## 一

1. Java中多态是怎么实现的

静态绑定（编译时绑定）：重载-同名方法不同参数类型和个数

实现原理是动态绑定：它允许基类的指针或引用指向派生类的对象，而在具体访问时实现方法的动态绑定（运行时绑定）

https://blog.csdn.net/sinat\_34311901/article/details/52208124

多态详解https://www.cnblogs.com/mz-lover/p/4786337.html

1. Java中的几种锁

synchronized ：有适应自旋、锁消除、锁粗化、轻量级锁及偏向锁等

当线程通过synchronized等待锁时是不能被Thread.interrupt()中断

ReentrantLock ：公平锁和非公平锁

除了能完成synchronized所能完成的所有工作外，还提供了诸如可响应中断锁(lockInterruptibly)、可轮询锁请求(tryLock)、定时锁等(tryLock(timeout))避免多线程死锁的方法；

Semaphore ：Semaphore主要用于控制当前活动线程数目，就如同停车场系统一般，而Semaphore则相当于看守的人，用于控制总共允许停车的停车位的个数，而对于每辆车来说就如同一个线程，线程需要通过acquire()方法获取许可，而release()释放许可。如果许可数达到最大活动数，那么调用acquire()之后，便进入等待队列，等待已获得许可的线程释放许可，从而使得多线程能够合理的运行。

AtomicInteger：volatile value计算，cas更新数据，效率低于LongAdder

LongAdder: cells[]分段计算，依赖cas,最后sum汇总，效率最高

http://youzhixueyuan.com/4-kinds-of-java-thread-locks.html

1. 数据库隔离级别 脏读 幻读 ACID mysql的隔离级别

https://blog.csdn.net/huwenhu2007/article/details/73718719

1. mysql索引实现，如何解决慢查询

<https://www.cnblogs.com/haiyan123/p/7512567.html>

0.先运行看看是否真的很慢，注意设置SQL\_NO\_CACHE

1.where条件单表查，锁定最小返回记录表。这句话的意思是把查询语句的where都应用到表中返回的记录数最小的表开始查起，单表每个字段分别查询，看哪个字段的区分度最高

2.explain查看执行计划，是否与1预期一致（从锁定记录较少的表开始查询）

3.order by limit 形式的sql语句让排序的表优先查

4.了解业务方使用场景

5.加索引时参照建索引的几大原则 ：

https://www.cnblogs.com/garfieldcgf/p/6378690.html

6.观察结果，不符合预期继续从0分析

1. 数据库锁是怎么实现的

https://blog.csdn.net/C\_J33/article/details/79487941

1. 死锁的条件，进程和线程区别

死锁发生在锁交叉，避免1.避免交叉，缩小锁的粒度，2.使用可重入锁

一个应用程序包括至少一个进程，一个进程包括至少一个线程

锁状态：https://blog.csdn.net/u014484649/article/details/27338819

1. tcp/ip模型，tcp和udp区别

https://blog.csdn.net/yulyu/article/details/69062288

1. Linux查看网络 内存 日志命令

网络：netstat命令用来打印网络连接状况、系统所开放端口、路由表等信息

常用的关于netstat的命令就是这个 netstat -lnp （打印当前系统启动哪些端口）以及 netstat -an （打印网络连接状况）

内存：top, jmap -histo:live [pid], jmap -dump:live [pid]

日志：tail ,head,cat

IO: iostat -x 查看磁盘使用

1. spring中有哪些模块

Core,context, dao,ORM,mvc,web,aop,

1. HashMap和HashTable以及ConcurrentHashMap的区别

Hashmap不同步，允许null键和null value

Hashtable同步，不允许null键和null 值

hashcode生成方式不同：

1.hashtable:取模即除法散列法(key.hashcode & 0x7FFFFFFF) % tab.length

2.Hashmap:1.7:key.hashcode & (length - 1) 1.8:key.(hashCode()) ^ (key.hashCode() >>> 16)

<http://ifeve.com/hashmap-concurrenthashmap-相信看完这篇没人能难住你%ef%bc%81/>

Hashtable是加了synchronize的hashmap

详解:https://www.cnblogs.com/peizhe123/p/5790252.html

1. CAS的底层实现

使用 Unsafe类，unsafe底层实现是汇编的cmpxchg 实现，处理器执行cmpxchg是原子性操作（如果是多处理器，也需要对cmpxchg操作加锁）

1. 谈Java GC

<https://blog.csdn.net/snow_114/article/details/80214332>

<http://iamzhongyong.iteye.com/blog/1447314>

GC1:https://tech.meituan.com/g1.html

1. 栈和队列

https://www.cnblogs.com/CherishFX/p/4608880.html

1. 10万个URL去重

Hash去重，如果是占用内存较大，则分成多个小文件处理

Top K问题，使用堆解决

1. TCP的状态？TCP和UDP的区别？TIME\_WAIT

https://www.cnblogs.com/E-star/p/5572050.html

## 二

1. Volatile关键字

1.volatile保证可见性

2.禁止指令重排序-保证有序性

不能保证原子性-通过synchronize和lock保证原子性

<http://www.importnew.com/24082.html>

实现原理

1 volatile变量进行写操作时，jvm向处理器发送lock前缀指令，将这个变量所在缓存行数据写回到系统内存，确保更新主内存数据

Contented注解来进行缓存行填充（padding）。解决多线程情况下的伪共享冲突问题。

2缓存行更新：每个处理器通过嗅探在总线上传播的数据检查自己的缓存是否过期，，过去则设置为无效，当读取该变量时，强制从系统内存读取数据到缓存

Lock前缀指令实际上相关于一个内存屏障，确保禁止指令重排

1. JVM的垃圾回收算法有哪些

<https://www.cnblogs.com/xing901022/p/7725961.html>

垃圾回收器

串行收集器(Serial)：这种收集器就是以单线程的方式收集，垃圾回收的时候其他线程也不能工作

并行收集器(Parallel)：以多线程的方式进行收集

并发标记清除收集器(CMS:Concurrent Mark Sweep Collector):

大致的流程是：初始标记--并发标记--重新标记-并发清除

G1收集器(Garbage First Collector):

大致的流程是:初始标记--并发标记-最终标记-筛选回收

1. 讲讲JVM内存模型

内存结构：jvm

https://www.cnblogs.com/xing901022/p/7725961.html

内存模型：jmm

<https://www.cnblogs.com/chenyangyao/p/5269622.html>

<http://ifeve.com/java-memory-model-4/>

1. 启动线程有哪几种方式，线程池有哪几种？

1继承extend。

2实现runnable接口，

3实现callable接口，通过futuretask包装器创建

4 使用线程池启动线程

线程池：http://youzhixueyuan.com/use-of-java-thread-pool.html

1. springIOC原理？自己实现IOC要怎么做，哪些步骤？

Ioc 和aop详解 <https://www.cnblogs.com/ITtangtang/p/3978349.html>

应用实例参考项目代码: ProceedingJoinPoint,获取注解和目标类等相关数据，实现业务

1. MongoDB是行存储还是列存储

列存储，查询解析效率较高

1. Hbase，Redis，Nginx？
2. 用过mysql吗？为啥加索引会变快？聚簇型索引和非聚簇型索引的区别？

http://www.cnblogs.com/crazylqy/p/7615457.html

## 三

1. JVM调优，程序挂起后如何排查原因？

<https://blog.csdn.net/paincupid/article/details/51724273>

从高内存，cpu占用入手

1. redis如何实现分布式锁

http://ifeve.com/基于redis的分布式锁/#more-40175

1. 聊到dubbo,zookeeper了解吗？netty实战过吗？
2. 你参与的项目，画出集群部署图，以及服务之间的关联，设计的核心点。

# 基础知识

1. 深拷贝和浅拷贝

深拷贝和浅拷贝都是对象拷贝；

1.1浅拷贝仅仅复制所考虑的对象，而不复制它所引用的对象（默认clone）；

1.2 深拷贝把要复制的对象所引用的对象都复制了一遍（重写clone方法/反序列化对象）。

详解 <https://blog.csdn.net/baiye_xing/article/details/71788741>

2. spring 控制bean不放到容器里的方法（也就是定义bean的三种方式：注解，xml和基于java的bean定义（BeansConfiguration））

3. final关键字

final : “最终”修饰符：

1，可以修饰类，函数，变量。

2，被final修饰的类不可以被继承。作用：为了避免被继承，被子类复写功能。

3，被final修饰的方法不可以被复写。

4，被final修饰的变量是一个常量只能赋值一次，既可以修饰成员变量，又可以修饰局部变量。

当在描述事物时，一些数据的出现值是固定的，那么这时为了增强阅读性，都给这些值起个名字。方便于阅读。

而这个值不需要改变，所以加上final修饰。作为常量：常量的书写规范所有字母都大写，如果由多个单词组成，单词间通过\_连接。

5，内部类定义在类中的局部位置上是，只能访问该局部被final修饰的局部变量。

4.内部类

内部类：特点：

1，内部类可以直接访问外部类中的成员，包括私有。  
（原因：内部类中持有了一个外部类的引用，格式：外部类名.this）  
 2，外部类要访问内部类，必须建立内部类对象。  
 3, 当内部类在成员位置上，就可以被成员修饰符所修饰。

1. lamda

<https://www.cnblogs.com/aoeiuv/p/5911692.html>

# 网络编程

# 操作系统

# Jdk IO，UTIL源码