问答日志

小白-:20:44:22
面试的时候 CGLib 会考吗。
soso-李卓:20:44:43
看聊不聊呗
杨升玮-上海:20:44:44
你永远猜不到面试官想秀什么
小白-:20:45:01
哦哦哦
小白-:20:48:53
proxy 参数一般用在什么时候？
解答：

```
JDK
        InvocationHandler{
                target
                invoke(proxy,method,args){

                        //调用目标对象(反射的API)
                        method.invoke(target, args)
                }
        }

CGLib
        MethodInterceptor{

                intercept(proxy,method,args,proxyMethod){

                        //调用目标对象 (代理对象的API)
                        //代理对象又会调用目标对象
                        proxyMethod.invokeSuper(proxy,args)
                        //method.invoke(target,args)
                }
        }
```

Proxy 作为参数，代表被代理的对象，当执行代理对象的方法时传入（spring 做的）

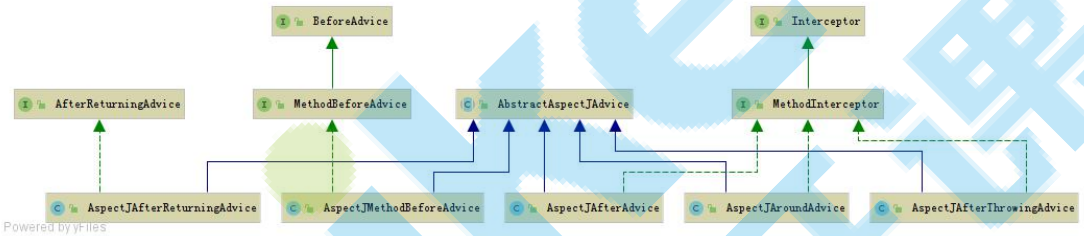
xxzx_4112410:20:53:54
这个地方有没有源码
soso-李卓:20:56:46
还是注解用的方便
soso-李卓:20:58:17
为啥呢，不方便管理？
xxzx_2655052:20:58:36
代码入侵很大吧
吕小布:20:59:44
注解不好阅读

soso-李卓:20:59:55
嗯 明白意思了，但也分场景
姚远:20:59:57
注解也是配置一次的吧～
RISE:21:00:11
springboot 都不需要配置，用一个注解就行
xxzx_2655052:21:00:22
这个见仁见智的，不用纠结
soso-李卓:21:00:25
如果我不是个纯通用的切面，还是可以考虑用注解
吕小布:21:00:56
这个底层都是一样就是表现成的区别
攀登:21:02:18
自定义标签
攀登:21:11:37
表达式
soso-李卓:21:11:50
匹配
攀登:21:12:59
切入点表达式完成类级别和发级别的解析
攀登:21:14:02
Advisor ——顾问
RISE:21:16:42
Advisor 是通知器吧？
解答：是的
切入点 Pointcut
通知 Advice
Advisor 是 Pointcut 和 Advice 的配置器，它包括 Pointcut 和 Advice，是将 Advice 注入程序中 Pointcut 位置的代码

杨升玮-上海:21:17:05
aspect = advise +poincut
马崇-java-上海-1 年:21:19:39
Advisor advice 都整混了
xxzx_4112410:21:21:35
aop 面向切面编程~

杨升玮-上海:21:30:54
我明白了 这 10 个对象 都是对应 xml 上的 标签
杨升玮-上海:21:31:08
5 个 是 通知
马崇-java-上海-1 年:21:31:32

还有一个是封装通知的
杨升玮-上海:21:31:36
aspectjadivcepoint 这个对象 等于 aop：aspect
杨升玮-上海:21:32:06
aspecjexpressionpointcut = point 的表达式
杨升玮-上海:21:33:08
ref 里的对象工厂 就是 产生 通知类的
杨升玮-上海:21:33:31
应该说是 增强类
杨升玮-上海:21:33:41
method 其实就是 增强类里的 方法
杨升玮-上海:21:33:46
就是 pointcut
跑丢一只鞋-北京-2 年:21:35:14
5 个通知 是 AspectjExp…… 的子类吗？
解答：不是




```
public interface BeforeAdvice extends Advice {

}
public interface Advice {

}
```

Ghost:21:36:22
反正就是根据一堆标签 生成了一堆对象
Ghost:21:36:34
大部分都是执行的时候调的吧
杨升玮-上海:21:36:39
跟 mybatis 一样的
杨升玮-上海:21:36:47
把 xml 解析成为 对象

攀登:21:42:04
业务层的目标对象？？？
解答：

是的，就是要被代理的对象

攀登:21:43:08
AspecrJAwareAdvisorAutoProxyCreator
soso-李卓:21:43:53
JDK
杨腾飞:21:45:44
proxy
杨升玮-上海:21:50:29
匿名内部类
soso-李卓:21:54:07
嗯 怎样把 目标方法包起来
Mr 陈:21:55:34
责任链设计模式
soso-李卓:21:55:47
调用完目标对象 还会调用 拦截器
soso-李卓:21:56:18
每一条通知
小白-:21:56:53
我一直以为是多个代理对象，嵌套执行。 看样子是一个集合进行循环之星
杨腾飞:21:57:29
methodInterceptor 应该实现了一个拥有 proceed 方法的接口
余爽:22:00:18
ObjectFactory

攀登:22:16:20
拦截器
soso-李卓:22:19:35
要根据 before after 这种类型判断 list 执行顺序？
解答：是的

soso-李卓:22:24:08
嗯 适配不同的类型
soso-李卓:22:25:08
插头
soso-李卓:22:27:12
嗯 还在说适配
Mr 陈:22:27:19
能
攀登:22:31:05
AOP 是 Spring 的水下的冰山呀
攀登:22:31:20
IoC 就是水上的冰山
soso-李卓:22:31:24
就是重载要单独子类处理 是吧
解答：对的
攀登:22:31:44
同一化
soso-李卓:22:32:48
哈 也关心是怎样验证的
soso-李卓:22:33:34
是，要判断类型吧？ beftoe after
Mr 陈:22:35:19
有点像 冒泡排序
Mr 陈:22:35:34
冒泡排序的感觉
Mr 陈:22:37:06
责任链模式
soso-李卓:22:38:56
通知的代码
soso-李卓:22:39:30
增强的代码
soso-李卓:22:43:07
多个 advice 呢？
soso-李卓:22:43:15
多个 befroe
余爽:22:43:25
before 类似队列，after 类似栈
跑丢一只鞋-北京-2 年:22:43:49

几个 befor 都是一样的处理吧
杨腾飞:22:44:10
递归了
跑丢一只鞋-北京-2 年:22:44:22
不同的目标对象 有不同的 befor
Mr 陈:22:44:30
安卓的 okhttp 源码 请求网络返回 也是这么写的 特别像 简直一样的
soso-李卓:22:44:33
刚刚 哪里 就一个 advice.before
soso-李卓:22:44:45
外面套的 是吧
跑丢一只鞋-北京-2 年:22:45:28
配置文件里 配了几个 before 就有几个吧
Mr 陈:22:46:17
看不见了呀
强-上海:22:46:20
这是利用 JVM 中方法属于栈结构做的吧
杨腾飞:22:51:05
环绕呢
乔之恋:22:51:38
around
杨腾飞:22:51:48
middleware
黄海-成都:22:51:48
这个整个 aop 就是和 ioc 那里 getbean 关联起来的么？
解答：正解

跑丢一只鞋-北京-2 年:22:58:04
List 里面的顺序 是按照配置文件配置的顺序的吧，实际增强的顺序 就是 是 after 的 就是按照 finally 的方式处理 是这样的