一、怎么理解AOP（Aspect Orient Programming，面向切面编程）

1、动态代理的实现

JDK动态代理：使用JDK中的Proxy、Method、InvocationHandler类或接口创建代理对象，该方式要求目标类必须是接口的实现类

CGLIB动态代理：是使用第三方该工具库，创建代理对象，其原理是继承。通过继承目标类，创建子类，在子类中完成功能增强。要求就是目标类和目标方法不能是final修饰的。

动态代理的作用：

1、在目标类源代码不改变的情况下，增加功能

2、减少代码的重复

3、只需要专注业务逻辑代码

4、解耦合，让业务功能和日志、提交事务等非业务功能分离

2、AOP（面向切面编程）

面向切面编程，就是基于动态代理的，可以使用JDK和CGLIB的方式。

因为动态代理是一种很灵活的技术，不同的开发人员使用都有其独特的使用方式，很复杂。而AOP就是动态代理的规范化，把两种动态代理实现的方式的步骤都规定好了，让开发人员用一种统一的方式实现动态代理。使得在项目中大家都能简单快速读懂代码，提高开发效率

理解Aspect Orient ProgrammingAspects

Aspect：切面，给目标类增加的功能就是切面。就如前面复习的例子项目中用的日志和事务都算是切面。切面的特点，一　　般都是非业务方法，独立使用。没有该切面，也不会影响业务的实现

Orient：中文是面向、对象的意思

Programming：编程的意思

怎么理解切面编程：

1）、需要在项目分析功能时，找出哪些功能是切面（然后用方法实现）

2）、合理的安排切面的执行时间（在目标方法前执行，还是在目标方法后执行）

3）、合理的安排切面的执行位置，在哪个类中、哪个方法中增加增强功能

除了AOP，我们还见过OOP（Object Orient Programming，面向对象编程）

面向对象编程，就是开始进行项目开发的时候，要把项目中的所有功能分解出来，然后分析这些功能应该创建什么java类去实现，到最后就是用这些类的对象去实现所有的功能，完成项目的开发。这也需要遵从开闭原则（OCP）

二、切面编程的术语

1、Aspect：即切面，表示增强的功能，就是一堆代码一个方法，完成某一个功能。是非业务功能
2、JoinPoint：连接点，连接业务方法和切面的位置。就是类中的业务方法，例如在前面Handler类中执行doSome方法的invoke方法就是连接前面的日志切面和后面的事务切面的连接点
3、Pointcut：切入点，指的是对个连接点的集合。业务方法的集合
4、目标对象：哪个类的方法需要增加功能，这个类就是目标对象
5、Advice：通知，通知表示切面功能执行的时间，是在连接点前面执行还是在后面执行

说一个切面，一般包括三个要素：

1、切面的功能代码，即要干什么，代码的实现方法

2、切面的执行位置，使用Pointcut来表示执行位置

3、切面的执行时间，使用Advice表示

三、AOP的具体实现

AOP（面向切面编程）是一个规范，是动态代理的规范化，一个标准。所以需要我们实现它

AOP的技术实现框架：

1、Spring：Spring的内部实现了AOP规范，能做AOP工作，我们之前使用注解实现DI时所要用到的spring-aop子框架里面就实 现了AOP。Spring主要在事务处理时使用AOP。通常我们不使用Spring实现的AOP，原因是其使用起来比较复杂， 很笨重

2、aspectJ：这是一个开源的专门做AOP的框架。并且Spring集成了该框架，通过Spring就能使用aspectJ的功能。这才是我们 常用的。

aspectJ框架实现AOP有两种方式：

1、基于XML的配置文件方式（主要使用在全局配置事务方面）
2、使用注解的方式。项目中要做AOP功能，一般使用的是注解方式，aspectJ的注解有五个（都是Advice的注解）：