AOP：编程范式（面向切面编程）不是编程语言，解决特定问题（代码重复，关注点分离（水平分离，垂直分离，切面分离（分离功能性需求和非功能性需求））），补充oop

好处：集中处理某一关注点/横切逻辑，方便添加和删除关注点，侵入性少，增强代码可读和可维护性

spring aop 使用方式两大类，xml配置和注解方式，都表现切面有什么，切面的表达式和切面的内容

aop关于切面的注解

@Aspect：声明切面类

@PointCut声明切入目标，Pointcut express（切面表达式）

@Advice：声明植入时机

Pointcut express：designators（指示器）匹配java类的方法 wildcards：通配符 operators：运算符

wildcards：\*匹配任意数量的字符+匹配指定类及其子类 **.**.一般用于匹配任意数的子包或参数

operators：&&与||或！非

designators：匹配方法execution()

匹配注解 @target() @args() @within() @anntation()

匹配包/类型within()

匹配对象this() bean() target()

匹配参数args()

匹配包/类型

//匹配ProductService类里的所有方法

@Pointcut(“within(com.service. ProductService)”)

public void matchType(){}

//匹配com.service包及子包的所有类的方法

@Pointcut(“within(com.service. .\*)”)

public void matchPackage(){}

匹配对象

//匹配aop对象目标对象为指定类型的方法，即DemoDao的aop代理对象的方法，一般用//在动态代理的方法

@Pointcut(“this(com.service. DemoDao)”)

public void thisDemo(){}

//匹配实现IDao接口的目标对象（不是aop代理的对象）的方法这里指DemoDao的方法

@Pointcut(“target(com.service. IDao)”)

public void targetDemo(){}

//匹配所有以Service结尾的bean里的方法

@Pointcut(“this(\*Service)”)

public void beanDemo(){}

匹配参数

//匹配任何以find开头而且只有一个Long参数的方法

// execution除了要定义参数类型还要定义方法名称，返回值类型和修饰符等

@Pointcut(“execution(\*\*..find\*(Long))”)

public void argsDemo1(){}

//匹配任何以find开头而且第一个为Long参数的方法

@Pointcut(“execution(\*\*..find\*(Long,..))”)

public void argsDemo2(){}

//匹配任何只有一个Long参数的方法

@Pointcut(“args(Long)”)

public void argsDemo3(){}

//匹配任何第一个参数为Long的方法

@Pointcut(“args(Long,..)”)

public void argsDemo3(){}

匹配注解

//匹配方法标有AdminOnly的注解的方法

@Pointcut(“@annotation(com.Adcdfd)”)

public void annoDemo(){}

//匹配标注有ProductService的类底下的方法，要求annotation的RetentionPolicy级别为CLASS

@Pointcut(“@within(com.service. ProductService)”)

public void annoWithinDemo(){}

//匹配标注有Repository的类底下的方法，要求annotation的RetentionPolicy级别为RUNTIME

@Pointcut(“@target(com.service. Repository)”)

public void annoTargetDemo(){}

//匹配传入参数标有Repository注解的方法

@Pointcut(“@args(com.service. Repository)”)

public void annoArgsDemo(){}

execution（）有？的可以省略

格式execution(

modifier-pattern?//修饰符

ret-type-pattern//返回值

declaring-type- pattern? //描述包名

name- pattern (param- pattern) //描述方法名（描述方法参数）

throws- pattern（匹配抛出异常）

)

advice注解

1、@Before 前置通知

2、@After（finally）后置通知，方法执行后

3、@AfterReturning 返回通知，成功执行后

4、@ArterThrowing 异常通知，抛出异常后

5、@Around 环绕通知

spring aop实现原理

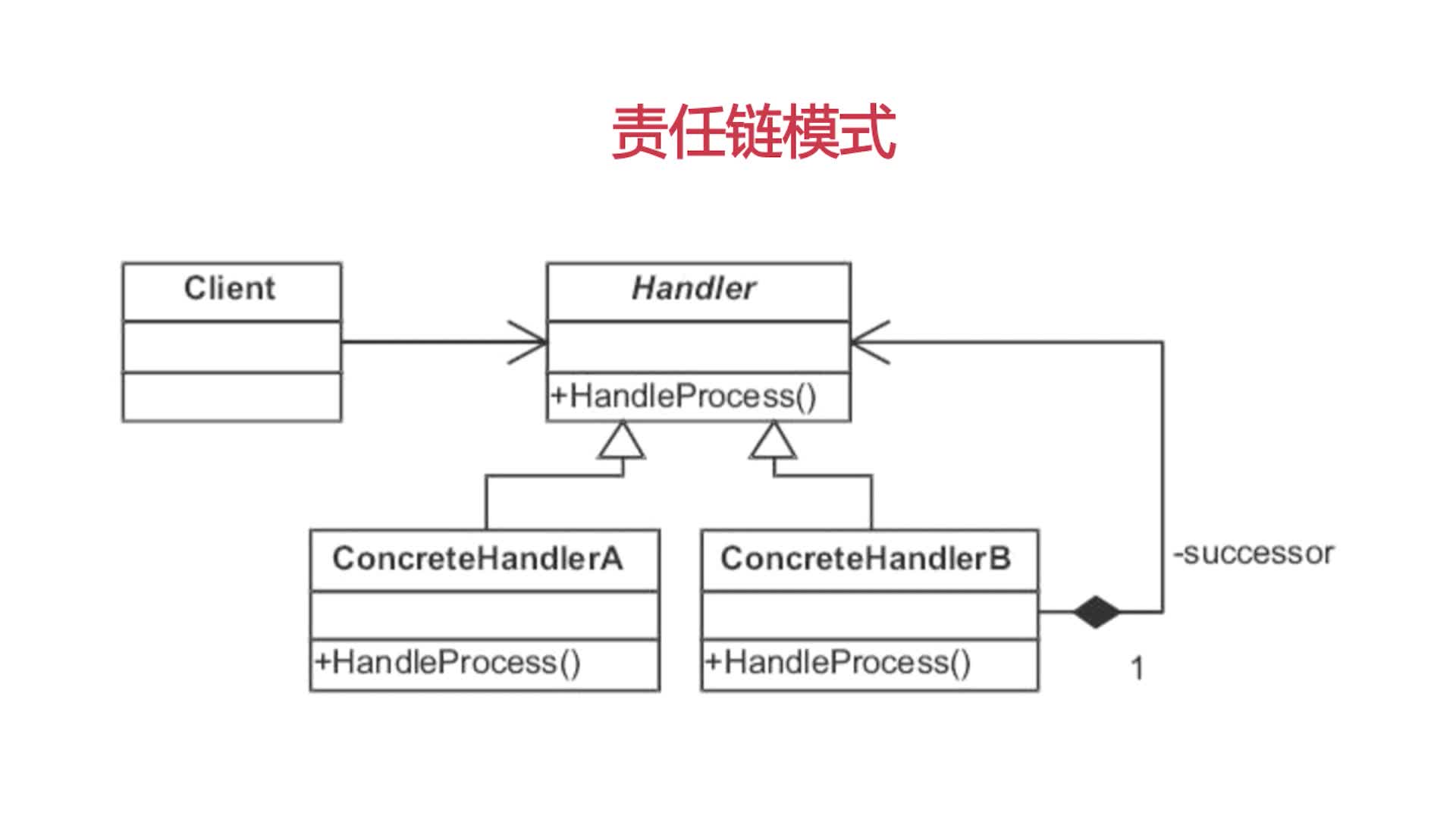
如果目标对象实现接口，默认jdk动态代理，没有实现接口，成功；ib代理，实现接口但强制cglib代理使用cglib代理

@SpringBootApplication

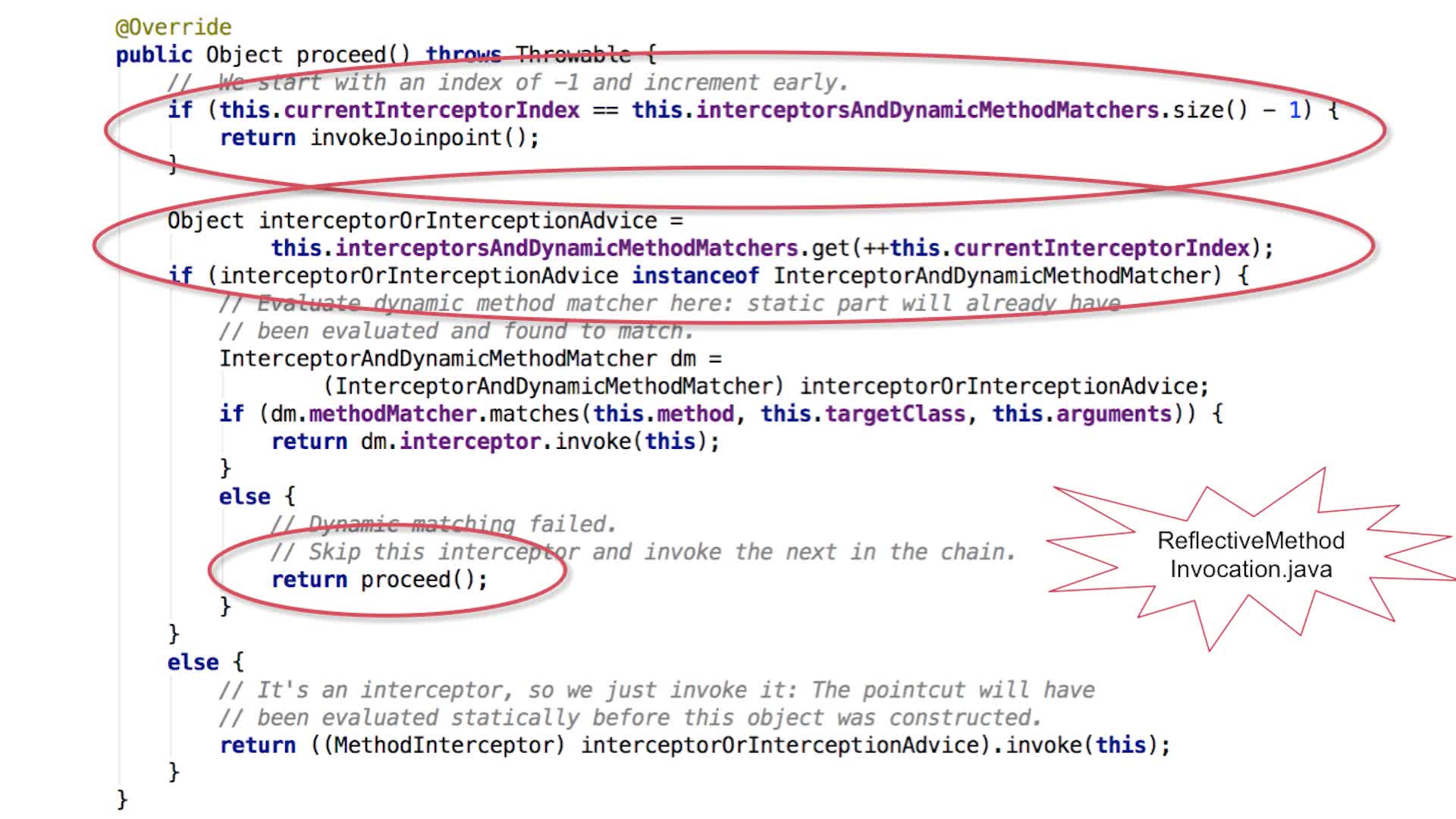
//强制使用cglib代理

@EnableAspectJAutoProxy(proxyTargetClass=true)

aop如何链式调用



spring责任链源码



spring aop应用源码解析

@Transactional事物控制

@PreAuthorize安全

@Cacheable缓存

