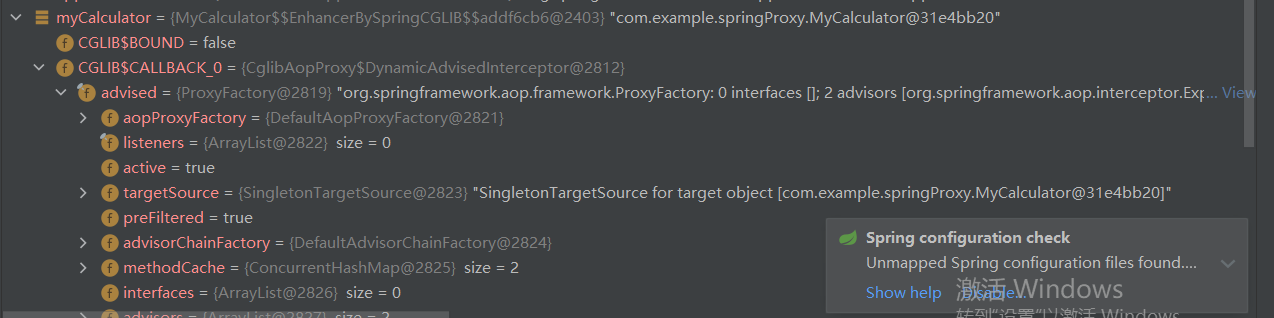
创建完成代理对象后，被代理对象虽然没放到一级缓存中，但是它被存储到CallBack中，后续进行方法拦截的时候还是可以访问到被代理对象的。

proxyBean.getName()

->proxyBean.callBack[0].inceptor(methodProxy,args)

->Object o = callBack.targetSource(); methodProxy.invoke(o,args);

每个CallBack都会存储proxyFactory

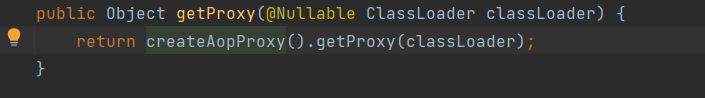


因此，被代理对象其实还是存储下来，并没有被释放。

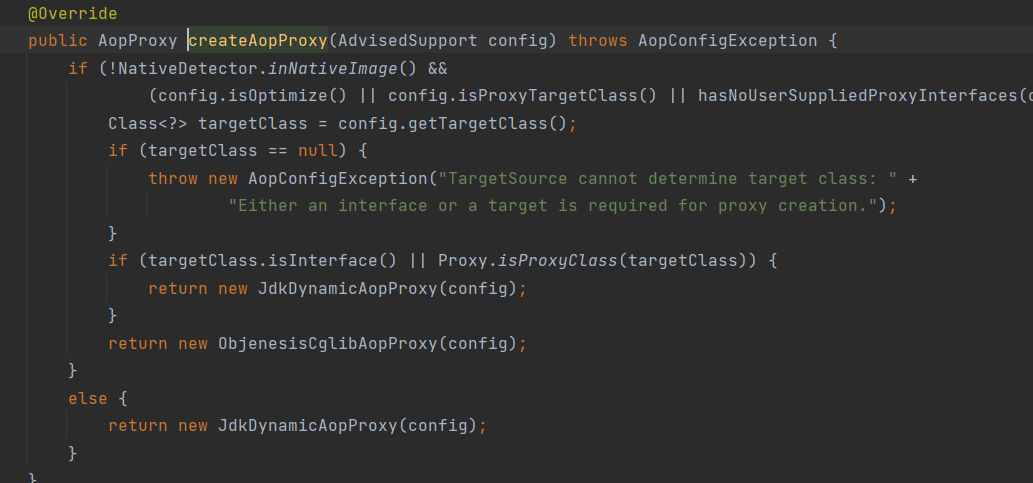
1. 创建代理对象
2. 执行方法

Advised->proxyFactory

1. AbstractAutoProxyCreator：createProxy，创建proxyFactory，将advisors、targetSource(被代理对象)加入proxyFactory
2. proxyFactory.getProxy(classLoader)：创建代理对象

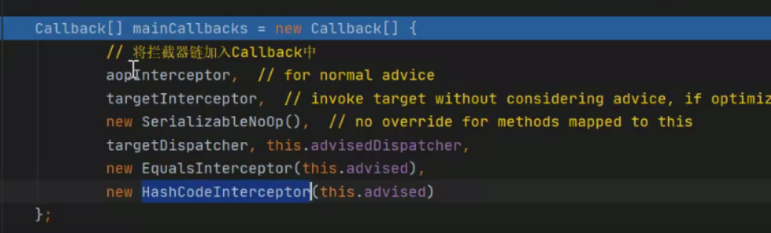


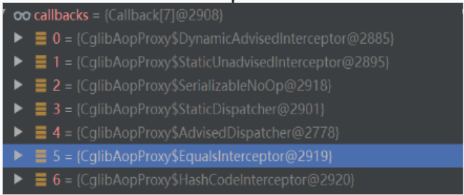
（1）createAopProxy()：通过cglib或者jdk代理，并设置相关属性（advisors）



1. getProxy(classLoader) 创建代理对象
2. getProxy创建代理对象
3. 创建Enhancer
4. 设置Enhancer的属性

* Superclass
* CallBack：根据Advisor（切点）设置CallBack（拦截器）



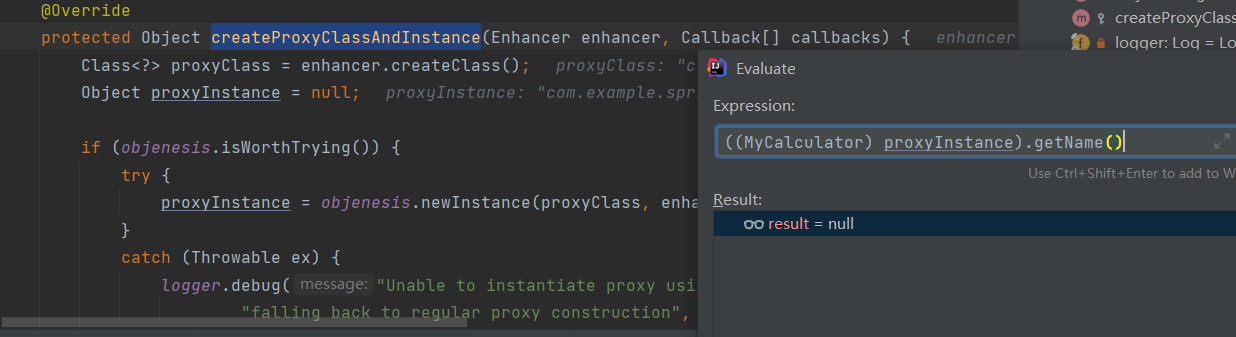


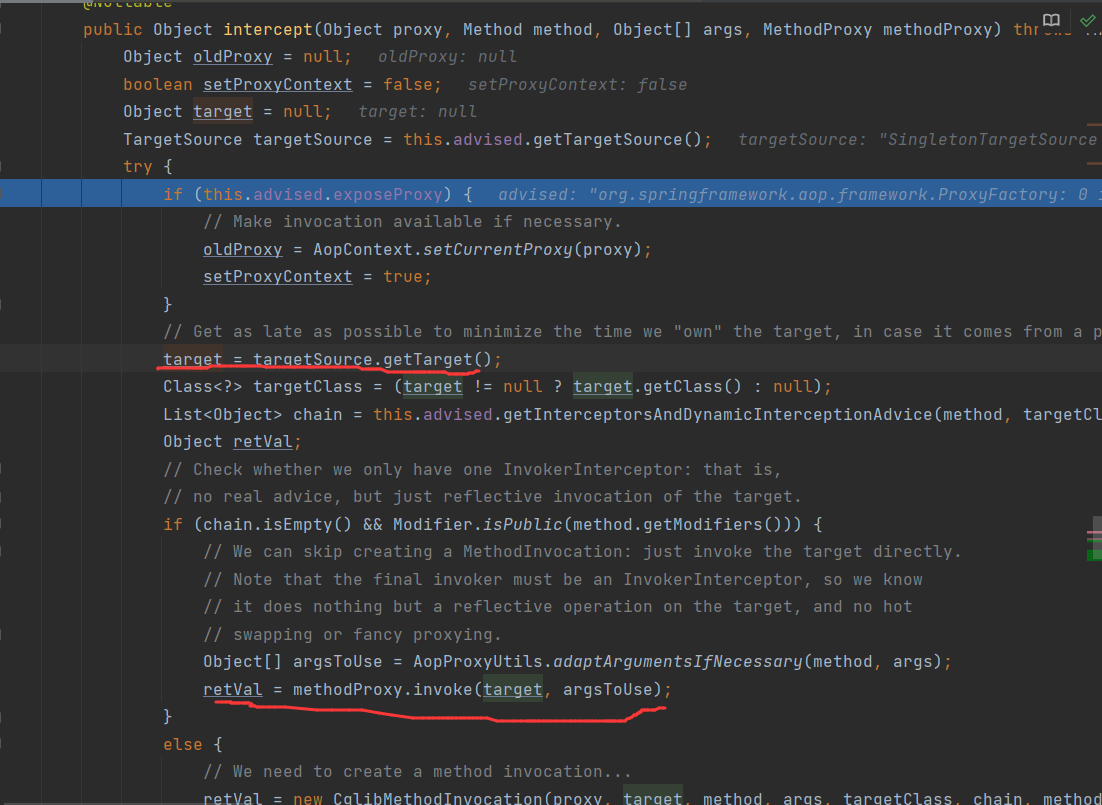
1. 通过Enhancer创建代理对象

由于代理对象继承自被代理对象，所以实例化代理对象的时候，会调用父类的构造方法，因此属性都会设置进去。

1. createProxyClassAndInstance

创建代理对象proxyInstance





代理对象被没有设置属性值，因此获取代理对象的属性值未null，不过通过get方法获取属性值，是可以获取到的，因为在CallBack里面会存储被代理对象，在调用get方法的时候，会执行到被代理对象的方法。