# B简历：

### 基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| 基本情况 | | | |
| **姓名** | 刘玉康 | 性别 | 男 |
| **出生日期** | 1997-05-20 | 属相 | 牛 |
| **现住地** | 高新区秦庄家园 |  |  |
| 大学情况 | | | |
| **学校** | 山东现代学院 | 专业 | 计算机应用技术 |
| **入学时间** | 2014-9 | 毕业时间 | 2018-7 |
| **专业课** | **数据结构课程设计、数据库原理课程设计、数据库及应用、操作系统实训、Java程序设计** | | |
| **学历** | **本科** | 学位 | **学士学位** |
| **学校地址** | 济南经十东路 | 校长 | 刘春静 |
| **去学校的方式：** | **在火车站去二环东路或燕山立交倒车** | | |
| **上小学日期：** | **2002年9月 6岁六年** | | |
| **上初中日期：** | **2008年9月 12岁三年** | | |
| **上高中日期：** | **20011年9月 15岁三年** | | |
| **上大学日期：** | **2014年9月 18岁四年** | | |
| **毕业时间：** | **2018年 7月 22岁** | | |
| **在校学生：** | **一万多** | | |
| **有教师：** | **不太清楚** | | |

|  |  |
| --- | --- |
| 第一家公司 ： | 济南骐源物联网（外包） |
| **地址：** | 历城区浪潮工业园 |
| **入职时间：** | **2018年7月初** |
| **离职时间：** | **2019年11月** |
| **公司性质：** | **民营** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **陈金波** |
| **薪资方面：** | **刚刚毕业转正以后工资是6K左右，过完年大概四月份涨到7K左右** |
| **路线：** | 907 .310 |

|  |  |
| --- | --- |
| **第二家公司情况：** | **济南悦码** |
| **公司地址**： | **济南齐鲁软件园 世纪财富中心** |
| **公司规模：** | **我们项目组7人，一个项目经理，3名开发人员，1名前端，1个测试，1个运维** |
| **公司性质：** | **民营** |
| **公司行业：** | **医药营销推广平台** |
| **去公司路线：** | **电瓶车** |
| **入职时间：** | **2019年11月** |
| **离职时间：** | **2021年4中旬** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **蔡林** |
| **税前：** | **10.5K** |
| **税后：** | **10K** |
| **缴税：** | **不到一千K左右(具体原因不清楚当时公司就是这样扣的税)** |
| **期望薪资：** | **10K** |

|  |
| --- |
| 四年涨薪： **2018年工作的时候是5.6K，税后大概5.4K左右，**  **转正6K左右**  **2019上半年涨了1K左右 ，总7K左右。**  **2019年8月跳槽涨了2K左右 总9K**  **2019年12月涨0.5K总 9.5K左右**  **2020年9月工作一年涨了1K多**  **2021年4月初离职时工资为10。5K** |

**面试常见问题：**

|  |
| --- |
| **你为什么从上家公司离职？** |
| **想换个工作环境，不断的提高充实自己** |
| **你交社保了吗？为啥没交？** |
| **没交**  **在上家公司的时候，人事说咱们公司的五险一金如果要上的话都是从自己工资里面扣的，当时感觉没啥必要也就没上。** |
| |  | | --- | | **三个词形容自己**： 交流表达能力强，团队协作能力强，抗压自学能力强 (**可以综合自己自学的知识表达**) | | **你的优缺点是啥？** |   **优点:**  **抗压能力强（有一次项目就要上线了，连着加班了一周，每天都到夜里很晚，但我最后也是成功完成任务。），**  **乐于分享，**  **团队协作能力强**  **缺点: 我这个人说话比较直，直男嘛，像在和团队成员探讨问题的时候，容易得罪人，**  **不过现在已经改进的差不多了** |
| **你知道五险一金都是啥吗？** |
| **养老保险、医疗保险、失业保险、工伤保险、生育保险，和住房公积金。** |
| **你能接受加班吗** |
| **能** |
| **你能接受出差吗** |
| **能** |
| 你期望薪资多少？你上家工资多少？税后拿到手的有多少？扣了多少钱的税？ |
| 期望薪资 16K 上家工资15K 税后14K左右 扣了1K左右的税 |
| 你的五年规划是啥？ |
| **前2,3年 继续加强自己的技术功底，**  **然后朝着项目经理(技术经理,产品经理)方面发展** |
| 是统招吗 是是本科吗 是能在学信网查到吗 我当时上的是民办大学，能在民教网查到！ |
| 你们公司还有什么福利？多少薪： 13薪  **车补**  220左右  **项目奖金**  8k-10k，做完项目，收完尾款的时候发。  **年终奖**  就是1个月的工资，  最后一个月和工资一块儿发/年后发  **电话补助**  200  **饭/餐补**  **25**/天  **电脑补**  **220** |

**公司内其他部门：**

**技术部，财务部，行政部**

**---------------------------------------------------------------------**

**其他问题：**

**笔试问题：我工作这么多长的时间了，理论上的知识忘得差不多了，你和你们经理联系一下直接面试吧，不行的话那么我就去直接去下一家！**

**---------------------------------------------------------------------**

### 思路:

springBoot的理解 --> 项目中的运用细节 -->

--------------------------------------------------------------------------------------------

### 介绍项目：

你好，我叫刘玉康，是来应聘咱们公司的Java开发岗位的。

我原来在济南工作，因为一些原因来到咱们郑州来应聘工作。我在之前的公司做的是医药营销推广管理平台。这个项目分为了药事管理系统和管家管理系统。使用了 **Spring Boot+springCloud 来做的微服务项目的开发。**

说到SpringBoot，我先给您简单介绍下SpringBoot的运行原理。

我们从springboot项目的的启动类中可以看到最核心的两行代码，@SpringBootApplication和SpringApplication.run方法。

在 @SpringBootApplication 的内部包含了3个注解

@Configuration用于定义配置类，一般加在主类上

@EnableAutoConfiguration允许springboot自动配置注解加载到IoC容器中。

@ComponentScan它的作用就是让spring Boot扫描到Configuration类并把它加入到程序上下文。

简单点说，springboot就是一个微服务，主要做接口开发，

接口开发：同一个路径可以发送不同的请求方式

  知道前台传什么值，后台要返回什么样的值。

**Springboot的优点**

（1）所有 Spring 开发提供一个更快更广泛的入门体验。

①入门快

（2）零配置。没有多余的代码生成和XML 强制配置，遵循“约定大于配置” 。

　　　①“约定大于配置”（内部有默认的配置，不用特意去集成）

（3）集成了大量常用的第三方库的配置， Spring Boot 应用为这些第三方库提供了几乎可以零配置的开箱即用的能力。 ①开箱即用

（4）提供一系列大型项目常用的非功能性特征，如嵌入式服务器①内部嵌入tomcat、安全性、度量、运行状况检查、外部化配置等。**使用嵌入式的Servlet容器，应用最终可以打成Jar包的形式独立运行**

（5）Spring Boot 不是Spring 的替代者，Spring 框架是通过 IOC 机制来管理 Bean 的。

SpringBoot的五大组件有，包括：Eureka注册中心、Ribbon负载均衡、Hystrix容断器、zuul网关、feign调用中心。

首先我们说一下**eureka注册中心**，主要是来注册保存我们的服务的地址列表，为消费者提供订阅服务的功能，在pom文件中引入Eureka的依赖，之后在启动类中加入 @EnableEurekaServer的注解证明该项目的作用是作为注册中心使用，并且在对应的application.properties中指明Eureka注册中心服务器的地址，方便提供者和消费者对其进行访问，并且服务会主动向注册中心发送续约请求，在一段时间不续约后就认为该服务已经挂掉了，就需要从注册中的地址列表中剔除。这个检查剔除服务的时间间隔可以在注册中心上配置。为了增强注册中心的高可用性，我们做了Eureka集群，防止因为单台Eureka导致的单点故障问题。

如果没有注册中心，客户端通过ip地址访问服务端，如果服务越来越多，调用也会越来越复杂，一旦ip地址发生变动，则所有用到的地方都需要发生改变，维护起来会特别麻烦；如果服务端宕机了，所有访问的客户端也都会受到影响。通过注册中心可以让服务端注册到注册中心上去，并且可以进行负载均衡；这样就可以达到高可用以及提高并发的 目的。客户端 也不需要直接调用服务端，而是通过注册中心获取地址列表，从地址列表中选一个服务端接口进行调用，后续的维护各方面也会比较方便。

在项目中要实现微服务之间调用我们就要用到Ribbon负载均衡。它的负载均衡是软负载,也就是客户端服务器从Eureka Server拉取已注册的服务信息,然后根据负载均衡策略,直接发送请求,这整个过程都是在客户端完成的,并不需要注册中心的参与,SpringCloud在结合了Ribbon的负载均衡的实现中,封装增加了HTTP Client,是spring提供的对httpClient进行的封装，Ribbon的主要组件有ServerList,IRule,ServerListFilter等,他的整个流程是这样子的,首先,通过ServerList,获取所有的可用服务列表,然后通过ServerListFilter,过滤掉一部分地址,最后剩下的地址中,通过IRule选择一个实例,作为最终目标结果，**Ribbon就是将用户的请求平摊的分配到多个服务上，在微服务或分布式集群中经常使用。**

hystrix熔断器，在整个微服务搭建的过程中有可能发生微服务雪崩的问题，就是A->B->C,如果C这个微服务出现问题，就会影响到B,同样B出现问题，就会导致A也出问题，最后导致所有的微服务都不可用。咱们都知道Hystrix可以进行熔断，降级和资源隔离。熔断其实就可以理解成咱们家里面的保险丝。保险丝是为了保证在高负载情况下电路的安全，如果某个电路出现了负载过大情况，保险丝就会熔断，从而保证其他电路不受影响。Hystrix的熔断功能比较强大，它还可以进行自动检测并修复。（作为了解：咱们都知道Hystrix可以进行熔断，降级和资源隔离。熔断其实就可以理解成咱们家里面的保险丝。保险丝是为了保证在高负载情况下电路的安全，如果某个电路出现了负载过大情况，保险丝就会熔断，从而保证其他电路不受影响。Hystrix的熔断功能比较强大，它还可以进行自动检测并修复。

具体的原理是这样的，默认情况下在10s内，每当20个请求中，有50%失败时，就会触发熔断，导致hystrix从闭合状态切换到打开状态，这样就不需要每次都去调用远程微服务而是直接走调用者本地的降级方法。5秒后hystrix会进入到半打开状态，这时如果有新的请求再过来hystrix就会尝试再次进行远程微服务调用，如果调用成功就会进入到闭合状态，如果不成功就会再次进入到打开状态，继续进行熔断。这个就是hystrix的自动检测并修复的能力。）

zuul网关 我们的微服务架构，服务除了内部相互之间调用和通信之外，最终要以某种方式暴露出去，才能让外界系统访问到。不同的微服务一般会有不同的网络地址，那么这里就会遇到一个问题，APP/Browser怎么去访问这些后端的服务?客户端在访问这些微服务时必须记住几十甚至几百个地址，这对于客户端方来说太复杂也难以维护，因此，我们需要一个微服务网关，介于客户端与服务器之间的中间层，所有的外部请求都会先经过微服务网关。

（作为了解：继承于ZuulFilter的类，重写里面的filterType()方法filterOrder()方法，shouldFilter()方法，还有一个最核心的用来写具体业务逻辑的run()方法。其中filterType（）：的作用是用来返回一个字符串，指明该过滤器的类型，经常用到的有pre类型，就是在请求被发送到微服务之前调用；还有post类型，就是微服务执行完后再执行该过滤器

filterOrder()：返回一个int类型的值，用来指明该过滤器的执行顺序，数字越小表示优先级越高，就越先执行，shouldFilter()方法返回一个boolean值，用来指明该过滤器是否执行，true表示执行,false表示不执行。

run方法中就是之前说的，用来放具体的处理逻辑。）

我在项目中负责的是登录和订单模块

**调查问卷是**

**JWT的登录认证**

在我们做单体架构时，登录之后的用户信息是保存在服务端session中的，每次发送请求都会自带cookie值进行验证。随着我们现在用分布式架构，session和cookie已经不适用了。而我们又想保证用户信息的安全，所以使用到了JWT进行登录验证。JWT可以理解为加密工具，在我们的用户登录成功之后，把用户对象放进去，用JWT声明token令牌，把值传到前台，并存入sessionStorage，这样我们每次发送请求时都自带token值，防止越权访问。我们现在后台用的就是分布式架构，通过zuul网关去验证token，zuul网关就相当于拦截器+过滤器，我们创建一个类去继承zuulFilter过滤器，重写方法，在run方法中我们通过request.getParameter获取token值，通过JWT去解密验证token，如果token为空，则用户没有进行登录，如果token值解密错误，则用户非法访问或者token值被人为改过。当然token也有失效时间，默认是30分钟，这时解密的token值可能就为空了，但能确定不是非法访问。如果认为还是不安全，我们也可以在登录成功之后用base64再进行加密。

补充：①**sessionStorage与localStorage的区别？**

localStorage的生命周期是永久性的，即使关闭浏览器，也不会让数据消失，除非主动的去删除数据；

sessionStorage 的生命周期是在浏览器关闭前，在整个浏览器未关闭前，其数据一直都是存在的；

sessionStorage不在不同的浏览器窗口中共享；localstorage在所有同源窗口中都是共享的；

**（restful接口）**

之前工作的过程中，我们的经理给我说，我们项目需要给前台提供restful接口,当时对这个restful只是了解并没有真正的在项目中使用过,所以下去我也查阅了许多关于restful接口开发的文档,通过文档得知,restful是ROA面向资源编程的一种设计风格，restful不是一项新技术,就是新瓶装旧酒,其实就是符合restful风格的url,都被称之为restful类型的接口,因为在互联网中可以通过url去访问任意的资源，restful是基于http协议现有的Get动作进行查询，Post动作进行增加，Put动作进行修改，Delete动作进行删除。

**JVM虚拟机调优**

我们的linux服务器,安装centos7.0, 64位的操作系统和64位的软件，配置了8个cpu,128G内存。

我们在jvm优化的时候是这样做的首先JVM将内存划分为：年轻代年老代永久代（方法区）其中年轻代和年老代属于堆内存，永久代不属于堆内存，有虚拟机直接分配。年轻代：年轻代用来存放JVM刚分配的Java对象年老代：年轻代中经过垃圾回收没有回收掉的对象将被放到年老代永久代：永久代存放Class类、Method方法元信息，其大小跟项目的规模、类、方法的量有关，一般设置为256M就足够，预留30%的空间。通过-XX:PermSize=256M -XX:MaxPermSize=256M 来设置永久代（方法区）的大小。

jvm的垃圾回收算法有[GC]

串行算法（单线程）

并行算法

并发算法

吞吐量优先的并行收集器

响应时间优先的并发收集器

我们的项目设置的是响应时间优先的并发收集器，

将堆大小通过 -Xms -Xmx设置为4G,将年轻代通过 -Xmn 设置为2g,

设置年老代为并发收集，当时设置的是运行6次GC以后对 内存空间进行压缩、整理。打开对年老代的压缩，可以消除碎片。

**Tomcat调优**

1. 启用线程池，默认的tomcat没有启用线程池，在tomcat中每一个用户请求都是一个线程，所以可以使用线程池提高性能。我们在项目中将maxThreads[最大线程数]设置为1000，将排队数acceptCount和maxThreads设置相等。当tomcat的线程数达到maxThreads后，新的请求就会排队等待，超过排队数的请求会被拒绝。
2. 使用64位的tomcat和jdk，禁用AJP协议。
3. 开启APR通讯模式，支持高并发。因为默认tomcat采用的是性能最低的BIO【阻塞IO】模式。
4. tomcat中设置JVM参数 通过-server开启server模式 通过-Xms4000m和-Xmx4000m设置初始堆大小和最大堆大小，通常将两个值设置为一样，避免堆空间不断增大和缩小所带来的性能损耗。

jvm的 内存结构：

方法区【持久代】

堆【java对象】

栈【基本数据类型，对象引用（指针/对象名）】

本地方法栈

计数器

**（sql优化方面）**

1、对于大数据量的分页不要用常规的 limit 开始位置，每页条数；这样越靠后查询时间越长，效率越低，可以使用id>=的方法来替代，比如select 字段名 From 表名 where id >= (select id from 表名 limit 开始位置，1) Limit 每页条数

2、SELECT语句中避免使用'\*’，只查询需要返回的字段，这样可以减少解析sql语句的时间，以及减少 带宽，cpu,内存，io等 各方面的消耗。

3、 外键必须加索引，这样可以提升多表联查的性能

1. 通过冗余字段避免多表连查，从而提高性能。比如在商品表中可以不仅有分类的id，还可以加上分类的名字，这样就不用为了显示分类名字而去进行多表联查了。又比如在会员表中不仅有地区的id，还可以有地区的名字这样也可以避免为了显示地区名而进行多表联查。

5、在 where，order by 以及GROPU BY 涉及的列上建立索引，为了避免全表扫描。

6、In和exists的区别：主要是就是驱动顺序的改变，这是性能变化的关键，如果是exists，那么以外层表为驱动表，先被访问，如果是IN，那么先执行子查询，以内层表为驱动表。所以IN适合于外表大而内表小的情况；EXISTS适合于外表小而内表大的情况。

7、通常要用小表去驱动大表，这样可以提高性能，在left join 中 左边的表为驱动表，所以应该让左边的表尽可能的为小表，在right join 中 右边的表为驱动表，所以应该让右边的表尽可能的小，在 inner join 中 mysql会自动选择较小的表为驱动表。

8、避免在 where 子句中对有索引的字段进行运算,这会导致索引失效，从而进行全表扫描。比如select 字段名 from 表名 where 库存量+1>=10,就需要改为 select 字段名 from 表名 where 库存量>=9

9、避免在where字句中对字段进行null值判断，这样会导致即便字段上有索引也会进行全表扫描，好的处理方式就是给字段设置默认值。

什么是索引

通过索引可以提高查询的速度，它就相当于书的目录；索引不是越多越好，因为索引会占用额外的空间，并且在进行增删改的时候还需要额外维护索引。创建索引的原则：给频繁查询的字段上创建索引根据当前字段的业务含义，来区分是创建唯一索引还是创建普通索引考虑到索引不是越多越好，也可以根据业务情况创建复合索引，

想要使其复合索引起作用，在查询的时候需要按照当时创建

复合索引的字段的顺序来,也就是最左匹配原则。

执行计划:通过explain查看sql语句的执行计划key:代表mysql实际使用的索引。type:这个字段比较重要，它表示MySQL在表中找到所需行的方式：

Const：主键在where条件中

Range：扫描部分索引，常见于between、<、>等的查询

Index：代表扫描全部 索引树

ALL：代表扫描全 表/全表扫描/不走索引。

创建索引-- 普通索引create index index\_status on t\_log(status);

-- 创建唯一索引create unique index index\_username on t\_user(userName);

https://www.cnblogs.com/summer0space/p/7247778.html-- 创建复合索引create index index\_status\_logincount on t\_user(status, loginCount);

索引的类型

【索引特性上来说】

唯一索引 普通索引

【字段的个数来说】

单个索引/单一索引 复合索引/组合索引

【字段的个数，如果对多个字段同时加索引则称之为复合索引】

聚集索引和非聚集索引：

聚集索引：主键就属于聚集索引。在一张表上聚集索引只能有一个。

非聚集索引: 除了主键以外的唯一索引，普通索引。在一张表上非聚集索引可以有多个。

唯一索引和普通索引的区别：

唯一索引所在字段中的值必须是唯一的。主键字段会自动创建唯一索引。

创建索引，不一定就能使用上索引；想要索引生效必须满足两个条件，第一，该字段上有索引；第二，所写的sql语句必须支持索引。

什么时候索引不起作用/什么时候索引会失效?

1.如果在有索引的字段上进行运算，那么索引失效，会进行全表扫描。

2.在使用like的时候如果%在前面则索引失效

全文检索技术

java:[开源的搜索引擎技术]

Lucene：

Solr【是对lucene的封装】：企业级的搜索引擎技术

ES【是对lucene的封装】：互联网级的搜索引擎技术

3.避免在查询条件中使用is null,可以采用默认值的方法来解决

大批量数据的插入/更新?

1.一条条插入

2.分段 批量 提交【推荐】【减少和数据库的交互次数】

3.批量插入和数据库交互一次

删除?delete ... in

**Redis**

1. **介绍**

Redis 是一个开源的、使用 C 语言编写的，采用key-value（键值对）的形式的，非关系型数据库。

**②．持久化方案**

redis提供两种方式进行持久化，一种是RDB持久化（原理是将Reids在内存中的数据库记录定时dump到磁盘上的RDB持久化），另外一种是AOF（append only file）持久化（原理是将Reids的操作日志以追加的方式写入文件）。

优缺点的对比：恢复的快慢，回复数据的完整性

**③．数据类型**

Redis支持丰富的数据类型：string字符串，list 列表（消息队列），hash散列值，set 集合，zset 有序集合。

**④．使用场景**

热点数据的缓存，

限时业务的运用，

计数器相关问题，

延时操作，

分布式锁

**⑤．集群的三种方案**

主从复制

主从复制是指将一台Redis服务器的数据，复制到其它的Redis服务器，实现读写分离。用来解决数据冗余，故障恢复，负载均衡，高可用基石。

哨兵

基于Redis主从复制，主要作用便是解决主节点故障恢复的自动化问题，进一步提高系统的高可用性。哨兵模式拥有的功能监控，自动故障转移，通知等

集群

Redis集群采用无中心结构，集群由多个节点(Node)组成，Redis的数据分布在这些节点中。一般用于数据量很大的情况下

**⑥．Redis缓存出现的问题**

**雪崩**

缓存雪崩表示在某一时间段，缓存集中失效，导致请求全部走数据库，有可能搞垮数据库，使整个服务瘫痪。

原因：redis服务器挂掉了或者对缓存数据设置了相同的过期时间

解决：可以实现redis的高可用，采用哨兵模式集群。或者设置随机的过期时间，避免在同一时间集中过期

**击穿**

请求的数据大量的没有获取到缓存，导致走数据库，有可能搞垮数据库，使整个服务瘫痪。

解决：针对在数据库中找不到记录的，将该空数据存入缓存中，当然一般会设置一个较短的随机的过期时间。

**穿透**

缓存击穿表示某个key的缓存非常热门，有很高的并发一直在访问，如果该缓存失效，那同时会走数据库，压垮数据库。

解决：使用互斥锁，通过redis的setnx实现互斥锁。

## ****MongDB****

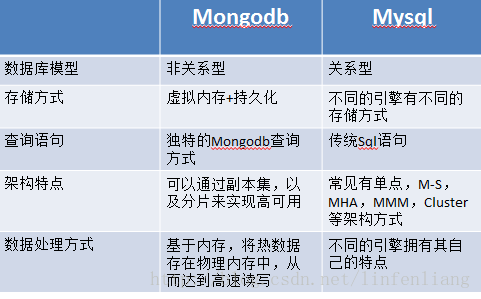
### 什么是mongodb

* 1. MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。
  2. MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。

### MongoDB的特点

* 1. 文档数据类型
     1. SQL类型的数据库是正规化的，可以通过主键或者外键的约束保证数据的完整性与唯一性，所以SQL类型的数据库常用于对数据完整性较高的系统。MongoDB在这一方面是不如SQL类型的数据库，且MongoDB没有固定的Schema，正因为MongoDB少了一些这样的约束条件，可以让数据的存储数据结构更灵活，存储速度更加快。
  2. 即时查询能力
     1. MongoDB保留了关系型数据库即时查询的能力，保留了索引（底层是基于B tree）的能力。这一点汲取了关系型数据库的优点，相比于同类型的NoSQL redis 并没有上述的能力。
  3. 复制能力
     1. MongoDB自身提供了副本集能将数据分布在多台机器上实现冗余，目的是可以提供自动故障转移、扩展读能力。
  4. 速度与持久性
     1. MongoDB提供了Journaling日志的概念，实际上像mysql的bin-log日志，当需要插入的时候会先往日志里面写入记录，再完成实际的数据操作，这样如果出现停电，进程突然中断的情况，可以保障数据不会错误，可以通过修复功能读取Journaling日志进行修复。
  5. 数据扩展
     1. MongoDB使用分片技术对数据进行扩展，MongoDB能自动分片、自动转移分片里面的数据块，让每一个服务器里面存储的数据都是一样大小。

### ****MongoDB与MySQL的区别****



### Mongodb常用的命令

* 1. 切换数据库 sue dbs
  2. 插入语法
     1. db.users.insert({username:"smith"})
     2. db.users.save({username:"smith"})
     3. 区别若新增的数据中存在主键 ，insert() 会提示错误，而save() 则更改原来的内容为新内容。
  3. 查找语句
     1. db.users.find()
     2. db.users.count()
  4. 集合的相关操作

db.createCollection() 创建集合

db.集合名称.drop() 删除集合

show collections( tables) 查看所有集合

db.集合名.insert(文档) 集合添加数据 save insertone insermany

* 1. 文档的操作

db.集合名.insert(文档) 集合添加数据 save insertone insermany

db.集合名.remove

db.集合名.update

db.集合名.find()

## ****Nginx****

### ****Nginx是什么****

* 1. 在服务器集群中，Nginx起到一个代理服务器的角色（即反向代理），为了避免单独一个服务器压力过大，将来自用户的请求转发给不同的服务器。

### Nginx负载策略

* 1. 轮询 这也是nginx的默认负载策略
  2. Weight 这是权重策略 根据权重进行分配 类似7,3开
  3. ip\_hash IP策略 就是所用户访问了一台服务之后就一直访问这个服务 可以保证session回话
  4. least\_conn 这是最少链接的策略 也就是说那台服务器连接数少就分配到那一台

## ****事务****

面试话术：介绍事务，事务的四个特性，可说五个隔离级别和七个传播特性。如果要介绍，就说一下默认常用的，别的不常用就没做过多的了解

事务指逻辑上的一组操作，组成这组操作的各个单元，要不全部成功，要不全部不成功。事务是为解决[数据安全](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%AE%89%E5%85%A8&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)操作提出的，事务控制实际上就是控制数据的安全访问。

**①．事务的4个特性（ACID）：**

1) 原子性（atomicity）：事务是数据库的逻辑工作单位，而且是必须是原子工作单位，对于其数据修改，要么全部执行，要么全部不执行。

2) 一致性（consistency）：事务在完成时，必须是所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。（实例：转账，两个账户余额相加，值不变。）

3) 隔离性（isolation）：一个事务的执行不能被其他事务所影响。

1. 持久性（durability）：一个事务一旦提交，事物的操作便永久性的保存在DB中。即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。
2. **七个传播行为**

 该特性是保证事务是否开启，业务逻辑是否使用同一个事务的保证。当事务在传播过程中会受其影响。其传播特性包括：

Ⅰ、PROPAGATION\_REQUIRED：如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中。这是最常见的选择。（默认 常用）

Ⅱ、PROPAGATION\_REQUIRES\_NEW：新建事务，如果当前存在事务，把当前事务挂起。（常用）

Ⅲ、PROPAGATION\_SUPPORTS：支持当前事务，如果当前没有事务，就以非事务方式执行。

Ⅳ、PROPAGATION\_MANDATORY：支持当前事务，如果当前没有事务，就抛出异常。

Ⅴ、PROPAGATION\_NOT\_SUPPORTED：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

Ⅵ、PROPAGATION\_NEVER：以非事务方式执行，如果当前存在事务，则抛出异常。

Ⅶ、Propagation.NESTED如果一个活动的事务存在，则运行在一个嵌套的事务中. 如果没有活动事务, 则按TransactionDefinition.PROPAGATION\_REQUIRED 属性执行。需要JDBC3.0以上支持。

1. **五个隔离级别**

Ⅰ、ISOLATION\_DEFAULT：这是一个PlatfromTransactionManager默认的隔离级别，使用数据库默认的事务隔离级别.

另外四个与JDBC（数据库）的隔离级别相对应；

Ⅱ、ISOLATION\_READ\_UNCOMMITTED（读未提交）：这是事务最低的隔离级别，它充许别外一个事务可以看到这个事务未提交的数据。

这种隔离级别会产生脏读，不可重复读和幻像读。

Ⅲ、ISOLATION\_READ\_COMMITTED（读已提交） ：保证一个事务修改的数据提交后才能被另外一个事务读取。另外一个事务不能读取该事务未提交的数据。

这种事务隔离级别可以避免脏读出现，但是可能会出现不可重复读和幻像读。

Ⅳ、ISOLATION\_REPEATABLE\_READ（可重复读）：这种事务隔离级别可以防止脏读，不可重复读。但是可能出现幻像读。

它除了保证一个事务不能读取另一个事务未提交的数据外，还保证了避免下面的情况产生(不可重复读)。

Ⅴ、ISOLATION\_SERIALIZABLE（序列化）：这是花费最高代价但是最可靠的事务隔离级别。事务被处理为顺序执行。

除了防止脏读，不可重复读外，还避免了幻像读。

**④．如果不考虑隔离性引发安全性问题:**  
 脏读 :一个事务读到了另一个事务的未提交的数据  
 不可重复读 :一个事务读到了另一个事务已经提交的 update 的 数据导致多次查询结果不一致.  
 虚幻读 :一个事务读到了另一个事务已经提交的 insert 的数据导致多次查询结果不一致.

**分布式事务**

1. **是什么：**

前提：1支持事务 2 可以直接操控数据库

就是一次大的操作由不同的小操作组成，这些小的操作分布在不同的服务器上，且属于不同的应用，分布式事务需要保证这些小操作要么全部成功，要么全部失败。本质上来说，分布式事务就是为了保证不同数据库的数据一致性。

**②．CAP：**

CAP 理论在分布式系统中

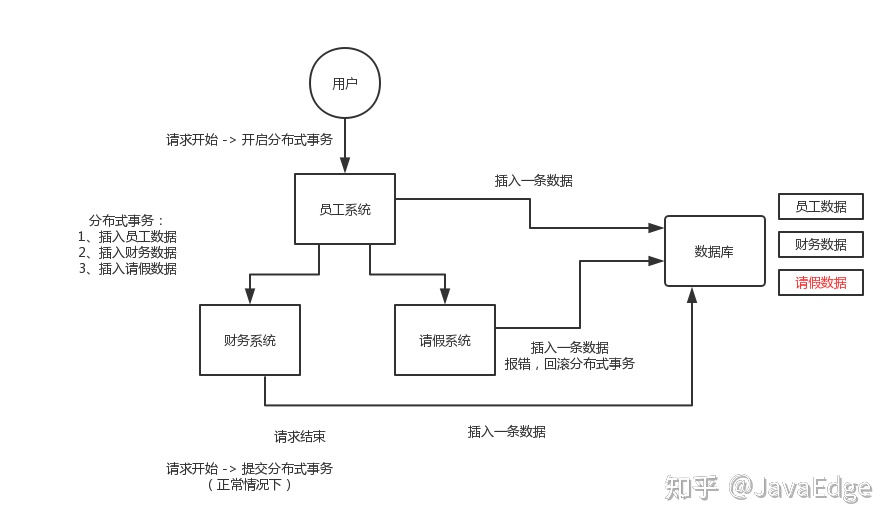
一致性：分布式环境下多个节点的数据是否强一致

可用性：分布式服务能一直保证可用状态。当用户发出一个请求后，服务能在有限时间内返回结果

分区容忍性：特指对网络分区的容忍性

其中，分区容忍性又是不可或缺的。

**③．分布式系统里的事务图解**



**④．使用场景：**

最经典的场景就是支付了，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

**⑤．Seate**

Seata 是一款开源的分布式事务解决方案，致力于提供高性能和简单易用的分布式事务服务。Seata 将为用户提供了 AT、TCC、SAGA 和 XA 事务模式，为用户打造一站式的分布式解决方案。

一个典型的分布式事务过程

TM 向 TC 申请开启一个全局事务，全局事务创建成功并生成一个全局唯一的 XID；

XID 在微服务调用链路的上下文中传播；

RM 向 TC 注册分支事务，将其纳入 XID 对应全局事务的管辖；

TM 向 TC 发起针对 XID 的全局提交或回滚决议；

TC 调度 XID 下管辖的全部分支事务完成提交或回滚请求。

**⑥．分库分表**

（1）水平切分，主要解决单表过大造成的性能问题，单表过大造成的单服务器空间问题。

优点：

表关联基本能够在数据库端全部完成；

不会存在某些超大型数据量和高负载的表遇到瓶颈的问题；

应用程序端整体架构改动相对较少；

事务处理相对简单；

只要切分规则能够定义好，基本上较难遇到扩展性限制；

缺点：

切分规则相对更为复杂，很难抽象出一个能够满足整个数据库的切分规则；

后期数据的维护难度有所增加，人为手工定位数据更困难；

应用系统各模块耦合度较高，可能会对后面数据的迁移拆分造成一定的困难。

（2）垂直切分，主要解决表与表之间资源争用问题，锁争用机率小，实现核心与非核心的分级存储，如UDB登陆库拆分成一级二级三级库，数据库同步压力问题。

优点：

数据库的拆分简单明了，拆分规则明确；

应用程序模块清晰明确，整合容易；

数据维护方便易行，容易定位；

缺点：

部分表关联无法在数据库级别完成，需要在程序中完成；

对于访问极其频繁且数据量超大的表仍然存在性能平静，不一定能满足要求；

分布式事务处理复杂；

依然存在单表数据量过大的问题（需要水平切分）

**Docker**

1. **什么是Docker**

docker就是用来解决运行环境和配置问题的容器引擎，其中包括了镜像，容器，仓库，

其中镜像是 Docker 运行容器的前提，仓库是存放镜像的场所，库分为本地仓库，官方仓库，和私有仓库。

容器是一种轻量级、可移植的软件打包技术。

优点：

能高效地构建应用、对于运维开发来说，能快速的交付和部署、高效的资源利用轻松的迁移扩展、简单的更新管理

1. **Docker的命令**

帮助命令：docker info

查看镜像：docker images

删除镜像：docker rmi 镜像

搜索镜像：docker search 镜像名

拉取镜像：docker pull 镜像名

按照镜像名称搜索镜像docker search

创建容器：docker run -id –name 容器名 -p 8080:8080 镜像名:版本号

删除容器：docker rm -f 容器ID

查容器：docker ps 查询当前运行 -a 查询运行和不运行

停止容器：docker stop 容器id（名称） 停止容器

启动容器：docker start 容器id（名称）

退出容器 Ctrl+P+Q

1. **制作镜像**

commit

build（Dockerfile）

编写Dockerfile文件

语法 ：

FROM 指定基础镜像

MAINTAINER 指定镜像维护者信息

RUN 用于执行指定脚本命令

CMD 指定启动容器时执行的命令

EXPOSE 指定容器暴露的端口

ENV 指定环境变量

ADD 将文件从宿主机复制到容器指定位置，同时对压缩文件有自动解压功能

COPY 将文件从宿主机复制到容器指定位置

ENTRPOINT 设置容器启动时需要运行的命令

WORKDIR 为后续的如RUN、CMD、ENTRYPOINT、COPY、ADD指定工作目录

## ****什么是maven****

maven是用来构建项目和管理依赖的工具。构建项目就是创建项目，而且可以在项目编码完成之后,对项目进行编译、测试、打包、部署操作。管理依赖呢，就是对项目中所需要的jar包进行管理。

### **1.**   **为什么要使用Maven(Maven的好处)**

1.    使用maven创建出来的项目具有统一的结构

2.    规范jar包的下载

3.    节约项目的空间，让项目不是那么臃肿

4.    Maven可以把一个jar包所依赖的jar也一并下载下来

5.    通过maven可以对项目进行编译、测试、打包、部署操作

### **2.**   **Maven仓库分为哪几种?**

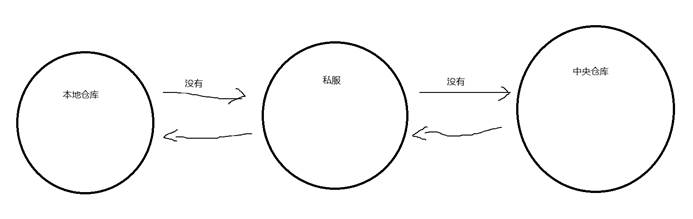
**本地仓库**:自己电脑上的一个文件夹，本地仓库默认设在了${user.home}/.m2/repository，本地仓库设在C盘有一点不好的地方，就是重装操作系统之后C盘被格式化了，之前仓库里的jar包就没了，所以咱们通常会修改本地仓库的位置，在maven安装目录下的conf/setting.xml中可以更改本地仓库的位置。

**中央仓库:**maven官方维护的一个仓库，里面包含了市面上绝大部分开源项目的jar包。但是中央仓库服务器所在遥远的美国，由于我们都懂的原因，访问速度比较慢，还有可能会中断。想要连接中央仓库必须得保证电脑联网了。

**第三方仓库(也就是私服)**:是架设在局域网的仓库，通常情况下公司会搭建自己的Maven私服，这样可以保证开发人员拿到的jar包都是统一的。而且我们班级之前也老师搭建的也有maven私服，大家都连接这个maven私服就可以下载jar包了。

### **3.**   **Maven获取jar包的顺序**

当本地仓库没有你需要的jar包时而且你也没有配置私服仓库地址时，maven会去中央仓库下载jar包。如果你的电脑上配置有私服仓库地址，则maven会去私服里下载jar包，如果私服也没有jar包，则私服会去中央仓库下载jar包。



### **4.**   **Setting(塞挺).xml和pom(炮木).xml**

settings.xml是maven的全局配置文件。Settings.xml中包含类似**本地仓库位置**、**修改远程仓库服务器**、**认证信息**等配置。而**pom.xml**主要描述了**项目的maven坐标**，**依赖关系**以及**其他所有的项目相关因素**，是项目级别的配置文件。

### **5.**   **Maven常用命令(掌握5个)**

mvn clean(科林)    清空产生的项目( target里)

mvn compile(凯木排欧) 编译源代码

mvn package(排K继)     打包

mvn test           运行测试

mvn test-compile   编译测试代码

mvn -Dtest package  只打包不测试

mvn  source(扫斯).jar   源码打包

# 持久层框架

### 一.mvc思想

模型（Model）：持久层，负责实现应用程序的业务逻辑，封装有各种对数据的处理方法。它不关心它会如何被视图层显示或被控制器调用，它只接受数据并处理，然后返回一个结果

控制层（Controller）：负责控制应用程序的流程，它接收从视图层传过来的数据，然后选择业务层中的某个业务来处理，接收业务层返回的结果并选择视图层中的某个视图来显示结果

视图层（View）：负责应用程序对用户的显示，它从用户那里获取输入数据并通过控制层传给业务层处理，然后再通过控制层获取业务层返回的结果并显示给用户。

优点：

低耦合 重用性高

生命周期成本低：MVC模式使开发和维护用户接口的技术含量降低。

开发效率快：它使后台程序员集中精力于业务逻辑，界面程序员集中精力于表现形式上 减少开发时间

可维护性高：分离视图层和业务逻辑层使得WEB应用更易于维护和修改

### 二，orm

1、hibernate使用orm思想对数据库进行crud操作

2、在web阶段学习javabean更正确的叫法是：实体类

3、orm： object relational mapping 对象关系映射

\* O：面向对象领域的Object（JavaBean对象）

\* R：关系数据库领域的Relational（表的结构）

\* M：映射Mapping（XML的配置文件）

### 三，mybatis+plus

半ORM思想使用sql语句进行crud操作 数据库的 持久层框架

它支持自定义 SQL、存储过程以及高级映射。

MyBatis 免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作。

MyBatis 可以通过简单的 XML 或注解来配置和映射原始类型、接口和 Java POJO（Plain Old Java Objects，普通老式 Java 对象）为数据库中的记录。

使用select delete insert update 标签做增删改查

（1）优点：

① 基于SQL语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL写在XML里，解除sql与程序代码的耦合，便于统一管理；提供XML标签，支持编写动态SQL语句，并可重用。

② 与JDBC相比，减少了50%以上的代码量，消除了JDBC大量冗余的代码，不需要手动开关连接；

③ 很好的与各种数据库兼容（因为MyBatis使用JDBC来连接数据库，所以只要JDBC支持的数据库MyBatis都支持）。

④ 能够与Spring很好的集成；

⑤ 提供映射标签，支持对象与数据库的ORM字段关系映射；提供对象关系映射标签，支持对象关系组件维护。

（2）缺点：

① SQL语句的编写工作量较大，尤其当字段多、关联表多时，对开发人员编写SQL语句的功底有一定要求。

② SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

（3）#{}和${}的区别是什么？

${}是字符串拼接，#{}是占位符赋值；

Mybatis在处理${}时，就是把${}直接替换成变量的值。而Mybatis在处理#{}时，会对sql语句进行预处理，将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；

使用#{}可以有效的防止SQL注入，提高系统安全性。

（4）,使用MyBatis的mapper接口调用时有哪些要求？

Mapper接口方法名和mapper.xml中定义的每个sql的id相同；

Mapper接口方法的输入参数类型和mapper.xml中定义的每个sql 的parameterType的类型相同；

Mapper接口方法的输出参数类型和mapper.xml中定义的每个sql的resultType的类型相同；

Mapper.xml文件中的namespace即是mapper接口的类路径。

（5）.Mybatis动态sql有什么用？执行原理？有哪些动态sql？

Mybatis动态sql可以在Xml映射文件内，以标签的形式编写动态sql，执行原理是根据表达式的值 完成逻辑判断 并动态拼接sql的功能。

Mybatis提供了9种动态sql标签：trim | where | set | foreach | if | choose | when | otherwise | bind。

1. 常用的方法

selectById() 根据id查询

selectList(); 查List<对象>

selectPage() 分页

selectOne()按条件查

insert()新增

updateById()根据id查询

deleteById()根据id删除

### 四，hibernate

hibernate是全自动跨数据库开发效率高于jdbc的持久成框架

使用的是hql语句最终会转化为sql语句.语法是 from 对象名

（1），hibernate运行原理

1.通过configuration加载了hibernate.cfg.xml文件。

2.加载后创建sessionFactory.(sessionFactory是线程安全的)

3.然后同sessionFactory获得session。（创建会话，相当于jdbc中的connection 是线程不安全的）

4.通过session操作数据库，最后通过transaction来进行事物的控制.

（2），十．hibernate的五大核心接口

1.Configuration(类):

加载配置文件hibernate.cfg.xml文件中的配置信息，从而得到：

1）.hibernate的底层信息：

数据库连接，jdbc驱动，方言，用户名，密码。

2）.hibernate的映射文件(\*.hbm.xml)

2.sessionFactory（接口）

通过configuration创建sessionFactory。

可以用来获得session。

sessionFactory是线程安全的，

里边加载的数据信息有数据库的配置信息和映射关系.

3.Session(接口)

线程不安全。

相当于jbdc的connection.

用来操作数据库.

4.Transaction（接口）

进行事物的控制实现事物的commit（提交)和rollback（回滚）

5.Query(接口)

（3），hibernate主键生成策略

1》increment(银科瑞门特) :先选出当前表的最大ID，在此基础上加1，hibernate会自行处理，适用于int short,long类型的主键

2》sequence:oracle数据库会自行处理，会增加一个序列配置

<generator class="sequence">

<param name="sequence">sequence\_id</param>

</generator>

3》native（通用多个数据库，数据库的本地生成策略）

4》uuid：生成一个32位的，不重复的字符串，可以达到夸数据库

（4）．hibernate中session中的常用方法

save():保存

saveOrUpdate()保存或者修改

delete()删除

update()修改

get()根据唯一标示获取对象 立即加载

load()根据唯一标示获得对象 ，延迟加载

# 控制层框架

### 一，springmvc

（1）springMVC的运行原理

1. 客户端请求提交到DispatcherServlet(diss趴扯Servlet)

2. 由DispatcherServlet控制器查询一个或多个HandlerMapping(憨的了卖屏)，找到处理请求的handler(憨的了)。

3. DispatcherServlet将请求转发给到handler。

4. 通过HandlerAdapter(憨的了额搭铺特)执行handler调用业务逻辑处理后，返回ModelAndView。

5. DispatcherServlet查询一个或多个ViewResolver()视图解析器，将model渲染到views。

6.将views结果显示到客户端

（2）接参传参

接参：request 基本数据类型 对象（注意：时间和文件）

传参：转发传值 json传值 artreebute

（3） 页面跳转

页面跳转的方式有 转发forward 重定向redirect

Redirect和forward的区别：

转发(forward)和重定向(redirect)的区别：

转发地址栏不发生变化，数据共享，一次请求，执行效率高；

重定向地址栏发生变化，数据不共享，多次请求，执行效率低。

1. 常用的注解有：@controller @RequestMapping @ResponseBody

### 二，springboot

微服务 进行接口开发的

Spring Boot是由 Pivotal 团队提供的全新的框架，spring boot 整合了所有的框架让spring应用程序尽快跑起来并且尽可能减少你的配置文件

（1）优点：

1）为所有 Spring 开发提供一个更快更广泛的入门体验

2）零配置。无冗余代码生成和XML 强制配置，遵循“约定大于配置”（我们可以直接用它里面默认的配置，不需要去修改）

3）集成了大量常用的第三方库的配置， Spring Boot 应用为这些第三方库提供了几乎可以零配置的开箱即用的能力

4）提供一系列大型项目常用的非功能性特征，如嵌入式服务器web、安全性、度量、运行状况检查、外部化配置等

5）Spring Boot 不是Spring 的替代者，Spring 框架是通过 IOC 机制来管理 Bean 的。Spring Boot 依赖 Spring 框架来管理对象的依赖。Spring Boot 并不是Spring 的精简版本，而是为使用 Spring 做好各种产品级准备

（2）常用注解：

@restcontroller @getmapping @PathVariable @MapperScan

@springbootapplication（他是由三个注解组成的分别是@springbootConfiguration读取配置文件，配置文件的路径是当前根目录@EnableAutoConfiguration开启自动配置，扫描当前的所有依赖的jar包，发现新的依赖出现将会根据依赖完成各种自动配置@ComponentScan属于spring 框架(@Component,@Service,@Controller,@Repository,@Entity)，扫描范围默认情况下是启动类所在的同名包及其子孙包）

### 三，Servlet

（1）Servlet工作过程

1、Web Client 向Servlet容器（Tomcat）发出Http请求

                   2、Servlet容器接收Web Client的请求

                    3、Servlet容器创建一个HttpRequest对象，将Web Client请求的信息封装到这个对象中

                    4、Servlet容器创建一个HttpResponse对象

                    5、Servlet容器调用HttpServlet对象的service方法，把HttpRequest对象与HttpResponse 对象作为参数传给 HttpServlet对象

                   6、HttpServlet调用HttpRequest对象的有关方法，获取Http请求信息

                   7、HttpServlet调用HttpResponse对象的有关方法，生成响应数据

                   8、Servlet容器把HttpServlet的响应结果传给Web Client

（2）Servlet生命周期

在Servlet框架中所有的Servlet类都必须实现Servlet这个接口。其中定义了三个方法：

                            1、init方法：负责初始化Servlet对象。

                            2、service方法：用于响应客户端的请求

                            3、destroy：销毁Servlet对象，释放占用的资源。

                       Servlet生命周期四个阶段：

                             ●  加载阶段:加载并实例化(创建Servlet实例)

                             ●  初始化阶段:调用init()方法

                             ●  响应客户请求阶段：调用service()方法,doGet、doPost

                             ●  终止阶段：调用destroy()方法

1. HTTP的请求方式包括DELETE、GET、OPTIONS、POST、PUT和TRACE,在HttpServlet类中分别提供了相应的服务方法,它们是doDelete()、doGet()、doOptions()、doPost()、doPut()和doTrace()。不过目前很多服务器只支持get和post请求，所以通常自定义的Servlet中只需要重写doPost和doGet方法即可。