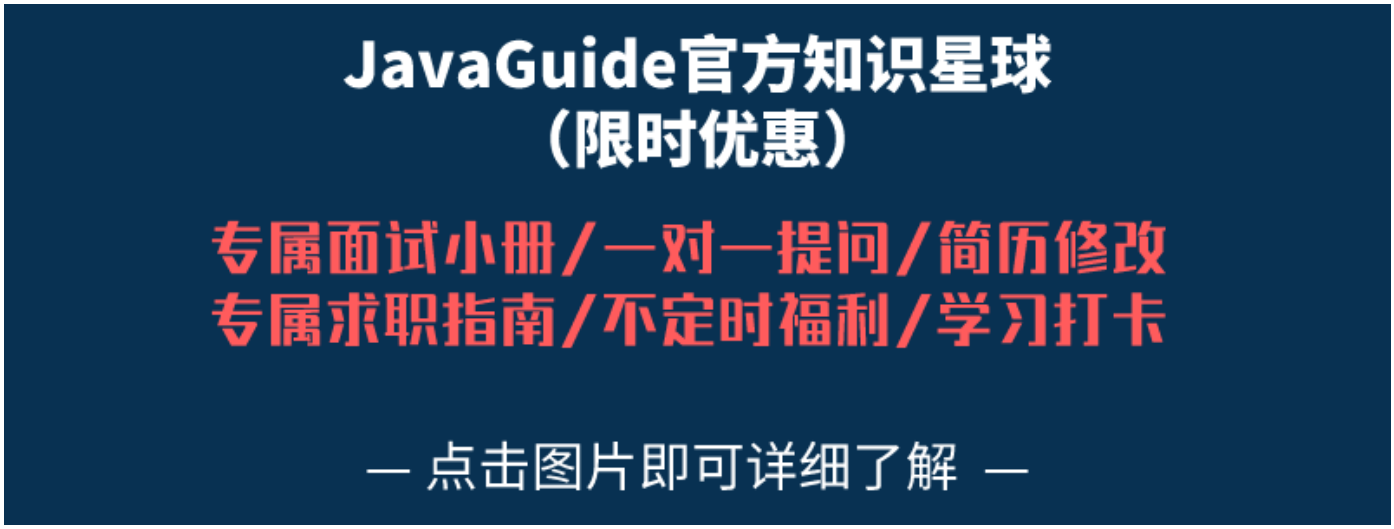
说明

本文档为 JavaGuide 官方知识星球专属，后续会在我的知识星球里持续同步更新完善！

欢迎准备 Java 面试以及学习 Java 的同学加入我的 知识星球（点击链接即可查看星球的详细介绍），干货非常多，学习氛围也很不错！收费虽然是白菜价，但星球里的内容或许比你参加上万的培训班质量还要高。



下面是星球提供的部分服务：

我的知识星球能为你提供什么？

努力做一个最优质的 Java 面试交流星球！加入到我的星球之后，你将获得：

1. 6 个高质量的专栏永久阅读，内容涵盖面试，源码解析，项目实战等内容！价值远超门票！
2. 多本原创 PDF 版本面试手册。
3. 免费的简历修改服务（已经累计帮助 4000+ 位球友修改简历）。
4. 一对一免费提问交流（专属建议，走心回答）。
5. 专属求职指南和建议，帮助你逆袭大厂！
6. 海量 Java 优质面试资源分享！价值远超门票！
7. 读书交流，学习交流，让我们一起努力创造一个纯粹的学习交流社区。
8. 不定期福利：节日抽奖、送书送课、球友线下聚会等等。
9. ……

其中的任何一项服务单独拎出来价值都远超星球门票了。

JavaGuide 官方网站地址：**javaguide.cn** 。

前言

这篇文章对之前总结分享的线上常见问题案例（https://t.zsxq.com/0dsAac47U）进行了补充，可以作为参考，用来准备面试中类似下面的提问：

- 你在做这个项目的时候遇到了什么问题？（OOM 问题、GC 问题等等）
- 你用过哪些分析定位 Java 故障/性能的工具？（JDK 自带工具、MAT、Arthas 等等）
- 如果项目遇到了 OOM 问题，你会如何排查？（常用 MAT）
- 有什么办法可以监控到 JVM 的实时运行状态？（Arthas）
- 生产环境有一个接口很慢，如何排查？（Arthas）
- CPU100%，你是怎么处理的？（jstack 或者 Arthas）
- 你是如何定位线上问题的？（说说自己了解的工具，然后根据面试官提示继续深入聊即可）
- 项目中遇到了哪些线上故障，你是如何解决的？
- ……

Java

常用工具

《Java 业务开发常见错误 100 例》极客时间专栏中的两篇文章，介绍了分析定位 Java 问题常用的一些工具比如 JDK 自带工具、 JVM 堆转储的工具 MAT（Memory Analyzer）、全能的故障诊断工具 Arthas：

- 分析定位 Java 问题，一定要用好这些工具（一） - 《Java 业务开发常见错误 100 例》
- 分析定位 Java 问题，一定要用好这些工具（二） - 《Java 业务开发常见错误 100 例》

上面这两篇写的非常不错，另外继续补充分享一些相关的文章。

Arthas

- Arthas 官方文档（中文，很详细。火焰图这块面试常问，）
- 如何使用 Arthas 定位 Java 性能问题 - CodingBetterLife - 2023
- 如何使用 Arthas 提高日常开发效率？ - 阿里开发者 - 2021

如果面试中聊到 Arthas 的话，面试官很可能会继续追问你 Arthas 的底层实现原理。

MAT

- JVM 内存分析工具 MAT 的深度讲解与实践—入门篇
- JVM 内存分析工具 MAT 的深度讲解与实践—进阶篇

JDK 自带工具

- JDK 监控和故障处理工具总结
- 6 个 Java 工具，轻松分析定位 JVM 问题！（就是上面推荐的《Java 业务开发常见错误 100 例》其中一篇文章的内容）

基础

不规范的枚举类代码引发的一场事故 - 阿里云开发者 - 2023

- **现象**：对象的状态属性（枚举类型）在插入数据库后丢失，变成了空白字符串。
- **分析**：通过 Arthas `watch` 命令发现 `PayRequest` 对象在转换过程中 `status` 属性丢失。
- **原因**：直接原因：枚举类 `Status` 提供了 `setter` 方法，违反了枚举使用规范。根本原因：第三方库 `podam` 在解析枚举类时，错误地通过反射调用了 `setter` 方法。
- **建议**：首要的是写代码还是要注意规范，最好本地装一些扫描工具，例如 sonar，发现风险一定要尽快按照建议修复。

一行 MD5 居然让小伙伴都回不了家！ - 京东云开发者 - 2023

- **现象**：测试环境中，首次请求超时，后续请求迅速返回。
- **分析**：使用 Arthas `trace` 命令查看方法耗时。

- **原因**：Hutool 自带的 MD5 算法使用存在问题（第一次加载很慢），使用 Google 提供的 MD5 算法成功解决问题。
- **建议**：使用第三方包的时候要小心，尤其是没有大公司背书的第三方包。

多线程

- 一次棘手的线程池使用不当导致的生产问题 - why 技术 - 2022（分析问题的思路超赞）：拒绝策略使用不当，由严选技术的一个线程池拒绝策略引发的血案这篇文章引申而来。
- 线程池运用不当的一次线上事故 - IT 人的职场进阶 - 2021：父任务和子任务共用一个线程池导致的死锁问题。
- Java 线程池最佳实践 - JavaGuide - 2022 ： 我总结的一篇线程池使用的一些优秀实践比如建议不同类别的业务用不同的线程池。
- 记一次因@Async 引发的程序 bug - Linyb 极客之路 - 2022：`Controller` 加了 `@Asyn` 异步就失效了，不会被 MVC 进行映射处理，这样就导致了 404 的问题。推荐的解决方法是将 `@Async` 逻辑抽离出 `Controller`，新建一个 `Service` 类进行处理。
- 详解一次由读写锁引起的内存泄漏 - Coder 的技术之路 -2021：为了实现 LRU 功能，通过重写 `LinkedHashMap` 的 `get()` 方法，在 get 之后，将当前节点移动到链表最后。因此，即使是 get 操作，其实际依然是一个写操作。所以，不应该使用读写锁，而应该使用互斥锁。还是更推荐的还是使用分布式缓存或者 Guava 缓存、Caffeine 这些比较完善的本地缓存的实现。

常见生产问题

CPU 使用率高（重要）

相关面试题：

1. CPU100%，你是怎么处理的?
2. 线上服务 CPU100% 问题，如何快速定位?
3. CPU 占用情况如何分析?

解决 CPU 使用率高的常用工具：

1. jvm 自带的工具 jstack：使用 `top` 命令定位高 CPU 占用线程，再使用 `jstack` 命令查看线程栈信息。
2. 阿里开源的全能的故障诊断工具 Arthas（更推荐）：
    - `dashboard` 命令查看 TOP N 线程，`thread` 命令查看堆栈信息。
    - 内部集成的火焰图工具 async-profiler

利用 jstack 解决线上服务 CPU 100% - HeapDump - 2021

- **现象**：钉钉告警：CPU 超过告警阈值，达 100%；客户反馈：系统登录失败，报 504 错误。
- **分析**：使用 `top -Hp <进程ID>` 定位高 CPU 占用线程，再使用 `jstack <进程ID> > jstack.txt` 查看线程栈信息
- **建议**：对于此类耗资源的操作，一定要做好相应的限制。比如可以限制请求量，控制最大分页大小，同时可以限制访问频率，比如同一用户一分钟内最多请求多少次。
- **资料**：字节面试：CPU被打满/CPU100%，如何处理？ - 技术自由圈

用 Arthas 解决线上正则表达式导致 CPU100%的问题 - 掘金 - 2023

- **现象**：一个服务出现了 cpu 达到 100%的问题。
- **分析**：使用 arthas 查看线程信息，输入 `thread -n 10`，列出 10 个占用 cpu 最高的线程堆栈信息。

不经意的两行代码把 CPU 使用率干到了 90%+ - HeapDump - 2023

- **现象**：应用使用 schedulerx 执行定时任务，每小时执行一次，每次约 5 分钟，执行任务期间 CPU 使用率 90%+。
- **分析**：Arthas 内部集成的火焰图工具 async-profiler 比较适用 CPU 使用率持续较高的场景。通过对热点火焰图的分析，`NoSuchMethodException` 异常相关代码占用了很多 CPU 时间。为了更准确的定位相关业务代码，我们需要知道抛出 `NoSuchMethodException` 的线程栈，可以使用 Arthas stack，从线程栈我们可以知道在【哪个类哪个方法哪行】发出的调用。

OOM/内存泄露（重要）

相关面试题：

1. 如果项目遇到了 OOM 问题，你会如何排查?
2. 你遇到过 OOM 问题吗？最后怎么解决的?

解决 OOM/内存泄露的常用工具：

1. JDK 自带的 VisualVM：使用 `jmap` 命令生成堆转储快照也就是 dump 文件，然后通过 VisualVM 分析 dump 文件。
2. MAT (Memory Analyzer Tool)：MAT 也可以分析 dump 文件。

阿里云开发者发表过一篇三万字长文：JVM内存问题排查Cookbook - 2024，堪称应用内存问题排查保姆级指南。不过，这篇写的过于详细，实际面试中是需要介绍这么详细的，直接针对自己项目遇到的问题去介绍即可。

一次线上 OOM 问题分析 - 艾小仙 - 2023

- **现象**：线上某个服务有接口非常慢，通过监控链路查看发现，中间的 GAP 时间非常大，实际接口并没有消耗很多时间，并且在那段时间里有很多这样的请求。
- **分析**：使用 JDK 自带的 `jvisualvm` 分析 dump 文件。
- **建议**：对于 SQL 语句，如果监测到没有 `where` 条件的全表查询应该默认增加一个合适的 `limit` 作为限制，防止这种问题拖垮整个系统
- **资料**：实战案例：记一次 dump 文件分析历程转载 - HeapDump - 2022。

生产事故-记一次特殊的 OOM 排查 - 程语有云 - 2023

- **现象**：网络没有问题的情况下，系统某开放接口从 2023 年 3 月 10 日 14 时许开始无法访问和使用。
- **临时解决办法**：紧急回滚至上一稳定版本。
- **分析**：使用 MAT (Memory Analyzer Tool)工具分析 dump 文件。
- **建议**：正常情况下， `-Xmn` 参数（控制 Young 区的大小）总是应当小于 `-Xmx` 参数（控制堆内存的最大大小），否则就会触发 OOM 错误。

- **资料**：最重要的 JVM 参数总结 - JavaGuide - 2023

一次大量 JVM Native 内存泄露的排查分析（64M 问题） - 掘金 - 2022

- **现象**：线上项目刚启动完使用 top 命令查看 RES 占用了超过 1.5G。
- **分析**：整个分析流程用到了较多工作，可以跟着作者思路一步一步来，值得学习借鉴。
- **建议**：远离 Hibernate。
- **资料**：Linux top 命令里的内存相关字段（VIRT, RES, SHR, CODE, DATA）

GC 问题

记一次 CMS GC 耗时 46.6 秒的 排查与解决过程 - 掘金 - 2023

- **现象**：HTTP 接口请求延时过高。
- **分析**：使用 MAT (Memory Analyzer Tool) 工具分析 dump 文件（老套路了）。
- **建议**：监控工具非常重要，SQL 语句的编写需要谨慎。

YGC 问题排查，又让我涨姿势了！ - IT 人的职场进阶 - 2021

- **现象**：广告服务在新版本上线后，收到了大量的服务超时告警。
- **分析**：使用 MAT (Memory Analyzer Tool) 工具分析 dump 文件。
- **建议**：学会 YGC（Young GC） 问题的排查思路，掌握 YGC 的相关知识点。

听说 JVM 性能优化很难？今天我小试了一把！ - 陈树义 - 2021

通过观察 GC 频率和停顿时间，来进行 JVM 内存空间调整，使其达到最合理的状态。调整过程记得小步快跑，避免内存剧烈波动影响线上服务。 这其实是最为简单的一种 JVM 性能调优方式了，可以算是粗调吧。

你们要的线上 GC 问题案例来啦 - 编了个程 - 2021

- **案例 1**：使用 guava cache 的时候，没有设置最大缓存数量和弱引用，导致频繁触发 Young GC

- **案例 2**： 对于一个查询和排序分页的 SQL，同时这个 SQL 需要 join 多张表，在分库分表下，直接调用 SQL 性能很差。于是，查单表，再在内存排序分页，用了一个 List 来保存数据，而有些数据量大，造成了这个现象。

Java 中 9 种常见的 CMS GC 问题分析与解决 - 美团技术团 - 2020

这篇文章共 2w+ 字，详细介绍了 GC 基础，总结了 CMS GC 的一些常见问题分析与解决办法。

数据库

MySQL

死锁问题（重要）

手把手教你分析解决 MySQL 死锁问题 - 后端元宇宙 - 2024

生产环境中，MySQL 死锁还是挺常见的，面试被提问的概率也较大。这篇文章详细介绍了 MySQL 死锁问题该如何排查和解决。

解决 MySQL 死锁问题的思路：

1. 使用 `show engine innodb status;` 查看死锁日志，然后分析死锁日志。
2. 如果第一种方式无法解决死锁问题，可以通过 binlog 日志获取锁事务所执行的全部 SQL。有了具体 SQL 语句，就能进行具体的锁冲突分析了。

类似的文章：

- MySQL 死锁系列-线上死锁问题排查思路 - 程序员历小冰 - 2020
- MySQL死锁是什么，如何解决？ - 技术自由圈 - 2024

时区问题

一个诡异的 MySQL 查询超时问题，居然隐藏着存在了两年的 BUG - CoderW 喜欢写博客 - 2021

1. 时区没对上： SQL 日志记录里面的时区都是标准时区，任务执行的时候是北京时间，标准时区和北京时区是差了 8 个小时。
2. 底层在取时间区间时，调了一个 RPC 接口，这个接口预期返回的时间区间只有几天，结果返回了七个月的时间区间。查询的日期区间从 2020 年 9 月到 2021 年 4 月，时间跨度 7 个月。MySQL 成本计算的时候认为区间太大，走索引还不如直接扫描全表，最终没有走索引扫描了 1800W 条数据。

慢查询

慢查询引发的车祸现场，案例分析！ - 月伴飞鱼 - 2021

分享了几个导致慢查询的案例。

隐式转换

一个 MySQL 隐式转换的坑，差点把服务器整崩溃了 - 古时的风筝 - 2022

我们在平时的开发过程中，尽量要避免隐式转换，因为一旦发生隐式转换除了会降低性能外， 还有很大可能会出现不期望的结果。

事务隔离级别错误

MySQL 可重复读，差点就让我背上了一个 P0 事故！ - 楼下小黑哥 - 2020

真实事件，交易系统，p0 事故：余额多扣！

Redis

一次 Redis 主从切换导致的数据丢失与陷入只读状态故障

Redis 主从切换导致的数据丢失与集群陷入只读状态故障。事故最后的原因很简单，运行配置和静态配置不一致导致的。修改配置时，切记分析清楚，到底是要永久性的，还是一次性的。

Redis——由分布式锁造成的重大事故 - 浪漫先生 - 2020

分布式锁使用不当导致超卖。

消息队列

Kafka

一次 Kafka 消息堆积问题排查 - 张乘辉 - 2020

项目中某 Kafka 消息组消费特别慢，有时候在 kafka-manager 控制台看到有些消费者已被踢出消费组。该消费组在短时间内重平衡了 600 多次。

RabbitMQ

RabbitMQ 消息延迟和堆积实战 - 菜农曰 - 2021

- 消息延迟： TTL + 死信队列（比较麻烦）、RabbitMQ 延迟队列插件（更简单，相关阅读：RabbitMQ 延迟插件的使用）
- 消息堆积解决：增加消费者、多线程、扩大队列的容量、惰性队列（更灵活但消息的时效性降低，接收到消息后直接存入磁盘而非内存，支持百万级消息的存储）

网络

万字长文让你掌握网络问题排查技巧！ - 安琪拉的博客 - 2021

网络问题很考验工程师解决问题的功力，这类问题需要对 TCP 的原理，抓包工具有比较深入的理解和运用，这篇文章以实战角度为你讲述了网络问题的排查思路，非常值得一看！