# Java相关

## 乐观悲观锁的设计，如何保证原子性，解决的问题；

<https://blog.csdn.net/liao1989126/article/details/52196160>

<https://blog.csdn.net/rexct392358928/article/details/52230737>

<https://blog.csdn.net/zhangwj0101/article/details/50946054>

<https://blog.csdn.net/a362212624/article/details/80534865>

悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作。

乐观锁：假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。 乐观锁不能解决脏读的问题。

乐观锁：版本、时间戳

## char和double的字节，以及在内存的分布是怎样；

<https://blog.csdn.net/shenhuan1104/article/details/51900030>

ByteOrder.nativeOrder()

## 对象内存布局，然后讲下对象的死亡过程？

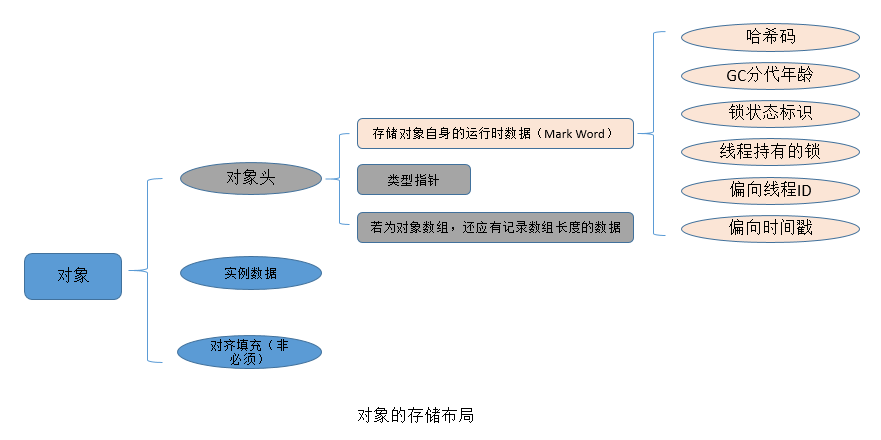
<https://www.cnblogs.com/z-sm/p/7145180.html?utm_source=itdadao&utm_medium=referral>

<https://blog.csdn.net/aitangyong/article/details/46416667>

<http://www.importnew.com/19429.html>

<https://www.cnblogs.com/plxx/p/4642405.html>

### 对象布局



### 对象的存活与死亡

<https://www.cnblogs.com/lrh-xl/p/5292495.html>

<https://blog.csdn.net/yozhangxin/article/details/78830375>

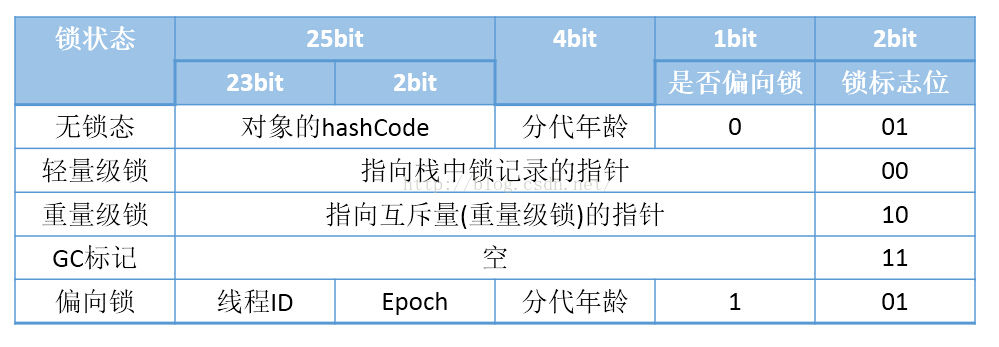
<https://blog.csdn.net/LiuXudongHnu/article/details/62447943>

<https://www.cnblogs.com/LoganChen/p/6829165.html>

即使在可达性分析算法中不可达的对象，也并非是非死不可得，这时候它们是暂时处于缓刑阶段，要真正宣告一个对象死亡，至少要经历两次标记过程：如果在对象进行可达性分析后发现没有与GC Roots相连接的引用链，那么它将会被第一次标记并且进行一次筛选，筛选的条件是此对象是否有必要执行fianlize()方法。当对象没有覆盖finalize()方法，或者finalize()方法已经被虚拟机调用过，虚拟机将这两种情况都视为没有必要执行。即，如果对象没有覆盖finalize()方法，finalize()方法已经被JVM调用过，那么没有必要执行finalize()方法。（注：fianlize()方法作用：GC确定不存在该对象的更多引用时，由GC调用该对象的fianlize方法）。若判定为有必要执行finalize()方法，那么这个对象将放置在一个叫做F－Queue的队列中，并在稍后由一个由虚拟机自动建立的、低优先级的Finalizer线程去执行它。

## 对象头，详细讲下；

<https://blog.csdn.net/zhoufanyang_china/article/details/54601311>



## sync原理详细，sync内抛异常会怎样，死锁吗？还是释放掉？怎么排查死锁？死锁会怎样？有没有什么更好的替代方案？

## 详细讲一下集合，HashSet源码，HashMap源码，如果要线程安全需要怎么做？

## 多线程是解决什么问题的？线程池解决什么问题？

## 线程池，如何设计的，里面的参数有多少种，里面的工作队列和线程队列是怎样的结构，如果给你，怎样设计线程池？

## AQS原理，ReentranLock源码，设计原理，整体过程。

## 继续聊多线程源码，sync原理，然后一个场景设计题；

## float f = 1.4f;double d = 1.4d; 与 float f = 1.5f;double d = 1.5d; 是否为true，内存是怎样的；

<https://www.cnblogs.com/chenjfblog/p/7737332.html>



## split的源码，split("a|b|c");得出多少个数组；

## 把所有认识熟用的JUC( java.util.concurrent(简称JUC)包)下的类写出来，讲下使用，然后讲下原生的线程操作;

## 开闭原则，解析工厂方法模式，建造者模式，区别。手撸出来。

<https://blog.csdn.net/carson_ho/article/details/54910597>

## 讲下JVM的大页模式，JVM内存模型;

<https://blog.csdn.net/wender/article/details/78888886>

<https://www.cnblogs.com/dingyingsi/p/3760447.html>

<https://blog.csdn.net/zero__007/article/details/52926366>

## 什么是敏捷开发，防御性编程，并行编程。Team Leader的思考;

<https://www.cnblogs.com/jinguangguo/p/3304941.html>

## 逃逸分析是什么，作用是什么，用途是什么;

<https://blog.csdn.net/legend_x/article/details/17636237>

<https://blog.csdn.net/iceman1952/article/details/6238536>

逃逸分析（Escape Analysis）是分析 指针动态范围 的方法，这同 指针分析 和 外形分析 相关联

## 怎么认为一个类是线程安全？线程安全的定义是什么？Java有多少个关键字进行同步？为什么这样设计？（聊了一大堆，一堆为什么）；

## 两个线程设计题。记得一个是：t1,t2,t3，让t1，t2执行完才执行t3，原生实现。

## 写个后缀表达式，为什么要设计后缀表达式，有什么好处？然后写下中缀。

<https://blog.csdn.net/lcy6239/article/details/52239832>

<https://blog.csdn.net/wu_lai_314/article/details/8274978>

<https://blog.csdn.net/xiazdong/article/details/7272693>

后缀表达式的特点就是计算机运算非常方便，需要用到栈；计算机处理过程只需要顺序读入，如果遇到数字，则放入栈中，如果是运算符，则将两个栈中数字取出进行运算；

## 我看你做过性能优化，比如你怎么分析项目里面的OOM的，内存泄露呢？详细说思路;

## 说下多线程，我们什么时候需要分析线程数，怎么分析，分析什么因素;

## 抽象方法和类方法的区别，static的抽象方法可以吗？

## 说下Java的克隆体系;

## 涉及OOM、JVM优化、源码问题、数据库优化、多线程等问题;

## CPU高？什么情况CPU高？解决什么问题？

## 你有遇到过临界区问题吗？有遇到过吗？你在项目遇到这个问题是怎样解决的？

## volatile关键字作用;

## Java的多态怎么实现;

<https://blog.csdn.net/sinat_34311901/article/details/52208124>

<https://blog.csdn.net/huangrunqing/article/details/51996424>

<https://www.cnblogs.com/ShaneZhang/p/4972550.html>

## 解释一下自旋;

<https://blog.csdn.net/raintungli/article/details/7109182>

<https://blog.csdn.net/sunp823/article/details/49886051>

## 解释一下信号量;

## 什么情况下会触发类加载；

<https://blog.csdn.net/liang_70121385/article/details/52496028>

<http://www.importnew.com/6579.html>

创建类的实例

访问类的静态变量(除常量【被final修辞的静态变量】原因:常量一种特殊的变量，因为编译器把他们当作值(value)而不是域(field)来对待。如果你的代码中用到了常变量(constant variable)，编译器并不会生成字节码来从对象中载入域的值，而是直接把这个值插入到字节码中。这是一种很有用的优化，但是如果你需要改变final域的值那么每一块用到那个域的代码都需要重新编译。

访问类的静态方法

反射如(Class.forName("my.xyz.Test"))

当初始化一个类时，发现其父类还未初始化，则先出发父类的初始化

虚拟机启动时，定义了main()方法的那个类先初始化

## Java内存抖动严重，优化的思路；

<https://blog.csdn.net/hpc19950723/article/details/53581143>

<https://blog.csdn.net/chenliguan/article/details/53220578>

<https://www.jianshu.com/p/69e6f894c698>

# 数据库相关

## SQL优化思路，联合索引与底层树结构的映像关系，索引结构（B+、B-），为什么用这样的结构；

<https://www.cnblogs.com/xyxxs/p/4440187.html>

## 讲下MySQL的集群？集群遇到过什么问题？sql的优化？

## 你目前为止遇到的最大数据量是多少？知道100万时候怎么设计吗？1000万呢？过几十亿呢？

## MySQL有多少个参数可调，除了最大连接数。全部列出来，一个个分析。

## 聊下优化过的索引，怎么优化;

## 红黑树和平衡树的区别，为什么数据库不用红黑树;

## mysql有哪些锁，意向锁有什么用;

<https://blog.csdn.net/arkblue/article/details/53895150>

<https://www.jianshu.com/p/e937830bc2de>

## 数据库高并发下的优化思路;

<https://blog.csdn.net/kidoo1012/article/details/54691561>

<https://blog.csdn.net/Mis_lixiaoli/article/details/6331312>

<https://www.cnblogs.com/xuecanmeng/p/5489869.html>

<http://mengqingyu.iteye.com/blog/1830572>

索引

读写分离

SQL优化

## 数据库什么情况下索引会失效;

<https://www.cnblogs.com/zhangweizhong/p/4424393.html>

<https://blog.csdn.net/sdujava2011/article/details/78310426>

<https://blog.csdn.net/colin_liu2009/article/details/7301089>

Like

不等

组合索引

Or？

# 数据结构和操作系统相关

## 数据结构学过吧，聊一下？学过什么结构？讲下树和队列？B树呢？

## 操作系统学过吧，聊一下？讲一下系统内存是怎样的？分段分页虚拟内存？

## 页面置换算法呢？多少种？有最优的置换算法吗？

<https://blog.csdn.net/fengjie_123/article/details/51416138>

<https://blog.csdn.net/c_flybird/article/details/50512552>

<https://www.cnblogs.com/huxinga/p/6701105.html>

<https://blog.csdn.net/wangsifu2009/article/details/6757352>

先入先出法（FIFO）

最优置换算法（OPT）

最久未使用算法（LRU）

第二次机会算法（SCR）

## 你学过什么课程？然后聊下操作系统，内核、用户之类。

## 反转链表手撸;

<https://blog.csdn.net/fx677588/article/details/72357389>

## 快排，给一串数组，把具体每次patition写下，最终结果也写45, 32, 41, 35, 38, 20, 50;

## 一个整数status, 判断第K个比特位是否为比特1;

## 把递归实现的快排改成非递归，你知道非递归有什么好处吗;

## 举例使用分治思想的算法;

# 网络相关

## 讲下请求头细节？

<https://blog.csdn.net/ythunder/article/details/65664309>

<https://blog.csdn.net/u012398362/article/details/52382923>

<https://www.cnblogs.com/shenpengyan/p/5912567.html>

IP头部信息：

头部长度：通常20字节，有选项时更长，总共不超过60字节。

IP数据报长度：65535字节。



TCP/IP协议族中另一个重要协议：TCP协议，在传输层。

特点：可靠性。通过连接管理（三握四挥），序列号，确认号，拥塞控制，重传控制来保证可靠性。

头部长度：一般为20字节，选项最多40字节，限制60字节。



## Http和Https？Http1.0,1.1,2.0，讲下长连接和短连接？Https是怎样的？如果我篡改了公钥呢？怎么防止？

<https://blog.csdn.net/renwotao2009/article/details/51068676>

<https://blog.csdn.net/jasonjwl/article/details/50985271>

## Get和Post，讲下区别，要我模拟出抓包来。

<https://www.cnblogs.com/huaxingtianxia/p/5895236.html>

<https://blog.csdn.net/yipiankongbai/article/details/24025633>

<http://www.w3school.com.cn/tags/html_ref_httpmethods.asp>

GET 方法

请注意，查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：

/test/demo\_form.asp?name1=value1&name2=value2

有关 GET 请求的其他一些注释：

GET 请求可被缓存

GET 请求保留在浏览器历史记录中

GET 请求可被收藏为书签

GET 请求不应在处理敏感数据时使用

GET 请求有长度限制

GET 请求只应当用于取回数据

POST 方法

请注意，查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的：

POST /test/demo\_form.asp HTTP/1.1

Host: w3schools.com

name1=value1&name2=value2

有关 POST 请求的其他一些注释：

POST 请求不会被缓存

POST 请求不会保留在浏览器历史记录中

POST 不能被收藏为书签

POST 请求对数据长度没有要求

## 详细讲下Cookie和Session，Token，OAuth2.0协议;

## 拥塞算法知道吗？哪些，分别怎样？

<https://blog.csdn.net/smilesundream/article/details/71149434>

<https://blog.csdn.net/dog250/article/details/55652190>

慢启动

## 学过计算机网络是吧？socket熟悉吗？对它的读写缓冲区有理解吗？怎么的？那滑动窗口是怎样的？为什么这样设计？

## 再聊下Http的Http basic authentication;

<https://www.cnblogs.com/alex-13/p/4845658.html>

<https://blog.csdn.net/u014006264/article/details/46813925>

## Https的过程;

<https://www.cnblogs.com/mddblog/p/6948980.html>

<https://blog.csdn.net/woshinannan741/article/details/71550436>

<https://blog.csdn.net/u011803341/article/details/79708886>

<https://blog.csdn.net/kobejayandy/article/details/52433660>

# 框架相关

聊下Spring源码，知道多少，都聊一下；

聊下Spring注解，@Autowire，@Resource，以及他们的解析过程；

聊一下架构，接入层架构，服务层架构。聊下技术栈，Spring Boot，Spring Cloud、Docker；

Spring ioc的具体优势，和直接New一个对象有什么区别;

Servlet生命周期，是否单例，为什么是单例;

Spring Mvc初始化过程；

# 分布式相关

## 多少种RPC框架？

<https://blog.csdn.net/songguo_color/article/details/72229124>

<https://www.cnblogs.com/wzj4858/p/7856353.html>

<https://blog.csdn.net/jek123456/article/details/70208049>

RMI

## 一致性哈希是干嘛的？

## 搭建高并发高可用系统需要怎样设计？考虑哪些东西，有多少说多少。

<https://blog.csdn.net/a_fengzi_code_110/article/details/52433651>

<https://www.cnblogs.com/zhudongchang/p/6826602.html>

<https://www.cnblogs.com/gdsblog/p/7128497.html>

<https://blog.csdn.net/yangbutao>

<https://blog.csdn.net/yangbutao/article/details/16896669>

<https://blog.csdn.net/yangbutao/article/details/12242441>

空间换时间

水平扩展

## 你对缓存有什么理解？缓存是解决什么问题？后端缓存有哪些，分别解决什么问题？

<https://www.cnblogs.com/dinglang/p/6133501.html>

<https://blog.csdn.net/ghj1976/article/details/1768676>

redis

<https://www.cnblogs.com/xuliangxing/p/7151812.html>

<https://www.cnblogs.com/java-zhao/p/5205771.html>

<https://yq.aliyun.com/articles/38441>

redis 提供 6种数据淘汰策略：

volatile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰

no-enviction（驱逐）：禁止驱逐数据

## 聊一下分布式锁；

<https://www.cnblogs.com/austinspark-jessylu/p/8043726.html>

基于数据库实现分布式锁 基于缓存（redis，memcached，tair）实现分布式锁 基于Zookeeper实现分布式锁

基于数据库表

基于数据库排它锁

基于redis的setnx

基于zookeeper

## 你是怎么设计系统缓存的，为什么，什么场景；

## 也来说下，**削峰的多种实现**，Redis？MQ？

<https://blog.csdn.net/u011676417/article/details/70168194>

<https://blog.csdn.net/u010954257/article/details/54178173>

## 为什么用mq就能削峰？解决什么问题？

<https://www.cnblogs.com/zrbfree/p/6709099.html>

<https://www.cnblogs.com/syjkfind/p/7979713.html>

# 设计题

## 有几台机器存储着几亿淘宝搜索日志，你只有一台2g的电脑，怎么选出搜索热度最高的十个搜索关键词;

分而治之/hash映射 + hash统计 + 堆/快速/归并排序；

双层桶划分

Bloom filter/Bitmap；

Trie树/数据库/倒排索引；

外排序；

分布式处理之Hadoop/Mapreduce。

## 如何设计算法压缩一段URL;

<https://blog.csdn.net/u012913404/article/details/51098388>

<https://blog.csdn.net/wolflz/article/details/46342501>

<https://blog.csdn.net/blueZhangFun/article/details/49201075>

<https://blog.csdn.net/it_man/article/details/8973156>

<https://blog.csdn.net/wgw335363240/article/details/6568794>

<https://blog.csdn.net/zhenyongyuan123/article/details/51829171>

<https://www.zhihu.com/question/29270034>

<https://hufangyun.com/2017/short-url/>

直接提供一个发号器服务提供短网址服务

## 有一个页面能同时展示两个广告，现在有五个广告，设计算法使五个广告展示概率为1:2:3:4:5；

<https://www.cnblogs.com/superdo/p/4923975.html>

## 有25匹马，五个赛道，用最少比赛次数将25匹马排序；

<https://blog.csdn.net/cmsbupt/article/details/17404183>

<https://blog.csdn.net/HarbinZJU/article/details/5838898>

<https://blog.csdn.net/chen825919148/article/details/8053980>

<https://blog.csdn.net/u013028556/article/details/54907968>

# 其他相关

## Tomcat缓存，聊下缓存的整体理解，知道多少种缓存；

<https://blog.csdn.net/redarmy_chen/article/details/7032671>

<https://blog.csdn.net/he20101020/article/details/17390817>

<https://blog.csdn.net/lq305/article/details/6079455>

## 解释下Mucene原理，倒排索引，怎样进行中文分词，基于什么进行分词；

<https://blog.csdn.net/tele/article/details/250456>

<https://blog.csdn.net/wm_1991/article/details/52224897>

## TopN的大数据量题；

## 你对接入层要思考什么东西？遇到过哪些问题？搭建系统要考量哪些因素？

## 然后项目问题，优化问题；

## 熟悉maven是吧？我们来聊下Maven的源码原理，**Maven冲突的时候，怎么选择依赖包，我们怎么查，我们遇到两个不一样的版本**，我们应该如何去选择，为什么？

<https://www.cnblogs.com/godtrue/p/6220512.html>

<https://www.jianshu.com/p/3c554c9e959f>

<https://blog.csdn.net/sinat_39789638/article/details/78005945>

<https://blog.csdn.net/zhou2s_101216/article/details/78207232>

<https://blog.csdn.net/cll1224666878/article/details/79006328>

1、第一声明优先原则：

在pom.xml配置文件中，如果有两个名称相同版本不同的依赖声明，那么先写的会生效。

所以，先声明自己要用的版本的jar包即可。

2、路径近者优先：

直接依赖优先于传递依赖，如果传递依赖的jar包版本冲突了，那么可以自己声明一个指定版本的依赖jar，即可解决冲突。

3、排出原则：

传递依赖冲突时，可以在不需要的jar的传递依赖中声明排除，从而解决冲突。

4、版本锁定原则（最常使用）

在配置文件pom.xml中先声明要使用哪个版本的相应jar包，声明后其他版本的jar包一律不依赖。解决了依赖冲突。

## 项目如何分组，性能优化小组应该做哪些;

## 我们来说下接入层的搭建，认知分析;

## 问下项目的系统构建，思考，为什么这样构建？

## 如何判断一段代码的好坏;