**自我介绍：**

面试官你好，我叫邢萌昊，来自河南，之前在北京阳光云视科技有限公司做后台开发，在这期间用到的框架有Springboot，SSM等主流框架，今天来咱们公司面试java开发工程师，数据库方面用的Mysql数据库，还有非关系型数据库redis等一些技术，例如SpringCloud，Dubbo等分布式系统，通过这几年的而工作，对于自身的能力也有很大的提高，这些大概就是我的基本信息。

第一家公司：6k 离职时7k

第二家公司：9k 离职时11k

**住址：**

东城区法华寺26栋2楼202

**坐车：**

**公交地铁**

**上家公司资料：**

技术开发、技术转让、技术资讯、技术服务、计算机技术培训；应用软件服务、销售计算机、电子产品。

**期望薪资:**

13K

**人员配备：**

产品经理：2人，确定需求以及给出产品原型图

项目经理：1人，项目管理，分配任务。

前端团队：3人，根据产品经理飞出的原型制作静态页面。

后端团队：6人，实现产品功能。

测试团队：3人，测试所有的功能。

运维团队：2人，项目的发布以及维护。

**项目名称:**

博识知识库

**项目介绍：**

博识全流程企业培训知识赋能系统主要为企业员工进行赋能，提升企业竞争力。基于云架构，将传统的知识学习和培训与互联网相结合。让管理者清晰了解每位员工的知识技能状态，准确把握培训需求，让培训有的放矢，让考核结果准确可靠；轻松构建企业自己的知识体系，结合常态化的培训与考核，实现优胜劣汰，不断提升企业员工的综合能力。

博识旨在帮助企业使用视频化应用方式建立知识库，实现直播培训、视频培训课件统一管理、分享以及培训后出题考核全流程，各类型企业对内方便员工自主学习，为员工赋能，对外完成消费者培育、产品推广，提供企业培训解决方案。

项目采用SpringBoot、SpringCloud的微服务框架，数据层使用SpringDataHPA，将各个组件达成docker镜像，上传托换到docker私有仓库中，通过Eur实现注册信息集中化管理，微服务之间通过feign相互调用。使用zuul对前台提供提议的API接口与APP与web端进行对接，对缓存通过redis、MongoDB进行处理，注册短信验证使用Rabbitmq/阿里大鱼短信接口，使用redis位图存储防止用户多次而已发送短信。登陆通过JWT进行token验证。定义了全局异常处理，统一返回数据处理。使用jenkins持续集成工具，将代码打包为docker镜像中，完成环境的无缝移植。

**个人指责：**

负责首页的全文检索（ElasticSearch）模块，用户在登陆上客户端的时候，可以根据自己想要找到的知识进行一个全文的检索，我们检索采用的是一个实时的搜索和分析引擎，使用ElasticSearch进行的一个全文检索，让用户可以更快的去找到自己想要的课程。

负责考试管理模块,支持考生的新增，编辑，删除，批量删除，支持题库中添加试题，只能添加试题，Excel添加试题，预览试卷，保存试卷、试卷一键发布功能。支持指定可考人员考试，默认创建考试个人所属人进行考试。

负责在线测评模块，题型丰富的互联网考评系统，让员工考评趣味化，智能组卷，自动判卷，有效降低培训师工作强度，让考评变得轻松便利，有效保障员工评价数据鲜活。

负责”购买课程模块”，购买课程我们采用了第三方支付，使用的是支付宝的第三方接口(蚂蚁金服开放平台)。

负责”课程管理模块”：上传课程模块就是用户对一些教学课程视频的上传功能，包括标题、封面、介绍、科室、头像、视频等。

首先我们的项目有一个超级管理员在配置文件里面定死的，拥有所有的权限。

登陆我们使用的时jwt，

项目分为老师端和学生端。

老师端：工作台、小组知识库、专题课堂、直播课堂、试题库、标签库、企业知识库、专题课堂、点播课堂、考试管理、试卷库。

学生端：首页、在线课堂、知识库、资讯、小测验。

**负责：**课程管理、课程详情页、订单管理、第三方支付。

后台课程管理：这个系统主要分为运营商后台和网站前台，主要是面对运营人员、运营人员登陆进入之后、点击课程管理进行上传课程、删除课程、修改课程、填写一些课程的信息：课程的信息、课程的描述、课程的讲师是谁、大概的时间是多少、在下面可以添加课程的章和节，一个张对应多个小节，每个小节会有对应的视频，点击视频上传，上传完毕之后返回一个外链，将外链保存到数据库中。

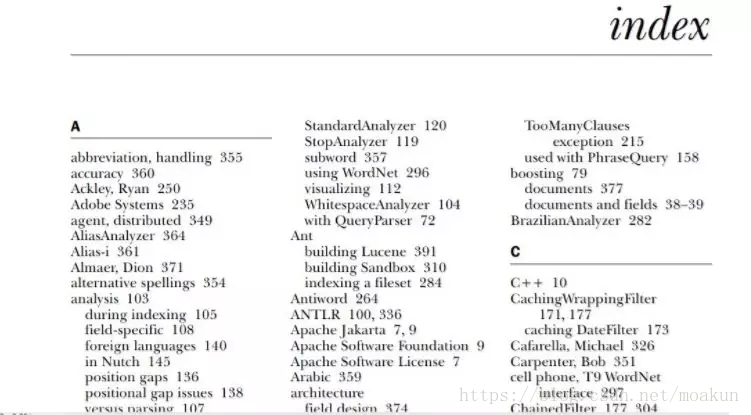
视频是公司自己的视频。公司有自己的讲师，录制完毕之后，将视频传给后台运营人员，然后后台运营人员进行上传。

**ElasticSearch**

Elasticsearch中的倒排索引是什么？

倒排索引是搜索引擎的核心。搜索引擎的主要目标是在查找发生搜索条件的文档时提供快速搜索。倒排索引是一种像数据结构一样的散列图，可将用户从单词导向文档或网页。它是搜索引擎的核心。其主要目标是快速搜索从数百万文件中查找数据。

一般情况下，像下面的一样，在书中我们已经倒过来索引。根据这个词，我们可以找到这个词所在的页面。



ElasticSearch中的集群、节点、索引、文档、类型是什么？

群集是一个或多个节点（服务器）的集合，它们共同保存您的整个数据，并提供跨所有节点的联合索引和搜索功能。群集由唯一名称标识，默认情况下为“elasticsearch”。此名称很重要，因为如果节点设置为按名称加入群集，则该节点只能是群集的一部分。节点是属于集群一部分的单个服务器。它存储数据并参与群集索引和搜索功能。索引就像关系数据库中的“数据库”。它有一个定义多种类型的映射。索引是逻辑名称空间，映射到一个或多个主分片，并且可以有零个或多个副本分片。 MySQL =>数据库 ElasticSearch =>索引文档类似于关系数据库中的一行。不同之处在于索引中的每个文档可以具有不同的结构（字段），但是对于通用字段应该具有相同的数据类型。 MySQL => Databases => Tables => Columns / Rows ElasticSearch => Indices => Types =>具有属性的文档类型是索引的逻辑类别/分区，其语义完全取决于用户。

ElasticSearch中的分片是什么？

ElasticSearch中的副本是什么？

**支付模块**

**微信支付流程**

**微信扫码支付模式一和模式二的区别**

**模式一，先扫码，再生成订单。**

**模式二，先生成订单，再扫码。**

**应用场景：**模式一，更适合无人职守的自动售卖机。所有的商品都有一个固定的二维码，价格相对稳定，当用户使用微信支付扫描了二维码，微信再请求自动售卖机的服务提供商的 后台接口，注意，这个请求中，是包含了商品ID以及用户信息的，这样，商户系统就可以根据 商品ID，以及用户的身份，再来确定用户实际要支付的金额。

模式二，更适合有人职守的，支付金额非常不确定的场合。比如，你去饭馆吃饭，虽然每个菜的金额是固定的，但一桌子饭菜的金额不固定，甚至是你还可能使用饭馆事先发放的代金券。这个时候，就需要收银员，预先创建一个订单，确定好金额，然后你再来扫描这个二维码来支付。

**用到了支付日志表（**提交给微信支付接口的订单号，可能会有多个订单号，和order哪里的订单号不一样，为什么订单表和支付日志表是一对多的关系呢？一条记录可能会对应多个订单标号，支付订单号可能会因为商户的不同而相同，但是交易流水号是唯一的，他是微信系统的生成订单的时候要将订单记录增加到日志记录表，添加支付日志要把订单id放到记录表中需要在订单提交以后修改订单记录表中的状态为已支付，并且增加交易流水号，需要在订单表中将支付状态改为已支付**）和订单表**

**大体流程：**

**在支付页面上生成支付二维码，并显示订单号和金额，用户拿出手机，打开微信扫描页面上的二维码，然后在微信中完成支付。**

**(1)在用户下单时，判断如果为微信支付，就像支付日志表添加一条记录，包括支付总金额、订单ID(多个)，用户ID、下单时间等信息，支付状态为0(未支付)**

**(2)生成的支付日志对象放入redis中，以用户id作为key,这样在生成支付二维码时就可以从redis中提取支付日志对象中的金额和订单号**

**(3)当用户支付成功后，修改支付日志的支付状态1(已支付)，并记录微信传递给我们的交易流水号，根据订单ID(多个)修改订单的状态为2(已付款)**

**微信支付模式二：**

**主要是两部分，调用统一下单API()生成已支付交易链接并返回预支付交易链接(code\_url),然后将链接生成二维码图片，第二部分是调用查询订单API()，返回支付的状态看支付是否成功和微信支付交互都是通过和一个地址来交互的把需要的参数post给地址，分然后返回需要的数据与微信支付接口交互的时候需要提供微信公众账号，**

**实现类中的逻辑**

**1.参数封装**

以map集合的形式去分装，公众账号id，他是从配置文 件来的从.properties中的四个参数的一个 ，商户号id也是从配置文件中获取，partner=,随机字符串也是必须要提交的(WXPayUtil工具类中有的可以生成随机字符串)，扫完码马之后有个商品描，商户订单号是前台传来的，还有标价金额,还有终端ip,(负责记录)，通知地址，也是回调地址（用不到必须给），交易类型必须给，签名会自动产生，根据param参数生成一个带签名的xml

**2. 发送请求**

**3.返回结果**

**前台在生成二维码的时候就调用查询方法，因为调用了方法之后会在contrller之中进行一个循环知道得到成功或者失败的的结果跳出循环了，才会在页面上跳到成功或者失败页面**

**微信支付遇到的问题：前台支付页面关掉之后后台的循环还在继续，这个时候需要在controller中修改代码，让其在五分钟后显示二维码超时，在前台需要重新生成二维码。**

**具体实现：**

我们电商项目呢引入了目前两个最主流的支付方式，一个是微信支付一个是支付宝支付，微信支付呢我们采用的是电脑扫码的支付功能，然后它除了这个支付之外呢还有app支付，然后还有公众号支付，还有h5支付等等接口，电脑扫码支付呢主要有两种模式我们采用的是方式2，在用户选择微信支付的时候我们需要将订单的id呢传给我们的controller，然后根据订单id去(订单表)查询我们订单的总价，然后将总价转化为以分为单位的一个金额，然后这一点呢和支付宝不太相同，支付宝支付的时候是以元为单位，然后将订单的金额，商户号，公众号的id,还有公众号的秘钥，然后还有接受支付结果的异步回调路径，然后等等一些信息，还有一些随机字符串儿等等，然后把它封装到一个map里，然后呢微信支付为了验证这些数据是否来源于开发者，他需要将我们这些数据加密，把所有的这些参数的key按照字典序进行排序然后把所有参数的键和值拼接成一个字符串，在再这个字符串的后面在拼接上我们申请商户号时候所获得的apikey,然后呢将这个完整的字符串进行md5加密，得到一个新的32位的字符串，然后这个字符串呢我们叫它签名sin,然后呢我们需要把这个签名加上之前提到的订单金额，随机字符串，商户号，公众号id等等十几个参数然后呢封装成一个xml的字符串，然后再封装成xml字符串的时候呢，微信的官网上为我们提供了一个demo和一个sdk的下载，从里面呢我们可以找到一个maptoxml的一个工具类，然后除了这个工具类呢我们还可以借助一些jar包比如有charsfreem,然后可以用它里面的一些方法，然后把map转成一个xml的字符串，当然我们也可以自己去封装工具类，然后呢将转成的xml的字符串用httpclint传给微信下单里面的统一下单api,然后呢微信在接收到我们这段xmlbaowener之后，他会取出我们公众号的id还有商户号然后获取到我们的这个秘钥，然后拿着所有的参数加入秘钥去签一个名字，然后呢和我们的签名去做对比如果一致的话它就会去处理我们的请求如果它签出来的名字和我们所传的签名不一致，那么说明我们所传的这个api的key就是伪造的，然后他就会拒绝我们的请求，然后等他验证通过以后，他会把我们的这些数据存到他们的服务器里并且为我们生成一个预支付的订单，然后他会将这个预支付订单的id,还有我们做扫码支付时需要用到的二维码的url路径，以xml的的形式返回给我们，然后就可以通过httpclint获取她的response,然后获取resopnse里面的getentity把它转成String类型的字符串然后得到一段xmlbaowener,然后他返回的这个二维码的url路径呢并不是说我们引用这个路径就可以得到这个二维码，而是我们需要将这个路径转化成一张二维码，转化成一张图片，那么我们这个时候就可以借助谷歌的工具类一个jar包jxing吧，将这个字符串转成一张图片并且用response对象响应回这个页面做展示然后至此我们服务端的第一步就完成了，然后下面的用户通过手机去扫描这个二维码进行付款，当用户付款成功以后呢微信除了会向手机端发送回一个支付结果以外还会主动的调用一个异步回调接口，所以我们需要提供一个接口用来接收支付结果，当我们接收到这个支付结果以后呢，我们需要从支付结果中取出订单的id还有他的实付金额然后根据订单id去订单表中查询这个订单到底需要付多少钱，然后将查出来的金额和用户所付的金额作对比判断是否一致 ，如果一致的话我们需要去修改订单状态如果不一致说明用户篡改了我们的支付请求，那么我们这个时候可以把钱退回给客户，然后将订单修改为异常，然后当我们收到支付结果以后，并且确认这个支付金额是有效的时候我们需要给微信服务器通过resopnse.getrighter.print 给他返回一个success字符串，表示我们已经接收到并且处理好这个支付结果了，然后这样的话，微信服务器就不会重复的给我们发送这个支付结果否则的话微信服务器会在24小时发送大概7.8次吧这个结果到我们的异步接口上异步回调路径上一般来说我们首次就可以接受并处理好这个请求然后页面中为了实现付款之后就自动跳转的功能我们可以通过在页面里面在js里面去设定一个setinterwol去开启一个定时的一个刷新每十秒或者每五秒去调用我们的controller,然后去做一个订单的查询，那么当我在异步接口里接受完支付结果并且修改状态为已支付的时候实际上我们可以把订单的信息缓存到redis中，然后我们每次页面里触发这个定时任务的时候然后都可以根据id去redis里查询是否有id 的支付结果如果有的话我们需要给前台返回一个true,然后前台页面去做判断如果支付成功的话就可以跳到支付成功页面给用户一个信息的提示就算页面没有跳转的话，当用户查看订单页面的时候我们的订单状态也已经改变了，如果订单状态异常的话我们也可以给页面返回一些错误代码然后由前台给用户一些提示，然后跳转到相应的提示页面

**订单管理模块:**

**首先是订单表的设计，主要包括订单表和订单详情表，订单表主要包含订单的主要信息，比如订单的编号、总额、数量、状态、收货人信息等。其中收货人信息必须要冗余到订单表中，不能简单用Id进行管理。订单详情表和订单表是多对一关系，订单详情表主要计量订单中的商品的详细信息，这些信息也要冗余进来，不能通过id进行简单的关联，因为订单一旦生成，这些信息一般不会再允许改变。**

**订单在用户结算购物车时生成，如果同时购买多个商家的商品，在结算购物车时需要进行分单，同时生成多张订单。在用户中心，每个用户都可以看到并跟踪自己的订单，进行支付、申请退货、确认收货、评价等操作。商家后台可以看到商家自己的所有订单，进行确认发货操作。而在运营管理平台，可以监控所有的订单，但是不能进行操作。 订单的状态主要包括：待支付、待发货、已发货、已完成、已取消；生成订单时，应该对库存进行一次校验，防止超卖；**

## **注释:订单生成方式的几种方式:**

1. **UUid**组成：当前日期+时间+时钟序列+机器识别号（Mac地址或其他）没有mac网卡的话会有别的东西识别。在分布式系统中，所有元素（WEB服务器）都不需要通过中央控制端来判断数据唯一性。几十年之内可以达到全球唯一性
2. 2.redis(产生自增的序号:主键的访问需要Resids,对Redis有依赖);
3. 3.Oracle 数据库对象序列(与表无关),只有数据库用orcal才能用
4. 4.自己写算法;(雪花算法,生成器switter)

**===============支付宝支付整体流程===================**

**一、申请企业账号**

需要提交好多资料

**二、创建应用，让应用上线**

**三、签约支付功能**

**四、设置应用的秘钥和授权回调地址等信息**

**五、看开发文档写代码**

**1、在商品页面点击购买按钮提交后台，**

后台接受哪些参数：商品id、商品数量、商品价格

**2、后台做哪些操作？**

**（1）获取用户id，判断用户是否登录，没登录，提示登录，登录后继续执行**

**（2）根据商品id查询商品信息，获取商品价格等，商品信息查询错误，直接返回提示购买信息错误，请重试。查询成功，根据用户id、商品id、商品价格、商品名称、购买时间等字段，生成一条订单表数据**

**（3）调起支付宝支付**

支付宝支付直接复制API上的代码，填充写参数就可以了。

①调用支付宝填写的必要参数

请求URL，appid、public-key和private-key

设置notifyurl和returnUrl。

notifyUrl是通知给咱们平台支付结果

returnUrl支付成功后返回到哪个页面。

②调用支付宝业务参数需要哪些

商品的id、商品的价格、商品的名字

**用户在支付宝中输入支付支付完成后返回咱们平台。**

**（4）支付完成后会有哪些操作？**

支付宝调用我们平台，通知下支付结果

（1）先根据支付宝提供的验证方法，对返回来的参数进行校验，校验失败，返回失败；

校验成功，根据返回来的商品id，更新订单表中的订单状态，改为已支付，并返回给支付宝一个字符串success。

支付宝如果收到返回失败，就把账单变成异常账单，需要人工处理这个账单，如果是返回success等于支付成功，支付宝会两周内把钱打入到咱们账号中。

**如果支付宝未调用到你的回调方法，会怎样？**

支付宝会尝试7次调用我们的回调方法，调用时间为：4m,10m,10m,1h,2h,6h,15h。如果7次中有一次调用成功了将不会再调用了，如果一直没有调用成功，支付宝会将这条信息状态标记为异常。

**你是如何保证你们支付的幂等性的？**

**（1）提交时的幂等性，做个token拦截，防止用户重复提交**

①cookie和redis中放一个都放一个token

②提交到后台后，获取cookie中的token，比较redis当中的token，把token删除掉，防止用户重复提交

**（2）支付宝通知时防止的幂等性**

①根据支付宝自己提供的校验，把返回来的参数进行校验

②校验成功后，根据商品id把订单表里校验下，查看订单表中是否有数据，如果没数据返回失败

③如果有数据，根据订单表中查询回来的对象价格和支付宝返回来的价格进行比较，查看下支付的价格是否和我平台的一致，如果一致，说明支付成功，并修改我数据库订单表中的状态为已支付。

**通过前台的这几次校验来保证支付的幂等性。**

因为我们项目是一个分布式的系统，有好多web工程都需要使用发送短信的功能，比如说 注册、支付还有很多场景都用到了发短信的功能，我们采用的架构是SOA 面向服务的架构，而短信不涉及具体业务，而是一个通用的功能，所以我们项目中,是采用SpringBoot 和 ActiveMQ实现的一个微服务抽取出所有发送短信的功能。短信我们使用的是阿里大于提供的短信服务。阿里的大于短信微服务接口,是现在一块很不错的短信服务这一块

**注册登录：**

在做注册的时候，我们根据公司提供的API文档，找到对应的发送短信的接口写入到我们的controller当中，在我们的service层中生成验证码，这个验证码使用random随机数生成，当时我们控制这个随机数为六位数这个范围，也就是大于等于100000小于等于999999，得到验证码后，将验证码存到我们的redis当中，并且设置这个验证码的过期时间，因为这个验证码并不会一直都有用，用户使用一次之后，就没有什么太大的用户了，处理好这些后，我们将用户的手机号和验证码放到一个map集合当中，然后将整个集合发送到rabbitMQ指定的队列上等待被消费，这里用rabbitTemplate这个模板调用converAndSend这个方法便可以直接塞入到指定的队列当中，我们在调用短信接口阿里大于的时候，因为我们这个是一个微服务的项目，所以我们专门写了一个工程，这个工程就是专门为了获取rabbitMQ队列上的数据调用阿里大于给我们客户发送短信用的，在这里我们写了一个消息监听类，引入@RabbitListener和@Component以及@RabbitHandler便可消费我们rabbitMQ上的消息，我们往rabbmtMQ传的时候用的MAP集合装的，获取的时候当然也就用Map集合来进行获取，我们在使用阿里云发送短信时，他就提供给我们一个短信工具类，这个短信工具来就可以自动的帮我们发送短信了，我们的短信模板，短信签名，手机号在阿里云官网就可以直接弄好了，他就可以直接给我们指定的手机发送短信，然后我们的短信监听类在使用阿里云提供的的短信工具类这些模板和签名的时候，直接使用@Autowired注入就好了，然后调用工具类里的发送短信方法，便可成功发送短信，我们用户在进行注册的时候，输入验证码后，去拿着我们验证码和我们redis缓存当中的验证码做对比，如果验证码为空和输入有误，则提示用户验证码有问题，如果正常的话则对我们的用户密码进行加密，这里加密的时候采用的是Bcrypt加密算法，我们当时也考虑过用MD5加密，但是后来发现这个东西能够被破解，而Bcrypt这个东西里面有时间戳，时间戳这个东西是几乎不可能被破解的，Bcrypt这个加密算法是Spring Security安全框架提供的，所以我们只需要引入Spring Security这个依赖便可以了，但是我们们引入这个安全框架之后，所有的地址就得被安全框架所控制了，而我们只是用单纯的加密算法，所用这个时候我们建立一个安全配置类，里面引入@Configuration 和@EnableWebSecurity这两个注解，便可配置所有的地址匿名访问，在启动类里面bean注入BCryptPasswordEncdor就可以加密了，我们再进行登陆效验密码的时候，BCryptPasswordEncdor给我们提供了两个方法，encode和matches方法，用encode方法进行加密，matches进行用户前台传入的密码和我们数据库加密后的密码进行比对，我们在登录的时候，选择了基于JWT的Token认证机制，客户端使用用户名和密码进行请求登陆，服务端收到请求，去验证用户名和密码，验证成功后，服务端会发送一个Token，再把这个Token发送给客户端，客户端收到Token后把他存储起来，比如经他放到cookie当中，客户端每次向服务器请求资源的时候，都会带着这个Token串，服务端收到请求后，会拿着前台传过来的Token串和我们服务端的Token串进行对比验证，成功就返回给客户请求的数据，  
  
创建token方式首先导入JJWT依赖，设置签发时间，设置密钥，每次的token都是不一样的，因为载荷当中存在时间.

**考试管理模块：**

首先我们的表设计的是多表联查，一个试题表，一个选项表，一个答案表，一个用户表，一个老师表，一个课程表。

首先用户登录之后进入到我们的页面，然后我们后台会有一个定时器，到我们规定的时间之后他才可以参加考试，这个操作是我们将redis里面存储一个数据，然后我们用户在进行点击考试的时候会做一个拦截器，拦截规则是当他查询到redis里面的有哪个倒计时的字段，就会给他拦截到当前的页面，如果没有倒计时的哪个字段就会给他放行，当我们的后台管理员在发布试题的时候我们会将这些试题存放到redis里面，通是我们还是用freemark生成一个静态页面，我们将这个静态页面放在resource目录下创建一个templates文件夹，然后我们后台直接返回静态页面的地址就行了，当我们的用户点击参加考试的时候就会跳转到我们的静态页面，然后我们的用户开始参加考试，当我们的用户参加考试的时候，我们会有一个一个小时之内自动提交试卷的功能，这个功能是我们在前台的JS里面写的一个计时器，这个计时器访问的时候我们服务器的时间，防止用户自己篡改本机电脑的时间，然后进行重新考试，这样的话我们就能够达到一个同一个的时间。为了防止考生作弊，当考生没有及时的参加考试的话，我们考生可以找到自己的老师，然后老师再去找管理员让管理员再重新生成一份试题，或者是我们出题人直接给老师发一份word的试题，让没有参加考试的考生进行考试。然后就是我们考生考试完成之后我们会有一个系统自动判卷的功能，我们的系统判卷就是让用户再页面上选择的答案与我们答案表里面的答案做对比，如果相同就给分，如果不相同的话就没有分，我们的多选题是用户在前台做完题之后我们前台做一个字符串拼接，拼接完之后我们再将这个拼接好的字符串传到我们的后台，我们根据后台查询到答案表的数据与我们前台传过来的字符串对比，逻辑同上。

**支付：**

用户下单，下单点击支付之后，将订单编号传到后台，然后后台根据订单编号去查询课程金额、课程名称、课程描述、然后将这些参数传给支付宝统一下单的API，传完之后支付宝那边进行处理，处理完毕之后返回的是一个表单页，然后表单页拿到前台直接呈现就可以了，然后呈现之后用户就跟支付宝做交互，此时就不跟我们的平台坐交互了，交互完之后有两个访问地址returnURL,notiflyURL，同步通知的地址、异步通知的地址，如果说用户支付完毕之后，他还停留在浏览器的页面，肯定会走同步通知的页面，同步通知的页面就是跳转到支付成功的页面，告诉用户支付成功了，金额是多少。

异步通知事作为同步通知的一种补偿机制。因为同步通知是具有关联性的比如说刚才说的用户支付完之后关闭了肯定就请求不到了，如果拿同步通知进行对订单状态的控制，那肯定会出现问题，所以说用的是异步通知，因为异步通知有一个重试机制，重试机制会在25小时重试8次，支付成功之后他会请求你的这个接口，如果你的这个接口不给他返回success，他就会异步尝试8次，后台需要有个接口，支付宝会回调我们的这个接口，因为支付宝回传过来的URL后面跟了一大堆参数，我们使用一个map来接，我们将URL里面的参数封装到map里面，从map集合里面取，验签的操作是支付宝封装好的，主要是我们先要接到她返回的参数封装到map集合中，然后调用支付宝的验签的操作，验签完毕之后，在后再根据他回传回来的订单编号去数据库中再核对订单的金额，还有APPID，因为怕篡改，验签包括验证金额都是为了保证金额都要去验证，验证通过之后去修改订单的状态，修改完订单的状态返回给支付宝一个SUCCESS。

并发：做一个表单重复提交，做一个token，当用户第一次提交之后后台生成一个token，也可以在前台，用户点击之后去把这个按钮置灰

**搜索模块：**

在做这个搜索模块的时候 我们用到的是ES搜索引擎 ：（Elasticsearch是一个实时的分布式搜索和分析引擎。是当前流行 的企业级搜索引擎。设计用 于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便）

我们也想到了用solr 但是通过对比 发现ES更符合我们这个项目的需求

因为ES它在实时搜索的时候效率是高于solr的 Solr更适合传统搜索应用，在搜索大的数据的情况下 ES的效率会很高 solr需要配域而ES不需要 相对来说比较简单

我们在做搜索的时候主要是根据用户的需求进行课程搜索，我们是使用docker搭建一个ES的公共服务

一开始我们是用postman调用restapi新建索引和文档 在新建文档的时候 id是由系统自动生成的。在修改文档的时候是根据id来进行修改的 如果id不存在 它就会创建新的文档 删除文档的话也是根据id来进行删除的 通过这个我们发现 如果都是通过rest请求的方式使用ES未免太过麻烦，而且也不够人性化。然后就想到了使用图形化界面来实现ES的日常管理，最常用的就是Head插件 在安装好Head插件后 我们发现了一个问题 就是它在启动之后 点击连接按钮没有反应，我们按F12发现了一个错误 “No 'Access-Control-Allow-Origin' header is present on the requested resource” 这个错是由于ES默认不允许跨域调用，而es-head是属于前端工程，所以报错。然后我就修改了ES的配置 让它允许跨域调用,我们需要在elasticSearch.yml里面加两条命令(http.cors.enabled: true http.cors.allow-origin: "\*"),设置为true让他允许跨域。

我们在做测试的时候 发现它的分词不太智能 默认的中文分词是将每个字看成一个词 这样是不符合要求的 所以我们就用了IK分词器来解决这个问题 IK 分词器有两种算法 其中ik\_smart为最少切分，ik\_max\_word为最细粒度划分

我们还可以自定义词库

然后 我们又要考虑一个问题 因为我们查询的数据是来自不同的数据库或者表 这样的话就很难以一个比较快的速度去查询数据 这个时候我们就想到了用这个logstash进行同步数据 同步数据库与索引库 （solr是用的定时器 而ES用的是logstash）logstash是一款轻量级的日志搜集处理框架， 咱们在公司里边用这个ES也是分两种 一种是增量索引更新一种是全量索引更新 当我们安装好logstash 需要在logstash的bin文件夹下创建一个.config结尾的文件，在里面配置mysql的链接还有查询课程表的sql语句、es的链接，最主要还是需要配置一个定时任务。这个定时任务就是精确到年月日时分。在这儿我配的是每1小时同步一次。 可以让它定时的去读我的数据库里边的东西，读完之后再把他所读东西搜集起来指定给ES 这样的话就相当于隔1小时ES就被同步一次

面试官：项目中搜索为什么要选择使用Elasticsearch？

面试者：

Elasticsearch是一个实时的分布式搜索和分析引擎。它可以帮助你用前所未有的速度去处理大规模数据。ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发 的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用 于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便

优点

1.elastcisearch是分布式的。不需要其他组件，分发是实时的。

2.处理多租户不需要特殊配置，而solr则需要更多的高级设置。

3.elasticsearch采用gateway的概念，使得完备份更加简单。

4.各节点组成对等的网络结构，某些节点出现故障时会自动分配其他节点替其进行工作。

面试官：能说说ES和solr 的差别吗？

面试者：

Elasticsearch与Solr的比较总结

Solr利用Zookeeper进行分布式管理，而Elasticsearch自身带有分布式协调管理功能

Solr支持更多格式的数据，而Elasticsearch仅支持json文件格式

Solr官方提供的功能更多，而Elasticsearch本身更注重于核心功能，高级功能多有第三方插件提供

Solr在传统的搜索应用中变象好于Elasticsearch，但在处理实时搜索应用时效率明显低于Elasticsearch

Solr是传统搜索应用的有力解决方案，但Elasticsearch更实用于新兴的实时搜索应用。

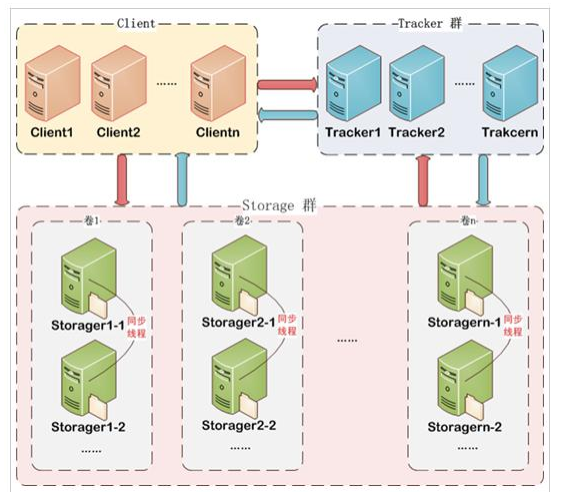
## **什么是FastDFS**

FastDFS 是用 c 语言编写的一款开源的分布式文件系统。FastDFS 为互联网量身定制，充分考虑了冗余备份、负载均衡、线性扩容等机制，并注重高可用、高性能等指标，使用 FastDFS很容易搭建一套高性能的文件服务器集群提供文件上传、下载等服务。

FastDFS 架构包括 Tracker server 和 Storage server。客户端请求 Tracker server 进行文件上传、下载，通过 Tracker server 调度最终由 Storage server 完成文件上传和下载。

Tracker server 作用是负载均衡和调度，通过 Tracker server 在文件上传时可以根据一些策略找到 Storage server 提供文件上传服务。可以将 tracker 称为追踪服务器或调度服务器。

Storage server 作用是文件存储，客户端上传的文件最终存储在 Storage 服务器上，Storageserver 没有实现自己的文件系统而是利用操作系统 的文件系统来管理文件。可以将storage称为存储服务器。



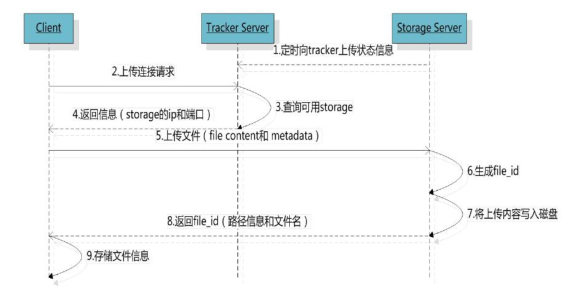
服务端两个角色：

Tracker：管理集群，tracker 也可以实现集群。每个 tracker 节点地位平等。收集 Storage 集群的状态。

Storage：实际保存文件 Storage 分为多个组，每个组之间保存的文件是不同的。每个组内部可以有多个成员，组成员内部保存的内容是一样的，组成员的地位是一致的，没有主从的概念。

## 5.2文件上传及下载的流程

### 5.2.1 文件上传流程



客户端上传文件后存储服务器将文件 ID 返回给客户端，此文件 ID 用于以后访问该文件的索引信息。文件索引信息包括：组名，虚拟磁盘路径，数据两级目录，文件名。



 **组名**：文件上传后所在的 storage 组名称，在文件上传成功后有 storage 服务器返回，需要客户端自行保存。

 **虚拟磁盘路径**：storage 配置的虚拟路径，与磁盘选项 store\_path\*对应。如果配置了

store\_path0 则是 M00，如果配置了 store\_path1 则是 M01，以此类推。

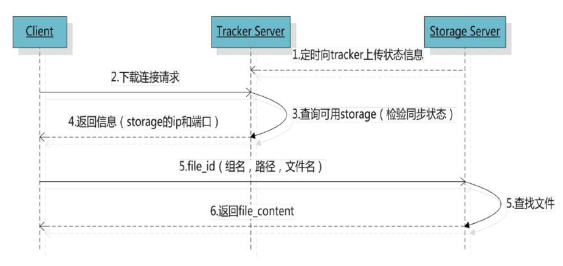
 **数据两级目录**：storage 服务器在每个虚拟磁盘路径下创建的两级目录，用于存储数据

文件。

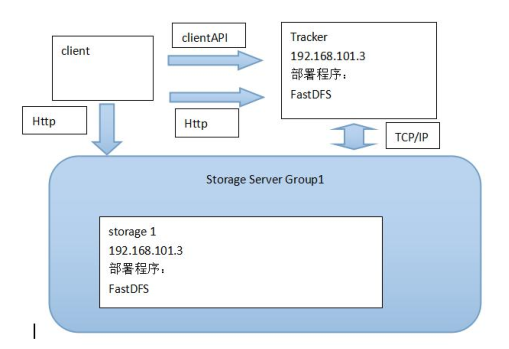
 **文件名**：与文件上传时不同。是由存储服务器根据特定信息生成，文件名包含：源存储

服务器 IP 地址、文件创建时间戳、文件大小、随机数和文件拓展名等信息。

### 5.2.2 文件下载流程



## 5.3最简单的 FastDFS 架构



# **FastDFS文件上传大小限制**

1.文件上传限制：  
对于Spring1.4之前的版本：

multipart:

maxFileSize: 50MB

maxRequestSize: 50MB

对于Spring 1.4 之后的版本:

spring:

http:

multipart:

maxFileSize: 50Mb

maxRequestSize: 50Mb

**教育经历**

1、2011年就读于郑州工程远程15年毕业。

**自我评价**

1、自己经常到csdn、博客园与楼主、博主进行探讨交流。并研究过多线程以及hashmap底层，对redis有深入的研究。

2、可以适应各种环境，并从中找到自己的精准定位，在项目开发中能够守时守质、独立完成领导交给我的任务，在工作中遇到的问题及解决方案都会进行总结。并会在开源中国等网站学习新技术并与团队分享，善于思考感悟，从而积极面对工作。