|  |
| --- |
| 技术问题(至少50个) 问题1：为什么有线程安全问题  回答：当多个线程同时共享，同一个全局变量或静态变量，做写的操作时，可能会发生数据冲突问题，也就是线程安全问题。但是做读操作是不会发生数据冲突问题。  问题2：为什么使用线程同步或使用锁能解决线程安全问题  回答：将可能会发生数据冲突问题(线程不安全问题)，只能让当前一个线程进行执行。代码执行完成后释放锁，让后才能让其他线程进行执行。这样的话就可以解决线程不安全问题。  问题3：什么是多线程之间同步  回答：当多个线程共享同一个资源,不会受到其他线程的干扰。  问题4：运行时异常与一般异常有何异同  回答：异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。Java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出为被捕获的运行时异常  问题5：Java 中为什么会有 GC 机制呢  回答：安全性考虑，减少内存泄露，减少程序员工作量  问题6：类什么时候被初始化  回答：1创建类的实例，也就是 new 一个对象  2访问某个类或接口的静态变量，或者对该静态变量赋值  3调用类的静态方法  4反射 5初始化一个类的子类 6）JVM 启动时标明的启动类，即文件名和类名相同的那个类  问题7：Java的类加载器的种类都有哪些  回答：1、根类加载器2、扩展类加载器  3、系统(应用)类加载器4、自定义加载器  问题8：什么是HashMap  回答：HashMap是基于哈希表的 Map 接口的实现。此实现提供所有可选的映射操作，并允许使用 null 值和 null 键。HashMap可以理解成是数组和链表的结合。当新建一个HashMap时，会自动创建一个长度为16的数组 |
| 问题9：throw 和 throws 的区别  回答：throw：  1）throw 语句用在方法体内，表示抛出异常，由方法体内的语句处理。  2）throw 是具体向外抛出异常的动作，所以它抛出的是一个异常实例，执行 throw 一定是抛出了某种异常。  throws：  1）throws 语句是用在方法声明后面，表示如果抛出异常，由该方法的调用者来进行异常的处理。  2）throws 主要是声明这个方法会抛出某种类型的异常，让它的使用者要知道需要捕获的异常的类型。  3）throws 表示出现异常的一种可能性，并不一定会发生这种异常。  问题10：抽象类和接口有什么异同  回答：抽象类中可以定义构造器，可以有抽象方法和具体方法，抽象类中的成员可以是 private、默认、protected、public，抽象类中可以定义成员变量，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，一个类只能继承一个抽象类  接口中不能定义构造器，方法全部都是抽象方法，接口中的成员全都是 public 的，接口中定义的成员变量实际上都是常量，接口中不能有静态方法 ，一个类可以实现多个接口  问题11：什么是索引  回答：索引用来快速地寻找那些具有特定值的记录，所有MySQL索引都以B-树的形式保存。如果没有索引，执行查询时MySQL必须从第一个记录开始扫描整个表的所有记录，直至找到符合要求的记录。表里面的记录数量越多，这个操作的代价就越高。如果作为搜索条件的列上已经创建了索引，MySQL无需扫描任何记录即可迅速得到目标记录所在的位置。如果表有1000个记录，通过索引查找记录至少要比顺序扫描记录快100倍。  问题12：**sleep() 和 wait() 有什么区别**  回答：sleep()方法是使线程停止一段时间的方法。在sleep 时间间隔期满后，线程不一定立即恢复执行。这是因为在那个时刻，其它线程可能正在运行而且没有被调度为放弃执行，除非(a)“醒来”的线程具有更高的优先级 (b)正在运行的线程因为其它原因而阻塞。  wait()是线程交互时，如果线程对一个同步对象x 发出一个wait()调用，该线程会暂停执行，被调对象进入等待状态，直到被唤醒或等待时间到。  问题13：**error和exception有什么区别**  回答：error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。  exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。  问题14：**启动一个线程是用run()还是start()**  回答：启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。  问题15：& 和 && 的区别  回答：&运算符有两种用法：(1)按位与；(2)逻辑与。  &&运算符是短路与运算。逻辑与跟短路与的差别是非常巨大的，虽然二者都要求运算符左右两端的布尔值都是  true 整个表达式的值才是 true。  &&之所以称为短路运算是因为，如果&&左边的表达式的值是 false，右边的表达式会被直接短路掉，不会进行运算。很多时候我们可能都需要用&&而不是&，例如在验证用户登录时判定用户名不是 null 而且不是空字符串，应当写为 username != null &&!username.equals("")，二者的顺序不能交换，更不能用&运算符，因为第一个条件如果不成立，根本不能进行字符串的 equals 比较，否则会产生 NullPointerException 异常。注意：逻辑或运算符（|）和短路或运算符（||）的差别也是如此。  问题16：重载（overload）和重写（override）的区别  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17：==和 equals 的区别  回答：equals 和== 最大的区别是一个是方法一个是运算符。  ==：如果比较的对象是基本数据类型，则比较的是数值是否相等；如果比较的是引用数据类型，则比较的是对象的地址值是否相等。  equals()：用来比较方法两个对象的内容是否相等。  问题18：final、finally、finalize 的区别  回答：1）final：用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，被其修饰的类不可继承。  2）finally：异常处理语句结构的一部分，表示总是执行。  3）finalize：Object 类的一个方法，在垃圾回收器执行的时候会调用被回收对象的此方法，可以覆盖此方法  提供垃圾收集时的其他资源回收，例如关闭文件等。该方法更像是一个对象生命周期的临终方法，当该方法被系统调用则代表该对象即将“死亡”，但是需要注意的是，我们主动行为上去调用该方法并不会导致该对象“死亡”，这是一个被动的方法（其实就是回调方法），不需要我们调用。  问题19：HashMap 和 HashTable 有什么区别  回答：HashMap 是线程不安全的,HashMap 是一个接口,是 Map 的一个子接口,是将键映射到值得对象,不允许键值重复, 允许空键和空值;由于非线程安全,HashMap 的效率要较 HashTable 的效率高一些.  HashTable 是线程安全的一个集合,不允许 null 值作为一个 key 值或者 Value 值;  HashTable 是 sychronize,多个线程访问时不需要自己为它的方法实现同步,而 HashMap 在被多个线程访问的时候需要自己为它的方法实现同步;  问题20：List和Map区别  回答：一个是存储单列数据的集合，另一个是存储键和值这样的双列数据的集合，List 中存储的数据是有顺序，并且允许重复；Map 中存储的数据是没有顺序的，其键是不能重复的，它的值是可以有重复的  问题21：接口是否可继承接口? 抽象类是否可实现(implements) 接口?抽象类是否可 抽象类是否可继承具体类 继承具体类(concrete class)? 抽象类中是否可以有静态的main方法  回答：接口可以继承接口。抽象类可以实现(implements)接口，抽象类可以继承具体类。抽象类中可以有静态的 main 方法。  问题22：HTTP 请求的 GET 与 与 POST  回答：servlet 有良好的生存期的定义，包括加载和实例化、初始化、处理请求以及服务结束。这个生存期由 javax.servlet.Servlet 接口的 init,service 和 destroy 方法表达。  问题23：什么是多线程  回答：及时提高程序的效率（同时执行，互不影响）  问题24：多线程应用场景  回答：要能体现到多线程提高程序效率。  举例: 迅雷多线程下载、分批发送短信、爬虫等  问题25：创建多线程有哪些方式  回答：1、使用继承Therad类方式 继承Thread类重写run方法 2、使用实现runlabe接口方式3、使用匿名内部类方式 4、callable 5、使用线程池创建线程。  问题26：对分布式事务了解吗  回答：简单的说，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，而且属于不同的应用，分布式事务需要保证这些小的操作要么全部成功，要么全部失败，分布式事务就是为了保证不同的数据库一致性。  问题27：线程的5种状态   1. 回答：创建（new Thread()）、 2. 就绪（调用线程start()方法）、 3. 运行（当线程获得CPU事件后，才进入运行状态，真正开始执行run()方法）、 4. 阻塞（由于各种原因进入阻塞状态：1>线程通过调用sleep()方法进入睡眠状态；   2>线程调用一个在I/O上被阻塞的操作，即该操作在输入输出操作完成之前不会返回到它的调用者；3>线程试图得到一个锁，而该锁正被其他线程持有；4>线程在等待某个触发条件。）   1. 死亡（两个原因会导致线程死亡：1）run方法正常退出而自然死亡，2）出现异常而未被捕获，从而终止了run方法而使线程猝死。   为了确定线程在当前是否存活（要么是可运行的，要么是被阻塞了），需要使用isAlive()方法。如果是可运行或被阻塞，这个方法返回true；如果线程仍旧是new状态且不是是可运行的，或者线程死亡了，则返回false；  问题28：拦截器和过滤器有哪些区别  回答：1.拦截器是基于java反射机制的，过滤器是基于函数回调  2.拦截器只对action请求起作用，而过滤器可以对几乎所有请求起作用  3.拦截器不依赖于servlet容器，过滤器依赖于servlet容器  4.在action生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化的时候被调用一次。  问题29：**Spring框架的优点**  回答：轻量：Spring是轻量级的，基本的版本大小为2MB。  •控制反转：Spring通过控制反转实现了松散耦合，对象们给出它们的依赖，而不是创建或查找依赖的对象们。  •面向切面的编程AOP:Spring支持面向切面的编程，并且把应用业务逻辑和系统服务分开。  •容器：Spring包含并管理应用中对象的生命周期和配置  •MVC框架： Spring-MVC。  •事务管理：Spring提供一个持续的事务管理接口，可以扩展到上至本地事务下至全局事务JTA。  •异常处理：Spring提供方便的API把具体技术相关的异常。  问题30：**Eureka**  回答：Eureka是Netflix开发的服务发现框架，主要用于定位运行在AWS(亚马逊公司旗下云计算服务平台)域中的中间层服务，以达到负载均衡和中间层服务故障转移的目的。  Eureka包含两个组件：Eureka Server和Eureka Client。  Eureka Server提供服务注册服务，各个节点启动后，会在Eureka Server中进行注册，这样EurekaServer中的服务注册表中将会存储所有可用服务节点的信息，服务节点的信息可以在界面中直观的看到。  Eureka Client是一个java客户端，用于简化与Eureka Server的交互，客户端同时也就是一个内置的、使用轮询(round-robin)负载算法的负载均衡器。  问题31：什么是垃圾回收机制  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  C++ C z中的清理都是手动清理的，但是java就是GC清理，这个对象有没有响应，就报内存泄漏的错误  问题32：java内存模型  回答：决定了一个线程与另一个线程是否可见。分为主内存和私有本地内存  问题33：一个线程使用同步函数，另一个线程使用同步代码块(this),能够同步吗  回答：能同步  问题34：  回答：  问题35：  回答：  问题36：  回答：  问题37：  回答：  问题38：  回答：  问题39：  回答：  问题40：  回答：  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：  问题51：  回答： |
| **人资问题(至少30个)** |
| 问题1：你得离职原因是啥  回答：因为我在上一家单位我们的项目一期已经做得差不多了,我们组的老大说留下几位技术骨干就ok了。我想了一下与我的职业发展规划不符，希望获得一份更好地工作。对自己要求比较高然后就选择离职。  回答2：你对加班的看法  回答：在我看来加班分为两种,一种是我没有完成领导交代给我的任务,自己自愿加班,完成自己的工作进度。从而迎接明天新的任务。在有另外一种是在我们这一行加班是不可避免的，如果公司在需要我在工作上加班时我会义不容辞的,因为这样可以更好地提升自己,利于我的职业成长  回答3：说说你最大的缺点  回答：我觉得我有时候会过分在意别人的感受, 比方说, 不敢直接表达不同意见, 因为觉得会让对方丢面子,其实这样做很不利于快速有效地开展工作。我希望自己能够逐渐学会更加爽快, 对人对事更加直接。  回答4：  回答：  回答5：  回答：  回答6：  回答：  回答7：  回答：  回答8：  回答：  回答9：  回答：  回答10：  回答：  回答11：  回答：  回答12：  回答：  回答13：  回答：  回答14：  回答：  回答15：  回答：  回答16：  回答：  回答17：  回答：  回答18：  回答：  回答19：  回答：  回答20：  回答：  回答21：  回答：  回答22：  回答：  回答23：  回答：  回答24：  回答：  回答25：  回答：  回答26：  回答：  回答27：  回答：  回答28：  回答：  回答29：  回答：  回答30：  回答： |