## SpringAop是什么？从源码的角度分析

Aop(Aspect-Oriented Programming)面向切面编程，能够将那些与业务无关却又为业务模块所共同调用的逻辑或责任（例如事务处理、日志管理、权限控制等）封装起来，便于减少系统的重复代码，降低模块间的耦合度，并有利于未来的可拓展性和可维护性。

Aop通过在配置类上面增加@EnableAspectJAutoProxy注解来启用Aop,同时在切面类上面标记@Aspect注解,通过@PointCut声明切点,调用@Before,@After,@Around等声明调用时机。

从源码的角度上讲,SpringAop是基于动态代理的,每一个标记为@Aspect的都是一个Advisor，形成一个调用链,在源码中通过递归的方式重复调用Advisor的调用链来实现调用切入点的@Before,@After,@Around方法，通过Bean的后置处理器实现的,创建每个bean的时候调用初始化后的beanPostProcessor拿到所有的Advistors，循环跟当前的bean做对比，如果是@Aspect标记的bean就创建动态代理。

动态代理分为两种jdkProxy和cglib, 如果要代理的对象，实现了某个接口，那么Spring AOP会使用**JDK Proxy**，去创建代理对象，而对于没有实现接口的对象，就无法使用 JDK Proxy 去进行代理了，这时候Spring AOP会使用**Cglib** ，这时候Spring AOP会使用 **Cglib** 生成一个被代理对象的子类来作为代理.

## SpringAop的四种实现方式?

1. 经典的基于接口的配置
2. 纯POJO切面
3. @Aspect注解驱动的切面
4. 注入式@AspectJ切面

## 介绍下AspectJ和AOP的关系

1. SpringAop属于运行时增强，AspectJ属于编译时增强。
2. SpringAop基于代理(Proxying),而AspectJ基于字节码操作的，AspectJ应该算的上是java生态系统中最完善的Aop框架了。AspectJ相比于Spring Aop功能更加的强大，但是SpringAop相对于来说更加的简单。
3. 切面少Aop性能较高，切面多AspectJ性能高。
4. AspectJ新创建的文件都是以.aj结尾的文件，Aop创建实现的都是java文件。

## [@RestController](mailto:4.@RestController)与@Controller之间的区别