**CGLIB和Spring framework**

Spring框架支持两类主要的AOP方法：Spring代理和AspectJ[1]，两类方法都可以将aspect注入到已有代码中。

## Spring代理

每个Spring Bean都可以作为一个管理组件。如果用Java注解或者XML配置将一个类设定为Spring扫描对象，那么这个类将会被Spring框架实例化为可管理的单例bean。Spring使用@Autowired注解将这个单例bean注入到使用这个bean的其他bean中（很拗口）。这里被注入的Bean不是原始的类对象，而是一个将原始类对象封装的代理类。

默认情况下，Spring使用JDK动态代理库[2]创建新实例代理被注入对象的接口功能完成如上功能。如果被注入bean无接口，那么需要CGLIB[3]通过扩展类实现代理，CGLIB也支持JDK动态代理功能。

如果在被代理的方法中继续调用同一个对象的其他需代理方法，那么第二个方法调用将会是直接调用原始对象的方法。解决上述问题的方法是使用AspectJ[4]。

AspectJ扩展了Java编译器，在编译阶段解决了上述问题。因此这个方法和Spring没有必然关系。在对象内部AOP代码部分，其可以触发另一个AOP方法。在解决问题的同时，AspectJ也会引入新的问题。幸运的是，Spring代理和AspectJ可以共存。

1. <https://www.credera.com/blog/technology-insights/open-source-technology-insights/aspect-oriented-programming-in-spring-boot-part-2-spring-jdk-proxies-vs-cglib-vs-aspectj/>
2. <https://www.ibm.com/developerworks/java/library/j-jtp08305/index.html>
3. <https://dzone.com/articles/cglib-missing-manual>
4. <https://www.baeldung.com/spring-aop-vs-aspectj>