你管这叫Java反射?

大家好，我是徒手敲代码。

今天来介绍一下Java中的反射，以在面试中如何应对。

1、什么是反射?

人如果想要看到自己穿的衣服、弄的发型，一般要通过照镜子的方式。同时，一个人还可以根据自己的镜像，来做出调整衣服、化妆等一系列的动作。

而对于程序来说，反射机制就为程序提供了一面镜子，让它知道某个类的类型和构造参数等信息。同样道理，程序也可以根据这些信息，进行类及其对象的操作和管理，比如：创建对象、调用方法、修改字段值等。

一般来说，在使用某个类的时候，程序在编译阶段，就能够知道这个类的类型以及构造参数。比如：

```
Person person = new Person(”Gary”, 20);
```

这行代码中，我们指定创建的person对象，名字为Gary，年龄为20。而编译器会检查这些信息是否正确，比如：是否存在这个类是否存在、构造参数的数量和类型等，确保程序在运行的时候，能够正常地创建出这个对象。

如果使用反射来创建对象，程序要在**运行**的时候，才能知道类名、构造函数及其入参。比如：

```
// 类名作为字符串存储
String className = ”com.example.Person”;
// 根据类名字符串动态获取 Class 对象
```

```
Class<?> clazz = Class.forName(className);
// 获取特定参数类型的构造函数
Constructor<?> constructor = clazz.getConstructor(String.class, int.class);
// 使用构造函数动态创建对象
Object personInstance = constructor.newInstance(”Gary”, 20);
```

直接new出来创建对象的方式，属于静态加载类；而使用反射来创建对象，属于动态加载类；

2、优缺点

优点：代码可以在不依赖具体类型的情况下操作对象，这样可以减少代码间的紧耦合，有利于模块化设计和代码重用。

缺点：把原本编译时候干的事情，留到运行时才干，会增加运行时的开销，造成性能损耗；同时反射会绕过访问修饰符的限制，直接访问类的私有成员，使用私有方法，可能造成安全问题。

3、应用

回到实战当中，反射的应用处处可见，常见的应用如下：

第一、在Spring的应用。最明显的就是**依赖注入**和**AOP**。

对于依赖注入，Spring容器在启动的过程中，通过扫描配置文件（xml、注解、配置类），解析Bean的定义信息。利用反射来创建Bean的实例，设置属性值，注入依赖关系，使得组件之间的依赖关系在运行时动态建立，提高代码的松耦合度和可测试性。

对于AOP，Spring是通过动态代理的方式来实现的。对于目标类**有接口**的情况，Spring会使用 Proxy 这个类，来动态创建代理对象，并在指定的方法前后，执行相应的操作，这些操作可以由程序员来指定，比如事务控制、日志打印等。如果**没接口**，Spring会用 cglib 来创建代理对象。

第二、常见的ORM框架，比如MyBatis、Hibernate，在处理对象和数据库表之间的映射关系时，会利用反射来读取实体类的注解、字段信息，自动生成

SQL语句，执行数据库操作。

4、面试

在面试当中，主要考察反射的概念、优缺点以及应用场景。具体的回答参考如下：

反射：允许程序运行时，通过Reflection API得到一个类的内部信息，包括成员变量、成员方法、构造器。加载完这个类之后，在堆中会有一个Class类型的对象，这个对象包含了类的完整结构，因为这个对象就像一面镜子，通过它可以看到类的完整结构，所以这个机制被称为反射。

SpringBoot 里面，就有很多使用反射的场景。例如：通过反射机制，SpringBoot 会扫描加了 @Service @Mapper @Configuration @Component 注解的类，将这些类添加进 SpringIOC 容器进行管理。对于使用 @AutoWired 的对象，会根据这个对象的类型，在IOC容器中找到并注入这个类；对于使用@Resource的对象，会根据这个对象的类的名字，在 IOC 容器中找到并注入这个类。对于使用 @Value的变量，会直接从配置文件中获取，如果要对这些变量的值修改，可以直接修改配置文件，而不用去修改源码。结合类似 nacos这样的配置中心使用，可以更加方便地对已经上线的系统作配置上的修改，而不至于修改源码之后要重新打包部署。

利用反射的机制，可以实现 java 代码的动态加载，意思是运行时才加载需要的类，如果运行时没有用到这个类，即使缺少这个类也不会报错，降低代码的依赖性。而静态加载，编译时就把类加载到内存，如果没有这个类就会报错，依赖性很强。 但是反射会绕过访问修饰符的限制，直接访问类的私有成员，使用私有方法，可以造成安全问题。同时反射也会增加运行时的开销，因为它设计到了动态类型检查等操作。

今天的分享到这里结束了，如果你喜欢这种讲解知识的方式，可以在下方留言喔。你的支持，是我创作的最大动力！

公众号“徒手敲代码”，让我们在技术的星辰大海、生活的诗与远方中共勉同行，一起书写属于我们的精彩篇章！

原文链接: https://juejin.cn/post/7355815508279459852