​

**图解Spring AOP实现原理**

​编辑

**概述**

        研究了一段时间spring原理，对spring的ioc和aop有了更深刻的理解，最大的体会就是spring的核心就是ioc和aop，spring的功能都是基于这两大特性延展开的，spring ioc管理了Java bean，spring aop又可以切入Java方法的执行，这样一来spring实现了统一Java的江湖了，有了ioc就可以和其他框架进行很好的组合，有了aop,spring又可以管理数据库的事务了，可以说Java开发其实是基于spring框架的。

**一、相关技术**

  Spring AOP的实现通过**动态代理**实现的。Spring  AOP使用了两种动态代理，分别是**JDK的动态代理**，以及**CGLib的动态代理，**接下来将介绍一下这两种代理实现的原理。

* 1. **JDK动态代理**

JDK 动态代理是使用 Java 标准库中的 java.lang.reflect.Proxy 类来生成代理对象的机制。它要求目标类实现一个或多个接口，代理对象会实现这些接口，并将方法调用转发到 InvocationHandler 的实现类上。

创建一个接口，定义要被代理的方法。

创建一个实现 InvocationHandler 接口的类，实现 invoke 方法来实现方法的拦截和增强逻辑。

使用 Proxy.newProxyInstance() 方法创建代理对象，传入目标类的类加载器、接口数组和实现 InvocationHandler 的对象。

* 1. **CGLib的动态代理**

CGLIB（Code Generation Library）是一个字节码生成库，它可以在运行时动态生成类的子类，用于实现代理。CGLIB 动态代理要求目标类不是 final 类型，并且不需要实现接口。

导入 CGLIB 相关库。

创建一个继承目标类的子类，重写要被代理的方法，并在方法中添加拦截和增强逻辑。

使用 CGLIB 的 Enhancer 类创建代理对象。

**二、相关概念**

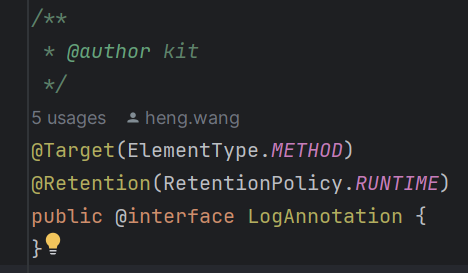
        AOP 的全称是 “Aspect Oriented Programming”，即**面向切面编程**。

* **连接点(Join point)**：能够被拦截的地方，Spring AOP 是基于动态代理的，所以是方法拦截的，每个成员方法都可以称之为连接点；
* **切点(Poincut)**：每个方法都可以称之为连接点，我们具体定位到某一个方法就成为切点；
* **增强/通知(Advice)**：表示添加到切点的一段逻辑代码，并定位连接点的方位信息，简单来说就定义了是干什么的，具体是在哪干；
* **织入(Weaving)**：将增强/通知添加到目标类的具体连接点上的过程；
* **引入/引介(Introduction)**：允许我们向现有的类添加新方法或属性，是一种特殊的增强；
* **切面(Aspect)**：切面由切点和增强/通知组成，它既包括了横切逻辑的定义、也包括了连接点的定义。

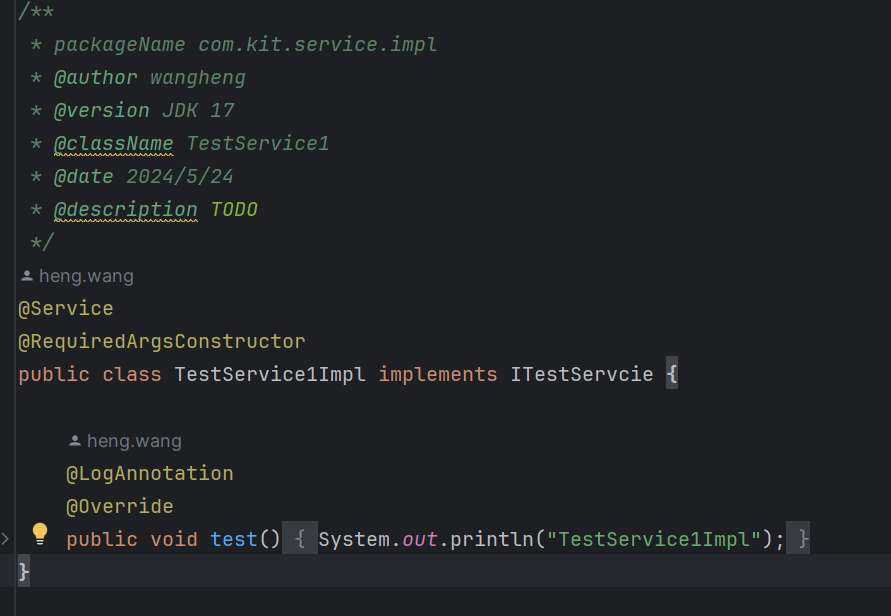
5 种通知的分类：

* **前置通知(Before Advice)**：在目标方法被调用前调用通知功能；
* **后置通知(After Advice)**：在目标方法被调用之后调用通知功能；
* **返回通知(After-returning)**：在目标方法成功执行之后调用通知功能；
* **异常通知(After-throwing)**：在目标方法抛出异常之后调用通知功能；
* **环绕通知(Around)**：把整个目标方法包裹起来，在被调用前和调用之后分别调用通知功能。

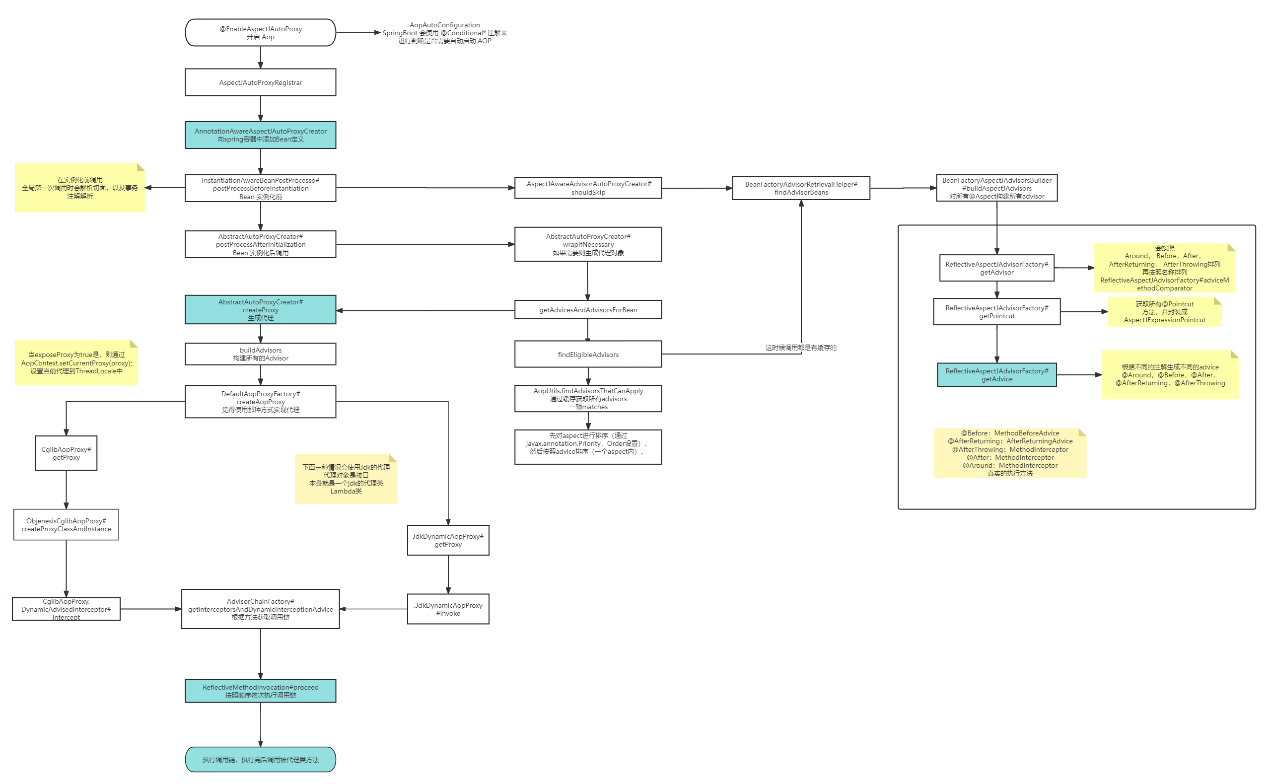
**三、简单示例**

****

****

****

**四、设计总览**



由上图可以看出

**总结**

        经过一段时间的写作之后，经常有一个想法在脑海里盘旋，怎么样写出和其他博主不一样的内容呢？经过反复思考和重新审视了自己之前推出的文章之后，发现之前的文章质量并不太高，于是我想到了图解的方式，就和自己在理解产品需求和做软件设计那样的方式把一项技术讲清楚，所以今后将会通过图解的方式把每一个主题讲清楚。

        本篇即是图解系列的第一篇，本篇为了讲清楚Spring AOP这个一主题，我通过相关技术、相关概念、基础示例、设计总览这四个方面展开讲解，以求达到让读者阅读之后既能够把Spring AOP应用起来，也能够理解到这一项技术的实现原理的目标。

**我是kit。本文已收录到**公众号：能源恒观**中**

​