品优购项目面试问题 200 问

导读

各位朋友大家好，当你翻开这个文档时，相信你和我一样作为一个追求理想和进步的青年，我们一起在传智播客.黑马程序员见证奇迹，实现高薪梦想。经过不断的修练，我们进阶到电商项目，这一步将要求我们实现技术层面质的飞跃，通过代码量的积累，新的架构和开发方式，常见电商项目架构解决方案，分布式开发及存储解决方案，相信通过这些技术的沉淀，我们已经树立了强烈的自信心，思想将像决堤的洪水，前面的一切困难和阻挠将变得不懈一击。

在传智播客的学习是一种快乐，也是一种人生成长的特殊经历，我们痛并快乐着。技术让我们开心，让我们充实，让我们改变思维改变命运，但朋友们同时也要记住“这个时代的技术变革重来没有停止，尤其是信息技术每天都在变化，我们要抱着活到老学到老的态度去对待”，传智播客.黑马程序员将持续跟进新技术变革，不断推陈出新。

后面我们一起开启传智播客.黑马程序员电商项目新篇章。。。

感谢北京顺义校区 JavaEE 面授团队的讲师及就业指导的辛苦工作。。。

第1章 SSM 框架面试问题

1.1讲下 springmvc 框架的工作流程

1、用户向服务器发送请求，请求被 SpringMVC 的前端控制器 DispatcherServlet 截获。
2、DispatcherServlet 对请求的 URL（统一资源定位符）进行解析，得到 URI(请求资源标识符)，然后根据该 URI，调用 HandlerMapping 获得该 Handler 配置的所有相关的对象，包括 Handler 对象以及 Handler 对象对应的拦截器，这些对象都会被封装到一个 HandlerExecutionChain 对象当中返回。
3、DispatcherServlet 根据获得的 Handler，选择一个合适的 HandlerAdapter。HandlerAdapter 的设计

符合面向对象中的单一职责原则，代码结构清晰，便于维护，最为重要的是，代码的可复制性高。HandlerAdapter 会被用于处理多种 Handler，调用 Handler 实际处理请求的方法。

4、提取请求中的模型数据，开始执行 Handler(Controller)。在填充 Handler 的入参过程中，根据配置，spring 将帮助做一些额外的工作

消息转换：将请求的消息，如 json、xml 等数据转换成一个对象，将对象转换为指定的响应信息。

数据转换：对请求消息进行数据转换，如 String 转换成 Integer、Double 等。

数据格式化：对请求的消息进行数据格式化，如将字符串转换为格式化数字或格式化日期等。

数据验证：验证数据的有效性如长度、格式等，验证结果存储到 BindingResult 或 Error 中。

5、Handler 执行完成后，向 DispatcherServlet 返回一个 ModelAndView 对象，ModelAndView 对象中应该包含视图名或视图模型。

6、根据返回的 ModelAndView 对象，选择一个合适的 ViewResolver(视图解析器)返回给 DispatcherServlet。

7、ViewResolver 结合 Model 和 View 来渲染视图。

8、将视图渲染结果返回给客户端。

以上 8 个步骤，DispatcherServlet、HandlerMapping、HandlerAdapter 和 ViewResolver 等对象协同工作，完成 SpringMVC 请求—>响应的整个工作流程，这些对象完成的工作对于开发者来说都是不可见的，开发者并不需要关心这些对象是如何工作的，开发者，只需要在 Handler(Controller)当中完成对请求的业务处理。

1.2图片怎么上传

前端使用 angularJS 异步上传，后端使用 springmvc 的 MultipartFile 类型来接收，放到分布式图片服务器中，服务器返回图片路径把路径返回页面回显图片

1.3微服务和 SOA 有什么区别？

如果一句话来谈 SOA 和微服务的区别，即微服务不再强调传统 SOA 架构里面比较重的 ESB 企业服务总线，同时 SOA 的思想进入到单个业务系统内部实现真正的组件化。

说的更直白一点就是微服务被拆分的粒度更小

1.4spring 框架 AOP 执行原理简单说下？还有就是 AOP 在事务管理方面是怎么实现的？

Spring AOP 使用的动态代理，所谓的动态代理就是说 AOP 框架不会去修改字节码，而是在内存中临时为方法生成一个 AOP 对象，这个 AOP 对象包含了目标对象的全部方法，并且在特定的切点做了

增强处理，并回调原对象的方法。

Spring AOP 中的动态代理主要有两种方式，JDK 动态代理和 CGLIB 动态代理。JDK 动态代理通过反射来接收被代理的类，并且要求被代理的类必须实现一个接口。JDK 动态代理的核心是 InvocationHandler 接口和 Proxy 类。如果目标类没有实现接口，那么 Spring AOP 会选择使用 CGLIB 来动态代理目标类。CGLIB（Code Generation Library），是一个代码生成的类库，可以在运行时动态的生成某个类的子类，注意，CGLIB 是通过继承的方式做的动态代理，因此如果某个类被标记为 final，那么它是无法使用 CGLIB 做动态代理的。

AOP 在事务管理方面，Spring 使用 AOP 来完成声明式的事务管理有 annotation 和 xml 两种形式。开发中，方便代码编写，很多时候都是在 spring 配置文件中配置事务管理器并开启事务控制注解。在业务类或业务类方法中添加@Transactional 实现事务控制。

1.5Spring 分布式事务如何处理的

回答：

第一种方案：可靠消息最终一致性，需要业务系统结合 MQ 消息中间件实现，在实现过程中需要保证消息的成功发送及成功消费。即需要通过业务系统控制 MQ 的消息状态

第二种方案：TCC 补偿性，分为三个阶段 TRYING-CONFIRMING-CANCELING。每个阶段做不同的处理。

TRYING 阶段主要是对业务系统进行检测及资源预留

CONFIRMING 阶段是做业务提交，通过 TRYING 阶段执行成功后，再执行该阶段。默认如果 TRYING 阶段执行成功，CONFIRMING 就一定能成功。

CANCELING 阶段是回对业务做回滚，在 TRYING 阶段中，如果存在分支事务 TRYING 失败，则需要调用 CANCELING 将已预留的资源进行释放。

1.6Springboot 用过没，跟我说说，他的特点？

Springboot 是从无数企业实战开发中总结出来的一个更加精炼的框架，使得开发更加简单，能使用寥寥数行代码，完成一系列任务。

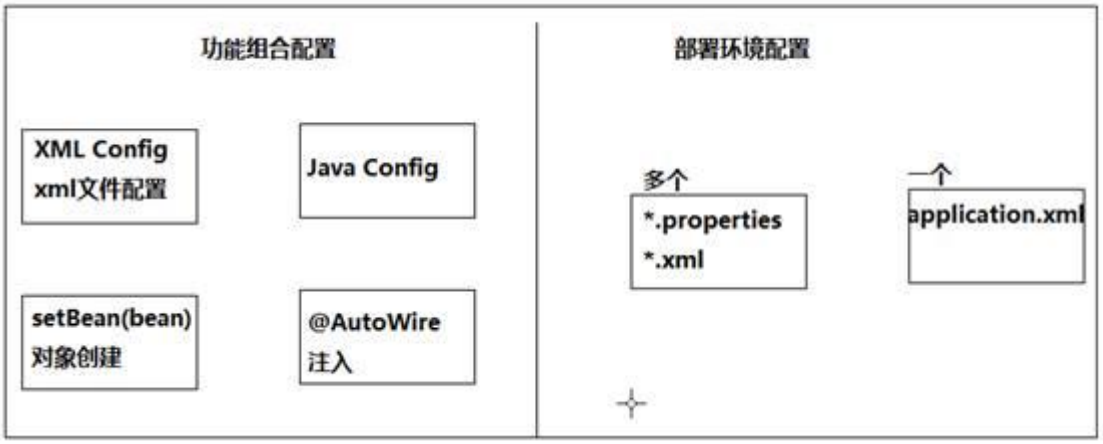
1） Springboot 解决那些问题

a) 编码更简单

i. Spring 框架由于超重量级的 XML，annotation 配置，使得系统变得很笨重，难以维护

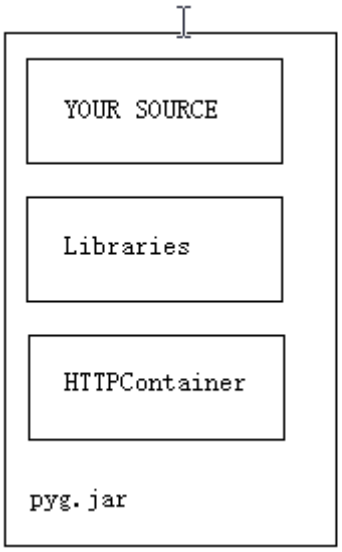
ii. Springboot 采用约点大于配置的方法，直接引入依赖，即可实现代码的开发

b) 配置更简单



Xml 文件使用 javaConfig 代替，XML 中 bean 的创建，使用@bean 代替后可以直接注入。
配置文件变少很多，就是 application.yml

c) 部署更简单



一键启动
一键解压
运行：java -jar pyg.jar

不需要不是应用服务器：
Tomcat（X）
weblogic(X)

降低对运行环境基本要求：
部署环境只需要有JDK即可
默认内置Tomcat服务器

d) 监控更简单

Spring-boot-start-actuator:
可以查看属性配置
线程工作状态
环境变量
JVM 性能监控

第2章 支付面试问题

2.1支付接口是怎么做的？

调用微信的支付接口，参考微信提供的 api

使用了微信的统一下单接口和查询支付状态接口

每个接口需要的参数放入到 map 中使用微信提供的 sdk 转成 XML 字符串，httpClient 远程提交参数和接收结果。

第3章 项目业务面试问题

3.1哪些情况用到 activeMq

商品上架后更新 solr 索引库、更新静态页、发送短信

3.2秒杀的时候，只有最后一件物品，该怎么去抢或者分配？

秒杀商品的库存都会放到 redis 中，在客户下单时就减库存，减完库存会判断库存是否为大于 0,如果小于 0,表示库存不足,刚才减去的数量再恢复,整个过程使用 redis 的 watch 锁

3.3 solr 和 lucene 的区别

solr 是 Apache 的用于实现全文检索的开源项目，是一个 war 包，直接可以放入 tomcat 服务器中配置一下就可以使用，而 Lucene 是全文检索的底层技术，solr 是在 Lucene 的基础上开发的 solr

3.4solr 怎么设置搜索结果排名靠前（得分）？

设置 boots 值

3.5 你项目对于订单是怎么处理的，假如一个客户在下订单的时候没有购买怎么办，对于顾客在购买商品的时候你们怎么处理你们的库存？

订单表中设置了一个过期时间，每天会有定时任务来扫描订单表数据，如果到达预订的过期时间没有付款就会取消此订单交易。

关于库存的设计是这样的：

普通商品在发货时才去更新库存，如果库存不足商家会马上补货

秒杀的商品会在客户下单时就减库存，如果在规定时间（半个小时）没有付款，会取消此订单把库存还原

3.6插入商品的话，要求级联插入几张表，你们当时是怎么实现的？

三张表 商品表、商品描述表、sku 表，他们的关系是一对一对多，三张表使用了组合类表现他们的关系，在页面使用 angularjs 双向绑定了组合类

3.7redis 存储格式的选择

redis 支持的数据结构总共有 5 种：hash、value、list、set、zset，其中项目中用到最多是 hash

3.8 商品表中的数据是哪里来的不知道

商品表的数据是在商家管理后台中由商家录入的。数据分别录入到商品表、商品描述表和商品项表

3.9 当初设计项目时预计的访问量计划是多少

访问量计划是 3000 至 5000

3.10 店主申请开店是从哪里申请的，所需要的信息都有哪些

商家申请开店在商家管理后台申请入驻的。在运营商管理后台审核通过后，方可登陆系统完成后续操作，例如：录入商品信息等。所需要的信息可以参考：注册页面学习。需要信息有：商家登录名、密码、店铺名称、公司名称、公司电话、联系人、联系人电话、营业执照号、税务登记号、组织机构代码证、法人代表信息、开户行信息等。

3.11 模板表的设计思想

模板表设计主要是为了商品表与品牌表、规格表进行数据关联。方便商品录入时，选择对应的品牌和规格数据。

3.12 简单介绍一下你的这个项目以及项目中涉及到的技术框架以及使用场景以及你主要负责项目中的哪一块？

项目介绍时，先整体介绍是什么项目，项目主要是做啥的。例如：XXX 电商项目，是一个 B2B2C 综合电商平台。由三个系统组成，包含：运营商管理后台、商家管理后台、网站前台。运营商平台主要负责基础数据维护、商家审核、商品审核等。商家管理后台主要负责商家入驻、商品录入/修改、商品上下架等。网站前台主要负责商品销售。包含：网站首页、商品搜索、商品详情展示、购物车、订单、支付、用户中心等模块。

再介绍自己在项目中做的功能模块。例如：运营商管理后台的品牌、规格数据录入，已经商品管理后台商品录入功能。同时，实现了网站前台系统中的商品搜索、购物车等功能模块。

然后介绍里面使用的技术：例如：dubbo 分布式框架、ssm、angularjs、solr、redis、

freemarker、activeMQ、微信扫码支付等等。最好是结合技术讲解项目功能点如何实现。

3.13 秒杀系统中如何防止超售？如何避免脚本进行恶意刷单？

防止超售解决方案：将存库从 MySQL 前移到 Redis 中，所有的写操作放到内存中，由于 Redis 中不存在锁故不会出现互相等待，并且由于 Redis 的写性能和读性能都远高于 MySQL，这就解决了高并发下的性能问题。然后通过队列等异步手段，将变化的数据异步写入到 DB 中。当达到库存阀值的时候就不在消费队列，并关闭购买功能。
避免脚本恶意刷单：采用 IP 级别的限流，即针对某一个 IP，限制单位时间内发起请求数量。

3.14 单点登录你们是自己编写的还是使用通用的 CAS?

项目使用通用的 CAS 框架。

3.15 如果一个用户的 token 被其他用户劫持了，怎样解决这个安全问题。

a、在存储的时候把 token 进行对称加密存储，用时解开。

b、将请求 URL、时间戳、token 三者进行合并加盐签名，服务端校验有效性。

c.HTTPS 对 URL 进行加密

3.16 项目部署上线后，运营商管理，商品审核等后台流量问题？

回答：
先询问流量是指哪方面？流量分为三种，一种是商家流量，另一种是用户流量，第三种运营商流量。

解决方案：

这三种流量对系统运行造成很大压力，随着项目上线时间增长，压力会越来越大，因此我们要减轻系统访问压力 ，就需要做一系列优化措施。

具体优化如下：

数据层面的优化：

从数据库层面做优化，比如:索引，缓存，集群，读写分离，主从复制，分表，分库。

从数据库设计层面的优化：比如减少表关联，加入冗余字段

从缓存方面优化：比如 redis 实现数据缓存，减轻数据库压力

从搜索上进行优化：比如查找索引库

项目层面的优化：

采用面向服务的分布式架构：分担服务器压力 ，提高项目并发量

比如 dubbox+zookeeper 分布式架构

采用分布式文件系统实现海量文件存储：如采用 fastdfs 实现海量图片存储，提高文件的访问速度。

采用 mq 使用服务进一步解藕：同步索引库，同步静态资源，短信发送

服务器层面的优化

集群思想的使用：tomcat,zookeeper,redis,mysql 等

Tomcat 异步通信的使用，tomcat 连接池配置

3.17 秒杀和团购业务实现思路

回答：

将商品数量查询出存入到 redis 中，所有用户下单后，减掉 redis 中的数量

如果并发量很大时，还要考虑高并发问题，所以可以加入 mq 消息中间件处理抢单问题，再结合 redis 实现库存减少操作。高并发方面还可以考虑 CDN，Nginx 负载均衡等

3.18 你们项目中使用的安全框架是什么？

使用 springSecurity，校验用户登录和用户权限！

3.19 项目中使用到的应用服务器是什么？

Tomcat+nginx

3.20 讲一下每台服务器的集群数量：

项目中一共 15 台项目服务，那么为了每一台高可用一主一备，但首页项目高并发设为四台服务器，则一共 32 台项目服务器，再加 redis 集群用了 3 台，为了每一台高可用一主一备一共 6 台，fastdfs 一个 trackerServer 一个 storageServer 搭建集群一共 6 台，solr 集群 7 台服务器，nginx 为了高可用一主一备一共 2 台，mysql 数据库集群 3 台！activemq 消息中间件高可用 2 台；

共计：58 台服务器！

1：你在项目开发中碰到过哪些重大且棘手的问题

场景一：需求不明确困境

在项目开发中，项目采用迭代开发，开发需求不是很明确，对于项目开发初期来说非常困难，进度非常慢，有时开发的出的产品结果往往不能令老板满意，或者是甲方满意，项目还需要不停的迭代，修改。

比如说：

在开发品优购项目的时候，客户定位是一个综合性的商务平台，可以实现在线第三方商家对接，实现商品的销售

但是并没有明确的需求，因此开发全凭借电商的项目经验来实现里面的相关的业务，后期慢慢迭代。

场景二：使用 cas 单独登录不能很好实现想要的效果

场景三： solr 高亮不能显示的问题

前台使用 angularJS 加载搜索结果，但是发现高亮不能展示。

问题原因：

angularJS 底层使用 ajax，异步加载高亮信息返回给页面后，页面没有刷新，就直接显示返回的数据。此时会把所有的数据作为普通的文本数据进行加载。因此就没有高亮的效果。

解决方案：

使用 angularJS 过滤器过滤文本数据 ,此时 angularJS 过滤器把 html 文本数据解析为浏览器能识别的 html 标签。高亮就能展示了。

场景四：Nginx 静态页面服务跳转到购物车跨域问题

在 Nginx 中部署了静态页面，添加购物车时必须从静态页面跳转到购物车系统,实现购物车添加操作。

由于在静态页面中使用 angularJS 实现的跳转，发现跳转到购物车系统完全没有问题，但是并不能跳转回到购物车系统页面。

问题分析：

从静态详情系统跳转到购物车系统，会存在跨域问题，因此不能进行回调函数的数据传递。所以在回调函数中的页面跳转就不能实现。

解决方案：

使用 angularJS 跨域调用及 springmvc 跨域配置，解决问题。

场景五：activeMQ 存在运行时间长了以后，收不到消息的现象

时间长了就会出现，卡死，新的数据不能从队列接听到。只能重启程序。

解决方案：

1）不要频繁的建立和关闭连接

JMS 使用长连接方式，一个程序，只要和 JMS 服务器保持一个连接就可以了，不要频繁的建立和关闭连接。频繁的建立和关闭连接，对程序的性能影响还是很大的。这一点和 jdbc 还是不太一样的。

2）Connection 的 start()和 stop()方法代价很高

JMS 的 Connection 的 start()和 stop()方法代价很高，不能经常调用。我们试用的时候，写了个 jms 的 connection pool，每次将 connection 取出 pool 时调用 start()方法，归还时调用 stop()方法，然而后来用 jprofiler 发现，一般的 cpu 时间都耗在了这两个方法上。

3）start()后才能收消息

Connection 的 start()方法调用后，才能收到 jms 消息。如果不调用这个方法，能发出消息，但是一直收不到消息。不知道其它的 jms 服务器也是这样。

4）显式关闭 Session

如果忘记了最后关闭 Connection 或 Session 对象，都会导致内存泄漏。这个在我测试的时候也发现了。本来以为关闭了 Connection，由这个 Connection 生成的 Session 也会被自动关闭，结果并非如此，Session 并没有关闭，导致内存泄漏。所以一定要显式的关闭 Connection 和 Session。

5）对 Session 做对象池

对 Session 做对象池，而不是 Connection。Session 也是昂贵的对象，每次使用都新建和关闭，代价也非常高。而且后来我们发现，原来 Connection 是线程安全的，而 Session 不是，所以后来改成了对 Session 做对象池，而只保留一个 Connection。

6） 集群

ActiveMQ 有强大而灵活的集群功能，但是使用起来还是会有很多陷阱

场景六：activeMQ 存在发生消息太大，造成消息接受不成功

多个线程从 activeMQ 中取消息，随着业务的扩大，该机器占用的网络带宽越来越高。

仔细分析发现，mq 入队时并没有异常高的网络流量，仅仅在出队时会产生很高的网络流量。

最终发现是 spring 的 jmsTemplate 与 activemq 的 prefetch 机制配合导致的问题。

研究源码发现 jmsTemplate 实现机制是：每次调用 receive()时都会创建一个新的 consumer 对象，用完即销毁。

正常情况下仅仅会浪费重复创建 consumer 的资源代价，并不至于产生正常情况十倍百倍的网络流量。

但是 activeMQ 有一个提高性能的机制 prefetch，此时就会有严重的问题。

prefetch 机制：
每次 consumer 连接至 MQ 时，MQ 预先存放许多 message 到消费者（前提是 MQ 中存在大量消息），预先存 放 message 的数量取决于 prefetchSize（默认为 1000）。此机制的目的很显然，是想让客户端代码用一个 consumer 反复进行 receive 操作，这样能够大量提高出队性能。

此机制与 jmsTemplate 配合时就会产生严重的问题，每次 jmsTemplate.receive()，都会产生 1000 个消息的网络流量， 但是因为 jmsTemplae 并不会重用 consumer，导致后面 999 个消息都被废弃。反复 jmsTemplate.receive()时，表面上看 不出任何问题，其实网络带宽会造成大量的浪费。

解决方案：

1、若坚持使用 jmsTemplate，需要设置 prefetch 值为 1，相当于禁用了 activeMQ 的 prefetch 机制，此时感觉最健壮， 就算多线程，反复调用 jmsTemplate.receive()也不会有任何问题。但是会有资源浪费，因为要反复创建 consumer 并频繁与服务器进 行数据通信，但在性能要求不高的应用中也不算什么问题。

2、不使用 jmsTemplate，手工创建一个 consumer，并单线程反复使用它来 receive()，此时可以充分利用 prefetch 机制。配合多线程的方式每个线程拥有自己的一个 consumer，此时能够充分发挥 MQ 在大吞吐量时的速度优势。

切记避免多线程使用一个 consumer 造成的消息混乱。大吞吐量的应用推荐使用方案 2，能够充分利用 prefetch 机制提高系 MQ 的吞吐性能。

3.21 商品的价格变化后，如何同步 redis 中数以百万计的购物车数据。

解决方案：

购物车只存储商品 id,到购物车结算页面将会从新查询购物车数据，因此就不会涉及购物车商品价格同步的问题。

3.22 系统中的钱是如何保证安全的。

在当前互联网系统中钱的安全是头等大事，如何保证钱的安全可以从以下 2 个方面来思考：

1）钱计算方面

在系统中必须是浮点数计算类型存储钱的额度，否则计算机在计算时可能会损失精度。

2）事务处理方面

在当前环境下，高并发访问，多线程，多核心处理下，很容易出现数据一致性问题，此时必须使用事务进行控制，访问交易出现安全性的问题，那么在分布式系统中，存在分布式事务问题，可以有很多解决方案：

使用 jpa 可以解决

使用 tcc 框架可以解决等等。

3.23 订单中的事物是如何保证一致性的。

使用分布式事务来进行控制，保证数据最终结果的一致性。

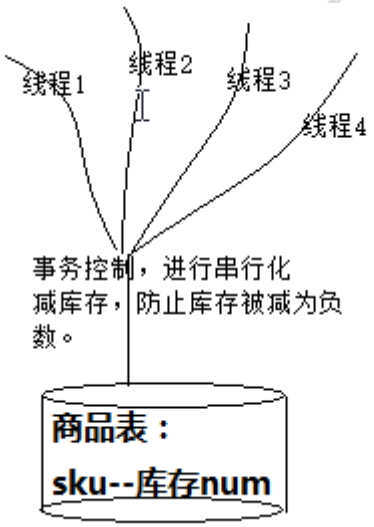
3.24 当商品库存数量不足时，如何保证不会超卖。

当库存数量不足时，必须保证库存不能被减为负数，如果不加以控制，库存被减为小于等于 0 的数，那么这就叫做超卖。

那么如何防止超卖的现象发生呢?

场景一： 如果系统并发要求不是很高

那么此时库存就可以存储在数据库中，数据库中加锁控制库存的超卖现象。



场景二：系统的并发量很大

如果系统并发量很大，那么就不能再使用数据库来进行减库存操作了，因为数据库加锁操作本身是以损失数据库的性能来进行控制数据库数据的一致性的。

但是当并发量很大的时候，将会导致数据库排队，发生阻塞。因此必须使用一个高效的 nosql 数据库服务器来进行减库存。

此时可以使用 redis 服务器来存储库存，redis 是一个内存版的数据库，查询效率相当的高，可以使用 watch 来监控减库存的操作，一旦发现库存被减为 0，立马停止售卖操作。

3.25 你们系统的商品模块和订单模块的数据库是怎么设计的

商品模块设计：

商品模块一共 8 张表，整个核心就是模板表。采用模板表为核心的设计方法，来构造商品数据。

订单设计：

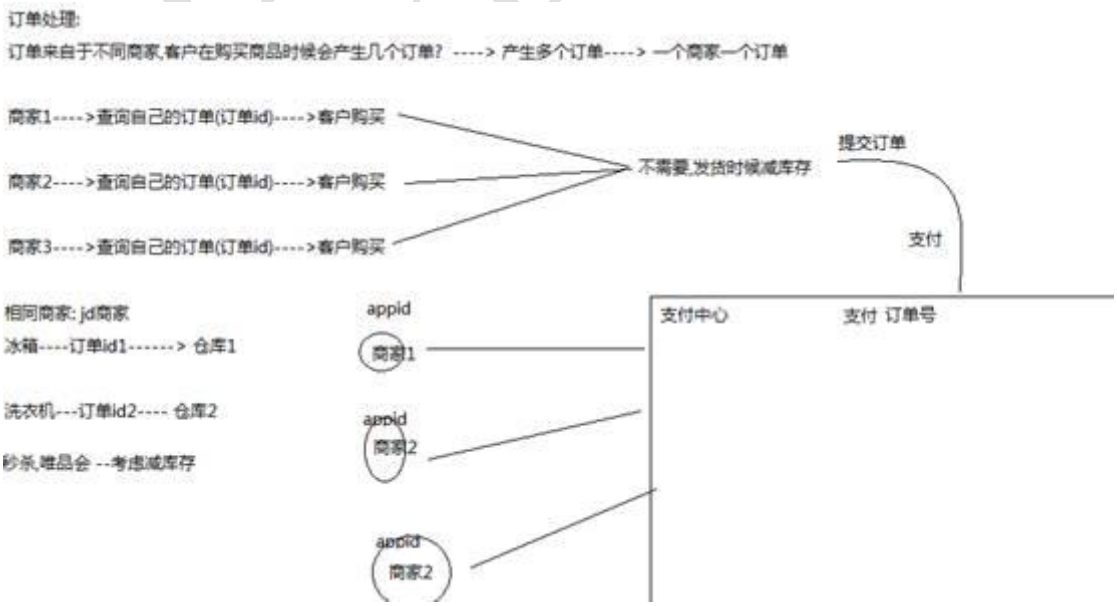
订单涉及的表有：

1） 收货人地址

2） 订单信息

3） 订单明细

订单关系描述：



3.26 系统中商家活动策划以及上报相关业务流程。

品优购系统中有以下活动：

1） 秒杀活动

a) 后台设置秒杀商品

b) 设置秒杀开启时间，定时任务，开启秒杀

c) 秒杀减库存（秒杀时间结束，库存卖完，活动结束）

2） 促销活动

3） 团购活动

4） 今日推荐

以上活动销售记录，统计，使用图形化报表进行统计，可以查看销售情况。

11，涉及到积分积累和兑换商品等业务是怎么设计的

积分累计有 2 大块：

积分累计：

根据用户购买的商品的价格不同，没有购买一定价格的商品，获取一定的积分。

积分商城：

积分商城是用户可以使用积分商品换取商品的区域。

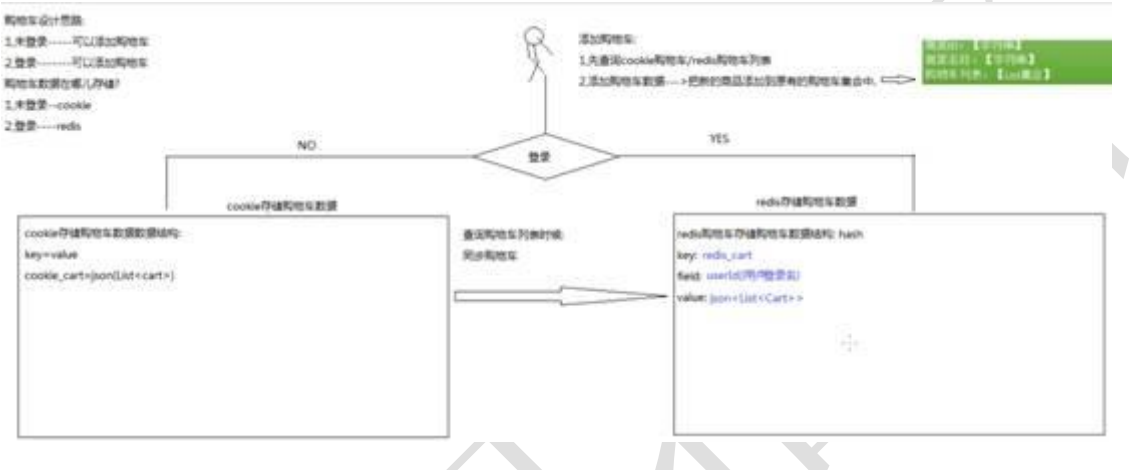
3.27 介绍下电商项目，你觉得那些是亮点？

这个项目是为 xxx 开发的 b2b2c 类型综合购物平台，主要以销售家用电器，电子产品为主要的电子商城网站。
项目的亮点是：
1） 项目采用面向服务分布式架构（使用 dubbo,zookeeper）
a) 解耦
b) 提高项目并发能力
c) 分担服务器压力
2） 项目中使用 activeMQ 对项目进一步解耦
a) 提高项目并发能力
b) 提高任务处理速度

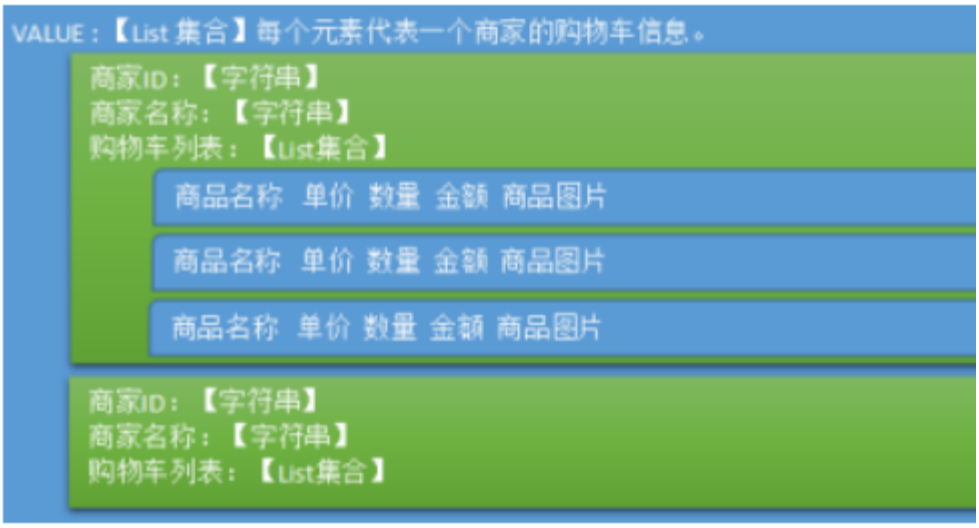
3） 使用微信支付，支付宝支付（自己总结）
4） 使用阿里大鱼发生短信
5） 使用第三方分布式文件系统存储海量文件
6） Nginx 部署静态页面实现动静分离

3.28 购物车功能做了吗，实现原理说一下？

购物车设计示意图：



购买商品: 商品可能来自于不同的商家? 商家数据购物车该如何表示? List<Cart>



一个用户可以购买很多商家的商品,因此使用list集合对商家进行封装.

一个商家对应一个商家对象 Cart.

List<cart>

1,未登录可以添加购物车,A电脑未登录添加购物车数据,然后在B能否看见,如果看不见,如何解决?

答: B电脑看不见购物车数据,此购物车数据此时存储A电脑本地磁盘的.

如何解决:

A电脑收藏商品后,必须先登录,然后再查询时候同步购物车数据,把cookie购物车数据同步服务器端redis服务器.

2,如果cookie被禁用,能否添加购物车?

cookie被禁用,首先不能使用cookie添加购物车

cookie被禁用,也不能登录,登录状态也不能添加购物车,因此cookie被禁用,不能实现购物车添加.

3,同步购物车在何时进行同步?

查询购物车列表时候(下一次操作,结算)合并购物车.

购物车添加业务流程:

1,查询购物车列表

1) 判断用户是否处于登录状态

2) 如果未登录---->查询cookie购物车列表数据

3) 如果登录------>查询redis购物车数据

2,添加购物车

1) 构造购物车数据结构:List<Cart>

> 判断是否有相同商家(购买的商品是否来自此商家)

i,如果有相同的商家

* 判断是否有相同的商品

* 有相同的商品--商品数量相加,商品总价重新计算

* 没有相同商品--直接添加即可

ii,如果没有商家--此时购买商品来自的新的商家

* 新建一个cart对象(新建一个商家),直接添加购物车列表即可

2) 判断用户是否处于登录状态

3) 如果未登录---把List<Cart>添加到cookie购物车即可

4) 登录,----把List<Cart>添加redis数据库.

3.29 你们的项目上线了吗？这么大的项目怎么没上线？

项目上线问题回答：

1） 项目没有上线

可以是项目没有上线之前，你离职了，这个一个创业型的公司，或者此项目是给甲方做的项目，你没有参与上线。以此来回避这个问题

2） 项目上线

项目已经上线了

上线环境：

a) Centos7
b) Mysql
c) Jdk8
d) Tomcat8

关于上线，那么面试官一定会问您，上线遇到什么问题没有？
因此必须把项目中遇到的问题准备 2 个。

3.30 订单怎么实现的，你们这个功能怎么这么简单？

订单实现：
从购物车系统跳转到订单页面，选择默认收货地址
选择支付方式
购物清单展示
提交订单
订单业务处理：
一个商家一个订单，不同的仓库发送的货品也是属于不同的订单。因此会产出不同的订单号。

订单处理：根据支付的状态进行不同的处理
1） 在线支付

a) 支付未成功—从新发起支付
b) 支付超时---订单关闭

2) 货到付款

3.31 你们这个项目有秒杀吗，怎么实现的？

所谓“秒杀”，就是网络卖家发布一些超低价格的商品，

所有买家在同一时间网上抢购的一种销售方式。通俗一点讲就是网络商家为促销等目的组织的网上限时抢购活动。由于商品价格低廉，往往一上架就被抢购一空，有时只用一秒钟。

秒杀商品通常有两种限制：库存限制、时间限制。

需求：

（1）商家提交秒杀商品申请，录入秒杀商品数据，主要包括：商品标题、原价、秒杀价、商品图片、介绍等信息

（2）运营商审核秒杀申请

（3）秒杀频道首页列出秒杀商品（进行中的）点击秒杀商品图片跳转到秒杀商品详细页。

（4）商品详细页显示秒杀商品信息，点击立即抢购实现秒杀下单，下单时扣减库存。当库存为 0 或不在活动期范围内时无法秒杀。

（5）秒杀下单成功，直接跳转到支付页面（微信扫码），支付成功，跳转到成功页，填写收货地址、电话、收件人等信息，完成订单。

（6）当用户秒杀下单 5 分钟内未支付，取消预订单，调用微信支付的关闭订单接口，恢复库存。

数据库表分析

Tb_seckill_goods 秒杀商品表

列名	数据类型	长度	默认	主键?	非空?	Unsigned	自增?	Zerofill?	注释
id	bigint	20		☑	☑	☐	☑	☐	
goods_id	bigint	20		☐	☐	☐	☐	☐	spu ID
item_id	bigint	20		☐	☐	☐	☐	☐	sku ID
title	varchar	100		☐	☐	☐	☐	☐	标题
small_pic	varchar	150		☐	☐	☐	☐	☐	商品图片
price	decimal	10,2		☐	☐	☐	☐	☐	原价格
cost_price	decimal	10,2		☐	☐	☐	☐	☐	秒杀价格
seller_id	varchar	100		☐	☐	☐	☐	☐	商家ID
create_time	datetime			☐	☐	☐	☐	☐	添加日期
check_time	datetime			☐	☐	☐	☐	☐	审核日期
status	varchar	1		☐	☐	☐	☐	☐	审核状态
start_time	datetime			☐	☐	☐	☐	☐	开始时间
end_time	datetime			☐	☐	☐	☐	☐	结束时间
num	int	11		☐	☐	☐	☐	☐	秒杀商品数
stock_count	int	11		☐	☐	☐	☐	☐	剩余库存数
introduction	varchar	2000		☐	☐	☐	☐	☐	描述

Tb_seckill_order 秒杀订单表

列名	数据类型	长度	默认	主键?	非空?	Unsigned	自增?	Zerofill?	注释
id	bigint	20		☑	☑	☐	☐	☐	主键
seckill_id	bigint	20		☐	☐	☐	☐	☐	秒杀商品ID
money	decimal	10,2		☐	☐	☐	☐	☐	支付金额
user_id	varchar	50		☐	☐	☐	☐	☐	用户
seller_id	varchar	50		☐	☐	☐	☐	☐	商家
create_time	datetime			☐	☐	☐	☐	☐	创建时间
pay_time	datetime			☐	☐	☐	☐	☐	支付时间
status	varchar	1		☐	☐	☐	☐	☐	状态
receiver_address	varchar	200		☐	☐	☐	☐	☐	收货人地址
receiver_mobile	varchar	20		☐	☐	☐	☐	☐	收货人电话
receiver	varchar	20		☐	☐	☐	☐	☐	收货人
transaction_id	varchar	30		☐	☐	☐	☐	☐	交易流水

秒杀实现思路

秒杀技术实现核心思想是运用缓存减少数据库瞬间的访问压力！读取商品详细信息时运用缓存，当用户点击抢购时减少 redis 中的库存数量，当库存数为 0 时或活动期结束时，同步到数据库。 产生的秒杀预订单也不会立刻写到数据库中，而是先写到缓存，当用户付款成功后再写入数据库。

3.32 你们这个项目用的什么数据库，数据库有多少张表？

项目使用 mysql 数据库，总共有 103 张表，其中商品表共计有 8 张。

3.33 项目部署做过吗，能不能部署？

做过，可以部署。

项目服务器：集群部署

数据库服务器：集群部署

Nginx 集群：负载均衡

3.34 单点登录怎么做的，用别人知道原理吗？

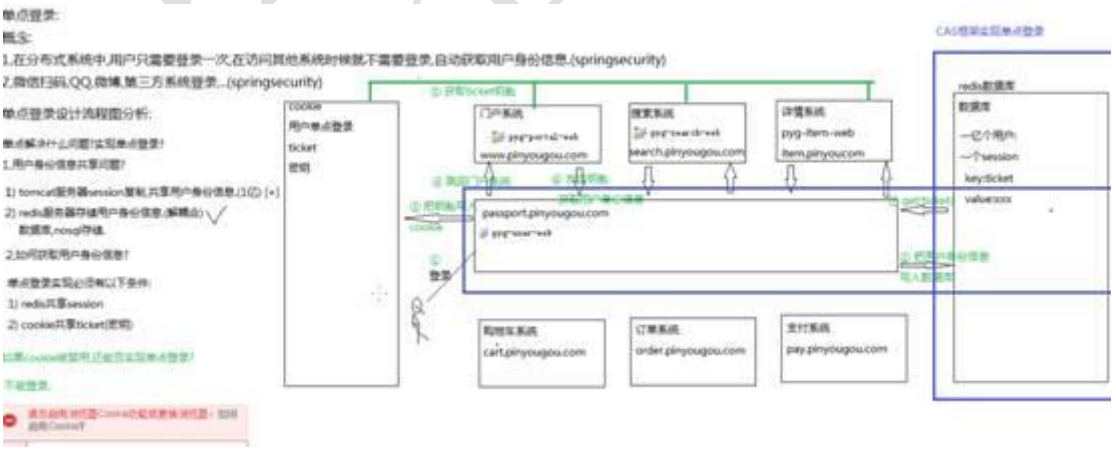
在分布式项目中实现 session 共享，完成分布式系统单点登录

3） Cookie 中共享 ticket

4） Redis 存储 session

分布式系统共享用户身份信息 session，必须先获取 ticket 票据，然后再根据票据信息获取 redis 中用户身份信息。

实现以上 2 点即可实现 session 共享。



目前项目中使用的 springsecurity + cas 来实现的单点登录，cas 自动产生 ticket 票据信息，每次获取用户信息，cas 将会携带 ticket 信息获取用户身份信息。

3.35 支付做了吗，支付宝还是微信，实现说下？

微信支付
微信支付：
1） 调用微信支付下单接口
2） 返回支付地址，生成二维码
3） 扫描二维码即可完成支付
问题： 微信支付二维码是我们自己生成的，因此必须时刻监控微信支付二维码的状态，确保支付成功。

第4章 缓存及优化方面的面试问题

4.1怎么提高 redis 缓存利用率？

1、从业务场景分析，预计会高频率用到的数据预先存放到 redis 中，
2、可以定时扫描命中率低的数据，可以直接从 redis 中清除。

4.2怎么实现数据量大、 并发量高的搜索

创建 solr 索引库，数据量特别大时采用 solr 分布式集群

4.3 怎么分词

使用第三方的分词器 IKAnalyzer，会按照中国人用此习惯自动分词。

4.4seo 怎么优化

使用 restful，或静态页这样能更好的被搜索引擎收录。

4.5怎么加快访问速度

硬件上加大网络带宽、和服务器内存

代码的处理：静态页面、缓存、优化 sql、创建索引等方案

4.6 讲到 redis 缓存的时候说不清楚

1、 redis 中项目中的应用。1.主要应用在门户网站首页广告信息的缓存。因为门户网站访问量较大，将广告缓存到 redis 中，可以降低数据库访问压力，提高查询性能。2.应用在用户注册验证码缓存。利用 redis 设置过期时间，当超过指定时间后，redis 清理验证码，使过期的验证码无效。3.用在购物车模块，用户登陆系统后，添加的购物车数据需要保存到 redis 缓存中。

2、 技术角度分析：

2.1 内存如果满了，采用 LRU 算法进行淘汰。

2.2Redis 如何实现负载的？采用 Hash 槽来运算存储值，使用 CRC16 算法取模运算，来保证负载问题。

2.3Redis 缓存穿透问题？将数据查询出来如果没有强制设置空值，并且设置过期时间，减少频繁查询数据库。

4.7 能讲下 redis 的具体使用场景吗？使用 redis 存储长期不改变的数据完全可以使用也看静态化，那么你们当时是为什么会使用 redis？

redis 在项目中应用：1.主要应用在门户网站首页广告信息的缓存。因为门户网站访

问量较大，将广告缓存到 redis 中，可以降低数据库访问压力，提高查询性能。2.应用在用户注册验证码缓存。利用 redis 设置过期时间，当超过指定时间后，redis 清理验证码，使过期的验证码无效。3.用在购物车模块，用户登陆系统后，添加的购物车数据需要保存到 redis 缓存中。

使用 redis 主要是减少系统数据库访问压力。从缓存中查询数据，也提高了查询性能，挺高用户体验度。

4.8redis 中对一个 key 进行自增或者自减操作，它是原子性的吗？

是原子性的。对于 Redis 而言，命令的原子性指的是：一个操作的不可以再分，操作要么执行，要么不执行。Redis 的操作之所以是原子性的，是因为 Redis 是单线程的。对 Redis 来说，执行 get、set 以及 eval 等 API，都是一个一个的任务，这些任务都会由 Redis 的线程去负责执行，任务要么执行成功，要么执行失败，这就是 Redis 的命令是原子性的原因。Redis 本身提供的所有 API 都是原子操作，Redis 中的事务其实是要保证批量操作的原子性。

4.9你们项目中使用到的数据库是什么？你有涉及到关于数据库到建库建表操作吗？数据库创建表的时候会有哪些考虑呢？

项目中使用的是 MySQL 数据库，
数据库创建表时要考虑
a、大数据字段最好剥离出单独的表，以便影响性能
b、使用 varchar，代替 char，这是因为 varchar 会动态分配长度，char 指定为 20，即时你存储字符“1”，它依然是 20 的长度
c、给表建立主键，看到好多表没主键，这在查询和索引定义上将有一定的影响
d、避免表字段运行为 null，如果不知道添加什么值，建议设置默认值，特别 int 类型，比如默认值为 0，在索引查询上，效率立显。
e、建立索引，聚集索引则意味着数据的物理存储顺序，最好在唯一的，非空的字段上建立，其它索引也不是越多越好，索引在查询上优势显著，在频繁更新数据的字段上建立聚集索引，后果很严重，插入更新相当忙。

f、组合索引和单索引的建立，要考虑查询实际和具体模式

4.10 mysql 中哪些情况下可以使用索引，哪些情况不能使用索引？mysql 索引失效的情形有哪些？

使用索引：

a、 为了快速查找匹配 WHERE 条件中涉及到列。

b、 如果表有一个 multiple-column 索引，任何一个索引的最左前缀可以通过使用优化器来查找行

c、 当运行 joins 时，为了从其他表检索行。MySql 可以更有效的使用索引在多列上如果他们声明的类型和大小是一样的话。在这个环境下，VARCHAR 和 CHAR 是一样的如果他们声明的大小是一样的

d、 为了找到 MIN() or MAX()的值对于一个指定索引的列 key_col.

总之，就是经常用到的列就最好创建索引。

不能使用引用：

a) 数据唯一性差（一个字段的取值只有几种时）的字段不要使用索引

比如性别，只有两种可能数据。意味着索引的二叉树级别少，多是平级。这样的二叉树查找无异于全表扫描

b) 频繁更新的字段不要使用索引

比如 logincount 登录次数，频繁变化导致索引也频繁变化，增大数据库工作量，降低效率

c) 字段不在 where 语句出现时不要添加索引,如果 where 后含 IS NULL /IS NOT NULL/ like ‘%输入符%’等条件，不建议使用索引只有在 where 语句出现，mysql 才会去使用索引

d） where 子句里对索引列使用不等于（<>），使用索引效果一般

索引失效：

a.如果条件中有 or，即使其中有条件带索引也不会使用(这也是为什么尽量少用 or 的原因)

注意：要想使用 or，又想让索引生效，只能将 or 条件中的每个列都加上索引

b.对于多列索引，不是使用的第一部分，则不会使用索引

c.like 查询是以%开头

d.如果列类型是字符串，那一定要在条件中将数据使用引号引用起来,否则不使用索引

e.如果 mysql 估计使用全表扫描要比使用索引快,则不使用索引

8，java 中的多线程在你们的这个项目当中有哪些体现？

a，后台任务：如定时向大量(100W 以上)的用户发送邮件；定期更新配置文件、任务调度(如 quartz)，一些监控用于定期信息采集

b， 自动作业处理：比如定期备份日志、定期备份数据库

c， 异步处理：如发微博、记录日志

4.11 Redis 分布式锁理解

回答：
实现思想
获取锁的时候，使用 setnx 加锁，并使用 expire 命令为锁添加一个超时时间，超过该时间则自动释放锁，锁的 value 值为一个随机生成的 UUID，通过此在释放锁的时候进行判断。
获取锁的时候还设置一个获取的超时时间，若超过这个时间则放弃获取锁。
释放锁的时候，通过 UUID 判断是不是该锁，若是该锁，则执行 delete 进行锁释放。

4.12 Redis 怎么设置过期的？项目过程中，使用了哪一种持久化方式

回答：
设置过期：
this.redisTemplate.expire("max",tempTime,TimeUnit.SECONDS);
持久化方式：Redis 默认的 RDB 方式

4.13 项目添加 Redis 缓存后，持久化具体怎么实现的。

回答：
RDB：保存存储文件到磁盘；同步时间为 15 分钟，5 分钟，1 分钟一次，可能存在数据丢失问题。
AOF：保存命令文件到磁盘；安全性高，修改后立即同步或每秒同步一次。
上述两种方式在我们的项目中都有使用到，在广告轮播的功能中使用了 redis 缓存，先从 redis 中获取数据，无数据后从数据库中查询后保存到 redis 中

采用默认的 RDB 方式，在广告轮播的功能中使用了 redis 缓存，先从 redis 中获取数据，无数据就从数据库中查询后再保存到 redis 中

4.14 项目中有用到过 redis，访问 redis 是通过什么访问的？

redis 能够存储的数据类型有哪几种？

Redis 通过 SpringDataRedis 访问的.
Redis 支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及 zset(sorted set：有序集合)

4.15 怎样进行程序性能调优

系统性能就是两个事：
Throughput，吞吐量。也就是每秒钟可以处理的请求数，任务数。
Latency， 系统延迟。也就是系统在处理一个请求或一个任务时的延迟。
那么 Latency 越好，能支持的 Throughput 就会越高。因为 Latency 短说明处理速度快，于是就可以处理更多的请求。
提高吞吐量：
分布式集群，模块解藕，设计模式
系统延迟：
异步通信

第5章 数据库设计的面试问题

5.1你有了解 mysql 的隔离级别吗？mysql 默认的隔离级别是什么？

数据库事务的隔离级别有四种，隔离级别高的数据库的可靠性高，但并发量低，而隔离级别低的数据库可靠性低，但并发量高，系统开销小。
1. READ UNCIMMITTED（未提交读）

2. READ COMMITTED（提交读）

3. REPEATABLE READ（可重复读）
4. SERIALIZABLE（可串行化）

mysql 默认的事务处理级别是'REPEATABLE-READ',也就是可重复读。

5.2sql 语句中关于查询语句的优化你们是怎么做的？

1、应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。
2、对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。
3、应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描
4、尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描
5、in 和 not in 也要慎用，否则会导致全表扫描
6、应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描。
7、应尽量避免在 where 子句中对字段进行函数操作，这将导致引擎放弃使用索引而进行全表扫描
8、不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。
9、在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使 用，并且应尽可能的让字段顺序与索引顺序相一致。
10、索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。
11、尽可能的使用 varchar/nvarchar 代替 char/nchar ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。
12、任何地方都不要使用 select * from t ，用具体的字段列表代替“*”，不要返回用不到的任何字段。

5.3mysql 索引失效的场景有哪些？like 做模糊查询的时候会失效吗？

1.WHERE 字句的查询条件里有不等于号（WHERE column!=…），MYSQL 将无法使

用索引

2.类似地，如果 WHERE 字句的查询条件里使用了函数（如：WHERE DAY(column)=...），MYSQL 将无法使用索引

3.在 JOIN 操作中（需要从多个数据表提取数据时），MYSQL 只有在主键和外键的数据类型相同时才能使用索引，否则即使建立了索引也不会使用

4.如果 WHERE 子句的查询条件里使用了比较操作符 LIKE 和 REGEXP，MYSQL 只有在搜索模板的第一个字符不是通配符的情况下才能使用索引。比如说，如果查询条件是 LIKE 'abc%',MYSQL 将使用索引；如果条件是 LIKE '%abc'，MYSQL 将不使用索引。

5.在 ORDER BY 操作中，MYSQL 只有在排序条件不是一个查询条件表达式的情况下才使用索引。尽管如此，在涉及多个数据表的查询里，即使有索引可用，那些索引在加快 ORDER BY 操作方面也没什么作用。

6.如果某个数据列里包含着许多重复的值，就算为它建立了索引也不会有很好的效果。比如说，如果某个数据列里包含了净是些诸如“0/1”或“Y/N”等值，就没有必要为它创建一个索引。

7.索引有用的情况下就太多了。基本只要建立了索引，除了上面提到的索引不会使用的情况下之外，其他情况只要是使用在 WHERE 条件里，ORDER BY 字段，联表字段，一般都是有效的。 建立索引要的就是有效果。 不然还用它干吗？ 如果不能确定在某个字段上建立的索引是否有效果，只要实际进行测试下比较下执行时间就知道。

8.如果条件中有 or(并且其中有 or 的条件是不带索引的)，即使其中有条件带索引也不会使用(这也是为什么尽量少用 or 的原因)。注意：要想使用 or，又想让索引生效，只能将 or 条件中的每个列都加上索引

9.如果列类型是字符串，那一定要在条件中将数据使用引号引用起来,否则不使用索引

10.如果 mysql 估计使用全表扫描要比使用索引快,则不使用索引

问题二：Like 模糊查询，建立索引会失效

5.4项目中关于表结构拆分，你们是业务层面的拆分还是表结构层面的拆分？

表结构层面的拆分。通过 mycat 数据库中间件完成数据库分表操作。

业务层面也有拆分，比如商品模块拆分成 8 张表来实现存储

5.5有了解过大数据层面的分库分表吗？以及 mysql 的执行计划吗？

分库
通过 Mycat 结点来管理不同服务器上的数据库，每个表最多存 500 万条记录
分表
重直切割，水平切割
MySql 提供了 EXPLAIN 语法用来进行查询分析，在 SQL 语句前加一个"EXPLAIN"即可。mysql 中的 explain 语法可以帮助我们改写查询，优化表的结构和索引的设置，从而最大地提高查询效率。

5.6有了解过数据库中的表级锁和行级锁吗？乐观锁和悲观锁你有哪些了解？

MySQL 的锁机制比较简单，其最显著的特点是不同的存储引擎支持不同的锁机制。比如，MyISAM 和 MEMORY 存储引擎采用的是表级锁（table-level locking）；InnoDB 存储引擎既支持行级锁（ row-level locking），也支持表级锁，但默认情况下是采用行级锁。
MySQL 主要的两种锁的特性可大致归纳如下：
表级锁： 开销小，加锁快；不会出现死锁(因为 MyISAM 会一次性获得 SQL 所需的全部锁)；锁定粒度大，发生锁冲突的概率最高,并发度最低。
行级锁： 开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低,并发度也最高。

乐观锁：通过 version 版本字段来实现
悲观锁：通过 for update 来实现

5.7Mysql 优化有没有工具

回答：
三个 MySQL 性能测试工具：The MySQL Benchmark Suite、MySQL super-smack、MyBench。除了第一个为 MySQL 性能测试工具，其他两个都为压力测试工具。

5.8你们项目中使用到的数据库是什么？你有涉及到关于数据库到建库建表操作吗？数据库创建表的时候会有哪些考虑呢？

项目中使用的是 MySQL 数据库，
数据库创建表时要考虑
a、大数据字段最好剥离出单独的表，以便影响性能
b、使用 varchar，代替 char，这是因为 varchar 会动态分配长度，char 指定为 20，即时你存储字符“1”，它依然是 20 的长度
c、给表建立主键，看到好多表没主键，这在查询和索引定义上将有一定的影响
d、避免表字段运行为 null，如果不知道添加什么值，建议设置默认值，特别 int 类型，比如默认值为 0，在索引查询上，效率立显。
e、建立索引，聚集索引则意味着数据的物理存储顺序，最好在唯一的，非空的字段上建立，其它索引也不是越多越好，索引在查询上优势显著，在频繁更新数据的字段上建立聚集索引，后果很严重，插入更新相当忙。
f、组合索引和单索引的建立，要考虑查询实际和具体模式

5.9mysql 中哪些情况下可以使用索引，哪些情况不能使用索引？mysql 索引失效的情形有哪些？

使用索引：
a、 为了快速查找匹配 WHERE 条件中涉及到列。
b、 如果表有一个 multiple-column 索引，任何一个索引的最左前缀可以通过使用优化器来查找行
c、 当运行 joins 时，为了从其他表检索行。MySql 可以更有效的使用索引在多列上如果他们声明的类型和大小是一样的话。在这个环境下，VARCHAR 和 CHAR 是一样的如果他们声明的大小是一样的
d、 为了找到 MIN() or MAX()的值对于一个指定索引的列 key_col.
总之，就是经常用到的列就最好创建索引。

不能使用引用：
a) 数据唯一性差（一个字段的取值只有几种时）的字段不要使用索引
比如性别，只有两种可能数据。意味着索引的二叉树级别少，多是平级。这样的二叉树查找无异于全表扫描
b) 频繁更新的字段不要使用索引
比如 logincount 登录次数，频繁变化导致索引也频繁变化，增大数据库工作量，降低效率

c) 字段不在 where 语句出现时不要添加索引,如果 where 后含 IS NULL /IS NOT NULL/ like ‘%输入符%’等条件，不建议使用索引只有在 where 语句出现，mysql 才会去使用索引

d） where 子句里对索引列使用不等于（<>），使用索引效果一般

索引失效：

a.如果条件中有 or，即使其中有条件带索引也不会使用(这也是为什么尽量少用 or 的原因)

注意：要想使用 or，又想让索引生效，只能将 or 条件中的每个列都加上索引

b.对于多列索引，不是使用的第一部分，则不会使用索引

c.like 查询是以%开头

d.如果列类型是字符串，那一定要在条件中将数据使用引号引用起来,否则不使用索引

e.如果 mysql 估计使用全表扫描要比使用索引快,则不使用索引

8，java 中的多线程在你们的这个项目当中有哪些体现？

a，后台任务：如定时向大量(100W 以上)的用户发送邮件；定期更新配置文件、任务调度(如 quartz)，一些监控用于定期信息采集

b， 自动作业处理：比如定期备份日志、定期备份数据库

c， 异步处理：如发微博、记录日志

5.7 怎样进行数据库优化？

a,选取最适用的字段

在创建表的时候，为了获得更好的性能，我们可以将表中字段的宽度设得尽可能小。另外一个提高效率的方法是在可能的情况下，应该尽量把字段设置为 NOTNULL，

b,使用连接（JOIN）来代替子查询(Sub-Queries)

c,使用联合(UNION)来代替手动创建的临时表

d,事物：

a)要么语句块中每条语句都操作成功，要么都失败。换句话说，就是可以保持数据库中数据的一致性和完整性。事物以 BEGIN 关键字开始，COMMIT 关键字结束。在这之间的一条 SQL 操作失败，那么，ROLLBACK 命令就可以把数据库恢复到 BEGIN 开始之前的状态。

b) 是当多个用户同时使用相同的数据源时，它可以利用锁定数据库的方法来为用户提供一种安全的访问方式，这样可以保证用户的操作不被其它的用户所干扰。

e,锁定表

f,使用外键

锁定表的方法可以维护数据的完整性，但是它却不能保证数据的关联性。这个时候我们就可以使用外键。

g,使用索引

h,优化的查询语句

5.8 怎样进行数据库性能调优

一应用程序优化

（1）把数据库当作奢侈的资源看待，在确保功能的同时，尽可能少地动用数据库资源。

（2）不要直接执行完整的 SQL 语法，尽量通过存储过程实现数据库操作。

（3）客户与服务器连接时，建立连接池，让连接尽量得以重用，以避免时间与资源的损耗。

（4）非到不得已，不要使用游标结构，确实使用时，注意各种游标的特性。

二基本表设计优化

（1）表设计遵循第三范式。在基于表驱动的信息管理系统中，基本表的设计规范是第三范式。

（2）分割表。分割表可分为水平分割表和垂直分割表两种：水平分割是按照行将一个表分割为多个表。

（3）引入中间表。

三 数据库索引优化

索引是建立在表上的一种数据组织，它能提高访问表中一条或多条记录的特定查询效率。

聚集索引

一种索引，该索引中键值的逻辑顺序决定了表中相应行的物理顺序。

聚集索引确定表中数据的物理顺序。

非聚集索引

一种索引，该索引中索引的逻辑顺序与磁盘上行的物理存储顺序不同.

第6章 分布式开发面试问题

6.1分布式架构 session 共享问题，如何在集群里边实现共享。

用了 CAS，所有应用项目中如果需要登录时在 web.xml 中配置过滤器做请求转发到 cas 端工作原理是在 cas 登录后会给浏览器发送一个票据（ticket），浏览器 cookie 中会缓存这个 ticket，在登录其他项目时会拿着浏览器的 ticket 转发到 cas，到 cas 后根据票据判断是否登录

6.2项目中如何配置集群？

配置了 redis 集群，使用 redis3.0 版本官方推荐的配置方式
solr 集群使用了 solrCloud，使用 zookeeper 关联 solrCloud 的配置文件
zookeeper 也配置了集群
应用层使用 Nginx 负载均衡

6.3对分布式，dubbo，zookeeper 说的不太清楚

分布式是从项目业务角度考虑划分项目整个架构。可以将项目基于功能模块划分再

分别部署。Dubbo 是实现分布式项目部署框架。在 zookeeper 是 dubbo 分布式框架的注册中心，管理服务的注册和调用。

6.4 从前端到后台的实现的过程描述的也不清楚

项目前端采用 angularjs 框架在 controller 控制器中完成数据组装和数据展示，在服务层（service）代码完成中后台请求操作。后端基于前端的接口调用，完成数据的增删改查操作。前后端数据交互通过 json 格式字符串完成。

6.5Dubbo 为什么选择 Zookeeper，而不选择 Redis

回答：
引入了 ZooKeeper 作为存储媒介，也就把 ZooKeeper 的特性引进来。
首先是负载均衡，单注册中心的承载能力是有限的，在流量达到一定程度的时候就需要分流，负载均衡就是为了分流而存在的，一个 ZooKeeper 群配合相应的 Web 应用就可以很容易达到负载均衡；资源同步，单单有负载均衡还不够，节点之间的数据和资源需要同步，ZooKeeper 集群就天然具备有这样的功能；
命名服务，将树状结构用于维护全局的服务地址列表，服务提供者在启动 的时候，向 ZK 上的指定节点/dubbo/${serviceName}/providers 目录下写入自己的 URL 地址，这个操作就完成了服务的发布。 其他特性还有 Mast 选举，分布式锁等。

6.6项目中 Zookeeper 服务器挂了，服务调用可以进行吗

回答：
可以的，消费者在启动时，消费者会从 zk 拉取注册的生产者的地址接口等数据，缓存在本地。
每次调用时，按照本地存储的地址进行调用

6.7ActiveMq 消息被重复消费，丢失，或者不消费怎么办

回答：
重复消费：Queue 支持存在多个消费者，但是对一个消息而言，只会有一个消费者可以消费。
丢消息：用持久化消息，或者非持久化消息及时处理不要堆积，或者启动事务，启动事务后，commit() 方法会负责任的等待服务器的返回，也就不会关闭连接导致消息丢失了。
不消费：去 ActiveMQ.DLQ 里找找

6.8怎样解决 activeMQ 的消息持久化问题？

A：持久化为文件

这个你装 ActiveMQ 时默认就是这种，只要你设置消息为持久化就可以了。涉及到的配置和代码有

```
<persistenceAdapter>
<kahaDB directory="${activemq.base}/data/kahadb"/>
</persistenceAdapter>
producer.Send(request, MsgDeliveryMode.Persistent, level, TimeSpan.MinValue);
```

B：持久化为 MySql

加载驱动 jar，为数据中创建三个数据库表，存储 activemq 的消息信息

6.9如果 activeMQ 的消息没有发送成功，怎样确保再次发送成功。

重新传递消息的情况
ActiveMQ 在接收消息的 Client 有以下几种操作的时候，需要重新传递消息：
1：Client 用了 transactions（事务），且在 session 中调用了 rollback()
2：Client 用了 transactions，且在调用 commit()之前关闭
3：Client 在 CLIENT_ACKNOWLEDGE 的传递模式下，在 session 中调用了 recover()
确保客户端有几种状态，检测状态，只要提交了那就说明客户端成功！

6.10 Zookeeper 怎样进行服务治理。

接受提供者的接口信息和提供者 ip 地址进行存储，然后管理消费者和提供者之间调用关系！

6.11 如果 activeMQ 的服务挂了，怎么办？

1、在通常的情况下，非持久化消息是存储在内存中的，持久化消息是存储在文件中的，它们的最大限制在配置文件的<systemUsage>节点中配置。但是，在非持久化消息堆积到一定程度，内存告急的时候，ActiveMQ 会将内存中的非持久化消息写入临时文件中，以腾出内存。虽然都保存到了文件里，但它和持久化消息的区别是，重启后持久化消息会从文件中恢复，非持久化的临时文件会直接删除。

2、考虑高可用，实现 activemq 集群。

6.12 如果 zookeeper 服务挂了怎么办？

注册中心对等集群，任意一台宕掉后，会自动切换到另一台
注册中心全部宕掉，服务提供者和消费者仍可以通过本地缓存通讯
服务提供者无状态，任一台宕机后，不影响使用
服务提供者全部宕机，服务消费者会无法使用，并无限次重连等待服务者恢复

6.13 Dubbo 有 3 次重试，假如新消息被重复消费怎么处理

回答：
1、去掉超时重试机制
2、服务端增加幂等校验，服务器加入校验机制，如果这个消息已被 消费就不再重复消费

6.14 mq 消费者接收不到消息怎么办。

Mq 消费者接受不到消息存在 2 中情况：
1. 处理失败 指的是 MessageListener 的 onMessage 方法里抛出 RuntimeException。
2. Message 头里有两个相关字段：Redelivered 默认为 false，redeliveryCounter 默认为 0。
3. 消息先由 broker 发送给 consumer，consumer 调用 listener，如果处理失败，本地 redeliveryCounter++，给 broker 一个特定应答，broker 端的 message 里 redeliveryCounter++，延迟一点时间继续调用，默认 1s。超过 6 次，则给 broker 另一个特定应答，broker 就直接发送消息到 DLQ。
4. 如果失败 2 次，consumer 重启，则 broker 再推过来的消息里，redeliveryCounter=2，本地只能再重试 4 次即会进入 DLQ。
5. 重试的特定应答发送到 broker，broker 即会在内存将消息的 redelivered 设置为 true，redeliveryCounter++，但是这两个字段都没有持久化，即没有修改存储中的消息记录。所以 broker

重启时这两个字段会被重置为默认值。

6.15 系统的高并发问题是怎么解决的。

并发问题高，这个问题的解决方案是一个系统性的，系统的每一层面都需要做优化：

1） 数据层

a) 集群

b) 分表分库

c) 开启索引

d) 开启缓存

e) 表设计优化

f) Sql 语句优化

g) 缓存服务器（提高查询效率，减轻数据库压力）

h) 搜索服务器（提高查询效率，减轻数据库压力）

2） 项目层

a) 采用面向服务分布式架构（分担服务器压力，提高并发能力）

b) 采用并发访问较高的详情系统采用静态页面

c) 使用页面缓存

d) 用 ActiveMQ 使得业务进一步进行解耦，提高业务处理能力

e) 使用分布式文件系统存储海量文件

3） 应用层

a) Nginx 服务器来做负载均衡

b) Lvs 做二层负载

6.16 并发数多少，项目中怎么解决并发问题？

面试中项目的并发数不宜说的过大，安装目前品优购项目拆分规模，这个项目的并发是在10000+，但是学生面试不能说的这么高。

可以有以下 2 方面的回答：

1） 项目并发并不清楚（只是底层程序员）

2） 参与核心业务设计，知道并发是多少（测试峰值，上线并发）

3000---5000 吧

面对项目高并发，项目必须做各种优化措施了：

4） 数据层

a) 集群

b) 分表分库

c) 开启索引

d) 开启缓存

e) 表设计优化

f) Sql 语句优化

g) 缓存服务器（提高查询效率，减轻数据库压力）

h) 搜索服务器（提高查询效率，减轻数据库压力）

5） 项目层

a) 采用面向服务分布式架构（分担服务器压力，提高并发能力）

b) 采用并发访问较高的详情系统采用静态页面

c) 使用页面缓存

d) 用 ActiveMQ 使得业务进一步进行解耦，提高业务处理能力

e) 使用分布式文件系统存储海量文件

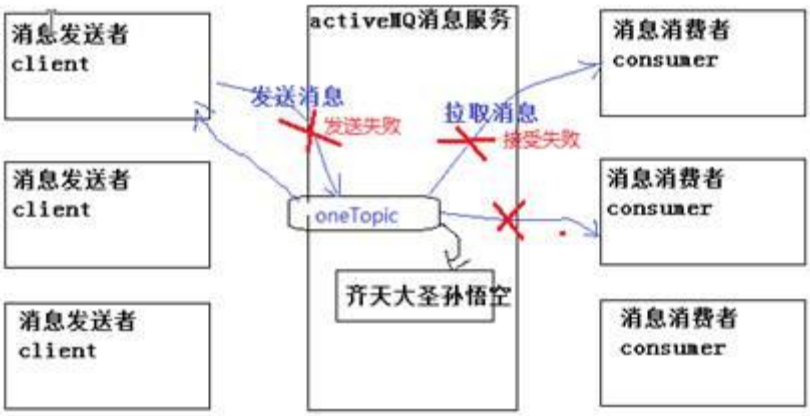
6） 应用层

a) Nginx 服务器来做负载均衡

b) Lvs 做二层负载

6.17 消息发送失败怎么处理，发送数据，数据库已经保存了数据，但是 redis 中没有同步，怎么办。或者说如何做到消息同步。

消息发送失败，可以进行消息的重新发送，可以配置消息的重发次数。



如果消息重发完毕后，消息还没有接受成功，重启服务。

6.18 Dubbo 的通信原理？

Dubbo 底层使用 hessain2 进行二进制序列化进行远程调用

Dubbo 底层使用 netty 框架进行异步通信。NIO

第7章 其他技术面试问题

7.1 单点登录的访问或者跨域问题

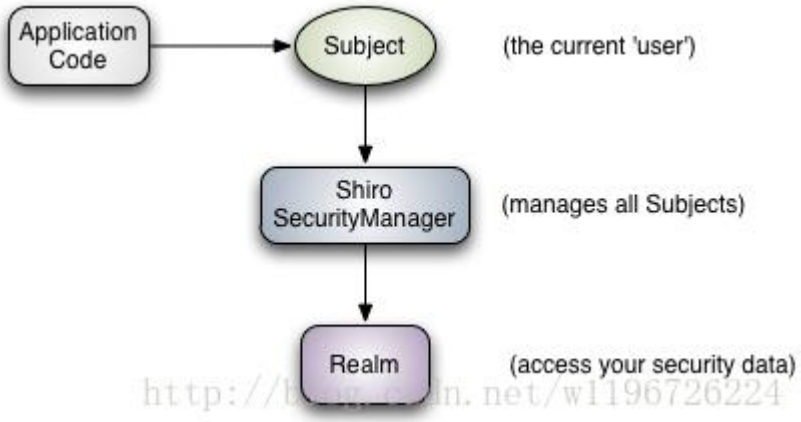
首先要理解什么是单点登录。单点登录是相互信任的系统模块登录一个模块后，其他模块不需

要重复登录即认证通过。项目采用的是 CAS 单点登录框架完成的。首先 CAS 有两大部分。客户端和服务端。服务端就是一个 web 工程部署在 tomcat 中。在服务端完成用户认证操作。每次访问系统模块时，需要去 CAS 完成获取 ticket。当验证通过后，访问继续操作。对于 CAS 服务端来说，我们访问的应用模块就是 CAS 客户端。

跨域问题，首先明白什么是跨域。什么时候涉及跨域问题。当涉及前端异步请求的时候才涉及跨域。那什么是跨域呢？当异步请求时，访问的请求地址的协议、ip 地址、端口号任意一个与当前站点不同时，就会涉及跨域访问。解决方案：1、jQuery 提供了 jsonp 实现 2、W3C 标准提供了 CORS（跨域资源共享）解决方案。

7.2 shiro 安全认证时如何做的

要明白 shiro 执行流程以及 shiro 的核心组件，可参考下图



认证过程：

在 application Code 应用程序中调用 subject 的 login 方法。将页面收集的用户名和密码传给安全管理器 securityManager，将用户名传给 realm 对象。Realm 对象可以理解为是安全数据桥，realm 中认证方法基于用户名从数据库中查询用户信息。如果用户存在，将数据库查询密码返回给安全管理器 securityManager，然后安全管理器判断密码是否正确。

7.3 solr 的用途

solr 在系统中主要完成商品搜索功能，提高搜索性能。

7.4 分布式锁的问题

针对分布式锁的实现，目前比较常用的有以下几种方案：
1.基于数据库实现分布式锁
2.基于缓存（redis，memcached，tair）实现分布式锁
3.基于 zookeeper 实现分布式锁

7.5 solr索引中使用了IK分词器，你们项目中使用到了分词器的哪种工作模式？

IK 分词器，基本可分为两种模式，一种为 smart 模式，一种为非 smart 模式。
例如：张三说的确实在理
smart 模式的下分词结果为：
张三 | 说的 | 确实 | 在理
而非 smart 模式下的分词结果为：
张三 | 三 | 说的 | 的确 | 的 | 确实 | 实在 | 在理
可见非 smart 模式所做的就是将能够分出来的词全部输出；smart 模式下，IK 分词器则会根据内在方法输出一个认为最合理的分词结果，这就涉及到了歧义判断。

项目中采用的是 smart 模块分词的。

7.6java 中关于多线程的了解你有多少？线程池有涉及吗？

同一类线程共享代码和数据空间，每个线程有独立的运行栈和程序计数器(PC)，线程切换开销小。线程分为五个阶段：创建、就绪、运行、阻塞、终止。
Java 线程有五种基本状态
新建状态（New）：当线程对象对创建后，即进入了新建状态，如：Thread t = new MyThread();
就绪状态（Runnable）：当调用线程对象的 start()方法（t.start();），线程即进入就绪状态。处于就绪状态的线程，只是说明此线程已经做好了准备，随时等待 CPU 调度执行，并不是说执行了 t.start()此线程立即就会执行；
运行状态（Running）：当 CPU 开始调度处于就绪状态的线程时，此时线程才得以真正执行，即进入到运行状态。注：就 绪状态是进入到运行状态的唯一入口，也就是说，线程要想进入运行状态执行，首先必须处于就绪状态中；
阻塞状态（Blocked）：处于运行状态中的线程由于某种原因，暂时放弃对 CPU 的使用权，停止执行，此时进入阻塞状态，直到其进入到就绪状态，才 有机会再次被 CPU 调用以进入到运行状态。根据阻塞产生的原因不同，阻塞状态又可以分为三种：

1.等待阻塞：运行状态中的线程执行 wait()方法，使本线程进入到等待阻塞状态；
2.同步阻塞 -- 线程在获取 synchronized 同步锁失败(因为锁被其它线程所占用)，它会进入同步阻塞状态；
3.其他阻塞 -- 通过调用线程的 sleep()或 join()或发出了 I/O 请求时，线程会进入到阻塞状态。当 sleep()状态超时、join()等待线程终止或者超时、或者 I/O 处理完毕时，线程重新转入就绪状态。
死亡状态（Dead）：线程执行完了或者因异常退出了 run()方法，该线程结束生命周期。

Java 中线程的创建常见有如三种基本形式
1.继承 Thread 类，重写该类的 run()方法。
2.实现 Runnable 接口，并重写该接口的 run()方法，
该 run()方法同样是线程执行体，创建 Runnable 实现类的实例，并以此实例作为 Thread 类的 target 来创建 Thread 对象，该 Thread 对象才是真正的线程对象。
3.使用 Callable 和 Future 接口创建线程。
具体是创建 Callable 接口的实现类，并实现 clall()方法。并使用 FutureTask 类来包装 Callable 实现类的对象，且以此 FutureTask 对象作为 Thread 对象的 target 来创建线程。
线程池：线程池是一种多线程处理形式，处理过程中将任务添加到队列，然后在创建线程后自动启动这些任务。线程池线程都是后台线程。每个线程都使用默认的堆栈大小，以默认的优先级运行，并处于多线程单元中。如果某个线程在托管代码中空闲（如正在等待某个事件），则线程池将插入另一个辅助线程来使所有处理器保持繁忙。如果所有线程池线程都始终保持繁忙，但队列中包含挂起的工作，则线程池将在一段时间后创建另一个辅助线程但线程的数目永远不会超过最大值。超过最大值的线程可以排队，但他们要等到其他线程完成后才启动。

7.7如何实现线程的同步？

为何要使用同步？

java 允许多线程并发控制，当多个线程同时操作一个可共享的资源变量时（如数据的增删改查），将会导致数据不准确，相互之间产生冲突，因此加入同步锁以避免在该线程没有完成操作之前，被其他线程的调用，从而保证了该变量的唯一性和准确性。
线程同步（5 种同步方式）
1.同步方法 2.同步代码块 3.使用特殊域变量(volatile)实现线程同步 4.使用重入锁实现线程同步 5.使用局部变量实现线程同步

7.8遍历 hashmap 有几种方式？

Map 的四种遍历方式
(1) for each map.entrySet()
(2) 显示调用 map.entrySet()的集合迭代器

(3) for each map.keySet()，再调用 get 获取
(4) for each map.entrySet()，用临时变量保存 map.entrySet()

7.9简单介绍一下solr全文检索在整个系统中的应用，在更新索引库的同时会产生索引碎片，这个碎片是如何处理的？

根据商品的名称，分类，品牌等属性来创建索引进行商品搜索。

更新索引库时会先删除索引，然后再重建。而对于删除聚集索引，则会导致对应的非聚集索引重建两次(删除时重建，建立时再重建).

直接删除碎片。

7.10 java 并发包下有哪些并发组件？

分为两层组成

外层框架主要有 Lock(ReentrantLock、ReadWriteLock 等)、同步器（semaphores 等）、阻塞队列（BlockingQueue 等）、Executor（线程池）、并发容器（ConcurrentHashMap 等）、还有 Fork/Join 框架；

内层有 AQS（AbstractQueuedSynchronizer 类，锁功能都由他实现）、非阻塞数据结构、原子变量类(AtomicInteger 等无锁线程安全类)三种。

7.11 讲一下 jvm 调优。

a,堆大小设置
b,回收器选择
c,辅助信息

JVM 提供了大量命令行参数，打印信息，供调试使用；

7.12 讲一下 jvm 的组成。

JVM 由类加载器子系统、运行时数据区、执行引擎以及本地方法接口组成

7.13 讲一下 ThreadLocal 类。

ThreadLocal，很多地方叫做线程本地变量，也有些地方叫做线程本地存储，其实意思差不多。可能很多朋友都知道 ThreadLocal 为变量在每个线程中都创建了一个副本，那么每个线程可以访问自己内部的副本变量；

ThreadLocal 在每个线程中对该变量会创建一个副本，即每个线程内部都会有一个该变量，且在线程内部任何地方都可以使用，线程之间互不影响，这样一来就不存在线程安全问题，也不会严重影响程序执行性能。

但是要注意，虽然 ThreadLocal 能够解决上面说的问题，但是由于在每个线程中都创建了副本，所以要考虑它对资源的消耗，比如内存的占用会比不使用 ThreadLocal 要大；

7.14 Solr 热搜索

回答：

和 Redis 热搜索一样将经常搜索的词，新建 collection，对这些搜索内容单独进行索引。

7.15 简单介绍一下 solr 全文检索在整个系统中的应用，在更新索引库的同时会产生索引碎片，这个碎片是如何处理的？

根据商品的名称，分类，品牌等属性来创建索引进行商品搜索。

更新索引库时会先删除索引，然后再重建。而对于删除聚集索引，则会导致对应的非聚集索引重建两次(删除时重建，建立时再重建).

直接删除碎片。

7.16 怎么确保 session 共享？

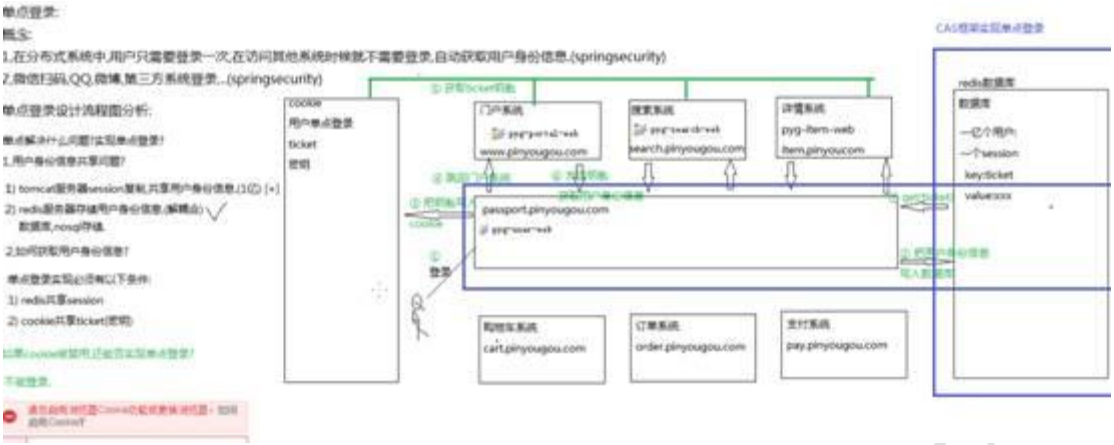
在分布式项目中实现 session 共享必须做以下准备工作：

1） Cookie 中共享 ticket

2） Redis 存储 session

分布式系统共享用户身份信息 session，必须先获取 ticket 票据，然后再根据票据信息获取 redis 中用户身份信息。

实现以上 2 点即可实现 session 共享。



目前项目中使用的 springsecurity + cas 来实现的单点登录，cas 自动产生 ticket 票据信息，每次获取用户信息，cas 将会携带 ticket 信息获取用户身份信息。

7.17 项目中哪块涉及了线程问题，怎么处理的？

项目的高并发访问就是一个多线程问题。
项目中普通的业务开发基本没有涉及多线程问题，不过你可以谈谈你使用的框架中使用的多线程技术：
因为我们项目使用的框架进行开发的，因此多线程处理多让框架非我们处理结束了。

1） 高并发就是多线程，这里的多线程让 servlet 服务器给处理了谈谈 Tomcat 多线程配置;
   a) 配置线程池，扩大并发能力
   b) 开启 NIO 能力等等
2） 框架多线程：mybatis 框架底层使用的连接池

7.18 传统 IO 和 NIO 的区别？

IO 是直接通信

NIO 是由缓存作为中间层，先把数据放入缓存（内存），数据的消防者可以从缓存中获取数据。