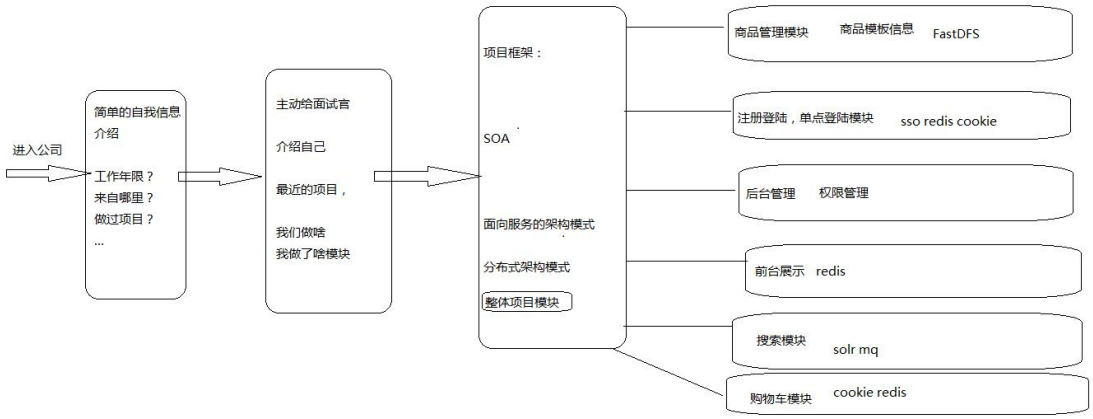
一.面试指导方针

是什么？有什么特点？ what
为什么用它? why
使用的场景?使用的具体步骤?使用过程中问题总结？ how

面试自我介绍之大套路



什么是分布式？什么是集群

把系统按照模块拆分成多个子系统
同一个工程部署到多台服务器上

什么是 SOA 架构？简单介绍一下什么是面向服务的架构？

SOA 代表了面向服务的架构。 简单理解为表现层和服务层进行分离

SOA 它是一种使用松耦合子服务构建业务应用的体系架构，这些服务可以通过编排连接在一起以实现特定的功能。然后可以把咱们讲的架构图给面试官说一下

在简单介绍下可重用服务这块 （服务是一个自主的，可重复使用的，可发现的，无状态的，有一定粒度的功能，并且是一个复合应用程序或一个组合服务的一部分）

1、你们公司日志这一块是怎么记录的？

日志处理我们使用的是 log4j, 有一个 log4j 的配置文件, 可以配置 log 输出的位置和 log 的输出形式, 我们对于整个项目，设置了一个全局异常，当出现异常信息的时候，将异常信息记录到 log 中

```
Logger logger = LoggerFactory.getLogger(GloableException.class);
logger.error("-------------出错了-----------------");
```

当有些需要记录内容的信息，也可以通过日志文件进行记录。
对于用户登陆日志记录,我们需要自己封装一个日志记录的工具类,可以将用户登陆的信息记录到数据库中。(具体操作步骤看如下链接)
http://www.cnblogs.com/marcello/articles/4501479.html

2、库存警告是怎么使用的？

一般我们都是去，恩，就是说当我们每次做完出库后，我们后台每隔 5 分钟会跑批查看库存小于 100 的商品，然后说就是会有相应的短信提醒，告诉库管或者商户具体哪个商品少了，然后商户或者库管登录后台进行补充库存

3、权限管理你们是单独的一个呀，还是怎么弄的？

我们之前做过权限管理，我们做的时候使用的五张表去做的这样的权限管理，有一张用户表，一张角色表，一张权限表，还有一张用户角色的桥表和角色权限的桥表。
我们是通过用户的 id 查询它所对应的角色，通过角色查询他所对应的权限，通过权限找到相对应的 url,其实我还是知道有一个 shiro 框架也是可以控制权限的,目前项目里还没用过呢.

4、项目中你主要负责那块？

这就集思广益了，按照自己简历上的项目，然后针对性的找出自己要说哪几点，首先介绍这个模块是干什么的，然后这个是怎么用的，然后在哪里用的。

5、Spu 和 Sku 分别是什么意思？

sKU 是库存量单位,区分单品,.另外还有 SPU,是标准化的产品单元,区分品种,例如 iphon8 这是一个 SPU 苹果 8 (64G 黑色 移动) 这就是一个 SKU.

6、项目中的管理工具用的什么

你是说的项目版本管理,还是项目进度管理,我们项目版本管理用的是 SVN,我们项目进度管理项目经理用的是禅道,还有那个 project 软件管理的进度,还有我们项目里 jar 包管理项目之间的依赖用的是 maven 做的.

7、项目中用的什么做的压力测试

我知道测试那边使用 jmeter 或者是 LoadRunner 做的压力测试,loadRunner 还能做负载测试, 我原来接触过那个 loadRunner,他就是是 Windows 系统下的一个测试工具,产品给提出性能需求,他们进行测试,什么是性能需求?,比如说产品经理要求并发同时提交 1000 个订单,要求成功率是 95%以上,或者是负载的需求,模拟 1000 个用户同时在线下单或者浏览商品看下要求系统能坚持 10 分钟以上,等这些需求.大概我知道就这样.

8、项目当中的多线程，线程池是怎么回事

多线程其实在我们项目中用的不多,我知道多线程就是为了解决多任务同时执行的需求，合理使用 CPU 资源。多线程的运行是根据 CPU 切换完成，如何切换由 CPU 决定，因此多线程运行具有不确定性。
线程池：现在服务器端的应用程序几乎都采用了“线程池”技术，其实和数据库连接池差不多,这主要是为了提高系统效率。因为如果服务器对应每一个请求就创建一个线程的话，在很短的一段时间内就会产生很多创建和销毁线程动作，导致服务器在创建和销毁线程上花费的时间和消耗的系统资源要比花在处理实际的用户请求的时间和资源更多。线程池就是为了尽量减少这种情况的发生。（适用于短时间内多任务的情况，如果线程执行时间较长不适用线程池）

9、用户注册怎么知道谁有 URL 权限

给不同类型的用户设置不同的角色，通过角色匹配权限,其实我们权限控制无非就是控制他的程序的入口 URL,用拦截器做

10、购物车模块 cookie 被禁止了，怎么解决？

URL 重写,对所有页面涉及的连接都使用 url 重写方式。从而将 JsessionID 以参数的方式链接到 URL 后面。保证每次页面提交时服务器都能获得 sessionID 从而维持和客户端的状态。

11、项目中遇到的难点你们是怎么克服的，有什么固定的标准流程吗？

比方说公司要用 redis，买一些关于 redis 实战的书进行学习，网上也找找关于 redis 方面的资料进行学习，遇到一些问题问问我们项目组长，技术总监，他们有什么响应的解决方案，然后自己去解决去。

12、项目迭代是怎么实现？

我们做项目迭代的时候,新增完功能以后会有一个系统集成测试（SIT 测试），测试完了之后再进行上线。

13、第三方支付

我们常用的第三方支付有：支付宝、微信、聚合支付、付钱啦等。他们的原理都差不多。都是在点击支付时，直接调用第三方支付接口，传入 appid、appsecret、订单编号、订单金额、回调 url，直接跳转到第三方支付页面，接下来的支付过程，我们都不需要管，支付成功以后，第三方支付平台会直接回调我们的 url。给我们返回：状态码、订单编号、支付流水号三个参数。我们首先根据订单编号，找到我们的订单，把支付流水号和状态码更新到我们的订单里边。回调 url，一般有两种，一种用同步 get 方法回调，一种用异步的类似 ajax 方法回调，同步方法回调，一般是成功以后才会回调，并且只回调一次，回调成功以后我们可以直接跳转到我们的支付成功页面、异步方法回调，一般要求我们返回一个 success 字符串，第三方平台如果没有接受到 success，就会认为没有调用成功，他会重复多次调用。比如支付宝会在 25 小时之内，调用 8 次；一般情况下第三方支付都采用第二种方式，因为比较安全，但支付宝是同时采用了两种。
我之前接触过一个 B2B 的电商,他们由于交易金额比较大,第三方支付无法实现，所以是直接和银行对接。大体上是，首先平台和银行签订合同，银行为平台开设一个总账号，当企业在平台注册以后，平台会为企业调用银行接口，创建一个子账号，这个子张号是挂在总账号下边的，也是一个在银行实际存在的账号，但是，只能通过外部银行卡给里边转账，而不能给外部银行卡转出。可以在子行号直接互相转账。

14、第三方登录

第三方登录,我的理解就是基于用户在第三方平台上已有的账号和密码来快速完成己方应用的登录或者注册的功能。遵循一个 Oauth2.0 国际通用协议，允许用户在不提供用户名和密码的情况下，让第三方应用访问一些资源。使用第三方登录时，我们不需要用户再次输入用户名和密码，而是直接通过一个唯一 openid 来进行授权登录。对于普通用户来说，如果能用 QQ、微信、百度、新浪这些平台的账号一键注册登录各个平台，无疑会方便很多。对于我们的应用来说，通过授权，借助 QQ、微信这些用户量比较大的第三方平台增强自己的知名度也非常划算。
我们的平台集成了 QQ、微信、百度、新浪四种第三方登录方式，实现的方式都是类似的。首先去各大开放平台进行注册成为开发者，并创建应用，填写回调地址，获取 appid（应用唯一的识别标志）、appkey（给应用分配的密钥），（名称可能不一样）；下载 api 文档和 sdk 开发工具包；就可以开始开发了。
首先在我们网站的登录页面根据 api 集成第三方登录的 logo 图标，并给与点击

事件，当用户点击此图标时，发送请求，直接跳转到第三方平台的登录页面，第三方平台也会自动检测电脑是否有已登录的账号。登录成功以后，第三方平台会自动调用我们传递的回调地址，并传递回一个 code 参数；我们拿到 code 以后，再次调用第三方 api 提供的接口，传入 code、app_id、appkey 等参数，调用获取 access_token 的接口(接口调用，有第三方提供的 sdk 包，直接导入 jar 包，根据 api 文档，传递参数调用方法就可以，我们没必要太过关心第三方平台是用 webservic 接口或 httpclient 接口。)。获取到 access_token 同时，会获取到 openid，拿到 openid 以后，就相当于拿到了登录授权。用 openid 去自己的用户表中查找是否与对应的用户，如果有，就直接查出用户信息，创建自己的 session 就可以了。如果没有，则新创建一个用户，把 openid 放进去。如果还需要其他信息，可以通过 openid 再次调用第三方平台的接口获取用户信息，如果用户信息还是不够，可以创建完用户以后再次跳转一个页面，让用户不全信息。信息补全以后，创建 session，完成登录。这样一个第三方登录就完成了。

15、开发过程中你是如何确保开发的进度和质量的？

一般都是项目经理定的，规定一下项目工作日，根据功能点来估计一下工作日

16、前后端通过接口调用的时候,接口安全如何做的？

我们使用 JWT 做的前后端接口的安全控制,访问我们接口的时候必须通过我们约定好的在 header 中存放 token 信息,判断这个 token 信息是否是我们这个后台给提供的 token 信息,token 信息在什么产生的?是在登录的时候产生的 token 码.还有一种情况,用户初次进入到我们的系统,用户是没有登录的,这时候我们跟前台确定一个 token 码,前台要给我我们 Token 码+时间戳请求我们后台,我们后台会根据约定好了,进行相应的截取时间戳和 token 码,进行相应的判断.

17、你们项目经理主要是做什么的？

(米纳)项目经理是这个产品的第一负责人，管理整个项目的进度和质量，从需求阶段就带着我们和产品经理讨论、分析产品原型的需求，然后指定开发计划并分配任务，在开发过程中经常组织我们开会讨论分析问题，了解任务进展；还要经常与其它部门人员沟通、协调各种问题。

18、那你们前端是和你们一起做的，还是并行的？

我们是并行开发的,我都有接口文档,我们后台根据接口文档进行开发就行,前台的同事也是通过接口文档来对接的,有问题的话我们都在一起的,调试就行了。

19.那数据和页面的融合是谁做的？

前台和我们一块儿配合的做，他们调试前台，我们负责调试以后台。

20.你们做前后端交互的时候都用的什么？

我们交互的时候是异步的 ajax，我记得前端好像用的 vue 或者 racet 框架做的.

21.那你一般都将什么数据进行一些安全处理？你们怎么保证数据的安全性？

涉及到钱的都需要处理，支付宝不是有扫码嘛，先用沙箱环境测试，测试完以后在配置（公钥、私钥、商户 ID 还有其他的一些），配置完再用公司自己写的安全框架，我们用的是 JWT 验证用户的信息，有些功能需要验证用户信息，比如说下单，加入购物车，支付等这些都得验证用户信息.

22.产品经理直接给你原型，那后期有些功能不明确，你怎么处理？

我先自己琢磨琢磨，琢磨完在跟产品经理和项目经理确认一下再进行开发。

23.你主要负责是什么？

商品管理，秒杀、redis 缓存、单点登录。

24.单点登录跟谁单点？

他这个是 soa 架构分布式嘛，那用户在操作时，肯定会访问不同的服务器，要是做 session 共享的话，肯定没有单点登录方便。

那你们的项目经理是你们的还是其他的？

我们的项目经理，这项目组就是我们的，我们的身份还是简铭公司。

25.你们公司的接口是怎么管理的？

那个接口开发，我们就是这么做的，就是用的那个 springMVC 结合 swgger 来

做的那个接口开发，swgger 提供一些注解，可以把接口文档通过网页的形式可以展示出来，上面能标清楚他的 url,参数，返回类型什么的，比较方便，原来我们没有用 swgger 的时候，用的都是那个 word 文档，但是用容易丢，不好维护，后来才用的 swgger,现在同类型的产品也挺多的，我们公司反正用的都是 swgger。

26.pc 端和 app 是不是共享一个接口,为什么

一般得看需求因为 controller 不是一个的有可能 pc 端的是返回页面移动端的是返回 json 数据格式根据实际的需求而定

27.秒杀这一块你们是如何控制超卖的?

我们把库存信息也是存放到了 redis 中,因为 redis 是单线程的,当库存达到 0 的时候我们就直接让商品下架了.

28.你们这个框架怎么搭建的?

我们后台是用最近做的项目是后台用的是 Spring+springMVC+MyBatis 去实现的,中间件用的是 redis,solr,activeMQ,dubbo,这些技术,还有开发工具用的是 Idea

29.你们订单号是怎么组成的?

我们订单号是用时间戳+推特的那个分布式 id 生成器实现的.

30.fastDFS 下载

文件上传后返回一个路径信息和文件名,拿到路径信息我们就可以存到数据库或调用了。

31.跨域是怎么解决的

跨域这块呢，之前我们用过 JSONP 来解决跨域问题，现在项目用到 springMVC 了，它里边有个 @CrossOrigin(origins="http://localhost:9105",allowCredentials="true")
开发时侯跨域调用这块还用到过 HttpClient 这个技术，把它放到工具类里变用的时候直接调用就行了，项目开发中也用到过一些其他这样的技术，像遇到 C 语言向 java 语言这种跨语言的通信，就用到 WebService 技术，它的底层通信用的是 soap 协议，用的是 cxf 方式来发送的。

32.单点登录 session 共享是如何做的？

单点登录：
因为我们的项目是分布式的，我们单点登录使用 cas 做的，用户在 cas 系统中一次登录后，在其他项目中也能访问到该用户的信息

如果不用 cas 的话怎么实现单点登录?
我们可以结合的 redis 一块使用，用户登录-->成功则存入 redis，设置有效期，保存。进入主页面。并向 cookie 发送一个 token 值，当中包含用户信息。用户如果需要执行其他操作，需要携带 token 值去 redis 中进行获取用户数据，验证成功则继续下一步操作，不成功则重新进行登录操作。

33.你们项目是分布式的,那你有了解过分布式事务么？

当然有了，因为我们项目比较大访问用户也比较多，我们在支付的时候，和下单的时候都用到了分布式事务. 比如实时支付吧，一笔支付，是对买家账户进行扣款，同时对卖家账户进行加钱，这些操作必须在一个事务里执行，要么全部成功，要么全部失败。而对于买家账户属于买家中心，对应的是买家数据库，而卖家账户属于卖家中心，对应的是卖家数据库，对不同数据库的操作必然需要引入分布式事务。还有就是用户下单买家在电商平台下单，往往会涉及到两个动作，一个是扣库存，第二个是更新订单状态，库存和订单一般属于不同的数据库，需要使用分布式事务保证数据一致性。我们使用的解决方案是使用支付宝用得那个 TCC 补偿性分布式事务解决方案. 问到什么是 TCC，按照下面 TCC 的原理去说.

(TCC 是什么?)

TCC 是三个英文单词的首字母缩写，分别对应 Try、Confirm 和 Cancel 三种操作，这三种操作的业务含义如下：

Try：预留业务资源
Confirm：确认执行业务操作
Cancel：取消执行业务操作

1、Try：尝试执行业务。
完成所有业务检查(一致性)
预留必须业务资源(准隔离性)
2、Confirm：确认执行业务。
真正执行业务
不做任何业务检查
只使用 Try 阶段预留的业务资源
3、Cancel：取消执行业务
释放 Try 阶段预留的业务资源

TCC 的原理

我给你用这个账务拆分为说一下 TCC 吧，比如说我们账务拆分的业务场景是，分别位于三个不同分库的帐户 A、B、C，A 账户和 B 账户一起向 C 账户转帐共 80 元：

1、Try：尝试执行业务。

完成所有业务检查(一致性)：检查 A、B、C 的帐户状态是否正常，帐户 A 的余额是否不少于 30 元，帐户 B 的余额是否不少于 50 元。
预留必须业务资源(准隔离性)：帐户 A 的冻结金额增加 30 元，帐户 B 的冻结金额增加 50 元,这样就保证不会出现其他并发进程扣减了这两个帐户的余额而导致在后续的真正转帐操作过程中，帐户 A 和 B 的可用余额不够的情况。
2、Confirm：确认执行业务。
真正执行业务：如果 Try 阶段帐户 A、B、C 状态正常，且帐户 A、B 余额够用，则执行帐户 A 给账户 C 转账 30 元、帐户 B 给账户 C 转账 50 元的转帐操作。
不做任何业务检查：这时已经不需要做业务检查，Try 阶段已经完成了业务检查。
只使用 Try 阶段预留的业务资源：只需要使用 Try 阶段帐户 A 和帐户 B 冻结的金额即可。
3、Cancel：取消执行业务
释放 Try 阶段预留的业务资源：如果 Try 阶段部分成功，比如帐户 A 的余额够用，且冻结相应金额成功，帐户 B 的余额不够而冻结失败，则需要对帐户 A 做 Cancel 操作，将帐户 A 被冻结的金额解冻掉。
(TCC 怎么用的)
Github 上有他们的源码,我们直接把源码挡下来,安装到我们本地的仓库里,用的时候我们把需要使用分布式事务的代码,上加上@Compensable 注解,里面还有一些其他的属性配置上就可以了

34.项目中有没有涉及到分库分表,怎么拆分的?(MyCat)

我们用的 MyCat 中间件拆分的,他支持 mysql 集群，或者 mariadb，提供高可用性数据分片集群。我知道拆分数据库的方式一般分为垂直拆分和水平拆分,我们在项目中用的是水平拆分。后端用户可以把 MyCat 它看作是一个数据库代理，用 MySQL 客户端工具和命令行访问。

35.你们项目的边界是什么?

我们公司的项目边界都是架构师和项目经理定的,这个我没参与过.
项目边界其实就是针对整个项目要完成到什么程度的一个定义,就是至少需要哪些个功能点啦,达到什么样的要求,都可以称之为项目边界。红色字体用于理解
理论:在执行项目的过程中，有两次机会定义范围。高端范围在预定义的项目过程中加以定义。这些范围声明有助于建立项目的边界。收集商业需求时，范围定义得越详细越好。如果把范围看作是一个箱子，那么高端范围用来定义箱子的大小和形状；而需求则定义箱子的内容。

36.你的工程是如何操作 MongoDB 的?

我们使用 spring data mongodb

37.在项目的哪些场景下使用 MongoDB？

吐槽 、文章评论 、商品评价.

38.为什么在吐槽和文章评论中使用 Mongodb 而不使用 mysql?

吐槽和评论都是数据量较大且价值较低的数据，为了减轻 mysql 的压力，我们使用 mongodb

39.高并发的接口调用

我们项目中解决高并发都是用的中间件解决的,然后把 redis,ActiveMQ,solr,Freemarker 这些个中间件使用场景都说一遍

40.你怎么理解高可用的?

就是短时间内大量用户访问项目中的某一个功能点,给数据库造成比较大的 IO 压力就是高并发

41. 如何解决高并发?

我在公司采用的，都是通过 solr 全文检索、redis 缓存，mycat 分库分表,把上面中间件的使用场景全部都说一遍..。

二、项目架构和项目部署问题

1.什么是微服务?为什么使用微服务

微服务也是一种分布式的架构,把项目中的每一个模块采用单独的业务逻辑封装，每个项目都可以独立部署和扩展,服务之间可以通过 springCloud 的 Eureka 组件作为注册中心使用 Feign 组件进行服务间的互相通信,我个人认为使用了微服务架构可以降低我们代码的耦合度,在开发的时候,比较好分配项目,还有就是后期再进行项目功能扩展的时候也比较灵活.

2.BS 架构和 CS 架构的区别?

BS 架构就是浏览器和服务器交互　CS 就是客户端和服务器交互

3.简单介绍下分布式

分部式项目就是项目中的模块,把每一个模块拆分成为一个个的工程，这样我们各个模块之间的耦合度不就降低了嘛，然后我们各个模块之间通过 dubbo 框架来相互通讯，我们公司用到的是一个第三方的插件 dubbo 来实现项目之间通信，这样我们的代码的拓展性不就更好了嘛，最后我们在部署项目的时候，可以根据用户的访问量来给各个模块的服务器增大配置，我们还可以采用 nginx 负载均衡来实现一个反向代理，当用户请求后台服务器的时候，有 nginx 来决定哪台服务器空闲或者压力小，来让用户访问。我们这样做的目的不都是为了防止高并发嘛

4.说一下 Docker

用过,Docker 就是为了缩短代码从开发、测试到部署、上线运行的周期，能让项目具备可移植性，易于构建，并易于协作。（通俗一点说，Docker 就像一个盒子，里面可以装很多物件，如果需要这些物件的可以直接将该大盒子拿走，而不需要从该盒子中一件件的取。比如说我们可以在这个容器里装好 zookeeper,redis,mysql,tomcat 等软件,用的时候直接用就可以,项目部署的时候,直接把当前的 Docker 给测试组就可以,或者是运维项目组就行.）

5.用过 Junit 的断言吗?

我们公司有要求做单元测试,但是都是后期补的,有的补全了,有的没补全,主要是我们公司里没有 QA,所以这一块要求的不严， 其实断言就是断定结果的,就是符合我们预期输出的结果就返回 true 不符合返回 false 给出错误信息

@Before	初始化方法
@After	释放资源
@Test	测试方法，在这里可以测试期望异常和超时时间
@Ignore	忽略的测试方法
@BeforeClass	针对所有测试，只执行一次，且必须为 static void
@AfterClass	针对所有测试，只执行一次，且必须为 static void
@RunWith	指定测试类使用某个运行器
@Parameters	指定测试类的测试数据集合

@Rule	允许灵活添加或重新定义测试类中的每个测试方法的行为
@FixMethodOrder	指定测试方法的执行顺序

6.什么是 RestFull 风格架构?

就是一中架构风格,可以直接通过 URL 访问 controller,我们前后端分离的项目不都是采用这样的方式架构的么.

7.docker 容器常用的命令

记住常用的创建容器,删除容器,查询所有容器,启动容器命令就行,其他的看看就行.

镜像和容器的区别?

镜像可以理解为 java 类, 容器理解成为 java 中的对象.

命令总结

- 启动 docker 命令: systemctl start docker 停止 stop 状态 status
- 查看镜像文件命令: docker images
- 搜索镜像: docker search 搜索名字
- 拉取镜像: docker pull 镜像名字
- 删除镜像: docker rmi 镜像 Id
- 删除所有镜像 docker rmi `docker images -q`
- 创建交互式容器: docker run -it --name=mycentos centos:7 /bin/bash
- 退出容器: exit
- 创建了守护式容器: docker run -di --name=mycentos2 centos:7
- 进入守护容器: docker exec -it mycentos2 /bin/bash
- 查询所有的容器: docker ps -a
- 查询正在运行的容器: docker ps
- 查询最后一次运行的容器 docker ps -l
- 查询已经停止的容器: docker ps -f status=exited
- 启动容器: docker run 容器 id/容器名字
- 文件复制(宿主机与容器文件的复制): docker cp 本地目录 容器名称:目录 docker cp /usr/local mycentos2:/usr/local
- 目录挂载(创建容器的时候,把宿主机的一个文件目录,与容器中的某一个目录进行映射,达到的目的是我们往宿主机的目录中添加了文件,他会直接自动复制到对应的容器目录中,): docker run -di -v /usr/local/myhtml:/usr/local/mh --name=mycentos3 centos:7
- 查询容器中的 ip 地址(因为我们后期会创建多个容器,容器与容器之间

要进行同行,就得依靠这个 IP 地址): docker inspect 容器名称
- 删除容器: docker rm 容器名称
- 删除全部容器: docker rm `docker ps -a -q`

备份和恢复流程
- 备份流程: 容器-->(docker commit youlexuan_nginx mynginx)-->镜像-->tar 压缩包 docker save -o mynginx.tar mynginx
-
- 恢复流程 压缩包--(docker load -i mynginx.tar)--> 镜像--(docker run -di --name=youlexuan_nginx -p 80:80 mynginx)-->容器

8.用过集群吗?

solrCloud 是 solr 提供的分布式搜索方案 solrClund 是基于 solr 和 zookeeper 的分布式搜索方案, 主要思想是使用 zookeeper 作为集群的配置中心 特色功能: 1.集中式的配置信息 2.自动容错 3.近实时搜索 4.查询时自动负载均衡 。SolrCloud结构 需要由多台服务器共同完成索引和搜索任务 实现的思路是将索引数据进行 shard(分片) 拆分, 每个分片由多台的服务器共 同完成。当一个索引或搜索请求过来时会分别从不同的 shard 的服务器中操作 索引。 solrCloud 需要 solr 基于 zookeeper 部署, zookeeper 是一个集群管理软件, 由 solrCloud 需要由多台服务器组成。由 zookeeper 来进行协调管理

9.项目中共部署了几个项目

我们的项目采用分布式的架构, 每一个模块都可以作为一个项目, 每一个项目都可以分为服务端和消费端, 按照其功能分可以分为 7 个, 分别是 e3-web-manager,cart,item,portal,Search,sso,order.这几个项目组成

10.你们项目有几个 war 包啊?

就以我们最近这个电商项目来说吧,这个项目最终上线的时候差不多有 8 个 war 包商品管理,搜索,用户,平台后台,商家后台,门户网站,购物车,还有一个秒杀

11.Tomcat 目录, 各个文件夹都放些什么

我知道 bin 目录里面放的是去那些启动命令,conf 里放的是配置文件,部署的项目都是在 webapps 这个目录里,还有 log 日志目录啥的.

三、项目模块描述

3.1【秒杀模块】

我们组还负责这个商品秒杀，秒杀这一块我们是这样来做的，当时设计了两张表，一张是存商品表，另一张是存订单表，因为这个秒杀用户的并发量会非常大，所以我们用 redis 结合的来做的。同样，秒杀商品需要商家将这个秒杀商品的开始时间、结束时间、原价格、秒杀价格、库存等信息提交给运营商进行审核，审核通过就开始秒杀，秒杀开始我们首先把数据库中所有秒杀的商品查询出来并将其存入 redis 中并进行页面的相关展示，当顾客过来秒杀时候我们减库存，而且我们这个秒杀的规则就是只能秒杀一件，当顾客提交订单，我们就将订单的 userID 作为 key，订单信息作为 value，存入 redis 中，并且将 redis 中对应商品的库存量减 1，并且规定顾客下单后 5 分钟内必须支付，如果 5 分钟内支付成功，将订单存入数据库中并将 redis 中的订单删除；如果 5 分钟内没有支付，就会提示支付超时，并且删除 redis 中对应的订单，对相应的商品库存进行回滚+1，然后对这个微信支付进行关闭，在微信关闭的时候，有时候顾客可能在关闭的一瞬间进行支付，所以有时返回的状态码不是 SUCCESS，可能是这个 order_id,那么是这个状态码我们也提示支付成功，并将订单存入数据库，为什么呢？，因为这个支付成功的订单是不能关闭的。接下来还有这个秒杀数据的更新，我们用 spring 中的 Task 这个轮询机制，类似这个 timer 定时器，定时的做这个商品增量更新，什么是增量的更新？就是在我们这个秒杀的过程中，可能商家会添加新的秒杀商品，所以就需要我们定时的去查询数据库看是否有添加新商品，当时我们项目经理规定的是每 5 分钟进行一次轮询，查数据库看是否有商品在 redis 中没有，如果有那我们就将其加进 redis 中，做一个增量的更新；还有就是商品的删除，在秒杀过程中，可能某些商品比较热销，很快就会售罄，那么售罄了的商品我们就不能让她继续展示，否则影响用户体验，同样我们采用轮询机制，每个 2 分钟对 redis 中商品进行查询，如果有这个库存为 0 的商品，我们就将其删除，保证客户秒杀到的商品都是可以秒杀的。

3.2【商品管理模块】

您好！我负责的是商品管理模块，这个模块呢，我们做的时候有这么几张表，
tb_goods（spu 表），字段：商家 id，状态，三个类目 id，类目 id，品牌
tb_goods_desc(商品描述表)，
tb_item(sku 表)，
tb_item_cat（商品类目表）
tb_specification(规格表)，
tb_specification_option（规格项表），
tb_brand(品牌表)，
tb_type_template（模板表）
这个项目是属于 b2b2c 的，分为运营商商品管理和商家商品管理。

【运营商商品管理】：

商品的查询，审核，删除功能。

【商家商品管理：】

有基本的 crud 操作功能，分页用的是 MyBatis 的一个插件 PageHelper，批量删除，修改，添加，在添加和修改这块有个图片上传，分布式开发么，我们用的是 FastDFS 服务器做的。

还有就是，商品添加的时候，我们是先把商品信息添加到 spu 表中，然后根据商品类别来找到对应的模板，接着我们将商品品牌，分类，规格，规格项设计成 json 格式，直接存储在 sku 表中，举个简单的例子手机对应的模板有外观，外观下的分类有颜色，尺寸，提交的时候都是以 json 的形式存储的，我们用这样的方式来解决一对多关系关联的，查询的时候更加方便，也方便了前后台数据的的传输，因为我们在表现层用的是@RestController 进行 json 数据的传输，前台用 AngularJs 发送 Ajax 请求进行数据传输。

我们在添加商品介绍这块还做了一个功能点就是用到了富文本编辑器 KindEditor，做这个的时候我们是在页面中添加富文本编辑器的 JS 代码，这个主要是在前台进行操作的，参考 Api 相当简单。

【这块自己可以先不介绍，可以让面试官来反问你】

FastDFS 服务器搭建比较复杂，是搭建在 linux 系统上的，这个是我配合我们组长参考文档搭建的，当时用了差不多两个多小时吧。

FastDFS 服务器也属于一个中间件，我们是结合 Nginx 实现负载均衡的，里面配置了 TrackerServer 服务和 StorageServer 服务。用 TrackerServer 来管理 StorageServer，StorageServer 不定时的将自己的状态传输给 TrackerServer，是空闲还是占用状态，那么，我们在调用上传方法时，由 TrackerServer 来支配服务，将图片存储在 StorageServer 服务器中，返回一个图片的路径，完成图片上传。

在调用图片上传方法这一块，我们是弄了一个工具类 FastDFSClient，里面封装了一些个操作文件上传下载的方法，再添加修改的时候直接调用就可以了。

3.3【门户管理模块】

门户展示这块比较简单，他就是一个首页广告位，还有大广告位，小广告位，还有那个前后的楼层广告，我们都是用 redis 实现的，因为他的首页访问的人数比较多，压力比较大，通过 redis 去给他缓存一下数据，大家访问的时候都从 redis 中读取数据，而不是从数据库中读取，这样给客户的体验度比较好，还有这个楼层广告位展示的时候，对应的后台有个 cms 系统，这块也是我们做的，cms 系统这块主要针对他的广告类别，比如说楼层广告，二楼广告，以及别的页面的广告，详情页广告，这些都有 cms 系统进行支持，然后我们根据类别去添加相应的广告内容去，把我们的一些图片信息，url 链接添加进去，那边保存完之后，显示到对应的地方去，显示到对应的地方后，在页面中就有相应的展示，这是我们门户网站首页楼层广告位的展示

3.4【搜索模块】

我负责的是商品搜索模块，商品搜索是我们用的 solr 技术实现的，当时也考虑过用 elasticsearch，最后组长决定用的 solr 我们就用 solr 了，使用 solr 我就去学了一下 solr，其实也不难，它的前身就是 lucence,但是 lucence 现在用的人不多了，大部分人都是用的 solr 去做的，使用 solr 呢就是事先定义好一些域，我们把商品从数据库当中把商品信息都添加到索

引库中，我们在搜索是都是从索引库读取的，调用 solr 时是使用 springdatasolr，它的底层也是封装的 solrJ。用这个去调用的 solr 索引库的东西。Solr 中里面有高亮显示啊，分页啊，还有设置了复制域啊，索引域啊，ik 分词器啊，还有在项目里面我们涉及到了自定义的分词，springdatesolr 里面可以自定义分词，像字典里没有的词都可以通过自定义分词来进行设置。我们开发时是在 window 系统上开发的，后期我们在部署的时候会部署在 linux 系统上。
因为后台我们会经常添加商品，它涉及到同步的问题，还有静态页面它也涉及到同步问题，我们使用了 activemq 消息队列来实现的同步商品信息和索引库。生成静态页面就是在我们运营商审核商品通过的时候，我们就发送个消息，发送到我们的 pageservice 项目里面，因为我们的项目时分布式吗，在 pageservice 里面我们配置了 activemq 的监听，在里面把这个消息获取到，这个消息的内容就是商品的 id，我们根据这个商品的 id 生成相应的静态页面。同时我们还有一个索引的项目 search 项目，我们获取到这个商品 id 后去查询数据库，把这个商品的信息给它放到索引库当中去。这个 activemq 就是在这些地方用到过，activemq 就是一个消息队列机制，其实市面上同类型的产品也有很多，像 rebbitmq,rocketmq 等等。这个 activemq 好处就是遵循 jms 协议，我们在用的时候就是用的 spring 给我们封装好的 jms，我们进行调用就行了。。

3.5【单点登录】

单点登录这块使用的是 cas,cas 它是耶鲁大学一个学生创建的开源项目。用起来也比较方便，cas 支持好几种语言，像 java .Net, PHP, Perl, Apache 等，这些语言也都支持，用的话直接拿过来配置，配置的话主要配置一些 xml 文件，首页等这些功能都进行配置，配置完以后，单点登录的话直接用这个系统进行登录就可以了。

我们用的时候结合 security 来使用，做的是门户网站的登录。还有就是注册这个功能，用户再注册的时候不需要再添加繁琐的信息，我们只需要用户添加一个电话号码，短信验证码，密码就可以了， 短信验证码我们用到的是阿里大于的短信验证，调用短信验证码，发送一条短信，进人注册，然后用・户就可以直接进行登录了。

3.6【订单模块】

提交订单这块儿，我们是把 redis 中的信息拿出来，然后进行一下查询，还有就是我们会有满多少可以有一个优惠券，当用户选中优惠券的时候，后台会进行一个相应的判断，然后进行满减，提交完订单之后，还有一个功能，就是支付这块，我们组做了一个扫码支付，就是一个微信扫码支付，其他的支付没来的急做，最后交给其他的组了，微信支付的话其实也不难实现，微信支付这块儿我们公司是申请了一个微信的公众号，就是什么公众号 ID，appID 还有它的一个密钥，支付的时候就是把密钥，还有随机数，还有订单号这些都给他发送过去，还有金额都发送过去，如果成功的话让他返回相应的信息，告诉你支付成功，还会给你返回一个微信的流水号，可以把微信的流水号存入数据库中去，做微信支付的时候他有一个查询订单的功能，查询它是否支付成功，我们当时做的时候，让用户进行扫码，如果支付成功的话，就进行页面跳转，提示用户支付成功了，查询的时候把订单号给他发送过去，这个流水号就是用于之后用户的一个退款用的。微信支付这块大概就这么多了。

3.7【商品详情展示】

我们组还负责这个商品详情展示模块，在这里主要是用到了这个 freemarker 技术，因为我们做的是 B2B2C 项目嘛，所以当这个商家新增了商品是需要通过这个运营商审核商品信息的合法性，商品才能上架，在这个运营商审核通过商品时，我们用到这个 freemarker 技术，通过制定 freemarker 模板将这个商品的动态信息生成一个静态页面并以该商品的（SPU）id 来命名的，使用 freemarker 技术主要是为了程序运行效率。当前台进行展示时，只需要调用这个静态页面进行展示即可。因为我们是分模块开发的嘛，这个商品详情和商品管理不在一个模块中，所以我们用到这个消息队列的技术，现在比较流行的这种消息队列有：rabbitMQ、RacktMQ、ActiveMQ 等，我们采用的是 ActiveMQ 技术，activeMQ 有这种两种方式点对点和这个订阅式的，我们采用的是这种订阅式的方式，具体呢，就是在这个商品管理模块中配置了一个生产者就是当这个商品审核通过时，就会发送者商品的 ID 到消息队列中，在这个生成静态页面的模块中配置了这个监听，那边发送了商品 ID 这边就能监听到并且调用 onMessage（）回调函数，通过得到的 ID 来查询商品的信息并加载 freemarker 模板来生成静态详情页面。还有就是商品详情页面点击这个商品规格的时候，这个 title 和图片会对应变化，这块我们是用这个 angularJs 来实现的，商品详情模块大概就是这样。

3.8【购物车模块】

我们负责购物车这模块,我们的购物车模块是参考的京东商城的购物车来做的.就是用户在不登录的情况下也能添加购物车.
用户在没登陆的时候,购物车信息是保存在 cookie 中.当用户添加商品信息到购物车的时候,先判断该商品是否存在,如果存在,就在数量上做添加.如果不存在,就根据 id 值取出商品的信息保存到集合中,把集合添加到到 cookie 中,再把 cookie 返回给浏览器.
当用户登录后，购物车信息保存在 redis 中,这就涉及到了 cookie 跟 redis 的信息同步.我们是这样处理的,
将 cookie 中的商品信息集合,遍历依次添加到 redis 中,添加成功后,删除 cookie 中的信息.然后再查询的时候直接从 redis 中查询就好。
在这涉及到一个同步问题,用户在登录前购物车中添加的信息,在登录后没有及时查看购物车就清空缓存,导致 cookie 信息丢失,同步失败.我们在登录的时候设置拦截,提前将 cookie 中的信息直接取出来,存到 redis 中,避免这一情况的发生.
在购物车这块,我们还涉及到了优惠券、满减活动，当用户查看购物车的时候，先判断商品是否满足使用优惠券的条件，满足的话，将使用优惠券后的价格展示在购物车页面。支付时在后台做相应的运算。

3.9【支付模块】

【第一种话术】：你们商城的支付系统是如何实现的？
我们系统采用微信扫码支付。 首先需要到微信平台申请微信公众账号（ 服务号） ，然后再申请开通微信支付。 开通后会得到 appid、 秘钥等数据， 用于调用微信支付接口。
微信支付接口是通过 httpClient 方式向微信支付平台发送数据并获取结果。
（ 1） 调用统一下单接口获取 url

（2） 根据获取的 url 在页面上通过 qrcode.js 生成二维码
（3） 生成二维码后， 实时轮询查询订单状态， 当返回成功时， 跳转至成功页面

【第二种话术】支付主要流程实现：app 发起支付，服务端接受请求，生成订单号跟第三方参数规定，调用第三方支付平台的下单接口，成功后返回相应支付凭据。把支付凭据返还给 app，app 根据支付凭据信息，在客户端完成支付。如果 支付实际成功后第三方平台会调用我们提供的 http 回调接口进行回调。验证签名比对价格，更新订单状态，通过 ActiveMQ 将支付成功消息异步推送给交易模块。以上就是我负责的模块以及信息，请问您还有没有什么想问的

四、项目开发中如何和 App 交互模块

你好，我在开发中负责了一些模块，比如像 xxx，xxxx(这里替换成自己负责的具体模块)，同时呢，我们项目也开发了 app 版本，方便人们在手机上访问。（或者说我们项目主要是应对移动端客户的，只做了移动端开发，没有做网页版）。在移动端开发 app 的时候，我还主要负责了接口的开发，就是移动端设备，像 android 手机，ios 手机在进行数据获取的时候，我们给手机端提供一个接口访问。

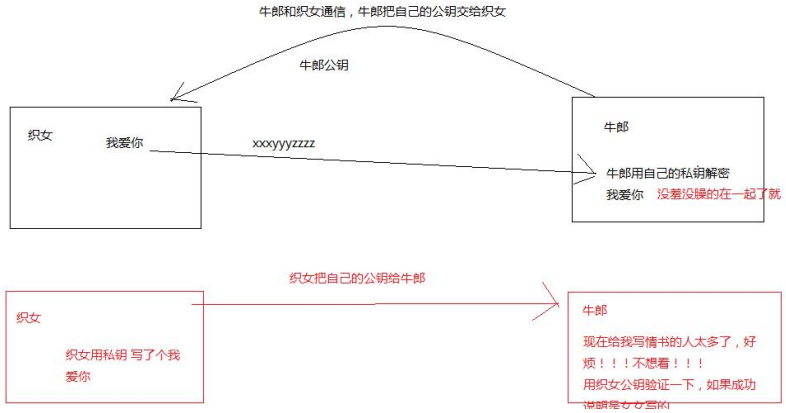
像 app 上有好多信息的展示，比如我们做得是电商 app，手机端所有的商品信息都是从我们的服务器上获取到的，这时候我们就为 app 提供一个获取商品信息的接口，当 app 调用后台接口获取商品信息时，我们会把商品的信息，包括商品详情，规格模板信息，查询出来，封装成 json 类型的数据,将数据输出给 app 端。

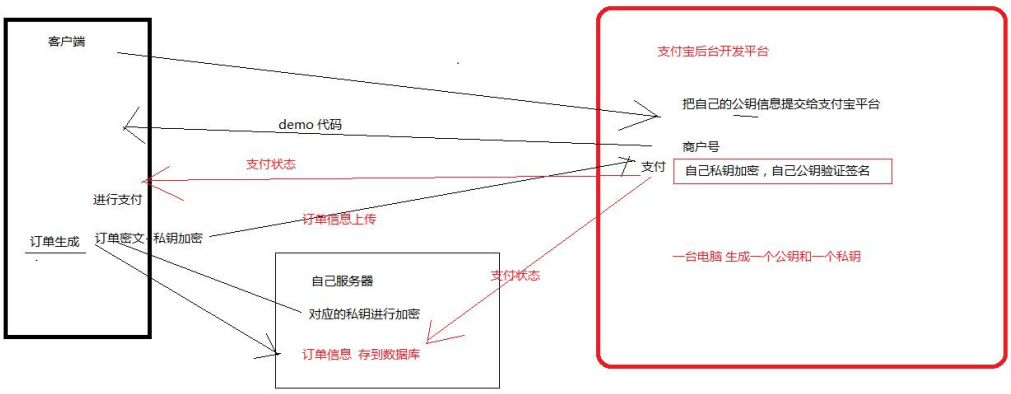
这里我就拿手机的商品搜索做个案例来讲吧，手机上商品的搜索，其实还是搜索的我们服务器中的数据，复杂的业务逻辑还是在后台，app 端只是传递一些请求条件参数。后台我们搜索这块使用的是 solr 服务来进行的，把 solr 那块知识点给它整到这里来说...此处省略 1000 字

这里还要提一下手机支付加密这块，我们在做支付的时候，用到了支付宝，我就说一下手机调用支付宝加密的一个流程吧。支付宝 app 端加密的方式是用到了一

个 RSA 不对称加密算法加密算法。

就 是 公 钥 私 钥 加 密 算 法 ，





这里 app 端进行订单生成支付时，会将订单信息发送给后台，然后后台通过公司服务器电脑生成的加密私钥进行加密，将信息返给移动端，移动端再请求支付宝支付，支付宝会通过我们的私钥验证签名。 了解

五、算法相关面试题

1.JAVA 中常用的加密算法

对于不可逆的加密算法有
MD5 和 SHA， 通过散列算法进行加密 SHA 加密比 MD5 安全性更高，常用 SHA-256 加密算法。

DES 加密算法，对称加密，客户端和服务端公用一个 key，该 key 最好是随机生成，对于这种加密算法加密效率高，但是据说 24 小时以内可以破解。

AES 加密算法，不对称加密算法，通过公钥加密，私钥解密，私钥加密，公钥验证签名。(支付宝) 是目前比较安全的加密算法，但效率偏低。安全性是 DES 加密算法的 1000 多倍数。

2.排序算法？说一下时间复杂度和空间复杂度

冒泡排序：

每次拿两个数进行比较，总是把小的放在最前面，大的放在最后面，第二次是第二个数和第三个数比较，如果第三个小于第二个，那么第三个和第二个替换位置，一次这样比较替换，第一轮循环完了呢，放在最后面的那个数就是这 一串数字中最大的数。第二轮循环和第一轮一样，第一个和第二个比较，最小的放在最前面，依次类推。如果有 10 个数字，那么就进行 9 次循环，最小的在最前面，最大的放在最后面。
好处呢，就是查询速度快

二分查找：

二分查找就是从一串数字中找到某个数字。
原理就是必须有一串数字是从小到大排序，把这串数字进行划分，分为三段，前半段，中止段，和中止后半段。查找数字或者字符先从中止段查找，查到的数字或者字符和中止正好相等，那么久直接取出来，如果比它大就从后半段查找，后半段在进行二分法，进行递归的调用查找，如果比中止小，走前面的中止段，在进行拆分，进行二分查找法。

3. 什么是时间复杂度和空间复杂度?

六、23 种设计模式

1.单例和工厂是如何实现的?

单例：其实单例本质的就是控制对象的产出，要想产生一个唯一的实例，你就要私有化构造方法，然后私有化一个的静态的全局变量，并且提供一个公共的静态的方法供外界访问，单例大概就是这样的。
工厂：它是 java 设计模式的一种，工厂模式的优势还是很多的，我

举一个简单的共厂的例子吧，就是假如我们有好多接口，好多的接口的实现类，我们如果使用工厂模式的话，当我们有需求需要改变接口的实现类的话，直接在工厂里面改变返回的类型就好。这样会应对不同的需求变化。

2.你在项目哪些地方用过设计模式

七、开发工具使用

1.你说一下你们项目中的 maven 工具是怎么用的?

Maven 简介及优点：maven 是一个项目管理工具，其核心特点就是通过 maven 可以
进行包的依赖管理，保证 jar 包版本的一致性，以及可以使多个项目共享 jar 包，从
而能够在开发大型 j2ee 应用的时候，减小项目的大小，并且和 ant 比起来，maven
根据“约定优于配置”的特性，可以对其项目的编译打包部署进行了更为抽象的封
装，使得自己不需要像 ant 那样进行详细配置文件的编写
Maven 常用命令：maven -v 查看版本号
Maven package 项目打包工具 Maven test 测试 Maven install 模板安装，将打包的 jar/war 复制到本地仓库中 Maven deploy 发布命令 Maven clean 删除

2.平常用什么开发工具

IDEA,Eclipse,MyEclipse,数据库操作用 nvacat 或者 sqlyong,还有数据库建模用 powerDesner

2. SVN 使用流程

我每天早上过去先下载下最新的代码,我写完了要提交的话,就是先同步,再提交,有时候也会遇到冲突的时候. 冲突怎么解决的? 我是先把自己的代码备份一份,然后把服务器上最新的版本代码下载下来,然后把我新增的代码方进行,再提交就行.

4.Git 使用流程

我们项目经理给创建的分支,我们直接在分支上开发就行,写完以后我们提交到分支上,项目经理去合并去,这个我们不用管.

5.git 常用命令有哪些?

Git 他是个分布式的版本控制工具，我觉的用他的好处就是可以创建分支，在分支上进行开发，解决单点故障，我们公司在用的时候都是每个项目组会创建一个分支，都是在分支上进行开发，开发完测试没啥问题了，再由项目经理进行下分支合并就行。 常用命令：git clone， git add git commit 啥的 上传到远程服务器就是 push 从远程服务器下载代码就是 pull。别的也没啥了。

6.你们 git 服务器是用的 gitlab 还是 github 还是本地的?

我们公司搭建的是 Gitlab 做的远程服务器端的.

7.maven 的 jar 包冲突怎么解决

八.介绍一下你们图片服务器

(What FastDFS 是什么)

我们是搭建的 FastDFS 服务器，是 c 语言编写的一款开源的分布式文件系统。FastDFS 服务器对冗余备份、负载均衡、做的非常好，使用 FastDFS 下载上传不管是速度和性能上，都不错，现在应该是大部分公司都使用 FastDFS 来搭建图片服务器。

FastDFS 架构包括 一个 Tracker server 和一个 Storage server。客户端请求 Tracker server 进行文件上传、下载，通过 Tracker server 调用 Storage server 完成文件上传和下载。

Tracker server 可以完成负载均衡，它其实是这样的，这个 storage server 会定时的向 tracker server 发送自己的状态，tracker server 就能知道那个 storage server 是空闲还是忙碌，当上传下载的时候，就会调用空闲的 storage server，当上传或者下载的时候，这个 storage server 会生成 file id,并将文件写到磁盘.返回一个路径信息和文件名，拿到路径信息我们就可以存到数据库或者调用了。

(Why 为什么用 FastDFS)

图片服务器，这块，如果做 demo 我们可以在 tomcat 上配置虚拟路径，来专门搞一个文件的文件夹来放置图片,但是当项目大了，有多个服务器时，图片调用就会有一个问题，所以呢，我们会专门用一台服务器来放置文件

(How 怎么使用 FastDFS)

首先配置 fastDFS 服务器
这个一般进公司，多数公司都有一台配置好的 FastDFS 服务器。这个配置之前有配过，也是在 linux 上配置的，我们当时用的是 centos,

解压 tar 包
编译安装
配置 tracker server
配置 storage server

一般情况下，fastDFS 还会结合 nginx 来做，因为 FastDFS 的 HTTP 服务较为简单，无法提供负载均衡等高性能的服务，那就需要配置 nginx 来完成这个负载均衡。

这个 fastDFS 代码操作的话比较简单就是需要 pom 文件中配置 jar 包信息

代码中创建 traker server 和 storage server 对象，搞一个 client(storage client)的对象，调用上传方法。这里上传方法我们在做的时候，是配置一个工具类，执行完上传方法时，会获取到图片在服务器上的位置。

九.说一下 FreeMarker 这个技术吧

(What FreeMarker 是什么)

FreeMarker 呢，它是一个用 Java 语言编写的模板引擎，它可以根据一个模板生成一个文本，具体来定位的话，FreeMarker 应该算是文本生成的一个工具。
它的工作原理是这样的:

咱们先定义一个模板文件，比如我们常写成 ftl 的形式，可以是 jsp 或者 html 文件改写来的，freemarker 这个引擎，有一些自己的指令，比如判断语法的话也是 if else，只不过 else 是套在 if 里边的

比如 <#if 条件>执行 if 条件操作<#else>执行非 if 条件操作</#if>
遍历的话使用的是#list。然后为生成静态模板准备数据，这样数据+模板就可以生成一个静态模板了。

(Why 为什么要用到 FreeMarker)

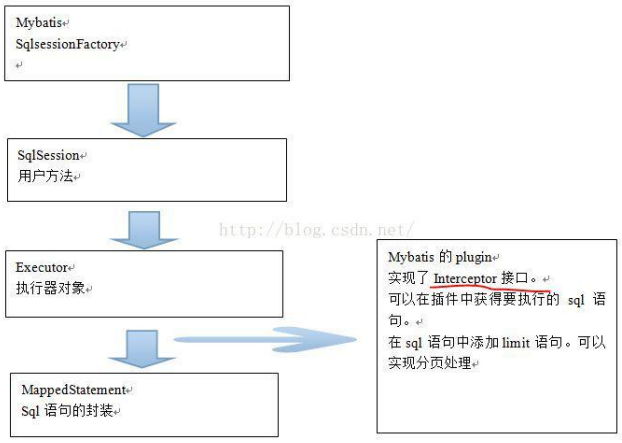
Freemarker 这种技术优点就是可以而分离表现层和业务层逻辑，模板只负责数据展示类型，不涉及任何逻辑代码，这样分工可以更明确一些。并且 freeMarker 由于生成的是静态界面，访问速度特别快

(How FreeMarker 怎么用)

我们项目中，商品详情界面使使用的 FreeMarker 来做的，商品添加的时候，直接把静态界面给生成，比如一个商品，一个 id，一个 id 对应一个详情界面，然后存到服务器上，当点击商品详情的时候，直接根据 id 去服务器上查找自己的详情界面就好了。

当然我一开始就提过，我们在做的时候，是分模块的开发的，做了分布式，生成静态模板的时候，是通过 ActiveMQ 发送消息，同步生成静态界面。（同学，你又可以扯扯 ActiveMQ 啦）

十.讲一下你们项目 PageHelper 实现原理



PageHelper 是 MyBatis 的一个分页插件，这个在做分页的时候，效率比较高，只需要在 MyBatis 的配置文件中配置一个插件就可以

MyBatis 的 plugin 实现了 Interceptor 接口，可以在插件中获得要执行的 sql 语句，那当拦截

到 sql 语句的时候，如果使用 PageHelper 设置了分页的条件，那我们就可以在 sql 语句后边拼接上 limit，就能执行分页操作了。

对于 PageHelper 中具体的拦截位置，

需要先讲一下 MyBatis 的执行流程，MyBatis 在执行 sql 操作的时候，先读取 MyBatis 的配置文件 SqlMapConfig.xml 这个配置文件中我们一般配置连接池，事务管理，数据库环境的配置，别名等等环境,当然，当我们整合 Spring 之后，这些配置信息多数都交给 Spring 进行管理。

通过读取配置文件信息，会获取到我们配置的 sqlSession 工厂类，通过这个工厂类可以获取到 MyBatis 的一个会话，这个 sqlSession 是 MyBatis 的一个接口，面向用户的，用户通过 sqlSession 操作数据库，有一个 Executor 执行器，来执行数据库操作。

当准备执行的时候，就在这里进行拦截，获取到要执行的 sql，然后进行拼接

拼接完 sql，就会执行相应的查询操作。

11.数据库 SQL 这块公司怎么处理

http://blog.csdn.net/zhiyuan0932/article/details/68978666

在公司开发中，对于 sql 我们一般是自己写,像基本的增删改查，模糊查询，排序，分页,单表，多表，常用的多表连接方式，内连接，左外连接，右外连接，笛卡尔积查询都比较数据。甚至是四五上表的联查，这些都没有问题。

12.数据库常用性能优化

设置引擎方式 innodb

数据库性能优化这块，我们考虑比较多的还是查询这块，互联网项目对数据查询非常频繁，对效率，性能要求比较高。

查询这块优化的话，主要就需要使用索引这种方式，所谓索引就是建立一种快速查找的方式，比如我们查字典,有一个 ABCD 的索引.

举个例子,如果我们创建一个表 create table user(id integer ,name varchar(20)，job varchar(20))；如果我们数据库中有 1000 万条数据，当我查询 select * from user where name=’张三’的时候,这种查询方式就类似于整个数据库的扫描，效率非常低。

我们可以给这个 name 设置一个索引 create index n on user (name)；这是设置一种普通（normal）索引，然后我们查询的时候，有了这个索引，效率就会大大提升，当然对于索引，它的方式有 BTree 类型和 Hash 类型，是两种管理数据库索引的方式，这个我没有深入研究。这个我们可以自己设置。默认是 btree。

索引类型的话，有 normal 类型，还有主键索引，还有非空索引，还有聚集索引。

主键索引，primary key 在设置的时候，已经指定了，其实也是非空索引。

非空索引是 not null,设置这种方式的该字段下内容不能为空，

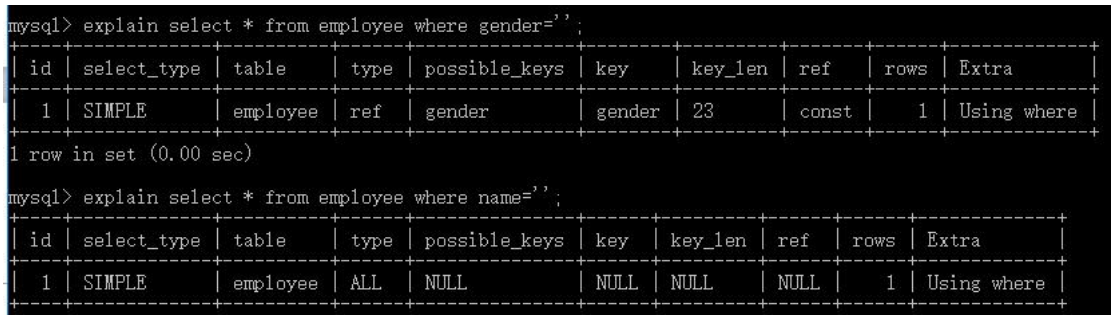
聚集索引，是在设置多个查询条件的时候使用。比如 创建一张表，有名字，有工作,我们想经常频繁的用到名字和工作它俩结合在一起来查询数据库中表的数据。这个时候，可以将名字和工作指定为聚集索引。create index m on user(name,job)；这样当我们指定 select * from user where name=xxx and job=xxx 的时候，就会按照索引方式来做。

这种优化方式就是索引优化，在使用索引优化方案的时候，我们需要注意避免在索引字段上使用条件函数等操作。

当然在使用 sql 的时候，还有好多注意点，比如尽量不用 in not in 这种 sql 语句。

13.数据库性能检测方式

在设计 SQL 的时候，我们一般会使用 explain https://segmentfault.com/a/1190000008131735 检测 sql，看是否使用到索引，避免出现整表搜索方式查询[filesort(不是以索引方式的检索，我们叫做 filesort)]（我在这张表中把 gender 设置成 normal 索引，name 没有任何设置）



```
mysql> explain select * from employee where gender='';
+----+-------------+----------+------+---------------+--------+---------+-------+------+-------------+
| id | select_type | table    | type | possible_keys | key    | key_len | ref   | rows | Extra       |
+----+-------------+----------+------+---------------+--------+---------+-------+------+-------------+
|  1 | SIMPLE      | employee | ref  | gender        | gender | 23      | const |    1 | Using where |
+----+-------------+----------+------+---------------+--------+---------+-------+------+-------------+
1 row in set (0.00 sec)

mysql> explain select * from employee where name='';
+----+-------------+----------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+----------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | employee | ALL  | NULL          | NULL | NULL    | NULL |    1 | Using where |
+----+-------------+----------+------+---------------+------+---------+------+------+-------------+
```

对比看的，对有索引的字段,在检测的时候，会显示是一个引用的 key。

还可以使用 profiling 方式检测数据库执行的方式，可以查询 sql 的运行时间。http://www.jb51.net/article/31870.htm

show variables like '%profiling%';(查看 profiling 信息)

select goods_name from ecs_goods where goods_id <5000

set profiling=1;（开启 profiling）

show profiles;

第九页有做修改

14.关于日志处理

日志处理我们使用的是 log4j,有一个 log4j 的配置文件,可以配置 log 输出的位置以及 log 的输出形式，并指定内容拼接方式。

对于整个项目，设置了一个全局异常，当出现异常信息的时候，将异常信息记录到 log 中

```
Logger logger = LoggerFactory.getLogger(GloableException.class);
logger.error("----------出错了----------");
```

当有些需要记录内容的信息，也可以通过日志文件进行记录。

对于用户登陆日志记录,我们需要自己封装一个日志记录的工具类,可以将用户登陆的信息记录到数据库中。(具体操作步骤看如下链接)

http://www.cnblogs.com/marcello/articles/4501479.html

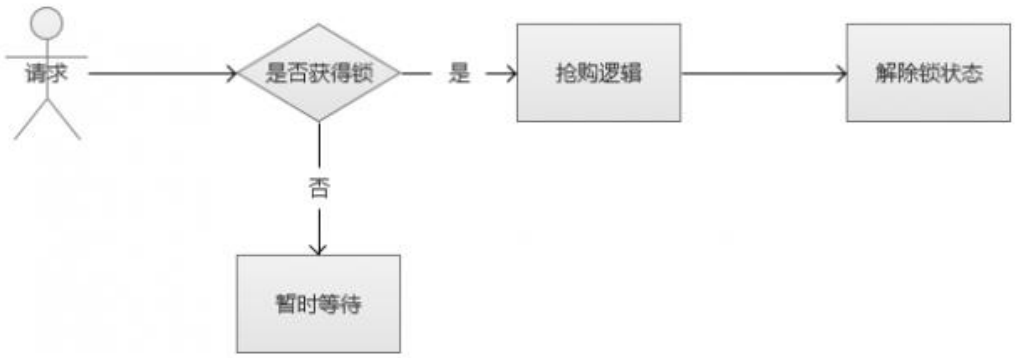
```plain
# 定义 DEBUG 优先级， R 为日志输出目的的

log4j.rootLogger= DEBUG, R

# 设置日志输出类型 ， 为文件类型

log4j.appender.R= org.apache.log4j.FileAppender

# 设置日志文件名 logRecord.log, 输出到 tomcat 服务器的 logs 目录下

log4j.appender.R.file= ../logs/logRecord.log

# 每次在文件尾写入新的日志信息

log4j.appender.R.Append= true

# 日志输出信息格式类型

log4j.appender.R.layout= org.apache.log4j.PatternLayout

# 日志输出信息格式为 换行、日期、优先级、 [ 全类名 ] 、日志信息、换行

log4j.appender.R.layout.ConversionPattern= %n%d%p [%l] %m%n
```

15.关于悲观锁和乐观锁

悲观锁思路

解决线程安全的思路很多，可以从“悲观锁”的方向开始讨论。

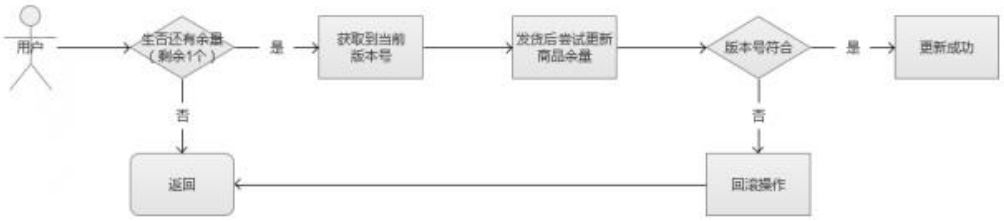
悲观锁，也就是在修改数据的时候，采用锁定状态，排斥外部请求的修改。遇到加锁的状态，就必须等待。



虽然上述的方案的确解决了线程安全的问题，但是，别忘记，我们的场景是“高并发”。也就是说，会很多这样的修改请求，每个请求都需要等待“锁”，某些线程可能永远都没有机会抢到这个“锁”，这种请求就会死在那里。同时，这种请求会很多，瞬间增大系统的平均响应时间，结果是可用连接数被耗尽，系统陷入异常。

乐观锁思路

这个时候，我们就可以讨论一下“乐观锁”的思路了。乐观锁，是相对于“悲观锁”采用更为宽松的加锁机制，大都是采用带版本号（Version）更新。实现就是，这个数据所有请求都有资格去修改，但会获得一个该数据的版本号，只有版本号符合的才能更新成功，其他的返回抢购失败。这样的话，我们就不需要考虑队列的问题，不过，它会增大 CPU 的计算开销。但是，综合来说，这是一个比较好的解决方案。



有很多软件和服务都“乐观锁”功能的支持，例如 Redis 中的 watch 就是其中之一。通过这个实现，我们保证了数据的安全。

16.JAVA 中常用的加密算法

对于不可逆的加密算法有
MD5 和 SHA，通过散列算法进行加密 SHA 加密比 MD5 安全性更高,常用 SHA-256 加密算法。

DES 加密算法，对称加密，客户端和服务端公用一个 key,该 key 最好是随机生成,对于这种加密算法加密效率高,但是据说 24 小时以内可以破解。

AES 加密算法，不对称加密算法，通过公钥加密，私钥解密，私钥加密，公钥验证签名。(支付宝) 是目前比较安全的加密算法，但效率偏低。安全性是 DES 加密算法的 1000 多倍数。

```
public class SHAUtil {

	/***
	 * SHA 加密 生成 40 位 SHA 码
	 *
	 * @param 待加密字符串
	 * @return 返回 40 位 SHA 码
	 */
	public static String shaEncode(String inStr) throws Exception {
		MessageDigest sha = null;
		try {
			sha = MessageDigest.getInstance("SHA");
		} catch (Exception e) {
			System.out.println(e.toString());
			e.printStackTrace();
			return "";
		}

		byte[] byteArray = inStr.getBytes("UTF-8");
		byte[] md5Bytes = sha.digest(byteArray);
		StringBuffer hexValue = new StringBuffer();
		for (int i = 0; i < md5Bytes.length; i++) {
			int val = ((int) md5Bytes[i]) & 0xff;
			if (val < 16) {
				hexValue.append("0");
			}
			hexValue.append(Integer.toHexString(val));
		}
		return hexValue.toString();
	}

}
```

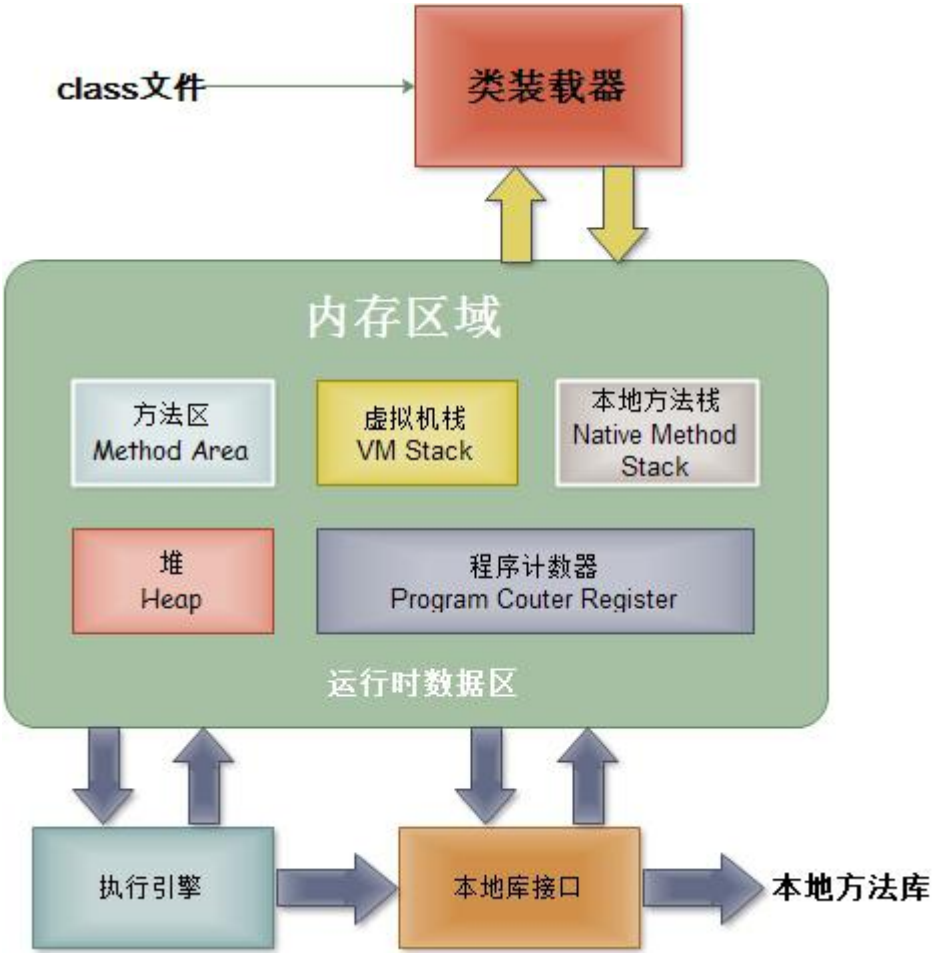
```java
package com.example.encode;


import java.security.SecureRandom;


import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;


public class DESUtil {
	/**
	 * 加密  1234   hahah   ---->dhskdhskhd
	 */
	public static byte[] encrypt(byte[] datasource, String password) {
		try {
			SecureRandom random = new SecureRandom();
			DESKeySpec desKey = new DESKeySpec(password.getBytes());
			// 创建一个密匙工厂，然后用它把 DESKeySpec 转换成
			SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
			SecretKey securekey = keyFactory.generateSecret(desKey);
			// Cipher 对象实际完成加密操作
			Cipher cipher = Cipher.getInstance("DES");
			// 用密匙初始化 Cipher 对象
			cipher.init(Cipher.ENCRYPT_MODE, securekey, random);
			// 现在，获取数据并加密
			// 正式执行加密操作
			return cipher.doFinal(datasource);
		} catch (Throwable e) {
			e.printStackTrace();
		}
		return null;
	}
	/**
	 * 解密  dhskdhskhd  hahaa---->1234
	 */
	public static byte[] decrypt(byte[] src, String password) throws Exception {
		// DES 算法要求有一个可信任的随机数源
		SecureRandom random = new SecureRandom();
		// 创建一个 DESKeySpec 对象
		DESKeySpec desKey = new DESKeySpec(password.getBytes());
		// 创建一个密匙工厂
		SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
```

```
            // 将 DESKeySpec 对象转换成 SecretKey 对象
            SecretKey securekey = keyFactory.generateSecret(desKey);
            // Cipher 对象实际完成解密操作
            Cipher cipher = Cipher.getInstance("DES");
            // 用密匙初始化 Cipher 对象
            cipher.init(Cipher.DECRYPT_MODE, securekey, random);
            // 真正开始解密操作
            return cipher.doFinal(src);
        }
}
```

17.Jvm 虚拟机原理

Java 虚拟机(Jvm)是可运行 Java 代码的假想计算机 Java 虚拟机包括一套字节码指令集、一组寄存器、一个栈、一个垃圾回收堆和一个存储方法域。我们都知道 Java 源文件，通过编译器，能够生产相应的.Class 文件，也就是字节码文件，而字节码文件又通过 Java 虚拟机中的解释器，也就是前面所有的 Java 虚拟机中的字节码指令集….编译成特定机器上的机器码 1.Java 源文件—->编译器—->字节码文件

2.字节码文件—->Jvm—->机器码 每一种平台的解释器是不同的，但是实现的虚拟机是相同的。这也就是 Java 为什么能够跨平台的原因了 当一个程序从开始运行一个程序，这时虚拟机就开始实例化了。多个程序启动就会存在多个虚拟机实例。程序退出或者关闭。则虚拟机实例消亡。多个虚拟机实例之间数据不能共享。



垃圾回收器（又称为 gc）：是负责回收内存中无用的对象（好像地球人都知道），就是这些对象没有任何引用了，它就会被视为：垃圾，也就被干掉了。虚拟机内存或者 Jvm 内存，冲整个计算机内存中开辟一块内存存储 Jvm 需要用到的对象，变量等，运行区数据有分很多小区，分别为：方法区，虚拟机栈，本地方法栈，堆，程序计数器

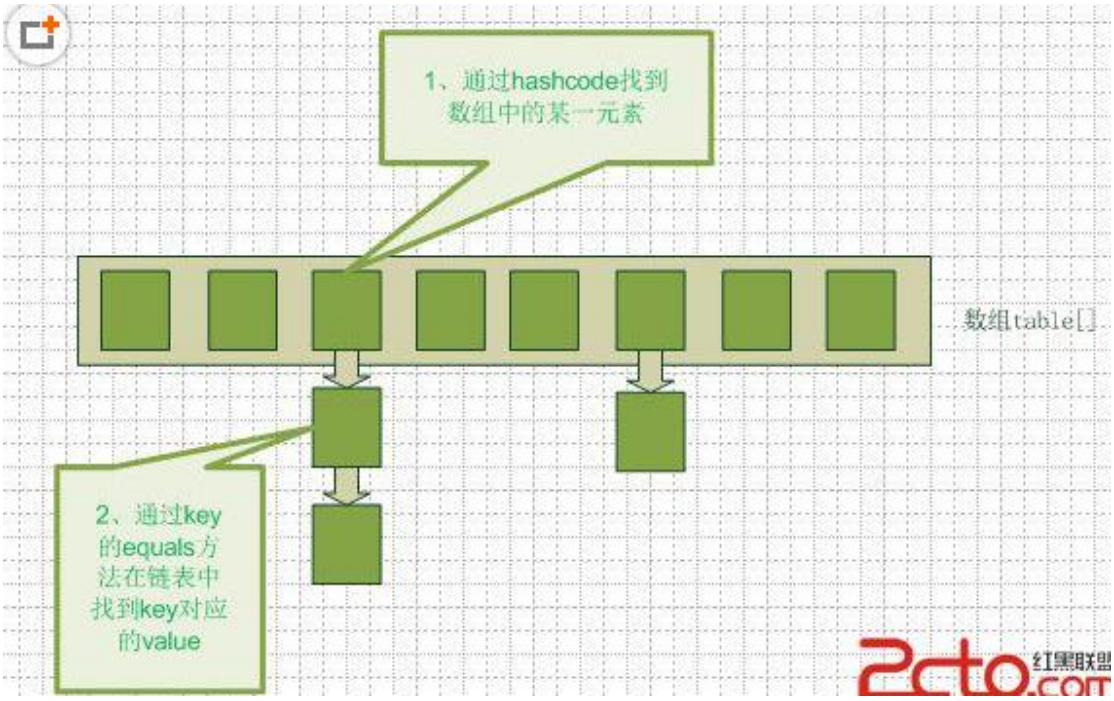
1. 程序计数器 当前线程执行字节码的信号指示器，线程是私有的，它的生命周期和线程相同分支、循环、跳转、异常处理、线程恢复等基础功能都需要依赖这个计数器来完成。2. 虚拟机栈 Java 虚拟机栈描述的是 Java 方法（区别于 native 的本地方法）执行的内存模型：每个方法被执行的时候都会同时创建一个栈帧（Stack Frame）用于存储局部变量表、操作栈、动作链接、方法出口等信息。线程私有，生命周期和线程相同，都有各个独立的计数器，各不影响。每个方法被调用直至执行完成的过程，就对应着一个栈帧在虚拟机栈中从入栈到出栈的过程。本地方法栈 和虚拟机方法栈差不多类似，但是本地方法栈是服务于虚拟机所使用到的 Native 方法服务。本地方法区： 只是执行 Native 方法。如果这个区的内存不足

也是会抛出 StackOverflowError 和 OutOfMemoryError 异常。堆这块区域是 Jvm 中最大的，应用的对象和数据都是存在这个区域。这块区域也是线程共享的。也是 gc 主要的回收区。

18.HashMap 底层原理

HashMap 底层数据结构是链表散列结构,就是链表和数组的结合体。
当新建一个 HashMap 的时候，会初始化一个数组，这个数组是一个 Entry[],entry 数组中存放到是 key value 键值对，每一个 entry 都持有下一个元素的引用。

当储存一个 entry 时，会根据 hash 算法计算出 key 的 hash 值，来决定 entry 的存储位置，当取出一个 entry 时，也是根据 hash 计算出要取得 key 的 hash 值，根据 equals 从 entry 数组中取出对应的 entry



19.你们公司的 double 注册中心用的什么？

用的 zookeeper,我们开发的时候用的是 1 台服务器，然后上线的时候用的是 3 台。

20.项目中有没有涉及到分库分表,怎么么拆分的?(MyCat)

MyCat 一个新颖的数据库中间件产品支持 mysql 集群，或者 mariadb cluster，提供高可用性数据分片集群。MyCat 分片根据其切分规则的类型，分为垂直切分和水平切分我们在项目中用的是水平切分。前端用户可以把它看作是一个数据库代理，用 MySQL 客户端工具和命令行访问，而其后端可以用 MySQL 原生协议与多个 MySQL 服务器通信，也可以用 JDBC 协议与大多数主流数据库服务器通信，其核心功能是分表分库，即将一个大表水平分割为 N 个小表，存储在后端 MySQL 服务器里或者其他数据库里。
它支持 MySQL、SQL Server、Oracle、DB2、PostgreSQL 等主流数据库，也支持 MongoDB 这种新型 NoSQL 方式的存储，当我们的应用只需要一台数据库服务器的时候我们并不需要 Mycat，而如果你需要分库甚至分表，这时候应用要面对很多个数据库的时候，就需要对数据库层做一个抽象，来管理这些数据库，而最上面的应用只需要面对一个数据库层的抽象或者说数据库中间件就好了，这就是 Mycat 的核心作用。
所以也可以这样理解：数据库是对底层存储文件的抽象，而 Mycat 是对数据库的抽象。

21.说一下 Docker

用过,Docker 就是为了缩短代码从开发、测试到部署、上线运行的周期，能让项目具备可移植性，易于构建，并易于协作。（通俗一点说，Docker 就像一个盒子，里面可以装很多物件，如果需要这些物件的可以直接将该大盒子拿走，而不需要从该盒子中一件件的取。比如说我们可以在这个容器里装好 zookeeper,redis,mysql,tomcat 等软件,用的时候直接用就可以,项目部署的时候,直接把当前的 Docker 给测试组就可以,或者是运维项目组就行.）

22.你们项目是分布式的,那你有了解过分布式事务么?

当然有了,因为我们项目比较大访问用户也比较多,我们把表都用 mycat 进行拆分了,我们当时拆分的方式是(说下第 29 题),我们在支付的时候,和下单的时候都用到了分布式事务.比如时候支付吧，一笔支付，是对买家账户进行扣款，同时对卖家账户进行加钱，这些操作必须在一个事务里执行，要么全部成功，要么全部失败。而对于买家账户属于买家中心，对应的是买家数据库，而卖家账户属于卖家中心，对应的是卖家数据库，对不同数据库的操作必然需要引入分布式事务。还有就是用户下单买家在电商平台下单，往往会涉及到两个动作，一个是扣库存，第二个是更新订单状态，库存和订单一般属于不同的数据库，需要使用分布式事务保证数据一致性。我们使用的解决方案是使用支付宝用得那个 TCC 补偿性分布式事务解决方案.

(what TCC 是什么?)

TCC 是三个英文单词的首字母缩写,分别对应 Try、Confirm 和 Cancel 三种操作，这三种操作的业务含义如下：

Try：预留业务资源

Confirm：确认执行业务操作

Cancel：取消执行业务操作

1、Try：尝试执行业务。

完成所有业务检查(一致性)

预留必须业务资源(准隔离性)

2、Confirm：确认执行业务。

真正执行业务

不做任何业务检查

只使用 Try 阶段预留的业务资源

3、Cancel：取消执行业务

释放 Try 阶段预留的业务资源

TCC 的原理

我给你用这个账务拆分为说一下 TCC 吧，比如说我们账务拆分的业务场景是，分别位于三个不同分库的帐户 A、B、C，A 账户和 B 账户一起向 C 账户转帐共 80 元：

1、Try：尝试执行业务。

完成所有业务检查(一致性)：检查 A、B、C 的帐户状态是否正常，帐户 A 的余额是否不少于 30 元，帐户 B 的余额是否不少于 50 元。

预留必须业务资源(准隔离性)：帐户 A 的冻结金额增加 30 元，帐户 B 的冻结金额增加 50 元，这样就保证不会出现其他并发进程扣减了这两个帐户的余额而导致在后续的真正转帐操作过程中，帐户 A 和 B 的可用余额不够的情况。

2、Confirm：确认执行业务。

真正执行业务：如果 Try 阶段帐户 A、B、C 状态正常，且帐户 A、B 余额够用，则执行帐户 A 给账户 C 转账 30 元、帐户 B 给账户 C 转账 50 元的转帐操作。

不做任何业务检查：这时已经不需要做业务检查，Try 阶段已经完成了业务检查。

只使用 Try 阶段预留的业务资源：只需要使用 Try 阶段帐户 A 和帐户 B 冻结的金额即可。

3、Cancel：取消执行业务

释放 Try 阶段预留的业务资源：如果 Try 阶段部分成功，比如帐户 A 的余额够用，且冻结相应金额成功，帐户 B 的余额不够而冻结失败，则需要对帐户 A 做 Cancel 操作，将帐户 A 被冻结的金额解冻掉。

(How TCC 怎么用的)

Github 上有他们的源码,我们直接把源码挡下来,安装到我们本地的仓库里,用的时候我们把

需要使用分布式事务的代码,上加上@Compensable 注解,里面还有一些其他的属性配置上就可以了

Git 可以直接抽取指定类吗？