集则解析交集ids集合，根据ids在数据库中查找用户信息（UserDTO）并返回。

1. 实现分页查询关注推送（Feed流，此处用推模式，用redis中的sortedSet数据结构来完成）

3.1、推送到粉丝邮箱，每次某个用户上传了新的笔记之后，查询笔记作者的所有粉丝，for循环获取粉丝id，将其作为key，将笔记id作为值，存储在redis的sortedSet中，并设置当前的时间戳（需要按照时间戳进行排序）。

3.2、分页查询关注推送

获取当前的登录用户id—根据id作为key查询redis中的笔记（reverseRangeByScoreWithScores(key, 0, max, offset, 2)，key，最小值一直为0，最大值为上次查询到的最小的时间戳，offset为传入的偏移量（第一次查询时为0，以后都为根据情况而定），每页显示数）。--判断取出数据是否为空，不是则继续操作。--解析数据中的blogId、时间戳和偏移量。--根据blogId查询blog+查询blog有关的用户+查询blog是否被点赞。将查到的blog信息+偏移量+最大时间戳进行封装并返回。

1. 附近商户（使用redis 的GEO）
2. 在测试类中将数据库中的商户信息通过type\_id分类存储到redis中（包含shopId和经纬度坐标，经纬度坐标只是用来在redis中计算距离方便最后封装返回）。
3. 是否有传入中心坐标判断是否需要根据坐标排序查询商户
4. ----不需要坐标查询则根据数据库进行查询
5. ----计算分页参数（from和end）
6. ---- 根据分页参数，通过GEO的search查询出redis中对应key（typeId）的所有商铺信息（包括商铺id和坐标）
7. ----for循环中分别解析出商户id和距离，距离对应商户id存储，商户id再另外存储。
8. ----根据单独存储的商户id查询数据中的商户信息，得到商户信息的对象，再次循环将对应商户id中存储的距离存储在对应商户id的对象中。最总返回所有shop对象shops给浏览器。



1. 用户签到
2. 签到功能（bitMap）

----拼接key，（准备好前缀+获取当前的用户id，获取当前日期的年份和月份）

----获取今天是本月的第几天

在对应的key中的二进制中对应第几位的值设置为1（签到）/0（未签到）

1. 签到统计‘

----拼接key，（准备好前缀+获取当前的用户id，获取当前日期的年份和月份）

----通过 key获取（bitField）本月截止今天为止的所有签到记录（返回一个十进制的数）

----判断是否获得签到结果，是就在while循环中用该十进制数上的从低到高的每一位分别跟1做与运算，知道某一位为0结束循环，同时统计低位到高位中1的首次连续次数（为当前本月已经连续签到的天数）。

### 七、UV统计（统计一天内总共来了多少用户量（同个用户多次访问算一个）（用redis中的HyperLogLog））