JVM 的内存结构

根据 JVM 规范，JVM 内存共分为虚拟机栈、堆、方法区、程序计数器、本地方法栈五个部分。

1、Java 虚拟机栈：

线程私有；每个方法在执行的时候会创建一个栈帧，存储了局部变量表，操作数栈，动态连接，方法返回地址等；每个方法从调用到执行完毕，对应一个栈帧在虚拟机栈中的入栈和出栈。

2、堆：

线程共享；被所有线程共享的一块内存区域，在虚拟机启动时创建，用于存放对象实例。

3、方法区：

线程共享；被所有线程共享的一块内存区域；用于存储已被虚拟机加载的类信息，常量，静态变量等。

4、程序计数器：

线程私有；是当前线程所执行的字节码的行号指示器，每条线程都要有一个独立的程序计数器，这类内存也称为“线程私有”的内存。

5、本地方法栈：

线程私有；主要为虚拟机使用到的 Native 方法服务。

强引用，软引用和弱引用的区别

强引用：

只有这个引用被释放之后，对象才会被释放掉，只要引用存在，垃圾回收器永远不会回收，这是最常见的 New 出来的对象。

软引用：

内存溢出之前通过代码回收的引用。软引用主要用户实现类似缓存的功能，在内存足够的情况下直接通过软引用取值，无需从繁忙的真实来源查询数据，提升速度；当内存不足时，自动删除这部分缓存数据，从真正的来源查询这些数据。

弱引用：

第二次垃圾回收时回收的引用，短时间内通过弱引用取对应的数据，可以取到，当执行过第二次垃圾回收时，将返回 null。弱引用主要用于监控对象是否已经

被垃圾回收器标记为即将回收的垃圾，可以通过弱引用的 isEnQueued 方法返回对象是否被垃圾回收器标记。

数组在内存中如何分配

1、简单的值类型的数组，每个数组成员是一个引用（指针），引用到栈上的空间（因为值类型变量的内存分配在栈上）
2、引用类型，类类型的数组，每个数组成员仍是一个引用（指针），引用到堆上的空间（因为类的实例的内存分配在堆上）

springmvc 的核心是什么，请求的流程是怎么处理的，控制反转怎么实现的

核心：
控制反转和面向切面
请求处理流程：
1、首先用户发送请求到前端控制器，前端控制器根据请求信息（如 URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即以前的控制器的控制逻辑部分；
2、页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象，并进行验证，然后将命令对象委托给业务对象进行处理；处理完毕后返回一个 ModelAndView（模型数据和逻辑视图名）；
3、前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染；
4、前端控制器再次收回控制权，将响应返回给用户。

控制反转如何实现：
我们每次使用 spring 框架都要配置 xml 文件，这个 xml 配置了 bean 的 id 和 class。
spring 中默认的 bean 为单实例模式，通过 bean 的 class 引用反射机制可以创建这个实例。
因此，spring 框架通过反射替我们创建好了实例并且替我们维护他们。
A 需要引用 B 类，spring 框架就会通过 xml 把 B 实例的引用传给了 A 的成员变量。

mybatis 如何处理结果集

MyBatis 的结果集是通过反射来实现的。并不是通过 get/set 方法。在实体类中无论是否定义 get/set()方法，都是可以接收到的。

如果面试只是考你这个点的话就恭喜了。如果继续深问流程，那就需要自己找一些源码来阅读了。

java 的多态表现在哪里

主要有两种表现形式：重载和重写

重载：

是发生在同一类中，具有相同的方法名，主要是看参数的个数，类型，顺序不同实现方法的重载的，返回值的类型可以不同。

重写：

是发生在两个类中（父类和子类），具有相同的方法名，主要看方法中参数，个数，类型必须相同，返回值的类型必须相同。