### 互联网 Java 工程师面试题

##### MyBatis 面试题

###### 什么是Mybatis？

|  |
| --- |
|  |

###### Mybaits 的优点：

|  |
| --- |
|  |

###### MyBatis 框架适用场合：

|  |
| --- |
|  |

1. 占位

###### MyBatis 与 Hibernate 有哪些不同？

|  |
| --- |
|  |

###### #{}和${}的区别是什么？

|  |
| --- |
|  |

###### 当实体类中的属性名和表中的字段名不一样 ，怎么办 ？

|  |
| --- |
|  |

###### 模糊查询 like 语句该怎么写?

|  |
| --- |
|  |

1. 占位

###### Mybatis 是如何进行分页的？分页插件的原理是什么？

|  |
| --- |
|  |

###### Mybatis 是如何将 sql 执行结果封装为目标对象并返回的？

|  |
| --- |
|  |

###### 如何执行批量插入?

|  |
| --- |
|  |

###### 如何获取自动生成的(主)键值?

|  |
| --- |
|  |

###### 在 mapper 中如何传递多个参数?

|  |
| --- |
|  |

###### Mybatis 动态 sql 有什么用？执行原理？有哪些动态 sql？

|  |
| --- |
|  |

###### Xml 映射文件中，除了常见的 select|insert|updae|delete

|  |
| --- |
|  |

###### 17 、 Mybatis 的 Xml 映射 文件 中， 不同 的 Xml 映射 文件 ， id 是否 可以 重复 ？

|  |
| --- |
|  |

###### 18、为什么说 Mybatis 是半自动 ORM 映射工具？它与全自动

|  |
| --- |
|  |

###### 19、 一对一、一对多的关联查询 ？

|  |
| --- |
|  |

###### 20、 MyBatis 实现一对一有几种方式?具体怎么操作的？ 45

|  |
| --- |
|  |

###### 21、 MyBatis 实现一对多有几种方式,怎么操作的？

|  |
| --- |
|  |

###### 22、 Mybatis 是否支持延迟加载？如果支持，它的实现原理是

|  |
| --- |
|  |

###### 23、 Mybatis 的一级、二级缓存:

|  |
| --- |
|  |

###### 24、什么是 MyBatis 的接口绑定？有哪些实现方式？

|  |
| --- |
|  |

###### 26、 Mapper 编写有哪几种方式？

|  |
| --- |
|  |

###### 27、简述 Mybatis 的插件运行原理，以及如何编写一个插件。

|  |
| --- |
|  |

##### ZooKeeper 面试题

###### 1. ZooKeeper 面试题？

|  |
| --- |
|  |

###### 2. ZooKeeper 提供了什么？

|  |
| --- |
|  |

###### 3. Zookeeper 文件系统

|  |
| --- |
|  |

###### 4. ZAB 协议？

|  |
| --- |
|  |

###### 5. 四种类型的数据节点 Znode

|  |
| --- |
|  |

###### 6. Zookeeper Watcher 机制 -- 数据变更通知

|  |
| --- |
|  |

###### 7. 客户端注册 Watcher 实现

|  |
| --- |
|  |

###### 8. 服务端处理 Watcher 实现

|  |
| --- |
|  |

###### 9. 客户端回调 Watcher

|  |
| --- |
|  |

###### 10. ACL 权限控制机制

|  |
| --- |
|  |

###### 11. Chroot 特性 52

|  |
| --- |
|  |

###### 12. 会话管理 52

|  |
| --- |
|  |

###### 13. 服务器角色 52

|  |
| --- |
|  |

###### 14. Zookeeper 下 Server 工作状态 53

|  |
| --- |
|  |

###### 15. 数据同步 53

|  |
| --- |
|  |

###### 16. zookeeper 是如何保证事务的顺序一致性的

|  |
| --- |
|  |

###### 17. 分布式集群中为什么会有 Master？

|  |
| --- |
|  |

###### 18. zk 节点宕机如何处理？

|  |
| --- |
|  |

###### 19. zookeeper 负载均衡和 nginx 负载均衡区别

|  |
| --- |
|  |

###### 20. Zookeeper 有哪几种几种部署模式？

|  |
| --- |
|  |

###### 21. 集群最少要几台机器，集群规则是怎样的?

|  |
| --- |
|  |

###### 22. 集群支持动态添加机器吗？

|  |
| --- |
|  |

###### 23. Zookeeper 对节点的 watch 监听通知是永久的吗？为什么不是永久的?

|  |
| --- |
|  |

###### 24. Zookeeper 的 java 客户端都有哪些？

|  |
| --- |
|  |

###### java 客户 端： zk 自带 的 zkclient 及 Apache 开源 的 Curator

|  |
| --- |
|  |

###### 25. chubby 是什么，和 zookeeper 比你怎么看？

|  |
| --- |
|  |

###### 26. 说几个 zookeeper 常用的命令。

|  |
| --- |
|  |

###### 27. ZAB 和 Paxos 算法的联系与区别？

|  |
| --- |
|  |

###### 28. Zookeeper 的典型应用场景

|  |
| --- |
|  |

##### Zookeeper附加题

###### 5.1 面试官 ：工作中使用过 Zookeeper 嘛？你知道它是什么 ，

|  |
| --- |
|  |

###### 2.2 面试官 ：说下什么是命名服务，什么是配置管理， 又什么

|  |
| --- |
|  |

###### 5.3 面试官 ：你提到了znode 节点， 那你知道znode 有几种

|  |
| --- |
|  |

###### zookeeper 的数据模型

|  |
| --- |
|  |

###### znode 的 4 种类型

|  |
| --- |
|  |

###### 5.4 面试官 ：你知道znode 节点里面存储的是什么吗？每个

|  |
| --- |
|  |

###### znode 节点里面存储的是什么？

|  |
| --- |
|  |

###### 每个节点的数据最大不能超过多少呢

|  |
| --- |
|  |

###### 5.5 面试官 ：你知道znode 节点上的监听机制嘛？讲下

|  |
| --- |
|  |

###### Watcher 监听机制 63

###### Watcher 监听机制的工作原理

|  |
| --- |
|  |

###### Watcher 特性总结

|  |
| --- |
|  |

###### 5.6 面试官 ：你对 Zookeeper 的数据结构都有一定了解， 那

|  |
| --- |
|  |

###### 5.7 面试官 ：你刚提到顺序一致性， 那 zookeeper 是如何保

|  |
| --- |
|  |

###### 5.8 面试官 ：你提到了 Leader ，你知道 Zookeeper 的服务器有几种角色嘛？Zookeeper 下 Server 工作状态又有几种呢？

|  |
| --- |
|  |

###### Zookeeper 服务器角色

|  |
| --- |
|  |

###### Zookeeper 下 Server 工作状态

|  |
| --- |
|  |

###### 5.9 面试官 ：你说到服务器角色是基于 ZooKeeper 集群的，那你画一下 ZooKeeper集群部署图吧？ZooKeeper是如何保证主从节点数据一致性的呢？

|  |
| --- |
|  |

###### ZooKeeper 集群部署图

|  |
| --- |
|  |

###### ZooKeeper 如何保证主从节点数据一致性

|  |
| --- |
|  |

###### 5.10 面试官 ：Leader 挂了， 进入崩溃恢复，是如何选举Leader 的呢？你讲一下 ZooKeeper 选举机制吧

|  |
| --- |
|  |

###### 服务器启动的 Leader 选举

|  |
| --- |
|  |

###### 服务器运行期间的 Leader 选举

|  |
| --- |
|  |

###### 5.11 面试官 ：你前面提到在项目中使用过Zookeeper 的分布式锁，讲一下zk 分布式锁的实现原理吧？

###### 5.12 面试官 ：好的，最后一道题，你说说dubbo 和Zookeeper 的关系吧 ，为什么选择 Zookeeper 作为注册中心 ？

|  |
| --- |
|  |

##### Dubbo 面试题

###### 1、为什么要用 Dubbo？

|  |
| --- |
|  |

###### 3、默认使用的是什么通信框架，还有别的选择吗?

|  |
| --- |
|  |

###### 4、服务调用是阻塞的吗？

|  |
| --- |
|  |

###### 5、一般使用什么注册中心？还有别的选择吗？

|  |
| --- |
|  |

###### 6、默认使用什么序列化框架，你知道的还有哪些？

|  |
| --- |
|  |

###### 7、服务提供者能实现失效踢出是什么原理？

|  |
| --- |
| 撒打发 |

###### 8、服务上线怎么不影响旧版本？

|  |
| --- |
|  |

###### 9、如何解决服务调用链过长的问题？

|  |
| --- |
|  |

###### 10、说说核心的配置有哪些？

|  |
| --- |
|  |

###### 11、 Dubbo 推荐用什么协议？

|  |
| --- |
|  |

###### 12、同一个服务多个注册的情况下可以直连某一个服务吗？

|  |
| --- |
|  |

###### 13、画一画服务注册与发现的流程图？

|  |
| --- |
|  |

###### 14、 Dubbo 集群容错有几种方案？

|  |
| --- |
|  |

###### 15、 Dubbo 服务降级，失败重试怎么做？

|  |
| --- |
|  |

###### 16、 Dubbo 使用过程中都遇到了些什么问题？

|  |
| --- |
|  |

###### 17、 Dubbo Monitor 实现原理？

|  |
| --- |
|  |

###### 18、 Dubbo 用到哪些设计模式？

|  |
| --- |
|  |

###### 19、 Dubbo 配置文件是如何加载到 Spring 中的？

|  |
| --- |
|  |

###### 20、 Dubbo SPI 和 Java SPI 区别？

|  |
| --- |
|  |

###### 21、 Dubbo 支持分布式事务吗？

|  |
| --- |
|  |

###### 22、 Dubbo 可以对结果进行缓存吗？

|  |
| --- |
|  |

###### 23、服务上线怎么兼容旧版本？

|  |
| --- |
|  |

###### 24、 Dubbo 必须依赖的包有哪些

|  |
| --- |
|  |

###### 25、 Dubbo telnet 命令能做什么？

|  |
| --- |
|  |

###### 26、 Dubbo 支持服务降级吗？

|  |
| --- |
|  |

###### 27、 Dubbo 如何优雅停机？

|  |
| --- |
|  |

###### 28、 Dubbo 和 Dubbox 之间的区别？

|  |
| --- |
|  |

###### 29、 Dubbo 和 Spring Cloud 的区别？

|  |
| --- |
|  |

###### 30、你还了解别的分布式框架吗？

|  |
| --- |
|  |

##### Elasticsearch 面试题

###### 1、 elasticsearch 了解多少，说说你们公司 es 的集群架构，索引数据大小，分片有多少，以及一些调优手段 。

|  |
| --- |
|  |

###### 2、elasticsearch 的倒排索引是什么

|  |
| --- |
|  |

###### 3、elasticsearch 索引数据多了怎么办，如何调优，部署

|  |
| --- |
|  |

###### 4、 elasticsearch 是如何实现 master 选举的

|  |
| --- |
|  |

###### 5、详细描述一下 Elasticsearch 索引文档的过程

|  |
| --- |
|  |

###### 6、详细描述一下 Elasticsearch 搜索的过程？

|  |
| --- |
|  |

###### 7、 Elasticsearch 在部署时，对 Linux 的设置有哪些优化方法

|  |
| --- |
|  |

###### 8、 lucence 内部结构是什么？

|  |
| --- |
|  |

###### 9、Elasticsearch 是如何实现 Master 选举的？

|  |
| --- |
|  |

###### 12、详细描述一下Elasticsearch 索引文档的过程。

|  |
| --- |
|  |

###### 13、详细描述一下 Elasticsearch 更新和删除文档的过程。

|  |
| --- |
|  |

###### 14、详细描述一下 Elasticsearch 搜索的过程。

|  |
| --- |
|  |

###### 15、在 Elasticsearch 中，是怎么根据一个词找到对应的倒排索引的？

|  |
| --- |
|  |

###### 16、 Elasticsearch 在部署时，对 Linux 的设置有哪些优化方法？

|  |
| --- |
|  |

###### 17、对于 GC 方面，在使用 Elasticsearch 时要注意什么？

|  |
| --- |
|  |

###### 18、Elasticsearch 对于大数据量(上亿量级) 的聚合如何实现？

|  |
| --- |
|  |

###### 19、在并发情况下，Elasticsearch 如果保证读写一致？

|  |
| --- |
|  |

###### 20、如何监控 Elasticsearch 集群状态？

|  |
| --- |
|  |

###### 21、介绍下你们电商搜索的整体技术架构。

|  |
| --- |
|  |

###### 22、介绍一下你们的个性化搜索方案？

|  |
| --- |
|  |

###### 23、是否了解字典树？

|  |
| --- |
|  |

###### 24、拼写纠错是如何实现的？

|  |
| --- |
|  |

##### Memcached 面试题

###### 1、Memcached 是什么，有什么作用？

|  |
| --- |
|  |

###### 2、 Memcached 服务分布式集群如何实现？

|  |
| --- |
|  |

###### 3、 Memcached 服务特点及工作原理是什么？

|  |
| --- |
|  |

###### 4、简述 Memcached 内存管理机制原理？

|  |
| --- |
|  |

###### 5、 memcached 是怎么工作的？

|  |
| --- |
|  |

###### 6、 memcached 最大的优势是什么？

|  |
| --- |
|  |

###### 7、 memcached 和 MySQL 的 query

|  |
| --- |
|  |

###### 8、memcached 和服务器的 local cache (比如 PHP 的 APC、mmap 文件等)相比，有什么优缺点？

|  |
| --- |
|  |

###### 9、 memcached 的 cache 机制是怎样的？

|  |
| --- |
|  |

###### 10、 memcached 如何实现冗余机制？

|  |
| --- |
|  |

###### 11、 memcached 如何处理容错的？

|  |
| --- |
|  |

###### 13、如果缓存数据在导出导入之间过期了，您又怎么处理这些数据呢？

|  |
| --- |
|  |

###### 14、 memcached 是如何做身份验证的？

|  |
| --- |
|  |

###### 15、 memcached 的多线程是什么？如何使用它们？

|  |
| --- |
|  |

###### 16、 memcached 能接受的 key 的最大长度是多少？

|  |
| --- |
|  |

###### 17、 memcached 最大能存储多大的单个 item？

|  |
| --- |
|  |

###### 18、 memcached 能够更有效地使用内存吗？

|  |
| --- |
|  |

###### 19、什么是二进制协议，我该关注吗？

|  |
| --- |
|  |

###### 20、 memcached 的内存分配器是如何工作的？为什么不适用malloc/free！？为何要使用 slabs？

|  |
| --- |
|  |

###### 22、如何实现集群中的 session 共享存储？

|  |
| --- |
|  |

###### 23、 memcached 与 redis 的区别？

|  |
| --- |
|  |

##### Redis 面试题

###### 1、什么是 Redis?

|  |
| --- |
| 1.开源，BSD协议的key-value数据库  2.支持持久化，丰富的数据结构string hash zset list 等  3.支持数据备份-master-slave模式数据备份  4.优势 高性能 丰富数据结构 原子 丰富特性 |

###### 2、 Redis 的数据类型？

|  |
| --- |
| String hash zset list ；支持事务；丰富特性； |

###### 3、使用 Redis 有哪些好处？

|  |
| --- |
| 速度快；丰富数据类型；支持事务；丰富特性 |

###### 4、Redis 相比 Memcached 有哪些优势？

|  |
| --- |
| 1.数据类型多 2.速度快 3.可持久化 |

###### 5、 Memcache 与 Redis 的区别都有哪些？

|  |
| --- |
| 1.除了上面不同，redis是自己构建的vm机制，比一般系统调用系统函数速度快 |

###### 6、 Redis 是单进程单线程的？

|  |
| --- |
| 是的，redis利用队列技术将并发访问串行，减少现场切换损耗的时间。 |

###### 7、 一个字符串类型的值能存储最大容量是多少？

|  |
| --- |
| 512M |

###### 8、Redis 的持久化机制是什么？各自的优缺点？

|  |
| --- |
| RDB：fork子进程处理持久化，元数据，每隔一段时间做一次，恢复快。  AOF：另启现场持久化，AOF文件日志数据体积大，恢复慢，每秒一次所以丢失的风险小；恢复慢； |

###### 9、Redis常见性能问题和解决方案：

|  |
| --- |
| 1.master不要写内存快照 2.重要数据AOF 1秒一次 3.Master和slave在同一局域网下 4.主从用单向链表连接比较稳定 |

###### 10、redis 过期键的删除策略？

|  |
| --- |
| 1.定时删除 2.定期删除。3惰性删除。 |

###### 11、Redis 的回收策略(淘汰策略) ?

|  |
| --- |
| 从（volatile，allkeys）和（lru，ttl，ramdom）的排列组合  No-enviction永远不回收-可能存在内存泄漏风险。 |

###### 12、为什么 redis 需要把所有数据放到内存中？

|  |
| --- |
| 1.放内存读写性能高 2.异步持久化到磁盘数据安全 兼顾二者的优秀设计 |

###### 13、Redis的同步机制了解么？

|  |
| --- |
| 主从同步，1.第一次同步做一次bgsave，此时后续改动放入buffer，2.rdb全量完成后，在同步这期间的主节点改动信息。 |

###### 14. Pipeline 有什么好处，为什么要用 pipeline？

|  |
| --- |
| 可以将多次IO返回的时间缩减为１次，redis - benchmark压测是这个是QPS的重要因素是pipeline的批次指令数目 |

###### 15、是否使用过 Redis 集群，集群的原理是什么？

|  |
| --- |
| 1.redis Sentinal:着眼于高可用 2.Redis Cluster着眼于扩展性 |

###### 16、Redis集群方案什么情况下会导致整个集群不可用？

|  |
| --- |
| A，B，C三个节点中B宕机。 |

###### 17、Redis支持的Java客户端都有哪些？官方推荐用哪个？

|  |
| --- |
| Redsison Jedis等，官方支持Redisson。 |

###### 18、Jedis与Redisson 对比有什么优缺点？

|  |
| --- |
| 1.Jedis是java实现的客户端，空能丰富。  2.Redission 功能简单，不支持事务 |

###### 19、Redis如何设置密码及验证密码？

|  |
| --- |
| 设置 密码 ： config set requirepass 123456  授权 密码 ： auth 123456 |

###### 20、说说Redis哈希槽的概念？

|  |
| --- |
| 16384个哈希槽，每个节点负责一部分hash槽。 |

###### 21、Redis集群的主从复制模型是怎样的？

|  |
| --- |
| 高可用，主从复制有N-1个复制品。 |

###### 22、Redis集群会有写操作丢失吗？为什么？

|  |
| --- |
| Redis不能保证数据强一致性 |

###### 23、Redis集群之间是如何复制的？

|  |
| --- |
| 异步复制 |

###### 24、Redis集群最大节点个数是多少？

|  |
| --- |
| 16384个 |

###### 25、Redis集群如何选择数据库？

|  |
| --- |
| 集群客户端是不支持多数据库db的，只能使用数据库0; |

###### 26、怎么测试Redis 的连通性？

|  |
| --- |
| Ping |

###### 27、怎么理解Redis事务？

|  |
| --- |
| 事务是一个单独隔离操作，所有命令都会序列化，是一个原子操作。 |

###### 28、Redis事务相关的命令有哪几个？

|  |
| --- |
| MULTI EXEC DISCARD WATCH |

###### 29、Redis key的过期时间和永久有效分别怎么设置？

|  |
| --- |
| EXPIRE 和 PERSIST命令 |

###### 30、Redis如何做内存优化？

|  |
| --- |
| 尽量使用散列表（hash） |

###### 31、Redis回收进程如何工作的？

|  |
| --- |
| 一个新的命令结果导致大量内存被使用，超过内存边界就会启动约定好的策略 进行回收。 |

###### 33、Redis 的内存用完了会发生什么？

|  |
| --- |
| 内存达到上线就会冲刷掉旧的日志 |

###### 34、一个 Redis 实例最多能存放多少的 keys？ List、Set、Sorted Set 他们最多能存放多少元素？

|  |
| --- |
| 2^32个 |

###### 35、MySQL 里有 2000w 数据，redis 中只存 20w 的数据，如何保证 redis 中的数据都是热点数据？

|  |
| --- |
| 1实行淘汰策略 2.redis的内存 3.将MySql垂直分表，让重要数据对应的数据放在Reids里面。 |

###### 36、Redis最适合的场景？

|  |
| --- |
| 1.会话缓存 2.全页缓存 3.队列 4.排行榜、计数 5.发布、订阅 |

###### 37、假如 Redis 里面有 1 亿个 key，其中有 10w 个 key 是以某个固定的已知的前缀开头的，如果将它们全部找出来？

|  |
| --- |
|  |

###### 38、如果有大量的 key 需要设置同一时间过期，一般需要注意什么？

|  |
| --- |
| 防止雪崩，不要再同一秒失效，要均匀分布在这个时间点前后一段时间。 |

###### 40、使用过 Redis分布式锁么，它是什么回事？

|  |
| --- |
| 先拿 setnx来争抢锁，抢到之后， 再用 expire 给锁加一个过期时间防止锁忘记了释放 |

##### 2. Redis附加题 115

###### 2.1 什么是 Redis？它主要用来什么的？

|  |
| --- |
| C语言编写，支持网络，基于内存可持久化的k-v数据库 |

###### 2.2 说说 Redis 的基本数据结构类型

|  |
| --- |
| 基本类型：String hash zset list Set  特殊类型： Geospatial Hyperloglog Bitmap |

2.2.1 Redis 的五种基本数据类型

|  |
| --- |
| String int（小于8）embstr（小于39）raw（大于）  Hash:hset key filed value, k-v中v本身就是一个键值对  List：lpush+lpop =栈；lpush+rpop 队列； lpush+itrim 有限集合 ； lpush+brpop=消息队列  Set:无重复原生  ZSet:无重复排序元素 |

2.2.2 Redis 的三种特殊数据类型

|  |
| --- |
| GEO：地利位置 HyperLogLog:基数统计 ； Bitmaps：比特单位的数组 |

###### 2.3 Redis 为什么这么快？

|  |
| --- |
| 2.3.1 基于内存存储实现  2.3.2 高效的数据结构  2.3.3 合理的数据编码 SDS简单动态字符串  2.3.4 合理的线程模型 I\O多路复用 |

###### 2.4 什么是缓存击穿、缓存穿透、缓存雪崩？

|  |
| --- |
| 2.4.1 缓存穿透问题：对个无效的缓存请求发过来  2.4.2 缓存雪奔问题：多个缓存key同时失效  2.4.3 缓存击穿问题：某一个key短时间收到了超巨量的请求 |

###### 2.5 什么是热 Key 问题，如何解决热key 问题

|  |
| --- |
| 热key：访问频率高的key，大批量请求会导致宕机  对策：1识别热key，2快速启动Redis扩容，3使用二级缓存即JVM本地缓存，减少redis请求 |

###### 2.6. Redis 过期策略和内存淘汰策略

###### 2.7 说说 Redis 的常用应用场景

|  |
| --- |
| 2.7.1 缓存  2.7.2 排行榜  2.7.3 计数器应用  2.7.4 共享 Session  2.7.5 分布式锁  2.7.6 社交网络  2.7.7 消息队列  2.7.8 位操作 :使用 setbit、getbit、bitcount 命令。原理是： redis 内构建一个足够长的数组 ，数组的每个元素只能是 0 和 1 两个值 ，然后这个数组的下标 index 用来表示用户 id ( 必须是数字哈 ) |

###### 2.8 Redis 持久化有哪几种方式，怎么选？

###### 2.9 怎么实现 Redis 的高可用？(Redis 主从、哨兵、集群)

|  |
| --- |
| 2.9.1 Redis 主从  2.9.1.1 Redsi 主从概念  2.9.1.2 Redis 主从同步过程  2.9.1.3 Redis 主从的一些注意点：主从不一致，读取过期数据，全量复制主库压力（优化为增量复制）  2.9.2 Redis 哨兵  2.9.2.1 哨兵作用  2.9.2.2 哨兵模式  2.9.2.3 哨兵如何判定主库下线  2.9.2.4 哨兵的工作模式  2.9.2.5 哨兵是如何选主的？  2.9.2.6 由哪个哨兵执行主从切换呢？  2.9.2.7 故障转移 slave顶替下线的leader  2.9.3 Redis Cluster 集群  2.9.3.1 哈希槽 ( Hash Slot )  2.9.3.2 MOVED 重定向和 ASK 重定向  2.9.3.3 Cluster 集群节点的通讯协议 ：Gossip  2.9.3.4 故障转移  2.9.3.5 加餐 ：为什么 Redis Cluster 的 Hash Slot 是 16384 ？ |

###### 2.10 使用过 Redis 分布式锁嘛？有哪些注意点呢？

|  |
| --- |
| 2.10.1 命令 setnx + expire 分开写  **if** ( jedis.setnx(key,lock\_value)==1 ) {//加锁  expire ( key ，100 ) ; //设置过期时间  try {  **do** something //业务请求  (){  }**catch**  }  finally  jedis.del(key); //释放锁  }  }  未过期宕机可能出现长生不老锁 |
| 2.10.2 setnx + value 值是过期时间  longexpires=System.currentTimeMillis()+expireTime;//系统时间+设置的过期时间  StringexpiresStr=String.valueOf(expires);  //如果当前锁不存在， 返回加锁成功  **if**(jedis.setnx(key,expiresStr)==1){  return **true**;  }  //如果锁已经存在， 获取锁的过期时间  StringcurrentValueStr=jedis.get(key);  //如果获取到的过期时间，小于系统当前时间，表示已经过期  **if**(currentValueStr !=null&&Long.parseLong(currentValueStr)<System.currentTime  Millis()){  // 锁已过期 ，获取上一个锁的过期时间，并设置现在锁的过期时间(不了解 redis 的 getSet 命  令的小伙伴 ，可以去官网看下哈 )  StringoldValueStr=jedis.getSet(key\_resource\_id,expiresStr);  **if**(oldValueStr !=null&&oldValueStr.equals(currentValueStr))  {  // 考虑多线程并发的情况， 只有一个线程的设置值和当前值相同，它才可以加锁  return **true**;  }  }  //  其他情况， 均返回加锁失  败  return**false**;  }  1.各个机器时间要同步 2.可能被其他客户端误解开 3.释放锁之后，多个锁覆盖下次的过期时间 |
| 2.10.3 ： set 的扩展命令( set ex px nx )(注意可能存在的问题)  longexpires=System.currentTimeMillis()+expireTime;//系统时间+设置的过期时间  StringexpiresStr=String.valueOf(expires);  //如果当前锁不存在， 返回加锁成功  if(jedis.setnx(key,expiresStr)==1){  //如果锁已经存在， 获取锁的过期时间  String currentValueStr = jedis.get(key);  //如果获取到的过期时间，小于系统当前时间，表示已经过期  if(currentValueStr !=null&&Long.parseLong(currentValueStr)<System.currentTimeMillis()){  // 锁已过期 ，获取上一个锁的过期时间，并设置现在锁的过期时间(不了解 redis 的 getSet 命  令的小伙伴 ，可以去官网看下哈 )  String oldValueStr = jedis.getSet(key\_resource\_id,expiresStr);  if(oldValueStr !=null&&oldValueStr.equals(currentValueStr))  // 考虑多线程并发的情况， 只有一个线程的设置值和当前值相同，它才可以加锁  Return true;  }  //  Return false;  }  1.锁被误删除 2.锁到期但是任务还未执行完。 |
| 2.10.4 set ex px nx + 校验唯一随机值,再删除  **if** ( jedis.set(key,uni\_request\_id,"NX","EX",100s)==1 ) {//加锁  try {  **do** something //业务处理  (){  }**catch**  }  finally  jedis.del(key); //释放锁  }  }  }  Jdis.del()不是原子操作，可能被误删。 |

###### 2.11 使用过 Redisson 嘛 ？说说它的原理

|  |
| --- |
| redisson解决了锁过期释放但是业务没有执行完的问题。用watch dog定时查看锁还在则延长。 |

###### 2.12 什么是 Redlock 算法

|  |
| --- |
| 如果线程一在 Redis 的 master 节点上拿到了锁， 但是加锁的 key 还没同步到slave 节点。 恰好这时， master 节点发生故障 ，一个 slave 节点就会升级为master 节点。 线程二就可以获取同个 key 的锁啦 ，但线程一也已经拿到锁了 ，锁的安全性就没了。 |
| 1按顺序向 5 个 master 节点请求加锁  2根据设置的超时时间来判断 ，是不是要跳过该 master 节点。  3如果大于等于三个节点加锁成功 ，并且使用的时间小于锁的有效期 ，即可认定加锁 成功啦。 |

###### 2.13 Redis 的跳跃表

|  |
| --- |
| Zset的底层实现之一，平均复杂的O（logn），最欢O（n） |

###### 2.14 MySQL 与 Redis 如何保证双写一致性

|  |
| --- |
| 2.14.1 延时双删？  2.14.2 删除缓存重试机制  2.14.3 读取 biglog 异步删除缓存:  . 可以使用阿里的 canal 将 binlog 日志采集发送到 MQ 队列里面  . 然后通过 ACK 机制确认处理这条更新消息 ，删除缓存， 保证数据缓存一致性 |

###### 2.15 为什么 Redis 6.0 之后改多线程呢？

|  |
| --- |
| Redis几乎不会CPU瓶颈，主要是IO和网路瓶颈，因此redis使用多线程处理数据的读写和解析从而提高IO效率，指令还是单线程来处理。 |

###### 2.16 聊聊 Redis事务机制

|  |
| --- |
| . 开始事务( MULTI )  . 命令入队  . 执行事务( EXEC )、 撤销事务( DISCARD )  . WTTCH：检视key |

###### 2.17 Redis的Hash 冲突怎么办

|  |
| --- |
| 链式hash，  为了检视hash，Redis做了rehash处理，使用了两个hash表一主一备。 |

###### 2.18 在生成 RDB 期间， Redis 可以同时处理写请求么？

|  |
| --- |
| Save指令会阻塞  Bgsave则是fork一个子线程来处理。 |

###### 2.19 Redis底层，使用的什么协议?

|  |
| --- |
| RESP协议， 主要有实现简单、解析速度快、可读性好等优点。 |

###### 2.20 布隆过滤器

|  |
| --- |
| 一个很长的二进制向量和一组hash函数组成，检查元素是否在集合中。检查有则大概率有，检查没有则一定没有。 |

##### MySQL 面试题

###### 1、MySQL 中有哪几种锁？

|  |
| --- |
| 行锁  表锁  页面锁  悲观锁 乐观锁区间锁 |

###### 2、MySQL 中有哪些不同的表格？

|  |
| --- |
| InnoDb MyISAM |

###### 3、简述在 MySQL 数据库中 MyISAM 和 InnoDB 的区别

|  |
| --- |
| MyISAM：不支持事务，表锁，三个文件。  InnoDb：四种级别 |

###### 4、MySQL 中 InnoDB 支持的四种事务隔离级别名称 ，以及逐级之间的区别？

|  |
| --- |
| 1 读未提交 2. 不可重复度 3.可重复读 4.串行事务 |

###### 5、CHAR 和 VARCHAR 的区别？

|  |
| --- |
| CHAR：固定长度 VARCHAR：可变长度 |

###### 6、主键和候选键有什么区别？

|  |
| --- |
| 主键：唯一；候选键：多个不唯一 |

###### 7、myisamchk 是用来做什么的？

|  |
| --- |
| 它用来压缩 MyISAM表， 这减少了磁盘或内存使用 |

###### 8、如果一个表有一列定义为TIMESTAMP，将发生什么？

|  |
| --- |
| 时间戳。 |

###### 9、你怎么看到为表格定义的所有索引？

|  |
| --- |
| SHOW INDEX FROM <tablename>; |

###### 10、LIKE 声明中的％和\_是什么意思？

|  |
| --- |
| %：0个或者多个；\_：1个； |

###### 11、列对比运算符是什么？

|  |
| --- |
| 使 用 = ，<> ，<= ，< ，> = ，> ，<< ，>> ，<=>，AND，OR 或 LIKE运算符 |

###### 12、BLOB 和 TEXT有什么区别？

|  |
| --- |
| BLOB是一个二进制对象 |

###### 13、MySQL\_fetch\_array和MySQL\_fetch\_object的区别是什么？

|  |
| --- |
| 返回数据格式不一样； |

###### 14、MyISAM表格将在哪里存储，并且还提供其存储格式？

|  |
| --- |
| .frm .MYD .MYI |

###### 15、MySQL如何优化DISTINCT？

|  |
| --- |
| DISTINCT 在所有列上转换为GROUP BY，并与ORDER BY子句结合使用 。  SELECT DISTINCT t1 .a FROM t1,t2 where t1 .a=t2 .a; |

###### 16、如何显示前 50 行？

|  |
| --- |
| SELECT \* FROM LIMIT 0,50; |

###### 17、可以使用多少列创建索引？

|  |
| --- |
| 16 |

###### 18、NOW()和CURRENT\_DATE()有什么区别？

|  |
| --- |
| 秒和日的差别； |

###### 19、什么是非标准字符串类型？

|  |
| --- |
| 1 、TINYTEX  2 、 TEXT  3 、MEDIUMTEXT  4 、 LONGTEXT |

###### 20、什么是通用SQL函数？

|  |
| --- |
| CONCAT NOW HOUR |

###### 21、MySQL 支持事务吗？

|  |
| --- |
| autocommit 模式。 |

###### 22、MySQL 里记录货币用什么字段类型好

|  |
| --- |
| salary DECIMAL(9,2)； |

###### 23、MySQL 有关权限的表都有哪几个？

|  |
| --- |
| MySQL\_ install\_db |

###### 24、列的字符串类型可以是什么？

|  |
| --- |
| 1 、 SET  2 、 BLOB  3 、 ENUM  4 、 CHAR  5 、 TEXT |

###### 25、MySQL数据库作发布系统的存储，一天五万条以上的增量，预计运维三年,怎么优化？

|  |
| --- |
| 分库分表 |

###### 26、锁的优化策略

|  |
| --- |
| 1读写分离 2.分段加锁 3.减少锁时间 4.顺序获取资源 |

###### 27、索引的底层实现原理和优化

|  |
| --- |
| B+树 |

###### 28、什么情况下设置了索引但无法使用

|  |
| --- |
| 1.% 2. or 3.数据类型’’; 4.数据函数 5.联合索引 |

###### 29、实践中如何优化 MySQL

|  |
| --- |
| 1.SQL语句编写  2.系统设计  3.硬件升级 |

###### 30、优化数据库的方法

|  |
| --- |
| 1、选取 最适 用的 字段 属性 ，尽可 能减 少定 义字 段宽 度 ，尽量 把字 段设 置 NOTNULL， 例如 ’省份 ’ 、 ’性别 ’最好适用ENUM  2、使用连接 (JOIN)来代替子查询  3、适用联合 (UNION)来代替手动创建的临时表  4、事务处理  5、锁定表 、优化事 务处理  6、适用外键，优化锁定表  7、建立索引  8、优化查询语句 |

###### 32、数据库中的事务是什么?

|  |
| --- |
| 1原子性 2一致性 3隔离性 4.持久性 |

###### 33、SQL 注入漏洞产生的原因？如何防止？

|  |
| --- |
| 1.特殊字符过滤 2.预编译 |

###### 34、为表中得字段选择合适得数据类型

|  |
| --- |
| 字段 类型 优先 级: 整形 >date,time>enum,char>varchar>blob,text |

###### 35、存储时期

|  |
| --- |
| Datatime:以 YYYY - MM - DD HH:MM:SS 格式 存储 时期 时间 |

###### 36、对于关系型数据库而言，索引是相当重要的概念，请回答

|  |
| --- |
| 1.目的  2.负面影响  3.索引原则  4.什么情况不宜建立索引 |

###### 37、解释 MySQL 外连接、内连接与自连接的区别

|  |
| --- |
| 外连接：交叉连接-笛卡尔积  内连接：  自连接； |

###### 38、Myql中的事务回滚机制概述

|  |
| --- |
| Redo和undo日志 |

###### 39、SQL 语言包括哪几部分？每部分都有哪些操作关键字？

|  |
| --- |
| 数据 定义 ： Create Table,Alter Table,Drop Table, Craete/Drop Index 等  数据 操纵 ： Select ,insert,update,delete,  数据 控制 ： grant,revoke  数据 查询 ： select |

###### 40、完整性约束包括哪些？

|  |
| --- |
| 实体完整性  域完整性  参照完整性  用户定义完整性 |

###### 41、什么是锁？

|  |
| --- |
| 加 锁 是 实 现 数 据 库 并 发 控 制 的 一 个 非 常 重 要 的 技 术 。 |

###### 42、什么叫视图？游标是什么？

|  |
| --- |
| 视图：虚拟表；游标：对查询出来的结果作为一个单元来有效处理 |

###### 43、什么是存储过程？用什么来调用？

|  |
| --- |
| 简单来说就是一段SQL的封装，到时候直接调用 |

###### 44、如何通俗地理解三个范式？

|  |
| --- |
| 1.原子不能切割 2.其他完成的依赖主键 3. 其他信息直接依赖主键 |

###### 45、什么是基本表？什么是视图？

|  |
| --- |
| 基本 表是 本身 独立 存在 的表 ，在 SQL 中一 个关 系就 对应 一个 表  视图：虚拟表 |

###### 46、试述视图的优点？

|  |
| --- |
| 1.简洁 2.多角度看同一事务 3.逻辑独立性 4.数据保护 |

###### 47、NULL是什么意思

|  |
| --- |
| 1.不同于“”，空字符串，就是空，用 IS NULL来判断 |

###### 48、主键、外键和索引的区别？

|  |
| --- |
| 1.主键唯一 2.外键是索引，可重复可多个 3.索引可重复，可多个 |

###### 49、你可以用什么来确保表格里的字段只接受特定范围里的值?

|  |
| --- |
| Check限制， |

###### 50、说说对 SQL 语句优化有哪些方法？(选择几条)

|  |
| --- |
| 1.where表之间的链接放其他where之前  2. EXISTS 代替IN 3.避免在索引上进行计算  4.避免全表扫描，首先考虑where和order by 5.避免在where子语句上做null判断  在回表次数，索引等方面进行优化 |

##### MySql附加题

###### 1.1 MySQL 索引使用有哪些注意事项呢？

* + 1. 索引哪些情况会失效

|  |
| --- |
| 1.or 2.字符串和where 3.like 4.联合索引 5.索引列内置函数 6.判断符7.is nulll 8.格式不一 9.全表扫描快 |

* + 1. 索引不适合哪些场景

|  |
| --- |
| 1.数量少 2.变动频繁 3.区分度低 |

* + 1. 索引的一些潜规则

|  |
| --- |
| 1.索引覆盖 2.回表 3.b+树 4.最左前缀原则 5.索引下推（顺带check一下后面的条件） |

###### 1.2. MySQL 遇到过死锁问题吗，你是如何解决的？

|  |
| --- |
| 1.查看死锁日志 show engine innodb status；  2.找出死锁sql 3.分析加锁情况 4.模拟死锁案发 5.分析死锁日志 6.分析死锁结果 |

###### 1.3. 日常工作中你是怎么优化 SQL 的？

|  |
| --- |
| 1.加索引 2.避免不必要数据 3.适当分批操作 4.优化sql结构 5.分库分表 6.读写分离 |

###### 1.4. 说说分库与分表的设计

|  |
| --- |
| 1.水平分库 2.水平分表 3.垂直分库 4.垂直分表 |
| Sharding-jdbc Mychart |
| 1.事务问题 2.跨节点join问题 3.跨节点count，order by，grout by问题 4.数据迁移 5.ID问题 6.跨分片排序 |

###### 1.5. InnoDB 与 MyISAM 的区别

|  |
| --- |
| 1.事务 2.外键 3.MVCC 4.select count(\*) 5.全文索引 6.表，行锁 7.主键 8.插入顺序 9.存储空间 |

###### 1.6. 数据库索引的原理，为什么要用 B+树，为什么不用二叉

|  |
| --- |
| 1.查询快 2.深度浅 3.排列有序前后关联 |

###### 1.7. 聚集索引与非聚集索引的区别

|  |
| --- |
| 1.一个表只有一个聚簇 2.聚簇叶子是数据 3.聚簇索引的物理存储按照索引排序 |

###### 1.8 limit 1000000 加载很慢的话，你是怎么解决的呢

|  |
| --- |
| 方案一：如果 id 是连续的，可以这样 ，返回上次查询的最大记录(偏移量），在往下limit  selectid ，namefromemployeewhereid>1000000limit10.  方案二：在业务允许的情况下限制页数：  方案三： order by + 索引( id 为索引)  selectid ，namefromemployeeorderbyidlimit1000000 ，10  方案四： 利用延迟关联或者子查询优化超多分页场景。( 先快速定位需要获取的ID段，然后再关联）  SELECTa.\*FROMemployeea,(selectidfromemployeewhere条件LIMIT1000000,10)b  wherea.id=b.id |

###### 1.9 如何选择合适的分布式主键方案呢？

|  |
| --- |
| 1.单独数据库自增 2.UUID 3.snowflake |

###### 1.10 事务的隔离级别有哪些？ MySQL 的默认隔离级别是什

|  |
| --- |
| 1.读未提交 2.读已提交 3.可重复读 4.串行化 |

###### 1.11 什么是幻读，脏读，不可重复读呢？

|  |
| --- |
| 参考1.10 |

###### 1.12 在高并发情况下，如何做到安全的修改同一行数据？ 181

1.12.1 使用悲观锁

|  |
| --- |
| select\*fromUserwherename=‘jay’**for**update |

1.12.2 使用乐观锁

|  |
| --- |
| Cas |

###### 1.13 数据库的乐观锁和悲观锁。

1.13.1 悲观锁：

1.13.2 乐观锁：

|  |
| --- |
|  |

###### 1.14 SQL 优化的一般步骤是什么，怎么看执行计划 ( explain )

|  |
| --- |
| show status 命令了解各种sql 的执行频率  . 通过慢查询日志定位那些执行效率较低的sql 语句  . explain 分析低效 sql 的执行计划 (这点非常重要，日常开发中用它分析Sql ，会大大降低 Sql 导致的线上事故 ) |

###### 1.15. select for update 有什么含义，会锁表还是锁行还

1.15.1 select for update 含义

1.15.2 select for update 加锁验证

|  |
| --- |
| 加的悲观锁 |

###### 1.16 MySQL 事务得四大特性以及实现原理

|  |
| --- |
| 原子性 隔离性 持久性 一致性 |

###### 1.17如果某个表有近千万数据，CRUD比较慢，如何优化。

|  |
| --- |
| 1.分库分表 2.SQL优化 3.索引优化 |

###### 1.18如何写 sql 能够有效的使用到复合索引。

|  |
| --- |
| 三个索引 最左侧原则 |

###### 1.19 mysql 中 in 和 exists 的区别。

|  |
| --- |
| select\*fromAwheredeptId**in**(selectdeptIdfromB);  select\*fromAwhereexists(select1fromBwhereA.deptId=B.deptId);  若A数据量大于B则用exit，否则用in |

###### 1.20 数据库自增主键可能遇到什么问题。

|  |
| --- |
| 分库分表可能无法全局唯一 |

###### 1.21 MVCC 熟悉吗，它的底层原理？

|  |
| --- |
| 1.事务版本号 2.表的隐藏列 3.undo log 4.read view |

###### 1.22 数据库中间件了解过吗， sharding jdbc ，mycat ？

|  |
| --- |
| 1.sharding-jdbc基于jdbc驱动，无需额外proxy 2.Mycat基于proxy，复写MySql协议，伪装成一个数据库 |

###### 1.23 MYSQL 的主从延迟，你怎么解决？ 188

1.23.1 主从同步延迟的原因

|  |
| --- |
| Binlog网路耗时，积压则会出现同步延时 |

1.23.2 主从同步延迟的解决办法

|  |
| --- |
| 1.安全性要求高则sync\_binlog=1 2.slave设备性能高 3.从节点只做备份 4增加从服务器分散读压力 |

###### 1.24 说一下大表查询的优化方案

|  |
| --- |
| 1.优化sql语句+索引 2.加redis缓存和jvm本地缓存 3.主从复制，读写分离 4.分库分表 |

###### 1.25 什么是数据库连接池?为什么需要数据库连接池呢?

|  |
| --- |
| 1.资源重用 2.更快响应 3.资源分配 4.统一管理避免泄露 |

###### 1.26 一条 SQL 语句在 MySQL 中如何执行的？

|  |
| --- |
| 1.查权限 2.查缓存，么有就三，3语法分析 4.优化执行方案 5.权限校验和调用引擎执行并返回 |

###### 1.27 InnoDB 引擎中的索引策略，了解过吗？

|  |
| --- |
| 1.覆盖索引 2.索引下推 3.最左前缀原则 |

###### 1.28 一条 sql 执行过长的时间，你如何优化，从哪些方面入

|  |
| --- |
| 1.有子查询，冗余处理 2.索引优化 3.单表数量 4.主从分离，读写分离 5.explain 分析 6.查看mysql日志 |

###### 1.29 MYSQL数据库服务器性能分析的方法命令有哪些?

|  |
| --- |
| 1.Show status 2.show profile |

###### 1.30 Blob和text有什么区别？

|  |
| --- |
| 1.blob存储二进制数据，text存储大字符串 2.blob没有字符集，排序基于字节数，text基于字符集排序 |

###### 1.31 mysql 里记录货币用什么字段类型比较好？

|  |
| --- |
| salary DECIMAL(9,2) |

###### 1.32 Mysql 中有哪几种锁，列举一下？

|  |
| --- |
| 1.加锁机制：乐观锁，悲观锁 2.锁粒度：表锁，页锁，行锁 3.兼容性：共享锁，排它锁  4.锁模式：记录锁，gap锁，意向锁，next-key锁，插入意向锁 |

###### 1.33 Hash 索引和 B+树区别是什么？你在设计索引是怎么抉

|  |
| --- |
| 1.范围查找 2.联合索引和最左侧原则 3.order by排序 4.等值查询 5模糊查询 |

###### 1.34 mysql 的内连接、左连接、右连接有什么区别？

|  |
| --- |
| 1. Inner join 内连接，只保留两张表完全匹配的结果 2.左右连接，以左（右）为主 |

###### 1.35 说说 MySQL 的基础架构图

|  |
| --- |
| 1.客户端查缓存 2.有缓存则直接返回，无则分析-->优化-->执行 |

###### 1.36 什么是内连接、外连接、交叉连接、笛卡尔积呢？

|  |
| --- |
| 见上 |

###### 1.37 说一下数据库的三大范式

|  |
| --- |
| 1.每一列不可拆 2.其他列完全依赖主键 3. 非主键完全直接依赖主键 |

###### 1.38 Mysql 的binlog 有几种录入格式？分别有什么区别？

|  |
| --- |
| 1.statement 2.row 3.mixed（一致性较好） |

###### 1.39 InnoDB 引擎的 4 大特性，了解过吗 ？

|  |
| --- |
| 1.插入缓冲 2.二次写 3.自适应哈希索引 4.预读 |

###### 1.40 索引有哪些优缺点？索引有哪几种类型？

1.40.1 索引有哪些优缺点？

|  |
| --- |
| 优点：1.唯一索引保证 2.减少查询时间  缺点：1.耗时 2.耗空间 3.增删改需要维护费时 |

1.40.2 索引有哪几种类型？

|  |
| --- |
| 1.主键索引 2.唯一索引 3.普通索引 4.全文索引 5.覆盖索引 6.组合索引 |

###### 1.41 创建索引有什么原则呢？

|  |
| --- |
| 1.最左前缀匹配 2.频繁查询的列才索引 3.频繁变更的列不做索引 4.索引不能参与计算、函数操作 5.优先考虑扩展索引而不是新建索引 6.order by 或者group by的剧中中主要索引顺序 7.定义外键的数据列要建立索引。 8.text，images不要建立索引。 9.删除不用或者少用的索引。 |

###### 1.42 创建索引的三种方式

|  |
| --- |
| CREATETABLE`employee`(  `id`int(11)NOTNULL,  `name`varchar(255)DEFAULTNULL,  `age`int(11)DEFAULTNULL,  `date`datetimeDEFAULTNULL,  `sex`int(1)DEFAULTNULL,  PRIMARY KEY (`id`),  KEY`idx\_name`(`name`)USINGBTREE  )ENGINE=InnoDBDEFAULTCHARSET=utf8;  ALTERTABLEtable\_nameADDINDEXindex\_name(column);  CREATEINDEXindex\_nameONtable\_name(column); |

###### 1.43 百万级别或以上的数据，你是如何删除的？

|  |
| --- |
| 1.先删除索引 2.批量删除无用数据 3.重新 建立索引 |

###### 1.44 什么是最左前缀原则？什么是最左匹配原则？

|  |
| --- |
| where 子句中使用最频繁的一列放在最左边。 |

###### 1.45 B 树和 B+树的区别，数据库为什么使用 B+树而不是

|  |
| --- |
| 1.叶子节点区别 2.B+树单独有条链相连 |

###### 1.46 覆盖索引、回表等这些，了解过吗？

|  |
| --- |
| 1.覆盖索引 2.回表 |

###### 1.47 非聚簇索引一定会回表查询吗？

|  |
| --- |
| 有可能是覆盖索引 |

###### 1.48 联合索引是什么？为什么需要注意联合索引中的顺序？

|  |
| --- |
|  |

###### 1.49 数据库隔离级别与锁的关系

|  |
| --- |
| RC每次读取数据前都生成一个ReadView，而RR只在第一次读取数据时生成一个ReadView。  RC隔离级别下，加锁的select, update, delete等语句，使用的是记录锁，其他事务的插入依然可以执行，因此会存在幻读~  RR隔离级别下，加锁的select, update, delete等语句，会使用间隙锁+ 临键锁，锁住索引记录之间的范围，避免范围间插入记录，以避免产生幻影行记录。 |

###### 1.50 聊聊 InnoDb 的锁，什么是死锁？怎么解决？

|  |
| --- |
| 死锁是指两个或多个事务在同一资源上相互占用，并请求锁定对方的资源 ，从 而导致恶性循环的现象。看图形象一点，如下：  预防：1.并发存取时，按照约定的相同顺序访问 2.一次拿到所有资源 3.升级锁颗粒度 4.分布式锁或者乐观苏 5.合理优化索引 |

###### 1.50.1 聊聊 InnoDb 的锁 199

|  |
| --- |
| 1.50.1.1 共享/排他锁  1.50.1.2 意向锁  1.50.1.3 记录锁( Record Lock )  1.50.1.4 间隙锁( Gap Lock )  1.50.1.5 临键锁(Next-Key Lock)  1.50.1.6 插入意向锁  1.50.1.7 自增锁  1.50.2 什么是死锁？怎么解决？ |

###### 1.51 count(1)、count(\*) 与 count(列名) 的区别？

|  |
| --- |
| 1.count（1）忽略所有列 2.count（\*）包括所有列，包括null 3.count(列名)忽略null |

###### 1.52 什么是存储过程？有哪些优缺点？ 207

|  |
| --- |
| 优点：1.预编译代码块，执行效率高 2.存储在服务端，客户端减压 3.复用 4.数据量传输小 5.数据安全 |
| 1缺点：调试麻烦 2.可移植性差 3.重新编译问题 |

###### 1.53 SQL 约束有哪几种呢？

|  |
| --- |
| 1.NOT NULL 2.UNIQUE 3.PRIMARY KEY 4.FOREING KEY 5.CHECK |

###### 1.54 varchar(50)中 50 的涵义

|  |
| --- |
| 可变长度 |

###### 1.55 mysql 中 int(20)和 char(20)以及 varchar(20)的

|  |
| --- |
| Int 显示长度为20 char：固定长度20； varchar：可变长度20 |

###### 1.56 drop、delete 与 truncate 的区别

|  |
| --- |
| Delete < truncate <drop |

###### 1.57 UNION 与 UNION ALL 的区别？

|  |
| --- |
| Union对两个结果集进行并集操作 union All 是所有进行并集。 |

###### 1.58 SQL 的生命周期？

|  |
| --- |
| 1.建立连接 2.拿到SQL 3.解析并生成执行计划4.读取数据到内存并进行逻辑处理 5.发送到客户端 6.关闭 |

###### 1.59 一条 Sql 的执行顺序？

|  |
| --- |
| 1.from<左表的名字> 2.on<join的条件> 3.<join的类型><右表的名字> 4.where<where的条件>  5.group by<> 6.Having 7.select 8.DISTINCT 9.order by 10 limit |

###### 1.60 关心过业务系统里面的sql 耗时吗？统计过慢查询吗？

|  |
| --- |
| . 分析语句，是否加载了不必要的字段/数据。  . 分析 SQl 执行句话，是否命中索引等。  . 如果 SQL 很复杂 ，优化 SQL 结构  . 如果表数据量太大， 考虑分表 |

###### 1.61 主键使用自增 ID 还是 UUID ，为什么？ mysql 自增主

|  |
| --- |
| 雪花算法snowID |

###### 1.62 MySQL 数据库 cpu 飙升的话，要怎么处理呢？

|  |
| --- |
| 1.top指令看是否是mysqld导致的 2.是则show processlist查看是否有消耗资源的sql在运行 3.找出高消耗 |
| Kill掉这些线程 进行相应的调整 重新跑这些SQL |

###### 1.63 读写分离常见方案？

|  |
| --- |
| 根据业务逻辑来判断 |

###### 1.64 MySQL 的复制原理以及流程

|  |
| --- |
| 1.主库更新事件写入binlog 2.从库链接到主库 3.binlog发送到从库4.从库创建IO线程读取binlog到relay log  5.从库创建SQL线程读取relay log并执行 6.这期间的更新写入buffer并传给从库执行 |

###### 1.65 Innodb 的事务实现原理？

|  |
| --- |
| 1.原子性：undo log来实现，配合rullback 2.持久性：redo log来实现，系统崩溃靠他恢复 3.隔离性MVCC 4.一致性：通过回滚，恢复以及并发隔离来实现。 |

###### 1.66 谈谈 MySQL 的 Explain

|  |
| --- |
| system>const>eq\_ref>ref>ref\_or\_null>index\_merge>unique\_subquery>index\_subquery>range>index>ALL |

###### 1.67 Innodb 的事务与日志的实现方式

1.67.1 有多少种日志

|  |
| --- |
| Redo 和undo |

1.67.2 日志的存放形式

|  |
| --- |
| Redo：先写到buffer 然写到redo log文件系统缓存里，最后一步刷盘  Undo:放ibdata之外 |

1.67.3 事务是如何通过日志来实现的

|  |
| --- |
| 修改页面是先记undo，undo之前先记undo的redo，然后修改数据，在记修改的redo  若要回滚，有undo，可以回滚到之前的状态  若有commit记录，用redo前滚到改事务完成时 |

**1.68 你们数据库是否支持emoji 表情存储，如果不支持 ，如何操作？**

|  |
| --- |
| 更换字符集 utf8-- >utf8mb4 |

###### 1.69 一个 6 亿的表 a ，一个 3 亿的表b ，通过外间tid 关联 ， 213

|  |
| --- |
| 1.若id连续则select \* from a,b where a.tid = b.id and a.tid>50000 limit 200;  2.若A表Tid不连续，使用覆盖索引，TID要么是主键，要么是辅助索引，B表id也需要有索引  Select \* from b, (select tid from a limit 50000,200)a where b.id = a.tid |

###### 1.70 Mysql 一条 SQL 加锁分析

##### Java 并发编程(一)

###### 1、在 java 中守护线程和本地线程区别？ 214

|  |
| --- |
| 唯一的区 别是判断虚拟机(JVM)何时离开 |

###### 2、线程与进程的区别？

|  |
| --- |
| 进程：操作系统分配资源的最小单元 线程：操作系统调度的最小单元 |

###### 3、什么是多线程中的上下文切换？

|  |
| --- |
| 让cpu轮流执行各个线程 |

###### 4、死锁与活锁的区别，死锁与饥饿的区别？

|  |
| --- |
| 死锁：各自持有资源，各自盼着别人的资源，而且不会主动放手资源也不能强行夺取资源  饥饿 ： 一 个或 者多 个线 程因 为种 种原 因无 法获 得所 需要 的资 源， 导致 一直 无法 执 行的 状态 。 |

###### 5、Java 中用到的线程调度算法是什么？

|  |
| --- |
| 时间片轮转的方式，可以设置优先级 |

###### 6、什么是线程组，为什么在 Java 中不推荐使用？

|  |
| --- |
| ThreadGroup类 ，可以把线程归属到某一个线程组中 ，不好管理，有线程池可选。 |

###### 7、为什么使用 Executor 框架？

|  |
| --- |
| 1.减少new Thread（）的性能消耗 2.便于管理 3功能扩展 定时 定期执行等。 |

###### 8、在Java中 Executor 和 Executors 的区别？

|  |
| --- |
| Executors 工 具 类 的 不 同 方 法 按 照 我 们 的 需 求 创 建 了 不 同 的 线 程 池 ， 来 满 足 业 务 的 需 求 。  Executor 接口对象能执行我们的线程任务 。 |

###### 9、如何在 Windows 和 Linux 上查找哪个线程使用的 CPU时间最长？

|  |
| --- |
| Top |

###### 10、什么是原子操作？在 Java Concurrency API 中有哪些原子类(atomic classes)？

|  |
| --- |
| 原子类：AtomicBoolean AtomicInteger AtomicLong  原子数组：AtomicIngeterArray AtomicLongArrray AtomicReferenceArray |

###### 11、Java Concurrency API 中的 Lock 接口(Lock interface)是什么？对比同步它有什么优势？

|  |
| --- |
| Lock:提供更多个扩展性，可公平，可中断，可释放锁，可轮询，可定时 |

###### 12、什么是 Executors 框架？

|  |
| --- |
| Executor 框架是一个根 据一组执行策略调 用，调度 ，执 行和 控制 的异 步任 务的 框 架。 |

###### 13、什么是阻塞队列？阻塞队列的实现原理是什么？如何使用阻塞队列来实现生产者-消费者模型？

|  |
| --- |
| 阻塞 队列 (BlockingQueue) 是 一个 支持 两个 附加 操作 的队 列。这两 个附 加的 操作 是： 在队 列为 空时 ， 获 取元 素的 线程 会等 待队 列变 为非 空。 当 队列 满时 ， 存 储元 素的 线程 会等 待队 列可 用。 |

###### 14、什么是 Callable 和 Future?

|  |
| --- |
| Callable被线程执行之后，可以返回值，这个返回值可以被Future拿到 |

###### 15、什么是 FutureTask?使用 ExecutorService 启动任务

|  |
| --- |
| 在 Java 并 发 程 序 中 FutureTask 表 示 一 个 可 以 取 消 的 异 步 运 算 。尚未完成时get就会阻塞 |

###### 16、什么是并发容器的实现？ 219

|  |
| --- |
| 何 为 同 步 容 器 ：可 以 简 单 地 理 解 为 通 过 synchronized 来 实 现 同 步 的 容 器 ，如 果 有 多 个 线 程 调 用 同 步 容 器 的 方 法 ， 它 们 将 会 串 行 执 行 。 |

###### 17、多线程同步和互斥有几种实现方法，都是什么？

|  |
| --- |
| 同步：一个线程执行依赖另一个线程的消息  互斥：多个线程抢占一个资源 |

###### 18、什么是竞争条件？你怎样发现和解决竞争？ 219

|  |
| --- |
| 当多 个进 程都 企图 对共 享数 据进 行某 种处 理， 而最后的结果又取决于进程运行的顺序时， 则我们认为这发生了竞争条件( race condition) 。 |

###### 19、你将如何使用 thread dump？你将如何分析 Thread dump？

|  |
| --- |
| 1.新建状态（new） 2.就绪状态（Runnable） 3.运行状态（Running） 4.阻塞状态（Blocked：wait,lock,wothers）5.死亡状态（dead） |

###### 20、为什么我们调用 start()方法时会执行 run()方法，为什么我们不能直接调用 run()方法？ 227

|  |
| --- |
| Start()创建新线程，run()是普通方法 |

###### 21、Java 中你怎样唤醒一个阻塞的线程？

|  |
| --- |
| Notify() notifyAll() ,只是解除该对象阻塞线程的阻塞状态。 |

###### 22、在 Java 中 CycliBarriar 和 CountdownLatch 有什么区别？ 227

|  |
| --- |
| CycliBarriar 可清零重复使用。CountdownLatch 一次性使用。 |

###### 23、什么是不可变对象，它对写并发应用有什么帮助？

|  |
| --- |
| 不可改变，线程安全。 |

###### 24、什么是多线程中的上下文切换？

|  |
| --- |
| CPU为各个线程分配执行时间。 |

###### 25、Java 中用到的线程调度算法是什么？

|  |
| --- |
| 线程调度是指按照特定机制为多个线程分配CPU的使--用权 .分时调度模型和抢占式调度模型 |

###### 26、什么是线程组，为什么在 Java 中不推荐使用？

|  |
| --- |
| 有能管理线程的线程池。 |

###### 27、为什么使用 Executor 框架比使用应用创建和管理线程好？

|  |
| --- |
| 1.降低开销 2.有效控制线程数量 3.功能增强 |

###### 28、java 中有几种方法可以实现一个线程？

|  |
| --- |
| 1.继承Thread 2.实现Runnable接口 3.实现Callable接口，需要实现call()方法。 |

###### 29、如何停止一个正在运行的线程？

|  |
| --- |
| 如处于阻塞中的线程要停止，可以使用Thread提供的interupt()方法，可抛出异常并退出阻塞 |

###### 30、notify()和 notifyAll()有什么区别？

|  |
| --- |
| 唤醒一个和多个 |

###### 31、什么是 Daemon 线程？它有什么意义？

|  |
| --- |
| 后台进行的守护线程 |

###### 32、java 如何实现多线程之间的通讯和协作？

|  |
| --- |
| 中断和共享变量 |

###### 33、什么是可重入锁(ReentrantLock)？

|  |
| --- |
| 1. synchronized 、 ReentrantLock 都是可重入的锁，可重入锁相对来说简化了并发编程的开发 |

###### 34、当一个线程进入某个对象的一个 synchronized的实例方法后，其它线程是否可进入此对象的其它方法？

|  |
| --- |
| Synchronized修饰对象的方法，则该方法被锁，不影响其他方法。 |

###### 35、乐观锁和悲观锁的理解及如何实现，有哪些实现方式？

|  |
| --- |
| 悲观：最坏思路； 乐观锁：CAS（ABA问题，循环开销，只能保证一个变量是原子操作） |

###### 36、SynchronizedMap 和 ConcurrentHashMap 有什么区别？

|  |
| --- |
| SynchronizedMap 一次一个线程，而ConcurrentHashMap 有16个hash桶分段加锁 |

###### 37、CopyOnWriteArrayList 可以用于什么应用场景？

|  |
| --- |
| CopyOnWriteArrayList读写分离，最终一致性，使用另外开辟空间来解决冲突。 |

###### 38、什么叫线程安全？ servlet 是线程安全吗?

|  |
| --- |
| 线程安全就是多个能够正确处理和共享变量。  Servlet为单实例多线程，不能保证线程安全。 |

###### 39、volatile 有什么用？能否用一句话说明下 volatile 的应用场景？

|  |
| --- |
| 1.保证了可见性和指令重排  重 排 序 不 会 影 响 单 线 程 环 境 的 执 行 结 果 ， 但 是 会 破 坏 多 线 程 的 执 行 语 义 。 |

###### 40、为什么代码会重排序？

|  |
| --- |
| 提高执行效率 |

###### 41、在 java 中 wait 和 sleep 方法的不同？

|  |
| --- |
| Wait会让出锁，而sleep不会。 |

###### 43、一个线程运行时发生异常会怎样？

|  |
| --- |
| Thread .getUncaughtExceptionHandler() |

###### 44、如何在两个线程间共享数据？

|  |
| --- |
| 两个线程间的共享变量。 |

###### 45、Java 中 notify 和 notifyAll 有什么区别？

|  |
| --- |
| 唤醒一个和多个 |

###### 46、为什么 wait, notify 和 notifyAll 这些方法不在 thread类里面？

|  |
| --- |
| 应为想要这些操作都需要获取当前对象的锁 |

###### 47、什么是 ThreadLocal 变量？

|  |
| --- |
| 每个线程创立的一个私有变量，消除竞争条件 |

###### 48、Java 中 interrupted 和 isInterrupted 方法的区别？

|  |
| --- |
| interrupt将线程标记为中断状态，后面某个时间将会被处理  Isinterrupt查看线程状态，如果中断了第一次返回true，第二次返回false。 |

###### 49、为什么 wait 和 notify 方法要在同步块中调用？

|  |
| --- |
| Java api强制要求的，否则会报错。也是为了避免wait和notify之间产生竞争条件 |

###### 50、为什么你应该在循环中检查等待条件?

|  |
| --- |
| 防止错误警报和伪唤醒，可以让线程hang一会避免马上应为条件不允许而退出。 |

###### 51、Java 中的同步集合与并发集合有什么区别？

|  |
| --- |
| 不 过并 发 集合 的可 扩展 性更 高 |

###### 52、什么是线程池？ 为什么要使用它？

|  |
| --- |
| 1.避免开销 2.便于控制数量 3.功能增强 |

###### 53、怎么检测一个线程是否拥有锁？

|  |
| --- |
| java . lang .Thread 中有 一个 方法 叫 holdsLock() |

###### 54、你如何在 Java 中获取线程堆栈？

|  |
| --- |
| kill -3 [java pid] |

**55 、 JVM中哪个参数是用来控制线程的栈堆栈小的 ?**

|  |
| --- |
| -Xss |

###### 56、Thread 类中的 yield 方法有什么作用？

|  |
| --- |
| 当前线程从执行状态变为可运行状态 |

###### 57、Java 中 ConcurrentHashMap 的并发度是什么？

|  |
| --- |
| 默认值16  在 JDK8 后 ， 它 摒 弃 了 Segment (锁 段 ) 的 概 念 ， 而 是 启 用 了 一 种 全 新 的 方 式 实 现 ,利 用 CAS 算 法 。 同 时 加 入 了 更 多 的 辅 助 变 量 来 提 高 并 发 度 ，具 体 内 容 还 是 查 看 源 码 吧 。 |

###### 58、Java 中 Semaphore 是什么？

|  |
| --- |
| Java 中的 Semaphore 是一种新的同步类， 它是一个计数信号。 |

###### 59、Java 线程池中 submit() 和 execute()方法有什么区别？

|  |
| --- |
| 两个方法都可以向线程池提交任务 ，execute()方法的返回类型是 void ，它定义在 Executor 接口中。而submit()方法可以返回持有计算结果的 Future 对象，它定义在 ExecutorService 接口中，它扩展了Executor 接口其它线程池类像ThreadPoolExecutor 和 ScheduledThreadPoolExecutor都有这些方法。 |

###### 60、什么是阻塞式方法？

|  |
| --- |
| 阻塞式方法是指程序会一直等待该方法完成期间不做其他事情，ServerSocket 的 accept()方法就是一直等待客户端连接。 |

###### 61、Java 中的 ReadWriteLock 是什么？

|  |
| --- |
| 读写锁 |

###### 62.volatile 变量和 atomic 变量有什么不同？

|  |
| --- |
| Volatile：静止指令重排和线程可见性  Atomic：AtomicInteger类提供的atomic方法可以让这操作具有原子性 |

###### 63、可以直接调用 Thread 类的 run ()方法么？

|  |
| --- |
| 可以，但是不会创建新的线程 |

###### 65、你对线程优先级的理解是什么？

|  |
| --- |
| 高优先级的线程有高的优先占有权，优先级可量化 |

###### 66、什么是线程调度器(Thread Scheduler)和时间分片(Time Slicing）

|  |
| --- |
| 是一个操作系统服务，为runnable状态的线程分配CPU时间。 |

###### 67、你如何确保 main()方法所在的线程是 Java 程序最后结束的线程？

|  |
| --- |
| 使用jion方法 |

###### 68、线程之间是如何通信的？

|  |
| --- |
| 线程共享资源  Object类中的wait()\notify()\notifyAll()可以用于线程通讯关于资源锁的状态 |

###### 69、为什么线程通信的方法 wait(), notify()和 notifyAll()被定义在 Object 类里？

|  |
| --- |
| 持有锁的线程才能调用该方法， |

###### 70、为什么 wait(), notify()和 notifyAll ()必须在同步方法或者同步块中被调用？

|  |
| --- |
| 这些方法都是作用于等待对象的锁或者通知其他线程对象监视器的，而java线程又没提供使用的锁和同步器，所以就放在Object里面了。 |

###### 71、为什么Thread类的sleep()和yield ()方法是静态的？

|  |
| --- |
| 应为只有持锁线程调用sleep()和yield()才有意义，如果是实例方法，则可能出现矛盾。 |

###### 72、如何确保线程安全？

|  |
| --- |
| 同步，使用原子类，使用并发锁 |

###### 73、同步方法和同步块，哪个是更好的选择？

|  |
| --- |
| 块更好，不会锁住整个实例对象 |

###### 75、什么是 Java Timer 类？如何创建一个有特定时间间隔的任务？

|  |
| --- |
| java.util.Timer是一个工具类， 可以用于安排一个线程在未来的某个特定时间执行 。 |

##### Java 并发编程(二)

###### 1、并发编程三要素？

|  |
| --- |
| 原子性，可见性，有序性 |

###### 2、实现可见性的方法有哪些？

|  |
| --- |
| synchronized和lock可以，volitale |

###### 3、多线程的价值？

|  |
| --- |
| 1.发挥多核CPU优势 2.放在阻塞 3.便于建模 |

###### 4、创建线程的有哪些方式？

|  |
| --- |
| 1.继承Thread 2.实现Runnable接口 3.实现Callable接口 4.通过线程池 |

###### 5、创建线程的三种方式的对比？

|  |
| --- |
| 1.实现Runnable Callable接口时，还可以继承其他类，稍微复杂  2.继承Thread，简单，但是不能在基础其他了 |

###### 6、线程的状态流转图

|  |
| --- |
|  |

###### 7、Java 线程具有五中基本状态

|  |
| --- |
| 见上 |

###### 8、什么是线程池？有哪几种创建方式？

|  |
| --- |
| 创建并管理若干个线程，通过Executor接口的实现用于创建线程池。 |

###### 9、四种线程池的创建

|  |
| --- |
| 1 、newCachedThreadPool 创 建 一 个 可 缓 存 线 程 池  2 、newFixedThreadPool 创 建 一 个 定 长 线 程 池 ， 可 控 制 线 程 最 大 并 发 数 。  3 、newScheduledThreadPool 创 建 一个定长线程池 ， 支 持定 时及 周 期 性 任 务 执 行 。  4 、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 。 |

###### 10、线程池的优点？

|  |
| --- |
| 1.开销 2.管控，避免竞争了阻塞 3.功能加强 |

###### 11、常用的并发工具类有哪些？

|  |
| --- |
| 1.CountDownLatch 2. CyclicBarrier 3.Semaphore 4 .Exchanger |

###### 12、CyclicBarrier 和 CountDownLatch 的区别

|  |
| --- |
| 见上 |

###### 13、synchronized 的作用？

|  |
| --- |
| 控制线程同步 |

###### 14、volatile 关键字的作用

|  |
| --- |
| 有序性和可见性 |

###### 15、什么是 CAS

|  |
| --- |
| Compare and set |

###### 16、CAS 的问题

|  |
| --- |
| 1.ABA问题，加版本号 2.轮询消耗问题 3.不能保证原子性 |

###### 17、什么是 Future？

|  |
| --- |
| 通过 实现 Callback 接口 ， 并 用 Future 可以 来接 收多 线程 的执 行结 果。Future 表示 一个 可能 还没 有完 成的 异步 任务 的结 果， 针对 这个 结果 可以 添 加 Callback 以便 在任 务执 行成 功或 失败 后作 出相 应的 操作 。 |

###### 18、什么是 AQS

|  |
| --- |
| 抽象队列同步器， 它 是 一 个 Java 提 高 的 底 层 同 步 工 具 类 ， 用 一 个 int 类 型 的 变 量 表 示 同 步 状 态 ， 并 提 供 了 一 系 列 的 CAS 操 作 来 管 理 这 个 同 步 状 态 。  AQS 是 一 个 用 来 构 建 锁 和 同 步 器 的 框 架 ，使 用 AQS 能 简 单 且 高 效 地 构 造 出 应 用 广 泛 的 大 量 的 同 步 器 ， 比 如 我 们 提 到 的 ReentrantLock ， Semaphore ， 其 他 的 诸 如 ReentrantReadWriteLock ， SynchronousQueue ， FutureTask 等 等 皆 是 基 于 AQS 的 。 |

###### 19、AQS 支持两种同步方式：

|  |
| --- |
| 独占式和共享式 |

###### 20、 ReadWriteLock 是什么

|  |
| --- |
| 实 现 了读写的分 离 ， 读锁 是共 享的 ， 写锁 是独 占的 ， 读和 读之 间不 会互 斥 ， 读和 写 、 写和 读 写和 写之 间才 会互 斥， 提升 了读 写的 性能 。 |

###### 21、 FutureTask 是什么

|  |
| --- |
| FutureTask表示一个异步运算的任务 FutureTask 里面 可以 传入 一个 Callable 的具 体实 现类 ， 可 以对 这个 异步 运算 的任 务的 结果 进行 等 待获 取、 判断 是否 已经 完成 、取 消任 务等 操作 。当 然， 由于 FutureTask 也是 Runnable 接口 的实 现类 ， 所 以 FutureTask 也可 以放 入线 程池中。 |

###### 22、synchronized 和 ReentrantLock 的区别

|  |
| --- |
| synchronized 是和 if 、else 、for 、while 一样的关键字 ，使用的对象头中的mark work。  ReentrantLock 是类，更灵活。  ReentrantLock 可获取锁的等待时间，灵活实现多路通知，底层用Unsafe和park方法加锁。 |

###### 23、什么是乐观锁和悲观锁

|  |
| --- |
| 见上 |

###### 24、线程 B 怎么知道线程 A 修改了变量

|  |
| --- |
| 1.Volatile修饰变量，synchronized 修饰 修改 变量 的方 法 2.Wait、notify 3.while轮询 |

###### 25、synchronized、volatile、CAS 比较

|  |
| --- |
| 1.synchronized 悲观锁，抢占式，会引起阻塞 2.volatile 可见和有序性 3.CAS 基于冲突检查的乐观锁 |

###### 26、sleep 方法和 wait 方法有什么区别?

|  |
| --- |
| Sleep抱着锁。 |

###### 27、ThreadLocal 是什么？有什么用？

|  |
| --- |
| 一个本地线程副本变量工具类。 |

###### 28、为什么 wait()方法和 notify()/notifyAll()方法要在同步块中被调用

|  |
| --- |
| 这些方法需持有锁者才能调用 |

###### 29、多线程同步有哪几种方法？

|  |
| --- |
| Synchronized关键字， Lock锁实现，分布式锁等 。 |

###### 30、线程的调度策略

|  |
| --- |
| 1.线程调度按照优先级 2.yeild sleep IO阻塞 更高优先级的线程出现 时间片用完 |

###### 31、ConcurrentHashMap 的并发度是什么

|  |
| --- |
| 16 |

###### 32、 Linux 环境下如何查找哪个线程使用 CPU 最长

|  |
| --- |
| ps -ef | grep java  top - H - p pid， 顺 序 不 能 改 变 |

###### 33、Java 死锁以及如何避免？

|  |
| --- |
| 避免四个条件中的一个 |

###### 34、死锁的原因

|  |
| --- |
| 竞争关系 互相持锁不让 不能强夺他锁 一直等着不释放 |

###### 35、怎么唤醒一个阻塞的线程

|  |
| --- |
| 普通阻塞，中断线程即可唤醒 IO阻塞无能为力 |

###### 36、不可变对象对多线程有什么帮助

|  |
| --- |
| 无需同步策略，安全 |

###### 37、什么是多线程的上下文切换

|  |
| --- |
| 见上 |

###### 38、如果你提交任务时，线程池队列已满，这时会发生什么 248

|  |
| --- |
| 排队--》扩大线程池数量--》抛出异常 |

###### 39、Java 中用到的线程调度算法是什么

|  |
| --- |
| 抢占式 |

###### 40、什么是线程调度器(Thread Scheduler)和时间分片(TimeSlicing)？

|  |
| --- |
| 线 程 调 度 器 是 一 个 操 作 系 统 服 务 ，它 负 责 为 Runnable 状 态 的 线 程 分 配 CPU 时 间 |

###### 41、什么是自旋

|  |
| --- |
| 短时间没了锁应该先别阻塞，hang一会儿，阻塞设计内核态和用户态切换，消耗性能。 |

###### 42、Java Concurrency API 中的 Lock 接口(Lock interface)是什么？对比同步它有什么优势？

|  |
| --- |
|  |

###### 43、单例模式的线程安全性

|  |
| --- |
| 饿汉式安全 懒汉式不安全 双检锁安全 |

###### 44、Semaphore 有什么作用 249

|  |
| --- |
| Semaphore 就是 一个 信号 量 ，它的 作用 是限 制某 段代 码块 的并 发数 。 |

###### 45、 Executors 类是什么？

|  |
| --- |
| Executors 为 Executor ， ExecutorService ，ScheduledExecutorService，ThreadFactory 和 Callable 类提 供了 一些 工具 方法 。  Executors 可以 用于 方便 的创 建线 程池 |

###### 47、同步方法和同步块，哪个是更好的选择?

|  |
| --- |
| 见上 |

###### Java 线程数过多会造成什么异常？

|  |
| --- |
| 1.线程生命周期开销高 2.CPU消耗多 3.降低稳定性。 |

##### Java 面试题(一) 250

###### 1、面向对象的特征有哪些方面？

|  |
| --- |
| 抽象，继承，封装，多态 |

###### 2、访问修饰符 public,private,protected,以及不写(默认)时的区别？

|  |
| --- |
| Public protected defualt private  其他包 子类 同包 当前类 |

###### 3、String 是最基本的数据类型吗？

|  |
| --- |
| No |

###### 4、float f=3.4;是否正确？ 252

|  |
| --- |
| Float f = 3.4f; |

###### 5、short s1 = 1; s1 = s1 + 1;有错吗?short s1 = 1; s1 += 1;有错吗？

|  |
| --- |
| +=这个连体操作会隐式改变数据类型 |

###### 6、Java 有没有 goto？

|  |
| --- |
| 暂时没有 |

###### 7、int 和 Integer 有什么区别？

|  |
| --- |
| 基本类型和包装类型的区别 |

###### 8、&和&&的区别？

|  |
| --- |
| 前者与，后者短路运算 |

###### 9、解释内存中的栈(stack)、堆(heap)和方法区(method area)的用法。

|  |
| --- |
| 栈：对象和方法的指针，常量  堆：构造器等创建的对象  方法区：被JVM加载的类信息，常量，静态变量等 |

###### 10、Math.round(11.5) 等于多少？ Math.round(-11.5)等于多少？

|  |
| --- |
| 向上取整 |

###### 11、switch 是否能作用在 byte 上，是否能作用在 long 上，是否能作用在 String 上？

|  |
| --- |
| JDK7开始可以了 |

###### 12、用最有效率的方法计算 2 乘以 8？

|  |
| --- |
| 2>>4 |

###### 13、数组有没有 length()方法？ String 有没有 length()方法？

|  |
| --- |
| 属性和方法 |

###### 14、在 Java 中，如何跳出当前的多重嵌套循环？

|  |
| --- |
| 在循环口写一个标识，break 标识； |

###### 14、 构造器(constructor)是否可被重写(override)？

|  |
| --- |
| 可以 |

###### 16、两个对象值相同(x.equals(y) == true)，但却可有不同的hash code，这句话对不对？

|  |
| --- |
| 不可以 |

###### 17、是否可以继承 String类？

|  |
| --- |
| Final修饰的类不可以被继承 |

###### 18、当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传 递还是引用传递？

|  |
| --- |
| 值传递 |

###### 19、String 和 StringBuilder、StringBuffer 的区别？

|  |
| --- |
| String 不可变 stringBuffer并发安全 |

###### 20、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？

|  |
| --- |
| 只能根据入参区分 |

###### 21、描述一下JVM 加载 class 文件的原理机制？

|  |
| --- |
| 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  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 |

###### 22、char 型变量中能不能存贮一个中文汉字，为什么？

|  |
| --- |
| 可以 一 个 char 类型 占 2 个字 节 |

###### 24、静态嵌套类(Static Nested Class)和内部类(Inner Class)的不同？

|  |
| --- |
| Static Nested Class 是被 声明 为静 态 (static) 的内 部类 ，它可 以不 依赖 于外 部类 实例 被实 例化 。而通常的内部类需要在外部类实例化后才能实例化， 其语法看起来挺诡异的 |

###### 25、Java 中会存在内存泄漏吗，请简单描述。

|  |
| --- |
| 比如栈的pop操作就可能造成内存泄露 |

###### 26、抽象的(abstract)方法是否可同时是静态的(static) ,是否可同时是本地方法(native) ，是否可同时被 synchronized 修饰？

|  |
| --- |
| 抽象方法等着被实现故不可以，synchronize的实现和对象等有关，本地方法不可以 |

###### 27、阐述静态变量和实例变量的区别。

|  |
| --- |
| 静态变量在类类中，可直接调用。实例变量是实例化或者为各个具体对象所有。 |

###### 28、是否可以从一个静态(static)方法内部发出对非静态(non-static)方法的调用？

|  |
| --- |
| 不可以，非静态有可能还没来得及实例化 |

###### 29、如何实现对象克隆？

|  |
| --- |
| 1.实现cloneable接口，并重写Object类中的clone()方法  2.实现serializble接口，通过对象的序列化和反序列化进行深度克隆。 |

###### 30、GC 是什么？为什么要有 GC？

|  |
| --- |
| 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 。 |

###### 31、String s = new String(“xyz”);创建了几个字符串对象？

|  |
| --- |
| 两个：一个在静态区“xyz”，一个用new创建在堆上。 |

###### 33、一个”.java”源文件中是否可以包含多个类(不是内部类)？有什么限制？

|  |
| --- |
| 只能一个公开类Public class |

###### 34、Anonymous Inner Class(匿名内部类)是否可以继承其它类？是否可以实现接口？

|  |
| --- |
| 可以 |

###### 35、内部类可以引用它的包含类(外部类)的成员吗？有没有什么限制？

|  |
| --- |
| 一个内部类对象可以访问创建他的外部类对象的成员，包括私有成员。 |

###### 36、Java 中的 final 关键字有哪些用法？

|  |
| --- |
| 修饰 |

###### 37、指出下面程序的运行结果

|  |
| --- |
| 1静态成员 2.父类构造函数 3.非静态函数 4.自身构造器 |

###### 38、数据类型之间的转换：

|  |
| --- |
| 隐式转换 |

###### 39、如何实现字符串的反转及替换？

|  |
| --- |
| 简单 |

###### 40、怎样将 GB2312 编码的字符串转换为 ISO-8859-1 编码的字符串？ 279

|  |
| --- |
| String s2 = new String(s1.getBytes("GB2312"), "ISO-8859-1"); |

###### 41、日期和时间：

|  |
| --- |
| java .time . LocalDateTimel |

###### 42、打印昨天的当前时刻。

|  |
| --- |
|  |

###### 43、比较一下 Java 和 JavaSciprt。 282

|  |
| --- |
| Java：编译-介绍 js:解释性语言 |

###### 44、什么时候用断言(assert)？ 283

|  |
| --- |
| 断言 在软 件开 发中 是一 种常 用的 调试 方式 ， 很 多开 发语 言中 都支 持这 种机 制。 |

###### 45、 Error 和 Exception 有什么区别？

|  |
| --- |
| Error:不好捕获和处理的报错  Exception：可预见性的报错，需捕获 |

###### 46、try{}里有一个 return 语句，那么紧跟在这个 try 后的finally{}里的代码会不会被执行，什么时候被执行，在 return 前还是后?

|  |
| --- |
| Finally里面一定会执行，而里面的return会为他先暂停一下 |

###### 47、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？

|  |
| --- |
| 1.throws是捕获，2.throw是抛出具体错误 3.try cathc finally组合 |

**48、****运行时异常与受检异常有何异同？**

|  |
| --- |
| 运行时异常不必处理 受检时异常需要捕获和处理 |

###### 49、列出一些你常见的运行时异常？

|  |
| --- |
| ArithmeticException (算术异常) ClassCastException (类转换异常) IllegalArgumentException (非法参数异常) IndexOutOfBoundsException (下标越界异常) NullPointerException (空指针异常)  SecurityException (安全异常) |

###### 50、阐述 final、finally、finalize 的区别。

|  |
| --- |
| Final:修饰符 finally是try catch的配套 finalize:销毁回收的对象时调用 |

###### 51、类 ExampleA 继承 Exception，类 ExampleB 继承ExampleA。

|  |
| --- |
| try {  throw new ExampleB("b")  } catch (ExampleA e) {  System.out.println("ExampleA");  } catch (Exception e) {  System.out.println("Exception");  }  \*\*请问执行此段代码的输出是什么？  答：输出 ： ExampleA 。 (根据里氏代换原则[能使用父类型的地方一定能使用子类型]， 抓取 ExampleA 类型异常的 catch 块能够抓住 try 块中抛出的 ExampleB 类型的 异常) |

###### 53、阐述 ArrayList、Vector、LinkedList 的存储性能和特性。

|  |
| --- |
| 1.ArrayList和Vector底层都是数组，LinkedList底层是双向链表 2.查找ArrayList快，插入删除LingedList快 |

###### 54、Collection 和 Collections 的区别？

|  |
| --- |
| Collection是一个接口，是Set，List等容器的父接口；Coolections是一个工具类，对容器搜索排序等。 |

###### 55、List、 Map、 Set 三个接口存取元素时，各有什么特点？

|  |
| --- |
| List特定索引来存取元素，可重复；Set不可重；Map是件之多映射； |

###### 57、Thread 类的sleep()方法和对象的wait()方法都可以让线程暂停执行，它们有什么区别?

|  |
| --- |
| 抱锁睡觉 |

###### 58、线程的 sleep()方法和 yield()方法有什么区别？

|  |
| --- |
| 1.Sleep是让出CPU执行权,自己在设定时间内不在抢夺，yield()让出给更高优先级线程执行，进入ready状态； 2.sleep可抛出异常InterruptedException，yield方法没有申明异常。 |

###### 60、请说出与线程同步以及线程调度相关的方法。

|  |
| --- |
| Wait sleep notify notifyAll 等方法 |

###### 61、编写多线程程序有几种实现方式？

|  |
| --- |
| 1.继承Thread类 2.实现Runnable接口并从写run 3.实现Callable接口，并重写call |

###### 62、synchronized 关键字的用法？

|  |
| --- |
| 关键字标记到类 对象 方法等上面 |

###### 63、举例说明同步和异步。

|  |
| --- |
| 1.同步，阻塞操作，2.异步，非阻塞操作 |

###### 64、启动一个线程是调用 run()还是 start()方法？

|  |
| --- |
| Start() |

###### 65、什么是线程池(thread pool)？

|  |
| --- |
| 统一规划和管理线程的池子 |

###### 66、线程的基本状态以及状态之间的关系？

|  |
| --- |
| 5种 |

###### 67、简述 synchronized 和 java.util.concurrent.locks.Lock的异同？

|  |
| --- |
| Synchronized是偏向锁，是个关键词，同步时全自动。Luck是一个新的API，功能强化 |

###### 68、Java中如何实现序列化，有什么意义？

|  |
| --- |
| 序列 化就 是一 种用 来处 理对 象流 的机 制， 所谓 对象 流也 就是 将对 象的 内容 进行 流  化，可以持久化也可以深克隆。 |

###### 69、Java 中有几种类型的流？

|  |
| --- |
| 字节流：InputStream 、 OutputStream，  字符流：Reader 、 Writer |

###### 70、写一个方法，输入一个文件名和一个字符串，统计这个字符串在这个文件中出现的次数

|  |
| --- |
| 笔试题 |

###### 71、如何用 Java 代码列出一个目录下所有的文件？

|  |
| --- |
| 笔试题 |

###### 72、用 Java 的套接字编程实现一个多线程的回显(echo)服务器。

|  |
| --- |
| 笔试题 |

###### 73、XML 文档定义有几种形式？它们之间有何本质区别？解析XML 文档有哪几种方式？

|  |
| --- |
| XML 文档 定义 分为 DTD 和 Schema 两种形式 |

###### 74、你在项目中哪些地方用到了 XML？

|  |
| --- |
| 数据 交换 和信 息配 置 |

**75、****阐述 JDBC 操作数据库的步骤。**

|  |
| --- |
| 加载驱动 创建链接 创建语句 执行语句 关闭资源 |

**76、****Statement 和 PreparedStatement 有什么区别？哪个性能更好？**

|  |
| --- |
| ① PreparedStatement 接口 代表 预编 译的 语句 ，它主 要的 优 势在 于可 以减 少 SQL 的编 译错 误并 增加 SQL 的**安全性** (减 少 SQL 注射 攻击 的可能性 ) ；② PreparedStatement 中的 SQL 语句 是**可以带参数** 的 ，避免 了用 字符 串 连接 拼接 SQL 语句 的麻 烦和不安全；③当**批量处理** SQL 或频 繁执 行相 同的 查询 时 ， PreparedStatement 有明 显的 性能 上的 优势 ， 由 于数 据库 可以 将编 译优 化后 的 SQL 语句 缓存 起来 ， 下 次执 行相 同结 构的 语句 时就 会很 快(不用 再次 编译 和生 成 执行 计划 ) 。 |

###### 77、使用JDBC操作数据库时，如何提升读取数据的性能？如何提升更新数据的性能？

|  |
| --- |
| 1.可以批量操作 2.定量抓取数量 |

###### 78、在进行数据库编程时，连接池有什么作用？

|  |
| --- |
| 1.省开销 2.加强功能 3. |

###### 79、什么是 DAO 模式？

|  |
| --- |
| 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 |

###### 80、事务的 ACID 是指什么？

|  |
| --- |
| 1.A是可用性 2.C是一致性 3.I隔离性 4.D是持久性 |

###### 81、JDBC中如何进行事务处理

|  |
| --- |
| Connection 提 供 了 事 务 处 理 的 方 法 ， 通 过 调 用 setAutoCommit(false)可 以 设 置 手 动 提 交 事 务 ；当 事 务 完 成 后 用 commit()显 式 提 交 事 务； 如 果 在 事 务 处 理 过 程 中 发 生 异 常 则 通 过 rollback()进 行 事 务 回 滚 。 除 此 之 外 ，从 JDBC 3.0 中 还 引 入了Savepoint (保 存 点 ) 的 概 念 ，允 许 通 过 代 码设置保存点并让事务回滚到指定的保存点 。 |

###### 82、JDBC 能否处理 Blob 和 Clob？

|  |
| --- |
| Blob 是指 二进 制大 对象 (Binary Large Object) ， 而 Clob 是指 大字 符对 象 |

###### 83、简述正则表达式及其用途

|  |
| --- |
| 查找 匹配 |

###### 84、 Java中是如何支持正则表达式操作的？

|  |
| --- |
| String str = "北京市(朝阳区)(西城区)(海淀区)";  Pattern p = Pattern.compile(".\*?(?=\\()");  Matcher m = p.matcher(str); |

###### 85、获得一个类的类对象有哪些方式？

|  |
| --- |
| 1.类型.class 例如String.class 2.对象.getClass() 例如 “hello”.getClass()  3：Class.forName()，例如：Class.forName(“java.lang.String”) |

###### 85、如何通过反射创建对象？

|  |
| --- |
| 方法 1 ：通过类对象调用 newInstance()方法，例如：String.class.newInstance()  方法 2：通过类对象的 getConstructor()或 getDeclaredConstructor()方法获得构造器(Constructor)对象并调用其 newInstance()方法创建对象,例如String.class.getConstructor(String.class).newInstance(“Hello”) |

###### 86、如何通过反射获取和设置对象私有字段的值？

|  |
| --- |
| String str = "hello";  Method m = str.getClass().getMethod("toUpperCase");  System.out.println(m.invoke(str)); // HELLO |

###### 88、如何通过反射调用对象的方法？

|  |
| --- |
| String str = "hello";  Method m = str.getClass().getMethod("toUpperCase");  System.out.println(m.invoke(str)); // HELLO |

###### 89、 简 述一 下面 向对 象的 ”六原 则一 法则 ” 。 333

|  |
| --- |
| 单一 开闭 依赖 里氏 接隔 合成 迪米特 |

###### 90、简述一下你了解的设计模式。

|  |
| --- |
| 工厂 单利 代理 发布订阅 桥梁 |

###### 91、用 Java 写一个单例类。

|  |
| --- |
| public class Singleton {  private Singleton(){}  private static Singleton instance = new Singleton();  public static Singleton getInstance(){  return instance;  }  } |

###### 92、什么是 UML？

|  |
| --- |
|  |

###### 93、UML 中有哪些常用的图？ 337

|  |
| --- |
| 建 模 语 言 |

###### 94、 用 Java 写 一 个 冒 泡 排 序 。

|  |
| --- |
| 笔试题 |

###### 95、用 Java 写一个折半查找。 340

##### Java 面试题(二) 342

多线程、并发及线程的基础问题

###### 1、Java 中能创建 volatile 数组吗？

|  |
| --- |
| 1.只能对数组的引用这个串做可见性可限制指令重排 |

###### 2、volatile 能使得一个非原子操作变成原子操作吗？

|  |
| --- |
| 不可，除了long等极为特殊情况 |

###### 3、volatile 修饰符的有过什么实践？

|  |
| --- |
| 一种实践是用volatile修饰long和double 变量，使其能按原子类型来读写 |

###### 4、volatile 类型变量提供什么保证？

|  |
| --- |
| 可见性和禁止指令重排 |

###### 5、10 个线程和 2 个线程的同步代码，哪个更容易写？

|  |
| --- |
| 都一样 |

###### 6、你是如何调用wait () 方法的？使用if 块还是循环？为什么？

|  |
| --- |
| 使用while()子循环 |

###### 7、什么是多线程环境下的伪共享 (false sharing) ？

|  |
| --- |
| 物理内存访问出现了碰撞 |

###### 8、什么是Busy spin？我们为什么要使用它？

|  |
| --- |
| Busy spin是一种在不释放CPU的基础上等待事件的技术。 |

###### 9、Java 中怎么获取一份线程 dump 文件？

|  |
| --- |
| kill -3 PID |

###### 10、Swing 是线程安全的？ 346

|  |
| --- |
| 不是， Swing不是线程安全的。你不能通过任何线程来更新Swing组件 |

###### 11、什么是线程局部变量？

|  |
| --- |
| 线程局部变量是局限于线程内部的变量，属于线程自身所有，不在多个线程间共享，注意内存泄露问题 |

###### 12、用wait-notify写一段代码来解决生产者-消费者问题？

|  |
| --- |
| 笔试题 |

###### 13、用Java 写一个线程安全的单例模式(Singleton)？ 346

|  |
| --- |
| 笔试题 |

###### 14、Java 中 sleep 方法和wait方法的区别？

|  |
| --- |
| Sleep抱着锁 |

###### 15、什么是不可变对象 (immutable object) ？Java 中怎么创建一个不可变对象？

|  |
| --- |
| 不可继承，线程安全 |

###### 16、我们能创建一个包含可变对象的不可变对象吗？

|  |
| --- |
| 可以，但是谨慎处理 |

###### 17、Java中应该使用什么数据类型来代表价格？

|  |
| --- |
| BigDecimal |

###### 18、怎么将byte 转换为 String？

|  |
| --- |
|  |

###### 19、Java 中怎样将 bytes 转换为 long 类型？

###### 20、我们能将 int强制转换为 byte 类型的变量吗？如果该值大于byte类型的范围，将会出现什么现象？

|  |
| --- |
| 可以，丢失24位 |

###### 22、哪个类包含 clone 方法？是 Cloneable 还是 Object？

|  |
| --- |
| java . lang .Cloneable 是一 个标 示性 接口 ， 不 包含 任何 方法 ，clone 方法 在object 类中 定义 。并且 需要 知道 clone() 方法是一个本地方法 这意味着它是由c或c++或其他本地语言实现的。 |

###### 23、Java 中 ++ 操作符是线程安全的吗？

|  |
| --- |
| 不安全 |

###### 24、a = a + b 与 a += b 的区别

|  |
| --- |
| 后者会做隐式转换 |

###### 25、我能在不进行强制转换的情况下将一个 double 值赋值给long 类型的变量吗？

|  |
| --- |
| 不可 |

###### 26、 3\*0.1 == 0.3 将会返回什么？ true 还是 false？

|  |
| --- |
| False ，浮点数不能完全精确表示 |

###### 27、int 和Integer哪个会占用更多的内存？

|  |
| --- |
| Integer |

###### 28、为什么Java中的 String 是不可变的 ( Immutable)？

|  |
| --- |
| Final |

###### 29、我 们能 在 Switch 中使 用 String 吗？

|  |
| --- |
| JDK1.7时就可以了 |

###### 30 、 Java 中的 构造器链 是什 么？

|  |
| --- |
| 当你 从一 个构 造器 中调 用另 一个 构造 器 ，就是 Java 中的 构造 器链 。这种 情况 只在 重载 了类 的构 造器 的时 候才 会出 现。 |

###### 31、64 位 JVM 中， int 的长度是多数？

|  |
| --- |
| 32 |

###### 33、 32 位和 64 位的 JVM ，int 类型变量的长度是多数？ 349

|  |
| --- |
| 32 |

###### 34、Java 中 WeakReference 与 SoftReference 的区别？

|  |
| --- |
| WeakReference 是回光返照，SoftReference在内存不足发起GC的时候有用 |

###### 35、WeakHashMap 是怎么工作的？

|  |
| --- |
| 底层是数组加链表（红黑树） |

###### 36、JVM 选项 -XX:+UseCompressedOops 有什么作用？为什么要使用？

|  |
| --- |
| 32位虚拟机升级位64位时，通 过 -XX:+UseCompressedOops 选 项 ，JVM 会 使 用 32 位 的 OOP，而 不 是 64 位 的 OOP。 |

###### 37、怎样通过 Java 程序来判断 JVM 是 32 位 还是 64位？

|  |
| --- |
|  |

###### 38、 32 位 JVM 和 64 位 JVM 的最大堆内存分别是多数？

|  |
| --- |
| 2^32 2^64 |

###### 39、JRE、JDK、JVM 及 JIT 之间有什么不同？

|  |
| --- |
| JIT 代表即时编译 ( Just In Time compilation) |

###### 40、解释 Java 堆空间及 GC？

|  |
| --- |
| 堆存放实例化的对象 |

###### JVM 底层面试题及答案

###### 41、你能保证 GC 执行吗？

|  |
| --- |
| 不能 ，虽然你可以调用System .gc()或者Runtime .gc()，但是没有办法保证GC 的执行。 |

###### 42、怎么获取 Java 程序使用的内存？堆使用的百分比？

|  |
| --- |
| Runtime .freeMemory() |

###### 43、Java 中堆和栈有什么区别？

|  |
| --- |
| 共享云否，存放的东西不一样 |

###### 44、“a==b”和”a.equals(b)”有什么区别？

|  |
| --- |
| 复杂类型是，A==b是内存地址的比较，a.equals(b)比的是内容 |

###### 45、a.hashCode() 有什么用？与 a.equals(b) 有什么关系？

|  |
| --- |
| a.equals(b) =>a.hashCode() |

###### 46、final、finalize 和 finally 的不同之处？

|  |
| --- |
| 不同的位置 |

###### 47、Java中的编译期常量是什么？使用它又什么风险？

|  |
| --- |
| 公共静态不可变(public static final)变量也就是我们所说的编译期常量，这里的public可选的。 |

###### 48、List、Set、Map 和 Queue 之间的区别(答案)

|  |
| --- |
| List链表 Set是无重复的集合，Map是键值对，Queue是队列先进先出 |

###### 49、poll() 方法和 remove() 方法的区别？

|  |
| --- |
| Poll()空了时返回null，而remove为空是抛出异常 |

###### 50、Java中LinkedHashMap和 PriorityQueue的区别是什么？

|  |
| --- |
| PriorityQueue保证最高或者最低优先级的的元素总是在队列头部，但是 LinkedHashMap 维持的顺序是元素插入的顺序 。 |

###### 51、用哪两种方式来实现集合的排序？

|  |
| --- |
| TreeSet 或 TreeMap |

**52、****ArrayList 与 LinkedList 的不区别？**

|  |
| --- |
| 底层实现不一样 |

###### 53、Java 中怎么打印数组？

|  |
| --- |
| Arrays.toString() 和 Arrays.deepToString() |

###### 54、Java 中的 LinkedList 是单向链表还是双向链表？

|  |
| --- |
| 双向链表 |

###### 55、Java 中的 TreeMap 是采用什么树实现的？

|  |
| --- |
| 红黑树 |

###### 56、Hashtable 与 HashMap 有什么不同之处？

|  |
| --- |
| 线程安全问题 |

###### 57、Java 中的 HashSet，内部是如何工作的？

|  |
| --- |
| HashSet 的内 部采 用 HashMap 来实 现 ，不允许充分的key |

###### 59、我们能自己写一个容器类，然后使用 for-each 循环码？

|  |
| --- |
| 可以，注意实现iterator() |

###### 60、ArrayList 和 HashMap 的默认大小是多数？

|  |
| --- |
| ArrayList是10，HashMap是16 |

###### 61、有没有可能两个不相等的对象有有相同的 hashcode？

|  |
| --- |
| 有可能 |

###### 62、两个相同的对象会有不同的的 hashcode 吗？

|  |
| --- |
| 不会，重写了的话hashcode一定相同 |

###### 63、我们可以在 hashcode() 中使用随机数字吗？

|  |
| --- |
| 不可以，hashcode是和内容一对一严格映射的 |

###### 64、Java 中， Comparator 与 Comparable 有什么不同？

|  |
| --- |
| Comparable 接口用于定义对象的自然顺序，而comparator通常用于定义用户定制的顺序 。 |

###### 65) 为 什 么 在 重 写 equals 方 法 的 时 候 需 要 重 写 hashCode 方 法 ？ (答 案)

|  |
| --- |
| 不重写会出现各种问题 |

###### 66、在我 Java 程序中，我有三个 socket，我需要多少个线程来处理？

|  |
| --- |
|  |

###### 67、Java 中怎么创建 ByteBuffer？ 355

|  |
| --- |
| byte[] bytes = new byte[10];  ByteBuffer buf = ByteBuffer .wrap(bytes); |

###### 68、Java 中，怎么读写 ByteBuffer ？

|  |
| --- |
|  |

###### 69、Java 采用的是大端还是小端？ 355

|  |
| --- |
|  |

###### 70、 ByteBuffer 中的字节序是什么？ 355

|  |
| --- |
|  |

###### 71、Java 中，直接缓冲区与非直接缓冲器有什么区别？ 355

|  |
| --- |
|  |

###### 72、Java 中的内存映射缓存区是什么？ 355

|  |
| --- |
|  |

###### 73、socket 选项 TCP NO DELAY 是指什么？ 355

|  |
| --- |
|  |

###### 74、TCP 协议与 UDP 协议有什么区别？ 356

|  |
| --- |
|  |

###### 75、Java 中，ByteBuffer 与 StringBuffer 有什么区别？ (答案) 356

|  |
| --- |
|  |

###### 76、Java 中，编写多线程程序的时候你会遵循哪些最佳实践？

|  |
| --- |
| 1.给线程命名 2.最小同步范围 3.能用volatile就不用synchronized 4.使用更高级的并发工具。 |

###### 77、说出几点 Java 中使用 Collections 的最佳实践 356

|  |
| --- |
| 1.使用正确的集合 2.优先使用并发集合 3.使用接口代表和访问集合 4.使用迭代器来循环集合 5使用泛型 |

###### 78、说出至少 5 点在 Java 中使用线程的最佳实践。

|  |
| --- |
| a) 对 线程 命名  b) 将 线程 和任 务分 离， 使用 线程 池执 行器 来执 行 Runnable 或 Callable。  c) 使 用线 程池 |

###### 79、说出 5 条 IO 的最佳实践(答案)

|  |
| --- |
| a) 使 用有 缓冲 区的 IO 类， 而不 要单 独读 取字 节或 字符 。b) 使用NIO和NIO2 c)在finally块中关闭流，或者使用try -with - resource语句 d)使用内存映射文件获取更快的IO。 |

###### 80、列出5个应该遵循的JDBC最佳实践

|  |
| --- |
| b) 使 用 PreparedStatement 来避 免 SQL 异常 ， 并 提高 性能 。  c) 使 用数 据库 连接 池 d) 通 过列 名来 获取 结果 集， 不要 使用 列的 下标 来获 取。 |

###### 81、说出几条 Java 中方法重载的最佳实践？

|  |
| --- |
| a) 不要重载这样的方法：一个方法接收int参数， 而另个方法接收Integer参数 b) 不要重载参数数量一致，而只是参数顺序不同的方法 。c) 如果重载的方法参数个数多于5个，采用可变参数 |

###### 82、在多线程环境下，SimpleDateFormat 是线程安全的吗？

|  |
| --- |
| 不是 ，非常不幸，DateFormat的所有实现，包括SimpleDateFormat都不是线程安全的， |

###### 83、Java 中如何格式化一个日期？如格式化为 ddMMyyyy的形式？

|  |
| --- |
| SimpleDateFormat 类或 者 joda -time 库来 格式 日期 |

###### 84、Java 中，怎么在格式化的日期中显示时区？ 358

|  |
| --- |
|  |

###### 85、Java 中 java.util.Date 与 java.sql.Date 有什么区别？ 358

|  |
| --- |
|  |

###### 86、Java 中，如何计算两个日期之间的差距？ 358

|  |
| --- |
|  |

###### 87、Java 中，如何将字符串 YYYYMMDD 转换为日期？ 358

|  |
| --- |
|  |

###### 89、如何测试静态方法？ (答案) 358

|  |
| --- |
| 可以 使用 PowerMock 库来 测试 静态 方法 。 |

###### 90、怎么利用 JUnit 来测试一个方法的异常？ 359

|  |
| --- |
| [http://javarevisited.blogspot.sg/2013/04/JUnit-tutorial-example-test-exce](http://javarevisited.blogspot.sg/2013/04/JUnit-tutorial-example-test-exception-thrown-by-java-method.html) ption -thrown - by -java - method . html |

###### 91、你使用过哪个单元测试库来测试你的 Java 程序？ 359

|  |
| --- |
|  |

###### 92、@Before和 @BeforeClass有什么区别？ 359

|  |
| --- |
| [http://javarevisited.blogspot.sg/2013/04/JUnit-tutorial-example-test-exce](http://javarevisited.blogspot.sg/2013/04/JUnit-tutorial-example-test-exception-thrown-by-java-method.html) ption -thrown - by -java - method . html |

###### 93、怎么检查一个字符串只包含数字？解决方案

|  |
| --- |
| 笔试题 |

###### 94、Java 中如何利用泛型写一个 LRU 缓存？

|  |
| --- |
| 笔试题 |

###### 95、写一段 Java 程序将 byte 转换为 long？

|  |
| --- |
| [http://java67.blogspot.com/2012/12/how-to-reverse-string-in-java-string](http://java67.blogspot.com/2012/12/how-to-reverse-string-in-java-stringbuffer-stringbuilder.htm) buffer - stringbuilder. htm |

###### 95、在不使用 StringBuffer 的前提下，怎么反转一个字符串？ 359

|  |
| --- |
| 笔试题 |

###### 97、Java 中，怎么获取一个文件中单词出现的最高频率？

|  |
| --- |
| 笔试题 |

###### 98、如何检查出两个给定的字符串是反序的？

|  |
| --- |
| 笔试题 |

###### 99、Java 中，怎么打印出一个字符串的所有排列？

|  |
| --- |
| 笔试题 |

###### 100、Java 中，怎样才能打印出数组中的重复元素？

|  |
| --- |
| Tosring() |

###### 101、Java 中如何将字符串转换为整数？

|  |
| --- |
| i=Integer . parseInt(s); |

###### 102、在没有使用临时变量的情况如何交换两个整数变量的值？

|  |
| --- |
| 笔试题 |

###### 103、接口是什么？为什么要使用接口而不是直接使用具体类？ 360

|  |
| --- |
| 接 口 中 不 允 许 写 代 码 ， 以 此 来 保 证 抽 象 ， 但 是 Java 8 中 你 可 以 在 接 口 声 明 静 态 的 默 认 方 法 ， 这 种 方 法 是 具 体 的 。 |

###### 104、Java 中，抽象类与接口之间有什么不同？

|  |
| --- |
| 接口笔抽象类更加抽象；但是最重要的一个是 Java 中限制一个 类只能继承一个类， 但是可以实现多个接口。抽象类可以很好的定义一个家族类 的默认行为， 而接口能更好的定义类型，有助于后面实现多态机制。 |

###### 105、除了单例模式，你在生产环境中还用过什么设计模式？

|  |
| --- |
| 工厂模式 代理模式 观察者模式 发布订阅模式 单例模式 |

###### 106、你能解释一下里氏替换原则吗?

|  |
| --- |
| 父类运行的地方，继承他的子类也可以替换掉他 |

###### 107) 什么情况下会违反迪米特法则？为什么会有这个问题？

|  |
| --- |
| 迪米特法则建议“只和朋友说话，不要陌生人说话”以此来减少类之间的耦合 |

###### 108、适配器模式是什么？什么时候使用？

|  |
| --- |
| 接口转换的时候 |

###### 109、什么是“依赖注入”和“控制反转”？为什么有人使用？ 361

|  |
| --- |
| 控制 反转 (IOC) 是Spring框架的核心思想，用我自己的话说，就是你要做一件事，别自己可劲new了，你就说你要干啥，然后外包出去就好 ~  依赖 注入(DI)在我浅薄的想法中，就是通过接口的引用和构造方法的表达，将一些事情整好了反过来传给需要用到的地方 ~ |

###### 110、抽象类是什么？它与接口有什么区别？你为什么要使用过抽象类？

|  |
| --- |
| 类只能单继承，接口可以多继承 |

###### 111、构造器注入和 setter 依赖注入，那种方式更好？

|  |
| --- |
| 构造器注入保证所有的注入都被初始化，但是setter注入提供更好的灵活性来设置可选依赖。如果使用XML来描述依赖，Setter注入的可读写会更强。经验法则是强制依赖使用构造器注入，可选依赖使用setter注入 。 |

###### 112、依赖注入和工程模式之间有什么不同？

|  |
| --- |
| 虽然 两种 模式 都是 将对 象的 创建 从应 用的 逻辑 中分 离， 但是 依赖 注入 比工 程模 式 更清 晰。 通过 依赖 注入 ， 你 的类 就是 POJO， 它 只知 道依 赖而 不关 心它 们怎 么获 取。 使用 工厂 模式 ， 你 的类 需要 通过 工厂 来获 取依 赖。 因此 ， 使 用 DI 会比 使用 工厂 模式 更容 易测 试。 |

###### 113、适配器模式和装饰器模式有什么区别？ 362

|  |
| --- |
| 虽然 适配 器模 式和 装饰 器模 式的 结构 类似 ， 但 是每 种模 式的 出现 意图 不同 。适 配 器模式被用于桥接两个接口 ，而 装饰 模式 的目 的是 在不 修改 类的 情况 下给 类增 加 新的 功能 。 |

###### 114、适配器模式和代理模式之前有什么不同？ 362

|  |
| --- |
| 这个 问题 与前 面的 类似 ， 适 配器 模式 和代 理模 式的 区别 在于 他们 的意 图不 同。 由 于适 配器 模式 和代 理模 式都 是封 装真 正执 行动 作的 类， 因此 结构 是一 致的 ， 但 是 适配 器模 式用 于接 口之 间的 转换 ， 而 代理 模式 则是 增加 一个 额外 的中 间层 ， 以 便 支持 分配 、控 制或 智能 访问 。 |

###### 115、什么是模板方法模式？ 362

|  |
| --- |
| 模板 方法 提供 算法 的框 架， 你可 以自 己去 配置 或定 义步 骤。 例如 ， 你 可以 将排 序 算法 看做 是一 个模 板。 它定 义了 排序 的步 骤， 但是 具体 的比 较， 可以 使用 Comparable 或者 其语 言中 类似 东西 ， 具 体策 略由 你去 配置 。列 出算 法概 要的 方 法就 是众 所周 知的 模板 方法 。 |

###### 116、什么时候使用访问者模式？ 362

|  |
| --- |
| 访问 者模 式用 于解 决在 类的 继承 层次 上增 加操 作， 但是 不直 接与 之关 联。 这种 模 式采 用双 派发 的形 式来 增加 中间 层。 |

###### 117、什么时候使用组合模式？

|  |
| --- |
| 组合 模式 使用 树结 构来 展示 部分 与整 体继 承关 系。 它允 许客 户端 采用 统一 的形 式 来对 待单 个对 象和 对象 容器 。当 你想 要展 示对 象这 种部 分与 整体的继承关系时采用组合模式。 |

###### 118、继承和组合之间有什么不同？ 363

|  |
| --- |
| 虽然 两种 都可 以实 现代 码复 用， 但是 组合 比继 承共 灵活 ， 因 为组 合允 许你 在运 行 时选 择不 同的 实现 。用 组合 实现 的代 码也 比继 承测 试起 来更 加简 单。 |

###### 119、描述 Java 中的重载和重写？ 363

|  |
| --- |
| 重载在同一个类之间 |

###### 120、Java 中，嵌套公共静态类与顶级类有什么不同？

|  |
| --- |
| 类的 内部 可以 有多 个嵌 套公 共静 态类 ， 但 是一 个 Java 源文 件只 能有 一个 顶级 公 共类 ， 并 且顶 级公 共类 的名 称与 源文 件名 称必 须一 致 |

###### 121、 OOP 中的 组合、聚合和关联有什么区别？

|  |
| --- |
| 组合 和聚 合是 面向 对象 中 的两 种形 式的 关联 。组 合是 一种 比聚 合更 强力 的关 联。 |

###### 122、给我一个符合开闭原则的设计模式的例子？

|  |
| --- |
| Collections .sort()方法，这就是基于策略模式，遵循开闭原则的，你不需为新的对象修改sort()方法，你需要做的仅仅是实现你自己的Comparator接口。 |

###### 123、抽象工厂模式和原型模式之间的区别？ 363

|  |
| --- |
| 抽象 工厂 模式 ： 通 常由 工厂 方法 模式 来实 现。 但一 个工 厂中 往往 含有 多个工厂方法生成一系列的产品。这个模式强调的是客户代码一次保证只使用一个系列的产品。当要切换为另一个系列的产品，换一个工厂类即可 |

###### 125、嵌套静态类与顶级类有什么区别？

|  |
| --- |
| 一个 公共 的顶 级类 的源 文件 名称 与类 名相 同，而嵌 套静 态类 没有 这个 要求 。一个嵌套 类位 于顶 级类 内部 ，需要使用顶级类 的名 称来 引用 嵌套 静态 类，如 HashMap . Entry是一个嵌套静态类 ，HashMap是一个顶 级类 ，Entry是一个嵌 套静 态类 。 |

###### 126、你能写出一个正则表达式来判断一个字符串是否是一个数字吗？

|  |
| --- |
| 笔试题 |

###### 127、Java 中，受检查异常和不受检查异常的区别？

|  |
| --- |
| 受检异常在编译期间出现，需强制处理 |

###### 128、Java 中， throw 和 throws 有什么区别

|  |
| --- |
| 位置不同 |

###### 129、Java 中，Serializable与Externalizable 的区别？

|  |
| --- |
| Serializable接口是一个序列化Java类的接口。Externalizable允许你控制整个序列化过程，指定特定的二进制格式，增加安全机制。 |

###### 130、Java 中， DOM 和 SAX 解析器有什么不同？

|  |
| --- |
| DOM 解 析 器 将 整 个 XML 文 档 加 载 到 内 存 来 创建 一 棵 DOM 模 型 树 ，这 样 可 以 更 快 的 查 找 节 点 和 修改 XML 结 构 |

###### 131、说出 JDK 1.7 中的三个新特性？

|  |
| --- |
| Swatch支持string try-with-tesource |

###### 132、说出 5 个 JDK 1.8 引入的新特性？

|  |
| --- |
| Lambda ，接口中可以有静态方法，可重复注解 |

###### 133、Java 中， Maven 和 ANT 有什么区别？

|  |
| --- |
| 虽然 两者 都是 构建 工具 ， 都 用于 创建 Java 应用 ， 但 是 Maven 做的 事情 更多 |

##### 6：Java 附加题

###### 6.1 equals 与==的区别

|  |
| --- |
| 复杂类型时，Equal比较类容，==比较地址 |

###### 6.2 final, finally, finalize 的区别

|  |
| --- |
| 见上 |

###### 6.3 重载和重写的区别

|  |
| --- |
| 见上 |

###### 6.4 两个对象的 hashCode()相同，则 equals()是否也一定

|  |
| --- |
| 不一定，hash碰撞 |

###### 6.5 抽象类和接口有什么区别

|  |
| --- |
| 见上 |

###### 6.6 BIO、NIO、AIO 有什么区别？

|  |
| --- |
| BIO：阻塞性IO，线程一直阻塞直至任务完成。  NIO：线程发起IO请求，立即返回；内核咋做好IO操作准备之后，用过调用注册的回调函数通知线程做IO操作，线程开始阻塞，直到完成任务。  AIO：线程发起IO请求，立即返回。内存做好IO操作准备之后，做IO操作，直到IO操作完成或者失败，通过调用注册的回调函数通知线程做IO操作完成或者失败。 |

###### 6.7 String ，Stringbuffer ，StringBuilder 的区别

|  |
| --- |
| 见上 |

###### 6.8 JAVA中的几种基本数据类型是什么， 各自占用多少字节

|  |
| --- |
| Bite short int long boolean char float double |

###### 6.9 Comparator 与 Comparable 有什么区别？

|  |
| --- |
| Comparable&Comparator都是用来实现集合中元素的比较、排序的，只是Comparable 是在集合内部定义的方法实现的排序，Comparator是在集合外部实现的排序 |

###### 6.10 String 类能被继承吗，为什么。

|  |
| --- |
| 不可 |

###### 6.11说说Java中多态的实现原理

|  |
| --- |
| 多态机制包括静态多态 ( 编译时多态 )和动态多态 ( 运行时多态 )  静态多态比如说重载， 动态多态 一般指在运行时才能确定调用哪个方法。 |

###### 6.12 Java 泛型和类型擦除

|  |
| --- |
| 其本质是参数化类型，解决不确定具体对象类型的问题。泛型类、泛型接口、泛型方法。  ？ 表示不确定的 java 类型  T (type) 表示具体的一个java类型  K V (key value) 分别代表java键值中的Key Value  E (element) 代表Element |

###### 6.13 int 和 Integer 有什么区别，还有 Integer 缓存的实

|  |
| --- |
| 基本类型和包装类型，128以内在常量池里面公用 |

###### 6.14 说说反射的用途及实现原理，Java 获取反射的三种方法

|  |
| --- |
| 1.Class.forName 2.使用类的.class方法 3.使用实例对象的getClass()方法 |

###### 6.15 面向对象的特征

|  |
| --- |
| 封装 继承 多态 |

###### 6.16 &和&&的区别 376

|  |
| --- |
| 1.a&b 转为二进制数，在进行运算 2.逻辑和短路 |

###### 6.17 Java 中 IO 流分为几种? 376

|  |
| --- |
| 1.一种是字节流，一种是字符流 2.IO 流分别由四个抽象类来表示 ( 两输入两输  出 ) :InputStream ，OutputStream ，Reader ，Writer |

###### 6.18 讲讲类的实例化顺序，比如父类静态数据，构造函数

|  |
| --- |
| 父类静态代码块/静态域 -> 子类静态代码块/静态域 -> 父类非静态代码块->父类构造器 -> 子类非静态代码块->子类构造器 |

###### 6.19 Java 创建对象有几种方式

|  |
| --- |
| 1.new 2.使用反射 3.调用对象的clone 4.运用反序列化手段 5.使用Unsafe |

###### 6.20 如何将 GB2312 编码的字符串转换为 ISO-8859- 1

|  |
| --- |
| String s1 = "你好";  String s2 = new String(s1.getBytes("GB2312"), "ISO-8859-1"); |

###### 6.21 守护线程是什么？用什么方法实现守护线程 378

|  |
| --- |
| 特殊线程，例如垃圾回收 |

###### 6.22 notify()和 notifyAll()有什么区别？

|  |
| --- |
| Notify唤醒一个wait状态线程，notifyAll则是唤醒所有 |

###### 6.23 Java 语言是如何处理异常的，关键字

|  |
| --- |
| Throws throw tyr catch finally |

###### 6.24 谈谈 Java 的异常层次结构

|  |
| --- |
| Throwable(Error,Exception), Error(OutOfMemoryError 等) Exception(RuntimeException (NullPointerException), Checked Exception(IOException,ClassCastException) ) |

###### 6.25 静态内部类与非静态内部类有什么区别

|  |
| --- |
| 静态内部类：可有静态成员，只能访问外部类的静态成员  静态内部类和非静态的实例化方式不同  静态内部类的静态成员可直接通过类名来调用。 |

###### 6.26 String s 与 new String 与有什么区别

|  |
| --- |
| String str =”xhw” 会先从常量池中查找(无则在栈中创建)，并让str指向此地址  String str = new String(“whx”) 会在队中创建”whx”, 在栈中创建”whx”并让stri指向此地址 |

###### 6.27 反射中， Class.forName 和ClassLoader 的区别

|  |
| --- |
| ClassLoader加载java类的字节码到虚拟机中去，  lass.forName 和 ClassLoader 的区别， 就是在类加载的时候 ， class.forName 有参数控制是否对类进行初始化。 |

###### 6.28 JDK 动态代理与cglib 实现的区别

|  |
| --- |
| 动态代理是利用反射机制生成一个实现代理接口的匿名类，在调用体法前调用InvokeHandler 来处理。  cglib 动态代理是利用asm 开源包，修改字节码生成子类来处理，该类最好不要用final  JDK基于java自带的类，只能实现接口生成代理 |

###### 6.29 error 和exception 的区别 ，

|  |
| --- |
| Error:系统错误，无法处理 ；Exception:分为运行时异常和可检查异常 |

###### 6.30 深拷贝和浅拷贝区别

|  |
| --- |
| 复制引用和复制堆内存的内容 |

###### 6.31 JDK 和 JRE 有什么区别？

|  |
| --- |
| JDK：java开发和运行环境，JRE：java运行时环境 |

###### 6.32 String 类的常用方法都有那些呢？ 385

|  |
| --- |
| IndexOf charAt replace trim split getBytes length |

###### 6.33 谈谈自定义注解的场景及实现 385

|  |
| --- |
| 之前我这边有这么一个业务场景 ，用 Redis 控制接口调用频率 ，有使用过自定义注 解。  . 通过 AOP ( 动态代理机制 ) 给方法添加切面， 通过反射来获取方法包含的注解， 如果包含自定义关键字注解， 就通过 Redis 进行校验拦截请求。 |

###### 6.34 说说你熟悉的设计模式有哪些？

|  |
| --- |
| 创建型：工厂 抽象工厂 单利 建造者 原型  结构型：适配器 装饰着，代理 外观 桥接 ，组合 ，享元模式  行为型：策略 模板方法 观察者，迭代，责任链，命令，备忘录模式，状态，访问者，中介者，解释器 |

###### 6.35 抽象工厂和工厂方法模式的区别？

|  |
| --- |
| 前者针对一个产品等级结构，一个抽象产品对象。 |

###### 6.36 什么是值传递和引用传递？

|  |
| --- |
| 传递后的对象改变时，是否影响被传递的对象。 |

###### 6.37 可以在static 环境中访问非static变量吗？

|  |
| --- |
| 不可，非static的变量可能还未加载 |

###### 6.38 Java 支持多继承么,为什么？

|  |
| --- |
| 不可，处于安全考虑。可通过结构和内部类来克服此弱点 |

###### 6.39 用最有效率的方法计算2乘以 8 ？

|  |
| --- |
| 2>>>3 |

###### 6.40 构造器是否可被重写？

|  |
| --- |
| 不能被继承，故不能被重写。 |

###### 6.41 char 型变量中能不能存贮一个中文汉字，为什么？

|  |
| --- |
| Char占两个字节，可以存一个汉字 |

###### 6.42 如何实现对象克隆？

|  |
| --- |
| 1.实现cloneable接口，重写clone()方法。  2.结合序列化完成深拷贝。 |

###### 6.43 object 中定义了哪些方法？

|  |
| --- |
| getClass() hansCode() equal() clone() toString() notify() wait() finalize() |

###### 6.44 hashCode 的作用是什么？ 389

|  |
| --- |
|  |

###### 6.45 for-each 与常规for 循环的效率对比

|  |
| --- |
| 一样 |

###### 6.46 写出几种单例模式实现， 懒汉模式和饿汉模式区别

|  |
| --- |
| 懒汉-线程不安全；懒汉-线程安全； 俄汉；双重校验锁。 |

###### 6.47 请列出 5 个运行时异常。

|  |
| --- |
| NullPointerException ，ArithmeticException，IndexOutOfBoundsException，ClassNotFoundException，IllegalArgumentException |

###### 6.48 2 个不相等的对象有可能具有相同的 hashcode 吗？

|  |
| --- |
| 可能 hash冲突 |

**6.49 访问修饰符 public,private,protected, 以及default 的区别？**

|  |
| --- |
| Public protected default private 当前类，同包 ，子类，其他类 |

###### 6.50 谈谈 final 在 java 中的作用？

|  |
| --- |
| 保护，线程安全 |

###### 6.51 java 中的 Math.round(-1.5) 等于多少呢？

|  |
| --- |
| -1 |

###### 6.52 String 属于基础的数据类型吗？

|  |
| --- |
| No |

###### 6.53 如何将字符串反转呢？

|  |
| --- |
| StingBuilder() |

###### 6.54 描述动态代理的几种实现方式，它们分别有什么优缺点

|  |
| --- |
| JDK ,cglib |

###### **6.55 在自己的代码中，如果创建一个java.lang.String 类 ,这个类是否可以被类加载器加载？为什么。**

|  |
| --- |
| 不可以，双亲委派 |

###### **6.56 谈谈你对java.lang.Object 对象中 hashCode 和equals 方法的理解。在什么场景下需要重新实现这两个方法**

|  |
| --- |
| 不写的话，会违背既定原则 |

###### 6.57 在 jdk1.5 中，引入了泛型， 泛型的存在是用来解决什

|  |
| --- |
| Java泛型( generics )是JDK5中引入的一个新特性，其本质是参数化类型，解决不确定具体对象类  型的问题。 |

###### 6.58 什么是序列化，怎么序列化， 反序列呢？

|  |
| --- |
| 1.序列化就是对象转字节码 2.反序列化就是字节码转对象 |

###### 6.59 java8 的新特性。

|  |
| --- |
| Lambda ，方法引用，默认方法， Optional 类 |

###### 6.60 匿名内部类是什么？如何访问在其外面定义的变量呢？ 394

##### Spring 面试题(一)

###### 1、一般问题 395

|  |
| --- |
|  |

###### 1.1、不同版本的 Spring Framework 有哪些主要功能？

|  |
| --- |
| Spring2.5:第一个支持注解，Spring3.0 Spring4.0支持java8 |

###### 1.2、什么是 Spring Framework？ 395

|  |
| --- |
| Spring 是 一 个 开 源 应 用 框 架 ， 旨 在 降 低 应 用 程 序 开 发 的 复 杂 度 。 |

###### 1.3、列举 Spring Framework 的优点。

|  |
| --- |
| 分层架构，按需索取，基于依赖注入和控制反转，操作简化，免费开源。 |

###### 1.4、Spring Framework 有哪些不同的功能？

|  |
| --- |
| 1.IOC和AOP， |

###### 1.5、Spring Framework 中有多少个模块，它们分别是什么？

|  |
| --- |
| SpringCore;Spring Bean;Spring Context; JDBC类的；web(web-servlet,web-socket,web-protlet) AOP;TEST; |

###### 1.6、什么是 Spring 配置文件？

|  |
| --- |
| XML ,yml等。 |

###### 1.7、Spring 应用程序有哪些不同组件？

|  |
| --- |
| 1.接口--定义功能； 2.Bean类，包含setter和getter函数等 3.AOP，面向切面 4.用户程序。 |

###### 1.8、使用Spring有哪些方式？

|  |
| --- |
| 1.SpringWeb程序 2.用作三分WEB框架 3.作为企业级java Bean |

###### 2、依赖注入(Ioc)

###### 2.1、什么是 Spring IOC 容器？

|  |
| --- |
| 创建对象，并管理他们的完整生命周期。 |

###### 2.2、什么是依赖注入？

|  |
| --- |
| 在依赖注入中，您不必创建对象，但必须描述如何创建它们。您不是直接在代码中将组件和服务连接在一起，而是描述配置文件中哪些组件需要哪些服务 。由IoC 容器将它们装配在一起 。 |

###### 2.3、可以通过多少种方式完成依赖注入？

|  |
| --- |
| 构造函数，setter，接口注入 |

###### 2.4、区分构造函数注入和setter 注入。

|  |
| --- |
| 构造函数：没有部分注入，不会覆盖setter属性，任意修改都会创建新实例，适用于很多属性。  Setter：相反 |

###### 2.5、spring中有多少种IOC容器？

|  |
| --- |
| BeanFactory - BeanFactory就像一个包含bean 集合的工厂类。它会在客户端要求时实例化bean。  ApplicationContext - ApplicationContext 接口扩展了BeanFactory接口 。它 在 BeanFactory 基础 上提 供了 一些 额外的功能。 |

###### 2.6、区分 BeanFactory 和 ApplicationContext。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | BeanFactory | ApplicationContext | | 它使用懒加载 | 它使用即时加载 | | 它使用语法显式提供资源对象 | 它自己创建和管理资源对象 | | 不支持国际化 | 支持国际化 | | 不支持基于依赖的注解 | 支持基于依赖的注解 | |

###### 2.7、列举 IoC 的一些好处。 399

|  |
| --- |
| 1.它将最小化应用程序中的代码量 2.它将使您的应用程序易于测试 3.它以最小的影响和最少的侵入机制促进松耦合。 4.它支持即时的实例化和延迟加载服务。 |

###### 2.8、Spring IoC 的实现机制。

|  |
| --- |
| Spring 中 的 IoC 的 实 现 原 理 就 是 工 厂 模 式 加 反 射 机制 。 |

###### 3、 Beans

###### 3.1、什么是 spring bean？

|  |
| --- |
| 1.它们是构成用户应用程序主干的对象 2.Bean 由 Spring IoC 容器管理 3.它们由 Spring IoC 容器实例化，配置，装配和管理 4.Bean 是基于用户提供给容器的配置元数据创建 |

###### 3.2、spring 提供了哪些配置方式？ 401

|  |
| --- |
| 基于XML |
| bean 所 需 的 依 赖 项 和 服 务 在 XML 格 式 的 配 置 文 件 中 指 定 。 这 些 配 置 文 件 通 常 包 含 许 多 bean 定 义 和 特 定 于 应 用 程 序 的 配 置 选 项 。 它 们 通 常 以 bean 标 签 开 头 。 例 如 ：  <bean id="studentbean" class="org.edureka.firstSpring.StudentBean"> <property name="name" value="Edureka"></property>  </bean> |
| 基于注解配置：您可 以通 过在 相关 的类 ， 方 法或 字段 声明 上使 用注 解， 将 bean 配置 为组 件类 本 身，而不是使用XML来描述bean装配 。 |
| <beans>  <context:annotation-config/>  <!-- bean definitions go here -- >  </beans> |
| 基于 Java API 配置 |
| Spring 的 Java 配置 是通 过使 用 [@Bean](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)和 [@Configuration](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)来实 现。  1 、[@Bean](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)注解 扮演 与 <bean/> 元素 相同 的角 色。  2 、[@Configuration](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)类允 许通 过简单地调用同一个类中的其他[@Bean](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)方法来定义bean间依赖关系。  例如 ：  @Configuration  public class StudentConfig {  @Bean  public StudentBean myStudent() {  return new StudentBean();  }  } |

###### 3.3、spring 支持集中 bean scope？

|  |
| --- |
| Singleton ； Prototype；**Request** |

###### 3.4、spring bean 容器的生命周期是什么样的？ 402

|  |
| --- |
| 1.Spring 容 器 根 据 配 置 中 的 bean 定 义 中 实 例 化 bean  2.Spring 使 用 依 赖 注 入 填 充 所 有 属 性 ， 如 bean 中 所 定 义 的 配 置 。  3. 如 果 bean 实 现BeanNameAware 接 口 ， 则 工 厂 通 过 传 递 bean 的 ID 来 调 用 setBeanName() 。  4. 如 果bean实现BeanFactoryAware接口，工厂通过传 递 自 身 的 实 例 来 调 用 setBeanFactory() 。  5. 如 果 存 在 与 bean 关 联 的 任 何 BeanPostProcessors， 则 调 用 preProcessBeforeInitialization() 方 法 。  6. 如 果 为 bean指定了init 方 法 ( <bean> 的 init - method 属性) ，那么将 调 用 它 。  7. 最 后 ， 如 果 存 在 与 bean 关 联 的 任 何 BeanPostProcessors， 则 将 调用 postProcessAfterInitialization() 方法 。  8. 如 果bean实现DisposableBean 接口 ， 当 spring 容器 关闭 时 ， 会调 用 destory() 。  9. 如果为 bean指定了destroy方法( <bean> 的 destroy - method 属性)，那么 将 调用 它。 |

###### 3.5、什么是 spring 的内部 bean？

|  |
| --- |
| 1.只有 将 bean 用作 另一 个 bean 的属 性时 ， 才 能将 bean 声明 为内 部 bean 。 |

###### 3.6、什么是 spring 装配

|  |
| --- |
| 1.当bean在Spring 容器中组合在一起时 ，它被 称为 装配 或 bean 装配 。Spring 容器 需要 知道需要什么bean以及容器应该如何使用 依赖 注入 来将 bean 绑定 在一 起， 同时 装配 bean。 |

###### 3.7、自动装配有哪些方式？

|  |
| --- |
| No,byName byType autodetect |

###### 3.8、自动装配有什么局限？

###### 4、注解

###### 4.1、什么是基于注解的容器配置 405

|  |
| --- |
| 不 使 用 XML 来 描 述 bean 装 配 ，开 发 人 员 通 过 在 相 关 的 类 ，方 法 或 字 段 声 明 上 使 用 注 解 将 配 置 移 动 到 组 件 类 本 身 。 它 可 以 作 为 XML 设 置 的 替 代 方 案 。 例 如 ：Spring 的 Java 配 置 是 通 过 使 用 [@](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**[Bean](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**和 [@](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**[Configuration](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**来 实 现 |

###### 4.2、如何在 spring 中启动注解装配？

|  |
| --- |
| 默 认 情 况 下 ， Spring 容 器 中 未 打 开 注 解 装 配 。 因 此 ， 要 使 用 基 于 注 解 装 配 ， 我 们 必 须 通 过 配 置 <context： annotation -config/> 元 素 在 Spring 配 置 文 件 中 启 用 它 。 |

###### 4.3、@Component, @Controller, @Repository,@Service有何区别？

|  |
| --- |
| [@](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**[Component](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**： 这 将 java 类 标 记 为 bean。它 是 任 何 Spring 管 理 组 件 的 通 用 构 造 型 。 spring 的组件扫描 机 制 现 在 可 以 将 其 拾 取 并 将 其 拉 入 应 用 程 序环境中 。[@Controller](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)：这将一 个类标记为Spring Web MVC控制器。标有它的Bean会自 动导入到IoC容器中 。  [@](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**[Service](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**： 此 注 解 是 组 件 注 解 的 特 化 。 它 不 会 对 [@Component](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)注 解 提 供 任 何 其 他 行 为 。 您 可 以 在 服 务 层 类 中 使 用[@Service](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)而 不 是 @Component ， 因 为 它 以 更 好 的 方 式 指 定 了 意 图 。  [@](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**[Repository](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**：这 个 注 解 是 具 有 类 似 用 途 和 功 能 的  [@](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**[Component](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)**注 解 的 特化 。 它 为 DAO 提 供 了 额 外 的 好 处 。 它 将 DAO 导 入 IoC 容 器 ， 并 使 未 经 检 查 的 异 常 有 资 格 转 换 为 Spring DataAccessException。 |

###### 4.4、@Required注解有什么用？

|  |
| --- |
| 应 用 于 bean 属 性 setter 方 法 。 此 注 解 仅 指 示 必 须 在 配 置 时 使 用 bean 定 义 中 的 显 式 属 性 值 或 使 用 自 动 装 配 填 充 受 影 响 的 bean 属 性 。 如 果 尚 未 填 充 受 影 响 的 bean 属 性 ， 则 容 器 将 抛 出 BeanInitializationException。 |

###### 4.5、@Autowired注解有什么用？

|  |
| --- |
| [@Autowired](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)可 以 更 准 确 地 控 制 应 该 在 何 处 以 及 如 何 进 行 自 动 装 配 。 此 注 解 用 于 在 setter 方 法 ， 构 造 函 数 ， 具 有 任 意 名 称 或 多 个 参 数 的 属 性 或 方 法 上 自 动 装 配 bean 。 默 认 情 况 下 ， 它 是 类 型 驱 动 的 注 入 。 |

###### 4.6、@Qualifier注解有什么用？

|  |
| --- |
| 当您 创建 多个 相同 类型的 bean 并希 望仅 使用 属性 装配 其中 一个 bean 时 ，您可 以使 用[@Qualifier](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)注解和[@Autowired](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)通过 指定 应该 装配 哪个确切的bean来消 除歧 义。 |

###### 4.7、@RequestMapping注解有什么用？

###### 5、数据访问 409

|  |
| --- |
| [@RequestMapping](https://www.yuque.com/yevxum/bd3ufw/ytt7gf/pdf?toc=false&key=exports%3Adocx%3Adoc%3A2456644%3A1566790188000&x-yuque-fc-token=xa-KwZfJsLj7JPG0Z6MoLA%3D%3D%7C5dmVbAUI-5hvwkn6L5Vz1aqe2KWsJfck6E1Tg2cTaZM%3D)注解用于将特定 HTTP 请求方法映射到将处理相应请求的 控制器中的特定类/方法。此注释可应用于两个级别： |

###### 5.1、spring DAO 有什么用？

|  |
| --- |
| Spring DAO 使 得 JDBC ， Hibernate 或 JDO 这 样 的 数 据 访 问 技 术 更 容 易 以 一  种 统 一 的 方 式 工 作 。 这 使 得 用 户 容 易 在 持 久 性 技 术 之 间 切 换 。 它 还 允 许 您 在 编 写 代 码 时 ， 无 需 考 虑 捕 获 每 种 技 术 不 同 的 异 常 。 |

###### 5.2、列举 Spring DAO 抛出的异常。

|  |
| --- |
|  |

###### 5.3、spring JDBC API 中存在哪些类？ 410

|  |
| --- |
| * JdbcTemplate ;SimpleJdbcTemplate |

###### 5.4、使用 Spring 访问 Hibernate 的方法有哪些？

|  |
| --- |
| 使 用 Hibernate 模 板 和 回 调 进 行 控 制 反 转 |

###### 5.5、列举 spring 支持的事务管理类型 411

|  |
| --- |
| 1.程 序 化 事 务 管 理 2.声 明 式 事 务 管 理 |

###### 5.6、spring 支持哪些 ORM 框架

###### 6、AOP

###### 6.1、什么是 AOP？

|  |
| --- |
| AOP(Aspect -Oriented Programming), 即面向切面编程,而AOP中的基本单元是Aspect(切面 ) |

###### 6.2、什么是 Aspect？

|  |
| --- |
| aspect 由 pointcount 和 advice 组 成 , 它 既 包 含 了 横 切 逻 辑 的 定 义 , 也 包 括 了 连 接 点 的 定 义 . |

###### 6.3、什么是切点(JoinPoint)

|  |
| --- |
| 在 Spring AOP 中 , join point 总是 方法 的执 行点 |

###### 6.4、什么是通知(Advice)？

|  |
| --- |
| JoinPoint 处的 Aspect 所采 取的 动作 称为 Advice |

###### 6.5、有哪些类型的通知(Advice)？ 412

|  |
| --- |
| Before ;After Returning;After (finally) ;Around |

###### 6.6、指出在 spring aop 中 concern 和 cross-cuttingconcern 的不同之处。

|  |
| --- |
|  |

###### 6.7、AOP 有哪些实现方式？ 413

|  |
| --- |
| 1.静态代理  2.动态代理 |

###### 6.8、Spring AOP and AspectJ AOP 有什么区别？ 413

|  |
| --- |
|  |

###### 6.9、如何理解 Spring 中的代理？

|  |
| --- |
| 将 Advice 应用 于目 标对 象后 创建 的对 象称 为代 理。 在客 户端 对象 的情 况下 ， 目  标对 象和 代理 对象 是相 同的 。  Advice + Target Object = Proxy |

###### 6.10 什么是编制（Weaving）

|  |
| --- |
| 为 了 创 建 一 个 advice 对 象 而 链 接 一 个 aspect 和 其 它 应 用 类 型 或 对 象 ， 称 为 编 织 (Weaving) 。 |

###### 7、 MVC

###### 7.1、Spring MVC 框架有什么用？

|  |
| --- |
| Spring Web MVC 框 架 提 供 模型 -视图 -控制 器 架 构 和 随 时 可 用 的 组 件 ， 用 于 开 发 灵 活 且 松 散 耦 合 的 Web 应 用 程 序 。 |

###### 7.2、描述一下 DispatcherServlet 的工作流程

|  |
| --- |
|  |

###### 7.3、介绍一下 WebApplicationContext

##### Spring 面试题(二)

###### 1、什么是 spring?

|  |
| --- |
| Spring 是 个 java 企 业 级 应 用 的 开 源 开 发 框 架 。 |

###### 2、使用 Spring 框架的好处是什么？

|  |
| --- |
| 1.轻量 2.控制反转 3.AOP 4.容器 5.MVC框架 6.事务管理 7.一场处理 |

###### 3、Spring 由哪些模块组成?

|  |
| --- |
| 1.core 2.Bean 3.Context 4.Expression Language 5.JDBC 6.ORM 7.Transaction 8.Web 9.web-servlet 10.web-Struts 11.Web-portlet |

###### 4、核心容器(应用上下文) 模块。

|  |
| --- |
| 这 是 基 本 的 Spring 模 块 ， 提 供 spring 框 架 的 基 础 功 能 ， BeanFactory 是 任 何 以 spring 为 基 础 的 应 用 的 核 心 。Spring框架建立在此模块之上,它使Spring成 为一个容器 。 |

###### 5、 BeanFactory – BeanFactory 实现举例。

|  |
| --- |
| Bean 工 厂 是 工 厂 模 式 的 一 个 实 现 ，提 供 了 控 制 反 转 功 能 ，用 来 把 应 用 的 配 置 和 依 赖 从 正 真 的 应 用 代 码 中 分 离 。 |

###### 6、XMLBeanFactory

|  |
| --- |
| 它 根据 XML 文件 中的 定义 加载 beans 。该容 器从 XML 文件 读取 配置 元数 据并 用它 去创 建一 个完 全配 置的 系统 或应 用 |

###### 7、解释 AOP 模块

|  |
| --- |
| AOP 模块 用于 发给 我们 的 Spring 应用 做面 向切 面的 开发 |

###### 8、解释 JDBC 抽象和 DAO 模块。

|  |
| --- |
| 通过 使用 JDBC 抽象 和 DAO 模块 ，保证 数据 库代 码的 简洁 ，并能 避免 数据 库资 源 错误 关闭 导致 的问 题 |

###### 9、解释对象/关系映射集成模块。 417

|  |
| --- |
|  |

###### 10、解释 WEB 模块。 417

|  |
| --- |
|  |

###### 12、Spring 配置文件 417

|  |
| --- |
|  |

###### 13、什么是 Spring IOC 容器？ 417

|  |
| --- |
|  |

###### 15、ApplicationContext 通常的实现是什么? 418

|  |
| --- |
|  |

###### 16、 Bean 工厂和 Application contexts 有什么区别？ 418

|  |
| --- |
|  |

###### 17、一个 Spring 的应用看起来象什么？ 418

|  |
| --- |
|  |

###### 18、什么是 Spring 的依赖注入？ 418

|  |
| --- |
|  |

###### 19、有哪些不同类型的 IOC (依赖注入)方式？ 418

|  |
| --- |
|  |

###### 20、哪种依赖注入方式你建议使用 ，构造器注入 ，还是 Setter方法注入？ 419

###### Spring Beans 419

|  |
| --- |
|  |

###### 21.什么是 Spring beans? 419

|  |
| --- |
|  |

###### 22、一个 Spring Bean 定义 包含什么？ 419

|  |
| --- |
|  |

###### 23、如何给 Spring 容器提供配置元数据? 419

|  |
| --- |
|  |

###### 24、你怎样定义类的作用域? 419

|  |
| --- |
|  |

###### 25、解释 Spring 支持的几种 bean 的作用域。 419

|  |
| --- |
|  |

###### 26、Spring 框架中的单例 bean 是线程安全的吗? 420

|  |
| --- |
|  |

###### 27、解释 Spring 框架中 bean 的生命周期。 420

|  |
| --- |
|  |

###### 28、哪些是重要的 bean 生命周期方法？你能重载它们吗？ 420

|  |
| --- |
|  |

###### 29、什么是 Spring 的内部 bean？ 421

|  |
| --- |
|  |

###### 30、在 Spring 中如何注入一个 java 集合？ 421

|  |
| --- |
|  |

###### 31、什么是 bean 装配? 421

|  |
| --- |
|  |

###### 32、什么是 bean 的自动装配？ 421

|  |
| --- |
|  |

###### 33、解释不同方式的自动装配 。 421

|  |
| --- |
|  |

###### 34.自动装配有哪些局限性 ? 421

|  |
| --- |
|  |

###### 35、你可以在 Spring 中注入一个 null 和一个空字符串吗？ 422

###### Spring 注解 422

|  |
| --- |
|  |

###### 36、什么是基于 Java 的 Spring 注解配置? 给一些注解的例子. 422

|  |
| --- |
|  |

###### 37、什么是基于注解的容器配置? 422

|  |
| --- |
|  |

###### 38、怎样开启注解装配？ 422

|  |
| --- |
|  |

###### 39、@Required注解 422

|  |
| --- |
|  |

###### 40、@Autowired注解 422

|  |
| --- |
|  |

###### 41、@Qualifier注解 422

###### Spring 数据访问 422

|  |
| --- |
|  |

###### 42.在 Spring 框架中如何更有效地使用 JDBC? 423

|  |
| --- |
|  |

###### 43、JdbcTemplate 423

|  |
| --- |
|  |

###### 44、Spring 对 DAO 的支持 423

|  |
| --- |
|  |

###### 45、使用 Spring 通过什么方式访问 Hibernate? 423

|  |
| --- |
|  |

###### 46、Spring 支持的 ORM 423

|  |
| --- |
|  |

###### 47.如何通过 HibernateDaoSupport 将 Spring 和 Hibernate结合起来？ 423

|  |
| --- |
|  |

###### 48、Spring 支持的事务管理类型 424

|  |
| --- |
|  |

###### 49、Spring 框架的事务管理有哪些优点？ 424

|  |
| --- |
|  |

###### 50、你更倾向用那种事务管理类型？ 425

###### Spring 面向切面编程(AOP) 425

|  |
| --- |
|  |

###### 51、解释 AOP 425

|  |
| --- |
|  |

###### 52、Aspect 切面 425

|  |
| --- |
|  |

###### 52、在 Spring AOP 中，关注点和横切关注的区别是什么？ 425

|  |
| --- |
|  |

###### 54、连接点 425

|  |
| --- |
|  |

###### 55、通知 425

|  |
| --- |
|  |

###### 56、切点 426

|  |
| --- |
|  |

###### 57、什么是引入? 426

|  |
| --- |
|  |

###### 58、什么是目标对象? 426

|  |
| --- |
|  |

###### 59、什么是代理? 426

|  |
| --- |
|  |

###### 60、有几种不同类型的自动代理？ 426

|  |
| --- |
|  |

###### 61、什么是织入。什么是织入应用的不同点？ 426

|  |
| --- |
|  |

###### 62、解释基于 XML Schema 方式的切面实现。 426

|  |
| --- |
|  |

###### 63、解释基于注解的切面实现 426

###### Spring 的 MVC 426

|  |
| --- |
|  |

###### 64、什么是 Spring 的 MVC 框架？ 427

|  |
| --- |
|  |

###### 65、 DispatcherServlet 427

|  |
| --- |
|  |

###### 66、WebApplicationContext 427

|  |
| --- |
|  |

###### 67、什么是 Spring MVC 框架的控制器？ 427

|  |
| --- |
|  |

###### 68、@Controller注解 427

|  |
| --- |
|  |

###### 69、@RequestMapping注解

##### 微服务 面试题

###### 1、您对微服务有何了解？

|  |
| --- |
| 微服务架构 |

###### 2、微服务架构有哪些优势？

|  |
| --- |
| 1.独立开发；独立部署；故障隔离；混合技术堆栈； |

###### 3。微服务有哪些特点？

|  |
| --- |
| 解耦；组件化；业务能力；自治；持续交付；责任；分散治理；敏捷； |

###### 4、设计微服务的最佳实践是什么？

|  |
| --- |
| 微服务访谈问题 |

###### 5、微服务架构如何运作？

|  |
| --- |
| 1.客户端 2.身份提供商 3.网关 4.今天太内容 5.管理-负载均衡 6.服务发现 ，7 内容交付网络 |

###### 6、微服务架构的优缺点是什么？

|  |
| --- |
| 1.故障排除 2.延时 3配置复杂 4一致性问题 |

###### 7、单片， SOA 和微服务架构有什么区别？

|  |
| --- |
|  |

###### 8、在使用微服务架构时，您面临哪些挑战？

|  |
| --- |
| 1.自动化组件 2.易感性 3.配置管理 4.调试 |

###### 9、SOA 和微服务架构之间的主要区别是什么？

|  |
| --- |
|  |

###### 10、微服务有什么特点？

|  |
| --- |
| 见3 |

###### 11、什么是领域驱动设计？

|  |
| --- |
|  |

###### 12、为什么需要域驱动设计(DDD)？

|  |
| --- |
|  |

###### 13、什么是无所不在的语言？

|  |
| --- |
| 过 |

###### 14、什么是凝聚力？

|  |
| --- |
| 模块 内部 元素 所属 的程 度被 认为 是凝聚 力 。 |

###### 15、什么是耦合？

|  |
| --- |
| 组件之间的依赖关系强度 |

###### 16、什么是 REST / RESTful 以及它的用途是什么？

|  |
| --- |
| 一 种帮 助计 算机 系统 通过 Internet 进行 通信 的架 构风 格。 |

###### 17、你对 Spring Boot 有什么了解？

|  |
| --- |
| 随着 新功 能的 增加 ， 弹簧 变得 越来 越复 杂 。 如果 必须 启动 新的 spring 项 目 ，则必 须添 加构 建路 径或 添加 maven 依赖 项 ，配置 应用 程序 服务 器 ，添加 spring 配置 。所 以一 切都 必须 从头 开始 。  Spring Boot 是解 决这 个问 题的 方法 。使 用 spring boot 可以 避免 所有 样板 代码 和配 置。 |

###### 18、什么是 Spring 引导的执行器？

|  |
| --- |
| Spring Boot 执行 程序 提供 了 restful Web 服务 ，以访 问生 产环 境中 运行 应用 程序 的当 前状 态。 在执 行器 的帮 助下 ， 您 可以 检查 各种指标并监控您的应用程 序。 |

###### 19、什么是 Spring Cloud？

|  |
| --- |
| Spring Cloud 为开 发人 员提 供了 快速 构建 分布 式系 统中 一些 常见 模式 的工 具(例如 配置管理 ，服 务发现， 断路 器，智能 路由，领导选举，分布式会话，集 群状 态) 。 |

###### 20、Spring Cloud 解决了哪些问题？ 435

|  |
| --- |
| 与分布式系统相关的复杂性 – 包括网络问题，延迟开销，带宽问题，安全问题。  处理服务发现的能力 – 服务发现允许集群中的进程和服务找到彼此并进行通信。  解决冗余问题 – 冗余问题经常发生在分布式系统中。  负载平衡 – 改进跨多个计算资源(例如计算机集群，网络链接，中央处理单元)的工作负载分布。  减少性能问题 – 减少因各种操作开销导致的性能问题。 |

###### 21、在 Spring MVC 应用程序中使用 WebMvcTest 注释有什么用处？

|  |
| --- |
| 执行 此单 元测 试时 ， 不 会启 动所 有其 他控 制器 和映 射。 |

###### 22。你能否给出关于休息和微服务的要点？

|  |
| --- |
| 您 可以 说 REST 是构 建微 服务 的媒 介。 |

###### 23、什么是不同类型的微服务测试？

|  |
| --- |
| 底层：面向技术测试；中间层：压力测试； 顶层测试：用例； |

###### 24、您对 Distributed Transaction 有何了解？

|  |
| --- |
|  |

###### 25、什么是 Idempotence 以及它在哪里使用？

|  |
| --- |
|  |

###### 26、什么是有界上下文？

|  |
| --- |
| 有界 上下 文是 域驱 动设 计的 核心 模式 。 DDD 战略 设计 部门 的重 点是 处理 大型 模型 和团 队。 DDD 通过 将大 型模 型划 分为 不同 的有 界上 下文 并明 确其 相互 关系 来处 理 大型 模型 |

###### 27、什么是双因素身份验证？

|  |
| --- |
|  |

###### 28、双因素身份验证的凭据类型有哪些？

|  |
| --- |
|  |

###### 29、什么是客户证书？

|  |
| --- |
| 客户 端系统用于向远程 服务 器发 出经 过身 份验 证的 请求 的一 种数 字证 书称 为客户 端证 书。 |

###### 30、PACT在微服务架构中的用途是什么？

|  |
| --- |
| PACT 是一 个开 源工 具， 允许 测试 服务 提供 者和 消费 者之 间的 交互 ， 与 合同 隔离 ， 从而 提高 微服 务集 成的 可靠 性。 |

###### 31、什么是 OAuth？

|  |
| --- |
| OAuth 代表 开放 授权 协议 。这允 许通 过在 HTTP 服务 上启 用客 户端 应用 程序 (例 如第 三方 提供 商 Facebook ， GitHub 等) 来访 问资 源所 有者 的资 源。 |

###### 32、康威定律是什么？

|  |
| --- |
| 任何设计系统的组织 (广泛定义)都将产生一 种设计 |

###### 33、合同测试你懂什么？

|  |
| --- |
| 合同 测试 是在 外部 服务 边界进行的测 试，用于验证 其是 否符 合消 费服 务预 期的 合同 。 |

###### 34、什么是端到端微服务测试？

|  |
| --- |
| 说端 到端 测试 是一 种测 试， 在特 定时 期后 测试 所有 东西 。 |

###### 35、Container 在微服务中的用途是什么？

|  |
| --- |
| 容器 是管 理基 于微 服务 的应 用程 序以 便单 独开 发和 部署 它们 的好 方法 |

###### 36、什么是微服务架构中的 DRY？

|  |
| --- |
| DRY 代表 不要 重复 自己 |

###### 37、什么是消费者驱动的合同(CDC)？

|  |
| --- |
| 这基 本上 是用 于开 发微 服务 的模 式， 以便 它们 可以 被外 部系 统使 用 |

###### 38、Web ， RESTful API 在微服务中的作用是什么？

|  |
| --- |
| RESTful API 基于 Web 的开 放网 络原 则， 为构 建微 服务 架 构的 各个 组件 之间 的接 口提 供了 最合 理的 模型 |

###### 39、您对微服务架构中的语义监控有何了解？

|  |
| --- |
| 将自 动化 测试 与监 控应 用程 序相 结合 ， 以 检测 业务失 败因 素。 |

###### 40、我们如何进行跨功能测试？

|  |
| --- |
| 过 |

###### 41、我们如何在测试中消除非决定论？

|  |
| --- |
| 过 |

###### 42、 Mock 或 Stub 有什么区别？

|  |
| --- |
| 过 |

###### 43、您对 Mike Cohn 的测试金字塔了解多少？ 442

|  |
| --- |
| Mike Cohn 提供 了一 个名 为 Test Pyramid 的模 型 。这描述了软件 开发所需 的自动化测试 类型 |

###### 44、 Docker 的目的是什么？

|  |
| --- |
| Docker 提供 了一 个可 用于 托管 任何 应用 程序 的容 器环 境 。在此 ，软件 应用 程序 和 支持 它的 依赖 项紧 密打 包在 一起 。 |

###### 45、什么是金丝雀释放？

|  |
| --- |
| 是一 种降 低在 生产 中引 入新 软件 版本 的风 险的 技术 |

###### 46、什么是持续集成(CI)？

|  |
| --- |
| 持续 集成 ( CI ) 是每 次团 队成 员提 交版 本控 制更 改时 自动 构建 和测 试代 码的 过程 |

###### 47、什么是持续监测？

|  |
| --- |
| 持续 监控 深入 监控 覆盖 范围 ， 从 浏览 器内 前端 性能 指标 ， 到 应用 程序 性能 ， 再 到 主机 虚拟 化基 础架 构指 标。 |

###### 48、架构师在微服务架构中的角色是什么？

|  |
| --- |
| 1.决定整个软件系统的布局 2.帮助确定组件的分区 3.为开发微服务的团队提供某些工具和技术的建议  4.提供技术治理，以便技术开发团队遵循微服务原则 |

###### 49、我们可以用微服务创建状态机吗？

|  |
| --- |
| 我们 知道 拥有 自己 的数 据库 的每 个微 服务 都是 一个 可独 立部 署的 程序 单元 ， 这 反 过来 又让 我们 可以 创建 一个 状态 机。 因此 ， 我 们可 以为 特定 的微 服务 指定 不同 的 状态 和事 件。例如 ， 我 们可 以定 义 Order 微服 务。 订单 可以 具有 不同 的状 态。 Order 状态 的转 换可 以是 Order 微服 务中 的独 立事 件。 |

###### 什么是微服务中的反应性扩展？

|  |
| --- |
| Reactive Extensions 也称 为 Rx。这 是一 种设 计方 法， 我们 通过 调用 多个 服务 来 收集 结果 ， 然 后编 译组 合响 应。 这些 调用 可以 是同 步或 异步 ， 阻 塞或 非阻 塞。 Rx 是分 布式 系统 中非 常流 行的 工具 ， 与 传统 流程 相反 。 |

##### Linux 面试题

###### 1、绝对路径用什么符号表示？当前目录、上层目录用什么表示？

###### 2、怎么查看当前进程？怎么执行退出？怎么查看当前路径？

|  |
| --- |
| Ps exit pwd |

###### **3、 怎么清屏？怎么退出当前命令？怎么执行睡眠？怎么查看当前用户 id？查看指定帮助用什么命令？**

|  |
| --- |
| Clear ctrl+c |

###### 4、Ls 命令执行什么功能？ 可以带哪些参数，有什么区别？

|  |
| --- |
| ls 执行 的功 能： 列出 指定 目录 中的 目录 ， 以 及文 件 |

###### 5、建立软链接(快捷方式)，以及硬链接的命令。

|  |
| --- |
| 软链 接： ln -s slink source  硬链 接： ln link source |

###### 6、目录创建用什么命令？创建文件用什么命令？复制文件用什么命令？ 445

|  |
| --- |
| Mkdir 如 touch vi |

###### 7、查看文件内容有哪些命令可以使用？

|  |
| --- |
| vi 文件 名 #编辑 方式 查看 ， 可 修改  cat 文件 名 #显示 全部 文件 内容  more 文件 名 #分页 显示 文件 内容  less 文件 名 #与 more 相似 ， 更 好的 是可 以往 前翻 页  tail 文件 名 #仅查 看尾 部， 还可 以指 定行 数  head 文件 名 #仅查 看头 部 ,还可 以指 定行 数 |

###### 8、随意写文件命令？怎么向屏幕输出带空格的字符串，比如”hello world”?

|  |
| --- |
| Vi :echo hello world |

###### 9、终端是哪个文件夹下的哪个文件？黑洞文件是哪个文件夹下的哪个命令？ 446

|  |
| --- |
| 终端 /dev/tty  黑洞 文件 /dev/null |

###### 10、移动文件用哪个命令？改名用哪个命令？

|  |
| --- |
| mv |

###### 11、复制文件用哪个命令？如果需要连同文件夹一块复制呢？

|  |
| --- |
| cp cp - r ？ ？ ？ ？ |

###### 12、**删除文件用哪个命令？如果需要连目录及目录下文件一块删除呢？删除空文件夹用什么命令？**

|  |
| --- |
| Rm -rf |

###### 13、 Linux 下命令有哪几种可使用的通配符？分别代表什么含义? 447

|  |
| --- |
| ? \* |

###### 14、用什么命令对一个文件的内容进行统计？ (行号、单词数、字节数) 447

|  |
| --- |
| wc 命令 - c 统计 字节 数 - l 统计 行数 - w 统计 字数 。 |

###### 15、Grep 命令有什么用？ 如何忽略大小写？ 如何查找不含该串的行? 447

|  |
| --- |
| 是一 种强 大的 文本 搜索 工具 ， 它能 使用 正则 表达 式搜 索文 本 ，并把匹配的行打印出来 。  grep [stringSTRING] filename grep [^string] filename |

###### 16、 Linux 中进程有哪几种状态？在 ps 显示出来的信息中，分别用什么符号表示的？ 448

|  |
| --- |
|  |

###### 17、怎么使一个命令在后台运行? 448

|  |
| --- |
| & |

###### 18、利用 ps 怎么显示所有的进程? 怎么利用 ps 查看指定进程的信息？ 448

|  |
| --- |
| ps -ef (system v 输出)  ps -aux bsd 格式输出  ps -ef | grep pid |

###### 19、哪个命令专门用来查看后台任务? 449

|  |
| --- |
| job - l |

###### 20、把后台任务调到前台执行使用什么命令?把停下的后台任务 449

|  |
| --- |
|  |

###### 21、终止进程用什么命令? 带什么参数? 449

|  |
| --- |
| kill [ - s <信 息 名 称 或 编 号 >][程 序 ] 或 kill [ - l <信 息 编 号 >] |

###### 22、怎么查看系统支持的所有信号？ 449

|  |
| --- |
| Kill -l |

###### 23、搜索文件用什么命令? 格式是怎么样的? 449

|  |
| --- |
| find <指 定 目 录> <指 定 条 件> <指 定 动 作> |

###### 24、查看当前谁在使用该主机用什么命令? 查找自己所在的终 450

|  |
| --- |
| Who am i who |

###### 25、使用什么命令查看用过的命令列表? 450

|  |
| --- |
| history |

###### 26、使用什么命令查看磁盘使用空间？ 空闲空间呢?

|  |
| --- |
| df -hl |

###### 27、使用什么命令查看网络是否连通?

|  |
| --- |
| netstat |

###### 28、使用什么命令查看 ip 地址及接口信息？

###### 29、查看各类环境变量用什么命令?

|  |
| --- |
| 查看 所有 env  查看 某个 ， 如 home： env $HOME |

###### 30、通过什么命令指定命令提示符?

|  |
| --- |
|  |

###### 31、查找命令的可执行文件是去哪查找的? 怎么对其进行设置 451

|  |
| --- |
|  |

###### 32、通过什么命令查找执行命令? 452

|  |
| --- |
|  |

###### 33、怎么对命令进行取别名？ 452

|  |
| --- |
| alias la='ls -a' |

###### 34、du 和 df 的定义，以及区别？ 452

|  |
| --- |
| du 显示 目录 或文 件的 大小  df 显示 每个 <文件 >所在 的文 件系 统的 信息 ， 默 认是 显示 所有 文件 系统 。 |

###### 35、awk 详解。

|  |
| --- |
|  |

###### 36、当你需要给命令绑定一个宏或者按键的时候，应该怎么做呢？

|  |
| --- |
|  |

###### 37、如果一个 linux 新手想要知道当前系统支持的所有命令的列表，他需要怎么做？

|  |
| --- |
|  |

###### 38、如果你的助手想要打印出当前的目录栈，你会建议他怎么做？ 454

|  |
| --- |
|  |

###### 39、你的系统目前有许多正在运行的任务，在不重启机器的条件下，有什么方法可以把所有正在运行的进程移除呢？

|  |
| --- |
|  |

###### 40、 bash shell 中的 hash 命令有什么作用？

|  |
| --- |
|  |

###### 41、哪一个 bash 内置命令能够进行数学运算。 455

|  |
| --- |
|  |

###### 42、怎样一页一页地查看一个大文件的内容呢？ 455

|  |
| --- |
|  |

###### 43、数据字典属于哪一个用户的？ 455

|  |
| --- |
|  |

###### 44、怎样查看一个 linux 命令的概要与用法？假设你在/bin 目录中偶然看到一个你从没见过的的命令，怎样才能知道它的作用 和用法呢？ 455

|  |
| --- |
|  |

###### 45、使用哪一个命令可以查看自己文件系统的磁盘空间配额呢？ 456

##### Spring Boot 面试题 456

###### 1、什么是 Spring Boot？

|  |
| --- |
| 多 年 来 ， 随 着 新 功 能 的 增 加 ， spring 变 得 越 来 越 复 杂 。如 果 必 须 启 动 一 个 新 的 Spring 项 目 ， 我 们 必 须 添 加 构 建 路 径 或 添 加 Maven 依 赖 关 系 ， 配 置 应 用 程 序 服 务 器 ， 添 加 spring 配 置 。 因 此 ， 开 始 一 个 新 的 spring 项 目 需 要 很 多 努 力 ， 因 为 我 们 现 在 必 须 从 头 开 始 做 所 有 事 情 。  Spring Boot 是 解 决 这 个 问 题 的 方 法 。 Spring Boot 已 经 建 立 在 现 有 spring 框 架 之 上 。 使 用 spring 启 动 ， 我 们 避 免 了 之 前 我 们 必 须 做 的 所 有 样 板 代 码 和 配 置 。 因 此 ， Spring Boot 可 以 帮 助 我 们 以 最 少 的 工 作 量 ， 更 加 健 壮 地 使 用 现 有 的 Spring 功 能 。 |

###### 2、Spring Boot 有哪些优点？

|  |
| --- |
| 1 、减 少 开 发 ， 测 试 时 间 和 努 力 。  2、使 用 JavaConfig 有 助 于 避 免 使 用 XML。  3、避 免 大 量 的 Maven 导 入 和 各 种 版 本 冲 突 。  4 、提 供 意 见 发 展 方 法 。  5 、通 过 提 供 默 认 值 快 速 开 始 开 发 。  没 有 单 独 的 Web 服 务 器 需 要 。 这 意 味 着 你 不 再 需 要 启 动 Tomcat ， Glassfish 或 其 他 任 何 东 西 。  7 、需 要 更 少 的 配 置 因 为 没 有 web .xml 文 件 。只 需 添 加 用 @ Configuration 注 释 的 类 ，然 后 添 加 用 @Bean 注 释 的 方 法 ，Spring 将 自 动 加 载 对 象 并 像 以 前 一 样 对 其 进 行 管 理 。您 甚 至 可 以 将 @Autowired 添 加 到 bean 方 法 中 ， 以 使 Spring 自 动 装 入 需 要 的 依 赖 关 系 中 。  8 、基 于 环 境 的 配 置 使 用 这 些 属 性 ，您 可 以 将 您 正 在 使 用 的 环 境 传 递 到 应 用 程 序 ： -Dspring.profiles.active = {enviornment} 。 在 加 载 主 应 用 程 序 属 性 文 件 后 ，Spring 将 在 (application{environment} . properties)中加 载 后 续 的 应 用 程 序 属 性 文 件 。 |

###### 3、什么是 JavaConfig？ 457

|  |
| --- |
| Spring JavaConfig 是 Spring 社 区 的 产 品 ，它 提供了配 置 Spring IoC 容 器 的 纯 Java 方 法 |

###### 4、如何重新加载 Spring Boot 上的更改，而无需重新启动服务器？

|  |
| --- |
| Spring-boot-devtools设置热启动嵌 入 式 tomcat 将 重 新 启 动 |

###### 5、Spring Boot 中的监视器是什么？ 457

|  |
| --- |
| Spring boot 监 视 器 可 帮 助 您 访 问 生 产 环 境 中 正 在 运 行 的 应 用 程 序 的 当 前 状 态 。 有 几 个 指 标 必 须 在 生 产 环 境 中 进 行 检 查 和 监 控 。 即 使 一 些 外 部 应 用 程 序 可 能 正 在 使 用 这 些 服 务 来 向 相 关 人 员 触 发 警 报 消 息 。 监 视 器 模 块 公 开 了 一 组 可 直 接 作 为 HTTP URL 访 问 的 REST端点来检查状态 。 |

###### 6、如何在 Spring Boot 中禁用 Actuator 端点安全性？

|  |
| --- |
| 默 认 情 况 下 ，所 有 敏 感 的 HTTP 端 点 都 是 安 全 的 ，只 有 具 有 ACTUATOR 角 色 的 用 户 才 能 访 问 它 们 |

###### 7、如何在自定义端口上运行 Spring Boot 应用程序？

|  |
| --- |
| application.properties 中 指 定 端 口 。 |

###### 8、什么是 YAML？

|  |
| --- |
| YAML 是 一 种 人 类 可 读 的 数 据 序 列 化 语 言 。 它 通 常 用 于 配 置 文 件 。 |

###### 9、如何实现 Spring Boot 应用程序的安全性？

|  |
| --- |
| 我 们 使 用 spring - boot - starter - security 依 赖 项 ， 并 且 必 须 添 加 安 全 配 置 。 它 只 需 要 很 少 的 代 码 。 配 置类将必须扩 展 WebSecurityConfigurerAdapter并覆盖其方法 。 |

###### 10、如何集成 Spring Boot 和 ActiveMQ？

|  |
| --- |
| 我 们 使 用依 赖 关 系 。 它 只 需 要 很 少 的 配 置 ， 并 且 不 需 要 样 板 代 码 。 |

###### 11、如何使用 Spring Boot 实现分页和排序？

|  |
| --- |
| 使 用 Spring Data -JPA 可 以 实 现 将 可 分 页 的传 递 给 存 储 库 方 法 |

###### 12、什么是 Swagger？你用 Spring Boot 实现了它吗？

|  |
| --- |
| wagger 广 泛 用 于 可 视 化 API |

###### 13、什么是 Spring Profiles？

|  |
| --- |
| Spring Profiles 允 许 用 户 根 据 配 置 文 件 (dev ， test ， prod 等 ) 来 注 册 bean 。 |

###### 14、什么是 Spring Batch？ 459

|  |
| --- |
| Spring Boot Batch 提 供 可 重 用 的 函 数 ，这 些 函 数 在 处 理 大 量 记 录 时 非 常 重 要 ，包 括 日 志/跟 踪 ， 事 务 管 理 ， 作 业 处 理 统 计 信 息 ， 作 业 重 新 启 动 ， 跳 过 和 资 源 管 理 |

###### 15、什么是 FreeMarker 模板？

|  |
| --- |
| FreeMarker 是 一 个 基 于 Java 的 模 板 引 擎 ，最 初 专 注 于 使 用 MVC 软件架构进行动态网页生成 。 |

###### 16、如何使用 Spring Boot 实现异常处理？ 459

|  |
| --- |
| 我 们 通过 实 现 一 个 ControlerAdvice 类 ， 来 处 理 控 制 器 类 抛 出 的 所 有 异 常 。 |

###### 17、您使用了哪些 starter maven 依赖项？

|  |
| --- |
| spring-boot-starter-activemq |

###### 18、什么是 CSRF 攻击？ 459

|  |
| --- |
| CSRF 代 表 跨 站 请 求 伪 造 |

###### 19、什么是 WebSockets？

|  |
| --- |
| WebSocket 是 一 种 计 算 机 通 信协议，通 过 单 个 TCP 连 接 提 供 全 双 工 通 信 信 道 。 |

###### 20、什么是 AOP？ 460

|  |
| --- |
| 将 这 些 横 切 关 注 与 业 务 逻 辑 分 开 是 面 向 方 面 编 程 (AOP) 的 地 方 。 |

###### 21、什么是 Apache Kafka？

|  |
| --- |
| Apache Kafka 是 一 个 分 布 式 发 布 - 订 阅 消 息 系 统 。 它 是 一 个 可 扩 展 的 ， 容 错 的 发 布 - 订 阅 消 息 系 统 ， 它 使 我 们 能 够 构 建 分 布 式 应 用 程 序 。 这 是 一 个 Apache 顶 级 项 目 。 Kafka 适 合 离 线 和 在 线 消 息 消 费 |

###### 22、我们如何监视所有 Spring Boot 微服务？

##### Spring Cloud 面试题

###### 1、什么是 Spring Cloud？

|  |
| --- |
| Spring cloud 流 应 用 程 序 启 动 器 是 基 于 Spring Boot 的 Spring 集 成 应 用 程 序 ， 提 供 与 外 部 系 统 的 集 成 。 Spring cloud Task， 一 个 生 命 周 期 短 暂 的 微 服 务 框 架 ， 用 于 快 速 构 建 执 行 有 限 数 据 处 理 的 应 用 程 序 。 |

###### 2、使用 Spring Cloud 有什么优势？

|  |
| --- |
| 1、与 分布 式系 统相 关的 复杂 性 -这种 开销 包括 网络问题 ，延 迟开 销，带宽问题，安全问题 。  2.服务发现 3.冗余 4.负载均衡 5.性能问题 6.部署复杂性。 |

###### 3、服务注册和发现是什么意思？ Spring Cloud 如何实现？

|  |
| --- |
| 配置变了后，主动发现而不用手动更改 |

###### 4、负载平衡的意义什么？

|  |
| --- |
| 负载均衡 |

###### 5、什么是 Hystrix？它如何实现容错？ 462

|  |
| --- |
| Hystrix 是一 个延 迟和 容错 库 |

###### 6、什么是 Hystrix 断路器？我们需要它吗？

|  |
| --- |
| employee -consumer 公开 服务 会引 发异 常。 在这 种情 况下 使用 Hystrix 我们 定义 了一 个回 退方 法 |

###### 7、什么是 Netflix Feign？它的优点是什么？

|  |
| --- |
| 我 们有 多个 应用 程序 使用 Spring Cloud Config 读取 属性 ， 而 Spring Cloud Config 从 GIT 读取 这些 属性 。 |

###### 8、什么是 Spring Cloud Bus？我们需要它吗？

##### RabbitMQ 面试题

###### 1、什么是 rabbitmq

|  |
| --- |
| 采 用 AMQP 高 级 消 息 队 列 协 议 的 一 种 消 息 队 列 技 术 ,最 大 的 特 点 就 是 消 费 并 不 需 要 确 保 提 供 方 存 在 ,实 现 了 服 务 之 间 的 高 度 解 耦 |

###### 2、为什么要使用 rabbitmq

|  |
| --- |
| 1.异步 削峰，解耦，2.持久化，3高并发 |

###### 3、使用 rabbitmq 的场景 468

|  |
| --- |
| 1.跨服务通讯，2.顺序消费，3.定时任务，4请求削峰 |

###### 4、如何确保消息正确地发送至 RabbitMQ？ 如何确保消息接收方消费了消息？

|  |
| --- |
| **发送 方确 认模 式**  **接收 方消 息确 认机 制**  如果 消费 者接 收到 消息，在 确认之前断开了连接或取消订阅，RabbitMQ 会认 为 消息 没有 被分 发， 然后 重新 分发 给下 一个 订阅 的消 费者 。(可能 存在消息 重复 消 费的 隐患 ，需 要去 重)  如果 消费 者接 收到 消息 却没 有确 认消 息， 连接 也未 断开 ， 则 RabbitMQ 认为 该消 费者 繁忙 ， 将 不会 给该 消费 者分 发更 多的 消息 。 |

###### 5.如何避免消息重复投递或重复消费？

|  |
| --- |
| 生产 者发 送的 消息 生成 一个 inner - msg - id， 作为去重的依据  在消 息消 费时 ，要求消息体中必须要有一个bizId作为去重的依 据，避免同一条消 息被 重复 消费 。 |

###### 6、消息基于什么传输？

|  |
| --- |
| 会 造成 性能 瓶 颈。 RabbitMQ 使用 信道 的方 式来 传输 数据 。信 道是 建立 在真 实的 TCP 连接 内的 虚拟 连接 ， 且 每条 TCP 连接 上的 信道 数量 没有 限制 。 |

###### 7、消息如何分发？

|  |
| --- |
| 若该 队列 至少 有一 个消 费者 订阅 ，消息 将以 循环 (round - robin) 的方 式发 送给 消 费者每 条消 息只 会分 发给 一个 订阅 的消 费者 (前 提是 消费 者能 够正 常处 理消 息 并进 行确 认) 。  通过 路由 可实 现多 消费 的功 能 |

###### 8、消息怎么路由？

|  |
| --- |
| 消息 提供 方 - >路由 - >一至 多个队列  路 由键 (routing key)  交换机：fanout，direct，topic(可用通配符) |

###### 9、如何确保消息不丢失？ 470

|  |
| --- |
| 发送端；存储端；消费端 |

###### 10、使用 RabbitMQ 有什么好处？ 470

|  |
| --- |
| 异步 解耦 削峰 |

###### 11、 RabbitMQ 的集群 470

|  |
| --- |
| 镜像集群模式 |

###### 12、 mq 的缺点 47

|  |
| --- |
| 系统可用性降低，系统复杂，一致性维妮塔 |

##### kafka 面试题

###### 1、如何获取 topic 主题的列表

|  |
| --- |
| bin/kafka-topics.sh -- list --zookeeper localhost:2181 |

###### 2、生产者和消费者的命令行是什么？ 471

|  |
| --- |
| bin/kafka-console-producer.sh -- broker - list 192 . 168 .43 .49:9092 --topic Hello - Kafka  接下 来每 个新 行就 是 输入 一条 新消 息。  消费 者接 受消 息：  bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic Hello - Kafka --from - beginning |

###### 3、consumer 是推还是拉？

|  |
| --- |
| producer 将消 息推 送到 broker ，consumer从broker拉取消息 |

###### 4、讲讲 kafka 维护消费状态跟踪的方法

|  |
| --- |
| Kafka 采 用 了 不 同 的 策 略 。Topic 被 分 成 了 若 干 分 区 ，每 个 分 区 在 同 一 时 间只 被 一 个 consumer 消 费 。 这 意 味 着 每 个分 区 被 消费 的 消 息 在 日志 中 的 位 置 仅 仅 是 一 个简 单 的 整 数 ：offset 。这 样 就 很 容 易 标 记 每个 分 区 消费 状 态 就 很 容易 了 ，仅 仅 需 要 一 个 整 数 而 已 。 这 样消 费 状 态的 跟 踪 就 很 简单 了 。 |

###### 5、讲一下主从同步\*\*

|  |
| --- |
| [Kafka](https://so.csdn.net/so/search?q=Kafka&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/honglei915/article/details/_blank)允许topic的分区拥有若干副本，判断存活，异步复制，leader选择，副本管理 |

###### 6、为什么需要消息系统， mysql 不能满足需求吗？

|  |
| --- |
| 1.解耦 2.冗余 3.扩展性 4.灵活性，峰值处理能力 5.可恢复性 6.顺序保证 7.缓冲 8.异步通讯 |

###### 7、Zookeeper 对于 Kafka 的作用是什么？

|  |
| --- |
| 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  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。 |

###### 8、数据传输的事务定义有哪三种？

|  |
| --- |
| 1.最多一次 2.最少一次 3.精确一次 |

###### 9、Kafka 判断一个节点是否还活着有那两个条件？

|  |
| --- |
| 1.心跳检测 2.时间范围内可同步数据 |

###### 10、Kafka 与传统MQ消息系统之间有三个关键区别

|  |
| --- |
| 1.持久化日志可重复被读和无限期保留 2.天然分布式系统，三高 3.支持实时流式处理 |

###### 11、讲一讲 kafka 的 ack 的三种机制

|  |
| --- |
| 0，1，-1 |

###### 13、消费者故障，出现活锁问题如何解决？

|  |
| --- |
| 出现 “ 活 锁 ”的 情 况， 是它持续的发送心跳 ，但是没有处理 。  消费 者提 供两 个配 置设 置来 控制 poll 循环：  max.poll.interval.ms，max . poll. records： |

###### 14、如何控制消费的位置

|  |
| --- |
| kafka 使 用 seek(TopicPartition, long)指 定 新 的 消 费 位 置 。 |

###### 15、 kafka 分布式(不是单机)的情况下，如何保证消息的顺序消费?

|  |
| --- |
| 同 一 个 partition 用 一 个 write ahead log 组 织 ， 所 以 可 以 保 证 FIFO 的 顺 序 。 不 同 partition 之 间 不 能 保 证 顺 序 。 但 是 绝 大 多 数 用 户 都 可 以 通 过 message key 来 定 义 ， 因 为 同 一 个 key 的 message 可以保 证只发 送 到 同 一 个partition。 |

###### 16、 kafka 的高可用机制是什么？

|  |
| --- |
| 回 答 出 kafka 的 系 统 特 点 ， leader 和 follower 的 关 系 ，消息读写的 顺 序即可 。 |

###### 17、kafka 如何减少数据丢失

|  |
| --- |
| Producer端：同步复制,消息发送重试  Consumer端：enable.auto.commit=false  关闭自动提交位移,在消息被完整处理之后再手动提交位移 |

###### 18、kafka 如何不消费重复数据？比如扣款，我们不能重复的扣。

##### 3 消息队列附加题

###### 3.1 什么是消息队列

|  |
| --- |
|  |

###### 3.2 消息队列有哪些使用场景。

|  |
| --- |
| 1.应用解耦 2 流量削峰 3 异步处理 4 消息通讯 5 远程调用 |

###### 3.3 消息队列如何解决消息丢失问题？

|  |
| --- |
| 1.生产者保证不丢消息：同步发送  2.存储端不丢消息：同步刷盘  3.消费阶段不丢消息：消费者执行完业务逻辑 ，再反馈会 Broker 说消费成功 |

###### 3.4 消息队列如何保证消息的顺序性。

|  |
| --- |
| 那还能怎么办才能保证消息的顺序性呢？将 M1 和 M2 发往同一个消费者 ，且 发送 M1 后 ，等到消费端 ACK 成功后 ，才发送 M2 就得了。 |

###### 3.5 消息队列有可能发生重复消费 ，如何避免 ，如何做到幂等？

|  |
| --- |
| 生产端为了保证消息的可靠性，它可能往 MQ 服务器重复发送消息 ，直到拿到成功 的 ACK。  如何避免：带唯一业务标记的 ，利用主 键或者唯一性索引， 每次处理业务， 先校验一下就好啦。 又或者用 redis 缓存 下业务标记， 每次看下是否处理过了 |

###### 3.6 如何处理消息队列的消息积压问题

|  |
| --- |
| 首先排除bug，优化逻辑。  临时紧急扩容：1修复consumer，2停掉consumer，新建topic,partion10倍于当前，10倍于当前的消费queue, 写一个临时分发数据的consumer程序部署，直接轮询10倍的queue. 4. 征用10倍的机器来部署 5.恢复之前的部署 |

###### 3.7 消息队列技术选型， Kafka 还是 RocketMQ ，还是 Rabbit MQ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  | Kafka | RocketMQ | RabbitMQ | | 单 机 吞 吐 量 | 17.3w/s | 11.6w/s | 2.6w/s ( 消 息 做 持 久 化) | | 开 发 语 言 | Scala/Java | Java | Erlang | | 主 要 维 护 者 | 基 于 topic ，按 照 topic 进行正则匹配 | 基 于 topic/messageTag ， 按照消息类型、 属性进行正 则匹配的发布订阅模式 | 提 供 了 4 种 ：direct, topic ,Headers 和 | | 持 久 化 | 支持大量堆积 | 支持大量堆积 | 支持少量堆积 | | 顺 序 消 息 | 支持 | 支持 | 不支持 | | 集 群 方 式 | 天 然 的 Leader- Slave ,无 状态 集群 ，每台服 务 器 既 是 Master 也是 Slave | 常用 多对’Master-Slave’ 模 式 ，开 源 版 本 需 手 动切 换 Slave 变成 Master | 支持简单集群 ，'复制’ 模式， 对高级集群模式 支持不好。 | | 稳定性 | 较差 | 一般 | 好 | |

###### 3.8 消息中间件如何做到高可用

|  |
| --- |
| Kafka:数据都会同步到其它机器上， 形成多个副本。然后所有的副本会选举一个leader 出来 ，让 leader 去跟生产和消费者打交道 ，其他副本都是 follower。写数据时，leader 负责把数据同步给所有的follower，读消息时，直接读leader 上的数据即可。挂了重新选举。 |

###### 3.9 如何保证数据一致性，事务消息如何实现

|  |
| --- |
|  |

###### 3.10 让你写一个消息队列， 该如何进行架构设计？

|  |
| --- |
| 1整体流程，2.RPC设计，3borker持久化优化 4.消费关系保存 5可靠性 6.高可用 7.消息事务 8.伸缩性 |

##### 4计算机网络

###### 4.1 说说 HTTP 常用的状态码及其含义？ 491

|  |
| --- |
|  |

###### 4.2 HTTP 常用的请求方式，区别和用途？ 492

|  |
| --- |
| Get POST PUT DELETE .. |

###### 4.3 请简单说一下你了解的端口及对应的服务？

|  |
| --- |
| 22:ssh服务 ，80 HTTP服务 |

###### 4.4 说下计算机网络体系结构

|  |
| --- |
| 物 链 网 运 （会 表）应 |

###### 4.4.1 OSI 七层模型

|  |
| --- |
| 物 链 网 运 （会 表）应 |

###### 4.4.2 TCP/IP 四层模型

|  |
| --- |
| 1 网络接口层(物 链) ，2 网际层(网) 3.传输层 4.应用层 |

###### 4.4.3 五层体系结构

|  |
| --- |
| 物 链 网 运 （会 表）应 |

###### 4.5 如何理解 HTTP 协议是无状态的

|  |
| --- |
| 每次HTTP请求都是独立的，无相关的，默认不需要保存上下文信息的。我们来看个便于理解的例子： |

###### 4.6 从浏览器地址栏输入 url 到显示主页的过程

|  |
| --- |
| 1.DSN解析 2.三次握手TCP连接 3.向server发送HTTP请求 4.服务器返回 5.浏览器渲染 6TCP四次挥手 |

###### 4.7 说下 HTTP 1.0 ，1.1 ，2.0 的区别

|  |
| --- |
| http1.0:默认短链接，可开启长链接;  http1.1引入持久链接 tcp不关可复用,分块传输编码,管道机制，同一个TCP，客户端同时发送多个请求  http2.0: 二进制协议 完全多路复用，报头压缩 |

###### 4.8 POST 和GET 有哪些区别？

|  |
| --- |
| 可以从数据包、编码方式、 请求参数、 收藏为书签、历史记录、安全性等几方面去回答哈。 |

###### 4.9 在交互过程中如果数据传送完了，还不想断开连接怎么办 ， 499

|  |
| --- |
| 在 HTTP 中响应体的 Connection 字段指定为 keep-alive 即可 |

###### 4.10 HTTP 如何实现长连接？在什么时候会超时？

|  |
| --- |
| TCP 长连接可以复用一个 TCP 连接， 来发起多次的 HTTP 请求 ，这样就可 以减少资源消耗 ，比如一次请求 HTML ，如果是短连接的话，可能还需要 请求后续的 JS/CSS。  **实现**：通过在头部 ( 请求和响应头 )设置 Connection 字段指定为 keep-alive ，HTTP1 .0 协议支持 ，但是是默认关闭的，从 HTTP 1.1 以后 ，连接默 认都是长连接。  **超时：** HTTP 一般会有 httpd 守护进程 ，里面可以设置 keep-alive timeout ，当 tcp 连接闲置超过这个时间就会关闭 ，也可以在 HTTP 的 header 里面设置超 时时间 |

###### 4.11 HTTP 与 HTTPS 的区别。

|  |
| --- |
| HTTPS= HTTP+SSL/TLS |

###### 4.12 Https 流程是怎样的？

|  |
| --- |
| 这道题实际上考察的知识点是 HTTPS 的工作流程 ，大家需要回答这几个 要点， **公私钥、数字证书、加密、对称加密、非对称加密。** |

###### 4.13 说说 HTTP 的状态码 ，301 和 302 的区别？

|  |
| --- |
| 301 ：( 永久性转移 ) ，302 ：( 暂时性转移 ) |

###### 4.14 说说什么是数字签名？ 什么是数字证书？ 503

|  |
| --- |
| 数字证书是指在互联网通讯中标志通讯各方身份信息的**一个数字认证** ，人们可 以在网上用它来识别对方的身份。它的出现，是为了避免身份被篡改冒充的比如 Https 的数字证书 ，就是为了避免公钥被中间人冒充篡改： 公钥和个人信息、数字签名共同构成数字证书 |

###### 4.15 对称加密与非对称加密有什么区别

|  |
| --- |
| 加密和解密的密钥是否相同 |

###### 4.16 说说 DNS 的解析过程？

|  |
| --- |
| 1.首先会查找浏览器的缓存 2.将请求发往给本地 DNS 服务器 3.本地 DNS 服务器向根域名服务器发送请求,根域名服务器返回负责 .com 的顶 级域名服务器的 IP 地址的列表。 4.本地 DNS 服务器再向其中一个负责 .com的顶级域名服务器发送一个请求 ，返回负责 .baidu 的权威域名服务器的 IP 地址列表。 |

###### 4.17 什么是 CSRF 攻击 ，如何避免

|  |
| --- |
| CSRF ，跨站请求伪造 ( 英文全称是 Cross-site request forgery )，是一种挟制用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。  **应对**：CSRF ，跨站请求伪造 ( 英文全称是 Cross-site request forgery )，是一种  挟制用户在当前已登录的 Web 应用程序上执行非本意的操作的攻击方法。 |

###### 4.18 聊聊五层计算机网络体系结构中， 每一层对应的网络协

|  |
| --- |
| 1.物理层：网线 2.链路层：PPP 3.网络层：IP icmp 4.传输层：TCP DUDP 5.应用层：HTTP FRTP |

###### 4.19 说说 WebSocket 与socket 的区别

|  |
| --- |
| **Socket** 其实就是等于 IP 地址 + 端口 + 协议。 Socket是一套标准 ，它完成了对 TCP/IP 的高度封装 ，屏蔽网络细节，以方便开发者更好地进行网络编程。  **WebSocket** 是一个持久化的协议 ，它是伴随 H5 而出的协议 ，用来解决http不支持持久化连接的问题 |

###### 4.20 什么是 DoS、DDoS、DRDoS 攻击？

|  |
| --- |
| **DOS: (Denial of Service)**,翻译过来就是拒绝服务, 一切能引起 DOS 行为的 攻击都被称为 DOS 攻击。最常见的 DoS 攻击就有**计算机网络宽带攻击、连通 性攻击。**  **DDoS: (Distributed Denial of Service),**翻译过来是分布式拒绝服务  **DRDoS: (Distributed Reflection Denial of Service)** ，中文是分布式反射 拒绝服务， 该方式靠的是发送大量带有被害者 IP 地址的数据包给攻击主机 ，然 后攻击主机对 IP 地址源做出大量回应 ，从而形成拒绝服务攻击。 |

###### 4.21 什么是 XSS 攻击 ，如何避免?

|  |
| --- |
| 它指的是恶意攻击者往 Web 页 面里插入恶意 html 代码 ，当用户浏览该页之时， 嵌入其中Web 里面的 html 代码会被执行，从而达到恶意攻击用户的特殊目的。 XSS 攻击一般分三种类型 ：  . 首先 ，就不能相信用户的输入， 对输入进行过滤 ，过滤标签等， 只允许合法值。  . HTML 转义  . 对于链接跳转 ，如 <a href="xxx" 等，要校验内容 ，禁止以script开头的非法链接。  . 限制输入长度等等 |

###### 4.23 Http 请求的过程与原理

|  |
| --- |
| 1.DNS解析 2.三次握手 3.传输信息 4.四次挥手 |

###### 4.24 forward 和 redirect 的区别？

|  |
| --- |
| 直接转发方式( Forward ) ，间接转发方式( Redirect ) |

###### 4.25 聊聊 SQL 注入？

|  |
| --- |
| 它通过在 web 应用接口传入一些特殊参数字符 ，来欺骗应用服务器，执行恶意的 SQL 命令，以达到非法获取系统信息的目的。它目前是黑客对数据库进行攻击的最 常用手段之一。 |

###### 4.26.1 SQL 注入是如何攻击的？

|  |
| --- |
|  |

###### 4.27.2 如何预防 SQL 注入问题 516

|  |
| --- |
| 1). 使用#{}而不是 ${}  2). 不要暴露一些不必要的日志或者安全信息 ，比如避免直接响应一些sql 异常信息。  3). 不相信任何外部输入参数，过滤参数中含有的一些数据库关键词关键词  4). 适当的权限控制 |

###### 4.28 请详细介绍一下TCP 的三次握手机制 517

|  |
| --- |
|  |

###### 4.29 TCP 握手为什么是三次 ，为什么不能是两次？不能是四 519

|  |
| --- |
| 1.第一次：SYN=1 seq=x 发送完后客户端进入**SYN\_SEND**状态  2.第二次：SYN=1，ACK=1，seq=y,ack=x+1,发送 完了之后，服务端进入**SYN\_RCV**状态  3.第三次：ACK=1，ack=y+1,seq=x+1发送完之后客户端进入**ESTABLISHED**状态，服务端收到后也ESTAABLISHED状态 |

###### 4.30 TCP 四次挥手过程?

|  |
| --- |
| 第一次：FIN=1，seq =u,发完之后客户端进入**FIN\_WAIT\_1**  第二次：ACK=1，ack=u+1,seq = v,发完后服务端进入**CLOSE\_WAIT**，客户端收到确认包进入**FIN\_WAIT\_2**  第三次：FIN=1，ACK=1，seq=w,ack=u+1;发送完之后，服务端进入**LAST\_ACK**状态，等着最后一个ack  第四次：ACK=1，seq=u+1,ack=w+1,客户端进入**TIME\_WAIT**状态，等待2MSL后关闭，服务端收到后CLOUSE状态 |

###### 4.31 TCP 挥手为什么需要四次呢？

|  |
| --- |
| 四次是最少能够完整传输数据后并关闭的次数 |

###### 4.32 TCP 四次挥手过程中，为什么需要等待 2MSL,才进入 525

|  |
| --- |
| 1.为了保证客户端发送的最后一个 ACK 报文段能够到达服务端。  2. 防止已失效的连接请求报文段出现在本连接中。 |

###### 4.33 说说 TCP 是如何确保可靠性的呢？

|  |
| --- |
| 三次握手和四次挥手，有状态：哪些数据被接收，可控制：超时重传，失序数据重传，丢弃重复数据 |

###### 4.34 说说 TCP 报文首部有哪些字段，其作用又分别是什么？

|  |
| --- |
| 16位端口号；32位序列号；32位确认号；4位头部长度；6位标识位。。。 |

###### 4.35 请简述 TCP 和 UDP 的区别 527

|  |
| --- |
| TCP 面向连接(如打电话需要先拨号)， UDP 面向无连接( 即发送数据前不需要 建立连接)。 |

###### 4.36 说说 TCP 的重传机制

|  |
| --- |
|  |

###### 4.37.1 超时重传 528

|  |
| --- |
| 在发送数据报文时，设定一个定时器， 每间隔一段时间间隔，如果没收到对方的 ACK 应答报文 ，就会重发该报文。 一般RTO 略大于 RTT |

###### 4.37.2 快速重传

|  |
| --- |
| 快速重传机制 ，它不以时间驱动 ，而是以数据驱动。它基于接收端的反馈信息 来引发重传。  但快速重传还可能会有个问题： ACK 只向发送端告知最大的有序报文段，到底 是哪个报文丢失了呢？并不确定 ！那到底该重传多少个包呢？  是重传 Seq3 呢？还是重传 Seq3、Seq4、Seq5、Seq6 呢？ 因为发送端并 不清楚这三个连续的 ACK3 是谁传回来的。 |

###### 4.37.3 带选择确认的重传 ( SACK )

|  |
| --- |
| 为了解决快速重传的问题： 应该重传多少个包? TCP 提供了SACK 方法 ( 带选 择确认的重传 ，Selective Acknowledgment )。 |

###### 4.37.4 D-SACK

|  |
| --- |
| D-SACK ，即 Duplicate SACK(重复 SACK)，在 SACK 的基础上做了一些 扩展 ，，主要用来告诉发送方，有哪些数据包自己重复接受了。 |

###### 4.38 聊聊 TCP 的滑动窗口

|  |
| --- |
| 就是接受方每次收到数据包 ，在发送确认报文的时候 ，同时告诉发送方，自己的缓存区还有多少空余空间， 缓冲区的空余空间， 我们就称之为接受窗口大小。这就是 win。  它是操作系统开辟的一个缓存空间。窗口大小值表示无需等待确认应答 ，而可以继续发送数据的最大值。 |

###### 4.38 说说 TCP 的流量控制？

|  |
| --- |
| TCP 提供一种机制可以让发送方根据接收端的实际接收能力控制发送的数据量 ， 这就是流量控制。 |

###### 4.39 TCP 的拥塞控制

|  |
| --- |
| 慢启动算法、 拥塞避免、拥塞发生、快速恢 复算法。  拥塞控制是作用于网络的， 防止过多的数据包注入到网络中， 避免出现网络负 载过大的情况。它的目标主要是最大化利用网络上瓶颈链路的带宽。它跟流量 控制又有什么区别呢？流量控制是作用于接收者的， 根据接收端的实际接收能 力控制发送速度 ，防止分组丢失的 |

###### 4.39.1 慢启动算法

|  |
| --- |
| 它表示 TCP 建立连接完成后 ，一开 始不要发送大量的数据，而是先探测一下网络的拥塞程度。 由小到大逐渐增加 |

###### 4.39.2 拥塞避免算法

|  |
| --- |
| 一般来说，慢启动阀值 ssthresh 是 65535 字节 ，cwnd 到达慢启动阀值后  . 每收到一个 ACK 时， cwnd = cwnd + 1/cwnd  . 当每过一个 RTT 时， cwnd = cwnd + 1 |

###### 4.39.3 拥塞发生

|  |
| --- |
| 当网络拥塞发生丢包时，会有两种情况： RTO 超时重传， 快速重传 |

###### 4.39.4 快速恢复

|  |
| --- |
| 快速重传和快速恢复算法一般同时使用。 快速恢复算法认为，还有 3个重复 ACK 收到，说明网络也没那么糟糕 ，所以没有必要像 RTO 超时那么强烈。 正如前面所说，进入快速恢复之前 |

###### 4.40 说说半连接队列和 SYN Flood 攻击的关系

|  |
| --- |
| SYN Flood 是一种典型的 DDos 攻击 ，它在短时间内， 伪造不存在的 IP 地址, 向服务器大量发起 SYN 报文。 |

###### 4.41 TCP 的粘包和拆包

|  |
| --- |
| 一 个完整的包可能会被 TCP 拆分成多个包进行发送 ，也有可能把多个小的包封装 成一个大的数据包发送 ，这就是所谓的 TCP 粘包和拆包问题。  要发送的数据小于 TCP 发送缓冲区的大小,TCP 将多次写入缓冲区的数据一次发送出去 将会发生粘包 ； |

###### 4.42 Nagle 算法与延迟确认 549

4.42.1 Nagle 算法

|  |
| --- |
| Nagle 算法就是为了尽可能发送大块数 据， 避免网络中充斥着许多小数据块。 |

4.42.2 延迟确认

|  |
| --- |
| 如果接受方刚接收到发送方的数据包 ，在很短很短的时间内， 又接收到第二个 包。那么请问接收方是一个一个地回复好点，还是合并一起回复好呢？ |

###### 4.43 IP地址有哪些分类？

|  |
| --- |
| 一般可以这么认为， IP 地址= 网络号+主机号。 |

###### 4.44 说下ARP协议的工作过程？

|  |
| --- |
| ARP 协议协议 ，Address Resolution Protocol ，地址解析协议 ，它是用 于实现 IP 地址到 MAC 地址的映射。 |

###### 4.45 有了 IP 地址 ，为什么还要用 MAC 地址 ？

|  |
| --- |
| 但计算机的 IP 地址可由用户自行更改， 管理起来就相对困难 ，而 MAC 地址不 可更改， 所以一般会把 IP 地址和 MAC 地址组合起来使用。 |

###### 4.46 TCP 和 UDP 分别对应的常见应用层协议有哪些？

|  |
| --- |
| TCP 的应用层协议有： HTTP、FTP、SMTP、TELNET、SSH  UDP 的应用层协议 ：DNS、TFTP、SNMP |

###### 4.47 聊聊保活计时器的作用

|  |
| --- |
| 除时间等待计时器外， TCP 还有一个保活计时器( keepalive timer)。客户已主动与服务器建立了TCP 连接。 但后来客户端的主机突然 发生故障。显然 ，服务器以后就不能再收到客户端发来的数据。 因此 ，应当有 措施使服务器不要再白白等待下去。这就需要使用保活计时器了。 |

###### 4.48 聊聊 ARP 协议 554

|  |
| --- |
| ARP 协议 ，地址解析协议 ，是一个由 IP 地址获取 MAC 物理地址的 TCP/IP协议 |

###### 4.49 forward 和 redirect 的区别？ 555

|  |
| --- |
| 见上 |

###### 4.50 URI 和 URL 的区别 556

|  |
| --- |
| URI ，全称是 Uniform Resource Identifier) ； URL ，全称是 Uniform Resource Location) ， |

###### 4.51 Session 和 Cookie 的区别。

|  |
| --- |
| Cookie 是保存在客户端的一小块文本串的数据。  session 指的就是服务器和客户端一次会话的过程。 Session 对象存储着特定用户会话所需的属性及配置信息。 |

##### 7. JVM专栏背面试题

###### 7.1 什么情况下会发生栈内存溢出。

|  |
| --- |
| 栈：线程私有 xss,线程请求的栈深度大于虚拟机所允许的最大深度 ，将抛出 StackOverflowError，动态扩展失败报outOfMemory |

###### 7.2 详解 JVM 内存模型 560

|  |
| --- |
| 21fc636989cd46f18fcb91d97f512f56 |

###### 7.3 JVM 内存为什么要分成新生代 ，老年代 ，持久代。新生代

###### 1 ) 共享内存区划分 561

|  |
| --- |
| 1共享内存区 = 持久带 + 堆；2持久带 = 方法区 + 其他；3Java 堆 = 老年代 + 新生代；4新生代 = Eden + S0 + S1 |

###### 2 )一些参数的配置 561

|  |
| --- |
| – XX:NewRatio； –XX:SurvivorRatio MaxTenuringThreshold |

###### 3)为什么要分为 Eden 和 Survivor?为什么要设置两个 Survivor

|  |
| --- |
| 1.标记复制算法，多次筛选 2.让GC大多发生在新生代 3.解决垃圾碎片 |

###### 7.4 JVM 中一次完整的 GC 流程是怎样的， 对象如何晋升到老

|  |
| --- |
| 1.大对象直接进入老年代  2.当Eden满了出发Minor GC，存货的进入survivor,达到次数进入老年代，，老年代满了进行FULL GC。  3.Major GC发生在老年代的 GC，清理老年区，经常会伴随至少一次Minor GC，比Minor GC慢10倍以上 |

###### 7.5 你知道哪几种垃圾收集器， 各自的优缺点，重点讲下 cms 563

1 )几种垃圾收集器：

|  |
| --- |
| 1.新生代：Serial 收集器；ParNew 收集器；Parallel Scavenge 收集器（都是复制算法）  2.老年代：Serial Old 收集器；Parallel Old 收集器；CMS(Concurrent Mark Sweep) 收集器(标记整理)  3.G1：标记整理，初始标记，并发标记， 最终标记，筛选标记。不会产生空间碎片，可以精确地控制停顿 |

2 ) CMS 收集器和 G1 收集器的区别：

|  |
| --- |
| CMS 收集器以最小的停顿时间为目标的收集器 ；（标记清除）  G1 收集器可预测垃圾回收的停顿时间；（标记整理） |

###### 7.6 JVM 内存模型的相关知识了解多少 ，比如重排序，内存屏 564

1 )Java 内存模型图： 见上

2 ) 指令重排序：优化执行顺序

3 )内存屏障：**用于控制特定条件下的重排序和 内存可见性问题。LoadLoad**

4 ) happen-before

|  |
| --- |
| 单线程的，线程启动的，线程中断的，线程终结的，对象创建的。 |

###### 7.7 简单说说你了解的类加载器，可以打破双亲委派么，怎么打破

1) 什么是类加载器？

|  |
| --- |
| 类加载器 就是根据指定全限定名称将 class 文件加载到 JVM 内存， 转为 Class 对象。包括启动类加载器，其他类加载器，扩展类加载器，应用程序类加载器 |

2 ) 双亲委派模型

|  |
| --- |
| 如果一个类加载器收到类加载的请求 ，它首先不会自己去尝试加载这个类，而 是把这个请求委派给父类加载器完成。 |

3 )为什么需要双亲委派模型？

|  |
| --- |
| 安全，防止冲突 |

4 )怎么打破双亲委派模型？

|  |
| --- |
| 打破双亲委派机制则不仅要继承 ClassLoader 类，还要重写 loadClass 和 findClass 方法。 |

###### 7.8 说说你知道的几种主要的 JVM 参数

1 ) 堆栈配置相关

|  |
| --- |
| java -Xmx3550m -Xms3550m -Xmn2g -Xss128k  -XX:MaxPermSize=16m -XX:NewRatio=4 -XX:SurvivorRatio=4 -XX:MaxTenuringThreshold=15 |

2 ) 垃圾收集器相关

|  |
| --- |
| -XX:+UseParallelGC  -XX:ParallelGCThreads=20  -XX:+UseConcMarkSweepGC  -XX:CMSFullGCsBeforeCompaction=5  -XX:+UseCMSCompactAtFullCollection ： |

3 ) 辅助信息相关

|  |
| --- |
| -XX:+PrintGC  -XX:+PrintGCDetails |

###### 7.9 怎么打出线程栈信息。

|  |
| --- |
| 可以说一下 jps ，top，jstack 这几个命令，再配合一次排查线上问题进行解答。 |

7.10 强引用、 软引用、 弱引用、 虚引用的区别？

|  |
| --- |
| 不清理，不够时扫到清理，发现就清理，弥留之际 |

##### 8 OPPO面试8.1 聊聊你印象最深刻的项目，或者做了什么优化。 574

|  |
| --- |
| SQL:1批量操作，2.异步操作 3.使用缓存 4.优化逻辑，5.优化SQL 6.压缩传输类容 7.MQ解耦.8.产品沟通  [记一次接口性能优化实践总结：优化接口性能的八个建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488004&idx=1&sn=00840efd9c0bd0a7f172b59eb2ca130f&chksm=cf21cd2df856443bf21d8e09cfe5c8452ecaf82e3c2210fca3b28829ded04defddcf63c0a59b&token=162724582&lang=zh_CN&scene=21#wechat_redirect)  [工作四年，分享](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect) **[50](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect)** [个让你代码更好的](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect) [小建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect)  [看一遍就理解：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)**[order](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)****[by](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)** [详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)  [看一遍就理解：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)**[group](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)****[by](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)** [详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)  [实战！聊聊如何解决](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd) **[MySQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd)** [深分页问题](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd)  [后端程序员必备：书写高质量](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [的](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[30](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [条建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)  [阿里一面，给了几条](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)**[，问需要执行几次树搜索操作？](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)  [生产问题分析！](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)**[delete](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)****[in](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [子查询不走索引？！](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)  [面试官问如何优化慢](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247498342&idx=1&sn=1171168d3475dd5c46da40341731960f&chksm=cf22254ff855ac595c7b35264704be19262d1f31223fdbc25eafb398f32c92d3cc031a5422a5&token=2044040586&lang=zh_CN#rd) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247498342&idx=1&sn=1171168d3475dd5c46da40341731960f&chksm=cf22254ff855ac595c7b35264704be19262d1f31223fdbc25eafb398f32c92d3cc031a5422a5&token=2044040586&lang=zh_CN#rd)**[？](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247498342&idx=1&sn=1171168d3475dd5c46da40341731960f&chksm=cf22254ff855ac595c7b35264704be19262d1f31223fdbc25eafb398f32c92d3cc031a5422a5&token=2044040586&lang=zh_CN#rd) |

###### 8.2 你项目提到分布式锁，你们是怎么使用分布式锁的?

|  |
| --- |
| [《七种方案！探讨](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488142&idx=1&sn=79a304efae7a814b6f71bbbc53810c0c&chksm=cf21cda7f85644b11ff80323defb90193bc1780b45c1c6081f00da85d665fd9eb32cc934b5cf&token=162724582&lang=zh_CN&scene=21#wechat_redirect) **[Redis](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488142&idx=1&sn=79a304efae7a814b6f71bbbc53810c0c&chksm=cf21cda7f85644b11ff80323defb90193bc1780b45c1c6081f00da85d665fd9eb32cc934b5cf&token=162724582&lang=zh_CN&scene=21#wechat_redirect)** [分布式锁的正确使用姿势》](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488142&idx=1&sn=79a304efae7a814b6f71bbbc53810c0c&chksm=cf21cda7f85644b11ff80323defb90193bc1780b45c1c6081f00da85d665fd9eb32cc934b5cf&token=162724582&lang=zh_CN&scene=21#wechat_redirect)  [《美团二面：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490243&idx=1&sn=ff11c3aab9ada3b16d7f2b57c846d567&chksm=cf21c5eaf8564cfc59e3d0d56fd02b0f5513015005f498381be4d12db462442a49aabe4159ef&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)**[Redis](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490243&idx=1&sn=ff11c3aab9ada3b16d7f2b57c846d567&chksm=cf21c5eaf8564cfc59e3d0d56fd02b0f5513015005f498381be4d12db462442a49aabe4159ef&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [与](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490243&idx=1&sn=ff11c3aab9ada3b16d7f2b57c846d567&chksm=cf21c5eaf8564cfc59e3d0d56fd02b0f5513015005f498381be4d12db462442a49aabe4159ef&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[MySQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490243&idx=1&sn=ff11c3aab9ada3b16d7f2b57c846d567&chksm=cf21c5eaf8564cfc59e3d0d56fd02b0f5513015005f498381be4d12db462442a49aabe4159ef&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [双写一致性如何保证？》](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490243&idx=1&sn=ff11c3aab9ada3b16d7f2b57c846d567&chksm=cf21c5eaf8564cfc59e3d0d56fd02b0f5513015005f498381be4d12db462442a49aabe4159ef&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) |

###### 8.3 常见分布式事务解决方案

|  |
| --- |
| 1. 2PC 2.TCC 3.本地消息表 4.最大努力通知 5.seata  [看一遍就理解：分布式事务详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247498358&idx=1&sn=aa6c7ceb61b73267d68d1b4fb7ccc2ed&chksm=cf22255ff855ac495861d57df276517e89779006267fa8413fe925cc15b0c3e0b0f1b1a5675e&token=2044040586&lang=zh_CN#rd)  **[1.4](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488144&idx=1&sn=3493379523f678fc46a558f814551468&chksm=cf21cdb9f85644af4b09ac499ab16a5ae2b9d313d853b60fadcb9c888ac710b3df4e93b55d52&token=2044040586&lang=zh_CN#rd)****[w](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488144&idx=1&sn=3493379523f678fc46a558f814551468&chksm=cf21cdb9f85644af4b09ac499ab16a5ae2b9d313d853b60fadcb9c888ac710b3df4e93b55d52&token=2044040586&lang=zh_CN#rd)** [字，](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488144&idx=1&sn=3493379523f678fc46a558f814551468&chksm=cf21cdb9f85644af4b09ac499ab16a5ae2b9d313d853b60fadcb9c888ac710b3df4e93b55d52&token=2044040586&lang=zh_CN#rd)**[25](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488144&idx=1&sn=3493379523f678fc46a558f814551468&chksm=cf21cdb9f85644af4b09ac499ab16a5ae2b9d313d853b60fadcb9c888ac710b3df4e93b55d52&token=2044040586&lang=zh_CN#rd)** [张图让你彻底掌握分布式事务原理](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488144&idx=1&sn=3493379523f678fc46a558f814551468&chksm=cf21cdb9f85644af4b09ac499ab16a5ae2b9d313d853b60fadcb9c888ac710b3df4e93b55d52&token=2044040586&lang=zh_CN#rd)  [后端程序员必备：分布式事务基础篇](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487969&idx=1&sn=1a7c255439810aa12d0417a69c709bce&chksm=cf21cec8f85647dee38af93bb6747fc4e597c9918ad83f5ad30fe726f2918e017e390b2b7413&token=2044040586&lang=zh_CN#rd) |

###### 8.4 你们的接口幂等是如何保证的？

|  |
| --- |
| . select+insert+主键/唯一索引冲突  . 直接 insert + 主键/ 唯一索引冲突  . 状态机幂等  . 抽取防重表  . token 令牌  . 悲观锁(如 select for update ，很少用)  . 乐观锁  . 分布式锁 |

###### 8.5 你们的 mySQL 架构是怎样的？ 579

8.5.1 双机主备:简单，自动切换；主机工作压力大

8.5.2 一主一从：提高并发，易切换，简单；并发不太够

8.5.3 一主多从：多从库可读，分担压力，读并发高；写并发不高

8.5.4 MariaDB 同步多主机集群：负载均衡，高可用，任意节点可读写；大事务容易卡死，影响其他节点；

8.5.5 数据库中间件 583

|  |
| --- |
| mycat 分片存储， 每个分片配置一主多从的集群。.优点：解决高并发高数据量的高可用方案  缺点： 维护成本比较大。 |

###### 8.6 常见的索引结构有？哈希表结构属于哪种场景？

|  |
| --- |
| 哈希表、有序数组和搜索树哈希表这种结构适用于只有等值查询的场景  . 有序数组适合范围查询，用二分法快速得到，时间复杂度为 O(log(N))。查询还好 ， 如果是插入， 就得挪动后面所有的记录， 成本太高。 因此它一般只适用静态存储引 擎，比如保存 2018 年某个城市的所有人口信息。  . B+树适合范围查询， 我们 一般建的索引结构都是 B+树。。 |

###### 8.7 给你ab,ac,abc 字段，你是如何加索引的？

|  |
| --- |
| 这个最左前缀可以是联合索引的最左 N 个字段。比如组合索引 ( a,b,c ) 可以相 |

###### 8.8 数据库隔离级别是否了解？你们的数据库默认隔离级别是？

|  |
| --- |
| 读未提交，读已提交，可重复度，串行化 |

8.8.1 为什么 MySQL 的默认隔离离别是 RR?

|  |
| --- |
| RR 隔离级别下， 更新数据的 时候不仅对更新的行加行级锁，还会加间隙锁 ( gap lock ) |

8.8.2 我们的数据库隔离级别最后选的是读已提交( RC )。

|  |
| --- |
| 提高并发，减少死锁。 |

###### 8.9 RR 隔离级别实现原理，它是如何解决不可重复读的？ 587

8.9.1 什么是不可重复读：

|  |
| --- |
| 事务 A 被事务 B 干扰到了 ！在事务 A 范围内， 两个相同的查询， 读取同一条记 录， 却返回了不同的数据，这就是不可重复读。 |

8.9.2 undo log 版本链 + Read View 可见性规则

|  |
| --- |
| RR 隔离级别实现原理， 就是 MVCC多版本并发控制 ，而 MVCC 是是通过 Read View+ Undo Log 实现的，Undo Log 保存了历史快照 ，Read View 可见性规则帮助判断当前版本的数据是否可见。 |

8.9.3 RR 如何解决不可重复读

|  |
| --- |
| 1. 获取事务自己的版本号， 即事务 ID  2. 获取 Read View  3. 查询得到的数据， 然后 Read View 中的事务版本号进行比较。  4. 如果不符合 Read View 的可见性规则， 即就需要 Undo log 中历史快照;  5. 最后返回符合规则的数据 |

###### 8.10 你们项目使用了 RocketMQ 对吧？那你知道如何保证

8.10.1 生产者保证不丢消息：见上

8.10.2 存储端不丢消息：见上

8.10.3 消费阶段不丢消息 ：见上

###### 8.11 事务消息是否了解？场景题：比如下单清空购物车 ，你是如何设计的？

|  |
| --- |
| 1. 生产者产生消息 ，发送一条半事务消息到 MQ 服务器  2. MQ 收到消息后 ，将消息持久化到存储系统 ，这条消息的状态是待发送状态。  3. MQ 服务器返回 ACK 确认到生产者 ，此时 MQ 不会触发消息推送事件  4. 生产者执行本地事务  5. 如果本地事务执行成功 ，即 commit 执行结果到 MQ 服务器 ；如果执行失败 ，发 送 rollback。  6. 如果是正常的 commit ，MQ 服务器更新消息状态为可发送 ；如果是 rollback ，即 删除消息。  7. 如果消息状态更新为可发送 ，则 MQ 服务器会 push 消息给消费者。消费者消费完 就回ACK。  8. 如果 MQ 服务器长时间没有收到生产者的commit 或者 rollback ，它会反查生产 者 ，然后根据查询到的结果执行最终状态。 |

###### 8.12 如何快速判断一个数是奇数还是偶数，除开对 2 取余呢。

|  |
| --- |
| X&1 |

###### 8.13 Spring 声明式事务原理？哪些场景事务会失效？

8.13.1 声明式事务原理

|  |
| --- |
| spring 声明式事务， 即@Transactional,它可以帮助我们把事务开启、提交或者回滚的操作， 通过 Aop 的方式进行管理。**[Spring](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497609&idx=1&sn=4a1e5c567961ba8ca1e98960a38ce56e&chksm=cf2228a0f855a1b6ab227e7d941b10c7ee97f3fb4b5284183973d6b1dc0c563acf5db07e62d1&token=2044040586&lang=zh_CN#rd)** [中的](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497609&idx=1&sn=4a1e5c567961ba8ca1e98960a38ce56e&chksm=cf2228a0f855a1b6ab227e7d941b10c7ee97f3fb4b5284183973d6b1dc0c563acf5db07e62d1&token=2044040586&lang=zh_CN#rd)**[@Transactional](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497609&idx=1&sn=4a1e5c567961ba8ca1e98960a38ce56e&chksm=cf2228a0f855a1b6ab227e7d941b10c7ee97f3fb4b5284183973d6b1dc0c563acf5db07e62d1&token=2044040586&lang=zh_CN#rd)** [实现原理](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497609&idx=1&sn=4a1e5c567961ba8ca1e98960a38ce56e&chksm=cf2228a0f855a1b6ab227e7d941b10c7ee97f3fb4b5284183973d6b1dc0c563acf5db07e62d1&token=2044040586&lang=zh_CN#rd) |

8.13.2 spring 声明式事务哪些场景会失效

|  |
| --- |
| . 方法的访问权限必须是 **public** ，其他 private 等权限 ，事务失效  . 方法被定义成了**final** 的，这样会导致事务失效。  . 在**同一个类**中的方法直接内部调用，会导致事务失效。  . 一个方法如果**没交给spring 管理**， 就不会生成spring 事务。  . **多线程调用**， 两个方法不在同一个线程中， 获取到的数据库连接不一样的。  . **表的存储引擎不支持事务**  . 如果自己 **try.. .catch 误吞了异常**，事务失效。  . **错误的传播**  [聊聊](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247494570&idx=2&sn=17357bcd328b2d1d83f4a72c47daac1b&chksm=cf223483f855bd95351a778d5f48ddd37917ce2790ebbbcd1d6ee4f27f7f4b147f0d41101dcc&token=2044040586&lang=zh_CN#rd) **[spring](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247494570&idx=2&sn=17357bcd328b2d1d83f4a72c47daac1b&chksm=cf223483f855bd95351a778d5f48ddd37917ce2790ebbbcd1d6ee4f27f7f4b147f0d41101dcc&token=2044040586&lang=zh_CN#rd)** [事务失效的](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247494570&idx=2&sn=17357bcd328b2d1d83f4a72c47daac1b&chksm=cf223483f855bd95351a778d5f48ddd37917ce2790ebbbcd1d6ee4f27f7f4b147f0d41101dcc&token=2044040586&lang=zh_CN#rd) **[12](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247494570&idx=2&sn=17357bcd328b2d1d83f4a72c47daac1b&chksm=cf223483f855bd95351a778d5f48ddd37917ce2790ebbbcd1d6ee4f27f7f4b147f0d41101dcc&token=2044040586&lang=zh_CN#rd)** [种场景，太坑了](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247494570&idx=2&sn=17357bcd328b2d1d83f4a72c47daac1b&chksm=cf223483f855bd95351a778d5f48ddd37917ce2790ebbbcd1d6ee4f27f7f4b147f0d41101dcc&token=2044040586&lang=zh_CN#rd) |

8.14 你们是微服务架构嘛？如果你来设计一个类似淘宝的系

|  |
| --- |
| 可以按业务领域 ，把用户、社区、 商品信息、消息等模块等划分。  . 单 一功能职责 ，按功能拆分，比如订单、支付、物流、权限。  . 按重要程度划分，区分核心和非核心功能，比如支付、订单就是核心功能。 |

8.15 你们是怎么分库分表的？分布式 ID 如何生成？

|  |
| --- |
| 使用了水平分库的方式。雪花算法是一种生成分布式全局唯一 ID 的算法 |

8.16 所有异常的共同的祖先是？运行时异常有哪几个？

|  |
| --- |
|  |

##### 9.小厂后端十连问

###### 9.1 Redis 的 key 和 value 可以存储的最大值分别是多少？

|  |
| --- |
| 512M |

###### 9.2 怎么利用 Redis 实现数据的去重？

|  |
| --- |
| Redis 的 set  Redis 的 bit:它可以用来实现比 set 内存高度压缩的计数  HyperLogLog  bloomfilter 布隆过滤器;  [面试必备：布隆过滤器是](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247493887&idx=2&sn=a18d3a5726d04c245d6f05bbeb024c4f&chksm=cf2237d6f855bec031c444f026ccb6e08923b720503d605a327f4308ef5c79638eff3784f9ac&token=1967970286&lang=zh_CN#rd) [什么？有什么用？](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247493887&idx=2&sn=a18d3a5726d04c245d6f05bbeb024c4f&chksm=cf2237d6f855bec031c444f026ccb6e08923b720503d605a327f4308ef5c79638eff3784f9ac&token=1967970286&lang=zh_CN#rd) |

###### 9.3 Redis 什么时候需要序列化？ Redis 序列化的方式有哪

|  |
| --- |
| . 序列化： 将 Java 对象转换成字节流的过程。  . 反序列化： 将字节流转换成 Java 对象的过程。 |

**9.4 MySQL 的B+树的高度怎么计算？(比如有 100w 的数据，字段为 int 类型)**

|  |
| --- |
| 1页16K，  叶子节点：16k/1k =16个，非叶子节点：16/14B = 10\*1024B/4B =1170  总共：1170\*1170\*16=21902400 千万级别 |

###### 9.5 线程池的状态有哪些？获取多线程并发执行结果的方式有哪些？

|  |
| --- |
| RUNNING,SHUTDOWN,STOP,TIDYING,TERMINATED。 |

###### 9.6 线程池原理？各个参数的作用。

|  |
| --- |
| publicThreadPoolExecutor(intcorePoolSize,intmaximumPoolSize,longkeepAliveTime,  TimeUnitunit,  BlockingQueue<Runnable>workQueue,  ThreadFactory threadFactory,  RejectedExecutionHandlerhandler)  . AbortPolicy(抛出一个异常，默认的)  . DiscardPolicy(直接丢弃任务)  . DiscardOldestPolicy ( 丢弃队列里最老的任务， 将当前这个任务继续提交给线程 池)  . CallerRunsPolicy (交给线程池调用所在的线程进行处理) |

###### 9.7 ThreadLocal 的使用场景有哪些？原理？内存泄漏？

|  |
| --- |
| 即线程本地变量。如果你创建了一个 ThreadLocal 变量 ，那 么访问这个变量的每个线程都会有这个变量的一个本地拷贝  **场景：**  . 数据库连接池  . 会话管理中使用  **原理：**  . Thread 对象中持有一个 ThreadLocal .ThreadLocalMap 的成员变量。  . ThreadLocalMap 内部维护了 Entry 数组 ，每个 Entry 代表一个完整的对象 ， key 是 ThreadLocal 本身 ，value 是 ThreadLocal 的泛型值。  **内存泄漏：**  如何解决内存泄漏问题？使用完 ThreadLocal后 ，及时调用 remove()方法释放内存空间。 |

###### 9.8 kafka 是如何保证消息的有序性？

|  |
| --- |
| 写 N 个内存 queue ，具有相同 key 的数据都到同一个内存 queue ；然后对于 N 个线程 ，每个线程分别消费一个内存 queue 即可，这样就能保证顺序性。  在发消息的时候指定Partition Key ，Kafka 对其进行 Hash 计算 ，根据计算结果决定放入哪个 Partition。这样 Partition Key 相同的消息会放在同一个 Partition。然后多消 费者单线程消费指定的 Partition。 |

###### 9.9 Nacos 的选举机制了解嘛？说下 Raft 算法？

|  |
| --- |
| Nacos 作为配置中心的功能是基于 Raft 算法来实现的。  Raft 算法是分布式系统开发首选的共识算法，它通过“一切以领导者为准”的方式，实现一系列值的共识和各节点日志的一致。  Raft 选举规程 涉及三种角色和任期 ( Term )，  . **Follower** ：默默地接收和处理来自 Leader 的消息 ，当等待Leader心跳信息超时的时候 ，就主动站出来，推荐自己当Candidate。  . **Candidate** ：向其他节点发送投票请求 ，通知其他节点来投票 ，如果赢得了大多数 ( N/2+1 )选票 ，就晋升 Leader。  . **Leader** ：负责处理客户端请求 ，进行日志复制等操作， 每 一轮选举的目标就是选 出一个领导者； 领导者会不断地发送心跳信息 ，通知其他节点“我是领导者 ，我还活着 ，你们不要发起新的选举，不用找个新领导者来替代我。”  . **Term** ：这跟民主社会的选举很像 ，每 一届新的履职期称之为一届任期 |

###### 9.10 聊一聊 TCC 补偿机制

|  |
| --- |
| **try** 阶段： 尝试去执行， 完成所有业务的一致性检查， 预留必须的业务资源。  . **Confirm** 阶段： 该阶段对业务进行确认提交，不做任何检查， 因为 try 阶段已经检 查过了，默认 Confirm 阶段是不会出错的。  . **Cancel** 阶段： 若业务执行失败 ，则进入该阶段，它会释放 try 阶段占用的所有业 务资源 ，并回滚 Confirm 阶段执行的所有操作。 |

##### 10.腾讯云

###### 10.1 聊聊项目，好的设计，好的代码

|  |
| --- |
| [记一次接口性能优化实践总结：优化接口性能的八个建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488004&idx=1&sn=00840efd9c0bd0a7f172b59eb2ca130f&chksm=cf21cd2df856443bf21d8e09cfe5c8452ecaf82e3c2210fca3b28829ded04defddcf63c0a59b&token=162724582&lang=zh_CN&scene=21#wechat_redirect)  [工作四年，分享](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect) **[50](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect)** [个让你代码更好的小建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488708&idx=1&sn=6e2e0a740f5d42a59641487a0bf1e3bf&chksm=cf21cbedf85642fbb485fa1c7bf9af21923d8503f2542b6f8283ce79ddc683f7d9e45da83100&token=162724582&lang=zh_CN&scene=21#wechat_redirect)  . [看一遍就理解：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)**[order](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)****[by](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)** [详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)  . [看一遍就理解：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)**[group](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)****[by](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)** [详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)  . [实战！聊聊如何解决](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd) **[MySQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd)** [深分页问题](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd)  . [后端程序员必备：书写高质量](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [的](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[30](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [条建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)  . [阿里一面，给了几条](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)**[，问需要执行几次树搜索操作？](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)  . [生产问题分析！](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)**[delete](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)****[in](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [子查询不走索引？！](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495170&idx=1&sn=ce914de3abdb0d887e286b680b25111f&chksm=cf22312bf855b83d31a00da110626747df8e69fca1bc310642c56e39d663b006a8105f9fb1e1&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) |

###### 10.2 谈谈什么是零拷贝？

|  |
| --- |
| 零拷贝是指计算机执行 IO 操作时， CPU 减少或不需要将数据从一个存储区域复制到 另一个存储区域 ，从而可以减少上下文切换以及 CPU 的拷贝时间。它是一种 I/O 操作优化技术  **. mmap+write 4+3**  **. Sendfile 2+3**  **. 带有 DMA 收集拷贝功能的 sendfile 2+2** |

###### 10.3 一共有几种 IO 模型？ NIO 和多路复用的区别？

|  |
| --- |
| . 阻塞 IO 模型  . 非阻塞 IO 模型：NIO(非阻塞 IO 模型)存在性能问题， 即频繁的轮询  . IO 多路复用模型：等到内核数据准备好了，主动通知应用进程再去进行系统调用。  . IO 模型之信号驱动模型  . IO 模型之异步 IO(AIO) |

###### 10.4 Future 实现阻塞等待获取结果的原理？

|  |
| --- |
| Future.get()用于异步结果的获取。它是阻塞的， 背后原理是什么呢？FutureTask 就是 **Runnable 和 Future 的结合体** ，我们可以把 Runnable 看作生产者， Future 看作消费者。而 FutureTask 是被这两者共享的， 生产者运行 run 方法计算结果， 消费者通过 get 方法获取结果。 |

###### 10.5 ReentrantLock 和 Synchronized 的区别？

|  |
| --- |
| . Synchronized 是依赖于 JVM 实现的，而 ReenTrantLock 是 API 实现的  . 在 Synchronized 优化以前， synchronized 的性能是比 ReenTrantLock 差很多 的， 但是自从Synchronized 引入了偏向锁， 轻量级锁(自旋锁) 后 ，两者性能 就差不多了。  . Synchronized 的使用比较方便简洁 ，它由编译器去保证锁的加锁和释放。而ReenTrantLock 需要手工声明来加锁和释放锁，最好在finally 中声明释放锁。  . ReentrantLock 可以指定是公平锁还是非公平锁。而synchronized 只能是非公 平锁。  . ReentrantLock 可响应中断、可轮回，而 Synchronized 是不可以响应中断的  **[Synchronized](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487942&idx=1&sn=bbc68d1b9da23bf6474378e310b1ef1b&chksm=cf21ceeff85647f9ad7a08226849fcba3f9481387d13b17a5787fb94027647de81c349f9e390&token=162724582&lang=zh_CN&scene=21#wechat_redirect)** [解析——如果你愿意一层一层剥开我的心](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487942&idx=1&sn=bbc68d1b9da23bf6474378e310b1ef1b&chksm=cf21ceeff85647f9ad7a08226849fcba3f9481387d13b17a5787fb94027647de81c349f9e390&token=162724582&lang=zh_CN&scene=21#wechat_redirect) |

###### 10.6 聊聊 AOS ？ReentrantLock 的实现原理？

|  |
| --- |
| . state 状态的维护。  . CLH 队列  . ConditionObject 通知  . 模板方法设计模式  . 独占与共享模式。  . 自定义同步器。  [AQS 解析与实战](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487939&idx=1&sn=560f9ec0fdbc081949383bbee2407b0e&chksm=cf21ceeaf85647fc24537661ca063f9537b5cb5090da1c4ecf1f4d8326a5359391143bd16e1a&token=1496082535&lang=zh_CN#rd) |

###### 10.7 乐观锁和悲观锁， 让你来写你怎么实现？

|  |
| --- |
| . SQL 语句 select ...for update 就是悲观锁的一种实现  . 还有 Java 的 synchronized 关键字也是悲观锁的一种体现  它不会上锁， 只是在更新数据时， 再去判断其他线程在这之前有没 有对数据进行过修改。  乐观锁实现方式 ：乐观锁一般会使用版本号机制或 CAS 算法实现。  **[CAS](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487937&idx=1&sn=206a37bf6d6a7aa1d05674c479ed7a72&chksm=cf21cee8f85647fe7a082049a41c0f640f54976d2cdf4302b24c5517ca42b854eb84b13ece10&token=477542327&lang=zh_CN#rd)** [乐观锁解决并发问题的一次实践](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487937&idx=1&sn=206a37bf6d6a7aa1d05674c479ed7a72&chksm=cf21cee8f85647fe7a082049a41c0f640f54976d2cdf4302b24c5517ca42b854eb84b13ece10&token=477542327&lang=zh_CN#rd) |

###### 10.8 Paxos 协议了解？工作流程是怎么样的？

10.8.1 为什么需要 Paxos 算法？：Paxos 算法是一种基于消息传递的分布式一致 性算法。

10.8.2 Paxos 的角色 ：

|  |
| --- |
| . **Proposer** ：它可以提出提案 (Proposal) ，提案信息包括提案编号和提案值。  . **Acceptor** ：接受接受 ( accept ) 提案。一旦接受提案，提案里面的提案值 (可以用V表示) 就被选定了  . **Learner**: 哪个提案被选定了, Learner 就学习这个被选择的提案 |

10.8.3 Paxos 算法推导过程???

10.8.4 算法流程???

|  |
| --- |
|  |

###### 10.9 B+树聊一下？ B+树是不是有序？ B+树和 B-树的主要

|  |
| --- |
| 1.B-树内部节点是保存数据的;而 B+树内部节点是不保存数据的，只作索引作用,它的叶子节点才保存数据  2.B+树相邻的叶子节点之间是通过链表指针连起来的， B-树却不是。  3.查找过程中，B-树在找到具体的数值以后就结束，而B+树则需要通过索引找到叶子结点中的数据才结束  4.B-树中任何一个关键字出现且只出现在一个结点中，而 B+树可以出现多次。 |

###### 10.10 TCP 怎么实现拥塞控制？

|  |
| --- |
| 1.慢启动算法 2.拥塞避免算法 3.拥塞发生 4.快速恢复 见上 |

###### 10. 11. JVM 调优

10.11.1 一般什么时候考虑 JVM 调优呢？

|  |
| --- |
| . Heap 内存( 老年代 ) 持续上涨达到设置的最大内存值 ；  . Full GC 次数频繁；  . GC 停顿时间过长( 超过 1 秒 ) ；  . 应用出现 OutOfMemory 等内存异常 ；  . 应用中有使用本地缓存且占用大量内存空间 ；  . 系统吞吐量与响应性能不高或下降。 |

10.11.2 JVM 调优的目标

|  |
| --- |
| 1.延迟： GC 低停顿和GC 低频率 ；2.低内存占用 ；3.高吞吐量; |

10.11.3 JVM 调优量化目标

|  |
| --- |
| 1、Heap内存使用率70%内 2.Old generation内存使用率70%内，3.augpause<=1秒 4.Full Gc 0；  [深入理解JVM虚拟机——JVM参数以及调优实战 (baidu.com)](https://baijiahao.baidu.com/s?id=1747377064336158073&wfr=spider&for=pc) |

10.11.4 JVM 调优的步骤

|  |
| --- |
| . 分析 GC 日志及**dump** 文件， 判断是否需要优化， 确定瓶颈问题点 ；  . 确定 JVM 调优**量化目标** ；  . **确定 JVM 调优参数**( 根据历史 JVM 参数来调整 ) ；  . 依次**调优内存、延迟、 吞吐量等指标** ；  . 对比观察调优前**后的差异** ；  . 不断的分析和调整 ，直到**找到合适的JVM参数配置**；  . 找到最合适的参数，将这些参数应用到所有服务器，并进行后续跟踪。 |

10.11.5 常见的 JVM 参数 ：见上

10.11.6 常用调优策略

|  |
| --- |
| . 选择合适的垃圾回收器  . 调整内存大小(垃圾收集频率非常频繁,如果是内存太小，可适当调整内存大小)  . 调整内存区域大小比率 (某一个区域的GC 频繁 ，其他都正常。)  . 调整对象升老年代的年龄 ( 老年代频繁 GC ，每次回收的对象很多。)  . 调整大对象的标准(老年代频繁 GC ，每次回收的对象很多,而且单个对象的体积都 比较大。 )  . 调整 GC 的触发时机(CMS ，G1 经常 Full GC ，程序卡顿严重。 )  . 调整 JVM 本地内存大小(GC 的次数、时间和回收的对象都正常， 堆内存空间充足 ， 但是报 OOM) |

###### 10.12 数据库分库分表的缺点是啥？

|  |
| --- |
| 1.事务问题 2.跨节点jion的问题 3.ID问题 4.跨分片排序分页问题 |

###### 10.13 分布式事务如何解决？TCC了解？

|  |
| --- |
| 1.CAP理论 2.BASE理论 3.TCC补偿机制  [后端程序员必备：分布式事务基础篇](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487969&idx=1&sn=1a7c255439810aa12d0417a69c709bce&chksm=cf21cec8f85647dee38af93bb6747fc4e597c9918ad83f5ad30fe726f2918e017e390b2b7413&token=162724582&lang=zh_CN&scene=21#wechat_redirect) |

###### 10.14 RocketMQ 如何保证消息的准确性和安全性？

|  |
| --- |
| [消息队列经典十连问](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497847&idx=1&sn=29a32672b712e7dfadfa36c9902b2ec7&chksm=cf22275ef855ae484fb3f51a5726e9a4bc45222e8fbbd33631d177dc4b5619c36889ea178463&token=1077989845&lang=zh_CN#rd) |

###### 10.15. 三个数求和

##### 11. 社招21连问

###### 11.1 Mysql 索引在什么情况下会失效

|  |
| --- |
| . 查询条件**包含 or** ，可能导致索引失效  . 如何字段类型是**字符串** ，where 时一定用引号括起来 ，否则索引失效  . **like 通配符**可能导致索引失效。  .  **联合索引**，查询时的条件列不是联合索引中的第一个列，索引失效。  . 在索引列上使用 mysql 的**内置函数**，索引失效。  . 对**索引列运算** (如， +、-、\*、/ )，索引失效。  . 索引字段上使用( **！= 或者 < > ，not in** )时，可能会导致索引失效。  . 索引字段上使用 **is null** ， is not null ，可能导致索引失效。  . **左连接查询**或者右连接查询查询关联的字段编码格式不一样 ，可能导致索引失效。  . mysql 估计使用**全表扫描**要比使用索引快,则不使用索引。 |

###### 11.2 MySql 的存储引擎 InnoDB 与 MyISAM 的区别

|  |
| --- |
| 1.事务 2.外键 3.MVCC多版本并发控制 4.select count(\*) 5.行锁 6.主键必要 7.空间 8.插入顺序 9.事务 |

###### 11.3 mysql 在项目中的优化场景 ，慢查询解决等

|  |
| --- |
| 我们面对慢查询， 首先想到的就是加索引。你可以给面试官描述一下，一个加了索引的 SQL ，是怎么执行查找的，打开慢查询日志 slow\_query\_log ，确认 SQL 语句是否占用过多资源 ，用 explain 查询执行计划、对 group by、order by、join等语句优化，如果数据量 实在太大，是否考虑分库分表等等。  [阿里一面，给了几条](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)**[，问需要执行几次树搜索操作？](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247488130&idx=1&sn=2f225ba5100c882089eec8a2666fee54&chksm=cf21cdabf85644bd91c8f0bc223d883a53896f3bd71e06a4167e050155fe64f55872c7ad97d8&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)  [看一遍就理解：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)**[order](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)****[by](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)** [详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247490571&idx=1&sn=e8638573ec8d720fd25da5b2b0d90ed2&chksm=cf21c322f8564a34461acd9811730d14d12075cf5c7438a3a11433725b9ce463fcb78e7916a1&token=500637053&lang=zh_CN&scene=21#wechat_redirect)  [看一遍就理解：](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)**[group](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)****[by](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)** [详解](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497527&idx=1&sn=1f30251d88b0e935bfffc3e8eaf53f28&chksm=cf22281ef855a1084fe84a7b257db5734c7b982c6ddaf9ef497d4e31e60faebd5f329e3c55a6&token=2101142450&lang=zh_CN#rd)  [实战！聊聊如何解决](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd) **[MySQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd)** [深分页问题](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247495139&idx=1&sn=9dd98a8e09af48440cc5f01d3aafd87e&chksm=cf2232caf855bbdc4ea538550ecde6c575c91a1d1b1c42f3bc6091c715dde1a4a5e90d3f7ce2&token=1913427154&lang=zh_CN#rd)  [后端程序员必备：书写高质量](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[SQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [的](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) **[30](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect)** [条建议](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247487972&idx=1&sn=cd035a7fcd7496658846ab9f914be2db&chksm=cf21cecdf85647dbc53e212bf1a2b95d0eb2bffe08dc0141e01f8a9b2088abffc385a2ef584e&token=1495321435&lang=zh_CN&scene=21#wechat_redirect) |

###### 11.4 Mysql有什么索引，索引模型是什么

|  |
| --- |
| 数据结构维度：B+树索引，哈希索引，全文索引，R-Tree索引  物理存储维度：聚集索引，非聚集索引  逻辑维度：主键索引，普通索引，联合索引，唯一索引，空间索引 |

###### 11.5 B-树与 B+树的区别？为什么不用红黑树

|  |
| --- |
| 区别见上：  1.哈希只适合等值查询，不适合范围查询  2. 二叉树可能会特殊化为一个链表，导致全表扫描  3. 红黑树是特化的平衡二叉树，MySQL数据量大导致索引体积较大，内存放不下而从磁盘读取，树层太高而从磁盘读取次数变多  4.B-Tree，叶子节点和非叶子节点都保存数据，相同的数据量，B+树更短壮，查询磁盘量更小。 |

###### 11.6 Mysql 主从同步怎么做

|  |
| --- |
| 1.主库的更新 SQL(update、insert、delete)被写到 binlog  2.从库发起连接，连接到主库。  3. 此时主库创建一个 binlog dump thread ，把 binlog 的内容发送到从库。  4.从库启动之后 ，创建一个 I/O 线程 ，读取主库传过来的bin log 内容并写入到 relay log  5. 从库还会创建一个 SQL 线程 ，从 relay log 里面读取内容 ，从 ExecMasterLog\_Pos 位置开始执行读取到的更新事件， 将更新内容写入到 slave 的 db  [面试必备：聊聊](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497982&idx=1&sn=bb589329cceb5462fc41f66ec63dbf56&chksm=cf2227d7f855aec16dd4d3b3425c0401850eeaf2c9cdc82e82722d38a00c24ee9ccfa3353774&token=1913427154&lang=zh_CN#rd) **[MySQL](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497982&idx=1&sn=bb589329cceb5462fc41f66ec63dbf56&chksm=cf2227d7f855aec16dd4d3b3425c0401850eeaf2c9cdc82e82722d38a00c24ee9ccfa3353774&token=1913427154&lang=zh_CN#rd)** [的主从](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497982&idx=1&sn=bb589329cceb5462fc41f66ec63dbf56&chksm=cf2227d7f855aec16dd4d3b3425c0401850eeaf2c9cdc82e82722d38a00c24ee9ccfa3353774&token=1913427154&lang=zh_CN#rd) |

###### 11.7 乐观锁与悲观锁的区别？

|  |
| --- |
| select ...for update 就是悲观锁一种实现。  乐观锁实现方式：乐观锁一般会使用版本号机制或 CAS算法实现。 |

###### 11.8 聊聊 binlog 日志

|  |
| --- |
| binlog 是归档日志 ，属于 MySQL Server 层的日志。可以实现主从复制和数据恢复两个作用。  binlog 日志有三种格式，分别是 statement ，row 和mixed。  statement 采用SQL原文，他可能会导致主库不一 致(主库和从库选的索引不一样时)  Row: 精确到具体一条数据，并操作，保证了准确性倒是体积太大  Mixed: 这种方案，是 row 和 statement 格式混合 使用， 当 MySQL 判断可能数据不一致时， 就用 row 格式， 否则使用就用statement 格式。 |

###### 11.9 Redis 持久化有哪几种方式，怎么选？

###### 11.9.1 AOF 持久化

|  |
| --- |
| 日志记录追加，默认不开启，主要解决持久化实时性问题。  三种写回策略：always,everysec,no  AOF有重写机制，压缩删除无效，冗余，过期指令  AOF优点：数据一致性和完整性高，秒级丢失； AOF缺点：AOF体积大，恢复慢 |

###### 11.9.2 RDB

|  |
| --- |
| 快照形式，默认开启；手动触发：save,bigsave；自动触发：save m n;  fork子进程进行bigsave或者save，恢复快，可增量快照 |

###### 11.9.3 如何选择 RDB 和AOF

|  |
| --- |
| 可混合使用 |

###### 11.10 Redis 主从同步是怎样的过程？

|  |
| --- |
| 第 一阶段：主从库间建立连接、 协商同步。从库发送psync命令，主库收到后响应runID和offect  第二阶段：主库把数据同步到从库，从库收到数据后 ，完成本地加载。主库bgsave，将RDB文件发送到从库，过程中记录新的写操作到replication buffer。  第三阶段，主库把新写的命令，发送到从库。主库发送RDB后，将replication buffer修改发送到从库。 |

###### 11.11 聊聊 Redis 的 zset ，它是怎么实现的？

|  |
| --- |
| 它的底层内部编码 ：ziplist ( 压缩列表)、skiplist ( 跳跃表) |

###### 11.12 Redis 过期策略和内存淘汰策略

11.12.1 Redis 的过期策略

11.12.2 Redis 内存淘汰策略

|  |
| --- |
| 过期策略：定时过期，惰性过期，定期过期  内存淘汰策略：volatile-lru ， volatile-lfu ，volatile-random，noeviction |

###### **11.13. Hashmap 是怎样实现的？为什么要用红黑树，而不 平衡二叉树？为什么在 1.8 中链表大于 8 时会转红黑树？ HashMap 是线性安全的嘛？如何保证安全？**

11.13.1 Hashmap 是怎样实现的？ jdk1.7是数组+链表 jdk1.8后是数组+链表+红黑树

11.13.2 为什么要用红黑树，为什么不用二叉树？为什么不用平衡二 叉树

|  |
| --- |
| 红黑树既不会出现二叉树最坏为链表的情况，也不会因为保持平衡而不断自旋消耗CPU |

11.13.3 为什么在 1.8 中链表大于 8 时会转红黑树？ ：提交检索效率 log2(8)=3 <(8/2)

11.13.4 HashMap 是线性安全的嘛？如何保证安全？

|  |
| --- |
| 不安全可以使用HashTable、Collections.synchronizedMap、以及ConcurrentHashMap 可以实现线程安全 |

###### 11.14 select 和 epoll 的区别

11.14.1 IO 多路复用之select

|  |
| --- |
| 应用进程通过调用select 函数，可以同时监控多个 fd ，在 select 函数监控的 fd 中，只要有任何一个数据状态准备就绪了， select 函数就会返回可读状态 ，这时应用进程再发起 recvfrom请求去读取数据。  缺点：监听上限1024；select返回后得遍历fdset。 |

11.14.2 IO 多路复用之epoll

|  |
| --- |
| epoll 先通过epoll\_ctl()来注册一个 fd ( 文件描述符 )，一旦基于某个 fd 就绪 时，内核会采用回调机制 ，迅速激活这个 fd ，当进程调用epoll\_wait() 时便得到通知。这里去掉了遍历文件描述符的坑爹操作，而是采用监听事件回调的机制。 |

###### 11.15 http 与 https 的区别， https 的原理，如何加密的？

|  |
| --- |
| HTTPS= HTTP+SSL/TLS ，可以理解 Https 是身披 SSL(Secure Socket Layer,安全套接层)的 HTTP。 |

###### 11.16 Raft 算法原理 682

16.1 Raft 角色:**跟随者 ( Follower ) ;候选人 ( Candidate ) ;领导者 ( Leader )**

11.16.2 领导选举过程

|  |
| --- |
| 1.所有节点一开始都是follower，随机设定一个选举超时时间  2.某个节点在规定时间内么有收到来自leader的心跳，A节点自告奋勇将自己切换为Candidate状态，并向其他follower发送请求，询问是否选举自己为leader  3.其他候选人收到请求后，如果在这届任期内还未投过票，则将票投给A  4.当收到过半投票后，A则即将成为本届Leader，并发送消息告知其他节点，阻止Follower发起新的选举。 |

11.16.2 日志复制

|  |
| --- |
| 当有了 leader ，系统应该进入对外工作期了。客户端的一切请求来发送到leader ，leader 来调度这些并发请求的顺序，并且保证 leader 与 followers 状态的一致性。Leader 接收到来自客户端写请求后 ，处理写请求的过程其实 就是一个日志复制的过程。 |

###### 11.17 消息中间件如何做到高可用

|  |
| --- |
| Kafka 的基础集群架构，由多个broker 组成 ，每个broker 都是一个节点。当你创建一个 topic 时，它可以划分为多个partition ，而每个partition 放一部分数据，分别存在于不同的broker 上。也就是说，一个topic 的数据，是分散放在多个机器上的，每个机器就放一部分数据。 |

###### 11.18 消息队列怎么保证不丢消息的

11.18.1 生产者保证不丢消息

11.18.2 存储端不丢消息

11.18.3 消费阶段不丢消息

###### 11.19 Redis 如何保证高可用？聊聊 Redis 的哨兵机制

|  |
| --- |
| . 发送命令，等待 Redis 服务器( 包括主服务器和从服务器) 返回监控其运行状态 ；  . 哨兵监测到主节点宕机 ，会自动将从节点切换成主节点， 然后通过发布订阅模式通 知其他的从节点，修改配置文件， 让它们切换主机 ；  . 哨兵之间还会相互监控 ，从而达到高可用。 |

###### 11.20 无重复字符的最长子串

##### 12. 蚂蚁金服一面

###### 12.1 用到分布式事务嘛?为什么用这种方案,有其他方案嘛?

|  |
| --- |
| 2pc ;TCC;本地消息表；最大努力通知；Saga事务；业界目前使用本地消息表这种方案是比较多的，它的核心思想就是将分布式事 务拆分成本地事务进行处理。可以看一下基本的实现流程图吧 ： |

###### 12.1.2 什么是分布式事务

|  |
| --- |
| 分布式事务需要需要知道 CAP 理论和 BASE 理论。 |

###### 12.1.3 分布式事务的几种解决方案

|  |
| --- |
|  |

###### 12.2 JDK6、7、8 分别提供了哪些新特性

|  |
| --- |
| JDK6:插入式注解处理 API ( lombok 框架基于这个特性实现)  JDK 7：switch，try-with-resources，整数类型如( byte ，short ，int ，long )能够用二进制来表示，泛型实例化类型自动推断,即”<>”，增强的文件系统  JDK8：lambada 表达式，函数式接口，Base64， 重复注解， JVM 的新特性， Date Time API (如 LocalDate |

###### 12.3 https 原理，工作流程：主要是加解密

###### 12.4 讲讲 java jmm volatile 的实现原理

|  |
| --- |
| volatile 关键字是 Java 虚拟机提供的的最轻量级的同步机制 ，它作为一个修饰 符 ，用来修饰变量。它保证变量对所有线程可见性， 禁止指令重排 ，但是不保  证原子性。 |

###### 12.5 讲 一讲 7 层网络模型， tcp 的为什么要三次握手 ：见上

###### 12.6 说说线程池的工作原理 709

|  |
| --- |
|  |

###### 12.7 你们数据库的高可用是怎么实现的?

12.7.1 主从或主主半同步复制：简单，直接切换；但是依赖半同步复制，转为异步后影响一致性。

12.7.2 半同步复制优化：部署使用简单，直接切换，比方案1更能保证一致性。修改源码难度大。

12.7.3 高可用架构优化：zookeeper集群搭建，高可用易扩展，引入Zookeeper系统更加复杂。

12.7.4 共享存储

|  |
| --- |
| 共享存储实现了数据库服务器和存储设备的解耦 ，不同数据库之间的数据同步 不再依赖于 MySQL 的原生复制功能，而是通过磁盘数据同步的手段， 来保证 数据的一致性。 |

12.7.5 分布式协议

|  |
| --- |
| 分布式协议可以很好解决数据一致性问题。 常见的部署方案就是 **MySQL** **cluster** ，它是官方集群的部署方案， 通过使用 NDB 存储引擎实时备份冗余数据，实现数据库的高可用性和数据一致性。配置复杂 |

###### 12.8 读写分离的场景下,怎么保证从数据库读到最新的数据?

|  |
| --- |
| 在高并发场景或者网络不佳的场景 ，如果存在较大的主从同步数据延迟，这时 候读请求去读从库， 就会读到旧数据。这时候最简单暴力的方法， 就是强制读 主库。实际上可以使用缓存标记法。 |

###### 12.9 如何保证 MySQL 数据不丢？ 719

12.9.1 binlog 日志

|  |
| --- |
| Binlog:  write是讲日志写到文件系统的page cache中，速度快；  Fsync:刷盘操作 氛围0 1 N三种类型； |

12.9.2 redo log 日志

|  |
| --- |
| Redo log: 重做日志文件。记录事务对数据页做了哪些修改。三种策略 0 1 2  **0：edo log buffer 1：Disk 2：page cache** |

###### 12.10 高并发下如何设计秒杀系统？

|  |
| --- |
| 1.页面静态化 2.按钮置灰 3.服务单一职责4.秒杀链接加盐 5.限流&熔断&降级 6.分布式锁 7.MQ异步处理 |

##### 13.虾皮服务端 15 连问

###### 13.1 排序链表

###### 13.2 对称与非对称加密算法的区别

|  |
| --- |
| 秘钥是否相同 |

###### 13.3 TCP 如何保证可靠性

|  |
| --- |
| 首先3次握手 4次挥手；其次：TCP有状态，记录了哪些已接收哪些未接收，并按序到达； 在此：还体现在TCP可控制，有报文校验，ack应答，超时重传，去重，流量控制，拥塞控制上。 |

###### 13.4 聊聊五种 IO 模型 见上

###### 13.5 hystrix 工作原理

|  |
| --- |
|  |

###### 13.6 延时场景处理

|  |
| --- |
| 1.JDK的DelayQueue延迟队列 2.时间轮询 3.数据库定时任务 4.Redis Zset实现 5.MQ延时队列实现。 |

###### 13.7 https 请求过程

|  |
| --- |
| HTTPS = HTTP + SSL/TLS 见上 |

###### 13.8 聊聊事务隔离级别，以及可重复读实现原理 743

13.8.1 数据库四大隔离级别：1.读未提交 2.读已提交 3.可重复读 3.串行化。

13.8.2 Read View 可见性规则 了解

13.8.3 可重复读实现原理

|  |
| --- |
| InnoDB 实现 **MVCC** ，是通过 Read View+ Undo Log 实现的， Undo Log 保存了 历史快照 ，Read View 可见性规则帮助判断当前版本的数据是否可见。 |

###### 13.9 聊聊索引在哪些场景下会失效？

|  |
| --- |
| 见上 |

###### 13.10 什么是虚拟内存

|  |
| --- |
| 零拷贝实现思想 ，就利用了虚拟内存这个点： 多个虚拟内存可以指向同一个物 理地址 ，可以把内核空间和用户空间的虚拟地址映射到同一个物理地址 |

###### 13.11 排行榜的实现，比如高考成绩排序

|  |
| --- |
| zaddkeyscoremember[scoremember ...] ，zrankkeymember |

###### 13.12 分布式锁实现

13.12.1 命令 setnx + expire 分开写

13.12.2 setnx + value 值是过期时间

13.12.3 ：set 的扩展命令( set ex px nx )(注意可能存在的问

13.12.4 set ex px nx + 校验唯一随机值,再删除

13.12.5 Redisson

|  |
| --- |
|  |

###### 13.13. 零拷贝

13.13.1 传统 IO 流程

13.13.2 mmap+write 实现的零拷贝

13.13.3 sendfile 实现的零拷贝

13.13.4 sendfile+DMA scatter/gather 实现的零拷贝

|  |
| --- |
| 见上 |

###### 13.14 synchronized

13.14.1 反编译后 ，monitorenter、monitorexit、ACC\_SYNCHRONIZED

13.14.2 monitor 监视器

13.14.3 Java Monitor 的工作机理

13.14.4 对象与 monitor 关联

|  |
| --- |
| 想要获取 monitor 的线程,首先会进入\_EntryList 队列。  当某个线程获取到对象的 monitor 后,进入 Owner 区域 ，设置为当前线程, 同时计数器 count 加 1。  如果线程调用了wait()方法，则会进入 WaitSet 队列。它会释放monitor 锁，即将 owner 赋值为 null,count 自减 1,进入 WaitSet 队列阻塞等待。 |

###### 13.15 分布式 id 生成方案有哪些？什么是雪花算法？

数据库自增 UUID ，snowID，

##### 14. 字节跳动面试

###### 14.1 说说 Redis 为什么快

14.1.1 基于内存存储实现

14.1.2 高效的数据结构

14.1.2.1 SDS 简单动态字符串

14.1.2.2 字典

14.1.2.3 跳跃表

14.1.3 合理的数据编码

|  |
| --- |
| . String ：如果存储数字的话，是用 int 类型的编码;如果存储非数字，小于等于 39 字节的字符串 ，是 embstr ；大于 39 个字节 ，则是 raw 编码。  . List ：如果列表的元素个数小于 512 个，列表每个元素的值都小于64 字节 (默 认)，使用 ziplist 编码 ，否则使用 linkedlist 编码  . Hash ：哈希类型元素个数小于 512 个， 所有值小于64 字节的话，使用ziplist 编 码,否则使用 hashtable 编码。  . Set ：如果集合中的元素都是整数且元素个数小于512 个，使用 intset 编码 ，否 则使用 hashtable 编码。  . Zset ：当有序集合的元素个数小于 128 个， 每个元素的值小于64 字节时，使用 ziplist 编码 ，否则使用 skiplist ( 跳跃表) 编码 |

14.1.4 合理的线程模型

|  |
| --- |
| Epoll多路复用IO模型 |

###### 14.2 Redis 有几种数据结构，底层分别是怎么存储的

14.2.1 Redis 的五种基本数据类型 ：String ，Hans，List，Set，zset，

14.2.2 Redis 的三种特殊数据类型 :Geospatial（地理定位）, Hyperloglog（基数统计UV）, BitMap

###### 14.3 Redis 有几种持久化方式 见上

###### 14.4 多线程情况下,如何保证线程安全？

|  |
| --- |
| 加锁，比如悲观锁 select for update ，sychronized 等，如乐观锁，乐观锁如 CAS等，还有redis 分布式锁等等。 |

###### 14.5 用过 volatile 吗？它是如何保证可见性的，原理是什么

|  |
| --- |
| Volitile关键字修饰后，编译多了个lock,lock 指令相当于一个内存屏障 ，它保证以下这几点：  1. 重排序时不能把后面的指令重排序到内存屏障之前的位置  2. 将本处理器的缓存写入内存  3. 如果是写入动作，会导致其他处理器中对应的缓存无效。 |

###### 14.6 MySQL 的索引结构，聚簇索引和非聚簇索引的区别

|  |
| --- |
|  |

###### 14.7 MySQL 有几种高可用方案，你们用的是哪一种 见上

###### 14.8 说说你做过最有挑战性的项目, 你负责那个模块,哪些最有挑战性，说说你做了哪些优化？

|  |
| --- |
|  |

###### 14.9 秒杀采用什么方案。 见上

###### 14.10 聊聊分库分表,分表为什么要停服这种操作，如果不停服可以怎么做

14.10.1 分库分表方案 ：水平和垂直分库分表

14.10.2 常用的分库分表中间件： sharding-jdbc MyCat

14.10.3 分库分表可能遇到的问题：1.事务问题，2.跨节点join问题 3.数据迁移，4.ID问题 5.跨分片排序

14.10.4 分表要停服嘛？不停服怎么做？

|  |
| --- |
| 1. 编写代理层，加个开关  2. 发版全量后 ，开启双写  3. 通过脚本把旧表的存量数据写入新表  4. 停读旧表改读新表，  5. 当读写新表一段时间之后 ，如果没有业务问题， 就可以停写旧表啦 |

###### 14.11 redis 挂了怎么办？持久化问题，见上

###### 14.12 你怎么防止优惠券有人重复刷？

|  |
| --- |
| [聊聊幂等设计](https://mp.weixin.qq.com/s?__biz=Mzg3NzU5NTIwNg==&mid=2247497427&idx=1&sn=2ed160c9917ad989eee1ac60d6122855&chksm=cf2229faf855a0ecf5eb34c7335acdf6420426490ee99fc2b602d54ff4ffcecfdab24eeab0a3&token=529683793&lang=zh_CN#rd)  1.对同一用户限流 2.对某个IP限流 3.每次请求都更滑IP请求 |

###### 14.13 抖音评论系统怎么设计，如果加入好友关系呢？

|  |
| --- |
|  |

###### 14.14 怎么设计一个短链地址 ，要考虑跨机房部署问题

14.14.1 为什么需要短连接？

14.14.2 短链接的原理

14.14.3 短链接生成的方法 ：自增ID生成后，转为62进制。

###### 14.15 有一个整型数组 ，数组元素不重复，数组元素先升序后 降序，找到最大值。 二分法查找

##### 15. 字节二面

###### 15.1 http 请求头里， expire 和 cache-control 字段含义

15.1.1 expire字段含义 用来区分对缓存机制的支持情况， 请求 头和响应头都支持这个属性

cache-control 含义：过期时间，如果设置了时间，则浏览器会在设置的 时间内直接读取缓存，不再请求。

15.1.2 常见 HTTP 状态码

|  |
| --- |
|  |

###### 15.2 https 原理，数字签名，数字证书。见上

###### 15.3 tcp 连接 client 和 server 有哪些状态 ，time\_wait

15.3.1 tcp 连接

|  |
| --- |
| tcp 连接时， 客户端 client 有 SYN\_SEND、ESTABLISHED 状态 ，服务端 server 有 SYN\_RCVD、ESTABLISHED 状态。 |

15.3.2 time\_wait 状态

|  |
| --- |
| 四次挥手：client有 FIN\_WAIT-1,FIN\_WAIT-2,TIme-WAIT,CLOSED  server端有：CLOSE\_WAIT ,CLOSE\_ACK, CLOSED |

###### 15.4 什么是虚拟内存? 什么是物理内存？ 见上

###### 15.5 一 台机器最多可以建立多少个tcp 连接， client 端 ，server端，超过了怎么办？

|  |
| --- |
| TCP 连接理论受限于ip\_local\_port\_range 参数 65535； |

###### 15.6 Eureka 原理，是否是强一致性， eureka 集群。宕机

15.6.1 eureka 架构

|  |
| --- |
| . Eureka Server ：提供服务注册和发现， 多个 Eureka Server 之间会同步数据， 做到状态一致  . Service Provider ：服务提供方， 将自身服务注册到 Eureka ，从而使服务消费方 能够找到  . Service Consumer ：服务消费方，从 Eureka 获取注册服务列表，从而能够消费 服务  Eureka server 可以集群部署 ，多个节点之间会通过 Replicate ( 异步方式) 进行数据同步 ，保证数据**最终一致性** |

15.6.2 基于集群的 Eureka 架构图

15.6.3 宕机了服务还能调用么？

|  |
| --- |
| Eureka 挂了， 微服务是可以调通的，不过有个前提 ：provider 的地址没变 ！ 如果provider 换了一个 IP 地址或者端口 ，这个时候 ，consumer 就无法及时感知到这种变化， 就会调不通。 |

15.6.4 Eureka 和ZooKeeper 对比

|  |
| --- |
| Zookeeper 保证 CP(一致性和分区容错性) ，但是不保证可用性， ZK 的 leader 选 举期间，是不可用的Eureka 保证 AP (可用性和分区容错性)，它优先保证可用性 ，几个节点挂掉不 会影响正常节点的工作 |

###### 15.7.Hystrix 了解嘛？说说 Hystrix 的工作原理

|  |
| --- |
| 不知 |

###### 15.8 zookeeper 一致性保证 ，zab 协议原理 ，

|  |
| --- |
| Zab 协议 ，英文全称是Zookeeper Atomic Broadcast ( Zookeeper 原子广播 )。 Zookeeper 是通过 Zab 协Zab 协议要求每个 Leader 都要经历三个阶段： 发现，同步 ，广播。  **发现**：要求zookeeper 集群必须选举出一个 Leader 进程 ，同时 Leader 会维护一个 Follower 可用客户端列表。 将来客户端可以和这些 Follower 节点进行通信。  **同步** ：Leader 要负责将本身的数据与 Follower 完成同步 ，做到多副本存储。这 样也是提现了 CAP 中的高可用和分区容错。 Follower 将队列中未处理完的请求消 费完成后 ，写入本地事务日志中。  **广播** ：Leader 可以接受客户端新的事务 Proposal请求，将新的 Proposal请求广 播给所有的 Follower  议来保证分布式事务的最终一致性。 |

###### 15.9 聊聊 zookeeper 选举机制

15.9.1 服务器启动的 Leader 选举

|  |
| --- |
| 1. 服务器 1 ( myid=1 ) 启动 ，当前只有一台服务器， 无法完成 Leader 选举  2. 服务器 2 ( myid=2 ) 启动 ，此时两台服务器能够相互通讯 ，开始进入 Leader 选 举阶段  2.1. 每个服务器发出一个投票  2.2 接受来自各个服务器的投票  2.3. 处理投票  . 优先检查 ZXID。ZXID比较大的服务器优先作为 leader。  . 如果 ZXID 相同的话， 就比较myid，myid 比较大的服务器作为leader。 |

15.9.2 服务器运行期间的 Leader 选举

|  |
| --- |
| 1. 变更状态（LOOKING）  2. 每个服务器发起投票  3. 接受来自各个服务器的投票  4. 处理投票  5. 统计投票  6. 改变服务器状态 |

###### 15.10 算法：给定一个字符串s ，请你找出其中不含有重复字符的最大长度

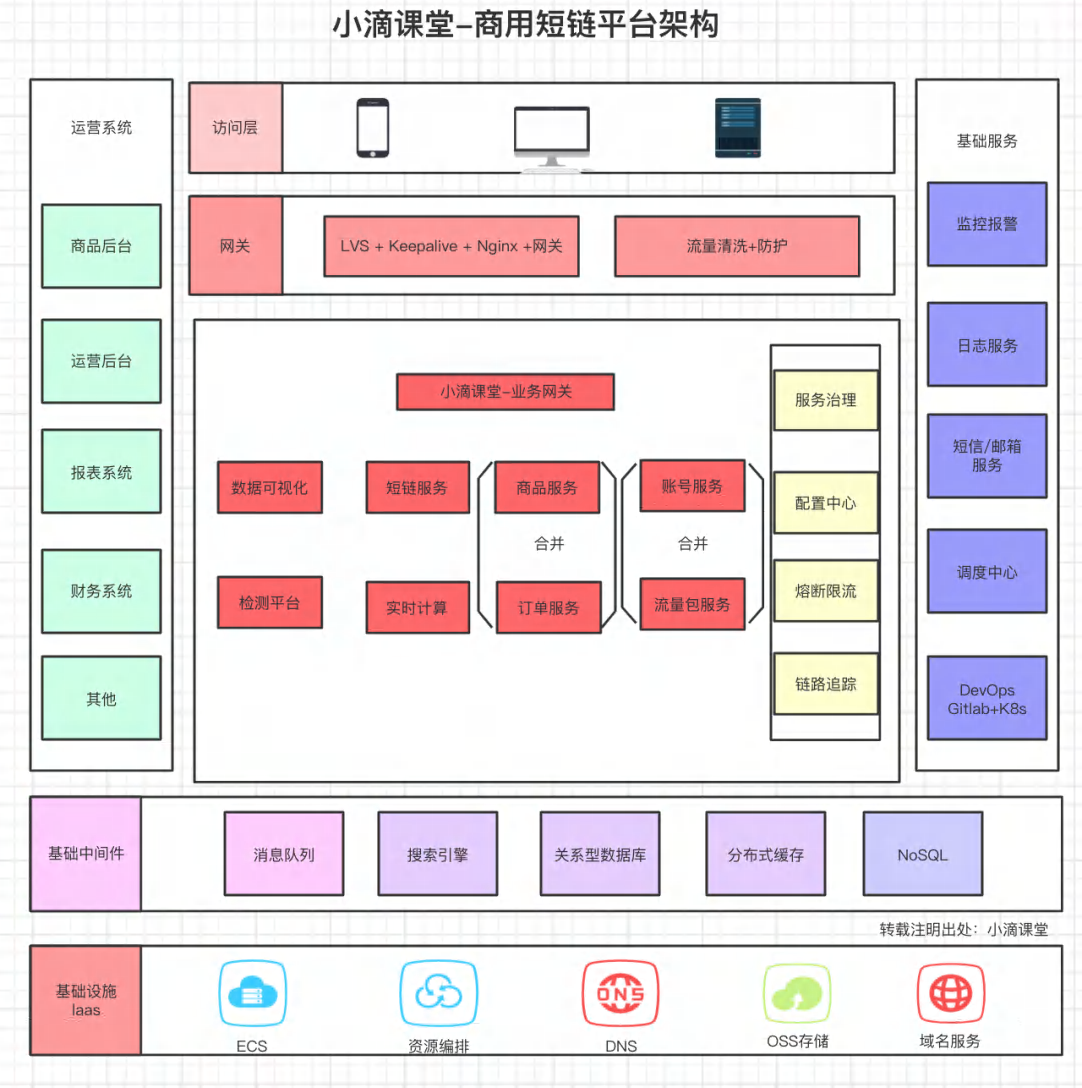
问题遗留：

Kafka的分片消费

[深入理解JVM虚拟机——JVM参数以及调优实战 (baidu.com)](https://baijiahao.baidu.com/s?id=1747377064336158073&wfr=spider&for=pc)

项目描述：

#### 一：海量数据大课



|  |  |  |
| --- | --- | --- |
| **第一章 小滴课堂-海量数据处理商用短链平台大课介绍** | **账号模块**：  1.图形验证码防刷；  2.短信验证码接入；  3.高并发下新用户免费流量表处理；  4.OOS分布式文件存储；  **流量表模块**：  1.海数下流量表过期处理；2.  2.海树下每日流量包更新方案；  3.高并发下流量表扣减；  4.高并发-多场景分布式事务+锁，  5.复杂业务逻辑处理；冷热数据归档处理  **商品和订单模块**：  1.AOP应用+自定义注解防虫提交设；  2.Sharding Sphere从0到1到多案例；  3.订单数据快照+分库分表；  4.分库分表后C端解析和B端查询处理解决方案一  **多渠道支付模块：**  1.方新版微信扫描支付V3接口文档对接；  **2.**PC端支付宝支付扫描对接；  3.多渠道支付回调通知+异步MQ承接  **短链服务**：  1.多案例短链设计方案+落地；  2.高性能短链解析；  3.多案例数据上报设计和方案对比；  4.C端解析+B端查询分库分表常见解决方案；  5.海数分库分表处理；  6.分库分表后C端解析和B端查询处理解决方案二；  7.分库分表后短链CURD[冗余双写]分布式事务解决方案；8.单一Partition Key做分库分表解决方案三；  9.短链库表数据库[免迁移扩容]解决方案四  **实时计算服务:**  1.从0~1掌握ClickHouse新一代OLAP数据库(P2)  2.Kafka接入组件封装+lp获取地理位置信息库使用(P2)  3.后端工程师角度进阶数据仓库+分层+大数据领域技术视野( P1)  4.Flink+Kafka短链数据实时计算+多维度数据处理 (P1)  5.Async异步关联查询+多维度宽表拓展( P1)  6.Flink多流合井+DWS层整合Click House存储( P1) | |
| 第1集 海量数据处理商用短链平台大课介绍 |
| 第2集 商用短链平台技术栈介绍和观看形式-更新周期 |
| 第3集 海量数据处理商用短链平台项目核心亮点《上》 |
| 第4集 海量数据处理商用短链平台项目核心亮点《下》 |
| 第5集 大课解决的问题和跳槽职业发展规划 |
| **第二章 不在其位要谋其政，技术Leader能力模型提升** |  |  |
| 第1集 互联网大厂里 技术Leader的能力模型-不单写代码 |  |  |
| 第2集 技术人的产品运营能力提升-竞品分析 |  |  |
| 第3集 跳出自己的技术思维模型-上司给你团队安排任务 |  |  |
| **第三章 商用短链平台介绍和技术Leader常用方法论介绍** | 1.公司电商产品推⼴、业务活动⻚、⼴告落地⻚ 缺少实时【数据反馈和渠道效果分析】  2.⽼项⽬业务推⼴【没⼈维护，⽆法做埋点】需要统计效果  3.APP和营销活动发送营销短信链接过⻓，【浪费短信发送费⽤】 |  |
| 第1集 商用短链平台业务介绍+需求背景说明 |  |
| 第2集 为什么需要学方法论 避免无章法做事和跳槽面试 |  |
| 第3集 公司行业前景分析-PEST方法论-选择大于努力 |  |
| 第4集 技术Leader必备方法论-用户增长的数据分析模型AARRR |  |
| 第5集 技术Leader必备方法论-SWOT态势分析法-个人能力与技术解决方案 |  |  |
| 第6集 技术Leader必备方法论SMART衡量需求、工作的利器 |  |  |
| **第四章 商用短链平台-功能需求介绍和微服务拆分讲解** | 1.dcloud-common 2.dcloud-app; 3.dcloud-account 4.dcloud-data ;  5 dcloud-gateway; 6.dcloud-link; 7.dcloud-shop  JDK11  SpringBoot 2.5.5  SpringCloud 2020.0.4  AlibabaCloud 2021.1  Sharding-JDBC 4.1.1  Mysql 8.0  Nacos 2.0.2  Redis 6.2.4  RabbitQM 3.8.15  Kafka : wurstmeister/kafka:2.13-2.7.0 |  |
| 第1集 学以致用-商用短链平台需求文档拆分和总结 |  |
| 第2集 商用短链平台-微服务拆分和技术栈版本说明 |  |
| 第3集 商用短链平台-业务架构图讲解 | 见上 |  |
| **第五章 阿里云Linux服务器选配和常用中间件环境安装** | 环境安装 |  |
| 第1集 云服务器配置和搭建Docker相关讲解 |  |
| 第2集 云服务器基础设施安装之Mysql8.0+Redis6.X安装 |  |
| 第3集 云服务器基础设施安装之Nacos2.x+Mysql8配置持久化-避坑 |  |
| 第4集 云服务器基础设施安装之RabbitMQ安装 |  |
| **第六章 短链平台项目创建+git代码管理+开发分层规范讲解** |  |  |
| 第1集 短链平台实战-Maven聚合工程创建微服务项目 |  |  |
| 第2集 Gitee仓库介绍和项目纳入版本管理+ignore文件配置 |  |  |
| 第3集 阿里编码规范里面Manager分层介绍-和开发规范说明 | 1.N方库说明 2.各个层级约束规范 |  |
| 第4集【重要】dcloud-common通用模块配置使用实战和新版改动 | **第5集**1.统⼀业务状态码 BizCodeEnum开发  2.接⼝统⼀协议 JsonData⼯具类开发  @Data  @AllArgsConstructor  @NoArgsConstructor |  |
| 第5集 统一接口响应协议-响应工具类封装 |  |
| 第6集 微服务自定义全局异常+处理器handler开发 | ⾃定义全局异常:extends RuntimeException  ⾃定义异常处理器:@ControllerAdvice |  |
| 第7集 common通用工具和时间格式化工具类讲解 | 1.时间格式化⼯具类封装  2.Json序列化⼯具类封装 |  |
| **第七章 商用短链平台实战-账号微服务+流量包设计** |  |  |
| 第1集 账号微服务和流量包数据库表+索引规范讲解《上》 | 1.主键：pd\_字段名 2.唯一索引 uk\_字段名  3.普通索引：idx\_字段名 |  |
| 第2集 账号微服务和流量包数据库表+索引规范讲解《下》 | 海量数据下每天免费次数怎么更新？  海量数据付费流量套餐包每天次数限制怎么更新？  ⾼性能扣减流量包设计怎么做？  流量包数据更新处理-⾼并发下分布式事务怎么解决？ |  |
| 第3集 账号微服务-流量包业务模型概念补充 |  |
| 第4集 项目引入Mybatis-plus-generator代码自动生成工具 | Mybatis plus配置控制台打印⽇志 |  |
| 第5集 账号微服务注册Nacos+配置文件增加 | @MapperScan("net.xdclass.mapper")  @EnableTransactionManagement（事务）  @EnableFeignClients  @EnableDiscoveryClient  @SpringBootApplication |  |
| **第八章 账号微服务注册模块+短信验证码+阿里云OSS开发实战** |  |  |
| 第1集 账号微服务注册功能和流程介绍 | 1.密码不能使⽤简单的MD5加密  2.异步+池化思想 |  |
| 第2集 第三方短信验证码平台接入申请操作指引 |  |  |
| 第3集 账号微服务短信验证码发送工具类封装实战 |  |  |
| 第4集 账号微服务短信验证码发送工具类单元测试 |  |  |
| **第九章 架构核心技术-池化思想-异步结合 性能优化最佳实践《上》** |  |  |
| 第1集 接口压测和常用压力测试工具对比 |  |  |
| 第2集 Jmeter5.X基础功能组件介绍+线程组和Sampler |  |  |
| 第3集 调用第三方服务组件改造+Jmeter5.x性能压测实践 |  |  |
| 第4集 高并发下异步请求解决方案- @Async注解应用实战 |  |  |
| 第5集 异步调用-压测高QPS后的背后原因和问题拆解 |  |  |
| 第6集 【底层原理】Async+ThreadPoolTaskExecutor自定义线程池进阶实战 |  |  |
| 第7集 ThreadPoolTaskExecutor线程池的面试题你知道怎么回答不 |  |  |
| 第8集 实践出真知-线程池多参数调整-性能压测+现象对比分析 |  |  |
| **第十章 架构核心技术-池化思想-异步结合 性能优化最佳实践《下》** |  |  |
| 第1集 RestTemplate里面的存在的问题你知道多少- Broken pipe错误 |  |  |
| 第2集 高性能RestTemplate连接池封装配置实战 |  |  |
| 第3集 【10倍+QPS提升】Jmeter5.x压测 优化后RestTemplate前后性能对比 |  |  |
| **第十一章 账号微服务-发送短信验证码-池化+异步结合最佳实践** |  |  |
| 第1集 调用第三方短信验证码组件性能优化实战 |  |  |
| 第2集 说说小滴课堂之前的被薅羊毛的事情+防范解决方案 |  |  |
| 第3集 图形验证码开发之谷歌kaptcha引入 |  |  |
| 第4集 池化思想应用-Redis6.X配置连接池实战 |  |  |
| 第5集 账号微服务开发图形验证码加入缓存+Try-with-resource知识巩固 |  |  |
| 第6集 账号微服务之注册短信验证码接口开发 |  |  |
| 第7集 关于注册短信验证码防刷设计方案你能想到几个 |  |  |
| **第十二章账号微服务-阿里云OSS接入实战** |  |  |
| 第1集 分布式文件存储业界常见解决方案介绍 |  |  |
| 第2集 阿里云OSS分布式对象存储介绍开通 |  |  |
| 第3集 权限知识 RBAC-ACL模式应用之阿里云RAM访问控制 |  |  |
| 第4集 阿里云OSS客户端SDK集成和上传组件功能开发 |  |  |
| 第5集 账号微服务头像上传阿里云OSS接口和PostMan测试 |  |  |
| **第十三章 账号微服务注册-登录功能开发完善** |  |  |
| 第1集 账号微服务注册功能业务介绍和代码编写 |  |  |
| 第2集 注册手机号唯一性保证方案和作业-分库分表下的思考 |  |  |
| 第3集 账号微服务开发之登录模块逻辑和解密 |  |  |
| 第4集 分布式应用下登录检验解决方案 JWT讲解 |  |  |
| 第5集 登录校验Json Web Token实战之封装通用方法 |  |  |
| 第6集 请求路径调整+账号微服务注册登录全流程测试 |  |  |
| 第7集 账号微服务之通用登录拦截器开发和用户信息传递 |  |  |
| 第8集 登录拦截器InterceptorConfig拦截和放行路径开发配置 |  |  |
| **第十四章 初恋的感觉-海量数据下的分库分表知识阶段一** |  |  |
| 第1集 账号微服务里面的流量包业务模型梳理和需求讲解 |  |  |
| 第2集【面试题】业务增长-数据库性能优化思路讲解 |  |  |
| 第3集 走进Mysql数据库分库分表后带来的优点和缺点《上》 |  |  |
| 第4集 走进Mysql数据库分库分表后带来的优点和缺点《下》 |  |  |
| 第5集 海量数据处理之Mysql【垂直分表-垂直分库】讲解 |  |  |
| 第6集 海量数据处理之Mysql【水平分表-水平分库】讲解 |  |  |
| **第十五章 如漆似胶-海量数据下的分库分表策略讲解** |  |  |
| 第1集 Mysql数据库水平分库分表常见策略介绍-range |  |  |
| 第2集 Mysql数据库水平分库分表策略介绍-Range延伸进阶 |  |  |
| 第3集 Mysql数据库水平分库分表策略介绍-Hash取模 |  |  |
| **第十六章 热恋的感觉-海量数据下的分库分表技术栈讲解** |  |  |
| 第1集 大话业界常见数据库分库分表中间件介绍 |  |  |
| 第2集 分库分表中间件Apache ShardingSphere急速认知 |  |  |
| 第3集 分库分表和Sharding-Jdbc常见概念术语讲解 |  |  |
| 第4集 分库分表和Sharding-Jdbc常见分片算法讲解 |  |  |
| **第十七章 流量包模块-海量数据下的分库分表《青铜玩法》** |  |  |
| 第1集 账号微服务-流量包模块水平分表需求讲解和开发 |  |  |
| 第2集 账号微服务-流量包模块水平分表策略配置和测试实战 |  |  |
| 第3集 分库分表暴露的问题-ID冲突和分布式id生成介绍 |  |  |
| 第4集 小D-带你彻底掌握分布式 ID 生成算法Snowflake原理 |  |  |
| 第5集 分布式ID生成器Snowflake里面的坑你是否知道 |  |  |
| 第6集 分布式ID生成器Snowflake自定义wrokId实战 |  |  |
| 第7集 shardingjdbc-Snowflake时间回拨问题解决和封装ID生成器 |  |  |
| **第十八章 短链服务-业务需求和短链码解决方案讲解** |  |  |
| 第1集 短链服务介绍和应用场景讲解 |  |  |
| 第2集 需求出发-带你详细一个短链的生命周期 |  |  |
| 第3集 短链服务生成短链URL的问题你能想到多少 |  |  |
| 第4集 短链服务问题解决方案讲解-业务关系+跳转问题 |  |  |
| 第5集 短链服务问题解决方案讲解-短链码生成解决方案《上》 |  |  |
| 第6集 【重要】敏感数据+自增ID暴露的商业秘密 |  |  |
| 第7集 短链服务问题解决方案讲解-短链码生成解决方案《下》 |  |  |
| **第十九章 短链服务-Murmur哈希算法封装组件** |  |  |
| 第1集 Guava框架里面的Murmur哈希算法测试 |  |  |
| 第2集 短链生成组件ShortLinkComponent封装 |  |  |
| 第3集 组件ShortLinkComponent测试和疑惑解答 |  |  |
| **第二十章 短链服务-数据库表建立和业务代码开发** |  |  |
| 第1集 数据库表模型讲解-短链分组和短链 |  |  |
| 第2集 MybatisPlus逆向工具生成短链服务相关java对象 |  |  |
| 第3集 同学遇到的坑-配置文件修改-yml转properties |  |  |
| 第4集短链分组管理-CURD接口开发实战《上》 |  |  |
| 第5集 短链分组管理-CURD接口开发实战《下》 |  |  |
| 第6集 短链分组管理-水平分库分表配置实战《青铜玩法》 |  |  |
| **第二十一章 短链服务分库分表-如何做到扩容免数据迁移《黄金玩法》** |  |  |
| 第1集 短链服务-ShortLink分库分表解决方案讲解《青铜》 |  |  |
| 第2集 短链服务-分库分表扩容免数据迁移解决方案讲解《黄金玩法》 |  |  |
| 第3集 短链服务-分库分表相关库表建立 |  |  |
| 第4集 短链服务水平分库分表实战-标准分片策略-精准分片算法《上》 |  |  |
| 第5集 短链服务水平分库分表实战-标准分片策略-精准分片算法《下》 |  |  |
| 第6集 短链服务-短链码配置生成库表位实战 |  |  |
| 第7集 短链服务-Manager层模块CRUD开发 |  |  |
| 第8集 短链服务-自定义分库分表策略单元测试实战 |  |  |
| 第9集 加权负载均衡思想应用-数据库表扩容-数据不均匀问题解决方案 |  |  |
| **第二十二章 短链服务-分库分表多维度查询解决方案《钻石玩法》** |  |  |
| 第1集 短链服务-短链URL跳转302跳转接口开发实战 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |