Java面试通关要点 汇总集【最终版】

原创 *2018-02-28 梁桂钊* 服务端思维

首先，声明下，以下知识点并非阿里的面试题。这里，笔者结合自己过往的面试经验，整理了一些核心的知识清单，帮助读者更好地回顾与复习 Java 服务端核心技术。本文会以引出问题为主，后面有时间的话，笔者陆续会抽些重要的知识点进行详细的剖析与解答。敬请关注「服务端思维」微信公众号，获取最新文章。

基础篇

基本功

- 面向对象的特征
- final, finally, finalize 的区别
- int 和 Integer 有什么区别
- 重载和重写的区别
- 抽象类和接口有什么区别
- 说说反射的用途及实现
- 说说自定义注解的场景及实现

- HTTP 请求的 GET 与 POST 方式的区别
- session 与 cookie 区别
- session 分布式处理
- JDBC 流程
- MVC 设计思想
- equals 与 == 的区别

集合

- List 和 Set 区别
- List 和 Map 区别
- Arraylist 与 LinkedList 区别
- ArrayList 与 Vector 区别
- HashMap 和 Hashtable 的区别
- HashSet 和 HashMap 区别
- HashMap 和 ConcurrentHashMap 的区别
- HashMap 的工作原理及代码实现
- ConcurrentHashMap 的工作原理及代码实现

线程

- 创建线程的方式及实现

守护线程和用户线程的区别在于：守护线程依赖于创建它的线程，而用户线程则不依赖

- sleep() 、join（ ）、yield（ ）有什么区别
- 说说 CountDownLatch 原理

CountDownLatch内部维护一个计数器，计数器的值为待完成的任务数N，需要等待这N个任务完成的线程调用CountDownLatch的await()方法使自己进入休眠等待状态。

当某一个任务线程完成某一个任务后调用CountDownLatch的countDown()方法来表示自己的任务已完成，此时CountDownLatch的计数器值减１，当所有的任务完成式，计数器的值为０。当计数器值为0时，CountDownLatch将唤醒所有因await()方法进入休眠的线程。

- 说说 CyclicBarrier 原理

CyclicBarrier是一个同步工具类，它允许一组线程在到达某个栅栏点(common barrier point)互相等待，发生阻塞，直到最后一个线程到达栅栏点，栅栏才会打开，处于阻塞状态的线程恢复继续执行.它非常适用于一组线程之间必需经常互相等待的情况。CyclicBarrier字面理解是循环的栅栏，之所以称之为循环的是因为在等待线程释放后，该栅栏还可以复用

- 说说 Semaphore 原理

Semaphore翻译成字面意思为 信号量，Semaphore可以控同时访问的线程个数，通过 acquire() 获取一个许可，如果没有就等待，而 release() 释放一个许可。

CountDownLatch和CyclicBarrier都能够实现线程之间的等待，只不过它们侧重点不同：

CountDownLatch一般用于某个线程A等待若干个其他线程执行完任务之后，它才执行；

而CyclicBarrier一般用于一组线程互相等待至某个状态，然后这一组线程再同时执行；

另外，CountDownLatch是不能够重用的，而CyclicBarrier是可以重用的。

Semaphore其实和锁有点类似，它一般用于控制对某组资源的访问权限。

- 说说 Exchanger 原理

Exchanger可以在两个线程之间交换数据，只能是2个线程，他不支持更多的线程之间互换数据

- 说说 CountDownLatch 与 CyclicBarrier 区别

- ThreadLocal 原理分析

最常见的ThreadLocal使用场景为 用来解决 数据库连接、Session管理等。

- 讲讲线程池的实现原理
- 线程池的几种方式

可缓存线程池 **newCachedThreadPool**

指定工作线程数量的线程池 **newFixedThreadPool、**

一个单线程化的Executor **newSingleThreadExecutor**

定长的线程池 支持定时的以及周期性的任务执行

newScheduleThreadPool

newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。
newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。
newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。
newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。
newWorkStealingPool创建一个拥有多个任务队列（以便减少连接数）的线程池。

- 线程的生命周期

创建 分配内存，并初始化其成员变量的值
就绪 调用了start()方法之后，该线程处于就绪状态
运行 就绪状态的线程获得了CPU，开始执行run()方法的线程执行体，则该线程处于运行状态
阻塞 被阻塞的线程会在合适的时候重新进入就绪状态
死亡

锁机制

- 说说线程安全问题
- volatile 实现原理

保证了其在多线程之间的可见性，即每次读取到volatile变量，一定是最新的数据
为了获取更好的性能JVM可能会对指令进行重排序，多线程下可能会出现一些意想不到的问题。使用volatile则会对禁止语义重排序，当然这也一定程度上降低了代码执行效率

- ThreadLocal、Volatile、synchronized、Atomic

Atomic 通过Lock-Free+原子操作指令来确定

1、循环（for(;;)、while）
2、CAS（CompareAndSet）
3、回退（return、break）

Volatile

什么是可见性？就是在一个线程的工作内存中修改了该变量的值，该变量的值立即能回显到主内存中，从而保证所有的线程看到这个变量的值是一致的

volatile不具有操作的原子性

- synchronize 实现原理

可以保证方法或者代码块在运行时，同一时刻只有一个方法可以进入到临界区，同时它还可以保证共享变量的内存可见性

重入锁ReentrantLock+一个Condition，所以说是Lock的简化版本，因为一个Lock往往可以对应多个Condition

Java对象头

Mark Word。

Mark Word用于存储对象自身的运行时数据

Klass Point是

是对象指向它的类元数据的指针，虚拟机通过这个指针来确定这个对象是哪个类的实例

monitor

同步代码块是使用monitorenter和monitorexit指令实现的

- synchronized 与 lock 的区别
- CAS 乐观锁

乐观锁就是，每次不加锁而是假设没有冲突而去完成某项操作，如果因为冲突失败就重试，直到成功为止。乐观锁用到的机制就是CAS，Compare and Swap。

- ABA 问题

CAS虽然很高效的解决原子操作，但是CAS仍然存在三大问题。ABA问题，循环时间长开销大和只能保证一个共享变量的原子操作

ABA问题。因为CAS需要在操作值的时候检查下值有没有发生变化，如果没有发生变化则更新，但是如果一个值原来是A，变成了B，又变成了A，那么使用CAS进行检查时会发现它的值没有发生变化，但是实际上却变化了。ABA问题的解决思路就是使用版本号。在变量前面追加上版本号，每次变量更新的时候把版本号加一，那么A - B - A 就会变成1A-2B - 3A。

循环时间长开销大

只能保证一个共享变量的原子操作

- 乐观锁的业务场景及实现方式

这里就回到前面说的乐观锁是有一定的不安全性的,B在检查版本的时候A还没有修改,在B检查完版本后更新数据前(例子中的输出语句),A更改了value值,这时B执行更新数据(例子中的输出语句)就发生了与现存value不一致的现象.

针对这个问题,我觉得乐观锁要解决这个问题还需要在检查版本与更新数据这个操作的时候能够使用悲观锁,比如加上synchronized,让它在最后一步保证数据的一致性.这样既保证多线程都能同时执行,牺牲最后一点的性能去保证数据的一致.

核心篇

数据存储

- MySQL 索引使用的注意事项
- 说说反模式设计
- 说说分库与分表设计
- 分库与分表带来的分布式困境与应对之策
- 说说 SQL 优化之道
- MySQL 遇到的死锁问题
- 存储引擎的 InnoDB 与 MyISAM
- 数据库索引的原理
- 为什么要用 B-tree
- 聚集索引与非聚集索引的区别
- limit 20000 加载很慢怎么解决
- 选择合适的分布式主键方案
- 选择合适的数据存储方案
- ObjectId 规则
- 聊聊 MongoDB 使用场景
- 倒排索引
- 聊聊 ElasticSearch 使用场景

缓存使用

- Redis 有哪些类型

String Set SortedSet List hash

- Redis 内部结构

http://www.cnblogs.com/jaycekon/p/6227442.html

- 聊聊 Redis 使用场景
- Redis 持久化机制

https://blog.csdn.net/tr1912/article/details/70197085?foxhandler=RssReadRenderProcessHandler

- Redis 如何实现持久化

一种是写RDB文件方式，另一种是写AOF文件，默认执行的是RDB文件持久化方

- Redis 集群方案与实现
- Redis 为什么是单线程的
- 缓存奔溃
- 缓存降级
- 使用缓存的合理性问题

消息队列

- 消息队列的使用场景
- 消息的重发补偿解决思路
- 消息的幂等性解决思路
- 消息的堆积解决思路
- 自己如何实现消息队列
- 如何保证消息的有序性

框架篇

Spring

- BeanFactory 和 ApplicationContext 有什么区别
- Spring Bean 的生命周期
- Spring IOC 如何实现
- 说说 Spring AOP
- Spring AOP 实现原理
- 动态代理（cglib 与 JDK）
- Spring 事务实现方式
- Spring 事务底层原理
- 如何自定义注解实现功能
- Spring MVC 运行流程
- Spring MVC 启动流程
- Spring 的单例实现原理
- Spring 框架中用到了哪些设计模式
- Spring 其他产品（Srping Boot、Spring Cloud、Spring Secuirity、Spring Data、Spring AMQP 等）

Netty

- 为什么选择 Netty
- 说说业务中，Netty 的使用场景
- 原生的 NIO 在 JDK 1.7 版本存在 epoll bug
- 什么是TCP 粘包/拆包
- TCP粘包/拆包的解决办法
- Netty 线程模型
- 说说 Netty 的零拷贝
- Netty 内部执行流程

- Netty 重连实现

微服务篇

微服务

- 前后端分离是如何做的
- 微服务哪些框架
- 你怎么理解 RPC 框架
- 说说 RPC 的实现原理
- 说说 Dubbo 的实现原理
- 你怎么理解 RESTful
- 说说如何设计一个良好的 API
- 如何理解 RESTful API 的幂等性
- 如何保证接口的幂等性
- 说说 CAP 定理、 BASE 理论
- 怎么考虑数据一致性问题
- 说说最终一致性的实现方案
- 你怎么看待微服务
- 微服务与 SOA 的区别
- 如何拆分服务
- 微服务如何进行数据库管理
- 如何应对微服务的链式调用异常
- 对于快速追踪与定位问题
- 微服务的安全

分布式

- 谈谈业务中使用分布式的场景
- Session 分布式方案
- 分布式锁的场景
- 分布是锁的实现方案
- 分布式事务
- 集群与负载均衡的算法与实现
- 说说分库与分表设计
- 分库与分表带来的分布式困境与应对之策

安全&性能

安全问题

- 安全要素与 STRIDE 威胁
- 防范常见的 Web 攻击
- 服务端通信安全攻防
- HTTPS 原理剖析
- HTTPS 降级攻击
- 授权与认证
- 基于角色的访问控制
- 基于数据的访问控制

性能优化

- 性能指标有哪些
- 如何发现性能瓶颈
- 性能调优的常见手段
- 说说你在项目中如何进行性能调优

工程篇

需求分析

- 你如何对需求原型进行理解和拆分
- 说说你对功能性需求的理解
- 说说你对非功能性需求的理解
- 你针对产品提出哪些交互和改进意见
- 你如何理解用户痛点

设计能力

- 说说你在项目中使用过的 UML 图
- 你如何考虑组件化
- 你如何考虑服务化
- 你如何进行领域建模
- 你如何划分领域边界
- 说说你项目中的领域建模
- 说说概要设计

设计模式

- 你项目中有使用哪些设计模式
- 说说常用开源框架中设计模式使用分析
- 说说你对设计原则的理解

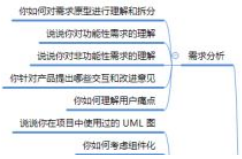
- 23种设计模式的设计理念
- 设计模式之间的异同，例如策略模式与状态模式的区别
- 设计模式之间的结合，例如策略模式+简单工厂模式的实践
- 设计模式的性能，例如单例模式哪种性能更好。

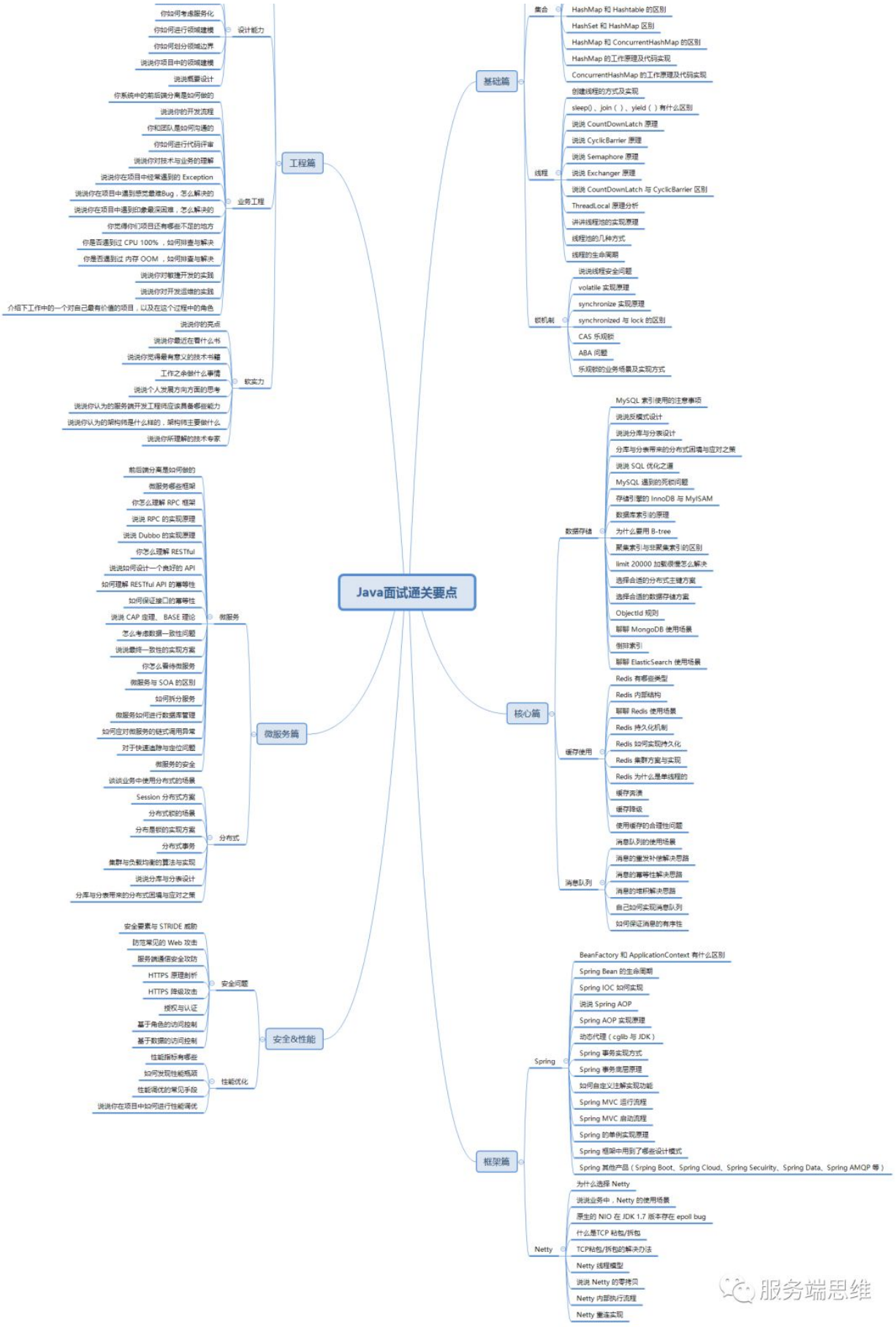
业务工程

- 你系统中的前后端分离是如何做的
- 说说你的开发流程
- 你和团队是如何沟通的
- 你如何进行代码评审
- 说说你对技术与业务的理解
- 说说你在项目中经常遇到的 Exception
- 说说你在项目中遇到感觉最难Bug，怎么解决的
- 说说你在项目中遇到印象最深困难，怎么解决的
- 你觉得你们项目还有哪些不足的地方
- 你是否遇到过 CPU 100% ，如何排查与解决
- 你是否遇到过 内存 OOM ，如何排查与解决
- 说说你对敏捷开发的实践
- 说说你对开发运维的实践
- 介绍下工作中的一个对自己最有价值的项目，以及在这个过程中的角色

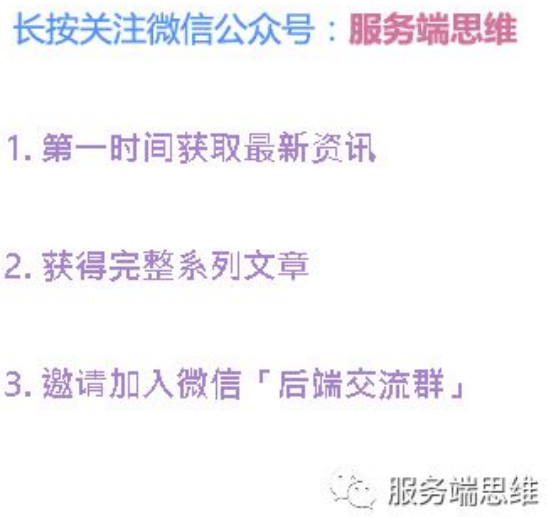
软实力

- 说说你的亮点
- 说说你最近在看什么书
- 说说你觉得最有意义的技术书籍
- 工作之余做什么事情
- 说说个人发展方向方面的思考
- 说说你认为的服务端开发工程师应该具备哪些能力
- 说说你认为的架构师是什么样的，架构师主要做什么
- 说说你所理解的技术专家





这里，特别感谢「寻找李先森」的提议与整理。



- Java面试通关要点（一）基础篇
- Java面试通关要点（二）核心篇
- Java面试通关要点（三）框架篇
- Java面试通关要点（四）微服务篇
- Java面试通关要点（五）工程篇

阅读原文