这里罗列出的设计模式一般是最常考的，最好都看看。

4 JVM调优

6 责任链模式

7 单例模式

9 类的初始化部分

10 对象的创建过程

11 双亲委派模型

12 finally的执行时机

13 七大设计原则

14 静态代理和动态代理？

15 迭代器模式

20 内部类

21 equals， hashCode， ==

22 什么是不可变对象

23 Java的序列化/反序列化机制

28 Set (HashSet，LinkedHashSet，TreeSet)

29 Map及Map的三种常用实现[链表数组HashMap，LinkedHashMap(HashMap的子类，带头结点的双向链表 + HashMap)，基于红黑树的TreeMap(对key排序)]

31 ConcurrentHashMap

32 ConcurrentHashMap，HashMap 与 HashTable

34 BIO, NIO, AIO

36、System.gc() 与 Object.finalli()ze

37、函数式接口

38、JDK 8 新特性

39、双亲委派模型 与 反双亲委派模型

40 为什么要有Survivor区？” 以及“为什么要设置两个Survivor区？”

42、ThreadLocal 的内存泄漏问题

脑图：

**数据结构与算法**

**操作系统**

（1）基础

协程

<https://blog.csdn.net/weixin_42898819/article/details/82012503>

<https://blog.csdn.net/zhangbijun1230/article/details/80474988> 故事

协程关键词：1 生产者消费者问题，线程协作需要加锁以及通知。而协程不用操作系统。更加轻量级。开销极小。2 协程只用一个线程完成 3 不像线程那样要依靠wait notify来完成，而暂停是真的暂停了，不是让出CPU。消费者一直在yield那里等待，直到数据到来。然后重新进入上次暂停的地方。 4 高并发，高扩展性。

静态链接与动态链接 <https://blog.csdn.net/kang___xi/article/details/80210717>

https://blog.csdn.net/u012414189/article/details/84111636

（什么是静态链接？什么是动态链接？ 各自的优缺点？）

   静态链接和动态链接两者最大的区别就在于链接的时机不一样，静态链接是在形成可执行程序前，而动态链接的进行则是在程序执行时。

在我们的实际开发中，不可能将所有代码放在一个源文件中，所以会出现多个源文件，而且多个源文件之间不是独立的，而会存在多种依赖关系

一是浪费空间，因为每个可执行程序中对所有需要的目标文件都要有一份副本，所以如果多个程序对同一个目标文件都有依赖，如多个程序中都调用了printf()函数，则这多个程序中都含有printf.o，所以同一个目标文件都在内存存在多个副本；另一方面就是更新比较困难，因为每当库函数的代码修改了，这个时候就需要重新进行编译链接形成可执行程序。但是静态链接的优点就是，在可执行程序中已经具备了所有执行程序所需要的任何东西，在执行的时候运行速度快。

假设现在有两个程序program1.o和program2.o，这两者共用同一个库lib.o,假设首先运行程序program1，系统首先加载program1.o，当系统发现program1.o中用到了lib.o，即program1.o依赖于lib.o，那么系统接着加载lib.o，如果program1.o和lib.o还依赖于其他目标文件，则依次全部加载到内存中。当program2运行时，同样的加载program2.o，然后发现program2.o依赖于lib.o，但是此时lib.o已经存在于内存中，这个时候就不再进行重新加载，而是将内存中已经存在的lib.o映射到program2的虚拟地址空间中，从而进行链接（这个链接过程和静态链接类似）形成可执行程序。

（2）Linux

硬链接与软链接 <https://blog.csdn.net/wdr2003/article/details/79700457>

<https://blog.csdn.net/qq_22313585/article/details/79114702>

1硬链接相当于一个文件有多个文件名（像是实际文件的别名）。相当于产生一个和原来一模一样的文件，这个文件有个不同的文件名。这个文件的更改也会同步到原来的文件上。软连接相当于产生一个新文件，访问这个文件就会跳转到另一个文件（像指针）。类似于快捷方式2 硬链接必须在一个文件系统中，软链接可以跨文件系统。3 硬链接是防止真实文件被误操作。文件只有所有的别名被删除，链接数为0，才会删除原始文件。

硬连接指通过索引节点来进行的连接。在Linux的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(Inode Number)。  
在Linux中，多个文件名指向同一索引节点是存在的。一般这种连接就是硬连接。

文件系统 暂时看书为主，主要看文件结构。

（一个分区通常能建立几个文件系统？文件系统的组成是什么样的？文件读取过程是什么样的，有哪两种？磁盘碎片？一个block能放几个文件？一个文件能放几个block？inode所包含的一些信息？一个inode对应几个文件？如果文件非常大，inode该如何引用它？ Linux的目录配置是什么样的？如果Linux突然断电会如何？目录是怎么记录文件信息的？）

* inode：一个文件占用一个 inode，记录文件的属性，同时记录此文件的内容所在的 block 编号；
* block：记录文件的内容，文件太大时，会占用多个 block。

除此之外还包括：

* superblock：记录文件系统的整体信息，包括 inode 和 block 的总量、使用量、剩余量，以及文件系统的格式与相关信息等；
* 建立一个目录时，会分配一个 inode 与至少一个 block。block 记录的内容是目录下所有文件的 inode 编号以及文件名。

常用命令

显示目录和文件的命令

   Ls：用于查看所有文件夹的命令。

   Dir：用于显示指定文件夹和目录的命令   Tree： 以树状图列出目录内容

   Du：显示目录或文件大小

创建和删除，重命名，复制文件的命令

  Touch：创建一个新的文件

   Vi:创建一个新的文件

   Rm：删除文件或目录

Mkdir 创建目录。

   Mv：重命名或移动文件的命令

   Cp：复制命令

Init：改变系统运行级别

Shutdown：-r 关机后立即重启

**僵尸进程与孤儿进程 （什么是僵尸进程与孤儿进程？如何处理？）**

一个进程在调用exit命令结束自己的生命的时候，其实它并没有真正的被销毁，而是留下一个称为僵尸进程（Zombie）的数据结构（系统调用exit，它的作用是使进程退出，但也仅仅限于将一个正常的进程变成一个僵尸进程，并不能将其完全销毁）。在Linux进程的状态中，僵尸进程是非常特殊的一种，它已经放弃了几乎所有内存空间，没有任何可执行代码，也不能被调度，仅仅在进程列表中保留一个位置，记载该进程的退出状态等信息供其他进程收集，除此之外，僵尸进程不再占有任何内存空间。 **也就是占个PCB**

（如果一个子进程改变了它的状态，父进程中会发生什么？sigchld信号包含什么内容？父进程调用wait()后要怎样才能回收死掉的子进程？）

**Sigchld信号**

当一个子进程改变了它的状态时（停止运行，继续运行或者退出），有两件事会发生在父进程中：

* 得到 SIGCHLD 信号；
* waitpid() 或者 wait() 调用会返回。

其中子进程发送的 SIGCHLD 信号包含了子进程的信息，比如进程 ID、进程状态、进程使用 CPU 的时间等。

在子进程退出时，它的进程描述符不会立即释放，这是为了让父进程得到子进程信息，父进程通过 wait() 和 waitpid() 来获得一个已经退出的子进程的信息。

**Wait()**

调用wait() 会一直阻塞，直到收到一个子进程退出的 SIGCHLD 信号，之后 wait() 函数会销毁子进程并返回。

如果成功，返回被收集的子进程的进程 ID；如果调用进程没有子进程，调用就会失败，此时返回 -1，同时 errno 被置为 ECHILD。

https://github.com/CyC2018/CS-Notes/blob/master/docs/notes/Linux.md

<https://blog.csdn.net/Eunice_fan1207/article/details/81387417> 最佳

https://baike.baidu.com/item/僵尸进程/1036577?fr=aladdin

<https://baike.baidu.com/item/孤儿进程/16751450>

如何处理僵尸进程？

要消灭系统中大量的僵尸进程，只需要将其父进程杀死，此时僵尸进程就会变成孤儿进程，从而被 init 进程所收养，这样 init 进程就会释放所有的僵尸进程所占有的资源，从而结束僵尸进程。

如果父进程并不是很繁忙我们就可以通过直接调用wait/waitpid来等待子进程的结束。当然这会导致父进程被挂起。比如第一种情况中(父进程循环了三次，子进程循环了五次，子进程先结束，父进程调用wait等待子进程)父进程循环结束后并不会结束，而是被挂起等待子进程的结束。

但是如果父进程很忙。我们不希望父进程一直被挂起直到子进程的结束

那么我们可以使用信号函数sigaction为SIGCHLD设置wait处理函数。这样子进程结束后，父进程就会收到子进程结束的信号。并调用wait回收子进程的资源

<https://www.cnblogs.com/wuchanming/p/4020463.html>

**网络**

1. 基础

重要性主要为二档的

以太网 https://blog.csdn.net/conleyfree/article/details/50890505

集线器，交换器，路由器的区别P56P122P194 IP数据报

TCP、UDP报文组成？

UDP：源端口，目的端口，长度，检验和

TCP：源端口 目的端口 序号 确认号 检验和 窗口

ICMP协议（是什么，） http://m.elecfans.com/article/574448.html

1. Http

1档：连接管理 （几种连接是什么？并行连接一定更快吗？并行连接是让哪边允许多条？执行多条什么？所以一般限制并行连接的条数） https://www.jianshu.com/p/1102f00002ff Https 版本比较

Http与FTP <https://blog.csdn.net/only_musm/article/details/78983364>

(HTTP是什么？FTP是什么？ 两者的异同?) 如果在同一个会话期间，用户还需要传输另一个文件，FTP会如何做？ FTP和HTTP哪个是无状态，哪个是有状态，什么意思？

2档：缓存（链接见脑图）https://www.cnblogs.com/chenqf/p/6386163.html

（什么是HTTP缓存？有哪几种？Expires和Cache-Control中的max-age区别？两种缓存机制的区别?）

Https/2

1. Socket

1档：I/O模型 多路复用（什么是IO多路复用？select存在的问题？poll修复的问题？epoll？）

https://blog.csdn.net/qq\_28098067/article/details/80537564

**第一种方法就是最传统的多进程并发模型 (每进来一个新的I/O流会分配一个新的进程管理。)第二种方法就是I/O多路复用 (单个线程，通过记录跟踪每个I/O流(sock)的状态，来同时管理多个I/O流 。)**

多看看这个图就了解了。提醒下，ngnix会有很多链接进来， epoll会把他们都监视起来，然后像拨开关一样，谁有数据就拨向谁，然后调用相应的代码处理。

* select 如果任何一个sock(I/O stream)出现了数据，select 仅仅会返回，但是并不会告诉你是那个sock上有数据，于是你只能自己一个一个的找，10几个sock可能还好，要是几万的sock每次都找一遍，这个无谓的开销就颇有海天盛筵的豪气了。
* select 只能监视1024个链接， 这个跟草榴没啥关系哦，linux 定义在头文件中的，参见FD\_SETSIZE。

select 不是线程安全的

https://www.jianshu.com/p/db5da880154a

2档：Java NIO

<https://www.cnblogs.com/xiaoxi/p/6534926.html>

Java提供了一系列改进的输入/输出处理的新特性，被统称为NIO(即New I/O)

亲自看看吧。里面干货挺多

什么是通道？缓冲区？

<https://www.cnblogs.com/xiaoxi/p/6576588.html>

关于java流的几个概念：IO、BIO、NIO、AIO

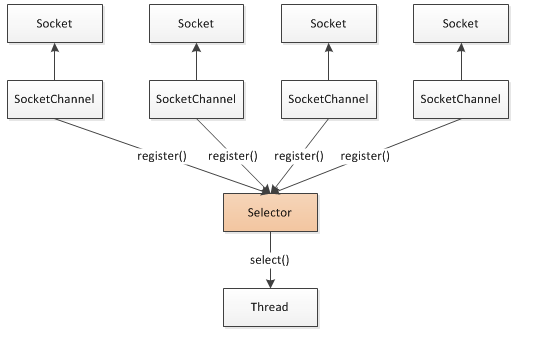
BIO：就是指IO，即传统的Blocking IO,即同步并阻塞的IO

即一个请求对应一个线程，如果线程数不够连接则会等待空余线程或者拒绝连接

NIO：新的IO,即New IO或者Non-Blocking IO,即同步不阻塞的IO

它加入了Buffer、Channel、Selector等概念。它是基于事件驱动的，采用了Reactor模式，它使用一个线程管理所有的socket通道，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。它的特点是要不断主动地去询问数据有没有处理完，一般只适用于连接数目较大但连接时间短的应用，如聊天应用等。

AIO



AIO的做法是，每个水壶上装一个开关，当水开了以后会提醒对应的线程去处理。  
NIO的做法是，叫一个线程不停的循环观察每一个水壶，根据每个水壶当前的状态去处理。  
BIO的做法是，叫一个线程停留在一个水壶那，直到这个水壶烧开，才去处理下一个水壶。

<https://blog.csdn.net/luzhensmart/article/details/82230076>最佳

<https://blog.csdn.net/wufaliang003/article/details/80573405>

（Buffer、Channel、Selector等概念？）

1）缓冲区buffer

    buffer是一个对象，包含了读取和写入的数据，在nio中，所有的数据都是通过缓冲区来处理的。在写入数据时，也是写入到缓冲区中。任何时候访问NIO中的数据，都是通过缓冲区进行操作。

**缓冲区实际是一个数组结构**，并提供了对数据结构化访问以及维护读写位置等信息。

    8种基本类型都有相应的缓冲区：ByteBuffe、CharBuffer、 ShortBuffer、IntBuffer、LongBuffer、FloatBuffer、DoubleBuffer。他们实现了相同的接口：Buffer。

（2）通道channel

    我们对数据的读取和写入都要通过channel，它就像水管一样，是一个通道。**通道不同于流的地方就是通道是双向的，可以用于读、写和同时读写操作。**

**底层的操作系统的通道一般都是全双工的，所以全双工的Channel比流能更好的映射底层操作系统的API。**

数据库

1. SQL
2. 系统原理

1档：ACID（事务四大原则） MVCC

https://blog.csdn.net/w2064004678/article/details/83012387

SQL与NoSQL （为什么产生了NOSQL？NOSQL的好处与缺点？一般评论都放在哪里，为什么？NOSQL的应用场景？为什么外键会大大提高扩充表的成本？）

1. 扩展困难：由于存在类似Join这样多表查询机制，使得数据库在扩展方面很艰难;

2. 读写慢：这种情况主要发生在数据量达到一定规模时由于关系型数据库的系统逻辑非常复杂，使得其非常容易发生死锁等的并发问题，所以导致其读写速度下滑非常严重;

3. 成本高：企业级数据库的License价格很惊人，并且随着系统的规模，而不断上升;

4. 有限的支撑容量：现有关系型解决方案还无法支撑Google这样海量的数据存储;

1.简单的扩展：典型例子是Cassandra，由于其架构是类似于经典的P2P，所以能通过轻松地添加新的节点来扩展这个集群;

2.快速的读写：主要例子有Redis，由于其逻辑简单，而且纯内存操作，使得其性能非常出色，单节点每秒可以处理超过10万次读写操作;

3.低廉的成本：这是大多数分布式数据库共有的特点，因为主要都是开源软件，没有昂贵的License成本;

NoSQL并不是任何场景，NoSQL都要优于关系型数据库。下面我们来具体聊聊，什么时候使用NoSQL比较给力：

1) 数据库表schema经常变化

2)数据库表字段是复杂数据类型

对于复杂数据类型，比如SQL Sever提供了可扩展性的支持，像xml类型的字段。

3)高并发数据库请求

此类应用常见于web2.0的网站，很多应用对于数据一致性要求很低，而关系型数据库的事务以及大表join反而成了”性能杀手”。

4)海量数据的分布式存储

海量数据的存储如果选用大型商用数据，如Oracle，那么整个解决方案的成本是非常高的，要花很多钱在软硬件上。NoSQL分布式存储，可以部署在廉价的硬件上，是一个性价比非常高的解决方案。

https://blog.csdn.net/xgf415/article/details/75370783

2档：封锁（就是锁）

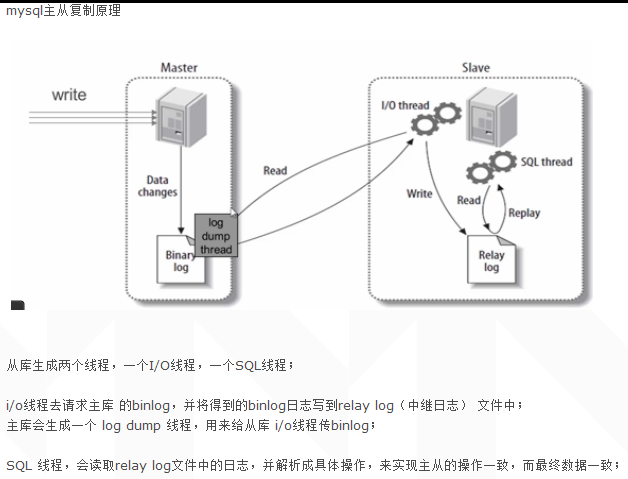
1. MYSQL

1档：查询优化

2档：主从复制（主从复制是什么？主从复制的形式？用途是什么？主从复制的原理？主从复制存在的问题以及解决方式？主从复制和读写分离的区别？）

<https://www.cnblogs.com/Aiapple/p/5792939.html>

mysql主从复制用途

* 实时灾备，用于故障切换
* 读写分离，提供查询服务
* 备份，避免影响业务
* 

日志（日志有哪几种）https://www.cnblogs.com/jevo/p/3281139.html

1. Redis

1档：使用场景（三个特点？） https://blog.csdn.net/qq\_30683329/article/details/80530246 与Memchached的比较（如何保证都是热点数据？Memecache数据存储到哪里？） <https://blog.csdn.net/m0_37450089/article/details/78740692>

1. **数据高并发的读写**
2. [**海量数据**](https://www.baidu.com/s?wd=%E6%B5%B7%E9%87%8F%E6%95%B0%E6%8D%AE&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)**的读写**
3. **对扩展性要求高的数据**

**1. 使用redis有哪些好处？**

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

**2. redis相比memcached有哪些优势？**

(1) memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型

(2) redis的速度比memcached快很多

(3) redis可以持久化其数据

2档：

<https://blog.csdn.net/a724888/article/details/79086600>

https://github.com/CyC2018/CS-Notes/blob/master/docs/notes/Redis.md

字典和跳跃表 （字典的数据结构？字典重哈希的过程是什么样的？字典的两个哈希表都时刻发挥作用吗？跳跃表的数据结构是什么样的？跳跃表的查找过程是什么样样的？）

RDB与AOF持久化机制（这两种机制中文名是什么？分别的原理？RDB存在的问题是什么？RDB能保证数据完全不丢失吗？AOF的同步选项有哪些？当AOF文件过大的时候会怎么处理？） <https://blog.csdn.net/a724888/article/details/79349833>

数据淘汰机制

**事件驱动模型（为什么Redis这么快？为什么Redis是单线程的？）** https://www.jianshu.com/p/fda1d6096d48

Epoll https://blog.csdn.net/u011671986/article/details/79449853

主从复制（主从复制是什么？Redis复制工作过程？三种复制的机制？） https://blog.csdn.net/a724888/article/details/79349755

集群与分布式（集群一定是一主多从吗？集群的瓶颈是什么？） https://blog.csdn.net/yejingtao703/article/details/78484151

事务 https://blog.csdn.net/weixin\_39040059/article/details/79120459

https://blog.csdn.net/a724888/article/details/79349255

线程安全问题

反射学得太浅！！ 此外最好根据面经进一步加深基础知识。

此外项目的介绍或许应该更加深一点？

面向对象

装饰者模式

https://blog.csdn.net/a724888/article/details/71104023

迭代器模式

https://blog.csdn.net/zhengzhb/article/details/7610745

MVC模式

https://www.cnblogs.com/xiaoxi/p/6164383.html

**Linux的五种IO模型**

http://mp.weixin.qq.com/s?\_\_biz=MzU3OTYxOTU4NA==&mid=2247483776&idx=1&sn=1681aa6e527e146eac5700620f301e20&chksm=fd621fefca1596f965383bc6224b9bf4aa228e28a530a0c515bfe455bf4a7e8231657e002a62&mpshare=1&scene=23&srcid=0130BMxQMZl00NCnmGZKYy1t#rd

怎样的过程算作一种IO，关于钓鱼的比喻，三个空间？ 哪五种IO模型，各自是怎样的IO过程？ 应用进程通过系统调用什么接收数据？ 这四种是同步的还是异步的，为什么？

同步异步区别

https://blog.csdn.net/qq\_37954693/article/details/80720458

系统设计（很大块的内容）

Java

中间件

其他知识

零散知识：

int和string如何相互转化 https://www.cnblogs.com/bluestorm/p/3698810.html

final http://how2j.cn/k/interface-inheritance/interface-inheritance-final/313.html

Scanner <http://how2j.cn/k/operator/operator-scanner/658.html>

结束外部循环（break结束的是哪个循环。如何用if结束外层循环） http://how2j.cn/k/control-flow/control-flow-break-out/279.html

所以如果指定了数组的内容，就不能同时设置数组的长度

 int[] c = new int[3]{100,102,444,836,3236};

数组Arrays.equals()比较的是什么？ Arrays 有反转的方法吗?

<http://how2j.cn/k/array/array-arrays/516.html>

内部类 http://how2j.cn/k/interface-inheritance/interface-inheritance-inner-class/322.html#nowhere

字符串怎么转化为int，int怎么转化为字符串？

http://how2j.cn/k/number-string/number-string-parse/317.html

--四舍五入, 随机数，开方，次方，π，自然常数

http://how2j.cn/k/number-string/number-string-math/319.html#nowhere

--格式化输出

Character的常见方法

字符串的格式化?? 操纵字符串？？

str1  ==  str2 和 是否是同一个对象的特例？内容是否相同? http://how2j.cn/k/number-string/number-string-compare/326.html

如果要字符串反转，要如何做？

重点：进程调度算法的特点以及使用场景 虚拟内存的作用

页面置换算法的原理 页面置换算法的原理，特别是 LRU 的实现原理，最好能手写，再说明它在 Redis 等作为缓存置 换算法

★★☆ 缓存 的Cache-Control 字段，特别是 Expires 和 max-age 的区别。ETag 验证原理https://github.com/CyC2018/CS-Notes/blob/master/docs/notes/HTTP.md

* ★★★ 长连接与短连接原理以及使用场景，流水线。
* ★★★ HTTP 存在的安全性问题，以及 HTTPs 的加密、认证和完整性保护作用。
* ★★☆ HTTP/1.x 的缺陷，以及 HTTP/2 的特点。
* ★★★ HTTP/1.1 的特性。
* ★★★ select、poll、epoll 的原理、比较、以及使用场景；epoll 的水平触发与边缘触发。

Spring知识 1 动态加载？ 2 spring的 Autowired 和Resoured区别

https://www.cnblogs.com/xiaoxi/p/5935009.html

3 如果配置文件名字更改会如何？

忽略的知识

哪些集合类是线程安全的？ https://blog.csdn.net/xiaojie\_570/article/details/80872338

为什么wait要放在块里？ Redis数据结构源码（还是按照面经复习靠谱。）？ Redis主要的数据结构？

Hashmap新特性？

复习的总内容 ：

1 算法 66 + leetcode100

2 面试1 +面试2 +最近的新内容 红加黑 以及以前的内容总览，比如how2j ，几个面试的

3 SQL语句使用，默写

4 jvm虚拟机 jvm调优，以及参数 类加载机制

5 Spring IOC 的书写

6 项目内容的介绍与表述

7 Spring MVC 和spring的常见问题 （spring的优点？springAOP是什么，好处？动态代理是什么？静态代理是什么？springIOC的理解？springIOC的三种注入方式？spring用到了哪些设计模式？）

8 天猫网站项目的SSM框架，各个配置文件作用。

spring

https://blog.csdn.net/a745233700/article/details/80959716

springMVC

https://blog.csdn.net/a745233700/article/details/80963758

CurrentHashMap 知识 如何保证数据的一致性问题。

项目的具体功能要再看一遍。

Crud sql 是什么意思？

各种排序都要过一遍。 根据网易

B+树的数据结构。查询过程

hash

大数据的内容 。 回去了解currentMap的具体实现。

高并发编程的学习。

堆的参数？

<https://www.cnblogs.com/marcotan/p/4256885.html>

双亲委派模型 https://www.jianshu.com/p/6dbe64a333e7

接下来要进一步加强的：、

1 工作时的技术栈

2 高并发编程的学习

3 高并发的测压，部署

4 spring框架的进一步掌握，

5 算法题，起码要掌握学习的部分。

6 面试技巧的整理

8 SSM框架总结

1Category.xml是做什么的，含有哪些内容？

2 applicationContext.xml 是做什么的？为什么说这个是Mabatis和spring的集成文件？

3 web.xml含有什么内容？

4 springmvc.xml含有什么内容？

5 配置文件可以改名吗?

5 扫一眼lmpl controller mapper等的文件，不用记，留个印象。

6 数据库连接池，线程池的参数有哪些？

7 表结构整理。 对外键的理解有偏差。

1 （1）封装SQL语句 （2）规定namespace命名空间，也就是Mapper的路径

2 （1）启动对注解的识别 （2）对jdbc.properties的引用 （3）配置数据库内容，得到数据源（这是什么意思?）也就是配置数据库连接池 （4）配置sessionFactory（这是mabatis的核心配置），相当于beanFactory和Mabatis的集成 （5）Mapper的扫描。

3 spring springmvc的核心配置。以及中文配置。 是这两个的爹，spring优势Mabatis的爹。从这里可以看出spring和springmvc实际上并没有真正直接的交互。

4 （1）前端控制器，也就是对注解的识别（2）视图定位，视图定位到某个文件夹（3）对上传文件的解析。 蛮简单的。

5 可以，看web.xml的配置文件，配置了这两个配置文件的文件名，你只要改了，就可以改成其他的文件名。

6 url 用户名密码 获取连接超时时间 一个连接在池中最小生存时间

初始化多少线程 如果10个线程不够用，就会增加到15个 如果经过多少秒，多出来的还没接到活，就回收多出来的。

public class INode {

public boolean isLeaf=false;

public INode parent=null;

public ArrayList<Float> keys=new ArrayList<Float>();

public ArrayList<INode> childNodes=new ArrayList<INode>();

}

public class TreeLeaf extends INode {

public ArrayList<Object> values=new ArrayList<Object>();

public TreeLeaf rightBrother=null;

}

阿里的问题：如果绕过按钮，要怎么解决安全性问题?

要习惯阿里的打断说话的面试方式。实习生要考察到什么程度，他们自己清楚，他们就是要考察你挖掘的深度，以及自己的想法，具体场景，如何解决。如果你没深挖，只是做了，想得不够多，平时没有积累，那么你就凉凉。我一直求快，所以一直以来，都没能解决深度的问题。对于很多深挖的点还是积累不够，或者说我只是追求现有的解决方案。记住知识点以及定义。也就是说我的项目是可以进一步深挖的。包括安全性问题。

还有面试的时候不要说那么多废话，要清楚，面试官一方面想要的是答案，另一方面想要的是你的思考，以及学习的主动性。

不是所有时候都能让你用上Redis的。不用的时候要如何解决。

Redis雪崩的问题。首先是Key没有了，还有突然涌向服务器

Redis保持数据一致性的问题。，有些时候数据密集，有些时候数据不密集，所以要解决这个问题，如何能够实时地让数据更新。

Spring动态代理的问题。

Spring它的内核究竟是什么样的，这个如果弄清楚了，说，熟悉Spring源码，这样的话一定是一个超级加分项。

Spring哪里使用了反射。动态代理，静态代理。

工厂模式，抽象工厂模式，以及动态工厂。如何完成？

所以之后如果努力的话，可以把这些知识进一步扩充。不过。。。这么说吧，这些亮点，如果现在准备的话，恐怕有点晚了，因为阿里这种级别的厂，在秋招再进的话真的是千难万难。春招是最适合进大厂的时机，而关于spring的理解，这种筹码很大程度上最春招用是最好的，秋招用就显得积累了。

Haoxiang\_Chen