|  |
| --- |
| 技术问题(至少50个) |

|  |
| --- |
| 问题1：线程与进程  回答：进程是所有线程的集合，每一个线程是进程中的一条执行路径  问题2：为什么要用多线程  回答：主要能体现到多线程提高程序效率  问题3：是继承Thread类好还是实现Runnable接口好  回答： Runnable接口好，因为实现了接口还可以继续继承，Thread类不能在继续继承  问题4： 你在哪里用到了多线程  回答： 主要能体现到多线程提高程序效率  问题5：什么是垃圾回收机制  回答：就是不定时的清理堆内存不可达对象，回收不常用的堆内存，且当你进行手动回收时也不一定会百分百会回收这个垃圾  问题6：什么是垃圾收集器？  回答：串行收集器 单线程收集垃圾，效率低 tomcat调优，初始化的内存越小吞吐量越小Gc回收的垃圾就越多，初始化内存和最大内存越大，那么GC就越少，而且吞吐量就越大  并行收集器 多线程手机垃圾，效率高 tomcat调优就用的并行  问题7：内存泄漏：  回答：定义了很多静态变量，静态变量又在老年取，但是垃圾不会回收    问题8：对SpringBoot的理解  回答：  简单的说，spring boot就是整合了很多优秀的框架，不用我们自己手动的去写一堆xml配置然后进行配置。  从本质上来说，Spring Boot就是Spring,它做了那些没有它你也会去做的Spring Bean配置。它使用“习惯优于配置”（项目中存在大量的配置，此外还内置了一个习惯性的配置，让你无需手动进行配置）的理念让你的项目快速运行起来。使 用Spring Boot很容易创建一个独立运行（运行jar,内嵌Servlet容器）、准生产级别的基于Spring框架的项目，使用Spring Boot你可以不用或者只需要很少的Spring配置。  快速开发，快速整合，配置简化、内嵌服务容器  问题9： 什么是SpringCloud  回答： SpringCloud是微服务的一种解决方案，依赖SpringBoot实现。包含注册中心(eureka)、客户端负载均衡(Ribbon)、网关(zull)、分布式锁、分布式会话等  问题10：为什么要使用SpringCloud  回答：SpringCloud是一套非常完整的微服务解决方案，俗称“微服务全家桶”，几乎内置了微服务所使用的各种技术，可以不必集成第三方依赖。  问题11： 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12： 说说服务雪崩效应  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13：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14：服务降价 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15： SpringBoot如何实现异步执行  回答：在启动类添加@EnableAsync表示开启对异步任务的支持，在异步服务上添加@Async  问题16：SQL语句优化  回答：1）应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。  2）应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：  select id from t where num is null  可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：  select id from t where num=0  3）很多时候用 exists 代替 in 是一个好的选择  4）用Where子句替换HAVING 子句 因为HAVING 只会在检索出所有记录之后才对结果集进行过滤    问题17：MySQL中myisam与innodb的区别，至少5点  回答： 1.InnoDB支持事物，而MyISAM不支持事物  2.InnoDB支持行级锁，而MyISAM支持表级锁  3.InnoDB支持MVCC, 而MyISAM不支持  4.InnoDB支持外键，而MyISAM不支持  5.InnoDB不支持全文索引，而MyISAM支持。  6.InnoDB不能通过直接拷贝表文件的方法拷贝表到另外一台机器， myisam 支持  7.InnoDB表支持多种行格式， myisam 不支持  8.InnoDB是索引组织表， myisam 是堆表  问题18：索引的底层实现原理和优化  回答： B+树，经过优化的B+树  主要是在所有的叶子结点中增加了指向下一个叶子节点的指针，因此InnoDB建议为大部分表使用默认自增的主键作为主索引。  问题19：实践中如何优化MySQL  回答：我当时是按以下四条依次回答的，他们四条从效果上第一条影响最大，后面越来越小。  ① SQL语句及索引的优化  ② 数据库表结构的优化  ③ 系统配置的优化  ④ 硬件的优化  问题20：数据库中的事务是什么  回答：事务（transaction）是作为一个单元的一组有序的数据库操作。如果组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有操作完成，事务则提交，其修改将作用于所有其他数据库进程。如果一个操作失败，则事务将回滚，该事务所有操作的影响都将取消。ACID 四大特性,原子性、隔离性、一致性、持久性。  问题21：对于关系型数据库而言，索引是相当重要的概念，请回答有关索引的几个问题：  回答：  a)、索引的目的是什么？  快速访问数据表中的特定信息，提高检索速度  创建唯一性索引，保证数据库表中每一行数据的唯一性。  加速表和表之间的连接  使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间  b)、索引对数据库系统的负面影响是什么？  负面影响：  创建索引和维护索引需要耗费时间，这个时间随着数据量的增加而增加；索引需要占用物理空间，不光是表需要占用数据空间，每个索引也需要占用物理空间；当对表进行增、删、改、的时候索引也要动态维护，这样就降低了数据的维护速度。  c)、为数据表建立索引的原则有哪些？  在最频繁使用的、用以缩小查询范围的字段上建立索引。  在频繁使用的、需要排序的字段上建立索引  d)、 什么情况下不宜建立索引？  对于查询中很少涉及的列或者重复值比较多的列，不宜建立索引。  对于一些特殊的数据类型，不宜建立索引，比如文本字段（text）等  问题22： #{}和${}的区别是什么  回答：${}是Properties文件中的变量占位符，它可以用于标签属性值和sql内部，属于静态文本替换，比如${driver}会被静态替换为com.mysql.jdbc.Driver。#{}是sql的参数占位符，Mybatis会将sql中的#{}替换为?号，在sql执行前会使用PreparedStatement的参数设置方法，按序给sql的?号占位符设置参数值，比如ps.setInt(0, parameterValue)，#{item.name}的取值方式为使用反射从参数对象中获取item对象的name属性值，相当于param.getItem().getName()。  问题23： Xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？  回答：还有很多其他的标签，<resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，trim|where|set|foreach|if|choose|when|otherwise|bind等，其中<sql>为sql片段标签，通过<include>标签引入sql片段，<selectKey>为不支持自增的主键生成策略标签。  问题24：最佳实践中，通常一个Xml映射文件，都会写一个Dao接口与之对应，请问，这个Dao接口的工作原理是什么？Dao接口里的方法，参数不同时，方法能重载吗？  回答：Dao接口，就是人们常说的Mapper接口，接口的全限名，就是映射文件中的namespace的值，接口的方法名，就是映射文件中MappedStatement的id值，接口方法内的参数，就是传递给sql的参数。Mapper接口是没有实现类的，当调用接口方法时，接口全限名+方法名拼接字符串作为key值，可唯一定位一个MappedStatement，举例：com.mybatis3.mappers.StudentDao.findStudentById，可以唯一找到namespace为com.mybatis3.mappers.StudentDao下面id = findStudentById的MappedStatement。在Mybatis中，每一个<select>、<insert>、<update>、<delete>标签，都会被解析为一个MappedStatement对象。  Dao接口里的方法，是不能重载的，因为是全限名+方法名的保存和寻找策略。  Dao接口的工作原理是JDK动态代理，Mybatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。  问题25：为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？  回答：Hibernate属于全自动ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以，称之为半自动ORM映射工具。  问题26：为什么使用消息队列  回答：解耦,异步,削峰  问题27：使用消息队列有什么缺点  回答：系统可用性降低  系统复杂性增加  问题28：用Runnable还是Thread？  回答：这个问题是上题的后续，大家都知道我们可以通过继承Thread类或者调用Runnable接口来实现线程，问题是，那个方法更好呢？什么情况下使用它？这个问题很容易回答，如果你知道Java不支持类的多重继承，但允许你调用多个接口。所以如果你要继承其他类，当然是调用Runnable接口好了。  问题29：如何处理线程不安全问题 有2种解决方法  回答：1.放在栈里面的数据都是线程安全  2.同步块,同步关键字修饰的都是线程安全  3.final修饰的变量都是线程安全  4.ThreadLoacl放置的变量可以解决线程安全  5.可以考虑JDK5提供的线程安全集合和类  第一，是采用原子变量，毕竟线程安全问题最根本上是由于全局变量和静态变量引起的，只要保证了对于变量的写操作要么全写要么不写，就可以解决线程安全，定义变量用sig\_atomic\_t和volatile。  第二，就是实现线程间同步啦，用互斥锁，信号量。让线程有序的访问变量就可以啦  问题30：解释一下，在数据制作过程中，你如何能从Kafka得到准确的信息?  回答：在数据中，为了精确地获得Kafka的消息，你必须遵循两件事: 在数据消耗期间避免重复，在数据生产过程中避免重复。  这里有两种方法，可以在数据生成时准确地获得一个语义:  • 每个分区使用一个单独的写入器，每当你发现一个网络错误，检查该分区中的最后一条消息，以查看您的最后一次写入是否成功  • 在消息中包含一个主键(UUID或其他)，并在用户中进行反复制  问题31：解释如何提高远程用户的吞吐量?  回答：如果用户位于与broker不同的数据中心，则可能需要调优套接口缓冲区大小，以对长网络延迟进行摊销。  问题32：解释Kafka的用户如何消费信息?  回答：在Kafka中传递消息是通过使用sendfile API完成的。它支持将字节从套接口转移到磁盘，通过内核空间保存副本，并在内核用户之间调用内核。  问题33：解释Kafka的Zookeeper是什么?我们可以在没有Zookeeper的情况下使用Kafka吗?  回答：Zookeeper是一个开放源码的、高性能的协调服务，它用于Kafka的分布式应用。  不，不可能越过Zookeeper，直接联系Kafka broker。一旦Zookeeper停止工作，它就不能服务客户端请求。  • Zookeeper主要用于在集群中不同节点之间进行通信  • 在Kafka中，它被用于提交偏移量，因此如果节点在任何情况下都失败了，它都可以从之前提交的偏移量中获取  • 除此之外，它还执行其他活动，如: leader检测、分布式同步、配置管理、识别新节点何时离开或连接、集群、节点实时状态等等。  问题34：Kafka服务器能接收到的最大信息是多少?  回答：Kafka服务器可以接收到的消息的最大大小是1000000字节。  问题35： SpringBoot装配Bean的原理  回答：通过@EnableAutoConfiguration自动获取配置类信息，使用反射实例化为spring类，然后加载到spring容器  问题36：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37： 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38： ZooKeeper应用场景   1. 回答：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  问题39： ZooKeeper集群选举原理  回答：每台Zookeeper服务器启动时会发起投票，每次投票后，服务器统计投票信息，如果有机器获取半数以上的投票数则leader产生。  问题40：请说明Kafka相对传统技术有什么优势?  回答：Apache Kafka与传统的消息传递技术相比优势之处在于：  快速:单一的Kafka代理可以处理成千上万的客户端，每秒处理数兆字节的读写操作。  可伸缩:在一组机器上对数据进行分区和简化，以支持更大的数据  持久:消息是持久性的，并在集群中进行复制，以防止数据丢失。  设计:它提供了容错保证和持久性  问题41：  回答：  问题42：  回答：  问题43：  回答：  问题44：  回答：  问题45：  回答：  问题46：  回答：  问题47：  回答：  问题48：  回答：  问题49：  回答：  问题50：  回答：  ....... |
| **人资问题(至少30个)** |
| 回答1：简单做一下自我介绍？  面试官您好，我叫白树浩，今年27岁，来自江苏徐州，今天来咱们公司是来应聘JAVA工程师职位的。我毕业后一直从事于JAVA开发工作到现在已经3年多了。我在上一家公司做的是JAVA开发工程师，主要的工作职责是对项目模块的设计开发，做过金融和社交交友的项目。最近做的是社交项目，我们利用当前最流行的技术进行开发,实现前后端分离。我在项目中主要负责用户模块，点赞模块，和交友模块。以上就是我简单的自我介绍 |