美团二面面经，最后竟然有惊喜？

面试现场

面试官：**首先你来讲讲进程和线程有什么区别？**

独白：老八股文了哈哈

大彬：进程是系统进行资源分配和调度的独立单位，每一个进程都有自己的内存空间和系统资源

大彬：线程是进程的一个实体，是CPU调度和分派的基本单位，它是比进程更小的能独立运行的基本单位

大彬：多线程是实现并发机制的一个有效手段。进程和线程一样都是实现并发的基本单位

面试官：**那为什么要用多线程呢？**

独白：嘿嘿，这个简单

大彬：使用多线程最主要的原因是提高系统的资源利用率。

大彬：多个线程同时运行，可以减少线程上下文切换的开销，提高并发的能力和CPU的利用效率。

大彬：在平时工作中多线程也是常见的。比如Tomcat每处理一个请求都会从线程连接池里取一个线程去处理。

面试官：**嗯，平时在使用多线程的时候，可能会遇到线程安全的问题吧。讲讲什么是线程安全？**

大彬：我是这么理解的，当多个线程访问一个对象时，如果不用考虑这些线程在运行时环境下的调度和交替执行，也不需要进行额外的同步，调用这个对象的行为都可以获得正确的结果，那这个对象就是线程安全的。

面试官：**那你平时怎么处理线程安全问题的？**

大彬：这个还得具体问题具体分析。首先判断有没有线程安全问题，若有则根据具体的情况去处理线程安全的问题。

大彬：比如涉及到操作的原子性，可以考虑使用 `atomic` 包下的原子类。

大彬：如果涉及到对线程的控制，可以考虑线程工具类 `CountDownLatch` / `Semaphore` 等等。

大彬：集合类的话，考虑 `java.util.concurrent` 包下的集合类。

大彬：还有 `synchronized` 和 `lock` 包下的类，redis分布式锁等。

面试官：**嗯哼，刚提到Redis分布式锁，你觉得什么场景下需要使用分布式锁呢?**

大彬：在单机环境下，线程安全问题可以通过 `ReentrantLock` 、 `synchronized` 以及 `concurrent` 并发包下一些线程安全的类等来避免。

大彬：而在多机部署环境，需要在多进程下保证线程的安全性，Java提供的这些API仅能保证在单个JVM进程内对多线程访问共享资源的线程安全，已经不满足需求了。这时候就需要使用分布式锁来保证线程安全。

大彬：Redis 2.6.12 之前的版本中采用 `setnx + expire` 方式实现分布式锁。在 Redis 2.6.12 版本后 `setnx` 增加了过期时间参数，只需要使用 `setnx` 就可以实现分布式锁了。

面试官：**那再讲讲Redis分布式锁的原理?**

独白：面试造火箭，入职拧螺丝?

大彬：首先介绍下Redis的加锁逻辑。

大彬： `setnx` 争抢key的锁，如果已有key存在，则不作操作，过段时间继续重试，保证只有一个客户端能持有锁。

大彬：value设置为 `requestId` （可以使用机器ip拼接当前线程名称），表示这把锁是哪个请求加的，在解锁的时候需要判断当前请求是否持有锁，防止误解锁。比如客户端A加锁，在执行解锁之前，锁过期了，此时客户端B尝试加锁成功，然后客户端A再执行 `del()` 方法，则将客户端B的锁给解除了。

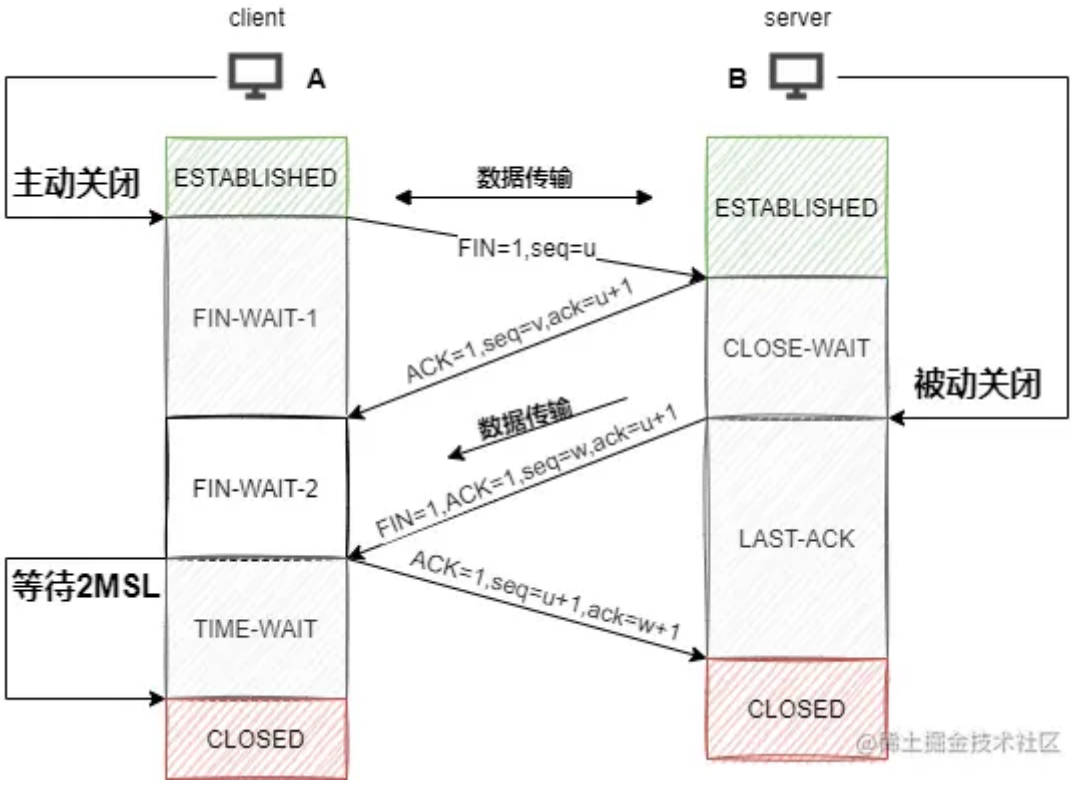
大彬：再用 `expire` 给锁加一个过期时间，防止异常导致锁没有释放。

大彬：然后是解锁逻辑。

大彬：首先获取锁对应的value值，检查是否与 `requestId` 相等，如果相等则删除锁。使用lua脚本实现原子操作，保证线程安全。

面试官：**不错，看你简历上写了熟悉TCP，来介绍下TCP四次挥手?**

独白：嗯，这个嘛，很熟悉



大彬：假设A是client端，B是server端。

0. 首先A的应用进程先向其TCP发出连接释放报文段（ `FIN=1，seq=u` ），并停止再发送数据，主动关闭TCP连接，进入 `FIN-WAIT-1` （终止等待1）状态，等待B的确认。
1. B收到连接释放报文段后即发出确认报文段（ `ACK=1，ack=u+1，seq=v` ），B进入 `CLOSE-WAIT` （关闭等待）状态，此时的TCP处于半关闭状态，A到B的连接释放。
2. A收到B的确认后，进入 `FIN-WAIT-2` （终止等待2）状态，等待B发出的连接释放报文段。
3. B发送完数据，就会发出连接释放报文段（ `FIN=1，ACK=1，seq=w，ack=u+1` ），B进入 `LAST-ACK` （最后确认）状态，等待A的确认。
4. A收到B的连接释放报文段后，对此发出确认报文段（ `ACK=1，seq=u+1，ack=w+1` ），A进入 `TIME-WAIT` （时间等待）状态。此时TCP未释放掉，需要经过时间等待计时器设置的时间 `2MSL` （最大报文段生存时间）后，A才进入 `CLOSED` 状态。B收到A发出的确认报文段后关闭连接，若没收到A发出的确认报文段，B就会重传连接释放报文段。

面试官：**建立连接时三次握手，为什么连接释放要四次挥手，三次不行吗?**

大彬：因为建立连接时，当Server端收到Client端的 `SYN` 连接请求报文后，可以直接发送 `SYN+ACK` 报文。

大彬：**但是在关闭连接时，当Server端收到Client端发出的连接释放报文时，很可能并不会立即关闭SOCKET**，所以Server端先回复一个 `ACK` 报文，告诉Client端我收到你的连接释放报文了。只有等到Server端所有的报文都发送完了，这时Server端才能发送连接释放报文，之后两边才会真正的断开连接。故需要四次挥手。

面试官：**嗯，你了解https吗？https是为了解决什么问题？**

独白：一点也不慌哈哈

大彬：HTTP是明文传输，容易被黑客窃听或篡改，不安全。

大彬： HTTPS 主要解决了 HTTP 明文协议的缺陷，在 HTTP 的基础上加入 SSL/TLS 协议，依靠 SSL 证书来验证服务器的身份，为客户端和服务器端之间建立SSL通道，确保数据传输安全。

面试官：那http跟https具体有什么区别呢？

大彬：http和https的区别如下：

0. HTTP是超文本传输协议，信息是**明文传输**；HTTPS则是具有**安全性**的ssl加密传输协议。
1. HTTP和HTTPS用的端口不一样，HTTP端口是80，HTTPS是443。
2. HTTPS协议**需要到CA机构申请证书**，一般需要一定的费用。
3. HTTP运行在TCP协议之上；HTTPS运行在SSL协议之上，SSL运行在TCP协议之上。

面试官：**不错，再来问点MySQL相关的**

面试官：**什么情况下索引会失效？**

大彬：主要有这么几种情况会导致索引失效。

- 对于组合索引，不是使用组合索引最左边的字段，则不会使用索引
- 以%开头的like查询如 `%abc` ，无法使用索引；非%开头的like查询如 `abc%` ，相当于范围查询，会使用索引
- 查询条件中列类型是字符串，没有使用引号，可能会因为类型不同发生隐式转换，使索引失效
- 判断索引列是否不等于某个值时

- 对索引列进行运算
- 查询条件使用 `or` 连接，也会导致索引失效

面试官：很好，明天能入职吗？

独白：马甲头盔箱子三件套？