|  |
| --- |
| 技术问题(至少50个) |
| **问题1：SpringMVC的运行原理**  回答：整个处理过程从一个HTTP请求开始：  Tomcat在启动时加载解析web.xml。找到Spring mvc的前端总控制器DispatcherServlet，并且通过DispatcherServlet来加载相关的配置文件信息。  DispatcherServlet接收到客户端请求，找到对应HandlerMapping，根据映射规则，找到对应的处理器（Handler）  调用相应处理器中的处理方法，处理该请求后，会返回一个ModelAndView  DispatcherServlet根据得到的ModelAndView中的视图对象，找到一个合适的ViewResolver（视图解析器），根据视图解析器的配置，DispatcherServlet将要显示的数据传给对应的视图，最后显示给用户。  **问题2：你对分布式事务了解吗 你对如何解决分布式事务得？**  因为你的简历 的项目 都是分布式架构或者是微服务，肯定问分布式事务，即使你没有处理过分布式事务，也要知道有哪些解决方案及优缺点  1、两阶段提交方案/XA方案  举个栗子 团建(tb) team building,tb主席（负责人）  第一个阶段：tb 活动 主要是询问一下大家得想法，都OK  第二个阶段：大家可以一起去完成了  tb 事务管理器，主要负责协调多个数据库得事务，去执行各个得本地事务，然后就可以正式提交整个事务了。如果有一个数据库不同意得话，就需要回滚  这个方案 他适合域单体应用，会连接多个数据源，会跨库操作，会严重依赖于数据库层面操作事务，操作效率低，不合适高并发得场景，如果去实现得话，可以使用spring+JTA，找个demo  现在得架构微服务架构很多，主张得是一个应用连接一个数据库，不能跨库操作  2、TCC方案  T:TRY :对各个服务得资源做确认，及对资源的锁定或者保留  C:Confirm：各个服务实际得操作  C:Cancel：只要有其中一个服务得业务方法出现出错，需要补偿，即将已经执行成功得操作回滚  强一致性  举个例子：跨行转账：  T:将俩个银行账户中得资金冻结。  C:实际进行转账，A 减钱 B加钱  C;如果有任何一个银行得操作失败，需要回滚进行补偿，比如 A已经扣减完成了，B在增加资金得时候失败了，此时就的吧A得钱加回去  缺点：回滚操作时严重依赖于开发人员得代码进行回滚或者补偿得，代价比较大，使用得场景时有限得。一般时要求数据一致性特别高得时候，才采用这种方案，核心中核心业务  3:本地消息表：  1：A系统在自己得本地事务操作得同时，会插入一条数据到消息表  2:接着A系统会把找个数据发送到MQ,A系统就是消息得生产者  3:B系统接收到消息（B系统是个消息得消费者），在自己得本地事务中，往自己得本地消息表中插入一个数据，同事执行自己得业务如果找个消息被处理过了，那么找个事务就需要回滚  4：B系统执行成功侯，会更新自己得本地消息表及A系统得本地消息表得状态  5：如果B系统处理失败了，首先不会更新自己得本地消息表，A系统会自己定时扫描自己得消息表，如果发现数据没有改变成功得状态，则会往MQ中再次发送该消息，让B系统再次处理  6：这个方案 保证得时消息得最终一致性，哪怕B失败无数次，A永远不放弃，一直发送消息，直到B成功  这个方案时严重依赖雨数据库得消息表来管理事务得，不合适雨高并发得场景，实际使用少  4可靠消息得最终一致性 https://www.jianshu.com/p/fc47ebaff6a4?utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation  没有 本地消息表，直接使用MQ来完成这个事务得操作，activeMQ,RocketMQ，RabbitMQ  1,A先发送prepared消息到MQ,如果消息发送失败，就取消操作别执行  2，如果发送成功了，MQ会给A发送回调消息，那么A就处理自己得本地事务  ，如果失败了，告诉MQ进行回滚  3，A系统给MQ发送确认消息，此时B消息就会接收到这个消息，然后执行本地事务  4.mq会自动定时轮询所有prepared消息回调你的接口，问你，这个消息是不是本地事务处理失败了，所有没发送确认消息？那是继续重试还是回滚？一般来说这里你就可以查下数据库看之前本地事务是否执行，如果回滚了，那么这里也回滚吧。这个就是避免可能本地事务执行成功了，别确认消息发送失败了。  5.这个方案里，要是系统B的事务失败了咋办？重试咯，自动不断重试直到成功，如果实在是不行，要么就是针对重要的资金类业务进行回滚，比如B系统本地回滚后，想办法通知系统A也回滚；或者是发送报警由人工来手工回滚和补偿  这个还是比较合适的，目前国内互联网公司大都是这么做的  ---------------------------------------------------------------  我们现在想保证我们的某个系统非常的可靠，任何一个数据都不能错，我们用的是微服务架构，几十个服务。结果我们一盘点，发现，如果到处都要搞的话，一个系统要做几十个分布式事务出来。  我们的经验，带几十人的team，最大的一个项目，起码几百个服务，复杂的分布式大型系统，里面其实也没几个分布式事务。  你其实用任何一个分布式事务的这么一个方案，都会导致你那块儿代码会复杂10倍。很多情况下，系统A调用系统B、系统C、系统D，我们可能根本就不做分布式事务。如果调用报错会打印异常日志。  每个月也就那么几个bug，很多bug是功能性的，体验性的，真的是涉及到数据层面的一些bug，一个月就几个，两三个？如果你为了确保系统自动保证数据100%不能错，上了几十个分布式事务，代码太复杂；性能太差，系统吞吐量、性能大幅度下跌。  99%的分布式接口调用，不要做分布式事务，直接就是监控（发邮件、发短信）、记录日志（一旦出错，完整的日志）、事后快速的定位、排查和出解决方案、修复数据。  每个月，每隔几个月，都会对少量的因为代码bug，导致出错的数据，进行人工的修复数据，自己临时动手写个程序，可能要补一些数据，可能要删除一些数据，可能要修改一些字段的值。  比你做50个分布式事务，成本要来的低上百倍，低几十倍  trade off，权衡，要用分布式事务的时候，一定是有成本，代码会很复杂，开发很长时间，性能和吞吐量下跌，系统更加复杂更加脆弱反而更加容易出bug；好处，如果做好了，TCC、可靠消息最终一致性方案，一定可以100%保证你那快数据不会出错。  1%，0.1%，0.01%的业务，资金、交易、订单，我们会用分布式事务方案来保证，会员积分、优惠券、商品信息，其实不要这么搞了  **问题3：关于redis**  回答：Redis是一款开源的、基于内存、单线程的高速缓存数据库。使用C语言编写，Redis是一个key-value存储系统，它支持丰富的数据类型，如：string、list、set、zset、hash。我们在项目中经常使用redis作为缓存数据库使用，因为他基于内存所以处理速度非常快，又提供了完善的持久化机制。  数据库个数：16  单线程  默认端口号：6379  Redis的持久化：RDB  概念：在指定的时间间隔内将内存中的数据集快照写入磁盘，它恢复时是将快照文件直接读到内存里  Redis的持久化：AOF  概念：以日志的形式来记录每个写操作，将Redis执行过的所有写指令记录下来（读操作不记录），只许追加文件，但不可以改写文件，redis启动之初会读取该文件重新构建数据，换言之，redis重启的话就根据日志文件的内容将写指令从前到后执行一次以完成数据的回复工作。  **问题4：关于线程池，ThreadPoolExecutor类是线程池中最核心的一个类**  corePoolSize：核心池的大小，这个参数跟后面讲述的线程池的实现原理有非常大的关系。在创建了线程池后，默认情况下，线程池中并没有任何线程，而是等待有任务到来才创建线程去执行任务，除非调用了prestartAllCoreThreads()或者prestartCoreThread()方法，从这2个方法的名字就可以看出，是预创建线程的意思，即在没有任务到来之前就创建corePoolSize个线程或者一个线程。默认情况下，在创建了线程池后，线程池中的线程数为0，当有任务来之后，就会创建一个线程去执行任务，当线程池中的线程数目达到corePoolSize后，就会把到达的任务放到缓存队列当中；  maximumPoolSize：线程池最大线程数，这个参数也是一个非常重要的参数，它表示在线程池中最多能创建多少个线程；  keepAliveTime：表示线程没有任务执行时最多保持多久时间会终止。默认情况下，只有当线程池中的线程数大于corePoolSize时，keepAliveTime才会起作用，直到线程池中的线程数不大于corePoolSize，即当线程池中的线程数大于corePoolSize时，如果一个线程空闲的时间达到keepAliveTime，则会终止，直到线程池中的线程数不超过corePoolSize。但是如果调用了allowCoreThreadTimeOut(boolean)方法，在线程池中的线程数不大于corePoolSize时，keepAliveTime参数也会起作用，直到线程池中的线程数为0；  unit：参数keepAliveTime的时间单位，有7种取值，在TimeUnit类中有7种静态属性  workQueue：一个阻塞队列，用来存储等待执行的任务  threadFactory：线程工厂，主要用来创建线程；  handler：表示当拒绝处理任务时的策略  **问题5：介绍一下SpringCloud？**  五个常用组件：  服务发现——Netflix Eureka  客服端负载均衡——Netflix Ribbon  断路器——Netflix Hystrix  服务网关——Netflix Zuul  分布式配置——Spring Cloud Config  问题6：关于金融项目招标模块  信用贷借款,跳转页面前,获取当前用户的id,判断用户是否具有借款资格,false:跳转提示页面;true:model放置最小借款金额、model放置最小投标金额，从session取出登录用户id数据库查询账目表,并判断账户金额是否合法，跳转申请界面 borrow\_apply.html页面填写借款金额,利息,借款期限,还款方式,---最小投标,招标期限,借款标题,借款描述。借款申请提交，查询当前用户账户account，判断用户是否具有借款资格，false：跳转提示页面；true：创建申请借款对象，保存申请，数据库操作，改变当前用户的状态码 数据库操作（修改时候,支持乐观锁）。重定向，跳转页面前查询  问题7：关于金融项目招标审核  审核信用贷借款：跳转页面前查询list，list放在model 跳转审核页面，页面提交审核结果。id查询当前申请表，判断此申请是否处于待发布状态 ，true：创建一个审核历史对象，set 属性值，添加一个审核历史记录表，判断前台传来state值；true：审核通过状态码由待发布-->招标，由当前日期和招标时间,set截止日期，set发标日期，添加备注；false：审核通过状态码由待发布-->拒绝，查询用户信息,包含用户此时状态码，移除；有一个借款流程在申请的状态码，修改当前用户的状态码，修改借款申请表属性，返回JSONResult对象。  问题8：关于金融项目投标模块  投资:分页list 查询有哪些标可以投,携带id跳转后台,查询id用户的申请表信息和用户userinfo表,判断是否有此申请标的记录,true:使用借款对象实体的id重新查询借款人的信息,通过借款对象的实名认证id查询实名认证表信息,通过申请人查询风控资料表,将借款人的实名认证信息,风控资料信息,以及借款申请信息 放在model返回前台,判断当前登录用户是否不为空;true:判断此时点击看出借款信息的是否是本人,false:将验证结果放在model,查询当前登录对象账户表,并验证,是否资金异常,账户未有异常,则将当前用户账户查询后,放入model;true:将验证结果放在model;false:前台不予显示借款人信息；false：根据是否登录,是否本人,是否账户资金异常,前端不同提示 跳转到borrow\_info.html，借款人信息展示页面。  投标人：投标：根据前台传来的标的id 查询此借贷申请的所有信息，查询当前登录用户的账目信息 并校验是否账户金额异常，判断，true：执行投标 创建投标对象，set相关属性，投标表insert此次投标，登录用户的账目对象的可用金额减少，账目对象冻结金额增加，生成投标人账户流水记录，创建流水对象，set相关属性，账目流水表插入资金流水记录，投标申请对象的投标人属性+1，投标申请对象的已投标金额增加，修改投标人账目表-->乐观锁，修改借贷申请表 --> 乐观锁，返回投标结果对象JSONResult。满标后进入满标审核状态。  问题：广告项目内部微服务相互调用解决方案？  回答：使用feign组件进行调用，我们会将微服务的元数据注册到eureka中，包括application.name 如果application.name一样，则表示是用一个服务的集群。通过feign进行调用，feign默认负载均衡方式是使用ribbon。  问题：如何平衡用户体验？  回答：有两种无效情况的流量即重复展示（对单一用户频繁，甚至用户已经产生了有效点击、转化行为）、无价值（对单一用户无需求）。我们可以对于长时间展示无操作，考虑更换广告内容、减少展示频次或不展示；对于已经产生有效行为，考虑更换广告内容或不展示；增加关闭按钮（若用户主动关闭X次，考虑减少展示频次或不展示）；直接用户反馈（考虑减少展示频次或不展示）；分析用户需求（通过机器学习方法建立用户模型，分析判断用户喜好，考虑针对性投放/更换广告）  问题：和并发相关的方案？  回答：同步和锁。同步更多指的是应用程序的层面，多个线程进来，只能一个一个的访问，java中指的是syncrinized关键字。 锁也有2个层面，一个是java中谈到的对象锁，用于[线程同步](https://www.baidu.com/s?wd=%E7%BA%BF%E7%A8%8B%E5%90%8C%E6%AD%A5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)；另外一个层面是数据库的锁；如果是分布式的系统，显然只能利用数据库端的锁来实现。  问题：什么是乐观锁？  回答：乐观锁意思是不锁定表的情况下，利用业务的控制来解决并发问题，这样即保证数据的并发可读性又保证保存数据的排他性，保证性能的同时解决了并发带来的脏数据问题。（乐观锁认为竞争不总是会发生，因此它不需要持有锁，将比较-替换这两个动作作为一个原子操作尝试去修改内存中的变量，如果失败则表示发生冲突，那么就应该有相应的重试逻辑。）（悲观锁认为竞争总是会发生，因此每次对某资源进行操作时，都会持有一个独占的锁，就像synchronized，不管三七二十一，直接上了锁就操作资源了。）  问题：如何实现乐观锁？  回答：前提：在现有表当中增加一个冗余字段，version版本号, long类型。 原理：1）只有当前版本号》=数据库表版本号，才能提交；  2）提交成功后，版本号version ++。 在ormapping增加一属性optimistic-lock="version"即可，样例片段如： <hibernate-mapping> <class name="com.insigma.stock.ABC" optimistic-lock="version" table="T\_Stock" schema="STOCK">  问题：Object 对象的方法有哪些？分别有什么作用？该什么场景用？  回答：clone方法：保护方法，实现对象的浅复制，只有实现了Cloneable接口才可以调用该方法，否则抛出CloneNotSupportedException异常。主要是JAVA里除了8种基本类型传参数是值传递，其他的类对象传参数都是引用传递，我们有时候不希望在方法里讲参数改变，这是就需要在类中复写clone方法。  getClass方法：final方法，获得运行时类型。  toString方法：该方法用得比较多，一般子类都有覆盖。  finalize方法：该方法用于释放资源。finalize方法在垃圾回收时一定会被执行，而如果在此之前显示执行的话，也就是说finalize会被执行两次以上，而在第一次资源已经被释放，那么在第二次释放资源时系统一定会报错，因此一般finalize方法的访问权限和父类保持一致，为protected。  equals方法：该方法是非常重要的一个方法。一般equals和==是不一样的，但是在Object中两者是一样的。子类一般都要重写这个方法。  hashCode方法：该方法用于哈希查找，可以减少在查找中使用equals的次数，重写了equals方法一般都要重写hashCode方法。这个方法在一些具有哈希功能的Collection中用到。一般必须满足obj1.equals(obj2)==true。可以推出obj1.hash- Code()==obj2.hashCode()，但是hashCode相等不一定就满足equals。不过为了提高效率，应该尽量使上面两个条件接近等价。如果不重写hashcode(),在HashSet中添加两个equals的对象，会将两个对象都加入进去。  wait方法：wait方法就是使当前线程等待该对象的锁，当前线程必须是该对象的拥有者，也就是具有该对象的锁。wait()方法一直等待，直到获得锁或者被中断。wait(long timeout)设定一个超时间隔，如果在规定时间内没有获得锁就返回。调用该方法后当前线程进入睡眠状态，直到以下事件发生。（1）其他线程调用了该对象的notify方法。（2）其他线程调用了该对象的notifyAll方法。（3）其他线程调用了interrupt中断该线程。（4）时间间隔到了。此时该线程就可以被调度了，如果是被中断的话就抛出一个InterruptedException异常。  notify方法：该方法唤醒在该对象上等待的某个线程。  notifyAll方法：该方法唤醒在该对象上等待的所有线程。  问题：面向对象的特征有哪些方面？  回答：有四个特征，即抽象（将同类对象的共同特征提取出来构造类）、继承（基于基类创建新类）、封装（将数据隐藏起来，对数据的访问只能通过特定接口）、多态性（不同子类型对象对相同消息作出不同响应）。  问题：float f=3.4;是否正确？  答:不正确。3.4是双精度数，将双精度型（double）赋值给浮点型（float）属于下转型（down-casting，也称为窄化）会造成精度损失，因此需要强制类型转换float f =(float)3.4; 或者写成float f =3.4F;  问题：解释内存中的栈(stack)、堆(heap)和方法区(method area)的用法。  回答：通常我们定义一个基本数据类型的变量，一个对象的引用，还有就是函数调用的现场保存都使用JVM中的栈空间；而通过new关键字和构造器创建的对象则放在堆空间，堆是垃圾收集器管理的主要区域，由于现在的垃圾收集器都采用分代收集算法，所以堆空间还可以细分为新生代和老生代，再具体一点可以分为Eden、Survivor（又可分为From Survivor和To Survivor）、Tenured；方法区和堆都是各个线程共享的内存区域，用于存储已经被JVM加载的类信息、常量、静态变量、JIT编译器编译后的代码等数据；程序中的字面量（literal）如直接书写的100、"hello"和常量都是放在常量池中，常量池是方法区的一部分。栈空间操作起来最快但是栈很小，通常大量的对象都是放在堆空间，栈和堆的大小都可以通过JVM的启动参数来进行调整，栈空间用光了会引发StackOverflowError，而堆和常量池空间不足则会引发OutOfMemoryError。  问题：重载（Overload）和重写（Override）的区别。重载的方法能否根据返回类型进行区分？  回答：方法的重载和重写都是实现多态的方式，区别在于前者实现的是编译时的多态性，而后者实现的是运行时的多态性。重载发生在一个类中，同名的方法如果有不同的参数列表（参数类型不同、参数个数不同或者二者都不同）则视为重载；重写发生在子类与父类之间，重写要求子类被重写方法与父类被重写方法有相同的返回类型，比父类被重写方法更好访问，不能比父类被重写方法声明更多的异常（里氏代换原则）。重载对返回类型没有特殊的要求。  问题：描述一下JVM加载class文件的原理机制？  回答：JVM中类的装载是由类加载器（ClassLoader）和它的子类来实现的，Java中的类加载器是一个重要的Java运行时系统组件，它负责在运行时查找和装入类文件中的类。由于Java的跨平台性，经过编译的Java源程序并不是一个可执行程序，而是一个或多个类文件。当Java程序需要使用某个类时，JVM会确保这个类已经被加载、连接（验证、准备和解析）和初始化。类的加载是指把类的.class文件中的数据读入到内存中，通常是创建一个字节数组读入.class文件，然后产生与所加载类对应的Class对象。加载完成后，Class对象还不完整，所以此时的类还不可用。当类被加载后就进入连接阶段，这一阶段包括验证、准备（为静态变量分配内存并设置默认的初始值）和解析（将符号引用替换为直接引用）三个步骤。最后JVM对类进行初始化，包括：1)如果类存在直接的父类并且这个类还没有被初始化，那么就先初始化父类；2)如果类中存在初始化语句，就依次执行这些初始化语句。类的加载是由类加载器完成的，类加载器包括：根加载器（BootStrap一般用本地代码实现，负责加载JVM基础核心类库rt.jar）、扩展加载器（Extension从java.ext.dirs系统属性所指定的目录中加载类库，它的父加载器是Bootstrap）、系统加载器（System又叫应用类加载器，其父类是Extension。它是应用最广泛的类加载器。它从环境变量classpath或者系统属性java.class.path所指定的目录中加载类，是用户自定义加载器的默认父加载器）和用户自定义类加载器（java.lang.ClassLoader的子类）。从Java 2（JDK 1.2）开始，类加载过程采取了父亲委托机制（PDM）。PDM更好的保证了Java平台的安全性，在该机制中，JVM自带的Bootstrap是根加载器，其他的加载器都有且仅有一个父类加载器。类的加载首先请求父类加载器加载，父类加载器无能为力时才由其子类加载器自行加载。JVM不会向Java程序提供对Bootstrap的引用。  问题：抽象类（abstract class）和接口（interface）有什么异同？  回答：抽象类和接口都不能够实例化，但可以定义抽象类和接口类型的引用。一个类如果继承了某个抽象类或者实现了某个接口都需要对其中的抽象方法全部进行实现，否则该类仍然需要被声明为抽象类。接口比抽象类更加抽象，因为抽象类中可以定义构造器，可以有抽象方法和具体方法，而接口中不能定义构造器而且其中的方法全部都是抽象方法。抽象类中的成员可以是private、默认、protected、public的，而接口中的成员全都是public的。抽象类中可以定义成员变量，而接口中定义的成员变量实际上都是常量。有抽象方法的类必须被声明为抽象类，而抽象类未必要有抽象方法。  问题：Java 中会存在内存泄漏吗，请简单描述一下。  回答：理论上Java因为有垃圾回收机制（GC）不会存在内存泄露问题（这也是Java被广泛使用于服务器端编程的一个重要原因）；然而在实际开发中，可能会存在无用但可达的对象，这些对象不能被GC回收，因此也会导致内存泄露的发生。例如Hibernate的Session（一级缓存）中的对象属于持久态，垃圾回收器是不会回收这些对象的，然而这些对象中可能存在无用的垃圾对象，如果不及时关闭（close）或清空（flush）一级缓存就可能导致内存泄露。  问题：如何实现对象克隆？  回答：有两种方式： 实现Cloneable接口并重写Object类中的clone()方法； 实现Serializable接口，通过对象的序列化和反序列化实现克隆，可以实现真正的深度克隆，  问题：GC是什么？为什么要有GC？  回答：GC是垃圾收集的意思，内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显式操作方法。Java程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用方法：System.gc() 或Runtime.getRuntime().gc() ，但JVM可以屏蔽掉显式的垃圾回收调用。垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低优先级的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清除和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。在Java诞生初期，垃圾回收是Java最大的亮点之一，因为服务器端的编程需要有效的防止内存泄露问题，然而时过境迁，如今Java的垃圾回收机制已经成为被诟病的东西。移动智能终端用户通常觉得iOS的系统比Android系统有更好的用户体验，其中一个深层次的原因就在于Android系统中垃圾回收的不可预知性。垃圾回收机制有很多种，包括：分代复制垃圾回收、标记垃圾回收、增量垃圾回收等方式。标准的Java进程既有栈又有堆。栈保存了原始型局部变量，堆保存了要创建的对象。Java平台对堆内存回收和再利用的基本算法被称为标记和清除，但是Java对其进行了改进，采用“分代式垃圾收集”。这种方法会根据Java对象的生命周期将堆内存划分为不同的区域，在垃圾收集过程中，可能会将对象移动到不同区域：  伊甸园（Eden）：这是对象最初诞生的区域，并且对大多数对象来说，这里是它们唯一存在过的区域。  幸存者乐园（Survivor）：从伊甸园幸存下来的对象会被挪到这里。  终身颐养园（Tenured）：这是足够老的幸存对象的归宿。年轻代收集（Minor-GC）过程是不会触及这个地方的。当年轻代收集不能把对象放进终身颐养园时，就会触发一次完全收集（Major-GC），这里可能还会牵扯到压缩，以便为大对象腾出足够的空间。  问题：比较一下Java和JavaSciprt。  回答：JavaScript 与Java是两个公司开发的不同的两个产品。Java 是原Sun Microsystems公司推出的面向对象的程序设计语言，特别适合于互联网应用程序开发；而JavaScript是Netscape公司的产品，为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言。JavaScript的前身是LiveScript；而Java的前身是Oak语言。 两种语言间的异同作如下比较：  基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object-Based）和事件驱动（Event-Driven）的编程语言，因而它本身提供了非常丰富的内部对象供设计人员使用。  解释和编译：Java的源代码在执行之前，必须经过编译。JavaScript是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行。（目前的浏览器几乎都使用了JIT（即时编译）技术来提升JavaScript的运行效率）。  强类型变量和弱类型变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量是弱类型的，甚至在使用变量前可以不作声明，JavaScript的解释器在运行时检查推断其数据类型。  代码格式不一样。  其实Java和JavaScript最重要的区别是一个是静态语言，一个是动态语言。目前的编程语言的发展趋势是函数式语言和动态语言。在Java中类（class）是一等公民，而JavaScript中函数（function）是一等公民，因此JavaScript支持函数式编程，可以使用Lambda函数和闭包（closure），当然Java 8也开始支持函数式编程，提供了对Lambda表达式以及函数式接口的支持。对于这类问题，在面试的时候最好还是用自己的语言回答会更加靠谱，不要背网上所谓的标准答案。  问题：Java语言如何进行异常处理，关键字：throws、throw、try、catch、finally分别如何使用？  回答：Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它是Throwable类或其子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并可以对其进行处理。Java的异常处理是通过5个关键词来实现的：try、catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果系统会抛出（throw）一个异常对象，可以通过它的类型来捕获（catch）它，或通过总是执行代码块（finally）来处理；try用来指定一块预防所有异常的程序；catch子句紧跟在try块后面，用来指定你想要捕获的异常的类型；throw语句用来明确地抛出一个异常；throws用来声明一个方法可能抛出的各种异常（当然声明异常时允许无病呻吟）；finally为确保一段代码不管发生什么异常状况都要被执行；try语句可以嵌套，每当遇到一个try语句，异常的结构就会被放入异常栈中，直到所有的try语句都完成。如果下一级的try语句没有对某种异常进行处理，异常栈就会执行出栈操作，直到遇到有处理这种异常的try语句或者最终将异常抛给JVM。  问题：举例说明同步和异步。  回答：如果系统中存在临界资源（资源数量少于竞争资源的线程数量的资源），例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就必须进行同步存取（数据库操作中的排他锁就是最好的例子）。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。事实上，所谓的同步就是指阻塞式操作，而异步就是非阻塞式操作。  问题：描述下网页一个 Http 请求，到后端的整个请求过程。  回答：app/web端发送域名到网关（有外网、内网），由nginx进行负载均衡和方向代理把请求下发给不同的网关，请求经过网关访问到微服务（只有内网，外网关闭）。  问题：说一下JWT?  回答：JSON Web Token（JWT）是目前最流行的跨域身份验证解决方案。JWT的原则是在用户登录，服务器经过身份验证之后，将生成一个JSON对象并将其发送回客户端，由客户端进行存储于Cookie或localStorage中（如果将它存储在Cookie中，就可以自动发送，但是不会跨域，因此一般是将它放入HTTP请求的Header Authorization字段中。Authorization: Bearer，当跨域时，也可以将JWT被放置于POST请求的数据主体中。）。此后客户端将在与服务器交互中都会带JWT。用于之后的每一次与服务器通信的身份安全校验。JWT对象为一个长字串，字符之间通过"."分隔符分为三个子串，各字串之间也没有换行符，每一个子串表示了一个功能块，总共有三个部分：JWT头、有效载荷和签名。JWT头部分是一个描述JWT元数据的JSON对象，签名使用的算法默认为HMAC SHA256，JWT令牌类型统一写为JWT，使用Base64 URL算法将JSON对象转换为字符串保存。有效载荷部分，是JWT的主体内容部分，也是一个JSON对象，包含需要传递的数据。 JWT指定七个默认字段供选择：iss发行人、exp到期时间、sub主题、aud用户、nbf在此之前不可用、iat发布时间、jti JWT ID用于标识该JWT，也可以自定义私有字段。默认情况下JWT是未加密的，任何人都可以解读其内容，因此不要构建隐私信息字段，存放保密信息，以防止信息泄露。签名哈希部分是对上面两部分数据签名，通过指定的算法生成哈希，以确保数据不会被篡改。在计算出签名哈希后，JWT头，有效载荷和签名哈希的三个部分组合成一个字符串，每个部分用"."分隔，就构成整个JWT对象。  问题：怎样生成签名？  回答：首先需要指定一个密码（secret）。该密码仅仅为保存在服务器中，并且不能向用户公开。然后使用标头中指定的签名算法（默认情况下为HMAC SHA256）根据以下公式生成签名。  HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload),secret)  问题：JWT存在什么问题？  回答：JWT默认不加密，但可以加密。生成原始令牌后，可以使用改令牌再次对其进行加密。当JWT未加密方法是，一些私密数据无法通过JWT传输。JWT不仅可用于认证，还可用于信息交换。善用JWT有助于减少服务器请求数据库的次数。JWT的最大缺点是服务器不保存会话状态，所以在使用期间不可能取消令牌或更改令牌的权限。也就是说，一旦JWT签发，在有效期内将会一直有效。JWT本身包含认证信息，因此一旦信息泄露，任何人都可以获得令牌的所有权限。为了减少盗用，JWT的有效期不宜设置太长。对于某些重要操作，用户在使用时应该每次都进行进行身份验证。为了减少盗用和窃取，JWT不建议使用HTTP协议来传输代码，而是使用加密的HTTPS协议进行传输。  问题：说一下流程？  回答：我个人主要负责登陆系统和检索系统的核心业务开发，首先广告主登陆的方式是手机号加动态验证码的方式，这里的话，为了短信接口的可用性，我们接入了两个短信接口，分别是阿里大于和容联。前台广告主点击发送短信之后，后台拿到手机号，生成一个随即的6位验证码，然后调用短信接口发送短信，如果发送成功，则将验证码存到redis，并设置过期时间，过期时间为两分钟，然后前台会根据手机接收到的短信输入验证码，我们进行从redis取出并进行判断，如果没有过期并且一致，就说明登陆成功，我们将广告主的IP+手机号+用户名+时间添加到登陆日志表里，并且会查看数据库里有没有这个手机号的用户，如果有，则代表是登陆，如果没有，则添加广告主信息到数据库，然后根据广告主的手机号生成token，并且设置有效期，一般都是7200S,然后返回给前台，前台每次向后台发送请求都会经过zuul网关，然后zuul网关进行拦截并且验证token，如果token有效并且正确，则验证通过，开始转发请求，进行处理业务，  问题：在进行数据库编程时，连接池有什么作用？  回答：由于创建连接和释放连接都有很大的开销（尤其是数据库服务器不在本地时，每次建立连接都需要进行TCP的三次握手，释放连接需要进行TCP四次握手，造成的开销是不可忽视的），为了提升系统访问数据库的性能，可以事先创建若干连接置于连接池中，需要时直接从连接池获取，使用结束时归还连接池而不必关闭连接，从而避免频繁创建和释放连接所造成的开销，这是典型的用空间换取时间的策略（浪费了空间存储连接，但节省了创建和释放连接的时间）。池化技术在Java开发中是很常见的，在使用线程时创建线程池的道理与此相同。基于Java的开源数据库连接池主要有：C3P0、Proxool、DBCP、BoneCP、Druid等。在计算机系统中时间和空间是不可调和的矛盾，理解这一点对设计满足性能要求的算法是至关重要的。大型网站性能优化的一个关键就是使用缓存，而缓存跟上面讲的连接池道理非常类似，也是使用空间换时间的策略。可以将热点数据置于缓存中，当用户查询这些数据时可以直接从缓存中得到，这无论如何也快过去数据库中查询。当然，缓存的置换策略等也会对系统性能产生重要影响，对于这个问题的讨论已经超出了这里要阐述的范围。  问题：什么是DAO模式？  回答：DAO（Data Access Object）顾名思义是一个为数据库或其他持久化机制提供了抽象接口的对象，在不暴露底层持久化方案实现细节的前提下提供了各种数据访问操作。在实际的开发中，应该将所有对数据源的访问操作进行抽象化后封装在一个公共API中。用程序设计语言来说，就是建立一个接口，接口中定义了此应用程序中将会用到的所有事务方法。在这个应用程序中，当需要和数据源进行交互的时候则使用这个接口，并且编写一个单独的类来实现这个接口，在逻辑上该类对应一个特定的数据存储。DAO模式实际上包含了两个模式，一是Data Accessor（数据访问器），二是Data Object（数据对象），前者要解决如何访问数据的问题，而后者要解决的是如何用对象封装数据。  问题：说一下ACID事务？ 回答：ACID有四个特性，即原子性（所谓的原子性就是说，在整个事务中的所有操作，要么全部完成，要么全部不做，没有中间状态。对于事务在执行中发生错误，所有的操作都会被回滚，整个事务就像从没被执行过一样）、一致性（事务的执行必须保证系统的一致性，就拿转账为例，A有500元，B有300元，如果在一个事务里A成功转给B50元，那么不管并发多少，不管发生什么，只要事务执行成功了，那么最后A账户一定是450元，B账户一定是350元）、隔离性（所谓的隔离性就是说，事务与事务之间不会互相影响，一个事务的中间状态不会被其他事务感知）、持久性（所谓的持久性，就是说一单事务完成了，那么事务对数据所做的变更就完全保存在了数据库中，即使发生停电，系统宕机也是如此）这种特性简称刚性事物。问题：分布式事务产生的场景？ 回答：在分布式系统，都会垂直拆分数据库，就比如我做的点阅广告系统就有广告主数据库、物料数据库等业务组成，分为多个数据源，会产生分布式事物问题。 问题：spring事务和分布式事务的区别是什么? 回答：spring事务，本地事务；分布式事务是跨服务间的通讯（不同的数据库连接）。问题:CPA理论？ 回答：CAP关于数据一致性(consistency)、服务可用性(availability)、分区容错性(partition-tolerance)。  数据一致性(consistency)：如果系统对一个写操作返回成功，那么之后的读请求都必须读到这个新数据；如果返回失败，那么所有读操作都不能读到这个数据，对调用者而言数据具有强一致性(strong consistency) (又叫原子性 atomic、线性一致性 linearizable consistency) 。服务可用性(availability)：所有读写请求在一定时间内得到响应，可终止、不会一直等待。分区容错性(partition-tolerance)：在网络分区的情况下，被分隔的节点仍能正常对外服务。  问题：什么是XA接口？  回答：XA是一个分布式事务协议，由Tuxedo提出。XA中大致分为两部分：事务管理器和本地资源管理器。其中本地资源管理器往往由数据库实现，比如Oracle、DB2这些商业数据库都实现了XA接口，而事务管理器作为全局的调度者，负责各个本地资源的提交和回滚。  问题：什么是Jta？  回答：作为java平台上事务规范JTA（Java Transaction API）也定义了对XA事务的支持，实际上，JTA是基于XA架构上建模的，在JTA 中，事务管理器抽象为javax.transaction.TransactionManager接口，并通过底层事务服务（即JTS）实现。像很多其他的java规范一样，JTA仅仅定义了接口，具体的实现则是由供应商(如J2EE厂商)负责提供，目前JTA的实现主要几种：J2EE容器所提供的JTA实现(JBoss)。独立的JTA实现:如JOTM，Atomikos.这些实现可以应用在那些不使用J2EE应用服务器的环境里用以提供分布事事务保证。如Tomcat,Jetty以及普通的java应用。  问题：分布式事物解决方案？  回答：分布式事物问题，在互联网公司比较常见，可以使用全局事物2pc（两段提交协议）、3pc(三段提交协议)，消息中间件、tcc、gts、提供回滚接口、分布式数据库。  问题：索引的优缺点，什么时候使用索引，什么时候不能使用索引？  回答：索引最大的好处是提高查询速度，缺点是更新数据时效率低，因为要同时更新索引对数据进行频繁查询进建立索引，如果要频繁更改数据不建议使用索引。  问题：volatile类型变量提供什么保证?  回答：volatile 主要有两方面的作用:1.避免指令重排2.可见性保证.例如，JVM 或者 JIT为了获得更好的性能会对语句重排序，但是 volatile 类型变量即使在没有同步块的情况下赋值也不会与其他语句重排序。 volatile 提供 happens-before 的保证，确保一个线程的修改能对其他线程是可见的。某些情况下，volatile 还能提供原子性，如读 64 位数据类型，像 long 和 double 都不是原子的(低32位和高32位)，但 volatile 类型的 double 和 long 就是原子的。  问题：java中用到的线程调度算法是什么？  回答：抢占式。一个线程用完CPU之后，操作系统会根据线程优先级、线程饥饿情况等数据算出一个总的优先级并分配下一个时间片给某个线程执行。  问题：如果你提交任务时，线程池队列已满，这时会发生什么？  回答：如果你使用的LinkedBlockingQueue，也就是无界队列的话，没关系，继续添加任务到阻塞队列中等待执行，因为LinkedBlockingQueue可以近乎认为是一个无穷大的队列，可以无限存放任务；如果你使用的是有界队列比方说ArrayBlockingQueue的话，任务首先会被添加到ArrayBlockingQueue中，ArrayBlockingQueue满了，则会使用拒绝策略RejectedExecutionHandler处理满了的任务，默认是AbortPolicy。  问题：如何在两个线程间共享数据？  回答：通过在线程之间共享对象就可以了，然后通过wait/notify/notifyAll、await/signal/signalAll进行唤起和等待，比方说阻塞队列BlockingQueue就是为线程之间共享数据而设计的。  问题：阐述 Java 7 和 Java 8 的区别。  回答：lambda 表达式，Java 8 版本引入的一个新特性。lambda 表达式允许你将功能当作方法参数或将代码当作数据。lambda 表达式还能让你以更简洁的方式表示只有一个方法的接口 (称为函数式接口) 的实例。方法引用，为已命名方法提供了易于阅读的 lambda 表达式。默认方法，支持将新功能添加到类库中的接口，并确保与基于这些接口的旧版本的代码的二进制兼容性。重复注解，支持在同一声明或类型上多次应用同一注解类型。类型注解，支持在任何使用类型的地方应用注解，而不仅限于声明。此特性与可插入型系统一起使用时，可增强对代码的类型检查。 |
| **人资问题(至少30个)** |
| 问题：你介绍一下自己？  回答：面试官您好，我叫晋相瑞，来自山西运城，今年24岁，今天来咱们公司是来应聘java工程师职位，我是15年毕业的，大学学的计算机专业，毕业后从事这方面工作已经三年了。我的上一家公司做的是广告系统的开发，应用的技术是Spring boot框架及Spring Cloud 的一些组件，用到的开发工具是idea，我对现在市面上的一些主流框架也基本都能熟练掌握。  很高兴今天能来贵公司面试，也希望这次面试顺利，能在贵公司长期稳定的发展。以上是我的自我介绍。谢谢！您看，您还有什么想问的或想了解的吗？  问题：你得离职原因是啥？  回答：我希望有一个好的学习和成长的发展平台。  问题：说一下你3到5年的职业规划？  回答：假如有幸入职贵公司，我会首先熟悉公司业务和技术，以及自己的岗位职责，达到胜任自己的岗位，并且在工作之余结合公司的发展学习新技术，使自己更加的优秀。  问题：你上家公司的薪资是多少？  回答：13K    问题：你的期望薪资是多少？  回答：15K  问题：你对于加班有什么看法？  回答：我感觉有利于锻炼我的能力，使自己的各方面能力有更快的提升，在这个竞争激烈的时代我认为是有必要的。并且对于企业而言，我认为合理的加班也有利于企业的快速发展，使个人和企业之间达到共赢。  问题：上家公司地址  回答： 北京步峰科技有限公司 北京市海淀区上地东里一区4号楼220室  北京恒盛昊月科技有限公司 北京昌平区沙河沙阳路18号  问题：大学在哪里读的？  回答：山西建筑工程专修学院，在太原杏花岭三桥  问题：说说你最大的缺点？  回答提示：这个问题企业问的概率很大，通常不希望听到直接回答的缺点是什么等，如果求职者说自己小心眼、爱忌妒人、非常懒、脾气大、工作效率低，企业肯定不会录用你。绝对不要自作聪明地回答“我最大的缺点是过于追求完美”，有的人以为这样回答会显得自己比较出色，但事实上，他已经岌岌可危了。企业喜欢求职者从自己的优点说起，中间加一些小缺点，最后再把问题转回到优点上，突出优点的部分，企业喜欢聪明的求职者。  问题：你对加班的看法？  回答提示：实际上好多公司问这个问题，并不证明一定要加班，只是想测试你是否愿意为公司奉献。  回答样本：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：你对薪资的要求？  回答提示：如果你对薪酬的要求太低，那显然贬低自己的能力；如果你对薪酬的要求太高，那又会显得你分量过重，公司受用不起。一些雇主通常都事先对求聘的职位定下开支预算，因而他们第一次提出的价钱往往是他们所能给予的最高价钱，他们问你只不过想证实一下这笔钱是否足以引起你对该工作的兴趣。  回答样本一：我对工资没有硬性要求，我相信贵公司在处理我的问题上会友善合理。我注重的是找对工作机会，所以只要条件公平，我则不会计较太多。  回答样本二：我受过系统的软件编程的训练，不需要进行大量的培训，而且我本人也对编程特别感兴趣。因此，我希望公司能根据我的情况和市场标准的水平，给我合理的薪水。  回答样本三：如果你必须自己说出具体数目，请不要说一个宽泛的范围，那样你将只能得到最低限度的数字。最好给出一个具体的数字，这样表明你已经对当今的人才市场作了调查，知道像自己这样学历的雇员有什么样的价值。  问题：在五年的时间内，你的职业规划？  回答提示：这是每一个应聘者都不希望被问到的问题，但是几乎每个人都会被问到，比较多的答案是“管理者”。但是近几年来，许多公司都已经建立了专门的技术途径。这些工作地位往往被称作“顾问”、“参议技师” 或“高级软件工程师”等等。当然，说出其他一些你感兴趣的职位也是可以的，比如产品销售部经理，生产部经理等一些与你的专业有相关背景的工作。要知道，考官总是喜欢有进取心的应聘者，此时如果说“不知道”，或许就会使你丧失一个好机会。最普通的回答应该是“我准备在技术领域有所作为”或“我希望能按照公司的管理思路发展”。  问题：你朋友对你的评价？  回答提示： 想从侧面了解一下你的性格及与人相处的问题。  回答样本一：我的朋友都说我是一个可以信赖的人。因为，我一旦答应别人的事情，就一定会做到。如果我做不到，我就不会轻易许诺。  回答样本二：我觉的我是一个比较随和的人，与不同的人都可以友好相处。在我与人相处时，我总是能站在别人的角度考虑问题。  问题：你还有什么问题要问吗？  回答提示：企业的这个问题看上去可有可无，其实很关键，企业不喜欢说“没问题”的人，因为其很注重员工的个性和创新能力。企业不喜欢求职者问个人福利之类的问题，如果有人这样问：贵公司对新入公司的员工有没有什么培训项目，我可以参加吗？或者说贵公司的晋升机制是什么样的？企业将很欢迎，因为体现出你对学习的热情和对公司的忠诚度以及你的上进心。  问题：如果通过这次面试我们单位录用了你，但工作一段时间却发现你根本不适合这个职位，你怎么办？  回答提示：一段时间发现工作不适合我，有两种情况：①如果你确实热爱这个职业，那你就要不断学习，虚心向领导和同事学习业务知识和处事经验，了解这个职业的精神内涵和职业要求，力争减少差距；②你觉得这个职业可有可无，那还是趁早换个职业，去发现适合你的，你热爱的职业，那样你的发展前途也会大点，对单位和个人都有好处。  问题：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？  回答提示：①.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。②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。③.还有一种情况，假如领导要求的方式违背原则，我会坚决提出反对意见，如领导仍固执己见，我会毫不犹豫地再向上级领导反映。  问题：如果你的工作出现失误，给本公司造成经济损失，你认为该怎么办？  回答提示：①我本意是为公司努力工作，如果造成经济损失，我认为首要的问题是想方设法去弥补或挽回经济损失。如果我无能力负责，希望单位帮助解决。②分清责任，各负其责，如果是我的责任，我甘愿受罚；如果是一个我负责的团队中别人的失误，也不能幸灾乐祸，作为一个团队，需要互相提携共同完成工作，安慰同事并且帮助同事查找原因总结经验。③总结经验教训，一个人的一生不可能不犯错误，重要的是能从自己的或者是别人的错误中吸取经验教训，并在今后的工作中避免发生同类的错误。检讨自己的工作方法、分析问题的深度和力度是否不够，以致出现了本可以避免的错误。  问题：如果你做的一项工作受到上级领导的表扬，但你主管领导却说是他做的，你该怎样？  回答提示：我首先不会找那位上级领导说明这件事，我会主动找我的主管领导来沟通，因为沟通是解决人际关系的最好办法，但结果会有两种：①我的主管领导认识到自己的错误，我想我会视具体情况决定是否原谅他。②他更加变本加厉的来威胁我，那我会毫不犹豫地找我的上级领导反映此事，因为他这样做会造成负面影响，对今后的工作不利。  问题：谈谈你对跳槽的看法？  回答提示：①正常的“跳槽”能促进人才合理流动，应该支持。②频繁的跳槽对单位和个人双方都不利，应该反对。  问题：工作中你难以和同事、上司相处，你该怎么办？  回答提示：①我会服从领导的指挥，配合同事的工作。②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。③如果我找不到原因，我会找机会跟他们沟通，请他们指出我的不足，有问题就及时改正。④作为优秀的员工，应该时刻以大局为重，即使在一段时间内，领导和同事对我不理解，我也会做好本职工作，虚心向他们学习，我相信，他们会看见我在努力，总有一天会对我微笑的。  问题：假设你在某单位工作，成绩比较突出，得到领导的肯定。但同时你发现同事们越来越孤立你，你怎么看这个问题？你准备怎么办？  回答提示：①成绩比较突出，得到领导的肯定是件好事情，以后更加努力。②检讨一下自己是不是对工作的热心度超过同事间交往的热心了，加强同事间的交往及共同的兴趣爱好。③工作中，切勿伤害别人的自尊心④不再领导前拨弄是非。  问题：你最近是否参加了培训课程？谈谈培训课程的内容。是公司资助还是自费参加？  回答提示：自费参加，就是XX的培训课程（可以多谈谈自己学的技术）。  问题：你对于我们公司了解多少？  回答提示：在去公司面试前上网查一下该公司主营业务。如回答：贵公司有意改变策略，加强与国外大厂的OEM合作，自有品牌的部分则透过海外经销商。  问题：请说出你选择这份工作的动机？  回答提示：这是想知道面试者对这份工作的热忱及理解度，并筛选因一时兴起而来应试的人，如果是无经验者，可以强调“就算职种不同，也希望有机会发挥之前的经验”。  问题：你最擅长的技术方向是什么？  回答提示：说和你要应聘的职位相关的课程，表现一下自己的热诚没有什么坏处。  问题：你能为我们公司带来什么呢？  回答提示：①假如你可以的话，试着告诉他们你可以减低他们的费用——“我已经接受过XX近两年专业的培训，立刻就可以上岗工作”。② 企业很想知道未来的员工能为企业做什么，求职者应再次重复自己的优势，然后说：“就我的能力，我可以做一个优秀的员工在组织中发挥能力，给组织带来高效率和更多的收益”。企业喜欢求职者就申请的职位表明自己的能力，比如申请营销之类的职位，可以说：“我可以开发大量的新客户，同时，对老客户做更全面周到的服务，开发老客户的新需求和消费。”等等。 |