# Java面试全集

## 基础篇

1. 基本功
2. 面向对象的特征

封装,继承,多态

1. final, finally, finalize 的区别

1. final修饰符,加了就是终态,不可变 2.finally是异常中跟在trycatch后面的,是异常捕获处理结束后会继续执行的阶段,就算trycatch里有return也会执行fanally,finalize是gc垃圾回收的关键字

1. int 和 Integer 有什么区别

Int是基础类型非引用类型,integer是包装类,是对象引用类型,有自己的内部调用方法,

1. 重载和重写的区别

重载是同一个类中,相同的方法名,但是不同的参数类型或个数的方法,重写是子类继承父类,然后覆盖掉父类的同名方法

1. 抽象类和接口有什么区别

抽象类是一个类,可以没得抽象方法,用abstract修饰,类中的抽象方法,子类可以不用重写实现,一个类只能继承一个抽象类,但可以实现多个接口,接口用interface来修饰 里面不能见非静态变量,只有抽象方法,接口也可以继承接口

1. 说说反射的用途及实现

反射能在程序运行期间获得对象的类对象,能直接获取类的变量及方法的属性,可以通过 对象.getclass(),类名.class()的方式获取类对象

1. 说说自定义注解的场景及实现

自定义注解文件名为annotation,用interface修饰,而且要加入元注解,并且要写个工具类,通过反射来进行注解解析

1. HTTP 请求的 GET 与 POST 方式的区

Get方法会把参数通过?和&的方式直接把参数拼在地址中的,安全性差,而post却可以通过requestbody把参数封装起来传到后台,安全性高些

1. session 与 cookie 区别

Session是服务端用在存放信息数据等的,放在服务端的,一次会话结束后自动关闭,cookie是服务的用来存放信息的,一般是存放在浏览器缓存中的.

1. session 分布式处理

建一个主从模式+哨兵的高可用redis服务集群,用来存放session,然后自己实现session manager用来管理session的生命周期

1. JDBC 流程

先加入jar包到项目中,然后创建连接,编写sql语句,然后执行语句,获取返回结果,关闭连接.

1. MVC 设计思想

m指的是model是用来存放数据的模型,v是view视图,用来展示后台传来的数据是显示层,c是controller控制层,用来控制视图与模型之间的连接切换的,mvc主要是为了视图与模型的分离,降低之间的耦合度.以达到更高的灵活性及复用性.

1. equals 与 == 的区别

==比较的是对象的地址引用,如果俩个地址指向的是同一片内存地址则返回true.

Equals比较的俩个对象的实际内容是否相等,string的equals 重写了object的equals,他把俩个字符串拆分成char[] ,先比较俩个字符数组,然后循环用==对比每一个char,如果不同就会返回false

## 集合

1. List 和 Set 区别

List和set都是继承collection ,list为有序且可重复的集合,set为无序且不可重复的集合

1. List 和 Map 区别

List是有序直接存对象,通过下标可以获取指定对象,map是无序通过key-value的方式存放对象的,通过get(key)的方式获取指定对象

1. Arraylist 与 LinkedList 区别

Arraylist是基于可变数组来实现的,创建一个空Arraylist时的默认大小是10.每次通过add方法添加数据达到长度上限时会以1.5倍长度自动扩容,arraylist通过下标查询快.但是在指定位置下标插入或删除慢,linkedlist是基于链表来实现的,所以没有默认长度的限制,无需扩容,它因为是通过链表的连接点依次往后遍历,所以查询慢,但是插入数据很快,应该只需要修改连接点的引用即可.

1. ArrayList 与 Vector 区别

Arraylist 为单线程的,线程不安全,但是效率更高,vecter为线程安全的,但是效率低

1. HashMap 和 Hashtable 的区别

Hashmap没加同步锁,所以只在单线程安全,并且可以存null值,hashtable有同步锁,多线程下也是安全的,不能存null值,不过在单线程下 hashmap效率高点

1. HashSet 和 HashMap 区别
2. HashMap 和 ConcurrentHashMap 的区别

主要的区别就是hashmap是线程不安全的,而concurrenthashmap基于锁分离技术,实现了并发安全

1. HashMap 的工作原理及代码实现

Hashmap实际是一个链表散列的数据结构,是数组加链表的组合实现,底层是一个数组结构,每一个数组里存的一个链表

1. ConcurrentHashMap 的工作原理及代码实现

Concurrenthashmap,主要是使用了锁分离技术,使用了多个锁控制hash表的不同部分的修改操作,每一个部分都是一个小的hashtable,所以并发是安全的.

## 线程

1. 创建线程的方式及实现

继承thread类,或者实现runable接口,需要多线程实现的代码放在run方法里

1. sleep() 、join（）、yield（）有什么区别

Sleep是休眠当前的线程,但是不释放对象锁,并要指定休眠多少毫秒,join是在一个线程中加入另一个线程,本线程进入阻塞状态,当加入的线程执行完,本线程才会再次进入runable状态,yield方法会使线程让出cpu资源进入阻塞状态,等待下一次资源分配再进入运行,

1. 说说 CountDownLatch 原理
2. 说说 CyclicBarrier 原理
3. 说说 Semaphore 原理
4. 说说 Exchanger 原理
5. 说说 CountDownLatch 与 CyclicBarrier 区别
6. ThreadLocal 原理分析

Threadlocal本质是为每一个线程访问的并发访问的数据提供一个副本,多消耗了内存,但是提高了性能

1. 讲讲线程池的实现原理
2. 线程池的几种方式
3. 线程的生命周期

通过new 等方法进入新建状态,start方法进入runnable状态,当cpu调度就绪后,进入运行状态,当执行sleep,wait,join,yield等方法进入阻塞状态,当run方法里面代码执行完毕,或抛异常了,线程就进入死亡状态.

1. 锁机制
2. 说说线程安全问题

在多个线程同时对一个对象进行修过等操作时,会出现不符合业务逻辑的结果,比如库存变成负数,一般会把对象加锁,来达到多线程的数据安全操作.

1. volatile 实现原理
2. synchronize 实现原理
3. synchronized 与 lock 的区别
4. CAS 乐观锁
5. ABA 问题
6. 乐观锁的业务场景及实现方式

## 核心篇

1. 数据存储
2. MySQL 索引使用的注意事项
3. 说说反模式设计

反模式设计就是反三范式设计,三范式的主要作用是用时间来换取空间,而反范式就是用空间来换取时间,以到达牺牲物理空间的代价来达到更快速的查询

1. 说说分库与分表设计
2. 分库与分表带来的分布式困境与应对之策
3. 说说 SQL 优化之道

尽量不要使用like ,is null等关键词,能过滤掉的更多的条件放在最后面,能用一条语句查询所有结果的就不要用多条语句分多次查询

1. MySQL 遇到的死锁问题
2. 存储引擎的 InnoDB 与 MyISAM
3. 数据库索引的原理

索引实质是为数据库中表的列排序的一种数据结构,底层是一个b+tree,能加快查询的速度,但是会占用很多内存

1. 为什么要用 B-tree

因为b+tree想对于2叉树而言,同等的数据量会有更少的数节点层级,也就是更矮胖的树,这个查询一个指定的值io的次数就更少,效率更高

1. 聚集索引与非聚集索引的区别

聚集索引是表中的各行的物理顺序和索引的逻辑相同,但是一个表只能有一个,非聚集索引指定表的逻辑顺序,然后数据和索引存在俩个位置

1. limit 20000 加载很慢怎么解决

Top20000

1. 选择合适的分布式主键方案

使用redis的string型存储主键,然后每次存数据到数据库时,调用redis的incre方法会使对应key的值加1然后返回,

1. 选择合适的数据存储方案
2. ObjectId 规则
3. 聊聊 MongoDB 使用场景
4. 倒排索引
5. 聊聊 ElasticSearch 使用场景

## 缓存使用

1. Redis 有哪些类型

Redis有5种数据结构类型,string,list,set,hash(map),zset(sortset)

1. Redis 内部结构
2. 聊聊 Redis 使用场景

Redis主要是作为缓存使用的,比如用来统计网站用户登录的数据

分布式用户session储存,也可以做消息队列来使用

1. Redis 持久化机制

Redis数据先是存放在内存中的,每隔一段时间会把数据刷到硬盘中储存,

1. Redis 如何实现持久化

Redis有俩种持久方案,一种是RDB快照的方式,定期全盘数据从内存存入硬盘,一种是AOF模式,aof的频率更高,更安全,但是占用硬盘更大,RDB效率更高

1. Redis 集群方案与实现

Redis集群一般使用主从复制模式,主服务器负责写,从服务器负责读,并用哨兵模式来应对主服务器宕机的应急处理

1. Redis 为什么是单线程的

采用单线程，避免了不必要的上下文切换和竞争条件，也不存在多进程或者多线程导致的切换而消耗 CPU，不用去考虑各种锁的问题，不存在加锁释放锁操作，没有因为可能出现死锁而导致的性能消耗；

1. 缓存奔溃
2. 缓存降级
3. 使用缓存的合理性问题
4. 消息队列

常用的消息队列有activemq, rabbitmq,因为redis中的list是用链表实现,所以采用左进右出或者右进左出的方式也能实现队列,而且现在redis也默认支持生产者/消费者的 发布订阅模式

1. 消息队列的使用场景

队列能有效的解决分布式的同步通讯的问题,像系统发送邮件给用户,商城系统秒杀时可以使用队列来缓存请求,来缓解服务器的并发压力.防止服务器崩溃

1. 消息的重发补偿解决思路
2. 消息的幂等性解决思路
3. 消息的堆积解决思路

排查是消息生产方重复发送,还是消费方的消息处理逻辑有问题

1. 自己如何实现消息队列

一般是使用链表来实现队列,因为队列的先进先出特性能很好的实现队列,然后需要加入多线程同步锁,让队列中消息为空时进入等待阻塞状态,防止浪费资源

1. 如何保证消息的有序性

链表中的数据默认是先进先出,并且有序的

## 框架篇

1. Spring
2. BeanFactory 和 ApplicationContext 有什么区别

Beanfactory是创建bean实例的工厂,ApplicationContext 是指应用上下文环境

1. Spring Bean 的生命周期
2. Spring IOC 如何实现
3. 说说 Spring AOP
4. Spring AOP 实现原理
5. 动态代理（cglib 与 JDK）
6. Spring 事务实现方式
7. Spring 事务底层原理
8. 如何自定义注解实现功能
9. Spring MVC 运行流程
10. Spring MVC 启动流程
11. Spring 的单例实现原理
12. Spring 框架中用到了哪些设计模式
13. Spring 其他产品（Srping Boot、Spring Cloud、Spring Secuirity、Spring Data、Spring AMQP 等）
14. Netty
15. 为什么选择 Nett

Netty是一个基于NIO的非阻塞式的高性能网络框架,

1. 说说业务中，Netty 的使用场景
2. 原生的 NIO 在 JDK 1.7 版本存在 epoll bug
3. 什么是TCP 粘包/拆包
4. TCP粘包/拆包的解决办法
5. Netty 线程模型
6. 说说 Netty 的零拷贝
7. Netty 内部执行流程
8. Netty 重连实现

## 微服务篇

1. 微服务
2. 前后端分离是如何做的

通过rpc和rest进行远程互相调用

1. 微服务哪些框架

Springcloud生态圈,dubbo加zookeeper

1. 你怎么理解 RPC 框架
2. 说说 RPC 的实现原理
3. 说说 Dubbo 的实现原理
4. 说说 Spring Cloud 的实现原理
5. 你怎么理解 RESTful
6. 说说如何设计一个良好的 API
7. 如何理解 RESTful API 的幂等性
8. 如何保证接口的幂等性
9. 说说 CAP 定理、 BASE 理论
10. 怎么考虑数据一致性问题
11. 说说最终一致性的实现方案
12. 你怎么看待微服务
13. 微服务与 SOA 的区别
14. 如何拆分服务
15. 微服务如何进行数据库管理
16. 如何应对微服务的链式调用异常
17. 对于快速追踪与定位问题
18. 微服务的安全

## 分布式

1. 谈谈业务中使用分布式的场景
2. Session 分布式方案
3. 分布式锁的场景
4. 分布是锁的实现方案
5. 分布式事务
6. 集群与负载均衡的算法与实现
7. 说说分库与分表设计
8. 分库与分表带来的分布式困境与应对之策
9. 安全问题
10. 安全要素与 STRIDE 威胁
11. 防范常见的 Web 攻击
12. 服务端通信安全攻防
13. HTTPS 原理剖析
14. HTTPS 降级攻击
15. 授权与认证
16. 基于角色的访问控制
17. 基于数据的访问控