

Java初面  
一、电商项目

1. 说说你最近做的这个项目的背景，简单的介绍一下你这个项目

电商项目的背景一般是由市场推动的，比如行业竞争或者经营方式的改变(营销理念)。竞争的形态也发生了巨大的变化，从以产品、价格为主的竞争转向以服务为主的竞争，服务成为主导竞争格局的重要因素。渠道作为企业完成客户沟通、产品/服务交换过程以及实现价值、产生效益的重要载体，发挥了采集、传达客户和竞争对手等市场信息，为买卖双方提供便利，协调供需矛盾，为客户提供合适的产品与服务，向客户传递产品/服务信息，实现营销/服务目标等重要的功能。

XXX商城之前主要以实体店为主，进行批发与零售。业务也相对比较传统，为了提升业务绩效，增强客户满意度和粘性，另一方面，也为基于互联网的商务模式创新奠定基础。针对上述行业环境变化和业务战略目标，xxx商城网上终端预约销售基础上，即将启动网上商城建设项目，用于建立网上终端、营销案在线销售及相关辅助功能，包含商品管理、订单管理、类目管理、客户管理、合作商管理、客服管理、购物平台、内容管理等，很大程度上分担了人工的压力，对提高客户服务效率和客户满意度能够起到较好的作用。基于此，XXX公司提出建设网上商城建设项目工程。

项目介绍

xxx商城项目打造的是“社区+电商”的模式，用户不只是在社区中有自己的圈子，还可以将电商加入到社区中，整个电商网站实现的功能非常之多，采用分布式的架构设计，包括后台管理、前台系统、订单系统、单点登录系统、搜索系统、会员系统等。

1. 该项目是自己公司的产品，我们公司负责整个网站的运营，属于B2C平台；
2. 系统的用途，主要是提供B2C的平台，其中自营商品也有商家入住，类似天猫。
3. 系统架构，采用分布式的系统架构，其中前台系统和单点登录系统采用了集群的方式部署，在后台管理系统中采用了Maven的多模块化的管理，其中采用了水平切分的方式，将pojo、dao、service、web分层开发，这样做的好处就是可以重用性更高。
4. 部署方面，采用了Nginx+tomcat的模式，其中nginx的作用一方面是做反向代理、负载均衡、另一方面是做图片等静态资源的服务器。
5. 项目介绍应该从这个项目的模式、功能、架构、解决了什么问题、部署以及投入使用情况来说。
6. 整个项目的架构如何

一般情况这个问题要从两个方面去回答，从技术架构和功能架构去谈

技术架构：

本项目使用市场上较为主流的框架spring+springmvc+mybatis进行开发，采用分布式的系统架构，前台系统和单点登录系统采用了集群的方式部署，后台管理系统中采用了Maven的多模块化的管理，其中采用了水平切分的方式，将pojo、dao、service、web分层开发，这样做的好处就是可以重用性更高。系统内部接口调用采用Httpclient，并且使用Httpclient的连接池技术，接口提供端采用RESTful方式的接口定义；系统之间的通知机制采用MQ的方式，使用RabbitMQ的实现，使用了RabbitMQ的消息订阅模式的消息机制；系统的接口还对JS的跨域做了支持，采用了jsonp的解决方法，在后台接口中扩展了spirng提供的jackson数据转化器实现；搜索系统使用了solr实现，在保证系统高性能的前提下，尽可能为公司节约成本，我们使用MySQL数据库进行集群(oracle收费)。在部署方面，采用了Nginx+tomcat的模式，其中nginx的作用一方面是做反向代理、负载均衡、另一方面是做图片等静态资源的服务器。

功能架构：分布式系统架构

各个系统说明：

后台管理系统：管理商品、订单、类目、商品规格属性、用户管理以及内容发布等功能。

前台系统：用户可以在前台系统中进行注册、登录、浏览商品、首页、下单等操作。

会员系统：用户可以在该系统中查询已下的订单、收藏的商品、我的优惠券、团购等信息。

订单系统：提供下单、查询订单、修改订单状态、定时处理订单。

搜索系统：提供商品的搜索功能。

单点登录系统：为多个系统之间提供用户登录凭证以及查询登录用户的信息。

谈到分布式架构，我们必须对比传统架构才能彰显其优势。

1. 最为明显的一点，在传统的架构中，如果某个功能需要进行维护，那么我们必须停掉整个服务，这对于公司的运营会造成损失。分布式系统在核心功能模块使用单独服务器，维护部分模块不影响用户的其他操作。
2. 在海量数据处理方面，传统架构显得比较乏力；分布式系统架构采用服务器集群，使用负载均衡，海量数据处理游刃有余！
3. 在性能(检索)以及维护方面，分布式系统架构也有较为明显的优势。
4. 这个项目为用户提供了哪些服务，包括哪些功能

商品管理模块：其中包括品牌管理，属性管理商品录入/上下架管理，商品添加审核，静态页面发布

订单模块：其中包括使用activiti工作流订单的查询和订单的流转,定时作废

商品前台首页：其中主要负责首页商品列表筛选，首页上动态展示筛选条件，点击每一个筛选条件下面的商品列表要做联动

单品页面：采用freemarker来实现页面静态化，展示商品详情信息和商品购买，该页面采用静态化以减轻系统压力，使用了cxf框架发布服务

提交订单页面：提交用户的订单信息, 处理并发问题。

个人中心，包括用户的登录，个人信息的管理，收货地址的管理，用户所下的订单的管理

购物车：把购物车的信息存在cookie里面管理

1. 你承担这个项目的哪些核心模块

在项目中主要负责相关系统的开发，主要有：

* 1. 后台管理系统，主要实现商品管理、商品规格参数管理、订单管理、会员管理等、CMS(内容管理系统)等，并且提供了跨域支持
  2. 前台系统，主要是面向用户访问，使用Httpclient和后台系统接口做交互，并且该系统在部署上采用集群的方式；
  3. 单点登录系统，主要是提供集中用户登录凭证的集中解决方案，提供和用户信息相关的接口，比如说用户注册、查询等接口。
  4. 订单系统，主要是提供和订单相关的业务接口，在订单系统了做了严格的数据校验以及高并发写的支持（这里可以说使用队列实现），并且使用了Quartz定时任务实现对订单的定时扫描，比如说关闭超时未付款的订单；
  5. 搜索系统，主要是提供商品的搜索，采用开源企业级系统Solr实现，采用了MQ机制保证了商品数据可以及时同步到solr中；
  6. 会员系统，主要是维护用户的信息，已购买订单、优惠券、系统消息、修改密码、绑定手机等功能；
  7. 缓存，主要是用Redis实现，并且对Redis做了集群来保证Redis服务的高可用。8) 支付系统，主要是负责订单的支付、对账等功能，主要是对接了支付宝的接口；

二、多线程基础知识

1、并发编程

1.1 并发编程的优缺点

优点：

充分利用多核CPU的计算能力，通过并发编程的形式将多核CPU的计算能力发挥到极致，性能得到提升。

方面进行业务的拆分。提高系统并发能力和性能：高并发系统的开发，并发编程会显得尤为重要，利用好多线程机制可以大大提高系统的并发能力及性能；面对复杂的业务模型，并行程序会比串行程序更适应业务需求，而并发编程更适合这种业务拆分。cai

缺点：

并发编程的目的是为了提高程序的执行效率，提高程序运行速度，但并发编程并不是总能提高性能，有时还会遇到很多问题，例如：内存泄漏，线程安全，死锁等。

1.2 并发编程的三要素

并发编程的三要素：（也是带来线程安全所在）

1. 原子性：原子是不可再分割的最小单元，原子性是指一个或多个操作要么全部执行成功，要么全部执行失败。
2. 可见性：一个线程对共享变量的修改，另一个线程能看到
3. 有序性：程序的执行顺序按照代码的先后顺序

线程安全的问题原因有：

1. 线程切换带来的原子性问题

2. 缓存导致的可见性问题

3. 编译优化带来的有序性问题

1.3 并发和并行有和区别

并发：多个任务在同一个CPU上，按照细分的时间片轮流交替执行，由于时间很短，看上去好像是同时进行的。

并行：单位时间内，多个处理器或多核处理器同时处理多个任务，是真正意义上的同时进行。

串行：有n个任务，由一个线程按照顺序执行。

1.4 什么是多线程，多线程的优势？

定义：多线程是指程序中包含多个流，即在一个程序中可以同时进行多个不同的线程来执行不同的任务

优点:

可以提高CPU的利用率，在多线程中，一个线程必须等待的时候，CPU可以运行其它线程而不是等待，这样就大大提高了程序的效率，也就是说单个程序可以创建多个不同的线程来完成各自的任务。

缺点：

线程也是程序，线程也需要占内存，线程也多内存也占的也多。

多线程需要协调和管理，所以需要CPU跟踪线程。

线程之间共享资源的访问会相互影响，必须解决禁用共享资源的问题。

2、线程与进程

2.1什么是线程与进程

进程：

内存中运行的运用程序，每个进程都有自己独立的内存空间，一个进程可以由多个线程，例如在Windows系统中，xxx.exe就是一个进程。

线程：

进程中的一个控制单元，负责当前进程中的程序执行，一个进程至少有一个线程，一个进程可以运行多个线程，多个线程可以共享数据。

2.2线程与进程的区别

**根本区别**：

进程是操作系统资源分配的基本单元，而线程是处理器任务调度的和执行的基本单位。

资源开销：  
 每个进程都有自己独立的代码和空间(程序上下文)，程序之间的切换会有较大的开销；线程可以看作轻量级的进程，同一类线程共享代码和数据空间，每个线程都有自己独立的运行栈和程序计数器(PC)，线程之间切换的开销小。

包含关系：

如果一个进程内有多个线程，则执行的过程不是一条线的，而是多条线（多个线程），共同完成；线程是进程的一部分，可以把线程看作是轻量级的进程。

内存分配：

同一进程的线程共享本进程的地址空间和资源，而进程之间的地址空间和资源是相互独立的。

2.3 用户线程与守护线程

用户（User）线程：运行在前台，执行具体任务，如程序的主线程，连接网络的子线程都是用户线程。

守护（Daemon）线程：运行在后台，为其它前台线程服务，也可以说守护线程是JVM非守护线程的”佣人“，一旦所有线程都执行结束，守护线程会随着JVM一起结束运行。

main函数就是一个用户线程，main函数启动时，同时JVM还启动了好多的守护线程，如垃圾回收线程，比较明显的区别时，用户线程结束，JVM退出，不管这个时候有没有守护线程的运行，都不会影响JVM的退出。

2.4 什么是线程死锁

死锁是指两个或两个以上进程(线程)在执行过程中，由于竞争资源或由于彼此通信造成的一种堵塞的现象，若无外力的作用下，都将无法推进，此时的系统处于死锁状态。

2.5 形成死锁的四个必要条件

* 互斥条件：线程(进程)对所分配的资源具有排它性，即一个资源只能被一个进程占用，直到该进程被释放。
* 请求与保持条件：一个进程(线程)因请求被占有资源而发生堵塞时，对已获取的资源保持不放。
* 不剥夺条件：线程(进程)已获取的资源在未使用完之前不能被其他线程强行剥夺，只有等自己使用完才释放资源。
* 循环等待条件：当发生死锁时，所等待的线程(进程)必定形成一个环路，死循环造成永久堵塞。

2.6 如何避免死锁

我们只需破坏形参死锁的四个必要条件之一即可。

1. 破坏互斥条件：无法破坏，我们的🔒本身就是来个线程(进程)来产生互斥
2. 破坏请求与保持条件：一次申请所有资源
3. 破坏不剥夺条件：占有部分资源的线程尝试申请其它资源，如果申请不到，可以主动释放它占有的资源。
4. 破坏循环等待条件：按序来申请资源。

三、秒杀模块

1、秒杀流程

1. 登录进入商品列表页面，静态资源缓存
2. 点击进入商品详情页面，静态资源缓存，Ajax获取验证码等动态信息
3. 点击秒杀, 将验证码结果和商品ID传给后端，如果结果正确。动态生成随机串UUID,结合用户ID和商品ID存入redis，并将path传给前端。前端获取path后，再根据path地址调用秒杀服务
4. 服务端获取请求的path参数，去查缓存是否在
5. 如果存在，并且Redis还有库存，预减redis库存，看是否已经生成订单，没有的话就将请求入消息队列
6. 从消息队列中取消息：获取商品ID和用户ID，判断数据库库存，然后下单
7. 下单：减库存，生成订单
8. 前端轮询订单生成结果。50ms继续轮询或者秒杀是否成功和失败

四、SpringBoot基础问题

1、什么是SpringBoot

通过Spring Boot，可以轻松地创建独立的，基于生产级别的Spring的应用程序，您可以“运行”它们。大多数Spring Boot应用程序需要最少的Spring配置。

2、如何快速构建一个SpringBoot项目

1. 通过Web界面使用。http://start.spring.io
2. 通过Spring Tool Suite使用。
3. 通过IntelliJ IDEA使用。
4. 使用Spring Boot CLI使用。

3、SpringBoot自动配置原理

@EnableAutoConfiguration (开启自动配置)

该注解引入了AutoConfigurationImportSelector，该类中的方法会扫描所有存在META-INF/spring.factories的jar包。

4、SpringBoot读取配置相关注解有哪些

@PropertySource

@Value

@Environment

@ConfigurationProperties

5、SpringBoot核心组件

四大组件分别是：starter， autoconfigure, CLI 以及actuator

五、微服务

1、什么是微服务

微服务架构是一种架构模式或者说是一种架构风格，它提倡将单一应用程序划分成一组小的服务，每个服务运行在其独立的自己的进程中，服务之间互相协调、互相配合，为用户提供最终价值。 服务之间采用轻量级的通信机制互相沟通（通常是基于HTTP的RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立地部署到生产环境、类生产环境等。另外，应尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建，可以有一个非常轻量级的集中式管理来协调这些服务，可以使用不同的语言来编写服务，也可以使用不同的数据存储。

2、微服务之间是如何通讯的

1. 远程过程调用，直接通过远程过程调用来访问别的service

优点：简单，常见。因为没有中间件代理，系统更简单

缺点：只支持请求/响应的模式，不支持别的，比如通知、请求/异步响应、发布/订阅、发布/异步响应降低了可用性，因为客户端和服务端在请求过程中必须都是可用的

1. 消息，使用异步消息来做服务间通信。服务间通过消息管道来交换消息，从而通信。

优点：

把客户端和服务端解耦，更松耦合 提高可用性，因为消息中间件缓存了消息，直到消费者可以消费

支持很多通信机制比如通知、请求/异步响应、发布/订阅、发布/异步响应

缺点：

消息中间件有额外的复杂性

3、springcloud 与dubbo有哪些区别

相同点：

SpringCloud 和Dubbo可以实现RPC远程调用框架，可以实现服务治理。

不同点：

SpringCloud是一套目前比较网站微服务框架了，整合了分布式常用解决方案遇到了问题注册中心Eureka、负载均衡器Ribbon ，客户端调用工具Rest和Feign，分布式配置中心Config，服务保护Hystrix，网关Zuul Gateway ，服务链路Zipkin，消息总线Bus等。

Dubbo内部实现功能没有SpringCloud强大（全家桶），只是实现服务治理，缺少分布式配置中心、网关、链路、总线等，如果需要用到这些组件，需要整合其他框架。

4、什么是服务熔断，什么是服务降级

服务熔断

熔断机制是应对雪崩效应的一种微服务链路保护机制。

当扇出链路的某个微服务不可用或者响应时间太长时，会进行服务的降级，进而熔断该节点微服务的调用，快速返回"错误"的响应信息。当检测到该节点微服务调用响应正常后恢复调用链路。在SpringCloud框架里熔断机制通过Hystrix实现。Hystrix会监控微服务间调用的状况，当失败的调用到一定阈值，缺省是5秒内20次调用失败就会启动熔断机制。熔断机制的注解是@HystrixCommand。

Hystrix服务降级

其实就是线程池中单个线程障处理，防止单个线程请求时间太长，导致资源长期被占有而得不到释放，从而导致线程池被快速占用完，导致服务崩溃。

六、Mybatis基础

1、mybatis 有几种分页方式

2、mybatis 分页插件的实现原理是什么

3、说一下 mybatis 的一级缓存和二级缓存

七、MySql

1、数据库的三范式是什么

2、如何获取当前数据库版本

3、mysql 索引是怎么实现的

4、怎么验证 mysql 的索引是否满足需求

5、说一下 mysql 常用的引擎

6、说一下 mysql 的行锁和表锁

7、如何做 mysql 的性能优化

八、Redis

1、redis 是什么？都有哪些使用场景？

2、redis 有哪些功能？

3、redis 为什么是单线程的？

4、redis 支持的数据类型有哪些？

5、redis 怎么实现分布式锁？

6、redis 分布式锁有什么缺陷？

九、金融类

1、请您简述你所具备的的技能以及使用过的框架，尤其是Hibernate和Spring。

2、简要介绍你现在正在做的项目和使用的技术

3、Spring使用了那种设计模式，简要介绍。

4、IOC作用。

5、使用Spring框架有什么好处。

6、除了Spring核心，你还用过什么其他Spring组件