

封神总结：抖音、腾讯、阿里、美团社招服务端开发岗位硬核面试

原创

站在风口的java 于 2022-02-28 15:59:51 发布 65 收藏 1

分类专栏：[面试](#) 文章标签：[面试](#) [java](#) [程序人生](#)

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/m0_65634190/article/details/123184767

版权



[面试](#) 专栏收录该内容

28 篇文章 0 订阅

订阅专栏

前言

先报一下身份，笔者的师弟小民同学。小民同学应届校招从事服务端开发，参加校招面试以来，共面四个公司的服务端开发岗位：

- 抖音，IES互娱
- 腾讯，PCG 应用宝数据中台
- 阿里，新零售供应链
- 美团，到店事业部

目前为止，除了阿里还在等交叉，其他均已收到 offer，硬核技术实力。小民说自己是个技术渣，这是个要求很高（有点装 X）的同学。

本篇文章与大家分享下面试经历，希望能对参与一线互联网大厂面试的同学有些帮助。

腾讯 PCG 应用宝数据中台

PCG 感觉疯狂招人，面试难度忽高忽低，面试形式也很多样，腾讯视频zoom牛客网都有，比较随意。

腾讯三面 20min 算法题+一个基础问题面试就结束了……一度以为自己直接挂了，体验很差。

准备的话操作系统计算机网络多看看。如果会 C++ 多准备下，面试腾讯问题不大。

腾讯一面

- 项目讲了20分钟，问你做了什么，项目细节、你的成长之类的
- Java锁 synchronized volatile
- 快排如何变为稳定的
- 排行榜如何实现
- 如何查排行榜第2000个人的信息
- java游戏服务器 如何通信
- 算法coding，反转链表

腾讯二面

- 工厂模式 场景
- hashmap treemap
- 数据库索引 不用二叉树原因是因为 二叉树可能会变为O(n)
- 数据库如何加快 查询：cache 索引，分表分库，
- 网络粘包
- 网络两次握手不可以？
- Kafka水位(high watermark)
- ArrayList 和 LinkedList 遍历操作效率比较？ArrayList更快 操作系统预读

腾讯三面

- 算法：带父节点的二叉树 中序遍历查找的上一个节点？
zookeeper如何体现AP

抖音，IES互娱

字节面试效率是最高的，一下午一面二面连着面试，之后隔两天就三面。每次面试HR都可以当天晚上给出面试结果，效率极高，给HR小姐姐送爱心~

面试准备，基础点到为止，只要不答得很差就没事。主要还是算法，基本上每轮两个算法题，白板写代码，要运行的那种，还是挺有压力。

抖音后台一面（50min）

线程池 堵塞队列为什么要用堵塞的

一个线程在内存中如何存储

volicate ->内存重排序到底怎么避免的.....

http 握手的 wait time

访问一个 url 发生了什么

dns 是什么 原理

跳表 和平衡树区别

平衡树的种类

计算题：扑克牌两张王的概率

手写代码：合并N个链表 -> 优化为 $\log(n)$ -> null 判断 -> 不允许修改数据结构怎么实现

抖音后台二面（60min）

微服务注册中心原理

注册服务怎么判断上线下线

如果一个服务版本升级了 其他服务没升级怎么办

为什么要使用spring cloud

jvm参数 为什么要配置-> 8G内存的机器 java进程- 最大配置多少

策略模式 如何解藕 -> 项目中如何使用的

redis 持久化

redis 主从复制

算法题一

判断一个IP是否在国内。

输入：数据库中有几十万的国内IP段 (start_ip, end_ip)一个待验证的IP

输出：YES or NO

算法题二

用户在线波峰计算。

输入： 用户日志(time, user_id, login | logout)

输出：同时在线人数的峰值， 峰段(峰值的90%) eg (19:50到22:10, 峰值3亿, 最低2.7亿)

抖音后台三面

没啥技术问题，领导约谈人生，职业规划、打算之类的。

阿里，新零售供应链

阿里面试难度最高了，基础每个知识点会问到你直到答不出来为止。时间也是最长的，基本上每次面试都 50min+，面试官问的问题很细节，发现你有含糊其辞的会详细问，基础一定要复习好，还要多看看源码。

面试流程还很长，一周一面，整个流程下来很累，笔者至今还在等交叉面。

阿里一面

- mysql B+ B区别
- mysql 隔离级别 -> MVCC如何保证的 -> 间隙锁怎么使用的
- mysql hash索引?
- redis 为什么快-> 系统设计的时候如何优化的
- jenkins 如何用的
- 策略模式 不同策略怎么转化的
- Spring AOP如何实现的 -> 你项目中如何捕获aop异常 以及记录日志的
- java 枚举类型是否可以继承 (final)? 注解是否可以继承?
- java内存结构
- 对象创建过程
- 类加载器 -> 双亲委派-> ClassLoad find load的区别(和面向对象有关系)-> JDBC 加载机制 -> 面向对象的原则
- 滑动窗口 -> 客户端和服务端分别有哪些区域(已确认 传输未确认 未传输)
- volatile 怎么搞

阿里二面

实习的工作? 有什么感觉有难度的地方? 和团队其他人怎么协调的?

实习之后有哪些成长?

业务可以改进的点?

有没有博客? 开源项目?

未来三面职业规划?

看了看我大三的面试记录? 问了一下当时面试挂掉的原因? 我说算法当时不行, 顺便问了下我今年的笔试情况

常用的语言? python和java比较

最近在看什么书?

技术: HTTPS 和HTTP区别是什么? HTTPS 客户端服务器怎么交互的?

阿里三面

- 项目: 介绍 难点 实现细节 和二面差不多
- 项目的平行对比.....我们组的项目和平行项目 (hadoop spark) 相比 优势?
- eureka源码 Hystrix源码
- 分布式锁实现方式?
- zk的原理 源码
- Spring cloud/jdk设计模式 项目中的设计模式
- 序列化方式
- 最近在看什么书? 平时怎么学习的

美团-到店事业部

美团面试难度一般，基本上在问一些基础知识，多准备基础知识就可以。

美团一面

- MVCC 在读方面有什么用途
- Future的缺陷 CompletionService 在依赖任务之间是如何实现的
- Tomcat框架的 selevelt
- 算法稳定性的实际作用
- http 协议是什么 POST请求字段
- 算法：第 K大的元素
- 略去一些基础的问题，比较简单...

美团二面

算法1：第一个从n个数字的数组中等概率的取出m个数字

算法2：后缀树找最长重复的字符串

算法3：反转最后K个节点

http 请求 api 超时如何实现的(定时器有关)

mysql 索引 orderby 之后的字段要不要加进去 -> 以及 mysql orderby如何实现

操作系统两个进程写共享内存中一个位置 会不会出现不一致（和分页分段有关）

略去一些基础的问题，比较简单...

十万字面经

目录

- 基础篇
- JVM篇
- 多线程&并发篇
- Spring篇
- MyBatis篇
- SpringBoot篇
- MySQL篇
- SpringCloud篇
- Dubbo篇
- Nginx篇
- MQ篇
- 数据结构与算法篇
- Linux篇
- Zookeeper篇
- Redis篇
- 分布式篇
- 网络篇
- 设计模式
- maven篇
- ElasticSearch篇
- tomcat篇
- Git篇
- 软实力篇

笔记内容

基础篇

3、与平台无关性 (JVM是Java跨平台使用的根本)

4、可靠安全

5、支持多线程

2、面向对象和面向过程的差别

面向过程：是分析解决问题的步骤，然后用函数把这些步骤一步一步地实现，然后在使用的时候一调用则可。性能较高，所以单片机、嵌入式开发等一般采用面向过程开发

面向对象：是把构成问题的事分解成各个对象，而建立对象的目的也不是为了完成一个步骤，而是为了描述某个事物在解决问题的过程中所发生的行为。面向对象有**封装、继承、多态**的特性，所以易维护、易复用、易扩展。可以设计出低耦合的系统。但是性能上来说，比面向过程要低。

3、八种基本数据类型的大小，以及他们的封装类

基本类型	大小 (字节)	默认值	封装类
byte	1	(byte)0	Byte
short	2	(short)0	Short
int	4	0	Integer
long	8	0L	Long
float	4	0.0f	Float
double	8	0.0d	Double
boolean	-	false	Boolean
char	2	�0000(null)	Character

注：

1.int是基本数据类型，Integer是int的封装类，是引用类型。int默认值是0，而Integer默认值是null，所以Integer能区分出0和null的情况。一旦java看到null，就知道这个引用还没有指向某个对象，再任何引用使用前，必须为其指定一个对象，否则会报错。

2.基本数据类型在声明时系统会自动给它分配空间，而引用类型声明时只是分配了引用空间，必须通过实例化开辟数据空间之后才可以赋值。数组对象也是一个引用对象，将一个数组赋值给另一个数组时只是复制了一个引用，所以通过某一个数组所做的修改在另一个数组中也看的见。

虽然定义了boolean这种数据类型，但是只对它提供了非常有限的支持。在Java虚拟机中没有任何boolean值专用的字节码指令。Java语言表达式所操作的boolean值，在编译之后都使用Java虚拟机中的int数据类型来代替，而boolean数据将会被编码成Java虚拟机的byte数组，每个元素boolean元素占8位。这样我们可以得出boolean类型占了单独使用是4个字节，在数组中又是1个字节。使用int的原因是，对于当下32位的处理器（CPU）来说，一次处理数据是32位（这里不是指的是32/64位系统，而是指CPU硬件层面），具有高效存取的特点。

4、标识符的命名规则。

标识符的含义：是指在程序中，我们自己定义的内容，譬如，类的名字，方法名称以及变量名称等等，都是标识符。

命名规则：（**硬性要求**）标识符可以包含英文字母，0-9的数字，\$以及_。标识符不能以数字开头。标识符不是关键字

命名规范：（**非硬性要求**）类名规范：首字母大写，后面每个单词首字母大写（大驼峰式）。变量名规范：首字母小写，后面每个单词首字母大写（小驼峰式）。方法名规范：同变量名。

5、instanceof 关键字的作用

instanceof 严格来说是Java中的一个双目运算符，用来测试一个对象是否为一个类的实例，用法为：

```
boolean result = obj instanceof Class
```

其中obj为一个对象，Class表示一个类或者一个接口，当obj为Class的对象，或者是其直接或间接子类，或者是其接口的实现类，结果result都返回true，否则返回false。

注意：编译器会检查obj是否能转换成右边的Class类型，如果不能转换则直接报错，如果不能确定类型，则通过编译，具体看运行时定。

```
int i = 0;
System.out.println(i instanceof Integer); //编译不通过 i必须是引用类型，不能是基本类型
System.out.println(i instanceof Object); //编译不通过

Integer integer = new Integer(1);
System.out.println(integer instanceof Integer); //true

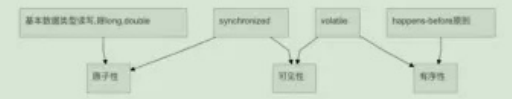
//false ,在Java5规范中对 instanceof 运算符的规定就是：如果 obj 为 null，那么将返回 false。
System.out.println(null instanceof Object);
```

JVM篇

10万字面试题总结.pdf

- 面试题
- JVM篇
 - 1. 知识总汇
 - 2. 知识总汇
 - 3. 知识总汇
 - 4. 概述一下JVM的内存模型
 - 线程私有区
 - 线程共享区
 - 5. 说堆和栈的区别
 - 6. 什么时候会触发FullGC
 - 7. 什么是Java虚拟机?为什么Java虚拟机
 - 8. Java内存结构
 - 9. 说对象分配规则
 - 10. 概述一下JVMtoClass文件的原理
 - 11. 说Java对象创建过程
 - 12. 知道类的生命周期吗?
 - 13. 概述Java的对象结构
 - 14. 如何判断对象可以被回收?
 - 15. JVM的永久代中会发生垃圾回收么
 - 16. 你知道哪些垃圾回收算法
 - 17. 请列举命令有哪些?
 - 18. 常见调优工具有哪些?
 - 19. Minor GC和Full GC分别在什么时候
 - 20. 你知道哪些JVM性能调优参数?
 - 1. 对像一定分配在堆中吗?有没有了
 - 2. 虚拟机为什么使用元空间替换了永
 - 3. 什么是Stop The World? 什么是
 - 4. 讲一下JVM 的主要组成部分及其作
 - 5. 什么是程序计数器?
 - 6. 什么是元空间?
 - 7. 对像存储在哪些内存空间?
 - 8. 你知道哪些JVM调优参数?
 - 9. 讲一下 JVM 需要哪些垃圾回收器?
 - 10. 如何理解垃圾回收器?
 - 11. 什么是垃圾回收?
 - 12. 什么是 tomcat 类加载机制?
- 多线程&并发篇
 - Spring篇
 - Mybatis篇
 - SpringBoot篇
 - MySQL篇
 - SpringCloud篇
 - Dubbo篇
 - Nginx篇

JMM是定义程序中变量的访问规则,线程对于变量的操作只能在自己的工作内存中进行,而不能直接对主内存操作,由于指令重排序,读写顺序会被打乱,因此JMM需要提供原子性,可见性,有序性保证。



3、说说类加载与卸载

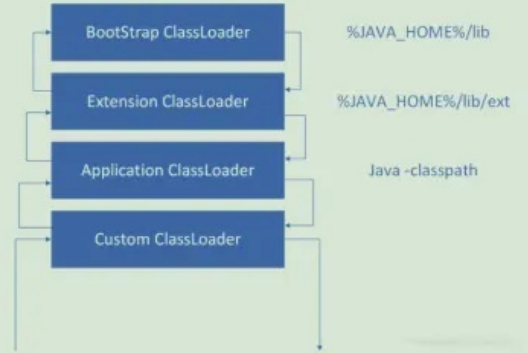
加载过程



其中验证,准备,解析合格链接

加载通过类的完全限定名,查找此类字节码文件,利用字节码文件创建Class对象。
验证确保Class文件符合当前虚拟机的要求,不会危害到虚拟机自身安全。
准备进行内存分配,为static修饰的类变量分配内存,并设置初始值(0或null),不包含final修饰的静态变量,因为final变量在编译时分配。
解析将常量池中的符号引用替换为直接引用的过程,直接引用为直接指向目标的指针或者相对偏移量等。
初始化主要完成静态块执行以及静态变量的赋值,先初始化父类,再初始化当前类,只有对类主动使用时才会初始化。
 触发条件包括:创建类的实例时,访问类的静态方法或静态变量的时候,使用Class.forName反射类的时候,或者某个子类类初始化的时候。
 Java自带的加载器加载的类,在虚拟机的生命周期中是不会被卸载的,只有用户自定义的加载器加载的类才可以卸载。

1. 加载机制-双亲委派模式



双亲委派模式,即加载器加载类时先请求委托给自己的父类加载器执行,直到顶层的启动类加载器,父类加载器能够完成加载则成功返回,不能则子类加载器才自己尝试加载。

优点:

- 避免类的重复加载
 - 避免Java的核心API被篡改
- #### 2. 分代回收
- 分代回收基于两个事实:大部分对象很快就不使用了,还有一部分不会立即无用,但也不会持续很长时间。

堆年代	老年代	新生代	永久代
年轻代	Olden	Survivor01	Survivor02
老年代	Tenured	Tenured	Tenured
永久代	PermGen/MetaSpace	PermGen/MetaSpace	PermGen/MetaSpace

年轻代->标记-复制 老年代->标记-清除

多线程&并发篇

10万字面试题总结.pdf

- 多线程&并发篇
 - 1. 说Java中类面多线程有几种方法
 - 2. 如何停止一个正在运行的线程
 - 3. notify()和notifyAll()有什么区别?
 - 4. sleep()和wait()有什么区别?
 - 5. volatile 是什么?可以保证有序性吗?
 - 6. Thread类中的start()和run()方法
 - 7. 为什么wait, notify 和 notifyAll只能
 - 8. 为什么wait和notify方法要在同步块
 - 9. Java中interrupted和isInterrupte
 - 10. Java中synchronized和Reentra
 - 11. 有两个线程T1,T2,T3,如何保证顺序
 - 12. SynchronizedMap和Concurrent
 - 13. 什么是线程安全
 - 14. Thread类中的yield方法有什么作
 - 15. Java线程池中的submit()和 execut
 - 16. 说一说自己对于synchronized关
 - 17. 说一说自己是怎么使用synchroniz
 - 18. 什么是线程安全?Vector是一个线
 - 19. volatile关键字的作用?
 - 20. 常用的线程池有哪些?
 - 21. 概述一下你的线程池的理解
 - 22. Java线程是如何执行的?
 - 23. 你对线程池了解吗?
 - 24. 你知道线程池的构造?
 - 25. 产生死锁的四个必要条件?
 - 26. 如何避免死锁?
 - 27. 线程池核心线程数怎么设置呢?
 - 28. Java线程池中有哪些数据类型?
 - 29. 线程安全需要保证几个基本特征?
 - 30. 说一下线程之间是如何通信的?
 - 31. CAS的原理呢?
 - 32. CAS有什么缺点吗?
 - 33. 引用类型有哪些?有什么区别?
 - 34. 说下ThreadLocal原理?
 - 35. 说下线程本地变量?以及核心参数
 - 36. 线程池的拒绝策略有哪些?
 - 37. 说你对JMM内存模型的理解?为
 - 38. 多线程有什么用?
 - 39. 说下CyclicBarrier和CountDownL
 - 40. 什么是AQS?
 - 41. 了解Semaphore吗?
 - 42. 什么是Callable和Future?
 - 43. 什么是阻塞队列?阻塞队列的实现
 - 44. 什么是多线程中的上下文切换?

优点: 编写简单, 如果需要访问当前线程, 则无需使用 Thread.currentThread() 方法, 直接使用 this 即可获得当前线程。
缺点: 因为线程类已经继承了 Thread 类, Java 语言是单继承的, 所以就不能再继承其他父类了。

2、如何停止一个正在运行的线程

- 使用退出标志, 使线程正常退出, 也就是当run方法完成后线程终止。
- 使用stop方法强行终止, 但是不推荐这个方法, 因为stop和suspend及resume一样都是过期的作废的方法。
- 使用interrupt方法中断线程。

```

class MyThread extends Thread {
    volatile boolean stop = false;

    public void run() {
        while (!stop) {
            System.out.println(getName() + " is running");
            try {
                sleep(1000);
            } catch (InterruptedException e) {
                System.out.println("wake up from block...");
                stop = true; // 在异常处理代码中修改共享变量的状态
            }
            System.out.println(getName() + " is exiting...");
        }
    }
}

class InterruptThreadDemo3 {
    public static void main(String[] args) throws InterruptedException {
        MyThread m1 = new MyThread();
        System.out.println("Starting thread...");
        m1.start();
        Thread.sleep(3000);
        System.out.println("Interrupt thread... " + m1.getName());
        m1.stop = true; // 设置共享变量为true
        m1.interrupt(); // 阻塞时退出阻塞状态
        Thread.sleep(3000); // 主线程休眠3秒以便观察线程m1的中断情况
        System.out.println("Stopping application...");
    }
}

```

3、notify()和notifyAll()有什么区别?

notify可能会导致死锁, 而notifyAll则不会
 任何时候只有一个线程可以获得锁, 也就是说只有一个线程可以运行synchronized 中的代码
 使用notifyAll可以唤醒 所有处于wait状态的线程, 使其重新进入就绪的等待队列中, 而notify只能唤醒一个。
 wait() 应配合while循环使用, 不应使用if, 务必在wait()调用前后都检查条件, 如果不满足, 必须调用notify()唤醒另外的线程来处理, 自己继续wait()直至条件满足再往下执行。
 notify() 是对notifyAll()的一个优化, 但它有很精确的应用场景, 并且要求正确使用。不然可能导致死锁。正确的场景应该是 WaitSet中等待的是相同的条件, 唤醒任何一个都能正确处理接下来的事项, 如果唤醒的线程无法正确处理, 务必确保继续notify()下一个线程, 并且自身需要重新回到WaitSet中。

4、sleep()和wait() 有什么区别?

对于sleep()方法, 我们首先要知道该方法是属于Thread类中的, 而wait()方法, 则是属于Object类中的。
 sleep()方法导致了程序暂停执行指定的时间, 让出cpu给其他线程, 但是他的监控状态依然保持者, 当指定的时间到了又会自动恢复运行状态。在调用sleep()方法的过程中, 线程不会释放对象锁。
 当调用wait()方法的时候, 线程会放弃对象锁, 进入等待对象的等待锁定池, 只有针对此对象调用notify()方法后本线程才进入对象锁定池准备, 获取对象锁进入运行状态。

5、volatile 是什么?可以保证有序性吗?

一旦一个共享变量 (类的成员变量、类的静态成员变量) 被volatile修饰之后, 那么就具备了两层语义:
 1) 保证了不同线程对这个变量进行操作时的可见性, 即一个线程修改了某个变量的值, 这新值对其他线程来说是立即可见的。volatile关键字会强制将修改的值立即写入主存。
 2) 禁止进行指令重排序。
 volatile 不是原子性操作
 什么叫保证部分有序性?
 当程序执行到volatile变量的读操作或者写操作时, 在其前面的操作的更改肯定全部都已经进行, 且结果已经对后面的操作可用, 在其后面的操作的更改还没有进行。

Spring篇

- 10万字源码总结.pdf
- 目录
- 基础篇
 - 1. 什么是Spring?
 - 2. 你们项目中为什么使用Spring框架?
 - 3. Autowired和Resource关键字的区别?
 - 4. 依赖注入的方式有几种, 各是什么?
 - 5. 讲一下什么是Spring
 - 6. 说说你对Spring MVC的理解
 - 7. Spring MVC常用的注解有哪些?
 - 8. 说说你对Spring的AOP理解
 - 9. Spring AOP和AspectJ AOP有什么在Spring AOP中, 关注点和切入点关注什么, 是通知? 有哪些通知类型?
 - 10. 说说你对Spring的IOC是怎么理解的
 - 11. 解释一下Spring bean的生命周期
 - 12. 解释Spring支持的几种bean的作用
 - 13. Spring基于xml注入bean的几种方式
 - 14. Spring基于xml注入bean的几种方式
 - 15. 说说Spring中ApplicationContext
 - 16. Spring框架中的事务 Bean 是怎么实现的?
 - 17. Spring 是怎么解决循环依赖的?
 - 18. 事务事务的隔离级别
 - 19. 事务事务的传播属性
 - 20. Spring 事务类型方式
 - 21. Spring 框架的事务管理有哪些类型?
 - 22. 事务三要素是什么?
 - 23. 事务注解的本质是什么?
- MyBatis篇
 - 1. 什么是MyBatis
 - 2. #{}和\${}的区别是什么?
 - 3. #{}和\${}的区别是什么?
 - 4. 高亮类中的属性名和表中的字段名不一致, MyBatis是如何进行分页的? 分页插件
 - 5. MyBatis是如何将sql执行结果封装为Java对象的?
 - 6. MyBatis是如何将sql执行结果封装为Java对象的?
 - 7. 如何执行批量插入?
 - 8. Xml映射文件中, 除了常见的select, insert, update, delete, 还有哪些? 都有哪些? 都有哪些? 都有哪些?
 - 9. MyBatis是如何将sql执行结果封装为Java对象的?
 - 10. MyBatis是如何将sql执行结果封装为Java对象的?
 - 11. 说说MyBatis的缓存机制
 - 12. JDBC 编程有哪些步骤?
 - 13. MyBatis 中见过什么设计模式?
 - 14. MyBatis 中见过 UserMapper.java
- SpringBoot篇
 - 1. 什么是SpringBoot
 - 2. 你们项目中为什么使用SpringBoot
 - 3. SpringBoot的启动原理
 - 4. SpringBoot的启动原理
 - 5. SpringBoot的启动原理
 - 6. SpringBoot的启动原理
 - 7. SpringBoot的启动原理
 - 8. SpringBoot的启动原理
 - 9. SpringBoot的启动原理
 - 10. SpringBoot的启动原理
 - 11. SpringBoot的启动原理
 - 12. SpringBoot的启动原理
 - 13. SpringBoot的启动原理
 - 14. SpringBoot的启动原理
 - 15. SpringBoot的启动原理
 - 16. SpringBoot的启动原理
 - 17. SpringBoot的启动原理
 - 18. SpringBoot的启动原理
 - 19. SpringBoot的启动原理
 - 20. SpringBoot的启动原理
 - 21. SpringBoot的启动原理
 - 22. SpringBoot的启动原理
 - 23. SpringBoot的启动原理
 - 24. SpringBoot的启动原理
 - 25. SpringBoot的启动原理
 - 26. SpringBoot的启动原理
 - 27. SpringBoot的启动原理
 - 28. SpringBoot的启动原理
 - 29. SpringBoot的启动原理
 - 30. SpringBoot的启动原理
 - 31. SpringBoot的启动原理
 - 32. SpringBoot的启动原理
 - 33. SpringBoot的启动原理
 - 34. SpringBoot的启动原理
 - 35. SpringBoot的启动原理
 - 36. SpringBoot的启动原理
 - 37. SpringBoot的启动原理
 - 38. SpringBoot的启动原理
 - 39. SpringBoot的启动原理
 - 40. SpringBoot的启动原理
 - 41. SpringBoot的启动原理
 - 42. SpringBoot的启动原理
 - 43. SpringBoot的启动原理
 - 44. SpringBoot的启动原理
 - 45. SpringBoot的启动原理
 - 46. SpringBoot的启动原理
 - 47. SpringBoot的启动原理
 - 48. SpringBoot的启动原理
 - 49. SpringBoot的启动原理
 - 50. SpringBoot的启动原理
 - 51. SpringBoot的启动原理
 - 52. SpringBoot的启动原理
 - 53. SpringBoot的启动原理
 - 54. SpringBoot的启动原理
 - 55. SpringBoot的启动原理
 - 56. SpringBoot的启动原理
 - 57. SpringBoot的启动原理
 - 58. SpringBoot的启动原理
 - 59. SpringBoot的启动原理
 - 60. SpringBoot的启动原理
 - 61. SpringBoot的启动原理
 - 62. SpringBoot的启动原理
 - 63. SpringBoot的启动原理
 - 64. SpringBoot的启动原理
 - 65. SpringBoot的启动原理
 - 66. SpringBoot的启动原理
 - 67. SpringBoot的启动原理
 - 68. SpringBoot的启动原理
 - 69. SpringBoot的启动原理
 - 70. SpringBoot的启动原理
 - 71. SpringBoot的启动原理
 - 72. SpringBoot的启动原理
 - 73. SpringBoot的启动原理
 - 74. SpringBoot的启动原理
 - 75. SpringBoot的启动原理
 - 76. SpringBoot的启动原理
 - 77. SpringBoot的启动原理
 - 78. SpringBoot的启动原理
 - 79. SpringBoot的启动原理
 - 80. SpringBoot的启动原理
 - 81. SpringBoot的启动原理
 - 82. SpringBoot的启动原理
 - 83. SpringBoot的启动原理
 - 84. SpringBoot的启动原理
 - 85. SpringBoot的启动原理
 - 86. SpringBoot的启动原理
 - 87. SpringBoot的启动原理
 - 88. SpringBoot的启动原理
 - 89. SpringBoot的启动原理
 - 90. SpringBoot的启动原理
 - 91. SpringBoot的启动原理
 - 92. SpringBoot的启动原理
 - 93. SpringBoot的启动原理
 - 94. SpringBoot的启动原理
 - 95. SpringBoot的启动原理
 - 96. SpringBoot的启动原理
 - 97. SpringBoot的启动原理
 - 98. SpringBoot的启动原理
 - 99. SpringBoot的启动原理
 - 100. SpringBoot的启动原理

@Resource默认按照ByName自动注入, 由2EE提供, 需要导入包javax.annotation.Resource, @Resource有两个重要的属性: name和type, 而Spring将@Resource注解的name属性解析为bean的名字, 而type属性则解析为bean的类型。所以, 如果使用name属性, 则使用ByName的自动注入策略, 而使用type属性时则使用byType自动注入策略。如果既不制定name也不制定type属性, 这时将通过反射机制使用ByName自动注入策略。

```
public class TestServiceImpl {
    // 下面两种@Resource只要使用一种即可
    @Resource(name="userDao")
    private UserDao userDao; // 用于字段上

    @Resource(name="userDao")
    public void setUserDao(UserDao userDao) { // 用于属性的setter方法上
        this.userDao = userDao;
    }
}
```

注: 最好是将@Resource放在setter方法上, 因为这样更符合面向对象的思想, 通过set、get去操作属性, 而不是直接去操作属性。

@Resource装配顺序:

- ①如果同时指定了name和type, 则从Spring上下文中找到唯一匹配的bean进行装配, 找不到则抛出异常。
 - ②如果指定了name, 则从上下文中查找名称(id)匹配的bean进行装配, 找不到则抛出异常。
 - ③如果指定了type, 则从上下文中找到类似匹配的唯一bean进行装配, 找不到或是找到多个, 都会抛出异常。
 - ④如果既没有指定name, 又没有指定type, 则自动按照ByName方式进行装配; 如果没有匹配, 则回退为一个原始类型进行匹配, 如果匹配则自动装配。
- @Resource的作用相当于@Autowired, 只不过@Autowired按照byType自动注入。

4. 依赖注入的方式有几种, 各是什么?

一、构造器注入 将被依赖对象通过构造函数的参数注入给依赖对象, 并且在初始化对象的时候注入。

优点: 对象初始化完成后便可获得可使用的对象。

缺点: 当需要注入的对象很多时, 构造函数列表将会很长; 不够灵活。若有多种注入方式, 每种方式只注入其中几个参数, 那么就需要提供多个构造函数, 可读性

二、setter方法注入 IoC Service Provider通过调用或变量提供的setter函数将被依赖对象注入给依赖类。

优点: 灵活, 可以选择性地注入需要的对象。

缺点: 依赖对象初始化完成后由于尚未注入被依赖对象, 因此还不能使用。

三、接口注入 依赖类必须实现指定的接口, 然后实现该接口中的一个函数, 该函数就是用于依赖注入。该函数的参数就是要注入的对象。

优点: 接口注入中, 接口的名字、函数的名字都不重要, 只要保证函数的参数是要注入的对象类型即可。

缺点: 侵入性太强, 不建议使用。

PS: 什么是侵入性? 如果类A要使用别人提供一个功能, 若为了使用这功能, 需要自己的类中增加额外的代码, 这就是侵入性。

5. 讲一下什么是Spring

Spring是一个轻量级的IoC和AOP容器框架, 是为Java应用程序提供基础性服务的一套框架, 目的是用于简化企业应用程序的开发, 它使得开发者只需关心业务需求。常见的配置方式有三种: 基于XML的配置、基于注解的配置、基于Java的配置。

主要由以下几个模块组成:

- Spring Core: 核心类库, 提供IOC服务;
- Spring Context: 提供框架式的Bean访问方式, 以及企业级功能 (JNDI、定时任务等);
- Spring AOP: AOP服务;
- Spring DAO: 对JDBC的抽象, 简化了数据库访问异常的处理;
- Spring ORM: 对现有的ORM框架的支持;
- Spring Web: 提供了基本的面向Web的综合特性, 例如多文件上传;
- Spring MVC: 提供面向Web应用的Model-View-Controller实现。

6. 说说你对Spring MVC的理解

什么是MVC模式

MVC: MVC是一种设计模式

MVC的原理图:

MyBatis篇

- 10万字源码总结.pdf
- 目录
- 基础篇
 - 1. 什么是Spring?
 - 2. 你们项目中为什么使用Spring框架?
 - 3. Autowired和Resource关键字的区别?
 - 4. 依赖注入的方式有几种, 各是什么?
 - 5. 讲一下什么是Spring
 - 6. 说说你对Spring MVC的理解
 - 7. Spring MVC常用的注解有哪些?
 - 8. 说说你对Spring的AOP理解
 - 9. Spring AOP和AspectJ AOP有什么在Spring AOP中, 关注点和切入点关注什么, 是通知? 有哪些通知类型?
 - 10. 说说你对Spring的IOC是怎么理解的
 - 11. 解释一下Spring bean的生命周期
 - 12. 解释Spring支持的几种bean的作用
 - 13. Spring基于xml注入bean的几种方式
 - 14. Spring基于xml注入bean的几种方式
 - 15. 说说Spring中ApplicationContext
 - 16. Spring框架中的事务 Bean 是怎么实现的?
 - 17. Spring 是怎么解决循环依赖的?
 - 18. 事务事务的隔离级别
 - 19. 事务事务的传播属性
 - 20. Spring 事务类型方式
 - 21. Spring 框架的事务管理有哪些类型?
 - 22. 事务三要素是什么?
 - 23. 事务注解的本质是什么?
- MyBatis篇
 - 1. 什么是MyBatis
 - 2. #{}和\${}的区别是什么?
 - 3. #{}和\${}的区别是什么?
 - 4. 高亮类中的属性名和表中的字段名不一致, MyBatis是如何进行分页的? 分页插件
 - 5. MyBatis是如何将sql执行结果封装为Java对象的?
 - 6. MyBatis是如何将sql执行结果封装为Java对象的?
 - 7. 如何执行批量插入?
 - 8. Xml映射文件中, 除了常见的select, insert, update, delete, 还有哪些? 都有哪些? 都有哪些? 都有哪些?
 - 9. MyBatis是如何将sql执行结果封装为Java对象的?
 - 10. MyBatis是如何将sql执行结果封装为Java对象的?
 - 11. 说说MyBatis的缓存机制
 - 12. JDBC 编程有哪些步骤?
 - 13. MyBatis 中见过什么设计模式?
 - 14. MyBatis 中见过 UserMapper.java
- SpringBoot篇
 - 1. 什么是SpringBoot
 - 2. 你们项目中为什么使用SpringBoot
 - 3. SpringBoot的启动原理
 - 4. SpringBoot的启动原理
 - 5. SpringBoot的启动原理
 - 6. SpringBoot的启动原理
 - 7. SpringBoot的启动原理
 - 8. SpringBoot的启动原理
 - 9. SpringBoot的启动原理
 - 10. SpringBoot的启动原理
 - 11. SpringBoot的启动原理
 - 12. SpringBoot的启动原理
 - 13. SpringBoot的启动原理
 - 14. SpringBoot的启动原理
 - 15. SpringBoot的启动原理
 - 16. SpringBoot的启动原理
 - 17. SpringBoot的启动原理
 - 18. SpringBoot的启动原理
 - 19. SpringBoot的启动原理
 - 20. SpringBoot的启动原理
 - 21. SpringBoot的启动原理
 - 22. SpringBoot的启动原理
 - 23. SpringBoot的启动原理
 - 24. SpringBoot的启动原理
 - 25. SpringBoot的启动原理
 - 26. SpringBoot的启动原理
 - 27. SpringBoot的启动原理
 - 28. SpringBoot的启动原理
 - 29. SpringBoot的启动原理
 - 30. SpringBoot的启动原理
 - 31. SpringBoot的启动原理
 - 32. SpringBoot的启动原理
 - 33. SpringBoot的启动原理
 - 34. SpringBoot的启动原理
 - 35. SpringBoot的启动原理
 - 36. SpringBoot的启动原理
 - 37. SpringBoot的启动原理
 - 38. SpringBoot的启动原理
 - 39. SpringBoot的启动原理
 - 40. SpringBoot的启动原理
 - 41. SpringBoot的启动原理
 - 42. SpringBoot的启动原理
 - 43. SpringBoot的启动原理
 - 44. SpringBoot的启动原理
 - 45. SpringBoot的启动原理
 - 46. SpringBoot的启动原理
 - 47. SpringBoot的启动原理
 - 48. SpringBoot的启动原理
 - 49. SpringBoot的启动原理
 - 50. SpringBoot的启动原理
 - 51. SpringBoot的启动原理
 - 52. SpringBoot的启动原理
 - 53. SpringBoot的启动原理
 - 54. SpringBoot的启动原理
 - 55. SpringBoot的启动原理
 - 56. SpringBoot的启动原理
 - 57. SpringBoot的启动原理
 - 58. SpringBoot的启动原理
 - 59. SpringBoot的启动原理
 - 60. SpringBoot的启动原理
 - 61. SpringBoot的启动原理
 - 62. SpringBoot的启动原理
 - 63. SpringBoot的启动原理
 - 64. SpringBoot的启动原理
 - 65. SpringBoot的启动原理
 - 66. SpringBoot的启动原理
 - 67. SpringBoot的启动原理
 - 68. SpringBoot的启动原理
 - 69. SpringBoot的启动原理
 - 70. SpringBoot的启动原理
 - 71. SpringBoot的启动原理
 - 72. SpringBoot的启动原理
 - 73. SpringBoot的启动原理
 - 74. SpringBoot的启动原理
 - 75. SpringBoot的启动原理
 - 76. SpringBoot的启动原理
 - 77. SpringBoot的启动原理
 - 78. SpringBoot的启动原理
 - 79. SpringBoot的启动原理
 - 80. SpringBoot的启动原理
 - 81. SpringBoot的启动原理
 - 82. SpringBoot的启动原理
 - 83. SpringBoot的启动原理
 - 84. SpringBoot的启动原理
 - 85. SpringBoot的启动原理
 - 86. SpringBoot的启动原理
 - 87. SpringBoot的启动原理
 - 88. SpringBoot的启动原理
 - 89. SpringBoot的启动原理
 - 90. SpringBoot的启动原理
 - 91. SpringBoot的启动原理
 - 92. SpringBoot的启动原理
 - 93. SpringBoot的启动原理
 - 94. SpringBoot的启动原理
 - 95. SpringBoot的启动原理
 - 96. SpringBoot的启动原理
 - 97. SpringBoot的启动原理
 - 98. SpringBoot的启动原理
 - 99. SpringBoot的启动原理
 - 100. SpringBoot的启动原理

- (2) MyBatis 可以使用 XML 来注解来配置和映射原生信息, 将 POJO 映射成数据库中的记录, 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。
- (3) 通过xml 文件或注解的方式将要执行的各种 statement 配置起来, 并通过Java对象和statement中sql的动态参数进行映射生成最终执行的sql语句, 最后由mybatis框架执行sql并将结果映射为Java对象并返回。(从执行sql到返回result的过程)。

2. 说说MyBatis的优点和缺点

优点:

- (1) 基于SQL语句编程, 相当灵活, 不会对应用程序或者数据库的现有设计造成任何影响, SQL写在XML里, 解除SQL与程序代码的耦合, 便于统一管理; 提供XML标签, 支持编写动态SQL语句, 并可重用。
- (2) 与JDBC相比, 减少了50%以上的代码量, 消除了JDBC大量冗余的代码, 不需要手动开关连接;
- (3) 很好的与各种数据库兼容 (因为MyBatis使用JDBC来连接数据库, 所以只要JDBC支持的数据库MyBatis都支持)。
- (4) 能够与Spring很好的集成;
- (5) 提供映射标签, 支持对象与数据库的ORM字段关系映射; 提供对象关系映射标签, 支持对象关系组件维护。

缺点:

- (1) SQL语句的编写工作量较大, 尤其当字段多、关联表多时, 对开发人员编写SQL语句的功底有一定要求。
- (2) SQL语句依赖于数据库, 导致数据库移植性差, 不能随意更换数据库。

3. #{}和\${}的区别是什么?

#{}是预编译处理, \${}是字符串替换。

Mybatis在处理#{}时, 会将sql中的#{}替换为?, 调用PreparedStatement的set方法来赋值;

Mybatis在处理\${}时, 就是把\${}替换成变量的值。

使用#{}可以有效的防止SQL注入, 提高系统安全性。

4. 当实体类中的属性名和表中的字段名不一样, 怎么办?

```
<select id="selectOrder" parameterType="int" resultType="me.gacl.domain.order">
    select order_id id, order_no orderno, order_price price from orders where
    order_id=#{id};
</select>
```

第2种: 通过来映射字段名和实体类属性名的一一对应的关系。

```
<select id="getOrder" parameterType="int" resultMap="orderResultMap">
    select * from orders where order_id=#{id}
</select>

<resultMap type="me.gacl.domain.order" id="orderResultMap">
    <!--用id属性来映射主键字段-->
    <id property="id" column="order_id">
    </id>
    <!--用result属性来映射非主键字段, property为实体类属性名, column为数据库中的属性-->
    <result property="orderno" column="order_no"/>
    <result property="price" column="order_price" />
</resultMap>
```

5. Mybatis是如何进行分页的? 分页插件的原理是什么?

Mybatis使用RowBounds对象进行分页, 它是针对ResultSet结果集执行的内存分页, 而非物理分页。可以在sql内直接拼带有物理分页的参数来完成物理分页功能, 也可以使用分页插件来完成逻辑分页, 比如: MySQL数据库的时候, 在原有SQL后面拼with limit。

分页插件的基本原理是使用Mybatis提供的插件接口, 实现自定义插件, 在插件的拦截方法内拦截待执行的sql, 然后重写sql, 根据dialect方言, 添加对应的物理分页语句和物理分页参数。

6. Mybatis是如何将sql执行结果封装为目标对象并返回的? 都有哪些映射形式?

第一种是使用标签, 逐一定义数据库列名和对象属性名之间的映射关系。

第二种是使用sql列的别名功能, 将列的别名书写为对象属性名。

有了列名与属性名的映射关系后, Mybatis通过反射创建对象, 同时使用反射给对象的属性逐一赋值并返回, 那些找不到映射关系的属性, 是无法完成赋值的。

7. 如何执行批量插入?

首先, 创建一个简单的insert语句:

SpringBoot篇

10万字面试题总结.pdf

- 目录
- 面试
- 面试
- 基础篇
- JVM篇
- 多线程与并发篇
- Spring篇
- MyBatis篇
- SpringBoot篇
 - 1. 为什么要用SpringBoot
 - 2. Spring Boot 的核心注解都有哪些？在Spring Boot 中有哪几种方式？
 - 3. 运行Spring Boot有哪几种方式？
 - 4. 如何理解 Spring Boot 中的 Starters
 - 5. 如何理解 Spring Boot 的自动配置原理？
 - 6. Spring Boot 需要独立的容器运行吗？
 - 7. Spring Boot 中的监视器是什么？
 - 8. 如何理解 Spring Boot 的异常处理？
 - 9. 你知道 Spring Boot 中的 Starter
 - 10. springboot 常用的 starter 有哪些？
 - 11. Spring Boot 实现外部部署有哪几种方式？
 - 12. 如何理解 Spring Boot 配置加载顺序？
 - 13. Spring Boot 的核心配置文件有哪些？
 - 14. 如何集成 Spring Boot 和 ActiveMQ
- MySQL篇
- SpringCloud篇
- Dubbo篇
- Nginx篇
- MQ篇
 - 一、数据结构与算法
- Linux篇
- Zookeeper篇
- Redis篇
- 分布式篇
- 网络篇
- 设计模式
- maven篇
- ElasticSearch篇
- tomcat篇
- Git篇
- 软考篇

Spring Boot官方的启动器都是以spring-boot-starter-命名的，代表了一个特定的应用类型。第三方的启动器不能以spring-boot开头命名，它们都被Spring Boot官方保留。一般一个第三方的应该这样命名，像mybatis的mybatis-spring-boot-starter。

Starters分类：

1. Spring Boot应用类启动器

启动器名称	功能描述
spring-boot-starter	包含自动配置、日志、YAML的支持。
spring-boot-starter-web	使用Spring MVC构建web工程，包含restful，默认使用Tomcat容器。
...	...

1. Spring Boot生产启动器

启动器名称	功能描述
spring-boot-starter-actuator	提供生产环境特性，能监控管理应用。

1. Spring Boot技术类启动器

启动器名称	功能描述
spring-boot-starter-json	提供对JSON的读写支持。
spring-boot-starter-logging	默认的日志启动器，默认使用Logback。
...	...

1. 其他第三方启动器

5、如何在Spring Boot启动的时候运行一些特定的代码？

如果你想在Spring Boot启动的时候运行一些特定的代码，你可以实现接口ApplicationRunner或者CommandLineRunner，这两个接口实现方式一样，它们都只提供了一个run方法。

CommandLineRunner：启动获取命令行参数

6、Spring Boot 需要独立的容器运行吗？

可以不需要，内置了 Tomcat/ Jetty 等容器。

7、Spring Boot中的监视器是什么？

Spring boot actuator是Spring启动框架中的重要功能之一。Spring boot监视器可帮助你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报警消息。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。

8、如何使用Spring Boot实现异常处理？

Spring提供了一种使用ControllerAdvice处理异常的非常有用的方法。我们通过实现一个ControllerAdvice类，来处理控制器类抛出的所有异常。

9、你如何理解 Spring Boot 中的 Starters？

Starters可以理解成启动器，它包含了一系列可以集成到应用层面的依赖包，你可以一站式集成Spring及其他技术，而不需要到处找示例代码和依赖包。如果你想使用Spring JPA访问数据库，只要加入spring-boot-starter-data-jpa启动器依赖就能使用了。

10、springboot常用的starter有哪些

spring-boot-starter-web 嵌入tomcat和web开发需要servlet与jsp支持

spring-boot-starter-data-jpa 数据库支持

spring-boot-starter-data-redis redis数据库支持

spring-boot-starter-data-solr solr支持

mybatis-spring-boot-starter 第三方的mybatis集成starter

11、SpringBoot 实现热部署有哪几种方式？

主要有两种方式：

- Spring Loaded
- Spring-boot-devtools

12、如何理解 Spring Boot 配置加载顺序？

在Spring Boot 里面，可以使用以下几种方式来加载配置。

1) properties文件；

MySQL篇

10万字面试题总结.pdf

- 目录
- 面试
- 面试
- SpringBoot篇
- MySQL篇
 - 1. 数据库的三范式是什么
 - 2. MySQL索引引擎有哪些
 - 3. 索引的B+树与MyISAM区别
 - 4. 数据库的事务
 - 5. 索引是什么
 - 6. SQL优化手段有哪些
 - 7. 简单说一下drop、delete与truncate
 - 8. 什么是视图
 - 9. 什么是左外连接、右外连接
 - 10. 并发事务带来哪些问题
 - 11. 事务隔离级别有哪些？MySQL的默认隔离级别是什么？
 - 12. 大表如何优化？
 - 1. 降低数据的范围
 - 2. 读/写分离
 - 3. 垂直分区
 - 4. 水平分区
 - 13. 分库分表之索引 键如何设置？
 - 14. 说说在MySQL中，索引索引SQL
 - 15. 索引有什么优缺点？
 - 16. MySQL中varchar与char的区别
 - 17. int(11)中的11代表什么意义？
 - 18. 为什么SELECT COUNT(*) FROM
 - 19. 说说InnoDB与MyISAM有什么区别
 - 20. MySQL索引类型有哪些？
 - 21. 什么情况下不能使用索引？
 - 22. 说说什么是MVCC
 - 23. MVCC可以为数据库解决什么问题
 - 24. 说说MVCC的实现原理
 - 25. MySQL事务隔离级别？
 - 26. 请说说MySQL数据库的锁？
 - 27. 说说什么是锁升级？
 - 28. 死锁现象和死锁预防
 - 29. 怎样尽量避免死锁的出现？
 - 30. 使用MySQL的索引应该注意哪些？
 - 31. CHAR和VARCHAR的区别？
 - 32. 主键和候选键有什么区别？
 - 33. 主键与索引有什么区别？
 - 34. MySQL如何做到可用方案？
- SpringCloud篇
- Dubbo篇
- Nginx篇
- MQ篇
 - 一、数据结构与算法
- Linux篇

事务的特性：

数据库事务特性：原子性(Atomic)、一致性(Consistency)、隔离性(Isolation)、持久性(Durability)，简称ACID。

- 原子性：组成一个事务的多个数据库操作是一个不可分割的原子单元，只有所有操作成功，整个事务才会提交。任何一个操作失败，已经执行的任何操作都必须撤销，让数据库返回初始状态。
- 一致性：事务操作成功后，数据库所处的状态和它的业务规则是一致的。即数据不会被破坏。如A转账100元给B，不管操作是否成功，A和B的账户总额是不变的。
- 隔离性：在并发数据操作时，不同的事务拥有各自的数据空间，它们的操作不会对彼此产生干扰
- 持久性：一旦事务提交成功，事务中的所有操作都必须持久化到数据库中。

5、索引是什么

- 官方介绍索引是帮助MySQL高效获取数据的数据结构。更通俗的说，数据库索引好比是一本书前面的目录，能加快数据库的查询速度。
- 一般来说索引本身也很大，不可能全部存储在内存中，因此索引往往是存储在磁盘上的文件中的（可能存储在单独的索引文件中，也可能和数据一起存储在数据文件中）。
- 我们通常所说的索引，包括聚集索引、覆盖索引、组合索引、前缀索引、唯一索引等，没有特别说明，默认都是使用B+树结构组织（多路搜索树，并不一定是二叉的）的索引。

6、SQL优化手段有哪些

- 查询语句中不要使用Select *
- 尽量减少子查询，使用关联查询（left join,right join,inner join）替代
- 减少使用IN或者NOT IN,使用exists, not exists或者关联查询语句替代
- or 的查询尽量用 union或者union all 替代(在确认没有重复数据或者不用删除重复数据时，union all会更好)
- 应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。
- 应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：select id from t where num is null 可以在num上设置默认值0，确保表中num列没有null值，然后这样查询：select id from t where num=0

7、简单说一说drop、delete与truncate的区别

SQL中的drop、delete、truncate都表示删除，但是三者有一些差别

delete和truncate只删除表的数据不删除表的结构 速度,一般来说: drop> truncate > delete delete语句是dml,这个操作会放到rollback segment中,事务提交之后才生效,如果有相应的trigger,执行的时候将被触发,truncate,drop是ddl,操作立即生效,原数据不放到rollback segment中,不能回滚,操作不触发trigger.

8、什么是视图

视图是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增、改、查、操作，视图通常是一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更简单，相比多表查询。

9、什么是内联接、左外联接、右外联接？

- 内联接（Inner Join）：匹配2张表中相关联的记录。
- 左外联接（Left Outer Join）：除了匹配2张表中相关联的记录外，还会匹配左表中剩余的记录，右表中未匹配到的字段用NULL表示。
- 右外联接（Right Outer Join）：除了匹配2张表中相关联的记录外，还会匹配右表中剩余的记录，左表中未匹配到的字段用NULL表示。在判定左表和右表时，要根据表名出现在Outer Join的左右位置关系。

10、并发事务带来哪些问题？

在典型的应用程序中，多个事务并行运行，经常会操作相同的数据库来完成各自的任务（多个用户对同一数据进行操作）。并发虽然是必须的，但可能会导致以下的问题。

- 脏读（Dirty read）：当一个事务正在访问数据并且对数据进行了修改，而这种修改还没有提交到数据库中，这时另外一个事务也访问了这个数据，然后使用了这个数据。因为这个数据是还没有提交的数据，那么另外一个事务读到的这个数据是“脏数据”，依据“脏数据”所做的操作可能是错误的。
- 丢失修改（Lost to modify）：指在一个事务读取一个数据时，另外一个事务也访问了该数据，那么在第一个事务中修改了这个数据后，第二个事务也修改了这个数据。这样第一个事务内的修改结果就被丢失，因此称为丢失修改。例如：事务1读取某表中的数据A=20，事务2也读取A=20，事务1修改A=A+1，事务2也修改A=A+1，最终结果A=19，事务1的修改被丢失。
- 不可重复读（Unrepeatable read）：指在一个事务内多次读同一数据。在这个事务还没有结束时，另一个事务也访问该数据，那么，在第一个事务中的两次读数据之间，由于第二个事务的修改导致第一个事务再次读取的数据可能不一样。这就发生了在一个事务内两次读到的数据是不一样的情况，因此称为不可重复读。
- 幻读（Phantom read）：幻读与不可重复读类似。它发生在一个事务（T1）读取了几行数据，接着另一个并发事务（T2）插入了一些数据时。在随后的查询中，第一个事务（T1）就会发现多了一些原本不存在的记录，就好像发生了幻觉一样，所以称为幻读。

10万字面试题.pdf

- 面试题
- 基础篇
- JVM篇
- 多线程及并发篇
- Spring篇
- MyBatis篇
- SpringBoot篇
- MySQL篇
- SpringCloud篇
- Dubbo篇
 - 1. 说说一次Dubbo服务请求流程?
 - 2. 说说Dubbo工作原理?
 - 3. Dubbo支持哪些协议?
 - 4. 注册中心挂了, consumer还能不能调用?
 - 5. 怎么实现动态感知服务下线呢?
 - 6. Dubbo负载均衡策略?
 - 7. Dubbo容错策略?
 - 8. Dubbo动态代理策略有哪些?
 - 9. 说说Dubbo与Spring Cloud的区别?
 - 10. Zookeeper和Dubbo的关系?
- Ngix篇
- MQ篇
- 数据库与算法篇
- Linux篇
- Zookeeper篇
- Redis篇
- 分布式篇
- 网络篇
- 设计模式
- maven篇
- ElasticSearch篇
- tomcat篇
- Git篇
- 软实力篇

上图角色说明:

节点	角色说明
Provider	暴露服务的提供方
Consumer	调用远程服务的消费方
Registry	服务注册与发现的注册中心
Monitor	统计服务的调用次数和调用时间的监控中心
Container	服务运行容器

2. 说说Dubbo工作原理

工作原理分10层:

- 第一层: service层, 接口层, 给服务提供者和消费者来实现的(留给开发人员来实现);
- 第二层: config层, 配置层, 主要是对Dubbo进行各种配置的, Dubbo相关配置;
- 第三层: proxy层, 服务代理层, 透明生成客户端的stub和服务器端的skeleton, 调用的是接口, 实现类没有, 所以得生成代理, 代理之间再进行网络通讯, 负载均衡等;
- 第四层: registry层, 服务注册层, 负责服务的注册与发现;

- 第六层: monitor层, 监控层, 对rpc接口的调用次数和调用时间进行监控;
- 第七层: protocol层, 远程调用层, 封装rpc调用;
- 第八层: exchange层, 信息交换层, 封装请求响应模式, 同步转异步;
- 第九层: transport层, 网络传输层, 抽象mina和netty为统一接口;
- 第十层: serialize层, 数据序列化层。

这是个很坑爹的面试题, 但是很多面试官又喜欢问, 你真的要背么? 你能背那还是不错的, 我建议不要背, 你就想想Dubbo服务调用过程中应该涉及哪些技术, 把这些技术串起来就OK了。

面试扩散

如果你设计一个RPC框架, 你会怎么做? 其实你就把上面这个工作原理中涉及到的技术点总结一下就行了。

3. Dubbo支持哪些协议?

还有三种, 混个眼熟就行: Memcached协议, Redis协议, Rest协议。

上面基本上把序列化的方式也罗列出来了。

详细请参考: [Dubbo官网: http://dubbo.apache.org/zh-cn/docs/user/references/protocol/dubbo.html](http://dubbo.apache.org/zh-cn/docs/user/references/protocol/dubbo.html)

4. 注册中心挂了, consumer还能不能调用provider?

可以。因为刚开始初始化的时候, consumer会将需要的所有提供者的地址等信息拉取到本地缓存, 所以注册中心挂了可以继续通信。但是provider挂了, 那就没法调用了。

10万字面试题.pdf

- 面试题
- 基础篇
- JVM篇
- 多线程及并发篇
- Spring篇
- MyBatis篇
- SpringBoot篇
- MySQL篇
- SpringCloud篇
- Dubbo篇
- Ngix篇
 - 1. 简述一下什么是Ngix, 它有什么优势?
 - 2. Ngix是如何处理一个HTTP请求的呢?
 - 3. 列举一些Ngix的特性
 - 4. 请列举Ngix和Apache之间的不同点
 - 5. 在Ngix中, 如何使用未定义的服务器名称来阻止处理请求?
 - 6. 请解释Ngix服务器上的Master和Worker进程
 - 7. 请解释代理中的正向代理和反向代理
 - 8. 解释Ngix进程
- MQ篇
- 数据库与算法篇
- Linux篇
- Zookeeper篇
- Redis篇
- 分布式篇
- 网络篇
- 设计模式
- maven篇
- ElasticSearch篇
- tomcat篇
- Git篇
- 软实力篇

2. Ngix是如何处理一个HTTP请求的呢?

Ngix是一个高性能的Web服务器, 能够同时处理大量的并发请求。它结合多进程机制和异步机制, 异步机制使用的是异步非阻塞方式, 接下来就给大家介绍一下Ngix的多线程机制和异步非阻塞机制。

1. 多进程机制

服务器每当收到一个客户端时, 就有服务器主进程(master process)生成一个子进程(worker process)出来和客户端建立连接进行交互, 直到连接断开, 孩子进程就结束了。

使用进程的好处是各个进程之间相互独立, 不需要加锁, 减少了使用锁可能造成的影响, 同时降低编程的复杂度, 降低开发成本。其次, 采用独立的进程, 可以让进程互相之间不会影响, 如果一个进程发生异常退出时, 其它进程正常工作, master进程则很快启动新的worker进程, 确保服务不会中断, 从而将风险降到最低。

缺点是操作系统生成一个子进程需要进行内存复制等操作, 在资源和时间上会产生一定的开销, 当有大量请求时, 会导致系统性能下降。

2. 异步非阻塞机制

每个工作进程使用异步非阻塞方式, 可以处理多个客户端请求。

当某个工作进程收到客户端的请求以后, 调用IO进行处理, 如果不能立即得到结果, 就去处理其他请求(即为非阻塞); 而客户端在此期间也无需等待响应, 可以去处理其他事情(即为异步)。

当IO返回时, 就会通知此工作进程; 该进程得到通知, 暂时挂起当前处理的事务去响应客户端请求。

3. 列举一些Ngix的特性

- Ngix服务器的特性包括:
- 反向代理/负载均衡器
- 嵌入式Perl解释器
- 动态二进制编译

4. 请列举Ngix和Apache之间的不同点

Ngix	Apache
<ul style="list-style-type: none"> Ngix是一个基于事件的web服务器 所有请求都由一个线程处理 Ngix避免子进程的概念 Ngix类似于速度 Ngix在内存消耗和连接方面比较好 Ngix在负载均衡方面表现较好 对于PHP来说, Ngix可能更可取, 因为它支持PHP Ngix不支持像IBM和OpenVMS一样的OS。 Ngix只具有核心功能 Ngix的性能和可伸缩性不依赖于硬件 	<ul style="list-style-type: none"> Apache是一个基于流程的服务器 单个线程处理单个请求 Apache是基于子进程的 Apache类似于功率 Apache在内存消耗和连接上没有提高 当流量到达进程的极限时, Apache将拒绝新的连接 Apache支持的PHP, Python, Perl和其他语言使用插件, 当应用程序基于Python或Ruby时, 它非常有用 Apache支持更多的OS Apache提供了比Ngix更多的功能 Apache依赖于CPU和内存硬件组件

5. 在Ngix中, 如何使用未定义的服务器名称来阻止处理请求?

只需将请求删除的服务器名称就可以定义为:

```
Server{
  listen 80;
  server_name "";
  return 444;
}
```

这里, 服务器名称被保留为一个空字符串, 它将在没有“主机”头字段的请求下匹配请求, 而一个特殊的Ngix的非标准代码444被返回, 从而终止连接。

一般推荐worker进程数与CPU内核数一致, 这样一来不存在大量的子进程生成和管理任务, 避免了进程之间竞争CPU资源和进程切换的开销。而且Ngix为了更好的利用多核特性, 提供了CPU亲和性的绑定选项, 我们可以将某一个进程绑定在某一个核上, 这样就不会因为进程的切换带来Cache的失效。

对于每个请求, 有且只有一个工作进程对其处理。首先, 每个worker进程都是从master进程fork过来。在master进程里面, 先建立好需要listen的socket(listenfd)之后, 然后再fork出多个worker进程。

- 目录
- 面试题
- 基础篇
- JVM篇
- 多线程及并发篇
- Spring篇
- MyBatis篇
- SpringBoot篇
- MySQL篇
- SpringCloud篇
- Dubbo篇
- Nginx篇
- MQ篇
 - 1. 为什么要使用MQ
 - 2. MQ有什么类型
 - 3. RabbitMQ、ActiveMQ、ZeroMQ 三者中，综合来看，RabbitMQ 是首选。
 - 4. 如何保证高可用的？
 - 5. 如何保证消息的可靠传输？如果消息丢失怎么办？
 - 6. 如何保证消息的顺序性？
 - 7. 如何解耦消息队列的监听以及过期丢失？
 - 8. 让你来设计一个消息队列，你会怎么做？
- 数据结构与算法篇
- Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 网络篇
 - 设计模式
 - maven篇
 - ElasticSearch篇
 - tomcat篇
 - Git篇
 - 软实力篇

按照目前网络上的资料，RabbitMQ、activeM、ZeroMQ 三者中，综合来看，RabbitMQ 是首选。

2.持久化消息比较

ActiveMq 和 RabbitMq 都支持。持久化消息主要是控我们机器在不可抗力因素等情况下挂掉了，消息不会丢失的机制。

3.综合技术实现

可靠性、灵活的路由、集群、事务、高可用的队列、消息排序、问题追踪、可视化管理工具、插件系统等等。

RabbitMq / Kafka 最好，ActiveMq 次之，ZeroMq 最差。当然ZeroMq 也可以做到，不过自己必须手动写代码实现，代码量不小。尤其是可靠性中的：持久性、投递确认、发布者证实和高可用性。

4.高并发

毋庸置疑，RabbitMQ 最高，原因是它的实现语言是天生具备高并发高可用的erlang 语言。

5.比较关注的比较，RabbitMQ 和 Kafka

RabbitMq 比Kafka 成熟，在可用性上，稳定性上，可靠性上，RabbitMq 胜于 Kafka（理论上）。

另外，Kafka 的定位主要在日志等方面，因为Kafka 设计的初衷就是处理日志的，可以看做一个日志（消息）系统一个重要组件，针对性很强，所以如果业务方面还是建议选择 RabbitMq。

还有就显，Kafka 的性能（吞吐量、TPS）比RabbitMq 要高出来很多。

4、如何保证高可用的？

RabbitMQ 是比较有代表性的，因为是基于主从（非分布式）做高可用性的，我们就以 RabbitMQ 为例子讲解第一种 MQ 的高可用性怎么实现。RabbitMQ 有三种模式：单机模式、普通集群模式、镜像集群模式。

单机模式，就是 Demo 级别的，一般都是你本地启动了玩玩儿的？，没人生产用单机模式

普通集群模式，意思就是在多台机器上启动多个 RabbitMQ 实例，每个机器启动一个。你创建的 queue，只会放在一个 RabbitMQ 实例上，但是每个实例都同步 queue 的元数据（元数据可以认为是 queue 的一些配置信息，通过元数据，可以找到 queue 所在实例）。你消费的时候，实际上如果连接到了另外一个实例，那么那个实例会从 queue 所在实例上拉取数据过来。这方案主要是提高吞吐量的，就是说让集群中多个节点来服务某个 queue 的读写操作。

镜像集群模式：这种模式，才是所谓的 RabbitMQ 的高可用模式。跟普通集群模式不一样的是，在镜像集群模式下，你创建的 queue，无论元数据还是 queue 里的消息都会存在于多个实例上，就是说，每个 RabbitMQ 节点都有这个 queue 的一个完整映像，包含 queue 的全部数据的意义。然后每次你写消息到 queue 的时候，都会自动把消息同步到多个实例的 queue 上。RabbitMQ 有很好的管理控制台，就是在后台新增一个策略，这个策略是镜像集群模式的策略。指定的时候是可以要求数据同步到所有节点的，也可以要求同步到指定数量的节点，再次创建 queue 的时候，应用这个策略，就会自动将数据同步到其他的节点上去了。这样的话，好处在于，你任何一个机器宕机了，没事儿，其它机器（节点）还包含了这个 queue 的完整数据，别的 consumer 都可以到其它节点上去消费数据。坏处在于，第一，这个性能开销也太大了吧，消息需要同步到所有机器上，导致网络带宽压力和消耗很重！RabbitMQ 一个 queue 的数据都是放在一个节点里的，镜像集群下，也是每个节点都放这个 queue 的完整数据。

Kafka 一个最基本的架构认识：由多个 broker 组成，每个 broker 是一个节点；你创建一个 topic，这个 topic 可以划分为多个 partition，每个 partition 可以存在于不同的 broker 上，每个 partition 就放一部分数据。这版是天然的分分布式消息队列，就是说一个 topic 的数据，是分放在多个机器上的，每个机器就放一部分数据。Kafka 0.8 以后，提供了 HA 机制，就是 replica（复制品）副本机制。每个 partition 的数据都会同步到其它机器上，形成自己的多个 replica 副本。所有 replica 会选举一个 leader 出来，那么生产和消费都跟这个 leader 打交道，然后其他 replica 就是 follower。写的时候，leader 会负责把数据同步到所有 follower 上去，该的时候就直接读 leader 上的数据即可。只能读写 leader？很简单，要是你可以随意读写每个 follower，那么就要 care 数据一致性的问题，系统复杂度太高，很容易出问题。Kafka 会均匀地将一个 partition 的所有 replica 分布在不同的机器上，这样可以提高容错性。因为如果某个 broker 宕机了，没事儿，那个 broker 上面的 partition 在其他机器上都有副本的，如果这上面有个 partition 的 leader，那么此时会从 follower 中重新选举一个新的 leader 出来，大家继续读写那个新的 leader 即可。这就有所谓的高可用性了。写数据的时候，生产者就写 leader，然后 leader 将数据落地地写本地磁盘，接着其他 follower 自己主动从 leader 来 pull 数据。一旦所有 follower 同步好数据了，就会发送 ack 给 leader，leader 收到所有 follower 的 ack 之后，就会返回写成功的消息给生产者。（当然，这只是其中一种模式，还可以适当调整这个行为）消费的时候，只会从 leader 去读，但是只有当一个消息已经被所有 follower 都同步成功返回 ack 的时候，这个消息才会被消费者读到。

5、如何保证消息的可靠传输？如果消息丢了怎么办

数据的丢失问题，可能出现在生产者、MQ、消费者中

生产者丢失：生产者将数据发送到 RabbitMQ 的时候，可能数据就在半路给搞丢了，因为网络问题的，都有可能。此时可以选择用 RabbitMQ 提供的事务功能，就是生产者发送数据之前开启 RabbitMQ 事务 channel.txSelect，然后发送消息，如果消息没有成功被 RabbitMQ 接收到，那么生产者会收到异常报抛，此时就可以回滚该事务 channel.txRollback，然后重试发送消息；如果收到了消息，那么可以提交事务 channel.txCommit。吞吐量会下来，因为太耗性能。所以一般来说，如果你要确保读写 RabbitMQ 的消息丢失，可以开启 confirm 模式，在生产者那里设置开启 confirm 模式之后，你每次写的消息都会分配一个唯一的 id，然后如果写入了 RabbitMQ 中，RabbitMQ 会给

数据结构与算法篇

Linux篇

- 目录
- 面试题
- 基础篇
- JVM篇
- 多线程及并发篇
- Spring篇
- MyBatis篇
- SpringBoot篇
- MySQL篇
- SpringCloud篇
- Dubbo篇
- Nginx篇
- MQ篇
- 数据结构与算法篇
- Linux篇
 - 1. 如何查看进程？如何查看进程？如何查看进程？
 - 2. 怎么查看当前进程？怎么执行退出？怎么查看进程？
 - 3. 查看文件有哪些命令？
 - 4. 列举几个常用的Linux命令
 - 5. 你平时是怎么查看日志的？
- Zookeeper篇
- Redis篇
- 分布式篇
- 网络篇
- 设计模式
- maven篇
- ElasticSearch篇
- tomcat篇
- Git篇
- 软实力篇

-f 循环读取 -q 不显示处理信息 -v 显示详细的处理信息 <<数目> 显示的字节数 -n<行数> 显示行数 -q, --quiet, --silent 从不输出给出文件名首部 -s, --sleep-interval=S 与 -f 合用,表示在每次反复的间隔休眠S秒

例如：

```
tail -n 10 test.log 查询日志尾部最后10行的日志；
tail -n +10 test.log 查询10行之后的所有日志；
tail -fn 10 test.log 循环实时查看最后1000行记录（最常用的）
```

一般还会配合grep搜索用，例如：

```
tail -fn 1000 test.log | grep '关键字'
```

如果一次性查询的数据量太大,可以进行翻页查看，例如：

```
tail -n 4700 aa.log | more -1000 可以进行多屏显示(ctrl + f 或者 空格键可以快速键)
```

2、head

跟tail是相反的head是看前多少行日志

```
head -n 10 test.log 查询日志文件中的头10行日志；
head -n -10 test.log 查询日志文件除了最后10行的其他所有日志；
```

head其他参数参考tail

3、cat

cat 是由第一行到最后一行连续显示在屏幕上

一次显示整个文件：

```
$ cat filename
```

从键盘创建一个文件：

```
$ cat > filename
```

将几个文件合并为一个文件：

```
$ cat file1 file2 > file 只能创建新文件,不能编辑已有文件
```

将一个日志文件的内容追加到另外一个：

```
$ cat -n textfile1 > textfile2
```

清空一个日志文件：

```
$ cat > textfile2
```

注意：>意思是创建，>>是追加，千万不要再弄错了。

cat其他参数参考tail

4、more

more命令是一个基于vi编辑器文本过滤器，它以全屏的方式按页显示文本文件的内容，支持vi中的关键字定位操作。more名单中内置了若干快捷键，常用的有H（获得帮助信息），Enter（向下翻一行），空格（向下滚动一屏），Q（退出命令），more命令从前向后读取文件，因此在启动时就加载整个文件。

该命令一次显示一屏文本，满屏后停下来，并且在屏幕的底部出现一个提示信息，给出至今已显示的该文件的百分比：-More- (XX%)

- more的语法：more 文件名
- Enter 向下n行，需要定义，默认为1行
- Ctrl f 向下滚动一屏
- 空格键 向下滚动一屏
- Ctrl b 返回上一屏
- h 输出当前行的行号
- :f 输出文件名和当前行的行号
- v 调用vi编辑器
- !命令 调用Shell，并执行命令
- q退出more

5、sed

这个命令可以查找日志文件特定的一段，根据时间的一个范围查询，可以按照行号和时间范围查询

按照行号

```
sed -n '5,10p' filename 这样就可以只看文件的第5行到第10行。
```

按照时间段

```
sed -n '/2014-12-17 16:17:20/,/2014-12-17 16:17:36/p' test.log
```

6、less

Zookeeper篇

- 目录
- 数据库结构与部署
 - Linux篇
 - Zookeeper篇
 - 1. 为什么说Zookeeper是分布式系统？
 - 2. Zookeeper 有哪些应用？
 - 命名服务
 - 配置管理
 - 集群管理
 - 分布式通知与协调
 - 分布式锁
 - 分布式队列
 - 3. 简述Zookeeper的工作原理？
 - 4. 请描述一下Zookeeper的通知机制？
 - 5. Zookeeper 对节点的 watch 监听？
 - 6. Zookeeper 集群中有哪些角色？
 - 7. Zookeeper 集群中Server有哪些工作？
 - 8. Zookeeper 集群中是怎样选举leader？
 - 9. Zookeeper 是如何保证事务的顺序？
 - 10. Zookeeper 集群中个服务器怎么交互？
 - 11. Zookeeper 分布式锁怎么实现的？
 - 12. 了解Zookeeper的系统架构吗？
 - 13. Zookeeper为什么这么设计？
 - 14. 你知道Zookeeper中有哪些角色？
 - 15. 你知道Zookeeper节点ZNode吗？
 - 节点有哪些类型？
 - 节点属性有哪些？
 - 16. 请描述Zookeeper的选举流程？
 - 17. 为什么Zookeeper集群的数目，一般是奇数？
 - 18. 知道Zookeeper监听器的原理吗？
 - 19. 简述Zookeeper中的ACL 权限控制？
 - 20. Zookeeper 有哪几种认证策略？
 - 21. Zookeeper集群中事务的分布式实现？
 - 22. 简述一下 ZAB 协议？
 - 23. ZAB 和 Paxos 算法的区别？
 - 24. ZooKeeper 宕机如何处理？
 - 25. 简述一下 ZooKeeper 的 session？
 - 26. ZooKeeper 负载均衡和Nginx区别？
 - 27. 简述Zookeeper的序列化？
 - 28. 为什么Zookeeper中Zxid 是什么？有什么作用？
 - 29. 讲解一下 ZooKeeper 的持久化机制？
 - 30. Zookeeper 选举中投票策略的五大元？
 - 31. 简述Zookeeper中的副本？
 - 32. Zookeeper 能做什么原因导致？
 - 33. Zookeeper 是如何解决脑裂问题的？
 - 34. 简述 Zookeeper 的 CAP 问题？
 - 35. watch 监听为什么是一次性的？

Zab协议有两种模式，它们分别是复制模式（选主）和广播模式（同步）。

Zab协议的全称是 Zookeeper Atomic Broadcast**（Zookeeper原子广播），Zookeeper 是通过 Zab 协议来保证分布式事务的最终一致性。Zab协议要求每个 Leader 都要经历三个阶段：发现，同步，广播。

当服务器启动或者在领导者崩溃后，Zab就进入了复制模式，当领导者被选举出来，且大多数Server完成了和 leader的状态同步以后，复制模式就结束了。状态同步保证了leader和Server具有相同的系统状态。

为了保证事务的顺序一致性，zookeeper采用了递增的事务id号（zxid）来标识事务。所有的提议（proposal）都在被提出的时候加上了zxid，实现中zxid是一个64位的数字，它高32位是epoch用来标识leader关系是否改变，每次一个leader被选出来，它都会有一个新的epoch，标识当前属于那个leader的统治时期。低32位用于递增计数。

epoch：可以理解成皇帝的年号，当前的皇帝leader产生后，将有一个新的epoch年号。

每个Server在工作过程中有三种状态：

- LOOKING：当前Server不知道leader是谁，正在搜寻。
- LEADING：当前Server即为选举出来的leader。
- FOLLOWING：leader已经选举出来，当前Server与之同步。

4, 请描述一下 Zookeeper 的通知机制是什么？

Zookeeper 允许客户端向服务端的某个 znode 注册一个 Watcher 监听，当服务端的一些指定事件触发了这个 Watcher，服务端会向指定客户端发送一个事件通知来实现分布式的通知功能，然后客户端根据 Watcher 通知状态和事件类型做出业务上的改变。

大致分为三个步骤：

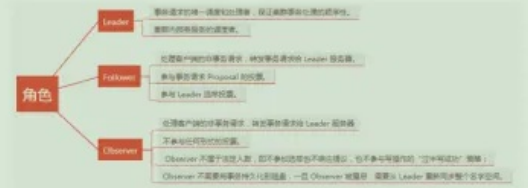
- 客户端注册 Watcher
 - 调用 `getData, getChildren, exist` 三个 API，传入 Watcher 对象。
 - 标记请求 request，封装 Watcher 到 `WatchRegistration`。
 - 封装成 `Packet` 对象，发服务端发送 request。
 - 收到服务端响应后，将 Watcher 注册到 `ZkWatcherManager` 中进行管理。
 - 请求返回，完成注册。
- 服务端处理 Watcher
 - 服务端接收 Watcher 并存储。
 - Watcher 触发。
 - 调用 `process` 方法来触发 Watcher。
 - 客户端回调 Watcher

client 就会对某个 znode 建立一个 watcher 事件，当该 znode 发生变化时，这些 client 会收到 zk 的通知，然后 client 可以根据 znode 变化来做出业务上的改变等。

5, Zookeeper 对节点的 watch 监听通知是永久的吗？

不是，一次性的。无论是服务端还是客户端，一旦一个 Watcher 被触发，Zookeeper 都会将其从相应的存储中移除。这样的设计有效的减轻了服务端的压力，不然对于更新非常频繁节点，服务端会不断的向客户端发送事件通知，无论对于网络还是服务端的压力都非常大。

6, Zookeeper 集群中有哪些角色？



在一个集群中，最少需要 3 台，或者保证 $2k + 1$ 台，即奇数。为什么保证奇数？主要是为了选举算法。

7, Zookeeper 集群中Server有哪些工作状态？

- LOOKING
 - 寻找 Leader 状态：当服务器处于该状态时，它会认为当前集群中没有 Leader，因此需要进入 Leader 选举状态
- FOLLOWING
 - 跟随者状态：表明当前服务器角色是 Follower
- LEADING
 - 领导者状态：表明当前服务器角色是 Leader

Redis篇

- 目录
- 数据库结构与部署
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 1. 为什么要用缓存？
 - 2. 缓存 Redis 有啥好处？
 - 3. 什么是 Redis？
 - 4. 为什么 缓存Redis而不缓存Memca？
 - 5. 为什么Redis处理速度比Memca也快？
 - 6. 简述Redis的存储原理？
 - 7. 为什么 Redis 需要把所有数据放到内存中？
 - 8. Redis 的同步机制了解是什么？
 - 9. pipeline 有什么好处，为什么要用？
 - 10. 说一下Redis有什么优点和缺点？
 - 11.Redis复制策略有哪些？
 - 12.Redis持久化方式有哪些？以及有什么优缺点？
 - 13. 持久化有几种，应该怎么选择？
 - 14. 如何使用 Redis 实现消息队列？
 - 15. 简述你对Redis事务的理解？
 - 16. Redis 为什么设计成单进程的？
 - 17. 什么是 bigkey？会存在什么影响？
 - 18. 简述Redis的集群模式？
 - 19. 是否使用过 Redis Cluster 集群？
 - 20. Redis Cluster 集群方案什么情况？
 - 21. Redis 集群的复制策略有哪些？
 - 22. 简述 Redis 集群的组成？
 - 23. Redis 集群的扩容和缩容方案？
 - 24. Redis 集群有 1 亿个 key，怎么办？
 - 25. 如果有大量的 key 需要删除怎么办？
 - 26. 什么情况下可能会导致 Redis 阻塞？
 - 27. 缓存和数据库谁先更新？
 - 28. 怎么避免缓存命中？
 - 29. Redis 如何解决 key 冲突？
 - 30. Redis 如何防止内存溢出？
 - 31. 简述Redis持久化机制？
 - 32. 缓存雪崩、缓存穿透、缓存预热。
 - 33. 持久化和冷数据是什么？
 - 34. Memcache与Redis的区别有哪些？
 - 35. 单线程的Redis为什么这么快？
 - 36. Redis的数据模型，以及每种数据类型？
 - 37. Redis的过期策略以及内存淘汰机制？
 - 38. Redis 为什么是单线程的？
 - 39. Redis 常见性能问题和解决方案？
 - 40. 为什么Redis的操作是原子性的？
 - 41. 了解Redis的部署吗？
 - 42. Redis 的数据模型与存储原理？

这样便完成了一次通信。不要怕这段文字，结合图看，一篇不行两篇，实在不行可以网上点点资料结合看看，一定要搞清楚，否则前面吹的牛逼就白费了。

7, 为什么 Redis 需要把所有数据放到内存中？

Redis 将数据放在内存中有一个好处，那就是可以实现最快的数据读取，如果数据存储在硬盘中，磁盘 I/O 会严重影响 Redis 的性能。而且 Redis 还提供了数据持久化功能，不用担心服务器重启对内存中数据的影响。其次现在硬件越来越便宜的情况下，Redis 的使用也会被应用得越来越多，使得它拥有很大的优势。

8, Redis 的同步机制了解是什么？

Redis 支持主从同步、从从同步。如果是第一次进行主从同步，主节点需要使用 `bgsave` 命令，再将后续修改操作记录到内存的缓冲区，等 RDB 文件全部同步到复制节点，复制节点接受完成后将 RDB 镜像加载到内存中。等加载完成后，复制节点通知主节点将复制期间修改的操作记录同步到复制节点，即可完成同步过程。

9, pipeline 有什么好处，为什么要用 pipeline？

使用 pipeline（管道）的好处在于可以将多次 I/O 往返的时间缩短为一次，但是要求管道中执行的指令之间没有因果关系。

用 pipeline 的原因在于可以实现请求/响应服务器端的功能，当客户端尚未读取响应时，它也可以处理新的请求。如果客户端存在多个命令发送到服务器时，那么客户端无需等待服务器端的每次响应才能执行下一个命令，只需最后一步从服务器读取回复即可。

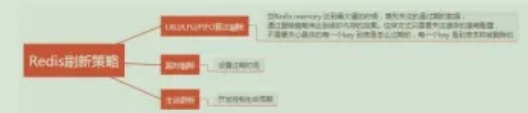
10, 说一下 Redis 有什么优点和缺点

- 优点
 - 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)。
 - 支持丰富的数据结构：支持 String，List，Set，Sorted Set，Hash 五种基础的数据结构。
 - 持久化存储：Redis 提供 RDB 和 AOF 两种数据的持久化存储方案，解决内存数据库最担心的万一 Redis 挂掉，数据会消失。
 - 高可用：内置 Redis Sentinel，提供高可用方案，实现主从故障自动转移。内置 Redis Cluster，提供集群方案，实现基于槽的分片方案，从而支持更大的 Redis 规模。
 - 丰富的特性：Key过期、计数、分布式锁、消息队列等。
- 缺点

删除数据。

- 如果进行完整重同步，由于需要生成 `rdm` 文件，并进行传输，会占用主机的 CPU，并会消耗网络的带宽。不过 Redis 2.8 版本，已经有部分重同步的功能，但是还是有完整重同步的。比如，新上线的备机。
- 修改配置文件，进行重启，将硬盘中的数据加载进内存，时间比较久。在这个过程中，Redis 不能提供服务。

11 Redis 缓存刷新策略有哪些？



12, Redis 持久化方式有哪些？以及有什么区别？

Redis 提供两种持久化机制 RDB 和 AOF 机制：

- RDB 持久化方式
 - 是指用数据快照的方式非持久化模式记录 Redis 数据库的所有键值对，在某个时间点将数据写入一个临时文件，持久化结束后，用这个临时文件替换上次持久化的文件，达到数据恢复。
 - 优点：
 - 只有一个文件 `dump.rdb`，方便持久化。
 - 容灾性好，一个文件可以保存到安全的磁盘。
 - 性能最大化，fork 子进程来完成写操作，让主进程继续处理命令，所以是 IO 最大化。使用单线程子进程来进行持久化，主进程不会进行任何 IO 操作，保证了 Redis 的高性能。
 - 相对于数据集大时，比 AOF 的启动效率更高。
 - 缺点：
 - 数据安全性低。RDB 是每隔一段时间进行持久化，如果持久化之间 Redis 发生故障，会发生数据丢失，所以这种方式更适合数据要求不严谨的时候。
- AOF=Append-only file 持久化方式

分布式篇

- 面试题
 - 基础篇
 - JVM篇
 - 多线程&并发篇
 - Spring篇
 - MyBatis篇
 - SpringBoot篇
 - MySQL篇
 - SpringCloud篇
 - Dubbo篇
 - Nginx篇
 - MQ篇
 - 数据库与算法篇
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 1. 分布式幂等性如何设计?
 - 2. 简单一次完整的 HTTP 请求所经历的步骤
 - 3. 谈谈你对分布式事务的了解
 - 4. 如何保证分布式事务的最终一致性?
 - 5. 什么是二阶段提交?
 - 6. 什么是三阶段提交?
 - 7. 什么是补偿事务?
 - 8. 消息队列怎么实现的?
 - 9. 分布式Sagas事务模型
 - 10. 分布式事务有哪几种方案?
 - 11. 幂等性设计有哪些?
 - 12. 常见分布式事务方案有哪些?
 - 13. 你知道两阶段提交吗?
 - 14. 两阶段提交是什么? (两阶段)算法
 - 15. 两阶段提交是两阶段提交
 - 16. 两阶段提交是两阶段提交
 - 17. 两阶段提交是两阶段提交
 - 18. 两阶段提交是两阶段提交
 - 19. 如何保证两阶段提交的数据一致性?
 - 20. 长连接和短连接有什么区别?
 - 21. 长连接和短连接有什么区别?
 - 22. 如何提高系统的并发能力?
 - 网络篇
 - 设计模式
 - maven篇
 - ElasticSearch篇
 - tomcat篇
 - Git篇
 - 实战案例

- 一致性：在分布式系统中的所有数据备份，在同一时刻是否同样的值。
- 可用性：在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。
- 分区容错性：以实际效果而言，分区相当于对通信的隔离要求。系统如果不能在时限内达成数据一致性，就意味着发生了分区的情况，必须就当前操作在A和A之间做出选择。

BASE理论

BASE理论是对CAP中的一致性和可用性进行一个权衡的结果，理论的核心思想就是：我们无法做到强一致，但每个应用都可以根据自身的业务特点，采用适当的方式来使系统达到最终一致性。

- Basically Available (基本可用)
- Soft state (软状态)
- Eventually consistent (最终一致性)

4. 你知道哪些分布式事务解决方案？

我目前知道的有五种：

- 两阶段提交(2PC)
- 三阶段提交(3PC)
- 补偿事务(TCC=Try-Confirm-Cancel)
- 本地消息队列(MQ)
- Sagas事务模型(最终一致性)

说完上面五种，面试官一般都会继续问下面这几个问题（可能就问一两个，也可能全问）。

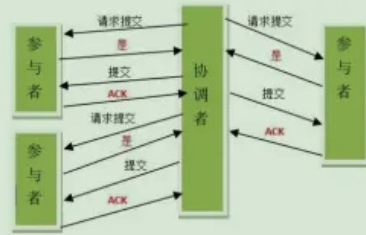
5. 什么是二阶段提交？

两阶段提交2PC是分布式事务中最强大的事务类型之一，两阶段提交就是分两个阶段提交：

- 第一阶段询问各个事务数据源是否准备好。
- 第二阶段才真正将数据提交给事务数据源。

为了保证该事务可以满足ACID，就要引入一个协调者（Coordinator），其他的节点被称为参与者（Participant）。协调者负责调度参与者的行为，并最终决定这些参与者是否要把事务进行提交。

处理流程如下：



阶段一

- 协调者向所有参与者发送事务内容，询问是否可以提交事务，并等待答复。
- 各参与者执行事务操作，将 undo 和 redo 信息记入事务日志中（但不提交事务）。
- 如参与者执行成功，给协调者反馈 yes，否则反馈 no。

阶段二

如果协调者收到了参与者的失败消息或者超时，直接给每个参与者发送回滚(rollback)消息；否则，发送提交(commit)消息。两种情况处理如下：

情况1：当所有参与者均反馈 yes，提交事务

- 协调者向所有参与者发出正式提交事务的请求（即 commit 请求）。
- 参与者执行 commit 请求，并释放整个事务期间占用的资源。
- 各参与者向协调者反馈 ack(应答)完成的消息。
- 协调者收到所有参与者反馈的 ack 消息后，即完成事务提交。

情况2：当有一个参与者反馈 no，回滚事务

- 协调者向所有参与者发出回滚请求（即 rollback 请求）。
- 参与者使用阶段 1 中的 undo 信息执行回滚操作，并释放整个事务期间占用的资源。
- 各参与者向协调者反馈 ack 完成的消息。
- 协调者收到所有参与者反馈的 ack 消息后，即完成事务。

网络篇

- 目录
 - Dubbo篇
 - Nginx篇
 - MQ篇
 - 数据库与算法篇
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 网络篇
 - 1. HTTP 请求有哪些？分哪些？
 - 2. Forward 和 Redirect 的区别？
 - 3. Get 和 Post 请求有哪些区别？
 - 4. 说说 TCP 与 UDP 的区别，以及各自特点
 - 5. 说一下 HTTP 和 HTTPS 的区别
 - 6. 说说 HTTP、TCP、Socket 的关系
 - 7. 说一下 HTTP 的长连接和短连接的区别
 - 8. TCP 为什么要三次握手，两次不行吗？
 - 9. 说一下 TCP 粘包是怎么产生的？
 - 10. TCP 如何保证可靠性
 - 11. OSI 七层模型都有哪些？
 - 12. 浏览器中输入：www.w3cjava.com
 - 13. 如何实现跨域？
 - 1. JSONP 方式
 - 2. CORS 方式
 - 3. 代理方式
 - 14. TCP 为什么三次握手，两次不行？
 - 15. 说一下 TCP 粘包是怎么产生的？
 - 16. HTTP1.0, HTTP1.1, HTTP2.0 的区别
 - 17. 说说 HTTP 协议与 TCP/IP 协议的区别
 - 18. 如何理解 HTTP 协议是无状态的？
 - 19. 什么是长连接和短连接？
 - 20. 长连接和短连接的优点？
 - 21. 说说长连接短连接的操作过程
 - 22. 说说 TCP 三次握手和四次挥手的过程
 - 23. OSI 七层模型都有哪些？
 - 24. OSI 这种分层有什么好处？
 - 25. 说说 TCP/IP 四层模型
 - 26. 说说域名解析过程
 - 27. IP 地址分为几类，每类都代表什么？
 - 28. 说说 TCP 如何保证可靠性的？
 - 设计模式
 - maven篇
 - ElasticSearch篇
 - tomcat篇
 - Git篇

request 中的信息在 servlet 间是共享的。Redirect 发起了两次 HTTP 请求分别使用不同的 request。

- 请求的次数：Forward 只有一次请求；Redirect 有两次请求。

3. Get 和 Post 请求有哪些区别？

用途：

- get 请求用来从服务器获取资源
- post 请求用来向服务器提交数据

表单的提交方式：

- get 请求直接将表单数据以 name=value1&name2=value2 的形式拼接到 URL 上 (http://www.baidu.com/action?name1=value1&name2=value2)，多个参数参数值需要用 & 连接起来并且用 > 拼接到 action 后面；
- post 请求将表单数据放到请求头或者请求的消息体中。

传输数据的大小限制：

- get 请求传输的数据受到 URL 长度的限制，而 URL 长度是由浏览器决定的；
- post 请求传输数据的大小理论上来说是没有限制的。

参数的编码：

- get 请求的参数会在地址栏明文显示，使用 URL 编码的文本格式传递参数；
- post 请求使用二进制数据多重编码传递参数。

缓存：

- get 请求可以被浏览器缓存被收藏为标签；
- post 请求不会被缓存也不能被收藏为标签。

4. 说说 TCP 与 UDP 的区别，以及各自的优缺点

- TCP面向连接（如打电话要先拨号建立连接）：UDP是无连接的，即发送数据之前不需要建立连接。
- TCP提供可靠的服务。也就是说，通过TCP连接传输的数据，无差错，不丢失，不重复，且按序到达；UDP尽最大努力交付，即不保证可靠交付。tcp通过校验和，重传控制，序号标识，滑动窗口，确认应答实现可靠传输。如差包时的重发控制，还可以对次序乱掉的包进行顺序控制。
- UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信。

4. 每一条TCP连接只能占到一个的IP地址——一对一，一对多，多对一和多对多的交互通信

5. 说一下 HTTP 和 HTTPS 的区别

端口不同：HTTP和HTTPS的连接方式不同没用的端口也不一样，HTTP是80，HTTPS用的是443
消耗资源：和HTTP相比，HTTPS通信会因为加解密的处理消耗更多的CPU和内存资源。

开销：HTTPS 通信需要证书，这类证书通常需要向认证机构申请或者付费购买。

6. 说说HTTP、TCP、Socket 的关系是什么？

- TCP/IP 代表传输控制协议/网络协议，指的是一系列协议族。
- HTTP 本身就是一个协议，是从 Web 服务器传输超文本到本地浏览器的传送协议。
- Socket 是 TCP/IP 网络的 API，其实就是一个门面模式，它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面。对用户来说，一组简单的接口就是全部，让 Socket 去组织数据，以符合指定的协议。

综上所述：

- 需要 IP 协议来连接网络
- TCP 是一种允许我们安全传输数据的机制，使用 TCP 协议来传输数据的 HTTP 是 Web 服务器和客户端使用的特殊协议。
- HTTP 基于 TCP 协议，所以可以使用 Socket 去建立一个 TCP 连接。

7. 说一下HTTP的长连接与短连接的区别

HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。

短连接

在HTTP/1.0中默认使用短连接，也就是说，浏览器和服务端进行一次HTTP操作，就建立一次连接，但任务结束后就中断连接。如果客户端访问的某个HTML或其他类型的Web资源，如 JavaScript 文件、图像文件、css 文件等。当浏览器遇到这样一个Web资源，就会建立一个HTTP会话。

长连接

从HTTP/1.1起，默认使用长连接，用以保持连接特性。在使用长连接的情况下，当一个网页打开完成后，客户端和服务端之间用于传输HTTP数据的TCP连接不会关闭。如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。

8. TCP 为什么要三次握手，两次不行吗？为什么？

设计模式

- 面试题
 - 基础知识
 - JVM篇
 - 多线程及并发篇
 - Spring篇
 - MyBatis篇
 - SpringBoot篇
 - MySQL篇
 - SpringCloud篇
 - Dubbo篇
 - Nginx篇
 - MQ篇
 - 数据库与算法篇
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 网络篇
 - 设计模式
 - maven篇
 - 1. 什么是maven?
 - 2. Maven能帮我们解决什么问题?
 - 3. 说说maven有什么优缺点?
 - 4. 什么是Maven的坐标?
 - 5. 讲一下Maven的生命周期?
 - 6. 说说你熟悉哪些maven命令?
 - 7. 如何解除依赖传递引起的版本冲突?
 - 8. 说说maven的依赖原则?
 - 9. 说说maven的解析机制?
 - 10. 说说maven的解析机制?
 - 11. 说说maven的解析机制?
 - ElasticSearch篇
 - tomcat篇
 - Git篇
 - 软实力篇

比如，以上面的程序员A为例，来天我的程序员B了，这个程序员B想喝奶茶，这个奶茶就是直接修改简单工厂里面的代码，这种做法不但不够优雅，也不符合软件设计的“开闭原则”，因为每次新增品类都要修改原来的代码。这个时候就可以使用抽象工厂类了，抽象工厂里只声明方法，具体的实现交给子类（子工厂）去实现，这个时候再新增品类的需求，只需要新创建代码即可。

5、装饰器模式是什么

答：装饰器模式是动态地给一个对象增加一些额外的功能，同时又不改变其结构。

优点：装饰类和被装饰类可以独立发展，不会相互耦合，装饰器是继承的一个替代模式，装饰模式可以动态扩展一个实现类的功能。

装饰器模式的关键：装饰器中使用了被装饰的对象。

比如，创建一个对象“laowang”，给对象添加不同的装饰，穿上夹克，戴上帽子.....，这个执行过程就是装饰器模式。

一句名言：人靠衣裳马靠鞍，都是装饰器模式的生活写照。

6、代理模式和装饰器模式有什么区别？

答：都是结构型模式，代理模式重在访问权限的控制，而装饰器模式重在功能的加强。

7、模板方法模式

答：模板方法模式是指定义一个算法骨架，将具体内容延迟到子类去实现。

优点：

- 提高代码复用性：将相同部分的代码放在抽象的父类中，而将不同的代码放入不同的子类中；
- 实现了反向控制：通过一个父类调用其子类的操作，通过对子类的具体实现扩展不同的行为，实现了反向控制并且符合开闭原则。

喝茶茶：烧水—放入茶叶—喝茶。放入的茶叶每个人自己的喜好不一样，有的是普洱、有的是铁观音等等。

每日工作：上班打卡—工作—下班打卡。每个人工作的内容不一样，后端开发的、前端开发、测试。产品每个人的工作内容不一样。

8、知道享元模式吗？

答：顾名思义就是被共享的单元。享元模式的意图是复用对象，节省内存，前提是享元对象是不可变对象。

另一种讲法：当一个系统中存在大量相同类型的对象，如果这些对象都是不可变对象，我们就可以利用享元模式将对象设计成享元，在内存中只保留一份实例，供多处代码引用。这样可以减少内存中对象的数量，起到节省内存的目的。

典型的使用场景：Integer中cache，就是享元模式很经典的实现。

怎么看起来享元模式和单例模式是一毛一样的？面试官很有可能继续追问：

9、享元模式和单例模式的区别？

答：单例模式是创建型模式，重在只能有一个对象。而享元模式是结构型模式，重在节约内存使用，提升程序性能。

享元模式：把一个或者多可对象雷同起来，用的时候，直接从缓存里获取。也就是说享元模式不一定只有一个对象。

10、说说策略模式在我们生活的场景？

答：策略模式是指定义一系列算法，将每个算法封装起来，并且使他们之间可以相互替换。

优点：遵循了开闭原则，扩展性良好。

缺点：随着策略的增加，对外暴露越来越多。

条条大路通罗马，条条大路通北京。

我们去北京的交通方式（策略）很多，比如说坐飞机、坐高铁、自己开车等方式。每一种方式就可以理解为每一种策略。

这就是生活中的策略模式。

11、知道责任链模式吗？

答：是行为型设计模式之一，其将链中每一个节点看作是一个对象，每个节点处理请求均不同，且内部自动维护一个下一节点对象。当一个请求从链式的首端发出时，会沿着链的路径依次传递给每一个节点对象，直至有对象处理这个请求为止。

优点

- 解耦了请求与处理；
- 请求处理者（节点对象）只需关注自己感兴趣的请求进行处理即可，对于不感兴趣的请求，直接转发给下一级节点对象；
- 具备链式传递处理请求功能，请求发送者无需知晓链结构，只需等待请求处理结果；
- 链结构灵活，可以通过改变链结构动态地新增或删减责任；
- 易于扩展新的请求处理类（节点），符合 开闭原则；

maven篇

- 目录
 - 面试题
 - 基础知识
 - JVM篇
 - 多线程及并发篇
 - Spring篇
 - MyBatis篇
 - SpringBoot篇
 - MySQL篇
 - SpringCloud篇
 - Dubbo篇
 - Nginx篇
 - MQ篇
 - 数据库与算法篇
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 网络篇
 - 设计模式
 - maven篇
 - 1. 什么是maven?
 - 2. Maven能帮我们解决什么问题?
 - 3. 说说maven有什么优缺点?
 - 4. 什么是Maven的坐标?
 - 5. 讲一下Maven的生命周期?
 - 6. 说说你熟悉哪些maven命令?
 - 7. 如何解除依赖传递引起的版本冲突?
 - 8. 说说maven的依赖原则?
 - 9. 说说maven的解析机制?
 - 10. 说说maven的解析机制?
 - 11. 说说maven的解析机制?
 - ElasticSearch篇
 - tomcat篇
 - Git篇
 - 软实力篇

开发过程中我们需要用到很多jar包，每个jar包在官网获取的方式不尽相同，给工作带来了额外困难。但是使用Maven可以以坐标的方式下载一个jar包，Maven从中央仓库进行下载，并同时下载这个jar包依赖的其他jar包。

④将项目拆分为多个工程模块

项目的规模越来越大，已经不可能通过package结构来划分模块，必须将项目拆分为多个工程协同开发。

3、说说maven有什么优缺点？

优点

- 简化了项目依赖管理
- 易于上手，对于新手来说了解几个常用命令即可满足日常工作
- 便于与持续集成工具（jenkins）整合
- 便于项目升级，无论是项目本身还是项目使用的依赖
- maven有很多插件，便于功能扩展，比如生产站点，自动发布版本等
- 为什么使用Maven中的各点

缺点

- Maven是一个庞大的构建系统，学习难度大。（很多都可以这样说，入门容易[优点]但是精通难[缺点]）
- Maven采用约定优于配置的策略，虽然上手容易但是一旦出现问题，难于调试中网络环境较差，很多repository无法访问

5、什么是Maven的坐标？

Maven其中一个核心的作用就是管理项目的依赖，引入我们所需的各种jar包等。为了能自动化的解析任何一个Java构件，Maven必须将这些jar包或者其他资源进行唯一标识，这是管理项目的依赖的基础，也就是我们要说的坐标。包括我们自己开发的项目，也是要通过坐标进行唯一标识的，这样才能才其它项目中进行依赖引用。

maven的坐标通过groupid, artifactid, version唯一标志一个构件。groupid通常为公司或组织名字，artifactid通常为项目名称，versionid为版本号。

6、讲一下maven的生命周期

Maven的 生命周期：从我们的项目构建，一直到项目发布的这个过程。



阶段	处理	描述
validate	验证项目	验证项目是否正确且所有必需资源可用的
compile	执行编译	源代码编译成字节码
test	测试	使用适当的单元测试框架例如JUnit进行测试。
package	打包	创建JAR/WAR包或在pom.xml中定义提及的包
verify	检查	对集成测试的结果进行检查，以保证质量达标
install	安装	安装打包的项目到本地仓库，以供其他项目使用
deploy	部署	将完成的工程包到远程仓库中，以供其他开发人员使用

7、说说你熟悉哪些maven命令？

mvn archetype:generate 创建Maven项目

mvn compile 编译源代码

mvn deploy 发布项目

mvn test-compile 编译测试源代码

mvn test 运行应用程序中的单元测试

mvn site 生成项目相关信息的网站

mvn clean 清除项目目录中的生成结果

mvn package 根据项目生成的jar

mvn install 在本地Repository中安装jar

mvn eclipse:eclipse 生成eclipse项目文件

mvn jetty:run 启动jetty服务

mvn tomcat:run 启动tomcat服务

mvn clean package -Dmaven.test.skip=true 清除以前的包后重新打包，跳过测试类

8、如何解决依赖传递引起的版本冲突？

可通过dependency的exclusion元素排除依赖。

9、说说maven的依赖原则

- 最短路径原则（依赖传递的路径越短越优先）

ElasticSearch篇

- 目录
 - 面试篇
 - 基础篇
 - JVM篇
 - 多线程与并发篇
 - Spring篇
 - MyBatis篇
 - SpringBoot篇
 - MySQl篇
 - SpringCloud篇
 - Dubbo篇
 - Nginx篇
 - MQ篇
 - 数据结构与算法篇
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 网络篇
 - 设计模式
 - maven篇
 - ElasticSearch篇
 - 1. 索引原理与倒排索引的原理
 - 2. 索引分片存储的原理
 - 3. 索引分片合并的原理
 - 4. 了解文本索引 TF-IDF
 - 5. 能说说ElasticSearch 写索引的逻辑吗？
 - 6. 熟悉ElasticSearch 集群中搜索数据的过程吗？
 - 7. 了解ElasticSearch 深翻页的问题及解决吗？
 - 8. 熟悉ElasticSearch 性能优化
 - tomcat篇
 - 1. Tomcat的默认端口是多少，怎么修改？
 - 2. tomcat 有哪几种Connector 运行模式？
 - 3. Tomcat有哪几种部署方式？
 - 4. tomcat容器是如何创建servlet类实例？
 - 5. tomcat 如何优化？
 - 6. 熟悉tomcat的哪些配置？
 - 7. Tomcat是什么？
 - 8. 什么是Servlet呢？
 - 9. 什么是Servlet呢？
 - 10. 为什么我们将tomcat称为Web容器呢？
 - 11. tomcat是如何处理Http请求流程的？
 - 12. tomcat结构目录有哪些？
 - Git篇
 - 1. 基础篇

TF-IDF = TF / IDF

复杂的公式，就不写了，主要理解他的思想即可。

5、能说说ElasticSearch 写索引的逻辑吗？

ElasticSearch 是集群的 = 主分片 + 副本分片。

写索引只能写主分片，然后主分片同步到副本分片上。但主分片不是固定的，可能网络原因，之前还是 Node1 是主分片，后来就变成了 Node2 经过选举成了主分片了。

客户端如何知道哪个是主分片呢？看下面过程。

1. 客户端向某个节点 NodeX 发送写请求
2. NodeX 通过文档信息，请求会转发到主分片的节点上
3. 主分片处理完，通知到副本分片同步数据，向 NodeX 发送成功信息。
4. NodeX 将处理结果返回给客户端。

6、熟悉ElasticSearch 集群中搜索数据的过程吗？

1. 客户端向集群发送请求，集群随机选择一个 NodeX 处理这次请求。
2. NodeX 先计算文档在哪个主分片上，比如有主分片 A，它有三个副本 A1，A2，A3。那么请求会轮流三个副本中的一个完成请求。
3. 如果无法确认分片，比如检索的不是一个文档，就遍历所有分片。

补充一点，一个节点的存储量是有限的，于是有了分片的概念。但是分片可能有丢失，于是有了副本的概念。

比如：

ES 集群有 3 个分片，分片 A、分片 B、分片 C，那么分片 A + 分片 B + 分片 C = 所有数据，每个分片只有大概 1/3。分片 A 又有副本 A1 A2 A3，数据都是一样的。

7、了解ElasticSearch 深翻页的问题及解决吗？

深翻页：比如我们检索一次，返回所有分片，汇集结果，根据 TF-IDF 等算法打分，排序后将前 10 条数据返回。用户感觉不错，说我要看下一页。ES 依然是检索所有分片，汇集结果，根据 TF-IDF 等算法打分，排序后将前 11-20 条数据返回。

对用户来说，翻页应该很快，但是实际上，第一次检索多复杂，下一次检索就多复杂。

解决的话，可以把用户的检索结果，存入 Redis 中 10 分钟。这样翻页就很快，超过 10 分钟，用户不翻页，也就不会翻页了，数据就可以清除了。

8、熟悉ElasticSearch 性能优化

1. 批量提交

背景是大量的写操作，每次提交都是一次网络开销，网络永久是优化要考虑的重点。

2. 优化硬盘

索引文件需要落地硬盘，段的想法又带来了更多的小文件，磁盘 IO 是 ES 的性能瓶颈，一个固态硬盘比普通硬盘好太多。

3. 减少副本数量

副本可以保证集群的可用性，但是严重影响了写索引的效率。写索引时不只要写入索引，还要完成索引到副本的同步。ES 不是存储引擎，不要考虑数据丢失，性能更重要。如果是批量导入，建议就关闭副本。

9、ElasticSearch 查询优化手段有哪些？

设计阶段预优化

- (1) 根据业务增量需求，采取基于日期模板创建索引，通过 roll over API 滚动索引；
- (2) 使用别名进行索引管理；
- (3) 每天凌晨定时对索引进行 force_merge 操作，以释放空间；
- (4) 采取冷热分离机制，热数据存到 SSD，提高检索效率；冷数据定期 shrink 操作，以缩减存储；
- (5) 采取 curator 进行索引的生命周期管理；
- (6) 仅针对需要分词的字段，合理的设置分词器；
- (7) Mapping 阶段充分结合各个字段的属性，是否需要检索，是否需要存储等。.....

写入调优

- (1) 写入前副本数设置为 0；
- (2) 写入前关闭 refresh_interval 设置为 -1，禁用刷新机制；
- (3) 写入过程中：采取 bulk 批量写入；
- (4) 写入后恢复副本数和刷新间隔；
- (5) 尽量使用自动生成的 id。

查询调优

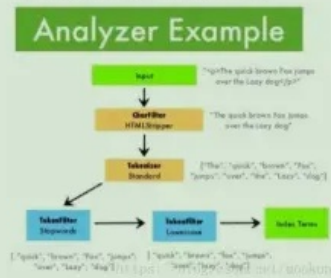
- (1) 禁用 wildcard；
- (2) 禁用批量 terms (成百上千的场景)；

tomcat篇

- 目录
 - 面试篇
 - 基础篇
 - JVM篇
 - 多线程与并发篇
 - Spring篇
 - MyBatis篇
 - SpringBoot篇
 - MySQl篇
 - SpringCloud篇
 - Dubbo篇
 - Nginx篇
 - MQ篇
 - 数据结构与算法篇
 - Linux篇
 - Zookeeper篇
 - Redis篇
 - 分布式篇
 - 网络篇
 - 设计模式
 - maven篇
 - ElasticSearch篇
 - tomcat篇
 - 1. Tomcat的默认端口是多少，怎么修改？
 - 2. tomcat 有哪几种Connector 运行模式？
 - 3. Tomcat有哪几种部署方式？
 - 4. tomcat容器是如何创建servlet类实例？
 - 5. tomcat 如何优化？
 - 6. 熟悉tomcat的哪些配置？
 - 7. Tomcat是什么？
 - 8. 什么是Servlet呢？
 - 9. 什么是Servlet呢？
 - 10. 为什么我们将tomcat称为Web容器呢？
 - 11. tomcat是如何处理Http请求流程的？
 - 12. tomcat结构目录有哪些？
 - Git篇
 - 1. 基础篇

21、什么是ElasticSearch中的过滤器？

数据由Tokenizer处理后，在编制索引之前，过滤器会对其进行处理。



22、启用属性，索引和存储的用途是什么？

enabled属性适用于各类ElasticSearch特定/创建领域，如index和size。用户提供的字段没有“启用”属性。存储意味着数据由Lucene存储，如果查询，将返回这些数据。

存储字段不一定是可搜索的。默认情况下，字段不存储，但源文件是完整的。因为您希望使用默认值(这是有意义的)，所以不要设置store属性 该指数属性用于搜索。

索引属性只能用于搜索。只有索引域可以进行搜索。差异的原因是在分析期间对索引字段进行了转换，因此如果需要的话，您不能检索原始数据。

tomcat篇

1、Tomcat的默认端口是多少，怎么修改？

默认端口为8080，可以通过在tomcat安装包conf目录下，service.xml中的Connector元素的port属性来修改端口。

这三种模式的不同之处如下：

BIO：一个线程处理一个请求。缺点：并发量高时，线程数较多，浪费资源。Tomcat7版本或更低版本中，在Linux系统中默认使用这种方式。

NIO：利用java的异步IO处理，可以通过少量的线程处理大量的请求。tomcat8.0.x中默认使用的是NIO。Tomcat7必须修改Connector配置来启动：

```
<Connector port="8080" protocol="org.apache.coyote.http11.Http11NioProtocol"
connectionTimeout="20000" redirectPort="8443"/>
```

APR：即Apache Portable Runtime，从操作系统层面解决io阻塞问题。Tomcat7或Tomcat8在Win7或以上的系统中启动默认使用这种方式。

3、Tomcat有几种部署方式？

- 利用Tomcat的自动部署：把web应用拷贝到webapps目录(生产环境不建议放在该目录中)。Tomcat在启动时会加载目录下的应用，并将编译后的结果放入work目录下。
- 使用Manager App控制台部署：在Tomcat主页点击“Manager App”进入应用管理控制台，可以指定一个web应用的路径为war文件。
- 修改 conf/server.xml 文件部署：在 server.xml 文件中，增加Context节点可以部署应用。
- 增加自定义的Web部署文件：在 conf/Catalina/localhost/ 路径下增加 xyz.xml文件，内容是Context节点，可以部署应用。

4、tomcat容器是如何创建servlet类实例？用到了什么原理？

1. 当容器启动时，会读取在webapps目录下所有的web应用中的web.xml文件，然后对 xml文件进行解析，并读取servlet注册信息。然后，将每个应用中注册的servlet类都进行加载，并通过反射的方式实例化。(有时候也是在第一次请求时实例化)
2. 在servlet注册时加上1如果为正数，则从一开始就实例化，如果不写或为负数，则第一次请求实例化。

5、tomcat 如何优化？

tomcat作为Web服务器，它的处理性能直接关系到用户体验，下面是几种常见的优化措施：

掉对web.xml的监视，把jsp提前编译成Servlet。有冗余物理内存的情况，加大tomcat使用的jvm的内存

服务器所能提供CPU、内存、硬盘的性能对处理能力有决定性影响。

- 对于高并发情况下会有大量的运算，那么CPU的速度会直接影响到处理速度。

Git篇



软实力篇



面试突击手册

目录

- 一 备战面试
 - 1.1 如何准备面试
 - 1.2 程序员简历就该这样写
 - 1.3 大部分程序员在面试前很关心的一些问题
 - 1.4 如何学习？学会各种框架有必要吗？
- 二 Java基础+集合+多线程+JVM
 - 2.1 Java基础
 - 2.2 Java集合
 - 2.3 多线程
 - 2.4 JVM
- 三 计算机基础
 - 3.1 计算机网络
 - 3.2 数据结构
 - 3.3 算法
 - 3.4 操作系统
- 四 数据库面试题总结
 - 4.1 MySQL
 - 4.2 Redis
- 五 常用框架面试题总结
 - 5.1 Spring面试题总结
 - 5.2 MyBatis面试题总结
 - 5.3 Kafka面试题总结
 - 5.4 Netty 面试题总结
- 六 认证授权
 - 6.1 认证授权面试题总结
 - 参考
- 七 优质面经
 - 五面阿里,终获offer
 - 蚂蚁金服实习生面经总结
 - Bigo的Java面试,我挂在了第三轮技术面上.....
 - 2020年字节跳动面试总结
 - 2019年蚂蚁金服、头条、拼多多的面试总结
 -
 - 逆流而行！从考研失败到收获到自己满意的Offer,分享一下自己的经历
 - Java后端实习面经,电子科大大三读者投稿！看了之后感触颇深！很
 - 关于我
 - 准备面试
 - 面试真题
 - 写在最后
- 八 微服务/分布式
- 九 真实大厂面试现场
 - 我和阿里面试官的一次邂逅(上)
 - 我和阿里面试官的一次邂逅(下)
- 十 开源项目推荐

书签

- 一 备战面试
 - 1.1 如何准备面试
 - 1.1.1 如何获取大厂面试机会...
 - 1.1.2 准备自己的自我介绍
 - 1.1.3 搞清楚技术面可能会问...
 - 1.1.4 休闲着装即可
 - 1.1.5 随身带上自己的成绩单...
 - 1.1.6 如果需要笔试就提前刷...
 - 1.1.7 花时间一些逻辑题
 - 1.1.8 准备好自己的项目介绍
 - 1.1.9 提前准备技术面试
 - 1.1.10 面试之前做好定向复习
 - 1.1.11 面试之后记得复盘
 - 1.2 程序员简历就该这样写
 - 1.2.1 为什么说简历很重要?
 - 1.2.2 关于简历你必须知道的...
 - 1.1.8 准备好自己的项目介绍
 - 1.1.9 提前准备技术面试
 - 1.1.10 面试之前做好定向复习
 - 1.1.11 面试之后记得复盘
 - 1.2.3 写简历必须了解的两大...
STAR法则 (Situation Tas...

不论是校招还是社招都避免不了各种面试、笔试，如何去准备这些东西就显得格外重要。不论是笔试还是面试都是有章可循的，我这个“有章可循”说的意思只是说应对技术面试是可以提前准备。我其实特别不喜欢那种临近考试就提前背啊记啊各种题的行为，非常反对！我觉得这种方法特别极端，而且在稍有一点经验的面试官面前是根本没有用的。建议大家还是一步一个脚印踏踏实实地走。

1.1 如何准备面试

1.1.1 如何获取大厂面试机会？

在讲如何获取大厂面试机会之前，先来给大家科普/对比一下两个校招非常常见的概念—春招和秋招。

1. 招聘人数：秋招多于春招；
2. 招聘时间：秋招一般7月左右开始，大概一直持续到10月底。**但是大厂（如BAT）都会早开始早**

结束，所以一定要把握好时间。春招最佳时间为3月，次佳时间为4月，进入5月基本就不会再有春招了（金三银四）。

3. 应聘难度：秋招略大于春招；
4. 招聘公司：秋招数量多，而春招数量较少，一般为秋招的补充。

综上，一般来说，秋招的含金量明显是高于春招的。

下面我就说一下我自己知道的一些方法，不过应该也涵盖了大部分获取面试机会的方法。

1. 关注大厂官网，随时投递简历（走流程的网申）；
2. 线下参加宣讲会，直接投递简历；
3. 找到师兄师姐/认识的人，帮忙内推（能够让你避开网申简历筛选、笔试筛选，还是挺不错的，

书签

- 1.2.6 排版注意事项
- 1.2.7 其他的一些小tips
- 1.2.8 推荐的工具/网站
- 1.3 大部分程序员在面试前很...
 - 1.3.1 我是双非/三本/专科学...
 - 1.3.2 非计算机专业的学生能...
 - 1.3.3 如何学好Java后端呢?
 - 1.3.4 我没有实习经历的话找...
 - 1.3.5 我该如何准备面试呢? ...
 - 1.3.6 我该自学还是报培训班...
 - 1.3.7 没有项目经历/博客/Gi...
 - 没有项目经验怎么办?
 - 没有博客怎么办?
 - 没有开源项目怎么办?
 - 1.3.8 从招聘要求看大厂青睐...
- 1.4 如何学习? 学会各种框架有...
 - 1.4.1 我该如何学习?
 - 1.4.2 学会各种框架有必要吗...
- 二 Java基础+集合+多线程+JVM

1.3 大部分程序员在面试前很关心的一些问题

身边的朋友或者公众号的粉丝很多人都向我询问过：“我是双非/三本/专科学校的，我有机会进入大厂吗？”、“非计算机专业的学生能学好吗？”、“如何学习Java？”、“Java学习该学那些东西？”、“我该如何准备Java面试？”.....这些方面的问题。我会根据自己的一点经验对大部分人关心的这些问题进行答疑解惑。

希望这篇可以给已经在Java方向走了几年的朋友或者正在准备往Java后端方向发展的朋友们一点帮助。道理懂了如果没有实际行动，那这篇文章对你或许没有任何意义。

如果觉得内容不错的话，可以分享给到朋友圈让你的朋友看到，感谢！

1.3.1 我是双非/三本/专科学校的，我有机会进入大厂吗？

我自己也是非985非211学校的，结合自己的经历以及一些朋友的经历，我觉得让我回答这个问题再好不过。

首先，我觉得学校歧视很正常，真的太正常了，如果要抱怨的话，你只能抱怨自己没有进入名校。但是，千万不要动不动说自己学校差，动不动拿自己学校当做自己进不了大厂的借口，学历只是筛选简历的很多标准中的一个而已，如果你够优秀，简历够丰富，你也一样可以和名校同学一起同台竞争。

企业HR肯定是更喜欢高学历的人，毕竟985，211优秀人才比例肯定比普通学校高很多，HR团队肯定会优先在这些学校里选。这就好比相亲，你是愿意在很多优秀的人中选一个优秀的，还是愿意在很多普通的人中选一个优秀的呢？双非本科甚至是二本、三本甚至是专科的同学也有很多进入大厂的，不过比率相比于名校的低很多而已。从大厂招聘的结果上看，高学历人才的数量占据大头，那些成功进入

书签

- 二 Java基础+集合+多线程+JVM
 - 2.1 Java基础
 - 1. 面向对象和面向过程的区别
 - 2. Java 语言有哪些特点?
 - 3. 关于 JVM JDK 和 JRE 最...
 - JVM
 - JDK 和 JRE
 - 4. Oracle JDK 和 OpenJDK...
 - 5. Java 和 C++的区别?
 - 6. 什么是 Java 程序的主类 ...
 - 7. Java 应用程序与小程序之...
 - 8. 字符型常量和字符串常量...
 - 9. 构造器 Constructor 是否...
 - 10. 重载和重写的区别
 - 重载
 - 重写
 - 11. Java 面向对象编程三大...
 - 封装
 - 继承

2.1 Java基础

1. 面向对象和面向过程的区别

- **面向过程**：面向过程性能比面向对象高。因为类调用时需要实例化，开销比较大，比较消耗资源，所以当性能是最重要的考量因素的时候，比如单片机、嵌入式开发、Linux/Unix 等一般采用面向过程开发。但是，面向过程没有面向对象易维护、易复用、易扩展。
- **面向对象**：面向对象易维护、易复用、易扩展。因为面向对象有封装、继承、多态性的特性，所以可以设计出低耦合的系统，使系统更加灵活、更加易于维护。但是，面向对象性能比面向过程低。

参见 issue：[面向过程：面向过程性能比面向对象高??](#)

这个并不是根本原因，面向过程也需要分配内存，计算内存偏移量，Java 性能差的主要原因并不是因为它是面向对象语言，而是 Java 是半编译语言，最终的执行代码并不是可以直接被 CPU 执行的二进制机械码。

而面向过程语言大多都是直接编译成机械码在电脑上执行，并且其它一些面向过程的脚本语言性能也并不一定比 Java 好。

2. Java 语言有哪些特点?

1. 简单易学;
2. 面向对象 (封装, 继承, 多态);
3. 平台无关性 (Java 虚拟机实现平台无关性);

书签

- 一 备战面试
- 二 Java基础+集合+多线程+JVM
- 三 计算机基础
- 四 数据库面试题总结
- 五 常用框架面试题总结
- 六 认证授权
- 七 优质面经
- 五面阿里 终获 offer
 - 前言
 - 一面(技术面)
 - 二面(技术面)
 - 三面(技术面)
 - 四面(半个技术面)
 - 五面(HR面)
 - 总结
- 蚂蚁金服实习生面经总结
- Bigo的Java面试, 我挂在了第...
- 2020字节跳动面试题总结
- 2019年蚂蚁金服、头条、拼多...

前言

在接触 Java 之前我接触的比较多的就是硬件方面，用的比较多的语言就是C和C++。到了大三我才正式选择 Java 方向，到目前为止使用Java到现在大概有一年多的时间，所以Java算不上很好。刚开始投递的时候，实习刚辞职，也没准备笔试面试，很多东西都忘记了。所以，刚开始我并没有直接就投递阿里，毕竟心里还是有一点点小害怕的。于是，我就先投递了几个不算大的公司来练手，就是想着刷刷经验而已或者说是练练手 (ps: 还是挺对不起那些公司的)。面了一个月其他公司后，我找了我实验室的学长内推我，后面就有了这5次面试。

下面简单的说一下我的这5次面试：4次技术面+1次HR面，希望我的经历能对你有所帮助。

一面(技术面)

1. 自我介绍 (主要讲自己会的技术细节，项目经验，经历那些就一语带过，后面面试官会问你的)。
2. 聊聊项目 (就是一个很普通的分布式商城，自己做了一些改进)，让我画了整个项目的架构图，然后针对项目抛了一系列的提高性能的问题，还问了我做项目的过程中遇到了那些问题，如何解决的，差不读就这些吧。
3. 可能是我前面说了我会数据库优化，然后面试官就开始问索引、事务隔离级别、悲观锁和乐观锁、索引、ACID、MVVC这些问题。
4. 浏览器输入 URL 发生了什么? TCP和UDP区别? TCP如何保证传输可靠性?
5. 讲下跳表怎么实现的?哈夫曼编码是怎么回事? 非递归且不用额外空间 (不用栈)，如何遍历二叉树
6. 后面又问了很多JVM方面的问题，比如Java内存模型、常见的垃圾回收器、双亲委派模型这些
7. 你有什么问题要问吗?

我和阿里面试官的一次邂逅(上)

自我介绍

项目介绍

消息队列

Redis

计算机网络

Java基础

我和阿里面试官的一次邂逅(下)

操作系统

内存管理机制主要是做什...

操作系统的内存管理机制...

分页机制和分块机制对比

逻辑地址和物理地址

进程和线程

多线程

为什么要使用多线程?

多线程死锁

从实现一个线程安全的单...

从CPU缓存模型聊到JMM(...)

我和阿里面试官的一次邂逅(上)

本文的内容都是根据读者投稿的真实面试经历改编而来，首次尝试这种风格的文章，花了几天晚上才总算写完，希望对你有帮助。

本文主要涵盖下面的内容：

1. 分布式商城系统：架构图讲解；
2. 消息队列相关：削峰和解耦；
3. Redis 相关：缓存穿透问题的解决；
4. 一些基础问题：
 - o 网络相关：1.浏览器输入 URL 发生了什么？ 2.TCP 和 UDP 区别？ 3.TCP 如何保证传输可靠性？
 - o Java 基础：1. 既然有了字节流,为什么还要有字符流？ 2.深拷贝 和 浅拷贝有啥区别呢？

下面是正文！

面试开始，坐在我前面的就是这次我的面试官吗？这发量看着根本不像程序员啊？我心里正嘀咕着，只听面试官说：“小伙，下午好，我今天就是你的面试官，咱们开始面试吧！”。

自我介绍