# 云计算 1607C 党银涛

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 面试官，您好！我今天给大家带来的项目是网交平台，网交平台是一个交友平台，就是为了能够让群众能够更好的交流。我们项目采用前后端分离的系统架构，整个项目我们采用了SpringCloud微服务框架，后台我们用了SpringBoot+SpringMVC+SpringDataJPA技术，我们平台提供了头条，问答，活动，交友，吐槽，招聘六大功能。我们前台有注册，登录系统，使用JWT进行验证，增加安全性，登录成功之后，用户可以发表自己的文章，也可以对别人发表的作品进行点赞和评论，因为这些数据量较大，我们采用MongoDB进行存储评论，用户点赞的时候我们采用Redis缓存技术，所以在整个项目中数据库我们用到了关系型数据库MySQL，非关系型数据库Redis和文件存储数据库MongoDB。如果有喜欢的人还可以进行添加，对方同意之后可以进行聊天和约会。  后端使用：SpringBoot+SpringCloud+SpringMVC+SpringDataJPA。  前端使用：VUE+NodeJS+ElementUI |
|  |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| 1. 注册登录微服务：   用户想要在平台上面发表自己的文章，或者是对别人进行评论点赞，以及其他的操作，首先要注册一个自己的账号,如果去注册的话我们输入手机号进行短信验证,对于短信验证这一块我们利用的是阿里大于短信接口进行验证的,当前台点击验证码时后台随机生成一个6位数(我们利apache.commons.lang3)将随机数利用redis进行存储，另外发送到消息中间件rabbitMQ中，通过监听类进行监听，然后调用阿里大于短信接口实现短信发送 之后将验证码填入与之前存储的进行比对，这样就可以实现用户注册了。登录成功以后才能进行发表自己的文章和观点，进行添加好友等一系列操作。而且我们在登录的时候，用MD5对密码进行加密，注册登录还基于JWT的Token认证机制实现，Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准。该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。这样在用户登录的账号密码会和数据库中的进行判断，如果一致才能登录成功，这样能够对用户的信息更加安全。   1. 评论微服务：   当用户在浏览别人发表的文章的之后，可以对别人的文章发表一下自己的观点和看法，根据自己的喜爱程度发表自己的想法，对于吐槽这一模块来说我们首先选择的是MongoDB数据库，因为吐槽的信息数据量，写作频繁，而且我们吐槽一些问题和人的话数据丢失几条也没有什么影响的，正好MongoDB正好符合这一特点,还有吐槽对于表结构设计时是有特点的,我们在最后设计一个parentid 与主键\_id设置主外键关系,不是真正的主外键而是为了有树形结构这种体系。我们都知道在吐槽时是可以接着你吐槽的东西继续吐槽的，当然在相互点赞的同时我们也是利用了MongoDB里面的一个自增列,使用 $inc 运算符，对某列进行增加或减少，使用MongoDB数据库能够更好的存储数据。   1. 点赞微服务：   用户可以对别人发表的文章进行点赞操作，但是因为要限制点赞次数，我们需要在业务中添加一个判断条件，判断用户是否已经点过赞，如果点过赞就提示用户不能重复点赞，如果用户还未，点赞，则对用户提示点赞成功。我们在点赞中用到了Redis数据库，因为Redis数据库是非关系型数据库，而且存储也比较安全。   1. 搜索微服务：   我们的搜索模块使用的是Elasticsearch，因为Elasticsearch是一个实时的分布式搜索分析引擎，可以快速的去处理大规模数据，提高搜索效率。但是我们在搜索的时候不能把查询到的信息更加细化，所以我们在搜索的时候用到了IK分词器，还有就是我们这个搜索的分词，默认的中文分词是将每个字看成一个词，比如说“我是程序员”这个词他就会给你分成每个字一个词，我这里用的是IK分词器，使用相当的简单，解压出来以后，给这个文件夹重命名为IK，之后把这个文件夹拷贝到ElasticSearch/plugins目录下，重新启动就可以加载IK分词器  IK分词器提供了两个分词算法，ik\_smart和ik\_max\_word其中ik\_smart为最少切分，ik\_max\_word为最细粒度划分，我选用的是最细粒度。Elasticsearch和IK分词器对我们的搜索有特别大的优处。   1. 交友微微服务：   用户如果有喜欢的人，可以对喜欢的用户进行好友添加和关注，然后等待别人的好友同意，如果得到同意就可以和对方就行聊天和约会。在交友微服务中在添加别人的同时可能会发生不同的情况，如果对方同时也喜欢你的时候就可以成为好友，如果对方不喜欢你的时候就会是单向的喜欢，但是你还可以关注对方，这时候你就是对方的粉丝。  对于交友模块的话,当用户登陆后在推荐好友列表中点击“心”，表示喜欢此人 ，在数据库tb\_friend表中插入一条数据，islike 为0,当你点击了喜欢过的人，也喜欢了你,表示互粉成功！也向tb\_friend表中插入一条数据，islike为1并且将你喜欢她的数据islike也修改为1，当你点击了不喜欢某人（点击了叉），向tb\_nofriend添加记录.当两个人互粉后，其中一人不喜欢对方了，删除好友表中的记录 ，向非好友表中添加记录。我们在用户模块里面写了一个更新好友粉丝和用户关注的方法时,我们让交友模块进行调用,利用的是SpringCloud里面的一个服务调用的组件Feign,通过在交友模块写了一个接口,利用@FeignClient进行方法的调用。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 首先项目主要用到的是微服务架构和分布式架构,项目中就会遇到有关分布式场景的一些问题,例如分布式事务,分布式锁,跨域等问题.  **分布式事务:**产生的原因是在多个系统中数据库垂直分割,数据库之间不能跨数据库进行访问,所以每个数据库的事务,都是在自己数据库独立进行管理的.这样就需要解决分布式事务的解决方案.而解决分布式事务的原理则是利用了Java的JTA,而java的JTA则是java操作XA接口,XA相当于协调者提供资源管理器与事务管理器之间进行通信的标准接口。而他们遵循2pc,3pc协议而我们绝大部分采用柔性事务都遵循CPA和Base理论,可以暂时不一致,但是最终要保证数据的一致性而我们公司最开始的探讨研究过分布式的解决方案第一种利用的是java Automatic框架,最开始感觉java Automatic有一定的缺点他遵循的是两段提交协议,他可能出现延时,当别人没有进行通知时可能进行卡死而我们公司发现Automatic用于多数据的情况下,用于传统的项目,而我们公司采用的是分布式项目所以不适合采用.在当时我们有采用另外一种方案,采用MQ解决分布式事务,但是由于我们的项目利用的是Springcloud,用MQ会比较麻烦所以最后又采用新的方案支付宝回调方式,就是类似于支付方式去做分布式事务管理但是又要自己写补偿机制和重拾机制还有要解决一些幂等性问题还是比较麻烦的,最后在github上找到一个新的框架tcc,lcn框架,发现lcn框架写的比较好,然后我们自己研究了一番,发现lcn底层原理的最大核心我不管理事务,而只是事务的搬用工lcn的原理是发起方调用接口,去xml创建事务分组,将事务分组的id传入,参与方接收到id时,操作一个假的关闭,不提交事务,当发起方代码执行完成以后,通知协调者,然后转发给参与方,看事务回滚还是提交.这是我们分布式事务的一个解决方案.  **第二个难题就是分布式锁**,什么是分布式锁,我们当时在项目当中遇到一种场景,需要做一个全局的id,但是全局id需要一个时间戳,如果在集群的情况下,时间戳因为服务器,可能在时间戳产生相同的情况下,当时想了好多的办法去解决全局id,之后利用分布式锁去解决.分布式锁相当于在多个服务器之间进行通信时,能够保证JVM性能的安全性问题,分布式锁跨Jvm,分布式锁在多个jvm进行通信时,能够保证多个JVM只有一个进行执行.在当时也有两种解决方式一个是利用reids,但是redis的有效期是比较麻烦的,有可能产生死锁,另外一种通过zookeeper的节点机制,首先zookeeper通过临时节点,当别人可以将临时节点创建成功时,就可以拿到我们这个锁了,因为节点是不允许重复的,当别人创建节点时,是不可创建的,需要去等待,等锁释放完成以后才可以拿到锁,那么zookeeper里面是怎样管理锁的释放了,zk有一个比较好的概念,就是有一个临时节点的特征.临时节点相当于session会话一关闭的情况下,整个连接将会断开,断开以后节点清空.所以锁就会被释放.  **跨域问题**：在前后端分离的情况下难免会出现跨域问题,在当时前端通过Ajax发请求时访问不到,跨域问题产生于浏览器方面,不是在服务端,而在当时我们就在服务器端解决跨域问题,同时也考虑过客户端解决,但是客户端解决有一些缺陷,通过jsonp,jsonp本身原理是通过标签进行转发,有这样一个缺陷,这个缺陷在当时我们解决方式是因为它只支持get请求,不支持post请求,所以我们当时就搭建网关的方式,网关方式相当于全部请求到网关,然后进行拦截,将部分分发到具体服务器上,从而保证域名相同,或者我们可以使用httpclient请求进行转发,但是这个比较占内存,或者使用设置请求头进行跨域这是我在项目中要到的问题,分布式事务,分布式锁,和跨域问题,这几个问题的遇到让我有很大的提升. |

1. 项目技术亮点分析

|  |
| --- |
| 你看  Sprin   1. **redis技术：**   redis是一种内存型、非关系型数据库；redis的数据会存储到硬盘上；它保存数据一种采用的是rdb(快照的形式进行数据的保存，是redis默认开启的方案，它将内存中的数据都保存到磁盘上，放到rdb文件中，重新启动服务器是将rdb的内容重新取出来,再回复到内存中。还有一种是aof默认是不开启的，因为他会频繁的操作磁盘，每秒钟同步一次，所以默认是不开启的。开启之后数据的恢复就是用aof。  Resid持久化方式：  RDB持久化  RDB 是在某个时间 点将数据写入一个临时文件，持久化结束后，用这个临时文件替换上次持久化的文件，达到数据恢复。  优点：使用单独子进程来进行持久化，主进程不会进行任何 IO 操作，保证了 redis 的高性能  缺点：RDB 是间隔一段时间进行持久化，如果持久化之间 redis 发生故障，会发生数据丢失。所以这种方式更适合数据要求不严谨的时候。  AOF持久化  Append-only file，将“操作 + 数据”以格式化指令的方式追加到操作日志文件的尾部，在 append 操作返回后(已经写入到文件或者即将写入)，才进行实际的数据变更，“日志文件”保存了历史所有的操作过程；当 server 需要数据恢复时，可以直接 replay 此日志文件，即可还原所有的操作过程。AOF 相对可靠，它和 mysql 中 bin.log、apache.log、zookeeper 中 txn-log 简直异曲同工。AOF 文件内容是字符串，非常容易阅读和解析。  优点：可以保持更高的数据完整性，如果设置追加 file 的时间是 1s，如果 redis 发生故障，最多会丢失 1s 的数据；且如果日志写入不完整支持 redis-check-aof 来进行日志修复；AOF 文件没被 rewrite 之前（文件过大时会对命令进行合并重写），可以删除其中的某些命令（比如误操作的 flushall）。  缺点：AOF 文件比 RDB 文件大，且恢复速度慢。  (2)mongoDB技术：MongoDB是一个面向文档的数据库，它并不是关系型数据库，直接存取BSON，BSON是为效率而设计的，它只需要使用很少的空间，这意味着MongoDB更加灵活，因为可以在文档中直接插入数组之类的复杂数据类型，并且文档的key和value不是固定的数据类型和大小，所以开发者在使用MongoDB时无须预定义关系型数据库中的”表”等数据库对象，设计数据库将变得非常方便，可以大大地提升开发进度。而且mongoDB能够存储非常大的数据量。刚好用在吐槽评论方面，用来存储用户对浏览文章后的评论。   1. 项目框架：项目采用得**spring boot+springcloud**微服务框架好处是 SpringCloud是基于SpringBoot的一整套实现微服务的框架。他提供了微服务开发所需的配置管理、服务发现、断路器、智能路由、微代理、控制总线、全局锁、决策竞选、分布式会话和集群状态管理等组件。最重要是，   跟spring boot框架一起使用的话，会让你开发微服务架构的云服务非常好的方便。SpringBoot 快速开发单个微服务，专注于快速、方便集成的单个微服务个体  而且SpringBoot 使用“习惯优于配置”，减少对jar包配置，可以单独的运行项目。  **SpringCloud**好处:Spring提供了一系列工具，可以帮助开发人员迅速搭建分布式系统中的公共组件（比如：配置管理，服务发现，断路器，智能路由，微代理，控制总线，一次性令牌，全局锁，主节点选举， 分布式session, 集群状态）。协调分布式环境中各个系统，为各类服务提供模板性配置。使用Spring Cloud, 开发人员可以搭建实现了这些样板的应用，并且在任何分布式环境下都能工作得非常好，小到笔记本电脑， 大到数据中心和云平台。  Spring Cloud官网的定义比较抽象，我们可以从简单的东西开始。Spring Cloud是基于Spring Boot的， 最适合用于管理Spring Boot创建的各个微服务应用。要管理分布式环境下的各个Spring Boot微服务，必然存在服务的注册问题。所以我们先从服务的注册谈起。既然是注册，必然有个管理注册中心的服务器，各个在Spring Cloud管理下的Spring Boot应用就是需要注册的client  Spring Cloud使用erureka server, 然后所有需要访问配置文件的应用都作为一个erureka client注册上去。eureka是一个高可用的组件，它没有后端缓存，每一个实例注册之后需要向注册中心发送心跳，在默认情况下erureka server也是一个eureka client ,必须要指定一个 server。  **SpringBoot**的好处：  1.良好的基因  因为SpringBoot是伴随着Spring 4.0而生的，boot是引导的意思，也就是它的作用其实就是在于帮助开发者快速的搭建Spring框架，因此SpringBoot继承了Spring优秀的基因，在Spring中开发更为方便快捷。  2.简化编码  ，比如我们要创建一个 web 项目，使用 Spring 的朋友都知道，在使用 Spring 的时候，需要在 pom 文件中添加多个依赖，而 Spring Boot 则会帮助开发着快速启动一个 web 容器，在 Spring Boot 中，我们只需要在 pom 文件中添加如下一个 starter-web 依赖即可。  3.简化配置  Spring 虽然使Java EE轻量级框架，但由于其繁琐的配置，一度被人认为是“配置地狱”。各种XML、Annotation配置会让人眼花缭乱，而且配置多的话，如果出错了也很难找出原因。Spring Boot更多的是采用 Java Config 的方式，对 Spring 进行配置。  4.简化部署  在使用 Spring 时，项目部署时需要我们在服务器上部署 tomcat，然后把项目打成 war 包扔到 tomcat里，在使用 Spring Boot 后，我们不需要在服务器上去部署 tomcat，因为 Spring Boot 内嵌了 tomcat，我们只需要将项目打成 jar 包，使用 java -jar xxx.jar一键式启动项目。  另外，也降低对运行环境的基本要求，环境变量中有JDK即可。  5.简化监控  我们可以引入 spring-boot-start-actuator 依赖，直接使用 REST 方式来获取进程的运行期性能参数，从而达到监控的目的，比较方便。但是 Spring Boot 只是个微框架，没有提供相应的服务发现与注册的配套功能，没有外围监控集成方案，没有外围安全管理方案，所以在微服务架构中，还需要 Spring Cloud 来配合一起使用。   1. Elasticsearch搜索技术：我们项目的搜索方式采用的是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，是当前流行的企业级搜索引擎。，能够达到实时搜索，稳定，可靠，快速，安装使用方便。而且可以用于对海量的数据进行近实时的数据分析，而且Elasticsearch是实时性的搜索引擎，可以快速准确的搜索出用户想要的信息，但是有时候搜索出来的信息不能特别精准，所以我们用到了IK分词器，IK能够帮用户把想要的信息进行更加细分化。 |
|  |

1. 项目答辩中亮点简历审核评定标准

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考核内容** | **简历样式是否规范** | **项目内容是否完整** | **项目中是否有亮点** | **技术难点解决办法是否高效** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |
| 简历是否合格 | | | | 合格 | 不合格 |

1. 项目答辩成绩评定标准

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **考核内容** | **亮点简历与项目描述是否匹配** | **项目流程是否清晰** | **学生讲解是否流畅** | **项目介绍是否到位** | **技术亮点考核是否合格** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |  |
| 项目答辩是否合格 | | | | 合格 | 不合格 |  |