

## 美团面经

### 1、Spring : 有没有用过 Spring , Spring IOC、AOP 机制与实现 , Spring MVC

其实我挺不想被问到 Spring 的细节的，框架这些我都没有复习不太记得了。所以我对面试官说 Spring 里面的一些比较重要的机制我理解的还不错，然后我用一个实际的例子把我对 IOC、AOP 理解讲了一下，他听了说对，理解的不错（难得遇到一个边面试边能给反馈的面试官，好开心）。

Spring MVC 其实我用过，我就对面试官讲了我的项目中用到的 Servlet，jsp 和 javabean 实现的 MVC，以及 MVC 各个模块职责以及每个模块是怎么联系到一起的，最后我补充了一句我想 SpringMVC 的思想其实跟这个是一样的（他说对的，嘿嘿有反馈真好）。

### 2、多线程：怎么实现线程安全，各个实现方法有什么区别，volatile 关键字的使用，可重入锁的理解，Synchronized 是不是可重入锁

这里我就主要讲了 Synchronized 关键字，还有并发包下面的一些锁，以及各自的优缺点和区别。volatile 关键字我主要从可见性、原子性和禁止 JVM 指令重排序三个方面讲的，再讲了一下我在多线程的单例模式 double-check 中用到 volatile 关键字禁止 JVM 指令重排优化。

### 3、集合：HashMap 底层实现，怎么实现 HashMap 线程安全

我讲了一下 HashMap 底层是数组加单链表实现，Node 内部类，add 的过程，Hash 冲突解决办法，扩容，三种集合视图。HashMap 线程安全的实现方式主要讲了 Hashtable、ConcurrentHashMap 以及 Collections 中的静态方法 SynchronizedMap 可以对 HashMap 进行封装。以及这三种方式的区别，效率表现。

#### **4、JVM 内存管理，GC 算法，HotSpot 里面的垃圾回收器、类加载**

JVM 内存主要分为五个区，哪些是线程共享的，哪些是线程独享的，每个区存放什么。

GC 方面：怎么判断哪些对象需要被 GC，GC 的方法，Minor GC 与 Full GC。HotSpot GC 算法以及 7 种垃圾回收期，主要讲了 CMS 和 G1 收集器。类加载：类加载的过程，Bootstrap classloader-ExtClassLoader-AppClassLoader，父类委托机制。

#### **5、进程和线程的区别**

从调度、并发性、拥有的资源和系统开销四个方面回答的。

#### **6、HTTP 有没有状态，我说无状态，怎么解决 HTTP 无状态**

怎么解决 HTTP 无状态其实就是怎么进行会话跟踪，有四种方法：URL 重写、隐藏表单域、Cookie、Session。

#### **7、Java IO，NIO，Java 中有没有实现异步 IO**

Java IO 实现的是同步阻塞，它是怎么实现同步阻塞的。我拿了 read()方法举例来讲的。

NIO 实现的是同步非阻塞，我详细讲了一下 Selector 中的 select()方法轮询说明它是如何实现多路复用 IO 的。然后对比了一下他们的效率。面试官可能看我对这一块比较了解，

又继续问我 Java 中有没有实现异步 IO，我感觉好像没有，但面试官说有，让我想想，其实这里我并不清楚啦，所以我就对面试官讲了一下我对 Unix 中异步 IO 模型的理解，然后说至于 Java 里面有没有我真的不太清楚。（他居然笑了！说你理解是对的，Java 里面有没有不重要！哈哈）

## 8、前端会不会，Ajax 是什么，Ajax 实现原理

前端我只是会用一些 js 而已，用过 jquery 框架，问我 Ajax 全称是啥，我猜是异步的 js 和 xml。Ajax 实现原理其实我也不懂，我就只简单讲了一下它通过 XMLHttpRequest 对象进行异步查询，Ajax 引擎在客户端运行，减少了服务器工作量。

## 9、让我设计一个线程池

因为我简历中有写到我多线程、并发这一块理解比较好。所以他老问这方面的题。这个问题因为我之前看过 ThreadPoolExecutor 的源代码，所以我就仿照那个类的设计思路来想的，详细讲了一下核心池、创建线程可以用工厂方法模式来进行设计、线程池状态、阻塞队列、拒绝策略这几个方面。设计的还算比较周全。

## 10、讲几个设计模式，哪些地方用到了，为什么要用

单例模，jdk 中的 getRuntime()；工厂方法模式，ThreadPoolExcutor 用到 ThreadFactory；观察者模式：java.util 包下面的 Observable 和 Observer。最后主要讲了一下工厂方法模式的使用场景。