AOP叫做面向切面编程，可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。我们知道想日志功能、事务管理、权限控制等一些公共的行为，与很多的类具体核心业务功能是无关，但是这些操作又是必不可少的，如果在每个类中都进行编写，势必会导致大量的代码重复，所以我们将这些公共的行为封装成一个独立的模块，AOP所做的任务就是能够将模块中的公共的行为再次加入到业务逻辑中，当我们在进行数据库操作时，如果需要用到事务就是会发生程序流转的变化，转而去执行我们通知，这样我们真正的业务处理以及事务处理都能够进行到

AOP是基于代理设计模式实现的，当被代理者执行的某个方法之前、之后以及执行方法之时，还需要执行其他的操作。而在设计代理时，这些操作的代码是和业务代码分离的，如何在执行业务方法时执行这些操作呢？

首先，需要理解几个概念：切面(Aspect)、连接点(Joinpoint)、通知(Advice)、切入点(Pointcut)、目标对象(Target Object)。

① 切面：完成特定功能的类就是一个切面，在这个类中可以设置方法执行之前、之后以及执行前后要执行的具体的操作。

② 连接点：被代理对象中某个特定的方法，表明在执行该方法时，程序的顺序行会被打乱，转而去执行额外的操作。

③ 通知：切面中的方法，在目标方法执行之前执行的操作叫做前置通知。在目标方法执行之后执行的操作叫做后置通知。在目标方法执行前后执行的操作叫做环绕通知。不管目标方法是否执行，都会执行的操作叫做最终通知。在目标方法执行时，如果抛出异常就会执行异常通知。值得注意的是：后置通知和异常通知是互斥，两个通知只会执行一个。

④ 切入点：匹配连接点的断言。

⑤ 目标对象：真正调用方法的对象。

当一个对象执行时，先会与切入点表达式进行匹配，如果匹配成功，就会为这个类创建代理对象，代理对象执行时，就会执行切面中的通知。

(3) AOP的执行原理

当一个对象方法执行时，先会与切入点表达式进行匹配，如果匹配成功，就会为这个类创建代理对象，代理对象执行方法时就是执行切面中的通知。

20、AOP实现

（1）、配置切入点

（2）、配置通知

（3）、配置切面

21、AOP中出现问题：

（1）、通知中的参数有多个时，应将JoinPoint或ProceedingJoinPoint参数放在 第一位

（2）、配置文件中的返回参数应该与后置通知中的参数一致

22、通知的执行规则：

1.当异常通知执行时，后置通知将不会执行。异常通知和后置通知是互斥的。

2.最终通知无论如何都会被执行。

3.当目标方法执行时出现异常时，环绕通知的后半部分将不会被执行。

4.当环绕通知遇到后置通知时，如果后置通知想得到返回值，那么环绕通知必须添加return 将返回值返回

5.当多个环绕通知同时执行时，其执行顺序是嵌套结构

joinPoint.proceed() 作用有两个

1.如果有下一个通知，就执行下一个通知，

2.如果没有下一个通知 则执行目标方法。