|  |
| --- |
| 技术问题(至少50个) |
| 问题1：对Spring事务的理解  回答：事物就是一系列操作要么全部成功，要么全部执行失败  四大特性：原子性、一致性、隔离性、持久性  隔离级别：默认的、读未提交、读已提交、可重复读、串行化  问题2：什么是进程？什么是线程？  回答：进程 是执行中的程序，多个线程的集合，线程是独立运行的一条路径。一个操作系统有多个进程，一个进程中一定有一个主线程。使用多线程，提高程序效率，提高用户体验，使用多线程，多个线程之间互不影响，都在自己独立运行。  问题3：创建线程的方式  回答： 1）继承Thread类创建线程  2）实现Runnable接口创建线程  3）使用Callable和Future创建线程  4）使用线程池例如用Executor框架  问题4：sleep(),wait(),join()区别  回答：sleep():使线程进入睡眠状态，不会释放锁，睡眠时间过后会进入就绪状态  wait():释放锁，唤醒时使用notify()  Join()：当前线程中调用，指让出CPU调度，让其他线程先执行。  问题5：线程安全问题？  回答：当多个线程共享同一个全局变量（局部变量不会），做写的时候，可能会受到其他线程的影响，就叫做线程安全问题（读的时候不会导致线程安全问题）  线程如何实现同步：（保证数据的原子性，保证数据安全）  Synchronized自动锁，抢锁的资源  Synchronized底层通过Monitorenter（加1得到锁）和Monitorexit（减1释放锁）  分布式锁：redis中setnx命令  有一个线程已经拿到锁，其他线程等待  条件：必须有两个线程以上，  非静态同步函数：在方法上加Synchronized使用的是this锁  静态同步函数：（static修饰）不能使用this锁  加锁保证数据安全  多线程死锁同步中嵌套同步，无法释放，一直等待，形成死锁  Volatile保证可见性，但不能保证原子性，禁止指令重排（强制将数据从线程内存副本刷新的主内存）  Lock锁: 手动锁，手动上锁，手动释放锁，灵活性高，放在finally()中释放锁  停止线程：stop()：中断不安全，中断循环  ThreadLock:本地线程，原理，  问题6：在上家公司你做了哪些项目？   1. 回答：用户微服务：用户这个模块呢？主要是登录和注册功能，注册的话有一个短信验证的功能，我们用的是阿里大于进行的发短信，用户注册的时候点击获取验证码，后台生成一个6位随机数，一式两份Redis中一份，Mq 中一份，发短信模块呢，进行监听，监听到之后会调用阿里大于发短信的接口，用户接受短信填写验证码，跟Redis中进行对比，验证通过就是在数据裤用户表中加一条数据，登录这块，首先就是先判断这个用户是不是存在，如果是数据库有着一条数据，那么就是通过JWT生成一个token令牌，将这个token返回给前台 ，如果用户访问哪个微服务的时候，而当前操作是需要用户的权限的，那么这个微服务就是需要配置拦截器，但是这个拦截器不是对前进行拦截，而是对请求中的token进行解析，判断角色字段，将用户和管理员分别放在作用域中，如果访问当前的方法需要用户或者是管理员权限，就通过获取作用域中的token，判断这个token是否存在，不存在就不让执行抛出异常，存在就放行，程序继续执行 2. 问答微服务：问答模块呢？主要是实现让用户实现在平台可以发布在面试中遇到的问题，请求帮助，对于好的回答可以进行点赞，这里用到了Redis防止重复点赞，用户也可以对已经有的问题进行答复，因为问答或者评论数据量是很大的并且是属于一层层的数据，所以我们没有用MySQL，而是用到了MongoDB这个文档型数据，后台我们在创建回答的时候会先判断他的上级Id是否存在，如果是存在的话上级的用户的回复数加1，MongoDB提供了一个inc方法这个方法是实现了当前字段的自增。 3. 搜索微服务：搜索这块呢我们用的是es，首先是将数据库中数据同步到es中，用的是logstash，   （4）交友微服务：用户点击交友这一按钮，会列出一些其他用户，对于自己喜欢的人，可以对其进行关注，对于不喜欢的人呢可以对其进行拉黑，后台操作的话就是判断是否喜欢，喜欢向好友喜欢表中添加后双方的Id，还要判断是否已经喜欢和是否互相喜欢，若互相喜欢该两条记录的状态为1表示相互喜欢，如果是不喜欢，还需要判断是否已经是好友，若已经是好友，还要判断是否是相互喜欢，是则删除单向喜欢数据，修改相反喜欢的状态为0，再向非好友中添加数据。如果不是好友，直接向非好友表中添加，对应好友粉丝数和自己的关注数要进行修改。  问题7：新生代和老年代？  回答：新生代：刚创建的对象放在新生代的Eden中，新生代分为Eden、s0、s1三个区域。S0和s1目的是复制算法 。  老年代：如果对象频繁的被使用，对象会放在老年代，  垃圾回收机制主要回收新生代，  回答8：这个项目的数据量和并发量多大？  回答：并发量是300 支持最大1000  回答9：这个项目用的什么技术？  回答：SpringBoot+SpringCloud微服务框架，SpringBoot是纯注解开发是项目，写起来也是相对听简单的，SpringCloud相当于一个架构，里面有很多的组件，都是SpringCloud集成的，SpringDataJpa持久层框架，用来和数据库交互的  问题10： 什么是守护线程？什么是非守护线程  回答：守护线程和主线程相关，和主线程一起销毁（gc线程）  用户线程（非守护线程）是用户自己创建的线程，主线程销毁和用户线程无关  问题11：Vector和ArrayList？  回答：Vector线程安全，Vector中add()方法加上Synchronized锁所以安全  问题12：并发包  回答：concurrentHashMap：线程安全，内部采用分段锁，讲一个整体分成16段，分成16个HashTable，所以线程安全，效率高  CountDownLatch：计数器，没执行一次减1，到0时执行下面代码，不为0，会阻塞  问题13：线程池？  回答： 经常创建、启动、销毁线程是很耗性能，  通过线程池管理复用线程，提高程序效率，  好处：降低资源消耗（重复利用），提高响应速度  问题14：创建线程池的方式  回答：Java通过Executors（jdk1.5并发包）提供四种线程池，分别为：  newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。  newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。  newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。  newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。  问题15：CUP密集：频繁的调度，线程数和CPU数相同  回答：IO密集：2\*CUP数  问题16：java锁机制  回答：  悲观锁：每次在拿数据的时候，都会上锁，缺点：效率低，因为只能保证一个连接进行操作。  乐观锁：版本标识，乐观锁:乐观锁会乐观的认为每次查询都不会造成更新丢失,利用版本字段控制  分段锁：  重入锁：重入锁（也叫做递归锁，指的是同一线程 外层函数获得锁之后 ，内层递归函数仍然有获取该锁的代码，但不受影响。  读写锁：两个线程同时读一个资源没有任何问题，所以应该允许多个线程能在同时读取共享资源。但是如果有一个线程想去写这些共享资源，就不应该再有其它线程对该资源进行读或写（有读可以写，有写不能再读）  CAS无锁：它包含三个参数CAS(V,E,N): V表示要更新的变量，E表示预期值，N表示新值。仅当V值等于E值时，才会将V的值设为N，如果V值和E值不同，则说明已经有其他线程做了更新，则当前线程什么都不做。最后，CAS返回当前V的真实值。  自旋锁：自旋锁是采用让当前线程不停地的在循环体内执行实现的，当循环的条件被其他线程改变时 才能进入临界区。  排他锁：  问题17：ThreadLocal 造成内存溢出  回答：每个Thread的对象都有一个ThreadLocalMap，当创建一个ThreadLocal的时候，就会将该ThreadLocal对象添加到该Map中，其中键就是ThreadLocal，值可以是任意类型。  在该类中，我觉得最重要的方法就是两个：set()和get()方法。当调用ThreadLocal的get()方法的时候，会先找到当前线程的ThreadLocalMap，然后再找到对应的值。set()方法也是一样。  用法：  1、ThreadLocal.get: 获取ThreadLocal中当前线程共享变量的值。  2、ThreadLocal.set: 设置ThreadLocal中当前线程共享变量的值。  3、ThreadLocal.remove: 移除ThreadLocal中当前线程共享变量的值。  4、ThreadLocal.initialValue: ThreadLocal没有被当前线程赋值时或当前线程刚调用remove方法后调用get方法，返回此方法值  问题18：线程副本  回答：  问题19：同时插入100万条数据到数据库  回答：分批  问题20：1t文件数字（字母）乱序，进行排序  回答：NIO(非阻塞时IO)将文件按行切割，读文件，  问题21：什么是方法区？  回答：方法区（永久），、存放static关键字修饰、常量信息，当Class文件被加载的时候，就会被初始化。  问题22：java内存模型（多线程JMM）------java内存结构（JVM虚拟机储存空间）  回答：class文件🡪类加载器  堆：创建对象，new出来的创建，数组，堆中内存是被所有线程共享  栈：局部变量，代码运行完毕，自动释放内存，每个线程私有，互不干涉，不会产生线程安全问题。  本地方法栈：主要是调用C语言，JNI，执行引擎：执行字节码文件  类的方法存在栈  问题23：jvm调优？  回答：1、定义常量不要太多，因为常量存在方法区，GC是不会回收的，非常占内存，所有线程共享常量。  2、尽量减少垃圾回收的次数  3、新生代回收次数比老年代回收次数多  问题24：JVM参数调优总结  回答：在JVM启动参数中，可以设置跟内存、垃圾回收相关的一些参数设置，默认情况不做任何设置JVM会工作的很好，但对一些配置很好的Server和具体的应用必须仔细调优才能获得最佳性能。通过设置我们希望达到一些目标：  • GC的时间足够的小  • GC的次数足够的少  • 发生Full GC(新生代和老年代)的周期足够的长  前两个目前是相悖的，要想GC时间小必须要一个更小的堆，要保证GC次数足够少，必须保证一个更大的堆，我们只能取其平衡。  （1）针对JVM堆的设置，一般可以通过-Xms -Xmx限定其最小、最大值，为了防止垃圾收集器在最小、最大之间收缩堆而产生额外的时间，我们通常把最大、最小设置为相同的值  （2）年轻代和年老代将根据默认的比例（1：2）分配堆内存，可以通过调整二者之间的比率NewRadio来调整二者之间的大小，也可以针对回收代，比如年轻代，通过 -XX:newSize -XX:MaxNewSize来设置其绝对大小。同样，为了防止年轻代的堆收缩，我们通常会把-XX:newSize -XX:MaxNewSize设置为同样大小  （3）年轻代和年老代设置多大才算合理？这个我问题毫无疑问是没有答案的，否则也就不会有调优。我们观察一下二者大小变化有哪些影响  • 更大的年轻代必然导致更小的年老代，大的年轻代会延长普通GC的周期，但会增加每次GC的时间；小的年老代会导致更频繁的Full GC  • 更小的年轻代必然导致更大年老代，小的年轻代会导致普通GC很频繁，但每次的GC时间会更短；大的年老代会减少Full GC的频率  • 如何选择应该依赖应用程序对象生命周期的分布情况：如果应用存在大量的临时对象，应该选择更大的年轻代；如果存在相对较多的持久对象，年老代应该适当增大。但很多应用都没有这样明显的特性，在抉择时应该根据以下两点：（A）本着Full GC尽量少的原则，让年老代尽量缓存常用对象，JVM的默认比例1：2也是这个道理 （B）通过观察应用一段时间，看其他在峰值时年老代会占多少内存，在不影响Full GC的前提下，根据实际情况加大年轻代，比如可以把比例控制在1：1。但应该给年老代至少预留1/3的增长空间  问题25：什么是虚拟机参数配置  回答：在虚拟机运行的过程中，如果可以跟踪系统的运行状态，那么对于问题的故障排查会有一定的帮助，为此，在虚拟机提供了一些跟踪系统状态的参数，使用给定的参数执行Java虚拟机，就可以在系统运行时打印相关日志，用于分析实际问题。我们进行虚拟机参数配置，其实就是围绕着堆、栈、方法区、进行配置  问题26: SpringCloud服务注册发现原理  回答：每个SpringCloud服务器启动后向注册中心注册本服务器信息，如服务别名、服务器IP、端口号等，其他服务进行请求时先根据服务别名从注册中心获取到目标服务器IP和端口号，并将获取到的信息缓存到本地，然后通过本地使用HttpClient等技术进行远程调用。  问题27：Eureka如何实现高可用  回答：启动多台Eureka服务器，然后作为SpringCloud服务互相注册，客户端从Eureka集群获取信息时，按照注册的Eureka顺序对第一个Eureka进行访问。  问题28：Nginx与Ribbon的区别  回答：Nginx是反向代理同时可以实现负载均衡，nginx拦截客户端请求采用负载均衡策略根据upstream配置进行转发，相当于请求通过nginx服务器进行转发。Ribbon是客户端负载均衡，从注册中心读取目标服务器信息，然后客户端采用轮询策略对服务直接访问，全程在客户端操作。  问题29：Ribbon底层实现原理  回答：Ribbon使用discoveryClient从注册中心读取目标服务信息，对同一接口请求进行计数，使用%取余算法获取目标服务集群索引，返回获取到的目标服务信息。  问题30：谈谈服务雪崩效应  回答：雪崩效应是在大型互联网项目中，当某个服务发生宕机时，调用这个服务的其他服务也会发生宕机，大型项目的微服务之间的调用是互通的，这样就会将服务的不可用逐步扩大到各个其他服务中，从而使整个项目的服务宕机崩溃.发生雪崩效应的原因有以下几点  1单个服务的代码存在bug. 2请求访问量激增导致服务发生崩溃(如大型商城的枪红包，秒杀功能). 3.服务器的硬件故障也会导致部分服务不可用  问题31：在微服务中，如何保护服务  回答：一般使用使用Hystrix框架，实现服务隔离来避免出现服务的雪崩效应，从而达到保护服务的效果。当微服务中，高并发的数据库访问量导致服务线程阻塞，使单个服务宕机，服务的不可用会蔓延到其他服务，引起整体服务灾难性后果，使用服务降级能有效为不同的服务分配资源,一旦服务不可用则返回友好提示，不占用其他服务资源，从而避免单个服务崩溃引发整体服务的不可用  问题32：服务雪崩效应产生的原因  回答：因为Tomcat默认情况下只有一个线程池来维护客户端发送的所有的请求，这时候某一接口在某一时刻被大量访问就会占据tomcat线程池中的所有线程，其他请求处于等待状态，无法连接到服务接口。  问题33：谈谈服务降级、熔断、服务隔离  回答：服务降级：当客户端请求服务器端的时候，防止客户端一直等待，不会处理业务逻辑代码，直接返回一个友好的提示给客户端。  服务熔断是在服务降级的基础上更直接的一种保护方式，当在一个统计时间范围内的请求失败数量达到设定值（requestVolumeThreshold）或当前的请求错误率达到设定的错误率阈值（errorThresholdPercentage）时开启断路，之后的请求直接走fallback方法，在设定时间（sleepWindowInMilliseconds）后尝试恢复。  服务隔离就是Hystrix为隔离的服务开启一个独立的线程池，这样在高并发的情况下不会影响其他服务。服务隔离有线程池和信号量两种实现方式，一般使用线程池方式。  问题34：Zuul与Nginx有什么区别？  回答：Zuul是java语言实现的，主要为java服务提供网关服务，尤其在微服务架构中可以更加灵活的对网关进行操作。Nginx是使用C语言实现，性能高于Zuul，但是实现自定义操作需要熟悉lua语言，对程序员要求较高，可以使用Nginx做Zuul集群。  问题35：ZuulFilter常用有那些方法  回答：Run()：过滤器的具体业务逻辑  shouldFilter()：判断过滤器是否有效  filterOrder()：过滤器执行顺序  filterType()：过滤器拦截位置  问题36：如何实现动态Zuul网关路由转发  回答：通过path配置拦截请求，通过ServiceId到配置中心获取转发的服务列表，Zuul内部使用Ribbon实现本地负载均衡和转发。  问题37：Zuul网关如何搭建集群  回答：使用Nginx的upstream设置Zuul服务集群，通过location拦截请求并转发到upstream，默认使用轮询机制对Zuul集群发送请求。  问题38：什么是SpringBoot  回答：SpringBoot是快速开发的Spring框架，能够快速整合主流框架，简化xml配置，采用全注解化，内置Http服务器（如tomcat、jetty等），通过java部署运行  问题39：SpringBoot与SpringCloud 区别  回答：SpringBoot是快速开发的Spring框架，SpringCloud是完整的微服务框架， SpringCloud依赖于SpringBoot。  问题40：SpringBoot中用那些注解  回答：@EnableAutoConfiguration作用  自动扫描并添加jar包依赖  @SpringBootApplication原理  是一个组合注解，相当于@EnableAutoConfiguration和@ComponentScan  问题41：热部署原理是什么？  回答：热部署的实现原理主要依赖java的类加载机制，在实现方式可以概括为在容器启动的时候起一条后台线程，定时的检测类文件的时间戳变化，如果类的时间戳变掉了，则重新加载整个应用的class文件，同时重启服务，重新部署。  问题42：SpringBoot多数据源事务如何管理  回答：第一种方式是在service层的@TransactionManager中使用transactionManager指定DataSourceConfig中配置的事务  第二种是使用jta-atomikos实现分布式事务管理  问题43：SpringBoot性能如何优化  回答：如果项目比较大，类比较多，不使用@SpringBootApplication，采用@Compoment指定扫包范围  在项目启动时设置JVM初始内存和最大内存相同  将springboot内置服务器由tomcat设置为undertow  问题44：SpringBoot执行流程  回答：使用SpringApplication.run()启动，在该方法所在类添加@SpringBootApplication注解，该注解由@EnableAutoConfiguration和@ComponentScan等注解组成，@EnableAutoConfiguration自动加载SpringBoot配置和依赖包，默认使用@ComponentScan扫描当前包及子包中的所有类，将有spring注解的类交给spring容器管理  问题45：SpringBoot底层实现原理  回答：使用maven父子包依赖关系加载相关jar包，使用java操作Spring的初始化过程生成class文件，然后用java创建tomcat服务器加载这些class文件  问题46：分布式Session一致性问题  回答：a、使用Nginx反向代理，即IP绑定，同一个ip只能在同一个机器上访问  b、使用数据库，但性能不高  c、tomcat内置了对session同步的支持，但可能会产生延迟  d、使用Spring-Session框架，相当于把session放到redis中  e、使用token令牌代替session  问题47：谈谈网站跨域解决方案  回答：a、使用jsonp 缺点只能发送get请求  b、使用httpclient进行转发，效率低  c、设置响应头允许跨域  d、使用Nginx搭建api网关  e、使用Zuul微服务搭建api接口网关  问题48：使用Zookeeper实现服务Master选举原理  回答：多个服务器在启动时候，会在Zookeeper上创建相同的临时节点，谁如果能够创建成功，谁就为主。如果主服务器宕机，其他备用节点获取监听信息，重新创建节点，选出主服务器。  问题49：ZooKeeper实现分布式锁与Redis实现分布式锁区别  回答：Zookeeper通过创建临时节点和利用监听事件实现分布式锁，Redis使用setnx命令创建相同的key，因为Redis的key保证唯一，先创建的先获取锁。  不断的去尝试，去获取锁，比较耗性能  Zookeeper实现分布式锁，即使获取不到锁，创建对锁的监听即可，不需要不断去尝试获取 锁，性能开销小  Redis实现分布式锁，如果客户端获取到锁的时候遇到bug或挂了，还需要等到超时时间过了以后才能重新获取锁  Zookeeper实现分布式锁，创建的是临时节点，客户端挂了，节点自然删除，也就达到了自动释放锁的效果  问题50：ZooKeeper应用场景  回答：A、 服务注册与发现的中心  B、利用临时节点特性解决分布式锁  C、分布式配置中心  D、基于哨兵机制实现选举策略  E、实现本地负载均衡  F、基于节点事件通知特性可做消息中间件  G、分布式事务 |
| **人资问题(至少30个)** |
| 问题1：自我介绍  回答：面试官您好，我叫王顺虎，来自河北邯郸，从事Java开发已经有三年了。上家公司是南京的一家公司，因为我就是在南京上的大学嘛，所以当时就直接在那边实习了，中间有换过一次公司，然后在上家公司做的项目是一个关于交友和招聘等多功能的平台，主要用SpringBoot+SpringCloud微服务框架进行开发，最后这个项目我们做了将近一年，我离职前是已经在测试阶段了，我呢，平时有时间会自己浏览网上的一些博客之类的，了解一些新的知识，这就是我的一个简单的介绍，您还有什么要了解的吗?  问题2：你为什么要离开上家公司？你上家公司在xxx，我们公司在xxx，离这么远为什么要选择我们这里？  回答：离职的原因呢？是因为公司的这个项目已经做完了，也就剩下在进行测试bug，等待上线了，在公司一天天的就是好多人都是在闲着所以我就选择了离职，至于为什么来北京这边呢？这要是因为我本身就是北方人嘛，北京这边也是有好多亲戚朋友，当然父母也是在这边，其实他们早就想让我回这边工作了，我也是因为那边的项目一直没有结束，所以就一直拖着，现在项目也完成差不多了，当然北京也是属于一线城市，对于我个人的发展也是有所帮助的，所以就回北京这边工作了。  问题3：你觉得自己的优点是？你觉得自己有啥缺点？  回答：首先呢我是很看重别人对我的看法的一个人，所以我会很努力让自己做的更好，比如在公司中我会认真高效的完成上级交给我的任务。会服从领导和公司是安排，争取在领导心中有个好印象。  缺点的话，就是说可能有的时候工作中遇到很难解决是问题，我可能就是会四口到底吧，到最后整的自己还挺累，最后还是得请教同事和领导。  问题4：你有哪些 offer？  回答：到目前为止呢，我才面试过两家，咱们是第二个，所以说offer暂时还没有。  问题5：上家公司的同事和领导是怎么评价你的？  回答：挺好  问题6：你在项目中是什么角色？  回答：技术人员，负责敲代码  问题7：这个项目的数据量和并发量多大？  回答：并发量是300 支持最大1000  问题8：这个项目用的什么技术？  回答：SpringBoot+SpringCloud微服务框架，SpringBoot是纯注解开发是项目，写起来也是相对听简单的，SpringCloud相当于一个架构，里面有很多的组件，都是SpringCloud集成的，SpringDataJpa持久层框架，用来和数据库交互的  问题9：项目过程中有遇到什么很印象深刻的问题吗？  回答： 分布式事务的解决  问题10：是怎么解决的这些问题？  回答：1通过本地事务  2.两阶段提交  2消息中间件保证最终一致性  问题11：你对加班的看法？  回答：如果是工作需要我会义不容辞加班，我现在单身，没有任何家庭负担，可以全身心的投入工作。但同时，我也会提高工作效率，减少不必要的加班。  问题12：谈谈你对跳槽的看法？  回答：正常的“跳槽”能促进人才合理流动，应该支持。频繁的跳槽对单位和个人双方都不利，应该反对。  问题13：工作中你难以和同事、上司相处，你该怎么办  回答：我会服从领导的指挥，配合同事的工作。我会从自身找原因，仔细分析是不是自己工作做得不好让领导不满意，同事看不惯。还要看看是不是为人处世方面做得不好，如果是这样的话 我会努力改正。  问题14：为什么要离职?  回答：我希望能获得一份更好的工作，如果机会来临，我会抓住。我觉得目前的工作，已经达到顶峰，即没有升迁机会。  问题15：你为什么愿意到我们公司来工作？  回答：我认为贵公司能够给我提供一个与众不同的发展道路  问题16：你希望与什么样的上级共事？  回答：希望我的上级能够在工作中对我多指导，对我工作中的错误能够立即指出  问题17：与上级意见不一是，你将怎么办？  回答：我会给上级以必要的解释和提醒，在这种情况下，我会服从上级的意见  问题18：为什么选择我们这家公司？  回答：曾经在网上看过关于贵公司的报道，与自己所追求的理念有志一同。而贵公司在业界的成绩也是有目共睹的，而且对员工的教育训练、升迁等也都很有制度。  问题19：为什么选择这个职务？  回答：这一直是我的兴趣和专长，经过这几年的磨练，也累积了一定的经验及人脉，相信我一定能胜任这个职务的。  问题20：怎样看待学历和能力？  回答：学历我想只要是大学专科的学历，就表明觉得我具备了根本的学习能力。剩下的，你是学士也好，还是博士也好，对于这一点的讨论，不是看你学了多少知识，而是看你在这个领域上发挥了什么，也就是所说的能力问题。一个人工作能力的高低直接决定其职场命运，而学历的高低只是进入一个企业的敲门砖，如果贵公司把学历卡在博士上，我就无法进入贵公司，当然这不一定只是我个人的损失，如果一个专科生都能完成的工作，您又何必非要招聘一位博士生呢？  问题21：何时可以到职？  回答：如果被录用的话，到职日可按公司规定上班  问题22：谈谈如何适应办公室工作的新环境？  回答：办公室里每个人有各自的岗位与职责，不得擅离岗位，根据领导指示和工作安排，制定工作计划，提前预备，并按计划完成。多请示并及时汇报，遇到不明白的要虚心请教。抓间隙时间，多学习，努力提高自己的政治素质和业务水平。  问题23：为什么我们要在众多的面试者中选择你？  回答：根据我对贵公司的了解，以及我在这份工作上所累积的专业、经验及人脉，相信正是贵公司所找寻的人才。而我在工作态度、ＥＱ上，也有圆融、成熟的一面，和主管、同事都能合作愉快。  问题24：如何安排自己的时间？会不会排斥加班？  回答：基本上，如果上班工作有效率，工作量合理的话，应该不太需要加班。可是我也知道有时候很难避免加班，加上现在工作都采用责任制，所以我会调配自己的时间，全力配合。  问题25：在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么做？  回答：原则上我会尊重和服从领导的工作安排；同时私底下找机会以请教的口吻，婉转地表达自己的想法，看看领导是否能改变想法。如果领导没有采纳我的建议，我也同样会按领导的要求认真地去完成这项工作。  问题26：眼下你生活中最重要的是什么？  回答：对我来说，能在这个领域找到工作是最重要的，能在贵公司任职对我说最重要。  问题27：什么会让你有成就感？  回答：为贵公司竭力效劳，尽我所能，完成一个项目。  问题28：怎样对待自己的失败？  回答：我们大家生来都不是十全十美的，我相信我有第二个机会改正我的错误。  问题29：你通常如何处理别人的批评？  回答：沈默是金，不必说什么，否则情况更糟，不过我会接受建设性的批评。②我会等大家冷静下来再讨论。  问题30：你欣赏哪种性格的人？  回答：诚实、不死板而且容易相处的人、有“实际行动”的人。 |