06、商品搜索--------------------

方案一：thymeleaf页面静态化

1、创建商品详情的thymeleaf模板

2、创建RabbitMQ消费者，收到消息后生成静态页面（D:/detail/26774635180.html）

3、搭建nginx服务器，返回静态页面

方案二：redis

a、redis缓存商品详情

1、先查询redis，如果有直接返回

2、再查询mysql，并把查询结果装到redis中再返回

b、如何保证redis不满？redis中都是热点商品？

设置商品的实现时间：86400

c、怎么保存商品信息（数据类型）？

ITEM\_INFO:123456:BASE

ITEM\_INFO:123456:DESC

ITEM\_INFO:123456:PARAM

缓存穿透

 缓存穿透是指缓存和数据库中都没有数据，而用户不断发起请求则这些请求会穿过缓存直接访问数据库，如发起为id为“-1”的数据或id为特别大不存在的数据。假如有恶意攻击，就可以利用这个漏洞，对数据库造成压力，甚至压垮数据库。

## 4.2.解决方案

缓存空对象：

当存储层不命中后，即使返回的空对象也将其缓存起来，同时会设置一个过期时间（避免控制占用更多的存储空间），之后再访问这个数据将会从缓存中获取，保护了后端数据源；

# 缓存击穿

缓存击穿，是指一个key非常热点，在不停的扛着大并发，大并发集中对这一个key不停进行访问，当这个key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库，就像在一个屏障上凿开了一个洞。

解决方案：

1. 设置热点数据永远不过期
2. 加分布式锁

1、如何释放锁？del

2、业务处理失败？expire

# 缓存雪崩

缓存雪崩，是指在某一个时间段，缓存集中过期失效。

## 6.2.解决方案：

缓存数据的过期时间设置随机，不同分类商品缓存不同周期或热门类目的商品缓存时间长一些

---------------------------08、单点登录------------------------------

一、单点登录介绍

1、什么是单点登录？

SSO英文全称Single Sign On，单点登录，登录一次处处可用

## 为什么要有单点登录系统

集群环境下会出现要求用户多次登录的情况。

解决方案：

1、配置tomcat集群(500)。配置tomcat Session复制。节点数不要超过5个。

2、可以使用Session服务器（sso系统），保存Session信息。需要模拟Session。

session和redis的共同特点：

1、kv形式存储

2、过期时间

单点登录系统是使用redis模拟Session，实现Session的统一管理。

2、思路

存(登录)：redis(token,user) cookie(token\_key,token)

取(查询用户)：cookie(token)--------->redis(user)

# 注册信息校验

# 对用户的注册信息(用户名与电话号码)做数据校验

# 1、查询条件根据参数动态生成：1、2分别代表username、phone

# 2、从tb\_user表中查询数据

# 3、判断查询结果，如果查询到数据返回false。

# 如果没有返回true。 否则反之

# .通过token查询用户信息

1、从url中取参数token

2、根据token查询redis

3、如果查询不到数据，前台删除cookie中的用户信息

4、如果查询到数据，说明用户已经登录需要重置key的过期时间

## 购物车功能

1. 添加购物车商品
2. 展示购物车列表页面
3. 修改购物车商品数量
4. 删除购物车商品

## 购物车设计

1. 用户未登录状态下：在不登陆的情况下把购物车信息写入cookie

优点：

1、不占用服务端存储空间

2、代码实现简单。

3、用户体验好

缺点：

1、cookie中保存的容量有限。最大4k

2、把购物车信息保存在cookie中，更换设备购物车信息不能同步。

1. 用户已登录状态下：把购物车信息保存到服务端的 Redis 中

优点：

1、更换设备购物车信息可以同步

缺点：

1、占用服务端存储空间

# 未登录状态操作购物车

## 2.1.业务逻辑：

1、从cookie中查询商品列表：Map<itemId,TbItem> 商品购买数量使用TbItem的num保存

购物车已存在则直接返回

购物车不存在则创建空的购物车并返回

2、添加商品到购物车：

如果购物车存在该商品，商品数量相加。

如果购物车不存在该商品，根据商品id查询商品信息并添加到购车列表

3、把购车商品列表写入cookie。

读写cookie可以使用CookieUtils工具类实现

# 购物车同步

## 业务逻辑

在用户未登录情况下往购物车添加商品，添加结束用户登录，此时需把cookie中的数据同步到redis：

a) 如果redis中有数据cookie中也有数据，需要做数据合并。相同商品数量相加，不同商品添加一个新商品。

b) 把cookie中的数据删除，避免下次重复同步

## 订单功能

1、在购物车页面点击“去结算”按钮跳转到订单确认页面

a) 展示商品列表

2、订单确认页面需要根据用户查询配送地址，展示订单确认页面之前，应该确认用户身份

a) 如果用户未登录或登录过期跳转到登录页面

d) 登录成功后再跳转到订单确认页面

3、提交订单

a) 生成订单

4、扣减库存

5、关闭超时订单

a) 定时扫描超时2天未付款的订单，并关闭订单

b) 加回库存

# 用户身份认证

在展示订单确认页面之前，需要对用户身份进行认证，要求用户必须登录。

## 功能分析

1、使用springmvc的拦截器拦截所有订单的请求

2、业务逻辑

a) 从cookie中取token。

b) 根据token调用sso服务查询用户信息。

d) 如果查不到用户信息则跳转到登录页面。

e) 查询到用户信息放行。

# 展示订单确认页面

## 3.1.功能分析

1、在购物车页面点击“去结算”按钮跳转到订单确认页面。

2、请求的url：/frontend/order/goSettlement

3、参数：ids，userId，token

4、查询redis中的购物车数据并返回给前端

5、配送地址列表，需要用户登录。需要根据用户id查询收货地址列表。静态数据。

6、支付方式。静态数据。

# 提交订单

## 3.1.如何生成订单号

订单号需求是什么？ 1、唯一 2、可读性高 a) 纯数字

如何生成： 1、时间戳 3、时间戳 +随机数 2、手机号 4、时间戳 +自增id a)可行（使用Redis的INCR命令完成）（初值：100544） 5、时间戳+用户id/手机号 6、时间戳+用户id +店铺id a)可行，适用于C2C、B2C平台

## 3.2.功能分析

1、保存订单信息

a、在订单确认页面点击“提交订单”按钮生成订单。

b、请求的url：/frontend/order/insertOrder

c、参数：订单、订单商品、订单物流

d、返回值：{"status":200,"msg":"OK","data":"订单ID"}

# 扣减库存

## 4.1.分析

因为提交订单成功后 要减少库存

将消息写入消息队列，业务逻辑以异步的方式运行，加快响应速度

# 关闭超时订单

## 5.1分析

1、定义job扫描订单表：

a、修改订单的状态为关闭状态、结束时间、关闭时间、修改时间为当前时间

b、把订单中商品的库存数量加回去

扫描条件：状态是未付款 并且 创建时间 <= 当前时间 – 2天 并且付款方式为在线支付的订单

2、 定义触发条件

a) 理论上需要实时触发（性能问题）

b) 1分钟触发一次 0 \*/1 \* \* \* ?

# quartz集群任务重复执行问题

## 6.1.分析

quartz 在集群的时候，任务会出现重复执行的情况：

使用redis分布式锁解决quartz 集群任务重复执行的问题

拷贝usian\_order\_service，测试是否存在重复关闭订单的问题